Tohme: Detecting Curb Ramps in Google Street View Using Crowdsourcing, Computer Vision, and Machine Learning

Kotaro Hara[1,2], Jin Sun, Robert Moore[1,2], David Jacobs, Jon E. Froehlich[1,2]
[1]Makeability Lab | [2]Human Computer Interaction Lab (HCIL)
Computer Science Department, University of Maryland, College Park
{kotaro, jinsun, dwj, jonf}@cs.umd.edu; rmoore15@umd.edu

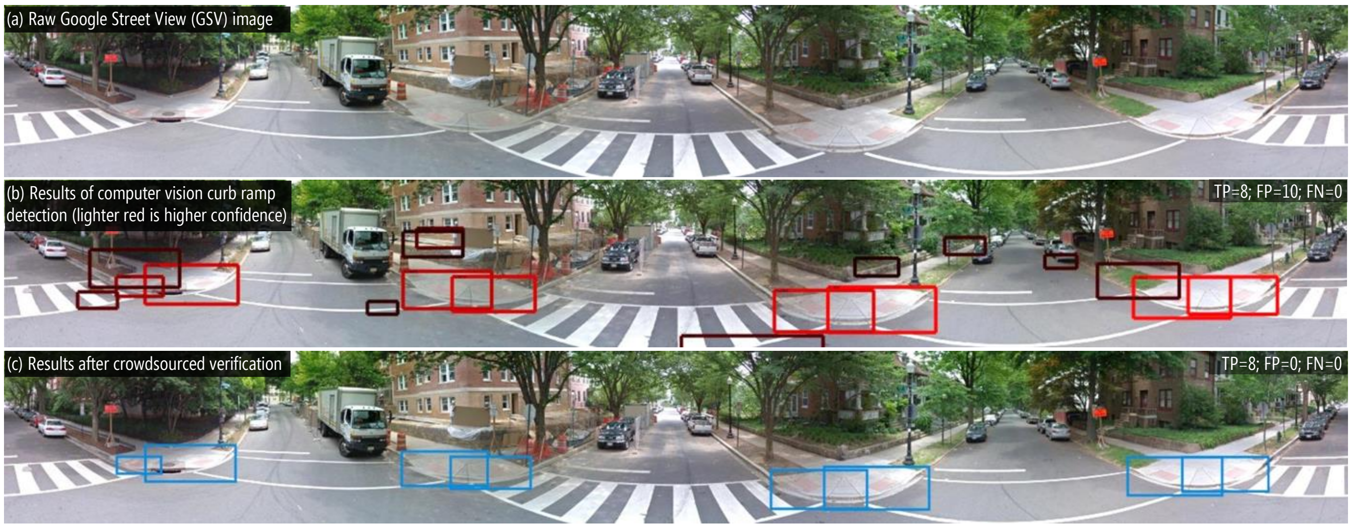

(a) Raw Google Street View (GSV) image
(b) Results of computer vision curb ramp detection (lighter red is higher confidence)
TP=8; FP=10; FN=0
(c) Results after crowdsourced verification
TP=8; FP=0; FN=0

Figure 1: In this paper, we present *Tohme*, a scalable system for semi-automatically finding curb ramps in Google Streetview (GSV) panoramic imagery using computer vision, machine learning, and crowdsourcing. The images above show an actual result from our evaluation.

ABSTRACT

Building on recent prior work that combines Google Street View (GSV) and crowdsourcing to remotely collect information on physical world accessibility, we present the first "smart" system, *Tohme*, that combines machine learning, computer vision (CV), and custom crowd interfaces to find curb ramps remotely in GSV scenes. Tohme consists of two workflows, a human labeling pipeline and a CV pipeline with human verification, which are scheduled dynamically based on predicted performance. Using 1,086 GSV scenes (street intersections) from four North American cities and data from 403 crowd workers, we show that Tohme performs similarly in detecting curb ramps compared to a manual labeling approach alone (F-measure: 84% *vs*. 86% baseline) but at a 13% reduction in time cost. Our work contributes the first CV-based curb ramp detection system, a custom machine-learning based workflow controller, a validation of GSV as a viable curb ramp data source, and a detailed examination of why curb ramp detection is a hard problem along with steps forward.

UIST '14, October 05 - 08 2014, Honolulu, HI, USA
Copyright 2014 ACM 978-1-4503-3069-5/14/10$15.00.
http://dx.doi.org/10.1145/2642918.2647403

Author Keywords
Crowdsourcing accessibility, computer vision, Google Street View, Amazon Mechanical Turk

INTRODUCTION

Recent work has examined how to leverage massive online map datasets such as Google Street View (GSV) along with crowdsourcing to collect information about the accessibility of the built environment [22–26]. Early results have been promising; for example, using a manually curated set of static GSV images, Hara *et al.* [24] found that minimally trained crowd workers in Amazon Mechanical Turk (turkers) could find four types of street-level accessibility problems with 81% accuracy. However, the sole reliance on *human* labor limits scalability.

In this paper, we present Tohme[1], a scalable system for remotely collecting geo-located curb ramp data using a combination of crowdsourcing, Computer Vision (CV), machine learning, and online map data. Tohme lowers the overall human time cost of finding accessibility problems in GSV while maintaining result quality (Figure 1). As the first work in this area, we limit ourselves to sidewalk curb ramps (sometimes called "curb cuts"), which we selected because of their visual salience, geospatial properties (*e.g.*, often located on corners), and significance to accessibility.

[1] Tohme is a Japanese word that roughly translates to "remote eye."

For example, in a precedent-setting US court case in 1993, the court ruled that the "*lack of curb cuts is a primary obstacle to the smooth integration of those with disabilities into the commerce of daily life*" and that "*without curb cuts, people with ambulatory disabilities simply cannot navigate the city*" [2].

While some cities maintain a public database of curb ramp information (*e.g.*, [1, 12]), this data can be outdated, erroneous, and expensive to collect. Moreover, it is not integrated into modern mapping tools. In a recent report, the National Council on Disability noted that they could not find comprehensive information on the "*degree to which sidewalks are accessible*" across the US [38]. In addition, the quality of data available in government systems is contingent on the specific policies and technical infrastructure of that particular local administration (*e.g.*, at the city and/or county level). While federal US legislation passed in 1990 mandates the use of ADA-compliant curb ramps in all new road construction and renovation [45], this is not the case across the globe. Our overarching goal is to design a scalable system that can remotely collect accessibility information for any city across the world that has streetscape imagery, which is now broadly available in GSV, Microsoft Bing Maps, and Nokia City Scene.

Tohme is comprised of four custom parts: (i) a web scraper for downloading street intersection data; (ii) two crowd worker interfaces for finding, labeling, and verifying the presence of curb ramps; (iii) state-of-the-art CV algorithms for automatic curb ramp detection; and (iv) a machine learning-based workflow controller, which predicts CV performance and dynamically allocates work to either a human labeling pipeline or a CV + human verification pipeline. While Tohme is purely a data collection system, we envision future work that integrates Tohme's output into accessibility-aware map tools (*e.g.*, a heatmap visualization of a city's accessibility or a smart navigation system that recommends accessible routes).

To evaluate Tohme, we conducted two studies using data we collected from 1,086 intersections across four North American cities. First, to validate the use of GSV imagery as a reliable source of curb ramp knowledge, we conducted physical audits in two of these cities and compared our results to GSV-based audit data. As with previous work exploring the concordance between GSV and the physical world [4, 9, 22, 26, 41], we found high correspondence between the virtual and physical audit data. Second, we evaluated Tohme's performance in detecting curb ramps across our entire dataset with 403 turkers. Alone, the computer vision sub-system currently finds 67% of the curb ramps in the GSV scenes. However, by dynamically allocating work to the CV module or to the slower but more accurate human workers, Tohme performs similarly in detecting curb ramps compared to a manual labeling approach alone (F-measure: 84% *vs*. 86% baseline) but at a 13% reduction in human time cost.

In summary, the primary contribution of this paper is the design and evaluation of the Tohme system as a whole, with secondary contributions being: (i) the first design and evaluation of a computer vision system for automatically detecting curb ramps in images; (ii) the design and study of a "smart" workflow controller that dynamically allocates work based on predicted scene complexity from GIS data and CV output; (iii) a comparative physical *vs.* virtual curb ramp audit study (Study 1), which establishes that GSV is a viable data source for collecting curb ramp data; and (iv) a detailed examination of why curb ramp detection is a hard problem and opportunities for future work in this domain.

RELATED WORK

We describe work in sidewalk assessment, crowdsourcing, computer vision, and dynamic workflow allocation.

Sidewalk Assessment

Traditionally, sidewalk assessment has been conducted via in-person street audits [42, 47] which are labor intensive and costly [41], or via citizen call-in reports, which are done on a reactive basis. Recent mobile apps such as *seeclickfix.com* or NYC311 allow citizens to report street infrastructure problems including damaged or missing curb ramps. However, these systems require *in situ* observation and thus do not scale as well as remote, virtual inquiry.

Crowdsourcing

Recently, Bigham *et al.* argued that current technological infrastructure provides unprecedented access to large sources of human power that can be harnessed to address accessibility challenges [6] (*e.g.*, via crowdsourcing). Examples include VizWiz [5] and Legion:Scribe [35]. Most relevant to this paper is the recent exploration of combining GSV and crowdsourcing for collecting street-level accessibility data including sidewalks [24], bus stops [26], and intersections [22]. Though this prior work demonstrates GSV as a potential accessibility data source, the studies do not examine semi-automatic methods (*e.g.*, using machine learning or CV) as we do here.

Tohme's performance is contingent on crowd workers' speed and accuracy in processing GSV imagery. Prior work exists in studying how to efficiently collect image labels (*e.g.*, [14, 43]). Su *et al.* investigated cost-performance tradeoff between majority vote based labeling and verification based data collection [43], finding quality control via verification improves cost-effectiveness. Recent work by Deng [14] explores methods of efficiently collecting multiclass image annotations by incorporating heuristics such as correlation, hierarchy, and sparsity (*e.g.*, the presence of a keyboard in an image also suggests the presence of correlated objects such as mouse and monitor); however, to our knowledge, no prior work exists on efficiently collecting image labels from crowd workers on large panoramic imagery.

Computer Vision

There is a growing body of research applying CV techniques to GSV [49–53]. For example, Xiao *et al.*

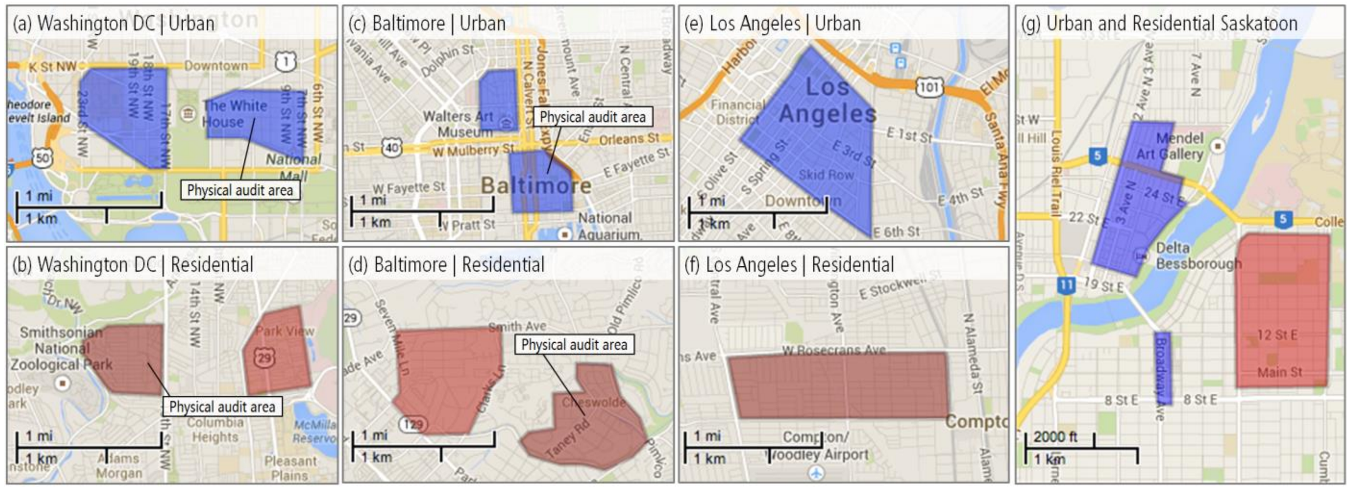


Figure 2: The eight urban (blue) and residential (red) audit areas used in our studies from Washington DC, Baltimore, LA, and Saskatoon. This includes 1,086 intersections across a total area of 11.3km^2. Among these areas, we physically surveyed 273 intersections (see annotations in a-d).

	WASHINGTON DC		BALTIMORE		LOS ANGELES		SASKATOON		OVERALL
Region Type	Downtown	Residential	Downtown	Residential	Downtown	Residential	Downtown	Residential	
Total Area (km^2)	1.52	1.13	0.73	2.24	1.91	1.89	0.74	1.13	11.28
# of Intersections	140	124	132	139	132	132	141	146	1,086
# of Curb Ramps*	818	352	476	229	358	186	321	137	2877
# of Missing Curb Ramps*	8	35	32	69	43	214	24	222	647
Avg. GSV Data Age (SD)	1.9 yrs (0.77)	1.6 yrs (0.63)	2.1 yrs (0.75)	0.4 yrs (0.65)	2.0 yrs (0.31)	0.9 yrs (0.24)	4.0 yrs (0.0)	4.0 yrs (0.0)	2.2 (1.3)

Table 1: A breakdown of our eight audit areas. Age calculated from summer 2013. *These counts are based on ground truth data.

introduced automatic approaches to model 3D structures of streetscape and building façades using GSV [49, 50]. Zamir *et al.* showed that large-scale image localization, tracking, and commercial entity identification are possible [51–53]. This work demonstrates the potential of combining CV with GSV; however, automatically detecting curb ramps or other accessibility features has not been studied.

Tohme builds on top of existing object detection algorithms from the CV community [11, 17, 46]. For example, we use *Deformable Part Models* (DPMs) [17, 18], one of the top-performing approaches in the PASCAL Visual Object Classes (VOC) challenge, a major object detection and recognition competition [17]. Despite a decade-long effort, however, object detection remains an open problem [7, 48]. For example, even the DPM, which won the "Lifetime Achievement" Prize at the aforementioned PASCAL VOC challenge, has reached 30% precision and 70% recall in 'car' detection [17]. Due to their variation in size, shape, and appearance, curb ramps are similarly difficult to detect. Consequently, we incorporate a "smart" workflow algorithm that attempts to predict poor CV performance and, in those instances, route work to human labelers.

Dynamic Workflow Allocation

Tohme uses machine learning to control its workflow for efficiently collecting data from GSV. Typical workflow adaptions include: varying the number of workers to recruit for a task [31, 48], assigning stronger workers to harder versions of a task [10], and/or fundamentally changing the task an individual worker is given [30, 36] These workflow decisions are made automatically by workflow controllers often by analyzing worker performance history, inferring task difficulty, or estimating cost.

Most relevant to our work is workflow adaptation research in crowdsourcing systems [31, 36, 48]. For example, Lin *et al.* and Welinder *et al.* rely on worker performance histories to either assign different tasks [36] or recruit different numbers of workers [48]. More similar to our work is [30, 31] who infer task difficulty via automated methods and adapt work accordingly. For example, Kamar *et al.* [31] analyzed image features with CV algorithms to predict worker behaviors *a priori* on image annotation tasks and used this to dynamically decide the number of workers to recruit.

Though similar, our work is different both in problem domain (finding curb ramps) as well as in approach. Rather than vary the number of workers per task, our workflow controller infers CV performance and decides whether to use crowd worker labor for verifications or labeling. In addition, we do not simply rely on image features or CV output to determine workflow but also contextual information such as intersection complexity and 3D-point cloud data.

DATASET

Because sidewalk infrastructure can vary in quality, design, and appearance across geographic areas, our study sites include a range of neighborhoods from four North American cities: Washington DC, Baltimore, Los Angeles, and Saskatoon, Saskatchewan (Figure 2; Table 1). For each city, we collected data from dense urban cores (shown in blue) and semi-urban residential areas (shown in red). We

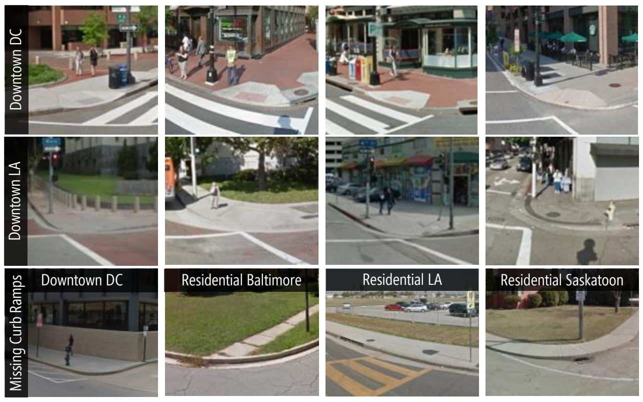


Figure 3: Example curb ramps (top two rows) and missing curb ramps (bottom row) from our GSV dataset.

emphasized neighborhoods with potential high demand for sidewalk accessibility (*e.g.*, areas with schools, shopping centers, libraries, and medical clinics).

We used two data collection approaches: (i) an automated web scraper tool that we developed called *svCrawl*, which downloads GIS-based intersection data, including GSV images, within a geographically defined region; and (ii) a physical survey of a subset of our study sites (four neighborhoods totaling 273 intersections), which was used to validate curb ramp infrastructure found in the GSV images. In all, we used *svCrawl* to download data from 1,086 intersections across 11.3km^2 (Table 1).

To create a ground truth dataset, two members of our research team independently labeled all 1,086 scenes using our custom labeling tool (svLabel). Label disagreements were resolved by consensus. From the ground truth data, we discovered 2,877 curb ramps and 647 missing curb ramps (Figure 3). Of the 1,086 scenes, 218 GSV scenes did *not* require marking a curb ramp or missing curb ramp because the location was not a traditional intersection (*e.g.*, an alleyway with no vertical drop from the sidewalk). These 218 scenes are useful for exploring false positive labeling behavior and were kept in our dataset. The remaining 868 intersections had on average 3.3 curb ramps (*SD*=2.3) and 0.75 missing curb ramps (*SD*=1.3) per intersection. A total of 603/868 intersections were marked as *not* missing any curb ramps. We use the ground truth labels for training and testing our machine learning and CV algorithms and to evaluate crowd worker performance.

At download time (summer 2013), the average age of the GSV images was 2.2 years (*SD*=1.3). As image age is one potential limitation in our approach, it is necessary to first show that GSV is a reasonable data source for deriving curb ramp information, which we do next.

STUDY 1: ASSESSING GSV AS A DATA SOURCE

To establish GSV as a viable curb ramp data source, we must show: (i) that it presents unoccluded views of curb ramps, (ii) that the curb ramps can be reliably found by humans and, potentially, machines, and (iii) that the curb ramps found in GSV adequately reflect the state of the physical world. This study addresses each of these points. Multiple studies have previously demonstrated high concordance between GSV-based audits and audits conducted in the physical world [4, 9, 22, 26]; however, prior work has not examined curb ramps specifically. Though this audit study was labor intensive, it is important to establish GSV as a reliable data source for curb ramp information, as it is the crux of our system's approach.

We conducted physical audits in the summer of 2013 across a subset of our GSV dataset: 273 intersections spanning urban and residential areas in Washington DC and Baltimore (Figure 1). We followed a physical audit process similar to Hara *et al.* [26]. Research team members physically visited each intersection, capturing geo-timestamped pictures (*Mean*=15 per intersection; *SD*=5). These images were analyzed *post hoc* for the actual audit. Surveying the 273 intersections took approximately 25 hours as calculated by image capture timestamps.

Auditing Methodology.

For the auditing process itself, two additional research assistants (different from the above) independently counted the number of *curb ramps* and *missing curb ramps* at each intersection in both the physical and GSV image datasets. An initial visual codebook was composed based on US government standards for sidewalk accessibility [32, 45]. Following the iterative coding method prescribed by Hruschka *et al.* [29], a small subset of the data was individually coded first (five intersections from each area). The coders then met, compared their count data, and updated the codebook appropriately to help reduce ambiguity in edge cases. Both datasets were then coded in entirety (including the original subset, which was recoded). This process was iterated until high agreement was reached.

Calculating Inter-Rater Reliability between Auditors

Before comparing the physical audit data to the GSV audit data, which is the primary goal of Study 1, we first calculated inter-rater reliability between the two coders for each dataset. We applied the Krippendorff's Alpha (α) statistical measure, which is used for calculating inter-rater reliability of count data (see [34]). Results after each of the three coding passes using the iterative scheme from [29] are shown in Table 2. Agreement was consistently high, with the 3rd pass representing the reliability of codes in the final code set. There was initially greater inconsistency in coding *missing* curb ramps *vs.* coding existing curb ramps, perhaps because identifying a missing ramp requires a deeper understanding of the intersection and proper ramp placement.

	PHYSICAL AUDIT IMAGE DATASET			GSV AUDIT IMAGE DATASET		
	1st Pass (α)	2nd Pass (α)	3rd Pass (α)	1st Pass (α)	2nd Pass (α)	3rd Pass (α)
Curb Ramp	0.959	0.960	0.989	0.927	0.928	0.989
Missing C. Ramp	0.647	0.802	0.999	0.631	0.788	0.999
Overall	0.897	0.931	0.996	0.883	0.917	0.996

Table 2: Krippendorff's alpha inter-rater agreement scores between two researchers on both the physical audit and GSV audit image datasets. Following Hruschka *et al.'s* iterative coding methodology, a 3rd audit pass was conducted with an updated codebook to achieve high-agreement scores—in our case, α > 0.996.

Comparing Physical vs. GSV Audit Data

With high agreement verified within each dataset, we can now compare the count scores *between* the datasets. Similar to [26, 41], we calculate a Spearman rank correlation between the two count sets (physical and GSV). This was done for both the curb ramp and missing curb ramp counts. To enable this calculation, however, we first merged the two auditor's counts by taking the average of their counts for missing curb ramps and the average for present curb ramps at each intersection. Using these average counts, a Spearman rank correlation was computed, which shows high correspondence *between* datasets: ρ=0.996 for curb ramps and ρ=0.977 for missing curb ramps ($p < 0.001$). Overall, 1,008 curb ramps were identified in the virtual audit compared to 1,002 with the physical audit; differences were due to construction. The number of missing curb ramps was exactly the same for both datasets (89).

Study 1 Summary

Though the age of images in GSV remains a concern, Study 1 demonstrates that there is remarkably high concordance between curb ramp infrastructure in GSV and the physical world, even though the average image age of our dataset was 2.2 years. With GSV established as a curb ramp dataset source, we now move on to describing Tohme.

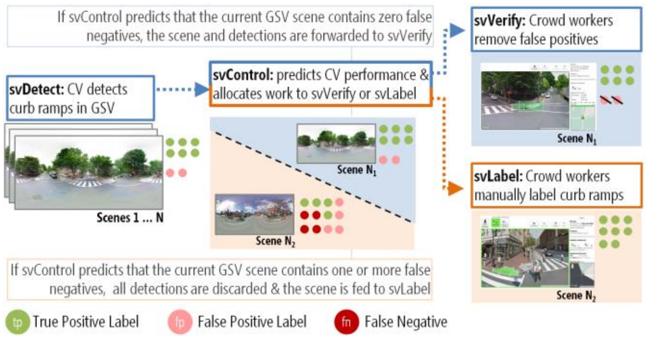
A SCALABLE SYSTEM FOR CURB RAMP DETECTION

Tohme is a custom-designed tool for remotely collecting geo-located curb ramp information using a combination of crowdsourcing, CV, machine learning, and online map data. It is comprised of four parts depicted in Figure 4: (i) a web scraper, *Street View Crawl (svCrawl)*, for downloading street intersection data; (ii) two crowd worker interfaces for finding, labeling, and verifying the presence of curb ramps called *svLabel* and *svVerify*; (iii) state-of-the-art CV algorithms for automatically detecting curb ramps (*svDetect*); and (iv) a machine learning-based workflow, called *svControl*, which predicts CV performance on a scenes and allocates work accordingly.

We designed Tohme iteratively with small, informal pilot studies in our laboratory to test early interface ideas. We also performed larger experiments on Amazon Mechanical Turk (MTurk) with a subset of our data to understand how different interfaces affected crowd performance and, more generally, how well crowds could perform our tasks. The CV sub-system, svDetect, also evolved across multiple iterations, and was trained and evaluated using the aforementioned ground truth labels. While our ultimate goal is to deploy Tohme publicly on the web, the current prototype and experiments were deployed on MTurk. Below, we describe each Tohme sub-system.

svCrawl: Automatic Intersection Scraping

svCrawl is a custom web scraper tool written in Python that downloads GIS-related intersection data over a predefined geographic region (Figure 2). It uses the Google Maps API (GMaps API) to enumerate and extract street intersection points within selected boundaries. For each intersection, svCrawl downloads four types of data:



If svControl predicts that the current GSV scene contains zero false negatives, the scene and detections are forwarded to svVerify
svVerify: Crowd workers remove false positives
svDetect: CV detects curb ramps in GSV
svControl: predicts CV performance & allocates work to svVerify or svLabel
Scenes 1 ... N
Scene N1
Scene N2
Scene N1
svLabel: Crowd workers manually label curb ramps
If svControl predicts that the current GSV scene contains one or more false negatives, all detections are discarded & the scene is fed to svLabel
Scene N2
tp True Positive Label
fp False Positive Label
fn False Negative

Figure 4: A workflow diagram depicting Tohme's four main sub-systems. In summary, *svDetect* processes every GSV scene producing curb ramp detections with confidence scores. *svControl* predicts whether the scene/detections contain a false negative. If so, the detections are discarded and the scene is fed to *svLabel* for manual labeling. If not, the scene/detections are forwarded to *svVerify* for verification. The workflow attempts to optimize accuracy and speed.

1. **A GSV panoramic image** at its source resolution (13,312 x 6,656px). This is our primary data element (*e.g.,* Figure 1).
2. **A 3D-point cloud**, which is captured by the GSV car using LiDAR [3]. The depth data overlays the GSV panorama but at a coarser resolution (512 x 256px; Figure 10). This is used by svDetect to automatically cull the visual search space and by svControl as an intersection complexity input feature.
3. **A top-down abstract map image** of the intersection obtained from the GMaps API (Figure 13), which is used as a training feature in our work scheduler, svControl, to infer intersection complexity (like the depth data).
4. **Associated intersection GIS metadata**, also provided by the GMaps API, such as latitude/longitude, GSV image age, street and city names, and intersection topology.

svLabel: Human-Powered GSV Image Labeling

In Tohme, intersections are labeled either manually, via svLabel, or automatically via svDetect. svLabel is a fully interactive online tool written in Javascript and PHP for finding and labeling curb ramps and missing curb ramps in GSV images (Figures 5-7). Unlike much previous crowd-sourcing GSV work, which uses static imagery to collect labels [22–24], our labeling tool builds on *Bus Stop CSI* [26] to provide a fully interactive 360 degree view of the GSV panoramic image. While this freedom increases user-interaction complexity, it allows the user to more naturally explore the intersection and maintain spatial context while searching for curb ramps. For example, the user can pan

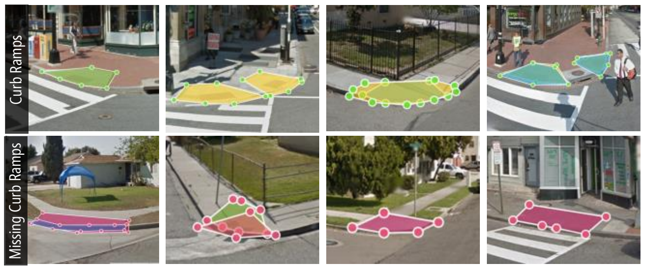
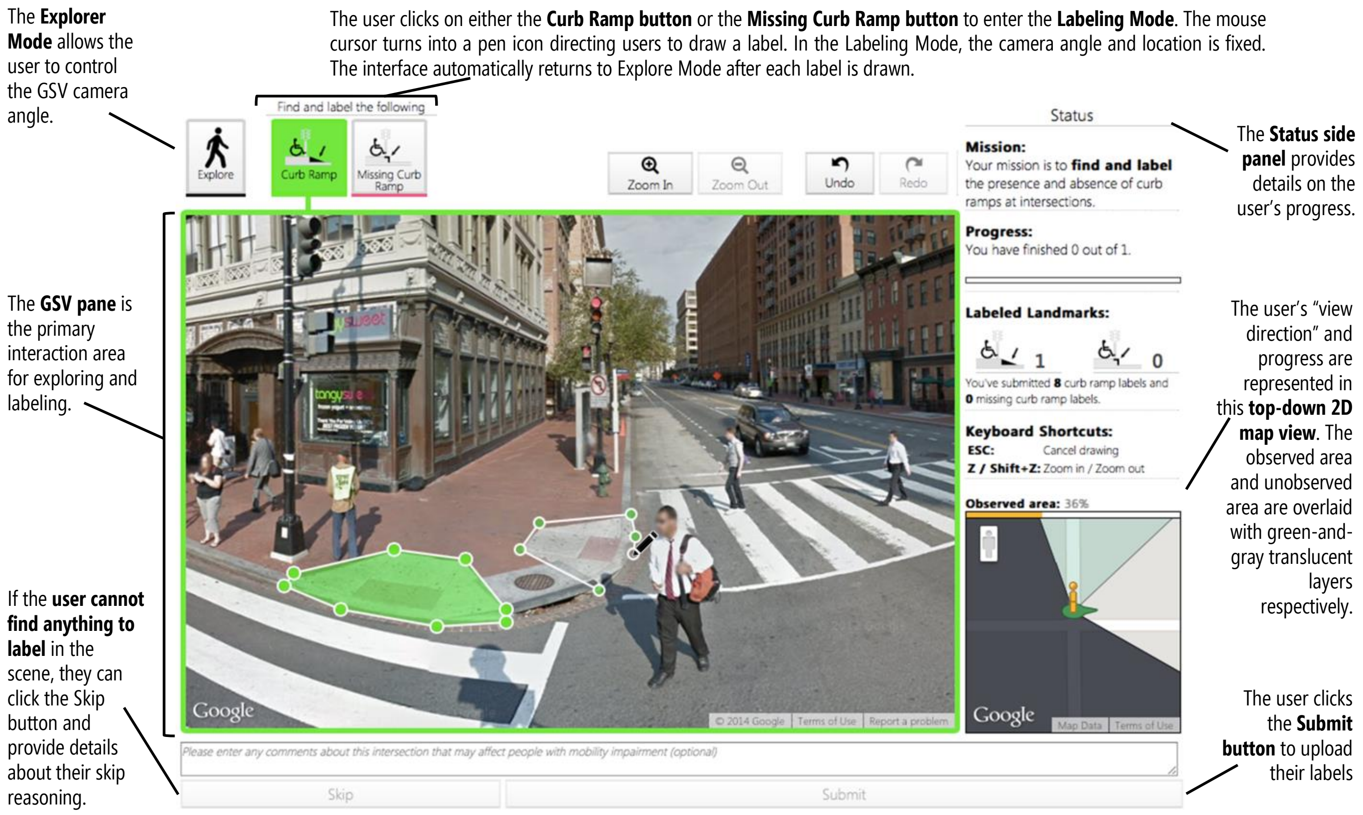


Figure 5: Example curb ramp and missing curb ramp labels from our turk studies. The green/pink outline points denote presence/absence.


The **Explorer Mode** allows the user to control the GSV camera angle.

The user clicks on either the **Curb Ramp button** or the **Missing Curb Ramp button** to enter the **Labeling Mode**. The mouse cursor turns into a pen icon directing users to draw a label. In the Labeling Mode, the camera angle and location is fixed. The interface automatically returns to Explore Mode after each label is drawn.

The **Status side panel** provides details on the user's progress.

The **GSV pane** is the primary interaction area for exploring and labeling.

The user's "view direction" and progress are represented in this **top-down 2D map view**. The observed area and unobserved area are overlaid with green-and-gray translucent layers respectively.

If the **user cannot find anything to label** in the scene, they can click the Skip button and provide details about their skip reasoning.

The user clicks the **Submit button** to upload their labels

Mission:
Your mission is to **find and label** the presence and absence of curb ramps at intersections.

Progress:
You have finished 0 out of 1.

Labeled Landmarks:
1 0
You've submitted **8** curb ramp labels and **0** missing curb ramp labels.

Keyboard Shortcuts:
ESC: Cancel drawing
Z / Shift+Z: Zoom in / Zoom out

Observed area: 36%

Please enter any comments about this intersection that may affect people with mobility impairment (optional)

Figure 6: The svLabel interface. Crowd workers use the *Explorer Mode* to interactively explore the intersection (via pan and zoom) and switch to the *Labeling Mode* to label curb ramps and missing curb ramps. Clicking the *Submit* button uploads the target labels. The turker is then transported to a new location unless the HIT is complete.

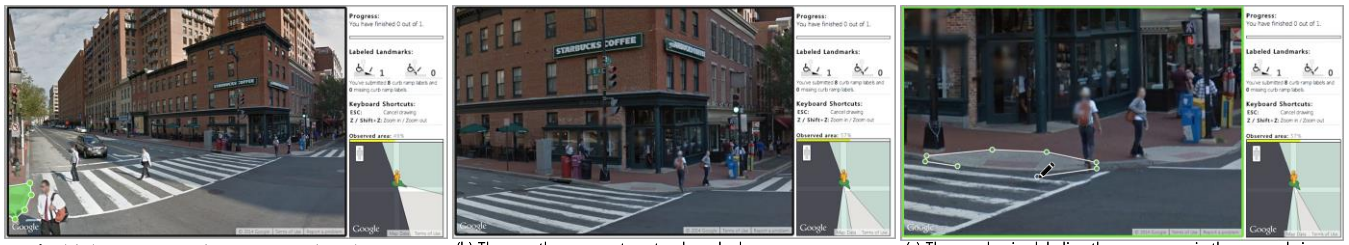

(a) After labeling one corner, the user pans to the right. (b) The user then zooms to get a closer look (c) The user begins labeling the new corner in the zoomed view

Figure 7: svLabel automatically tracks the camera angle and repositions any applied labels in their correct location as the view changes. When the turker pans the scene, the overlay on the map view is updated and the green "explored" area increases (bottom right of interface). Turkers can zoom in up to two levels to inspect distant corners. Labels can be applied at any zoom level and are scaled appropriately.

around the virtual 3D-space from one corner to the next within an intersection.

Using svLabel. When a turker accepts our HIT, they are immediately greeted by a three-stage interactive tutorial (see supplementary video included with paper). The stages progressively teach the turker about the interface (*e.g.*, the location of buttons and other widgets), user interactions (*e.g.*, how to label, zoom, and pan), and task concepts (*e.g.*, the definition of a curb ramp). If mistakes are made, our tutorial tool automatically provides corrective guidance. Turkers must successfully complete one tutorial stage before moving on to the next.

Once the tutorials are completed, we automatically position the turker in one of the audit area intersections and the labeling task begins in earnest. Similar to Bus Stop CSI [26], svLabel has two primary modes of interaction: *Explorer Mode* and *Labeling Mode* (Figure 6). When the user first drops into a scene, s/he defaults into Explorer Mode, which allows for exploration using Street View's native controls. Users are instructed to pan around to explore the 360 degree view of the intersection and visual feedback is provided to track their progress (bottom-right corner of Figure 6). Note: users' movement is restricted to the drop location.

When the user clicks on either the *Curb Ramp* or *Missing Curb Ramp* buttons, the interface switches automatically to Labeling Mode. Here, mouse interactions no longer control the camera view. Instead, the cursor changes to a pen, allowing the user to draw an outline around the visual target—a curb ramp or lack thereof (Figure 5). We chose to have users outline the area rather than simply clicking or drawing a bounding box because the detailed outlines provide a higher degree of granularity for developing and experimenting with our CV algorithms. Once an outline is drawn, the user continues to search the intersection. Our tool automatically tracks the camera angle and repositions any applied labels in their correct location as the intersection view changes. In this way, the labels appear to "stick" to their associated targets. Once the user has surveyed the entire intersection by panning 360 degrees, s/he can submit the task and move on to the next task in the HIT, until all tasks are complete.

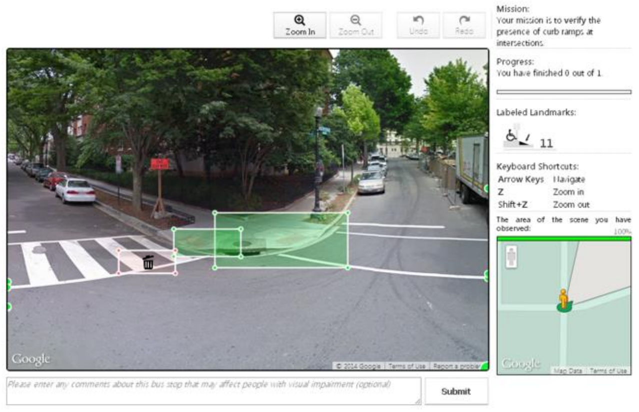
Ground Truth Seeding. A single HIT is comprised of either five or six intersections depending on whether it contains a ground truth scene (a scene is just an intersection). This "ground truth seeding" [40] approach is commonly used to dynamically examine, provide feedback about, and improve worker performance. In our case, if a user makes a mistake at a ground truth scene, after hitting the submit button, we provide visual feedback about the error and show the proper corrective action (see video). The user must correct all mistakes before submitting a ground truth task. If no mistakes are detected, the user is congratulated for their good performance. In our current system, there is a 50% chance that a HIT will contain one ground truth scene. The user is not able to tell whether they are working on a ground truth scene until after they submit their work.

svVerify: Human-Powered GSV Label Verification

In addition to providing "curb ramp" and "missing curb ramp" labels, we rely on crowd workers to examine and verify the correctness of previously entered labels. This verification step is common in crowdsourcing systems to increase result quality (*e.g.*, [24, 43]). svVerify (Figure 8) is similar to svLabel in appearance and general workflow but has a simplified interaction (clicking and panning only) and is for an easier task (clicking on incorrect labels).

While we designed both svLabel and svVerify to maximize worker efficiency and accuracy, our expectation was that the verification task would be significantly faster than initially providing manual labels [43]. For verification, users need not perform a time-consuming visual search looking for curb ramps to label but rather can quickly scan for incorrect labels (false positives) to delete. And, unlike labeling, which requires drawing polygonal outlines, the delete interaction is a single click over the offending label (similar to [46]). This enables users to rapidly eliminate *false positive* labels in a scene.

To maintain verification efficiency, however, we did not allow the user to spatially locate *false negatives*. This would essentially turn the verification task into a labeling task, by asking users to apply new "curb ramp" or "curb ramp missing" labels when they noticed a valid location that had not been labeled. Instead, svVerify gathers information on false negatives at a coarser-grained level by asking the user if the current scene was missing any labels after s/he clicks the submit button. Thus, svVerify can detect the presence of


Mission:
Your mission is to verify the presence of curb ramps at intersections.

Progress:
You have finished 0 out of 1.

Labeled Landmarks:
11

Keyboard Shortcuts:

Arrow Keys	Navigate
Z	Zoom in
Shift+Z	Zoom out

The area of the scene you have observed: 100%

Please enter any comments about this bus stop that may affect people with visual impairment (optional)

Figure 8: The svVerify interface is similar to svLabel but is designed for verifying rather than labeling. When the mouse hovers over a label, the cursor changes to a garbage can and a click removes the label. The user must pan 360 degrees before submitting the task.

false negatives in an intersection but not their specific location or quantity.

Similar to svLabel, svVerify requires turkers to complete an interactive tutorial before beginning a HIT, which includes instructions about the task, the interface itself, and successfully verifying one intersection. Because verifications are faster than providing labels, we included 10 scenes in each HIT (*vs.* the 5 or 6 in svLabel). In addition, we inserted one ground truth scene into *every* svVerify HIT rather than with 50% probability as was done with svLabel. Note that not all scenes are sent to svVerify for verification, as discussed in the svControl section below. We move now to describing the two more technical parts of Tohme: svDetect and svControl.

svDetect: Detecting Curb Ramps Automatically

While svLabel relies on manual labeling for finding curb ramps, svDetect attempts to do this automatically using CV. Because CV-based object detection is still an open problem—even for well-studied targets such as cars [18] and people [11]—our goal is to create a system that functions well enough to reduce the cost of curb ramp detection *vs.* a manual approach alone.

svDetect uses a three-stage detection process. First, we train a *Deformable Part Model (DPM)* [18], one of the most successful recent approaches in object detection (*e.g.*, [15]), as a first-pass curb ramp detector. Second, we post-process the resulting bounding boxes using non-maximum suppression [37] and 3D-point cloud data to eliminate detector redundancies and false positives. Finally, the remaining bounding boxes are classified using a Support Vector Machine (SVM) [8], which uses features not leveraged by the DPM, further eliminating false positives.

svDetect was designed and tested iteratively. We attempted multiple algorithmic approaches and used preliminary experiments to guide and refine our approach. For example, we previously used a linear SVM with a Histograms of Oriented Gradients (HOG) feature descriptor [27] but found

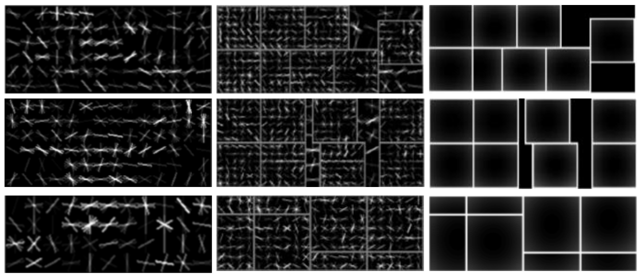


Figure 9: The trained curb ramp DPM model. Each row represents an automatically learned viewpoint variation. The root and parts filter visualize learned weights for the gradient features. The displacement costs for parts are shown in (c).

that the DPM was able to recognize curb ramps with larger variations. In addition, we found that though the raw GSV image size is 13,312 x 6,656 pixels, there were no detection performance benefits beyond 4,096 x 2,048px (the resolution used throughout this paper). Because it helps explain our design rationale for Tohme, we include our evaluation experiments for svDetect in this section rather than later in the paper.

First Stage: The Curb Ramp Deformable Part Model (DPM)

DPMs are comprised of two parts: a coarse-grained model, called a root filter, and a higher resolution parts model, called a parts filter. DPMs are commonly applied to human detection in images, which provides a useful example. For human detection, the root filter captures the whole human body while part filters are for individual body parts such as the head, hand, and legs (see [17]). The individual parts are learned automatically by the DPM—that is, they are not explicitly defined *a priori*. In addition, how these parts can be positioned around the body (the root filter) is also learned and modeled via displacement costs. This allows a DPM to recognize different configurations of the human body (*e.g.*, sitting *vs.* standing).

In our case, the root filter describes the general appearance of a curb ramp while part filters account for individual components (*e.g.*, edges of the ramp and transitions to the road). DPM creates multiple *components* for a single model (Figure 9) based on bounding box aspect ratios. We suspect that each component implicitly captures different viewpoints of a curb ramp. For our DPM, we used code provided by [20].

Second Stage: Post-Processing DPM Output

In the second stage, we post-process the DPM output in two ways. First, similar to [37], we use non-maximum suppression (NMS) to eliminate redundant bounding boxes. NMS is common in CV and works by greedily selecting bounding boxes with high confidence values and removing overlapping boxes with lower scores. Overlap is defined as the ratio of intersection of the two bounding boxes over the union of those boxes. Based on the criteria established by the PASCAL Visual Object Classes challenge [16], we set our NMS overlap threshold to 50%.

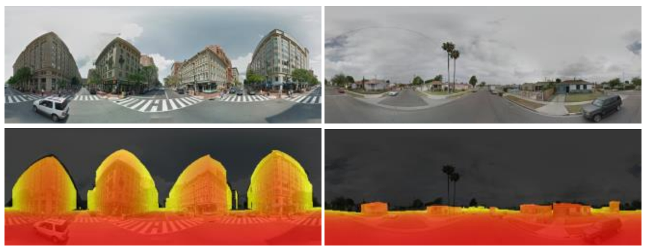


Figure 10: Using code from [39], we download GSV's 3D-point cloud data and use this to create a ground plane mask to post-process DPM output. The 3D depth data is coarse: 512 x 256px.

Our second post-processing step uses the 3D-point cloud data to eliminate curb ramp detections that occur above the ground plane (*e.g.*, bounding boxes in the sky are removed). To do so, the 512 x 256px depth image is resized to the GSV image size (4096 x 2048px) using bilinear interpolation. For each pixel, we calculate a normal vector and generate a mask for those pixels with a strong vertical component. These pixels correspond to the ground plane. Bounding boxes outside of this pixel mask are eliminated (Figure 10 and 11).

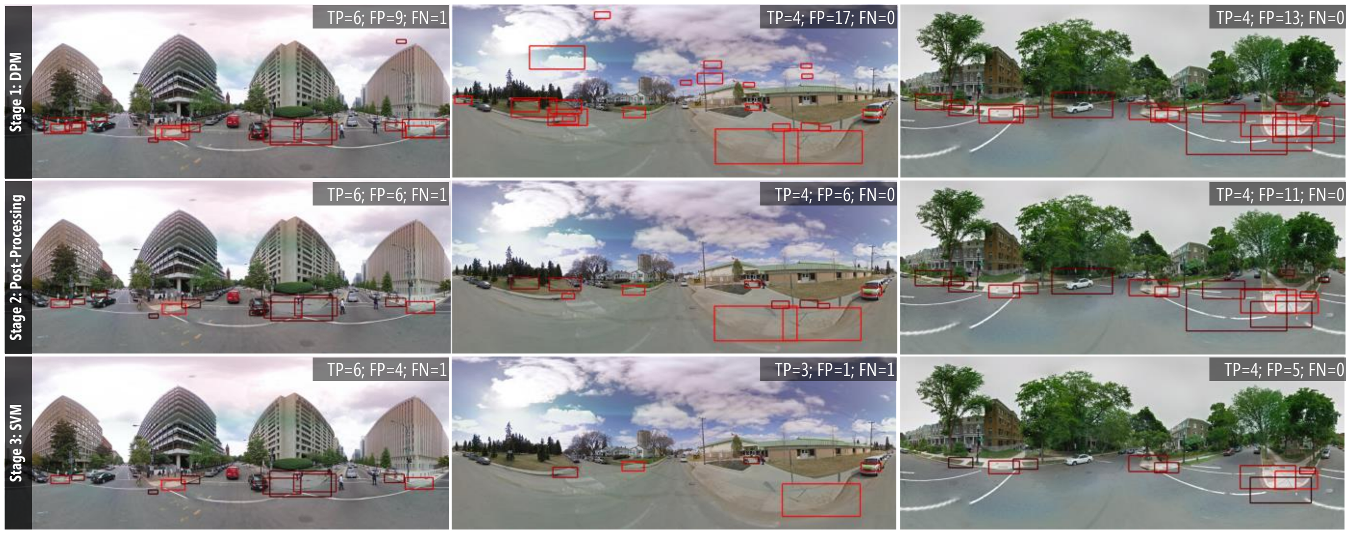
Third Stage: SVM-Based Classification

Finally, in the third stage, the remaining bounding boxes are fed into an additional classifier: an SVM. Because the DPM relies solely on gradient features in an image, it does not utilize other important discriminable information such as color or position of the bounding box. Given that street intersections have highly constrained geometrical configurations, curb ramps tend to occur in similar locations—so detection position is important. Thus, for each bounding box, we create a feature vector that includes: RGB color histograms, the top-left and bottom-right corner coordinates of the bounding box in the GSV image along with its width and height, and the detection confidence score from the DPM detector. We use the SVM as a binary classifier to keep or discard detection results from the second stage.

svDetect Training and Results

Two of the three svDetect stages require training: the DPM in Stage 1 and the SVM in Stage 3. For training and testing, we used two-fold cross validation across the 1,086 GSV scenes and 2,877 ground truth curb ramp labels. The GSV scenes were randomly split in half (543 scenes per fold) with one fold initially assigned for training and the other for testing. This process was then repeated with the training and testing folds switched.

To train the DPM (Stage 1), we transform the polygonal ground truth labels into rectangular bounding boxes, which are used as positive training examples. DPM uses a sliding window approach, so the rest of the GSV scene is treated as negative examples (*i.e.*, comprised of negative windows). For each image in the training set, the DPM produces a set of bounding boxes with associated confidence scores. The number of bounding boxes produced per scene is contingent on a minimum score threshold. This threshold is often learned empirically (*e.g.*, [1]). A high threshold would



(a) Downtown DC (b) Residential Saskatchewan (c) Residential DC

Figure 11: Example results from svDetect's three-stage curb ramp detection framework. Bounding boxes are colored by confidence score (lighter is higher confidence). As this figure illustrates, setting the detection threshold to -0.99 results in a relatively low false negative rate at a cost of a high false positive rate (false negatives are more expensive to correct). Many false positives are eliminated in Stages 2 and 3. The effect of Stage 2's ground plane mask is evident in (b). Acronyms: TP=true positive; FP=false positive; FN=false negative.

produce a small number of bounding boxes, which would likely result in high precision and low recall; a low threshold would likely lead to low precision and high recall.

To train the SVM (Stage 3), we use the post-processed DPM bounding boxes from Stage 2. The bounding boxes are partitioned into positive and negative samples by calculating area overlap with the ground truth labels. Though there is no universal standard for evaluating "good area overlap" in object detection research, we use 20% overlap (from [19]). Prior work suggests that even 10-15% overlap agreement at the pixel level would be sufficient to confidently localize accessibility problems in images [24]. Thus, positive samples are boxes that overlap with ground truth by more than 20%; negative samples are all other boxes. We extract the aforementioned training features from both the positive and negative bounding boxes. Note that SVM parameters (*e.g.*, coefficient for slack variables) are automatically selected by grid search during training.

Results. To analyze svDetect's overall performance and to determine an appropriate confidence score cutoff for svDetect, we stepped through various DPM detection thresholds (from -3-to-3 with a 0.01 step) and measured the results. For each threshold, we calculated true positive, false positive, and false negative detections for each scene. True positives were assessed as bounding boxes that had 20% overlap with ground truth labels and that had a detection score higher than the currently set threshold. The results are graphed on a precision-recall curve in Figure 12. To balance the number of true positive detections and false positives in our system, we selected a DPM detection threshold of -0.99. At this threshold, svDetect generates an average of 7.0 bounding boxes per intersection (*SD*=3.7); see Figure 11 for examples. Note: svDetect failed to generate a bounding box for 15 of the 1,086 intersections. These are still included in our performance comparison.

In the ideal, our three-stage detection framework would have both high precision and high recall. As can be observed in Figure 12, this is obviously not the case as ~20% of the curb ramps are never detected (*i.e.*, the recall metric never breaches 80%). With that said, automatically finding curb ramps using CV is a hard problem due to viewpoint variation, illumination, and within/between class variation. This is why Tohme combines automation with manual labor using svControl.

svControl: Scheduling Work via Performance Prediction

svControl is a machine-learning module for predicting CV performance and assigning work to either a manual labor

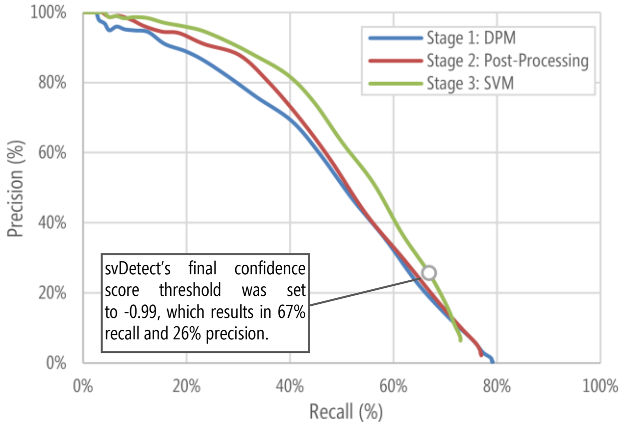


Figure 12: The precision-recall curve of the three-stage curb ramp detection process constructed by stepping through various DPM detection thresholds (from -3-to-3 with a 0.01 step). For the final svDetect module, we selected a DPM detection threshold of -0.99, which balances true positive detections with false positives.

pipeline (svLabel) or an automated pipeline with human verification (svDetect + svVerify)—see Figure 4. We designed svControl based on three principles: first, that human-based verifications are fast and relatively low-cost compared to human-based labeling; second, CV is fast and inexpensive but error prone both in producing high false positives and false negatives; third, false negatives are more expensive to correct than false positives.

From these principles, we derived two overarching design questions: first, given the high cost of human labeling and relative low-cost of human verification, could we optimize CV performance with a bias towards a low false negative rate (even if it meant an increase in false positives)? Second, given that false negatives cannot be eliminated completely from svDetect, can we predict their occurrence based on features of an intersection and use this to divert work to svLabel instead for human labeling?

Towards the first question, biasing CV performance towards a certain rate of false negatives is trivial. It is simply a matter of selecting the appropriate threshold on the precision/recall curve (recall that the threshold that we selected was -0.99). The second question is more complex. We iterated over a number of prediction techniques and intersection features before settling on a linear SVM and Lasso regression model [44] with the following three types of input features:

- **svDetect results (16 features):** For each GSV image, we include the raw number of bounding boxes output from svDetect, the average, median, standard deviation, and range of confidence scores of all bounding boxes in the image, and descriptive statistics for their XY-coordinates. Importantly, we did not use the *correctness* of the bounding box as a feature since this would be unknown during testing.
- **Intersection complexity (2 features):** We calculate intersection complexity via two measures: *cardinality* (*i.e.,* how many streets are connected to the target intersection) and an *indirect measure* of complexity, for which we count the number of street pixels in a stylized top-down Google Map. We found that high pixel counts correlate to high intersection complexity (Figure 13).
- **3D-point cloud data (5 features)**: svDetect struggles to detect curb ramps that are distant in a scene—*e.g.,* because the intersection is large or because the GSV car is in a sub-optimal position to photograph the intersection. Thus, we include descriptive statistics of depth information of each scene (*e.g.*, average, median, variance).

We combine the above features into a single 23-dimensional feature vector for training and classification.

svControl Training and Test Results

We train and test svControl with two-fold cross validation using the same train and test data as used for svDetect. Given that the goal of svControl is to predict svDetect performance, namely the occurrence of false negatives, we define a svDetect *failure* as a GSV scene with at least one false negative curb ramp detection. The SVM model is trained to make a binary failure prediction with the

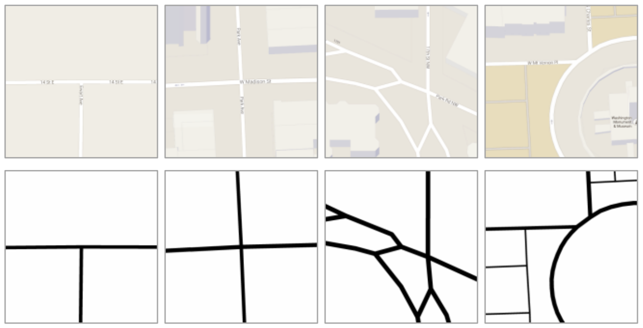


Figure 13: We use top-down stylized Google Maps (bottom row) to infer intersection complexity by counting black pixels (streets) in each scene. A higher count correlates to higher complexity.

aforementioned features. Similarly, the Lasso regression model is trained to predict the *raw number* of false negatives of svDetect (regression value > 0.5 is failure).

To help better understand the important features in our models, we present the top three correlation coefficients for both. For the SVM, the top coefficients were the label's x-coordinate variance (0.91), the mean confidence score of automatically detected labels (0.69), and the minimum scene depth (0.67). For the Lasso model, the top three were mean scene depth (0.69), median scene depth (-0.28), and, similar to the SVM, the mean confidence score of the automatically detected labels (0.21). If either the SVM or the Lasso model predicts failure on a particular GSV scene, svControl routes that scene to svLabel instead of svVerify.

***svControl Results*.** We assessed svControl's prediction performance across the 1,086 scenes. While not perfect, our results show that svControl is capable of identifying svDetect failures with high probability—we correctly predicted 397 of the 439 svDetect failures (86.3%); however, this high recall comes at a cost of precision: 404 of the total 801 scenes (50.4%) marked as failures were false positives. Given that we designed svControl to be conservative (*i.e.,* pass more work to svLabel if in doubt about svDetect), this accuracy balance is reasonable. Below, we examine whether this is sufficient to provide performance benefits for Tohme.

STUDY 2: EVALUATING TOHME

To examine the effectiveness of Tohme for finding curb ramps in GSV images and to compare its performance to a baseline approach, we performed an online study with MTurk in spring 2014. Our goal here is threefold: first, and most importantly, to investigate whether Tohme provides performance benefits over manual labeling alone (baseline); second, to understand the effectiveness of each of Tohme's sub-systems (svLabel, svVerify, svDetect, and svControl); and third, to uncover directions for future work in preparation for a public deployment.

Tohme Study Method

Similar to Hara *et al.* [24], we collected more data than necessary in practice so that we could simulate performance with different workflow configurations *post hoc*. To allow

	Turkers	GSV Scenes	HITs	Tasks	Avg. Turkers / Intersection	Label Stats	Avg. Task Time
SVLABEL	242	1,046	1,270	6,350	6.1 (0.6)	20,789 labels (17,327CRs, 3,462MCRs)	94.1s (144.4s)
SVVERIFY	161	1,046	582	5,820	5.6 (0.6)	42,226 verified labels (28,801RLs, 13,425KLs)	43.2 (48.7s)

Table 3: An overview of the MTurk svLabel and svVerify HITs. While Tohme's svControl system would, in practice, split work between the svLabel and svDetect+svVerify pipelines, we fed every GSV scene to both to perform our analyses. Acronyms above include CRs=Curb Ramps; MCRs=Missing Curb Ramps; RLs=Removed Labels; KLs=Kept Labels. svVerify was 2.2x faster than svLabel.

us to compare Tohme *vs.* feeding all scenes to either workflow on their own (svLabel and svDetect+svVerify), we ran *all* GSV scenes through both. To avoid interaction effects, turkers hired for one workflow (labeling) could not work on the other (verifying) and vice versa.

Second, to more rigorously assess Tohme and to reduce the influence of any one turker on our results, we hired at least three turkers per scene for each workflow and used this data to perform Monte Carlo simulations. More specifically, for both workflows, we randomly sampled one turker from each scene, calculated performance statistics (*e.g.*, precision), and repeated this process 1,000 times. Admittedly, this is a more complex evaluation than simply hiring one turker per scene and computing the results; however, the Monte Carlo simulation allows us to derive a more robust indicator of Tohme's expected future performance.

Of the 1,086 GSV scenes (street intersections) in our dataset, we reserved 40 for ground truth seeding, which were randomly selected from the eight geographic areas (5 scenes from each). We calculated HIT payment rates based on MTurk pilot studies: $0.80 for svLabel HITs (five intersections; $0.16 per intersection) and $0.80 for svVerify (ten intersections; $0.08 per intersection). As noted in our system description, turkers had to successfully complete interactive tutorials before beginning the tasks.

Analysis Metrics

To assess Tohme, we used the following measures:

- **Label overlap compared to ground truth:** as described in the svDetect section, we use 20% overlap as our correctness threshold (from [24]).
- **We calculate standard object detection performance metrics** including precision, recall, and F-measure based on this 20% area overlap—the same overlap used by svDetect.
- **Human time cost**: cost is calculated by measuring completion times for each intersection in svLabel and svVerify.

Tohme Study Results

We first present high-level descriptive statistics of the MTurk HITs before focusing on the comparison between Tohme *vs.* our baseline approach (pure manual labeling with svLabel). We provide additional analyses that help explain the underlying trends in our results.

Descriptive Statistics of MTurk Work

To gather data for our analyses, we hired 242 distinct turkers for the svLabel pipeline and 161 turkers for the svVerify pipeline (Table 3). As noted previously, all 1,046 GSV scenes were fed through both workflows. For svLabel, turkers completed 1,270 HITs (6,350 labeling tasks) providing 17,327 curb ramp labels and 3,462 missing curb ramp labels. For svVerify, turkers completed 582 HITs (5,820 verification tasks) and verified a total of 42,226 curb ramp labels. On average, turkers eliminated 4.9 labels per intersection (*SD*=2.9). We hired an average of 6.1 (*SD*=0.6) turkers per intersection for svLabel and 5.6 (*SD*=0.6) for svVerify.

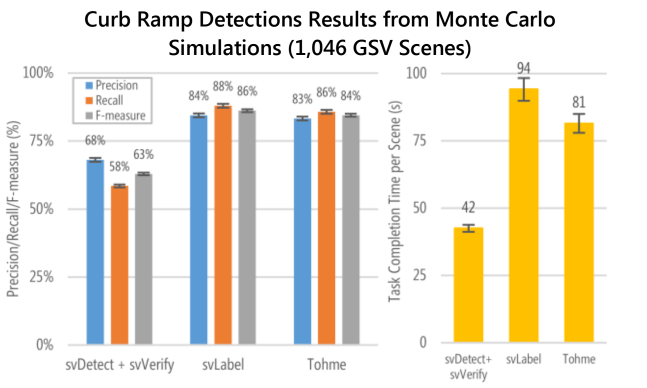


Figure 14: Tohme achieves comparable results to a manual labeling approach alone but with a 13% reduction in time cost. Error bars are standard deviation.

Evaluating Tohme's Performance

To evaluate Tohme's overall performance, we first examined how well each pipeline would perform on its own across the entire dataset (1,046 scenes). This provides two baselines for comparison: (i) the *svDetect* + *svVerify* results show how well Tohme would perform if the svControl module passed *all* work to this pipeline and, similarly, (ii) the *svLabel* results show what would happen if we only relied on manual labor for finding and labeling curb ramps.

We found that Tohme achieved similar but slightly lower curb ramp detection results compared to the manual approach alone (F-measure: 84% *vs.* 86%) but with a much lower time cost (13% reduction); see Figure 14. As

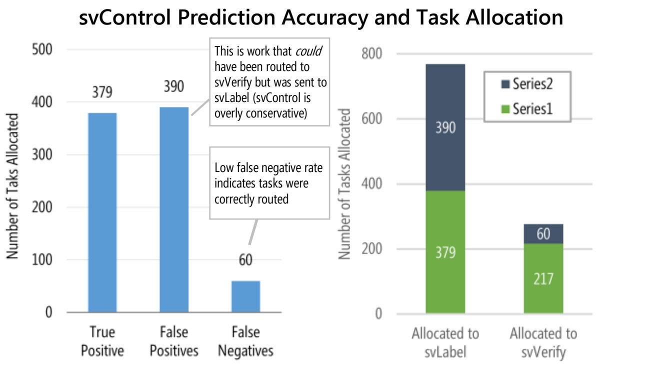


Figure 15: svControl allocated 769 scenes to svLabel and 277 scenes to svVerify. 379 out of 439 scenes (86.3%) where svDetect failed were allocated "correctly" to svLabel. Recall that svControl is conservative in routing work to svVerify because false negative labels are expensive to correct; thus, the 86.3% comes at a high false positive cost (390).

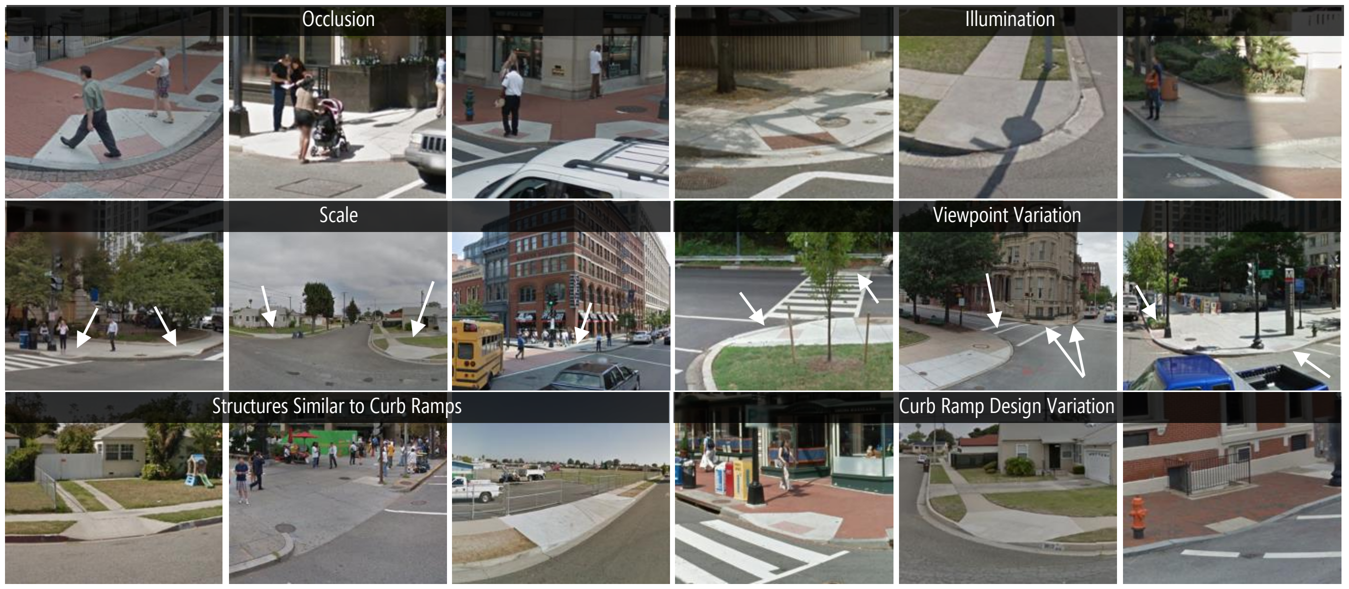


Figure 16: Finding curb ramps in GSV imagery can be difficult. Common problems include occlusion, illumination, scale differences because of distance, viewpoint variation (side, front, back), between class similarity, and within class variation. For between class similarity, many structures exist in the physical world that appear similar to curb ramps but are not. For within class variation, there are a wide variety of curb ramp designs that vary in appearance. White arrows are used in some images to draw attention to curb ramps. Some images contain multiple problems.

expected, while the svDetect + svVerify pipeline is relatively inexpensive, it performed the worst (F-measure: 63%). These findings show that the svControl module routed work appropriately to maintain high accuracy but at a reduced cost. Tohme reduces the average per-scene processing time by 12 seconds compared to svLabel alone. The overall task completion times were 12.3, 27.3, and 23.7 hours for svDetect + svVerify, svLabel, and Tohme respectively.

The above results were calculated using the aforementioned Monte Carlo method. If we, instead, use only the *first* turker to arrive and complete the task, our results are largely the same. The F-measures are 63%, 86%, and 85% respectively for svDetect + svVerify, svLabel, and Tohme with a 10% drop in cost for Tohme (rather than 13%). This includes 65 distinct turkers for svDetect + svVerify, 97 for svLabel, and 149 for Tohme.

Task Allocation by svControl

As the workflow scheduler, the svControl module is a critical component of Tohme. Because the svVerify interface does not allow for labeling (*e.g.,* correcting false negatives), the svControl system is conservative—it routes most of the work to svLabel otherwise many curb ramps would possibly remain undetected. Of the 1,046 scenes, svControl predicted svDetect to fail on 769 scenes (these results are the same as presented in the svControl section but with the 40 ground truth scenes removed). Thus, 73.5% of all scenes were routed to svLabel for manual work and the rest (277) were fed to svVerify for human verification (Figure 15). Again, svControl's true positive rate is high: 86%. However, if svControl worked as a perfect classifier, 439 scenes would have been forwarded to svLabel and 607 to svVerify. In this idealized case, Tohme's cost drops to 27.7% compared to a manual labeling approach with the same F-measure as before (84%). Thus, assuming limited improvements in CV-based curb ramp detections in the near future, a key area for future work will be improving the workflow control system.

Where Humans and Computers Struggle

The key to improving both CV and human labeling performance is to understand *where* and *why* each sub-system makes mistakes. To assess the detection accuracy of human labelers, we calculated the average F-measure score per scene based on the average number of true positives (TP), false positives (FP), and false negatives (FN). For example, if the average for a scene was (TP, FP, FN) = (1, 1, 2), then (Precision, Recall, F-measure) = (0.5, 0.3, 0.4). For CV, we simply used the F-measure score for each scene based on our svDetect results. We sorted the two F-measure lists and visually inspected the best and worst performing scenes for each. For the top and bottom 10, the average F-measure scores were 99% and 0% for CV and 100% and 25% for human labeling respectively. Common problems are summarized in Figure 16.

Crowd workers struggled with labeling distant curb ramps (scale) or due to placement and angle (viewpoint variation). To mitigate this, future labeling interfaces could allow the worker to "walk" around the intersection to select better viewpoints (similar to [26]); however, this will increase user-interaction complexity and labeling time. Perhaps as should be expected, crowd workers were much more adept at dealing with occlusion than CV—even if a majority of a curb ramp was occluded, a worker could infer its location and shape (*e.g.,* middle occlusion picture). CV struggled for all the reasons noted in Figure 16. Given the tremendous variation in curb ramp design and capture angles, a larger training set may have improved our results. Moreover, because multiple views of a single intersection are available

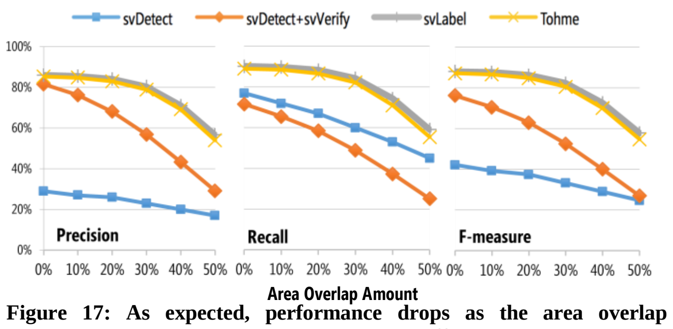


Figure 17: As expected, performance drops as the area overlap threshold increases; however, the relative difference between Tohme and baseline (svLabel) remains consistent.

in GSV via neighboring panoramas, these additional perspectives could be combined to potentially improve scene structure understanding and mitigate issues with occlusion, illumination, scale, and viewpoint variation. The semantic issues—*e.g.*, confusing structures similar to curb ramps—are obviously much more difficult for CV than humans. We describe other areas for improvement in the Discussion.

Effect of Area Overlap Threshold on Performance

As noted previously, there is no universal standard for selecting an area overlap threshold in CV; this decision is often domain dependent. To investigate the effect of changing the overlap threshold on performance, we measured precision, recall, and F-measure at different values from 0-50% at a step size of 10% (Figure 17). For *overlap=0%*, at least 1px of a detected bounding must overlap with a ground truth label to be considered correct.

A few observations: first, as expected, performance decreases as the overlap threshold increases; however, the relative performance difference between Tohme and baseline (svLabel) stays roughly the same. For example, at 0% overlap, the (Precision, Recall, F-measure) of Tohme is (85%, 89%, 87%) and (86%, 90%, 88%) for svLabel and at 50% overlap, (54%, 55%, 55%) *vs.* (57%, 59%, 58%). Thus, Tohme's relative performance is consistent regardless of overlap threshold (*i.e.*, slightly poorer performance but cheaper). Second, there appears to be a more substantial performance drop starting at ~30%, which suggests that obtaining curb ramp label agreement at the pixel level between human labelers and ground truth after this point is difficult. Finally, though svDetect + svVerify has much greater precision than svDetect alone, this increase comes at a cost of recall—a gap which widens as the overlap threshold becomes more aggressive. So, though human verifiers help increase precision, they are imperfect and sometimes delete true positive labels.

DISCUSSION

Our research advances recent work using GSV and crowdsourcing to remotely collect data on accessibility features of the physical world (*e.g.,* [22–24, 26]) by integrating CV and a machine learning-based workflow scheduler. We showed that a trained CV-based curb ramp detector (svDetect) found 63% of curb ramps in GSV scenes and fast, human-based verifications further improved the overall results. We also demonstrated that a novel machine-learning based workflow controller, svControl, could predict CV performance and route work accordingly. Below, we discuss limitations and opportunities for future work.

Improving Human Interfaces

How much context is necessary for verification? We were surprised that verification tasks were only 2.2x faster than labeling tasks. Though we attempted to design both interfaces for rapid user interaction, there is some basic overhead incurred by panning and searching in the 360-degree GSV view. In an attempt to eliminate this overhead, we have designed a completely new type of verification interface, *quickVerify*, that simply presents detected bounding boxes in a grid view (Figure 18). Similar to the facial recognition verifier in Google Picasa, these boxes can be rapidly confirmed or rejected with a single-click and a new bounding box appears in its place. In a preliminary experiment using 160 GSV scenes and 59 distinct turkers, however, we found that accuracy with quickVerify dropped significantly. Unlike faces, we believe that curb ramps require some level of surrounding context to accurately perceive their existence. More work is needed to determine the appropriate amount of surrounding view context to balance speed and accuracy.

Improving human labeling. Human labeling time could be reduced if point-and-click interactions were used for labeling targets rather than outlining; however, as demonstrated in Figure 16, curb ramps vary dramatically in size, scale, and shape. Clicking alone would be insufficient for CV training. Moreover, labeling will *always* be more costly than verification because it is a more difficult task (*i.e.*, finding elements in an image requires visual search and a higher mental load). With that said, we currently discard *all* svDetect bounding boxes—even those with a high confidence score—when a scene is routed to svLabel. Future work should explore how to, instead, best utilize this CV data to improve worker performance (*e.g.*, by showing

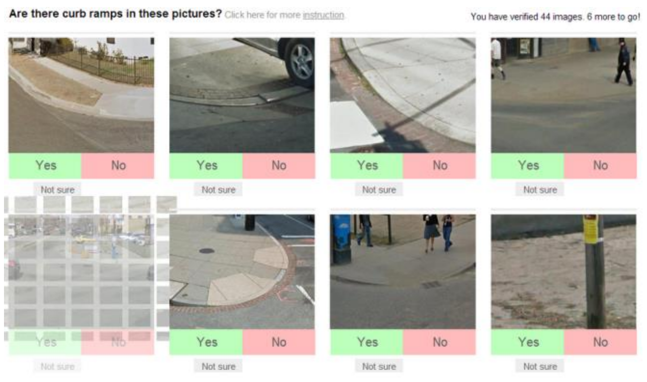

Are there curb ramps in these pictures? Click here for more instruction
You have verified 44 images. 6 more to go!
Yes
No
Not sure

Figure 18: In the *quickVerify* interface, workers could randomly verify CV curb ramp detection patches. After providing an answer for a given detection, the patch would "explode" (bottom left) and a new one would load in its place. Though fast, verification accuracies went down in an experiment of 160 GSV scenes and 59 turkers.

detected bounding boxes with high scores to the user or as a way to help *verify* human labels). Finally, similar to quickVerify, future work could explore GSV panorama labeling that is not projected onto a 3D-sphere but is instead flattened into a 2D zoomable interface (*e.g.,* [33]) or specially rendered to increase focus on intersection corners.

Improving Automated Approaches

As the first work in automatically detecting curb ramps using CV, there are no prior systems with which to directly compare our performance. Having said that, there is much room for improvement and advances in CV will only increase the overall efficacy of our system.

Improving CV-based curb ramp detection. We are currently exploring three areas of future work: (i) *Context integration.* While we use some context information in Tohme (*e.g.,* 3D-depth data, intersection complexity inference), we are exploring methods to include broader contextual cues about buildings, traffic signal poles, crosswalks, and pedestrians as well as the precise location of corners from top-down map imagery. (ii) *3D-data integration.* Due to low-resolution and noise, we currently use 3D-point cloud data as a ground plane mask rather than as a feature to our CV algorithms. We plan to explore approaches that combine the 3D and 2D imagery to increase scene structure understanding (*e.g.,* [28]). If higher resolution depth data becomes available, this may be useful to directly detect the presence of a curb or corner, which would likely improve our results. (iii) *Training*. Our CV algorithms are currently trained using GSV scenes from all eight city regions in our dataset. Given the variation in curb ramp appearance across geographic areas, we expect that performance could be improved if we trained and tested per city. However, in preliminary experiments, we found no difference in performance. We suspect that this is due to the decreased training set size. In the future, we would like to perform training experiments to study the effects of per-city training and to identify minimal training set size. Relatedly, we plan to explore active learning approaches where crowd labels train the system over time.

Improving the workflow controller. While our current workflow controller focuses on predicting CV performance, future systems should explore modeling and predicting human worker performance and adapting work assignments accordingly. For example, struggling workers could be fed scenes that are predicted to be easy, or hard scenes can be assigned to more than one worker to take majority vote [10, 31]. Similar to CV detection, per-city training and active learning should also be explored.

Who pays? The question of who will pay for data collection (or if payment is even necessary) in the future is an important, unresolved one. Our immediate plans are to build an open website where anyone can contribute voluntarily. From conversations with motor impaired (MI) persons and the accessibility community as a whole (*e.g.,* non-profit organizations, families of those with MI), we believe there is a strong demand for this system. For example, with a public version of Tohme, a concerned, motivated father could easily label over 100 intersections in his neighborhood in a few hours. A website akin to walkscore.com could then visualize the accessibility of that neighborhood using heatmaps and also calculate accessible pedestrian routes.

Limitations

There are two primary limitations to our work. First, there is a workload imbalance between svLabel and svDetect. svLabel gathers explicit data on both curb ramps *and* missing curb ramps while svDetect only detects the former. It is likely that if the svLabel task involved only labeling curb ramps, the labeling task completion time would go down, which would affect our primary results. And, while the *lack* of a detected curb ramp could be equated to a missing curb ramp label for svDetect, we have not yet performed this analysis. Clearly, more explorations are needed here but we believe our initial examinations are sufficient to show the potential of Tohme.

Second, there is no assessment of how our curb ramp detection results compare to traditional auditing approaches (*e.g.,* performed by city governments). Anecdotally, we have found many errors in the DC government curb ramp dataset [12]; however, more research is necessary to uncover whether our approach is faster, cheaper, and/or more accurate. Ultimately, Tohme must produce sufficiently good data to enable new types of accessibility-aware GIS applications (*e.g.,* pedestrian directions routed through an accessible sidewalk path).

CONCLUSION

This paper contributes the design and evaluation of a new tool, Tohme, for semi-automatically detecting curb ramps in GSV images using crowdsourcing, computer vision, and machine learning. To our knowledge, we are the first work to design and investigate CV algorithms for curb ramp detection, an important sidewalk accessibility attribute. We are also the first to combine crowdsourcing with automated methods for collecting accessibility information about the physical world in GSV scenes. Tohme's custom workflow controller predicts CV performance and routes work accordingly to balance accuracy and human labor. Through an MTurk study of 1,086 intersections across four North American cities, we showed that Tohme could provide comparable curb ramp detection accuracy at a 13% reduction in cost. As computer vision and machine learning algorithms continue to improve, Tohme should only become more efficient.

While this paper focuses specifically on curb ramps, we believe a similar approach could be applied to analyze the accessibility of external building facades (*e.g.,* the presence of stairways), the safety of intersections (*e.g.,* the presence of painted cross walks), or even the accessibility of store aisles as mapping companies increasingly focus on the indoors (*e.g.,* [21]).

REFERENCES

[1] 311 Online http://311.dc.gov/: .

[2] 3rd Circuit, C. of A. 1993. *Kinney v. Yerusalim, 1993 No. 93-1168*.

[3] Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L. and Weaver, J. 2010. Google Street View: Capturing the World at Street Level. *Computer*. 43, 6 (Jun. 2010), 32–38.

[4] Badland, H.M., Opit, S., Witten, K., Kearns, R.A. and Mavoa, S. 2010. Can virtual streetscape audits reliably replace physical streetscape audits? *Journal of urban health : bulletin of the New York Academy of Medicine*. 87, 6 (Dec. 2010), 1007–16.

[5] Bigham, J.P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R.C., Miller, R., Tatarowicz, A., White, B., White, S. and Yeh, T. 2010. VizWiz: nearly real-time answers to visual questions. *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (New York, NY, USA, 2010), 333–342.

[6] Bigham, J.P., Ladner, R.E. and Borodin, Y. 2011. The design of human-powered access technology. *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '11)* (New York, NY, USA, 2011), 3–10.

[7] Branson, S., Wah, C., Babenko, B., Schroff, F., Welinder, P., Perona, P. and Belongie, S. 2010. Visual Recognition with Humans in the Loop. *European Conference on Computer Vision (ECCV)* (Heraklion, Crete, 2010).

[8] Burges, C.C. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*. 2, 2 (1998), 121–167.

[9] Clarke, P., Ailshire, J., Melendez, R., Bader, M. and Morenoff, J. 2010. Using Google Earth to conduct a neighborhood audit: reliability of a virtual audit instrument. *Health & place*. 16, 6 (Nov. 2010), 1224–9.

[10] Dai, P., Mausam and Weld, D.S. 2011. Artificial Intelligence for Artificial Artificial Intelligence. *AAAI* (2011).

[11] Dalal, N. and Triggs, B. 2005. Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.* (2005), 886–893 vol. 1.

[12] Data Catalog http://data.dc.gov/: .

[13] Dean, T., Ruzon, M.A., Segal, M., Shlens, J., Vijayanarasimhan, S. and Yagnik, J. 2013. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. *2013 {IEEE} Conference on Computer Vision and Pattern Recognition ({CVPR})* (Jun. 2013), 1814–1821.

[14] Deng, J., Russakovsky, O., Krause, J., Bernstein, M., Berg, A. and Fei-Fei, L. 2014. Scalable Multi-label Annotation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)* (2014), TBD.

[15] Everingham, M., Van~Gool, L., Williams, C.K.I., Winn, J. and Zisserman, A. The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2012 {(VOC2012)} {R}esults.

[16] Everingham, M., Van~Gool, L., Williams, C.K.I., Winn, J. and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*. 88, 2 (Jun. 2010), 303–338.

[17] Felzenszwalb, P., McAllester, D. and Ramaman, D. 2008. A Discriminatively Trained, Multiscale, Deformable Part Model. *CVPR* (2008).

[18] Felzenszwalb, P.F., Girshick, R.B., McAllester, D. and Ramanan, D. 2010. Object Detection with Discriminatively Trained Part-Based Models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 32, 9 (Sep. 2010), 1627–1645.

[19] Ferrari, V., Fevrier, L., Jurie, F. and Schmid, C. 2008. Groups of Adjacent Contour Segments for Object Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 30, 1 (Jan. 2008), 36–51.

[20] Girshick, R.B., Felzenszwalb, P.F. and McAllester, D. Discriminatively Trained Deformable Part Models, Release 5.

[21] Google Maps Business View: *http://www.google.com/maps/about/partners/businessview/*. Accessed: 2014-04-15.

[22] Guy, R. and Truong, K. 2012. CrossingGuard: exploring information content in navigation aids for visually impaired pedestrians. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)* (New York, NY, USA, 2012), 405–414.

[23] Hara, K., Le, V. and Froehlich, J. 2012. A Feasibility Study of Crowdsourcing and Google Street View to Determine Sidewalk Accessibility. *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '12), Poster Session* (New York, NY, USA, 2012), 273–274.

[24] Hara, K., Le, V. and Froehlich, J. 2013. Combining Crowdsourcing and Google Street View to Identify Street-level Accessibility Problems. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)* (New York, NY, USA, May 2013), 631–640.

[25] Hara, K., Le, V., Sun, J., Jacobs, D. and Froehlich, J. 2013. Exploring Early Solutions for Automatically Identifying Inaccessible Sidewalks in the Physical World Using Google Street View. *Human Computer Interaction Consortium* (2013).

[26] Hara, K., Shiri, A., Campbell, M., Cynthia, B., Le, V., Pannella, S., Moore, R., Minckler, K., Ng, R. and Froehlich, J. 2013. Improving Public Transit Accessibility for Blind Riders by Crowdsourcing Bus Stop Landmark Locations with Google Street View. *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility Technology* (2013), 16:1–16:8.

[27] Hara, K., Sun, J., Chazan, J., Jacobs, D. and Froehlich, J. 2013. An Initial Study of Automatic Curb Ramp Detection with Crowdsourced Verification using Google Street View Images. *Proceedings of the 1st Conference on Human Computation and Crowdsourcing (HCOMP'13), Work-in-Progress* (2013).

[28] Hoiem, D., Efros, A. and Hebert, M. 2008. Putting Objects in Perspective. *International Journal of Computer Vision*. 80, 1 (2008), 3–15.

[29] Hruschka, D.J., Schwartz, D., St.John, D.C., Picone-Decaro, E., Jenkins, R.A. and Carey, J.W. 2004. Reliability in Coding Open-Ended Data: Lessons Learned from HIV Behavioral Research. *Field Methods* . 16 , 3 (Aug. 2004), 307–331.

[30] Jain, S.D. and Grauman, K. 2013. Predicting Sufficient Annotation Strength for Interactive Foreground Segmentation. *ICCV* (2013), 1313–1320.

[31] Kamar, E., Hacker, S. and Horvitz, E. 2012. Combining Human and Machine Intelligence in Large-scale Crowdsourcing. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1* (Richland, SC, 2012), 467–474.

[32] Kirschbaum, J.B., Axelson, P.W., Longmuir, P.E., Mispagel, K.M., Stein, J.A. and Yamada, D.A. 2001. *Designing Sidewalks and Trails for Access, Part II of II: Best Practices Design Guide, Chapter 7*.

[33] Kopf, J., Chen, B., Szeliski, R. and Cohen, M. 2010. Street Slide: Browsing Street Level Imagery. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*. 29, 4 (2010), 96:1 – 96:8.

[34] Krippendorff, K.H. 2003. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Inc.

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

[36] Lin, C.H., Daniel, M. and Weld, S. 2012. Dynamically switching between synergistic workflows for crowdsourcing. *In Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI '12* (2012).

[37] Malisiewicz, T., Gupta, A. and Efros, A.A. 2011. Ensemble of Exemplar-SVMs for Object Detection and Beyond. *ICCV* (2011).

[38] National Council on Disability 2007. *The Impact of the Americans with Disabilities Act: Assessing the Progress Toward Achieving the Goals of the ADA*.

[39] Princeton Vision Toolkit: *http://vision.princeton.edu/code.html*. Accessed: 2014-04-07.

[40] Quinn, A.J. and Bederson, B.B. 2011. Human Computation: A Survey and Taxonomy of a Growing Field. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2011), 1403–1412.

[41] Rundle, A.G., Bader, M.D.M., Richards, C.A., Neckerman, K.M. and Teitler, J.O. 2011. Using Google Street View to audit neighborhood environments. *American journal of preventive medicine*. 40, 1 (Jan. 2011), 94–100.

[42] Stollof, E.R. and Barlow, J.M. 2008. *Pedestrian Mobility and Safety Audit Guide*.

[43] Su, H., Deng, J. and Fei-Fei, L. 2012. Crowdsourcing Annotations for Visual Object Detection. *AAAI Technical Report, 4th Human Computation Workshop* (2012).

[44] Tibshirani, R. 1994. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*. 58, (1994), 267–288.

[45] United States Department of Justice, C.R.D. 1990. *Americans with Disabilities Act of 1990, Pub. L. No. 101-336, 104 Stat. 328*.

[46] Viola, P. and Jones, M. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*. 1, (2001), 511.

[47] Walk Audit: *http://streetswiki.wikispaces.com/Walk+Audit*. Accessed: 2014-04-15.

[48] Welinder, P. and Perona, P. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels. *In W. on Advancing Computer Vision with Humans in the Loop* (2010).

[49] Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E. and Quan, L. 2008. Image-based Fa\ÇAde Modeling. *ACM Trans. Graph.* 27, 5 (Dec. 2008), 161:1–161:10.

[50] Xiao, J., Fang, T., Zhao, P., Lhuillier, M. and Quan, L. 2009. Image-based Street-side City Modeling. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 114:1–114:12.

[51] Zamir, A.R., Darino, A. and Shah, M. 2011. Street View Challenge: Identification of Commercial Entities in Street View Imagery. *ICMLA (2)* (2011), 380–383.

[52] Zamir, A.R., Dehghan, A. and Shah, M. 2012. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. *ECCV (2)* (2012), 343–356.

[53] Zamir, A.R. and Shah, M. 2010. Accurate Image Localization Based on Google Maps Street View. *Proceedings of the European Conference on Computer Vision (ECCV)* (2010).