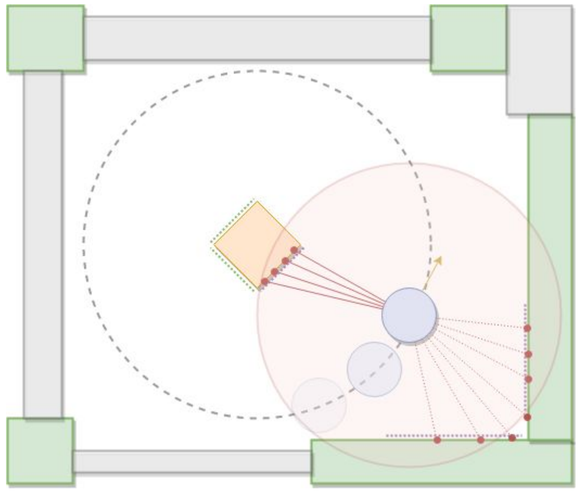
# SCAR

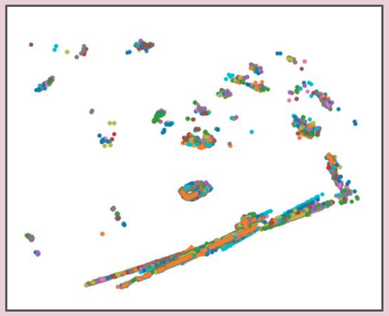
Ben Ziemann, Jamie Cho



The ***SCAR*** (**SC**anning **A**nd **R**econstruction) project is an attempt to use the Neato as a mobile scanner by driving around an object and using the LIDAR and camera to recreate it digitally. We are treating this as a simpler, though still well within scope, implementation of a SLAM algorithm, operating with the constraint of an isolated and deterministic environment.

## I. Introduction

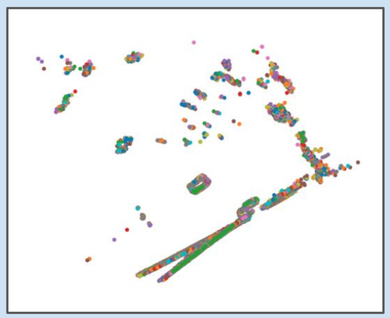
Going into this project we figured that the LIDAR and odometry alone would not be enough to accurately create a map and we were right:



Odometry and physical robot drift, catching on obstacles, and lag between sensor updates meant uneven scans that, while still able to sort of make out what they are, are largely unrecognizable.

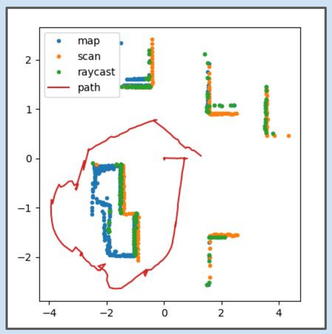
## II. Methods

Our pipeline primary targetted rectifying such errors and drift over time. In the implementation, we employed the ICP (iterative closest points) algorithm, which was best suited for this task, as our operating conditions were aligned with its key assumptions of locally close corresponding points.



Accordingly, we matched the incoming points with a built up map and computes the pose correction based on the found ransforms; the improvements were immediately apparent over the plotting of raw points.

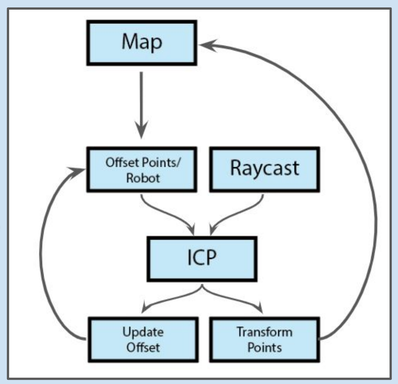
Problems persisted, however, as ICP did not always match the scan and map correctly. Overtime these small changes resulted in a kind of “padding” around the actual object to build up. To counteract this we implemented a form of raycasting. Drawing rays out from the robot’s estimated position, we could effectively filter any points located behind other points, as such points could not have been visible in the sensor’s perceptive field.



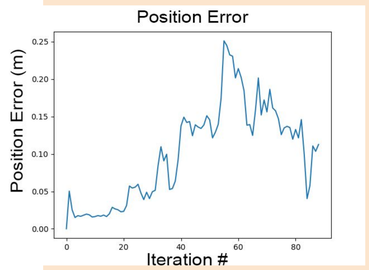

The implementation of the map queries also helps to moderate itself: points are only visualized and used in ICP if the probability that point actually exists exceeds a threshold; This probability is increased in the log-odds representation as the observation accumulates over time; likewise, the probability is decremented if is not seen when it should be in range and in sight of the robot.

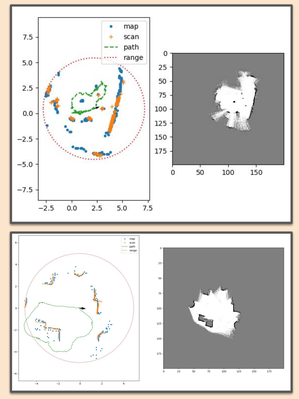
## III. Workflow

Our architecture consisted of a dataflow which essentially took the scans and gradually found the best representation and the map and the robot’s pose simultaneously. Our dataflow builds upon itself as time goes on, using a built up map to more confidently make transformations on points currently under analysis. After an initial map consisting of the first several scans with the essentially stationary robot is generated, raycasting and ICP begin to be performed. Raycasting limits points already on the map to physically viable points based on the robot’s positon; ICP then takes that reduced map and fits the current scan points to it. The resulting transformation is applied to the current scan points, which are in turn added to the ever growing map. This rectifying transformation is also used to update known map-to-odom coordinate offsets, giving us a headstart on correcting the robot’s position estimate and the scan point locations in the next iteration of the cycle.




## IV. Results



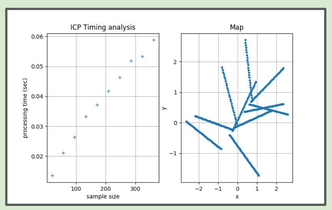



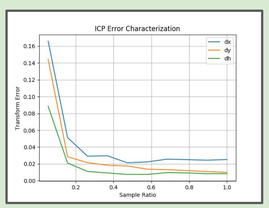

We have validated the performance of our algorithm, and report the results accordingly; as seen, the pose rectification is such that the drift is controlled within a margin of approximately .3m, both in Gazebo simulator (right top) and on the physical Neato (right bottom).

In our experiments, we also observed the constraint that the environment be somewhat feature rich so that ICP can correctly orient the scan. This limits our projects usage as a “scanner” and instead more delegates it to mapping a given area.

## V. Analysis

To build intuition on the characteristics of the algorithm, we ran a number of rudimentary benchmarking experiments on the algorithm with varying input conditions.




On the left, we have the timing analysis with respect to the input size on a randomly generated map; as seen, we find that the ICP processing time is approximately linear to the number of input correspondences. Furthermore, we find that the ICP performance is mostly invariant to the ratio of sample-query overlap above a threshold of approximately 30%.

## VI. Conclusion

Based on a fairly straightforward architecture, we achieved compelling performance on the abstraction of a 2D reconstruction algorithm as a a minimalistic SLAM problem; the algorithm produces relatively high-fidelity maps of the environment in a repeatable manner, both in gazebo-simulated environments and on a real Neato XV11/BotVac platforms.

Natural next steps from here include detecting and resolving loop-closures, graph optimization, and local-map tracking.