Object Detection for Mapping Road Signs via Google Street View

Cyrus Wentz

Original Paper:

- Detecting and mapping traffic signs from Google Street View images using deep learning and GIS. Andrew Campbell, Alan Both, Qian (Chayn) Sun 2019. Computers, Environment and Urban Systems, vol. 77. September 2019. https://doi.org/10.1016/j.compenvurbsys.2019.101350.

Introduction:

Motivation:

As the use of GPS navigation apps and self-driving vehicles have expanded greatly in the past decade, the need for accuracy therein has become commensurately critical for navigation, safety, and ease of use. Suffice it to say, with both human and AI drivers aided by accurate navigation information, driving has never been easier. Of particular interest for this project in this regard is the accurate information of road signage such as speed limits, stop signs, and traffic lights.

However, in many places this navigational information is either incomplete, inaccurate and/or non-existent, due to the cost and complexity of surveying and archiving such information. Manually created GIS and asset management systems are not financially feasible for every municipality and additionally require trained individuals to create and manage such. The lack of such information then, in turn, limits the effectiveness of navigational software in less accurately mapped locales. Automated methods then may present a cost effective solution with an acceptable level of accuracy. In particular, automated methods leveraging public data such as Google Street View offer the possibility of a low-cost, accurate, widespread method to document road signage types and locations in under-mapped or un-mapped areas.

Problem Statement:

The purpose of this project is then to ascertain the effectiveness of using existing road signage datasets to fine-tune a pre-trained object detection architecture to then be used to photogrammetrically locate and map objects to real-world space from a given image detection and GSV geolocation data, following in the method of Campbell, et. al. (Campbell, et. al. ,p.6). More specifically, this project attempts to fine-tune an instance of SSD MobileNet architecture pre-trained on ImageNet data to detect and classify four types of road signs, up from the 2 of the original Campbell, et. al. paper. The trained model would then be used on a small collection of Google Street View Images to validate the efficacy of the photogrammetric methods of the original paper, comparing them to their estimated real-world locations.

Challenges:

The nature of road signage itself poses a significant challenge to the efficiency of an automatic road sign mapping system due to multiple, interrelated issues, namely:

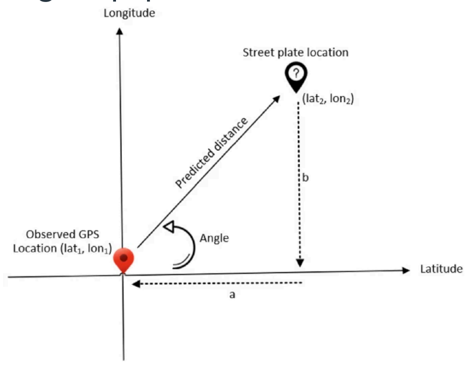
- The real world imbalance of common and rare types of signage affecting any road signage dataset's balance of classes
- The high variability of certain types of signs such as speed limit signs and others with variable text, as well as non-standard colors, shapes, etc.
- The different systems of road signage around the world limit the interoperability of any single automatic mapping system with signs of different languages, purposes, shapes, etc.

Perhaps due to the above, the breadth of road signage datasets found for this project were limited, both in quality and quantity, often combining multiple systems of road signage. Nevertheless, with careful analysis of a dataset and application of augmentation techniques, these issues can be mitigated, even in the case of international datasets (such as the classes of such a dataset each being uniform in their system of signage, i.e. from one country). Additionally, by focusing on a few select types of road signage, models may fare better than those trained on a multitude of different classes of road signs.

Related Work:

ODRP: a new approach for spatial street sign detection from EXIF using deep learning-based object detection, distance estimation, rotation and projection system, Taşyürek, M.:

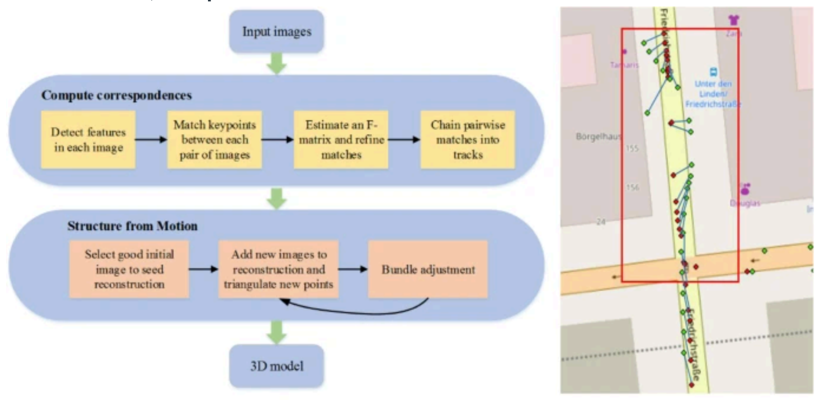
The ODRP approach outlined in this paper follows a very similar methodology to the original paper. However, instead of utilizing Google Street View images, the ODRP approach makes use of EXFIL images to provide the geolocation data for mapping images to physical locations, not being limited to streets unlike GSV imagery (though other locations exist for GSV, consistent and high quality is not guaranteed. Additionally, the method of location mapping is markedly simpler than that in the original paper, mitigating sources of error in the location mapping. However, notably the simpler method requires the target location to be focused in the center of the image. The paper's slightly simpler method provides an excellent alternative to potentially gauge the effectiveness of the original paper's method.



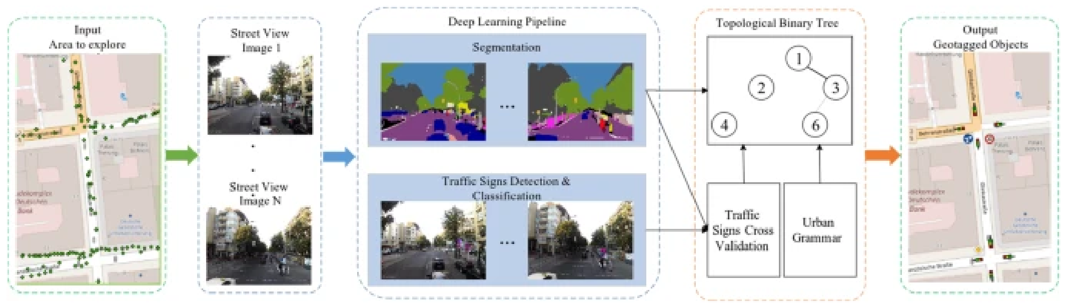
(visualization of ODRP localization method (Taşyürek, p. 991))

Automated detecting and placing road objects from street-level images, Zhang, C., Fan, H. & Li, W.:

In contrast to both the original Campbell et. al. paper and the Taşyürek paper, the method of road sign mapping in the Zhang et. al. paper is based on a considerably different method of mapping road sign locations. Instead of a pure object detection/classification method, the paper describes a three-module pipeline: the first acts as preprocessing and outputs a sequence of street view images generated by Mapillary additionally using a technique called Structure from Motion to correct misaligned images via matching image features across images in a sequence to form a 3d model of image points., The second module is a segmentation/object detection architecture based off of PSPNet for segmentation and a combination of YOLOv3 and a small convolutional model developed for this paper for detection/classification called ShallowNet, working together to provide robust detections and classifications for the results of the semantic segmentation. Finally, the third module acts as the object localization method, using a novel Attributed Topological Binary Tree to define the spatial relationships between objects in an image with rules based on real-world standards and norms. This approach notably has the advantage of being robust, consistent, and accurate in its predicted locations. However, this comes at the cost of flexibility, with the urban rules being particularly important to be accurate to the surveyed area(s). In regards to the original paper and this project's methodology, it is both a reminder of the tradeoff between accuracy and flexibility in this task and the paper's idea of common urban rules informed what augmentations should (or should not) be present in a dataset.


Input images
Compute correspondences
Detect features in each image
Match keypoints between each pair of images
Estimate an F-matrix and refine matches
Chain pairwise matches into tracks
Structure from Motion
Select good initial image to seed reconstruction
Add new images to reconstruction and triangulate new points
Bundle adjustment
3D model
Friedrichstraße
Unter den Linden/Friedrichstraße

(a) (b)



(overview of SfM technique (top) (Zhang et. al., p.5), overview of 3-module pipeline (bottom) (Zhang et. al., p.4))

Methodology:

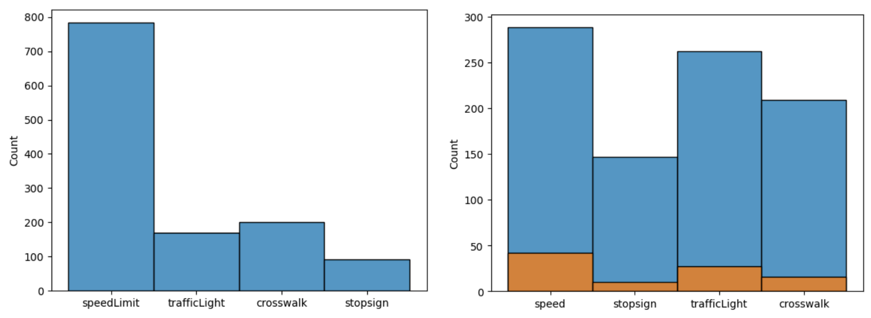
The overall methodology of this project follows that of Campbell et. al. in "Detecting and mapping traffic signs from Google Street View images using deep learning and GIS". However, due to a limited time and scope of this project and some technical limitations, this project differs in two significant ways:

1. Due to the original paper's dataset consisting of curated and manually annotated Google Street View Images, this project has opted for a pre-existing road sign database, to additionally investigate the efficacy of existing datasets in this task.
2. This project uses an instance of SSD MobileNet pre-trained on the ImageNet 1k version 1 dataset.

Preprocessing:

Some preprocessing methods on this dataset were necessitated due to the variable image sizes, including resizing each image to a standard size of 320x320 pixels with 3 color channels. As the bounding boxes for each object in an image were defined by pixel vertices in each image, each set of vertices had to be converted into a binary image, resized to the standard 320x320, then processed back into pixel vertices. In some cases with small bounding boxes, the area would be zero and throw errors when passed through the SSD model, so a 1x1 padding was added to each bounding box. Additionally, a random horizontal flip, along with brightness and contrast augmentations were applied to the training set. The custom preprocessing method to handle these augmentations would additionally build these into lists of tensors to be turned into batchable datasets.

Due to the data concerns listed in the challenges section of the introduction, the exploration of preprocessing techniques was critical in the process of maximizing model performance. The dataset used was particularly imbalanced, with speed limit signs consisting roughly 60% of the total objects in the dataset, with multiple methods tried to mitigate the imbalance: class weights in the SSD MobileNet architecture, majority undersampling, minority oversampling via augmentations , and without any alterations. Class weights however, were by and large incompatible with the Pytorch implementation of SSD MobileNet used and thus were omitted as a method to mitigate the class imbalance.



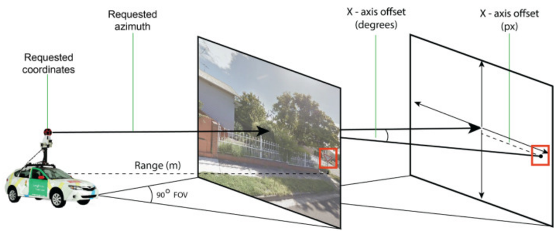
(dataset distribution before balancing (left) and after using under-over sampling (right))

Model Training and Evaluation:

The model training methodology took the form of a naive hyperparameter search, iterating over six hyperparameters (trainable backbone layers in MobileNet, learning rate, number of detections per image, the top-k detections per class, detection score threshold, and bounding box overlap threshold (NMS threshold)) and training and evaluating each combination.The highest performing model would be saved and taken for further hyperparameter testing not included in the hyperparameter search. Metrics used for model evaluation included the batch mean Intersection over Union average per image for bounding box regression accuracy for training and validation, mean average precision to gauge classification accuracy for training and validation, and training and validation losses to judge the training routine. The Mean Average Precision was used as the deciding metric for best model, as the implementation of this metric took into account the bounding box IoU and thus represented a holistic metric of model performance.

Google Street View Photogrammetry:

Using the Google Street View API, street view images nearest to a known road sign could be requested given longitude, latitude, and a bearing in degrees (Wheeler). Using this method, a very small collection of locations and headings near road signs were gathered to test the photogrammetry methods of the original paper (Campbell, et. al. ,p.6).



$$offset\ (mm) = (sign\ \ length\ mean\ (px) - image\ focal\ center\ (px)) \tag{1}$$

$$offset(\theta) = arctan\left(\frac{offset\ (mm)}{focal\ length}\right) \tag{2}$$

$$sign\ \ bearing\ (\theta) = offset\ (\theta) + GSV\ vehicle\ bearing\ (\theta) \tag{3}$$

$$range\ (m) = 1000 \times focal\ length\left(\frac{sign\ \ height\ (m)}{sign\ \ height\ (px) \times pixel\ size}\right) \tag{4}$$

$$Sign\ Easting(m) = GSV\ vehicle\ Easting\ (m) + range\ (m) \times sin(sign\ \ bearing\ (\theta)) \tag{5}$$

$$Sign\ Northing\ (m) = GSV\ vehicle\ Northing\ (m) + range\ (m) \times cos(sign\ \ bearing\ (\theta)) \tag{6}$$

(visual representation of photogrammetry method (left), photogrammetry method formulas (right) (Campbell, et. al. ,p.6))

The above process was applied to the curated list of Google Street View Images after having been passed through the fine-tuned SSD MobileNet instance, logging the process's values and finally averaging the error across all tests.

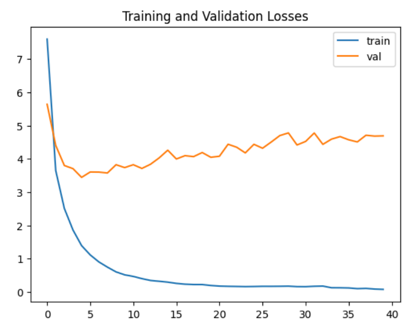
Results:

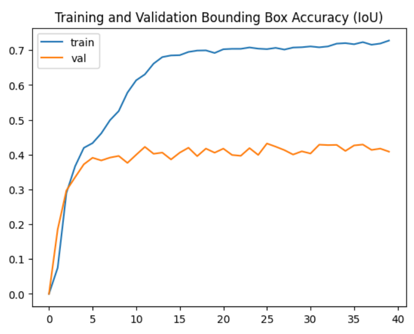
SSD MobileNet Fine-Tuning:

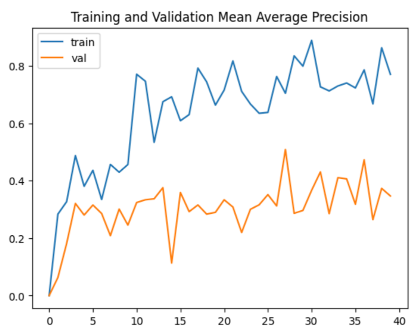
Despite many attempts, techniques, and methods of optimization, the fine-tuning results for this project were fairly poor. Likely due to the dataset used, being far too variable and of inadequate quantity for this task, or insufficient hyperparameter testing, the model struggled to learn the dataset, limiting the potential performance of any fine-tuned model instance. Additionally, the Pytorch implementation used for SSD MobileNet was fairly inflexible, and attempts with add class weights did not turn out to be very successful. Hyperparameter search could possibly been improved, with a better algorithm and larger range of hyperparameters, however computational and temporal feasibility prevented a more in depth hyperparameter search. The best hyperparameters found through hyperparameter search and manual testing are as follows:

- Pretrained Weights: IMAGENET1K_V1
- Trainable Backbone Layers: 3
- Detections per image: 100,
- Topk Candidates: 7
- Score threshold: 0.8
- NMS threshold: 0.55
- Learning rate: 0.002

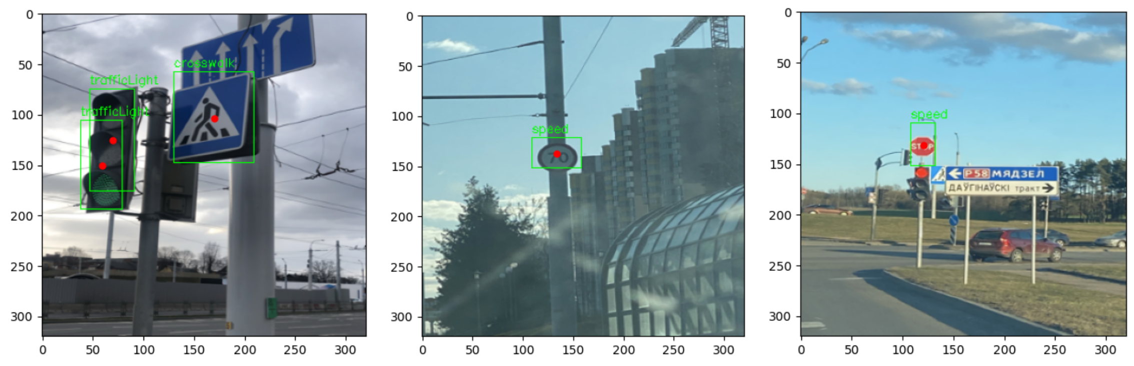
It should be noted that some hyperparameters can be adjusted at inference time to improve performance. In regards to performance metrics, as previously mentioned, Intersection over Union was tracked as the accuracy for bounding boxes, Mean Average Precision was used to gauge classification accuracy, and validation loss provided an additional, more general measure of model performance. Below are graphs of the best model instance fine-tuned:





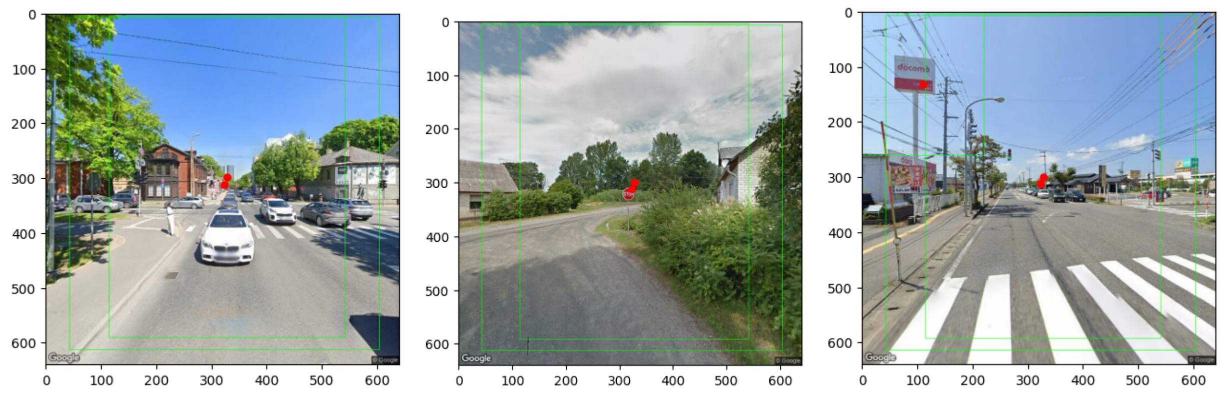


As can be seen in the above graphs, the model struggles significantly to converge, just barely starting to overfit to the training data around epoch 5. Similarly both IoU and MAP metrics separate training and validation scores around this point, indicating a maximum level of learning being found for the validation set for this instance. Disappointing metric results aside, when testing the trained object detection on the validation set, somewhat promising results can be seen:



(Examples of fine-tuned object detection model performance on validation set)

However, when tested on completely different images, performance suffers greatly. Some amount of performance degradation is expected when testing on new images, but this level implies more serious issues:

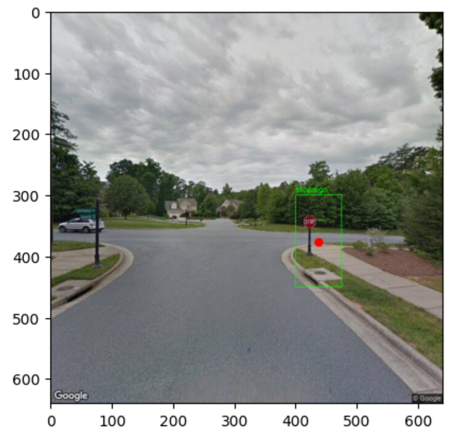


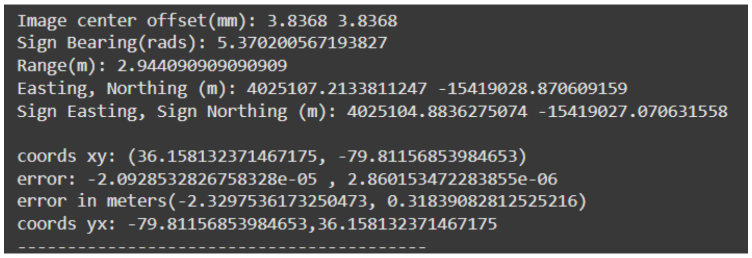
(Examples of object detection on completely unseen images)

Whilst the centers of each bounding box (noted with a red dot) are somewhat accurate to detectable objects, the bounding boxes are way too large, hampering any real use of them and there is still considerable misclassification though it is not as bad as the bounding box regression.

Google Street View Photogrammetry:

In contrast to the fine-tuning results, the results from the Google Street View photogrammetry object localization method proved to be quite successful. While the results of the object localization depend significantly on the results of the object detection, taking fairly accurate detections and using such as the basis for object localization still gives fairly accurate results:





```
Image center offset(mm): 3.8368 3.8368
Sign Bearing(rads): 5.370200567193827
Range(m): 2.944090909090909
Easting, Northing (m): 4025107.2133811247 -15419028.870609159
Sign Easting, Sign Northing (m): 4025104.8836275074 -15419027.070631558

coords xy: (36.158132371467175, -79.81156853984653)
error: -2.0928532826758328e-05 , 2.860153472283855e-06
error in meters(-2.3297536173250473, 0.31839082812525216)
coords yx: -79.81156853984653,36.158132371467175
----------------------------------------
```

(demonstration of photogrammetry method accuracy with a manually created bounding box)

Even with poor bounding boxes, this method can reach a reasonable level of accuracy, seen below with the average results of the small test set of GSV images:

```
Average Coordinate Error Lon/Lat: [-5.06259080e-05 -6.88975367e-05]
Average Coordinate Error in meters:(-5.635650295371984, -7.669638702071685)
```

Conclusions:

Overall, this project highlights the lack of quality in existing road signage datasets and how critical quality and specificity is in this task. As road signage contains highly variable features, having a highly specific dataset, limited to only a few classes with minimal variety, and limited to only one system, country, etc. of road signage is of the utmost importance to achieving competitive performance in this task. Despite the poor findings of this project, it has illuminated the strict requirements of this task. On the other hand, the testing of the photogrammetry method of the original paper proved to be very fruitful, even if with room for improvement. As such, any future work will focus heavily on rigorous collection of specific data and likely a reevaluation of model selection to focus on a more suitable model for this task, as well as tweaking the object localization method for the target area (i.e. a projection more accurate to the target area for swapping between coordinates and northing/easting). Whilst it may indeed be possible for random road signage datasets to perform well in this task, the top priority should be to reinforce the basal, abstract concepts of road signs to ensure a model's generalization on such highly variable data. As a final note, while this project did not achieve the results of the original Campbell et. al. paper, it served as a great learning experience for the limits of ML and the varying technical requirements that a task such as this may require.

Works Cited

Campbell, Andrew, et al. “Detecting and mapping traffic signs from Google Street View images using Deep Learning and GIS.” *Computers, Environment and Urban Systems*, vol. 77, Sept. 2019, p. 101350, https://doi.org/10.1016/j.compenvurbsys.2019.101350.

Taşyürek, Murat. “ODRP: A new approach for spatial street sign detection from exif using deep learning-based object detection, distance estimation, rotation and projection system.” *The Visual Computer*, vol. 40, no. 2, 17 Mar. 2023, pp. 983–1003, https://doi.org/10.1007/s00371-023-02827-9.

Zhang, Chaoquan, et al. “Automated detecting and placing road objects from street-level images.” *Computational Urban Science*, vol. 1, no. 1, 4 Aug. 2021, https://doi.org/10.1007/s43762-021-00019-6.

Wheeler, Andrew. “Using Python to Grab Google Street View Imagery.” *Andrew Wheeler*, 10 Feb. 2024, andrewpwheeler.com/2015/12/28/using-python-to-grab-google-street-view-imagery/.

Sharing:

If despite this paper’s poor results, feel free to share, but please make it anonymous