Guidance for Autonomous Vehicle Development Environment Setup

Preface:

The installation of GPU enabled development environments for autonomous vehicles (AV) really troubles us for a long time at the very beginning of AV research. There are a few resources spread over open communities (such as Github, Tensorflow, stackoverflow) can be referred to. Checking the solutions for the facing problems through the sparse resources really cost a lot of time, and sometimes you even cannot find a feasible solution to your problem after spending several hours, which is really annoyed. Thus, after successfully building the AV development environment, we decided to publish our installation notes to the following AV researchers. We hope this guidance can help you build the AV development environment smoothly. There is a file called "installation website" together with this file, which provides necessary website addresses for the environment installation.

Also, this file includes the errors we met, and the solutions for the corresponding errors are addressed.

This file will be updating.

CATS Tom 6/9/2020

Updated on 06/09/2020 by Tom and Zhaohui
Updated the Carma installation and troubleshooting on 06/10/2020 by Tom and Zhaohui
Updated on 06/14/2020 by Zhaohui
Updated on 1/15/2021 by Zhaohui
Updated on 4/8/2021 by Zhaohui Fixed the GUI glitch question

Table of Contents

1. NVIDIA CUDA Toolkit and Graphics Card Driver Installation

To install autoware with GPU enabled, CUDA toolkit and graphics card driver need to be preinstalled. Due to project requirements, both CUDA toolkit 9.0 and 10.0 with NVIDIA driver 440.82 at Ubuntu 16.04 and CUDA 10.0 with NVIDIA driver 440.82 at Ubuntu 18.04 are installed successfully with the following steps.

1. Block the original nouveau driver and Reboot into text mode.

Use the following oreder to block the nouveau:

sudo nano /etc/modprobe.d/blacklist-nouveau.conf

add following sentences to the conf file:

blacklist nouveau

options nouveau modeset=0

Ctrl+X to save the file and press enter to exit

Reload the configuration with the following sentence.

sudo update-initramfs -u

Reboot the system. Then using following sentence to check whether the nouveau driver is blocked.

lsmod | grep nouveau # no output indicates success.

Then we are ready to install the CUDA toolkit and NVIDIA driver which can be download from the NVIDIA official website. You can search them or use the link we provided at the "Useful website" file.

Use the following order to boot into text mode

Sudo systemctl set-default multi-user.target

Reboot the system. Then you cannot enter the normal mode of the system. You suppose to use your account and password to login the system (For some systems you need to use CTRL+ALT+F1 to see the terminal).

Use the following code to install NVIDIA driver (replace contents in <>). The default download location is at Download file.

cd <where your run file placed>

sudo sh ./NVIDIA-Linux-x86_64-440.82 #you may want to check the driver you needed fro m NVIDIA official website. Our card is RTX 2080 and thus 440.82 is selected.

Use the following code to install CUDA9.0 (replace contents in <>). Press space key until the end of the installation. Don't install acceleration drivers since you already installed it. Enter yes for the X configuration and link symbols.

cd <where your run file placed>

sudo sh ./cuda_9.0.176_384.81_linux.run #if after the installation the system dies at the log in in. You can add "--no-opengl-files" at the end of installation command

After installation of the CUDA toolkit, use the following command to boot into graphical mode

Sudo systemctl set-defualt graphical.target

Reboot

Then you need to add add CUDA path to ~/.bashrc,

gedit ~/.bashrc

add

export PATH=/usr/local/cuda-9.0/bin${PATH:+:${PATH}}

export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64

${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}

Save and exit

Use *source ~/.bashrc* to source the path and Library

use *$PATH* and *$ LD_LIBRARY_PATH* to check path and library.

Use *nvcc -V* and *nvidia-smi* in a terminal to check if it installed correctly.

Now the installation of CUDA toolkit and graphics driver are completed. If you want to uninstall NVIDIA driver:

sudo apt-get remove –purge nvidia*

2. Qt Update

For unbuntu 16.04, open a new terminal enter the following order

sudo add-apt-repository ppa:beineri/opt-qt-5.11.1-xenial

sudo apt update

sudo apt install qt511-meta-full

use the following order to creat file named “default.conf” in /etc/xdg/qtchooser

sudo nano /etc/xdg/qtchooser/default.conf

add

/opt/qt511/bin

/opt/qt511/lib

CTRL+X to save the file and enter to exit.

Use *qmake -v* to check if it installed correctly

For unbuntu 18.04, open a new terminal enter the following order

Sudo apt-get install build-essential

Sudo apt-get install qtcreator

Sudo apt-get install qt5-default

Use *qmake -v* to check if it installed correctly

3. Installation of ROS

Follow the ros wiki instruction in

http://wiki.ros.org/kinetic/Installation/Ubuntu

make sure your ROS setup.bash is the last line of ~/.bashrc or the CUDA may fail

Use *rosversion –d* to check ros installation

Note that the ROS versions for Ubuntu 16.04 and 18.05 are different. Pick up the one fitting your system to install.

4. Installation of Autoware

Follow the autoware installation instruction in

https://github.com/Autoware-AI/autoware.ai/wiki/Source-Build

By using the version and installation check command to make sure that you satisfy the corresponding requirements for Autoware installation.

Compile the workspace with CUDA support.

5. Installation of CARMA

Follow the CARMA installation instruction in

https://usdot-carma.atlassian.net/wiki/spaces/CAR/pages/486178827/Development+Environment+Setup

Some things you need to know during installation:

Setting User ID. The CARMA platform image should share the same user and group ids. This can be achieved by the following command. (substitute contents in <>)

sudo usermod –uid 1000 <your root user name>

sudo groupmod –gid 1000 <your root user name>

Follow the official guide to setup you SSH connection instead of the souce code provided by CARMA and remember to check your SSH connectivity.

https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh

on the final step “Setup CARMA Config”

you should also type the following command before you start it:

carma config set usdotfhwastoldev/carma-config:develop-development

after ***carma start all*** you can then open a browser and type ***localhost*** in the address bar, you will be able to see the web UI.

you can use ***carma help*** to see its’ usage.

6. Installation of Spinnaker-2.0.0.109-amd64

Follow readme Section 1.2.1, Section 2 and Section 4 to install it.

Once the spinnaker is built. The Autoware should be rebuilt to compile the driver for the spinnaker.

After the building of the Autoware, source the ‘setup.bash’, run roslaunch autoware_pointgrey_drivers spinnaker.launch

Then the camera is opened. Otherwise the Autoware didn’t compile the driver for camera.

The camera driver location of Autoware is at /src/drivers/awf_drivers/pointgrey. Check Cmakelist for it, delete the space in find_package function. Then the Autoware is compiled successfully with the camera driver.

7. Trouble shooting

1. If Autoware compile fails at ndt_cpu, ndt_gpu pkg, the previous CUDA must be wrong. Reinstall the CUDA toolkit and driver can solve this problem.
2. if the compile fails in autoware_health_checker. Reinstall the Nvidia driver as previous step, it should work.
3. No libcublas.so.9.0 is found.
   Solution: need to add PATH environment parameter and Library to ./bashrc file
4. Darknet compile problem.
   Solution: change the architecture based on your graphics card.

For RTX 2080 it is sm_75.

-gencode arch=compute_75,code=[sm_75,compute_75]

5. Lose internet after install Docker.

   Reason:Docker will create a network bridge which will use IP 172.17.1.0, it may be contradict to the host machine's IP address.

   Solution: delete docker0 configuration in your Network Connection.

   Modify /etc/default/docker by ***sudo nano /etc/default/docker***

   Adds ***DOCKER_OPTS="--bip=192.168.7.1/24"***, then save and exit.

   Modify /etc/systemd/system/docker.service by ***sudo nano /etc/systemd/system/docker.service*** (if it doesn't exist, check /lib/systemd/system/docker.service by ***sudo nano /lib/systemd/system/docker.service***)

   Adds the following content

   EnvironmentFile=-/etc/default/docker #(after [services])

   Modify ***ExecStart=/usr/bin/dockerd -H fd:// $DOCKER_OPTS***

   Then save and exit, reboot, you should be able to reconnect your network.

6. In Ubuntu18.04 and Autoware.AI 12.0, you may encounter ghosting and glitch when open GUI.

   This question is not related to wxpython. **DON'T** follow any instruction of updating the wxpython.

   Follow the instruction here [Fix Runtime Manager GUI bug in ubuntu18.04 + ROS melodic (!25) · Merge requests · Autoware Foundation / MovedToGitHub / utilities · GitLab]