

## Leveraging 1,000 and 10,000-Fold Increases: Considering the Implications of Moore's law on Future UI Tools Research

Scott E. Hudson, HCI Institute, Carnegie Mellon University

The 1984 Macintosh was introduced at the around the time of much of the “classic” UI tools work (e.g., the phrase “User Interface Management System” was coined at the Seattle workshop in 1982, and the Seeheim workshop was in 1985). This machine executed about 1 million instructions per second, had 128K of RAM, and a 400K floppy disk. In contrast, a personal computer today could easily be 5,000 times faster and be equipped with 16,000 times as much RAM, and 500,000 times as much persistent storage.

This is of course old news – we are all familiar with the effects of the technological advances described by Moore's law and its corollaries. Or at least we *think* we are – in fact, it's very difficult for us to understand what exponential change really means. Consider for example that Moore's law predicts that every single bit of the CPU speed increase between 1984 and today – *all of it* – will occur again by the time of UIST '06. If we (optimistically) think our research will have an impact in 5 or 6 years, Moore's law predicts that we will see 15 times that much change by then.

A large part of the goal of this workshop is to try to define new directions for working ahead in a new world, just as the Seattle and Seeheim workshops which served as the precursors to the UIST conference did. However, because the changes in front of us are so much larger than the changes of the past (e.g., since 1984) their effects will be a much more dominant force than they have been. So while the effects predicted by Moore's law seem to be almost a cliché, I think we have not in fact paid enough attention to the coming changes, and that any future research agenda is well served to explicitly try to leverage these changes. This position paper looks at a number of implications of this view, and considers a number of research directions which are designed to directly make use of vastly more computing power than we have available today.

One general approach to leveraging enormously more computing power is to reconsider the notion of efficiency which serves as at least an implicit background for much of systems implementation. We should (*almost*) take on a mindset of finding ways of *wasting* cycles (in service of the user interface, of course). As a simple example, we might well want to consider constantly “computing ahead” – modeling the user's actions and precomputing the results along perhaps the five most likely next inputs (or more likely as many high probability next input alternatives as there are time for, prior to the next input). This can be used to provide new kinds of feedback as well as shortcuts for the user, but can also enable proactive prefetching from slower media (e.g., disk and especially network resources). This benefit is particularly noteworthy because one of the less noted consequences of exponential change is that differences in scaling rates become very important. For example, the ratio of CPU speed to disk latency is going to skyrocket. We need to find clever ways for using our newly expanded resources to compensate for the resources improving much more slowly.

Related to the concept of “computing ahead” (and probably more interesting) is the notion that we should move from the idea of a single state of a system or object of interest, to maintaining multiple alternative states simultaneously. To do this well, we need to support new object models which handle this gracefully. This will be useful both in interesting new interaction techniques which allow “what if” explorations to happen naturally [Terr04] and as basic support for dealing with ambiguity and asynchrony (see below).

Another use of substantially more computing power is in the application of sensing and recognition techniques. Here we have the possibility of opening up substantially new and more natural interaction modalities. While these techniques are already of great interest in the community, to take best advantage of the changes to come we need to consider techniques which currently seem impractical. For example, rather than thinking in terms of a single computer vision input, it would be advantageous to work on the basis of 20 or even 50 separate vision input processing streams devoted to a single user, with each camera feeding several special purpose recognition components. As a simple example, we might use 20 cameras with narrow but overlapping field of view to cover a single user. The raw data from such a camera array might be processed in a number of ways simultaneously. First, a set of recognizers might be used to track gross position of people in a space. Based on this, specialized recognizers would be targeted to recognition of critical elements such as faces [Viol04], eyes [Shel03], arms, hands, and work objects of particular interest [Kim04].

Several challenges arise from this approach. First, as soon as recognition is applied – even recognition with substantially better accuracy than we have now – the inevitable result is the introduction of ambiguity to inputs. To cope with this we will need underlying mechanisms to model ambiguity and make it much easier to deal with [Mank00], including easy maintenance of multiple viable alternatives, and sophisticated models of current and prior probability (probably based on machine learning approaches which can adapt to users and tasks).

Another important implication of the multiple-sensor approach indicated above is the introduction of asynchrony and delay into input processing (along side ambiguity). While delay might not seem to be a likely consequence of vastly faster machines, it is fact intrinsic in the environments we should expect in the future because of the combination of concurrency and differential scaling. Part of future speed increases will inevitably come from concurrency, and such concurrency is a good match for systems with many sensors. However, the combination of concurrency and differential scaling of capacity increases will inevitably lead to asynchrony – to some inputs being delayed with respect to others. This will again require us to reconsider our basic infrastructures for representing the state of objects so that we can partially proceed based on the inputs we have, but easily back up and reconsider when new evidence arrives.

Finally, in thinking with respect to Moore’s law I propose it will be fruitful to look for techniques which in the past have been applied to large problems in an “off-line” or “batch” setting, which can be adapted for our use. For example, the use of numerical optimization techniques to undertake problems previously approached algorithmically [Foga03]. Similarly, we might consider use of large cognitive models to provide on-the-fly tacking of user’s behavior with respect to task models, as well as large scale statistical data gathering (across many users at internet scales [vonA04]) which can be used for clustering and subsequent individual user action matching for predictive purposes.

Across all of these different approaches to leveraging coming large scale change comes a single meta-challenge: how do we explore the use of vast amounts of computing power now, before it is cheaply and simply available. A partial answer to this, at least when slightly higher latencies can be tolerated, might be the application of clusters of current machines connected by fast local networks. These are now routinely applied to large numerical problems. The UI tools community needs to consider creating the infrastructure needed to easily employ these systems for UI problems that we currently think of implementing with a single PC. Another approach may be the development of flexible, small but specialized components. For example, we might consider a development of a “vision appliance” which combines an inexpensive camera, and relatively low cost digital signal processing chip in a simple package which could eventually be mass-produced cheaply. For any of these solutions, however, it will be critical to develop not only the basic computing infrastructure, but also a level of software infrastructure above it which makes those capabilities reusable and accessible to a wider set of researchers who do not wish to expend time or effort on the underlying computing platform. Fortunately, creating reusable high-level programming abstractions is one of the activities that the UI tool community has had particular success with.

## References

- [Foga03] J. Fogarty and S. Hudson, “GADGET: a toolkit for optimization-based approaches to interface and display generation”, *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pp.125-134, 2003.
- [Kim04] J. Kim, S. Seitz, and M. Agrawala, “Video-based document tracking: unifying your physical and electronic desktops”, *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp.99-107, 2004.
- [Mank00] J. Mankoff, S. Hudson, and G. Abowd, “Providing integrated toolkit-level support for ambiguity in recognition-based interfaces”, *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp.368-375, 2000.
- [Shel03] J. Shell, R. Vertegaal, and A. Skaburskis, “EyePliances: attention-seeking devices that respond to visual attention”, *CHI '03 extended abstracts on Human factors in computing systems*, pp. 770-771, 2003.
- [Terr04] M. Terry, E. Mynatt, K. Nakakoji, and Y. Yamamoto, “Variation in element and action: supporting simultaneous development of alternative solutions”, *Proceedings of the 2004 conference on Human factors in computing systems*, pp.711-718, 2004.
- [Viol04] P. Viola, and M. Jones “Robust Real-Time Face Detection”, *Int. J. Comput. Vision*, Kluwer Academic Publishers, 57(2), pp.137-154, 2004.
- [vonA04] L. von Ahn and L. Dabbish, “Labeling images with a computer game”, *Proceedings of the 2004 conference on Human factors in computing systems*, pp.319-326, 2004.