Exploring Early Solutions for Automatically Identifying Inaccessible Sidewalks in the Physical World using Google Street View

Kotaro Hara, Victoria Le, Jin Sun, David Jacobs, Jon E. Froehlich

Department of Computer Science, University of Maryland, College Park, MD 20742, USA
{kotaro, jinsun, djacobs, jonf}@cs.umd.edu; vnle@umd.edu

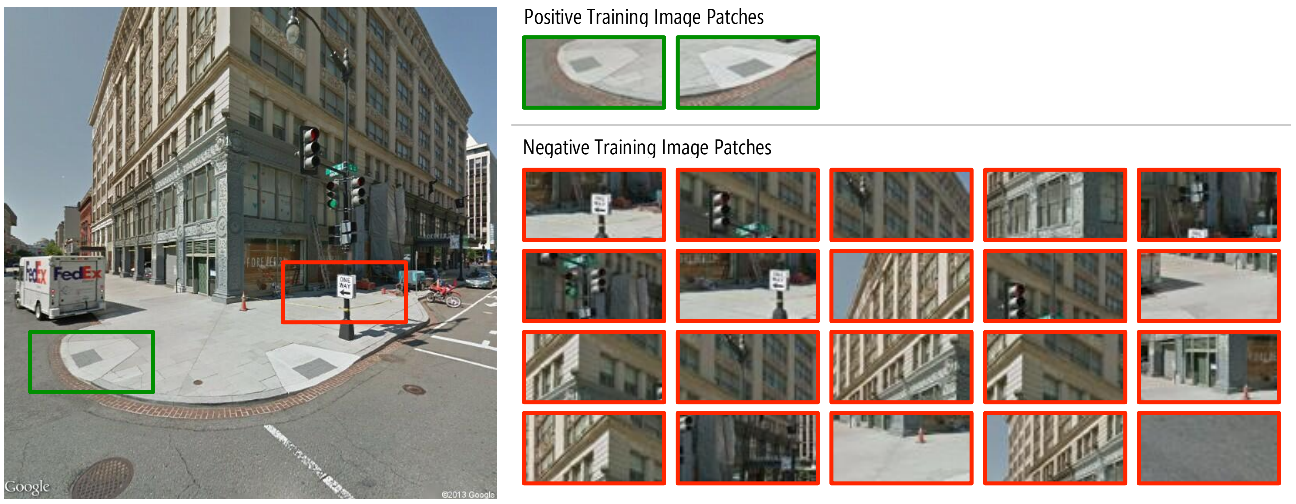
Positive Training Image Patches
Negative Training Image Patches

Figure 1: In this paper, we perform an early exploration of applying computer vision techniques to automatically identify sidewalk accessibility attributes in Google Street View (GSV). Specifically, we explore the automatic identification of the presence/absence of curb ramps (or "curb cuts"). We manually trained our classification/detection algorithm with positive training examples (green outlines above) and negative training example (red outlines) from GSV images.

Abstract

Poorly maintained sidewalks, missing curb ramps, and other obstacles pose considerable accessibility challenges. Although pedestrian- and bicycle-oriented maps and associated routing algorithms continue to improve, there has been a lack of work focusing on accessibility. There is currently no way for a user to determine accessible areas of a city prior to travel. In this paper, we explore the use of computer vision techniques (a linear SVM) to detect sidewalk accessibility problems in Google Street View imagery. This is early work in a large on-going project. Here, we focus on automatically identifying one key accessibility barrier: the presence/absence of curb ramps. Our preliminary results point to the potential of using automatic, highly-scalable approaches to extract information about physical accessibility from online map imagery.

Introduction

The lack of street-level accessibility information can have a significant negative impact on the independence and mobility of citizens (Nuernberger 2008; Thapar et al. 2004). Although maintenance issues such as buckled or cracked sidewalks can pose significant accessibility challenges so too do larger, more permanent infrastructural issues such as utility poles or fire hydrants directly in sidewalk paths or the lack of curb ramps at intersections. The problem is not just that sidewalks remain inaccessible, but also that there are no mechanisms to determine accessible areas of a city before travel.

In our previous work, we investigated whether crowd workers (turkers) recruited from Amazon Mechanical Turk (AMT) could locate, categorize, and assess sidewalk accessibility problems in a manually curated set of GSV images (Hara, Le, and Froehlich 2013). With simple quality control methods, we found that turkers were were able to identify sidewalk accessibility problems with an accuracy of 93%. However, crowd sourcing is, by nature, labor intensive. Here, we focus on investigating automated methods to improve the scalability and efficiency of our approach (see Figure 1 and 2).

We expect that computer vision and machine learning can be used in a variety of ways including **(i) triage:** automatically scanning large map regions for potential accessibility problems and utilizing crowdsourcing for verification; **(ii) view selection:** automatically identifying the most appropriate view to show crowdworkers

Copyright is held by the author/owner(s)
HCIC'13, June 23-27, 2013, Pacific Grove, California, USA

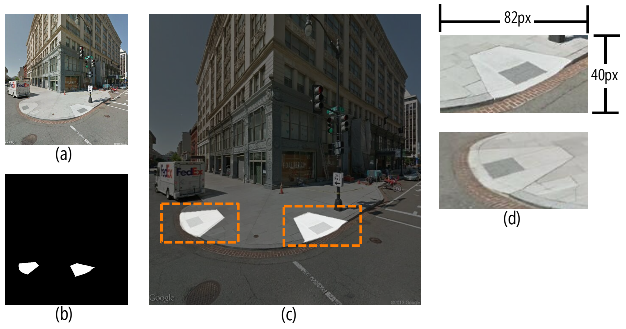

Figure 2: (a) The original GSV image; (b) A curb ramp mask provided by a member of the research team; (c) Masked parts of the original image are cropped; (d) The cropped image patches are resized to 82x40 pixel.

unobscured by cars or other occlusions; **(iii) mensuration:** it's difficult to ascertain distance and measurement properties in a GSV scene—computer vision can help determine key attributes such as a width of a sidewalk.

In this boaster, we focus on point (i) above: in particular, exploring a simple, automated approach to detecting the presence/absence of curb ramps in GSV images.

Related Work

Our research is related to four threads of prior work: machine learning-based object detection; the application of computer vision in the domain of assistive technologies; the use of omnidirectional street-level images as a datasource; and the use of crowdsourcing for accessibility. We briefly describe each in turn.

The sliding window approach for object detection, which we employ here, is widely used in computer vision (Dalal and Triggs 2005; Liu et al. 2012; Rowley, Baluja, and Kanade 1996). Rowley *et al.* introduced this technique to detect human faces by classifying whether a small window that is moved over an image (thus a "sliding window") contains a face. Liu *et al.* used a similar approach for identifying a dog breed by detecting a face and also face parts of a dog. Dalal and Trigg used it for detecting pedestrians in city scenes. In this paper, we use this approach to detect curb ramps in images of street intersections collected from GSV.

Applications of computer vision-based object detection algorithms in the accessibility domain are primarily aimed at supporting people with visual impairments (Lee, Leung, and Medioni 2012; Ivanchenko, Coughlan, and Shen 2009; Vázquez and Steinfeld 2012). For example, Lee *et al.* built a real-time staircase detector to support low-vision or blind people. Similarly, Ivanchenko *et al.* developed a crosswalk detector to help blind users appropriately navigate crosswalks. Vázquez and Steinfeld developed a mobile application that suggests a better picture composition to support blind users to take a better picture. Although related, our approach is different; we are using GSV imagery for offline detection of sidewalk accessibility attributes—in particular curb ramps—to build a database that can be used for accessibility-aware routing algorithms.

The use of GSV in computer vision has also become increasingly popular although not in the context of accessibility (*e.g.,* Doersch et al. 2012; Xiao and Quan 2009; Zamir and Shah 2010). Xiao *et al.* used GSV imagery as dataset to investigated novel techniques of multiple-view image segmentation; Zamir and Shah developed an algorithm to match user-generated images and GSV images to localize where user-generated images were taken; Doersch used GSV images to extract geographically informative elements of different cities (*e.g.,* shape of a window in Paris, iron railing in London), potentially useful for computational geography. However, to our knowledge, nobody has worked on detecting sidewalk accessibility attributes in GSV imagery.

Although computer vision techniques continue to improve, it is unlikely that a purely automated approach will achieve a sufficiently high level of accuracy to create a useful system. Thus, we expect that any solution will likely incorporate manual annotations or corrections from human users as well. With the advent of the Internet and the emergence of crowdsourcing labor markets (*e.g.,* Mechanical Turk), there has been an increase of crowdsourcing for accessibility (Bigham, Ladner, and Borodin 2011). For example Lasecki *et al.* developed a real-time captioning algorithm that harness human-power to achieve high-quality video captioning (Lasecki et al. 2012). Most relevant to this boaster is our past work utilizing crowdsourcing and GSV (i) to identify sidewalk accessibility problems to support people with mobility impairments (Hara, Le, and Froehlich 2012; Hara, Le, and Froehlich 2013) and (ii) to find bus stop landmarks to help blind people locate bus stops (Hara et al. 2013).

Dataset and Study Method

To train an automatic curb ramp detector, we collected images of intersections in Washington DC from GSV. We used a linear Support Vector Machine (SVM) classifier trained with Histogram of Oriented Gradient (HOG) features and evaluated its classification performance.

Collecting Curb Ramp Image Patches

We manually collected 73 GSV images with visible curb ramps from intersection in Washington DC (Figure 2a). In each image, one member of the research team drew an outline around curb ramps to create mask images (Figure 2b). We cropped a rectangular image patch from each GSV image so the outlined area fit in an image patch (Figure

2c). We measured the median width and median height of cropped image patches, and reshaped all of them to fit to that dimension (*window size*=82x40 pixel) (Figure 2d). Using this method, we obtained 109 curb ramp image patches across the 73 GSV images. Similarly, we randomly cropped 1,090 image patches that do not contain curb ramps from the same GSV images.

Curb Ramp Classifier

To build a curb ramp classifier, we used a linear Support Vector Machine (SVM) implemented in SVM-light (Joachims 1999)—a well-known SVM library—with the commonly used HOG feature descriptor implemented in VLFeat (Vedaldi and Fulkerson 2008). Training a classifier is a two-step process: we extract a set of features (feature vector) from an image patch and label it as a positive (+1) or a negative (-1); we then feed a set of pairs of a label and a feature vector into a training algorithm.

We extracted HOG feature vectors from manually-labeled positive and negative curb ramp image patches. To obtain a HOG feature vector from an image patch, we first calculate a gradient vector—a pair of values that describe how pixel color intensity changes along x- and y-axis—at each pixel on the image patch. We split the image patch into 10x5=50 small subsections ("*cells*") of 8x8=64 pixels. This is one of the cell sizes that is used in Dalal's original work that introduced the HOG (Dalal and Triggs 2005). In each cell, we count a number of gradient vectors within ranges of angles. We discretized gradient vector angles by 11.6 degrees to split them into 31 bins (31x11.6=360). This produces a gradient histogram. Note: 11.6 degrees is a default setting in VLFeat. We concatenate gradient histograms from all of 50 cells and obtain 1,550-dimension (31x50) feature vector for each image patch. A feature vector that contains a curb ramp image patch is labeled +1 while a feature vector that contains no curb ramp image patches is labeled as -1. We use these labels and associated vectors as positive and negative training examples.

We train a SVM classification model implemented in SVM-light. We used 87 out of 109 positive examples and 870 out of 1090 negative examples to train the classifier. We used the remaining 22 positive examples and 220 negative examples for testing.

Curb Ramp Detector

We use the curb ramp classifier we obtained to implement a sliding window curb ramp detector, which is a three-step process: First, we crop an image patch from the top-left corner of an image with a 82x40 pixel window. Then we use the trained classifier to classify if the cropped image patch is a curb ramp or not. To do this, we move the window to the left for one pixel and continue the process until we reach the bottom-right corner. Because consecutive windows with visible curb ramps will likely be classified as curb ramps, we cluster detected points that are close to each other with a mean shift clustering—a clustering algorithm that does not require a number of clusters before runtime.

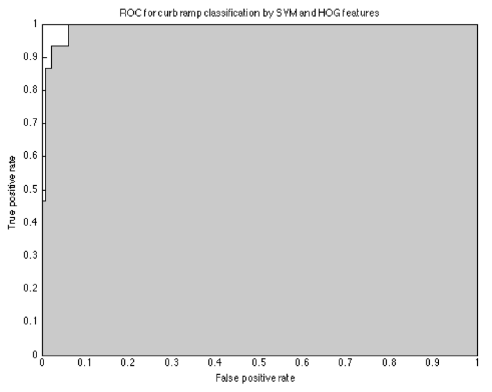

Figure 3: The ROC curve that shows the performance of SVM classifier trained with HOG features on classifying presence or absence of a curb ramp in an image patch. The vertical axis shows true-positive rate of classification (i.e., correctly identifying all the curb ramp image patches), whereas the horizontal axis shows false-positive rate (i.e., classifying something that is not a curb ramp as a curb ramp.)

ROC Curve and Area Under the Curve

We used two common measures used in computer vision to evaluate the performance of our curb ramp classification: the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) measure. These allow us to observe the trade-off between false positive and false negative detections, as the parameters of the system are varied. We have two axes in a chart for the ROC curve: a vertical axis representing true-positive rate (TPR) and a horizontal axis representing false-positive rate (FPR). An ideal classifier draws a curve that intersects with a point (TPR, FPR) = (1.0, 0.0) which correspond to the top-left corner of a chart, thus we have AUC=1.0. Therefore, closer the AUC to 1.0, better the classifier performance.

Results

In this section, we investigate how well our curb ramp classifier can classify the curb ramp image patches.

Figure 3 shows the ROC curve for our curb ramp classifier. As we noted in the previous section, the ideal classifier will have TPR=1.0 and FPR=0, which corresponds to the top-left corner of the graph, and AUC=1. As we vary a parameter, TPR and FPR changes and draws a curve with AUC=0.996. When FPR=0 we have TPR=0.773, *i.e.,* the classifier manages to correctly classify about three quarter of curb ramp images, but

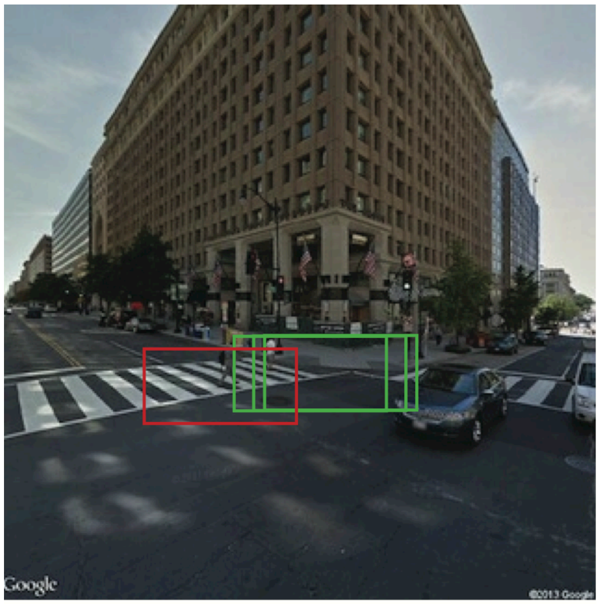

Figure 4: An example of sliding window curb ramp detection. Points classified as a curb ramp is enclosed by red rectangles. Curb ramps are correctly enclosed by the rectangles, but you can also see a part of the crosswalk is classified as a curb ramp, which is incorrect.

importantly does not misclassify any non-curb ramp image as a curb ramp. On the other hand, when TPR=1.0, we have FPR=0.04, which means we identify all curb ramp images but misclassify 4% of non-curb ramp images as a curb ramps.

In Figure 4, we visually illustrate a detection result with an arbitrarily selected GSV image. Rectangles represent curb ramp classifications; there are four in total—three green rectangles are correct. The fourth (*i.e.,* red rectangle) is part of the crosswalk, which is misclassified as a curb ramp.

Conclusion and Future Work

In this boaster, we presented early work of our first attempt in developing a curb ramp detector. We showed it is possible to classify curb ramp and other using SVM classifier with HOG features (AUC=0.996) albeit with a very small dataset. Future work includes using a larger dataset to train and test the curb ramp classification performance and to explore detection mechanisms for other types of sidewalk accessibility attributes (*e.g.,* sidewalk obstacles such as poles and fire hydrants).

Acknowledgements

This work was supported by an NSF grant (IIS-1302338) and a Google Faculty Research Award.

References

Bigham, Jeffrey P, Richard E Ladner, and Yevgen Borodin. 2011. "The Design of Human-powered Access Technology." In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '11)*, 3–10. New York, NY, USA: ACM.

Dalal, N, and B Triggs. 2005. "Histograms of Oriented Gradients for Human Detection." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference On*, 1:886–893 vol. 1.

Doersch, Carl et al. 2012. "What Makes Paris Look Like Paris?" *ACM Trans. Graph.* 31 (4) (July).

Hara, Kotaro et al. 2013. "Improving Public Transit Accessibility for Blind Riders by Crowdsourcing Bus Stop Landmark Locations with Google Street View." In *In Submission to The Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility Technology*.

Hara, Kotaro, Victoria Le, and Jon Froehlich. 2012. "A Feasibility Study of Crowdsourcing and Google Street View to Determine Sidewalk Accessibility." In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12), Poster Session*, 273–274. New York, NY, USA: ACM.

Hara, Kotaro, Victoria Le, and Jon Froehlich.. 2013. "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)*, tbd. New York, NY, USA: ACM.

Ivanchenko, V, J Coughlan, and H Shen. 2009. *Staying in the Crosswalk: A System for Guiding Visually Impaired Pedestrians at Traffic Intersections*. *Assistive Technology Research Series*. Vol. 25. NIH Public Access.

Joachims, Thorsten. 1999. "Advances in Kernel Methods." In , ed. Bernhard Schölkopf, Christopher J C Burges, and Alexander J Smola, 169–184. Cambridge, MA, USA: MIT Press.

Lasecki, Walter et al. 2012. "Real-time Captioning by Groups of Non-experts." In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, 23–34. New York, NY, USA: ACM.

Lee, Young Hoon, Tung-Sing Leung, and G Medioni. 2012. "Real-time Staircase Detection from a Wearable Stereo System." In *Pattern Recognition (ICPR), 2012 21st International Conference On*, 3770–3773.

Liu, Jiongxin et al. 2012. "Dog Breed Classification Using Part Localization." In *Computer Vision – ECCV 2012 SE - 13*, ed. Andrew Fitzgibbon et al., 7572:172–185. Springer Berlin Heidelberg.

Nuernberger, Andrea. 2008. "Presenting Accessibility to Mobility-Impaired Travelers (Ph.D. Thesis)". University of California, Santa Barbara.

Rowley, Henry, Shumeet Baluja, and Takeo Kanade. 1996. "Human Face Detection in Visual Scenes." In *Advances in Neural Information Processing Systems 8*, 875–881.

Thapar, Neela et al. 2004. "A Pilot Study of Functional Access to Public Buildings and Facilities for Persons with Impairments." *Disability and Rehabilitation* 26 (5) (January 1).

Vázquez, Marynel, and Aaron Steinfeld. 2012. "Helping Visually Impaired Users Properly Aim a Camera." In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, 95–102. New York, NY, USA: ACM.

Vedaldi, A, and B Fulkerson. 2008. "{VLFeat}: An Open and Portable Library of Computer Vision Algorithms."

Xiao, Jianxiong, and Long Quan. 2009. "Multiple View Semantic Segmentation for Street View Images." In *Computer Vision, 2009 IEEE 12th International Conference On*, 686–693.

Zamir, Amir Roshan, and Mubarak Shah. 2010. "Accurate Image Localization Based on Google Maps Street View." In *Proceedings of the European Conference on Computer Vision (ECCV).*