Structural Vectorization of Raster Images

Philip Buchanan
University of Canterbury
University Drive, Ilam 8041
Christchurch, New Zealand
philip.buchanan@
pg.canterbury.ac.nz

Michael Doggett
Lunds Universitet
Box 117, 221 00
Lund, Sweden
mike@cs.lth.se

R. Mukundan
University of Canterbury
University Drive, Ilam 8041
Christchurch, New Zealand
mukundan@canterbury.ac.nz

ABSTRACT

This paper presents a new automatic algorithm for extracting vector information from raster images. The algorithm extracts structural information from the lines that is formatted to allow easy processing and evaluation of the image structure. Vectorization results are comparable with commonly used algorithms, however the outlined method differs from prior work by providing information in a more accessible form. This algorithm provides topological information at the cost of visual fidelity. Properties such as line topology and width are important for image processing, including object decomposition, author recognition and line style modification.

Categories and Subject Descriptors

I.3.3 [**Picture/Image Generation**]: Line and curve generation

General Terms

Algorithms

Keywords

Vectorization; Image Structure; Skeletonization

1. INTRODUCTION

Traditionally, image analysis is performed on raster images based upon global or local features. However some types of algorithm such as style and stroke analysis [8] [9] perform better or must be performed on vector data. Vector data always contains a line topology made from line position and connectivity data, and may also include width, color, and border properties. While modern tools allow rapid drawing directly into vector formats, many artists and studios still use raster images for cartoon work. Additionally, older artwork only exists in raster format, which must be vectorized before it can be used. Storing image data in a vector format has the added benefit that it is highly efficient in comparison to the source image.

Current algorithms that vectorize images while preserving visual quality often do so at the cost of either stroke or topological information. This paper outlines a vectorization algorithm that extracts line data in a format that allows for easy access and use in further analysis. Figures 1, 3 and fig:structureComparison show the advantages over edge vectorization and morphological skeletonization respectively.

Our algorithm vectorizes an image in three main stages. A resolution independent gradient map is generated for the image, containing vectors orthogonal to the lines; the line

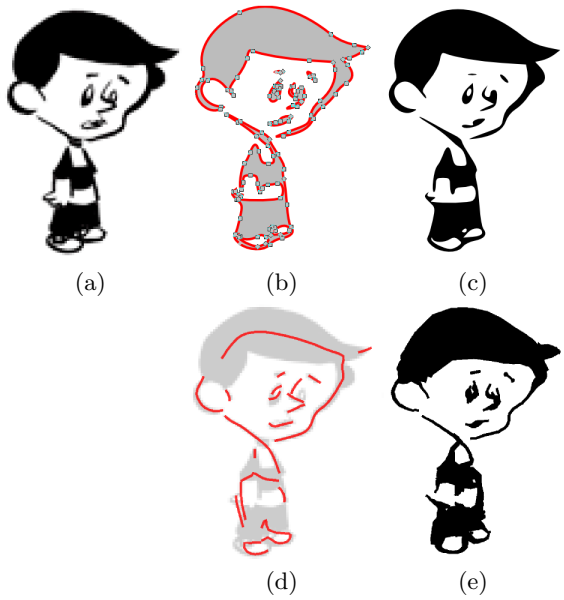
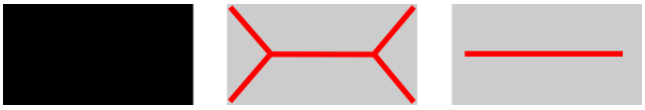


Figure 1: Comparison between Structural Vectorization and a polygon based vectorization technique [18], both taken from the same source image (a) [6]. Polygon based vectorization produces images with high visual fidelity (c), but results in complex topography (b). Structural vectorization extracts dominant strokes (d), and line width information. While not as visually accurate as polygon vectorization, strokes and widths are nevertheless able to represent the source image (e), and are of more use in image processing.

IVCNZ '12, November 26 - 28 2012, Dunedin, New Zealand



(a) (b) (c)

Figure 3: Morphological skletonization operators such as the medial axis transform provide a geometric decomposition (b) of a shape that even when thinned does not always represent the human recognised structure our algorithm extracts (c).

centers are found with subpixel accuracy by analysing cross sections aligned to the gradient field; and the vectors are created using a weighted nearest-neighbour algorithm to join the centres. Section 4 shows the output from this process and compares it to existing methods.

2. PRIOR RESEARCH

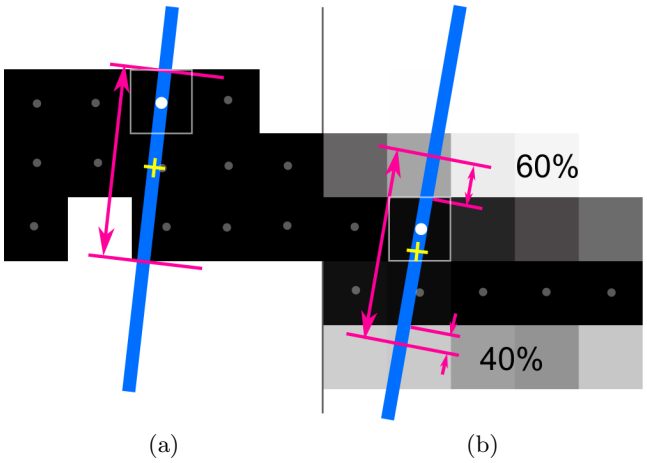
Vectorization is a common analysis problem, and many solutions exist. Two algorithms that extract line data are proposed by Elliman [7] and Dori et al. [4], however both are suited to technical drawings with straight lines and exhibit problems with irregular shapes. Research specific to cartoon drawings has recently gained a higher profile, with Chenga [15] and Zhang [19] releasing vectorization papers that deal with irregular shapes. Chenga et al. [15] provide the better algorithm due to their accurate stroke segmentation and image complexity reduction. However, both papers require even line widths and must be tuned for specific line profiles. Hand-drawn cartoon input rarely has even line widths, and so any vectorization algorithm must cope with width variation, as the structural vectorization algorithm presented here does.

Recent research released by Huang et al. [10] presents a stroke extraction algorithm that does not rely on even line widths. They provide a robust stroke extraction algorithm but unfortunately stop short of vectorization. Our focus upon cartoon imagery means that source imagery already has clearly defined strokes, and unless the range of input images is extended, stroke extraction is unnecessary.

Once an image has been reduced to black and white strokes, a common vectorization method is the Potrace algorithm by Selinger [18]. This method produces graphically accurate representations of an image by treating it as a series of geometric shapes. This is useful for preserving fidelity, but makes structural processing difficult due to the lack of line centre and width information.

In addition to stoke extraction and recognition, many morphological and topological skeletonization algorithms exist that produce outputs ranging from unconnected point clouds [5] to β-skeletons [2] that contain the topology in a connected graph. Our algorithm reaches a compromise that allows for disconnected elements but strives to join line vertices when possible.

Medial transforms are perhaps the most well established method for skeletonization, having been proposed in 1967 [3] and tweaked in various different ways up until the present [16] to solve problems such as the influence of surface noise on branching. Another method with the same result but a different approach is joining the centers of bi-tangent circles or maximal disks within a shape [1]. The medial transform



(a) (b)

Figure 4: Subpixel accuracy can be obtained by taking advantage of cues such as feathering. Image (a) shows centrepoint placement for a monochrome image, while (b) shows this extended to antialiased pixels. Grey points represent pixel centres, with the white dots and square outlines indicating the pixel being evaluated. The blue slice line has been placed based upon the image gradient from Equation 3

produces geometric skeletons, however as can be seen in Figure 3 even simple shapes can produce a skeleton that does not correspond logically to the underlying structure.

This problem arises even when different approaches are taken [14], while papers that retrieve a clean structural topology do so by limiting images to a specific domain such as handwriting recognition [11]. In addition, Lam, Lee & Suen found that most skeletonization algorithms do not store width or colour data [13].

Line topology and width are two of the most important properties when attempting image analysis on vector images. Access to these properties can help when analysing object composition and make it easier to decompose objects into sections. Line data can be used to change the drawing style of an image by modifying the brush stroke properties, and analysis of artistic style can be carried out by looking at properties such as line length and camber. Storing image data in a vector format is also highly efficient in comparison to the source image.

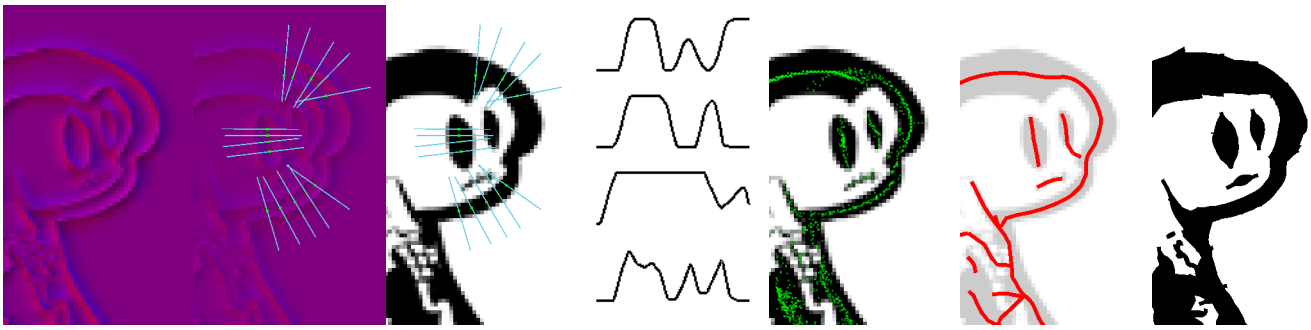
The method outlined in this paper is computationally expensive, but extracts line centres even if the line has an irregular profile or the image has unusual topology.

3. STRUCTURAL VECTORIZATION

The vectorization algorithm is composed of several stages as shown in Figure 2. It preserves structural information and is able to process complex images such as the one shown in Figure 1(a), where relevant lines may not be obvious.

The process begins by identifying line centers. Several recent vectorization algorithms proposed a thinning step, including Olsen et al. [17] who use erosion as a core step to identify line centers. However if lines within the image have different widths, using an erosion step can lead to distorted or entirely incorrect identification of line centres.

To avoid this issue, our algorithm finds midpoints between matching edges. Edges are considered to be sharp changes



(a) (b) (c) (d) (e) (f) (g)

Figure 2: Raster images are vectorised to preserve structural information about line centres and widths at the cost of visual fidelity. A vector field (a) is calculated for the source image, and used to slice the shortest path from each pixel to the nearest edge (b). When values are taken from the greyscale image, these slices (c) measure the value profile (d) and are subsequently used to place control points at the local maxima (e) which represents the line centre. Joining these with a nearest-neighbour algorithm creates a structural representation of the image (f) that together with line width information is enough to store a representation of the input image. (g)

in intensity, and for a line to be detected it must have two opposing edges. These are detected by creating slices based upon the gradient of the image at each pixel. The gradient at a point is calculated using the difference of intensity within a given radius, with the radius being varied across a range and results averaged to produce a scale independent gradient map. Intensities are also weighted based upon distance from the sample area, using a standard 2d gaussian kernel K:

$$K(x, y, \delta) = exp\left(-\frac{x^2}{2\delta^2} - \frac{y^2}{2\delta^2}\right) \quad (1)$$

where δ is the spread of the gaussian kernel.

This gaussian kernel is then convoluted with the image across a range of different δ and averaged:

$$I'_{(x,y)} = \frac{1}{l} \sum_{0<\delta<l} I_{(x,y)} \odot K_{((x-\frac{w}{2}),(y-\frac{h}{2}),\delta)}$$

$$\text{where} \qquad l = \frac{\sqrt{w^2+h^2}}{4} \quad (2)$$

where I is the original 2D image with width w and height h.

Using a large maximum sample area introduces low-frequency directional artifacts in the empty area surrounding an image, but provides the important tangent information at larger distances from lines; similarly, smaller areas limit the maximum detection width for thick lines in the image but result in better fidelity close to line borders. The ideal maximum size for our dataset was found to be a quarter of the image diagonal.

The gradient at a given point on the image, across the entire range of gaussian kernels, is generated by Equation 3. $\hat{G}$ gives a 2D vector at (x, y) orthogonal to the closest line or edge structure in the raster image:

$$\hat{G}(x,y) = \sum_{\substack{0<-\delta<l \\ -\delta<x'<\delta \\ -\delta<y'<\delta}} \overrightarrow{(x+x', y+y')} \cdot I'(x+x', y+y') \cdot K(x', y', \delta) \quad (3)$$

(a) (b) (c)

Figure 5: Disconnects can be caused in detected lines (b) by light pixel values, noise, or patterns in the source bitmap (a). Short pieces of line with co-linear ends are therefore joined so as to better represent the source image (c).

The kernel size is based on the image size, so that the resulting gradient field will preserve the sharp changes in gradient at edges, while still registering a gradient for flat colours that occur at a distance from edges. This 2D gradient field provides more information than the computationally faster single-convolution operators such as the Laplacian, Prewitt, Canny and Sobel. The gradient field is shown in Figure 2(a), with the x and y components of the vector field represented by the *red* and *blue* channels respectively. The resulting vector field makes it possible to determine the direction to the closest edge from any pixel and the tangent and normal vectors for a line before it is vectorized. This allows the centre of the line to be found with sub-pixel accuracy.

The centre of the line in the raster image is found by searching each pixel within the same colour block. The gradient map approximates the line normal, and is used to place a cross-sectional cut, or slice, across the line, as in Figures 2(c) and 4(a). This allows accurate measurement of the line width and centre at that point, as shown with the line profile in Figure 2(d).

In cases where the image antialiased and the profile does not change sharply between black to white, the relative in-

tensity values of the neighbouring pixels are used to adjust the centrepoint. For each non-white pixel at the edge of the line, the centrepoint is moved by $\frac{intensity}{2}$ units along the slice as shown in Figure 4(b).

A control point is then created at the centre of the line, and the process repeated for the next pixel. Figure 2(e) shows the result after all pixels have been processed, with green dots representing the control points. Important lines in an image tend to be isolated or wide and therefore receive the thickest control point clusters, while noise, detail and insignificant lines receive few points. These control points are joined using a nearest-neighbour algorithm that attempts to match points along a curve as shown in Equation 4,

$$\begin{aligned} a &= l_n - l_{n-10} \\ b &= v_x - l_n \\ l_{x+1} &= \sum_0^N \begin{cases} v_x & \text{if } (\hat{a} \cdot \hat{b} > -\frac{\sqrt{2}}{2}) \text{ and } (\|b\| < 1.5) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (4)$$

where l_x are the vertices in a line segment with n vertices, and v_x are a pointcloud of size N.

Any points remaining after the lines have been created are considered outliers and culled. Points lying within the same pixel are considered to be duplicate vertices and are also removed. Figure 2(f) shows the result of point joining and culling. Line widths at each vertex are calculated based upon the previously measured line profile and stored per vertex. The distance threshold value of 1.5 units was chosen because datapoints are spaced on average 1 unit apart in the tangent direction. Small differences can be caused due to antialiasing, but the total value of antialiased pixels will never exceed 1 and therefore the maximum distance a point can be dislodged is 0.5 units. The angle threshold value is not as easy to calculate, and the best value depends upon image complexity. 45° was found to work best across our sample dataset, as higher values produce disconnects at intersections and smaller values cause disconnects at corners. This value could be adjusted by the user if necessary.

Lines are smoothed and jitter removed by removing unnecessary control points. Reducing the number of control points also simplifies the data for processing after vectorization. This is performed on a point-by-point basis by iteratively removing control points until the new segment no longer approximates the correct shape sufficiently well. The error between the simplified line and the original line is measured using Equation 5,

$$\begin{aligned} A &= v(0) - v(x) \\ B &= v(x) - v(n) \\ t &= \frac{\|A\|}{\|A+B\|} \\ error &= \sum_{x=0}^{n} s(t) - v(x) \end{aligned} \quad (5)$$

where v is the set of vertices of size n and $s(t)$ is the section of line currently being simplified.

The simplification process continues until the error rises over a user-specified threshold, after which the end vertices are fixed and the process run on the next segment of the line.

In the case of thick lines or shapes, it is possible that two orthogonal centrelines are found and an extra culling step must be performed. All intersecting lines are compared for overlap and culled according to Equation 6,

(a) (b) (c)

Figure 7: The topology generated by our algorithm (b) differs from typical morphological algorithms that produce geometric centers (c). Note how the side of the table is represented by fewer lines in our model.

$$\begin{aligned} w_s &= \sum_{x=0}^{n} \frac{v_s(x)}{n} \\ w_l &= \sum_{x=w_s-t_l}^{w_s+t_l} \frac{v_l(x)}{2w_s} \\ \text{cull } v_s \text{ if } & \quad \|w_s(n) - w_s(0)\| < w_l \end{aligned} \quad (6)$$

where v_l is the set of vertices in the shorter line , v_s vertices in the longer, and t_l is the intersection point on the longer line.

Likewise, extremely short lines are culled to reduce noise. A user-defined threshold is used, and can be adjusted based upon the image being vectorized. The default value is defined to be the average line width in the image, and lines with lengths falling under this threshold are often able to be removed without impacting the overall results.

The remaining lines follow the centre of the rasterised contours correctly, with one vectorized line per contour. However in many cases, the lines are short and therefore not representative of the image structure. Figure 5(b) shows an example of this. To provide more coherent data, lines are therefore joined in the cases where they terminate near to each other and the ends are co-linear. The effect of this can be seen in Figure 5(c).

While the result in Figure 2(g) is not a perfect reconstruction of the image, the underlying line-based vector structure provides more accessible information than polygon based deconstruction.

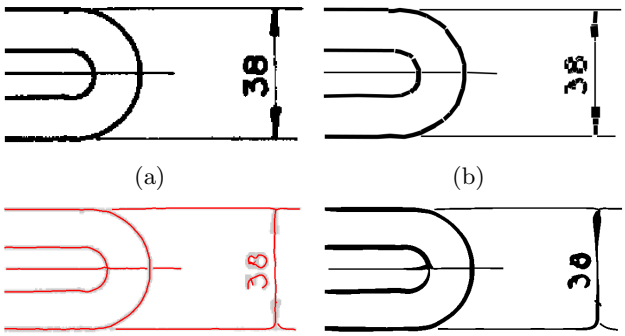
4. RESULTS

Figure 6 shows the input and output stages of the Structural Vectorization process performed on a complex object. The underlying structure of the image is extracted without significant distortion or interference, and without overlapping lines. Given the structure and the line widths, the original image can be reconstructed without the need for polygon outline information.

Structural Vectorization provides a different set of information to other common vectorization techniques. Figure 1 uses a typical cartoon image to illustrate the trade-off between visual quality and the underlying data, while Figure 8 shows the algorithms performance on typical structural data. Figure 7 shows how our generated topology differs from morphological algorithms.

(a) (b) (c)

Figure 6: Structural Vectorization performed on a low-resolution complex object (a) [12]. Significant lines are extracted from the image (b) along with line width information. This combination allows easy processing for image analysis or modification, while still retaining sufficient data (c) to represent the original image.



(a) (b) (d) (e)

Figure 8: Principally designed for organic and cartoon images, structural vectorization can also be applied to architectural or engineering drawings (a). Compared with a basic line-based vectorization method (b) [4], structural vectorization produces comparable structural data (c), and performs better on curved lines (d).

Complex images also provide vectorization challenges that would require context awareness to solve. Specifically, the algorithm has difficulty distinguishing independent shapes that are obscured by objects of the same colour. In Figure 6, this causes the middle section of the tail to become joined to the shadow. Note that the tip of the tail is sufficiently unique to be extracted as a separate stroke.

There is no single correct solution for the vectorization of strokes in a cartoon image, and therefore our algorithm attempts to provide a general solution that results in usable data across a range of input images. User tunable parameters also provide additional control over results. Tuned correctly, the output line data is useful for a range of applications. Analysis of line properties such as line length and camber can be used to identify and classify the author of an image, and vector graphics using line centres provides better time-stability than polygons when working with animation. In analysing hand drawn cartoon images, the topological line data produced by the algorithm shows that line based structural vectorization does not need to be limited to fixed-width strokes or technical drawings.

5. ACKNOWLEDGMENTS

To the New Zealand Government Foundation for Research Science and Technology, Stickmen Studios, ELLIIT and Intel Visual Computing Instituite, Saarbruecken, Germany for funding. Copyright of original images belong to the respective authors and are reproduced here with permission. Some images are creative commons.

6. REFERENCES

[1] C. Arcelli and G. S. di Baja. Euclidean skeleton via centre-of-maximal-disc extraction. *Image and Vision Computing*, 11(3):163 – 173, 1993.

[2] X. Bai, L. Latecki, and W.-Y. Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):449 –462, march 2007.

[3] H. Blum. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[4] D. Dori and W. Liu. Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:202–215, March 1999.

[5] E. Dougherty. *An introduction to morphological image processing*. Tutorial texts in optical engineering. SPIE Optical Engineering Press, 1992.

[6] C. Eliopoulos. Misery Loves Sherman. http://www.miserylovessherman.com, 2010.

[7] D. Elliman. A really useful vectorization algorithm. In *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances*, GREC '99, pages 19–27, 2000.

[8] W. T. Freeman, J. B. Tenenbaum, and E. C. Pasztor. Learning style translation for the lines of a drawing. *ACM Trans. Graph.*, 22, January 2003.

[9] A. Hertzmann, N. Oliver, B. Curless, and S. M. Seitz. Curve analogies. In *Proceedings of the 13th*

Eurographics workshop on Rendering, pages 233–246, 2002.

[10] M. Huang, M. Yang, F. Liu, and E.-H. Wu. Stroke extraction in cartoon images using edge-enhanced isotropic nonlinear filter. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry*, VRCAI '10, pages 33–38, 2010.

[11] B. Kegl and A. Krzyzak. Piecewise linear skeletonization using principal curves. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 131 –134 vol.3, 2000.

[12] O. Knörzer and P. Andini. Sandra and Woo. http://www.sandraandwoo.com, 2010.

[13] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, Sept. 1992.

[14] N. Mayya and V. Rajan. Voronoi diagrams of polygons: A framework for shape representation. *Journal of Mathematical Imaging and Vision*, 6:355–378, 1996.

[15] S.-m. H. Ming-ming Cheng. Curve structure extraction for cartoon images. Technical Report TR-081201, Tsinghua University, Beijing, China, 2008.

[16] A. S. Montero and J. Lang. Skeleton pruning by contour approximation and the integer medial axis transform. *Computers & Graphics*, 36(5):477 – 487, 2012. Shape Modeling International (SMI) Conference 2012.

[17] L. Olsen and F. F. Samavati. Stroke extraction and classification for mesh inflation. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 9–16, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[18] P. Selinger. Potrace: a polygon-based tracing algorithm. 2003.

[19] S.-H. Zhang, T. Chen, Y.-F. Zhang, S.-M. Hu, and R. R. Martin. Vectorizing Cartoon Animations. *IEEE Transactions on Visualization and Computer Graphics*, 2009.