FAST POINT CLOUD MULTI-PERSON FILTERING

Andrea Rigo
andrea.rigo@studenti.unitn.it, 222101

Eliana Battisti
eliana.battisti-1@studenti.unitn.it, 223701

University of Trento
Course: Computer Vision
A. Year: 2020/2021

ABSTRACT

The objective of this work is to devise a fast technique to extract all people present in a 3D point cloud given their skeleton, filtering out the rest. While the majority of existing approaches are mainly neural, we propose two geometric approaches, where the person's skeleton is taken from Joo et al. [1]; while our work filters points geometrically by exploiting the skeleton's information. The two proposed approaches are:

1. cylinder-based: we check if each point belongs to a person's body using a system of equations. This approach proved to be too slow.
2. bounding-box-based: we leverage a popular and modern library called Open3D [2] to define bounding boxes and filter the points inside them very efficiently, achieving a speedup factor of about 1083 on point clouds with three people w.r.t. the first approach we tried

INTRODUCTION

Point clouds have recently gained attention due to the development of 3D acquisition technologies (LiDARs, RGB-D cameras, 3D scanners,…) and they have a lot of applications in a wide range of areas in the fields of computer vision, autonomous driving and robotics.

Semantic segmentation is the process of classifying each 3D point in a point cloud with a label defining to which object a particular point belongs. It distinguishes different categories of objects, but not different instances of the same category.

Point clouds can also be exploited to build a model of the human body for various purposes, i.e. by fitting meshes to a large variety of point clouds representing people.

This work lies somewhere in between these two topics. We essentially perform semantic segmentation by classifying each point as belonging to a person or not. We do this by modelling the human body with a set of geometrical shapes with position and dimensions determined from the skeleton joints, so that each of the shapes wraps a different part of the body. The points inside are then considered as belonging to a person.

RELATED WORKS

Nowadays the most used algorithms to solve the Semantic Segmentation problem use deep learning techniques and neural networks:

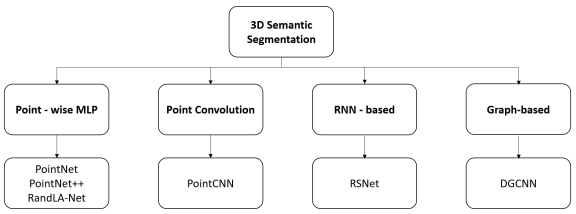

Figure 1

Given a point cloud of a scene, the goal is to separate it into several subsets according to the semantic meanings of points. There are different techniques: Point-wiseMLP, Point Convolution, RNN-based and Graph-based. Since our approach is purely geometric we will focus more on the body modeling approaches.

Regarding body modeling, Loper et al. [3] propose Skinned Multi-Person Linear model (SMPL), a

learned model able to create realistic animated human bodies that can represent different body shapes, deform naturally with pose, and exhibit soft-tissue motions like those of real humans. The SMPL model animates different body shapes by decomposing each one into identity-dependent shape, and pose. This allows the authors to generate new skinned body meshes by blending an initial artist-created template mesh with different identity shapes and desired poses. They do so by learning blend shapes from a large dataset, which are vectors of meshes' vertex displacements. They learn two different sets of blend shapes: pose blend shapes and shape blend shapes. SMPL then learns the contribution of blend shapes, corrects skinning errors and produces realistic pose information. These are then used with standard methods to deform the original template and thus produce the final mesh. The contribution of pose blend shapes changes with pose, allowing to avoid the typical artifacts of standard techniques. SMPL achieves better accuracy and realism w.r.t. previous state-of-the-art approaches. All parameters are learned from data but the techniques used to deform the template are all standard, making the approach very efficient and fully compatible with all the major existing graphics pipelines.

In Hogg et al. [4] work, one of the first works in this topic, the authors used a much simpler model where each body part was represented by a cylinder:

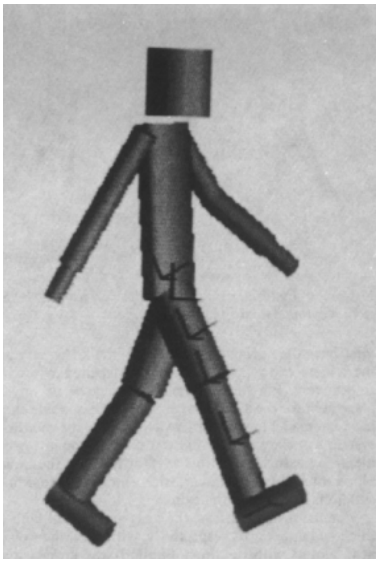

Figure 2

Since we start from clean point clouds from [1] and the objective is to devise a very fast algorithm, we used a simple model similar to the one of Hogg et al.

DATASET

For this project we used the CMU Panoptic Dataset [1], a dataset created to measure social interactions through the perceptual integration of a large variety of viewpoints. The dataset was collected by capturing people engaged in various social interactions in the panoptic studio, a dome equipped with 480 cameras and 5 Kinect Ⅱs. The dataset contains a variety of point clouds, each with skeletons, facial and hand key points. How the skeletons and keypoints are computed is outside of the scope of this work but we will roughly describe it for completeness.

The system takes 480 synchronized video streams of multiple people as input and it produces skeletal trajectories with an associated set of labelled 3D trajectories for each body part. The structure of the algorithm to create skeletons follows these steps:

1) For each of the 480 views available, use a 2D pose detector to compute joints' score maps: probability maps showing where are the probable locations for each joint
2) Combine score maps from all views to obtain 3D score maps
3) Use Non-Maxima Suppression and thresholding to extract the most probable node (joint) proposals
4) Generate part proposals by connecting a pair of node proposals. The pose detector generates a connectivity score for each pair in each view
5) Keep the part proposals that have a good score on most views
6) Propagate part proposals in time to generate part trajectory proposals and score their validity using the 3D score maps
7) Considering the model as an undirected graph, use Dynamic Programming to select the best combination of part trajectories, obtaining the skeletal trajectory proposals

8) Score skeletal proposals with a distance function and keep the best using Non-Maxima Suppression and thresholding

IMPLEMENTATION AND TESTING

The chosen dataset comes with skeletal keypoints computed from neural networks and other complex computations. In addition to that, computing a point cloud from such a high number of cameras is computationally expensive. Since the objective of this work is to be efficient and fast enough to work in real time, we decided to use a geometric approach to avoid additional overhead on top of [1].

We devised two methods, the first based on a system of cylinder equations which however proved to be too slow and lead us to develop a second, way faster approach based Open3D's bounding boxes.

CYLINDER-BASED APPROACH

The first algorithm that we implement constructs a cylinder and two planes perpendicular to the cylinder in order to truncate it.

Each skeleton is composed of: 19 joints plus some additional points to model hands. Each human body part, i.e. arms and legs, is described by two joints.

For each pair of points corresponding to a body part, we construct a cylinder with the line connecting the two joints as the central axis, obtaining a model of the human body composed of cylinders.

Then, we truncate it by defining two planes centered in the two points and perpendicular to the cylinder. This way we obtain a truncated cylinder of the same length of the body part we are filtering:

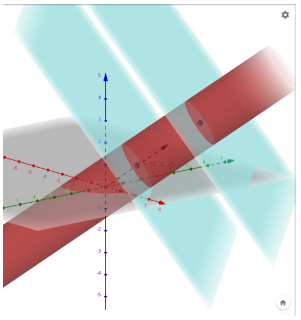

Figure 3

In the case of wrists, knees and shoulders which are represented by a single point we simply used a sphere centered in it instead. Head and hands instead are represented by a handful of points. We computed their centroids and used a sphere centered on each of them.

Then we mark all the points inside this truncated cylinder as belonging to a person:

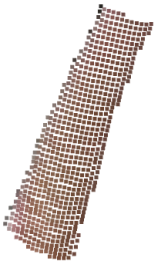

Figure 4

All the geometrical shapes mentioned above are defined by their equation in the three dimensional space and therefore to check if a point belongs to the space enveloped by a shape we used the following inequalities:

$$\begin{cases} n \cdot (x - x_1) \leq 0 \\ n \cdot (x - x_2) \geq 0 \\ \frac{\|(x-x_1)\times(x-x_2)\|}{\|x_2-x_1\|} - r \leq 0 \end{cases}$$

where the first two inequalities are the planes' equations, the third one is the cylinder's equation, x_1 and x_2 are the body part's joints, n is the joints' axis used as normal of both planes, and r is the cylinder radius.

The library we use to construct the cylinders, planes and spheres is NumPy.

The main problem of this approach is that it's very slow. The time it took to process a single frame (point cloud) containing three people is about 32,5 sec on an average consumer PC. The algorithm has such a high time complexity because it computes all the geometrical shapes' equations for each single point and has to repeat the process for each person in the scene. Thus, with three people it processes the entire point cloud three times.

BOUNDING BOX-BASED APPROACH

In order to reduce the time complexity, we exploited two functions already implemented in Open3D [2] which are optimised. The functions are the following:

- create OrientedBoundingBox(): it creates an OrientedBoundingBox object from center, rotation R and extent in x,y and z.
- get_point_indices_within_bounding_box(self, points): returns indices of the points that are within the bounding box

The idea is basically the same as the previous approach. We define a bounding box around a body part and extract the points that fall inside it. To create a bounding box using the first function we need a center and a rotation matrix in order to align the box with the body part axis, the line connecting the two joints. The center is simply the median of the two joints.

By default, in Open3D a bounding box with no rotation is oriented along the z axis. Therefore we compute the axis of rotation between the z axis and the body part axis using the cross product:

$$h = v \times z$$

where v is the joints' axis and z is the z axis.

Then we compute the angle of rotation between the two vectors using the dot product:

$$\theta = arccos\ \frac{v \cdot z}{\|v\|}$$

Once we have a rotation axis and angle, we need to compute the corresponding rotation matrix to create the bounding box oriented in the correct way. To do this we used the Euler-Rodrigues Formula. The formula describes the rotation of a vector in three dimensions with four Euler parameters: a, b, c, and d:

$$\vec{x}' = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2+c^2-b^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2+d^2-b^2-c^2 \end{pmatrix} \vec{x}$$

Where the Euler parameters can be computed directly from the axis of rotation and angle:

$$a = cos\frac{\varphi}{2}$$

$$b = k_x\, sin\frac{\varphi}{2}$$

$$c = k_y\, sin\frac{\varphi}{2}$$

$$d = k_z\, sin\frac{\varphi}{2}$$

This way we obtain the rotation matrix corresponding to the axis and angle of rotation efficiently. Regarding the dimensions of the boxes, we set the extent along z equal to the length of the body part and the extent along the other two axes is set to the same value multiplied by a scalar. The scalar should represent the typical length-width ratio of each body part. We determined the ratio to use empirically. In case of parts represented by a single point, we apply the same method but we use a cubic bounding box. By using this ratio trick the boxes should better fit each individual.

Once we build the bounding box, we use get_point_indices_within_bounding_box to extract the points that fall inside it and repeat the process for all body parts.

This approach took 0.03 sec to filter the same point cloud of the other algorithm on the same machine.

RESULTS

In figure 5 there is a point cloud as input; The figure 6 shows the output: the point cloud without the background. With respect to the first approach, the second performed about 1083 times better than the first. For completeness we should calculate the asymptotic computational

complexity of both algorithms to compare them, unfortunately this is not trivial because we don't know the computational complexity of the Open3D functions we used.

Figure 5

Figure 6

FUTURE WORK

While the ratio trick allows the bounding boxes to change dimensions depending on the body part length and thus fit different people, having fixed ratios is not an ideal solution. Different people might not have the same length-width ratio. Therefore a possible future development of this work could be to set each ratio, or the bounding boxes' dimensions directly, with an adaptive technique.

CONCLUSION

In conclusion, by exploiting skeleton information and Open3D optimized functions, we obtained a simple and yet fast and efficient filtering algorithm that adds very little overhead on top of the skeleton computation and can work in real-time applications.

REFERENCES

[1] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Panoptic Studio: A Massively Multiview System for Social Motion Capture, ICCV 2015

http://domedb.perception.cs.cmu.edu/tools.html

[2] http://www.open3d.org/

[3] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, Michael J. Black. SMPL: A Skinned Multi-Person Linear Model. 2015

[4] David Hogg, Model-based vision: a program to see a walking person, 1983