PCD: Partially Covered Detection Of Humans Using Point Cloud Data

Matthew Bradbury
Tarnveer Takhtar
Harman Bassi
Kyen So

1. Background

Point cloud data refers to a data type that represents the collection of 3D points in space. This data format is often used to create detailed 3D models of environments or objects. The aim of our project is to use these point clouds in order to consistently, and correctly, detect a human person when partially covered. More specifically, we want to be able to reliably detect humans, whether they are partially covered or not, and be able to estimate how much space they occupy. Additionally, we will be using traditional point cloud functions as well as 2D and 3D manipulation techniques offered by PCL and OpenCV to accomplish this task. We are also excluding the use of pre-trained artificial intelligence and machine learning models.

There has been significant research into this particular area of study, but those pre-existing papers focus on utilizing and leveraging machine learning techniques to facilitate the detection [1][2][3][4][5][6]. Furthermore, many of these papers focus more on handling a wide variability in human poses [7] as opposed to looking into partially obscured detection, or are using input methods that do not align with our project [8]. Some of the techniques we employ for our process are inspired by ideas in other documentation. For inspiration on proper skin detection, we looked at a paper talking about human skin detection using RGB and HSV [9]. Furthermore, when implementing region growing, we turned to the PCL documentation to help with proper application of the functions [10][11].

Regarding our project specifics, we will be supporting both off-line and live detection of people. For the off-line option, a PCD file can be inputted and our algorithm will display the people detected within it. For the live option, a continuous stream of point cloud data is read in and the detection is displayed and updated in real time. As for our restrictions, on top of the aforementioned limitation on deep learning, the live PCD will be supplied by a Kinect. In particular, we will be using the Kinect V2 model for our project.

2. Methodology

Prior to our process, we have to get our input cloud from the Kinect. We utilize the kinect2_grabber library header [12], which was modified by our team to be compatible with the latest version of PCL (Point Cloud Library). We use this grabber to pull raw point cloud data from the Kinect and parse that information into its correct PCL types.
Our process can be split up into multiple distinct steps:

I. OpenCV Filtering

The first step in our process is to filter the cloud in OpenCV. This helps the ensuing seed point acquisition be more reliable and accurate. The specific filtering that we perform in this step

is plane filtering. We use our own custom function to detect planes and remove them from the ensuing seed point acquisition to ensure that the detected seed points are not on planes.

This custom function detects planes based on checking the depth gradients, surface normals, depth variance, and depth discontinuity. First, we use sobel operators to calculate the spatial depth gradients to identify abrupt changes, filtering out non-planar regions. These gradients are then normalized by depth to account for bias, ensuring consistent thresholds across varying depths. We then move on to check surface normal consistency. These normals are approximated from depth gradients and normalized to detect regions with uniform orientation. With this information on the normals, angular consistency checks occur to reject non-planar areas by comparing normals across local neighborhoods. After this, we compute depth variance. This local variance is used to distinguish textured surfaces (eg. clothing) from flat planes. The low variance regions found here are retained as potential planar candidates. Finally, we detect depth discontinuity. Adaptive thresholds are used to identify edges by comparing neighboring depth values, which filters out boundaries of objects. Using all this gathered information, we guess that the only retained objects are our best guess at where the planes are. To finish off this function, we use morphological operations (closing/opening) to refine the plane mask, removing noise and filling gaps, as well as visualizing/marking off distinct planes labelled by connecting components to show where skin points cannot lie.

All of this concludes in us having a good guess on where the planes lie inside of the cloud. Using the placement of these planes, we can rule out later skin points that are detected to be on these planes, since they would not be attached to a person. Furthermore, we can perform noise removal along these planes to remove excess points.

II. Seed Point Acquisition (OpenCV)

The next step is the aforementioned seed point acquisition. This function uses various HSV masks in order to confidently find skin points on a human to use as seed points for later functions. These points serve as the starting node for our process to detect the whole human.

Initially, we start with determining our initial skin mask. This mask uses preset HSV colour space ranges (determined via trial and error) to isolate skin-toned pixels, leveraging hue , saturation, and value ranges typical of human skin. After this, we move on to filtering out the red region of the mask. Since red hues are included in the HSV skin range we have defined earlier, we require another mask to filter these out, reducing false positives. Morphological closing merges fragmented red regions for robust exclusion. Finally, a filtered skin mask is created by intersecting both the HSV skin mask and the inverse of the red region mask.

To filter down the points even more, we use a final confidence mask. This mask utilizes averages of local intensity to identify high-confidence filtered skin regions, for the purpose of removing sparse noise. Thresholding and morphological smoothing create a refined binary mask that contains our best guess at where the skin points on a person lie. Finally, contour analysis is performed from the skin points to further reduce the chance of false positives. Each contour is created by approximating a cluster of detected skin points with a characteristic centroid. By

looking at the area and convexity of the produced contours we filter out smaller ones, which may be artifacts or false positives. The remaining centroids, which are assumed to be from valid contours, are then projected into 3D using depth data to seed PCL region growth later on.

III. PCL Filtering

Prior to using the previously discovered skin points, we first perform some filtering in PCL. We use a variety of methods to mitigate and remove noise from the point cloud in order to reduce the complexity of the image, leading to faster processing and more accurate region growing in following steps. Specifically, we perform voxel downsampling, noise removal, and plane removal to achieve this.

Firstly, utilizing voxel downsampling, the point density of the cloud is reduced uniformly, balancing computational efficiency with spatial resolution. Next, noise is further removed through statistical outlier removal and radius filtering. Statistical outlier removal eliminates isolated points, such as noise from the sensor. Radius filtering removes sparse outliers that do not belong to coherent structures/clusters.

IV. 3D Cluster Extraction (region growing)

After filtering out noise and other points from the point cloud, we move on to 3D cluster extraction. Using the previously detected and mapped seed points, we perform region growing starting from said seed points. This grows the skin regions (mapped out by skin points) into full human clusters.

Before region growing, multi-radius searches are performed around the mapped seed points to ensure robust initialization in noisy data. To grow the region of the person from the seed points, normals and euclidean distance thresholds are used to guide the region expansion. This ensures geometric consistency during the region growing. The region growing itself follows a queue-based approach that prioritizes neighboring points, leading to clusters being grown iteratively. During this process, the growth direction is tracked for future use in region estimation. Growth vectors and neighborhood centroids are stored to analyze expansion patterns. Spatial hashing aggregates centroids into simplified clusters, which are then used to form simplified neighborhood centroids, reducing computational complexity and simplifying analysis for the region estimation process.

Finally, clusters are validated based on size and dimensional constraint. These constraints include maximum height/width, maximum area, and meeting the minimum number of cluster points. This validation is used to further eliminate noise and remove clusters that are too small/large to be a person. Before moving on, we try to recover some of the points that surround the cluster of points that make up the person. This growing and noise removal can be aggressive at times, making it hard to see the person's cluster in the final visualization, and so we recover some points that surround the person's cluster to make visualization easier.

V. Region Estimation

Finally, after defining the points in the cloud that make up the person, we perform various analytical techniques to determine the growth direction of the cluster. First, we perform a line of best-fit analysis against our simplified neighbourhood centroids. These simplified centroids are derived by clustering our current neighbourhood centroids to form a simplified set of centroids, reducing the number of neighbours to analyze and simplifying the process for determining a line of best fit to perform a cluster linearity analysis.

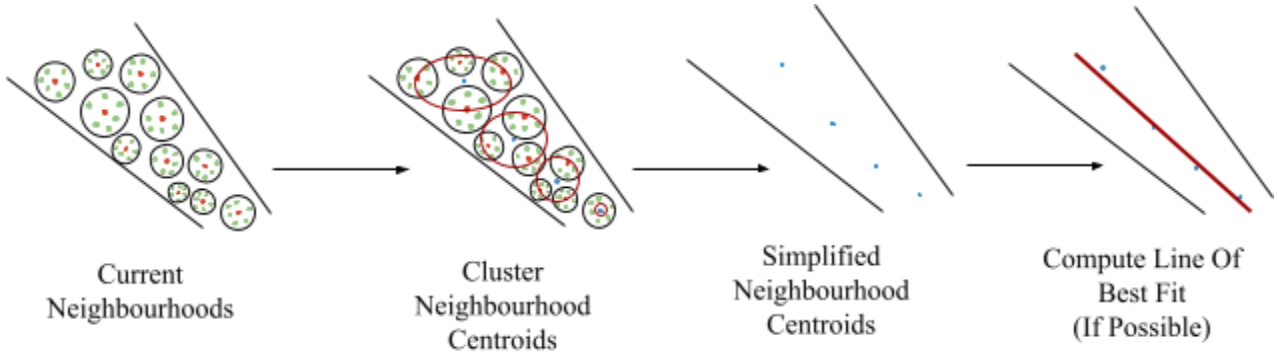

Figure 1: Simplified Neighbourhood and Line Of Best Fit Determination

If a line of best fit can be computed, which determines the likelihood of a limb being visible, we then analyze the angle of the line of best fit (normalized from 0-90 degrees) to classify the line as horizontal or vertical. Currently, with a 90 degree reference angle, an angle between 0 and 75 degrees is classified as horizontal, potentially indicating the human is further to the right/left of the visible limb.

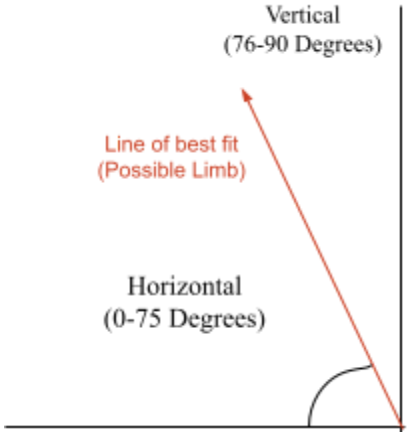

Figure 2: Horizontal Angle Thresholds For Potential Limb Clusters

Second, we perform a general growth analysis, utilizing manually accumulated growth directions from neighborhood expansions that are averaged to capture localized movement trends. We also employ PCA (principal component analysis) on our simplified neighborhood centroids (aggregated via spatial hashing) to compute the dominant growth direction. We blend the observed growth vectors from the growth directions and the dominant direction from PCA to to balance global and local patterns.

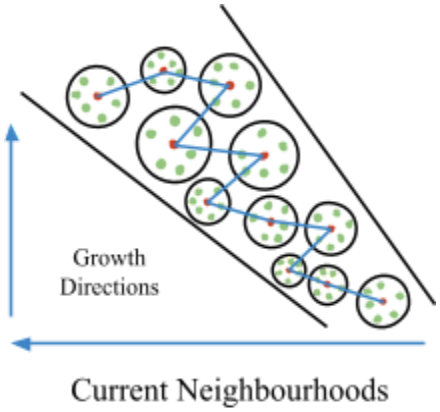

Figure 3: Neighbourhood Pathing and Directional Vectors

After performing both analyses, we then determine the placement for the estimation of where the occluded human lies. We have simplified our estimated region to be central, relative to the visible portion of the human, or offset (left/right), relative to the visible person (e.g. arms or legs sticking out away from the body). When a line of best fit can be established using simplified neighborhood centroid patterns, the offset direction is resolved through a multi-step process: the furthest seed-centroid pair identifies spatial extremes, with relative x-coordinate differences between centroid and seed dictating left/right bias. Consistency is validated by comparing the x-component of the average growth vector (derived from neighborhood expansion trends) to a threshold. A weighted decision matrix then integrates three metrics: furthest pair direction (40% weight), average growth vector alignment (40%), and line slope (20%), to compute a final score with positive/negative values determining lateral placement. Conversely, in cases where linearity cannot be determined (e.g. large scattered centroid patterns such as the torso), the system defaults to central placement. When the system defaults to, or confidently determines, central placement through verticality of the estimated region, it is doing so with the assumption that enough of the body is visible for it to lie at the centroid and be drawn in the centre of the visible cluster. Once the estimated region is determined, visible feedback via 3D bounding boxes are drawn to highlight these estimated regions.

3. Results

In this report, we have outlined our novel method for the algorithmic detection of humans. Using this method, we have a relatively fast and accurate way to detect humans, whether partially obscured or not, completely without the use of trained models. Regarding the accuracy of our process, we sometimes get false positives from the skin detection, leading to an incorrectly assumed person, as well as some minor fragmentation from the background.

The actual output of the process displays the points that we believe belong to the detected person as well as a bounding box surrounding them representing the area they may inhabit. Through our use cases we've identified 3 distinct scenarios: when the person is fully visible,

partially obscured, and mostly out of frame. For each output, we have the ability to both display the individual points/clusters that were region grown with the estimation box, as well as the estimation bounding box surrounding the person on the original cloud.

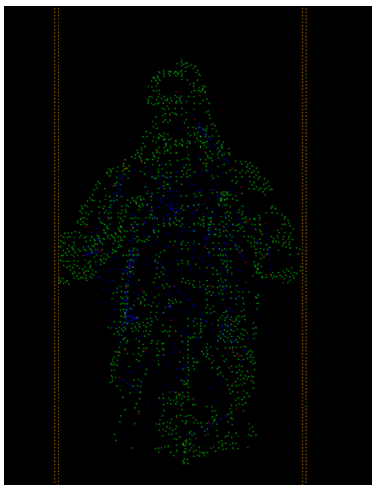

Figure 4: Clustered Output

Figure 5: Estimation on Original Cloud

In terms of accuracy, our process is successful most of the time, within the scope of its abilities. More specifically, when the background is relatively uncomplicated and when the person has skin that is visible to the camera. The accuracy decreases with more complex backgrounds, i.e. condensed areas with a high density of objects and areas that have colour palettes similar to various skin tones. Furthermore, with the process that we have demonstrated, the person will not be detected if they have no skin points visible to the Kinect.

Regarding the performance of our process, we find that the processing time stays low regardless of the complexity of the environment, for both offline and live uses. Regarding offline processing, our testing showed that it took about 0.5 seconds to fully process the cloud. This number does not include the time it takes for the cloud to be visualized and displayed. Furthermore, for live processing, we found that it would take 0.2 seconds to process per frame on average, with a worst case of 0.74 seconds/frame. This data all satisfies one of our initial performance constraints by keeping the processing latency low.

4. Discussion

Moving forward with this project, there are a few ways for the results to be improved, along with some additional directions the project can go. Firstly, the main areas of improvement center around the initial seed points. The region growing itself is fairly consistent and reliable, so most of the bottleneck comes from reliably (and correctly) finding the seed points while reducing the amount of false positives.

Additionally, the project can take some different paths in the future. One of these paths is to work on a more detailed segmentation/estimation process. Currently, we estimate just the region that we think the person is inhabiting. Expanding on this, you could refine body

segmentation to be able to identify individual body parts and thus modify the region estimation to fill in specific body parts that aren't detected.

Furthermore, another path that the project could be expanding upon is by furthering the integration with actual robots. In the initial project goal, it was stated that the technology would be used on top of robots that could have a wide variety of uses. To this end, this path would be focusing on the integration between the technology we have created and an actual robot that can move. Diving deeper, the challenges here would be twofold. One is fixing the outputs so that the robot could act upon them, i.e. when we output that the rest of the person is estimated to be to the left of the screen, the robot knows to turn to the left. The other challenge is adapting the current process to work in more dynamic environments. As part of our assumptions, we assume fairly consistent lighting and camera placement/angle. When transitioning to using a camera mounted on the head of a robot, the process we created would have to be adapted to work in more dynamic cases.

References

[1]
M. Ashikur. Rahman, “Computer Vision Based Human Detection,” *International Journal of Engineering and Information Systems (IJEAIS)*, vol. 1, no. 5, pp. 62–85, Aug. 2017, doi: 10.34894/VQ1DJA.

[2]
C. W. Wang and A. Hunter, “Robust pose recognition of the obscured human body,” *International Journal of Computer Vision*, vol. 90, no. 3, pp. 313–330, Jun. 2010, doi: 10.1007/S11263-010-0365-3/METRICS.

[3]
A. Takmaz *et al.*, “3D Segmentation of Humans in Point Clouds with Synthetic Data.” pp. 1292–1304, 2023.

[4]
“View of Real-Time Human Detection and Counting System Using Deep Learning Computer Vision Techniques.” Accessed: Apr. 02, 2025. [Online]. Available: https://ojs.bonviewpress.com/index.php/AIA/article/view/391/227

[5]
“The Complete Guide to OpenPose in 2025 - viso.ai.” Accessed: Apr. 02, 2025. [Online]. Available: https://viso.ai/deep-learning/openpose/

[6]
S. Sivachandiran, K. Jagan Mohan, and G. Mohammed Nazer, “Deep Learning driven automated person detection and tracking model on surveillance videos,” *Measurement: Sensors*, vol. 24, p. 100422, Dec. 2022, doi: 10.1016/J.MEASEN.2022.100422.

[7]
“Human3D.” Accessed: Apr. 02, 2025. [Online]. Available: https://human-3d.github.io/

[8]
Z. Song *et al.*, “an infrared dataset for partially occluded person detection in complex environment for search and rescue”, doi: 10.1038/s41597-025-04600-0.

[9]
S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia, “Human Skin Detection Using RGB, HSV and YCbCr Color Models,” vol. 137, pp. 324–332, 2017.

[10]
“Region growing segmentation — Point Cloud Library 0.0 documentation.” Accessed: Apr. 02, 2025. [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/latest/region_growing_segmentation.html

[11]
“Color-based region growing segmentation — Point Cloud Library 0.0 documentation.” Accessed: Apr. 02, 2025. [Online]. Available: https://pcl.readthedocs.io/projects/tutorials/en/master/region_growing_rgb_segmentation.html

[12]

UnaNancyOwen, “KinectGrabber/kinect2_grabber.h at Kinect2Grabber · UnaNancyOwen/KinectGrabber,” GitHub, 2025.
Accessed: Apr. 02, 2025. [Online]. Available: https://github.com/UnaNancyOwen/KinectGrabber/blob/Kinect2Grabber/kinect2_grabber.h