Photo Reconstruction and Object Classification with OpenCV[1] and PointNet[2]

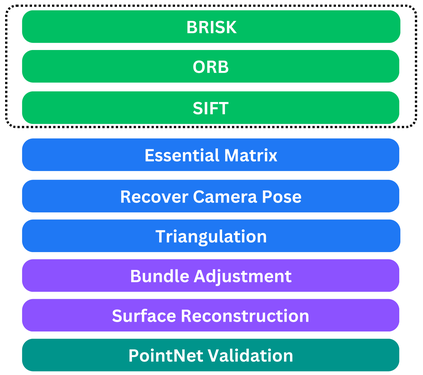
Tim Nguyen

BOSTON LATIN SCHOOL

1 Background

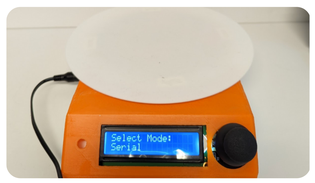
Images in their purest form can only provide information in a two-dimensional space, which lacks the information that can be found in the three-dimensional space of the real world. Information such as the cross-sectional shape of each part of an object and the volumetric properties of the object is lost, and these factors are often a method used to help classify complex objects, such as fossils and manufacturing defects. To leverage the versatility that images provide and transform them into a medium with more detail, photogrammetry is a popular method to do the task. With programs like COLMAP[3] and Meshroom[4], the user simply puts in the images of an object and the program tries its best to produce a 3D version of the object using keypoint detection. However, existing programs often require a lot of computational power, and with a push toward edge computing, the need for a lightweight photogrammetry package has never been greater.

2 Design



3 Arduino Rigs

- 3D printed housing and powered by Arduino Unos
- Can communicate to computers via Serial Com
- Used to help accelerate the data collection process

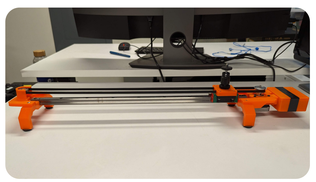


Turntable[5]

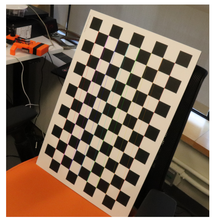
- Has a rotating plate for small objects
- Used to capture all angles
- Camera mounted on an external tripod

Linear Rail[6]

- One meter-long rail
- Camera mount slides across the entire rail
- Used for larger objects (ex. Sofas)



4 Camera Calibration



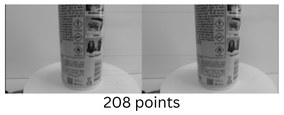
- Extracts camera matrix (3x3) and distortion coefficients
- Camera matrix includes *x* and *y* focal lengths and optical centers
- Distortion coefficients are used to counteract lens distortion

5 Keypoint Detectors

SIFT[7]

- Uses Gaussian Blur[8]
- Oldest and slowest
- Usually extracts most

1657 points



208 points

ORB[9]

- Checks nearby pixels[10] (FAST)
- Runs BRIEF Descriptors
- Relatively fast

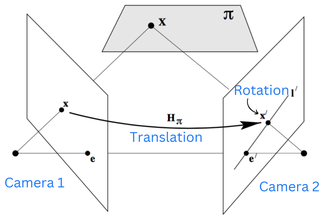
BRISK[11]

- Concentric Rings check[12]
- Intensity Descriptors, similar to ORB
- Relatively fast

547 points

6 Essential Matrix & Recovering Camera Pose

- Essential Matrix: 3x3 matrix that forms epipolar lines

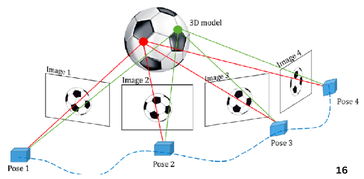


Epipolar lines form a triangle[13]

- Camera Rotation matrix & Translation Vectors recovered
- Coordinates of points in 3D space are also recovered
- 3D points are used to make an initial estimate of point cloud

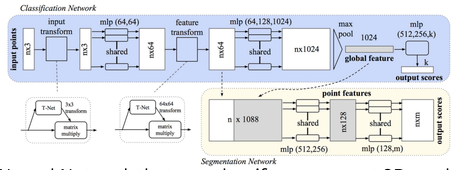
7 Bundle Adjustment & Surface Reconstruction

- BA: Correcting camera pose by reprojecting 3D points to 2D space
- An optimization problem



- Outliers are rejected & faces are created using Poisson[14] or alpha surface reconstruction with Open3D[15]

8 PointNet

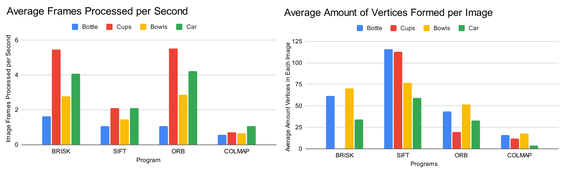


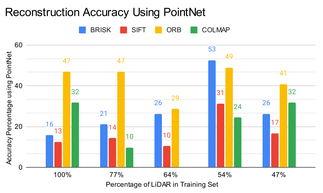
- Neural Network that can classify or segment 3D models
- The model is trained on ModelNet40[17] and LiDAR datasets
- Used Tensorflow to build and train the model
- PointNet is a quantifiable way to measure quality

9 Data & Results

- Data is obtained by taking pictures or videos of objects
  - Came from a variety of phones and cameras
  - A group of volunteers assisted with data collection
- Benchmarks were performed on a 12-core CPU, 30-core GPU, 32GB RAM ARM-based laptop
- Running COLMAP Sparse as it is the closest equivalent
  - COLMAP Dense reconstruction provides color at the expense of time (> 4 hours)

Perfomance Tests

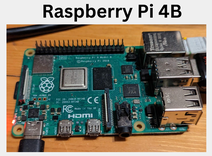




Performance Test on Can w/ 100 images

	RPi 4B (8GB)	Laptop
BRISK	0:14:39	0:01:09
ORB	0:28:09	0:03:52
SIFT	2:53:53	0:17:10

Raspberry Pi 4B



Specs[18]:

- 1.5GHz Quad Core ARM processor
- 8GB RAM
- Size of a credit card

Sample Outputs

Sample Image

ORB with Alpha Recon

SIFT with Alpha Recon

BRISK with Alpha Recon

COLMAP Sparse

COLMAP Dense

*BRISK failed to run the cups dataset

10 Discussion

- PointNet may not be robust against noise
  - LiDAR data is noisier than ModelNet40
  - Correlation between more LiDAR and accuracy rate
  - One of the best tools that can quantify mesh quality
- COLMAP uses SIFT for keypoint detection
  - Explains the similarity in speed
  - However, COLMAP uses significantly more resources, with upwards of 25GBs of RAM used during sparse reconstruction
  - Dense reconstruction can take hours and requires dedicated GPUs
- The package can successfully run on low-powered hardware
- The package provides solid reconstruction faster than existing implementations
  - With accuracy at times even better than current methods

11 Conclusions

- ORB and BRISK have more efficient ways of keypoint extraction
  - Faster than SIFT & COLMAP
- ORB is more reliable than BRISK as it is able to go through all datasets during testing
  - Vertex count seems to be close to each other
- ORB provided mesh that was relatively close to the original mesh
  - On average, the highest accuracy out of all models
- COLMAP's speed can be contributed to additional computations
- Runs on Raspberry Pi successfully
  - It does take longer, which is expected for the power
  - Opens the possibility for edge computing uses

12 Future Directions

- Would like to have time to do more extensive testing on mobile hardware
  - Preliminary tests on a Raspberry Pi 4B have shown the same quality of reconstruction at a slower pace
- Use a different implementation of Bundle Adjustment
  - GTSAM bundle adjusts with landmarks and factor graphs
- Implement into real-world applications
  - Interesting to see this project in fossil categorization and defect detection
- Using a different validation tool
  - Perhaps exploring 3D anomaly segmentation

References

1. https://opencv.org/
2. https://stanford.edu/~rqi/pointnet/
3. https://colmap.github.io/
4. https://github.com/alicevision/Meshroom
5. https://hackaday.io/project/168301-arduino-controlled-photogrammetry-3d-scanner
6. https://learn.adafruit.com/bluetooth-motorized-camera-slider
7. https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf
8. https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html
9. https://arxiv.org/pdf/1710.02726.pdf
10. https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html
11. https://ieeexplore.ieee.org/document/6126542
12. https://gilscvblog.com/2013/11/08/a-tutorial-on-binary-descriptors-part-4-the-brisk-descriptor/
13. https://imkaywu.github.io/blog/2017/06/fundamental-matrix/
14. https://www.cs.jhu.edu/~misha/MyPapers/SGP06.pdf
15. http://www.open3d.org/
16. https://www.researchgate.net/publication/338638478/figure/fig7/AS:867955038490633@1583948015581/Bundle-adjustment-illustrative-example.png
17. https://modelnet.cs.princeton.edu/
18. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/