**OpenAI LineFollower SmartSystems**

**Software install**

Visualstudio.microsoft.com/downloads, Community downloader

* MSVC v142 – VS 2019 C++ x64/x86 build tools (v14.23 or v14.24) OR MSVC v140 VS 2015 C++ build tools (v14.00)
* Windows 10SDK (10.0.16299.0)
* Windows universal CRT SDK

**Python packages**

To check library list “pip freeze”

* clone gym\_line\_follower from github -> “Pip install –e .”
* Tensorflow requires version 1.14.0, “pip install tensorflow==1.14.0”
* conda install -c powerai gym & pip install pybullet
* pip install git+https://github.com/benelot/pybullet-gym

**Running errors**

* 'cannot re-register id: Linefollower-v0', comment code in \_init\_.py in dir "\gym\_line\_follower"

**LineFollower\_Env**

* If error " cannot re-register id: Linefollower-V0" then GOTO LINE 106 in 'Registration.py' and change = > (env, "\_gym\_disable\_underscore\_compat", **True**):

on LINE 146 add '**del self.env\_specs[id]**' OR pip uninstall gym\_line\_follower (weird fix)

* step

**LineFollower\_Bot**

**LineFollower\_Camera**

**Track generation**

Some parameters are still hardcoded in the follower\_bot.urdf file.

Changing the track that is loaded is done in Track.py:

- In func "generate" the track 'file' gets made and returned as pts.

-line 98 bezier func. comment this line to make the track points given in the Generate() drawn in order.

-line72 in getcurve() in the for change length points from -1 to -2.

-Finally In line\_followeR\_env.py line 135-150 defines random positions and track. Change these accordingly to use with a fixed track (mcu) defined with the above steps.

**Training a model**

Changing the track to the MCU is done with previous chapter. When the track is changed we can train a model with the ddpg.py. The default linefollower gets imported as gym enviroment, a network is created with the build\_agent().

the keras-rl library has different agents to choose from. the ddpg agent is chosen and imported as DPPGAgent.

With the DPPGAgent we can send the agent an enviroment and it will look the at the actions available. It can execute and train with the rewards gotten from the observation of the Line\_env. The train func() calls the build\_agent() which creates the network for our model. In this network the following is build:

**Actor network**

-

**Critic network**

**-**

The agents get compiled with the DPPGAgent() and with parameters for the amount of actions, the 2 networks, how to react on a result, the learning rate ... AANVULLEN.

**Loading a saved pretrained model**

Change the pretrained\_path variable to the dir folder of the models trained. In train function add "agent.load\_weights(os.path.join(pretrained\_path,"weights"))". 'Weights' is the name of your file with weights to be loaded.