Scale Invariant Feature Transform (SIFT): Performance and Application

Vedrana Andersen, Lars Pellarin and Renée Anderson

June 4, 2006

1 Introduction

In 2004, David G. Lowe published his paper “Distinctive Image Features from Scale-Invariant Keypoints” (Lowe, 2004, [2]), outlining a method he developed for finding distinctive, scale and rotation invariant features in images that can be used to perform matching between different views of an object or scene. His method, *Scale-Invariant Feature Transform (SIFT)* combines scale-space theory and feature detection, geared toward a broad variety of applications in the field of computer vision, such as object recognition and stereo correspondence.

As a part of the course Advanced Image Analysis at the Technical University of Denmark (DTU), we conducted a mini-project where we 1) studied Lowe’s work, 2) tested SIFT, 3) implemented a portion of SIFT ourselves, and 4) applied SIFT (combined with RANSAC algorithm) to automatic image stitching and automatic calculation of the fundamental matrix.

2 SIFT algorithm

A hallmark function of SIFT is its ability to extract features that are invariant to scale and rotation; additionally, these features are robust with respect to noise, occlusion, some forms of affine distortion, shift in 3D perspective, and illumination changes (Lowe, 2004, [2]). The approach generates large number of features, densely covering the image over all scales and locations.

The components of the SIFT framework for keypoint detection are as follows:

1. *Scale-space extrema detection.* Using a cascade filtering approach a set of octaves are generated, each octave containing the difference-of-Gaussian images covering the range of scales. Local maxima and minima are then detected over all scales and image locations. This forms a set of candidate keypoints.

2. *Keypoint localization.* Each candidate keypoint is fit to a detailed model to determine location and scale. The points with low contrast and poorly localized edge points are rejected.

3. *Orientation assignment.* Based on local image gradient, each keypoint is assigned a direction. In case of more strong directions, additional keypoints are created.

4. *Keypoint descriptor.* This is accomplished by sampling image gradient magnitudes and orientations around each keypoint and putting those in an array of orientation histograms covering the region around the keypoint. Gradients are at the scale of the keypoint (providing scale invariance), and all orientations are relative to keypoint direction (providing rotation invariance). The entries of all histograms are then put in a descriptor vector which is also normalized to reduce the effects of illumination changes.

The results of carrying out these steps are depicted in the top row of Figure 1.

For image matching, descriptor vectors of all keypoints are stored in a database, and matches between keypoints are found based on Euclidean distance. The suggested method of matching to large database is the nearest neighbor algorithm combined with comparing the distance to the second-nearest neighbor (Lowe, 2004). (See Fig. 1, bottom row.)

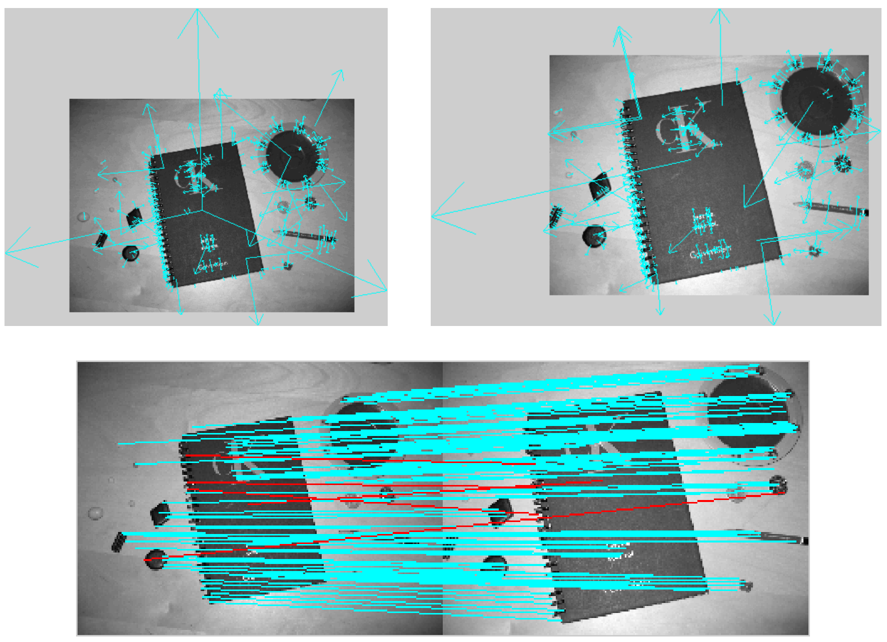


Figure 1: First row: SIFT keypoints for two different images of the same scene. Keypoints are displayed as vectors indicating location, scale, and orientation. Bottom row: Keypoint matches for the two images. Out of 459 keypoints in the left image and 566 keypoints in the right image, 177 matches were made. Knowing the setting of the scene, we were able to use RANSAC and find the outliers (wrong matches). Only 5 outliers were found, representing 2.8% of all matches.

In the same paper Lowe describes SIFT application for recognition of small or highly occluded objects. Many false matches may arise from the background; therefore, it is recommend to identify objects by clustering in pose space using the Hough transform.

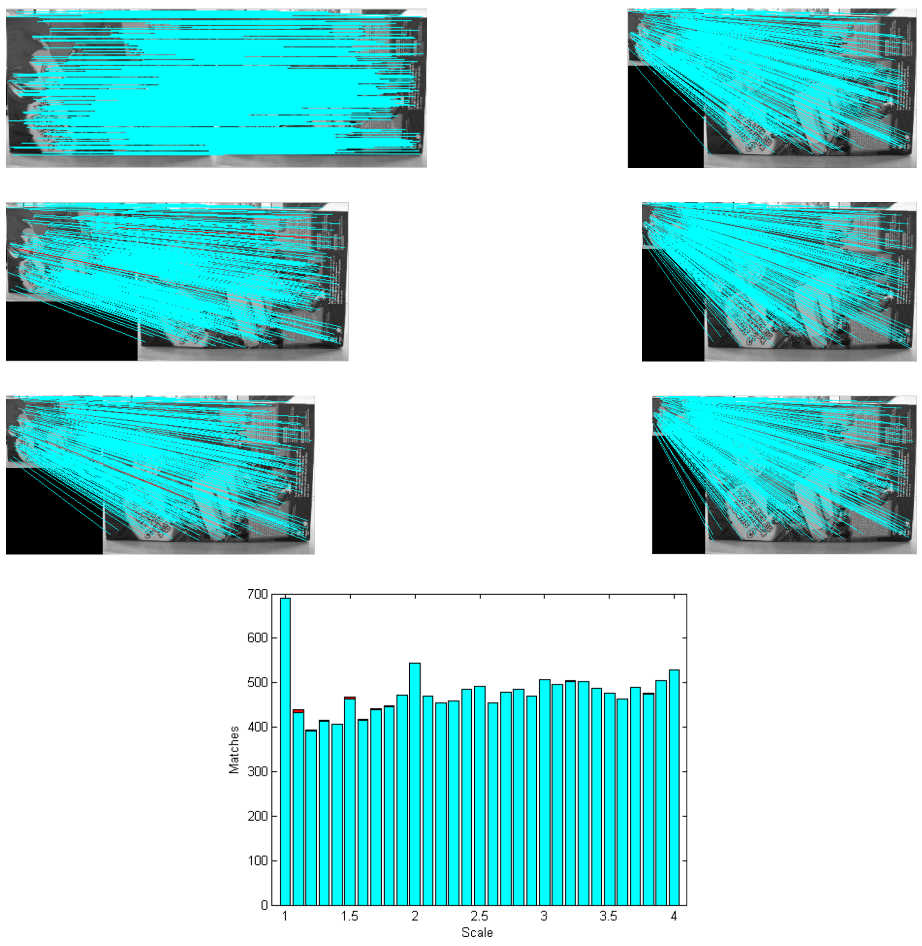


Figure 2: Scale testing. From top left, scales at 1:1, 1:1.6, 1:2.2, 1:2.8, 1:3.4, 1:4. Bottom row, bar plot depicting the comparison of results.

3 Performance: Testing SIFT

We tested SIFT's performance in a series of controlled tests. A testing image was matched against itself, yet modified by various transformations. Knowing the applied transformation, we were able to sort out possible false matches. A match was labeled false if the matched keypoints were at the distance of more than 2 pixels. We tested scale and rotation invariance, robustness to projective transformations and given the presence of noise. The results of each test are presented in bar plots, with the red tips of bars indicating the number of false matches. The size of the original (left) image is always 320×240 pixels.

3.1 Scaling

We tested SIFT at a variety of scales, from 1:1 to 1:4. Shown in Figure 2 are the results of keypoint matching at a variety of scales, depicting the number of correct and false keypoint matches.

In the lowest row of the figure, we present the comparison of the number of keypoints matched for each scale we tested. The highest number of matches, not surprisingly, was for 1:1 scale, but in general the number of matches does not change dramatically at other scales. The number of false matches never exceeds 2%.

We can conclude that SIFT is indeed scale-invariant.

3.2 Rotation

We also tested SIFT's invariance to rotation. Figure 3 presents the results of keypoint matching when the target image was rotated 15, 30, 45, 60, 75, and 90 degrees, and gives the true and false keypoint matches.

The bar graph illustrates the comparison of the number of keypoints matched for each rotation we tested. There was never more than 1% false matches. In the worst case (60 degrees), 4 out of 508 matches were false. SIFT otherwise clearly favors rotations of 90 degrees, those being a rotations in which interpolation plays no role. We can therefore conclude that SIFT is also rotation-invariant.

3.3 Projectivity

SIFT's robustness was tested under various projective transformations; the testing image was shrunk and one of it's sides scaled by factors of from 1 to 0.3, as illustrated in Figure 4.

The number of matching keypoints falls steadily with decreasing p, partly because the area of the warped image falls with p^2. Results included few outliers: in the worst case 11%, on average 3%.

We concluded that, even though the number of matches drops steadily, we can still rely on SIFT to match keypoints correctly; this confirms SIFT's robustness to projective transformations.

We conducted another experiment (Fig. projectivity scaled), in which we wished to eliminate the size discrepancy between the two images, to learn whether this discrepancy was what was affecting the results. In this second experiment, we sized the target figure so that its area was the same as the original. We noted that although the number of matched keypoints still dropped dramatically, the drop was not as drastic as

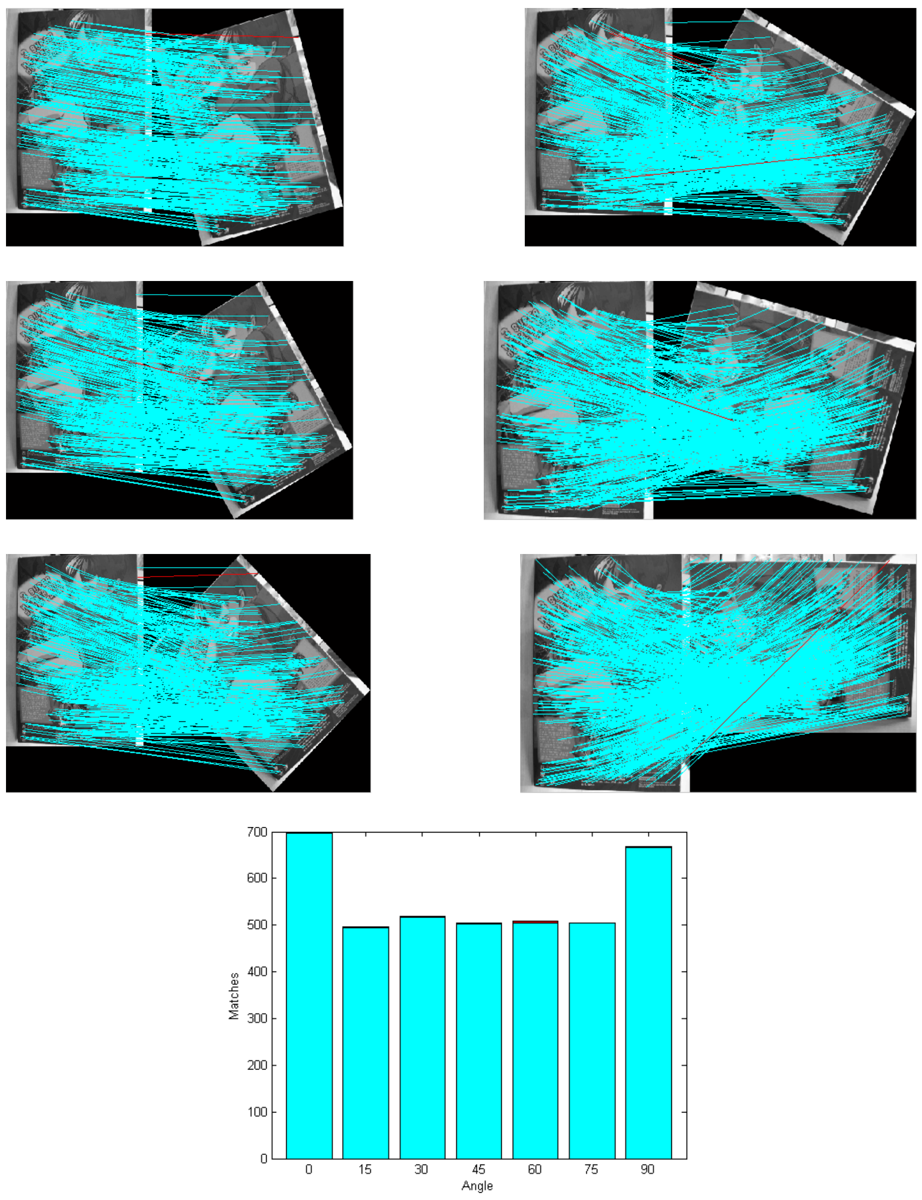


Figure 3: Rotation testing, $angles = 15, 30, \ldots, 90deg$. Percentage of outliers never over 1. Worst case: 4 false matches per 508 true matches. Original image: 320×240 pixels. Distance threshold: 2 pixels.

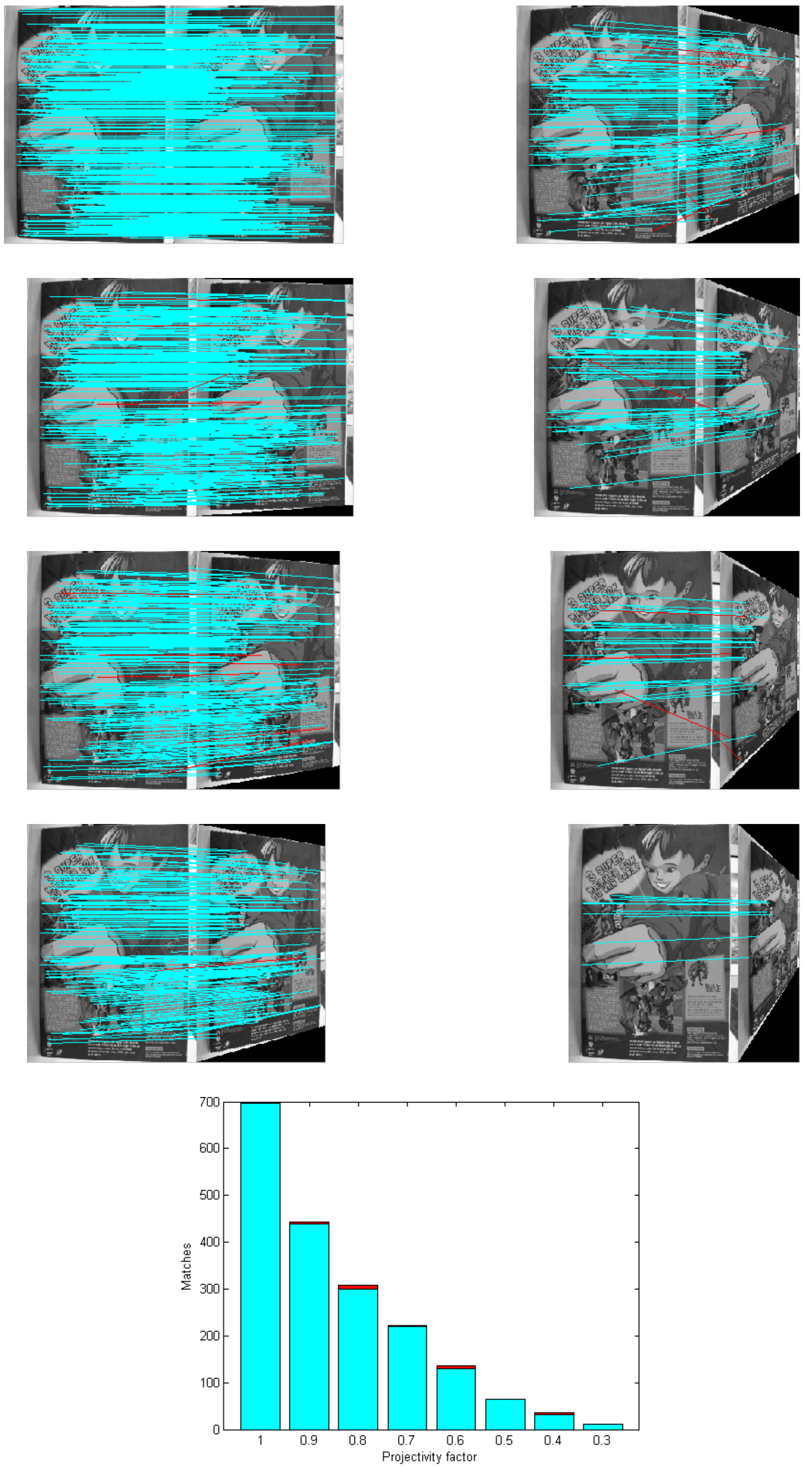
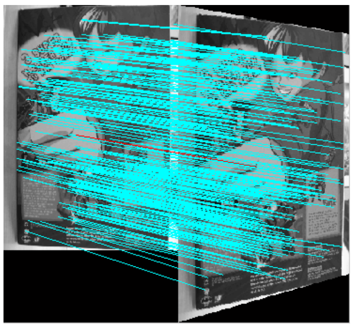
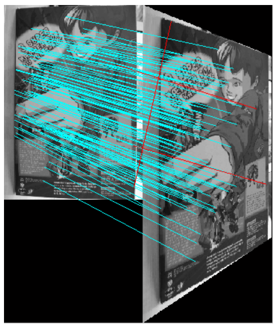


Figure 4: Projectivity testing, $p = 1, 0.9, /ldots 0.8$. There was never a high percentage of outliers—in the worst case 11%, on average 3%. The number of matching features falls, but the area of the warped image falls with p^2. Original image: 320×240 pixels, the area of the smallest warped image approx. $1/5$ of the original size.





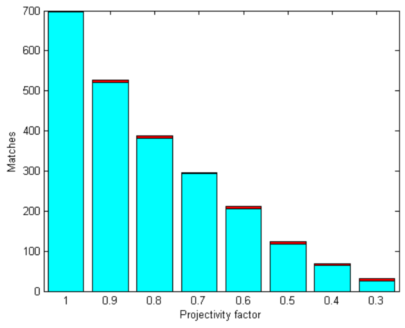


Figure 5: Projectivity testing, continued. Similar experiment to the previous, but the target image is scaled so that its area is the same as that of the original. We provide only two examples here, with $p = 0.7$ and $p = 0.5$, but results for all tests are provided in the bar graph.

in the first experiment. We can therefore conclude that the projectivity will influence the number of keypoints matched, but not the correctness of the matches.

3.4 Noise

We also tested SIFT's robustness given the presence of noise. An image is matched against itself, with 5% Gaussian noise added with each iteration, for 30 total iterations. In Figure 6, we show the keypoint matches for selected iterations.

After the first additions of noise, the number of keypoints matched drops significantly, but matches are still mostly correct. After 15 iterations, the drop in the number of matches slows, but false matches represent up to one quarter of all matches.

We can thereby conclude that SIFT is reasonably robust given the presence of noise.

4 Application: Using SIFT

David Lowe, the developer of SIFT, describes its application for object recognition, and having to deal with a great many outliers. He proposes using the Hough transform for clustering.

We applied SIFT for determining stereo correspondence. We used SIFT to identify initial corresponding points between two views of the same scene. Knowing the geometric setting of the problem, and because we did not expect the presence of many outliers, we were able to use RANSAC (Kovesi, 2000, [1]) to determine the set of inliers and to estimate the transformation between the images. This framework allowed us to implement automatic stitching of panoramic images and automatic estimation of fundamental matrix for stereo view.

4.1 Automatic Image Stitching

We begin with a sequence of panoramic images that we would like to stitch together into one seamless composite. The images were taken from the same location, so the relationship between the images is determined by a homography (a projective transformation). At least four corresponding points are needed to determine the homography between the images.

SIFT produces a set of initial corresponding points, which are fed to RANSAC for fitting the homography. One of the images is then warped according to the homography (we used bilinear transformation when warping), and images are stitched together (we used the "hat" weighting function, ranging from 0 at the borders to 1 in the center, when stitching to obtain a smoother result).

Applying SIFT in this way results in fully automated image stitching. The only user input that may be required is to set the distance threshold for RANSAC. We set that threshold to a rather low value ($t = 0.01$, corresponding to a couple of pixels), which possibly eliminates a few of the inliers, but at least we could be rather certain not to have any outliers sneaking in.

The number of matches returned by SIFT varies depending on the size of the overlapping area, but the percentage of inliers is always large enough for RANSAC to estimate a homography that results in a satisfying stitching. In Figures 7 and 8 we display both an example of many matches and one of relatively few matches.

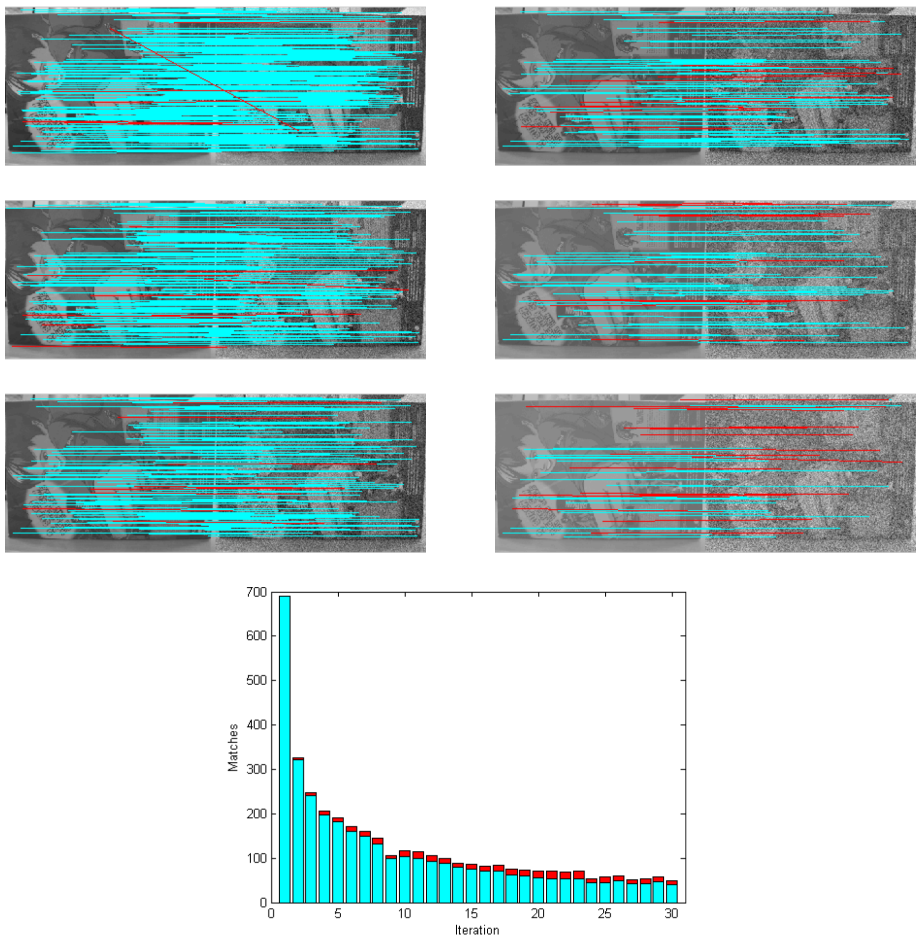


Figure 6: Noise testing. An image is matched against itself, with 5% Gaussian noise added to one of them for every iteration. From top left, iterations 2, 4, 6, 10, 15, 25. The original image appears to gradually lose contrast, but that is caused by the displaying function scaling both the original and target images together for each iteration.

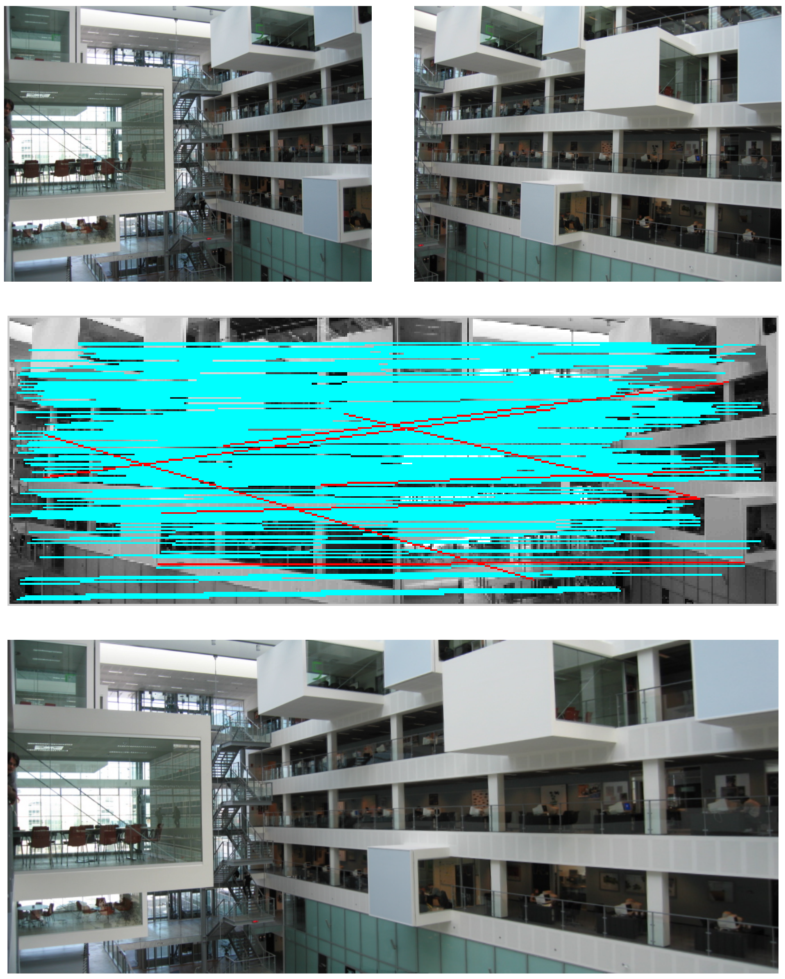


Figure 7: Stitching two images of the ITU building: Original images, SIFT matches (with outliers eliminated by RANSAC shown in red), and final stitch. Size of original images: 480×640 pixels. Size of the final image: 480×1110 pixels. There were 279 matches, with 270 inliers.

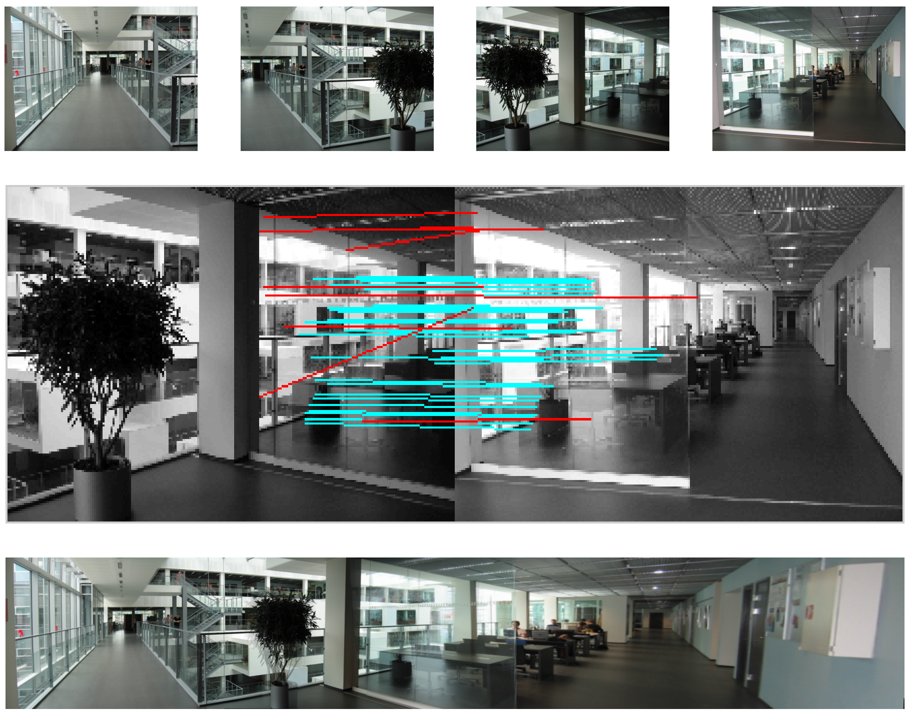


Figure 8: A mosaic composed of four individual images. The middle row shows the keypoint matches found for images three and four: only 46 keypoint matches, with 36 inliers, were found, but the mosaic could still be completed successfully. Size of the final image: 469 $\times$ 2796 pixels.

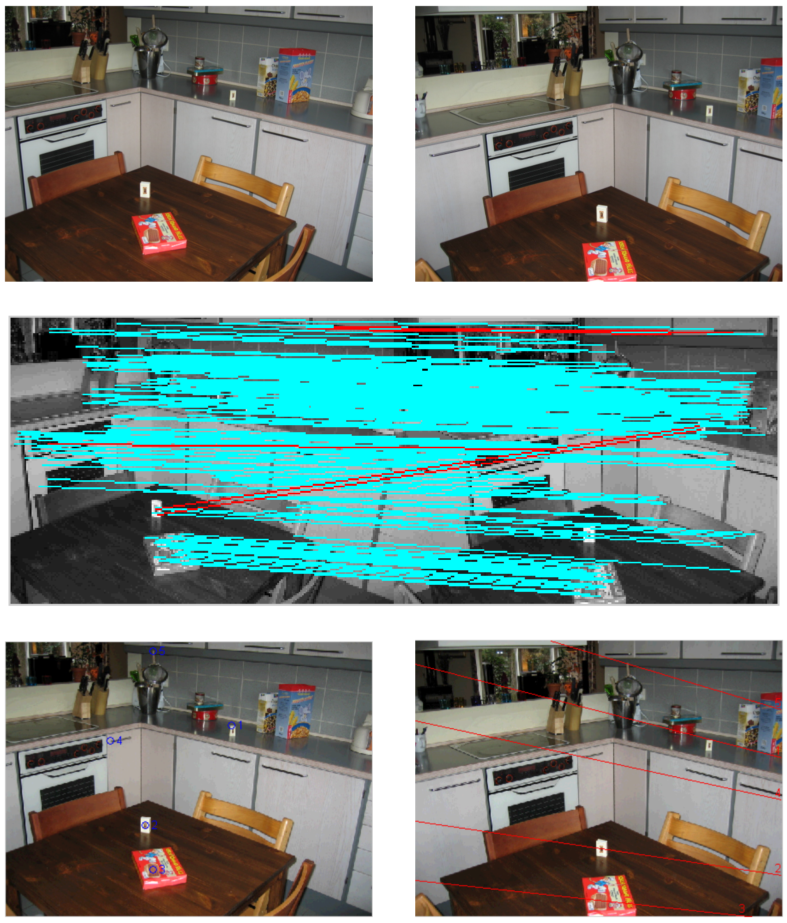


Figure 9: SIFT found 891 and 1028 keypoints in the left and right images, respectively, which resulted in 274 matches. Applying RANSAC then produced a set of 269 inliers. At the right of the figure are original images, with the set of testing points and corresponding epipolar lines.

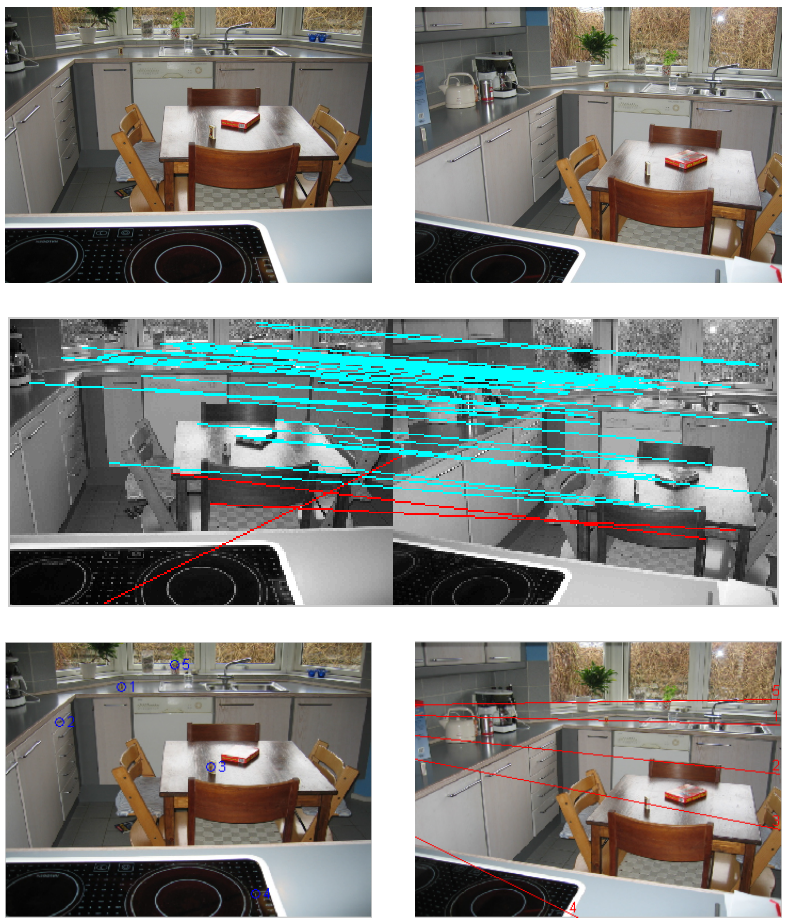


Figure 10: SIFT found 927 and 1176 keypoints in the left and right image respectively, which resulted in only 41 matches. RANSAC eliminated only 3 outliers, leaving a set of 38 inliers.

4.2 Automatic Fundamental Matrix Estimation

The fundamental matrix is essential for two-view geometry; it describes mapping between points in one image and corresponding epipolar lines in another image. At least eight point correspondences are needed to estimate the fundamental matrix. As in the previous example, SIFT produces the initial corresponding points, and RANSAC is then used fit the fundamental matrix.

Shown here in Figures 9 and 10 are two examples of applying SIFT in automatic fundamental-matrix estimation. In the first example, many keypoint matches were returned by SIFT. The second example resulted in fewer, but still accurate, keypoint matches.

References

[1] P. D. Kovesi. Matlab and octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from : <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[2] David G. Lowe. Distinctive image features from scale - invariant keypoints. *International Journal of Computer Vision*, 2004.