Two Tools are Better Than One: Tool Diversity as a Means of Improving Aggregate Crowd Performance

Jean Y. Song, Raymond Fok, Alan Lundgard, Fan Yang, Juho Kim[†], Walter S. Lasecki

CROMA Lab | MISC Group
Electrical Engineering and Computer Science
University of Michigan – Ann Arbor
{jyskwon, rayfok, arlu, yangtony}@umich.edu
wlasecki@umich.edu

[†]KIXLAB
School of Computing
KAIST – Daejeon, Republic of Korea
juhokim@kaist.ac.kr

ABSTRACT

Crowdsourcing is a common means of collecting image segmentation training data for use in a variety of computer vision applications. However, designing accurate crowd-powered image segmentation systems is challenging because defining object boundaries in an image requires significant fine motor skills and hand-eye coordination, which makes these tasks error-prone. Typically, special segmentation tools are created and then answers from multiple workers are aggregated to generate more accurate results. However, individual tool designs can bias how and where people make mistakes, resulting in shared errors that remain even after aggregation. In this paper, we introduce a novel crowdsourcing workflow that leverages *multiple tools* for the same task to increase output accuracy by reducing systematic error biases introduced by the tools themselves. When a task can no longer be broken down into more-tractable subtasks (the conventional approach taken by microtask crowdsourcing), our multi-tool approach can be used to further improve accuracy by assigning different tools to different workers. We present a series of studies that evaluate our multi-tool approach and show that it can significantly improve aggregate accuracy in semantic image segmentation.

CCS Concepts

•**Information systems** → **Crowdsourcing;** •**Human-centered computing** → **Human computer interaction (HCI);** •**Computing methodologies** → *Computer vision;*

Author Keywords

Crowdsourcing; human computation; multi-tool aggregation; tool diversity; semantic image segmentation; computer vision

IUI'18, March 7–11, 2018, Tokyo, Japan

ISBN 978-1-4503-4945-1/18/03…$15.00

DOI: https://doi.org/10.1145/3172944.3172948

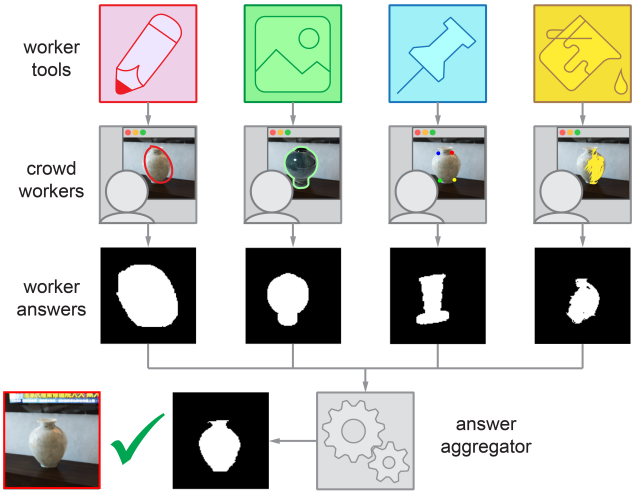


Figure 1: This paper introduces an approach to leveraging *tool diversity* that uses multiple different tools for the same task to improve aggregate crowd performance by reducing systematic error biases that might otherwise result from using any one tool type alone. Our findings on an image segmentation task demonstrate that tool diversity can provide output at least comparable to the best constituent tool and always yields significantly better results than the weakest constituent tool.

INTRODUCTION

The goal of *image segmentation* is to demarcate objects in a visual scene from the background, allowing computer vision (CV) systems to learn to recognize specific objects. These CV systems can, in turn, enable autonomous cars to identify pedestrians, surveillance drones to focus on potential threats, and in-home robots to help people with motor or mobility impairments live more comfortably and independently.

Perceiving the boundaries of physical objects comes naturally for people, but it remains a challenging open problem for CV systems due to the complexity of understanding the semantics of visual scenes [14, 26]. Crowd-powered object segmentation

tools can bridge this gap by leveraging human understanding to produce large, manually-demarcated training data sets (e.g., [2, 12, 25]) for CV systems. However, designing crowd-powered tools that produce high-accuracy training data and scale efficiently (with respect to human-time cost) remains an open problem because the task of manually marking object boundaries requires significant hand-eye coordination and fine motor skills, resulting in a high error rate if these tasks are performed too quickly by workers.

Many web-based image segmentation tools (e.g., [1, 2, 5, 11, 24, 32]) have been designed to help workers reduce the effort needed to complete a task and to increase the accuracy of their output. However, different tool designs induce different error biases in worker performance, which can lead to systematic mistakes when only a single tool is used. For example, some tools [2, 32] provide polygon drawing functionality to help trace object boundaries, but [2] reported that workers often skip selecting part of the object if automatic scrolling during selection is not provided. We consider this to be a *systematic error bias* because the same error pattern would be unlikely to emerge if the tool were designed differently. In other words, it would be unlikely for worker outputs from Click'n'Cut [5] (which asks workers to use left/right mouse clicks to identify foreground and background regions of an image) to make the same mistakes as when using the polygon drawing tool. However, Click'n'Cut may exhibit its own systematic error pattern, induced by limitations in its own design. More generally, we consider error patterns that are found to be common among worker outputs from a single tool to be systematic error biases, as they are likely to be induced by the design of the tool itself. These errors are different from, for example, human perceptual biases that may also systematically affect outcomes.

In this paper, we propose the idea of leveraging *tool diversity* as a means of overcoming these systematic error biases by improving aggregate crowd performance. Tool diversity is the extent to which tools designed for the same task differ from one another in the systematic error biases that they induce. Unlike standard aggregation methods in crowdsourcing, which try to design and use the best single tool available with many workers in order to reach high accuracy, we show that using multiple effective tools can diversify the error patterns in worker responses, and help systems achieve higher *combined* accuracy (Figure 1). This insight is motivated by ensemble learning methods in machine learning that use multiple learning algorithms to obtain better prediction than can be obtained from any of the constituent algorithms alone [7].

To demonstrate our proposed workflow, we introduce a multi-tool crowd-powered image segmentation system, FourEyes, which leverages pairs of tools to generate better aggregate responses. Our evaluation shows the effectiveness of tool diversity through comparisons of aggregate crowd performance across different tool pairings with equally-sized groups of workers using the same tools. We show that the output accuracy of heterogeneous tool pairs are better than homogeneous sets. Moreover, we model the multi-tool aggregation problem as an optimization problem and use expectation maximization (EM) [6] to estimate consensus image segmentations. For certain tool pairs, EM-based multiple tool aggregation significantly improved output accuracy over majority voting. Finally, we characterize our problem and summarize the properties of crowdsourcing tasks that are amenable to our approach: those that are objective, are tractable enough for workers to produce nearly-correct responses, and increase in correctness as additional answers are provided, can benefit from our approach.

The key contributions of this research are:

- A novel crowdsourcing approach that combines input across different *types* of tools to improve aggregate quality.
- FourEyes, a crowd-powered image segmentation system that implements our approach, combining the output of four different tool types to improve the collective accuracy of a group of workers using a single segmentation tool.
- Experimental results from 51 objects across 12 indoor scenes segmented by 288 crowd workers using four different tools that validate our system's effectiveness and suggest the benefits of our multi-tool approach.

RELATED WORK

Conventional approaches to improving crowd worker output accuracy include microtask decomposition and consensus-based aggregation. These approaches are usually intended to reduce task complexity and correct for variance in individual worker responses, respectively. However, when it comes to systematic error biases induced by a tool's design, errors can persist even after decomposition or aggregation, since all workers use the same tool and the tool used in the workflow may still induce biases in worker responses. Our tool diversity strategy builds on prior work in crowdsourcing workflows and answer aggregation strategies to reduce these systematic error biases. In this section, we discuss related work in designing crowdsourcing workflows and improving output quality by answer aggregation.

Crowdsourcing Workflows

In crowdsourcing, breaking large tasks into smaller microtasks has been a popular strategy to increase the accuracy of crowd workers' answers. Microtasks are small, context-free units of work that are widely used in crowdsourcing workflows. Crowdsourcing platforms, such as Amazon Mechanical Turk, post these small units of work that (typically quasi-anonymous [22]) crowd workers can accept and complete. Soylent [3] showed that dividing a larger task into Find-Fix-Verify steps improves the accuracy of crowd workers' answers in document editing tasks. ToolScape [16] introduced a Find-Verify-Expand workflow to enhance the process of extracting different steps in how-to videos.

Other research has explored how to use crowd workflows to collectively accomplish what no single worker could alone. For instance, continuous [20, 18] or instantaneous [27] crowdsourcing workflows have made real-time crowdsourcing possible. These allow systems to get on-demand human responses when immediate judgments are needed. TimeWarp [19] introduced a workflow that invites a group of workers to perform a task that otherwise cannot be accomplished by a single worker.

In their speech captioning application, working in a group helps avoid a systematic bias of the captioning task without adding any delay. CrowdMask [15] uses a pyramid workflow to mask private content in images. While the system cannot hide sensitive content from a single worker, their method segments and distributes segments of user content so that workers can mark potentially private content without fully viewing it. Similarly, WearMail [34] introduced a workflow that allowed crowds to train a system on demand to accomplish an email search task without ever revealing email contents to workers.

Our work contributes to this line of research by introducing a novel crowdsourcing approach that aggregates multiple crowd-powered tools to offer better performance than from any of the constituent tools alone. To the best of our knowledge, we are the first to study methods for using tool diversity to increase aggregate accuracy in a crowdsourcing environment.

Improving Output Quality by Aggregating Answers

A common strategy to improve output quality in crowdsourcing systems is to aggregate independent workers' answers on the same task into a single response, typically via a consensus method like voting. Even simple majority voting has been shown to produce accurate results for crowdsourcing tasks, such as linguistic annotation tasks [33] and document editing tasks [3]. In terms of image segmentation tasks, ground truth segmentations of objects have been generated via majority pixel voting with manually-collected answers from multiple crowd workers or experts [12, 25]. More sophisticated approaches using unsupervised learning have been used to weight workers' answers by using models of their abilities [4, 23, 36, 37]. Deluge [4] models workers' sensitivity and specificity to detect noisy workers, and LazySusan [23] tracks workers by assigning different weights based on the accuracy of a worker's answer. In [36] and [37], the authors present probabilistic approaches to model not only the workers, but also the properties of the data being labeled.

One well-known weighted aggregation method is the EM algorithm [6], which predicts unknown (latent) correct answers by estimating weights for each crowd worker's answers. Dawid and Skene [6] showed that the EM algorithm significantly outperforms majority voting when a majority of workers' responses are correct and they are conditionally independent given the ground truth answer. The EM algorithm is suitable for utilizing tool diversity in image segmentation tasks because: i) the majority of the pixels selected by any tool are assumed to be correct and ii) the tools' performances are independent of each other. When designing tools, a tool's abilities are not known in advance because designers are typically unaware of the input images that the system will see. Because the performance of each tool can vary with images or object types, we can consider a tool's ability as the latent variable to be predicted. Therefore, we apply the EM algorithm across different tools with the goal of maximizing the performance of the aggregated output.

APPROACH

Prior work has used task decomposition—the process of breaking down larger tasks into more manageable, focused pieces of work—to make tasks more approachable for non-expert crowd workers. Once task decomposition has been used to break down a larger unit of work as much as possible within a corresponding workflow, most crowdsourcing systems then recruit multiple workers in parallel to further improve accuracy by aggregating their answers. We propose using multiple different tools across different workers to complete the same [sub]task, instead of having all workers complete the same task with the same interface or tool. Our proposed approach fills in the gap where traditional task decomposition leaves off.

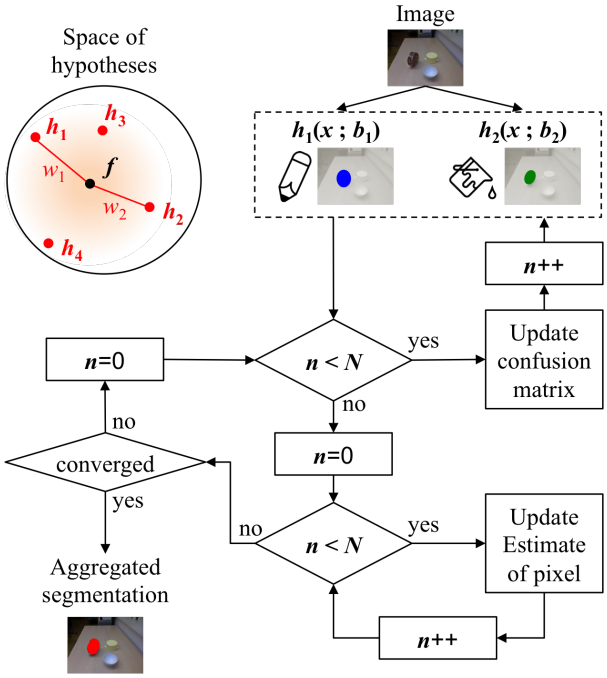


Figure 2: The top left diagram shows the hypothesis space of the possible segmentation tools, including the best performing tool (f) and other possible hypotheses ($h_1 \ldots h_4$). The flowchart on the right shows the EM algorithm we adopted for the optimization. Two different image segmentation tools, h_1 and h_2, each with different biases, b_1 and b_2 (respectively), pass segmented images to the system. We estimate the weights,w_1 and w_2 to approximate the performance of f.

Our work is conceptually motivated by ensemble learning in machine learning. Ensemble learning methods are machine learning algorithms that construct a set of learning algorithms and predict a new data point by taking a weighted vote of the predictions from each learning algorithm [7, 8]. It has been proven that ensembles can often perform better than any single member [7]. Algorithm accuracy (i.e., better than random guessing) and diversity are necessary and sufficient conditions for a combination of algorithms to be more accurate than any of its individual constituents [13]. The top left diagram in Figure 2 shows how ensemble methods work. In the diagram, a learning algorithm can be viewed as searching a space of hypotheses to identify the best performing hypothesis f, which

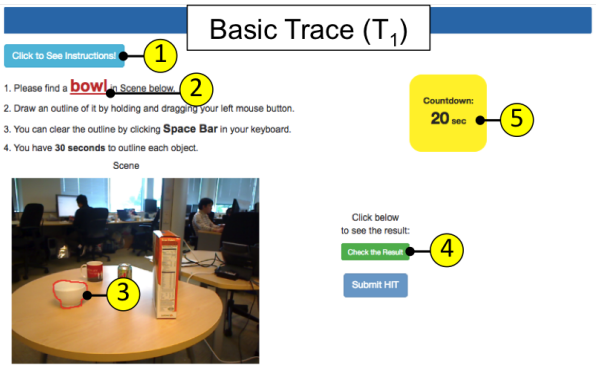


Figure 3: Overview of the Basic Trace image segmentation tool. (1) shows the full instruction when clicked. (2) describes the query object which is shown to the workers in random order. (3) shows an example of a trace line a worker provided. (4) shows the segmentation result when clicked. (5) is the task timer, which serves as an encouragement to workers to consider time in their work.

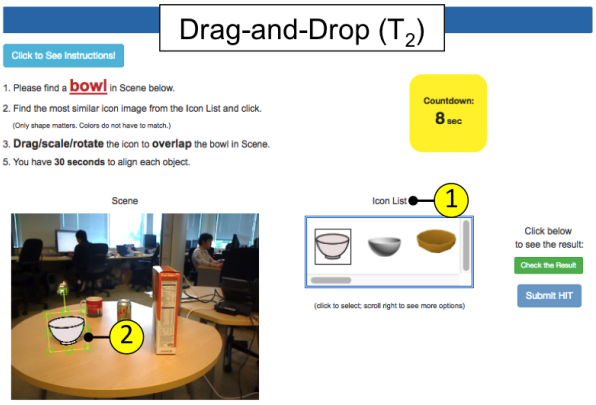


Figure 4: Overview of the Drag-and-Drop image segmentation tool. (1) shows the template images list. (2) shows an example of the chosen template image being aligned to the query object.

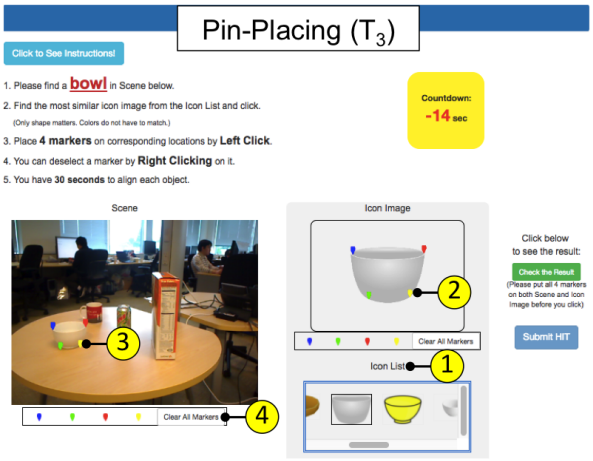


Figure 5: Overview of the Pin-Placing image segmentation tool. (1) shows template images list. (2) and (3) show examples of chosen points by a worker. (4) lets workers reset the points to start over.

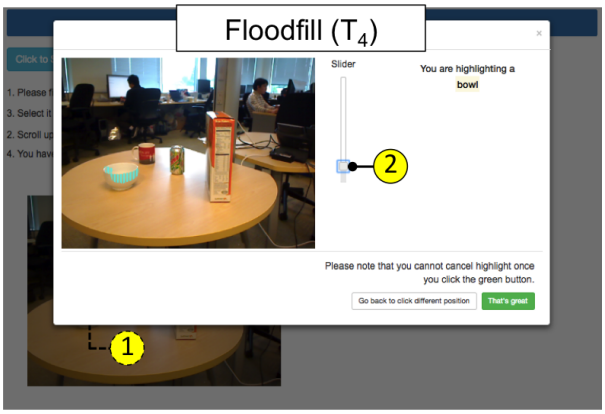


Figure 6: Overview of the Floodfill image segmentation tool. (1) shows that a worker can click on a query object in the given scene. (2) is a slider that allows workers to adjust a parameter to control the propagation of the selection area.

can be computationally difficult to find. Ensemble learning constructs a combination of two alternative hypotheses h_1 and h_2 with proper weights (w_1 and w_2), and approximates the best hypothesis f by averaging the two. Our tool diversity approach is analogous to ensemble learning methods in that two image segmentation tools are combined to produce a better segmentation result. In Figure 2, the bottom left shows two different realizations of web-based image segmentation tools, $h_1(x;b_1)$ and $h_2(x;b_2)$, with different error biases, b_1 and b_2, respectively. We propose using the EM method to optimize the weights w_1 and w_2 that maximize the combined accuracy of h_1 and h_2.

CHOOSING THE TOOLS

In this section, we introduce four web-based segmentation tools that we designed to instantiate and test the tool diversity concept. We considered one key question when designing the tools: "How can we diversify the errors produced by different tools?" Because it is hard to predict what errors will be induced by a given tool, we built tools specialized to work well with objects with different characteristics, such as small or transparent objects, objects with fuzzy materials, and reflective surfaces. These objects are current challenges to both automatic segmentation methods and human annotators. The way we built these tools increases the likelihood that each tool performs differently for different types of objects, resulting in greater error diversity. We categorized these object types into three groups and created tools that are designed to minimize errors in each particular object category. The spaces we explored and the tools we designed are summarized in Figure 7. We used the Question (Q), Option (O), and Criteria (C) representation [28] of the design space for deciding which tools to build. The Question indicates a key design issue, the

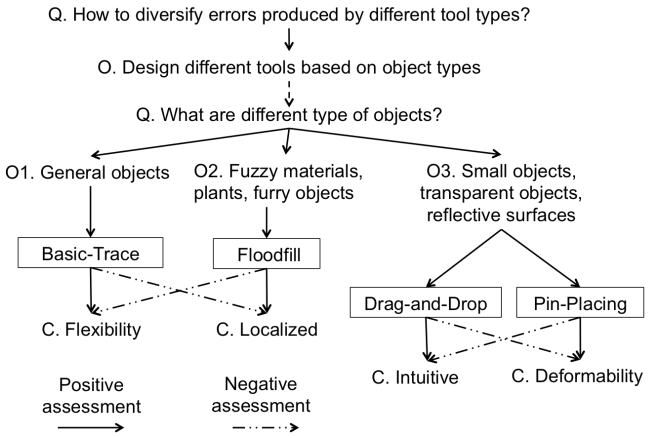


Figure 7: Design space we considered when choosing the tools for the study. We used the Question (Q), Option (O), and Criteria (C) representation of the design space.

Option node suggests possible answers to the Question, and the Criteria item represents the core properties expected from choosing an Option. For one of the Options (O3 in Figure 7), we differed the interface in two ways (Drag-and-Drop and Pin-Placing) so that the interaction of users can create different artifacts. We observed that different interactions lead to different error patterns, so we include both of the tools in the experiment section. In the rest of this section, we provide detailed descriptions of the four tools developed.

Basic Trace

The first tool is a free-form drawing tool shown in Figure 3. With Basic Trace, workers click and drag their mouse to trace the outline of the query object in a scene (Figure 3 ③). Once a worker submits the initial trace line, a simple image processing algorithm connects the gaps and fixes the irregularities in the traced line in order to form a smooth shape. It then highlights the pixels inside the traced shape, and returns the result as the final object segmentation. Of the four tools, the Basic Trace is the most manual, with the highest degree of freedom. The strength of this tool is that it is highly flexible and workers can segment any type of objects if sufficient time is given. However, the weakness of the tool is that if a worker is idle and not careful enough, the output can easily be very poor, e.g., a worker may draw a rough box around an object instead of carefully following the demarcation from the background.

Drag-and-Drop

The second tool lets workers select an object template from a list (Figure 4 ①), which is generated by searching images of a target object from an image search engine like Google or Bing. These images are then filtered for transparency and size, and the top N (in this paper we use $N = 12$) are downloaded to construct a template list for each query object. Workers are asked to select the template that most accurately matches that object in the scene based on its shape, proportion of dimensions, and perspective. In Drag-and-Drop, workers overlay their selected template onto the object identified in the scene (Figure 4 ②). Workers are able to scale, rotate, and drag the template to adjust the angle and dimensions of the template in an attempt to closely match the shape of the actual object. Based on a worker's transformation of the template, the system determines the final object segmentation by identifying and annotating the overlapping pixels between the template and the scene. The strength of this tool is that it is very intuitive to use. However, the weakness of the tool is that it is hard to map deformable objects, or even rigid objects if being viewed from different perspectives.

Pin-Placing

The third tool is also a template-based tool called Pin-Placing. A template list is generated in the same manner as in Drag-and-Drop (Figure 5 ①). With this tool, workers select four arbitrary points on their selected object template (Figure 5 ②), and pair them with four corresponding points on the object in the scene (Figure 5 ③). Workers can modify individual points, or clear all points at once (Figure 5 ④). After the four pairs of points are submitted, an automatic transformation algorithm is run to transform the template image to produce the final object segmentation. Note that four pairs of control points are the minimum necessary to perform a non-linear deformation between two images, given the perspective limitation inherent in a fixed-angle view in two dimension. Pin-Placing's working mechanism is similar to sophisticated techniques (e.g., that professional radiologists use for diagnosing lesions), but it is more intuitive to novice workers. One drawback of template-based approaches is that if an object in a scene has an atypical shape, none of the template images in the list may have a shape similar to the object. In this case, a possible solution may be to let workers ask to switch to a different tool that does not require choosing a template image.

We note that the two template-based tools, Drag-and-Drop and Pin-Placing, force workers to select occluded parts of a target object when there exist overlaps with other objects. This is a more useful functionality in domains like robotics, where ground truth object geometry includes hidden parts as well. However, in this study, we only consider the visible parts of a target object as the region of interest because it is a more general way of indicating objects in two-dimensional image segmentation. As a consequence, the tools necessarily select more false positive regions than the other tools.

Floodfill

Floodfill (AKA Bucket-fill) is a mostly autonomous tool, combining a simple region growing method [35] with minimal human input to initialize the seed point and tune a threshold parameter. Workers click on the object they want to segment (Figure 6 ①) and adjust a slider to tune one of the algorithm's threshold parameters (Figure 6 ②). This triggers the RGB Floodfill algorithm which highlights all neighboring pixels sharing an RGB value similar to the seed point that was clicked. If the segmentation is unsatisfactory, either failing to select the entire object or exceeding the object boundary, workers can adjust the slider to modify the highlighted area. The tool is strong if the shape of an object is complex with many curves, but only when the object is mostly monochromatic. If

a query object consists of polychromatic or shaded regions, the selection area can be smaller than the actual object boundaries because the algorithm cannot propagate across these regions.

SYSTEM INTERFACES

To collect image segmentations from the crowd, we created a system named FourEyes that implements the tools above. FourEyes begins by receiving a scene image and the user's request in the form of a natural language query, e.g., "mark the bowl." The query is parsed to find nouns and then the nouns are displayed to workers (Figure 3 ②) in order to help identify objects that need to be segmented from the scene. For each tool, a short four to five steps of instruction including the target object name is displayed to workers while conducting the task. Also, workers can check the segmentation result before they submit their work ("Check the Result" button in Figure 3 ④). To discourage workers from idling, we embedded a task timer (Figure 3 ⑤) that counts down from t seconds and goes into negative when time runs out. In this paper, we used $t = 30$ in all experimental conditions. The timer does not impact workers other than to serve as an encouragement to consider time in their work.

EXPERIMENTS

To understand the effect of tool diversity on improving aggregate crowd performance, we recruited 288 crowd workers using LegionTools [10] and routed them from Amazon Mechanical Turk to FourEyes. Workers were given one of the four tools to perform a task of image segmentation. Note that we gave different tools to different workers because we consider the smallest unit of microtask as one worker segmenting one object using a single tool. We recruited six unique workers for each tool-scene pair (four tools and 12 scenes), resulting in a total of 1224 object segmentations.

Dataset

We chose a dataset that included various indoor objects. The dataset included 12 different visual scenes, each containing three to seven objects, for a total of 51 objects. The scenes were gathered from publicly-available datasets[1,2], and represent typical indoor scenarios with commonplace objects. They ranged from a living room to a tabletop, and contained everyday objects (e.g., a plant, laptop, soda can, cereal box, flashlight, etc.). In the experiment, each worker was shown one scene and a series of objects to segment depending on the number of objects in the scene. For each task, the order of objects in each list was randomized to avoid any ordering bias. Each worker was given one scene with one tool to perform a segmentation task.

Instructions and Payment

Before crowd workers could begin the task, they were shown a short instructional video demonstrating the goal of the task and how to use the tool they would be provided with. The two tools Drag-and-Drop and Pin-Placing have longer videos because they explain choosing the most similar template image. This additional step delayed workers' performance time in the actual experiment as well. Workers were also shown pictures exemplifying desired and undesired segmentations (same example images for all tools) so that they understand the aim of the task to create a detailed boundary of a target object in a scene. If the worker decided to proceed, they were directed to the FourEyes worker UI and their subsequent interactions with the UI were recorded. Task instructions were also accessible at any time if necessary (Figure 3 ①). Each worker was paid between \$0.35 and \$0.60 per task, proportional to the number of objects they had to segment and on the expected time of a given tool (a pay rate of ∼\$10/hr). The expected time of each tool was determined by its average latency time from a dozen of preliminary experiments.

[1] https://rgbd-dataset.cs.washington.edu/dataset.html/

[2] https://www.doc.ic.ac.uk/ ahanda/VaFRIC/iclnuim.html/

Segmentation Quality Evaluation

To assess success on the image segmentation task, we measured the accuracy of each by comparing the similarity to the ground truth segmentation that was generated manually by the authors for the experiments. One author carefully completed the task and another author verified the quality of the resulting ground truth.

We used precision, recall, and F_1 score (the harmonic mean of precision and recall) to compute the pixel-level similarity (Equation 1). To do this, the number of true positive, false positive, and false negative pixels were counted for each crowdsourced segmentation.

$$\begin{aligned} \text{Precision} &= \frac{\text{true positive}}{(\text{true positive} + \text{false positive})} \\ \text{Recall} &= \frac{\text{true positive}}{(\text{true positive} + \text{false negative})} \\ \text{F1Score} &= \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})} \end{aligned} \quad \text{(Eq. 1)}$$

Results

The different tools had different error patterns (trade-offs) in terms of precision and recall. Figure 8 shows example worker segmentations from each tool, alongside the ground truth. Figure 9 shows precision-recall scatter plots of the segmentation result with each dot representing one object. We averaged the precision and recall of six workers who segmented the same object with the same tool, so one dot represents the average precision-recall of an object. As shown in Figure 9 (a) and (b), Basic Trace and Drag-and-Drop tended to show high recall but low precision. We observed that with these two tools, workers tended to select objects by putting large margins around the objects, resulting in high recall but low precision. Examples of segmentation using these two tools are shown in Figure 8 (a) and (b), respectively. Meanwhile, Pin-Placing resulted in the most scattered performance as shown in Figure 9 (c). This implies that the performance of the tool varies a lot depending on object types. An example of using Pin-Placing is shown in Figure 8 (c). In the example, a worker selected a template image that is very different from the query object, resulting in both low precision and low recall. Lastly, Figure 9 (d)

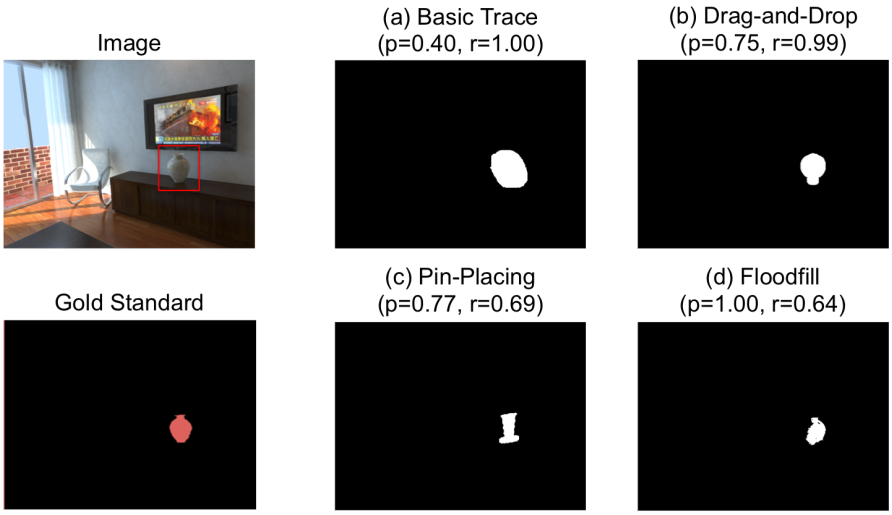
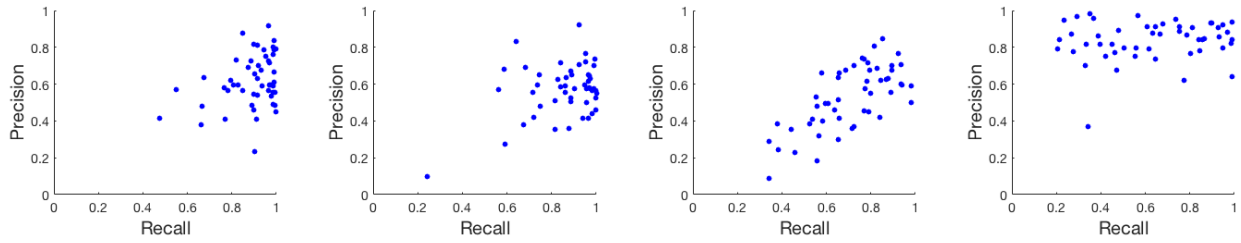


Figure 8: Original image (top left), ground truth image (bottom left), and exemplar segmentations using the four tools with their precision and recall values reported on top. (a) Basic Trace, (b) Drag-and-Drop, (c) Pin-Placing, and (d) Floodfill. In summary, the different tools have different precision-recall patterns. Therefore, we can expect that EM-based multiple tool aggregation would increase the overall accuracy of the segmentation task.



(a) Basic Trace (b) Drag-and-Drop (c) Pin-Placing (d) Floodfill

Figure 9: Precision-recall scatter plot of four different tools. We can see that the error patterns are diverse across different tools in terms of precision and recall. Basic Trace (a) and Drag-and-Drop (b) show high recall but low precision, Pin-Placing (c) shows the most scattered pattern, implying that the tool's performance highly depends on the query object, and Floodfill (d) shows high precision but low recall.

	Precision	Recall	F_1 score
Basic Trace	0.62 (0.14)	0.89 (0.12)	0.71 (0.13)
Drag-and-Drop	0.57 (0.14)	0.86 (0.15)	0.66 (0.13)
Pin-Placing	0.53 (0.17)	0.71 (0.17)	0.58 (0.17)
Floodfill	0.84 (0.11)	0.63 (0.25)	0.67 (0.20)

Table 1: Average performance (and standard deviation) of the four individual tools.

shows that Floodfill tended to give high precision but low recall performance. We observed that the selection area with Floodfill tended to be smaller than the actual object boundaries due to boundaries that were shaded or colored differently. An example segmentation of using Floodfill is shown in Figure 8 (d). Because one side of the vase was much brighter, a worker could not select the entire image with the seeded region growing algorithm.

Average accuracy metrics for each tool are summarized in Table 1. Since F_1 score is the harmonic mean of precision and recall, we use F_1 *score* as our measure of overall accuracy. In general, Floodfill gave the best performance in terms of precision and Basic Trace gave the best recall and F_1 score.

To calculate latency, we measured overall task time starting from when the worker began interacting with the task to when the worker clicked "submit" at the end of the task. After dropping outliers more than two standard deviations (2σ) from the mean latency, Basic Trace's average latency was 14.37 seconds ($\sigma = 8.08$), Drag-and-Drop's was 24.89 seconds ($\sigma = 11.25$), Pin-Placing's was 20.77 seconds ($\sigma = 7.90$), and Floodfill's was 12.62 seconds ($\sigma = 10.41$). The template-based tools had a higher segmentation latency than the other two tools. This was expected because the template-based tools are more involved and perhaps less intuitive to general-purpose crowd workers. Using Floodfill, some workers managed to

	Precision	Recall	F_1 score
Basic Trace	0.62 (0.17)	0.99 (0.04)	0.74 (0.14)
Drag-and-Drop	0.57 (0.16)	0.95 (0.11)	0.69 (0.14)
Pin-Placing	0.56 (0.17)	0.82 (0.16)	0.66 (0.16)
Floodfill	0.86 (0.12)	0.69 (0.23)	0.73 (0.17)

Table 2: Average performance (and standard deviation) of pixel-level majority voting on single-tool aggregation.

produce a satisfactory segmentation within three seconds, but others spent extra time trying to perfect their segmentation, with diminishing returns in accuracy.

From these primary results, we observed different error patterns (Figure 9) across the four tools that we designed. The result matches our design intent to diversify the errors produced by different tools. Now we can think of each tool as alternate hypotheses h_1, h_2, h_3, and h_4 of the optimal hypothesis f, with different error biases b_1, b_2, b_3, and b_4, respectively. As in ensemble learning, we expect that aggregating the different tool pairs will improve the output accuracy by reducing accumulated systematic error biases.

COMPARISON OF AGGREGATION METHODS

In order to evaluate the effectiveness of our tool diversity approach, we conducted a series of studies to examine the performance improvement achieved from an ensemble of two tools. As a baseline condition, we first investigate the segmentation quality of single tool aggregation based on majority voting of four different workers. We implement a pixel-level majority voting method, with each answer weighted equally. In the second study, we do the same majority voting, except on two tool pairs combining four workers' answers from two different tools. In the last study, we apply the EM method on multiple tool aggregation of two tools to optimize the tools' weights adaptively per object. Our studies show that the tool diversity approach is a safe design strategy that guarantees performance at least as good as the superior constituent tool.

Method 1. Single-Tool Aggregation (Baseline)

Single tool aggregation combines answers from the four workers who used the same tool to segment target objects. We ran 10 random sampling of all possible worker pairs from the collected data. This was performed to avoid any bias from accidentally choosing a good or bad pair of workers. For each query object in the scene, pixel-level majority voting was performed to annotate each pixel as either background or object. If more than two workers labeled a pixel as belonging to the query object, then the pixel was included. The accuracy of the final segmentation was computed as in the Segmentation Quality Evaluation section. The results of 10 random sampling were averaged per each query object. We summarized the average results in Table 2.

The change in precision was not significant with single tool aggregation compared to the average precision without aggregation. However, recall and F_1 scores improved. For example, recall of Basic Trace increased by 11% ($p < .01$) compared to its average performance without aggregation (see Table 1). The increase in recall is a natural consequence of answer aggregation with low agreement thresholds. If the agreement threshold is higher, recall would decrease because more consensus is needed to annotate a pixel as an 'object'. In the next sections, we observe if and how further improvements can be achieved with multiple tool aggregation.

Method 2. Multiple Tool Aggregation (Uniform Voting)

Adding multiple tools for the same task can improve the aggregate accuracy when the tools compensate systematic error biases of each other. In this section, we focus on the results of two-tool pairs to provide a more in-depth evaluation and discussion. For all possible pairs of two tools, we ran 10 random sampling and chose two workers from each tool (combinations of six things, taken two at a time), a total of four workers. The average performance of all possible tool pairs as in Method 1 is summarized in Table 3.

	Precision	Recall	F_1 score
Basic Trace × Drag-and-Drop	0.61 (0.13)	0.99 (0.02)	0.74 (0.10)
Basic Trace × Pin-Placing	0.60 (0.13)	0.95 (0.07)	0.72 (0.11)
Basic Trace × Floodfill	0.74 (0.13)	0.93 (0.12)	0.81 (0.12)
Drag-and-Drop × Pin-Placing	0.57 (0.15)	0.92 (0.11)	0.69 (0.13)
Drag-and-Drop × Floodfill	0.70 (0.12)	0.92 (0.09)	0.78 (0.09)
Pin-Placing × Floodfill	0.70 (0.13)	0.86 (0.13)	0.76 (0.12)

Table 3: Average performance (and standard deviation) of pixel-level majority voting of all possible tool pairs.

Multiple tool aggregation improves F_1 scores compared to single tool aggregation. Every tool pair except Basic Trace × Pin-Placing (0.72) showed increased F_1 scores compared to the single constituent tools. The pair gave better F_1 scores than aggregating Pin-Placing alone (0.66), but gave a 2.7% lower F_1 score than aggregating Basic Trace alone (0.74). However, there was no statistically significant difference between single tool aggregation of Basic Trace versus multiple tool aggregation of Basic Trace × Pin-Placing pair. One notable finding about the result is that the highest F_1 score achievable from single tool aggregation is 0.74 (aggregating Basic Trace alone), whereas that from multiple tool aggregation is 0.81 (aggregating Basic Trace × Floodfill pair), which is a 9.5% ($p < .01$) performance improvement with mixing tools. To emphasize the performance improvement in terms of F_1 score, we compared the F_1 scores of multiple tool aggregations (blue bars) with their constituent tools (red and green bars) in Figure 10.

Method 3. Multiple Tool Aggregation (EM method)

For this study, we used the same 10 worker groupings from Method 2 to do a fair performance comparison. We used the EM method from Dawid and Skene [6] to estimate the labels

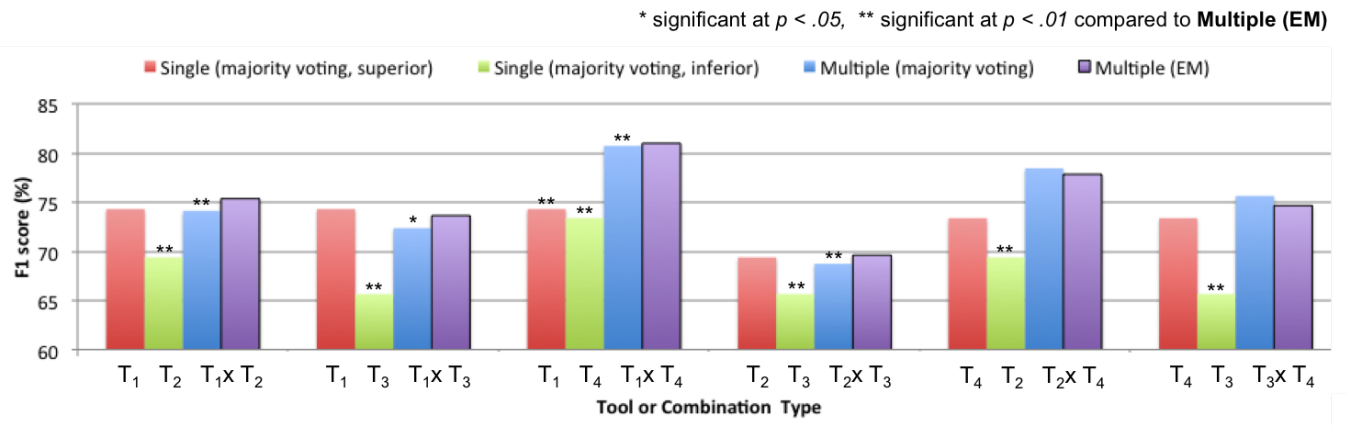


Figure 10: Accuracy (F_1 score) comparison of different aggregation methods based on four tools: Basic Trace (T_1), Drag-and-Drop (T_2), Pin-Placing (T_3), and Floodfill (T_4). Blue bars are multiple tool aggregation with majority voting and purple bars are multiple tool aggregation with the EM method. Red bars are single tool aggregation of the better performing tool and green bars are single tool aggregation of the less performing tool among the two constituent tools. * significant at $p < .05$; ** significant at $p < .01$ both compared to EM-based multiple tool aggregation (two tailed t-test). The tool diversity strategy always performed significantly better than the inferior constituent tool, and performed at least comparable to the superior tool.

of each pixel. We consider the ground truth labels as latent variables and estimate them jointly with the unknown parameters, the weight per tool, of our model. We model our problem of estimating the correct consensus image segmentations as follows:

Assume M crowd workers segment an object in an image A having N total pixels. Each pixel is labeled as either 1 (object) or 0 (background) by workers. The label a worker m assigns to each pixel is denoted as $z_{mn} \in \{0,1\}$. We denote all labels from worker m as a vector Z_m. The true label Y_n, where $n = 1,\ldots,N$, of each pixel is unknown. The true labels of A to be estimated are denoted as a vector Y. In the Dawid-Skene algorithm, it is assumed that the probability of worker m labeling a pixel is independent of choosing a pixel, i.e., it is a constant over n. In addition, we denote by θ the confusion matrices set to be estimated. We can estimate the true labels Y by maximizing the marginal log-likelihood of the observed worker labels.

$$l(\theta) := \log\Big(\sum_{Y\in\{0,1\}^n} L(\theta;Y,Z)\Big). \qquad \text{(Eq. 2)}$$

The EM algorithm works iteratively by applying two steps: the expectation step and the maximization step.

E Step: Calculate the expected value of the log-likelihood function, with respect to the conditional distribution of Y given Z under the current estimate of θ:

M Step: Find the estimate θ that maximizes the expectation of marginal log-likelihood.

The E and M steps are repeated until the estimations converge. The diagram on the right side of Figure 2 shows how the process is applied to our problem. The scene image is given as an input to two tools h_1 and h_2 that have different error models b_1 and b_2, respectively. Crowd workers use the tools to segment a query object, and the responses are transferred to the EM algorithm. Initial latent variables are set as the majority voting result, and the confusion matrix for each response is updated based on the initial assumption of the latent variables. Confusion matrices are updated by counting the number of false positive, false negative, true positive, and true negative pixels. Once the confusion matrices are updated for every pixel, the new estimations of latent variables are updated until convergence. The accuracy of multiple tool aggregation with the EM method is summarized in Table 3. The comparison of F_1 scores of the tool pairs with that of the constituent tools is summarized in Figure 10. The numbers for majority voting on single tool aggregation are obtained from Method 1, and the numbers for majority voting on multiple tool aggregation are obtained from Method 2. The p-values were computed using two tailed t-tests with Bonferroni correction applied after each t-test. The results show that the EM-based multiple tool aggregation always performed significantly better than the inferior constituent tool, and performed at least as well as the superior constituent tool.

The summarized result shows that the EM method significantly improves the performance of the tool pairs compared to uniform majority voting, except for the two tool pairs (Drag-and-Drop $\times$ Floodfill pair and Pin-Placing $\times$ Floodfill pair). From these exceptions, we observed that the trade-off between precision and recall was low because drop in recall was larger than the gain in precision, resulting in the overall F_1 scores not benefiting much from using the EM algorithm. Although there was a significant performance improvement, the gain using EM was small (under 3%).

DISCUSSION

We observed that the two highest aggregate-performance tool pairs were combinations of a high-precision (but low-recall)

	Precision	Recall	F_1 score
Basic Trace × Drag-and-Drop	0.63 (0.14)	0.98 (0.02)	0.75 (0.11)
Basic Trace × Pin-Placing	0.63 (0.14)	0.93 (0.09)	0.74 (0.12)
Basic Trace × Floodfill	0.75 (0.13)	0.93 (0.12)	0.81 (0.12)
Drag-and-Drop × Pin-Placing	0.59 (0.15)	0.90 (0.12)	0.70 (0.13)
Drag-and-Drop × Floodfill	0.71 (0.13)	0.90 (0.11)	0.78 (0.10)
Pin-Placing × Floodfill	0.72 (0.14)	0.81 (0.14)	0.75 (0.14)

Table 4: Average performance (and standard deviation) of all possible tool pairs using per-pixel EM algorithm-based weighted voting.

tool and a high-recall (but low-precision) tool, as shown in the third and fifth bar groups in Figure 10. Precision and recall often have an inverse relationship, where one can be increased at the cost of reducing the other. In the crowdsourcing literature, research has investigated different payment schemes to observe the trade-off between precision and recall on object annotation tasks [29]. Our work suggests that different tools can be built to target either high precision or high recall so that the harmonic means of both can be maximized by aggregating results from two methods. More generally, our results indicate that leveraging tool diversity in crowdsourcing tasks can improve aggregate crowd performance by compensating for various types of inherent individual systematic error biases.

Generalizability

While we demonstrate this new crowdsourcing paradigm using an image segmentation task, it could benefit any task where different approaches to solving the same problem can be devised. Specifically, tasks that have the following properties would be especially amenable to our approach:

- Expected correctness grows non-negatively with added worker input. In other words, on average, quality improves (collective answers converges to correct) as more worker responses are collected. Problems where majority voting works would belong to this class.
- The task is tractable enough to yield approximately-correct responses from workers, but responses can be expected to have imperfections. asks such as real-time captioning [18] or handwriting recognition [30] are examples of such tasks.
- The task has an objectively correct answer, but also tolerates imperfections from workers' responses. For example, creative writing tasks would *not* be a good fit because there is no single correct answer, and they do not tolerate imperfections well (e.g., incomplete sentences).
- The expected human error is distributed differently when using different tools. This way, a diverse tool set can complement a broad range of error types. If this were not the case (i.e., if the errors were all biased in the same direction), then we would not expect multiple tools to be significantly more effective than a single one alone.

Many common crowdsourcing problems (e.g., in computer vision, natural language processing, or robotic/UI manipulation) have these properties, suggesting that a range of domains beyond the one explored in this paper may also benefit from our approach.

CONCLUSION AND FUTURE WORK

In this paper, we have introduced a generalizable crowdsourcing approach of leveraging tool diversity to increase output accuracy of a system. When building a crowd-powered system, different tool designs can induce different worker performance, leading to systematic error biases when only a single tool is used for a task. We have shown that these systematic error biases can be reduced by using multiple tools to improve aggregate crowd performance on image segmentation tasks.

Overall, our findings open new opportunities and directions for pursuing a deeper understanding of how tool designs influence the aggregate performance on diverse crowdsourcing tasks. Future work may investigate ways to generalize methodologies for leveraging tool diversity in other domains, such as video coding [17], annotation of fine-grained categories [9], or activity recognition [21]. Tool diversity may also help seamlessly integrate computer and human input [31] to more efficiently train machine learning algorithms.

ACKNOWLEDGMENTS

The authors would like to thank Stephanie O'Keefe and Kyle Wang for their input on this work. This research was supported in part by the Denso Corporation, Toyota Research Institute, and MCity at the University of Michigan. This research was also supported in part by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017M3C4A7065960).

REFERENCES

1. Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. 2016. What's the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision*. Springer, 549–565.
2. Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. 2013. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 111.
3. Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. 2015. Soylent: a word processor with a crowd inside. In *Communications of the ACM*, Vol. 58. ACM, 85–94.
4. Jonathan Bragg, Mausam, and Daniel S. Weld. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *First AAAI conference on human computation and crowdsourcing*.
5. Axel Carlier, Vincent Charvillat, Amaia Salvador, Xavier Giro-i Nieto, and Oge Marques. 2014. Click'n'Cut:

Crowdsourced interactive segmentation with object candidates. In *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia*. ACM, 53–56.

6. Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979), 20–28.
7. Thomas G Dietterich and others. 2000. Ensemble methods in machine learning. *Multiple classifier systems* 1857 (2000), 1–15.
8. Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*. Springer, 23–37.
9. Timnit Gebru, Jonathan Krause, Jia Deng, and Li Fei-Fei. 2017. Scalable annotation of fine-grained categories without experts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 1877–1881.
10. Mitchell Gordon, Jeffrey P Bigham, and Walter S Lasecki. 2015. LegionTools: a toolkit+ UI for recruiting and routing crowds to synchronous real-time tasks. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 81–82.
11. Sai Gouravajhala, Jean Y Song, Jinyeong Yim, Raymond Fok, Yanda Huang, Fan Yang, Kyle Wang, Yilei An, and Walter S Lasecki. 2017. Towards Hybrid Intelligence for Robotics. *Collective Intelligence Conference (CI)* (2017).
12. Danna Gurari, Mehrnoosh Sameki, and Margrit Betke. 2016. Investigating the Influence of Data Familiarity to Improve the Design of a Crowdsourcing Image Annotation System. *HCOMP* (2016).
13. Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence* 12, 10 (1990), 993–1001.
14. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. *CoRR* abs/1703.06870 (2017).
15. Harmanpreet Kaur, Mitchell Gordon, Yiwei Yang, Jeffrey P Bigham, Jaime Teevan, Ece Kamar, and Walter S Lasecki. 2017. Crowdmask: Using crowds to preserve privacy in crowd-powered systems via progressive filtering. In *Proceedings of the AAAI Conference on Human Computation (HCOMP 2017)., HCOMP*, Vol. 17.
16. Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing Step-by-step Information Extraction to Enhance Existing How-to Videos. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 4017–4026.
17. Walter S. Lasecki, Mitchell Gordon, Danai Koutra, Malte F. Jung, Steven P. Dow, and Jeffrey P. Bigham. 2014. Glance: Rapidly coding behavioral video with the crowd. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 551–562.
18. Walter S. Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. 2012. Real-time captioning by groups of non-experts. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 23–34.
19. Walter S. Lasecki, Christopher D. Miller, and Jeffrey P. Bigham. 2013. Warping Time for More Effective Real-time Crowdsourcing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2033–2036.
20. Walter S. Lasecki, Kyle I. Murray, Samuel White, Robert C. Miller, and Jeffrey P. Bigham. 2011. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 23–32.
21. Walter S. Lasecki, Young Chol Song, Henry Kautz, and Jeffrey P. Bigham. 2013. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 1203–1212.
22. Matthew Lease, Jessica Hullman, Jeffrey P Bigham, Michael S Bernstein, Juho Kim, Walter S. Lasecki, Saeideh Bakhshi, Tanushree Mitra, and Robert C Miller. 2013. Mechanical turk is not anonymous. *Social Science Research Network (SSRN)* (2013).
23. Christopher H. Lin, Mausam Daniel, and S. Weld. 2012. Crowdsourcing control: Moving beyond multiple choice. In *In: Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence, UAI*.
24. Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. 2016. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3159–3167.
25. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
26. Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
27. Alan Lundgard, Yiwei Yang, Maya L. Foster, and Walter S. Lasecki. 2018. Bolt: Instantaneous Crowdsourcing via Just-in-Time Training. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA.

28. Allan MacLean, Richard M Young, Victoria ME Bellotti, and Thomas P Moran. 1991. Questions, options, and criteria: Elements of design space analysis. *Human–computer interaction* 6, 3-4 (1991), 201–250.

29. Andrew Mao, Ece Kamar, Yiling Chen, Eric Horvitz, Megan E Schwamb, Chris J Lintott, and Arfon M Smith. 2013. Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing. In *First AAAI conference on human computation and crowdsourcing*.

30. Tom Ouyang and Yang Li. 2012. Bootstrapping Personal Gesture Shortcuts with the Wisdom of the Crowd and Handwriting Recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2895–2904.

31. Olga Russakovsky, Li-Jia Li, and Li Fei-Fei. 2015. Best of both worlds: human-machine collaboration for object annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2121–2131.

32. Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. 2008. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision* 77, 1 (2008), 157–173.

33. Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 254–263.

34. Saiganesh Swaminathan, Raymond Fok, Fanglin Chen, Ting-Hao Kenneth Huang, Irene Lin, Rohan Jadvani, Walter S. Lasecki, and Jeffrey P. Bigham. 2017. WearMail: On-the-Go Access to Information in Your Email with a Privacy-Preserving Human Computation Workflow. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 807–815.

35. Shane Torbert. 2016. *Applied computer science*. Springer. 158 pages.

36. Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. 2010. The Multidimensional Wisdom of Crowds. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2424–2432.

37. Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2035–2043.