Moving Objects Detection using Freely Moving Depth Sensing Camera

Yong-Deuk Shin, Jae-Han Park, Ga-Ram Jang, Moon-Hong Baeg
Applied Robot technology R&D Group, Korea Institute of Industrial Technology, Korea
{syd, hans1024, river.j, mhbaeg}@kitech.re.kr

Abstract

The detection of moving objects for the surveillance and monitoring has been studied in the computer vision community for many years. Traditionally, the studies assume the use of a stationary camera. When using a 3D point cloud, research is restricted to the fixed laser scanner because of the slow data acquisition time. In this paper, we propose a method for detecting moving objects based on a freely moving sensor that provides two-dimensional-three-dimensional (2D-3D) fused data. Our method is a frame-differencing, which compares two consecutive frames combining visual features and a 3D point cloud. The RANSAC and ICP algorithms are applied for more accurate results. The moving objects can be separated in the 3D point cloud by adopting RANSAC outliers.

1. Introduction

Awareness of moving objects in changing environments is one of the fundamental issues in computer vision. Since the earliest study in the late 1970s[1], many researchers have been studying moving object detection. Most conventional methods are region-based approaches such as background subtraction with a static camera. However, this method is sensitive to illumination changes and the movements of small objects such as the waving of branches or leaves. Many papers have proposed ways for achieving robustness to these limitations[2,3]. Increasing interest in cameras on mobile platforms such as cars, cellular phones, and robots in the recent years makes overcoming static camera limitations more important[4,5]. Research based on the assumption of a moving camera relies on the estimation of ego-motion compensation using homography or 2D affine transforms. However, along with the limitation of the camera, these methods are restricted to environments that are well approximated by planes. Moreover, camera motion is limited to rotation with no translation because the metric information of the environment is unknown.

The metric information of the environment can be easily acquired when a laser scanner is used. However, the corresponding studies are limited to change detection in the environment because data acquisition is very slow when detecting moving objects. Therefore, most studies assume a static laser scanner for monitoring changes in building sites, the interiors of facilities or geographic information[6,7]. Recently, depth sensing cameras such as Kinect enable the achievement of 2D images and 3D point clouds of the environment in real-time and at a low price. In this work, the Kinect sensor is used to detect moving objects.

2. Moving Objects Detection

We do not assume the sensor to be fixed and the objects to be stationary. That is, the sensor and objects move independently. The moving object detection problem can be solved by comparing two successive frames $F^{t-1}$ and $F^t$. However, directly comparing frame $F^{t-1}$ with $F^t$ is not efficient because the frame data are not represented with respect to the same coordinate system owing to the moving sensor. Moreover, the sensor has congenital noise.

To directly compare two successive frames, the successive point cloud of each frame must be represented in a common coordinate system. This is called registration. The 2D-3D fused sensor data is used in this paper. The SURF[8] of the 2D image and 3D point cloud data is applied to registration and also used to detect moving objects. The frame differences between frames $F^{t-1}$ and $F^t$, which are represented by
voxels, provide information on objects that are moved while the sensor position is changed.

However, there are many mismatched points because of sensor noise and the registration error. This results in ghost points when frame differentiering is performed. This problem is solved using the concept of segmentation.

2.1. Registration

The registration method is similar to that of P. Henry[10]. The 2D image features are used to find the corresponding 3D points. SURF descriptors are extracted from images \( I^{t-i} \) and \( I' \) of frames \( F^{t-i} \) and \( F' \), and matches are found. The 3D points in the 3D point cloud \( C^{t-i} \) and \( C' \) of the frames \( F^{t-i} \) and \( F' \) are correspond to these 2D matches. The matched 2D features and 3D points are clues to the solution of the registration. Though, the SURF is distinctive, there might be false matches and these outliers could lead to wrong results. Therefore these outliers must be removed for registration. The RANSAC algorithm is used to remove the outliers.

The registration process is composed of three steps. The first step is a SURF matching process between the 2D image and the 3D point cloud. The outliers are then rejected by the RANSAC algorithm. The final process involves refining the results with the ICP[9] algorithm.

2.1.1. Outlier Rejection by RANSAC

The nearest neighbor search is used as the matching method between the previous and current images. However, this method has drawbacks generating many outliers. These outliers could lead to large registration error. The RANSAC algorithm is used to remove outliers. Algorithm 1 shows the outlier rejection method with RANSAC. \( N \) corresponding matches are randomly selected and the rigid transformation \( H_{r}^{t-i} \) is calculated. The objective function used in the RANSAC algorithm determines whether matches can be inliers by measuring the Euclidean distance between the 3D points and the transformed 3D points. This means that a 3D point transformed by \( H_{r}^{t-i} \) should be located at the matched 3D point. The most voted candidate is selected as the transformation matrix in the RANSAC algorithm.

2.1.2. Refinement with ICP

Registration refinement is performed by using the ICP algorithm. In the first iteration, the ICP is initialized by the RANSAC result. ICP is a scan matching algorithm for solving rigid body transformation by finding point-pairs that have the minimum Euclidean distance. Corresponding point-pairs are re-evaluated after iteratively applying rigid body transformation. Consequently, as the number of iterations increases, the average Euclidean distance between corresponding point-pairs decreases.

The rejection technique is applied for a more accurate registration. The ICP rejection technique removes any point-pairs whose Euclidean distance is larger than the threshold, and dynamically reduces the threshold. It is essential to control the decreasing rejection threshold speed. If the speed can be properly controlled, the iteration number can be decreased and the RMS alignment error can be reduced. The rejection threshold \( e_{j} \) is adaptively evaluated at iteration \( j \) as:

\[
e_{j} = \alpha \mu_{j-1}
\]

where \( \mu_{j-1} \) is defined as follows:

\[
\mu_{j-1} = \frac{1}{n} \sum_{i} d_{i}^{t-i}
\]

where \( d_{i}^{t-i} \) and \( n \) are the Euclidean distance and the number of corresponding point-pairs, respectively.

2.2. Frame Differencing

In this paper, the subtraction process is performed using the voxels. This is searching for occupied voxels. Unoccupied voxels are candidates of the moving object.

However, as shown in the figure 3, there are point sets in the frame subtraction result, even though the registration is accurate. This is due to occlusion, that is, the point clouds that are occluded by objects in frame \( F^{t-i} \) are revealed when objects are moved. In addition, sensor noise and some registration errors generate
ghost points that are not rejected by the subtraction process. The ghost points and occluded point clouds are then removed as shown Section 2.3.

2.2.1. Voxel Representation

A simple voxel data structure is implemented as a 3D array. Each voxel has corresponding points. However, the implementation of the voxel with a 3D array needs a significant amount of additional memory. The use of tree data structure whose keys are coded voxel coordinates and whose values are the set of point clouds in these voxel is one solution to this problem. The key of the tree data structure is coded as follows.

Let the voxel coordinates be \( v_x, v_y, \) and \( v_z \). Then, the keys of the voxel are coded with \( n \) bits, as shown in figure 1. Each voxel coordinate is calculated as follows:

\[
\begin{align*}
  v_x &= \left\lfloor p_x / \varepsilon + 0.5 \right\rfloor \\
  v_y &= \left\lfloor p_y / \varepsilon + 0.5 \right\rfloor \\
  v_z &= \left\lfloor p_z / \varepsilon + 0.5 \right\rfloor
\end{align*}
\]

where \( p = [p_x, p_y, p_z]^T \) is the coordinate of the 3D point and \( \varepsilon \) is the edge length of the voxel. The voxel data structure is implemented as a red-black tree whose search time is \( O(\log N) \), where \( N \) is the number of points.

![Figure 1. Coded keys of the tree data structure.](image)

2.2.2. Setting of the Region of Interest

In the registration result of two frames with a moving sensor, the space occupied by frame \( F' \) could be outside of the region of frame \( F^{n-1} \). Therefore this outer region is recognized as the non-occupied region, that is, as the changed space. Therefore, the region of interest must be set in order to compare frames \( F^{n-1} \) and \( F' \). The sensor range is not exactly known and differs with sensors.; therefore, five planes (frustum) are set by calculating the boundary of the 3D point clouds that are represented by voxels in the occupied region by frame \( F^{n-1} \).

2.3. Segmentation

The frame subtraction result contains many undesired points such as ghost points and occlusion regions even though the registration is accurate. Therefore, these points should be removed. At the Section 2.1.1, the outliers are matches in the 2D images; however, it is determined outliers because the space that the matches is changed by the moving objects. Two kinds of outliers exist in the Section 2.1.1. One is the set of points on the objects after moving and the other is the set before moving. These outliers are shown in figure 3. As shown in the right image, the ghost points and occluded points are undesired points. The segmentation is started at the outliers of the RANSAC whose points are on the object after moving and its connected voxels are merged. If there is an occupied voxel in the 26-neighborhoods of the given voxel, the occupied voxel is said to be connected to the given voxel. All connected voxels are selected using the 26-neighborhoods until every neighbor of its neighbor has no connected voxels. Figure 2 shows all the connected voxels with a given voxel \( v \). The process starts at voxel \( v \) and connected voxels are found. Voxel \( v_i \) is connected to voxel \( v \), and voxel \( v_j \) is connected to voxel \( v_i \). Voxels \( v_i \), \( v_j \), and \( v_k \) are disconnected from \( v \).

![Figure 2. Connected voxels with 8-neighborhoods](image)

3. Experiment and Result

The proposed method was tested using the Kinect sensor, which provides 3D point clouds and 2D images at 30 Hz. Therefore, it is a suitable sensor for detecting moving objects. The Kinect image size is 640×480 and the number of points in one frame is 307,200. The RANSAC threshold \( \delta \) was set to 0.1 m and the rejection ratio \( \alpha \) was set to 1.87. The voxel length \( \varepsilon \) was set to 0.01 m for frame subtraction. In the segmentation process, \( \varepsilon \) was set to 0.002 m to search for connected voxels.
In this paper, the test environment was set using the following schedules to verify each step of our method. First, the reference image and 3D point clouds were acquired, and an object was moved. Then, the hand-held Kinect was moved to detect the moving object. Figure 3 shows the result of the experiment with one and two moving objects. As shown in the right images, there are only moving objects with no undesired points.

The running time was 300~400 ms because the SURF descriptor extraction time and the ICP time depend on the environment. A Sandbridge i7 CPU was used for the experiment.

4. Conclusion

We proposed a method for detecting moving objects using 2D-3D fused data in real-time with the concept of frame subtraction for two consecutive frames. The experimental results show that our method is efficient in detecting moving objects, and the running time is fast enough to detect slowly moving objects. However, the current implementation cannot be guaranteed for the highest speed of the Kinect sensor of 30 Hz. We intend to apply CUDA programming to achieve real-time implementation for a 30 Hz speed in future.

References