COMP 431 Assignment3
Fatma Güçlü 110110082

#Assignment3 Visual Product Search

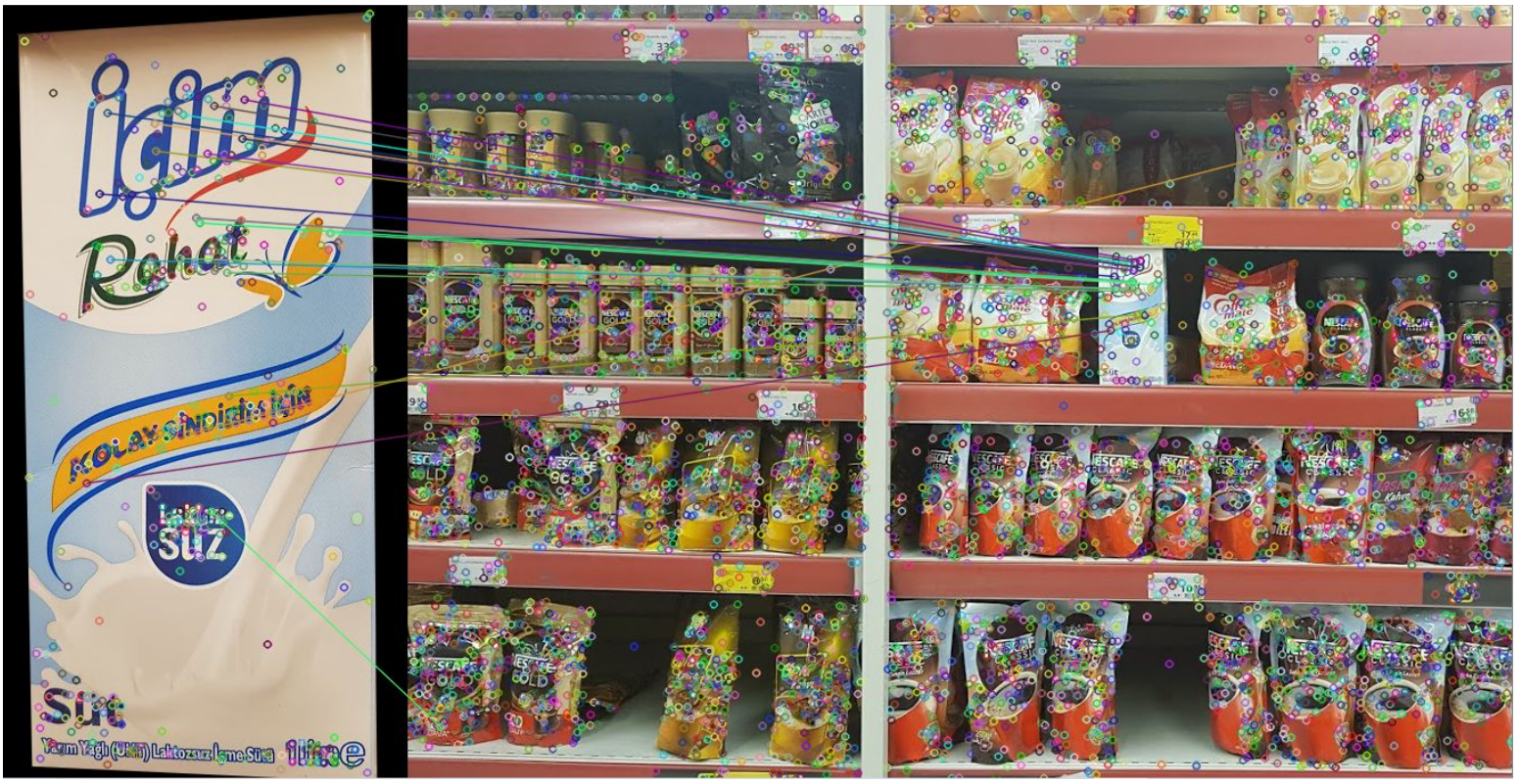
In this search, I used scale-invariant feature transform (SIFT) for feature detection and feature extraction. After that, I used Fast Library for Approximate Nearest Neighbours (FLANN) for feature matching. Then, I eliminated the bad matches by using Lowe's ratio test. Finally, I made a homography transformation and plotted all the steps for all products. At the end, I explained my experimental results, and challenges.

SUT

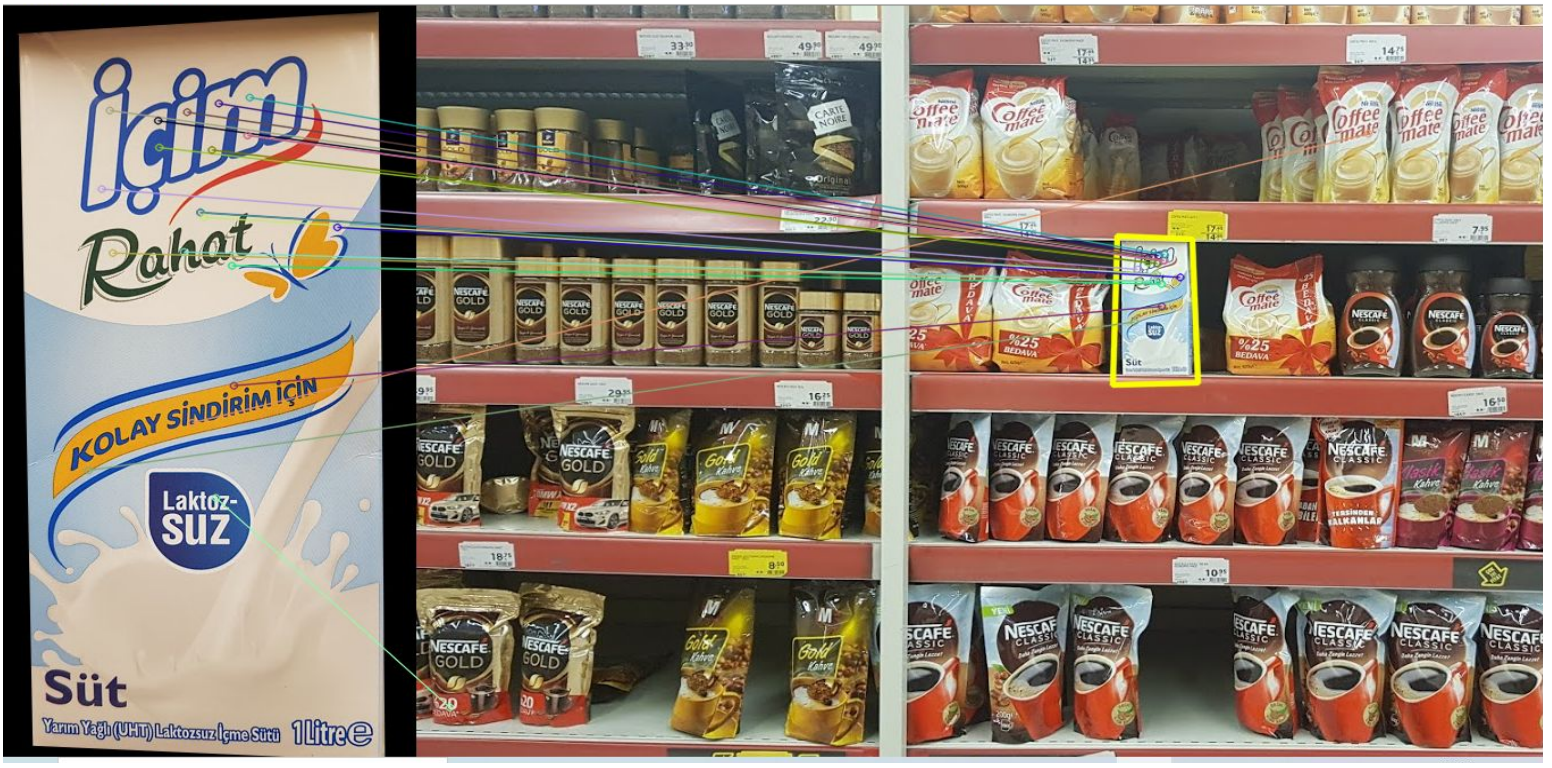
- Feature detection

- Feature matching

- Feature matching elimination



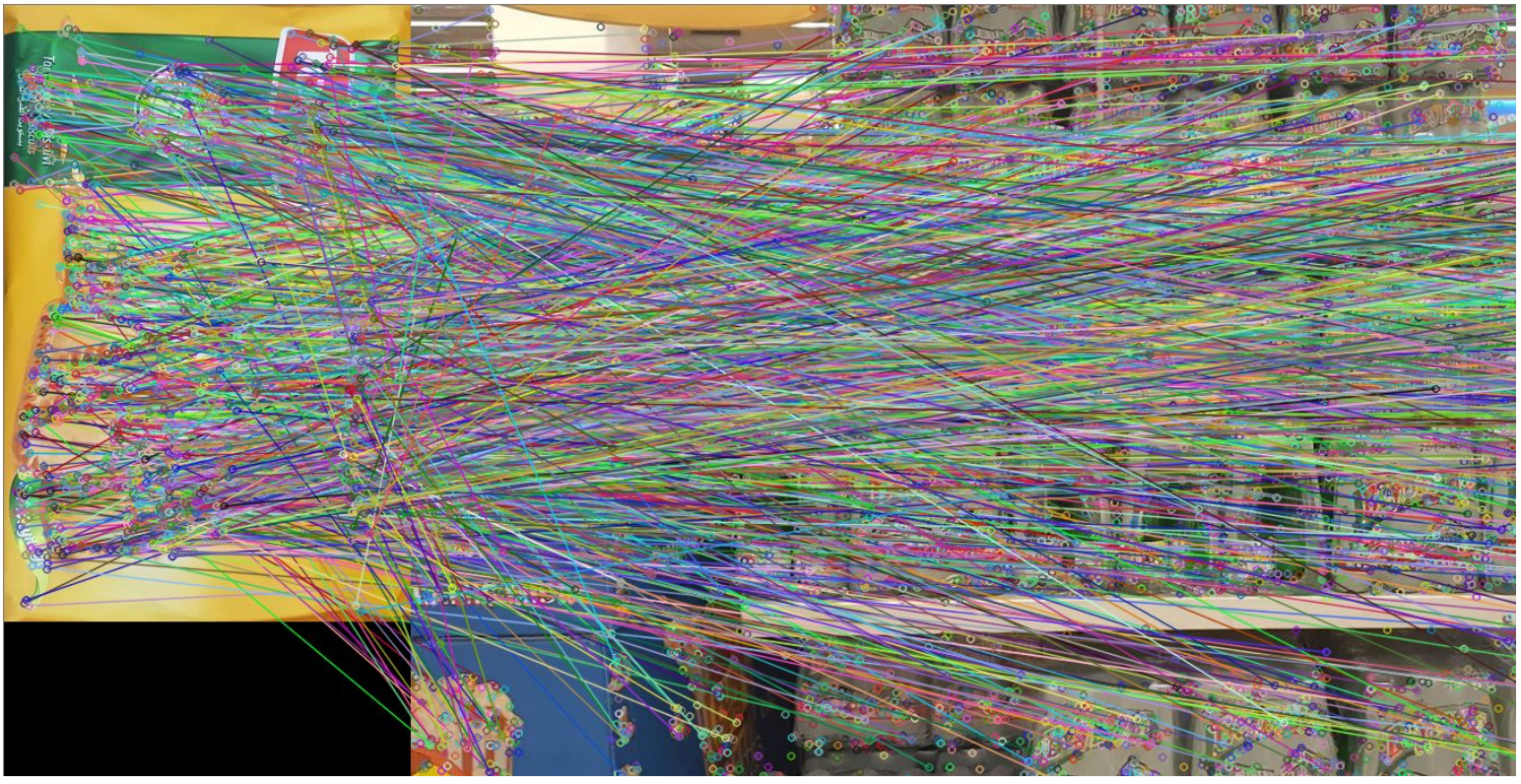
- Homography transformation



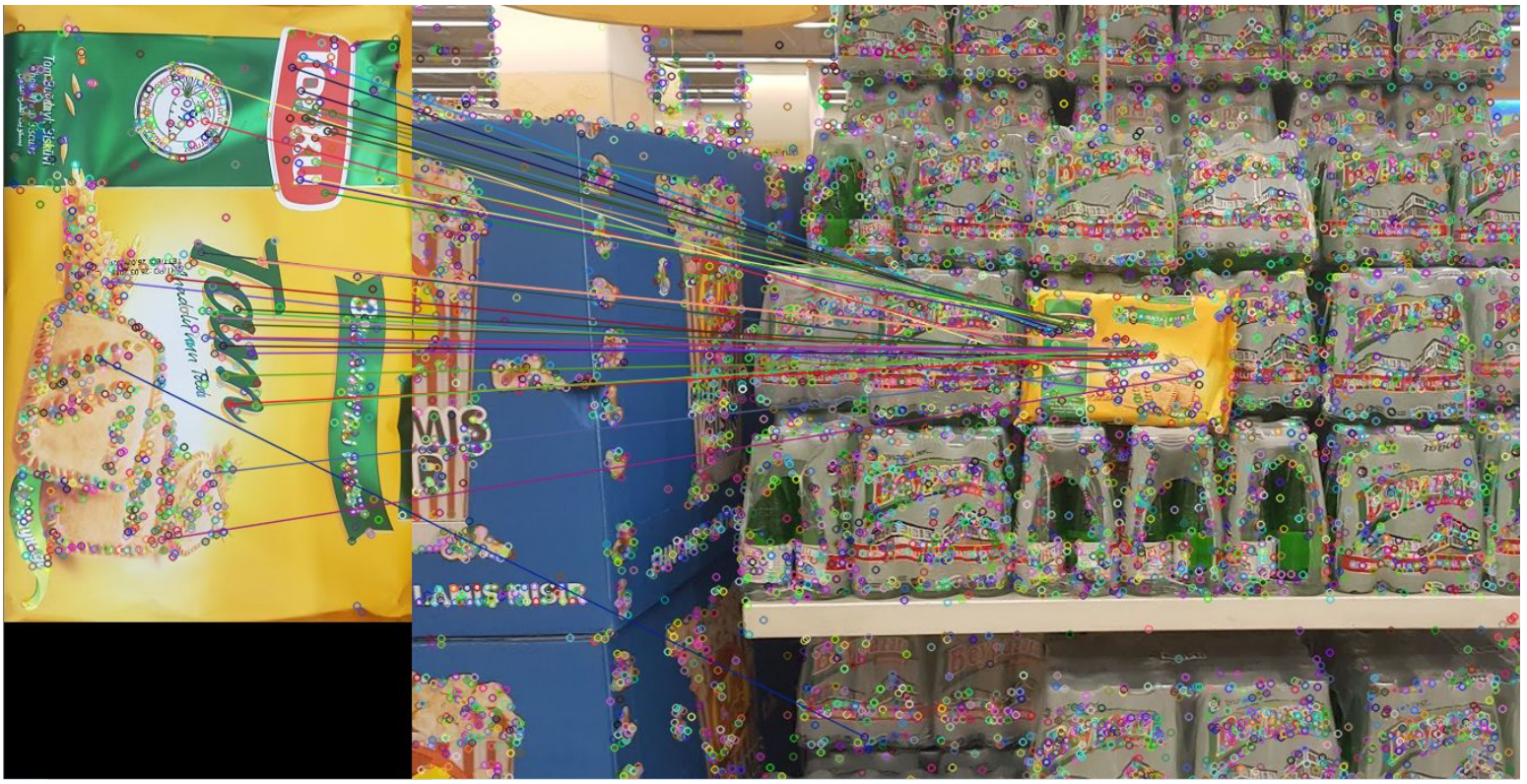
TORKU

- Feature detection

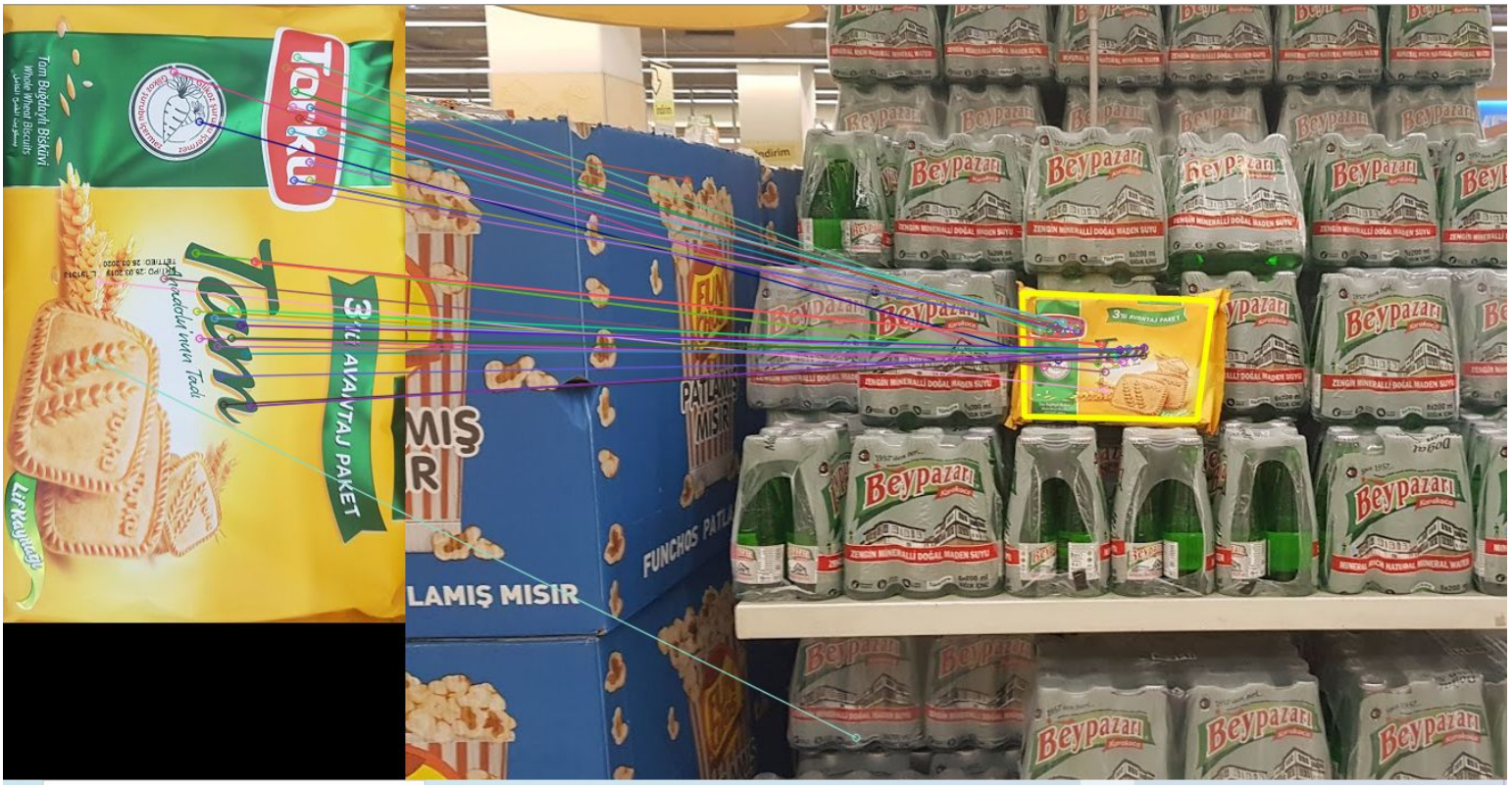
- Feature matching



- Feature matching elimination



- Homography transformation



PEPSİ

- Feature detection

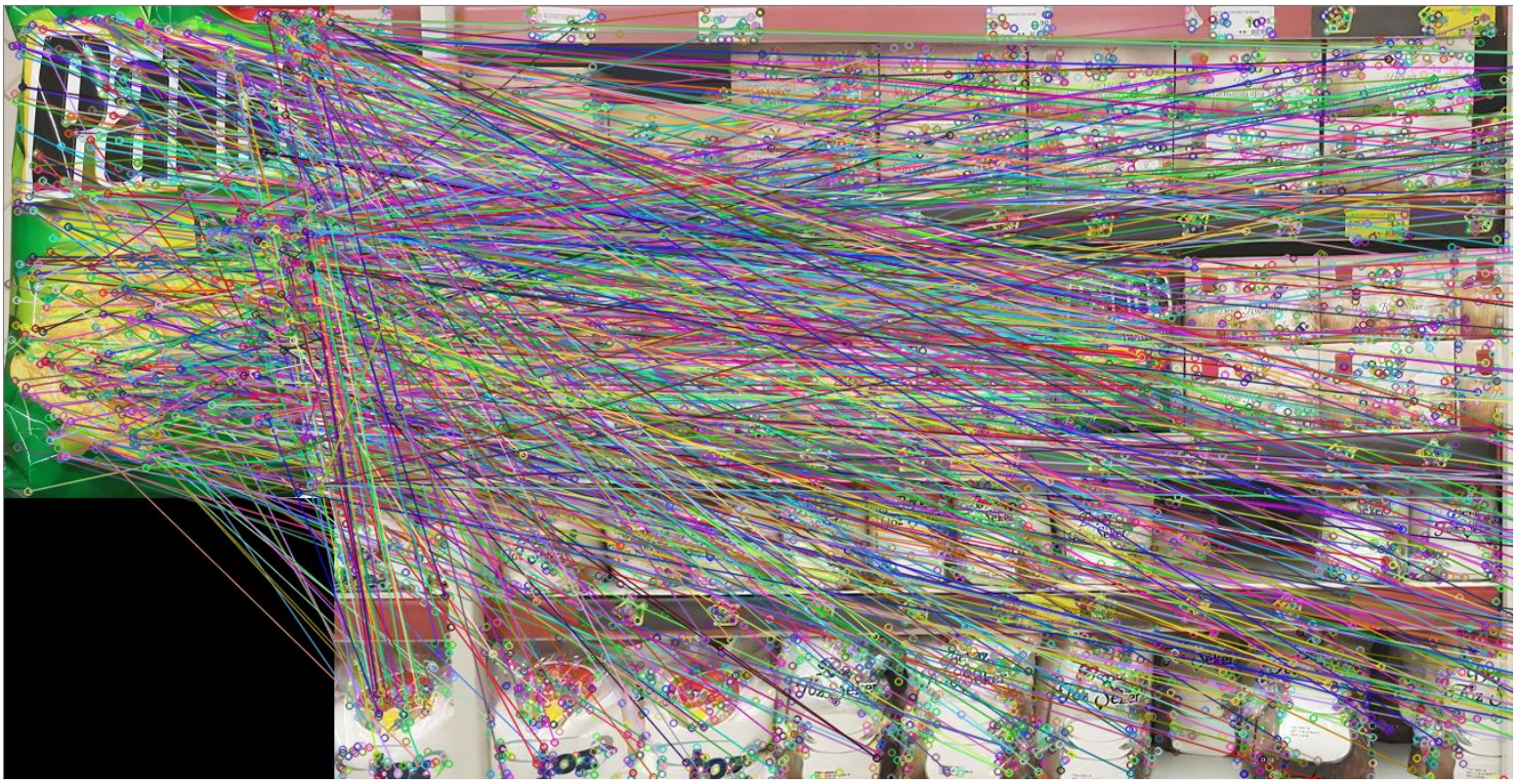
- Feature matching

- Feature matching elimination

PATOS

- Feature detection

- Feature matching



- Feature matching elimination

- Homography transformation

LIPTON

- Feature detection

- Feature matching

- Feature matching elimination

- Homography transformation

HELLMANN

- Feature detection

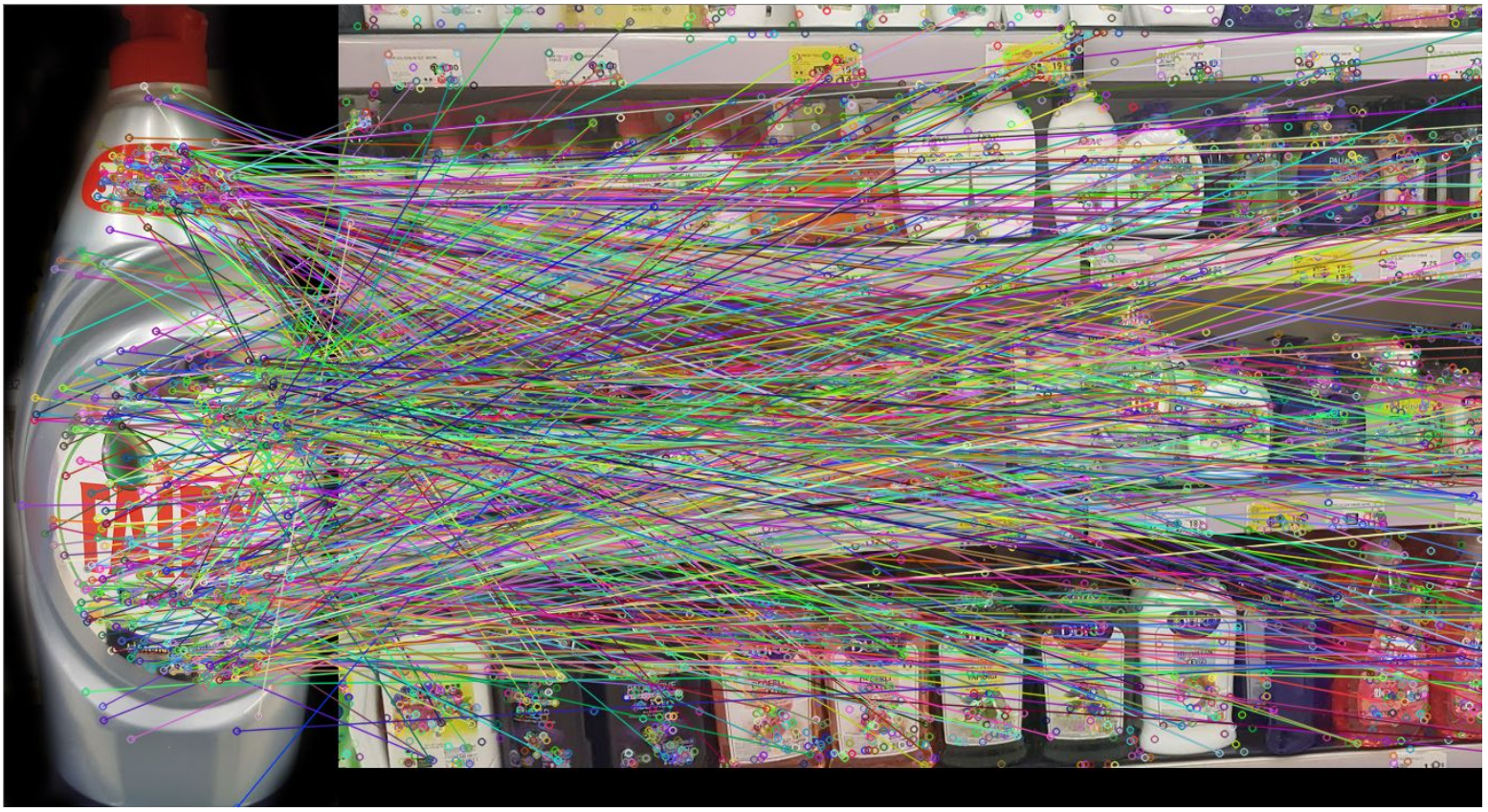
- Feature matching

- Feature matching elimination

FAIRY

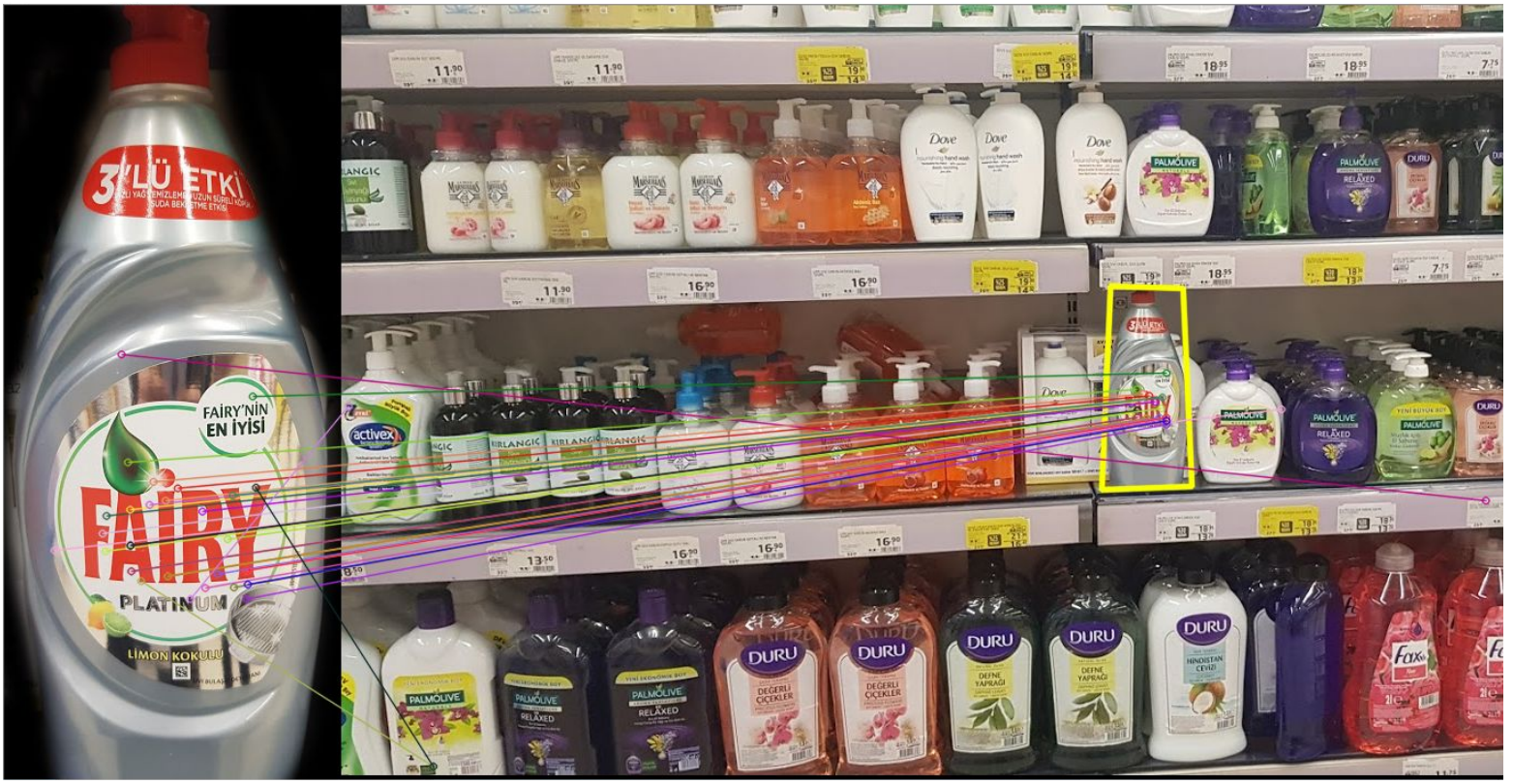
- Feature detection

- Feature matching



- Feature matching elimination

- Homography transformation



DOGANAY

- Feature detection

- Feature matching

- Feature matching elimination

- Homography transformation

Experimental Results

In this project, I found 6 products correctly. However, I could not find pepsi and hellmann products. The reason is that after elimination of the feature matching part, there were not enough and correct(inlier) matches to be able to make a homography transformation. I control whether homograpy will be done or not according to the number of good matches features after applied FLANN and Lowe's ratio test. Therefore, I missed 2 two product detections and did not make false detection in terms of product detection.

For each product, first, I found the number of matches between the product and the shelve, second, I found the number of good matches according to Lowe's ratito. Lowe's ratio eliminates bad matches when there is a enough difference between the distances of the key points in the products and the corresponding best and the second best matches. By adjusting the threshold of the ratio of difference of distances, I eliminated more outliers, however, I also lost some inliers too. Even though I adjusted thresholds for each product separately, I still had some outliers in some products and even lost inliers for pepsi, hellmann, so could not find these products. Then, by looking at the resulting output pictures, I identified the number of inliers and outliers in good matches. Results are in the table below:

	All matches (FLANN)	Good matches (Lowe's Ratio)	Inliers	Outliers
Sut	602	21	19	2
Torku	1058	40	39	1
Pepsi	114	1	1	0
Patos	736	57	53	4
Lipton	316	25	25	0
Hellman	263	3	1	2
Fairy	639	32	28	4
Doganay	184	8	8	0

Table 1

According to the table, I obtained best results for the product Lipton, because there is no outlier and the number of inliers are high. The worst one is the Hellmann product because there is only one inlier and two outliers at the end.

Conclusion

In this project, the most challenging part for me is to decide which feature detection, extraction and feature matching algorithm will be useful and give desired results at the end. Also, adjustment of the parameters for SIFT, FLANN, and for Lowe's Ratio was challenging but for some products I obtained better results by doing that. However, I believe that usage of different techniques such as SURF, Brute-Force matching and some pre-processing on images would improve the success of the visual product search project.

References

1) https://pysource.com/2018/03/21/feature-detection-sift-surf-obr-opencv-3-4-with-python-3-tutorial-25/
2) https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html
3) https://blog.francium.tech/feature-detection-and-matching-with-opencv-5fd2394a590
4) https://docs.opencv.org/master/d1/de0/tutorial_py_feature_homography.html
5) https://pysource.com/2018/03/23/feature-matching-brute-force-opencv-3-4-with-python-3-tutorial-26/