Line Tracking Algorithm for Scribbled Drawings

Alexandra Bartolo*, Kenneth P. Camilleri*, Simon G. Fabri*and Jonathan C. Borg†
*Department of Systems and Control Engineering
University of Malta
Email: [abbart, kpcami, sgfabr]@eng.um.edu.mt
†Department of Industrial and Manufacturing Engineering
University of Malta
Email: jonathan.borg@um.edu.mt

***Abstract*—This paper describes a line tracking algorithm that may be used to extract lines from paper based scribbles. The proposed algorithm improves the performance of existing sparse-pixel line tracking techniques that are used in vectorization by introducing perceptual saliency and Kalman filtering concepts to the line tracking. Furthermore, an adaptive sampling size is used such that it is possible to adjust the size of the tracking step to reflect the stroke curvature.**

I. Introduction

Scribbling is a fast and efficient method through which mental ideas are formalized. Although digital tablet technology is developing at a fast rate, it is unlikely that digital tablets will completely replace paper-based scribbling, particularly in shape and form design applications where designers tend to prefer paper-based scribbling to digital scribbling [1]. Scribbles however, cannot be integrated directly with CAD tools which are used in subsequent design stages. Sketch based interfaces that integrate the scribbles with CAD tools are therefore desirable so that the flexibility of paper-based drawings can be augmented with the 3D virtual CAD modeling, giving designers the opportunity to invest more time on design rather than interfacing with the CAD tools.

Research in this area may be broadly divided into two categories, namely the creation of algorithms that interpret paper-based drawings and the creation of interfaces that make digital sketching as close as possible to paper-based sketching. Unfortunately, paper-based interpretation algorithms are mostly restricted to neat, technical drawings and do not handle sloppy or scribbled drawings, whereas online interfaces, while allowing the interpretation of scribbled drawings, require human interaction and do not allow designers to use their preferred drawing medium.

This paper gives the details of a line tracking algorithm which forms part of a scribble simplification procedure described in [2]. This scribble simplification replaces traditional line location techniques such that when used in conjunction with an annotation-based sketching language described in [3], allows designers to create 3D CAD models from paper-based scribbles.

The rest of the paper is divided as follows: Section II gives a brief review of existing line extraction and vectorization algorithms, Section III presents our proposed solution; evaluation of the proposed solution and results are presented in Section V; and Section VI presents the conclusions that can be derived from this study.

II. Extracting Lines from Scribbled Drawings

Paper-based line drawings cannot be used as direct input to CAD systems since the digital representation of the scribbled drawings is saved in a raster format, whereas the CAD tools typically require graphics represented in vector format. Thus, in order to edit the paper-based drawing using the CAD tools, the designer's line strokes must be extracted from the raster image and represented with appropriate mathematical models whose parameters may be estimated. This may be carried out through vectorization, which may be broadly described as a three-step process consisting of line location, line fitting and line beautification.

Vectorization techniques have been traditionally developed for applications involving neat drawings, such as engineering drawings [5], architectural drawings [6] or cadastral maps [7]. Since such lines are rarely of a single pixel thickness, a line location procedure is required to eliminate redundant pixels from the line drawing, retaining only those which are necessary to maintain the shape and connectivity of the object [8]. Thus, the line drawing is represented by a thin sequence of pixels which may be referred to as the medial axis of the drawing [9]. However, although this representation retains only the pixels necessary to preserve the topology of the line drawing, the representation is still in a raster format and further processing is required in order to obtain the vector representation as typically required by CAD tools. This is carried out through the line fitting step which splits the connected sequence of pixels such that each pixel group may be approximated by equations of straight lines or curves of known parameters [8]. Most interpretation systems apply a subsequent, final line beautification step to this vector data before using the CAD tool [8]. This beautification is necessary to correct errors that may be due to the freehand nature of the drawing. Thus each vector is compared to its neighbouring vectors in order to identify relations such as collinearity or parallelism, hence introducing a degree of contextual knowledge to the vector data [8].

The line location procedure is the most crucial step in the vectorization process since this determines the accuracy of the resulting vectors with respect to the line drawing. For

this reason, the main difference between various vectorization algorithms is found in the line location step, with the goal of each algorithm being to find lines accurately with minimal computational expense. Vectorization techniques can therefore be divided into six main categories according to the techniques used to locate lines namely, Skeleton based methods, Contour based methods, Skeleton-Contour based methods, Hough based methods, Run-graph based methods and Sparse-pixel based methods [10].

The scribbled drawings have a mixture of linear and curved stroke segments, such that the line location algorithm should be capable of extracting linear and curved strokes without causing unnecessary segmentation of the strokes or require additional processing to locate the curved strokes. It is also necessary that the line location represents the strokes using the minimal number of points without under-sampling the strokes. Finally, unlike neat drawings, the scribbled drawing is not necessarily an exact representation of the designer's intent, such that perceptual grouping of strokes is necessary to identify the most likely object shape.

Existing line location techniques do not meet these criteria. In Skeleton and Contour based line location methods, the representation contains all the points on the medial axis. Although Run-based and Sparse-pixel based algorithms represent the medial lines by a sample of points, the sampling rate is a user-defined parameter such that the designer must identify a suitable sampling rate for the drawings. Furthermore, Run-based algorithms and some Sparse-pixel algorithms are more suitable for straight lines, segmenting curves unnecessarily. Other Sparse-pixel algorithms and Hough-based algorithms require a different line location process to locate the curves. These algorithms assume that the curves may be represented as circular arcs, however, this restricts the objects that may be represented, particularly, free-form object profiles which may be more suitably represented by splines. Furthermore, none of the vectorization algorithms described in the literature combines perceptual selection of strokes with the vectorization process.

III. The Proposed Line Tracking Algorithm

This work proposes the use of a sparse pixel tracking approach, similar to the Sparse Pixel Vectorization (SPV) algorithm described in [5]. This approach has been selected as it allows the introduction of the Gestalt laws of perceptual grouping into the line tracking procedure and this, together with other modifications described below, enhances the performance of the SPV algorithm.

The tracking algorithm is performed after processing the scribble by Gabor-filter based algorithm to group scribble line strokes together therefore, information on the orientation of each line passing through each pixel is known [2]. These orientation estimates for the basis of the perceptual selection of the line tracking direction at junction points. Since the proposed line tracking algorithm is algorithmically independent of the Gabor filter grouping algorithm, any other line group orientation estimation method can be used to obtain the required orientation estimates. Using these orientation estimates, the co-ordinates of the tracking point at an tracking instant $k+1$ may be determined from the position of the track point at a tracking instant k using (1), where θ_k is the line orientation at point (x_k, y_k) and D is the distance between two consecutive line points.

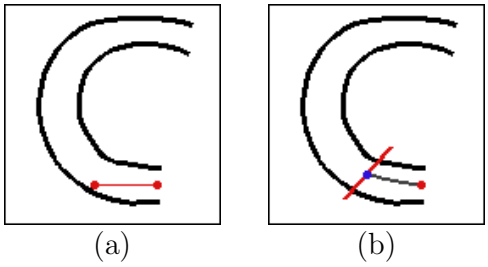

Fig. 1. *The piecewise linear path propagation is accurate in straight line segments such but may cause the path line to deviate from the medial axis points in curved segments as shown in (b). To solve this, scan lines perpendicular to the line segment are taken such that the tracking point is placed in the middle of the line profile as shown in (c)*

$$x_{k+1} = x_k + D\cos(\theta_k)$$
$$y_{k+1} = y_k + D\sin(\theta_k) \quad (1)$$

This piecewise linear tracking works well for straight line segments but in the case of curved line segments, determining the next point using (1) alone may cause the tracking points to offset from the line's medial points as shown in Fig. 1(a). This offset becomes more evident when the distance D between consecutive points is increased. In order to adjust for this offset, the orientation of the line at the new point is determined such that a scan line perpendicular to the line segment can be obtained. As shown in Fig. 1(b), this scan line gives a measure of the line's profile such that the midpoint (m_x, m_y) of the line may be determined. This point can therefore be considered as a better estimate for the point (x_{k+1}, y_{k+1}) determined through (1). The tracking procedure is repeated until either an end of line is reached or the tracking path reaches a segment region that has been previously tracked.

The tracking method described above is very similar to the tracking algorithm described in SPV [5]. However, in SPV tracking, the initial tracking step is carried out by moving the tracking point by a distance D either in a horizontal or in a vertical direction since the SPV tracking process has no prior knowledge of the line direction θ. This also affects the construction of the scan-line which may be constructed perpendicular to the line stroke, rather than using a vertical or horizontal scan of the line. Thus, the midpoint (m_x, m_y) obtained form the line profile is more reliable when evaluated using the proposed method than when using the SPV method. Using the line direction θ to evaluate the position of the next point has another advantage, namely that changes in the line direction are directly reflected as changes in the value of θ, such that curved paths may be tracked in a piece-wise linear manner with no adjustments to the tracking procedure. In

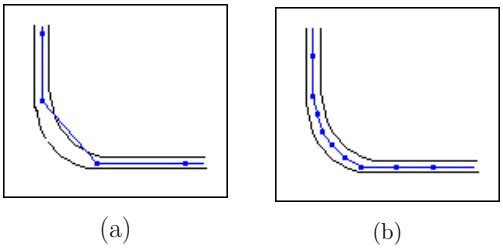

Fig. 2. *The tracking step D should reflect the properties of the line segments being tracked. (a) Using a constant tracking step size would result in poor track points in regions of high curvature. (b) A dynamic tracking step that adjusts according to the curvature allows finer tracking in regions of high curvature and a coarser tracking step in straight line regions.*

SPV however, a change in the line curvature of more than $\frac{\pi}{4}$ requires a change in the tracking direction from horizontal to vertical or vice versa. Such a change in direction is treated as a junction of two line segments such that the curved path cannot be extracted as a single entity, but is broken into smaller fragments.

A. Adaptive Tracking Step Size

Similar to the SPV algorithm, a tracking step size D must be initialized in order to perform the line tracking. This track step determines the sampling rate of the line path and must therefore be selected with care. Small values of D would result in a large number of sample points, increasing the data points extracted from the drawing and hence the computational time of the path following. On the other hand, large values may result in under-sampling of curved paths, as shown in Fig. 2(a). Therefore, it is necessary to determine a tracking step that retains only a subset of all the pixels forming the line path, while at the same time keeping a sampling step that retains sufficient points to represent curves accurately. Since the drawing may contain a range of segments with varying degrees of curvature, the tracking procedure ideally changes the track step adaptively such that large tracking steps are used in linear regions which may be defined by fewer tracking points, while using a finer tracking step size for curved regions whose representation requires a larger number of sample points.

In SPV, Liu and Dori change the value of the tracking step D when the tracking point encounters a junction region, in which case the tracking step is halved in an attempt to locate the exact start of the junction region. If the start of the junction is not located, the tracking step size is again halved and this is repeated until either the tracking step size is reduced to one pixel width, in which case the tracking is stopped, or the junction boundary is located. If the junction boundary is located, then the tracking step is reset to its original value in an attempt to proceed with the path tracking. Thus, SPV only reduces the step size if a junction region is encountered, otherwise, the tracking step is kept at a constant value.

In order to obtain an adaptive tracking step, the orientation of the strokes as well as the location of the tracking point, prior to its being moved to the centre of the scan line, are monitored at each tracking step. For straight line regions, the orientation at each tracking step is constant such that when (1) is used to determine the initial position of the next tracking step, this will be located within the stroke boundaries of the line segment. However, in curved strokes, (1) retains its validity for short tracking steps only such that if the tracking step is too large, the initial position is located outside of the stroke boundaries. Furthermore, a change in line orientation is measured at each tracking point, giving further indication of the curvature of the strokes.

For a given line path, the tracking step is too large when (1) propagate the tracking point to a new point on the image background. When this occurs, the tracking step is halved in an attempt to find an initial tracking point that lies within the stroke boundaries. If this is found then tracking resumes, using the new tracking step. If however, the new initial point is still beyond the stroke boundaries, the tracking step is again halved and the process repeated until a minimum tracking step of one pixel is reached, in which case the tracking is terminated.

On the other hand, a straight line region is detected when the tracked points are on the stroke foreground and the line orientation of the track points is constant. When this happens, the tracking step is incremented until either a maximum track step is reached or the stroke segments are no longer linear. Thus as shown in Fig. 2(b), it is possible to change the tracking step size such that the tracking uses a fine or coarse step size according to the characteristics of the line.

B. Treatment of Junction Points

The stroke orientations obtained through the Gabor filter provide to the proposed line tracking algorithm a distinct advantage over the SPV algorithm in the treatment of junction regions. Using SPV, the line tracking at junctions is terminated unless it is possible to continue tracking in the same tracking direction. In the proposed tracking algorithm however, path tracking may be pursued in any of the orientation estimates, corresponding to each line stroke present at the junction region. It is therefore necessary to select an orientation that allows the tracking algorithm to follow paths that are perceived as more salient than others. This motivates the use of the Gestalt laws of perceptual grouping which state that long, smooth continuous paths are preferred to others having a larger number of curvature changes while at the same time giving preference to closed objects over meandering paths. This implies that although smoothly continuous tracking is often a good tracking guide, the tracking procedure should not adhere strictly to this rule since a less smooth path can at times prove to give a better tracking solution. For example, at P1 in Fig. 3, smooth continuation would be preferred as this allows the tracking to retain the outer contour of the object. However, at P2, the smooth continuation would not support further tracking such that a less smooth tracking path would be preferred.

In this work we introduce the Gestalt laws to the tracking process in order to select the most salient orientation at junction regions where two or more line orientation estimates

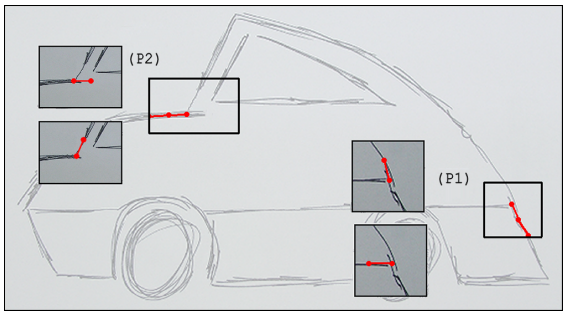

Fig. 3. *The smooth continuation criterion allows the tracking to follow the outer car strokes at P1. However, smooth continuation is not always desired since in instances such as P2 a change in the path curvature is required in order to maintain tracking.*

are obtained. The measure of saliency used has two parts, the first evaluates the smoothness of the path while the second evaluates the certainty with which the line parameters are known for each line at the junction. Since the line segments are only known up to the current tracking point, it is necessary to make a preliminary track step in each of the directions indicated by the dominant orientations obtained from the Gabor filter responses. Each of the dominant orientations will result in a preliminary track point (p_x, p_y) which will be used to evaluate the smoothness of the path on which it lies as well as the confidence of its neighbourhood.

The smoothness function used by [11], given by (2) is adopted, where α is the angle that that each preliminary tracking point (p_x, p_y) makes with the tacked path generated up till the current sample point.

$$f = \exp\left(\frac{2\alpha \tan\left(\frac{\alpha}{2}\right)}{D}\right) \tag{2}$$

However, the smoothness function does not determine if it is possible to maintain tracking in the selected direction, hence the need of a second saliency measure which determine whether tracking continuation is possible. This may be obtained by observing the pixel intensity $I(p_x, p_y)$ at each of the preliminary track points (p_x, p_y), where I represents the response obtained by a stroke-grouping algorithm such as the Gabor filter response. The total saliency s of the path can therefore be defined by (3).

$$s = I(p_x, p_y)f \tag{3}$$

Thus the saliency of the candidate line paths at the junction region depends on the smoothness of the path with respect to the current path as well as the possibility to extend the tracking beyond one tracking step.

The tracking algorithm described above determines two positions of the track point at each tracking step k. The first, initial track point is obtained through the line propagation defined by (1) and is therefore dependent on the line orientation θ_k only. The second position is determined from the scan line of the strokes and is therefore dependent on the accuracy and smoothness of the stroke boundaries. The reliability of the initial position (x_k, y_k) depends on the orientation resolution of the quadrature Gabor filter, and in the case of curved strokes, on the tracking step size used. The second position (m_x, m_y) is generally more accurate than the initial position since by locating the midpoint of the scan lines, the second point is adjusted such that it is closer to the medial axis of the line strokes. However, at junctions, where the accuracy of the scan lines is compromised, such that the error in the line boundaries may be greater than the orientation resolution of the Gabor filter, the initial tracking point is likely to be more accurate than the scan-line midpoints. Thus it is desirable to have a system that determines which of the two tracking points is more reliable, given the noise present in the line boundary at that instant and the noise due to the Gabor filter resolution.

The Kalman filter framework provides such a system by using the noise co-variance to determine a weight matrix that can be used to obtain a weighted average of the two tracking measurements. Although the Kalman filter is usually associated with dynamic systems, the Kalman filter has been applied to non-dynamic applications in order to track roads [12], characters in text documents [13] and lines in document images [14].

IV. Kalman Filter Tracking

The discrete time Kalman filter assumes that a system is governed by a process modeled by the linear stochastic model given by (4) for which a measurement $\mathbf{z}$ may be related to the system's state vector $\mathbf{x}$ using (5) [15].

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u} + \mathbf{w}_{k-1} \tag{4}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{5}$$

where $\mathbf{w}$ and $\mathbf{v}$ are random variables representing the process and measurement noise respectively which are assumed to be white Gaussian noise processes having zero mean and co-variance matrices Q and R respectively. Matrix A relates the state at the previous time instant $k-1$ to the state at the current time instant k, matrix B relates the optional control input $\mathbf{u}$ to the state $\mathbf{x}$ and matrix h relates the state $\mathbf{x}$ to the measurement $\mathbf{z}$.

The Kalman filter estimates the system state in two steps, referred to as the time update step and the measurement update step such that feedback obtained from the measurement $\mathbf{z}$ is used to obtain better estimates of the state $\mathbf{x}$. In the time update step, a prediction of the system state and error covariance is made using the knowledge gained about the process model, the system states and the error covariance up till just prior to the current time instant. These predictions are referred to as the *a priori* state estimate and the *a priori* error covariance and are denoted as $\mathbf{x}_{k|k-1}$ and $P_{k|k-1}$ respectively. They are given in (6).

$$\begin{aligned} X_{k|k-1} &= AX_{k-1|k-1} + BU_{k-1|k-1} \\ P_{k|k-1} &= AP_{k-1|k-1}A^T + Q \end{aligned} \tag{6}$$

The *a priori* state estimates and error covariance are then updated in the measurement update step to obtain state estimates that take into consideration the most recent measurements obtained from the system at the current time instant. These updated estimates are referred to as the *a posteriori* state and error covariance and are denoted as $\mathbf{x}_{k|k}$ and $P_{k|k}$ respectively. The measurement update step, defined by Equations 7 compares the actual measurement $\mathbf{z}_k$ to the 'predicted' measurement obtained when the measurement model H is applied to the *a priori* state estimates $\mathbf{x}_{k|k-1}$

$$\begin{aligned} K_k &= P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \\ X_{k|k} &= X_{k|k-1} + K_k(z_k - HX_{k|k-1}) \\ P_{k|k} &= (I - K_kH)P_{k|k-1} \end{aligned} \tag{7}$$

A gain matrix K, which is estimated to minimize the *a posterior* error covariance, is used to weight the difference between the two measurements and hence update the *a priori* state estimate $\mathbf{x}_{k|k-1}$ to obtain the *a posteriori* state estimate $\mathbf{x}_{k|k}$ [15].

A. Application of the Kalman filter to line tracking

The scribbled line stroke may be modeled by a pen moving along a trajectory in a piecewise linear manner such that the position of the pen defined by its (x, y) co-ordinates at any instant k may be written as

$$\begin{aligned} x_{k+1} &= x_k + \Delta x_k \\ y_{k+1} &= y_k + \Delta y_k \end{aligned} \tag{8}$$

where $\Delta x_k = D\cos\theta_k$ $\Delta y_k = D\sin\theta_k$ and θ_k is the line orientation obtained from the orientation estimates at the co-ordinates (x_k, y_k). The medial point obtained from the line profile taken perpendicular to the line path may be considered as the system measurements, obtaining an alternative measure for the line position of the tracking point.

In this way, the scribble may be considered as a process modeled by (8) having a state vector $\mathbf{x} = [x, y]^T$ and input $\mathbf{u} = [\Delta x, \Delta y]^T$. From the stroke-grouping response image I, measurements giving the position of the medial points may be obtained. By considering these measurements as the output of noisy sensors, the Kalman filter may be used to estimate the system states, and obtain better estimates of the medial axis. By comparing between (8) with (4) matrices A and B reduce to identity matrices of size 2×2. Furthermore, the measurements of the medial axis points are identical to the system states hence the measurement model H in (5) is also a 2×2 identity matrix.

Since the Kalman filter performs the state estimates using only the current and previous states and measurements, the filter is causal. This implies that initializing the filter from different starting points and using different directions, the Kalman filter would result in different state estimates. This is undesirable since the medial axis of the line drawing should not be affected by the tracking direction. This problem is present also in the Kalman filter implementations described in [12]–[14]. However, unlike these methods, the Kalman filter is not applied directly to the image but to a set of data points that approximate the medial axis and which are extracted from the drawing prior to its being processed by the Kalman filter. Thus, in this implementation, the causal nature of the Kalman filter may be removed by using the Smoothing Kalman filter. This estimates the smoothed distributions conditioned to all measurement data $\mathbf{z}_{1:N}$ where N is the total number of measurements. The smoothed distributions may be calculated from the Kalman filter results by using the backward recursions $k = n, n-1, \cdots 1$ on (9) [16].

$$\begin{aligned} J_k &= P_{k|k+1}A(P^{-1}_{k+1|k}) \\ X_{k|N} &= X_{k|k+1} + J_k(X_{k+1|N} - X_{k+1|k}) \\ P_{k|N} &= P_{k|k} + J_k(P_{k+1|N} - P_{k+1|k})J_k^T \end{aligned} \tag{9}$$

V. Evaluation and Results

In order to determine the performance of the proposed line-location algorithm, the algorithm was compared to the line location obtained by the SPV algorithm, using a number of test images containing a mixture of straight line strokes and curved strokes such as those shown in Fig. 4(a) and Fig. 4(c). The performance of the two algorithms was measured using the Pixel Recovery Index (PRI) defined in [17] which measures the location of the line strokes extracted by the line location algorithm with respect to the ground truth data. The proposed algorithm obtained a PRI of 87% whereas the SPV algorithm obtained a PRI of 82%, hence the lines extracted from the drawing using the proposed line tracking algorithm are better to the ground truth drawings than those extracted by the SPV algorithm. Furthermore, the proposed line tracking algorithm reduces the computational time of the SPV algorithm by 56%, hence reducing the time to obtain the stroke lines while improving the line quality.

The performance of the Kalman filter was also compared to smoothing obtained by simple moving spatial average filters. This was carried out by adding random positional noise having a uniform distribution in the range $[-5, 5]$ pixels to the medial lines, in order to simulate the contour noise. The error between the smoothed lines and the ground-truth lines can therefore be measured, obtaining a measure of the smoothing performance of the Kalman filter in comparison to the moving average filter.

The moving average filter requires the specification of a window size which determines the number of medial points over which smoothing is to take place. Larger window sizes are desirable in straight line regions as these would allow more smoothing of the medial lines. However, large window sizes would shift the medial lines to the centre of curvature of curved paths. Thus in moving average filtering, a compromise between smoothing and displacement of medial lines must be reached. For the medial lines used in this test, which consisted of medial lines extracted from scribbles similar to those shown in Fig. 4, a window size of 7 pixels was found to be more suitable, maximizing the error reduction to 37.85%.

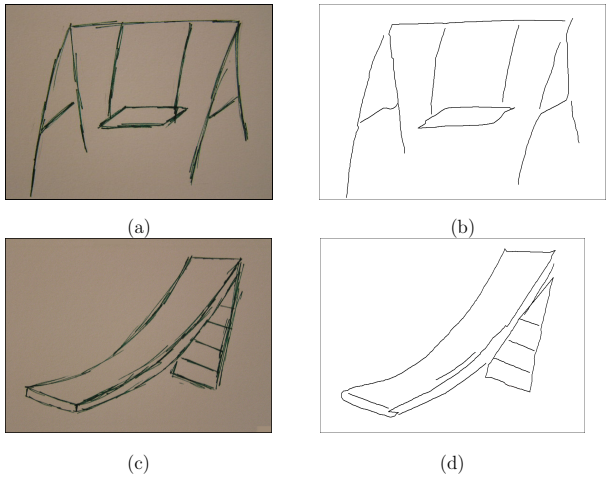

Fig. 4. *The results obtained after performing the proposed line tracking on a sample of scribble drawings*

The Kalman filter does not require the specification of a window size but performs the smoothing by considering the measurement and process noise. The forward estimation step of the Kalman filter reduces the error by a mean value of 22%, while the backward smoothing step causes a further reduction of 21.3% such that the Kalman filtering reduces the error in the medial points by a total of 43.3%. Hence, the Kalman filter gives better smoothing than the moving average filter without requiring user defined parameters.

Fig. 4 shows that the proposed line location algorithm gives well placed lines that give a faithful representation of the scribble. The proposed vectorization represents the scribbled drawing using few segments. This emerges since this line tracking procedure updates the line parameters at each tracking step, such that although the scribble is traced in a piecewise linear manner, the tracking step and direction may adapt to the line curvature, enabling continuous tracking.

VI. Conclusion

In this paper a sparse pixel based line location algorithm has been described. This line location algorithm differs from other sparse-pixel line tracking algorithms in that the sampling rate is determined adaptively, adjusting the tracking step to reflect the curvature of the drawing. Furthermore, Kalman filtering is used in order to adjust the position of the data points, using the estimates of the scan line midpoints and the piecewise tracking to obtain a better estimate of the stroke's medial points. The proposed line tracking algorithm also introduces perceptual selection of the line tracking direction at junction regions, allowing the tracking to follow the most salient path. This, together with the constant adjustments in the tracking direction, allow the proposed the line tracking algorithm to trace the scribbled drawing using fewer line strokes, hence considerably reducing the computational times required in comparison to the other sparse-pixel tracking algorithms while retaining the good quality line strokes obtained by these algorithms. The line location requires subsequent processing to link the extracted lines forming completely connected objects. This is typically carried out through the beautification step. In future, we propose to introduce further perceptual grouping techniques to this beautification step in order to select the most perceptually salient link formation.

References

[1] J. A. Landay and B. A. Myers, "Sketching Interfaces: Toward More Human Interface Design," *IEEE Computer*, vol. 34, no. 3, pp. 56–64, March 2001.

[2] 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 *Eurographics Workshop on Sketch-Based Interfaces and Modelling*, M. V. d. P. Eric Saund, Ed., 2007.

[3] A. Bartolo, K. P. Camilleri, P. J. Farrugia, and J. C. Borg, "A new sketch based interface using the gray-level co-occurrence matrix for perceptual simplification of paper based scribbles in proceedings of the," in *Eurographics Workshop on Sketch-Based Interfaces and Modelling*, S. Stahovich and Jorge, Eds., 2006, pp. 91 – 98.

[4] O. Hori, S. Tanigawa, and S. Shimotsuiji, "Precise Line Detection form an Engineering Drawing Using a Figure Fitting Method based on Contours and Skeletons," in *Proceedings of the 12th IAPR Int. Conf. on Computer Vision and Image Processing*, 1994, pp. 356 – 360.

[5] 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.

[6] X. Hilaire and K. Tombre, "Robust and Accurate Vectorization of Line Drawings," *IEEE Transactions on Pattern Analysis and Machine Interpretation*, vol. 28, no. 6, pp. 890–904, 2006.

[7] M. Röösli and G. Monagan, "Adding Geometric Constraints to the Vectorisation of Line Drawings," in *Graphics Recognition - Methods and Applications, GREC'95, Lecture Notes in Computer Science*, vol. 1072. Springer-Verlag, 1996, pp. 49–56.

[8] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone, "Stable and Robust Vectorization: How to Make the Right Choices," *Graphics Recognition - Recent Advances, GREC'99, Lecture Notes in Computer Science, Springer-Verlag*, vol. 1941, pp. 3–18, 2000.

[9] R. A. Katz and S. M. Pizer, "Untangling the Blum Medial Axis Transform," *International Journal of Computer Vision*, vol. 55, no. 2-3, pp. 139–153, November 2004.

[10] L. Wenyin and D. Dori, "A Survey of Non-thinning Based Vectorization Methods," in *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*. Springer-Verlag, 1998, pp. 230–241.

[11] A. Sha'ashua and S. Ullman, "Structural Saliency: The Detection of Globally Salient Structures using a Locally Connected Network," in *Second International Conference on Computer Vision*, Dec. 1988, pp. 321–327.

[12] J. Zhou, W. Bischof, and T. Caelli, "Road Tracking in Aerial Images Based on Human-Computer Interaction and Bayesian Filtering," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2006, pre-print version.

[13] P. Lallican and C. Viard-Gaudin, "A Kalman Approach for Stroke Order Recovering from Off-line Handwriting," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 1997, pp. 519 – 522.

[14] A. Lemaitre and J. Camillerapp, "Text Line Extraction in Handwritten Document with Kalman Filter Applied on Low Resolution Images," in *Proceedings of the Second International Conference on Document Analysis for Libraries DIAL'06*, 2006.

[15] P. S. Maybeck, *Stochastic Models, Estimation and Control*. Academic Press Inc. (London) Ltd., 1979, vol. 1.

[16] S. Sarkka, A. Vehtari, and J. Lampinen, "Time Series Prediction by Kalman Smoother with Cross-validated Noise Density," *Proceedings. 2004 IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1653–1657, 2004.

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