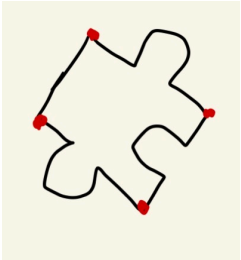
Goal

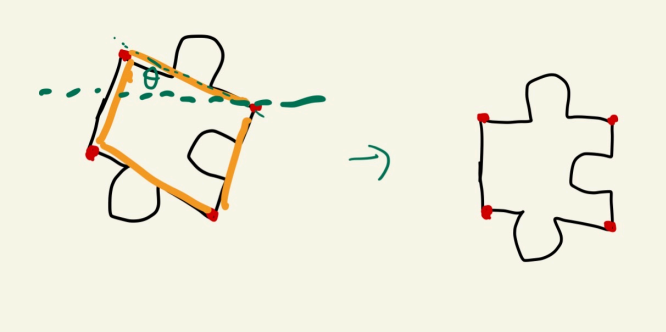
- Our goal for this machine project was to utilize the OpenCV Library to solve a jigsaw puzzle(in the sense that the computer will tell you where the pieces should go). Some assumptions we made is that the input would be separate images of all the puzzle pieces, with the pieces themselves being white and the background being black. We also made an assumption that the images would be taken under the same distance away from the camera.

Methods:

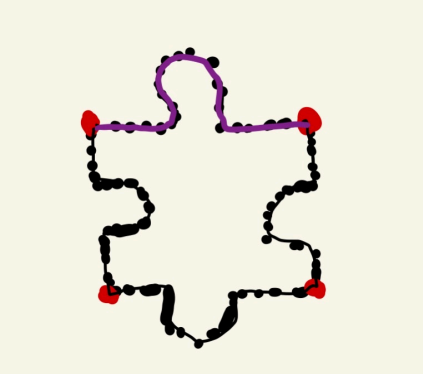
1. We would first turn the images of the puzzle pieces into black and white for optimal separation of puzzle piece from background.
2. Use Harris Corner detection from OpenCV library to get candidates for 4 corners.
3. Do some math to develop some algorithms to select the 4 corners by maximizing the “rectangle-ness” and area.



4. Utilize the “rectangle” we found previously to compute some rotational matrix and rotate all puzzle pieces to be horizontal/vertical



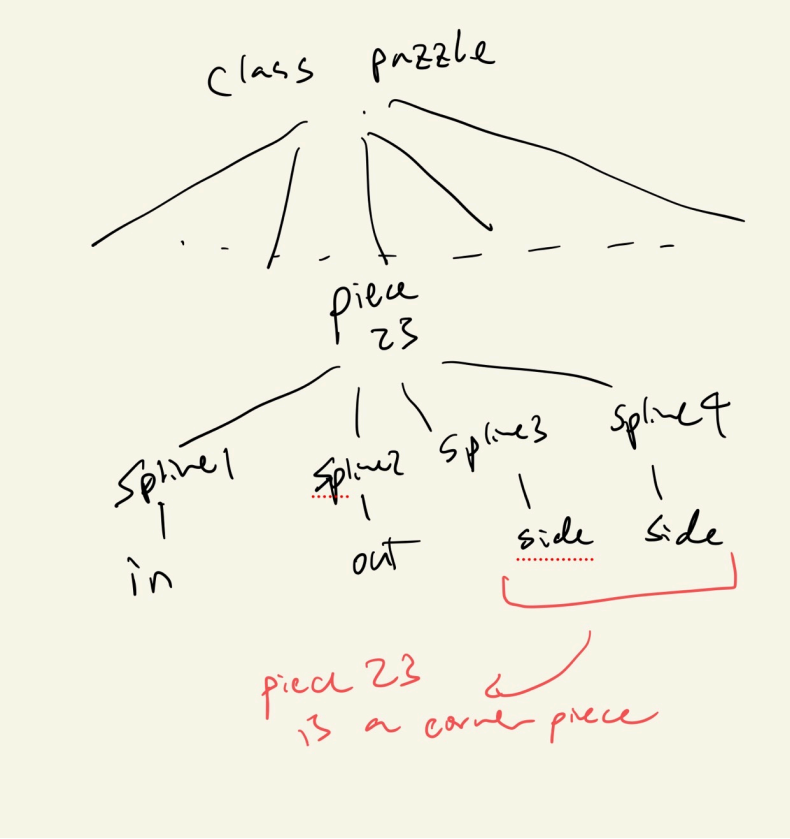
5. Use Canny edge detection to get edges
6. Parse edges to 4 splines by performing shortest path algorithms between 2 selected corners as the “start” and “end” nodes. We turn every edge point into a node in a graph, and connect the neighboring nodes together by adding edges between the nodes to form a complete graph.



7. We then take all locations of edge points in the puzzle graph and average them to get a center point (x,y). Then we compute the average distance of the points on one spline to the average distance of the points on a straight line on the same side as the spline. If the distance of the spline to the center is greater than the distance of the line to the spline, then we categorize that specific spline as an “outward” spline. Otherwise categorize the specific spline as an “inward” spline. However, if the distance is very close or near zero, that means we found a “side” pieces, which is those pieces on the very outer edges of the puzzle.

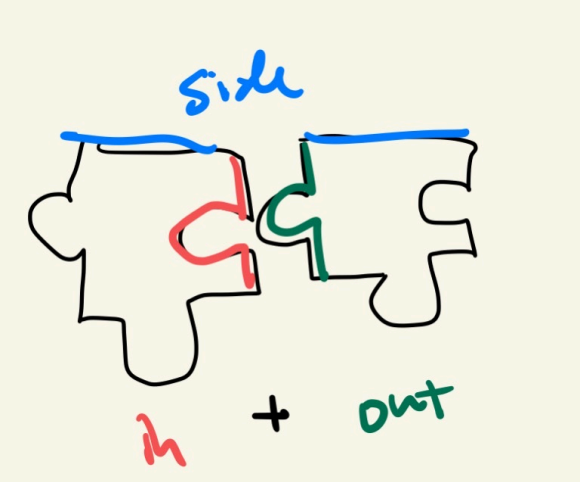
8. The code architecture looks somewhat like this:

   Have a class called puzzle pieces where the individual puzzle pieces are the objects that live in this class. Then each puzzle piece objects have 4 splines as the attributes in order of clockwise, with each spline a dictionary labeling them as either “outward”,“inward”, or “side”.

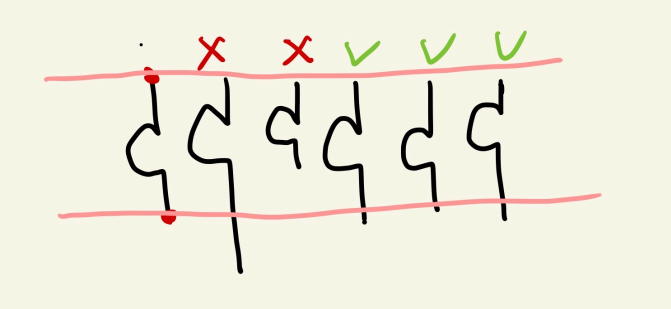


9. Iterate over every puzzle piece object and find all pieces that has 2 splines labeled as “side”. That should come to 4 pieces and they would be our corner pieces.

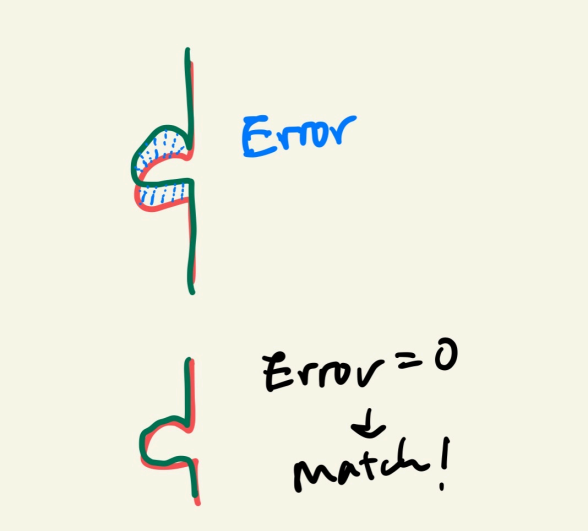
10. Pick one of the splines from a corner piece(that is not an edge!) to initiate the spline-matching process. First we see if the target spline is “inward” or “outward”. Know that only the “inward” splines have an opportunity to match with the “outward” splines. “Side” splines are unmatchable because they go on the outside.



11. Once we are left with only “outward” or “inward” splines, we then overlay 2 splines by generating a rotation matrix by comparing the start and end point of the two splines. Once two splines are in their correct orientation, we then compute the euclidean distance of the start and end point of the target spline. Then compare it with every other spline. Set some tunable threshold to filter out splines that don’t match in length.



12. Overlay two splines by matching the start and end points of the splines to the best of our ability and then make both splines have the same points
13. Calculate the accumulated error over matching points and find the least error spline–Assume we found a match!



14. Build a tree, but we didn't have enough time to think through how this would work.

Design Decision

One of the decisions that we needed to make is what algorithm we should use to robustly parse the puzzle piece into 4 edges to perform spline matching. At first we tried categorizing each point based on their proximity to the closest corners. But there are irregular points in the "concave in" splines that would get categorized as the wrong splines. We then considered using a method where we connect adjacent corners to create 4 quadrants. Then categorize the points based on which quadrant they lie in. However, this method is also not robust enough. When we were looking through all our test pieces, we found that this method doesn't work well on some pieces where there are very "concaved in" splines. We ended up using a shortest path algorithm to parse the edges. However, this method can become pretty computationally expensive when having a big puzzle to solve. One of the method that we didn't try but thought of after is that if some points are miscategorized, we can implement additional checks to compute the closest neighbor points and categorize those point in the same splines as their neighbor. This is only to help with those points in the "concave in" which proved to be tricky to deal with.

Challenges

We definitely underestimated how complex this project is. A lot of the problem comes with that we only realize that there are more steps to be done in the previous

steps when we get to a specific step. There were also challenges with scheduling and a teammate getting really sick in the middle phase of the project. Another teammate doesn't have extensive experiences with coding, therefore she feels like a lot of the time she understands what to do conceptually, but really struggles in the implementation stage. She also suffered from burning out/ lack of motivation when her teammate fell ill.

Future Improvement

We weren't able to finish the coding, in fact, we stopped at a point earlier than we would like to. We would really like to have a code that actually works.

Lessons

Ariel: I would definitely say that I would scope the project more suitable to my actual coding experiences. I feel like this is a recurring theme in this class where I just feel like I conceptually know what I am supposed to do but I have trouble implementing it by code. I also would like to ask for more help and do more communication. Sometimes I feel so lost I just don't really go to any office hours because I didn't really know what I was struggling with therefore I cannot ask for help effectively. I also feel like office hours are usually used as more of a conceptual/high level clarification. I also noticed a pattern in bigger projects for myself that I suffer major loss of motivation after the project goes stagnant/encounters some major pain points. I have found that I perform a lot better when I have a general skeleton code structure and knowing exactly what I have to do like in the localization project.