

1、下载需要用到的数据。首先根据作业要求中的链接下载原始数据，或者直接用百度云作业资料“CH3三维空间变换”中提供的已经下载好的数据（我用的是百度云中提供的数据）。



2、放置下载好的原始数据。首先做第二个kitti\_ros，因为kitti\_ros中有比较详细的环境参数等配置教程，配好之后可以直接调用kitti\_lidar\_camera。首先打开如图红圈链接进行教程阅读，本人后续步骤均根据此链接中的教程来走，遇到的问题我会在此文档中列举出来。

## 作业 2：学习 KITTI 的坐标转换、Lidar 与图像融合（进阶）

### 作业要求

1. 使用数据 2011\_09\_26\_drive\_0001 (0.4 GB) synced+rectified\_data  
[http://www.cvlibs.net/datasets/kitti/raw\\_data.php](http://www.cvlibs.net/datasets/kitti/raw_data.php)
2. 简述其 lidar 点云旋转并投影到图像的实现方法（复制其相应代码及添加说明注释均可）

### 参考资料

- 1、KITTI: [http://eigen.tuxfamily.org/dox/group\\_TutorialGeometry.html](http://eigen.tuxfamily.org/dox/group_TutorialGeometry.html)
- 2、kitti\_lidar\_camera:

[https://github.com/LidarPerception/kitti\\_lidar\\_camera](https://github.com/LidarPerception/kitti_lidar_camera)

[https://github.com/LidarPerception/kitti\\_ros](https://github.com/LidarPerception/kitti_ros)

3、由于python脚本有写好的调用数据路径，所以必须根据图片所示将“CH3三维空间变换”中的原始数据放在指定位置，放置结构如下图，放置的主路径为：

catkin\_ws/data/2011\_09\_26/, 2011\_09\_26中的数据根据图示结构进行放置。

- kitti\_data\_path : KITTI raw data directory.
  - default \$(find kitti\_ros)/../../data/2011\_09\_26/2011\_09\_26\_drive\_0005\_sync , that is \$(CATKIN\_WS)/data/2011\_09\_26/2011\_09\_26\_drive\_0005\_sync .

```

2011_09_26
├── 2011_09_26_drive_0005_sync
│   ├── image_00
│   │   ├── data
│   │   │   ├── 0000000xxx.png
│   │   │   └── ...
│   │   └── timestamps.txt
│   ├── image_01
│   │   ├── data
│   │   │   ├── 0000000xxx.png
│   │   │   └── ...
│   │   └── timestamps.txt
│   ├── image_02
│   │   ├── data
│   │   │   ├── 0000000xxx.png
│   │   │   └── ...
│   │   └── timestamps.txt
│   ├── image_03
│   │   ├── data
│   │   │   ├── 0000000xxx.png
│   │   │   └── ...
│   │   └── timestamps.txt
│   ├── oxts
│   │   ├── data
│   │   │   ├── 0000000xxx.txt
│   │   │   └── ...
│   │   ├── dataformat.txt
│   │   ├── timestamps.txt
│   │   ├── tracklet_labels.xml
│   │   ├── velodyne_points
│   │   │   ├── data
│   │   │   │   ├── 0000000xxx.bin
│   │   │   │   └── xxx
│   │   ├── timestamps_end.txt
│   │   ├── timestamps_start.txt
│   │   └── timestamps.txt
│   └── 201?_??_??_drive_0???_sync
│       ├── ...
│       └── ...
├── calib_cam_to_cam.txt
├── calib_imu_to_velo.txt
└── calib_velo_to_cam.txt
  
```

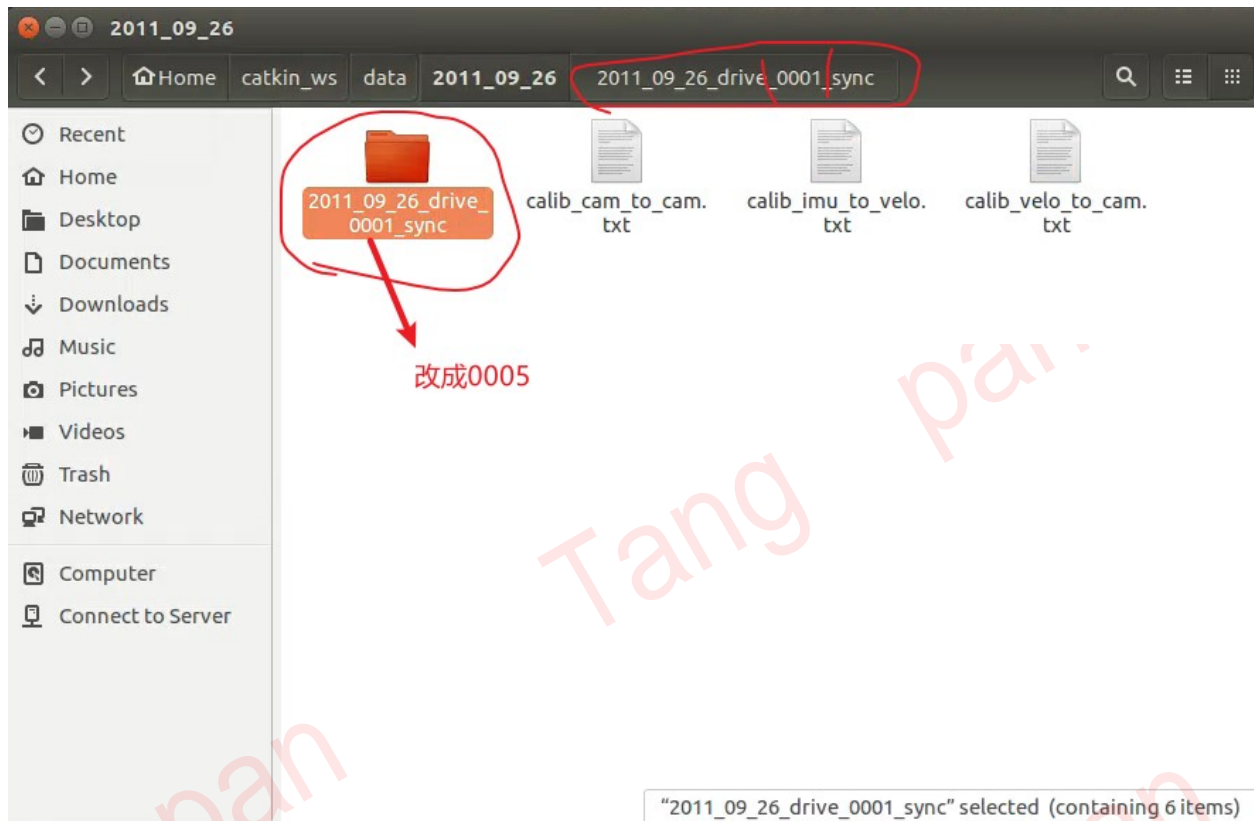
4、由于所有程序中的路径调用代码都是0005，所以需要更改此压缩文档中的0001为0005。

睿慕课三维点云与深度学习... > CH3三维空间变换

搜索"CH3三维空间变换"

名称	修改日期	类型	大小
reference	2020/10/24 17:18	文件夹	
2011_09_26_calib	2020/10/24 15:58	WinRAR ZIP 压缩...	4 KB
2011_09_26_drive_0001_sync	2020/10/24 16:15	WinRAR ZIP 压缩...	447,895 KB

将绿色圈中的0001改成0005



5、根据下图中的命令进行操作。注意事项已在图中列出，在执行catkin build -DCMAKE\_BUILD\_TYPE=Release命令之前需要删除catkin\_ws下除了src和新建的放置原始数据的data之外的包括build、devel、log等在内的其它所有文件夹，否则该命令会执行失败。

We name your ros workspace as `CATKIN_WS` and `git clone kitti_ros` as a ros package.

```
# clone source code
$ cd $(CATKIN_WS)/src
$ git clone https://github.com/LidarPerception/kitti_ros
$ cd kitti_ros
# install python's dependencies
$ pip install -r requirements.txt

# build your ros workspace
$ cd $(CATKIN_WS)
$ catkin build -DCMAKE_BUILD_TYPE=Release

# change Mode for Keyboard Listening Device
$ sudo chmod 777 /dev/input/event3

# [demo] launch kitti_ros's kitti_player with rviz
$ source devel/setup.bash
$ roslaunch kitti_ros demo.launch kitti_data_path:=path-to-your-KiTTI-dataset
```

执行此命令之前需要删掉catkin\_ws下除了src和新建的放置原始数据的data文件外包括build在内的所有文件夹，否则执行失败，自己可以先不删除执行一下该命令看看错误提示。

删掉

6、如图所示，catkin build -DCMAKE\_BUILD\_TYPE=Release命令执行成功。



```

In file included from /home/song/catkin_ws/src/rangeimage/src/range_image.cpp:16
:0:
/usr/include/pcl-1.7/pcl/visualization/cloud_viewer.h:202:14: warning: 'template
<class> class std::auto_ptr' is deprecated [-Wdeprecated-declarations]
    std::auto_ptr<CloudViewer_impl> impl_;
    ^
In file included from /usr/include/c++/5/bits/locale_conv.h:41:0,
                  from /usr/include/c++/5/locale:43,
                  from /usr/include/c++/5/iomanip:43,
                  from /usr/include/boost/math/policies/error_handling.hpp:12,
                  from /usr/include/boost/math/special_functions/round.hpp:14,
                  from /opt/ros/kinetic/include/ros/time.h:58,
                  from /opt/ros/kinetic/include/ros/ros.h:38,
                  from /home/song/catkin_ws/src/rangeimage/src/range_image.cpp:1:
/usr/include/c++/5/bits/unique_ptr.h:49:28: note: declared here
    template<typename> class auto_ptr;
    ^
cd /home/song/catkin_ws/build/rangeimage; catkin build --get-env rangeimage | ca
tkin env -si /usr/bin/make --jobserver-fds=6,7 -j; cd -
.....
Finished <<< rangeimage [ 28.9 seconds ]
[build] Summary: All 9 packages succeeded!
[build] Ignored: None.
[build] Warnings: 3 packages succeeded with warnings.
[build] Abandoned: None.
[build] Failed: None.
[build] Runtime: 30.7 seconds total.
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$

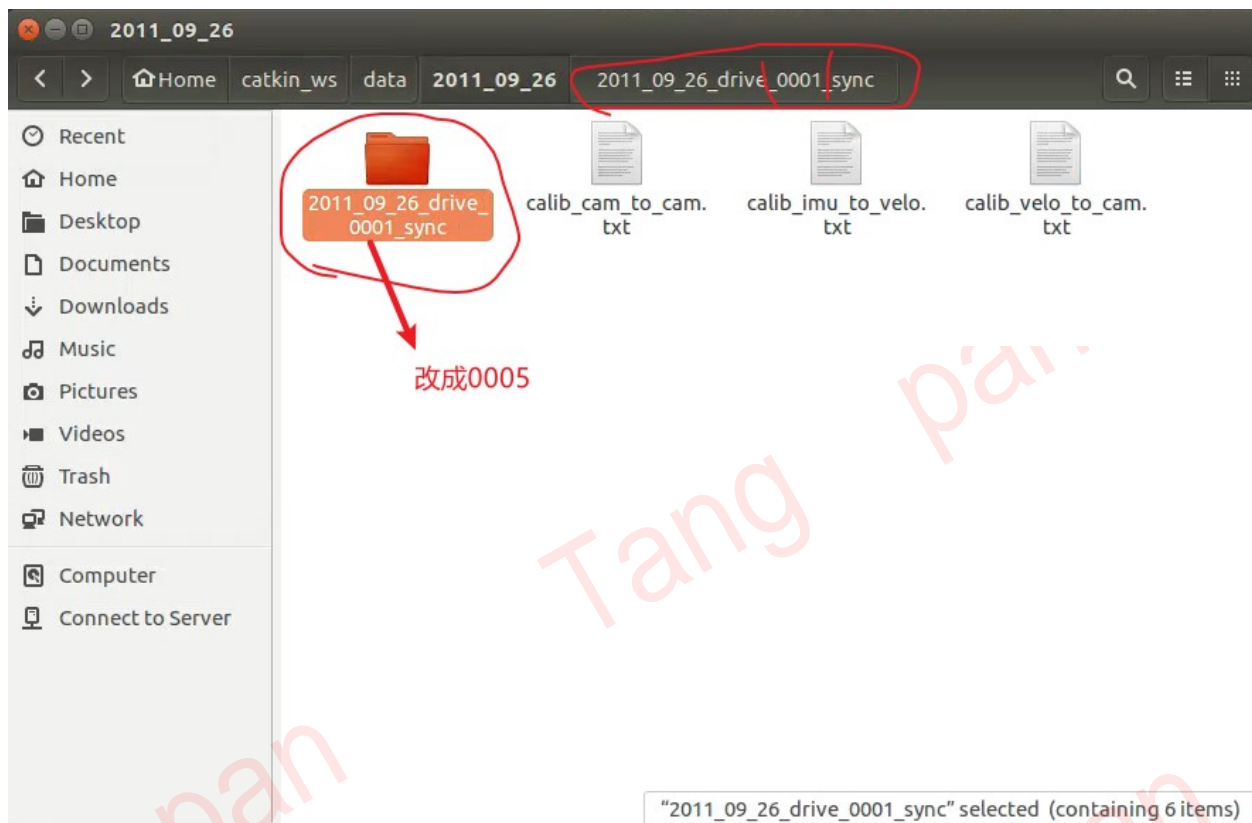
```

7、若出现下图错误则说明原始数据存放路径中0001未改成0005，而本身下载的kitti\_ros中给出的python脚本kitti\_player.py中没有0001路径。

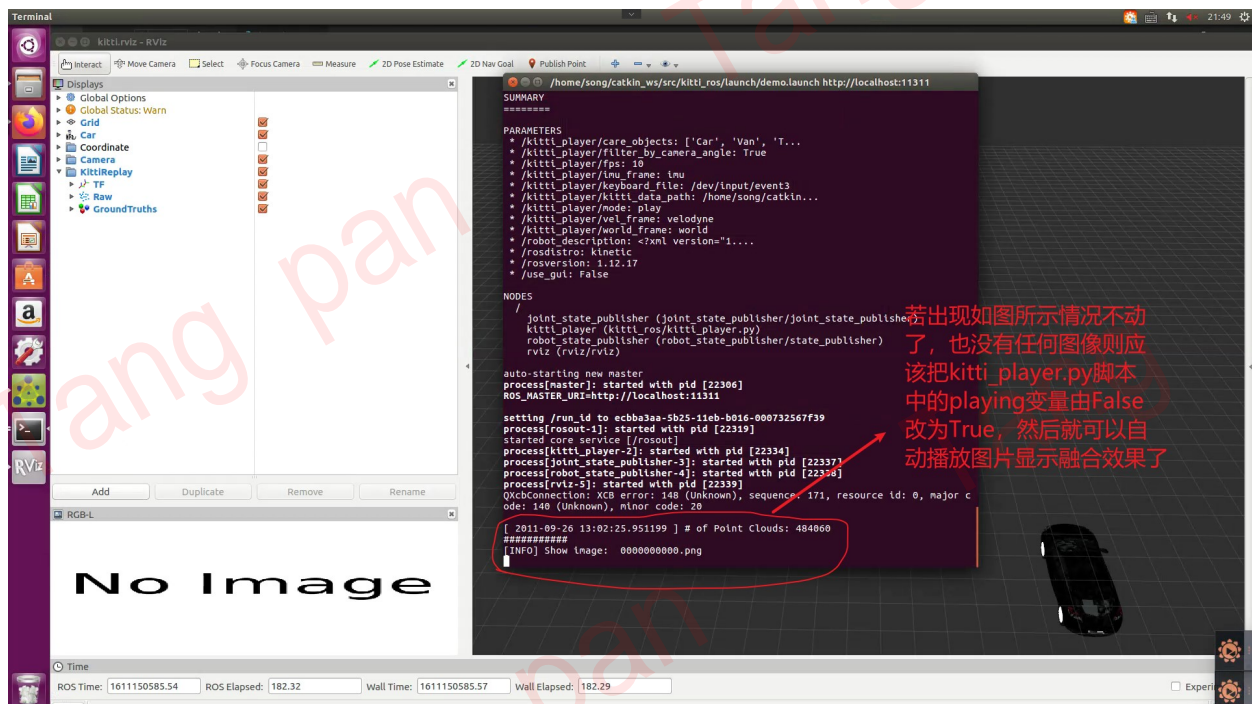
```

started core service [/rosout]
process[kitti_player-2]: started with pid [9086]
process[joint_state_publisher-3]: started with pid [9089]
process[robot_state_publisher-4]: started with pid [9090]
process[rviz-5]: started with pid [9091]
QXcbConnection: XCB error: 148 (Unknown), sequence: 171, resource id: 0, major c
ode: 140 (Unknown), minor code: 20
Traceback (most recent call last):
  File "/home/song/catkin_ws/src/kitti_ros/scripts/kitti_player.py", line 117, i
n <module>
    with open(timestamp_file, 'r') as f:
IOError: [Errno 2] No such file or directory: '/home/song/catkin_ws/src/kitti_ro
s/../../data/2011_09_26/2011_09_26_drive_0005_sync/velodyne_points/timestamps.tx
st'
[kitti_player-2] process has died [pid 9086, exit code 1, cmd /home/song/catkin_
ws/src/kitti_ros/scripts/kitti_player.py __name:=kitti_player __log:=/home/song/
.ros/log/e67ddc0c-5b24-11eb-b016-000732567f39/kitti_player-2.log].
log file: /home/song/.ros/log/e67ddc0c-5b24-11eb-b016-000732567f39/kitti_player-
2*.log
[rviz-5] process has finished cleanly
log file: /home/song/.ros/log/e67ddc0c-5b24-11eb-b016-000732567f39/rviz-5*.log
^C[robot_state_publisher-4] killing on exit
[joint_state_publisher-3] killing on exit
[rosout-1] killing on exit
[master] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
song@song-SIMATIC-IPC3000-SMART:~/catkin_ws$

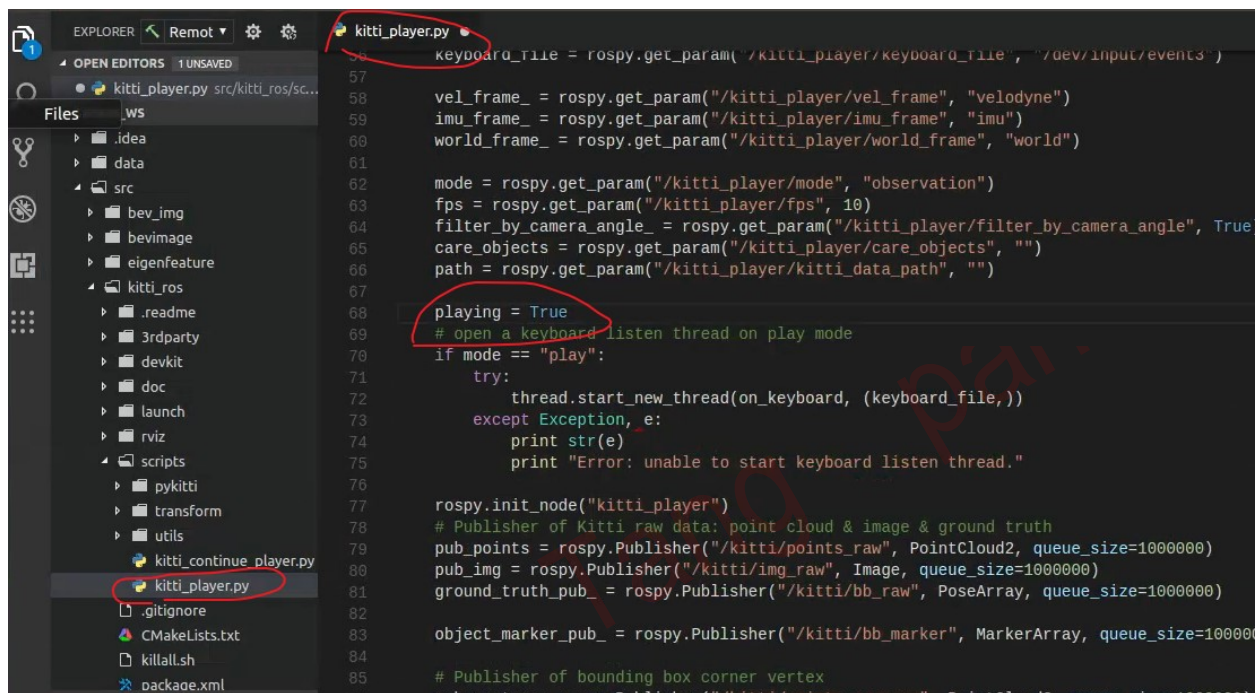
```



8、执行roslaunch命令之前需要将kitti\_player.py 中的playing变量由False改为True。更改变量之后再执行roslaunch kitti\_ros demo.launch命令，若没有更改playing变量则会出现如图所示情况：

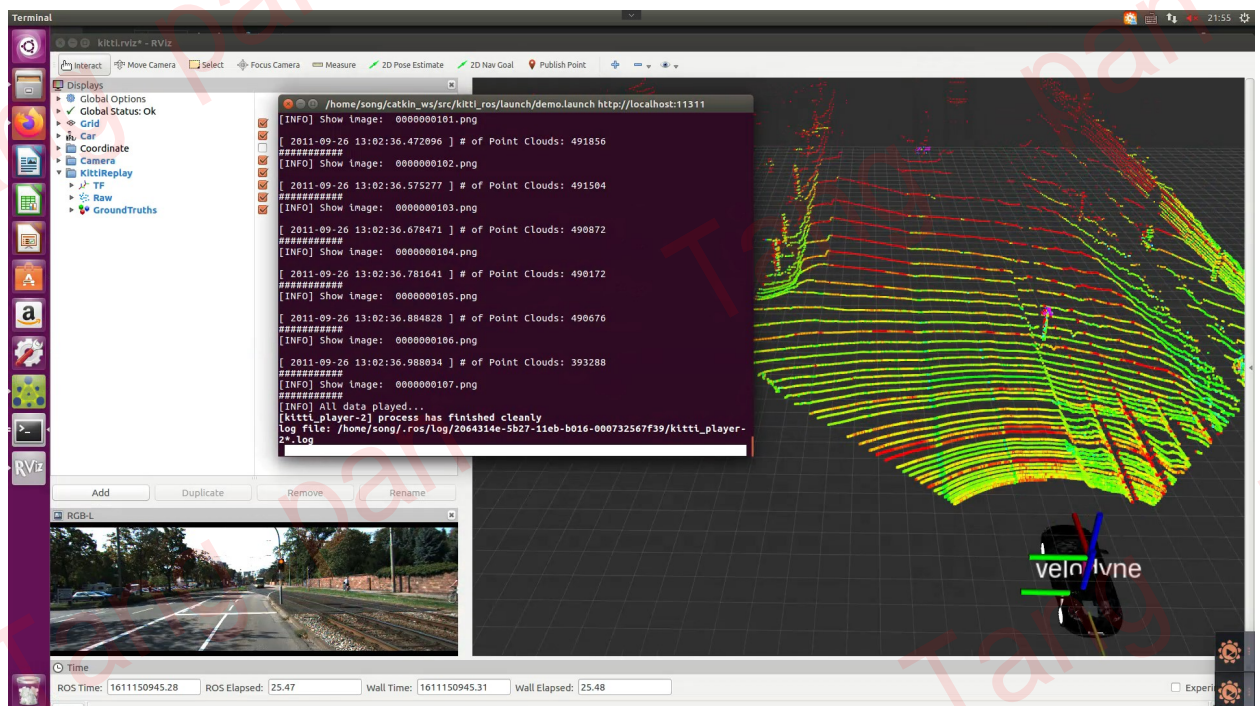






```
56 keyboard_file = rospy.get_param("/kitti_player/keyboard_file", "/dev/input/events")
57
58 vel_frame_ = rospy.get_param("/kitti_player/vel_frame", "velodyne")
59 imu_frame_ = rospy.get_param("/kitti_player/imu_frame", "imu")
60 world_frame_ = rospy.get_param("/kitti_player/world_frame", "world")
61
62 mode = rospy.get_param("/kitti_player/mode", "observation")
63 fps = rospy.get_param("/kitti_player/fps", 10)
64 filter_by_camera_angle_ = rospy.get_param("/kitti_player/filter_by_camera_angle", True)
65 care_objects = rospy.get_param("/kitti_player/care_objects", "")
66 path = rospy.get_param("/kitti_player/kitti_data_path", "")
67
68 playing = True
69 # open a keyboard listen thread on play mode
70 if mode == "play":
71     try:
72         thread.start_new_thread(on_keyboard, (keyboard_file,))
73     except Exception, e:
74         print str(e)
75         print "Error: unable to start keyboard listen thread."
76
77 rospy.init_node("kitti_player")
78 # Publisher of Kitti raw data: point cloud & image & ground truth
79 pub_points = rospy.Publisher("/kitti/points_raw", PointCloud2, queue_size=1000000)
80 pub_img = rospy.Publisher("/kitti/img_raw", Image, queue_size=1000000)
81 ground_truth_pub_ = rospy.Publisher("/kitti/bb_raw", PoseArray, queue_size=1000000)
82
83 object_marker_pub_ = rospy.Publisher("/kitti/bb_marker", MarkerArray, queue_size=1000000)
84
85 # Publisher of bounding box corner vertex
86 pub_vertex = rospy.Publisher("/kitti/points_corner", PointCloud2, queue_size=1000000)
```

9、修改之后执行roslaunch kitti\_ros demo.launch正常运行：



10、kitti\_lidar\_camera中操作一样，如果前边配置好之后只需要按照下图指令重新走一遍就可以实现预期效果，别忘了改playing变量哦。

## How to use

We name your ros workspace as `CATKIN_WS` and git clone `kitti_lidar_camera` as a ros package.

```
# clone source code
$ cd $(CATKIN_WS)/src
$ git clone https://github.com/LidarPerception/kitti_lidar_camera

# build your ros workspace
$ cd $(CATKIN_WS)
$ catkin build -DCMAKE_BUILD_TYPE=Release

$ source devel/setup.bash
# change Mode for Keyboard Listening Device
$ sudo chmod 777 /dev/input/event3

# [option 1]launch kitti_lidar_camera's kitti_player with rviz
$ roslaunch kitti_lidar_camera demo.launch
# [option 2]launch kitti_lidar_camera's kitti_player without rviz
$ roslaunch kitti_lidar_camera kitti_lidar_camera.launch
```

最终效果如下图：

