# Improvements to the TCVD method to segment hand-drawn sketches

F. Albert[a], N. Aleixos[a*]

[a] *Inter-University Research Institute for Bioengineering and Human Centered Technology, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, España {fraalgi1,naleixos@dig.upv.es}*

[*] *Corresponding author: naleixos@dig.upv.es, Tel.: +34 96 387 95 14, Fax: +34 96 387 75 19*

**Abstract**— Tangent and Corner Vertices Detection (TCVD) is a method to detect corner vertices and tangent points in sketches using parametric cubic curves approximation, which is capable to detect corners with a high accuracy and a very low false positive rate, and also to detect tangent points far above other methods in literature. In this article, we present several improvements to TCVD method in order to establish mathematical conditions to detect corners and make the obtaining of curves independent from the scale, what increases the success ratio in transitions between lines and curves. The new conditions for obtaining corners use the radius as the inverse of the curvature, and the second derivative of the curvature. For the detection of curves, a new descriptor is presented, avoiding the parameters dependent of scale used in TCVD method.

In order to obtain the performance of the implemented improvements, several tests have been carried out using a dataset which contains sketches more complex than those used for validation of TCVD algorithm (sketches with more curves and tangent points and sketches of different sizes). For corners detection, the accuracy obtained was pretty similar to that obtained with the previous TCVD, however, for curves and tangent points detection the accuracy increases significantly.

**Index Terms**— Corner vertices detection, tangent points detection, sketch recognition, stroke segmentation, curvature functions, natural interfaces

– – – – – – – – – – Φ – – – – – – – – – –

## 1 INTRODUCTION

In the field of industrial design, the first stages of design are very important. It is in here where ideas are expressed freely by means of basically hand drawn sketches, and later, these ideas are converted into 3D prototypes that finally are manufactured as products. Actually, sketching is a well established part of the engineering culture, but the current available tools for Computer Aided Sketching (CAS) supported by commercial CAD (Computer Aided Design) applications are not yet as usable as traditional paper-and-pencil, owing to the lack of many necessary functionalities and flexibility [1].

Thus, once conceptual design has been completed, sketches are converted into 3D models in CAD applications, but not directly, that is, in commercial CAD applications, models are created from perfect outlined sketches, not from hand-drawn sketches, and therefore all the effort in the first stages of design cannot be exploited. This happens mainly because commercial CAD applications do not provide sketching tools (CAS), and in some CAD applications that do, the functionality is very limited, as in the case of ProEngineer (from Parametrics Technology Corporation), SIEMENS NX 10 (the formerly Unigraphics, from Siemens Industry Software) or Catia (from Dassault Systèmes). In some of them every time the user sketches something, a pop-up menu comes up to solve ambiguities, asking the user to choose the correct option since the system has not been capable to segment or interpret the intended sketched shape. Other application that provides a hand-drawn sketch user interface is AutoCAD 360, from Autodesk, developed for smartphones, tablets or desktop devices, which presents a poor recognition system and a worse user intent design capture when introducing sketches. In here, many evident curves in sketches are approximated to straight lines instead of arcs and most of the times the final shape is completely different to the intended one. Besides, none of the applications mentioned above discusses the intended tangency in the sketches, but we have to keep in mind that most products have smooth transitions in their outline, so it is essential to solve the task of finding tangent points with accuracy.

The limitations or lacks on the CAS tools provided are related to the poor segmentation of the sketch drawn, that has a lot to do with the efficiency for finding corners and tangent points, feature that is generalised for nearly any kind of commercial CAD software that supports sketching. And it is this feature which makes, consequently, conceptual design stages are completely detached from the rest of the stages of design. Thus, to achieve interfaces that support natural human-computer interaction it is necessary to develop intelligent techniques for finding corners and tangent points in hand-drawn sketches and so for the automatic recognition of sketches that allows users drawing as they naturally would without any constraints, like introducing the sketches in a particular order or requiring a previous training by the user to learn a set of specified symbols or shapes [2].

Thereby, these techniques have to deal with the problem of drawing complex shapes in a single stroke, being a stroke (or sketch) a continuous sequence of points between the two pen-down and pen-up events. In order to support this feature, it is necessary to split the stroke/sketch into its constituent primitives, what involves the development of techniques capable of finding corners and tangent points in the stroke. Once the corners and tangent points are found, the stretches between

them could be approximated to primitives' straight lines or curves; hence, user intent design could be captured maintaining the tangency between lines and curves or between curves, as applicable. This procedure is also known as segmentation process.

Regarding to the segmentation process, the TCVD method was presented by Albert et al. [3] as the best and more accurate current method to find corners and tangent points in hand-drawn sketches compared to others in literature. The main advantage of TCVD method was the use of the approximation of the sketch to parametric cubic curves, allowing finding tangent points with high accuracy, what makes TCVD a good method to find corners and tangent points in hand-drawn sketches with an All-or-Nothing Accuracy of 96% and 92% for corners and tangent points, respectively. Still, TCVD has some limitations. One of the most important drawbacks is the scale dependence, making it difficult to distinguish between large radius curves and straight lines. Other lack found in TCVD is the use of a threshold when finding corners. This threshold (based on the radius of curvature at neighbouring points) was defined heuristically. The main objective of this work is to solve the drawbacks found in TCVD method, to which end two improvements have been implemented. The first one has consisted in establishing a discriminatory function (based on the derivatives of the curvature) instead of a heuristic threshold when finding corners; and the second one has consisted in introducing some key parameters independent of the scale when finding curves.

Regarding to other related works in literature, a revision of most relevant methods to understand the sketched user input has been already done by Albert et al. [3], concluding that the detection of the tangent points in the stroke is a key feature, because the geometry of sketches has to be approximated to their corresponding primitives in order to create the out-lined section to later generate accurate 3D models that capture the user intent design. In engineering design, most of the models have tangent transitions between planar-curved surfaces or curved-curved surfaces, what makes essential the detection of the designer intention in the sketches, that is, the finding of tangent points, and although some interesting works have been carried out, the segmentation of sketched shapes still remains unsolved.

Among these methods is that of Yu [4] and Hse et al. [5] which used segmentation and primitive approximation to find points between straight lines and curves. The work presented by Paulson and T. Hammond in [6] also split the stroke and then recognised its primitives, and the works presented by Alvarado and Davis [7] and Hammond and Davis [8] recombined later the primitives using some geometrical rules. Sarkar et al. [9] first segmented the strokes before facing the corner finding using

genetic algorithms to fit digital curves to lines and arcs. Zhang et al. [10] extracted the primitives by a connected segment performing a growing process from a seed-segment and utilised relationships between them to refine the control parameters. Nguyen and Debled-Rennesson [11] applied two methods, the method based on a fixed parameter (the width of considered maximal blurred segments) and the method based on a multi-width approach without thresholds, obtaining only corners since curves were always fitted to lines.

More important methods were those presented in the following works. In [12] Wolin et al. developed the ShortStraw algorithm to find corners in strokes. This algorithm was found to be highly accurate in both total correct corners and all-or-nothing corner accuracy, but the main problem was that strokes only contained straight lines. An improvement of this algorithm was presented by Xiong and La Viola [13], called IStraw, which overcame some limitations and attempted to reduce the lacks maintaining its computational complexity and extending the ShortStraw to deal with strokes containing curves. This algorithm obtained significant improvements in all-or-nothing corner accuracy compared to ShortStraw, but did not consider tangent points in strokes. Although Pu and Gur [14] used mathematical curves to approximate the stroke, their method presented two important inconveniences. First it did not distinguish between corner vertices and tangent points, and second it performed a high post-process of refinement, remaining the number of false positive detections very high.

Other technique, that Herold and Stahovich [15] called ClassySeg, begun by identifying a set of candidate segment windows, each comprising a curvature maximum and its neighbouring points. Then, features were computed for each point in each window based on curvature and other geometric properties, most of them adapted from numerous prior segmentation approaches. These features were used to train a statistical classifier to identify which candidate windows contained true segment points. Although this approach is more oriented to classify symbols in different domains (being a very tedious and complex method since many techniques have to be combined and also a non intuitive previous training has to be made for each domain), authors also tested this technique to segment sketches on a data set of ten pen strokes including curves, but again only corner points were present, that is, there was no tangency in the sketches.

But as usual, all previous reviewed methods did not deal with tangent points. In sum, many research works just deal with polyhedral models, or reconstruct 3D models from simple sketches of isolated lines or arcs (like in [6]), because the main lack of obtaining curved models from sketches,

necessary in most of engineering models, is that segmentation algorithms are not capable of detecting smooth transitions from straight lines to curves or between curves, and those that try to detect this kind of transitions are not robust, mainly due to the bad results obtained or to the high number of false positives they reach.

The TCVD method seems to be the most advanced and accurate method to segment hand-drawn sketches finding corners and tangent points in sketches including curves, contrary what said in [15]. The work presented here intends to improve this method, solving the two lacks found: the use of a heuristic threshold when finding corners and the scale dependence, making it difficult to distinguish between large radius curves and straight lines.

This article is organised as follows: In section 2 a brief description of the TCVD method is done. Section 3 explains in detail the stages of the TCVD method that have been revised and improved. Section 4 describes the experimental work carried out comparing the results obtained from the original TCVD to those obtained after improvements proposed. Finally, section 5 and 6 reports the conclusions and further work, respectively.

## 2 THE TCVD METHOD

The TCVD method [3] uses the radius, as the inverse of the curvature, as a way to find corners in hand-drawn strokes. The radius allows setting more intuitively the value of thresholds (a small threshold for corner vertices and a larger one for arcs or curves) than the curvature, since for example, we understand better a radius value of 100 pixels than a value of 0.01 for curvature. To distinguish between straight lines and curves, as discrete radius is not stable in hand-drawn sketches, TCVD method obtained a piecewise parametric curve approximation of the stroke, and calculated the radius function from the mathematical expressions of the parametric curves in order to segment the stroke, getting the entities in the stroke and keeping the points of tangency between them.

This method has six differentiated stages. A brief description of each of them is laid out below:

1. Computing the discrete radius function. Where first, the digitised stroke is resampled and the noise removed. Second, the tangent vector at each point is calculated from differences between coordinates of its neighbouring points, and the curvature at each point is obtained from differences of the tangent angle between neighbouring points. Finally, the radius at each point is calculated.
2. Corner vertices detection. The corner vertices are located at points with local minima of

the radius, and with a radius sufficiently smaller than the points of its environment.

3. Piecewise parametric curves approximation. The resampled points between pairs of corners are approximated by means of piece-wise cubic curves until the distance from every approximated point to the resampled point does not exceed a threshold. When the distance is greater, the sequence of points is halved and the process is subsequently applied to both sides, forcing two curves to have the same tangent at the common point (the central point when the previous sequence is divided).
4. Computing the analytic radius function. The tangent vector at each point is calculated from derivative of piece-wise cubic curves, then, the curvature at each point is obtained from derivative of the tangent angle, and finally the radius at each point is calculated.
5. Straight lines and curves detection. A point lies on a candidate curve if the radius at that point is less than a threshold, otherwise the point belongs to a straight line. Therefore, a sequence of consecutive candidate curve points is definitely a curve if the distance between the points and the straight line from first sequence point to last one is greater than a threshold.
6. Tangent points detection. The tangent points are located at points of transition from straight lines to curves (and vice versa), and at points of transition between curves of radius with different sign, if corner vertices are not previously placed in such transitions.

## 3 REVISION OF TCVD METHOD. IMPROVEMENTS

In this work, we have implemented two improvements to the TCVD method. The first improvement is related to the corner vertices detection stage (Stage 2 of TCVD method, See Fig. 1) that consists on the establishment of a discriminatory function to clearly differentiate corner vertices avoiding further refinements. This function is based on the radius function (as the inverse of the curvature) and on the second derivative of the curvature. The calculation of the first and second derivative of the curvature (C′ and C″ respectively) has been carried out at the end of Stage 1. The second improvement is related to the lines and curves detection stage (Stage 4), where the main objective is to make the obtaining of curves independent of the scale.

The modifications proposed to TCVD method are represented in the flowchart of Figure 1. The following sections describe in deep these improvements.

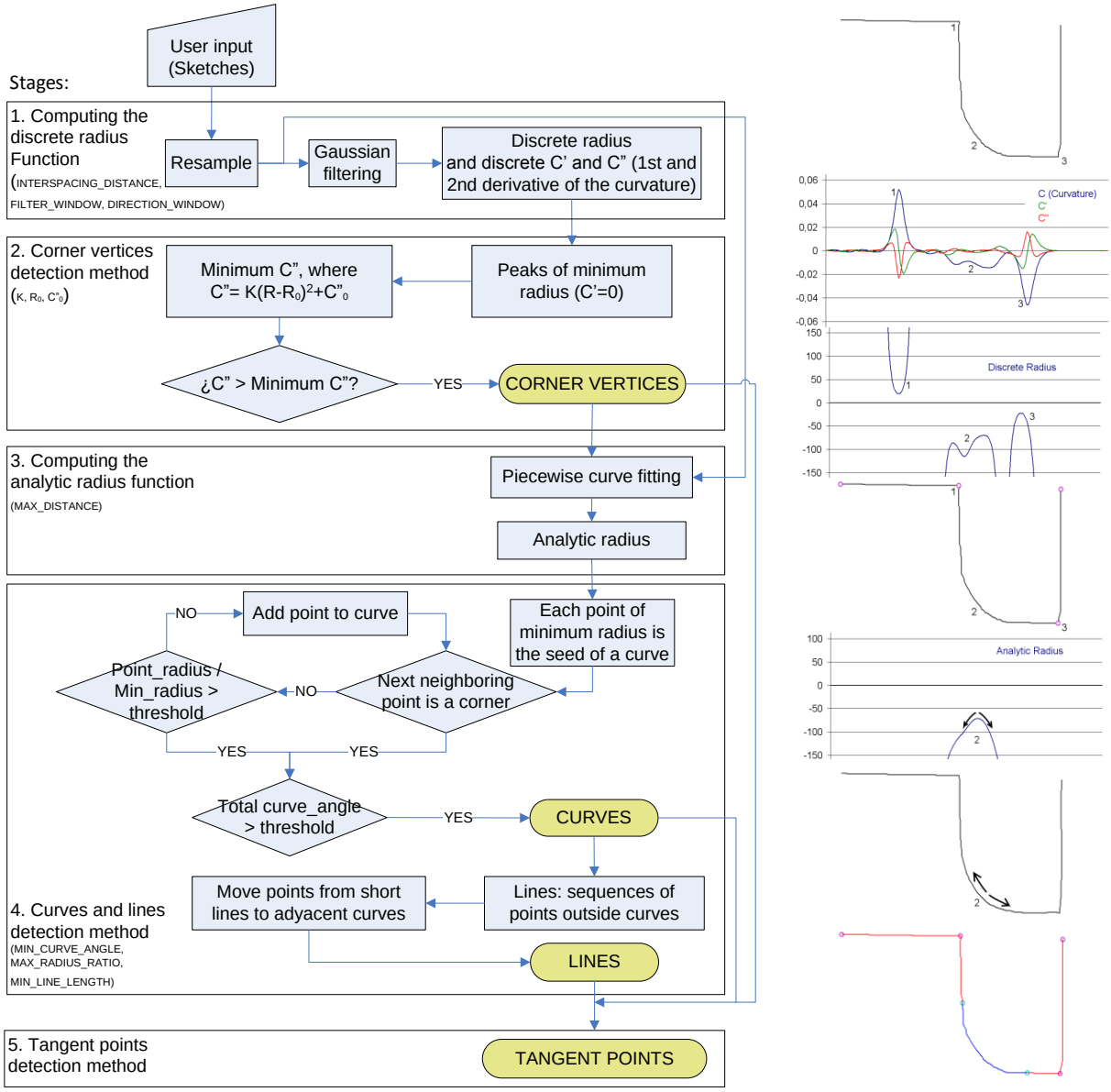



Figure 1. Flowchart of the modifications proposed to TCVD method. Only stages 1, 2 and 4 have been affected. The parameters used at each stage appear between parentheses after the number and the name of the stage

## 3.1 New discriminatory function for corner vertices

The main objective is to establish a new scientific discriminatory function instead of the heuristics used in TCVD method to detect corners.

The functions involved in the obtaining of corners are the first and second derivatives of the curvature and the radius. The curvature measures the changes in the direction of the tangent vector. When faster it changes the bigger the curvature is (in absolute value). In the computer display, where

the reference system is levorotatory, the curvature is positive when the turn is clockwise. The radio is the inverse of the curvature, where an infinite radius will correspond to a curvature of value zero (straight line).

The first derivative of the curvature is the slope of the curvature. The zero crosses correspond to points where the slope changes its sign, that is, to the local minimum and maximum values of the curvature. The second derivative measures the changes of the curvature. Then, when faster the curvature changes the bigger the second derivative is. Thus, for curves of constant curvature, although it is high, the second derivative will be zero and for corners it will be a high value.

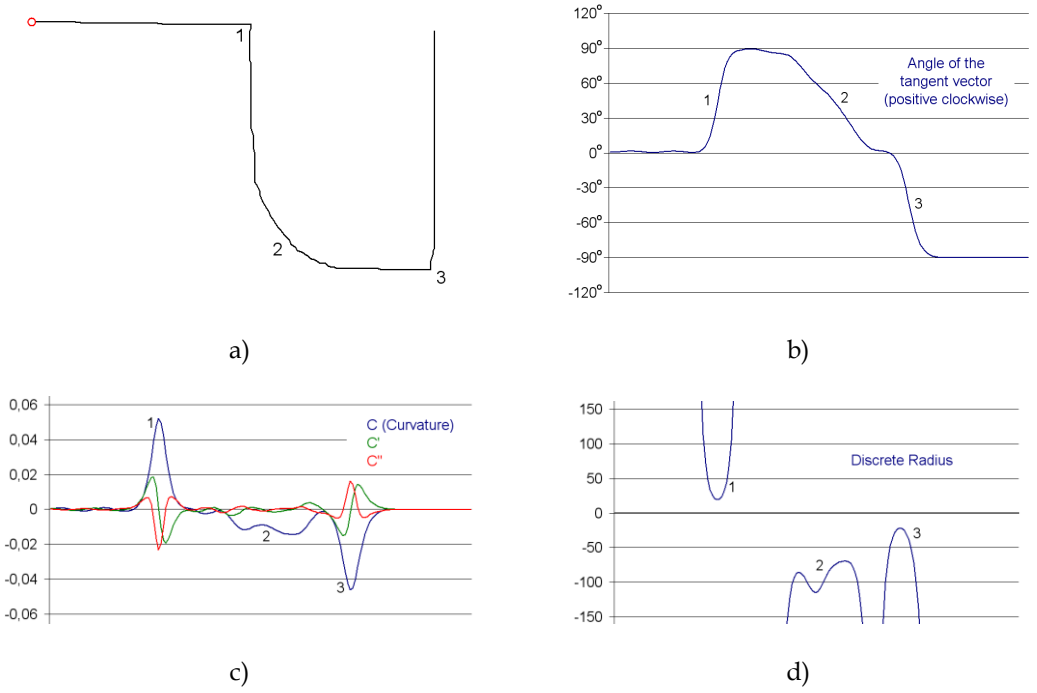



Figure 2. Functions for the stroke sample of Figure 1: a) Original stroke; b) Angle values; c) Curvature (*C*), first derivative of the curvature (*C'*) and second derivative of the curvature (*C''*) superimposed; and d) Radius

For this stroke, the functions obtained after filtering are shown in Fig. 2. As we can see in Fig. 2b, the angle clearly shows sharp changes of 90 º (labelled as '1' and '3') which correspond to the two corners in the stroke. Also a smooth transition of 90º between the two corners (labelled as '2') corresponds to the curve in the stroke. In the curvature function (Fig. 2c) we can observe the two maximum values (in absolute value) corresponding to the corners, and between them, we can see the intermediate values of the curvature corresponding to the curve. The zero crosses in the derivative of the curvature (C') correspond to maximum values of the curvature function, and the maximum values in the second derivative of the curvature (C'') correspond to stroke points where the curvature

changes fast (corners) whereas for the curve (almost constant curvature) the value of C″ can be considered zero.

The graphs of Figure 3 show a set of 3561 values of the second derivative of the curvature C″ (Y axis) with respect the radius values (X axis), both in absolute values, obtained from the zero crosses of the first derivative of curvature of a set of 200 strokes (belonging to the training set used). Such maximums are distributed in 469 corners (blue colour) and 3092 non corners (red colour). Only conflictive radius values are shown (<=100 pixels).

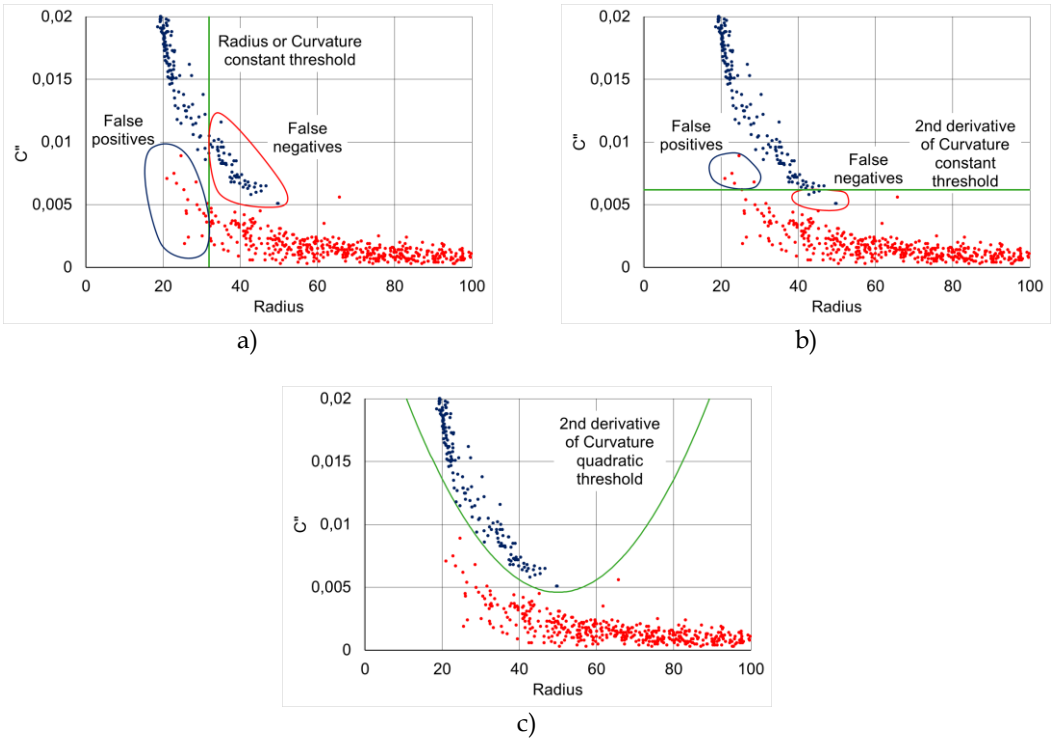
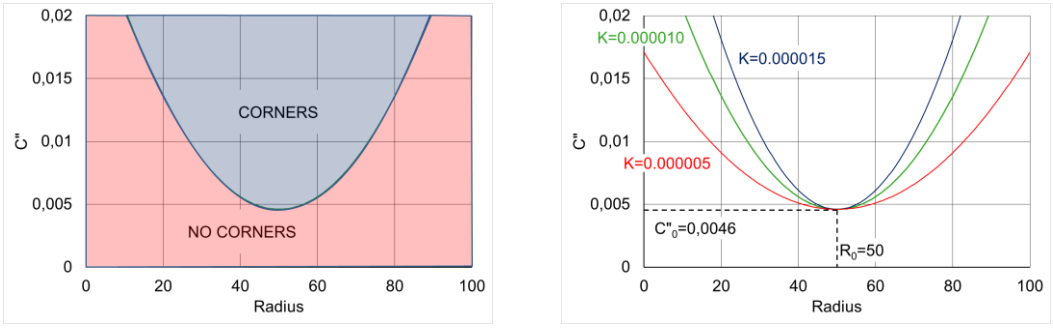



Figure 3. C″ values (Y axis) respect to radius values (X axis) of corners (blue) and non corners (red) including different thresholds for the training set of 200 strokes: a) Constant threshold for radius; b) Constant threshold for C″; and c) Quadratic threshold for C″

In order to separate corner vertices from non corner vertices, several functions can be proposed. In Fig. 3a, a threshold for radius has been chosen (equivalent to a curvature threshold, which is common in many sketch recognition methods) remaining a large number of false positives and false negatives of corner vertices, thus requiring ulterior refinement process of the corners detection. In Fig. 3b, a threshold for C″ is chosen, and the number of false positive and false negative corners found is lower than in Fig. 3a, so we can say that C″ is more appropriate than C to separate corner from non corner vertices. In Fig. 3c, a quadratic function for C″ is chosen, being the results much better. The proposed discriminatory function of C″ depending on the radius results as follows:

$$C''_{threshold}(R) = K \cdot (|R| - R_0)^2 + C''_0 \quad (1)$$

So that a vertex found will be a corner if its $|C''|$ value is upper than the $C''_{threshold}$ in (1). Where $K$ is a constant, $C''_0$ is the lowest value found for the second derivative of the curvature and $R_0$ is the radius value for $C''_0$ (see Fig. 4).

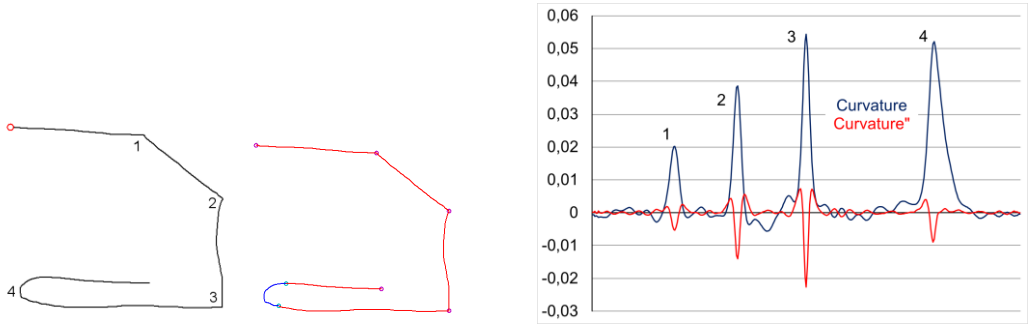



a) b)

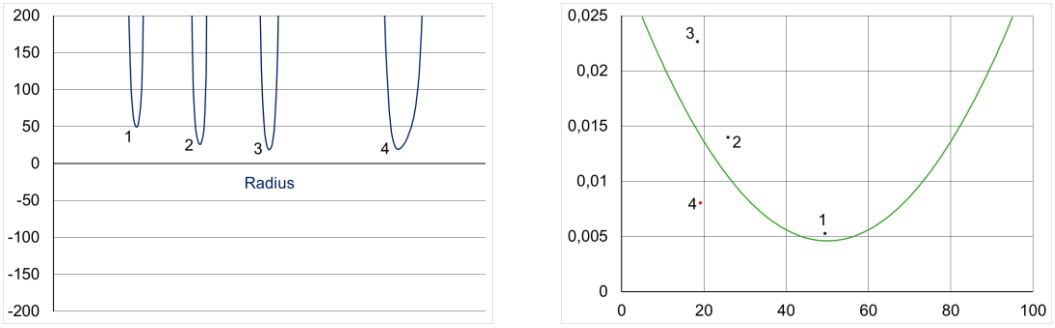
Figure 4. a) Zones where the corners (blue) and non corners (red) are positioned; and b) Several shapes of discriminatory functions depending of the constant $K$ (opening) and the values of $C''_0$ and $R_0$ as the coordinates for positioning the function

In our case, the $K$, $R_0$ and $C''_0$ values have been obtained using an implementation of the optimisation algorithm Simulated Annealing [16] for the data set proposed of 3561 values. The values obtained after the optimisation process are shown in Table 3.

Figure 5 shows the result, applying this discriminatory function, for a sample of a stroke that contains three corner vertices and one curve of a quite small radius. This sample has been chosen to show how this threshold is able to distinguish corners from curves with small radius.




a) b)




c) d)

Figure 5. a) Stroke sample and its segmentation; b) Curvature (*C*) and second derivative of the curvature (*C″*) overlapped; c) Radius; and d) |C″| values (Y axis) respect to the radius values (X axis) for points of minimum radius (dots) and $C''_{threshold}$ (green curve)

As we can see from Fig. 5c, the radius of the real curve of the stroke (last sharpen peak) is lower (and wider) than those from the first two corners. In Fig. 5d are represented the values of C'' for the minimums of radius of the stroke. Blue dots correspond to corners (above the threshold) and the red dot corresponds to a real curve (below the threshold). The values obtained for the curvature and radius functions and for *C″* threshold are depicted in Table 1. The corresponding segmented stroke appears in Fig. 5a, where the straight lines are represented in red and the curve in blue. The three intermediate corners found (and the two ends of the stroke) are represented in magenta colour and the two tangent points that limit the curve are represented in cyan.

Table 1. Results obtained for the minimum values of *R* for the stroke of Fig. 5a

| Point | R | \|C″\| | | $C''_{threshold}$ | Description |
|---|---|---|---|---|---|
| 1 | 49.4 | 0.0053 | > | 0.0046 | Corner 45º |
| 2 | 25.9 | 0.0130 | > | 0.0054 | Corner 45º |
| 3 | 18.4 | 0.0227 | > | 0.0133 | Corner 90º |
| 4 | 19.2 | 0.0080 | < | 0.0135 | Curve |

## 3.2 CURVES DETECTION

This improvement has been introduced in order to make the curve detection independent of the scale.

In the earlier TCVD method, the curve detection algorithm determined that a point lied on a curve candidate if the radius at that point was less than a threshold (MAX_CURVE_RADIUS), otherwise the point belonged to a straight line. Therefore, a sequence of consecutive curve points was definitely a curve if the distance between the points and the straight line from the first sequence point to the last one was greater than a threshold (MIN_DIST_CA). This way of detecting curves leads to a high dependence on the scale, causing misdetections when analysing the same shapes having different

sizes.

To avoid the problem of size requirement, the parameters proposed are: The *Angle* swept by the tangent vector to the stroke; and the *Ratio* between the major and minor radius values within the curve. In the case of the angle, this parameter informs about a minimum value from which the curve is appreciable. In the case of the ratio, the value is a maximum, because when the entity to check is a straight line, this value increases considerably. The values of these parameters were determined by using the Simulated Annealing technique mentioned previously (see Table 3).

The algorithm proposed (in pseudo-code) for curves detection is shown in Fig. 6.

*#Find minimum local values of radius between each pair of corners found. Each minimum is considered the initial point of a candidate for a curve, and will be a CURVE SEED.*

*#For each CURVE SEED, do:*

*#Grow curve on both sides, adding neighbouring points to the curve, as three conditions are met:*

*#The radius ratio of the current added point divided by the radius ratio of the curve seed is below the fixed parameter MAX_RADIUS_RATIO*

*#The current added point is not a corner (a corner has not been reached)*

*#The radius of the current added point keeps its sign (it is not an inflexion point)*

*#If two candidate for a curves, from different starting curve seeds and with the radius of the same sign, join, then both curves belonging to different curve seeds will be chained*

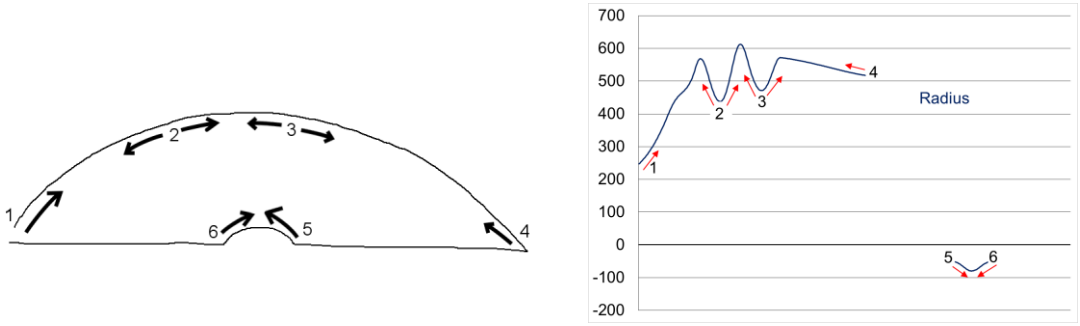
*#When no more points are added (growing process ends), the sequence of points candidate to a curve, will be definitely considered a curve if the absolute value of the angle swept by the tangent vector to the stroke between both ends of the sequence is above the parameter MIN_CURVE_ANGLE*

Figure 6. Algorithm proposed for curves detection

To illustrate better how the curve detection algorithm operates independent of the scale, the stroke of Fig. 7a has been chosen. This stroke consists of two straight lines and two curves with very different radius but with similar angles. This sample will prove that neither the arc length nor the radius differences between the two curves are important to segment both as curves.

Figure 7 shows, both in the stroke (Fig. 7a) and in the graph of the radius (Fig. 7b), the growing process of the curves, starting from the seeds or points with minimum radius. The values that do not appear in the graph of the radius correspond to absolute values extremely high (curvature almost zero) of the straight lines. The bigger curve presents radius values about 500 pixels, whereas the small curve presents radius values about 65 pixels (in absolute values). The curves from the first four

seeds join and the curves from the last two seeds also.




a) b)

Figure 7. a) Original stroke; and b) Radius values. Both figures show the location of the seeds in the original stroke and in the radius values, respectively

Figure 8a shows the angle of the tangent vector of the previous stroke, which rotates in each curve 90º. Figure 8b shows the segmented stroke, resulting in two curves and two straight lines separated by corners. Table 2 shows the features for both curves found in the stroke.


a) b)

Figure 8. a) Tangent vector angle values for the stroke of Figure 7a; and b) Segmented stroke with two curves (in blue) and two lines (in red) separated by corners

Table 2. Extracted features for curves of the stroke of Figure 8

| Curves | Radius range | $R_{major}/R_{minor}$ | Curve angle | Description |
|---|---|---|---|---|
| Big | [370,600] | 1.62 (<5) | 105º (>30º) | Real curve |
| Small | [-55,-79] | 1.44 (<5) | 109º (>30º) | Real curve |

## 4 EXPERIMENTAL WORK AND RESULTS

In order to evaluate the improvements made to TCVD, we have used a data set of 20 different shapes, with 9 shapes belonging to the data set established in [3] and an additional data set of 11 different shapes containing more oriented engineering sketches with curves and tangencies (see Fig. 9 for the outlined models of the different sketches used). For now, all the strokes are considered open shapes. The dataset contains 54 corners, 84 straight lines, 30 curves and 39 tangent points (including

five inflection points).

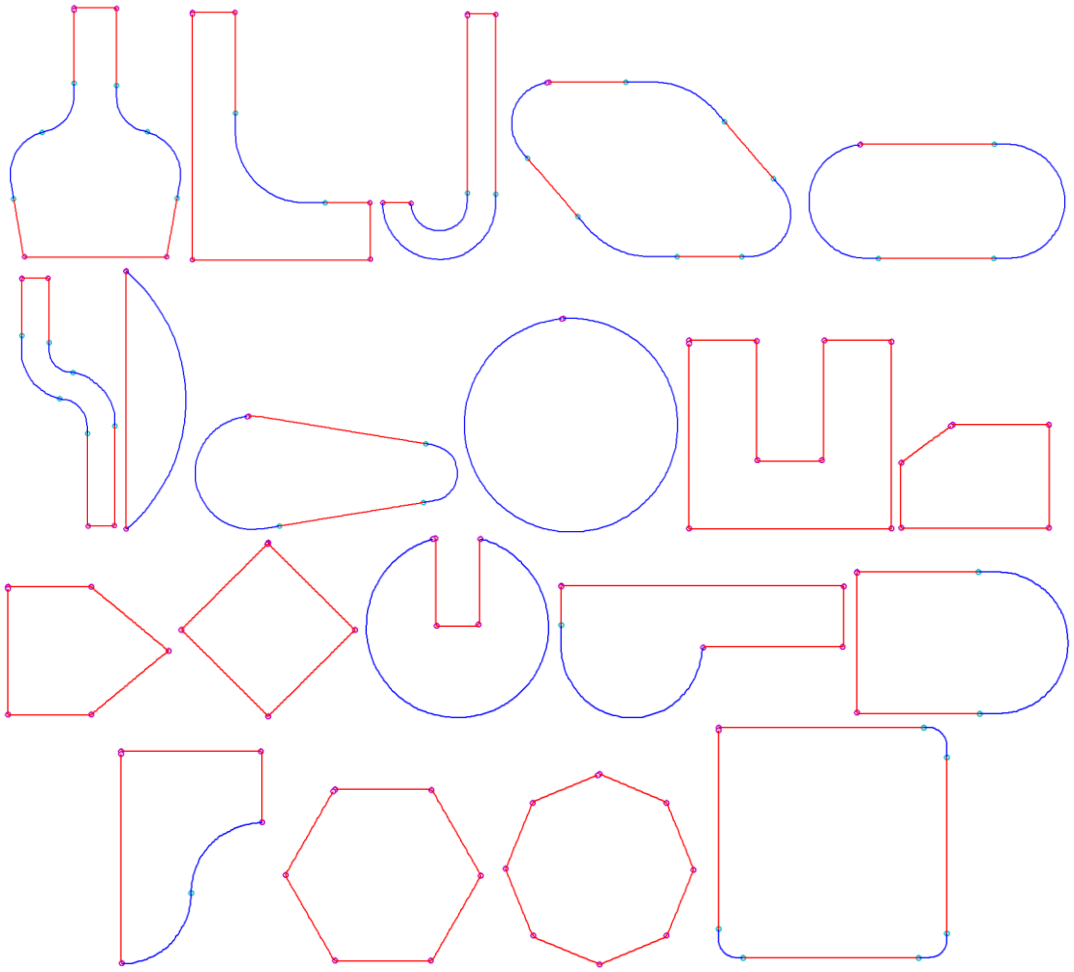


Figure 9. Strokes with straight lines (drawn in red), curves (drawn in blue), corner vertices (drawn in magenta) and tangent points (drawn in blue cyan)

We collected data from 10 different users, and each user drew 5 times each shape, making a total of 1000 strokes. Each user drew each shape with different sizes. If a stroke was drawn wrong, that is, at first glance it did not correspond with the model, it was removed and redrawn. The training data-set is formed with a stroke of each shape for each user (200 strokes) and the rest (800 strokes) are left for the test dataset. The sketched shapes, Data-set-train.rar and Data-set-test.rar are available in the Downloads section of the following address: http://www.cofilab.com.

And in [3], the parameters used for this approach were optimised by means of Simulated Annealing algorithm in order to achieve best results, which is explained in detail in a previous work [17]. This process is based on the Simulated Annealing technique, which allows us tuning the parameters to improve the segmentation results, and where the parameters have been formulated as an optimisation problem where the function cost is expressed as the number of errors in the segmentation of the training set.

The temporal cost is about 8 ms per shape using a computer with an Intel i5-4460 3.20 GHz and the operating system Windows 8.1, being about 90% for the Stage 3 (piecewise parametric curves approximation, see Fig. 1).

The optimised parameters are in table 3, and these values are directly set in the TCVD algorithm.

Table 3. Parameters and their default values of the improved TCVD algorithm

| TCVD Parameters | Description | Value from SA |
|---|---|---|
| INTERSPACING_DISTANCE | Interspacing distance between resampled points | 3.5 |
| FILTER_WINDOW | Window size for Gaussian filter | 8 |
| DIRECTION_WINDOW | Window size for stroke direction calculation | 4 |
| K | Constant of the quadratic threshold to determine which points are corners | 0.00001 |
| $C''_0$ | Lowest value of the quadratic threshold to determine which points are corners | 0.0046 |
| $R_0$ | Radius value for $C''_0$ in the quadratic threshold to determine which points are corners | 50 |
| MAX_DISTANCE | Maximum distance between resampled points and parametric curve approximation | 3.0 |
| MAX_RADIUS_RATIO | Maximum ratio between the maximum and the minimum radius along a curve | 5.3 |
| MIN_CURVE_ANGLE | Minimum angle swept by the tangent vector angle along a curve | $34.7^o$ |
| MIN_LINE_LENGTH | Minimum length of a straight line | 41.8 |

The following tables show the results for both original TCVD algorithm and the improved TCVD, called from now on TCVD-R (TCVD-Revised) to distinguish them.

The tables 4-7 show the results separately for corners, lines, curves and tangent points, although the false positives (incorrect entities) and false negatives (entities not found) of different entities are related, as seen in Figure 10. The results are expressed in the measures “Accuracy” and “False Positive Rate”. The first measure is equal to the number of correct entities found divided by the total number of entities; and the second one is equal to the number of incorrect entities found divided by the total number of entities. The 20 shapes of the dataset contain 54 corners, 84 straight lines, 30 curves and 39 tangent points, and the test dataset contains 40 strokes of each shape (800 strokes), that is: 2160 corners, 3360 straight lines, 1200 curves and 1560 tangent points.

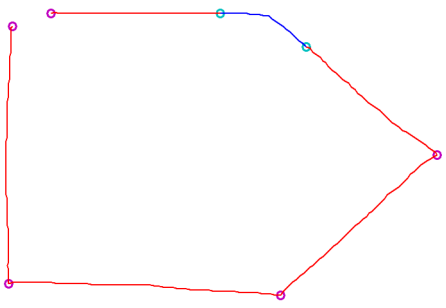
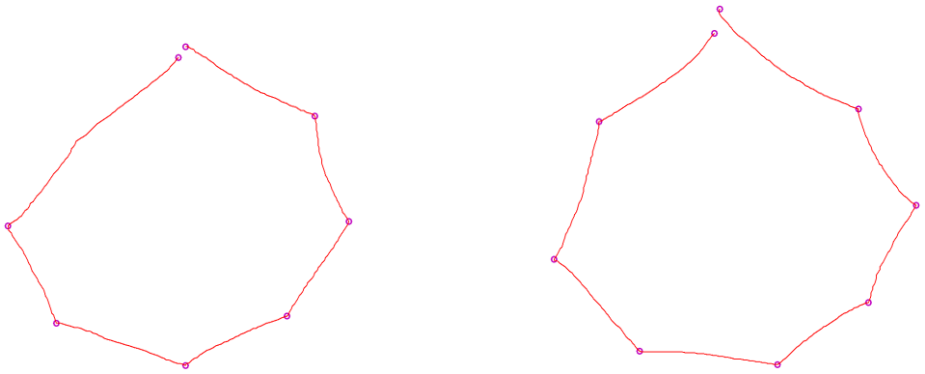


Figure 10. Corner not found (false negative) which also generates one false curve and two false tangent points

Table 4. Accuracy results for corner vertices

| | TCVD | TCVD-R | Improvement |
|---|---|---|---|
| False Positives Corners | 4 | 0 | |
| False Negatives Corners | 24 | 22 | |
| Correct Corners Found (Total - False Negatives) | 2136 | 2138 | |
| Total Corners (54x40) | 2160 | 2160 | |
| Correct Corners Accuracy (Correct / Total) | **98.9%** | **99.0%** | **0.1%** |
| False Positive Rate (False Positives / Total) | **0.2%** | **0%** | **0.2%** |

The fact that there is no false positive corners in TCVD-R means that the tuning of parameters is conservative, but it is the set of parameters that best performed. The majority of errors (false negatives) occurs in the octagons, which are difficult to draw by hand and always have some ambiguous corners (Fig. 11a). In these cases, it is best to bend/emphasize (slightly) the sides of the polygon to highlight the corners (Fig 11b). Another important feature to emphasize the corners is to stop to change the direction at each of them to avoid making a small curve. Therefore, the speed is a feature that is often used in the literature; however, the speed may be slow without performing a corner, whereby it is not a very reliable feature.



a) b)

Figure 11. Ambiguous corners not found (false negative) and corners emphasised in octagons

Table 5. Accuracy results for straight lines

| | TCVD | TCVD-R | Improvement |
|---|---|---|---|
| False Positives Straight Lines | 50 | 12 | |
| False Negatives Straight Lines | 14 | 15 | |
| Correct Straight Lines Found (Total - False Negatives) | 3346 | 3345 | |
| Total Straight Lines (84x40) | 3360 | 3360 | |
| Correct Straight Lines Accuracy | **99.6%** | **99.6%** | **0.0%** |
| False Positive Rate (False Positives / Total) | **1.5%** | **0.4%** | **1.1%** |

Table 6. Accuracy results for curves

| | TCVD | TCVD-R | Improvement |
|---|---|---|---|

| | | | |
|---|---|---|---|
| False Positives Curves | 25 | 15 | |
| False Negatives Curves | 3 | 0 | |
| Correct Curves Found (Total - False Negatives) | 1197 | 1200 | |
| Total Curves (30x40) | 1200 | 1200 | |
| Correct Curves Accuracy | **99.8%** | **100%** | **0.2%** |
| False Positive Rate (False Positives / Total) | **2.1%** | **1.3%** | **0.8%** |

Table 7. Accuracy results for tangent points

| | **TCVD** | **TCVD-R** | **Improvement** |
|---|---|---|---|
| False Positives Tangent Points | 96 | 40 | |
| False Negatives Tangent Points | 16 | 16 | |
| Correct Tangent Points Found (Total - False Negatives) | 1544 | 1544 | |
| Total Tangent Points (39x40) | 1560 | 1560 | |
| Correct Tangent Points Accuracy | **99.0%** | **99.0%** | **0%** |
| False Positive Rate (False Positives / Total) | **6.2%** | **2.6%** | **3.6%** |

The above tables show that improvements do not seem very large. This is for two reasons: the results of the original TCVD were already very good, and there are many entities that are easy to recognise in which TCVD never fails. The results in corners finding are very similar because the heuristic was very good (it also valuates the variation in radius values) and the size of the figure/shape does not affect the corners. Moreover, the improvement is greater in finding straight lines, curves and tangent points because the TCVD-R is more robust against the scale and the test dataset contains shapes of different sizes.

The table 8 shows the "All-or-Nothing Accuracy", which measures the number of correctly segmented strokes (completely) divided by the total number of strokes. As in the previous tables, we see that the larger improvement is obtained in finding straight lines, curves and tangent points. The "All-or-Nothing Accuracy" measure is the most significant one because it means that the entire stroke is well recognised.

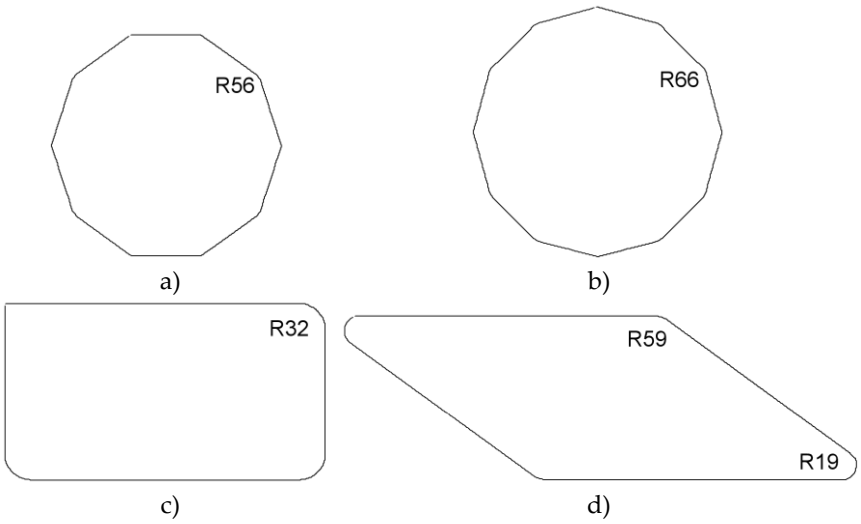
Table 8. All-or-nothing accuracy for full strokes

| | **TCVD** | **TCVD-R** | **Improvement** |
|---|---|---|---|
| Incorrect Strokes (only corners) | 25 | 20 | |
| Incorrect Strokes | 80 | 46 | |
| Correct Strokes (only corners) | 775 | 780 | |
| Correct Strokes | 720 | 754 | |
| Total Strokes | 800 | 800 | |
| All-or-Nothing Acc. (only corners) | **96.9%** | **97.5%** | **0.6%** |
| All-or-Nothing Acc. | **90.0%** | **94.3%** | **4.3%** |

In order to check how accurate is the TCVD-R vs. TCVD method, they have also been tested with the shapes of the dataset drawn with a commercial CAD application (from now on, CAD set). In this

394 case, the All-or-Nothing Accuracy is 100% using the same set of parameters of the Table 3.

395 To test the limits of the quadratic threshold function for corners, other CAD shapes have also been
396 drawn: polygons of 10 and 12 sides, and squares and parallelograms with rounded corners (Fig. 12).
397 The radius (in pixels) obtained at each corner or rounded corner is shown. The parameters obtained,
398 which allow finding the corners of the decagon and the dodecagon, are shown in Table 9, and the
399 quadratic threshold function is shown in Fig. 13. From now on, the CAD set plus decagon and do-
400 decagon it is called 'CAD+ set').




401

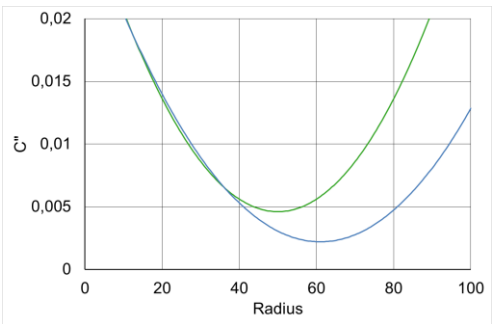
402 Figure 12. Additional CAD shapes with the radius (in pixels) obtained at each corner or rounded
403 corner

404

405 Table 9. Parameters obtained by means of Simulated Annealing for the training set and for the
406 'CAD+' set

| TCVD Parameters | Training set | CAD+ set |
|---|---|---|
| K | 0.000010 | 0.000007 |
| $C''_0$ | 0.0046 | 0.0022 |
| $R_0$ | 50 | 61 |

407




408

Figure 13. Comparison of the threshold obtained for the training set (green) and for the CAD+ set (blue)

With the parameters obtained for the training set, the corners of the decagon and dodecagon are not found. Moreover, with the parameters obtained for the CAD+ set, a false corner is found at the rounded corner of radius 59 of Fig. 12d, this is because the curve is very short and the radius varies quickly (straight line – curve – straight line). The shape of Fig. 12c is well segmented with both sets of parameters. Once again, it is proved that the variation of the radius (or curvature) it is more significant than its value.

The executable version of the algorithm TCVD-R is available, as the shapes sets, in the Downloads section of the following address: http://www.cofilab.com.

## 5. Conclusions

The TCVD method is an important improvement in the field of the free-hand sketches recognition, being the main contribution of this method the detection of tangent points in strokes containing curves. The accuracy of TCVD obtaining corner vertices is higher than others found in literature mainly because it has very few false positives, but also TCVD is able to find curves and straight lines, which allows obtaining tangent points between curves and between curves and straight lines, even with very few false positives, making it unique in this field. Even so, two important improvements to this method have been implemented in the TCVD-R algorithm.

The first improvement has consisted in establishing mathematical conditions to detect corners. Instead of a heuristic threshold, a discriminatory function is used to obtain corners avoiding further refinements, and it is based on the second derivative of the curvature as a quadratic function of the radius. The second improvement is the obtaining of curves independent from the scale by means of two parameters: the *Ratio* between the major and minor radius and the *Angle* swept by the tangent vector to the stroke. With these parameters the scale does not matter, it is important that the radius variation along the curve is not too high and that the curve turn a significant angle.

The results in corners finding are very similar (the "All-or-Nothing Accuracy" for corners increases 0.6%, from 96.9% to 97.5%) because the size of the strokes does not affect the corners, and also the previous heuristic was already pretty good (like the new discriminant function, it also evaluated more the variation in radius than the radius itself). However, it solves one of the common criticisms in [15] to corner finding techniques: the reliance on heuristics. Due to its reliance on the second de-

rivative of the curvature rather than on the curvature, the TCVD-R has been able to distinguish between curves with a radius of 19 pixels and corners with a radius of 66 pixels. The improvement is greater in finding curves, straight lines and tangent points (the "All-or-Nothing Accuracy" increases 4.3%, from 90.0% to 94.3%) because the test dataset contains shapes of different sizes and the TCVD-R is more robust against the scale. The TCVD-R has also been tested with the same shapes of the dataset but drawn with a commercial CAD application. In this case, with the same set of parameters obtained for hand-drawn strokes used, the All-or-Nothing Accuracy is 100% both for corners and for curves, straight lines and tangent points.

## 6. FURTHER WORK

The next challenge is to perform the necessary extensions to deal with closed shapes. To achieve this aim, two new parameters are needed. First, a *Distance* threshold between the first and last points of the stroke to determine whether it is open or closed. Then, an *Angle* threshold to determine the continuity by comparing the direction of the tangent to the stroke at the first and last points. If the difference between both angles is greater than the threshold, there is a corner, otherwise, there will be a tangent point if the start and end entities are different (straight line and curve) or they are two curves with radius of opposite sign (inflection point).

Another possible improvement could be to apply the TCVD-R to contours of objects detected by machine vision applications. In this case, the contours are always closed and, unlike closed strokes, the first and last points coincide accurately and continuity must be obtained as in the rest of points. This feature forces to perform all the stages (resampling, filtering, obtaining curvatures, curve fitting, etc.) for cyclic sequences of points.

## ACKNOWLEDGMENTS

Spanish Ministry of Science and Education and the FEDER Funds, through HYMAS project (Ref. DPI2010-19457) and INIA project VIS-DACSA (Ref. RTA2012-00062-C04-03) partially supported this work.

## REFERENCES

[1] P. Company, M.Contero, P.A.C. Varley, N. Aleixos, F. Naya, Computer-aided sketching as a tool to promote innovation in the new product development process, Computers in Industry 60(8) (2009) 592–603.

[2] D.G. Fernández-Pacheco, F. Albert, N. Aleixos, J. Conesa, A new paradigm based on agents

applied to free-hand sketch recognition, Expert Systems with Applications 39 (2012) 7181–7195.

[3] F. Albert, D.G. Fernández-Pacheco, N. Aleixos, New method to find corner and tangent vertices in sketches using parametric cubic curves approximation, Pattern Recognition 46 (2013) 1433–1448.

[4] B. Yu. Recognition of freehand sketches using Mean Shift, in: Proceedings of the 8th International Conference on Intelligent user Interfaces, IUI '03, 2003, ACM, pp. 204–210.

[5] H. Hse, M. Shilman, A.R. Newton. Robust sketched symbol fragmentation using templates, in: Proceedings of the 9th International Conference on Intelligent User Interfaces, IUI'04, 2004, pp. 156–160.

[6] B. Paulson and T. Hammond. "Paleosketch: Accurate primitive sketch recognition and beautification". In IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces, 2008, pp. 1-10.

[7] C. Alvarado and R. Davis. "Sketchread: a multi-domain sketch recognition engine". In UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology, NY USA, 2004, ACM Press, pp. 23-32.

[8] T. Hammond and R. Davis. "Ladder, a sketching language for user interface developers". Elsevier, Computers and Graphics, 28 (2005), pp. 518-532.

[9] B. Sarkar, L.K. Singh, D. Sarkar, Approximation of digital curves with line segments and circular arcs using genetic algorithms, Pattern Recognition Letters 24 (15) (2003) 2585–2595.

[10] X. Zhang, J. Song, G. Dai, M. R. Lyu, Extraction of line segments and circular arcs from free-hand strokes based on segmental homogeneity features, IEEE Transactions on Systems, Man, and Cybernetics—Part B:Cybernetics36(2) (2006).

[11] T.P. Nguyen, I. Debled-Rennesson, A discrete geometry approach for dominant point detection, Pattern Recognition 44 (2011) 32–44.

[12] A. Wolin, B. Eoff, T. Hammond, Shortstraw: A simple and effective corner finder for polylines, in: Proceedings of the EURO-GRAPHICS 5th Annual Workshop on Sketch-Based Interfaces and Modeling, 2008, pp. 33–40.

[13] Y. Xiong, J. J. LaViola Jr., Revisiting ShortStraw—Improving corner finding in sketch-based interfaces, in: Proceedings of the EUROGRAPHICS 6th Annual Workshop on Sketch-Based Interfaces and Modeling, 2009, pp. 101–108.

[14] J. Pu, D. Gur, Automated freehand sketch segmentation using radial basis functions, Com-

puter-Aided Design 41 (2009) 857–864.

[15] J. Herold and T. F. Stahovich, A machine learning approach to automatic stroke segmentation, Computers and Graphics 38 (2014) 357-364.

[16] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, 220 (4598), 671-680.

[17] D.G. Fernández-Pacheco, F. Albert, N. Aleixos, J. Conesa, M. Contero, Automated tuning o parameters for the segmentation of free hand sketches, in: Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAP2011), 2011, pp. 321–329.