Daily Work Report

Updated 19 Days Ago   All Users   Actions

Last Author huyong
Subscribers *None*

2017-03-13
Install Autoware with docker. Failed to run with X window

2017-02-21
Try to get the curves of track ahead.
Two ways available:

1. Edit the scr_server robot and rebuild Torcs (Require to dive in Torcs source code)
2. Read the map with python and process it .

2017-02-07
Train DDPG to learn single lane driving. Two experiments are conducted:

1. Training from nothing
2. Training with weights from previous failed case, succeed!

Seems that Fine tuning of network should better start from meaningful results. Otherwise, it will get nothing.

2017-01-20
Train DDPG to learn ACC. Set braking with OU noise (-0.1, 1.0, 0.05),

2017-01-16
Install DeepDrive. Use it to predict the distance of front vehicle.

2017-01-11
Talk with IDIRVIER about cooperation. Try to install gym_torcs on GPU server.
To do next: 1. try run torcs with ACC. run a rot car and learn to follow.

2. try to change rewards to make the result driving more stable.

2017-01-10
Give a presentation about Reinforcement Learning in Vehicle control.

2016-12-30
Create a new idl with the traffic light labeling for ucar dataset. Now we have a full set of traffic light labeling.

2016-12-19
Try to use DDPG method to train Torcs run by itself

2016-12-28
Build gym_torcs on my desktop

2016-12-27
batch normalization traffic light trained successfully:
yonghu@gpu4:/home/yonghu/data/yonghu/ucar-auto-driving-data-collection-system/perception/trafficlight/output/tlr_train_2016-12-26_004804

2016-12-19
Cropping the kitti data improves the performance of depth prediction.
However, after 10,000 steps, the depth prediction deteriorates. Early stop at 80,000 steps.

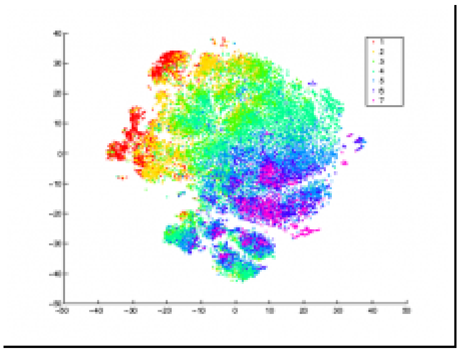
2016-12-14
Try Hungarian match without considering depth error. (/home/yonghu/data/yonghu/ucar-auto-driving-data-collection-system/perception/trafficlight/depth_regression/output/lstm_rezoom_2016_12_13_15.10)
Crop the Kitti data, instead of downsize it, so that we have depth labels for larger cars.
(/home/yonghu/data/yonghu/ucar-auto-driving-data-collection-system/perception/trafficlight/depth_regression/output/lstm_rezoom_2016_12_14_11.34)

2016-12-12
Train Tensorbox with depth, using all UCAR data (160K) and duplicated Kitti data (7500 x 20).
Results located in gpu2: /home/yonghu/data/yonghu/slim_tensorbox/output/lstm_rezoom_2016_12_08_18.41

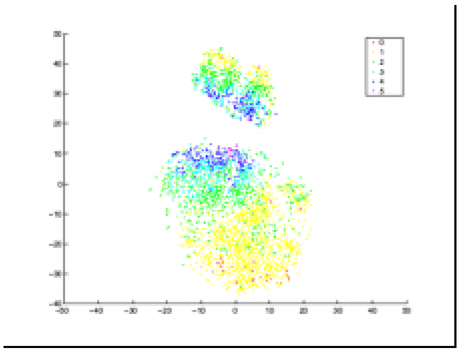
After a very long training time (300,000 steps with batchsize = 10), depth predition is fine in long distance, while terrible for veichles nearby.
Two reasons may explain: 1. kitti pictures only have uppper half, while set totally black in the lower part.

2. hungarain methods doesn't take depth into consideration, which is not defferentiable then.

2016-12-07
Visualizing the relationship between tensorbox cnn features(without training it with depth cases) and depth.

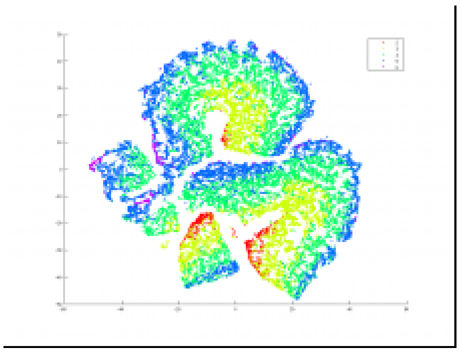


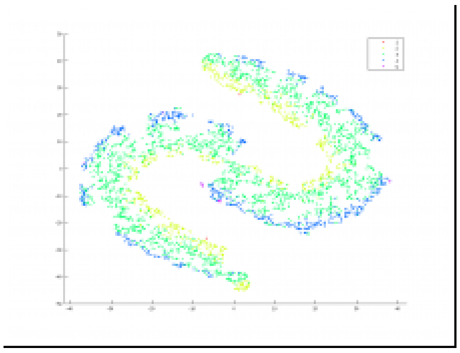
Vichle



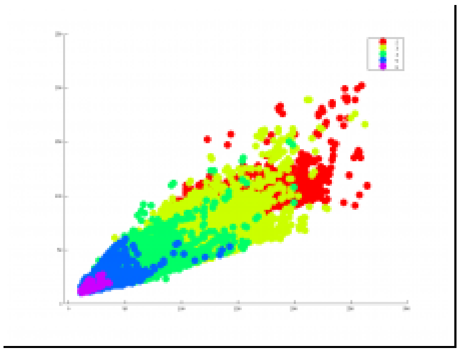
Pederstrain

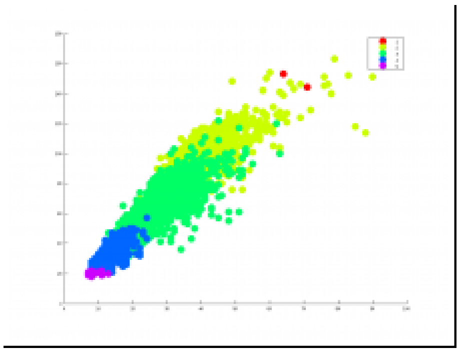
Visualizing the relationship between box informations (location and size) and depth.



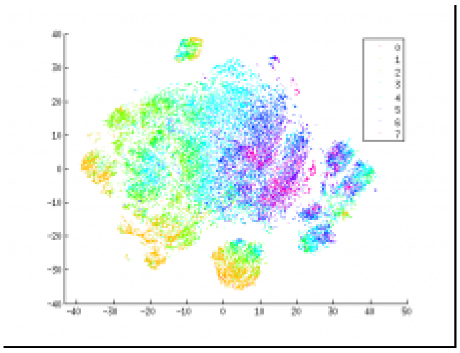


Relationship between depth and box size(width and hight).





2016-12-02
Learn on depth features.
To see the separability of these data. I remap depth to 7 classes.
[0, 5]: 0, (5, 10]: 1, (10, 15]: 2, (15, 20]:3, (20, 30]: 4, (30, 40]:5, (40, 50]:6, (50, inf):7



It shows that depths features are separable.

2016-12-01
Depth features extraction completed.

2016-11-30
Write a python library for traffic light.
Extract Tensorbox Features for Depth.

2016-11-16
Rewrite the batch generation, shuffle index instead of data.
Try to realize data prefetch threading, but failed. Two graphs required for train and test, and they have to have the same batch size.
http://ec2-52-9-34-181.us-west-1.compute.amazonaws.com/rADDC34f30b1deb54dea215a9bd1c612774101d876464

2016-11-15
Realize batch generation in traffic light trainnig process. Speedup the trainnig process.

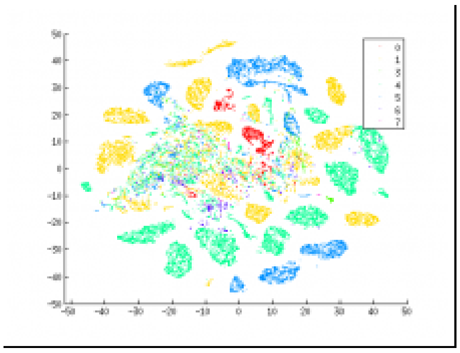
2016-11-14
Debug on restoring checkpoint files. Use tf.get_variable(name, reuse=True) to reuse prious stored variables.

2016-11-11
Refractor traffic light train code, delelte dependency on extra npy data files.

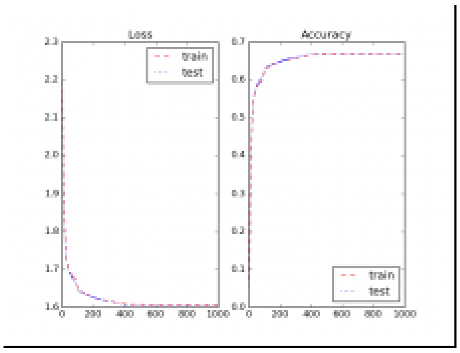
2016-11-10
Reinstall Tensorflow and Opencv.

2016-11-09
Improve the accuracy of the softmax classifier to 94%.
Add the classfier with Tensorbox detector(python version). http://ec2-52-9-34-181.us-west-1.compute.amazonaws.com/w/reports/by_person/yong_hu/traffic_light_recognition_based_on_tensorbox_cnn_features/

2016-11-07
Train softmax classifier on output features of Tensorbox.



tsne visualization shows that these cases are highly seperatable.
Until now, the best result I got is



Here a 1524x128x8 fc layers + softmax are used. Batch size is set to be the training dataset size, initializer stddev = 5.0, weights regularizer = slim.l2(0.5)

2016-11-03
Extract CNN features from Tensorbox and build a Softmax classifier.
http://ec2-52-9-34-181.us-west-1.compute.amazonaws.com/T16

2016-11-02
add Tensorbox interface to output CNN features for further TL classification.
http://ec2-52-9-34-181.us-west-1.compute.amazonaws.com/D142

2016-11-01
Add YCbCr traffic light recognition method in tensorbox.
http://ec2-52-9-34-181.us-west-1.compute.amazonaws.com/T7#116

2016-10-31
Add class_id in Tensorbox to support multiclass python detector.

2016-10-25
Try YCbCr colorspace for pixel level color classification.

2016-10-24
Pixel level color classification with rgb normalization.
rgb normalization: [r g b] --> [r g b]/(r+g+b)

Pixel color classification by model:
red [255 0 0] : R' - G' > 0.35, G'-B' < 0.08, G' <0.5
amber [255 255 0] : R-B >0.5, G'-B'>0.15, G'>0.14
green [0 255 0]: R'-G'<-0.15, G'-B'>0.17,G'>0.25
black [0 0 0]: R+G+B < 0.3
white [255 255 255]: otherwise

2016-10-20
Set up the environment (Vim and its plug-ins, tensorflow) on the new GPU server in Beijing.
Lidar calibration.
Read paper about Colorspace and Spectrum method on TLs recognition.

2016-10-19
Meet with AWS sellers, talk about the platform and resources we may need in our project.
Go with Yifeng Zhong, Yong Tian for Lidar data collection.

2016-10-18
Build a simple linear classification model to recognize TLs according to the mean RGB values in the bulbs.
Refer to the Wiki page for detail.

2016-10-17
Analysis the relationship between mean RGB values and kinds of TLs. More detail refer to Traffic Light Recognition With Image Processing.

2016-10-14
Try recognize traffic light status from TL boxes.

- get discs in image
- choose the most bright one
- determine the color in this disc

Results refer to

2016-10-13
Abadon SVM for precision (see Section 5.4 in Fast R-CNN) and complexity (http://stats.stackexchange.com/a/168073).

Things still go into two directions:

- Recognize by traditional image processing
- Recognize by learning-based method (utilize features from CNN)

2016-10-12

1. Get about 36,000 TLs boxes from Lisa dataset. More statisctics will be done soon.
2. Model based method to recogize only regular red/blue/green lights.

In regular, red/blue/green lights are located as top/middle/buttom or left/middle/right. So we would recoginize TLs in the following step:

- divide the box into three parts along the longest side
- compute the mean of RGB value in each part
- classify according the biggest mean

2016-10-11

1. View Velodyne VLP-16 and HDL-32E in the morning meeting, learn how to use HDL-32E and save Lidar data.
2. Meet with professors in North China University of Technology, talking about TLs recogition.
    - Realtime TL control data exists, but only inside the ministry of communications. An 100$ equipment would broadcast the TLs status on Internet.
    - They tried recognize TLs video recognizing method, which is proved to be unreliable.

2016-10-10

1. Clean my log data on GPU2 with a simple python script tensorbox_log_cleaner.py.

Tensorbox produces checkpoint data file every 10000 steps by default. It is reasonable in training time. While after that, a larger timestep is acceptable (such as 50000).
This script will do this for you. Take a look!

2. Try to get all TLs bulbs from Lisa dataset for TLs recogition system.

2016-10-09

Plan A recognition by detection seems not work currently for the following reasons:

- Lara and Lisa data only contains TL, while UCAR data doesn't specify differences between TLs.
- Huge bias between TLs exists in Lara and Lisa datasets.
- Many false positive exsits in Tensorbox right now.

Details of plan B (detection and recognition):

- TLs detection using existing Tensorbox
- TLs recognition with a new classification
  - prepare images contain only TLs
  - training traditional ML method
  - evaluation

2016-10-08

Find out available dataset for traffic light detection.

Dataset	#Frames	Size	#Instances	#Sub instances	PIC format	Labels
Lisa	14,386	1280x960	21,421	{goForward:0, warningLeft:291, warning:834, stop:15356, stopLeft:6987, go:14085, goLeft:865}	MPEG-2, JPEG, JSEQ, RTMaps	Filename;Annotation tag;Upper left corner X;Upper left corner Y;Lower right corner X;Lower right corner Y;Origin file;Origin frame number;Origin track;Origin track frame number
Lara	11,179	640×480	9,168	{go:3381, warning:58, red:5280, ambiguous:449}	MPEG-2, JPEG, JSEQ, RTMaps	Timestamp / frameindex x1 y1 x2 y2 id 'type' 'subtype'
Ucar	> 4565	multi-scale	*	NULL	png	FileName: (x1, y1, x2, y2):confidence/subtype[,...];

A/B plans on traffic light recognition.
recognition by detection
Feed Tensorbox with different color lights, and let it do the location and classification in the same time.
pros: simple and direct, pure DL method
cons: greedy for huge data and require labels for lights with different colors
detection and recognition
Tensorbox captures bulb locations first, then use classfication method to recognize.

pros: Tensorbox detections are good but not perfection, standalone recognition would improve detection
cons: extra classification step with reasonable cost