3D Sketch Stroke Segmentation and Fitting in Virtual Reality

Michele Fiorentino, Giuseppe Monno, Pietro Alexander Renzulli, Antonio E. Uva
D.Dis – Politecnico di Bari, Bari, Italy
{m.fiorentino, gmonno, a.uva}@poliba.it

Abstract

In this paper we present a method which tries to automatically represent the designer's intention while sketching three-dimensional curves in a Virtual Reality environment. We translate conceptual sketch strokes into a suitable B-spline representation with a three step method. Firstly a data filter is used to eliminate redundancy and noise in 'pen' position recorded by the 3D tracking system. Secondly an knowledge based algorithm tries to interpret user's intention, according to direction, speed and curvature of the virtual pen segmenting the stroke into two types of curves joined respectively with C^0 and G^1 continuity. Lastly an algorithm translates the points of each segmented sketch stroke into a cubic B-spline with adaptive approximation. This method, which has been integrated in our Virtual Reality sketching system, is illustrated and tested with various types of sketches.

Keywords: *Geometric modelling, VR-virtual reality, 3D tracking, sketching, curve segmentation, spline approximation.*

1. INTRODUCTION

1.1 Curve sketching in conceptual design

Conceptual design is one of the initial phases of the design process, performed by the industrial designer. This professional figure must take into account the market and user requirements to determine aesthetic and visual impact of the product in order to give it the 'added value' and 'desirability'. In particular the designer represents the conceptual ideas in the form of various solutions by sketching. These ideas are then passed on to the engineering designers for the preliminary and detailed design. The engineering design phase nowadays is nearly all computerised. On the other hand in the conceptual design process there has been much less use of computing[1]. Current CAD software do not provide sketching facilities that are as intuitive and flexible as traditional tools such as pen and paper. In order to represent sketches into a digital format some mathematical knowledge is normally needed. Splitting of curves, number of control points and order of curves should be taken into account by the user. Such approach represents a concrete limit to the free expression of ideas and alternatives evaluation, typical of conceptual design. However, the ongoing research and the increase in computer performance are contributing to an acceleration of the integration of the sketching phase into the rest of the design cycle. Use of Virtual Reality (VR) techniques, instead of traditional two-dimensional devices (monitor, keyboard and mouse), have made possible sketching directly in 3D space in a more intuitive fashion. Previous works provided some functionalities for creating free form surfaces from 3D curves. They showed the unexploited potentiality of real 3D conceptual design in VR. We think that Virtual Reality offers a better perception of three-dimensionality that in combination with 3D interaction, provides direct drawing and positioning and constitutes an environment for expressing ideas and concepts. Understanding the user's intention while sketching in the virtual environment motivates the work described in this paper.

1.2 Previous Work

Many sketching systems have been developed in the past, some which use a mouse, others which use a pen and tablet, others, for example Grossman et al. [5], use a tracking system to trace the movements in 3D space. Most of these interfaces limit the curve tracing to a constrained 2d plane (table or tablet) and then for example, create 3D shapes by extrusion or sweeping along a path which is drawn on the same constrained plane. In the "digital tape drawing system" [5], 3D curves are drawn onto a monoscopic display by using "depth planes": a 2D curve is extruded along a line to form a curved plane and subsequently another 2D curve is projected onto the plane to give it its 3D form. Wesche and Droske [6] proposed some sketching tools based on an energy approach for the conceptual drawing of curves and surfaces in a particular Virtual Environment called Responsive Workbench. In a further work Wesche and Seidel [7] presented for the same environment some tools which perform indirect drawing and modification of a curve network for surface design.

The majority of these sketching systems seems to need the splitting of the inputted curves into separate segments. Van Dijk's et al. [3] algorithm does not perform segmentation, but the authors conclude that one way to improve the 2D sketching could be the recognition of 'sharp corners'.

Many authors have concentrated their efforts solely on the segmentation and representation of the curves without an investigation of the interface. Initial research concentrated solely on polygonal representation of each segment in which straight lines are used to connect neighbouring points/pixels to form the overall curve [8] ,[9]. More recent methods also use arcs, circles, other conic sections and freeform parametric curves for representation [2],[5]. Eggli et al. [4] let the user select different predetermined modes of sketching which allow representations preferring either lines and circles, or horizontal and vertical lines or B–spline curves. Van Dijk and Mayer [3] use free-form curves exclusively. The segmentation process has been pursued either by breakpoint detection or edge approximation. The first involves detection of corner points by looking at the discontinuity of the tangent to the contour or of the curvature. The second approach, as in Ray [8], fits lines or arcs to the entire sketched segment, calculating a 'goodness of fit parameter', reducing the segment and repeating the evaluation until the curve is 'good' enough hence creating a sub-segment. Qin et al. [2] point out that these methods however can require heavy computation which makes them unsuitable for on-line segmentation.

Filtering may be necessary to eliminate noise especially for the breakpoint detection methods. Ji-Hwei and Li [11] use Gaussian scale-space filtering to eliminate false dominant points. Van Dijk et al [3], on the other hand, use a filter routine on the segments in order to have a more 'manageable number of points'. In Qin's et al. [2] points which are too close together are filtered out by fixing a minimum threshold and by considering sketching acceleration. This procedure for segmenting 2D freehand sketches uses adaptive thresholds and fuzzy logic based on dynamic features such as drawing direction, speed and acceleration to determine corner points. It is a very robust method that does not involve heavy computation and interprets the user's sketching in satisfactorily manner. Similarly, Eggli et al. [4] also take into account factors such as the length of the curve, designer's skill and sketching speed and to interpret the user's intention. Poedehl [4] identifies certain terms used by designers to communicate intentions. He describes and measures each term in order make such measures usable in optimisation algorithms. Likewise Giannini et al. [13] identify certain terms commonly used for shape evaluation and modification and have developed 'modifiers' to change the shape of curves and surfaces. Many modeller interfaces allow direct manipulation of the generated curves via the use of control points. Van Dijk's and Mayer's method [3] of modifying a curve, by locally adding more strokes, is very similar to what happens when sketching with pen and paper but, it seems to need some improvement to work properly. Although many studies about free form sketching have been carried out in the past, most of them rely on a planar paradigm based on geometrical information without taking in account additional parameters provided during the sketching phase, and some are not suitable for real time computation.

1.3 Our Approach

We propose a method for understanding, analysing and representing the user's intention when sketching curves directly in 3D space, using VR techniques. Three main steps are followed:

- a preliminary filter eliminates superfluous data and noise;
- a segmentation algorithm, which uses position, speed and curvature to determine dynamic thresholds, extracts two types of curve segments. The first type, called *type A*, represents segments, divided by sharp corners, which should only have C^0 continuity between them. The second, *type B*, represents segments with smoother joints having G^1 continuity;
- a spline approximation algorithm translates the points each segment into a spline, automatically specifying the number of control points, according to a style adaptive weight.

The whole algorithm has been implemented not just to be a post-processing phase of a 3D tracking task, but to accept a continuous flow of data and to continuously generate splines that approximate the original tracked points according to the user's intention. A first prototype was implemented with Matlab [16] software in order to set and fine tune all the parameters. Hence, the algorithm has been ported in our Virtual Reality sketching system called in order to perform real time sketching.

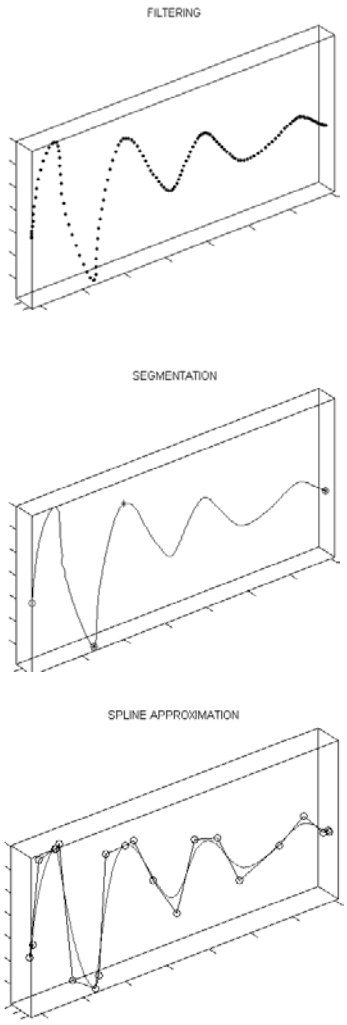

Figure 1*: a) the filtered points, b) segmentation, c) spline approximation.*

This paper is organized as follows. In section 2 we present the approach for the filtering and curve segmentation. In section 3 we describe the spline approximation technique, while section 4 includes a presentation of the case study results. In section 5 the results are discussed, some concluding remarks are made together with some suggestions for future work.

1.4 The VR System

Our system is a VR based cad system developed for conceptual design. Built with an expandable architecture, is support 3D input and output. The user is provided with a pen device to sketch stokes in 3D space and a menu tablet to browse options. The tracking system used, is an optical 3D tracking system by Art (Advanced Realtime Tracking GmbH). The system comprises two infrared (IR) emitters and camera receivers which receive the rays reflected by 'markers', triangulating their position in space. The tracking markers, which are IR-reflective spheres, are attached to the interaction device. When an activator (sketch button) on the pen is pressed, data is received as a series of 3D point coordinates. The sampling frequency is customizable and is set to 60 Hz. The tracked information is sent via LAN to

the processing and graphical unit for computations. The scene is displayed in passive stereo mode, using two projectors and polarized filters. The user who is wearing polarized glasses perceives a single 3D image of the sketch.

2. CURVE SPLITTING ACCORDING TO USER INTENTION.

2.1 The input data and filtering.

An accurate experimental study has been performed by the authors [17] to evaluate some important factors affecting tracking data: the accuracy of the optical tracking system we use, and human factors which influence interaction in a VR environment such as and limb posture, speed, and direction during pointing. Previously used VR tracking systems, like magnetic and acoustic ones, suffered from drawbacks in precision, latency, resolution and repeatability of measurements. The higher precision achieved with optical systems allows research to be addressed to better reduce the error generated by human factors. We found that the optical tracking system itself has a positional error with an average of 0,35mm with a standard deviation (noise) of 0,045mm. This error is way lower than the human error (this was not true with magnetic trackers) therefore we decide to ignore the error of the system. One aspect emphasized by our experiments is the high anisotropy in the user error while pointing in Virtual Environments. The error along the direction between the user head and the pointing tip is always higher and ranging between 1,8 and 2,6 times the error along the horizontal and the vertical directions. The following table shows maximum and mean values for the error along the three directions referred to the user head.

Table 1: Statistic error values (mm) for the performed test

ERROR	Total	Horiz. Error	Vert. Error	Depth Error
Max Value	17,31	7,28	9,53	19,50
Mean Value	6,21	4,81	5,29	10,12

Such considerations lead to the design and the calibration of a preliminary anisotropic data filter. By pursuing a similar approach to the one used by Qin et al. [2], the points which are too close to each other are eliminated. For a series of n points $\{P_i \mid i = 1,2,3,\ldots n\}$ each with 3 coordinate values attributed to them, each point is recalculated as follows:

$P_{i+1} = P_i$ if the components of the points distance along the aforementioned directions are all lower than three thresholds defined according to the tests performed:

$$\overline{P_{i+1} - P_{i\ horizontal}} < 5mm;$$
$$\overline{P_{i+1} - P_{i\ vertical}} < 6mm;$$
$$\overline{P_{i+1} - P_{i\ depth}} < 11mm;$$

For the reasons explained above, the input data is affected by quantization errors and noise. This error propagates in the discrete speed calculations and angular deviation more significantly for higher sampling frequencies. Therefore we smooth the position data by applying a Gaussian filter.

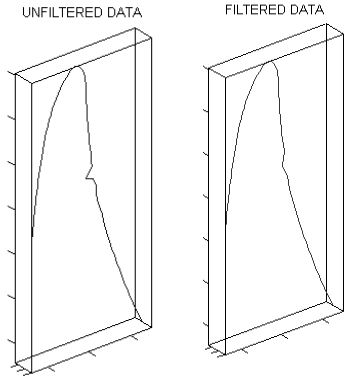

Figure 2*: a) data directly from the tracking system, b) data after filtering.*

2.2 Primary Segmentation – *type A* curves

In a similar manner to the approach followed by Qin et al. [2] we distinguish two types of curves according to segmentation points. We first detect primary segmentation points where the angular deviation is important.

For each point P_i a directional deviation α, i.e. the angle subtended by two vectors, is evaluated. The first vector is calculated between the point P_i and the preceding point with index (i-m); the second vector is calculated between the point P_i and the successive point with index (i+m),

$$\alpha = \text{angle}\left(\overrightarrow{P_i - P_{(i-m)}}, \overrightarrow{P_{(i+m)} - P_i}\right)$$

where m is the alpha adaptive support length given by:

$$m = \text{round}\left(2 \times \frac{S_{\text{avg}}}{S} + 0.5\right)$$

where S_{avg} is the average sketching speed and S is the speed at point P_i.. The purpose of m is to avoid picking points too close to each other, whilst the pen is moving slowly, when it is likely that unintentional drawing deviations may occur. Similarly when the pen is moving quickly, unintentional drawing deviations are unlikely and hence small values of m may be used to calculate the directional deviation α. We had to limit the value of the support m to 10 due to the fact that sketching with our system, compared to Qin's et al. which uses a mouse, is more fluid and reaches higher speeds. As the pen moves along, if a point with value of α superior to a predetermined threshold is encountered, it is classified as a primary segmentation point. If a series of successive points have values of α superior to the threshold, the algorithm carries on until a point with an inferior α value is reached and then classifies as a primary segmentation point the one in the series that has the max α value. The first point is always considered as primary segmentation point. For our tests we set the α threshold to 90. Hence all the points between the first point and the first primary segmentation point encountered are

considered as part of a *type A* curve segment which is flushed onto the following step for the secondary segmentation analysis. In the meantime this algorithm moves on to the new points acquired by the tracking system, classifying the first point as a primary segmentation point and searching again for a second one to create a new segment.

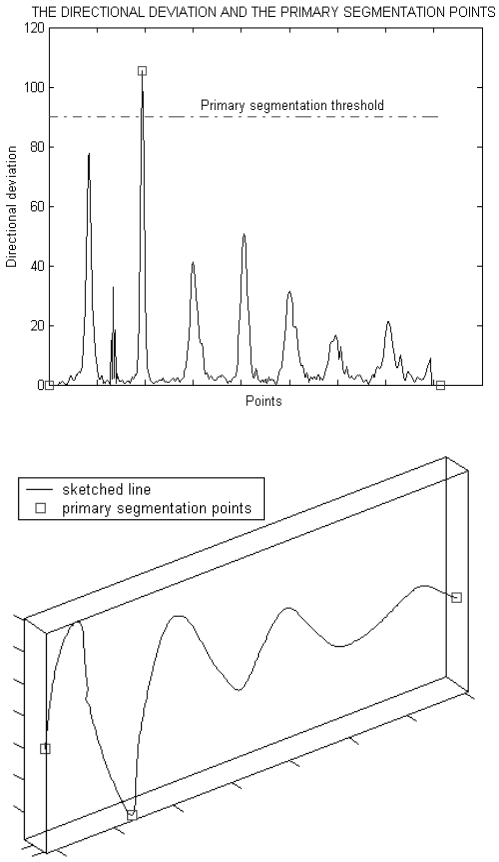

Figure 3*: a) directional deviation plot for all the points; one point is above the alpha threshold (apart from the initial and last point which are by definition segmentation points). b) The primary segmentation points of the sketched curve.*

2.3 Secondary Segmentation - *type B* curves

This secondary segmentation analysis is performed to capture turning points which may occur due to a combination of slight changes in curvature and velocity. For each point in the segment the value of α, previously calculated, is compared to an adaptive total threshold β_t and the speed is compared to an adaptive speed constraint ϖ.

The adaptive total threshold β_t is itself a combination of various thresholds. Firstly a basic angle tolerance β_0 is established. It represents a limit below which a point is unlikely to be a turning point. This threshold was established by asking several users to sketch a straight line in 3D space and then measuring the max value of the directional deviation α for each user. We found that appropriate value for $\beta_0 = [3 \div 5]$

Secondly a linearity adaptive threshold β_{lin} is calculated. The purpose of β_{lin} is to take into account the fact that when sketching a curve, the total threshold above which a point is classified as a turning point should be larger than when sketching a line. The linearity adaptive threshold calculation considers the linearity of a segment, which is defined as the ratio of the distance between the two segmentation points to the cumulative arc length between the two points. Hence for each point of the segment,

$$\beta_{\text{lin}} = 4 + 20 \cdot \left(1 - \text{L}_\text{p} \times \text{L}_\text{n}\right)$$

where L_n is the linearity between point P_i and the last point of the segment (the first primary segmentation point) and L_p is the linearity between point P_i and the first point of the segment (the second primary segmentation point).

Next an adaptive velocity threshold β_s is established. This threshold takes into account the fact that if the pen is moving with a high speed, it is likely to trace a smooth curve, hence an abrupt change in direction is more likely to be a tuning point than one at slow speed. So

$$\beta_\text{s} = 7 \cdot \frac{S_\text{avg}}{S};$$

β_s is limited to a maximum value of 15.

The adaptive total threshold β_t can now be calculated

$$\beta_\text{t} = \beta_0 + \beta_{lin} + \beta_\text{s}$$

Finally, to account for the fact that when in proximity of a turning point the sketching speed tends to be lower than average, an adaptive speed constraint ϖ is introduced where

$$\varpi = (\text{S}_\text{avg} \times \text{L}_\text{p} \times \text{L}_\text{n})$$

For every point of the *type A* segment, the point becomes a turning point (secondary segmentation point) if three conditions are fulfilled:

$$\beta_t < \alpha$$

$$speed < \varpi$$

$$speed > K_{s\,\text{min}} \times \varpi$$

As a result two smaller segments called *type B* segments are created from the *type A* segment. We noticed that when sketching with a pen directly in 3D space, the user at times reaches the end of the curve and does not immediately release the button. Consequently the tracking system records unwanted points which are close to each other with low speed therefore they are potential turning points. The last speed condition is introduced to avoid recognizing such points as turning points. We found that a value of K_{smin} of 3% gave satisfactory results. Once the *type B* analysis is finished the algorithm starts again to look for the next primary segmentation point. The whole process of segmentation is carried out on-line until the user releases the pen button.

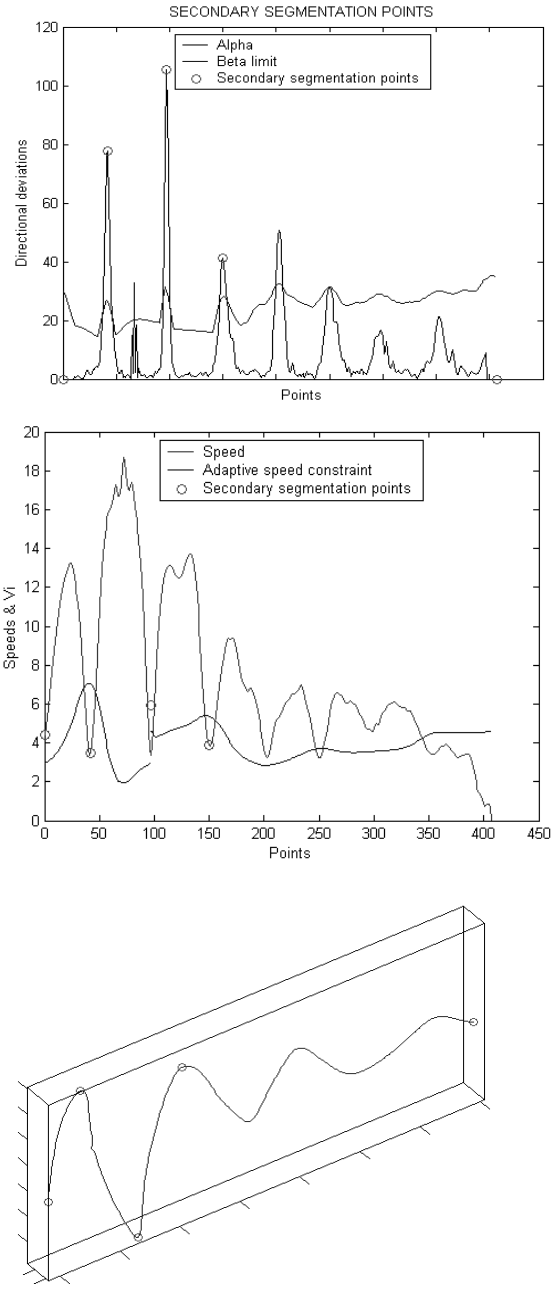

Figure 4: *Secondary segmentation points are detected if the angular deviation is above β_{limit} (fig a) and if the velocity is below the speed threshold ϖ (fig b). Secondary segmentation points include the primary segmentation points (fig c).*

3. SPLINE GENERATION

Curve fitting follows the generation of the segments. B splines are used to approximate the segments.

3.1 Detail Level Function.

We noted that when sketching if the speed is low, it is more likely that the intention is to represent something in a detailed manner. On the contrary if the speed is high, the user is not trying to represent a detail with the generated curve. Moreover, during the modification of a curve generated at high speed it is unlikely that the user will need many control points for local curve repositioning. Similarly if the curvature is low, the control points necessary for modification will be few. An adaptive 'detail level' weight L_i is evaluated according to curvature and speed for each segment. This weight tries to understand with how much 'tension' the user wants to represent the segment. The adaptive 'detail level' weight associated to each segment is evaluated as follows for each segment:

$$L_i = \frac{\left(1 - \frac{Length_{Chord}}{Length_{Arc}}\right)}{S_{avg}}$$

where $Length_{Chord}$ is the distance between the extremities of the segment, $Length_{Arc}$ is the segment arc length and S_{avg} is the average speed of the segment.

3.2 B-spline Approximation

The final result of the complete algorithm is to provide a set of mathematical representations of the curves keeping them simple for further analysis and modification. Therefore we decide to use B-spline representation for curves where the spline *Sp* is specified by its nondecreasing knot sequence knotS and by the control points sequence. The input in this step is basically a set of n points P_i. We perform a least-squares spline approximation. The original problem of least-squares approximation with B-spline curves has been solved by Carl de Boor [15] whose implementation has been followed in our approach. Basically, given the order k and the not-decreasing knot sequence *knotS*, the algorithm gives the control points CPs of the approximant B-spline *Sp* for which the weighted standard deviation:

$$StdDeviation = \sqrt{\frac{1}{n}\sum_{i=1}^{n} w_i \cdot dist(P_i, Sp)^2}$$

is minimized, where the weights w_i associated to P_i forces the spline to pass closer to the points. Since after the previous segmentation phase each segment is quite smooth, we decide 'a priori' to use cubic splines to keep C^2 continuity inside the segment (order = 4), and *knotS* uniformly distributed with multiplicity equal to order at extremities. We also use a higher weight on the extreme last points to force the fit to come very close to the extremities of the sampled curve segment and to try to preserve the tangent direction. The number of control points is evaluated adaptively for each segment *i* according to the detail level function in section 3.1.

$$NumCPs = \min NumCPs + round(70 \cdot L_i)$$

For this application we decide to keep the minimum number of control points equal to 4. Since the result is an approximating B-spline, and it is not guaranteed to pass through the extremities, we force the control points in order to have C^0 continuity on the joint between two type A segments and G^1 continuity on the joint between type B segments.

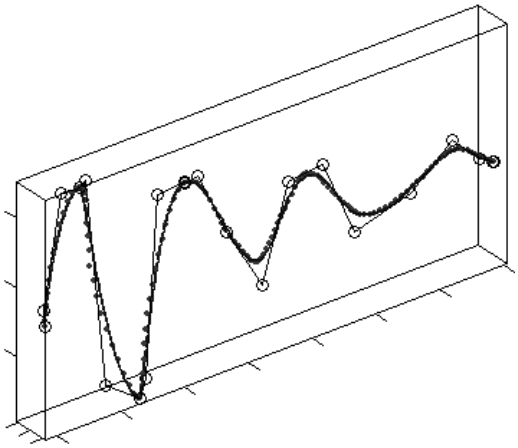

Figure 5: *The generated spline (continuous curve) generated from the points (dotted curve). The control points are represents by the small circles. The control graph joins the CPs with straight lines.*

4. RESULTS

Real-time sketch cases have been generated and tested with the algorithm described in this paper. Figure 6 shows the input points, the approximating spline (the continuous curve) and the CPs (small empty circles) with their control graph. The primary segmentation points are represented with a coloured square, whilst the secondary segmentation points are represented with a coloured circle.

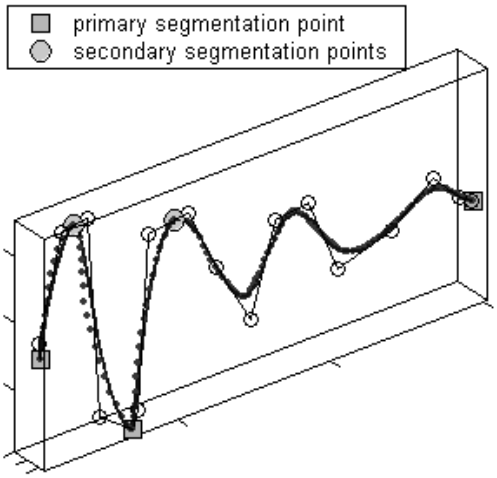

Figure 6: *The approximating spline and its segmentation points.*

The straighter segments have only 4 CPs whist the more curvilinear segment has 12 CPs. In Figure 7 the shortest segment has the same number of CPs as the other longer and more curvilinear segments. This is because that segment is drawn at a lower velocity (the points are closer together).

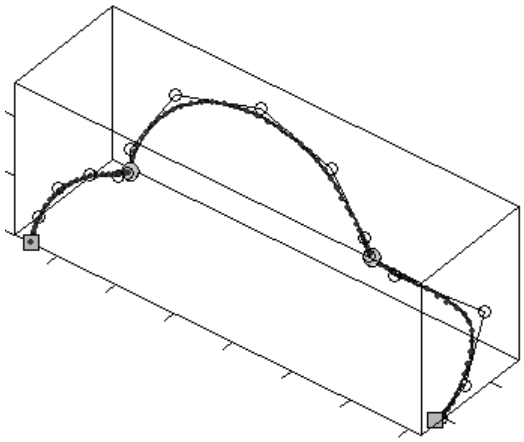

Figure 7: *The approximating spline and its segmentation points.*

In Figure 8 the same curve is sketched twice but with different speeds. As intended, the slower sketch has more segmentation points and control points to represent the details and to allow for more 'local' modification with the control points.

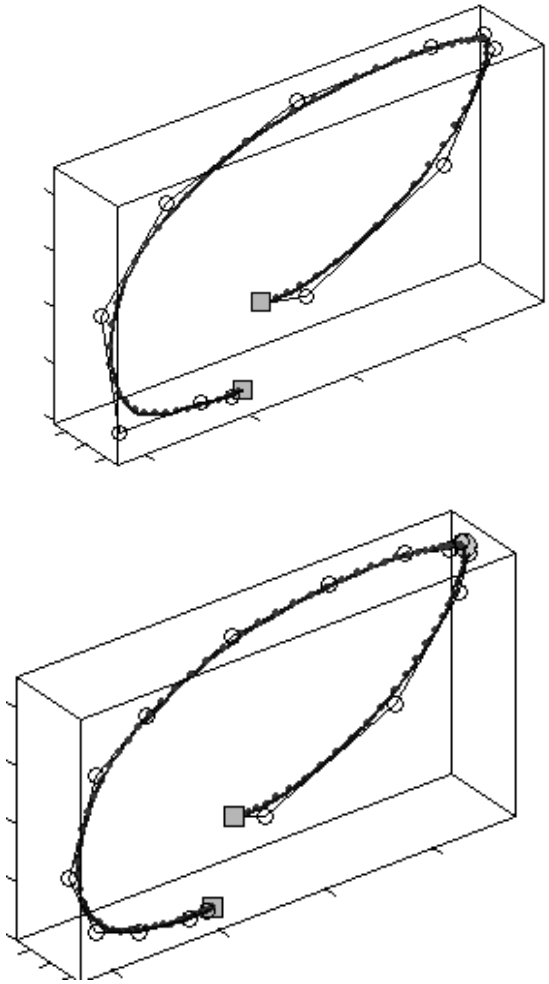

Figure 8: *Influence of speed: top sketch is drawn at a higher speed than bottom one.*

Other cases of free-form sketches and their representation are illustrated in the following Figure 9.

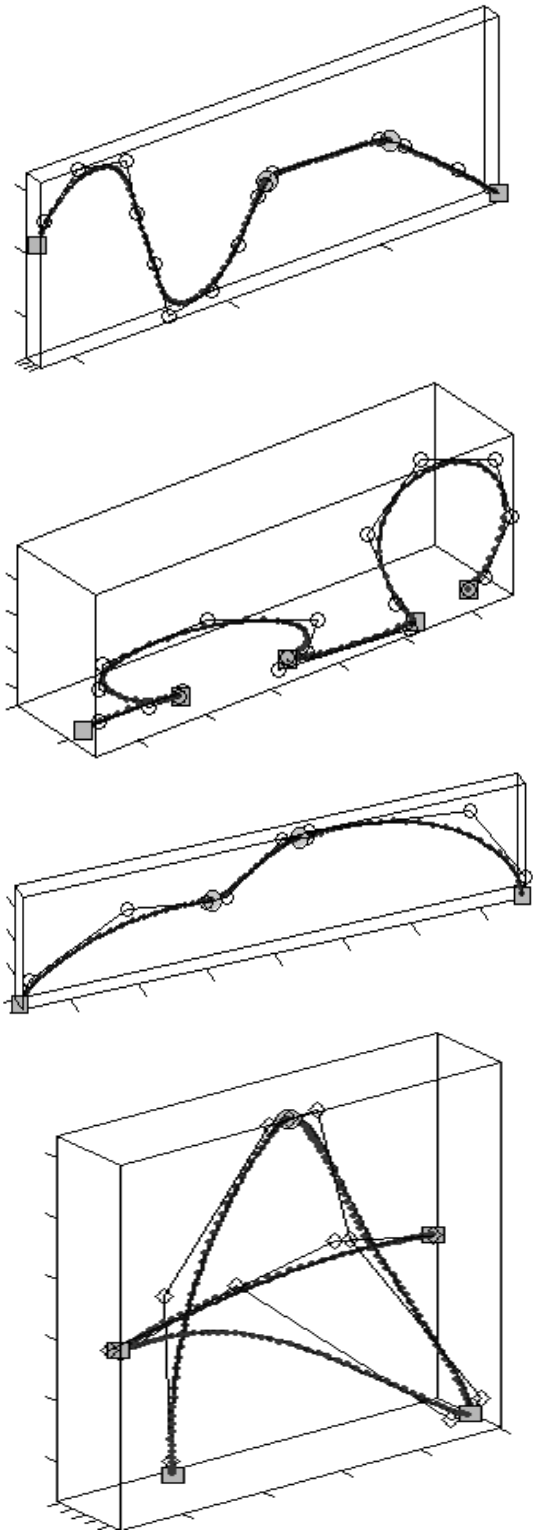

Figure 9: *Test Cases*

Sharp corners are recognized as join points with C^0 continuity and smoother corners as join points with G^1 continuity. Splines representing straighter segments and those drawn at higher have few CPs. Whilst, splines drawn with a large curvature or at low speed are represented with splines with a larger number of CPs to allow for more 'local' modification.

Figure 9: *User session in Spacedesign.*

5. CONCLUSION AND FUTURE WORK

An algorithm for representing the designer's intention while sketching on-line three-dimensional curves has been implemented in our VR Cad system. The algorithm uses position, speed and curvature data to determine dynamic thresholds needed for the segmentation of the sketch stroke. The same data are also used to evaluate a style adaptive weight which influences the number of spline control points used to approximate each segment of the stroke.

We tested the proposed algorithm integrated in our VR system, with multiple users. A series of sketch strokes have been automatically converted into B-spline accordingly to nature of the stroke. The segmentation procedure work quite well in nearly all the test cases, extracting correctly the two different classes of curves. There are some cases however where the intended sketch is not interpreted precisely. This is due mainly to unpredictable movements of the user's hand which are difficult to discriminate from the desired intention.

In the future our intention is to find a better trade-off between error filtering and detection accuracy. We also want to provide the designer with post-sketching editing function which may preserve the parametric relationships between the segments.

6. REFERENCES

[1] M. Tovey; "Styling and design intuition analysis in industrial design". Design Studies 18 (1997) pp.5-31.

[2] Qin, Sheng-Feng, David K. Wright, and Ivan N. Jordanov, “On-line segmentation of freehand sketches by knowledge-based nonlinear thresholding operations”. Pattern Recognition, vol.34 (2001), number 10, pp. 1885-1893.

[3] Dijk C.G.C. van , Mayer A.A.C., "Sketch input for conceptual surface design." Computers In Industry 1997, vol.34, Iss. 1, pp.125-137.

[4] Eggli, Lynn, Ching-yao Hsu, Beat D Bruederlin, and Gershon Elber, “Inferring 3D Models from Freehand Sketches and Constraints”. Computer-Aided Design, 29(2): 101-112, 1997.

[5] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, B. Buxton, “Creating Principal 3D Curves with Digital Tape Drawing”. ACM CHI 2002, vol. 4, Isssue no.1, p.121-128.

[6] G. Wesche and M. Droske, “Conceptual Free-Form Styling on the Responsive Workbench”. VRST 2000 Proceedings, pages 83-91, Seoul, Korea, October 2000.

[7] G. Wesche and H.-P. Seidel, “FreeDrawer-A Free-Form Sketching System on the Responsive Workbench”. VRST 2001 Proceedings, pages 167-174, Banff, Alberta, Canada, November 2001.

[8] Ray, B.K. and K.S. Ray, “A new split and merge technique for polygonal approximation of chain coded curves”. Pattern Recognition Letters, vol.16 (1995), number 2, pp. 161-169.

[9] Horng, Ji-Hwei, “Improving fitting quality of polygonal approximation by using the dynamic programming technique”. Pattern Recognition Letters, vol. 23 (2002), number 14 pp. 1657-1673.

[10] Ichoku, C., B. Deffontaines, and J. Chorowicz, “Segmentation of digital plane curves: A dynamic focusing approach”. Pattern Recognition Letters, vol. 17 (1996), number 7, pp. 741-750.

[11] Ji-Hwei Horng, Johnny T. Li, “A dynamic programming approach for fitting digital planar curves with line segments and circular arcs”. Pattern Recognition Letters, Volume 22 (Feb. 2001), Number 2, February 2001 183-197.

[12] Gerd Podehl, “Terms and Measures for Styling Properties”. 7th International Design Conference – Design (May 2002).

[13] F. Giannini, M. Monti, “An Innovative Approach to the Aesthetic Design”. Common Ground – The Design Research Society Conference 2002.

[14] M. Fiorentino, R. De Amicis, A. Stork, G. Monno; “Surface Design In Virtual Reality As Industrial Application”. In Proc. of Design Conference - Design 2002, Dubrovnik, Croatia, May 14 - 17, 2002.

[15] De Boor C., “A Practical Guide to Splines”. Springer-Verlag, New York, 1978.

[16] http://www.mathworks.com/

[17] M. Fiorentino, G. Monno, P. A. Renzulli, A. E. Uva, “3D Pointing in Virtual Reality: Experimental study”, International Conference on Tools And Methods Evolution In Engineering Design, Napoli, Italy, 4-6 June 2003.

About the authors

Michele Fiorentino is a Researcher at Politecnico di Bari, Italy faculty of Mechanical Engineering. His contact email is m.fiorentino@poliba.it.

Giuseppe Monno is a Full Professor at Politecnico di Bari, Italy faculty of Mechanical Engineering. His contact email is gmonno@poliba.it.

Pietro Alexander Renzulli is a Ph.D. student at Politecnico di Bari, Italy faculty of Mechanical Engineering.

Antonio E. Uva is a Researcher at Politecnico di Bari, Italy faculty of Mechanical Engineering. His contact email is a.uva@poliba.it.