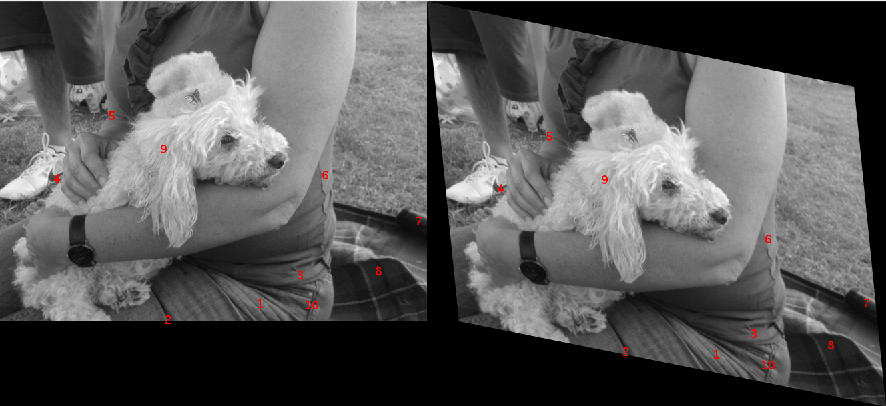
**EX3 – Explanations**

1. The following images presents an image and the same image after tansformation:



The transformation matrix used is the following:

H = [1 .2 0; .1 1 0; 0.5 0.2 1]

1. The best 10 matches ordered by match ratio:

379.33, 437.83 -> 423.39, 513.94

241.54, 461.97 -> 287.57, 510.16

440.09, 394.79 -> 480.3, 482.8

75.39, 250.4 -> 100.41, 265.37

158.35, 156.01 -> 174.26, 187.07

476.96, 245.2 -> 501.38, 340.81

616.78, 313.85 -> 647.94, 436.97

557.03, 388.95 -> 594.79, 500.29

235.3, 205.58 -> 255.94, 252.44

450.63, 440.35 -> 494.52, 530.4

3. In the next image, you can see the image transformed by the given transformation (left), and the image transformed by the calculated transformation (right):



The calculated image looks almost the same like the computed one, but clearly that there is a difference. If the DLT had better matches, it could have produces a more accurate result.

5. In this exercise, we used the RANSAC function in order to get better results from the DLT algorithm. We gave the RANSAC all our matches, and he filtered only the good matches ("inliers") using the distance function we implemented in order to decide which match is an inlier or outlier. We also gave it a parameter to compare the distance function against, which we decided was a good one after some trials. Also, we gave it a function in order to check if the sampled matches are collinear in order to perform better, and loop boundaries to prevent from the inner loop to run forever.

An example of RANSAC not improving the DLT algorithm, is presented in Example1.m. Basically, we executed the script and saw that transforming the original image using the transformation calculated using DLT and transforming the original image using the RANSAC produces 2 images that look the same. We also used the ComputeError function from question 4 and noticed that both errors are practically the same. We used the same image and the same transformation from question 1. The reason RANSAC didn't improve the result is due to the high value of the parameter that is used in the distance calculation. The given parameter causes the RANSAC function to consider all matches as inliers, so it's doing the same thing the DLT algorithm does. All other specific information can be found in Exanple2.m. (Including code explanations and figures)

An example of RANSAC improving the DLT algorithm, is presented in Example2.m. Basically, we executed the script and saw that transforming the original image using the transformation calculated using DLT and transforming the original image using the RANSAC produces 2 images that look almost the same, being that the one produces by the RANSAC looks somewhat more like the one produced on question 1 more than the other. We also used the ComputeError function from question 4 and noticed that the error of the RANSAC function is smaller as absolute, and that the division of the RANSAC error by the DLT error is smaller than 1. We used the same image and the same transformation from question 1. The reason RANSAC improve the result is due to the sweet spot of the value of the parameter that is used in the distance calculation. The given parameter causes the RANSAC function to consider the right amount of matches as inliers, so that the DLT using those Inliers can produce a much more accurate result. It is important to notice that the RANSAC is a probability algorithm, so it improves the result most of the time, but might not prove the result sometimes. This is due the parameter deciding the maximum iterations the algorithm does: The higher the value is, the more chance that we will get an improvement, but the longer the algorithm might take on order to finish. All other specific information can be found in Exanple2.m. (Including code explanations and figures)