# Operating modes in actual versus virtual paper-and-pencil design scenarios

**Pedro Company**
Universitat Jaume I
Dpt. Mechanical Engineering and Structures
pcompany@emc.uji.es
+34 964728119

**Peter Ashley Clifford Varley**
Universitat Jaume I
Dpt. Mechanical Engineering and Structures
varley@emc.uji.es
+34 96472820

**ABSTRACT**
Conceptual design remains "unplugged" from other computer-aided design tasks. Engineers and designers continue to favor pencil and paper over computerized sketching tools. It has been suggested that one reason for this is the bewilderingly large range of options available to users of CAD tools. The simplicity of pencil and paper, it was argued, is a virtue, not a drawback.

In this paper, we show that, in the hands of a skilled user, a pencil is a complex tool in itself, capable of several different modes of operation. We illustrate this versatility with examples from the field of mechanical engineering. To be competitive, computer-aided sketching tools must support the various operating modes which pencil and paper allows and skilled design engineers habitually use. We discuss to what extent this is already the case, and what further steps can be made towards this goal.

**Author Keywords**
Engineering design. Conceptual design. Sketch-based modeling. Input modes.

**ACM Classification Keywords**
H.1.2.b. Human-centered computing. H.5.2.f. Graphical user interfaces. I.7.5.c Graphics recognition and interpretation.

## INTRODUCTION: PENCIL AND PAPER VS COMPUTER

A pencil costs approximately €0.50; a tablet PC suitable for sketching costs over €1000. A pencil weighs 30 grams (1 ounce); a tablet PC weighs over 50 times more. A pencil can be replaced immediately; replacing a computer is expensive and time-consuming. A pencil works straight out of the pocket; a computer takes five minutes to power up, and longer when being used for the first time. Why would anyone use a computer for sketching when pencils are available?

Ideally, if current non-invasive computer-aided sketching (CAS) tools were genuinely comparable with actual paper and pencil, users would use them, since the computer provides other potential advantages. There is a broad consensus on some of the main advantages of computer-based systems:

- Work is easier to edit.
- Work is easier to file.
- Work is easier to interface to other applications.

However, replicating all the capabilities of actual paper and pencil is not so easy. Pencils are surprisingly flexible: a pencil can be used without training by a novice user, but (as we shall show in Section 2) is nevertheless a powerful tool in the hands of an experienced user. In contrast, CAS tools can be bewildering to novice users, and several authors have suggested that it is this bewilderment which prevents the wider acceptance of CAS [1]. At the other end of the scale, it is sometimes tacitly assumed that CAS tools provide all the operating modes which an experienced user requires. As we shall show in Section 3, this assumption is questionable.

For the computer to replace actual paper and pencil during conceptual design, its interface should become as “paper-like” as possible (hence the goal of “menu-free” applications—paper has no menus). However, computers must interpret not only freehand drawings but also annotations, sketched commands and other interactions.

Each of these distinct interactions can be regarded as an *operating mode*. As we shall show in Section 2, operating modes have been around for some time—they were not invented for CAS tools. However, in the context of CAS, the presence of distinct operating modes has been identified as a main cause of the unfriendliness of current virtual paper-and-pencil interfaces. Ideally, CAS systems should automatically detect changes of operating mode.

In this paper we address the subject of operating modes in sketch-based modeling (SBM) environments. First, we analyze the large variety of activities in actual pencil and paper scenarios. Next we review the related works in virtual sketch-based modeling. We conclude by highlighting the main problems still unsolved.

## MODES IN REAL PAPER-AND-PENCIL SKETCHING

In this paper, an *activity* is a set of actions which accomplish a job. A *task* is a member of such a set. Two main *activities* are commonly perceived while designers sketch in the considered domain of conceptual design of engineering products: *inking* and *erasing*. However, we shall see that a more detailed analysis of sketching process reveals the existence of further important activities and sub-activities.

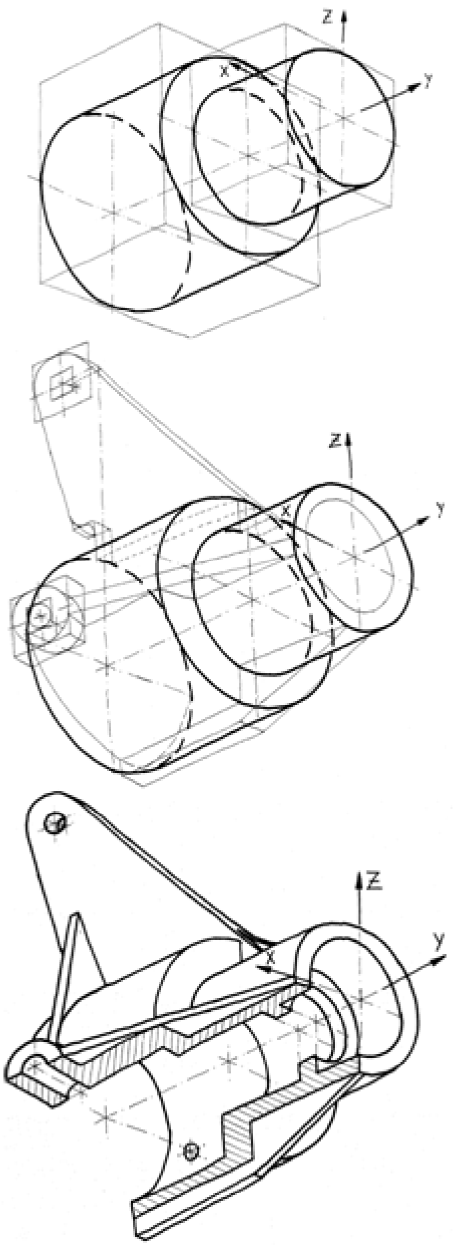

Figure 1. Auxiliary lines "scaffolding" a complex shape

### Inking modes

Sketches are usually drawn iteratively. The initial lines create a simple skeleton or scaffold to hold the desired shape. The process is illustrated in the sequence depicted in Figure 1, where it can clearly be seen how the initial parallelepiped shapes are used as frames to draw the cylindrical shapes, which in turn are complemented with other cylindrical shapes and flippers.

The sequence is shown in separated images for clarity, but in real sketching, all three figures would have been superimposed (Figure 2).

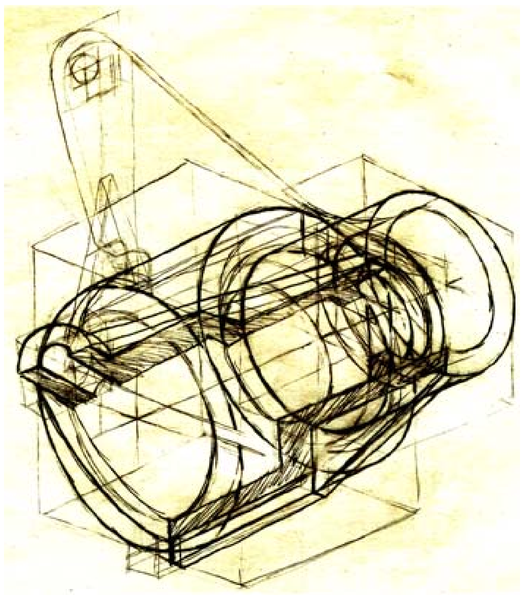

Figure 2. Auxiliary lines "scaffolding" a corner plate

Designers are not confused by their own superimposed lines, as they compare the drawing with their own mind's eye model. They distinguish *main* from *auxiliary* lines, usually through thickness: thick lines are main lines, and thin lines are auxiliary ones. The way to switch from thin to thick lines varies. Some designers change from hard tip pencils (e.g. 2H type) to soft ones (e.g. 2B). Others, instead, use the same pencil, and simply modify the pressure: high pressure for thick lines and low pressure for thin ones.

One important fact to note here is that switching from thin to thick lines changes the *operating mode*. Regardless of how the change is effected, drawing thin lines is "scaffolding", and drawing thick lines is "highlighting". The distinction is obviously rooted in perception: readers may clearly distinguish the two cylinders drawn in the upper part of Figure 1 as noticeably independent from, and more important than, the parallelepiped scaffolding that surrounds them.

Another important fact to be noted is that switching between both modes does not interrupt the ideation flow. On the contrary, we assume that the physical act of increasing pressure or switching pencils for highlighting some lines helps designers to "reinforce" their ideas [2].

The strategy of switching between scaffolding and highlighting is clearly perceivable for simple representation of engineering parts, where most if not all thin lines are part of the skeleton. Unfortunately, discrimination between modes is not always so easy: first because attributes are not always consistently maintained, and second because the same attribute may be shared for different purposes. Figure 3 shows an example where thin lines are indistinctly used to represent construction lines, hidden lines and axes. In computer science terminology, the thin line action is *overloaded*. Worse, the thin line action is not used consistently, as some (but not all) hidden lines are drawn using dashed lines, and some axis lines are drawn using dash-dot lines. However, humans perceive the different meanings and make sense of the whole drawing. Consequently, their experience of reading drawings helps them distinguish between scaffoldings and drawing axes. And the same happens between visible and hidden edges.

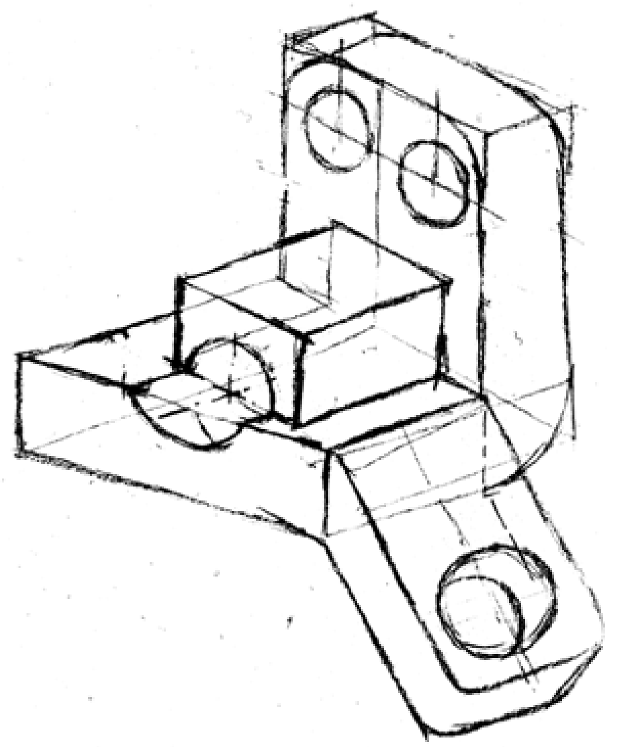
Figure 3. Auxiliary lines “scaffolding” a corner plate

According to accepted drawing standards, hidden edges and contours should be always represented by dashed lines, whether thin or thick. Obviously, using dashed lines for representing hidden edges is “highlighting the rear side”, as they convey information about occluded edges. So, switching from continuous to discontinuous lines visualizes the change of activity that happens when the mind’s eye of the designer moves from the front to the rear side of the part. However, Figure 3 shows clearly that designers may switch their minds from visible to hidden edges, without switching the hand from continuous to discontinuous lines. One plausible reason for not using dashed lines is simply that drawing discontinuous lines slows down the inking task and breaks the ideation flow. Even so, designers like to see some visual difference, so they change from thick to thin lines (which are faster to draw than dashed lines).

This overloading of thin lines does not bewilder designers, who interpret correctly their own drawings as they are linked to their memories. But what about other observers who must interpret the finished drawing? While the drawing does not explicitly communicate the meaning of individual thin lines, it appears that the human ability to perceive 3D structures from 2D drawings [3] allows observers to disambiguate hidden lines and scaffolding lines. This means that the change of mode actually happens in the human vision process, not on paper, as it is deeply linked with our perception of depth.

Axes differ from scaffolding, in that they convey different and sometimes richer information. They not only help in fixing the position of some particular lines, but are signals of the existence of some kind of complex relationships, such as axis of revolution or plane of bilateral symmetry. Again, designers find it useful to draw such lines with a different but fast line type (i.e. thin continuous line), instead of the standardized, slow-to-draw dash-dot thin line which should properly be used. However, other observers still succeed in assigning the right role to such lines, since (for example) they perceive the existence of revolved elements and, consequently, interpret the auxiliary line as a revolution axis, or detect a bilateral symmetry and interpret the auxiliary line as a bilateral symmetry axis.

*Overtracing* or redrawing is a natural way in which designers allow designs on paper to interact with designs in their minds [4]. Examples of overtracing can clearly be seen in the sketched part of Figure 3. Overtracing appears in both scaffolding and highlighting, and is different both visually and conceptually from highlighting. Highlighting specifically produces tidied line drawings over “dirty” scaffolding lines. Overtracing is more flexible. It has been interpreted by some authors as a strategy for preserving the designer’s mental image of the object [5]. In fact, we can distinguish different kinds of overtracing: *decoration* (introducing shadows, textures and so on), *thinking over the line* (thinking about the design without stopping the inking process) and *auto-correction* (perceiving that the line is being drawn with error and trying to correct it on the fly). Overtracing for decoration conveys extra information (e.g. curvatures), while *thinking* and *auto-correction* overtracing should be interpreted as simple lines.

The simplest kind of overtracing for decoration is that of hatching cross-sections. If carefully executed it is easy to perceive (Figure 1 bottom), but this clearly slows down the drawing process. In practice, it is commonly done carelessly (as can be seen in Figures 2 and 4). Hence, it may be mistaken with other types of overtracing (i.e. differences between hatching and other types of overtracing in Figure 4 are very subtle and rely mainly on the context).

Several other operating modes are visible in the detail of a timber roof given in Figure 4, where different types of thin lines appear as “bubbles” that delimit areas which require greater detail, in symbols such as “+”, which conventionally represents the head of a nail, and even in the arrows which tie annotations to the specific part of the drawing being annotated.

Despite this extreme overloading of line types, Figure 4 remains comprehensible. Something must be there implicitly helping the observer to disambiguate such lines. Our hypothesis is that this “something” contains at least three components: (a) the context (the human ability to interpret 2D drawings as representing 3D objects, as mentioned before), (b) a formal code (obviously, the code for engineering drawings is in the drawing standards, ISO, ASME, EN), and (c) an informal convention, the culture of engineering drawing practice passed down from teacher to student [6]. Effectively, we do after all have a “mode switch” to change between different interpretations of thin lines, but this mode switch exists in the mind of the observer as a collection of rules, experiences and inferences, not in the tools (paper and pencil) which the designer used to create the drawing.

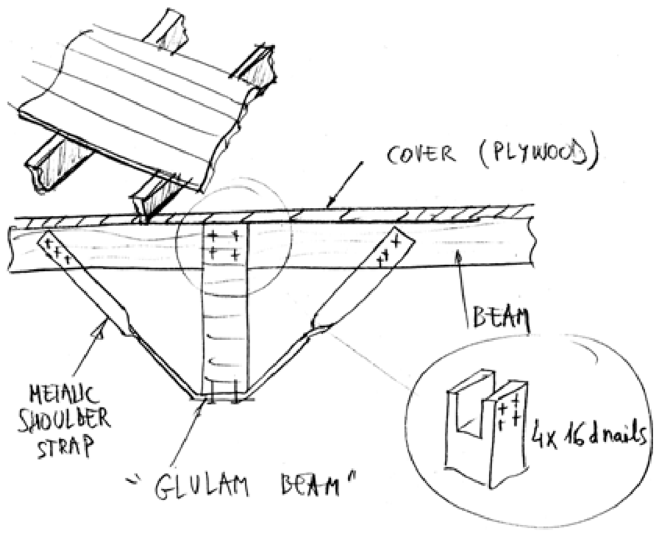

Figure 4. Annotating an engineering sketch

Finally, another interaction mode appears in Figure 4: text. Some text labels are simply names, but other text labels convey detailed information that would otherwise require high quality drawings.

It is also worth noting that Figure 4 contains an informal mixing of an orthographic view (complemented with a detailed view) with a pictorial view.

It will be noted that none of the drawings above uses color. The historical reason was the difficulty of obtaining copies of colored documents (e.g. blueprints were just blue and white). Hence formal codes were developed which excluded color. Nowadays, there is no reason why color could not advantageously be used in some circumstances without compromising the advantages of pencil and paper—colored pencils are not much more expensive than plain graphite pencils, and are just as portable. Nevertheless, design engineers do not use them. This reinforces our hypothesis that culture is an important factor in creating and interpreting drawings. Colored pencils are simply not part of the design engineer’s culture.

To sum up our analysis of inking modes, we find that many different inking activities add value to the drawing: inking construction lines (“scaffolding”); inking main lines (“highlighting”); inking hidden edges (“highlighting the rear side”); inking axes; hatching; overtracing for decoration, and annotating.

**Editing modes**

Erasing is the clearest example of editing task. Erasing is clearly different from inking. The designer even requires a different instrument to erase (e.g. the rubber eraser, or the correction fluid visible in the inner circles of Figure 5).

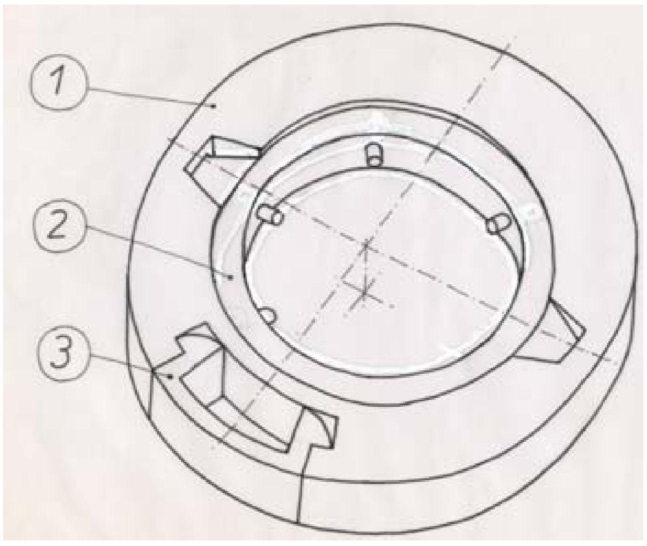

Figure 5. Sketch of a tape roller corrected with correction liquid

Changing the instrument is an obvious cue to a change of mode. Again, the physical action of changing tools does not interrupt the ideation flow. The designer realizes that the sketch contains an error and voluntarily starts an action to correct it. Or, the designer realizes that the sketch is too messy and contains too many unnecessary lines, and starts an action to clean it. In both cases, the physical action runs in parallel with the mental flow, and does not interrupt it.

Cut and paste is another common editing action. Most design sketches are created during ideation stages. In fact, one of the main purposes of engineering sketches is to help the designer to fix the detailed design. Paradoxically, the sketches become messy and disordered at the same time as the designer reaches in his mind's eye a clear and detailed design. At this point, a clean and tidy sketch would be helpful to share the detailed design with others. Some designers use "hard" cut and paste to create a "collage". Again, the creation of the patchwork means a change of mode of interaction between the user and the paper, and requires different tools (scissors and tape). And, again, this change of mode does not interrupt the ideation process. On the contrary, it helps the designer to freely clean and tidy up the image when he or she likes. Figure 6 shows a photographic image of one such a patchwork, in which the cut and paste process remains visible.

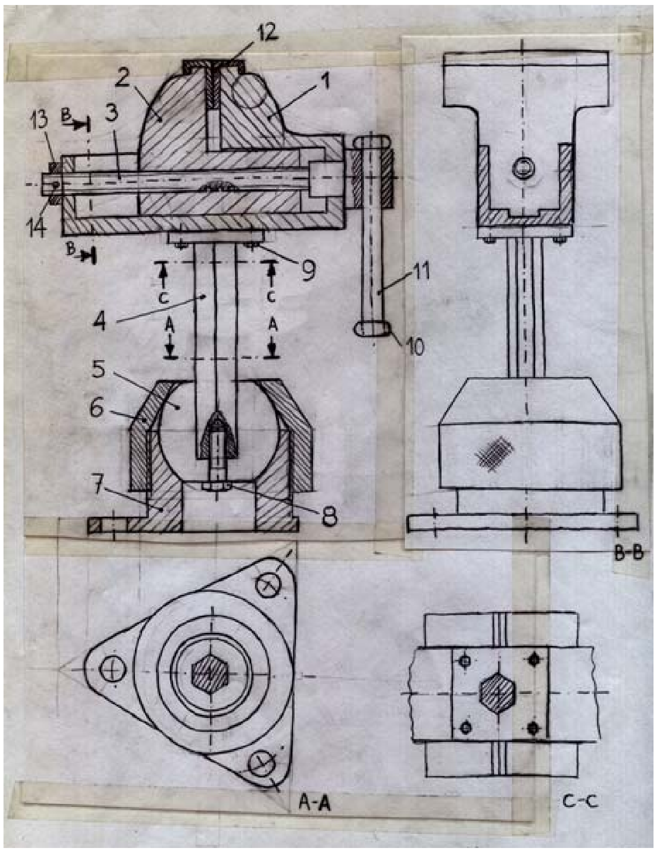

Figure 6. Sketch of a vise rearranged with hard cut and paste (notice the tape)

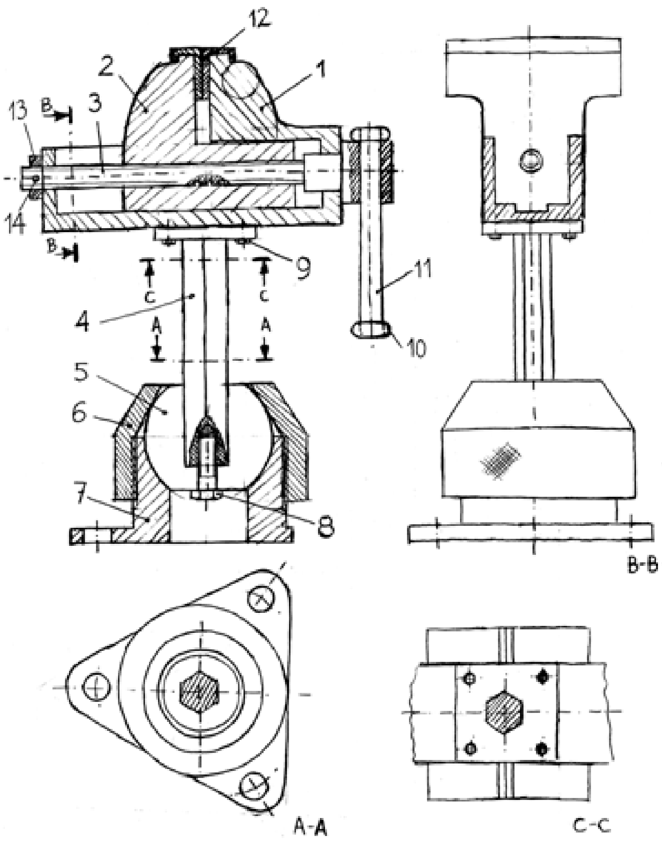

Figure 7. Sketch of a vise tidied up by a suitable b/w scanning

Figure 7 shows the tidied appearance of the vise plan after photocopying it with black and white quality, i.e. after "filtering" it. Again, the availability of the required tool (photocopying machine in this case) is the major factor limiting those editing modes that have been described as commonplace for designers and architects [5].

Other activities such as copying with symmetry are less frequent, simply because they are more sophisticated. For example, obtaining bilateral symmetry involves photocopying over a transparent sheet and then photocopying again after pasting the original with the reversed copy. Hence, the copy-reverse-and-paste is only done when the drawing is very complex, or the final version is required. Figures 8 and 9 show an example where an incomplete sketch was used to obtain a tidied and complete sketch, by simply copying some features that had been drawn one time in different locations.

The copies were done though transparent paper superimposed and conveniently displaced over the original drawing. Otherwise, the symmetry should have been indicated through the suitable axis and/or annotation, but the final shape would have never been drawn.

Again, we conclude that editing activity is commonly subdivided into erasing, cut and paste and other activities only limited by the "hardware" availability (rubber eraser, scissors and tape, photocopying machine, etc.).

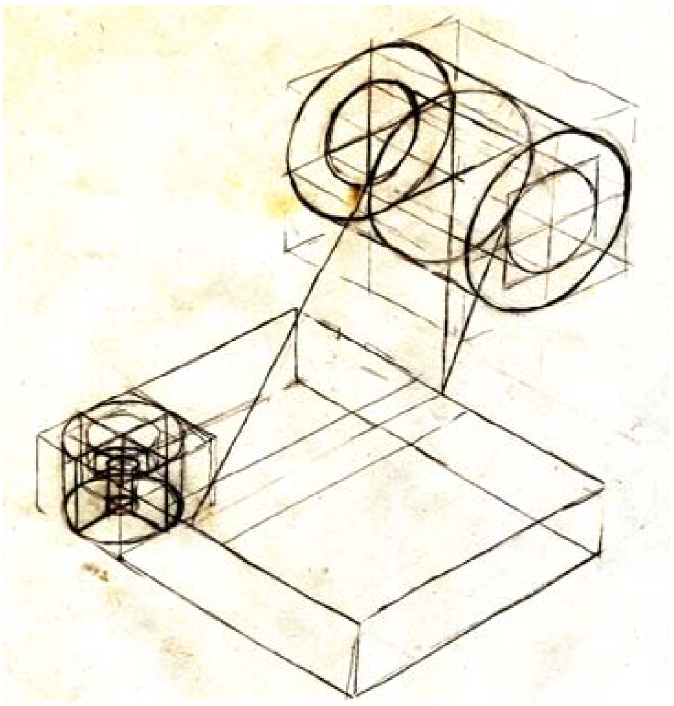
Figure 8. Original incomplete sketch of a symmetric part.

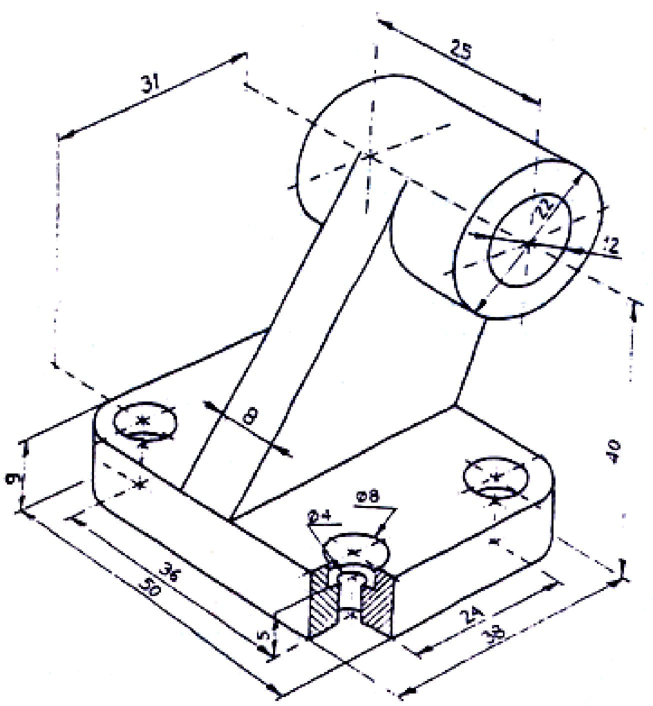

Figure 9. Part drawing completed by a "copy and paste" done though a transparent paper superimposed and conveniently displaced over the original drawing

## STATE OF THE ART IN CAS TOOLS

Sketch-based interfaces aimed at computer-aided *conceptual* design were launched in parallel with the early development of CAD tools addressed at aiding during detailed design stages. Important early contributions by Negroponte and Herot can be found back in the 70's [5, 7, 8]. Nevertheless, the conclusion, according to the same Herot [9], was that any further advances in sketch recognition would require both advances in hardware (which have happened) and advances in artificial intelligence (which, in general, have not happened). Hence, it was not until the 90's, with the advent of appropriate hardware (e.g. tablet PCs), that the "sketch-based modeling" concept began to grow through works like those by Egglí et al. [10] and Turner et al. [11].

Non-invasive sketch-based applications that assist users during conceptual design stages are called "paper-like" [2] "magic paper" [12], or "virtual paper and pencil". We refer the reader to two recent papers, [13] and [14], which respectively discuss the states of the art of sketch understanding and sketch-based modeling.

User studies (e.g. [15]) assert that current computer CAS tools are still less usable than paper-and pencil sketches and do not posses significantly improved functionality. We believe that replicating as many as possible of the actual operating modes used in a paper and pencil environment, and switching between them in a natural way, would be helpful in reducing the gap.

### Operating modes

*Operating modes* distinguish the various configurations under which the same device works to produce different effects or perform different tasks.

In considering inking modes, any reasonable sketching tool will supply a variety of these. The important ergonomic question is: how invasive is the mechanism for switching between these modes?

Two distinct paradigms continue to dominate human-to-computer interaction: command-driven (such as UNIX's *shell*) and menu-driven (such as Window's *Explorer*). Although menu-driven systems are simpler and easier to learn, they are generally not as flexible as command-driven systems. Hence, with the advent of handwritten text recognition and sketch-based input interfaces, the paradigm has been that of so-called "menu-free" applications where interaction takes place via hand-sketched commands.

Which is more appropriate to CAS? True command-driven interfaces such as *shell* are marketed at advanced users, not at beginners. However, given the limitations of OCR, handwritten interaction is greatly restricted as compared with such interfaces. It is not clear what community would find such a reduced command-driven interface useful.

In contrast, menu-driven interfaces are marketed at relative beginners (although, can be seen from the existence of commercial training courses which train people to use menu-driven software, they are in reality far from ideal for true beginners). There is clearly a market niche for very simple interfaces for true beginners, and sketch-based interfaces could reasonably fill that niche.

What is missing is an interface paradigm which would suit experienced design engineers. Identifying such a paradigm is a difficult task in itself. We can take an initial step in this direction by analysing what the two current dominant paradigms have to offer.

The two paradigms have different approaches to switching between activities. In a menu-driven environment, menus are usually arranged in groups and sub-groups and visually presented in sub-menus or toolbars. Selecting a menu command or toolbar button automatically switches to the corresponding activity. One advantage of this paradigm is that two unrelated tasks with similar names should never be confused, as the user selects them from different activities/toolbars.

In menu-free applications, the user interacts with the computer through *gestures*, graphic symbols that convey messages. Supposedly, the message conveyed by each gesture is intuitive and unambiguous. Gestures convey commands, but do not usually convey activity switching.

It is generally accepted that the only way to avoid, or at least minimize, explicit user involvement during sketch interpretation is feeding the system with extra-knowledge. The easiest available extra knowledge is the *drawing sequence*, which has sometimes been assumed to convey the user's drawing style [5]. Drawing sequence is widely used in the low-level recognition of drawn symbols and spatial relations among them, e.g. [16] and, as an essential part of Chinese character recognition, has received much attention in the field of OCR [17]. Feature based gesture recognition methods coexists with approaches that rely mainly on sequence information. Kawatani's work, like everything else which processes Chinese characters, uses stroke order. [18] and [19] represent recent work, more directly related to sketch-based modeling, which also identifies symbols through sequential information. But, as was stated long ago [7], more extra knowledge is needed.

Knowledge about both *domain* and *activity* are commonly used to greatly improve the success ratio of current sketch-based modeling systems. Domains are usually used to constraint the universe of solutions. Some domains considered in the literature are: two-and-one-half dimensional engineering parts [20]; tree diagrams, rooms, and poly-lines [Gro94]; and GUI's [21]. Recently, some approaches have been claimed to be domain-independent (e.g. LADDER, [18, 22]), and certainly they can be customized for different domains, using a "domain specific sketch grammar". However, developing a customizable computational approach does not mean that domains do not exist. On the contrary, they implicitly accept that different domains exist and their differences affect the SBM flow.

Activities and their consequent operating modes have also been considered in the SBM literature. For instance, the SILK approach ([21, 23]) explored electronic sketching aimed at conceptual design stages. SILK includes both text and graphics, and uses a "controls window" to change modes and perform editing operations on the sketch. However its domain is the design of GUIs [24], and it only deals with sketches of 2D figures. Plimmer and Apperley [25] in its pursuit of a "deliberate minimalist environment" distinguished three modes: drawing, handwriting and editing. Hammond and Davies [18] discriminate between "sketching (pen gestures intended to leave a trail of ink) and editing gestures (pen gestures intended to change existing ink)". Comparing these limited lists of modes with the variety described in Section 2, we can conclude that further work is required.

The unsolved problem of replacing the switching strategy implicit in the menu arrangement by a transparent, easy-to-use one in the context of menu-free systems is perhaps the key to obtain a fluent interaction. To this end, different strategies have been suggested. For example, [26] use gestures to issue commands to their sketching tool. However, switching between different operating modes uses a different mechanism, that of changing color. At present, their work is limited to two colors, but there is no reason in principle why it could not be extended to more. Whether (a) color would be culturally acceptable to design engineers, who are accustomed to working in monochrome, and (b) how a system would react to "deliberate mistakes" such as those in Figure 3 where thin lines are used to represent hidden edges and axes, remain open questions.

One particularly interesting example is the "lasso and tap" approach of LaViola and Zeleznik [27, 28], which lets users associate handwritten mathematics with free-form drawings. The lasso plus erase and lasso plus copy and paste have proved to be extremely useful in shortening the drawing process in a system aimed at pre-processing structural information [29].

**Additional features in virtual paper and pencil**

Obviously, there are things which can be done on a computer which cannot be done using pencil and paper. We have already mentioned filing and interaction with other applications.

For example, many CAD applications support the useful tasks of extruding and sweeping 2D profiles to produce 3D objects. This would clearly be useful, for example when processing the counterbored holes in Figure 9 to create a solid model.

When discussing Figure 8, we noted that enforcement of symmetry was rarely done in a pencil and paper environment, not because design engineers do not want to do it (they do) but because the tools available in that environment do not readily support it. Creation of bilateral symmetry is straightforward in CAS (after drawing half an object, the user marks a mirror plane, and the system creates the rest of the object by reflection through the mirror plane), and this is one area where CAS is clearly superior to pencil and paper.

Despite these additional features offered by computer environments, it remains the case that they will not supersede pencil and paper until they are competitive with its core actions, those described in Section 2.

## POSSIBLE ADVANCES

The final goal is designing a computer-based (i.e. “virtual”) paper-and-pencil environment that equals or surpasses actual paper and pencil, in order to get most designers to abandon actual in favor of virtual paper and pencil for conceptual design purposes.

To this end, the strengths of actual paper and pencil should be replicated in the virtual scenario. Specific strengths of actual paper and pencil not yet fully replicated in the virtual paper may require hardware improvements. For example, tablets have been reported to be less comfortable to use than pencil and paper because of the small gap (both in time and distance) between the cursors and the pencil tip [15].

Additionally, corresponding improvements in the software are also required: use and maintenance of computers still requires technical knowledge that some designers, quite rightly, do not see as part of their job. When starting a new virtual drawing becomes as simple as taking a new physical piece of paper from a pad, designers may be persuaded to look on computers as a tool rather than an obstruction.

Apart from trying to give users more freedom in inputting and editing in a virtual paper and pencil scenario, two other advances seem relevant and accessible: replicating as many modes as possible from actual paper and pencil, and finding the best mode-switching strategies.

### Advances in replicating modes

We still require a full taxonomy of operating modes, including their mutual relationships and descriptions of cues used to discriminate between them. The list in Section 2 is illustrative of the nature of the problem, but far from exhaustive.

Designers usually distinguish main from auxiliary lines through thickness: thick lines are main lines, and thin lines are auxiliary ones. Automatic interpretation of such sketches (like Figure 2) should require some sort of strategy to distinguish auxiliary lines from main lines, similar to the ones employed for SBM overdrawing approaches [30].

Some researchers have suggested that sequence may help in discovering the most important lines, e.g. “presuming that the most recent lines are the most intended” [5]. However, humans routinely interpret drawings without knowing the drawing sequence.

Other researchers have developed virtual drawing environments sensitive to pressure. This has proved useful, but some problems remain unsolved: 1) some devices are insensitive to pressure; 2) some designers do not use pressure to distinguish between thin and thick lines (preferring, for example, different types of pencil); 3) overtracing is still difficult to distinguish from highlighting. Note that interpreting overtracing in freehand sketches (as per [31]) is a solved problem *only* for sketches that contain no auxiliary lines.

Distinguishing the various possible implications of auxiliary lines will be more difficult. Some of this can be rule-based, incorporating the various engineering standards. But, as shown in Figure 3, such standards are not always followed—there are unofficial conventions as to which rules can be broken, and when. Engineering culture is an equally important factor, and one which has barely been investigated. The third important factor, the human vision process, has been the subject of many investigations, but few conclusions have been reached.

Other interactions modes require combined efforts. Interpreting drawing labels from line drawings is more difficult than interpreting plain text, for several reasons. Firstly, labels use a particular syntax that prevents syntactic parsers from making the right choice when different plausible interpretations appear. Developing OCR parsers tailored to cope with such “telegraphic” messages written in engineering jargon would improve the success ratio of the recognition of engineering drawing labels. Secondly, labels of engineering drawings contain interspersed symbols—they are closer to mathematical notation than to plain text. OCR tailored for mathematical formulations should perform better than general purpose OCR. The “shape descriptions” approach of [32] looks suitable.

In conclusion, to obtain improved virtual paper-and-pencil scenarios, operating modes are required. The ideal is to find the optimum number of modes: a set which includes all those modes which designers require, and excludes all those which designers do not require. But optimization is different from the current paradigm of making them disappear: less is better, but nothing is worst!

### Advances in switching modes

If each gesture replaces one menu-like command, gesture management becomes increasingly complex as the total amount of commands grows. The number of different labels in a menu-driven application has few practical limitations. The number of different icons in toolbars has more practical limitations: one cannot fill the entire screen with toolbars, as most of the area must remain available for drawing. Gestures, by contrast, are limited by our ability to recognize them and to distinguish between them. Misinterpretations and misunderstandings due to similarities between different icons or due to cultural differences between different users have been reported [33, 34].

Limiting the number of gestures that may be easily remembered and clearly distinguished from each other drastically reduces the total set of available commands.

Worse still, in some cases, only the context makes the difference between distinct useful gestures, or parts of a depicted figure. For example, differences between “zero”, “circle” or “lasso” do not depend on the shape itself, but on the context, including the scale (both absolute and relative to other nearby ink), the orientation (both absolute and relative to other nearby ink), and the interpretation(s) of other nearby ink.

Two questions must be posed. Firstly, how many functions can be provided without buttons and menus? Secondly, how many functions does a design engineer require? If the second answer is larger than the first, we need a new paradigm.

To sum up: recent approaches have recognized the need for friendly ways of changing modes in sketch-based environments, but none of them has proved to address the full control mode problem while being clearly better than the others. More importantly, none of them as yet beats actual pencil and paper.

## CONCLUSIONS

It is obvious that computers will remain less cheap, portable and simple than pencil and paper. The undeniable advantages of computers (CAS tools make work easier to edit, easier to file, and easier to interface to other applications) do not as yet compensate for their inherent disadvantages.

According to the dominant paradigm, paper-like interfaces are seen as the best way to get user-friendly computer applications which help engineers and designers during conceptual design stages. Modeless applications are assumed to be the goal.

We argue here that *full* modeless applications are not the final goal. If we wish to replicate real paper-and-pencil scenarios in virtual environments, we must be aware that actual paper-and-pencil scenarios include a rich variety of different modes.

To support our assertion that multiple modes are used for sketching engineering parts and assemblies with actual pencil and paper, we have described some typical actions commonly done by designers while sketching.

In reviewing the current state of the art of virtual paper and pencil, we conclude that replicating paper-and-pencil scenarios in virtual environments is still unfeasible. Although the goals of CAS have supposedly already been accomplished, practical implementations have resulted in unfriendly interfaces.

The way forward is (a) to replicate as many modes as possible from actual paper and pencil, and (b) to find a non-intrusive switching strategy. Determining the real requirements through further field studies with actual designers seems a necessary first step in this direction.

## ACKNOWLEDGMENTS

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. Lim S., Qin S.F., Prieto D., Shackleton J. A study of sketching behaviour to support free-form surface modeling from on-line sketching. Design Studies 25, (2004), 393–413.
2. Gross M.D. and Do E.Y.L. Ambiguous Intentions: A paper-like interface for creative design. Proc. of 9th Annual Symposium for User Interface Software and Technology (UIST), 1996, pp 183-192.
3. Hoffmann D. Visual Intelligence. How we create what we see. Norton Publishing. 1998
4. Do, E. and Gross M.D. Drawing as a Means to Design Reasoning. In Visual Representation, Reasoning and Interaction in Design Workshop notes, Artificial Intelligence in Design '96 (AID '96), 1996, 22-27.
5. Negroponte N Sketching: A Computational Paradigm for Personalized Searching. JAE, Vol. 29, No. 2, Describing Places (1975), pp. 26-29. http://www.jstor.org/stable/1424480
6. Bertoline G.R., Wiebe E.N., Miller C.L. and Nasman L.O. Technical graphics communication. Irwin, 1995.
7. Herot C.F. Graphical input through machine recognition of sketches. Proceedings of the 3rd annual conference on Computer Graphics and Interactive Techniques. SIGGRAPH '76. ACM Press. (1976) pp. 97-102. http://doi.acm.org/10.1145/563274.563294. ACM SIGGRAPH Computer Graphics, Volume 10 Issue 2.
8. Herot C.F. Sketch recognition for computer-aided design. UODIGS '76. Proceedings of the ACM/SIGGRAPH workshop on User-oriented design of interactive graphics systems. (1976) pp. 31-35.
9. Herot C. Christopher Herot’s Weblog. Some insights into communication and social media. Sketch Recognition. (2008) http://herot.typepad.com/cherot/2008/09/sketch-recognition.html
10. Eggli, L., Bruderlin, B.D. and Elber G. (1995). Sketching as a solid modeling tool. Symposium on Solid Modeling and Applications - Proceedings, pp. 313-321
11. Turner A., Chapman D. and Penn, A. Sketching space. Computers and Graphics 24 (6), (2000), pp. 869-879
12. Davis, R. Magic Paper: Sketch-Understanding Research. Computer. Vol. 40, Issue 9, (2007) pages: 34-41.

13. 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, AAAI Spring Symposium. 2002. pp. 73–77.
14. Company P., Piquer A., Contero M. and Naya F. A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling. Computers & Graphics. Vol. 29, No 6. 2005. pp. 892-904.
15. 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). 2006. pp. 139-146.
16. Gross, M. Recognizing and Interpreting Diagrams in Design. In T. Catarci. M. Costabile, S. Levialdi, G. Santucci eds., Advanced Visual Interfaces '94, ACM Press. 1994.
17. Kawatani, T. Handwritten Kanji Recognition Using Combined Complementary Classifiers in a Cascade Arrangement, in Proceedings of the 51h ICDAR, Bangalore, (1999) 503–506.
18. Hammond, T., and Davis, R. 2003. Ladder: A language to describe drawing, display, and editing in sketch recognition. Proc. of the 2003 Int. Joint Conference on Artificial Intelligence (IJCAI). (2003)
19. Sezgin, T.M., Davis, R. Sketch recognition in interspersed drawings using time-based graphical models. Computers and Graphics 32 (5), (2008). pp. 500-510.
20. Hosaka M. and Kimura F., "Using Handwriting Action to Construct Models of Engineering Objects", Computer. Volume: 15, Issue: 11. 1982. pp. 35- 47.
21. Landay, J.A., Myers, B.A., "Interactive Sketching for Early Stages of User Interface Design", Proc. of CHI'95, 1995, pp. 43-50.
22. Hammond T., Davis R.: Automatically transforming symbolic shape descriptions for use in sketch recognition. In AAAI-2004 (2004).
23. Landay J.A., "SILK: Sketching Interfaces Like Krazy", Proc. of Human Factors in Computing Systems (Conf. Companion), ACM CHI '96, 1996. pp. 398-399.
24. Landay, J.A. and Myers B.A., "Sketching Interfaces: Toward More Human Interface Design", IEEE Computer, 2001. 34(3): p. 56-64.
25. Plimmer, B. and Apperley, M. Computer-Aided Sketching to Capture Preliminary Design. In Proc. Third Australasian User Interface Conference (AUIC2002). CRPIT, 7. Grundy, J. and Calder, P., Eds. ACS. (2002) 9-12.
26. Bartolo A., Farrugia P., Camilleri K. and Borg, J. A Profile-Driven Sketching Interface for Pen-and-Paper Sketches, in VL/HCC Workshop: Sketch Tools for Diagramming, 2008.
27. LaViola J, Zeleznik R., "MathPad2: a system for the creation and exploration of mathematical sketches", ACM Trans Graphics. (Proc. SIGGRAPH 2004); 23(3):432–40.
28. LaViola Jr. J.J., "An initial evaluation of MathPad2: A tool for creating dynamic mathematical illustrations", Computers & Graphics. Vol 31. No. 4. (2007), pp. 540-553.
29. Company P., Aleixos N., Naya F., Varley P.A.C., Contero M. and Fernandez-Pacheco D.G. A New Sketch-Based Computer Aided Engineering Pre-Processor. Proc. Sixth Int. Conf. on Engineering Computational Technology. 2008. Paper-149.
30. Naya F., Conesa J., Contero M., Company P. and Jorge J. Smart Sketch System for 3D Reconstruction Based Modeling. Smart Graphics, Proceedings. LNCS. Volume 2733. 2003, pp 58-68.
31. Ku D.C., Qin S.F. and Wright D.K. Interpretation of Overtracing Freehand Sketching for Geometric Shapes. WSCG'2006, 2006.
32. Hammond T, Davis R. Interactive learning of structural shape descriptions from automatically generated near-miss examples. In: Intelligent user interfaces (IUI); 2006.
33. Tzeng O.C.S., Trung N.T. and Rieber R.W. Cross-Cultural Comparisons on Psychosemantics of Icons and Graphics. Int. J. of Psychology, Volume 25 Issue 1 1990. 77-79.
34. Piamonte P.T., Abeyseker J.D.A. and Ohlssonb K. Understanding small graphical symbols: a cross-cultural study. Int. J. of Industrial Ergonomics. 27 (6), 2001, Pages 399-404.