GIS-Assisted Object Detection

Alejandro Torroella Amir Roshan Zamir
University of Central Florida
Center for Research in Computer Vision
alejandrotorroella@knights.ucf.edu
amir.roshanzamir@gmail.com

Abstract

We propose a method to improve object detection results by introducing GIS data into the fold. Our current system uses post-processing techniques on the object detection results to get rid of most of the false positive detections and acquire some of the true positives. We show that GIS data can greatly improve object detection results on images which have a GPS tag of where the image was taken and metadata on the orientation (yaw, pitch, roll) of the camera at the time the image was taken. Our method is in its early stages and even still we are seeing promising results.

1. Introduction

Object detection has been a long studied problem within computer vision. It's a fundamental task in many applications and till this day has been plagued with numerous false positives and missed true positives. Novel object detection methods always improve over previous algorithms, but never reach perfect results for all cases.

Our model takes a different approach by instead introducing GIS data that can be used to improve any object detection method. Using GPS coordinates of various objects, camera location and orientation, we can project onto the image an estimation of where we expect these objects to be located within the image. Occlusion handling is taken into consideration using polygon data of building outlines, as well as GPS coordinates of foliage and the objects in question themselves. Using the obtained back projection we can immediately coarsely prune bounding boxes to improve the object detection results. We further improve results by generate all possible combinations of the remaining bounding boxes and compare them to our back projected estimate using our devised cost function. The combination that returns the best result from our cost function is deemed the best arrangement and the bounding boxes that compose this combination are deemed correct.

Our unique approach at object detection has given us promising results thus far, which will be shown later in the paper. We plan on continuing our research to achieve even better results and plan on submitting our work to be published in CVPR 2014.

2. Object Detection

One of the first steps in our process is actually to run an object detection algorithm on our query images. There are many object detection algorithms available, but they are all similar in that they must all have some sort of scanning process to find objects within the image, as well as results being defined by some bounding box pixel locations. By exploiting these two similarities among object detection methods, our method can be applied to virtually any object detection algorithm to improve on its results.

2.1. Discriminately Trained Part Based Models

For our experiments we decided to implement object detection via Discriminately Trained Part Based Models (DPM) by Pedro *et al.* [1]. We trained DPMs on eight object classes including: traffic signs, traffic signals, street lights, bus stops, ATMs, trash cans, fire hydrants and gas pumps. The training dataset we used was largely on images obtained from Image-Net, which was a great resource for annotated images, however some object classes did require additional annotated images. The images were annotated in PASCAL format to make the DPM training code easy to implement.

Figure 1 shows some of our object detection results using the DPMs. As you can see, the results are almost random for some images, however while our results aren't a good representation of the capabilities of DPMs, even the best DPM results don't come close to 100% accuracy. We purposefully chose a low threshold for high recall in order to at least get the true positives among the sea of false positives. This is necessary for our method to even have a chance of obtaining perfect results.

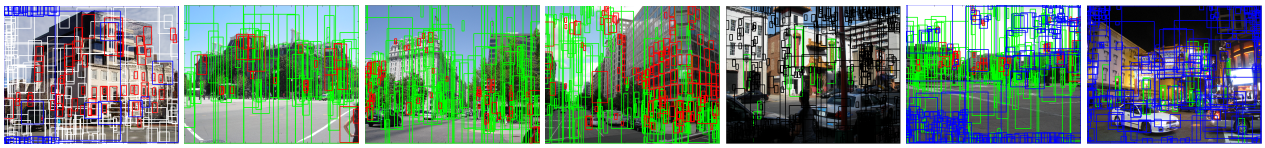


Figure 1. DPM object detection results before GIS fusion. Colored bounding boxes represent different object classes. Blue bounding boxes represent traffic signs, red bounding boxes represent traffic signals, green bounding boxes represent street lights, white bounding boxes represent trash cans, and black bounding boxes represent fire hydrants.

3. Sensor Model

In order to project our GPS coordinates onto the image plane we implement a robust sensor model. Image metadata should include: the *GPS coordinate* of where the image was taken, the *focal length* used, the *sensor dimensions*, and the *yaw* (α), *pitch* (β), and *roll* (γ) of the camera at the time the image was taken. Publicly available GIS data will at least give latitude and longitude coordinates for any given object class. Some available GIS data may also specify a altitude for the objects in question; this is useful, but not necessary. We will assume that altitude is not know for any of our object classes and instead will use empirical data to estimate the height of our object classes.

3.1. Coordinate conversion

All of our data is in geodetic coordinates, which makes it near impossible to project our objects onto the image plane. For this reason, we first convert our geodetic coordinates to the Earth-Center, Earth-Fixed (ECEF) Cartesian coordinate system. To simplify things even further we convert to a local tangential coordinate system with the origin at the location of the camera, the z-axis pointing cardinal north, the x-axis pointing east, and the y-axis pointing up. By converting to this local East-North-Up (ENU) coordinate system, we make projecting onto the image plane almost trivial.

$$X_{Geo}(lat, long, alt) \rightarrow X_{ECEF}(X, Y, Z)$$

$$X_{ECEF}(X, Y, Z) \rightarrow X_{ENU}(E, N, U)$$

3.2. Perspective projection

To correctly back project our objects onto the image plane we want objects that are further away from the camera to appear smaller than objects that are closer. We achieve this by implementing a perspective projection model. First we calculate the angle of view using the sensor dimensions and the focal length f of the camera. We also find the pixel to meter ratios m_x and m_y by using the sensor dimensions and the image resolution. We assume a pin-hole camera model with the principal points (p_y, p_y) being at the center of the image. Objects within the angle of view are then projected onto the image using the following transformations:

$$x_{pixel} = KRX_{ENU}$$

where

$$K = \begin{bmatrix} fm_x & 0 & p_x \\ 0 & fm_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma)$$

where

$$R_z(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & sin\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix}$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

3.3. Occlusion handling

One of the problems we ran into when implementing our sensor model was that objects that were in close proximity and within the angle of view of our camera, but occluded by some obstruction were being projected onto the image as if they were visible in the image. We compensate for this by again exploiting the utility of GIS data. Objects that are behind polygons of building outlines, foliage, or other objects which we have GIS data for are excluded in the projection process.

3.4. Additional constraint

Even after taking into account occlusions, we still are left with the possibility that objects that are extremely far away are being considered to be visible within the image. However, due to the limitations of computer vision, which relies on there being sufficient pixels to be able to differentiate and detect objects, these extremely far projected objects need to be discarded because there is simply no possibility



Figure 2. Sensor model back projections on some of our query images. Blue artifacts represent traffic signs, red artifacts represent traffic signals, green artifacts represent street lights, white artifacts represent trash cans, and black artifacts represent fire hydrants.

that the object detection method will even detect them. Figure 2 shows our back projections for the images we ran the DPMs on.

4. GIS Fusion

Once we have our back projected estimate of the true arrangement and our preliminary object detection bounding boxes, we attempt to fuse the two to improve the object detection.

4.1. Coarse pruning

We begin by first sifting through the many bounding boxes and attempt to remove ones that are completely erroneous. We assume that our back projection has a significant amount of error due to GIS data not being completely reliable most of the time. Assuming this, we want to very coarsely sift the bounding boxes to make sure that the true positives are not lost in the pruning process due to the error in the GIS, otherwise our method is set up for failure.

We prune in two ways, once spatially and again by size. We spatially sift our bounding boxes by taking a look at our back projection and removing those bounding boxes that completely disagree with our back projection. For example; if there are bounding boxes for a particular class of object on the left side of the image, while according to our back projection there are none on that side of the image, then those bounding boxes are removed. We sift by size by looking at the pixel height of our bounding boxes and compare them to the pixel height of our back projection results and apply a low threshold to further rid of bounding boxes that are definitely incorrect. Figure 3 shows our results of our pruning process.

4.2. Cost function

Once we have our coarsely pruned bounding boxes, we have our object detection input which we use to find the best combination. However, to find the "best" combination we need some sort of quantitative measure as to how good a particular combination is. Taking into account that the back projection has some degree of error, we devise a cost function that generates a value when given the back projection and a particular combination, defined as

$$cost = \sqrt{2 * std_x^2 + std_{y1}^2 + std_{y2}^2}$$

where

$$std_x = std(GIS_{px} - combination_{ix})$$

$$std_{y1} = std(GIS_{py1} - combination_{iy1})$$

$$std_{y2} = std(GIS_{py2} - combination_{iy2})$$

We use the standard deviation of the pixel distances between the back projection and the combination in question. By using the standard deviation, we make our cost function more invariant to the degree of error in the back projection. This is due to our assumption that the degree of error in the back projection will be somewhat consistent among all the objects within the field of view.

4.3. Combinatorial fusion

Now that we have defined our cost function, we apply it to our approach by traversing all possible combinations of bounding boxes, where order doesn't matter and no repetitions can occur. Each combination is passed to our cost function which gives a score to each one. The best combination is that which fulfills the following.

$$Best = \underset{combination_i}{\operatorname{argmin}} \; cost(GIS_p, combination_i)$$

The obtained combination is our final result and our results show that there is a very significant improvement on the object detection results.

4.4. Multi-class fusion

As you may have noticed, traversing through all possible combinations while taking into account multiple classes can be ludicrously expensive. Taking into account *j* classes this translates to:

$$\binom{N_1}{k_1} * \binom{N_2}{k_2} \cdots \binom{N_i}{k_i}$$

where N_i is the number of bounding boxes for class i and k_i is the number of back projections for class i. As you



Figure 3. DPM object detection results after our coarse pruning step. Color representations are the same as previous figures.

can see the number of possible combinations increases at an alarming rate as we increase the number of classes to detect for.

To facilitate the computation complexity we attempt to further prune our coarsely pruned bounding boxes by applying our combinatorial fusion algorithm on every possible combination of classes, choosing two at a time and keeping track of the top performing combinations. Bounding boxes found in these top performing combinations are kept, otherwise they are discarded.

By running the combinatorial fusion algorithm on all possible combinations of classes, we minimize the risk of potentially losing valuable bounding boxes, while still greatly decreasing the amount of combinations we have to traverse.



Figure 4. Traffic Signals: 1/1 Trash Cans: 0/1

5. Results

Our results show that GIS data can assist object detection in a significant way. Our method is in its early stages and yet we are seeing promising results.

6. Conclusion

In conclusion we've shown that GIS data can improve object detection results by very effectively discarding most false positives, as well finding the true positives. GIS data in general has a significant degree of error, whether objects are unaccounted for or their locations are off by some degree. Not all GIS data is created equal, however our method needs the GIS data to be correct to some degree in order to get good results. With that in mind, GIS technologies can only improve over time, which can only relate to more accurate data and in turn, our method will perform better.

Even with improvements in GIS technologies, we can safely assume that there will always be some error and there are inaccuracies in the camera's metadata as well that we have no control over. This calls for suitable optimization methods. Our method is in its early stages, but we see that a more complex cost function is necessary.

7. Future Work

Our future research will involve developing a more complex cost function in order to further improve our results. We also plan on introducing an early fusion step. So far our method only incorporates late fusion, where we post process the object detections results to obtain the true positives. In the future, we wish to use the GIS data in the actual detection step. This should translate to less false positives before our coarse pruning step and less missed true positives.

References

[1] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), September 2010.

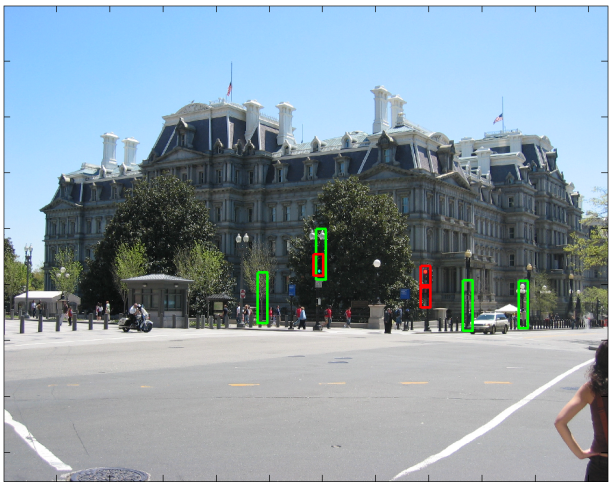


Figure 5. Street Lights: 0/4 Traffic Signals: 2/4



Figure 6. Street Lights: 2/2 Traffic Signals: 2/5

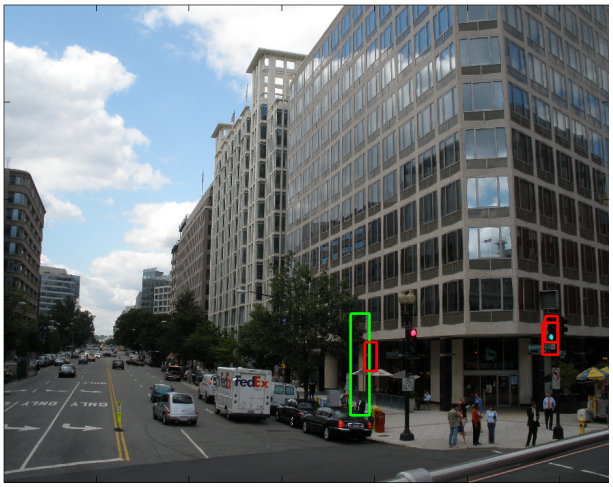


Figure 7. Street Lights: 0/1 Traffic Signals: 1/3



Figure 8. Fire Hydrants: 1/1 Street Lights: 2/3

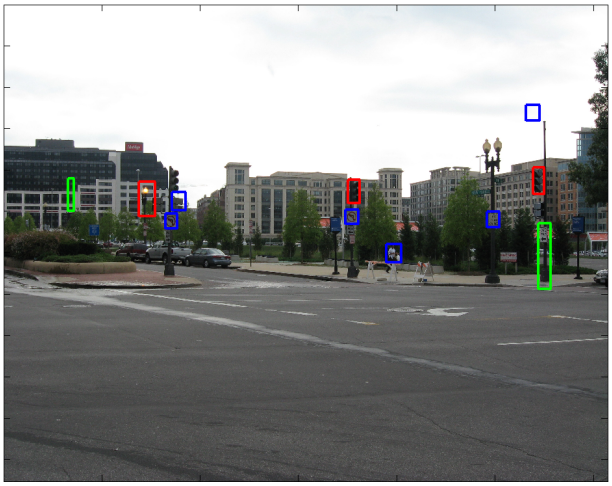


Figure 9. Street Lights: 0/2 Traffic Signals: 2/3 Traffic Signs: 3/6



Figure 10. Street Lights: 1/3 Traffic Signs: 0/2