Internal Report 2012-2013-10 June 2013

# Universiteit Leiden

# Opleiding Informatica

Robustness in Salient Point Detection:

a Comparison of SIFT and SURF

Umut Özaydın

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

# Robustness in Salient Point Detection: a Comparison of SIFT and SURF

Umut Özaydın
uozaydin@liacs.nl
Leiden Institute of Advanced Computer Science
Niels Bohrweg 1, 2333CA, Leiden, The Netherlands

**Abstract**

Salient point algorithms are one of the most prominent methods used in computer vision. Many salient point detection algorithms exist today, two of which are SIFT and SURF. The aim of this paper is to test and compare these two algorithms, trying to measure the repeatability, stability and scale and rotation invariance. This is done by doing experiments with both algorithms and comparing the sets of salient points found.

## 1 Introduction

Computer vision is a field in computer science that tries to analyze and interpret images. Computer vision has many applications, some of which are object matching, augmented reality (AR) and facial recognition. Salient point detection is one of the methods used in computer vision. Salient point detectors analyze images and try to find special points of interest that can be used to match other keypoints, as introduced by Schmid and Mohr [8]. An important attribute of salient point detectors is *robustness*. Robustness is the ability to maintain a good performance under different situations. As humans, we can recognize objects regardless of the perspective and lighting and other variable conditions. A salient point detector that keeps up its performance under these changes is considered to be robust.

Salient point detection can, for example, be used in AR to replace markers, that would otherwise be necessary for the computer to be able to ”see”. In markerless AR, with the use of salient points, the natural environment can be used to do the tracking. The images that are overlaid onto the real world need to integrate into the environment, and therefore the tracking needs to be done precisely, as images might otherwise *shake* or *jitter*. Robustness of salient point detectors is therefore very important. Yuan proposes a method for using salient point detection in AR [11].

Two fairly recent methods of salient point detection are Scale Invariant Feature Transform (SIFT) [6, 7] and Speeded Up Robust Features (SURF) [2, 3]. This project will focus on these two algorithms and aim to compare the robustness they offer.

**Outline** First the criteria for comparing the algorithms and different situations that can occur in the real world will be defined in Section 2. Then, in Section 3, the SIFT and SURF algorithms will be explained. The way the experiments were performed will be described in Section 4. Finally, the results will be analyzed in Section 5 and discussed in Section 6.

# 2 Criteria

Being able to perform under different kinds of situations is important, because the conditions will never be exactly as they are desired to be. In other words, a salient point detection algorithm needs to be robust.

First of all, if objects appear in the same position, salient points need to appear in the same positions as well, every single time, without suddenly appearing, disappearing or moving too much from its original position. Of course, not every keypoint has to stay perfectly stationary, as a match is found if the number of corresponding salient points exceeds a certain threshold. The more corresponding points, however, the better the algorithm.

Second, when objects move, rotate or resize, the salient points need to adjust accordingly. When matching two objects, their sizes will rarely ever be exactly the same. Therefore, scale invariance is a very important attribute. The same goes for rotation. However, there are three types of rotation: yaw, pitch and roll. Yaw rotation occurs when rotating around the axis that is directly looked upon. When looking in the $z$ direction, and onto the $xy$ plane, the $z$-coordinates stay the same, as the $x$ and $y$-coordinates change while rotating. In pitch rotation, the object rotates around the $x$-axis, rotating forward and backward. In roll rotation, the object rotates around the $y$-axis and, as the name suggests, rotates (or rolls) sideways. In this paper, pitch and roll rotation will be considered equivalent, as it's mainly a matter of perspective (or changing the orientation of the object). Salient point algorithms should be yaw rotation invariant, as an object rotated in this manner practically looks the same, except for its orientation. The other two types of rotation are much more tricky however, because these rotations change the appearance of the object. Salient point algorithms should be partially invariant with respect to these transformations.

Third, objects can possibly be seen under different lighting situations, and therefore salient point detectors should also work well under differing lighting conditions and contrast.

And finally, important for real-time applications using salient point detection, the efficiency of the algorithms matters. When tracking the environment for augmented reality, for example, the performance of the algorithm is of utmost importance to prevent the projected images from lagging.

Note that in this project, only the keypoint detection is tested, but not feature description.

# 3 The Methods

## 3.1 Scale Invariant Feature Transform

The creators of the SIFT algorithm claim it is invariant to image scale and rotation and it works well under different circumstances, such as added noise and change in illumination or 3D viewpoint. The SIFT detector consists mainly of three parts: scale-space extrema detection, keypoint selection and orientation assignment.

When looking at objects in the small scale, details are important. When looking at objects in the larger scale, these details are less significant and the big picture is what's important. The SIFT algorithm looks at the details, as well as the big picture by building a scale space of the image and incrementally removing more and more details. The only possible way of removing these details is using the Gaussian kernel [4,5]. The original image is increasingly convolved with the Gaussian function, which results in an octave, consisting of different scales. The image is then resized and again blurred increasingly, which results in another octave. This is done several times, which results in the scale space.

To approximate the Laplacian of Gaussian, which can be used to locate edges and corners, the difference-of-Gaussian (DOG) is calculated. This simply means calculating the difference of each consecutive scale. Now, instead of applying a complex Laplace operator, a simple subtraction is used. If an octave contains $n$ scales, it results in $n-1$ DOGs, as seen in Figure 1.

Locating extrema is now rather simple and is done by comparing each point to its 26 neighbors, eight in the scale itself, and nine in both neighboring scales (as shown in Figure 2). A point is

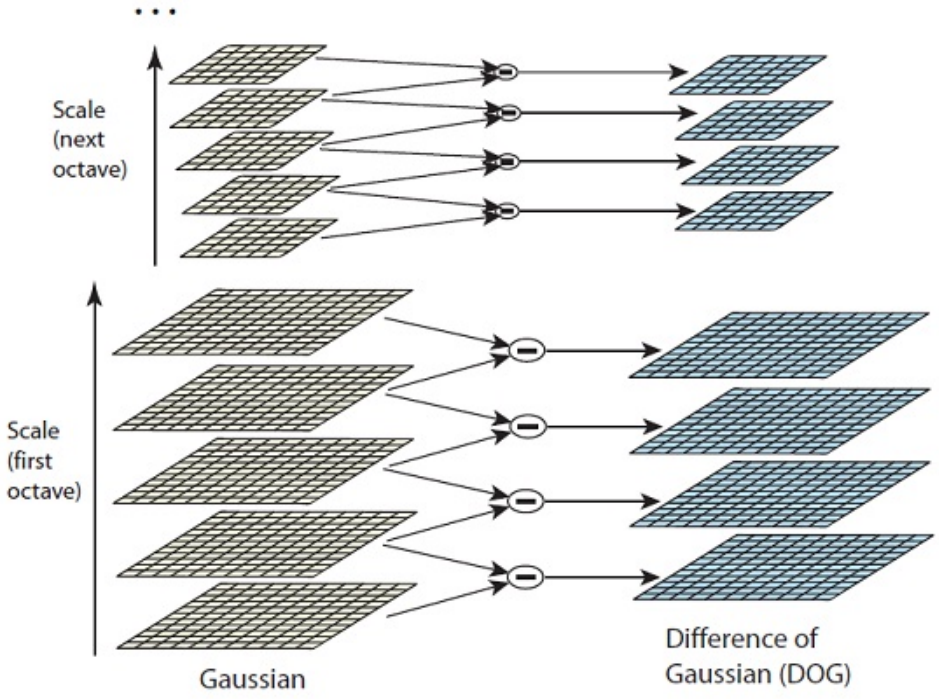


Figure 1: For each octave, the difference-of-Gaussian is calculated and this results in one image less than the amount of scales.

considered a maximum if it's bigger than all its neighbors, and a minimum if it's smaller. The two images at the ends that have only one neighboring scale do not itself participate in this process. That means that if an octave has $n$ scales only $n-3$ images are used to find extrema.

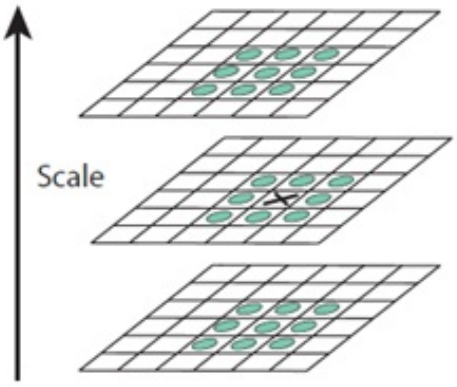


Figure 2: Extrema are found by comparing each point to its 26 neighbors.

With this method, many candidate keypoints are found. However, not all of them are useful and therefore need to be filtered out. First of all, points whose contrast is too low are eliminated, as these points tend to be sensitive to noise. To do this, their intensity is checked. If it's below a certain threshold, they're removed. Secondly, points that lie on edges need to be removed, as these are also undesirable as keypoints. If a certain point lies along an edge, it has one large and one small principle curvature. This is indirectly calculated with the ratio between the eigenvalues

of the Hessian matrix.

To ensure rotation invariance, an orientation is assigned to each keypoint. To determine the orientation of a keypoint, its gradient is generated, with an area equal to the point's scale. All orientations are then put into bins and their magnitudes are added up. The orientation assigned to the keypoint is the orientation with the greatest magnitude. If other orientations have a magnitude more than 80% of the largest magnitude, new keypoints are added with the same location and scale, but with the corresponding orientations.

## 3.2 Speeded Up Robust Features

Instead of using difference-of-Gaussians to approximate the Laplacian values, the SURF algorithm uses box filters (Figure 3). This means the algorithm can work with integral images [10], which should highly increase performance.

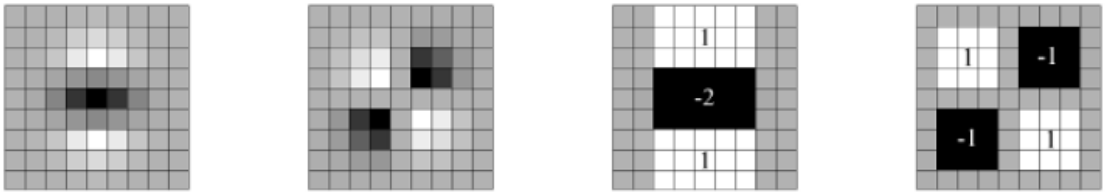


Figure 3: Gaussian second order partial derivatives (left) and the approximation with box filters (right).

The scale space is not, as in the SIFT algorithm, constructed by incrementally scaling the image down, while progressively convolving with the Gaussian kernel, but by scaling the filter size up for each scale and octave (Figure 4). The 9x9 filter in Figure 3 is considered to be the initial filter, and the first octave is obtained by increasing each scale by 6 pixels, which results in the filters of sizes 9x9, 15x15, 21x21 and 27x27. In each octave the step sizes is doubled. The second octave consists of filters of sizes 15x15, 27x27, 39x39 and 51x51. The third and fourth octave are constructed in the same way. Because the difference in filter size is rather large (1.7 times from 9 to 15), a second scale space is used. The original image is doubled in size, and the first octave starts with a filter of size 15x15 and incrementally increases the size by 6 pixels. Then the second octave is obtained with scale steps of 12 and then the third and fourth octaves are produced in the same fashion. Now the difference in filter size is smaller (1.4 times from 15 to 21).

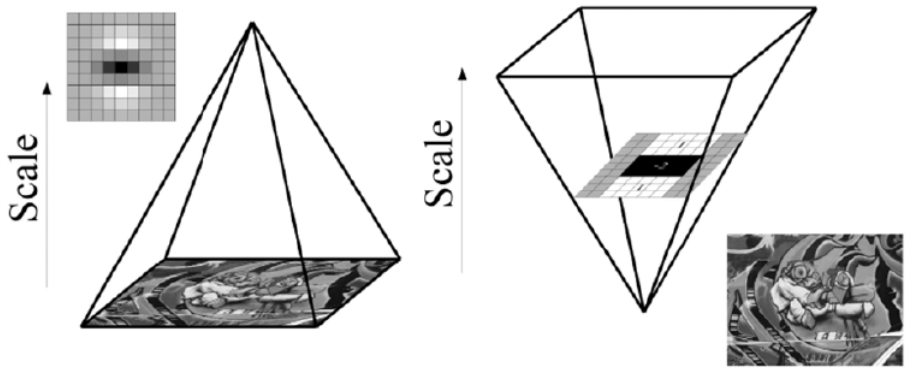


Figure 4: The image is not scaled down for every scale, but instead the filter size is scaled up.

To find interest points, the same technique as in SIFT is used. Every pixel, above some

threshold, is compared to its 26 neighbors (as shown in Figure 2). Those points that are smaller or larger than all neighbors become minima and maxima (respectively). The higher the threshold, the less points are found, but the better the keypoints. The found points are then interpolated in scale and space, to increase accuracy.

The final step is to assign an orientation to the found points. Around each point, the Haar wavelet responses of size $4s$ are calculated in $x$ and $y$ directions in a radius of $6s$. Here, $s$ is the scale of the interest point. Then, the Gaussian function is applied to the responses, centered around the keypoint. The responses are then represented as points in vector space. A small window of size $\frac{1}{3}\pi$ is then rotated around the circle, in which the sum of the points is calculated. The orientation of the keypoint is then decided by the window where this sum is the largest (see Figure 5).

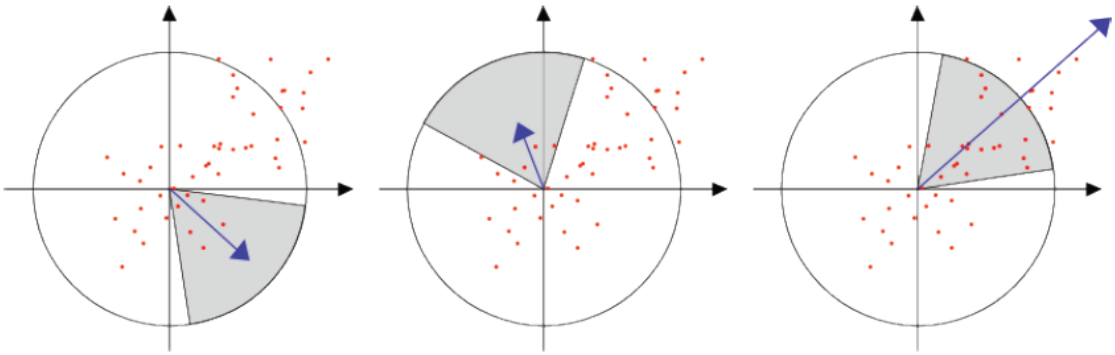

Figure 5: The Haar wavelet responses are represented as points. A window is rotated, in which the points are summed up into a new vector. The largest vector decides the orientation.

# 4 Test Setup

To test the SIFT and SURF algorithms, five objects have been recorded. Once in a properly lit environment and once in a darker environment. The objects recorded are a book, a comb, a cap (of a bottle), a hand and an iPod case (see Figure 6). The objects and camera were moved as little as possible to get the most reliable results. From each video eight seconds of footage was used for the tests. Five attributes were tested: repeatability, stability, scale invariance and two types of rotation invariance (yaw and pitch).

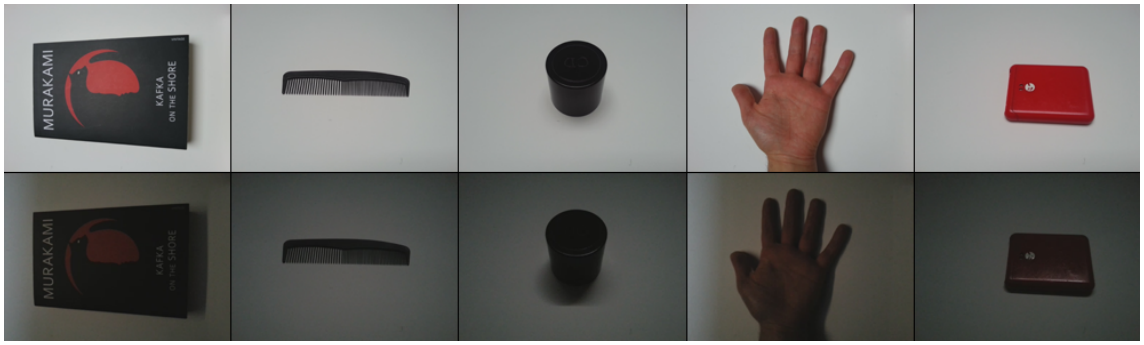


Figure 6: The five objects used for testing (from left to right): a book, a comb, a (bottle) cap, a hand and an iPod case, both in a lit environment (top) and unlit environment (bottom).

## Repeatability

An important attribute of salient point detection is the *repeatability rate* [9], which is defined by the amount of points that appear in the same parts of two images, or, in other words, are repeated. The location at which a point should appear can change due to changes in camera position and angle. In this case, however, the camera and objects remain stationary and all points should stay in the same location. For each video, every frame was compared to the previous frame and this resulted in a certain amount of *matches*, keypoints that were repeated, and *errors*, keypoints that either appeared, disappeared or changed more than the set margins. The margins chosen were 2 pixels for $x$ and $y$-coordinates, 2 pixels for size and 5° for orientation.

## Stability

Another important attribute of the algorithms is how much keypoints move throughout the frames. To get a measure of this, for each keypoint in each frame, the nearest keypoint in the previous frame was measured. From all these minimum distances, the mean distance per keypoint and standard deviation were calculated.

## Scale invariance

If the algorithms are scale invariant, keypoints should stay the same for different scales of the images. Each frame was therefore scaled up to 1.1, 1.2, 1.3 and 2 times the original size and scaled down to 0.9, 0.8, 0.7, 0.5, 0.25 and 0.125 times the original size (see Figure 7). Actually, the images were scaled to $1.1^2$, $1.2^2$ etc. times the original size, as it was scaled with these values in both $x$ and $y$-dimensions. In the rest of this paper the $^2$ will be dropped for the sake of convenience. After scaling, the original image was compared to each of the resized images. Just like in the first test, repeatability was measured. However, this time the keypoint coordinates were not the same, but were calculated by multiplying with the appropriate scale. The same goes for the keypoint size, which is the diameter of the keypoint. The angle should stay the same throughout the scaling.

Figure 7: A comb, scaled 0.7, 1 and 1.3 times.

## Yaw rotation

Similar to the scale invariance test, each frame was rotated from 20° up to 340° (see Figure 8) and the original image was compared to each of the rotated images. This time, the $x$ and $y$-coordinates of the keypoints in the original image were rotated with the same angle as the rotated images. The size stayed the same and the angle of rotation was also added to the keypoint orientation. Again, the repeatability of the salient points was measured in matches and errors.

Figure 8: A hand, rotated 0°, 20° and 40° (yaw rotation).

### Pitch rotation

Testing pitch rotation is more difficult, without having images of which the angle of rotation is exactly known. Therefore, for this test, three objects were photographed from different angles, approximating 0°, 10°, 20°, 30°, 40° and 50° of pitch rotation (see Figure 9). To test if keypoints stay the same for these rotated images, the image without rotation was manually compared to the rotated images, searching how many keypoints were repeated in both images. The objects used for this test were a playing card, a rubber duck and a computer mouse.

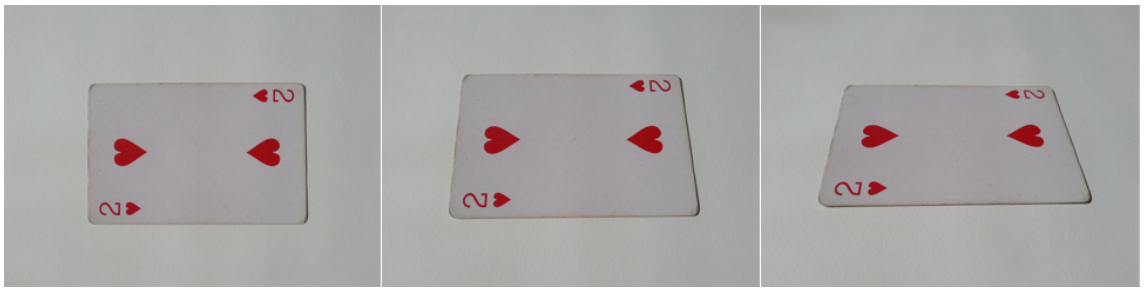

Figure 9: A playing card, rotated 0°, 20° and 40° (pitch rotation).

## 5 Test Results

The objects chosen all have different shapes, sizes and colors, and thus each object yields a different amount of keypoints. In general, the more keypoints are found, the better. The amount of keypoints found for each object are shown in Figure 10. It is immediately clear that the SIFT algorithm finds much more keypoints than the SURF algorithm. When the objects are properly lit, in all cases SIFT finds at least twice as much keypoints than SURF. In a badly lit environment, SURF seems to struggle in particular, while SIFT does it rather properly, and even does better than in a properly lit environment in some cases. Also notable is the fact that SURF is unable to find any keypoint at all for the cap, while SIFT does a decent job with about 5 keypoints per frame and 12 keypoints per frame in a lit and unlit environment, respectively. The cap does offer very little to these algorithms, as it is a very simple shape in a solid black color on a solid white background. This proves, however, how robust the SIFT algorithm actually is, as it still manages to find keypoints.

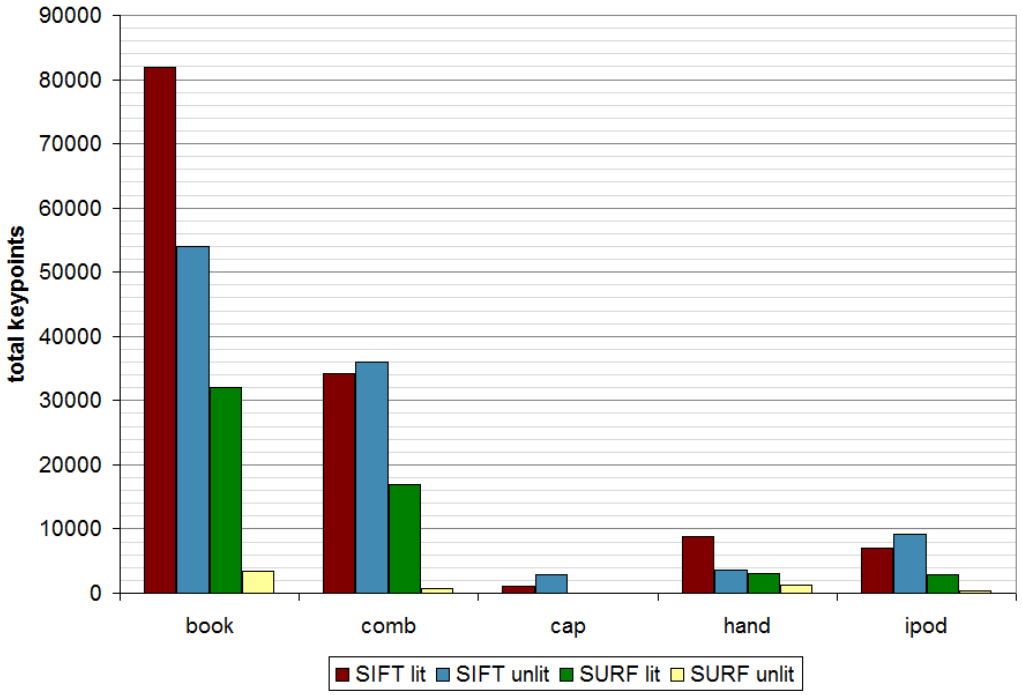


Figure 10: The total amount of keypoints (in all eight seconds of footage) found by the SIFT and SURF algorithms, both in a lit environment, and an unlit environment, for all five objects.

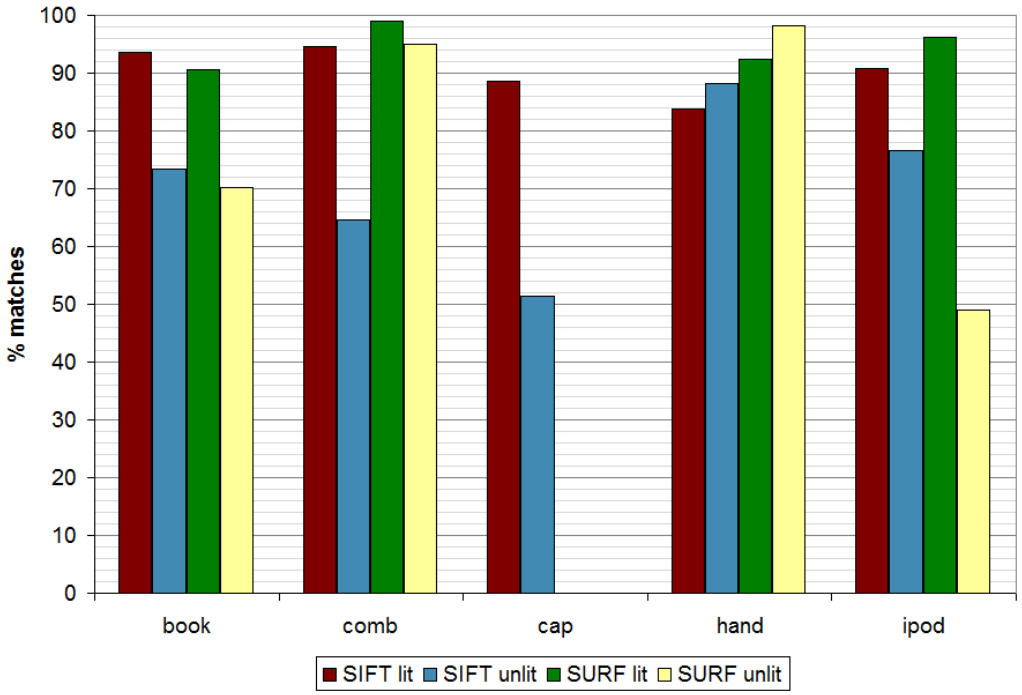


Figure 11: The results of the repeatability test for both algorithms, in lit and unlit environments. The amounts are the percentage of matches found in each consecutive frame.

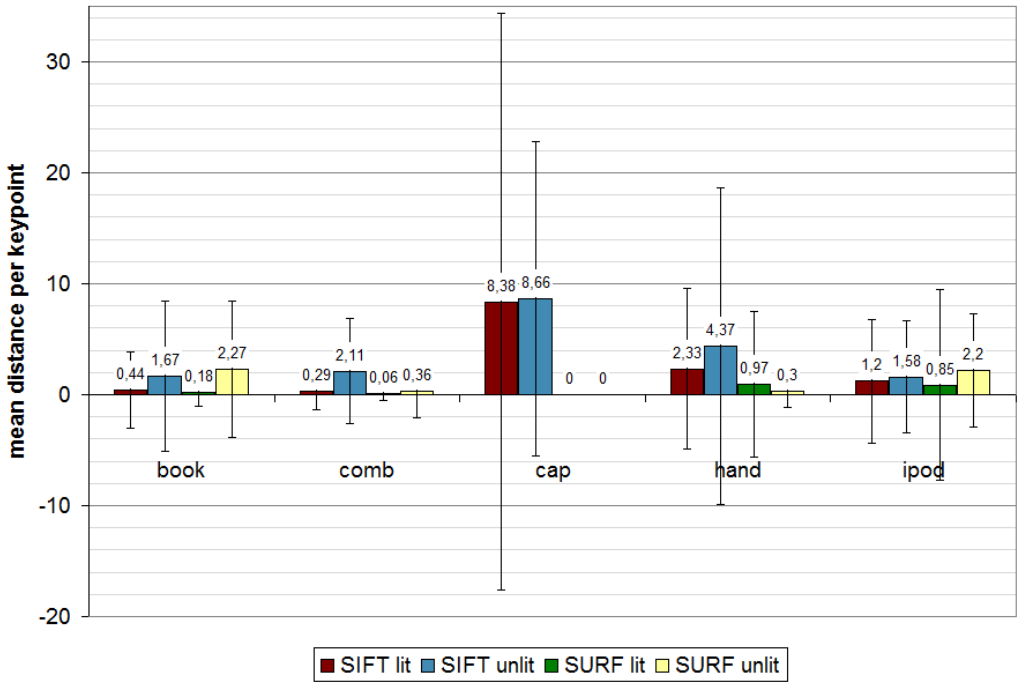


Figure 12: The second test results, testing stability. The amounts are the mean distance per keypoint, where for each keypoint in one frame the distance to its closest counterpart in the other is measured. The error lines show the standard deviation.

The results of the first test, measuring repeatability, are shown in Figure 11. Both algorithms seem to do rather well. It is obvious that objects in a lit environment seem to give better results than in an unlit one, except when the object is a hand. For a comparison of the two algorithms, there doesn't seem to be a clear winner. In some cases SIFT has more matches, and in others SURF has more. They perform quite equally for the most part.

For the second test, stability of keypoints was measured. The results are shown in Figure 12. Immediately noticeable is the fact that in all cases, the standard deviation is larger than the mean values. This is likely the result of some big outliers. In each frame, there is a certain amount of keypoints that pop up or disappear and these keypoints are the outliers if there are no other keypoint near. Again, a darker environment seems to negatively influence the results. As in the repeatability test, both algorithms perform relatively similar, although it seems that here, SURF performs a little better in general. Also note that, the more keypoints there are, the smaller the average distance will most likely be. Therefore, SURF's performance can be interpreted as slightly better, as there are less keypoints found.

For the next test, the scale invariance of the algorithms is tested. Figure 13 shows the total amount of keypoints found for each rescaled object. There seems to be a direct connection between the image size and the amount of keypoints found, which is not very surprising. Again, SIFT finds many more keypoints than SURF, which especially struggles in the unlit environment. Comparing the amount of matches of the two algorithms (Figure 14), it is clear that SIFT ensures better scale invariance than SURF. More than half the keypoints in the SIFT algorithm stay invariant for most objects with a scale larger than 0.5. Scale the images lower than that and the amount of keypoints found drops and so does the percentage of matches. The percentages of the SURF algorithm are all over the place. The results aren't very consistent, and it could be said that SURF cannot guarantee as much scale invariance as SIFT can.

The next test is aimed at measuring yaw rotation invariance. The total amount of keypoints

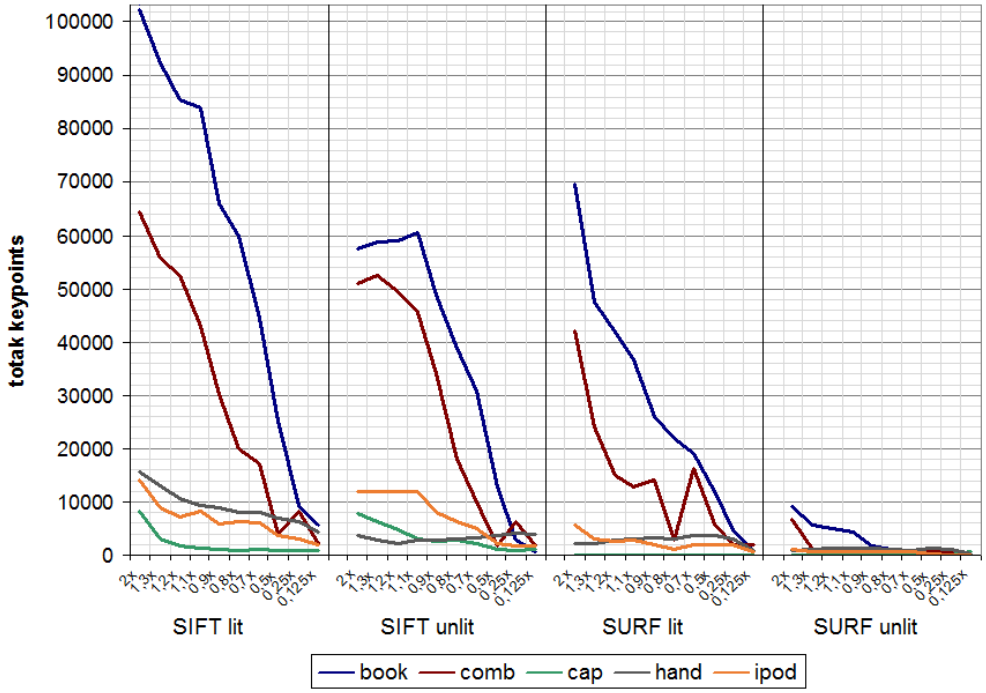


Figure 13: The total amount of keypoints for the scaled images. The images are scaled from 2x the original size to 0.125x the original size. The original size is 640x480 pixels.

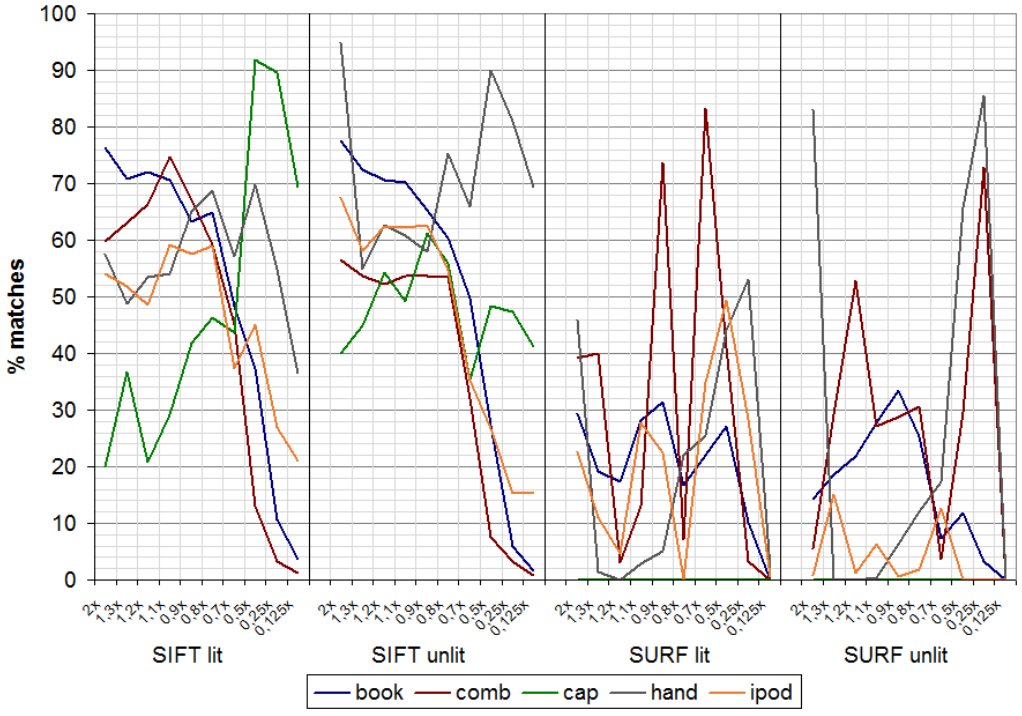


Figure 14: The percentage of matches in the scaled images compared to the original image.

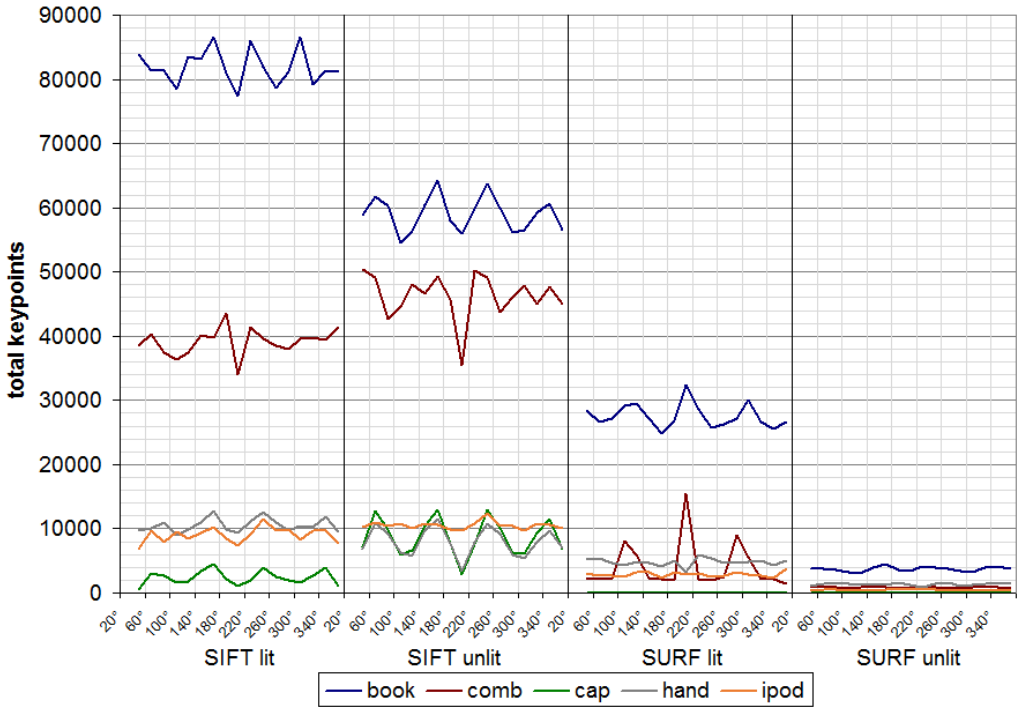


Figure 15: The total amount of keypoints for the rotated images. The images are rotated per 20° around the $z$-axis.

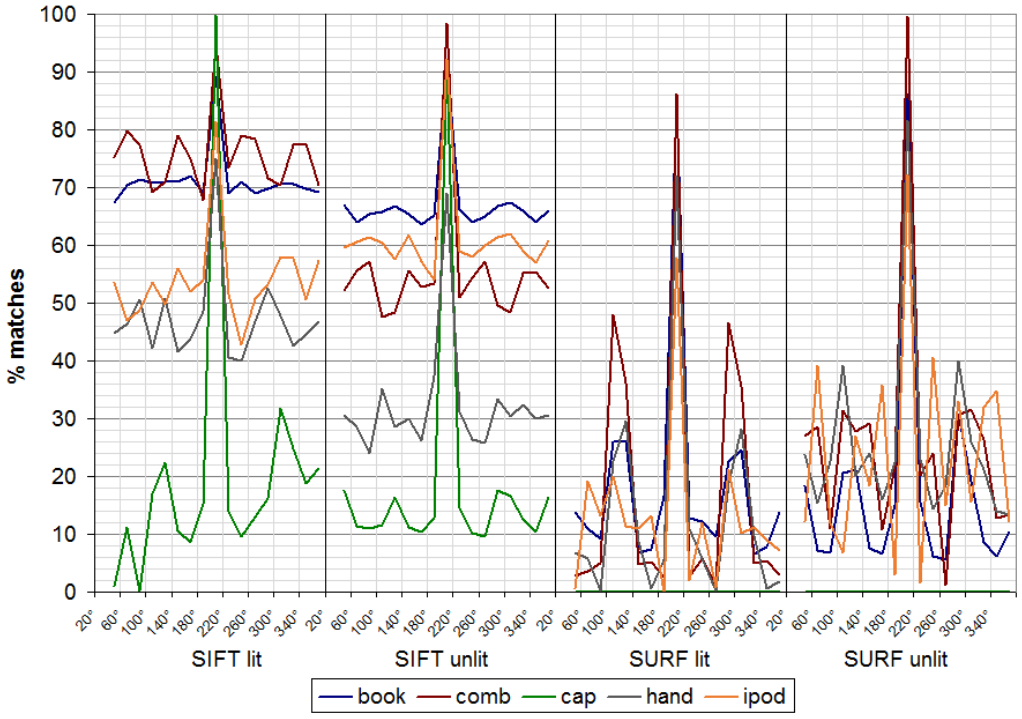


Figure 16: The percentage of matches in the rotated images compared to the original image.

found in each rotated image are shown in Figure 15. Unsurprisingly, all rotated images yield approximately the same amount of keypoints as the original image. Looking at the percentage of matches, in Figure 16, immediately noticeable is the highest peaks in the middle of the graphs. These are the images that are rotated 180°. For SIFT, other noticeable, but smaller, peaks are around 45°, 90°, 135°, 225°, 270° and 315°. In other words, with every step of $\frac{1}{4}\pi$. SURF does well at 90° and 270°, but not very well at 45°, 135°, 225° and 315°. This might be attributed to the fact that the shape of the filters used in SURF is square [2]. SURF dips very low in between these angles (in some cases even near 0%), whereas SIFT stays generally more constant throughout the rotation. And again, the amount of matches in the SIFT algorithm are much higher than those of SURF. It seems obvious that, once again, SIFT has the upper hand.

In the final test pitch rotation invariance is measured. This time, instead of 232 images (8 seconds, 29 frames per second), there is only 1 image per angle. Figure 17 shows how many keypoints were found in the images. The card gave a good amount of keypoints, but the other two objects, the rubber duck and computer mouse, unfortunately yielded very little keypoints; too little to be useful for a comparison. The SIFT algorithm did find around two dozen keypoints for these objects, so it can still be interesting to evaluate. Figure 18 shows the amount of matches found for each object. In the SIFT algorithm, a clear decline in matches can be seen as the angle gets bigger, as would be expected. The mouse however doesn't do very well, as there are few matches found at an angle of 10°, and none with a bigger angle. SURF also shows a steady decline in matches for the card, except an increase in matches between 10° and 20°. As it is only a single case, not much can be said about this. Most likely it is not a pattern, but an isolated case.

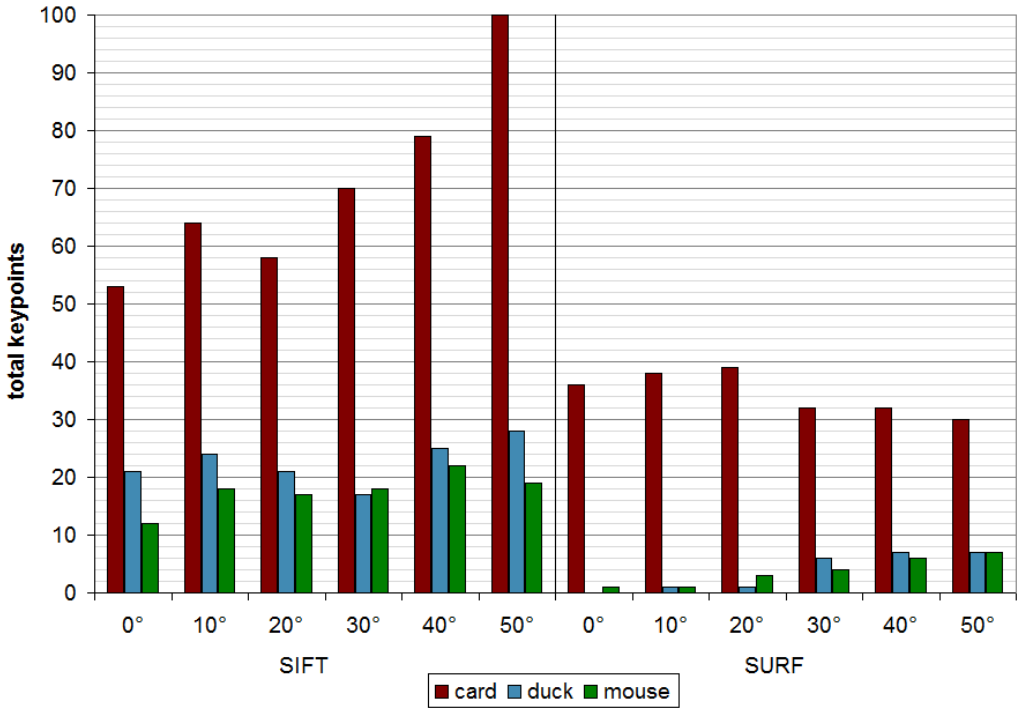


Figure 17: The total amount of keypoints for the rotated images. The images were taken from a 0° angle up to a 50° angle, with intervals of 10°. The images are rotated around the $x$-axis

Finally, the execution time of both algorithms is compared. Both algorithms were run on all objects, with only keypoint detection and without displaying keypoints. The average execution times can be seen in Figure 19. $system1$ is a few year old laptop, $system2$ is a rather new computer, with a relatively fast CPU. On $system1$, SURF runs almost 2.3 times faster than SIFT. On $system2$, SURF runs almost 1.5 times faster than SIFT.

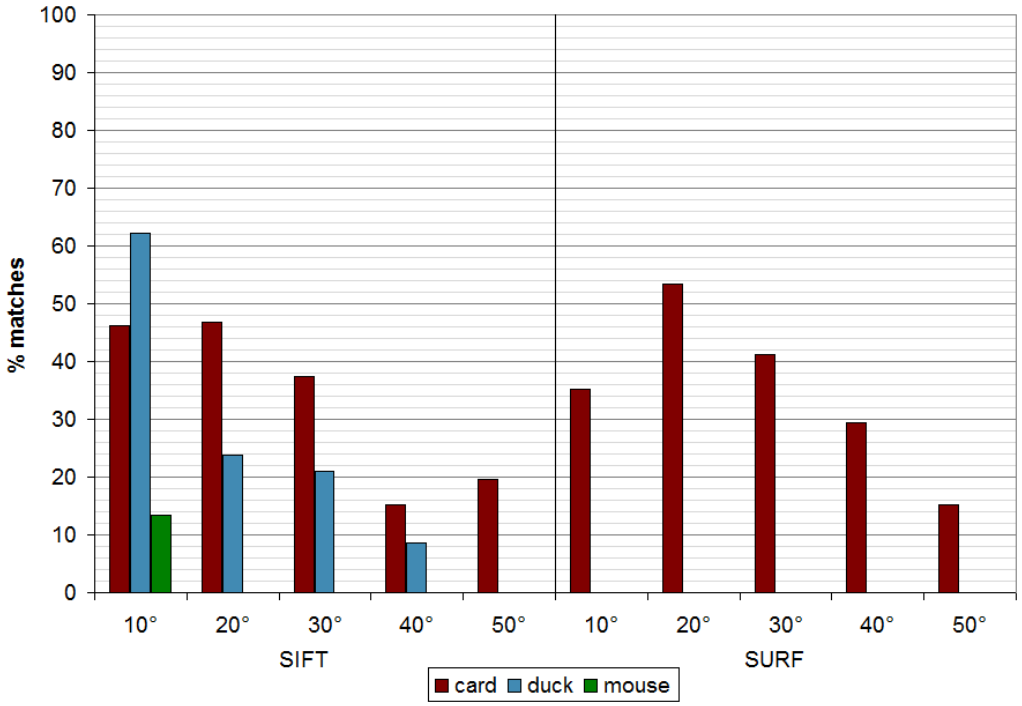


Figure 18: The percentage of matches in the rotated images compared to the original image.

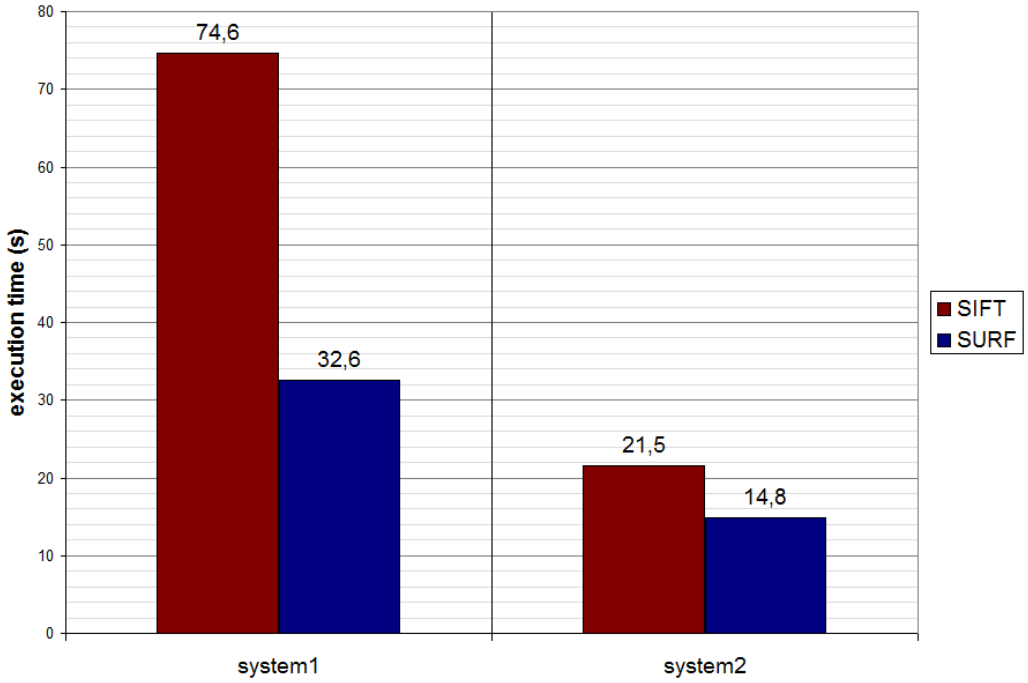


Figure 19: The average execution time of both algorithms. All objects were tested on two different computers and these are the average computation times in seconds.

# 6 Discussion

In general, SIFT seems to outperform SURF when it comes down to robustness. SIFT seems to be more scale and rotation invariant than SURF, while on repeatability and stability both perform quite similarly. One advantage of SURF over SIFT, however, is that computationally SURF is at least 1.5 times faster. This is a highly desirable trait in real-time applications and because of this, even though SIFT might perform better, in these applications SURF might be preferable. With only keypoint detection, on a fast system, SIFT yields a frame rate of 10.8 frames per second and SURF yields a frame rate of 15.7 frames per second.

One of the additional things that could be done, with regards to testing the algorithms, is comparing the images from the lit environment with the images from the unlit environment, which has not been done in this project, where only the difference in performance between working in a lit environment and unlit environment was tested. This is relevant, because when matching objects in two images, the images could have completely different lighting conditions. Algorithms should still perform well under such conditions.

Something to point out in the tests that were done, is that in finding a match between keypoints, the $x$ and $y$-coordinates, sizes and rotations were compared. The margin for the size difference was a set value of 2. In reality, it might have been better to do a relative comparison, meaning a bigger margin for bigger keypoints and a smaller margin for smaller keypoints. For this reason, some keypoints may have been rejected (counted as errors), while they were actually similar enough to be counted as a match.

As noted before, getting pitch and roll rotation invariance is much more difficult. This is mainly because, when performing such rotations, there are parts of objects that become visible that were not visible before, and parts that were once visible now are not. This makes matching the objects a challenge, especially when the angles of rotation get bigger. Another attribute that influences results, as became clear in the fifth test, is the reflectiveness of objects. When rotating, part of the keypoints seemed to follow the reflection, instead of the object itself. This obviously influenced the results in a negative way. The object (a computer mouse) was very slightly reflective. The effect will even be bigger when objects are more reflective. This might be important, because the real world is filled with reflective objects. When trying to perform augmented reality, reflective objects could possibly corrupt the results. A possible solution for this problem might be trying to recognize reflections, like proposed in [1].

Both algorithms have their respective advantages and disadvantages. SIFT is more robust, while SURF is faster. Depending on the application, either algorithm might be preferable to the other.

# References


[1] AHMED, M. A., PITIE, F., AND KOKARAM, A. Reflection detection in image sequences. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition* (2011), pp. 705–712.

[2] BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. Speeded-up robust features (SURF). *Computer Vision and Image Understanding 110*, 3 (2008), 346–359.

[3] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. SURF: Speeded-up robust features. In *Proceedings of the ninth European Conference on Computer Vision* (2006), pp. 404–417.

[4] KOENDERINK, J. J. The structure of images. *Biological Cybernetics 50*, 5 (1984), 363–370.

[5] LINDEBERG, T. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics 21*, 2 (1994), 225–270.

[6] LOWE, D. G. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision* (1999), pp. 1150–1157.

[7] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[8] SCHMID, C., AND MOHR, R. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 5 (1997), 530–534.

[9] SEBE, N., AND LEW, M. S. Comparing salient point detectors. *Pattern Recognition Letters 24*, 1-3 (2003), 89–96.

[10] VIOLA, P., AND JONES, M. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition* (2001), pp. 511–518.

[11] YUAN, C. Markerless pose tracking for augmented reality. In *Proceedings of the Second international conference on Advances in Visual Computing* (2006), pp. 721–730.