University of Groningen

Vectorisation of sketches with shadows and shading using COSFIRE filters

Bonnici, Alexandra; Bugeja, Dorian; Azzopardi, George

Published in:
DocEng '18 Proceedings of the ACM Symposium on Document Engineering 2018

DOI:
[10.1145/3209280.3209525](https://doi.org/10.1145/3209280.3209525)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2018

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Bonnici, A., Bugeja, D., & Azzopardi, G. (2018). Vectorisation of sketches with shadows and shading using COSFIRE filters. In *DocEng '18 Proceedings of the ACM Symposium on Document Engineering 2018* [23] ACM Press Digital Library. <https://doi.org/10.1145/3209280.3209525>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Vectorisation of Sketches with Shadows and Shading using COSFIRE filters

Alexandra Bonnici
Faculty of Engineering
University of Malta, Malta
alexandra.bonnici@um.edu.mt

Dorian Bugeja
Faculty of ICT
University of Malta, Malta
bugejadorian@gmail.com

George Azzopardi
Faculty of Science and Engineering
University of Groningen, the
Netherlands
g.azzopardi@rug.nl

ABSTRACT

Engineering design makes use of freehand sketches to communicate ideas, allowing designers to externalise form concepts quickly and naturally. Such sketches serve as working documents which demonstrate the evolution of the design process. For the product design to progress, however, these sketches are often redrawn using computer-aided design tools to obtain virtual, interactive prototypes of the design. Although there are commercial software packages which extract the required information from freehand sketches, such packages typically do not handle the complexity of the sketched drawings, particularly when considering the visual cues that are introduced to the sketch to aid the human observer to interpret the sketch. In this paper, we tackle one such complexity, namely the use of shading and shadows which help portray spatial and depth information in the sketch. For this reason, we propose a vectorisation algorithm, based on trainable COSFIRE filters for the detection of junction points and subsequent tracing of line paths to create a topology graph as a representation of the sketched object form. The vectorisation algorithm is evaluated on 17 sketches containing different shading patterns and drawn by different sketchers specifically for this work. Using these sketches, we show that the vectorisation algorithm can handle drawings with straight or curved contours containing shadow cues, reducing the salient point error in the junction point location by 91% of that obtained by the off-the-shelf Harris-Stephen's corner detector while the overall vectorial representations of the sketch achieved an average F-score of 0.92 in comparison to the ground truth. The results demonstrate the effectiveness of the proposed approach.

CCS CONCEPTS

• **Computing methodologies** → **Image manipulation**; • **Applied computing** → **Graphics recognition and interpretation**; *Document preparation*;

KEYWORDS

Vectorisation, freehand sketches, junction detection, COSFIRE filters

Figure 1: Sketches of 3D objects include cues such as (1) shading, (2) different line-weights and (3) haloed-lines to illustrate depth. Sketch has been kindly provided by John McGowan¹.

ACM Reference Format:

Alexandra Bonnici, Dorian Bugeja, and George Azzopardi. 2018. Vectorisation of Sketches with Shadows and Shading using COSFIRE filters. In *DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3209280.3209525>

1 INTRODUCTION

Engineering design often starts with freehand sketches, drawn using real or digital ink. The sketch documents the ideas and concepts developed by the client and designer. By allowing an informal style, requiring little if any, design knowledge, sketching provides the means for quick externalisation and exploration of ideas [24]. The sketch is, therefore, a powerful means of communication between the client and the designer. Communication of three-dimensional form on flat paper can, however, give rise to ambiguities due to the missing depth plane. To counteract such ambiguities, sketches of 3D objects often include depth cues, such as the use of perspective or orthographic projections which provide the impression of depth [9, 16]. Projections alone, however, do not resolve all ambiguities as famously illustrated in the Necker drawings [16]. Sketches will, therefore, include additional cues such as shading and shadows, line-weights and haloed-lines among others to further illustrate 3D characteristics and spatial relationships between different parts of the drawing as shown in Figure 1 [9].

¹<http://www.coroflot.com/johnmcgowan/profile>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DocEng '18, August 28–31, 2018, Halifax, NS, Canada

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5769-2/18/08...\$15.00

<https://doi.org/10.1145/3209280.3209525>

Despite the strength of pen-and-paper sketching, the sketch serves only as an initial working document. Initial sketches are often re-drawn using computer-aided-design (CAD) tools to obtain blueprints required for prototyping or to benefit from virtual or augmented reality interactions with the product [24]. The use of specific CAD tools changes the nature of the document from one which allows collaboration from experts and novices and which is intended solely for human interpretation, to one which requires expertise to create but which can be interpreted by CAD software. Ideally, it would be possible to obtain the CAD representation directly from the sketched drawing without the need to re-draw the sketch, however, while humans are tolerant to many inaccuracies in the drawings, CAD tools require precision and accuracy [35]. Moreover, although cues enhance the human perception of the sketch, they also make the sketch more difficult for CAD tools to interpret. For example, in the presence of cues, it becomes necessary to distinguish between ink strokes that constitute the geometric form of the object and others which serve to enhance the interpretation of the object. Sketching with digital ink can circumvent some of the interpretation difficulties by monitoring the order with which the pen is in contact with the digital paper as well as the velocity and direction of the pen strokes. However, such information is not available when sketches are drawn using traditional pen-and-paper, and as yet, existing commercial software do not allow for direct input of freehand sketches.

The automated conversion of the sketch document to one which can be used by various CAD tools requires a vectorisation process which extracts line stroke information from the raster image and re-maps it to a vector format. The task of the vectorisation algorithm is, therefore, to extract stroke information about the geometric form of the object from all ink strokes within the sketch.

In this work we propose a vectorisation algorithm which builds models of intersection points to detect where sketched line strokes intersect in the drawing. Following this, a line tracking step is performed to determine the position of the form of the line connecting pairs of intersection points, while curve fitting is performed to smoothen the line-stroke variations typically associated with free-hand sketching. For the purpose of this work, we focus on drawings of objects which have smooth surfaces separated by edges which represent a discontinuity in the surface normal. This makes the proposed vectorisation algorithm suitable for the vectorisation of many man-made objects [6]. We also assume that edges in the drawing are drawn explicitly, that is, there are no object edges which are implied through the shading rather than sketched. This means that object edges are always darker than the shading strokes. Since paper-based drawings are still very much in use, the proposed vectorisation algorithm does not make any assumptions about the line-stroke ordering, and thus, the algorithm described in this paper does not discriminate between digital or paper-based sketches.

The rest of this paper is organised as follows: In Section 2 we give an account of related works described in the literature, in Section 3 we present our proposed vectorisation algorithm, in Section 4 we describe the evaluation protocol adopted in this work, and present the results obtained in Section 5. Finally, we draw conclusions in Section 6.

2 RELATED WORK

Converting sketches into a vector format requires the localisation of the lines in the image, discarding pixels that account for the line width or shading cues, keeping only those pixels necessary to retain the topology of the object represented in the sketch. Such pixels form the medial-axis of the sketched line strokes [18]. Skeletonisation algorithms are perhaps the most natural choice for this purpose [?]. Standard off-the-shelf thinning algorithms, however, result in spurious line segments at junction points or where the line stroke is not smooth, and thus, these algorithms rely heavily on post-processing the resulting skeletal lines through line fitting and beautification. Hilaire and Tombre [15] combine line fitting with edge-vertex graphs to make adjustments to line positions and correct junction displacements while Janssen and Vossepel [17] apply morphology-based corrections. Alternatively, Chiang [5] describes specialised thinning algorithms such as maximum inscribed circles to reduce the number of spurious lines associated with skeletonisation, thus shifting the focus of the vectorisation from the post-processing step to the skeletonisation step.

Line localisation is, however, not restricted to skeletonisation-based techniques. Edge detection provides one alternative, describing each line stroke as a pair of opposite edges or contours. A one-to-one mapping between contour fragments define the regular part of the line stroke, that is, the part of the line stroke not involved in a junction. The junction position is then found mathematically through line fitting [28]. While this approach works well for neat, machine-generated drawings, human sketches are sloppy, and thus lines do not usually intersect neatly at the junctions, resulting in poorly located junctions [29].

Rather than using contour pixels, Boatto et al. [3] and Monagan and Roosli [21] use horizontal and vertical run lengths along the line strokes, thus simplifying the problem of matching contour pixels, but also reducing the susceptibility of changes in line stroke widths along the line. Similar to contour-based approaches, regular and irregular runs identify the position of junction regions. However, instead of line fitting, Keyser and Breuel [19] fit geometric primitives over the regular runs. This approach resolves the potential mathematical issues of over-constraints but does not necessarily improve the junction localisation. The Hough transform provides another alternative to line fitting. Here, the task of line localisation becomes that of finding peaks within the accumulator array [13, 25, 32].

While the algorithms described thus far require visiting each pixel to determine its contribution to the line-stroke topology, tracking-based vectorisation algorithms take a different approach, using square [10, 22] or rectangular [8, 33] window to sample the line stroke. There, the vectorisation process focuses on heuristic tracking rules which are used to propagate the sampler on the line. In those approaches, rather than terminate the line at the junction point, attempts are made to propagate the sampler through the junction. Post-processing on all strokes is then required to locate the junctions. Since junction points play an essential role in the vectorisation process, more recent vectorisation works focus on the proper localisation of junction points, borrowing from computer vision approaches of finding corners in natural images, but adapting this to sketched drawings. Notably, Chen et al. [4] use a polar

Figure 2: Example of junctions where the adjacent contours do not intersect at common points.

curve to determine the number of branches at a potential junction point, hence establishing the junction order as well as locating the junction position. Noris et al. [23], Pham et al. [26], Favreau et al. [11] and Bessmeltsev and Solomon [2] characterise the topology of junctions typically found in sketches, describing the different possible points of contact between the central-lines of two strokes at every the junction. The works described above assume that the sketch image grey-levels follow a bimodal distribution, resulting in a simple segmentation of the image into the foreground strokes and the paper background. Images containing shading cues do not fit this image model, and binarisation may lead to misclassified pixels. Consequently, the detection of the stroke medial-axis points or junction locations may suffer from the shading strokes. Patterned shading strokes such as those typically found in hatched shading, exacerbate the problem since these strokes can be mistaken for object edges. There is, therefore the need for a vectorisation algorithm which can obtain line strokes from sketches containing shading and shadows.

3 METHOD

Our proposed vectorisation algorithm starts by locating the junctions present in the sketch. We note that sketched drawings are not accurate and thus, lines converging at a junction may not necessarily intersect at the same point as shown in Figure 2. Human observers are tolerant to such inaccuracies, and thus, the machine interpretation of the sketch should be equally tolerant to obtain the junction points expected by the human observer. For this reason, we look at computer vision models that mimic the human visual process. In particular, we propose the use of the *Combination of Shifted Filter Responses* (COSFIRE) [1] to model and locate the sketched junctions.

Through the localisation of the junctions, we may obtain the local orientations of the lines adjacent to junction. However, to fully vectorise the sketch, we also require the full topology of the object, that is, we need to determine the interconnectivity between junctions. Since we do not know the curvature of the sketched line strokes, and this is not necessarily evident at a junction point, we initially model the sketched line strokes as piece-wise linear

segments. This model allows us to use local measurements of the line stroke orientation to propagate a tracking element along the sketched stroke. Suppose that the line stroke consists of a perfectly straight line and that we can centre the tracking element somewhere along it, then propagating the tracking element in the direction defined by the local orientation of the line will retain the centering of the tracking element. However, curvature and varying stroke widths will cause the tracking element to displace from the ideal centre line. Such displacement will recursively worsen as the tracking element traces the sketched line stroke. Therefore, we need to adjust the position of the tracking element to ensure that this remains centred along the sketched line stroke. For this reason, we sample the region around the tracking element with a circular sampler. Since the circle is isomorphic and the tracking element lies on the centre-line of the sketched stroke, we can use this circle sampler to adjust the position of the tracking element such that it aligns with the centre of the sketched stroke.

We further note that sketched line strokes may be over-sketched, that is, the sketcher draws long line strokes as a series of short, overlapping strokes. Since human observers perceive such over-sketched strokes as single lines, then so too should the machine interpretation of the sketch. For this reason, we propose to use the COSFIRE algorithm for the detection of curvilinear segments. The result of the COSFIRE filtering process is enhancement of the object contours and suppression of shadows and shading. Since the COSFIRE is an oriented filter, we use this filter to enhance the object contours while simultaneously estimate the local stroke orientation.

Figure 3 illustrates the pipeline of the proposed vectorisation algorithm. The following sections describe each of the steps involved.

3.1 Junction detection

The detection of junctions starts by modelling the different types of junctions found in sketches. Figure 4 shows the junctions that characterise drawings of objects with up to a tetrahedral structure [36]. Such junctions and their reflected and rotated variants describe the majority of junctions present in engineering drawings [36]. Junction detection, therefore, requires processing a given sketch image with the modes of all junction prototypes, including their reflected and rotated variants. Such an approach would require a high computational effort. However, we note that we can describe higher-order junctions as the union of different variants of the L-junction and thus, junctions can be detected using only L-junction prototypes.

We use the COSFIRE algorithm [1] to configure COSFIRE filters selective for different L-junction apertures found in the junctions shown in Figure 4. The COSFIRE algorithm consists of two main steps, namely configuration and application. In the configuration stage, a given L-junction prototype is automatically analysed in order to determine a set of key-points and their properties around the vertex. In practice, these key-points are identified by first applying a bank (eight orientations denoted by θ and five scales denoted by λ) of (orientation-selective) Gabor filters to the given prototype. Then, the local maximum responses along a set of concentric circles with given radii denoted by ρ are chosen to be the key-points of the prototype. From each key-point i , a 4-tuple $(\lambda_i, \theta_i, \rho_i, \phi_i)$ is formed, where λ_i and θ_i are the parameters of the Gabor filter that achieves

Figure 3: The pipeline of the proposed vectorisation algorithm.

Figure 4: The junctions that characterise drawings of objects with up to a tetrahedral structure

the maximum response at that position, and (ρ_i, ϕ_i) are the polar coordinates with respect to the vertex of the prototype. We denote by $S = \{(\lambda_i, \theta_i, \rho_i, \phi_i) \mid i = 1 \dots n\}$ the set of 4-tuples representing the n identified key-points.

The application stage consists of three main steps; *convolve-blur-multiply*. The first two steps are applied for each tuple in the set S and the last step is performed only once at the end and it combines the intermediate feature maps generated for the concerned tuples. More specifically, for each tuple i , the Gabor filter with parameters λ_i and θ_i is convolved with the given input image, which can be of any size. In order to allow for some tolerance with respect to the preferred position of each key-point we blur the Gabor response maps with a nonlinear function. This non-linear blurring function uses a sliding window and for each window it takes the maximum of the element-wise multiplication between the Gabor responses in that window and the coefficients of a Gaussian function whose standard deviation σ' increases with increasing distance from the

COSFIRE filter support center: $\sigma' = \sigma_0 + \alpha \rho_i$. The parameters σ_0 and α , which we set to 0.67 and 0.1, respectively, control the tolerance that we give to each key-point. The blurred Gabor response map of each tuple i is then shifted by ρ_i pixels in the direction opposite to ϕ_i , so that the responses of all key-points meet at the support center of the COSFIRE filter. Finally, the shifted and blurred Gabor response maps, or feature maps for brevity, are combined by geometric mean function, essentially multiplication. The local maximum points in a COSFIRE output map indicate the positions at which local patterns are found to be similar to the prototype pattern that was used for the configuration of the concerned COSFIRE filter. For more technical details, we refer the reader to [1], where more detail is given also on how COSFIRE filters achieve tolerance to rotation, scale and reflection.

The blurring step in the COSFIRE algorithm allows the resulting filters to respond to corners, and thus to junctions, even if they are characterized with multiple strokes, or the adjacent contours are a bit deformed. The AND-type output function of COSFIRE filters ensures that they only react to local patterns consisting of at least two lines that intersect each other, and therefore they are insensitive to noise that may be due to shading and shadows, among others.

In ideal drawings, where all junction lines intersect at a single point, the responses of the COSFIRE filters which are selective for L-junctions with different apertures, would coincide and all junctions would be easily detected. In sketched drawings, however, lines at a junction do not necessarily intersect at the same point, and thus, the candidate junction points detected by the COSFIRE filters may not coincide. These candidate junction points will, however, be near each other and thus, a threshold on the Euclidean distance between pairs of all candidate junction points allows us to identify which of the candidate junction points belong to a single junction. The simplest way of obtaining the desired junction point would be to keep the junction with the largest COSFIRE filter response. However, since the COSFIRE filters are used to detect L junctions, such a

Figure 5: Sketched line strokes do not necessarily intersect at a single point thus, candidate junction points detected by the COSFIRE filter may not coincide. The detected junction points must be grouped into a single junction point.

response may be optimal for a particular L junction configuration but not the overall junction as illustrated in 5. A better approach would be to take the centroid of the detected points. However, such an approach is known to displace the location of the junction point [29]. We, therefore, configure a set of 11 COSFIRE filters that are selective for the complete junctions shown in Figure 4, and apply them to a small region around the detected junction points.

At this point, we are not only interested in the location of the junction but also in determining the type of junction as this is necessary to build the sketch drawing topography. To identify the type of junction, we may consider the junction prototype which gives the maximal COSFIRE response. While this approach should produce the desired junction type, we note that there may be instances where the scale of the junction is not consistent throughout the junction, for example, when a junction has one short branch due to proximity to another junction. We may also have instances when, due to sketching inaccuracies, junction branches do not match the proper junction prototype geometry. In such cases, it is possible to obtain a maximal response with junctions which have fewer branches than the sketched junction, hence misclassifying the junction type.

The k-nearest neighbour selection may resolve this problem. We form an 11-element feature vector V_p with the idealised responses of all 11 COSFIRE filters configured to be selective for the junctions shown in Figure 4. For each candidate junction we then apply the 11 full-junction-selective COSFIRE filters and form a feature vector of 11 elements by taking the maximum response of each filter across the cropped region of the given candidate junction. Then, the junction label of the vector with idealized responses that has the shortest Euclidean distance from the generated vector is assigned to the given candidate junction. In this application we set the parameters of COSFIRE filters as follows: t_1 is set to 0.75 of the maximal response obtained by the Gabor filter bank, five concentric circles with radii $\rho \in [0, 4, 9, 15, 22]$ pixels sample the region around the junction point.

3.2 Path tracking

As discussed earlier, detecting junction points is not sufficient to create the vectorial representation of the sketched drawing, and it is necessary to create the topology of the sketch which shows how each junction connects to each other. We have also shown

that sketched contours may be embedded in shadows and may contain over-sketching. To form the topology of the sketch, we, therefore, need to emphasise the perceived contour of the sketched strokes, while suppressing shading strokes to simplify the localisation of the lines in the sketched images and hence link adjacent junctions. To achieve this, we adopt the bar-selective B-COSFIRE filter approach [34]. This filter replaces the Gabor filters with a pool of Difference-of-Gaussians (DoG) filters whose supports are linearly aligned. This approach is inspired by orientation-selective simple cells in the brain, which receive input from LGN cells with center-surround receptive fields. The DoG filters achieve the highest responses for bars whose stroke width and orientation match the standard deviations of the inner Gaussian functions, and their alignment, respectively. In order to detect line strokes at any orientation we apply a B-COSFIRE filter in rotation invariance mode, whose afferent DoG functions have standard deviations that correspond to the most common thickness of line strokes in the given images. Since the filter response is expected to be maximal at the line centre, non-maxima suppression thins the filtered response, while maintaining stroke connectivity. Moreover, besides computing a response to every location in an image the B-COSFIRE algorithm also gives the discretized orientation that roughly matches the orientation of the local contour.

With the thinned lines and their corresponding line orientations as well as the junction locations, we may now trace the line paths connecting adjacent junction points. A cautious path tracking would advance the pixel tracking element one pixel at a time along the line stroke. On the other hand, a sub-sampling approach advances the tracking element by some distance r along the line [8]. Sub-sampling has the advantage of reducing the number of sample points along the path but is typically used with caution since there is the possibility of missing junctions on the line stroke due to large sample sizes. However, in this case, junction positions are known, and thus, the sub-sampling approach is more advantageous than single pixel tracking, allowing us to model the sketched stroke as piecewise linear vectors.

If the line strokes are perfectly straight, the local line orientation is sufficient to locate the next sample point on the line. However, since the sketches may contain strokes having some curvature, whether intended or due to the freehand nature of the sketch, the local line orientation may only indicate where the next sample point may be. To ensure that the sample point is re-aligned with the line stroke, we centre a circular sampler of radius r on the sample point and look for a line intersection in an arc with central angle η around the local line orientation. The point of intersection of the sketched stroke and the circle circumference within this arc defines the location of the next sample point as illustrated in Figure 6. The circle sampler serves a dual purpose. Any junction point located within the sector defined by η indicates that the tracked path reached a junction point and tracking can terminate at this junction.

In this application, the B-COSFIRE filter was used with an angular resolution of $\frac{\pi}{12}$ and nine DoG filters, at displacements of $[-8, -6, -4, -2, 0, 2, 4, 6, 8]$ pixels from the filter support centre. The radius of the sampling circle used to sample the line paths was set to $r = 15$ pixels.

Figure 6: At sketched strokes which exhibit some curvature, propagating a tracking element in the direction of the local orientation results in a position p' which does not fall on the stroke. To re-align the tracking element with the line stroke, we place a sampling circle on the sample point. The line stroke is expected to intersect the circle circumference at a point p within an arc defined by the angle η around the local line orientation.

3.3 Building an adjacency graph

The outcome of the path tracking step is a piece-wise linear vector with initial and final points being junction points on the drawing. To form the drawing topology we therefore require all such vectors, hence determining all interconnections between junctions in the sketch. Thus, we randomly select a junction and randomly select a branch from that junction and proceed to track the line emerging from the selected branch until the tracking reaches the adjacent junction. We repeat the process until the tracking algorithm tracks all branches of all junctions. The topology graph consists of vertices representing each junction in the drawing and edges which represent the paths connecting the drawing junctions. In the ideal scenario, the junction detection step detects and correctly labels all junctions in the drawing, and likewise, the path following step extracts all the sketched line strokes to their full extent from some originating junction to some other destination junction. Thus, the topology graph will consist of closed cycles representing the faces of the sketched object.

However, the junction detection and path following steps may occasionally fail to detect junctions or track all the line path. Such failures can occur, among others, when: lines strokes do not intersect, but still imply a junction; when, due to the freehand nature of the sketch, the sketched junction is a poor representation of the ideal junction; or when the sketched strokes have insufficient contrast with the shaded regions. Thus, in creating the adjacency graph, we attempt to catch and correct potential errors in the drawing vector representation.

The first instance that the algorithm flags a potential case for a missed junction occurs during the path tracking step. Since the line strokes at a junction have different orientations, when the circle sampler attempts to trace the line stroke through a missed junction, the line stroke intersection with the sampler circumference occurs at an angle which is outside the expected η range. In such a case, path tracking terminates with an open-end line vector, and the

adjacency graph will have two vertices with missing edge links. To resolve such an error in the adjacency graph, we take any such open-ended line segments and model them with straight lines, inserting a junction at the intersection point of the straight line models, thereby inserting the missing vertex in the adjacency graph as illustrated in Figure 7.

Open-ended line-strokes may also occur when the line detection fails to detect the underlying line stroke. In this case, however, the circle sampler tracing the path does not intersect with any line stroke other than the previously tracked line. Thus, we increase the radius of the circle sampler to expand the search for a continuation of the line stroke. If the circle sampler locates another line segment within the sector defined by the central angle η , we join the two edge segments with a straight line. Subsequently, we update the topology graph with a new edge. We repeat this process until no further adjustments to the adjacency graph are possible.

3.4 Line fitting

The result obtained thus far is a piece-wise linear vector representation of the sketched drawing. Although this representation is usable with CAD software, curve fitting results in vectors of better quality. In this work, we choose three curve fitting techniques, namely fitting straight lines, circular arcs [27] or cubic Bezier curves [7]. In all three curve fitting styles, we keep the stroke end-points fixed, thus ensuring that all strokes at a given junction all terminate at the same junction after line fitting. Thus straight-line fitting merely fits a straight line through the two junction points. If the squared error between a sample point on the stroke and the straight line fit is larger than a threshold t_{se} the sample points deviate too much from the line fit, and thus, a higher order fitting is required. Similarly, we first attempt to model the strokes with a circular arc, and if this too results in a squared error larger than t_{se} , we adopt the Bezier fit.

4 EVALUATION METHODOLOGY

We evaluate the proposed vectorisation algorithm using 17 sketches. Six of these sketches are digital sketches, drawn using a Samsung tablet and stylus and the AutoDesk Sketch Book² pre-packaged with the tablet. The remaining 11 sketches are paper-based, drawn by different persons, on A4 paper using HB or B pencils and digitised using a standard mobile-phone camera. These sketches contain examples of the most commonly used junction types as well as junctions which have been sketched with varying edge lengths. Moreover, these sketches contain examples of shadows sketched with smooth strokes and more challenging examples where the pencil marks are clearly visible.

To quantify the performance of the vectorisation algorithm, we obtain ground-truth representations of these sketches by manually selecting and labelling the junctions in the sketch and by tracing over the strokes.

We measure the performance of the vectorisation using the appropriate evaluation protocols as described in the literature. Since the vectorisation relies on the location of the junction points, we start the evaluation by determining the salient point error, defined as

²<https://www.sketchbook.com/?locale=en>

Figure 7: Recovering a missed junction. In (a) the sketched stroke is traced in a counter clock-wise direction. As the tracking element reaches the point marked A, the line stroke intersects the circle sampler beyond the arc defined by the central angle α . Thus, path tracking terminates at A. Likewise, in (b) tracking in a clockwise direction sees the path tracking terminate at point B. The local line orientations at A and at B are used to model the line fragments as straight lines, such that their intersection is used as an approximation of the missing junction point.

$$E = \frac{1}{2J_r} \left(\sum_{i \in J_r} \min_{j \in J_t} D(i, j) + \sum_{j \in J_t} \min_{i \in J_r} D(i, j) \right) \quad (1)$$

where J_r is the set of manually labelled reference points, J_t is the set of junction points obtained through the proposed algorithm and $D(i, j)$ is the Euclidean distance between a pair of junctions i and j [20]. If the proposed algorithm detects all the junctions in the sketch, then $|J_r| = |J_t|$ and the salient point error metric reflects the spatial distance between the detected and reference junction points. If, however, the junction detection algorithm introduces false junctions or misses junctions, then the salient point error metric will be further penalised by the mismatch between the reference and detected junctions [20].

To assess the goodness of fit of the resulting vectors, Wenyin and Dori [37] proposed an assessment protocol based on the principle that the detected vector should overlap with the sketched line strokes. The vectorisation evaluation protocol, therefore, measures the intersection of the vector representation with the image strokes. However, the underlying assumption of this evaluation protocol is that the drawings vectorised are accurate drawings, such as printed blueprints. In such drawings, it is desirable that vectors obtained from the vectorisation algorithms match as closely as possible the underlying image strokes. When drawings are of freehand sketches, however, there is an inherent degree of inaccuracy in the sketch which is not necessarily desirable in the resulting vectors. Thus, representing an unintentionally curved stroke in the drawing line with a straight line vector is acceptable in this context, even though the vector only partially overlaps the image stroke. Such an occurrence results in a substantial penalty on the goodness of fit measure which, however, does not reflect the human-perceived goodness of fit of the vector. For this reason, we borrow from the contour evaluation performance measurement described in [12]. The principle of this measurement is similar to the Wenyin and Dori protocol, in that the evaluation compares the recovered vector to the ground truth and determines whether the detected vector intersects with the ground truth. The difference between this approach and the Wenyin and Dori protocol is the definition of an $n \times n$ window around the ground-truth lines, considering the detected vector as overlapping with the ground truth lines if it falls anywhere within the $n \times n$ window. Thus, this protocol accounts for the displacement caused due to curve fitting with the tolerance window, and the measures are a better reflection of the localisation of the vectors. Precision and recall measures are used to quantify the vector localisation.

5 RESULTS

Figure 8 gives the sketches used for the evaluation with the vectors obtained from the proposed vectorisation algorithm superimposed on the sketch. The sketches also show the manually labelled junctions for evaluation of the salient point error. For comparison, Figure 9 shows the results obtained for three of these sketches using Adobe's LiveTrace³, which is among the state of the art commercial editing tools. From these sketches, we may note that LiveTrace does not deal well with sketches containing shadow strokes. On the other hand, the proposed algorithm detects only the object edges and disregards the shadow strokes.

In Table 1 we present the salient point errors obtained for these sketches. To our knowledge, algorithms which detect junction points from line drawings with shadows are not available for direct comparison. Thus, we turn to corner detection algorithms which are designed to locate corners in images which contain variations in intensity. For this purpose, we compare the salient point errors of the proposed algorithm with those obtained by three standard, off-the-shelf corner detection algorithms, namely the Harris-Stephens' corner detector [14], the minimum eigen-value algorithm [31] and the FAST algorithm [30]. The tabulated results reinforce our observation that the junction locations obtained by our vectorisation algorithm are close to the manually labelled junctions. The sketches

³<https://goo.gl/Fo16gq>

Figure 8: Sketches used for evaluation with the proposed vectorisation results superimposed on the sketch. In these sketches, the red circle markers indicate the detected junction points while the green cross markers indicate the manually marked junctions, used as ground-truth junction points.

have stroke widths of 5.75 ± 2.28 pixels whereas the salient point error is 3.79 ± 1.78 pixels and hence we may expect detected junctions to lie within the stroke boundaries. The Harris-Stephens', minimum eigen-value and FAST corner detectors, on the other hand, detect a larger number of corners due to the shaded regions and thus result in a larger salient point errors.

Table 2 provides the precision, recall and F-score values obtained from the junction type classification for the first six sketches shown in Figure 8. This Table compares the values obtained when using the k-nearest neighbour classifier with the maximal COSFIRE response. From this Table, we note that as anticipated, the k-nearest neighbour classification outperforms the maximal COSFIRE response. This happens because the misaligned sketched strokes combined with the shadow strokes leads to maximal responses at prototypes which do not correspond to the underlying junction. The k-nearest neighbour classifier reduces the incidence of such misclassified

junctions. There remain however instances such as the W-junction highlighted in Figure 10 when the shadow stroke aligns in such a way that it aids the support of the wrong junction type. Such a misclassification in the junction type will result in an initial search for the line path in an incorrect direction. Since the image does not sustain the propagation of the path in the wrong direction, path tracking along this path will not progress beyond the junction point. The path tracking and topology graph will, therefore, provide the possibility to correct for such misclassification error.

Table 3 gives the F-scores obtained from the contour evaluation. Here, we note that the vectorisation algorithm scored high F-scores for the majority of the drawings. The lower scores obtained for Sketches 7, 8, 9, 10 and 11 are a consequence of the sketches drawn without rulers or other aids. The effect is felt more strongly in these sketches because they contain longer sketched strokes, in part due to longer object edges and in part because these sketches are about

Table 1: The salient point error obtained by the proposed junction detector. For comparison, the salient point errors obtained by the Harris-Stephens' corner detector are also reported.

Sketch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Proposed junction detector	2.14	1.87	2.36	2.86	2.34	2.49	3.15	5.25	3.46	3.28	1.64	7.89	4.66	6.06	6.02	5.33	3.71
Harris-Stephens' detector	22.22	12.60	9.62	15.12	7.78	8.83	50.71	25.84	40.93	17.86	28.47	106.67	66.07	19.87	161.17	22.92	45.59
minimum eigen-value algorithm	37.65	52.16	128.34	36.68	23.62	22.95	36.50	24.20	67.07	43.08	45.95	42.7	49.99	42.42	43.88	42.85	43.15
FAST algorithm	49.90	77.14	141.08	33.42	24.03	22.99	22.24	48.47	29.12	70.15	31.88	43.67	41.60	42.42	31.98	26.12	38.86

Figure 9: The results obtained from Adobe's LiveTrace on Sketches 4, 5 and 13.**Table 2: Evaluation of the junction type classification for Sketches 1 - 6, comparing the classification using the k-nearest neighbour (kNN) classification with the maximal COSFIRE response (Max)**

Sketch	Precision		Recall		F-score	
	kNN	Max	kNN	Max	kNN	Max
1	1.00	0.33	1.00	0.05	1.00	0.09
2	1.00	0.45	1.00	0.33	1.00	0.35
3	0.85	0.45	0.94	0.63	0.87	0.44
4	0.97	0.25	0.97	0.11	0.97	0.15
5	0.97	0.36	0.96	0.62	0.96	0.34
6	0.89	0.37	0.90	0.59	0.89	0.32

three to four times larger than the other drawings. When sketching long strokes quickly, it is common to do so using a quick arm or wrist movement resulting in curvature [9]. In our vectorisation, these strokes have been modelled as straight lines but in doing so, the resulting vectors are placed outside the 5×5 pixel window the contour evaluation algorithm uses as a tolerance region. Hence, the number of true positives decrease while at the same time the number of false positives increase, lowering both precision and recall values. Visual inspection of the vectors however does not show these to be erroneously placed with respect to the drawing. Note that decreasing the maximum squared Euclidean distance acceptable for line fitting will model these strokes as splines which

Figure 10: The W-junction highlighted in this sketch is incorrectly labelled as having the edge configuration represented in red. This occurs because one of the sketched line strokes deviates considerably from the junction prototype that best describes this junction, illustrated here in green. Moreover, the shading strokes provide enough support for the wrong junction to gain a larger COSFIRE support than the correct junction.

increases the F-scores of these sketches to 0.997, 0.997, 0.967, 0.991, 0.999 and 0.993, respectively.

6 SUMMARY AND CONCLUSIONS

The vectorisation algorithm presented in this paper applies brain-inspired COSFIRE filters to the detection of junctions and lines from sketches, with a specific focus on sketches in which the edge strokes are embedded in shadows and shading. The algorithm is based on the detection of junction points, line enhancement and local orientation measurements, path following and the construction of a topology graph, followed by line fitting. The results obtained show that the junction points located are close to the intended junction points in the drawing. This is demonstrated by the low salient point error. Moreover, the vectors obtained are well placed on the line strokes as demonstrated by the large F-score values obtained for the test drawings.

The algorithms that we propose allow for the automated extraction of line vectors, hence reducing the burden for designers in redrawing the sketches using CAD software. It also makes it easier for non-professional users to have access to CAD tools and their applications in model building and virtual prototyping. Thus, the vectorisation algorithm proposed here would sit well within a suite of tools that would allow users to interact with the drawing, making

Table 3: The recall, precision F-score and accuracy measures obtained from the evaluation of the line vectors with the proposed algorithm.

Sketch	Recall	Precision	F-score	Accuracy
1	0.9997	1	0.9997	1
2	0.9993	1	0.9997	1
3	0.9665	0.9665	0.9665	0.9997
4	0.9937	0.9891	0.9914	0.9999
5	0.9995	0.9995	0.9995	1
6	0.9942	0.9909	0.9926	0.9999
7	0.8284	0.8014	0.8417	0.999
8	0.7948	0.7961	0.7955	0.9984
9	0.6226	0.5932	0.6076	0.9983
10	0.8207	0.8212	0.8209	0.9992
11	0.7335	0.7462	0.7498	0.9975
12	0.9735	0.9694	0.9744	0.9997
13	0.9989	0.9966	0.9977	1
14	0.9987	0.9966	0.9977	1
15	1	0.9996	0.9998	1
16	0.9942	0.9942	0.9942	0.9999
17	0.9967	0.9828	0.9897	0.9998

modifications either the raster or vector drawing according to preference or ease of use. This would require synchronisation between the different versions of the sketched and vector documents.

In future, we look towards extending the scope of the proposed vectorisation algorithm to include a wider variety of stylistic sketching, such as the use of cross-hatching techniques used to sketch shadows in ink-based drawings. Here the shadow strokes would have the same intensity as the edge strokes and therefore, require the inhibition of the cross-hatch patterns to allow for edge localisation and tracking.

REFERENCES

- [1] G. Azzopardi and N. Petkov. 2013. Trainable COSFIRE Filters for Keypoint Detection and Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 2 (Feb. 2013), 490–503.
- [2] Mikhail Bessmeltsev and Justin Solomon. [n. d.]. Vectorization of Line Drawings via PolyVector Fields. ([n. d.]). arXiv:1801.01922v1
- [3] L. Boatto, V. Consorti, M. Del Buono, S. Di Zenzo, V. Eramo, A. Esposito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci. 1992. An interpretation system for land register maps. *Computer* 25, 7 (July 1992), 25–33. <https://doi.org/10.1109/2.144437>
- [4] JiaZhou Chen, Qi Lei, YongWei Miao, and QunSheng Peng. 2015. Vectorization of line drawing image based on junction analysis. *Science China Information Sciences* 58, 7 (July 2015), 1.
- [5] J. Y. Chiang. 1995. A new approach for binary line image vectorization. In *Proc. Man and Cybernetics 1995 IEEE Int. Conf. Systems Intelligent Systems for the 21st Century*, Vol. 2. 1489–1494 vol.2. <https://doi.org/10.1109/ICSMC.1995.537983>
- [6] M. Cooper. 2008. *A Rich Labeling Scheme for Curved Objects*.
- [7] F Rogers David and J Alan Adams. 1990. Mathematical elements for computer graphics. *McGraw-Hill International* (1990).
- [8] Dov Dori and Wenyin Liu. 1999. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Transactions on pattern analysis and machine intelligence* 21, 3 (1999), 202–215.
- [9] Koos Eissen and Roselien Steur. 2007. *Sketching: drawing techniques for product designers*. Bis.
- [10] A. A. El-Harby and G. F. Meyer. 1999. Automatic line extraction by the square scan algorithm. In *Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, Vol. 2. 832–835 vol.2.
- [11] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: A Global Approach to Line Drawing Vectorization. *ACM Trans. Graph.* 35, 4, Article 120 (July 2016), 10 pages.
- [12] C. Grigorescu, N. Petkov, and M. A. Westenberg. 2003. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing* 12, 7 (July 2003), 729–739.
- [13] R. F. C. Guerreiro and P. M. Q. Aguiar. 2012. Connectivity-Enforcing Hough Transform for the Robust Extraction of Line Segments. *IEEE Transactions on Image Processing* 21, 12 (Dec. 2012), 4819–4829. <https://doi.org/10.1109/TIP.2012.2202673>
- [14] Chris Harris and Mike Stephens. 1988. A combined corner and edge detector. In *Alvey vision conference*, Vol. 15. Citeseer, 10–5244.
- [15] X. Hilaire and K. Tombre. 2006. Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 6 (June 2006), 890–904. <https://doi.org/10.1109/TPAMI.2006.127>
- [16] Donald D Hoffman. 2000. *Visual intelligence: How we create what we see*. WW Norton & Company.
- [17] Rik DT Janssen and Albert M Vossepoel. 1997. Adaptive vectorization of line drawing images. *Computer vision and image understanding* 65, 1 (1997), 38–56.
- [18] Robert A Katz and Stephen M Pizer. 2003. Untangling the Blum medial axis transform. *International Journal of Computer Vision* 55, 2 (2003), 139–153.
- [19] Daniel Keyzers and Thomas M Breuel. 2005. Optimal line and arc detection on run-length representations. In *International Workshop on Graphics Recognition*. Springer, 369–380.
- [20] T. Lu, C. L. Tai, H. Yang, and S. Cai. 2009. A Novel Knowledge-Based System for Interpreting Complex Engineering Drawings: Theory, Representation, and Implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 8 (Aug. 2009), 1444–1457.
- [21] Gladys Monagan and Markus Roosli. 1993. Appropriate base representation using a run graph. In *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. IEEE, 623–626.
- [22] M. Nidelea and A. M. Alexei. 2012. Method of the Square; A new algorithm for image vectorization. In *2012 9th International Conference on Communications (COMM)*. 115–118. <https://doi.org/10.1109/ICComm.2012.6262618>
- [23] Gioacchino Noris, Alexander Hornung, Robert W. Sumner, Maryann Simmons, and Markus Gross. 2013. Topology-driven Vectorization of Clean Line Drawings. *ACM Trans. Graph.* 32, 1, Article 4 (Feb. 2013), 11 pages.
- [24] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103.
- [25] Clark F Olson. 1999. Constrained Hough transforms for curve detection. *Computer Vision and Image Understanding* 73, 3 (1999), 329–345.
- [26] The-Anh Pham, Mathieu Delalandre, Sabine Barrat, and Jean-Yves Ramel. 2014. Accurate junction detection and characterization in line-drawing images. *Pattern Recognition* 47, 1 (2014), 282 – 295.
- [27] Vaughan Pratt. 1987. Direct least-squares fitting of algebraic surfaces. In *ACM SIGGRAPH computer graphics*, Vol. 21. ACM, 145–152.
- [28] Jean-Yves Ramel, Nicole Vincent, and Hubert Emptoz. 1997. A coarse vectorisation as an initial representation for the understanding of line drawing images. In *International Workshop on Graphics Recognition*. Springer, 48–57.
- [29] L. Ros and F. Thomas. 2002. Overcoming superstrictness in line drawing interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 4 (April 2002), 456–466. <https://doi.org/10.1109/34.993554>
- [30] E. Rosten and T. Drummond. 2005. Fusing points and lines for high performance tracking. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. 1508–1515 Vol. 2.
- [31] Jianbo Shi and C. Tomasi. 1994. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 593–600.
- [32] Jiqiang Song and Michael R Lyu. 2005. A Hough transform based line recognition method utilizing both parameter space and image space. *Pattern recognition* 38, 4 (2005), 539–552.
- [33] Jiqiang Song, Feng Su, Chiew-Lan Tai, and Shijie Cai. 2002. An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 8 (Aug. 2002), 1048–1060.
- [34] N. Strisciuglio and N. Petkov. 2017. Delineation of line patterns in images using B-COSFIRE filters. In *Proc. Int. Conf. and Workshop Bioinspired Intelligence (IWOB)*. 1–6.
- [35] Karl Tombre, Christian Ah-Soon, Philippe Dosch, Gérald Masini, and Salvatore Tabbone. 1999. Stable and robust vectorization: How to make the right choices. In *International Workshop on Graphics Recognition*. Springer, 3–18.
- [36] P. A. C. Varley, R. R. Martin, and H. Suzuki. 2004. Making the Most of Using Depth Reasoning to Label Line Drawings of Engineering Objects. In *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications (SM'04)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 191–202.
- [37] Liu Wenyin and Dov Dori. 1997. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications* 9, 5-6 (1997), 240–250.