The algorithm I designed is simple, the feelforce script subscribes to scanner data to detect ranges of obstacles, processes this data, and then publishes this processed data to the turn and move scripts which react appropriately. More specifically, the feelforce script takes ranges of the scanned data representing directions relative to the turtlebot, and takes the minimum of the distances scanned within each range (left, right, and front) and publishes these three minimums to the turn and move scripts. The turn script has an established threshold for each direction published to it from feelforce, if any of the thresholds are broken, the turn script will initiate either a left or right turn at maximum angular velocity. The left or right turn is determined randomly (with a left turn bias to limit progress inhibition) by the move script. The move script has the same directional thresholds established as the turn script, and when a threshold is broken, the robot will stop, generate a random number associated with left or right, and publish that to the turn script, the turn script will then actuate a turn until the robot is situated such that no thresholds are being broken. This algorithm essentially means that the robot will constantly seek space, and will choose to navigate to that space with a bias toward left navigation, and a bias against spaces with small openings due to the thresholds established. Due to the randomness introduced with the navigation direction, as well as the latency of turn commands depending on publishing of the feelforce, the robot exhibits sufficient robustness to overcome most geometry offered by the different poses provided.

There are multiple other algorithms I could have chosen. The first Idea that came to mind was essentially wall following. The robot would move forward until hitting a wall, and then turn so that it's right flank would always be close to the wall, leading it slowly along the walls to its destination. This algorithm works, however I did not like the thought of handling some of the implementation details such as overcoming turns that would create tight spaces for the robot, or turns around corners that would snag the robot and determining how to react. Much better to seek out open space in my opinion. The next Idea that I came up with was to introduce a random oscillation into the robot's movement rather than relying so heavily on stopped turns. This idea also worked for the most part, and was quite interesting to work with, however it struggled with a particular pose and could not be overcome without significant modification so the plan was scrapped. In the end none of these algorithms really provides much in the way of intelligence for the robot, however I found myself enjoying tuning the thresholds and the turn biases to create paths that would never be repeated from one test to another.