Plexiglass: Multiplexing Passive and Active Tasks for More Efficient Crowdsourcing

Akshay Rao, Harmanpreet Kaur, Walter S. Lasecki
Crowds + Machine (CROMA) Lab | MISC Group
Computer Science & Engineering and School of Information
University of Michigan – Ann Arbor
{akshayro,harmank,wlasecki}@umich.edu

Abstract

Efficiently scaling continuous real-time crowdsourcing tasks—which engage crowd workers over long periods of time to complete tasks, such as monitoring video for critical events—is challenging largely because of the cost of keeping people consistently engaged. Worse, for many continuous tasks on which progress cannot be immediately made, which we term *passive tasks*, this engagement effort is wasted until something becomes true about the environment (e.g., the annotation of a critical event cannot happen until the event is seen). In this paper, we present the idea of *passive-active task multiplexing*, in which continuous tasks that involve waiting for a change in state are completed concurrently with traditional (offline/non-real time) tasks. We then implement this idea in PLEXIGLASS, a system that answers visual queries made by blind and low-vision users more efficiently by concurrently presenting a (passive) real-time sensing task and an (active) offline visual question answering tasks to workers. We explore different approaches to accomplishing this, and validate that task multiplexing can lead to an improvement in efficiency in these settings of more than 40% in terms of overall worker time taken and 85% decrease in cost of running continuous real-time tasks. This work has implications on how real-time crowdsourcing tasks are presented to workers, and it increases the feasibility of deploying continuous real-time crowdsourcing systems in real-world settings.

Introduction

Continuous real-time crowdsourcing tasks—those that require crowd workers to engage with a task on an ongoing basis, rather than with a fixed-size task—solve an important class of problems that have applications in domains ranging from accessibility to interactive control to sensing and activity monitoring (Laput et al. 2015; Lasecki et al. 2014; 2011; 2013a; Salisbury, Stein, and Ramchurn 2015b). In many of these tasks, a majority of the time is spent on waiting for a state of the scene that allows actions to be taken (e.g., a person must enter a scene before activity recognition labels can be provided). We define these to be *passive tasks*.

Asking crowd workers to do continuous monitoring tasks is prohibitively expensive because of the long term nature of these tasks. In this paper, we study mechanisms for reducing the cost of a subset of these tasks: the passive ones that require waiting for a future state to become active. We do so by interleaving the passive continuous monitoring tasks with an active stream of microtasks. For example, observing a live video stream to identify a certain event does not require a worker's full attention until a certain state of the world arises (Salisbury, Stein, and Ramchurn 2015a). Crowd workers can perform an active stream of microtasks while passively observing the live stream for the required state.

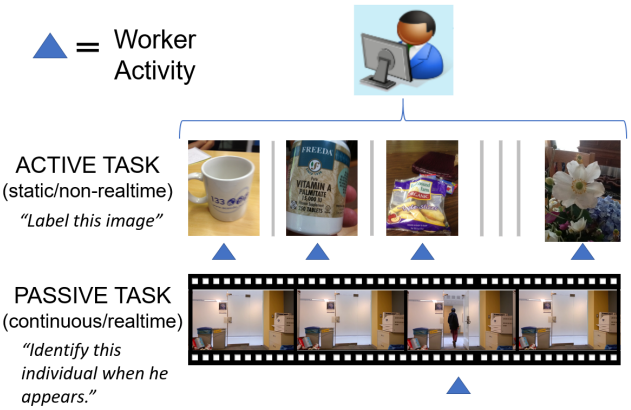

Figure 1: Overview of task multiplexing. Some continuous, real-time tasks (which we define as "passive" continuous tasks) can be completed along microtasks like the ones ubiquitous on crowdsourcing platforms like Amazon Mechanical Turk (e.g., image labeling).

We present PLEXIGLASS, a system that instantiates our multiplexing approach (i.e., interleaving active and passive streams of tasks) for answering visual questions for blind and low-vision users. Visual question answering (VQA) systems can provide blind and low-vision users with better access to the world around them by providing feedback on visual scenes via images or video. Unfortunately, automated systems alone cannot robustly handle all user queries in real-world settings because it requires an understanding of *both* natural language and visual content. Prior systems, such as VizWiz (Bigham et al. 2010) and Chorus:View (Lasecki et al. 2013b), addressed this limitation by leveraging crowd workers to quickly and reliably answer visual questions, but these approaches do not scale for continuous monitoring tasks. Using our approach, we are able to accurately answer

visual questions about a live video stream with 42.5% less total human time, and an 85% reduction in cost. We discuss the implications of this improvement in worker efficiency for future crowdsourcing systems that aim to scale up passive continuous monitoring tasks in real-time deployments.

Envisioned Applications

We frame our discussion of task multiplexing and the PLEXIGLASS system around passive tasks in the domain of accessibility for people with visual impairments. Consider Gabrielle, a visually impaired user visiting a remote office location that her company operates. The building is equipped with experimental robot assistants that accompany all visitors to help answer any questions they may have during their stay. Gabrielle can navigate just fine on her own, but wants to keep an eye out for her colleague, Jose, who she believes may be in the same office today. Unfortunately, this kind of distant visual search is impossible for Gabrielle, so she makes a request to the personal assistant robot to let her know if it spots her colleague. Automated computer vision methods would need to be trained with significant amount of data to recognize Jose, and even then, might not operate reliably in all settings (e.g., a new office building). The robot thus hands this task off to the crowd – workers helping to complete this task see a live video stream of what the robot sees, and a profile picture drawn from Jose's work profile. The workers need to monitor the video feed for the specified event, which may occur at any point throughout the span of Gabrielle's interaction with the robot assistant.

Our example above demonstrates a passive task because crowd workers cannot respond affirmatively until they see Jose in the video stream, and need not check anything in the video unless at least *some* person is currently present in it. More generally, many of these states are presented as natural language queries currently not understood or robustly detected by automated approaches, so the crowd is ultimately responsible for monitoring in these situations. While ambient sensors powered by the crowd like the one in our example have been proposed (Lasecki et al. 2013a; Laput et al. 2015), this potentially high ratio of inactive, but necessary, time on tasks makes such systems prohibitively expensive. In our example, a set of workers may spend most of their day keeping an eye out for Jose, costing $210 ($10/hour * 7 hours/day * 3 workers) in the process. As a result, such an application is infeasible in practice, even if potentially helpful in making spaces more accessible.

The video monitoring application guides our initial experiments, but our multiplexing approach is applicable broadly to any task based on streaming media containing sparse events. These include audio and text alert systems (e.g., airport notification screens); video processing (e.g., Closed Circuit Television (CCTV) footage for criminal activity); audio monitoring; or activity recognition as described above. In most of these cases, the state being observed is not static. That is, the event being monitored can change based on the situation. Designing an automated system that detects a general event does not work for these dynamic instances, whereas crowds can monitor these media streams based on any natural language query.

Related Work

Our work builds on previous research in crowdsourcing and access technology. Most directly, we extend the literature related to crowd-powered sensing applications, and on the effects of interruptions and multitasking in crowd work.

Crowd-powered Sensing

Crowdsourcing has become a key method for powering and training systems that aim to understand visual scenes. Glance (Lasecki et al. 2014) and Legion:AR (Lasecki et al. 2013a) are crowd-powered tools that focus on recognizing human behaviors in video inputs for gaining insights into video and for real-time activity recognition. Zensors (Laput et al. 2015) uses a similar set of approaches to not only create crowd-powered visual sensors on-demand, but also train computer vision systems during deployment to eventually automate it over time. Accessibility research has highlighted the potential of using crowd-powered sensing systems to create more powerful access technologies. For example, VizWiz (Bigham et al. 2010) uses crowd-powered sensing to answer visual questions for blind and low-vision users in near-realtime. Adrenaline (Bernstein et al. 2011) explored methods for quickly recruiting workers to a task to make these systems more responsive.

Continuous Crowdsourcing

In order to support more rich, on-going interaction, crowdsourcing has moved beyond traditional microtask settings that leveraged discrete, context free units of work. Legion (Lasecki et al. 2011) was the first system to enable continuous real-time crowdsourcing tasks. Specifically, it allowed end users to control their desktop via natural language commands by handing off collective control of a user's desktop interface to the crowd. Toolkits like LegionTools (Gordon, Bigham, and Lasecki 2015) have simplified the process of organizing large, synchronous groups of workers for experimental tasks, and have helped make such applications more common (Lasecki et al. 2014; Salisbury, Stein, and Ramchurn 2015a; 2015b). Chorus:View (Lasecki et al. 2013b) leverages this model to improve access to the world for people with visual impairments by having multiple workers to engage in a continuous interaction with the user, enabling more interactive (conversational) assistance that can be as much as an order of magnitude faster than prior (microtask-based) approaches to visual question answering. Our work builds on the idea of continuous crowdsourcing by introducing a method for interleaving active and passive crowdsourcing tasks, thus enabling faster and more cost-efficient continuous real-time crowdsourcing approaches.

Interruptions and Task Interleaving

Prior work has shown that interruptions can cause significant disruptions to work (Cutrell, Czerwinski, and Horvitz 2001; Czerwinski, Horvitz, and Wilhite 2004), and common practices in crowd work often incur similar penalties (Lasecki et al. 2015). However, research has also highlighted the possible benefits of periodic interruptions during tasks that require passive levels of attention over long periods of time.

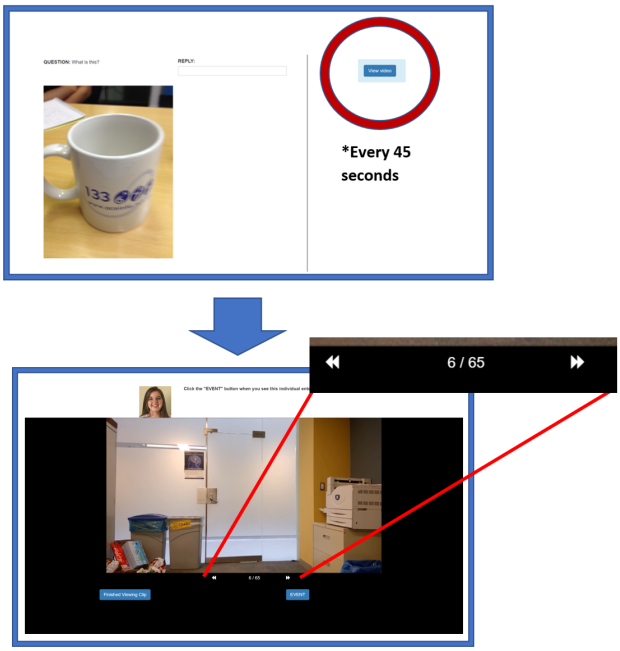

Figure 2: PLEXIGLASS worker interface for the Task Switching condition. Workers periodically get an alert that prompts them to check on the status of a video stream containing potential events of interest.

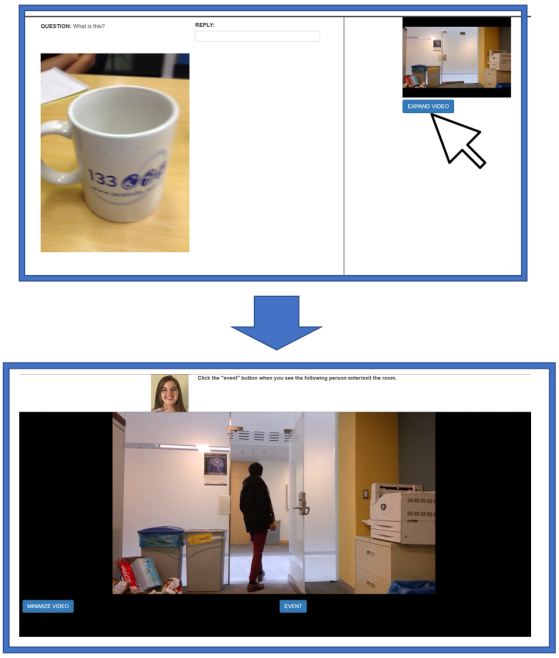

Figure 3: PLEXIGLASS worker interface for the Picture-in-Picture condition. Workers can monitor a video stream containing potential events of interest in a small window embedded within a larger interface for completing microtasks.

Work by Cabon et al. (Cabon, Coblentz, and Mollard 1990) on interruption of "monotonous" activity highlights how the periodic inclusion of "sustained attention tasks" helps maintain the attention of individuals engaged in long "vigilance tasks." We explore a parallel idea, codified as "active" and "passive" tasks within the context of crowd work.

More recent work has applied this idea in the context of crowdsourcing. As more crowd-powered systems augment automation, crowd workers are often engaged with streams of similar, monotonous tasks. To that end, Dai et al. (Dai et al. 2015) emphasize the benefits of "micro-diversions" in improving worker retention rate. Similarly, Elmalech et al. (Elmalech et al. 2016) studied the performance of crowd workers as they completed video monitoring tasks (sifting through CCTV footage), and found that the strategic insertion of "dummy events" into a stream of video helped maintain worker attention over long periods.

As a result of the anonymous and remote nature of crowd worker recruitment, expectations about crowd worker behavior are often unclear or hidden to requesters. For example, workers may choose to multitask while completing requests on platforms such as MTurk, either on multiple MTurk tasks or with entertainment / social media outlets. Since inattentiveness to tasks often leads to lower quality responses and lower productivity; some researchers have tried to mitigate this through automatic interventions to encourage workers to stay on task (Gould, Cox, and Brumby 2016). Our proposed approach focuses instead on encouraging multitasking (which we relate to the idea of "multiplexing" from the Electrical Engineering signals literature) in specific settings, with the goal of improving overall worker productivity. As we will see, *how* this is done has a significant impact of worker performance.

PLEXIGLASS System

To explore task multiplexing, we created PLEXIGLASS, a system that answers visual questions for blind and low-vision users. PLEXIGLASS uses a modular architecture that allows different active and passive tasks to be served using the same worker interface. For our experiments, we use video for continuous passive tasks, and images with text queries for offline active tasks.

System Design

The goal of PLEXIGLASS's design is to allow crowd workers to interleave multiple MTurk tasks while mitigating the productivity costs associated with shifting attention between items. Given our definitions of active and passive tasks, we hypothesize that these two types of tasks leverage worker attention in ways that don't overlap: directly making progress on a task vs. waiting for a cue to respond to. The majority of our design decisions were made based on this principle.

The worker interface (Figure 2 and 3) presents workers with the active task in the main window (left side) and the passive task on the right. We make two assumptions: (1) workers will be spending the majority of their time and attention on completing active tasks while waiting for events to happen in the passive task, and (2) workers are used to processing tasks left-to-right (western ordering). The

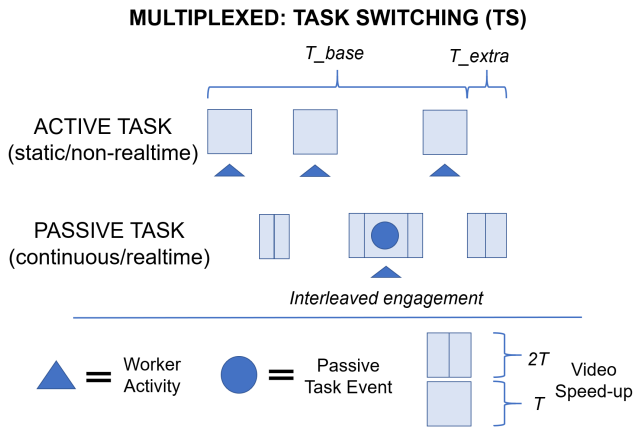

Figure 4: Task Switching approach. Workers receive alerts every 45 seconds to watch the video feed that they missed since their last check. Workers may choose to speed up the feed by 2x until they catch back up to the "live" position in the stream. Workers may also ignore the alert – this caused workers to spend an average of 95.4 seconds of extra time (T_extra) beyond the 8-minute (T_base) video, despite being able to speed up parts of playback.

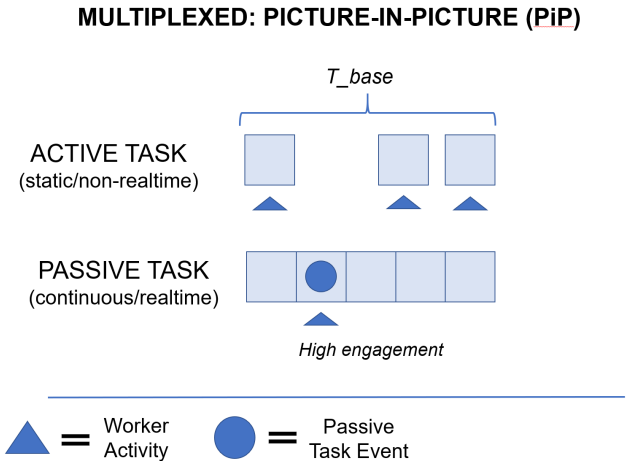

Figure 5: Picture-in-Picture approach. As workers complete active labeling tasks at their own pace, a live video feed plays within their sight, allowing them to notice and identify passive events in between. The worker is fully engaged during the entire base video time (T_base, 8 minutes).

method for accessing the passive task (modality, window configuration, etc.) can be tailored in our setup. In this paper, we explore two different approaches to presenting multiplexed active and passive tasks: i) explicit interleaving (*task switching approach*), and ii) simultaneously monitoring the video and asking workers to complete active tasks at their own pace *picture-in-picture approach*). These approaches are presented in Figure 4 and Figure 5 respectively, where the horizontal axis represents time spent on tasks, individual boxes represent an individual (active) task, and groups of boxes represent a continuous (passive) task. The interfaces for each approach are shown in Figure 2 and 3. For each of these approaches, our system is able to handle a wide range of visual events, including those described as natural language queries. If we were to implement automated approaches to aid workers in completing their tasks (e.g., a system which alerts the worker when a face is in focus, or when any movement is detected), these would be limited to certain preset queries or events that can be robustly recognized via automated approaches.

Task-Switching (TS) Approach

Our first approach allows workers to switch back and forth between their active and passive tasks (interface: Figure 2, approach: Figure 4). This is similar to what workers would do if presented with both tasks in different MTurk HITs (separate accepted tasks). However, we improve on this interaction in two ways: First, we remove the need to switch between tabs by embedding a link to the passive task on the right side of the worker UI. We do not show live video directly in this panel to avoid distracting workers from their main task. Second, we add the ability to speed up content to "catch up" with the live stream that was missed while focusing on the active task. This is similar to the approach used by TimeWarp (Lasecki, Miller, and Bigham 2013) to allow groups of crowd workers to access delayed content while decreasing latency on an audio captioning task.

Since this interface does not require the worker to watch the video at all times, latency in response is to be expected. To minimize this, we included an alert system that would prompt workers to view sections of the video feed that they had missed until that point. Workers were not required to respond to the prompt every time, i.e., they could choose to respond to the prompt in longer time intervals. The frequency of alerts may be adjusted depending on the expected engagement of the task. While the video presented was prerecorded footage, we simulated a live viewing by only allowing workers to catch up to "present" time, relative to when the video first began. Additionally, we had the workers demonstrate their understanding of the use of the interface before routing them to the task, to ensure that a failure to identify passive events was not due to our setup.

Picture-in-Picture (PiP) Approach

Our second approach for task multiplexing is to show both tasks simultaneously, with the video live streaming on the right hand side of the worker's UI and the self-paced active task in the main (left) section (interface: Figure 3, approach: Figure 5). PiP contrasts our TS approach by assuming that distractions from the passive task are not problematic for the active task. This assumption is reasonable largely because our definition of passive tasks implies less activity / points of interest will arise in the video (though not none). Workers may choose to expand the video to full screen at any point, in order to clarify specific details of an event (e.g., matching the face of a target individual with the face of someone who entered the video frame).

Study

We tested our system with crowd workers recruited from Amazon Mechanical Turk. Workers were assigned to one of the following experimental conditions: multiplexing active and passive tasks using the TS approach or the PiP approach. Our active tasks were self-paced visual Q&A tasks (similar to VizWiz (Bigham et al. 2010) questions). For example, "Could you describe the pattern on this shirt?", paired with a corresponding photo. We did not change the ordering of active task questions for different workers. Our passive tasks were live-streaming videos of various situations in a modern office environment. While both of these tasks are in the problem space of assisting blind and low-vision users, they are representative of a broader subset of visual annotation tasks and, more generally, a broader class of media streaming tasks with sparse events being monitored.

We used two passive tasks in our experiments: a `numbering` task and an `identification` task. The `numbering` task asked workers to monitor a video feed of a small meeting room, and update a counter of the current number of people in the video frame as the video runs. The state changed three times over the course of the video for this task. For the `identification` task, workers monitored a video feed of the entrance to a room, through which people walk in and out. They were given a profile picture of one of these individuals, and were told to hit a button whenever they see this particular person enter or exit the room. In our video, five people enter and exit, only one of whom was the target person.

We limit active tasks to 8 minutes of work time to match the 8-minute video clip in our passive task. This length was chosen to provide workers with enough time to familiarize themselves with the task setting, and participate for long enough that their attention may drift (short tasks; workers are unlikely to miss content due to novelty effects). For both task types, we set the time between video alerts in our TS approach to 45 seconds. This was chosen to be well shorter than the time between events in our test cases, while still allowing the worker enough time to focus on completing microtasks. For both these values, we expect the specific setting to have little or no effect on worker performance.

Baselines

Our first baseline offers workers a traditional UI to complete active or passive tasks in isolation, which we name the `active-only` and `passive-only` conditions. This baseline recreates sequential completion of each task separately, meaning that the overall completion time will be approximately twice the individual completion time (depending on the pace of active task completion).

Multi-Window Approach In addition to administering tasks in isolation, we consider the case where workers complete tasks concurrently by opening multiple browser windows (the `Mult` condition). Similar to the payment scheme introduced in the retainer model (Bernstein et al. 2011), this method reduces costs for the requester by adjusting payment to reflect the lower effort required by the worker. For example, Bernstein et al.'s Adrenaline paid $1-2/hr for a task that required no work until a specific time, at which it required completion of a short task. In response, workers were allowed to complete multiple tasks at once to earn a more reasonable wage. We simulated this condition by placing two tasks in separate browser windows—one active task window and one passive task window—and requiring workers to complete both at the same time.

Measures of Success

For active tasks, we measure performance as the number of tasks completed in the 8-minute window. For the `numbering` passive task, workers need to correctly update a value three times over the duration of the video, each at the time the specified event occurs. We define a successful event identification as a correct answer within a 10 second window of the event occurring in the video. We chose this as an a reasonable real-time response upper-bound for our application domain. We calculate worker accuracy as a ratio of events successfully identified out of a total of three event occurrences. For the `identification` passive task, workers need to successfully identify the target individual entering the frame the one time it happens during the entire video. The outcome is binary: the worker can either succeed or fail at this task (93% of workers who eventually succeeded, succeeded within this bound). Our definition for success for this task assumes the same ± 10 seconds rule stated above.

Hypotheses

We designed our study to evaluate two hypotheses:

$H_{ActivePerformance}$: Crowd workers using a multiplexed method will perform as well as workers in the control group at completing active tasks, measured by the number of labeling tasks completed over 8 minutes. The `Mult` baseline will be the primary comparison case for this hypthothesis.

$H_{PassivePerformance}$: Crowd workers using a multiplexed method will identify events in the passive tasks (either `numbering` or `identification`) with similar accuracy as those in the control group, in terms of a ratio of events identified to events occurred. The `active-only` and `passive-only` baselines are significant for this hypothesis, since we would expect optimal performance when the worker is only given one task to focus on.

Results

We compare the time it takes to complete active and passive tasks using our TS and PiP multiplexing approaches to `active-only` and `passive-only` individual baselines and the `Mult` baseline. All significance testing is done using two-tailed, paired t-tests with a threshold of $\alpha = 0.05/3 = 0.0169$ (using Bonferroni correction). Figure 6 compares the number of active tasks completed in our baseline conditions and our multiplexing approaches, and Figures 7- 8 compare passive task accuracy across these conditions.

Individual Baselines

In our `active-only` individual baseline, workers completed an average of 40.19 tasks ($SD = 14.82$, $N = 21$).

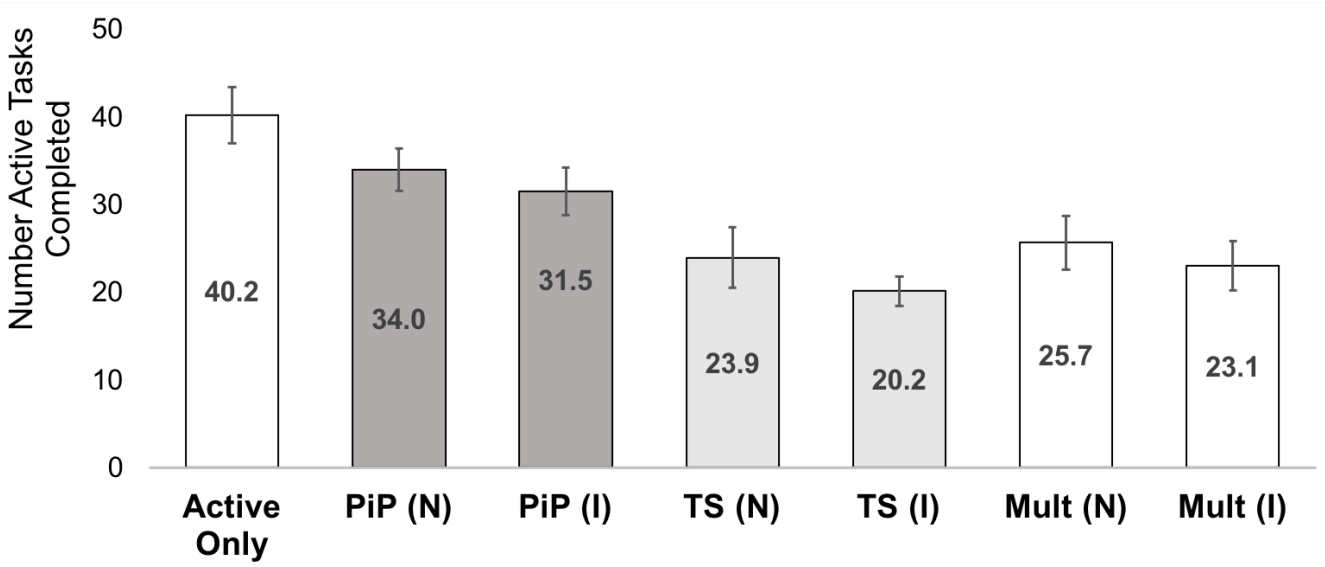

Figure 6: Number of active tasks completed for all conditions. For PiP, TS, and Mult approach, active tasks are completed in combination with either Numbering passive tasks (N) or Identification passive tasks (I).

For `passive-only` individual baseline, workers successfully identified an average of 86% ($SD = 22\%$, $N = 14$) of events for the `numbering` task with an average latency of 4.19 seconds ($SD = 0.64$), and 71% workers ($SD = 2.7$, $N = 35$) successfully identified the target individual in the `identification` task with an average latency of 3.83 seconds ($SD = 0.51$).

Multi-Window Baseline

In the `Mult` baseline, workers completed an average of 25.7 active tasks ($SD = 11.8$) with the `numbering` passive task, and 23.1 active tasks ($SD = 10.5$) with the `identification` passive task. For the `numbering` passive task, workers identified 57% of events ($SD = 30\%$, $N = 14$) with an average latency of 7.84 seconds ($SD = 1.36$), and for the `identification` task, 64% of workers successfully identified the individual ($SD = 1.8\%$, $N = 14$) with an average latency of 4.79 seconds ($SD = 0.94$).

Multiplexing: Task Switching

When multiplexing using the TS approach, workers completed an average of 23.9 active tasks ($SD = 13.4$) with the `numbering` passive task, and 20.2 active tasks ($SD = 7.3$) with the `identification` task (Figure 6). Workers using this approach complete significantly less active tasks than the `active-only` baseline ($p > 0.001$ when combined with both numbering and identification passive tasks), but there is no significant difference compared to the `Mult` baseline ($p > 0.7$ with numbering passive task and $p > 0.3$ with identification passive task).

For multiplexed passive tasks, workers successfully identified 90% of events ($SD = 16\%$, $N = 13$) for the `numbering` task with an average latency of 8.79 seconds ($SD = 1.42$), and 90.5% ($SD = 1.3$, $N = 21$) workers were successful in identifying the individual for the `identification` task with an average latency of 6.87 seconds ($SD = 1.05$) (Figures 7-8). Compared to the `passive-only` individual baseline, there was no significant difference in the performance for `numbering` or `identification` tasks ($p > 0.5$ and $p > 0.09$ respectively). Compared to the `Mult` baseline, TS approach is significantly better for `numbering` tasks ($p > 0.002$) and not significantly different for `identification` tasks ($p > 0.06$).

Multiplexing: Picture-in-Picture

When multiplexing active and passive tasks using the PiP approach, workers completed an average of 34 active tasks ($SD = 8.78$) when paired with the `numbering` task, and 31.5 active tasks ($SD = 10.1$) with the `identification` task (Figure 6). Compared to the `active-only` baseline, PiP approach performed significantly worse for active tasks paired with the `numbering` task ($p > 0.1$), and significantly worse when paired with the `identification` task ($p > 0.04$). There is also a near-significant improvement when using the PiP approach compared to `Mult` ($p > 0.04$ for both passive tasks). In addition, we found that workers in the `numbering` task kept the video in "expanded" mode for about 15% ($SD = 9\%$) of the entire task completion time, while workers in `identification` kept the video expanded for 28% of the time ($SD = 12\%$).

On comparing the two multiplexing approaches with one another in terms of active task completion, we find that the PiP approach does not perform significantly different than the TS approach when paired with the `numbering` passive task ($p > 0.02$), and significantly better with `identification` passive task ($p > 0.001$). The difference in our multiplexing approaches implies that explicitly separating workers' attention between active and passive

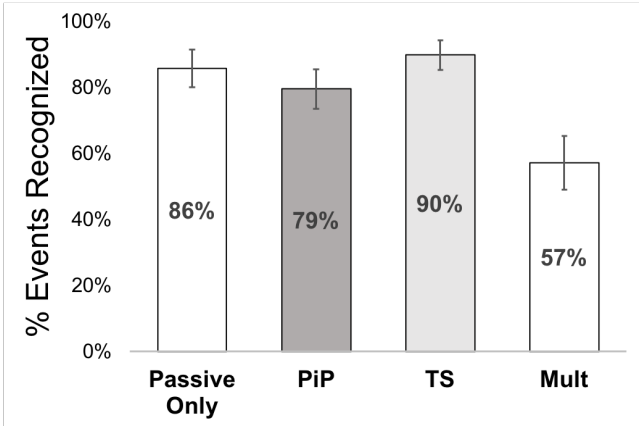

Figure 7: Performance in the `Passive Numbering` task is measured as a ratio of events successfully identified out of total event occurrences.

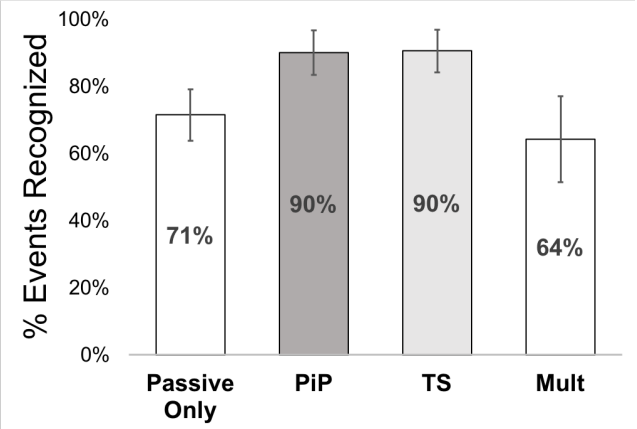

Figure 8: Performance in the `Passive Identification` task is measured as the number of correct identifications made across all tasks. Given that this identification event is binary, the number also reflects the percentage of workers that accurately did the task.

tasks (TS approach) does not work as well as allowing them to control what they want to pay attention to by placing tasks side-by-side (PiP approach). Further, PiP approach's performance is comparable or better than either baseline, confirming $H_{ActivePerformance}$.

For multiplexed passive tasks in the PiP approach, workers successfully identified 79% of events ($SD = 22\%$, $N = 13$) for the `numbering` task with an average latency of 7.24 seconds ($SD = 1.19$), and 90% ($SD = 1.3$, $N = 20$) of workers were successful in identifying the individual for the `identification` task with an average latency of 4.26 seconds ($SD = 0.75$) (Figures 7-8). Compared to the `passive-only` individual baseline, there was no significant difference in the performance for `numbering` or `identification` tasks ($p > 0.4$ and $p > 0.10$ respectively). Compared to the `Mult` baseline, PiP approach is significantly better for `numbering` tasks ($p > 0.005$) and not significantly different for `identification` tasks ($p > 0.07$). Performance between the two approaches is also not significantly different for both passive tasks ($p > 0.15$ for numbering task, $p > 0.9$ for identification task).

Comparing all passive task performances ($H_{PassivePerformance}$), event recognition accuracy does not decrease among the four testing groups (individual baselines, `Mult`, TS, and PiP) – it is either comparable or significantly better when using multiplexing approaches. This implies that despite dividing their attention between watching a continuous video stream and answering visual questions, workers in multiplexing conditions are able to identify events at the same level of accuracy as those who only needed to watch the video as the primary task. The data does indicate the possibility of workers in both multiplexed instances performing better than the baseline, but this effect is not always significant. One potential reason for this improvement could be that in multiplexing conditions, the active tasks act as "micro-diversions" that maintain worker attention and engagement for the multiplexed passive task. Further research is required to confirm this hypothesis.

Summary

Our results show that the PiP multiplexing approach significantly outperforms all baselines on active tasks. Further, this improvement does not come at the cost of accuracy on passive tasks – we have comparable passive task performance to the best-case baseline. For active tasks, there is a 42.5% decrease in total worker time with PiP approach compared to completing the individual baselines concurrently (which takes 16 minutes in total). For the passive tasks, PiP approach has comparable recall rate to the `passive-only` baseline and performs significantly better than `Mult` baseline (Figures 7- 8). Since we can do comparably well in terms of passive task monitoring while completing 34 active tasks in parallel, we get an effective cost reduction of 85% (34/40) on our passive task when using PLEXIGLASS. Thus, our previously calculated cost of $210 for a 7-hour video monitoring task may be reduced to ∼$42. While we only demonstrated these cost-savings for a particular combination of microtasks and video monitoring, our approach generalizes to other streaming media settings with sparse events, providing a means of drastically reducing the cost of continuous crowd-powered systems.

Our first hypothesis that crowd workers using a multiplexed method will perform as well as workers in the baseline groups ($H_{ActivePerformance}$) is confirmed by our findings for active tasks with PiP approach. Our second hypothesis, $H_{PassivePerformance}$, which states that crowd workers using multiplexed methods will identify events in passive tasks with a similar level of accuracy as the baselines is also confirmed using the PiP approach – we see comparable or better performance for both metrics.

Discussion

Our results suggest task multiplexing can improve crowd worker productivity for passive continuous tasks despite the presence of (typically-detrimental) workflow interruptions.

Prior work has repeatedly demonstrated that workflow interruptions can have a significant negative effect on overall productivity (Czerwinski, Horvitz, and Wilhite 2004; Iqbal and Horvitz 2007), including in crowdsourcing settings (Lasecki et al. 2015). However, unlike prior work, our work targets a newly-defined category of tasks—passive continuous tasks—and shows that these tasks are amenable to a form of task interleaving (multiplexing) that we would expect to be distracting in other settings. The dramatic reduction in task cost we present above can have an important affect on where continuous crowdsourcing can be applied. In our earlier example with Gabrielle, this improvement reduces the operating cost of keeping an eye out for Jose in the building from $210 to just $42.

Of course, our approach does not present a perfect solution to scaling continuous crowd tasks. Paying nearly $42 for our example visual search application still prevents the approach from being feasible for someone with a visual impairment funding their own access technology. However, it does take a large step (over 85% savings) towards this goal. Further, as a result of this improvement, companies and large organizations may now be able to afford to provide visual question answering support via a tool like PLEXIGLASS. Our core approach works for general combinations of crowd tasks and can be re-used in future systems that may even design complementary ways to further improve efficiency.

Future work may explore how to further optimize methods for multiplexing tasks, such as varying the frequency of interruptions in order to benefit more from "breaks" after long spans (Dai et al. 2015). Knowing that interruptions could improve productivity on long-running, monotonous tasks, creates a well-defined space within crowdsourced work for a multiplexing method with interruption frequencies tailored to maximize worker engagement. Additionally, providing more intuitive control over the multiplexing interface and characterizing worker performance on different combinations of tasks holds the promise of creating a broad set of design guidelines for system designers. We hope our work sparks considerable future work in this area.

Finally, PLEXIGLASS serves as a model for systems that aim to help crowd workers maximize their own productivity as well as their community's. After accepting a task, a worker could annotate it as "passive" and the system can help quickly find other (compatible) tasks for the worker to complete in unison. This serves to help the worker increase their effective pay rate, while also providing an annotation that can be shared with the broader community of crowd workers to help others benefit from this knowledge as well.

Conclusion

In this paper, we have presented an approach that lets us improve the efficiency of completing passive crowd tasks by adding active tasks that can be completed in parallel. Because passive tasks are characterized by waiting on a pending state of the world (a sparse event), they often do not require the same continual attention that active tasks do. We implemented our solution in PLEXIGLASS, a system that multiplexes passive and active visual questions from people with visual impairment. In PLEXIGLASS, answering continuous passive tasks costs requesters (end users) >85% less than traditional systems would, with a 40% reduction in overall worker time spent. Our work is the first to explore and define passive tasks, and presents approaches that we hope will open new lines of work on making continuous crowdsourcing systems more efficient by thinking about the broader ecosystem of tasks and how they can be considered jointly for mutual benefit.

Acknowledgements

We would like to thank Stephanie O'Keefe, Gaole Meng, Zihan Li, Tianle Lu, Sinmisola Kareem, Ashley Foster, Clement Sutjiatma, and Varun Kutirakulam for their valuable input throughout this work. We also thank the reviewers for their feedback; and finally, all of our study participants for their help. This work was supported in part by Mcity, the Toyota Research Institute, and the University of Michigan.

References

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

Bigham, J. P.; Jayant, C.; Ji, H.; Little, G.; Miller, A.; Miller, R. C.; Miller, R.; Tatarowicz, A.; White, B.; White, S.; 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*, 333–342. ACM.

Cabon, P.; Coblentz, A.; and Mollard, R. 1990. Interruption of a monotonous activity with complex tasks: effects of individual differences. In *Human Factors Society*.

Cutrell, E.; Czerwinski, M.; and Horvitz, E. 2001. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. In *Proc. INTERACT*, 263–269.

Czerwinski, M.; Horvitz, E.; and Wilhite, S. 2004. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 175–182. ACM.

Dai, P.; Rzeszotarski, J. M.; Paritosh, P.; and Chi, E. H. 2015. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 628–638. ACM.

Elmalech, A.; Sarne, D.; David, E.; and Hajaj, C. 2016. Extending workers' attention span through dummy events. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*.

Gordon, M.; Bigham, J. P.; and Lasecki, W. S. 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, 81–82. ACM.

Gould, S. J.; Cox, A. L.; and Brumby, D. P. 2016. Diminished control in crowdsourcing: an investigation of crowdworker multitasking behavior. *ACM Transactions on Computer-Human Interaction (TOCHI)* 23(3):19.

Iqbal, S. T., and Horvitz, E. 2007. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 677–686. ACM.

Laput, G.; Lasecki, W. S.; Bigham, J. P.; Wiese, J.; Xiao, R.; and Harrison, C. 2015. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

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

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

Lasecki, W. S.; Thiha, P.; Zhong, Y.; Brady, E.; and Bigham, J. P. 2013b. Answering visual questions with conversational crowd assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 18. ACM.

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

Lasecki, W. S.; Rzeszotarski, J. M.; Marcus, A.; and Bigham, J. P. 2015. The effects of sequence and delay on crowd work. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 1375–1378. ACM.

Lasecki, W. S.; Miller, C. D.; and Bigham, J. P. 2013. Warping time for more effective real-time crowdsourcing. In *Proceedings of the International ACM Conference on Human Factors in Computing Systems*, 2033–2036. ACM.

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

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