Going the Distance: Human Computing to Find Accessible Routes in Real-Time

Dianna Hu
Harvard University
dhu@college.harvard.edu

Joy Ming
Harvard University
jming@college.harvard.edu

ABSTRACT

Trying to find a wheelchair-accessible route often results in a "greedy" physical algorithm where the person travels as far as they can before rerouting and trying again. However, crowdsourcing technology can provide a much simpler approach to this problem. We describe a human computation approach that uses a section, label, and route workflow, leveraging workers on Amazon Mechanical Turk using Google Maps and Google Street View to check the accessibility of a path and produce accessible paths in a collaborative manner. In this paper, we describe in detail the design challenges and iterative methods used to carefully outline each step of the workflow as well as conduct a preliminary pilot test of this system.

INTRODUCTION

Motivation

Locally traveling from A to B, for most, is as simple as looking up and following a pre-computed route using Google Maps. However, to others, poorly maintained sidewalks, missing curb ramps, and other obstacles present accessibility challenges. The Google Maps route presents no indication of whether its suggested routes are obstacle-free. Often the answer to this question is solved by a "greedy" physical algorithm of simply traveling along the suggested route and, if necessary, re-routing when an obstacle is encountered. However, this approach is costly and frustrating for the traveler. Even if there exist repositories of accessibility information, it is uncommon for such information to be easily visualized or sufficiently updated. We believe we can do better.

Crowdsourcing aided by Amazon Mechanical Turk has been able to approach a wide variety of problems, including, but not limited to document editing and protein folding [8, 4]. While human computation has been applied to some accessibility and mapping tasks, this space of route-finding has not been completely explored [11, 5, 7].

We present a system that will proxy the user's physical investigation of the obstacles that may appear in a given route. The system will reveal potential obstacles, evaluate their severity, and re-route to the shortest accessible route based on these findings. To do this, we present these problems in sections to workers on Amazon Mechanical Turk (MTurk) to get feedback on the accessibility of a given route and change the route if obstacles are encountered using Google Maps and Google Street View (GSV).

Breaking the problem down into a palatable workflow of parts that are both easily understandable but still relatively substantial is a challenging design question. The entire process needs to be completed in reasonable time and cost. Additionally, the data will be noisy because road conditions are constantly changing due to weather, construction, and traffic. Therefore, workers have to spend time analyzing the data they are presented. Another challenge is the coordination of many workers working in parallel at each stage or working cumulatively on other workers' results. Furthermore, this approach combines verification and re-routing.

Approach

We present a human computation based solution to the problem of finding accessible routes. Given a routing question such as "How do I get from 58 Plympton Street to 10 Garden Street in Cambridge, MA?", the system will check the accessibility of the top route given by Google Maps. Workers will *section* each of the routes by intersection, *label* the accessibility of each of these paths and intersections, and attempt to *route* a path that has the least number of inaccessible sections.

If there are no inaccessible sections, the route is returned to the user. Otherwise, the unlabeled sections will then be put into the beginning of the pipeline and undergo the same *section*, *label*, and *route* process until a fully accessible route is found.

Contributions

This project presents the following contributions:

- A novel application of human computation to the problem of finding accessible routes to produce a simple-to-use system as a contribution in accessibility technologies.
- The Section-Label-Route workflow for crowdsourcing to break down the complicated task in a pipeline that includes an overall looping structure and task parallelization can be applied to other tasks that require re-routing.
- The design and development of each task assigned to a worker that leverages of the human strength of visually solving map-based questions from a global perspective to produce collective wisdom to answer a larger question.
- The iterative process of design and evaluation of the usability and accuracy of this system through laboratory and Amazon Mechanical Turk studies.

Paste the appropriate copyright statement here. ACM now supports three different copyright statements:
• ACM copyright: ACM holds the copyright on the work. This is the historical approach.
• License: The author(s) retain copyright, but ACM receives an exclusive publication license.
• Open Access: The author(s) wish to pay for the work to be open access. The additional fee must be paid to ACM.
This text field is large enough to hold the appropriate release statement assuming it is single spaced.
Every submission will be assigned their own unique DOI string to be included here.

(a) Step 1: Section

(b) Step 2a: Label (paths)

(c) Step 2b: Label (intersections)

(d) Step 3: Route

Figure 1: Visual depiction of the workflow.

RELATED WORK

Crowdsourcing

Human computation has gained popularity in its potential to engage a geographically distributed workforce to complete complex tasks on demand and at scale [12]. However, this project focuses specifically on two main aspects of crowdsourcing: workflow and collaboration.

Workflow

Crowd workflows are most successful at highly targeted tasks and there are many ways to approach splitting up a larger problem [12]. Soylent, a word processing interface that enables writers to call on Mechanical Turk workers to shorten, proofread, and otherwise edit parts of their document on demand, presents the crowd programming pattern of Find-Fix-Verify [4]. Another example is PlateMate, a system that allows users to take photos of their meals and receive estimates of food intake and composition, which presents the workflow with the stages of Tag-Identify-Measure [15]. However, these approaches have been generally linear to some extent and have not fully introduced the concepts of loops.

Collaboration

Crowdsourcing has focused on small, focused tasks that can be performed independently [12]. However, many complex tasks require a certain level of coordination and interdependence between various components that interact, allowing MTurk workers to broaden the range and complexity of tasks that can be accomplished through crowdsourcing. For instance, collaborative crowdsourcing mechanisms have made significant strides in text translation, including rapid-time translation of texts from Creole to English to facilitate emergency responses after the Haiti earthquake [2]. Corresponding workflows for collaborative text translation have proposed a structure that layers the levels of linguistic expertise involved [14].

Our work proposes to create a collaborative environment in which the results of one task depends upon the results of its predecessor in a cascade of potential route sections to evaluate. In this research, we replace the spontaneous collaboration exhibited in Ambati et. al. 2012 with a more structured setting that depends not on expertise level but rather on iterative separation and aggregation steps of accessibility mapping in small sections with endpoints ultimately aligning to produce a single, unified accessible route [2].

Crowdsourcing for accessibility

Bigham & Ladner 2011 argue that because people with disabilities have always solicited the assistance of others to make accessible what their own senses could not, they are early adopters of interactive crowdsourcing strategies [6]. Entire organizations have been formed around volunteer or paid tasks such as sign language interpretation, real-time captioning, personal assistance, or reading support services [6].

Most technologies have been designed with this idea of "information translation" in mind. One example is VizWiz, which allows individuals who are blind to take a picture, speak a question, and have it answered by workers on Amazon Mechanical Turk quickly and cheaply [5]. There are also projects that translate from audio to visual through remote sign language interpretation [16].

These technologies have effectively applied crowdsourcing technology in a real-world setting by making the answers relevant in real-time, as per the Legion system of tools [13, 3]. However, these technologies are still limited in their scope, assuming problems that are in the *abstract* space of information. For example, VizWiz is translating from visual inputs to audio outputs. However, these examples of technology are not able to impact the *physical* space.

Mapping

Human Sensors

Some projecjects extend crowdsourcing to disabilities into the physical space such as using "human sensors" to determine the conditions on the ground. An example is Tiramisu, which is a transit application that focuses on crowdsourcing information about bus locations and fullness, arrival time of buses, and reporting problems with specific information and reporting needs for riders with disabilities [17]. The IBM Citizen Sensing and IBM Breadcrumb application extend this idea by using the mobile devices to collect information in order to collaboratively generate accessibility maps [7].

However, such systems that require "human sensors" also require adequate intrinsic incentivization of individuals to participate in the information collection process, which may result in sparse data as accessibility involves on a highly specialized domain of the population. We instead employ extrinsic incentivization through MTurk for producing the complete dataset required.

Google Street View

Previous studies have successfully utilized MTurk in combination with Google Street View (GSV) imagery to crowdsource the presence and evaluation of a variety of accessibility features involved in street-level commute, including landmark locations at bus stops for individuals with visual impairments and curb ramp data for individuals who use wheelchairs [9, 11]. Furthermore, it has been shown that general accessibility issues can be identified with high fidelity between MTurk workers and first-hand accessibility experts (i.e., wheelchair-users) using static, cached GSV images [10].

Our work shifts the examination of street images to a more interactive level by allowing MTurk workers to make full use of GSV image manipulation functionality (e.g., panning and zooming) in order to optimize the ability to capture accessibility information. Furthermore, we plan to extend these existing methods of identifying accessibility features to the space of dynamically generating accessible routes based on these features. Our work also presents a more global approach in allowing workers to see the full route as well as the section they are evaluating [18].

SYSTEM DESCRIPTION

Technology stack

The system is built with a base of HTML/CSS and Javascript, utilizing the jQuery and Twitter Bootstrap frameworks as well as the Google Maps API [1]. A working version of the interface can be found at `http://www.hcs.harvard.edu/~jming/cs279/finalproject/step#` where # is replaced with the step number, either 1, 2, 3, or 4.

Workflow

The workflow presented by the system implements the novel Section-Label-Route paradigm for approaching crowdsourcing work that requires looping. Each one of the steps has to be done consecutively, (ie, section needs to be done before label and label before route), however, each of the smaller subtasks within each of the steps can be done in parallel with many workers doing different parts of the larger task.

Step 1: Section

In the first step, workers are presented with a route calculated by the Google Maps API. Workers are asked to section the route by clicking on the intersections that the route passes through. Workers can right-click to remove a marking. After they are done sectioning the route, the system receives a list of the coordinates of each of the intersections.

Step 2: Label

Then, workers are presented with different sections of the route and asked to label the accessibility of those routes. The interface the workers are presented with includes the Google Maps highlighted route on the left pane and a pane with the Google Street View at a specific point along the path. Users are instructed on how to orient themselves correctly in the GSV pane and asked to travel throughout their assigned sections labeling accessibility. The two main types of sections include path sections as well as intersections.

- **a. Path**. Workers are asked to imagine they are walking the section from the direction of the starting point to the ending point of the global route. Along the path, the workers are asked to report any *obstacles* they encounter on either the left or right pedestrian path, including construction, stairs, missing sidewalk, pothole, or "other" obstacle.
- **b. Intersection**. Then workers are asked to orient themselves as if they were crossing the intersection away from the starting point of the route and towards the ending point of the route. In that position, they then indicate the side(s) of the pedestrian path they can reach given they start at an initial side of the path.

By presenting the paths and the intersections as separate tasks, the system reduces the computational burden for each worker. Additionally, the two-fold separation overall provides more specific and comprehensive route information for determining accessible routes in the subsequent step.

Step 3: Route

In the final step, workers determine accessible routes using a map with information collected from the previous steps. The paths are highlighted yellow to for left accessibility, blue for right accessibility, green for both sides are accessible, and red for neither side is accessible. The intersections are green to indicate crossable and red to indicate not crossable based on the accessibility information of the path section before and after the intersection.

The workers are tasked with the challenge of dragging and dropping the route to indicate paths that link together sections of the road that are accessible and avoid ones that are inaccessible. This challenge is presented as a color-matching problem, where the green sections match with all possible colors,

blue and yellow sections only match with sections of the same color or green sections, and red sections do not match with any section at all.

By abstracting the task into a color-matching challenge, workers are able to leverage their intuitive knowledge of color mixation in the context of puzzle-solving, which reduces cognitive load in the accessibility space and transforms the task of finding a globally optimal route into a gamified challenge of pattern-matching and problem-solving.

Design challenges/iterations

Defining accessibility

Initially, we planned to represent the accessibility of a section with a simple binary indicator. Although such a representation was more straightforward and easier to understand, it fails to capture the different definitions of accessibility that pertain to different individuals. Therefore, we ultimately defined accessibility in terms of a feature vector that encompasses multiple facets of accessibility. Our system allows workers to identify the following accessibility challenges:

- Stairs
- Construction
- Surface problems (e.g., potholes or sidewalk cracks)
- Lack of sidewalk
- Other obstacles (with accompanying text field)

With this vector of accessibility features, our system is capable of tailoring its definition of accessibility to one that the user provides.

Task breakdown and assignment

Originally, the tasks were assigned to simulate the physical labeling of accessibility. In this case, the worker navigated a specific section of the route, from one intersection to the next, and checked for any obstacles along the path, including, but not limited to, sidewalk obstacles such as potholes as well as intersection obstacles, such as missing curb ramps.

Then we decided that sidewalk obstacles and intersection obstacles should be considered separately, simply because it is impossible to find one located in the same type of location as another; such obstacle types are mutually exclusive. Furthermore, we implemented location-tracking so that individuals are marking the exact coordinates of an obstacle rather than simply localizing the report to the given section of the sidewalk. We provided the workers with a start intersection, a middle intersection, and an end intersection so that the overlap in intersection could help increase overall coverage.

Ultimately, we also decided to separate the intersection and path information elicitation because we found that the tasks became different enough that it is easier to have an individual "specialize" in a specific type of task to reduce the overhead in task switching.

Two-sidewalk problem

We found that Google Maps routes generally focused on the main street, or the driving path, as opposed to the pedestrian paths (sidewalks) that span the two sides of the road. For our purposes, however, accessibility concerns the pedestrian paths. For instance, it may be possible for an individual to traverse a road on the left but not the right (path), and it may be possible for an individual to cross from left to right at an intersection but not from right to left (intersection). Our system must be able to account for such specialized restrictions when providing a finalized route, and we performed several design iterations to encapsulate this information.

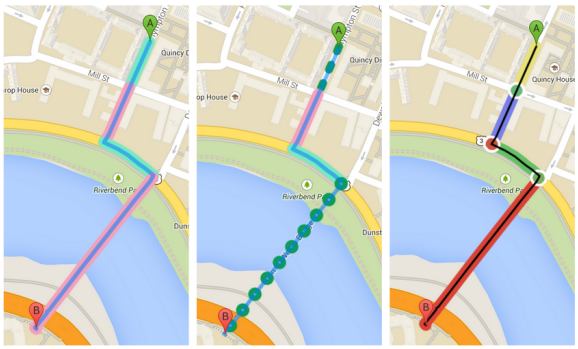


Figure 2: Design iterations of street and and intersection information for Step 3: Route.

Initially, our design displayed the binary information for streets based on examination of the main street (Figure 2a), but this design failed to capture the relevant and more complex pedestrian path accessibility information. We then represented the pedestrian path accessibility with linear and circular dashes for left and right accessibility, respectively (Figure 2b), but initial pilot studies reported that this design was unintuitive, and we were still unable to capture the intersection information.

As described in Step 3: Route of the Workflow section, we finally decided upon a color scheme wherein the color yellow implies that the left side is accessible, blue implies that the right side is accessible, green (the result of mixing yellow and blue) implies that both sides are accessible, and red implies that neither side is accessible (Figure 2c). The green and red color subset of this color scheme is extended to the intersection information as well. An intersection is colored green if it is possible to cross that intersection given the side accessibility information of the path before the intersection and of the path after the intersection.

Intersection information elicitation

Originally, we planned to have the workers match the intersection they were looking at in Google Street View with a template intersection, as seen in Figure 3. Then, they indicate with letters and numbers which crosswalks (yellow, letters) and curbramps (red, numbers) exist on a given route, giving a sense of which are missing in those rotes as well.

While this was a nifty way of translating the physical GSV image into an abstract space, this proved to be less applicable in the circuitous Boston streets where oftentimes, intersections do not fit in the simple four- and three-way templates we presented. Furthermore, translating this information into a map to be displayed for the routing step requires complex

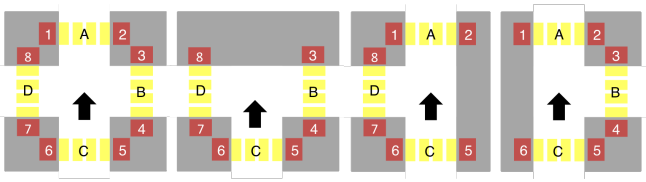


Figure 3: Example of intersection templates workers were asked to fit to the given intersection in order to examine crossability.

calculations based on the direction of their situation. However, this method did provide extra details about the intersection that could be applied any time the intersection was taken on a route, essentially increasing the overhead for potentially useful information.

We ultimately simplified the intersection information to look solely at whether it is *crossable*, or if a route exists from the starting left sidewalk to the next left and right sidewalks past the intersection in the direction of the route. This was simpler to fit all types of intersections. However, this route-specific information and resulted in multiple analyses of the same intersection for re-routing and future routing, which becomes more costly in subsequent iterations of the pipeline.

We found that the most difficult aspect of intersection information elicitation was balancing the trade-off between obtaining a lot of information about an intersection in a specific and inflexible manner, as in this example, versus simplifying and abstracting the intersection representation to get the least information possible despite possible needs to elicit the same information in future iterations of the pipeline.

EVALUATION

Study 1: In-person

We tested the instructions and interfaces of each of the steps on 8 subjects over a period of 2 weeks. The studies were conducted in-person using the same software and hardware for each. Subjects were encouraged to attempt the task as it was presented to the and ask clarifying questions about confusing instructions. From these user studies, we were able to identify some areas of improvement, explained below.

Precise language

When we first presented the Step 1: Section instructions, one of the subjects was able to complete the task, but felt the definition of *intersection* could be clarified to indicate whether they should mark the turns on the route or any time another road crossed another. We fixed this ambiguity by specifying that workers mark locations "where a street crosses the highlighted path." In Step 2: Label (Path) we were prompted to more clearly indicate the *pedestrian* nature of an obstacle as opposed to marking an obstacle on the driving road.

A similar phenomena occurred in Step 2: Label (Intersection) where we were instructing the workers to see if they were able to continue from the left side of the street to the right side of the street. First, the subject helped us clarify that it was important to specify that we were, once again, talking about the left *sidewalk* as opposed to simply the side of the driving road. Furthermore, the subject helped us formulate the need to differentiate between the sidewalks before the intersection and the ones after the intersection. However, while we did edit the words, we found that words were not enough, rather, we needed to supplement images to make it more salient for the subjects, in a process described in the next subsection.

Visual examples

For all steps within our workflow, we found the integration of visual examples with text to convey both the overall purpose and specific details of the task more clearly than text-based instructions alone. Furthermore, our visual examples coupled the real-world imagery that a user views from the Street View interface with a simplified diagrammatic image. One such example of this visual dichotomy is included the instructions for Step 2: Label (Intersection), as seen in Figure 4.

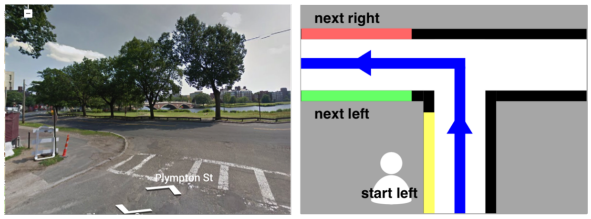


Figure 4: Visual dichotomy for instructions in Step 2: Label (Intersection)

In this example, we provided side-by-side images that depicted the same scenario of determining crossability from the left side of the street before an intersection to either the left or right side of the street after the intersection. The left image provides a real-life representation of a scenario that a worker might encounter within the Street View interface. The right image provides a simplified, abstracted representation of the information that a worker should extract from the left image, namely that it is possible for an individual to cross the intersection and continue walking on the left but not the right side of the Street View interface.

Overall, we found that the presentation of this visualization dichotomy provided the optimal coverage of task complexity and worker comprehension.

Study 2: MTurk

The tasks were deployed on Amazon Mechanical Turk asking for workers with a HIT approval rate (%) for all requesters' HITs greater than or equal to 98 and number of HITs approved greater than or equal to 5,000.

Step 1: Section

For the section step, we presented three distinct workers with the task and a reward of $0.10 for completion. We conducted a visual comparison the workers' results with a ground-truth set of points that we created. We looked for missing intersection markings (*false negatives*), extra intersection markings (*false positives*), and intersections that were off outside a tolerance threshold compared to the ground truth data points. This experiment produced results described by the following statistics:

Worker	Time (s)	Missing	Extra	Off
1	189	3	2	0
2*	91	1	0	1
3	54	0	0	0

Table 1: Summary of MTurk trial for Step 1: Section.

Although these results are not perfect, they are promising for the accuracy of this particular step of the approach. Looking at the three attempts, even though one worker's results were not correct, the aggregate of all of their results were sufficient.

In future iterations, especially with corrected information communication procedures that no longer rely on a simple copy and paste, it might be possible to have 2 instead of 3 workers on this task and maintaining the same payments or continuing to use the same number of workers and lowering the payment slightly to $0.07 each for the completion of the task.

Step 2: Label

a. Path

We first presented three distinct workers with the task with the task of marking the paths between the first five intersections and a reward of $0.50 for completion. We check the workers' results against our ground truth by marking where the users raised a flag and checking to see if there was an obstacle at that location on the reported sidewalk. In this context, we define false positives (FP) to include all instances wherein the worker identifies the path as accessible (lacking accessibility challenges) when the ground truth does not, and we define false negatives (FN) to include all instances wherein the worker identifies the path as inaccessible when the ground truth identifies the path as accessible. A similar line of reasoning follows for true positives (TP) and true negatives (TN). Based on these measures, below is a summary of the results we obtained:

Worker	Time (s)	TP	TN	FP	FN
1	248	0	3	0	1
2	1632	0	3	0	1
3	122	–	–	–	–
4	2729	0	0	0	4
5	5	0	3	0	1
6	6	0	4	0	0

Table 2: Summary of MTurk trial for Step 2: Label (Path).

These results were not as promising as those of the previous step. For instance, Worker 3 copied and pasted an incomplete string, so we only are aware of the worker's task completion time and are unsure of any accessibility problems that the worker may have identified. Workers 1 and 2 reported obstacles that were located outside of the scope of their highlighted and assigned sections. These were located at one intersection before the route and one intersection after the route. This could potentially be because the start and end instructions were not entirely clear or that they were unable to delete accidental obstacle reports.

Worker 4 considered street sewers to be potholes and thus identified pothole problems on every street. Such a response demonstrates both the worker's thoroughness and attention to detail as well as the ambiguity in our own instructions. We incorporated this feedback and made the appropriate changes to our interface for the next iteration of design and testing.

Finally, we noted that Workers 5 and 6 exhibited an exceptionally short task completion time; however, Worker 5 identified an accessibility problem within the middle of the path, which implies a non-trivial amount of time invested in the task (certainly more than five seconds). One possible explanation for the displayed time results may be that the workers first completed the task before accepting the hit and then only spent the five or six seconds copying and pasting their result from the output of the interface into the provided Mechanical Turk textbox. As a result, we exclude these two exceptionally small task completion times when computing the overall time needed for the system workflow.

Though more testing on different types of paths needs to be completed, it seems that having 3 workers being paid slightly less, around $0.40 should be adequate and fair.

b. Intersection

We presented two sets of hits, each to 3 distinct workers with the task of marking the first 4 intersections of the route and a reward of $0.50 for completion for a total of 6 workers. We conducted a comparison of the workers' results to a visually determined ground truth of those intersection. The results of the workers are presented below, summarizing the number of false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN) for the overall intersections.

W	FP	FN	TP	TN
1	0	3	11	2
3	0	4	10	2
4	2	0	14	0
5	1	2	12	1

Table 3: Summary of MTurk Trial for Step 2(b): Label (Intersections).

Once again, one out of every set of three workers had trouble copying and pasting the long string of text to the Amazon Mechanical Turk input. This will be resolved with future iterations that have a more robust approach to data storage.

Another interesting pattern is that one of the workers consistently put that all possible intersection combinations are crossable. It is difficult to tell whether this individual misunderstood the instructions or simply put less effort into the task and clicked all of the possible intersections. It is more likely the individual misunderstood the task because they exerted more than the least possible effort of marking none of the intersections as crossable. However, it is difficult to tell how the individual misunderstood the instructions. Ultimately, this is okay because on average, most intersections are fully crossable. This raises the need to have a means of feedback based on the workers' trials.

This specific case also indicates the need for a more diverse test set to truly better infer and understand the reasons why individuals are making more mistakes. For example, in this specific trial, the path ended up having three out of four intersections that were fully crossable, which seems to be a more common occurrence. However, to truly assess the robustness of the system, more interesting examples have to be tested among even more people.

Looking at these results, we see an interesting pattern among the workers in terms of their aggregate statistics. The earlier workers seem to be more *conservative*, with a total of 0 false positives, and err on the side of under-reporting instead of over-reporting. This is generally good, as discussed in similar papers [11].

Furthermore, it seems that the workers are somewhat consistent in their totals for false negatives and true positives being similar. This is interesting in terms of aggregating the results of the workers in the future. For the majority of the intersections, at least one of the workers actually completely matches up with the ground truth. Future approaches could weigh different workers' responses with the number of practice screening questions they get correct. Or this could be balanced with the amount of time the individual spent on the task, assuming more time is proportional to better results. This is further detailed in the discussion.

Looking at the time the workers put into the task (1376, 758, 504, 218, 1308, and 1160 seconds), it seems that the $0.50 payment is justified on average. It seems like in order to have a more sensible aggregate of the workers' results, using around 3 workers should be suitable.

Step 3: Route

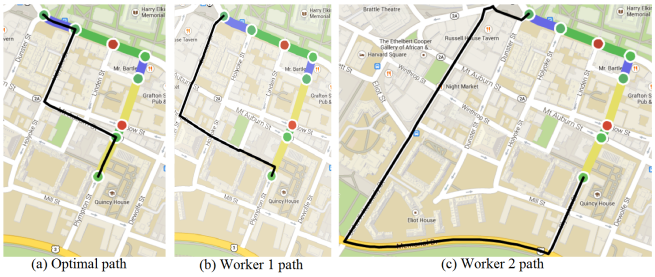
In this step we had 2 sets of 3 distinct workers complete the task, with a reward of $0.10 for the first set of workers and then $0.20 for the second set of workers to incentivize faster completion. In order to check the correctness of the workers' outputs, we overlay the resulting route on the color-coded map sections to see if what the workers produce truly is an accessible or investigatable path. In this first trial, workers were presented with only a section of the route with coloring randomly generated to create a more nuanced test case and focus on workers' abilities to understand instructions versus their ability to work with a real-life example.

We determine the optimality of a particular route according to the following criteria, listed in order of most to least important:

1. Accessibility
2. Coloration
3. Length

In particular, Criterion 1 considers the most accessible route as the one that has the fewest number of inaccessible sections. Note that uncolored sections are not considered inaccessible, i.e., Criterion 1 does not penalize the presence of uncolored sections in a particular route. However, Criterion 2 prefers the more colored route (by proportion) of two equally accessible routes because the more highly colored route exhibits higher accessibility certainty and involved fewer sections to pass to the subsequent iteration of the pipeline. Finally, if multiple paths exhibit the same optimal level of accessibility and coloration, Criterion 3 favored the shortest of the alternatives, considering length to be the most important factor after accessibility and certainty in accessibility.

With these criteria in mind, we provide a visual summary of the Step 3: Route trial with the optimal and representative worker-produced routes below:



(a) Optimal path (b) Worker 1 path (c) Worker 2 path

Figure 5: Visualization of results for Step 3: Route

Figures 5b and 5c illustrate two representative worker-produced routes with the optimal route depicted in 5a. First, we noted that both worker-produced paths satisfy Criterion 1, accessibility, considered the most important criterion. Specifically, both paths contained no inaccessible sections and thus achieved optimal accessibility. However, both paths also produced sub-optimal solutions for Criterion 2, coloration, in that neither of the produced routes contained any colored section, whereas the optimal route contains two colored sections (the first and last segments). Finally, the path produced by Worker 1 (Figure 5c) was sub-optimal with respect to Criterion 3, length, but the path produced by Worker 2 (Figure 5b) had practically the same length as the optimal route.

We believe that our prioritization of the route optimality criteria could have been better emphasized in the worker instructions. For instance, Worker 1 produced a sub-optimal coloring path with respect to Criterion 2 but was able to optimize path length, Criterion 3. This aligns with an intuitive notion to consider the shortest path to be optimal, but further emphasis on coloration (Criterion 2) and the criteria ordering in general may allow workers to adjust their conceptualization of route optimality.

Worker 2 may have been led astray by unintentional biasing from our instructions. In particular, our instructions included an example of selecting an uncolored section by necessity because no colored sections were available. Although we mentioned this rationale in our text instructions and encouraged workers to select colored sections when possible, it may have been clearer and more effective to provide an explicit visual example of this scenario first.

Finally, one worker produced an exceptionally large path (approximately 10 miles) with altered start and stop endpoints, which were not possible to alter from direct interaction with our interface alone. We suspect that this worker injected code

to attack our system, and so we excluded this worker's responses from our results. This result speaks to an essential consideration of increasing the robustness of our system as well as implementing verification steps to exclude erroneous results, whether produced mistakenly or maliciously.

DISCUSSION

Projected expenses

Based on Study 2 with the MTurk trials, an overall run of the system for a 20-minute walking route of approximately 1 mile and 25 intersections has the following costs:

step	secs	workers	sec time	sec money	total time	total money
1	1	2	60	0.10	60	0.20
2a	5	2	240	0.40	240	4.00
2b	5	2	720	0.80	720	8.00
3	1	3	300	0.50	300	1.50
total	1	3	1320	1.80	1320	13.70

Table 4: Costs of an average run of the system.

Each of the steps has either 1 or 5 sections that are running in parallel, each of which has 2-3 workers. Therefore, looking at the total time, if the task is parallelized properly using real-time tools, this is approximately the average time of all of the workers, or at least as slow as the slowest worker in the group. The total cost for this is equal to the product of the number of sections, number of workers per section, and amount paid to each worker.

Summing all of these cost estimates, this results in a total of $13.70 over 1320 seconds (22 minutes). This seems like a fairly steep cost for doing a simple routing task, which is free and happens in virtually no time for an average user. Some potential improvements to the workflow that are inspired by the pilot study include the following:

- **Increasing training**. While we cannot actively seek users who have experience in Google Maps and GSV to decrease the time per worker and effectively decrease the monetary cost per worker as well, we can try to make workers better "experts" at the task they are assigned. By including more thorough explanations and instructions, as well a "practice round" where workers are unable to continue unless they get a few preliminary exercises correct, helps increase the worker's expertise in these tasks and reduce the need for so many workers.
- **Reducing difficulty**. However, having each worker spend more time to become more qualified could potentially result in each worker spending more time on the task, which means paying each worker more. This trade-off needs to be better studied to understand the balance between expertise/time and difficulty. Another approach could be simply breaking the task down into even smaller and easier parts that more workers could do with less time and less expertise each.
- **Better aggregation**. Moreover, having better aggregation methods for including information from each of the workers could improve the data that is already being collected. Throughout the pilot study, it was found that in most cases, at least one of the workers is actually able to complete the task to almost perfect accuracy. While screening for this worker might be difficult, putting more weight on this credible worker's output through better aggregation methods could be helpful.
- **More effective worker compensation**. Another possibility is paying the workers more effectively. We are currently using a flat fee to compensate workers, meaning that each worker gets paid the same amount regardless of their performance. Amazon Mechanical Turk has a possibility for using bonuses for workers, so implementing some variation of this bonus system for incentives could be more effective.

It does seem like each individual is putting at least some effort into each of the tasks, whether this is indicated by the amount of time each individual spent on the task (generally consistent) or the amount of information the individuals provided. However, there are some better approaches to this dynamic of Amazon Mechanical Turk based on previous literature and our current experiences that can be translated into further exploration of this space.

Performance measures

Rather than using the word "accuracy," we present four main statistics that are taken to assess the closeness of the workers' results compared to a baseline we have established. We look at *false positives*, *false negatives*, *true positives*, and *true negatives*.

In the example of Step 2, where workers are asked to indicate the accessibility of a path or an intersection, *false positives* are defined as instances where the workers have incorrectly marked the section as accessible when it is not, *false negatives* are where workers have marked the section as inaccessible when it actually is, *true positives* are where workers have correctly marked an accessible section accessible, and finally *true negatives* are where workers have marked an inaccessible section inaccessible.

While true positives and negatives are sweepingly considered good and false positives and false negatives are generally considered bad, it is not as simple as that. False positives are ultimately more harmful than false negatives, because the user might be routed towards an inaccessible route versus being less inconvenienced by missing an accessible route.

As mentioned in the results for Step 2(b): Label (Intersections), it is better that the workers are generally *conservative*, erring on the side of missing crossable intersections versus misreporting an uncrossable intersection. This is promising for future trials and helps indicate the types of aggregation that needs to be done with the resulting information from the workers.

That is not to say we can simply take the results produced by the workers for granted. In the case where we no longer have the ground truth as a dataset for discrimination among correctness, it is important to consider other measures such as *reliability* and *validity*. In future iterations of this system, we need to look at the extent to which results of workers agree

to each other in addition to the extent to which the results of workers agree with the ground truth. Calculations such as Fleiss's kappa will be helpful in training classifiers to look at weighing results from the workers.

Pros and cons of workflow

In our system, we utilize an integration of both local and global task perspectives. In particular, workers are assigned a specific task that involves a series of local (inter)sections in the route, which decreases the computational burden per worker and allows for specialized focus on one type of task.

Beyond the local perspective, our interface allows workers to view the global route to gain a holistic understanding of the overall task at hand. In fact, such global route-viewing is necessary in Step 3: Route, as the optimal route with respect to both accessibility and length may not necessarily be the sum of its optimal parts. In this manner, we surpass the original greedy approach that involves restrictions to the longest locally accessible path before rerouting, and our interface instead allow workers to search for a global optimum.

We found that the current task breakdown according to the Section-Label-Route paradigm allows for the simultaneous locality of completing a specific task and global perspective of finding accessible routes. However, alternative mechanisms for task breakdown can be explored that make the overall workflow more manageable and effective. For instance, based on our analysis of the time required and performance metrics for each section, it may be possible to distribute the concentration of workers to the most demanding portions of the workflow. It may also be possible to assign multiple workers to a task simultaneously so as to increase collaboration in completing the same task more quickly or accurately.

Assumptions and Limitations

Below we describe several of the assumptions and limitations that we accept in order to abstract away certain technical difficulties in order to focus on core conceptual components:

Scope of Google Street View

Similar to the Tohme system that relies on GSV for curb ramp data [11], we assume that the views of the routes can be found reliably by both human and machine sources and that the GSV route views accurately reflect the state of the world. These paths may be inherently limited in *locality*, or including local paths that are not currently captured in GSV data or *temporality*, including images that are inconsistently timed with some street segments more recent than others.

Knowledge of the path

We acknowledge that some in-person subjects had greater knowledge of the Cambridge-specific paths that were used in the initial rounds compared to the MTurk subjects and that this prior knowledge may influence the way the subjects interact with the system.

Robustness of the prototype

The current prototype is a very beta version of the final product. One extreme limitation is its lack of database. The next iteration will no longer rely on the workers to copy and paste a long string of text, rather, store the information on the database and include only a confirmation string. Moreover, the weaknesses in the software also made it difficult for some workers to engage fully in the task, such as not being able to backtrack or remove some of their previous actions before proceeding to the next.

CONCLUSION AND FUTURE WORK

Future work

The present state of our work provides ample opportunity for future progress and expansion. First and foremost, we plan to implement some of the changes brought up in the Discussion section, including the smaller scale and direct changes to implementation:

- **More robust system.** Consistently throughout the trials, workers had difficulty copying and pasting the text from the task back to Amazon Mechanical Turk. Future iterations of this testing will include a more robust system that can better handle these database requests and also increase user capabilities to report irregularities.
- **Better selection of task sections.** In the pilot testing, workers were presented with sections of a single route. While the degree of intricacy of the route was taken into account when selecting the test cases, the route is not by any means the most interesting or intricate. Therefore, future test cases should include more corner cases.
- **Better cost-efficiency.** Future iterations should implement some of the ideas brought up to decrease expenses, including having more practice tasks to screen workers and build their expertise, as well as different incentives to reward better answers.

Once the human computation strategy has been tested, the system can be better improved for usage. The processes that are currently performed manually, mainly the transitions between the different steps, can be automated by an end-to-end backend system. Additionally, the system can be more complete by fitting it with a user-facing aspect that allows the user to ask a question and receive a detailed route. Another feature could be allowing the users to specify some of their preferences, including their own definitions of "accessibility." For example, some users may find potholes and other surface tensions to be a major obstacle that warrants rerouting and others may find they are able to pass these with less trouble.

Moreover, there are other more fundamental changes to the approach that can be taken to explore this space of human computation to find accessible routes. For instance, we may be able to combine human worker responses with an intelligent system that can provide a preliminary basis for accessible routing. For instance, the accessibility information that human workers provide in Step 2: Label (Path) and Step 2: Label (Intersection) can serve as input to a more sophisticated version of breadth-first or depth-first search. The preliminary route that emerges from this path-finding algorithm can be further refined by human workers, and the refinement can be incorporated into the intelligent system; such a positive feedback loop can increase the overall accuracy of the system in providing accessible routes.

Conclusion

In this paper, we present a system that leverages the knowledge of human computation to the new application of finding a wheelchair-accessible route. We detail some of the design challenges that we face in eliciting the proper information from the Amazon Mechanical Turk workers in a way that is effective in terms of temporal and monetary costs while also staying close to the ground truth. From these design iterations, we present the section, label, and route paradigm that we evaluate using both in-person subjects as well as online Mechanical Turk workers. We find that while there is potential for human computation to find accessible routes, this is only one small step in the journey of creating a scalable and effective system.

ACKNOWLEDGEMENTS

We express our sincere gratitude to our stellar subjects and supporters. All have been tremendously helpful in this project.

REFERENCES

1. Google maps api. `https://developers.google.com/maps/`.
2. Ambati, V., Vogel, S., and Carbonell, J. Collaborative workflow for crowdsourcing translation. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ACM (2012), 1191–1194.
3. Bernstein, M. S., Brandt, J., Miller, R. C., and Karger, D. R. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 33–42.
4. Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 313–322.
5. Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 333–342.
6. Bigham, J. P., and Ladner, R. E. Universal interactions-what the disability community can teach us about interactive crowdsourcing. *Interactions-New York 18*, 4 (2011), 78.
7. Cardonha, C., Gallo, D., Avegliano, P., Herrmann, R., Koch, F., and Borger, S. A crowdsourcing platform for the construction of accessibility maps. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, ACM (2013), 26.
8. Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., et al. Predicting protein structures with a multiplayer online game. *Nature 466*, 7307 (2010), 756–760.
9. Hara, K., Azenkot, S., Campbell, M., Bennett, C. L., Le, V., Pannella, S., Moore, R., Minckler, K., Ng, R. H., and Froehlich, J. E. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, ACM (New York, NY, USA, 2013), 1–8.
10. Hara, K., Le, V., and Froehlich, J. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, ACM (New York, NY, USA, 2013), 631–640.
11. Hara, K., Sun, J., Moore, R., Jacobs, D., and Froehlich, J. E. Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proc. of UIST* (2014).
12. Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*, ACM (2013), 1301–1318.
13. Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 23–32.
14. Munro, R. Crowdsourced translation for emergency response in haiti: the global collaboration of local knowledge. In *AMTA Workshop on Collaborative Crowdsourcing for Translation* (2010).
15. Noronha, J., Hysen, E., Zhang, H., and Gajos, K. Z. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 1–12.
16. Steinberg, A. G., Barnett, S., Meador, H. E., Wiggins, E. A., and Zazove, P. Health care system accessibility. *Journal of general internal medicine 21*, 3 (2006), 260–266.
17. Steinfeld, A., Zimmerman, J., Tomasic, A., Yoo, D., and Aziz, R. D. Mobile transit rider information via universal design and crowdsourcing. In *Proceedings from Transportation Research Board 2011 Annual Meeting* (2011).
18. Zhang, H., Law, E., Miller, R., Gajos, K., Parkes, D., and Horvitz, E. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2012), 217–226.