US 20210026361A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2021/0026361 A1

Ravi et al. (43) Pub. Date: Jan. 28, 2021

(54) SYSTEM AND METHOD FOR FREE SPACE ESTIMATION

(71) Applicant: **DEKA Products Limited Partnership**, Manchester, NH (US)

(72) Inventors: **Abhishek Ravi**, Manchester, NH (US); **Gregory J. Buitkus**, Dracut, MA (US); **Sai Ravi Teja Boggavarapu**, Manchester, NH (US); **Raajitha Gummadi**, Manchester, NH (US); **Derek Kane**, Manchester, NH (US); **Yu-Hsuan Chang**, Manchester, NH (US)

(21) Appl. No.: **16/938,786**

(22) Filed: **Jul. 24, 2020**

Related U.S. Application Data

(60) Provisional application No. 62/879,391, filed on Jul. 26, 2019.

Publication Classification

(51) **Int. Cl.**

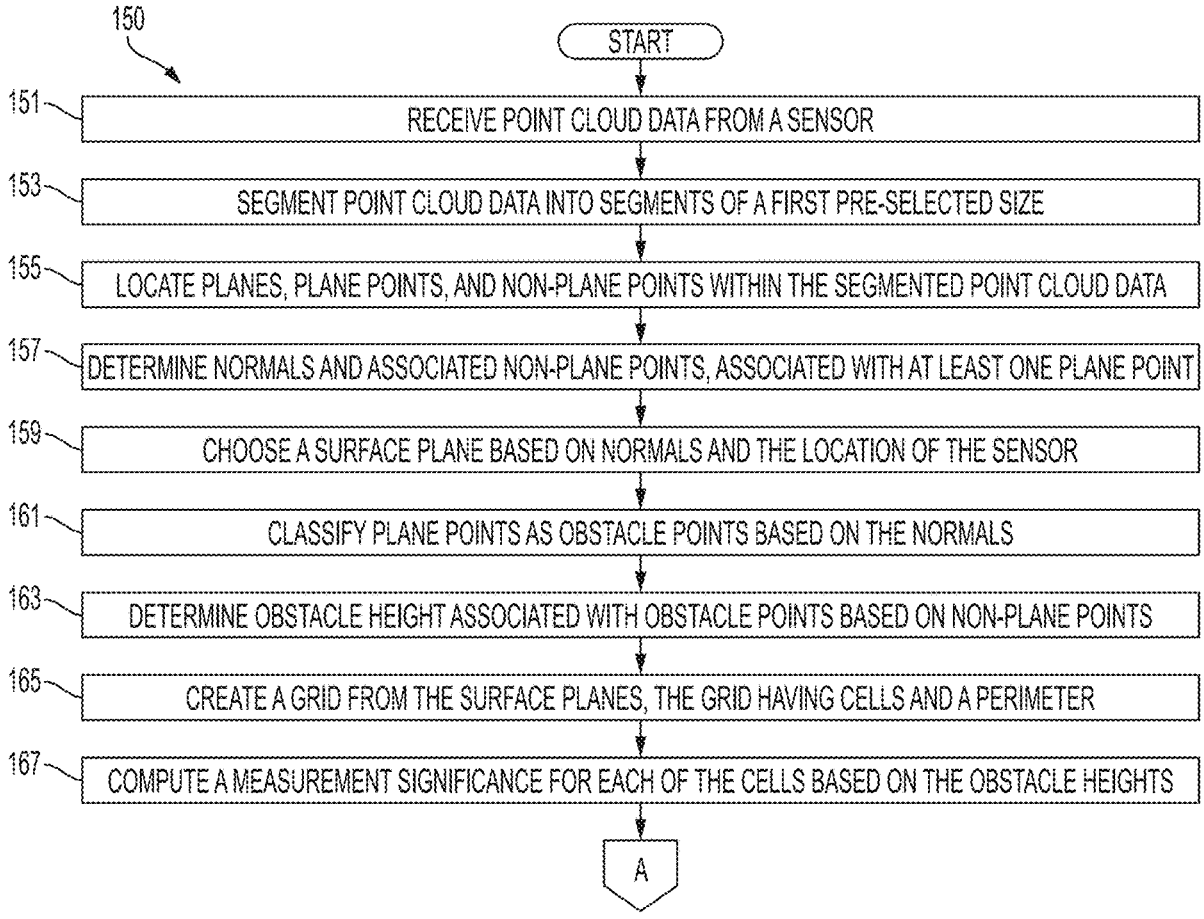
G05D 1/02	(2006.01)
G06K 9/00	(2006.01)
G06K 9/46	(2006.01)
G01S 17/931	(2006.01)
G01S 17/894	(2006.01)

(52) **U.S. Cl.**

CPC ***G05D 1/0214*** (2013.01); ***G06K 9/00805*** (2013.01); *G05D 1/0257* (2013.01); ***G01S 17/931*** (2020.01); ***G01S 17/894*** (2020.01); ***G06K 9/4604*** (2013.01)

(57) **ABSTRACT**

A system and method for estimating free space and assigning free space probabilities in point cloud data associated with an autonomous vehicle traveling on a surface, including taking into account sensor noise, sensor availability, obstacle heights, and distance of obstacles from the sensor. System and method can include determining surface planes and classifying point cloud points according to whether or not the points fall on surface planes, among other factors.



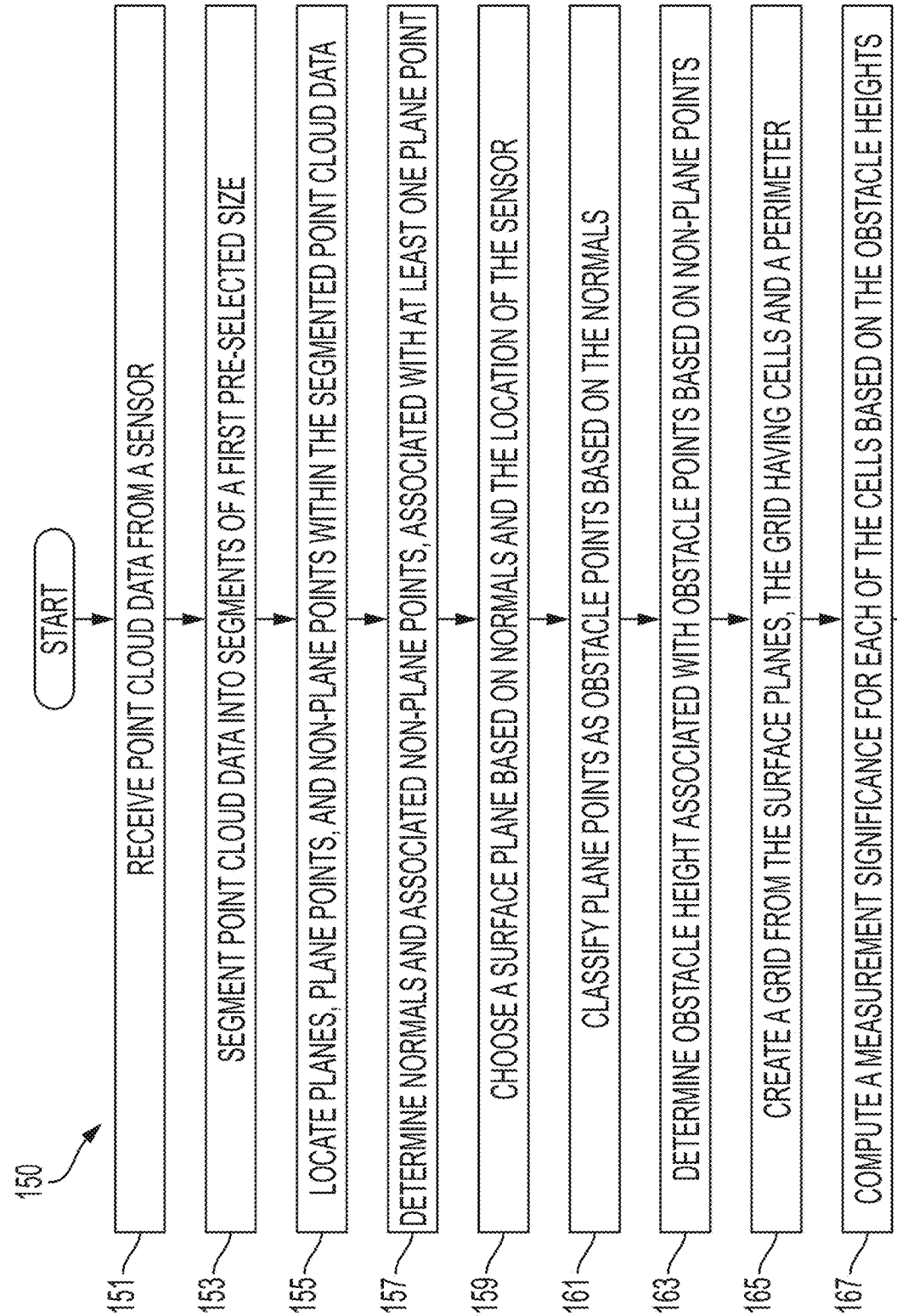

150
START
RECEIVE POINT CLOUD DATA FROM A SENSOR
151
SEGMENT POINT CLOUD DATA INTO SEGMENTS OF A FIRST PRE-SELECTED SIZE
153
LOCATE PLANES, PLANE POINTS, AND NON-PLANE POINTS WITHIN THE SEGMENTED POINT CLOUD DATA
155
DETERMINE NORMALS AND ASSOCIATED NON-PLANE POINTS, ASSOCIATED WITH AT LEAST ONE PLANE POINT
157
CHOOSE A SURFACE PLANE BASED ON NORMALS AND THE LOCATION OF THE SENSOR
159
CLASSIFY PLANE POINTS AS OBSTACLE POINTS BASED ON THE NORMALS
161
DETERMINE OBSTACLE HEIGHT ASSOCIATED WITH OBSTACLE POINTS BASED ON NON-PLANE POINTS
163
CREATE A GRID FROM THE SURFACE PLANES, THE GRID HAVING CELLS AND A PERIMETER
165
COMPUTE A MEASUREMENT SIGNIFICANCE FOR EACH OF THE CELLS BASED ON THE OBSTACLE HEIGHTS
167
A

FIG. 1A

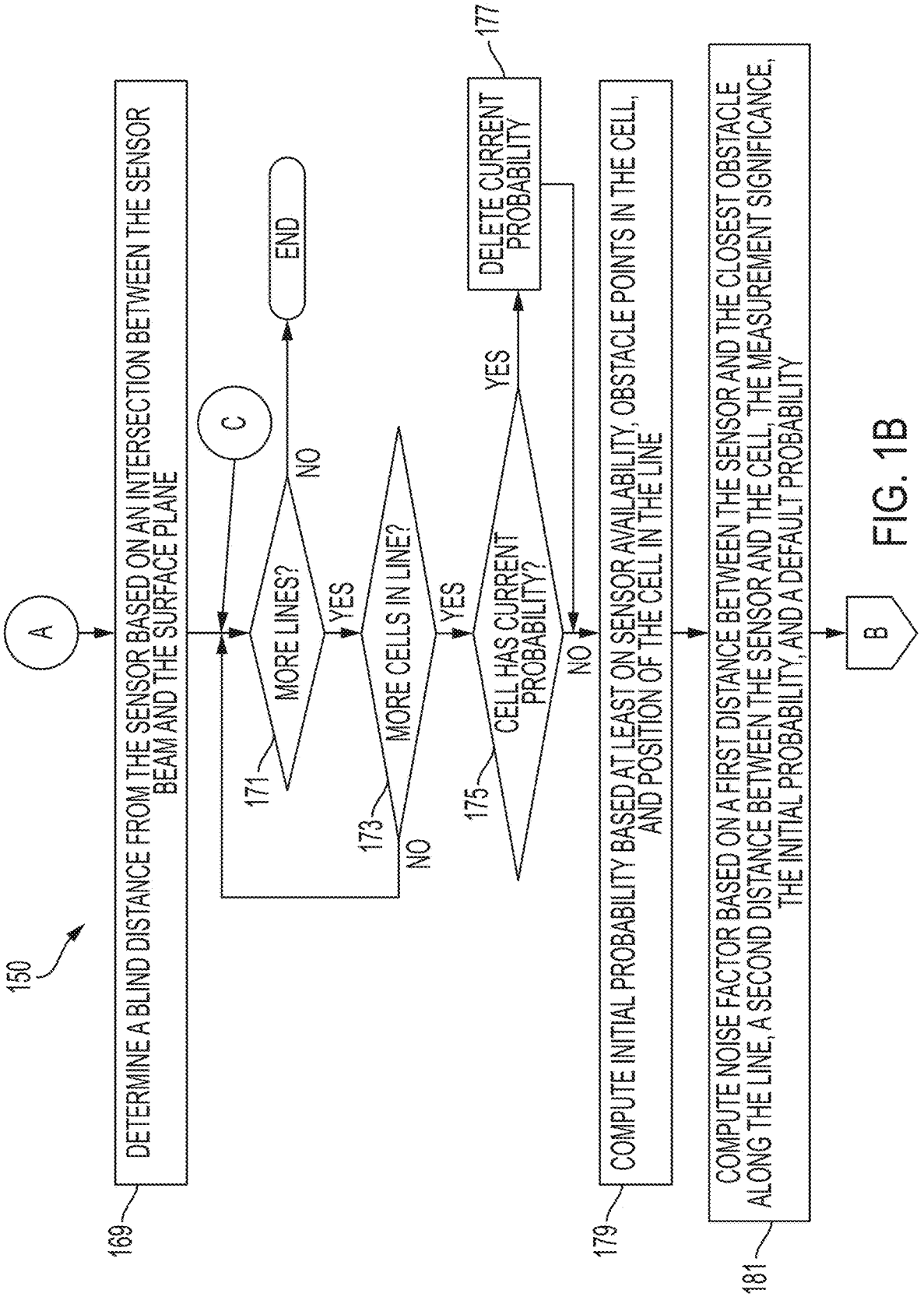

150
A
169
DETERMINE A BLIND DISTANCE FROM THE SENSOR BASED ON AN INTERSECTION BETWEEN THE SENSOR BEAM AND THE SURFACE PLANE
C
171
MORE LINES?
NO
END
YES
173
MORE CELLS IN LINE?
NO
YES
175
CELL HAS CURRENT PROBABILITY?
YES
177
DELETE CURRENT PROBABILITY
NO
179
COMPUTE INITIAL PROBABILITY BASED AT LEAST ON SENSOR AVAILABILITY, OBSTACLE POINTS IN THE CELL, AND POSITION OF THE CELL IN THE LINE
181
COMPUTE NOISE FACTOR BASED ON A FIRST DISTANCE BETWEEN THE SENSOR AND THE CLOSEST OBSTACLE ALONG THE LINE, A SECOND DISTANCE BETWEEN THE SENSOR AND THE CELL, THE MEASUREMENT SIGNIFICANCE, THE INITIAL PROBABILITY, AND A DEFAULT PROBABILITY
B

FIG. 1B

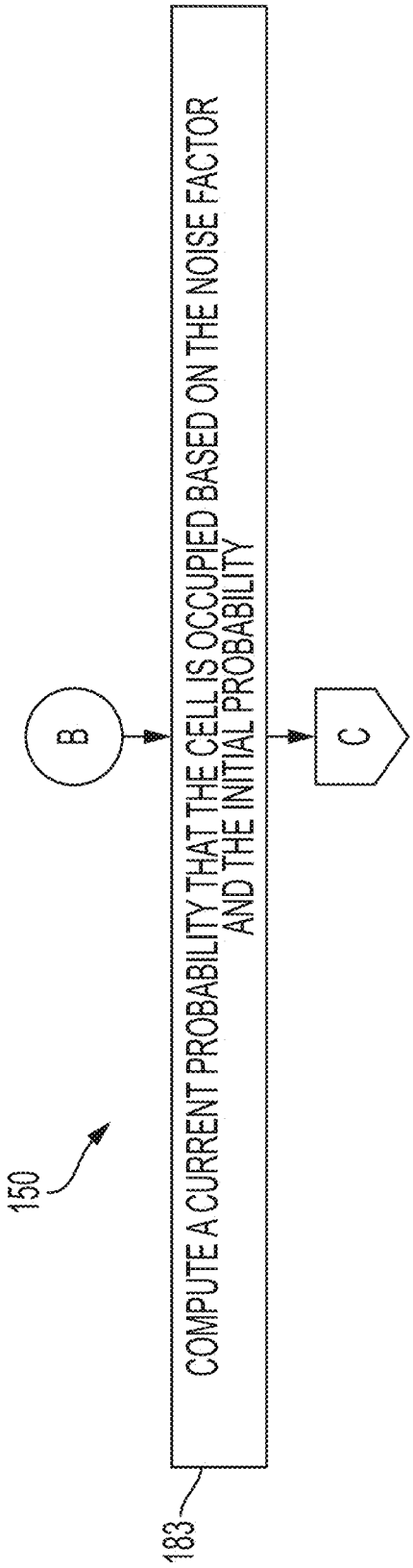

B
COMPUTE A CURRENT PROBABILITY THAT THE CELL IS OCCUPIED BASED ON THE NOISE FACTOR AND THE INITIAL PROBABILITY
C
150
183

FIG. 1C

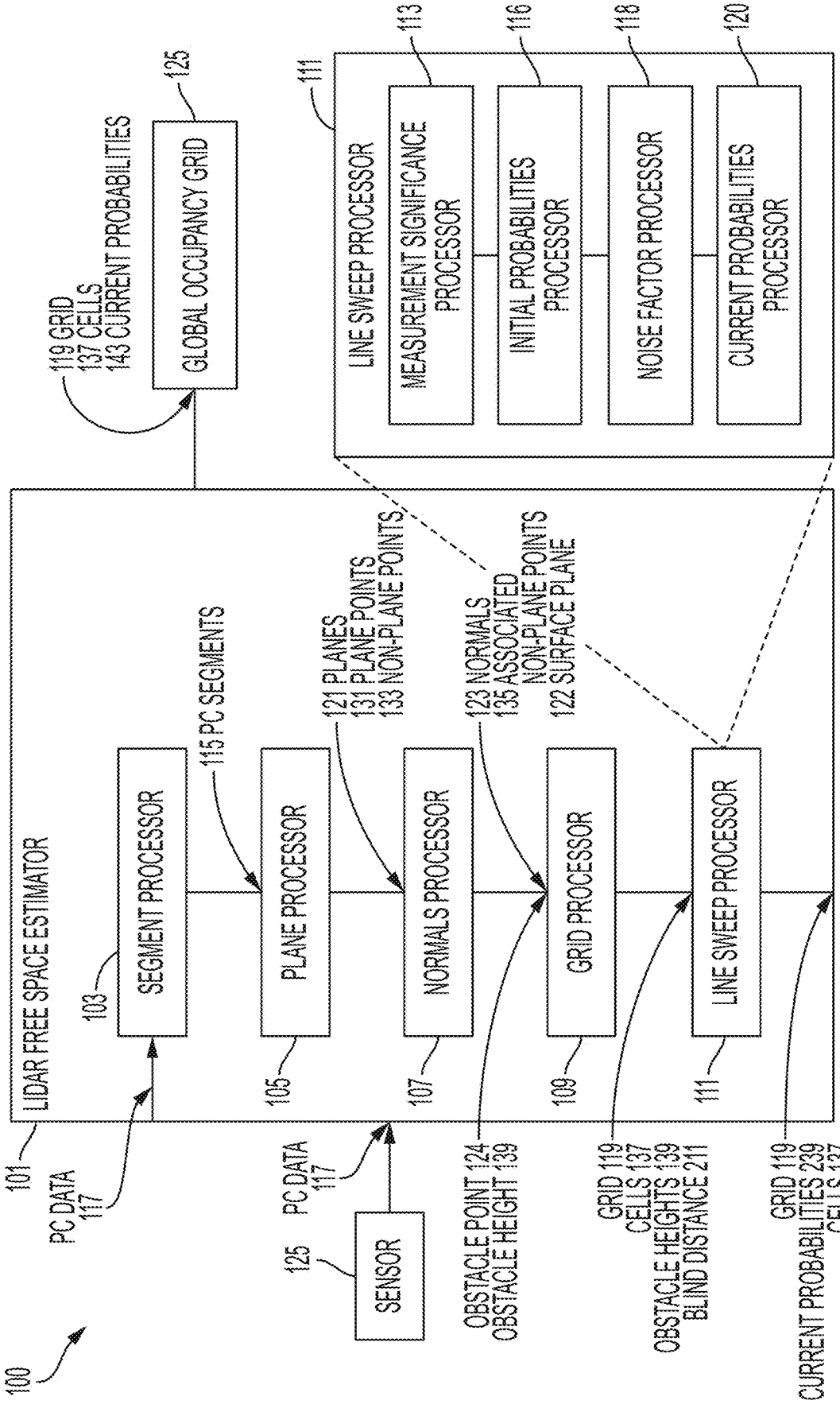

100
101
LIDAR FREE SPACE ESTIMATOR
PC DATA 117
SEGMENT PROCESSOR
103
115 PC SEGMENTS
PLANE PROCESSOR
105
121 PLANES
131 PLANE POINTS
133 NON-PLANE POINTS
NORMALS PROCESSOR
107
123 NORMALS
135 ASSOCIATED NON-PLANE POINTS
122 SURFACE PLANE
GRID PROCESSOR
109
LINE SWEEP PROCESSOR
111
125 SENSOR
PC DATA 117
OBSTACLE POINT 124
OBSTACLE HEIGHT 139
GRID 119
CELLS 137
OBSTACLE HEIGHTS 139
BLIND DISTANCE 211
GRID 119
CURRENT PROBABILITIES 239
CELLS 137
PERIMETER 138
119 GRID
137 CELLS
143 CURRENT PROBABILITIES
125
GLOBAL OCCUPANCY GRID
111
LINE SWEEP PROCESSOR
113
MEASUREMENT SIGNIFICANCE PROCESSOR
116
INITIAL PROBABILITIES PROCESSOR
118
NOISE FACTOR PROCESSOR
120
CURRENT PROBABILITIES PROCESSOR

FIG. 2

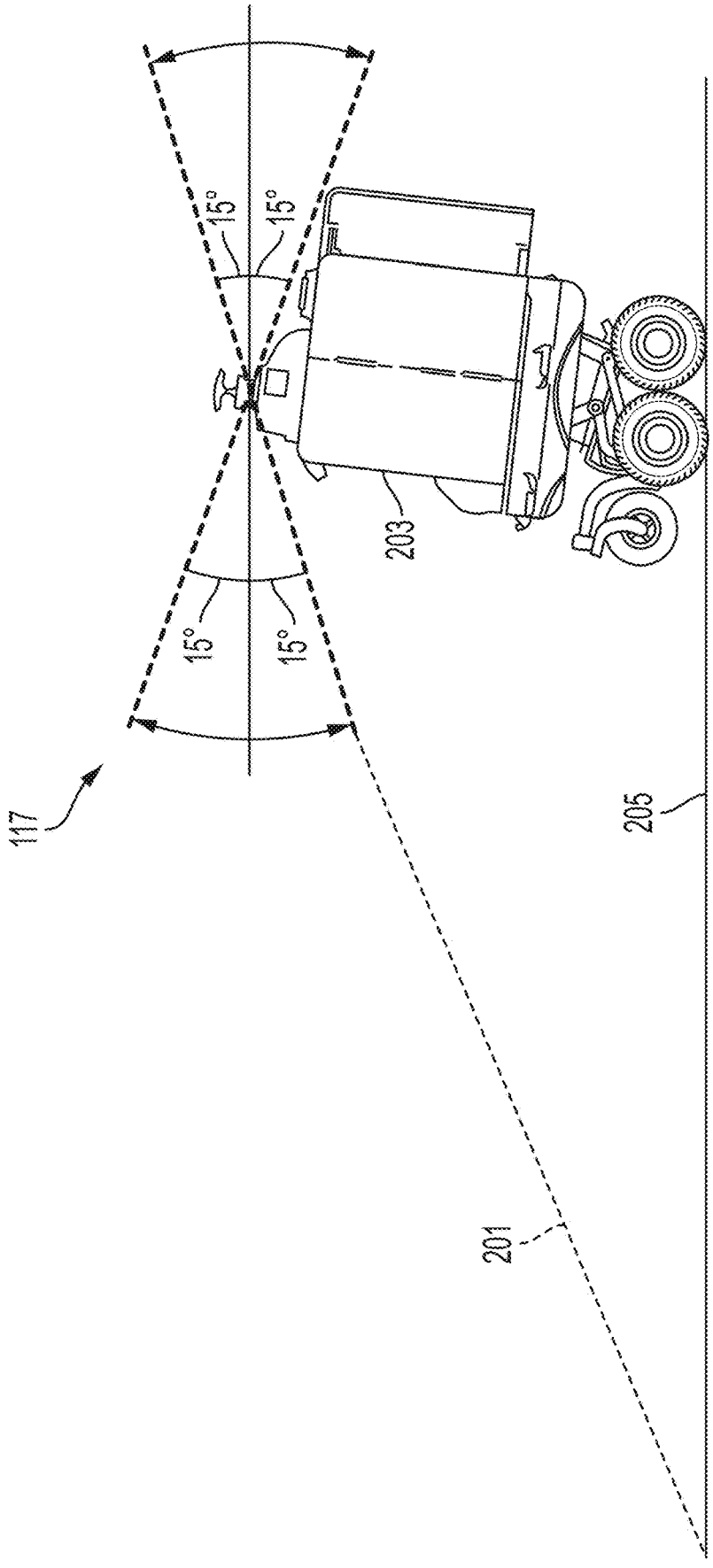


FIG. 3

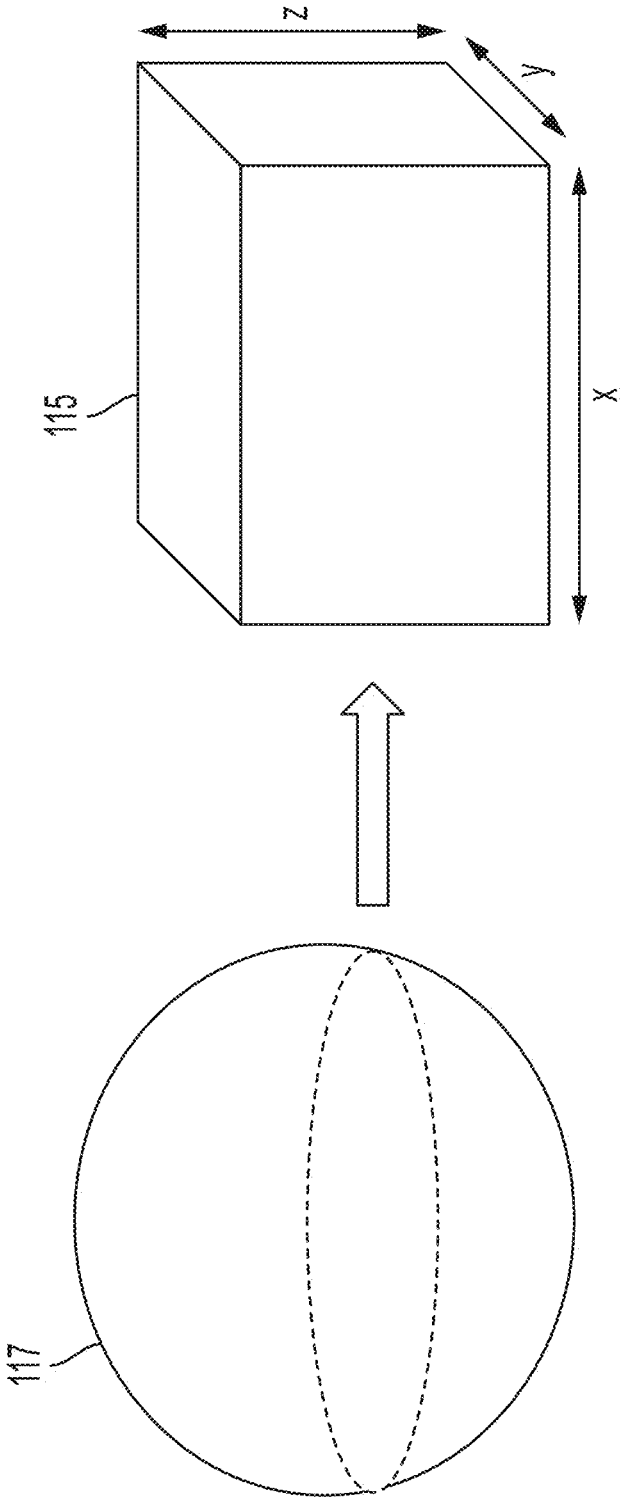


FIG. 4

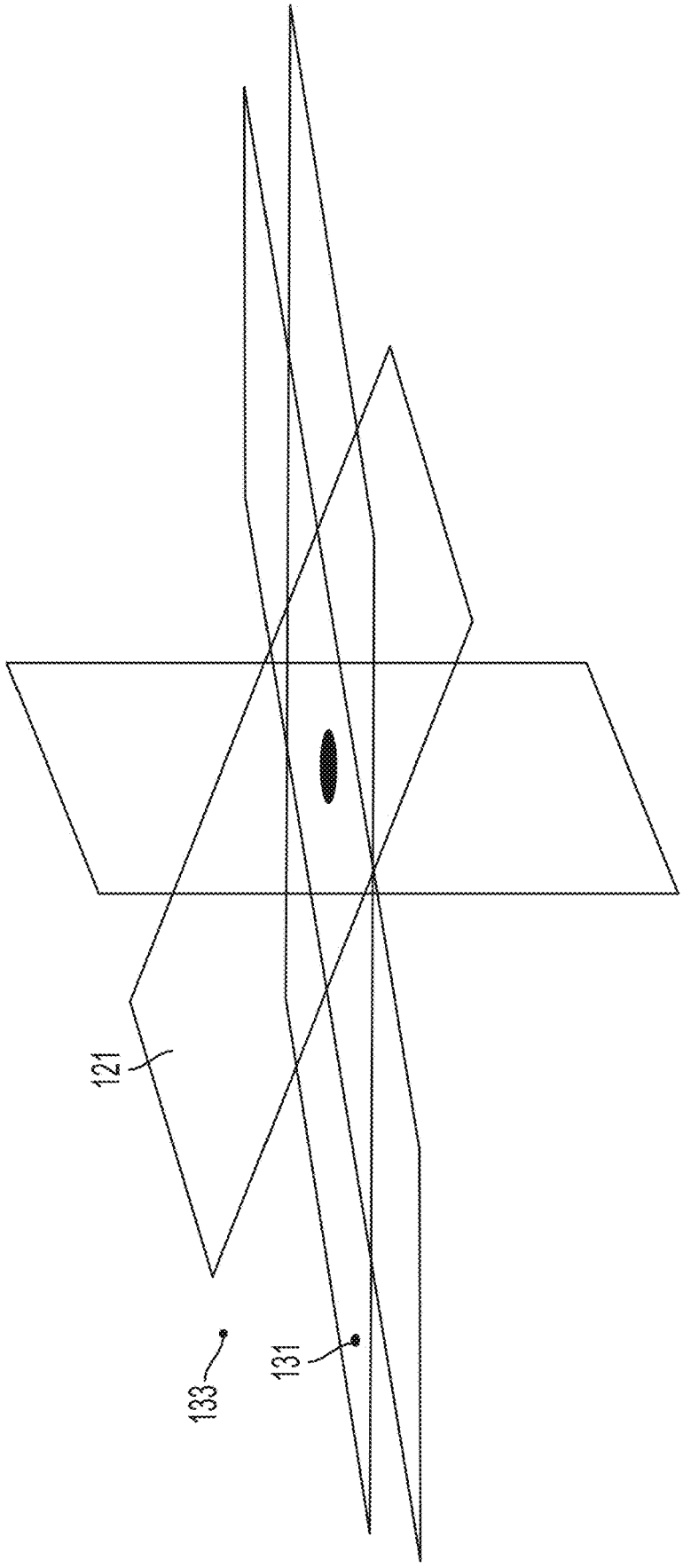


FIG. 5

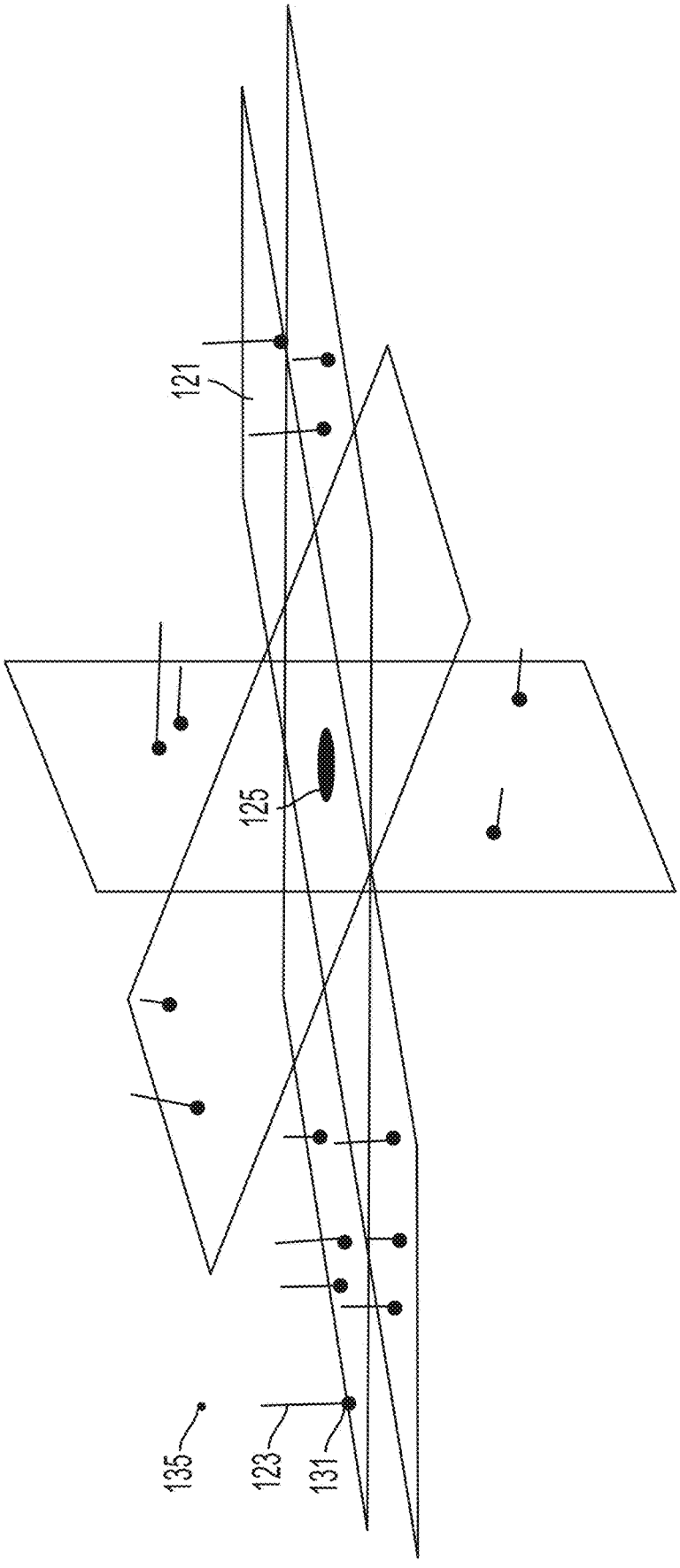


FIG. 6

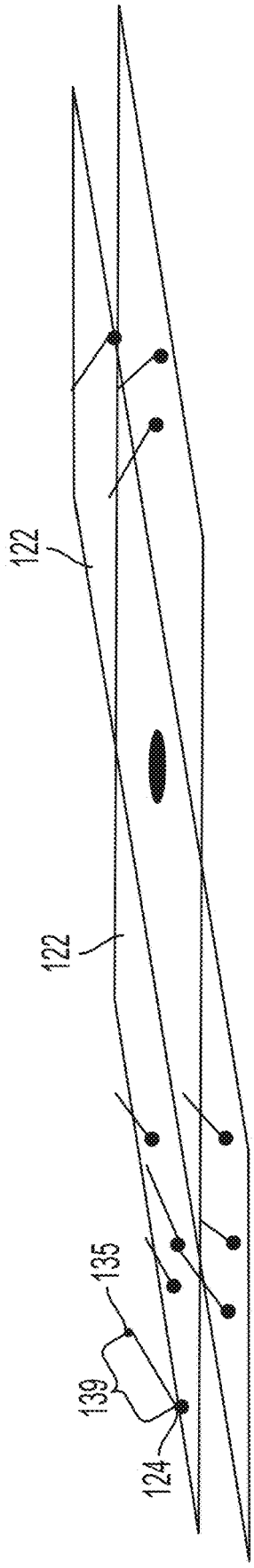


FIG. 7

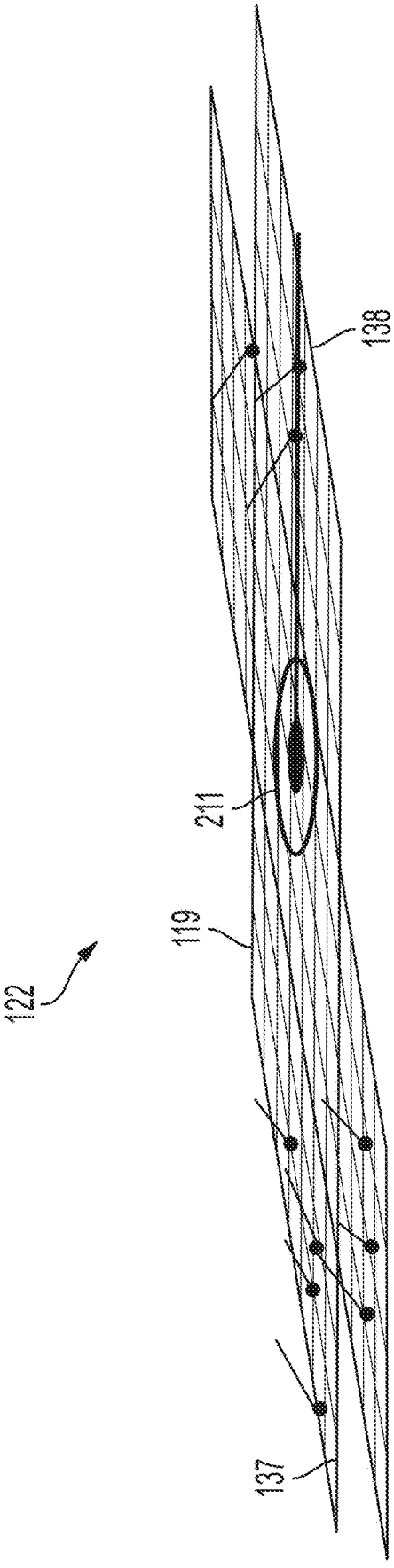


FIG. 8

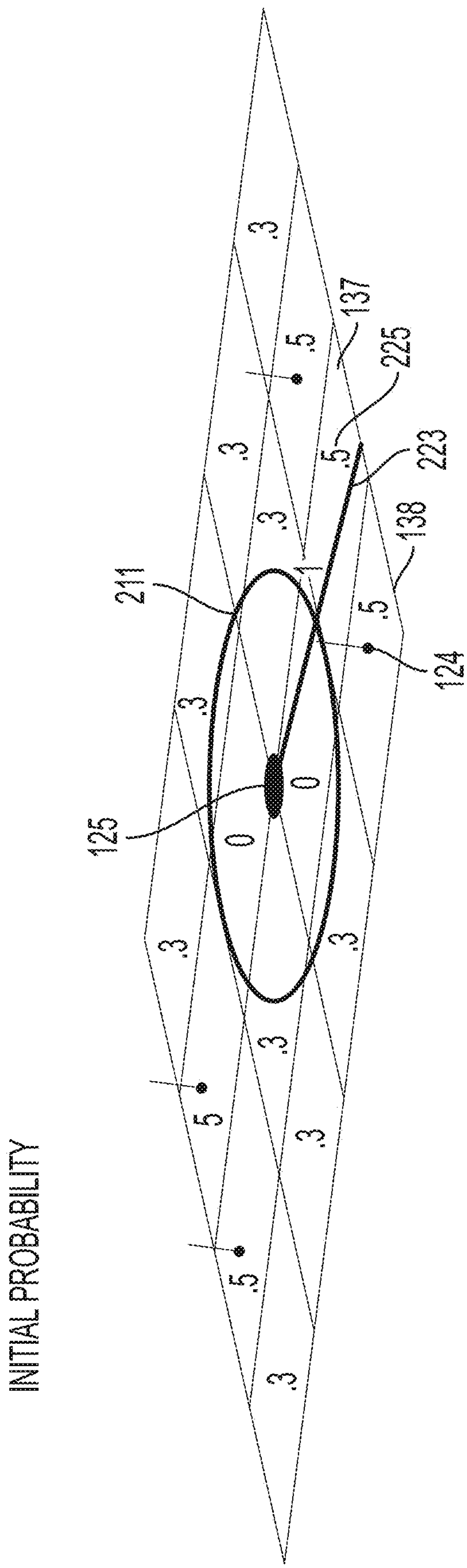


FIG. 9

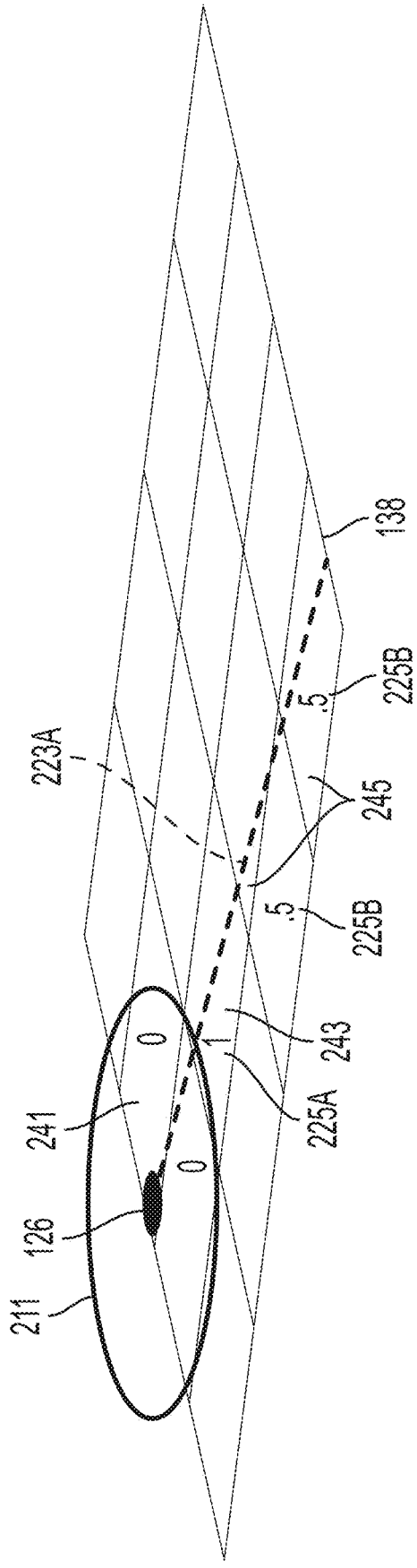


FIG. 9A

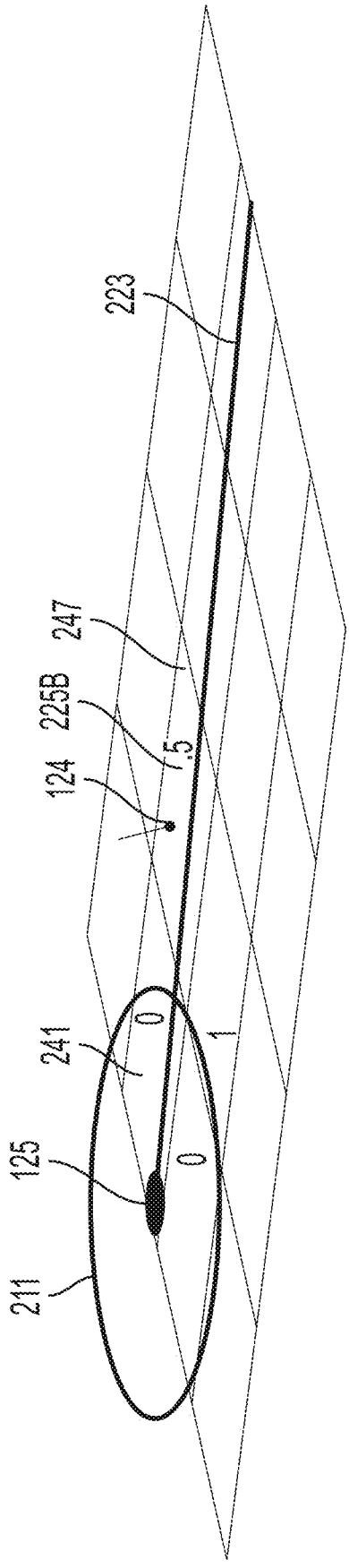


FIG. 9B

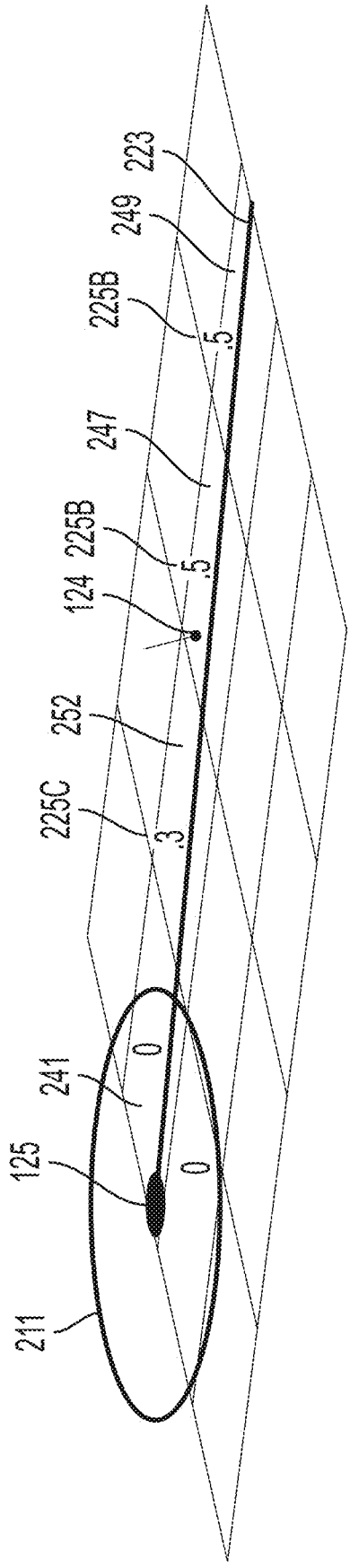


FIG. 9C

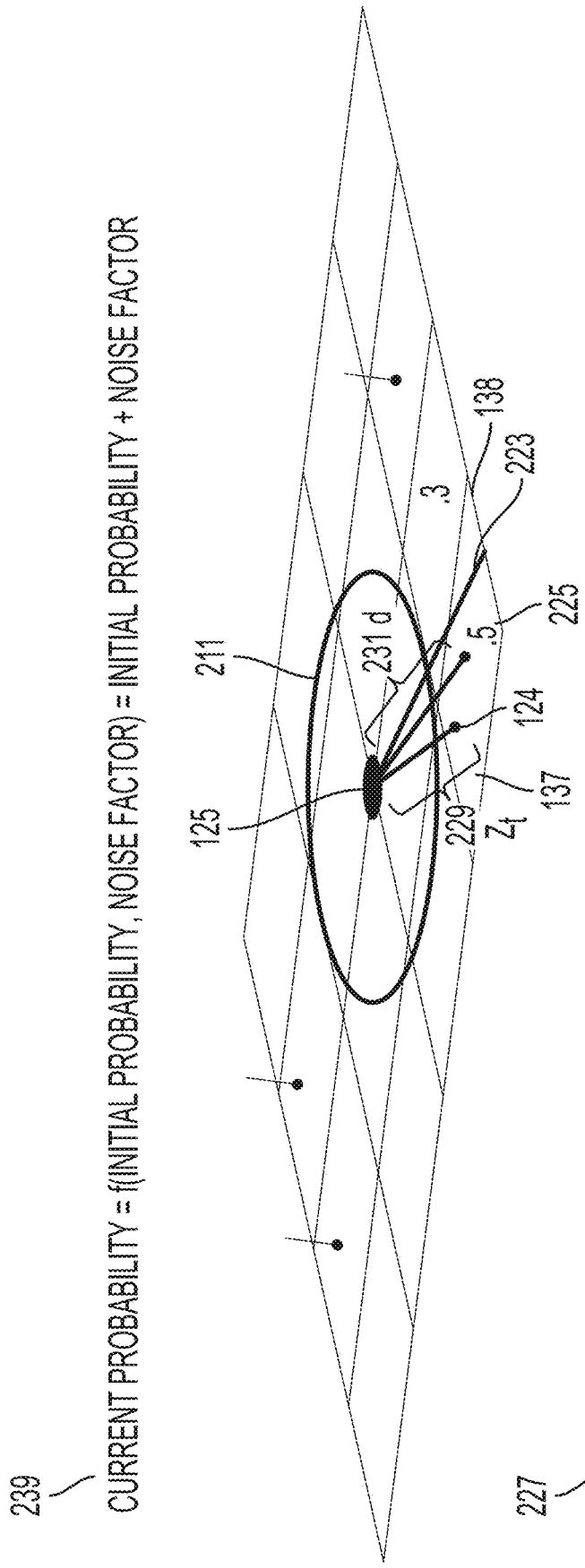

239
CURRENT PROBABILITY = f(INITIAL PROBABILITY, NOISE FACTOR) = INITIAL PROBABILITY + NOISE FACTOR
211
231 d
125
.3
138
223
.5
225
124
137
229
Z_t
227
NOISE FACTOR = ((((measurement significance/(σ√(2π))) + .5-the initial probability of the cell)*exp(-.5*((d-Z_t)/σ)2)
σ = Z_t2*.001
Measurement significance = .09*28.2*∑the obstacle heights in the cell

FIG. 10

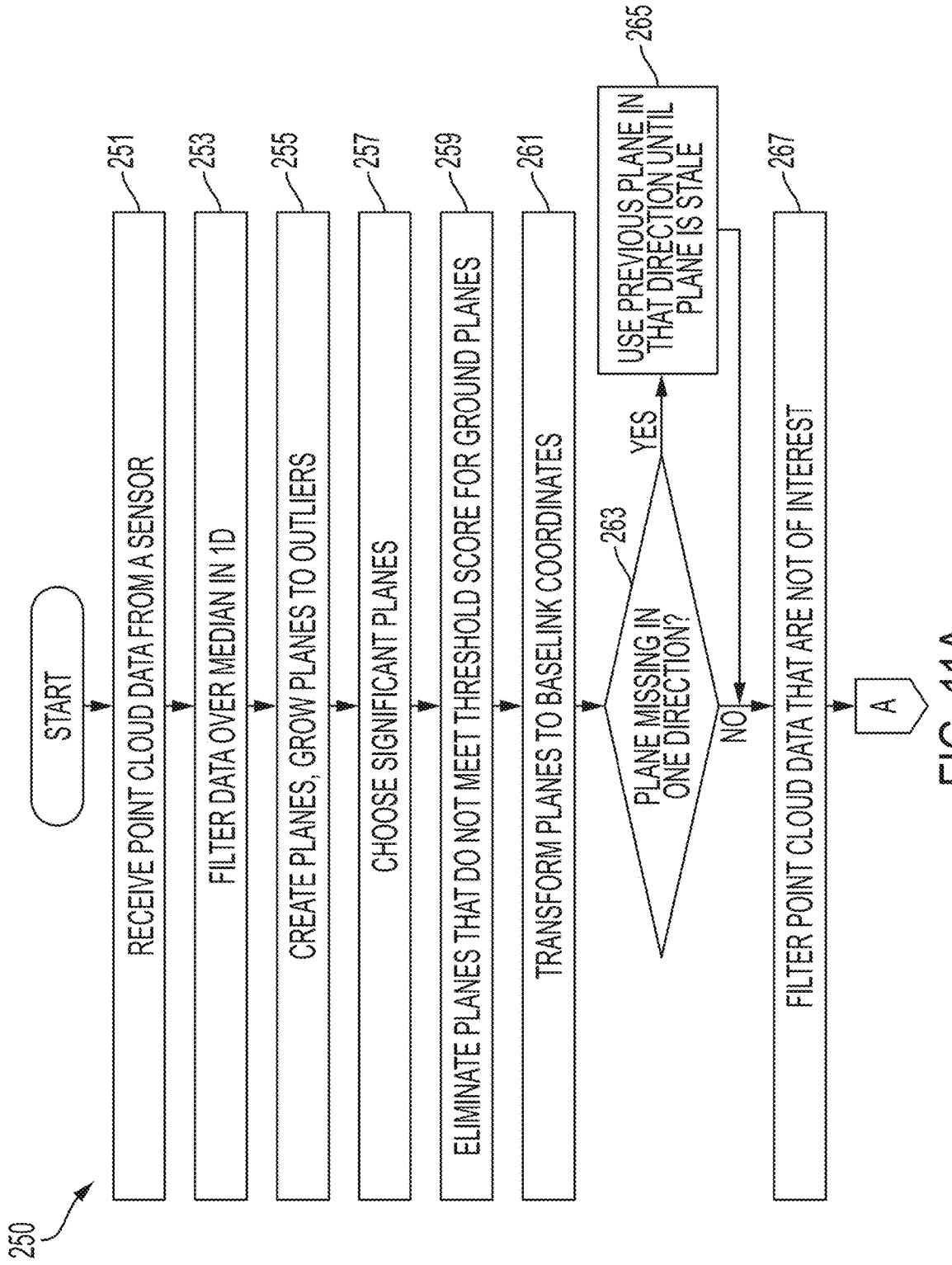

250
START
251
RECEIVE POINT CLOUD DATA FROM A SENSOR
253
FILTER DATA OVER MEDIAN IN 1D
255
CREATE PLANES, GROW PLANES TO OUTLIERS
257
CHOOSE SIGNIFICANT PLANES
259
ELIMINATE PLANES THAT DO NOT MEET THRESHOLD SCORE FOR GROUND PLANES
261
TRANSFORM PLANES TO BASELINK COORDINATES
263
PLANE MISSING IN ONE DIRECTION?
YES
265
USE PREVIOUS PLANE IN THAT DIRECTION UNTIL PLANE IS STALE
NO
267
FILTER POINT CLOUD DATA THAT ARE NOT OF INTEREST
A

FIG. 11A

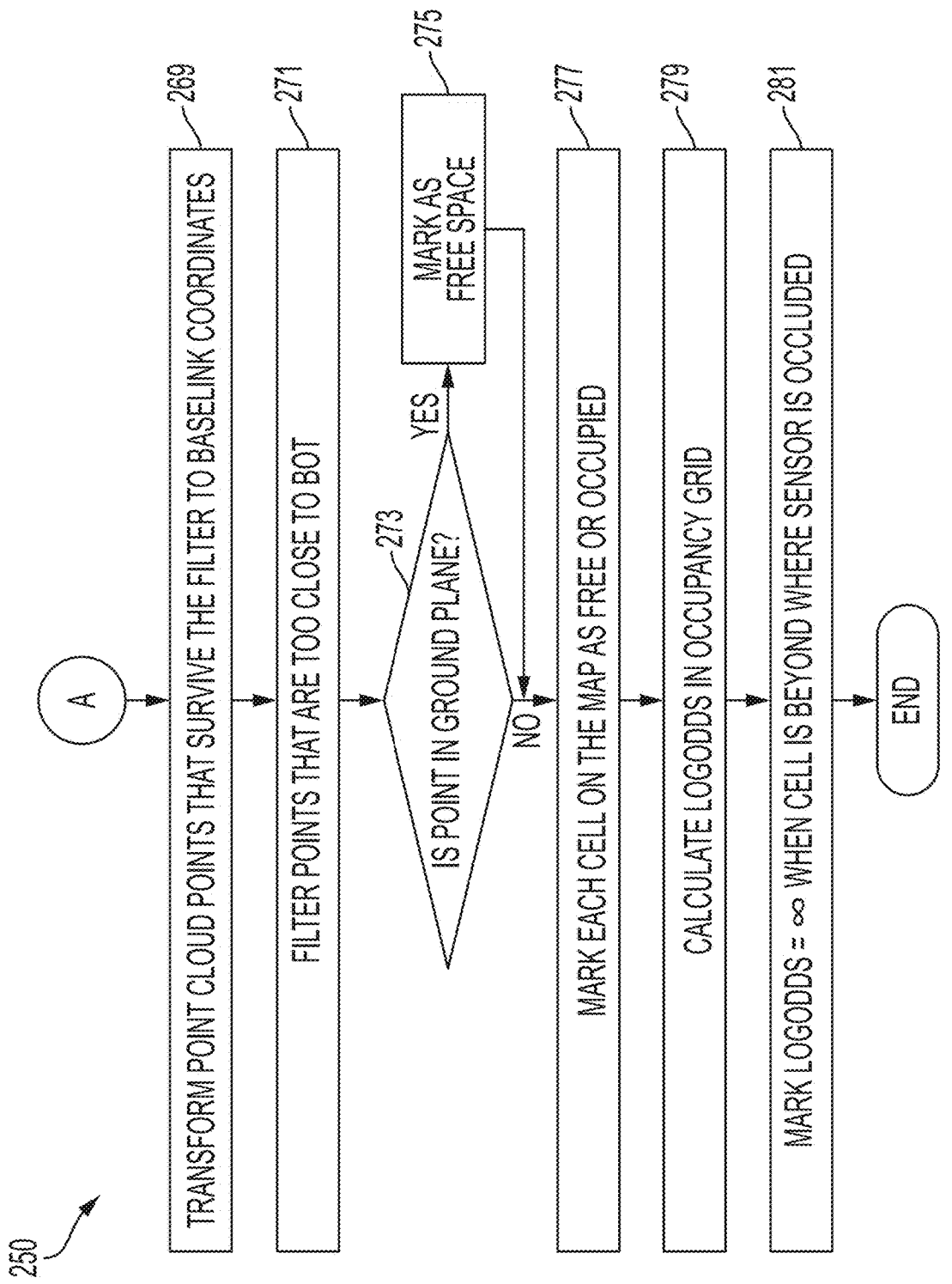

250
A
TRANSFORM POINT CLOUD POINTS THAT SURVIVE THE FILTER TO BASELINK COORDINATES
269
FILTER POINTS THAT ARE TOO CLOSE TO BOT
271
IS POINT IN GROUND PLANE?
273
YES
NO
MARK AS FREE SPACE
275
MARK EACH CELL ON THE MAP AS FREE OR OCCUPIED
277
CALCULATE LOGODDS IN OCCUPANCY GRID
279
MARK LOGODDS = ∞ WHEN CELL IS BEYOND WHERE SENSOR IS OCCLUDED
281
END

FIG. 11B

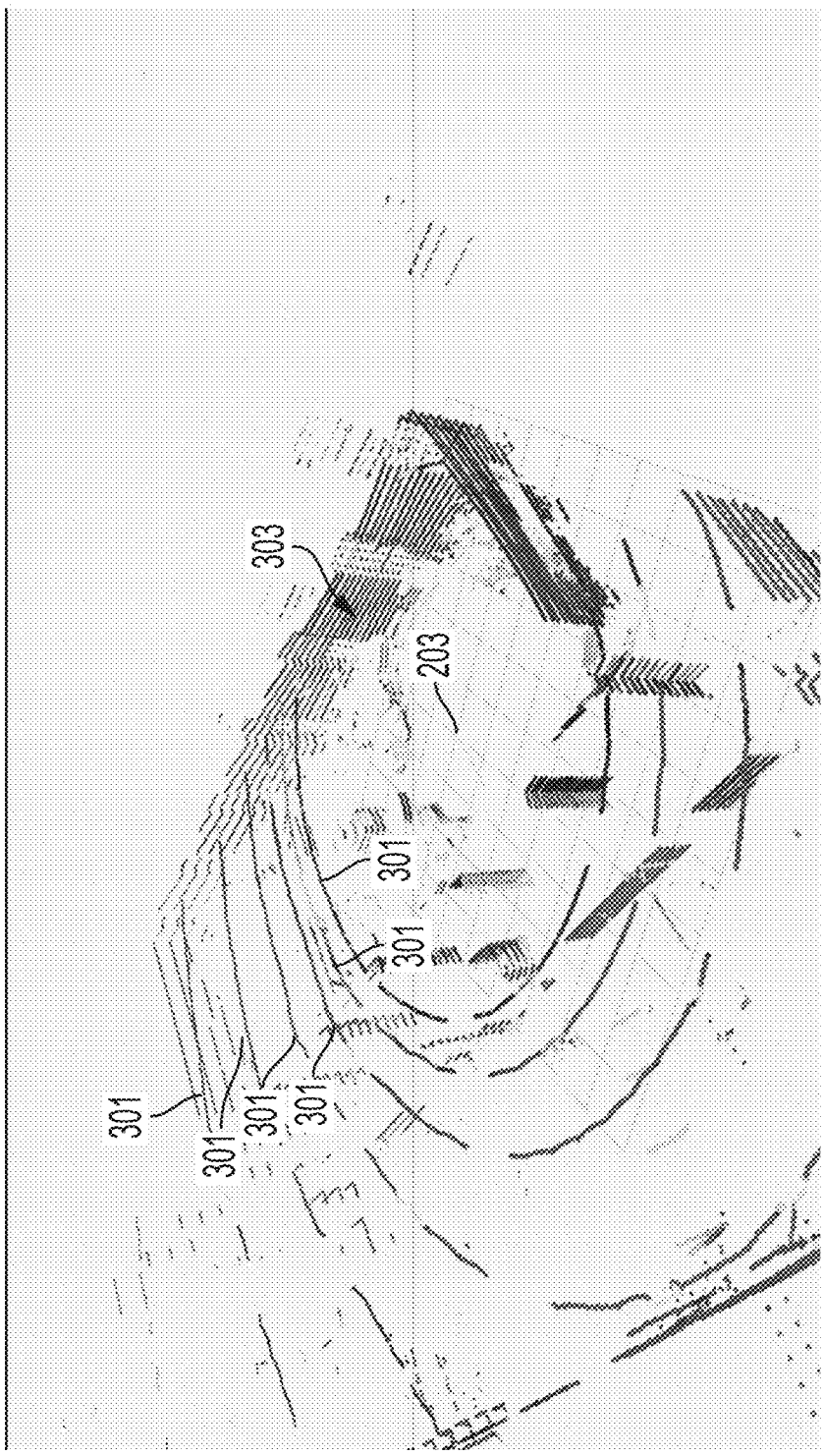


FIG. 12A

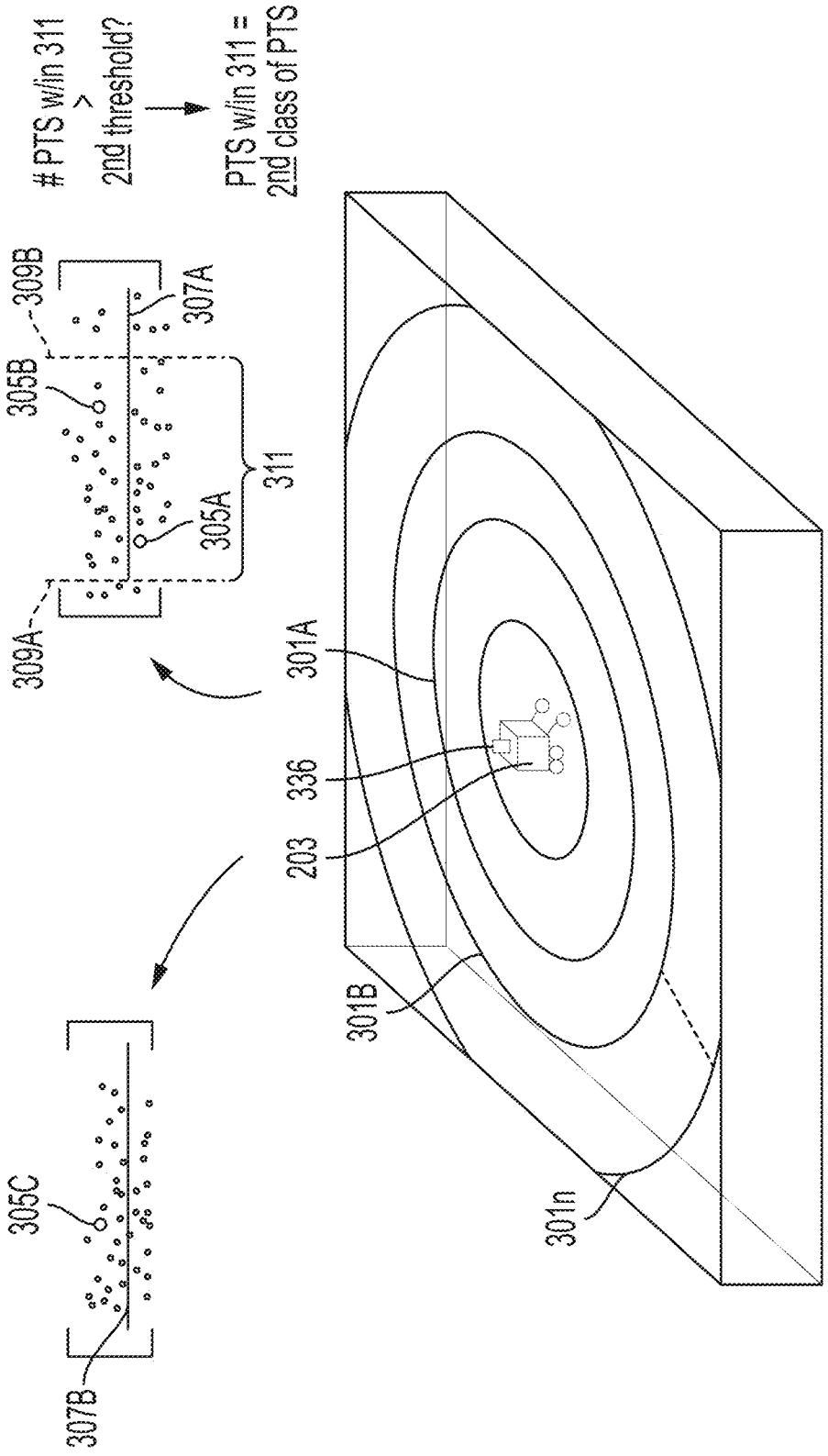

307B
305C
301n
301B
203
336
301A
309A
305A
311
305B
309B
307A
PTS w/in 311 > 2nd threshold?
PTS w/in 311 = 2nd class of PTS

FIG. 12B

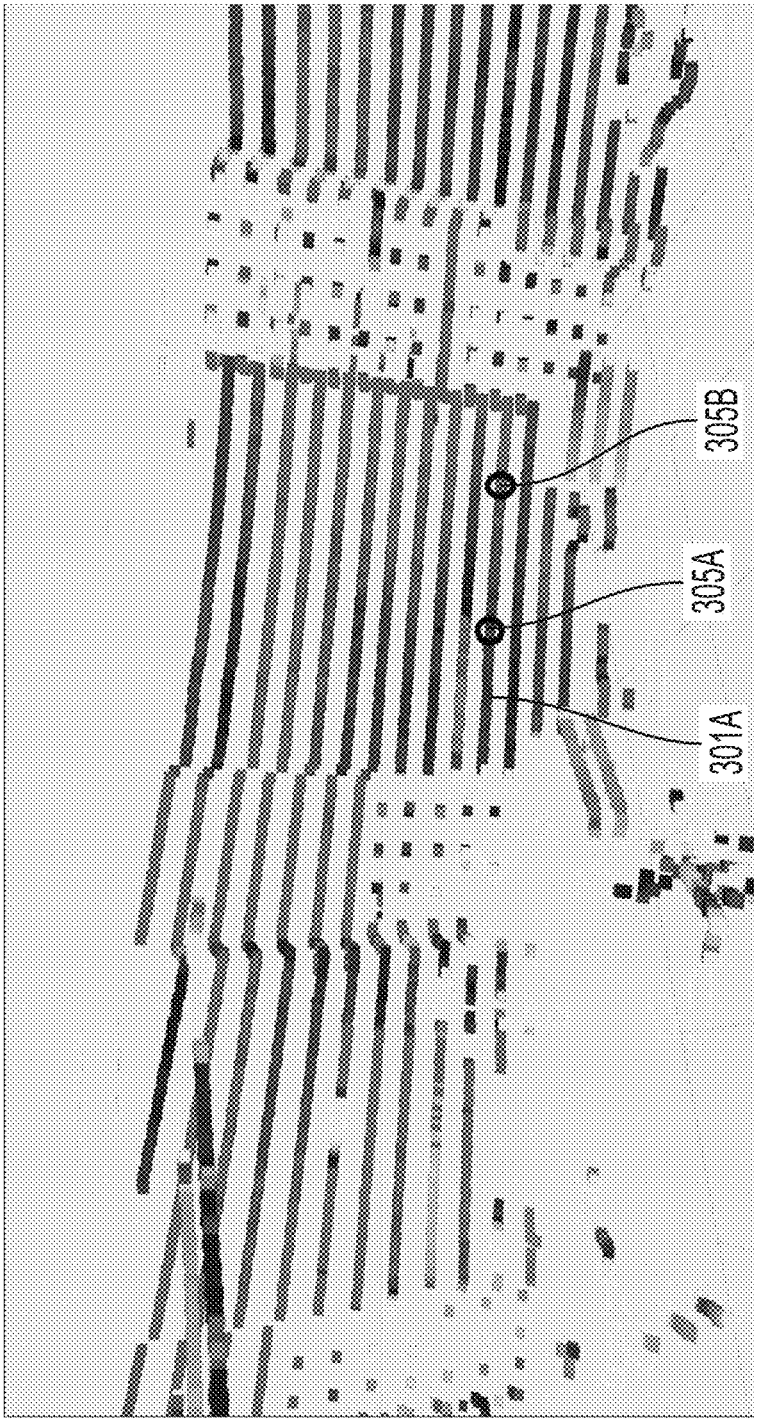


FIG. 13

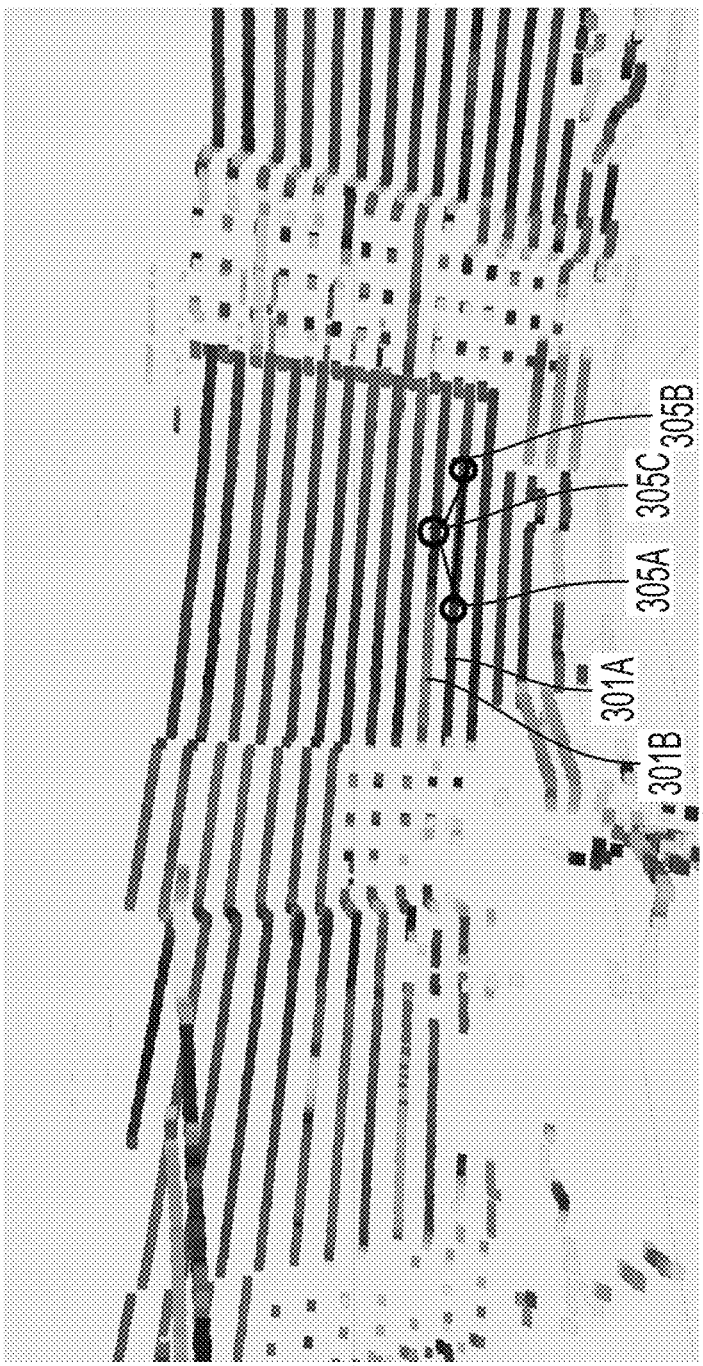


FIG. 14A

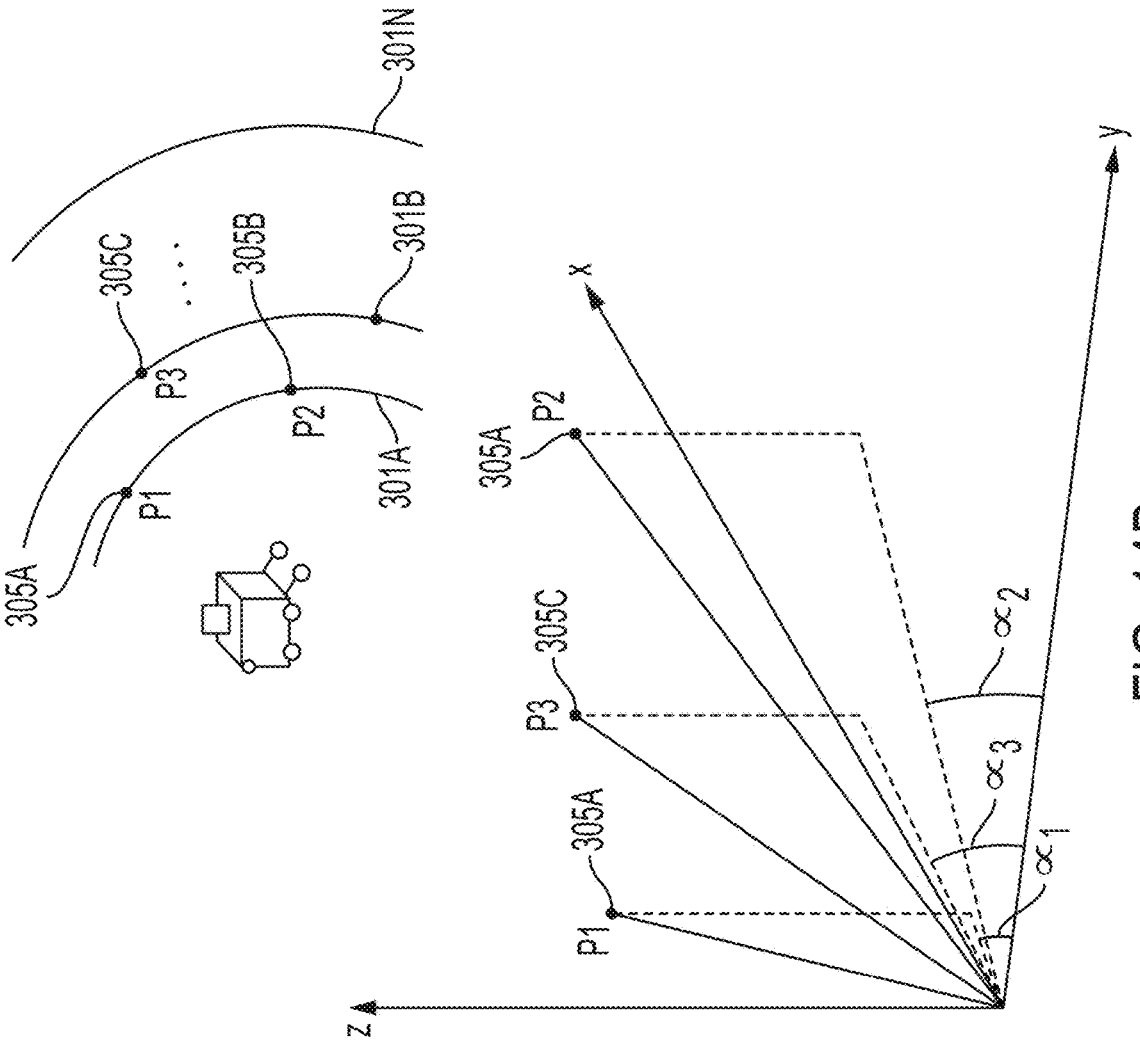

301N
305C
305B
301B
P3
P2
P1
301A
305A
x
y
z
305A
P2
305C
P3
305A
P1
α_2
α_3
α_1

FIG. 14B

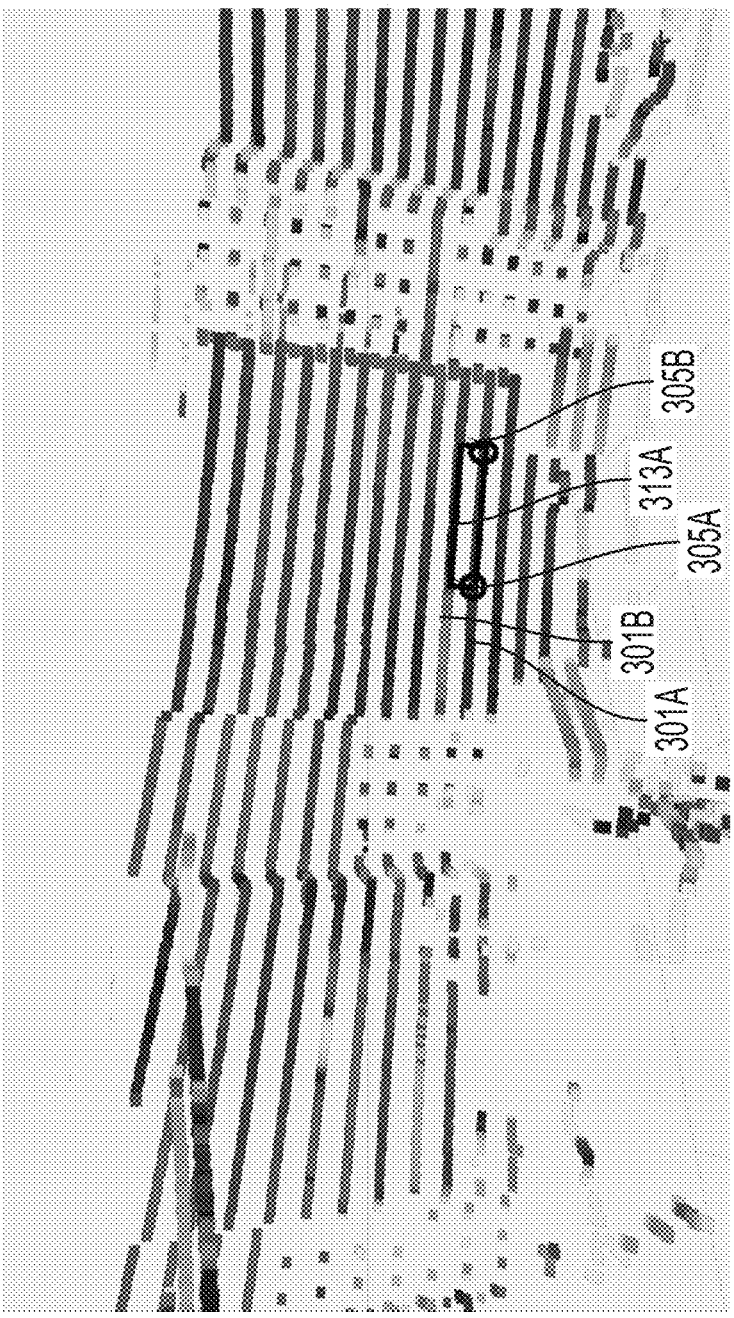


FIG. 15

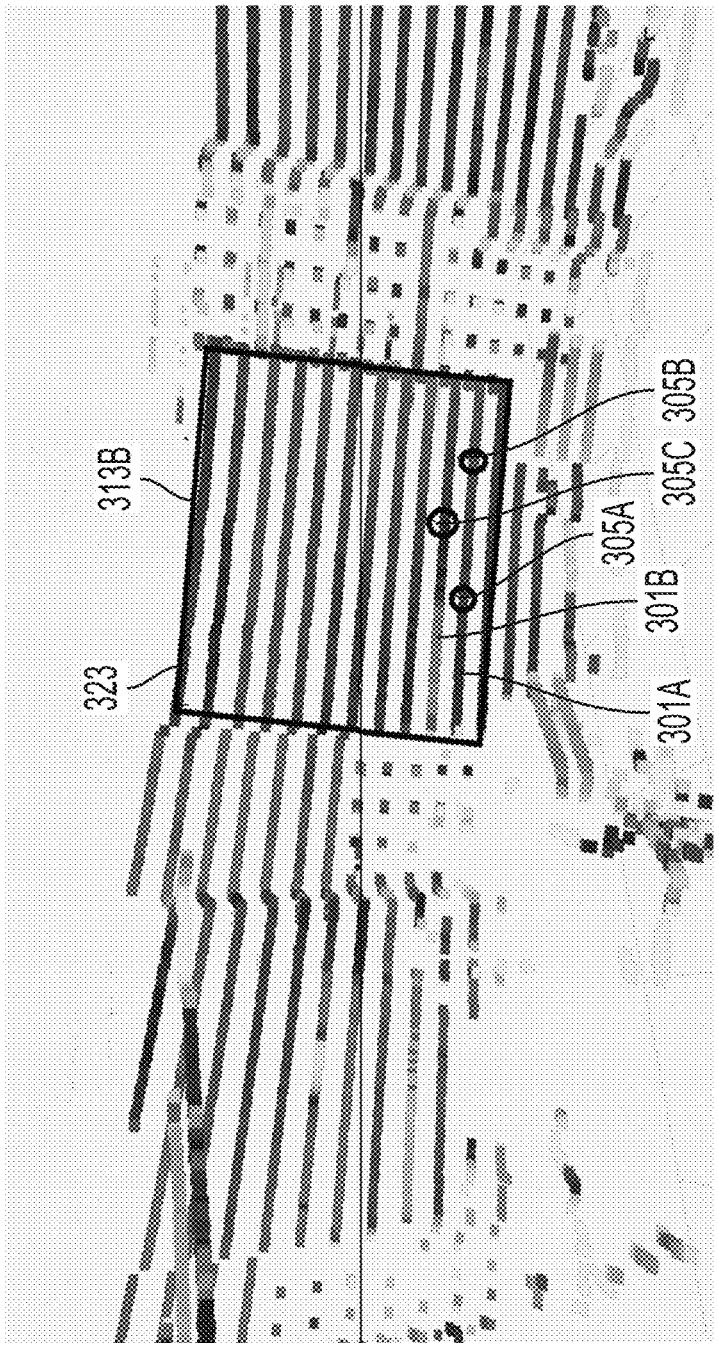


FIG. 16

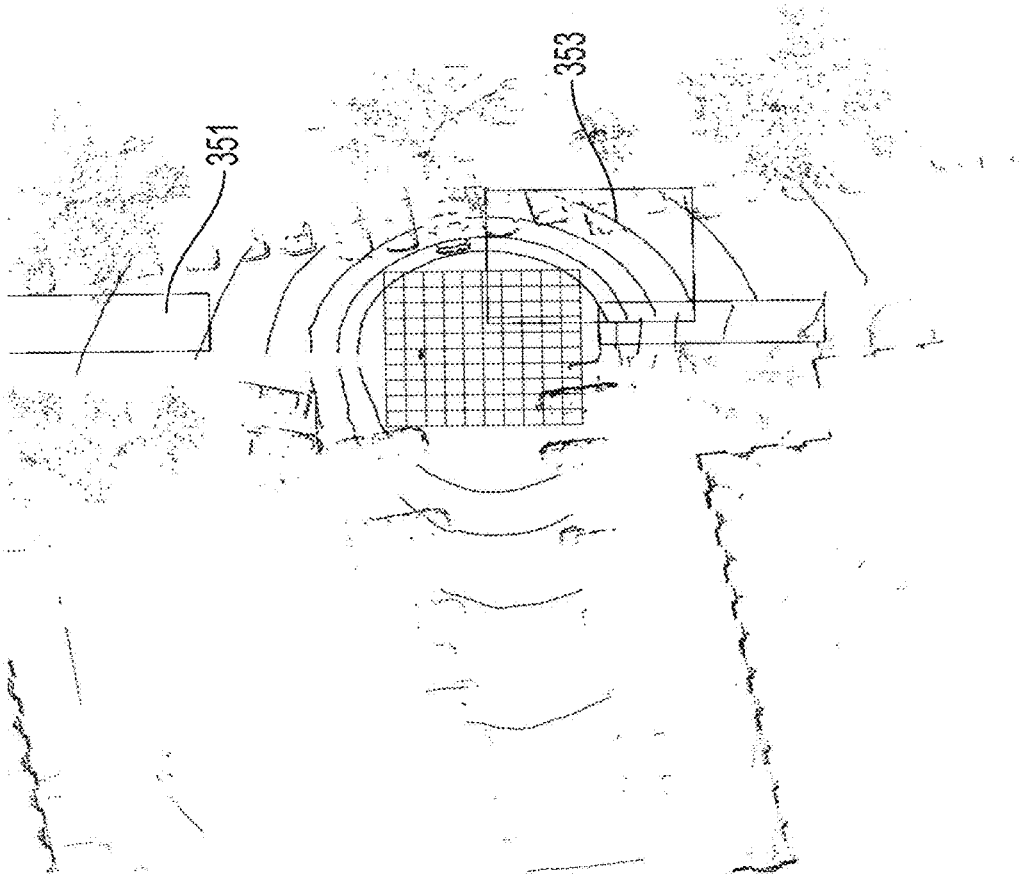


FIG. 17

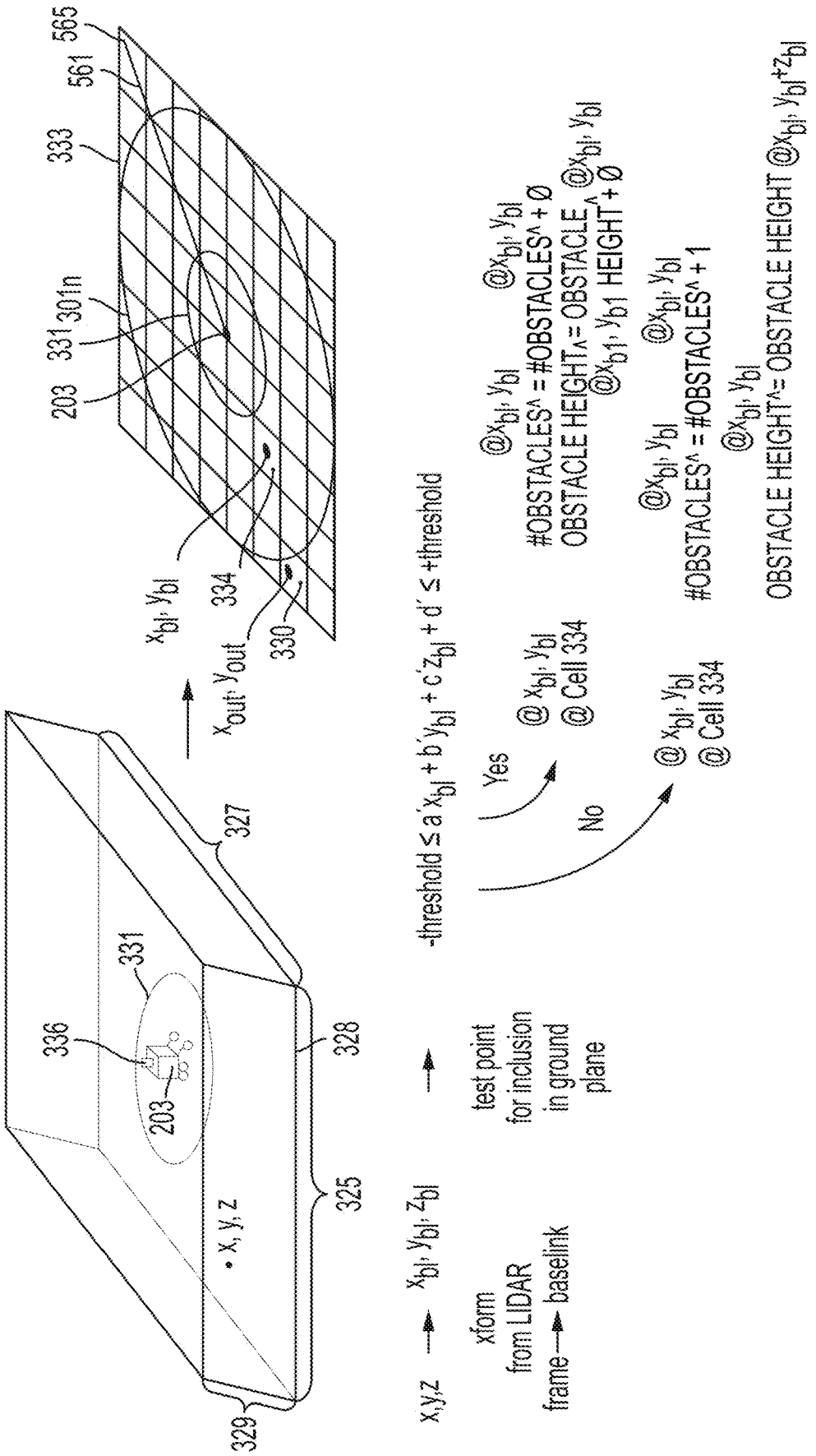

565
561
333
331
301n
203
334
330
327
328
325
329
336
331
203
• x, y, z
x,y,z
xform from LIDAR frame
baselink
x_{bl}, y_{bl}, z_{bl}
test point for inclusion in ground plane
$-\text{threshold} \le a' x_{bl} + b' y_{bl} + c' z_{bl} + d' \le +\text{threshold}$
Yes
No
@ x_{bl}, y_{bl}
@ Cell 334
@ x_{bl}, y_{bl}
@ Cell 334
x_{out}, y_{out}
x_{bl}, y_{bl}
@x_{bl}, y_{bl}
#OBSTACLES^ = #OBSTACLES^ + Ø
OBSTACLE HEIGHT^= OBSTACLE_^ @x_{bl}, y_{bl}
@x_{b1}, y_{b1} HEIGHT^ + Ø
@x_{bl}, y_{bl}
#OBSTACLES^ = #OBSTACLES^ + 1
@x_{bl}, y_{bl}
OBSTACLE HEIGHT^= OBSTACLE HEIGHT @x_{bl}, y_{bl}+z_{bl}

FIG. 18

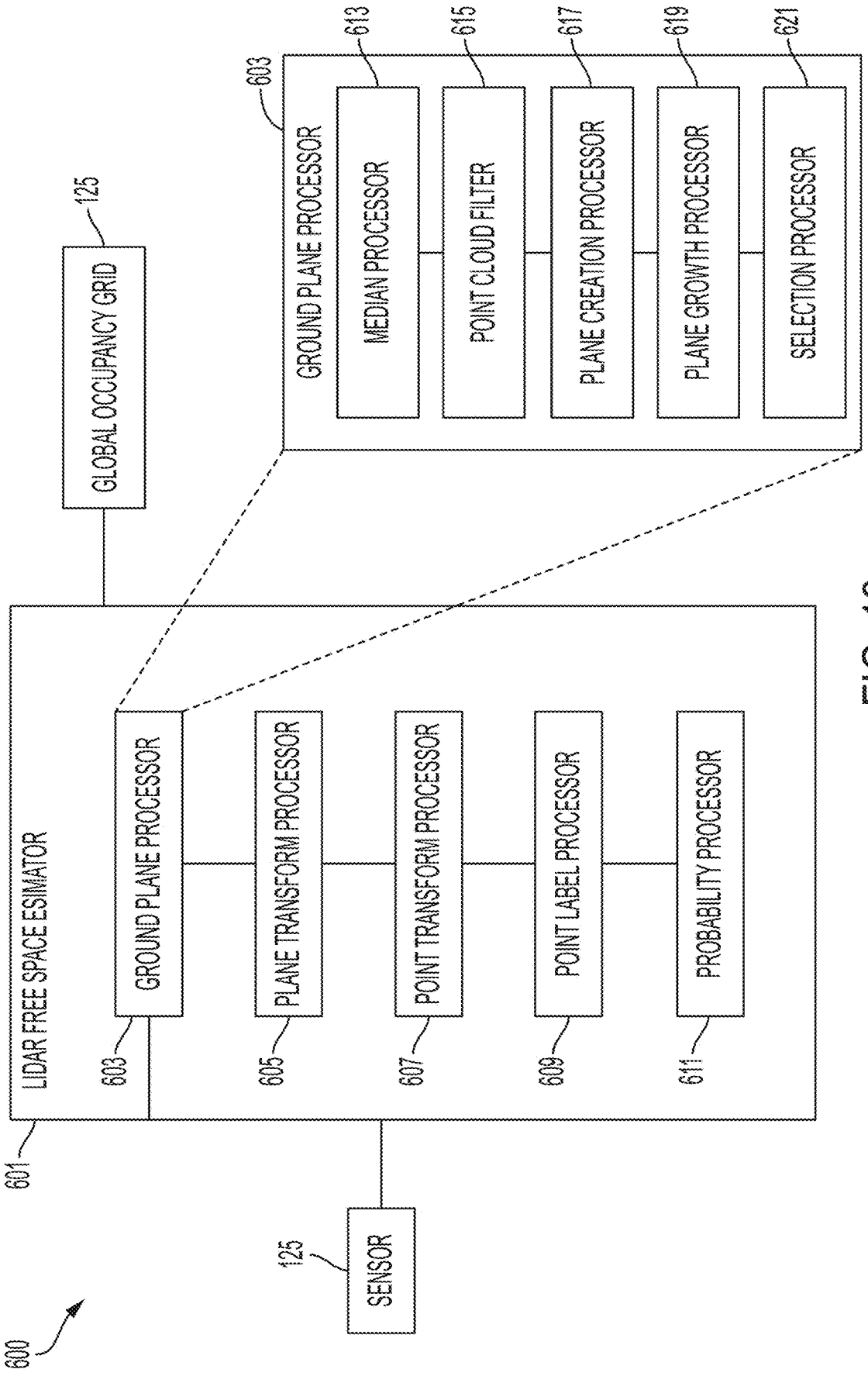

600
601
LIDAR FREE SPACE ESIMATOR
603
GROUND PLANE PROCESSOR
605
PLANE TRANSFORM PROCESSOR
607
POINT TRANSFORM PROCESSOR
609
POINT LABEL PROCESSOR
611
PROBABILITY PROCESSOR
125
SENSOR
125
GLOBAL OCCUPANCY GRID
603
GROUND PLANE PROCESSOR
613
MEDIAN PROCESSOR
615
POINT CLOUD FILTER
617
PLANE CREATION PROCESSOR
619
PLANE GROWTH PROCESSOR
621
SELECTION PROCESSOR

FIG. 19

SYSTEM AND METHOD FOR FREE SPACE ESTIMATION

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This utility patent application claims the benefit of U.S. Provisional Patent Application Ser. No. 62/879,391 filed Jul. 26, 2019, entitled System and Method for Free Space Estimation (Attorney Docket #AA027), which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] Vehicles travel on surfaces that are determined by their human operators to include free, unobstructed space. Humans use a complicated set of criteria to determine whether or not to traverse a path in or on a vehicle. Considerations can include how high an obstacle is, how much of the obstacle can be seen by the human, areas around the vehicle that the human cannot visualize, and how accurate the human's visioning system is at detecting obstructions given any number of factors such as, but not limited to, ambient lighting, weather, and windshield issues.

[0003] Navigating obstructions in an autonomous vehicle can require evaluating electronically some of the same complicated criteria routinely encountered by a human vehicle operator. Unobstructed (free) space must be quickly determined from available sensor data for the autonomous vehicle to continuously proceed along a navigation path. Previously, free space has been estimated from stereo camera data, from a sequence of images in a video acquired by a camera system, and from millimeter wave radar data, among other ways.

[0004] What is needed is an efficient system that computes, from sensor data, the probability that the path is obstructed. The sensor collecting the sensor data can be mounted upon the autonomous vehicle, for example. What is needed is a system that takes into account the realities of road travel and sensor limitations.

SUMMARY

[0005] The system and method of the present teachings for assigning free space probabilities in point cloud data associated with an autonomous vehicle traveling on a surface includes taking into account sensor noise, sensor availability, obstacle heights, and distance of obstacles from the sensor. The method of the present teachings can include, but is not limited to including, receiving the point cloud data from a sensor. The sensor can include a sensor beam, and the sensor beam can project at least from the sensor to the surface. In some configurations, the sensor can scan the area surrounding the autonomous vehicle, collecting data in a cone from the surface spanning a pre-selected angle. The method can include segmenting the point cloud data into segments of a first pre-selected size, and locating planes, plane points in the planes, and non-plane points associated with at least one of the plane points within the point cloud data. The method can include determining normals to the plane points and determining the non-plane points associated with the plane points. The method can include choosing at least one of the planes as a surface plane according to pre-selected criteria based at least on the normals and the location of the sensor, classifying each of the plane points as an obstacle point based at least on the associated non-plane points, and determining obstacle height associated with the obstacle points based at least on the non-plane points. The method can include creating a grid from the surface planes. The grid can include a pre-selected number of cells and a perimeter. The method can include computing a measurement significance for each of the cells based at least on the obstacle heights in the cells, and determining a blind distance from the sensor based at least on an intersection between the sensor beam and the surface plane. For each cell along each line between the blind distance and the perimeter, the method can include computing an initial probability of obstacles occupying the cell along the line. The initial probability can be based at least on availability of the sensor, the obstacle points in the cell, and a position of the cell along the line with respect to the sensor. For each cell along each line between the blind distance and the perimeter, the method can include computing a noise factor based at least on a first distance between the sensor and a closest of the obstacles along the line in the cell, a second distance between the sensor and the cell along the line, the measurement significance for the cell along the line, the initial probability for the cell along the line, and a default probability. For each cell along each line between the blind distance and the perimeter, the method can include computing a current probability of the obstacles occupying the cell along the line. The current probability can be based at least on the initial probability for the cell and the noise factor for the cell.

[0006] The first pre-selected size can optionally include a shape about the size of 40 m×40 m×2 m. The pre-selected criteria can optionally include choosing the surface plane when the normals of the at least one plane do not face the sensor. The pre-selected number of cells can optionally include 400×400. Computing the initial probability can optionally include (a) if the sensor is unavailable and the cell is a near cell, the near cell being near the blind distance, the initial probability of the cell can optionally equal 1.0, (b) if the sensor is unavailable and the cell is between the near cell and the perimeter, the initial probability of the cell can optionally equal 0.5, (c) if the sensor is available and there is at least one of the obstacle points in the cell, or if one of the cells along the line that was previously encountered included at least one of the obstacle points, the initial probability of the cell can optionally equal 0.5, and (d) if the sensor is available and there are none of the obstacle points in the cell and none of the cells previously encountered along the line included at least one of the obstacle points, the initial probability of the cell can optionally equal 0.3. The noise factor can optionally equal ((measurement significance/($\sigma\sqrt{2\pi}$))+0.5−the initial probability of the cell)*exp ($-0.5*((d-Z_t)/\sigma)^2$), where d=the second distance, Z_t=the first distance, and $\sigma=Z_t^2*0.001$. Computing the current probability can optionally equal the sum of the noise factor of the cell and the initial probability of the cell. The at least one plane can optionally include the non-plane points up to a first pre-selected distance from the at least one plane. The first pre-selected distance can optionally include 2 m.

[0007] The system of the present teachings for assigning free space probabilities from point cloud data can include, but is not limited to including, a sensor having a sensor beam. The sensor beam can project at least from the sensor to the surface. The system can include a segment processor receiving the point cloud data from the sensor. The segment processor segmenting the point cloud data into segments of

a first pre-selected size. The system can include a plane processor locating, within the point cloud data, planes, plane points in the planes, and non-plane points associated with at least one of the plane points. The system can include a normals processor determining normals to plane points and determining the non-plane points associated with the plane points. The normal processor can choose at least one of the planes as a surface plane according to pre-selected criteria based at least on the normals and the location of the sensor. The normals processor can classify each of the plane points as an obstacle point based at least on the associated non-plane points. The normals processor can determine obstacle height associated with the obstacle points based at least on the non-plane points. The system can include a grid processor creating a grid from the surface planes. The grid can include a pre-selected number of cells and a perimeter. The system can include a line sweep processor that can include a measurement significance processor. The measurement significance processor can compute a measurement significance for each of the cells based at least on the obstacle heights in the cells, The line sweep processor can include an initial probabilities processor determining a blind distance from the sensor based at least on an intersection between the sensor beam and the surface plane. For each cell along each line between the blind distance and the perimeter, the initial probabilities processor can include computing an initial probability of obstacles occupying the cell along the line. The initial probability can be based at least on the availability of the sensor, the obstacle points in the cell, and a position of the cell along the line with respect to the sensor. The line sweep processor can include a noise factor processor. For each cell along each line between the blind distance and the perimeter, the noise factor processor can compute a noise factor based at least on a first distance between the sensor and a closest of the obstacles along the line in the cell, a second distance between the sensor and the cell along the line, the measurement significance for the cell along the line, the initial probability for the cell along the line, and a default probability. The line sweep processor can include a current probabilities processor. For each cell along each line between the blind distance and the perimeter, the current probabilities processor can compute a current probability of the obstacle points occupying the cell along the line. The current probability can be based at least on the initial probability for the cell and the noise factor for the cell.

[0008] In some configurations, the method for assigning free space probabilities in sensor data for an autonomous vehicle can include, but is not limited to including, determining at least one surface plane in the sensor data, where the at least one surface plane can be associated with a surface where the autonomous vehicle is traveling. The method can include determining obstacles, if any, and heights of the obstacles, if any, in the sensor data associated with the at least one surface plane, and determining a blind distance from the autonomous vehicle based at least on a dimension of the autonomous vehicle. The method can include creating a grid on the at least one surface plane, where the grid can include a pre-selected number of cells and a perimeter. For each cell along each line on the at least one surface plane between the blind distance and the perimeter, the method can include computing an initial probability of the obstacles occupying the cell along the line. The initial probability can be based at least on availability of the sensor data, the obstacles in the cell, and a position of the cell along the line with respect to the autonomous vehicle. For each cell along each line between the blind distance and the perimeter, the method can include computing a noise factor based at least on a first distance between the autonomous vehicle and a closest of the obstacles along the line in the cell, a second distance between the autonomous vehicle and the cell along the line, the obstacle heights for the cell along the line, the initial probability for the cell along the line, and a default probability. For each cell along each line between the blind distance and the perimeter, the method can include computing a current probability of the obstacles occupying the cell along the line, the current probability being based at least on the initial probability for the cell and the noise factor for the cell.

[0009] The pre-selected number of cells can optionally include 400×400. Computing the initial probability can optionally include (a) if the sensor data are unavailable and the cell is a near cell, the near cell being near the blind distance, the initial probability of the cell can optionally equal 1.0, (b) if the sensor data are unavailable and the cell is between the near cell and the perimeter, the initial probability of the cell can optionally equal 0.5, (c) if the sensor data are available and there is at least one of the obstacles in the cell, or if one of the cells along the line that was previously encountered included at least one of the obstacles, the initial probability of the cell can optionally equal 0.5, and (d) if the sensor data are available and there are none of the obstacles in the cell and none of the cells previously encountered along the line included at least one of obstacles, the initial probability of the cell can optionally equal 0.3. Computing the noise factor can optionally equal ((0.09*28.2*Σ the obstacle heights in the cell/($\sigma\sqrt{2\pi}$))+0.5−the initial probability of the cell)*exp(−0.5*((d−Z_t)/σ)^2), wherein d=the second distance, Z_t=the first distance, and $\sigma=Z_t^2*0.001$. Computing the current probability can optionally equal the sum of the noise factor of the cell and the initial probability of the cell. The at least one surface plane can optionally include non-plane points up to a first pre-selected distance from the at least one surface plane. The first pre-selected distance can optionally include 2 m. The method can optionally include determining the obstacle heights based at least on the non-plane points.

[0010] In another configuration, free space estimation from LIDAR data, where the LIDAR data includes rings, can include, but is not limited to including, receiving LIDAR data, and filtering a pre-selected number of points in each ring. Filtering can include identifying a median value in each pre-selected number of points and retaining points that are within a pre-selected range from the median. Among the retained points can be discontinuities in which the Cartesian distance between the points is greater than a pre-selected value. Points that are between the discontinuities are labeled as good points if they have passed the median filter. Good points can be expected to have low sensor noise. Where the number of good points between discontinuities is greater than a pre-selected value, then those good points are retained. Discontinuities, or break points, can be found at the edges of features, so that when there is a sudden change in the distance between points, an edge could be found.

[0011] At this point, each of the filtered points can be associated with a LIDAR ring. The filtered point data can be divided sections of 64 points each. A random section is selected, and two points from the random section can be

chosen. The two points can be intelligently chosen based on how the points were labeled in the filtering process, i.e. good points and break points. For example, two points between the same two discontinuities can be chosen. A third point is chosen from an adjacent ring, and a plane is formed from the three points. The plane is evaluated with respect to certain pre-selected criteria according to an algorithm that can eliminate planes that are not significant, or are not interesting. All points on the adjacent LIDAR ring (the LIDAR ring corresponding to third point) that are within the azimuth range of first two points are evaluated as candidates to be included in the first growth stage. The plane equation is calculated with the updated set of points. Then the points are evaluated again to grow this plane along the points data structure axes that correspond to LIDAR rings and azimuth angles respectively. At each growth stage, the orientation and residual error of the plane are checked. Residual error is calculated as a plane is fitted to a set of points, and orientation is the angle check between the gravity vector and the normal vector of the plane. Growth on each side is done until the residual error for the new set of points along a direction, for example, growing towards an outward ring would check the residual error of the new set of points being added from that ring, is above the threshold set, or until the edge of the point cloud is reached. When growing stops, the orientation and residual error are checked again, and the planes are classified as either valid (orientation angle with respect to the gravity vector and residual error within a pre-selected threshold range) or invalid. If valid, in some configurations, the plane can be assigned a score. Additional checks are completed to filter the planes further. For example, the number of times the process outline herein occurs can be compared to a desired maximum that can be configured. The planes can be tested against ground plane criteria according to, for example, but not limited to, a scoring algorithm that can be used to assess the quality of the planes.

[0012] The final set of planes is used to compare LIDAR data against to determine if the LIDAR data represent ground data or obstacle data. The planes are transformed to the frame of reference of the autonomous vehicle, referred to herein as the baselink frame, and a new plane equation is created. The optimum situation is if there is a plane in all directions, i.e. in front of the autonomous vehicle, behind the autonomous vehicle, to the left of the autonomous vehicle and to the right of the autonomous vehicle. If any plane is missing, a plane from a previous time when synchronized LIDAR data and corresponding ground planes were received may be used until the plane becomes stale. If a plane is stale, then no points are fitted in the direction of the stale plane. If there is no plane in any direction, a default X/Y plane can be used. In some configurations, if a plane has not been detected for a period of seven iterations, a default plane in which z+d=0 can be used.

[0013] Now that the ground planes are ready for comparison, the LIDAR points that don't fall within a pre-selected boundary, or that are above a certain height, are filtered from the LIDAR data. The remaining points are transformed to the autonomous vehicle frame of reference, and points that are located in the blind spot of the LIDAR are filtered out. The transformed points are classified according to whether or not they are within one of the previously-determined ground planes. If the points fall on the ground planes, they are marked as free space, otherwise, classified as an obstacle. The points are located on a grid map. If there is already an obstacle at that location, the heights of the obstacles are added together and the number of obstacles at the location is incremented. For each location on the grid map, the probability of occupancy for each cell in the grid map depends upon distance of the cell to the sensor, LIDAR noise, measurement significance, whether the cell is on an obstacle, the presence of a measurement, free space and if the LIDAR is blocked. The probability of occupancy as computed herein follows a Gaussian probability. Logodds of the location's being occupied is computed over the probability of occupancy. The logodds increases when the obstacle is closer to the autonomous vehicle and decreases as the obstacle is farther away from the autonomous vehicle. Beyond the LIDAR range, the logodds are marked as infinite.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The present teachings will be more readily understood by reference to the following description, taken with the accompanying drawings, in which:

[0015] FIGS. **1**A-**1**C are flowcharts of the process of the present teachings;

[0016] FIG. **2** is a schematic block diagram of the system of the present teachings; and

[0017] FIG. **3** is a pictorial diagram of sensor configuration of the present teachings;

[0018] FIG. **4** is a pictorial diagram of segmented point cloud data of the present teachings;

[0019] FIG. **5** is a pictorial diagram of planes within the segmented data of the present teachings;

[0020] FIG. **6** is a pictorial diagram of normals of the present teachings;

[0021] FIG. **7** is a pictorial diagram of obstacle points of the present teachings;

[0022] FIG. **8** is a pictorial diagram of the obstacle grid, blind radius, and sensor beam of the present teachings;

[0023] FIG. **9** is a pictorial diagram of initial probabilities of the present teachings;

[0024] FIG. **9**A is a pictorial diagram of probabilities when the sensor is unavailable;

[0025] FIG. **9**B is a pictorial diagram of probabilities when the sensor is available;

[0026] FIG. **9**C is a pictorial diagram of probabilities when the sensor is available and a cell includes an obstacle;

[0027] FIG. **10** is a pictorial diagram of current probabilities of the present teachings;

[0028] FIGS. **11**A and **11**B are flowcharts of the method of the present teachings;

[0029] FIG. **12**A is a photographic representation of LIDAR rings around an autonomous vehicle;

[0030] FIG. **12**B is a pictorial representation of steps of the method of the present teachings;

[0031] FIG. **13** is a photographic representation of point choices on a LIDAR ring;

[0032] FIG. **14**A is a photographic representation of point choices on adjacent LIDAR rings;

[0033] FIG. **14**B is a pictorial representation of further steps of the method of the present teachings;

[0034] FIG. **15** is a photographic representation of phase one of plane growing of the present teachings;

[0035] FIG. **16** is a photographic representation of phase two of plane growing of the present teachings;

[0036] FIG. **17** is a pictorial representation of a pattern of LIDAR ring arcs formed by a ground plane discovered by the method of the present teachings;

[0037] FIG. **18** is a pictorial representation of grid map updating of the present teachings; and

[0038] FIG. **19** is a schematic block diagram of a second configuration of the system of the present teachings.

DETAILED DESCRIPTION

[0039] The system and method of the present teachings can estimate the free space surrounding an autonomous vehicle in real time.

[0040] Referring now to FIGS. **1**A-**1**C, free space can be estimated from sensor data by sweeping **360** from the sensor, which is placed at the center of a grid, to the perimeter of the grid. Probabilities are calculated and each cell is associated with a logodds of the occupied probability. Logodds can be forced to zero for all cells that fall within a blind radius and do not have a probability of being occupied. The blind radius is the area around the sensor that is blocked by the autonomous vehicle. During the sweep, if a cell has already been visited, new computations can replace previous computations. For each cell, an initial probability can be assigned to each cell based on initially-known information, such as, for example, but not limited to, whether the sensor beam was blocked or the sensor data were unavailable for any reason, and whether an obstacle can be found in the cell. For each cell that is visited outside of the blind radius, a noise factor can be computed. The noise factor recognizes that sensor noise can increase based at least on the distance between the sensor and any obstacles encountered. For each cell, the initial probability and the noise factor can be combined to produce a current probability for the cell, and the grid can be populated with the logodds probabilities. One advantage of drawing a line between the sensor and the perimeter of the grid is that it is possible to see what the sensor detects with a single beam. Another advantage is, if there are no valid returns along the line, it can be assumed that the sensor was blocked or for some other reason unavailable along that line.

[0041] Continuing to refer to FIGS. **1**A-**1**C, method **150** for assigning free space probabilities in point cloud data, where the point cloud data can be associated with an autonomous vehicle traveling on a surface, the method can include, but is not limited to including, receiving **151** the point cloud data from a sensor. The sensor can include a sensor beam, and the sensor beam can project at least from the sensor to the surface. Method **150** can include segmenting **153** the point cloud data into segments of a first pre-selected size, and locating **155** planes, plane points in the planes, and non-plane points associated with at least one of the plane points within the point cloud data. Method **150** can include determining **157** normals to plane points and determining the non-plane points associated with the plane points, and choosing **159** at least one of the planes as a surface plane according to pre-selected criteria based at least on the normals and the location of the sensor. Method **150** can include classifying **161** each of the plane points as an obstacle point based at least on the associated non-plane points, determining **163** obstacle height associated with the obstacle points based at least on the non-plane points, and creating **165** a grid from the surface planes. The grid can include a pre-selected number of cells and a perimeter. Method **150** can include computing **167** a measurement significance for each of the cells based at least on the obstacle heights in the cells, and determining **169** a blind distance from the sensor based at least on an intersection between the sensor beam and the surface plane. The measurement significance can optionally include a value ranging between about 0.09 and 1.0 that can be calculated based on the sum of all obstacle heights in that cell. The higher the total height, the larger is the significance of the measurement. In some configurations, measurement significance=0.09*(28.2*the sum of the obstacle heights in the cell). In some configurations, measurement significance can be restricted to a range of between 0.0.09 and 1.0. If **171** there are more lines to process, and if **173** there are more cells in a line, and if **175** the cell doesn't have a current probability, method **150** can include computing **179** an initial probability of obstacles occupying the cell along the line. If **175** the cell has a current probability, method **150** can include deleting **177** the current probability. The initial probability can be based at least on availability of the sensor, the obstacle points in the cell, and a position of the cell along the line with respect to the sensor. For each cell along each line between the blind distance and the perimeter, method **150** can include computing **181** a noise factor for the cell. The noise factor can be based at least on a first distance between the sensor and a closest of the obstacles along the line in the cell, a second distance between the sensor and the cell along the line, the measurement significance for the cell along the line, the initial probability for the cell along the line, and a default probability. The default probability can include, but is not limited to including, a value of 0.5.

[0042] For each cell along each line between the blind distance and the perimeter, method **150** can include computing **183** a current probability of the obstacles occupying the cell along the line. The current probability can be based at least one the initial probability for the cell and the noise factor for the cell. The first pre-selected size can optionally include about 40 m×40 m×2 m. The pre-selected criteria can optionally include choosing the surface plane when the normals of the at least one plane do not face the sensor. The pre-selected number of cells can optionally include 400×400. Computing the initial probability can optionally include (a) if the sensor is unavailable and the cell is a near cell, the near cell being near the blind distance, the initial probability of the cell=0.9, (b) if the sensor is unavailable and the cell is between the near cell and the perimeter, the initial probability of the cell=0.5, (c) if the sensor is available and there is at least one of the obstacle points in the cell, or if one of the cells along the line that was previously encountered included at least one of the obstacle points, the initial probability of the cell=0.5, and (d) if the sensor is available and there are none of the obstacle points in the cell and none of the cells previously encountered along the line included at least one of the obstacle points, the initial probability of the cell=0.3. Computing the noise factor can optionally include computing:

$$((\text{measurement significance}/(\sigma\sqrt{2\pi}))+0.5-\text{the initial probability of the cell})*\exp(-0.5*((d-Z_t)/\sigma)^2) \quad (1)$$

wherein

d=the second distance,

Z_t=the first distance, and $\sigma=Z_t^2*0.001$.

[0043] For example, if Z_t=10 meters, d=7 meters, and the initial probability=0.3. In this case, σ=0.1, the exp value ~0, making the noise factor ~0. On the other hand, if the distance between the sensor and the observation is closer to the distance between the sensor and the cell, the noise factor will

be non-zero. For example, if Z_t=10 meters, d=9.9 meters, the measurement significance=0.09, and the initial probability=0.3, then, σ=0.1, and the exp value 0.6065. The noise factor=((0.09/(0.1*$\sqrt{2\pi}$))+0.5-0.3)*0.6065=0.339, making sensor noise more important in determining the occupancy probability of the cell. Computing the current probability can optionally include adding the noise factor of the cell to the initial probability of the cell. The at least one plane can optionally include the non-plane points up to a first pre-selected distance from the at least one plane. The first pre-selected distance can optionally include about 2 m.

[0044] Referring now primarily to FIG. **2**, system **100** for assigning free space probabilities in point cloud data **117**, where point cloud data **117** can be associated with autonomous vehicle **203** (FIG. **3**) traveling on surface **205** (FIG. **3**), system **100** can include, but is not limited to including, sensor **125** having sensor beam **201** (FIG. **3**) that can project at least from sensor **125** to surface **205** (FIG. **3**). System **100** can include LIDAR free space estimator **101** that can include, but is not limited to including, segment processor **103** that can receive point cloud data **117** from sensor **125** and segment point cloud data **117** (FIG. **4**) into segments **115** (FIG. **4**) of a first pre-selected size x/y/z (FIG. **4**). LIDAR free space estimator **101** can include plane processor **105** that can locate, within point cloud segments **115** (FIG. **5**), planes **121** (FIG. **5**), plane points **131** (FIG. **5**) in planes **121** (FIG. **5**), and non-plane points **133** (FIG. **5**) associated with at least one of plane points **131** (FIG. **5**). LIDAR free space estimator **101** can include normals processor **107** that can determine normals **123** (FIG. **6**) to plane points **131** (FIG. **6**) and can determine non-plane points **135** (FIG. **6**) associated with plane points **131** (FIG. **6**). Normals processor **107** can choose at least one of planes **121** (FIG. **6**) as surface plane **122** (FIG. **7**) according to pre-selected criteria based at least on normals **123** (FIG. **6**) and the location of sensor **125**. Normals processor **107** can classify each of plane points **131** as obstacle point **124** based at least on associated non-plane points **135** (FIG. **7**), and normals processor **107** can determine obstacle height **139** (FIG. **7**) associated with obstacle points **124** (FIG. **7**) based at least on non-plane points **135** (FIG. **7**). LIDAR free space estimator **101** can include grid processor **109** that can create grid **119** (FIG. **8**) from surface planes **122** (FIG. **7**). Grid **119** (FIG. **8**) can include a pre-selected number of cells **137** (FIG. **8**) and perimeter **138** (FIG. **8**). LIDAR free space estimator **101** can include line sweep processor **111** that can include measurement significance processor **113**. Measurement significance processor **113** can compute a measurement significance for each of cells **137** (FIG. **8**) based at least on obstacle heights **139** (FIG. **7**) in cells **137** (FIG. **8**).

[0045] Continuing to refer primarily to FIG. **2**, line sweep processor **111** can include initial probabilities processor **116** that can determine blind distance **211** (FIGS. **3**, **9**) from sensor **125** based at least on an intersection between sensor beam **201** (FIG. **3**) and surface plane **205** (FIG. **3**). Initial probabilities processor **116** can, for each cell **137** (FIG. **9**) along each line **223** (FIG. **9**) between blind distance **211** (FIG. **9**) and perimeter **138** (FIG. **9**), compute initial probability **225** (FIG. **9**) of obstacles occupying cell **137** (FIG. **9**) along line **223** (FIG. **9**). Initial probability **225** (FIG. **9**) can be based at least on availability of sensor **125**, obstacle points **124** (FIG. **9**), if any, in cell **137** (FIG. **9**), and a position of cell **137** (FIG. **9**) along line **223** (FIG. **9**) with respect to sensor **125**. Line sweep processor **111** can include noise factor processor **118** that can, for each cell **137** (FIG. **9**) along each line **223** (FIG. **9**) between blind distance **211** (FIG. **9**) and perimeter **138** (FIG. **9**), compute noise factor **227** (FIG. **10**) based at least on first distance **229** (FIG. **10**) between sensor **125** (FIG. **10**) and closest of obstacle points **124** (FIG. **10**) along line **223** (FIG. **10**) in cell **137** (FIG. **10**), second distance **231** (FIG. **10**) between sensor **125** (FIG. **10**) and cell **137** (FIG. **10**) along line **223** (FIG. **10**), the measurement significance for cell **137** (FIG. **10**) along line **223** (FIG. **10**), initial probability **225** (FIG. **10**) for cell **137** (FIG. **10**) along line **223** (FIG. **10**), and a default probability. Line sweep processor **111** can include current probabilities processor **119** that can, for each cell **137** (FIG. **10**) along each line **223** (FIG. **10**) between blind distance **211** (FIG. **10**) and perimeter **138** (FIG. **10**), compute current probability **239** (FIG. **10**) of obstacle points **124** (FIG. **10**) occupying cell **137** (FIG. **10**) along line **223** (FIG. **10**). Current probability **239** (FIG. **10**) can be based at least on initial probability **225** (FIG. **10**) for cell **137** (FIG. **10**) and noise factor **227** (FIG. **10**) for cell **137** (FIG. **10**). Segment **115** (FIG. **4**) can optionally include a shape in which x=40 m, y=40, and z=2 m, for example. Surface planes **122** (FIG. **7**) can optionally be chosen because their normals **123** (FIG. **6**) do not face sensor **125** (FIG. **6**). Surface planes **122** (FIG. **8**) can optionally include 400 cells in an x-direction and 400 cells in a y-direction.

[0046] Referring now to FIG. **9**A, when sensor **126** becomes unavailable, initial probability **225**A in cell **243** along line **223**A that is nearest blind distance **211** can be set to 1.0. Cell **241** within blind distance **211** can have a probability set to 0 because the autonomous vehicle is occupying the cell. Line **223**A is the line a sensor beam would traverse if sensor **126** were not blocked or otherwise unavailable. When sensor **126** becomes unavailable, initial probability **225**B in cells **245** along line **223**A that fall between cell **243** and perimeter **138** can be set to 0.5.

[0047] Referring now to FIG. **9**B, if sensor **125** is available, initial probability **225**B can be set to 0.5 if there is at least one of obstacle points **124** in cell **247**.

[0048] Referring now to FIG. **9**C, if sensor **125** is available, initial probability **225**B for cell **249** can be set to 0.5 if one of the cells along line **223**, for example, cell **247**, that was previously encountered during the traversal of line **223** included at least one of obstacle points **124**. If sensor **125** is available and there are none of obstacle points **124** in cell **252**, for example, and none of the cells that were previously encountered along line **223** included at least one of obstacle points **124**, initial probability **225**C of cell **252** can be set to 0.3.

[0049] Referring now to FIG. **10**, noise factor **227** can depend upon the sum of all obstacle heights **139** (FIG. **7**) in cell **137**, for example. Noise factor **227** can depend upon the square of first distance Z_t **229**, and the difference between first distance Z_t **229** and second distance d **231**, and upon initial probability **225**. Current probability **239** can be computed as the sum of initial probability **225** and noise factor **227**.

[0050] Referring now to FIGS. **11**A and **11**B, in another configuration, free space estimation using point cloud data can include locating ground planes from the point cloud data, marking points from the point cloud as free space if they are located on a ground plane, and saving obstructed and free space designations in an occupancy grid as logodds data. Method **250** for performing these functions can

include, but is not limited to including, receiving **251** point cloud data from a sensor, filtering **253** data over the data median in one dimension, creating **255** planes and growing planes to outliers, choosing **257** significant planes, eliminating **259** planes that do not meet threshold score for ground planes, and transforming **261** planes to baselink coordinates. If **263** there are not planes representing points in front of, to the left of, to the right of, and behind the autonomous vehicle, method **250** can include using **265** a previously-used plane that had been available in the immediate previous time step until the previously-used plane had been used in this way for a pre-selected number of iterations. When the previously-used plane has been used up to the pre-selected number of iterations, a default plane can be used. If **263** there are planes representing points in front of, to the left of, to the right of, and behind the autonomous vehicle, method **250** can include filtering **267** point cloud data that are not of interest, transforming **269** point cloud points that survive the filter to baselink coordinates, and filtering **271** transformed points that are too close to the autonomous vehicle. If **273** a transformed and filtered point is located in a ground plane, method **250** can include labeling **275** the point as free space, otherwise, the point is labeled as an obstacle. Method **250** can include labeling **277** each cell on a grid map as free or occupied, depending upon point markings, calculating **279** logodds in an occupancy grid, and setting **281** logodds to ∞ when the cell is beyond the point where the sensor is occluded.

[0051] Referring now to FIG. **12**A, point cloud data can be received as 1D string **303** of points along each LIDAR ring **301** surrounding autonomous vehicle **203**. In some configurations, three 1D arrays can be used to store x, y, and z points. All points along a LIDAR ring can be stored in order of azimuth angle, and the LIDAR rings can be stored consecutively in a row major fashion. Each of the rings can be divided into segments of a pre-selected size, for example, but not limited to, 64 points.

[0052] Referring now to FIG. **12**B, 1D string **303** can be filtered according to a process that can include, but is not limited to including, filtering 1D string **303** around the median of the points of 1D string **303** in each LIDAR data ring. Filtering can include locating points with measured values close to the median and eliminating the rest of the points for this part of the analysis. In some configurations, values that are close to the median are less than 0.1 m from the median. The points that have passed the median filter can be termed a first class of points. Along the median can be found discontinuities in the point data. Discontinuities can be identified in any suitable way, for example, but not limited to calculating the Cartesian distance between points, comparing the distance to a first pre-selected threshold, and identifying discontinuities **309**A/**309**B (FIG. **12**B) or edges of the data as a second class of points when the distance between the points is greater than the first pre-selected threshold. In some configurations, discontinuities arise when

$$\text{abs}(D2-D1)>0.08*2*A$$

[0053] where

[0054] P1=last good point

[0055] P3=point being tested

[0056] P2/P3=consecutive points

[0057] D1=distance between P2 and sensor

[0058] D2=distance between P3 and sensor

[0059] 2=# of points since last good point

[0060] A=(D1+D2)/2

[0061] The points between discontinuities **309**A/**309**B (FIG. **12**B) can be counted and labeled as a third class of points if the number of points exceeds a second pre-selected threshold. In some configurations, the second pre-selected threshold can include eight points. Points lying between other pairs of discontinuities can be discarded.

[0062] Referring now to FIG. **13**, significant planes are expected to fit the terrain around the autonomous vehicle. They have relatively low residual error, are large enough, and are generally representative of the ground points around the autonomous vehicle. In some configurations, a residual error threshold can include 0.1. To determine significant planes, points can be chosen, such as first point **305**A and second point **305**B, from points on the same ring. In some configurations, first point/second point **305**A/B can be selected at random from the third class of points lying between adjacent discontinuities **309**A/**309**B. Other criteria can be used to increase the probability that the points belong to a significant plane.

[0063] Referring now to FIGS. **14**A and **14**B, third point **305**C can be selected from adjacent ring **301**B. Third point **305**C can have an azimuth angle α_3 that can lie between the azimuth angles α_1 and α_2 of first point **305**A and second point **305**B. First/second/third points **305**A/B/C form a plane having a defining equation that can be evaluated for its relationship to a gravity vector. In some configurations, evaluating the plane can include checking the orientation of the plane by choosing planes having a normal vector no more than 60° with respect to the gravity vector provided by, for example, but not limited to, an inertial measurement sensor located on the autonomous vehicle. As the planes are grown and points are added, the orientation angle can be scaled down to 20°.

[0064] Referring now to FIG. **15**, all points remaining from previous filter steps stated herein can be evaluated for their inclusion in polygon **313**A. The edges of polygon **313**A can be defined by first/second points **305**A/**305**B and rings **301**A/B.

[0065] Referring now to FIG. **16**, the plane can be grown in four directions and vertically, forming polygon **313**B. Growing the plane can include evaluating points in all four directions away from the autonomous vehicle towards rings and along azimuth angles that are farther and farther from originally-chosen polygon **313**A of points. The plane can be grown as described herein, and the plane equation can be updated based on the newly-included points, and evaluated for orientation with respect to the gravity vector. Each direction can grow independently until the residual error for that side breaks the threshold or if the side reaches edge **323** of the point cloud. At that time, orientation and residual error checks can be made and, if passed, the plane can be classified as preliminarily significant. Additional checks like number of points, number of growth cycles, and number of vertical growth cycles etc. can be performed to assist in further filtering. If a plane has experienced ten lateral growth cycles or two vertical growth cycles and the plane is not deemed significant, plane growth for that plane can be terminated.

[0066] Referring now to FIG. **17**, data from rings **353** can be assessed as described herein to form planes **351**. From the set of significant planes, surface planes can be identified by subjecting the planes to a scoring function such as, for example, but not limited to:

Residual Score =

$$\begin{cases} 100 & \text{if, residual\_error} < 0.001 \\ \dfrac{-10000 * \text{residual\_error}}{9 + \left(\frac{910}{9}\right)} & \text{if, } 0.001 \leq \text{residual\_error} < 0.01 \\ \dfrac{-1000 * \text{residual\_error}}{9} + \left(\frac{820}{9}\right) & \text{if, } 0.01 \leq \text{residual\_error} < 0.1 \\ -200 * \text{residual\_error} + 100 & \text{if, } 0.1 \leq \text{residual\_error} \end{cases}$$

$$\text{Growth Score} = \begin{cases} 100 & \text{if, num\_vGrowth} > 4 \\ 10 * \text{num\_vGrowth} + 60 & \text{if, num\_vGrowth} \leq 4 \end{cases}$$

$$\text{Area Score} = \frac{\arctan(0.7 * planeArea) * 180}{\pi} + 10$$

$$\text{Angle Score} = \begin{cases} 100 & \text{if, angle} \leq 15 \\ \dfrac{-20 * \text{angle}}{3} + 200 & \text{if, angle} > 15 \end{cases}$$

Final Score = Residual Score + Growth Score + Area Score + Angle Score

The scoring function can be used to assess the quality of the planes, higher scores indicate more likely candidates, and planes that don't meet a pre-selected threshold are discarded.

[0067] Referring now to FIG. **18**, points in the ground planes can be classified as obstacle or free space, and the odds of occupancy at particular locations can be determined. Ground planes can include planes to the right, left, front, and rear of the autonomous vehicle. Each plane is defined by a plane equation and by the type of plane it is. The coordinates of the planes are relative to the location of the sensor and must be transformed to the coordinate system of the autonomous vehicle, the baseline frame of reference. The ground planes each have an equation of the form ax+by +cz+d=0, where the coefficients are a, b, c, and d. To transform the ground plane equations to the baselink frame of reference, rotation and translation are required. One matrix can provide the transformation:

$$[R \mid t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right] \quad (2)$$

From the coefficients, a unity vector can be created:

$$\tilde{u} = \left[\frac{a}{\sqrt[2]{a^2+b^2+c^2}}, \frac{b}{\sqrt[2]{a^2+b^2+c^2}}, \frac{c}{\sqrt[2]{a^2+b^2+c^2}}\right] \quad (3)$$

And d can be normalized:

$$d_{norm} = \frac{d}{\sqrt[2]{a^2+b^2+c^2}} \quad (4)$$

Transformed plane coefficients a, b, and c are:

$$\tilde{v} = [R \mid t]_{3\times4}\begin{bmatrix} \tilde{u} \\ 0 \end{bmatrix}_{4\times1} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{3\times1} \quad (5)$$

Plane coefficient d can be transformed as follows:

$$\tilde{d} = [R \mid t]_{3\times4}\begin{bmatrix} -d_{norm} * \tilde{u}_x \\ -d_{norm} * \tilde{u}_y \\ -d_{norm} * \tilde{u}_z \end{bmatrix} = \begin{bmatrix} \tilde{d}_x \\ \tilde{d}_y \\ \tilde{d}_z \end{bmatrix} \quad (6)$$

And d and d' can be solved for in the following equation:

$$a' * \tilde{d}_x + b' * \tilde{d}_y + c' * \tilde{d}_z + d' = 0 \quad (7)$$

Therefore the transformed plane equation is:

$$a'x + b'y + c'z + d' = 0 \quad (8)$$

In the case in which ground planes are not found in all directions, a previously-used plane in the corresponding direction can be reused, unless the previously-used plane has become stale. A stale count can be maintained for each plane and a previously-used plane can be re-used if it has not exceeded its stale count and if a new plane is not available in its direction. An X/Y plane, in which plane coefficients a=0, b=0, c=1, z+d=0, where d is obtained from the translation in z between LIDAR and baselink frame, is appended to the list of planes in front, left, right and back by default. This is performed as a fail-safe condition for instances when no ground planes are detected.

[0068] Referring now to FIG. **18**, the original point cloud data can be filtered according to, for example, x distance **327** and y distance **325** from the autonomous vehicle, and height **329** above the surface. These parameters and their thresholds can be adjusted based upon the application and the girth and height of the autonomous vehicle. In some configurations, the x distance can include 12.8 m and the y distance can include 12.8 m. The z distance **329** can include a height above which the autonomous vehicle is no longer concerned with whatever obstacle might be present, for example, the height of the point is high enough so that the autonomous vehicle would not encounter the obstacle that the point represents. Whatever points lie in the circumscribed area between blind area **331** and boundaries **325**/**327**, and up to height **329**, are transformed from the sensor frame of reference to the baselink frame of reference: x_{bl}, y_{bl}, z_{bl}. This transformation is obtained by multiplying LIDAR point (x,y,z) with a transformation matrix [R|t] encompassing rotation and translation between the two frames of reference according to equation (2). Therefore (x_{bl}, y_{bl}, z_{bl})=[R|t] *(x,y,z). A Euclidean distance from the autonomous vehicle for each point is computed. If the point satisfies the required boundary requirements, it is plugged into equation (7) for each of the ground planes and checked for satisfying the following inequality:

$$-\text{threshold} \leq a'x_{bl} + b'y_{bl} + c'z_{bl} + d' \leq +\text{threshold} \quad (9)$$

If the above condition is satisfied, then the point x_{bl}, y_{bl}, z_{bl} is said to represent free space on gridmap **333**. Otherwise, the point x_{bl}, y_{bl}, z_{bl} is said to represent an obstacle on gridmap **333**. Gridmap **333** of x_{bl}, y_{bl} locations can be updated based on the results of executing equation (9). In

particular, if the condition is satisfied, the values at point x_{bl}, y_{bl} on gridmap **333** remain unchanged. If the condition is not satisfied, the number of obstacles at point x_{bl}, y_{bl} on gridmap **333** is updated by 1, and the height of the obstacles is updated by the value of z_{bl}. In some configurations, the threshold can include a range of 4-10 in (0.102 m−0.0.254 m). In some configurations, the threshold can include 8 in (0.203 m).

[0069] Continuing to refer to FIG. **18**, for each cell **334** in gridmap **333**, a logodds that the cell is occupied can be computed based on a Gaussian probability, pMapCell, related to the characteristics of the point with respect to, for example, but not limited to, the distance between cell **334** and sensor **336**, noise from sensor **336** (possibly as computed by equation (1)), whether the cell includes a measurement, the significance of the measurement, whether the cell includes an obstacle, the distance between the obstacle and the autonomous vehicle, whether the cell includes free space, and whether the laser is blocked. Significance of measurement is calculated as a product of height of obstacle at that cell, base significance and height normalizer. Base significance and height normalizer are empirically calculated constants. Therefore, the higher the height of obstacle, the higher is the significance of the measurement. The logodds can be computed according to equation (10):

$$\text{LogOdds} = \log\left(\frac{pMapCell}{1 - pMapCell}\right) \tag{10}$$

In some configurations, the pMapCell calculation is different for cells based on pre-selected conditions. For each cell along line **561** joining the position of the autonomous vehicle and end cell **565** of a gridmap, when there are no valid returns from the LIDAR sensor along line **561** joining the position of the autonomous vehicle and end position **565**, something must be blocking the sensor. Therefore, the travel path for the autonomous vehicle should be blocked for safety. For cells that are 0.2 m surrounding the blind distance, pMapCell=PBlocked=0.9. This pMapCell value results in a maximum allowed probability of occupancy. For other cells, pMapCell=PUnknown=0.5. This value results in maximum uncertainty of occupancy. When there are no obstacles found along the line, this means that that line has free space, i.e. pMapCell=PMin=0.3, the minimum allowed probability of occupancy. When the autonomous vehicle encounters a cell that contains an obstacle:

$pMapCell = pOccR + noiseFactor$

[0070] where pOccR=POnObstacle=0.5,

[0071] noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential

[0072] noiseStdDev=square of z_t*LidarNoise

[0073] LidarNoise is an empirical constant

[0074] gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2)))

[0075] z_t=euclidean distance to the obstacle from autonomous vehicle

[0076] d2cell=euclidean distance to the cell from autonomous vehicle

[0077] measurementSignificance=BaseSignificance*(HeightNormalizer*Total_Obstacle_Height)

When the autonomous vehicle encounters a cell in advance of or beyond a cell containing an obstacle:

$pMapCell = pOccR + noiseFactor$

[0078] where pOccR=PMin=0.5,

[0079] noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential

[0080] noiseStdDev=square of z_t*LidarNoise

[0081] LidarNoise is an empirical constant

[0082] gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2)))

[0083] z_t=euclidean distance to the obstacle from autonomous vehicle

[0084] d2cell=euclidean distance to the cell from autonomous vehicle

[0085] measurementSignificance=BaseSignificance

When the autonomous vehicle encounters a cell that is beyond the last obstacle, or when the cell is beyond the last measurement available from LIDAR, for example, beyond boundary **328**, or at point x_{out}, y_{out} **330**,

$pMapCell = 1.0$ (results in infinite logodds)

In some configurations, BaseSignificance and HeightNormalizer can be empirically determined.

In some configurations, BaseSignificance=0.09 and HeightNormalizer=28.2.

[0086] Referring now to FIG. **19**, system **600** for determining free space in a navigation path for an autonomous vehicle can include, but is not limited to including, ground plane processor **603** determining ground planes from point cloud data received from a sensor, each of the ground planes being associated with a ground plane equation, the sensor having a sensor frame of reference. System **600** can include plane transform processor **605** transforming the ground plane equation from the sensor frame of reference to a vehicle frame of reference associated with the autonomous vehicle, point transform processor **607** transforming points in the point cloud data from the sensor frame of reference to the vehicle frame of reference, point label processor **609** labeling points in the point cloud data as free space if the transformed points satisfy the transformed ground plane equation, and probability processor **611** providing occupancy grid data to augment an occupancy grid based at least on the labeled points. System **600** can optionally include executable code including computer instructions substituting a default plane when none of the ground planes can be determined, removing the points from the point cloud data if the points exceed a pre-selected distance from the autonomous vehicle, removing the points from the point cloud data if the points exceed a pre-selected height based at least on a vehicle height of the autonomous vehicle, and removing the points from the point cloud data if the points are within a pre-selected distance from the autonomous vehicle.

[0087] Continuing to refer to FIG. **19**, ground plane processor **603** can optionally include, but is not limited to including, median processor **613** computing a median of at least two rings of the point cloud data, point cloud filter **615** filtering the point cloud data based at least on a distance of the points in the point cloud data from the median, plane creation processor **617** creating planes from the filtered point cloud data, each of the created planes having at least one azimuth angle. Ground plane processor **603** can include plane growth processor **619** growing the created planes from the point cloud data extending away from the autonomous

vehicle along the at least one azimuth angle, and selection processor **621** choosing the ground planes from the grown planes based at least on an orientation and residual error of each of the created planes. Plane creation processor **617** can include, but is not limited to including, executable code including computer instructions selecting a first point and a second point from a first ring of sensor data, the first point and the second point lying within boundaries formed by discontinuities in the point cloud data on the first ring, the first point having a first azimuth angle and the second point having a second azimuth angle, selecting a third point from a second ring of sensor data, the second ring being adjacent to the first ring, the third point having a third azimuth angle between the first azimuth angle and the second azimuth angle, and creating one of the planes including the first point, the second point, and the third point.

[0088] Continuing to refer to FIG. **19**, plane transform processor **605** can optionally include executable code including computer instructions computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d, normalizing the d constant, transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector, and transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients. Plane transform processor **605** can optionally include executable code including computer instructions computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d according to

$$\tilde{u} = \left[\frac{a}{\sqrt[2]{a^2+b^2+c^2}}, \frac{b}{\sqrt[2]{a^2+b^2+c^2}}, \frac{c}{\sqrt[2]{a^2+b^2+c^2}} \right],$$

normalizing the d constant according to

$$d_{norm} = \frac{d}{\sqrt[2]{a^2+b^2+c^2}},$$

transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector according to

$$\bar{v} = [R \mid t]_{3\times 4} \begin{bmatrix} \tilde{u} \\ 0 \end{bmatrix}_{4\times 1} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{3\times 1},$$

and transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients according to

$$\tilde{d} = [R \mid t]_{3\times 4} \begin{bmatrix} -d_{norm} * \tilde{u}_x \\ -d_{norm} * \tilde{u}_y \\ -d_{norm} * \tilde{u}_z \end{bmatrix} = \begin{bmatrix} \tilde{d}_x \\ \tilde{d}_y \\ \tilde{d}_z \end{bmatrix}$$

$$a' * \tilde{d}_x + b' * \tilde{d}_y + c' * \tilde{d}_z + d' = 0.$$

[0089] Continuing to refer to FIG. **19**, point label processor **609** can optionally include executable code including computer instructions plugging each of the transformed point cloud points into the transformed ground plane equations, individually labeling the transformed point cloud points as free space if the transformed point cloud points satisfy the transformed ground plane equation: −threshold ≤a'x+b'y+c'z+d'<+threshold. Probability processor **611** can optionally include executable code including the computer instructions updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation, and computing a probability that a cell in the occupancy grid includes an obstacle based at least on the grid map. Probability processor **611** can optionally include executable code including the computer instructions updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation, and computing a probability that a cell in the occupancy grid includes an obstacle based at least on the grid map, sensor noise, and a height of the obstacle. Probability processor **611** can optionally include executable code including the computer instructions updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation, and computing a probability that a cell in the occupancy grid includes an obstacle according to

$$LogOdds = \log\left(\frac{pMapCell}{1 - pMapCell} \right),$$

where

[0090] pMapCell=0.9 when a distance from the autonomous vehicle to the cell falls in a blind spot of the sensor,

[0091] pMapCell=0.3 when a line between the cell and the autonomous vehicle includes none of the obstacles,

[0092] pMapCell=0.5 for all other cells,

[0093] pMapCell=pOccR+noiseFactor when the cell spatially coincident with the autonomous vehicle includes the obstacle,

[0094] where pOccR=PObstacle=0.5,

[0095] noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+PObstacle−pOccR)*gausianNoiseExponential,

[0096] noiseStdDev=square of z_t*LidarNoise,

[0097] gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2))),

[0098] z_t=Euclidean distance to the obstacle from the autonomous vehicle,

[0099] d2cell=Euclidean distance to the cell from the autonomous vehicle,

[0100] measurementSignificance=BaseSignificance*(HeightNormalizer*Total_Obstacle_Height),

[0101] LidarNoise, BaseSignificance, and HeightNormalizer include values based at least on a configuration of the autonomous vehicle and the sensor,

[0102] pMapCell=pOccR+noiseFactor when the cell falls spatially before or beyond the autonomous vehicle along the line includes the obstacle,

[0103] where pOccR=PMin=0.5,

[0104] noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential

[0105] noiseStdDev=square of z_t*LidarNoise

[0106] LidarNoise is an empirical constant

[0107] gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2)))

[0108] z_t=euclidean distance to the obstacle from the autonomous vehicle,

[0109] d2cell=euclidean distance to the cell from the autonomous vehicle,

[0110] measurementSignificance=BaseSignificance, LidarNoise, BaseSignificance, and HeightNormalizer include the values, and

[0111] pMapCell=1.0 when the cell falls spatially along the line beyond a last of the obstacles or beyond a last of the point cloud data.

BaseSignificance can optionally equal 0.09. HeightNormalizer can optionally equal 28.2.

[0112] Configurations of the present teachings are directed to computer systems for accomplishing the methods discussed in the description herein, and to computer readable media containing programs for accomplishing these methods. The raw data and results can be stored for future retrieval and processing, printed, displayed, transferred to another computer, and/or transferred elsewhere. Communications links can be wired or wireless, for example, using cellular communication systems, military communications systems, and satellite communications systems. Parts of the system can operate on a computer having a variable number of CPUs. Other alternative computer platforms can be used.

[0113] The present configuration is also directed to software for accomplishing the methods discussed herein, and computer readable media storing software for accomplishing these methods. The various modules described herein can be accomplished on the same CPU, or can be accomplished on different computers. In compliance with the statute, the present configuration has been described in language more or less specific as to structural and methodical features. It is to be understood, however, that the present configuration is not limited to the specific features shown and described, since the means herein disclosed comprise preferred forms of putting the present configuration into effect.

[0114] Methods can be, in whole or in part, implemented electronically. Signals representing actions taken by elements of the system and other disclosed configurations can travel over at least one live communications network. Control and data information can be electronically executed and stored on at least one computer-readable medium. The system can be implemented to execute on at least one computer node in at least one live communications network. Common forms of at least one computer-readable medium can include, for example, but not be limited to, a floppy disk, a flexible disk, a hard disk, magnetic tape, or any other magnetic medium, a compact disk read only memory or any other optical medium, punched cards, paper tape, or any other physical medium with patterns of holes, a random access memory, a programmable read only memory, and erasable programmable read only memory (EPROM), a Flash EPROM, or any other memory chip or cartridge, or any other medium from which a computer can read. Further, the at least one computer readable medium can contain graphs in any form, subject to appropriate licenses where necessary, including, but not limited to, Graphic Interchange Format (GIF), Joint Photographic Experts Group (JPEG), Portable Network Graphics (PNG), Scalable Vector Graphics (SVG), and Tagged Image File Format (TIFF).

[0115] While the present teachings have been described above in terms of specific configurations, it is to be understood that they are not limited to these disclosed configurations. Many modifications and other configurations will come to mind to those skilled in the art to which this pertains, and which are intended to be and are covered by both this disclosure and the appended claims. It is intended that the scope of the present teachings should be determined by proper interpretation and construction of the appended claims and their legal equivalents, as understood by those of skill in the art relying upon the disclosure in this specification and the attached drawings.

What is claimed is:

1. A method for assigning free space probabilities in point cloud data, the point cloud data associated with an autonomous vehicle traveling on a surface, the method comprising:

receiving the point cloud data from a sensor, the sensor having a sensor beam, the sensor beam projecting at least from the sensor to the surface;

segmenting the point cloud data into segments of a first pre-selected size;

locating planes, plane points in the planes, and non-plane points associated with at least one of the plane points within the point cloud data;

determining normals to the plane points and determining the non-plane points associated with the plane points;

choosing at least one of the planes as a surface plane according to pre-selected criteria based at least on the normals and a location of the sensor;

classifying each of the plane points as an obstacle point based at least on the associated non-plane points;

determining obstacle height associated with the obstacle points based at least on the non-plane points;

creating a grid from the surface planes, the grid having a pre-selected number of cells and a perimeter;

computing a measurement significance for each of the cells based at least on the obstacle heights in the cells;

determining a blind distance from the sensor based at least on an intersection between the sensor beam and the surface plane;

for each of the cells along each line between the blind distance and the perimeter, computing an initial probability of obstacles occupying the cell along the line, the initial probability being based at least on availability of the sensor, the obstacle points in the cell, and a position of the cell along the line with respect to the sensor;

for each of the cells along each line between the blind distance and the perimeter, computing a noise factor based at least on a first distance between the sensor and a closest of the obstacles along the line in the cell, a second distance between the sensor and the cell along

the line, the measurement significance for the cell along the line, the initial probability for the cell along the line, and a default probability; and

for each of the cells along each line between the blind distance and the perimeter, computing a current probability of the obstacles occupying the cell along the line, the current probability being based at least on the initial probability for the cell and the noise factor for the cell.

2. The method as in claim **1** wherein the first pre-selected size comprises about 40 m×40 m×2 m.

3. The method as in claim **1** wherein the pre-selected criteria comprise choosing the surface plane when the normals of the planes do not face the sensor.

4. The method as in claim **1** wherein the pre-selected number of cells comprises 400×400.

5. The method as in claim **1** wherein computing the initial probability comprises:

if the sensor is unavailable and the cell is a near cell, the near cell being near the blind distance, the initial probability of the cell=1.0;

if the sensor is unavailable and the cell is between the near cell and the perimeter, the initial probability of the cell=0.5;

if the sensor is available and there is at least one of the obstacle points in the cell, or if one of the cells along the line that was previously encountered included at least one of the obstacle points, the initial probability of the cell=0.5; and

if the sensor is available and there are none of the obstacle points in the cell and none of the cells previously encountered along the line included at least one of the obstacle points, the initial probability of the cell=0.3.

6. The method as in claim **1** wherein computing the noise factor comprises:

$$((\text{measurement significance}/(\sigma\sqrt{2\pi}))+0.5-\text{the initial probability of the cell})*\exp(-0.5*((d-Z_t)/\sigma)^2),$$

wherein

d=the second distance,

Z_t=the first distance, and

$\sigma=Z_t^2*0.001$.

7. The method as in claim **1** wherein computing the current probability comprises:

the noise factor of the cell+the initial probability of the cell.

8. The method as in claim **1** wherein each of the planes comprises the non-plane points up to a first pre-selected distance from the planes.

9. The method as in claim **8** wherein the first pre-selected distance comprises 2 m.

10. A system for assigning free space probabilities in point cloud data, the point cloud data associated with an autonomous vehicle traveling on a surface, the system comprising:

a sensor having a sensor beam, the sensor beam projecting at least from the sensor to the surface;

a segment processor receiving the point cloud data from the sensor, the segment processor segmenting the point cloud data into segments of a first pre-selected size;

a plane processor locating, within the point cloud data, planes, plane points in the planes, and non-plane points associated with at least one of the plane points;

a normals processor determining normals to the plane points and determining the non-plane points associated with the plane points, the normals processor choosing at least one of the planes as a surface plane according to pre-selected criteria based at least on the normals and a location of the sensor, the normals processor classifying each of the plane points as an obstacle point based at least on the associated non-plane points, the normals processor determining obstacle height associated with the obstacle points based at least on the non-plane points;

a grid processor creating a grid from the surface planes, the grid having a pre-selected number of cells and a perimeter;

a line sweep processor including a measurement significance processor computing a measurement significance for each of the cells based at least on the obstacle heights in the cells, the line sweep processor including

an initial probabilities processor determining a blind distance from the sensor based at least on an intersection between the sensor beam and the surface plane, the initial probabilities processor including for each of the cells along each line between the blind distance and the perimeter, computing an initial probability of obstacles occupying the cell along the line, the initial probability being based at least on availability of the sensor, the obstacle points in the cell, and a position of the cell along the line with respect to the sensor;

a noise factor processor including for each of the cells along each line between the blind distance and the perimeter, computing a noise factor based at least on a first distance between the sensor and a closest of the obstacles along the line in the cell, a second distance between the sensor and the cell along the line, the measurement significance for the cell along the line, the initial probability for the cell along the line, and a default probability; and

a current probabilities processor including for each of the cells along each line between the blind distance and the perimeter, computing a current probability of the obstacle points occupying the cell along the line, the current probability being based at least on the initial probability for the cell and the noise factor for the cell.

11. A method for assigning free space probabilities in sensor data, the sensor data associated with an autonomous vehicle traveling on a surface, the method comprising:

determining at least one surface plane in the sensor data, the at least one surface plane being associated with the surface;

determining obstacles, if any, and heights of the obstacles, if any, in the sensor data associated with the at least one surface plane;

determining a blind distance from the autonomous vehicle based at least on a dimension of the autonomous vehicle;

creating a grid on the at least one surface plane, the grid having a pre-selected number of cells and a perimeter;

for each of the cells along each line on the at least one surface plane between the blind distance and the perimeter, computing an initial probability of the obstacles occupying the cell along the line, the initial probability being based at least on availability of the sensor data, the obstacles in the cell, and a position of the cell along the line with respect to the autonomous vehicle;

for each of the cells along each line between the blind distance and the perimeter, computing a noise factor based at least on a first distance between the autonomous vehicle and a closest of the obstacles along the line in the cell, a second distance between the autonomous vehicle and the cell along the line, the obstacle heights for the cell along the line, the initial probability for the cell along the line, and a default probability; and

for each of the cells along each line between the blind distance and the perimeter, computing a current probability of the obstacles occupying the cell along the line, the current probability being based at least on the initial probability for the cell and the noise factor for the cell.

12. The method as in claim **11** wherein the pre-selected number of cells comprises 400×400.

13. The method as in claim **11** wherein computing the initial probability comprises:

if the sensor data are unavailable and the cell is a near cell, the near cell being near the blind distance, the initial probability of the cell=1.0;

if the sensor data are unavailable and the cell is between the near cell and the perimeter, the initial probability of the cell=0.5;

if the sensor data are available and there is at least one of the obstacles in the cell, or if one of the cells along the line that was previously encountered included at least one of the obstacles, the initial probability of the cell=0.5; and

if the sensor data are available and there are none of the obstacles in the cell and none of the cells previously encountered along the line included at least one of obstacles, the initial probability of the cell=0.3.

14. The method as in claim **11** wherein computing the noise factor comprises:

$$((0.09*28.2*\Sigma \text{ the obstacle heights in the cell}/(\sigma\sqrt{2\pi}))+0.5-\text{the initial probability of the cell})*\exp(-0.5*((d-Z_t)/\sigma)^2)$$

wherein

d=the second distance,

Z_t=the first distance, and

$\sigma=Z_t^2*0.001$.

15. The method as in claim **11** wherein computing the current probability comprises:

the noise factor of the cell+the initial probability of the cell.

16. The method as in claim **11** wherein the at least one surface plane comprises non-plane points up to a first pre-selected distance from the at least one surface plane.

17. The method as in claim **16** wherein the first pre-selected distance comprises 2 m.

18. The method as in claim **16** further comprising:

determining the obstacle heights based at least on the non-plane points.

19. A method for determining free space in a navigation path for an autonomous vehicle, the method comprising:

determining ground planes from point cloud data received from a sensor, each of the ground planes being associated with a ground plane equation, the sensor having a sensor frame of reference;

transforming the ground plane equation from the sensor frame of reference to a vehicle frame of reference associated with the autonomous vehicle;

transforming points in the point cloud data from the sensor frame of reference to the vehicle frame of reference;

labeling points in the point cloud data as free space if the transformed points satisfy the transformed ground plane equation; and

providing occupancy grid data to augment an occupancy grid based at least on the labeled points.

20. The method as in claim **19** wherein determining the ground planes comprises:

computing a median of at least two rings of the point cloud data;

filtering the point cloud data based at least on a distance of the points in the point cloud data from the median;

creating planes from the filtered point cloud data, each of the created planes having at least one azimuth angle;

growing the created planes from the point cloud data extending away from the autonomous vehicle along the at least one azimuth angle; and

choosing the ground planes from the grown planes based at least on an orientation and residual error of each of the created planes.

21. The method as in claim **20** wherein creating the planes comprises:

selecting a first point and a second point from a first ring of sensor data, the first point and the second point lying within boundaries formed by discontinuities in the point cloud data on the first ring, the first point having a first azimuth angle and the second point having a second azimuth angle;

selecting a third point from a second ring of sensor data, the second ring being adjacent to the first ring, the third point having a third azimuth angle between the first azimuth angle and the second azimuth angle; and

creating one of the planes including the first point, the second point, and the third point.

22. The method as in claim **19** further comprising:

substituting a default plane when none of the ground planes can be determined.

23. The method as in claim **19** further comprising:

removing the points from the point cloud data if the points exceed a pre-selected distance from the autonomous vehicle.

24. The method as in claim **19** further comprising:

removing the points from the point cloud data if the points exceed a pre-selected height based at least on a vehicle height of the autonomous vehicle.

25. The method as in claim **19** further comprising:

removing the points from the point cloud data if the points are within a pre-selected distance from the autonomous vehicle.

26. The method as in claim **19** wherein transforming the ground planes comprises:

computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d;

normalizing the d constant;

transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector; and

transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients.

27. The method as in claim **19** wherein transforming the ground planes comprises:

computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d;

$$\tilde{u} = \left[\frac{a}{\sqrt[2]{a^2+b^2+c^2}}, \frac{b}{\sqrt[2]{a^2+b^2+c^2}}, \frac{c}{\sqrt[2]{a^2+b^2+c^2}} \right]$$

normalizing the d constant;

$$d_{norm} = \frac{d}{\sqrt[2]{a^2+b^2+c^2}}$$

transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector; and

$$\bar{v} = [R \mid t]_{3\times4} \begin{bmatrix} \tilde{u} \\ 0 \end{bmatrix}_{4\times1} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{3\times1}$$

transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients

$$\tilde{d} = [R \mid t]_{3\times4} \begin{bmatrix} -d_{norm} * \tilde{u}_x \\ -d_{norm} * \tilde{u}_y \\ -d_{norm} * \tilde{u}_z \end{bmatrix} = \begin{bmatrix} \tilde{d}_x \\ \tilde{d}_y \\ \tilde{d}_z \end{bmatrix}$$

$$a' * \tilde{d}_x + b' * \tilde{d}_y + c' * \tilde{d}_z + d' = 0.$$

28. The method as in claim **27** wherein labeling points comprises:

plugging each of the transformed point cloud points into the transformed ground plane equations;

individually labeling the transformed point cloud points as free space if the transformed point cloud points satisfy the transformed ground plane equation:

$$-\text{threshold} \le a'x+b'y+c'z+d' \le +\text{threshold}.$$

29. The method as in claim **28** wherein providing occupancy grid data comprises:

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes the obstacle based at least on the grid map.

30. The method as in claim **28** wherein providing occupancy grid data comprises:

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes an obstacle based at least on the grid map, sensor noise, and a height of the obstacle.

31. The method as in claim **28** wherein providing occupancy grid data comprises:

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes an obstacle according to

$$LogOdds = \log\left(\frac{pMapCell}{1 - pMapCell} \right)$$

Where

pMapCell=0.9 when a distance from the autonomous vehicle to the cell falls in a blind spot of the sensor,

pMapCell=0.3 when a line between the cell and the autonomous vehicle includes none of the obstacles,

pMapCell=0.5 for all other cells,

pMapCell=pOccR+noiseFactor when the cell spatially coincident with the autonomous vehicle includes the obstacle,

where pOccR=POnObstacle=0.5,

noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential,

noiseStdDev=square of z_t*LidarNoise,

gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2))),

z_t=Euclidean distance to the obstacle from the autonomous vehicle,

d2cell=Euclidean distance to the cell from the autonomous vehicle,

measurementSignificance=BaseSignificance*(HeightNormalizer*Total_Obstacle_Height),

LidarNoise, BaseSignificance, and HeightNormalizer include values based at least on a configuration of the autonomous vehicle and the sensor,

pMapCell=pOccR+noiseFactor when the cell falls spatially before or beyond the autonomous vehicle along the line includes the obstacle,

where pOccR=PMin=0.5,

noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential

noiseStdDev=square of z_t*LidarNoise

LidarNoise is an empirical constant

gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2)))

z_t=euclidean distance to the obstacle from the autonomous vehicle,

d2cell=euclidean distance to the cell from the autonomous vehicle,

measurementSignificance=BaseSignificance,

LidarNoise, BaseSignificance, and HeightNormalizer include the values, and

pMapCell=1.0 when the cell falls spatially along the line beyond a last of the obstacles or beyond a last of the point cloud data.

32. The method as in claim **31** wherein BaseSignificance comprises 0.09.

33. The method as in claim **31** wherein HeightNormalizer comprises 28.2.

34. A system for determining free space in a navigation path for an autonomous vehicle, the system comprising:

a ground plane processor determining ground planes from point cloud data received from a sensor, each of the ground planes being associated with a ground plane equation, the sensor having a sensor frame of reference;

a plane transform processor transforming the ground plane equation from the sensor frame of reference to a vehicle frame of reference associated with the autonomous vehicle;

a point transform processor transforming points in the point cloud data from the sensor frame of reference to the vehicle frame of reference;

a point label processor labeling points in the point cloud data as free space if the transformed points satisfy the transformed ground plane equation; and

a probability processor providing occupancy grid data to augment an occupancy grid based at least on the labeled points.

35. The system as in claim **34** wherein the ground plane processor comprises:

a median processor computing a median of at least two rings of the point cloud data;

a point cloud filter filtering the point cloud data based at least on a distance of the points in the point cloud data from the median;

a plane creation processor creating planes from the filtered point cloud data, each of the created planes having at least one azimuth angle;

a plane growth processor growing the created planes from the point cloud data extending away from the autonomous vehicle along the at least one azimuth angle; and

a selection processor choosing the ground planes from the grown planes based at least on an orientation and residual error of each of the created planes.

36. The system as in claim **35** wherein the plane creation processor comprises:

executable code including computer instructions

selecting a first point and a second point from a first ring of sensor data, the first point and the second point lying within boundaries formed by discontinuities in the point cloud data on the first ring, the first point having a first azimuth angle and the second point having a second azimuth angle;

selecting a third point from a second ring of sensor data, the second ring being adjacent to the first ring, the third point having a third azimuth angle between the first azimuth angle and the second azimuth angle; and

creating one of the planes including the first point, the second point, and the third point.

37. The system as in claim **34** further comprising:

executable code including computer instructions substituting a default plane when none of the ground planes can be determined.

38. The system as in claim **34** further comprising:

executable code including computer instructions removing the points from the point cloud data if the points exceed a pre-selected distance from the autonomous vehicle.

39. The system as in claim **34** further comprising:

executable code including computer instructions removing the points from the point cloud data if the points exceed a pre-selected height based at least on a vehicle height of the autonomous vehicle.

40. The system as in claim **34** further comprising:

executable code including computer instructions removing the points from the point cloud data if the points are within a pre-selected distance from the autonomous vehicle.

41. The system as in claim **34** wherein transforming the ground planes comprises:

executable code including computer instructions

computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d;

normalizing the d constant;

transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector; and

transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients.

42. The system as in claim **34** wherein transforming the ground planes comprises:

executable code including computer instructions

computing a unity vector from coefficients of the ground plane equation, the ground plane equation including ax+by+cz+d=0, the coefficients including a, b, and c, a constant including d;

$$\tilde{u} = \left[\frac{a}{\sqrt[2]{a^2+b^2+c^2}}, \frac{b}{\sqrt[2]{a^2+b^2+c^2}}, \frac{c}{\sqrt[2]{a^2+b^2+c^2}} \right]$$

normalizing the d constant;

$$d_{norm} = \frac{d}{\sqrt[2]{a^2+b^2+c^2}}$$

transforming the a, b, c coefficients of the ground plane equation based on a rotation/translation matrix and the unity vector; and

$$\bar{v} = [R \mid t]_{3\times4} \begin{bmatrix} \tilde{u} \\ 0 \end{bmatrix}_{4\times1} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}_{3\times1}$$

transforming the normalized d constant based on the normalized d constant, the rotation/translation matrix, the unity vector, and the transformed a, b, c coefficients,

$$\tilde{d} = [R \mid t]_{3\times4}\begin{bmatrix} -d_{norm} * \tilde{u}_x \\ -d_{norm} * \tilde{u}_y \\ -d_{norm} * \tilde{u}_z \end{bmatrix} = \begin{bmatrix} \tilde{d}_x \\ \tilde{d}_y \\ \tilde{d}_z \end{bmatrix}$$

$$a' * \tilde{d}_x + b' * \tilde{d}_y + c' * \tilde{d}_z + d' = 0.$$

43. The system as in claim **34** wherein the point label processor comprises:

executable code including computer instructions

plugging each of the transformed point cloud points into the transformed ground plane equations;

individually labeling the transformed point cloud points as free space if the transformed point cloud points satisfy the transformed ground plane equation:

$$-\text{threshold} \leq a'x+b'y+c'z+d' \leq +\text{threshold}.$$

44. The system as in claim **43** wherein the probability processor comprises:

the executable code including the computer instructions

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes an obstacle based at least on the grid map.

45. The system as in claim **43** wherein the probability processor comprises:

the executable code including the computer instructions

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes an obstacle based at least on the grid map, sensor noise, and a height of the obstacle.

46. The system as in claim **43** wherein the probability processor comprises:

the executable code including the computer instructions

updating locations on a grid map corresponding to the transformed point cloud points based at least on results from the transformed ground plane equation; and

computing a probability that a cell in the occupancy grid includes an obstacle according to

$$LogOdds = \log\left(\frac{pMapCell}{1 - pMapCell}\right)$$

Where

pMapCell=0.9 when a distance from the autonomous vehicle to the cell falls in a blind spot of the sensor,

pMapCell=0.3 when a line between the cell and the autonomous vehicle includes none of the obstacles,

pMapCell=0.5 for all other cells,

pMapCell=pOccR+noiseFactor when the cell spatially coincident with the autonomous vehicle includes the obstacle,

where pOccR=POnObstacle=0.5,

noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential,

noiseStdDev=square of z_t*LidarNoise,

gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2))),

z_t=Euclidean distance to the obstacle from the autonomous vehicle,

d2cell=Euclidean distance to the cell from the autonomous vehicle,

measurementSignificance=BaseSignificance*(HeightNormalizer*Total_Obstacle_Height),

LidarNoise, BaseSignificance, and HeightNormalizer include values based at least on a configuration of the autonomous vehicle and the sensor,

pMapCell=pOccR+noiseFactor when the cell falls spatially before or beyond the autonomous vehicle along the line includes the obstacle,

where pOccR=PMin=0.5,

noiseFactor=((measurementSignificance/(noiseStdDev*Root of 2Pi))+POnObstacle−pOccR)*gausianNoiseExponential

noiseStdDev=square of z_t*LidarNoise

LidarNoise is an empirical constant

gausianNoiseExponential=pow(EXP, (−0.5*pow(((d2Cell−z_t)/noiseStdDev), 2)))

z_t=euclidean distance to the obstacle from the autonomous vehicle,

d2cell=euclidean distance to the cell from the autonomous vehicle,

measurementSignificance=BaseSignificance,

LidarNoise, BaseSignificance, and HeightNormalizer include the values, and

pMapCell=1.0 when the cell falls spatially along the line beyond a last of the obstacles or beyond a last of the point cloud data.

47. The system as in claim **46** wherein BaseSignificance comprises 0.09.

48. The system as in claim **46** wherein HeightNormalizer comprises 28.2.

* * * * *