Task Decomposition and Human Computation in Graphics and Vision

Dan B Goldman
Adobe
dgoldman@adobe.com

Joel Brandt
Adobe
joel.brandt@adobe.com

ABSTRACT
We propose a few research challenges for crowdsourcing in the computer vision and computer graphics domains, and pose broader questions about the scope and capacity of crowdsourcing platforms to solve ill-defined large scale problems. Specifically we consider how crowds can be employed for task decomposition. We envision human computations that are both self-assembling and self-debugging.

Author Keywords
task decomposition, human computation, crowdsourcing, computer graphics, computer vision

ACM Classification Keywords
C.2.4 Computer systems organization: Computer-communication networks—*Distributed systems*

General Terms
Algorithms, Design, Economics, Experimentation, Human Factors, Management

INTRODUCTION
Because the human visual system is a central actor in both computer graphics and computer vision, it is likely that crowdsourcing can play several important roles in these domains. There are at least three such roles: First, crowds will be used to validate synthesis models and approaches by large-scale subjective evaluation. Second, crowds will provide massive amounts of training data for machine learning algorithms. And third, where current computer vision and graphics algorithms fall short on their own, crowds will become a part of the algorithm, supplanting the huge data capacity and brute speed of current computers with higher cognition of (many) human brains.

The first two of these roles are already coming to fruition, by lowering barriers to larger scale studies and data acquisition. However, the third role – getting humans in the loop effectively – is almost completely unexplored in the research community. This role presents several fundamental challenges:

1. Developing a means to reliably distinguish between tasks that are more easily approached by developing new algorithms or by turning to the crowd.

2. Figuring out how to easily extract those answers from humans. How do we break down the tasks? What are the right rewards? What are the right distribution systems?

3. Figuring out how to integrate these tasks into larger systems. How do we make it fast enough that people could use it in the real world? How do we control quality?

In this paper we propose possible directions for addressing these three challenges. Using these directions as motivations, we then look more broadly at the possibility of using crowdsourcing to solve large, ill-defined problems.

NEAR-TERM RESEARCH QUESTIONS
In crowdsourcing systems, can we rely on humans to faithfully report the behavior of their own visual system? In previous work, we used Mechanical Turk to determine the visual importance of differing regions in a pair of similar images [5]. Using an interactive "marquee" selection user interface, we asked participants to select or highlight regions that appeared significantly different between the images. But we did not address whether this self-reported "cosaliency" would match up with traditional measures of saliency, for example, via gaze tracking. Although results in related areas [4] give us some confidence in this methodology, our specific approach has not yet been cross-validated.

While our prior work looks only at static images, the potential impact of crowdsourced data is perhaps more interesting for *video*. At present there are no large-scale datasets of video annotated with measures of human visual saliency. But if we could rely on some self-reporting mechanism to find out what objects or regions attracted visual attention in a video sequence, crowds could quickly provide a massive database of video saliency for training machine learning algorithms.

Note that the naive solution – just ask participants what they were looking at in a video – is not particularly helpful. For example, in a three-second clip of a person walking, a respondent might look at the person's hands, face, feet, and the background. This is the data that computer vision re-

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

searchers would hope to capture. However, many respondents might reasonably report "I was looking at the person." The challenge here is to design a stimulus in which the self-reported result *must* be from the actual center of gaze at any given moment.

Another immediate problem comes from a technical challenge that we face at Adobe, best explained by example: A designer might use InDesign to produce a print magazine layout for a specific page size. As mobile devices become the preferred medium for magazine consumption, the same magazine must now be redesigned for multiple mobile screen resolutions, sizes, and aspect ratios. At present, our design programs are largely built to accommodate fixed layouts, and we have limited technology that can assist in solving this problem for our customers: The cost of layout design is essentially multiplied by the number of form factors.

While producing the first layout of a magazine page requires great skill, does the task of adapting an *existing* layout to a new form factor really require the same skills and degree of effort? Certainly there is some taste involved in choosing among alternatives, but the individual tasks involved are in large part mechanical: Change the number of columns, scale an image, remove a graphic, move it to a different page, etc. Is it possible to break up an inherently visual task into small pieces much like, *e.g.* Soylent [2] breaks up proofreading tasks? Do design patterns like find/fix/verify extend to such domains? How would they need to evolve to solve visual problems?

Again, the naive solutions don't apply. One drawback of Soylent is that individual tasks are removed from their context. This causes problems at a holistic level. For example, parallel structure is not always preserved when splitting up a proofreading task into many small-grained tasks. Such problems might be even more acute for visual page layout, which is even more holistic in nature, such that changing an element on one side of the page might require changes throughout the rest of it (or throughout a multi-page document). On the other hand, it is easier to implement holistic "verification" tasks for balance and consistency because they require less time-consuming careful reading than text-based tasks.

We make no claims that these are easy problems to solve, but they may be tractable starting places for near future research.

LONGER-TERM RESEARCH QUESTIONS

The first author of this paper spent a previous career as a visual effects artist and supervisor. Among many other projects, he was involved in the design and development of asset management systems, which came to encompass workflow management as well. When, for example, a change was made to a 3D model, a notification was automatically routed to the artist who painted it, in order to let them know it might need to be retouched. Moreover, this workflow chain was configurable, so that if the sequence of steps to create an asset were changed from one production to the next, a supervisor could adjust the workflow easily.

As crowdsourcing has matured, we have begun to reflect on how this sort of reconfigurable process automation could be extended to virtual crowds. In particular, different individuals have different skill-sets and expertise. Some individuals may excel at organizing work into structured processes, while others may work best at executing predetermined processes. Thus far, research into crowdsourcing has focused primarily on the latter category, but relatively little effort has been expended on the former. In projects like Soylent [2], the researchers themselves created the sequence of steps for the crowds to execute. There are far fewer examples in which the crowd itself participates in task decomposition and process description. One notable exception is FoldIt [3], an online game for solving protein structures. FoldIt players can create "recipes" that can be used by other players to solve puzzles. (See `http://fold.it/portal/recipes?sort=desc&order=Rating` for example.)

Some additional research questions along these lines include: Can all large problems be broken down into small ones? This is a central tenet of software engineering, but does it apply to people? Can crowds self-organize by skill-set? Can crowds build hierarchical human programs, in which experts provide high-level task breakdown and amateurs apply more constrained labor?

In order to talk reasonably about these questions, one needs to be very precise about his definition of crowdsourcing. One really narrow definition is "getting something done by posting really small tasks to Mechanical Turk, exactly as the vanilla version works today". Under that definition, the answer to the above questions is almost certainly "no". Of course, a broader definition of crowdsourcing is "using money to pay people to do stuff you don't want to do". Using that definition, the answer to the above questions is "yes:" In this definition, everything in the world that gets done today gets done through crowdsourcing. Humans self-organize into groups that can get things done. Some people break down big problems in to smaller ones. People with different skill sets choose to do the things they're better at. Eventually, every large task gets broken down into a series of 5-minute-long actions that a human can complete. As Adar has observed [1], proving that "crowds can do what humans can do" is not a great recipe for groundbreaking research.

But things get much more interesting if one has a very precise definition of your worker pool. For any set of constraints, one can then try to achieve a higher upper bound of what pools with those constraints are capable of. In this way, crowdsourcing research is a lot like both management science and algorithms research. In the former, one might ask "Can managers get *remote* employees to do X, where X is something that co-located employes already do?" instead of "Can employees do X?" In the latter, one might ask "can I find the median of m numbers if I can only have $n < m$ in memory at any time and can only see each number once?" instead of "can I find the median of m numbers?" (Shahaf and Horvitz [6] have done some pioneering work along these

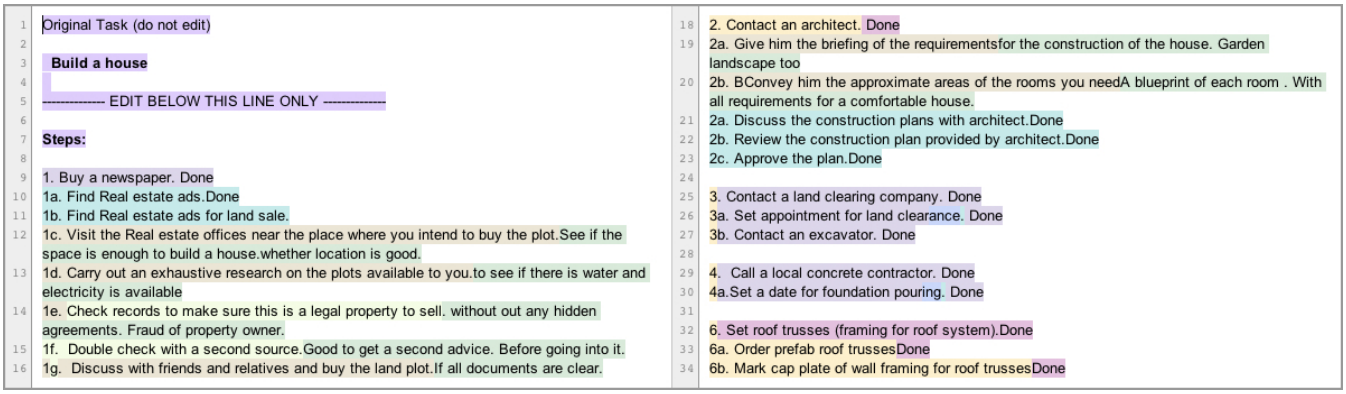

Figure 1. Steps resulting from 9 Mechanical Turk users who were asked to break down the task "Build a house" into 5-minute tasks.

lines, by optimizing mixtures of both human and machine computation based on their different characteristics.)

A PRELIMINARY EXPERIMENT

To dip our toe in the waters of self-assembling human computations, we conducted a quick (and very unscientific) study exploring the bounds of Mechanical Turk and its worker pool.

To assess how users of Mechanical Turk would break down tasks, we posted HITs to break down a complex task into small pieces. Specifically, we began with three single large tasks of varying specificity: "Write a story about a great man whose pride brings about his downfall," "Prepare a romantic dinner," and "Build a house." (See Figure 1.) Participants for all three tasks were asked to do the following:

> Read through the list of steps below, and do the following: If you see a step that you think would take less than five minutes, write "DONE" at the end of the step (if it does not already say "DONE").
>
> If you see a step that you think would take more than five minutes, delete that step and replace it with two (or more) smaller steps.

Participants were paid $0.10 if they rewrote one large step as two or more smaller steps, a bonus of $0.05 for every additional step broken down into smaller steps (up to $0.25), and $0.01 for every step correctly marked as "DONE" (up to $0.10). Participants worked simultaneously on the same document using PiratePad (`http://piratepad.net`). 38 turkers participated in the tasks: 13 for the fiction-writing task, 6 for the romantic dinner, and 9 for the building task.

Although some participants were more careless than others (*e.g.*, marking very complex subtasks as DONE, or skipping major steps like "choose what to eat for dinner"), the process turned out to be somewhat self-correcting. For example, later participants would occasionally "unmark" steps marked as DONE, or add intermediate steps that were previously skipped.

None of the turkers used conditional statements or loops, not even in the instructional style (*e.g.* "repeat steps 3 through 5 until X."). Yet some subtasks were surprisingly specific and technical, like this subtask from the house-building task: "Mark cap plate of wall framing for roof trusses." In this case, one knowledgeable participant seemed to "gum up" the process for later turkers, who could not assess the length or difficulty of such a task. It seems likely that this could be ameliorated with some type of feedback or "undo" to back up to a more comprehensible set of instructions.

Although our experiment was too simplistic to draw any actionable conclusions, it suggests that even on Mechanical Turk, where there is no threshold for participation at all, participants were capable and willing to break down large tasks into smaller ones. Given additional prompting, we expect they could also be encouraged to employ conditional and looping constructs necessary for flexible task breakdown.

CONCLUSION

To conclude, we envision a greatly expanded role for human computation in computer graphics and computer vision. More broadly, we foresee that the current paradigm of manually crafted human programming may quickly give way to self-assembling human computation. Someday, the creator of crowdsourcing tasks may be free to use broad guidelines and allow the crowd to assist in providing the specificity normally required for mechanical computation. To achieve this, such systems must be self-debugging, including corrective mechanisms to revise a program if the outcome does not meet objectives. Designing and building methodologies and platforms for these self-assembling and self-debugging human computations will be a significant challenge for the decades to come.

REFERENCES

1. Adar, E. Why I hate Mechanical Turk research, Nov. 2010. `blog.cond.org/?p=28`.

2. Bernstein, M., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: A word processor with a crowd inside. In *Proc. UIST* (2010).

3. Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J.,

Beenen, M., Leaver-Fay, A., Baker, D., Popovic, Z., and Foldit players. Predicting protein structures with a multiplayer online game. *Nature 466* (2010), 756–760.

4. Heer, J., and Bostock, M. Crowdsourcing graphical perception: using Mechanical Turk to assess visualization design. In *Proc. CHI* (2010), 203–212.

5. Jacobs, D. E., Goldman, D. B., and Shechtman, E. Cosaliency: Where people look when comparing images. In *Proc. UIST* (2010).

6. Shahaf, D., and Horvitz, E. Generalized task markets for human and machine computation. In *Proc. AAAI* (2010), 986–993.

BIOGRAPHIES

Dan Goldman is a senior research scientist at Adobe Systems in Seattle, working at the intersection of computer graphics, computer vision, and human-computer interaction. Dan's main area of interest is developing technologies to simplify the manipulation of digital media, especially digital video. He pursued his graduate studies at the University of Washington from 2002 to 2007, advised by David Salesin, Brian Curless and Steve Seitz. His thesis described novel visualization and interaction techniques for video manipulation. Additional research interests include computational photography, rendering algorithms, and geometry acquisition. He received his Bachelors and Masters from Stanford University, and spent 12 years as a computer graphics artist, engineer and supervisor at Industrial Light and Magic before pursuing his current research career. Dan is a member of the Visual Effects Society and ACM Siggraph.

Joel Brandt is a research scientist at Adobe Systems, focusing on human-computer interaction. His current research interests include software development, crowdsourcing, and digital books. Joel completed his Ph.D. at Stanford University in 2010, and was a member of Stanford's HCI Group, advised by Scott Klemmer. His thesis work explores the role that information resources play during software development. As part of this work, he built tools that make it easier for programmers to locate and use instructive example code. This software is used by thousands of programmers on a daily basis. He received a BS with majors in computer science and mathematics and an MS in computer science from Washington University in St. Louis.