Fast Obstacle Detection for Monocular Autonomous Mobile Robots

Naoshi Kaneko *, Takeshi Yoshida **, and Kazuhiro Sumi **

Abstract: This paper proposes a monocular vision based obstacle detection algorithm for autonomous mobile robots. Our main algorithm consists of two stages. In the first stage, we use an inverse perspective mapping (IPM) based method for detecting small portions of an obstacle in the input image. In the second stage, we perform image abstraction and geodesic distance computation for segmenting the obstacle. We use the simple linear iterative clustering (SLIC) super-pixel algorithm for decomposing the image into basic elements that preserve relevant structure, but abstract undesirable detail. The source superpixel for geodesic distance computation is selected according to semi-local texture features. Then we compute the obstacle score for accurate segmentation. Experimental results have shown that our proposed method achieves accuracy comparable to the state-of-the-art method, with more than 7 times faster computation.

Key Words: segmentation, obstacle detection, mobile robots, inverse perspective mapping, geodesic distance.

1. Introduction

In recent years, consumer autonomous mobile robots such as robotic vacuum cleaners have become very popular. Since such robots often need to travel through cluttered domestic space, they must detect obstacles (furniture, household appliances, walls, etc.) to avoid them. Detecting obstacles around moving robots is one of the challenging and important problems in mobile robot navigation. Two well-known obstacle detection strategies for such robots are the subsumption architecture [1] and the simultaneous localization and mapping (SLAM) based approach [2]. Both strategies have a long research history in the robotics community.

Most commercial mobile robots [3]–[5] use the former strategy, which depends on the collision between the robot and obstacles. The robots employing this architecture detect obstacles by collision sensors mounted on the front bumper. In addition to the main sensor, auxiliary infrared/ultrasonic sensors are often used. These auxiliary sensors reduce the impact of collisions by slowing down the robot when it approaches the obstacle. Although this strategy reliably finds most obstacles, it is not always ideal for domestic environments since collisions possibly cause damage to furniture or household goods. The auxiliary sensors often miss the obstacles because of their narrow field of view and low angular resolutions.

Fewer commercial robots use the latter strategy [6],[7]. This strategy uses laser rangefinders for calculating the distance between the robot and the surrounding objects to generate an obstacle map around the robot. These robots employ a simple single laser rangefinder because of cost and power restrictions. The advantage of this strategy is that the robot can avoid obstacles before it physically collides with them. However, it has three major shortcomings. First, the map construction is time consuming since the rangefinder can only scan a single line at one time. The robot needs to rotate repeatedly for scanning the obstacles in travel direction. Second, it cannot detect the obstacles lower than the robot height since the rangefinder’s laser radiates parallel to the ground. Third, rangefinders are comparatively costly and require more electric power than other mounted sensors.

In contrast, vision based systems provide natural and powerful information of the environment at a high frame rate with a wide field of view. We believe that vision based systems could become a reliable complement of the above strategies. Various obstacle detection algorithms for different types of vision sensors have been proposed [8]–[11]. Some of the works employ monocular camera, stereo/multi camera, or depth camera systems for detection. Taking into account the comparatively low-power platform of small mobile robots, we choose a monocular camera as the input device. Monocular cameras are inexpensive and light weight, have a high frame rate, and are a computationally low cost device which is suitable for small mobile robots. However, monocular vision-based obstacle detection algorithms for such robots have to overcome two specific hardware limitations: low-mounted cameras and low-power computational resources.

This paper proposes a monocular vision based algorithm for fast and accurate obstacle detection designed for dealing with these limitations. Our main algorithm consists of two steps: (1) inverse perspective mapping (IPM) based obstacle region detection and (2) geodesic distance based obstacle segmentation. In the first step, we detect small portions of obstacles using perspective mapping between two consecutive frames. Then in the second step, we perform a geodesic distance transform over the image for distinguishing obstacles from the background. The obstacle region detected in the first step is used as a seed for the geodesic distance transform.

This detect-then-segment scheme is inspired by the state-of-the-art work of Lee et al. [12]. Compared to the method of Lee et al., our algorithm achieves much faster computation with comparable detection accuracy. Since the accuracy and the
computation time is critical for obstacle avoidance, our algorithm is more practical than the conventional method especially on mobile robot platforms.

Figure 1 shows an example of our obstacle segmentation results. In Fig. 1, the black rectangle shows the region of interest (ROI) of our algorithm and the white region shows the detected obstacle pixels.

The rest of this paper is structured as follows. Section 2 introduces related work. Sections 3–6 describe our proposed algorithm. In Section 7, we show the experimental results. Finally, Section 8 concludes this paper.

2. Related Work

In this section, we review obstacle detection approaches that make use of vision-based sensors. Traditional works employ geometrical properties for efficient obstacle detection. Mallot et al. [13] discovered IPM can be used for efficient optical flow computation, and proposed a motion field based obstacle detection algorithm. Bertozzi et al. constructed the generic obstacle and lane detection (GOLD) system [14] which is a stereo vision system for obstacle and lane detection. The GOLD system uses the difference between two IPM images computed from left and right stereo images. The approach of Zhou and Li [15] uses normalized homography calculated by tracking feature points for ground plane detection. Simond and Parent [16] use IPM and super-homography for obstacle detection. They employed edge based feature and perspective effects for distinguishing obstacles from the ground plane. Although these classical approaches are efficient and well studied, their detection accuracy is not so high. Moreover, their obstacle representations are sparse, which is not suitable for accurate obstacle detection.

A number of researches which employ dense obstacle representation also have been conducted. The work of Ulrich and Nourbakhsh [17] is an appearance based segmentation method which uses the histograms in hue, saturation, and intensity (HSI) color space. It segments obstacles by comparing the color histograms of ground plane and image pixels. Li and Birchfield [18] combine thresholding, edge detection, and graph based segmentation for evaluating the likelihood of a horizontal wall-floor boundary. These appearance-based segmentation methods tend to fail when the floor appearance is close to the obstacle appearance.

Cui et al. proposed a floor segmentation method based on the direction of plane normals [19]. The plane normals are estimated by motion fields and homography based calculation. Then, regions in the image are iteratively merged into the ground plane segment if the direction of the estimated plane normal for a region is close to the direction of the ground plane. Jia et al. proposed a real-time obstacle detection algorithm for autonomous vehicles [8]. They differentiate obstacles from the ground by motion features calculated using feature point tracking. The method of Kumar et al. [20] formulates the obstacle detection problem as a Markov random field (MRF). It combines superpixel based rough segmentation, line segment features, and optical flow based homography for reliable segmentation. Although these methods combined appearance and geometrical models, they require robust feature point tracking. Generally, the camera setting of small mobile robots is close to the ground. This setting reduces the amount of disparities and makes robust point tracking difficult.

Lee et al. [12] proposed a monocular vision based approach for small robots using IPM based coarse detection and MRF based fine segmentation. To the best of our knowledge, this is the first algorithm designed for “thin” robots like robotic vacuum cleaners. In their experiment, a camera was mounted at 63 mm above ground. In their coarse detection step, they compute a homography matrix for IPM using the odometry information of the robot. Their fine segmentation step calculates a probability density function (PDF) of texture and intensity features for the MRF based segmentation using graph cuts. Although this method achieves high segmentation accuracy, further improvement is required for more challenging cases. As reported in [12], their algorithm produces good results when the camera is located at about 60 mm or higher. However, for much lower camera settings, it produces relatively unstable results due to stronger perspective distortion of the warped image. The stronger distortion makes IPM boundary ambiguous and makes accurate coarse detection difficult. Figure 2 shows the segmentation results for a simple scene with different camera height settings. Considering the increasing demand for robots for small workspaces, much lower camera settings should be dealt with. Another important aspect of the algorithm is computation speed. Their fine segmentation algorithm requires high computational power, which may put a burden at practical situations.

With the evolution of deep neural networks, some modern research uses deep architecture for obstacle detection. Levi et al. [21] proposed obstacle detection and road segmentation algorithm using deep architecture. They combine convolutional neural networks and conditional random fields for reliable segmentation. Nguyen et al. [22] proposed an obstacle detection, recognition, and tracking algorithm using stereo disparity and deep neural networks. Although deep architecture based detection algorithms achieve high detection accuracy, they require a high performance graphics processing unit (GPU) platform for learning and prediction. Because of the hardware limitations of small mobile robots, deep architecture based methods are not suitable for this research.
The algorithm consists of the following two steps: we decide that there is no obstacle in the image. Our main algorithm consists of the following two steps:

1. **IPM based obstacle region detection.**
   - We compute the perspective mapping between two images acquired at different times. By subtracting mapped image from real image, we detect small portions of the obstacle.

2. **Geodesic distance based obstacle segmentation.**
   - We segment the image into superpixels using the modified version of simple linear iterative clustering (SLIC) algorithm [24],[25]. Then we calculate geodesic distances over the superpixels for distinguishing the obstacle from background.

If no obstacle region is detected in the first step, we skip the second step. We describe the details of our algorithm in the following sections.

4. **Decision for Triggering Main Algorithm**
   - Before starting the main algorithm, we check whether obstacle-like objects appear in front of the robot or not. Since most obstacles contain edges, we employ a simple edge based method. First, we detect edges in two consecutive images using the Canny edge detector [26]. Then, we take the XOR of the two edge images for checking the edge appearance changes. If the number of XOR edge pixels exceeds a certain threshold, the following main algorithm is triggered.

5. **IPM Based Obstacle Region Detection**
   - In the first step, we use IPM for detecting portions of the obstacle. IPM is a classical and reliable approach for detecting obstacles in front of autonomous robots [12]–[14],[16]. The robot acquires images \(C_t\) and \(C_{t+1}\) at different times \(t_1\) and \(t_2\). If the pixel \(c_{t_1}\) of image \(C_{t_1}\) belongs to the ground, it can be properly mapped to image \(C_{t_2}\) using a floor homography matrix \(H\). We calculate the floor homography matrix \(H\) by

   \[
   H = K \cdot (R - t \cdot n^T/d) \cdot K^{-1},
   \]

   where \(K\) is the camera intrinsic matrix, \(R\) is the relative camera rotation, \(t\) is the relative camera translation, \(n\) is the floor normal vector, and \(d\) is the camera-floor distance. We calculate \(R\) and \(t\) using the wheel odometry of the robot.

   - If there is no obstacle below the floor horizon line of \(C_{t_1}\), all pixels below the line are properly mapped to \(C_{t_2}\). However, if an obstacle appears below the line, the obstacle pixels are distortedly mapped to \(C_{t_2}\) because the obstacle does not belong to the floor plane surface. Therefore, by simply subtracting the mapped image from \(C_{t_2}\), small portions of obstacles can be detected. This calculation is given by

   \[
   c_{IPM} = \text{absdiff}(c_{t_1} - H \cdot c_{t_1}),
   \]

   where absdiff(·) is the absolute difference between two pixel values. After IPM detection, we use simple region growing for detecting the bounding box of detected pixels. Figure 5 shows an example of IPM detected pixels and their bounding box. In Fig. 5, the gray rectangle represents the ROI and the white rectangle represents the detected bounding box. This bounding box based representation of IPM detected pixels abstracts the obstacle region well and reduces the undesirable effects of ambiguous IPM boundaries.

6. **Geodesic Distance Based Obstacle Segmentation**

6.1 **Image Abstraction**
   - In the second step, we calculate the geodesic distance over the image for segmenting the obstacle from the background. First, we aim to decompose the image into basic elements that preserve relevant structure, but abstract undesirable detail.
Since mobile robots travel through domestic space, there may be dust on the floor. Although this dust is not an obstacle, the obstacle detection algorithm may be affected by its appearance. To reduce such effects, we segment the image into superpixels using the modified version of the SLIC algorithm [25]. The SLIC algorithm segments an image using $K$-means clustering in 5D space (3D for CIELab color space, 2D for pixel positions). The modified version of SLIC [25] uses a slightly different distance measure, which guarantees the compactness of superpixels. Figure 6 shows the superpixel segmentation result.

### 6.2 Graph Construction

Second, we construct a weighted graph structure $G = (V, E)$ for geodesic distance computation. Geodesic distance is used in various vision communities such as image segmentation or human pose estimation [27],[28]. Let $S = \{s\}$ represent the superpixels. We consider each superpixels as a graph vertex, i.e. $V = S$. We connect two vertices $x, y \in V$ if corresponding superpixels are adjoining. Edge weights are set to the mean color difference between $x$ and $y$ in CIELab space. The color difference $\Delta E^*$ is computed as

$$\Delta E^*(x, y) = \sqrt{(L_1^*-L_2^*)^2 + (a_1^*-a_2^*)^2 + (b_1^*-b_2^*)^2}. \quad (3)$$

Thus, the edge weight of edge $e \in E$ is given by

$$w(e) = \Delta E^*(x, y). \quad (4)$$

Geodesic distance $d_G(x, y)$ between graph vertices $x$ and $y$ is given by

$$d_G(x, y) = \sum_{e \in SP(x, y)} w(e), \quad (5)$$

where $SP(x, y)$ contains all edges along the shortest path between $x$ and $y$. Given a single source vertex, the shortest paths to all other points in the graph can be computed efficiently using Dijkstra’s algorithm.

### 6.3 Source Selection

To compute geodesic distances, we need to choose a source vertex that represents obstacle appearances well. Here, we use the obstacle bounding boxes detected in Section 5 for source selection. First, we compute semi-local geometry based texture feature described in [29] over the ROI. The texture feature $F$ is defined as

$$F = \exp \left( -\frac{|\det(g_{xy})|}{\sigma^2} \right), \quad (6)$$

where $\det(g_{xy})$ is the determinant of the metric tensor of the square patch around the pixel. $\sigma^2$ is a scale parameter (please refer to [29] for details). We use a $6 \times 6$ square patch in our implementation. As shown in Fig. 7, this texture feature describes the appearance difference between the floor and the obstacle well. Then, we find the source bounding box $B$ by

$$B = \arg\max_{m \in M} \sum_{e \in m} F(e), \quad (7)$$

where $M$ is the set of IPM detected bounding boxes and $F(e)$ denotes the texture feature of pixel $e$. Figure 8 shows the selected source bounding box (white rectangle) and other bounding boxes (black rectangles).

Finally, we find the superpixel $x_O$ from the inside of the bounding box $B$ which has minimum $\Delta E^*$ with the mean color over the bounding box. In addition to that, we also find the source superpixel $x_F$ that represents floor appearances. Selecting $x_F$ is much simpler than finding $x_O$. We assume that the floor area which the robot has already traveled is free of obstacles. Thus, we compute the mean color of the floor by using the lower $N = 15$ rows of the image. We find $x_F$ from outside the bounding box which has a minimum $\Delta E^*$ with the mean color of the floor.

### 6.4 Obstacle Score Computation

By using $x_O$ and $x_F$ as source vertices, we compute two geodesic distances $d_G(x_O)$ and $d_G(x_F)$. In $d_G(x_O)$, the shorter geodesic distance means more obstacle-like appearance. In contrast, the shorter distance in $d_G(x_F)$ means more floor-like appearance. Figure 9 illustrates geodesic distances from different source vertices (brighter pixels represent longer distances). By subtracting $d_G(x_O)$ from $d_G(x_F)$, we can calculate obstacle score as
Fig. 9 Geodesic distance from \(x_O\) (top) and \(x_F\) (bottom).

Fig. 10 Obstacle score.

\[ P(x_O, x_F) = d_G(x_F) - d_G(x_O). \]  

Lastly, obstacle score \(P(x_O, x_F)\) is binarized to detect obstacle pixels \(O\) by a certain threshold

\[ O_\lambda = \{ x : P(x_O, x_F) > \lambda \}. \]  

Figure 10 shows the obstacle score (brighter pixels represent higher scores).

7. Experiments

In this section, we evaluate the performance of our algorithm. The experiments use the mobile robot described in Section 3. Figure 11 illustrates our experimental setup. In the experiments, the travel speed of the robot is 0.2 m/s to 0.3 m/s. The images from the camera are acquired at a resolution of 640x480 pixels at approximately 5 fps. Before conducting the experiment, the camera is calibrated by the well-known checkerboard based method. The input images are resized to 320 x 240 for speeding up calculation (same as for the conventional method).

For the experiments, the following obstacles are selected: two types of chairs, a cup, an electric fan, a stand, a garment, a plant pot, a wire, a cleaner, and a trash box. Several obstacles are tested with different floor variations including wooden and checkered patterns.

The obstacle pixels are manually labeled for quantitative evaluation. Then, all pixels in the ROI are classified into the following four classes: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). We evaluate segmentation accuracy, precision, false positive rate, and recall as follows.

\[
\begin{align*}
\text{Accuracy} & = \frac{TP + TN}{TP + TN + FP + FN}, \\
\text{Precision} & = \frac{TP}{TP + FP}, \\
\text{False Positive Rate} & = \frac{FP}{TP + TN + FP + FN}, \\
\text{Recall} & = \frac{TP}{TP + FN}.
\end{align*}
\]

(10)

Table 1 Segmentation accuracy comparison.

<table>
<thead>
<tr>
<th></th>
<th>Proposed Method</th>
<th>Conventional Method [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>93.8%</td>
<td>92.9%</td>
</tr>
<tr>
<td>Precision</td>
<td>76.4%</td>
<td>61.0%</td>
</tr>
<tr>
<td>False positive rate</td>
<td>5.3%</td>
<td>6.9%</td>
</tr>
<tr>
<td>Recall</td>
<td>88.1%</td>
<td>95.1%</td>
</tr>
</tbody>
</table>

Table 2 Computation time comparison.

<table>
<thead>
<tr>
<th></th>
<th>Proposed Method</th>
<th>Conventional Method [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation time</td>
<td>26.6 ms (37.6 fps)</td>
<td>193.6 ms (5.2 fps)</td>
</tr>
</tbody>
</table>

the IPM detection described in [12] produces unstable detection results. Thus, we instead use the bounding box based region detection described in Section 5 of our paper.

Figure 12 shows the Precision-Recall graph for our obstacle datasets. The performance of the proposed method is shown as Precision-Recall curve by changing the thresholding parameter \(\lambda\). In Fig. 12, the conventional method [12] is marked as an gray dot since it has no critical thresholding parameters for controlling Precision-Recall relationship. The graph shows that the proposed algorithm achieves segmentation performance comparable to the state-of-the-art method.

Table 1 shows the accuracy comparison between the proposed algorithm at the best parameter cutting point (\(\lambda = 0.5\)) and the conventional method. The proposed method achieves better accuracy, precision, and false positive rate than the conventional method. In contrast, the conventional method achieves higher recall.

Table 2 shows the computation time of the proposed algo-
<table>
<thead>
<tr>
<th></th>
<th>Chair 1</th>
<th>Chair 2</th>
<th>Cup</th>
<th>Fan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Image</strong></td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td><strong>Proposed method</strong></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td><strong>Conventional method [12]</strong></td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Stand</th>
<th>Garment</th>
<th>Plant pot</th>
<th>Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Image</strong></td>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
<td><img src="image16" alt="Image" /></td>
</tr>
<tr>
<td><strong>Proposed method</strong></td>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
</tr>
<tr>
<td><strong>Conventional method [12]</strong></td>
<td><img src="image21" alt="Image" /></td>
<td><img src="image22" alt="Image" /></td>
<td><img src="image23" alt="Image" /></td>
<td><img src="image24" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Cleaner 1</th>
<th>Cleaner 2</th>
<th>Trash box 1</th>
<th>Trash box 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Image</strong></td>
<td><img src="image25" alt="Image" /></td>
<td><img src="image26" alt="Image" /></td>
<td><img src="image27" alt="Image" /></td>
<td><img src="image28" alt="Image" /></td>
</tr>
<tr>
<td><strong>Proposed method</strong></td>
<td><img src="image29" alt="Image" /></td>
<td><img src="image30" alt="Image" /></td>
<td><img src="image31" alt="Image" /></td>
<td><img src="image32" alt="Image" /></td>
</tr>
<tr>
<td><strong>Conventional method [12]</strong></td>
<td><img src="image33" alt="Image" /></td>
<td><img src="image34" alt="Image" /></td>
<td><img src="image35" alt="Image" /></td>
<td><img src="image36" alt="Image" /></td>
</tr>
</tbody>
</table>

Fig. 13 Obstacle segmentation results.
rithm and the conventional method. The computation time is measured using Intel Xeon E5-2687W v2 CPU running at 3.4 GHz without multithreading. Our algorithm is approximately 7.3 times faster than the conventional method. The most time consuming step of [12] is the per pixel texture/intensity PDF calculation for MRF based segmentation. Our algorithm abstracts the feature calculation by superpixel segmentation. Moreover, efficient geodesic distance calculation using Dijkstra’s algorithm also reduces the computation time. Figure 13 shows examples of qualitative segmentation results for our obstacle datasets. The experiments have shown that our algorithm achieves much faster computation with segmentation accuracy comparable to the conventional method.

8. Conclusions

This paper proposed a monocular vision based obstacle detection algorithm for autonomous mobile robots. In the first stage, we use IPM based method for obstacle region detection. In the second stage, we perform image abstraction and geodesic distance computation for obstacle segmentation. We use SLIC superpixels for decomposing the image into basic elements that preserve relevant structure, but abstract undesirable detail. The source superpixel for geodesic distance computation is selected according to the semi-local texture feature. Two geodesic distances computed from different source vertices give the obstacle score for accurate segmentation. Experimental results have shown that our proposed method is fast enough and achieves accuracy comparable to the state-of-the-art method. In the future, we wish to improve segmentation accuracy by using more robust distance metrics for geodesic distance computation. We are also planning to apply the proposed method for obstacle avoidance and path planning of mobile robots.

Acknowledgments

The authors are grateful to Prof. M.J. Dürst from Aoyama Gakuin University for his careful proofreading and kind advices on the manuscript.

References


Naoshi KANEKO

He received his B.S. and M.S. degrees from Aoyama Gakuin University, Japan, in 2012 and 2014, respectively. He is currently working toward the Ph.D. degree in the Graduate School of Science and Engineering, Aoyama Gakuin University. His research interests include computer vision and human sensing.

Takeshi YOSHIDA (Member)

He received his B.S., M.S., and Ph.D. degrees from Kobe University, Japan, in 2007, 2009, and 2013, respectively. In 2015, he joined Aoyama Gakuin University, where he is currently an Assistant Professor in the Department of Integrated Information Technology. His research interests include computer and robot vision.

Kazuhiro SUMI (Member)

He received his B.S., M.S., and Ph.D. degrees from Kyoto University, Japan, in 1982, 1984, and 1997, respectively. In 2011, he joined Aoyama Gakuin University, where he is currently a Professor in the Department of Integrated Information Technology. His research interests include computer vision for automation, surveillance, and biometrics.