2D keypoint detection and description

Willem Dijkstra and Tonnie Boersma

Abstract— The image processing industry requires a way to detect and compare keypoints in images. SIFT is over a decade old and still one of the most precise and robust options. SURF is SIFT his potential successor and claims to have similar results with less computational effort. In recent years other 2D keypoint detection and description algorithms like BRISK, ORB and KAZE appeared which all claimed to be better then SIFT or SURF in some aspect. This paper compares SIFT, SURF, BRISK, ORB and KAZE, based on their respective algorithm and an experiment which compares the repeatability of the corresponding algorithms. This paper discusses common goals and hurdles each method has or faces and aims to provide an unbiased advice of when to use which method.

Index Terms—2D keypoint detection, 2D keypoint description, SIFT, SURF, ORB, BRISK, KAZE

1 Introduction

Many computer vision applications use 2D keypoint detection and description algorithms as initial step. These applications can be used for object recognition or motion tracking. 2D keypoint detection can be described as locating points of interest in images, while description refers to the representation of each keypoint.

In this paper we discuss and compare the following 2D keypoint detection and description algorithms: SIFT, SURF, ORB, KAZE and BRISK. Each of these algorithms varies widely in the types of features that are detected (e.g. edges, corners or blobs), the computation time required for detecting and describing the keypoints and the repeatability of the keypoints. Repeatability in this case refers to detecting similar keypoints in similar scenes regardless of distortion. Each comparison is based on the original papers with respect to the algorithms in combination with our own experiments. The focus of these experiments is the repeatability of the keypoints and the robustness of the detection, without taking into account the design goals of the algorithm itself. For example the repeatability of algorithms designed for repeatability is compared to an algorithm designed for low computational cost.

The sRD-SIFT [3] dataset is used for the experiments, this dataset contains two sub sets: planar scenes and object scenes. Each scene consists of multiple images grouped by radial distortion, this combination is ideal for our experiments.

In this paper we start with a background section containing the general idea of computer vision, the aims of feature detecting and description and a general introduction to the five 2D keypoint detection and description algorithms. In section 3 we discuss the sRD-SIFT dataset in combination with the used methods. Followed by section 4 in which we discuss the results of the experiments. In section 5 the five algorithms are compared using their respective papers in combination with our experiments, followed by the final section 6 containing the conclusion and suggestions for future research.

2 Background

This section consists of a short introduction to computer vision, followed by different aims of feature detection and description algorithms and a general explanation of each algorithm.

2.1 Computer vision

The simplistic idea of computer vision is the process of acquiring, processing and analyzing digital images. Based upon the resulting analyses, data can be extracted which can be used to, for example, make decisions.

Regardless of the goal of a computer vision application it requires a specific initial step. This step dictates how to locate objects of interest in images in order to analyze them. In general for this step one of the following three methods can be selected: 1) Writing a dedicated algorithm. 2) Using a feature detector in combination with a feature descriptor, or 3) Deep learning.

Writing a dedicated algorithm is the quickest solution in terms of computational time. If you want to locate strawberries in a field, you can use predefined information of the strawberries, whereas strawberries have a red color. By using this information an algorithm can detect objects relatively fast, but is in practice often limited by constraints, because strawberries are initially green.

Using a feature detector and descriptor is more robust but requires in general more computational time. This method detects similarities between the reference images (e.g. strawberries) and the provided image (e.g. field with strawberry plants). In general providing more reference images results in a more robust solution, due to more reference points for comparison. But the extra comparisons do increase the computational cost.

Finally deep learning, this method learns based on a ground-truth. Providing a large quantity of images in combination with the desired output a generic system is trained to classify new input. While in theory this method is in production the best solution in terms of both computational cost and robustness it requires a large amount of reference data and a computational intense training phase.

Dedicated algorithms are in general fast but are limited in usage, because you have to tailor the algorithm for the specific goal. Deep learning on the other hand requires a large amount of training data and a computational heavy training phase before you are able to work with it. Feature detectors and descriptors are in the middle ground. Feature detectors and descriptors are relatively fast and require no training phase, but they can still produce proper results. The next section will give more information about feature detectors and descriptors in general.

2.2 Feature detectors

Feature detectors are used to locate point of interest in images and its quality is based on the following points:

- Repeatability
  Running the feature detector multiple times on images of a same scene but from a different viewpoint should result in similar, if not identical keypoints.

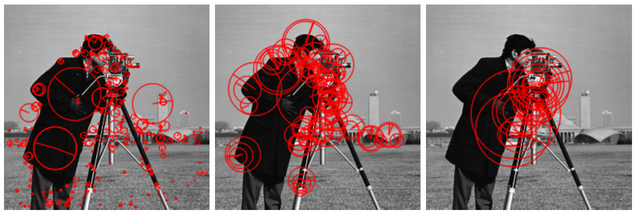
- Distinctiveness
  Each keypoint should be based on distinctive features.

• *Willem Dijkstra is a MSc. Computing Science student at the University of Groningen, E-mail: w.dijkstra.16@student.rug.nl.*
• *Tonnie Boersma is a MSc. Computing Science student at the University of Groningen, E-mail: t.boersma.3@student.rug.nl.*

- Robustness
  The influence of noise and transformation (translation, rotation, scale, etc.) on the keypoints should be minimal.

- Computational time
  The detection process should take little computational time.

In figure 1 three examples of feature detectors are shown. In the left image of figure 1 features detected by SIFT are shown. The middle image shows features detected by SURF and the right image shows features detected by ORB. This shows the different features that can be detected by different feature detectors.



(a) Features detected by SIFT (b) Features detected by SURF (c) Features detected by ORB

Fig. 1: Feature detector examples using different algorithms

2.3 Feature description

Feature descriptors encode relevant information into a feature vector and are also referred to as some sort of numerical "*fingerprint*" which can be used to distinguish one feature from another. Feature descriptors are used to compute a unique description for each keypoint. This computation is performed by using a sampling grid. Figure 2 shows the sampling grid of SIFT, figure 3 shows the sampling grid of BRISK. The quality of a feature descriptor is based on the following points:

- Consistent description
  All different keypoints should have a unique description, while similar keypoints should have a similar description.

- Robustness
  The influence of noise and transformation (translation, rotation, scale, etc.) on the descriptions should be minimal.

- Vector size
  The size of the description should be as small as possible while minimizing the information loss. This is important since the description size is related to the time it takes to compare features.

- Type
  Multiple options are available, but binary vectors and feature vectors are the most common. Feature vectors can contain more information compared to a binary vector of the same size. However, the distance between binary vectors can be computed faster in comparison with the distance between feature vectors.

- Computational time
  The description process should take little computational time.

2.4 Feature detector and descriptor

In practice optimizing all previous mentioned points of a feature detector and descriptor may be difficult. That is why there are specialized detectors and descriptors for various goals. If you want to, for example, build a stitching or a real-time object tracking application each application has its requirements.

A stitching application has to be precise and requires detailed keypoint descriptions. However, if you want to stitch lunar images, they do not have to be computed in real time which allows a more in depth comparison. Also the provided data of the lunar images has little to no translation and rotation.

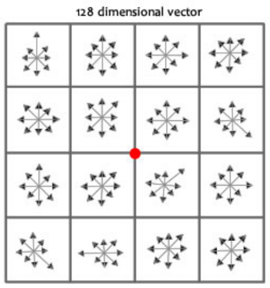


Fig. 2: Sampling grid of SIFT [2]

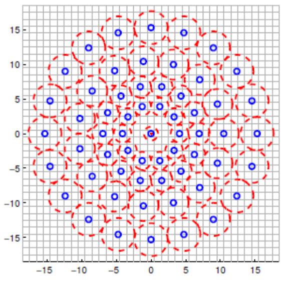


Fig. 3: Sampling grid of BRISK [7]

Real-time object tracking on the other hand requires a detector and descriptor with a small computational cost, it also has to be robust with respect to distortion and illumination. If not the object might be lost when rotated or if a different illumination is used.

It is possible to use a combination of a detector and descriptor of different methods, e.g. use the detector of SIFT and the descriptor of SURF. It is advised however to use a combination of the detector and descriptor of the same method since they complement each other. This is the reason that this paper focuses on the original detector and descriptor combinations.

In the following sections the considered feature detector and descriptor algorithms are discussed. In these sections an introduction to the considered algorithms is given. This includes for what purpose the algorithm is made and a short summary of what the algorithm does.

2.5 Distinctive Image Features from Scale-Invariant Keypoints (SIFT) [8]

The original state-of-the-art algorithm is SIFT, it is translation invariant and robust. Making it the best choice when robustness is key. The downside is its computation time, making it unfit for real-time applications. While being more than a decade old it is still today a worthy competitor.

2.5.1 Algorithm

First a scale space is created to ensure scale invariance. Then the Difference of Gaussian (*DoG*) is used to approximate Laplacian of Gaussian (*LoG*), in order to reduce the computation time. The minima and maxima in this approximations represent keypoints. Bad keypoints are filtered using an Harris Corner Detector variant making the following steps more robust. An orientation is calculated for each of the resulting keypoints. The description of the keypoints is relative to this orientation, making the description rotation invariant. The final step is generating 128 feature values representing each keypoint. These features are based on the gradient magnitudes and orientations found at the keypoint location weighted by a Gaussian function.

2.6 Speeded Up Robust Features (SURF) [5]

SURF, uses a similar approach as SIFT but uses either short-cuts or computational less expensive steps. This results in a similar results and reduced computation time, because of this it is more likely to be used in real-time applications.

2.6.1 Algorithm

Similar to SIFT the algorithm of SURF starts by creating a scale space and by approximating the Laplacian of Gaussian *LoG*. But instead of using Difference of Gaussian (*DoG*) box filters are used. These are less computational expensive and still able to give a proper approximation. The next step calculates the orientation of each keypoint using wavelet responses. These wavelet responses can be more easily calculated than gradients, because computing wavelet responses is computationally less expensive. The final step is generating 64 feature values representing each keypoint. These features are based on the horizontal and vertical wavelet response.

2.7 Oriented FAST and Rotated BRIEF (ORB) [10]

ORB, as its title suggests is a combination of the FAST keypoint detector and the BRIEF keypoint descriptor. It is designed to be used in real-time applications and in an environment like cellphones which have less processing power. At the same time ORB has a similar performance compared to SIFT.

2.7.1 Algorithm

ORB uses FAST detector in combination with scale pyramids to generate scale invariant keypoints. Since FAST does not provide an orientation option it is extended with intensity centroids resulting in Oriented FAST (OFAST) For keypoint description ORB uses BRIEF, which has a small computational cost but does not take rotation into account. BRIEF is made orientation aware using the orientation calculated with OFAST, which is called Steered BRIEF. The downside of Steered BRIEF is a significant drop in distinctiveness with respect to each description. This is negated using a greedy similarity search through a predefined training set with 300k keypoints, which results in features with the most variance. This method is called rBRIEF. The combination of OFAST and rBRIEF is called ORB, which results in a 256 binary feature vector.

2.8 KAZE Features (KAZE) [4]

The aim of KAZE is to take a step forward in performance both detection and description against previous state-of-the-art methods, while having a similar computational time. The successor of KAZE is called AKAZE which produces similar results, but with a lower computational cost.

2.8.1 Algorithm

The base of KAZE is its usage of Non-linear Scale Space. Gaussian blurring used in SIFT and SURF smooths equal for all the structures in the image, whereas in the non-linear scale space strong image edges remain unaffected. The detection itself is similar to SIFT implementation adapted to the different scale space. The orientation is calculated using the same method as SURF. For the description itself M-SURF is used which results in a 64 feature vector.

2.9 BRISK [7]

BRISK aims at reducing the computational cost while having similar results compared to SIFT. The major difference between BRISK and other mentioned algorithm is its descriptor. Instead of using a regular distributed grid (see figure 2) in combination with gradients, a circular grid(see figure 3) is used in combination with intensities. It is computational less expensive to calculate intensities in comparison with gradients, and the circular grid results in a shorter feature vector making it easier to compare. This again makes it more suitable for real-time applications.

2.9.1 Algorithm

First of all points of interest are identified across both the image and scale dimensions using a saliency criterion. The location and the scale of these keypoints are obtained in the continuous domain via quadratic function fitting. Finally the oriented BRISK sampling pattern is used to obtain pairwise brightness comparison results which are assembled into the binary BRISK descriptor.

3 Materials and methods

In this section the content of the sRD-SIFT dataset is explained, as well as the experimental methods used to compare the 2D keypoint detection and description algorithms.

3.1 Dataset

To compare the repeatability of the 2D keypoint detection and description algorithms, the sRD-SIFT [3] dataset is used. This dataset consists of two subsets, one with images of planar scenes and one with images of various objects. Each of these subsets consists of three different levels of radial distortion (10%, 25% and 45%). This radial distortion is a real lens distortion due to the fact that different types of lenses are used to acquire the images.

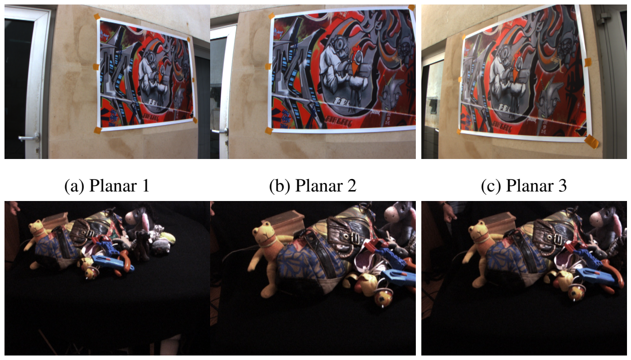
An overview of the subset with images of planar scenes is shown in table 1. This subset also includes projective transformation between planes which is also known as the homography. These are used as ground-truth during experiments since they represent the exact translation between images An overview of the subset with images of objects is shown in Table 2. These tables shows the amount of radial distortion as a percentage and the number of images taken. In figure 4 six images are shown providing a general impression of the sRD-SIFT dataset.

Planar scene	Radial distortion (%)	Number of images
1	10	13
2	25	13
3	45	13

Table 1: sRD-SIFT planar scenes subset

Object scene	Radial Distortion (%)	Number of images
1	10	7
2	25	7
3	45	7

Table 2: sRD-SIFT objects scenes subset


(a) Planar 1 (b) Planar 2 (c) Planar 3

(d) Object 1 (e) Object 2 (f) Object 3

Fig. 4: 6 images from sRD-SIFT dataset

3.2 Experiment framework

For performing the experiments a framework is developed in c++. This framework uses the OpenCV library [6] in combination with the *contrib* module. OpenCV with the *contrib* module contains the implementations for 2D keypoint detection and description algorithms such as SIFT, SURF, ORB, KAZE and BRISK, for that reason OpenCV is implemented in the framework to perform the experiments.

The plots shown in this paper are generated using Matlab with the data provided by the c++ framework.

3.3 Experiments

The first experiment is aimed at evaluating the repeatability of the considered keypoint descriptors. In this experiment we use the complete dataset of the planar scenes and the complete dataset of the object scenes. The second experiment is aimed at evaluating the robustness of the considered keypoint descriptors, and uses the complete subset containing planar images.

3.3.1 Repeatability

This experiment estimates the repeatability with respect to each algorithm and scene type.

The following process is repeated for each individual algorithm. Each image in the dataset is processed using the keypoint detector and descriptor, Followed by matching each combination using the OpenCV *Brute Force Matcher*. These matches represent the distance between keypoints. Whereas similar keypoints will have a minimal distance and uncorrelated keypoints have a maximum distance.

Correct matches are determined using the method proposed in the paper about SIFT [8]. In short the distance between the best and second best keypoint is used as reference in combination with a user defined ratio. Keypoints with a distance larger than this reference are marked as bad match and are excluded.

The repeatability itself is calculated using the correct matches in combination with the total amount of matches as described in [9]. The total amount of correct matches is divided by the minimum amount of the original matches with respect to the two images. Using these ratios the average repeatability with respect to algorithm and dataset is calculated.

3.3.2 Robustness

This experiment estimates the robustness with respect to each algorithm and scene type.

Similar as the previous experiment each image is processed using the individual algorithms. Using the resulting keypoints an homography is calculated, which contains the exact translation between two images. The planar dataset contains the original homography, comparing these the average deviation is calculated.

4 Results

This section describes the results of the performed experiments. The results of the repeatability experiments are shown in figures 5, 6 and 7. For each type of radial distortion (10%, 25% and 40%) a boxplot is generated. Each boxplot includes the planar scene and the object scene with respect to the radial distortion and algorithm.

Table **??** and table **??** show for each algorithm and each type of radial distortion the average of matching keypoints in terms of percentage.

In general it can be seen that regardless of the algorithm the planar scene has a higher score compared to the object scene. However when the radial distortion is increased this difference is still present but less prominent. In terms of general repeatability the SIFT and KAZE have the highest score while ORB is general has the lowest score. And radial distortion as expected has a negative influence on all algorithms, decreasing on average the repeatability by 9.2%

Due to time constraints the second experiment could not be completed and is excluded during the discussion. This experiment could provide valuable information, because of that it is included in the future work section.

Method	10% RD	25% RD	40% RD
SIFT	50.5	38.8	27.5
SURF	36.8	25.0	19.1
ORB	17.4	12.7	07.5
KAZE	46.2	32.0	21.7
BRISK	22.5	15.2	11.0

Table 3: Average matching keypoints in terms of percentage for the planar scenes, for each algorithm and each type of radial distortion (RD)

Method	10% RD	25% RD	40% RD
SIFT	26.1	31.9	22.2
SURF	24.7	28.1	20.1
ORB	05.3	11.6	08.5
KAZE	24.0	31.1	24.8
BRISK	08.1	15.6	07.7

Table 4: Average matching keypoints in terms of percentage for the object scenes, for each algorithm and each type of radial distortion (RD)

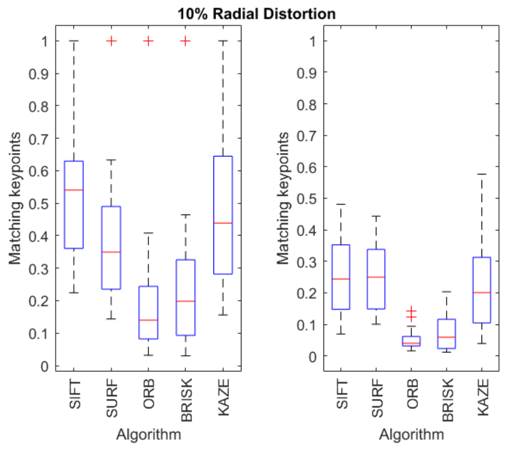


Fig. 5: Compare matching keypoints between 2D keypoint with 10% radial distortion. Left figure with planar scenes. Right figure with object scenes

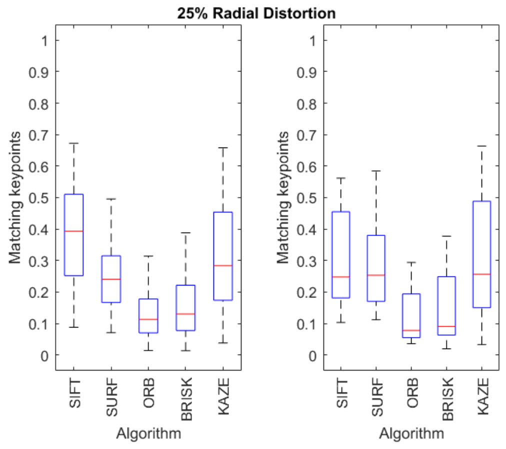


Fig. 6: Compare matching keypoints between 2D keypoint with 25% radial distortion. Left figure with planar scenes. Right figure with object scenes

5 Discussion

There are no experiments performed with regard to the computational cost of each algorithm. The comparison of the computational cost is therefore solely based upon the papers of each considered algorithm.

ORB is designed to minimize the computational cost and in general it has. Using a simple rotation invariant algorithm and making it rotation aware is the technique used in ORB. Due to its simplicity and optimization it is by far the quickest solution with regard to the considered algorithms.

The design goal of BRISK is similar to ORB. BRISK claims to provide similar results compared to SIFT and SURF, but BRISK comes with less computational cost. In BRISK the largest gain factor is the

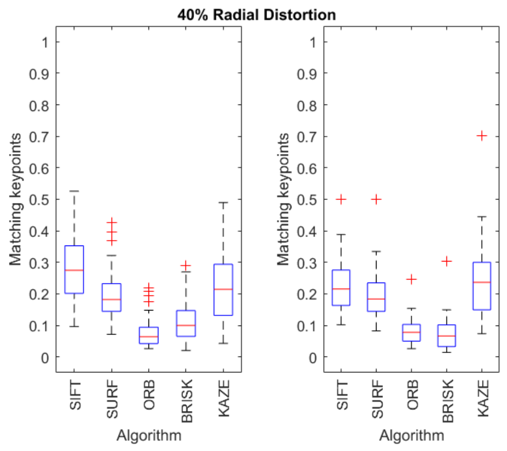


Fig. 7: Compare matching keypoints between 2D keypoint with 40% radial distortion. Left figure with planar scenes. Right figure with object scenes

binary descriptor which can be calculated and compared very efficiently resulting in the second spot.

SURF, as proven before, has a reduced computational cost compared to SIFT. Unfortunately SURF does not outperform ORB and KAZE.

KAZE and SIFT are more or less the same in terms of computational cost. Both algorithms are not designed for a low computational cost which is definitely noticeable when comparing these two algorithms with SURF, ORB and BRISK.

In terms of repeatability with little distortion the order solely based on the papers can be roughly estimated. The experiment done in this paper is to reinforce this estimation.

SIFT is the original state-of-the-art solution and has earned that spot for a reason. While almost all alternatives focus on a lower computational time with similar results, only KAZE has the focus to be an improvement.

Comparing the techniques of these two algorithms one thing becomes clear, both are based on a similar design. The major difference is the used scale space. SIFT uses a linear scale space while KAZE uses a non-linear scale space. This allows KAZE to apply Gaussian filtering locally resulting in a the removal of small detail while respecting the contours of large objects, which global Gaussian filtering is not able to do. This results in more exact keypoint locations allowing KAZE to make a more precise comparison compared to SIFT.

Intuitively this leads to the conclusion that KAZE is better than SIFT. Using the results of our experiments (see figure 5 and table **??** & **??**) this is unconfirmed. Although the object scenes differ only a little the planar scenes show an improvement. Due to this, SIFT wins with regard to the repeatability of the keypoints, whereas KAZE is a close second.

For the other algorithms it is difficult to determine the repeatability based on their respective papers. Since SURF, ORB and BRISK claim to have similar results compared to SIFT in specific situations or certain scenes. Using the differences from our own experiment as base line we conclude that SURF better than BRISK while BRISK is better than ORB.

In terms of repeatability with increasing distortion (see figure 6 & 7, table **??** & **??**) the order is difficult to estimate. Since most papers discuss the quality up to a certain point, but rarely include these types of outliers and none of the algorithms are build in a way to specifically deal with this. Still SIFT and KAZE are expected to remain as a solid choice, while ORB cannot decrease much further.

Again using the results of our own experiments it can be seen that all the repeatability ratios drop noticeably. Except for ORB which stay on average (both planar and object combined) and does not change much. Because of this the percentage difference between for example SIFT and KAZE becomes negligible. Resulting in SIFT and KAZE as still the best algorithms, followed by SURF, ORB and BRISK.

ORB with respect to the repeatability drop could be the winner, due to the fact that its repeatability, even though not great, remains almost constant on average.

In general a trend can be determined, whereas algorithms with a low computational cost pay for this in terms of repeatability. Hinting that cutting to many corners in the process is not a feasible strategy in the long run. Also different approaches are required to further improve both aspects without giving up on either one of them.

It is also clear that objects which can obscure each other differently due to translation are more difficult to track. This is simply because some keypoints can not be detected from certain angles. This is of course not something that can be fixed with an algorithm, but explains the difference in repeatability based on scene type.

6 Conclusion

SIFT which is the oldest algorithm is still the better algorithm in terms of design. SIFT is designed to be precise and robust and still today it tops most other feature detectors and descriptors. KAZE is a more recent algorithm which rivals SIFT in terms of robustness. KAZE even surpasses SIFT in certain situations, making it a better choice depending on the exact application. If robustness is the key target of the computer vision algorithm using one of these 2D keypoint detection and description algorithms is advised.

However, when a small computational cost is required (e.g. for smart phones, real-time application) ORB is the best contestant. Although ORB lacks some robustness, it is still able to yield decent results.

SURF and BRISK are in the middle ground, yielding both decent results for a reasonable computational cost.

Important to note is that both SIFT and SURF are patent protected. This means that for non educational purposes both SIFT and SURF require permission to be used. While this does not influence the performance of the algorithms it should still be taken into account when deciding what algorithm to choose.

To get a better comparison of the algorithms more experiments could be performed and more algorithms could be included in the research. The next section gives a few suggestions for future research.

6.1 Future work

Future work can include the following options:

The presented research only includes a comparison of five algorithms. More algorithms like for example AKAZE or FREAK can be taken into consideration, since a wider range of algorithms can result in an improved contrast between algorithms and highlight exceptional results.

In this research experiments which evaluate the computational cost of the algorithms have been omitted. Instead in this paper computational cost is based on the original papers and not on experiments. Experiments which evaluate and compare the computational costs of each algorithm could give a better comparison.

Due to lack of time the experiment which would evaluate the robustness of the algorithms is not performed. Performing these experiments could give a valuable insight in the actual robustness of each algorithm.

Also larger and/or different datasets to generate more exact results could be used.

Acknowledgements

We would like to thank the expert reviewer Nicola Strisciuglio and the anonymous reviewers who reviewed drafts of this paper

References

[1] Key point example. https://nl.mathworks.com/help/vision/feature-detection-and-extraction.html.

[2] Sift descriptor. http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features.

[3] srd-sift data sets. http://arthronav.isr.uc.pt/ mlourenco/srdsift/dataset.html.

[4] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer, 2012.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[6] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.