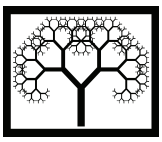

Proceedings of the Sixth International Conference
on Engineering Computational Technology,
M. Papadrakakis and B.H.V. Topping, (Editors),
Civil-Comp Press, Stirlingshire, Scotland

Paper 149

A New Sketch-Based Computer Aided Engineering Pre-Processor

P. Company[1], N. Aleixos[2], F. Naya[2], P.A.C. Varley[1]
M. Contero[2] and D.G. Fernández-Pacheco[3]
[1] Department of Mechanical Engineering and Construction
Universitat Jaume I, Castellon, Spain
[2] Department of Engineering Graphics
Polytechnic University of Valencia, Spain
[3] Department of Graphical Expression
Polytechnic University of Cartagena, Spain

Abstract

Two competing approaches exist for creating input data for Computer-Aided Engineering applications: stand-alone CAE pre-processors which define both geometric data and attributes, and the combination of CAD applications plus downstream CAE pre-processors. In the CAD-CAE sequence, the former defines the geometric information and the latter adds attributes to the imported CAD geometry. In both cases, data is input through WIMP (window/icon/menu/pointing device) user interfaces, which are appropriate during the detailed design phase, but clearly inappropriate during conceptual design.
This paper describes a sketch-based CAE pre-processor as an alternative to the WIMP paradigm. Our prototype implementation is aimed at structural analysis, particularly of 2D bar structures. More generally, we show that sketch-based pre-processors are not merely a valid option in earlier design process stages, but preferable as they increase ease-of-use of user interaction.

Keywords: sketch-based interfaces, CAE pre-processors.

1 Introduction

From the origins of Computer-Aided Engineering (CAE) in the 1950s, CAE applications have grown in power to enable today's users to simulate very complex engineering problems. The complexity of data involved in input and output processes grew in parallel with increasing "number-crunching" capability. Although the need for CAE-oriented graphical pre- and post-processors was soon realised, development of such utilities was initially slow because of the lack of hardware graphics capabilities.

In the early 1990s, Arabshahi et al [1] summarised the active components and system requirements of a more automated CAD-FEA transformation scheme. Today, all CAE applications include some sort of graphic post-processor. Indeed,

the most powerful post-processors are complete and efficient enough to cope with almost all post-processing needs [2]. Academically, the problem can be considered solved, the main open issue in this area being to ensure product data quality, in order to avoid data exchange problems and simplify integration of downstream applications into the design chain. For example, after post-processing, the user must move to the next step—usually CAM—and this requires the digital product data model (a CAE model) to be converted back to the "main view" (usually a standard CAD model) before it can be reconverted to the appropriate secondary view (i.e. the CAM model). Such transformations produce problems of ensuring data quality while "translating" between main and secondary views. We have considered these problems elsewhere [3].

In this paper we deal with the design and implementation of graphic pre-processors. This is an open issue because, in general, graphic pre-processors have not reached the "power" of general-purpose CAD applications, where "power" means capability to create and edit complex geometrical models easily in a natural environment. Engineers currently enjoy parametric solid modelling tools, but remain attached to non-parametric CAE tools [4]. As a consequence, many users feel more comfortable with CAD environments than with CAE pre-processors. Moreover, in spite of integration approaches such as [5], general purpose CAD applications do not usually include capabilities to create full sets of input data for CAE applications. Thus current CAD applications cannot fulfil CAE input tasks.

Furthermore, current "WIMP" interfaces (window/icon/menu/pointing device) for CAD and CAE applications are well-fitted for the detailed design phase, but force the user in the wrong direction during conceptual design, as they demand sequential actions and require complete and well-structured data while the user wants to consider options which meet an incomplete set of specifications. During conceptual design, sketching is more "natural" and useful than constructing complete and consistent models [6].

We have addressed these problems by designing and implementing a sketch-based CAE pre-processor. Our current aim is to demonstrate the feasibility of such an approach, as an easy-to-use alternative to the current alphanumeric inputs of some academic CAE applications presently used for research and teaching. The example tool we present here is aimed at structural analysis of input data, particularly to 2D bar structures.

In the next section, we briefly discuss the state of the art to place the problem we address in context, and we introduce our sketch-based approach. In Section 3, we describe the design and implementation of a sketch-based CAE pre-processor as a practical demonstration of our approach. Section 4 contains a brief analysis of our current results. The paper ends with lessons learned and conclusions.

2 State of the art

It has been clear since the late 1980s that the ability to apply attributes such as material properties, analysis type, loads and constraints to a solid design model is one of the most fundamental steps along the road to integrating CAD and FEA [1,7,8,9]. Today, some approaches exist for defining, instantiating, and managing

attributes in a CAD environment. Downstream CAE applications may then retrieve these attributes from the database to automate pre-processing. The attribution system results in a set of standardised attribute definitions which forms the basis for communicating attributes universally among different downstream CAE applications [4]. We note that, strictly, the standards ASME Y14.41-2003 [10] and ISO 16792:2006 [11] define *annotations* as dimensions, tolerances, notes, text or symbols visible without any manual or external manipulations, while *attributes* are not visible but available upon interrogation of the model. Users annotate sketches to define attributes. However, for the sake of simplicity, in this paper we shall ignore visibility and use the two terms as synonyms.

In spite of the advances in CAD-CAE integration, using CAD to define geometric data and then using a pre-processor to add attributes is still quite a common workflow. There are two reasons for this. First, almost all pre-processors accept input data in common CAD formats (IGES, STEP, etc), and the subsequent transformation from CAD to CAE models works well in most practical situations (although not all, as we noted in the introduction). However, when CAD geometry is imported using neutral formats, the parametrics governing the rules and relationships in the model are lost. Second, it is relatively easy to tailor different data piping flows, because communication between different modules is usually done via neutral or proprietary format text files. For example, MSC.Patran is the pre- and post-processor associated with the analysis code MSC.Nastran [2]. Their interconnections are done through proprietary-format files ("dat", "bdf", …).

The situation of academic CAE applications is more critical: most of them use text editors to create alphanumeric input files. This is obviously tedious and error-prone, although it has the benefit of saving researchers from wasting a lot of time in implementing a brand-new graphic pre-processor, or adapting their particular code to any of the expensive, huge and sometimes poorly-documented commercial solutions.

Summarising the above, we can identify the convenience of a brand new pre-processor which is easier to use and generates output files compatible with input requirements of current analysis codes.

In creating such a pre-processor, we take ideas from the field of Sketch-Based Interfaces and Modelling (SBIM). SBIM is an emerging field which aims at the goal of designing and implementing easy-to-use interfaces for conceptual design stages. Such interfaces are required because CAD systems pay more attention to producing a description of the final solution than to supporting the process of searching for that solution [12]. In the past, the lack of appropriate hardware and software prevented the development of hand-sketching graphical interactions. Nowadays, the general situation is better, as a recent survey [13] shows.

There are also precedents for using SBIM tools for CAE purposes. One of the earliest is that of Lipson and Shpitalni [14], who used a sketch-based interface to obtain three-dimensional geometry and then calculated some preliminary aspects of product cost and properties, such as number of bending operations, total facet area (for painting), total material volume, product weight and overall packing volume. The work of Masry and Lipson [15], concerned with analysis of structures, is one of the most representative. Their sketch-based interface for Computed Aided Design

(CAD) encompasses engineering design and finite element analysis: a 3D shape is created by sketching, and its structural properties can then be examined using finite element analysis. The object can be quickly modified using a pen-based interface, and the results of the analysis automatically update to match the change. Their work emphasises the reconstruction process, including an interesting approach to interpreting curved contours, and even includes a mesh generation step. It does not explore sketch-based addition of attributes to the CAD models to convert them into CAE models. Another related approach is the one by Kuester et al. [16], which allows users to sketch a structural system over a reference image of an existing structure using standard engineering symbols and nomenclature, and subsequently to simulate and analyse the responses of the sketched system under various loading conditions using a finite element approach. The reference image acts as a template for sketching an input model, and this image can be warped to show the behaviour of the structure under different load conditions. Their sketcher does not allow sketch-based addition of attributes to CAD models.

The problem of interpreting freehand drawing views mixed with engineering annotations can be traced back to the pioneering work of Hosaka and Kimura [17]. More recent work such as the SILK approach [18] explored electronic sketching aimed at conceptual design stages. SILK includes both text and graphics. It has additional advantages over traditional paper-and-pencil sketching [19]. However, SILK was not entirely menu-free, as it used a "controls window" to change modes and perform editing operations on the sketch. In addition, it was mainly oriented at design of GUIs [20]. Hong et al. [21] describe the drawbacks of sketch recognisers at this time. Recent approaches have tried to solve the problem of changing modes in sketch-based environments. One particularly interesting example is the "lasso and tap" approach of LaViola and Zeleznik [22,23], which lets users make dynamic illustrations by associating handwritten mathematics with free-form drawings and provides a set of tools for graphing and evaluating mathematical expressions and solving equations. Thus, we can see that the problem of interpreting annotated drawings is still an academic challenge, even before we come to consider the specific problems posed by sketch-based CAE pre-processors.

To sum up: designing and implementing a brand new sketch-based CAE pre-processor should be innovative, in that it should simultaneously manage geometric data and attributes in a menu-free environment. Annotated sketches can provide menu-free input for CAE purposes, but in order to interpret annotated sketches, three separate capabilities are required: interpreting strokes as geometric entities, interpreting sketched symbols and annotations as attributes, and interpreting gesture-based commands to control the workflow in a menu-free environment. Much previous research exists in all three areas [24-34], but none of this addresses the particular needs of a CAE pre-processor.

3 Our approach

The problem addressed in this paper is the automatic interpretation of those sketches containing conceptual design that typically designers draw aside to fix their ideas

before interacting with WIMP style CAE pre-processors. A very simple example is shown in figure 1.

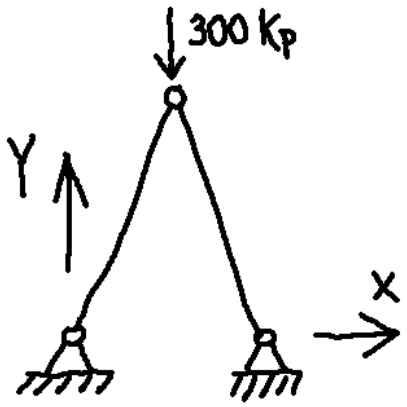

Figure 1: Sketch containing the conceptual design of a simple structure.

Our input sketches are drawn directly onto a computer screen acting as "virtual paper", not side-drawn on an actual paper sheet. We use whatever suitable peripheral is at hand (tablet PC, stylus, mouse, etc.). This requires that in designing and implementing new graphic pre-processors we must meet a new design paradigm, that of sketch-based interfaces, not WIMP interfaces. We assume that the user is still in the process of conceptual design and is not yet ready to progress to a detailed design stage. These requirements define our goals: we try to supply the user with an interface similar to classical paper-and-pencil (goal 1), and we try to minimise the amount of information the user must provide, and try to give him/her more freedom in inputting and editing it (goal 2).

Our application, *Pre/Adef*, distinguishes between geometric entities, symbols associated with annotations and gestures associated with editing tasks (i.e. "sketched commands"). The distinction follows from our conclusions in Section 2, and designing our application around it allows us to use different recognition strategies for the three types. Commands must be detected and executed immediately to edit the sketch, while geometric entities and annotations are more flexible, and could be recognised either immediately ("on-line"), or after sketching and editing tasks and just in time to produce the output file ("off-line"). Hence, our implementation comprises three "recognisers": *RecoGeo*, *RecoSym* and *RecoGes*.

RecoGeo [35] analyses and converts the sketched shape into its constituent primitives. In this first version of *Pre/Adef*, the task of *RecoGeo* is simply identifying straight lines that represent bars.

RecoSym identifies the symbols currently used to represent node constraints, loads and element geometry. The first row of symbols in Figure 2 contains examples of node types. The second row includes some symbols for specifying loads. Finally, the third row includes three examples of profile specifications.

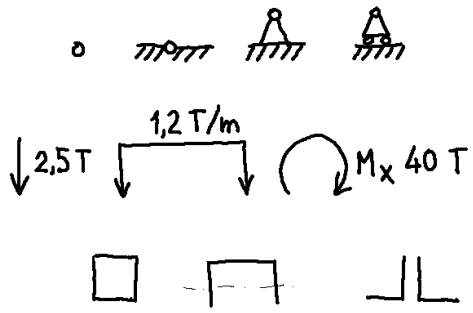

Figure 2: Examples of three groups of symbols for a sketch-based pre-processor of 2D bar structures.

It is well-known that currently CAD-centric and CAE-centric approaches coexist in the integration of CAD and CAE [5], CAD-centric approaches being those where pre-processing is mainly mapping attributes onto geometry. In this sense, our approach is not CAD-centric but CAE-centric, as symbols representing profiles are used to complete the sketched CAE-like geometry. In our application, bar elements are sketched as simple lines, and the output is a CAE wireframe model and an accompanying list of attributes. Profile specification symbols (included in the list of attributes) are subsequently used to convert the CAE model into a CAD model, as in Figure 3. This choice is a consequence of goal 2, as it reduces the input work required of the user, and limits the amount of detail required during the first, most conceptual, steps of the design process.

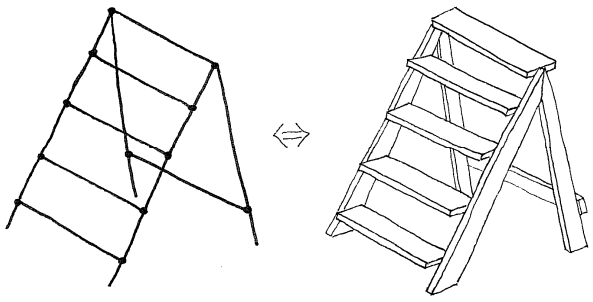

Figure 3: CAE wireframe and the corresponding CAD 3D model.

The gesture recogniser, *RecoGes*, furnishes the user with input commands to edit and manipulate the sketch. Earlier, we selected gestures as the best way to convey commands. The basic gestures which we currently implement are *erase* and *select*, respectively the "scribbles" and the "lassos" in Figure 4.

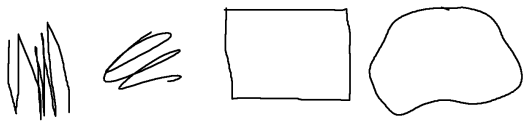

Figure 4: Gestures that control the edition of the sketch in the "virtual" paper.

Our system works in two main modes: “inking” (drawing shapes and annotations) and “editing” (sketching commands). In this preliminary version, two buttons switch between the two main modes (see Figure 5). This apparently conflicts with goal 1, but is not a return to WIMP paradigm, as the user is only asked to explicitly change the mode when moving to a very different task. Changing mode in this way is analogous to changing between a pencil and an eraser, which is a common action during a typical paper-and-pencil sketching session. In the sense that it replicates a real paper-and-pencil scenario, our virtual paper-and-pencil scenario remains compatible with goal 1, and adds some extra tools not available with real paper and pencil (*lasso plus erase* and *lasso plus copy and paste*). Although the user can usually draw an entire sketch in the piece of virtual paper viewed through the original drawing window, we have in addition included the typical tools for navigating our virtual paper (rotate, pan, and zoom) through standard WIMP controls (menu icons and mouse buttons).

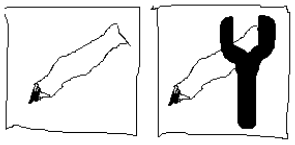

Figure 5: Buttons for switching between inking and editing modes.

Initially we used pen pressure as a mode discriminator to distinguish between two inking modes (drawing shapes vs. annotations, or strokes vs. symbols): high pressure means drawing and low pressure means annotating. However, as some peripherals (notably mice) do not detect pressure, we later included a third button to toggle between inking modes.

After processing commands, and recognizing geometric entities and symbols, *Pre/Adef* must make sense of the full drawing. We must first isolate and identify the different symbols, and then determine which annotations refer to which geometric entities (see Figure 6). Next, we must determine the meaning of each annotation, take care to include orientation information. For example, in Figure 1, the arrow of the 300 kp force is parallel (more or less) to the arrow of the “Y” axis symbol, but they have opposite senses. The vector force should thus be interpreted as (0, -300, 0). However, the moment in Figure 2 includes a sub-index indicating its intended orientation (“x”), so the orientation of the icon should be ignored, and the combined grouping interpreted as (40, 0, 0). Finally, having interpreted each group in isolation, we must combine them into a whole: we must connect the bar elements to the appropriate nodes and apply the loads to the right nodes or elements.

Our pre-processor implicitly assigns “soft” coordinates to each node. This strategy is quite common in parametric generation of 2D profiles in CAD modellers, and is useful in conceptual design, because the user is not urged by the system to define the exact dimensions. If no dimensions are given, the pre-processor adds a provisional set of valid dimensions which can be later modified by the user. The system also applies a default scale.

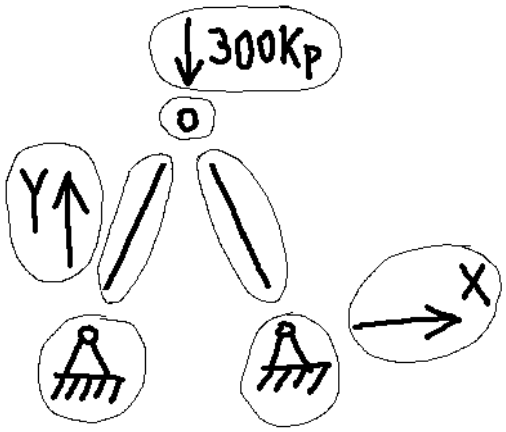

Figure 6: Grouping of symbols of a 2D truss structure sketch.

The user can add dimensions easily by sketching standard dimensioning symbols. If the user includes a few basic dimensions, the system recalculates the rest by assuming the same approximate scale throughout the drawing. The user is free to add as many dimensions he or she wants until completely constraining the model.

When the user indicates that the model is complete, we save the information contained in the database of our pre-processor to an output file meeting the specifications of the desired analysis code (see, for example, a DISSENY/ADEF-like file in Table 1).

```
>TITLE
  "2 BARS STRUCTURE"
>COORDINATES
  1   0.0   0.0  0.0
  2   0.5   1.0  0.0
  3   1.0   0.0  0.0
>MATERIALS
  1  2.1E+6
>GEOMETRIC PROPERTIES
  1    1.50E-4
>ELEMENTS
  1    1 3   1   1  0  1
  2    2 3   1   1  0  1
>CONSTRAINTS
  ALL            DZ GX GY GZ
  1       DX DY
  3       DX DY
>LOADS
  ESTATE 1
  NODE LOADS
    2  0.0  -300.0  0.0
```

Table 1: Output file for the two bars example.

4 Analysis

We tested our approach by sketching a set of examples and checking whether the output file was valid as input to the analysis code ADEF, previously developed by some of the authors [36]. Some examples are shown in figure 7.

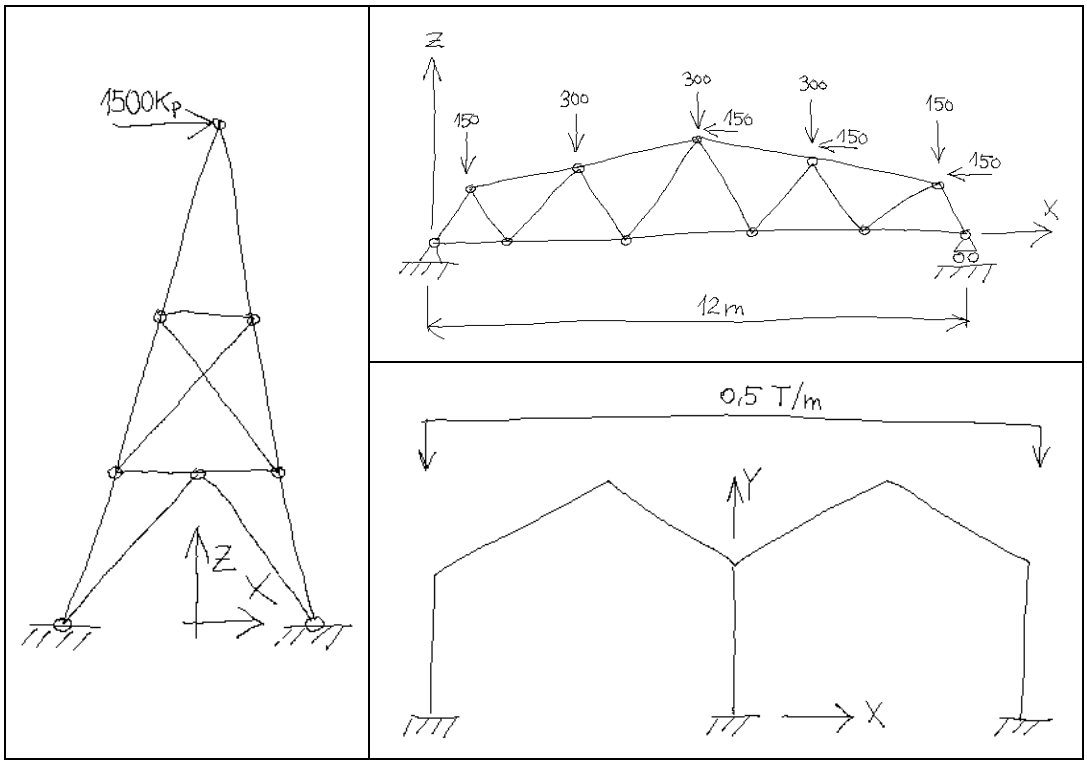

Figure 7: Examples of 2D bar structures sketched with *Pre/Adef*.

We discovered during our pilot test that users do not always feel comfortable with an on-line parser. The on-line parser which interactively interprets gestures does what the user expects: commands associated with gestures, such as *erase*, must be interpreted and executed just after they are drawn. However, changing strokes which represent bars or symbols into perfectly straight segments or neatly drawn symbols blocks the creativity of some designers, who come to feel that their actions are "definitive", as they are instantly interpreted to create a final output. The result is that some designers become more concerned with checking whether or not they could be misunderstood by the parser than with drawing the tentative design which is gradually taking shape in their imagination.

We want to retain the information unconsciously supplied by the designer while he/she draws: sequence, pauses, etc. Accordingly, what we intend to do is run the parser in the background and show the final results only when requested to do so by the user.

On the other hand, some designers, perhaps those with less sketching ability, feel quite comfortable with the "beautification" process following the parsing of every stroke. We must therefore implement both modes (on-line and off-line) before starting the formal process of validation to determine which is more useful.

Creating alphanumeric files is quite a common pre-processing strategy for academic software aimed at analysis and design of structures, and it is clear that using *Pre/Adef* is at least as easy as creating such files by hand. Accordingly, we have not performed any formal comparison of ease-of-use between our approach and previous alphanumeric input pre-processors.

However, it is not obvious that *Pre/Adef* necessarily improves usability overall. Our next step will be to compare the usability of our approach against current WIMP pre-processors, in a similar way as we did in [37].

The lasso-plus-task strategy has proved to be extremely useful in shortening the drawing process. For example, selecting the entire sketch with a lasso and adding the articulated-node symbol is an easy way to introduce articulations in all nodes (see Figure 8 left). Naturally, this does not prevent erasing or changing some nodes later. The same technique can be used to draw a set of loads (Figure 8 top right) or a section type (Figure 8 bottom right).

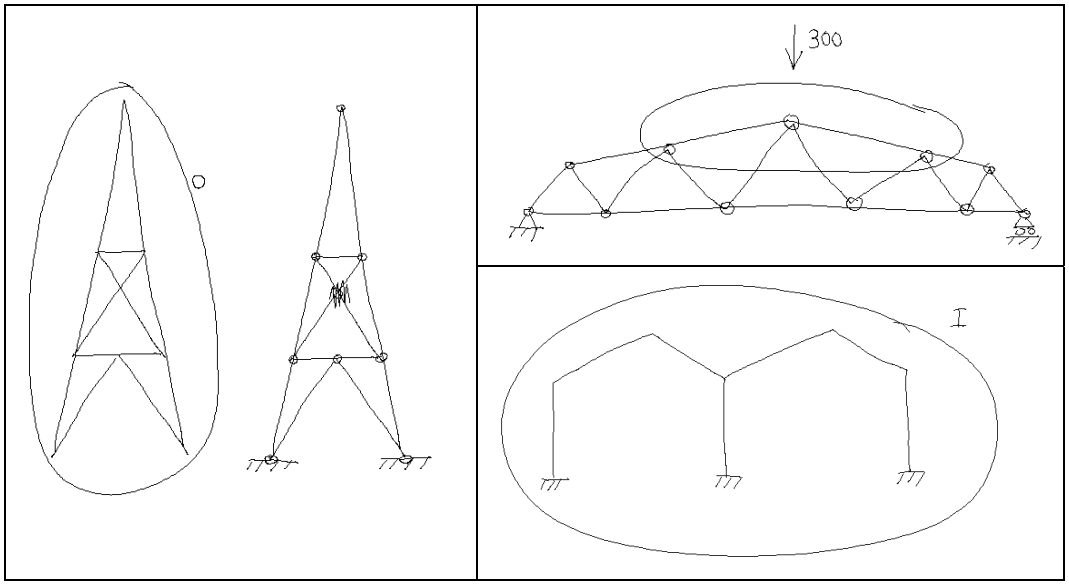

Figure 8: Examples of "lasso" used to reduce the drawing load and/or introduce properties, loads and other attributes.

In the future, we intend to improve *Pre/Adef* by taking account of regularities. Detection of geometric regularities is a major area of research in current sketch-based interfaces and modellers [38,39]. Some regularities (symmetry is the most obvious) may be of particular benefit in aiding structural analysis. Applying symmetry to simplify the analysis models is a classic strategy to reduce the computational effort in analysis of structures. For a further discussion of the

importance of symmetry in engineering design, see [40]. We could incorporate detection of bilateral symmetry (as described in [41]) into *Pre/Adef*, enhancing the calculation of provisional coordinates (by making them symmetrical) and even automatically identifying the consequent simplifications and changes of model attributes.

5 Conclusions

Although many commercial CAE applications include pre-processors capable of creating and editing complex geometrical models in more or less easy-to-use environments, most users still use general purpose CAD applications to create complex models, and then export them to the pre-processors, merely using the pre-processors to convert the CAD models into finite-element models and add annotations or attributes. Both CAD and CAE systems are still based on the WIMP paradigm, which is not the best interface for conceptual design.

We have presented a sketch-based CAE pre-processor which demonstrates the feasibility of sketch-based interfaces as a paradigm for CAE pre-processing. Our preliminary tests show that it at least equals the scope of alphanumeric input pre-processors, it is easier to learn and requires less time to prepare a finished data file.

Plane-bar structures are a particular simple case of structural design. Even modern truss-structured design applications entail non-wireframe shapes [42]. Future development of our pre-processor must consider the optimal design approach to more varied 3D shapes and topologies.

Future development of our pre-processor must also take into consideration the important fact that the specification of analysis attributes is a geometry-based task, since the structure used to support analysis attributes must be strongly tied to a geometric representation.

Acknowledgements

The Spanish Ministry of Science and Education and the European Union (Project DPI2007-66755) partially supported this work: DPI2007-66755-C02-01 (CUESKETCH: multi-agents based recognition of ideation sketches) and DPI2007-66755-C02-02 (PRESKETCH: Computer-aided prescriptive sketches system for engineering design). The support of the Ramon y Cajal Scholarship Programme is also acknowledged with gratitude.

References

[1] Arabshahi, S., Barton, D.C., and Shaw, N.K., “Steps Towards CAD-FEA Integration”, Engineering with Computers, Vol. 9, pp. 17-26. 1993.

[2] “MSC Software”, http://www.mscsoftware.com/products/core_products.cfm?Q=396

[3] Contero M, Company P., Vila C. and Aleixos N., “Product Data Quality and Collaborative engineering”, IEEE Computer Graphics and Applications. Vol. 22 no.3. pp 32-42. 2002.

[4] Baker T., “Attribution Standardization for Integrated Concurrent Engineering”, MS Thesis. Department of Mechanical Engineering. Brigham Young University. 2005.

[5] Lee S.H., “A CAD–CAE integration approach using feature-based multi-resolution and multi-abstraction modelling techniques”, Computer-Aided Design 37. 941–955. 2005.

[6] Landay J.A., “Interactive Sketching for Early Stages of User Interface Design”, PhD Thesis. School of Computer Science. Computer Science Division. Carnegie Mellon University. Pittsburgh, PA 1996. (Also appears as CMU-HCII-96-105).

[7] Shephard M.S., “The Specification of Physical Attribute Information for Engineering Analysis”, Engineering with Computers 4, 145-155, 1988.

[8] Arabshahi, S., Barton, D.C., and Shaw, N.K., “Towards integrated design and analysis,” Finite Elements in Analysis and Design, Vol. 9, pp. 271-293. 1991.

[9] Subrahmanyam S. DeVries W. and Pratt M.J., “Feature Attributes and their Role in Product Modeling”, Proceedings of the third ACM symposium on Solid modeling and applications. pp 115-124. 1995.

[10] [ASM03] ASME, “ASME Y14.41-2003. Digital Product Definition Data Practices”, American Society of Mechanical Engineers. 2003.

[11] ISO, “ISO 16792. Technical product documentation – Digital product definition data practices”, International Organization for Standardization. 2006.

[12] Jansen H., Nullmeier E. and Roediger K.H., “Handsketching as a human factor aspect in graphical interaction”, Computers & Graphics. Vol. 9. No. 3, pp 195-210. 1985.

[13] Company P., Piquer A., Contero M., Naya F., “A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling”, Computers & Graphics. Vol. 29, No 6. pp. 892-904. 2005.

[14] Lipson H., Shpitalni M., “Conceptual design and analysis by sketching”, Artificial Intelligence in Design and Manufacturing (AIEDAM). 1997.

[15] Masry M., Lipson H., “A Sketch-Based Interface for Iterative Design and Analysis of 3D Objects”, Sketch-Based Interfaces and Modeling. Eurographics Symposium Proceedings. SBM’05 pp. 109-118. 2005.

[16] Kuester F, Phair M.E. and. Hutchinson T.C., “Image centric finite element simulation”, Computers & Graphics 29, 379–392. 2005.

[17] Hosaka M. and Kimura F., “Using Handwriting Action to Construct Models of Engineering Objects”, Computer. Volume: 15, Issue: 11. pp. 35- 47. 1982.

[18] Landay, J.A., Myers, B.A., “Interactive Sketching for Early Stages of User Interface Design”, Proc. of CHI'95 , pp. 43-50. 1995.

[19] Landay J.A., “SILK: Sketching Interfaces Like Krazy”, Proceedings of Human Factors in Computing Systems (Conference Companion), ACM CHI '96, Vancouver, Canada, April 13--18, 1996. pp. 398-399. 1996.

[20] Landay, J.A. and Myers B.A., “Sketching Interfaces: Toward More Human Interface Design”, IEEE Computer, 2001. 34(3): p. 56-64. 2001.
[21] Hong J., Landay J., Long A.C., Mankoff J., “Sketch recognizers from the end-user’s, the designer’s, and the programmer’s perspective”, Sketch Understanding, Papers from the 2002 AAAI Spring Symposium. pp. 73–77. 2002.
[22] LaViola J, Zeleznik R., “MathPad2: a system for the creation and exploration of mathematical sketches”, ACM Trans Graphics. (Proceed SIGGRAPH 2004); 23(3):432–40. 2004.
[23] LaViola Jr. J.J., “An initial evaluation of MathPad2: A tool for creating dynamic mathematical illustrations”, Computers & Graphics. Vol 31. No. 4. pp. 540-553. 2007.
[24] Rubine D., “Specifying Gestures by Example”. Proc. of SIGGRAPH '91, pp. 329–337. 1991.
[25] Apte A., Kimura V.V., “Recognizing multistroke geometric shapes: an experimental evaluation”. In Proc. of ACM UIST’93, pp. 121–128. 1993.
[26] Qin S-F., Wright D.K., Jordanov I.N. “On-line segmentation of freehand sketches by knowledge-based nonlinear thresholding operations”. Pattern Recognition 34 1885–1893. 2001.
[27] Calhoun C., Stahovich T.F., Kurtoglu T., Kara L.B., “Recognizing Multi-Stroke Symbols”. Proc. of AAAI Spring Symposium on Sketch Understanding, pp.s 15–23. 2002.
[28] Zang D., Lu G., “A comparative study of Fourier descriptors for shape representation and retrieval”. Proc. of the 5th Asian Conference on Computer Vision, Melbourne, Australia, ACCV’02, pp. 1–6. 2002
[29] Yu, B., “Recognition of freehand sketches using mean shift”. In Proc. of IUI '03, pp. 204–210. 2003
[30] Pereira J. P., Branco V. A., Jorge J. A., Silva N. F., Cardoso T. D., Ferreira F. N., “Cascading recognizers for ambiguous calligraphic interaction”. Proc. of the Eurographics Workshop on Sketch-Based Modeling (SBM'04), pp. 63–72. 2004
[31] Harding P.R.G., Ellis T.J., “Recognizing hand gesture using Fourier descriptors”. Proceedings of the 17th International Conference on Pattern Recognition, vol. 3, pp.286–289. 2004.
[32] Licsar A., Sziranyi T., “Hand gesture recognition in camera-projector system”. Lecture Notes. Computer Science 3058, 83–93. 2004
[33] Hopkins J., Andersen T., “A Fourier descriptor based character recognition engine implemented under the Gamera open-source document processing framework”. Proceedings of the SPIE, vol. 5676, pp. 111–118. 2004.
[34] Park C.H., Park H., “Fingerprint classification using fast Fourier transform and non-linear discriminant analysis”, Pattern Recognition 38, 495–503. 2005.
[35] Aleixos N., Naya F., Company, P. and Contero, M., “Geometry and Gesture Recognizers for Prescriptive Sketch Interpretation”, Regeo Technical Report 03/2007. http://www.regeo.uji.es/publicaciones/regeo03.pdf. 2007.
[36] Martí-Montrull P. and Company-Calleja, P., “DISSENY: An Integrated System for the Structures and Structural Elements Optimal Design”, Advances

[37] in Optimization for Structural Engineering. B.H.V. Topping (Ed.). Civil-Comp Press. Edinburgh, 1996, pp. 23-29. (ISBN 0-948749-42-3).
[38] Company P., Contero M., Naya F. and Aleixos N., "A Study of Usability of Sketching Tools Aimed at Supporting Prescriptive Sketches", Eurographics Symposium Proceedings. Sketch-Based Interfaces and Modeling (SBIM06). (ISBN 3-905673-39-8). pp. 139-146. 2006.
[39] Lipson H., Shpitalni M., "Optimization-based reconstruction of a 3D object from a single freehand line drawing", Computer-Aided Design. vol. 11, pp. 24-36. 1996.
[40] Company P., Contero M., Conesa J. and Piquer A., "An optimisation-based reconstruction engine for 3D modelling by sketching", Computers & Graphics. Vol. 28, No 6. pp. 955-979. 2004.
[41] Tate S.J., Jared G.E.M. "Recognising symmetry in solid models", Computer-Aided Design. Vol.35, pp. 673-692. 2003.
[42] Piquer A., Martin R.R. and Company P., "Skewed Mirror Symmetry for Depth Estimation in 3D Line-Drawings", Lecture Notes in Computer Science. GREC 2003 Post-proceedings. Volume 3088. pp 138-149. 2004.
[43] Martinez P, Marti P, Querin O.M., "Growth method for size, topology, and geometry optimization of truss structures", Structural and Multidisciplinary Optimization 33 (1): 13-26. 2007.