|  |  |
| --- | --- |
| **Step** | **Reference Paper** |
| Camera Calibration | <https://medusa.fit.vutbr.cz/traffic/data/papers/2014-IEEE-ITS-CameraCalibration.pdf>  <https://pdfs.semanticscholar.org/fa90/8a206be873c9de34a6997d315ea9f5c31e00.pdf> |
| Vehicle Detection | <http://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w3/Sochor_GraphFIT_Submission_to_CVPR_2018_paper.pdf> |
| Construct 3D Bounding Box | <http://www.bmva.org/bmvc/2014/files/paper013.pdf>  <http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf> (extract vehicle silhouettes)  <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.1244&rep=rep1&type=pdf>  [C++ Code to detect shadows](https://github.com/goossaert/computer-vision/tree/master/lambertain) |
| Identify vehicle dimensions | <http://www.bmva.org/bmvc/2014/files/paper013.pdf> |
| Calculate speed | <http://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w3/Sochor_GraphFIT_Submission_to_CVPR_2018_paper.pdf> |

[Vanishing Point Detection – Matlab code](https://github.com/johndpope/BMVC-Dubska) – In the file diamond\_vanish.m, we are unable to understand how mx\_raster\_space, which refers to mx\_raster\_space.cpp, returns the required space. Steps to execute this code are below:

1. git clone <https://github.com/johndpope/BMVC-Dubska>
2. gopalg@aw10-158-159:~$ sudo apt-get update
3. gopalg@aw10-158-159:~$ sudo apt-get install octave
4. $sudo apt-get install octave-image
5. $sudo apt-get install octave-miscellaneous
6. $ sudo apt-get install liboctave-dev
7. gopalg@aw10-158-159:~$ octave --no-gui
8. Xlib: extension "XInputExtension" missing on display ":1".
9. pkg load image
10. pkg load miscellaneous
11. cd BMVC-Dubska/
12. ls

diamond\_vanish.m mx\_lines.mexmaci64 README.md YorkUrbanDB

mx\_lines.cpp mx\_raster\_space.cpp results

mx\_lines.log mx\_raster\_space.mexmaci64 run\_on\_dataset.m

1. mex mx\_lines.cpp
2. mex mx\_raster\_space.cpp
3. cd YorkUrbanDB/
4. load Manhattan\_Image\_DB\_Names.mat
5. load cameraParameters.mat
6. load ECCV\_TrainingAndTestImageNumbers.mat
7. run\_on\_dataset(Manhattan\_Image\_DB\_Names, 'YorkUrbanDB')
8. whos

Variables in the current scope:

Attr Name Size Bytes Class

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

Manhattan\_Image\_DB\_Names 102x1 918 cell

ans 1x1 153 struct

focal 1x1 8 double

pixelSize 1x1 8 double

pp 1x2 16 double

testSetIndex 51x1 408 double

trainingSetIndex 51x1 408 double

Total is 209 elements using 1919 bytes

1. python puremain.py

[Vanishing Point Detection – Python code](https://github.com/johndpope/optimization) – This is the same code as above but in Python. The same code about rasterize is available in [puremain.py](https://github.com/johndpope/optimization/blob/master/puremain.py). We were able to execute this code but got stuck again in correlating rasterize\_lines and find\_maximum functions.