Lab report 6 Computer Vision - Matteo De Gobbi

I decided to use SIFT features with FLANN based matching because they seemed to work better experimentally.

To extract the SIFT features from both images we create a Ptr<SIFT> detector, in opencv Ptr is just an alias to C++'s native shared pointer.
Then we call the detectAndCompute function in the SIFT object passing as input the image, a reference to a vector to store the keypoints and a reference to a Mat to store the descriptors.

Then we make a DescriptorMatcher specifying FLANNBASED as a parameter. I also tried the BRUTEFORCE version but it's slower than Flann without improving the matches.

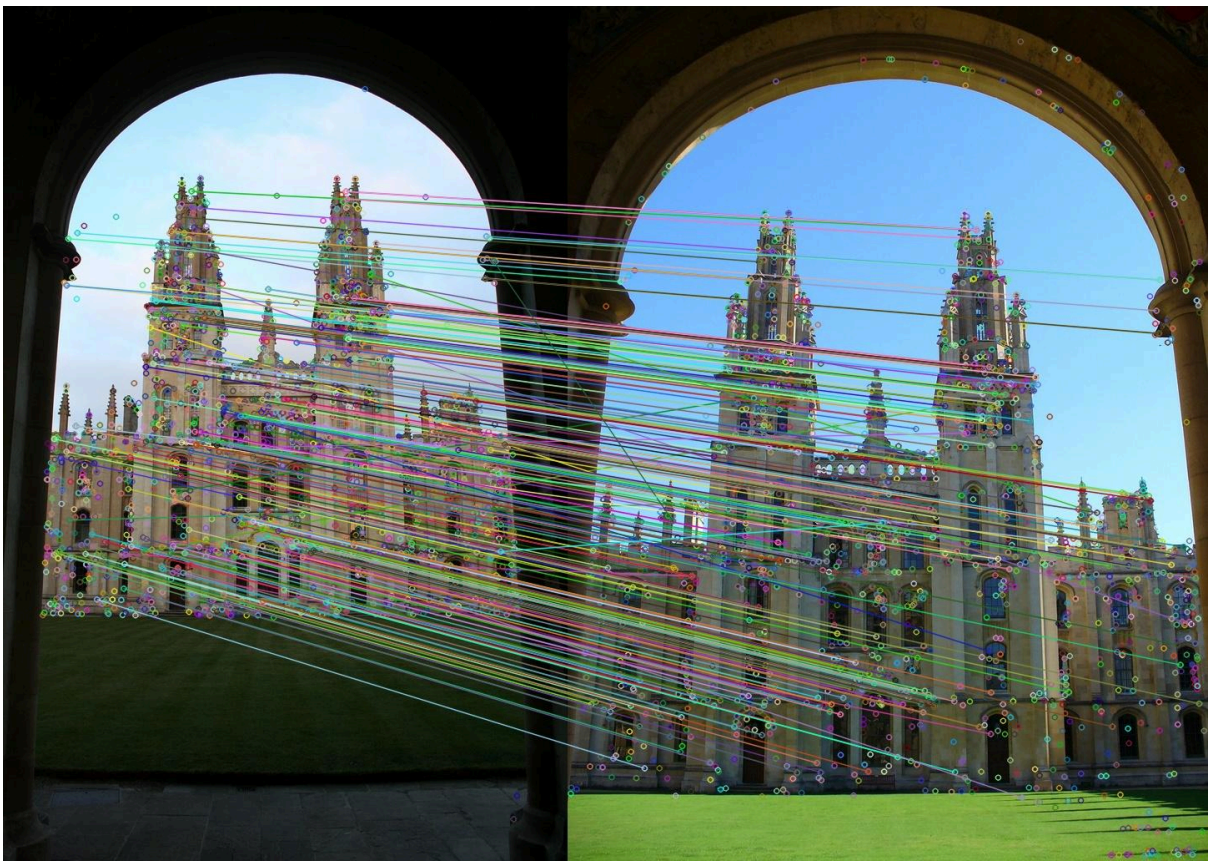
Then we match the descriptors of the two images using the K Nearest Neighbor approximate algorithm provided by the DescriptorMatcher class.

```
Ptr<SIFT> detector = SIFT::create();
std::vector<KeyPoint> keypoints1;
std::vector<KeyPoint> keypoints2;
detector->detectAndCompute(im1, noArray(), keypoints1, descriptors1);
detector->detectAndCompute(im2, noArray(), keypoints2, descriptors2);
Ptr<DescriptorMatcher> matcher =
    DescriptorMatcher::create(DescriptorMatcher::FLANNBASED);
std::vector<std::vector<DMatch>> knn_matches;
matcher->knnMatch(descriptors1, descriptors2, knn_matches, 2);
```

Then we only keep the good matches using the Lowe ratio test: we delete matches in which the first and second match have almost the same distance. This is because a single keypoint should only match to another single keypoint. I also use the default threshold of 0.7 because it works well.

```
std::vector<DMatch> good_matches;
for (uint i = 0; i < knn_matches.size(); i++) {
  if (knn_matches[i][0].distance <
      ratio_thresh * knn_matches[i][1].distance) {
    good_matches.push_back(knn_matches[i][0]);
  }
}
```

Then we use the drawMatches function to draw the good matches that are left after the ratio test, for example:

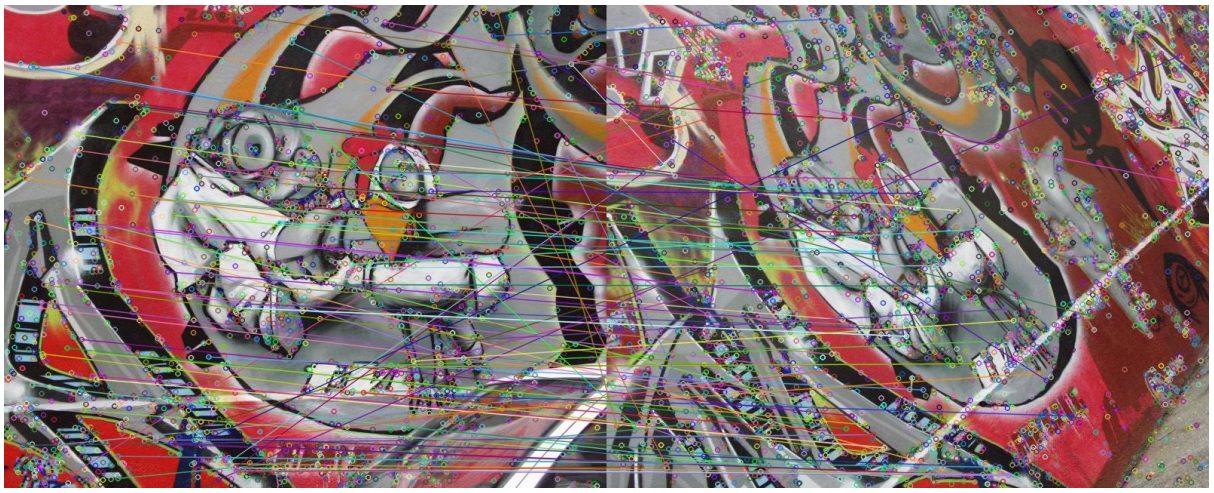


Then to determine whether or not the subject of the image is the same I defined a rule that is based on how many good matches are left after the Lowe ratio test compared to the total number of keypoints found in the images. I consider the subject to be the same if the number of good matches is more than 2% of the number of keypoints (I take the minimum of the number of keypoints between the two images because the number of matches could never exceed the minimum between the two).

```
if (good_matches.size() > percent_of_matches_threshold *
                                min(keypoints1.size(), keypoints2.size())) {
```

This works fine for most images in which the same object is represented but it fails on the images of the Bodleain, I also tried using different types of features like ORB and different thresholds but the number of matches is too low for these images:

On the other sets of images it works well and it correctly identifies if the images represent the same subject, some examples that are identified as the same object are:



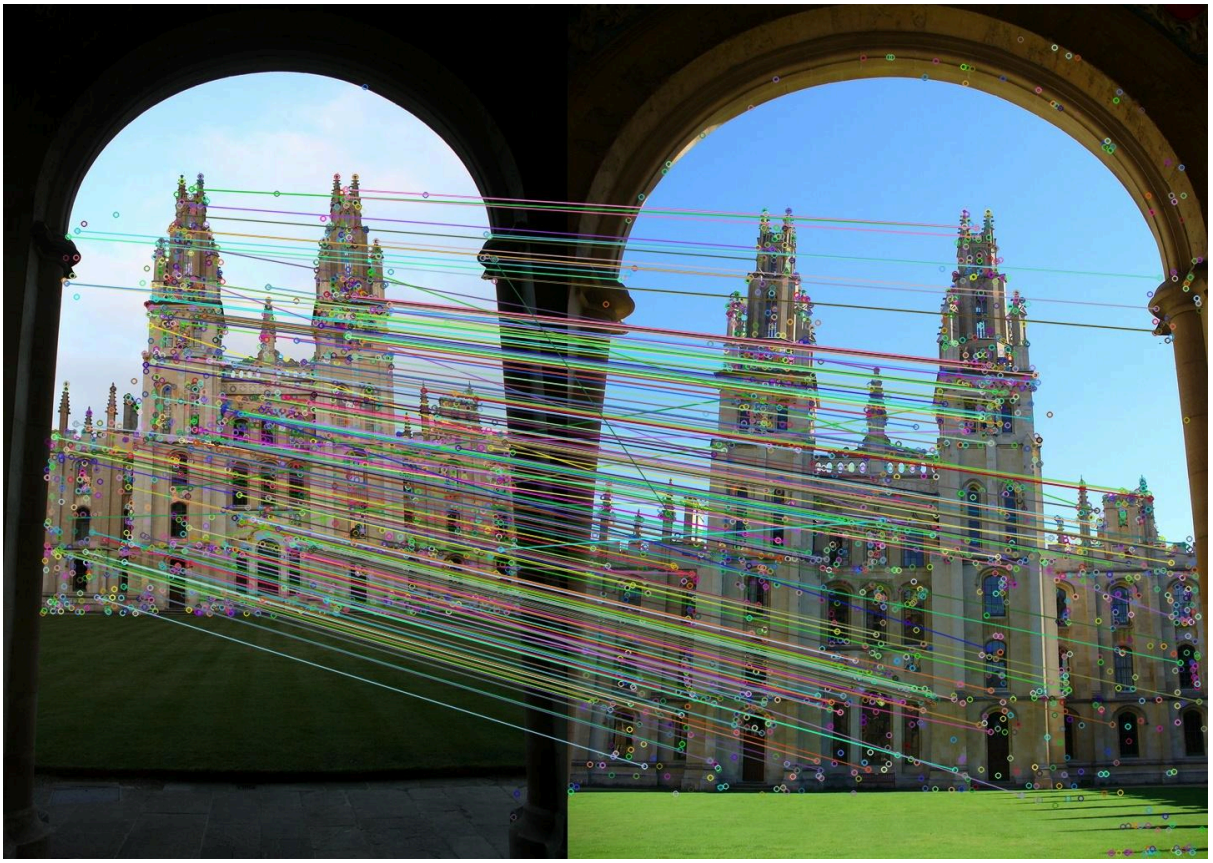
Determining if the object is rotated or the perspective has changed

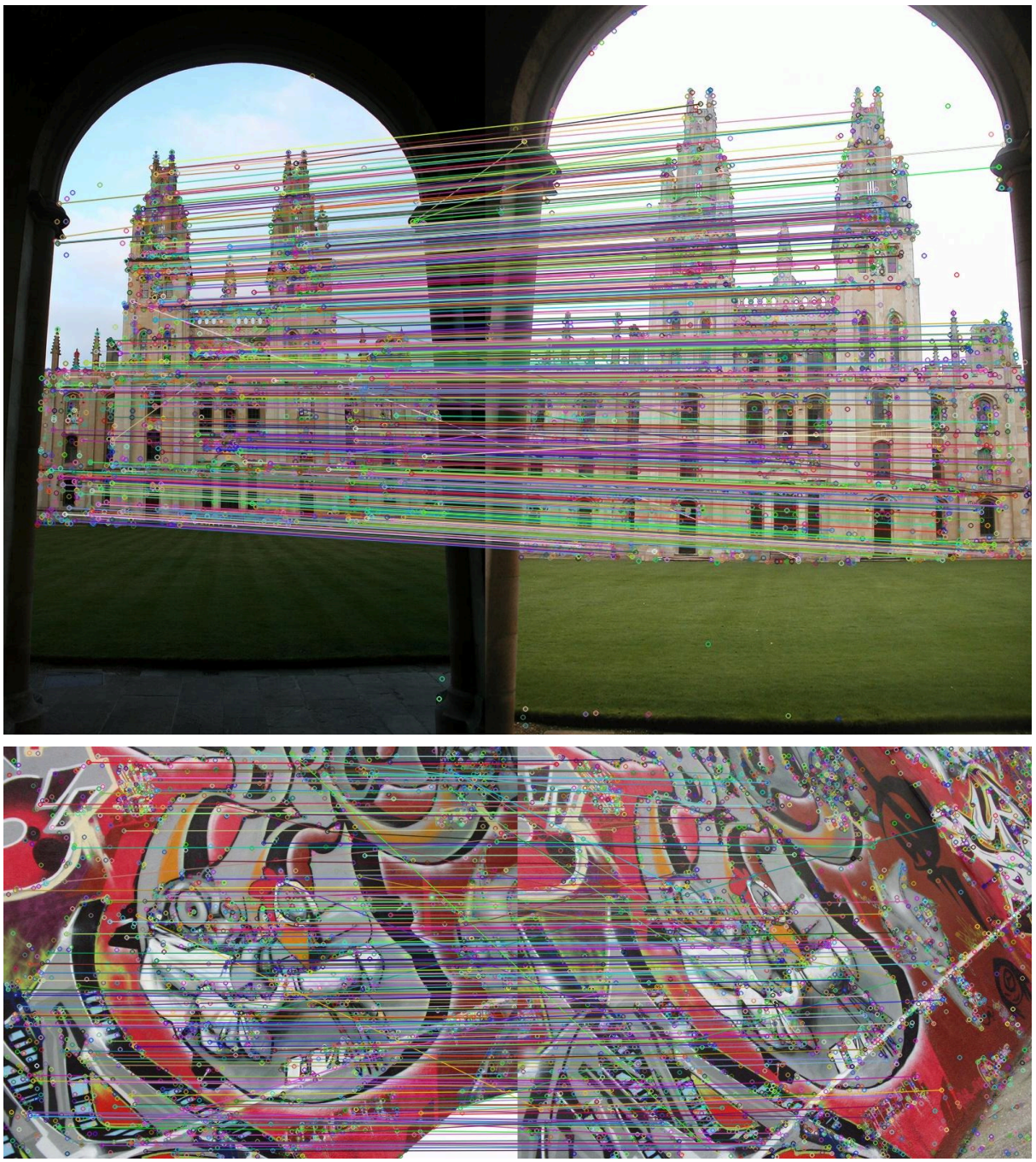
Lastly, to identify whether couples of images that are considered a good match represent the same object with a strong rotation/transformation applied or not, I used the findHomography function.

This function takes as input the two vectors of good keypoints from the two images and gives in output a matrix H that corresponds to the transformation from the first vector of keypoints to the second vector.
In order to use H to check whether the image is rotated/transformed or not I used a custom approach:

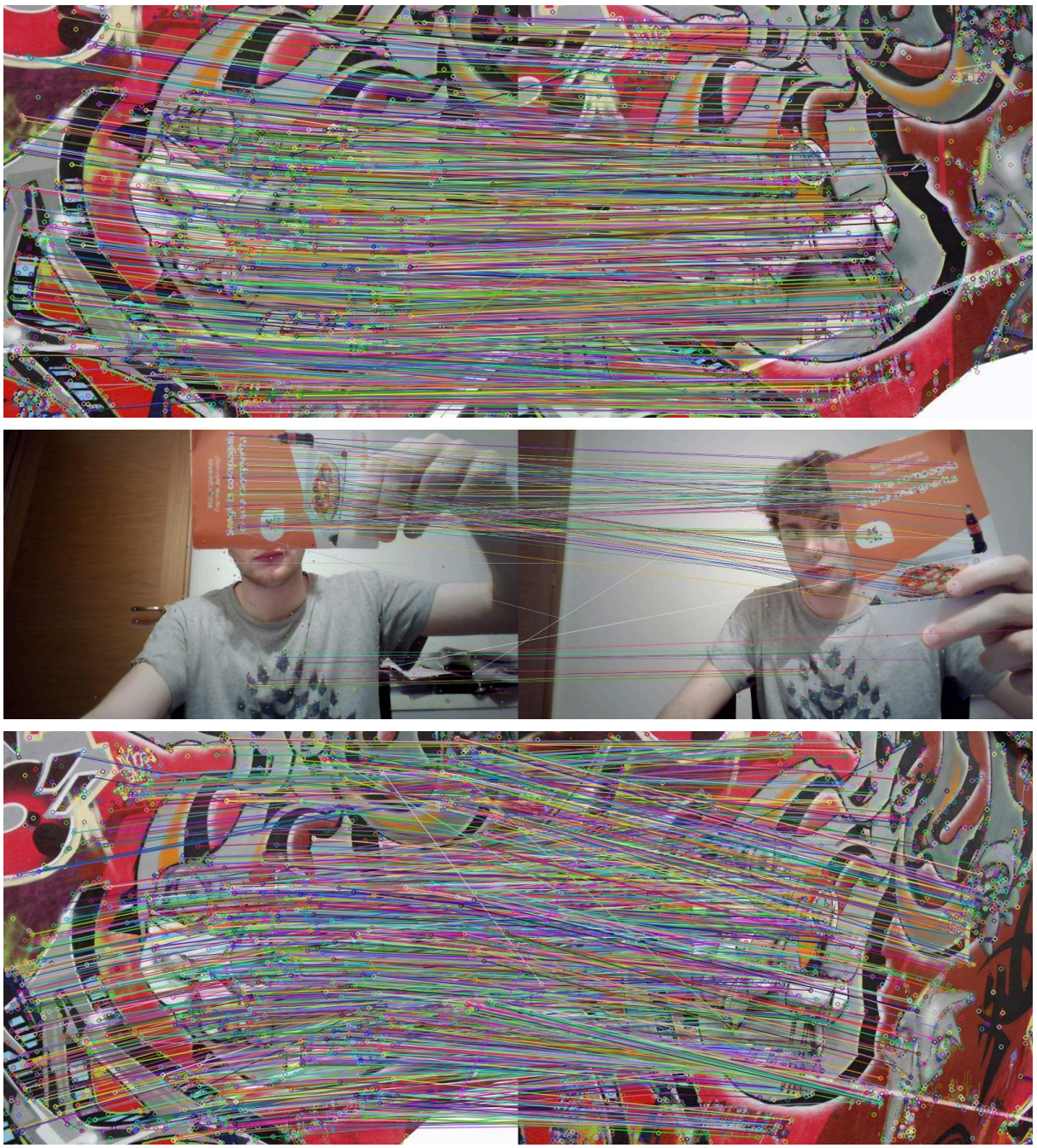
- Create an identity matrix I with the same dimensions of H
- Compute the matrix diff = H-I
- Compute the square root of the sum of the squared elements of diff
- Compare this sort of “distance” between H and the identity with a threshold, if the distance is smaller the transformation is close to the identity so the object is not rotated, if it’s bigger than the threshold the opposite holds

The threshold was determined experimentally and 100 seems to work very well, some examples of images that are considered not rotated are:





While some images that are considered as rotated/transformed are:



We can see how strong transformations are detected while minor changes of perspective are not.