**github:vincent0628**

**------------------------------------------------------------------------------------------------------------------------**

[ initial ]

**ssh nvidia@tx2-24.local**

**vncviewer -quality 10 -encodings "tight" tx2-24.local:5901**

**------------------------------------------------------------------------------------------------------------------------**

[ Git tutorial ]

1. **基本指令**

$ apt-get install git //在local 安裝git

$ ls -la // -l 參數為列出詳細資料，-a 為列出隱藏資料夾 (可忽略)

$ git status // 查看狀態 (可忽略)

$ git diff // 比較現在檔案和上次 commit 之間的差異，也就是說你做了哪些修改

讓 Git 知道這台電腦做的修改要連結到哪一個使用者 :

$ git config --global user.name "<Your Name>"

$ git config --global user.email "<your@gmail.com>"

1. **How to push?->到要上傳的資料外面[cd..[目標自料外]]**

$ git init

$ git add [目標物]

$ git commit -m " [say something]"

$ git commit -a -m "修改了 ????"

// 上兩行可以簡化成這行-a 是 add，-m 為 message 簡寫，後面接訊息資訊）

$ git remote add origin [remote 網址] // 本地端專案知道 origin 對應到遠端網址

$ git push -u origin master

**-----------------------------------------------------------------------------------------------------------------------------------**

[ docker build ]

**可以的 docker build -t="labmini" --no-cache -f=Dockerfile .**

docker build -t sis\_competition -f Dockerfile .

**docker run -it --rm --name** sis\_competition **--net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb** sis\_competition **bash**

rosp[ docker tear ]

**https://stackoverflow.com/questions/22049212/copying-files-from-docker-container-to-host/26150548**

**------------------------------------------------------------------------------------------------------------------------**

[ docker run ]

* **有BASH 會進去container, 可以改 但出來會全部消失**

**tx2 $ docker run -it [--rm] --name [name] --net host --privileged -v /dev/bus/usb:/dev/bus/usb [dockerhub account]/sis\_competition:[task\_name] bash**

* **沒BASH 直接roslaunch**

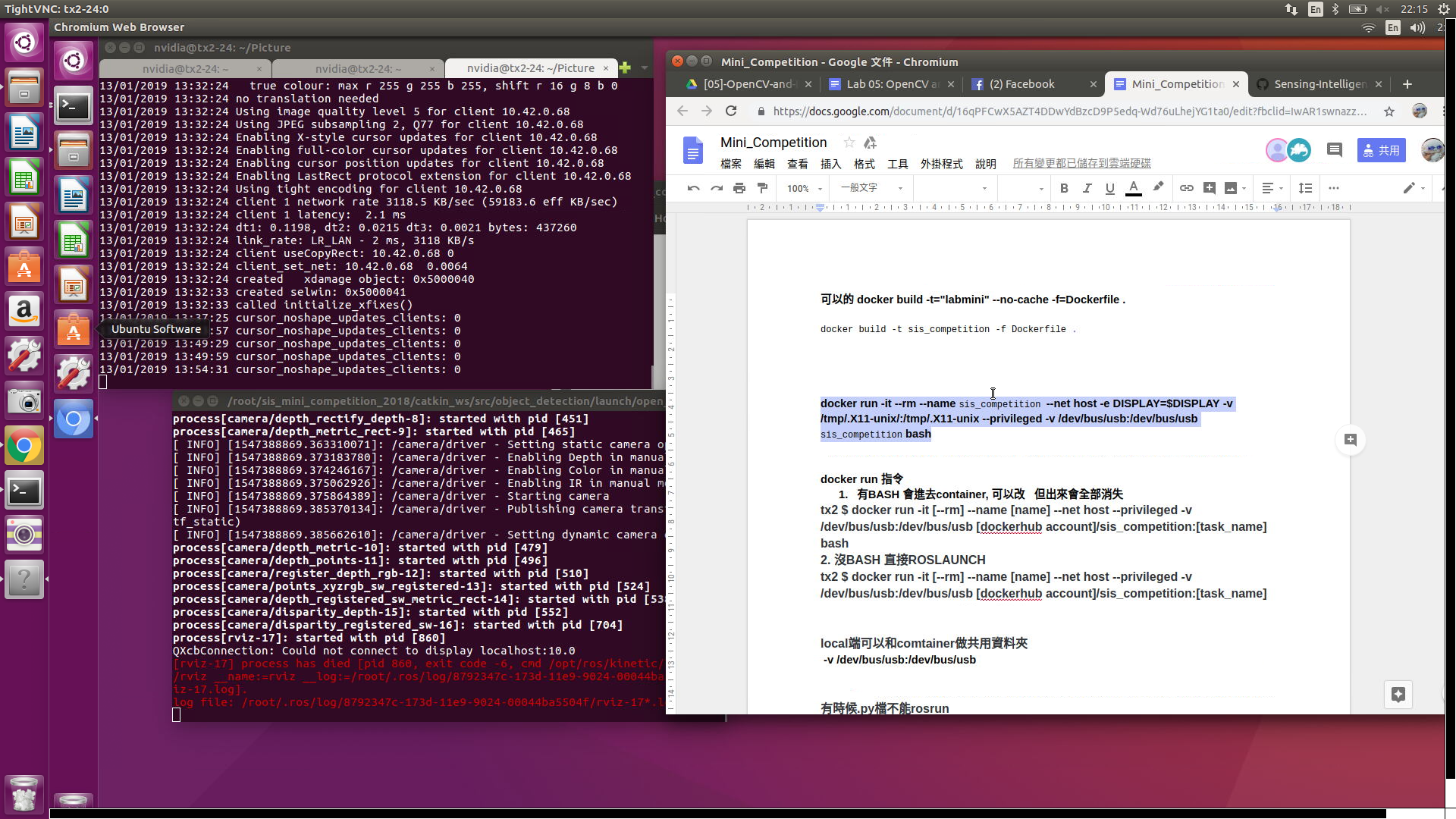
**tx2 $ docker run -it [--rm] --name [name] --net host --privileged -v /dev/bus/usb:/dev/bus/usb [dockerhub account]/sis\_competition:[task\_name]**

**有時候.py檔不能rosrun soluntion:變執行檔**

**chmod +x object\_detection.py**

**------------------------------------------------------------------------------------------------------------------------**

**1/13問題**

****

**用lab5的方法可以build docker 和run**

**但是**

**用助教的source方法不能開rviz**

**lab5**

**docker run -it --rm --name lab5 --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -w /home/nvidia/sis\_lab\_all/05-Opencv\_and\_Depth\_Sensing/catkin\_ws/ lab5 bash**

**@@@@@助教的@@@@@@**

**tx2 $ docker run -it --rm --name** sis\_competition **--net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb vincent0628/sis\_competition:object\_detection bash**

**chmod +x object\_detection.py**

**source devel/setup.bash**

**先catkin\_make再 source**

[Dockerfile](https://github.com/Sensing-Intelligent-System/sis_competition_task_template/blob/master/Dockerfile) **→ run\_task.sh →** launch sis\_arm\_planning master\_task.launch → 這邊應該要

**<include file="$(find object\_detection)/launch/open\_realsense\_and\_vis.launch"/>**

**image = windows**

**containner = computer**

**------------------------------------------------------------------------------------------------------------------------**

[ docker build S.O.P ]

**source docker\_build.sh**

**docker tag sis\_competition vincent0628/sis\_competition:labmini12**

**docker push vincent0628/sis\_competition:labmini12**

**------------------------------------------------------------------------------------------------------------------------**

[docker build in runnung docker ]

1. **go out to tx2 &&docker ps**
2. **docker commit [Container ID] [vincent0628/sis\_competition:Tag]**
3. **docker push vincent0628/sis\_competition:Tag**

**------------------------------------------------------------------------------------------------------------------------**

[ sync and docker run for debug]

**nvidia@tx2-24:~/sis\_mini\_competition\_2018$**

**docker run -it --name sis\_competition --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/nvidia/sis\_mini\_competition\_2018/sync:/root/sis\_mini\_competition\_2018/sync vincent0628/sis\_competition:task2done bash**

**local端可以和comtainer做共用資料夾 -v /dev/bus/usb:/dev/bus/usb**

**------------------------------------------------------------------------------------------------------------------------**

[手臂 pi3 SOP ]

**TX2$ docker run -it --name sis\_competition --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/nvidia/sis\_mini\_competition\_2018/sync:/root/sis\_mini\_competition\_2018/sync vincent0628/sis\_competition:task2done bash**

**[TX2 docker$** roslaunch pose\_estimation\_and\_pick moveit\_tutorial.launch **] //move it 打開**

**TX2$ ssh ubuntu@mm\_pi**

**TX2 docker$ roscore**

**pi$ cd sis\_mini\_competition\_2018/catkin\_ws && source devel/setup.bash**

**pi$ roslaunch sis\_mobile navigation\_pi.launch car\_id:=[13] # Make sure you have start ROS master in TX2**

**TX2$ docker ps docker exec -it [Container ID] bash**

**TX2$**

**TX2 docker $ source run\_task.sh**

[vim .py SOP ]

**TX2 docker $ cd catkin\_ws/src/moveit\_example/src/**

**TX2 docker $ cd ../../../**

**TX2 docker $ catkin\_make**

**TX2 docker $ cd ~/sis\_mini\_competition\_2018 && source run\_task.sh**

**------------------------------------------------------------------------------------------------------------------------**

[navi pi3 SOP ]

1.

**TX2 $ docker run --rm -it --net=host --name=navigation vincent0628/sis\_competition:task3done bash**

**TX2 docker $ source catkin\_ws/devel/setup.sh**

**TX2 docker $ roslaunch robot\_navigation robot\_navigation.launch**

**2.**

**TX2 $ docker exec -it navigation bash**

**TX2 docker $ source catkin\_ws/devel/setup.bash**

**TX2 docker $ roslaunch sis\_mobile tag\_detection.launch**

**3.**Then launch Sean’s navigation\_pi.launch with these parameter in second terminal

**TX2 $ ssh ubuntu@mm\_pi (passwd : ubuntu)**

**PI3 $ source ~/.bashrc**

**PI3 $ source sis\_mini\_competition\_2018/catkin\_ws/devel/setup.bash**

**PI3 $ roslaunch sis\_mobile navigation\_pi.launch x:=1 y:=0.9 th:=0 car\_id:=13 [mecanum:=true]**

4.

**TX2 $ docker exec -it navigation bash**

**TX2 docker $ source catkin\_ws/devel/setup.sh**

**TX2 docker $ rosrun robot\_navigation demo.py**

**------------------------------------------------------------------------------------------------------------------------**

[ rosbag ]

1. **roslaunc**h ⇒ publish all topics
2. In another terminal, go to container (**docker exec -it [ID] bash**)
3. **rosbag record [topic1] [topic2] [...]**
4. **ctrl C** to stop the record

How to use rosbag

1. roslaunch
2. just rosbag play [rosbag name] -l to replace camera

**object\_detection.py topic:**

/camera/rgb/camera\_info

/camera/rgb/image\_rect\_color

/camera/depth\_registered/sw\_registered/image\_rect

/camera\_rgb\_optical\_frame

/object\_detection/mask\_eroded

/object\_detection/mask\_eroded\_dilated

/object\_detection/img\_result

##

rosbag record /camera/rgb/camera\_info /camera/rgb/image\_rect\_color /camera/depth\_registered/sw\_registered/image\_rect /camera\_rgb\_optical\_frame /object\_detection/mask\_eroded /object\_detection/mask\_eroded\_dilated /object\_detection/img\_result

**------------------------------------------------------------------------------------------------------------------------**

[ color detection of object\_detection.py ]

**lower\_red=[170,50,50]**

**upper\_red=[180,255,255]**

**lower\_blue=[95,100,10]**

**upper\_blue=[120,255,255]**

**lower\_green=[65,50,50]**

**upper\_green=[85,255,255]**

**popo理解：**

**可以先做好**

**chmod +x→ docker\_build.sh**

**→ Dockfile(在包成image中已做好catkin make & source)**

**→ docker run (執行CMD， run\_task.sh)**

**→ launch master\_task.launch**

**→ 利用master\_task.launch launch我們要的task**

**之前把指令放在docker\_build.sh會catkin make失敗的原因：因為已先build完才執行catkin make 等同於在本機端執行並非docker內，而本機內未安裝ROS所以失敗。放在run\_task.sh可以的原因是我們會先docker run 才會進入run\_task.sh，這等同於在docker內執行catkin make，故成功。註：在Dockerfile中其實已經幫我們catkin make了**

**------------------------------------------------------------------------------------------------------------------------**

**########################################################################**

[柏辰借用區]

**source docker\_build\_test.sh**

**docker run -it --rm --name sis\_competition --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/nvidia/sis\_mini\_competition\_2018/sync:/root/sis\_mini\_competition\_2018/sync labyoyo bash**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**@@@@@@@@@@@ Remember to add @@@@@@@@@@@@@@@**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**docker\_build.sh**

**need to change the dir**

**/root/.torch/models/**

**srv ? task1out**

**docker run -it --rm --name lab14 --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -v ~/sis\_lab\_all:/home/nvidia/sis\_lab\_all -v /home/nvidia/sis\_mini\_competition\_2018/sync:/root/sis\_mini\_competition\_2018/sync -w /home/nvidia/sis\_lab\_all/14-PCL\_ICP/catkin\_ws/ peterx7803/ctsp-00-ros bash**

**docker run -it --name sis\_competition --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/nvidia/sis\_mini\_competition\_2018/sync:/root/sis\_mini\_competition\_2018/sync vincent0628/sis\_competition:task2\_bridge\_8 bash**

**######################### Task1 ###############################**

**ssh nvidia@tx2-**

**export DISPLAY=:0 && xrandr --fb 1800x900**

**x11vnc**

**ssh nvidia@tx2-**

**vncviewer -quality 10 -encodings "tight" tx2.local:**

**xhost +**

**######################################**

**docker run -it --name sis\_competition --net host -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix --privileged -v /dev/bus/usb:/dev/bus/usb vincent0628/sis\_competition:task2\_bridge\_7 bash**

**cd catkin\_make && source devel/setup.sh**

**roslanuch ob**

**[another]**

**docker exec**

**cd catkin\_make && source devel/setup.sh**

**rosrun**

**/camera/rgb/image\_rect\_color**”, “**/object\_detection/mask\_eroded**”, “**/object\_detection/mask\_eroded\_dilated**”, “**/object\_detection/img\_result**

**1-1**

**source run\_task.sh**

**/camera/rgb/image\_rect\_color**”, “**/object\_detection/mask\_eroded**”, “**/object\_detection/mask\_eroded\_dilated**”, “**/object\_detection/img\_result**”

**1-2**

**[another term]**

**cd catkin\_ws/src/object\_detection/src && python task1\_2\_pub.py**

**########################################################################**

**如遇到空間滿**

**docker system prune (notice:把所有沒再使用的docker刪除 包含暫存)**

**df -h 查看空間使用**

[**https://drive.google.com/drive/u/0/folders/1ENMUKBpGmBVuJXdtIRNjvoq1-kMj50BU**](https://drive.google.com/drive/u/0/folders/1ENMUKBpGmBVuJXdtIRNjvoq1-kMj50BU)

Select frame id as “camera\_rgb\_optical\_frame”

Subscribe “/camera/depth\_registered/points” topic (type: Polint cloud2)

Subscribe “/bounding\_box” (type: Marker)

Subscribe “/product\_pose” topic (type: Pose)