Obstacle Detection Planning (LRM)

Objective

Figure out an efficient way to implement obstacle detection on Lunar Rover using Sentis-ToF-M100 LiDAR sensor.

Obstacle Detection Algorithm

High Level Overview

My algorithm is designed under the assumption that the LiDAR is mounted at the centre, highest point on the front face of the Rover. I plan on using a slope-based obstacle detection algorithm that first transforms the XYZ coordinates of each pixel in an individual 160 by 120 pixel point cloud frame generated by the LiDAR, from the ground's reference frame to the LiDAR's reference frame, before calculating the pixel's slope with reference to a point on the ground 100 mm in front of it.

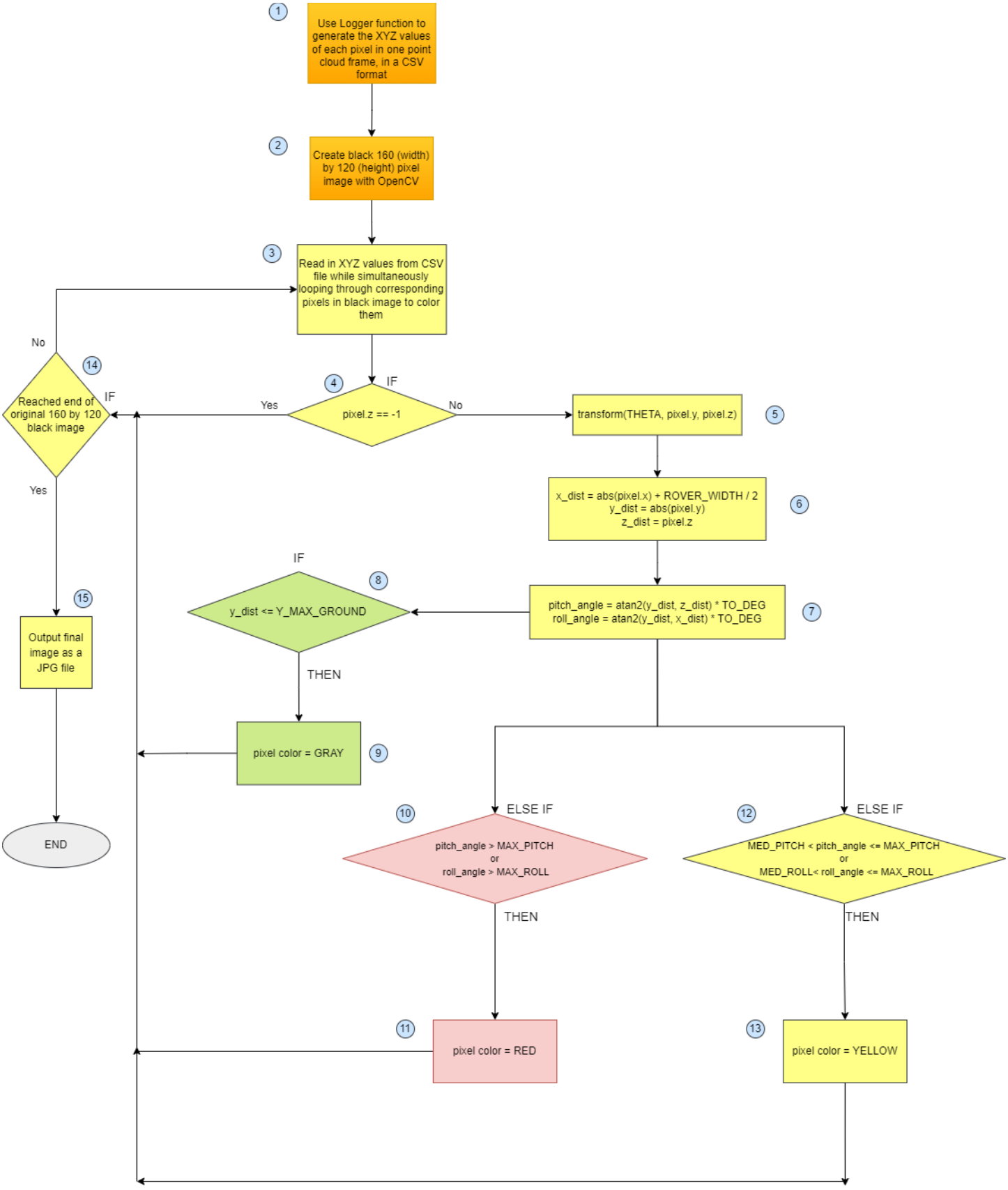
The algorithm starts by creating a black 160 by 120 pixel image. Then, the XYZ values of each pixel on the 160 by 120 pixel point cloud frame are read in while simultaneously iterating through the corresponding pixels in the black image to color these black pixels based on a few conditions.

Next, if the pixel has a valid depth value detected by the LiDAR, the pixel is transformed from the ground's reference frame to the LiDAR's reference frame. The pixel's slope with reference to a point on the ground 100 mm in front of it is used to calculate its Pitch and Roll Euler angles. Based on these angles, the pixel will then be assigned a specific color (RED = Large Pitch or Large Roll, YELLOW = Medium Pitch or Medium Roll) where red represents a large slope that should not be driven over as it exceeds either the Rover's Max Pitch or Max Roll parameter, while yellow represents a slope that should be driven over with caution as it will cause the Rover to Pitch or Roll a little but acceptable amount.

The pixels in the point cloud frame generated by the LiDAR that have no depth value, will remain black in the 160 by 120 pixel black image as they represent "unknown" areas that could be obstacles (e.g. ditch). If the pixel's absolute value height from the ground is below 40 mm, it will be colored gray to resemble an occupancy grid or indicate an obstacle-free zone that the operator could choose to drive the Rover towards.

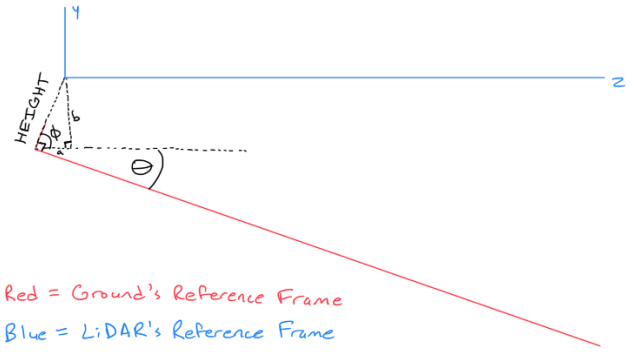
The algorithm stops after all the pixels in the original 160 by 120 pixel black image have been iterated through.

Here is a flowchart representation of the algorithm:



Here is a step-by-step explanation of each block of the algorithm:

1. Use the Logger function in the Sentis ToF API to generate the XYZ values of each pixel in one point cloud frame, in a CSV format.
2. Use OpenCV to create a black 160 (width) by 120 (height) pixel image.
3. Read in the CSV files to read in the XYZ values of each pixel in the point cloud frame generated by the LiDAR, while simultaneously iterating through the corresponding pixels in the 160 by 120 pixel black image to color these corresponding pixels.
4. Check if the pixel from the LiDAR point cloud frame has a valid depth value (z value of -1 means no depth value).
5. Since from the LiDAR's point-of-view the ground is tilted upwards by a positive angle even though the ground is completely flat in real life, this means that the LiDAR's reference frame is tilted upwards (meaning every point in the LiDAR's point-of-view is also tilted upwards) from the ground's reference frame by this angle. If the pixel has a valid depth value (z value != -1), we plug in THETA (the angle by which the ground's reference frame will be rotated about the x-axis to become parallel with the LiDAR's reference frame), the pixel's y value, and the pixel's z value into the transform(double angle, double y, double z) function which transforms the pixel from the ground's reference frame to the LiDAR's reference frame. See drawing below for explanation. The constant angle THETA was determined using the Linear Regression algorithm described later in the document.



Red = Ground's Reference Frame

Blue = LiDAR's Reference Frame

HEIGHT = Vertical height from LiDAR lens to ground

Θ = Rotation angle between ground and LiDAR reference frames

$\Phi = 90° - \Theta$

$a = HEIGHT \cos(\Phi)$ $b = HEIGHT \sin(\Phi)$

To tranform ground reference frame to LiDAR reference frame:

1) pixel.z += a

2) pixel.y += b

3) Rotate pixel by positive Θ (positive is counter-clockwise)

6. Store the magnitudes of the new x,y, and z distances of the pixel. Since the centre of the LiDAR also represents the centre point between the front wheels of the Rover, add half the width of the Rover to the x distance to store the distance from the pixel to the farthest point on the front wheel that the Rover will pivot on when rolling. Store this distance in the variable x_dist. The slope of the pixel with reference to the front wheel that the Rover will pivot on when rolling, will be used to calculate the Roll angle.

7. Take the inverse tan of the new y value divided by the new z value and convert to degrees to determine the pitch angle. Take the inverse tan of the new y value divided by the new x value and convert to degrees to determine the roll angle.
8. If the pixel's absolute value height from the ground is below or equal to 40 mm (Y_MAX_GROUND), the pixel can be considered "safe" for the Rover to drive over.
9. Since the pixel can be considered "safe", it will be colored Gray.
10. Else if the Pitch angle is greater than the Max Pitch threshold or the Roll angle is greater than the Max Roll threshold, the pixel will be considered "dangerous" indicating that the Rover should not attempt to drive over it.
11. Since the pixel can be considered "dangerous", it will be colored Red.
12. Else if the Pitch angle is between the Medium Pitch and Max Pitch threshold or the Roll angle is between the Medium Roll and Max Roll threshold, the pixel will be considered "safe but proceed with caution" since driving over it would cause the Rover to Pitch or Roll a little, but still a safe amount.
13. Since the pixel can be considered "safe but proceed with caution", it will be colored Yellow.
14. Checks if the algorithm has reached the end of the original 160 by 120 black image and continues iterating through the rest of the pixels in the 160 by 120 image if the end has not been reached.
15. Once the end of the 160 by 120 image has been reached, output the final image as a JPG file.

Estimation of Ground Plane Using Linear Regression with Method of Least Squares

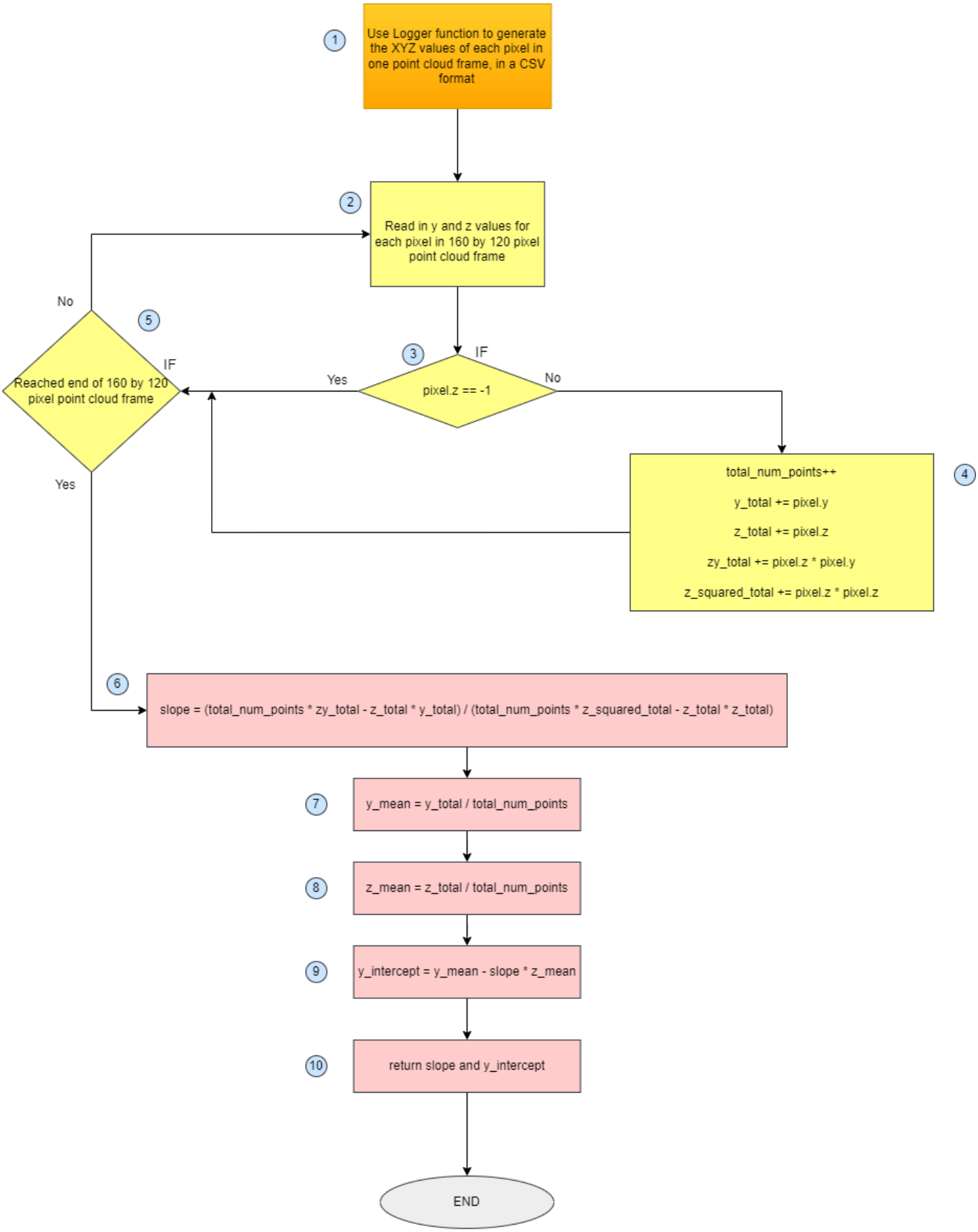
This section explains the algorithm used to determine the constant angle THETA that represents the angle by which the LiDAR's reference frame will be rotated to become parallel with the ground's reference frame. First, we need to estimate the ground plane using linear regression with the method of least squares to determine the slope of a completely flat ground plane as it appears in the LiDAR's point-of-view.

To determine the slope of a perfectly flat ground plane as it appears in the LiDAR's point-of-view, I placed the LiDAR on the ground on a completely flat surface where only this surface and no other obstacles or objects were present in the LiDAR's field-of-view.

I then used the Logger function in the Sentis ToF API to generate the XYZ values of each pixel in this field-of-view. Next, I created the linearRegression() function to read in these XYZ values from CSV files and perform linear regression on the y and z values of the pixels that have a valid depth value (the pixels with valid depth values are the pixels detected by the LiDAR that represent the ground plane), with the method of least squares.

The linearRegression() function outputs the slope and y-intercept of a line on this ground plane. The slope of this line represents the slope of a perfectly flat ground plane as it appears in the LiDAR's point-of-view. By taking the inverse tan of this slope and converting to degrees, we can determine the angle THETA by which the LiDAR's reference frame must be rotated to become parallel with the ground's reference frame.

Here is a flowchart explaining the linearRegression() algorithm in more detail:



Here is a step-by-step explanation of each block of this algorithm:

1. Use the Logger function in the Sentis ToF API to generate the XYZ values of each pixel in one point cloud frame, in a CSV format.
2. Read in the y and z values for each pixel in the 160 by 120 pixel point cloud frame generated by the LiDAR.
3. Check if the pixel has a valid depth value (z value of -1 means no depth value).
4. If the pixel has a valid depth value (z value != -1), add 1 to total_num_points (the total number of points on the ground plane that have been detected by the LiDAR), add the pixel's y value to y_total, add the pixel's z value to z_total, add the product of the pixel's z and y value to zy_total, square the pixel's z value before adding it to z_squared_total.
5. Checks if the algorithm has reached the end of the 160 by 120 pixel point cloud frame and continues iterating through the rest of the pixels in the point cloud frame if the end has not been reached.
6. Once the end of the point cloud frame has been reached, calculate the slope of a line on the ground plane.
7. Calculate the mean average of all the y values.
8. Calculate the mean average of all the z values.
9. Calculate the y-intercept of the line on the ground plane.
10. Return the slope and y-intercept of the line.