The Aerial Parking Lot dataset: A description

Nisim Hurst

Thursday 26 July 2018

Abstract

The Aerial Parking Space dataset (*APKLOT*) is intended for two audiences: algorithm designers and researchers who want to validate the state of the art measured by performance on the Parking Space dataset.

Contents

Introduction . . . 1
Objectives . . . 2
Related work . . . 2
Ground truth description . . . 2
Included scripts guide . . . 3
Format conversion . . . 4
Subset selection - `1. build_training_test_folders` . . . 4
Pascal format conversion - `2. pascal` . . . 4
Statistical features - `4. features` . . . 4
Mask evaluation . . . 5
Dataset expansion . . . 5
Download more images . . . 5
Jittering - `3. jittering` . . . 5
Useful Software . . . 6
Google Maps Api . . . 6
labelme . . . 6
dlib . . . 6
History and Background . . . 6
Organizers . . . 6
Support . . . 7
Legal Notice . . . 7
References . . . 7

Introduction

The main goal of this dataset is to segment parking spot spaces from several parking lots of the world in realistic satellite photos. Satellite photos were generated using the free Google Maps API service. Given a training set, this is fundamentally a supervised-learning learning problem.

For the purpose of this dataset, a parking spot is a painted area specially designed inside the parking lot for parking. we are not considering the following:

- Parking spots outside the parking lot.
- Badly parked vehicles, including those parked on the traffic lane and non-parking spot (benches, gardens, etc).
- Debris or machinery in the parking spot when it is used as manner of storage facility.
- Trees in the way of the parking spot.

These are the countries and cities included in *APKLOT*:

Table 1: Countries and cities included in *APKLOT*

Country	City	# Instances
México	México	90
México	Monterrey	6
México	Guadalajara	7
United States	New York	119
United States	Los Ángeles	78
United States	Chicago	25
United States	Houston	53
Chile	Santiago	62
Spain	Madrid	23
Japan	Tokyo	40

Objectives

- Provides aerial view image dataset for parking spot segmentation, i.e. generate pixel-wise areas given the class visible at each pixel through a mask.
- Provides benchmark code for evaluating the quality of the masks.
- Enables evaluation and comparison of different methods vs the results of obtained in the companion paper.

Related work

The authors on [1] present the CARPK dataset. It is the first large-scale dataset for counting cars from flying drones. CARPK provides 89,777 instances bounding boxes from moving video in 4 distinct parking lots.

Ground truth description

The ground truth is available in two formats:

1. **Pascal VOC 2010.** The PASCAL Visual Object Classes format [2] is one of the most used for segmentation. Here is a brief description of the each folder, however if you wish to dive deeper you can refer to the Pascal VOC page:
    1. **JPEGImages.** The binary image files on JPEG compression format are stored here.
    2. **Annotations.** Contains xml Pascal VOC annotation files. The most prominent feature of these files is that they do not contain the real shape polygon, just a bounding rectangle without orientation.
    3. **ImageSets\Segmentation.** 3 text files are included here: (1) train.txt, (2) trainval.txt and (3) val.txt. Each is a list of files without extension for the train, validation and test set respectively.
    4. **SegmentationClass.** Contains the masks for training by class i.e. a class-wise color is assigned.
    5. **SegmentationObject.** Contains the masks i.e. a object-wise color is assigned. Due we are using just one class, it is worthwhile to point out that this folder contains exactly the same images as the **SegmentationClass** folder and is included for compatibility issues.

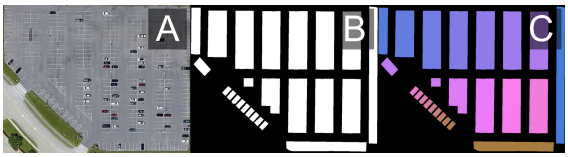

Figure 1: Example segmentation images on the Pascal VOC: (A) **JPEGImages** folder, (B) **SegmentationClass** folder, (C) **SegmentationObject** folder.

2. **LabelMe masks.** LabelMe mask format was introduced as part of a web site for image segmentation on [3]. Here are the most important elements of the ensued json file:
    1. **shapes.** An array containing each of the shape polygons that were labeled as a parking spot.
    2. **imageData.** This element was deleted on behalf of reducing redundancy and making the dataset smaller. There is one Jupyter Notebook on the **labelme** folder - *imagedata.ipynb* - which can restore or delete this dictionary element if one would like to modify some area mapping with the labelme python tool. You should restore the imageData dictionary element in order to run once again the labelme tool on the provided json file.
    3. **Other features.** lineColor, fillColor and image path is also provided for the *labelme* tool to render the polygons on edit mode.

Included scripts guide

All the scripts were written in Jupyter Notebooks with a python 3.6 kernel for readers' ease. The scripts and the folder that contains them are numbered for you to execute in order on behalf of achieving the following objectives:

1. Select the train, validation and tests sets for building the PASCAL dataset from the labelme annotations.
2. Generate each of the required folders for the PASCAL VOC 2010 format.
3. Expand the image set with jittering and corresponding annotations.
4. Extract statistical features to evaluate the datasets

```
+---1. build_training_test_folders
|   |   build.ipynb
|   \---builds
|       \---\<date\>
+---2. pascal
|   |   1. JPEGImages.ipynb
|   |   2. Annotations.ipynb
|   |   3. ImageSets.Segmentation.ipynb
|   |   4. SegmentationClass.ipynb
|   \---5. SegmentationObject.ipynb
+---3. jittering
|   |   1. Jittering.ipynb
|   |   2. Annotations.ipynb
|   |   3. ImageSets.Segmentation.ipynb
|   \---4. MoveEmpty.ipynb
\---4. features
    |   1. marked_area.ipynb
    |   2. stats features.ipynb
    \---3. evaluation.ipynb
```

Listing 1: Included scripts tree file structure

Format conversion

Subset selection - `1. build_training_test_folders`

Sometimes it is convenient to just reduce the training data for selecting the sample size or filter out some countries we want to ignore. In those cases having a script to rebuild the dataset is convenient.

That said, the purpose of this task is to allow change the training and test sets from all the images folder and two txt files that specify both training and testing filenames without extension. Thus, the input is the images folder containing both png images and json labelme annotations and two txt files list. The output is two folders containing those images. Subset selection is done in this folder, *1. build_training_test_folders*. Inside you will find *build.ipynb* notebook. You have to setup the following:

1. **training.txt** and **testing.txt** files.
2. Inside the script:
    1. **IMAGES_PATH.** Where are all the png images and labelme annotations located.
    2. **BUILD_DAY.** The name of the folder where the **training.txt** and **testing.txt** files are.
    3. **BUILD_OUTPUT.** Where the **training** and **testing** folders will be created for the new dataset.

Then, you can just run the script and the folders will be generated at the desired location.

Pascal format conversion - `2. pascal`

Statistical features - `4. features`

Finally, we want to assess how well suited is the data by itself to an specific algorithm we are developing. E.g., in some environments we would like to filter out parking spaces with very little annotated spaces to train more effectively, in others we would like to bound the algorithm in terms of computational power.

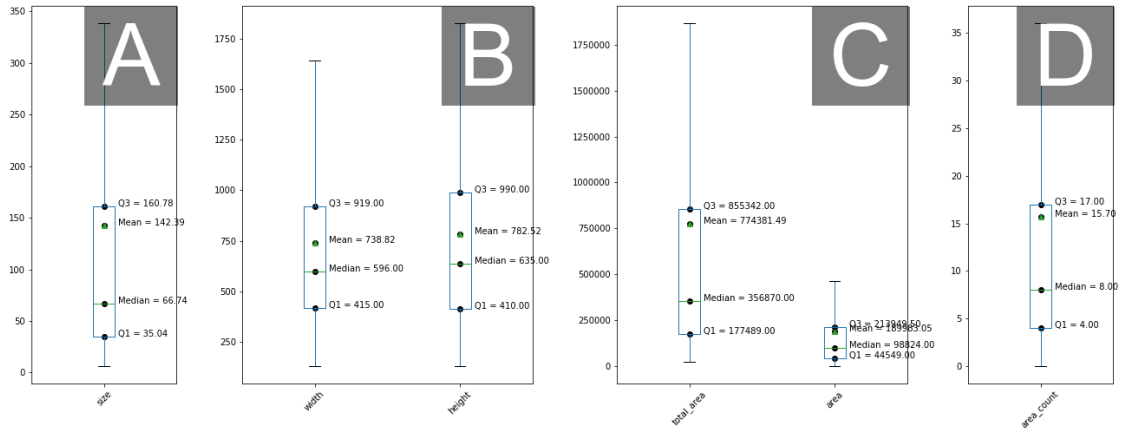

Figure 2: Stats features from the APKLOT dataset: (A) size in KB, (B) width and height of the image, (C) total area vs annotated area, (D) area count per image.

Figure 2 show some of the statistical features that were extracted by using the *features.ipynb* script:

- **(A) Size.** We can see that the interquatile range lies beneath 200KB. However, there maybe outliers reaching at most 3.4 MB. You should be careful in identifying these outliers in case your algorithm fall short of GPU memory.[1]

[1]By ignoring this warning and if you are using CUDA, a cudaMalloc error could be thrown.

- **(B) Width and Height.** We can see from theses section that most images have a quadratic proportion with more or less the same width than height. Just as the previous case, a researcher could filter out images out of proportion to cope with the required input dimensions, e.g. the input layer of a neural network.
- **(C) Total area vs Annotated area.** This plot evince that sparse annotations dominate the dataset. Subset of a larger proportion could be assembled. However, we should take into account that the total area will ever be greater than the annotated area, and that due to the image borders will be approximately $\frac{1}{4}$ of the total.
- **(D) Area count per image.** We can see that each image has marked about 15 disconnected parking spot regions. There are images in which this number is really small, meaning a big fully connected clustered region. Depending on the problem, these regions might be desirable or just as well should be avoided. E.g. If we want to boost our segmentation algorithm with the prior probability from near regions, then big clusters should be used for training. On the contrary, if we want our algorithm to be robust enough to be able to detect parking spots solely on the appearance of just one of them, then images with a large area count (disconnected spots) should be selected.

Mask evaluation

3. evaluation.ipynb

Intersection over union is an evaluation metric that is used extensively on the object detection task, specifically in the PASCAL VOC dataset. It is also known as the Jaccard index and measures the similarity pixel sets between bounding boxes. The formula for calculating the intersection over union o a bounding box is given by the following formula:

$$\frac{iTP}{(iTP + FP + iFN)}$$

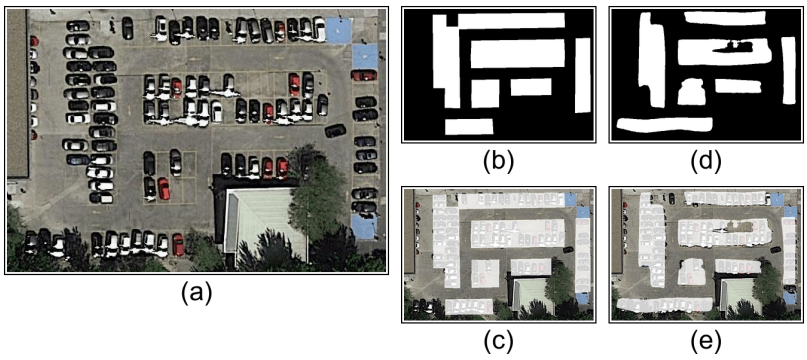

Figure 3: Example segmentation images on our dataset *APKLOT*.

Clear metrics [4].

Dataset expansion

Download more images

Jittering - `3. jittering`

Jittering was done by using the imgaug python library. The following listing shows the code that was used for jittering:

```
seq = iaa.Sequential([
        iaa.Crop(px=(0, 50)), # crop images from each side by 0 to 16px (randomly chosen)
        iaa.Fliplr(0.5), # horizontally flip 50% of the images
        iaa.Flipud(0.5),
        iaa.Affine(rotate=(-45, 45))
    ])
```

Listing 2: Jittering transformations

Lets give it a closer look and explain it line by line:

(2) Randomly crop by a value between 0 and 50 pixels
(3) Horizontally flip 50% of the images
(4) Vertically flip 50% of the images
(5) Rotate images by a value between -45 and 45 degrees

Useful Software

Google Maps Api

labelme

We already digressed around labelme but here we indulge in a more formal introduction. Labelme is a polygon manual annotation tool for segmentation that was published on [3]. It was first exposed to the public through a webpage and then through a python pip package.

dlib

Dlib is an open source library that was published on [5]. It has extensive documentation through its webpage.

History and Background

Year	Statistics	New developments	Notes
2018	Only 1 class discriminating between parking spot and other spaces. Train: 300 images 4034 labelme polygons Validation: 100 images 1513 labelme polygons Test: 101 images 1459 labelme polygons	One segmentation task: Parking Spot detection	Images were taken from Google Maps API.

Organizers

- Nisim Jonatan Hurst Tarrab (graduate student at ITESM).

- Leonardo Chang (post-doctoral researcher at ITESM).

with major contributions from

- Miguel González-Mendoza (ITESM)

Support

The preparation and running of this dataset is supported by the *Instituto Tecnológico y de Estudios Superiores de Monterrey* ITESM, and was funded up to some point by the *Consejo Nacional de Ciencia y Tecnología* CONACyT. If you wish to contribute to this dataset or have any doubt, please contact us at langheran@gmail.com.

Legal Notice

The Aerial Parking Lot dataset (APKLOT), is made available under the Creative Commons license found in the `license/` directory. A Creative Commons (CC) license is one of several public copyright licenses that enable free distribution of otherwise copyrighted work.

APKLOT consist of 500 still images with more than 7000 marked polygons of parking spots. The images were extracted from Google Maps API. Users are entitled to use this image under this conditions:

1. Comply with the *fair-use* google maps terms of service given at their page.
2. Comply with the license file on the `license/` folder.
3. Comply with best practices for attribution found here: https://wiki.creativecommons.org/wiki/best_practices_for_attribution
4. Understand that the authors make no guarantee or warranty of non-infringement with respect to the dataset.

References

[1] M. Hsieh, Y. Lin, and W. H. Hsu, "Drone-based object counting by spatially regularized regional proposal network," *CoRR*, vol. abs/1707.05972, 2017.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[3] A. Barriuso and A. Torralba, "Notes on image annotation," *CoRR*, vol. abs/1210.3448, 2012.

[4] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The clear 2006 evaluation," in *International evaluation workshop on classification of events, activities and relationships*, 2006, pp. 1–44.

[5] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, Dec. 2009.