A contribution to the theory and practice of cognitive prostheses

Eric Neufeld, David Callele, David Mould, Sonje Kristtorn and Raymond Rabu

Department of Computer Science
University of Saskatchewan, Saskatoon, SK, Canada
{eric,callele,mould }@cs.usask.ca

Abstract. Ford and Hayes suggest that rather than build independent AI technologies, we frame AI as a human/machine partnership, where AI "amplifies, rather than replaces human intellectual ability". We used their perspective to build a smart mixed initiative edge detection tool and believe this approach will be particularly useful in building intelligent graphical tools.

Introduction

Ken Ford and Patrick Hayes have described a new way to think about intelligent machines called *cognitive prostheses* [1,2,3]**.** Cognitive prostheses are a constructive solution to the problems they described earlier with Turing-Test AI [3]. Paraphrasing, the premise of cognitive prostheses is that the human and the computer should be linked together as equal partners in a problem solving system, with the computer component *amplifying*, but not replacing, human intellectual abilities, as eyeglasses improve a human's vision system.

This idea is has similarities to, but is different from adaptive software, also popular in AI subareas as AI and Education and Intelligent Tutoring Systems (e.g., intelligent Help tools). The two subareas are similar in that both use reactive AI, where the software attempts to respond to the user in real time. Adapative software forms a user model and adjusts its performance in order to provide assistance as unintrusively as possible. The difference between these two approaches is that adaptive software takes the initiative to give help to the user, whereas in the cognitive prosthetic, the user takes the initiative to give help to the software.

Adaptive software begins with a reasonable initial user model, which is subsequently modified by monitoring the user's performance mostly through the narrow bandwidth of keystrokes and mouse movements. From this limited information, the software must construct a diagnostic theory to explain any buggy behaviour and suggest a remedy. Our cognitive prostheses in the CG modeling domain reverse the roles of human and software. The human monitors the software's performance, through the high bandwidth of a graphical display, determines when the software's performance is buggy, and quickly provides a remedy through the low bandwidth of mouse gestures and keystrokes.

McCalla [9] describes the user modeling problem in ITS as AI-complete—that is, a solution to user modeling will be a key to the solution to all of AI. A successful and complete intelligent tutoring system will be a superior intelligence in some sense: the software must have mastery of deep knowledge of the subject domain, it must be able to use this deep knowledge to infer the user's performance errors from mere keystrokes, and it must then formulate a plan to guide the user not only to the right answer, but, in doing so, help the user gain a deep understanding of the answer. This approach seems to simplify the AI challenge.

The cognitive prosthetic framework changes the goals of AI by viewing the human and the software as equal, but different, partners in a problem solving partnership. The new artificial intelligence is the synergistic *mind* that from the software/brain combination. (We advise the reader that we overload the term *artificial intelligence* in this article. Ford and Hayes see cognitive prostheses as a new way to think of AI in general, but it is also possible to see them as a different way to do AI.)

Ford and Hayes [3] speculate that that the theory of cognitive prostheses is (literally) a more productive way to look at Artificial Intelligence than Turing Test AI is, which contemplates machines that independently perform intellectual activities as well as humans. Not only is the precise meaning of the test unclear, Ford and Hayes argue convincingly that Turing Test AI makes incremental progress difficult to measure. Not only is the test sensitive to who might be chosen as the judge, a machine that behaves like a human 30% of the time would be remarkable but would never fool any credible judge. Ford and Hayes conclude "… if we abandon the Turing Test vision, the goal naturally shifts from making artificial superhumans which can replace us, to making superhumanly intelligent artifacts which we can use to amplify and support our own cognitive abilities, just as people use hydraulic power to amplify their muscular abilities. This is in fact occurring, of course … our point here is only to emphasize how different this goal is from the one that Turing left us with." [3]

Although the preceding quote dates back to 1995, and Ford and Hayes have spoken about cognitive prostheses subsequently, there seem to be few details published on the theory and practice of cognitive prostheses. (An exception is the weather forecasting system STORM-LK. [4]) Nor has the idea of cognitive prosthetic hit the AI mainstream. For instance, Mackworth [8] does not include anything similar in a recent list of nine definitions of AI. This, we think, is due to the fact that, like computer graphics, the proof of a cognitive prosthetic is in the implementation, and therefore requires both a commitment prior to the development of the software and a good deal of development time.

We have had some success using this approach to build smart graphics tools in a modeling domain with immediate application. The results have been reported to the Computer Graphics/HCI community [12], but with only passing mention made of the new cognitive prosthetic paradigm. Here we wish to concentrate on the contributions to both to the theory and practice of cognitive prostheses, and the potential for success using this approach more widely, particularly in the construction of smart graphics tools by exploiting the incredible computing power of the human eye/brain combination.

Our experience suggests that successful cognitive prostheses should have the following features:

1 A cognitive prosthetic should be as easy to use as a traditional tool, but should improve the user's overall productivity. This does not completely solve the problem of how to *measure* progress, but users might be able to report whether the new tool seemed more productive.
2 The software component of a cognitive prosthetic should gracefully degrade as needed, probably all the way to some baseline algorithm. By baseline algorithm, we mean some algorithm that, from the perspective of a typical user, is well understood. Baselines may change with time.
3 In production environments, users should not feel that they are must occasionally work against an intelligent tool. A cognitive prosthetic should not be intrusive, nor should the user have to second guess the algorithm.
4 As software components improve, cognitive prostheses evolve, making older prostheses extinct.

The above features excludes no existing AI tradition so that foundational theories of AI, theorem provers, vision systems, even Turing Test AIs can develop independently with this stream. In this respect, we differ from Ford and Hayes by a quibble—we see the whole as greater than the sum of the parts. To borrow from the experience of logic programming: logic programming hoped that software could be divided into the narrow domain-specific aspect of casting problem knowledge into domain axioms (knowledge representations), and a general aspect of building an all-purpose control structure (automated theorem proving). Kowalski [5] summed this up with the equation "Algorithms = Logic + Control" and there was a great deal of sentiment that the two terms of Kowalski's equation could develop independently. In the context of Kowalski's equation, we see the task of cognitive prostheses as being to that of contemplating how best to implement the '+' operator.

Before moving on to specific CG based cognitive prostheses, an analogy may clarify the relationship between software and human in a cognitive prosthetic.

An analogy from the evolution of typesetting

The evolution of typesetting provides examples of frustrating smart tools that violate the above principles of good cognitive prostheses. The analogy should be meaningful to computer scientists, of whom many have done their own typesetting for several decades. Smart typesetters generate a range of complaints; two common (and related) ones are about hyphenation and layout.

Good hyphenation requires both literacy and aesthetic ability. In Gutenburg's era, typesetting began a craft overseen by an individual compositor who assembled type from a fixed stock of moveable letters into a frame. The invention of the linotype in the late 19th century fuelled the spread of newspapers and also created an interesting division of labour. The linotype was a big, complicated, hot, and noisy piece of machinery. A linotype operator had to be strong enough to pitch bars of lead and mount fonts (large frames of brass matrices), and literate enough to read and hyphenate continuous news text. The linotype operator would create the type, ink it, and print a "proof" by pressing blank paper against the type (hence the name

"letterpress"), which would be checked for accuracy by an editor or proofreader, who would mark corrections on the proof and send them back to the typesetter. The noise, inky grime, and heat of the printshop created a division of labour between those who composed the news and those who cast it into lead. However, looking at this division from the perspective of cognitive prosthetics, it can be said that both tasks required a good deal of intelligence and education for that day, with the editor ultimately controlling the appearance of the printed page. With both agents being human, communication in either direction was efficient.

Offset printing technology replaced letterpress in the mid-20th century and phototypesetting put the linotypesetter out of work. Ultimately, the computer became the machine of choice for typesetting.

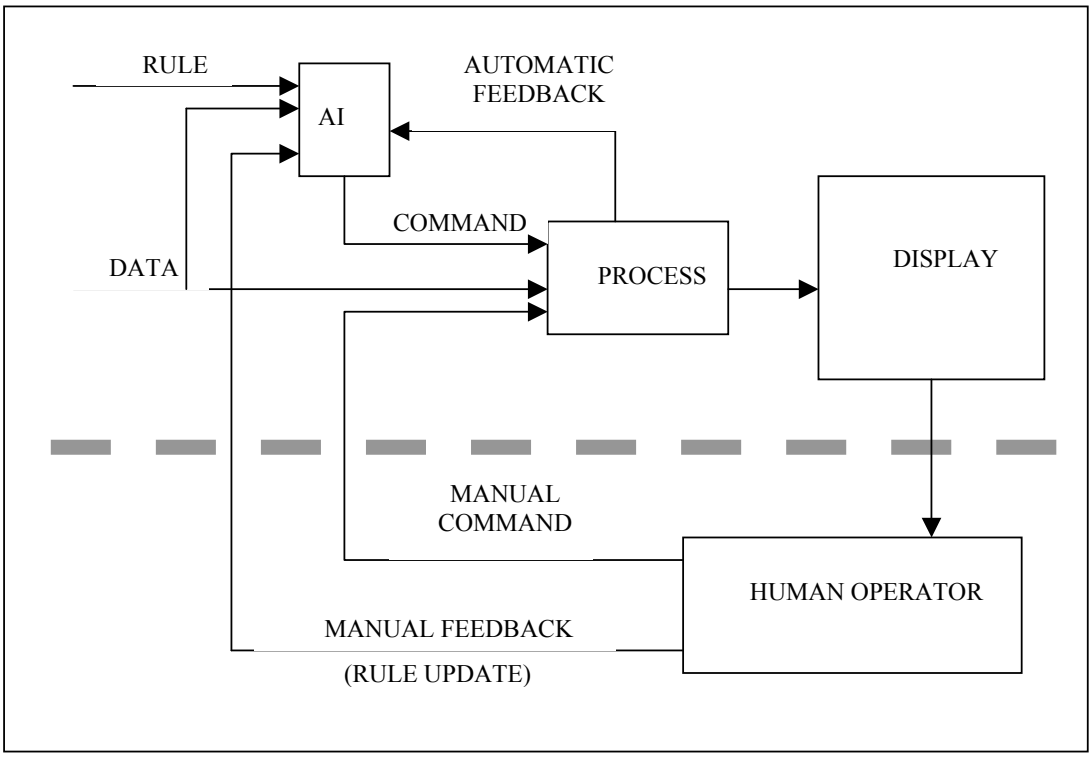
Computer typesetting passed through at least two generations of hyphenation algorithms. The first generation, rule-based algorithms, created a division of labour similar to the linotype era: the human, like the editor of the previous era, *composed* text; the machine *decided* layout, using a small rule base and decision structure. However, compared to communication between editor and linotypesetter, communication between human and algorithm was almost nonexistent. The results were frustrating, with humans even rewriting language to eliminate awkward typesetting artifacts. (Hyphenation was not the only problem. One of the great stories of computer science is that Donald abandoned his series on fundamental algorithms for ten years to build TeX after being appalled at the quality of the computer typeset proofs.) An early improvement was the phantom "discretionary hyphen", used by typists to indicate preferred hyphenations to the computer. However, in this setting, humans received low bandwidth communication from the computer, and frustration with typesetting was comparable to that experienced by users of modern word processors trying to produce a document with many diagrams.

Cheap memory made dictionary-based hyphenation algorithms possible where the user can even expand the computer's dictionary. Hyphenation is no longer a significant concern, at least as compared to the problem of layout of text that includes figures,

The preceding is idealized, but shows how responsibility for an intelligent task may migrate among intelligences as technology evolves. Within computing, and especially graphics, there are a wide range of layout problems broadly comparable to typesetting—layout of web pages, layout of 2D and 3D graphics scenes, planning of animations, tessellation of contours—that could divide difficult decisions between humans and software. Moreover, the assignment of a task to a human or machine intelligence might change as different components evolve unevenly.

To avoid repeating some of the mistakes described above, it is eye-opening to think of a cognitive prosthetic in terms of a control system, but with a human acting as a sensor in the feedback loop, as well as user. Many AI technologies were premised on the hope that it might be possible to fully axiomatize interesting domains and subsequently find goals by reasoning, as in logic programming, and blocks-world planners. This resulted in the "knowledge bottleneck".

Reactive planners addressed the problem of complete knowledge by letting the algorithm use sensors to learn about a changing landscape and adjust plans accordingly. This helped with the knowledge bottleneck, but introduced the problem of reaction time [7]. The following shows a block diagram for a cognitive prosthetic.



The bottom half is the user; the top the AI and software. Together, they form a cognitive prosthetic. In the cognitive prosthetic we have produced, the human *is* the sensor. Moreover, in smart graphics, it is the best possible choice to make the human eye/brain combination the sensor. This makes it possible for the human to provide the necessary response time for a useful product.

Our domain: 3D Models from Contour Data

Tessellating contour data traced from serially sectioned data images is a common way to build accurate anatomical models [13]. Typically, an anatomist traces outlines (contours) of an anatomical object from cross-sections, e.g., the Visible Human data set, which contains sections of male and female cadavers at 1mm and 0.33mm slices respectively.

Contour tracing can be tedious, time-consuming, and expensive. It must be done by persons with enough advanced medical or anatomical knowledge to identify subtle features of the target object in a range of image types. As well, because of the needed accuracy, the user must carefully input a large number of points.

This is an ideal problem for a machine, in particular, an AI solution, and a variety of segmentation/edge detection techniques have been deployed. However, in the absence of perfect solutions, smart solutions have a problem: the user must, *a posteriori*, both find and correct the smart algorithm's errors. It may well be that the total cost of using a smart algorithm plus the cost of finding and correcting its errors, is greater than the cost of manually tracing the contours. Users anecdotally report similar frustrations with other smart tools.

A literature search revealed a tool called *Intelligent Scissors (IS)*[11] close to what we had in mind, and similar to *snakes* [16] for contour detection. With IS, a user initially selects a seed pixel in an image with the mouse, normally a point on an object boundary. As the user drags the mouse in the general direction of the boundary, the curve from seed point to mouse cursor "snaps" to the boundary. This satisfies part the first of our four criterion, that the smart tool is as easy to use as a traditional tool, by making the performance imitate, as closely as possible, the snap line tool of familiar painting programs. Later we address productivity.

To find a boundary to snap to, IS uses a variety of traditional vision techniques as part of a multiattribute cost function. For instance, Laplacian zero-crossing and gradient magnitude favour pixels of similar intensity, while gradient direction chooses against sharp turns. Sooner or later, the algorithm runs afoul of the user's *intended contour*. At this point, the tool must be recalibrated to account for the new local features. In IS, this is costly because it requires recomputing parts of the cost function over every pixel in the image. In a series of papers, Barrett and Mortenson explore other cost function attributes, methods of minimizing the recalibration cost, as well as machine learning techniques that make intelligent guesses about when and where to recalibrate the algorithm based on an idea of "cooling" pixels on the path that have *persisted* for the longest time. A pixel's persistence is a combination of how many times it has appeared on some path from the seed point to the mouse, and how many milliseconds it has been on one of these paths. The intuition is that the user, while drawing the scene will move the mouse in hopes of selecting the 'best' path. Barrett and Mortenson's results [11] are very impressive. The depth of their work satisfies us that this particular problem remains clearly in AI's domain.

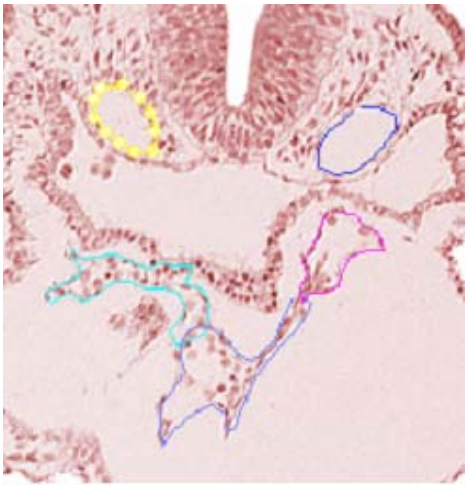


Figure 1. Contour data from the virtual human embryo project.

In our own implementation, we chose a different approach to the correction mechanism, based on the assumption that, for the immediate future, there remain many occasions when even a very sophisticated edge detection algorithm will eventually make mistakes. Figure 1 gives an example of real contour data from the Virtual Human Embryo project [14], for which it would be difficult to build

a highly reliable smart outlining tool. Although the exterior outlines are clear, interior contours might best be done with a manual tool. Somewhere in between the extremes of reasonably good data and very noisy data, it seems reasonable to expect that Even another, very fast *person* trying to trace *your* intended contour might eventually make a choice different from yours.

With a cognitive prosthetic, the problem is not making a bad decision, which is unavoidable, but recovering from it quickly. What makes the problem challenging is finding a way to let the edge detection algorithm to do its best to construct the intended contour using using pixel-based information, but still letting the user, upon observing the image and the traced contour, undo any error the AI makes and either recalibrate the edge detection tool, or override it.

We tried several approaches including using the user's mouse motions as recalibration cues, but this method seemed to be overly sensitive to vagaries of the user's hand movements. Adding a language of mouse motions did not seem to be a promising approach. We also incorporated a visual feedback tool that monitored feature calculations into the user interface, hoping that 1) the feedback graph might reveal some analytic property of error features that could be exploited, and 2) perhaps aspects of the monitored information would be translatable to something useful to the user.

The breakthrough was realizing that the dynamically drawn contour was itself, to a typical user, the most informative monitor. Our final solution was a visual cue called the *leash*. Figure 2 shows two details that illustrate the leash. The contour is rendered in three differently colored segments indicating 1) pixels that the algorithm believes are *certainly* on the contour, 2) pixels that are *probably* on the contour, and 3) pixels *possibly* (or not) on the contour. Following Barrett and Mortenson, the first segment is colored blue indicating cooling. The last two segments give the viewer the impression of a "leash" "leading" the new contour growth. The idea of the leash is inspired partly by Kyburg [6] who classifies knowledge as logical certainty, practical certainty, and probability. The user can use a mouse gesture to tell the program when to promote probable pixels to certain. If this is not possible, the user can drop into fully manual mode. This satisfies the second of the four criteria—the cognitive prosthetic gracefully degrades to a baseline algorithm.

The visual cues greatly reduce the cost of correcting the errors in the contours, since the user sees them as they arise. This satisfies our third criterion. Users do not have to, *a posteriori*, go back and undo the errors made by the smart algorithm, which might, in some cases, take longer than a fully manual approach. This removes the problem of reaction time [6] from the algorithm, and allocates responsibility to the human, who is well-suited to seeing when the new contour is going astray, thanks to he visual cues given by the leash, that advise both where and when to recalibrate. Thus, there is no marginal cost for *fixing* the error, since the worst case scenario is the same as tracing manually. This suggests that the productivity requirement of the first criterion is also met.

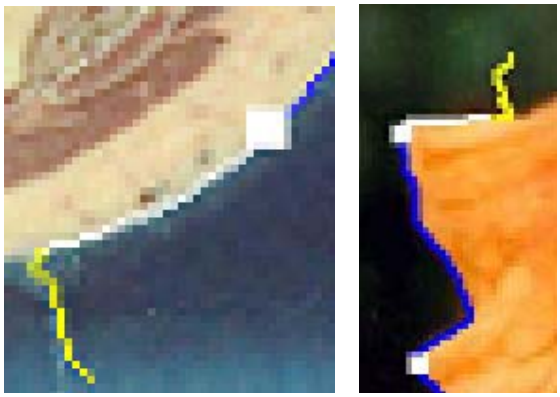


Figure 2. Two details illustrating the leash. Pixels known to be on the edge are colored blue. White indicates pixels probably on the edge and yellow indicates pixels the software is unsure about. Notice some yellow pixels are on the edge.

The cognitive prosthetic is not simply a complex piece of software. It may include its own cognitive prostheses. For instance, the leash performs a calculation for optimal leash length [12], which might also be overruled or adjusted by the user.

Although we diverged from Barrett and Mortenson by assuming that for the short term, practical smart outlining tools will need to be overruled by a human user, this architecture is designed so that the AI may become arbitrarily smart—the user will simply have to override it less often. This satisfies our fourth and last criterion.

Thus, the leash appears to satisfy all four criteria. It results in greater productivity because the user can easily correct errors as they arise, rather than having to hunt and find errors created by segmentation based algorithms. The leash gracefully degrades to the familiar snap tool in areas where image noise confuses the edge detection logic. The third criterion reduces the user's sense of fighting the tool and lastly, the design makes evolution of the whole tool very straightforward—if edge detection tools get faster and better, the functionality of the leash may become vestigial.

Future Work

We have begun investigation of the problem of contour alignment. Contours can be misaligned for a variety of reasons. A possible cause is misalignment of the image slices. Figure 3(a) illustrates misalignment in the Visible Male data. Contour misalignment results in the appearance of ridges or strata on the surface of the 3D object. Like contour drawing, alignment is a repetitive job that we would like to give to a machine, but it requires a good deal of intelligence. The transverse image in Figure 3(b) was created by selecting the same row from consecutive slices of the Visible Male. Figure 3(b) shows the result of applying an alignment correction algorithm. (The software introduced other unrelated artifacts.)

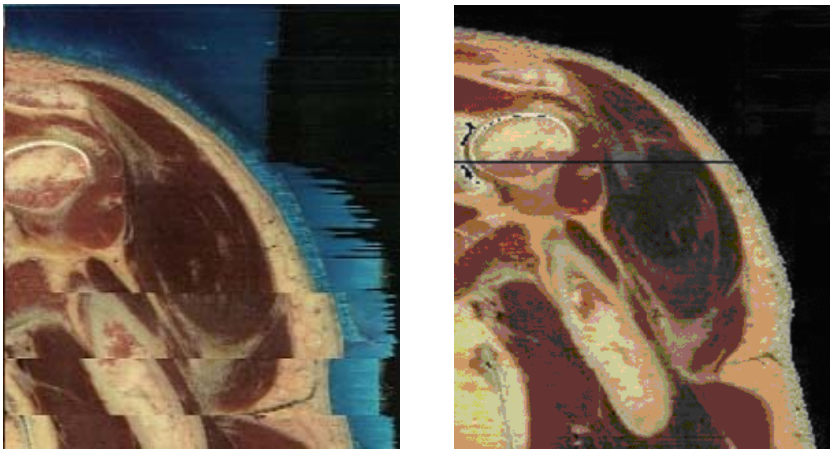


Figure 3. (a) (Left) Unaligned visible male data. (b) Same data after alignment.

The alignment algorithm minimizes the sum of differences of Euclidean distances of RGB values in consecutive slices by changing the image alignment. The algorithm worked well on the Visible Male, but there are problems. First and most obviously, the algorithm will eliminate skewing, even where the target object is supposed to be skewed. Secondly, this technique did not help with the virtual embryo data. Because of the domain knowledge needed, purely vision-based algorithms will be prone to error. This presents new challenges because it is not clear at this point that the lessons of the leash will generalize to three dimensions.

Summary and Conclusions

Our current work in smart graphics tools builds upon and extends the theory of cognitive prostheses as first articulated by Ford and Hayes as a human/machine synergy, and we have suggested four criteria for a successful cognitive prosthetic, at least in the area of smart graphics. The idea of the leash is a particularly crisp realization of this framework because the AI edge detection subsystem, the contour drawing tool and the leash interface are clearly separated. We also suggest that the area of smart graphics is particularly suited to this paradigm because of the high bandwidth with which a graphics program can communicate visibly with a human. Finally, we have also suggested that the human and the software together form a mind in the philosophical sense, creating a new kind of artificial mind.

Acknowledgements

Thanks for John Cork at LSU for the Virtual Embryo Data, and to 3D model builder Scott Lozanoff at the University of Hawaii at Manoa for many conversations about the model building enterprise. Research for this work has been supported variously by the University of Saskatchewan Institute for Computer and Information Technology, the Natural Science and Engineering Research Council of Canada.

Referee: the block diagram will be reduced, also please check the PDF for the images; not all details show up well on a B&W printer. We will enhance the final submitted images.

References

1. Ford, Kenneth M. and Patrick J. Hayes. Cognitive Prostheses. *Invited talk FLAIRS-99*,Orlando
2. Ford, private communication.
3. Hayes, Patrick J. and Kenneth M. Ford. Turing Test Considered Harmful. In *Proceedings of IJCAI-95* (1995) 972-977
4. Hoffman, R.R., J.W. Coffey, K.M. Ford, M.J. Carnot. STORM-LK A Human-Centered Knowledge Model for Weather Forecasting. In *Proceedings of the 45th Annual Meeting of the Human Factors Society*, Minneapolis (Oct 2001)
5. Kowalski, Robert. Algorithms = Logic + Control. *Communications of the ACM.* **22:7** (1979) 424-436
6. Kyburg, H.E., Jr. Science and Reason, Oxford University Press, Oxford, 1990
7. Lesperance, Y., K. Tam, K., and M. Jenkin. Reactivity in a logic-based robot programming framework. In *Proceedings of Cognitive Robotics—Papers from the 1998 AAAI Fall Symposium*, Orlando, Florida (1998) 98-105
8. Mackworth, Alan. Recycling the Cycle of Perception: A Polemic. Invited Talk, abstract *in Proceedings of AI-2002, the Canadian Conference on Artificial Intelligence*, Springer-Verlag LNAI 2338 (2002) 102
9. G.I. McCalla, "The Central Importance of Student Modelling to Intelligent Tutoring", in E. Costa (ed.), New Directions for Intelligent Tutoring Systems, Springer-Verlag, Berlin, 1992, pp. 107-131.
10. McGrenere, J., Baecker, R., and Booth, K. (2002). An evaluation of a multiple interface design solution for bloated software. ACM CHI (2002) 164-170.
11. Mortenson, E. and Barrett, W.A. Interactive Segmentation with Intelligent Scissors. Graphical Models and Image Processing **60** (1998) 349-384
12. Neufeld, E., Popoola H., Callele, D. and D. Mould. Mixed Initiative Interactive Edge Detection. *Proceedings of Graphics Interface 2003,* (to appear)
13. TOUCH telemedicine http://hsc.unm.edu/touch/.
14. Virtual Human Embryo
15. http://virtualhumanembryo.lsuhsc.edu/demos/Stage13/Stage13.htm
16. Williams, D.J. and M. Shah. A Fast Algorithm for Active Contours and Curvature Estimation. CVGIP: Image Understanding 55:1 (January 1992) 14-26