Individual Lab Report #7

Feb 2, 2017

Menghan Zhang

TeamA

Amit Agarwal

Harry Golash

Yihao Qian

Zihao Zhang

Individual progress

For the beginning of last week, our team spend lots of time on what we should to do and the discuss the specific detail of the project to give us direction of this semester and then, since we know that what need to be done so that we made the detailed schedule to make sure that will work. This time , my work can divided into 2 part

1. Get familiar with CUDA

The installation process is basically follow the instruction of the NVIDIA website, but there are several things that need to be noticed.

a) Stop the nouveau driver before install NVIDIA driver

This is really important, because at the first time I tried to install without stop this driver, it turns out that at the end of the installation, it can not successfully installed because the X is running.

```
$ lsmod | grep nouveau
```

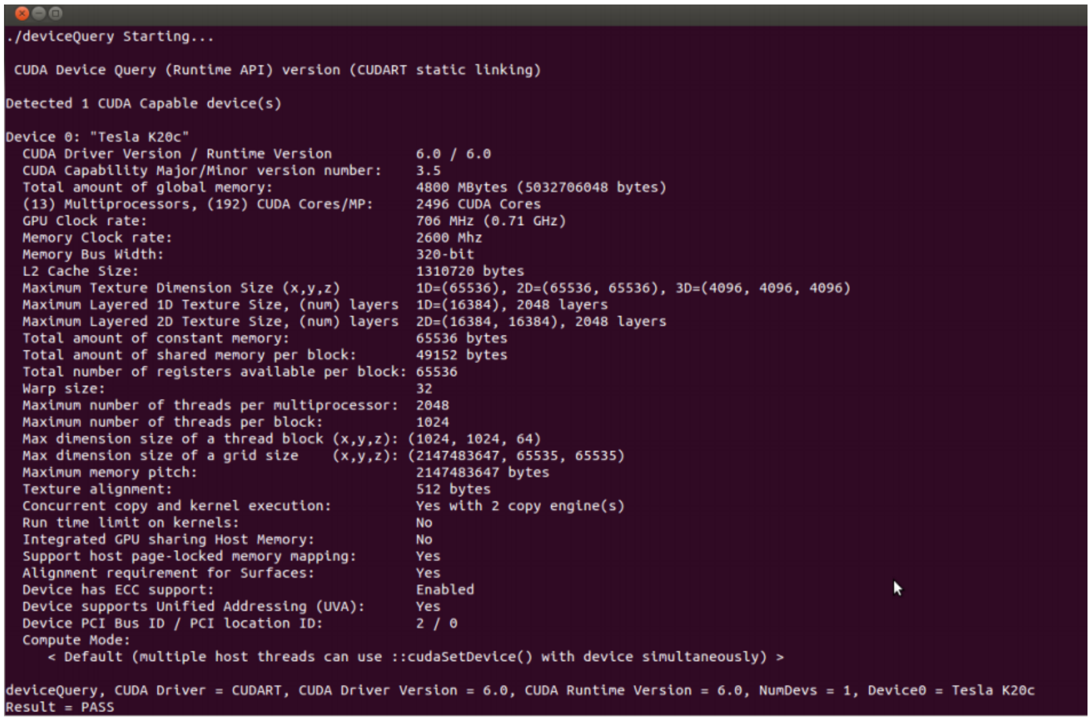
Use this command to show that nouveau is working, I tried several method to move this driver in the blacklist, but it doesn't work, and magical things happened, it was successfully blocked after a night, so I guess restart the computer can not refresh the change made immediately.

b) The most important things is not install OpenGL libraries during the installation. Because my computer have 2 GPU and Intel one is used to display, so if you choose to install OpenGL libraries, it will loop login interface again and again so I have to reinstall the Ubuntu to fix that.

c) device node verification

Go to the /dev catalog to check the NVIDIA file is appear or not, it turns out they don't exist. Because my computer system disallows setuid binaries, so that the device nodes can not generated automatically. So I had to create them manually by writing the startup script in the etc/init.d catalog and.

Finally test the cuda using the example given, it works, the result is shown in figure 1.



```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla K20c"
  CUDA Driver Version / Runtime Version          6.0 / 6.0
  CUDA Capability Major/Minor version number:    3.5
  Total amount of global memory:                 4800 MBytes (5032706048 bytes)
  (13) Multiprocessors, (192) CUDA Cores/MP:     2496 CUDA Cores
  GPU Clock rate:                                706 MHz (0.71 GHz)
  Memory Clock rate:                             2600 Mhz
  Memory Bus Width:                              320-bit
  L2 Cache Size:                                 1310720 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Bus ID / PCI location ID:           2 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 6.0, CUDA Runtime Version = 6.0, NumDevs = 1, Device0 = Tesla K20c
Result = PASS
```

Figure 1

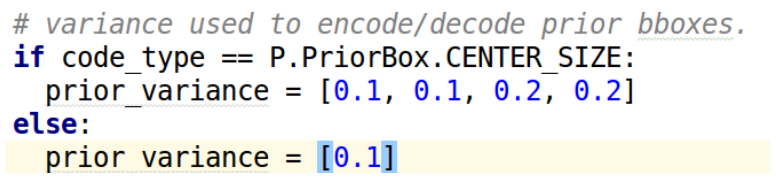
2. Extract label and bounding box location from detection algorithm

To integrate object detection algorithm with stereo vision and tracking algorithm in the future.We want to apply the algorithm in the videos recorded by both camera and the detection algorithm will give the labels and bounding boxes information of the objects, then use this region proposal, we can calculate the features in the bounding boxes and make the correlation between two cameras. Since our cameras are synchronized , we will be able to know that these two areas in two cameras' images is the same object at this time, so that we can go to the stereo vision to calculate the depth information of the objects. Same to the tracking, the input of the tracking algorithm is the image and location of the objects so that it can keep tracking this object. In that case, the first thing we should do is to extract the output from the detection algorithm.

I go through the source code of the ssd, I find out the structure of the code is shown below

a) load the basic model (VGG-16)
b) Add extra layers on top of a "base" network
c) set the parameters used in traning
d)
e) set the parameters in output
f) check file
g) create the test net
h) create the job file
i) run the job

I think they key point is in this part of the code and need to look up other related files to find exactly the bounding boxes information located.



```
# variance used to encode/decode prior bboxes.
if code_type == P.PriorBox.CENTER_SIZE:
  prior_variance = [0.1, 0.1, 0.2, 0.2]
else:
  prior variance = [0.1]
```

Challenge

As I mentioned before, the challenge is to find the exact location of the bounding box generated in tons of files in Caffe model and scripts and then save that information. I need to put more time on it and dig more deeper to find it

Also CUDA has too many reliable environment and if you make a small mistake you need to reinstall whole system so it take me lots of time to do that and next time maybe find more information online before tried to install that.

Teamwork

This time everybody worked on their own part, we had several meetings for the whole project goals and I had the brief discussion with Yihao about the information he need for the stereo vision. Zihao built the subscriber in Polysync, Yihao tried to transplant the stereo vision to C++ using OpenCV, Harry got familiar with Polysync and repaired car, Amit did some research on tracking.

Plan

For next week, I will continue to extract the information from object dectetion algorithm and then try to fine-tuning the model and improve the detection and classification performance.