UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

Corso di Laboratorio Ciberfisico

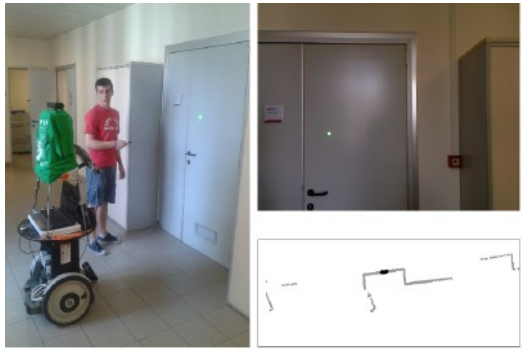
Modulo di Robot Programming with ROS

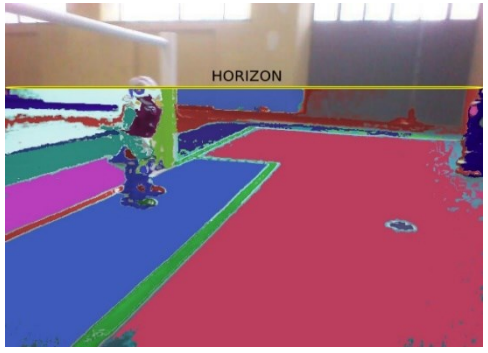
Esercitazione

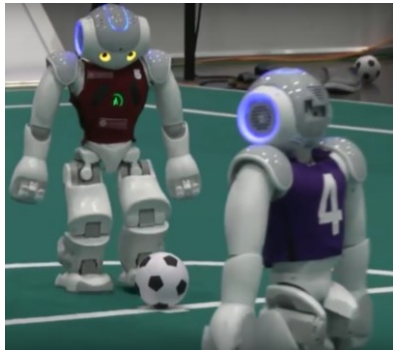
Face detection

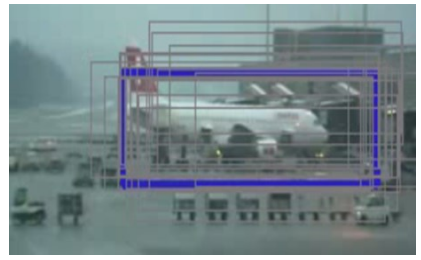
Docente:
Domenico Daniele Bloisi

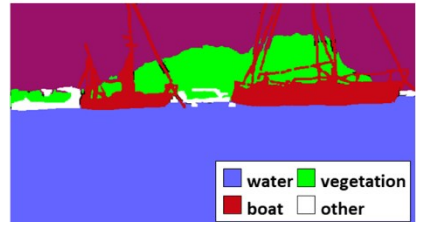
Maggio 2018



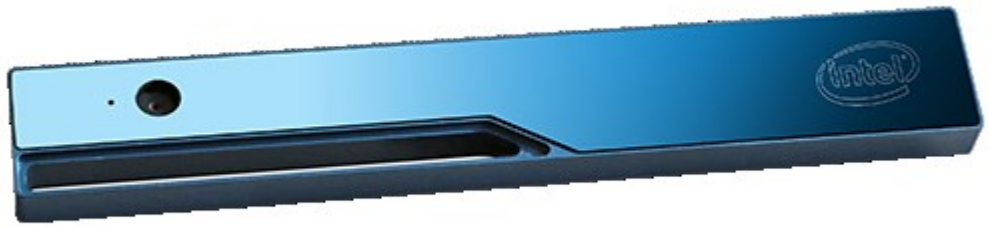


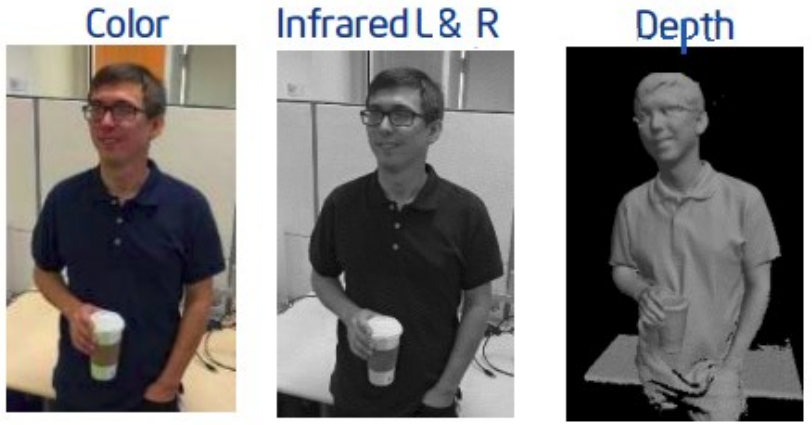






Intel RealSense Camera R200

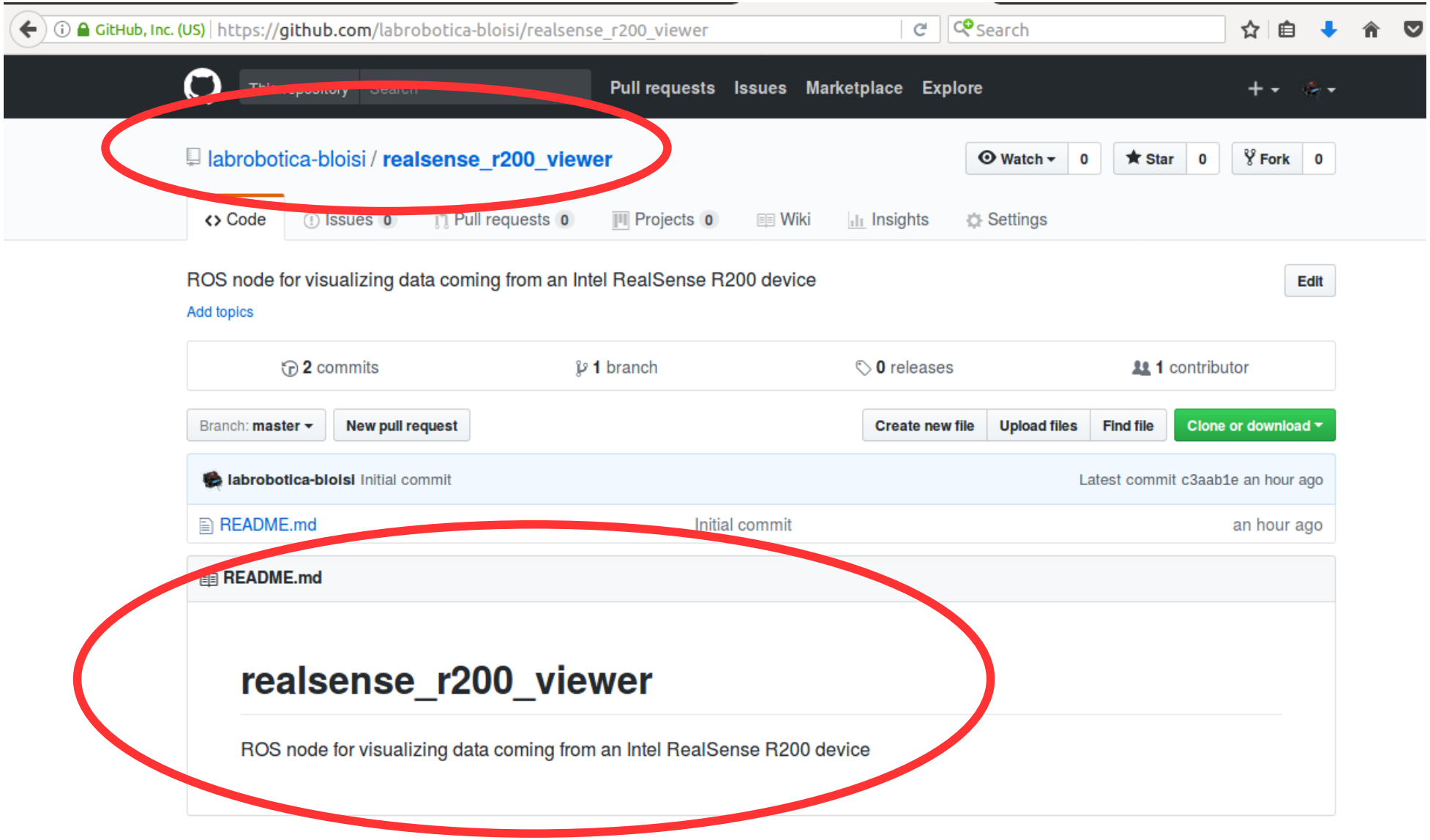




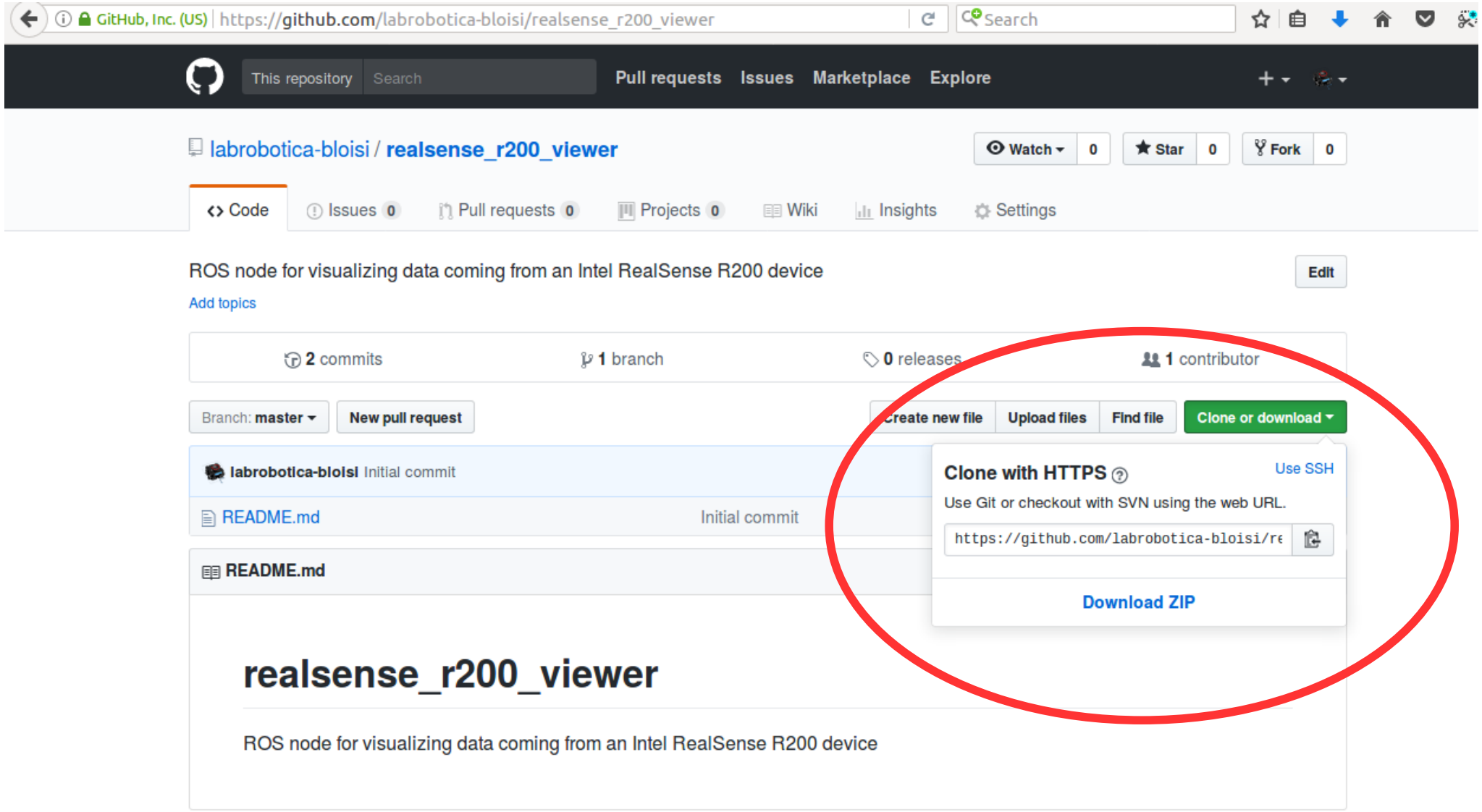
- The R200 actually has 3 cameras providing RGB (color) and stereoscopic IR to produce depth
- The inside range is approximately 0.5-3.5 meters and an outside range up to 10 meters
- With the help of a laser projector, the camera does 3D scanning for scene perception and enhanced photography

https://software.intel.com/en-us/articles/realsense-r200-camera

Realsense_r200_viewer

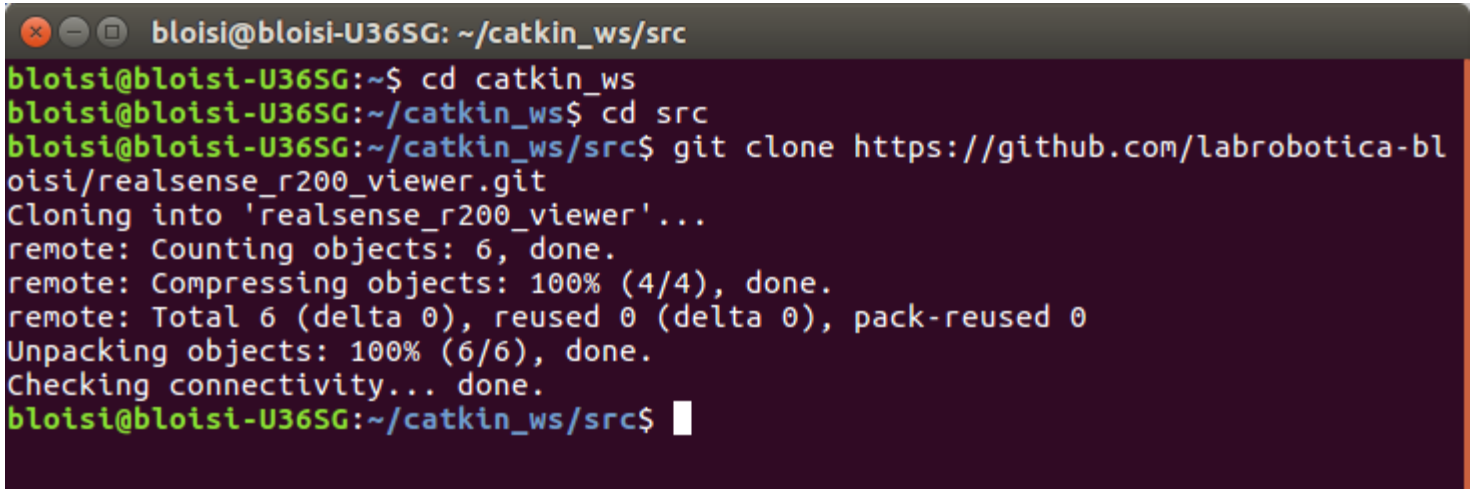


clone


GitHub, Inc. (US) https://github.com/labrobotica-bloisi/realsense_r200_viewer
Search
This repository
Search
Pull requests
Issues
Marketplace
Explore
labrobotica-bloisi / realsense_r200_viewer
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
ROS node for visualizing data coming from an Intel RealSense R200 device
Edit
Add topics
2 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
labrobotica-bloisi Initial commit
README.md
Initial commit
README.md
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/labrobotica-bloisi/re
Download ZIP
realsense_r200_viewer
ROS node for visualizing data coming from an Intel RealSense R200 device

Creazione del repository locale

Repository su GitHub
https://github.com/labrobotica-bloisi/realsense_r200_viewer



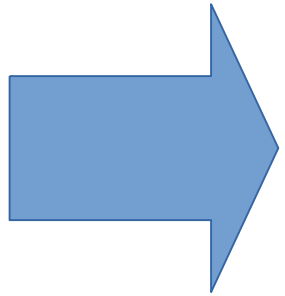
Repository locale
creato in
~/catkin_ws/src

Package.xml

```
<?xml version="1.0"?>
<package>
  <name>realsense_r200_viewer</name>
  <version>0.0.1</version>
  <description>realsense_r200_viewer package</description>
  <maintainer email="domenico.bloisi@gmail.com">Domenico Bloisi</maintainer>
  <license>LGPLv3</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>cv_bridge</build_depend>
  <build_depend>image_transport</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>sensor_msgs</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>cv_bridge</run_depend>
  <run_depend>image_transport</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>sensor_msgs</run_depend>
  <run_depend>std_msgs</run_depend>
</package>
```

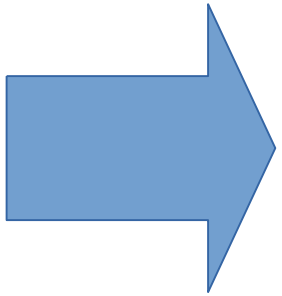
r200_viewer.cpp

```
#include <ros/ros.h>
#include <sensor_msgs/Image.h>
#include <cv_bridge/cv_bridge.h>

#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>

using namespace std;

void rgb_callback(const sensor_msgs::ImageConstPtr& msg){
    cv_bridge::CvImagePtr cv_ptr;

    try{
        cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
    } catch (cv_bridge::Exception& e){
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }

    cv::imshow("RGB", cv_ptr->image);
    cv::waitKey(30);
}
```



r200_viewer.cpp

```
void depth_callback(const sensor_msgs::ImageConstPtr& msg){
    cv_bridge::CvImagePtr cv_ptr;

    try {
        cv_ptr = cv_bridge::toCvCopy(msg, msg->encoding);
    } catch (cv_bridge::Exception& e) {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }

    cv::Mat depthMat = cv_ptr->image.clone();

    double dmin, dmax;
    cv::minMaxIdx(depthMat, &dmin, &dmax);
    cv::Mat adjMat;
    cv::convertScaleAbs(depthMat, adjMat, 255 / dmax);

    cv::Mat colorMat;
    cv::applyColorMap(adjMat, colorMat, cv::COLORMAP_HOT);
    cv::imshow("Depth", colorMat);

    cv::waitKey(30);
}
```



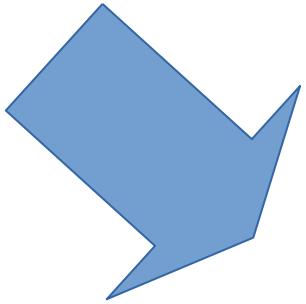
r200_viewer.cpp

```
int main(int argc, char **argv)
{
    ros::init(argc, argv, "realsense_r200_viewer");

    ros::NodeHandle nh;

    string topic_rgb = "camera/rgb/image_rect_color";
    string topic_depth = "camera/depth_registered/sw_registered/image_rect_raw";

    cout << "Subscriptions:" << endl;
    cout << "  - RGB topic: " << topic_rgb << endl;
    cout << "  - Depth topic: " << topic_depth << endl;

    ros::Subscriber rgb_sub = nh.subscribe(topic_rgb, 1, &rgb_callback);
    ros::Subscriber depth_sub = nh.subscribe(topic_depth, 1, &depth_callback);

    ros::spin();

    return 0;
}
```

catkin_make

```
catkin_make --pkg realsense_r200_viewer
```

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ catkin_make --pkg realsense_r200_viewer
```

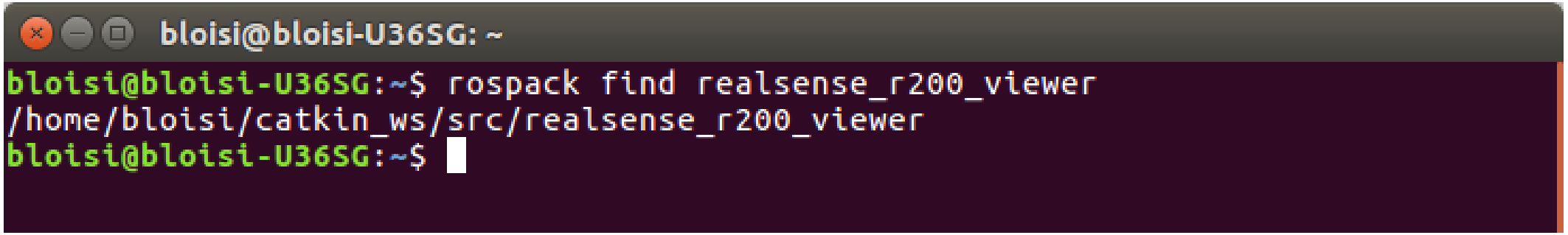


```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ catkin_make --pkg realsense_r200_viewer
Base path: /home/bloisi/catkin_ws
Source space: /home/bloisi/catkin_ws/src
Build space: /home/bloisi/catkin_ws/build
Devel space: /home/bloisi/catkin_ws/devel
Install space: /home/bloisi/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/bloisi/catkin_ws
/build"
####
####
#### Running command: "make -j4 -l4" in "/home/bloisi/catkin_ws/build/realsense_
r200_viewer"
####
[100%] Built target realsense_r200_viewer
bloisi@bloisi-U36SG:~/catkin_ws$
```

Finding a ROS package

Now that your package has a manifest, ROS can find it. Try executing the command:

```
rospack find realsense_r200_viewer
```



if ROS is set up correctly you should see the physical location where your package is stored

http://wiki.ros.org/ROS/Tutorials/Creating%20a%20Package%20by%20Hand

roscore + rosrun

Apriamo un terminale e lanciamo `roscore`

```
roscore http://localhost:11311/
bloisi@bloisi-U36SG:~$ roscore
... logging to /home/bloisi/.ros/log/ff2cc138-d525-11e7-a75c-dc85de574b1d/roslaunch-bloisi-U36SG-21837.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:40525/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [21848]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to ff2cc138-d525-11e7-a75c-dc85de574b1d
process[rosout-1]: started with pid [21861]
started core service [/rosout]
```

Apriamo un secondo terminale e lanciamo

`rosrun realsense_r200_viewer realsense_r200_viewer`

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws$ rosrun realsense_r200_viewer realsense_r200_viewer
Subscriptions:
  - RGB topic: camera/rgb/image_rect_color
  - Depth topic: camera/depth_registered/sw_registered/image_rect_raw
```

Cosa accade?

Sensore reale o bag file

Abbiamo bisogno di dati in ingresso per poterli visualizzare

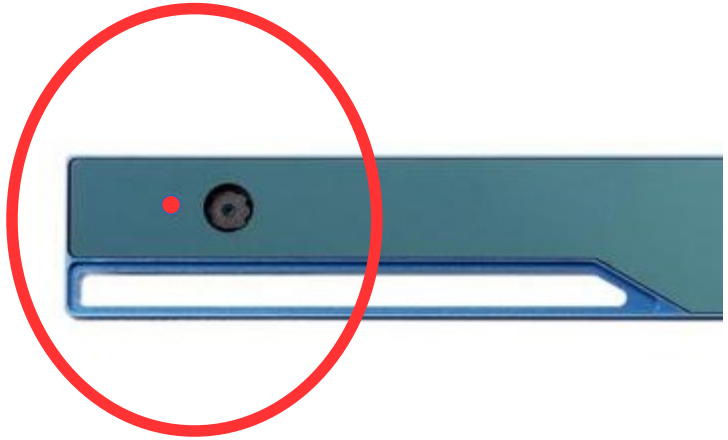
- i dati vengono acquisiti tramite il sensore reale
- utilizziamo i dati registrati contenuti in un bag file

realsense_camera node

```
roslaunch realsense_camera r200_nodelet_rgbd.launch
```

```
/opt/ros/kinetic/share/realsense_camera/launch/r200_nodelet_default.launch http://loc
[ INFO] [1511975468.786957868]: Initializing nodelet with 4 worker threads.
[ INFO] [1511975469.042562486]: /camera/driver - Detected the following camera:
                                - Serial No: 2511001026, USB Port ID: 4-1, Name:
 Intel RealSense R200, Camera FW: 1.0.71.06
[ WARN] [1511975469.042746704]: /camera/driver - Detected unvalidated firmware:
                                - 2511001026's current camera firmware is 1.0.71
.06, Validated camera firmware is 1.0.72.06
[ INFO] [1511975469.042882182]: /camera/driver - Connecting to camera with Seria
l No: 2511001026, USB Port ID: 4-1
[ INFO] [1511975469.618409936]: /camera/driver - Setting static camera options
[ INFO] [1511975469.625166570]: /camera/driver - Enabling Depth in manual mode
[ INFO] [1511975469.625406256]: /camera/driver - Enabling Color in manual mode
[ INFO] [1511975469.625718834]: /camera/driver - Starting camera
[ INFO] [1511975469.643368625]: /camera/driver - Publishing camera transforms (/
tf_static)
[ INFO] [1511975469.643552017]: /camera/driver - Setting dynamic camera options
(r200_dc_preset=5)
[ INFO] [1511975470.272138956]: /camera/driver - Initializing Depth Control Pres
et to 5
[ INFO] [1511975472.369285536]: /camera/driver - Setting dynamic camera options
```

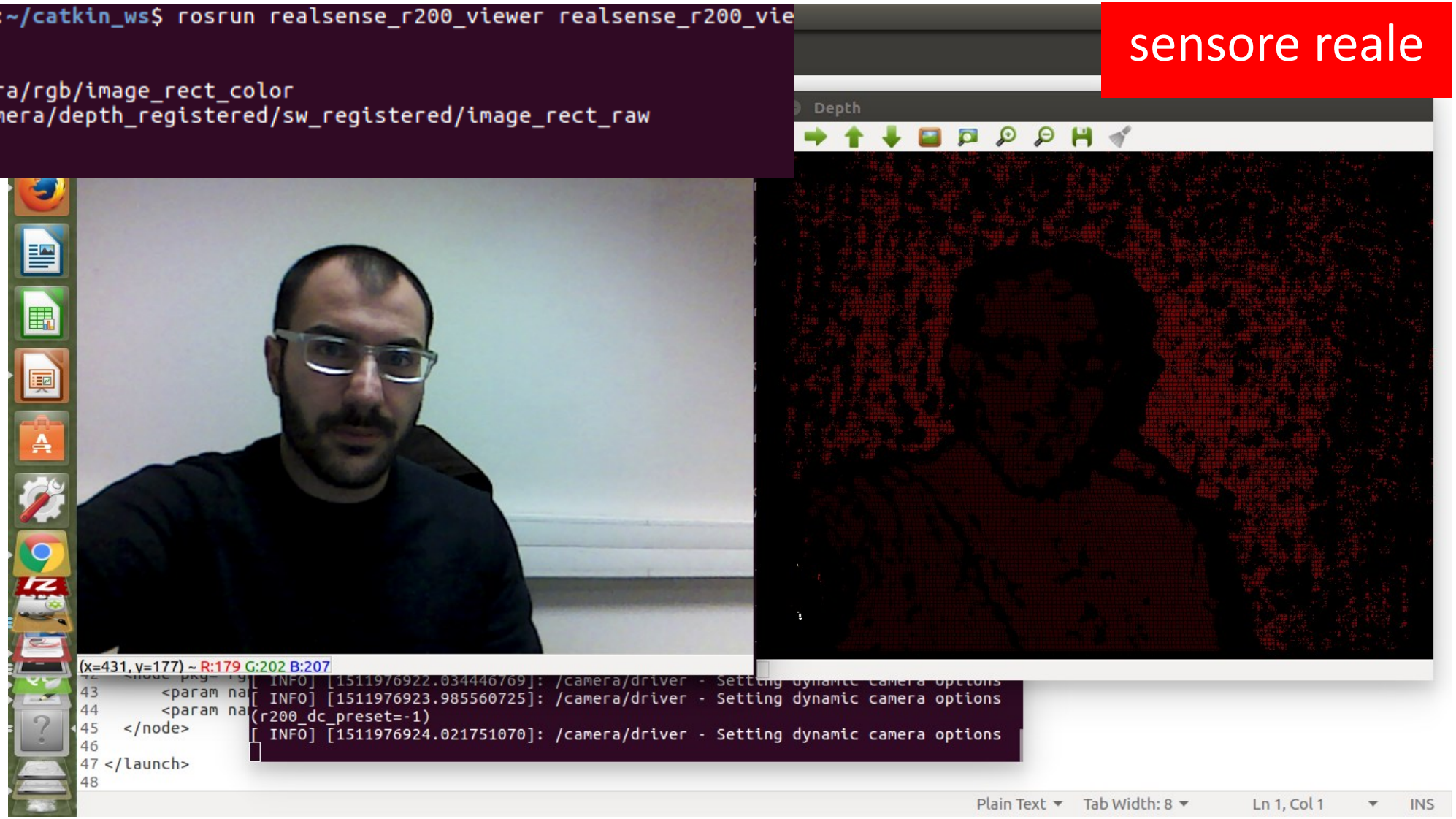
sensore reale



rosrun (riproviamo)

```
bloisi@bloisi-U36SG:~/catkin_ws$ rosrun realsense_r200_viewer realsense_r200_vie
wer
Subscriptions:
  - RGB topic: camera/rgb/image_rect_color
  - Depth topic: camera/depth_registered/sw_registered/image_rect_raw
init done
```

sensore reale



rosbag

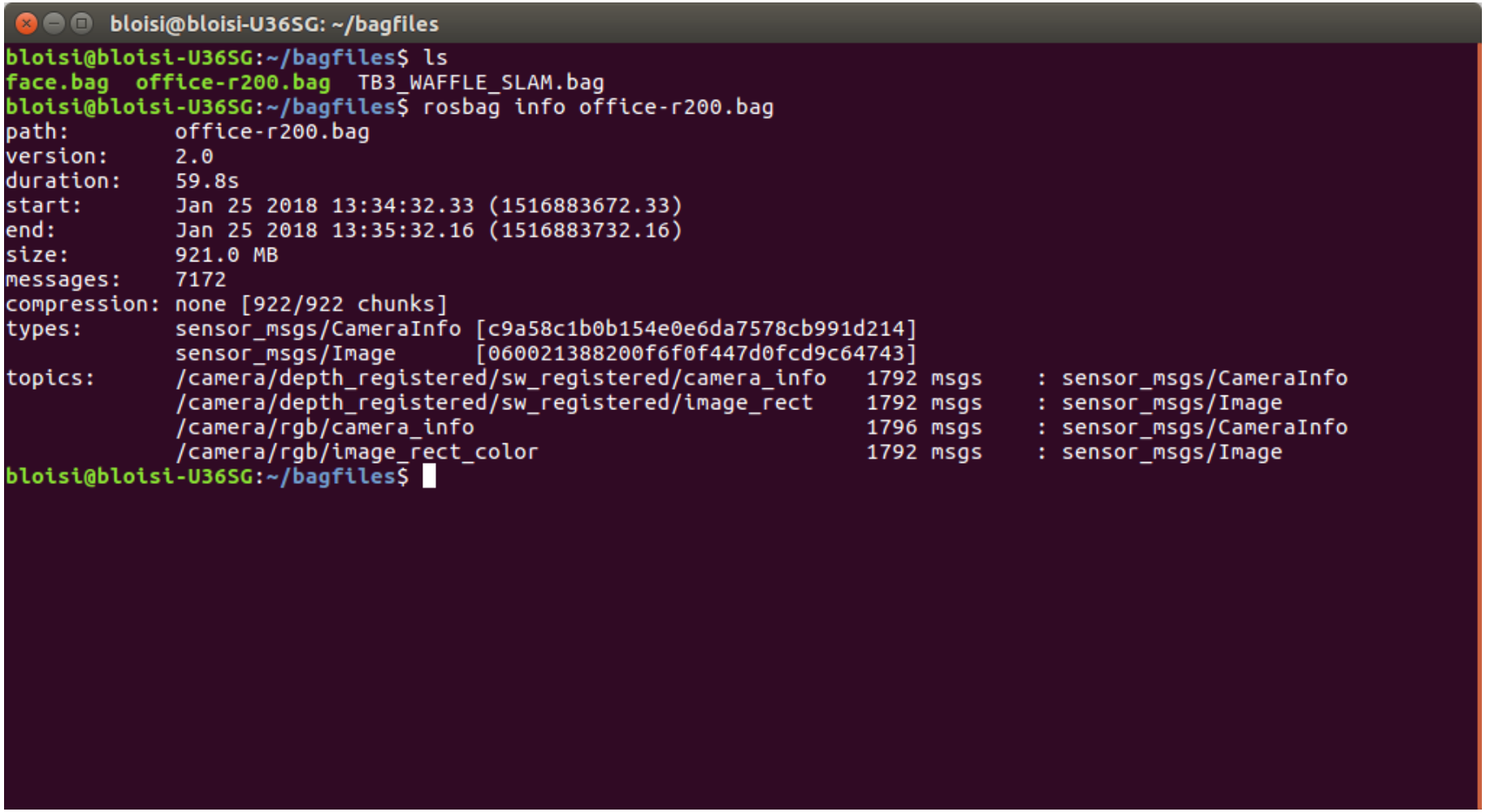
Scarichiamo la ROS bag da

http://profs.scienze.univr.it/~bloisi/corsi/lezionicf/office-r200.bag

Si tratta di un file con una grandezza maggiore di 900MB

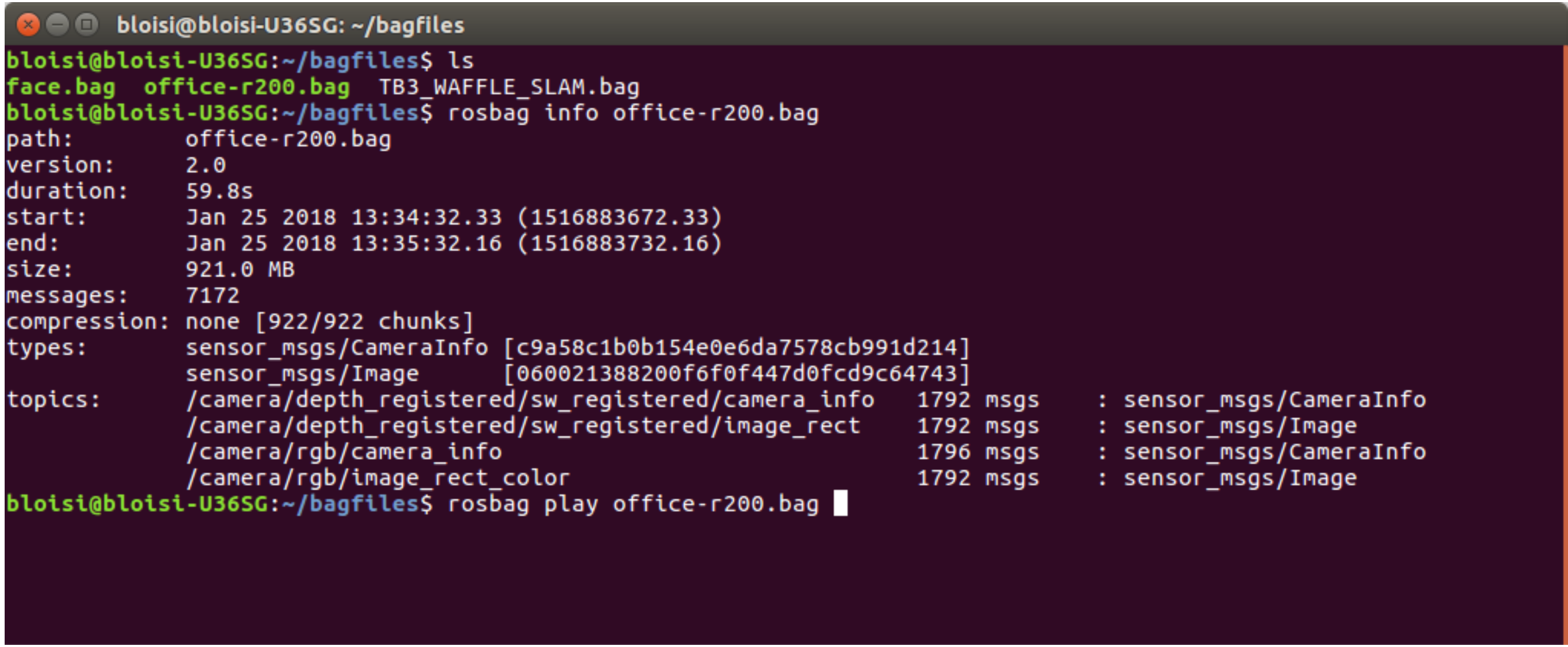
rosbag info

```
$ rosbag info office-r200.bag
```



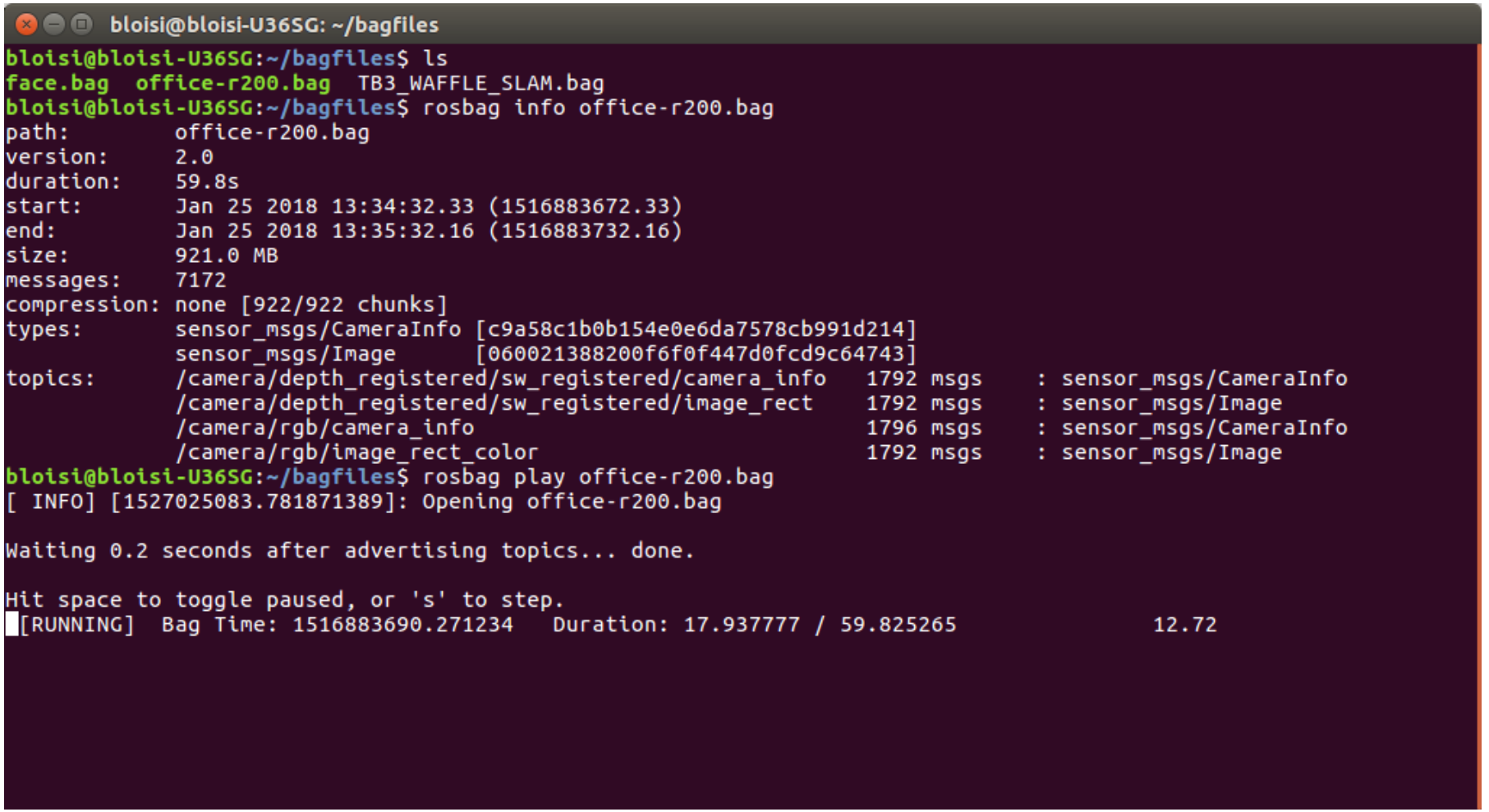
rosbag play

```
$ rosbag play office-r200.bag
```



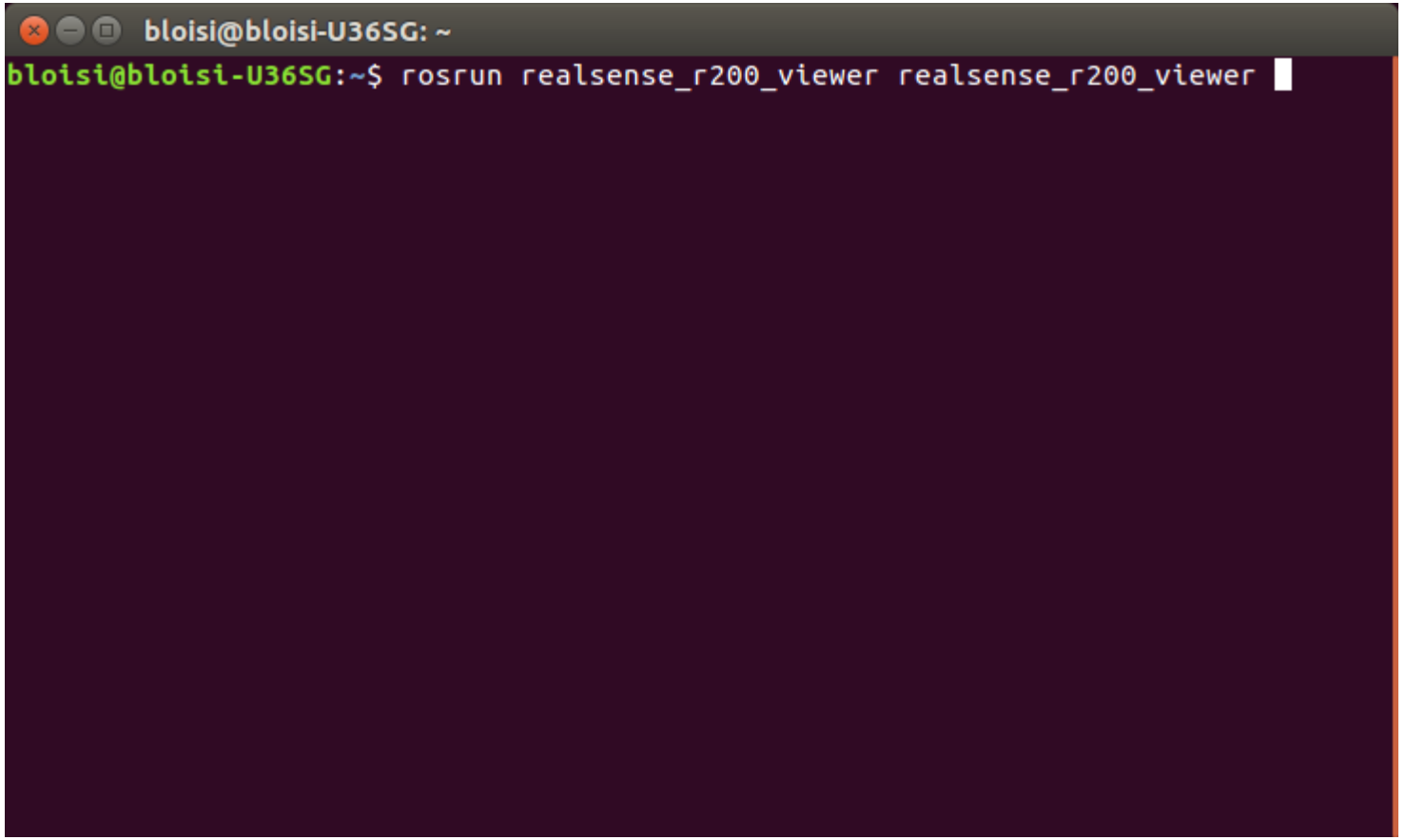
rosbag play - esecuzione

```
$ rosbag play office-r200.bag
```



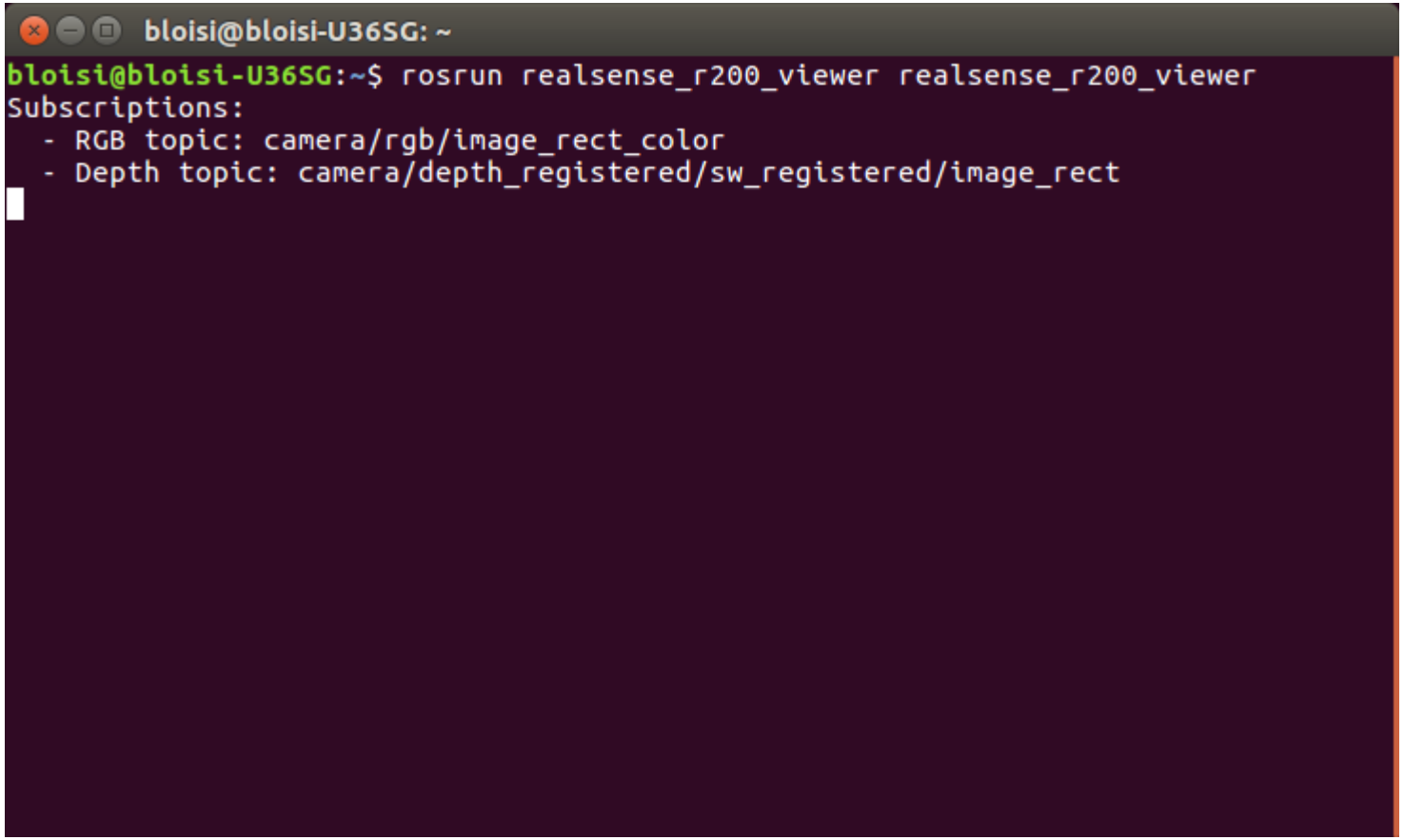
rosrun con bag file

```
$ rosrun realsense_r200_viewer real_sense_r200_viewer
```



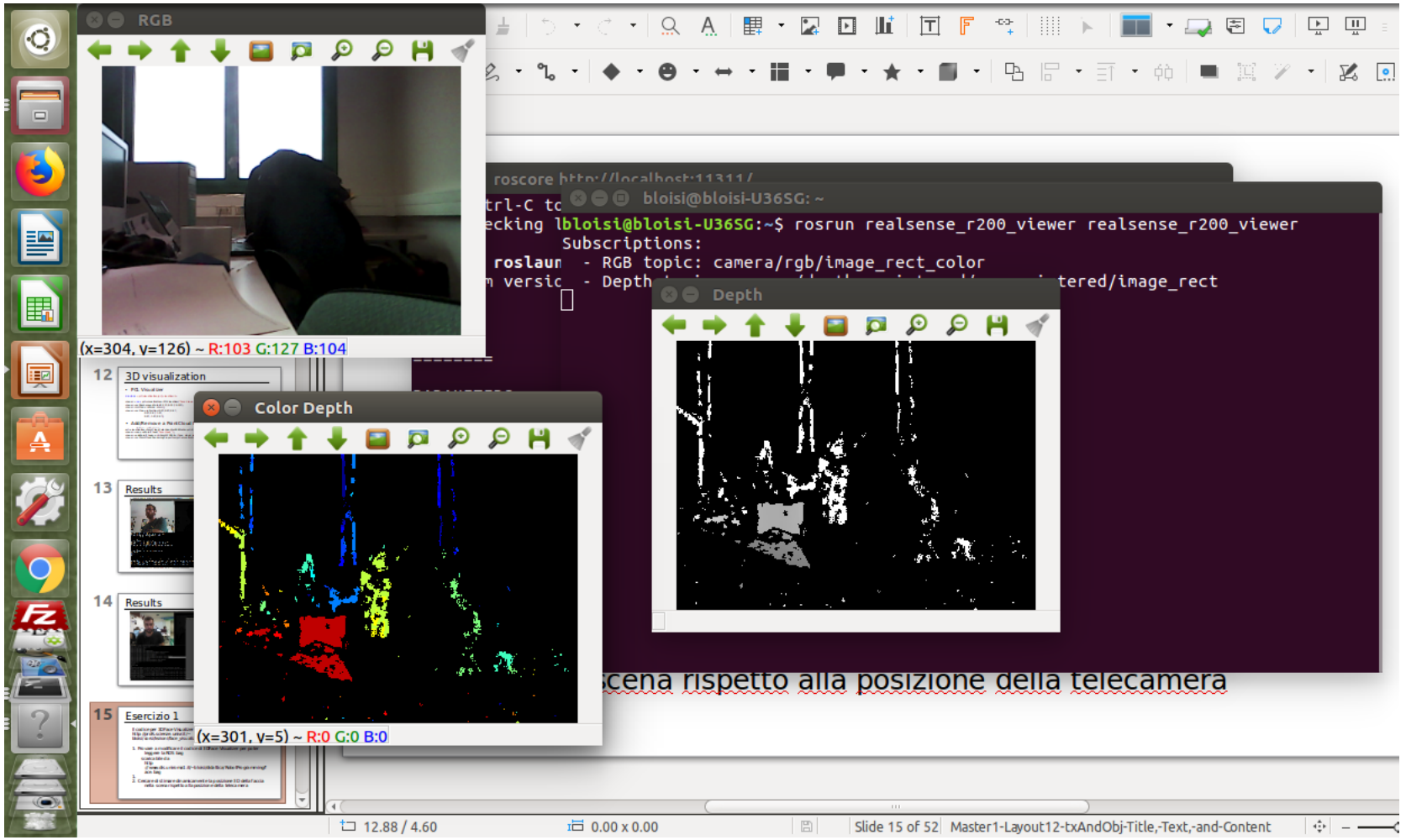
rosrun con bag file - esecuzione

```
$ rosrun realsense_r200_viewer real_sense_r200_viewer
```



rosrun con bag file - esecuzione

```
$ rosrun realsense_r200_viewer real_sense_r200_viewer
```



Esercizio

Scrivere un nodo ROS in grado di

1. leggere lo stream video proveniente dalla webcam integrata nel pc (o da una webcam esterna)
2. mostrare a video le immagini usando OpenCV

Pubblicare il codice usando un repository Git

3D Face Visualization

special thanks to
Roberto Capobianco and
Jacopo Serafin

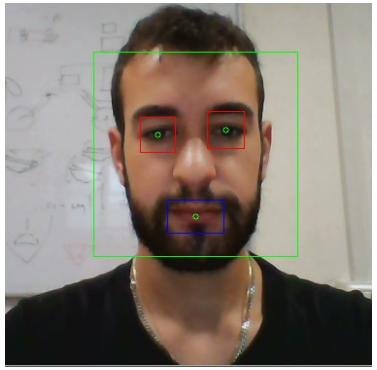
Tools

- Microsoft Kinect or Asus Xtion
- OpenCV (Open Computer Vision)
- PCL (Point Cloud Library)
- ROS (Robot Operating System)

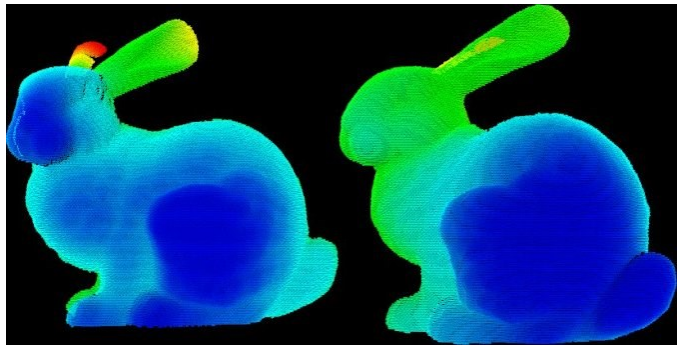
2D + Depth + 3D

- Data Acquisition (ROS + Kinect)
- Face Detection (OpenCV)
- 3D Visualization (PCL)

Depth image



RGB image



3D model

ROS topics

- Kinect topic subscription
  Receive messages published by the Kinect node
  Content of messages: Depth and RGB Images
  Depth registered topic: one-by-one pixel correspondence between Depth and RGB Images

- Topic synchronization
  Required for processing pairs of Depth and RGB Images close in terms of publishing time

ROS Callbacks

- Callback function:

  Binded to one or more (synchronized) topics

  Executed on a secondary thread whenever a new message is received

```
void callback(const ImageConstPtr& depthImage_) {
        …
}

void synchronized_callback(const ImageConstPtr& depthImage_,
                           const ImageConstPtr& rgbImage_) {
        …
}
```

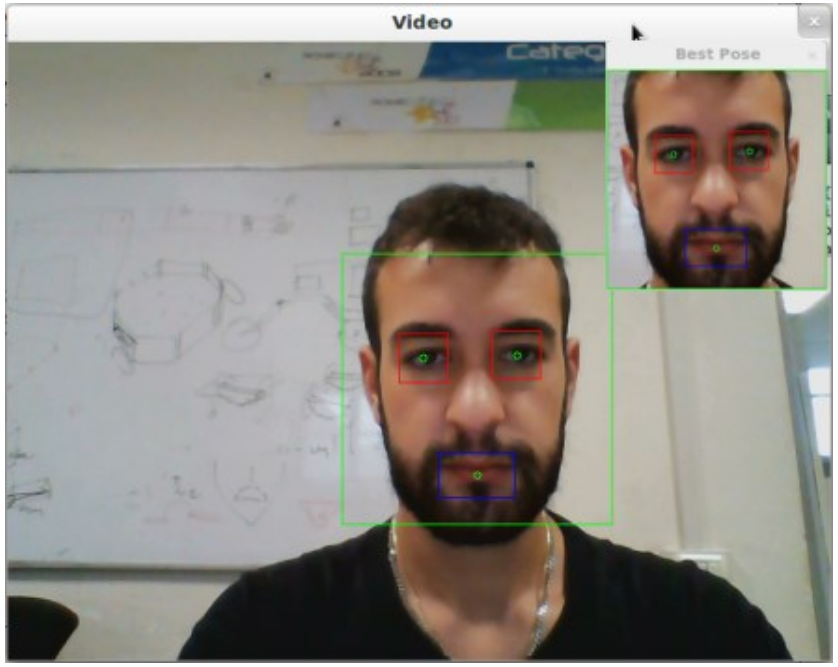
Acquisizione dei dati

- Kinect topics
  "/camera/depth_registered/image_rect_raw"
  "/camera/rgb/image_rect"
- Topic subscription, synchronization and callback registration

```
#include <message_filters/subscriber.h>
#include <message_filters/synchronizer.h>
#include <message_filters/sync_policies/approximate_time.h>

ros::NodeHandle nh;
message_filters::Subscriber<Image> depth_sub(nh, "topic1", 1);
message_filters::Subscriber<Image> rgb_sub(nh, "topic2", 1);

typedef sync_policies::ApproximateTime<Image, Image> syncPolicy;
Synchronizer<syncPolicy> sync(syncPolicy(10), depth_sub, rgb_sub);
sync.registerCallback(boost::bind(&callback, _1, _2));
```

Face Detection Demo

- Face detection on the whole image, both for frontal and profile faces
  The face which is selected among the alternatives is the most visible one or, more formally, the face with the biggest area



Face Detection Demo

- Haar-like cascade declaration

```
#include <opencv2/opencv.hpp>
#include <opencv2/imgproc/imgproc.hpp>

cv::CascadeClassifier frontal_face_cascade;
cv::CascadeClassifier profile_face_cascade;

if(!frontal_face_cascade.load(frontalFaceCascadeFilename) ||
   !profile_face_cascade.load(profileFaceCascadeFilename)) {
    std::cerr << "Error while loading HAAR cascades." << std::endl;
    return -1;
}
```

- Search the feature in the RGB image

```
frontal_face_cascade.detectMultiScale(grayImage, frontal_face_vector, 1.4, 4,
                                      0|CV_HAAR_SCALE_IMAGE, cv::Size(50, 50));
profile_face_cascade.detectMultiScale(grayImage, profile_face_vector, 1.4, 4,
                                      0|CV_HAAR_SCALE_IMAGE, cv::Size(50, 50));
```

3D visualization

PCL Visualizer is PCL's full-featured visualization class

- PointCloud visualization with RGB information
- Normal displaying
- Shape drawing
- Multiple viewports

2D to 3D

Depth point to 3D Cartesian point

$$\mathbf{p} = \mathbf{K}^{-1} \cdot (u, v, 1)^T$$

```
pcl::PointCloud<pcl::PointXYZRGB>::Ptr face_cloud(new pcl::PointCloud<pcl::PointXYZRGB>);
float cx = 319.5f; //optical center x coordinate
float cy = 239.5f; //optical center y coordinate
float f = 525.0f;  //focal length (the same for x and y)
pcl::PointXYZRGB point;
point.z = d / 1000.0f;
point.x = (imageWidth - cx) * point.z / f;
point.y = (imageHeight - cy) * point.z / f;
cv::Vec3b pixel = rgbImage.at<cv::Vec3b>(imageHeight, imageWidth);
point.r = pixel[2];
point.g = pixel[1];
point.b = pixel[0];
face_cloud->points.push_back(point);
```

3D visualization

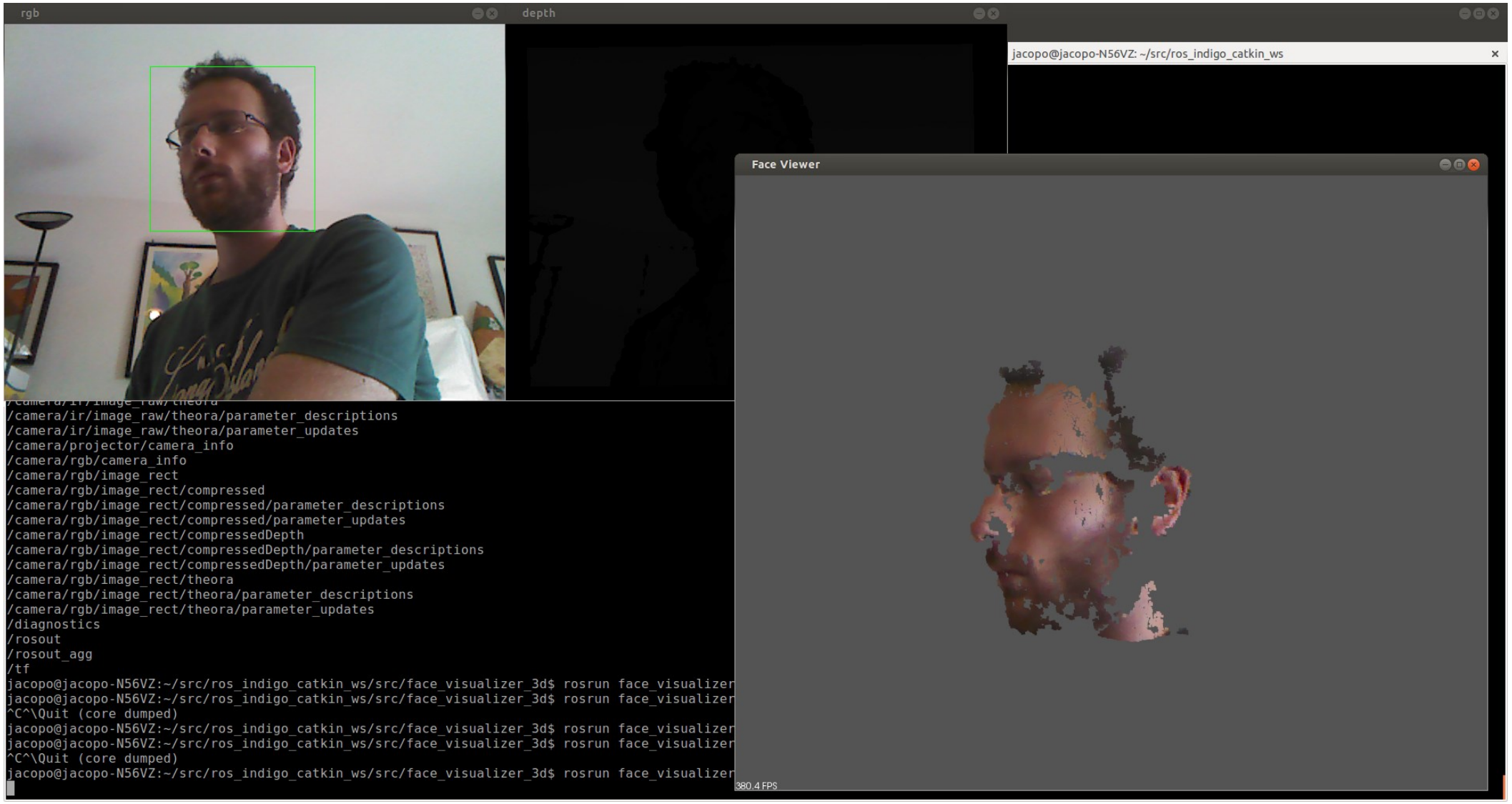
- PCL Visualizer

```
#include <pcl/visualization/pcl_visualizer.h>

viewer = new pcl::visualization::PCLVisualizer("Face Viewer");
viewer->setBackgroundColor(0.33f, 0.33f, 0.33f);
viewer->initCameraParameters();
viewer->setCameraPosition(0.0f, 0.0f, 0.0f,
                          0.0f, 0.0f, 1.0f,
                          0.0f, -1.0f, 0.0f);
```

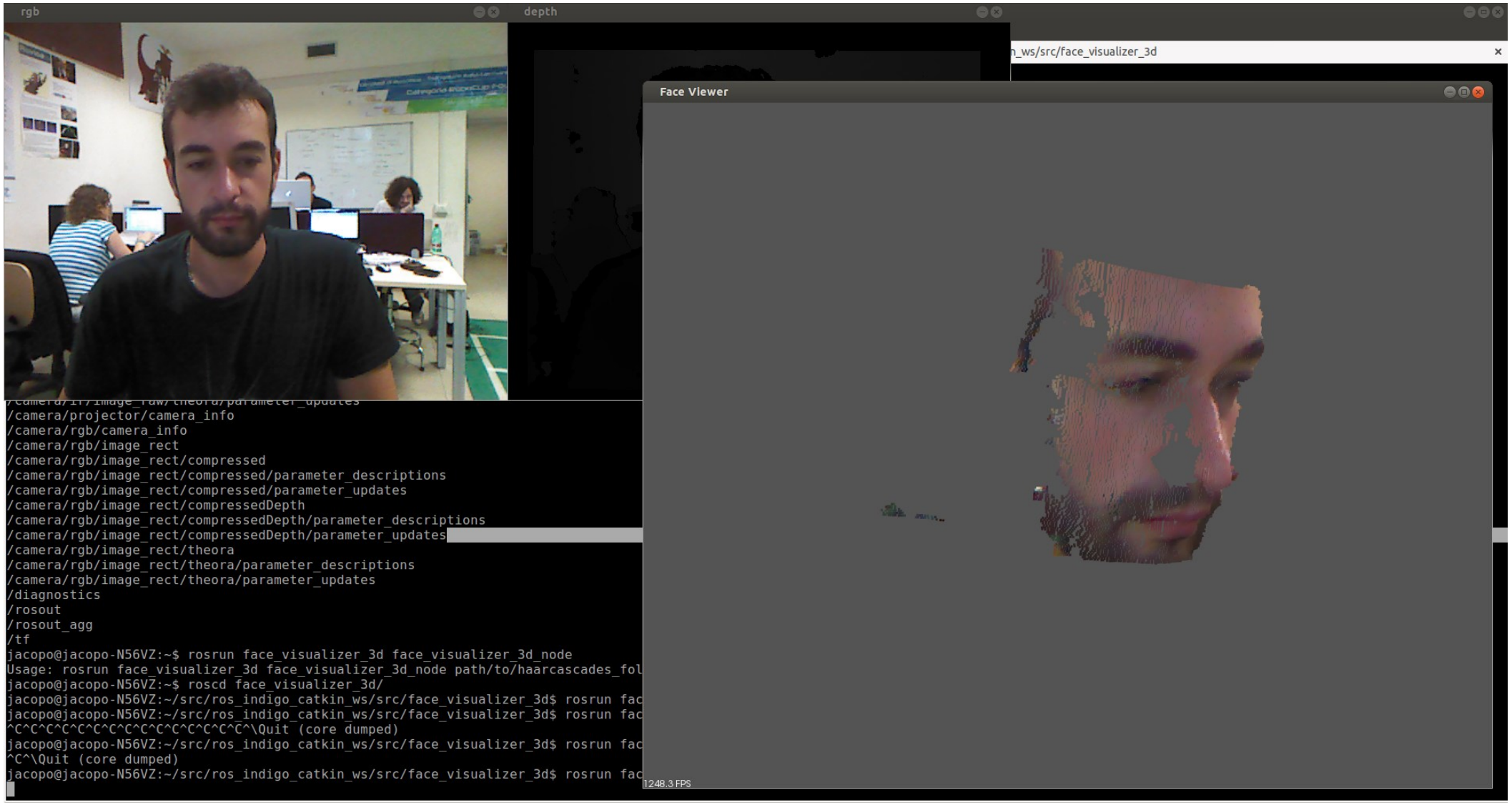
- Add/Remove a PointCloud to the Visualizer

```
pcl::visualization::PointCloudColorHandlerRGBField<pcl::PointXYZRGB> rgbHandler(face_cloud);
viewer->removePointCloud("face cloud");
viewer->addPointCloud<pcl::PointXYZRGB>(face_cloud, rgbHandler, "face cloud");
viewer->setPointCloudRenderingProperties(pcl::visualization::PCL_VISUALIZER_POINT_SIZE, 3, "face cloud");
```

Results


rgb
depth
jacopo@jacopo-N56VZ: ~/src/ros_indigo_catkin_ws
Face Viewer
/camera/ir/image_raw/theora/parameter_descriptions
/camera/ir/image_raw/theora/parameter_updates
/camera/projector/camera_info
/camera/rgb/camera_info
/camera/rgb/image_rect
/camera/rgb/image_rect/compressed
/camera/rgb/image_rect/compressed/parameter_descriptions
/camera/rgb/image_rect/compressed/parameter_updates
/camera/rgb/image_rect/compressedDepth
/camera/rgb/image_rect/compressedDepth/parameter_descriptions
/camera/rgb/image_rect/compressedDepth/parameter_updates
/camera/rgb/image_rect/theora
/camera/rgb/image_rect/theora/parameter_descriptions
/camera/rgb/image_rect/theora/parameter_updates
/diagnostics
/rosout
/rosout_agg
/tf
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun face_visualizer
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun face_visualizer
^C^\Quit (core dumped)
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun face_visualizer
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun face_visualizer
^C^\Quit (core dumped)
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun face_visualizer
380.4 FPS

Results


rgb
depth
n_ws/src/face_visualizer_3d
Face Viewer
/camera/projector/camera_info
/camera/rgb/camera_info
/camera/rgb/image_rect
/camera/rgb/image_rect/compressed
/camera/rgb/image_rect/compressed/parameter_descriptions
/camera/rgb/image_rect/compressed/parameter_updates
/camera/rgb/image_rect/compressedDepth
/camera/rgb/image_rect/compressedDepth/parameter_descriptions
/camera/rgb/image_rect/compressedDepth/parameter_updates
/camera/rgb/image_rect/theora
/camera/rgb/image_rect/theora/parameter_descriptions
/camera/rgb/image_rect/theora/parameter_updates
/diagnostics
/rosout
/rosout_agg
/tf
jacopo@jacopo-N56VZ:~$ rosrun face_visualizer_3d face_visualizer_3d_node
Usage: rosrun face_visualizer_3d face_visualizer_3d_node path/to/haarcascades_fol
jacopo@jacopo-N56VZ:~$ roscd face_visualizer_3d/
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun fac
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun fac
^C^C^C^C^C^C^C^C^C^C^C^C^C^C^C^\Quit (core dumped)
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun fac
^C^\Quit (core dumped)
jacopo@jacopo-N56VZ:~/src/ros_indigo_catkin_ws/src/face_visualizer_3d$ rosrun fac
1248.3 FPS

Esercitazione

Il codice per 3DFace Visualizer è scaricabile da
http://profs.scienze.univr.it/~bloisi/corsi/lezioni/face_visualizer_3d.zip

1. Utilizzare 3DFace Visualizer per leggere la ROS bag
   scaricabile da
   http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/face.bag

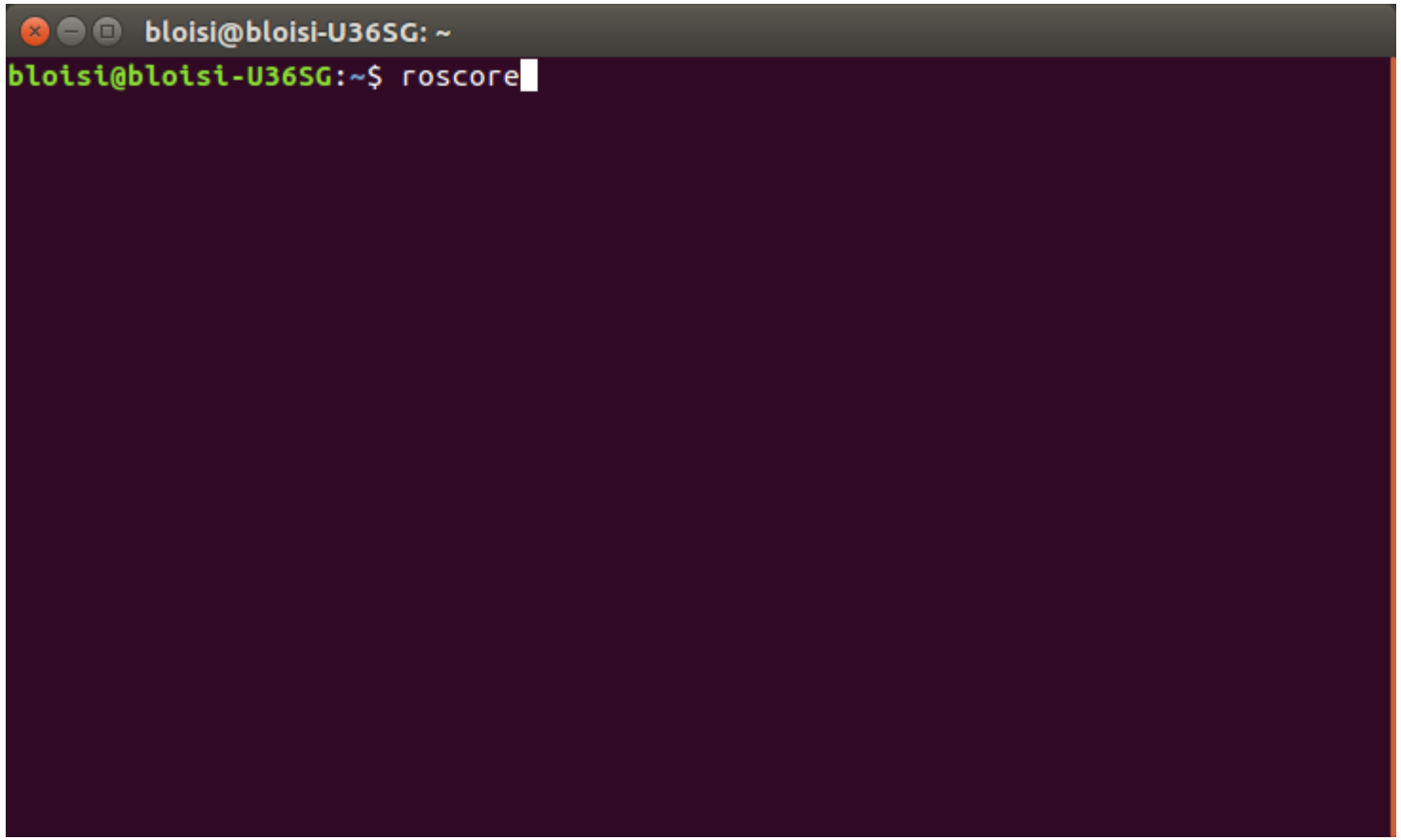
2. Visualizzare la nuvola di punti relativa al volto nelle immagini

rosbag info

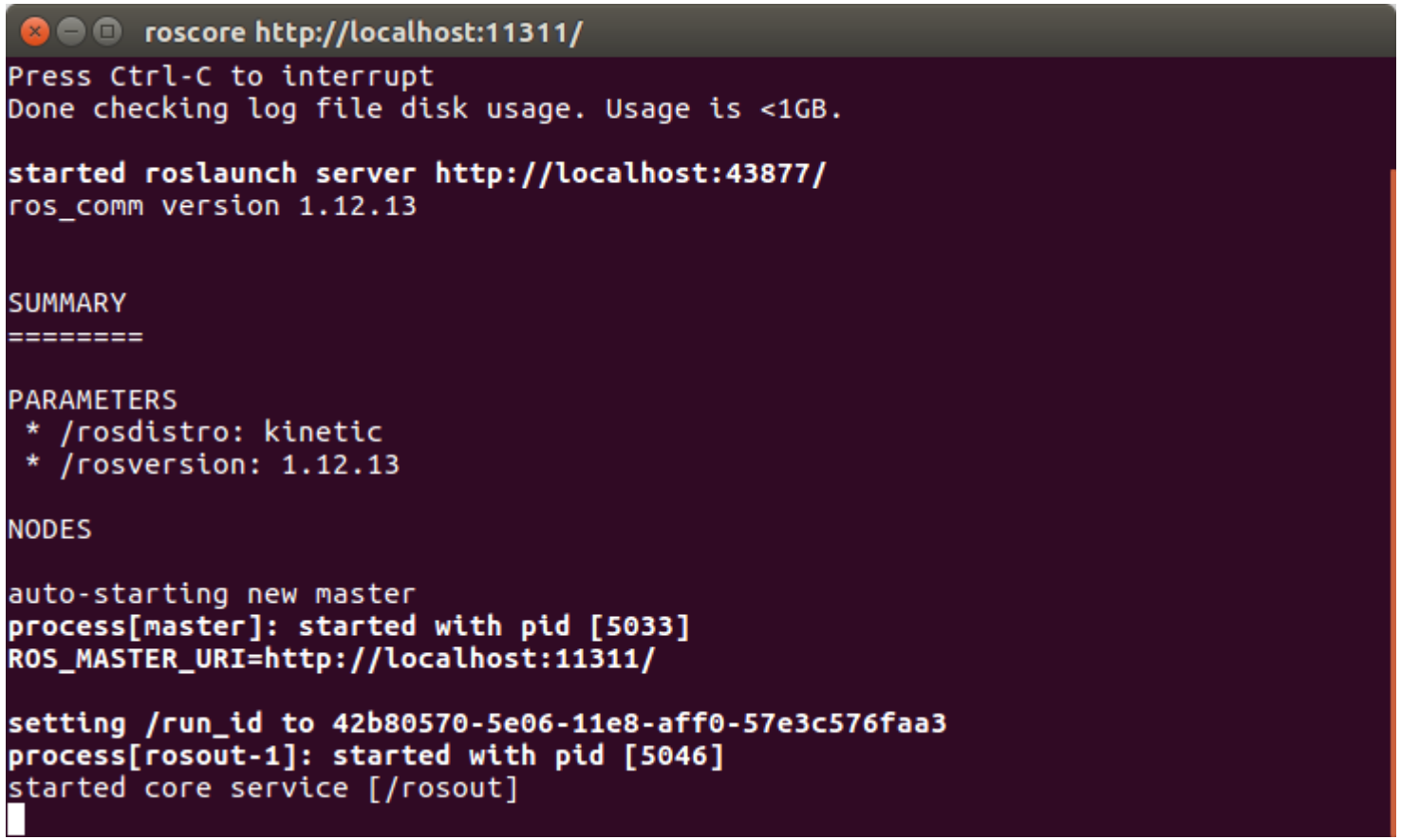
Una volta terminato il download verifichiamo che i dati nella bag siano corretti

```
bloisi@bloisi-U36SG: ~/bagfiles
bloisi@bloisi-U36SG:~/bagfiles$ rosbag info face.bag
path:        face.bag
version:     2.0
duration:    39.9s
start:       Oct 29 2014 15:01:16.62 (1414591276.62)
end:         Oct 29 2014 15:01:56.51 (1414591316.51)
size:        3.1 GB
messages:    10538
compression: none [1271/1271 chunks]
types:       diagnostic_msgs/DiagnosticArray       [60810da900de1dd6ddd437c3503511da]
             dynamic_reconfigure/Config            [958f16a05573709014982821e6822580]
             dynamic_reconfigure/ConfigDescription [757ce9d44ba8ddd801bb30bc456f946f]
             rosgraph_msgs/Log                     [acffd30cd6b6de30f120938c17c593fb]
             sensor_msgs/CameraInfo                [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/CompressedImage           [8f7a12909da2c9d3332d540a0977563f]
             sensor_msgs/Image                     [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/PointCloud2               [1158d486dd51d683ce2f1be655c3c181]
             stereo_msgs/DisparityImage            [04a177815f75271039fa21f16acad8c9]
             tf2_msgs/TFMessage                    [94810edda583a504dfda3829e70d7eec]
             theora_image_transport/Packet         [33ac4e14a7cff32e7e0d65f18bb410f3]
topics:      /camera/debayer/parameter_descriptions                                    1
 msg     : dynamic_reconfigure/ConfigDescription
             /camera/debayer/parameter_updates                                        1
 msg     : dynamic_reconfigure/Config
             /camera/depth/image/compressed/parameter_descriptions                    1
 msg     : dynamic_reconfigure/ConfigDescription
             /camera/depth/image/compressed/parameter_updates                         1
 msg     : dynamic_reconfigure/Config
             /camera/depth/image/compressedDepth/parameter_descriptions               1
 msg     : dynamic_reconfigure/ConfigDescription
             /camera/depth/image/compressedDepth/parameter_updates                    1
```

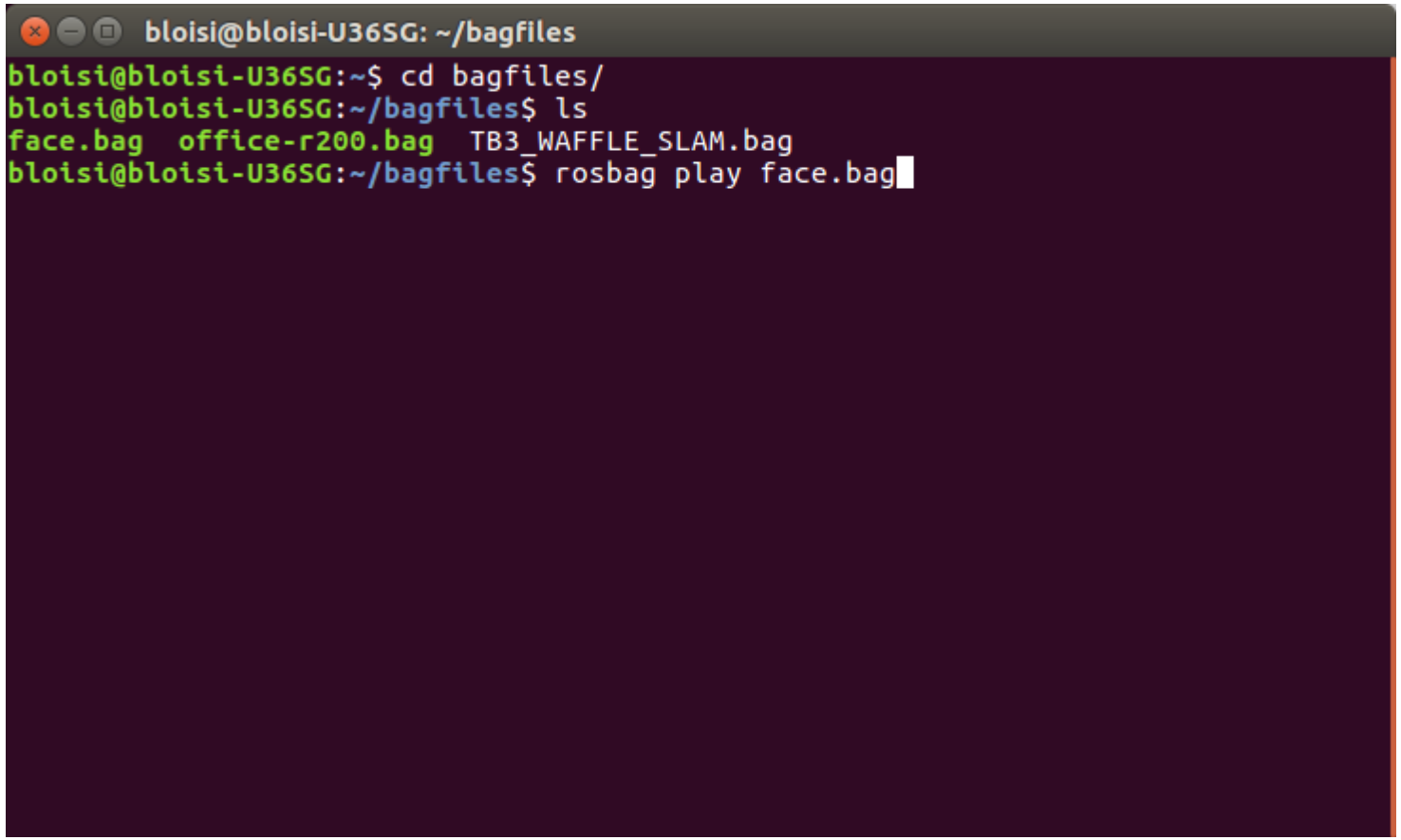
roscore



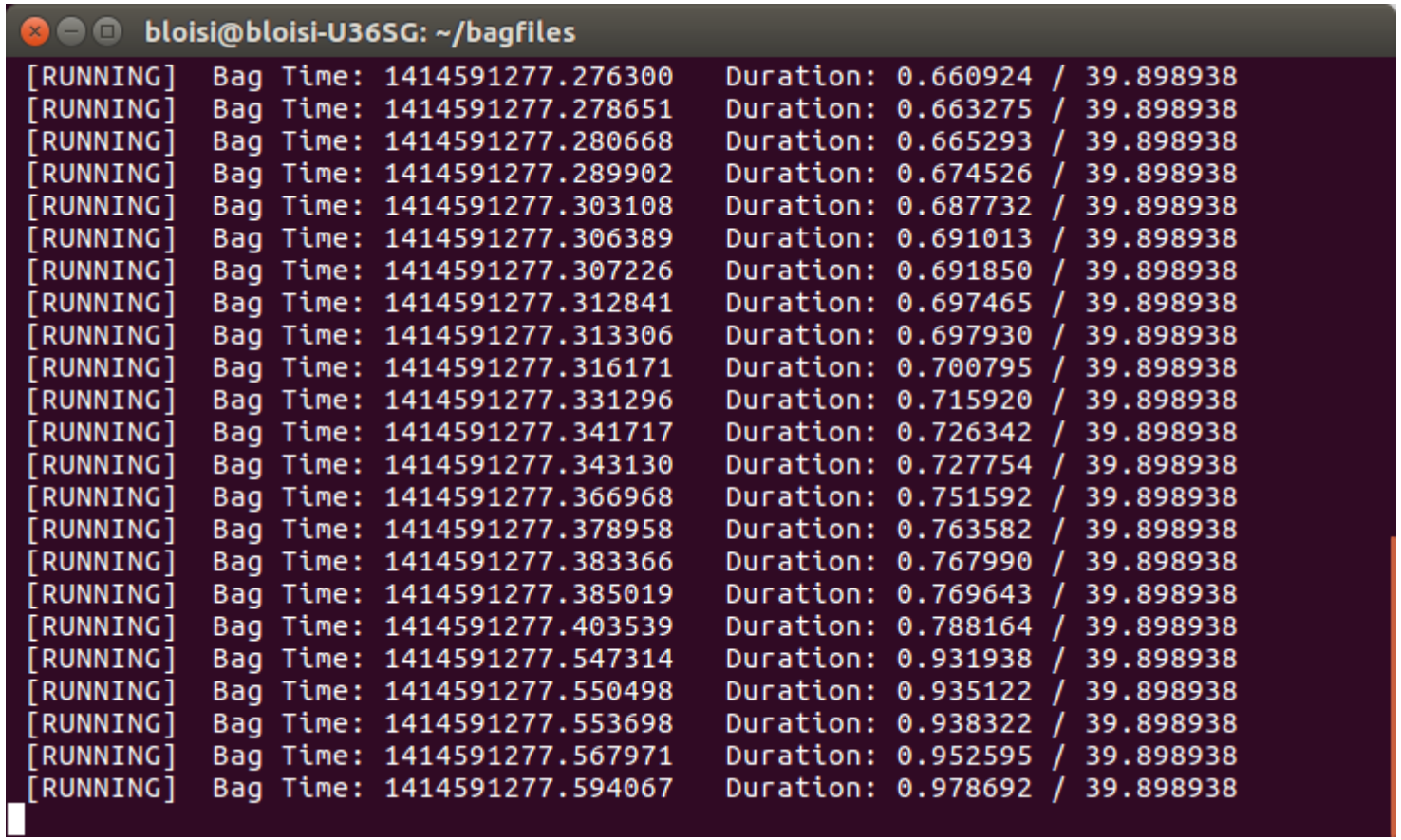
roscore - esecuzione



rosbag play


bloisi@bloisi-U36SG: ~/bagfiles
bloisi@bloisi-U36SG:~$ cd bagfiles/
bloisi@bloisi-U36SG:~/bagfiles$ ls
face.bag office-r200.bag TB3_WAFFLE_SLAM.bag
bloisi@bloisi-U36SG:~/bagfiles$ rosbag play face.bag

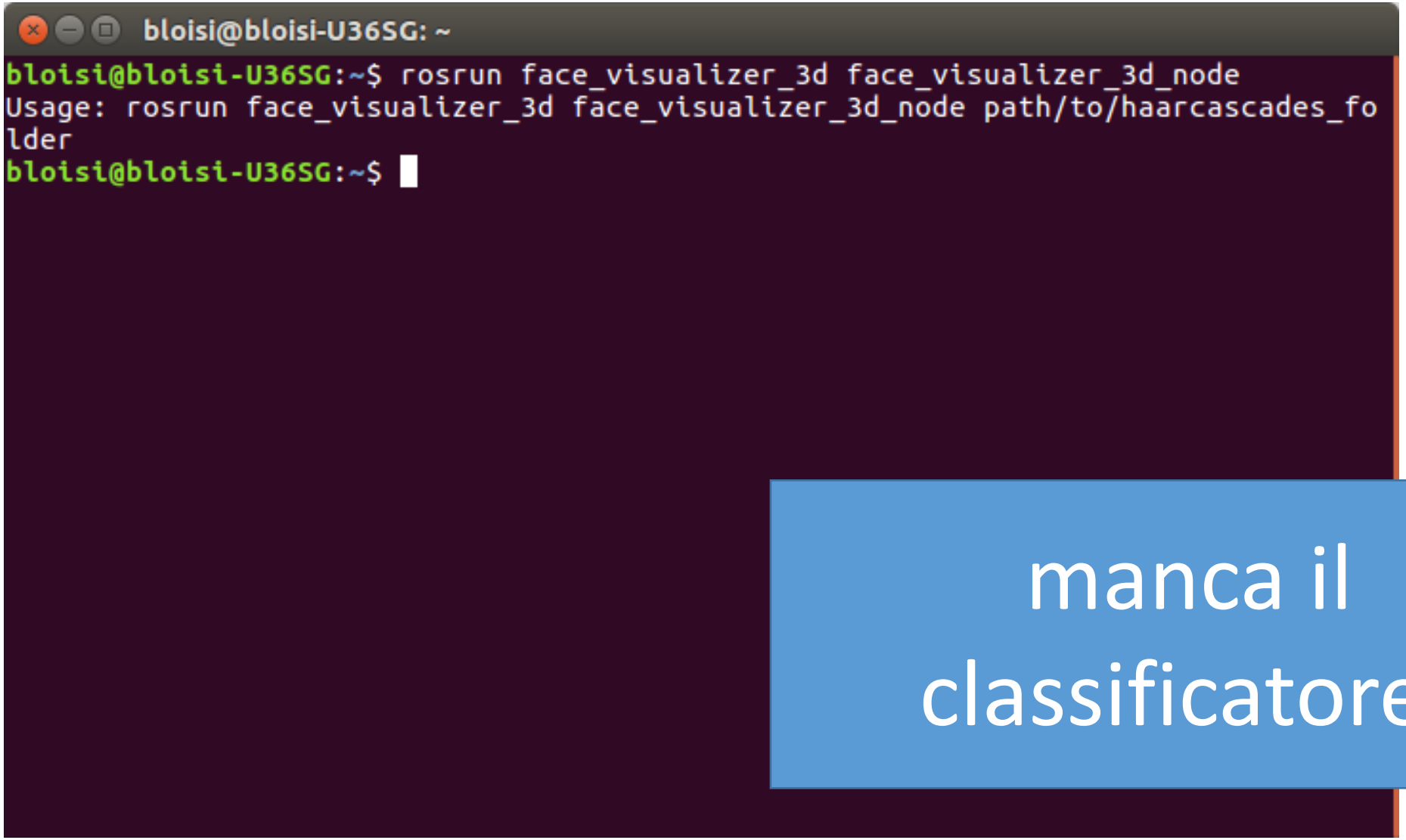
rosbag play - esecuzione



rosrun

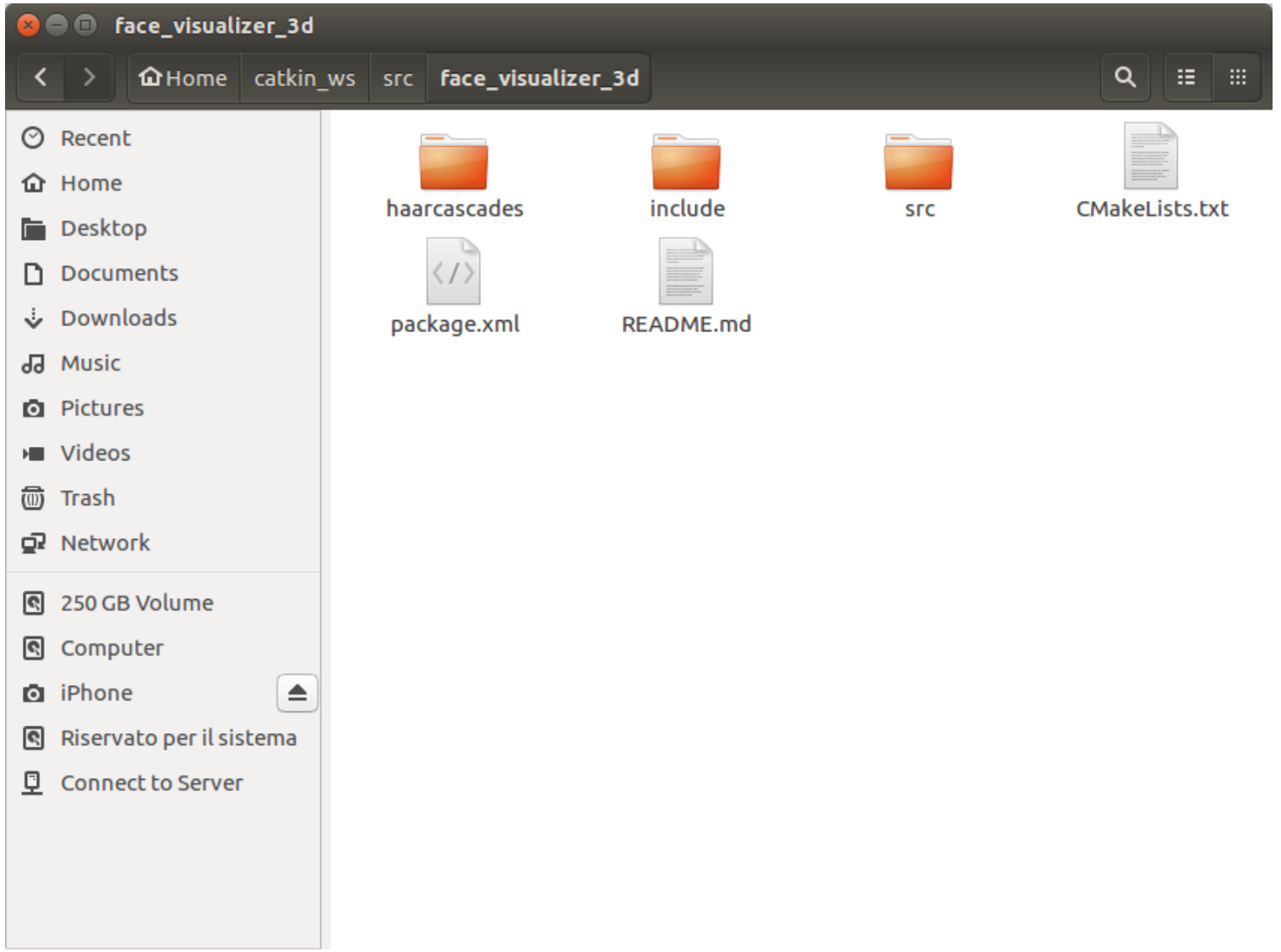

bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun face_visualizer_3d face_visualizer_3d_node

rosrun - esecuzione

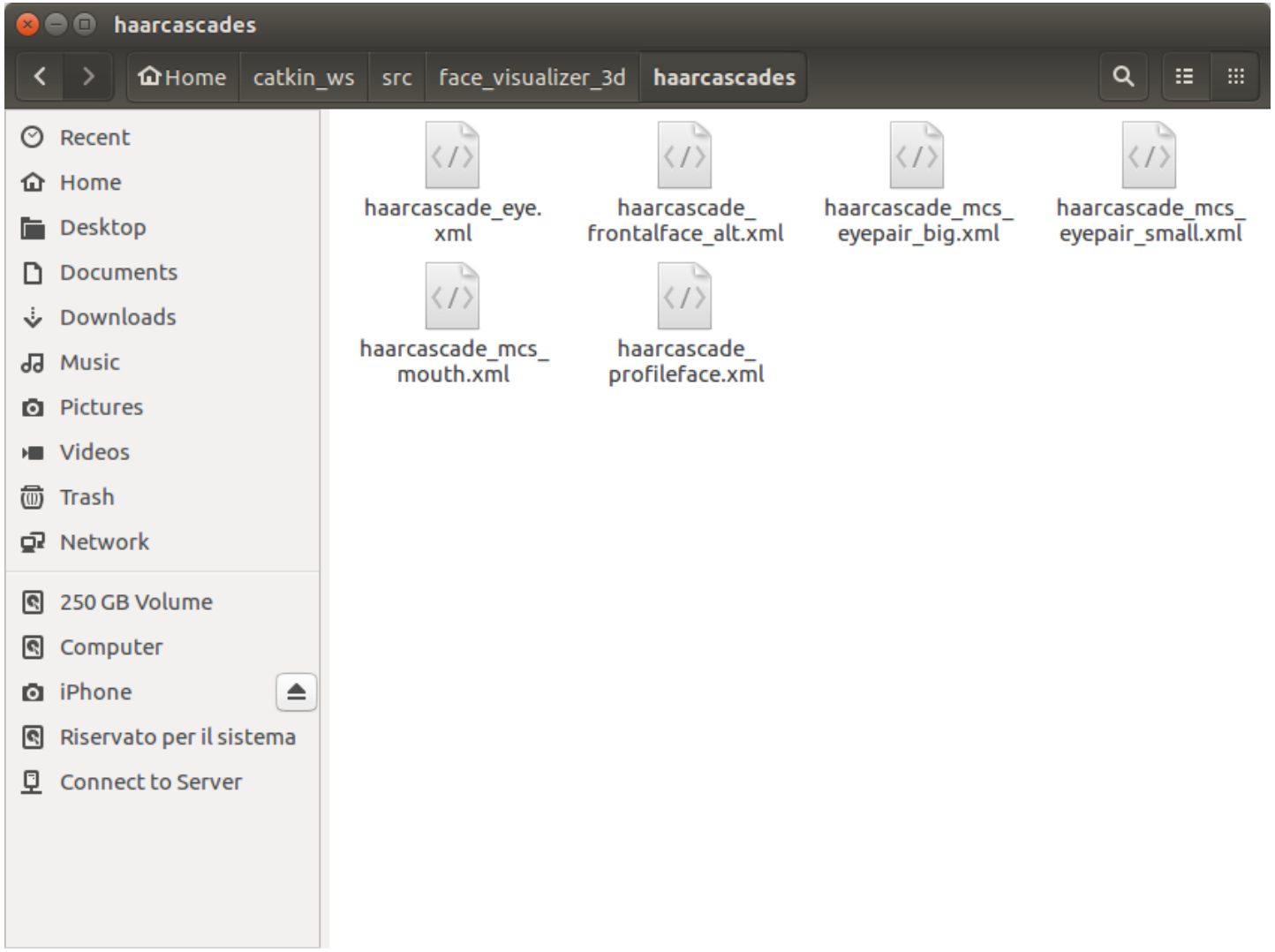


manca il classificatore!

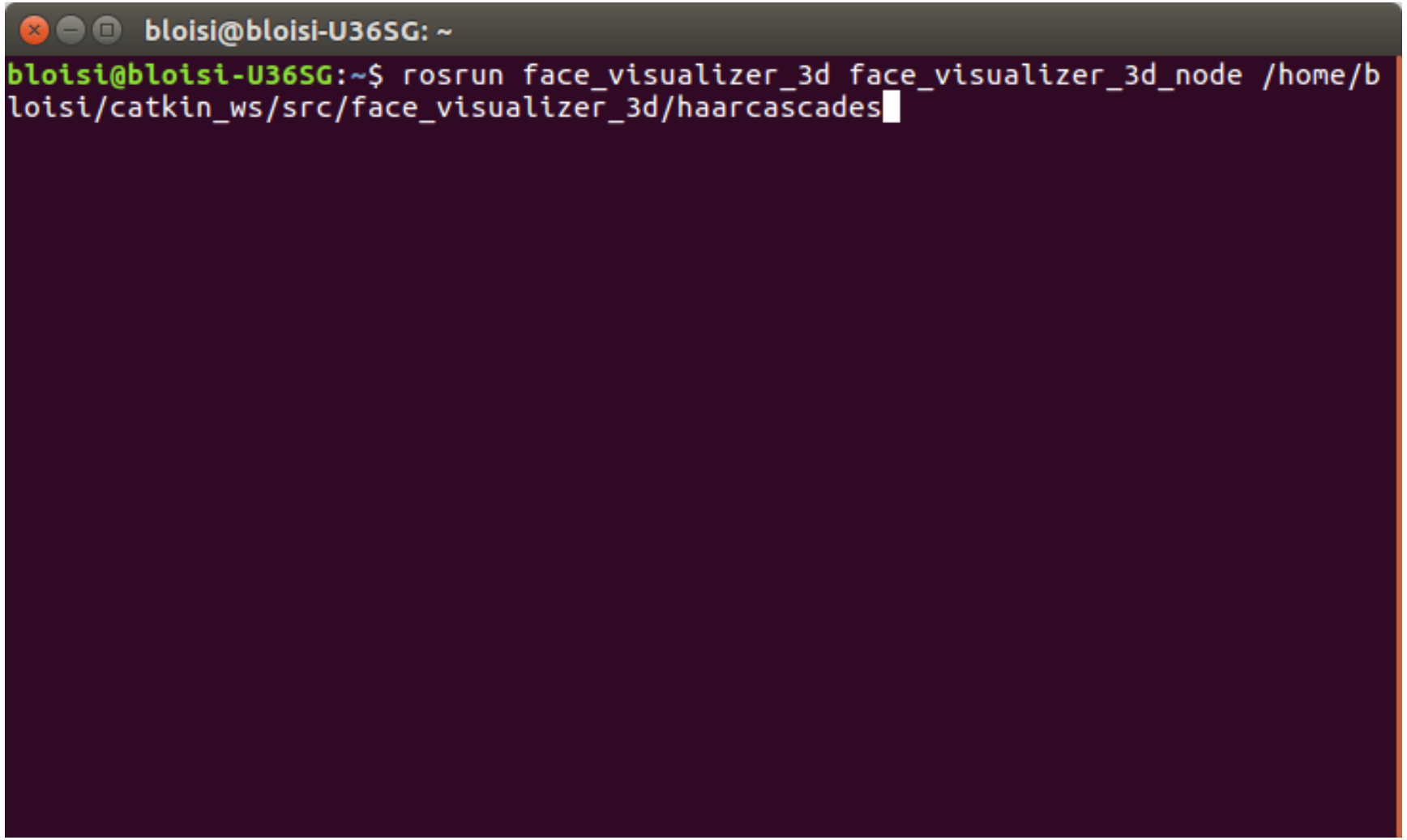
Classificatore



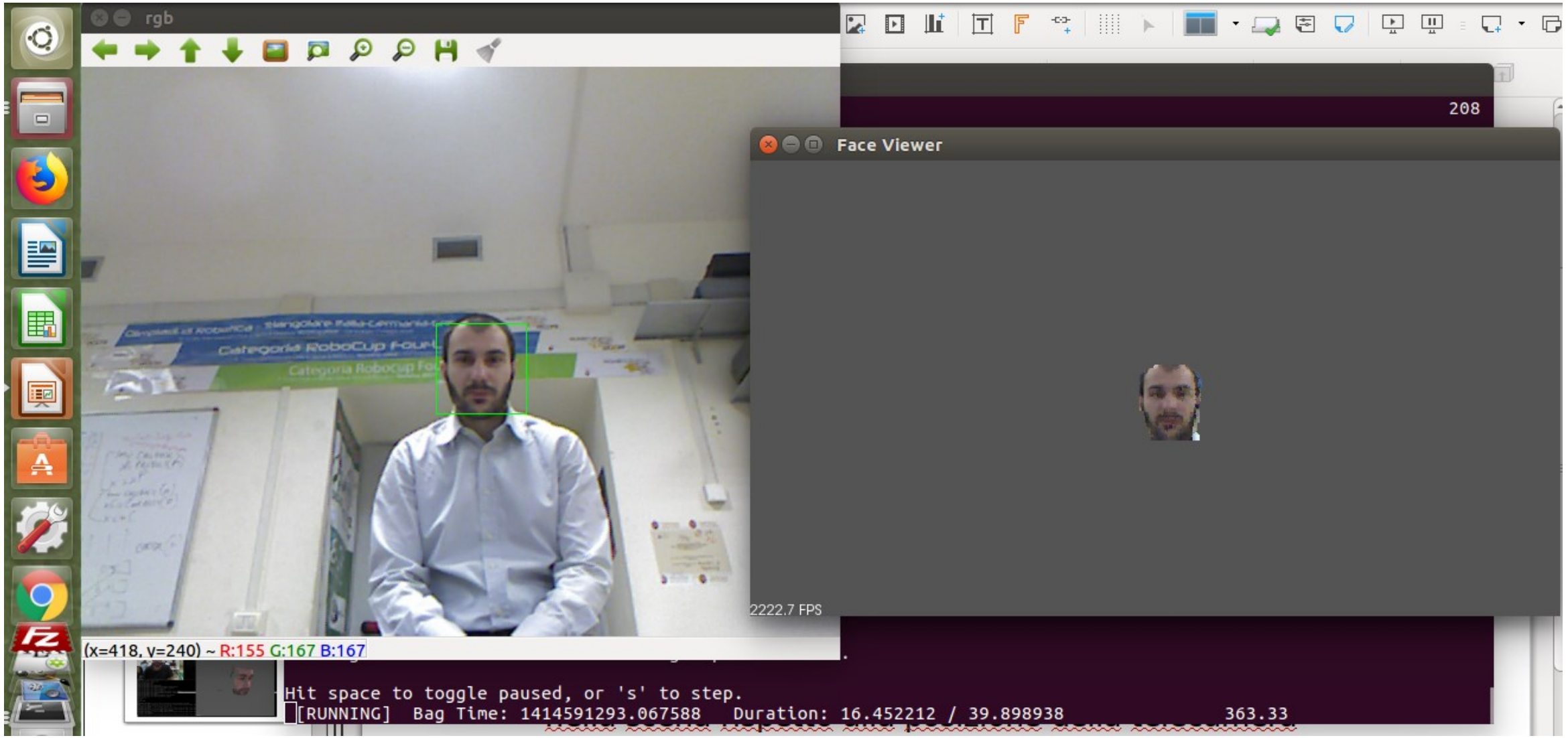
Classificatori forniti da OpenCV



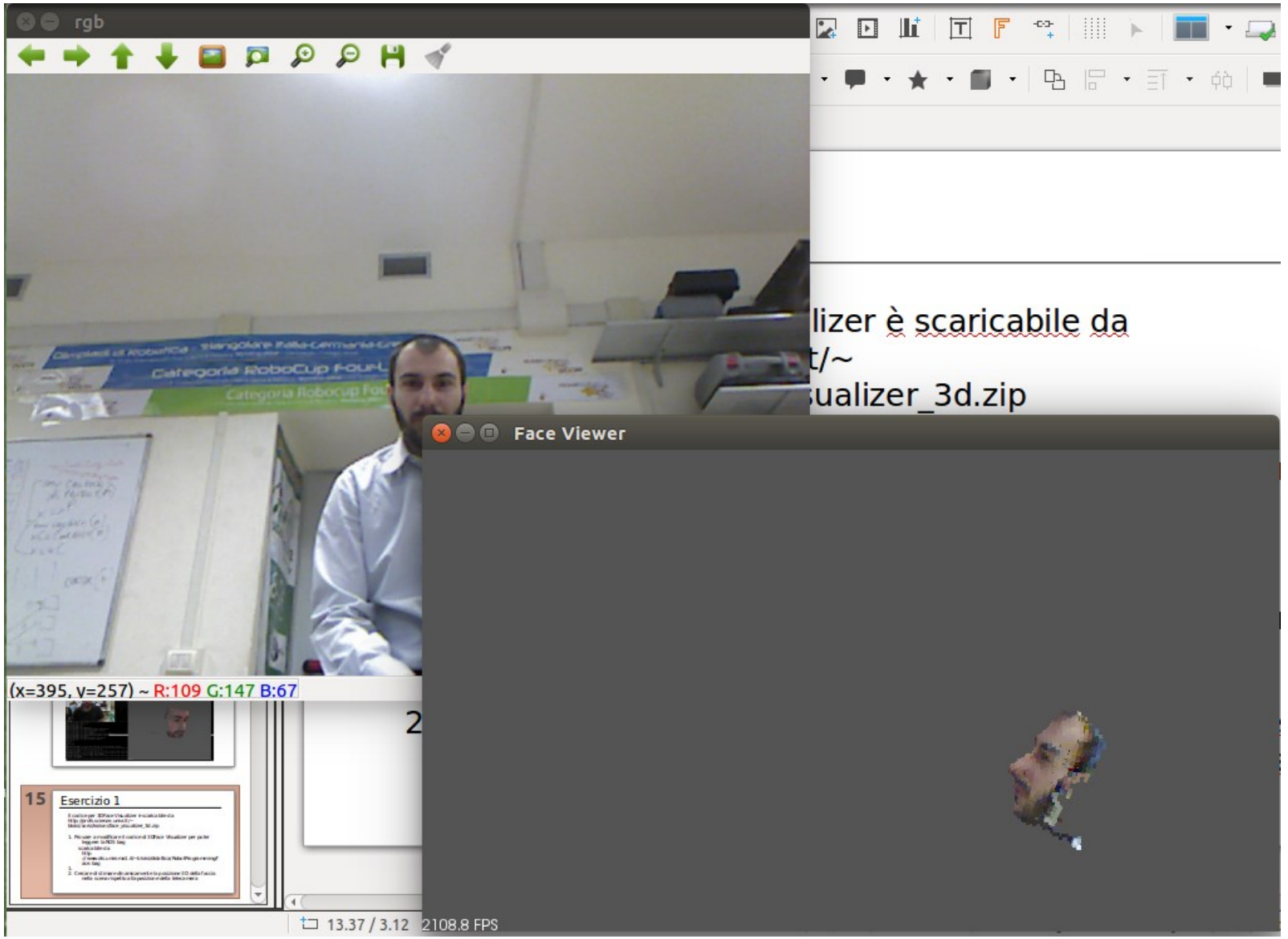
rosrun corretto con classificatore



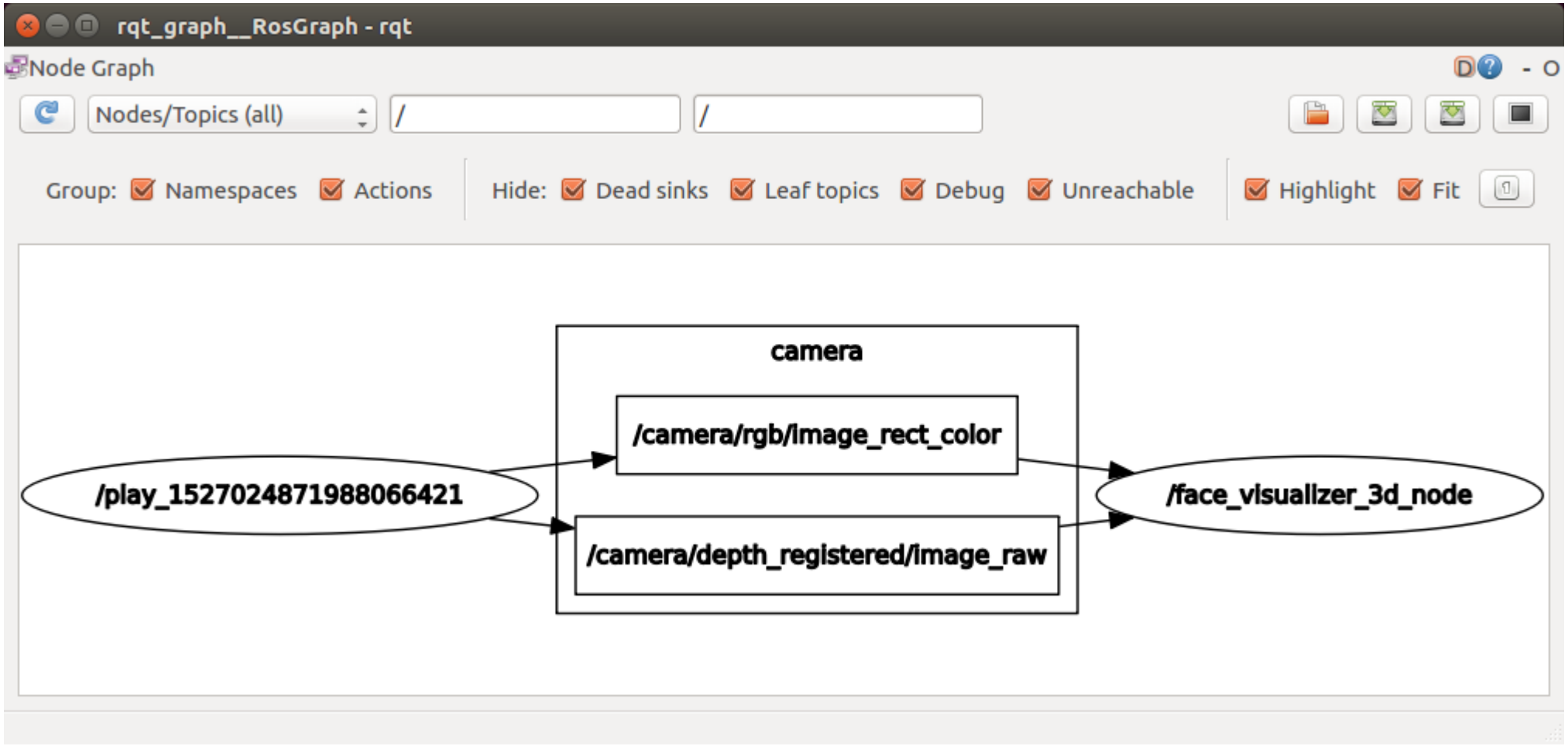
esecuzione


rgb
Face Viewer
208
2222.7 FPS
(x=418, y=240) ~ R:155 G:167 B:167
Hit space to toggle paused, or 's' to step.
[RUNNING] Bag Time: 1414591293.067588 Duration: 16.452212 / 39.898938
363.33

Visualizzazione 3D



rqt_graph


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
/
/
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit
camera
/camera/rgb/image_rect_color
/play_1527024871988066421
/face_visualizer_3d_node
/camera/depth_registered/image_raw

Esercizio

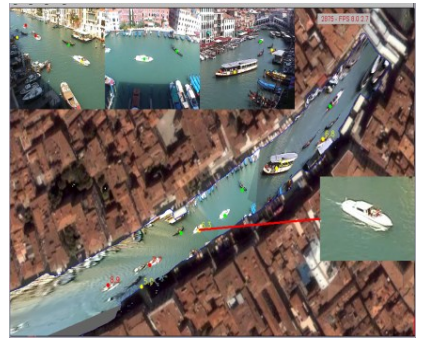
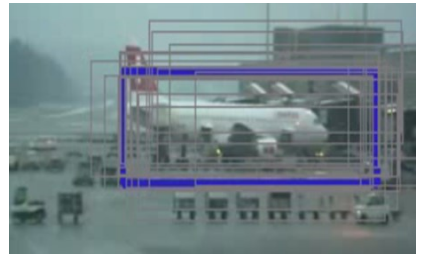
1. Provare a modificare il codice di 3DFace Visualizer per poter leggere la ROS bag scaricabile da http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/people.bag

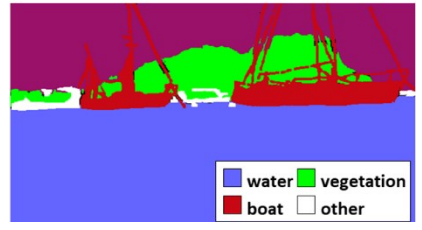
2. Cercare di estrarre oltre al volto anche il corpo delle persone nella bag

UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA





Corso di Laboratorio Ciberfisico

Modulo di Robot Programming with ROS

Esercitazione

3d face visualization

Docente:
Domenico Daniele Bloisi

Maggio 2018