###########################################################################

# THE KITTI VISION BENCHMARK SUITE: OBJECT BENCHMARK #

# Andreas Geiger Philip Lenz Raquel Urtasun #

# Karlsruhe Institute of Technology #

# Toyota Technological Institute at Chicago #

# www.cvlibs.net #

###########################################################################

This file describes the KITTI object detection and orientation estimation

benchmarks, consisting of 7481 training images and 7518 test images for each

task. Despite the fact that we have labeled 8 different classes, only the

classes 'Car' and 'Pedestrian' are evaluated in our benchmark, as only for

those classes enough instances for a comprehensive evaluation have been

labeled. The labeling process has been performed in two steps: First we

hired a set of annotators, to label 3D bounding boxe tracklets in point

clouds. Since for a pedestrian tracklet, a single 3D bounding box tracklet

(dimensions have been fixed) often fits badly, we additionally labeled the

left/right boundaries of each object by making use of Mechanical Turk. We

also collected labels of the object's occlusion state, and computed the

object's truncation via backprojecting a car/pedestrian model into the

image plane.

NOTE: WHEN SUBMITTING RESULTS, PLEASE STORE THEM IN THE SAME DATA FORMAT IN

WHICH THE GROUND TRUTH DATA IS PROVIDED (SEE BELOW), USING THE FILE NAMES

000000.txt 000001.txt ... CREATE A ZIP ARCHIVE OF THEM AND STORE YOUR

RESULTS (ONLY THE RESULTS OF THE TEST SET) IN ITS ROOT FOLDER.

Data Format Description

=======================

The data for training and testing can be found in the corresponding folders.

The sub-folders are structured as follows:

- image\_02/ contains the left color camera images (png)

- label\_02/ contains the left color camera label files (plain text files)

- calib/ contains the calibration for all four cameras (plain text file)

The label files contain the following information, which can be read and

written using the matlab tools (readLabels.m, writeLabels.m) provided within

this devkit. All values (numerical or strings) are separated via spaces,

each row corresponds to one object. The 15 columns represent:

#Values Name Description

----------------------------------------------------------------------------

1 type Describes the type of object: 'Car', 'Van', 'Truck',

'Pedestrian', 'Person\_sitting', 'Cyclist', 'Tram',

'Misc' or 'DontCare'

1 truncated Float from 0 (non-truncated) to 1 (truncated), where

truncated refers to the object leaving image boundaries

1 occluded Integer (0,1,2,3) indicating occlusion state:

0 = fully visible, 1 = partly occluded

2 = largely occluded, 3 = unknown

1 alpha Observation angle of object, ranging [-pi..pi]

4 bbox 2D bounding box of object in the image (0-based index):

contains left, top, right, bottom pixel coordinates

3 dimensions 3D object dimensions: height, width, length (in meters)

3 location 3D object location x,y,z in camera coordinates (in meters)

1 rotation\_y Rotation ry around Y-axis in camera coordinates [-pi..pi]

1 score Only for results: Float, indicating confidence in

detection, needed for p/r curves, higher is better.

Here, 'DontCare' labels denote regions in which objects have not been labeled,

for example because they have been too far away from the laser scanner. To

prevent such objects from being counted as false positives our evaluation

script will ignore objects detected in don't care regions of the test set.

You can use the don't care labels in the training set to avoid that your object

detector is harvesting hard negatives from those areas, in case you consider

non-object regions from the training images as negative examples.

（dontcare 是车但太小，不用检测也不要当做负类）

The coordinates in the camera coordinate system can be projected in the image

by using the 3x4 projection matrix in the calib folder, where for the left

color camera for which the images are provided, P2 must be used. The

difference between rotation\_y and alpha is, that rotation\_y is directly

given in camera coordinates, while alpha also considers the vector from the

camera center to the object center, to compute the relative orientation of

the object with respect to the camera. For example, a car which is facing

along the X-axis of the camera coordinate system corresponds to rotation\_y=0,

no matter where it is located in the X/Z plane (bird's eye view), while

alpha is zero only, when this object is located along the Z-axis of the

camera. When moving the car away from the Z-axis, the observation angle

will change.

To project a point from Velodyne coordinates into the left color image,

you can use this formula: x = P2 \* R0\_rect \* Tr\_velo\_to\_cam \* y

For the right color image: x = P3 \* R0\_rect \* Tr\_velo\_to\_cam \* y

Note: All matrices are stored row-major, i.e., the first values correspond

to the first row. R0\_rect contains a 3x3 matrix which you need to extend to

a 4x4 matrix by adding a 1 as the bottom-right element and 0's elsewhere.

Tr\_xxx is a 3x4 matrix (R|t), which you need to extend to a 4x4 matrix

in the same way!

Note, that while all this information is available for the training data,

only the data which is actually needed for the particular benchmark must

be provided to the evaluation server. However, all 15 values must be provided

at all times, with the unused ones set to their default values (=invalid) as

specified in writeLabels.m. Additionally a 16'th value must be provided

with a floating value of the score for a particular detection, where higher

indicates higher confidence in the detection. The range of your scores will

be automatically determined by our evaluation server, you don't have to

normalize it, but it should be roughly linear. If you use writeLabels.m for

writing your results, this function will take care of storing all required

data correctly.

Object Detection Benchmark

==========================

The goal in the object detection task is to train object detectors for the

classes 'Car', 'Pedestrian', and 'Cyclist'. The object detectors must

provide as output the 2D 0-based bounding box in the image using the format

specified above, as well as a detection score, indicating the confidence

in the detection. All other values must be set to their default values

(=invalid), see above. One text file per image must be provided in a zip

archive, where each file can contain many detections, depending on the

number of objects per image. In our evaluation we only evaluate detections/

objects larger than 25 pixel (height) in the image and do not count Vans as

false positives for cars or Sitting Persons as wrong positives for Pedestrians

due to their similarity in appearance. As evaluation criterion we follow

PASCAL and require the intersection-over-union of bounding boxes to be

larger than 50% for an object to be detected correctly.

Object Orientation Estimation Benchmark

=======================================

This benchmark is similar as the previous one, except that you have to

provide additionally the most likely relative object observation angle

(=alpha) for each detection. As described in our paper, our score here

considers both, the detection performance as well as the orientation

estimation performance of the algorithm jointly.

Mapping to Raw Data

===================

Note that this section is additional to the benchmark, and not required for

solving the object detection task.

In order to allow the usage of the laser point clouds, gps data, the right

camera image and the grayscale images for the TRAINING data as well, we

provide the mapping of the training set to the raw data of the KITTI dataset.

This information is saved in mapping/train\_mapping.txt and train\_rand.txt:

train\_rand.txt: Random permutation, assigning a unique index to each image

from the object detection training set. The index is 1-based.

train\_mapping.txt: Maps each unique index (= 1-based line numbers) to a zip

file of the KITTI raw data set files. Note that those files are split into

several categories on the website!

Example: Image 0 from the training set has index 7282 and maps to date

2011\_09\_28, drive 106 and frame 48. Drives and frames are 0-based.

Evaluation Code:

================

For transparency we have included the KITTI evaluation code in the

subfolder 'cpp' of this development kit. It can be compiled via:

g++ -O3 -DNDEBUG -o evaluate\_object evaluate\_object.cpp