Data Processing in Current 3D Robotic Perception Systems

Cang Ye

Department of Systems Engineering
University of Arkansas at Little Rock, Little Rock, AR 72204, USA
Tel: (501)-683-7284    Email: cxye@ualr.edu

1. Introduction

The past decade has witnessed tremendous advancements in robotic autonomy. One of the driving forces making this happen is the advance in 3D robotic vision. New 3D imaging sensors and data processing methods have drastically changed the way a robot interacts with its environment and improved its intelligence. The most popular 3D perception sensors in robotics include stereovision system and LIDAR (Light Detection and Ranging). Recently a new class of 3D imaging sensor—Flash LIDAR Camera (FLC)—has been developed and applied to robotic vision\(^1\). The commercially available products include the SwissRanger SR4000 and the TigerEyer FLCs.

1.1. Stereovision

Stereovision has been widely used in robotics. The representative applications of stereovision have been demonstrated in NASA’s Mars Exploration Rover [1, 2] program and the DARPA’s Learning Applied to Ground Robots [3, 4, 5, 6] program. Projects in both programs use stereovision as the major/single perception sensor for robot navigation including robot pose estimation and obstacle detection/avoidance. A stereovision resembles human eye’s depth perception. It performs stereo matching between features in the left and right images of the sensor and determines the depth (range) information of the matched features by using their disparity information. A stereovision system is cost-effective and has low power dissipation. However, it has the following disadvantages: (1) its range measurement accuracy drops quadratically with the true distance; (2) it cannot produce complete range data of the environment. These properties are illustrated by Fig. 1 and Fig. 2. The black curve in Fig. 1 depicts the Bumblebee2 stereovision system’s [3] range measurement error that increases quadratically with the true range. The measurement error is 1.6 cm (~1%) at 1.5 meters and 17 cm (3.4%) at 5 meters. This means that the system can maintain good measurement accuracy only for a short range. Figure 2b shows the range image of the Bumblebee2 for an indoor scene with a stairway. It

\(^1\) Since Microsoft introduced Kinect for Xbox 360 in 2010 as a remote-free user interface, the Kinect sensor has attracted much attention from the robotics community. The sensor projects light pattern on the scene and detect the range through triangulation. It produces both intensity and range image just like the FLC does. The methods developed for the FLC may be adapted to the Kinect with ease. The details of the Kinect sensor are beyond the scope of this article.
can be seen that there are quite a lot of missing data that may affect the usefulness of the range data.

Robot pose estimation is a process of determining a robot’s position \((X, Y, Z)\) and Euler angles \((\phi, \theta, \psi)\) in the world coordinate. The representative stereovision-based pose estimation approach is the Visual Odometry (VO) algorithm [2]. The VO method estimates the robot’s ego-motion (i.e., pose change) by detecting features in a stereo image pair and tracking them from one frame to the next. The features’ 3D positions in each frame are determined by stereo matching. Feature tracking between frames is performed by selecting features in an image and locating them in the subsequent image by spatial correlation search. The VO method is only effective in a short range, i.e., it may incur large pose estimation errors for distant features. This is because a stereovision system’s depth measurement error quadratically increases with the true distance. To retain certain pose estimation accuracy, the robot may need to point the stereovision system to the terrain at its immediate vicinity. This reduces the robot’s look-ahead distance and may affect the dual use of the stereovision data for obstacle avoidance. In principle, the VO method assumes a static environment for feature matching. This assumption does not hold when the robot’s shadow is within the field of view of the stereovision system. The robot shadow may produce moving features which sometimes may cause failure of the VO method [7].

Obstacle detection/avoidance consists of a procedure of collecting 3D data from the sensor, building a terrain map, analyzing the traversal property of the map and determining proper action for the robot. Usually an elevation map is used to describe terrain and the traversal property of each cell of the map can be described by its drivability (drivable, occupied, or unknown) [8] or the so-call Traversability Index (TI) [9]. The TI of a cell is calculated using the slope of the Least-Squares Plane (LSP) of the terrain patch surrounding the cell and the LSP fitting error. Terrain data captured by stereovision is not reliable for obstacle avoidance as it may contain excessive missing range data. This is because stereo-matching can only be performed on feature-rich points and thus featureless points generate missing data. Incomplete range data may affect the effectiveness of obstacle avoidance. Dense stereo [4, 5] may improve data density at the cost of a longer processing time and thus a lower frame rate. However, the range data is still incomplete and the density of range data depends on scene.

### 1.2. LIDAR

A LIDAR system [10, 11] outperforms a stereovision system in terms of range data completeness and accuracy. It has a consistent range measurement error of a few millimeters and is less prone to the missing data problem. As a result, it has been widely used in robotic perception. For example, the DARPA grand challenge vehicle [12] uses a sensor suite comprising cameras, RADAR sensors and 2D LIDARs for terrain mapping and obstacle avoidance. In the most recent Urban Challenge competition, the Velodyne 3D LIDARs [13] were used for autonomous navigation of the robotic vehicles [14, 15]. Although these robotic systems have successfully demonstrated much higher autonomy, the size, weight and power consumption of the sensor suites make such a multi-sensor approaches unsuitable for small robots.

The Velodyne LIDAR may generate dense laser data (133,333 points/frame with a 10 Hz frame rate) for terrain mapping. However, it is too expensive and bulky for a small robot. To overcome this problem, the author developed a 3D terrain mapping and navigation method using a single 2D LIDAR in [9]. In the mapping system, the robot movement sweeps the 2D LIDAR’s laser scan-line over the terrain and produces 3D terrain data over time. These laser scans are registered into a 3D map using the robot pose information computed from the data of wheel-odometry and a 3-axis gyro. The system requires accurate robot pose information for laser data registration. Its efficacy has been validated on much smaller robotic platforms—the Segway RMP [9] and the Pioneer robot [16].

### 1.3. Flash LIDAR Camera (FLC)

A FLC illuminates the entire scene with a single laser pulse (or modulated infrared light) and focuses the image onto the 3D sensor Focal Plane Array (FPA). Each pixel on the FPA measures the Time-Of-Flight (TOF) and thus the object distance to produces a cloud of points. The FLC also produces an intensity image simultaneously. Currently, the commercially available systems include the SwissRanger SR4000 [17] and the TigerEye [18] 3D FLCs. The former uses modulated infrared light and computes the TOF through phase shift measurement while the latter uses laser pulse and direct TOF measurement. A FLC has the following advantages over stereovision: (1) it measures depth by TOF and therefore has consistent measurement accuracy (~±1 cm) in its full measurement range; (2) it is able to produce complete range data of the scene; and (3) as an active sensor it can be used for both daytime and nighttime driving and is not affected by the self-shadow problem. Figure 1 compares the range measurement accuracies of the SwissRanger SR4000 and the Bumblebee2 stereovision system. The SR4000 (red) has a consistent measurement error (~±1 cm) for a range up to 5 m while the stereovision’s error (black) increases quadratically with range (~1 cm at 1.25 meters, and ~17 cm at 5 meters). The data completeness of the SR4000 is demonstrated in Fig. 2(c) and (d).

The SR4000 (Fig. 3) uses a CMOS imaging sensor. It is small in size (65x65x68 mm³) and...
has a power dissipation of 9.6W (even lower dissipation in trigger mode). The sensor illuminates its entire environment with diffuse and modulated infrared light. Based on phase shift measurement, it detects the range values up to \textbf{5 meters} (with \( \pm 1 \) cm accuracy) for every point sensed by the imaging pixels (176 \( \times \) 144 = 25,344 pixels). The sensor produces a 3D range image and an intensity image simultaneously at a rate up to 54 frames per second. The small size and the capability of producing both visual and range data has made it an ideal imaging sensor for small robot autonomy. Simultaneous visual and range data processing allows for robot pose estimation, obstacle detection, and scene recognition by using a single sensor modality.

The remainder of the chapter is organized as follows: Section 2 presents a complete LIDAR-based terrain mapping and navigation system. Section 3 introduces a robot pose estimation and a 3D range data segmentation methods in a FLC-based perception system. Section 4 concludes the chapter and discusses some future research directions.

2. A LIDAR-based terrain mapping and navigation system

The terrain mapping and navigation system is depicted in Fig. 4. It uses a Pioneer P3DX robot as mobile platform. The sensors used for mapping and navigation include a 2D LIDAR (Sick LMS 200) and a 3-axis fiber optic gyro. The gyro measures the robot’s Euler angular rate which is combined with the wheel-odometry data for robot pose estimation. The onboard computer uses a 2.0 GHz Pentium 4 processor. It acquires the laser range data from the LIDAR and the robot Euler rate from the gyro through a RS-422 card and the wheel-odometry data through a RS-232 port. A high speed RS422 serial card (Quatech DSC-200/300 PCI board) is used to communicate with the LIDAR at 500k baud. The gyro and odometry data are used to estimate the robot’s pose, based on which the laser data is registered in to a terrain map. The navigational software then analyzes the terrain map and determines suitable motion commands to control the robot. The onboard computer also transmits the robot’s pose and laser data to an off-board laptop computer via the wireless router. The process of terrain mapping and navigation is thus visualized on the laptop computer in real-time.

2.1. Overview of the Mapping & Navigation System

As depicted in Fig. 5, the terrain mapping and navigation system consists of three main modules: Terrain Mapping, Terrain Traversability Analysis (TTA) and Path Planning. The perception sensor (the LIDAR) is mounted on the front-end of the robot. It looks forward and downward at the terrain with an angle of -10° from the robot base. While the robot is in motion, the fanning laser beams profile the terrain ahead of the robot and produce a terrain map. In the system, a 2½D grid-type terrain map (elevation map) is used for navigation.

![Fig. 4. The P3DX robot equipped with sensors for mapping and navigation](image)

![Fig. 5. Diagram of the navigation system: the module within the dashed lines is the Terrain Mapping module](image)
The TTA module transforms the terrain map into a traversability map where each cell holds a value representing the degree of difficulty for the robot to move across that cell. This value is called Traversability Index (TI). The Path Planning module converts the traversability map into a number of PTIs each of which represents the level of difficulty for the robot to move along the corresponding direction. Based on the PTIs, the Path Planning module determines the steering and velocity commands of the robot and sends these motion commands to control the robot.

2.2. Terrain Mapping

Terrain mapping is a process of transforming laser range measurements to 3D points in the world coordinate (also called navigational frame) and registering these data points into a map. In this work, an Extended Terrain Map (ETM) is first built. Each ETM consists of an elevation map and a so-called “certainty map.” Both are 2D grid-type maps. Each cell in the elevation map holds a value representing the height of the object at that cell; while each cell in the certainty map holds a value representing the certainty that the corresponding cell in the elevation map is occupied by the object.

Homogeneous coordinate transformation is used to convert a range measurement into a point in the navigational frame. Figure 6 depicts the coordinate systems used for terrain mapping. The left figure depicts the diagram of the robot and the right figure details the coordinate systems fixed on the robot body. In Fig. 6, $x_b y_b z_b$ is the robot body frame fixed to the midpoint of the wheel axle. Frames $x_s y_s z_s$ and $x_l y_l z_l$ are the coordinate systems fixed to the LRF’s mounting bracket and the LRF, respectively. The origin of frame $x_s y_s z_s$ locates at $(0, m, g+h)$ in frame $x_b y_b z_b$. The angle between $z_s$ and $z_l$ is $\alpha$ which is the LRF’s initial tilt-down angle, i.e., -10°. Frame $x_r y_r z_r$ is the coordinate system attached to the laser receiver and its origin locates at $(0, b, p)$ in frame $x_l y_l z_l$. The navigational frame $x_n y_n z_n$ is aligned with the robot body frame when a navigation task is specified and it remains constant throughout the entire navigation task. The origin of the robot body frame is at $(u, v, w)$ in the navigational frame. The three Euler angles are defined as follows: roll ($\phi$), pitch ($\theta$) and yaw ($\psi$) which are the rotation angles around $y_b$, $x_b$, and $z_b$ axes, respectively.

The homogeneous transformation matrix $T_w^b$ can be derived through successive rotations and translations from frame $x_s y_s z_s$ to frame $x_l y_l z_l$. The coordinates of a point in frame $x_s y_s z_s$ can be converted into the coordinates of a point in frame $x_l y_l z_l$ by applying a roll rotation $\text{Rot}_y(\phi)$ followed by a pitch rotation $\text{Rot}_x(\theta)$, a yaw rotation $\text{Rot}_y(\psi)$ and a translation $\text{Trans}(u, v, w)$. The corresponding transformation matrix $T_w^b$ is given by

$$T_w^b = \text{Trans}(u, v, w)\text{Rot}_y(\psi)\text{Rot}_x(\theta)\text{Rot}_y(\phi) = \begin{bmatrix} (c \psi c \phi - s \psi s \phi) & -s \psi c \phi & c \psi s \phi + s \psi c \phi \vspace{1mm} \\
-s \psi c \phi & c \psi c \phi & s \psi s \phi - c \psi c \phi \vspace{1mm} \\
-c \phi & c \psi & s \phi \end{bmatrix},$$

where $c x$ and $s x$ stands for $\cos(x)$ and $\sin(x)$, respectively. Similarly, the transformation matrix that converts the coordinates of a point in frame $x_s y_s z_s$ into the coordinates of a point in frame $x_b y_b z_b$ is...
\[ T^b_r = \text{Trans}(0,m,g+h)\text{Rot}(\alpha)\text{Trans}(0,b,p) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & ca - sa & bc\alpha - psa + m \\ 0 & sa & bs\alpha + pca + g + h \\ 0 & 0 & 0 & 1 \end{bmatrix} \] (2)

Therefore, \( T^* \) is computed by

\[ T^* = T^b_r T^b_r = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \] (3)

A range measurement \( l \) is converted into a 3D point \( \vec{d} = (l \cos \beta, l \sin \beta, 0) \) in frame \( x_yz \), where \( \beta = \pi \times 1/180 \) is the scanning angle of the \( n \)th laser measurement. Transforming \( \vec{d} \) into a point \( (x_n, y_n, z_n) \) in the navigational frame produces

\[ \vec{d}^* = T^* \vec{d} = \begin{bmatrix} q_{11}l\cos \beta + q_{12}l\sin \beta + q_{14} \\ q_{21}l\cos \beta + q_{22}l\sin \beta + q_{24} \\ q_{31}l\cos \beta + q_{32}l\sin \beta + q_{34} \\ 1 \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix}. \] (4)

For every two consecutive measurements (at time steps \( t \) and \( t+1 \)) on the object at the same location \( (x_n, y_n) \), the expected change in height value \( \Delta z \) should not be bigger than

\[ \Delta z_{\max}' = \Delta u \tan \phi - \Delta v \tan(\theta + \alpha) + \Delta w + \frac{\partial z}{\partial \psi} \Delta \psi + \frac{\partial z}{\partial \theta} \Delta \theta + \frac{\partial z}{\partial \phi} \Delta \phi + \frac{\partial z}{\partial l} \Delta l, \] (5)

where \( \Delta l \) is the maximum measurement error of the LRF (32 mm according to [10]); \( \Delta u, \Delta v, \Delta w, \Delta \phi, \Delta \theta \) and \( \Delta \psi \) are the changes in the robot’s pose from \( t \) to \( t+1 \). \( \frac{\partial z}{\partial \theta}, \frac{\partial z}{\partial \phi}, \frac{\partial z}{\partial \psi}, \frac{\partial z}{\partial \theta}, \frac{\partial z}{\partial \phi}, \frac{\partial z}{\partial \psi}, \frac{\partial z}{\partial \phi} \) and \( \frac{\partial z}{\partial \psi} \) are the partial derivatives and they can be derived from (4).

In this work, a grid cell size of 80x80 mm is used. The height value of cell \((i, j)\) in the elevation map is denoted by \( h_{i,j} \), while the certainty value of this cell in the certainty map is denoted by \( c_{i,j} \). Using the above transformation, a range data is translated into a point \( (x_n, y_n, z_n) \). In order to register this point into the map at the associated location, the coordinate values \( x_n \) and \( y_n \) are converted to the grid indices \( i \) and \( j \), respectively. Then the certainty value \( c_{i,j} \) and height value \( h_{i,j} \) of the cell in the terrain map are updated. Specifically, \( c_{i,j} \) is updated by

\[ c_{i,j}^{t+1} = \begin{cases} c_{i,j}^t + a & \text{if } z_{i,j}^{t+1} - h_{i,j}^t > \Delta z_{\max} \text{ or } c_{i,j}^t = 0 \\ c_{i,j}^t & \text{otherwise} \end{cases} \] (6)

where \( a \geq 1 \) is the increment of the certainty value and it can be any positive integer. In this study the certainty value is represented by one byte and \( a=3 \) is used. As a result, all cells illuminated more than 84 times have the same certainty value. The height value \( h_{i,j} \) of the cell in the elevation map is updated by

\[ h_{i,j}^{t+1} = \begin{cases} z_{i,j}^{t+1} & \text{if } z_{i,j}^{t+1} > h_{i,j}^t \text{ or } c_{i,j}^t = 0 \\ h_{i,j}^t & \text{otherwise} \end{cases} \] (7)

\( c_{i,j} \) and \( h_{i,j} \) are initially zero and updated by (6) and (7), respectively. In every 13.3 milliseconds the 181 range measurements of the LRF are mapped into both maps. In (6), the condition \( |z_{i,j}^{t+1} - h_{i,j}^t| \leq \Delta z_{\max}' \) is called motion continuity constraint [19]. Every two consecutive measurements on a real object at the same location should satisfy this constraint. According to (6), cells occupied by real object are assigned continuous certainty increments and result in large certainty
values. Therefore, they can be identified easily. Mixed pixels and random noise, on the other hand, don’t satisfy the motion continuity constraint and thus result in small certainty values. In addition, they are spatially isolated in the elevation map. These characteristics can be used to filter out mixed pixels and random noises. Details of the filtering method are referred to [10].

2.3. Terrain Traversability Analysis

The task of TTA is to transform a terrain map into a traversability map by assigning a TI value to each cell in the terrain map. This process is divided into two steps: estimating terrain slopes and roughness, and computing the TI value for each cell.

A square terrain patch $P$ centering at the Robot Geometric Center (RGC) is formed in the terrain map. In this paper, $P$ is chosen in such a way that it exactly envelops the robot regardless of the robot’s orientation. The number of data points in the patch is $N = (2L+1)\times(2L+1)$, where $2L+1$ is the number of cells each side length of $P$ has. We then fit a least-square plane to $P$. The normal to the least-square plane $n = (n_x, n_y, n_z)$ and the residual of the fit $\sigma$ are found by using the Singular Value Decomposition (SVD) method [20].

2.3.1. Scheme 1 for TTA

For this scheme, the slope of the terrain patch is estimated by

$$\varphi = \cos^{-1}(n_z),$$

and the TI value of cell $(i,j)$ is calculated using $\sigma$ and $\varphi$ as follows:

$$\tau_{i,j} = F_1\varphi + F_2\sigma / N$$

2.3.2. Scheme 2 for TTA

Scheme 1 has been implemented in a Segway Robotic Mobility Platform (SRMP) recently [9]. The disadvantage of using (9) is that it doesn’t reflect the fact that the level of difficulty for a robot to traverse a slope is related to the robot’s yaw angle. To account for this, an alternative method of computing TI is proposed. As the method computes a TI along a specific direction, the robot yaw angle $\psi$ is known. Considering that the robot is on the fitted plane whose normal vector is $n$, the roll and pitch of the robot can be estimated by

$$\begin{align*}
\phi &= \sin^{-1}(n_z \cos\psi + n_x \sin\psi) \\
\theta &= \sin^{-1}((n_z \sin\psi - n_x \cos\psi) / \cos\phi)
\end{align*}$$

(10)

The TI value of cell $(i,j)$ is then calculated by

$$\tau_{i,j} = F_1 \max(|\phi|, |\theta|) + F_2\sigma / N.$$ 

(11)

Since the first term depends upon the yaw angle $\psi$, the use of the TI in (11) allows the robot to steer into a direction that minimizes $\tau_{i,j}$. This enable the robot to snake up/down a steep ramp. $F_1$ and $F_2$ in (9) and (11) are chosen by simulation runs in typical urban environments. In this paper, $F_1=3.7$ and $F_2=0.15$ are used and this set of parameters results in the roughness having a larger contribution than the slope.

Apparentely, the TTA process expands an obstacle’s boundaries in order to account for the dimensions of the robot. This allows us to treat the robot as a single point in planning the robot motion.

2.4. The PTI Histogram for Path Planning

The path planning algorithm determines the robot’s next heading direction and velocity based on the traversal property of the local terrain map surrounding the robot. In this work, the traversal property is described by the form of Polar Traversability Index (PTI).

In order to compute PTIs, a square-shaped local traversability map $S^*$ obtained by the above TTA method is formed at the RGC. There are $w_x \times w_y$ cells in $S^*$ and the RGC is at the center of $S^*$. Figure 7 is the traversability map obtained by applying the TTA method to a computer generated elevation map. A white grid in Fig. 7 represents a cell with zero TI while the other grids represent cells with nonzero TIs. A cell with nonzero TI in the traversability map generates an imaginary vector field, which exerts a virtual repulsive force on the robot and pushes it away from the cell. This is called
the “traversability field” that is defined in terms of a direction and a magnitude computed in the same way as in [21].

The direction of the traversability field generated by cell \((i, j)\) is given by
\[
\epsilon_{i,j} = \tan^{-1} \frac{y_j - y_0}{x_i - x_0},
\]
and the magnitude of the traversability field is
\[
m_{i,j} = \tau_{i,j}^2(d_{\text{max}} - d_{i,j}),
\]
where \(x_i\) and \(y_j\) are the coordinates of cell \((i, j)\); \(x_0\) and \(y_0\) are the present coordinates of the RGC; \(\tau_{i,j}\) is the TI value of cell \((i, j)\); \(d_{\text{max}}\) is the distance between the vertices (the four farthest cells) of \(S^*\) and the RGC; and \(d_{i,j}\) is the distance between cell \((i, j)\) and the RGC. In (13), \(\tau_{i,j}\) is squared such that the impact of a cell with a small/large TI value is diminished/magnified. Obviously, \(m_{i,j}\) decreases with increasing \(d_{i,j}\) and it is zero for each of the four vertices of \(S^*\).

In order to evaluate the overall difficulty of traversing along a direction, the magnitude of the traversability field produced by all cells in the same direction should be summed up. For this reason, \(S^*\) is divided into \(n\) sectors, each of which has an angular resolution of \(\xi=360/\pi\). Each sector \(k\), for \(k=0,\cdots,n-1\), has a discrete angle \(\rho=k\xi\). Cell \((i, j)\) in \(S^*\) is assigned to the \(k\)th sector according to
\[
k = \text{int}(\epsilon_{i,j}/\xi).
\]
For sector \(k\), the magnitude of the traversability field produced by all cells in this sector is calculated by
\[
h_k = \sum_{i,j} m_{i,j},
\]
\(h_k\) is the distance-weighted sum of the squared TIs of all cells in sector \(k\). Apparently, a larger value of \(h_k\) means that direction \(\rho=k\xi\) is harder for the robot to move along with. In other words, \(h_k\) is a TI representing the overall difficulty of traversing the terrain in the corresponding direction. Therefore, it is called the Polar Traversability Index (PTI). In this paper \(\xi=5^\circ\), i.e., there are 72 sectors. The PTIs are represented in a form of histogram. The path planning method then clusters the histogram into candidate valleys (comprising consecutive sectors with PTIs below a threshold) and hills (comprising consecutive sectors with PTIs above the threshold). One of the candidate valleys that is the closest to the target direction and incurs the smallest heading change is selected as the winning valley. Sector \(k_h\) located inside the winning valley is then chosen as the robot’s next heading direction. The details on how the path planning method determines the robot’s next heading direction are referred to [16].

Since the controller of the robot accepts steering rate and velocity as the control commands, the robot’s next heading direction is used to compute the steering rate as follow:
\[
\psi' = \left(k_s(t+1) - k_s(t)\right)/\Delta T,
\]
where \(k_s(t)\) and \(k_s(t+1)\) are the robot’s heading directions at time steps \(t\) and \(t+1\), respectively; and \(\Delta T=93.3\, \text{ms}\) is the control interval that equals to the time for the navigation system to acquire 7 sets of laser data. When the robot heads on an obstacle, a reduction in velocity is required since this gives the robot more time for ON. It also allows the robot to obtain more range data on the potential hazard. For this reason, the robot velocity is calculated by
\[
v' = v_{\text{max}}\left(1 - \min(h_h, h_a)/h_a\right),
\]
where \(v_{\text{max}}\) is the robot’s maximum velocity, \(h_h\) is the PTI in the robot’s current heading direction, and \(h_a\) is a constant that is empirically chosen to produce a sufficient reduction in velocity. A reduction in velocity is also required when the robot approaches the target. In this work, the velocity is further reduced by
\[ \dot{v} = v \min(d_i,d_m)/d_m, \]  

where \( d_i \) is the distance between the target and the RGC, and \( d_m \) is a constant (1.5 m in this work). Note that \( \dot{\psi} \) and \( \dot{V} \) are actually the projection of the steering rate \( \dot{\psi} \) and velocity \( V \) on \( x_ny_n \) plane. They are converted to \( \dot{\psi} \) and \( V \). A low pass filter is used to smooth the steering and velocity commands before being sent to the motion controller. In the system, the smoothed commands is computed by

\[ \Theta_s(t+1) = \lambda \Theta(t) + (1-\lambda)\Theta(t+1), \]

where \( \Theta_s(t) = \left( \dot{V}_s(t), \dot{\psi}_s(t) \right)^T \) and \( \Theta(t) = \left( \dot{V}(t), \dot{\psi}(t) \right)^T \). \( V_s(t) \) and \( \dot{\psi}_s(t) \) are the smoothed velocity and steering commands that are actually applied to the robot. A larger value of \( \lambda \) produces smoother control commands but a slower system response. \( \lambda = 0.75 \) is used in the system to provide a relatively smooth control.

### 2.5. Experimental Results

The complete terrain mapping and navigation system was tested with the pioneer robot on an outdoor terrain outside the Fribourgh Hall at the University of Arkansas at Little Rock. As shown in Fig. 8, the terrain is uneven and contains curbs and a ramp. A number of cardboard boxes were used to construct obstacle courses. A navigation task was specified to the robot by giving the robot its target location. The system was initialized using the robot’s current position as its start point (the robot pose was set to zero).

The terrain mapping and navigation system was implemented on the onboard computer that runs RTLinux as the OS. The LIDAR’s angular resolution was set to 1°. Each laser scan takes 13.3 milliseconds and contains 181 range data that covers a 180° field of view. The Real-time Data Acquisition (RDA) section of the terrain mapping module (Fig. 5) resides in the real-time Linux kernel as a real-time thread. The RDA thread stores the laser, gyroscopic and wheel-odometry data in a circular buffer. Each time a set of 7 laser scans and the associated gyroscopic and odometry data are fetched and sent into a shared memory, from which the Map Building & Filtering (MBF) section reads the data set, compute the robot pose, and registered the laser data into a terrain map. (The MBF also sends the data set to the remote laptop computer for visualization of navigation.) The TTA module then obtains the local terrain map (55×55 cells) and transforms it into a traversability map. Then the path planning module converts the traversability map into a PTI histogram, determines the motion commands, and sends the commands to the robot. The above process repeats every 93.3 ms (7×13.3 ms) until the robot reaches the target.

---

**Fig. 8.** Experiment on an outdoor terrain outside the Fribourgh Hall at UALR: the robot ran from \( S_3 \) to \( G_3 \). The red curve represents the robot trajectory. The robot in (b) is just for illustration. It is actually a point in the traversability map.
An RTLinux driver for the high speed RS422 serial card was written in C code for communication with the LIDAR and gyro. The serial driver runs in the real-time Linux kernel. The RDA thread reads data from the serial driver and communicates with the MBF through the shared memory. The MBF is a process in the regular Linux kernel which is treated as a process in the RTLinux kernel. To maintain data correctness in the inter-process communication, a data flag is created in the shared memory to indicate the state of data set, i.e., ready for the MBF to read or empty for the RDA to write. The RDA’s circular buffer may accommodate 10 laser scans and the associated gyroscopic and odometry data. This arrangement will prevent data loss if the shared memory is temporarily not available for RDA writing.

Figure 8 shows one of the autonomous runs from S3 to G3 on the terrain. S3 has a higher elevation than G3. The area immediately before the ramp is a valley that is lower than the neighboring area, including S3 and G3. The 3D terrain map (rendered on the remote laptop computer) is depicted in Fig 8b in a form of color-coded point cloud where the blue points represent data with negative elevation (below S3). The black spots are unperceived areas (no range data). The elevation values increase when the color changes from blue to purple and from purple to light yellow. (If displayed in black and white, a brighter point has a bigger elevation value.) Figure 8c displays the elevation map at the point where the robot was avoiding the first obstacle it encountered. Figure 8e shows the elevation map at the time when the robot was avoiding the curb and aligning itself with the ramp. Figure 8f shows the elevation map after the robot successfully traversed the ramp. In the elevation maps, the unperceived areas (e.g., area c) of the lower terrain surface look like obstacles because their Z coordinate values (zero) are bigger than that of the surrounding terrain surfaces whose elevation values are negative.

Data collected from all experimental runs has demonstrated that the roll, pitch, and yaw angles, and the velocity of the robot changed smoothly. This means that the navigation method produced smooth motion commands for the robot. Figure 9 depicts the robot’s roll, pitch and yaw angles during the experimental run from S3 to G3 in Fig. 8. The roll and pitch angles are limited in (-1°, 8°) and (-3°, 7°), respectively. This means that the navigator guided the robot to traverse moderate terrain. Both roll and pitch angles changed smoothly during the entire navigation task except for the points around ‘S’ in Fig. 9a and Fig. 9b where relatively larger changes in roll and pitch are observed. This is because the robot ran over the left curb at the upper end of the ramp (labeled by A in Fig. 8a). The curb is low-profiled and considered traversable. However, it generated relatively larger changes in the robot’s roll and pitch angles. As shown in Fig. 9c, the robot’s yaw angle changed very smoothly during the whole process. This indicates that the navigation method generated smooth steering commands for the navigation of the robot.

3. FLC-based Systems

As indicated earlier, a FLC is ideal for the autonomy of a small robot or robotic device. A robotic device is a portable robotic system with navigational functions such as positioning, obstacle detection, scene recognition, etc. A typical example is a computer vision enhanced white cane for the visually impaired [22]. For indoor navigation, the representative functions include feature-based robot/device pose estimation, object detection and scene recognition. Feature-based Pose Estimation (FPE) involved a procedure of visual and range data co-processing, called VR-odometry, while the key to object detection and scene recognition is 3D data segmentation.

3.1. VR-Odometry

3.1.1. Operating Principle

The proposed FPE method is to perform feature detection and matching in the SR4000’s intensity images and use the matched features’ range data to determine the pose change in two consecutive image frames. The method is termed VR-odometry as it uses both visual and range data to determine the pose change.
In this work, the SIFT (Scale-Invariant Feature Transform) feature detector [23] is used to extract features in an intensity image and match them to the SIFT features in the next intensity image. As the 3D coordinates of the matched features are known from the range data, the feature-matching process in fact solves the 3D points correspondence (data association) problem at each two sampling steps, and result in two 3D data sets, \( \{ p_i \} \) and \( \{ p'_i \} \); \( i = 1, \ldots, N \). \( N \) is the number of matched SIFT features in the two image frames. The pose estimation problem is then to find a rotation and translation matrices, \( R \) and \( T \), that minimize the error residual

\[
e^2 = \sum_{i=1}^{N} \left\| p_i - R p_i - T \right\|^2.
\]  

(20)

This least-square data sets fitting problem can be solved by the Singular Value Decomposition (SVD) method in [24]. As feature-matching in intensity images may result in incorrect data association (outliers), a RANSAC (Random Sample Consensus) process is implemented to reject the outliers. The entire method is as follows:

1) Extract the SIFT features in two consecutive images, find the matched features, and locate the corresponding 3D data sets \( \{ p_i \} \) and \( \{ p'_i \} \).

2) Randomly select 4 associated points from the two data sets and form \( \{ p_k \} \) and \( \{ p'_k \} \); \( k = 1, \ldots, 4 \). Then find the least-square rotation and translation matrices (\( \hat{R}_i \) and \( \hat{T}_i \)) for \( \{ p_k \} \) and \( \{ p'_k \} \).

3) Project the entire data set \( \{ p_i \} \) onto \( \{ p'_i \} \) using the found transformation (\( \hat{R}_i \) and \( \hat{T}_i \)) and compute the error

\[
e_i^2 = \left\| \hat{p}_i - \hat{R}_i p_i - \hat{T}_i \right\|^2
\]

(21)

for each data-pair \( i = 1, \ldots, N \). A threshold \( \varepsilon \) is used to score \( S_i \) for this transformation: \( S_i \) is incremented once for each \( e_i^2 < \varepsilon \).

4) Step 2 and 3 repeat for a fixed number of iterations or until exhausting all combination of point set selections, whichever is smaller. The transformation with the highest score is recorded. The corresponding data sets \( \{ p_j \} \) and \( \{ p'_j \} \); \( j = 1, \ldots, S_i \), where each data-pair satisfy the threshold test in step 3, are selected and used to compute the maximum likelihood transformation estimate \( \hat{R} \) and \( \hat{T} \) by the SVD least-square fitting method. The PRD’s rotation can be computed from \( \hat{R} \) and its translation is determined by \( \hat{T} \). The pose change is thus determined.

3.1.2. Experimental Results

A rudimentary implementation of the VR-odometry has been performed in Matlab environment to validate the idea. The implementation uses SIFT feature descriptors as they are invariant to translations, rotations and scaling and robust to residual small distortions. Figure 10 illustrates how the VR-odometry finds the correct correspondences between features in two successive images. The images were captured when a human handheld the SR4000 and moved in a hallway. Figure 10a depicts the detected SIFT features. They were matched with the features in the next frame based on the SIFT features’ scales and orientations. Figure 10b shows the initial matches that exhibit some outliers (mismatches). The RANSAC process removed the outliers and the results are shown in Fig. 10c.

![Fig. 10. SIFT feature matching and RANSAC for outlier removal: (a) SIFT features of an image frame (each circle and the straight line represent the scale and orientation of the SIFT feature located at the center of the circle.); (b) Initial matched features in two consecutive image frames; (c) Matched features after RANSAC. Intensity images were captured in a hallway. The time interval between the right and left images in (b) is 100 milliseconds. Therefore, the objects in the right image are closer to the sensor.](image)

A series of experiments were carried out in an office environment with objects ranging 1.5-4.5 meters from the SR4000. The sensor was installed on a pan-tilt unit. Experimental runs, with various combinations of pitch, yaw rotation and \( X, Y \) translations, were performed to quantify the measurement accuracy and repeatability. In all experiments, roll \( \phi \) and \( Z \)
translation are always zero. 1000 images for each pose were taken to compute pose changes and the error statistics. It is noted that in all experiment we use raw sensor data for pose estimation.

**Distribution of pose measurement errors:** two experiments were carried out to examine the distribution of pose measurement errors. In the first experiment, the SR4000’s pose change was zero. The experiment result demonstrates that the measurement error of pose ($\phi$, $\theta$, $\psi$, $x$, $y$, $z$) is zero-mean Gaussian with standard deviation ($0.1\, ^\circ$, $0.2\, ^\circ$, $0.2\, ^\circ$, 7 mm, 3 mm, 6 mm). The result indicates that the VR-odometry’s inherent error is a white Gaussian noise. Also, the measurement accuracy (mean error) and repeatability (standard deviation) are very good, meaning the sensor’s noise has limited effect on the VR-odometry’s performance in pose estimation.

In the second experiment, the SR4000 had a combination of rotation ($\theta$=-5.9°, $\psi$=5.0°) and translation ($X$=80 mm, $Y$=130 mm). The distribution of the pose measurement error follows a normal distribution whose mean and standard deviation are ($-0.1\, ^\circ$, $0.2\, ^\circ$, $-0.3\, ^\circ$, 8 mm, $-2\, mm$, 11 mm) and (0.5°, 0.4°, 0.4°, 13 mm, 5 mm, 11 mm), respectively. Compared with the first experiment, the results exhibit a bias in the mean error and relatively larger standard deviation.

Both experiments show a decent accuracy in orientation measurement. Considering that the SR4000’s average angular resolution is about 0.25°, the accuracy of orientation measurement is reasonable. The VR-odometry also demonstrates a very good accuracy and repeatability in measuring motion along $Y$ axis due to the sensor’s high accuracy and repeatability in depth measurement. The sensor’s resolution in measuring motion along $X$ and $Z$ axis drops proportionally with the depth ($Y$) value. At $Y$=5 meters, the resolution is about 23 mm. The VR-odometry’s measurement accuracy and repeatability in $X$ and $Z$ axes seem reasonable. However, a proper camera calibration and data filtering may improve the performance.

**Accuracy and repeatability of pose measurements:** Experiments were performed to inspect the pose estimation performance for each individual motion. Due to the constraint of test facility, only experiments with pitch and yaw rotations and $X$ and $Y$ translations were performed. In the first experiment, the sensor underwent a pitch rotation in the range [3°, 21°] (increment: 3°/step). 1000 images were captured before and after each pitch rotation for computing the pose change. It is observed that the accuracy (mean error) and the repeatability of the pitch measurement are quite decent for a pitch rotation in the range [3°, 18°]: the mean errors are in the range [0°, 0.21°] and the standard deviations are in [0.53°, 0.86°]. The relative mean errors are between 0.0% and 2.0%.

The second experiment is to test the VR-odometry’s performance in measuring yaw rotation in the range [-21°, -3°]. The mean errors are in the range [-0.36°, 0.03°] and the standard deviations are in [0.39°, 0.88°]. The relative mean errors are between -1.0% and 2.7%.

Table 2 shows the statistics of the roll, pitch and yaw measurements of both experiments. It can be observed that the measurements are accurate if the pitch/yaw rotation is within ±18°, a large image motion between image frames compared to the camera’s field of view. The mean errors are mostly within the camera’s angular resolution (~0.25°). There are a few exceptions that the mean errors go beyond ±0.25° (the worst case: -0.46°). This can be improved if proper data filtering and sensor calibration will be performed. For measurement repeatability, the standard deviations of the roll measurements are consistently small. This is probably because that there was no roll rotation in the experiments. The repeatability of pitch and yaw measurements need to be improved in future work.

The third experiment is to test the measurements of translation when the sensor moves along $X$ or $Y$ directions. A step size of 305 mm was used in the experiments. We found that the VR-odometry did not produce satisfactory results if the translation is bigger than 610 mm. (This suggests that a smaller step size should be used in the future study.) The results are tabulated in Table 3, from which we can observe that the VR-odometry has highly consistent accuracy and repeatability in measuring movement along $Y$ axis (depth). This is attributed to the sensor’s consistent measurement accuracy in $Y$ axis (±1 cm), an apparent advantage over the stereovision-based approach. However, it has relatively larger mean errors and/or standard deviations in $X/Z$ measurements (compared with the camera’s resolution). These need to be improved in our future work.

The results in Tables 2 and 3 were computed using raw sensor data. Proper data filtering and camera calibration will improve the measurement accuracy and repeatability in the future work. It should be noted that in some cases the VR-odometry did not find a solution to the pose estimation. The error states were recorded and the data were discarded, meaning that the samples for computing some of the statistics in Tables 2 and 3 may be slightly smaller than 1000. Some of the failures were resulted because the RANSAC process did not find a sufficient number matched features (possibly due to the use of an overly small threshold $\epsilon$). We have not yet looked into the cause of the other failure cases.

In our current implementation, we use the SIFT feature descriptors and the RANSAC method for the sake of the accuracy and reliability of pose estimation. Both approaches are computationally expensive. In our future work, we will develop a more efficient method with real-time performance. This may be achieved through the following efforts: (1) investigate other feature descriptors and adopt one with less computational cost such as the SURF (Speeded Up Robust Features) [25] for the VR-odometry; and (2) use spatial invariants between the detected features’ 3D points (e.g., distances between the 3D points) for outlier rejection (or inlier detection) to accelerate or remove the need of the RANSAC process. In term of improving the
method’s accuracy and repeatability, we will calibrate the camera, develop a filtering method to filter out data with low confidence levels and develop feature sampling method to select features with more accurate depth values.

3.2. 3D Data Segmentation

Segmentation of an indoor scene into planar surfaces and grouping them into structures is the essential step for object/obstacle detection and scene recognition. In this work, a range data frame is represented as a tri-band color image where each pixel’s RGB values represent the x, y components of the surface normal and the depth information (Y value) of the corresponding point. This scheme encodes a 3D point’s local geometric information (surface normal) and global information (location) into a pixel in the color image. This RGB image is called an Enhanced Range Image (ERI) since it enhances the edges and surfaces of objects in the original range image.

With this image enhancement, one may simply apply the Normalized Cuts (NC) method to the ERI and segment the range data. However, this process is computationally expensive as the number of pixels in the ERI is huge. To reduce the computational cost, the Mean Shift method is first used to segment the ERI into a number of regions, called Super-Pixels (SPs), each of which contains a group of homogeneous pixels. A graph is then constructed by treating each SP as a node and the edge weight, which is a measure of similarity between two nodes $V_i$ and $V_j$, is computed by

$$\theta_{ij} = \cos^{-1}(\vec{N}_i \cdot \vec{N}_j).$$

They are considered as parallel if $\theta_{ij}$ is sufficiently small, i.e., $\theta_{ij} < \varepsilon$.

B. Graph Construction and Partitioning

A graph $G = (V, E)$ is constructed by treating each planar SP as a node. The edge weight, which is a measure of similarity between two nodes $V_i$ and $V_j$, is calculated by

\begin{table}
\centering
\caption{Measurement accuracy of rotation}
\begin{tabular}{|c|c|c|c|c|}
\hline
MV: $\mu$ & Roll $\phi$ (°) & Pitch $\theta$ (°) & Yaw $\psi$ (°) \\
TV: $\sigma$ & Mean error & Mean error & Mean error \\
\hline
(0, 3, 0) & (0.06, 0.27) & (3.06, 0.53) & (-0.16, 0.57) \\
(0, 6, 0) & (0.07, 0.16) & (6.00, 0.64) & (0.02, 0.66) \\
(0, 9, 0) & (0.01, 0.21) & (9.03, 0.59) & (0.24, 0.65) \\
(0, 12, 0) & (0.22, 0.21) & (12.14, 0.80) & (0.24, 0.69) \\
(0, 15, 0) & (0.11, 0.33) & (15.21, 0.63) & (-0.12, 0.91) \\
(0, 18, 0) & (0.00, 0.31) & (18.04, 0.77) & (-0.46, 0.80) \\
(0, 21, 0) & (0.29, 0.44) & (22.09, 1.54) & (0.14, 1.06) \\
\hline
(0, -3) & (-0.03, 0.31) & (0.10, 0.52) & (-2.97, 0.39) \\
(0, -6) & (-0.03, 0.19) & (-0.02, 0.50) & (-6.15, 0.45) \\
(0, -9) & (0.02, 0.25) & (0.20, 0.60) & (-9.24, 0.61) \\
(0, -12) & (-0.02, 0.26) & (0.15, 0.66) & (-12.22, 0.64) \\
(0, -15) & (-0.09, 0.32) & (0.36, 0.68) & (-15.18, 0.76) \\
(0, -18) & (-0.13, 0.30) & (0.15, 0.69) & (-18.33, 0.78) \\
(0, -21) & (0.04, 0.39) & (0.20, 0.77) & (-21.36, 0.88) \\
\hline
\end{tabular}
\caption{Measurement accuracy of translation}
\begin{tabular}{|c|c|c|c|}
\hline
MV: $\mu$ & Roll $\phi$ (°) & Pitch $\theta$ (°) & Yaw $\psi$ (°) \\
TV: $\sigma$ & Mean error & Mean error & Mean error \\
\hline
(0, 3, 0) & (0.06, 0.27) & (3.06, 0.53) & (-0.16, 0.57) \\
(0, 6, 0) & (0.07, 0.16) & (6.00, 0.64) & (0.02, 0.66) \\
(0, 9, 0) & (0.01, 0.21) & (9.03, 0.59) & (0.24, 0.65) \\
(0, 12, 0) & (0.22, 0.21) & (12.14, 0.80) & (0.24, 0.69) \\
(0, 15, 0) & (0.11, 0.33) & (15.21, 0.63) & (-0.12, 0.91) \\
(0, 18, 0) & (0.00, 0.31) & (18.04, 0.77) & (-0.46, 0.80) \\
(0, 21, 0) & (0.29, 0.44) & (22.09, 1.54) & (0.14, 1.06) \\
\hline
(0, -3) & (-0.03, 0.31) & (0.10, 0.52) & (-2.97, 0.39) \\
(0, -6) & (-0.03, 0.19) & (-0.02, 0.50) & (-6.15, 0.45) \\
(0, -9) & (0.02, 0.25) & (0.20, 0.60) & (-9.24, 0.61) \\
(0, -12) & (-0.02, 0.26) & (0.15, 0.66) & (-12.22, 0.64) \\
(0, -15) & (-0.09, 0.32) & (0.36, 0.68) & (-15.18, 0.76) \\
(0, -18) & (-0.13, 0.30) & (0.15, 0.69) & (-18.33, 0.78) \\
(0, -21) & (0.04, 0.39) & (0.20, 0.77) & (-21.36, 0.88) \\
\hline
\end{tabular}
MV: Measured Values, TV: True Values, $\mu$: mean error, $\sigma$: standard deviation. Camera’s angular resolution: pitch: 0.25°, yaw: 0.24°. 1000 samples were used to compute the statistics.
where \( \sigma_{p} \) and \( \sigma_{j} \) are positive constants, \( F(p) \) is the color vector of node \( p \) for \( p = i, j \), and \( d_{ij} \) is the Euclidean distance between the LSPs of SPs \( i \) and \( j \) (i.e., nodes \( i \) and \( j \)). If \( \theta_{ij} < \epsilon \) is satisfied, then \( d_{ij} \) is the distance from the centroid of the data in SP \( i \) to the LSP of the data in SP \( j \). Otherwise, \( d_{ij} = 0 \). Two SPs are considered as neighbors if they have at least two neighboring pixels in the ERI space. The weight computation in (6) takes into account the statistics of 3D data points. This may result in a better segmentation result. We then apply the NC algorithm to the graph and cluster the SPs into a set of segments.

C. Labeling and Merging of Planar Segments

In the last step of the ENC method, the segments in \( S \) are labeled as planar and non-planar according to 3.2.1.A. Two neighboring planar segments, \( s_{i} \) and \( s_{j} \), are merged if \( \theta_{ij} \leq \epsilon \) and \( d_{ij} \) is less than a threshold.

The ENC plane extraction method is summarized as follows:

a) Construct the ERI from the range data and apply the MS algorithm to obtain a number of SPs.

b) Obtain planar SPs; for \( i = 1, \ldots, m \) from the resulted SPs according to III.B.

c) Construct a graph \( G \) on SPs; for \( i = 1, \ldots, m \) and compute the similarity matrix \( W \) of order \( n \times n \) by (23).

d) Apply the NC algorithm to graph \( G \) with \( W \) as the input and obtain \( N \) segments, \( s_{i} \) for \( i = 1, \ldots, N \), each of which contains a number of SPs. Each segment \( r \) in \( s_{i} \) is further classified to form a set of planar segments \( P = \{ p_{1}, p_{2}, \ldots, p_{t} \} \); \( t \leq N \).

e) Construct a binary matrix \( K = \{ k_{ij} \} \) to record the neighborhood relationship among segments in \( P \), where

\[
k_{ij} = \begin{cases} 1 & \text{if } p_{i} \text{ and } p_{j} \text{ are neighbors and } \theta_{ij} \leq \epsilon, d_{ij} \leq \tau \\ 0 & \text{otherwise} \end{cases}
\]  

(24)

It is noted that a segment is treated as its own neighbor. Therefore, \( k_{ij} = 1 \).

f) In the final step, the entire planar surfaces are extracted by merging those segments whose \( k \) values equal zero. This is done by using the depth first search algorithm.

3.2.2. Experimental Results

The segmentation method has been validated with range data captured from the SR4000 in a number of representative indoor environments. In all experiments, a pre-specified segment number \( N=75 \) is used for the NC method. For the results shown in this section, an unlabeled segment is represented in black and a labeled segment (a planar segment) with a random non-black color. The labeled segments are then overlaid on the ERI.

To justify the use of the NC algorithm, the proposed method without the NC component is also run on each data. This method is referred to as Mean-Shift Dominated (MSD) segmentation method. The segmentation performances of the two methods are compared using the following Segmentation Quality Index (SQI):

\[
SQI = (R - A)/A ,
\]  

(24)

where \( A \) is the actual number of planes (hand labeled in the experiments) and \( R \) is the resulting number of planes by the segmentation method. The sign of SQI indicates over-segmentation (positive sign) or under-segmentation (negative sign) and the magnitude of SQI represents the segmentation quality. A SQI value closer to 0 indicates a better segmentation result.

The first experiment is to segment the range data of a hallway (Fig. 11a). Figure 11b and Fig 11c display the range image and the ERI with the 4 prominent planes manually labeled. After applying the MS algorithm to the ERI, we obtained 270 SPs as shown in Fig. 11d. The use of SPs causes a reduction of node number from 25344 to 270. As the edge number of a n-node graph is \( n(n-1)/2 \), the number of edge-weight computations is reduced from 321146946 to 36315, about \( 10^5 \) times smaller. This computational reduction will be referred to as Computational Reduction Factor (CRF) further on. The initial grouping of the SPs by the NC method is shown in Fig. 11e. The extracted planar segments by the ENC method (after merging homogenous planar segments) are shown in Fig. 11f whose corresponding labeled point-cloud is depicted in Fig. 11g. The result of MSD is shown in Fig. 11h.

From a qualitative perspective, the MSD results in fragmented walls (in planes labeled as 2 and 3) whereas the ENC method is able to extract the walls in their entirety. There are some misclassifications at the intersection of the front wall and
floor in Fig. 11f. This is most likely due to the use of a small \( N \). However it should be noted that their impact on navigating a robot may be ignored as they occur at a faraway location from the sensor.

The 2nd experiment was carried out to examine the method’s performance on a scene with a stairway as shown in Fig. 12a. As we can see, the ENC method extracts most of the horizontal (tread) and vertical (riser) surfaces almost in their entirety.

The SQIs of the two methods in the 1st experiment are SQI MSD = 6 and SQI ENC = 1 and that of the 2nd experiment are SQI MSD = 4.9 and SQI ENC = 0.4. These indicate that the ENC method has a much better segmentation performance. We ran the USF segmentation method (as described in [27]) on the above two cases and found very large number of unclassified pixels, indicating a useless segmentation result, in each case.

Additional experiments with various configurations of objects were performed and the results are similar. This demonstrates that the inclusion of the NC method can substantially improve segmentation result. All of our experiments have also indicated that the proposed method performs much better than the USF method in the cases of noisy range data of the SR4000. More detailed information on the comparative study and the runtime performance of the ENC method can be found in [28].

4. Discussions and Conclusions

This Chapter introduces the Data processing methods for two typical robotic perception systems—LIDAR-based and FLC-Based systems. The LIDAR-based system employs a 2D LIDAR and utilizes the robot motion to perform in-motion mapping of the environment. Since each of the LIDAR’s data frame only contains 181 range data, accurate robot pose information is needed to register the laser data frames captured at different locations over time. The robot pose information is computed from the data of a 3D gyro and wheel-odometry. While the mapping method is cost-effective compared with a 3D LIDAR based system, it lacks reliability in a dynamic environment, i.e., an environment with moving object(s). For the navigation method, the use of a 2½D map in terrain traversability analysis incurs a disadvantage that the system cannot handle 3D objects (e.g. overhang object) with accuracy. A point-cloud processing capability will enable true 3D navigation capability. For the FLC-based system, the VR-odometry has good accuracy in estimating pose change between two image frames. This accuracy may be improved with refined VR-odometry method. For estimating long duration pose change, the VR-odometry output has to be integrated over time. This is usually achieved by using a state filter to track the robot pose. The ENC plane extraction method segments 3D point-cloud into planar surfaces with connectivity information. These
information may be used to group neighboring surfaces into objects for object/obstacle detection and scene recognition that may support robot localization, symbolic map-building and autonomous navigation.

Acknowledgment

The work on the LIDAR-based system was funded by Arkansas Space Grant Consortium under grant number UALR16800, a NASA EPSCOR Planning and Preparatory Award, and a matching fund from the Office of Research and Sponsored Programs of the University of Arkansas at Little Rock. The work on the FLC-based system was funded by the National Science Foundation (IIS-1017672), NASA (NNX09A072A), and Office of Naval Research Summer Faculty Research Fellow program. The author would like to thank GuruPrasad Hegde for his work in validating the data segmentation method. The development of the VR-odometry and the data collection were carried out at the Space and Naval Warfare Systems Center Pacific. The author would like to thank Michael Bruch for hosting the research.

References


