Assignment 1

שם: מוחמד אבו עביד **ת.ז.:** 206924888

שם: שהאב מלכאוי **ת.ז.:** 206987729

A. Algorithm:

For each puzzle:

1. Get the dimension for the final puzzle image and load all image pieces into the "pieces_list"

2. Warp the first piece into the right place in the puzzle image (perspective transform given)

3. Iterate on puzzle pieces until all are used, or reached the iterations limit:

3.1. Point matching:

given 2 images, the puzzle (in its current state, initially has the first piece in its right place) and the puzzle piece we are currently iterating, find features in each image.

Get matches:

For each feature in the puzzle piece, compare it with all the features found in the puzzle (as we saw in the lectures, by finding the 2 closest vectors and checking the ratio with a given threshold). If the feature passes the ratio test, we keep it and its match (it has a match feature in the puzzle).

3.2. Now, we have a set of matching features (2 points entries). We calculate the transformation matrix (Affine or Homography), and if there are not enough points (3 or 4 points) or we couldn't find a good enough transformation matrix we skip the puzzle piece

If there are enough points, for a given iterations number: (ransac)

choose (3 or 4) random matching points (from the points set), calculate the transformation matrix (cv2 lib), and check if it's good enough by counting inliers. If the inliers count reaches a certain threshold we stop, otherwise we keep doing the same process until we reach the iteration limit, and return the best matrix we found that has a minimum number of inliers. Count the piece if we used it

*Inlier point: calculated using residual (how close the point is after mapping from the matching point)

3.3. Now, we have the transformation matrix, we wrap the puzzle piece using that matrix (cv2 lib)

3.4. add the wrapped result to the current puzzle solution we have:

Since both have the same size, and the wrapped result might contain areas where it overlaps with the puzzle result we have thus far, we add the pixels from the wrapped result to the puzzle where the puzzle has missing pixels, that way, the areas where the 2 images overlap we don't add and keep that area as it is in the puzzle (that way there will be no areas where the intensity is very high due to adding the same value to the pixels more than once because of overlapping areas)

4. after merging the 2 images and finishing iterating on all the pieces, if we still have unused pieces and we didn't reach the iteration limit, do the process again on the unused pieces, otherwise we get the final puzzle result and save it

B.

*Used grayscale image:

Finding features to use for matching was done after converting the original image to grayscale, this allowed faster run times as advised

*Thresholds:

***Used for ratio test:**

*ratio, ratio test threshold:

-Depending on the puzzle, the puzzles that we saw had no problem finding matches we lowered the ratio to get very accurate ones. The easier it is to find matches the lower the ratio (until a certain point), that was the case also on pieces where matches were significantly hard to find, lowering the ratio, in that case, discards a lot of false positives, that may lead to a completely inaccurate match (sometimes such matches added a complete background in the puzzle result). Sometimes it leads to using fewer pieces but the pieces are used correctly and no obvious wrong piece is used.

-the puzzles where there is a need for more points and no fear of negative-positive matches we kept the ratio relatively high compared to the cases mentioned above and most of the accuracy is handled while finding the transformation matrix

***Used in finding the transformation matrix:**

*residuals threshold

*minimum inliers, minimum inliers needed to take the matrix as a candidate

*satisfying inliers, if reached a satisfying percentage/number of inliers we stop looking for other matrices

-tuning these thresholds is used for accuracy, and depends on how many matching points are we expected to get(not in numbers, just an estimate of whether will we or won't have enough points to discard/take as inliers), in some, puzzled where we allowed a higher number of matching points (higher ratio), we requested for more close inliers (high min-inliers, low residual) for example, that way we are more likely to add the puzzle piece in its right place accurately.

-these thresholds helped to prevent wrong warps, especially by setting the min-inliers higher, with a low residual. If we get a transform matrix that satisfies these demands we can be more confident that wrong warps won't happen since such warp would "throw" a lot of points in the wrong place, thus demanding high inliers count all the time eliminates such warps.

-from trying to tune the ratios, it was vital to not overdo it. Having stricter thresholds than necessary may not only lead to more unused pieces, but sometimes if we are left with too few points we get a warping matrix that is very specific to these points, and because they are few they also pass the ransac process (in most cases it was because we end up with very few points that are condensed/very close to each other)

<u>c.</u>

most results were ok, where we have the full picture (sometimes with cracks, or a missing piece due to not finding enough matches to apply the assembly algorithm).

*Missing piece

-Suggested algorithm: instead of iterating over the pieces in file order, we can at every given iteration look for the closest piece to our current puzzle result (the piece that would give the most matches with the current threshold). And with each full iteration over all the pieces, if we still have unused pieces, we can make softer thresholds (for example depending on how many unused pieces are left). That way we might use more pieces but not necessarily at the expense of quality because the pieces where there are good matches we still demand strict thresholds that give us better warping matrices, therefore better assembling for these pieces. Finally, for the pieces that are left unused we can gradually (at a certain point) lower our demands on the thresholds and use them even if they are not put in the best way possible, it still might add more information to the final result.

-if we reach the minimal threshold and still we didn't do any changes to the puzzle result (still not enough matches/good matrix), maybe we can do the matching between the unused pieces themselves to get a "puzzle result from unused pieces" and in each change in that image, we do the matching between it and the puzzle result we have. In case combining the unused pieces results in info that may give new matches

*<u>Example: Homography puzzle 10</u>

-it was very hard to finding matching features and if we put a high threshold, we do find matches but either they are wrong matches or not enough matches/good matches to calculate a good transform matrix, and in most cases if we end up finding a matrix the warping is wrong (it warps the piece mostly as a background)