Eliciting and Leveraging Input Diversity in Crowd-Powered Intelligent Systems

by

Jean Y. Song

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2019

Doctoral Committee:

Assistant Professor Walter S. Lasecki, Chair
Professor Jason J. Corso
Assistant Research Scientist Brent Griffin
Assistant Professor Juho Kim
Assistant Professor Casey Pierce

Jean Y. Song

jyskwon@umich.edu

ORCID iD: 0000-0003-4379-3971

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Walter S. Lasecki, the best advisor I can ask for, who provides endless support to his students and encourages to take on new, difficult, but the most exciting challenges. I would also like to thank my co-advisor Prof. Juho Kim who is not only my academic role model, but also the best father and a husband I have ever met. This dissertation would not have been possible without the support from my mentors, friends, and family. I would like to thank my dissertation committee Prof. Jason J. Corso, Dr. Brent Griffin, and Prof. Casey Pierce who gave me valuable, constructive, and critical feedback to strengthen the dissertation. I was lucky enough to have excellent mentors during my doctoral study, including Prof. Jeffrey A. Fessler, Prof. Charles R. Meyer , Prof. Yoonsik Choe, Prof. Demos Teneketzis, Prof. Sang Won Lee, Dr. Stephanie OKeefe, Dr. Jonathan Kummerfeld, Prof. Simon Perrault, and Prof. David Fouhey – who I thank for their valuable support and advices. I am grateful to have smart, thoughtful, and energetic academic friends all around the world. It was truly fun to interact and collaborate with these people during my doctoral study: Yan Chen, John Joon Young Chung, Sai R. Gouravajhala, Jordan Huffaker, Harman Kaur, Rebecca Krosnick, Anthony Liu, Divya Ramesh, Zhefan Ye, Jinyeong Yim, Stephan J. Lemmer, Michael Xieyang Liu, Shiyan Yan, Raymond Fok, Alan Lundgard, Fan Yang, Kyle Wang, Minsuk Chang, Jibon Naher, Yoonseo Choi, Kabdo Choi, Donghoon Han, Kyung Je Jo, Seoyoung Kim, Hyunwoo Kim, Eunyoung Ko, Sung-Chul Lee, Jisoo Lee, Hyungyu Shin, Saelyne Yang, Arti Thakur, Evey Huang, Manav Rao, Jaeyoon Song, Hyeungshik Jung, Ray

Hong, Paul Grau, Jaeyeon Lee, and many more. It was my family that supported me through hard times. I would like to thank all my family members, parents and brothers and sisters, for their ongoing love.

Last but not least, I am more than grateful to have my husband, Inyong Kwon, and my lovely son, Koojoon Kwon in my life, who make me want to be a better person, and work away to change the world be a better place.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

ABSTRACT

Collecting high quality annotations plays a crucial role in supporting machine learning algorithms, and thus, the creation of intelligent systems. Over the past decade, crowdsourcing has become a widely adopted means of manually creating annotations for various intelligent tasks, spanning from object boundary detection in images to sentiment understanding in text. This thesis presents new crowdsourcing workflows and answer aggregation algorithms that can effectively and efficiently improve collective annotation quality from crowd workers. While conventional microtask crowdsourcing approaches generally focus on improving annotation quality by promoting consensus among workers, this thesis proposes a novel concept of a *diversity-driven approach*. We show that leveraging diversity in workers' responses is effective in improving the accuracy of aggregate annotations because it compensates for biases or uncertainty caused by the system, tool, or the data. We then present techniques that elicit the diversity in workers' responses. These techniques are orthogonal to other quality control methods, such as filtering, training or incentives, which means they can be used in combination with existing methods. The crowd-powered intelligent systems presented in this thesis are evaluated through visual perception tasks in order to demonstrate the effectiveness of our proposed approach. The advantage of our approach is an improvement in collective quality even in settings where worker skill may vary widely, potentially lowering barriers to entry for novice workers and making it easier for requesters to find workers who can make productive contributions. This thesis demonstrates that crowd workers' input diversity can be a useful property, yielding better aggregate performance than homogeneous input.

Thesis statement: *By eliciting and leveraging the diversity of crowd workers' responses, it is possible to systematically reconstruct higher quality annotations.*

CHAPTER I

Introduction

Intelligent systems powered by crowdsourced human computation that is, systems that harness human intelligence as part of an algorithmic process via an open call can perform difficult cognitive and/or creative tasks that cannot easily be done by either computers or people alone (*Kittur et al.*, 2013; *Quinn and Bederson*, 2011; *Gordon et al.*, 2015; *Lasecki et al.*, 2012; *Bigham et al.*, 2010). Crowdsourcing platforms, such as Amazon Mechanical Turk, have made access to, typically quasi-anonymous (*Lease et al.*, 2013; *Gray and Suri*, 2019), crowds of workers quicker and easier than ever before, allowing researchers and developers to easily post tasks (*Bell et al.*, 2013; *Bernstein et al.*, 2010; *Lasecki et al.*, 2012) that require human computation. However, the quality of crowd responses depends on multifaceted factors, such as workers' experience, interests, attention, and skill level, as well as a task's workflow, interface design, incentive mechanisms, and platform quality (*Lasecki et al.*, 2014b; *Kim et al.*, 2018). In this thesis, we explore how our novel strategies in leveraging *diversity-driven approaches* can improve the collective quality of crowd responses even in settings where the worker skill varies widely.

While the benefit of leveraging diverse people in workplaces, communities, and societies has long been of interest for both researchers and practitioners (*Surowiecki*, 2005; *Page*, 2008; *Yu et al.*, 2016), how to design tasks to *systematically* elicit and

leverage *input diversity* to improve the quality of aggregated task performance has been under-explored. In this thesis, we define input diversity as follows and explore how to design crowd-powered intelligent systems with input diversity in mind:

Input Diversity: *The extent to which the error distributions differ within a set of responses to a system.*

This research explores three hybrid intelligence crowd-machine approaches (*Lasecki*, 2019; *Song et al.*, 2018, 2019a,b, 2020) that elicit and leverage diverse responses from crowd workers in a way to improve the accuracy of reconstructed annotations. The idea of designing crowdsourcing tasks by eliciting and leveraging input diversity to improve aggregate answer quality is different from prior microtask crowdsourcing approaches which were focused on *reducing* the diversity of the crowd's responses to find a single, most agreed upon answer.

One of the advantages of this paradigm is that the approaches can improve response quality regardless of workers' skill level, which benefits work access by lowering barriers to entry for novice workers. Moreover, the approaches are complementary to other quality control methods, such as filtering, training, or incentives, and may be used together to improve performance even further.

In this thesis, we describe three approaches that demonstrate the effectiveness of the proposed crowdsourcing paradigm which improves answer quality through leveraging input diversity (Figure 4.1). In Chapter III, we first introduce an approach that leverages multiple different *tools* for the same task, which enables the induced diverse answers from crowd workers to mitigate systematic error biases (*Song et al.*, 2019a, 2018). This strategy not only proposes a method to overcome accumulated systematic error biases, but also introduces a novel concept of tool decomposition, which fills in the gap where traditional task decomposition approaches leave off: cases where the task can no longer be divided into smaller subtasks to improve aggregate

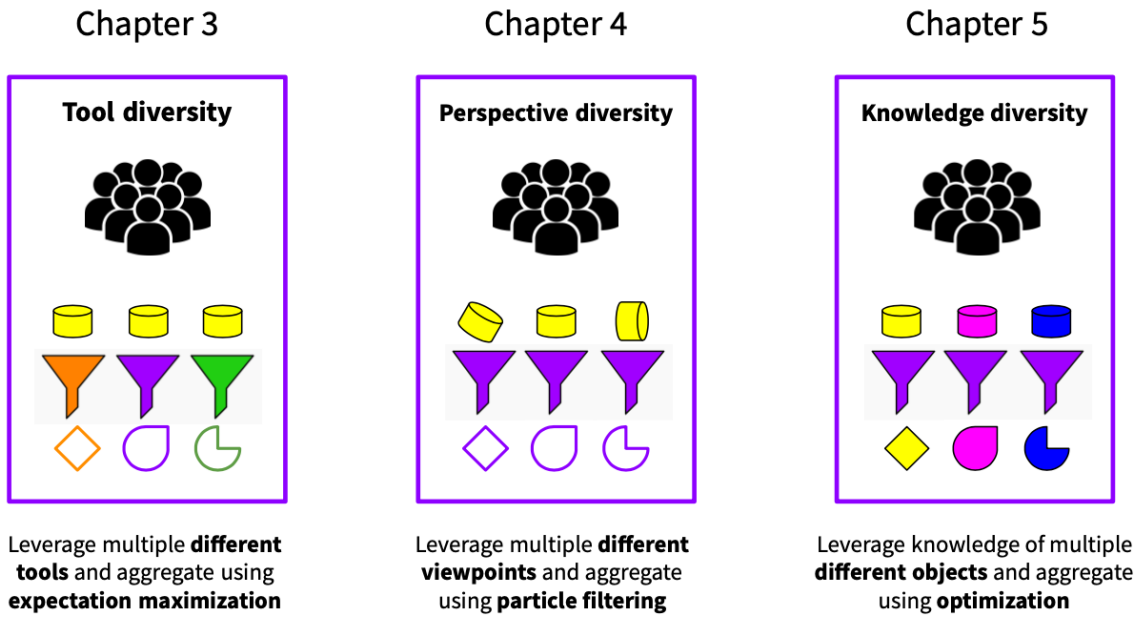


Figure 1.1: In this thesis, we introduce three different approaches to demonstrate the effectiveness and efficiency of leveraging input diversity in crowdsourcing tasks to power intelligent systems. The cylinders represent the target to be annotated, the funnels represent the tool or interface, and the geometrical figures represent the output responses. We show that systematically eliciting and leveraging diverse responses from the crowd workers can improve the accuracy of reconstructed annotations.

worker performance. In Chapter IV, we introduce an approach that aggregates crowd workers' annotations from multiple different *perspective* (video frames) of the same target object as a means to overcome erroneous or even missing annotations (*Song et al.*, 2019b). As a result, the system output improves the accuracy of 3D state estimation of objects in videos. We show that diverse annotations can complement the aggregation and reconstruction quality if the relationships between the annotations are identified, e.g., the temporal dependency between annotations. Lastly, in Chapter V, we introduce an approach that leverages *knowledge* diversity of crowd workers as a means to enable people with diverse knowledge, skills, and perspectives to contribute to a task regardless of their level of expertise (*Song et al.*, 2020). We propose a method to transform approximate crowd answers into soft constraints for

the system, narrowing down the search region in a solution space to increase the chance of finding a better solution even with the collection of only rough estimates.

1.1 Research Questions

To explore the thesis of this dissertation, we address three research questions.

- Research Question 1: How can we leverage tool diversity in annotation tasks so that the aggregated answer is more accurate than any individual tool alone would have been?
- Research Question 2: How can we leverage perspective diversity so that the lack of information from one instance can be supplemented by information from other related instances?
- Research Question 3: How can we leverage knowledge diversity of crowd workers so that each worker can contribute according to their capability and knowledge relevant to the task?

1.2 Dissertation Outline and Contributions

Chapter II begins by covering major research challenges addressed by the prior work in this space, primarily focusing on research in aggregation and reconstruction of crowdsourced microtasks.

The main part of the thesis introduces three crowdsourcing approaches that leverage input diversity of the crowd to generate more accurate reconstruction of annotations. These chapters span tool design, technical solutions, data analysis, and performance evaluation of crowd-machine hybrid tools.

In the remaining chapters, this dissertation will make the following contributions:

- Research Question 1: Chapter III introduces the concept of *tool decomposition* to augment the standard crowdsourcing practice of task decomposition. The effectiveness of the concept is demonstrated with a crowd-powered image segmentation task (*Song et al.*, 2018, 2019a).

- Research Question 2: Chapter IV introduces an approach to leveraging temporal dependency between video frames to fill in missing information from a single frame. The effectiveness and efficiency of the approach is demonstrated with a crowd-powered object 3D state estimation task (*Song et al.*, 2019b).

- Research Question 3: Chapter V introduces an approach to leveraging multiple granularities of answers from crowd workers to enable crowd workers to contribute to a task regardless of the specificity of their knowledge. The effectiveness and efficiency of the approach is demonstrated with a crowd-powered object 3D location estimation task (*Song et al.*, 2020).

By introducing the advantages of diversity-driven approaches in microtask crowdsourcing, this dissertation opens a new thread of research that elicits and leverages diverse answers from the crowd to benefit both requesters and crowd workers in contributing to a system. We offer a generalizable means of reconstructing high quality annotations from crowds' responses, which is demonstrated and evaluated with crowd-power hybrid intelligence systems solving challenging crowdsourcing problems.

CHAPTER II

Background

This dissertation builds upon four main areas of research: (i) designing crowdsourcing workflows to improve the collaborative performance of crowds and machines; (ii) quality control in crowdsourcing to improve the quality of output data; (iii) eliciting diverse responses from the crowd; and (iv) crowdsourcing visual annotations to enable building intelligent crowd-powered systems that can process complex visual information. This chapter reviews literature in these three domains.

2.1 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 (*Lease et al.*, 2013)) crowd workers can accept and complete. TurKit (*Little et al.*, 2010) introduced the crash-and-rerun programming model to recursively improve output of a challenging task by passing the task from worker to worker. Soylent (*Bernstein et al.*, 2010) showed that dividing a larger task into Find-Fix-Verify steps improves the accuracy of crowd workers' answers in document editing tasks. Similarly, ToolScape (*Kim et al.*, 2014) used a Find-Verify-

Expand workflow to enhance the process of extracting different steps in how-to videos. ConceptScape (*Liu et al.*, 2018) extends multi-stage workflows and divides the concept map generation task into three stages with multiple substeps within each stage. CrowdForge (*Kittur et al.*, 2011) introduced a MapReduce-like workflow to accomplish even complex and interdependent tasks using microtasks. Crowdlines (*Luther et al.*, 2015) introduced two different workflows for merging information from multiple sources to create an outline. Turkomatic (*Kulkarni et al.*, 2012) attempted to crowdsource the workflow itself, showing that the planning and execution of a task can be done given some level of requester supervision.

While this prior research has explored how to use crowd workflows to collectively accomplish what no single worker could alone, each task type was done using the same UI, and thus was subject to systematic error biases in each tool. More recently, continuous crowdsourcing has made real-time (*Lasecki et al.*, 2011, 2012; *Salisbury et al.*, 2015a,b) or even instantaneous (*Lundgard et al.*, 2018) crowdsourcing responses from crowds possible. These allow for the creation of interactive systems powered by human contributors (*Lasecki et al.*, 2014b). TimeWarp (*Lasecki et al.*, 2013a) introduced the idea of creating workflows that enable a group of workers to complete tasks in a manner that was not possible with a single worker. In the work, crowd workers were able to provide captions in real time while listening to half-speed audio to improve aggregate performance even though it is a challenging task for individuals. Plexiglass (*Rao et al.*, 2018) introduced a workflow that enables a single worker to interleave multiple tasks at a time. The idea is to multiplex "passive" and "active" tasks together in one UI to more efficiently complete work that would otherwise contain time spent idly waiting for a rare event to occur. CrowdMask (*Kaur et al.*, 2017) uses a pyramid workflow to mask private content in images using crowds. Their method segments and distributes the segments of user content so that workers can mark potentially private content without viewing enough of it to be harmful. WearMail (*Swaminathan*

et al., 2017) introduced a privacy-preserving workflow that allowed crowds to train a system on demand to algorithmically direct to an email search task without ever revealing the email contents to workers.

This thesis contributes to this line of research by introducing a novel approach that aggregates multiple crowd-powered tools to offer better performance than any of the constituent tools alone (Chapter III). AgentHunt (*Lin et al.*, 2012a) had a similar motivation when using multiple workflows to outperform a single best workflow, but their approach used decision-making models to *choose* among different workflows.

2.2 Quality Control in Crowdsourcing

Quality control is a challenging problem in crowdsourcing due to the fact that a crowd is typically composed of people with unknown and diverse skills, abilities, technological resources, and levels of understanding of the given task. There are two primary classes of methods for improving the quality of crowdsourced annotations: methods for post-hoc compensation for low quality work at the time of aggregation, and methods for preventing low quality work at the time of annotation collection (*Kittur et al.*, 2013; *Rzeszotarski and Kittur*, 2011). Post-hoc compensation for low quality work, such as majority voting on the result or weighting workers based on expectation maximization (*Dawid and Skene*, 1979; *Ipeirotis et al.*, 2010), is done after results have been submitted, usually in the aggregation stage. Powerful post-hoc techniques can complement poor quality results in crowdsourced datasets by leveraging the agreement between multiple workers. However, because answers exist in a large continuous space, agreement may not be possible in generative tasks. In this thesis, we introduce novel answer aggregation techniques, which leverage the shared relationship between the tasks or the data instances to combine heterogeneous annotations. The aggregation leads to improved annotation accuracy compared to aggregating a homogeneous set of input from the crowd.

2.2.1 Answer Aggregation in Crowdsourcing

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 (*Snow et al.*, 2008) and document editing tasks (*Bernstein et al.*, 2010). 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 (*Gurari et al.*, 2016; *Lin et al.*, 2014). More sophisticated approaches using unsupervised learning have been used to weight workers' answers by using models of their abilities (*Bragg et al.*, 2013; *Lin et al.*, 2012b; *Welinder et al.*, 2010; *Whitehill et al.*, 2009). Deluge (*Bragg et al.*, 2013) models workers' sensitivity and specificity to detect noisy workers, and LazySusan (*Lin et al.*, 2012b) tracks workers by assigning different weights based on the accuracy of a worker's answers. Researchers have also proposed probabilistic approaches to model not only the workers, but also the properties of the data being labeled (*Welinder et al.*, 2010; *Whitehill et al.*, 2009).

2.2.2 Pre-Filtering Answers in Crowdsourcing

Another strategy to improve the quality of aggregated answers is to prevent low quality work before it is submitted, e.g., training workers (*Gadiraju et al.*, 2015), screening workers (*Kamar et al.*, 2012), or applying different incentive strategies (*Mao et al.*, 2013). Recently, skip-based annotation techniques (*Chang et al.*, 2017; *Shah and Zhou*, 2015) have been explored in the labeling domain, which allow crowd workers to self-filter their labels based on their confidence about a question. Revolt (*Chang et al.*, 2017) introduced a collaborative crowdsourcing system that post-processes self-filtered questions and asks workers to discuss with each other the question to reach

a consensus. Shah and Zhou (*Shah and Zhou*, 2015) showed that incentivizing crowd workers to self-filter is the only incentive-compatible payment means possible.

2.2.3 Bias Correction in Crowdsourcing

Assigning differential weights to workers' answers during aggregation is a preprocessing step that aims to correct individual worker errors before combining the answers (*Rzeszotarski and Kittur*, 2011). Ipeirotis et al. (*Ipeirotis et al.*, 2010) showed that the EM algorithm can be used to separate biases from unrecoverable errors, providing more reliable scores of the quality of the workers. The EM algorithm (*Dawid and Skene*, 1979; *Ipeirotis et al.*, 2010) predicts unknown (latent) correct answers by estimating weights for each crowd worker's answers. Dawid and Skene (*Dawid and Skene*, 1979) showed that the EM algorithm significantly outperforms majority voting when a majority of workers' responses are correct and conditionally independent given the ground truth answer. The EM algorithm is suitable for exploiting 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 probability of tools labeling a pixel is independent of any particular chosen pixel. When designing a tool, its exact abilities and error biases are not typically known in advance because designers are not unaware of the input images that the system will see in final use. 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.

Several approaches have been introduced to combat biases of individual crowd workers, but there has been little work on correcting error biases induced by tools or interfaces. For example, (*Lasecki and Bigham*, 2012) and (*Kaspar et al.*, 2018) can be potentially used to correct systematic biases induced by workers, but require human mediators to correct biased answers.

2.3 Diverse Responses from the Crowd

Crowdsourcing serves as a powerful method in obtaining human-labeled datasets that leverages the "wisdom of crowds" (*Surowiecki*, 2005). The idea is that the average among diverse different responses tend to be close to the expected solution. While this conventional perspective regards the diversity in crowd responses more or less as noisy signal, recent studies started to look at this as a property to be leveraged.

In paraphrasing tasks, diverse responses are encouraged because novel paraphrases are expected, which can be elicited from priming the annotators with different example paraphrases (*Jiang et al.*, 2017). Similarly, in text summarization tasks, inducing crowd workers to create diverse different aspect-based summarization gave more accurate results than asking them to extract all key elements (*Jiang et al.*, 2018). In entity annotation tasks, it is shown that identifying diverse but valid crowd worker interpretation can improve the interpretability of the dataset (*Kairam and Heer*, 2016). Recent work has shown that crowd workers' diverse perspectives can be effectively leveraged in an emotion annotation task, such that response diversity enables the efficient construction of a large collective answer distribution (*Chung et al.*, 2019a). Another effective diversity elicitation approach has been demonstrated in crowd-powered GUI testing, where the diverse navigation paths increase the test coverage (*Chen et al.*, 2020).

Other work systematically elicits diverse responses from the crowd to obtain a single aggregated artifact. For example, the idea of decomposing tools for the same task to elicit different tool diversity has been studied in image segmentation tasks (*Song et al.*, 2018, 2019a). A similar idea of dynamically switching workflows for the same task has been demonstrated to be effective in generating diverse but valid data for NLP training (*Lin et al.*, 2012c). While these works focused on leveraging diverse responses to reduce error biases induced by the tools or the systems, there are other works that leverage diverse responses to compensate the uncertainty of data with rich

information (*Song et al.*, 2019b; *Chung et al.*, 2019a).

The common thread behind these research efforts is that they leverage diverse responses to increase the collective information, which can reduce aggregate noise or compensate biases when combined appropriately. This thesis contributes to this line of work, while we introduce novel strategies in designing crowdsourcing tasks with diversity in mind.

2.4 Crowdsourcing Visual Annotations

In this thesis, we evaluate our proposed approaches on visual perception tasks in order to demonstrate the effectiveness and efficiency of the proposed approaches. Therefore, in this section, we provide a brief background on crowdsourcing visual annotations on images and videos.

Despite the great progress in computer vision on problems such as object category detection and object 2D bounding box detection, many tasks still remain challenging including estimating the 3D properties of objects from a single RGB image (*Hoiem et al.*, 2005; *Saxena et al.*, 2007; *Eigen and Fergus*, 2014; *Tulsiani et al.*, 2017). This thesis demonstrates the effectiveness of the proposed diversity-driven approach using challenging image and video annotation tasks.

2.4.1 Crowdsourcing Image Annotations

Crowdsourcing techniques are widely used in visual data annotation in the 2D image-space, such as object segmentation (*Bell et al.*, 2013; *Lin et al.*, 2014; *Song et al.*, 2018; *Vernier et al.*, 2019), object detection (*Hara et al.*, 2014; *Sorokin et al.*, 2010), and visual question answering (*Bigham et al.*, 2010; *Krishna et al.*, 2017). There are many emerging web-based tools designed to assist crowd workers in image annotation tasks to make use of their efficiency and flexibility. Much of the recent success in automated computer vision has been driven by novel large-scale datasets,

and by powerful neural network models that can learn from this data. The large datasets (*Deng et al.*, 2009; *Lin et al.*, 2014; *Bell et al.*, 2013; *Bigham et al.*, 2010) are made possible by the emergence of crowdsourcing platforms, such as Amazon Mechanical Turk, which allows to recruit crowd workers to perform image annotations. Rapid crowdsourcing of image annotations can also enable in-home robots to interact with never-before-seen objects on a daily basis (*Gouravajhala et al.*, 2018).

2.4.2 Crowdsourcing Video Annotations

The techniques used for static image annotation do not optimally extend to video annotation, as they neglect dependencies in the temporal dimension. This imposes significant additional cost on the task and prevents scaling. Our work focuses on decreasing the cost of collecting annotations by increasing the aggregation efficiency.

Video annotation systems for some tasks (e.g., activity summarization (*Lasecki et al.*, 2013b) or event summarization (*Yuen et al.*, 2009)) provide a video stream to crowd workers and ask them to provide a summary of the clip based on the query from a requester. Other systems are designed to detect targeted events from a video stream (*Bernstein et al.*, 2011; *Kim et al.*, 2014; *Lasecki et al.*, 2014a; *Park et al.*, 2012), letting crowd workers refer to the temporal context to decide the specific moment of the targeted events. Some systems are built for more confined local annotation tasks, such as moving object detection (*Di Salvo et al.*, 2013), object tracking (*Vondrick et al.*, 2013), object annotation (*Yuen et al.*, 2009), or object segmentation (*Kaspar et al.*, 2018). Our work in Chapter IV contributes to this line of research by introducing a novel method to aggregate confined local annotations across video frames to improve the output quality of subsequent video processing steps. More specifically, we introduce a system that estimates the 3D state—position and orientation—of objects (*Su et al.*, 2015; *Szeto and Corso*, 2017) using novel collection and aggregation strategies.

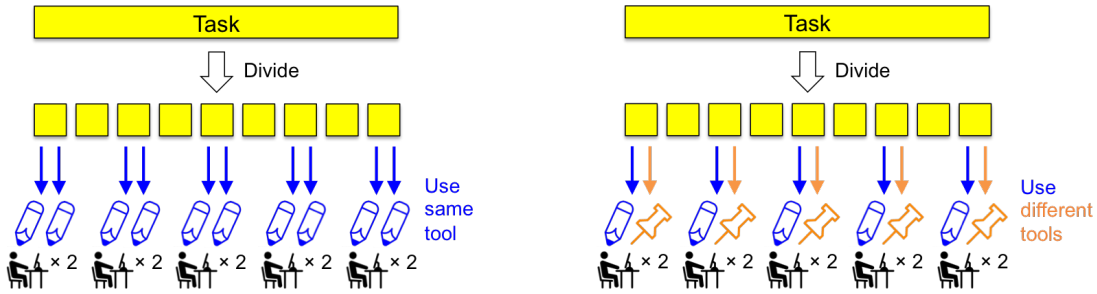
CHAPTER III

Tool Diversity as a Means of Improving Aggregate Crowd Performance

3.1 Motivation

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.

While perceiving the boundaries of physical objects comes naturally for people, it remains a challenging open problem for CV systems due to the complexity of understanding the semantics of visual scenes (*He et al.*, 2017; *Long et al.*, 2015). Crowd-powered object segmentation tools can bridge this gap by leveraging human understanding to produce large, manually-demarcated training data sets (e.g., (*Bell et al.*, 2013; *Gurari et al.*, 2016; *Lin et al.*, 2014; *Vijayanarasimhan and Grauman*, 2014)) for CV systems. However, designing crowd-powered tools that produce high-accuracy training data and scale efficiently (with respect to human-time cost) remains



(a) Conventionally, one tool type is used for the same task.

(b) We propose leveraging different tools for the same task.

Figure 3.1: This chapter introduces an approach to leveraging *tool diversity* that uses multiple different tools for the same task (as in (b)) to improve aggregate crowd performance by offsetting systematic error biases that might otherwise result from using any one tool type alone (as in (a)). Our findings on an image segmentation task demonstrate that using a combination of tools can significantly increase aggregate accuracy.

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., (*Bearman et al.*, 2016; *Bell et al.*, 2013; *Carlier et al.*, 2014; *Gouravajhala et al.*, 2017; *Lin et al.*, 2016; *Russell et al.*, 2008)) 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 patterns in worker performance, which can lead to repeated systematic mistakes when only a single tool is used. For example, some tools (*Bell et al.*, 2013; *Russell et al.*, 2008) provide polygon drawing functionality to help trace object boundaries, but Bell et al. (*Bell et al.*, 2013) reported that workers often skip selecting parts 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 (*Carlier et al.*, 2014) (which asks workers to use left/right mouse clicks to identify foreground and background regions of an

image) to result in the same mistakes as 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 (*Meissner and Brigham*, 2001), in that they are common to the outputs of a tool, not common to the annotations produced by an individual worker.

In this chapter, we propose the idea of leveraging *tool diversity* as a means of overcoming these systematic error biases to improve 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 a higher *combined* accuracy (Figure 3.1). 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 (*Dietterich et al.*, 2000). A strength of leveraging tool diversity is that the approach is orthogonal to, and thus may be combined with, many existing crowdsourcing methods for improving quality over time (e.g., training workers (*Dow et al.*, 2012; *Williams et al.*, 2016) or identifying high-performing contributors (*Rzeszotarski and Kittur*, 2011)). We also note that a similar concept was introduced in the TISM method *Griffin and Corso* (2019) where multiple different computer-generated annotations on the same object in the same image are automatically combined to produce consistent improvement in performance.

To demonstrate our proposed workflow, we design four different image segmen-

tation tools and introduce FourEyes, a multi-tool based crowd-powered system that leverages combinations of tools to generate better aggregate responses. After that, we report results from a series of studies that evaluate different aggregation conditions—such as majority voting versus expectation maximization (EM), and single-tool aggregation versus up to four-tool combination aggregation—with equally-sized groups of workers. Our evaluation demonstrates the effectiveness of tool diversity by showing that the output accuracy of heterogeneous tool combinations can be significantly higher than that of homogeneous sets, providing output at least comparable to the best constituent tool, and always yielding significantly better results than the weakest constituent tool.

Moreover, we explore the idea of adding post processing for multi-tool aggregation with respect to the error correction mechanism. When leveraging tool diversity, once the analysis on individual tool performance is conducted and the error pattern of each tool is revealed, a system designer can implement suitable correction mechanisms to further offset error biases To correct for errors in image segmentation tasks, we introduce a new region-based method for synthesizing more accurate bounds through averaging surrounding annotations. We explore the effects of mask size and threshold parameter, and show that the proposed method always increases the aggregate accuracy of any tool combination by up to 6%. We also investigate the effect of a threshold parameter in the EM method, and show that the threshold parameter value which yields the best performance differs by tool combination types.

Finally, we discuss generalizable guidelines for applying the multi-tool approach in other problem domains. We characterize our problem in a more general form and summarize the properties of crowdsourcing tasks that are amenable to our approach: those that are objective, tractable enough for workers to produce nearly-correct responses, and increase in correctness as additional answers are provided, can benefit from our approach.

This chapter presents an extended version of work published at the 2018 ACM International Conference on Intelligent User Interfaces (*Song et al.*, 2018) that first introduced the idea of leveraging tool diversity during aggregation as a crowdsourcing technique. In addition to a more in-depth evaluation of tool combinations (aggregation of three- and four-tool combinations), this chapter introduces a novel region-based error correction method and explores the impact of parameter selection on the region-based and EM methods as a means of pre-processing for multi-tool aggregation.

The key contributions of this chapter are:

- A novel crowdsourcing paradigm that leverages a system's or task's tool diversity in order to aggregate input across different *types* of tools to improve the combined accuracy of workers' answers by offsetting systematic error biases.
- 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.
- An evaluation of the aggregate results of each possible tool combination from FourEyes, and an exploration of the ability for correction mechanisms to further improve the accuracy of the combined results by exploiting the error bias patterns of the individual tools.

3.2 Approach

Conventional approaches to improving crowd worker output accuracy include microtask decomposition and consensus-based aggregation. These approaches are usu-

ally intended to reduce task complexity and correct for the 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, introducing biases into worker responses. Our tool diversity strategy builds on prior work in crowdsourcing workflows and answer aggregation strategies to reduce these systematic error biases.

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.

3.2.1 Motivation from Ensemble Learning

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 (*Dietterich et al.*, 2000; *Freund and Schapire*, 1995). It has been proven that ensembles often perform better than any single member (*Dietterich et al.*, 2000). 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 (*Hansen and Salamon*, 1990). The left diagram in Figure 3.2 shows how ensemble methods work. In the diagram, a learning algorithm can be viewed as searching a space of hypotheses

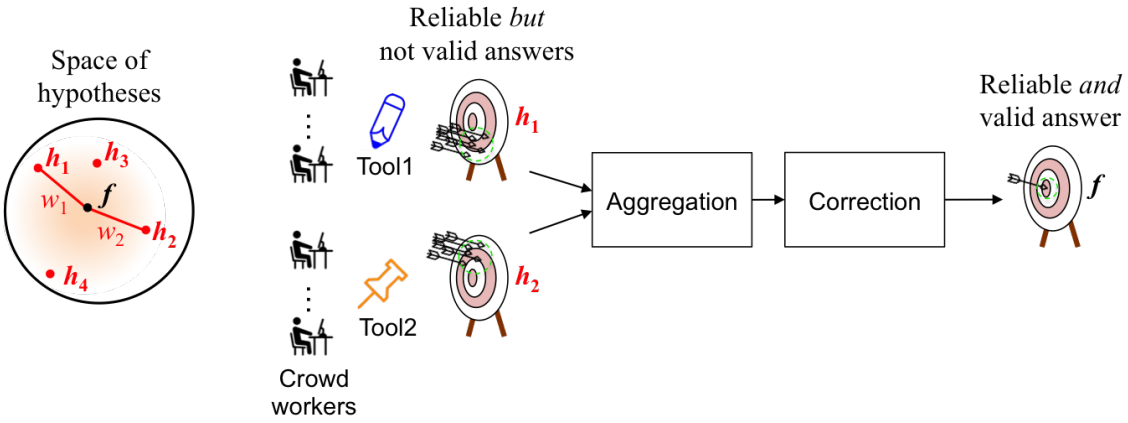


Figure 3.2: The left diagram shows the hypotheses space of the possible segmentation tools, including the best performing tool (f) and other possible hypotheses ($h_1 \ldots h_4$). We are motivated by ensemble learning methods that construct a combination of alternative hypothesis (h_1 and h_2) to approximate the best hypothesis f. The right flowchart shows a set of workers using two different tools to perform the same task. An aggregation and correction pipeline can output reliable (consistent) and valid (accurate) aggregate results (f) from two reliable but not valid answers (h_1 and h_2). This diagram represents the end-to-end process of the proposed tool diversity scheme: preparing different tools, aggregating, and correcting.

to identify the best performing hypothesis f, which 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 multiple image segmentation tools are combined to produce a better final result.

3.2.2 Aggregation of Reliable but Biased Tools

Even a carefully designed crowdsourcing system may often induce reliable (consistent) but not valid (accurate) answers. For example, a semantic image classification task of assigning classes that correspond to objects that appear in an image can have its systematic bias due to the design of the tool. If a tool is designed to type free-form answers, it may bias workers to only use a limited number of words that

they can spell or find easier to spell. On the other hand, if a tool is designed such that workers can click to select a word from a predefined list, the error pattern would be different. These errors can be defined as systematic error biases because the same error pattern would be unlikely to arise if the tools were designed differently.

Instead of trying to fix the bias of a specific tool, our approach aims to *combine* answers from these multiple biased tools to improve the aggregate result. Analogous to a necessary and sufficient condition in the ensemble learning scheme, a suggested condition for using multiple tools is that the tools are at least reliable, even if they are not valid. This allows for aggregation and correction mechanisms that can offset the expected biases, eventually achieving both reliable and valid results when aggregated. Figure 3.2 depicts the concept of tool aggregation within a crowdsourcing workflow. A researcher or requester can provide Tool 1 to one set of workers, and Tool 2 to a different set of workers. When the tools are reliable but not valid with output hypotheses h_1 and h_2, respectively, the aggregation and correction modules can combine the answers so that the final output is approximately f, the best hypothesis. In the next sections, we show how we realized these tools and designed aggregation and the correction mechanisms in the domain of semantic image segmentation.

3.3 FourEyes

FourEyes is an image segmentation system that leverages four different crowd-powered tools to produce accurate segmentation results by aggregating answers across different tool types. We describe the individual tool here, and then detail the novel aggregation methods in the next sections.

3.3.1 Choosing the Tools

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. We designed these tools to ideally perform differently for different types of objects, resulting in greater error diversity. We categorized these object into three groups and created tools that are designed to minimize errors in each object category. The spaces we explored and the tools we designed are summarized in Figure 3.3. We used the Question (Q), Option (O), and Criteria (C) representation (*MacLean et al.*, 1991) of the design space for deciding which tools to build. The Question indicates a key design issue, the 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 3.3), 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 following section, we provide detailed descriptions of the four tools developed.

3.3.2 Designing the Tools

The four tools implemented were Basic Trace, Drag-and-Drop, Pin-Placing, and Floodfill. They vary in the level of degree of freedom, interface layout, and amount of interaction needed from a worker. The differences are summarized in Table 3.1.

Basic Trace

The first tool is a free-form drawing tool shown in Figure 3.4(a). With Basic Trace, workers click and drag their mouse to trace the outline of the query object in a

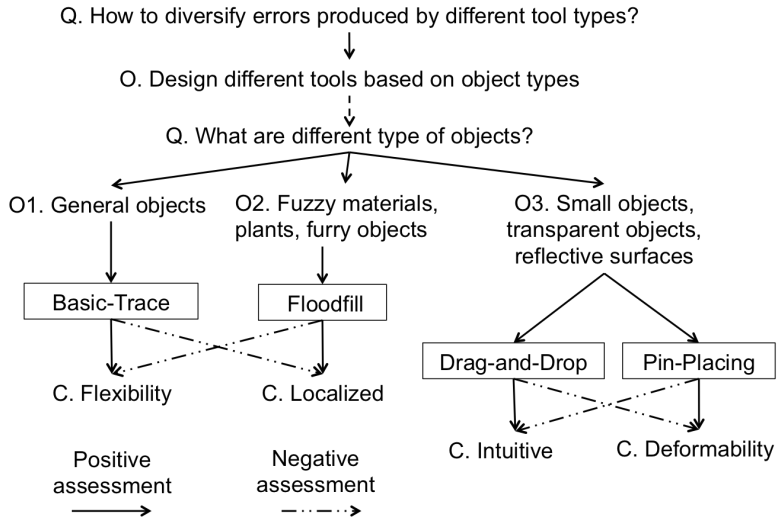
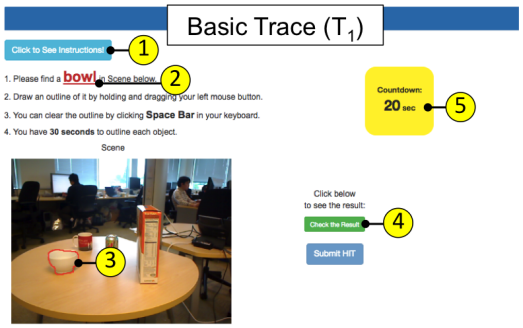


Figure 3.3: Design space we considered when choosing the tools for the study. We used the Question (Q), Option (O), and Criteria (C) representation of the design space.

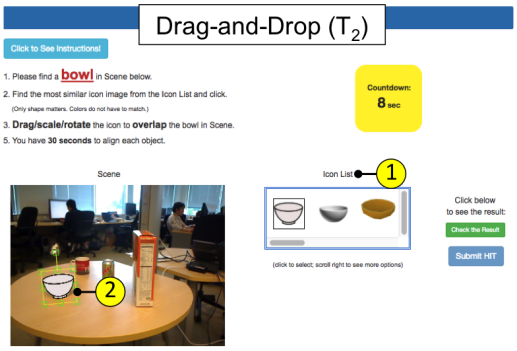
Design Element	Comparison
Degree of freedom & Amount of interaction	Basic Trace > Pin-Placing > Drag-and-Drop > Floodfill
Complexity of interface layout	Pin-Placing > Drag-and-Drop > Floodfill > Basic Trace

Table 3.1: A comparison of the four tools across two design elements.

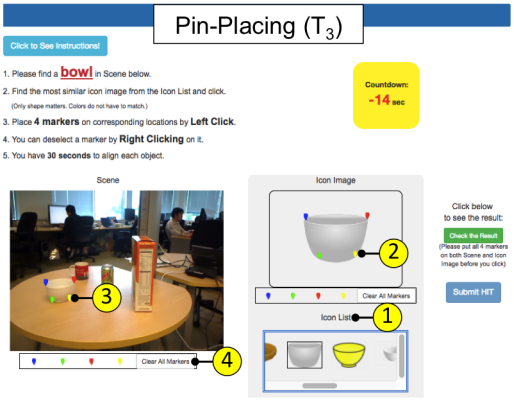
scene (Figure 3.4(a) ③). Once a worker submits the initial trace line, a simple image processing algorithm connects the gaps and fixes the irregularities in the traced line in order to form a smooth shape. It then highlights the pixels inside the traced shape, and returns the result as the final object segmentation. Of our four tools, the Basic Trace is the most manual and provides the highest degree of control. 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 boundary.



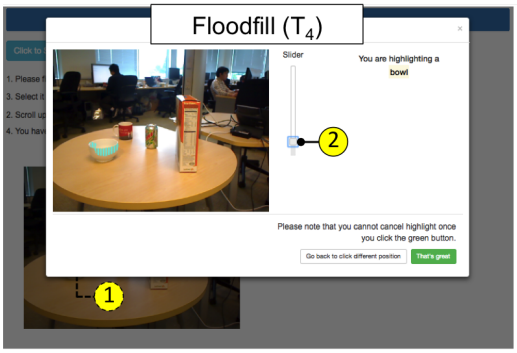
(a) Overview of the Basic Trace image segmentation tool. (1) shows the full instructions when clicked. (2) describes the query object which is shown to the workers in random order. (3) shows an example of a trace line a worker provided. (4) shows the segmentation result when clicked. (5) is the task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



(b) 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.



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



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

Figure 3.4: Worker interface of the four segmentation tools used in our experiments.

Drag-and-Drop

The second tool lets workers select an object template from a list (Figure 3.4(b) 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 chapter 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 3.4(b) ②). Workers are able to scale, rotate, and drag the template to adjust the angle its dimensions in an attempt to closely match the shape of the actual object. Based on the transformation of the template, the system determines the final object segmentation by identifying 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 rigid objects if being viewed from different angle.

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 3.4(c) ①). With this tool, workers select four arbitrary points on their selected object template (Figure 3.4(c) ②), and pair them with four corresponding points on the object in the scene (Figure 3.4(c) ③). Workers can modify individual points, or clear all points at once (Figure 3.4(c) 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 not 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 would be allowing workers to switch to a different non-template-based tool. 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 it overlaps with other objects. This is useful in domains like robotics, where ground truth object geometry includes hidden parts. 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, these two 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 (*Torbert*, 2016) with minimal human input to initialize the seed point and tune a threshold parameter. Workers click on the object they want to segment (Figure 3.4(d) ①) and adjust a slider to tune one of the algorithm's threshold parameters (Figure 3.4(d) 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 result 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 effective if the shape of an object is complex with many curves, but only when the object is mostly monochromatic. If a query object is polychromatic or contains shaded regions, the selection area can be smaller than the actual object boundaries because the algorithm cannot propagate across these regions.

3.3.3 System Interfaces

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 which are then displayed to workers (Figure 3.4(a) ②) as objects that need to be segmented from the scene. For each tool, a short series of instructions (including the target object in bold) is displayed to workers while they perform the task. Workers can also check the segmentation result before they submit their work ("Check the Result" button in Figure 3.4(a) ④). To discourage workers from idling, a task timer is embedded (Figure 3.4(a) ⑤) that counts down from t seconds and turns negative when time runs out. In this chapter, we used $t = 30$ in all experimental conditions. The timer serves as encouragement to complete the task in a timely manner, and does not otherwise affect the workers.

3.4 Measuring the Performance of Individual Tools

To understand the effect of tool diversity on improving aggregate crowd performance, we recruited 288 crowd workers from Mechanical Turk using LegionTools (*Gordon et al.*, 2015). 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 work as a microtask in which one worker segments one object using a single tool. To avoid learning effects and worker-induced bias in the annotation results, workers were randomly assigned to an annotation task (segmenting one scene using one tool), and they could not choose which tool they were given. We recruited six unique workers for each tool-scene pair, resulting in a total of 1224 object segmentations.

3.4.1 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 (*washington*, 2014; *VaFRIC*, 2012), 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.). Each worker was shown one scene and a series of object names to segment depending on the number of objects in the scene. For each task, the order of the 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.

3.4.2 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 lengths of the instructional videos were 36s, 54s, 78s, and 33s, respectively, for each tool: Basic Trace, Drag-and-Drop, Pin-Placing, and Floodfill. Two of the tools (Drag-and-Drop and Pin-Placing) had longer videos because they explained how to choose the most similar template image. This additional step delayed workers' task completion time in the actual experiment as well. Workers were also shown pictures exemplifying desired and undesired segmentations (the same example images were used 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 after watching the instructional video, they were directed to FourEyes's worker UI and their subsequent interactions with the UI were recorded. Task instructions were also accessible at any time if necessary (Figure 3.4(a) ①). Each worker was paid between $0.35 and $0.60 per task, proportional to the number of objects they had to

segment and the expected completion time using 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.

3.4.3 Segmentation Quality Evaluation

To assess success on the image segmentation task, we measured the accuracy of each by comparing the output similarity to the ground truth segmentation that was generated manually by the authors prior to 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 segmentation.

$$\begin{aligned} \text{Precision} &= \frac{\text{true positive}}{(\text{true positive} + \text{false positive})} \\ \text{Recall} &= \frac{\text{true positive}}{(\text{true positive} + \text{false negative})} \qquad \text{(Eq. 1)} \\ \text{F}_1 \text{ Score} &= \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})} \end{aligned}$$

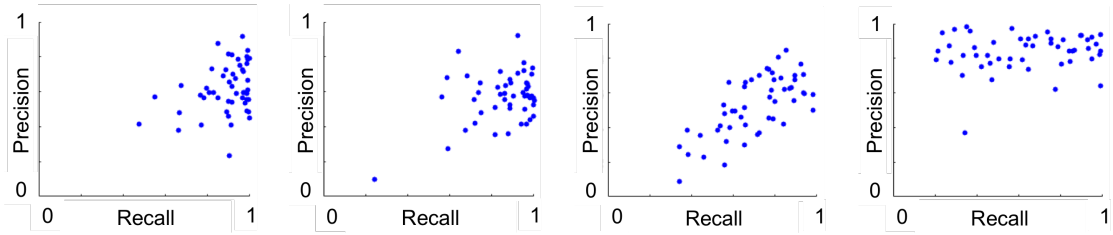
3.4.4 Results

The different tools had different error patterns (trade-offs) in terms of precision and recall. Figure 3.5 shows scatter plots of the overall segmentation result with each dot representing an average precision-recall of one object being segmented using one of the tools in FourEyes. That is, each dot is an average of six workers' segmentation results. As shown in Figure 3.5 (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 3.6 (a) and (b), respectively. Meanwhile, Pin-Placing resulted in the most scattered performance as shown in Figure 3.5 (c). This implies that the performance of the tool varies a lot depending on object types. We presume that the underlying mechanism of computing non-linear transformation of Pin-Placing is unfamiliar to novice workers, which led to scattered and low overall performance. An example of using Pin-Placing is shown in Figure 3.6 (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 3.5 (d) 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 3.6 (d). Because one side of the vase was much brighter, the worker could not select the entire image with the seeded region growing algorithm. Figure 3.6 shows typical example worker segmentations from each tool, alongside the ground truth.

In terms of reliability and validity (as discussed in Section 3.2.2), Basic Trace, Drag-and-Drop, and Floodfill can be considered reliable since their output pattern is expectable (either high recall or high precision). However, they are not valid because their output is biased (either low precision or low recall). On the other hand, Pin-Placing is neither reliable nor valid because the output pattern is not predictable being highly dependent on the query object.

For each tool, we recruited 72 workers. Each worker performed segmentation for one scene, where each scene contained three to seven objects. The cumulative distribution functions of performances (precision, recall, and F_1 score) of a single



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

Figure 3.5: Precision-recall scatter plot of our four different tools. The different tools have different error patterns (trade-offs) in terms of precision-recall metrics. (a) Basic Trace and (b) Drag-and-Drop show high recall but low precision tendency, implying that the tools are reliable but not valid. (c) Pin-Placing shows the most scattered pattern, implying that the tool's performance highly depends on the query object, which makes the tool neither reliable nor valid. (d) Floodfill shows high precision but low recall tendency, implying that the tool is reliable but not valid.

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

worker are summarized in Figure 3.7. From the precision plot (left), we can see that the Floodfill tool has more workers with high scores (> 0.8) compared to the other tools. From the recall plot (center), we can see that the Basic Trace and Drag-and-Drop tools have more workers with high scores compared to the other tools. The F_1 score plot suggests that the tool's harmonic performance is less diverse compared to precision or recall, due to the offset between the two. Average accuracy metrics for each tool are summarized in Table 3.2. We use F_1 score as our performance measurement. In general, Floodfill gave the best performance in terms of precision, and Basic Trace gave the best recall and F_1 score.

The average performances of each object are summarized in Figure 3.8. The hollow dots represent performance for individual objects (average performance of 24

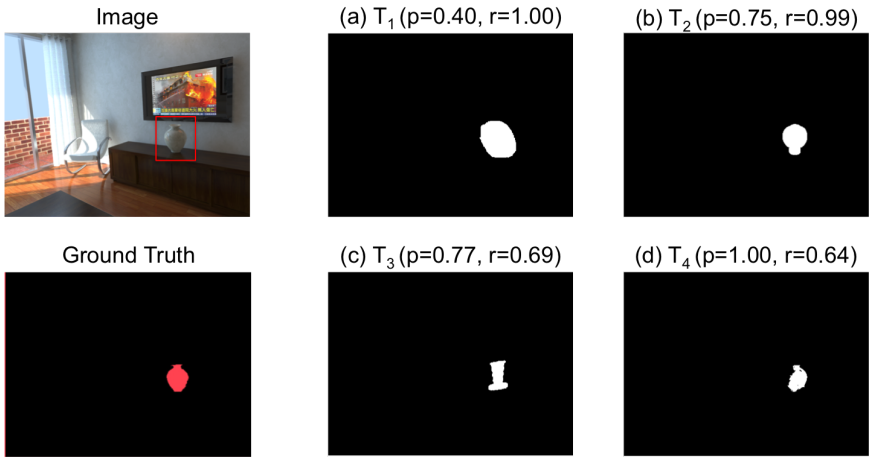


Figure 3.6: Original image (top left), ground truth image (bottom left), and exemplar segmentations using the four tools with their precision and recall values reported on top. (a) Basic Trace, (b) Drag-and-Drop, (c) Pin-Placing, and (d) Floodfill. The exemplar images represent a typical output of each tool.

workers who segmented that object), and the filled dots are average performance over all objects in a single scene. The performance varied across scenes due to the different characteristics of each scene. For example, one scene was shot in front of a window, which added a lot of lighting to the scene, and another scene had many rigid objects that were relatively easy to demarcate from the background. Regardless of the characteristics, the average F_1 score of scenes lay in between 0.5 to 0.8.

To calculate latency, we measured the overall task time starting from the moment 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 for general-purpose crowd workers. Using Floodfill, some workers managed to produce a

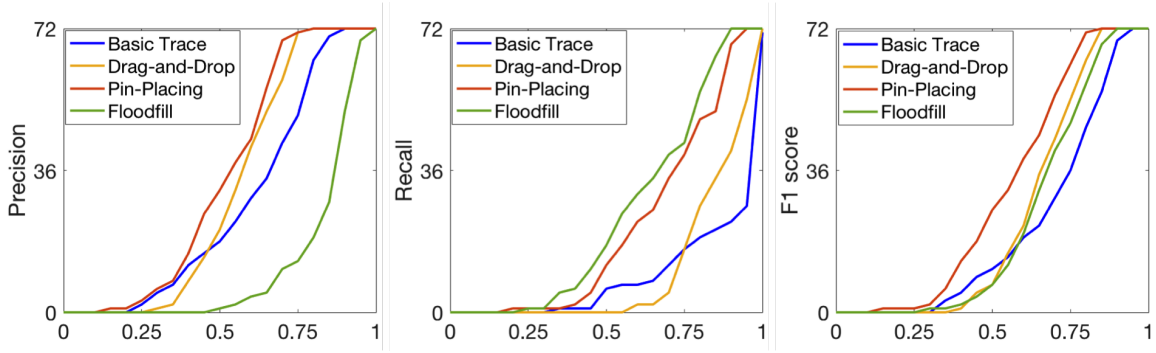


Figure 3.7: Precision (left), recall (center), and F_1 score (right) plots of the cumulative distribution functions of performances of a single worker per tool. In terms of precision, Floodfill has the most number of workers with high performance (> 0.8). In terms of recall, Basic Trace has the most number of workers with high performance. The F_1 score performance per worker is similar between tools compared to precision or recall, because the two offset each other when combined.

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 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, especially when the combined tools are reliable but not valid with complemented biases (as portrayed in Figure 3.2).

3.5 Evaluation of Multi-Tool Aggregation Scheme

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

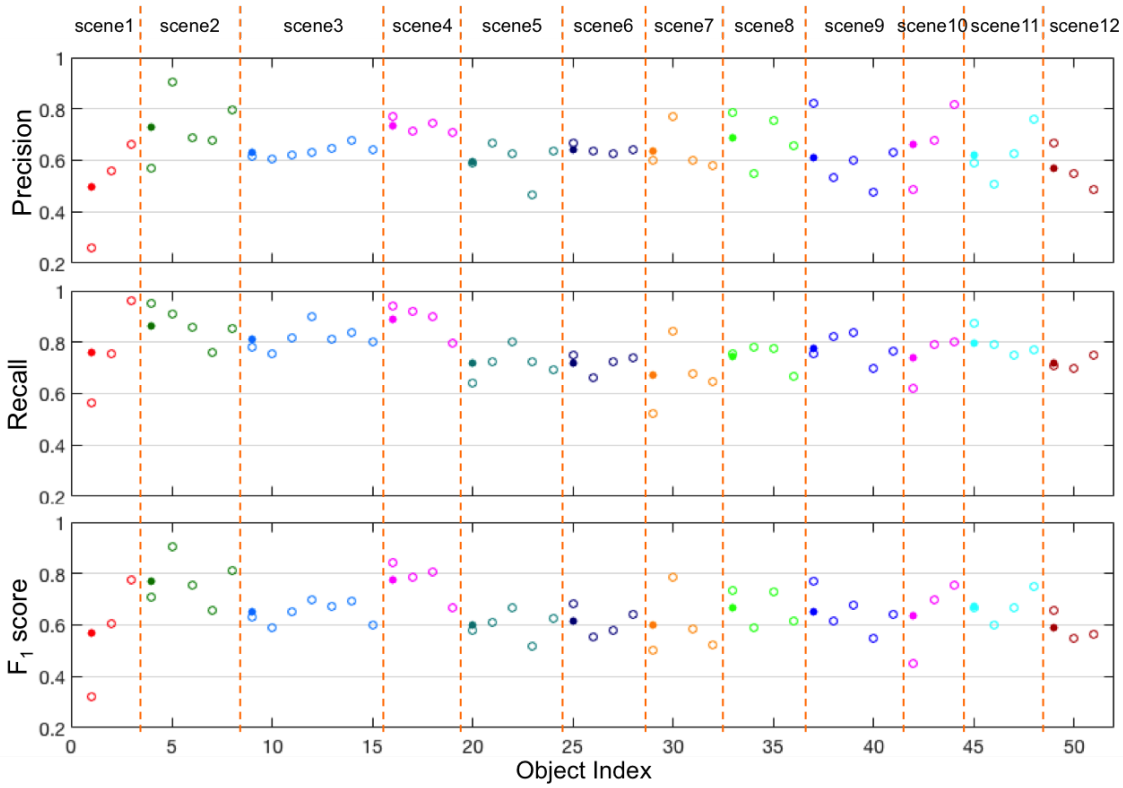


Figure 3.8: Precision (top), recall (middle), and F_1 score (bottom) of average segmentation result of each object and scene. The hollow dots represent performance for individual objects (average performance of 24 workers who segmented that object), and the filled dots are average performance over all objects in a single scene. Different scenes are separated with dotted vertical lines. The average performance of objects varied across different scenes, but lied in between 0.5 to 0.8 in terms of the F_1 score.

of different tools. In the studies, we compared the performance of every possible tool aggregation: from single-tool to four-tool aggregations. 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, three, and four tool combinations aggregating four workers' answers from different tools. In the last study, we apply the EM method on multi-tool aggregation to optimize the tools' weights adaptively per pixel. 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.

3.5.1 Method 1.

Single-Tool Aggregation with Majority Voting (Baseline)

Single-tool aggregation combines answers from the four workers who used the same tool to segment target objects. We randomly picked 15 worker combinations from the collected data. This was performed to avoid any bias from accidentally choosing a good or bad combination 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 (Section 3.4.3). The results of 15 randomly drawn combinations were averaged for all query objects. We summarized the average results in Table 3.3.

	Precision	Recall	F_1 score
Basic Trace	0.61 (0.18)	**0.99** (0.06)	**0.73** (0.15)
Drag-and-Drop	0.57 (0.15)	0.95 (0.11)	0.70 (0.13)
Pin-Placing	0.57 (0.16)	0.83 (0.16)	0.66 (0.16)
Floodfill	**0.85** (0.12)	0.70 (0.23)	**0.73** (0.18)

Table 3.3: Average performance (and standard deviation) of majority voting on single-tool aggregation.

The change in precision was not significant with single-tool aggregation compared to the average precision without aggregation (see Table 3.2). However, recall and F_1 scores improved. For example, recall of Basic Trace increased by 10% ($p < .01$) compared to its average performance without aggregation. 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 multi-tool aggregation.

3.5.2 Method 2. Multi-Tool Aggregation with Majority Voting

Adding multiple tools for the same task can improve the aggregate accuracy when the tools compensate for systematic error biases of each other. In this section, we look at the results of all possible tool combinations aggregated using pixel-level majority voting. We start by focusing on two-tool aggregate performance and then investigate three- and four-tool performance.

Two-Tool Aggregation

There are six possible two-tool aggregations for FourEyes, $\binom{4}{2} = 6$ (4 choose 2). For each tool pair, we randomly picked 15 pairs of workers from each tool, for a total of four workers. As in Method 1, we computed pixel-level majority voting. The average performance of all possible tool pairs is summarized in Table 3.4.

Two-tool aggregation improves F_1 scores compared to single-tool aggregation. Every tool pair except Drag-and-Drop $\times$ Pin-Placing (0.69) 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 0.9% lower F_1 score than aggregating Drag-and-Drop alone (0.70). However, there was no statistically significant difference between single-tool aggregation of Drag-and-Drop versus multi-tool aggregation of Drag-and-Drop $\times$ Pin-Placing pair. We believe this pair did not increase performance because Pin-Placing is a tool that is neither reliable nor valid, with the lowest and scattered performance distribution in terms of precision and recall metrics. One notable finding about the result is that the highest F_1 score achievable from single-tool aggregation is 0.73 (aggregating Floodfill alone), whereas that from multi-tool aggregation is 0.81 (aggregating Basic Trace $\times$ Floodfill pair), which is a 9.8% ($p < .005$) performance improvement with mixing tools. To emphasize the performance

improvement in terms of F_1 score, we compared the F_1 scores of two-tool aggregations (blue bars) with their constituent tools (red and green bars) in Figure 3.10 (a).

Three-Tool Aggregation

There are four possible three-tool aggregations for FourEyes, $\binom{4}{3} = 4$ (4 choose 3). For each tool aggregation, we randomly picked 15 combinations of workers: two from the first tool and one from each of the second and third tools, for a total of four workers. We maintained the same group size with two-tool aggregation to avoid interference from the effect of group size during comparison. The same pixel-level majority voting was conducted. The average performance of all possible tool combinations is summarized in Table 3.5.

Three-tool aggregation also improves F_1 scores compared to single-tool aggregation. Every tool aggregation except Basic Trace $\times$ Drag-and-Drop $\times$ Pin-Placing (0.72) showed increased F_1 scores compared to the single constituent tools. The aggregation gave a better F_1 score than aggregating Drag-and-Drop or Pin-Placing alone ($p < .005$), but there was no significant difference compared to Basic Trace. From the result, we observed that the aggregations that include both Basic Trace and Floodfill showed a significant performance improvement even compared to the superior constituent tools ($p < .05$). We hypothesize that including Floodfill in the tool set significantly improves accuracy because it has the most different error bias compared to the others. That is, the diversity of systematic error biases affects the multi-tool aggregation performance. However, compared to two-tool aggregation, adding a third tool did not improve the performance compared to only combining Basic Trace with Floodfill. This could be because we lost the benefits of within-group aggregation, since only one worker contributed to each of the second and third tool types. We compared the F_1 scores of three-tool aggregations (blue bars) with their constituent tools (red and green bars) in Figure 3.10 (b).

	Precision	Recall	F_1 score
Basic Trace × Drag-and-Drop	0.61 (0.13)	**0.98** (0.03)	0.74 (0.10)
Basic Trace × Pin-Placing	0.62 (0.13)	0.95 (0.08)	0.73 (0.11)
Basic Trace × Floodfill	**0.74** (0.12)	0.94 (0.11)	**0.81** (0.11)
Drag-and-Drop × Pin-Placing	0.57 (0.14)	0.92 (0.11)	0.69 (0.13)
Drag-and-Drop × Floodfill	0.71 (0.11)	0.93 (0.09)	0.79 (0.09)
Pin-Placing × Floodfill	0.69 (0.13)	0.86 (0.14)	0.75 (0.12)

Table 3.4: Average (and stdev) of majority voting on two-tool aggregation.

	Precision	Recall	F_1 score
Basic Trace × Drag-and-Drop × Pin-Placing	0.60 (0.13)	0.96 (0.05)	0.72 (0.11)
Basic Trace × Drag-and-Drop × Floodfill	**0.70** (0.12)	**0.97** (0.04)	**0.80** (0.09)
Basic Trace × Pin-Placing × Floodfill	0.69 (0.12)	0.94 (0.09)	0.78 (0.10)
Drag-and-Drop × Pin-Placing × Floodfill	0.65 (0.13)	0.92 (0.09)	0.75 (0.11)

Table 3.5: Average (and stdev) of majority voting on three-tool aggregation.

	Precision	Recall	F_1 score
All Four Tools	0.65 (0.12)	0.95 (0.07)	0.76 (0.09)

Table 3.6: Average (and stdev) of majority voting on four-tool aggregation.

Four-Tool Aggregation

For the aggregation of four tools, we randomly picked 15 combinations of workers, one from each tool. The average performance is summarized in Table 3.6. The comparison of F_1 scores of four-tool aggregation with that of the constituent tools is summarized in Figure 3.10 (c). Four-tool aggregation improves the F_1 score compared to any of the constituent tools. The four-tool aggregation results give us insight that increasing the number of tools to be combined does not linearly increase the aggregate performance. We hypothesize that the small group size hinders performance more than the benefits from adding more tool types, since having only one worker from one tool type results in a lack of error correction from within-tool aggregation. That is, small groups with more tools do not necessarily improve performance, and to fully benefit from adding more tools, the group size should increase as well.

3.5.3 Method 3. Multi-Tool Aggregation with EM Method

In this section, we model the multi-tool aggregation problem as an optimization problem and use expectation maximization (EM) to estimate consensus-based semantic image segmentations. For certain tool aggregations, EM-based multi-tool aggregation significantly improved output accuracy over majority voting.

We model our problem 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 $\boldsymbol{Z_m}$. 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 $\boldsymbol{Y}$. 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. That is, we assume i.i.d. (independent and and identical distributed) pixels. This assumption is acceptable because we do not have a priori knowledge about the relationship between different pixels, making all pixels have the same chance of being included in a selection. In addition, we denote by $\boldsymbol{\theta}$ 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.

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

The EM algorithm applies an expectation step and a maximization step iteratively:

Expectation Step: Calculate the expected value of log-likelihood, with respect to the conditional distribution of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ under the current estimate of $\boldsymbol{\theta}$.

Maximization Step: Find the estimate $\boldsymbol{\theta}$ that maximizes the expectation of marginal log-likelihood.

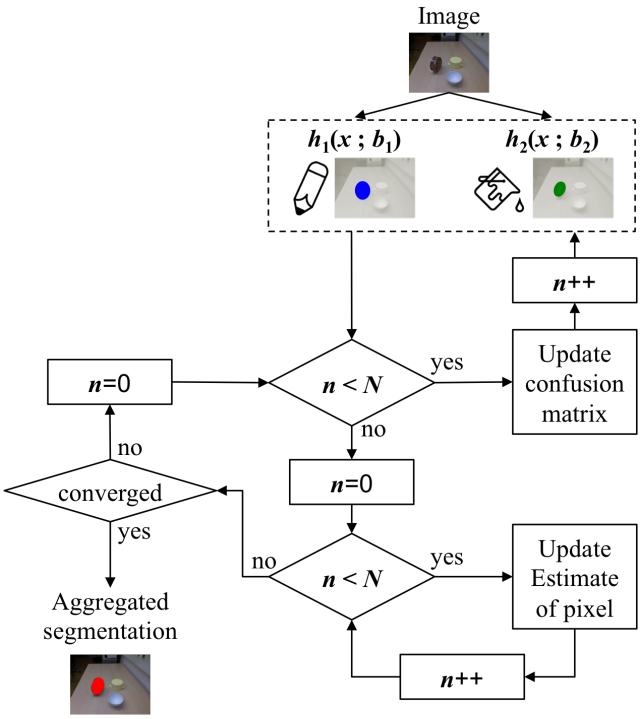


Figure 3.9: The flowchart shows the EM algorithm we adopted for the optimization. Two different segmentation tools, h_1 and h_2, each with different biases, b_1 and b_2 (respectively), pass segmented images to the system. We estimate the weights, w_1 and w_2, to approximate the performance of f.

The Expectation and Maximization steps are repeated until the estimations converge. The diagram in Figure 3.9 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.

	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 3.7: Average (and stdev) of the EM method on two-tool aggregation.

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

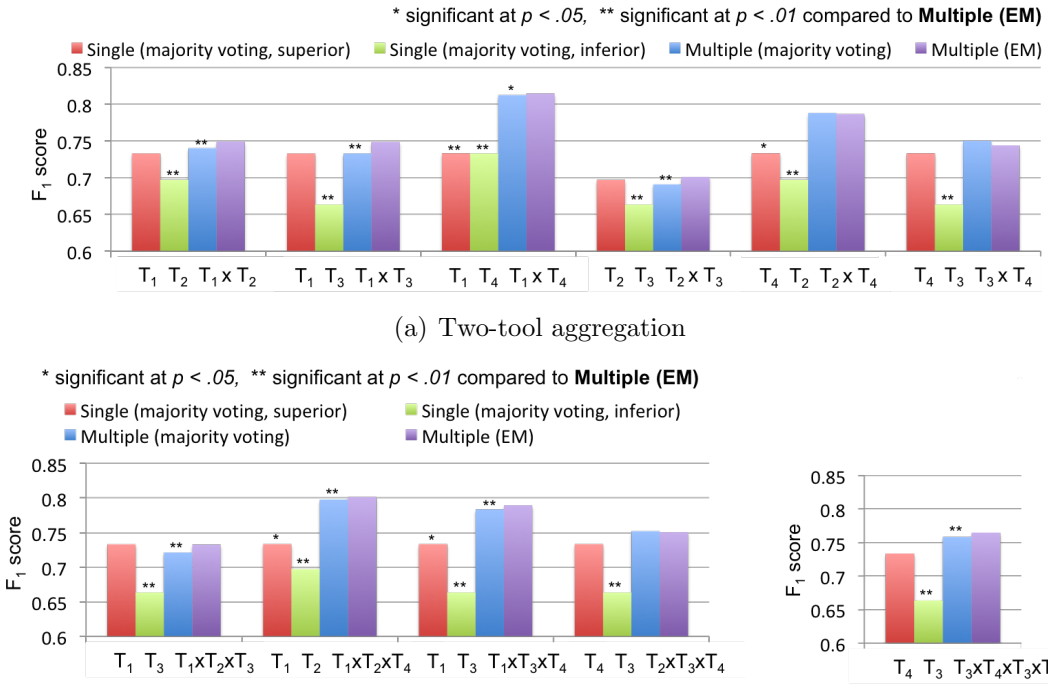
Table 3.8: Average (and stdev) of the EM method on three-tool aggregation.

	Precision	Recall	F_1 score
All Four Tools	0.61 (0.13)	0.99 (0.02)	0.74 (0.10)

Table 3.9: Average (and stdev) of the EM method on four-tool aggregation.

Two-Tool Aggregation

For a fair performance comparison, we used the same 15 worker groupings from Method 1 (Single-Tool Aggregation with Majority Voting) and 2 (Multi-Tool Aggregation with Majority Voting). We consider the ground truth labels as latent variables and estimate them jointly with the unknown parameters, the weight per tool on each pixel, of our model. The accuracy of two-tool aggregation with the EM method is summarized in Table 3.7. The comparison of F_1 scores of the tool pairs with that of the constituent tools is summarized in Figure 3.10 (a). The numbers for majority voting on single-tool aggregation are obtained from Method 1, and the numbers for majority voting on multi-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 EM-based multi-tool aggregation always performed sig-



(a) Two-tool aggregation

(b) Three-tool aggregation

(c) Four-tool aggregation

Figure 3.10: Accuracy 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). The blue bars are multi-tool aggregation with majority voting and the purple bars are multi-tool aggregation with the EM method. The red bars are single-tool aggregation of the best performing tool and the green bars are single-tool aggregation of the worst performing tool among all constituent tools. * significant at $p < .05$; ** significant at $p < .01$, both compared to EM-based multi-tool aggregation (two-tailed t-test). Leveraging tool diversity always performed significantly better than the inferior constituent tool, and performed at least as well as the superior tool.

nificantly 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). We observe that the highest aggregate-performance tool pairs were combinations of a high-precision (low-recall) and a high-recall (low-precision) tool ($< 3\%$), as shown in the third and fifth bar groups in Figure 3.10 (a).

Three-Tool Aggregation

For a fair performance comparison, we used the same 15 worker groupings from Method 1 and 2. We applied EM-based weight assignment for each pixel by setting the majority voting result as the initial weights. The accuracy of three-tool aggregation with EM is summarized in Table 3.8.

The comparison of F_1 scores of the three-tool aggregations with that of the constituent tools is summarized in Figure 3.10 (b). Similar to the two-tool aggregation result, the EM method improved the F_1 score significantly, but the gain was small (below 2%). The EM method significantly improved the performance of three-tool aggregation compared to majority voting, except for the aggregation of Drag-and-Drop $\times$ Pin-Placing $\times$ Floodfill. It is worth noting that while the EM method significantly improved accuracy for the tool pair Drag-and-Drop $\times$ Pin-Placing, adding Floodfill and forming a three-tool aggregation limited the benefits of the EM method. This implies that a more adaptive distribution of tool weights might be necessary when increasing tool aggregation complexity. This is further discussed in Section 3.6, where a correction mechanism is proposed to overcome the limitation of typical consensus-based pixel-level aggregation.

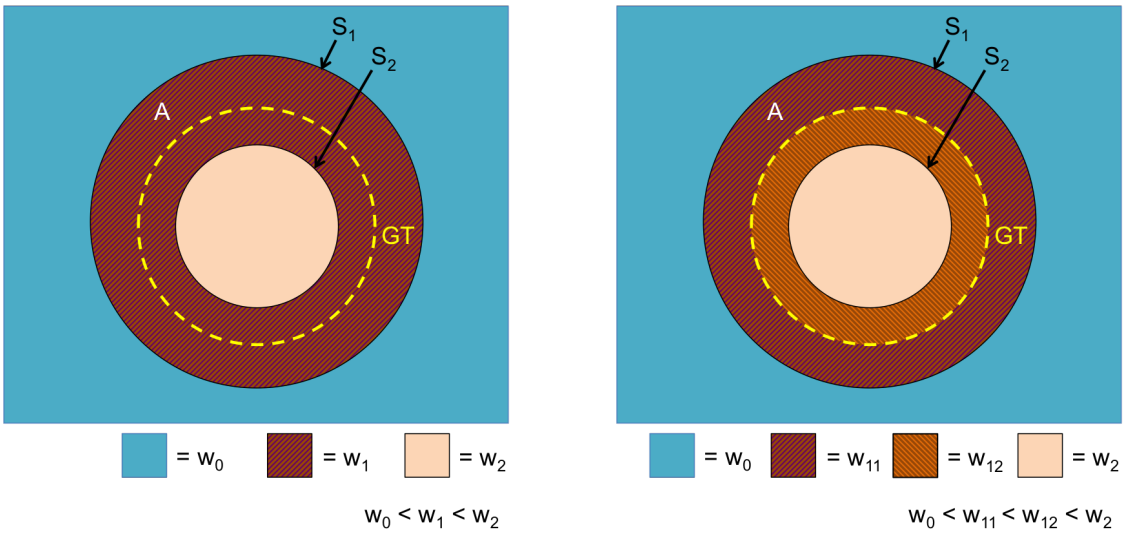
Four-Tool Aggregation

We apply EM-based pixel-level weight assignment to the four-tool aggregation condition to evaluate if the method could further improve aggregate accuracy. We used the same 15 worker groupings from Method 1 and 2. The result shows that the EM method significantly ($p < .01$) improves the aggregate performance of four-tool aggregations. However, as in other tool aggregations, the gain was small (below 1%). The accuracy is summarized in Table 3.9. The comparison of F_1 scores of the tool pairs with those of the constituent tools is summarized in Figure 3.10 (c).

In summary, combining answers from multiple tools increased the final segmentation accuracy compared to using the best single constituent tool alone. This performance improvement could be achieved by simple majority voting of segmentation results from different tools, and EM-based weight assignment to different tools could further improve the performance gains from majority voting. We analyzed multi-tool aggregation by varying tool combinations, and learned that 1) the diversity of systematic error biases across tools can lead to further improvement in the aggregation performance, 2) increasing the number of tools does not necessarily improve the multi-tool aggregation accuracy, and 3) offsetting the trade-off between precision and recall is critical to improving aggregate performance in the image segmentation domain. We believe that maintaining a sufficient amount of within group aggregation for each tool by increasing the group size of total workers is necessary to improve the multi-tool accuracy when increasing the number of tool types. In the next section, we will investigate how to further improve aggregate performance by exploring the error correction mechanism for multi-tool aggregation.

3.6 Error Correction Methods for Multi-Tool Aggregation

To further improve the accuracy of our tool diversity scheme, we explore the idea of using error correction mechanisms—post-hoc processes to further improve the aggregate performance by correcting errors that remain even after aggregation. The exploration extends the use of our tool diversity approach by providing options to improve aggregate accuracy even further when combining workers' answers across different tools. We first propose a morphological masking technique that can automatically balance errors between two biased tools. In image processing, morphological operations are defined as non-linear operations that transform images according to the shapes or features in an image. Our proposed method uses a non-linear operation to alter the label of each pixel by referring to the spatial feature of an aggregated



(a) Level of agreement by consensus-based aggregation.

(b) Ideal level of agreement to approximate ground truth.

Figure 3.11: The motivational concept of the morphological masking scheme. (a) S_1 indicates one segmentation, and S_2 indicates another. The yellow GT line indicates ground truth segmentation. Using general consensus-based aggregation (majority voting or EM), all the pixels within the area between S_1 and S_2 have the same level of agreement, w_1. However, to approximate GT, ideally, the area A ($S_1 \cap S_2{}^c$) needs a different level of agreement as in (b), with $w_1 1$ and $w_1 2$. Our correction mechanism can approximate GT by giving an updated level of agreement to pixels by referring to the agreement level of neighboring pixels.

result. Next, we explore the effect of varying the threshold parameter of the EM algorithm, and suggest a simple methodology to adjust the threshold to find the best optimization parameter for each object to be segmented.

3.6.1 Morphological Masking to Offset Biases between Different Tools

In Section 3.5, we observed that the aggregate accuracy of multiple tools can be improved when each of the tools have different systematic error patterns. In this section, we introduce a region-based morphological masking technique that synthesizes more accurate segmentations by propagating the level of agreement of neighboring annotations. The morphological masking technique further compensates for the

precision-recall trade-off by assigning an updated level of agreement to each pixel and segmenting the image based on a new theshold parameter.

The proposed correction mechanism can help improve aggregate accuracy by addressing several limitations faced by many consensus-based pixel-level aggregation methods, in particular those due to the fact that they do not fully make use of the rich spatial correlations between pixels in an image. The limitations and our corresponding solutions can be summarized as follows:

1. Conventionally, the label of each pixel is estimated independently without referring to its neighboring pixels' labels, despite the fact that they can have strong spatial correlations. We propose utilizing the spatial correlation by considering the average level of agreement of neighboring pixels when deciding the final level of agreement of a single pixel.

2. In most consensus-based methods, pixels with the same level of agreement for the same label are treated as equivalent, making it impossible to divide them into subgroups to increase the precision in labeling. We combat this problem by updating the level of agreement of pixels based on the average agreement of the neighboring pixels. For example, as in Figure 3.11 (a), let's assume that S_1 and S_2 are two different segmentation results, while GT is the ground truth segmentation to be estimated. In terms of precision and recall, S_1 results in low precision and S_2 results in low recall. With general consensus-based methods, we cannot approximate the ground truth boundary because the area $A\ (S_1 \cap S_2^{\mathsf{c}})$ with agreement level w_1 cannot be labeled with two different labels, e.g., foreground and background. However, by assigning the updated level of agreement to pixels as in Figure 3.11 (b), we can approximate GT by setting a threshold value between w_{11} and w_{12}.

Therefore, we propose a technique that applies morphological masking on each

pixel to refer to its neighboring pixels, so that, as in Figure 3.11 (b), pixels can have better updated level of agreements.

Method

The morphological masking that we introduce is a region-based operation that can modify the segmentation result by synthesizing more accurate bounds through referring to the average of the surrounding annotations. In the method, the label of a pixel is updated by referring to the sum of the neighboring pixels' level of agreement:

$$\bar{w}_p = \sum_{i=0}^{M-1} \frac{w_i}{M} \qquad \text{(Eq. 3)}$$

where $\bar{w}_p$ is the updated level of agreement of pixel p, M is the number of pixels inside the mask, and w_i is the original level of agreement of pixel i (the neighboring pixels). We also set a threshold parameter that decides the label of each newly updated pixel.

$$l_p = \begin{cases} 1, & \bar{w}_p > t \\ 0, & \text{otherwise} \end{cases} \qquad \text{(Eq. 4)}$$

where l_p is the label of pixel p, and t is the threshold parameter. The threshold parameter can be arbitrarily chosen within $0 < t < 1$ by the system designer.

Morphological masking updates the pixel agreement level around the transition area, for example, pixels close to line S_1 and S_2 in Figure 3.11 (a), where the level is changing from w_0 to w_1 and from w_1 to w_2, respectively. Note that the pixels far from the transition area do not get influenced by the masking. If t is set small, the aggregated recall increases because pixels with a low agreement level can be labeled as an object. If t is set large, the precision increases because only pixels with large

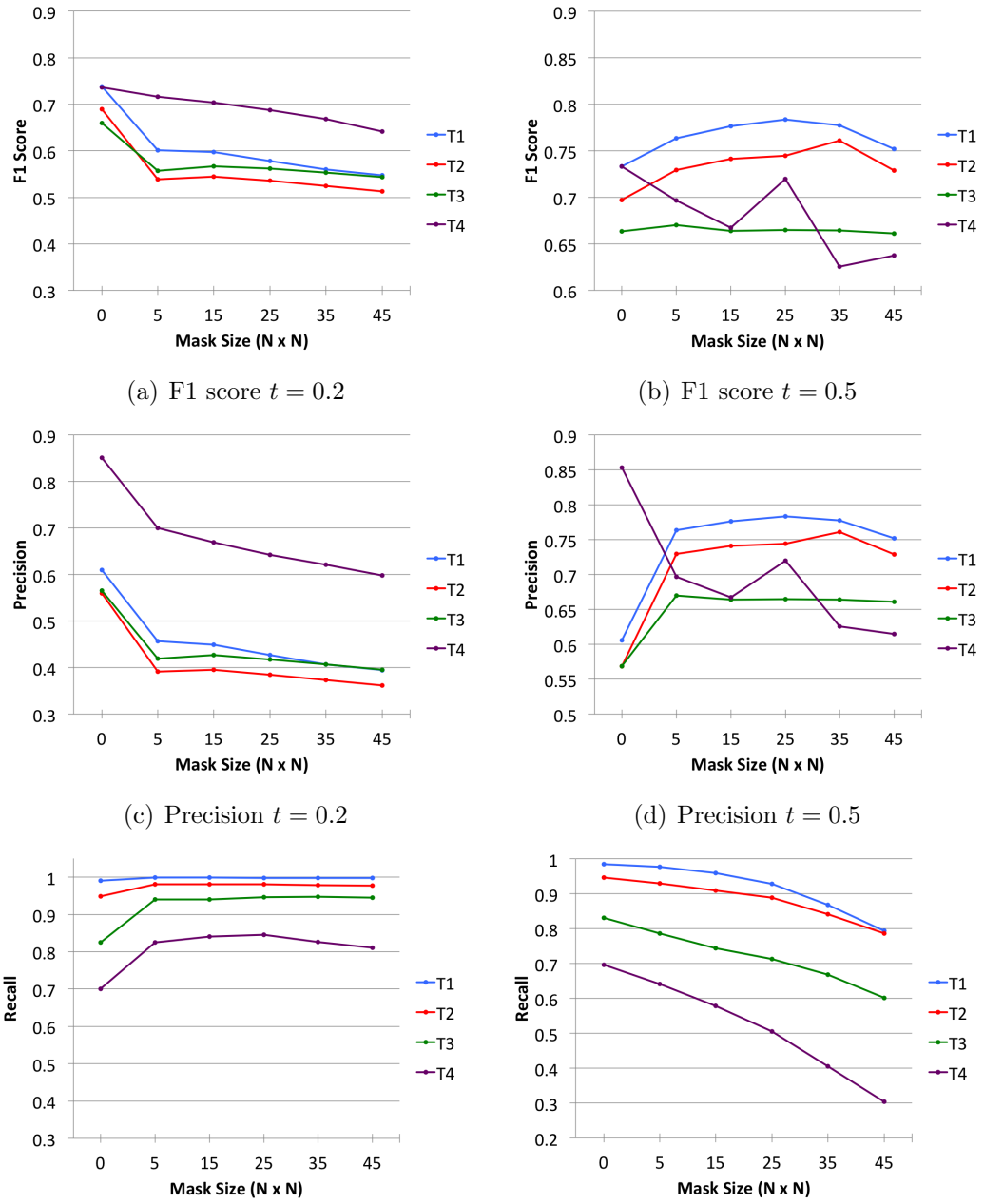
level of agreement can be labeled as an object. This feature makes it possible to better offset the precision-recall trade-off in multi-tool aggregation.

Evaluation and Results

We applied five different masking sizes and two different threshold parameters to explore the effect of our morphological masking technique. We randomly picked 10 random sampling of four workers for each tool combination types as in Section 3.5 to avoid any bias from repeatedly choosing a good or bad combination of workers. The mask sizes chosen are $N = [5, 15, 25, 35, 45]$ for $N \times N$ masks, and the threshold parameters chosen are $t = 0.2$ and $t = 0.5$. The experiment was implemented in Matlab 9.4 on 3.5 GHz Intel Core i7. We computed all five mask size conditions at once, and it took about a minute per object to compute majority voting of four workers, and about 1.4 minute per object to compute the EM-based weighting of four workers per object.

Figure 3.12 shows the morphological masking results of two different threshold parameters: 0.2 and 0.5. As expected, a low threshold ($t = 0.2$) increases recall but decreases precision, and a high threshold ($t = 0.5$) increases precision but decreases recall. The masking size also affected the performance. With $t = 0.2$, as the mask size increases, F_1 score decreased because of the steep decrease in precision but the low increase in recall. With $t = 0.5$, as the mask size increases, F_1 score increased except for Floodfill (annotated as T_4 in the figure) because precision largely increased while recall degraded no smaller than 0.55. The Floodfill (T_4) results in both Figure 3.12 (b) and (d) show a nonlinear spike at mask size 25×25. This is because there were less valid data points for higher thresholds and larger mask sizes. There were also many zero-precision data points when using Floodfill, which we did not include when computing the average.

To further explore the effect of our morphological masking, we investigate the



(a) F1 score $t = 0.2$

(b) F1 score $t = 0.5$

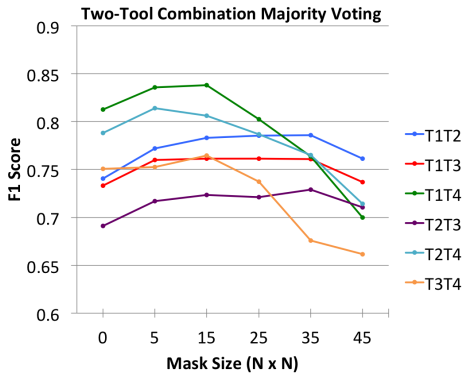
(c) Precision $t = 0.2$

(d) Precision $t = 0.5$

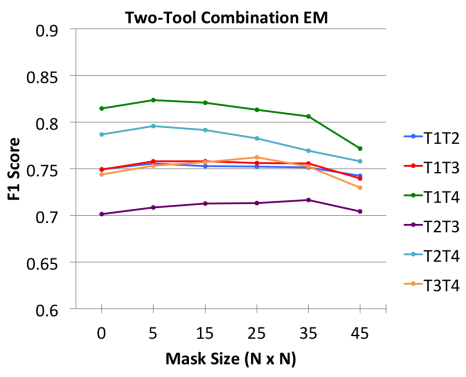
(e) Recall $t = 0.2$

(f) Recall $t = 0.5$

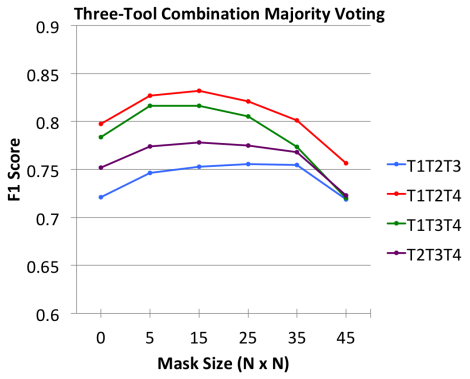
Figure 3.12: Results of single-tool aggregation with different threshold parameters for the morphological masking ($T1$ =Basic Trace, $T2$ =Drag-and-Drop, $T3$ =Pin-Placing, and $T4$ =Floodfill). The left column shows F_1 score, precision, and recall for $t = 0.2$ and the right column shows F_1 score, precision, and recall for $t = 0.5$. With $t = 0.2$, the F_1 score degraded by applying the mask. This is because of the large decrease in the precision with only a small increase in recall. With $t = 0.5$, the F_1 score improved by applying the mask up to 6%. (except for Floodfill). This is because the precision largely increased while recall degraded no larger than 0.23.



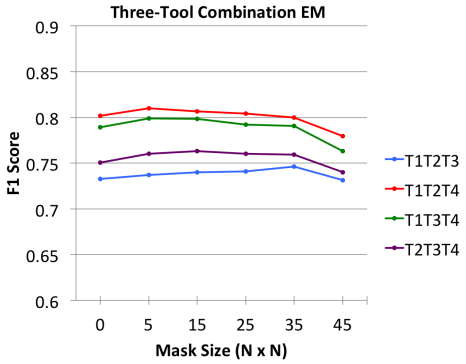
(a) 2 tools combination (Majority Voting)



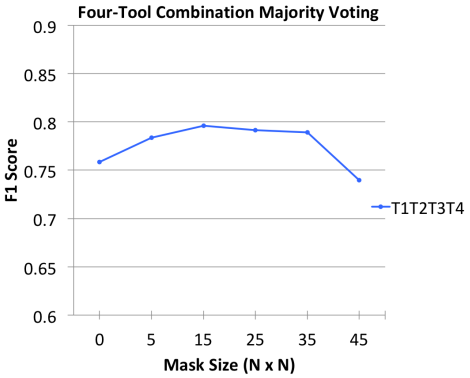
(b) 2 tools combination (EM)



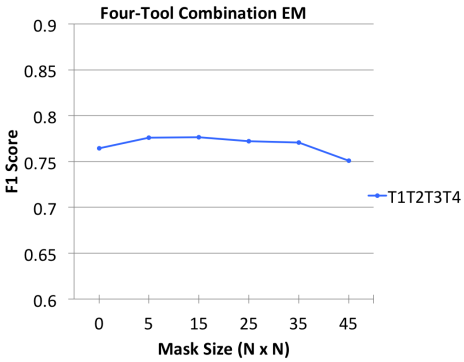
(c) 3 tools combination (Majority Voting)



(d) 3 tools combination (EM)

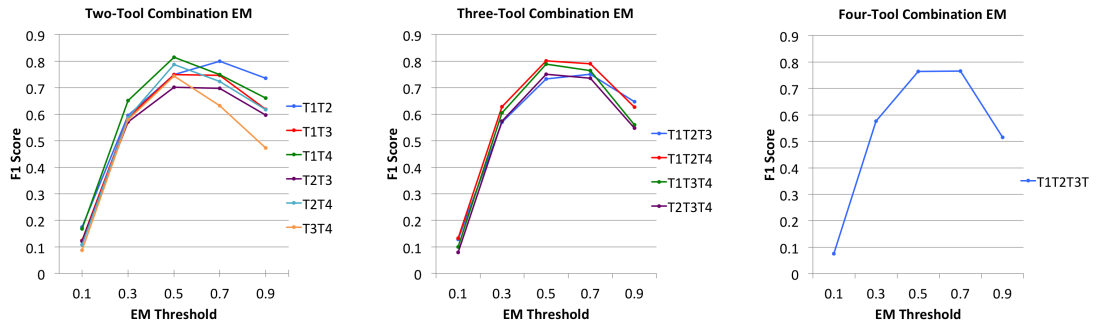


(e) 4 tools combination (Majority Voting)



(f) 4 tools combination (EM)

Figure 3.13: F_1 scores of multi-tool aggregation with different masking sizes ($T1$ =Basic Trace, $T2$ =Drag-and-Drop, $T3$ =Pin-Placing, and $T4$ =Floodfill). The left column is F_1 score of majority voting and the right column is F_1 score of EM-based weighted aggregation. First row is two tools pairs, second row is three tools combinations, and third row is four tools combination results. Every multi-tool combination condition improved accuracy up to 6% by applying our masking technique. The mask size that induced the largest performance improvement varied by tool combination types.



(a) Two tools combination (b) Three tools combination (c) Four tools combination

Figure 3.14: F_1 scores of every tool combination with five different EM thresholds (uniform intervals from 0.1 to 0.9). The result shows that the maximum performance that can be achieved varies by the threshold value, implying that correctly setting the EM threshold parameter can further improve the aggregate accuracy.

effect of mask size with $t = 0.5$ for all possible tool combinations of FourEyes. The result is shown in Figure 3.13. The left column shows the effect of masking on majority voting and the right column shows that on EM-based weighted aggregation. It is observed that masking *always* improves the aggregated result further up to maximum 5.8% for majority voting and 2.4% for EM. The mask size that induced the largest performance improvement varied by tool combination types. The mask size with 5, 15, and 35 induced maximum accuracy at least for one combination type.

In summary, applying our proposed morphological masking technique could further correct aggregation errors remaining in multi-tool aggregation. The effect of the masking technique was observed in every tool combination possible by FourEyes. This implies that not only the tools' design, but also the settings of the correction mechanism can affect the aggregate accuracy of multi-tool combinations.

3.6.2 The Effect of the EM Threshold

In Section 5.2, we saw that EM-based weighted aggregation can improve the aggregate accuracy of the tool diversity approach. However, we did not fully investigate the effect of the threshold parameter of EM on performance. Threshold parameter

adjustment can be a simple and easy technique to apply in the post-processing stage if it gives a better result. Thus, we investigate the effect of different threshold parameters for EM-based weighted aggregation. We used the same 10 random sampling of four workers in Section 3.5 to avoid any sampling bias. Figure 3.14 shows F_1 scores of each tool combination with different EM thresholds. In Figure 3.14(a), we see that while the best performing EM threshold was 0.5, for Basic Trace $\times$ Drag-and-Drop, the highest performance was achieved when the threshold was 0.7. This is likely because Basic Trace and Drag-and-Drop are the two tools with high recall but low precision characteristics. The large threshold parameter compensates the precision and recall trade-off by inducing higher precision when aggregated. In Figure 3.14(b), the highest accuracy was achieved when the threshold is 0.5, except for Basic Trace $\times$ Drag-and-Drop $\times$ Pin-Placing, which achieved the highest accuracy when threshold is 0.7. This is expected because without the Floodfill tool, which has the opposite characteristics from Basic Trace and Drag-and-Drop, trying to induce higher precision can help gain better compensation between the precision and recall trade-off. Thus, we suggest that future system designers to use methods like a parameter sweep to find the best threshold to leverage the characteristics of each tool.

In summary, the threshold parameter of EM affected the aggregate performance. Different tool combinations had different threshold parameters that maximize their performance. This implies that a post-hoc process of choosing the best threshold parameter can further improve the aggregate accuracy of crowdsourced answers. In the next section, we discuss some guidelines for system designers who aim to use tool diversity to improve performance of their crowd-powered system.

3.7 Discussion

FourEyes's approach of leveraging tool diversity in designing a crowd-powered system goes beyond the paradigm of conventional crowdsourcing strategies, which

divide a task into smaller microtasks and aggregate answers from workers using a single tool. FourEyes divides *tools*—and uses multiple different tools with different systematic error biases—to improve the accuracy of aggregate crowd answers.

A practical concern in applying the tool diversity approach is the cost and effort in building multiple tools with different (and even complementing) characteristics. In domains where multiple standardized tools already exist, e.g., image labeling or handwriting transcription, system designers can simply import and aggregate the existing tools without having to develop multiple customized tools. In this setting, we suggest that it is worth using all tools available and applying the various techniques introduced in this chapter, not just testing to find the best *one* tool.

However, we found that simply adding more tools does not linearly increase accuracy. This might be due to the small aggregate group size that we purposely constrained for fair comparisons with single-tool aggregation. That is, the insufficient number of *within-tool* workers contributing might have led to limited improvement when leveraging multiple tools. This implies that additional judgments such as the within tool group size could have effects on the overall accuracy of the multiple tools configuration. From our observation, we expect the multi-tool approach would improve accuracy with a larger aggregate group size, and it would be the same as we would expect for single-tool aggregation—adding more answers can improve the result, but sub-linearly. Also, the unique characteristics of a tool need to be considered as they might affect the crowd's answer: e.g., the amount of interaction, the complexity of interface layout, etc.

To maximize the benefit of plugging in multiple tools, the tools combination should be carefully chosen by the system designer to maximize benefits from leveraging tool diversity. The following section discusses guidelines on using the tool diversity approach in the system design.

3.7.1 Compensation of Biases in leveraging Tool Diversity

To benefit from leveraging tool diversity in building crowdsourcing systems, a system or a task should have clear and distinct trade-offs in it's accuracy and it should be possible to build tools that can target one aspect at a time. Once different tools are built with distinctive properties in terms of their systematic biases, an aggregated method that can offset the biases should be applied to merge the answers. Errors remaining after aggregation should be corrected using an error correction mechanism such as our morphological masking. 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 (*Mao et al.*, 2013). 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 different 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.

3.7.2 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. Tasks such as real-time captioning (*Lasecki et al.*, 2012) or handwriting recognition (*Ouyang and Li*, 2012) 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 chapter may also benefit from our approach.

3.7.3 Envisioned Scenario

In this section, we illustrate how our proposed multi-tool approach and the post processing methods, aggregation and correction, can be strategically used in a probable scenario.

Crystal is a developer at a computer vision startup company. Her team recently built several crowd-powered image segmentation tools that work pretty well in the lab, but she is not sure which one will work the best when deployed in the wild. Instead of trying to find the best performing tool among them, she decides to use all

the tools that the team built by leveraging tool diversity (Section 3.2). Therefore, for a single segmentation task, the crowd answers from every tool are collected. She knows that the results will get better than any single tool used alone if the EM method described in this chapter (Section 5.2) is applied. This improvement can be gained without having to know the characteristics of each tool a priori. To further improve the final accuracy, she performs a parameter sweep to find the best EM threshold for the tool sets based on a small manually-annotated ground truth sample from her dataset (Section 3.6.2). Additionally, she applies a correction method that updates the level of agreement on each pixel being labeled as foreground or background. The correction method leverages the characteristics of the tools to more precisely correct for each tools' biases (Section 3.6.1). With this process, the team built a system that gives more accurate segmentation result than any single tool they built.

3.8 Summary and Future Work

In this chapter, we have introduced a generalizable crowdsourcing approach of leveraging tool diversity to increase the output accuracy. When building a system, different tool designs can induce different worker performance, leading to different systematic error biases. Prior work has used task decomposition (into microtasks) to increase the reliability of a single task. However, systematic error biases can persist even after a task is divided as much as possible, if only a single tool is used for the task. We claim that these systematic error biases can be reduced by using multiple tools for the same task resulting in improved aggregate crowd performance. We demonstrated the effectiveness of the tool diversity strategy in the domain of the semantic image segmentation problem. In our experiments, we used FourEyes, a crowd-powered image segmentation system that consists of four different image segmentation tools, to segment diverse objects in different visual scenes. A series of studies showed that using multiple tools can significantly improve the aggregate

accuracy of a single task, especially when the trade-off between the aggregated tools is high and the aggregation and correction method offsets the trade-off in the right direction. Overall, our findings present new opportunities and directions for gaining a deeper understanding of how tool designs influence the aggregate performance on crowdsourcing tasks, and introduces a new way of thinking about decomposing tasks: based on tools instead of subtasks.

For future work, we plan to compare our EM-based aggregation method with TIS_M method (*Griffin and Corso*, 2019) that combines multiple different annotations on the same object with consistent improvement in segmentation accuracy. Performance comparison with both static images and dynamic videos would allow us to investigate the benefits of each method in different context.

Future work may investigate methodologies for leveraging tool diversity in other domains, such as video coding (*Lasecki et al.*, 2014a), annotation of fine-grained categories (*Gebru et al.*, 2017), or activity recognition (*Lasecki et al.*, 2013b). For instance, using multiple tools for the same task may benefit any NLP task with multiple channels. A system designer can devise a tool that focuses on processing a text channel while sacrificing the audio channel, and aggregate the result with a tool that focuses on the audio channel, while sacrificing processing the text channel. Furthermore, this approach may open new ways of optimizing the effort from both humans and computers—considering them as different resources with different systematic error biases—to leverage the best of both worlds.

CHAPTER IV

Perspective Diversity: Reconstructing 3D Video Using Particle Filtering to Aggregate Responses

While leveraging tool diversity allows for reducing systematic error biases that are induced by the tools used to perform the tasks, multiple tools cannot solve all types of biases. In this chapter, we introduce the approach of leveraging perspective diversity to reduce biases induced by the data instance itself. The different perspectives from multiple data instances enable us to complement the lack of information from one data instance source, reducing the bias created from limited source of information.

4.1 Motivation

Autonomous vehicles collect large quantities of training data by operating, or being operated, in their target environment. This data is used to teach vehicles how to adjust to and interact with the physical world (*Bojarski et al.*, 2016). However, research suffers from a lack of realistic training data, especially of rare and unusual events such as traffic accidents (*Kalra and Paddock*, 2016). To collect sufficient training instances of such rare events, autonomous vehicles need to run and record hundreds of billions of miles in the wild, corresponding to decades of operating a car on the roads.

To provide a specific example, Waymo's autonomous research vehicles travel and record approximately 25,000 miles every day on public roads (*Waymo*, 2018), while Americans drive a total of nearly three trillion miles every day (*Kalra and Paddock*, 2016), a factor of 120 million. Thus, creating realistic simulated 3D scenes (*Waymo*, 2017) from abundant *existing* traffic videos crawled from those available on the Web, such as on YouTube, is a more reasonable method for creating realistic training data of rare events at scale. This process of creating 3D scenes from real-world monocular video is called *3D video reconstruction*. Generally, manual annotations are necessary at some point of the process to bridge the sensory and semantic gap between 2D and 3D. To efficiently scale up manual work, one can benefit from crowd-powered tools that rapidly leverage human effort.

Even though crowdsourcing has been widely studied in image and video annotation tasks (*Bigham et al.*, 2010; *Laput et al.*, 2015; *Vondrick et al.*, 2013; *Zhong et al.*, 2015), crowdsourcing techniques for 3D video reconstruction remain underexplored. This is due to the high degree of difficulty of the task, where even a small error in the annotation results in a significant error when re-projected into 3D. For example, as shown in Figure 4.2, a three-pixel difference in the 2D annotation of vehicle height can result in a 26-meter difference in 3D position estimation. Therefore, quality control is a crucial component in both the answer aggregation and 3D state estimation stages to avoid such error amplification. One way to control annotation quality is to filter out as many poor annotations as possible before aggregation. Providing workers an option to skip questions about which they are unsure is known to clean the data at the time of collection (*Chang et al.*, 2017; *Shah and Zhou*, 2015). We define this action as the self-filtering of worker annotations. Self-filtering can be particularly useful in image and video annotation, as oftentimes it is nearly impossible to generate the correct annotation due to artifacts such as motion blur, change in angle of view, truncation, or cropping in individual frames (*Vondrick et al.*, 2013). However, this type of filtering

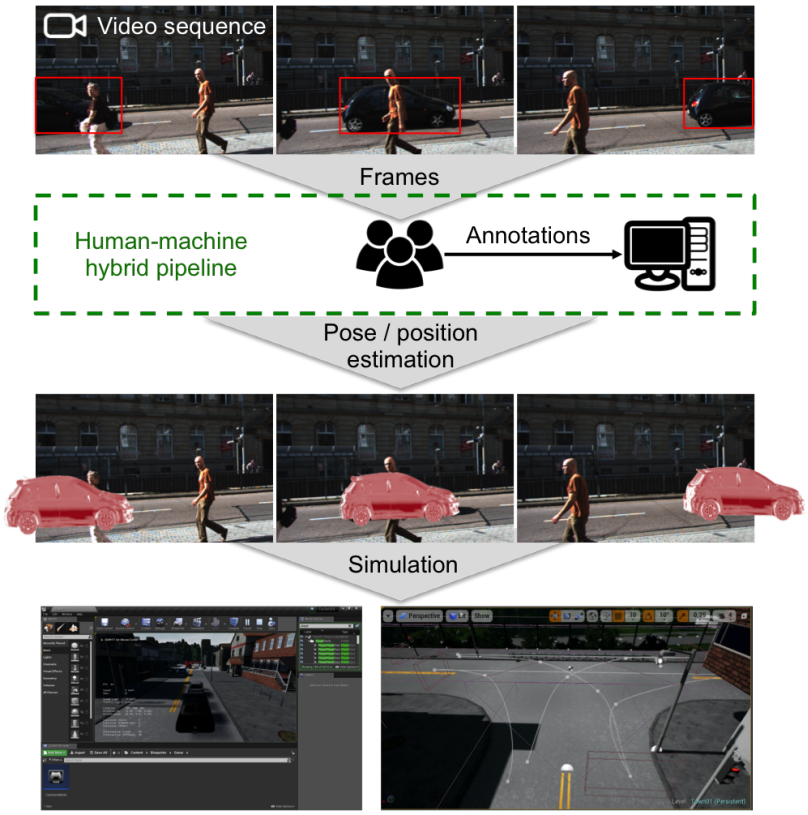


Figure 4.1: We propose a crowd-powered human-machine hybrid system for collecting and aggregating annotations for state estimation of 3D objects in 2D videos. Our approach leverages particle filtering to accurately reconstruct 3D scenes from 2D sources even with missing annotations, which can enable generating simulated realistic large 3D datasets.

should be handled carefully because it may result in missing annotations, e.g., where *all* the workers self-filtered, resulting in system failure due to the 3D reconstruction problem being underdetermined—having fewer equations than unknowns.

In this chapter, we propose a novel crowd-powered human-machine hybrid pipeline for 3D video reconstruction as in Figure 4.1. We make use of the additional information in the temporal dimension of the video to improve the quality of aggregated annotations and their corresponding 3D state estimates. Our particle filter (*Thrun*, 2000) based aggregation strategy allows us to utilize information from multiple time frames. This is especially useful when there are "missing" annotations, because the

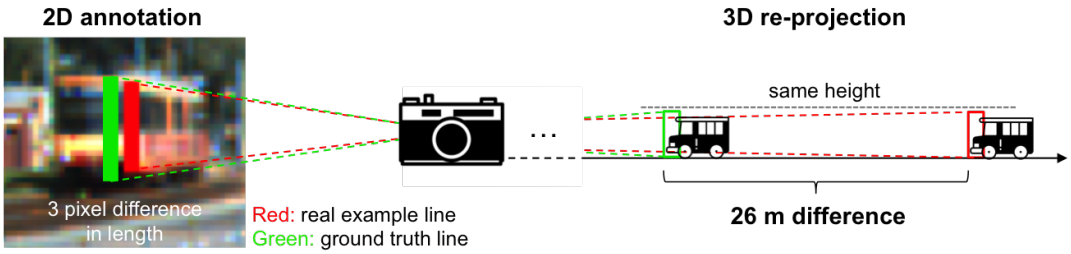


Figure 4.2: A small pixel error in 2D can be amplified in the Z-dimension, resulting in a severe position error. The vehicle image on the left shows a crowdsourced *height entry* dimension line annotation (in red) and the corresponding ground truth (in green). The z-dimension estimate can be calculated from the focal length and the object's actual height, which was 721 pixels and 3.59 meters in our experiment, respectively. The three-pixel difference in dimension line leads to a 26-meter difference in 3D location.

impact of these annotations can be mitigated by referring to neighboring frame estimates. This ability allows the requester to set more aggressive filtering thresholds, which improves theoverall annotation quality despite the risk of incomplete per-frame annotations. Our work proposes a generalizable solution for crowdsourcing annotations on videos, even in circumstances where annotation is challenging and especially error-prone.

Popup, our crowd-powered system, uses the crowd to collect annotations of 3D dimension lines on 2D videos and aggregates these annotations using particle filtering to generate 3D state estimates of objects of interest. We validate our method on videos from a publicly available and established dataset (*Geiger et al.*, 2012) of traffic scenes. The experimental results show that our proposed approach reduces the relative error by 33% in position estimation compared to a state-of-the-art baseline condition which uses L-BFGS-B (*Zhu et al.*, 1997) to optimize the re-projection of a 3D cuboid onto each video frame for state estimates. Further, our proposed aggregation method is more robust in cases with missing annotations, where the baseline method will fail due to the problem being underdetermined. Because Popup enables self-filtering, the annotation time for challenging frames can be reduced by 16%.

The main contributions of the chapter are as follows:

- A novel means of aggregating and processing multiple annotations at different frames in videos using particle filtering, which enables more accurate 3D scene reconstruction even with an incomplete annotation set.
- Popup, a crowd-powered system that estimates 3D position and orientation of objects from 2D images, using crowdsourced dimension line annotations on objects and their actual dimension lengths.
- Experimental results from 17 videos annotated by 170 crowd workers that validate the efficiency of our proposed annotation aggregation method that significantly improves the accuracy of 3D state estimation and enables quality control through more aggressive filtering thresholds.

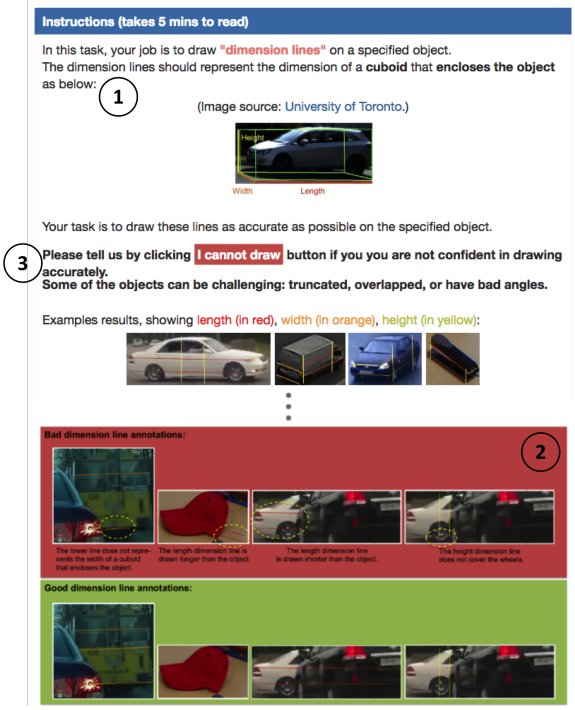
4.2 Approach

Crowdsourcing leverages human intelligence via platforms such as Amazon Mechanical Turk to help perform tasks that are otherwise difficult for fully automated systems. The task of reconstructing 3D animations from typical RGB videos can benefit from crowd-powered hybrid pipelines because human computation can provide manual annotations at scale to help the automated system bridge the semantic and sensory gap between 2D and 3D. Our work proposes novel crowdsourcing strategies for collecting and aggregating annotations for 2D to 3D scene reconstruction, which build on previous techniques for crowdsourcing video annotations and improving the quality of aggregated crowd answers. Our proposed particle-filtering-based aggregation method enables self-filtering because the missing annotation can be compensated for via temporal coherency.

To prevent the amplification of annotation errors in 3D state estimates, filtering as many low quality annotations as possible is a necessary step prior to aggregating

and estimating the 3D states of objects. However, missing annotations are difficult to handle in the subsequent steps and a specialized treatment is necessary to avoid system failure. Our work is conceptually motivated by inter-frame prediction techniques in video coding (*Wiegand et al.*, 2003), which takes advantage of temporal coherency between neighboring frames to predict pixel values of missing sub-blocks in subsequent frames.

To exploit this temporal coherency, we developed a particle filter which operates on the crowdsourced data. Particle filtering is a recursive Bayesian method which estimates a probability density function across a state space by maintaining a large set of particles, each of which represents a potential position ("hypothesis"). Particle filters are commonly used in simultaneous localization and mapping (SLAM) systems (*Montemerlo et al.*, 2002; *Montemerlo and Thrun*, 2007), as well as face (*Kwolek*, 2006), head (*Oka et al.*, 2005), and hand tracking (*Bray et al.*, 2004).We selected the particle filter as our state estimation technique for three main reasons: first, particle filters can utilize information from neighboring state estimates in tandem with temporal constraints (e.g., the object has a maximum speed) to refine the state estimate. Second, particle filters can support complex measurement functions, which are required to compare 2D annotations and 3D states. Last, the particle filter does not assume an underlying distribution, which allows it to maintain multimodal and non-Gaussian distributions. This is particularly useful, as incomplete annotations cause multiple correct hypotheses. To validate the efficiency of our proposed aggregation and estimation method, we applied two filtering methods for low quality annotations in two different stages: self-filtering at the time of annotation collection and outlier filtering at the time of aggregation.



(a) Instructions for crowd workers

(b) Worker UI for dimension line annotation

Figure 4.3: Crowd worker instructions and the interactive worker UI. (a) Step-by-step instructions with good and bad examples are provided. (b) Interactive Web UI that workers can use to create, adjust, erase, and redraw dimension lines.

4.3 POPUP

Popup leverages dimension line annotations in 2D videos using particle filtering to achieve efficient and accurate 3D position and orientation estimation. Popup's pipeline consists of three main components: (1) dimension line annotation with self-filtering, (2) outlier filtering of submitted annotation sets, and (3) particle filtering based estimation of the 3D position and orientation. The overall pipeline is described in Figure 5.4 By feeding the output from Popup to a simulator such as, CARLA (*Dosovitskiy et al.*, 2017), a user can reconstruct and replay in 3D an event captured in monocular video. This allows the user to generate large realistic simulated training datasets for data-driven algorithms.

4.3.1 Dimension Line Annotation Tool and Self-Filtering

Popup presents crowd workers with a visualization and annotation web application that allows them to crop the object of interest from a video frame and then draw dimension line annotations of the three dimension entries: length, width, and height, on the cropped object. The dimension lines can be directly drawn on objects in video frames (Figure 5.6(b) ①) to capture the 3D state of an object without any three dimensional interactions, e.g., rotation and scaling of a cuboid, which requires familiarity with interactive 3D tools.

When a crowd worker reviews the dimension line annotation task, an explanation on the goal of the task is given first (Figure 5.6(a) ①). Then, step by step instructions are provided with pictures exemplifying desired and undesired annotations as in Figure 5.6(a) ②. The instructions also state that workers should click the "cannot draw" button whenever they are not confident in drawing accurately for a particular entry (Figure 5.6(a) ③). Once the worker accepts the task, they can perform the first step: cropping the target object. The worker can click and drag on the given video frame to draw a box, and use the vertices of the box to adjust the size and ratio.

The coordinate information of the box is used in the post-hoc outlier filtering step, as explained in the next section. Once the worker is done cropping the target object, she can click the 'Done with Step 1' button and proceed to the next step. Note that a worker works on annotating single frame at a time. The sampling rate of frames to be annotated by workers can be arbitrarily determined by the user.

The second step is drawing the dimension line entries (length, width, and height) on the cropped vehicle. The interface has buttons that open a pop-up window to allow workers to draw line annotations for each dimension entry. Workers can choose which entry they want to draw first. The interactive pop-up window is shown in Figure 5.6(b). After drawing a line, a popover message appears at the end of the line and asks workers "Is this end closer to the camera than the other end of the line?" The worker can answer this using a radio button. We initially asked this question to avoid the Necker cube illusion (*Necker*, 1832) that occurs when drawing a cube with no visual cues for orientation. The illusion makes it impossible to distinguish the closer ends of the edges of a cube. However, the closer end annotation was not used in our final orientation estimation due to a large variance in the answers. Nevertheless, we report this step in the chapter because it affects the total time of completing the dimension line annotation task. The interface forces workers to draw more than one line per dimension to proceed to the next step. The interface allows adjusting already drawn dimension lines or redrawing them anytime if needed. Workers are provided with the "cannot draw" button (Figure 5.6(b) ②) which they can click on to self-filter dimension line annotations if they are not sure about their answer.

4.3.2 Outlier Removal

Popup is designed to robustly handle aggressively filtered annotation sets. Popup has two post-hoc filtering modules to control the quality of collected annotations. The post-hoc modules assume multiple submissions per frame so that distribution

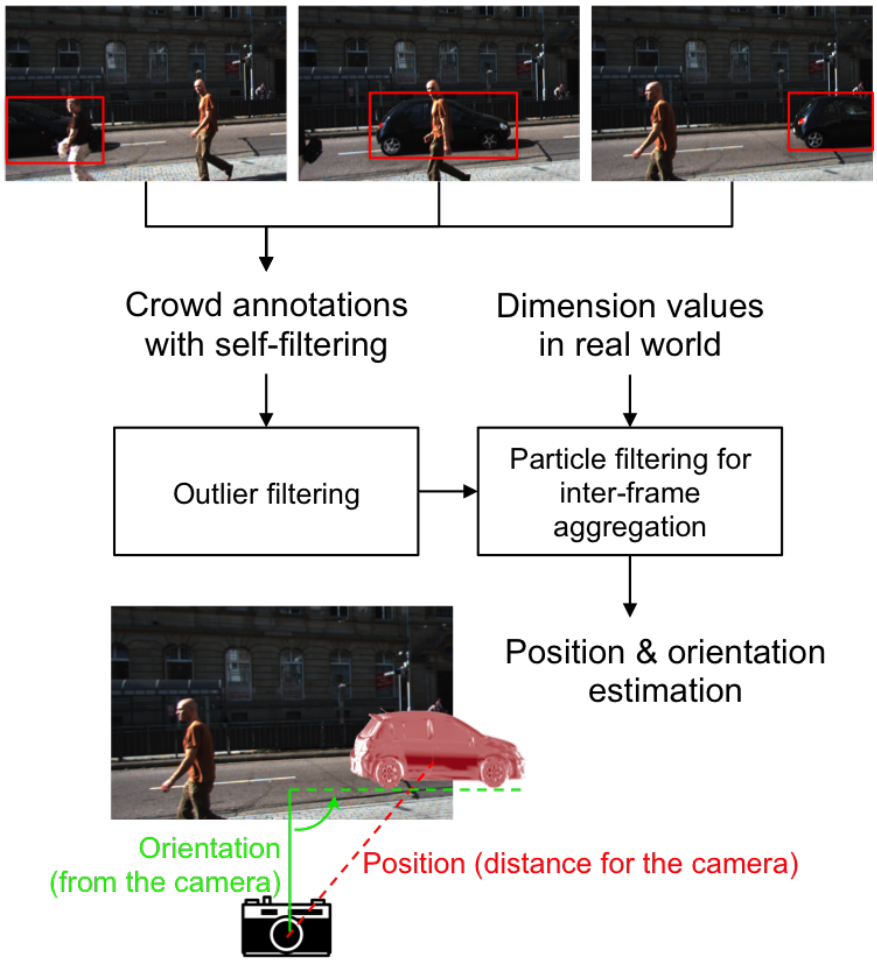


Figure 4.4: Overview of Popup pipeline. From workers' dimension line annotation input (on the 2D image) and additional input of real-world dimension values of the target vehicle (looked up from an existing knowledgebase), Popup estimates the position and orientation of the target vehicle in 3D.

statistics can be found. The first step uses the median location of the bounding boxes workers cropped from the given frame. The second step uses the standard deviation between the dimension lines drawn for the same object in the same frame.

Step 1: Filtering Annotation Sets

The first step calculates the median bounding box location of submissions to filter incorrect annotation sets. For each target object, the worker crops the object of interest from the given frame. Our assumption is that a malicious worker, careless

worker, or bot will fail to crop the correct target object. For width and height independently, if a cropped box does not overlap more than 50% with the median of the cropped boxes we drop the annotation set of all three entries, assuming the worker annotated the wrong object. This is designed to entirely filter poor submissions.

Step 2: Filtering individual Annotations

The second step compares the distance of the length and angle of submitted dimension line annotations from the medians. If a dimension line is outside 1.5×Interquartile Range (IQR) from the median, it is filtered. This is useful for filtering out mistakes, e.g., a height entry mistakenly drawn as a length entry or a line added by mistake and not removed, and to filter out low quality annotations. We use relative distances instead of absolute values as filtering criteria because the size of an object can differ from 30 pixels to 300 pixels.

4.3.3 Particle Filtering for Position and Orientation Estimation

Popup uses particle filtering as a method for aggregating annotations and estimating 3D states of the target object in the video. A particle filter works by generating many particles, each of which represents a potential state (hypothesis), and using them to track and combine three probability distributions at each time step (video frame) (*Thrun*, 2000):

(1) The *previous distribution*: where the object was previously.

(2) The *transition distribution*: where it could be, given where it was previously.

(3) The *measurement distribution*: where it could be, given the annotated lines.

Using these three distributions, the particle filter handles heavily filtered data and provides more refined estimates by leveraging temporal constraints. The particle filter used by Popup embodies these three probability distributions as follows:

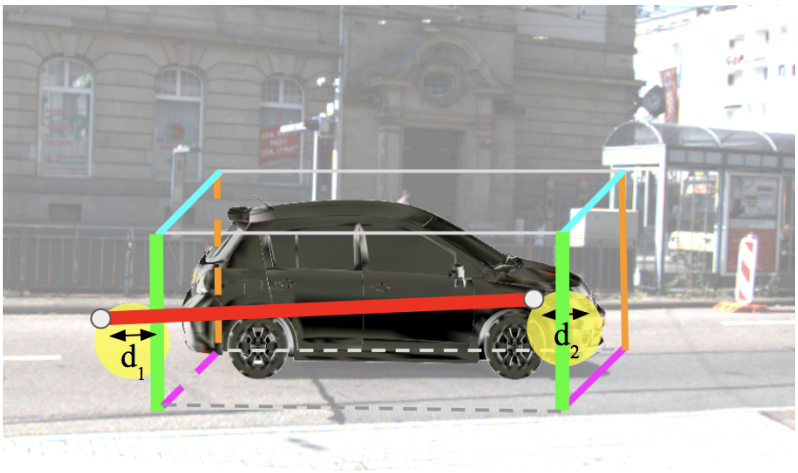


Figure 4.5: Perceptual distance calculation. The distances (arrows) between endpoints (grey dots of the red line) of an annotation (red line) and corresponding projected hypothesis 3D line pairs (orange, green, blue, pink) are calculated. The distances corresponding to the best-fitting 3D line pair are used to calculate probability. These probabilities are used to determine which hypothesis most closely represents the annotation line, and therefore the position in 3D space.

(1) Previous State Distribution: The previous distribution corresponds to the final distribution of the previous time step. As we do not have any information about the vehicle's initial pose, we set the distribution at $t = 0$ as uniform within the bounds described in Experimental Setting section.

(2) Transition Distribution: The transition distribution describes the probability of a particle being in a new location given its previous location. This distribution allows the filter to maintain knowledge of potential states across time, which has two important implications: first, it means a fully determined system is not necessary at every time step, so the system is tolerant to self and post-hoc filtering with tight thresholds. Next, it applies a spatiotemporal constraint by limiting how far the vehicle can move in successive frames, narrowing the solution space. Typically the transition distribution is based on knowledge of the vehicle's kinematics and control inputs, but as our system has knowledge of neither, we introduce uncorrelated zero-mean Gaussian noise that spans the set of reasonable vehicle motions.

(3) Measurement Distribution: The measurement distribution utilizes the crowdsourced annotation lines to determine the likelihood of the hypothesis. To test a hypothesis, we create the bounding cuboid in 3D space and project it onto the image. Then, for each annotation, we determine how close its endpoints are to an appropriate pair of edges (Figure 4.5). This distance is then placed on a normal distribution with a mean of 0 pixels and a standard deviation of 22 pixels. We also calculate the difference between the lengths of the annotation line and corresponding projected hypothesis line, and place that on a normal distribution with a mean of 0 pixels and a standard deviation of 22 pixels. The sum of these two probabilities is used as the probability of an annotation. This function is referred to as ERR in Algorithm 1.

Algorithm 1 Particle filter algorithm for FourEyes

Let $S = \{(s_1, w_1) \dots (s_N, w_N)\}$ be the set of N particles, where each particle $s_i = \{x_i, y_i, z_i, \theta_i, f_i\}$ is one hypothesis with probability $P(s_i) = w_i$. Let the initial set of particles S_0 be sampled uniformly from the given range for s_i:
for Every Frame t **do**
  RESAMPLE(S)
  for Every (s_i, w_i) in S **do**
    Next State Step: $s_{i,t} \leftarrow s_{i,t-1} + \mathcal{N}(0, \sigma)$
    $z \leftarrow 0$
    for Every Annotation Line **do**
      $z \leftarrow z + ERR(AnnotationLine, ParticleState)$
    end for
    $w_{i,t} = w_{i,t-1} \cdot z$
  end for
  $S \leftarrow NORMALIZE(S)$
  $estimate \leftarrow ARGMAX(w_i)$
end for

4.3.4 Implementation

The pseudocode for our system is shown in Algorithm 1. Our state space consists of five dimensions: x, y, z, θ, and f, where x, y, z denote the relative 3D location of an object from the camera and θ denotes the orientation as illustrated at the bottom of Figure 5.4. The last dimension, f, denotes the focal length of the camera. When

analyzing across a single frame, we perform the action and resampling steps after iterating through every set of annotations.

4.4 Evaluation

To evaluate the performance of our proposed 3D video reconstruction strategy, we investigate the accuracy of dimension line annotations before and after pre-processing filterings: self and outlier filtering. Next, we investigate our proposed annotation aggregation strategy that uses particle filtering to refer to neighboring frames. For the experiments, we recruited 170 workers using LegionTools (*Gordon et al.*, 2015) and routed them from Amazon Mechanical Turk to our crowd-powered system, Popup.

4.4.1 Experimental Setting

The evaluation is done using the KITTI dataset (*Geiger et al.*, 2012) that contains traffic scenes recorded from a moving vehicle using multiple sensor modalities. Along with 2D RGB video scenes, the dataset provides ground truth measurements of distance and orientation of objects relative to the camera. The scenes include occluded, truncated, and cropped objects that are challenging and thus appropriate to test the performance of Popup.

In this experiment, we targeted reconstructing the 3D state of one moving vehicle per video clip. There were a total of 21 video clips in the dataset, of which we used 17, dropping unfit ones. We sampled 10 frames from each video clip at a rate of 2 frames per second. For each video clip, we recruited 10 workers to provide annotations. Each worker annotated every other sampled frame, for a total of five frames. That is, for each frame, annotations from five different workers were collected. Each worker was paid \$1.10 per task to meet a pay rate of ∼\$9/hr. To understand the reason for self-filtering whenever it happens, we gave the workers a multiple-choice question on why they self-filtered. The choices were: 1) "The object is heavily occluded", 2) "I

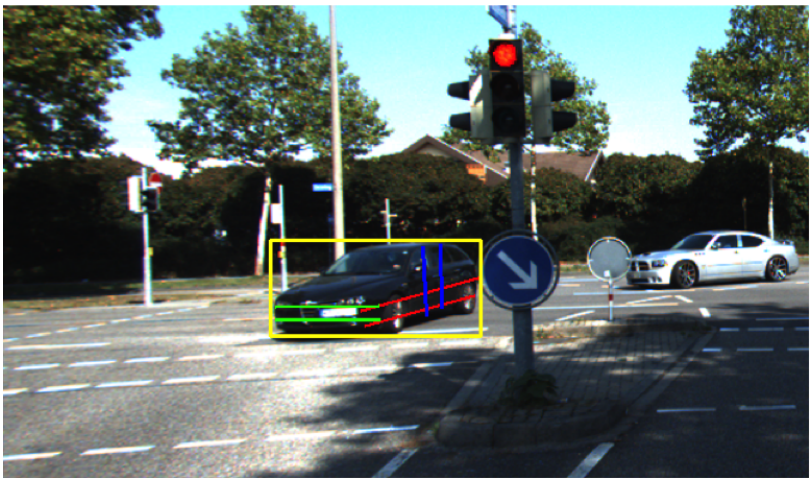


Figure 4.6: Example of dimension line annotations from one of the crowd workers who participated in our experiment. The yellow bounding box is the area that the worker cropped in Step 1 of the task, and the red, green, and blue lines are length, width, and height annotations, respectively, drawn in Step 2.

don't understand the instruction", and 3) "Others". We asked the workers to still draw the dimension line after reporting "cannot draw" to directly compare accuracy with and without workers' self-filtering.

To obtain the required 3D dimensions of the annotated vehicles, we utilized the ground truth information included in the KITTI dataset. In a real world deployment of Popup, the dimensions would be found online or in appropriate documentation prior to generating the 3D reconstruction. To reduce computation time and avoid suboptimal estimation, we set bounds for each estimated dimension: $-30 \leq x \leq 30$, $-4 \leq y \leq 4$, $1 \leq z \leq 140$, $0 \leq \theta < \pi$, and $500 \leq f \leq 1000$. Position is given in meters, orientation in radians, and focal length in pixels. We used 50,000 particles for all the particle filtering based conditions.

4.4.2 Results from Dimension Line Annotation

In this section, we present our experimental results in collecting 3D dimension line annotations in 2D videos using the crowd worker interface of Popup. We report the

rate of removed annotation sets and individually dropped annotations in the outlier filtering steps. We also report the rate of annotations dropped in the self-filtering step. Then, the average accuracy of dimension line annotations with and without our pre-processing filtering to drop poor annotations is reported. Lastly, we show the positive effect of self-filtering on latency in data collection.

Ratio of Filtering Annotation Sets

The first outlier filtering step removed low quality annotation sets based on bounding box coordinates, and filtered 7% of 850 submissions. We found that few incorrect objects (under 2%) remained after the filtering step, which typically occurred when the majority of workers (at least three out of five) annotated an incorrect object.

Ratio of Self-Filtering

After the first step of outlier filtering, 793 annotation sets remained in the collection. Each annotation set has three dimension entries (length, width, and height), resulting in a total 2379 entries submitted. Among the submissions, the number of self-filtered entries was 176, which were 7% of the total submitted dimension line entries. Of the self-filtered entries, 34% were filtered for the reason "The object is heavily occluded", and 66% were filtered for the reason "Others". There were no instances where the "I don't understand the instruction" option was chosen. When the "Others" option was chosen, workers could manually enter the reason behind their decision. Most explanations were related to insufficient visual information, e.g., "the object runs off the given image", "it's mostly back view", and "Bad angle, low resolution" as shown in Figure 4.9. We initially expected a higher self-filtering rate because we intentionally included scenes that are hard to annotate, e.g., truncated and occluded objects. In the discussion section, we discuss the potential reasons for the low self-filtering rate.

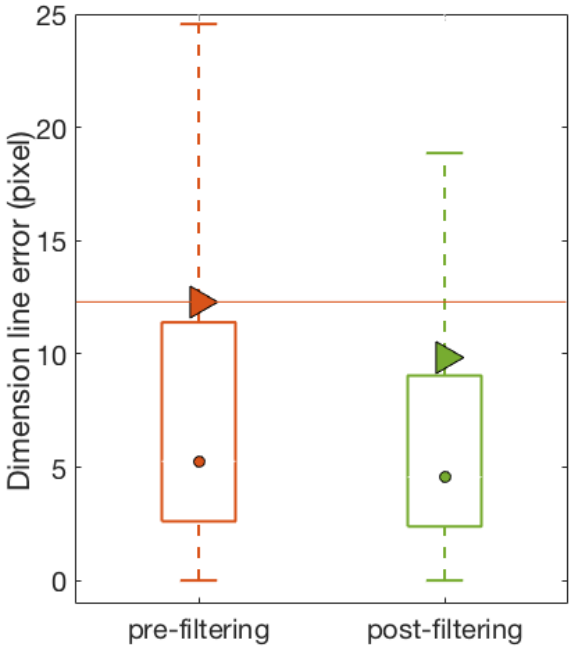


Figure 4.7: Height dimension line error of two different conditions (lower is better). The left is without any filtering, and the right is with both outlier and self-filtering. After filtering, the average error was reduced by 20% ($p < .05$). For each box plot, the circle denotes the median and the triangle denotes the mean. The lower and upper edges of boxes denote the 25-*th* and 75-*th* percentiles. The whiskers extend to the most extreme data-points not considered to be outliers.

Ratio of Filtering Individual Annotation

In the final outlier filtering step, we filtered individual annotations based on the dimension line's length and angular distance from the median. Of the individual annotations, 13% were considered outliers and filtered from the collection. We found that a few (under 3%) outlier annotations did not get filtered with our method. These were cases where the object was relatively small in the scene, and the variance within good annotations was very close to the variance between good and bad annotations.

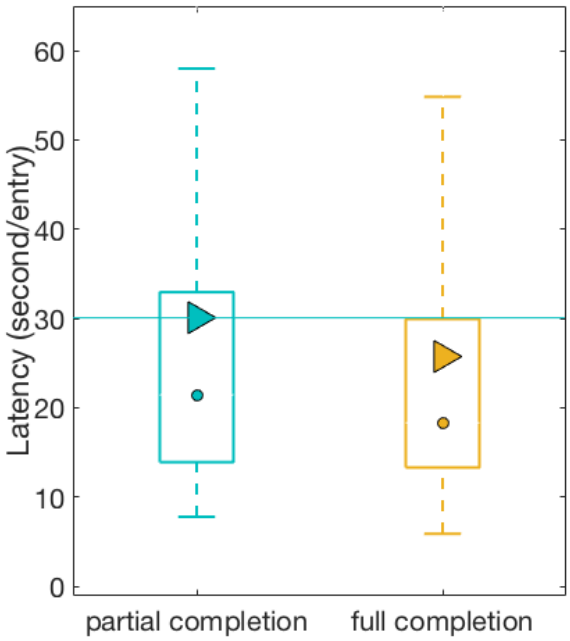


Figure 4.8: Average latency of partial and full annotation completion. The full completion represents typical entries – entries where no worker self-filtered. The partial completion represents entries that at least one worker self-filtered. The partial completion entries took an average of 16% more time to annotate ($p < .005$). For each box plot, the circle denotes the median and the triangle denotes the mean. The lower and upper edges of boxes denote the 25-*th* and 75-*th* percentiles. The whiskers extend to the most extreme data-points not considered to be outliers.

Accuracy of Dimension Line Annotations

We examined the effect of pre-processing filtering on the average accuracy of dimension line annotations. Since the dimension line ground truth is not provided by the KITTI dataset, we projected the actual vehicle height of the target vehicle onto the image plane, and compared the difference from the projected height line with the annotated dimension line in pixels. This analysis was not performed on width and length dimension lines as they are not parallel to the image plane. In our experiment, the distributions were all approximately normal, but with positive skew. Because the distributions were skewed, we computed p-values using Wilcoxon Rank-Sum test. As

shown in Figure 4.7, the filtering reduced the average error of dimension lines by 20% ($p < .05$) on average. Note that the mean error after filtering is under 10 pixels (9.8 pixels). Considering that the frame heights are 375-pixel, the average error is under 3% of the full height of a frame.

Time Savings from Self-Filtering

We investigated the average latency of partial and full annotation completion. Partial completion is defined as where at least one worker self-filtered, representing challenging entries. Full completion is defined where no worker self-filtered, representing typical entries. Because the distributions were skewed normal, we computed p-values using Wilcoxon Rank-Sum test. As shown in Figure 4.8, we found annotations took approximately 16% longer for partial completions ($p < .005$). The result suggests that two things: first, self-filtering can reflect a worker's confidence level as we intended in the design stage. we can reduce total latency in annotation collection if we encourage workers to self-filter the challenging entries, because they can save time on the drawing activity by skipping them. That is, we can save 16% of the annotation time for the hard annotations if the self-filtering option is provided.

4.4.3 Results from Aggregation and State Estimation

In this section, we evaluate our annotation aggregation and state estimation method under different conditions by comparing it against the ground truth from the KITTI dataset (*Geiger et al.*, 2012). For all evaluations, we dropped outliers; any data point outside 1.5×Interquartile Range (IQR) was removed for fair comparison between conditions.

Evaluation Metrics

For the evaluation of the accuracy of the state estimates, we used two metrics: a distance difference metric, and an angular difference metric. The distance difference metric is the Euclidean distance between the ground truth and the estimate. The angular difference metric corresponds to the smallest angular difference between estimated orientation and the ground truth orientation (Equation Eq. 1):

$$\begin{aligned} \text{DistanceDiff} &= \sqrt{(x_g - x_e)^2 + (y_g - y_e)^2 + (z_g - z_e)^2} \\ \text{AngularDiff} &= |(\theta_g - \theta_e) \ \% \ \pi/2| \end{aligned} \tag{Eq. 1}$$

where x_g, y_g, and z_g are the 3D ground truth, x_e, y_e, and z_e are the 3D state estimate, θ_g is the ground truth orientation, and θ_e is the orientation estimate.

Baseline and Conditions

To assess the success of the proposed inter-frame aggregation and 3D state estimation strategy, we compare the performance of our particle filtering based method with a state-of-the-art baseline method that uses geometric reprojection and L-BFGS-B (*Zhu et al.*, 1997) optimization. Note that the baseline does not refer to annotations in other frames. The two conditions that are compared to the baseline method are particle filtering without inter-frame referencing and particle filtering with inter-frame referencing. In addition, we look at the difference in window size for inter-frame referencing: using one, three, five, and 10 frames as window size.

- **Baseline:** The baseline method reprojects the 3D cuboid onto a given video frame and compares the corner location of the reprojection with the endpoints of the average dimension lines drawn for the target vehicle. This comparison is used as the cost function, and the L-BFGS-B optimization method (*Zhu*

et al., 1997) is used for minimization. The baseline method cannot handle cases where a whole entry (e.g., all height, length, or width annotations) is missing. The baseline method breaks in this condition because the optimization problem becomes underdetermined. Since the baseline method cannot refer to other frames' annotations by utilizing spatiotemporal constraints, the baseline was only run for single frame based estimation.

Video #1, frame #3: **4 out of 5 workers self-filtered 'length' entry.**
Reason: No side view.

Video #5, frame #10: **4 out of 5 workers self-filtered 'width' entry.**
Reason: Occluded.

Video #9, frame #2: **all 5 workers self-filtered 'length' entry.**
Reason: Low resolution.

Figure 4.9: Example of challenging frames where more than 3 out of 5 workers self-filtered. The cases include limited side view, occlusion, and low resolution.

- **C1. Particle filter without inter-frame referencing:** For a fair comparison with the baseline, this condition runs the particle filtering algorithm without referring to other frames. Since this condition does not refer to other frames, the solution is underdetermined under more aggressive filtering threshold. The comparison between the baseline and this condition emphasizes the effect of using inter-frame referencing which is enabled by particle filtering based annotation aggregation.

- **C2. Particle filtering with inter-frame referencing:** The strength of using particle filtering is that it refers to other frames to compensate for missing annotations. For this condition, we first evaluate the performance of different window sizes for inter-frame referencing: one, three, five, and 10. After evaluating the performance of different window sizes, we compare the performance of the best window size with the baseline performance.

Evaluation 1: Particle Filter Without Inter-Frame Referencing

To evaluate the performance of the particle filtering method, we compared the performance of condition C1 with the baseline. Figure 4.10(a) shows the position and orientation performance of the two conditions. In each graph, the left box plot shows the baseline condition applied to individual frames, and the right box plot shows condition C1: particle filtering applied to individual frames. Note that we present our results as a box and whisker plot through out this section, as the distributions were all approximately skewed normal with positive skew. Because they were skewed, we computed p-values using Wilcoxon Rank-Sum test. The summarized result shows that in terms of position estimation, the baseline and the proposed particle filtering method perform similarly (no significant difference was found). In terms of orientation estimation, we observed a 53% ($p = .11$) lower mean for our proposed particle filtering method compared to the baseline. However, while the effect size was medium-large

($d = 0.65$), the results were only approaching statistical significance ($p = 0.11$). We assume that the difference in performance comes from how the two methods incorporate dimension line evidence. The baseline method averages given dimension lines, and considers the average as an edge of the 3D re-projected cuboid to minimize the difference from the projection and the dimension lines. In contrast, the particle filtering based method does not compute an average, but compares each dimension line to the 3D re-projected cuboid to update the orientation estimation. This enables retaining information from all given dimension line annotations.

Overall, the result implies that without inter-frame referencing, our proposed method is comparable to the state-of-the-art baseline.

Evaluation 2: The effect of Inter-Frame Referencing

The primary strength of using particle filtering based estimation is in that we can refer to other frames' state estimates when estimating the current frame's state, which can fill in the missing information caused by either outlier-filtering or self-filtering. We looked at four different window sizes and window size of three had the lowest average error. There were no decrease in error was supported by enlarging the window size. Therefore, we performed statistical significant test of using window size three compared to not using inter-frame referencing. The summarized result in Figure 4.10(b) shows that referencing to three neighboring frames (including the current frame) results in 37% improved accuracy compared to not referring to neighboring frames in terms of position estimation ($p < .001$). However, orientation estimation accuracy did not improve by referring to neighboring frames.

Evaluation 3: Baseline vs. Proposed Method

To evaluate the performance of Popup, we compare the baseline result with Popup using three frame referencing—the best performing window size tested. Figure 4.10(c)

shows the position and orientation estimation results. In terms of position estimation, the average error was reduced by 28% ($p < .001$). In terms of orientation estimation, the average error was reduced by 54%, but the results were not significant ($p = .105$). The result shows that the proposed aggregation and estimation strategy for crowdsourcing image annotations in videos can handle noisy and incomplete annotation sets, and also outperform the baseline condition in terms of position estimation.

4.5 Discussion

In this section, we discuss about the effect of inter-frame referencing in video annotation, factors that can affect workers' self-filtering behavior, and factors affecting the final 3D estimation quality.

4.5.1 Inter-Frame Referencing in Video Annotation

The evaluation results show that referencing annotations from neighboring frames can increase estimation accuracy in video annotation tasks. We tested four different window sizes, and the window size affected the impact of inter-frame referencing. For position estimation, a larger window size seemed to improve the estimate accuracy, but the effect was not linear and maximum at the frame window size of three. For orientation estimation, the performance was consistent from window size one to five. The performance largely degraded for the 10-frame window. We speculate that the reason we did not observe progressive improvement in accuracy with increased window size is because of propagation of bad annotations. If one frame is poorly annotated, particularly early in the video, it will affect all other frames within the window. It follows that a larger window size allows local errors to affect more frames, which results in a larger aggregated overall error. For example, a critical error in frame k will affect only frame $k-1$ and $k+1$ in a three-frame window, but will affect all 10 frames in a 10-frame window.

4.5.2 Factors Affecting Self-Filtering

We believe that the self-filtering rate and accuracy can vary by various factors: the quality of instructions, the complexity of the task, and the mechanism of the crowdsourcing platform. The instruction and task can be designed to help workers clearly understand the benefit of self-filtering. Shah et al. (*Shah and Zhou*, 2015) gave a clear incentive plan to the workers, which encouraged them to use the self-filtering option ('I'm not sure' option) wisely. This setting resulted in the highest data quality in their experiment. We also believe that the mechanism of the platform affects workers' behavior on using the self-filtering option. In our post survey, we asked workers who completed our task if they think providing any answer is better than no answer at all when they are not confident enough to provide an answer. One worker answered, "I think an attempt at an answer is better than none at all. Even if you aren't sure an attempt at least shows your [*sic*] trying to help the study and not just wasting everyone's time". Another answered, "Try my level best to satisfy the requester". This implies that workers are reluctant to choose such an option due to the default incentive mechanism. Therefore, the requester should clearly design an incentive mechanism and mention in the task that how they would like workers to use the self-filtering option.

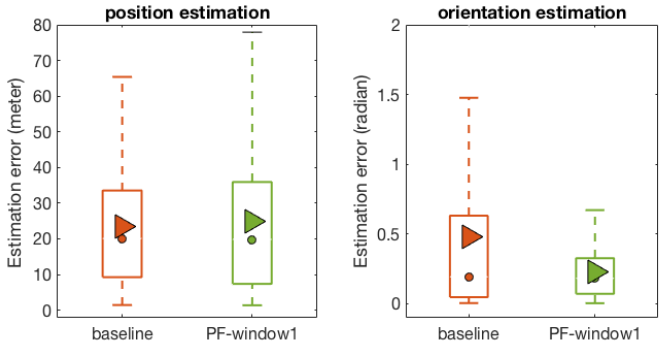
4.5.3 Other Factors Affecting State Estimation Accuracy

We also note that the performance of the optimization method can affect the 3D estimation results. For example, the parameters — such as the search bounds in solution space or number of iterations — should be carefully chosen to best fit the targeted source's profile. Moreover, the type of optimization model can affect the estimation result. Not all optimization methods would solve our problem in general because the parameters to be estimated do not have a linear relationship. Therefore, the optimization model should also be carefully chosen based on the characteristics of

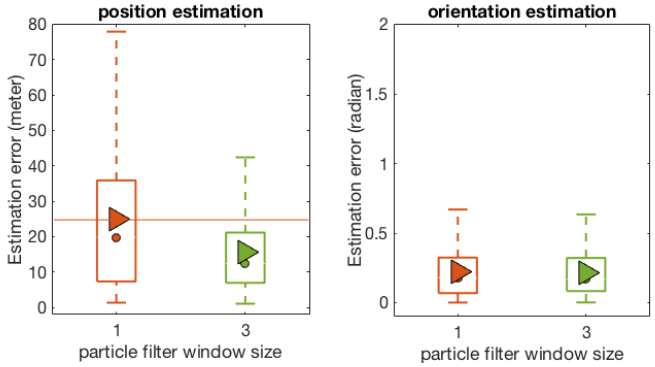
the problem to be solved. Another factor that affects 3D state estimation accuracy is the consensus between workers and the filtering methods. If the majority of workers provide wrong annotations, in the current setting, we cannot detect or filter them with the proposed filtering strategies. For example, we had a frame where three out of five workers annotated the wrong vehicle, so we failed to filter out the majority answers, even though they were incorrect. To avoid these cases, a human-in-the-loop step to check the quality of the bounding box or the annotation could be added.

4.6 Summary

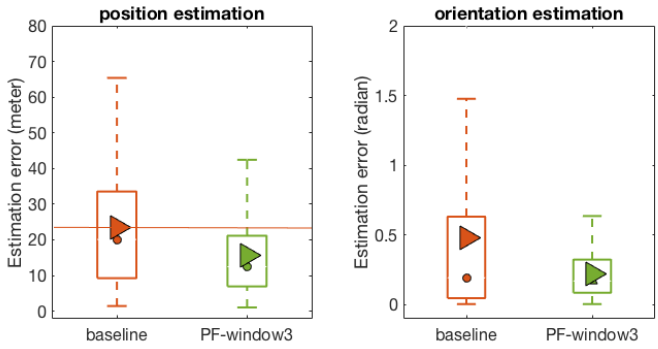
In this chapter, we have introduced a new crowdsourcing approach to collecting and aggregating image annotations in videos more efficiently and accurately. Our approach leverages particle filtering to aggregate annotations from multiple frames and provide an accurate final output even in the presence of incomplete or missing annotations. We introduced Popup, a human-machine hybrid system that realizes the proposed methods. The study results show that the proposed particle filtering based aggregation can not only handle noisy and missing annotations, but also provides more accurate 3D state estimations of objects in 2D videos. Because the proposed method is robust to missing annotations, one can reduce the overall latency of the data collection stage by allowing the annotators to self-filter. The output from Popup can be passed to simulation software to enable generating a realistic and large 3D training dataset of rare events for autonomous vehicles and machines to learn.



(a) Performance of baseline vs. C1



(b) Performance of particle filtering without- vs. with inter-frame referencing



(c) Performance of baseline vs. particle filtering with inter-frame referencing

Figure 4.10: (a) Without inter-frame referencing, the particle filter's performance is comparable to the baseline. (b) Inter-frame referencing reduced error significantly. Window size 1 indicates without inter-frame referecning. (c) Our proposed inter-frame referencing particle filtering method outperforms the baseline. For each box plot, the circle denotes the median and the triangle denotes the mean. The lower and upper edges of boxes denote the 25-*th* and 75-*th* percentiles. The whiskers extend to the most extreme-most data points that are not considered to be outliers.

CHAPTER V

Knowledge Diversity: Improving Accuracy of 3D Object Reconstruction via Crowdsourced Joint Object Estimation

The previous two chapters introduced the approach of leveraging tool diversity and perspective diversity as a means to reduce systematic biases that are due to the tools being used or the data instances given, respectively. In this chapter, we introduce the idea of leveraging the knowledge diversity of crowd workers as a means to overcome uncertainties caused by a lack of information.

5.1 Motivation

Extracting precise 3D spatial information from existing abundant 2D datasets to create high quality 3D training data is a grand challenge in computer vision (*Chen et al.*, 2019, 2016; *Konrad et al.*, 2012) due to its potential impact on facilitating real-world applications, such as self-driving vehicles (*Pan et al.*, 2017; *Ramakrishnan et al.*, 2019), interactive assistive robots (*James and Johns*, 2016; *Sorokin et al.*, 2010), or augmented and virtual reality (*Orts-Escolano et al.*, 2016; *Sankar and Seitz*, 2017). This conversion of 2D to 3D typically involves collecting manual annotations, where people provide computers with the necessary information to bridge the gap between

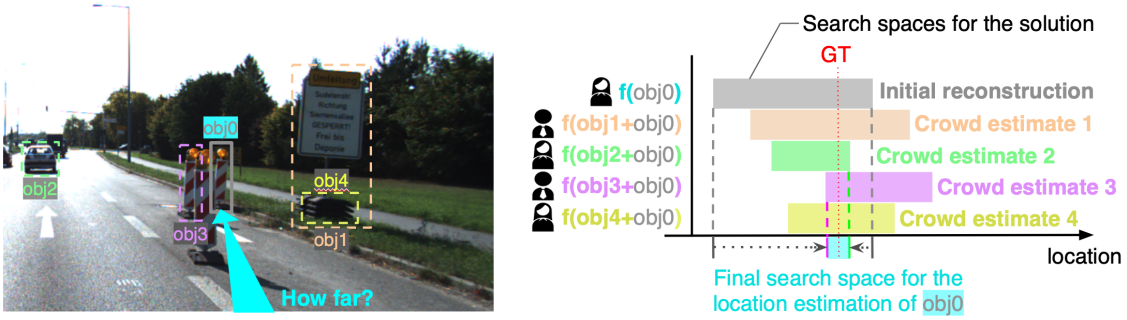


Figure 5.1: This chapter introduces an approach to crowd-powered estimation of the 3D location of a target object (here, `obj0`) by jointly leveraging approximate spatial relationships among other in-scene objects (`obj1`–`obj4`). Our approach lets crowd workers provide approximate measurements of familiar objects to improve collective performance via our novel annotation aggregation technique, which uses the spatial dependencies between objects as soft constraints that help guide an optimizer to a more accurate 3D location estimate.

2D and 3D, e.g., pixel-level indications of where the edges of an object are. To collect these annotations at scale, crowdsourcing is typically used due to its convenience in prompt and flexible recruiting (*Dai et al.*, 2017; *Bell et al.*, 2013).

In settings where there is insufficient information about the camera and/or scene to deterministically compute 3D information based on 2D annotations, a computational process such as parameter estimation with iterative optimization is performed to search for a solution based on the annotations (*Lowe*, 1991; *Lu et al.*, 2000; *Leng and Sun*, 2009; *Cao et al.*, 2011). However, there are factors that make the optimization process challenging to find an accurate solution, such as the search area for the solution being large. Moreover, the annotations are usually noisy, having limitations due to the finite resolution of the image, which limits subpixel-level information, and restricted perceptual and motor resolution of human annotators, which limits the annotation's preciseness (*Song et al.*, 2019b).

In this chapter we focus on the task of estimating the 6D pose (3D location and orientation) of a particular *target* object. Our key insight is that the joint use

of multiple in-scene objects enables more accurate estimation (that can go beyond the limits of pixel resolution) while providing a means of leveraging more diverse knowledge from the crowd. Our approach converts approximate judgments about the *reference* objects that are near the target object into soft constraints for an optimization algorithm that recovers the state estimation of the target object. As shown in Figure 5.1, the soft constraints for the optimizer penalize unlikely solutions and improve the chances of finding more accurate ones. By relaxing the precision requirements for usable annotations, our approach also allows crowd workers with diverse types and levels of knowledge to contribute to the system performance.

To understand the potential performance benefits of the proposed approach, we conducted a characterization study with a controlled experiment using a synthesized virtual dataset with absolute ground truth data. Based on the controlled study results, we developed C-Reference, a crowd-powered system that reconstructs the 3D location of a target object based on manual annotations of the target and reference objects. To demonstrate the effectiveness of our approach, we recruited 339 crowd workers from Amazon Mechanical Turk to annotate 15 realistic computer-generated images of indoor and outdoor scenes. The end-to-end experimental results, from annotation collection to 3D location estimation, show that our proposed approach significantly reduces the average 3D location estimation error by 40% with only 35% as much human time. This chapter makes the following contributions:

- We introduce a novel crowdsourcing approach that strategically elicits and leverages the diverse knowledge of crowd workers to improve collective accuracy even in settings where individuals do not have full knowledge of the task.

- We present a characterization study to demonstrate the effectiveness of our proposed approach via a controlled experiment with a large synthetic dataset.

- We create C-Reference, a system that implements our proposed multi-object

aggregation method to more accurately reconstruct 3D scenes from 2D images.

- We report experimental results from a study using C-Reference that demonstrates our proposed approach can more efficiently and accurately estimate the 3D location of a target object compared to using single-object annotations.

5.2 Evaluation Method

To add clarity for the remainder of the chapter, we begin by describing the dataset that we use for all of our empirical results, as well as our metrics of success.

5.2.1 Dataset

To evaluate our approach, we need a 2D image dataset with corresponding 3D ground truth answers. Existing datasets can be noisy due to the range-fidelity of LiDAR sensors and the errors made during the manual annotation of 3D bounding boxes around point cloud objects (*Chen et al.*, 2014). To avoid these errors, we synthesize a 3D dataset using the Unity 3D game engine (example in Figure 5.2(a)). The image resolution was 2880×1800 pixels. For each image, we select one *target object* at random to be estimated. To demonstrate the robustness of our approach, we include objects that are small, heavily occluded (more than 50%), or have a limited view angle from the camera's perspective. Next, we arbitrarily choose five *reference objects* for each test image. We assume that no information is known a priori about these reference objects.

5.2.2 Metrics

To assess the quality of the intermediate and final output of our system, we use percent error to represent deviation from the ground truth value. If the output is a range, we also measure precision.

Percent Error

If the output is a scalar value, we compute the percent error as follows:

$$\text{err}_s = \frac{|\tilde{m} - \text{GT}|}{\text{GT}} \times 100 \qquad \text{(Eq. 1)}$$

where $\tilde{m}$ is the estimate value, GT is the ground truth, and $|\cdot|$ is absolute value.

When the output value is a range with scalar valued bounds, we use a slightly modified formulation as follows:

$$\text{err} = \frac{|(\tilde{m}_\text{L} + \tilde{m}_\text{U})/2 - \text{GT}|}{\text{GT}} \times 100 \qquad \text{(Eq. 2)}$$

where $\tilde{m}_\text{L}$ is the lower bound of the output range and $\tilde{m}_\text{U}$ is the upper bound of it, which measures the percent error.

If the output is a vector value, we compute the percent error as follows:

$$\text{err}_v = \frac{||\bar{\boldsymbol{m}} - \mathbf{GT}||_2}{||\mathbf{GT}||_2} \times 100 \qquad \text{(Eq. 3)}$$

where $\bar{\boldsymbol{m}}$ denotes the estimated 3D location vector, $\mathbf{GT}$ denotes the ground truth 3D location vector, and $||\cdot||_2$ denotes Euclidean distance.

Precision

When the output is in a range, the precision of the range is computed as follows:

$$\text{precision} = \frac{(\tilde{m}_\text{L} - \tilde{m}_\text{U})}{(\tilde{m}_\text{L} + \tilde{m}_\text{U})/2} \times 100 \qquad \text{(Eq. 4)}$$

where $\tilde{m}_\text{L}$ and $\tilde{m}_\text{U}$ are the lower and upper bounds of the range, respectively. Note that for both percent error and precision, lower value means better performance.

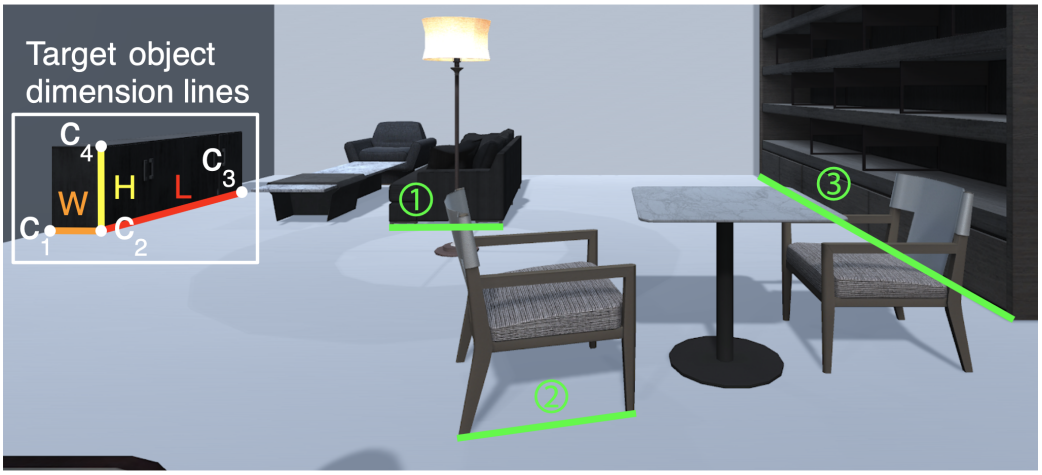


Figure 5.2: A test image with known ground truth of objects. Inside the white bounding box is the target object (a cupboard) to be estimated. The three colored lines on the object represent the ground truth dimension lines, length (L), width (W), and height (H). Green lines (①,②, and ③) are the reference object annotations.

5.3 C-Reference: Joint Object 3D Location Estimation

In this section, we introduce our proposed joint object estimation system, C-Reference, which estimates the 3D location of a target object using diverse sets of 2D annotations. Our approach transforms the approximate size or distance measurement annotations of multiple in-scene objects to soft constraints that are then used by an optimizer, making any level of measurement granularity useful. This enables C-Reference to leverage answers from workers with diverse levels of knowledge, collectively generating accurate system output.

5.3.1 Iterative Optimize to Estimate the 3D Location of a Target Object

For the 3D location estimation, we build upon the method from Popup (*Song et al.*, 2019b), which estimates the 3D state of a target object using the three "dimension line" (length, width, and height of an object) annotations drawn on 2D images. Dimension lines provide richer information compared to other annotation methods. Specifically, they can be used to determine both the 3D location and orientation

information of an object, while keypoint annotation (*Szeto and Corso*, 2017) can only provide orientation information, and 2D bounding boxes (*Geiger et al.*, 2011) can only provide location information.

The three dimension lines, as shown inside the white box in Figure 5.2, create four corners (c_1, c_2, c_3, and c_4) that enable to concretize the problem of 3D location estimation to a perspective-n-point problem (*Fischler and Bolles*, 1981; *Lepetit et al.*, 2009), where the intrinsic camera parameters are unknown and the manual annotations are noisy. Since we use the same scenario as in Popup (*Song et al.*, 2019b), the true size measurement of the target object, i.e., the actual 3D distance between the four corners, is assumed to be known.

Designing the Cost Function

We estimate five unknown variables, x, y, z, θ, and f, using the four corners of dimension line annotations of a target object, where x, y, z are the 3D location of the target object, θ is the yaw orientation of the object, and f is the camera's focal length. The corners are input to an iterative optimizer, which we implemented based on the L-BFGS-B algorithm (*Zhu et al.*, 1997; *Song et al.*, 2019b). We applied a basin-hopping technique to iterate multiple times with random initialization, only accepting a new solution when its cost is minimum among all the candidate solutions visited during the iteration. The objective function is designed as follows:

$$\text{cost} = \sum_{i=1}^{4} \left(-\log\text{Normal}(||\boldsymbol{t}_i - C_f(\boldsymbol{x}_s)_i||_2, 0, \sigma) + \big|||\boldsymbol{t}_i||_2 - ||C_f(\boldsymbol{x}_s)_i||_2\big| \right) \quad \text{(Eq. 5)}$$

where the optimizer finds $\bar{s} = \underset{\{s \in S, f\}}{\text{argmin}}(\text{cost})$. Here, $\bar{s}$ denotes the estimated 3D location, S denotes all valid state candidates, $\boldsymbol{t}_i$ denotes one of the four corners from a set of dimension lines, $\text{logNormal}(\cdot, 0, \sigma)$ denotes log-normal distribution with standard deviation σ, $C_f(\cdot)$ denotes the camera projection matrix, and $\boldsymbol{x}_s$ denotes the current

state of the virtual 3D bounding box of the target object. Lastly, $||\cdot||_2$ denotes the Euclidean distance and $|\cdot|$ denotes absolute value. Note that we did not use the particle filter-based method proposed in Popup (*Song et al.*, 2019b) because it is not applicable to static images.

Limitations

While the optimizer is designed to find the minimum cost of the objective function, there are a few factors that can cause poor estimation results. First, annotation noise in the dimension lines can affect the estimation result. Even a small discrepancy (e.g., smaller than five-pixel error in 2D) in annotation accuracy can lead to a significantly amplified error (e.g., larger than 20-meter error in 3D) in the estimation result depending on the camera parameters (*Song et al.*, 2019b). Unfortunately, collecting super-precise visual annotations on 2D images is challenging because of the factors such as limited human motor precision, limited pixel resolution, and limited human visual perception to precisely perceive scenes. Second, while the optimizer can find the global optimum in cases where input annotations have zero noise and the initial values are set at the global optimum, in other cases it fails to find the global optimum, instead finding local optima as a solution, making the performance highly dependent on the initial values chosen. To resolve the local minimum problem, more information such as additional constraints for the search area can be added to the optimization process to help avoid searching near infeasible solutions. In the next section, we introduce a novel annotation aggregation approach that helps the optimizer overcome these limitations by generating additional soft constraints from even rough and less precise annotations.

5.3.2 Joint Object Annotation Aggregation

To achieve better estimations that overcome limitations from pixels and local optimum, we introduce an approach that guides to better search area using soft constraints that are generated from diverse crowd annotations on multiple objects. The key insight is that having multiple objects as an option will allow crowd workers to use their diverse knowledge of familiar objects, not having to rely on a single source of information when annotating. To combine the heterogeneous annotations from different objects, we first introduce a penalty function that is generated by merging multiple soft constraints. The advantage of converting the annotations into soft constraints is that the accuracy requirement for usable annotations can be relaxed, allowing even rough approximate annotations to contribute to the system performance. Next, we introduce an aggregation method that uses the shared spatial relationship among the objects to unify and transform the annotations into useful input for the penalty function.

Designing a Penalty Function

We introduce a penalty function that creates a soft constraint for the optimizer using approximate search bounds. The soft constraints penalize the optimizer for selecting an infeasible solution and encourage finding a better solution near the ground truth. We design a penalty function based on a weighted sigmoid function which penalizes x if x is below l or above u as follows:

$$P(x) = S(l - x) + S(x - u) \qquad \text{(Eq. 6)}$$

where $S(x)$ is the weighted sigmoid function with a weight $x + 1$,

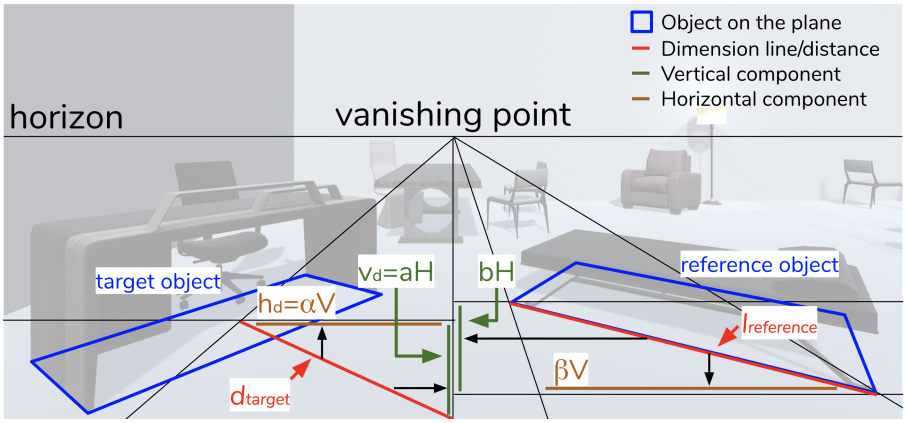
$$S(x) = max\left(\frac{x+1}{(1+e^{-ax})}, M\right) \qquad \text{(Eq. 7)}$$

and l is a lower bound, u is an upper bound, a is the sharpness parameter, and M is a threshold to prevent the penalty function dominating the objective function. While the two parameters a and M can be arbitrarily tuned, the additional information of l and u is best chosen to be near the ground truth so that the penalty function can narrow down the search area for the optimizer. In the next section, we design a joint object annotation aggregation method, which aggregates approximate annotations from diverse different reference objects—the rest of the in-scene objects besides the target object to be estimated—to obtain reasonable values for both l and u.

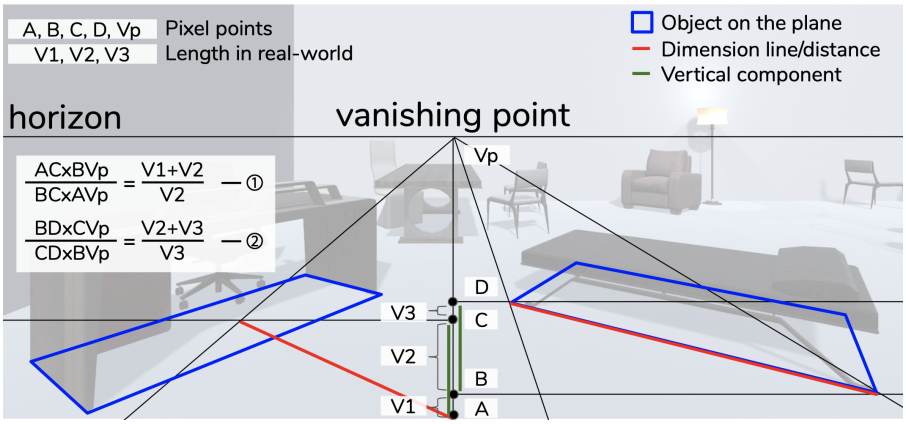
Annotation Aggregation Method

The proposed aggregation approach uses (i) a measurement value of a line in an image and (ii) the 2D annotation of the line marked on the image as input. The method utilizes the spatial relationships between the reference objects, the target object, and the ground plane they are on, to transform the 3D properties of the reference objects into useful information for the optimizer. We assume that both the reference objects and target object share the same ground plane and vanishing points. This setting enables to make use of the 3D affine measurements even with single perspective view of a 2D scene, given only minimal geometric information (e.g., vanishing points and horizontal line) from the image (*Criminisi et al.*, 2000).

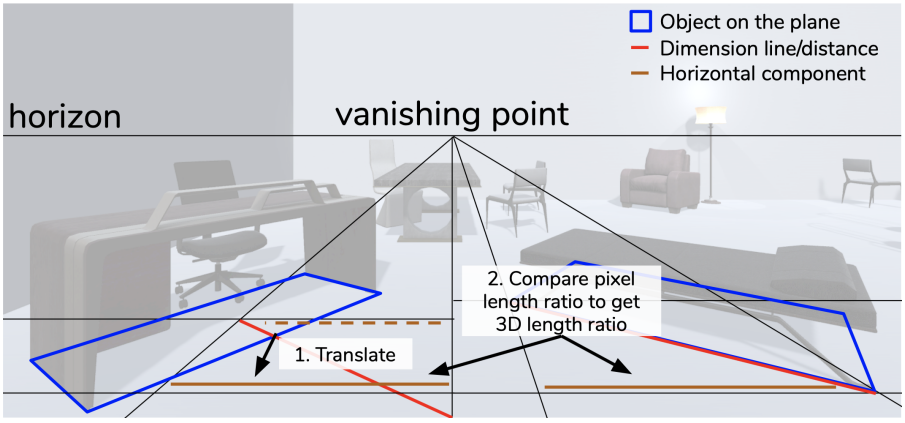
Our goal with joint object annotation aggregation is to approximate the position of the target object relative to the camera position. Specifically, we seek how far the target object is from the camera in the vertical and horizontal direction that the camera is looking at, which are denoted as v_d and h_d, respectively. To obtain these values, we use the following four pieces of information: (i) the center of the target object's bottom, which is computed from the target object dimension lines, (ii) a line on two reference objects each, which represents the length, width, or the distance from the camera, (iii) the approximate estimate of the lines' actual measurements,



(a) Aggregation Step 1: decompose annotation lines into vertical (along the vanishing point) and horizontal (parallel to the horizon) components.



(b) Aggregation Step 2: find the four points along the vertical line from camera center to a vanishing point to conduct the cross-ratio based computation, as in ① and ②.



(c) Aggregation Step 3: use the cross-ratio based computations (① and ② in (b)) to map the components' 3D length with the crowd workers' measurement estimation annotations.

Figure 5.3: Step-by-step aggregation of reference object annotations using cross-ratio and vanishing points.

and (iv) the horizontal line obtained from the vanishing points. The horizontal line could be estimated using off-the-shelf computer vision algorithms (*Denis et al.*, 2008; *Tardif*, 2009) or it could be obtained manually.

Figure 5.3 shows an example of how we compute the approximate distance of a target object. For the length, width, or distance line of a reference object, we first decomposed them into vertical and horizontal components as in Figure 5.3(a). Decomposition also disentangles the complex relation between horizontal and vertical components as they have different projection characteristics. After decomposition, for each component, we calculate the length ratio between the target object distance and each reference object's line. The length of each component is denoted by multiplying the ratio and the unit lengths (V and H) for the vertical component and horizontal component, respectively. Using the length ratio, we can set the following equation with one reference object line annotation:

$$(d_{target})^2 = (aH)^2 + (\alpha V)^2 \qquad \text{(Eq. 8)}$$

$$(l_{reference})^2 = (bH)^2 + (\beta V)^2 \qquad \text{(Eq. 9)}$$

where d_{target} is the target object's distance from the camera, l_{ref} is the length of the reference object annotation line, a and b are constants calculated from the ratio of the horizontal component length, and α and β are constants calculated from the ratio of the vertical component length. V and H are unit lengths for vertical and horizontal components. Here, d_{target}, V, and H are the three unknown variables. With two reference object annotations, we can come up with one equation for the target object (Eq. Eq. 8) and two equations for reference annotations (Eq. Eq. 9), which share three unknown variables and can solve the equation problem for these variables.

To obtain the ratio of vertical components between two lines, we used cross-ratio, which is a ratio relationship between four points on the same straight line as in

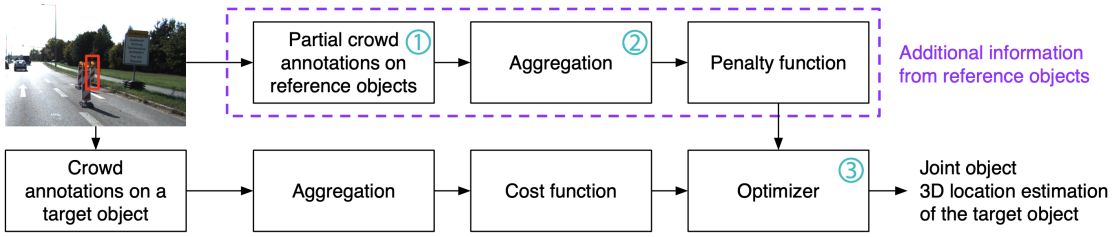


Figure 5.4: Overview of the pipeline of our prototype application, C-Reference, which estimates the 3D location of a target object using a novel joint object estimation approach. The additional information from the joint object annotations (①) is aggregated (②) and transformed into a soft penalty function (③), allowing diverse granularity of approximate annotations to contribute to improving the system performance.

Figure 5.3(b). Because cross-ratio has a projective invariant property, where the ratio in projected pixels and lengths in the 3D space are the same, we can use it to compute the length ratio of lines in the 3D space.

For the ratio of horizontal components between two lines, we first translate the decomposed horizontal component so that they can be on the same straight line parallel to the horizon as in Figure 5.3(c). The reason for this is that horizontal components can be compared as the 3D length only when they are at the same vertical distance from the camera. After the translation, we compute the length ratio of the two horizontal components with the ratio of the pixel length. When the annotations are poor with large noise, the system of equations will have no solutions, which can be used to filter out cases with no solution from being input to the penalty function.

5.3.3 System Implementation

Based on the proposed joint object annotation aggregation method, we implemented C-Reference, a crowd-powered 3D location estimation system that leverages approximate annotations from the reference objects to more accurately estimate the 3D location of a target object. Figure 5.4 shows the system pipeline of C-Reference.

The penalty function we designed (Eq. 6) is integrated into the objective function (Eq. 5) as follows:

$$\text{cost}' = \text{cost} + \sum_{i,j \in R} P_{ij}(\boldsymbol{x}_s) \qquad \text{(Eq. 10)}$$

where cost is the objective function in Eq. 5, R is a set of reference object annotations,

$$P_{ij}(\boldsymbol{x}_s) = S(d_{ij,l} - \boldsymbol{x}_s) + S(\boldsymbol{x}_s - d_{ij,u}) \qquad \text{(Eq. 11)}$$

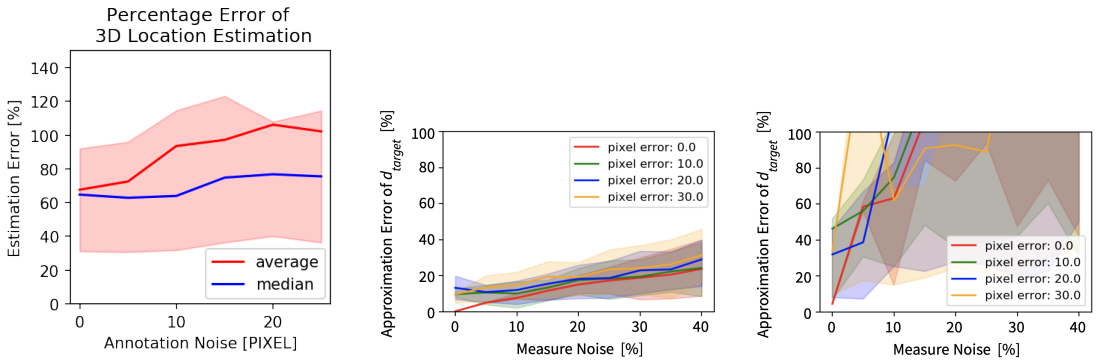
where $S(\cdot)$ is the weighted sigmoid function described in Eq. 7, d_{ij} is d_{target} approximated from reference object annotations i and j. Finally, l and u are the lower and upper bounds of the approximation of d_{ij}. Because the penalty function can accept lower and upper bounds, it is possible to leverage the approximate measurement of objects at any level of granularity from the annotators. Also, since our approach can leverage any line drawn on the ground in the image, it can be mapped to various different types of knowledge from crowd workers.

5.3.4 Controlled Study of Simulated Annotation Error

To systematically understand the performance of our approach, we conducted two control studies with computer-generated virtual data points with absolute ground truths. We first investigate the performance of module ③ in Figure 5.4 without the additional information from the penalty function. Next, we investigate the performance of our joint object aggregation method (module ② in Figure 5.4) which generates the input to the penalty function.

Performance of the Optimizer without the Penalty Function

We ran a controlled study with 748 virtual data points with varying annotation noise on each data point. The amplitude of the noise varied from zero pixel to 25 pixels in random direction, which we generated with five-pixel intervals. The annotation



(a) Using target object alone. (b) Using line 1 and 3. (c) Using line 1 and 2.

Figure 5.5: Results of the controlled studies. (a) Performance characterization of the optimizer *without* our annotation-derived penalty function. The result shows the error characterization result of 748 data points where each point was generated with zero to 25 pixels noise (five pixels interval) in random direction for each corner, c_1, c_2, c_3, and c_4. Shaded area is the interquartile range.The result shows that a noise floor of about 70% error in 3D location estimation is generated even with zero pixel annotation noise. This noise floor is reduced to zero when ground truth is set as initial values. (b) Performance characterization of our proposed joint object annotation aggregation method. The aggregation result approximates d_{target} in Eq. 8. The result shows the average error of aggregating line ① and line ③ in Figure 5.2. While the approximation error of d_{target} consistently increases according to both pixel and measurement noises, the error can be reduced to zero if no noise is added to the annotations. (c) The result shows the average error of aggregating line ① and line ② in Figure 5.2. Because the two parallel lines create a degenerate configuration with no unique solution, the approximation error becomes very noisy. Shaded areas indicate the interquartile range.

noises were added to the corners of each dimension line of the target object, L, W, and H in Figure 5.2. Then, each data point was input to the optimizer (Eq. 5) to compute the 3D location estimation of the target object. The σ value was set as 100, the basin-hopping iteration number as 10, optimizer stopping criteria as 1e-8, and the optimizer bound as $-10 \leq x \leq 10$, $-10 \leq y \leq 10$, $1 \leq z \leq 100$, $-\pi \leq \theta < \pi$, and $100 \leq f \leq 2000$. After the estimation was finished, we computed the percent error of the 3D location estimation of each data point as in Section 5.2 Eq. 3.

Figure 5.5(a) shows the controlled study result using the target object's three

dimension lines to estimate the object's 3D location. The result shows that there is about 70% noise floor in 3D location estimation even when the annotation noise is zero pixels. The average estimation error increased as the annotation noise increases. We observed that the noise floor is affected by the initial values selected since we got zero 3D location error when the initial values were the ground truth. This may indicate that the optimizer is converging to a local minimum, which depends on the randomly selected initial values. Therefore, using an additional penalty function that penalizes infeasible solutions would help avoid local minima and find a better solution (*Smith et al.*, 1997). In summary, this controlled study shows that the baseline optimization is vulnerable to both annotation noise and the initial search area.

Performance of the Propose Joint Object Annotation Aggregation

The design of our penalty function and annotation aggregation algorithm is robust to annotation noise because of two reasons; (i) the threshold parameter M in Eq. 7 prevents the penalty function from creating large basins that can dominate and confuse the objective function, and (ii) the poor annotations can be automatically filtered because the approximation value d_{target} in Eq. 8 will have no solution when the annotation pairs are low quality. The cases with no solution can be detected and filtered as a post processing of the aggregation.

Beyond the mathematical observation, we ran a similar controlled study as in Section 3.1.1 to better understand the performance of our proposed joint object annotation aggregation method. The ground truth lines in the unit of pixel and the ground truth measurement of the ground truth lines were known. We generated 748 virtual data points with varying noise from zero to 30 pixels in random direction, generated in 10 pixels interval. The image resolution was 2880 × 1800 pixels. The noise was added to the target object dimension lines and three reference object annotation lines, ①, ②, and ③, as shown in Figure 5.2. We also added the measurement

noise to these lines, from 0 percent noise to 40 percent noise compared to the ground truth, generated in 10 percent intervals.

Figure 5.5(b) and (c) show two example results from the controlled study. Figure 5.5(b) shows the average result using line ① and line ③ in Figure 5.2 to approximate d_{target} value. It is observed that the approximation error increases according to both the pixel noise and the measurement noise. While the error characterization of the optimizer from Figure 5.5(a) shows a noise floor of about 70% even with zero pixel noise, our joint object annotation aggregation method shows that the noise floor can be reduced to zero if there is zero noise in the reference object annotations. Even with 40% measurement noise and 30 pixel annotation line noise, the error in approximating d_{target} was below 30% on average. Figure 5.5(c) shows the average result using line ① and line ② in Figure 5.2 to approximate d_{target} value. A characteristic of these two lines is that they are parallel in 3D. These parallel lines create degenerated configuration for the system of equations, resulting in no unique solutions to output. Even though there should not be a unique solution, the synthesized noises added to these lines create large errors due to the lines intersecting at some point where the distance to the intersection could be very large due to the lines being near parallel.

Our controlled study demonstrates how the joint object annotation aggregation method fails to provide feasible penalization to the optimizer when the line pairs are parallel. However, the impact of these parallel lines can be minimized as more reference object annotation lines are combined, because the design of the penalty function (Eq. 11) makes the effect of these outliers negligible.

5.4 Eliciting Measurement Estimates

With our joint object estimation method in mind, we implemented a web interface that elicits measurement estimates from the the crowd (Figure 5.6). The interface also asks the annotators to mark the corresponding length lines on the reference objects,

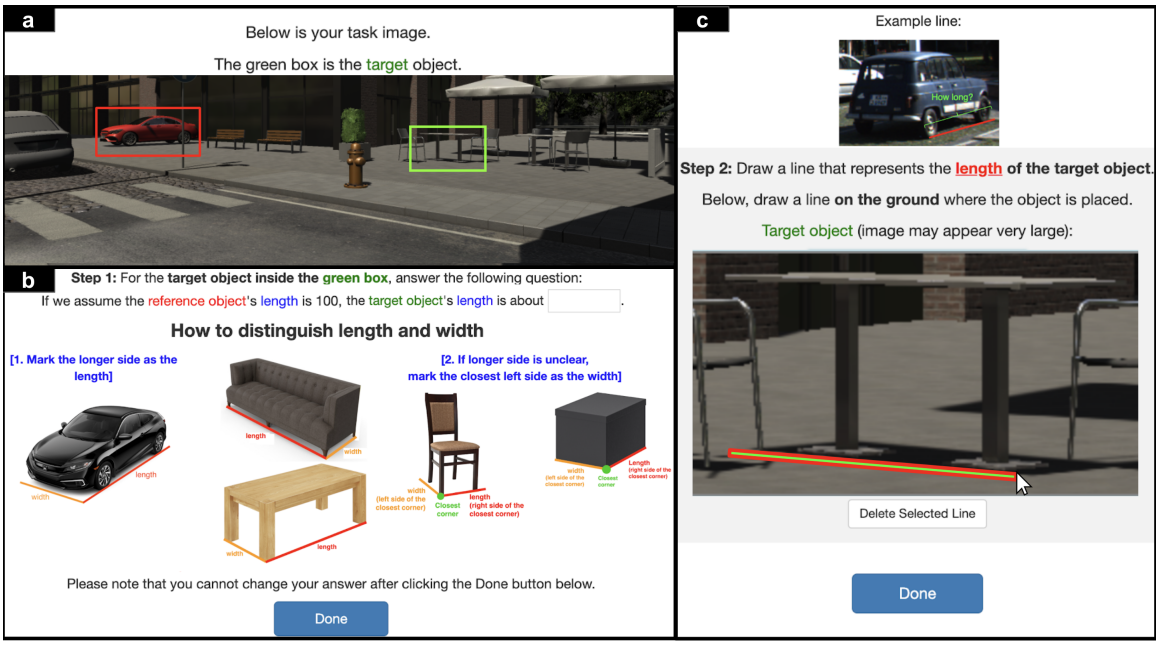


Figure 5.6: The interactive worker UI is comprised of three steps in which workers approximate object measurements and annotate dimension lines. (a) The instructions and task image step: the reference object to be annotated is marked with a green box. If the *relative* condition is assigned, the UI also provides an indication of the target object (red box). (b) The measurement-approximation step: each worker sees different instructions based on the condition they are assigned. (c) Length line annotation step: crowd workers were instructed to draw the line that represents the measurement they provided in the second step.

as shown in Figure 5.6(c). We explore diverse types of measurement estimates to understand tradeoffs in crowd worker performance (accuracy and precision) when generating different measurement estimates.

5.4.1 Types of Measurement Estimate Annotation

Providing various options to annotate measurement estimations helps facilitate the use of varied knowledge from crowd workers. While measurement estimates of reference objects can be asked and can be answered in various forms, we explored three different dimensions in designing the type for the measurement estimates: selection (length of an object, width of an object, or distance of an object from the camera),

directness (direct measurement or relative measurement compared to another object), and granularity (single valued answer or range valued answer). These types are orthogonal and thus can be combined (e.g., annotating the *length* of an object via a *range* that is *relative* to a different object's length).

Selection of Measures

While our annotation aggregation method can make use of any line drawn on the ground and its corresponding actual measurement value, it is hard for humans to estimate the actual measurement value of a line if there is no visual reference. Therefore, we asked workers to annotate lines that have visual object reference points in the scene. The length and width of objects can be estimated based on prior exposure to and knowledge about everyday objects, e.g., the length of a table is usually greater than that of a chair. We did not include height measurements of objects since they cannot be drawn on the ground. The object's distance from the camera can be inferred based on the scene in a given image, e.g., if there are three cars in a row, one can tell approximately how far the last car is from the camera. Example measurement estimates are:

- *Length*: "The object is about 165 inches long."
- *Width*: "The object is about 50 inches wide."
- *Distance*: "The object is about 35 feet away."

Note that because the camera location is not visible from the image, instead we designed the interface to ask crowd workers to consider the distance "from the bottom of the image". In the instructions, we provided example gif images that demonstrate how to draw length lines to help the workers understand the task.

Directness of Measures

Depending on the context, sometimes it can be easier to estimate a relative measurement than the direct measurement of an object. This is especially true when the object is not familiar to the annotator, because people naturally use prior knowledge of other objects to infer the properties of a new object (*Sternberg and Sternberg*, 2016; *Heit*, 1994). Therefore, we implemented not only an interface to input direct measurements, but also the relative measurements of objects. However, if we let crowd workers make the inference based on any object in the task image, it becomes hard for the computer to aggregate the annotations, because the computer does not know the true measurements of the other objects. Therefore, we restricted this comparison to be performed only with the target object to be reconstructed, which we already know the exact true size of. Example measurement estimates are:

- *Direct*: "The object's length is about 80 inches."
- *Relative*: "The object is about 10% longer than the target object."

Granularity of Measures

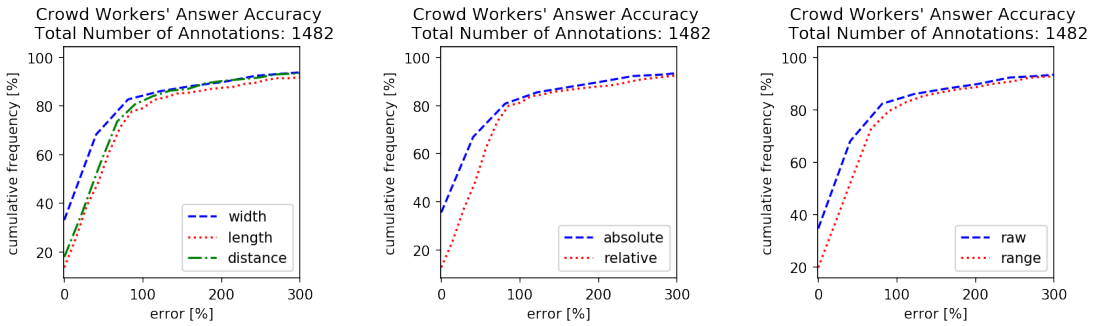
Unless the annotator knows the exact make and model of an object, it is infeasible to precisely identify the length or width of it. Similarly, for the distance measurement, it is hard to tell the exact distance of an object from a single image. Therefore, we designed two elicitation approaches; a single valued estimate, and a ranged valued estimate. For the range valued approach, the annotators can freely choose the granularity of their answer. Our proposed joint object annotation aggregation method can handle multiple granularities because the penalty function (Eq. 6) is designed to accept lower and upper bounds. Example measurement estimates are:

- *Single*: "The target is about 32 feet away."
- *Ranged*: "The target object is about 30 feet to 40 feet away."

We note that even though there are 12 total measurement estimation types that could be tested (3 selection$\times$ 2 directness$\times$ 2 granularity), we only use 10 types when investigating crowd workers' accuracy in estimating measurements. This is because the combination of two *relative distance* annotations cannot be computed with our joint object aggregation method. Therefore, we excluded the two combinations $distance \times relative \times single$ and $distance \times relative \times ranged$ from both the interface design and the data collection.

5.4.2 Task Interface

Our task interface presents crowd workers with step-by-step instructions and web annotation tools. After reading through the instructions, crowd workers can click a button to proceed to the task. Then they are shown an image with reference objects to be annotated, which are indicated with a green box. For the *relative* condition, the target object to be compared is also indicated with a red box. After checking the task image, the next step is to provide estimated measurement values, as in Figure 5.6(b). The workers are allowed to choose a unit of measurement from the following four options: meter, feet, inch, and yard. The last step is to mark the corresponding line on the selected reference object, as in Figure 5.6(c). Since the length and width of an object can be ambiguous in certain cases, e.g., when an object has no apparent longer side, we set a rule in distinguishing length and width. The rule is explained in the instructions with various example objects as in Figure 5.6(b). For distance estimate annotation, this instruction was hidden. The instructions in Step 2 included examples of corresponding lines, and we reminded workers that the line should be drawn on the ground in the image where the objects are positioned.



(a) Different selection (b) Different directness (c) Different granularity

Figure 5.7: Cumulative frequency of annotations is plotted with respect to the percent error of the annotation. No significant difference was observed within each dimension.

	length	**width**	**distance**
direct × single	61.98/120.43(180.05)	60.65/64.21(57.77)	51.02/60.09(61.84)
direct × range	67.31/167.56(270.63)	61.94/184.46(698.51)	49.88/55.15(49.19)
relative × singe	65.24/238.02(658.27)	65.02/109.38(151.22)	-
relative × range	58.51/291.16(1209.50)	68.01/104.58(129.45)	-

Table 5.1: Median/Average(Standard Deviation) percent error computed as in Eq. 1 and Eq. 2 to evaluate worker answers for different measure estimate types.

5.4.3 Evaluating the Impact of Measure Type

To evaluate the impact of measurement type on workers' annotation accuracy, we recruited 300 crowd workers. We asked the workers to annotate the 15 task images, 10 indoor and five outdoor images, using the 10 different measurement types. Images were grouped into fives to distinguish indoor and outdoor images. The order of the images within a group and the objects within an image were randomized to avoid learning effects. Each worker annotated one object per image using a single measurement type that was provided. Participants were limited to workers from the US who had a task acceptance rate $\geq$95%. Each worker could only participate once, and was paid \$1.35 per task, yielding an average wage of \$9/hour—above the 2019 U.S. federal minimum wage.

	length	**width**	**distance**
direct × range	25.00/28.98(17.32)	28.57/31.09(21.48)	28.57/36.66(24.08)
relative × range	18.18/25.82(27.38)	22.22/30.62(29.64)	-

Table 5.2: Median/Average(Standard Deviation) precision computed as in Eq. 4 to evaluate worker answers for different measure estimate types.

5.4.4 Results

A total of 1500 annotations were collected across the 10 types, but 18 annotations had to be dropped because the submission was incomplete. We ended up with a total of 1482 annotations. Figure 5.7 shows the cumulative frequency of the percent error for each element within each dimension. The trend is similar across types: a steep increase until 100 percent error, and then slows down the speed of increasing.

The median and average percent error of crowd workers' responses for the 10 measurement types are shown in Table 5.1. To compare the performance of the 10 measurement types, we ran $\binom{10}{2} = 45$ (10 choose 2) pairwise comparisons using a Mann-Whitney U test because the worker responses were skewed (non-normal). With Bonferroni correction, we considered the comparison result significantly different if the p-value was below $.05/45 = .0011$. From the 45 comparisons, the pairs with significant difference were the following four:

- *distance* × *direct* × *range* outperformed *length* × *direct* × *range*
  ($U = 8554.0$, $n_1 = 150$, $n_2 = 150$, and $p < .0005$)
- *distance* × *direct* × *single* outperformed *length* × *direct* × *range*
  ($U = 8658.0$, $n_1 = 150$, $n_2 = 150$, and $p < .0005$)
- *distance* × *direct* × *range* outperformed *width* × *relative* × *range*
  ($U = 8658.0$, $n_1 = 150$, $n_2 = 149$, and $p < .001$)
- *distance* × *direct* × *single* outperformed *width* × *relative* × *single*
  ($U = 8658.0$, $n_1 = 150$, $n_2 = 145$, and $p < .0001$)

The result shows that the overall crowd worker performance was similar across different types, but direct distance estimation types significantly outperformed some of the other types.

The median and average precision of crowd workers' responses for the five measurement types are shown in Table 5.2. All *single* types were ignored because precision is always 0 for a single value. To compare the performance of the five measurement types, we ran $\binom{5}{2} = 10$ (5 choose 2) pairwise comparisons using a Mann-Whitney U test because the worker responses were skewed (non-normal). With Bonferroni correction, we considered the comparison result significantly different if the p-value was below $.05/10 = .005$. From the 10 comparisons, the pairs with significant difference were the following five:

- $length \times relative \times range$ outperformed $length \times direct \times range$
  ($U = 8296.5$, $n_1 = 150$, $n_2 = 150$, and $p < .0001$)
- $length \times direct \times range$ outperformed $distance \times direct \times range$
  ($U = 9007.0$, $n_1 = 150$, $n_2 = 150$, and $p < .005$)
- $length \times relative \times range$ outperformed $width \times direct \times range$
  ($U = 8658.5$, $n_1 = 150$, $n_2 = 153$, and $p < .005$)
- $length \times relative \times range$ outperformed $distance \times direct \times range$
  ($U = 6562.0$, $n_1 = 150$, $n_2 = 150$, and $p < .0001$)
- $width \times relative \times range$ outperformed $distance \times direct \times range$
  ($U = 8399.5$, $n_1 = 149$, $n_2 = 150$, and $p < .0005$)

The result shows that workers tend to provide a narrower range when asked to annotate the *relative* measure. This might be because relative measurement is already an approximation, giving workers fewer options to extend either the lower or the upper bounds. We also report the task time difference in Table 5.3, which we did not find

	length	width	distance
absolute × raw	47.31/67.24(68.02)	44.77/56.05(43.48)	36.36/46.77(33.76)
absolute × range	45.97/62.63(53.54)	50.95/67.78(49.17)	37.48/58.21(72.93)
relative × raw	39.25/54.31(40.28)	46.45/71.61(117.76)	-
relative × range	53.78/85.37(139.65)	53.30/73.97(64.30)	-

Table 5.3: Median/Average (Standard Deviation) task time for different measure estimate types.

significant across measurement types. Overall, the average accuracy and precision of worker annotations was similar across different types, even though there were some cases with significant performance differences. While images may contain various different objects in varying context, providing the workers with as many types of measurements as possible will allow to cover the diverse cases of use, maximizing the benefit of knowledge diversity of workers.

5.5 System Evaluation

For the evaluation, we assumed that the target object's dimension values (length, width, and height) were known but were not mapped to the image, requiring dimension line annotations for the mapping, which is the same scenario as in (*Song et al.*, 2019b). We ran three analysis studies to investigate the performance of C-Reference: (i) investigation on the effect of the number of additional reference object annotations on the accuracy of 3D location estimation, (ii) investigation on the performance gain compared to a baseline method that uses the same number of total annotations, but only from the target object dimension line annotations, and (iii) investigation on the performance of the penalty function. While the first study allows us to understand the trade-off between the cost and performance of our approach, the second study allows us to understand the significant benefits of using our proposed approach. The last study allows us the understand the impact of the design of the penalty function on the overall system performance.

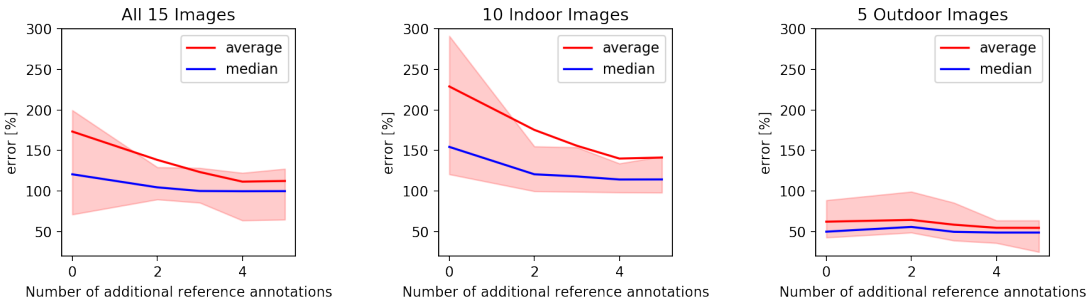
5.5.1 Experimental Setup

We recruited 39 crowd workers to collect the target object annotation on the 15 task images. Each image contained one target object whose 3D location is to be estimated. As mentioned in Section 5.2.1, we included challenging objects such as occluded objects, limited view angle, or small objects to make the task more complex. The image order was randomized the same way as in Section 5.4.3. A total of 13 annotations were collected for each target object. The workers were different from participants in the reference object annotation round. The eligibility and reward settings were the same as in Section 5.4.3. For reference object annotations, we used the same set of annotations, which was collected in Section 5.

Analysis 1: Effect of the number of reference object annotations

To understand the effect of the number of additional reference object annotations, we started from zero reference object annotations, only aggregating the target object dimension line annotations. For target object dimension line annotations, five annotations were randomly chosen from the 13 total annotations. To avoid selection bias, we ran this 50 times, drawing a random group from a total of $\binom{13}{5} = 1287$ (13 choose 5) possible cases each time. The median, average, and standard deviations of the percent error of the 50 samples were computed for each target object. When combining multiple target object dimension lines, we tested both taking the average and taking the median. We used the median throughout the study because the performance of taking the median was better due to being able to remove the effect of outliers.

For reference object annotations, we randomly selected one annotation from each reference object, and combined the selected annotations using our joint object annotation aggregation technique. We increased the number of reference objects from two to five as shown in Figure 5.8. We skipped one annotation because our aggregation method only works for more than two reference annotations. Whenever selecting a



(a) 36% error reduction (b) 39% error reduction (c) 13% error reduction

Figure 5.8: Percent error comparison of the different number of reference object annotations that are aggregated. Adding more reference object annotations decreased the percent errors increasingly. (a) shows the result of all 15 images. There was maximum error reduction of 36% from adding four reference object annotations, compared to adding no reference object annotations. (b) shows the results of all 10 indoor images. There was a maximum error reduction of 39% when adding four reference object annotations. (c) shows the results of all five indoor images. Maximum error reduction was 13% when four annotations were combined. More gain was observed with indoor images.

new annotation, we selected it from an entire pool of the 10 measurement types. The reason for this was to collect diverse combinations of the types, and investigate the effect of the combination of types on the final 3D location estimation accuracy. We ran this 50 times, the same as the target object annotations, drawing a random annotation from a total of $\binom{20}{1} = 20$ (20 choose 1) possible cases for each object at each time.

Analysis 2: Performance comparison with a baseline.

To investigate the benefit of using reference object annotations, we set the number of total annotations to 10 and compare C-Reference with a baseline as follows:

- Baseline (10 target object dimension line annotations): estimates a target object's 3D location without using reference object annotations, only using the target object's dimension line annotations. Ten target object dimension line annotations were randomly chosen from the 13 total annotations, a total of

50 times—drawing a random group from a total of $\binom{13}{10} = 286$ possible cases each time. Same as in Results Analysis 1, we used the median when combining multiple target annotation dimension lines to avoid the effect of outliers.

- C-Reference (five target object dimension line annotations and five reference object annotations): estimates a target object's 3D location using both target object annotations and reference object annotations. For target object dimension line annotations, we randomly chose five from the 13 total annotations, a total of 50 times—drawing a random group from a total of $\binom{13}{5} = 1287$ possible cases each time. Same as in Baseline, we used the median when combining multiple target annotation dimension lines. For reference object annotations, we randomly sampled five annotations with the same drawing scheme we used in Results Analysis 1.

For the performance evaluation of the 3D location estimation for both conditions, we used the percent error as in Eq. 1. The average, median, and standard deviations were computed.

Analysis 3: Handling reference object annotations with large noise.

To understand the performance of the penalty function, we conducted an in-depth analysis of C-Reference's output from Results Analysis 1. The goal is to determine whether the penalty function was able to automatically handle poor quality annotations. We used the estimation results from aggregating two reference object annotations and divided the results into two groups. One group had no input value for the penalty function, because no single pair of reference object annotations generated a valid d_{target} value (there was no solution to the corresponding system of equations). The second group had a value for the penalty function. We call these groups *skipped* and *non-skipped*, respectively. The *skipped* condition can be thought of as the penalty function being turned off, because there is no input or output from the function.

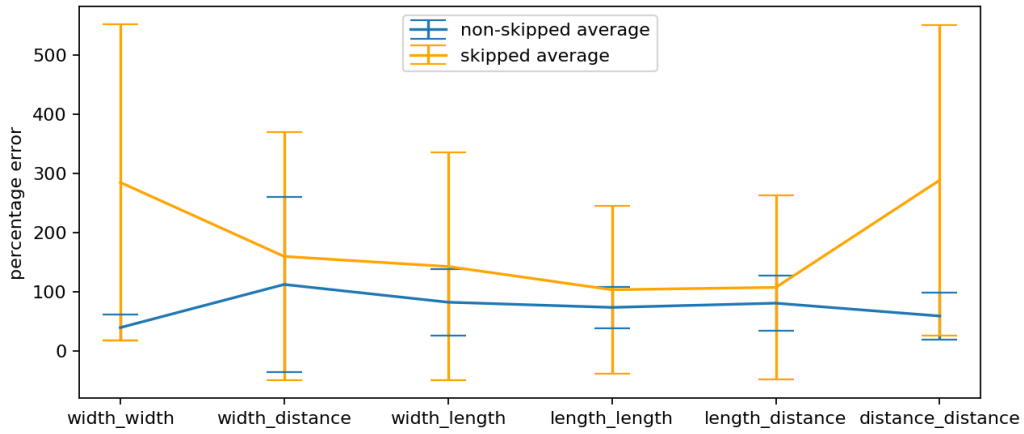


Figure 5.9: Performance comparison between *skipped* and *non-skipped* groups when two reference object annotations are aggregated. Here, we further divided the groups into six aggregation pairs. The average percent error of the *non-skipped* group was always lower than the *skipped* group, indicating the penalty function is beneficial.

5.5.2 Parameter Settings

For the experiment, we used the same optimization function as in Eq. 10 for the baseline. For the C-Reference method, we added the penalty function in Eq. 6 to the objective function in Eq. 10. We set σ to 100, and the basin-hopping iteration number as 200. We picked a large number of iterations to improve the overall performance for every condition including the baseline. The optimizer stopping criterion was set as 1e-8, L-BFGS-B bound as $-10 \leq x \leq 10$, $-5 \leq y \leq 5$, $1 \leq z \leq 50$, $-\pi \leq \theta < \pi$, and $100 \leq f \leq 3000$. The two parameters for the penalty function were set as $a = 8$ and $b = 50$. Lastly, we manually obtained the horizontal line using parallel lines on the ground plane.

5.5.3 Results

In this section, we investigate the performance of C-Reference, which implements our proposed joint-object aggregation method to elicit and leverage the knowledge diversity of crowd workers. We report the results of the three analysis studies below.

Analysis 1: Adding more reference annotations consistently improved performance until hitting a saturation point.

As shown in Figure 5.8, the percent error of the 3D location estimation of the target object consistently decreased as more reference annotations were added. Across all 15 target objects, the maximum improvement was obtained when four reference annotations were aggregated, which was the saturation point. To compare the performance of zero reference annotations with four reference annotations, we used a Mann-Whitney U test pairwise comparison because both of the results were skewed (non-normal). The result showed that there was a 36% performance improvement from four added annotations, which was significant improvement ($U = 209955.0$, $n_1 = 750$, $n_2 = 750$, and $p < .0001$). To understand the performance better, we separated images to indoor and outdoor images and computed the percent error of 3D location estimation of the target objects. While indoor images had a 39% error reduction, outdoor images only had a 13% error reduction. The average percent error of indoor and outdoor images at four reference annotations was 139.97% and 54.57%, respectively. The average ground truth distance of target objects in indoor images was 4.95 meters, and in outdoor images it was 29.29 meters, which indicates that the long ground truth distance may have reduced both the percent error of estimation results and the performance gain from adding reference annotations.

Analysis 2: C-Reference not only significantly outperformed the accuracy but also required less annotation time compared to the baseline.

We compared the performance of C-Reference with the baseline to understand the effect when the number of total annotations is the same for the two conditions. We used a Mann-Whitney U test pairwise comparison because both of the results were skewed (non-normal). The result showed that there was a significant 40.4% performance improvement from four added annotations ($U = 201574.0$, $n_1 = 750$, $n_2 = 750$,

and $p < .0001$). While C-Reference consistently improved the performance as more annotations were added, the baseline did not improve the performance according to the added dimension line annotations on the target object. We also computed the average task time for both annotation tasks: target object annotation and reference object annotation. The average task time for target object annotation was 183.5 seconds, while that for reference object annotation was 64.4 seconds, which is 35% as much task time compared to target object annotation. Therefore, collecting more reference object annotations instead of more target object annotations could save on average 65% time per added annotation.

Analysis 3: Penalty function was effective in penalizing infeasible solutions, and was robust to noisy input annotations.

We analyzed the penalty function performance by comparing the 3D location estimation result of the two groups: the *skipped* annotation group and the *non-skipped* annotation group. Then we further divided the estimation results into six subgroups based on the selection pairs to investigate if the observed pattern is consistent across aggregation pairs. As shown in Figure 5.9, we observed that the *non-skipped* group always performed better than the *skipped* group regardless of the selection pair. That is, when the penalty function is turned off (the *skipped* group), the performance is worse for any selection pair. This result along with the results from Results Analysis 2 demonstrate the benefit of our proposed joint object annotation aggregation approach in improving the performance of 3D location estimation. Because the penalty function could identify that the system of equations has no solution for a given annotation pair, we could put them into the *skipped* group. Another strategy that can be used in future work may be collecting additional annotations until the set of annotations becomes *non-skipped*. This may only aggregate and input high quality annotations to the penalty function, while still leaving the poor ones as *skipped*.

5.6 Discussion

Our study and analysis showed that crowd worker approximations can improve system performance when aggregated and transformed into a soft constraint for an optimizer. The experimental results using C-Reference demonstrate that our proposed annotation aggregation method can effectively create soft constraints from annotations on multiple different in-scene objects. In this section, we discuss 1) the generalizability of introducing soft constraints as a way to leverage approximate input from the crowd, and 2) the potential benefits of combining machine optimization with crowd-generated constraints to create the synergistic effect of performance improvement.

5.6.1 Generalizability of Soft Constraints in Crowdsourcing

As an aggregation technique, we introduced the approach of turning a range or single-valued estimations into a soft constraint with weighted sigmoid functions. Unlike conventional voting aggregation, which requires the majority of people to agree on one value, or averaging, which assumes that one error offsets the other, the soft constraint allows us to aggregate crowd answers with a more flexible assumption on the patterns of noises. For example, in our system, we could not assume a specific error pattern because crowd workers will have different knowledge and generate different errors. Thus, conventional aggregation approaches that assume specific error patterns would work for our problem. However, we could leverage our approach of combining soft constraint and optimization within our problem as it is more flexible to diverse error patterns.

An alternative to generating soft constraints based on workers' direct approximates could be combining the soft constraints with other answer elicitation approaches such as confidence rating (*Oyama et al.*, 2013a,b). For instance, if a worker estimates a value with high confidence, we can transform this into the soft constraint

with a small range with the center value being the estimated value. On the other hand, if the worker estimates a value with low confidence, we can turn it into a soft constraint with a wider range. As we applied maximum penalty function to prevent overshooting penalty with wrong estimations, we would also be able to apply such techniques to generate soft constraints with confidence. One design decision that should be made for turning confidence into soft constraints would be how to model the mapping between confidence and range.

5.6.2 Combining Machine Optimization and Crowd-generated Constraints

The approach of reinforcing machine optimization with crowd inputs is applicable to a set of tasks with a certain assumption. The machine should be able to conduct the search process of the optimization, as crowd inputs would only add information regarding which part of the state space is worth searching. For example, if the machine runs optimization for user intent recognition, it would be challenging to be aware of all possible search spaces (e.g., all possible human intention). This would make the machine's optimization process inefficient. For such optimization tasks, crowd workers would be leveraged in ways, such as expanding the search space to the extent that machines cannot know yet. The synergistic combination of machine optimization with crowd-generated constraints would be more valuable for cases where there are multiple unknown correlated variables that need to be estimated. If there is only one variable to be found, deterministic approaches such as averaging or voting would be sufficient based on the detection of the error patterns.

5.7 Summary

Converting widely-available 2D images and videos to 3D can help accelerate the training of machine learning systems in spatial reasoning domains ranging from in-home assistive robots to augmented reality and autonomous vehicles. While automat-

ing this task is challenging because it requires not only estimating object location and orientation, but also latent camera properties, leveraging people's spatial understanding of scenes by crowdsourcing visual annotations of 3D object properties can help the conversion process in a scalable way. Unfortunately, getting people to directly estimate 3D properties reliably is difficult due to the limitations of human motor accuracy, selectable pixels (resolution), and people's ability to perceive 3D precisely (i.e., humans do not "see" depth like a laser scanner). In this chapter, we propose a crowdsourcing approach that uses multiple objects and annotation granularities to help jointly reconstruct the 3D state of a target object. Our proposed annotation aggregation method transforms crowd workers' *approximations* of objects' size or distance into soft constraints for an optimizer by using the spatial relationship between the objects. This relaxes the assumption that workers will each have similar levels of knowledge or skill, helping more diverse answers to be used effectively. These soft constraints help penalize infeasible solutions, increasing the chance of an optimizer finding a more accurate solution. We evaluate our joint object estimation approach with 363 crowd workers and show that the proposed method can reduce errors in a target object's 3D location estimation by over 40%, while requiring only 35% as much human time. Our work introduces a novel way to aggregate collective perception in settings where precise annotation is challenging, but approximate annotation of multiple alternative elements with known relationships is feasible.

CHAPTER VI

Discussion and Future Directions

In this chapter, we discuss how the way that we approach input diversity in this thesis is different from previous perspectives on leveraging responses from the crowd. We then overview design considerations for systematically leveraging input diversity and suggest a set of guidelines for future work that aims to apply our approaches in new crowdsourcing tasks and systems. We also discuss potential ways for the effectiveness of these approaches to be measured using concepts from information theory. We use these ideas to generalize the findings and the describe limitations of where input diversity approaches can be usefully applied. We hope that future work can investigate more formal notions of when to leverage input diversity.

6.1 Aggregating Diverse Responses from the Crowd

Diversity is a property of the crowd that people have been interested in since the idea of the Wisdom of Crowds was first articulated (*Surowiecki*, 2005). If contributors make no errors, no aggregation or correction is needed. If the errors they make were all identical, there would be no way to make the collective answer more accurate using aggregation methods. However, in many tasks, including complex annotation tasks, responses contain a level of error or imprecision, but the variation (diversity) in errors means it is possible to aggregate responses such that the collective answer

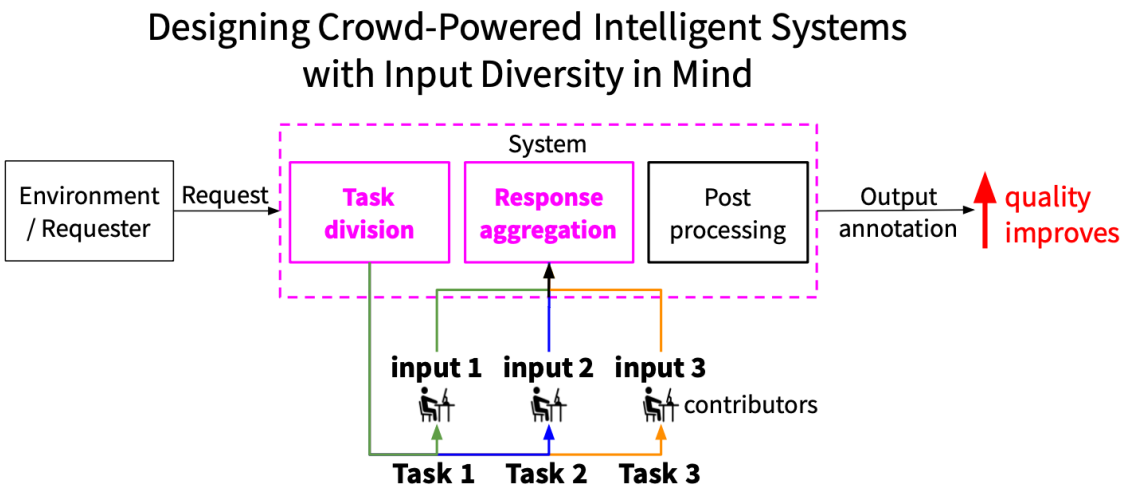


Figure 6.1: In this thesis we showed that the benefit from leveraging diverse input can be *systematically* achieved by designing the systems with the input diversity in mind. We showed that by jointly designing the task division step and the response aggregation step, we can achieve diverse responses from the workers, which could be aggregated in a way to improve the final output quality.

is more accurate potentially more so than any single constituent response. In some cases, this even allows sufficiently large groups of contributors to solve problems even experts cannot.

When aggregating responses, it would be ideal if the errors being aggregated were uniformly distributed around the ground truth answer. Widely adopted aggregation solutions such as averaging or majority voting assume uniform distribution of errors in the input. Other weighted aggregation methods such as expectation maximization (*Dawid and Skene*, 1979; *Ipeirotis et al.*, 2010) assumes weighted errors around the ground truth and assigns counter-weight to each response so that the result of the weighting can be uniform distribution. However, this requires contributors to provide many answers that can be compared to the group's answers in order to determine these weights. Alternative methods like Self-correcting Crowds (*Lasecki and Bigham*, 2012) could be viewed as asking crowd workers to correct a collective response them-

selves by adjusting their relative weight with a (live) view of the final outcome. This thesis showed that it is possible to design effective aggregation methods for a broader class of error patterns, and under more common crowdsourcing assumptions, by eliciting input diversity through the design of task, tools, and systems. This provides a new perspective on solving challenging problems by eliciting diverse error patterns in the collective responses, which can generate more accurate aggregated answers.

6.2 Designing Crowdsourcing Tasks with Diversity in Mind

The methods introduced in this thesis elicit crowd responses with diverse error patterns through task and interface design, and aggregate them in a way that improves the combined answer to be of higher quality than any individual response. This thesis identified problems where the tasks induce error biases which poorly approximate the ground truth answer. Then, we introduced task designs and aggregation methods to overcome these error patterns. While we only demonstrated the effectiveness of the approach in a specific set of problem domains, we believe that the concepts and techniques can be applied to a broader set of problems. We suggest a set of task properties that describe when we believe our proposed diversity-driven approaches would be applicable:

(1) The task can be broken down into subtasks that are all connected in a way that is useful during answer aggregation. For example, in the image segmentation task introduced in Chapter 3, the same target object is shared across different tools. The inter-frame aggregation technique introduced in Chapter 4 and the joint object aggregation technique introduced in Chapter 5 used the temporal or spatial relationships between images to aggregate annotations from heterogeneous input.

(2) The task is tractable enough for contributors to provide approximate answers, but the responses are expected to be imprecise due to limitations such as a lack of information, restricted human perception, or restricted human motor skills. For

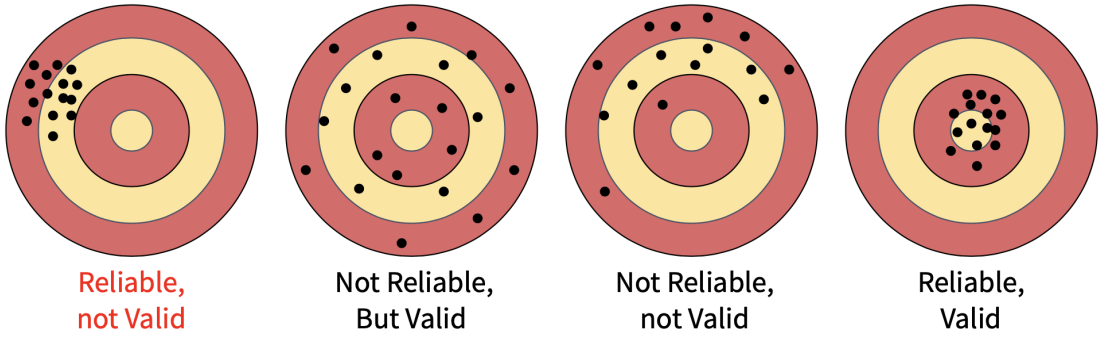


Figure 6.2: We defined "systematic bias" as reliable but not valid responses to a system as shown in the first target above. We claim that these systematic biases could be intentionally induced through the task design, which could lead to higher aggregate performance when combined appropriately. Responses that are not reliable but valid, or neither reliable nor valid (as shown in the second and third targets, respectively) are not desired because the aggregation may not lead to improved accuracy. Further discussion on the limitations of input diversity approach will be introduced in Section 6.4.

instance, the image annotation tasks introduced in this thesis are tractable enough to roughly mark the target object's boundary, but being pixel-level precise is very challenging and time consuming.

(3) The expected error patterns of the subtasks should be reliable even if they are biased. From systems engineering, reliability and validity are two properties of responses that describe the quality of the responses. Reliability denotes the extent to which the responses can be reproduced, and validity denotes the extent to which the aggregated responses represent the ground truth. As shown in the first target in Figure 6.2, when the responses are reliable but biased (not valid), the input diversity approach can be effective in producing better aggregate performance than any homogeneous type of input can do alone. If the input from each subtask is unreliable, a small number of responses would not be able to approximate the ground truth when aggregated. That means each subtask could generate reliable but biased error distributions in different directions, which could be canceled out when aggregated. The

desired conditions for input diversity approaches could also be interpreted from an information theory point of view, which will be discussed in the next section.

6.3 Joint Entropy and Mutual Information as a Means to Interpret the Effect of Leveraging Input Diversity

In this section we use the notion of joint entropy and mutual information from information theory to suggest an interpretation of why leveraging input diversity can improve the aggregate answer accuracy.

When systematically eliciting biased errors through the design of a task, we can consider each error source as a random variable created from the same input type. For example, in Chapter 3, all responses from Tool A to the EM estimator can be denoted as random variable X, while those from Tool B can be denoted as random variable Y. If we denote the amount of information from X as $H(X)$ and from Y as $H(Y)$, the joint entropy between the two variables can be denoted as $H(X, Y)$.

The joint entropy provides an upper bound on the potential accuracy of the collective answer (or annotation). That is, the different error patterns across inputs should actually contain diverse information rather than the same (repetitive) information. This was discussed in Chapter 3 Figure 3.10 as well, where the tools with similar error patterns did not show performance improvement when combined. Another factor that needs to be considered when designing systems to leverage input diversity is the amount of mutual information across random variables. That is, if the mutual information across random variables are small, it could imply that the aggregation of the variables may not lead to the approximation of the ground truth to be estimated. For example, in Chapter 3, if two image segmentations are from different objects in different images, the segmentations would have zero mutual information, and the aggregation of the two would be meaningless.

Combining the notion of both joint entropy and mutual information in input diversity, we suggest the following formula as a primitive form of expressing the amount of input diversity:

$$\mathrm{ID} = \underset{\{X,Y\}}{\operatorname{argmax}} \left(\alpha H(X,Y) + \beta I(X;Y) \right) \qquad \text{(Eq. 1)}$$

where X and Y are random variables whose values are response outcomes from a source, $H(X,Y)$ indicates their joint entropy, $I(X;Y)$ indicates the mutual information between random variables X and Y. ID indicates the amount of input diversity generated from X and Y. α and β represent the weight to each values. Interestingly, a similar approach was introduced for feature selection in machine learning, where the minimum redundancy maximum relevance (MRMR) framework enables to select features that provide more balanced coverage of the space capturing broader attribute of the dataset (*Peng et al.*, 2005; *Ding and Peng*, 2005). While the equation we provide is not a rigorous form with proofs, we believe that this approach, from an information theoretic point of view, can provide a conceptual guidelines for future researchers and practitioners in designing their systems to best leverage input diversity.

6.4 Limitations

As briefly mentioned in Section 6.2, input diversity may not always be a property that is desirable for a system to leverage. As depicted in the last case in Figure 6.2, if a system could be easily designed to elicit a single type of input responses that are both reliable and valid, a simple averaging method would be enough to use. Therefore, input diversity approaches may not be desired in these tasks. However, due to the bias-variance trade-off (*Hero et al.*, 1996), designing tools or interfaces with both low bias and low variance could be a challenging problem itself. A better solution in this case would be creating multiple high bias but low variance error clusters (input

types), and aggregating them to approximate the ground truth.

Another scenario is when it is possible to build a "zero error" tool or interface for a task. Usually, if a task is simple and easy enough to respond the exactly correct answer for anyone, it would not be desirable to complicate the design of a system by eliciting input diversity. However, there is a tradeoff between the cost of designing the "best" tool versus several imperefect ones. However, in general, the tasks that would benefit most from leveraging input diversity are those that are complex enough and hard for individuals to respond the correct answer by oneself.

6.5 Implications for the Future of Work

Humanity has evolved with the development of tools that allow us to automate or augment human labors to accomplish things that were only imaginable before. In its historical context, often times tools themselves shaped how we work and what we work on, e.g., popularization of railroads in 1860s innovated people to consider systematic management in everyday work, encouraging people to put emphasis on efficiency (*Yates*, 1993). Recently, crowdsourcing marketplaces such as Amazon Mechanical Turk, Upwork, and Crowdflower started to serve as tools to extend paid work into the online environment, enabling work in virtual or remote settings where workforce is okay to be anonymous and not collocated (*Kittur et al.*, 2013). The Crowd Agent architecture (*Lasecki*, 2015) introduced a new viewpoint on how to form and coordinate of groups in the context of continuous real-time crowdsourcing, which opened possibilities in creating hybrid intelligence systems. Research on Learnersourcing (*Kim*, 2015) introduced new ways to conduct large-scale data collection tasks where the data is passively produced as a by-product of engaging in online learning.

We believe that our work has important implications on how we think of redesigning work beyond crowdsourcing domain, considering how we leverage the di-

verse potential input sources for better collective performance. As Page argued in "Diversity Bonus (*Page*, 2008)": workforce diversity is indispensable in the knowledge economy to improve performance of organizations confronting complex challenges, acknowledging diversity in workplace as a winning strategy (more than simply the right thing to do for society to encourage inclusion) is an important step toward improving the bottom line of performance in online, remote workplaces. This thesis provide empirical evidences where

While we only explored three dimension of input diversity: tool diversity, perspective diversity, and knowledge diversity, there could be other dimensions of diversity in work. We discuss future directions in exploring other dimensions and applications in the next section.

6.6 Future Directions

We believe that this thesis has important implications beyond crowdsourcing, concerning how we think about leveraging different dimensions of diversity in organizing everyday work. The research and directions explored within this dissertation suggest the following opportunities for future work.

6.6.1 Expanding Input Diversity Approach to Real-Time, Continuous, or Interactive Crowdsourcing

The idea of eliciting and leveraging input diversity in crowdsourcing can be extended to real-time, continuous, or interactive crowdsourcing. The methods to elicit diverse responses from the crowd can be designed in a parallel architecture because the heterogeneous tasks are independent to each other. Therefore, these techniques can be combined with existing real-time crowdsourcing techniques such as instantaneous look ahead approach (*Lundgard et al.*, 2018) or tagging 3D point cloud (*Gouravajhala et al.*, 2018) to further improve the performance aggregate annotations. The

approaches we introduce can also be used in continuous crowdsourcing tasks (*Chung et al.*, 2019b) because leveraging diversity does not require maintaining the context. Lastly, it can also be used in interactive crowdsourcing (*Lasecki and Bigham*, 2013) where the system dynamically ask for the response that will reduce the systematic bias when aggregated.

6.6.2 Expanding Input Diversity Approaches to Creative and Cognitively Challenging Tasks

Future work may build on the proposed approach of leveraging input diversity to explore the effectiveness in other crowdsourcing domains, such as feedback generation (*Luther et al.*, 2014), labeling function generation (*Ratner et al.*, 2017), creative thinking tasks (*Yu et al.*, 2016; *Yu and Nickerson*, 2011; *Lee et al.*, 2018, 2019; *Lee*, 2018), and collective data exploration and discovery tasks (*Jin et al.*, 2017). While these tasks benefits from diverse responses and ideas from people, input diversity may not come naturally from crowdsourcing, e.g., workers may lean toward similar responses instead of being creative. Therefore, systematically eliciting diverse responses through task design would be required. For example, one can use priming each sub-group of workers with different examples to elicit different responses. Then an aggregation method that considers the different priming would be able to combine the ideas knowing the source of difference between them. The knowledge diversity approach we introduce in Chapter 5 may be a close example to this type of applications. As we induced each worker to focus on different objects in a scene for each task, a similar workflow can be designed to make crowd workers focus an different aspect of the task.

6.6.3 Expanding Input Diversity Approach to Include Minority Groups in Workplaces

We envision input diversity approaches could serve as one means of improving the inclusion of historically underrepresented groups, such as older adults and neurodiverse people, in workplaces. While our findings suggest that leveraging input diversity can improve aggregate performance of the groups, the natural diverse perspectives that these minorities can introduce to the workplace can serve as a valuable input pattern in many situations. For example, at Hewlett Packard Enterprise, neurodiverse software testers brought new perspective to detect a failure pattern in projects launch which help successfully redesign the whole process (*Austin and Pisano*, 2017). Similarly, older adults can provide different perspectives to a task based on their life experiences, which could encourage diverse thinking in problem solving.

6.6.4 Input Diversity Approach in Coordinated and Dependent Work

While the examples introduced in this thesis shows applications of input diversity approaches in independent subtasks, we believe that the approach has potential for coordinated and dependent work as well. For example, paraphrasing tasks could be designed to be sequentially dependent (as in the the classic "Telephone Game"), where diverse but paraphrases could be generated from different groups. This is possible due to the fact that the paraphrase could be primed based on the given context (*Jiang et al.*, 2017; *Mitchell et al.*, 2014). Similarly, diverse ways to explain task instructions or educational content can help improve outcomes (*Williams et al.*, 2016, 2018). When applying input diversity to coordinated and depended work, more factors should be considered such as the dynamics in the interaction between workers or the change in the members.

CHAPTER VII

Conclusion

This dissertation has explored the potential of diversity-driven approaches in crowdsourcing, which systematically elicit and leverage the input diversity of crowd workers to improve the quality of collective annotations. In this context, we have presented general contributions in designing crowdsourcing workflows and answer aggregation algorithms. We introduced three crowdsourcing approaches: (i) leveraging tool diversity, (ii) leveraging instance diversity, and (iii) leveraging knowledge diversity. For each approach, we introduced effective aggregation techniques that enable the practical usage of the conceptual approach in the visual annotation domain. The proposed aggregation techniques compensate biases or uncertainty caused by the tool, data, or the system, improving the accuracy of the collective output from crowd-powered and hybrid intelligence systems.

As a final remark, this dissertation has explored the following thesis:

By designing crowd-powered intelligent systems with input diversity in mind to elicit and aggregate diverse inputs with different error distributions, it is possible to systematically reconstruct higher quality annotations.

Strategically eliciting and leveraging diverse system inputs from the workers represents a new paradigm in solving not only crowdsourcing problems, but also various other problems in workplaces, communities, and societies. This thesis demonstrates a

new way of improving the aggregate quality of crowd-powered system output through systematically eliciting and leveraging the input diversity from participants. We hope this work will be a valuable resource for a broader research field, helping researchers and practitioners to improve the quality of task outputs through the design of tasks and the inclusion of diverse participants for complex systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

Austin, R. D., and G. P. Pisano (2017), Neurodiversity as a competitive advantage, *Harvard Business Review*, *95*(3), 96–103.

Bearman, A., O. Russakovsky, V. Ferrari, and L. Fei-Fei (2016), What's the point: Semantic segmentation with point supervision, in *European Conference on Computer Vision*, pp. 549–565, Springer.

Bell, S., P. Upchurch, N. Snavely, and K. Bala (2013), Opensurfaces: A richly annotated catalog of surface appearance, *ACM Transactions on Graphics (TOG)*, *32*(4), 111.

Bernstein, M. S., G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich (2010), Soylent: a word processor with a crowd inside, in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 313–322, ACM.

Bernstein, M. S., J. Brandt, R. C. Miller, and D. R. Karger (2011), Crowds in two seconds: Enabling realtime crowd-powered interfaces, in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 33–42, ACM.

Bigham, J. P., et al. (2010), Vizwiz: nearly real-time answers to visual questions, in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 333–342, ACM.

Bojarski, M., et al. (2016), End to end learning for self-driving cars, *arXiv preprint arXiv:1604.07316*.

Bragg, J., Mausam, and D. S. Weld (2013), Crowdsourcing multi-label classification for taxonomy creation, in *First AAAI conference on human computation and crowdsourcing*.

Bray, M., E. Koller-Meier, and L. Van Gool (2004), Smart particle filtering for 3d hand tracking, in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 675–680, IEEE.

Cao, X., A. C. Bovik, Y. Wang, and Q. Dai (2011), Converting 2d video to 3d: An efficient path to a 3d experience, *IEEE MultiMedia*, *18*(4), 12–17.

Carlier, A., V. Charvillat, A. Salvador, X. Giro-i Nieto, and O. Marques (2014), Click'n'cut: Crowdsourced interactive segmentation with object candidates, in *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia*, pp. 53–56, ACM.

Chang, J. C., S. Amershi, and E. Kamar (2017), Revolt: Collaborative crowdsourcing for labeling machine learning datasets, in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 2334–2346, ACM.

Chen, L.-C., S. Fidler, A. L. Yuille, and R. Urtasun (2014), Beat the mturkers: Automatic image labeling from weak 3d supervision, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3198–3205.

Chen, W., Z. Fu, D. Yang, and J. Deng (2016), Single-image depth perception in the wild, in *Advances in Neural Information Processing Systems 29*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, pp. 730–738, Curran Associates, Inc.

Chen, W., S. Qian, and J. Deng (2019), Learning single-image depth from videos using quality assessment networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5604–5613.

Chen, Y., M. Pandey, J. Y. Song, W. S. Lasecki, and S. Oney (2020), Improving crowd-supported gui testing with structural guidance, in *Proceedings of the SIGCHI conference on human factors in computing systems (submitted)*.

Chung, J. J., J. Y. Song, S. Kutty, S. R. Hong, J. Kim, and W. S. Lasecki (2019a), Efficient elicitation approaches to estimate collective crowd answers, in *Proceedings of the ACM conference on Computer-Supported Collaborative Work (CSCW '19)*, ACM, New York, NY, USA, doi:10.1145/3359164.

Chung, J. J. Y., F. Xiao, N. Banovic, and W. S. Lasecki (2019b), Towards instantaneous crowdsourcing in the wild with crowd prediction, in *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM.

Criminisi, A., I. Reid, and A. Zisserman (2000), Single view metrology, *International Journal of Computer Vision*, *40*(2), 123–148.

Dai, A., A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. NieBner (2017), Scannet: Richly-annotated 3d reconstructions of indoor scenes, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2432–2443, IEEE.

Dawid, A. P., and A. M. Skene (1979), Maximum likelihood estimation of observer error-rates using the em algorithm, *Applied statistics*, pp. 20–28.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009), Imagenet: A large-scale hierarchical image database, in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee.

Denis, P., J. H. Elder, and F. J. Estrada (2008), Efficient edge-based methods for estimating manhattan frames in urban imagery, in *European conference on computer vision*, pp. 197–210, Springer.

Di Salvo, R., D. Giordano, and I. Kavasidis (2013), A crowdsourcing approach to support video annotation, in *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*, p. 8, ACM.

Dietterich, T. G., et al. (2000), Ensemble methods in machine learning, *Multiple classifier systems*, *1857*, 1–15.

Ding, C., and H. Peng (2005), Minimum redundancy feature selection from microarray gene expression data, *Journal of bioinformatics and computational biology*, *3*(02), 185–205.

Dosovitskiy, A., G. Ros, F. Codevilla, A. Lopez, and V. Koltun (2017), Carla: An open urban driving simulator, *arXiv preprint arXiv:1711.03938*.

Dow, S., A. Kulkarni, S. Klemmer, and B. Hartmann (2012), Shepherding the crowd yields better work, in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1013–1022, ACM.

Eigen, D., and R. Fergus (2014), Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, *CoRR*, *abs/1411.4734*.

Fischler, M. A., and R. C. Bolles (1981), Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, *24*(6), 381–395.

Freund, Y., and R. E. Schapire (1995), A desicion-theoretic generalization of on-line learning and an application to boosting, in *European conference on computational learning theory*, pp. 23–37, Springer.

Gadiraju, U., B. Fetahu, and R. Kawase (2015), Training workers for improving performance in crowdsourcing microtasks, in *Design for Teaching and Learning in a Networked World*, pp. 100–114, Springer.

Gebru, T., J. Krause, J. Deng, and L. Fei-Fei (2017), Scalable annotation of fine-grained categories without experts, in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1877–1881, ACM.

Geiger, A., C. Wojek, and R. Urtasun (2011), Joint 3d estimation of objects and scene layout, in *Advances in Neural Information Processing Systems*, pp. 1467–1475.

Geiger, A., P. Lenz, and R. Urtasun (2012), Are we ready for autonomous driving? the kitti vision benchmark suite, in *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gordon, M., J. P. Bigham, and W. 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*, pp. 81–82, ACM.

Gouravajhala, S., J. Y. Song, J. Yim, R. Fok, Y. Huang, F. Yang, K. Wang, Y. An, and W. S. Lasecki (2017), Towards hybrid intelligence for robotics, *Collective Intelligence Conference (CI)*.

Gouravajhala, S. R., J. Yim, K. Desingh, Y. Huang, O. C. Jenkins, and W. S. Lasecki (2018), Eureca: Enhanced understanding of real environments via crowd assistance.

Gray, M. L., and S. Suri (2019), *Ghost Work: How to Stop Silicon Valley from Building a New Global Underclass*, Houghton Mifflin Harcourt.

Griffin, B., and J. Corso (2019), Tukey-inspired video object segmentation, in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1723–1733, IEEE.

Gurari, D., M. Sameki, and M. Betke (2016), Investigating the influence of data familiarity to improve the design of a crowdsourcing image annotation system, *HCOMP*.

Hansen, L. K., and P. Salamon (1990), Neural network ensembles, *IEEE transactions on pattern analysis and machine intelligence*, *12*(10), 993–1001.

Hara, K., J. Sun, R. Moore, D. Jacobs, and J. Froehlich (2014), Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning, in *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 189–204, ACM.

He, K., G. Gkioxari, P. Dollár, and R. B. Girshick (2017), Mask R-CNN, *CoRR*, *abs/1703.06870*.

Heit, E. (1994), Models of the effects of prior knowledge on category learning., *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *20*(6), 1264.

Hero, A. O., J. A. Fessler, and M. Usman (1996), Exploring estimator bias-variance tradeoffs using the uniform cr bound, *IEEE Transactions on Signal Processing*, *44*(8), 2026–2041.

Hoiem, D., A. Efros, and M. Hebert (2005), Geometric context from a single image, in *ICCV*.

Ipeirotis, P. G., F. Provost, and J. Wang (2010), Quality management on amazon mechanical turk, in *Proceedings of the ACM SIGKDD workshop on human computation*, pp. 64–67, ACM.

James, S., and E. Johns (2016), 3d simulation for robot arm control with deep q-learning, *arXiv preprint arXiv:1609.03759*.

Jiang, Y., J. K. Kummerfeld, and W. S. Lasecki (2017), Understanding task design trade-offs in crowdsourced paraphrase collection, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 103–109, Association for Computational Linguistics, Vancouver, Canada, doi:10.18653/v1/P17-2017.

Jiang, Y., C. Finegan-Dollak, J. K. Kummerfeld, and W. Lasecki (2018), Effective crowdsourcing for a new type of summarization task, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, vol. 2, pp. 628–633.

Jin, D., A. Leventidis, H. Shen, R. Zhang, J. Wu, and D. Koutra (2017), Perseus-hub: Interactive and collective exploration of large-scale graphs, in *Informatics*, vol. 4, p. 22, Multidisciplinary Digital Publishing Institute.

Kairam, S., and J. Heer (2016), Parting crowds: Characterizing divergent interpretations in crowdsourced annotation tasks, CSCW '16.

Kalra, N., and S. M. Paddock (2016), Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?, *Transportation Research Part A: Policy and Practice*, *94*, 182–193.

Kamar, E., S. Hacker, and E. Horvitz (2012), Combining human and machine intelligence in large-scale crowdsourcing, in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 467–474, International Foundation for Autonomous Agents and Multiagent Systems.

Kaspar, A., G. Patterson, C. Kim, Y. Aksoy, W. Matusik, and M. Elgharib (2018), Crowd-guided ensembles: How can we choreograph crowd workers for video segmentation?, in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 111:1–111:12, ACM, New York, NY, USA.

Kaur, H., M. Gordon, Y. Yang, J. P. Bigham, J. Teevan, E. Kamar, and W. 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.

Kim, J. (2015), Learnersourcing: improving learning with collective learner activity, Ph.D. thesis, Massachusetts Institute of Technology.

Kim, J., P. T. Nguyen, S. Weir, P. J. Guo, R. C. Miller, and K. Z. Gajos (2014), Crowdsourcing step-by-step information extraction to enhance existing how-to videos, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 4017–4026, ACM.

Kim, S., E. Marquis, R. Alahmad, C. S. Pierce, and L. P. Robert Jr (2018), The impacts of platform quality on gig workers' autonomy and job satisfaction, in *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pp. 181–184, ACM.

Kittur, A., B. Smus, S. Khamkar, and R. E. Kraut (2011), Crowdforge: Crowdsourcing complex work, in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 43–52, ACM.

Kittur, A., J. V. Nickerson, M. Bernstein, E. Gerber, A. Shaw, J. Zimmerman, M. Lease, and J. Horton (2013), The future of crowd work, in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 1301–1318, ACM.

Konrad, J., M. Wang, and P. Ishwar (2012), 2d-to-3d image conversion by learning depth from examples, in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–22, IEEE.

Krishna, R., et al. (2017), Visual genome: Connecting language and vision using crowdsourced dense image annotations, *International Journal of Computer Vision*, *123*(1), 32–73.

Kulkarni, A., M. Can, and B. Hartmann (2012), Collaboratively crowdsourcing workflows with turkomatic, in *Proceedings of the acm 2012 conference on computer supported cooperative work*, pp. 1003–1012, ACM.

Kwolek, B. (2006), Model based facial pose tracking using a particle filter, in *Geometric Modeling and Imaging–New Trends (GMAI'06)*.

Laput, G., W. S. Lasecki, J. Wiese, R. Xiao, J. P. Bigham, and C. Harrison (2015), Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds, in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 1935–1944, ACM.

Lasecki, W., and J. Bigham (2012), Self-correcting crowds, in *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pp. 2555–2560, ACM.

Lasecki, W. S. (2015), Crowd agents: interactive intelligent systems powered by the crowd, Ph.D. thesis, University of Rochester.

Lasecki, W. S. (2019), On facilitating human-computer interaction via hybrid intelligence systems, in *Proceedings of the 7th annual ACM Conference on Collective Intelligence. ACM.*

Lasecki, W. S., and J. P. Bigham (2013), Interactive crowds: Real-time crowdsourcing and crowd agents, in *Handbook of human computation*, pp. 509–521, Springer.

Lasecki, W. S., K. I. Murray, S. White, R. C. Miller, and J. P. Bigham (2011), Real-time crowd control of existing interfaces, in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 23–32, ACM.

Lasecki, W. S., C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham (2012), Real-time captioning by groups of non-experts, in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 23–34, ACM.

Lasecki, W. S., C. D. Miller, and J. P. Bigham (2013a), Warping time for more effective real-time crowdsourcing, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 2033–2036, ACM, New York, NY, USA.

Lasecki, W. S., Y. C. Song, H. Kautz, and J. P. Bigham (2013b), Real-time crowd labeling for deployable activity recognition, in *Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 1203–1212, ACM.

Lasecki, W. S., M. Gordon, D. Koutra, M. F. Jung, S. P. Dow, and J. P. Bigham (2014a), Glance: Rapidly coding behavioral video with the crowd, in *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 551–562, ACM.

Lasecki, W. S., C. Homan, and J. P. Bigham (2014b), Architecting real-time crowd-powered systems, *Human Computation Journal*.

Lease, M., J. Hullman, J. P. Bigham, M. S. Bernstein, J. Kim, W. S. Lasecki, S. Bakhshi, T. Mitra, and R. C. Miller (2013), Mechanical turk is not anonymous, *Social Science Research Network*.

Lee, S. W. (2018), Improving user involvement through live collaborative creation.

Lee, S. W., R. Krosnick, S. Y. Park, B. Keelean, S. Vaidya, S. D. O'Keefe, and W. S. Lasecki (2018), Exploring real-time collaboration in crowd-powered systems through a ui design tool, *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 104.

Lee, S. W., A. Willette, D. Koutra, and W. S. Lasecki (2019), The effect of social interaction on facilitating audience participation in a live music performance, in *Proceedings of the 2019 on Creativity and Cognition*, pp. 108–120, ACM.

Leng, D., and W. Sun (2009), Finding all the solutions of pnp problem, in *2009 IEEE International Workshop on Imaging Systems and Techniques*, pp. 348–352, IEEE.

Lepetit, V., F. Moreno-Noguer, and P. Fua (2009), Epnp: An accurate o (n) solution to the pnp problem, *International journal of computer vision*, *81*(2), 155.

Lin, C., M. Mausam, and D. Weld (2012a), Dynamically switching between synergistic workflows for crowdsourcing, in *AAAI Conference on Artificial Intelligence*.

Lin, C. H., Mausam, and D. S. Weld (2012b), Crowdsourcing control: Moving beyond multiple choice, in *In: Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence, UAI*.

Lin, C. H., M. Mausam, and D. S. Weld (2012c), Dynamically switching between synergistic workflows for crowdsourcing, in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pp. 87–93, AAAI Press.

Lin, D., J. Dai, J. Jia, K. He, and J. Sun (2016), Scribblesup: Scribble-supervised convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3159–3167.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014), Microsoft coco: Common objects in context, in *European conference on computer vision*, pp. 740–755, Springer.

Little, G., L. B. Chilton, M. Goldman, and R. C. Miller (2010), Turkit: human computation algorithms on mechanical turk, in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 57–66, ACM.

Liu, C., J. Kim, and H.-C. Wang (2018), Conceptscape: Collaborative concept mapping for video learning, in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 387, ACM.

Long, J., E. Shelhamer, and T. Darrell (2015), Fully convolutional networks for semantic segmentation, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lowe, D. G. (1991), Fitting parameterized three-dimensional models to images, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5), 441–450.

Lu, C.-P., G. D. Hager, and E. Mjolsness (2000), Fast and globally convergent pose estimation from video images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(6), 610–622.

Lundgard, A., Y. Yang, M. L. Foster, and W. 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.

Luther, K., A. Pavel, W. Wu, J.-l. Tolentino, M. Agrawala, B. Hartmann, and S. P. Dow (2014), Crowdcrit: crowdsourcing and aggregating visual design critique, CSCW '14.

Luther, K., N. Hahn, S. P. Dow, and A. Kittur (2015), Crowdlines: Supporting synthesis of diverse information sources through crowdsourced outlines, in *Third AAAI Conference on Human Computation and Crowdsourcing*.

MacLean, A., R. M. Young, V. M. Bellotti, and T. P. Moran (1991), Questions, options, and criteria: Elements of design space analysis, *Human–computer interaction*, *6*(3-4), 201–250.

Mao, A., E. Kamar, Y. Chen, E. Horvitz, M. E. Schwamb, C. J. Lintott, and A. M. Smith (2013), Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing, in *First AAAI conference on human computation and crowdsourcing*.

Meissner, C. A., and J. C. Brigham (2001), Thirty years of investigating the own-race bias in memory for faces: A meta-analytic review., *Psychology, Public Policy, and Law*, *7*(1), 3.

Mitchell, M., D. Bohus, and E. Kamar (2014), Crowdsourcing language generation templates for dialogue systems, in *Proceedings of the INLG and SIGDIAL 2014 Joint Session*, pp. 172–180.

Montemerlo, M., and S. Thrun (2007), Fastslam 2.0, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pp. 63–90.

Montemerlo, M., S. Thrun, D. Koller, B. Wegbreit, et al. (2002), Fastslam: A factored solution to the simultaneous localization and mapping problem, *Aaai/iaai*, *593598*.

Necker, L. A. (1832), Lxi. observations on some remarkable optical phenomena seen in switzerland; and on an optical phenomenon which occurs on viewing a figure of a crystal or geometrical solid, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *1*(5), 329–337.

Oka, K., Y. Sato, Y. Nakanishi, and H. Koike (2005), Head pose estimation system based on particle filtering with adaptive diffusion control., in *MVA*, pp. 586–589.

Orts-Escolano, S., et al. (2016), Holoportation: Virtual 3d teleportation in real-time, in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 741–754, ACM.

Ouyang, T., and Y. 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, pp. 2895–2904, ACM, New York, NY, USA.

Oyama, S., Y. Baba, Y. Sakurai, and H. Kashima (2013a), Accurate integration of crowdsourced labels using workers' self-reported confidence scores, in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 2554–2560, AAAI Press.

Oyama, S., Y. Baba, Y. Sakurai, and H. Kashima (2013b), Em-based inference of true labels using confidence judgments, in *First AAAI Conference on Human Computation and Crowdsourcing*.

Page, S. E. (2008), *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*, Princeton University Press.

Pan, X., Y. You, Z. Wang, and C. Lu (2017), Virtual to real reinforcement learning for autonomous driving, *arXiv preprint arXiv:1704.03952*.

Park, S., G. Mohammadi, R. Artstein, and L.-P. Morency (2012), Crowdsourcing micro-level multimedia annotations: The challenges of evaluation and interface, in *Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia*, pp. 29–34, ACM.

Peng, H., F. Long, and C. Ding (2005), Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8), 1226–1238.

Quinn, A. J., and B. B. Bederson (2011), Human computation: a survey and taxonomy of a growing field, in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1403–1412, ACM.

Ramakrishnan, R., E. Kamar, B. Nushi, D. Dey, J. Shah, and E. Horvitz (2019), Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution.

Rao, A., H. Kaur, and W. S. Lasecki (2018), Plexiglass: Multiplexing passive and active tasks for more efficient crowdsourcing, in *Proceedings of the AAAI 2018 Conference on Human Computation*, ACM.

Ratner, A., S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré (2017), Snorkel: Rapid training data creation with weak supervision, *Proceedings of the VLDB Endowment*, *11*(3), 269–282.

Russell, B. C., A. Torralba, K. P. Murphy, and W. T. Freeman (2008), Labelme: a database and web-based tool for image annotation, *International journal of computer vision*, *77*(1), 157–173.

Rzeszotarski, J. M., and A. Kittur (2011), Instrumenting the crowd: using implicit behavioral measures to predict task performance, in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 13–22, ACM.

Salisbury, E., S. Stein, and S. Ramchurn (2015a), Crowdar: augmenting live video with a real-time crowd, in *Third AAAI Conference on Human Computation and Crowdsourcing*.

Salisbury, E., S. Stein, and S. Ramchurn (2015b), Real-time opinion aggregation methods for crowd robotics, in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 841–849, International Foundation for Autonomous Agents and Multiagent Systems.

Sankar, A., and S. M. Seitz (2017), Interactive room capture on 3d-aware mobile devices, in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 415–426, ACM.

Saxena, A., J. Schulte, and A. Ng (2007), Depth estimation using monocular and stereo cues, in *IJCAI*.

Shah, N. B., and D. Zhou (2015), Double or nothing: Multiplicative incentive mechanisms for crowdsourcing, in *Advances in neural information processing systems*, pp. 1–9.

Smith, A., A. E. Smith, D. W. Coit, T. Baeck, D. Fogel, and Z. Michalewicz (1997), Penalty functions.

Snow, R., B. O'Connor, D. Jurafsky, and A. 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*, pp. 254–263, Association for Computational Linguistics.

Song, J. Y., R. Fok, A. Lundgard, F. Yang, J. Kim, and W. S. Lasecki (2018), Two tools are better than one: Tool diversity as a means of improving aggregate crowd performance, in *23rd International Conference on Intelligent User Interfaces*, IUI '18, pp. 559–570, ACM, New York, NY, USA.

Song, J. Y., R. Fok, J. Kim, and W. S. Lasecki (2019a), Foureyes: Leveraging tool diversity as a means to improve aggregate accuracy in crowdsourcing, *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *10*(1), 3.

Song, J. Y., S. J. Lemmer, M. X. Liu, S. Yan, J. Kim, J. J. Corso, and W. S. Lasecki (2019b), Popup: reconstructing 3d video using particle filtering to aggregate crowd responses, in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pp. 558–569, ACM.

Song, J. Y., J. J. Chung, D. Fouhey, and W. S. Lasecki (2020), C-reference: Improving 2d-3d object state reconstruction accuracy via crowdsourced joint object estimation, in *Proceedings of the ACM conference on Computer-Supported Collaborative Work (submitted)*.

Sorokin, A., D. Berenson, S. S. Srinivasa, and M. Hebert (2010), People helping robots helping people: Crowdsourcing for grasping novel objects, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2117–2122, IEEE.

Sternberg, R. J., and K. Sternberg (2016), *Cognitive psychology*, Nelson Education.

Su, H., C. R. Qi, Y. Li, and L. J. Guibas (2015), Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2686–2694.

Surowiecki, J. (2005), *The wisdom of crowds*, Anchor.

Swaminathan, S., R. Fok, F. Chen, T.-H. K. Huang, I. Lin, R. Jadvani, W. S. Lasecki, and J. 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*, pp. 807–815, ACM.

Szeto, R., and J. J. Corso (2017), Click here: Human-localized keypoints as guidance for viewpoint estimation, in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1604–1613, IEEE.

Tardif, J.-P. (2009), Non-iterative approach for fast and accurate vanishing point detection, in *2009 IEEE 12th International Conference on Computer Vision*, pp. 1250–1257, IEEE.

Thrun, S. (2000), Monte carlo pomdps, in *Advances in neural information processing systems*, pp. 1064–1070.

Torbert, S. (2016), *Applied computer science*, 158 pp., Springer.

Tulsiani, S., S. Gupta, D. Fouhey, A. A. Efros, and J. Malik (2017), Factoring shape, pose, and layout from the 2d image of a 3d scene, *arXiv*.

VaFRIC (2012), https://www.doc.ic.ac.uk/ ahanda/VaFRIC/.

Vernier, A. M., J. Y. Song, E. Sun, A. Kench, and W. S. Lasecki (2019), Towards universal evaluation of image annotation interfaces, in *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM.

Vijayanarasimhan, S., and K. Grauman (2014), Large-scale live active learning: Training object detectors with crawled data and crowds, *International Journal of Computer Vision*, *108*(1-2), 97–114.

Vondrick, C., D. Patterson, and D. Ramanan (2013), Efficiently scaling up crowdsourced video annotation, *International Journal of Computer Vision*, *101*(1), 184–204.

washington (2014), https://rgbd-dataset.cs.washington.edu/.

Waymo (2017), Inside waymo's secret world for training self-driving cars, https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation- facilities/537648/.

Waymo (2018), Waymo has the most autonomous miles, by a lot, https://www.forbes.com/sites/davidsilver/2018/07/26/waymo-has-the-most-autonomous-miles-by-a-lot/.

Welinder, P., S. Branson, P. Perona, and S. J. Belongie (2010), The multidimensional wisdom of crowds, in *Advances in Neural Information Processing Systems*, pp. 2424–2432, Curran Associates, Inc.

Whitehill, J., T. fan Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo (2009), Whose vote should count more: Optimal integration of labels from labelers of unknown expertise, in *Advances in Neural Information Processing Systems*, pp. 2035–2043, Curran Associates, Inc.

Wiegand, T., G. J. Sullivan, G. Bjontegaard, and A. Luthra (2003), Overview of the h. 264/avc video coding standard, *IEEE Transactions on circuits and systems for video technology*, *13*(7), 560–576.

Williams, J. J., J. Kim, A. Rafferty, S. Maldonado, K. Z. Gajos, W. S. Lasecki, and N. Heffernan (2016), Axis: Generating explanations at scale with learnersourcing and machine learning, in *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pp. 379–388, ACM.

Williams, J. J., A. N. Rafferty, D. Tingley, A. Ang, W. S. Lasecki, and J. Kim (2018), Enhancing online problems through instructor-centered tools for randomized experiments, in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 207, ACM.

Yates, J. (1993), *Control through communication: The rise of system in American management*, vol. 6, JHU Press.

Yu, L., and J. V. Nickerson (2011), Generating creative ideas through crowds: An experimental study of combination, in *Thirty Second International Conference on Information Systems*.

Yu, L., A. Kittur, and R. E. Kraut (2016), Encouraging "outside-the-box" thinking in crowd innovation through identifying domains of expertise, CSCW '16.

Yuen, J., B. Russell, C. Liu, and A. Torralba (2009), Labelme video: Building a video database with human annotations, in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1451–1458, IEEE.

Zhong, Y., W. S. Lasecki, E. Brady, and J. P. Bigham (2015), Regionspeak: Quick comprehensive spatial descriptions of complex images for blind users, in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2353–2362, ACM.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal (1997), Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, *ACM Transactions on Mathematical Software (TOMS)*, *23*(4), 550–560.