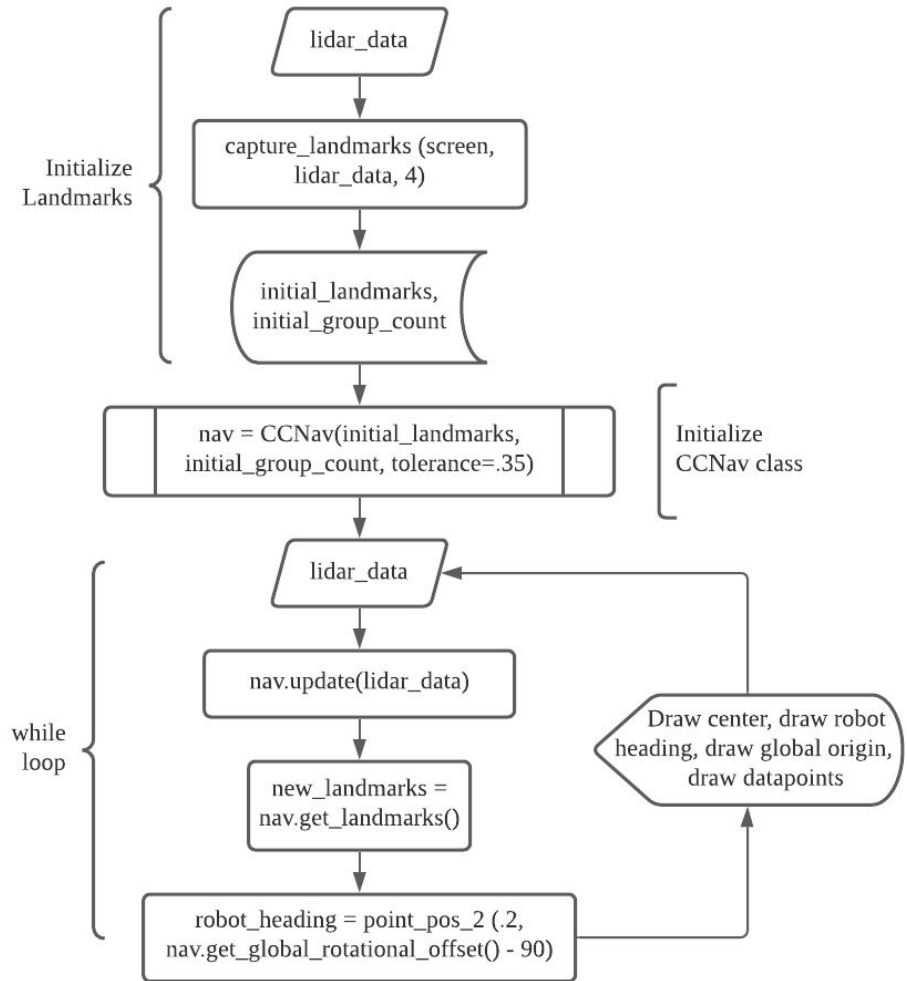# CCNav

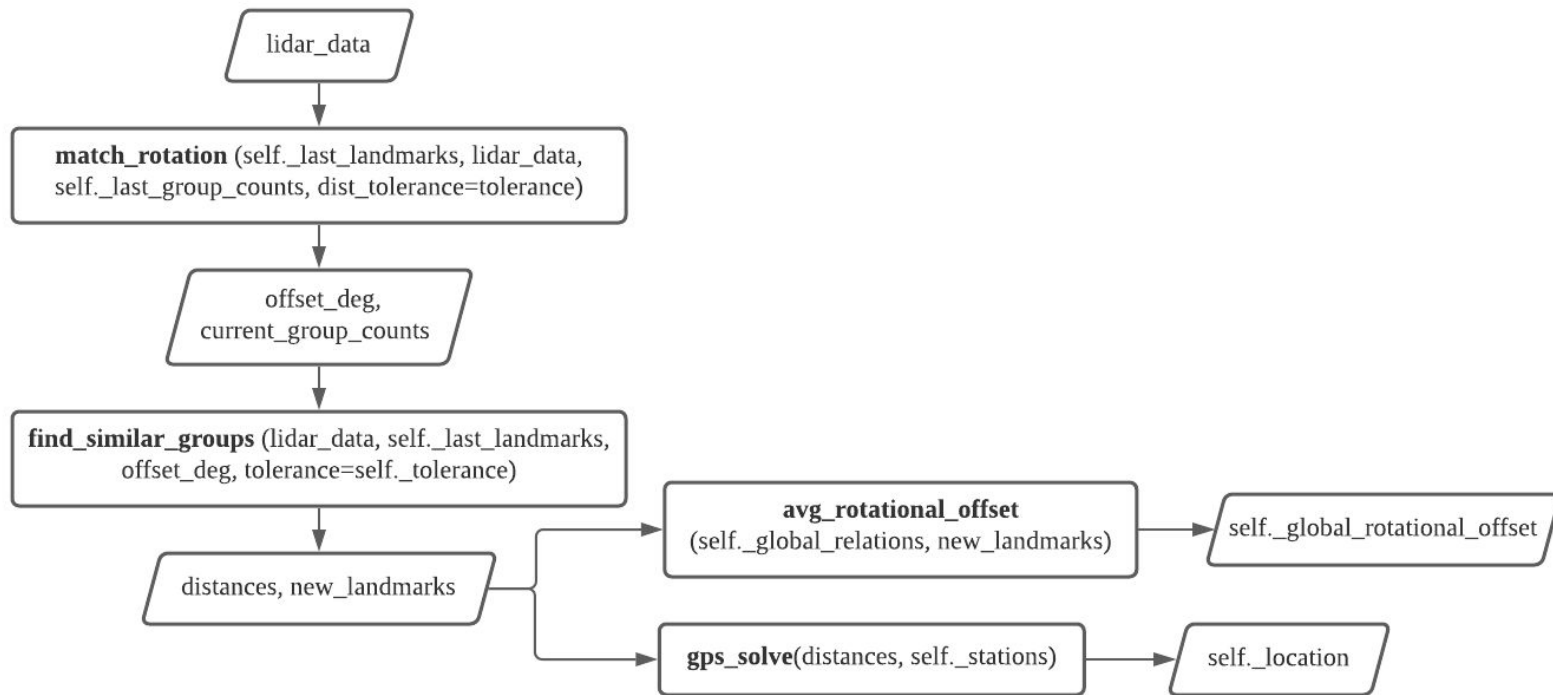A navigation method based on multiple preset landmarks

Real-life Setup

Four landmarks are placed to form a rectangle around the robot. The landmarks should be non-transparent, and thin enough.

Flowchart for the navigation process

Structure of CCNav.update( )

lidar_data

match_rotation (self._last_landmarks, lidar_data, self._last_group_counts, dist_tolerance=tolerance)

offset_deg, current_group_counts

find_similar_groups (lidar_data, self._last_landmarks, offset_deg, tolerance=self._tolerance)

distances, new_landmarks

avg_rotational_offset (self._global_relations, new_landmarks)

self._global_rotational_offset

gps_solve(distances, self._stations)

self._location

Why CCNav?

- **User-friendly**: doesn't need to run on a Linux computer

- **Efficient**: doesn't require high computing power

- **Accurate**: no accumulative error, lidar data provides more accuracy than Odom

- **Flexibility with the environment**: can work in places without any feature points