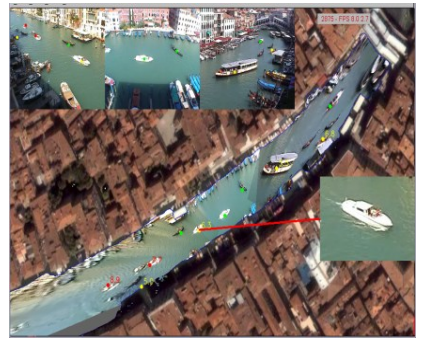
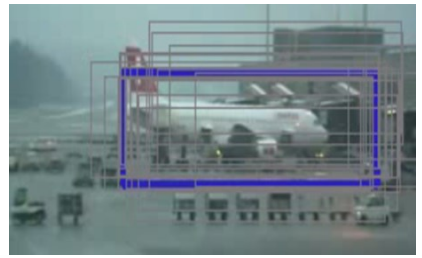
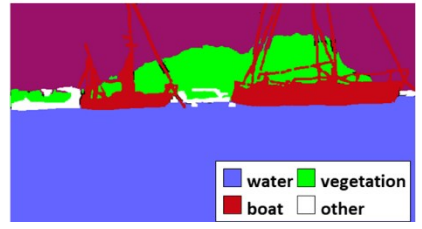
UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

Laurea magistrale in Ingegneria e scienze informatiche





Esercitazione

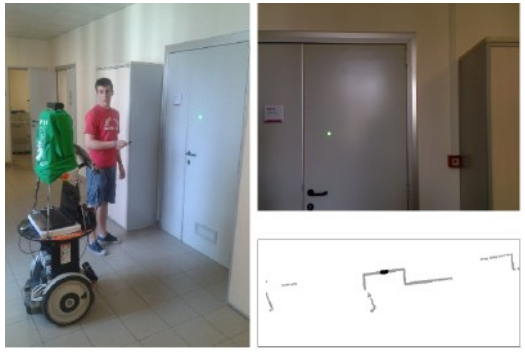
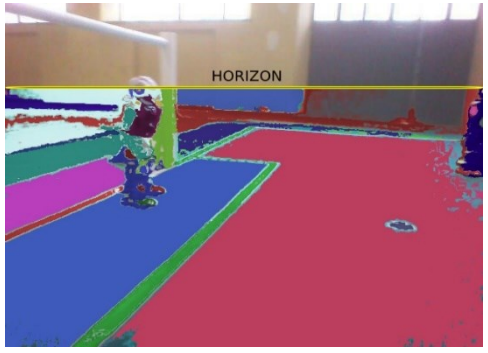
3d face visualization

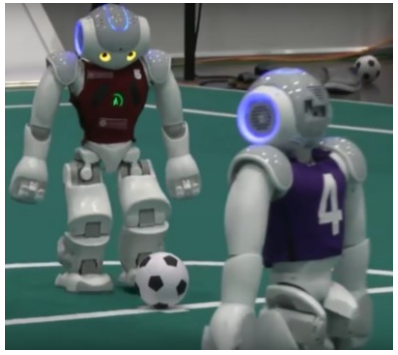
Corso di Robotica

Parte di Laboratorio

Docente:

Domenico Daniele Bloisi




Novembre 2017

Esercitazione

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)

OpenCV

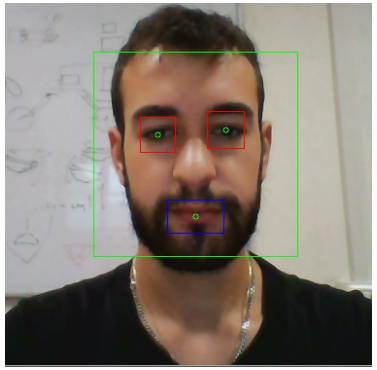
pcl

ROS

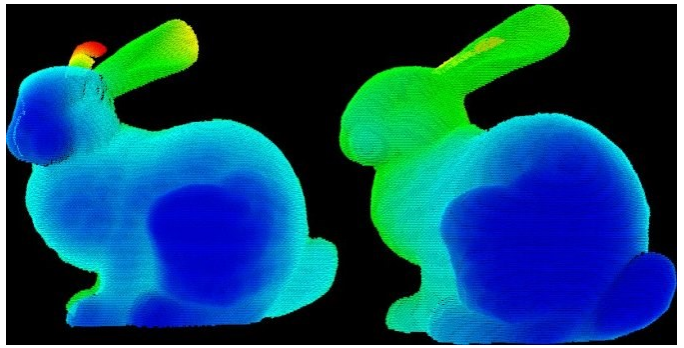
2D + Depth + 3D

- 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

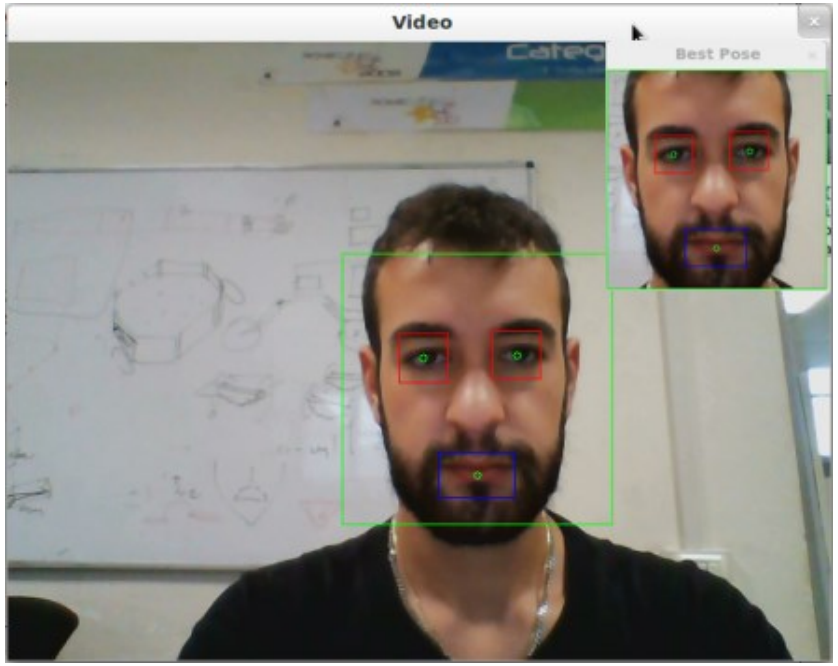
```
#include <message_filters/subscriber.h>
#include <message_filters/synchronizer.h>
#include <message_filters/sync_policies/approximate_time.h>

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

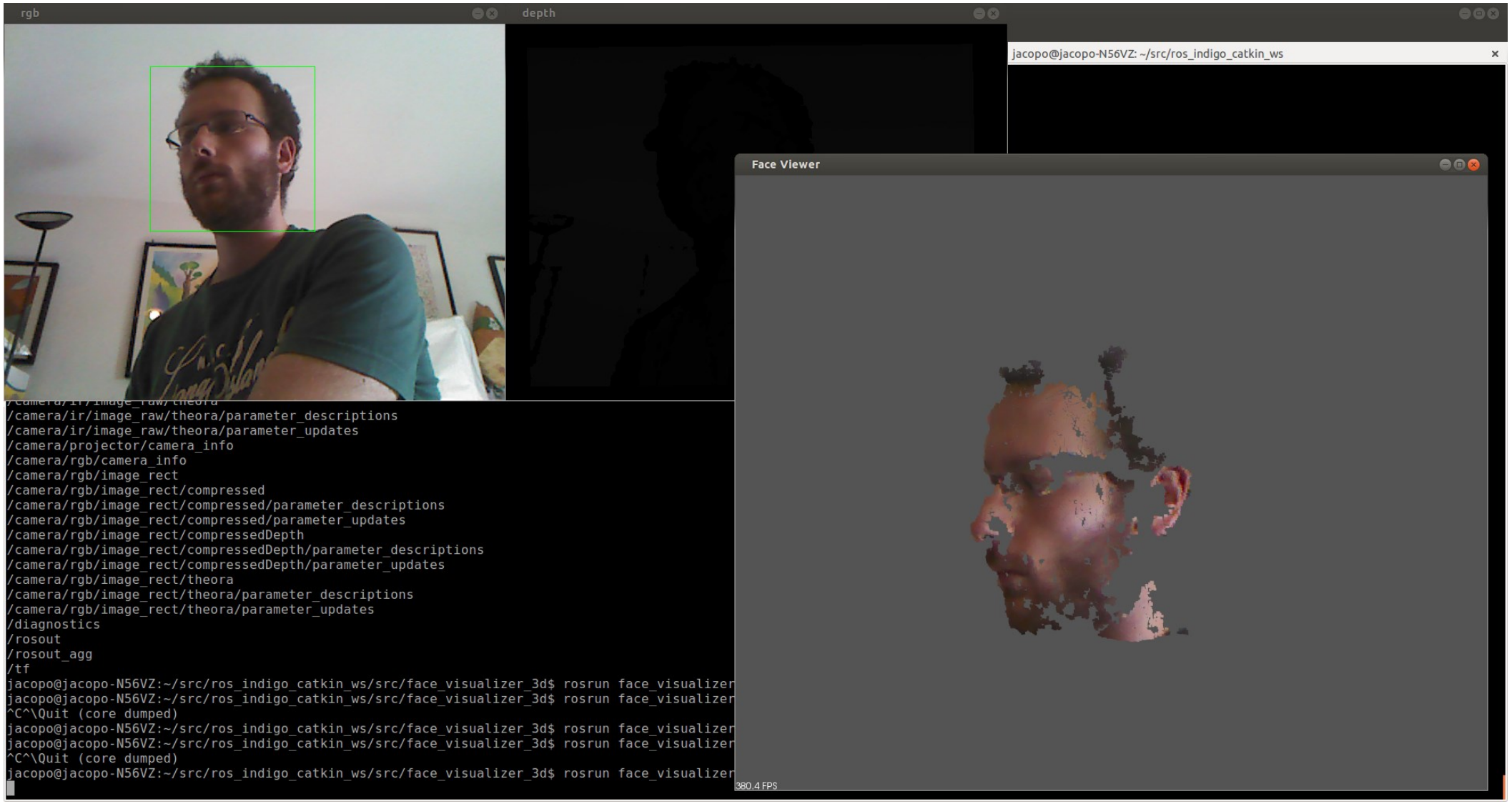
```
#include <pcl/visualization/pcl_visualizer.h>

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);
```

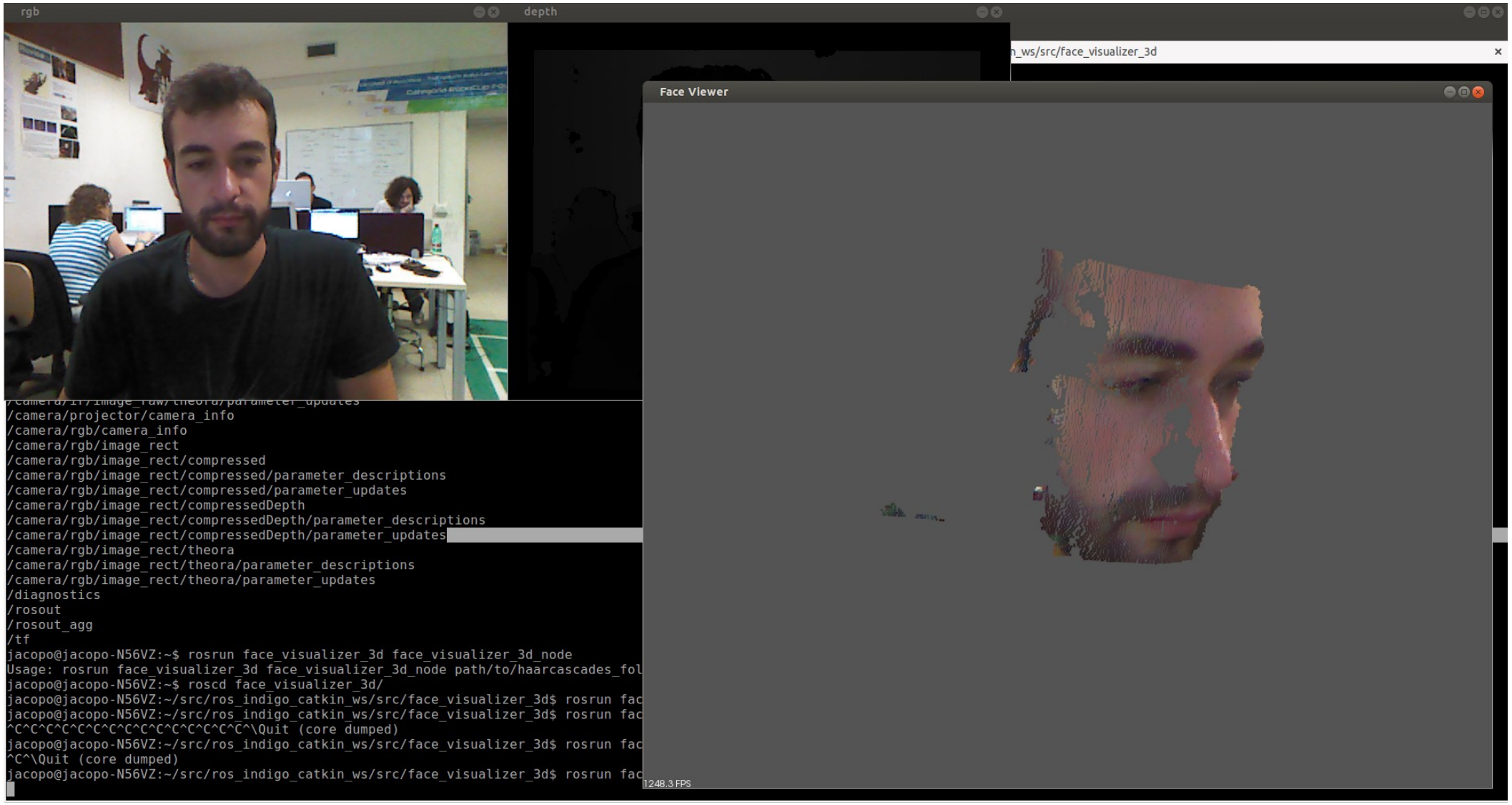
- Add/Remove a PointCloud to the Visualizer

```
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

Esercizio

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

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/face.bag

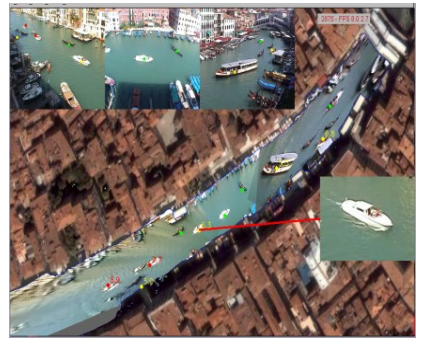
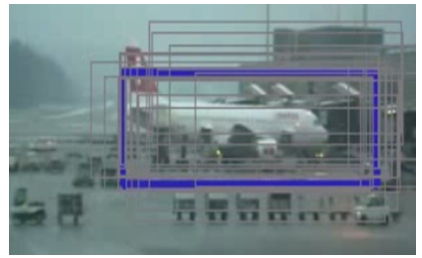
2. Cercare di stimare dinamicamente la posizione 3D della faccia nella scena rispetto alla posizione della telecamera

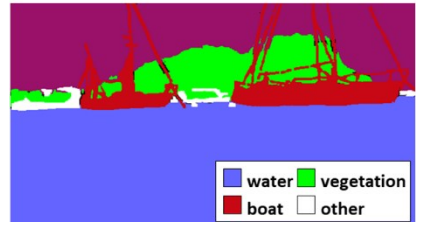
UNIVERSITÀ di VERONA

Dipartimento di INFORMATICA

Laurea magistrale in Ingegneria e scienze informatiche





Esercitazione

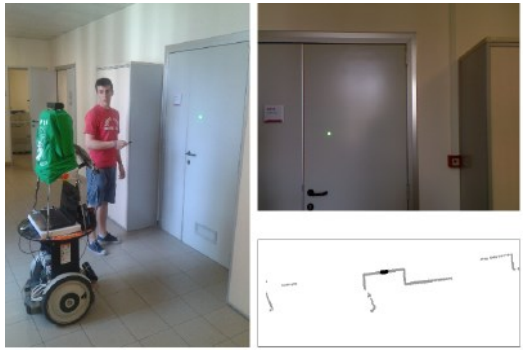
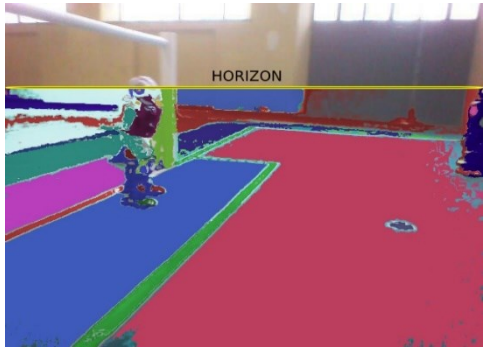
3d face visualization

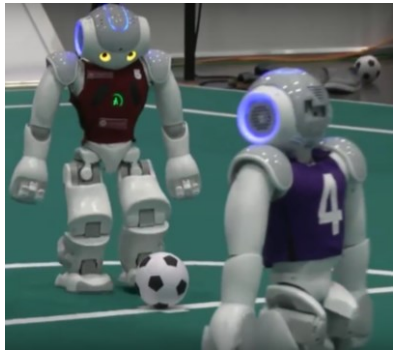
Corso di Robotica

Parte di Laboratorio

Docente:

Domenico Daniele Bloisi




Novembre 2017