www.regeo.uji.es
Technical Reports. Ref. 11/2018

Finding vertices in 2D line-drawings of polyhedral shapes

Pedro Company[1], Raquel Plumed[2], Peter A.C. Varley[1], Jorge D. Camba[3]

[1]Institute of New Imaging Technology, Universitat Jaume I, Spain
[2]Departament of Mechanical Engineering and Construction, Universitat Jaume I, Spain
[3]Department of Computer Graphics Technology, Purdue University, USA

Abstract

This technical report revisits the problem of detecting junctions in 2D sketches that depict 3D vertices. Our purpose is to use an artificial perception model to build an algorithm that not only detects junctions in careful sketches created by skilled engineers and designers, but also detects junctions when skilled people draw casually to quickly convey rough ideas.

Our approach scores the likelihood of the junctions, i.e., calculates a figure of merit that estimates how likely a junction is to be perceived as such. Some examples are included to demonstrate the capabilities and limitations of the approach.

Key words: Sketches-based modeling. Polyhedral shapes. Vertices. Junctions.

1. Introduction

A particular research area in the Sketch-Based Modeling (SBM) domain is aimed at processing geometric figures sketched in 2D to automatically generate the corresponding geometric 3D models such as those used by CAD/CAM/CAE applications.

Current Sketch-Based Modeling approaches for extracting junctions from digital images are mostly incomplete, as they simply merge endpoints that are near each other, thus ignoring the facts that different vertices may be represented by different (but close) junctions, and that the endpoints of lines that depict edges that share a common vertex may not necessarily be close to each other, particularly in quickly sketched drawings. Many reconstruction approaches assume that junctions are carefully depicted in the sketch ([XCS14] is a recent example). However, junctions are not necessarily contained explicitly in the original sketch or automatically obtained from the vectorization stage.

We describe and assess an improved approach for merging tips of vectorized lines to produce 2D junctions (points where the tips of the lines meet) that depict 3D vertices (points where the edges of polyhedral shapes meet). Our proposed approach is described in [CVPC18].

Our general assumptions are that (a) the problem of detecting junctions in sketches is not merely geometrical (see [Kan79], cited by [JGH09]), and (b) that skilled people may draw both carefully and casually. Careful sketches are used to convey detailed and/or finished ideas, whereas casual sketches convey on-the-fly and/or incomplete ideas.

The quality of sketches of an average population can be viewed as a continuum including good and bad sketches. For typical users of CAD oriented SBM tools this continuum is clearly biased towards good sketches, since engineers and designers are trained sketchers. However, we distinguish between careful and casual sketches, not as extremes of a continuum, but as two different modes used consciously for different purposes.

People use sketches because they convey information that can readily be perceived by humans (although not so readily by computers). In addition to the current limitations in artificial perception, a standalone global detection approach may get unnecessarily overloaded if casually sketched line-drawings are not beautified in advance. This is why we advocate for a local detection of junctions.

To coordinate local and global perception, our approach scores the likelihood of potential junctions in a sketch. This functionality results from our belief that sketches convey perceived geometry instead of actual geometry. Therefore, we do not assert “this is a vertex” but “this is more or less likely to be a vertex.” In this way, we provide a candidate beautified line drawing for a subsequent global perception mechanism based on our hypothesis that a local intersection

of a set of strokes which can be barely accepted as depicting a common junction when perceived separately, may be seen as a valid vertex when put into context (see, for instance, the lower left junction in the right example of Figure 1).

2. Terminology

The input for SBM approaches are *sketches*, which are freehand drawings comprised of strokes. *Strokes* are scribbled lines sampled by a set of consecutive nodes caught between "pen-down" and "pen-up" movements. The result is an ordered sequence of points, which are connected by segments to approximate the original scribble. A vectorized line can be obtained by suitable fitting-lines-into-stroke approaches.

The input for our vertex-merging approach is a set of vectorized line segments (or simply "lines," as referred to in the rest of the paper) delimited by their tips. Then, vertex detection must merge dangling tips to determine junctions that depict valid vertices. Hence, the output is a *line-drawing*: a list of lines and a list of junctions, where each line connects two junctions. Junctions are *x, y* coordinate pairs of shared endpoints that likely correspond to vertices of the depicted object.

We distinguish *careful* and *casual* sketches (also called *detailed* and *quick* sketches). Examples of both types of sketches are provided in Figure 1. Based on the classification of sketches defined by Ferguson [Fer92], we assume the idea that thought sketches are usually casual, while prescriptive sketches tend to be more careful.

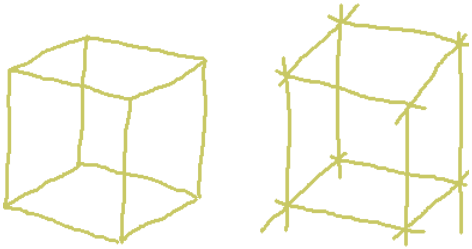


Fig. 1. Careful (left) and casual (right) sketches of a prism

3. The algorithm

The input for our algorithm is a list of tips, defined by their 2D coordinates, and a list of lines where each line connects two tips. Two or more lines may share a common tip. Thus, m= 2*n tips (at the most) are initially defined for n lines.

For implementation purposes we split each stroke into two semi-lines, in which tips are reversely ordered. Only the first tip of each semi-line is analyzed, while the other is simply used to calculate the length and orientation of the semi-line.

As output, clustered tips are merged. When a pair of tips is merged, one is removed, and the other is preserved (as is, or after changing its coordinates).

Lines are visited twice. For each semi-line, one tip is assigned as the first tip.

Functions implemented by the algorithm include:

- The function "*allowance (i, j)*" returns a threshold that prioritizes right angles. The return value varies within the range [0, 1] depending on the angle between lines i and j, explained in section 3.1 of [CVPC18]. The function generalizes for dangling lines that can merge to junctions by calculating the lowest allowance between the dangling line and each of the lines in the junction.

- The function "*meritRotation (NumJ, newPoint)"* returns a figure of merit that decreases when the lines connected to junction *NumJ* rotate when redirected to *newPoint*. Merit is 1 if the lines do not rotate, and 0 if at least one of them rotates more than *maxRot*.

- The function "*GetNeighbourTips()"* returns the lists of tips that are close to each tip. It is implemented as follows:
  1. The endpoint of each semi-line is defined as a tip (t).
  2. The list of tips (T) is sorted by ascending x coordinates.
  3. The Delaunay triangulation of T is calculated.
  4. A tip t is selected.
  5. All tips (nt) that share at least one triangle with t are saved as neighbors (∀ nt neighbor of t).
  6. The process is repeated (back to step 4) for all the tips (∀ t ∈ T).

- The function "*GetSemiLines()"* returns the list of semi-lines. It is implemented by duplicating each line, and swapping first and second tips.

- The function *MergeCarefulTips* () merges neighbor tips of carefully drawn lines. It is implemented as follows:
  1. The longest unvisited semi-line is selected (I).
     a. A semi-line that ends at a neighbor tip of I is selected (J).
     b. The merit $M_{I\text{-}J}$ of the candidate junction I-J is calculated (as described at the end of section 3.1 in [CVPC18]).
     c. If the junction is valid (tips are closer than *maxDist* and lines rotate less than *maxRot*), the candidate junction (I-J) and its merit ($M_{I\text{-}J}$) are saved.
     d. If at least one valid junction was saved, the two tips with the highest merit (max $M_{I\text{-}J}$) are merged.
     e. The merged tip and its connected tips are removed from the neighbor list, and the process repeats for all the remaining neighbor tips (back to step 1.a).
  2. The process is repeated for all semi-lines I (back to step 1).

- The function *MergeCasualTriplets* () merges tips of triplets of lines that intersect with one another (mainly if they intersect close to their tips). It is implemented as follows:

1. Semi-lines whose first tip is not shared with any other line are listed as dangling semi-lines.
2. For each dangling semi-line I:
    a. The dangling semi-lines that intersect semi-line I in the valid range (closer than *InTol* if the intersection is inside segment I, or closer than *OutTol* if the intersection is outside) are listed.
    b. Semi-lines with less than two valid intersections are removed (as we search for triplets).
    c. Intersecting semi-lines are sorted from closest to farthest to the tip of semi-line I (I_1, I_2,…, I_n).
3. The search for valid triplets begins.
    a. The longest unvisited dangling semi-line is selected (I).
    b. Semi-line I and the first two semi-lines in its intersect list (I_1, I_2) are a candidate triplet if:
        i. (I, I_2) or (I_2, I) are the first two consecutive semi-lines in the intersect list of I_1.
        ii. (I, I_1) or (I_1, I) are the first two consecutive semi-lines in the intersect list of I_2.
        iii. Mutual intersections between I, I_1 and I_2 are closer than *maxDistTriplets*.
    c. Otherwise, I is marked as visited, and the algorithm returns to a.
    d. Semi-lines I, I_1 and I_2 are connected to a common triplet junction located at the centroid of the three intersection points.
    e. The merit of the common junction (MT) is assigned as described in section 3.2 of [CVPC18].
    f. Semi-lines merged in the triplet (I, I_1, I_2) are removed from the list.
4. The procedure is repeated while new triplets are found.
5. The second search for valid triplets begins.
    a. Select the longest unvisited dangling semi-line (I).
    b. Semi-line I and the two consecutive semi-lines in its intersect list (I_j, I_{j+1}) are a candidate triplet if:
        i. (I, I_{i+1}) or (I_{i+1}, I) are two consecutive semi-lines in the intersect list of I_i.
        ii. (I, I_i) or (I_i, I) are two consecutive semi-lines in the intersect list of I_{i+1}.
        iii. Mutual intersections between I, I_i and I_{i+1} are closer than *maxDistTriplets*.
    c. Semi-lines I, I_j and I_{j+1} are connected to a common triplet junction located at the centroid of the three intersection points.
    d. The merit of the common junction (MT) is assigned as described in section 3.2 of [CVPC18].
6. The procedure is repeated while new triplets are found.

- The function *MergeCasualDanglingLines* () merges isolated dangling lines to neighboring junctions. It works as follows:
    1. One dangling semi-line is selected (I)
    2. For every junction J (tip of valence higher than 1) closer than a threshold (*maxDistDangling*allowance*), and within the *valid range* of semi-line I, the merit of merging I to J *(EDM*) is calculated as described in Section 3.2 of [CVPC18].
    3. The junction JJ with the highest positive merit is selected.
    4. The new merit of the junction JJ is the minimum between the current merit of the junction and the merit of merging the dangling line to the junction.
    5. The tip of the dangling line I is added to the junction JJ, only if the new merit is not negative.
    6. The procedure is repeated for all dangling semi-lines I.

The pseudocode of the vertices finder is as follows:

```
GetJunctions()
{
  GetLineLengths();
  ReorderLongerLinesFirst();
  GetSemiLines();
  GetNeighbourTips();
  MergeCarefulTips();
  MergeCasualTriplets();
  MergeCasualDanglingLines();
}
```

4. Validation

The performance of our new approach cannot be compared against a predefined ground truth, since there is no "geometrical" solution to the problem of detecting junctions in casual sketches. Even when a sketcher tries to explain her intentions, and depending on the quality and complexity of the sketch, people may interpret the same sketch differently. For this reason, we defined a collection of 12 casual sketches, and asked a group of subjects to number the perceived junctions and mark the tips of the lines that intersected at that junction.

Then, we parsed the examples to conclude that the algorithm accurately detects what humans perceive (Figure 2). The only junction commonly detected by humans that was not fully detected by the algorithm with the default

configuration was junction 12 in example 6, where tip 4 was not merged.

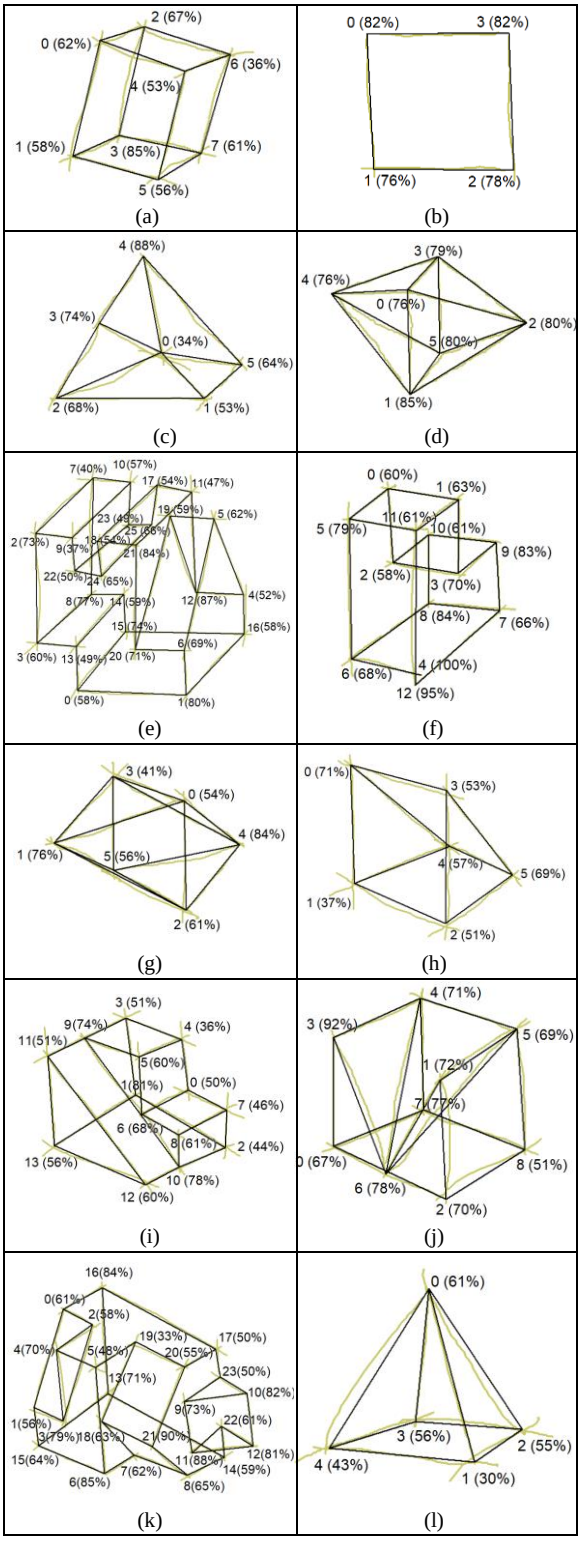


Fig. 2. Figures of the experiment, as detected by the algorithm

We built the benchmark based on our previous experience, and following key assumptions that have been stated elsewhere [CPVC18]: (1) there is no difference in the perception of vertices depicted by junctions in casual sketches between groups with different levels of exposure to technical drawings, and (2) subjects largely agree on perceiving the same junctions.

To further validate the benchmark, we conducted an additional experiment to demonstrate that there are no significant differences between our test sketches, and those collected from other subjects.

Our hypothesis was that there are no significant differences between the set of sketches we used with our original participants, and other sketches created by different individuals. We interviewed 91 subjects (17 from Spain, 61 from the US, and 13 from Italy). Most subjects had an engineering background (18 were engineering educators and 73 engineering students). Every participant was asked to casually sketch the models shown in Figure 3.

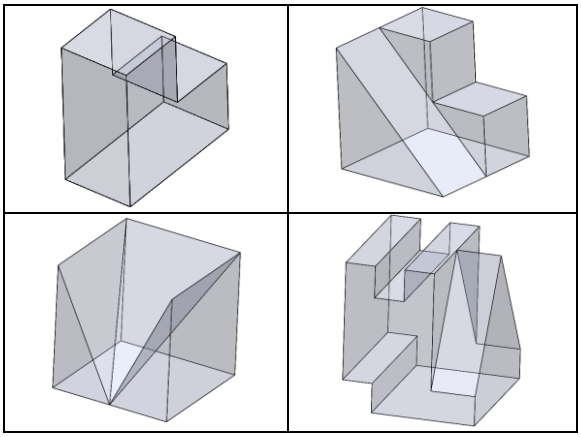


Fig. 3. Transparent models, built with SolidWorks, used in the experiment

To help them understand the task the example shown in Figure 4 was provided.

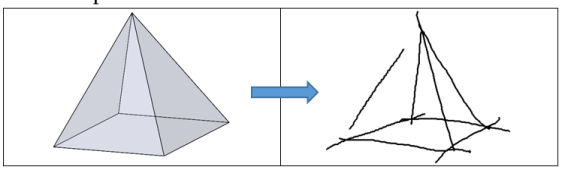


Fig. 4. Example shown in the instructions.

Participants were reminded to include the hidden edges of the model (but to not represent them as dashed lines), and draw each line individually (a single stroke for each line).

We asked them to draw all four sketches in a single piece of paper, which was later scanned and emailed to the authors. Not all the sketches we collected were included in the analysis. Incorrect sketches included: (a) missing strokes, (b) dashed hidden lines, (c) ignored short beautification strokes and (d) overtraced sketches. We discarded 44 sketches from model 1, 25 from model 2, 24 from model 3 and 43 from model 4. All from engineering students. We guess that Model 4 frequently included missing lines because of its complexity, while missing lines in Model 1 where possibly due to its point of view or orientation.

The participants' sketches were vectorized manually by the authors and the line drawings parsed against the "ground truth" of the vertices of the models. Since the subjects were asked to replicate four particular models, we assume they tried to replicate the junctions of the model. Thus, our algorithm was used to calculate which of those junctions were more or less frequently detected.

Table 1 summarizes the relative frequency with which vertices of each model were recognized correctly ("Ok" column), and the frequency with which some unexpected cases occurred. These cases are illustrated in Figure 5, which distinguishes between false negatives (first row), and false positives (second row).

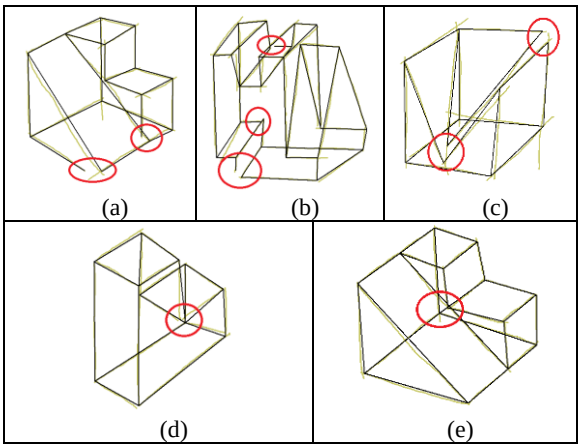


Fig. 5. Unexpected results, (a) dangling lines (lines with valence 1), (b) short dangling lines, (c) duplicate vertices, (d) grouped vertices and (e) lines grouped in incorrect vertex.

Table 1: Algorithm results

	OK	**(a)**	**(b)**	**(c)**	**(d)**	**(e)**
Model 1	**53%**	30%	0%	0%	11%	34%
Model 2	**64%**	29%	3%	8%	5%	8%
Model 3	**79%**	10%	0%	7%	1%	4%
Model 4	**61%**	11%	28%	0%	11%	22%

From the analysis of the frequencies, we can conclude that:

- The algorithm does not fail in recognizing all vertices that are "carefully" drawn.
- The algorithm always assigns high merits (greater than 0.5) to careful vertices.
- For more imperfect vertices, the assigned merits are in the range [0, 0.5].
- The algorithm tends to leave lines without converging to a vertex (dangling lines) before making false convergences. In borderline cases of particular points of view (such as Model 1) the algorithm becomes more sensitive to casual vertices.
- When the drawing contains strokes of very different lengths (Model 4), the algorithm finds it difficult to correctly group the short vertices.
- The algorithm seems to not be susceptible to the direction in which the strokes are drawn. The general metrics we used eliminate the possible bias that could be introduced by drawing with the right or left hand.

Examples with a large number of lines were not included in the previous figures so the junctions could be clearly displayed. Nevertheless, the algorithm can also solve these complex shapes successfully, as shown in Figure 6.

In fact, the increase in the number of junctions is not a problem for the algorithm, although it was noted during the experiment that people tend to get tired and lose some junctions when faced with sketches containing a growing number of strokes. Thus, we cannot test the validity of the algorithm compared to a statistically validated human perception for sketches populated with increasing number of junctions. Still, when a reduced group of persons was asked to agree or disagree with the junctions detected by the algorithm, the answer was always positive (although sometimes accompanied by surprise, as the algorithm detected junctions unnoticed by some people at first sight).

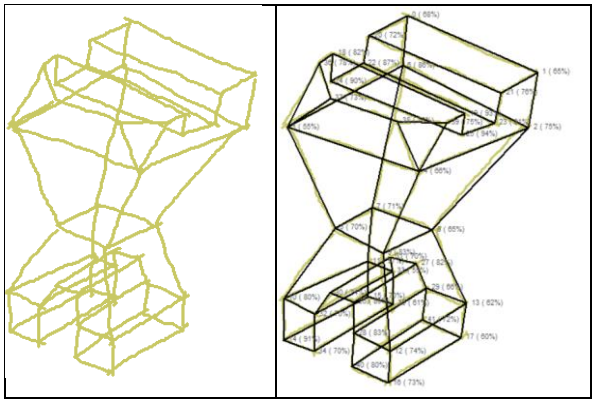
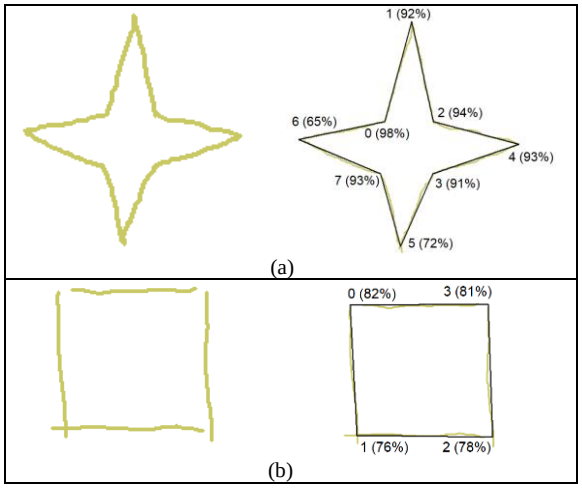


Fig. 6. Careful sketch of a complex 3D polyhedral shape

To further test the merits provided by the algorithm, we selected careful and casual sketches of 2D shapes (see Figure 7 for some representative examples), as well as careful and casual sketches of 3D polyhedral shapes (see Figure 8 for some representative examples).



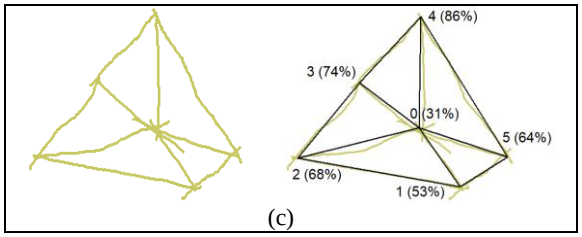


Fig. 7. Careful and casual sketches of simple 2D shapes, with their merits

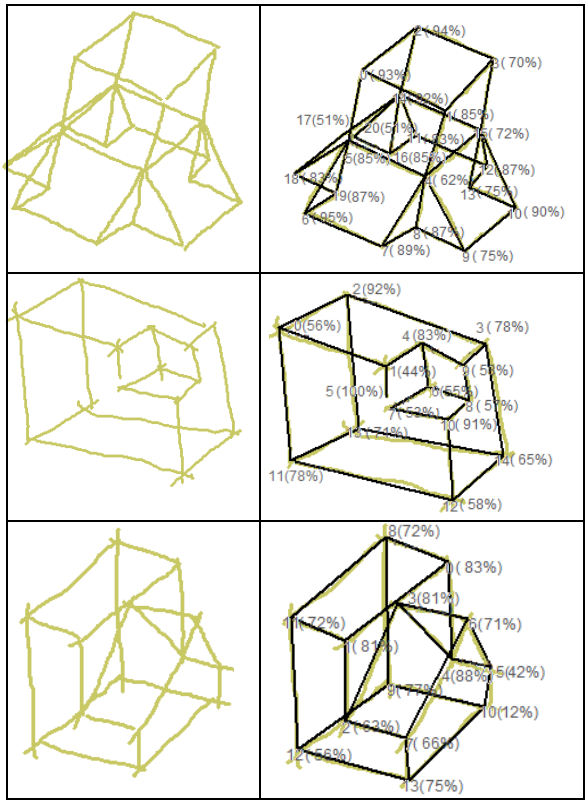


Fig. 8. Careful and casual sketches of 3D polyhedral shapes, with their merits

In most of the examples we tested, the perceptually correct solution was successfully found by the algorithm. Generally, neither false junctions were detected, nor close but different junctions were merged. In the worst cases, only certain junctions were incompletely merged (see Figure 9). Furthermore, the merits seem to be in accordance with the probability of the junctions to be perceived as such by humans. However, the algorithm is clearly sensitive to the thresholds defined as parameters (see Table 2). For this reason, some intersections perceived as junctions by humans are not perceived correctly by the algorithm when tuned with the default parameters (which were conservatively defined to prevent false positives).

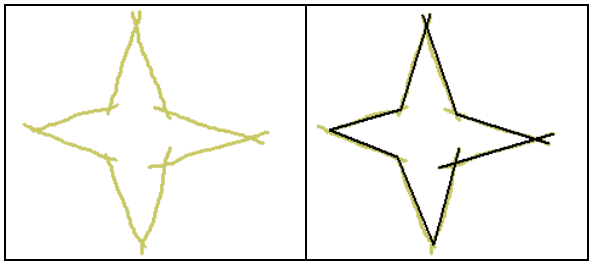


Fig. 9. Casual sketch out of range for the proposed thresholds

Tips of exceedingly casual sketches can still be safely grouped—thus reducing false negatives—if the thresholds are increased, which is acceptable only if different corners are not too close to each other. For instance, by increasing parameter *maxDist* up to 13%, the casual star from Figure 9 can be solved correctly. The same occurs for example 6 in Figure 2, which is correctly solved by increasing *maxDist* to 10%. Similarly, example 3 in Figure 8 is correctly solved by increasing *maxDist* to 14%. However, arbitrarily increasing the parameters is prone to produce false positives, as shown in Figure 10, where the parameter *maxDist* was increased to 34% for the star in Figure 9, but barely 9% for the prism on the right side of Figure 1.

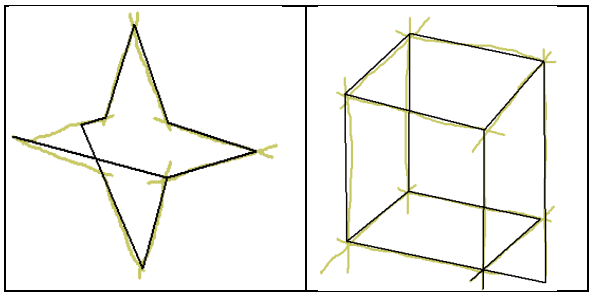


Fig. 10. Casual sketches solved with false positives

Reducing *maxDist* (from 8% to 4%, for example) would prevent the algorithm from accepting casual junctions, while increasing it (up to 20%) would allow the algorithm to detect very poorly sketched junctions (but would also produce the incorrect merge of close junctions). Thus, the parameter may vary from less than 5% for careful and/or dense sketches up to 20% for those very casual but not dense ("dense" sketches are those with nearly overlapping junctions). Similarly, reducing the valid range by half (inTol= 25% and outTol= 12%) prevents the algorithm from detecting false positives in dense careful sketches.

Varying threshold values is not recommended. Our belief is that most false positives (tips merged to define non-perceived junctions) will be hard to repair by users, whereas false negatives (junctions perceived by humans but not fully merged by the algorithm) can be clearly identified and are easy to edit manually (or in a subsequent automatic merging stage that could take advantage of global information). Still, while a balanced set of parameters is recommended, valid configurations for careful and casual sketches are tabulated in Table 2.

Table 2: Recommended algorithm configurations

Parameter	**Careful**	**Balanced**	**Casual**
maxDist	12%	8%	4%
maxRot	5°	10°	10°
RM	0.2	0.5	0.8
Valid range: inTol	25%	50%	50%
Valid range: outTol	12.5%	25%	25%
maxDistTriplets	5%	10%	15%
ETM	0.2	0.5	0.8
maxDistDangling	5%	10%	15%

The first and second configurations are valid for the two types of sketches represented in Figure 1, as the first prioritizes reification, while the second prioritizes emergence (and both configurations output similar merits). Higher values of *maxDist* are useful to detect lines that do not intersect close to each other (Figure 3 middle in [CVPC18]), but are counterproductive to detect lines with long tails (Figure 3 right in [CVPC18]), as the tips of the long tails may be close to an adjacent junction (Figure 10, right). We note that reducing *maxDist* nearly cancels reification, thus preventing poorly drawn tips of neighbor junctions from emerging to produce false reified junctions.

Examples of the type of extremely casual—and/or moderately dense—sketches parsed by the third configuration are shown in Figure 11.

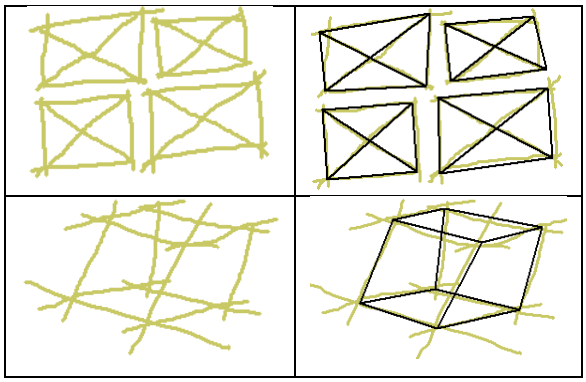


Fig. 11. Extremely casual sketches, which can be parsed by the third configuration in Table 2

We note *maxRot* is independent on the size of the line, as we realized that humans perceive and are highly sensitive to even small rotations, irrespective of the length of the line. Furthermore, we observed that people tend to accurately determine the orientation of close lines, thus allowing the algorithm for large rotations to result in merging endpoints of lines intended to be parallel. Hence, in an attempt to avoid false positives, we opted for a strategy that may produce false negatives for short, isolated and poorly drawn lines.

The established merits could be used to implement a dichotomous algorithm by simply considering valid those candidate junctions with merits greater than a fixed threshold (in accordance with Table 2). Thus, a threshold over 0.5 implies accepting only good junctions (which mainly include careful tips separated by less than *maxDist*/2), a threshold of 0.25 results in accepting good and average junctions (which mainly exclude casual tips separated by more than *maxDist*/2), and a threshold of 0 accepts all the merged junctions.

The algorithm is not intended to segment "passing" strokes (i.e., strokes that encompass two lines that should be split and connected to a close junction). For instance, the stroke that connects tips 0 and 2 in Figure 12 is not split into segments 0-6 and 6-2, which would better represent the wireframe of the depicted polyhedral shape. This is a future development that requires global considerations (regarding the so-called T-vertices, and possibly using information about faces), to prevent false positives such as the one that could easily appear in the wedge (sketch h) in Figure 2.

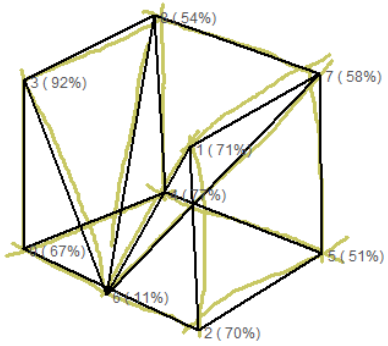


Fig. 12. Junctions (and merits) of a casual sketch of a polyhedral shape, were a passing line (0-2) was not merged to a near junction (6)

Finally, the current version of our algorithm is still unable to automatically solve extremely casual sketches with dense (nearly overlapping) junctions. This problem is not considered a limitation of our current algorithm, as it is mainly associated with casual sketches that do not depict 3D polyhedral shapes but flat patterns. Although some of these sketches could still be solved by suitably tuning our algorithm, we are still unable to automatically detect the type of the sketch. As a result, the algorithm needs to be re-tuned manually for every sketch type.

The experiment has shown that people also doubt and disagree when perceiving casual sketches, mostly if the sketches are also dense. Thus, the sensitivity of the algorithm cannot be considered a fault, as it replicates human behavior. The important thing is that the algorithm quantifies the level of uncertainty, thus allowing for a subsequent vectorization stage to solve those dubious junctions by using global information about the depicted model. We speculate that sketches that are drawn so poorly cannot be fully interpreted by providing local information only. Instead, global scene information is required for interpretation. It is only after assuming that Figure 9 depicts a four pointed star that finding their junctions becomes easy. Similarly, humans perceive the top corner of Figure 13 because the sketch is perceived as a pyramid.

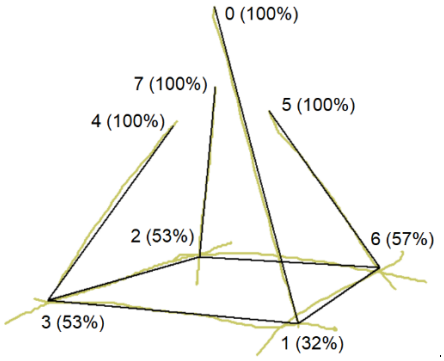


Fig. 13. Casual pyramid not solved by the default configuration of the algorithm which searches for local junctions without global information about the drawing.

5. Conclusions

Detection of vertices is an important and useful stage in Sketch-Based Modeling research. However, current methods for vertex detection are still ineffective when applied to casual sketches.

In this paper, we described an improved approach for determining the vertices of 2D line drawings. The algorithm detects junctions—which are likely to correspond to vertices of the depicted object—based on a local interpretation approach. The universe of drawings is limited to polyhedral shapes, but our method works with line drawings derived from both careful and casual sketches and does not require any information other than vectorized lines.

Our hypothesis that carefully sketched vertices must be reified, while casually sketched vertices must emerge, was indirectly validated, as it guided the design and implementation of an algorithm that improves on previous vectorization techniques, which were based exclusively on reification strategies.

The perceptual essence of the algorithm combined with its sequential detection provides efficiency, since the algorithm initially fixes the best defined junctions, which makes them difficult to distort during subsequent iterations.

Our approach quickly and correctly detects vertices in line drawings of polyhedral objects that have been vectorized from hand-drawn sketches. Vertices of polyhedral shapes depicted by highly casual sketches can still be detected provided that the sketch does not contain extremely close junctions. In addition, our tests suggest that our method, if re-tuned, may also be useful for other sketched shapes.

6. References

[1] [XCS14] Xu B., Chang W., Sheffer A., Bousseau A., McCrae J., Singh K. (2014) True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization, ACM Transactions on Graphics (Proc. SIGGRAPH), Vol.33, No 4.

[2] [CVPC18] Company P, Varley P.A.C., Plumed R., Camba J.D. Iberoamerican congress on pattern recognition CIARP18. Accepted paper.

[3] [Kan79] Kanizsa G. Organization in Vision: Essays on Gestalt Perception. Praeger, New York, 1979.

[4] [JGH09] Johnson G., Gross M.D., Hong J., Do E.Y.L. (2009) Computational support for sketching in design: a review. Foundations and Trends in Human–Computer Interaction, 2 (1), pp. 1–93.

[5] [Fer92] Ferguson E.S. Engineering and the Mind's Eye. The MIT Press, 1992.

[6] [GFP13] Governi L., Furferi R., Palai M., Volpe Y. (2013) 3D geometry reconstruction from orthographic views: A method based on 3D image processing and data fitting. Computers in Industry, 64, pp. 1290–1300.

[7] [CPVP8] Company P., Plumed R., Varley P.A.C. and Camba J.D. Algorithmic perception of vertices in sketched drawings of polyhedral shapes. Under review.