People Detection in RGB-D Data

Matteo Munaro
matteo.munaro@dei.unipd.it
Intelligent Autonomous Systems Lab (IAS-Lab)
PCL Tutorial – ICRA 2013
People Detection in RGB-D Data

People Detection

RGB-D data

Point Cloud Library (PCL)
- Dense scanning of RGB and depth images [1] (GPU processing required for real time)

- ROI tracking and dense scanning of the ROIs [2]

Objectives

- detecting people walking on a ground plane
- from a mobile robot
- in real time with standard CPU

Ground plane estimation and removal
Ground plane estimation and removal

Euclidean clustering of remaining points
➢ People vertically **split**ed into more clusters
- People vertically *splitted* into more clusters
People vertically **splitted** into more clusters

More people **merged** into the same cluster
Main issues

- People vertically **split**ted into more clusters

- More people **merged** into the same cluster
Main issues

- People vertically **split** into more clusters

- More people **merge** into the same cluster
Sub-clustering procedure
1. Candidates **pruning** based on **height** from the ground plane
1. Candidates **pruning** based on **height** from the ground plane
2. **Merging** of clusters **close** in ground plane coordinates
1. Candidates **pruning** based on **height** from the ground plane
2. **Merging** of clusters **close** in ground plane coordinates
3. **Subdivision** of big clusters by means of **head detection** (peaks in height from the ground plane)
1. Candidates **pruning** based on **height** from the ground plane
2. **Merging** of clusters **close** in ground plane coordinates
3. **Subdivision** of big clusters by means of **head detection** (peaks in height from the ground plane)
1. Candidates **pruning** based on **height** from the ground plane

2. **Merging** of clusters **close** in ground plane coordinates

3. **Subdivision** of big clusters by means of **head detection** (peaks in height from the ground plane)
1. Candidates **pruning** based on **height** from the ground plane
2. **Merging** of clusters **close** in ground plane coordinates
3. **Subdivision** of big clusters by means of **head detection** (peaks in height from the ground plane)
1. Candidates **pruning** based on **height** from the ground plane
2. **Merging** of clusters **close** in ground plane coordinates
3. **Subdivision** of big clusters by means of **head detection** (peaks in height from the ground plane)
Candidate **clusters** are **extended** until the ground plane for achieving robustness to lower limbs occlusion.
Candidate clusters are extended until the ground plane for achieving robustness to lower limbs occlusion.
Candidate **clusters** are **extended** until the ground plane for achieving robustness to lower limbs occlusion.
Candidate **clusters** are **extended** until the ground plane for achieving robustness to lower limbs occlusion

- **HOG detector** [4] applied to image patches that are projection of 3D clusters onto the RGB image

---

With merging and height constraint
HOG confidence

HOG trend for the central person
GroundBasedPeopleDetectionApp<\text{PointT}>

- PersonClassifier<\text{pcl::RGB}>
  - HOG

- PersonCluster<\text{PointT}>

- HeadBasedSubclustering<\text{PointT}>
  - HeightMap2D<\text{PointT}>

\text{pcl::people}
#include <pcl/people/ground_based_people_detection_app.h>
...

// Create classifier for people detection:
pcl::people::PersonClassifier<pcl::RGB> person_classifier;
person_classifier.loadSVMFromFile(svm_filename);  // load trained SVM
#include <pcl/people/ground_based_people_detection_app.h>
...

// Create classifier for people detection:
pcl::people::PersonClassifier<pcl::RGB> person_classifier;
person_classifier.loadSVMFromFile(svm_filename);    // load trained SVM

// People detection app:
pcl::people::GroundBasedPeopleDetectionApp<PointT> people_detector;  // people detection object
std::vector<pcl::people::PersonCluster<PointT> > clusters;  // vector containing persons clusters
people_detector.setIntrinsics(rgb_intrinsics_matrix);     // set RGB camera intrinsic parameters
people_detector.setClassifier(person_classifier);        // set person classifier
people_detector.setInputCloud(cloud);
people_detector.setGround(ground_coeffs);                // set floor coefficients
people_detector.compute(clusters);                      // perform people detection
// Optional settings:
people_detector.setVoxelSize(voxel_size); // set the voxel size (0.06)
people_detector.setHeightLimits(min_height, max_height); // set height limits (1.3, 2.3)
people_detector.setDimensionLimits(min_points, max_points); // set dimension limits (30, 5000)
people_detector.setMinimumDistanceBetweenHeads(min_dist); // set minimum distance between persons’ heads (0.3)
people_detector.setSensorPortraitOrientation(true); // set portrait/landscape orientation (false)
// Display pointcloud:
pcl::visualization::PointCloudColorHandlerRGBField<PointT> rgb(cloud);
viewer.addPointCloud<PointT> (cloud, rgb, "input_cloud");

// Add point picking callback to viewer:
struct callback_args cb_args;
PointCloudT::Ptr clicked_points_3d (new PointCloudT);
cb_args.clicked_points_3d = clicked_points_3d;
cb_args.viewerPtr = pcl::visualization::PCLVisualizer::Ptr(&viewer);
viewer.registerPointPickingCallback (pp_callback, (void*) &cb_args);

// Spin until 'Q' is pressed (to allow ground manual initialization):
viewer.spin();
// CMakeLists.txt

cmake_minimum_required(VERSION 2.8 FATAL_ERROR)
project(pcl_ground_based_rgbd_people_detection)

find_package(PCL 1.7 REQUIRED)
include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})

add_executable(pcl_ground_based_rgbd_people_detection
src/main_ground_based_rgbd_people_detection.cpp)
target_link_libraries(pcl_ground_based_rgbd_people_detection ${PCL_LIBRARIES})
// CMakeLists.txt
cmake_minimum_required(VERSION 2.8 FATAL_ERROR)
project(pcl_ground_based_rgbd_people_detection)

find_package(PCL 1.7 REQUIRED) include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS}) add_definitions(${PCL_DEFINITIONS})

add_executable(pcl_ground_based_rgbd_people_detection src/main_ground_based_people_detection.cpp)
target_link_libraries(pcl_ground_based_rgbd_people_detection ${PCL_LIBRARIES})

// Execution:
$ ./pcl_ground_based_rgbd_people_detector
Results
**Input**
- XYZRGB point cloud
- Ground coefficients

**Output**
- People clusters (centroid, points indices, ...)

**Framerate** *(on Intel Core2 @ 2.5GHz)*
- About 15 fps (VGA resolution, data grabbing and visualization not included)
- About 20 fps (with QQVGA resolution) on the original ROS-based implementation
Live Demo!
Any questions?

matteo.munaro@dei.unipd.it