Documentation of ZijieMA

标签（空格分隔）： 未分类

在此输入正文

Structure of the ZijieMA

- AlexeyAB: A repo from AlexeyAB for YOLO, train faster with this
- darknet: Original YOLO, dataset MSCOCO in the `darknet/data/coco`
- Data: code for visualization. using jupyter notebook
- KITTI: dataset from kitti
- KITTI-VOC: converted kitti to Pascal VOC format, to train the neural network using kitti, the actual data used in in this directory. The label are modified. Lefted are three categories Car person and truck. If you want to train with the original label, you should modify the step-wise python file to convert everything from the original dataset.
- PascalVOC: dataset from Pascal
- PyTorch-YOLOv3: code for the system
- Trained_archiv: some trained weights and the log of the training

for the test, run those code in different window:

```
$ roscore
# ----------------------new window-----------------------
$ sudo chmod 666 /dev/ttyUSB0 # access to rplidar
# password: schaeffler2019
$ roslaunch rplidar rplidar.launch
# ----------------------new window-----------------------
$ pycharm-commmunity # must open with bash, or the rospy won't work
```

Enviroment management

Enviroment is managed using `conda`. Conda is a enviroment management toolkit. To modify current enviroment pytorch, you should always activate the enviroment:

```
source activate pytorch_yolo
```

`pytorch_yolo` is the name of the enviroment. After corrct activation bash will show as `(pytorch_yolo)$`, which means that you are in a specific environment.

install package with conda or pip

Try with `conda install *the package you want*` first. And choose yes whatever package the conda suggest to upgrade or downgrade. Conda manages everything for you. You can google to check if there is 3rd party conda installation for the package. If conda does not work, you could use the classical `pip install` routine.

manage environment in pycharm

Open the preference in the pycharm and search for project interpreter. On the right side you choose the project interpreter you already set. You can click show all and the + buttom to import new environment. Choose `Conda environment` on the left hand side and `Existing environment` on the right hand side. It should pop-up the pytorch_yolo for you. Once you choose the right environment, you can use the project interpreter page to manage your environment.

jupyter notebook

It is a powerful tool for visualization. To start, make sure you are in the pytorch_yolo environment.

```
$ jupyter notebook
```

Useful for debugging. Since the block already run will store in the memory and you can call the variable without run them every time.

System code

Code in ~/ZijieMA/PyTorch-YOLOv3 Run with pycharm: then go to the `abschluss_demo.py` in the source tree on the left side. right click and hit Run 'abschluss_demo'

easy run without open the pycharm:

```
$ source activate pytorch_yolo
$ cd ~/ZijieMA/PyTorch-YOLOv3
$ python abschluss_demo.py
```

Code you can modify are `utils/utils.py` and `abschluss_demo.py`(or you can create you own)

`utils.py` contains many important function. most important: `ransac_with_bbox` and `get_frustum_rplidar_distance` They have detailed comment to explain almost everything

For vehicle detection, you need to change the line 139-146 in `abschluss_demo.py` because the vehicle detection return more points. Since this demo is only used for visualization, you can also ignore the vector detection part and capture the returned points coordination directly.

`config` folder contains configuration of different neural networks. `weights` folder contains the trained model of different neural netwoks. config and weights You can change the line 22-27 to apply different weights and configuration. `conf_thres` in line 29 indicates the amount of object predict. Higher thresh means higher standard and fewer bounding box predicted. Lower means more bounding box but not so accurate prediction. `nms_thresh` stay with 0.45 or 0.5.

The idea of this project is first setup with a neural network using pytorch.Model(). It reads the image in and prepares the image(resizes and recenterizes the image) and feed into a neural network. The prediction of the bounding box is in `detections = model(img)` Non maximal suppression eliminates the bounding boxes that are duplicated. The prediction of the bounding boxes stops here.

After that you can process with the distance estimation or the contour prediction part. The code in util.py should work out of the box. I suggest that you create a array or torch.tensor to store only the label and contour information.

Training

For training I suggest that you use the training module from AlexeyAB/darkent since it use C and is well-optimized. For the training using MS COCO, configuration file in the `darknet/cfg/coco.data` or `AlexeyAB/darknet/cfg/coco.data` indicates the directory to each of the file, which are lists that indicate each part of the dataset. `backup` is where the backup file is stored. Training with command:

```
$ ./darknet detector train cfg/coco.data cfg/*your configuration*.cfg
darknet53-74 -dont_show -gpus 0,1 | tee *training-log*.log
```

This enable multi-GPU training and store the training log for further analysis. `AlexeyAB` and `darknet` have their own explaination of how to use them. See: https://github.com/AlexeyAB/darknet
https://github.com/pjreddie/darknet

Modification of the configuration file

`yolov3.cfg` `v390.cfg` are good starting point of a modification. If you train on other dataset, please modify at least the layer before every [yolo] block in the file. filter=255 should be change to (5+number_of_category in the dataset)*number_of_anchor_per_scale

According to my test during the master thesis, 4 scale did provides better performance. But the inference time is longer. Haven't test yet if it runs at realtime.(about 50% slower) And don't know if it's better than yolov3.

You can use the `v390-4scales.cfg` as the starting point. Anchor and filter size should not be changed. Try with different ConvNet structure. Be careful with the [route] layer. Make sure the input-output matches. 4 scales for example, the largest will be a 104*104 input. Check the first few layers.

Train with own dataset.

Build up a dataset like Pascal VOC structure. Make labels using tool `Labelimg VOC`.

Problem with the project

There are some cases the system cannot handle. I think you should know that first.

- Object that are not on the same level: our lidar is a 2D lidar, if the object does not appear in the lidar point cloud, it would be troublesome.
- Object that have overlapping objects. Especially when the object near us is a unknown object.

- When the actual object not takes up most of the area of the bounding box. Like if a person open his arm. This works bad especially when there's objects at his back.
- Camera has a small field of view. This cause problem for detection. Luckly it can detect person even without the whole body.
- LiDAR can't handle detection far away from us. To match the field of view of the camera, the point cloud can be used is limited. With maximum point number 42. If the object locates far away, only few point would fall onto the object. For person especially, at 18 meter, we can only get 1 or 2 points.