Computer Vision - Augmented Reality Part 1:

Assumptions:

1. We photographed a picture with nice background and in our opinion with good subject and multiple visible features, and by so the algorithm used to find those features will find a lot more than other pictures or the background the picture is on.
2. We filmed the chosen picture from a different amount of angles and distances so that the video itself will look more realistic, the background itself for the video that we did is a surface with high light diffusion so the back light will not interfere with the rest of the demo.
3. The process itself is quiet easy to understand. The steps are:
   - Find the known image key points.
   - Find for each frame there key points.
   - Check if there is a match between the key points of the known image and the current frame.
   - Using a few linear algebra tricks get a matrix that represents the translation of the known image to the image in the frame.
   - Using the matrix convert the wanted picture to the size and shape of the picture in the current frame.
   - Combine the two images, the current frame and the result of the last step.

On each step I will further elaborate:

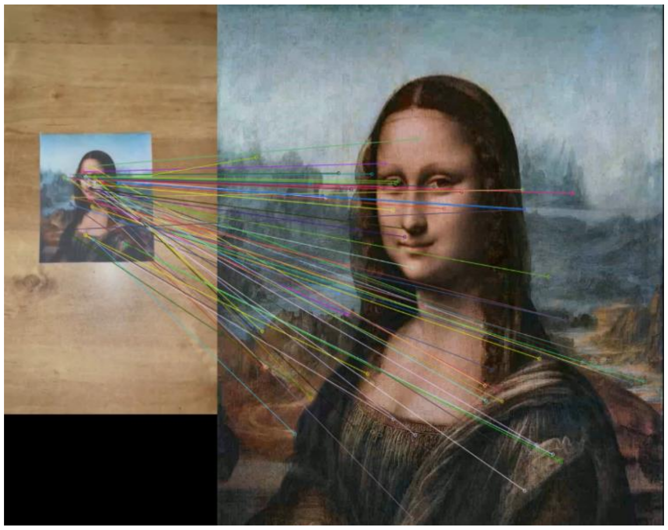
- ➢ The first and second steps are almost the same, using a feature extraction object in our project, we used the SIFT algorithm, we can extract the features just by using the 'detectAndCompute' method given by the object, the only difference between the steps is the picture the method is used on.
- ➢ Using a matcher object, in out project we used 'BFMatcher' (brute force matcher), we can use the method 'knnMatch' [the will find using the 'k-nearest-neighbors' algorithm] that will find the correlated key points between the two images. The method can find a few key points that are not relevant or even unwanted key points such as finding a correlation with the surface the picture is on to the background of the picture, so as a precaution we remove key points that the difference in distance is more than the minimum distance acceptable. In our tests we found that the minimum ration

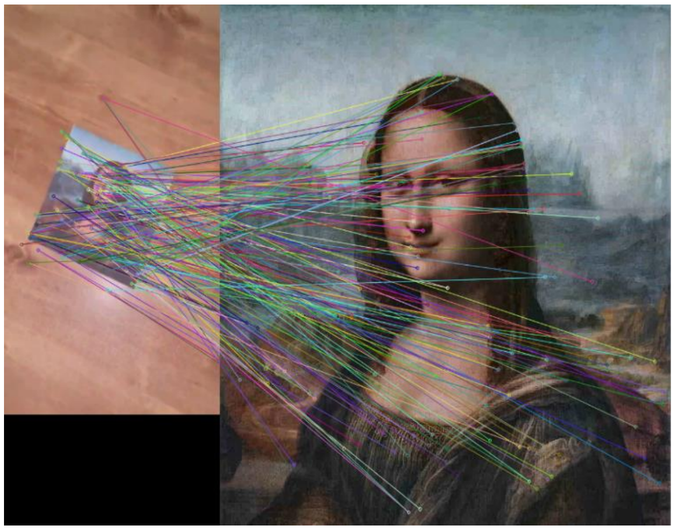
of distances must be bigger or equal to 0.5, and in about the 1.0 mark the matcher will find more and more mistakes and irrelevant key points.

- After we got the relevant key points out we will divide the set to the key points of the current frame and the key points of the known image.
  Using the method 'findHomography' of cv2 and the newly created frame key points, known key points and an algorithm in the name of RANSAC the method can find the only relevant points and find the homographic matrix between the known image and the image in the current frame.
  the homographic matrix is a matrix 3x3 that represent a combination of a few linear 'tricks' that can convert the known image to the same position of the image in the frame, one such trick is "translation" – move the object in space to a different (x, y) position without changing the object shape, another is "shear" – warp the shape in one direction an example for that is converting a rectangle into a parallelogram.
- Using "warpPerspective" method we can use the homographic matrix found in the last step and convert any image to the shape of the image in the frame, so we will use this method on the target image we wanted to place into the current frame. The resulting image after the method is the target image in the shape of the image in the frame with black background.
- Lastly we will combine the images to one. In the last step we created the target picture with the wanted shape and size but the background is not the background of the frame, so I created a mask that is exactly the same as the target picture but the background is white and the target image is all black, I then combined each pixel with the masked pixel using 'bitwise and' method with the current frame that gave me the current frame without the wanted image, and then I added the target image from the last step and got the resulting image.

Let’s see some example of a few of the steps:

Here we can see the matcher finds the corresponding key points between the two images. On the Left we can see a good amount of key points matched with about 0.5 minimum distance, and on the right we can see a lot more key points with a few mistakes, there are a few points that the matcher found that the background of the image is found in the background that the image is placed on, The right picture has been taken with minimum distance of 1.0 and as we can see there are more key points but more error created as a result.





In this picture we can see two videos together. The right picture is the real video and the left picture is the rendered video using the steps I described before.
We can see that the images align almost perfectly, but as all things we can see where the model fails, in this case we can see on the edges of the frame the blue sky from the background of the image beforehand.
But in most cases the model preforms better than expected.