Image Hashing Robust Against Cropping and Rotation

Martin Steinebach[1,*], Tiberius Berwanger[2] and Huajian Liu[1]

[1]*Fraunhofer SIT/ATHENE,Germany*
[2]*TU Darmstadt/ATHENE, Germany*
E-mail: martin.steinebach@sit.fraunhofer.de
[*]*Corresponding Author*

Received 30 November 2022; Accepted 04 December 2022;
Publication 28 April 2023

Abstract

Image recognition is an important mechanism used in various scenarios. In the context of multimedia forensics, its most significant task is to automatically detect already known child and adolescent pornography in a large set of images. When fighting disinformation, it is used to identify images taken out of context or image montages. For this purpose, numerous methods based on robust hashing and feature extraction are already known, and recently also supported by machine learning. However, in general, these methods are either only partially robust to changes such as rotation and pruning, or they require a large amount of data and computation. We present a method based on a simple block hash that is efficient to compute and memory efficient. To be robust against cropping and rotation, we combine the method with image segmentation and a method to normalize the rotation of the objects. Our evaluation shows that the method produces results comparable to much more complex approaches, but requires fewer resources.

Keywords: Image identification, robust hashing, feature-based hashing, rotation, segmentation.

Journal of Cyber Security and Mobility, Vol. 12_2, 129–160.
doi: 10.13052/jcsm2245-1439.1221

1 Motivation

Image recognition plays an important role in a number of security-related applications. It refers to the recognition of an image that has already been entered into a database in advance by an assessment, and whose appearance during an analysis process should trigger a reaction. This also includes the recognition of parts of an image. What is not meant is the recognition of similar images, the biometric recognition of persons or the classification of images by machine learning.

Even before the term "fake news" became popular, there were photo manipulations that contributed to disinformation. It is known that the first fake images were created in this context as early as 1864. [19] Various image manipulations are used to make images appear different. Even a slight cropping of an image can be used to misrepresent an image. A popular form of image manipulation is the creation of image montages. This involves taking an image element, such as a person, from an existing image and inserting it into the image of another person. For example, the Malaysian politician Jeffrey Wrong Su En created an image montage to increase his standing among the population. In doing so, he cut Ross Brawn out of a photo of him being knighted by the Queen of England and replaced it with a picture of himself. This gave the impression that Jeffrey Wong Su En was knighted.

Due to the continuous development of image processing programs, it is becoming increasingly difficult to detect manipulated images with the human eye. Therefore, automatic detection systems are needed to help detect image forgeries. Image manipulations such as rotations or changes in lighting conditions, which are additionally applied to image montages, make computer-aided detection more difficult.

In forensics, the detection of child pornography during searches of electronic evidence is particularly well known. In inverse image search, corresponding techniques are used to combat disinformation through images taken out of context. Upload filters detect images to enforce copyright protection.

The detection of unmodified or only slightly modified images can be technically solved very efficiently with robust hashing methods. Copies of an image that have been scaled or lossy compressed can be detected with a memory requirement of a few bytes and a computational effort of a few milliseconds, with error rates in the low per mille range. More complex changes, on the other hand, can only be answered with equally complex procedures. Today, cropping or rotating images can only be reliably solved with feature-based methods. However, this leads to higher memory and computational requirements.

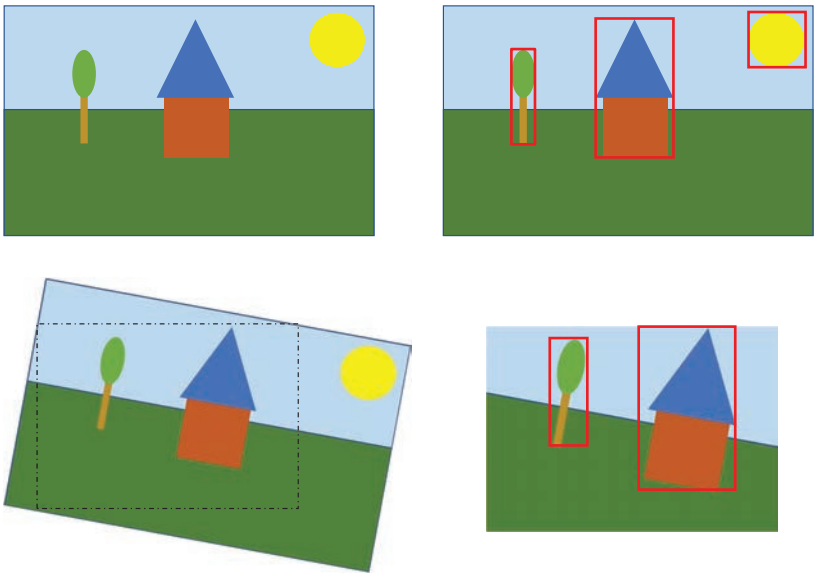

Figure 1 Cropping of an image with three objects.

It is therefore desirable to find a solution that combines the efficiency of robust hash methods with the robustness of feature methods. Initial approaches to this have already been successfully undertaken. They combine image segmentation with robust hash methods. Individual objects in an image are detected and hashed individually. Thus, as long as all objects are not removed when cropping an image, the image can be recognized again by detecting one object. Figure 1 shows this in the first line: the three objects "tree", "house" and "sun" are located and their bounding box would be hashed individually. As long as at least one of the three objects is preserved in a cropped version of the image, it can be detected.

However, this approach reaches its limits when rotation is added to the possible changes. In the second line of the Figure 1 you can see how the example is first rotated and then cropped. The cropping is done because otherwise there would be unfilled areas in the background. If we now perform object detection, the remaining objects will be detected. But since they are mapped at a different angle, the frame around them also changes. A hash that would be formed for this bounding box would not have sufficient similarity to the hash of the original object.

So a way must be found to undo the rotation or make the hash of the objects robust to rotations. Then a procedure like in Figure 2 would be

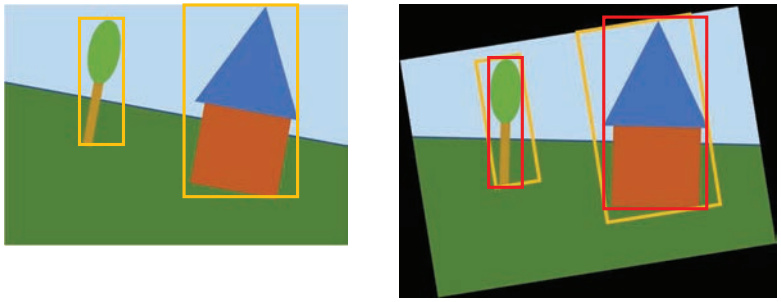

Figure 2 Object bounding box after de-rotation.

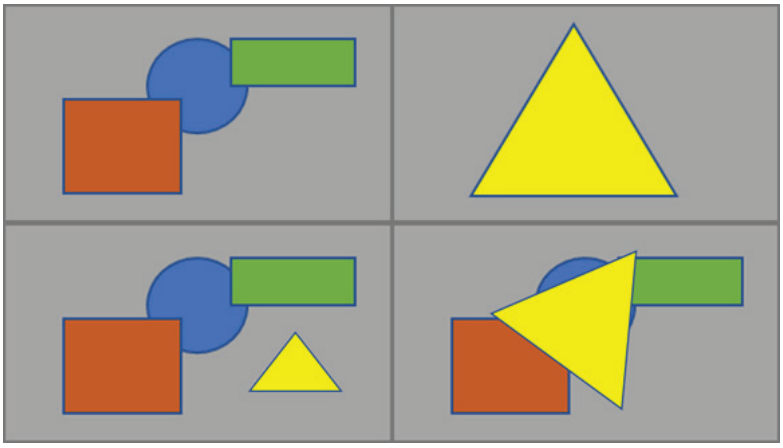

Figure 3 Abstract montage concept.

conceivable: First, the objects in a rotated image are detected and then either the entire image or each object is rotated back to its original orientation. Only now is the hash calculated, which in turn refers to the original box. The similarity would now be sufficient.

Figure 3 shows the basic idea of an image montage: the triangle in the upper right is to be inserted into the image in the upper left. The lower left figure shows a simple montage where the triangle is simply inserted next to the other image objects and scaled a bit. The bottom right shows an example with rotation, scaling, and overlapping of the objects.

One problem is that parts of the object bounding box may no longer be filled in when it is rotated to its original orientation, because when rotated they are part of the background that was previously erased by cropping. The frame around the house in the example in Figure 2 has a small area at the

top left to which this applies. In montages, this is even more urgent, since an object is usually placed in a different background. This can be worked around by not using boxes, but hashing only the detected object itself, or by blanket-filling the boxes with a background color surrounding the object. The advantage here is robustness to lost background data, the disadvantage is the smaller area that is filled with pixels of the image. This results in weaker hashes that are more prone to false positives, since fewer areas provide individual information for computing the hash.

However, the biggest challenge is determining the correct angle in relation to the rotation in the image. Forensic methods that calculate the angle based on pixel properties could be used, but are not very reliable. It is best to normalize the objects with respect to their orientation both when creating the reference data and when recognizing the image again. If there is an orientation for each object based on its properties, which can be determined by the center of gravity or other geometric properties, for example, then the object is first rotated to this normalized orientation and then the hash is computed. In this way, the hashes are always oriented in the same way, without the need to know how the angles of the two versions of the image differ during verification.

In this work, we design a system that essentially consists of three elements: an efficient, robust hash method, an image segmentation based on machine learning to find objects, and a method to normalize the orientation of an object. Both image montages and individual processed images are considered.

This system will be used to evaluate whether image recognition performance using rotations is comparable to feature-based approaches and how resource-efficient the developed approach is. Ideally, it should be possible to store a large number of images, i.e., several million, in a database and retrieve them with little effort. In this way, image retrieval for forensic investigations or disinformation detection can be done on a local device such as a laptop, which offers advantages in terms of privacy and resource utilization.

A previous work of ours addresses image montage detection based on feature recognition [39]. This achieves high detection rates and is robust to a variety of different manipulation techniques. However, the system created by [39] is limited in practical applications. Disk space and memory requirements grow with the number of feature descriptors stored and extracted from the original images.

This work is a combination and extension of the results presented in [38] for forensic image recognition and in [37] for image montage recognition.

2 Background

This section presents the methods that can be used to implement the previously described goal of efficiently achieving robustness against image cropping, rotation, and montages.

The task of image recognition can be found in several sub-disciplines in the literature. Examples include person, car, or face recognition. [23, 43, 52] The goal here is to determine if a particular object is included in the image. An example of this would be the re-identification of people, where the goal is to Recognizing persons in different images. Basically, however, the recognition disciplines that focus on specific areas such as people are grouped together as object re-identification. This means that not the whole image is recognized, but single parts (or the objects shown there) of the image.

Image re-identification is also used in image authentication, for example in content authentication. In the literature, the term "near duplicate detection" is often used for image recognition. Near duplicates are image copies of original images that have been slightly manipulated, e.g., by lighting conditions or lossy compression [47].

2.1 Image Recognition

In image recognition, a fundamental distinction must be made between feature-based and robust hash-based methods. Both try to achieve robustness against conventional image manipulation, i.e., to recognize images even if they have been modified. The basic concept of feature recognition is to find features from regions of interest. The areas are then extracted and described by a feature descriptor. This description can then be used for re-identification [20, 39].

Hash-based approaches in the image context are called robust hashes or perceptual hashes. These differ from traditional cryptographic hash algorithms such as MD5 hashes. They are designed to survive simple modifications that would already destroy a cryptographic hash. Thus, an image can be stored lossy without changing its (robust) hash.

2.1.1 Feature-based approach

The basic concept of feature detection is to find features from areas of interest. Subsequently, the areas are then extracted and described by a feature descriptor. This description can then be used for re-identification [2]. The basic method of an image re-identification is as follows: First, an image database is created that contains the original images. Then, feature descriptors are

extracted from all images and stored in a feature vector database. This is then used as a reference. If a new image is to be identified, the feature descriptors are also extracted from this image and compared with the feature vectors from the reference database. If a match is found, the requested image is marked as a near duplicate image. The match is thereby based on a reference value, or threshold, which specifies the minimum similarity of the vectors [47].

Feature-based approaches can be classified into three domains:

- **Keypoint based feature extraction**: Here, features are extracted based on keypoints. Keypoint-based feature detection includes, for example, the well-known algorithms SIFT and SURF. The two algorithms are briefly described below.

  The **SIFT (Scale Invariant Feature Transform)** algorithm was published by Lowe [29] in 1999. It is invariant to illumination changes, scaling, rotation, image noise, and to some degree robust to affine changes. [44] As a result, the keypoints are robust to geometric changes in the image. However, scalability is an issue as hundreds of keypoints need to be created by the SIFT algorithm. These are necessary to ensure recognizability [47].
  The SIFT keypoints are found by the Difference-of-Gaussian (DOG) operator. Through the DOG, local extrema are searched for over various image scalings. These are then used for the keypoint identification. Thus, even with different image scales, the same keypoints are retrieved. [29] The resulting keypoints are then described by a 128-dimensional feature vector. Due to the size of the vector, vector matching is slow and memory intensive. It also slows down the computation of the vectors [29, 44].

  SURF (Speeded Up Robust Features) was published by Bay in 2006. [3] The goal of SURF is to be a more robust and faster version than SIFT. SURF is used for classification approaches. SURF is suitable for object detection, object recognition or image recognition [47].
  Hessian matrix approximation is used to determine the keypoints. The keypoints are described by a 64-dimensional feature vector by computing the sum of the Haar wavelet response around the keypoints. [3] SURF is faster than SIFT. This is because SURF uses the Hessian matrix approximation. This also makes SURF even more robust against image noise than SIFT. SURF is invariant to illumination changes, scaling, rotation and image noise. However, its invariance to affine changes is lower than that of SIFT. [47] However, the biggest advantage over SIFT is speed. [44]

- **Pixel based feature extraction**: Here features are extracted from the pixels. For example, features can be extracted using color characteristics. [11] Another possibility is to obtain feature descriptors by edges in pixels. One such possible method is edge-SIFT. [55] Edge-SIFT is a binary descriptor based on the edges of feature areas. The method uses the SIFT feature detector in the first step. Based on the detected feature areas, the resulting feature patches are extracted and normalized. The patch extraction size is determined by this formula:

$$R_i = r \cdot scale_i$$

  The larger the value r is chosen, the larger the resulting feature patch. This means multiple edges contributing to the creation of the descriptor. However, it also increases the computation time. In the publication, the value 2.5 was used. Normalization is done by uniform scaling and orientation based on the feature information. The uniform scaling is indicated by the value D. This was set to 16. Thus, after normalization, a scaling of 16×16 pixels is achieved. After normalization, the descriptor is calculated. Here, a canny detection is performed on the resulting normalized feature patches. This results in an edge map. The pixel values have the values 1 if they are an edge and 0 if they are not. To make the descriptor even more robust, the resulting edge map is divided into four sub-edge maps. The four sub-edge maps each represent different orientation directions of [0, 45), [45, 90), [90, 135), and [135, 180). Thus, the edges are included in the respective sub-edge maps based on their orientation directions. This results in a $16 \times 16 \times 4 = 1024$ bit descriptor that can be used to determine the similarity of feature patches.
- **Area based feature extraction**: Area based feature extraction is a method in which features are computed over the entire image or in individual regions. For example, one such method would be a contrast context histogram as in the publication by Huang [21]. Here, the contrast features of local regions are used. Here, a technique was developed to compute contrast values of points in regions based on salient corners. A contrast value is calculated by the difference of the intensity values of a point and a prominent corner. A histogram based representation of the contrast values, the regions and the regions and the prominent corners is created.

  In the first step, the corners are extracted from a multi-scale Laplace pyramid by detecting the Harris corners on each plane of the pyramid.

This achieves invariance to scaling. A salient corner is selected if the minimum eigenvalue is greater than all the eigenvalues of its neighbors. In the second step, the descriptor is computed. For each salient corner p_c, a region R is spanned, p_c being the center in R. Now the contrast $C(p)$ of point p (sub-region of R) in R is computed by

$$C(p) = I(p) - I(p_c)$$

Based on $C(p)$, the descriptor of p_c is constructed.
The region is thereby mapped as a log-polar coordinate system. $I(p)$ and $I(p_c)$ are the intensity values of p and p_c, respectively.
To achieve invariance to rotation, the log-polar coordinate system is aligned with the edge orientation of p_c. A self-developed contrast histogram is constructed of each sub-region based on $C(p)$. The result gives a positive and a negative value. The descriptor is now obtained from the two contrast histogram values of each sub-region and the number of regions [21].

2.1.2 Hash-based Image Recognition

Hash-based algorithms are used in various application areas, such as image search, duplicate or near-duplicate detection, and image authentication. [14, 15, 31, 42] Robust hashes, as mentioned above, are not sensitive to slight changes such as lossy compression. Even with compression, the resulting hashes would be very similar. Therefore, when identifying images, the use of robust hashes is more appropriate than cryptographic hashes. A fundamental aspect of this is that images are recognized here via a similarity of the hash, but the hash need not be identical. Similarity is described by distances, for example by the Hamming distance. Whether an image is recognized is then controlled by a threshold value.

Several strategies for creating a robust hash can be found in the literature. Examples include: Discrete Cosine Transform (DCT) [13], Marr-Hildreth operator [4], Radial Variance [16], and Block Mean [50]. More recent approaches use deep learning and are therefore referred to as "deep hashes" [8, 54].

For this work, the robust hash rHash was used. This is a hash function based on the block mean. The original version was developed by Yang [50] and improved by us in [36]. The mean calculation was improved and a weighted distance metric was added, resulting in a more robust hash generation. In addition, robustness to image flipping has been implemented. The rHash is also robust to scaling and compression. During design, care was

taken to ensure that the computation of the robust hash must be fast in order to be used for forensic purposes. In earlier versions of the rHash, Zauner [53] already evaluated that the rHash is significantly faster than the complex methods such as DCT due to its simplicity. An extension was presented in [42]. Here, robustness to image pruning was implemented in combination with watershed segmentation. In the following, the generation and validation of a robust hash with rHash based on the improved version is described.

2.2 Image Segmentation

Segmentation of digital images or even videos is becoming more and more important in today's world. Segmentation is an essential part of computer vision and image processing. In particular, it is used in the fields of autonomous driving, medical image analysis, video surveillance and robot perception. [30] The task of segmentation is to divide a digital image or video into regions (segments) such that the pixels in each region have significant and/or similar visual characteristics. The resulting segments can then be used for analytical purposes [25].

The watershed algorithm has been an often used method to segment an image for several years. It was invented in 1979 by Beucher [34]. Over the years, the algorithm has been further developed by other scientists.

The watershed algorithm was also used in related work by us [42] to recognize images in combination with robust hashes.

For selecting the watershed algorithm, we followed the evaluation of Kornilov [25]. Here a evaluation of open-source watershed algorithms is provided. The most popular implementations were evaluated in terms of their speed and their memory consumption. The clear winner was the implementation of OpenCV, which was then also tested for this work.

The basic technique of watershed segmentation is based on representing the image to be segmented as a topographic map. Here the values of the pixels are decisive for the height. In a grayscale image where pixels have a value between 0 (black) and 255 (white), light pixels are considered high and dark pixels are considered low. From this, valleys are formed from dark pixel values (minima) and ridges (watershed lines) from light pixel values, which delimit the valleys (catch basins). Now, metaphorically, the image is flooded and the water level is raised. As this happens, the catch basins fill with water up to the watershed line that separates the individual valleys.

However, the problem arises here that the watershed transformation causes over-segmentation. This is because each minima is a catch basin. To avoid this, filters can be used.

2.3 Machine Learning Based Segmentation

Machine learning based segmentation offers several possibilities in contrast to the other methods mentioned above. Here, non-contiguous segments are formed, which are influenced by threshold values and objects lying within each other. The formed segments, which are produced by the machine learning, have a meaning. Here, individual pixels are divided into their respective previously trained classes. To achieve this, several intermediate steps must take place and a model must be trained for the respective task. In the literature, a model is usually trained as a comparison with the *Microsoft Common Objects in Context (COCO)* dataset is trained. [26] Segmentation by machine learning can be divided into four different categories. These are discussed in more detail below.

2.3.1 Object detection

Object detection makes it possible to identify different objects within an image, such as people, cars, or animals. The objects to be identified must be learned or trained beforehand. [1] A model that allows to recognize multiple objects in one image is YOLO (You Only Look Once) [32].

Once an object is detected, a bounding box is placed around it to ensure detectability. It is important to note that the objects are classified into instances, so that even if several identical objects overlap, they are detected separately. Object detection is used in Faster R-CNN to achieve a segmentation of single objects [33].

2.3.2 Semantic segmentation

Semantic segmentation is the process of assigning each pixel in the image to a specific class. [49] The difference with object detection is that the entire image is divided into classes, and the individual classes are marked individually rather than with a bounding box. This individual marking is called a mask. The disadvantage, however, is that when the same objects overlap they are not individually distinguishable from each other. They are marked with the same mask. The DeepLabV3 model allows semantic segmentation of an image [10].

2.3.3 Instance segmentation

Instance based segmentation can be considered as an evolution of semantic segmentation and object detection. Here, instance segmentation combines part of both properties. This makes it possible to form masks of objects

while still separating different but equal object types. [18] Same objects like persons that overlap are thus individually masked. However, unlike semantic segmentation, the entire image is not segmented, but only the objects found. The BlendMask model allows instance based segmentation of an image [9].

2.3.4 Panoptic segmentation

Panoptic segmentation is the combination of semantic and instance segmentation. It segments the whole image as in semantic segmentation while retaining the advantages of individual segmentation of the instance based variant. To make this possible, the two class groups *things* and *stuff* are defined. The class group *things* contains objects that should be separated from each other as in instance segmentation. These are for example cars or persons. The class group *stuff* contains similar amorphous regions. Similar amorphous regions are for example the sky or a road. [24] This is a classification like semantic segmentation, since this class group does not require individual segmentations from its properties. However, depending on the application, it is possible to define these different groups by yourself. An example classification and the corresponding data sets are provided by *Microsoft Common Objects in Context* [26].

2.4 Image Orientation Detection

To gain robustness against object rotations, in our approach image orientation detection is necessary. We describe two implementations for this task.

OpenCv

Using the *Principal Component Analysis (PCA)* class that OpenCV provides, the orientation of an object can be calculated. OpenCV provides [45] the following approach: PCA is used to extract features from an image. PCA allows finding a dominant direction where the data varies the most. In this process, eigenvectors are created that represent the *Principal Components* of the data. The eigenvalues of the eigenvectors indicate the magnitude of the eigenvector. The center of all data points is the beginning of the eigenvector. Now, when PCA is performed on an N-dimensional data set, N N-dimensional eigenvectors and an N-dimensional midpoint are created.

Scikit-Image

To calculate the orientation of individual objects, the *regionprops* function provided by Scikit-Image can be used. Orientation is detected by computing

an ellipsoid that has the same 2nd degree moment of area as the region of the given object. The angle is the value between the 0th axis and the major axis of the ellipsoid. An ellipsoid is stretched around the objects and the angle is calculated using the major axis (the longest blue line) of the ellipsoid and the 0th axis (the blue dashed line) [35].

2.5 Own Previous Work

The robust hash applied in the work is the ForBild block hash presented by us in [36]. It is the result of an evaluation of image hashing methods [53]. Based on this hash, we have added segmentation countermeasures based on face detection [43] and watershed image segmentation [42]. Beyond the recognition of images, we also addressed the possibility of combining privacy and robust hashing in [6]. As an alternative to robust hashing, we also evaluated feature-based montage detection utilizing SIFT and SURF in [39].

In this work image montage detection is done by recognizing image segments. The reasoning is that every montage is based on existing images. If these images are known and their usage is detected, one can reliably identify montages. An alternative to this is the forensic approach. Here image objects inserted into a background images are recognized by splicing detection, further discussed by us in [17]. Forensic approaches do not require image references, but their detection rates are significantly lower than image re-identification. Another alternative is the application of robust signature schemes as discussed in [40]. Here image montages could be recognized by a significant difference between reference and actual image. There are also digital watermarking concepts for detecting changes within images, especially fragile and feature-fragile algorithms. Here before image distribution a watermark is embedded as a security seal [7, 22, 27, 46].

3 Concept

The basic concept of this work is to develop a robust hash procedure that solves the problems addressed in the first section and is efficient at the same time. We consider a solution to be promising if it exceeds the recognition rates of existing approaches based on simple robust hash methods such as BlockHash after the described attacks, but is not significantly more complex. At the same time, we accept a lower detection rate than significantly more complex methods.

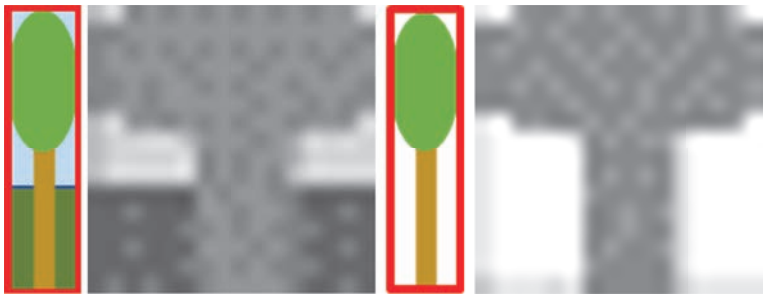

Figure 4 Removing the background image from the object leads to robust hash input data which is significantly more sparse. It is likely that the robust hash will become a silhouette of the object.

Our approach is therefore to extend known methods of robust hashing. There are approaches to make simple hashing methods robust to image pruning by image segmentation. We extend such an approach with a strategy that now also makes the segments resistant to rotation, which is not the case in the original work.

Our approach follows the following steps:

- First, an image is divided into segments or objects. To do this, we look for methods that are efficient and at the same time produce the same segmentation regardless of the changes made to the image. It is less important to match two adjacent objects cleanly than that both objects always form the same segment regardless of rotation and cropping.
- Then the orientation of the object is normalized. Again, it is only important that the object is always aligned the same way.
- From the now with respect to a rotation normalized object a robust hash is formed. Either the background can be included in the hash or only the recognized object in front of a monochrome background is hashed.

To recognize montage objects like the triangle in the lower right of Figure 3, we need to identify objects without the content of a bounding box. Figure 4 therefor shows an illustration of the two variants of the last step. Removing the background may make the object more robust against cropping of a part of the original bounding box due to rotation as shown in Figure 2. On the backside, a robust hash based on only the object without background image data may become sparse as the background is replaced by a single color. This may lead to hashes with a high false positive rate as silhouettes of objects may be very similar. We compare the performance of both variants in the evaluation section.

4 Implementation

To implement the concept of the previous section, we use the following modules:

4.1 Image Segmentation

We use the BlendMask framework to generate instance segmentations at the time of our experiments. [9] It is a *one-stage* instance segmentation method based on the object detector FCOS. BlendMask thereby combines the *top-down* and *bottom-up* methodologies of instance segmentation. [48] Top-down approaches use *high-level* features to predict whole instance. To extract *high-level* features, *bidirectional association-based pooling layers* can be used. [56] Bottom-up approaches pool local predictions and group local pixels into instances. Groupings can be done by various methods. For example, by *clustering* [5] or graph based algorithms. [9, 28] The tradeoff of the two methods is the field size. When the field is large, as in top-down approaches, the identification of instances is good, but fine details are often lost. Bottom-up approaches have the advantage of retaining high-resolution local information due to the small field size. However, it is difficult to correctly assign pixels in the grouping that then makes up the individual instances.

To combine *top-down* and *bottom-up* BlendMask uses a self-made Blender module. It combines the coarse top-level (high-level features) instance information with the fine low-level (low-level features) information. Thus, instance information and local information are used to produce a more accurate segmentation mask.

Instance segments are formed from previously learned objects. For learning, the *Microsoft COCO train2017 instance segmentation* dataset was used. It is important to segment all relevant objects from an image. The specified *confidence-threshold* plays an important role. Depending on the specified value, different numbers of objects are recognized. The default value given by AdelaiDet is 0.5 and the value in the BlendMask publication is 0.35. An example of different threshold values can be seen in Figures 5 and 6. The threshold of 0.30 was used for Figure 6 and the threshold of 0.20 was used for Figure 5. This found 34 instances in 0.28 s for the threshold of 0.20 and 10 instances in 0.20s for the threshold of 0.30. It would be possible to set the threshold even lower, but the runtime and the amount of false and double detected objects would increase considerably. For this work a value of 0.20 was chosen.

Figure 5 Illustration of BlendMask segmentation for threshold 0.20 (photo: www.cheerleader-potsdam.de).

Figure 6 Illustration of BlendMask segmentation for threshold 0.30 (photo: www.cheerleader-potsdam.de).

4.2 Rotation Normalization

All object segments are handled individually. This involves highlighting each instance segment from the image by setting the background or pixels around the object to white (255). A copy of the highlighted object segment is converted from the three-dimensional to the two-dimensional plane. This is necessary for the calculation of the orientation as the applied image orientation algorithm cannot handle three-dimensional images.

The resulting two-dimensional image is analyzed by the ScikitImage regionprops function. The object is discarded if it has a resolution smaller than 50×50, because it is assumed that such a small object has little

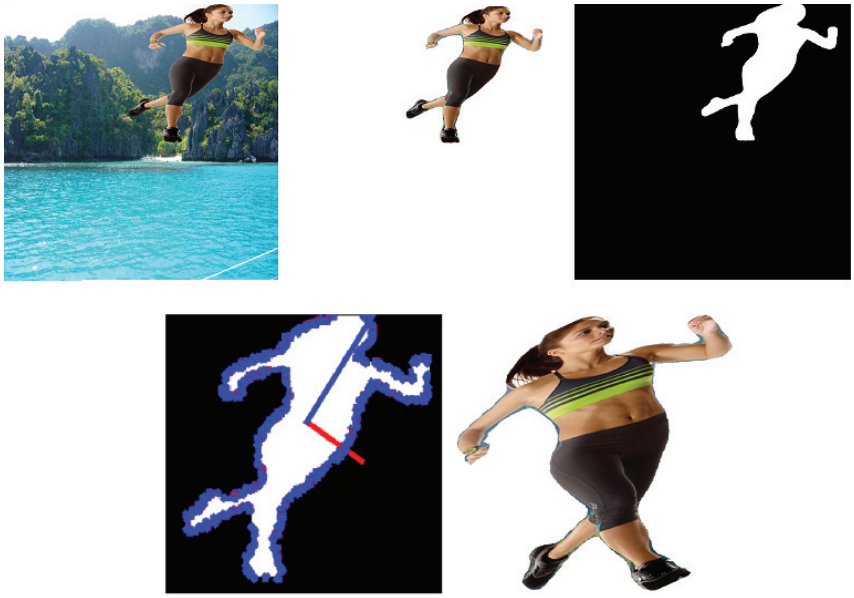

Figure 7 Process of object normalization.

relevance for a montage and rather an error of the object detection is present. If an object is large enough, the orientation is extracted and the image is rotated by the negative value of the orientation. Thus, the orientation of the object is normalized on the y-axis. The three-dimensional image is now used as the object to be rotated around which a bounding box is placed. For the rotation, it is important that the image is still displayed completely after the rotation, and is not cropped by the previous resolution ratios. The process is shown visually in Figure 7.

4.3 Robust Hash Generation

Robust hash generation is provided by a block hash algorithm [51] based on down-scaling an arbitrary image size to a 16 $\times$ 16 pixel gray scale image and calculating the hash bit values by comparison of individual pixels to the median of all pixels. Various additional steps are taken to improve robustness [41].

A first step ist the introduction of automatic mirroring. An image is divided into four equally sized square subsections and always mirrored so that its brightest subsection is at the top left. This makes the process robust against image mirroring.

The second optimization increases the selectivity. The original method tends to false detections when the structure of images is very similar. By splitting the original 16×16 pixel field into four 8×8 fields and computing the bit values based on the medians of these smaller fields, the resulting robust hash becomes less dependent on the basic structure of the image.

The third optimization introduces an alternative way of looking at the hash values. In the original method, a comparison of two binary hash values is performed over the Hamming distance. Decision. Whether two images are identical is accordingly made based on the differing bits. In practice, however, it turns out that not every bit value can be trusted equally. Bit values based on a value far from median can be trusted more than a value close to median. The latter can quickly find itself on the other side of the zero axis due to a slight change, thus producing a bit that no longer matches the original. Accordingly, a weighted distance is also computed in parallel with the Hamming distance, taking into account the distance from the median of the bits that do not match the hash value being compared. Specifically, the second moment of the distances of different bits is divided by that of identical bits and multiplied by the Hamming distance and the factor 1000. The latter serves only to obtain more manageable values.

This weighted distance now allows a two-dimensional decision whether two images are identical. We first compute the Hamming distance of all robust hashes of known images in a database to the image of interest. The hash value in the database with the lowest Hamming distance is now examined as a candidate match if the distance is at most 32. Now the weighted distance between the hash of the image and the hash in the database is calculated. The image is either considered known if the hamming distance is at most 8 or the weighted distance is at most 16.

4.4 Matching and Verification

Once all robust hashes have been computed, matching of the hashes is performed against the specified reference database. The Hamming distance, weighted Hamming distance, and similarity score are used for matching. The similarity score is the perceptual matching of the hash bits and can be derived from the Hamming distance. Each object segment is checked in turn. A hash table is created and the image to be checked is stored as a key value. The source references found from the matching check are now assigned to the key value. If there are two or more different source references in the key value, this is stored in a list that represents the found images. This list is output after all object segments have been checked.

5 Evaluation

We only present a small selection of our extensive test results here to prove the focus on the resistance to rotation. In addition to rotation by 10, −10, 20 and −20 degrees, we also examined the behavior after noise, changing the brightness and flipping the image.

For an exhaustive automated test, the required image sets were created using a montage creation script. For the construction of image modifications, 1000 images are used. Another 1000 images are for the false positive evaluation. This results in a large test image set due to the different manipulation parameters and their different resolutions that need to be tested. The image set used was the Cityscape dataset [12] and the Microsoft Common Objects in Context (COCO) dataset.

The following components were used for the evaluations:

- Operating system: Ubuntu 20.04.2 LTS
- Graphics card: NVIDIA GeForce RTX 2080 Ti
- Processor: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
- Memory: 32 GB, DDR4
- Hard disk storage: HDD, SATA 6Gb/s

5.1 Object Detection

Table 1 shows the results for rotation for the removed and kept background approach. In each case, the images were scaled down from their initial size, say 2040 × 1016, to a maximum edge length of 1000 pixels before further attacks or modifications. Thus, the results for rotation should be viewed as a combination of scaling and subsequent rotation. For rotation of 10 and −10 degrees, the approach without background behaves better, and for 20 degrees, the approach with preserved background is much more reliable. It is clear that

Table 1 Background removal shows slightly better results at 10 degree rotation, but significantly worse results at 20 degree rotation

	Background	
Rotation	Removed	Kept
10	821	810
20	768	939
−10	836	833
−20	791	950

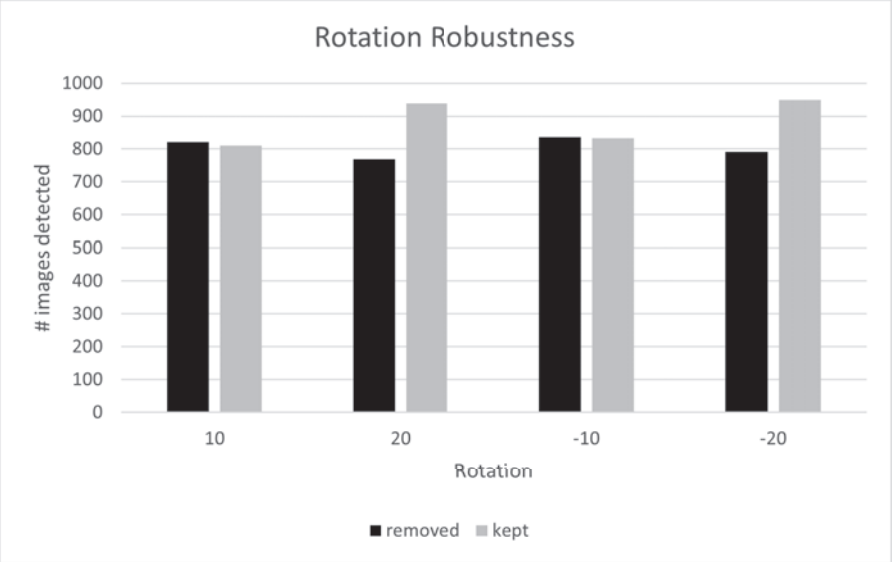

Figure 8 Illustration of rotation robustness with removed and kept background.

Table 2 Results for various attacks using the background removal strategy, pre-scaling to 1000 pixel

Attack	TPR	FPR	Precision
Image scaling	93.3%	0.1%	99%
Image rotation	>76%	19.6%	80%
Image noise	>90%	0.1%	99%
Image brightness	>89%	0.1%	99%
Image mirroring	>85%	0.1%	99%

well over 90 percent of the images were recognized here despite rotation. Figure 8 visualizes the results of Table 1.

Table 2 shows that the robustness against rotation comes with a cost. Attacks like scaling, addition of noise, change of brightness or mirroring do have a much greater impact on recognition rate of our implementation than on the original block hash. The sequence of object detection, rotation normalization and object hashing is more fragile than hashing the whole image. This is only of limited importance as the standard hash can be used together with the rotation-robust variant. More important is the increased false-positive rate (FPR). Image rotation causes a significantly higher FPR than other attacks potentially due to similar objects rotated in a normalized

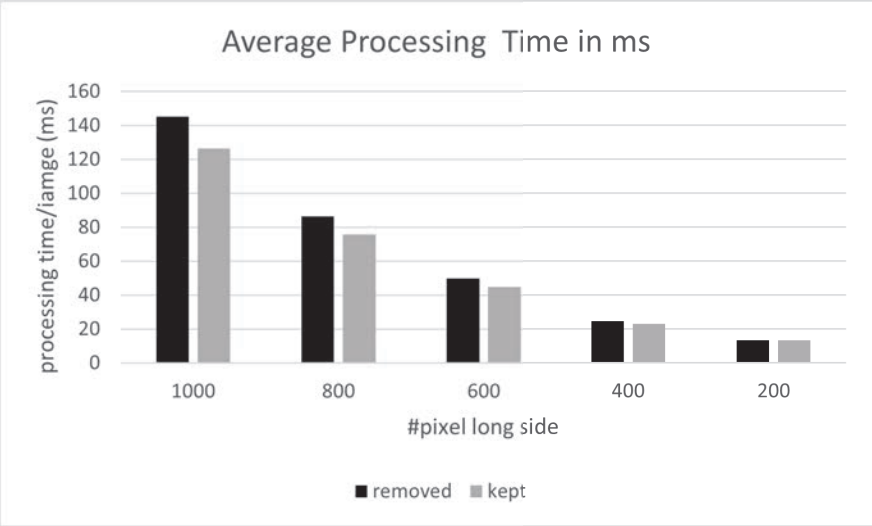

Figure 9 Processing speed comparison. Not removing the background reduces processing time.

perspective. This can be seen as a significant disadvantage of background removal as the FPR drops to 0.1% without background removal.

Figure 10 provides a more detailed look at scaling without rotation. As to be expected, detection decreases with the size of the resulting image. Especially at very small images the individual objects seem either not to be detectable anymore or the resulting hashes are too different from the original ones.

In Figure 11 we show a histogram of detected segments for scaling to 1000 pixel. One can see that in most cases, a small number of segments where detected per image. The range between 1 and 6 segments is dominant. Only in a few cases, more than 10 segments were found. For each image, up to 37 segments were stored in the test case.

Figure 9 compares the average processing time for detection after scaling. One can see that keeping the background is faster than removing it due to the additional steps necessary for cleaning the object box. For size 1000 pixel the difference is 13%, for size 600 10% and with 200 pixel only 0.66%. The average procedure of segmentation, de-rotation as well as hashing and searching for the individual objects in the database takes 145 ms with and 126 without background removal. This is significantly slower than the basic robust hashing algorithm but sufficiently fast for real-world applications.

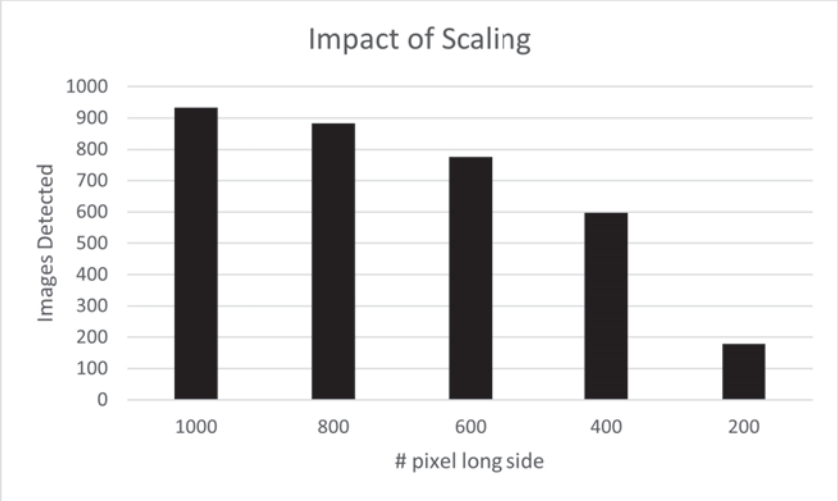

Figure 10 Scaling robustness. There is no comparison of removed or kept background as both variants provided identical detection results.

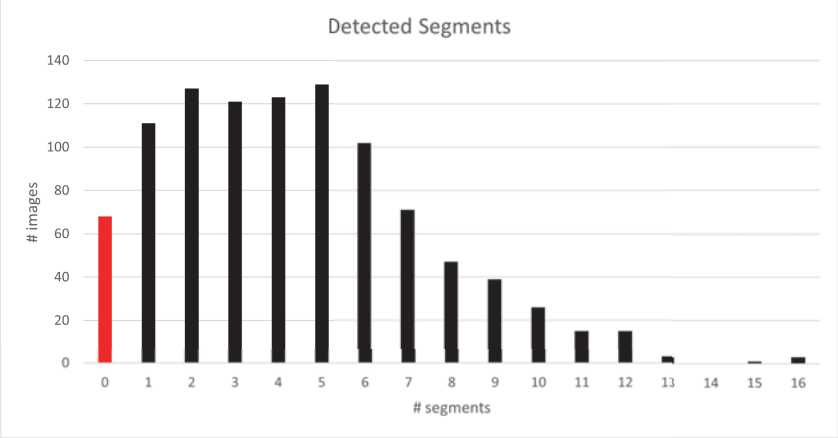

Figure 11 Segment detection histogram.

Memory usage depends on the number of objects identified within an image. For the 1,000 test images, a data base of 600 kB was sufficient. For each image, up to 37 individual segment hashes were stored. Scaled up to 1,000,000 images this would be a memory consumption of 580 MB. A SIFT-based approach would require 250 GB, SURF would still require 129 GB.

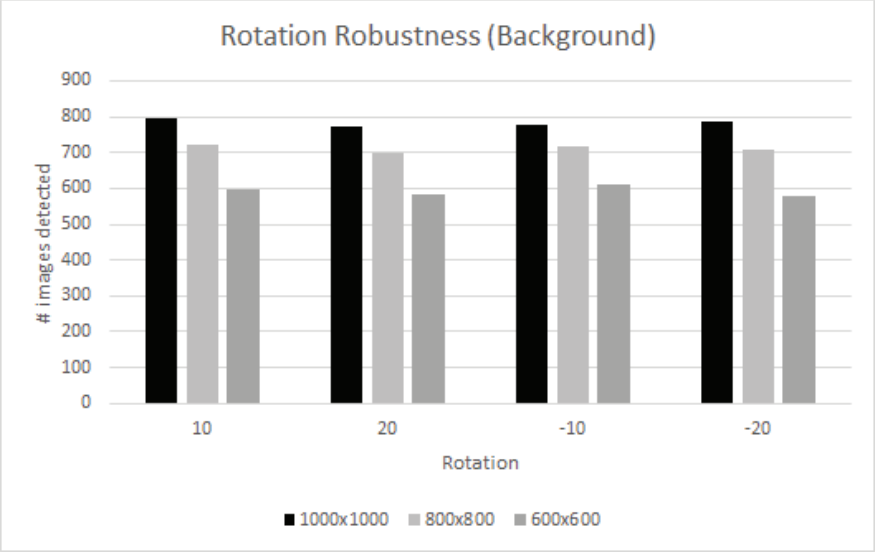

Figure 12 Object rotation results for background detection

.

5.2 Montage Background Detection

A second evaluations was performed for montage recognition as here both objects and background are detectable. Using the example from Figure 3, background recognition would find the upper left image as the background used in the lower montage examples. Object recognition would find the triangle from the upper right image. Thus, there are potentially different results for the chosen metrics as far as background recognition and object recognition are concerned. For example, a montage may not be detected because the object is not detected but the background is detected correctly.

The test refers to the background detection of the same images as used in the object rotation test above. The results are constant for all rotations and differ by a maximum of 2.1% at a resolution of 1000 $\times$ 1000 pixels. Thus, a TPR of over 77.4% is achieved. Based on the fact that the background has a resolution of 1000 $\times$ 1000 and a contained object of the object size of 50% of the resolution, and a recognition of 79.6% on average is achieved, this results in a recognition after the rotation modification of $\approx$98.4%. Thus, the scikit modification for orientation recognition biases the recognition by $\approx$1.6%. The FPR remains at 19.6 and the precision rounded down at 79%.

6 Summary and Conclusion

In the present work, we show that the combination of image segmentation, normalization of the orientation of the detected objects, and their final robust hashing can be used to develop a system that produces results comparable to feature-based methods as presented in [39]. The major advantage is the significantly lower memory requirement. Only one hash of 32 bytes is required per segment, and even with numerous segments per image (in our test set, there were up to 37 segments), the resulting databases remain compact. Thus, even large image collections of a million images can be kept in main memory without any problems, which is not possible with feature-based approaches. This is a significant advantage when used in mobile devices.

Recognition rates are significantly influenced by object detection and segmentation. Nevertheless, the recognition system delivers good recognition results as long as the images have an image scaling of 1000×1000 pixels. For much more scaled images, the approach reaches its limits. To what extent images with an edge size of 200 pixels are relevant for application scenarios must be decided individually. In most cases, the limit should rather be 600 pixels. The run-time averages 140 ms per image with an edge length of 1000 pixels, which should be fast enough for processing even large amounts of data. It seems advisable not to discard the background of objects. While potentially the detection rate, the false positive rate is reduced to 0.1%, which is a big advantage in many applications.

To increase the recognition rate of the image recognition, a hash of the entire image could be created before the segmentation. Subsequently, the segmentation is performed and the individual segments are hashed. All hashes are now stored in the database. The hash of the whole image now serves as an additional hash for recognition.

As a framework operation, the availability of the object images must be ensured. These are not provided by the system itself and must be available through other methods such as *image crawling*. The system allows, unlike the [39] variant to easily add to the database. Thus, new object images do not require the old object images to be read in again. This makes the application more suitable for automatic testing in practice than the [39] variant, since the database does not always have to be created again. Especially in areas like Twitter where many images need to be stored, this feature is important. Another feature that [39] does not have is the detection of duplicates during database generation. Thus it is possible that duplicates are stored and two different source references exist for the same object image.

Thus, if the same object image is identified, it would be falsely marked as a montage. The system presented in the thesis filters out duplicates when generating the database, so that no different source references for duplicates can arise. Requirement for the system is to provide high robustness to the image manipulations. Accuracy should be high, even if a falsely classified montage is quickly detected during manual verification. This is because the system does not simply mark an image as a montage, but also outputs the object images it consists of. Also a high correctness should be given, so that mounts can be detected reliably. For the creation of montages, practice-relevant properties are assumed. These correlate strongly with the image manipulations. A montage must have a certain added value for the montage creator. For example, a montage with a resolution of 100×100 pixels is not useful because it is much too small. Inserting an object rotated by more than 30 degrees also looks very unnatural. The goal of a montage is to make it appear realistic so that it is not identifiable as such. The last requirement is the search runtime. This was kept as low as possible.

Acknowledgment

This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. Object detection. In *Computer Vision*, pages 1–9. Springer International Publishing, Cham, 2020.

[2] Ali Ismail Awad and Mahmoud Hassaballah, editors. *Image Feature Detectors and Descriptors: Foundations and Applications*, volume 630 of *Studies in Computational Intelligence*. Springer International Publishing, Cham and s.l., 1st ed. 2016 edition, 2016.

[3] Herbert Bay, Tinne Tuytelaars, and Luc van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[4] S. Bhattacharjee and M. Kutter. Compression tolerant image authentication. In *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, pages 435–439. IEEE Comput. Soc, 1998.

[5] Bert De Brabandere, Davy Neven, and Luc van Gool. Semantic instance segmentation with a discriminative loss function.

[6] Uwe Breidenbach, Martin Steinebach, and Huajian Liu. Privacy-enhanced robust image hashing with bloom filters. In Melanie Volkamer and Christian Wressnegger, editors, *ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25–28, 2020*, pages 56:1–56:10. ACM, 2020.

[7] Hongliang Cai, Huajian Liu, Martin Steinebach, and Xiaojing Wang. A roi-based self-embedding method with high recovery capability. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1722–1726. IEEE, 2015.

[8] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[9] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask: Top-down meets bottom-up for instance segmentation.

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation.

[11] Sujan Chowdhury, Brijesh Verma, Mary Tom, and Mengjie Zhang. Pixel characteristics based feature extraction approach for roadside object detection. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[12] Cityscape. Dataset overview – cityscapes dataset, 2021-09-08.

[13] B. Coskun and B. Sankur. Video isaretlerinin algisal dayanikli kiyimi – robust videohas extraction. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, 2004*, pages 292–295. IEEE, 2004.

[14] Andrea Drmic, Marin Silic, Goran Delac, Klemo Vladimir, and Adrian S. Kurdija. Evaluating robustness of perceptual image hashing algorithms. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 995–1000. IEEE, 2017.

[15] Ling Du, Anthony T.S. Ho, and Runmin Cong. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication*, 81:115713, 2020.

[16] F. Lefèbvre, B. Macq, and J. Legat. Rash: Radon soft hash algorithm. In *2002 11th European Signal Processing Conference*, pages 1–4, 2002.

[17] Raphael Antonius Frick, Huajian Liu, and Martin Steinebach. Detecting double compression and splicing using benfords first digit law. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–9, 2020.

[18] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval*, 9(3):171–189, 2020.

[19] Hany Farid. Photo tampering throughout history.

[20] Mahmoud Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*, pages 11–45. Springer, 2016.

[21] Chun-Rong Huang, Chu-Song Chen, and Pau-Choo Chung. Contrast context histogram—an efficient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, 41(10):3071–3077, 2008.

[22] Stefan Katzenbeisser, Huajian Liu, and Martin Steinebach. Challenges and solutions in multimedia document authentication. In *Handbook of Research on Computational Forensics, Digital Crime, and Investigation: Methods and Solutions*, pages 155–175. IGI Global, 2010.

[23] Sultan Daud Khan and Habib Ullah. A survey of advances in vision-based vehicle re-identification. *Computer Vision and Image Understanding*, 182:50–63, 2019.

[24] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation.

[25] Anton Kornilov and Ilia Safonov. An overview of watershed algorithm implementations in open source libraries. *Journal of Imaging*, 4(10):123, 2018.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context.

[27] Huajian Liu and Martin Steinebach. Digital watermarking for image authentication with localization. In *2006 International Conference on Image Processing*, pages 1973–1976. IEEE, 2006.

[28] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity derivation and graph merge for instance segmentation.

[29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[30] Shervin Minaee, Yuri Y. Boykov, Fatih Porikli, Antonio J. Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[31] Dat Tien Nguyen, Firoj Alam, Ferda Ofli, and Muhammad Imran. Automatic image filtering on social networks using deep learning and perceptual hashing during crises.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks.

[34] S. Beucher. Use of watersheds in contour detection, 1979.

[35] Scikit. Regionprops funktion.

[36] Martin Steinebach. Robust hashing for efficient forensic analysis of image sets. In Pavel Gladyshev and Marcus K. Rogers, editors, *Digital Forensics and Cyber Crime*, volume 88 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 180–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[37] Martin Steinebach, Tiberius Berwanger, and Huajian Liu. Image montage detection based on image segmentation and robust hashing techniques. *Electronic Imaging*, 34:1–6, 2022.

[38] Martin Steinebach, Tiberius Berwanger, and Huajian Liu. Towards image hashing robust against cropping and rotation. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–7, 2022.

[39] Martin Steinebach, Karol Gotkowski, and Hujian Liu. Fake news detection by image montage recognition. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–9, New York, NY, USA, 2019. ACM.

[40] Martin Steinebach, Sebastian Jörg, and Huajian Liu. Checking the integrity of images with signed thumbnail images. *Electronic Imaging*, 2020(4):118–1, 2020.

[41] Martin Steinebach, Huajian Liu, and York Yannikos. Forbild: Efficient robust image hashing. In *Media Watermarking, Security, and Forensics 2012*, volume 8303, page 83030O. International Society for Optics and Photonics, 2012.

[42] Martin Steinebach, Huajian Liu, and York Yannikos. Efficient cropping-resistant robust image hashing. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 579–585. IEEE, 2014.

[43] Martin Steinebach, Huajian Liu, and York Yannikos. Facehash: Face detection and robust hashing. In Pavel Gladyshev, Andrew Marrington, and Ibrahim Baggili, editors, *Digital Forensics and Cyber Crime*, volume 132 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 102–115. Springer International Publishing, Cham, 2014.

[44] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10. IEEE, 2018.

[45] Theodore Tsesmelis and OpenCV. Introduction to principal component analysis (pca).

[46] Stefan Thiemert, Hichem Sahbi, and Martin Steinebach. Using entropy for image and video authentication watermarks. In *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, page 607218. International Society for Optics and Photonics, 2006.

[47] K. K. Thyagharajan and G. Kalaiarasi. A review on near-duplicate detection of images using computer vision techniques. *Archives of Computational Methods in Engineering*, 28(3):897–916, 2021.

[48] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection.

[49] Yunchao Wei, Xiaodan Liang, Yunpeng Chen, Zequn Jie, Yanhui Xiao, Yao Zhao, and Shuicheng Yan. Learning to segment with image-level annotations. *Pattern Recognition*, 59:234–244, 2016.

[50] Bian Yang, Fan Gu, and Xiamu Niu. Block mean value based image perceptual hashing. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 167–172. IEEE, 2006.

[51] Bian Yang, Fan Gu, and Xiamu Niu. Block mean value based image perceptual hashing. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 167–172. IEEE, 2006.

[52] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C. H. Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[53] Christoph Zauner, Martin Steinebach, and Eckehard Hermann. Rihamark: perceptual image hash benchmarking. In Nasir D. Memon, Jana Dittmann, Adnan M. Alattar, and Edward J. Delp III, editors, *Media Watermarking, Security, and Forensics III*, SPIE Proceedings, page 78800X. SPIE, 2011.

[54] Fanfeng Zeng, Shengda Hu, and Ke Xiao. Deep hash for latent image retrieval. *Multimedia Tools and Applications*, 78(22):32419–32435, 2019.

[55] Shiliang Zhang, Qi Tian, Ke Lu, Qingming Huang, and Wen Gao. Edge-sift: discriminative binary descriptor for scalable partial-duplicate mobile search. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 22(7):2889–2902, 2013.

[56] Jiaojiao Zhao and Cees G. M. Snoek. Liftpool: Bidirectional convnet pooling.

Biographies

Martin Steinebach is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. From 2003 to 2007 he was the manager of the Media Security in IT division at Fraunhofer IPSI. He studied computer science at the Technical University of Darmstadt and finished his diploma thesis on copyright protection for digital audio in 1999. In 2003 he received his PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt. He gives lectures on Multimedia Security as well as Civil Security. He is Principle Investigator at ATHENE and represents IT Forensics and AI security. Before he was Principle Investigator at CASED with the

topics Multimedia Security and IT Forensics. In 2012 his work on robust image hashing for detection of child pornography reached the second rank "Deutscher ITSicherheitspreis", an award funded by Host Görtz.

Tiberus Berwanger worte his thesis on the paper topic as a masterstudent of TU Darmstadt at Fraunhofer SIT. In 2021, he received his M.S degree in IT-Security from Technical University Darmstadt. He is currently working as a Cyber Security Specialist at SICK AG.

Huajian Liu received his B.S. and M.S. degrees in electronic engineering from Dalian University of Technology, China, in 1999 and 2002, respectively, and his Ph.D. degree in computer science from Technical University Darmstadt, Germany, in 2008. He is currently a senior research scientist at Fraunhofer Institute for Secure Information Technology (SIT). His major research interests include information security, digital watermarking, robust hashing and digital forensics.