UNIVERSITEIT UTRECHT

MASTER THESIS

Dense skeletons for corner detection

Author:
Diego Renders
Student number:
5740894

Supervisors:
prof.dr.ir. A.C. Telea
dr. M. Behrisch

A thesis submitted in fulfillment of the requirements for the degree of Computing Science

in the

Faculty of Science
Department of Information and Computing Sciences

March 4, 2021

UNIVERSITEIT UTRECHT

Abstract

Faculty of Science
Department of Information and Computing Sciences

Computing Science

Dense skeletons for corner detection

by Diego Renders

Within the topic of feature detection, corners have been important as stable repeatable features. Finding corners has been the goal of numerous feature detection algorithms because of a corner's distinctiveness. Methods such as the Harris and FAST corner detectors are the best known examples. In this thesis, the detection of corners using skeletons is discovered. While normally skeletons are only defined for binary shapes, recent research has extended their use for grey and coloured images. The proposed corner detector is evaluated and compared against popular corner detectors, showing competitive results in terms of the robustness and completeness.

Contents

Chapter 1

Introduction

Corner detection is a task used for many different applications. Such tasks include matching, tracking, and motion estimation [23], localization and mapping, Structure-from-Motion, camera calibration, and image matching [32]. Some more applications include image representation, image classification and retrieval, object recognition and matching, motion tracking, and several others [12]. Corner detection is thus an important part of many computer vision tasks and is often the preliminary task in an application [38]. Corners are chosen as salient points because of their stable and distinctive properties. However, there is no exact definition for a corner in the literature, because it is not important. The goal of corner detection in the before mentioned vision tasks is to reliably find the same features in multiple images. In this case, the corners are a means to an end, and therefore the definition of a corner is not important, as long as they can achieve the goal of the respective task. Further discussion of the definition of corners used in the literature is given in section 2.1.

Skeletons have been used as an image representation that captures geometry and topology of a shape, and their calculation has been a widely studied research topic with efficient algorithms existing for their extraction [7]. Using the topological information of these skeletons, the corners of a shape seem to correspond very well with the endpoints of the skeleton branches. It would thus seem natural to use skeletons to find corners of shapes. One problem that skeletons have is that they respond strongly to noise and using every endpoint would not lead to stable and distinctive features. Instead, a method is required to extract only the important and salient corners of a shape. This is is done by Telea [36] who uses skeleton simplification, such that only the most salient branches remain and for which the endpoints correspond with the salient corners of an image.

The goal of corner detection is to find corners in grey-scale and color images. However, skeletons are only defined for binary shapes. Recent research has extended the use of skeletons to represent grey-scale and color images by using Dense Medial Descriptors (DMDs) [39]. The DMD uses the threshold set representation of an image so that a binary skeleton can be generated for each of the threshold layers. This new image representation allows for the processing of grey-scale images using skeletons.

Using the skeletons of a binary to find corners has already been done. First of all, Dinesh and Guru [6] propose a corner detector for binary shapes by finding the intersections of the skeleton with the boundary. Secondly, Telea [36] used skeletons to smooth images by only keeping the most salient corners. With previous research already being done to find corners using skeletons, it would now be interesting to see how well they can be used to find corners in grey-scale images using the DMDs. The method of obtaining corners from the DMD of an image will be investigated in this thesis.

Based on earlier work done on binary skeletons and their use for grey-scale images, the research question is:

"How can dense skeletons efficiently and effectively be used for corner detection in grey-scale and color images?"

The effectiveness of the corner detector will be assessed for different scenes and images. Furthermore, the effectiveness is evaluated relative to other popular corner detection methods.

First, an overview of existing literature is given in chapter 2, establishing what a corner is considered to be, and next describing several corner detector methods. Secondly, the literature on the computation of binary skeletons and their use extended to grey-scale images is given in section 3. Using this existing literature on skeletons, chapter 4 will introduce the methodology for using the DMDs for finding corners in grey-scale images. A literature review for evaluating performance for corner detectors is given in chapter 5. The evaluation strategy proposed in this chapter is used and its results are shown in chapter 6. Lastly, these results will be discussed in chapter 7, in addition some possible research will be suggested based on findings in the foregoing chapters.

Chapter 2

Literature Review

In this chapter previous work will be discussed on the topic of corner detection. First, the definition of what a corner is considered to be in the literature will be explained in section 2.1. Once this is established, several corner detection methods are discussed in section 2.2. This is followed by section 2.3 which explains the relation between the use of DMDs and several feature detection methods using the threshold sets of an image. Next, section 2.4 will discus some detectors from recent years that have started to use machine learning. These are mentioned separately, because of the large difference between them and the so called hand-crafted corner detection methods. Lastly, the concept of scale in feature detectors is discussed in section 2.5, because of its close relation to corner detectors and corner detection in generals.

2.1 What is a corner?

A corner, point feature, feature, or interest point are all terms that are used for the same thing [32]. A corner would be a type of interest point. They further define a corner as a point of interest with variation in two dimensions. This similarity between an interest point and a corner is further shown by the interest point detector by DeTone et al. [3]. Their algorithm is described as an interest point detector. However, the terms 'interest point' and 'corner' are used interchangeably and their method is compared to several corner detection algorithms. Furthermore, a distinction is made between two types of corners that are found in images, actual corners of objects [32], and corners that arise from small patches of texture. Mokhtarian and Farahnaz [23] make a clear distinction between a corner and interest point. They define a corner as either an "image point where 2D change can be observed in the image" or "the boundary of a distinguishable region in the image undergoes a sharp change in direction" (p.1). Especially the latter of these two definitions is vague, because both 'distinguishable' and 'sharp' are subjective descriptions of a corner. They distinguish a corner from an interest point by saying that a corner is visually distinguishable by humans while an interest point is not necessarily. A corner could be classified as a specific interest point.

Not many different types of corners are mentioned in the literature. Although some distinction is made, such as a L-, T-, Y-, or X-corner [32, 33]. However, these types of corners are neither defined nor used in evaluation. The term T-corner is also used more extensively by Mokhtarian and Suomela [24], in their case this type of corner was harder to find with their method used and needed additional attention. However, this is the only type of corner mentioned, and is not used in evaluation. Awrangjeb and Lu [1] mention three types of corners: (1) strong, (2) weak (round), and (3) false. Strong corners are sharp corners. Weak corners are local maxima, but not sharp enough to be considered a corner. Finally, false corners are not actual corners but are caused by noise.

Corner detection can be classified as a form of feature detection. More specifically, it can be classified as a local feature detection, in contrast to global features such as color histograms. Local feature detection aims to find local regions in an image. Alternatives for this are image segmentation, random sampling, and sliding over the image with a window. A recent survey on local features [38] further distinguishes several types of local feature detection, of which the first are corner detectors. These are methods that aim to find the corners in an image. Secondly, there are blob detectors which aim to find some region with approximately constant intensity. The third are region detectors which find arbitrarily shaped regions. Important to note is that these categories only indicate the type of image pattern the method respond to, not the actual shape of the features.

In conclusion, no actual formal definition is given for a corner, and even less for an interest point, which quite literally is just an interesting point. Furthermore, corner and interest points are used somewhat interchangeably. The reason for this is because what a corner or interest point is, is not very relevant. It is only relevant that it is an important point that can be reliably found in multiple viewpoints. Points that are located on edges could also be labelled as interesting points. However, a point located on an edge cannot be reliably found across multiple viewpoints, because it is very similar to the other points along the edge. This can be further illustrated by considering a rectangular shape, which has in total four corners, but an infinite number of points along the edges. In this case it would thus make sense to use the corners as the reference points instead of the edges. It is said that a corner is easier to localize compared to a point on an edge.

2.2 Corner detectors

In this section some important hand crafted corner detectors will be discussed. The first one is SUSAN [35] which looks at a local neighbourhood around each point and classifies it as being a corner or not. This is followed by an extension of SUSAN, namely FAST [30], including some further extensions that have been made to this method. Additionally, there is the Harris corner [11], arguably the most important corner detector in the literature and still being used as comparison for evaluations. A notable exclusion are contour based methods, that look at the curvature of edges to detect corners. These are mentioned as one of the types of corner detector by Mokhtarian and Mohanna [23] but have fallen out of favour [38].

2.2.1 SUSAN

The SUSAN corner detector [35] classifies each pixel as a corner by considering a circular region centered on the pixel. This center pixel is called the nucleus. The area in the circle with similar intensity is called the Univalue Segment Assimilating Nucleus (USAN). The value of the USAN (its region size) can be used for detecting both corners and edges. A high value USAN means that the circle is very uniform in pixel intensity, and thus there is no interesting feature here. A low value USAN would mean only a small part of the circle has the same intensity and thus there is a large dissimilarity in the area, indicating that there could be a corner. The S in SUSAN stands for smallest, because they are interested in finding the points which have a small USAN value.

There are two thresholds for corner detection using SUSAN. The first is to determine when a pixel in the circular mask is similar to the nucleus. Two pixels are deemed similar if their absolute difference is smaller than the threshold t. Counting the number of similar pixels in the area gives the USAN value (region size). The second threshold g is to determine if a pixel is a corner. If the USAN value is smaller than g, the point is classified as a corner. However, the authors mention that this second threshold can be fixed to half of the maximum size of the USAN. The algorithm looks for points with a USAN below a certain threshold. Many points in the location of a corner would satisfy this condition. The final step is to perform NMS resulting in a set of detected corners.

The strengths of SUSAN are good performance when noise is present because no image derivatives are calculated. The authors mentions two types of thresholds, one for quality and the other for quantity. The threshold g is mainly concerned with the quality, i.e. a lower USAN value means that the corner is sharper. The threshold t looks at the brightness differences between pixels in the circular mask and the nucleus. This is a threshold that mostly effects the quantity of detected pixels. It is also mentioned that this threshold is dependent on the contrast in an image, images with low contrast require a smaller t to ensure enough corners are detected. In the skeleton based method these thresholds correspond to saliency (qualitative) and τ (quantitative) and will be discussed in section 4.4.3. Images with little contrast thus also require a lower value of τ. The other solution the authors mention is to return a fixed number of corners, i.e. the top 200 corners, which means that this quantity based threshold is not as important.

2.2.2 Fast

The FAST corner detector [30] is a simple detector designed to be used in real time applications and builds on SUSAN [38]. For each pixel a surrounding circle is examined, similar to SUSAN, but instead only the boundary of 16 pixels are included in the test. The point is classified as a corner if N contiguous pixel intensities are above $I(p) + t$ or below $I(p) - t$ where t is the threshold parameter. Empirical evidence shows that using $n = 9$ yields the best repeatability results [31, 32].

An extension to FAST uses machine learning to accelerate the test [31]. This approach addresses several weaknesses of the original FAST detector. The major weakness being that FAST uses a test that is optimized when $N = 12$, but does not generalize well to smaller values. To address this weakness a decision tree is built that classifies a point to being a corner or not a corner, by looking at the same 16 pixels around a point. The authors mention that this is very close to the original segment test. Another weakness of FAST is that it generates multiple features adjacent to one another. In the previous case with SUSAN this was solved using NMS. The problem that the authors mention with FAST is that detected corners do not have a response value, i.e. a value that quantifies the ’cornerness’. Therefore, a value V can be assigned to each pixel for which the formula is shown below.

$$V = max\left(\sum_{x\in S_{bright}} |I_{p\to x} - I_p| - t, \sum_{x\in S_{dark}} |I_p - I_{p\to x}| - t\right) \tag{2.1}$$

The formula looks at the differences between the point p and all its darker or brighter points on the circle. The maximum of these two summations is taken, because a point can be a corner if its much lighter or darker than its surroundings. An alternative value could be to use the maximum threshold for which a corner passes the segment test [32]. This however will cause many points to have the same value, because of the discrete nature of images.
The importance of having such a response value for every point is that NMS can be used. This technique was also previously used by SUSAN. The problem with having multiple detections in a small area are twofold. First of all, it gives an unfair advantage when comparing different methods [40]. Secondly, it is also not a desirable result for practical purposes, because you would be matching the same area multiple times.

Rosten and Drummond [31] use machine learning to improve the speed of the detector, and Rosten et al. [32] extend the use of machine learning to also increase performance and generalization. Another extension to fast is AGAST [18] which also uses a decision tree to speed up FAST. While this extension to fast use machine learning, it is quite different from the recent neural network approaches and are therefore discussed separately.

2.2.3 Harris

The Harris corner detector [11] extends on the Moravec corner detector [25]. The latter looks at a window of some fixed size and calculates how much changes occur when that window is moved a slight amount. The equation below is the sum of squared differences between all the pixels in the window and the slightly moved window.

$$E_{x,y} = \sum_{u,v} w_{u,v}(I_{x+u,y+v} - I_{u,v})^2 \tag{2.2}$$

The window function w is 1 for the pixels inside the window and 0 otherwise (or rather a Gaussian window is used by Harris et al. [11]). Originally this change was calculated in several directions so that $(x, y) \in (1,0),(1,1),(0,1),(-1,-1)$. Each pixel would thus have four different values of which the minimum is taken. The problem with this approach is that it only looks in several directions and is anisotropic, meaning that it is not rotation invariant. Harris solves this by approximating the shifted intensity using the Taylor expansion.

$$I_{x+u,y+v} \approx I_{x,v} + x\frac{\partial I}{\partial x} + y\frac{\partial I}{\partial y} \tag{2.3}$$

Where the partial derivatives are approximated for each pixel by using the filter (-1,0,1).

$$\frac{\partial I_{x,y}}{\partial x} \approx I_{x+1,y} - I_{x-1,y} \tag{2.4}$$

Thus for each pixel in a window the partial derivatives with respect to the x and y direction have to be estimated. The resulting equation is rewritten using matrices:

$$E(x,y) = (x,y)M(x,y)^T \tag{2.5}$$

$$M = \sum_s \sum_t w(s,t) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{2.6}$$

Where I_x and I_y are the partial derivatives evaluated at points (s,t). The result is that E is no longer directly calculated, but rather the matrix M is analysed. The eigenvalues of the matrix M indicate what the distribution of derivative are. If the eigenvalue of the first eigenvector is large, then there is one direction where the pixels in the window have a high partial derivative. This means that if the window would shift in this direction, many pixel values would change. If both eigenvalues are large, then there are two directions for which this is the case. To detect corners, the assumption is then made that for corners there would be a lot of change in multiple directions. Thus, if both eigenvalues are large the point is likely to be a corner. The corner response function R uses the eigenvalues to assign a value to each point. A high corner response would indicate that a point is likely to be a corner. The pixels that are maximum with respect to its 8 neighbours can then be thresholded to obtain all the detected corners. The original corner response function uses the determinant and trace of the matrix M instead of directly calculating the eigenvalues for computational reasons.

$$R = Det(M) - k \cdot Tr(M)^2 \tag{2.7}$$

The window that is used in the Harris corner detector is constant, thus the same size window would be used in images of different scales. When the scale changes by a factor of 2, the relevant area surrounding a point is also scaled by 2. If the same window size is used in both cases the resulting cornerness value is not scale invariant. This is not desirable if the method should be robust against scale changes.

An addition to the Harris corner detector is the Harris-Laplace corner detector [21], which uses scale space improve robustness against scale changes. A scale space of an image is a sequence of the original image in which each subsequent image is filtered with a Gaussian kernel. Each image in the sequence represents a smaller sized image. The Harris corner detector is used to detect the initial corners in each image scale. Then, the points whose Laplacian is locally maximum in comparison to the same point in the following and previous scale. There are thus two steps: (1) find initial Harris points in all scales (local maximum of Harris value of 8-neighbourhood), (2) retain points that are maximum with respect to scale space. For the second step any kind of value could be calculated (such as the Harris value). However, the authors found that instead using the Laplacian obtains better results. The equation that computes the derivative based function F is the following:

$$F(x) = |s^2(L_{xx}(\mathbf{x},s) + L_{yy}(\mathbf{x},s))| \tag{2.8}$$

Where s^2 is used to normalise the value over different scales, $\mathbf{x} = (x,y)$, $L_{xx}(\mathbf{x},s)$ and $L_{yy}(\mathbf{x},s)$ are the second partial derivatives of the image filtered with a Gaussian filter with standard deviation s. The local maximum is found by comparing $f(\mathbf{x},s_n)$ with $f(\mathbf{x},s_{n-1})$ and $f(\mathbf{x},s_{n+1})$, i.e. it is compared with the scale above and below it. The name Harris-Laplace thus comes from using the Laplacian to select the characteristic scale.

Interestingly the authors evaluate the performance for interest point detection. As previously mentioned the Harris value is not scale invariant because the window has a fixed size. The Harris-Laplace approach is aimed to improve the repeatability of a corner detector, by calculating cornerness values in multiple scales. However, the process also determines the characteristic scales (the scale where the point was found to be maximum along the scale axis), which allows for the detection of scale covariant features (circles). The authors already use this for a matching task. The descriptor of each point is computed at its characteristic scale, meaning that the size of the region for a feature is not fixed.

2.3 Feature detectors using threshold sets

2.3.1 MSER

Maximally stable extremal regions (MSER) [19] is a technique that was developed for the problem of stereo matching. The importance of the method is the properties that the extremal regions have; closed under continues transformation and monotonic transformation of image intensities. Although MSER would classify as a region detector, not a corner detector, it is included in this section because of its relation with dense skeletons. This relation comes from the use of threshold sets. To find the MSERs a component tree is built using a threshold set of the original image. A threshold set of an image T_i is a binary image using i as a threshold. Every pixel in the original image whose value is equal or greater than i will be 1 in the threshold image and 0 otherwise. A connected component of this binary image is then an extremal region. For the threshold image T_0 the connected component is the entire image. A maximal stable extremal region is an extremal region that is stable over some thresholds and locally maximum along the path of the component tree. The stability of an extremal region can be calculated with the following formula:

$$\Psi(R_j^{T+n\Delta T}) = \frac{|R_j^{T+(n-1)\Delta T}| - |R_j^{T+(n+1)\Delta T}|}{|R_j^{T+n\Delta T}|} \tag{2.9}$$

Where $|R_j^T|$ is the number of pixels of a region in the threshold level T, and δT is the increment step size of the threshold. For example, if the step size is 10, then to determine the stability of a region, it is compared to the corresponding region 10 threshold levels below and above it (corresponding here meaning that is located on the same path in the component tree).

The extremal regions have two important properties: (1) they are covariant with affine transformations, (2) and secondly the set of extremal regions is closed under monotonic changes in image intensity [38]. The skeleton detection method calculates skeletons over these extremal regions. Therefore, studying these properties gives valuable insight. Knowing that the extremal regions have these properties it would indicate the skeletons, and thus the corners, also have the properties. One of the limitations of MSER is the limited amount of regions found. Using skeletons relaxes the stability constraint, because only one corner has to be stable instead of the entire region. This should result in more detected corners. Furthermore the regions found are very irregular, they can be of any shape and therefore have to be adjusted in feature detector evaluation methods.

MSER improved

Ravindran and Mittal [28] mention the problems when matching parts of an image that lie on the boundary when the background changes. This is for two reasons. First, it is hard to match regions on an object boundary because the background can change. Secondly, it is hard to reliably find the same region in different images of the same scene, for the same reason. Level lines naturally represent object boundaries and are thus used for this purpose. Instead of finding stable extremal regions, they only look at stable parts of a level line. The authors note that MSER finds closed smaller level lines, while their method finds smaller stable parts of large level lines. The stableness of a part of a level line is similar to the stability in MSER, where a level line segment is compared with segments in a level set higher or lower (the distance between two segments is calculated using the area between them). Then, corners are found by centering on candidate points on the maximally stable segments, and for each of them calculating the Harris cornerness value on the covariance matrix. This covariance matrix contains the variances of the points along the curve, thus not image gradients of a rectangular or circular region, as is done with the Harris corner detector. The final corners are points on maximal stable level line segment, who are also a local maximum with respect to other points on the segments. The method is only evaluated for tracking and matching. They call their method CoMaL (Corners on Maximally-stable Level Line Segments), which perfectly summarises it and also shows the likeliness with corner detection using skeletons.

The same principle is used by Perdoch et al. [27], in their method only a part of a level line is required to be stable. Frames are constructed on the level lines based on two properties that are preserved under affine transformation, namely their bi-tangent lines and inflection points. Frames are then grouped based on their distance distance and the stability of such a group is determined by the number of frames it contains.

In short these two methods have two common properties:

1. They improve on MSER by requiring only the stability of parts of level lines.
2. The stability of features is based on the distance to its neighbours in level sets above and below.

Furthermore, the former can be classified as a corner detector and the latter as an affine covariant region detector. The first method is very similar to the skeleton corner detection method, because it finds corners on stable level lines. The skeleton corner detection method finds corners that are caused by stable segments of the level lines.

2.4 Machine learning methods

In this section, two recent neural network approaches will be discussed. Both have been compared in a recent large scale evaluation [14] showing very competitive results even though both method are still in their infancy. The first of the two methods, TILDE [40], was trained largely for illumination changes. The second method, DNet [15], focuses directly on the covariance constrain.

2.4.1 TILDE

One of the first machine learning methods for corner detection is TILDE [40]. The goal of TILDE is to find repeatable key-points under severe illumination changes and noise. These changes can be caused for example by different weather, season, and light. They regard corner detection as a regression problem, not a classification problem. For each pixel location it gives a response value. The resulting regression map is used by finding local maximum that correspond to the detected corners. To obtain the training set, images are first obtained from outdoor web-cam scenes with a static viewpoint but the images being taken on different times of the day and in different seasons. From these images positive training samples are obtained by detecting features using SIFT, and keeping only those features that were detected on most of the images in a sequence. The negative samples will be patches of the image far away from the detected features.

2.4.2 DNet

Another machine learning method that regards feature detection as a regression problem is proposed by Lenc and Vedaldi [15]. However, this approach has significant differences. The authors do not assign a 'confidence' value to each pixel using regression. The approach used is to learn a transformation, which can be seen as a regression problem, because each parameter of a transformation matrix can have values on a continuous scale. Arguably the most important aspect of a feature detection is its ability to detect the same features under different transformations. This covariance constraint is used directly in their learning objective. A function ϕ is learned that given an image patch x outputs a transformation matrix.

$$\phi(x) = \begin{bmatrix} a_u & b_u & p_u \\ a_v & b_v & p_v \\ 0 & 0 & 1 \end{bmatrix}$$

The covariance constraint is the following:

$$\phi(gx) \circ \phi(x)^{-1} \circ g^{-1} = 1 \tag{2.10}$$

Where g is some transformation, this constraint is similar to the idea used normally in corner detectors, where if a corner is detected on a point p then it should also be detected in the images transformed with H at the location Hp. The variables in the learned transformation matrix are the output of the neural network. Training samples are triples (x_1, x_2, g) consisting of an image patch x_1, a transformation g, and the transformed image patch x_2. The samples are obtained from the ImageNet dataset by extracting twenty random crops from images, keeping only those crops which are useful (uniform or low contrast crops are not useful).

The following loss function is used:

$$d^2(r, 1) = \sum_{q \in Q} ||g\phi(x) - \phi(gx)q||_F^2 \tag{2.11}$$

Where q is some residual transformation not covered by the learned transformation. The authors mention that this can usually be substituted by a constant value. $|| \cdot ||_F$ denotes the Frobenius norm which is the square root of the absolute squares of its elements [1]. This measures how well the covariance constraint is satisfied (i.e. the differences between the matrices). To learn the function ϕ a deep neural network is used and its parameters are optimized using the above mentioned loss function.

The authors describe a general approach to the problem that can handle multiple types of transformations, i.e. not all the parameters of the transformation matrix have to be learned. In their implementation only translations are considered, thus only two parameters of the matrix have to be learned.

In order to obtain an actual list of detected features, the function ϕ is convolutionally applied over the entire image. At each location a translation matrix is found which translates the center to a feature point. Multiple nearby feature points will be mapped to a single feature. Thus, the resulting features are those where multiple translation matrices of patches have pointed to, and their stability is given by the number of them. An important part of their contribution is the framework that they have proposed, which would allow future work to use deep learning models to learn affine invariant features and scale selection by using the covariant constraint to directly learn the relevant parameters of the transformation matrix.

2.5 Scale

So far corner detection algorithms are discussed which use points as features. Often these features are covariant with rotations and translations. However, the corners contain no information about the scale. In computer vision tasks where corner detection is preliminary step, it is often followed by computing descriptors for each feature. Such descriptors are calculated over an image area, but if the features contain no information about the scale of the corner, these descriptors will be calculated over a fixed size region. Because these corners contain no scale information they are often referred to as not being scale-covariant [38]. While not being scale-covariant, corner detectors should still be robust to scale changes, i.e. the same corner locations should be found in different images, even though they will not be assigned a certain scale.

In the following section the strategy of other methods to handle scale changes will be discussed. In most methods this is done by making use of the scale space of an image. The general idea of the scale representation of an image is to represent an image in a larger scale by smoothing out fine details by using a Gaussian kernel. Because a Gaussian is used to smooth the images, the scale space becomes a one-parameter family with the parameter being the standard deviation. A local maximum in cornerness is thus no longer being found in the (x, y) plane of the image, but in the scale space $(x, y, scale)$.

2.5.1 Scale covariant feature detectors

The two corner detectors FAST and Harris have both been extended to be scale covariant by the use of scale space. This extension for Harris is called Harris-Laplace and was already discussed in section 2.2. Scale space was used to increase the repeatability of the Harris corner detector given large scale changes. This approach could then also be used to provide a scale covariant feature detector, because it found

[1]https://mathworld.wolfram.com/FrobeniusNorm.html

a features characteristic scale. An extension to AGAST using scale space is used in the BRISK method that involves key-point detection, description, and matching [16]. It uses octaves and intra octaves. Octaves are obtained by successive down sampling of the original image with 2, and the intra octaves by successive down sampling with 1.5. Key-points are first obtained (in each layer) by finding local maximum in the 8-neighbour of saliency scores (maximum threshold for which the point is still a corner). Then it should also be maximum with respect to the scores in the layer above and below as well. Some interpolation is done to obtain the scores.

In MSER, scale covariance is obtained by the affine covariance property of the extremal regions [19]. A recent neural network approach proposed a framework that can directly learn the scale of the features [15], or secondly to also use a scale space pyramid. Another neural network approach suggests using scale space to make their NN feature detector scale covariant [40].

Evaluation of corner detectors can easily be extended to the evaluation of feature detectors. The same performance metric, namely repeatability, is used with a slightly different definition. For corner detection a match is two points below a certain distance threshold. For feature detection a match is two regions with an overlap above a certain threshold. For two regions $r, r' \subset \mathbb{R}$ the overlap is calculated as $o(r, r') = |r \cap r'| / |r \cup r'|$ (for finding corresponding pixels between r and r' a homography would be used). This approach is used by [14] who compare local feature detectors. The authors mention that by simply increasing the region sizes, the performance would be increased and thus requires the overlap score to be normalized. Furthermore, the same kind of datasets are used for both tasks, requiring a sequence of images with corresponding homographies [14].

Scale covariance using skeletons would not make use of scale space. Instead the affine covariance property of extremal regions [19] results in skeleton endpoints that should be affine covariant. For determining the characteristic scale of the detected corners, scale covariant attributes of a corner such as the distance transform could be used. This last step is required to transform the point feature to a scale covariant circle shaped feature. Not having to use scale space could be an advantage, because it introduces several more parameters.

2.6 Summary

It was found in section 2.1 that there is no formal definition of a corner in corner detection. Instead, corners are chosen because of their stable and distinctive properties. Therefore, many possible ways to extract these undefined points are possible, as was shown in section 2.2, and none of them have shown to be a clear favourite over the other. This yields room for the creation of better and different methods. For that reason a new corner detector using skeletons is proposed, which differs from existing hand-crafted corner detectors in a fundamental way, namely that it uses a different image representation to find corners, the DMD. A second result of the non-existence of a formal definition for corners is that it complicates the evaluation. How this problem is dealt with by setting up an evaluation strategy will be discussed in chapter 5. Lastly, scale was discussed because it showed how corner detectors in general deal with scale changes, which as a result turns these methods from corner detectors to region detectors (this is at least true for Fast and Harris). This last point is one of the fundamental difference between the traditional corner detection methods and the proposed skeleton corner detection method. The next chapter will extend the literature review to discuss the computation of skeletons and their use for grey-scale images.

Chapter 3

Skeletons

The goal of the proposed method is to detect corners. For this it was argued that skeletons could provide a good descriptor, because its endpoints relate to corners of an object. Therefore, before the proposed skeleton corner detection method is introduced, the preliminaries of skeletons will need to be discussed. This chapter will review existing literature regarding the computation of binary skeletons and the extension to grey-scale images, starting with the formal definition of skeletons in section 3.1. Secondly, an existing method for efficiently computing these skeletons is presented in section 3.2. Furthermore, skeletons are known to respond strongly to noise, and thus a skeleton simplification method is required, which will be discussed in section 3.3. These sections all relate to the computation of binary skeletons. Section 3.4 will discuss how the binary skeletons will be used to represent grey-scale images.

3.1 Skeleton definition

2D skeletons are computed from a binary shape, or more relevant in this case: an extremal region or connected component. A skeleton is the set of points of the shape that are equidistant from the border of the region [10], meaning that there are at least two points on the boundary with equal distance (this is not always the case for discrete shapes). These points are called feature points of the skeleton point. The closest distance between points on the shape Ω and its boundary $\delta\Omega$ is given by the distance transform $D : \Omega \rightarrow R$, and the feature points are given by the feature transform $F : \Omega \rightarrow \wp(\delta\Omega)$:

$$D(x \in \Omega) = \min_{y \in \delta\Omega} \|x - y\| \tag{3.1}$$

$$F(x \in \Omega) = \{y \in \delta\Omega | \|x - y\| = D(x)\} \tag{3.2}$$

The skeleton then consists of all the points that that have equal distance to at least two distinct points on the boundary. A skeleton of a shape can thus be given as an equation using the feature transform:

$$S(\Omega) = \{x \in \Omega | |F(x)| \geq 2\} \tag{3.3}$$

The Medial Axis Transform (MAT) gives a shape represenation using the collections of skeleton points along with the distance transform. Using the MAT the original shape can be reconstructed with the information of the skeleton points and their distance transform value, meaning that they are a dual representation. This can be done for example by drawing discs with a radius equal to the distance transform on each skeleton point and keeping the intersection of these discs. An algorithm which

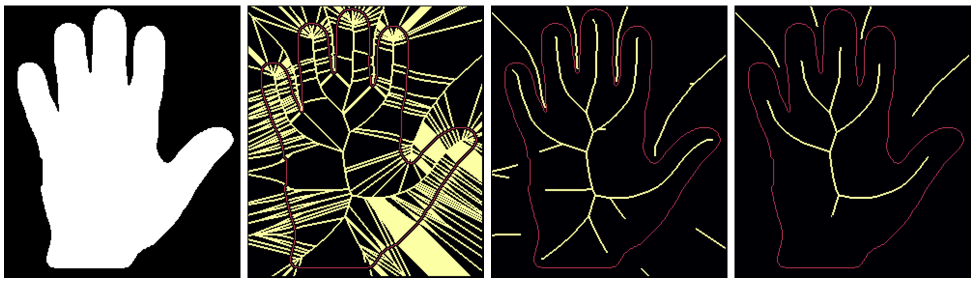


FIGURE 3.1: (a) Binary shape. (b) Obtained skeleton points of the hand shape and the background using no threshold (skeleton points = yellow, boundary = red). Importance threshold of 10 (c) and 100 (d) applied to the original skeleton.

uses the distance and feature transform for computing the skeleton of a binary shape is discussed in the following section.

3.2 Augmented fast marching method

The fast marching method [34] is an efficient way to calculate the distance transform for an image. It uses a 'band' that starts as the boundary of a shape, and moves inside the object, updating the distance transform along the way. The augmented fast marching method (AFMM) [7] is an extension to this, which also calculates the importance of each pixel. The importance of a skeleton point can be described by the length of the boundary segment between its two closest points on the boundary. The FMM method is extended to not only keep track of the distance transform, i.e. the distance to the closest boundary point, but also of the location of this point. This is done by assigning a monotonically increasing number to each boundary point starting with 0. The narrow band that moves inside updates both the distance transform value T of each point and the closest boundary point U. A point x is a skeleton point if the difference with a neighbour y in U values is > 2, meaning that one of the feature points of y is not a neighbour of the feature point of x. This difference in U values estimates the length of the boundary segment between the two feature points of x (the importance). The skeleton can be obtained by thresholding the U differences, and be further pruned by increasing this threshold.
The AFMM method thus computes for each pixel one if its feature points, and estimates the second one by looking at neighbouring pixels. This uses the previously mentioned definition of skeletons where a point belongs to the skeleton if it has two feature points.

3.2.1 False branches

A problem when using the AFMM for computing skeletons is mentioned by Telea and van Wijk [7]. The first pixel on the boundary will be the neighbour of the last pixel labelled. This causes a false branch for every extremal region in a threshold set. The importance is estimated as the difference between the labels of its two feature points (boundary pixels), which in those specific cases is estimated wrong, resulting in a single false branch for each region. A way to solve this is by doing the skeletonization twice [7], both with different starting points of the labellings, and combining the estimated importance of both images by taking the minimum for each pixel.

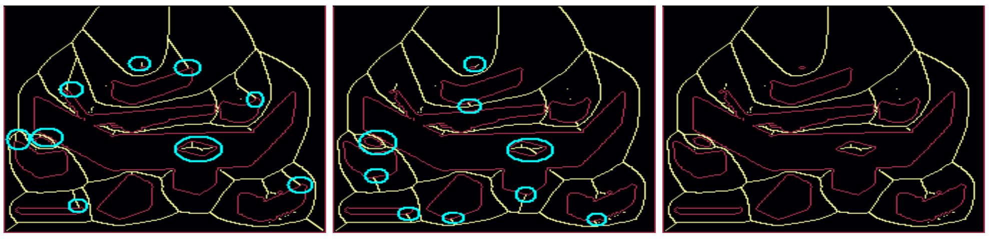


FIGURE 3.2: Image (a) shows the skeletons (yellow) with boundaries (red) when labelling the boundaries top to bottom. Image (b) shows the same for the labelling from bottom to top. Both images have the false branches marked by the blue circles. Image (c) is the bitwise AND operation applied to the two skeletons, meaning that a skeleton point should exist in both image (a) and (b).

Figure 3.2 shows two different labellings in (a) and (b) and their combination in (c), where all the false branches are removed.

3.3 Saliency

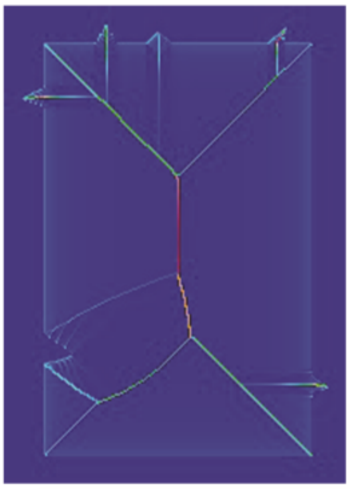
The problem with using solely the importance metric to prune a shape is that it smooths salient corners as much as cusps and boundary noise. In order to simplify a shape by only smoothing out the noise and small cusps, a saliency metric is introduced [36]. The goal is to keep corners sharp by keeping its related skeleton branch, while smoothing out noise by removing its branch. The observation is made that the saliency of a corner is both proportional with size (importance) and inversely proportional with local object thickness (distance transform). The saliency metric σ for a skeleton point is defined as:

$$\sigma(x \in S(\Omega)) = \frac{\rho(x)}{D(x)} \tag{3.4}$$

Where $S(\Omega)$ are the skeleton points of shape Ω, $D(x)$ the distance transform of point x, and $\rho(x)$ the importance value (the difference in U values). This will lead to similar values for the tips of skeleton branches, both for corners and noise, because their distance transform and importance are similar. However, the branch related to the noise has at its inception a low importance value but a large distance to the boundary, causing the saliency value to be lower compared to the tips of the branch. This is illustrated by figure 3.3 in which there is a sharp decline in the saliency value from the main skeleton to some of the branches. These are the points where the importance values are very low, because they are caused by small cusps that only encompass a small distance on the boundary.

Thresholding based on the saliency metric will result in the skeleton becoming disconnected. The salient corners should remain connected to the main shape, while the noise should not. After applying the threshold the main skeleton should be kept and the remaining disconnected segments discarded. The main skeleton is selected by finding the skeleton point with the largest importance value, and selecting the skeleton that contains this point.

FIGURE 3.3: Values of the saliency metric for skeleton points, dark blue = low value



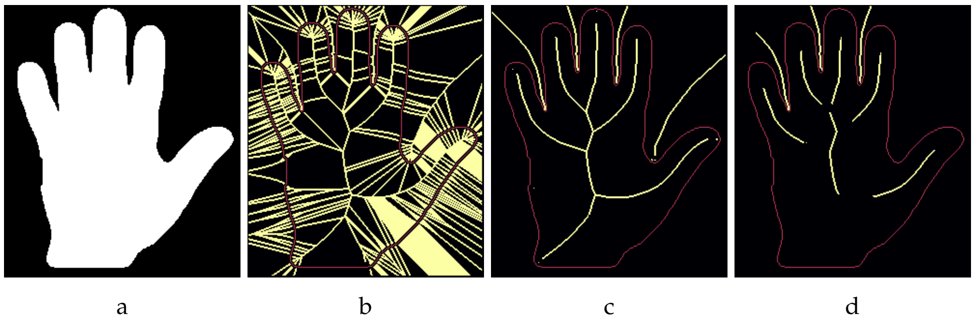


FIGURE 3.4: (a) binary shape. (b) Obtained skeleton points using no threshold (skeleton points = yellow, boundary = red). (c)(d) Skeleton points that remain after applying a saliency threshold of 1.6 and 5 respectively to the original skeleton.

3.4 Skeletons for grey-scale images

The limitation of skeletons is that they can only be computed for binary shapes. In order to generalise skeletons for grey-scale and color images the Dense Medial Axes is introduced by Van Der Zwan et al. [39]. The first step in creating the dense medial descriptor (DMD) is to obtain the threshold sets of a grey scale image. A threshold set contains all pixels in the original image above a certain threshold and is defined as followed:

$$T(v) = x|I(x) \geq v \tag{3.5}$$

Where I is the gray scale image. There are threshold sets equal to the number of possible intensity values in image I. Some important properties of a threshold sets are the following:

1. The size of the threshold set T_0 is equal to the number of pixels in the image
2. $T_i \subseteq T_i - 1$

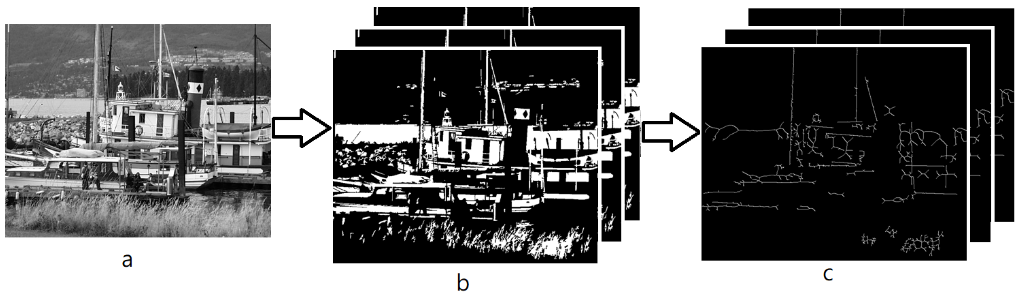


FIGURE 3.5: (a) Original grey-scale image. (b) Threshold set representation of the image. (c) DMD obtained by calculating the skeletons of the threshold sets.

The first of these is obvious, because each pixel has an intensity of value greater than 0. The second property means that the extremal regions cannot gain additional pixels, they can only decline in size. Small regions (both foreground and background) are removed if their region is less than 3% of the threshold's set total area. For every threshold set the MAT will be computed for each of its regions. To reduce the noise in these regions and reduce the size of the skeletons the previously discussed saliency metric is used. This results in a collection of MATs for each threshold layer that can be used to reconstruct the original image. Because of the smoothing in several of the steps the reconstructed image will not be identical to the original. However if all the smoothing steps are excluded (not removing islands and thresholding the salience metric) the reconstructed image should be close to identical.

The first use case for this method is to compress the image [41]. If enough smoothing is done and unimportant layers are excluded, the compressed size will be considerably less than the original while still having the most important characteristics. The interesting thing about this method is that it has a different effect on image quality than normal compression such as jpeg. The second application is segmentation of images by only selecting the most important layers. Finally the skeletons can be used for image manipulation, for example by giving an image an artistic painting effect, or moving around highlights on an object. The DMS method will be further discussed in section 4 including its use for detecting corners.

3.5 Summary

In contrast to corners, discussed in the previous chapter, skeletons do have a formal definition. Therefore the goal of a method that wants to find the skeleton is to efficiently compute it while still satisfying the formal definition of a skeleton. Luckily much research has been done for the computation of skeletons and an efficient method that uses the AFMM was discussed. A second problem, the noisy nature of binary skeletons, was also addressed and the saliency metric was shown to be a good way to keep salient corners while discarding noisy branches. Finally, the extension of binary skeletons for grey-scale images using the DMD is what will be used in the following chapter to detect corners.

Chapter 4

Skeleton corner detection

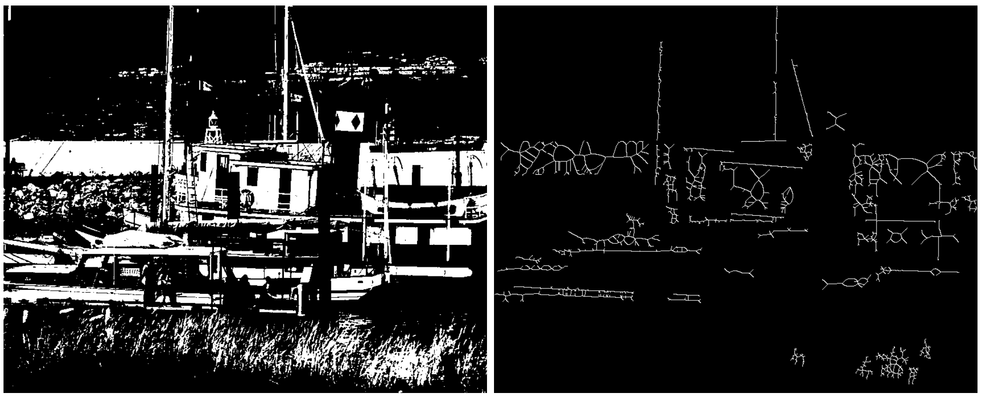
In the previous chapter the preliminaries regarding skeletons were discussed, which now allows the introduction of a novel corner detection method using the DMD. The goal is to use the DMD for extracting corners from images. This consists of two main steps: (1) represent grey-scale images using the DMD, and (2) using these skeletons to detect corners. DMDs were already introduced in the previous chapter, though section 4.1 will discuss their use specifically for corner detection. Sections 4.2 and 4.3 will address some minor practical problems when computing the skeletons. Lastly, section 4.4 will introduce the actual corner detection using the DMD and discusses all the required steps to obtain a list of corners from a grey-scale image.

4.1 Dense skeletons for corner detection

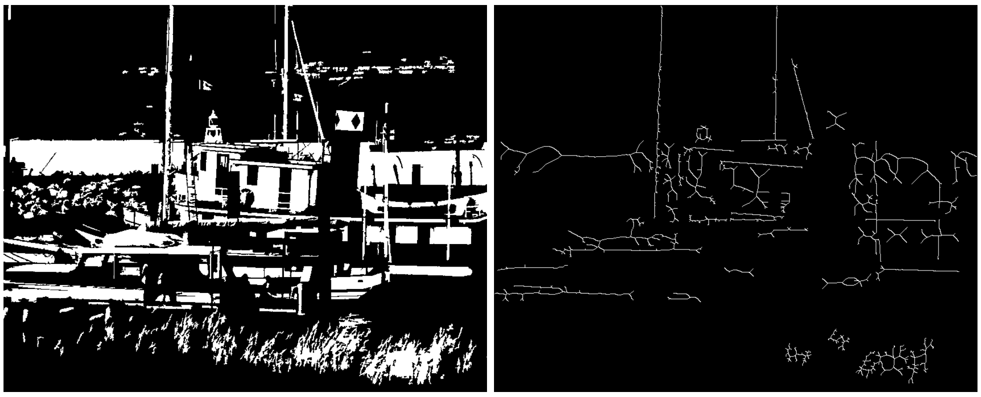
Previous work on dense skeletons show the necessary steps to use skeletons as a way to represent grey-scale images [41]. There are three important steps in this process: (1) removing small islands (connected components in a threshold set), (2) selecting only a subset of threshold layers, and (3) using the saliency metric to prune skeletons. These are discussed in detail below, including their purpose in the detection of corners.

1. **Island removal** This pre-processes the image by removing small islands. Connected components, both foreground and background, are removed from a threshold layer if they are below the island threshold. This technique is used for compression because it leads to simpler images. It would also be useful for the task of corner detection, because it is a way to remove noise. Small holes or islands unnecessary complicate the generated skeletons. However, choosing an island threshold that is too high could lead to corners of small objects no longer being found. In image matching, small objects do not necessary mean that they are not important or not salient. Therefore, the island threshold should be low enough to still remove small amounts of noise but not remove important small objects. The effect of different values for the island threshold are shown in 4.1. As can clearly be seen, using no threshold results in considerably bigger and more complex skeletons. A small black pixel inside a larger white object can drastically change the size and shape of the skeleton. However, between the images using a threshold of 0.005 compared to 0.015, little additional changes occur.

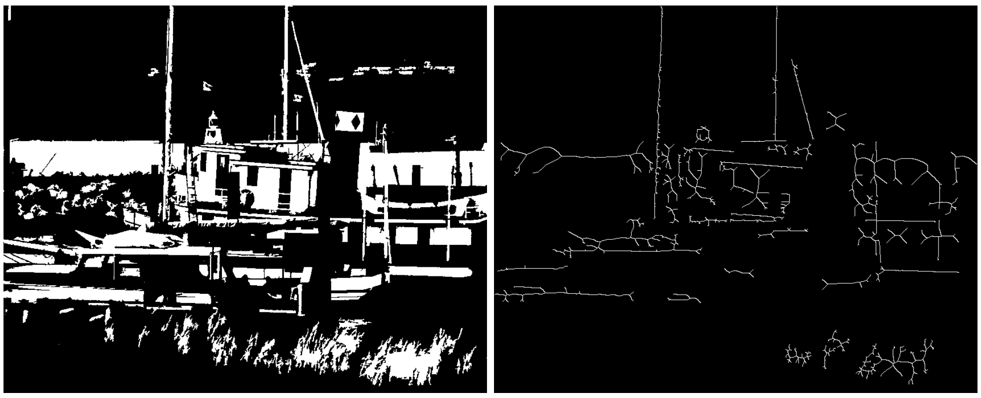
2. **Layer selection** The authors mention that not all layers are required for perceptually good results. Because image compression is the main goal here, selecting as few layers as possible is better. The layers are selected based on their layer importance. Selection of layers based on their importance does not make



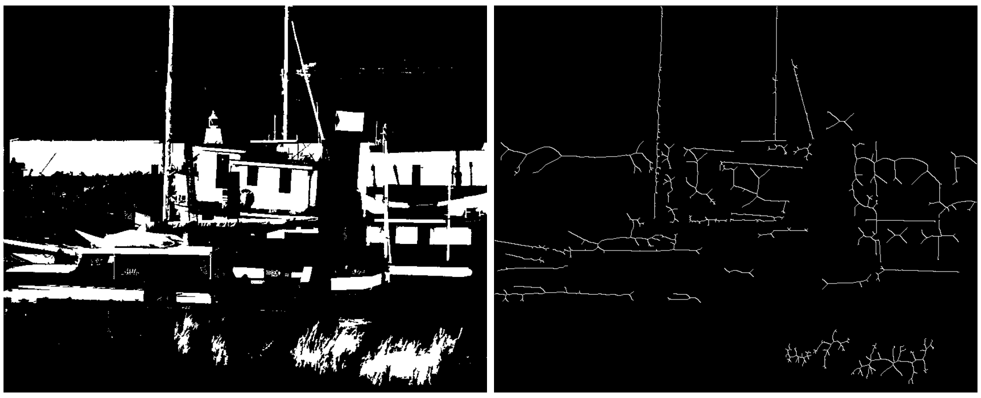
island threshold = 0



island threshold = 0.005



island threshold = 0.015



island threshold = 0.1

FIGURE 4.1: Left column shows the binary image of a single threshold layer for the island threshold values: 0, 0.005, 0.015, and 0.1 respectively. Right column shows the computed skeleton for the binary images on the left. Higher island threshold values remove an increasing number of connected components. Although this leads to less messy skeletons, some of the removed islands could be small objects.

sense for corner detection. A layer's importance is based on its distinctiveness of other layers. First of all, this is a global metric and many corners are of smaller local objects. Secondly, having multiple layers with similar information content is not a downside for corner detection, because finding endpoints in multiple layers is required to detect stable corners. Choosing a subset of the layers is thus less important, because the best choice would be to select all threshold sets. However, reducing the number of selected layers does reduce the computation time. Observing the trade-off between selecting fewer layers and a decrease in performance is an interesting task, and will be investigated in section 6.7. The MSER method for feature detection is similar because it also uses threshold sets. MSER has an initial value for which threshold set to use and in addition a step-size. Important to note is that the τ threshold value used in a later step has to change depending on the number of threshold sets that are used, because if more threshold layers are used a corner has to be detected in more of these layers to be considered a stable corner.

3. **Saliency** The last parameter is the saliency threshold that is used for skeleton pruning. For image compression, this results in much smaller and simpler skeletons, while retaining a perceptually good image. Having simpler skeletons is also beneficial for corner detection, because many skeleton endpoints are caused by very small cusps and noise. These would not be points that cannot be reliably found under different viewing conditions. The effect of the saliency metric is shown in 4.2. A small value for this threshold results in many branches related to non-salient corners of the shape. By increasing the saliency, the skeletons get successively pruned. In the final image a saliency threshold of 5 is used, at which point some of the branches become disconnected from the main skeleton. The skeleton endpoints found from the detected skeletons are also shown. By using a low saliency almost the entire boundary of the shape is covered, which is obviously undesirable because you do not want a corner detector to respond too much to edges. Secondly, the disconnected skeletons in the fourth image result in skeleton endpoints found in the middle of shape. The issue with disconnected skeletons and another issue regarding the skeleton points on the boundary will result in a following section.

   The saliency threshold is arguably the most important parameter for corner detection, because it affects the shape of the skeletons. The saliency threshold is the qualitative threshold [35] that determines how salient the detected corners are. While a lower saliency also increases the number of corners found, it directly affects the quality of corners. The best saliency threshold for corner detection will be discussed in section 6.6.

In conclusion, both the island threshold and the saliency threshold are very important for the corner detection using the dense skeletons. Both have a large effect on the resulting skeletons and thus also affect the skeleton endpoints that are detected. The layer selection, as was argued, is less important for corner detection.

4.2 Boundary skeleton points

From the definition of a skeleton it is clear that skeleton points cannot exist along the boundary. A skeleton point, in the continuous space, is of equal distance to two distinct boundary points. For discrete shapes this is not always the case, which is

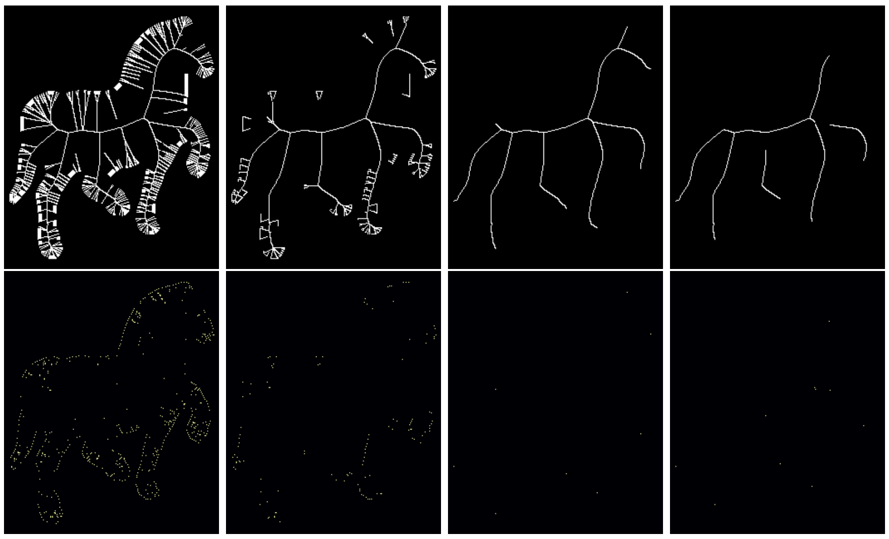


FIGURE 4.2: (top) Skeletons for a binary shape using increasing values of saliency: 0.2, 0.5, 1.5, and 5 respectively. (bottom) Skeleton endpoints found using the above skeletons.

why the skeleton is not always pixel thin. However, in both cases there should not be any detected skeleton branches along the boundary. The classification of these as skeleton points is a result of the estimation of the importance value. As earlier discussed, the skeletonization method used returns the one-point feature transform. To calculate the importance of a pixel both feature points are required, and thus for calculating the importance an estimation is made by looking at neighbouring pixels. This is done for every pixel in an image, thus also for the points along the boundary. The neighbouring pixels of a point along the boundary probably have a feature point with a distance of 1 to the known feature point. This results in an estimated importance of 1. In the first image of figure 4.8 the skeleton lines along the boundary are several pixels thick. This is because the estimated importance of these points is 1. With a distance transform of 5 the saliency is calculated as $\frac{1}{5} = 0.2$, and thus passes the threshold. In the second image of figure 4.8 there are still skeleton points along the boundary, but less, because the pixels with a distance transform > 2 no longer pass the saliency threshold.

The issue can thus be solved in two ways. This can be done by (1) increasing the saliency threshold so that these boundary points no longer pass the threshold (about 1.5 should be enough), or by (2) using an additional distance transform threshold. These wrongly classified pixels along the boundary with a distance transform of 1 have an estimated importance of 1 as well, thus the saliency is $\frac{1}{1} = 1$. Using a saliency of 1.5 should thus be high enough to remove all these skeleton points on the boundary. If a lower saliency threshold is wanted, for example to have more skeleton end points (and thus more detected corners), an additional distance transform threshold can be used. For instance, when a saliency threshold of 1 is used, a distance threshold of > 1 would be enough to remove any boundary skeleton points (with a DT = 2 the saliency of boundary points would be $\frac{1}{2} = 0.5$.

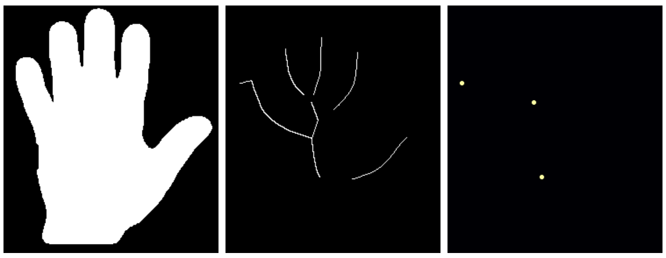


FIGURE 4.3: The disconnected skeleton for a binary shape using a saliency threshold of 5, and the resulting skeleton endpoints obtained by keeping the connected skeleton with the largest importance value.

4.3 Disconnected skeletons

The saliency metric can cause disconnected skeletons. In contrast to using the importance of a skeleton pixel, saliency is not monotonically increasing. It is possible for skeleton points at the start of a branch to be lower than the threshold, while the tips of the branch are above the threshold. This happens for example when the importance stays the same along a portion of the branch, but the distance to the boundary increases, resulting in a smaller saliency. There are two easy ways to fix this issue: (1) keep the largest connected skeleton, or (2) keep the skeleton that contains the highest importance. The latter option will be used. To illustrate this, the skeletonization of a binary shape is shown in 4.3 using a saliency threshold of 5. As can be seen from the last image of the sequence, skeleton endpoints are only found from the skeleton containing the largest importance value.

For a simple binary shape, there is only one object. For more complex images there will be multiple objects or extremal regions in the set of thresholded images. Each of these regions has separate skeletons and thus separate maximum of importance values. This can be seen in figure 4.1. Keeping the skeleton with the largest importance value is thus a process that to be done for each extremal region. This allows the proposed skeleton corner detector to find corners distributed among the entire image, and not just a single object.

4.4 Corner detection using skeletons

The steps and parameters above are those shared with previous methods for dense skeletons. Now the additional steps required for corner detection will be discussed. After the above steps the results are the simplified skeletons for each selected threshold layer.

4.4.1 Finding skeleton endpoints

By tracing each skeleton the skeleton endpoints are found. Due to the way in which skeletons behave, these will not be located exactly on the relevant corner. Sharp corners will have a related skeleton endpoint very close the boundary, while a round corner's skeleton endpoint will be further away. These endpoints can be used with the following two proposed methods:

1. Use the pixel location of the skeleton endpoint. The distance of the skeleton endpoint to the boundary depends on the shape of the corner/bump. This

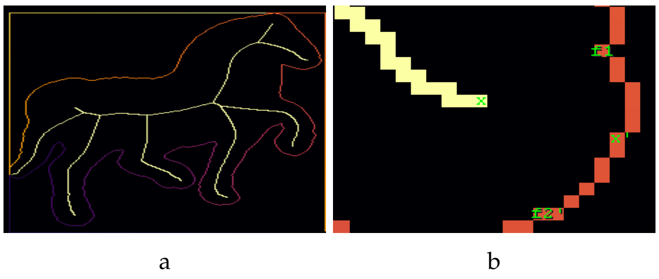


FIGURE 4.4: Adjusting the skeleton endpoint x by using its two feature points $f1$ and $f2'$. The pixel in the middle of the boundary segment between the two feature points is the adjusted endpoint x'.

will yield points that are not actually located at corners, but near them. However, not doing any additional computations or estimations might make the detected corners more repeatable.

2. Estimate the corner that the skeleton endpoint is related to. The feature transform of a point give its closest boundary points. Non-skeleton points will have 1 feature point, while skeleton points have 2. The exact position of the corner can be estimated by using the two feature points $f1$ and $f2$ of an endpoint. The boundary segment between the two feature points (the smallest boundary segment, because there will always be two) represents the bump of this corner. Choosing the point in the middle of this boundary segment gives an exact location of the tip of this bump and can be used as a corner position. The problem with this approach is the following: the skeletonization method only gives a single feature transform point, i.e. either $f1$ or $f2$. A solution to this problem is to estimate the two feature points by looking at the feature transform of points in the local neighbourhood. For example take the one point feature transform of all pixels in the eight-neighbourhood of a skeleton endpoint e. Then for each p such that $p \in N(e)$, add its feature transform $f1_p$ to a list of points F. Then for each pair (u, v) where $u, v \in F$ choose the pair that maximizes the angle between the three points (u, e, v). This pair is used as the estimated $f1$ and $f2$ of the original skeleton endpoint e. Figure 4.4 illustrates this concept.

The difference in performance between these two methods will be evaluated in section 6.8.

The detected corners are collected in an image initialized with 0 values. A pixel's value in which a corner is detected in any threshold layer is increased by one. The number of times a pixel is classified as a corner indicates how stable the corner is. The more stable it is the more contrast is present at the corner location. Contrast is required to detect corners, because it means the extremal region of a threshold layer is stable among several layers. A slow moving gradient means that the extremal region is slowly extending, meaning that the skeleton endpoint is moving as well. Figure 4.8 shows the images with detected skeleton endpoints in row (c). The first two columns are for two binary images and thus all the endpoints have an equal value. The third and fourth column have different levels of contrast and thus the endpoint responses vary.

0.004	0.016	0.023	0.016	0.004
0.016	0.063	0.094	0.063	0.016
0.023	0.094	0.141	0.094	0.023
0.016	0.063	0.094	0.063	0.016
0.004	0.016	0.023	0.016	0.004

0	0	0	0	0
0	0.063	0.125	0.063	0
0	0.125	0.25	0.125	0
0	0.063	0.125	0.063	0
0	0	0	0	0

FIGURE 4.5: A 3x3 and 5x5 gaussian filter, all values are rounded.

4.4.2 Density map

The same corner will likely not be detected at the exact same location in subsequent threshold layers. Therefore, these corner responses on different pixel coordinates should be combined into a single corner. One way to do this is by grouping neighbours together into a single connected component. However, sometimes the distance between the skeleton endpoints (that refer to the same object's corner) is more than a single pixel, in which case there would be multiple connected components for a single corner. To prevent this, a density map is created of the image. It combines all the detected skeleton points in the image into regions.

To create the density map, the image containing the skeleton endpoints is convolved with a Gaussian filter. This smooths the image and thus merges different nearby corner responses into a dense region. The size of this filter should be so that responses for the same corner are merged. Using a filter that is too big, depending on the distance in pixels between endpoints, can merge responses for different corners. The smoothing of the endpoint image will create high density areas on locations where multiple skeleton endpoints have been detected. The Gaussian filter is symmetric and has an odd size (a width of 1 means no changes will be made). For a width of 3 and 5 figure 4.5 shows the weights of the Gaussian filters. Thresholding the density map can increase the number of detected corners, because it can split connected components by removing weak 'bridges'. It can also decrease the number of corners by deleting entire 'islands'.

The theoretical minimum of a pixel in the density map is 0, but the maximum is less obvious. The maximum number of endpoints that can be found for a pixel is 255, thus for a width of 3 the maximum would be $0,25 * 255 = 63,75$, and for w=5 the maximum would be $0,140625 * 255 = 35,859375$. However, it is possible that in a same threshold set a small area can contain two endpoints. For a width of 3 this is not possible (the endpoints would be connected and thus not be endpoints), but for w=5 and larger sized Gaussian the filter area can contain more than 1 endpoint. An obvious problem is that using increased filter sizes result in much lower responses for each pixel, which therefore require thresholds used in subsequent steps to depend on the filter size. To combat this problem the Gaussian can be scale so that its height is equal to one. This is done in the implementation of the skeleton corner detection.

For the images in figure 4.8 a filter size of 5x5 was used. Note that using the filter reduces the values of skeletons endpoints responses significantly, and thus the

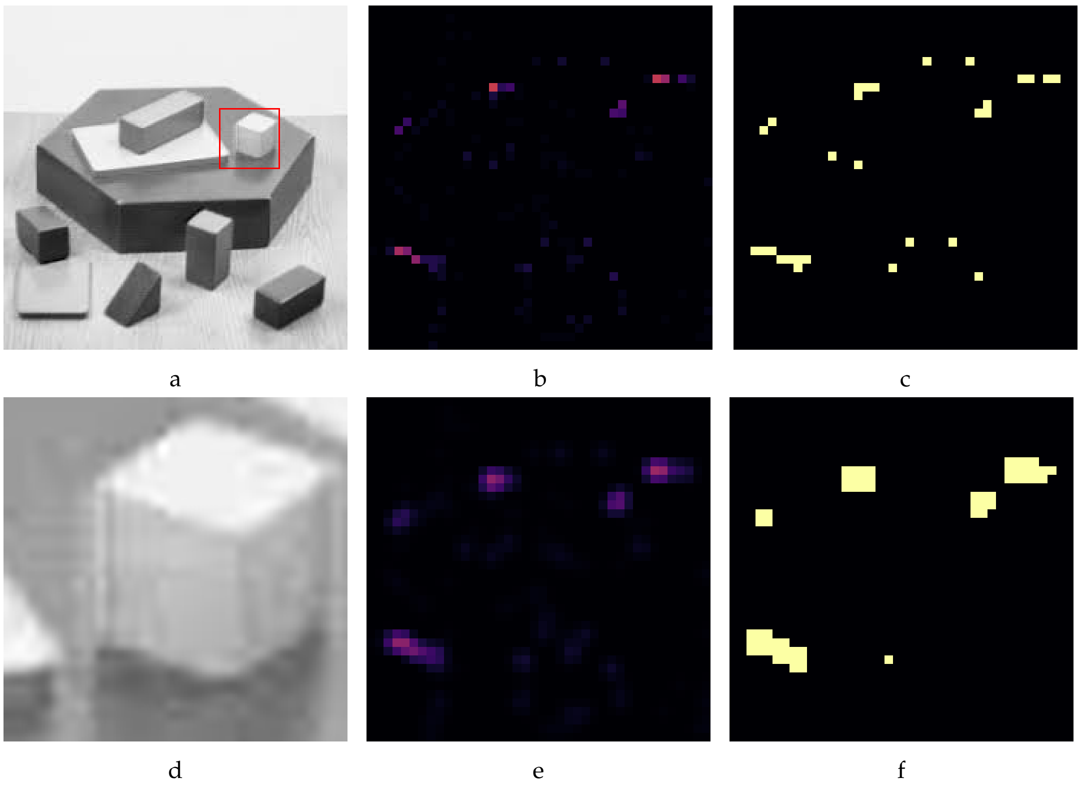


FIGURE 4.6: (a) Original image. (d) Highlighted segment zoomed in. (b) Density map for the highlighted segment without Gaussian smoothing. (c) Thresholded density map of (b) using a threshold of 3. (e) Density map using a 3x3 Gaussian filter. (f) Thresholded density map (e) using a threshold of 1.5.

threshold that is used in the next step should depend on the filter size used. This is shown by the changes in the maximum pixel value in figure 4.8 from step (c) (skeleton endpoints) to (d) (density map using a 5x5 Gaussian filter). The Gaussian filter is supposed to merge responses that refer to the same corner, but if the filter is too large it can also merge responses that belong to different corners. The problem of merging two distinct corners into one is illustrated with the leaf image in figure 4.8. The bottom two skeleton endpoints are merged in step (d), which is more clearly visible after thresholding in step (e). This indicates that a 5x5 filter might be too large for this image. Furthermore, figure 4.6 shows the density map using a 1x1 (no smoothing) and a 3x3 Gaussian filter for a small image segment. It is clear in this figure that the density map using the 3x3 filter (e) has merged several areas that were separated in the density map using a 1x1 filter (b). This is more clearly illustrated in the images (c) and (f) when a threshold is applied to the density map.

4.4.3 Thresholding the density map

The resulting density map is thresholded with the threshold τ so that only the high density areas remain. This results in an image with multiple connected components, each one representing a single detected corner. For each one its center is calculated which is used as the location of a detected corner. Its value is calculated as the sum of pixel values of the density map, similar to the 'cornerness' used in Harris. This last value allows for an ordering of the detected corners, which is useful because it allows you to obtain a specific number of corners (for example the 100 'best' corners).

This value is the certainty of the detector that a corner is present. Furthermore, τ can be classified as a quantitative threshold, because it directly effects the number of detected corners.

Figure 4.6, images (c) and (f), show the result of thresholding the density map. Note that different filter sizes were used for the smoothing, resulting in significantly different values and thus require a different threshold. This is also the reason why image (f) has less connected components than (c): the Gaussian smoothing has lowered the values for each pixel such that a lower threshold is required to keep these areas.

4.4.4 Cornerness

A corner detector is a function that given an image returns a set of corners. The corners returned can be controlled in two ways. Firstly, by using some threshold dependent on the specific corner detector. The second way is to return a fixed number n corners, meaning the top n corners (ranked based on some assigned value) will be returned. The former does not work well when comparing different methods, because they will not be returning an equal number of corns. The number of corners returned has a high impact on the performance [33]. Therefore, when comparing different detectors the performance is often evaluated on a fixed number of corners. When this is the case the problem is not just to detect corresponding corners in different images, but also to assign the same relative ranking to the two corners. For example, when a corner is ranked 20th in the first image, it should also be ranked 20th in the second image (ignoring corners whose projection is outside of the image window). Therefore, this value (cornerness), assigned to each corners is arguably more important than the threshold used. For example, in a recent local feature detector evaluation [14] the threshold was not relevant and was set at an arbitrary low value such that enough features were returned by the methods. However, for the proposed skeleton corner detector a lower value for this quantitative threshold does not necessarily result in a larger number of corners. This problem will be further discussed and illustrated in section 4.4.5, followed by a solution in section 4.4.5.

4.4.5 Analyzing parameters

To understand the behaviour of the in this chapter introduced parameters a short analysis is given in this section. More extensive evaluation will be done in chapter 6. Two of the parameters used in the corner detection method are the filter width, used to create the density map, and τ, which is used to threshold the density map. One way to analyse the behaviour of different parameter settings is to look at the number of corners obtained, because it is important for a corner detection method to provide enough corners for practical purposes and also for the evaluation. Figure 4.7 shows the amount of corners returned for different configurations of τ and the filter width. The horizontal line $y = 1000$ is drawn as a reference 'required' number of corners. Many corner detection evaluations use similar numbers to compare different methods, thus a method returning less could not be compared for those evaluations. Several observations can be made:

1. When the filter size increases the number of corners for small thresholds decreases. Furthermore, when a higher filter width is used, an increase in τ can result in more detected corners. A higher filter size merges islands in the density map, such that a threshold is required to split them up again. However,

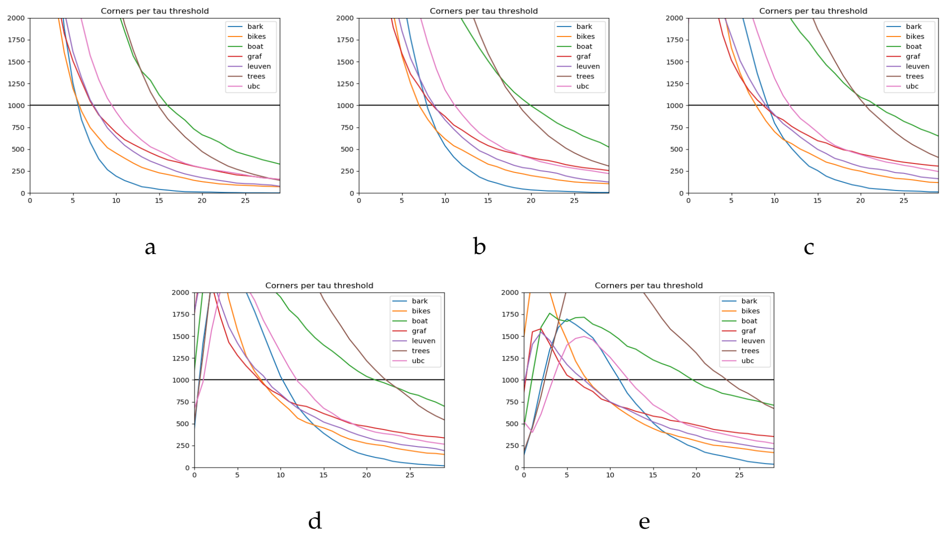


FIGURE 4.7: The images show the number of corners obtained for several different scenes using different τ threshold. The images a-e use a filter width of 1,3,5,7, and 9 respectively. Note that y-axis only goes to 2000, the actual number of corners can go up to 20.000. **X-axis**: value for τ. **Y-axis**: Number of corners found in the image. The saliency threshold is set to 1.5. Note that the number of corners detected is not monotonically decreasing with respect to an increasing τ threshold, which is an unintuitive property of such a threshold.

after the islands are split again their values are generally higher, because the neighbourhood for which endpoints are being found is increased. This causes higher filter sizes to yield more corners for larger values of τ.

2. In general a τ threshold over 10 will not yield enough corners (this 'maximum' τ decreases as the filter width increases).

3. When using a filter width over 1, a threshold is required to split the merged islands, the most optimal threshold would likely be between 2 and 10 (varying slightly based on the filter size).

From the images it can be concluded that τ should be somewhere between 2 and 10.

Alternative to thresholding

An obvious problem with any sort of threshold is that it introduces an additional parameter for the model. The ideal choice of threshold is different for each image. Most likely this ideal threshold is related to the contrast in the image. Getting rid of the threshold would make the method easier to use and to evaluate. The alternative strategy for obtaining a list of corners is the following:

1. Apply the Gaussian filter to the image with the corner responses as was previously done.

2. The value of each pixel is its 'cornerness'.

3. Use NMS to obtain a final list of corners.

4. Rank these corners based on their 'cornerness'.

The advantage would be that you do not need to choose a threshold for the density map. The goal of the Gaussian filter is to combine detected endpoints that are close together, because these would likely refer to the same object corner.

4.5 Summary

This chapter introduced a novel corner detection method using skeletons which uses three thresholds. First, the skeleton computation and simplification discussed in the previous chapter are used and thus it inherits the saliency threshold. Next, the use of DMD specifically for corners detection was discussed which uses an island threshold. Finally, the actual corner detection used the third threshold τ which thresholds pixels based on their response in the density map. However, we found that not using this threshold and instead ordering the pixels based on their response is a better solution. This still requires a parameter to be set, namely the number of corners that should be returned for the image. However, this type of threshold is much more convenient. The images showed that the corners detected using this method correspond with the expected corners in the grey-scale images. Now that this new corner detection method is proposed, an evaluation strategy is required to compare it with the existing methods discussed in chapter 2. The evaluation strategy, based on previous research, is introduced in the following chapter.

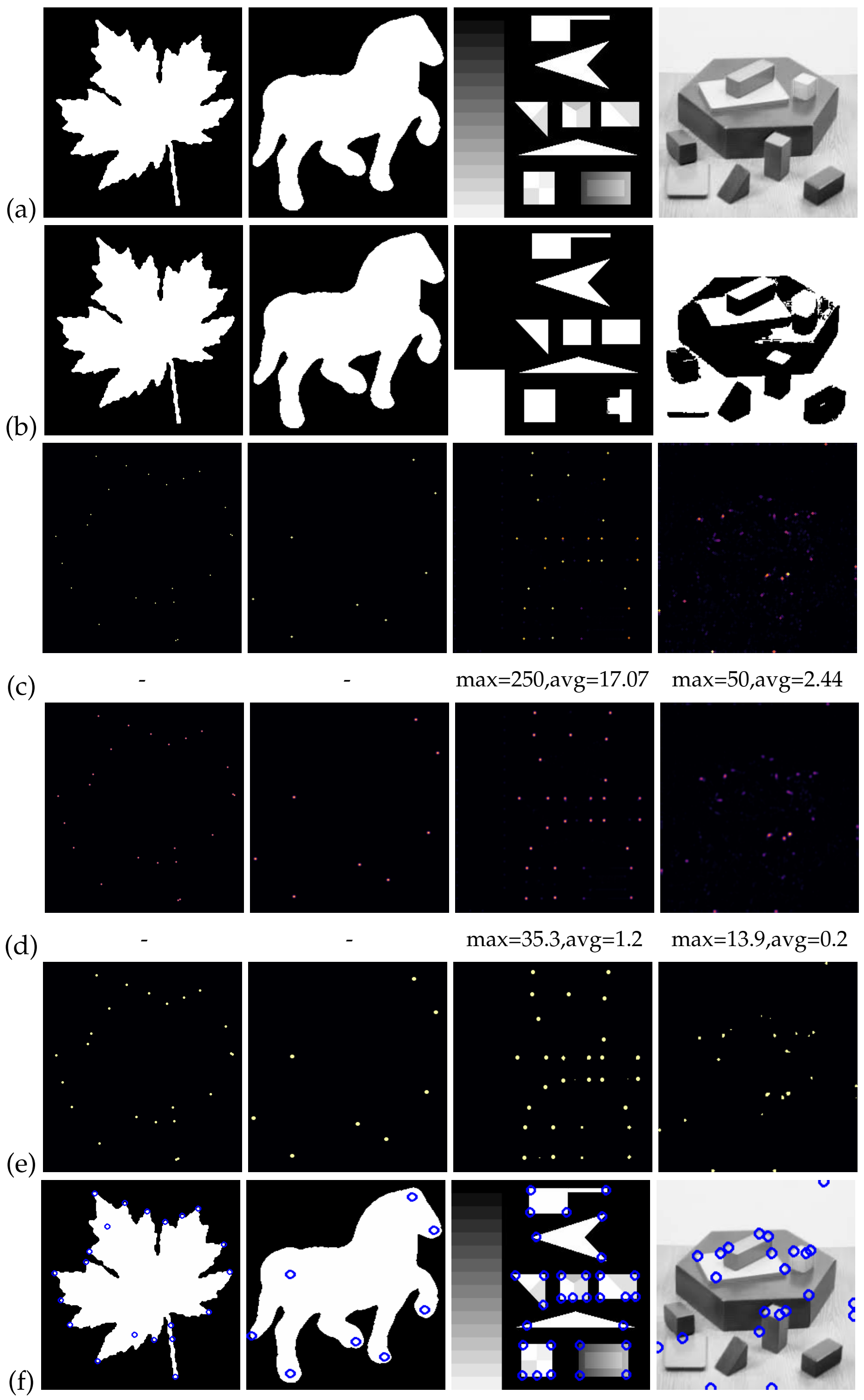


FIGURE 4.8: (a) Original image. (b) One of the threshold layers of the original image (i = 161, note that the first two images are binary and all their threshold layers are thus the same). (c) Skeleton endpoint response value for each pixel (number of skeleton endpoints detected for the pixel). Note that responses are plotted as a 3x3 cross and color-coded such that black means no response and yellow a high response. (d) Density map using a 5x5 Gaussian filter. (e) thresholded density map using $\tau = 3$. (f) Circles plotted centered on the detected corners

Chapter 5

Performance evaluation

In the previous chapter a novel corner detector was introduced. To evaluate the performance of this new method it will be compared with several prominent corner detectors discussed in chapter 2. First an overview of the type of evaluation methods is given in section 5.1 and table 5.1 gives a summary of the evaluation strategies used in the literature. Next, section 5.2 will discuss the generic evaluation approaches in more detail. Section 5.3 discusses the application specific approaches. To use the proposed performance metrics datasets have to be used. The options and characteristics of the available datasets are mentioned in section 5.4.

5.1 Evaluation overview

Rosten et al. [32] mention three ways to evaluate the performance of corner detectors. The first is corner detection as object recognition, meaning that there is a pre-defined set of corners in an image that have to be found by the algorithm. This leads to an obvious problem; there is no clear definition of a corner and thus labelling an image would require the subjective opinion of humans. This problem can be mitigated by using synthetic images, for example renderings of 3d objects, because the position of corners is known and there would be less ambiguity. However, this would not generalise well to natural images [32]. The second evaluation method is a second is a generic approach which aims to find the performance of a detector for several generic properties a corner detector should have. This gives an indication of the viability of a detector, however if the true performance of a detector for a certain task is wanted then an application specific evaluation has to be performed. The second evaluation approach is thus to evaluate the performance for an application, which normally has its own performance metrics.

5.2 Generic evaluations

First of all, what defines a good corner detector? According to [5] the four important properties of a feature detector are: sparseness, robustness, completeness, and speed. They are briefly explained below.

1. **Robustness** is the property where the most attention is given to in the literature. Robustness means how consistent a detector is for different deformations. This is almost solely done by calculating the repeatability, which is explained in more detail below.

2. **Sparseness** is not directly measured, instead most often the number of corners used to calculate repeatability is controlled. It is hard to call a method sparse, because it depends on the setting of its threshold parameter. The result could

Name	**Year**	**Feature type**	**Performance measures**	**Dataset**	**Image type**
EnCovDet [13]	2020	region	repeatability, matching score	hPatches, VGG, EF, WebCam.	real
superPoint [3]	2018	corner	repeatability, homography estimation, matching score, MAP, MLE	hPatches	real
evaluation [14]	2018	both	repeatability.	hPatches, VGG, Webcam	real
DetNet [15]	2017	corner	repeatability, matching score	VGG	real
Comal [28]	2016	corner	vehicle tracking, matching accuracy, nr of matches	KITTI vehicle dataset, KITTI flow	real
Tilde [40]	2015	corner	repeatability, speed	VGG, webCam, EF	real
AGAST [18]	2010	corner	speed	Five scenes	real
SFOP [9]	2009	region	repeatability, localization error, nr of feature	Three scenes	real
FAST [32]	2008	corner	repeatability, speed, area under repeatability curve	VGG, 'bas-relief', 'maze', 'box'	both
CPDA [1]	2008	corner	repeatability, localization error	23 images, synthetically transformed.	both
SAF [27]	2007	region	repeatability, nr of regions, image coverage [1].	VGG	real
FAST [31]	2006	corner	repeatability	'box', 'maze', 'bas-relief'	real
evaluation [23]	2005	corner	consistency, accuracy	five scenes, synthetically transformed, ground truth corners.	both
evaluation [22]	2005	region	repeatability	VGG	real
Harris laplace [21]	2001	both	matching, indexing	unknown	real
evaluation [33]	2000	corner	repeatability, information content	two scenes	real planar scenes
CSS [24]	1998	corner	visual inspection, speed	4 images	both
Susan [35]	1997	corner	visual inspection	several images	both
Harris [11]	1988	corner	visual inspection	one scene	real

TABLE 5.1: Summary of the performance evaluation used for local feature detectors and the datasets that are used. The entries with name evaluation are papers that compare several feature detectors, the remaining ones are the names of the detector. Image type refers to the type of images used for evaluation: real or synthetic. A small overview of some of the most prominent datasets is covered in table 5.2.

be sparse for some arbitrary parameter setting, and not at all for a different one. Therefore, a detector's performance is normally shown as a function of the number of corners n considered. The sparseness is thus controlled, and the result shows the detector's performance for different levels of sparseness.

3. **Completeness** is the least prominent property evaluated in the literature. It is concerned with how much information about the original image is contained in the set of detected features. Completeness is often neglected in evaluation, as is evident from the summary of detector evaluations in table 5.1. In this table only two performance measures related to this concept are used. Perdoch et al. [27] look at the image coverage, or the spatial distribution of the detected features. However, their method is informal and solely a visualization of the image coverage. A second way of measuring completeness is done by Schmid et al. [33] who use the information content of features.

4. **Speed**. Self explanatory and has little relation to the other properties.

Performance metrics should aim to the describe a detector's behaviour regarding the above mentioned properties. The evaluation methods present in the literature are discussed in more detail below.

5.2.1 Repeatability (robustness)

Repeatability directly measures the robustness but has several problems which are listed below:

1. One of the problems with repeatability is an inconsistent definition. Although having a clear intuition, definitions used in the literature have slight differences. Thus the question arises what the exact definition should be.

2. A second problem is how many corners should be selected to calculate the repeatability. As was found in the literature review, comparing repeatability of methods when a different number of corners is used is not a fair comparison. Whatever number is used, it should be consistently used for different methods. As was also discovered, the repeatability is often calculated over a range of corners to control for the sparseness. This leads to question what this range should be. However, no actual reasoning is given about why a certain range is used.

3. The final problem is how to aggregate the repeatability. One option is to simply average the repeatability over all different scenes, severity of the deformation, and the number of corners used. For the latter of these, a subset of the range of number of corners has to be selected. Lenc and Vedaldi [14] select two values: 200 and 1000.. Another option is to calculate the area of the repeatability curve [32], which requires a larger range of number of corners.

The first and last problem are further discussed in section 5.2.4, where our definition of repeatability is given and another performance metric is introduced that aggregates the repeatability results for a single image pair. The second problem does not have a clear solution, the range of number of corners in the literature is arbitrarily chosen. Therefore, using a large range, from 0 to 1000 seems like a reasonable choice.

5.2.2 Completeness

Completeness has one main problem: there is no precise definition of completeness. Two examples of measuring completeness is by (1) looking at the image coverage [27], and (2) calculating the information content [33]. These two methods call a number of detected corners as 'complete' if they cover a large image part, or contain the most information respectively. The completeness is thus obviously dependent on the sparsity of the result, and should thus be controlled. At the very least a consistent number of corners should be used for different methods, and ideally evaluated for a large range of numbers.

The methods aimed at measuring the completeness vary wildly, and there is no established method in the literature. A few methods proposed in the literature will be discussed next. Ehsan et al. [8] use the harmonic mean of all the distances for each interest point. Points are used to calculate distances, however the method is used to compare the coverage of region detectors, using the central points of each region. The coverage value depend on the image dimensions and is therefore not an intuitive metric when using multiple scenes/datasets. Dickscheid and Förstner [4] use the area of the convex hull of the features as a percentage with regards to the total image area. Dickscheid et al. [5] mention that simply using the image area covered by the features is not a good way to measure the completeness, and propose a method that uses image coding theory. This last method is perhaps more similar to that used by Schmid et al.[33] (information content) while the former is more similar to the method use by Perdoch et al. [27] (image coverage).

What all these methods have in common is that they propose a new metric to assess the completeness of a detector. Another approach is done by Rey-Otero and Delbracio [29], where the repeatability is changed so that it measure both the robustness and completeness. This is done by penalizing a detector for providing redundant features, i.e. features that cover the same image area. They call this the non-redundant repeatability. Note that this looks at the overlap of regions, thus if it were to be used for comparing corner detection methods, a fixed size region could be assigned to each corner. The results when comparing different feature detectors show both large absolute differences (the non-redundant repeatability is always lower), as well as relative differences (some detectors rank higher for the non-redundant repeatability than for the normal repeatability).

Visual analysis

One way of performance evaluation that is frequently used is to visually compare the results of detectors, i.e. their detected corners drawn on the original image. This can also be seen as a way to measure the completeness. A method for which the corners are either nonsensical, all concentrated in a small area, or caused by noise/textures is not complete. Visual analysis can thus give an indication of the completeness of a method, by analysing how well the corners are distributed and located at important objects. A downside is that it is hard to compare the results for different levels of sparsity, because using a large range of number of corners is infeasible, because it leads to too many images.

5.2.3 Sparseness and speed

As previously mentioned the sparseness is often controlled when calculating repeatability, because the repeatability depends heavily on the number of corners being returned. The same is most likely true for the completeness, when the number of features are increased the completeness increases as well. This is not true for the speed of the detector which is unrelated to the other three corner detector properties.

5.2.4 Calculating robustness

In most evaluations only a subset of corners is used (i.e. top n), meaning that the task of corner detection is very similar to ranking. Therefore, performance metrics used for ranking could be used for corner detection, in addition to the popular repeatability metric. One of those performance metrics is the mean average precision (MAP). Section 5.2.4 will thus introduce a new performance metric to aggregate repeatability results based on the MAP used traditionally in ranking. To calculate the MAP the repeatability needs to be discussed first, which is done in the following section. One problem with the repeatability is that the definitions used for it differ slightly. Three examples of evaluations using repeatability with small differences are those performed by Mikolajczyk and Schmid [21], Schmid et al. [33], and DeTone et al [3]. The definition used by the latter of these evaluations is also used for evaluating the skeleton corner detector.

Repeatability

For two images I_1 and I_2 a corner detector returns a sorted list of corners A and B respectively, with a_i and b_i the ith elements of the list. The corners are sorted based on a response value given by the corner detection method. The function $C(a), a \in A$ takes a corner from the first list and gives the smallest index of a repeated corner in B. i.e. the smallest i for which the following equation is true:

$$||H_{I_1-I_2}a - b_i|| \leq \epsilon \tag{5.1}$$

Where $H_{I_1-I_2}$ is the homography transforming points from I_1 to image I_2. A homography $H_{I_1-I_2}$ is a matrix that defines the transformation from points in image I_1 to points in the image I_2. Furthermore, a point x is often not detected exactly at $H_{1-2}x$ in the second image. The parameter ϵ is introduced as the distance threshold. The repeatability is calculated for a certain number n of corners (i.e. the top n corners of A and B). The correctness (repeated) of a corner a_i is given by the following function:

$$corr(a_i, n) = C(a_i) \leq n \wedge i < n \tag{5.2}$$

The repeatability for a given n is defined by the function R:

$$R(n) = \frac{|\{a_i \in A | corr(a_i, n)\}|}{n} \tag{5.3}$$

For a repeatability $R(n)$ the number of corners in the top n are counted that are repeated in the top n corners of the second image. The repeatability can thus be described as the following fraction: $\frac{\#repeatedcorners}{n}$. Note that this is calculated only for the first image, to match the definition given by DeTone et al. [3] it should also be calculated for the second image, and finally averaged for the two. A second important subtlety is that corners whose transformed coordinates that are outside of the bounds of the image plane are ignored, because these cannot possibly be repeated.

MAP calculation

When calculating the MAP, a definition for the precision and recall is required. For a query size of n the precision@n is the repeatability $R(n)$. Recall is simply the number of repeated corners. Knowing the precision and recall for different query sizes means that the MAP can easily be calculated. The average precision is an estimation of the area under the precision-recall curve, its definition is given below:

$$AP = \frac{\sum_{k=1}^{m} R(k) * rel(k)}{n} \tag{5.4}$$

$rel(k)$ is 1 if the corner at the kth location is repeated and 0 otherwise (i.e. if it increased the recall). The AP is calculated for a number of image pairs and the mean of these is the resulting MAP. An important thing to note when calculating the MAP is that m is different for each image pair. Calculating the area over the precision-recall curve only makes sense if the x axis (recall) goes from 0 to 1. However, in this case the question is when the recall is equal to 1. Instead of choosing n as a number of corners (repeated or not), n is chosen as the number of repeated corners. Thus m is the number for which a method has a recall of 1 (detected n repeated corners). In other words the detector found n 'usable' corners that are found in both images. The benefit of using the MAP is that it provides a single metric that aggregates repeatability results for a large number of image pairs while not focussing on a single number of corners. Furthermore, it penalizes corners more who are not repeated earlier in the list of corners.

Repeatability in practice

The way in which the repeatability is used differs. Rosten and Drummond [31] compare methods using repeatability for different number of corners. The repeatability for different number of corners are plotted for several methods to show their behaviour when the number of corners per frame is increased. Rosten et al. [32] use NMS with a 3x3 mask for all detectors, then calculate repeatability and aggregate the results by computing the area under the repeatability curve for 0-2000 corners. A recent comparison of feature detection methods is done by Lenc and Vedaldi [14]. Although not evaluating solely corner detectors, the evaluation metrics are practically the same. Repeatability is used as the performance metric. Only feature detectors that assign a value to each feature are used, allowing for an ordering of the detected corners. The authors mention that comparing methods returning different number of features is not fair. The strategy they are using is to set the detection threshold of a method very low so that enough features are given. They then look at the top n features where $n \in \{100, 200, 500, 1000\}$. Plotting repeatability for different values of n does not scale well when large datasets and many different methods are used. Thus, the results for different scenes and different n are aggregated by averaging them. They also calculate the standard deviation to measure a detector's stability. This last addition is especially useful, as it shows in their results that some methods perform very badly in some scenarios. These bad results in their case can be attributed to a translation/rotation covariant feature detector being used in a scene with large scale changes.

DeTone et al. [3] present a new corner detector. Here the mean average precision is used, but only for a small synthetic dataset. For real images they only use repeatability on a fixed number of 300 corners in each image. They also use two different values for NMS, 4 and 8. Huo et al. [13] use repeatability but does not give a precise definition and fixes the number of features detected to 200 and 1000.

In conclusion, repeatability is used in all the evaluations, but the way in which it is used differs. Several important observations can be made from these examples:

1. Use NMS to make sure the results are meaningful, i.e. a feature detector that detects 4 points all very close together might have high repeatability, but the points could be redundant and not very useful for matching tasks.
2. Do not use a single number of features for calculating repeatability as its result can be misleading.
3. Control the number of features for each method. By not comparing the methods on a fixed number of corners, a bias is introduced in choosing the parameters of the model. This is because the number of corners heavily depend on the choice of threshold parameter. As has been said before, comparing methods when they produce different numbers of features is not a fair comparison.
4. Aggregate results into a single metric for ease of comparison. This is especially important when larger datasets are being used. Both the results for different datasets and different number of corners could be aggregated.

5.3 Application-specific

Application-specific performance evaluation uses corner detection as only a part of a larger pipeline, and therefore uses a performance metric that is related to that application. In the performance evaluation summary only two application-specific evaluations were found: matching and vehicle tracking. The former of these is used 5 times and the latter only once. Even though matching is an application-specific evaluation method, it is used quite frequently. It seemingly tries to assess the completeness of the feature detector. The repeatability says nothing about the completeness, or usefulness of the detected features, and thus using them for a matching task gives an indication of the completeness of the features. This would likely punish methods returning very robust features that give little information of the image. Thus while not directly assessing the completeness, the results of a matching task, such as the matching score, gives insight to the completeness of a feature detector.

5.3.1 Matching

The matching score is introduced by Mikolajczyk et al. [22] as a way to measure the distinctiveness of detected features. In the original paper region detectors were compared, but the metric has also been used for corner detection [15]. The matching is calculated similar to repeatability, using an image pair with a ground truth homography. First, for every corner found in both images a feature vector is calculated using a descriptor. The choice of descriptor does not matter, however most often this is the Sift descriptor. The region over which the descriptor is calculated is a fixed 30x30 circular region [22] and the orientation should be determined by the descriptor. For the matching experiments in this section the opencv implementation of the daisy descriptor [37] is used because it also computes the orientation.

The matching score is the fraction of correct matches. The match of a corner in the first image is a corner in the second image with closest distance in the feature space. This match is correct if it is also the closest neighbour in the image plane (the second corner is transformed using the homography). The original proposal for the matching score is calculated for region detectors and thus use the overlap error instead of the euclidean distance.
The matching score is then calculated as follows:

$$R(n) = \frac{|\{a_i \in A | corrMatch(a_i, n)\}|}{n} \tag{5.5}$$

A correct match is determined as followed

$$corrMatch(a_i, n) = ||H_{I_1 - I_2} a - match(a_i, n)|| \leq \epsilon \tag{5.6}$$

Where $match(a_i, n)$ gives the location of the closest corner $b \in B$ in the feature space. Note that a correct match require the match to be its closest corner in the image plane, just that the distance is below ϵ, which is also the case for the repeatability.

5.4 Datasets

As discussed earlier there are three prominent methods of evaluating different detectors: visually analysing the detected corners, using the detection method in some specific system (such as matching), and finally by computing the repeatability. This last way of evaluation requires a dataset with image sequences and homographies between the images. The homographies can be obtained in two ways. The first way is by using synthetic images in which the exact transformation is known for the image sequence, and the second way is to estimate it by using corresponding points in the two images. Using non-synthetic images seems to be more widely used in evaluations [22, 14]. Synthetic images can also be used, as is done recently by DeTone et al. [3], but is not a great indicator of performance on real images, as can be seen from their results. Furthermore, in this specific case it is only a minor part of the evaluation. Besides this example most recent evaluations use real datasets such as will be discussed next.

Table 5.2 shows a summary of four datasets used for feature detection. Of these, Oxford [22] is the most used and is especially useful because the scenes are labelled with the type of deformation present. HPatches [2] is a newly proposed dataset and is considerably larger. The EF [42] dataset contains only a few scenes with large deformations. Lastly the WebCam [40] dataset only contains light changes.

HPatches was developed to provide a benchmark for the evaluation of local descriptors. Its main goal is thus not interest point detection or region detection. However, the images provided can still be used for that purpose [3] [14]. 57 scenes show photometric changes (illumination changes) and 59 scenes show geometric deformations resulting from a change in viewpoint. The dataset does not distinguish different

name	year	nr of scenes	nr of images	illumination	viewpoint
VGG [20]	2005	8	6 per scene	X	X
EF [42]	2011	5	38	X	X
WebCam [40]	2015	6	140	X	
HPatches [2]	2017	116	6 per scene	X	X

TABLE 5.2: Summary of several of the most prominent datasets used in performance evaluation. The columns illumination and viewpoint refer to the existence of these type of transformation within the scenes of the dataset.

geometric deformations, such as scale or rotation changes. Because of this, performance evaluation of different methods cannot be focused on a single type of transformation, which could highlight strengths and weaknesses. It is still a great way to evaluate the overall performance. Each scene consists of five images. Each image sequence comes with ground truth homographies that are the transformations from the reference image (the first image in the sequence) to the subsequent images of the scene. This dataset is publicly available [2].

The second, and older, dataset is the Oxford VGG dataset [22] that is used for evaluating affine covariant region detectors. It has a similar structure to the previous one. There are again sequences of images (scenes) containing different image transformations with corresponding ground truth homographies. In total there are only 8 scenes but they are classified with having a certain type of deformation: blur, viewpoint, zoom+rotation, light, and jpeg compression. Having these different classifications allows to clearly identify strengths and weaknesses. The dataset however is much smaller with only 8 scenes compared to the 110 scenes in HPatches. The VGG dataset is publicly available [3]. Lastly, the EF and WebCam datasets are used less often but are both publicly available.

5.5 Summary

This performance evaluation is based on the four properties important to a corner detector: robustness, completeness, sparseness, and speed. The latter of these is not included, and sparseness is controlled for rather then measured directly. For robustness only one performance metric was found, the repeatability, which was frequently used in previous evaluations. There is no consensus on how to measure completeness. The research shows several methods that aim to asses the usefulness and quality of a set of detected corners, however none of them are sufficient as a standalone metric. Therefore, multiple metrics related to completeness will be used separately.

Based on the findings of this chapter the evaluation strategy will be described. Below are the performance metrics that will be used:

1. Repeatability curves for a number of corners in the range (0,1000). These will be given for the Oxford dataset scenes, because they show the behaviour for specific types image deformations. The repeatability curve can also be shown

[2] `https://github.com/hpatches/hpatches-dataset`
[3] `http://www.robots.ox.ac.uk/~vgg/research/affine/`

as an aggregated result of the scenes of the Oxford, EF, and WebCam datasets seperately (i.e. a single graph for each dataset).

2. Matching curves using the same strategy as for the repeatability.
3. Visual analysis of detected corners for scenes in the Oxford datasets. A limited number of corners should be used, because larger amounts will cause the screen to be cluttered.
4. MAP for the Oxford, EF, and WebCam datasets.
5. Coverage values for the scenes of the VGG dataset.

The above described evaluation methodology will be applied and its results are given in the following chapter.

Chapter 6

Results

In this chapter the evaluation methodology introduced in the previous chapter is used. The proposed skeleton corner detector is compared with the two most famous, and still relevant [14], corner detectors: Harris [11] and Fast [30]. These detectors will be referred to as Skel, Harris, and Fast respectively. The evaluation is done using three of the datasets discussed in section 5.4: VGG [20], WebCam [40], and EF [42]. The scenes of the VGG dataset are shown in figure 6.1, because the performance on the individual scenes of the VGG dataset will be analysed.

For the EF dataset the 'panorama' scene is excluded. Furthermore, this dataset is considerably harder, because it contains large deformations, of which mostly viewpoint changes. These large viewpoint changes can cause the first image to be only a small part of a second image. Therefore, the number of corners found in the second image, that are also within the bounds of the first image (after the inverse homography transformation), is often small. The upper limit of 1000 corners used for the repeatability calculation is thus often too high, and instead an upper limit of 600 was chosen.

The WebCam dataset contains scenes with no viewpoint changes, only illumination changes. However, the scenes do contain changes in background.

6.1 Visual analysis

The visual comparison between detected corners of the different methods is shown in figure 6.2. For each scene the top 150 corners are shown for all methods. What can be observed from these images, is that overall the methods produce corners that visually make sense. In addition, it seems that the skeleton method produces corners that are more spread out over the image. One interesting scene is 'trees' where the methods produce a vastly different set of corners. This is also the scene where the skeleton obtains a very bad performance for the other performance metrics. A reason for this could be that the 'trees' scene has a lot of textures and the shapes of the tree, such as the branches and leafs, are mostly obstructed and small. Therefore, the threshold sets do not contain clear shapes and thus no proper skeletons are produced.

6.2 Repeatability

Repeatability is calculated using equation 6.1 which was introduced in section 5.2.4. The ϵ used for calculating repeatability is set to 3 and non maximum suppression to 8 (meaning that in a 8x8 region there can exist at most a single corner).

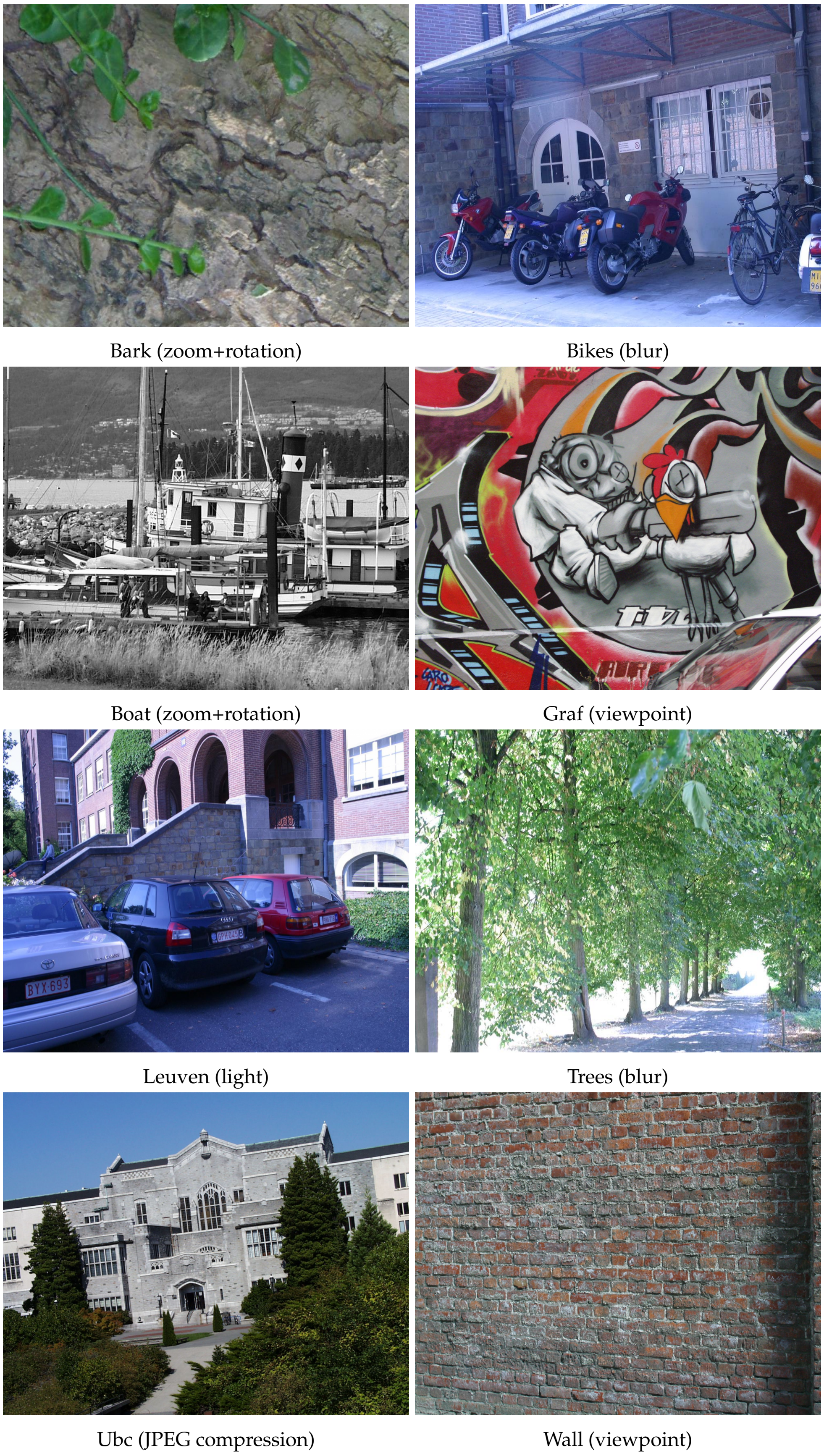


FIGURE 6.1: The 8 scenes of the VGG dataset and the type of deformation present in each scene given in brackets.

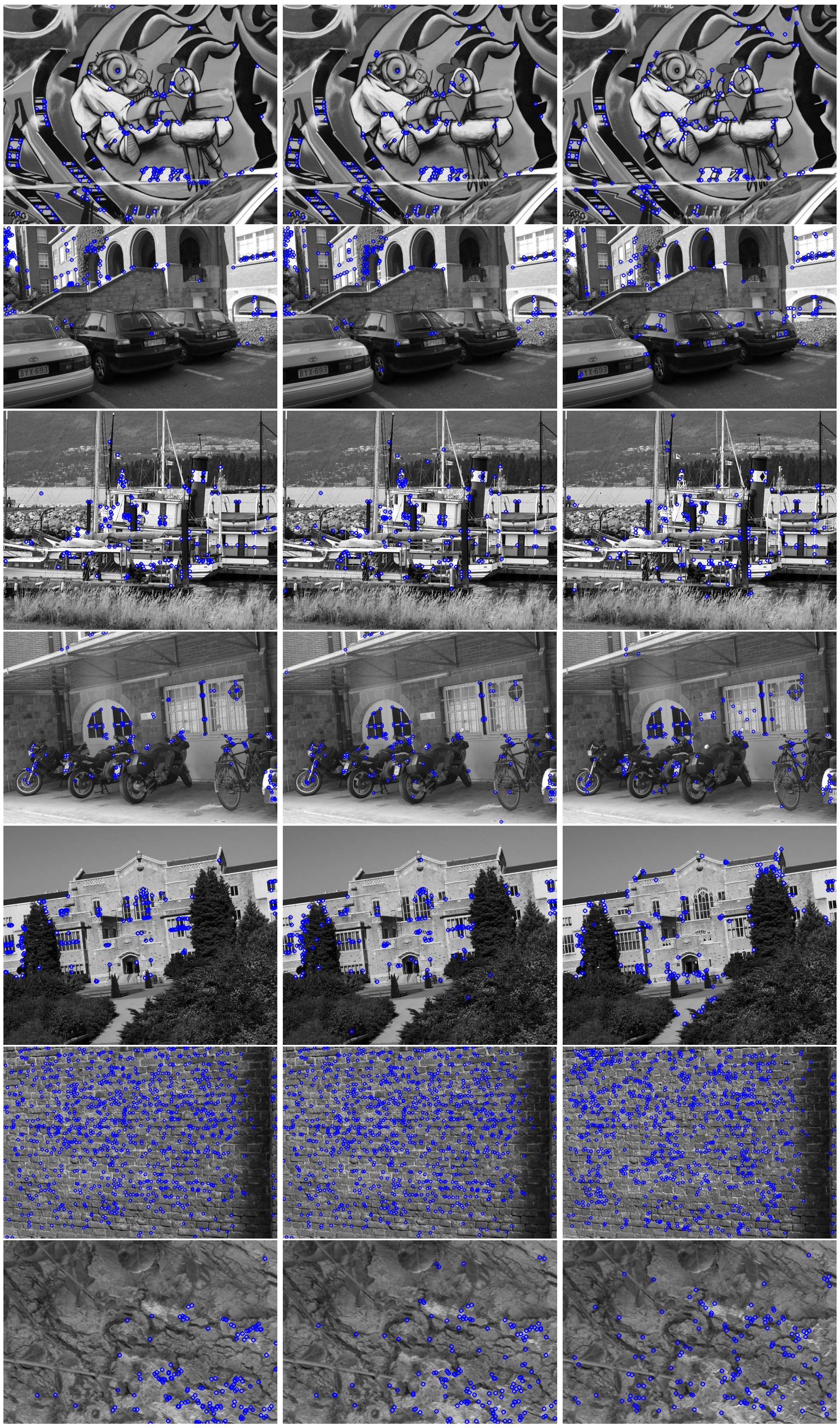


FIGURE 6.2: Top 150 corners for 4 of the scenes of the VGG dataset for the Harris (left), Fast (middle), and skeleton (right) corner detector.

$$R(n) = \frac{|\{a_i \in A | corr(a_i, n)\}|}{n} \tag{6.1}$$

The repeatability is plotted as a function of the number of corners used for its calculation. Figure 6.3 shows the repeatability for the scenes of the VGG dataset. The aggregated results of all the scenes of the datasets are shown in figures 6.5, 6.6, and 6.7 for the VGG, EF, and WebCam respectively. For the VGG dataset each scene has a separate graph which should give some insight in the performance for a method for a certain type of scene (with its own respective deformation. For the EF and WebCam dataset only the aggregated result is shown to avoid too much clutter.

The results for the VGG dataset show that on average the skeleton method is comparably in performance to Fast and Sift for a limited number of corners, although outperformed by Harris. There is no clear type of deformation where the skeleton method performs bad on for the VGG dataset. The three scenes where the skeleton method obtained the worst performance are: 'trees', 'bark', and 'wall', who have contain deformations of the type blur, zoom+rotation, and viewpoint respectively. The common factor between these scenes is that they all contain textures and do not have clear objects in the image. Especially the textures could be the cause for a bad performance obtained by using skeletons. The reason for this is illustrated in figure 6.4. The threshold sets are messy because of the textures and do not contain any distinguishable objects, resulting the computed skeleton to be messy as well with its endpoints not corresponding to the corners of actual objects. When looking at the computed skeletons in figure 6.4 it is clear that a bad performance is obtained for a scene like that.

More promising results can be observed on the EF dataset however. This dataset is considerably harder which is evident by the lower overall repatability obtained by all the methods. Here the skeleton method outperforms the fast corner detector and is on par with Harris for a limited number of corners. While the EF dataset is harder, the relative performance of the skeleton detector is better, indicating that the skeleton method is suited for scenes with complex deformations. This result makes sense, because skeletons intuitively should work well for large transformations, considering the affine covariant property of a skeleton. A slightly worse performance is obtained again for the WebCam dataset which only contains illumination changes. This bad performance can be partly explained by an outlier scene in which the skeleton method contained a very low performance, namely the 'Courbevoie' scene. Allthough for the other scenes in the WebCam dataset the skeleton corner detection method's performance is again on par with the Fast and Harris corner detectors for smaller number of corners.

One of the observations made is the fall of in performance for the skeleton corner detection method when the number of corners increases. An explanation for this could be that the method finds a limited number of salient corners and the decrease of performance is caused by low response corners that only appeared in a few threshold layers. Figure 6.8 shows the relative response values for the corners for each methods that were used in the repeatability evaluation. For example if the y value for the 500th corner is 0.2, then the response for this corner is 20% of the highest response corner found. These figures do not show a clear difference in the shape of curve between the corner detector methods. The initial decrease in relative response value is often lower for the skeleton method, but this can mean that

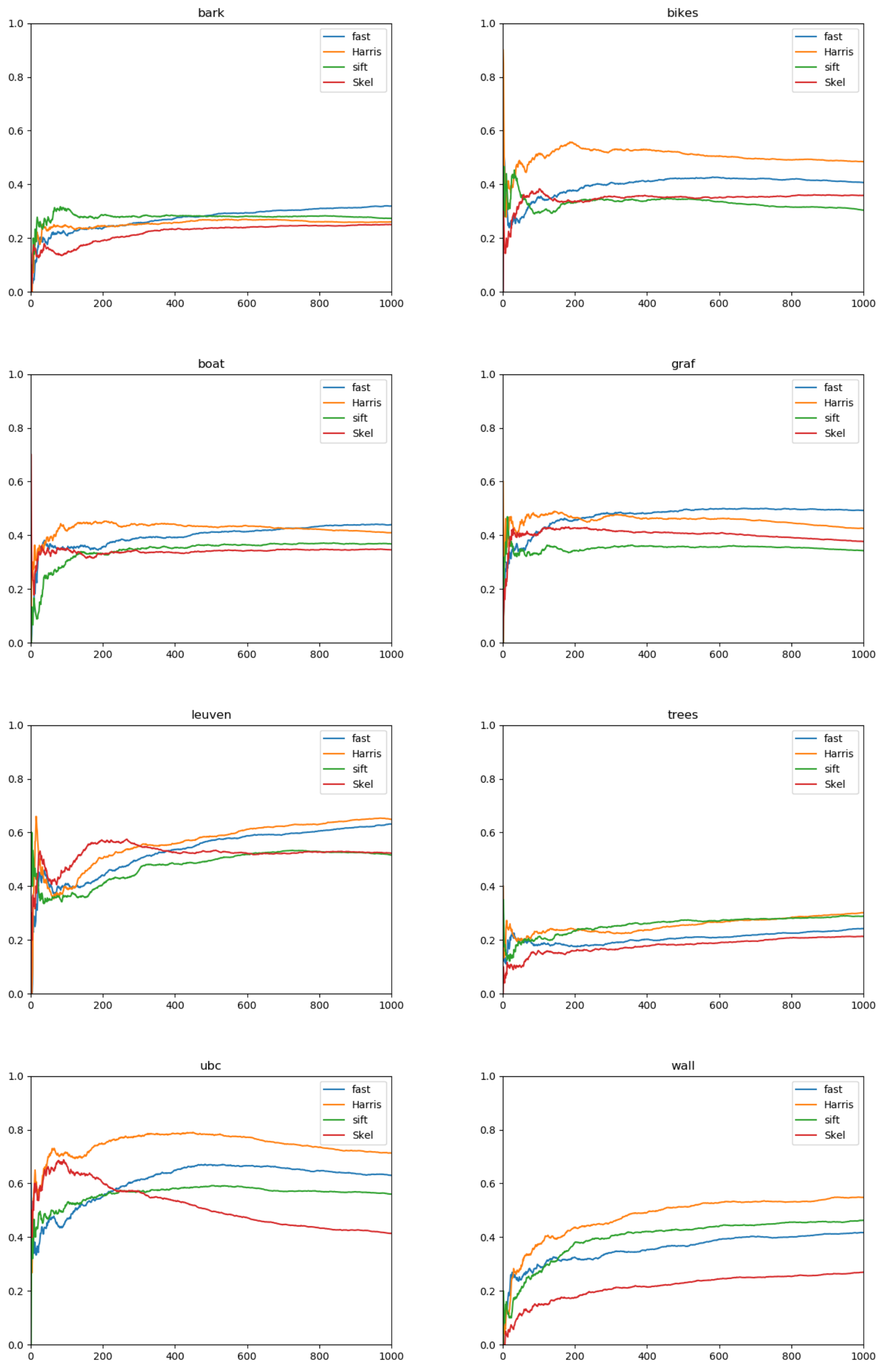


FIGURE 6.3: Repeatability for each scene of the VGG dataset. Values are averaged over the repeatability of all the image pairs of each scene.

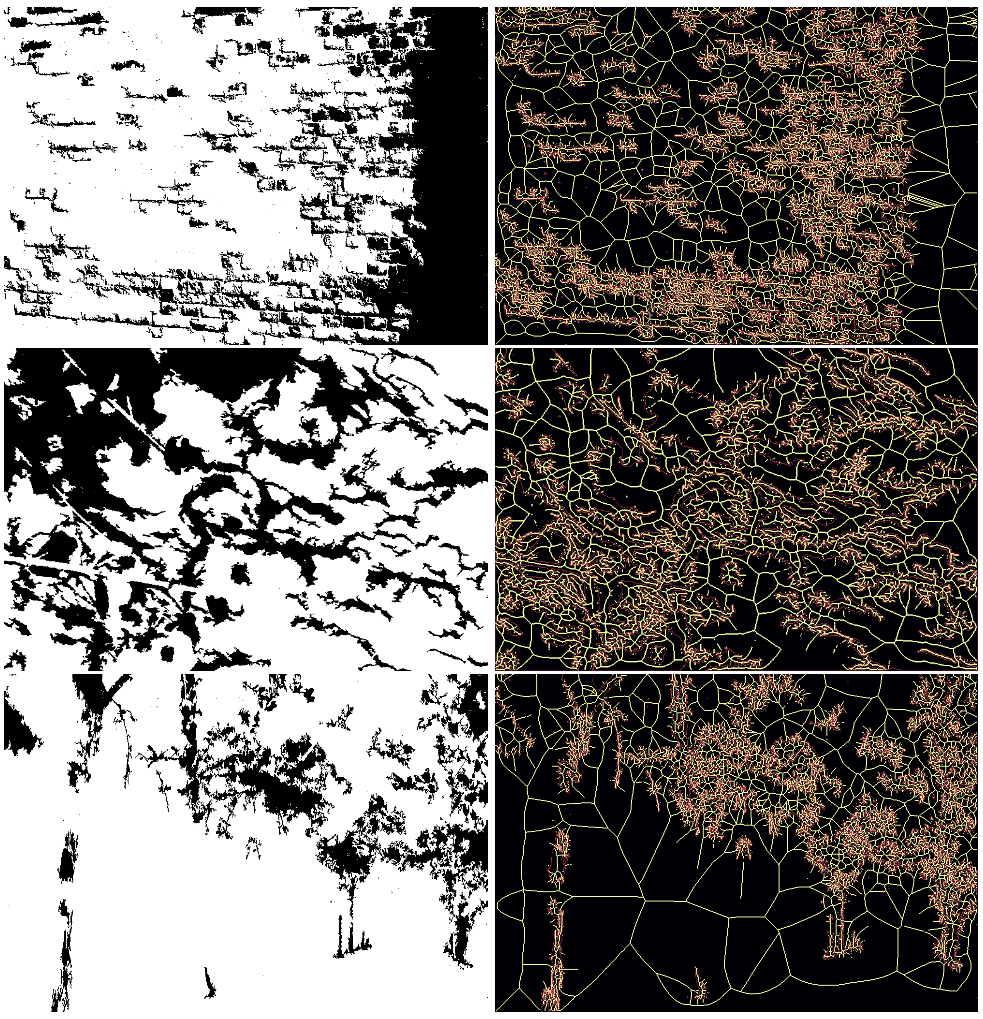


FIGURE 6.4: Examples of a threshold layer and computed skeletons for this layer for several scenes of the VGG dataset: 'walls' (top), 'bark' (middle), 'trees' (bottom). Left column shows the the 100th threshold layer, right column the skeleton for this threshold layer. For the skeleton image the yellow and red pixels are the skeleton and boundary pixels respectively.

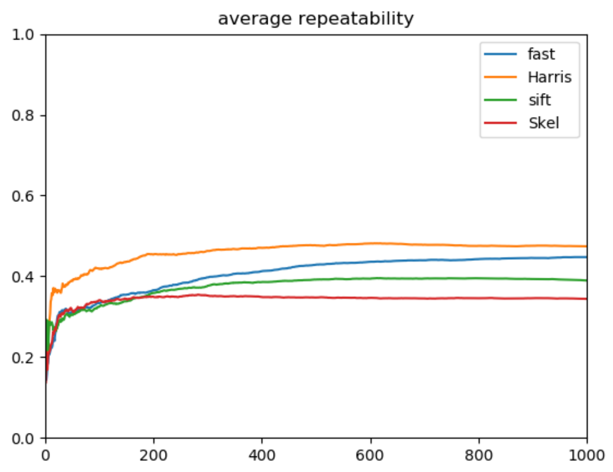


FIGURE 6.5: Repeatability averaged over all the scenes of the VGG dataset. The overall bad performance of the skeleton corner detector is partly explained by several scene problematic scenes.

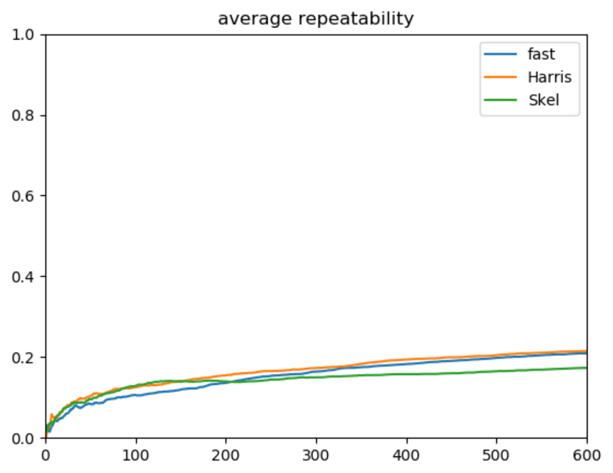


FIGURE 6.6: Repeatability averaged over all the scenes of the EF dataset. The relatively good performance for the skeleton corner detector is promising, because the EF dataset contains the most severe deformations.

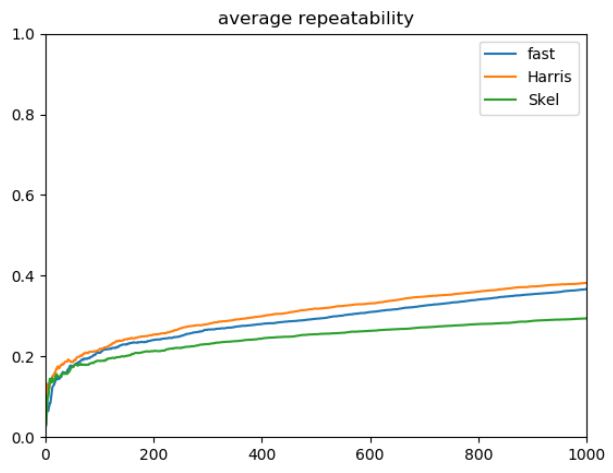


FIGURE 6.7: Repeatability averaged over all the scenes of the WebCam dataset. A worse performance for the skeleton corner detector is expected here, because the WebCam dataset contains only illumination changes.

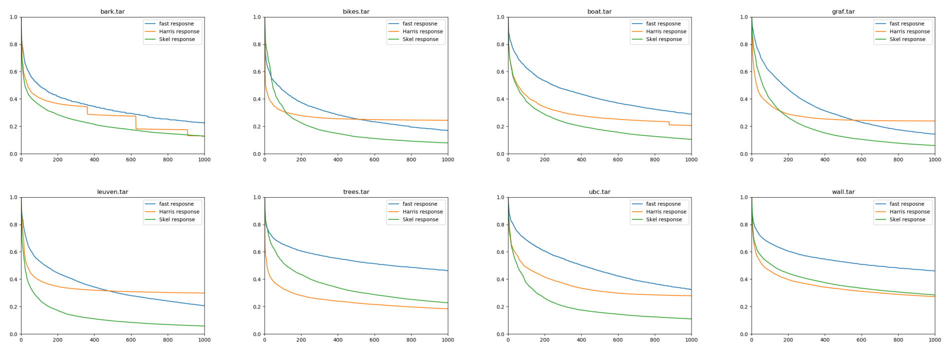


FIGURE 6.8: Relative responses for the corners obtained for the Fast, Harris, and skeleton corner detection methods. For each corner number on the x axis its relative response is given with respect to the maximum response found for that specific method.

its maximum response value is very high. Even for the scenes where the skeleton method obtained the worst , 'Tree', 'Wall', and 'Bark', performance there is no clear difference. This would indicate that the problem is not the lack of high response corners being found. Thus there were many corners found with a high response (found in many threshold layers) but they were not repeatable and probably caused by the noise in the threshold layers.

6.3 Matching

The matching score is calculated using equation (6.2) which was described in section 5.3.1.

$$corrMatch(a_i, n) = ||H_{I_1-I_2}a - match(a_i, n)|| \leq \epsilon \tag{6.2}$$

The results are visualised in the way same as was done for repeatability, plotting the matching score as a function of number of corners. The graphs for the matching score for each scene of the VGG datasets are shown in figure 6.9. The aggregated results are shown in figures 6.10, 6.11, and 6.12 for the VGG, EF, and WebCam dataset respectively.

The matching score has a high relation with the repeatability, i.e. for a corner to be matched correctly it also needs to be repeated. The interesting observations that can be made from the matching results are the relative changes compared to the repeatability. For example, if a method's matching score decreases compared to its repeatability more than other methods, it indicates that a lot of the corners are not distinctive.

From looking at the graphs with the matching scores and comparing them with the repeatability graphs several observations can be made. First of all, the 'Bark' scene has an overall large drop in matching score, because the scene contains zoom changes and does not have many distinctive objects. However, the matching score of the skeleton method is relatively good, especially the limited drop-off compared to the repeatability for this scene. The same is true for the 'Boat' scene. This bad overall performance is due to the descriptors calculated not being scale invariant. For the 'Graf' and 'Leuven' scenes the matching score of the skeleton method has less of a

Detector	VGG[20]	EF[42]	WebCam[40]
Fast	0.185	0.075	0.142
Harris	0.230	0.074	0.143
Skel	0.180	0.073	0.126

TABLE 6.1: MAP values for the Harris, Fast, and skeleton corner detectors on the VGG, EF, and WebCam datasets.

drop off compared to repeatability than the other methods, especially when comparing with Harris. For the remaining scenes nothing interesting happens. Overall the corners returned by the skeleton detection method seem to be more distinctive than the other methods, because there is less drop-off in matching score compared to the repeatability.

6.4 MAP

So far the performance metrics have been calculated for a range of number of corners. It can be convenient to aggregate these results and obtain a single value. One straightforward way to do this is by averaging over all the different number of corners, which would be calculating the area under the repeatability curve. Another suggested metric is the MAP which was proposed in section 5.2.4. Instead of considering a fixed range of number of corners, a required number of repeated corners is selected. This number can be changed depending on the requirements of an application. For this evaluation a required number of repeated corners is set to 100 for the VGG and WebCam datasets, meaning that for this number the recall is equal to 1. The EF datasets contains larger deformations and therefore a lower number of 50 is chosen. Table 6.1 shows the obtained MAP values.

6.5 Coverage

The last performance metric that will be used is the coverage which was proposed by Ehsan et al. [8]. This method which aims to measure the completeness was briefly discussed in section 5.2.2. The authors mention that the results for the coverage metric should be consistent with visual inspection, meaning that if a detector obtains a relatively high coverage the quality of the corners should also be better. The authors verify this by using the corners for homography estimation, and arguing that high coverage values correlate with a better homography estimation, and thus that their proposed performance metric is useful.

The coverage value uses the harmonic of the distances between the detected corners. First, for each corner i the harmonic mean D_i of all the distances is calculated following equation 6.3.

$$D_i = \frac{N-1}{\sum_{j=1,j\neq i}^{N}(\frac{1}{d_{ij}})} \tag{6.3}$$

Where N is the number of corners and d_{ij} the euclidean distance between corners i and j. Next, the harmonic mean of all the harmonic means D_i is calculated using equation 6.4, yielding the coverage C.

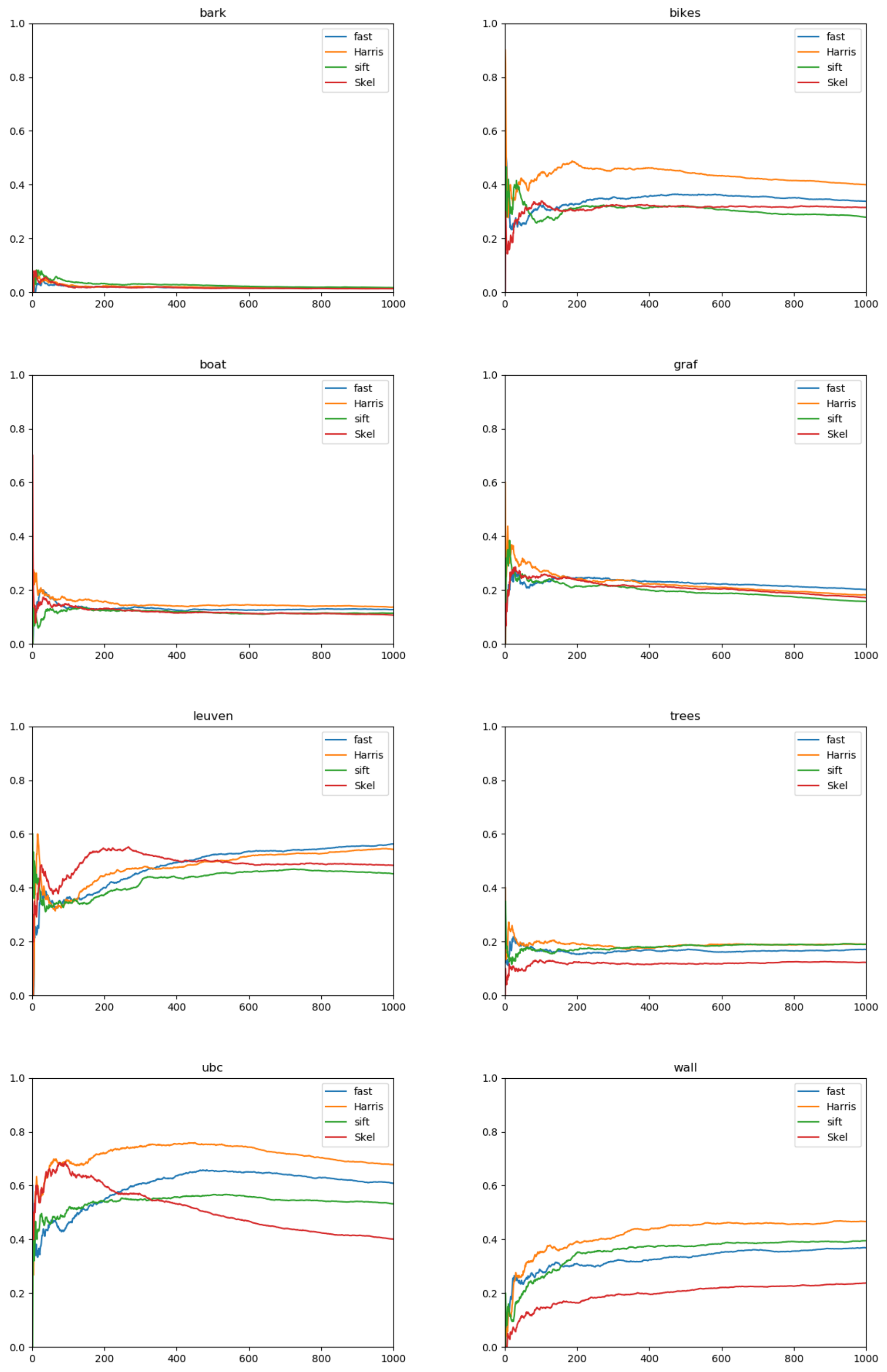


FIGURE 6.9: Matching score for each scene of the VGG dataset. Values are averaged over the repeatability of all the image pairs of each scene.

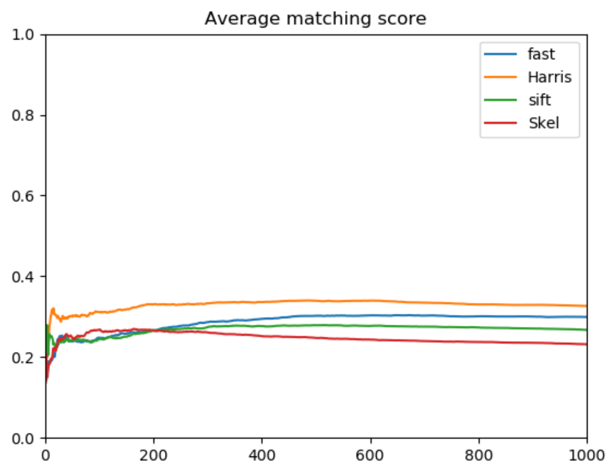


FIGURE 6.10: Matching score averaged over all the scenes of the VGG dataset.

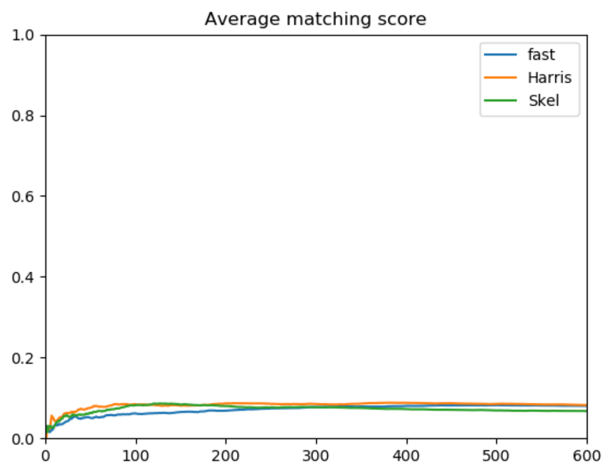


FIGURE 6.11: Matching score averaged over all the scenes of the EF dataset.

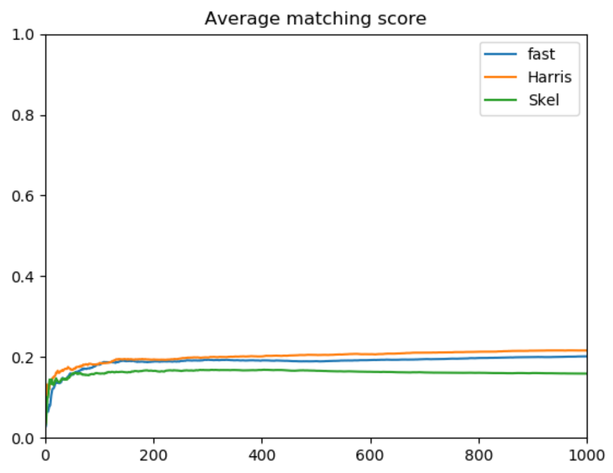


FIGURE 6.12: Matching score averaged over all the scenes of the WebCam dataset.

	200 corners			1000 corners		
Scene	**Harris**	**Fast**	**Skel**	**Harris**	**Fast**	**Skel**
bark	106.288	124.271	**133.648**	135.993	143.111	**150.65**
bikes	144.055	147.983	**156.635**	173.204	**189.665**	183.295
boat	99.1184	103.193	**109.536**	129.423	127.422	**132.9**
graf	126.82	134.783	**153.533**	**179.249**	176.39	171.626
leuven	109.07	116.698	**123.851**	139.175	146.828	**162.024**
trees	160.108	**170.265**	151.303	185.399	**189.625**	183.516
ubc	100.769	114.725	**132.767**	156.291	154.872	**166.479**
wall	170.501	**187.121**	174.55	182.454	193.786	**194.234**

TABLE 6.2: Coverage values computed for the Harris, Fast, and skeleton corner detector on the 8 scenes of the VGG dataset. Note that values should be compared within each scene only.

$$C = \frac{N}{\sum_{i=1}^{N}(\frac{1}{D_i})} \tag{6.4}$$

The coverage value depends on the image resolution and thus the values between different scenes cannot be directly compared. Therefore, the results will be shown for the scenes of the VGG dataset separately. Another choice is the number of corners that are used to calculate the coverage. In order to complement the previous performance evaluation, the number of corners used for calculating the coverage will be set to 200 and 1000. Table 6.2 show the coverage values computed for the scenes of the VGG dataset. Note that higher coverage values are better. These results will be discussed in section 7.1.2.

6.6 Saliency threshold

One of the main adjustable thresholds used for the skeleton corner detection is the saliency. The saliency threshold controls the type of corners that are being found the threshold sets. This is expected to have a large impact on the resulting corners and the performance. The results for different values for the saliency threshold are shown in figure 6.13. The average results are quite similar for the different values, with a saliency of 1.4 and 1.6 being slightly on top. However, larger differences can be observed for the individual scenes. This means that the performance of a saliency threshold depends on the scene, but on average a saliency of 1.4 or 1.6 seems ideal. For example, for the 'Graf' scene (figure 6.14 a higher saliency obtains the highest repeatability, while the opposite is true for the 'Ubc' scene (figure 6.14).

Visualizing the detected corners for the saliency values helps to analyse which threshold is the best. Figure 6.15 shows the top 500 corners for the two scenes with the most repeatability differences, 'Graf' and 'Ubc'. The two saliency values compared are 1.0 and 1.6, where the former is representing a low saliency threshold and the latter a medium to high threshold. The saliency threshold 1.6 resulted in the average highest repeatability, however in some individual scenes a lower saliency showed significantly higher performance. This raises the question whether for these a low saliency would be better. Looking at figure 6.15 this is obviously not the case. For both scenes the most noticeably difference is that a low saliency results in many corners being detected on edges, and otherwise low salient points. Especially in the

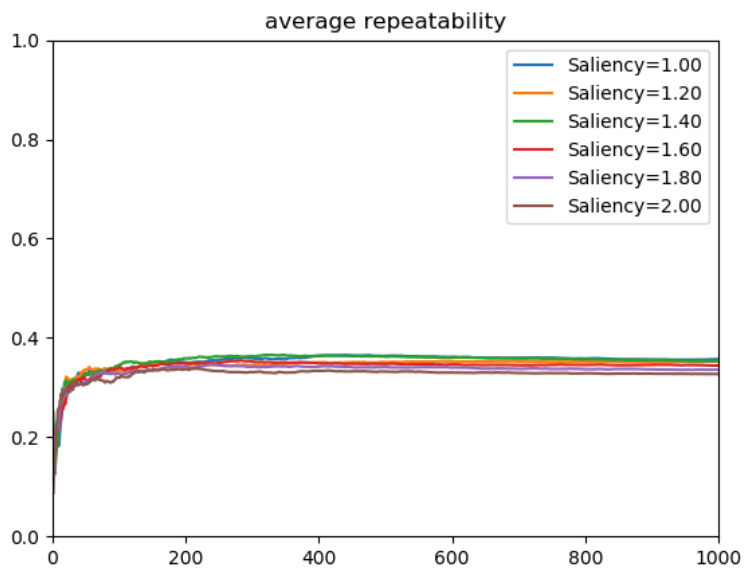


FIGURE 6.13: Repeatability score averaged over all the scenes of the VGG dataset for different saliency thresholds. Overall the different saliency thresholds yield similar results, however more significant differences can be observed for the individual scenes (see fig 6.14).

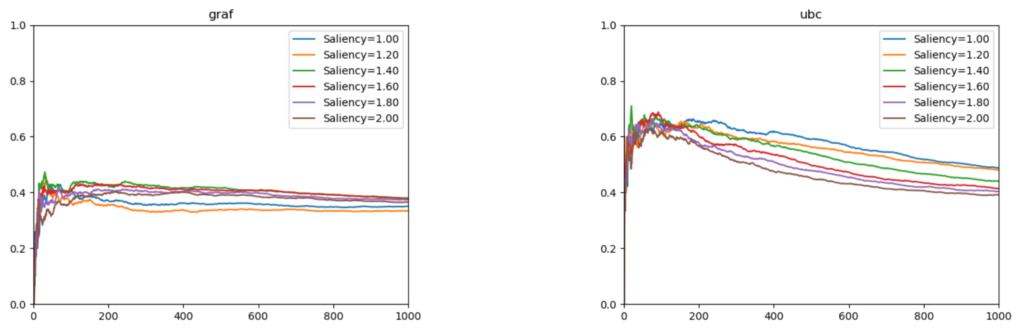


FIGURE 6.14: Repeatability for different saliency thresholds on the 'Graf' (left) and 'Ubc' (right) scene of the VGG dataset. The performance, in terms of repeatability, of different saliency thresholds depend on the scene. A low saliency value is optimal for the "Ubc' scene, while the opposite is true for the 'Graf' scene.

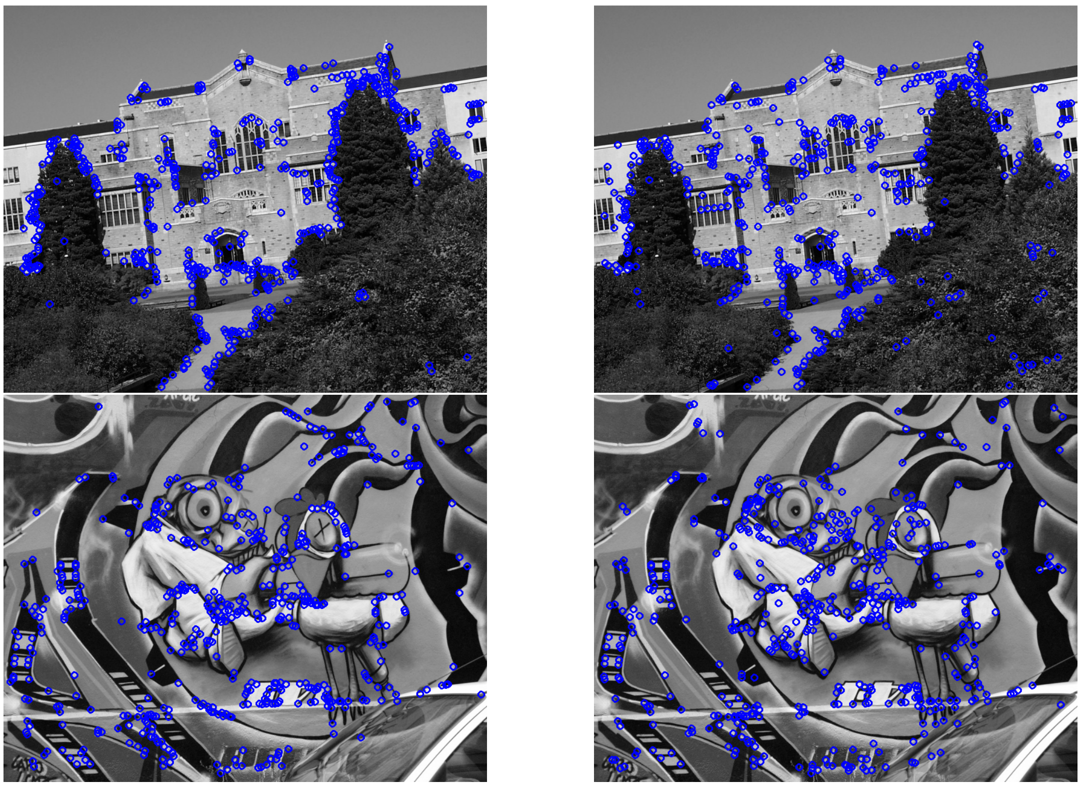


FIGURE 6.15: The top 500 corners for the 'Ubc' and 'Graf' scene of the VGG dataset. The left column are obtained using a saliency of 1.0 and the right column a saliency of 1.6. When a low saliency value is used, many non-salient corners are being detected, most noticeable are those located along the edges.

'Ubc' scene a low saliency gives many corners on the edges of the trees. This example also shows the shortcoming of the repeatability metric. Using the repeatability as the sole deciding metric would favour a low saliency for the 'Ubc' scene, but taking a closer look at the type of corners detected would suggest that a saliency of 1.6 gives more salient points.

6.7 Stepsizes

So far all threshold sets were used in finding the corners. However, an increase of the speed of the detector can be obtained by limiting the number of threshold sets. This is done by selecting every *Nth* threshold set starting from 1, in the default case the stepsize was set to 1. Increasing the step size from 1 to 2 should almost to double the speed. Larger stepsizes increase the speed gain. The only downside is the possible loss of performance. The trade-off between the speed increase and the performance decrease is plotted by using the repeatability on the VGG dataset in figure 6.16. The results show that using stepsizes in the range of 1 to 5 yields similar results, although there is a drop off in performance. Therefore, significant speed increases can be obtained with a limited performance decrease.

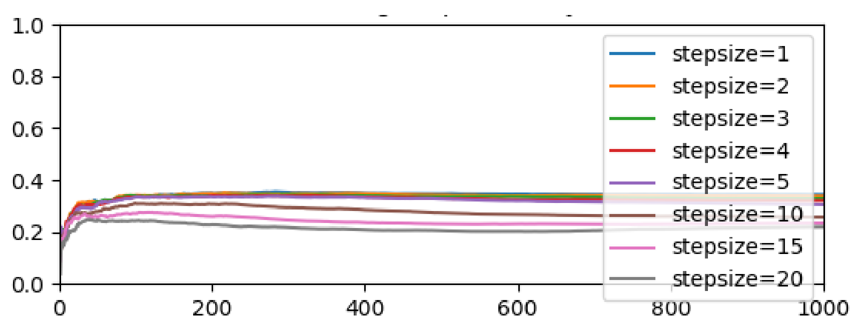


FIGURE 6.16: Repeatability averaged over all the scenes of the VGG dataset for different stepsizes. A small increase of the stepsize does not lead to a significant reduction in performance. However, it does lead to a significant speed-up in computation time.

	MAP		coverage	
Scene	**Normal**	**Adjusted**	**Normal**	**Adjusted**
bark	0.076	0.066	133.648	135.107
bikes	0.210	0.167	156.635	155.365
boat	0.164	0.127	109.536	111.582
graf	0.214	0.173	153.533	155.592
leuven	0.250	0.225	123.851	130.219
trees	0.092	0.083	151.303	166.827
ubc	0.329	0.290	132.767	138.993
wall	0.100	0.091	174.55	179.504

TABLE 6.3: MAP and coverage for the skeleton corner detector comparing two ways of using skeleton endpoints (normal and adjusted, see section 4.4.1) for the VGG dataset. MAP is calculated as was done in section 6.4, and the coverage is computed over 200 corners.

6.8 Adjusted endpoints

In section 4.4.1 two methods for using skeleton endpoints are mentioned. The first of these is to use the endpoints as corner locations, which is done in all the previous evaluations. The second proposed method is to adjust the endpoints so that they lie on the object boundary. This adjustment introduces several estimations and thus possible increases in localization error. Therefore, it will likely perform worse for a metric such as repeatability, where accurate localization is very important. However, this adjustment of the corners to the object boundaries could have other benefits. For example, Ravindran and Mittal [28] find corners located specifically on boundaries of extremal regions and find a good performance for application specific metrics related to vehicle tracking. The results comparing the two methods of finding endpoints using the MAP and coverage are shown in table 6.3. Although it was expected that adjusting the endpoints would lead to a decrease in the MAP, the decrease is quite severe for several of the scenes of the VGG dataset. The coverage value did not decrease, or rather it increased for most of the scenes.

6.9 Summary

In this chapter the performance of the proposed corner detection method was evaluated and compared against two prominent corner detectors using the evaluation methodology proposed in chapter 5. This consisted of an informal visual analysis of detected corners, and the two performance metrics repeatability and matching used to asses the robustness and completeness respectively. In addition, the behaviour of the method itself was analysed, by comparing different saliency thresholds and different numbers of threshold layers used. The results will be further discussed in the following chapter.

Chapter 7

Discussion and Conclusion

The previous chapter showed the results from the evaluation of the skeleton corner detector compared to two prominent detectors. This chapter will further discuss these results and conclude the effectiveness of the proposed method in section 7.1. Next some possible future research ideas based on all the findings are given section 7.2.

7.1 Performance analysis

The goal of the thesis was to assess how well the DMD can be used to find corners in grey-scale images. Chapter 5 outlined the four important properties that a corner detector should have. These were robustness, completeness, speed, and sparseness. The latter of these is not relevant for the evaluations performed, because it was always controlled. Lastly, another important aspect of a detector is its ease of use, which is mostly determined by the required parameters that need to be set.

7.1.1 Robustness

The results for the robustness (repeatability) are promising, and are competitive with other well established corner detection methods. Although overall lower than Harris for the VGG and WebCam dataset, it was equal in performance for the EF dataset. This is an indication that the skeleton corner detection method is less suitable for only illumination changes (WebCam dataset) and more so for very large viewpoint changes (EF dataset).

Two main problems were found with the skeleton corner detector regarding the robustness. The first was a poor performance for scenes containing textures. Textures cause the threshold layers to be messy and as a consequence the computed skeletons will not contain many salient endpoints. The second problem was the drop-off in performance when the number of corners was increased. It was observed however that this is not caused by a large drop-off in response values for these corners. Therefore the problem more likely lies in the type of corners being detected, potentially caused by the textures.

7.1.2 Completeness

Completeness was measured mainly with the visual analysis in chapter 5. The images showed no major differences between the evaluated methods. A second way to somewhat measure completeness was the matching score which was often used in the literature to measure distinctiveness of detected corners. The results for the matching score were also similar, although the skeleton corner detection method seemed to experience less of a drop-off in matching score compared to repeatability. Overall there were thus little differences found in terms of completeness of the detectors.

Another way to measure completeness was done using the coverage in section 6.5. The results were favourable for the skeleton corner detector, showing the highest coverage values for most scenes. Two observations consistent with previous results are, first of all, that the coverage values for the skeleton corner detector are relatively better for a smaller number of corners. Secondly, for 200 corners the two scenes 'Trees' and 'Wall' where the skeleton corner detector did not obtain the highest coverage are also the scenes where the worst performance for the repeatability and matching score was obtained. These results suggest that overall the skeleton corner detector may detects a more complete set of corners.

7.1.3 Speed

Speed was not a part of the evaluation. The reason for this was the novelty of the proposed method and thus the lack of an optimized implementation. Furthermore, the skeleton corner detector method requires the conversion from a normal grey-scale image to the DMD image representation, which takes up a bulk of the computation time. For reference, computing the skeleton for a single layer for the boat scene of the VGG dataset (850x680 image resolution) takes approximately 80ms, using a geforce GTX 960 graphics card. Note that this executes the AFMM algorithm described in section 3.2 twice, because of the reason described in section 3.2.1. Although there was no evaluation based on the speed, ways to accelerate the detection by selecting a subset of the threshold layers was discussed, and more importantly the decrease in performance that results from this speed-up.

7.1.4 Ease of use

A nice property of the proposed skeleton corner detector is that it does not require much parameter tuning. One of the main parameters, the saliency threshold, was found to produce different results for the scenes of the VGG dataset. However further analysis showed that a low saliency yielded many non salient points. So even though a low saliency resulted in higher repeatability for some scenes, even in those scenes a saliency of 1.6 would provide a more useful set of corners. The other parameter used for the island threshold removal was discussed in section 4.1. Its effect on the performance metrics used was not investigated and instead was fixed based on previous research on the DMD.

These two parameters are similar to the k threshold used for Harris, as both can confidently be fixed in a given range based on empirical evidence, which for the saliency would be a value close to 1.6. The threshold that determines the amount of corners is also very similar for both methods. Lastly, the filter size used for the creation of the density map for the skeleton corner detection is basically the window size for the Harris detector. The skeleton corner detection method thus does not introduce an enormous amount of parameters and instead is very similar to the amount and type of parameters used for the Harris corner detector.

7.1.5 Conclusion

Using the results and subsequent discussion the research question will be answered. The research question introduced in chapter 1 is repeated below:

"How can dense skeletons efficiently and effectively be used for corner detection in grey-scale and color images?"

First of all, the efficiency of the skeleton corner detector highly depends on the efficiency of the skeleton computation. This was discussed in chapter 3. An existing efficient skeleton computation method was described which made use of the AFMM. The effectiveness of the detector relative to other popular corner detectors was evaluated in chapter 6. This was done by measuring the two properties robustness and completeness. The results for the former showed that the skeleton corner detection method obtained a competitive but slight worse overall performance compared to Harris. Completeness was measured in several ways. Performance of the skeleton corner detector was found to be equal according to the matching score and visual analysis, but better for the coverage.

In short, the performance for the skeleton corner detector is competitive and thus can be effectively used for detecting corners in grey-scale images. However, the results show no clear improvement over existing corner detection methods and thus further research would be required. Some possibilities for improving the proposed corner detector are discussed in the following section.

7.2 Future work

This section discusses some possible future research that could increase the overall performance of the proposed method.

7.2.1 Scale estimation

Currently the use of the DMD was investigated for corner detection and compared with other corner detection methods. Another use case was discussed in section 2.5. Assigning a scale to a corner is useful for a task such as matching, so that the descriptor is calculated for the same part of an object in different scales. So far no scale was assigned yet, but doing this using the topological information of the skeletons could provide an accurate scale estimation. This would change the method from a corner detector to a region detector, for which large scale evaluations have already been done [14] [20]. Furthermore, this task could be more suited for the proposed method, because the use of scale space is not required. Therefore, the estimated scale might yield a more exact compared to methods that use scale space, because those methods only select a subset of the possible scales. Another point is that for region detection a correct estimation of the scale is more important than an accurate localization.

An example to illustrate this is shown in figure 7.1. In this case the branch length of an endpoint is used. For a corner the average branch length of all its endpoints is calculated and then assigned as the scale for this corner. The image on the right in figure 7.1 contains the same square as in the image on the left but with the scale

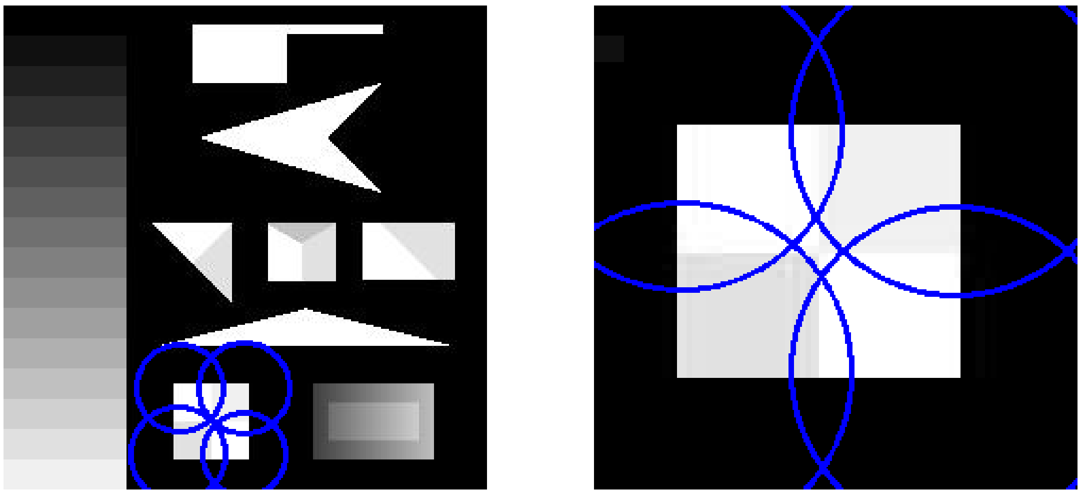


FIGURE 7.1: Regions detected on two images of the same resolution, the image on the right is a scaled up square from the left image. The regions are drawn using the scale assigned by the skeleton corner detector, which is proportional to the branch length of an endpoint. Note that only the relevant regions detected in the image on the left are shown.

increased. These images thus have the same resolution. Furthermore, In this example it is clear that skeletons can be used to correctly assign a scale for a simple image. However, to determine the viability of a region detector using the DMD for grey-scale images of complex scenes further research is required.

7.2.2 Improving the response function

One important part of corner detectors is the ranking of the corners, which requires the assignment of a 'cornerness' value to each corner. This term is slightly confusing because for the skeleton corner detection method it does not describe the geometrical properties of a corner, such as the angle of the corner. Instead it looks at the number of endpoints found in the different threshold sets. This is mostly related to the difference in intensity between the object to which the corner belongs and the background (or vice versa). Because this ranking is very important to the performance results in the evaluation additional attention could be given to it. For example, Li et al. [17] use machine learning to learn a ranking function to assign the 'cornerness' value. The features used in their case are image derivatives. Using additional information about the corners to achieve a better ranking could increase the overall performance. Such information could, besides image derivatives and gradients, also be geometrical properties. For example, the saliency, importance, or other topological information of the extremal region and its skeleton in which the endpoint was found. Furthermore, because a final corner consists of multiple endpoints, variance of the aforementioned properties could be included.

Furthermore the invariance properties are what decide the robustness of a corner detector. The response function should give the same ranking of corners for scenes undergoing deformations, and its relative ranking should thus not change. For the Harris corner detector the problem comes from the fixed sized window, making it not covariant to scale changes. The response function for the skeleton corner detector however counts the number of endpoints, and these are affine covariant (because of the binary skeleton properties). However, for the response function to be completely affine covariant, every step has to be affine covariant. The creation of the density map uses a fixed sized filter, which thus does not make it invariant to scale changes. There are two ways to adjust this. The first option is to accumulate the endpoints using a filter whose scale is covariant. A second option is to accumulate not based on spatial location alone, but also based on direction of the skeleton branch. The former is self explanatory. The latter accumulates points only if they belong to the same object. If the skeleton branches of two endpoints have the same orientation they likely belong to the same object.

7.2.3 Choosing the threshold layers

In the results section the topic of limiting the number of threshold layers was investigated. By reducing the number of layers the speed of the detector naturally improves. So far only a naive reduction of the layers was done by simply choosing the layers uniformly. More carefully choosing what layers to choose could yield speed increases while limiting the performance losses that come from limiting the number of layers. An example where this yields good results is for image compressing using the DMD of an image, which is done by Wang et al. [41]. They select relevant layers based on the histogram of an image.

Bibliography

[1] Mohammad Awrangjeb and Guojun Lu. "Robust image corner detection based on the chord-to-point distance accumulation technique". In: *IEEE transactions on multimedia* 10.6 (2008), pp. 1059–1072.

[2] Vassileios Balntas et al. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5173–5182.

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 224–236.

[4] Timo Dickscheid and Wolfgang Förstner. "Evaluating the suitability of feature detectors for automatic image orientation systems". In: *International Conference on Computer Vision Systems*. Springer. 2009, pp. 305–314.

[5] Timo Dickscheid, Falko Schindler, and Wolfgang Förstner. "Coding images with local features". In: *International journal of computer vision* 94.2 (2011), pp. 154–174.

[6] R Dinesh and DS Guru. "Corner detection using morphological skeleton: An efficient and nonparametric approach". In: *Asian Conference on Computer Vision*. Springer. 2006, pp. 752–760.

[7] D Ebert, P Brunet, and I Navazo. "An augmented fast marching method for computing skeletons and centerlines". In: *Proceedings of VisSym, Barcelona, Spain* (2002).

[8] Shoaib Ehsan et al. "Measuring the coverage of interest point detectors". In: *International Conference Image Analysis and Recognition*. Springer. 2011, pp. 253–261.

[9] Wolfgang Förstner, Timo Dickscheid, and Falko Schindler. "Detecting interpretable and accurate scale-invariant keypoints". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 2256–2263.

[10] Rafael C Gonzalez and Richard E Woods. *Digital image processing, 4th edition*. Pearson, 2018.

[11] Christopher G Harris, Mike Stephens, et al. "A combined corner and edge detector." In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[12] M Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. "Image features detection, description and matching". In: *Image Feature Detectors and Descriptors*. Springer, 2016, pp. 11–45.

[13] Zhanqiang Huo et al. "Improved covariant local feature detector". In: *Pattern Recognition Letters* (2020).

[14] Karel Lenc and Andrea Vedaldi. "Large scale evaluation of local image feature detectors on homography datasets". In: *arXiv preprint arXiv:1807.07939* (2018).

[15] Karel Lenc and Andrea Vedaldi. "Learning covariant feature detectors". In: *European conference on computer vision*. Springer. 2016, pp. 100–117.

[16] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. "BRISK: Binary robust invariant scalable keypoints". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2548–2555.

[17] Bing Li et al. "Rank-SIFT: Learning to rank repeatable local interest points". In: *CVPR 2011*. IEEE. 2011, pp. 1737–1744.

[18] Elmar Mair et al. "Adaptive and generic corner detection based on the accelerated segment test". In: *European conference on Computer vision*. Springer. 2010, pp. 183–196.

[19] Jiri Matas et al. "Robust wide-baseline stereo from maximally stable extremal regions". In: *Image and vision computing* 22.10 (2004), pp. 761–767.

[20] Krystian Mikolajczyk and Cordelia Schmid. "A performance evaluation of local descriptors". In: *IEEE transactions on pattern analysis and machine intelligence* 27.10 (2005), pp. 1615–1630.

[21] Krystian Mikolajczyk and Cordelia Schmid. "Indexing based on scale invariant interest points". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 525–531.

[22] Krystian Mikolajczyk et al. "A comparison of affine region detectors". In: *International journal of computer vision* 65.1-2 (2005), pp. 43–72.

[23] Farzin Mokhtarian and Farahnaz Mohanna. "Performance evaluation of corner detectors using consistency and accuracy measures". In: *Computer Vision and Image Understanding* 102.1 (2006), pp. 81–94.

[24] Farzin Mokhtarian and Riku Suomela. "Robust image corner detection through curvature scale space". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.12 (1998), pp. 1376–1381.

[25] Hans P Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* Tech. rep. Stanford Univ CA Dept of Computer Science, 1980.

[26] Glauco V Pedrosa and Celia AZ Barcelos. "Anisotropic diffusion for effective shape corner point detection". In: *Pattern Recognition Letters* 31.12 (2010), pp. 1658–1664.

[27] Michal Perdoch, Jiri Matas, and Stepan Obdrzalek. "Stable affine frames on isophotes". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[28] Swarna K Ravindran and Anurag Mittal. "Comal: Good features to match on object boundaries". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 336–345.

[29] Ives Rey-Otero and Mauricio Delbracio. "Is repeatability an unbiased criterion for ranking feature detectors?" In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2558–2580.

[30] Edward Rosten and Tom Drummond. "Fusing points and lines for high performance tracking". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. Ieee. 2005, pp. 1508–1515.

[31] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[32] Edward Rosten, Reid Porter, and Tom Drummond. "Faster and better: A machine learning approach to corner detection". In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 105–119.

[33] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. "Evaluation of interest point detectors". In: *International Journal of computer vision* 37.2 (2000), pp. 151–172.

[34] James A Sethian. "A fast marching level set method for monotonically advancing fronts". In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595.

[35] Stephen M Smith and J Michael Brady. "SUSAN—a new approach to low level image processing". In: *International journal of computer vision* 23.1 (1997), pp. 45–78.

[36] Alexandru Telea. "Feature preserving smoothing of shapes using saliency skeletons". In: *Visualization in Medicine and Life Sciences II*. Springer, 2012, pp. 153–170.

[37] Engin Tola, Vincent Lepetit, and Pascal Fua. "Daisy: An efficient dense descriptor applied to wide-baseline stereo". In: *IEEE transactions on pattern analysis and machine intelligence* 32.5 (2009), pp. 815–830.

[38] Tinne Tuytelaars and Krystian Mikolajczyk. "A survey on local invariant features". In: *Foundations and Trends in Computer Graphics and Vision* 3.3 (2008).

[39] Matthew Van Der Zwan, Yuri Meiburg, and Alexandru Telea. "A dense medial descriptor for image analysis". In: *VISAPP (1)*. 2013, pp. 285–293.

[40] Yannick Verdie et al. "Tilde: A temporally invariant learned detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5279–5288.

[41] Jieying Wang et al. "Quantitative Evaluation of Dense Skeletons for Image Compression". In: *Information* 11.5 (2020), p. 274.

[42] C Lawrence Zitnick and Krishnan Ramnath. "Edge foci interest points". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 359–366.