Segmentation of Humans from LIDAR Point Clouds Using Visual Pose Estimation

Gaini Kussainova, Luis E. Navarro-Serment and Martial Hebert

Research objective

- Detection of humans is an important problem which has many applications, such as motion tracking and activity recognition.
- Segmenting point clouds is challenging due to data noise, sparseness and uneven density of points.
- Most human segmentation algorithms use 3D human modeling which requires a high computational budget.
- In this work, we built a fast, computationally inexpensive and reliable algorithm for human segmentation from point clouds by leveraging human pose estimations from images. Our algorithm does not use any appearance-based descriptors. Instead, it relies only on the geometric relationship between body parts detected by OpenPose.[1]

Approach

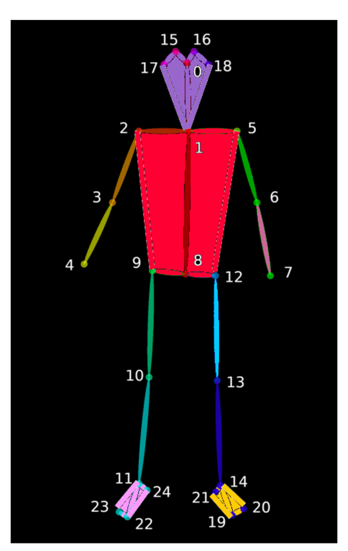

Fig.1. The 25 human body key points detection and 12 Regions.

The algorithm operates on data from a calibrated camera – LIDAR set. Images are processed to detect humans and to estimate their corresponding poses. This process is computationally expensive; therefore this result is further used to find the points in the point cloud which most likely belong to the human detected in the image.

- Project 3D point cloud data on image plane
- First stage of the segmentation:
  - Discard points far away from the skeleton
  - Detect points lying in one of 12 regions of interest (Fig. 1).
- Second stage of the segmentation:
  - Analyze 'human' labeled points from the first stage and discard the unlikely ones depending on the distances between points in 3D space.

Experiments

- We performed experiments on the KITTI[2] dataset. The dataset contains 3D data from a Velodyne 64 and does not contain ground truth for segmentation of point cloud data.
- We annotated by hand 10 point clouds from the dataset that included various representative examples of human pose, distance and orientation to sensors.

Results

- We used completeness to characterize the performance of the algorithm. The completeness represents the percentage of all **'human'** points that were labeled as **'humans'**. The completeness for our algorithm is 89.38%. Other statistics are presented in Table 1.
- It is worth mentioning that the performance subsequently depends on the accuracy of the pose estimation. A more thorough evaluation is needed and is considered part of future work.

	MAX (%)	MIN (%)	STDEV (%)
Completeness	95.5	83.6	4.96

Table 1. Segmentation performance.

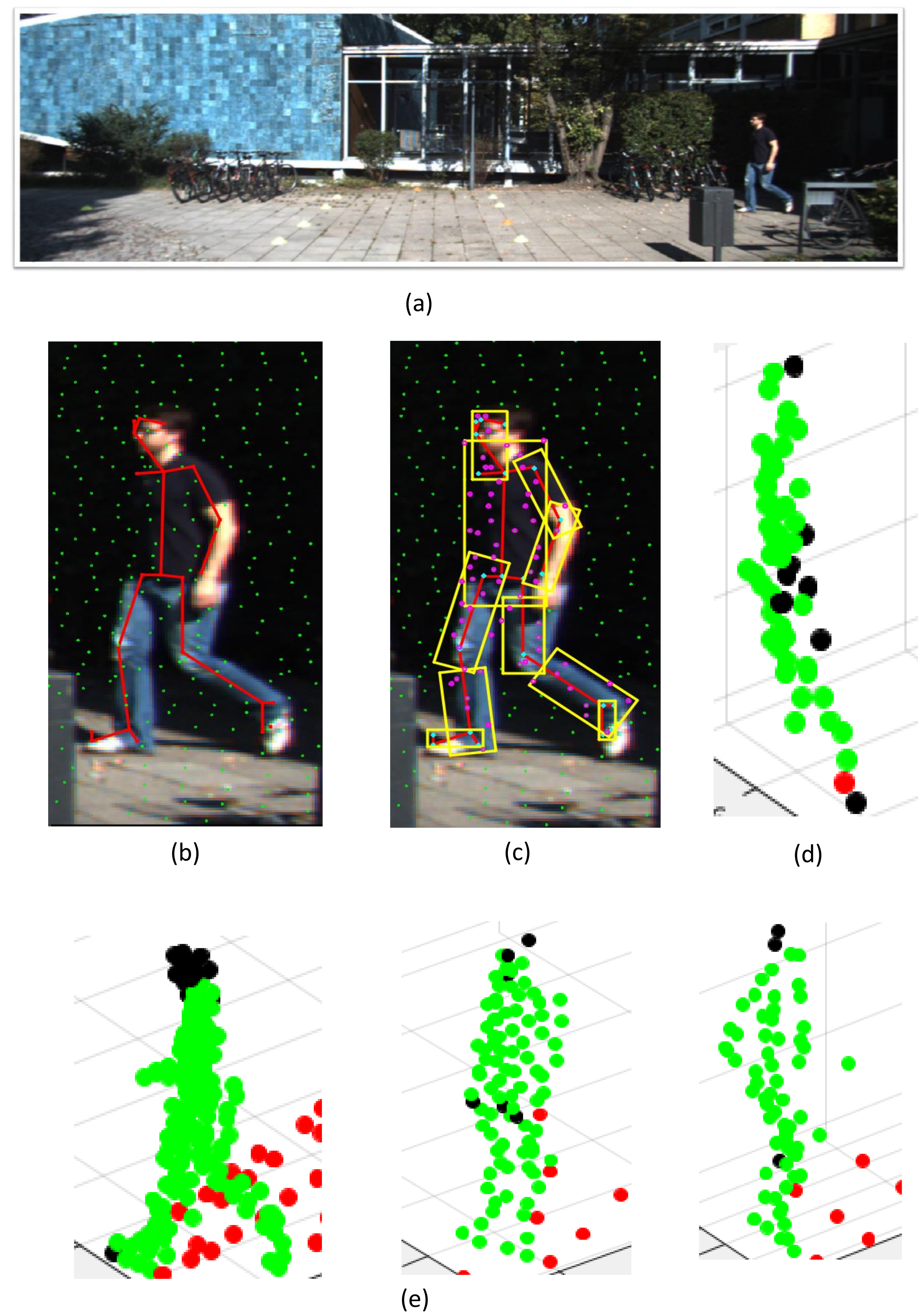

Fig. 2. The intermediate steps of the algorithm: (a) is the original image; (b) a region of the image showing the skeleton points (red) from OpenPose and the 3D point cloud data projected onto the image plane (green); (c) bounding boxes (yellow) around every body region and points (magenta) labeled as 'human' after the first stage of the algorithm; (d, e) the zoomed output of the algorithm: human black points – false negatives, green points – points correctly labeled 'human', red points – ground points.

Pseudo code for the proposed algorithm

```
Algorithm 1: Human Segmentation Algorithm
Input    : P_k = {p_1, p_2, .., p_k} - point cloud, where
           p_i =< x_i, y_i, z_i >
           S_m = {s_1, s_2, .., s_m} - pose estimations
           where s_j =< u_j, v_j >
Output:    HS_n = {hs_1, hs_2, .., hs_n}  segmented point
           cloud, where hs_l =< x_l, y_l, z_l >
Parameters: PR_k = {pr_1, pr_2, .., pr_k} - point cloud
            projected on image plane, where
            pr_j =< u_j, v_j >
            boundBox - [x_min, x_max, y_min, y_max] -
            skeleton bounding box
            R = {R_1, R_2, .., R_12} - bounding boxes
            around every body part
            factor - factor of region expansion
            H = {H_1, H_2, .., H_n} - human labeled
            points in image plane; H_i =< u_i, v_i >
Begin:
READ skeleton file
removeUndetectedSkeletonPoints(S_m)
PR_m = PROJECT(P_k, calibrationMatrices)
crop(PR_m, skeleton bounding box)
expandBoundingBoxes(R, factor)
for p_i in PR_m do
    for r_j in R do
        if p_i belongs to r_j then
            H_p[r_j].append(p_i)
        end
    end
end
meanPoint(H_p)
meanRegion(R)
for p_i in PR_m do
    for r_j in R do
        if meanPoint(p_i) < meanRegion(r_j) AND
         p_i in r_j then
            HS_n{p_i} = human
        end
    end
end
return HS_n
End

meanPoint(p):
for p_i in R do
    mean = mean(dist(p_i, p))
end
return mean

meanRegion(R):
for all combinations(p_i, p_j) in R do
    mean = mean(dist(p_i, p_j))
end
return mean
```

Future work

- Increase robustness against inaccurate body point detections.
- Enhance performance by considering other geometrical relationships between points, such as shape.
- Expand annotated dataset for more thorough evaluation.
- Include projection of pose estimations to 3D space to to increase the accuracy of detections.

https://github.com/CMU-Perceptual-Computing-Lab/openpose
http://www.cvlibs.net/datasets/kitti/

RISS ROBOTICS INSTITUTE SUMMER SCHOLARS