UNIVERSITÉ DE GENÈVE

DIGITIAL FORENSICS
14X065

TP 5: Copy-Move Forgery Detection

Author: Denis Fedosov

E-mail: Denis.Fedosov@etu.unige.ch

November 2024

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Exercise 1. Investigate copy–move forgery detection method based on local descriptors

Copy-move forgery is a type of image tampering where a part of the image is copied and pasted into another location within the same image. This technique is commonly used to conceal or duplicate objects within an image, thereby misleading viewers. Since the copied region comes from the same image, it matches the surrounding noise, color palette, and other properties, making it difficult to detect visually. In this lab we are going to explore some techniques to detect the tampering. Each method is based on identifying key points and matching features to find duplicated regions within the image.

AKAZE (Accelerated-KAZE)

AKAZE stands for Accelerated-KAZE, a feature detection algorithm that is optimized for computational efficiency. Unlike other methods, AKAZE uses nonlinear scale spaces instead of Gaussian scale spaces. The main steps in AKAZE-based copy-move forgery detection are:

- Detect key points in the image using the AKAZE algorithm.
- Extract descriptors around the detected key points.
- Match the descriptors between regions of the image to find duplicated areas.
- Filter out matches that are likely due to background similarity rather than actual copy-move forgery.

SIFT (Scale-Invariant Feature Transform)

SIFT is particularly effective in identifying features that are invariant to scale, rotation, and some affine transformations. In copy-move forgery detection, SIFT works as follows:

- Key points are detected in the image at multiple scales and orientations.
- Descriptors are generated for each key point, encapsulating its local appearance.
- These descriptors are compared within the same image to find matching pairs of key points that may indicate duplicated regions.
- Post-processing steps are applied to remove false matches.

SURF (Speeded-Up Robust Features)

SURF is similar to SIFT but is designed to be faster by approximating certain calculations. SURF works well for images where real-time processing is necessary. The steps for detecting copy-move forgery using SURF are:

- Detect interest points in the image using a fast approximation of the Hessian matrix.
- Generate descriptors based on these interest points.
- Perform matching between these descriptors to find potential duplicated regions.
- Refine the results by removing low-confidence matches.

Comparison of Techniques

- **AKAZE** is efficient and works well for real-time applications while providing robust performance.
- **SIFT** is highly accurate but can be slower due to its complexity, especially with larger images.
- **SURF** offers a balance between speed and accuracy, making it suitable for applications that require quick processing without sacrificing too much accuracy.

Exercise 2. Repeat the forgery detection results

I run 3 algorithms on images provided in the github repo.

Overview of results

Globally both 3 techniques detected well the image forgeries in different situations like background, multiple duplications, color palette, etc. However in figure 10 and 19 AKAZE had some hallucinations where a not duplicated part was selected.

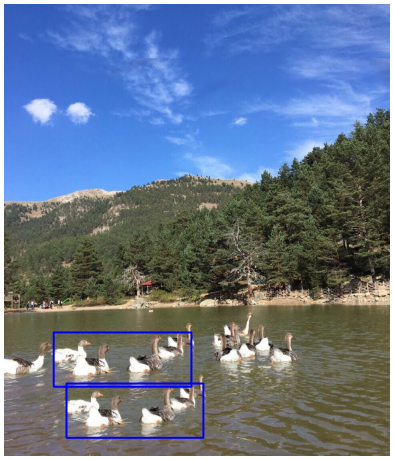

Figure 1: AKAZE

Figure 2: SIFT

Figure 3: SURF

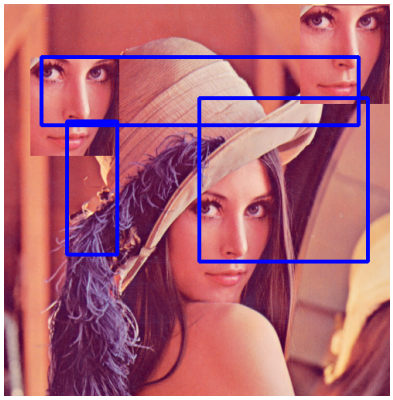

Figure 4: AKAZE

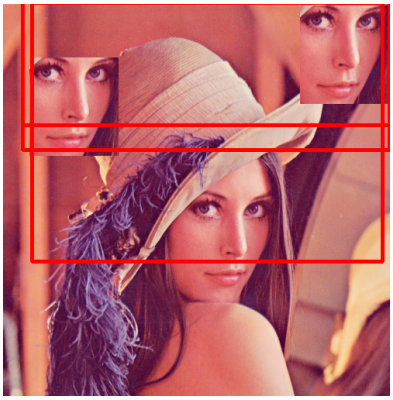

Figure 5: SIFT

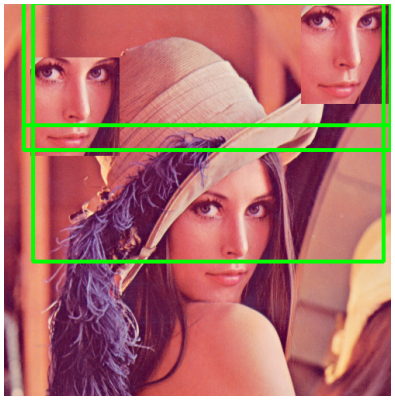

Figure 6: SURF

Figure 7: AKAZE

Figure 8: SIFT

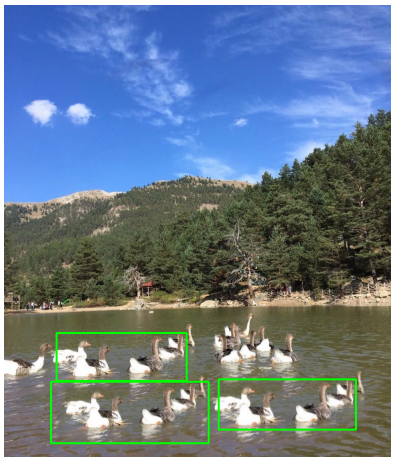

Figure 9: SURF

Figure 10: AKAZE

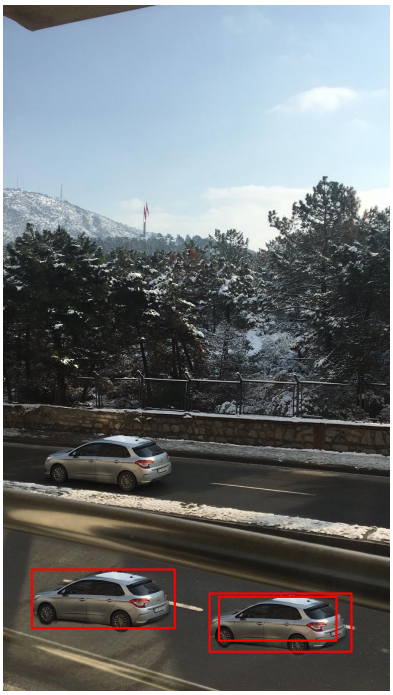

Figure 11: SIFT

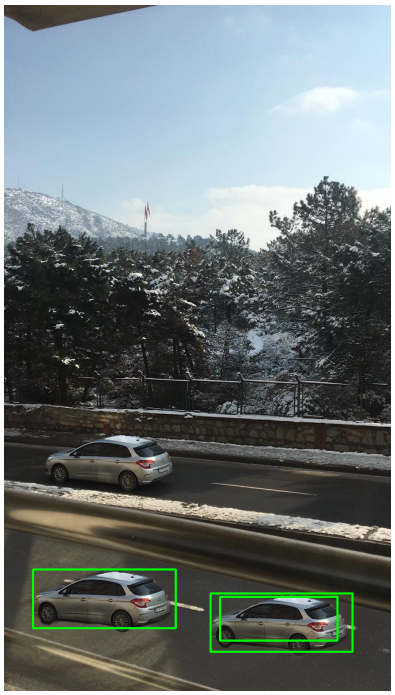

Figure 12: SURF

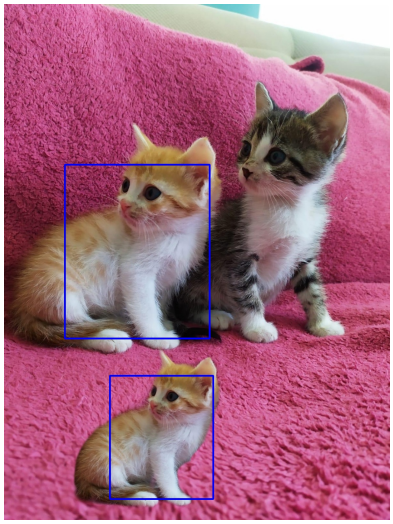

Figure 13: AKAZE

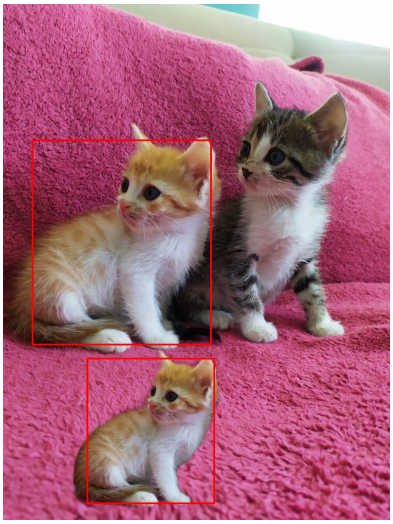

Figure 14: SIFT

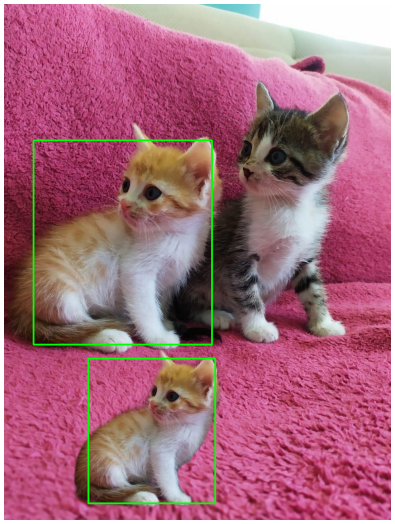

Figure 15: SURF

Figure 16: AKAZE

Figure 17: SIFT

Figure 18: SURF

Figure 19: AKAZE

Figure 20: SIFT

Figure 21: SURF

Exercise 3. Generate 9 copy-move modified images

In this part I worked with an image of a cat called Sherlock, and I applied some tampering techniques on it. Then I run the algorithms on the images to evaluate their performance.

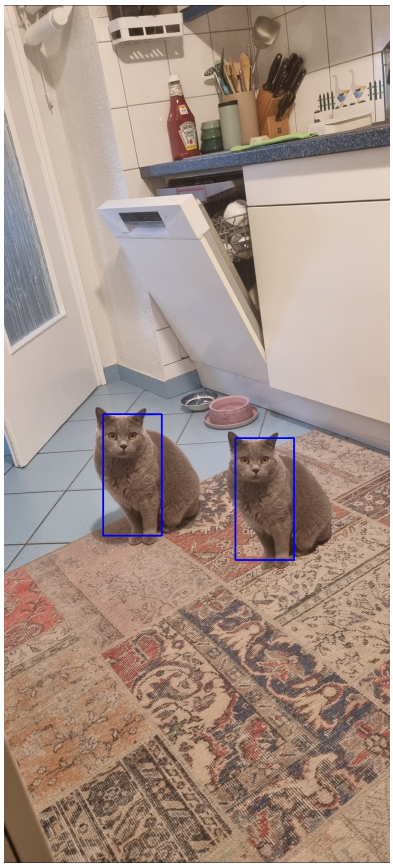

Figure 22: AKAZE

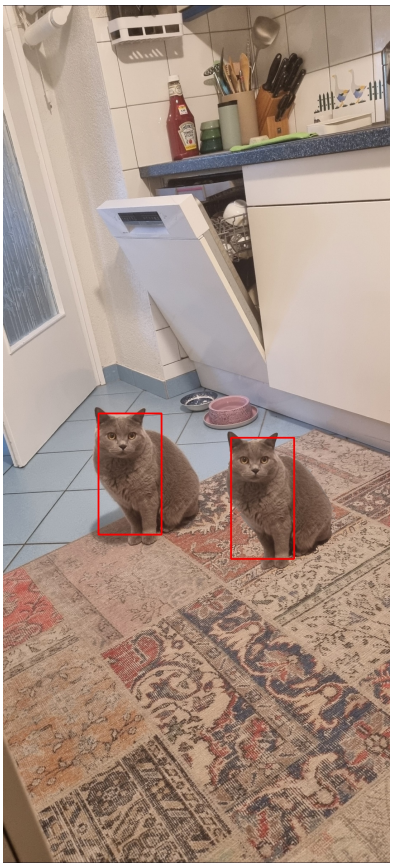

Figure 23: SIFT

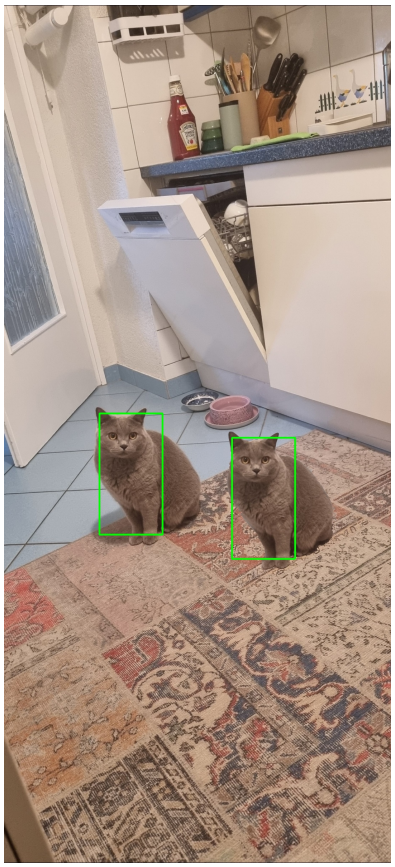

Figure 24: SURF

Moved without modifications

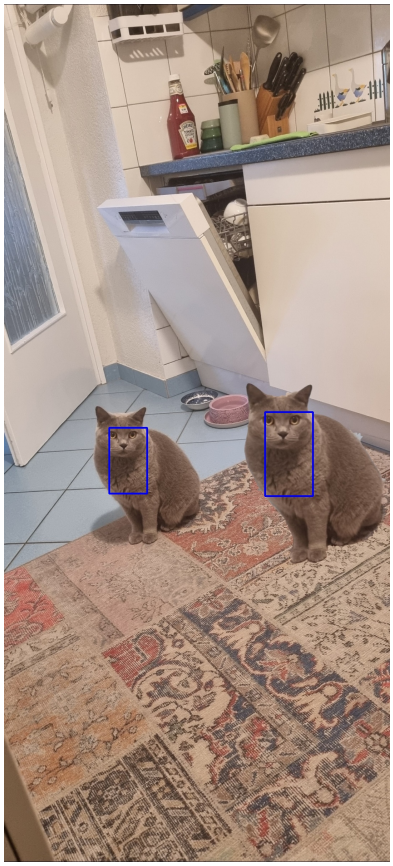

Figure 25: AKAZE

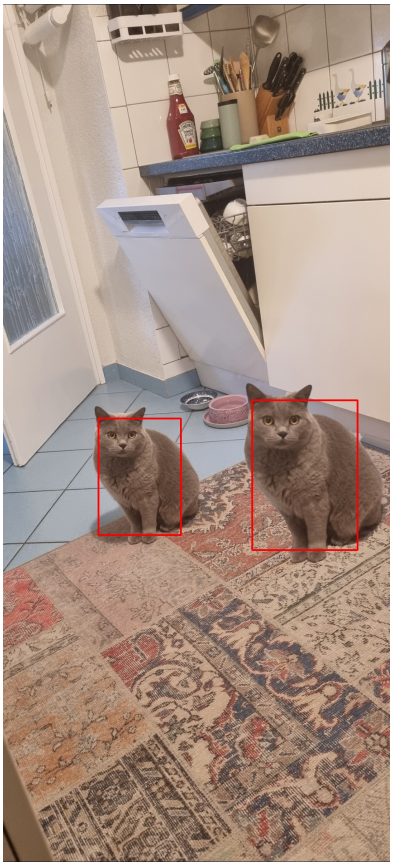

Figure 26: SIFT

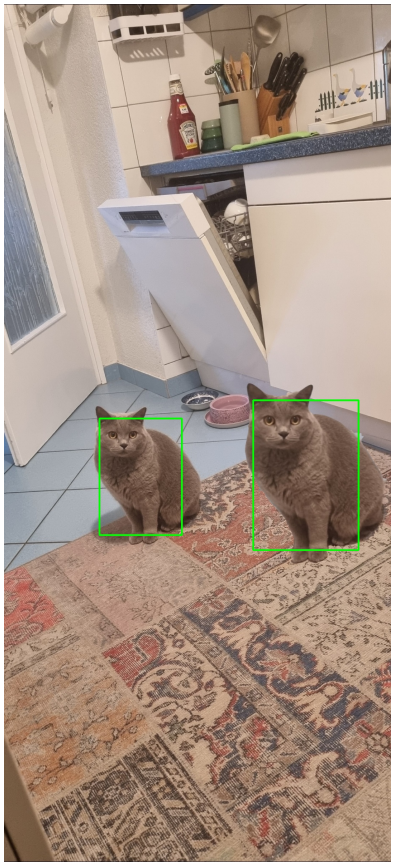

Figure 27: SURF

Moved with scaling (bigger)

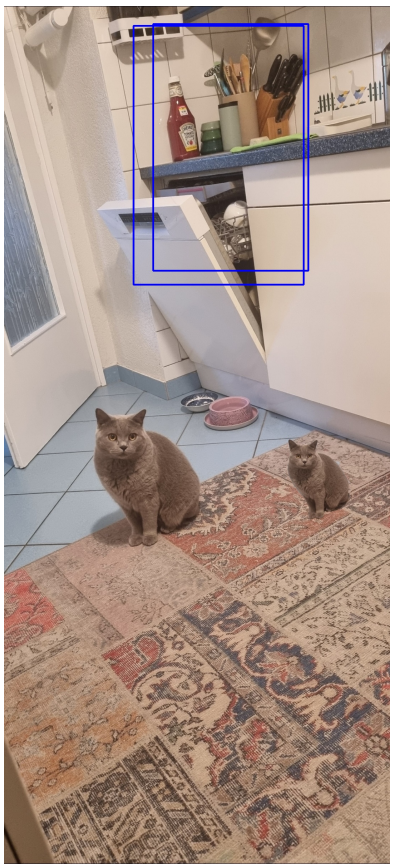

Figure 28: AKAZE

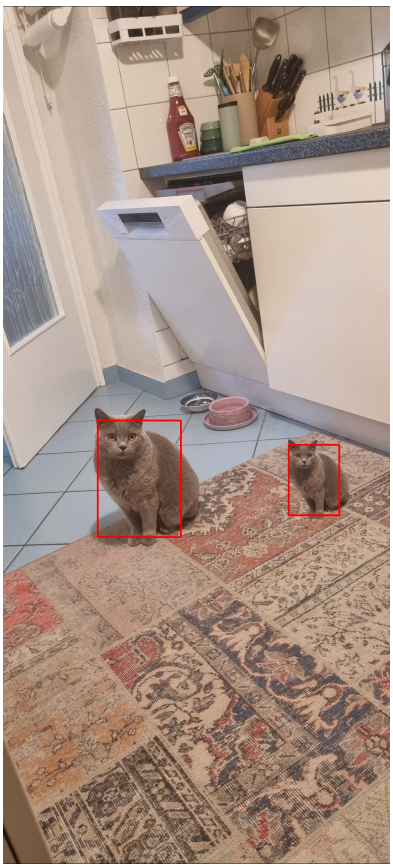

Figure 29: SIFT

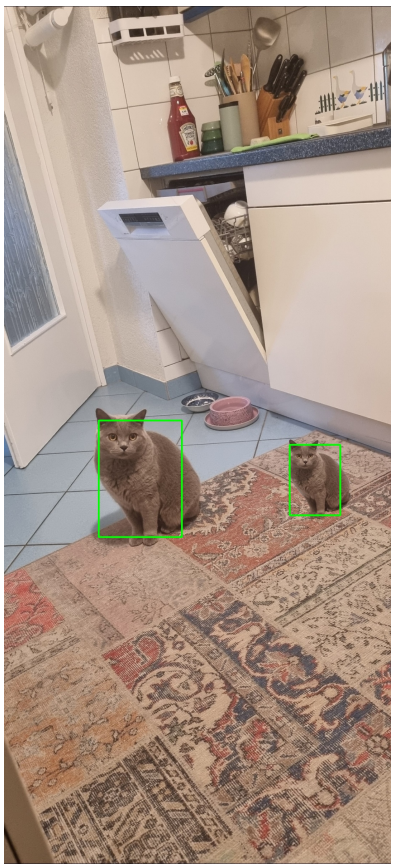

Figure 30: SURF

Moved with scaling (smaller)

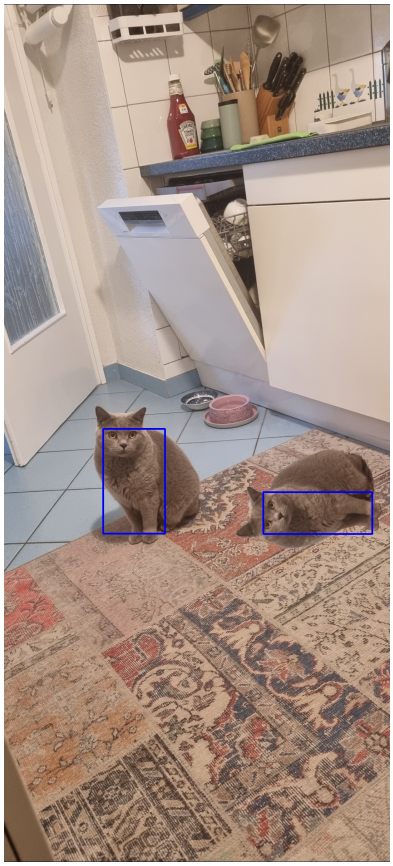

Figure 31: AKAZE

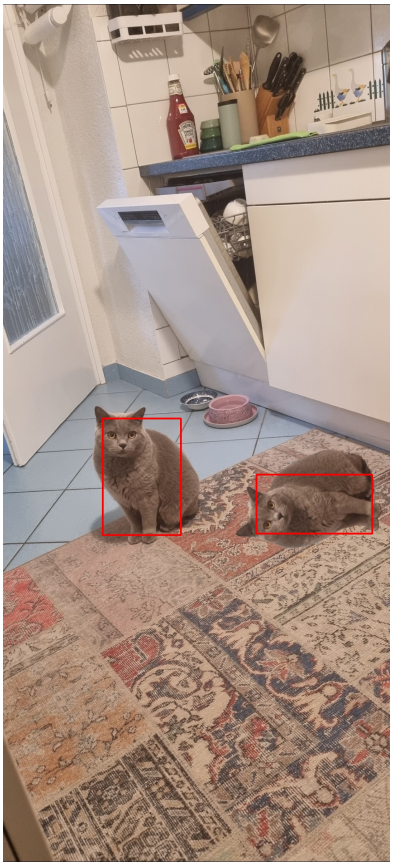

Figure 32: SIFT

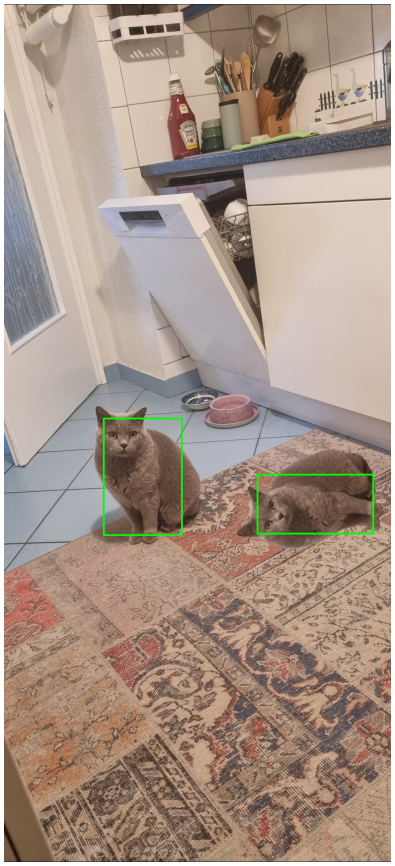

Figure 33: SURF

Moved with rotation (90 degrees)

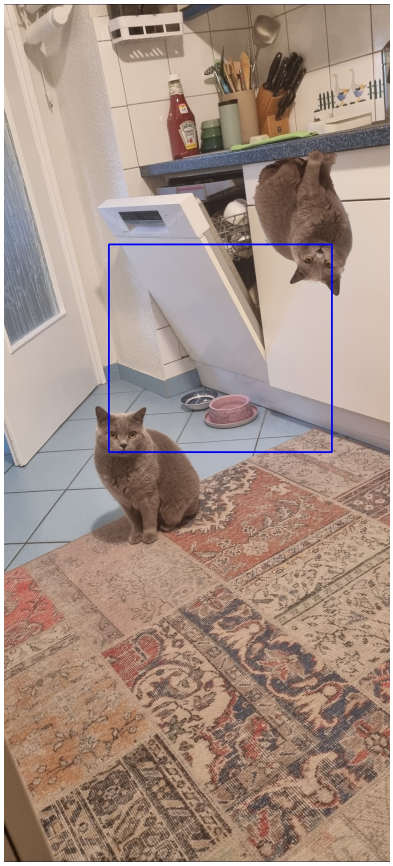

Figure 34: AKAZE

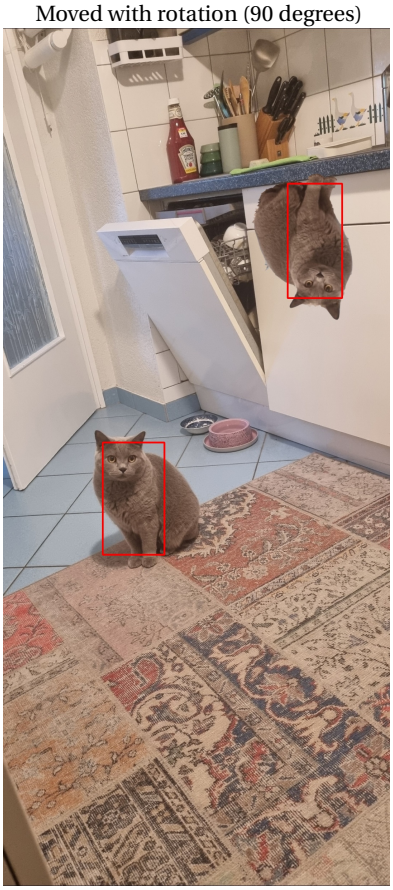

Figure 35: SIFT

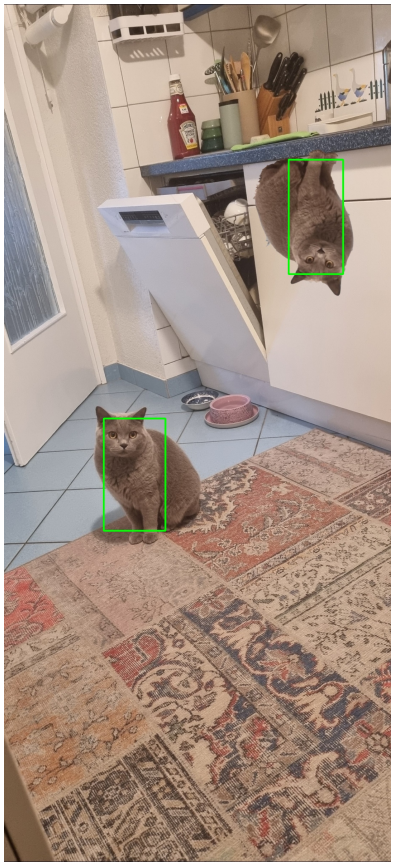

Figure 36: SURF

Moved with rotation (180 degrees)

Figure 37: AKAZE

Figure 38: SIFT

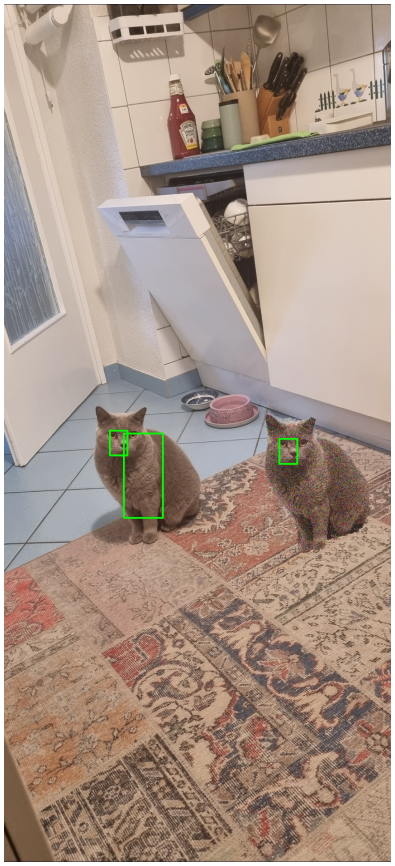

Figure 39: SURF

Moved with gaussian filter (sdt dev = 50)

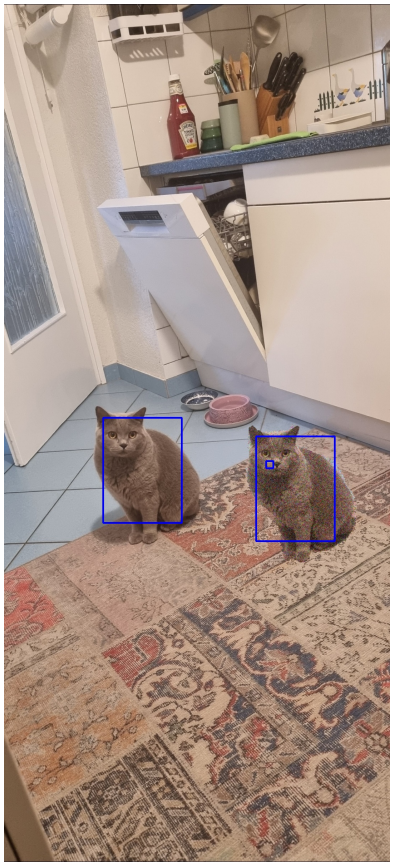

Figure 40: AKAZE

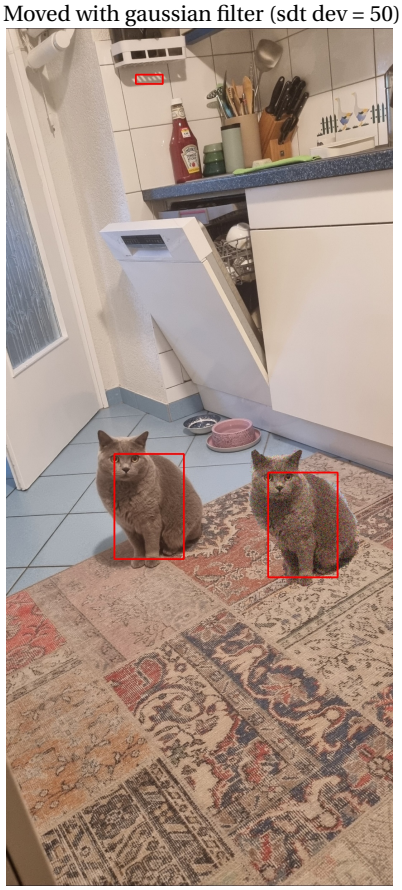

Figure 41: SIFT

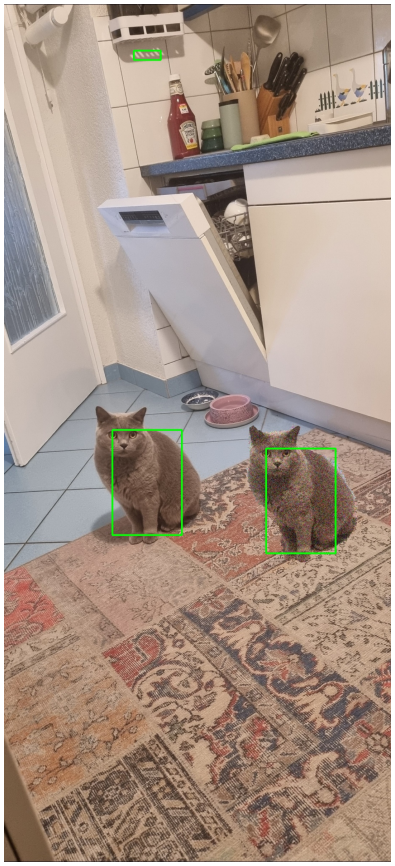

Figure 42: SURF

Moved with gaussian filter (std dev = 40)

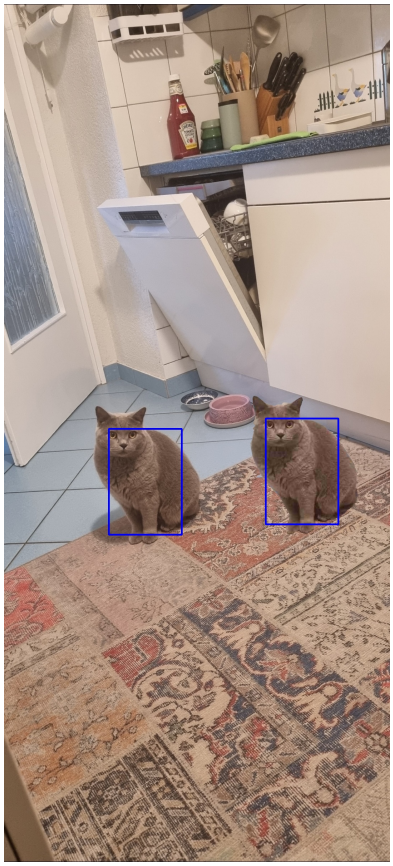

Figure 43: AKAZE

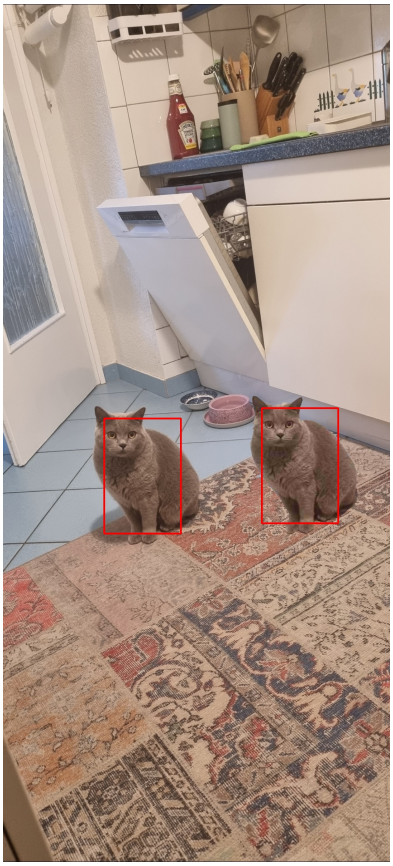

Figure 44: SIFT

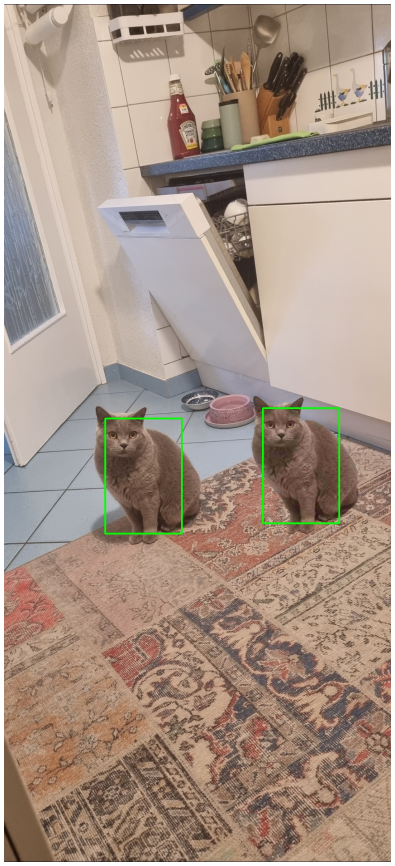

Figure 45: SURF

Moved with double jpeg compression (qf2 = 20)

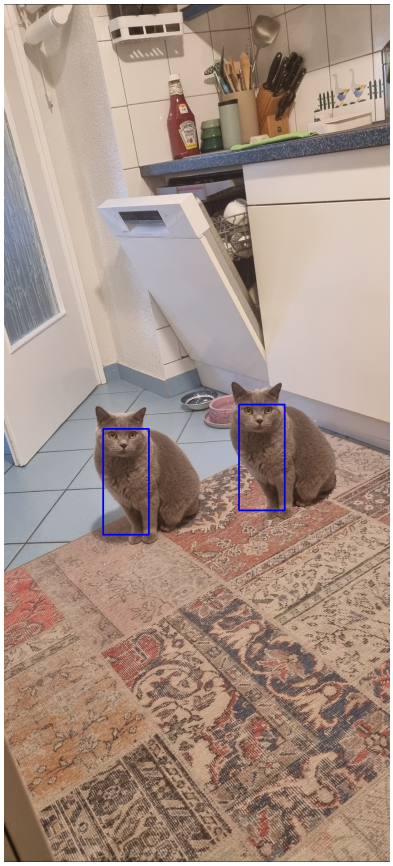

Figure 46: AKAZE

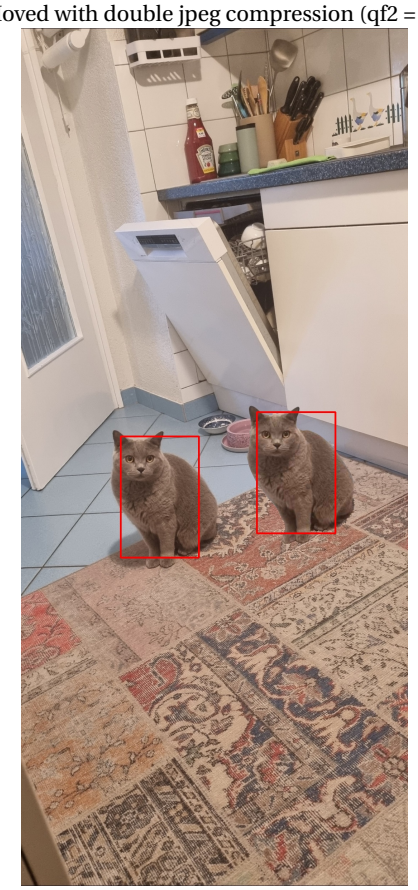

Figure 47: SIFT

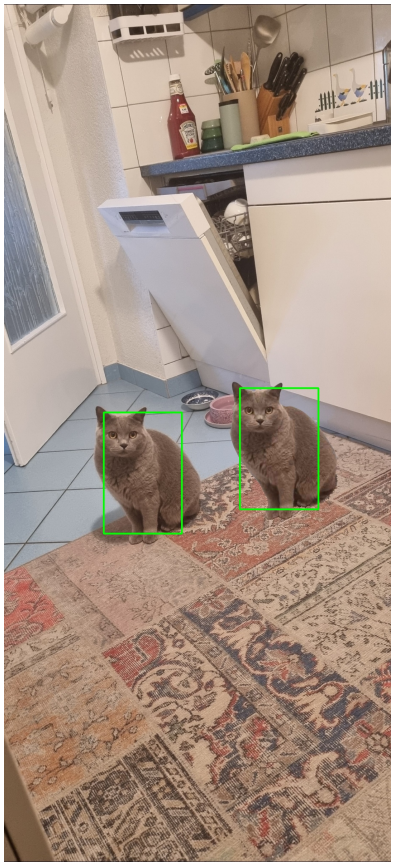

Figure 48: SURF

Moved with double jpeg compression (qf2 = 40)

Exercise 4. Investigate the robustness of copy-move forgery detection method to the distortions introduced into modified image objects

I ran the algorithms on my initial image, which was tampered with using the following modifications: copy-paste, scaling (both enlarging and reducing), rotation (90 and 180 degrees), copy-paste with Gaussian noise (levels 50 and 100), and copy-paste with double JPEG compression (QF2 values of 40 and 20).

Overall, all three algorithms successfully detected all tampering types. However, AKAZE again incorrectly flagged tampering in figure 28. Although it detected two objects, these detections were inaccurate, possibly due to graphic elements rather than algorithmic error.

Each method struggled with Gaussian noise; the detected area was smaller than usual. This could be because high levels of Gaussian noise significantly alter the image, making it difficult for the algorithms to identify duplications accurately.

Overall these algorithms are able to correctly detect manipulations even in "hard" conditions.