Robot Programming

Section of Elective in Artificial Intelligence

Master Artificial Intelligence and Robotics

A.A. 2014/2015

DEPARTMENT OF COMPUTER, CONTROL, AND MANAGEMENT ENGINEERING ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Image Processing with OpenCV

Domenico Daniele Bloisi
Andrea Pennisi

Outline

- Introduction to OpenCV
- Kinect sensor
- Kinect data acquisition in ROS
- Kinect demo (ROS+OpenCV)
- Background subtraction demo
- Point clouds
- Face Detection with Viola and Jones
- Face detection demo (ROS+OpenCV+PCL)

Contact

Domenico Daniele Bloisi, PhD
Assistant Professor

Location: DIAG A209

bloisi@dis.uniroma1.it
http://www.dis.uniroma1.it/~bloisi

Introduction to OpenCV

OpenCV

http://opencv.org/

- OpenCV (Open Source Computer Vision) is a library of programming functions for realtime computer vision.
- BSD Licensed - free for commercial use
- C++, C, Python and Java (Android) interfaces
- Supports Windows, Linux, Android, iOS and Mac OS
- More than 2500 optimized algorithms

OpenCV Modules

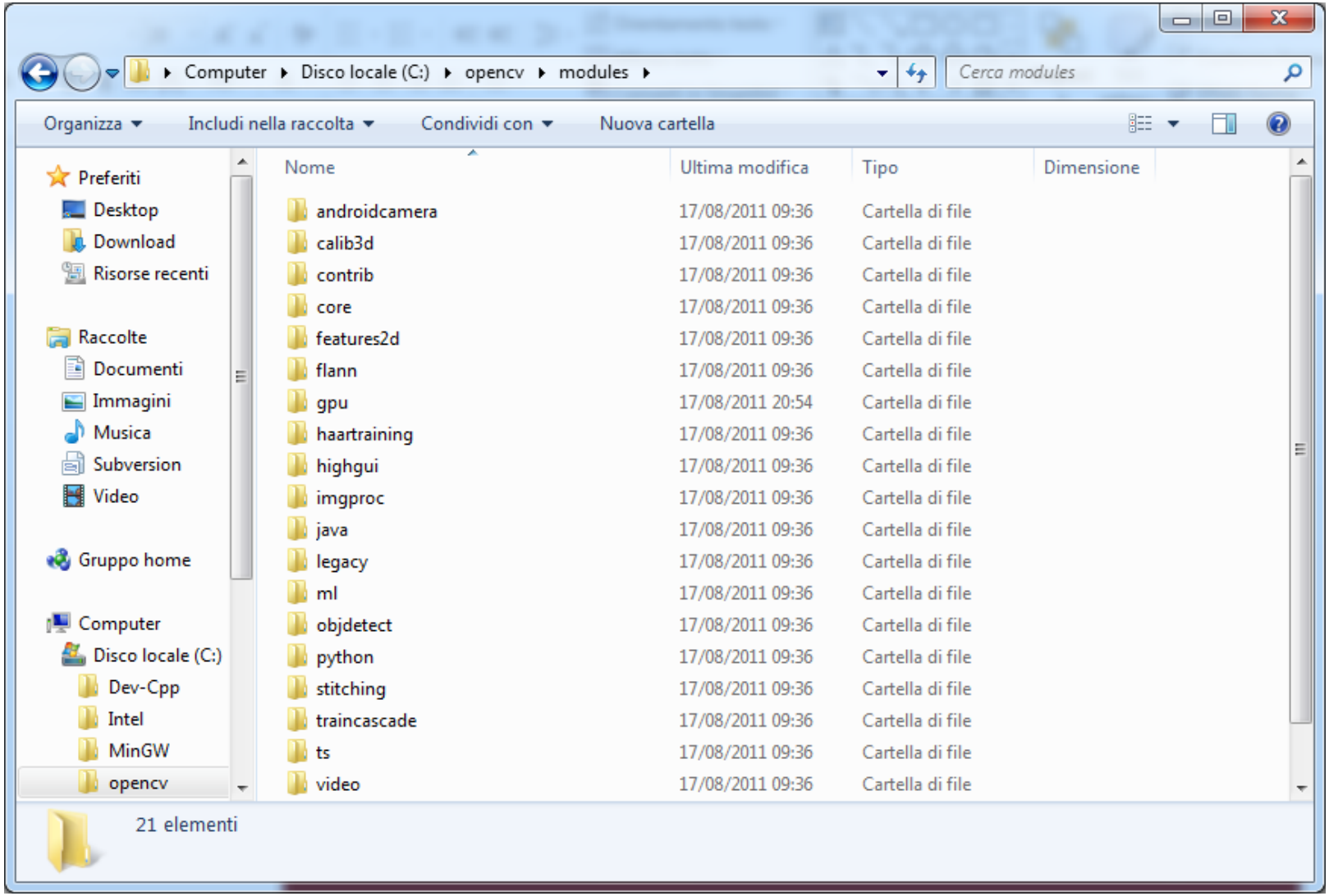
OpenCV has a modular structure

- **core**
- **imgproc**
- **video**
- **calib3d**
- **features2d**
- **objdetect**
- **highgui**
- **gpu**

http://opencv.org/documentation.html

http://docs.opencv.org/2.4.7/modules/core/doc/intro.html

OpenCV File Structure



Modules for Image Processing

core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

imgproc - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

Modules for Image Processing

features2d - salient feature detectors, descriptors, and descriptor matchers.

highgui - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.

How to include modules

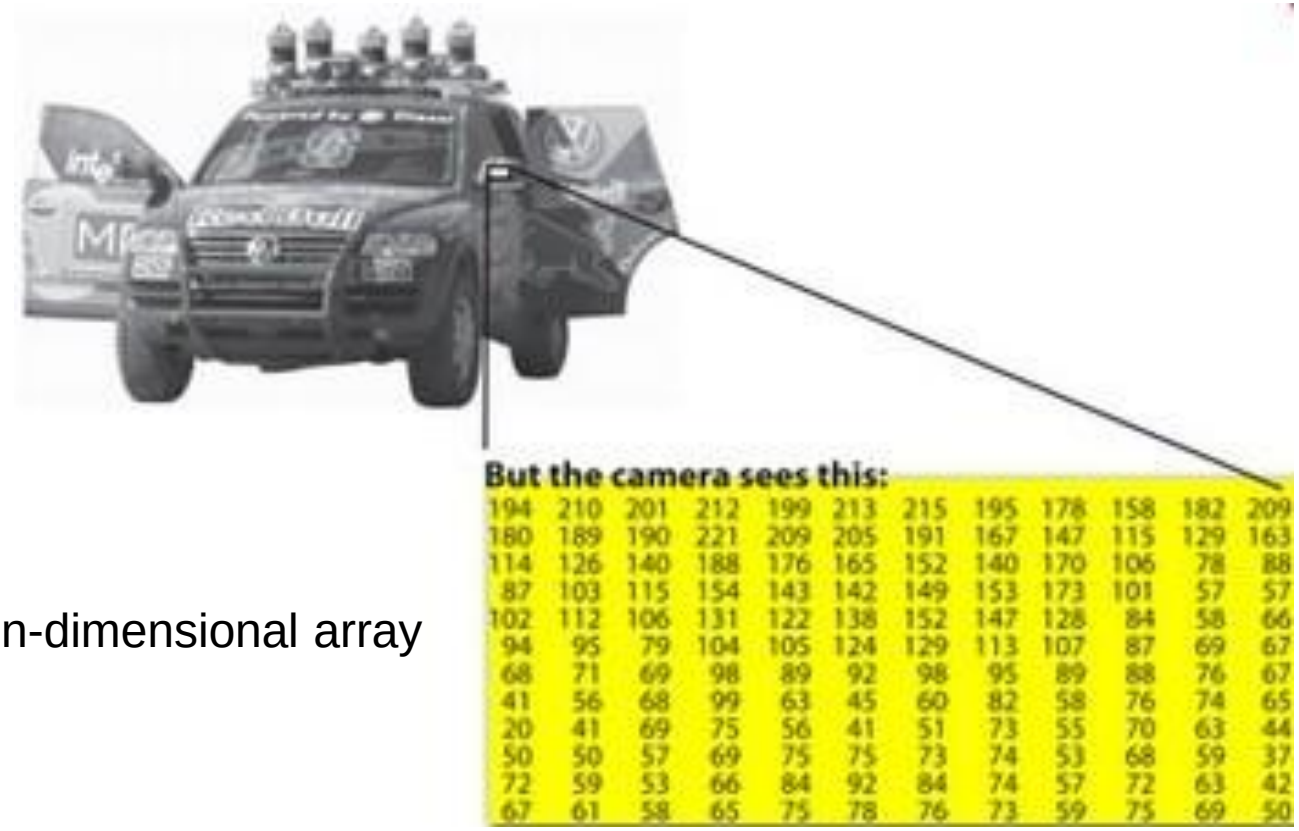
```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/features2d/features2d.hpp>
```

Data types

Set of primitive data types the library can operate on

- `uchar`: 8-bit unsigned integer
- `schar`: 8-bit signed integer
- `ushort`: 16-bit unsigned integer
- `short`: 16-bit signed integer
- `int`: 32-bit signed integer
- `float`: 32-bit floating-point number
- `double`: 64-bit floating-point number

Image representation in OpenCV: Mat



A Mat is a n-dimensional array

http://docs.opencv.org/modules/core/doc/basic_structures.html#mat

Mat (Header vs Data)

```
// creates just the header parts
Mat A, C;
// here we'll know the method used (allocate matrix)
A = imread(argv[1], CV_LOAD_IMAGE_COLOR);
// Use the copy constructor (copy by reference)
Mat B(A);
// Assignment operator (copy by reference)
C = A;
// creates a new matrix D with data copied from A
Mat D = A.clone();
// creates the header for E with no data
Mat E;
//sets the data for E (copied from A)
A.copyTo(E);
```

http://docs.opencv.org/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html#matthebasicimagecontainer

Creating a Mat object

```
Mat M(2,2, CV_8UC3, Scalar(0,0,255));
cout << "M = " << endl << " " << M << endl << endl;
```

```
M =
 [0, 0, 255, 0, 0, 255;
  0, 0, 255, 0, 0, 255]
```

```
Mat M;
M.create(4,4, CV_8UC(2));
M = Scalar(205,205);
cout << "M = "<< endl << " "  << M << endl << endl;
```

```
M =
 [205, 205, 205, 205, 205, 205, 205, 205;
  205, 205, 205, 205, 205, 205, 205, 205;
  205, 205, 205, 205, 205, 205, 205, 205;
  205, 205, 205, 205, 205, 205, 205, 205]
```

MATLAB style initializer

```
Mat E = Mat::eye(4, 4, CV_64F);
cout << "E = " << endl << " " << E << endl << endl;

Mat Z = Mat::zeros(3,3, CV_8UC1);
cout << "Z = " << endl << " " << Z << endl << endl;

Mat O = Mat::ones(2, 2, CV_32F);
cout << "O = " << endl << " " << O << endl << endl;
```

How the image matrix is stored in the memory?

	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row ...	...,0	...,1	...	..., m
Row n	n,0	n,1	n,...	n, m

gray scale image

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

BGR image

Note that the order of the channels in OpenCV is BGR instead of RGB.

How to scan gray scale images

```
cv::Mat I = ...

...

for( int i = 0; i < I.rows; ++i) {
   for( int j = 0; j < I.cols; ++j) {
      uchar g = I.at<uchar>(i,j);
      ...
    }
}
```

How to scan RGB images

```
cv::Mat I = ...

...

for( int i = 0; i < I.rows; ++i) {
   for( int j = 0; j < I.cols; ++j) {
      uchar blue = I.at<cv::Vec3b>(i,j)[0];
      uchar green = I.at<cv::Vec3b>(i,j)[1];
      uchar red = I.at<cv::Vec3b>(i,j)[2];
      ...
   }
}
```

Kinect Sensor

Kinect Data Acquisition in ROS

What we need

ROS Indigo Igloo

KINECT™

Version 1.0 or 1.1

OpenCV

Version 2.4.9

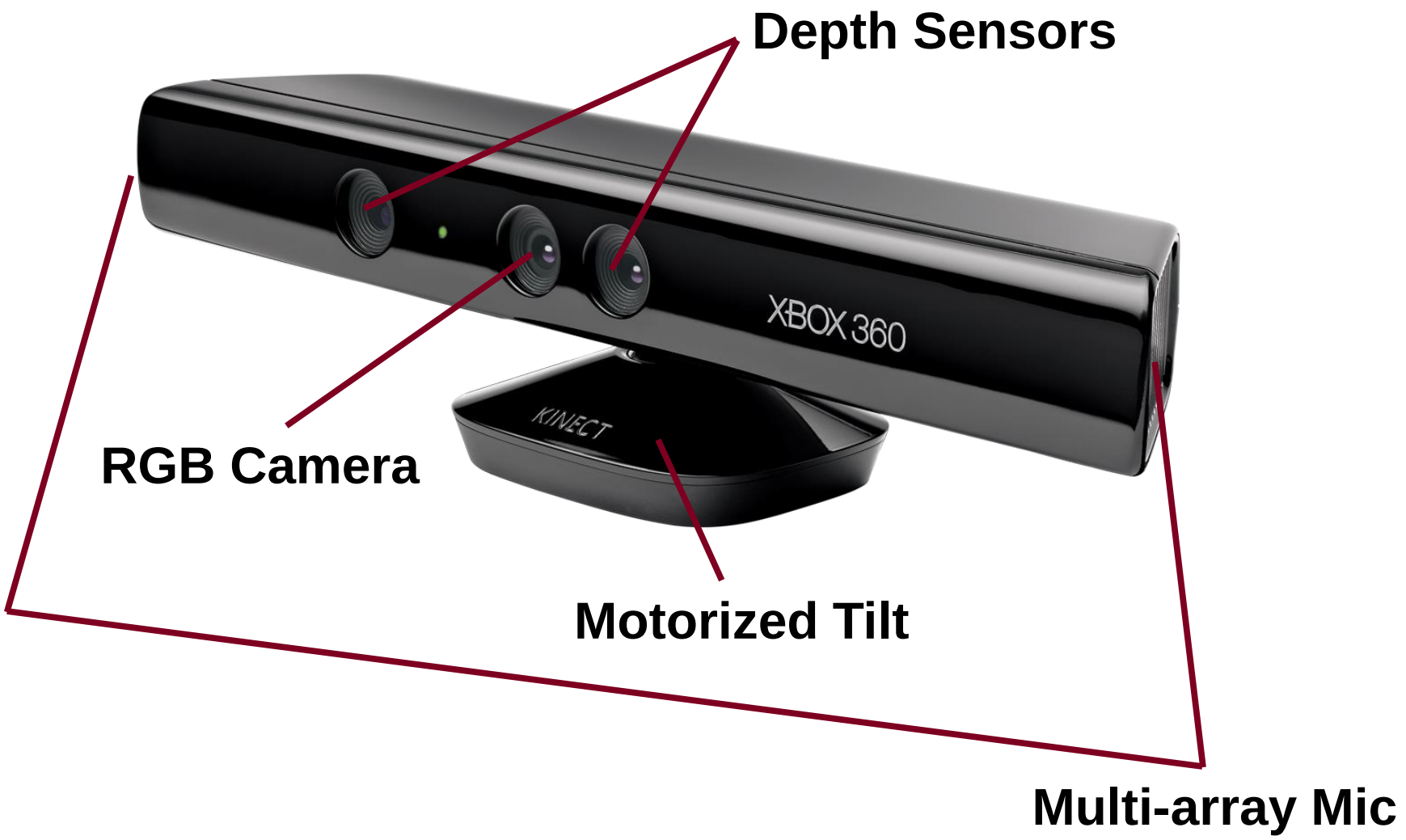
Microsoft Kinect

- Kinect was launched in North America on November 4th, 2010
- Kinect is a motion sensing input device

v1.0

v1.1

Microsoft Kinect

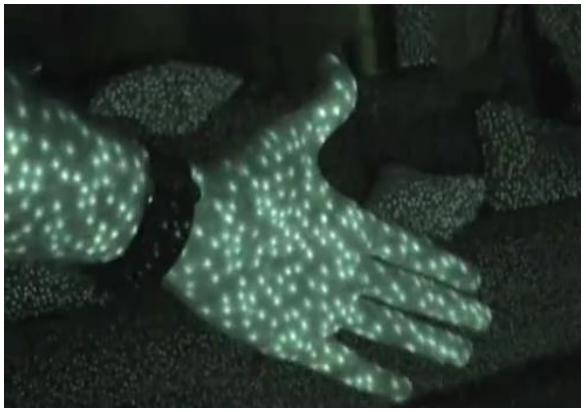


Depth Sensors

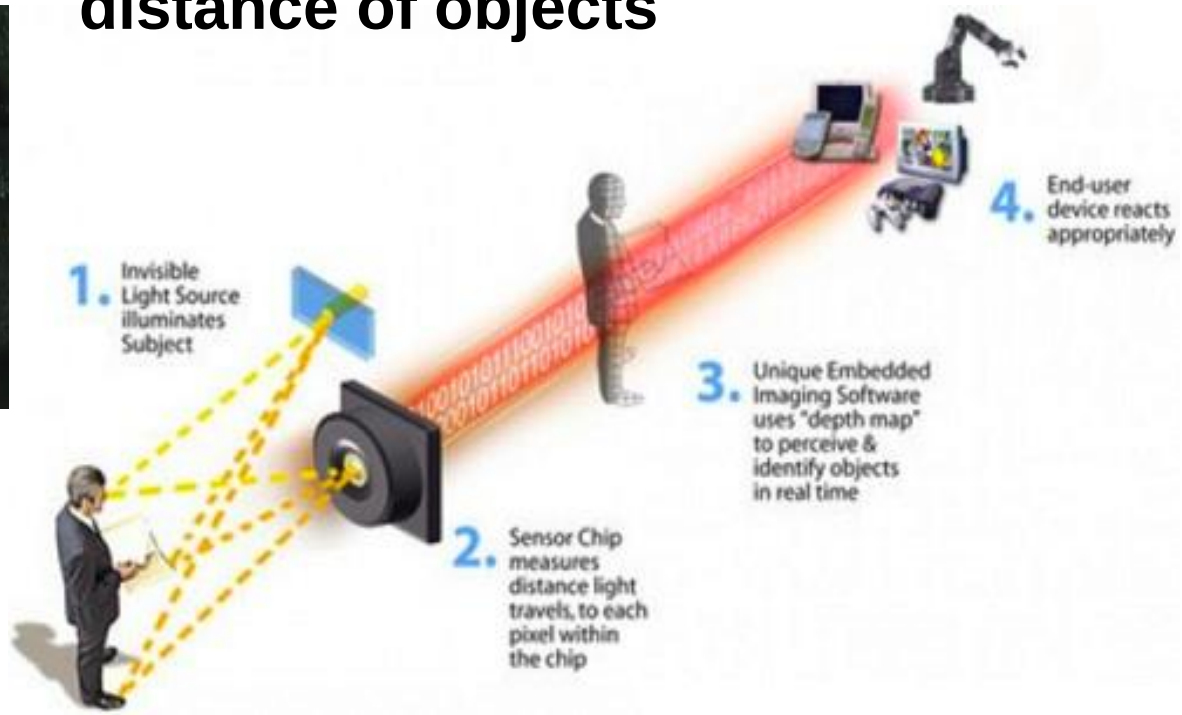
Consists of IR projector and a CMOS sensor

IR beam bounces off the subject and it is captured by the CMOS sensor

Depth Estimation



Sensor uses time to measure distance of objects



http://www.wired.com/gadgetlab/2010/11/tonights-release-xbox-kinect-how-does-it-work/

RGB Camera

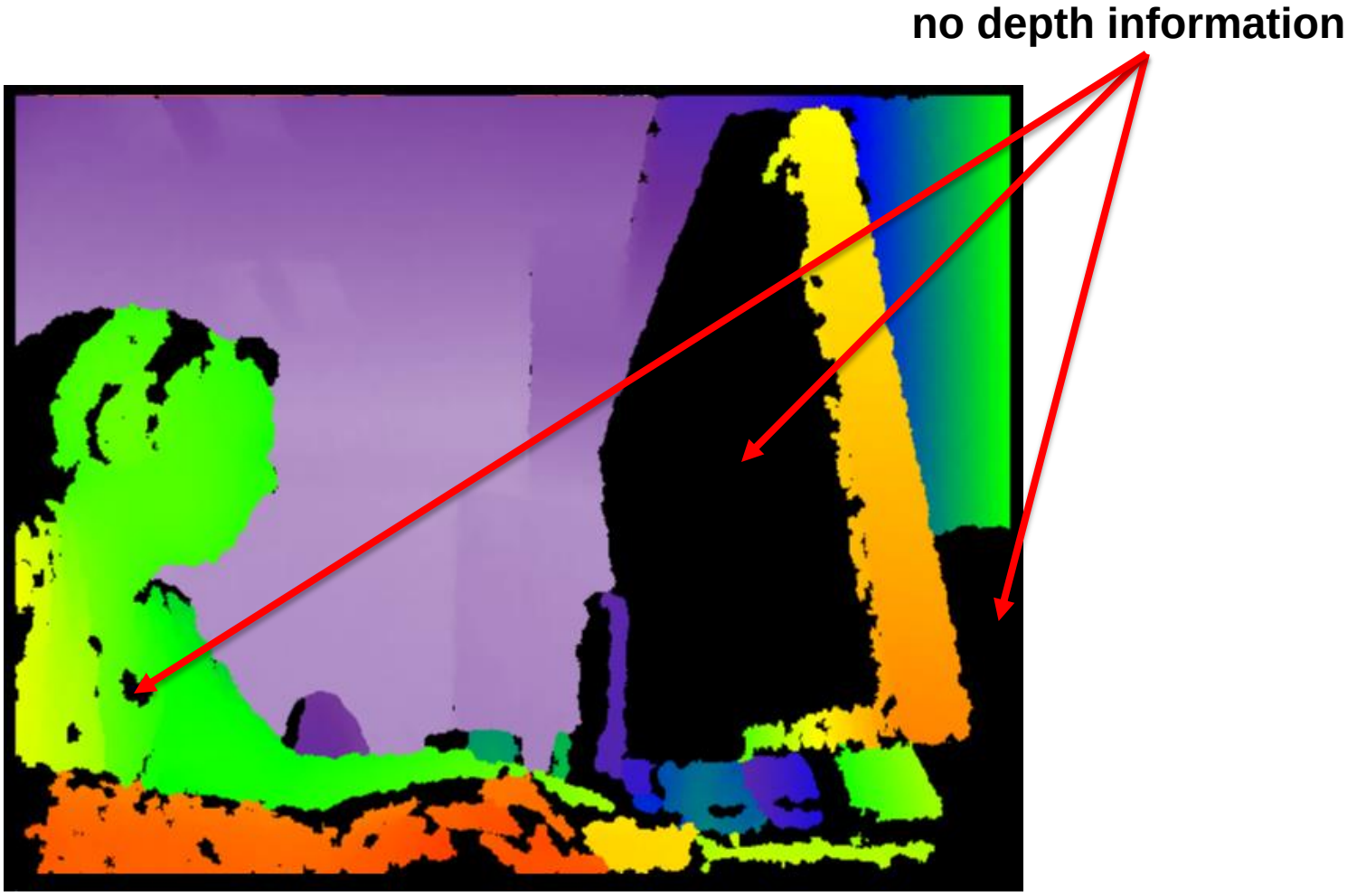
640 x 480 pixels at 30 frames per second (fps)

RGB + Depth

RGB data **11-bit depth data**

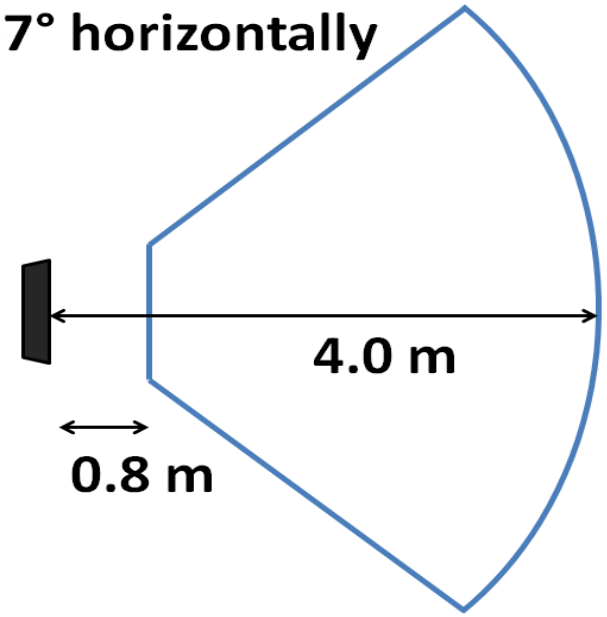
http://graphics.stanford.edu/~mdfisher/Kinect.html

Dense Depth Map

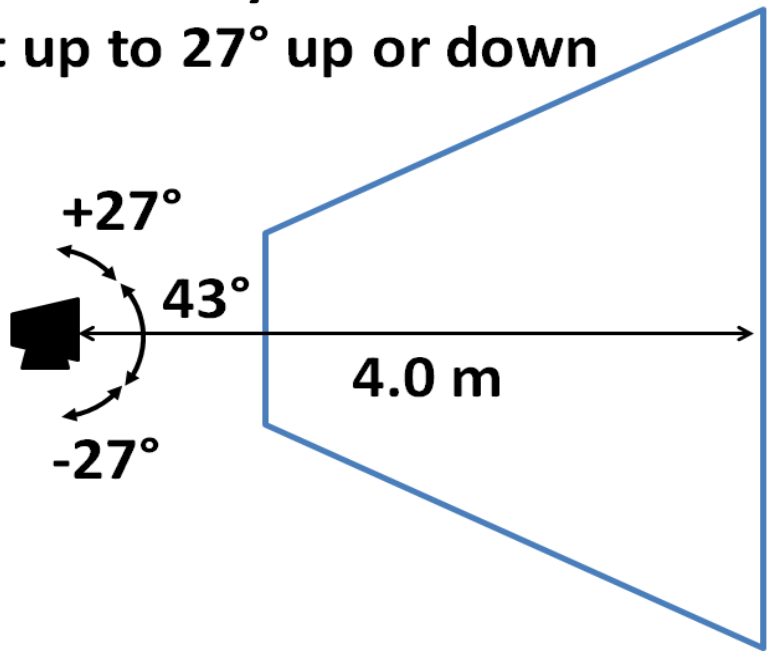


Kinect Physical Limitations

Angular field of view:
57° horizontally



Angular field of view:
43° vertically
Tilt up to 27° up or down

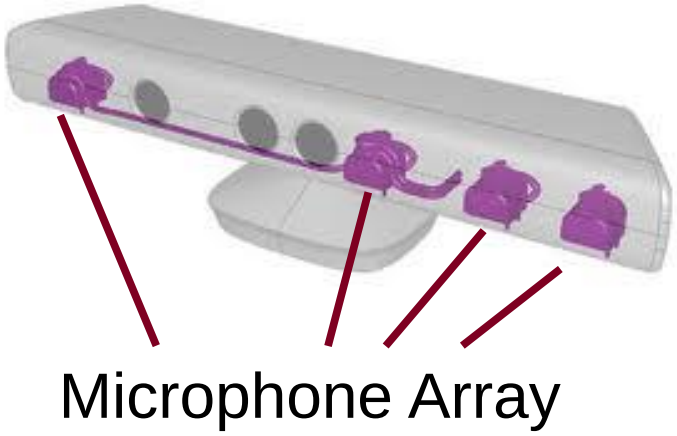


Microphones

4 Microphones on the device

Supports single speaker voice recognition

16-bit audio sampled at 16kHz



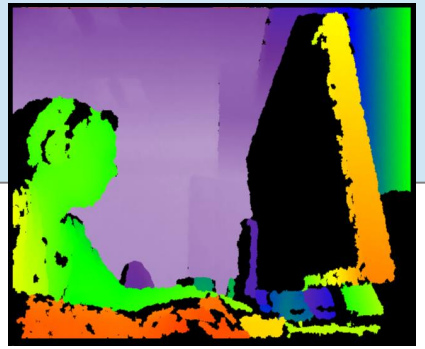
Visualizing RGBD Data

Image Transport Node

ROS

cv_bridge

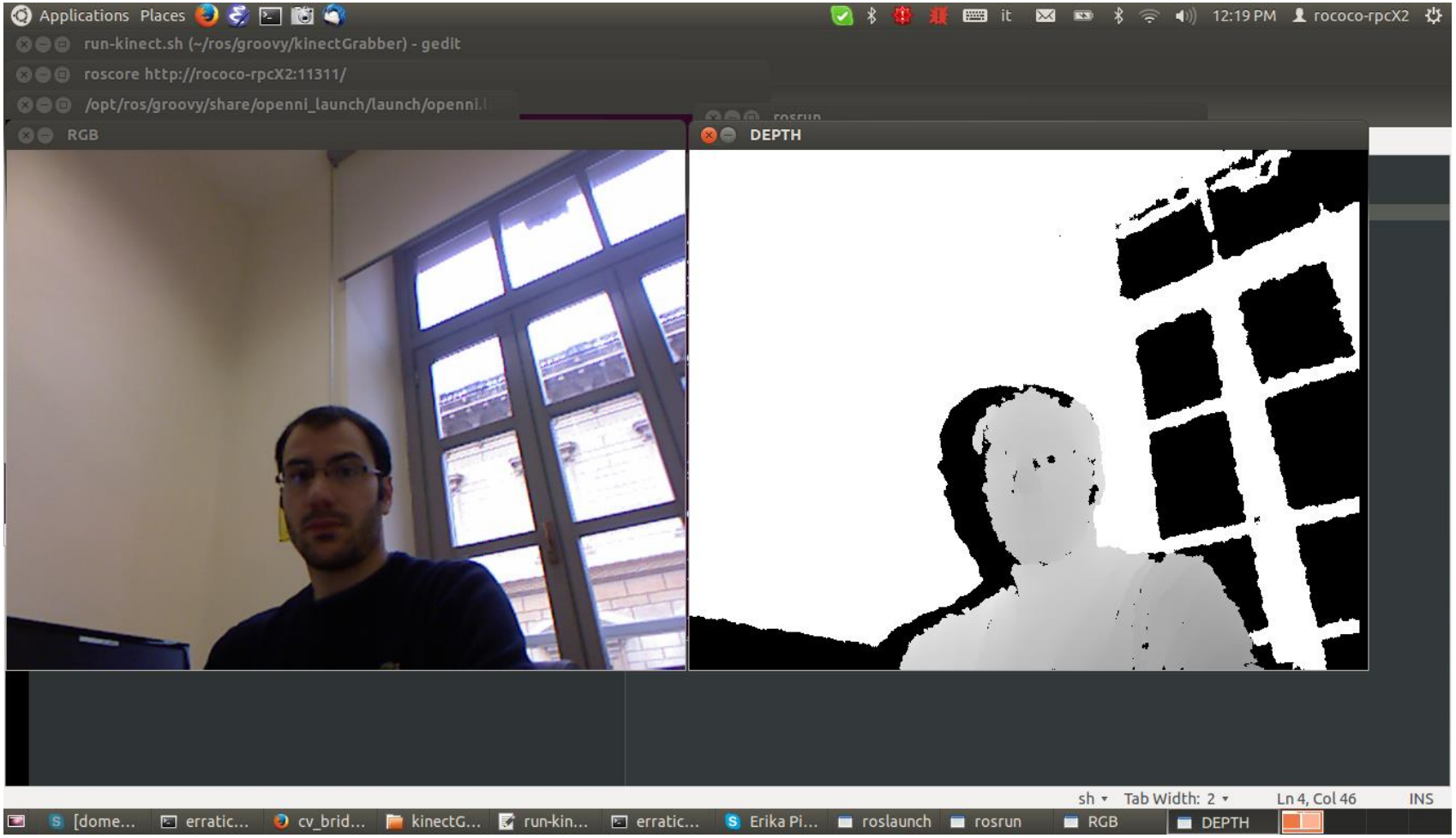
OPENCV



CV_8UC3

CV_16UC1

Kinect Viewer



Libraries

```
//ROS
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
//OpenCV
#include <opencv2/opencv.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
```

main

```
int main(int argc, char **argv) {
    ros::init(argc, argv, "kinectViewer");
    ros::NodeHandle n;
    ros::Subscriber sub =
       n.subscribe("/camera/rgb/image_color", 1, rgbCallback);
    ros::Subscriber depth =
       n.subscribe("/camera/depth/image", 1, depthCallback);
    ros::spin();
    return 0;
}
```

rgbCallback

```
void rgbCallback(const sensor_msgs::ImageConstPtr& msg)
{
  cv_bridge::CvImageConstPtr cv_ptr;
  try
  {
    cv_ptr =
        cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
  }
  catch (cv_bridge::Exception& ex)
  {
    ROS_ERROR("cv_bridge exception: %s", ex.what());
    exit(-1);
  }
  cv::imshow("RGB", cv_ptr->image);
  cv::waitKey(30); //!!!!!!
}
```

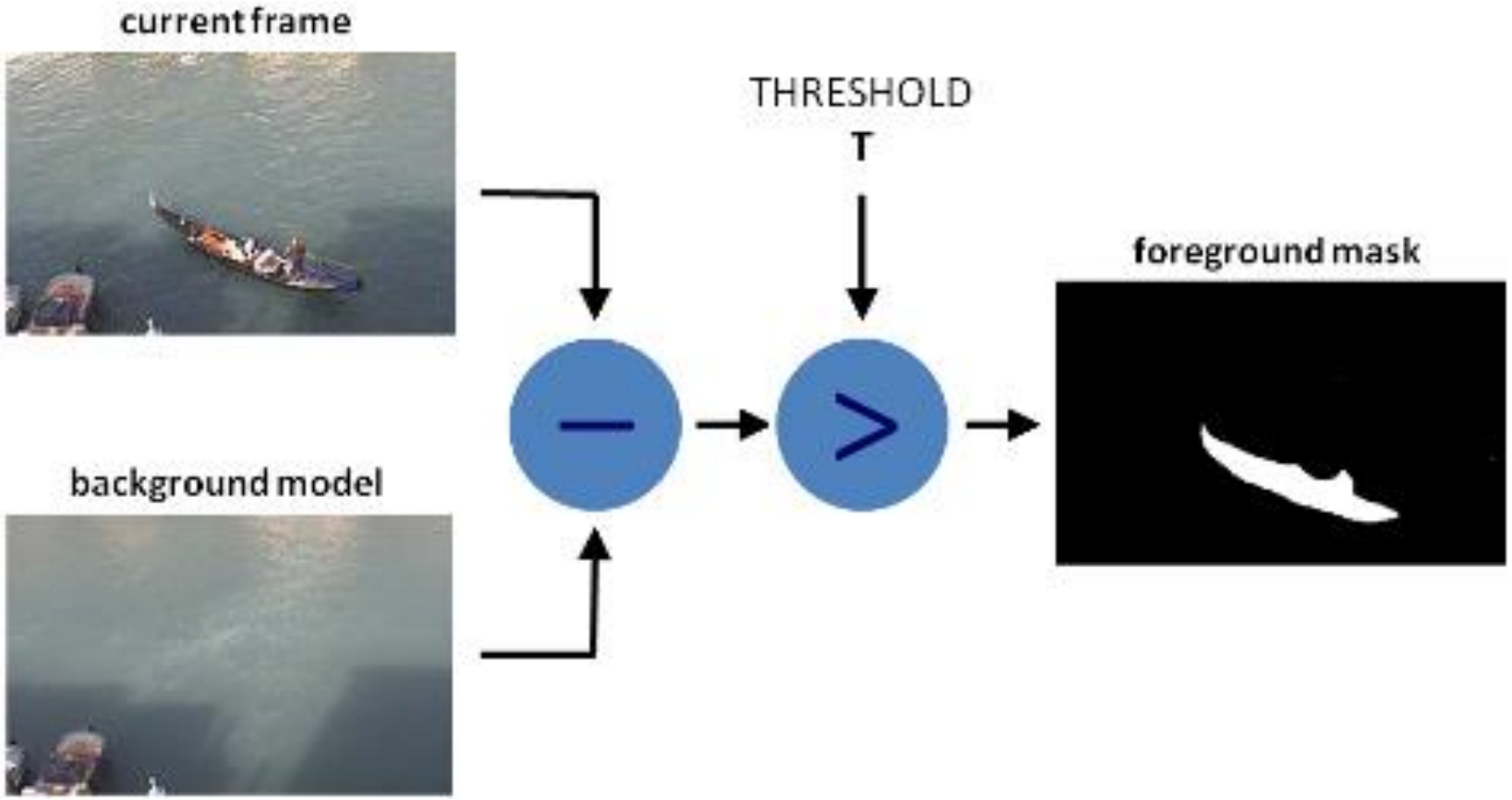
depthCallback

```
void depthCallback(const sensor_msgs::ImageConstPtr& msg)
{
  cv_bridge::CvImageConstPtr cv_ptr;
  try
  {
    cv_ptr =
        cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::TYPE_16UC1);
  }
  catch (cv_bridge::Exception& ex)
  {
    ROS_ERROR("cv_bridge exception: %s", ex.what());
    exit(-1);
  }
  cv::imshow("DEPTH", cv_ptr->image);
  cv::waitKey(30);
}
```

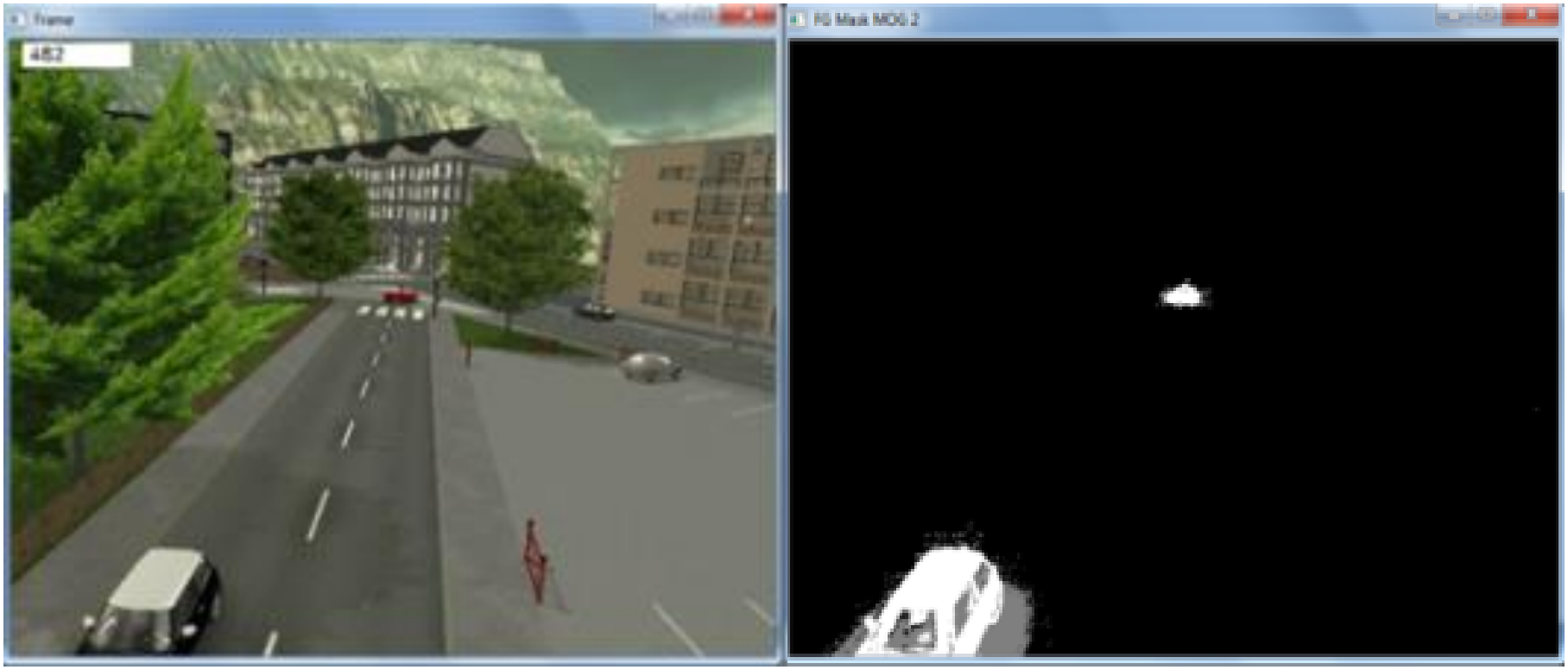
Downloading the source code

http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/kinectViewer.zip

Background Subtraction



Background subtraction issues



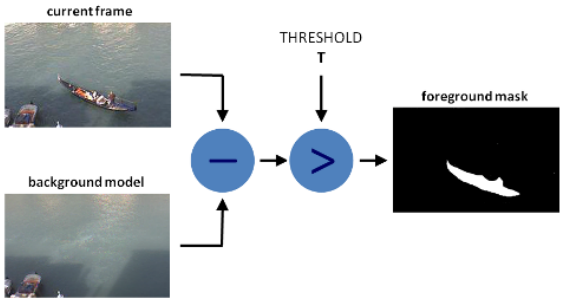
Background Subtraction in OpenCV

docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html

OpenCV 3.0.0-dev documentation » OpenCV Tutorials » *video* module. Video analysis » previous | next | index

How to Use Background Subtraction Methods

- Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.
- As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.



- Background modeling consists of two main steps:
  1. Background Initialization;
  2. Background Update.

  In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.
- In this tutorial we will learn how to perform BS by using OpenCV. As input, we will use data coming from the publicly available data set Background Models Challenge (BMC) .

OpenCV

Quick search

Go

Table Of Contents

Previous topic

video module. Video analysis

Next topic

http://docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html

Downloading the source code

http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/bgsub.zip

Point Clouds

PCL – Point Cloud Library

http://pointclouds.org/

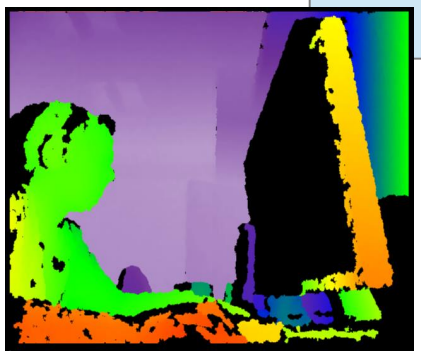
- The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing
- Collection of Libraries focused on Point Cloud processing
- More than 450 developers/contributors
- Over 60 Tutorials and many examples
- BSD Licensed - free for commercial use

Visualizing 3D Data

ROS

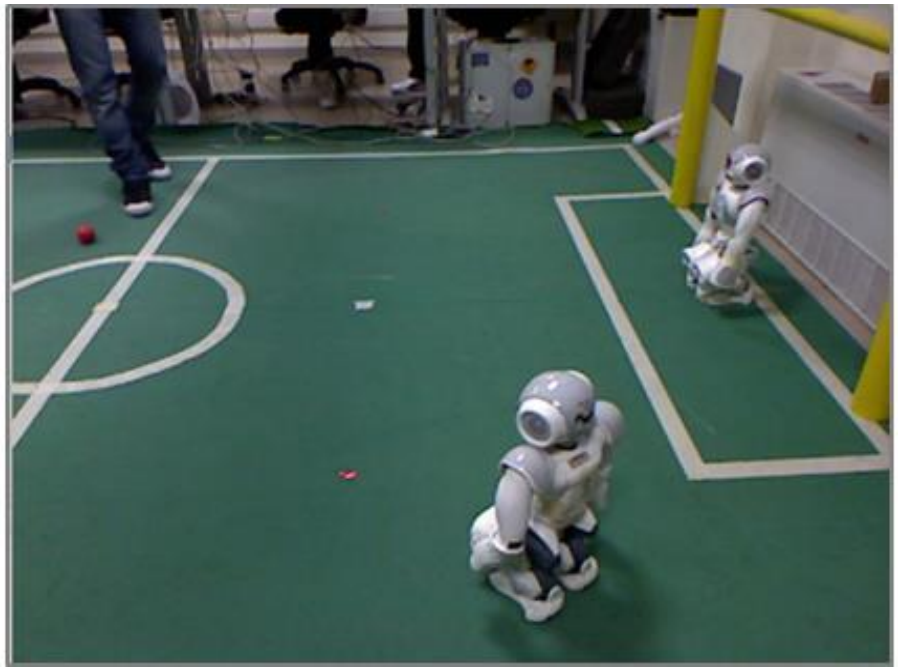
OPENCV

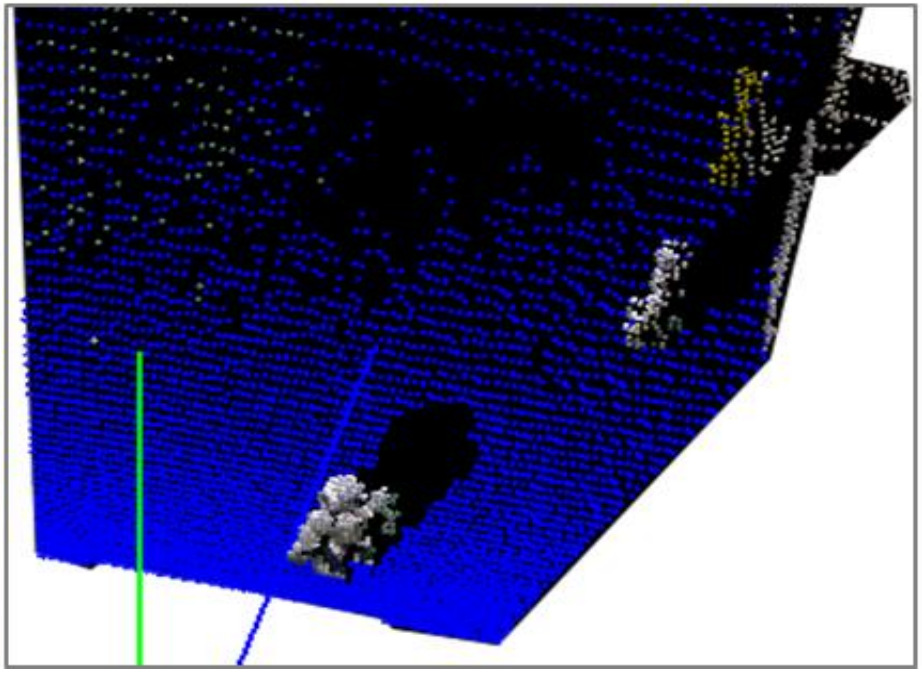
PCL



Lecture on Perception with RGBD sensor-3D

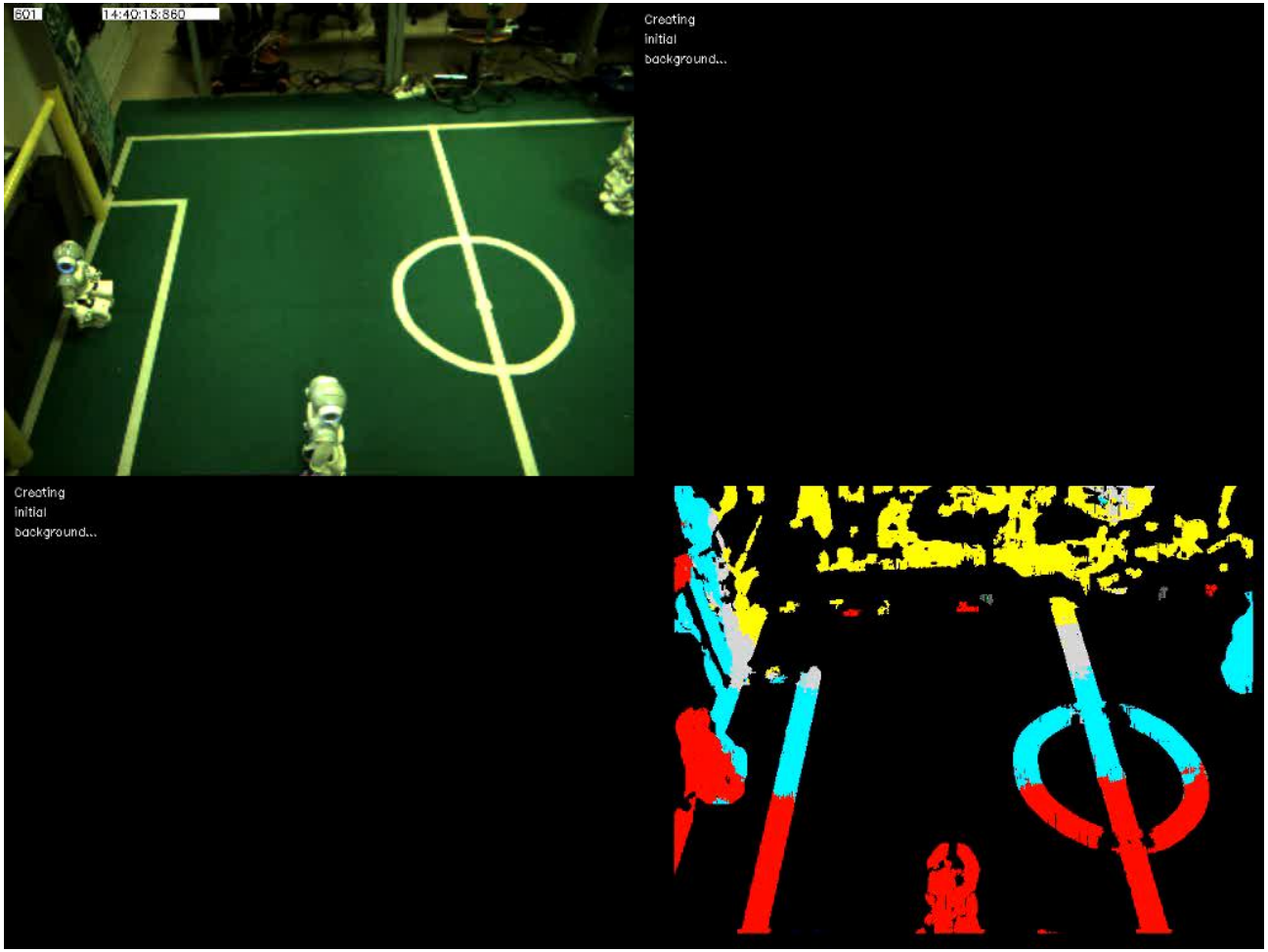
Point Clouds with NAO





Andrea Pennisi, Domenico D. Bloisi, Luca Iocchi, and Daniele Nardi
Ground Truth Acquisition of Humanoid Soccer Robot Behaviour
RoboCup 2013: Robot World Cup XVII

Offside recognition with NAO



http://www.dis.uniroma1.it/~pennisi/videoRoboticWeek/3.avi

3D Face Visualization

special thanks to
Roberto Capobianco and
Jacopo Serafin

Tools

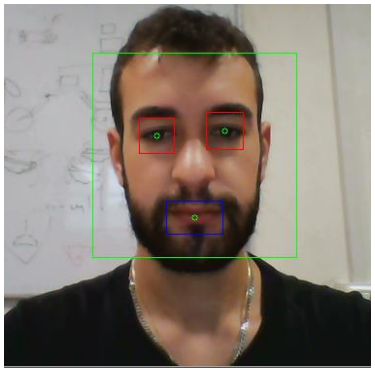
- Microsoft Kinect or Asus Xtion
- OpenCV (Open Computer Vision)
- PCL (Point Cloud Library)
- ROS (Robot Operating System)

1. **Infrared optics** — A projector and sensor map over 48 points on the human body.
2. **RGB camera** — The camera combines with the 3D map to create the image you see on screen.
3. **Motorized tilt** — Mechanical gears at the base let the game follow you.
4. **Multi-array microphone** — Four microphones cancel out ambient noise and pinpoint where you are in the room.

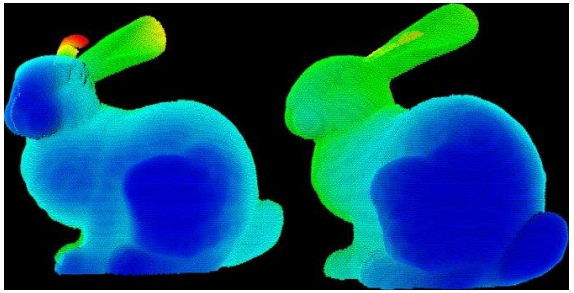
2D + Depth + 3D

- Data Acquisition (ROS + Kinect)

- Face Detection (OpenCV)



- 3D Visualization (PCL)



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_) {
        …
}
```

Data Acquisition

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

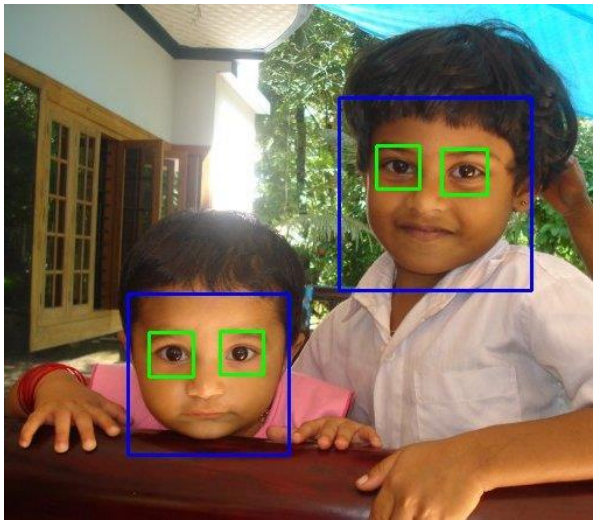
```
#include <message_filters/subscriber.h>
#include <message_filters/synchronizer.h>
#include <message_filters/sync_policies/approximate_time.h>

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 – Viola and Jones Method

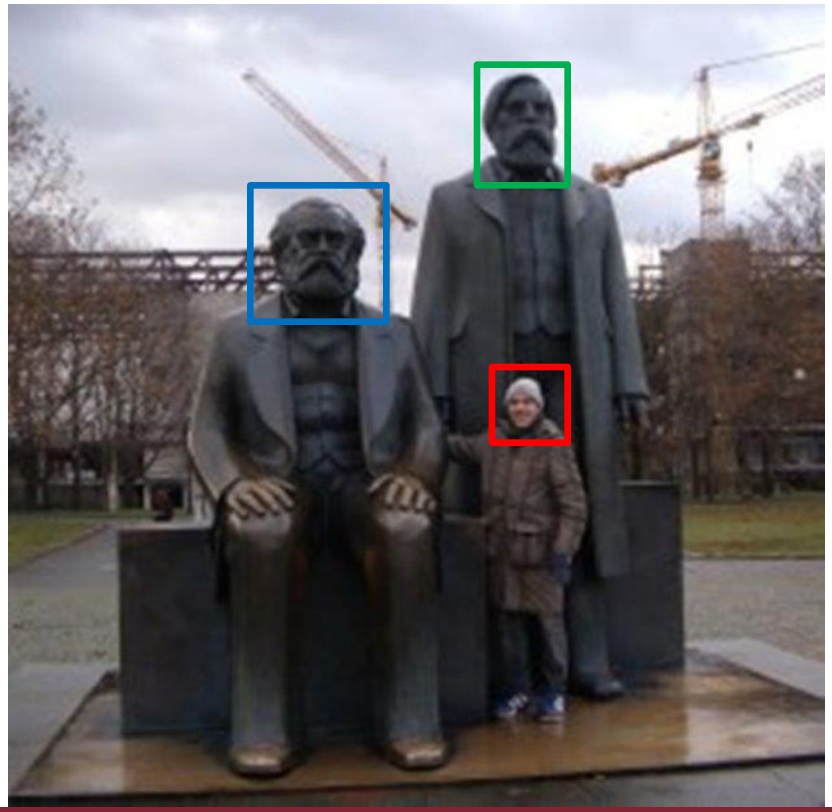
- Viola-Jones method implementation of the OpenCV library (by using haar-like cascades)



- OpenCV comes with a trainer as well as a detector
- OpenCV already contains many pre-trained classifiers for face, eyes, smile etc.

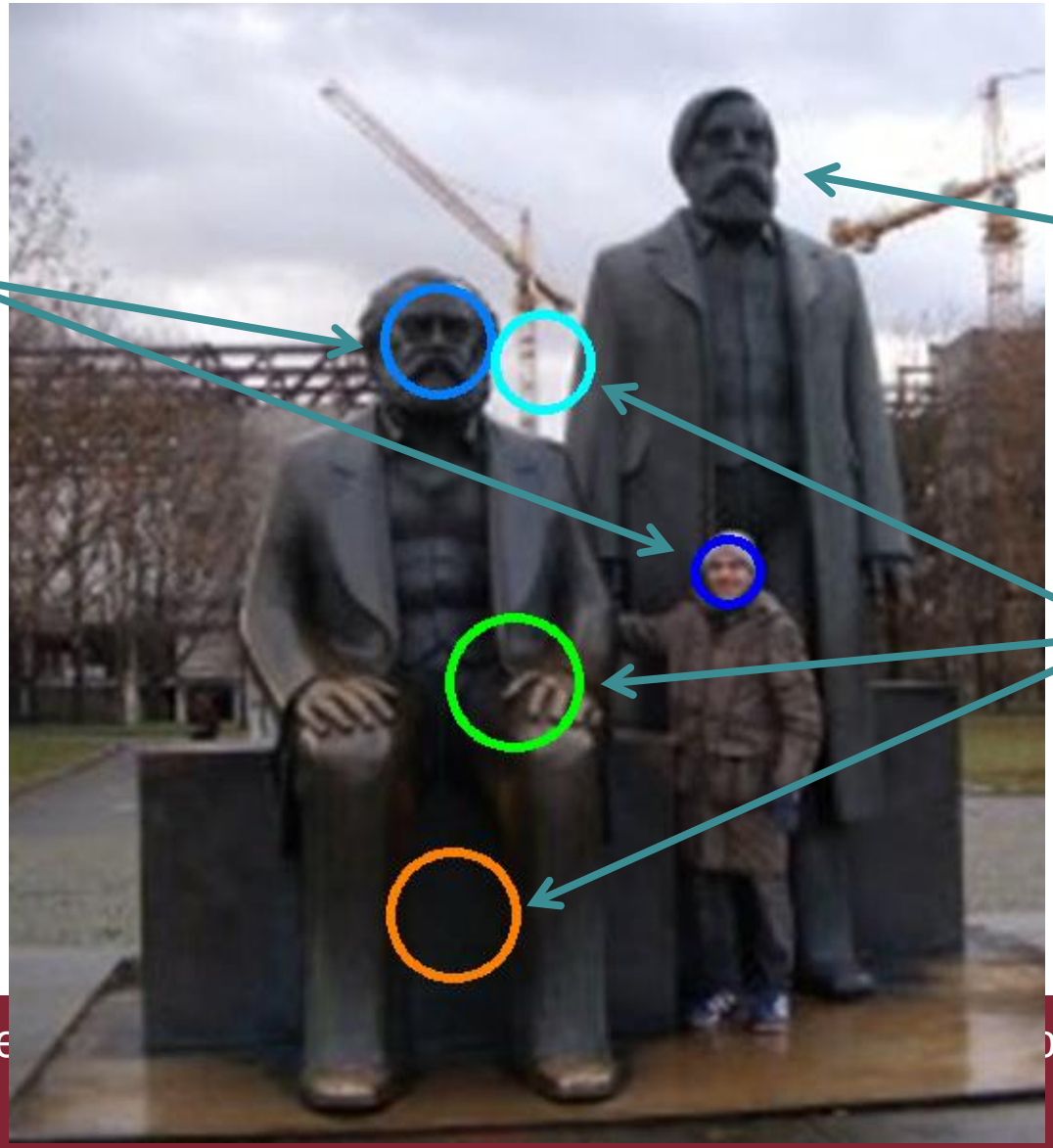
Face Detection Problem

Given an image, find regions in the image which contain instances of faces.



Detection Issues

TP
True Positive

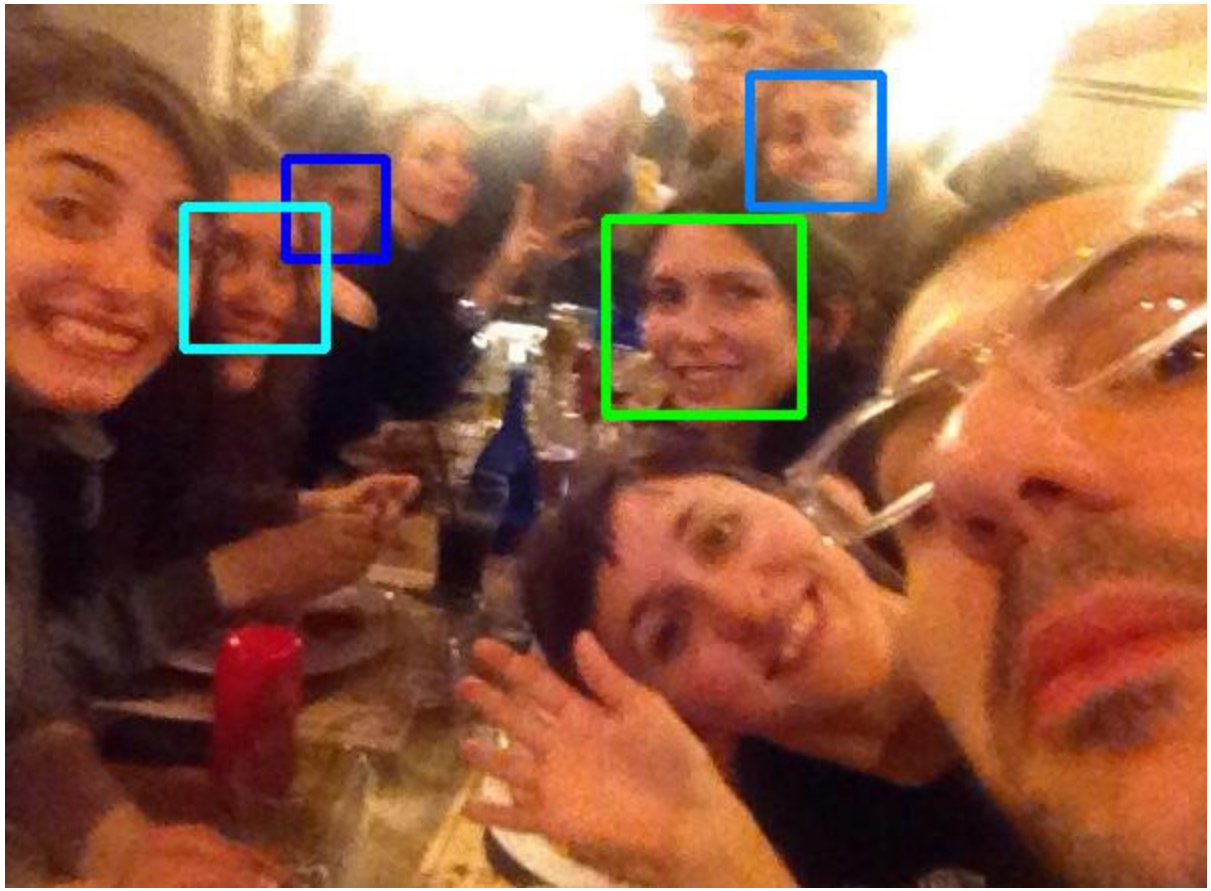


FN
False Negative

FP
False Positive

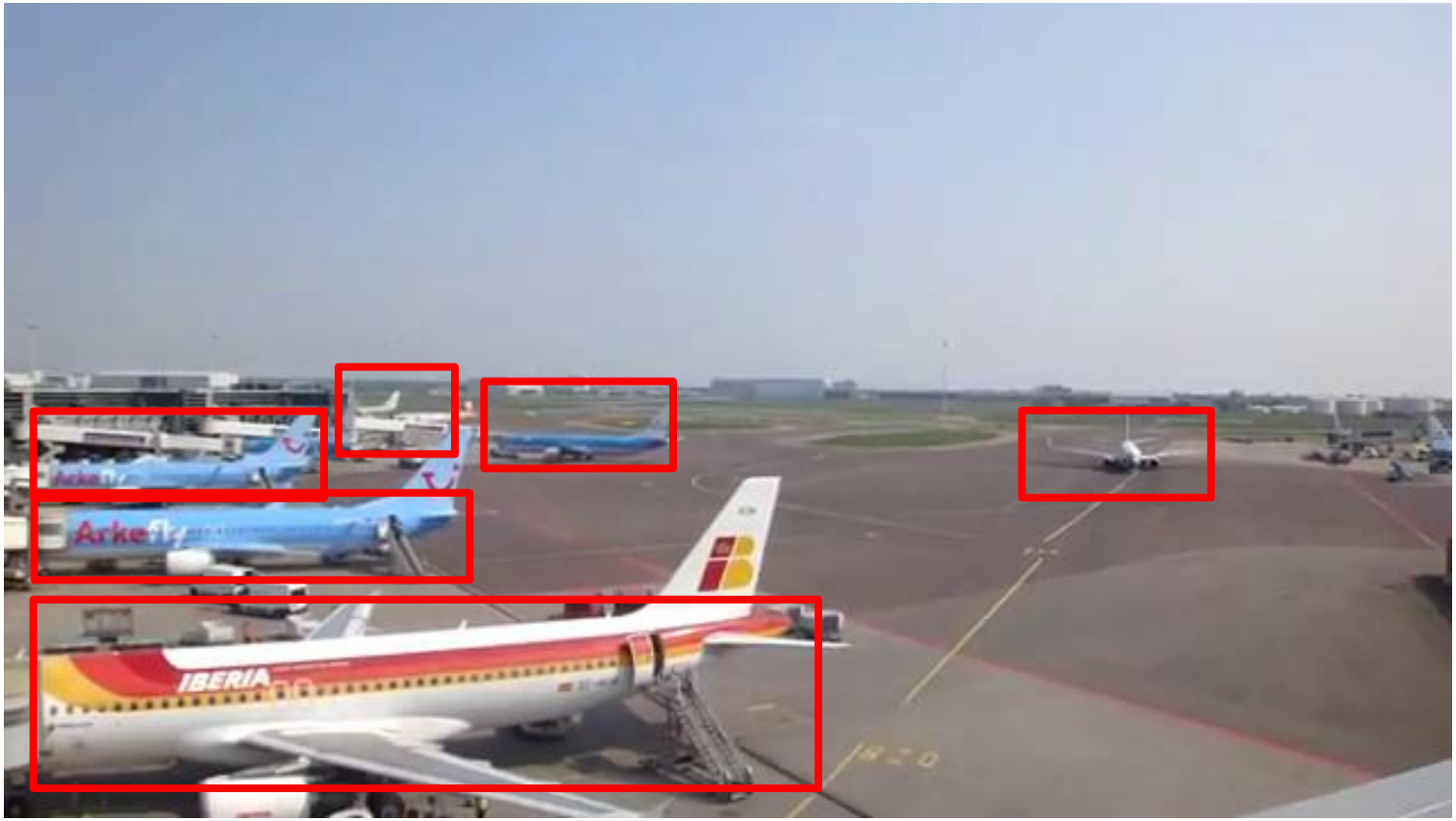
Additional Issues

- Rotation
- Blurring
- Illumination
- Occlusions
- Glasses
- …



General Problem

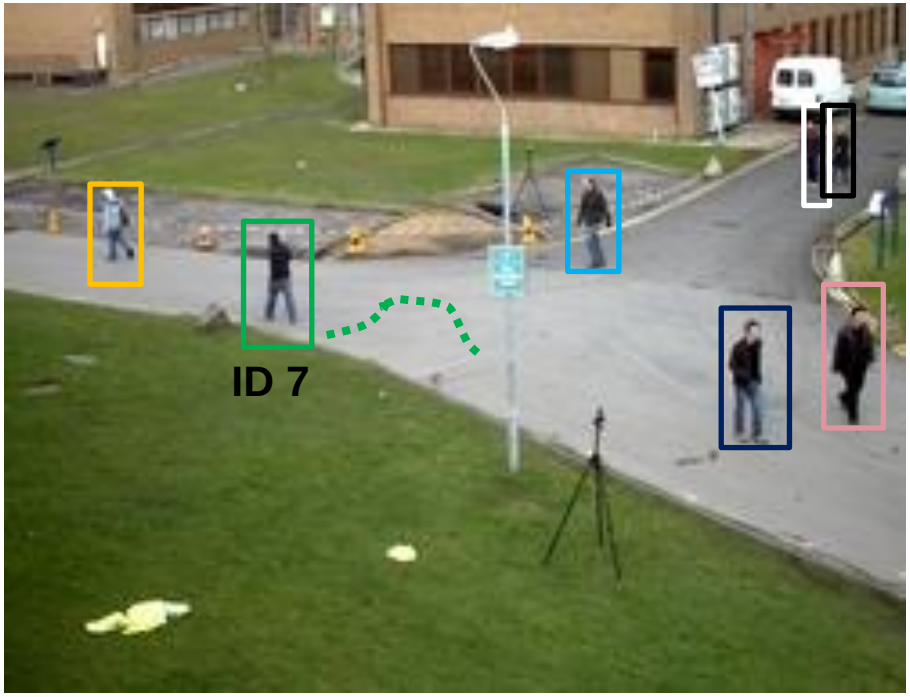
- Given an image, find a specific object in it.



Detection is not Identification

detection

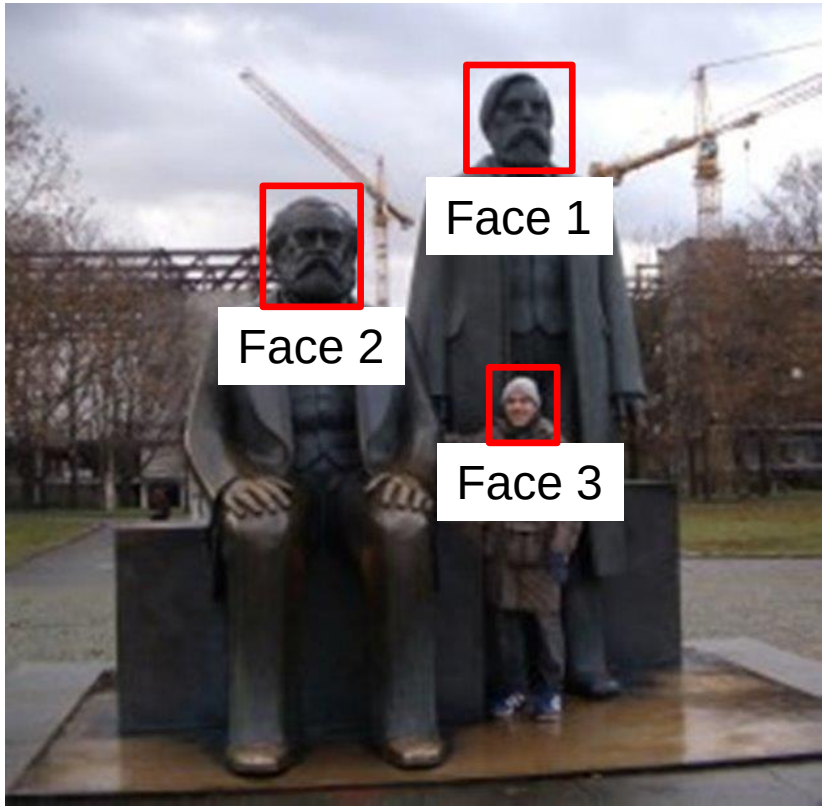
identification



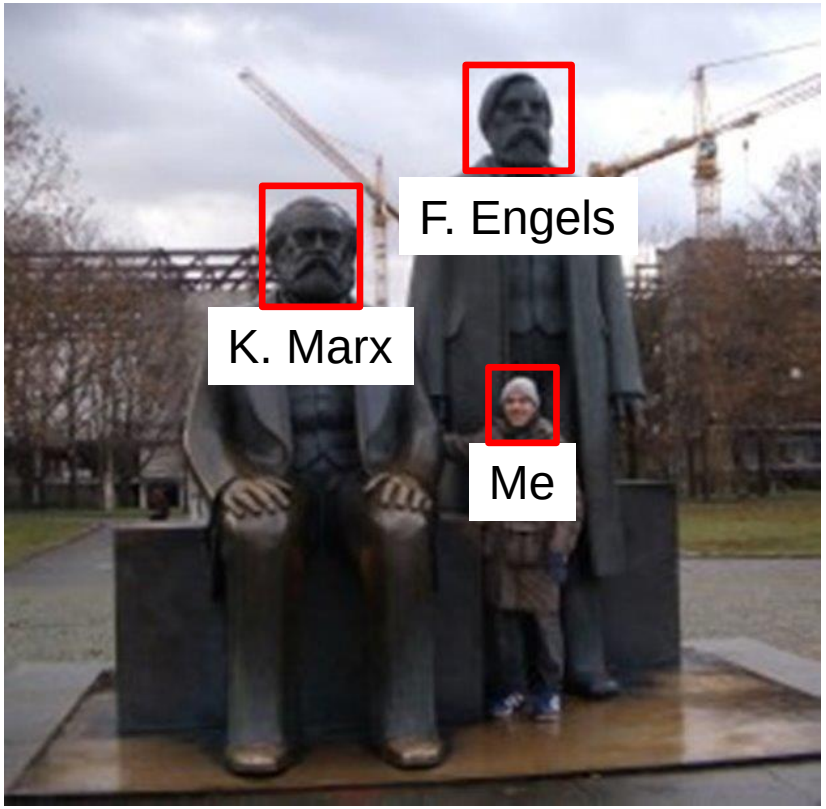
PETS 2009 dataset
http://www.cvg.rdg.ac.uk/PETS2009

Detection is not Recognition

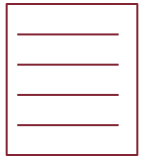
detection



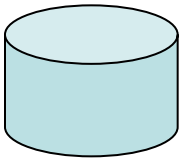
recognition



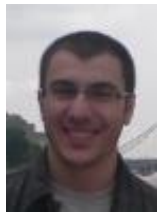
Detection vs Recognition



parameters



face database

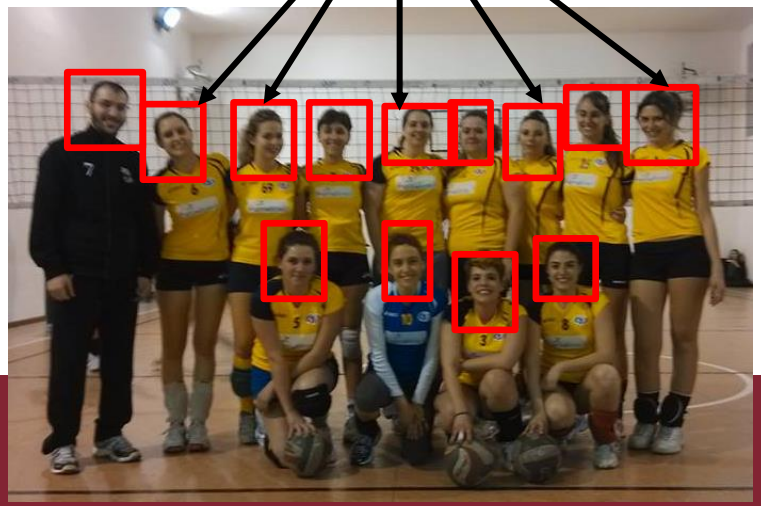


FACE DETECTION

FACE RECOGNITION

Output

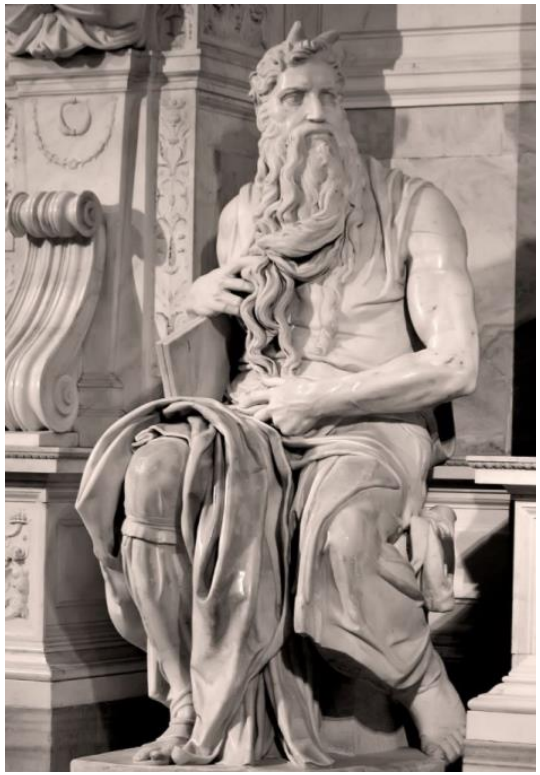
ME



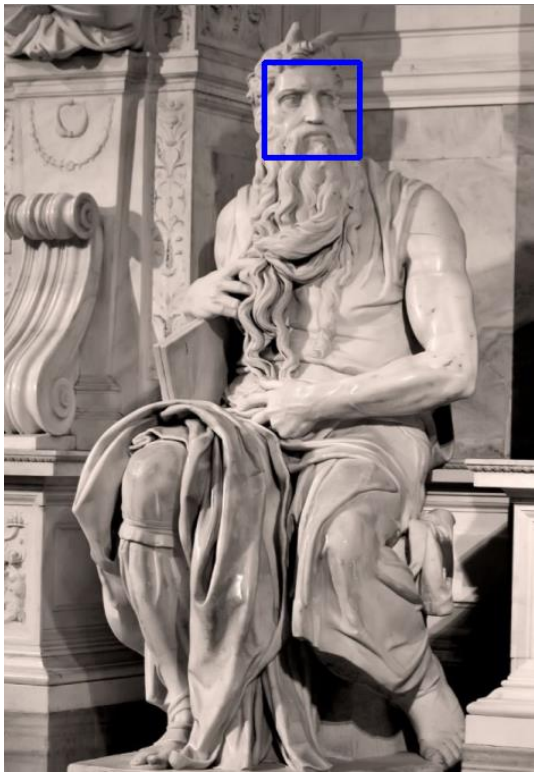
Classification-based Detection

The output is a bounding box around the instance(s) belonging to the class of objects for which the classifier has been trained.

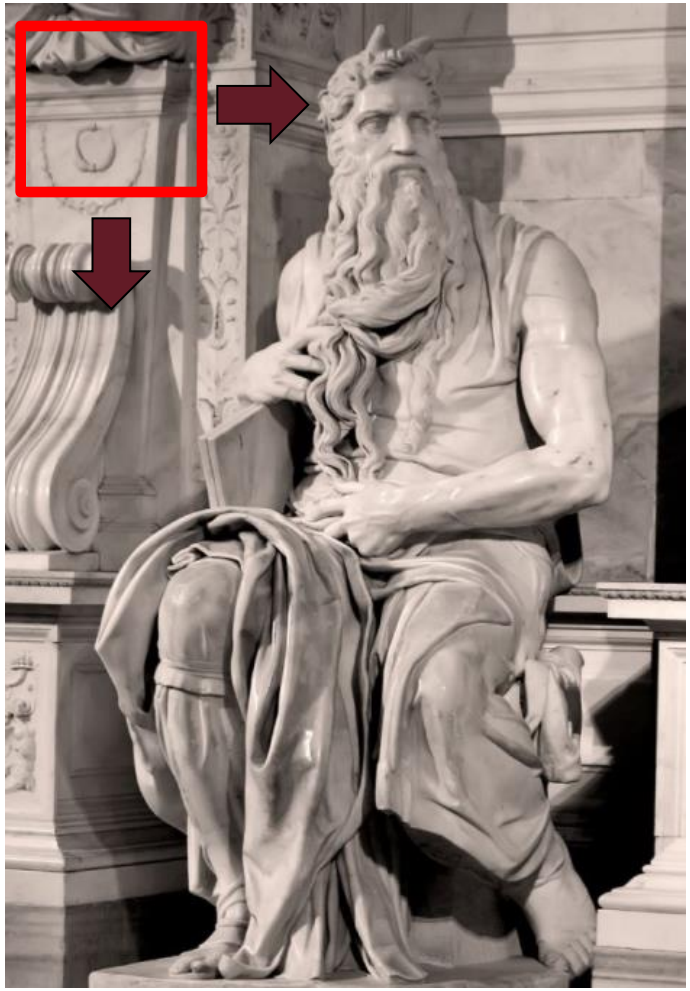
input image



detection



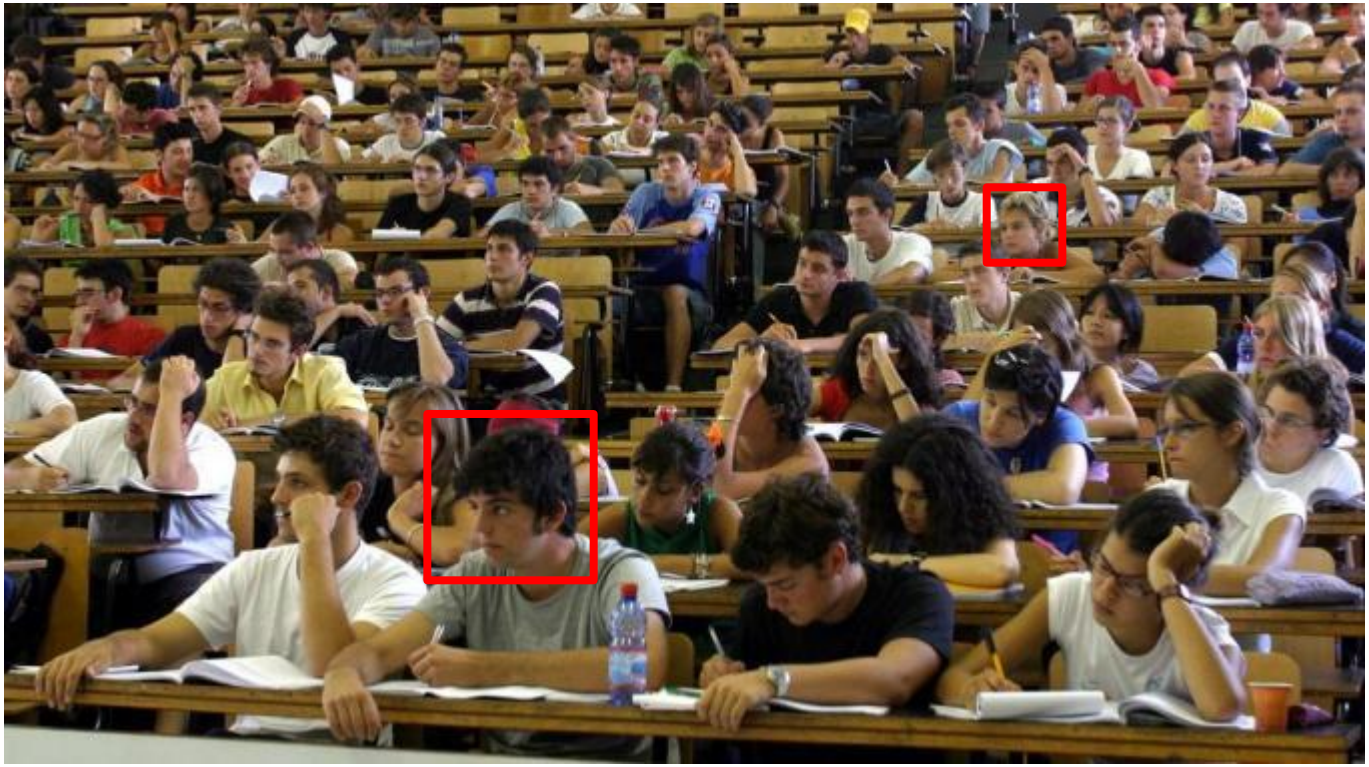
Basic Idea

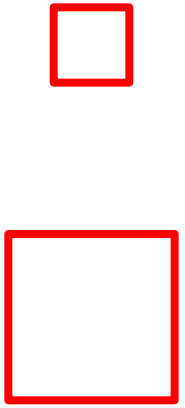


Slide a window (e.g., 30x30) across the image and evaluate the current portion of the image w.r.t. an object model at every location

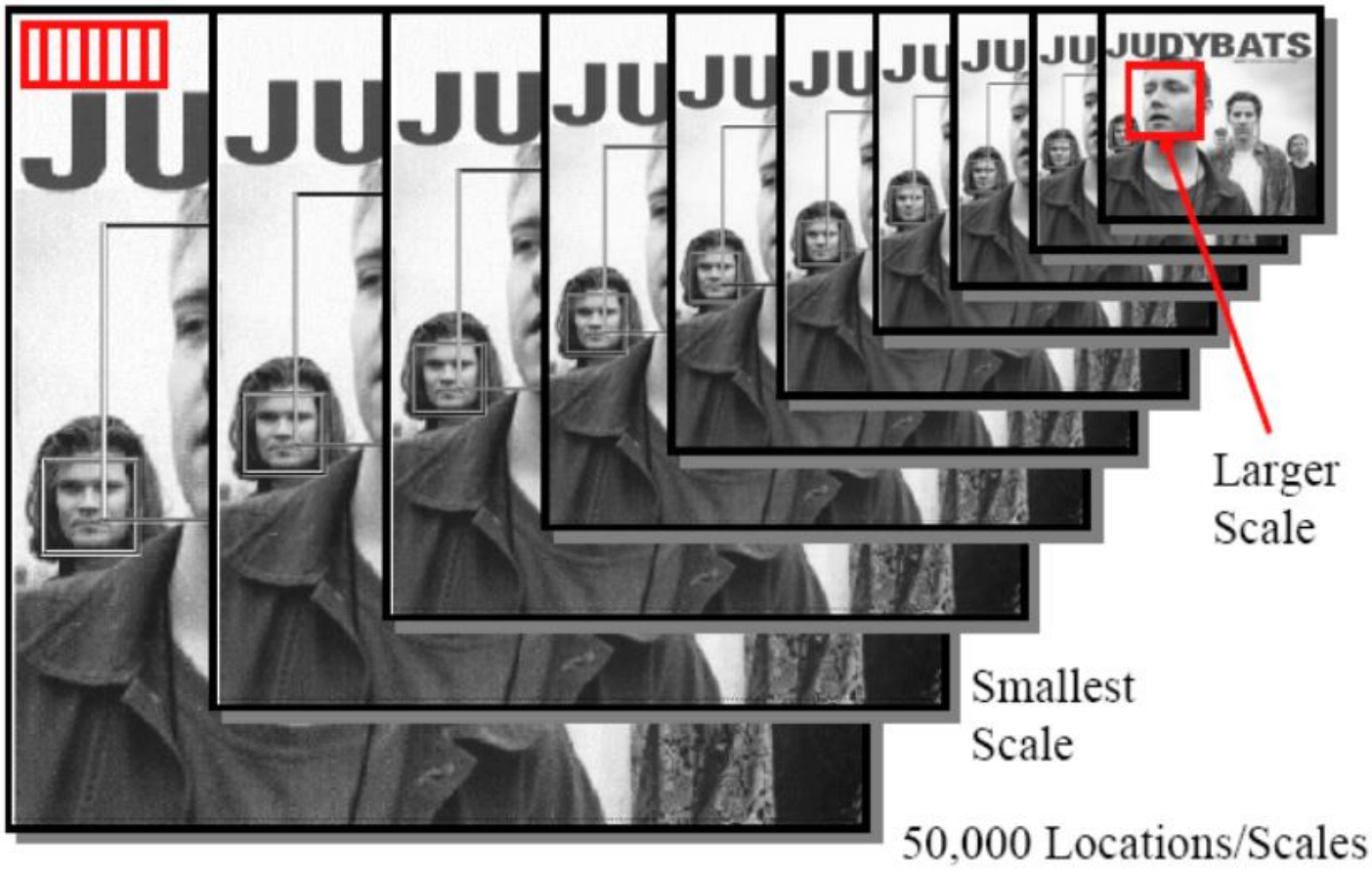
The number of locations where the object is present is very very small

Multiple Scales





Scan window over image pyramid



The Viola and Jones Method

- **The most famous method**
- **Recognition is very fast**
  (e.g., real-time for digital cameras)
- **Key contributions**
  1. **Integral image for fast feature extraction**
  2. **Boosting (Ada-Boost) for face detection**
  3. **Attentional cascade for fast rejection of non-face sub-windows**

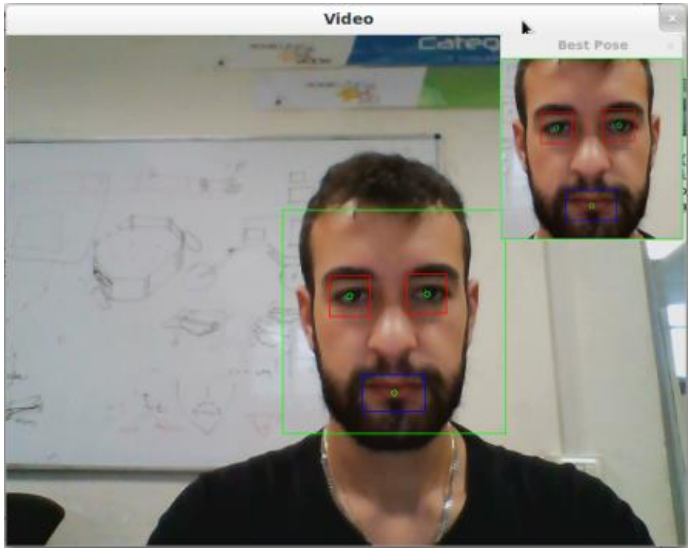
Training may need weeks

[1] P. A. Viola, M. J. Jones. Rapid object detection using a boosted cascade of simple features, CVPR, pp.511-518, 2001

[2] P. A. Viola, M. J. Jones. Robust Real-Time Face Detection. IJCV 57(2), pp. 137-154, 2004

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

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

```
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

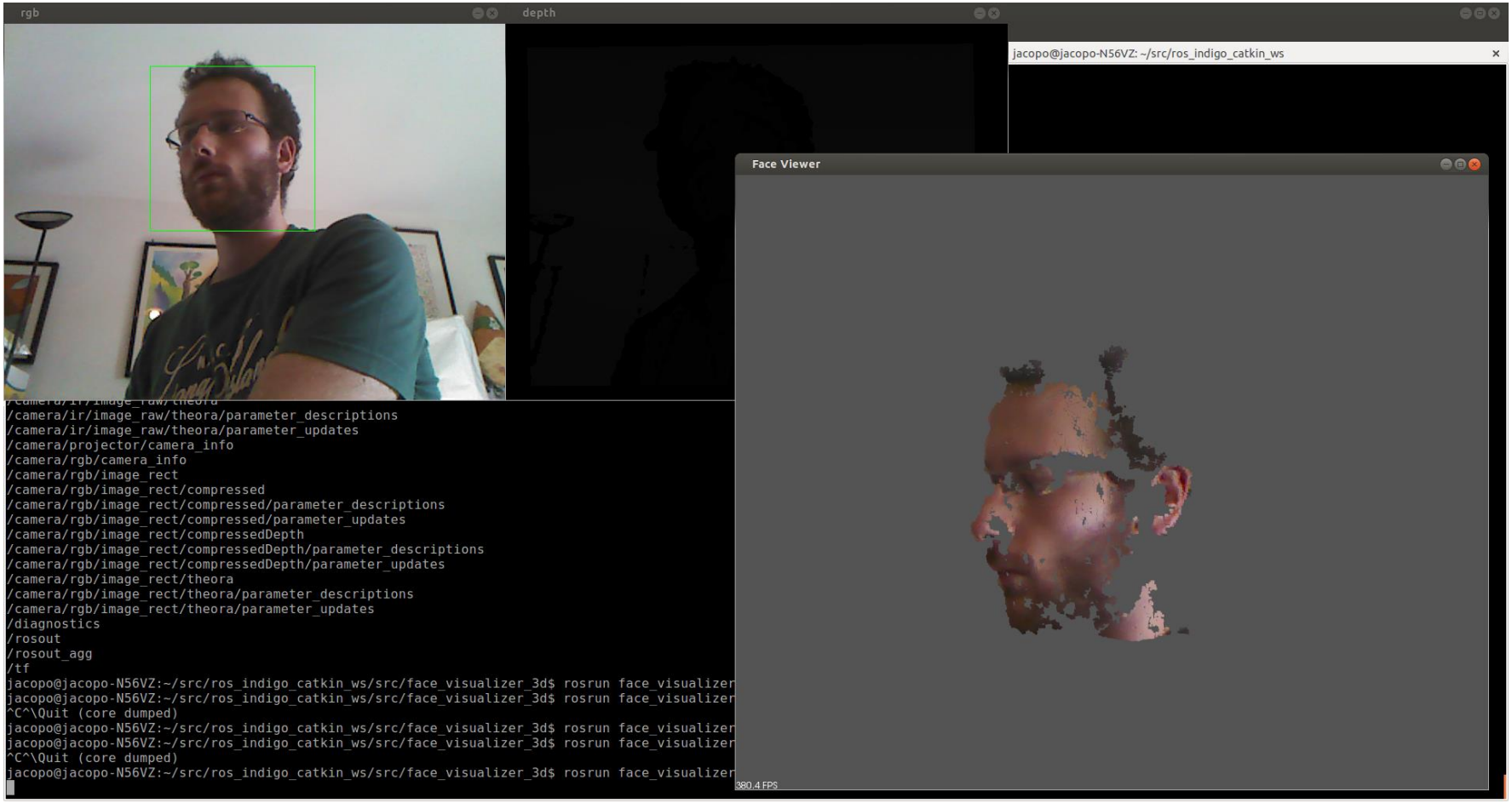
```
#include <pcl/visualization/pcl_visualizer.h>

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);
```

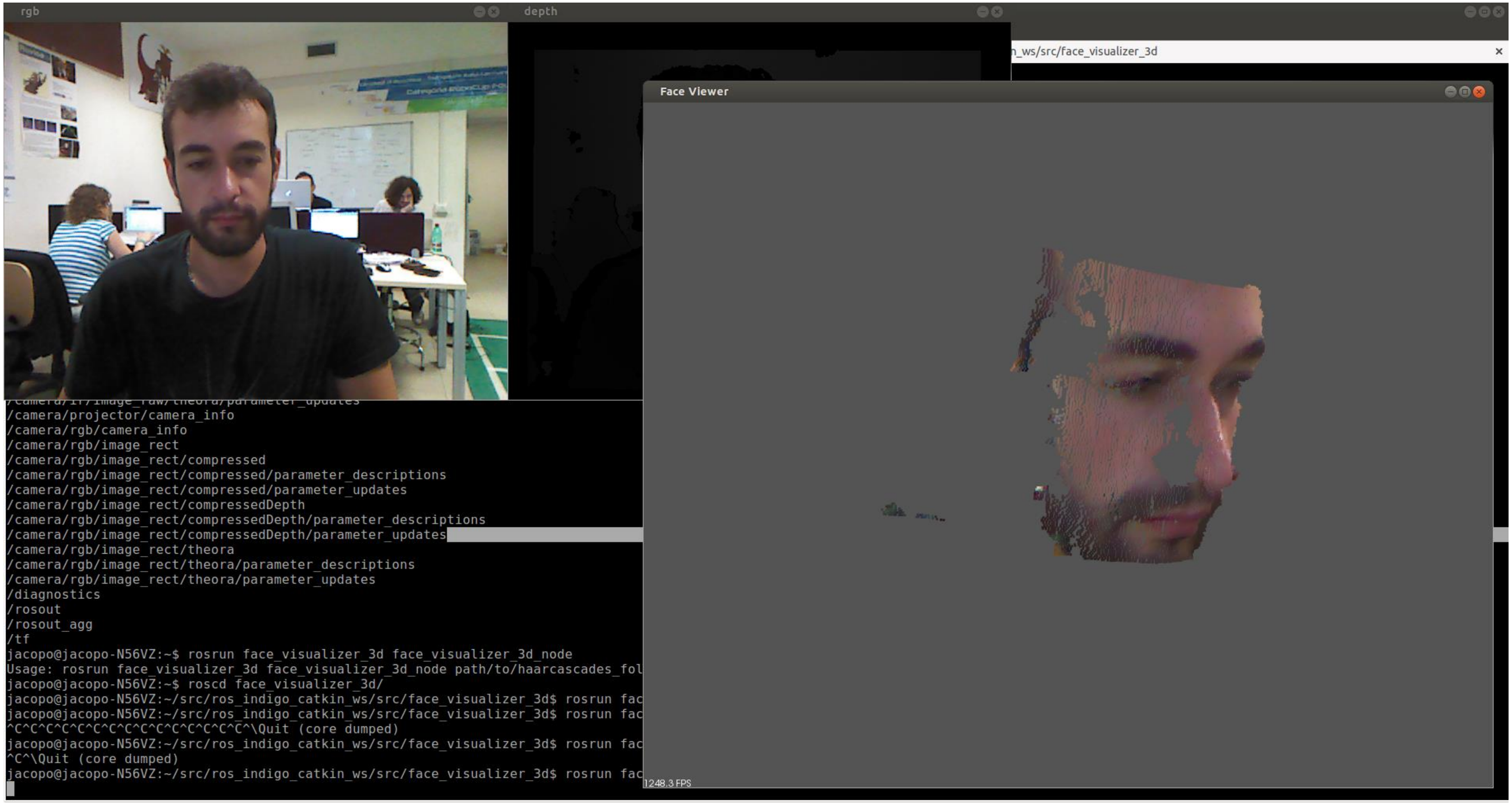
- Add/Remove a PointCloud to the Visualizer

```
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 - 1


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 - 2


rgb
depth
Face Viewer
_ws/src/face_visualizer_3d
/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

References and Credits

Some slides of this presentation adapted from

- KH Wong. “Ch. 6: Face detection”
- P. Viola and T.-W. Yue. “Adaboost for Face Detection”
- D. Miller. “Face Detection & Synthesis using 3D Models & OpenCV”
- S. Lazebnik. “Face detection”
- C. Schmid. “Category-level localization”
- C. Huang and F. Vahid. “Scalable Object Detection Accelerators on FPGAs Using Custom Design Space Exploration”
- P. Smyth. “Face Detection using the Viola-Jones Method”
- K. Palla and A. Kalaitzis. “Robust Real-time Face Detection”

Downloading the source code

http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/3DFaceVisualizer.zip

Homework

- Option 1
  Modify the code in
  http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/bgsub.zip
  to read the ROS bag
  http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/people.bag
  and to estimate dynamically the 3D position of the people in the scene with respect to the camera.

- Option 2
  Modify the code in
  http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/3DFaceVisualizer.zip
  to read the ROS bag
  http://www.dis.uniroma1.it/~bloisi/didattica/RobotProgramming/face.bag
  and to estimate dynamically the 3D position of the face in the scene with respect to the camera.

Projects

- 3D Coca-Cola can detector
- 3D People detection
- Face matching