Micro Perceptual Human Computation for Visual Tasks

YOTAM GINGOLD[1,2], ARIEL SHAMIR[2], and DANIEL COHEN-OR[1]
[1]Tel-Aviv University
[2]The Interdisciplinary Center

Human computation (HC) utilizes humans to solve problems or carry out tasks that are hard for pure computational algorithms. Many graphics and vision problems have such tasks. Previous HC approaches mainly focus on generating data in batch, to gather benchmarks or perform surveys demanding non-trivial interactions. We advocate a tighter integration of human computation into online, interactive algorithms. We aim to distill the differences between humans and computers and maximize the advantages of both in one algorithm. Our key idea is to decompose such a problem into a massive number of very simple, carefully designed, human *micro-tasks* that are based on *perception*, and whose answers can be combined algorithmically to solve the original problem. Our approach is inspired by previous work on micro-tasks and perception experiments. We present three specific examples for the design of Micro Perceptual Human Computation algorithms to extract depth layers and image normals from a single photograph, and to augment an image with high-level semantic information such as symmetry.

Categories and Subject Descriptors: I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

General Terms: Algorithms, Design, Human Factors

Additional Key Words and Phrases: human computation, crowdsourcing, perception, graphics, vision, images, depth, normals, symmetry

ACM Reference Format:

Gingold, Y., Shamir, A., and Cohen-Or, D. 2011. Micro Perceptual Human Computation for Visual Tasks ACM Trans. Graph. V, I, Article A (Month 2011), N pages.

This work was partially supported by Israeli Science Foundation grants 315/07 and 324/11.

Authors' addresses: Y. Gingold (corresponding author) email: yotam@yotamgingold.com; A. Shamir, Efi Arazi School of Computer Science, the Interdisciplinary Center, Herzliya, Israel; email: arik@idc.ac.il; D. Cohen-Or, School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel; email: dcor@tau.ac.il.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

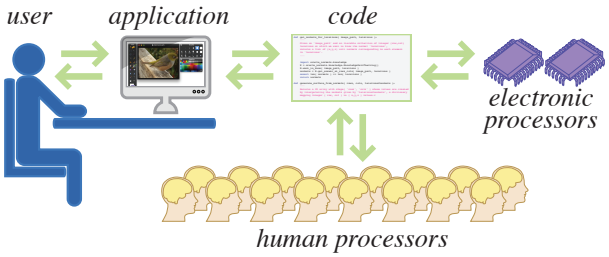


Fig. 1. We envision an interactive application where the user can choose an operation such as "create depth layers" that invokes an HC algorithm. The algorithm farms out many visual perceptual micro-tasks to a crowd of unskilled, isolated HPs, then combines their results and returns an answer at interactive rates.

DOI = AA.BBBB/C.D
http://doi.acm.org/AA.BBBB/C.D

1. INTRODUCTION

Human computation has emerged in recent years as a technique that involves humans performing tasks in a computational process. The name implies some role reversal: instead of humans using computers to solve problems, it is the other way around. Interestingly enough, not so long ago computers were in fact human [Grier 2005]. In our work we advocate a view that promotes a tighter synergy of humans and computers for computation. There are problems that computers solve much better than humans, but it is well known that some problems are very simple for humans and difficult for computers. Many graphics and vision problems exactly fit this category, as they rely on human perception. For example, it is simple for people to recognize, organize, segment, align, or correspond images and 3D objects. In contrast, the amount of research in the field towards solving these tasks indicates how difficult they are for pure machine computation.

Several classic human computation examples deal with graphics and vision problems, involving cognitive tasks that are difficult for automatic algorithms [Ahn et al. 2003; von Ahn and Dabbish 2004; Russel et al. 2008; Branson et al. 2010; Heer and Bostock 2010]. However, these works mostly use humans for off-line processing and gathering of data, and only loosely integrate human and machine computation. Most of these works create a game or a survey, which involve complex tasks, and focus on the incentives for people to participate. The results of the human computation are stored for future use in digital algorithms such as for machine learning.

We propose Micro Perceptual Human Computation (Micro-HC) as a paradigm that integrates the human computation into the online algorithm itself. Given a problem that is simple for humans and

difficult for computers, the question is not, "Can a human solve it?" because that is quite evident. Instead, what we ask is, "What is the key component of the problem that is easy for humans and hard for computers?" We aim to define an algorithm that uses "human processors" (HPs) just for this part in combination with digital processors. Distilling this difference between humans and computers leads to new designs for algorithms that maximize the advantages of both humans and computers. Our Micro Perceptual HC paradigm is inspired by previous work on perception experiments (especially [Cole et al. 2009], which ran their experiments online) and the concept of micro-tasks.

Humans are much worse at numerical computation than computers, but are better at performing cognitive tasks. Seeking to use only what humans are most effective at, we combine human computation tasks that are perceptual in nature yet very simple, with numerical computations in which computers excel. This is different than letting the human solve a task [Adar 2011]. First, humans solutions often require specific tools and expertise and use complex interactions. We seek simple, almost instantaneous, well-posed tasks that anyone can perform. Second, we rely on crowdsourcing, which calls for a highly parallel design. Such designs are usually *not* a straightforward partitioning of the task a single user would perform to solve the problem. We propose algorithms where the human computation is performed by a massive number of unskilled, oblivious, and isolated human "processors" (HPs) in parallel (Figure 1). This enables incorporating the HC into the algorithm itself, combining it with a machine computational part.

We envision a day when an artist working with an image processing program could choose an operation such as "Find Normals" on an object in a given image. The scene complexity, lighting, and reflections can prevent automatic algorithms from being reliable and successful (e.g. Figure 7). In such cases, the program could use a Micro-HC algorithm that dispatches the perceptual tasks to human processors over the Internet. The tasks incur some monetary cost and may take some time to complete. However, after a short while the answers of the returned queries are combined algorithmically to create the normal map, and the artist can continue editing, using this result (Figure 1).

The main advantage of using human computation is being able to solve problems that are difficult for pure machine computation. On the other hand, a Micro-HC algorithm has the potential to be both faster and cheaper than performing the same task by an expert user manually, for two reasons. First, HC exploits the massive parallelism of the crowd and, second, the HPs are mostly unskilled and therefore cost less. Although latency is still a major problem in many of our examples, there are ways to overcome this [Ipeirotis 2010; Chilton et al. 2010]. Already, receiving responses from anonymous HPs on the Internet can take on the order of seconds, and this may soon be instantaneous. Note that monetary cost is also becoming an issue for pure machine computation, especially in new models such as cloud computing [Hayes 2008; Armbrust et al. 2010].

Our contributions are presenting Micro Perceptual HC as a new algorithm design paradigm and providing three examples of its use to solve specific visual problems: defining the depth layers of an image, finding normal directions, and detecting symmetry in objects in an image. Micro Perceptual HC provides a way to apply tasks from perception experiments as micro-tasks in a tightly integrated human and machine computation algorithm. The main challenge in this design is to rephrase the given problem so that the solution could use micro-tasks that are small and simple (for humans), and whose answers can be computationally combined (by the computer) to solve the original larger problem. For graphics, these are usually phrased as visual perception queries. They are designed so that any unskilled human processor can perform them rapidly. They promote massive parallelism and enable simple verification for quality assurance.

2. RELATED WORK

Attempts to combine the answers of many users has been studied extensively. For example, in collaborative filtering, the preferences of many users are aggregated to generate reviews and recommendations [Goldberg et al. 1992; Adomavicius and Tuzhilin 2005]. However, only recently, in his seminal thesis, von Ahn [2005] proposed a definition of human computation. Still, in contrast to our approach, von Ahn and follow-up works have focused on offline computation and on finding ways to motivate HPs. Models suggesting payment for performing a task are already very successful [Amazon 2005], and we utilize them in our algorithms to dispatch HC micro-tasks.

The idea of iterative algorithms whose subroutines run as human intelligence tasks on Amazon's Mechanical Turk was explored in [Little et al. 2010; Bernstein et al. 2010] for text processing, perceptual sorting, and brainstorming. Making a similar argument as we do, [Bernstein et al. 2010] advocates integrating HC into a word processor. Shahaf et al. [2010] discuss task markets where human and machine intelligence are enlisted to solve problems and demonstrate language translation tasks. In another work using Amazon's Mechanical Turk, VizWiz provides answers to questions posed as a photograph and spoken question in one to three minutes [Bigham et al. 2010]. [Sorokin et al. 2010] introduced a workflow for using HC to create 3D reconstructions of objects for use with a grasping robot. In contrast to these works, our algorithms have a greater degree of parallelism, and our micro-tasks are designed to be answered by near-instant human visual perception.

Human computation can also be seen as a means to gather training data for machine learning algorithms. Many such projects have been created including the ESP Game [von Ahn and Dabbish 2004], LabelMe projects [Russel et al. 2008; Yuen et al. 2009], motion tracking [Spiro et al. 2010], and more. Such HC projects concentrate on gathering data; they involve complex tasks and, consequently, do not have the tight coupling of humans and electronic processors we suggest.

The use of HC for learning requires a model for the problem space. For example, [Talton et al. 2009] define a parametric space where the distribution of good models is estimated by tracking the modeling activity of a distributed community. Then, they enable novice users to navigate this space and create high-quality 3D models. [Kalogerakis et al. 2010] use a collection of labeled training meshes to learn an objective function of the mesh faces for mesh segmentation; the labeled training meshes are from [Chen et al. 2009], which introduced a benchmark for mesh segmentation (collecting data using Amazon's Mechanical Turk) but did not employ the results for segmentation per se. In the absence of a good parameter space for the problem and in the presence of missing information, our approach provides a solution by incorporating the HC into the algorithm itself instead of using it for gathering training sets or modeling the problem space.

Our algorithms are based on operations that can be viewed as visual-perception experimental tasks. Similarly, the tilting of gauge

figures [Koenderink et al. 1992] could be used as operations in a hypothetical Shape-from-Shading or Shape-from-Line Drawing algorithm [Cole et al. 2009], and the surface segmentation task of [Chen et al. 2009] could be recast as a surface segmentation operation for a human processor. However, in perception experiments, the primary goal is gathering data on humans [Heer and Bostock 2010] and less on the task itself; there is no special emphasis on low-cost, massively parallelizable micro-tasks that are completed quickly, nor is there a need for completely unsupervised or self-supervised experiments.

A major factor in terms of HC timing today is latency. However, it is not implausible that in the near future an HC algorithm will be able to post micro-tasks to a stream of jobs dispatched to HPs continuously. Studies on the properties of the Mechanical Turk show that completion activity by workers is relatively unaffected by the day of the week or time of day [Ipeirotis 2010; Huang et al. 2010]. We believe we are near the point in time when human computation can be interwoven directly into an interactive-rate application.

3. MICRO-TASKS

Computers are deterministic, fast, and efficient. Humans are slow and often inconsistent. When dealing with graphics problems the key characteristic that still provides a human advantage over machines is perception, and very often *visual* perception. For example, even an unskilled HP can recognize an object in an image much faster and more accurately than any machine algorithm today. To leverage the advantages of the two computation types we propose the design of Micro-HC algorithms based on visual perceptual queries. The design provides the following advantages.

Skill. We have little knowledge about the actual humans performing the task. They are assumed to be unskilled (not experts in any field), and, thus, we can only request that they perform very simple tasks. Simple tasks can also be performed by any HP without a lengthy training delay. Since HPs come online and go offline frequently, simple tasks ensure that the entire pool of workers is qualified and readily available.

Parallelism. Rather than relying on proficiency or skill, Micro-HC algorithms depend on parallel computation. By exploiting the massive pool of HPs and using simple queries for micro-tasks performed in parallel, the running time can be kept low. To obtain almost-complete parallelism, we seek to minimize data dependencies in our algorithms. The HPs are assumed to be isolated from one another and oblivious to the overall algorithm. Furthermore, compared to distributed machine computing systems, HPs execute very few operations per unit time and the latency between processors is high. With sufficient parallelism, the running time is dominated by the duration of a single operation.

Cost. Apart from the timing and skill benefits, using simple tasks also helps keep costs low by not paying for training, immediately or amortized. The goal is to design tasks such that using a massive number of unskilled workers in an HC algorithm would cost less than one (or a small number of) skilled workers solving the same problem.

4. QUALITY CONTROL

An important difference between machine and human computation is that sociological and economic considerations become a part of the algorithm design. Human computation is highly erroneous and noisy to begin with. This is due to a variety of factors. First, humans are not really deterministic and often inadvertently respond differently to the same question. This means there could be internal inconsistencies and inaccuracies within a single human processor. Second, even if each HP is internally consistent, there are varying biases or inconsistencies between HPs: different people may have different views and there could be more than one answer to a question. [Koenderink et al. 2001] and [Ipeirotis et al. 2010] illustrate internal inconsistencies and biases in humans.

For human processors an incentive must be attached to the task being performed. Different incentives were proposed in the HC literature, such as money [Amazon 2005; CrowdFlower 2007; txteagle 2008; Samasource 2008] and fun [von Ahn and Dabbish 2008]. However, the fact is that these do not guarantee truthfulness or seriousness. Putting it bluntly, HPs want to maximize their benefit while minimizing their effort (and they most probably will use their intelligence for this task instead of the one they need to perform). This directly leads to high failure rates and cheating. Even given reliable HPs, varying the design of operations can affect the quality of the output.

This means verification and quality assurance mechanisms must also be part of any HC algorithm, just as error correction must be part of network algorithms. In the following, we present some common methods previously used for validating the results of human computation, and in Section 7 we discuss some of our design alternatives. For a recent survey, see [Quinn and Bederson 2011].

Duplication. In this approach, the same operation is performed by N HPs. The simplest form of duplication is to use a function like mean, median, or mode on the N outputs to create the final result. This way HPs whose outputs are classified as outliers can be ignored and the remaining outputs can be smoothed. Taking the mode is akin to a simple plurality voting scheme. However, if the output values are not chosen from a set of distinct options, clustering may be required as a preprocessing step, which may require fine-tuning. A further complication is differing internal biases between HPs (see Section 5). This form of duplication can add latency to feedback given to HPs (i.e. payment) because N outputs must first be received.

An alternative approach to duplication requests the same operation to be performed more than once by the same HP ([Cole et al. 2009]). In graphics, one can apply simple transformations on the graphical object that are difficult for a human to reverse. For instance, the same query can be asked on a mirrored image. Such duplication detects internal inconsistencies, and, when these are too large, the HPs answers can be ignored.

Sentinel Operations. In this approach, HPs are given some "sentinel" micro-tasks to perform. Correct outputs for these tasks are already known (so-called gold data). This way, poor quality HPs are detected as those whose outputs from sentinel tasks deviate significantly from the known output, and their answers ignored. The number of sentinel micro-tasks can be increased as more computation is performed, by adding the results of trusted non-sentinel tasks. In essence, a minimum of one sentinel micro-task is needed to bootstrap this process. In interactive graphics, a user can "seed" this process by executing several of the micro-tasks himself and using these as the initial sentinel tasks.

Self-refereeing. In this approach, the results of some HPs are given to other HPs for approval, ranking, or selection. For instance, [Little et al. 2010; Bernstein et al. 2010] present Find-Fix-Verify: some

HPs are asked to improve a piece of text, while other HPs vote on whether to keep the changes. While successful in removing outliers from open-ended HC tasks, [Bernstein et al. 2010] reported that their Find-Fix-Verify algorithms produced erroneous output 12–14% of the time. Moreover, this approach increases the amount of HC and the data dependency between HPs, potentially delaying the final output and increasing the monetary cost.

As we seek rapid completion of simple, well-posed HC micro-tasks, we do not use self-refereeing for quality control. Instead, we send each HP a batch of micro-tasks containing each task twice in random order and we pose each micro-task to several HPs. We also insert in each batch sentinel operations originating from the real problem to prevent their identification. A discussion of the various parameters governing our choices is given in Section 7.

5. MICRO PERCEPTUAL HC ALGORITHMS

Micro-HC algorithm design does not assume that humans will solve the original problem. Rather, it seeks micro-tasks that can inform a solution by a machine algorithm. For this reason the chosen micro-tasks are not, in general, stated as the original problem. The machine part of the algorithm will *compose* the final solution using the results of the micro-tasks. Designing such an algorithm requires choosing simple visual queries as the micro-tasks, as well as the composition technique to combine them. Hence, a complex problem must be broken down to micro *human* computation tasks that can be distributed and solved very quickly. These tasks must be verified and then a *machine* composition part needs to obtain the final answer. This design is demonstrated in Figure 2.

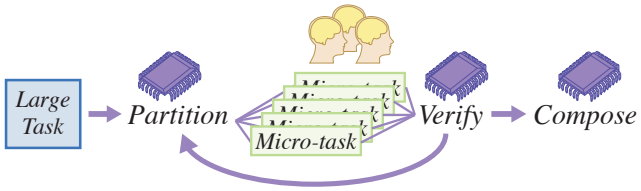


Fig. 2. A general pattern for the design of a Micro-HC algorithm: partition the problem into visual perception micro-tasks. Dispatch to human processors. Collect the answers and verify them (this may trigger more micro-tasks either because more details are needed or because results are not satisfactory). Finally, compose the answers to obtain the final result.

We demonstrate the design of Micro-HC algorithms in computer graphics for solving three hard problems: (i) extracting depth layers from a single photograph, (ii) extracting image normals from a photograph, and (iii) identifying bilateral symmetry of objects in a photograph. We implemented our algorithms using Amazon's Mechanical Turk [Amazon 2005] as the pool of human processors. Currently, a given task can be advertised on the Mechanical Turk with a simple description, time estimation, and amount of payment, and can be picked up and performed by any worker (Figure 3).

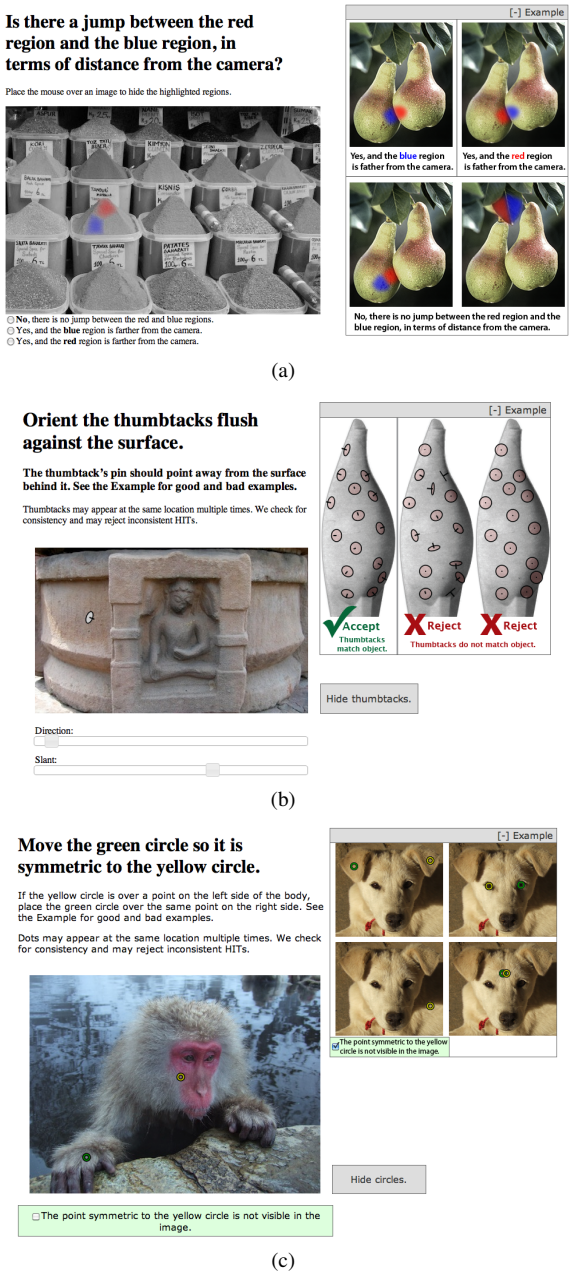


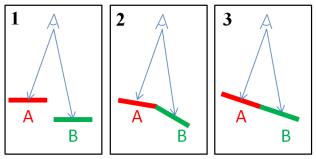
Fig. 3. Simple perceptual visual queries used as micro-tasks for human computation: (a) for finding depth, (b) for orienting normals, (c) for detecting symmetry. Note that no training is necessary as the simple instructions are shown with the query. (The photograph in (b), left, is copyright Warren Apel; photographs in (c) and the original photograph in (a), left, are copyright Yotam Gingold. The example in (b), right, depicts a piece of one of Brancusi's "Maiastra" sculptures.)

5.1 Depth Layers

Depth information (distance from the camera) is an important cue that can assist image manipulations. There are several ways to record the depth of pixels in an image at capture time, such as using stereo or depth cameras. However, estimating depths in a single, given image is much more challenging, and results on this problem are limited (see e.g. Figure 4). Calculating depths for fine-art or NPR pictures can be even more challenging (see e.g. Figure 5). Towards this end, we propose an HC algorithm that separates an image into layers using human computation. Our goal is similar to several previous works: we aim to be more robust than [Hoiem

et al. 2005; Assa and Wolf 2007; Saxena et al. 2009], which use automatic methods, and, in contrast to [Oh et al. 2001; Ventura et al. 2009; Sýkora et al. 2010], we do not require an expert user.

Micro-tasks. Full depth information means that we know the absolute depth value of each pixel in the given image. Designing a micro-task that asks "What is the distance of pixel (x,y) from the camera?" has several disadvantages: assessing the absolute distance of each pixel is difficult and in many cases unnecessary. On the other hand, asking for relative ordering between patches, e.g., "which part, A or B, is closer to the camera?" does not provide enough information as illustrated below (figure illustrates the scene from above). Knowing that A is closer than B cannot resolve ambiguities in the scene. In this case, the scene may contain: (1) discontinuous depth, (2) continuous depth but discontinuous gradients, or (3) continuous depth and smooth gradient.



As our goal is to extract layers from a single photograph, we target depth *discontinuities* and define a micro-task using a very simple interface. We ask the HP to determine whether there is a *depth-jump* between two adjacent, highlighted regions, and which, if any, is farther away (see Figure 3(a)). Other possibilities for designing the micro-tasks are discussed in Section 6. We use mean-shift [Comaniciu and Meer 2002] and SLIC [Achanta et al. 2010] to segment the image into regions. Then we gather all pairs of adjacent regions. We answer some queries on a small set of pairs to create a pool of sentinel operations, and also duplicate queries to test for consistency and seriousness. We repeat the process until we have $N = 3$ reliable answers per pair of neighboring regions and then take a majority vote. (No majority means no depth jump.) The HC algorithm is as follows:

DEPTH-LAYERS(image I, sentinel queries S)

```
1  Segment I into regions (using mean-shift and SLIC)
2  Insert all pairs of neighboring regions into Q
3  loop in parallel until each pair has been visited N times
4      Gather K random pairs from Q
5      Gather M random pairs from S
6      for each pair: Build the visual query & Duplicate it
7      Mix the 2K + 2M queries
8      results = send all queries to an HP
9      if average(consistency(results)) ≥ 0.75 and
10         average(sentinel(results)) ≥ 0.75
11             for each pair
12                 Add consistent results to the list of votes
13                 Increment #visited
14 for each pair of neighboring regions
15     final_result = majority(list of votes)
16 Solve the Laplace equation to construct a depth map
```

Composition. The relative depth ordering provides offsets of -1, 0, or 1 between adjacent regions in the image. Some applications such as simulated depth-of-field may not require more than region-to-region ordering. To reconstruct a continuous depth map we solve a Laplace equation with derivative constraints of -1, 0, or 1 across region boundaries. Results can be seen in Figure 4, and 5.

Note that solving a Laplace-equation is robust to inconsistencies, such as when one object is sometimes in front and sometimes behind another. At the cost of increased running time, the ± 1 constraints could be specified as simple inequality constraints in a convex optimization package, or an approach like [Amer et al. 2010] could be used. In the absence of cycles, an algorithm like topological sort could also be used.

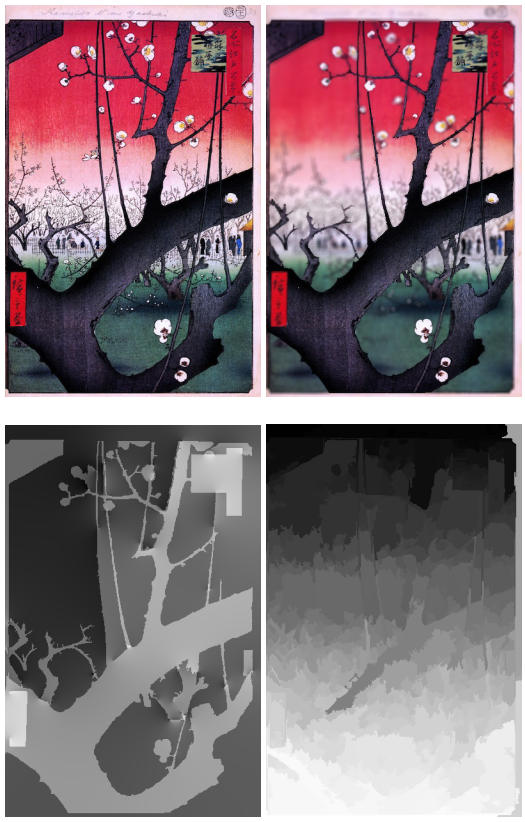


Fig. 5. Applying a depth-based blurring effect to Hiroshige's "Kameido Umeyashiki" woodblock print (top row) using the reconstructed depth layers of our human computation algorithm (bottom-left). Such inputs create a challenge for automatic algorithms; they are not cases [Saxena et al. 2009] was meant to handle (bottom-right).

5.2 Normal Map

Normal information is another important channel of information that can assist image manipulation [Bhat et al. 2010]. Using normals, one can alter the scene's lighting or modify its objects and update their illumination. One can also fit a surface to the normals to obtain a smooth depth map. Obtaining a normal map is difficult in general—as with depth maps, results are limited unless a special procedure was performed at capture-time. There are works that reconstruct normal maps based on user input such as [Wu et al. 2008], but we target a tighter coupling of HC into an interactive algorithm. We use a method similar to Koenderink et al. [1992] and Cole et al. [2009] to obtain normal directions from HPs, as it fits our micro-task requirements well.

Micro-tasks. A full solution would be to divide the image into pixels and ask, for each pixel, "What is the normal at pixel (x,y)?". Human observers perform this task via sophisticated cues within the image, as well as semantic information not contained in the im-

Fig. 4. Results of a Micro-HC algorithm for depth layering: The original photograph (left) & sample application (center-left): using the layers information to apply a depth-based blurring effect. Compare the reconstructed depth layers from HC micro-task results (center-right) to the reconstructed depth map from Make3D [Saxena et al. 2009] (right). Make3D performs well when an image's depth increases in the up direction and color similarity implies depth similarity (top: spices), but poorly in the absence of these two conditions (bottom: cups). (Original photographs copyright Yotam Gingold.)

age, such as familiarity with the world [Koenderink et al. 2001]. Again, it may be perceptually difficult to distinguish and analyze a single pixel, and since many regions are smooth, it may be enough to acquire a normal per small image patch. Hence, our decision was to use micro-tasks that ask for normals at the center of super-pixels defined on the image [Levinshtein et al. 2009]. Quality control is obtained by verifying consistency within the HP (duplication of each micro-task) and with sentinel operations. For each location we acquire two normals from each of three trusted HPs, and average the largest cluster. Note that prior to quality control and subsequent processing, we rectify raw output from each HP to account for humans' differing internal biases [Koenderink et al. 2001]. We do this by searching for the scale factor of normals' z components that minimizes the sum of angular differences between the HPs outputs for sentinel tasks and the correct outputs. (The total ambiguity is known as the *bas relief ambiguity* [Belhumeur et al. 1997], though this was found to be minimal in the micro-task design we use [Koenderink et al. 2001].)

We tested two micro-task designs. Both are borrowed from the perception literature [Koenderink et al. 1992], wherein normals are adjusted via two parameters, one for the xy direction and one for the slant. In one design, two sliders are linearly mapped to their angular quantities, the direction angle in the xy plane, and the angle between the 3D unit normal and the unit z direction (Figure 3(b)). In the other micro-task design, normals are specified by directly manipulating a large "controller" normal with the mouse (Figure 9, right). These variations are evaluated in Section 6.

As the HC algorithm is performed online, intermediate results can be used for adaptive refinement. In the case of calculating the normal maps, the criteria for refinement we used was the angle variation between neighboring superpixel samples. When the distance between them is not too small, and the angle variation is above a specified threshold we insert another sample between them and dispatch another micro-task. Note that adaptivity can increase accuracy and decrease the total amount of HC micro-tasks, as it concentrates them just where they are needed. However, it also reduces parallelism (i.e. the back arrow illustrated in Figure 2).

Composition. With the sparse map from (x,y) to a normal in $\mathbb{R}^3$, we generate a depth map by solving a BiLaplace equation ($\Delta^2 f = 0$) with sparse gradient constraints induced by the normals. This generates a depth map, which defines a surface and thus, via finite differencing, a consistent normal at every (x,y). See Figures 6 and 7 for results. Figures 15 and 14 compare our results to state-of-the-art automatic algorithms.

The HC algorithm outline is as follows:

NORMAL-MAP(image I, sentinel queries S)

```
1  Segment I into superpixels
2  Insert all superpixel centers into Q
3  loop in parallel until each element has been visited N times
4      Gather K random locations from Q
5      Gather M random locations from S
6      for each location: Build the visual query & Duplicate it
7      Mix the 2K + 2M queries
8      results = send all queries to an HP
9      results = unbias(results)
10     if average(consistency(results)) ≥ 0.75 and
11        average(sentinel(results)) ≥ 0.75
12           for each location
13                Add consistent results to the list of normals
14                Increment #visited
15 for each superpixel
16     final_normal = average(list of normals)
17 Solve the Bi-Laplace equation to construct a surface
18 Perform finite differencing to obtain a normal map
```

5.3 Bilateral Symmetry Detection

Detecting symmetry in objects in an image is an extremely difficult task that relies heavily on semantic information and prior knowl-

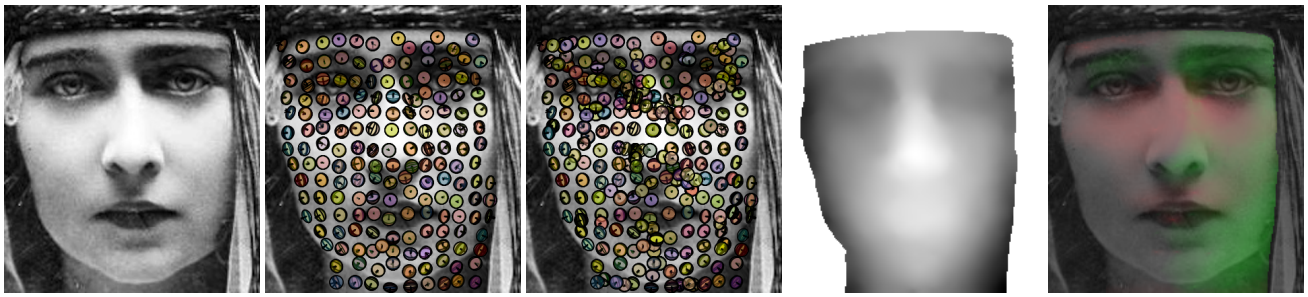


Fig. 6. Obtaining normals of an old photograph (from left to right): the input image and the oriented gauge figures returned by HPs, using uniform or adaptive algorithms, a depth map interpolating the normals, and adding two new lights to illuminate the face.

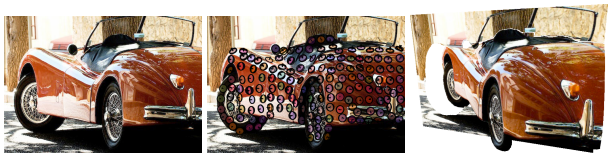


Fig. 7. The HC algorithm can be limited to a certain region of interest: finding normal on the car in this case. A surface reconstruction is used to render it from a different point-of-view. Note that this example is extremely challenging for automatic vision algorithms due to reflections and specularities. We have tried several automatic algorithms ([Saxena et al. 2009] and the Shape-from-Shading algorithms surveyed in [Durou et al. 2008]) but they could not provide meaningful results on this image (courtesy of Pedro Ribeiro Simões.)

edge [Chen et al. 2007]. On the other hand, it is very useful information for reconstruction, resizing, and other image processing operations. The task is not simple for a single human, either: drawing just the bilateral-symmetry curve on an image may be difficult and error-prone [Schmidt et al. 2009], and in any case is not sufficient, since the image contains a projection of a three-dimensional object with unknown geometry. Thus, given an image and a selected region, we propose an HC algoritm to build a point-wise bilateral symmetry map.

Micro-tasks. A dense solution to the bilateral symmetry problem would be a map from every point inside the object region to its symmetric opposite, excluding points with no visible symmetric opposite in the image. We assume that the symmetry map is a fairly smooth function, and create micro-tasks using a simple interface that displays a point in the image and asks the HP to find the symmetric point in the image, if it exists (see Figure 3(c)). As with the normal map algorithm, the set of sample points are the centers of super-pixels [Levinshtein et al. 2009], and our quality control uses duplication and sentinel operations. For each input point, we acquire two output points from each of three trusted HPs; the final output is the average of the largest cluster of these six points. The outline of this algorithm is similar to the Normal Map algorithm presented in the previous section, apart from the difference in query and composition.

Composition. With a sparse map mapping (x,y) points in the image to their corresponding (x',y') points (or to $\emptyset$), a dense symmetry map can be generated using a 2D interpolation technique such as moving least squares. See Figure 8 for example symmetry maps generated via our HC algorithm.

Fig. 8. Illustration of a Micro-HC algorithm for symmetry detection: input photographs and their bilateral symmetry maps. (Ballet photograph courtesy of flickr user dalbera.)

6. DESIGN VARIATIONS

There are various ways a problem can be partitioned into Micro-HC sub-problems. For visual tasks there are several choices of simple perceptual visual queries. Choosing which question to ask can be crucial to the quality of results as well as amount of time the algorithm will take to execute. Moreover, the machine part of the algorithm cannot be designed independently of the micro-task. In fact, an important principle of Micro-HC algorithm design is rephrasing the original problem into one which merits micro-tasks that can be performed easily and reliably by humans. Such micro-tasks involve apparent comparisons rather than obscure absolute quantity estimation. In addition, discrete micro-tasks are more reliable and easier to verify, since comparing output need not involve thresholds. We illustrate this principle with two variations on the design of our depth layers algorithm.

In Section 5.1, we presented the micro-task for our depth layers algorithm. In that micro-task, HPs determine whether adjacent image patches have discontinuous depth. We designed an alternative

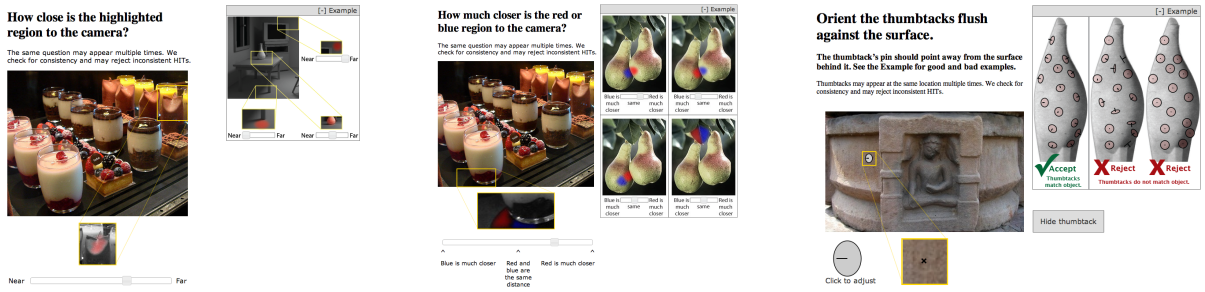


Fig. 9. Alternate micro-tasks designs: asking for absolute depth (left) and continuous relative depth (middle) to recover depth layers, and using direct manipulation of the thumbtack instead of sliders for normal maps (right). (Image of cups copyright Yotam Gingold, stone pillar copyright Warren Apel.)

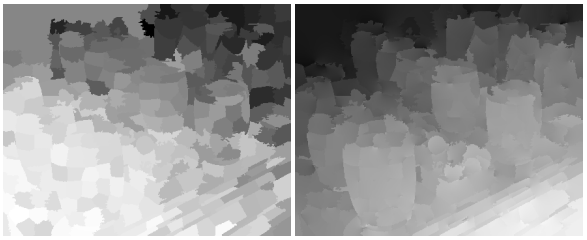


Fig. 10. Alternate micro-task results: Depth maps for the cups image from Figure 4, bottom, computed using the absolute depth micro-task (left) and the continuous relative depth micro-task (right).

micro-task that asks for absolute but *unitless* depth values for each superpixel. This micro-task displays a highlighted superpixel and asks HPs for a depth value on a continuous slider labeled "near" at one end and "far" at the other (Figure 9, left). The machine part of the algorithm simply averages the depths and then, for the purposes of finding depth layers, searches for discontinuities between neighboring superpixels. Figure 10, left, shows a depth map resulting from this micro-task. Unfortunately, it is well-established in the perception literature that humans are poor at monocular distance estimation [Healy et al. 2003]; the depth map is "noisy" and lacks discontinuities suitable for finding depth layers. Although approximately consistent with the scene, it is lower quality compared to the result from the depth layer algorithm presented in Section 5.1.

In a second micro-task variant (Figure 9, middle), we ask HPs to select the relative depth disparity between two neighboring superpixels on a *continuous* slider, rather than the discrete question we pose in Section 5.1. We then solve the Laplace equation using continuous rather than discrete depth difference constraints between superpixels. This micro-task design is "easier" than asking for absolute depth values because, to a first approximation, it involves an apparent comparison. However, HPs must still estimate an obscure quantity, the relative depth disparity. Compared to the discrete, discontinuous depth micro-task from Section 5.1, the average time per micro-task was somewhat higher, the results need rectification of humans' differing biases, and the final results are noisier, as shown in Figure 10, right. Hence, we found it preferable to obtain discrete, topological information from HPs, rather than deriving it from continuous, geometric data.

A different kind of design exploration involves testing micro-tasks' human factors. To illustrate this idea, we tested a variation of the normal map micro-task. We tested an interface for orienting the thumbtack using direct manipulation instead of sliders (Figure 9, right). We ran the algorithm using the two alternative micro-task designs on the same images at the same locations, and found that the two designs exhibit very similar statistical properties (the standard deviation from the average normal was similar—less than 10% difference—at corresponding locations).

7. EVALUATION

Quality Control. Most workers will perform the task correctly if they can, and if it is simple enough. Tasks which were more difficult to perform led to a higher failure rate of HPs, which increases the running time of the algorithm. This supports our decision to rely on the simplest visual queries. In our experience, this increase is primarily manifested as a higher percent of completely unreliable HPs; the reliability of other HPs is less affected.

Choosing the criteria for quality control is one of the most challenging aspects of an HC algorithm. The more complex the space of outputs, the more challenging the quality control mechanisms become. For discrete queries, as in our relative depth layer micro-tasks, comparing two outputs (for consistency or for agreement with sentinel operations) is simple. In contrast, output from a normal map micro-task requires rectifying biases, tolerances for determining whether two normals are close enough, and a parameter for clustering the output of all HPs at a given location to obtain the final result. Posing each micro-task to the HP twice (in random order) can be an effective consistency criteria, but malicious HPs can figure this out and answer consistently but incorrectly.

In all of our examples, sentinel queries were taken from the input image. This was necessary so that HPs could not differentiate between sentinel and non-sentinel queries. We believe this requirement could be obviated were there enough instances of the algorithm running in parallel to create micro-tasks from a large-enough variety of images such that HPs could not detect sentinels.

Since we deliberately did not include any training time, we designed a static "example" which is always visible to HPs (see Figure 3 and 9). Hence, the difference in timing per each of our examples could be caused either by confusing explanations or because of differences in complexity. Orienting a normal is more difficult than moving a point or answering a question. The complexity of the task did not directly affect the cost of our algorithm, since failed HPs need not be paid for their work. In our experience, HPs who are earnestly trying to complete tasks do not appreciate if the quality control criteria are too strict. Hence, we use two different measures: one to pay HPs and another to actually use their answers. Recall that micro-tasks are sent to an HP in batches, and each micro-task

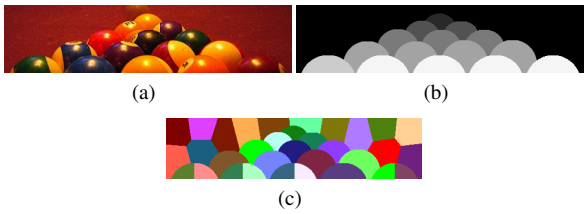


Fig. 11. (a) The 'billiards' input image for our quality control experiments. (b) The ground truth depth layers. (c) The segmentation used. (Photograph courtesy of flickr user fictures.)

is included twice in random order. We call a batch of HC *reliable* if it passes the consistency and sentinel tests; the consistency test is passed if the HP gave the same answer for both copies of each micro-task a high enough percent of the time, while the sentinel test is passed if the HP gave the correct answer for sentinel operations a high enough percent of the time. We set both our consistency and sentinel thresholds at 75%, and pay HPs for each batch of reliable HC. Only actually consistent answers are used in the algorithms.

To investigate the effects of varying parameters in our quality control setup, we created ground truth data for the depth layers algorithm (Figure 11). We call HC accurate if it matches the ground truth data. We investigated the effects of varying the thresholds used for consistency and sentinel tests, as well as varying the number of HPs to use when voting. All data presented for these experiments is averaged across four runs.

Figure 12(a) plots the percentage of HC batches that pass the consistency or sentinel test as a function of the threshold. Figure 12(a) also plots the average accuracy of all HC batches above the given threshold. We see that the vast majority (94%) of HC batches are 80% consistent (or more), though only 65% are 100% consistent. However, average accuracy is only marginally affected by increasing the consistency threshold: increasing it from 0% to 80% to 100% only increases the accuracy of HC from 87% to 88% to 90%.

HC batches are more stratified in their agreement with sentinel operations. Only 74% of HC batches gave the correct answer to 75% or more of the sentinel operations, and only 50% gave the correct answer to 100% of them. Increasing the sentinel threshold has a greater effect on average accuracy than increasing the consistency threshold; average accuracy increases from 87% to 94% as the sentinel threshold increases from 0% to 75%, and then to 96% accuracy with a sentinel threshold of 100% (at the cost of discarding 50% of HC batches). In Figure 12(b), we see that 57% of HC batches are discarded when both consistency and sentinel thresholds are set to 100%; with these thresholds, 97% average accuracy is achieved (see Figure 13, $N = 1$).

Increasing the number of HPs used in voting affects the accuracy of the final output by reducing the likelihood that the final output is affected by inaccurate HC that nonetheless passes the sentinel and consistency tests (and so is deemed reliable). Figure 13 visualizes this effect with varying sentinel and consistency thresholds. Each location in the plot depicts the probability that reliable HC from N different HPs produces the correct answer for the depth order between a pair of neighboring patches; the depicted value is the average over all pairs of neighboring patches. HC is chosen at random from among all reliable HC batches from all HPs in four experimental runs. The different heat maps depict the effect of varying the number N of reliable HPs used in voting.

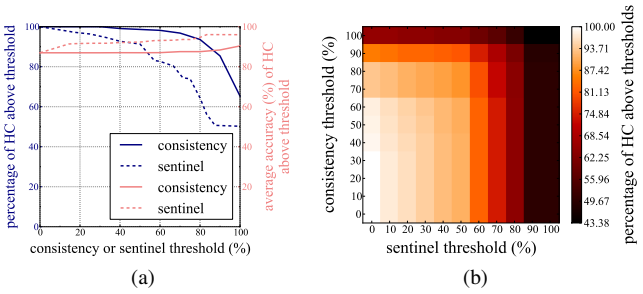


Fig. 12. The effect of varying consistency and sentinel thresholds across four runs of our depth layer algorithm on the 'billiards' image Figure 11. (a) The dark blue lines depict the percentage of HC batches above consistency or sentinel thresholds. For each line, only the consistency or only the sentinel test are used. The salmon-colored lines depict the average accuracy of HC above the given threshold. (b) A two-dimensional plot of consistency and sentinel thresholds; at each location, both tests are applied.

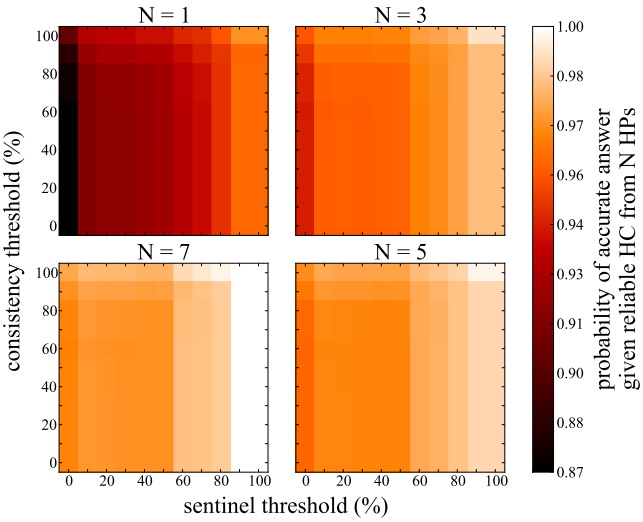


Fig. 13. The probability of obtaining accurate output when combining reliable HC from N different HPs chosen at random from among all reliable HC. Data is from four runs of our depth layer algorithm on the 'billiards' image (Figure 11). Probability is averaged over all needed micro-tasks (pairs of neighboring patches) for the input data.

While there is no obvious "sweet spot," it is interesting to note that one can achieve an expected 97% accuracy with only one reliable HC answer for each micro-task by setting the sentinel and consistency thresholds to 100%. However, these thresholds are too strict to use when deciding whether to pay HPs, because HPs expect to be paid for reasonable (∼75%), not perfect, performance. Still, it is more expensive to obtain 97% accuracy with reliable HC from $N = 3$ HPs and 75% consistency and sentinel thresholds (by paying for three times the number of micro-tasks actually needed) than to require 100% consistency and sentinel accuracy while paying HPs at 75% thresholds for reliable HC from $N = 1$ HP (paying for $\frac{100}{43} \cdot 72\% = 167\%$ of the number of accurate micro-tasks actually needed). Therefore, if 97% accuracy (or less) is sufficient, reliable HC should be collected from a single HP per batch. Identifying accurate HPs with lower sentinel and consistency overhead is an important direction for future work. The data we have pre-

sented implies that sentinel tasks alone may be sufficient. Higher than 97% accuracy cannot be achieved, however, without reliable HC from $N > 1$ HPs.

Granularity. Granularity is another important factor in algorithm design. On images one can define a micro-task at the level of one pixel, a small patch, or a whole object. Finer granularity can give higher precision but affects both timing and cost. Other factors to consider are the difficulty of image partitioning and characteristics of human perception. Pixels are very simple to extract while scene item segmentation is a challenging problem on its own, hence patches seem to strike the correct balance between the two. Also, it is perceptually more difficult for humans to answer pixel-size questions than region-based. We experimented with the depth layers algorithm and four granularities of the 'billiards' image (30, 60, 90, and 120 patches, taking care to capture object boundaries), and found no significant difference in accuracy.

As shown in Section 5.2, one can incorporate adaptiveness into more complex HC-algorithms. This has the potential to enhance quality and reduce the total cost of the algorithm, by focusing micro-tasks where most needed. On the other hand, adaptivity introduces data dependency which reduces parallelism. However, as discussed below, the primary real-world performance bottleneck on the Mechanical Turk is a lack of HPs, not saturating them.

Timing and Cost. Our micro-tasks were designed to be visual queries that could be answered almost instantly, and our results show that this is indeed the case: micro-tasks, on average, are completed in approximately 6–9 seconds (see Table III). Since we dispatch micro-tasks to HPs in batches of 20 (six unknown queries and four sentinel operations, doubled to test for self-consistency), the theoretically optimal running time (with no adaptivity) is approximately three minutes.

However, timing is a serious limitation today for interactive HC algorithms, due to the wait time between announcing tasks on the Mechanical Turk and workers beginning to work on it. In Table III, we show the delay between task announcement and completion for half and all of the HC needed for the algorithms. This wait time exhibits a strong "half-life" effect, wherein the rate at which micro-tasks are found by HPs decreases over time. The market properties of the Mechanical Turk have been studied extensively [Ipeirotis 2010; Chilton et al. 2010; Faridani et al. 2011; Mason and Suri 2011; Mason and Watts 2010], and it has been found that the completion rate of micro-tasks is correlated with payment, newness, and quantity of same-type micro-tasks (HPs favor micro-tasks that are newer and plentiful and pay more). Thus, since we ran our algorithms one-at-a-time, our micro-tasks became less favorable over time, extending time to completion. This effect can be mitigated or eliminated by requesting more tasks than necessary, or, as we envision in the future, by the continuous announcement of micro-tasks. (Paying HPs to wait for tasks, as in [Bigham et al. 2010; Bernstein et al. 2011], could be prohibitively expensive for the massive parallelism of our approach.)

The algorithms we presented cost \$.002–.003 per micro-task, or between \$3–17 to run on an image (Table I). Taking quality control (and Amazon's) overhead into account, the total cost per location for a normal or symmetric point using the HC normal map or symmetry map algorithms, is $\$.045–.065 \cdot (3/6) = \$.0225–.0325$, or $\$.0225–.0325 \cdot S$ for an image with S superpixels. The depth layer task operates on adjacent pairs of micro-tasks, so its cost is measured in terms of the number of adjacent superpixels E: $\$.045–.065 \cdot 3/6 \cdot E/2 = \$.01125–.01625 \cdot E$. For a study on the effects of varying the cost per micro-task, see [Mason and Watts 2010]; in their experiments, increasing payment increased the quantity but did not affect the quality of the HC performed.

Table I. Micro-tasks

example	micro-tasks used	ratio of used per executed	\$ per micro-task	total \$ cost
normal map	1620–4340	0.60	.002–.003	\$5.04–10.76
depth layers	2669–7620	0.76	.002	\$6.41–17.15
symmetry map	1020–1740	0.93	.002	\$3.24–3.92

Table II. Human Processors

example	total HPs	% completely unreliable	average reliability for reliable HPs	micro-tasks per HP	
				avg	median
normal map	61	42%	89%	123	33
depth layers	48	35%	87%	193	63
symmetry map	19	24%	99%	97	20

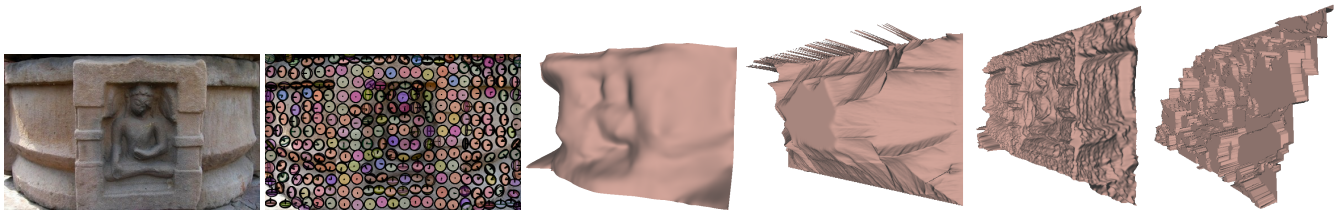
Table III. Timing

example	successful micro-task duration		algorithm delay until % complete	
	avg	median	50%	100%
normal map	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

8. COMPARISONS WITH OTHER ALGORITHMS

A Micro-HC algorithm needs to compete against both the best automatic algorithm and a manual human solution. Manual solutions can come in two flavors. First, an expert can create a solution based on high level interactive operations (see [Oh et al. 2001]). This option can also provide better control over the results. However, such an expert is usually more costly and not always available (e.g. in web applications). It may require dedicated software and considerable training and can still take more time. Moreover, the final results are more difficult to verify, as they are the work of one person that may have biases, inconsistencies, etc. The other evident option is to let a single person manually execute all the micro-tasks in any of our HC algorithms. In this case, fatigue and mistakes make the labor very difficult, and the advantage of using HC is similar to any embarrassingly parallel algorithm versus its sequential alternative.

On the other hand, when possible, we have compared the results of our algorithms to state-of-the-art fully automatic alternatives. We compared depth layer and normal map algorithms to the Make3D ([Saxena et al. 2009]) depth-from-a-single-image system and to several Shape-from-Shading approaches (via the recent survey of [Durou et al. 2008]). The results are shown in Figures 4, 5, 15, and



(a) (b) (c) (d) (e) (f)

Fig. 14. (a) The original photograph. (b) The sparse normal map created by human processors. (c) Using the normal map to reconstruct a smooth surface approximating the scene. The best outputs from the Shape-from-Shading approaches surveyed in [Durou et al. 2008]: (d) Falcone and Sagona and (e) Tsai and Shah. (f) The output from Make3D [Saxena et al. 2009] on the same input. (Original photograph copyright Warren Apel.)

Fig. 15. The face from Figure 6 (left) and the 3D surface reconstruction resulting from the HC normal map algorithm of Section 5.2 (middle) compared to Tsai and Shah, which gave the best output from the Shape-from-Shading approaches surveyed in [Durou et al. 2008] (right). Other Shape-from-Shading approaches failed to produce meaningful output, as did Make3D [Saxena et al. 2009].

14. We are not aware of algorithms capable of finding a general, nonrigid bilateral symmetry map in a 2D image (e.g. [Cornelius et al. 2007; Chen et al. 2007]; see [Liu et al. 2010] for a recent survey). Despite recent advances, automatic algorithms still make many simplifying assumptions. For example, no self-shadowing and simple lighting conditions for normal estimation, or that color similarity implies depth similarity and depth is increased as you move up the image for depth estimation. As can be seen in our examples, these conditions are not always satisfied.

9. CONCLUSIONS

We presented Micro Perceptual Human Computation, where there is a tight coupling between human and electronic processors. We presented a model where such algorithms could be effective in the context of visual tasks and graphics, using visual perceptual microtasks for humans. We demonstrated this model on three hard problems and compared our results to state-of-the-art techniques.

There are numerous avenues that this work could be extended to. Other hard graphics problems could be rephrased in terms of microtasks for human computation, including three-dimensional problems. The cost and timing should be reduced, as well as the quality control overhead. It would be interesting to embed Micro-HC algorithms in real applications as image manipulation tools. This would also spur interesting pricing issues for various tasks—how much will people be willing to pay to process their images?

Many open questions remain. Can any problem in graphics (and in general) be rephrased using simple visual perceptual HC queries? Can such queries be generalized? What other design possibilities for HC algorithms are there? Can HC be used effectively for synthesis (creativity) in addition to analysis? And what is the most efficient way to query the human visual system? This is akin to optimizing the efficiency of a perception experiment.

REFERENCES

ACHANTA, R., SHAJI, A., SMITH, K., LUCCHI, A., FUA, P., AND SUSSTRUNK, S. 2010. SLIC Superpixels. Tech. rep., EPFL.

ADAR, E. 2011. Why I hate Mechanical Turk research. In *CHI'11 Workshop on Crowdsourcing and Human Computation*.

ADOMAVICIUS, G. AND TUZHILIN, A. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Trans. on Knowledge and Data Engineering 17*, 734–749.

AHN, L. V., BLUM, M., HOPPER, N. J., AND LANGFORD, J. 2003. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology (Eurocrypt)*. 294–311.

AMAZON. 2005. Mechanical turk. http://www.mturk.com/.

AMER, M., RAICH, R., AND TODOROVIC, S. 2010. Monocular extraction of 2.1D sketch. In *ICIP*. 3437–3440.

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. 2010. A view of cloud computing. *Communications of the ACM 53*, 50–58.

ASSA, J. AND WOLF, L. 2007. Diorama construction from a single image. In *Eurographics '2007*. Eurographics Association.

BELHUMEUR, P. N., KRIEGMAN, D. J., AND YUILLE, A. L. 1997. The bas-relief ambiguity. In *Proceedings of IEEE CVPR*. 1060–1066.

BERNSTEIN, M. S., BRANDT, J., MILLER, R. C., AND KARGER, D. R. 2011. Crowds in two seconds: enabling realtime crowd-powered interfaces. In *Proceedings of ACM UIST*. 33–42.

BERNSTEIN, M. S., LITTLE, G., MILLER, R. C., HARTMANN, B., ACKERMAN, M. S., KARGER, D. R., CROWELL, D., AND PANOVICH, K. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of ACM UIST*. 313–322.

BHAT, P., ZITNICK, C. L., COHEN, M., AND CURLESS, B. 2010. GradientShop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph. 29*, 10:1–10:14.

BIGHAM, J. P., JAYANT, C., JI, H., LITTLE, G., MILLER, A., MILLER, R. C., MILLER, R., TATAROWICZ, A., WHITE, B., WHITE, S., AND YEH, T. 2010. VizWiz: nearly real-time answers to visual questions. In *Proceedings of ACM UIST*. 333–342.

BRANSON, S., WAH, C., BABENKO, B., SCHROFF, F., WELINDER, P., PERONA, P., AND BELONGIE, S. 2010. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*.

CHEN, P.-C., HAYS, J. H., LEE, S., PARK, M., AND LIU, Y. 2007. A quantitative evaluation of symmetry detection algorithms. Tech. Rep. CMU-RI-TR-07-36, Robotics Institute, Pittsburgh, PA. September.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A benchmark for 3D mesh segmentation. *ACM Trans. Graph. 28,* 3 (Aug.).

CHILTON, L. B., HORTON, J. J., MILLER, R. C., AND AZENKOT, S. 2010. Task search in a human computation market. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP)*. 1–9.

COLE, F., SANIK, K., DECARLO, D., FINKELSTEIN, A., FUNKHOUSER, T., RUSINKIEWICZ, S., AND SINGH, M. 2009. How well do line drawings depict shape? *ACM Trans. Graph. 28,* 3 (Aug.).

COMANICIU, D. AND MEER, P. 2002. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24,* 5, 603 – 619.

CORNELIUS, H., PERD'OCH, M., MATAS, J., AND LOY, G. 2007. Efficient symmetry detection using local affine frames. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*. 152–161.

CROWDFLOWER. 2007. Crowdflower. http://crowdflower.com/.

DUROU, J.-D., FALCONE, M., AND SAGONA, M. 2008. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding 109*, 22–43.

FARIDANI, S., HARTMANN, B., AND IPEIROTIS, P. 2011. What's the right price? pricing tasks for finishing on time. In *Proceedings of the AAAI Workshop on Human Computation (HCOMP)*.

GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM 35*, 61–70.

GRIER, D. A. 2005. *When Computers Were Human*. Princeton University Press.

HAYES, B. 2008. Cloud computing. *Communications of the ACM 51,* 7, 9–11.

HEALY, A. F., PROCTOR, R. W., AND WEINER, I. B., Eds. 2003. *Experimental Psychology*. Handbook of Psychology, vol. 4. Wiley.

HEER, J. AND BOSTOCK, M. 2010. Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. In *ACM Human Factors in Computing Systems (CHI)*. 203–212.

HOIEM, D., EFROS, A. A., AND HEBERT, M. 2005. Automatic photo pop-up. 577–584.

HUANG, E., ZHANG, H., PARKES, D. C., GAJOS, K. Z., AND CHEN, Y. 2010. Toward automatic task design: A progress report. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP)*.

IPEIROTIS, P. G. 2010. Analyzing the amazon mechanical turk marketplace. *ACM XRDS 17*, 16–21.

IPEIROTIS, P. G., PROVOST, F., AND WANG, J. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP)*.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D mesh segmentation and labeling. *ACM Trans. Graph. 29,* 3 (July).

KOENDERINK, J. J., VAN DOORN, A. J., AND KAPPERS, A. M. L. 1992. Surface perception in pictures. *Perception & Psychophysics 52,* 5, 487–496.

KOENDERINK, J. J., VAN DOORN, A. J., KAPPERS, A. M. L., AND TODD, J. T. 2001. Ambiguity and the 'mental eye' in pictorial relief. *Perception 30*, 431–448.

LEVINSHTEIN, A., STERE, A., KUTULAKOS, K. N., FLEET, D. J., DICKINSON, S. J., AND SIDDIQI, K. 2009. TurboPixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 2290–2297.

LITTLE, G., CHILTON, L. B., GOLDMAN, M., AND MILLER, R. C. 2010. TurKit: Human computation algorithms on Mechanical Turk. In *Proceedings of ACM UIST*.

LIU, Y., HEL-OR, H., KAPLAN, C. S., AND GOOL, L. V. 2010. Computational symmetry in computer vision and computer graphics. *Foundations and Trends in Computer Graphics and Vision 5*, 1–195.

MASON, W. AND SURI, S. 2011. Conducting behavioral research on amazon's mechanical turk. *Behavior Research Methods*.

MASON, W. AND WATTS, D. J. 2010. Financial incentives and the "performance of crowds". *SIGKDD Explor. Newsl. 11*, 100–108.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH*. 433–442.

QUINN, A. J. AND BEDERSON, B. B. 2011. Human Computation: a survey and taxonomy of a growing field. In *Proceedings of ACM SIGCHI*. 1403–1412.

RUSSEL, B. C., TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2008. LabelMe: a database and web-based tool for image annotation. *77,* 1–3 (May), 157–173.

SAMASOURCE. 2008. Samasource. http://www.samasource.org/.

SAXENA, A., SUN, M., AND NG, A. Y. 2009. Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 824–840.

SCHMIDT, R., KHAN, A., KURTENBACH, G., AND SINGH, K. 2009. On expert performance in 3D curve-drawing tasks. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*. 133–140.

SHAHAF, D. AND HORVITZ, E. 2010. Generalized task markets for human and machine computation. In *National Conference on Artificial Intelligence*.

SOROKIN, A., BERENSON, D., SRINIVASA, S., AND HEBERT, M. 2010. People helping robots helping people: Crowdsourcing for grasping novel objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

SPIRO, I., TAYLOR, G., WILLIAMS, G., AND BREGLER, C. 2010. Hands by hand: Crowd-sourced motion tracking for gesture annotation. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 17–24.

SÝKORA, D., SEDLÁČEK, D., JINCHAO, S., DINGLIANA, J., AND COLLINS, S. 2010. Adding depth to cartoons using sparse depth (in)equalities. *Computer Graphics Forum 29,* 2.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory modeling with collaborative design spaces. *ACM Trans. Graph. 28*, 167:1–167:10.

TXTEAGLE. 2008. txteagle. http://txteagle.com/.

VENTURA, J., DIVERDI, S., AND HÖLLERER, T. 2009. A sketch-based interface for photo pop-up. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*.

VON AHN, L. 2005. Human computation. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

VON AHN, L. AND DABBISH, L. 2004. Labeling images with a computer game. In *Proceedings of ACM SIGCHI*. 319–326.

VON AHN, L. AND DABBISH, L. 2008. General techniques for designing games with a purpose. *Communications of the ACM 51,* 8, 58–67.

WU, T.-P., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2008. Interactive normal reconstruction from a single image. *ACM Trans. Graph. 27*, 119:1–119:9.

YUEN, J., RUSSELL, B. C., LIU, C., AND TORRALBA, A. 2009. LabelMe video: Building a video database with human annotations. In *IEEE 12th International Conference on Computer Vision (ICCV)*. 1451–1458.

Received May 2011; revised October 2011; accepted December 2011