600.661 Computer Vision Homework #2
Yichuan Tang

Task 1. Feature Detection

Size of Gaussian kernel is 7 by 7

Window size used for corner detection is 9 by 9

Radius of window in nonmaxsupprt is 5

Factor k is 0.05

For different images the corner thresholds used are different. In my program the corner threshold is not a value of corner strength, instead it is a factor (in python program called **'corner_thresh_ratio'**) which multiplies the mean value of corner strength image to obtain the real corner threshold.

The following table shows corner threshold (factor).

Image	Corner_thresh_ratio
bikes1	1.5
bikes2	1.1
bikes3	1.1
graf1	2.4
graf2	2.8
graf3	2.2
leuven1	2.4
leuven2	2.2
leuven3	2.1
wall1	4.5
wall2	3.3
wall3	3.2

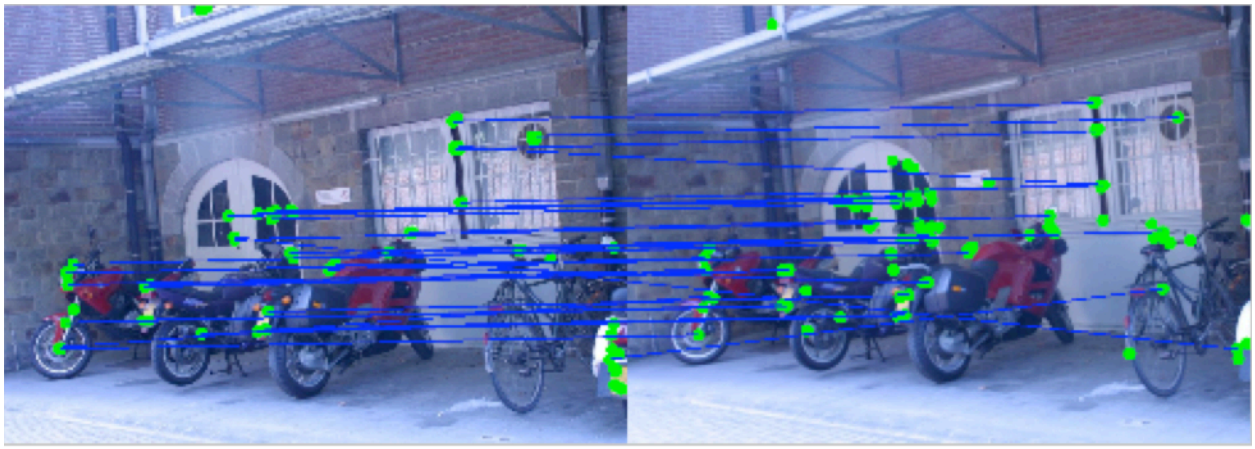
Here shows the corner features detected in bikes1 using Harris corner detection.

Task 2. Feature Matching

In feature matching normalized cross-correlation (NCC) is used to match similar features. Mutual marriage is used to find good matches occurring in both passes and drop bad matches.

Patch radius is 8

Here shows eight sets of matched images.

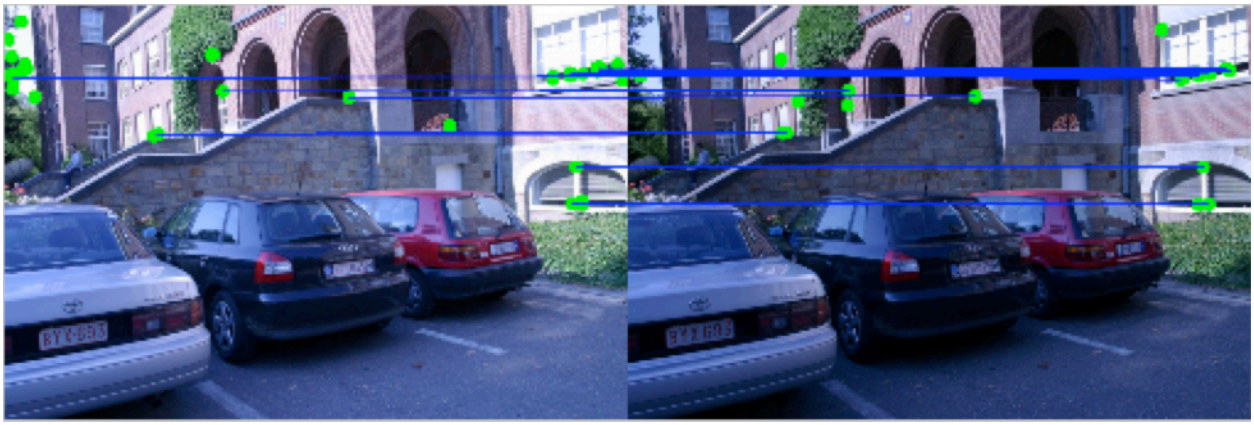


Matched: bikes1 and bikes2

Matched: bikes 1 and bikes3

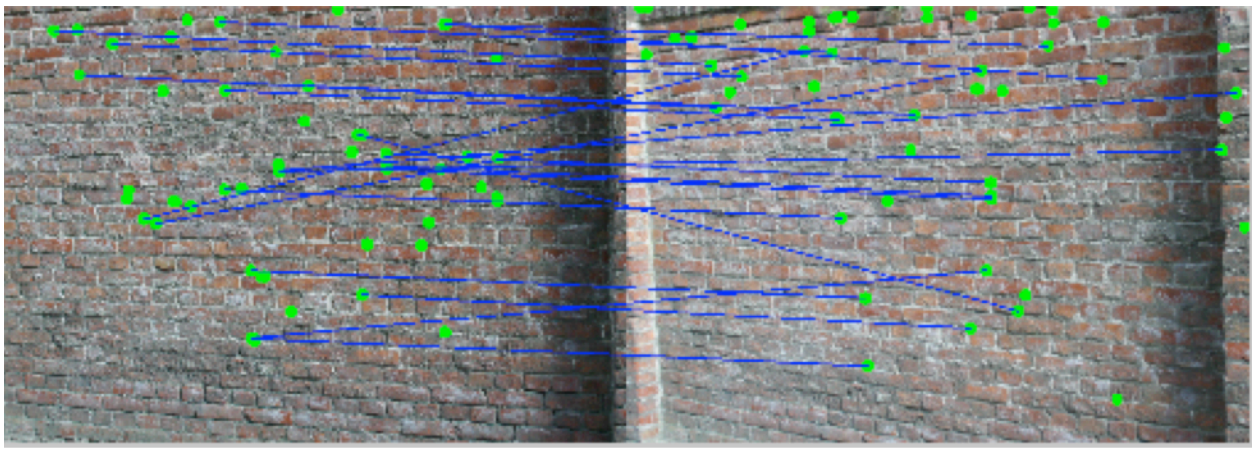
Matched: graf1 and graf2

Matched: graf1 and graf3

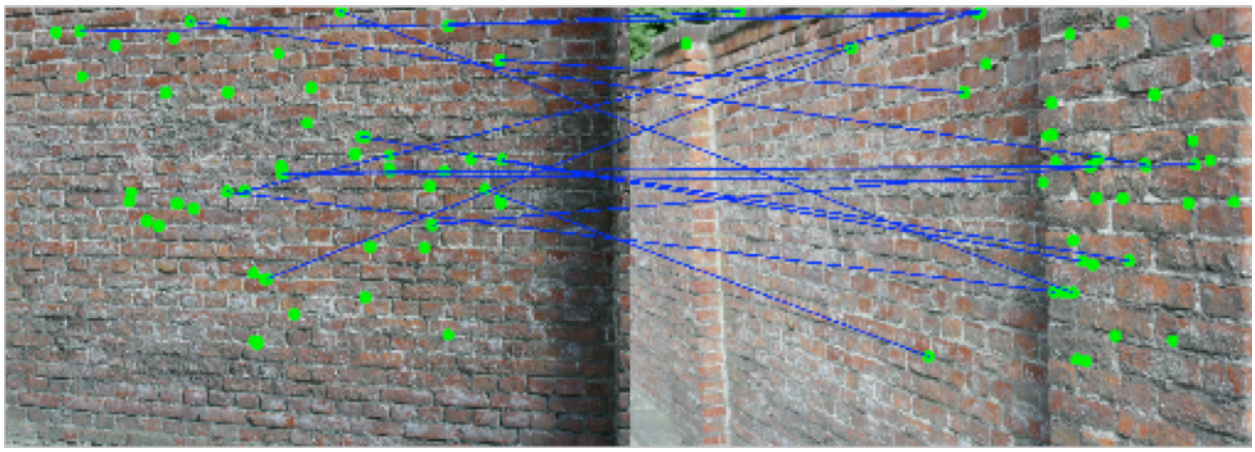


Matched: leuven1 and leuven2

Matched: leuven1 and leuven3



Wall1 and wall2



Wall1 and wall3

Task 3. Alignment and Stitching

To find the affine transformation, RANSAC algorithm randomly selects three point matches and compute an estimated affine transform using least square approach. Then this estimated transformation is verified by other matches. That is to apply the estimated transform on one point in the match (which is not used in computing this estimated transformation) and then compute the error between position of transformed point and position of the other point in the match. The summed errors serves as a criteria to tell whether the estimated affine transformation is a good one. We need to repeat previous process many times and find the estimated affine transformation which produces the minimum summed verification error, and this becomes the final output of RANSAC algorithm.

The computation of projection transformation using RANSAC follows the same logic. The difference lies in the math to compute two transforms. As affine transform matrix has six degrees of freedom while projective transformation has eight, the minimum number of matches required to estimate transformation is increase from three (affine) to four (projective).

Part1: Align and stitch image pairs using affine transformation

My code fails to find the affine in ‘graf1 and graf3’, ‘wall1 and wall3’. This is because the transformation between these image pairs are not affine, it is obvious that these image are taken from very different perspectives.

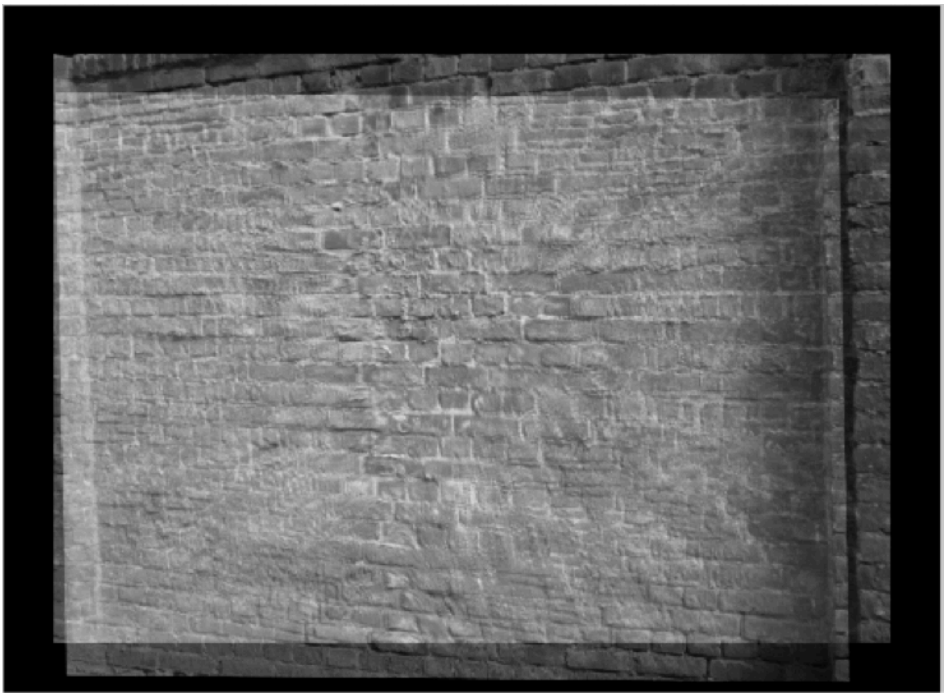
Stitch: bikes1 and bikes2 (affine)

Stitch: bikes1 and bikes3 (affine)

Stitch: graf1 and graf2 (affine)

Stitch: leuven1 and leuven2 (affine)

Stitch found: leuven1 and leuven3 (affine)



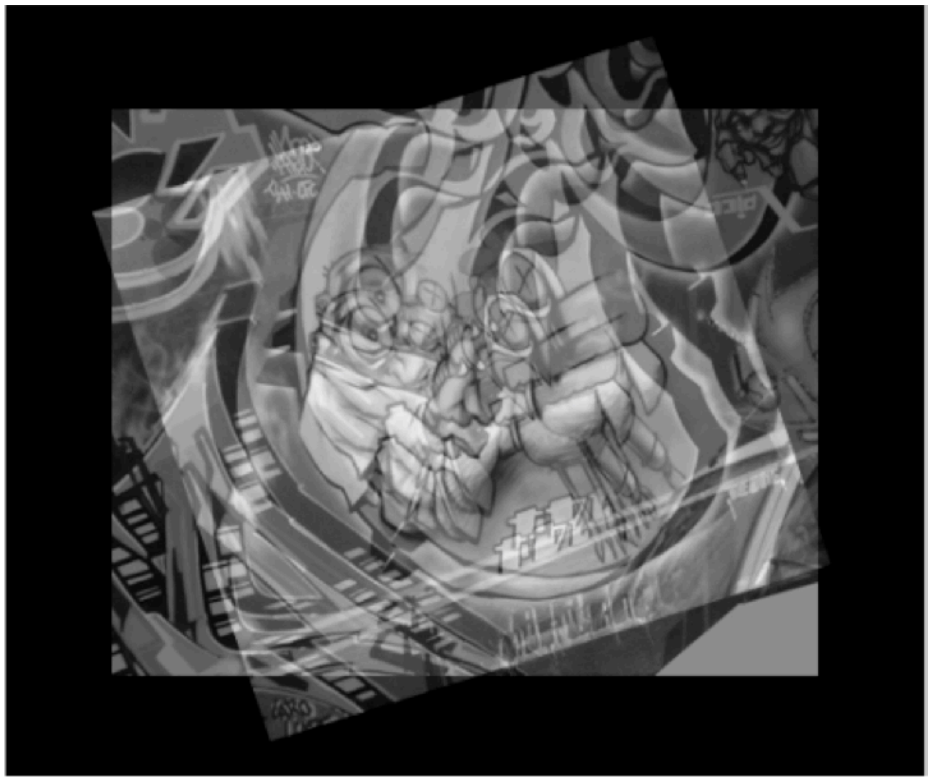
Stitch: wall1 and wall2 (affine)

Comments on the aligned stitches: the stitches for bikes, leavens are well aligned, this is because translation component dominates the affine. ‘wall1 and wall2’ looks reasonable, the central parts align well, and error becomes obvious in the boundary regions. Serious issues occur in ‘graf1 and graf2’, ‘graf1 and graf3’, ‘wall1 and wall3’. ‘garf1 and graf2’ can not be well aligned because the transformation between these two images cannot be well represented by affine transformation. Cases are similar in ‘graf1 and graf3’, ‘wall1 and wall3’.

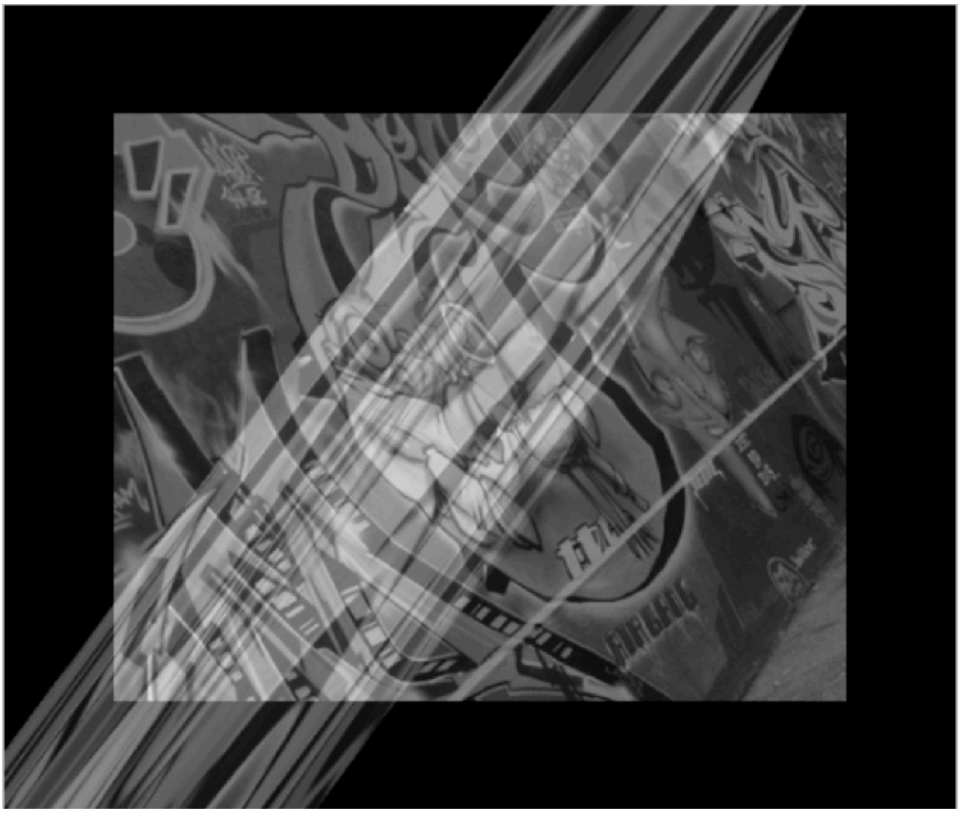
Part 2: The following shows the stitches generated by using fully projective transformation.

Stitch: bikes1 and bikes2 (projection)

Stitch: bikes1 and bikes3 (projection)



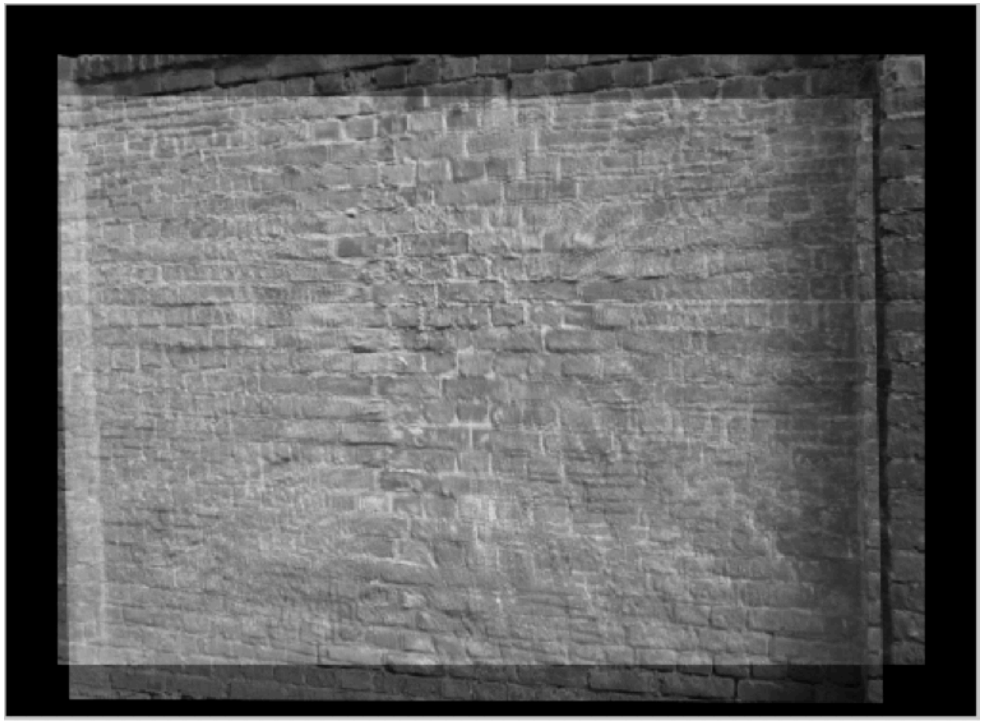
Stitch: graf1 and graf2



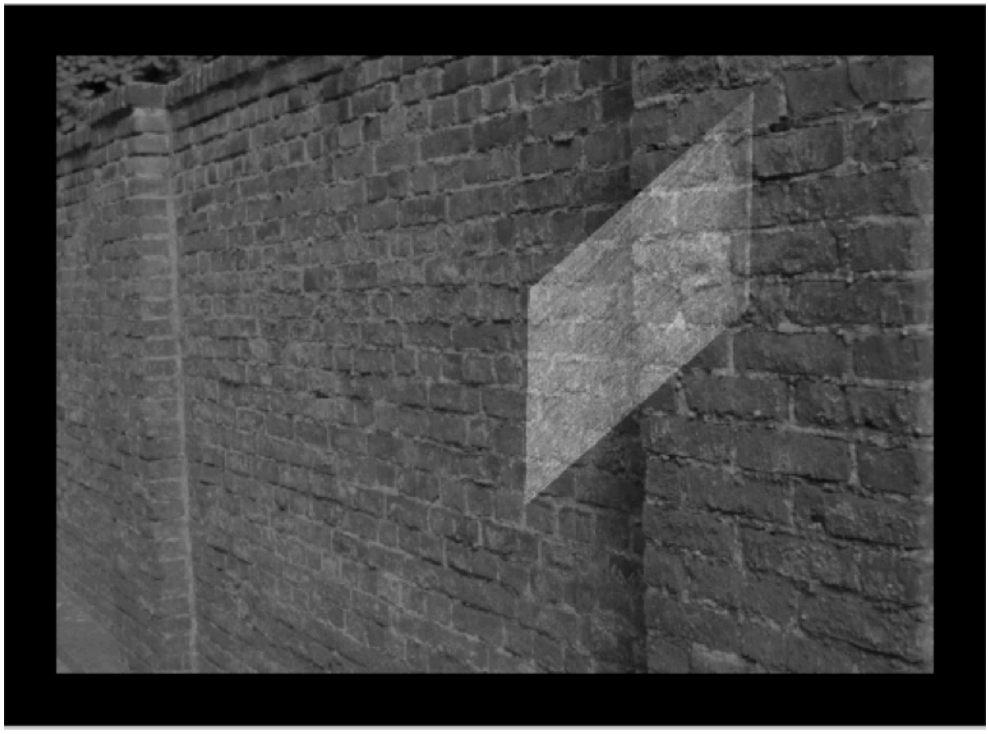
Stitch: graf1 and graf3 (projection)

Stitch: leuven1 and leuven2 (projection)

Stitch: leuven1 and leuven3 (projection)



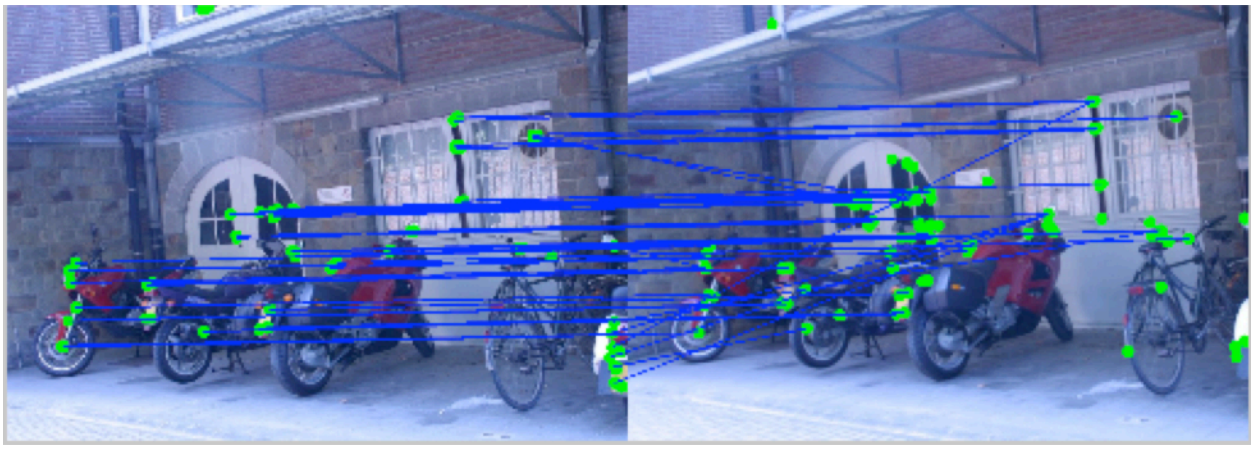
Stitch: wall1 and wall2 (projection)



Stitch: wall1 and wall3 (projection)

The stitches generated by using affine transformation and fully projective transformation are close. There is a very slight improvement in ‘graf1 and graf2’. Also the algorithm is able to produce stitch for ‘graf1 and graf3’, as well as ‘wall1 and wall3’, but the results doesn’t make much sense. Projection transformation is also not good for these images taken from different perspectives. However, in theory projection transformation should work in these cases. If we go back to the match image shown in task 2, it is easy to find that the features are badly matched, this leads to bad transformations because we do not have enough good matches for RANSAC to be effective (in slides it says >50% of matches must be good matches).

Task 4. Simplified SIFT

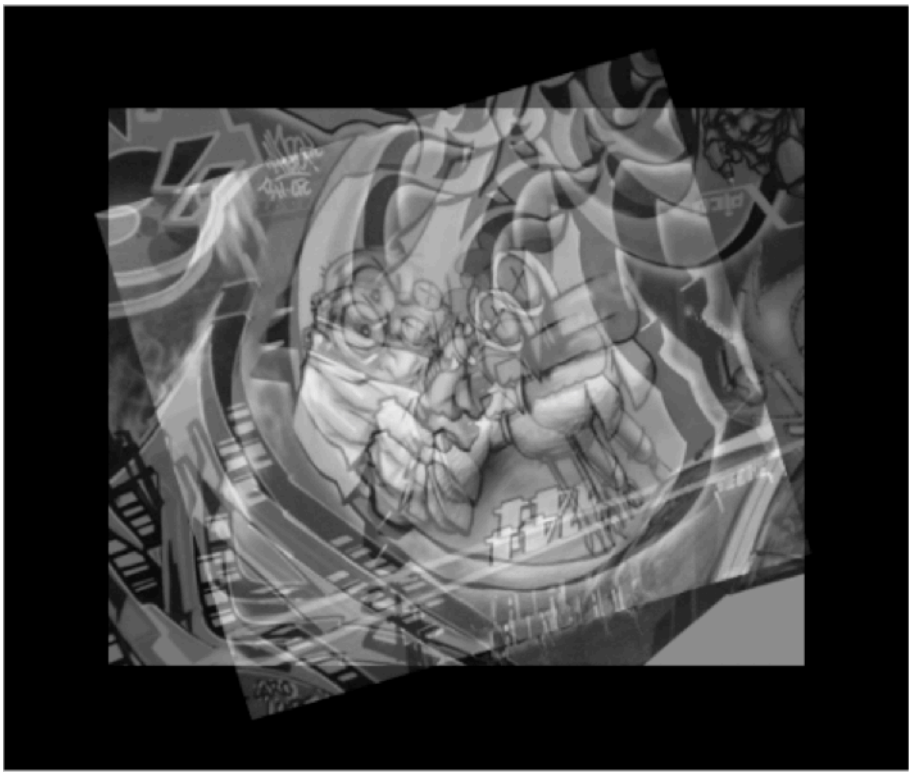


Matches: bikes1 and bikes2 (SIFT)

Stitch: bike1 and bikes2 (SIFT)

Matches: graf1 and graf2 (SIFT)

Matches are improved compared to NCC matching.



Stitch: graf1 and graf2 (SIFT)

There is no obvious improvement on the transformation.