Scribble Vectorization Using Concentric Sampling Circles

Alexandra Bonnici, Kenneth P. Camilleri
Department of Systems and Control Engineering
Faculty of Engineering
University of Malta
abbart@eng.um.edu.mt, kpcami@eng.um.edu.mt

***Abstract*—In this paper we introduce a path extraction algorithm for multi-stroke scribbled paths by making use of path-centred concentric sampling circles. Circle and line geometry is then exploited to efficiently obtain piece-wise linear models of the multi-stroke segments in the drawing. Parzen-window estimation is used to obtain the probability distribution of the grey-level profile of the sampling circles to determine the intersecting angle of the sampling circle with the stroke segments and hence determine the line model parameters. The results obtained show that the algorithm identifies the line models accurately while reducing considerably the computational time required to obtain the line models.**

***Keywords*-Scribbles, Vectorization, Parzen-windowing**

I. Introduction

Drawings are used in various disciplines as a means of expressing concepts which are not easily expressed through verbal communication alone. Scribbles are particularly useful for expressing visual concepts and are used continuously by designers, especially in the early design stages. These scribbles typically illustrate a rough outline of the conceptual object and are characterized by over-tracing, whereby, random strokes are arranged into stroke-groups that form the underlying drawing outline. It is interesting to note that despite the advent of computer-based drawing tools such as Paint Shop Pro® or Photo Shop®, designers rely on paper-based scribbling in the initial design stages as this places minimal drawing constraints and is by far the easiest to use. However, paper-based scribbling has its limitations and designers often introduce computer-based editing later in the design process. This will, for example, allow the designer to experiment with different rendering techniques such that a realistic representation of the initial design concept may be obtained. Furthermore, designers tend to rely on computer-aided design (CAD) tools to make the accurate drawings required in the advanced design stages. These tools give the added benefit of obtaining 3D, virtual prototypes of the design, allowing the designer to create 'walk-through' animations of the concept [1].

Thus, design solutions starting as simple paper-based scribbles evolve into rather complex graphic objects during a design process. However, to make this transition, the designer is often required to redraw the initial scribbles since CAD tools that allow the designer to create 3D models cannot interpret the designer's intent from the inaccurate scribble representations. Although research in computer-aided sketching has provided tools such as 'I Love Sketch' [2], FibreMesh [3] and SketchUp [4], these tools involve online drawing and assume that designers are willing to draw directly onto a computer. However, surveys with practicing designers show that designers tend to prefer paper-based scribbling in the early design stages [5]. The interpretation algorithms required by online scribbling systems are considerably different from those required to interpret paper-based scribbles, mainly due to the fact that for online scribbling it is possible to capture each stroke drawn by the user as a separate entity together with the temporal information about each stroke. This temporal information has been used by these systems to identify intended stroke-groups. In paper-based scribbling, this information cannot be made available without constraining the designers' freedom. This means that the interpretation of paper-based scribbles requires that the scribble stroke-groups intended by the designer are identified from the raster image, a processing step which online systems do not require.

Although the identification of the designer's intent from scribble drawings may seem to be a trivial matter since the human visual system allows us to do so quickly and with little effort, obtaining a machine interpretation of the scribbles is considerably complex and there is little literature on paper-based scribble vectorization methods. One such method is that described by Bartolo et al in [6]. This stroke simplification algorithm is based on Gabor filtering and requires post-processing to obtain the required vectors. This post-processing requirement motivates this work which makes use of concentric *sampling circles* to obtain histograms of the scribbled strokes to track the intended drawing outline while modeling the scribble strokes as piece-wise linear line segments. This enables us to obtain vector data from the scribble drawing through direct image-domain analysis.

The rest of the paper is organized as follows: Section II gives the literature background on the interpretation of paper-based scribble drawings, Section III outlines the proposed concentric circle tracking algorithm, Section IV presents and discusses the results obtained while Section V concludes the paper.

II. Literature review

The majority of the scribble drawing interpretation algorithms are directed for on-line sketching environments. These usually involve fitting mathematical models such as splines to the stroke segments in real time once each stroke has been completed. Proximity and orientation thresholds [7], [8] or gestural commands [9] are used to distinguish between strokes that are intended to form a new edge segment and those that are intended to modify existing edges. Since these methods utilize the temporal information available in online sketching environments to interpret the scribbled drawing incrementally, these methods cannot be adopted for the simplification of paper-based scribbles.

Vectorization algorithms such as Sparse Pixel Vectorization (SPV) [10] are normally used to represent paper-based drawings using vectors or parametric curves. However, such algorithms were developed to vectorize neat drawings and do not cater for the over-sketching that is normally present in scribbled drawings [6]. Other path extraction algorithms which allow the extraction of line paths from paper-based sketchy drawings include that described in [11]. This technique first thins the lines to single pixel thick lines after which they are tracked. Principal Component Analysis (PCA) is then applied at junction regions, which are points of ambiguity, to determine the most salient direction given the current tracking direction. This algorithm is shown to have good results for sketched drawings but it is not suitable for scribbled drawings which are characterized by over-sketching along the whole drawing contour. Bartolo et al. [6] developed an algorithm designed to be able to process such scribbled drawings. This method is based on a stroke grouping step achieved by using a specific Gabor filtering scheme followed by line tracing of the resulting stroke groups. This method was shown to have good results but depends on the resolution of the Gabor filter scheme and assumes that the entire scribbled drawing may be processed using the same filter resolution. The proposed technique avoids the filtering steps, thus removing this problem by obtaining sample statistics of the stroke groups from which piece-wise linear vectors are derived to model the underlying stroke groups.

III. Proposed Scribble Tracking Algorithm

The underlying concept of the proposed scribble path tracking algorithm involves placing concentric sampling circles on the stroke-groups in the scribbled drawing. The probability density function of the scribble strokes that intersect with the sampling circle is obtained and this is used to obtain an initial estimation of the orientation of the underlying line model as well as the width of the stroke group. The radius of the sampling circles is then iteratively increased, obtaining at each iterate a new probability density function for the stroke groups that intersect with the sampling circle and hence, new orientation and stroke width estimates. The orientation estimates are compared to the estimates of the underlying line model parameters in order to determine whether the line model parameters are still a valid representation of the stroke-group. If these are found to be valid, the new parameter estimates are used to update the line model, otherwise, a new line model is initiated, hence obtaining a piece-wise linear representation of the scribble. The rest of this section describes the concepts governing the different aspects of the algorithm.

A. Using circles to validate and determine line parameters

In this algorithm, we define each sampling circle as:

$$\mathbf{x}_s = \mathbf{x}_{os} + r_{s_n}[\cos\theta, \sin\theta]^T \tag{1}$$

where $\mathbf{x}_s = [x_s, y_s]$ defines the coordinates of the sampled points lying on the circumference of the sampling circle, $\mathbf{x}_{os} = [x_{os}, y_{os}]$ is the centre of the sampling circle while r_{s_n} is the radius of the n^{th} sampling circle. The angle $\theta \in [0, 2\pi]$ spans the circumference of the circle at an angular resolution of $\delta\theta$.

The underlying scribble strokes can be modelled, up to some approximation, by a straight line segment which can be represented by (2), where r_l and θ_l are the parameters of the line and $[x_l, y_l]^T$ are the coordinates of points on the straight line as shown in Figure 1(a).

$$r_l = x_l \cos\theta_l + y_l \sin\theta_l \tag{2}$$

A sampling circle centred on this line intersects the line at co-ordinates that satisfy (1) and (2), that is, satisfying (3)

$$r_l = (x_{os}\cos\theta_l + y_{os}\sin\theta_l) + r_{s_n}\cos(\theta - \theta_l) \tag{3}$$

One may note that since the circle centre lies on the line, then the coordinates of the circle centre $\mathbf{x_{os}}$ satisfy (2), such that $x_{os}\cos\theta_l + y_{os}\sin\theta_l = r_l$ and (3) is reduced to:

$$r_{s_n}\cos(\theta - \theta_l) = 0 \tag{4}$$

Hence, as shown in Figure 1, the sampling circle will intersect the line at two points namely at Y_1, where $\theta_1 - \theta_l = \frac{\pi}{2}$ and at Y_2 where $\theta_2 - \theta_l = \frac{3\pi}{2}$ such that the difference between the intersecting angles $|\theta_1 - \theta_2| = \pi$. Since the angle θ_l is a fixed parameter of the straight line segment, then, the intersections of concentric sampling circles of increasing radii will always occur at the same values of θ_1 and θ_2. This can be used for two purposes, namely to determine the extent to which the straight line model is a valid description of the line and to identify the unknown parameters of the straight line.

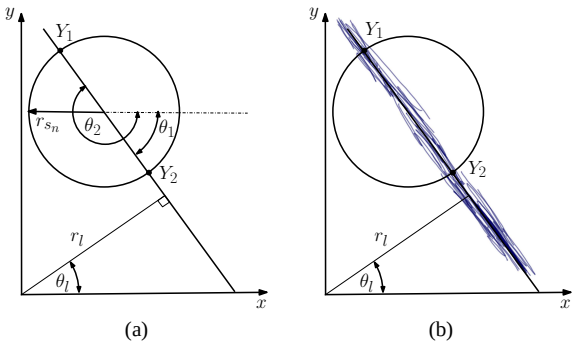


Figure 1: (a) A sampling circle centred on the line segment intersects the segment at two intersecting angles θ_1 and θ_2 (b) When the sampling circle is centred on the scribble stroke, it intersects with a number of sample points distributed on the circumference of the sampling circle.

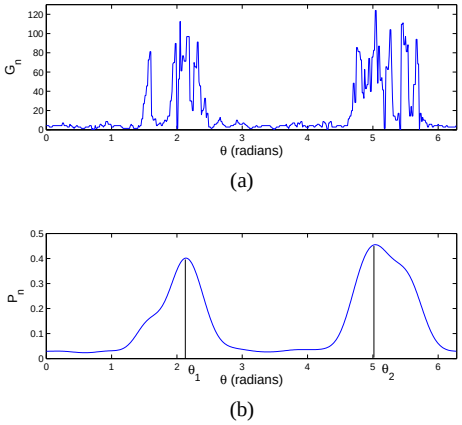


Figure 2: (a) The grey level profile obtained from a sampling circle centred on a scribble stroke. (b) The Parzen estimate of the probability density function obtained for this grey-level profile.

B. Circle and scribble stroke group intersection

In the scribbled drawing, the stroke segments are randomly distributed on the underlying drawing outline such that when the sampling circle is centred on the over-traced stroke group, this does not intersect at a single sample point, but on a number of sample points along the circumference of the sampling circle as shown in Figure 1(b). Therefore, for a scribbled drawing I, an estimate of the intersecting angle must be obtained by analyzing the grey-level distribution $\mathbf{G}_n(\theta_k) = I(\mathbf{x}_s)$ along the circumference of the sampling circle. Taking the stroke intensity to be above the background intensity, $\mathbf{G}_n(\theta_k)$ will result in a discrete probability density function of the intended line strokes as shown in Figure 2(a). The probability density function of each stroke group can be used to obtain an estimated intersection angle and an estimated stroke width and hence validate the line model. However, this probability density function is sensitive to the gaps present between the stroke groups and this makes it difficult to determine whether the distributions are due to a single stroke group or to multiple stroke groups. For this reason, Parzen estimation with a Gaussian kernel, defined by (5) is used to obtain a robust statistical model of the probability density function.

$$\mathbf{P}_n(\theta_k) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-\|\mathbf{G}_n(\theta_k)-\mathbf{G}_n(\theta_i)\|}{2\sigma^2}\right\} \quad (5)$$

where σ is the standard deviation of the Gaussian function, M is the length of the grey-level profile $\mathbf{G}_n(\theta_k)$ [12]. The probability density function will contain a number of high energy bands corresponding to the stroke-groups intersected by the sampling circle as shown in Figure 2(b). Each of these bands can be considered as a probability density function which models the statistics of the stroke group, such that it is necessary to separate the individual probability density functions. We achieve this by evaluating the mean energy of $\mathbf{P}_n(\theta_k)$ and using it as a threshold, below which the $\mathbf{P}_n(\theta_k)$ is considered as being due to the image background, hence separating the individual probability density functions.

The mode of each of the individual probability density function occurs either at strokes which have a higher intensity or at strokes which occur at a higher density. Both cases are indicative of higher adherence to the underlying drawing outline. Thus, the angle at which the mode of each probability density function occurs is used as the estimated intersection angle as shown in Figure 2(b).

C. Path extraction

The path extraction algorithm has three steps, namely an initialization step, a path following step and a junction recovery step. In the initialization step, the concentric circles are used to determine the parameters of the new linear segment. The path following step is used to determine the extent to which these parameters are valid while the re-initialization step is used to determine a new centre point once the line parameters are no longer valid.

1) The initialization step: The algorithm assumes that the centre of the first sampling circle is at the centre of the stroke group and that this centre point will be specified by the user. The algorithm also assumes that the radius r_{os_1} of the first sampling circle is such that the sampling circle is large enough to intersect with the stroke-group at two intersecting angles. This implies that the r_{os_1} must be greater than twice the width of the stroke-group on which it is centred. If r_{os_1} is smaller than this, the sampling circle will be located within the stroke-group and $\mathbf{P}_n(theta_k)$ would not have two distinct probability density functions. Thus, the initial value of r_{os_1} is iteratively increased in steps of δr until two distinct probability density functions are identified. This sampling circle is considered as the first sampling circle from which the estimated intersecting angles θ_1 and θ_2 are obtained. These estimated intersecting angles are used to verify that a line model may be initialized at that point, by determining the difference $|\theta_1 - \theta_2|$.

The initial point can be considered as a valid point for a line model if $|\theta_1 - \theta_2| = \pi \pm \theta_d$ where θ_d is a tolerance on the accuracy of the line model. The tolerance value θ_d is necessary to take into account the roughness of the scribbled drawing which causes shifts in the estimated intersection angles even if the scribbled strokes are visually perceived as forming part of the same line segment. If the selected centre point is found to lie on a straight line segment, the estimated intersection angles θ_1 and θ_2 as well as the width of the stroke-group are used to obtain initial estimates of the line parameters. These are then used in the subsequent path following step.

2) The path following step: This is an iterative step which increases the radius of the sampling circle by δr in order to determine the extent with which the underlying drawing outline may be represented by a straight line model. This can be achieved comparing the estimated intersecting angles obtained for the n^{th} sampling circle with the corresponding estimated intersecting angles, obtained from the previous, $(n-1)^{th}$, sampling circle. If the difference between the estimated intersecting angles $|\theta_n - \theta_{n-1}|$ is within some tolerance θ_d, then the line model parameters are considered a good representation of the underlying drawing outline.

In practice however, evaluating the probability distribution of the grey-level intensities over the entire sampling circle is unnecessary and an arc section of the sampling circle which contains just the concerned probability density function is sufficient to determine the suitability of the line model for the n^{th} sampling circle. To determine the location and size of this arc, the estimated intersecting angle and stroke width obtained for the n^{th} sampling circle are propagated onto the $(n+1)^{th}$ circle to determine the angular range α over which the over-traced stroke segment is expected to intersect with the $(n+1)^{th}$ circle. This is achieved by taking into account that the estimated intersecting angle is expected to remain the same for strokes that can be modeled by straight line segments and that the width of the strokes w is expected to remain consistent, such that the angular range α can be defined as

$$\alpha_{n+1} = \theta_n \pm \frac{w_n}{r_{n+1}} \tag{6}$$

This selection reduces the computational costs by ensuring that the Parzen window estimation is evaluated only for an arc of interest rather than for the entire sampling circle. Furthermore, by using only a selected arc of the sampling circle it is possible for the path following step to propagate the line path even when the stroke has neighbouring strokes which would intersect with the sampling circle should this be considered in its entirety.

If the new sampling circle determines that the scribbled strokes may be represented by the same line parameters, then the intersecting angle and arc-length determined from the $(n+1)^{th}$ sampling circle are used to update the line parameters and the path following step is repeated. On the other hand, if the new sampling circle determines that the line model is no longer suitable for the scribbled strokes, a re-initialization step is performed in order to initialize a new path.

3) The re-initialization step: This step is performed after the path following step in order to identify a new centre point for the sampling circles. This step will therefore allow the algorithm to initialize a new line model for the scribbled strokes. We achieve this by taking a new circle using the last valid track point as the centre of the circle. The Parzen-window estimation is performed on the grey-level profile obtained from the circumference of this circle. This will enable the algorithm to determine the stroke segments in the vicinity of the current stroke. One of these stroke segments is selected arbitrarily and the intersection point is used as the centre point for the new sampling circles. The line model is initiated using the new intersection angle as parameters. The path following step is again used in order to determine the full extent of the new line model and the process is repeated until no other stroke segments may be determined.

D. Implementation Issues

The path extraction algorithm makes use of five parameters namely, the orientation resolution $\delta\theta$, the standard deviation σ of the Parzen window estimator, the initial radius r_{os_1}, the increment in the length of the radius δr and the angle threshold θ_d. These parameters must be suitably set to ensure proper path tracking. The orientation resolution $\delta\theta$ determines the distance between two sampling points on the circumference of the circle. Thus, in order to ensure that the distance between these sampling points remains constant for sampling circles of different radii, the orientation resolution $\delta\theta$ must change accordingly. We choose to set the distance between two points on the circumference to one pixel to ensure that all pixels on the circumference of the circle are sampled. Then, the angular resolution must be set to $\delta\theta = \frac{1}{r_n}$.

This will imply that the length of the grey-level profile M will increase as the radius of the sampling circle increases, specifically, $M = 2\pi r_n$. Thus, the standard deviation for the Parzen window must also increase as the radius of the sampling circle increases as otherwise, the length of the arc over which the Gaussian function is effective will vary from one sampling circle to the other. This would be undesirable as it would imply that the Parzen window would be less effective for circles that have larger radii. In our implementation, we set the standard deviation to $0.03M$.

In a similar manner, the angle threshold θ_d must also vary with the radius of the sampling circle. In line fitting, it is standard practice to place a threshold on the maximum distance between a pixel and the line in order to determine if the pixel fits the line [13]. However, if a similar fixed tolerance value is applied to the intersecting angle, the path

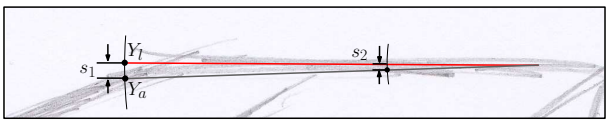


Figure 3: Fixed orientation tolerances will allow larger deviations from the line model for sampling circles with larger radii

following step would allow larger deviations from the line for sampling circles with larger radii as shown in Figure 3. Thus, rather than specifying a tolerance value for the intersection angle, we specify the maximum accepted distance s between the actual intersection point of the stroke-group and the sampling circle Y_a and the intersection point estimated by the line model Y_l as shown in Figure 3. For small angles, this corresponds approximately to the arc length between the predicted intersection angle and the actual mode angle such that $s \approx \theta_{d_n} r_n$. Thus, the angle tolerance is set to $\theta_{d_n} = \frac{s_m}{r_n}$.

The value selected for the initial radius r_{os_1} is not critical since the initialization adjusts the size of the initial sampling circle in order to obtain two distinct density functions. However, following an empirical study on the line widths of the scribbled strokes obtained from scribbles such as that shown in Figure 4(c), we determined that an initial value of r_{os_1} pixels was a suitable value for images having a resolution of 300dpi, requiring few adjustments in the initialization step.

The radius increment δr determines the number of sampling circles taken for a given stroke segment. While a larger radius increment reduces the number of sampling circles required and hence reduce the computational costs required to obtain the line models, smaller radius increments will allow more detailed path following. In particular, at junction regions, smaller radius increments will allow the path following to approach closer to the junction, hence obtaining a better representation of the stroke segments. We set the radius increment to $\delta r = 15$ pixels which allows us to obtain good precision in the line models.

IV. Results and Discussion

Figure 5 shows the result obtained by the proposed algorithm for the neat sketches and scribble shown in Figure 4 and compares this result to that obtained by the sparse-pixel vectorization (SPV) algorithm described in [14]. The performance of the two methods can be compared using the evaluation protocol described in [10]. In particular, the Pixel Recovery Index (PRI) is defined as

$$PRI = \gamma D_p + (1-\gamma)(1-F_p) \tag{7}$$

where $D_p = \frac{\#P_g \cap \#P_d}{\#P_g}$ is the pixel detection rate, $F_p = 1 - \frac{\#P_g \cap \#P_d}{\#P_d}$ is the pixel false alarm rate, P_g is the set of foreground pixels present in the ground truth image while P_d is the set of foreground pixels detected by the path extraction algorithm, γ is the relative importance of the detection and is set to 0.5. The two algorithms were used to perform path extraction on sketchy images such as that shown in Figure 4(b). For these images, the SPV algorithm obtained an average PRI of 0.69 with a standard deviation of 0.10 while the proposed algorithm obtained an average PRI of 0.70 with a standard deviation of 0.15. This shows that the performance of the proposed algorithm is comparable to that obtained by the SPV algorithm for relatively neat drawings. However, the SPV algorithm cannot be applied to the scribbled drawings such as that shown in Figure 4(c) whereas as shown in Figure 5 the proposed algorithm can extract line paths from these scribbles too. Figure 5(c) shows that the proposed algorithm results in good localization of the line paths even when these are obtained from the scribbled drawing. This is possible because the Parzen window estimation allows the grouping of the over-strokes present in the scribbled drawing such that the angle at which the sampling circles and the stroke segments intersect is evaluated for the stroke group rather than for the individual strokes. The Parzen estimation used in this algorithm is applied only to a section of the sampling circle, where this section is being selected according to the line model parameters. This contrasts with the scribble simplification algorithm described in [6] where the stroke simplification requires the use of a number of Gabor filters which are applied to the entire image. As a result, the proposed algorithm reduces the computational time required to obtain the line paths, in particular, for images of size 1980×1540 pixels the proposed algorithm requires a processing time of 310s on a 1.8GHz computer, whereas the scribble simplification algorithm described in [6] requires 880s.

V. Conclusion

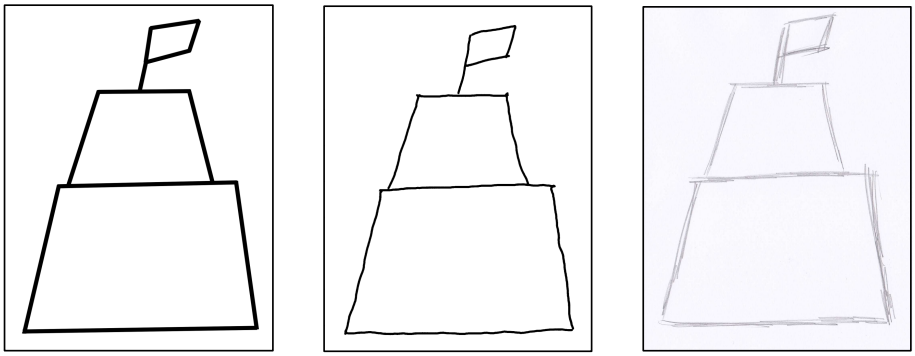
This paper describes a vectorization algorithm that can be used to extract piece-wise linear vectors from scribbled drawings. The results obtained show that the algorithm's accuracy compares well with the accuracy obtained by established vectorization algorithms such as SPV for neat drawings. Moreover, the algorithm performs faster than other scribble simplification algorithms.

Although the vectorization algorithm uses piecewise linear models, the work reported in this paper may be extended to include other stroke geometry models, including for example, circular arcs. This may be done by retaining the same framework of sampling circles but exploiting the intersections of the sampling circles with different geometric models.

References

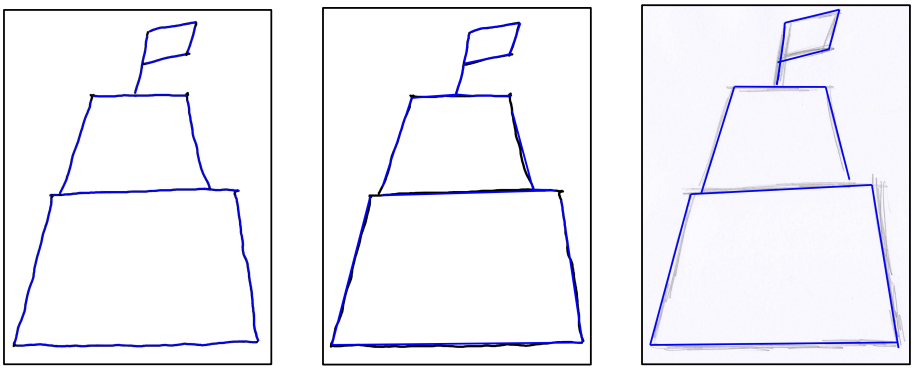
[1] A. Pipes, *Drawing for Designers*, R. Mason, Ed. Laurence King Publishing, 2007.

[2] S.-H. Bae, R. Balakrishnan, and K. Singh, "I Love Sketch: As-natural-as-possible sketching system for creating 3D curve models," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2008.



(a) (b) (c)

Figure 4: (a) A ground truth representation, (b) A neat sketch, (c) A scribbled drawing



(a) (b) (c)

Figure 5: Results obtained by (a) SPV algorithm for the neat sketch, (b) proposed algorithm for the neat sketch (c) proposed algorithm for the scribbled drawing. In each case, the vector results, shown by blue segments, are overlayed on the original images.

[3] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "FiberMesh: Designing Freeform Surfaces with 3D Curves," in *Proceedings of the 2007 SIGGRAPH conference on Computer Graphics*, vol. 26, no. 3, 2007.

[4] Google, "Google SketchUp," information page last accessed 15th April 2009. [Online]. Available: http://sketchup.google.com/

[5] J. A. Landay and B. A. Myers, "Sketching interfaces: toward more human interface design," *Computer*, vol. 34, no. 3, pp. 56–64, March 2001.

[6] A. Bartolo, K. P. Camilleri, S. G. Fabri, J. C. Borg, and P. J. Farrugia, "Scribbles to Vectors: Preparation of Scribble Drawings for CAD Interpretation,," in *Proceedings of the Eurographics Workshop on Sketch-Based Interfaces and Modeling 2007*, 2007.

[7] F. D. Fiore and F. V. Reeth, "A Multi-Level Skethcing Tool for "Pencil-and-Paper" Animation," in *AAAI Spring Symposium on Sketch Understanding*, 2002, pp. 32–36.

[8] D. C. Ku, S. F. Qin, and D. K. Wright, "Interpretation of Overtracing Freehand Sketching for Geometric Shapes," in *Proceedings of the 14th International Conference on Computer Graphics, Visualization and Computer Vision*, 2006, pp. 263–270.

[9] L. B. Kara and K. Shimada, "Sketch Based Design of 3D Geometry," in *In Eurographics Workshop on Sketch Based Interfaces*, J. J. and S. T., Eds., 2006, pp. 59–68.

[10] W. Liu and D. Dori, "A protocol for Performance Evaluation of Line Detection Algorithms," *Machine Vision Applications*, vol. 9, pp. 240–250, 1997.

[11] P. Rajan and T. Hammond., "From Paper to Machine: Extracting Stokes from Images for use in Sketch Recognition," in *Eurographics 5th Annual Workshop on Sketch-Based Interfaces and Modeling*, 2008.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*, M. Jordan, J. Kleinberg, and B. Scholopf, Eds. Springer, 2006.

[13] J. Sklansky and V. Gonzalez, "Fast polygonal approximation of digitized curves," *Pattern Recognition*, vol. 12, no. 5, pp. 327 – 331, 1980.

[14] W. Liu and D. Dori, "Sparse Pixel Vectorisation: An Algorithm and Its Performance Evaluation," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 202–215, 1999.