COMPUTER VISION FINAL EXAM PROJECT

**AR with OpenCV**

Introduction and problem description

The idea of this project is to show how we can use OpenCV to create an augmented reality experience or, in other words, to enable users to see virtual objects (images) in their real-life surroundings.

The project by itself doesn’t do much logically, it is simply put, just an augmented visualization of pictures on a live video stream, but it is in its essence the backbone of many games and applications.

Solution

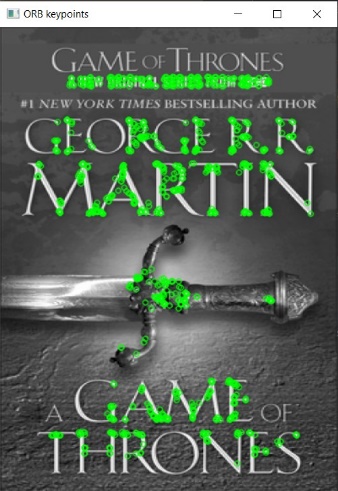
When it comes to approaching the project implementation, the first thing that needs to be done and understood is feature detection. To detect and describe key points (features) oriented FAST and oriented BRIEF are used. They are a part of the OpenCV library and work by detecting the strongest points, finding their orientation and as a result returning their descriptors.



Once the image of which we want to find and describe the key points is imported, resized and converted to grayscale, we can initialize the detector and draw the key points.



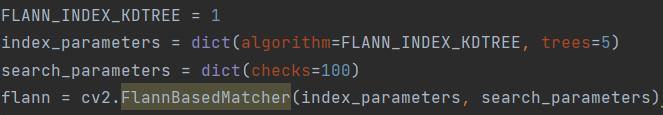
The output image would look something like this:

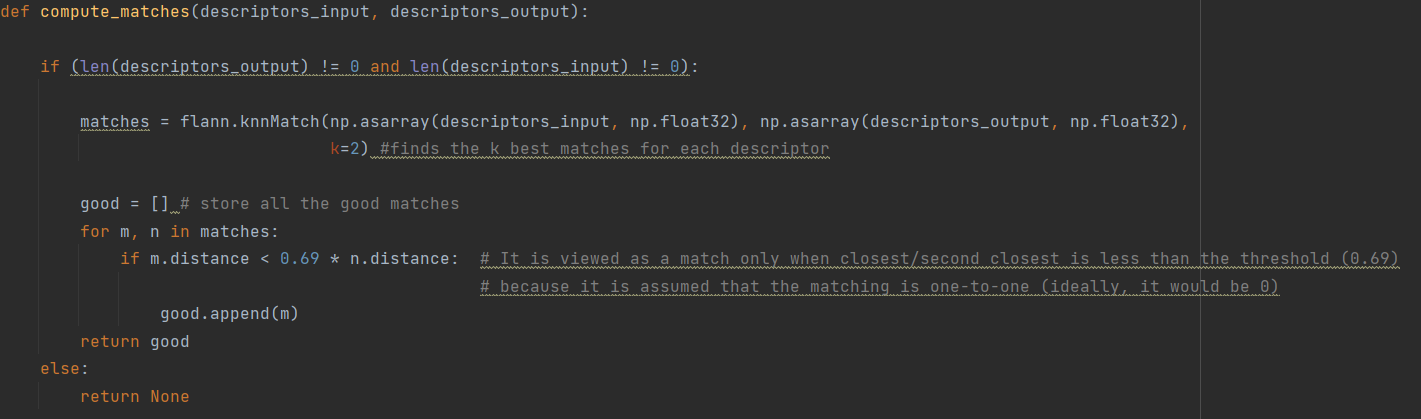
As we can see, many distinctive points have been  
 recognized by the detector.

These points mainly correspond to edges on the  
 letters.

Once the detection is done, the feature matching takes place. Feature matching in this case refers to the comparison of the key points in one image versus the key points in another image. Now, what is actually being compared within those key points are the descriptors that we got as a result of the feature detection. For this, FLANN Based Matcher is used.

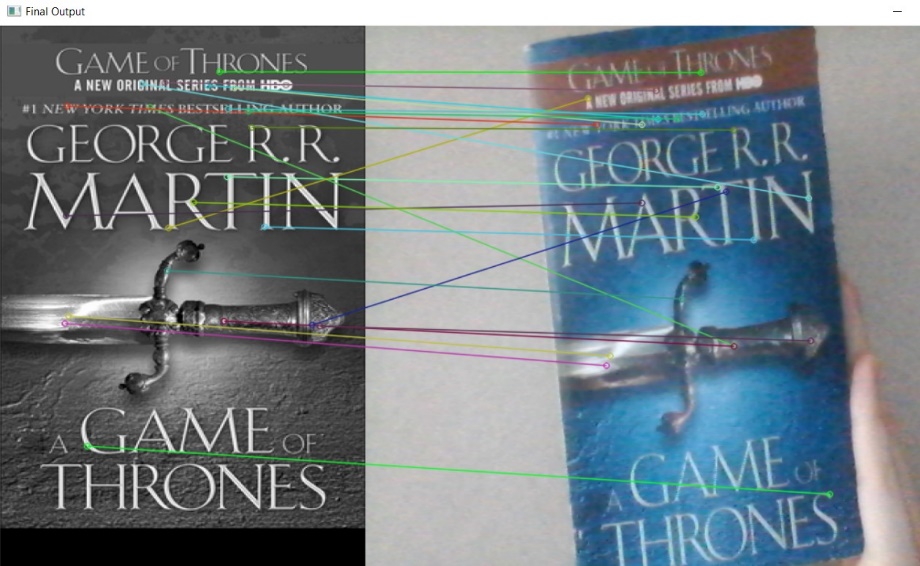
Moreover, only finding arbitrary matches is not sufficient, the matches that we need are the best possible ones. To ensure this, FLANN Based Matcher is used and it will return approximate nearest neighboring matches.





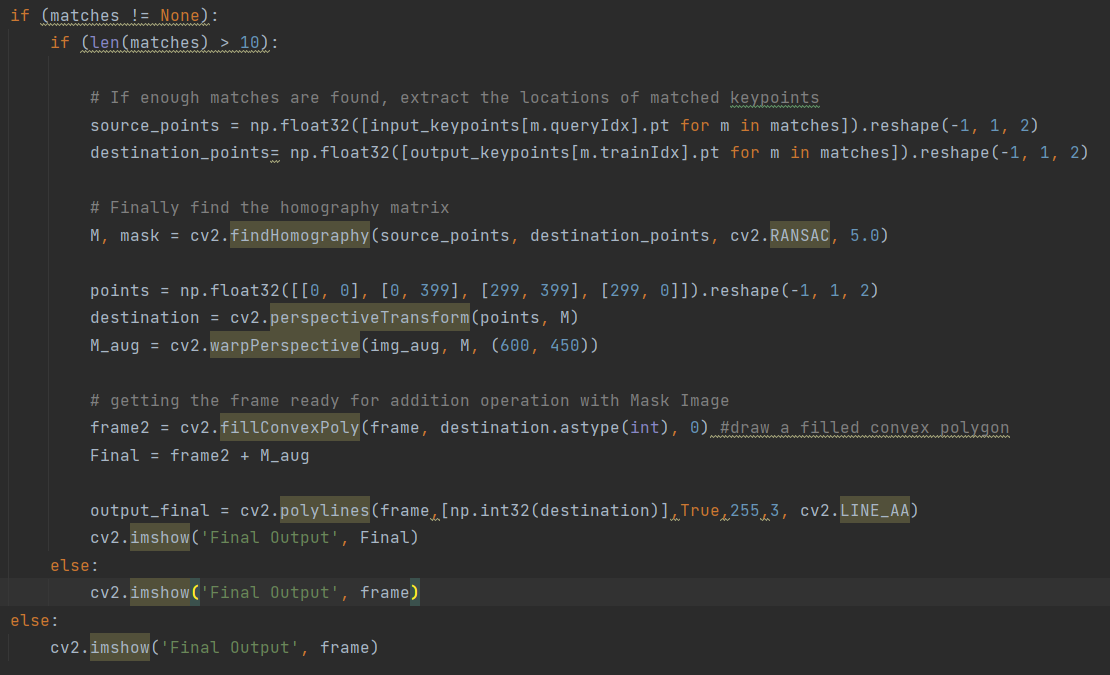
The KnnMatch function within the FLANN Based Matcher works by comparing the arrays of descriptors in the input image and in the output image, and finds 2 closest matches for each descriptor in those arrays. Now, because we want to match one-to-one and not one-to-two, all those good matches are checked against a threshold (0.69 in this case) and the one with the smaller distance is appended.

Once the points are connected and those connections are drawn the output should resemble this:



Finally, once we know the key points and once we have connected them, the image insertion that will be regarded as augmented reality can be made.

However, it is important to specify that we only want to make the insertion if a sufficient number of matches has been found, in this case I chose 10 matches as a minimum.



To insert the AR image we use OpenCV’s fillConvexPoly() and polylines() functions.

Conclusion

This project was made to demonstrate only one of the many possibilities that computer vision with OpenCV library offer. It is a project that, as said in the introduction, doesn’t do much on its own, but without projects like this the implementation of apps that integrate digital visual content such as Google Lens and IKEA place would not be possible. As a final statement, here is the image representing the end result of this project:

