Detecting Objects Using Google Street View Data

PRINCETON UNIVERSITY

Brian Matejek
Advisor: Thomas Funkhouser

Introduction

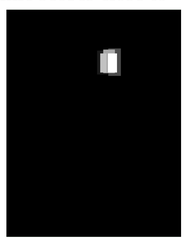
- Combining Google Street View (GSV) data and existing object recognition software, this project addresses two new methods for determining the real locations of objects at a street intersection.
- I focused on finding traffic lights and stop signs through a segment of New York City.
- The first algorithm uses a Hough voting procedure to locate regions that multiple images believe have a high probability of containing an object.
- The second algorithm generates a system of linear equations to define the object recognition problem.
  - The variables correspond to all of the pixels in the images and grid samples spaced every half meter.
  - The least squares solution provides the grid probabilities.

Existing Research

- Google Street View cars drive around various cities collecting LIDAR and image data.
  - GPS trackers, accelerometers, and inertial meters on the car provide a general path for the car.
  - I worked with a 3.35 square kilometer region of Manhattan that contained 150,525 images.
- Pedro Felzenszwalb, Ross Girshick, et. al. created an object detection software which produces prediction bounding boxes.
  - I adapted this program, referred to as the VOC detection software.
  - I used a series of C++ libraries created by members of the Graphics Lab and the linear equation solver (CSparse) created by Timothy A. Davis.

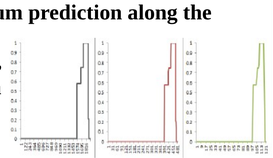
Creating Probabilities from VOC Predictions

- The VOC detection software returns bounding box predictions for objects with scores ranging from -1.0 to 2.0.
  - Scores close to -1.0 correspond to very low probabilities while scores close to 2.0 have very high probabilities.
- I used a low and a high threshold to further restrict the range of scores.
  - Any score below -0.5 received a probability of 0.0 and any score above 1.0 received a score of 1.0.
  - All scores in between were linearly assigned a value between 0.0 and 1.0.



Projection Into Two Dimensions

- In the largest considered New York City run, there are 511,485,829,120 pixels and 189,099,350 grid samples, assuming a grid sample every half meter.
  - These numbers produce infeasible memory and time demands.
  - I contracted the z-axis to help alleviate these demands.
- Image samples are determined by taking the maximum prediction along the vertical scanline for every x coordinate in the image.
- To further reduce the memory needed for large runs, I created multiple sampling functions by applying an averaging filter and sampling at a lower frequency.
- For simplicity, image samples will still be referred to as pixels.

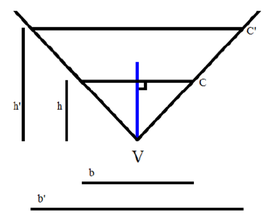


Evaluation Metrics

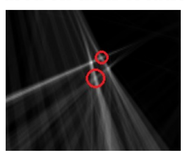
- I created three evaluation metrics to determine how well the existing object detection software and the newly created programs performed.
- The first evaluation tested how well the VOC predictions performed in three dimensions.
  - I projected every truth data point into the image to see if a point fell within a bounding box.
  - If no points projected into the bounding box, the box was a false positive.
  - I created precision and recall curves by iterating through all of the VOC bounding box predictions in decreasing order of score.
    - The recall was determined by the number of unique true objects found (i.e. two images that captured the same traffic light did not increase the recall).
- The second evaluation tested how well the VOC predictions performed in two dimensions.
  - After projecting every truth data point into the image, a true positive occurred when the projected x coordinate fell in the range of the bounding boxes x coordinates.
- The third evaluation tested the probability grids generated by my programs.
  - The truth points were projected into an X, Y grid space and a true positive was a prediction with 1.5 meters of an actual object.
  - By considering grid samples in order of decreasing probabilities, I generated precision and recall curves.
- I computed the average precision for these precision and recall curves.

Hough Voting

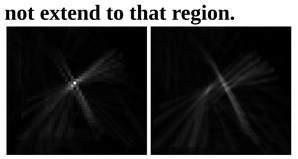
- Every pixel of every image casts a vote for all of the grid samples that intersected the ray from the camera viewpoint through the pixel.
  - This corresponds to all of the locations that the pixel could have captured.
  - The image to the right indicates a bias caused by this voting procedure.
    - The sum of the weights along each cross section is equal. However, the length of the cross sections varies leading to a higher density of votes for cross sections closer to the viewpoint.
    - This bias can be eliminated by weighting pixel votes proportionately to the distance from the cross section to the viewpoint.



- Since pixels vote for all of the grid samples along the ray starting from the viewpoint, there are many grid samples that receive a high number of votes based on their location.
  - The image on the right illustrates two of these locations.
  - After the initial round of Hough voting, I iterated through every pixel again and the pixels voted for the one grid sample along its ray that received the highest number of votes in the previous round.



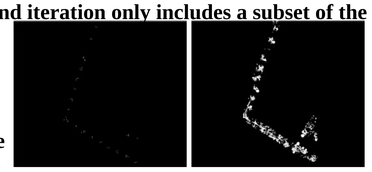
- I also created a maximum distance to limit how far a pixel voted.
  - At certain distances, an object's bounding box would be so small that no object detection software could accurately predict it.
  - The optimal maximum distance varied between traffic lights and stop signs and was determined by using separate training and validation data.
- This Hough voting method performed better than the standard object detection software on all but three combinations of objects and runs.
  - In one of the failures, the algorithm voted for three locations that actually did correspond to true locations but the LIDAR data did not extend to that region.
  - Another failure occurred when the car drove through a construction zone leading to a high number of false positives caused by the scaffolding.
  - The last failure could be fixed by using an aligned car path. The figure to the right illustrates this.



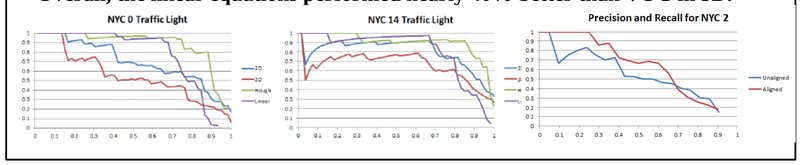
  - Multiple unaligned segments (right) spread their votes out leading to diluted predictions. The aligned segments (left) vote for one true location.
  - This Hough voting algorithm performed around 55% better than VOC in 3D.

Linear Equations

- I created a system of linear equations to model a city.
  - $p_{i,j}$ represents the probability that the j^{th} pixel in image i corresponds to part of an object of interest.
  - $g_{x,y}$ represents the probability that the location (x, y) contains an object of interest.
- A grid sample maps to a pixel if the ray from the viewpoint of the camera to the pixel intersects the grid sample.
- For every pixel, the sum of the probabilities of the grid samples that map to that pixel must equal the probability of that pixel having captured an object.
- For every pixel, there is a data term where the pixel equals the probability assigned by the VOC object detection software.
- There are smoothing terms for the grid samples and pixels.
- The least squares solution to this system of linear equations produces the grid probabilities for containing an object of interest.
- I implemented a coarse-to-fine algorithm which first solves the problem with grid samples spaced every 2 meters. The second iteration only includes a subset of the grid variables.
  - The image on the right shows the results for run 1 after the coarse level. The white spots on the left image are the true locations of traffic lights. The white areas on the right image are the locations that moved on to the fine algorithm.



  - The recall after the coarse pass matters most since locations that are not considered in the fine pass cannot be predicted.
- I also restrict the distance that pixels can map to for the same reason as mentioned in the Hough voting section.
- After the program solves the least squares problem, only local maxima are considered. The grid smoothing term creates regions with high probabilities based only on their placement near true locations. By considering only the local maxima, this effect is minimized.
- The system of linear equations performed well on most of the runs, although not as well as the Hough voting generally.
- For many runs, the linear equations had a high precision over most of the recall and dropped suddenly as seen below and to the left.
- The alignment errors in the car seem to have a large effect on this prediction program as seen in the bottom right graph.
- Overall, the linear equations performed nearly 40% better than VOC in 3D.



Future Developments

- I would like to collect truth data for multiple different street intersection objects to test how well these programs work on a diverse set of objects.
- I did not collect truth data for the five largest New York City runs, but I would like to eventually see how these algorithms scale.
  - Some of the runs overlap certain streets in New York City. I want to see how the algorithms perform when analyzing New York as a continuous unit.
- Most of the truth data came from unaligned GSV runs. After collecting truth data in the aligned runs, I would like to analyze which algorithm is most robust to errors in the car's path.
- With three dimensional LIDAR object detection software, I could augment the system of linear equations so that the grid variables have an additional data term equal to the detected probability in three dimensions.