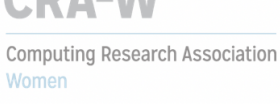



CREU 2017-2018 Final Report: Using Google StreetView Images to Identify Crosswalks

CREU Mentor Julie Medero, Harvey Mudd College

CREU Team Member Cordelia Stiff, Harvey Mudd College

1 GOALS AND PURPOSE

Street level infrastructure can dramatically affect communities. Research has shown that the addition of various street level features, such as crosswalks, can decrease the rate of collissions between cars and pedestrians [8]. However, most of the work done in this field relies on hand collected data, which can make studies harder to conduct. Google StreetView provides an avenue for researchers to gather large amounts of data on the infrastructure, and the changes that have ocurred in that infrastructure, in a given area. By utlizing image processing techniques, we can attempt to identify various street level features, and thus reduce the amount of necessary data gathering work. As Google StreetView also hosts historical imges of sites as far back as 2007, this also allows researchers to examine what infrastructure changes have ocurred over a period of time.

In this paper, we will focus on identifying crosswalks in Google StreetView images. Previous work has been done in identifying crosswalks in moving images, but less work has been done in identification of these features in still images [4][3]. Google StreetView images have their own unique challenges, most notably in image quality and modification of lines in the image.

In order to properly identify crosswalks, a neural network trained on the ImageNet database, along with a selective search algorithm, is used to identify various windows that are likely to be crosswalks [1] [2]. A series of Hough Line Transforms is then used to identify the vanishing point of the image, and with that, identify the area of the image that is most likely to be the street. Finally, a decision tree is trained on a variety of metrics calculated for each image, and decision tree is used to determine whether an image is likely to contain a crosswalk.

In testing on 76 positive images and 78 negative images, the trained classifier ranges from 46% to 77% accuracy depending on the separation of data between the training and testing sets, and the maximum depth of the decision tree allowed. On average, the classifier works correctly 61% of the time.

The rest of this paper is organized as follows. Section 2 describes the previous work that has been done in crosswalk detection. In Section 3, the methodology for finding the images and then classifying them is outlined. The results of the

classifier are detailed in Section 4, and the results are discussed in Section 5. Finally, Section 6 lists future work that could be done on this project.

2 RELATED WORK

Previous work has been done on identifying street level features in moving images, and in identifying various features within Google StreetView images.

2.1 Identifying Features in Moving Images

The increase in popularity of autonomous cars has led to an understandable increase in the detection of street level features in moving images. One area in which this is very common is the detection of road signs, and research in this area dates back to the 1990s [5].

Piccioli, De Michelib, Parodi, and Campani lay out a three step procedure for identifying signange. First, they restrict their search area using knowledge of where the signs should be (generally, on the right of the road). Then, they use color to restrict the search area even further, as they are able to search for the red piece of the sign. However, in order to detect the road signs themselves, they rely on the *shape* of the sign [5]. While the techniques of restricting based on area and color are applicable to the task of identifying crosswalks in Google StreetView images, shape is less possible due to the nature of StreetView Images. As crosswalks can be oriented in any direction, and may be only half visible in an image, there is no distinct shape to look for (as opposed to the circles and triangle shaped road-signs that Piccioli, De Michelib, Parodi, and Campani focused on). The breaking of lines due to the removal of the StreetView car can also distort the shape of the crosswalk, making a shape based approach likely untenable (see Figure 5).

Work is also prevalent in the detection of crosswalks themselves. However, many of these approaches rely on a camera moving towards the object, instead of a single image (see [3] and [4]). Since Google StreetView only provides single images (or, in the best case, images taken over multiple months), these methods cannot be applied to this work.

2.2 Previous work with Google StreetView

Google StreetView has been used by researchers to identify various features. Previous studies have used Google StreetView to identify the amount of greenery in urban areas, and street-level accessibility problems, such as missing curb ramps or poorly maintained sidewalks [6] [7]. While the study to identify greenery does use an automated mechanism to classify the amount of greenery, this work is not subject to many of the complications that arise from focusing on crosswalks - they mainly focus on the color levels of the image, and thus will not be affected by the line breaks that occur in StreetView images when the car is removed from the image [6].

Hara, Le, and Froehlich's is subject to some of the same issues with line modification, as it inspects features like

sidewalks and curbs, but this study relied on crowdsourcing their detection [7]. Our work differs in that we are aiming for a solely automatic approach, without a human element in the detection process.

3 PROCESS

This section describes our crosswalk detection algorithm. An overview of the process is shown in Figure 1.

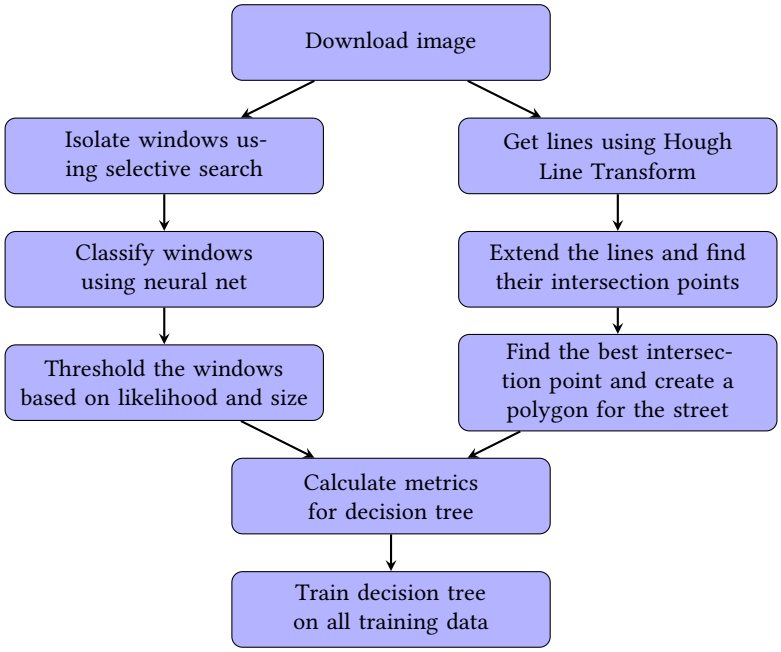


Fig. 1. A flowchart describing the steps of the algorithm

3.1 Data

First, images need to be obtained from Google StreetView. Using the Google StreetView API, we are able to download images from any location. In order to ensure that we were able to obtain enough crosswalk images for testing, we randomly generated latitude and longitude pairs within New York City, where there would likely be a higher ocurrence of crosswalks. We then looped through those pairs, retrieving the images for each pair (when they existed). Since one piece of the premise of this work is being able to identify changes in infrastructure over time, we restricted locations to those in which there were at least distinct sets of images (taken at different times). Once all the images were downloaded,

(a) A crosswalk that is simply an outline of the path. This would be considered a negative result.

(b) A crosswalk that is the more traditional "zebra stripe" style. This would be considered a positive result.

Fig. 2. Examples of the two what is considered a crosswalk for the purposes of this work

they were hand-classified to allow for testing. By continuing the classification until we had a reasonable sample of positive images, we ended up with 76 positive images, and 2,209 negative images.

3.2 Crosswalk Definition

First, we need to define what images we consider as "having crosswalks." In our dataset, we had two main types of crosswalks - the more traditional "zebra striped" crosswalks, and ones that are simply an outline of a path (examples of which can be seen in Figure 2.) In the classified dataset, we had 76 zebra stripe crosswalks, and 83 of the outlined crosswalks. When consulting the ImageNet database that the neural network is trained on, it is clear that the database only considers the zebra stripe style of crosswalk. Hence, for this project, we will only consider these types of crosswalks as positive results.

3.3 Window Isolation and Classification

The first step of our algorithm is identifying and classifying areas of the interest in the image. First, a previously existing selective search algorithm is used to identify areas of interest in the image. [1]. Once these areas are identified, a deep neural network trained on the ImageNet database is used to give the probability that each of these windows belongs to any of the 21,000 different classes that the classifier handles.[2] These results are then saved into a dataframe, which we can then filter to simply include the classification results of street level features for each window.

3.4 Thresholding

Once we've gotten the relevant classification data from the classification code, we normalize the dataframe based on various street level features. These features are "Pedestrian crossing", "Street sign", "Streetlight", and "Traffic light". While the ImageNet database does allow the classification of other street-related classes, such as intersections, these four were selected as they would represent a section of the image, instead of the entire image. This also allows us to better filter out windows that fit other common street features. Then, we remove the windows that (a) don't meet our threshold value, and (b) are larger than some fraction of the image. By removing windows that don't hit a certain threshold, we can attempt to restrict our results to just those that are the most likely to be the features we are looking for. This allows us to remove large windows that cover large sections of the image, and only serve to obscure the classification results.

3.5 Street Detection

Our approach depends on identifying the area of the image that most likely contains the street itself. First, we find the vanishing point of the image. For each image, we first use a Hough Line Transform to find the lines in the image. Then, after removing any line that is shorter than 20 pixels to eliminate noise, we extend all those remaining and calculate the various intersection points of these lines. Next, we loop through every intersection point, and create lists of sets of points, where each point is within 30 pixels of every other point in the set. This allows us to account for the fact that, while many of these points will not exactly overlap, they are close enough together to be considered as the same point. We then take the average of each smaller set of points, to get a list of points. Finally, we find the point in that list that represents the intersection of the most lines, and also is in the top half of the image (as vanishing points should not be in the bottom half of the image), and take this to be our vanishing point.

Now that we have the vanishing point, we can take advantage of a feature of Google StreetView images to identify the street. For most StreetView images, the street ends at approximately the same place - approximately 200 pixels up from the bottom corners of the image. Therefore, by simply drawing lines from our vanishing point to these points on the image, we can correctly find the street in most images, which can be seen in Figure 3.

3.6 Final Classification

As there is error in the window detection algorithm, we cannot rely solely on thresholding the windows. In order to isolate images that are on the street, we first calculate a number of values from the image. The values we calculate are:

(1) The number of pixels that the classifier believes are crosswalks
(2) The number of pixels that the classifier believes are crosswalks that are also on the street
(3) The number of pixels that the classifier believes are crosswalks that are included in at least two different windows
(4) The number of pixels that the classifier believes are crosswalks that are included in at least two different windows, and are also on the street

Fig. 3. An image with the street polygon outlined in blue.

(5) The number of windows that the classifier found
(6) The number of windows that the classifier found that are completely within the street
(7) The number of windows that the classifier found that are partially within the street

We take these numbers, and a variety of percentages calculated from these numbers (such as the percentage of pixels that are believed to be crosswalks, or the percentage of pixels that are on the street), and use them to train a decision tree. Weuse the decision tree to classify our dataset. For the decision tree, we first split our data using scikit learn's TRAIN_TEST_SPLIT, and then calculate the accuracy for a variety of different required maximum depths (see Figure 4 for an example of such a tree.)

4 RESULTS AND DISCUSSION

When the final decision tree code is run with a total sample of 154 images (with 76 positive and 78 negative), a threshold of 0.97 and removing windows whose size is greater than a quarter of the overall image, and a train/test split of 0.33, we get accuracies ranging from 46% to 70%, with an average of 60% over five trials, trying 9 different depths each time. This accuracy changes slightly when we consider positive results versus negative - on average, 51% of positive images were classified correctly, and 70% of negative images were classified correctly.

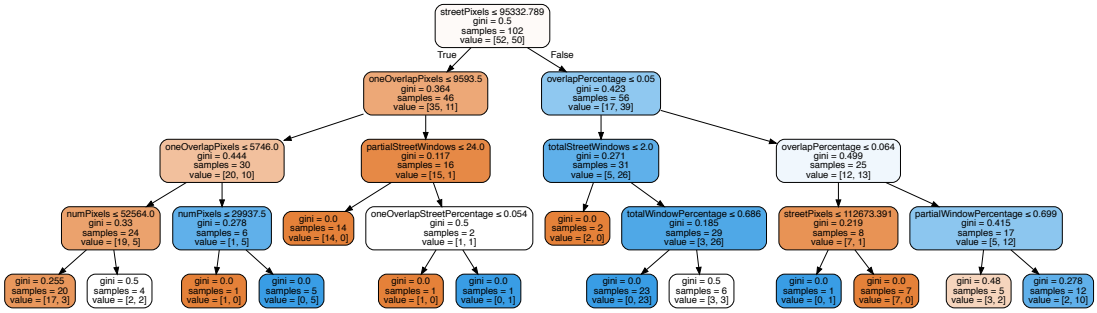


Fig. 4. An example of a decision tree, trained on a threshold of 0.97 and area restriction of 4. This tree had an accuracy of 70%.

Table 1. Results for with a threshold of 0.97, area restriction of 4, averaged over 5 trials.

Maximum Depth of Decision Tree	Accuracy on Positive Samples	Accuracy on Negative Samples	Overall Accuracy
1	0.529	0.808	0.666
2	0.429	0.914	0.672
3	0.506	0.652	0.58
4	0.484	0.632	0.56
5	0.548	0.676	0.61
6	0.518	0.634	0.574
7	0.568	0.676	0.622
8	0.54	0.646	0.594
9	0.494	0.648	0.572

These results change slightly when run on different thresholds. When the threshold values are changed from 0.97 and 4 to 0.93 and 3, our results become slightly worse, with an average accuracy of 52%.

Unfortunately, since the classifier is only slightly more accurate than a random choice, the results of this project are not as successful as they could be. However, there is slight improvement over the initial classifier in some metrics. The neural net that we are using has a classification accuracy of 41.9% for the first result (that is, 41.9% percent of the time, the classifier's highest result is accurate) [2]. Since we are able to classify with a higher accuracy, we do have a slight improvement over the initial classifier - however, this is still less successful than the classifier's output once the top five or top twenty initial results are considered.

This lack of results could be due to several different reasons. One clear problem is the frequency that a crosswalk-like pattern appears in street images. In a lot of images, there can be false positives from the classifier due to areas of the image that appear to be crosswalks (such as the windows being incorrectly classified in Figure 6). Thus, since the neural

Fig. 5. An example of an image where post-processing distorts the crosswalk in the image

net classifier legitimately believes that it has found a crosswalk, our various noise reduction steps (such as thresholding windows, or checking only overlapping windows) won't work. This is even more true when these patterns occur on a street, as even our metrics that restrict windows to just those on the street will still consider these windows as proper classsifications.

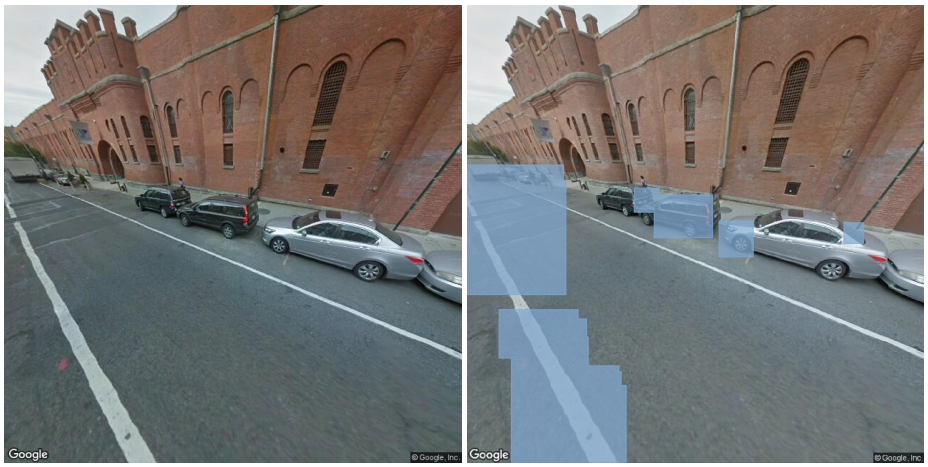
Another possible cause for error is the effect of post-processing that is done in Google StreetView images. As can be seen in Figure 5, Google StreetView does post-processing to remove the car collecting the data from the image. This can result in fragmented lines on crosswalks, which could also lead to problems in image classification.

5 FUTURE WORK

Since we've not yet obtained a definite result, this work could be continued in several directions.

5.1 Obtaining a Result

One way in which this work could possibly be retooled is by retraining the top layer of the neural network on just crosswalks. The neural net that we are currently using is based on an image database that has 21,000 different classification options. By taking the top layer of the neural net and just retraining on crosswalks (and possibly expanding the definition of a crosswalk to include non-striped crosswalks), we may be able to get more accurate results.



(a) Original Image (b) Classification Results

Fig. 6. Example of an image in which a non-crosswalk pattern on the street (in this case, the upper left corner) is identified as a crosswalk by our classifier.

Another possible avenue of future work involves the metrics that we are passing to the decision tree. There are several metrics that we could try, including metrics based on the color of the windows (we'd expect the white stripes), the approximate shape of the largest polygon (we'd expect some sort of rectangle), or some more analysis based on position of the windows themselves on the street.

5.2 Test Sample

Once we had an algorithm, we would need to test it more thoroughly. This would ideally be done by randomly generating latitude and longitude points, and then downloading all StreetView images for that location. However, since the number of places that don't have crosswalks is much smaller than those who do, we would also likely need to look for more crosswalks. This can be achieved by generating random latitude and longitude points within large cities, where crosswalks are far more common. This would be repeated until a large enough test sample is achieved.

5.3 Applications of Working Algorithm

Once a working algorithm was found, this work could be easily expanded to apply to street signs, simply by retooling the decision tree classification, and changing some of the thresholding parameters. This would likely also be true for any of the various street-level features that the ImageNet database contains.

Since this research provides a tool for identifying infrastructure on a larger scale, the results of this project also open

the door for research projects in several other fields. Projects that seek to determine where infrastructure changes (or doesn't change), what infrastructure levels are like in particular cities, or the equity of infrastructure access could be more easily conducted using the results of this work. This work also provides tools to non-profit organizations, who can better identify conditions on the ground in the various places that they are working.

6 WEB LINKS

Cordelia Stiff: cstiffactivetransportation.wordpress.com

CRA-W
Computing Research Association
Women

REFERENCES

[1] K.E.A. van de Sande, J.R.R. Uijlings, T. Gevers, and A.W.M. Smeulders. Segmentation as Selective Search for Object Recognition ICCV, 2011.

[2] Sergey Ioffe and Christian Szegedy Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[3] Choi, J., et al. "Crosswalk and Traffic Light Detection via Integral Framework." The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision, 2013, pp. 309-12. IEEE Xplore

[4] Haselhoff, A., and A. Kummert. "On Visual Crosswalk Detection for Driver Assistance Systems." 2010 IEEE Intelligent Vehicles Symposium, 2010, pp. 883-88. IEEE Xplore

[5] Piccioli, G., et al. "Robust Method for Road Sign Detection and Recognition." Image and Vision Computing, vol. 14, no. 3, Apr. 1996, pp. 209-23. CrossRef.

[6] Li, Xiaojiang, et al. "Assessing Street-Level Urban Greenery Using Google Street View and a Modified Green View Index." Urban Forestry & Urban Greening, vol. 14, no. 3, 2015, pp. 675-85. CrossRef.

[7] Hara, Kotaro, et al. "Combining Crowdsourcing and Google Street View to Identify Street-Level Accessibility Problems." ACM Press, 2013, p. 631. CrossRef.

[8] Retting, Richard A., et al. "A Review of Evidence-Based Traffic Engineering Measures Designed to Reduce Pedestrian-Motor Vehicle Crashes." American Journal of Public Health, vol. 93, no. 9, Sept. 2003, pp. 1456-63. Atypon.