α-shapes for local feature detection

Christos Varytimidis*, Konstantinos Rapantzikos, Yannis Avrithis, Stefanos Kollias

National Technical University of Athens, Iroon Polytexneiou 9, 15780 Zografou, Greece
Phone: +302107722521, Fax: +302107722492

Abstract

Local image features are routinely used in state-of-the-art methods to solve many computer vision problems like image retrieval, classification, or 3D registration. As the applications become more complex, the research for better visual features is still active. In this paper we present a feature detector that exploits the inherent geometry of sampled image edges using *α-shapes*. We propose a novel edge sampling scheme that exploits local shape and investigate different triangulations of sampled points. We also introduce a novel approach to represent the anisotropy in a triangulation along with different feature selection methods. Our detector provides a small number of distinctive features that is ideal for large scale applications, while achieving competitive performance in a series of matching and retrieval experiments.

Keywords: local features, point sampling, triangulation, α-shapes

1. Introduction

Local features provide a balance between the sparseness of global representations and the density of features extracted on a fixed grid of locations. By ignoring non-salient image parts and focusing on distinctive regions they provide repeatability, discriminative power, computational efficiency and compactness. These properties boost computer vision applications including large-scale recognition, retrieval or 3D reconstruction.

State-of-the-art detectors like Hessian-Affine [1], MSER [2] and SURF [3] have been used in many computer vision applications and are quite mature and popular. However, the speed, stability and image coverage provided by those detectors are not ideal. The speed and image coverage of the Hessian-Affine detector are limited, while multiple detections often appear on nearby locations at different scales. The MSER detector is fast, but often extracts sparse regular regions that are not representative enough. SURF is also fast, but detections are often not stable enough. Recent publications compare

*Corresponding author

Email addresses: `chrisvar@image.ntua.gr` (Christos Varytimidis), `rap@image.ntua.gr` (Konstantinos Rapantzikos), `iavr@image.ntua.gr` (Yannis Avrithis), `stefanos@cs.ntua.gr` (Stefanos Kollias)

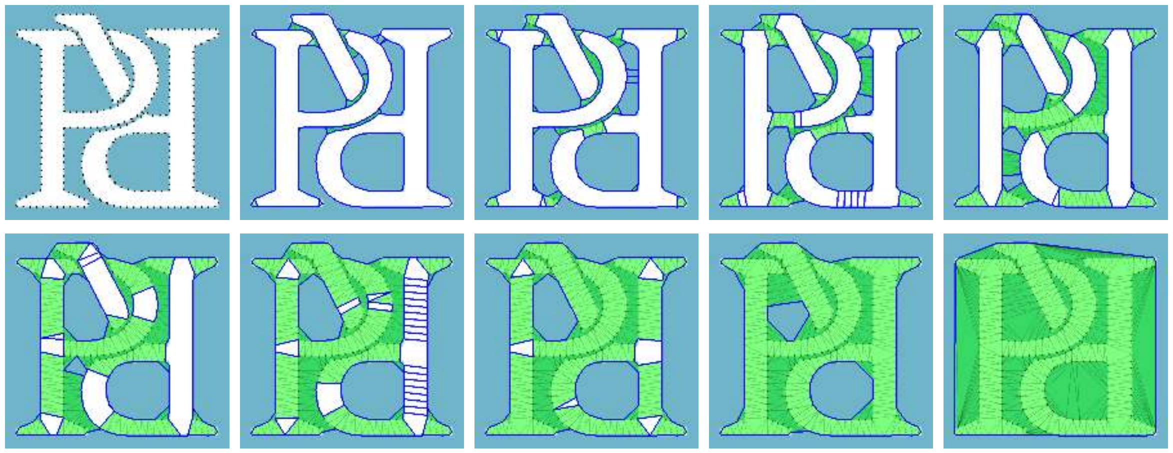


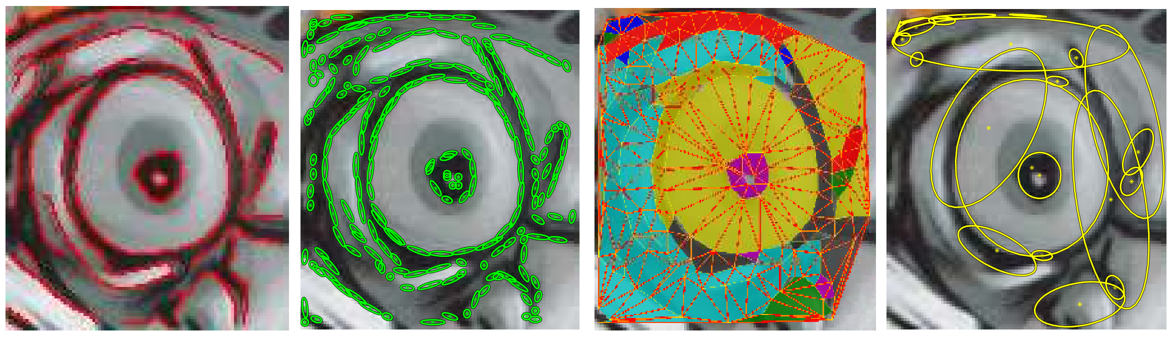
Figure 1: Example of the α-*filtration*. Different instances of the filtration for different values of α: starting from the point set for $\alpha = 0$ (top left) and adding triangles and edges, we end up to the *convex hull* for $\alpha = \infty$ (bottom right). Observe how the cavities of the α-shapes correspond to cavities of the objects and blob–like regions.

state-of-the-art detectors not only by common statistics (e.g. repeatability/matching score), but also in diverse applications like image classification [4] or retrieval [5, 6, 7].

In an attempt to capture the dominant structural features in an image, we propose a detector based on a local shape representation rather than first- or second-order image derivatives. In this direction we employ α-shapes, a well known method in computational geometry introduced by Edelsbrunner *et al*. [8]. An α-shape is a subset of a triangulation of a point set in a Euclidean space, where scale-like parameter $\alpha \geq 0$ determines the *faces* of the triangulation (points, edges or triangles) that are included in the particular subset (see section 3.3). Given a set of points, α-shapes involve a grouping process guided by α, and capture the shape of structures generated by this process. They can be thought of as a generalization of the *convex hull*, being parametrized by α. Starting from the point set for $\alpha = 0$, the subset of the triangulation expands to the convex hull at the other extreme $\alpha = \infty$ (see Fig. 1).

The set of all α-shapes (for all possible values of α) is a *filtration* over a triangulation of the point set, *i.e.* a partial ordering of simplices (edges and triangles in two dimensions) [9]. *Delaunay* triangulation is the most common choice, but since it is only based on point coordinates, it may be a poor representation depending on the application, *e.g.* a sampled function over an image may be more informative. *Weighted α-shapes* [10] on the other hand are based on a *regular* triangulation and provide a more flexible representation, by associating an additional scalar parameter per point. Teichmann & Capps introduce the *anisotropic* α-shapes [11], using an even richer representation per point. Anisotropic α-shapes are a generalization of weighted α-shapes, defined on non–Euclidean metric spaces.

In [12] we introduce WαSH, a detector based on weighted α-shapes that groups edge samples by



(a) (b) (c) (d)

Figure 2: Overview of our detector. (a) Edges of the input image are (b) non-uniformly sampled. (c) We create the α-filtration of a triangulation over the samples and track the evolution of connected components. (d) We extract features by selecting stable and prominent components.

exploiting location, gradient strength and local shape. To capture local shape, we devise an efficient way to overcome the main weakness of α-shapes, namely the automatic selection of α value that best represents the underlying point set. We also show how noisy points or groupings can be automatically filtered out by a shape-based stability measure. WαSH performs quite well and is controlled by a single and intuitive parameter.

In this paper we treat a local feature as a region delineated by a set of points sampled from its contour, but instead of using a uniform sampling scheme along the edges, as in WαSH, we explore a non-uniform scheme whose sampling density is guided by local edge shape. The major steps of the algorithm are summarized in Fig 2, namely (a) edge extraction, (b) non-uniform sampling based on local anisotropy, (c) triangulation of the samples and α-filtration construction, and (d) local feature extraction based on shape measures. Compared to WαSH, we improve and extend the method by

- applying non-uniform sampling based on image edges and local shape;
- introducing anisotropically weighted α-shapes, also adapted to local shape;
- comparing several triangulations to build the α-shapes; and
- proposing and evaluating different measures to select dominant components.

The remaining of the paper is organized as follows: In section 2 we discuss related work, followed by the description of our method. In section 3.1 we describe the different sampling strategies and in section 3.2 we provide an overview of the triangulations used. In section 3.3 we introduce anisotropically weighted α-shapes. In section 4.1 we describe the component trees used to track evolving shapes and in section 4.2 we propose different measures to select dominant components as features, followed by an overview of our algorithm in section 4.3. In section 4.4 we show visual examples and discuss the qualities of our detector. The performance of our detector is experimentally evaluated and compared to the state-of-the-art in section 5, followed by conclusions and discussion in section 6.

2. Related work

Early region detectors were based on extending ideas found in corner detectors like Beaudet [13] and Harris and Stevens [14], which were based on the Hessian and the second moment matrix respectively. In his inspiring work, Lindeberg *et al.* [15] studied scale-invariant detectors and established the theoretical foundations for making them affine-invariant [16]. Based on these foundations, Lowe [17] introduced the *scale invariant feature transform* (SIFT), still one of the most popular detectors, which achieves invariance to scale and rotation based on the Difference-of-Gaussian(DoG) operator. Mikolajczyk *et al.* [1] extended the Harris-Laplace and Hessian-Laplace operators towards affine invariance using the Laplacian-of-Gaussian (LoG) operator in affine scale space.

More recently, Alcantarilla *et al.* [7] introduce the KAZE operator, which detects maxima of the Hessian in a nonlinear scale space built by diffusion filtering. Although the statistics are comparable to the state-of-the-art, the creation of the nonlinear scale-space is computationally expensive and the number of features is high. The fast variant of KAZE in [18] is still slower than the state-of-the-art, while not providing better performance.

The *maximally stable extremal regions* (MSER) of Matas *et al.* [2], one of the best performing region detectors in [19], detects regions of stable intensity and therefore avoids common problems of gradient-based methods like localization accuracy and noise. The idea is to compute a watershed-like segmentation and to select those regions that remain stable over a predefined set of thresholds. MSER are widely adopted due to their high performance, especially on images with planar structures. Most importantly, MSER are arbitrarily shaped areas with explicitly represented boundaries, giving rise to more descriptor alternatives than *e.g.* elliptical regions.

The recent trend of achieving a good balance between efficiency and performance has led to a group of computationally efficient detectors like SURF [3], an approximate version of SIFT, as well as FAST along with its variants (FAST-9, FAST-ER) [20], introducing fast corner detection based on an intensity comparison test in a small neighborhood. BRISK [21] and ORB [22] detectors build on FAST and provide real-time detection and description of local features with matching performance being comparable to SIFT and SURF. Most of these detectors are invariant to scale and rotation, but not affine invariant.

Although image edges are naturally related to object boundaries and therefore to distinct regions, they have attracted less attention, mainly due to instability and computational cost. One of the earliest attempts, *edge-based region detector* (EBR) [23], starts from corner points and expands along nearby edges by measuring photometric quantities. It is suitable for scenes containing intersecting edges (*e.g.* man-made structures like buildings), but not for generic matching, as shown in [19]. Mikolajczyk *et al.* [24] propose an edge-based detector that combines Canny edge detection with automatic scale selection and use it for object recognition. For efficiency, they start from densely sampled edge points. Starting also from Canny edges, Rapantzikos *et al.* [25] compute the binary distance transform and detect regions by a grouping process that is initialized by local maxima of the distance transform and guided by the gradient strength of nearby edges.

Several recent methods are indirectly related to edges by exploiting gradient strength without thresholding. Zitnick *et al.* [5] apply an oriented filter bank to the input image and detect *edge foci* (EF), i.e. points that are roughly equidistant from edgels with orientations perpendicular to the points. This takes region shape into account, but is computationally expensive. Avrithis and Rapantzikos [6] compute the weighted medial axis transform directly from image gradient, partition it in a way similar to topological watershed, and hierarchically detect *medial features* (MFD) taking both contrast and shape into account. Although those methods exploit richer image information, the gradient strength is often quite sensitive to lighting and scale variations. Certain of the above edge-based detectors like [25, 6] yield arbitrarily shaped regions like MSER.

Computational geometry is rich in applications based on *α-shapes*, as opposed to computer vision. One of the earliest applications has been to surface reconstruction from an unorganized set of points [8, 10]. α-shapes have been also used for studying *pockets*, defined as regions with limited accessibility from the outside, measuring the surface area and volume of macromolecules, and the packing density of proteins [10]. Meine *et al.* [26] use them for reconstructing boundaries of noisy edge maps. Starting from binary edge samples they construct the Delaunay triangulation and select a subset of its edges to complete the object boundaries. Their main criteria are triangle size and average color.

Teichmann and Capps [11] introduce the *anisotropic α-shapes* in an attempt to overcome limitations mainly related to discontinuities of the shape, as well as neighboring surfaces that tend to merge. They do not explicitly create an anisotropic triangulation; they rather use Delaunay triangulation and then anisotropically reshape the space covered by each triangle and flip edges when necessary. On the other hand, Labelle and Shewchuk [27] create an anisotropic triangulation via an anisotropic Voronoi diagram that is always well defined, *i.e.* there are no orphan regions that do not contain their generating point.

Related is the work of Cazals *et al.* [28], introducing the *conformal α-shapes* and the corresponding filtration. Conformal α-shapes are based on a global scale parameter $\hat{\alpha}$ and two local ones, adjusted to some neighborhood of each point. This representation is applied to surface reconstruction of non-uniformly sampled surfaces. Along the same direction, Zomorodian *et al.* [9] predict protein structure by employing α-shapes to detect interacting atoms in a protein molecule.

Building on edge-based methods, we start from sampled edges like [24, 23]. We then create a

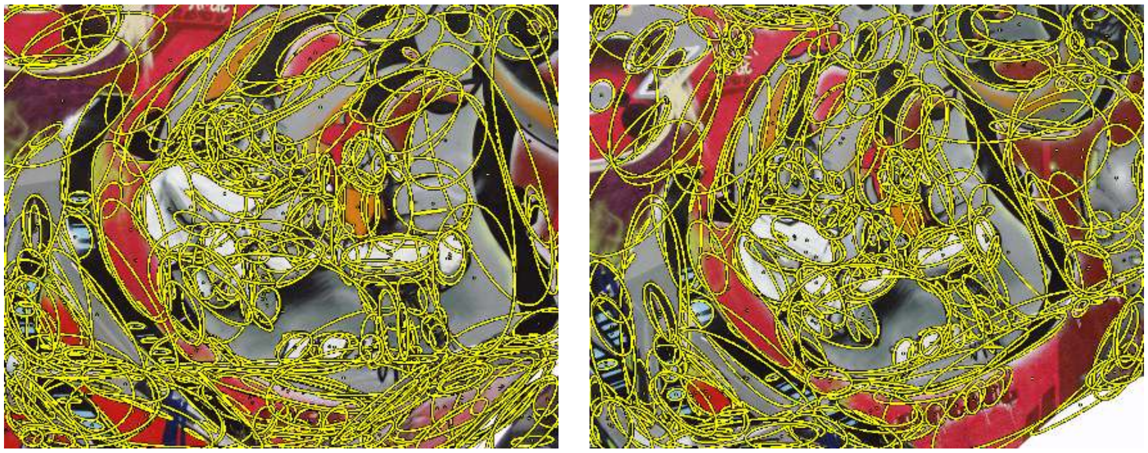


Figure 3: Features extracted by our detector, using anisotropically weighted α-shapes and non-uniform sampling.

2D triangulation of the points and construct the α-filtration like [8, 29, 11, 27], experimenting with different triangulations. We employ the component tree, which is a hierarchical representation of nested sets, similar to MSER [2]. However, we apply this representation on the α-filtration rather than the level sets of image intensity. To select image features we use shape-based criteria, a choice that bears similarities to [6], although our geometrical representation is entirely different.

Our method is able to detect regions that are adjacent to both brighter and darker ones, as opposed to MSER that can only detect bright or dark extremal regions. Furthermore, the proposed detector can handle regions determined by cavities of the boundary shape, bearing similarities to the *pockets* [10], and regions that are not enclosed by complete boundaries.

3. Building α-shapes from images

3.1. Image sampling

In this section we discuss our sampling methods, following our visual representation and edge detection. We introduce two alternative sampling methods. By *uniform sampling*, points are sampled uniformly along edges with a fixed *sampling interval* s. On the other hand, *non-uniform sampling* is controlled by a measure of local anisotropy. Samples typically do not correspond to key-points like maxima of curvature. Instead, the goal is to capture the shape of the detected boundaries while keeping samples as sparse as possible.

3.1.1. Image representation

We assume a grayscale input image is given as a function f on the plane and that g is its gradient magnitude $\|\nabla f\|$ normalized in $[0, 1]$. We apply a binary edge detector on g and sample the resulting edge map E to obtain a discrete set of edge points $P \subseteq \mathbb{R}^2$. The domain and range of functions f, g are both bounded and discrete, and P is finite.

The binary edge map is obtained by the *Canny* edge detector [30]. After the non-maxima suppression stage of the algorithm, we follow all candidate edge pixels and apply the hysteresis thresholds. We do not rely on a clear edge map and therefore the high and low hysteresis thresholds of the detector are kept fixed. We start from a random image pixel that is an edge candidate (maximum from the previous stage) and sample points along the edge based on an 8-connected neighborhood. The first point visited in an edge is automatically considered a sample.

3.1.2. Uniform sampling

Using a fixed sampling interval s, we count the steps taken from the initial sample while moving along an edge using an 8-connected neighborhood. When the number of steps reaches s, we sample a new point at the current location and reset the counter. This implies that the Euclidean distance between two adjacent samples will be greater than or equal to s.

For each sample point $p \in P$, we define its *weight* $w(p) \geq 0$ to be proportional to its gradient strength $g(p)$,

$$w(p) = g(p) \left(\frac{s}{2}\right)^2, \tag{1}$$

with $g(p) \in [0, 1]$. The reason for this choice of weight w will be made clear at the end of section 3.3.1.

3.1.3. Non-uniform sampling

To adapt the sampling interval to local edge shape, we capture the local affine shape at the current sample, measure a number of properties and determine the sampling interval for the following sample.

The local affine shape adaptation scheme we use is based on Lindeberg's shape adaptation [31, 32]. Given a sample point p, we follow an iterative process whereby at each iteration i we measure the local shape $U^{(i)}$ of a shape-adapted neighborhood $\mathbf{p}$ of p, where $U^{(i)}$ is modeled by a 2×2 transformation matrix. We initialize local shape to isotropic of unity scale, *i.e.* $U^{(0)}$ is the identity matrix I. At iteration i, we transform neighborhood $\mathbf{p}$ to $\hat{\mathbf{p}} = U^{(i)}\mathbf{p}$, resample image intensity on $\hat{\mathbf{p}}$, and compute the average *second-moment matrix*

$$\mu = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{12} & \mu_{22} \end{bmatrix} = \begin{bmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{bmatrix}. \tag{2}$$

over $\hat{\mathbf{p}}$, where L_x, L_y are the first order derivatives. The transformed neighborhood $\hat{\mathbf{p}}$ is resampled at the same size as the initial neighborhood $\mathbf{p}$, which is fixed (see section 5.2). In order to prevent overfitting in case of straight lines in the small neighborhood of p, we set an upper bound k on the eccentricity of the ellipse associated to μ. Following [32], we measure the following quantities for μ:

$$\begin{aligned} B &= \mu_{11} + \mu_{22} && (3) \\ C &= \mu_{11} - \mu_{22} && (4) \\ S &= 2\mu_{12} && (5) \\ Q &= \sqrt{C^2 + S^2} && (6) \end{aligned}$$

Here Q measures the degree of anisotropy, while $\hat{Q} = Q/B$ is normalized in $[0, 1]$. The eccentricity of the associated ellipse of μ is then $(1+\hat{Q})/(1-\hat{Q})$. It is thus shown in [32] that the *regularised diffusion matrix*

$$\Sigma = (\mu + \epsilon I)^{-1} \tag{7}$$

with $\epsilon = \frac{Q}{k-1}$, has an associated ellipse of the same orientation as that of μ, but not too elongated—its eccentricity is upper bounded by k.

We compute a square root of Σ and normalize its eigenvalues $\lambda_{\min}$, $\lambda_{\max}$, so that

$$\lambda_{\min}\lambda_{\max} = 1. \tag{8}$$

Finally, we concatenate transformation matrices by applying $\Sigma^{-\frac{1}{2}}$ to $U^{(i)}$ so that

$$U^{(i+1)} = \Sigma^{-\frac{1}{2}} U^{(i)}. \tag{9}$$

The ratio of the eigenvalues is considered for convergence. If it is close to 1, we assume to have reached convergence, at which point we measure the final local shape matrix U. Convergence typically occurs in 3 to 5 iterations.

The largest eigenvalue is used to compute the scale of the adapted neighborhood $\mathbf{p}$. Since the shape of $\mathbf{p}$ is described by $U^{(i)}$, its scale is the product of the largest eigenvalues over all iterations up to i. The sampling interval s is controlled by this scale, and is now dependent on p:

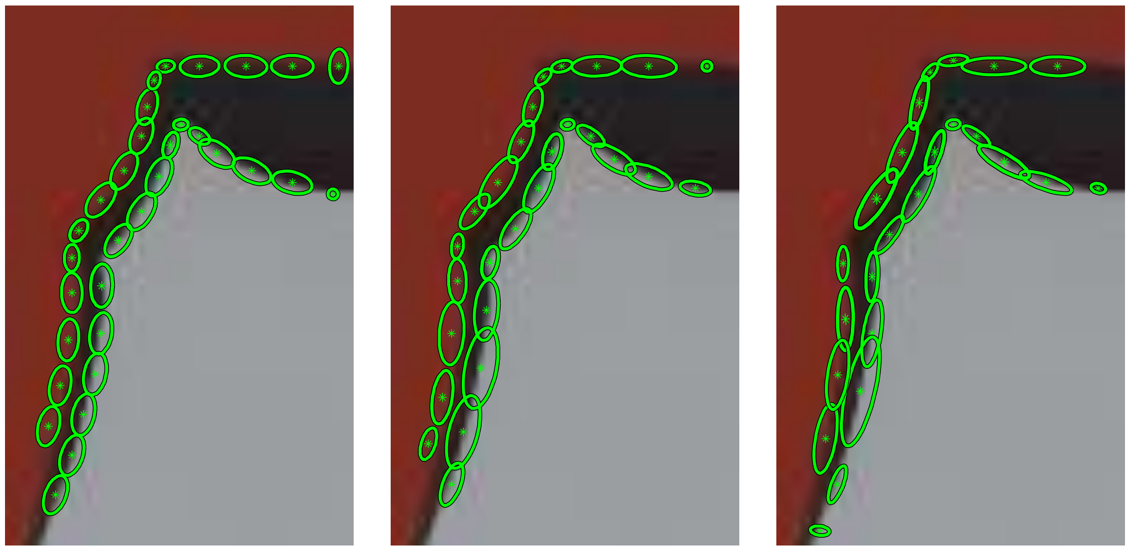
$$s(p) = \prod_i \lambda_{\max}^{(i)}. \tag{10}$$

This means that the next point after p is sampled after $s(p)$ steps along the edge. Finally, if U is the final local shape matrix measured at p at termination, metric tensor

$$M_p = U^{\mathrm{T}} U \tag{11}$$

is used to define distances in the neighborhood of p. This tensor is used by the anisotropically weighted α-shapes in section 3.3.2.

Fig. 4 illustrates how the proposed non-uniform sampling method captures local shape changes while minimizing the number of samples. Where edges tend to straight lines, the sampling is sparse,



$k = 2$ $k = 3$ $k = 6$

Figure 4: Sample points using variable density sampling. Increasing the upper bound of eccentricity k of the regions results in sparser samplings.

yet limited by the upper bound k on eccentricities. Where the curvature increases, the sampling density increases as well to capture the local details of the shape. The same figure illustrates the effect of upper bound k. Small k leads to fine representation, hence increased complexity; excessively large k not only leads to coarse representation, but may cause nodes to disappear due to overlapping of ellipses, as will be explained in section 3.2.1.

Discussion. In the above non-uniform sampling method, the sampling interval is affected by neighboring edges but there is no exact relation between the two in theory. An ϵ-sample of the edges as defined in [28] provides such theoretical guarantees; however, this requires constructing the *medial axis* first, which is not only harder to compute, but also typically requires some form of sampling in the first place.

Alternatively, the more common option of adjusting the sampling interval according to *edge curvature* is very fast since it effectively operates in one dimension, but does not take into account neighboring edges at all. This potentially results in too sparse representations when two edges are close, and can be detrimental to subsequent triangulation and feature detection. Our semi-local approach can be seen as a compromise between these two alternatives.

3.2. Triangulations

In the following we define the different triangulations used in our method, which form the basis for the computation of α-shapes.

3.2.1. Regular triangulation

Regular triangulations are in fact a family of triangulations, parametrized by a *height function.* A triangulation of a point set P in $\mathbb{R}^n$ is called *regular* if it can be obtained by projecting the lower envelop of a lifting of P to $\mathbb{R}^{n+1}$ [33]. A 2D set of points is lifted to 3D by assigning a *height* value to each point p, using height function $\omega : \mathbb{R}^2 \to \mathbb{R}$. A lifted point is a 3D point located at $(p_x, p_y, \omega(p))$. Next, the lower envelop of the lifted points in $\mathbb{R}^3$ is computed, which is the lower part of the 3D convex hull of the points. By projecting this lower envelop to $\mathbb{R}^2$, a tessellation of the 2D convex hull of the points is produced. This tessellation is a regular triangulation.

In this work, we use the *additively weighted* height function

$$\omega(p) = \|p\|^2 - w(p), \tag{12}$$

with $w(p)$ as defined in (1). A point $p \in P$ along with its weight $w(p)$ makes up a pair $(p, w(p))$ that is called a *weighted point.* A *weighted point* can be seen as a *circle* centered at p, with radius $\sqrt{w(p)}$. We will say this is the circle of point p. We will also use the same symbol p for both a circle or weighted point, and will disambiguate as necessary.

The weights w have to be carefully selected because they can make a point disappear. This occurs in case the circle of a point p is contained in the interior of the circle of another point q. In 3D space, the lifted point of p does not belong to the lower hull in this case.

The regular triangulation can be computed incrementally, by adding the points one by one. Edelsbrunner and Shah [34] have proved that such a construction is possible and its time complexity is $O(n \log n + n)$ in the 2D case, where $n = |P|$.

3.2.2. Delaunay triangulation

The Delaunay triangulation is the most commonly used triangulation. It belongs to the family of regular triangulations, and corresponds to height function $\omega(p) = \|p\|^2$. This is equivalent to zero weight, $w(p) = 0$ for all $p \in P$ in our regular triangulation with additive weights (12). Since the weight function is directly computed from the coordinates of the sample points in $\mathbb{R}^n$, no additional input is required.

An incremental *edge-flipping* algorithm is used to construct the Delaunay triangulation, with time complexity $O(n \log n)$. The algorithm starts from a set of three points determining a single triangle and adds one point at a time. The new triangles formed at each iteration are checked against the Delaunay properties and if these are not satisfied, then the common edge of two such adjacent triangles is flipped.

3.2.3. Constrained Delaunay triangulation

The constrained Delaunay triangulation is only partially Delaunay. The properties of the Delaunay or in general the regular triangulations are not necessarily satisfied here. As input, apart from the

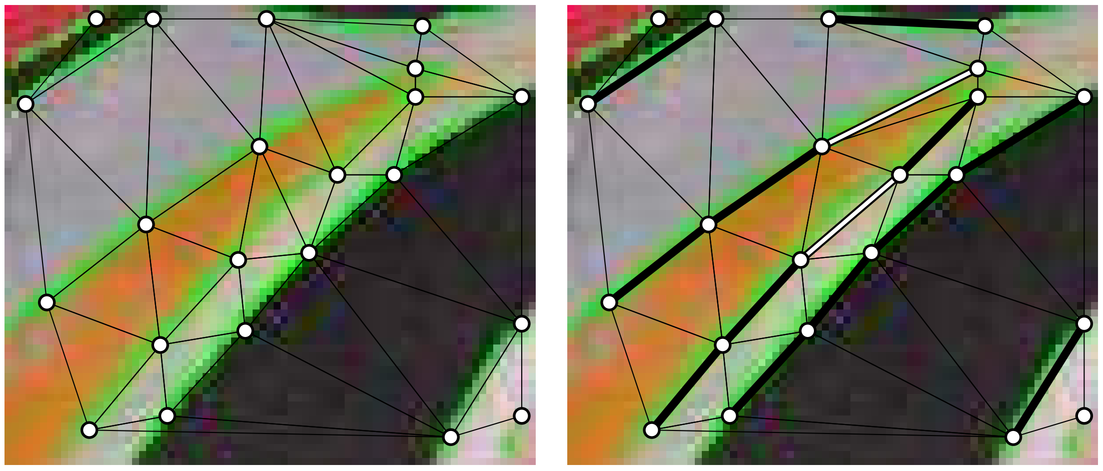


Figure 5: (a) Delaunay triangulation. (b) Constrained Delaunay triangulation, obtained from (a). Constraints are shown as thick black edges; constraints causing flips as thick white edges. Edge pixels are shown in green. Observe that constraints enforce triangulation edges to follow image edges, hence the triangulation represents the underlying structures more accurately.

spatial coordinates of the sample points in $\mathbb{R}^n$, a set of constraints is given, indicating the presence of particular edges. Each constraint c_{ij} is represented by an edge of the form $c_{ij} = (v_i, v_j)$, indicating that vertices v_i and v_j should be connected in the triangulation. The constraints override the Delaunay properties.

In our case, the constraints are defined as consecutive line segments on image edges. In particular, while following an image edge to sample points (see section 3.1), we add as a constraint each pair of consecutive sampled points. This ensures that the triangulation edges do not cross the image edges, creating a representation of the image that respects object boundaries.

A constrained triangulation is typically constucted by a variant of the previous algorithm for Delaunay. In this case, constraints are always added as edges; however, if two such edges intersect, they are first split at their intersection point. This may happen in the case of junctions in the image edges. On the other hand, a sequence of flips is performed to accommodate for non-constrained edges that intersect with constrained ones. An example of a constrained Delaunay triangulation is shown in Fig. 5.

3.3. α-shapes

The construction of α-shapes is based on the Delaunay triangulation. Weighted α-shapes are based on a regular triangulation with an additively weighted height function (12). Anisotropically weighted α-shapes further use metric tensor (11) obtained by non-uniform sampling.

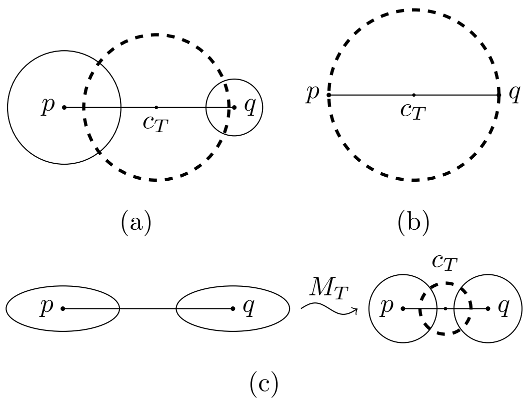


Figure 6: Size of a 2-point set $\{p, q\}$, as the squared radius of a circle (weighted point) c_T that is orthogonal to circles p, q, on a (a) regular and a (b) Delaunay triangulation. (c) On anisotropically weighted α-shapes, the space is transformed to "isotropic" before measuring the size.

3.3.1. Weighted α-shapes

The discussion given here mostly follows [29], but simplifies for the case of 2 dimensions. We begin with a definition of a regular triangulation that is alternative to that of section 3.2.1, because we need a number of additional quantities for the definition of weighted α-shapes. Recall that we use the same symbol p to refer to either a circle or weighted point $(p, w(p))$. Each triangle, line segment or point in a triangulation is a 2-, 1-, or 0-simplex respectively, and we will refer to it as *simplex* in general.

Given two weighted points $p, q \in P$, we define

$$\pi(p, q) = \|p - q\|^2 - w(p) - w(q). \tag{13}$$

Circles p, q intersect at right angles iff $\pi(p, q) = 0$; we then say that p, q are *orthogonal*. Given a point x of zero weight, $\pi(p, x)$ is called the *power* of point x with respect to circle p [35]. Given a subset $T \subseteq P$ of three non-collinear points, there is a unique circle that is orthogonal to all circles of T. Its center has equal powers with respect to the circles of T, and is called their *radical center* [35]. We denote the corresponding weighted point by c_T. Now, consider the 2-simplex (triangle) $\sigma_T = \text{conv}(T)$, the *convex hull* of T. This simplex is called *regular* if

$$\pi(p, c_T) = 0 \qquad \text{for all } p \in T, \tag{14}$$

$$\pi(p, c_T) > 0 \qquad \text{for all } p \in P \setminus T, \tag{15}$$

where (14) is equivalent to c_T being orthogonal to all points of T. The collection $\mathcal{R}$ of all regular triangles over P is called the *regular triangulation* of P. This definition is equivalent to that of section 3.2.1. Observe that, if $w(p) = 0$ for all $p \in T$, weighted point c_T reduces to the *circumcircle* of triangle σ_T, while power function (13) reduces to the Euclidean distance. In this unweighted case, the triangulation reduces to *Delaunay*.

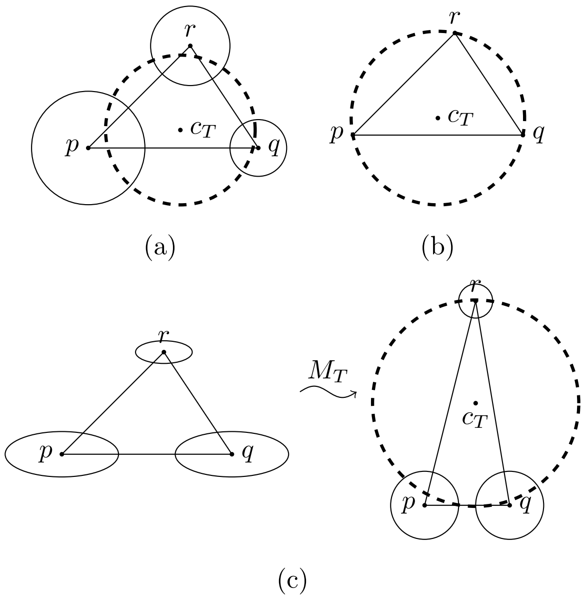


Figure 7: Size of a 3-point set $\{p, q, r\}$, as the squared radius of a circle c_T that is orthogonal to p, q, r, on a (a) regular and a (b) Delaunay triangulation. (c) On anisotropically weighted α-shapes, the space is transformed to "isotropic" before measuring the size.

The collection $\mathcal{K}$ of all 2-simplices (triangles) and their faces (line segments and points) in $\mathcal{R}$ is a *simplicial complex*. If we define a *size* $\rho_T \geq 0$ for each simplex $\sigma_T \in \mathcal{K}$, then the *weighted α-complex* of P is the subset of $\mathcal{K}$ containing all simplices up to a given size $\alpha \geq 0$,

$$\mathcal{K}_\alpha = \{\sigma_T \in \mathcal{K} : \rho_T < \alpha\}. \tag{16}$$

Finally, the *weighted α-shape* of P [10] is the union of all such simplices,

$$\mathcal{W}_\alpha = \bigcup_{\sigma \in \mathcal{K}_\alpha} \sigma. \tag{17}$$

$\mathcal{W}_\alpha$ is a *polytope* that is neither convex nor connected, in general. In the particular case of $\alpha = +\infty$,

$$\mathcal{W}_{+\infty} = \bigcup_{\sigma \in \mathcal{R}} \sigma = \operatorname{conv}(P), \tag{18}$$

the convex hull of P. The difference between the α-shape and the α-complex is analogous to the difference between the convex hull and the triangulation of a point set.

It remains to define the *size* ρ_T of a point, line segment or triangle σ_T, when T contains $1, 2$ or 3 points of P, respectively. The size of a point $p \in P$ is always zero. For line segments and triangles, we use again orthogonal circles, as illustrated in Fig 6, 7.

Given a set of two points $T = \{p, q\} \subseteq P$, there is a *pencil* (infinite collection) of *coaxial circles* that are orthogonal to circles p, q; their centers lie on a line perpendicular to the line segment $\sigma_T = \sigma_{\{p,q\}}$, that is called the *radical axis* of p, q [35]. However, there is a unique circle c_T of minimum radius over this pencil, subject to (15). The size ρ_T of line segment σ_T is the squared radius of circle c_T. In the absence of other points, the center of c_T lies on σ_T, as shown in Fig. 6a. In the Delaunay case (zero weights), σ_T is a diameter of c_T (see Fig. 6b).

Similarly, given a set of three non-collinear points $T = \{p, q, r\} \subseteq P$, there is a unique circle c_T that is orthogonal to p, q, r, as noted above. Again, the size ρ_T of triangle σ_T is the squared radius of circle c_T, as shown in Fig. 7a. In the Delaunay case, c_T is the circumscribed circle of triangle σ_T (see Fig. 7b). In general, since c_T is also orthogonal to p, q alone, and since the size of edge $\sigma_{\{p,q\}}$ is the smallest such circle, a triangle is by definition not smaller than any of its edges. This is evident by comparing Fig. 6a,b to Fig. 7a,b respectively.

From the definition of weight function (1) it follows that the weight $w(p)$ of any point p takes values in $[0, (s/2)^2]$, where s is the sampling interval. For any two consecutive points $T = \{p, q\}$ sampled along an image edge, this ensures that the associated circles shown in Fig. 6a do not intersect, in which case the size of the edge σ_T would be zero. This also prevents one circle being contained in another in general, which would make points disappear from the triangulation.

The above general definition of the *weighted α-shape* may apply to regular, Delaunay, or constrained Delaunay triangulations. For Delaunay, we simply set all point weights to zero. For constrained Delaunay, this does not suffice; we also set the size of all constrained edges to zero. As discussed in section 4.1, this makes constrained edges indeed preserve the boundary of adjacent image regions.

3.3.2. Anisotropically weighted α-shapes

Our anisotropically weighted α-shape is based on weighed samples accompanied by the metric tensor M (11) obtained by the shape adaptation process of non-uniform sampling, as described in section 3.1.3. This applies even to uniform sampling, by invoking the shape adaptation process and explicitly computing M at each sample.

We begin by creating a regular triangulation using weighted points as described in section 3.3.1. Given the metric tensor M_p associated with each point p, we compute an effective metric tensor M_T for each remaining simplex T in the triangulation. In particular, we define tensor M_T of 1- or 2-simplex (edge or triangle) T as the average of metric tensors of individual points in T, normalized for unit determinant such that area is preserved. That is, given points p, q, r with metric tensors M_p, M_q, M_r respectively, we define

$$M_T = \frac{M_p + M_q}{\sqrt{\det(M_p + M_q)}} \tag{19}$$

for an edge defined by $T = \{p, q\}$, and

$$M_T = \frac{M_p + M_q + M_r}{\sqrt{\det(M_p + M_q + M_r)}} \tag{20}$$

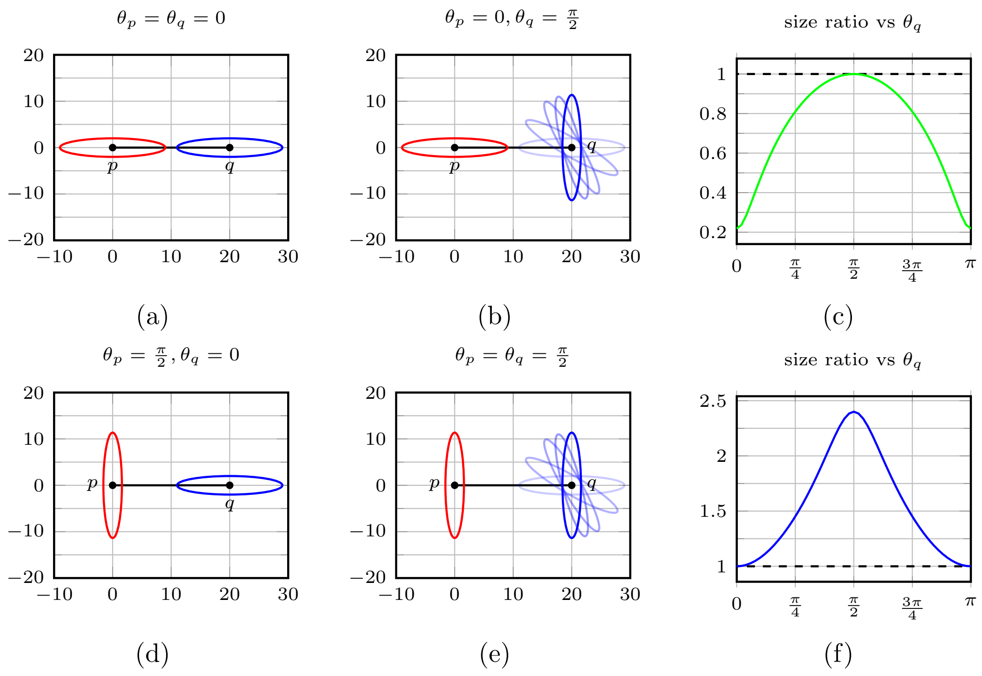


Figure 8: Measuring the *size* ρ_T of a line segment σ_T between two points p, q in the anisotropic case, for different orientations θ_p, θ_q of associated ellipses, where θ_p is fixed to horizontal in upper row (a,b) or vertical in lower row (d,e). In each case, the right column (c,f) shows the ratio of ρ_T to the same measurement in the isotropic case versus θ_q.

for a triangle defined by $T = \{p, q, r\}$.

Fig. 8 gives an example of applying (19) to the edge between points p and q for different relative orientations between associated ellipses, and the ratio of the resulting edge size to the corresponding measurement in the isotropic case. If the major axes of the ellipses are aligned as in Fig. 8a, the edge size decreases as if p, q are coming closer ($\theta_q = 0, \pi$ in Fig. 8c). At the other extreme, if the axes are parallel as in Fig. 8e, the edge size increases as if p, q are moving apart ($\theta_q = \pi/2$ in Fig. 8f). Finally, if the two major axes are normal as in Fig. 8b,d, the effects of the two metric tensors cancel and the edge size is equal to that of the isotropic case ($\theta_q = \pi/2$ in Fig. 8c and $\theta_q = 0, \pi$ in Fig. 8f).

The effect of metric tensor M_T can be seen as applying a linear transformation to the space around simplex T, given by an effective local shape matrix U, such that $M_T = U^{\mathrm{T}} U$. This transformation makes the space "isotropic", *i.e.* ellipses reduce to circles in Fig. 6c, 7c before measuring the simplex size. Equivalently, it gives raise to local inner product defined by $\langle x, y \rangle_T = \langle Ux, Uy \rangle = x^{\mathrm{T}} M_T y$ for $x, y \in \mathbb{R}^2$, hence to local distance metric d_T defined by

$$d_T(x, y) = \sqrt{(x - y)^{\mathrm{T}} M_T (x - y)} \qquad (21)$$

for $x, y \in \mathbb{R}^2$. We use d_T to measure all distances between two or three points in an edge or triangle T; these would be identical to the Euclidean distances measured in the transformed space of Fig. 6c, 7c, respectively. Given the distances and point weights, we measure the size of the simplex exactly as in the isotropic case.

Discussion. Our work differs from Teichmann and Capps [11] in that they only use an orientation per point and transform the space using prefixed scales, whereas we exploit the entire metric tensors obtained from shape adaptation, corresponding to general linear transformations. Further, the triangulation in [11] is initialized as Delaunay and subsequently transformed to partially anisotropic via a sequence of local flips, which is expensive and not guaranteed to terminate. To overcome this, the authors propose to only allow each edge to be flipped once, which is not deterministic since triangulation depends on the visiting order of simplices. We rather use a regular triangulation with direct measurements of simplex sizes using the metric tensors; this gives comparable results to [11] while being much faster.

On the other hand, Labelle and Shewchuk [27] start from an anisotropic Voronoi diagram using a metric tensor at each point to define a local distance metric per cell, as in (21). To convert to a triangulation, the Voronoi diagram is relaxed such that it does not contain any orphan regions. The time complexity of creating an anisotropic Voronoi diagram is $O(n^{2+\epsilon})$, where ϵ is a positive constant, which is prohibitive.

4. Exrtacting local features

4.1. Tracking components in the filtration

All simplices of the simplicial complex $\mathcal{K}$ are typically ordered by ascending size to obtain what is called a *weighted α-filtration* [10]. In this work, we deviate from this standard setting in two ways. First, we only consider *triangles* and their *edges* (line segments), discarding points. We thus construct complex

$$\mathcal{K}' = \{\sigma_T \in \mathcal{K} : |T| \geq 2\}. \tag{22}$$

This choice is justified because edge size helps control connectivity of triangles, while points do not. Second, contrary to (16), we consider the *upper α-complex*

$$\overline{\mathcal{K}}_\alpha = \mathcal{K}' \setminus \mathcal{K}_\alpha = \{\sigma_T \in \mathcal{K}' : \rho_T \geq \alpha\} \tag{23}$$

ordered by *descending* size. As in [10], we need only consider a finite set of values for α. In particular, we sort all $\sigma_T \in \mathcal{K}'$ by descending size ρ_T to obtain the sequence $(\sigma_1, \ldots, \sigma_n)$ where $n = |\mathcal{K}'|$. If ρ_i is the size of σ_i for $i = 1, \ldots, n$, then $\rho_1 \geq \ldots \geq \rho_n$. Now, if $K_i = \{\sigma_1, \ldots, \sigma_i\}$, we obtain the *upper α-filtration*

$$\emptyset = K_0 \subseteq \cdots \subseteq K_n = \mathcal{K}'. \tag{24}$$

Starting from the largest element of size ρ_1 and decreasing the value of α towards ρ_n, the upper α-complex models the growing *cavities* of the original shape. To capture its evolving topology, we construct a *component tree*, similar to [36].

To define *connectedness* on the complex, we specify neighboring relations as follows: the neighbors of each triangle $\sigma_T \in \mathcal{K}'$ (with $|T| = 3$) are its three edges, while the neighbors of each edge σ_T (with $|T| = 2$) are the two adjacent triangles in the triangulation. We denote the *neighborhood* of simplex $\sigma \in \mathcal{K}'$ by $N(\sigma)$. According to the descending order, and since an edge in a regular triangulation is not larger than its two adjacent triangles, the intuition is that this edge can keep the two triangles disconnected until it is processed itself. Eventually, this timing depends on image gradient and local shape.

Given this neighborhood system, we start off with all simplices in $\mathcal{K}'$ being individual components, and process them in descending order of size, joining them with their neighbors that have already been processed. This process is specified in Algorithm 1. For each component κ_T in the component tree, we define a size ρ_T. This is the size of the last simplex σ_T that was added to the component (smallest one) and changed its area. Alternatively, this is the first added simplex that caused a merging of two previously disjoint components.

In Fig. 9 we see an example of the upper α-filtration, along with the proposed neighborhood definition. For $\alpha = \infty$ we start by the empty set, and as the value of α decreases, more triangles and edges are added to the upper α-complex. For $\alpha = 0$ we end up with the convex hull of the input points. Observe the blue edges, which are not part of the α-complex at each instance. Using the proposed neighborhood definition, these edges prevent the different components from merging. The resulting components correspond to interpretable parts of the depicted objects.

4.2. Measuring significance

Starting from large values of α and tracking the evolution of the upper α-complex up to the entire image for $\alpha = 0$, different connected components are formed. These components typically lie on image regions with distinctive boundaries. The significance of a component depends on stability, as measured across the α-filtration, of the corresponding image region.

In order to measure this significance, we compute a *strength* for each component. In particular, consider component κ_U of size ρ_U, which is a set of simplices (triangles and edges) of size greater than ρ_U. When κ_U is joined via a boundary edge σ_T of size ρ_T to another component (say $\kappa_{U'}$) to create a new component κ_T, we compute the *strength* of κ_U (as well as $\kappa_{U'}$). This means that the image region underlying κ_U is surrounded by image edges and the length of its widest opening is $\sqrt{\rho_T}$.

We choose to evaluate strength measures requiring minor computational overhead, focusing on information that is readily available. We have experimented with different strength measures that

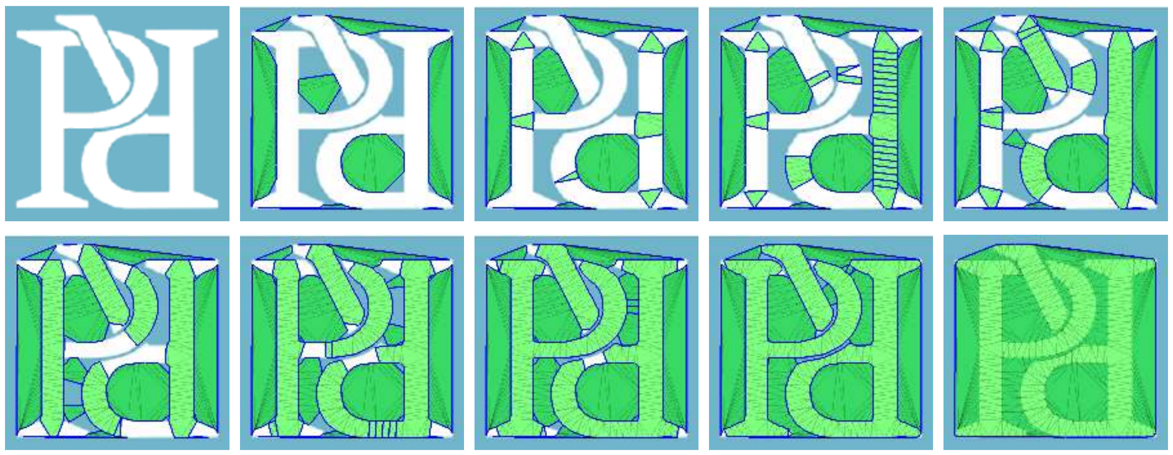


Figure 9: Example of the *upper α-filtration.* From top left to bottom right α decreases, resulting in different instances of the filtration. Edges in blue are not yet processed, keeping the components disjoint. This is in contrast to Fig. 1, where neighboring triangles always belong to the same component and α is in increasing order.

exploit different visual and geometrical properties of image regions under consideration. The first one, is based on the area of a component compared to its maximum boundary opening:

$$s(\kappa_U) = \frac{a(\kappa_U)}{\rho_T}, \tag{25}$$

where $a(\kappa_U)$ is the total area of component κ_U—precisely, the area of the union of all simplices in κ_U. Since we are processing simplices in descending order of size, ρ_T stands in fact for the largest opening over the boundary of component κ_U. It follows that this strength measure favors large components with small (or no) openings on their boundary, so it is called *closure.* Since simplex size ρ measures squared length, closure is a scale invariant quantity. In order to select a component as an image feature, we compare its strength against a threshold τ.

A second computationally inexpensive strength measure is associated with the lifetime of each component. The lifetime is measured as the difference between the component size ρ_U and the size ρ_T of the newly added edge, which forms component κ_T. This *lifetime* strength measure defined as

$$s(\kappa_U) = \rho_U - \rho_T, \tag{26}$$

which, due to the descending order, is always positive. Again, all components with lifetime greater than a threshold τ are selected as features.

A third measure is similar to that used in MSER. The MSER detector forms components on actual image pixels, where processing order is determined by intensity level. Its strength measure is defined as the relative change of the area of a component for a given change in intensity. In our case, when component κ_U is merged with another one and creates component κ_T, this measure would be

$$s(\kappa_U) = \frac{a(\kappa_U) - a(\kappa_T)}{a(\kappa_U)},$$

which is also positive. However, since changes in component areas are not as smooth in our case as when working with pixels, this measure is rather noisy. Hence we propose a strength measure that combines lifetime as well:

$$s(\kappa_U) = (\rho_U - \rho_T)\frac{a(\kappa_U)}{a(\kappa_U) - a(\kappa_T)}. \tag{27}$$

The form of (27) is inversely proportional to a relative rate of change in area, so this strength measure is called *stability.* As in MSER, a component is stable if its strength is locally maximum over the filtration. A stable component is selected as feature if this local maxima exceeds neighboring values by at least a threshold τ.

Finally, we exploit an *ellipticity* measure to evaluate its natural connection to the shape of regions we aim to detect. We first compute the moments up to order 2 (namely m_{00}, m_{10}, m_{01}, m_{11}, m_{20} and m_{02}) of all triangles in the triangulation. Moments are computed analytically using only the coordinates of the three vertices of each triangle. Then, while constructing the component tree, we incrementally compute the moments of every component in the filtration using the moments of its two children components. This requires only a few extra additions at each tree node.

After the component tree is constructed, we traverse it once more to compute the *central moments* μ at each tree node, as described in [37]. At the same time, we compute the simplest affine moment invariant, as described in [38]:

$$I_1 = \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4}.$$

Finally, the ellipticity of a shape is defined as

$$s(\kappa_U) = \begin{cases} 16\pi^2 I_1 & \text{if } I_1 \le \frac{1}{16\pi^2}, \\ \frac{1}{16\pi^2 I_1} & \text{otherwise.} \end{cases} \tag{28}$$

This measure takes values in $[0, 1]$, and is maximized for a perfect ellipse [39]. Again, ellipticity is compared to a threshold τ in order to select a component as a feature.

4.3. Feature detection algorithm

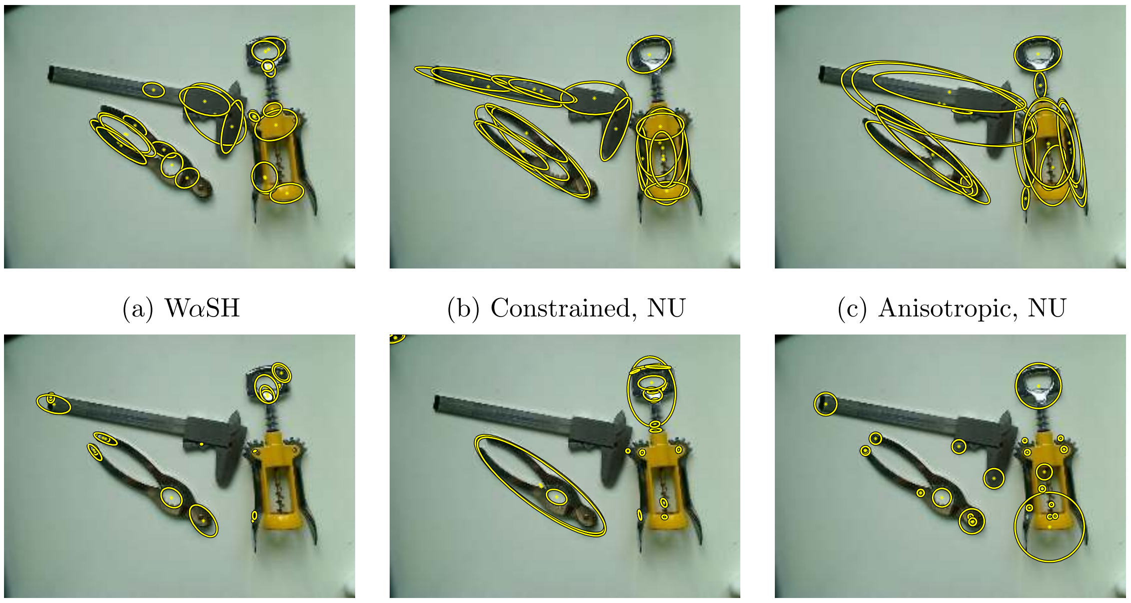
The pseudocode of the entire method is given in Algorithm 1. Initially, the normalized gradient per pixel is computed and given as input to the CANNY operator. The resulting edges E are sampled using one of the methods described in section 3.1. The sampled points P are triangulated using Delaunay, constrained Delaunay or regular triangulation as described in section 3.2. Given the triangulation $\mathcal{R}$, the simplicial complex $\mathcal{K}$ and the simplex size map ρ are then computed. In the case of anisotropically weighted α-shapes, the triangulation has to be regular. The neighborhood map N of the triangles and edges is computed, while complex $\mathcal{K}'$ is as defined in (22).

We use two different tree structures to keep track of connected components, as in [36]. The first is a forest where each simplex serves as the root of a subtree containing all larger simplices in the

Algorithm 1: Proposed feature detector

```
input  : grayscale image $f$
output: local feature set $F$

1  $g \leftarrow \|\nabla f\| / \max\{\|\nabla f\|\}$                          ▷ normalized gradient
2  $E \leftarrow$ Canny($g$)                                                    ▷ edge detection
3  $P \leftarrow$ Sample($E$; uniform | non-uniform)                            ▷ edge sampling
4  $\mathcal{R} \leftarrow$ Triangulation($P$; Delaunay | constrained | regular)
5  $(\mathcal{K}, \rho) \leftarrow$ Complex($\mathcal{R}$; iso | aniso)         ▷ simplicial complex + sizes
6  $N \leftarrow$ Neighbor($\mathcal{K}'$)                                      ▷ neighborhood system
7  $F \leftarrow \emptyset$
8  foreach $\sigma_T \in \mathcal{K}'$ do                                        ▷ initialize each simplex
9      MakeSet($\sigma_T$)                                                       ▷ as an individual component
10     $\sigma_T.root \leftarrow \sigma_T$
11 foreach $\sigma_T \in \mathcal{K}'$ in descending order of $\rho_T$ do        ▷ current simplex
12     $\kappa_T \leftarrow$ Find($\sigma_T$)                                    ▷ current component $\kappa_T$
13     $r_T \leftarrow \kappa_T.root$
14     foreach $\sigma_U \in N(\sigma_T)$ such that $\rho_U \geq \rho_T$ do      ▷ adjacent, processed
15         $\kappa_U \leftarrow$ Find($\sigma_U$)                                ▷ adjacent component $\kappa_U$
16         $r_U \leftarrow \kappa_U.root$
17         if $\kappa_T \neq \kappa_U$ then                                      ▷ if different components
18             if $|U| = 3 \wedge$ Strength($U$) $> \tau$ then
19                 $F \leftarrow F \cup r_U$                                     ▷ select $\kappa_U$ if triangle & strong
20             $r_T$.AddChild($r_U$)                                             ▷ add it below $\kappa_T$
21             $r_T.area \leftarrow r_T.area + r_U.area$                         ▷ merge areas
22             $\kappa_T \leftarrow$ Union($\kappa_T, \kappa_U$)                 ▷ merge disjoint sets
23             $\kappa_T.root \leftarrow r_T$
```



(a) WαSH (b) Constrained, NU (c) Anisotropic, NU

(d) Hessian Affine (e) MSER (f) Edge Foci

Figure 10: Detections on objects with simple texture. The number of features per image for all detectors is limited to around 20. WαSH: 23, Constrained: 20, Anisotropic: 21, Hessian-Affine: 21, MSER: 22, Edge Foci: 23 features. We use non-uniform sampling (NU) for our detector (constrained and anisotropic).

same component. We maintain a list of children for each simplex, using function AddChild, while all simplices are initially assumed to be leaves. The second is a standard disjoint set forest, with simplices pointing only to their parent, manipulated by the functions MakeSet, Find and Union [40].

The two structures are interconnected via pointer *root*. The second is used for efficiency, and the first is used to collect information. In particular, for each selected component, we collect its simplices via breadth-first search in the subtree of its root simplex and fit a patch to their convex hull, which is not shown in Algorithm 1. The complexity of constructing the component tree is quasi-linear in n, that is, linear for all practical purposes [36].

4.4. Qualitative results

In order to obtain a deeper understanding of the quality of detected features, we show in the sequel a number of examples that highlight different aspects of the proposed detector and visually compare to other detectors. Fig. 10, 11 show detected features on an image of the textureless dataset of [4], depicting three hand tools. All detectors are tuned to return approximately the same number of features per image in Fig. 10, while default settings are used in Fig. 11. WαSH [12] uses weighted α-shapes on a regular triangulation and uniform sampling.

Our detector captures well-delineated regions that cover all objects of interest. The anisotropic variant appears to better capture elongated structures compared to the constrained one. Both cover

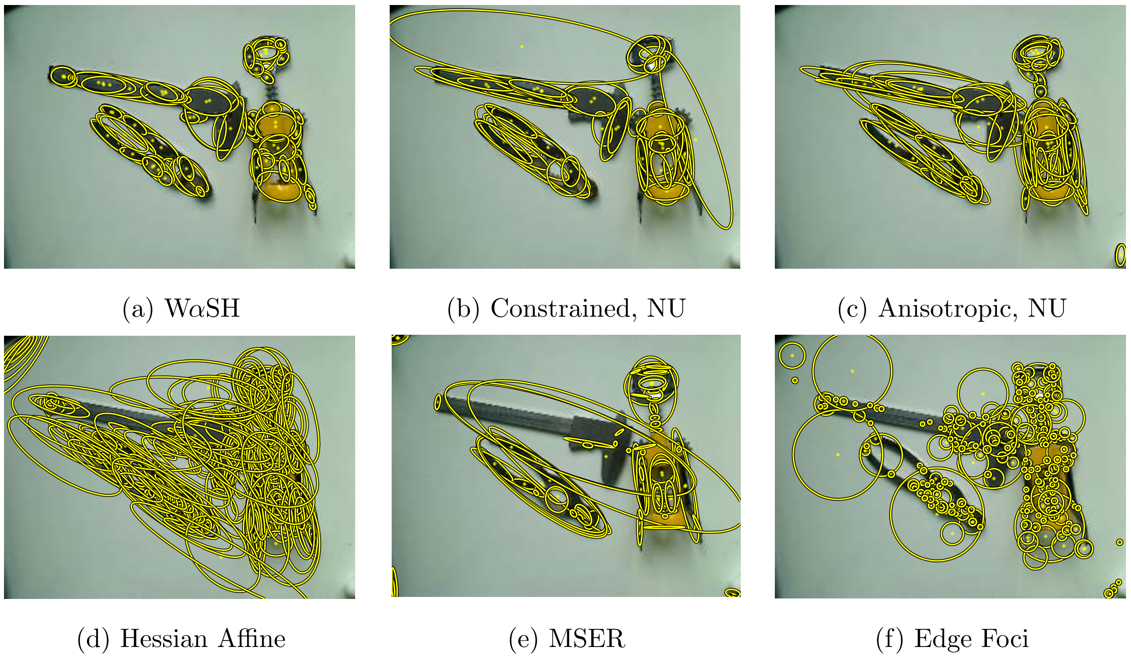


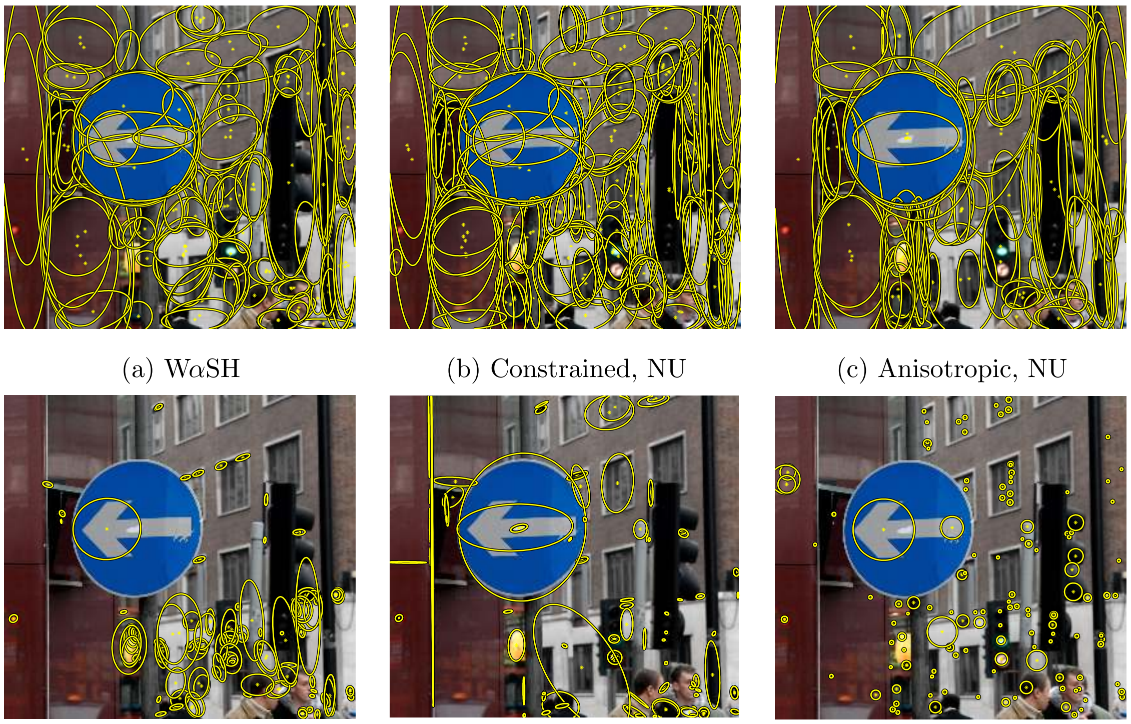
Figure 11: Detections on objects with simple texture, using default parameters. WαSH: 85, Constrained: 53, Anisotropic: 94, Hessian-Affine: 808, MSER: 88, Edge Foci: 284 features.

all essential parts of the objects when asked for more features, as shown in Fig. 11. The Hessian-Affine detector extracts multiple overlapping features at different scales, so fails to capture significant object details when the number of features is limited. This is corrected using the default settings, however the number of features increases dramatically in this case.

Fig. 12, 13 depict two more examples on an outdoor and an indoor scene. For fair comparison, detectors are again tuned towards a fixed number of features, different for each image. In Fig. 13, the anisotropic variant performs best, capturing most of the outstanding regions of the image like the almost circularly shaped color rings. Our detectors are the only ones to capture the repeating tile structure on the wall. MSER performs well, but also yields several noisy elongated regions along tile boundaries. Edge Foci are expected to perform very well on circular regions. However, the concentric circles of different radii are not detected here.

5. Experiments

We first investigate the pros and cons of all variants of the proposed detector and fine-tune parameters on a matching experiment using the standard dataset of Mikolajczyk *et al.* [19]. A comparison to the state-of-the-art follows on a matching and a retrieval experiment using the dataset of [19] and *Oxford 5K* [41], respectively. The first dataset involves matching across different viewpoint, rotation, zoom *etc.*, while performance is measured in terms of *repeatability* and *matching score*. To enhance



(a) WαSH (b) Constrained, NU (c) Anisotropic, NU

(d) Hessian Affine (e) MSER (f) Edge Foci

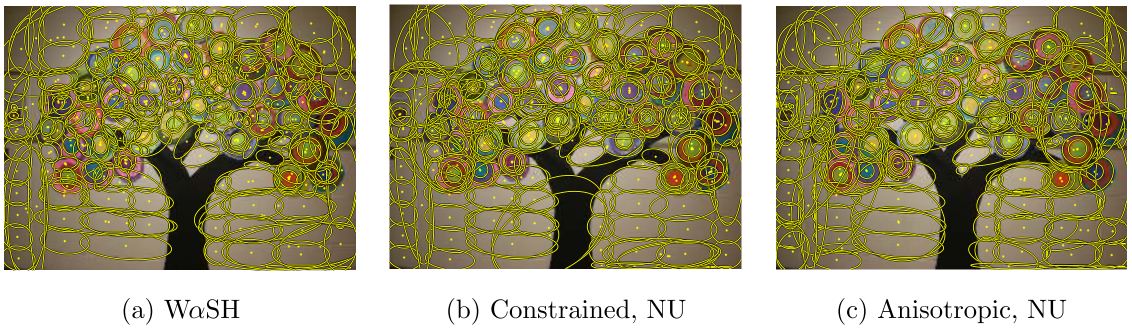
Figure 12: Detections in outdoor scene. Detector thresholds are tuned to extract around 130 features per image. W αSH: 122, Constrained: 136, Anisotropic: 114, Hessian-Affine: 137, MSER: 134, Edge Foci: 135 features.

objectiveness we use the VLBenchmarks framework, which is a recently introduced implementation by Lenc *et al.* [42]. The second dataset contains 5K images depicting buildings and 55 query images with ground-truth. It is used in a larger scale retrieval experiment, where performance is measured in terms of *mean average precision* (mAP).

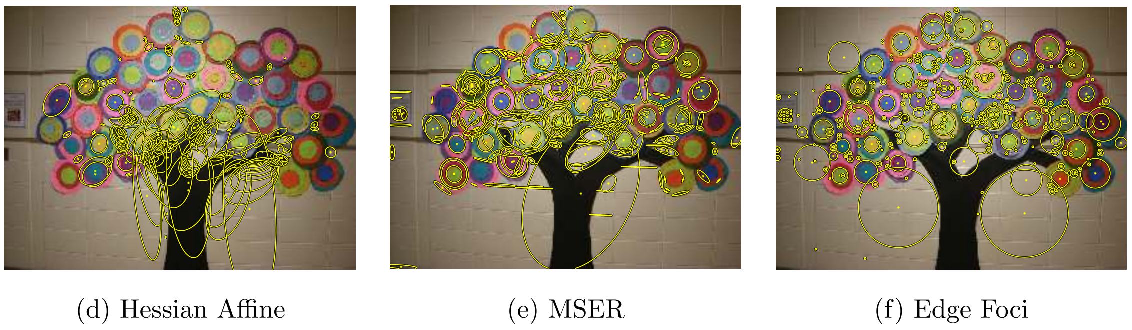
5.1. Strength measures

In this section we compare the strength measures of section 4.2 and select the most appropriate one. The selection is based upon computing the repeatability and matching score on a small-scale experiment. We extract approximately the same number of features per image by modifying the thresholds for every different strength measure.

We use six image sequences from the dataset of [19] to evaluate the effect of scale and rotation (*boat*), changes in viewpoint (*wall* and *graffiti*), illumination (*leuven*) and image blur (*bikes* and *trees*). Each image sequence consists of six images where matching of the first to the remaining five is increasingly difficult. Fig. 14 shows the measurements of our detector using all strength measures under varying effect of blur, scale, rotation, illumination and affine transformation. These measurements are obtained using weighted (isotropic) α-shapes on a regular triangulation with uniform sampling.



(a) WαSH (b) Constrained, NU (c) Anisotropic, NU



(d) Hessian Affine (e) MSER (f) Edge Foci

Figure 13: Detections in indoor scene. Detector thresholds are tuned to extract around 400 features per image WαSH: 413, Constrained: 380, Anisotropic: 405, Hessian-Affine: 425, MSER: 432, Edge Foci: 413 features. Observe the tiles at the lower part of the image that trigger our detector.

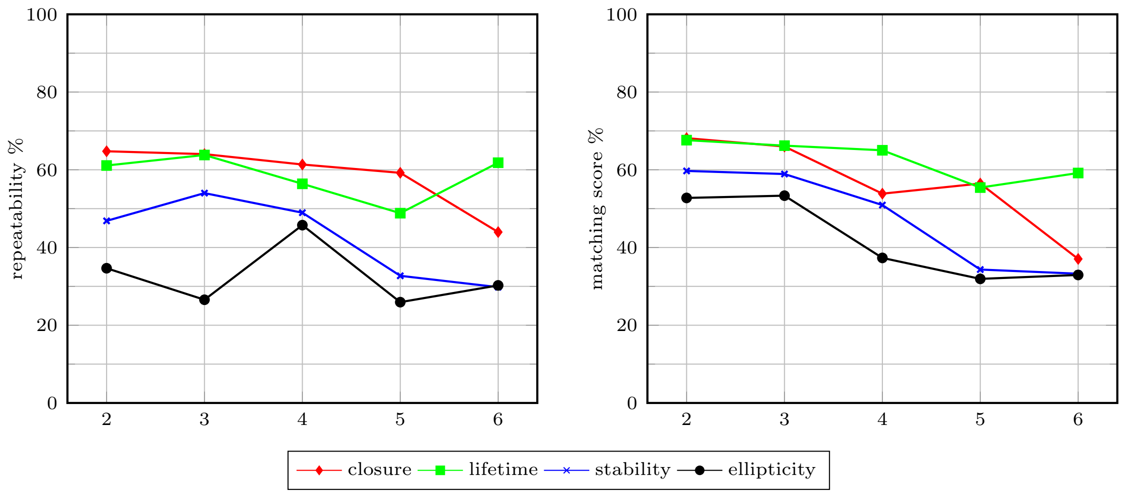


Figure 14: Evaluation of the proposed strength measures. Keeping the number of features approximately the same for each measure, we plot (a) repeatability and (b) matching score versus target image (2 to 6), with increasing difficulty from left to right. Measurements are averaged over the six image sequences.

While the number of features per image is kept approximately the same, we observe a high variation of the values depending on the different measures that is indicative of their role. Overall, the closure and lifetime measures exceed the other measures in both scores. For all the following experiments we adopt the *closure* measure.

5.2. Sampling, triangulation, and α-shapes

In the following we evaluate all α-shape variants, i.e. for each two sampling methods of section 3.1 (uniform, non-uniform), we consider four combinations of the triangulations discussed in section 3.2 (Delaunay, constrained Delaunay, regular) with the two α-shapes discussed in section 3.3 (weighted, anisotropic). The four combinations are given in Fig. 15. As in section 5.1, for each combination we measure the average repeatability and matching score on the same image sequences.

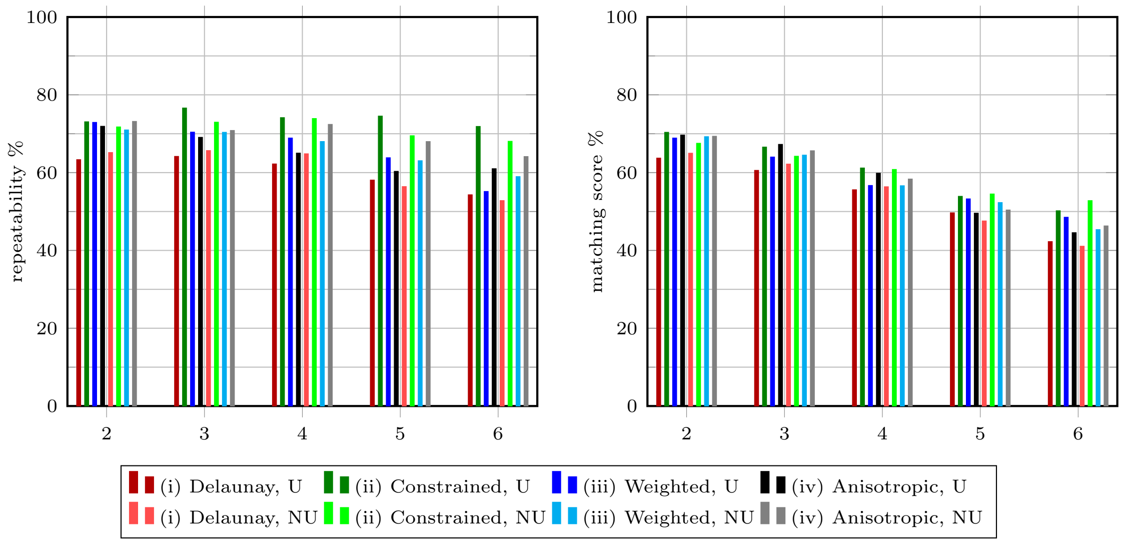
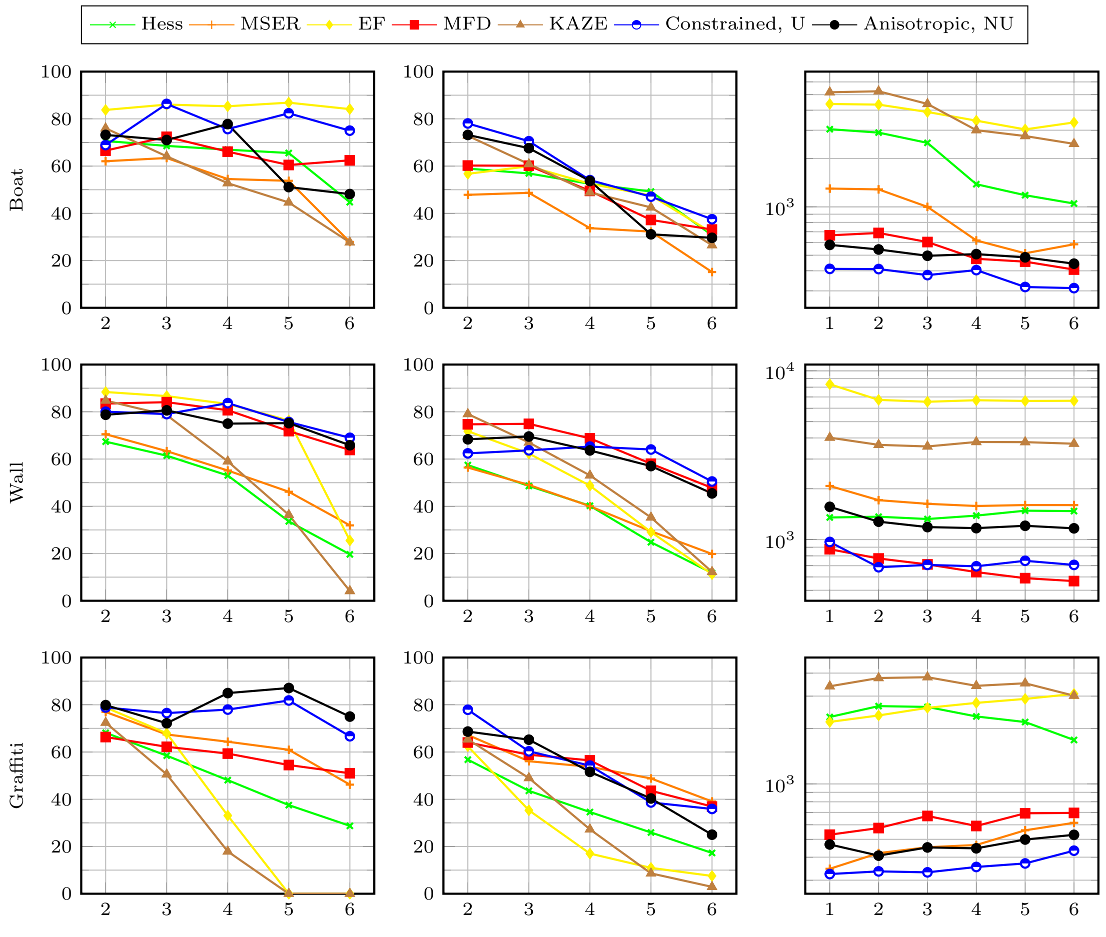


Figure 15: Repeatability and matching score for (i) α-shapes on Delaunay, (ii) α-shapes on constrained Delaunay, (iii) weighted (isotropic) α-shapes on regular, and (iv) weighted anisotropic α-shapes on regular triangulation. U: uniform sampling; NU: non-uniform sampling. Measurements are averaged over the six image sequences.

The results of the evaluation are depicted in Fig. 15. α-shapes on Delaunay triangulation with either uniform or non-uniform sampling are inferior to the rest. On the other hand, constrained Delaunay gives the best results followed by the anisotropically weighted α-shapes on regular triangulation, so we limit the following matching experiments to these two combinations.

The sampling step s is empirically set to 11 for the uniform sampling case. In certain cases, non-uniform sampling is outperformed by the uniform one, presumably because at these cases the fixed step of the latter is close to the optimal one. For the non-uniform sampling case, after a series of qualitative experiments on images of varying detail, we set the eccentricity upper bound of non-uniform sampling to $k = 3$ (see section 3.1.3). The initial size of neighborhood $\mathbf{p}$ of each sample point p is set to 11×11 pixels. Due to adaptation, the effect of this initial size on the final sampling density is minor, so we keep it relatively small to prevent increasing the computational cost of the sampling method.



(a) repeatability (b) matching score (c) #features

Figure 16: Comparison of our constrained and anisotropic detectors to state-of-the-art in sequences *boat, wall* and *graffiti*. #features: number of features detected per image. Hess: Hessian-affine. U: uniform; NU: non-uniform.

Focusing on the constrained and anisotropic cases of our detector, we compare to a number of state of the art methods on the matching experiment. In particular, we select the best performing detectors of [19], namely Hessian-Affine and MSER, along with three recent detectors, namely MFD [6], EF [5] and KAZE [7]. Features are extracted by the corresponding publicly available executables, which we have integrated in VLBenchmarks. For all detectors, default parameters are used.

The scores for all image sequences are depicted in Fig. 16-17 along with the number of detected features per image. Our detectors achieve a great balance between performance and number of features. They perform among the best in all cases, while keeping the number of features considerably low and remaining quite invariant to all examined transformations. The latter is mainly attributed to the proposed selection criterion applied to the α-filtration and—naturally—to the stability of image edges

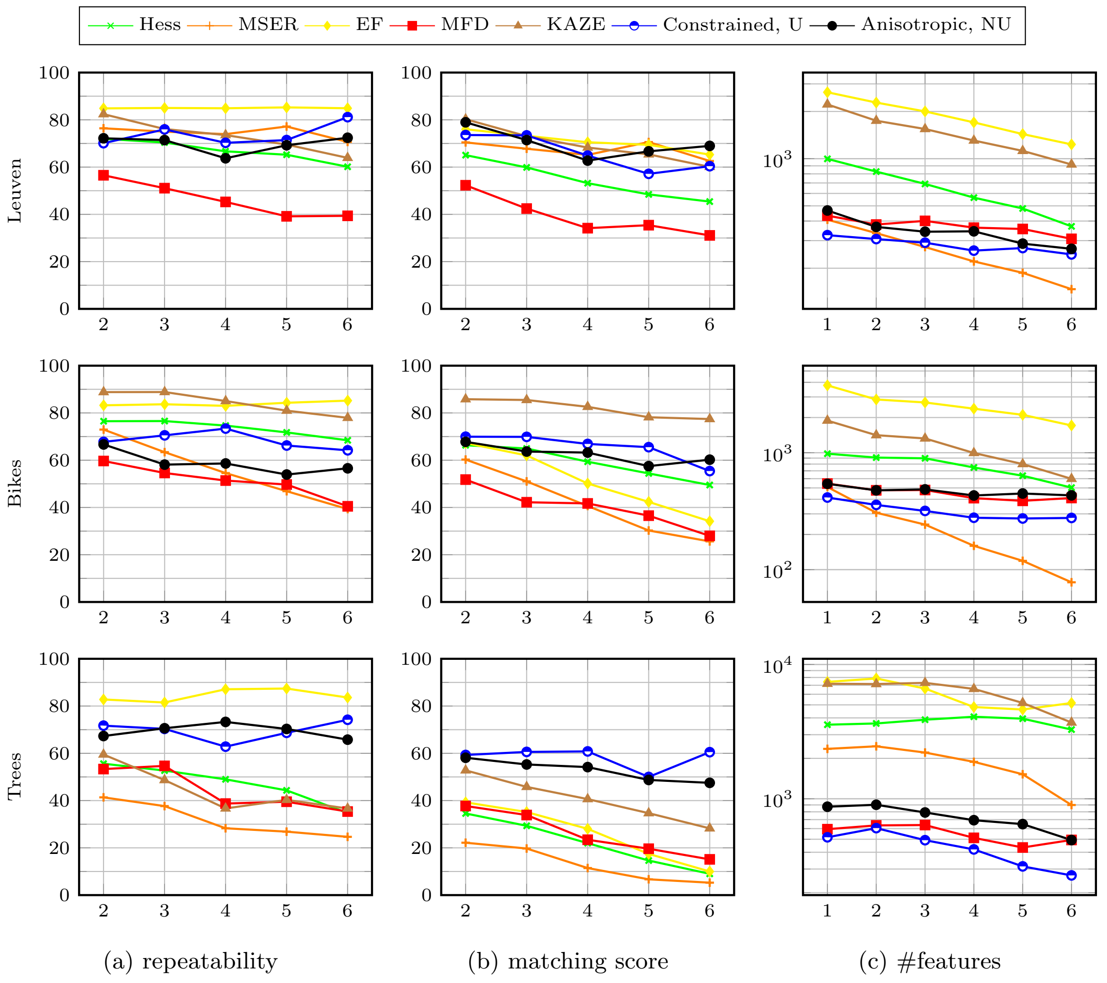


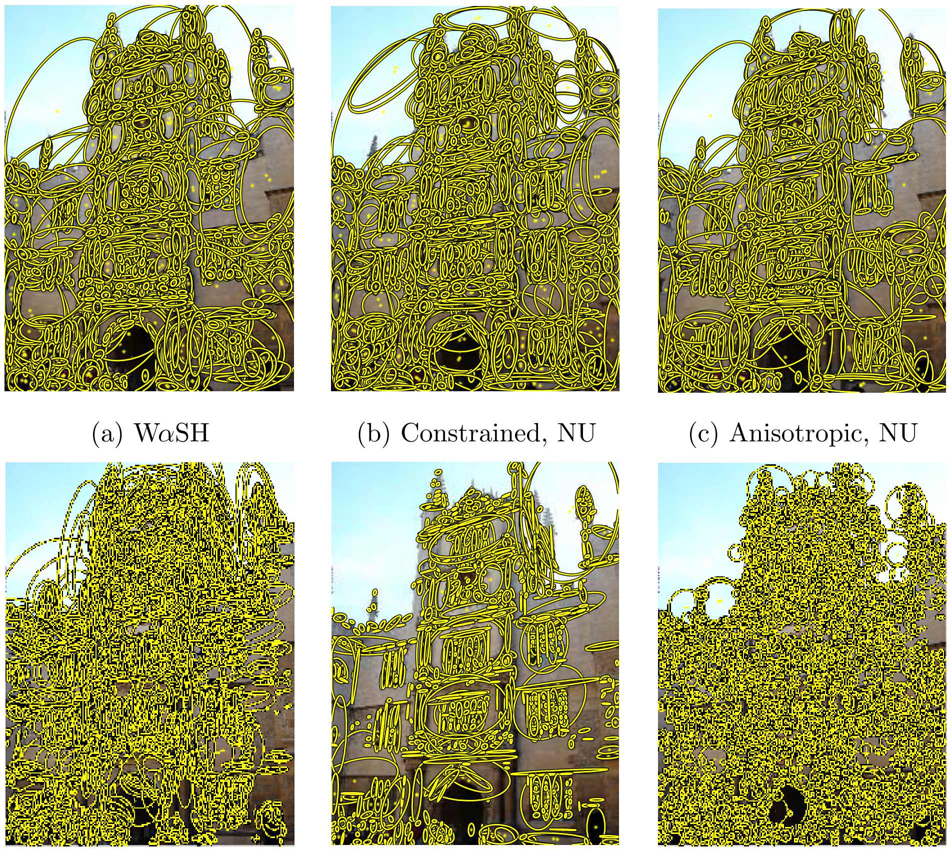
Figure 17: Comparison of our constrained and anisotropic detectors to state-of-the-art in sequences *leuven, bikes* and *trees*. #features: number of features detected per image. Hess: Hessian-affine. U: uniform; NU: non-uniform.

across scale.

5.4. Comparisons: image retrieval

In the following we evaluate the performance of our detectors and compare to state of the art by setting up an image retrieval experiment using the *Oxford 5K* dataset. Comparisons are performed against the same detectors as in the matching experiment of section 5.3 as well as the SIFT and SURF detectors which are commonly used.

Again, default parameters are used for all competitive detectors. This results in SURF having the lowest number of features (6.84×10^6 for the whole dataset). Considering that a small number of features is crucial for retrieval efficiency, we adjust the selection threshold of all our proposed detector variants so that they also give approximately 7×10^6 descriptors for the entire dataset. Though we are not reporting all relevant results, performance does not change much when choosing default



(a) WαSH (b) Constrained, NU (c) Anisotropic, NU

(d) Hessian Affine (e) MSER (f) Edge Foci

Figure 18: Detections on a sample image from Oxford 5K, using the parameters of the retrieval experiment. WαSH: 1453, Constrained: 1250, Anisotropic: 1315, Hessian-Affine: 6921, MSER: 1229, Edge Foci: 4294 features.

parameters instead, and in many cases it is higher, at the expense of more features. Fig. 18 shows example responses on an *Oxford 5K* image, using the settings of the retrieval experiment.

We extract SIFT descriptors for all detectors except SURF and KAZE, for which we use the SURF descriptor. This is a natural choice for SURF, while KAZE performs better with SURF descriptors than with SIFT. We build two visual codebooks of different sizes for each detector, namely 50K and 200K, by clustering descriptors on a sample of the dataset using *approximate k-means* [41]. We use the *bag-of-words* (BoW) model with *tf-idf* weighting for representation, an *inverted file* for indexing, and *fast spatial matching* (FastSM) [41] for spatial verification. BoW histograms are matched using histogram intersection following ℓ_1 normalization, while for verification we set the minimum number of inliers to 7. The evaluation metric is *mean Average Precision* (mAP).

Tables 1 and 2 provide detailed statistics including the total number of features, the average detection time per image, inverted file size, average query time and mAP measurements for all detectors, using the 200K and 50K visual codebooks respectively. The number of detected features and the

Detector		Features ($\times 10^6$)	Detection time (s)	Inverted file (MB)	BoW query (s)	FastSM query (s)	BoW (mAP)	FastSM (mAP)
HessAff		29.02	6.54	128.8	1.61	6.10	0.578	0.608
MSER		13.33	**0.40**	78.8	0.88	2.20	0.568	0.593
SIFT		11.13	5.24	84.0	0.95	5.29	0.494	0.516
SURF		**6.84**	0.43	53.5	**0.64**	3.45	0.575	0.591
EF		19.72	13.63	146.2	1.81	4.69	0.528	0.566
KAZE		13.82	6.59	99.6	1.67	1.91	0.487	0.541
MFD		7.64	2.98	58.4	0.68	**0.93**	0.600	0.600
uniform	Delaunay	7.54	1.38	52.1	0.88	1.11	0.580	0.577
	Constrained	7.17	1.57	50.3	0.84	1.01	0.588	0.590
	Weighted	6.85	2.01	**48.1**	0.83	1.16	0.595	0.594
	Anisotropic	7.00	3.90	48.6	0.85	1.08	**0.621**	**0.615**
non-uniform	Delaunay	7.09	2.89	50.4	0.84	0.96	0.592	0.592
	Constrained	7.27	3.52	50.9	0.86	0.99	0.610	0.597
	Weighted	7.71	3.98	53.6	0.88	1.07	0.557	0.560
	Anisotropic	7.47	6.60	50.4	0.89	1.08	0.602	0.594

Table 1: Retrieval results on *Oxford 5K* with a 200K codebook. The number of features refers to the entire dataset. Detection time is average per image. Query times are average per query.

detection time per image are the same in both cases, but they are repeated for easier reference and comparisons. The number of features used is critical, since it determines both the amount of memory used for the index and the query time especially for spatial verification, with FastSM being quadratic in the number of features.

The performance of all our proposed detector variants is at or above the state-of-the-art, despite using a low number of features, hence having a much lower memory footprint. In particular, we get roughly 1/4 of the features detected by Hessian-affine. The benefit in terms of query time is also considerable. A comparison of Tables 1, 2 shows clearly that increasing the size of the codebook boosts the performance of all detectors.

Using the anisotropically weighted α-shapes on a regular triangulation gives the best results, ex-

Detector		Features ($\times 10^6$)	Detection time (s)	Inverted file (MB)	BoW query (s)	FastSM query (s)	BoW (mAP)	FastSM (mAP)
HessAff		29.02	6.54	116.2	2.71	25.17	0.489	0.516
MSER		13.33	**0.40**	71.2	1.32	6.57	0.489	0.524
SIFT		11.13	5.24	75.9	1.51	8.35	0.422	0.446
SURF		**6.84**	0.43	47.8	**0.88**	3.75	0.466	0.497
EF		19.72	13.63	132.1	3.11	26.01	0.455	0.500
KAZE		13.82	6.59	89.4	2.62	7.30	0.403	0.464
MFD		7.64	2.98	51.9	0.94	2.45	0.531	0.540
uniform	Delaunay	7.54	1.38	47.0	1.15	1.60	0.521	0.537
	Constrained	7.17	1.57	45.3	1.20	1.27	0.541	0.553
	Weighted	6.85	2.01	**43.2**	1.05	1.37	0.544	0.566
	Anisotropic	7.00	3.90	43.6	1.12	1.32	**0.553**	**0.567**
non-uniform	Delaunay	7.09	2.89	45.4	1.08	1.43	0.514	0.526
	Constrained	7.27	3.52	45.9	1.11	1.45	0.551	**0.567**
	Weighted	7.71	3.98	48.3	1.13	1.51	0.476	0.465
	Anisotropic	7.47	6.60	45.4	1.13	**1.20**	0.532	0.551

Table 2: Retrieval results on *Oxford 5K* with a 50K codebook. The number of features refers to the entire dataset. Detection time is average per image. Query times are average per query.

ceeding the state-of-the-art, followed by the constrained and weighted cases, confirming the results of sections 5.2 and 5.3. The high performance of the anisotropic detector comes with the cost of higher detection time per image, which is due to the warping done per simplex (see section 3.3.2) and the extraction of the metric tensor describing the local shape (see section 3.1.3).

Non-uniform sampling in many cases slightly decreases performance. When combined with the weighted α-shapes, the performance drop is significant. The reason is that when the sampling step s increases along a straight image edge, so does the weight of the points by (1). This causes the circle of the weighted point to grow isotropically, and make points of neighboring edges disappear (see section 3.2.1).

Non-uniform sampling also adds a computational overhead that is not negligible. However, it

Detector		Features ($\times 10^6$)	Inverted file (MB)		BoW (mAP)		FastSM (mAP)	
			50K	100K	50K	100K	50K	100K
MFD		**2.59**	**18.8**	**20.4**	0.516	0.534	0.517	0.537
uniform	Delaunay	3.13	20.8	22.2	0.530	0.544	0.523	0.537
uniform	Constrained	3.06	20.3	21.7	0.537	**0.552**	0.524	0.548
uniform	Weighted	3.09	20.5	21.9	0.527	0.546	0.520	0.543
uniform	Anisotropic	3.27	21.4	22.8	0.532	**0.552**	**0.537**	**0.563**
non-uniform	Delaunay	3.19	21.6	23.0	0.522	0.532	0.530	0.538
non-uniform	Constrained	3.07	20.5	21.8	**0.539**	0.549	0.529	0.542
non-uniform	Weighted	2.96	20.0	21.4	0.469	0.486	0.460	0.476
non-uniform	Anisotropic	3.10	20.1	21.5	0.511	0.531	0.518	0.531

Table 3: Retrieval results comparing our detectors to MFD, with a lower number of detected features, targeting 3×10^6 features for the entire dataset. Here smaller 50K and 100K codebooks are used to avoid overfitting.

eliminates one parameter from the detector, which can significantly decrease the time needed for tuning. In small scale applications, where fine-tuning the sampling step is feasible, using uniform sampling can lead to better results. On the other hand, non-uniform sampling should be preferred for large-scale applications, where features are practically extracted without fine-tuning to minimize off-line preprocessing.

MFD performs similar to our detector and has approximately the same number of features, an observation that led us to an additional experiment where both detectors are tuned to produce a significantly smaller number of regions. Our aim is to test the ability of these detectors to scale up. By reducing the number of features detected, we need to also decrease the size of the codebook to prevent overfitting. In this setup, we create codebooks of 50K and 100K visual words. The performance of all proposed detectors is still high in all cases, especially with the 100K visual codebook, as shown in Table 3. These results verify our previous observations, as the anisotropically weighted α-shapes perform best.

6. Discussion

In this paper we have proposed a local feature detector based on edge-driven triangulation and geometrical representations that produces distinctive and stable image features. We extend our previous work [12] by proposing a non-uniform sampling method based on local image shape, requiring no user input. We also exploit local shape information at each sample by introducing the anisotropically weighted α-shapes. Finally, we propose and evaluate a number of different measures to select dominant components of the resulting α-filtration.

Our detector extracts a relatively small number of features that are highly distinctive and exhibit high image coverage. The resulting features are also remarkably tolerant to image transformations (scale, rotation and affine), lighting changes and blurring. These properties make our detector an ideal choice for large scale applications (*e.g.* large scale image retrieval or classification). Our experimental validation supports these observations: in most cases, our detector outperforms the state of the art in a number of matching and retrieval experiments.

References

[1] K. Mikolajczyk, C. Schmid, An affine invariant interest point detector, in: European Conference on Computer Vision (ECCV), Springer, 2002, pp. 128–142.

[2] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, Image and Vision Computing 22 (10) (2004) 761–767.

[3] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Computer Vision and Image Understanding (CVIU) 110 (2008) 346–359.

[4] F. Tombari, A. Franchi, L. D. Stefano, Bold features to detect texture-less objects, in: International Conference on Computer Vision (ICCV), 2013, pp. 1265–1272.

[5] C. Zitnick, K. Ramnath, Edge foci interest points, in: International Conference on Computer Vision (ICCV), 2011, pp. 359–366.

[6] Y. Avrithis, K. Rapantzikos, The medial feature detector: Stable regions from image boundaries, in: International Conference on Computer Vision (ICCV), 2011, pp. 1724–1731.

[7] P. F. Alcantarilla, A. Bartoli, A. J. Davison, Kaze features, in: European Conference on Computer Vision, Springer, 2012, pp. 214–227.

[8] H. Edelsbrunner, D. Kirkpatrick, R. Seidel, On the shape of a set of points in the plane, IEEE Transactions on Information Theory 29 (4) (1983) 551–559.

[9] A. Zomorodian, L. Guibas, P. Koehl, Geometric filtering of pairwise atomic interactions applied to the design of efficient statistical potentials., Computer-Aided Geometric Design 23 (6) (2006) 531–544.

[10] H. Edelsbrunner, Alpha shapes — a survey, in: Tessellations in the Sciences: Virtues, Techniques and Applications of Geometric Tilings, Springer Verlag, 2010.

[11] M. Teichmann, M. Capps, Surface reconstruction with anisotropic density-scaled alpha shapes, in: IEEE Visualization, IEEE, 1998, pp. 67–72.

[12] C. Varytimidis, K. Rapantzikos, Y. Avrithis, Wαsh: Weighted α-shapes for local feature detection, in: European Conference on Computer Vision (ECCV), Springer Berlin Heidelberg, Florence, Italy, 2012, pp. 788–801.

[13] P. Beaudet, Rotationally invariant image operators, in: International Joint Conference on Pattern Recognition, 1978, pp. 579–583.

[14] C. Harris, M. Stephens, A combined corner and edge detector, in: Alvey Vision Conference, 1988, pp. 147–151.

[15] T. Lindeberg, Feature detection with automatic scale selection, International Journal of Computer Vision (IJCV) 30 (2) (1998) 79–116.

[16] T. Lindeberg, J. Garding, Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure, Image and Vision Computing 15 (6) (1997) 415–434.

[17] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision (IJCV) 60 (2) (2004) 91–110.

[18] P. F. Alcantarilla, J. Nuevo, A. Bartoli, Fast explicit diffusion for accelerated features in nonlinear scale spaces, in: British Machine Vision Conf. (BMVC), BMVA Press, 2013, pp. 13.1–13.11.

[19] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Gool, A comparison of affine region detectors, International Journal of Computer Vision (IJCV) 65 (1) (2005) 43–72.

[20] E. Mair, G. D. Hager, D. Burschka, M. Suppa, G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test, in: European Conference on Computer Vision, Springer, 2010, pp. 183–196.

[21] S. Leutenegger, M. Chli, R. Siegwart, Brisk: Binary robust invariant scalable keypoints, in: International Conference on Computer Vision (ICCV), 2011, pp. 2548–2555.

[22] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: an efficient alternative to sift or surf, in: International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 2564–2571.

[23] T. Tuytelaars, L. Van Gool, Matching widely separated views based on affine invariant regions, International Journal of Computer Vision 59 (1) (2004) 61–85.

[24] K. Mikolajczyk, A. Zisserman, C. Schmid, Shape recognition with edge-based features, in: British Machine Vision Conference (BMVC), Vol. 2, 2003, pp. 779–788.

[25] K. Rapantzikos, Y. Avrithis, S. Kollias, Detecting regions from single scale edges, in: Intern. Workshop on Sign, Gesture and Activity (SGA), European Conference on Computer Vision (ECCV), Vol. 6553 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 298–311.

[26] H. Meine, U. Köthe, P. Stelldinger, A topological sampling theorem for robust boundary reconstruction and image segmentation, Discrete Applied Mathematics 157 (3) (2009) 524–541.

[27] F. Labelle, J. R. Shewchuk, Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation, in: Proceedings of the nineteenth annual symposium on Computational geometry, ACM, 2003, pp. 191–200.

[28] F. Cazals, J. Giesen, M. Pauly, A. Zomorodian, Conformal alpha shapes, in: Point-Based Graphics. Eurographics/IEEE VGTC Symposium Proc., IEEE, 2005, pp. 55–61.

[29] H. Edelsbrunner, Weighted Alpha Shapes, University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

[30] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 8 (6) (1986) 679–698.

[31] T. Lindeberg, J. Grding, Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure, Image and Vision Computing 15 (6) (1997) 415 – 434.

[32] A. Almansa, T. Lindeberg, Fingerprint enhancement by shape adaptation of scale-space operators with automatic scale selection, Image Processing, IEEE Transactions on 9 (12) (2000) 2027–2042.

[33] J. A. De Loera, J. Rambau, F. oSantos, Triangulations: Structures for algorithms and applications, Vol. 25, Springer, 2010.

[34] H. Edelsbrunner, N. R. Shah, Incremental topological flipping works for regular triangulations, Algorithmica 15 (3) (1996) 223–241.

[35] H. S. M. Coxeter, S. L. Greitzer, Geometry Revisited, The Mathematical Association of America, 1995.

[36] L. Najman, M. Couprie, Building the component tree in quasi-linear time, IEEE Transactions on Image Processing 15 (11) (2006) 3531–3539.

[37] M.-K. Hu, Visual pattern recognition by moment invariants, Information Theory, IRE Transactions on 8 (2) (1962) 179–187.

[38] J. Flusser, T. Suk, Pattern recognition by affine moment invariants, Pattern Recognition 26 (1) (1993) 167 – 174.

[39] P. L. Rosin, Measuring shape: Ellipticity, rectangularity, and triangularity, Machine Vision and Applications 14 (3) (2003) 172–184.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, MIT Press, 2009.

[41] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.

[42] K. Lenc, V. Gulshan, A. Vedaldi, VLBenchmarks, `http://www.vlfeat.org/benchmarks/`xsxs (2011).