



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사학위논문

**동적 환경에 강인한 모션 분류 기반의  
영상 항법 알고리즘 개발**

**Robust Visual Odometry via Rigid Motion  
Segmentation for Dynamic Environments**

2017 년 8 월

서울대학교 대학원

기계항공공학부

이 상 일

# 동적 환경에 강인한 모션 분류 기반의 영상 항법 알고리즘 개발

Robust Visual Odometry via Rigid Motion  
Segmentation for Dynamic Environments

지도교수 김 현 진

이 논문을 공학석사 학위논문으로 제출함

2017 년 6 월




서울대학교 대학원

기계항공공학부

이 상 일

이상일의 공학석사 학위论문을 인준함

2017 년 6 월

위원장	김 유 칸	
부위원장	김 현 진	
위원	방 찬 구	

**Robust Visual Odometry via Rigid Motion Segmentation  
for Dynamic Environments**

A Thesis

by

SANGIL LEE

Presented to the Faculty of the Graduate School of  
Seoul National University  
in Partial Fulfillment  
of the Requirements  
for the Degree of

MASTER OF SCIENCE

Department of Mechanical & Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

August 2017

## Abstract

# Robust Visual Odometry via Rigid Motion Segmentation for Dynamic Environments

Sangil Lee

Department of Mechanical & Aerospace Engineering

The Graduate School

Seoul National University

In the paper, we propose a robust visual odometry algorithm for dynamic environments via rigid motion segmentation using a grid-based optical flow. The algorithm first divides image frame by a fixed-size grid, then calculates the three-dimensional motion of grids for light computational load and uniformly distributed optical flow vectors. Next, it selects several adjacent points among grid-based optical flow vectors based on a so-called entropy and generates motion hypotheses formed by three-dimensional rigid transformation. These processes for a spatial motion segmentation utilizes the principle of randomized hypothesis generation and the existing clustering algorithm, thus separating objects that move independently of each other. Moreover, we use a dual-mode simple Gaussian model in order to differentiate static and dynamic parts persistently. The model measures the output of the spatial motion segmentation algorithm and updates a probability vector consisting of the likelihood of representing specific label. For the evaluation of the proposed algorithm, we use a self-made dataset captured by ASUS Xtion Pro live RGB-D camera and Vicon motion capture system. We compare our algorithm with the existing motion segmentation algorithm and the current state-of-the-art visual odometry algorithm respectively, and the proposed algorithm estimates the ego-motion robustly and accurately in dynamic environments while showing the competitive performance of the motion segmentation.

Keyword : Visual odometry, Dynamic environments, Motion segmentation, Grid-based optical flow, RGB-D camera.

Student Number : 2015-20784

# Table of Contents

	<b>Page</b>
Abstract . . . . .	iii
Table of Contents . . . . .	iv
List of Figures . . . . .	v
List of Tables . . . . .	vii
<b>Chapter</b>	
1 Introduction . . . . .	1
1.1 Literature review . . . . .	2
1.2 Thesis contribution . . . . .	5
1.3 Thesis outline . . . . .	5
2 Background Knowledge . . . . .	6
2.1 Rigid transformation . . . . .	6
2.2 Grid-based optical flow . . . . .	9
3 Motion Spatial Segmentation . . . . .	10
3.1 Motion hypothesis search . . . . .	10
3.2 Motion hypothesis refinement . . . . .	12
3.3 Motion hypothesis clustering . . . . .	13
4 Motion Temporal Segmentation . . . . .	14
4.1 Label matching . . . . .	14
4.2 Dual-mode simple Gaussian model . . . . .	16
4.2.1 Update model . . . . .	17
4.2.2 Compensate model . . . . .	19
5 Evaluation Results . . . . .	20
5.1 Dataset . . . . .	20
5.2 Motion segmentation . . . . .	21
5.3 Visual odometry . . . . .	26
6 Conclusion . . . . .	34

# List of Figures

1.1	Examples of dynamic environments: TUM fr3/walking_xyz, KITTI 2011_10_03/47, and service robot in the hallway (clockwise). . . . .	2
1.2	The pipeline of the proposed algorithm. It fetches RGB-D images, calculates grid-based optical flow vectors, segments motions, and estimates ego-motion sequentially. When it fails at some stage, it jumps to the next iteration by ignoring the current RGB-D image which caused failure. . . . .	3
2.1	The Euclidean transformation in the three-dimensional space. It preserves distances between every pair of points allowing only rotation and translation. . . . .	7
4.1	The framework of the dual-mode simple Gaussian model. Both apparent and candidate label models (“dual-mode”) are compensated and updated, iteratively. . . .	17
4.2	The procedure of updating a probability vector in the corresponding grid. Different labels are denoted as different colors. The probability vector makes the algorithm robust to an erroneous result of spatial segmentation based on the statistical approach. . . .	18
5.1	Internal parameters of the proposed motion segmentation algorithm. An object moves around as shown in the figure entitled <i>Current image</i> . <i>Grid-based optical flow</i> figure shows how the previous grid-based points move as denoted by white circle. Figure <i>x, y-flow</i> represents the magnitude and the direction of optical flow vectors as saturation and hue, respectively, whereas <i>z-flow</i> map only shows the magnitude of the <i>z</i> -directional flow, brighter as the object moves away from the camera. The figure entitled <i>Hypotheses</i> shows the result of motion spatial segmentation just before clustering motion hypotheses, and <i>Segments</i> shows the result after clustering motion hypotheses. Figure <i>Group</i> shows the result of the label matching explained in Section 4.1. Finally, figure <i>Model</i> is the output of the grid model. Note that different objects are denoted as different colors. . . . .	22

5.2	Qualitative analysis of motion segmentation algorithms. From top to bottom: original image, our segmentation, MCD5.8ms, and randomized voting. The proposed algorithm shows different motions in different colors, and MCD5.8ms detects dynamic pixels. On the other hand, randomized voting algorithm only divides into several parts which have different motions to each other, so label of the grid can be easily changed. . . . .	25
5.3	Absolute trajectory error and XYZ positions for <i>Fixed Camera 1</i> dataset. Absolute trajectory error is defined in Eq. (5.6). XYZ positions denotes the estimated position of the fixed camera in each $x, y, z$ -axis, and true positions of the camera are consistently zeros. . . . .	28
5.4	Absolute trajectory error and XYZ positions for <i>Fixed Camera 2</i> dataset. Please be informed that two objects appear around 10 seconds. . . . .	29
5.5	XYZ position errors and XYZ positions for <i>Vicon Room 1</i> dataset. XYZ errors mean the difference between the estimated position and the true in each $x, y, z$ -axis, and true values are denoted as black solid line. . . . .	30
5.6	XYZ position errors and XYZ positions for <i>Vicon Room 2</i> dataset. . . . .	31
5.7	XYZ position errors and XYZ positions for <i>Vicon Room 3</i> dataset. . . . .	32
5.8	XYZ position errors and XYZ positions for <i>Vicon Room 1</i> dataset. Proposed $\times$ {DVO, ORB-SLAM2, and ORB-VO} estimate the ego-motion with notable accuracy through combining with the proposed motion segmentation. For algorithms which are based on the same visual odometry technique, they denoted as the same line style. . . . .	33



# List of Tables

5.1	Description of datasets for evaluating visual odometry algorithm. . . . .	21
5.2	Definition of true positive, true negative, false positive, and false negative. . . . .	23
5.3	Quantitative analysis of the proposed and existing motion segmentation algorithms.	23
5.4	Evaluation of visual odometry algorithms. . . . .	27

# 1

## Introduction

Visual odometry is an essential computer vision technology to recognize the ego-motion of the camera itself using video input [1, 2]. Various visual odometry algorithms have been successful in well-defined datasets such as TUM [3] and KITTI [4], and well-conditioned environments. Most of existing visual odometry methods assume stationary environments so that it could estimate the position of the camera through the motion of the image taken. However, most real environments involve dynamic situations including moving objects such as residential road or crowded hallway as shown in Fig. 1.1. Although some of existing visual odometry methods can exclude a distinct motion or pixels by calculating ego-motion with RANSAC, they are restricted only to the non-stationary object that occupies small areas in the image. Therefore, in order to implement a robust visual odometry algorithm, it is required to separate non-stationary objects from the stationary background.

To distinguish between the static and dynamic elements, we use temporary motions distributed uniformly in the image since these motions could also be used to estimate the position of the camera. For calculating the motion of pixels, we utilize optical flow methods because optical flow calculates the temporary motion even when it is difficult to extract particular pixels by the feature-based method. There are roughly two types of optical flow: sparse and dense. Both methods show the remarkable difference in computational time and performance. A dense three-



Figure 1.1: Examples of dynamic environments: TUM fr3/walking\_xyz, KITTI 2011\_10\_03/47, and service robot in the hallway (clockwise).

dimensional optical flow, also known as dense scene flow [5, 6], has heavy computational load for high performance, whereas a sparse flow has a drawback similar to feature-based method as mentioned before. In order to implement a computationally-light application with a proper performance, therefore, we use grid-based optical flow taking advantage of both methods. After applying optical flow, motion segmentation is performed to distinguish between dynamic objects and stationary background with grid-based temporary motions and then motion estimation of the camera is executed using static elements.

In this paper, our approach focuses on the robustness of the performance for the fast visual odometry techniques. Fig. 1.2 shows the pipeline of our algorithm. The algorithm executes stages step by step as it satisfies heuristic criteria. Our proposed algorithm only uses three-dimensional motions of grid-based pixel points to separate dynamic objects from the stationary background without any restriction on the movement of objects and enables to estimate the ego-motion while classifying the moving object in the video sequence data.

## 1.1 Literature review

---

Most existing visual odometry algorithms [7–10] deal with the ego-motion in stationary environments. However, a number of non-stationary objects exist in a real environment such as crowded corridor or pavement. To deal with such dynamic environments, a few recent research attempt to

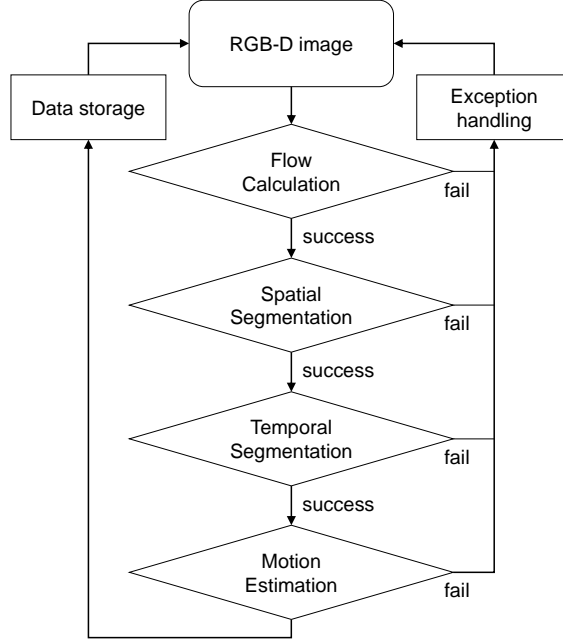


Figure 1.2: The pipeline of the proposed algorithm. It fetches RGB-D images, calculates grid-based optical flow vectors, segments motions, and estimates ego-motion sequentially. When it fails at some stage, it jumps to the next iteration by ignoring the current RGB-D image which caused failure.

improve robustness against moving objects.

Kitt *et al.* [11] utilizes support vector machine for distinguishing between stable features and unstable features which cause position estimate error. However, it has to learn features in advance, thus it is difficult to show reliable performance in an arbitrary situation.

To distinguish the stationary background from dynamic environments, some research uses the property that an independent object shows a different motion which is distinct from the movement of the stationary background. One of these studies applies existing structure from motion technique to separation of distinct motions [12,13]. By doing so, the algorithm distinguishes between ego-motion and so-called eoru-motion. In their paper, *eoru* is coined to describe the movement of a dynamic object. Their algorithm generates over 100 hypotheses within a specified number of frames to estimate ego-motion or eoru-motion. Although it separates independent motion trajectories without prior information about the number of objects, its computational time takes too long for real-time applications and longer when it generates more hypotheses for accuracy.

Background model-based RGB-D dense visual odometry (BaMVO) [14] estimates background by choosing pixels whose depth does not change significantly during a specified number of frames. In their experiment, a dynamic object like pedestrian moves in parallel with the camera direction so that its depth value changes drastically. Its performance can become degenerated if the dynamic object moves perpendicular to the camera direction reducing the depth transition.

Dib *et al.* [15] uses RANSAC for direct visual odometry while considering the dynamic environment. In their research, hypothesis generation and evaluation functions are the same as naïve direct visual odometry. However, the algorithm they proposed uses the photometric optimization using six patches, not all of the image. If any one of six image patches belongs to the dynamic object, its photometric error could not be smaller than the heuristic threshold. However, if all of these patches belong to the stationary background, then all patches become inliers, i.e. background. By repeating this process, their algorithm subtracts a moving object from the stationary background. In a similar study, Jung *et al.* [16] rejects a dynamic object by choosing a patch which has quite different motion with majority movement of the camera. The algorithm classifies image patches by categorizing according to depth value in advance. Then it rejects a class which has a disparate motion by comparing standard deviation of whole motion vectors including or excluding that class. Both methods still assume that the stationary background occupies a large part of the image so that it builds majority movement.

Among the motion segmentation methods, randomized voting (RV) [17] is one of the algorithms that could be extended to the visual odometry. Their algorithm utilizes epi-polar constraints to extract a motion using randomized voting algorithm they proposed. Although they claim that their algorithm is much faster and more efficient than the other state-of-the-art algorithms such as the principal angles configuration (PAC) [18] and sparse subspace clustering (SSC) [19], it also does not fulfil a real-time visual odometry application because of the computational time of 300 milliseconds per frame for executing motion segmentation only.

MCD5.8ms [20] has an advantage in that it detects a moving object with a low computational load while showing the execution time per frame of 5.8ms. It first divides an image by a fixed-size grid which has mean, variance, and age as model parameters. These parameters of each grid are compensated and updated using feature-based homography and pixel's intensities, respectively. Since it uses a traditional visual odometry technique for calculating homography, its performance

can degenerate severely when a moving object occupies more than half of image. Besides, it chooses pixels whose intensity value differs from a mean of the grid model, thus it is difficult to distinguish objects in the situation where the color of a dynamic object is similar to one of a static background.

## 1.2 Thesis contribution

---

Our main contributions can be summarized as follows:

1. We propose a visual odometry algorithm which is robust in dynamic environments. It estimates the motion of stationary scene (equal to background) and non-stationary objects separately using the designed motion segmentation algorithm, thus it calculates position and orientation of the camera itself versus the stationary background.
2. We design a rigid motion segmentation algorithm, which distinguishes between the background or moving objects. It calculates  $x, y$ -flows from RGB images and  $z$ -flows from depth images via optical flow. Then, it generates a specified number of motion hypotheses, and clusters them so that the algorithm separates distinct motions.
3. Our method separates independent rigid motions with no prior information such as shapes or the number of objects using three-dimensional optical flow, thus dynamic objects can be extracted robustly.

## 1.3 Thesis outline

---

The rest of this paper is organized as follows: In Chapter 2, we describe the basic calculation of rigid transformation for estimating the relationship between two views of the camera, and a grid-based optical flow algorithm for the RGB-D camera. Next, the motion spatial segmentation and motion temporal segmentation algorithms are explained in Chapter 3 and Chapter 4, respectively. Chapter 5 provides off-line experimental results and comparisons with state-of-the-art motion segmentation and visual odometry algorithms, and then discussions follow. Chapter 6 summarizes the issues of our topic covered in the paper.

# 2

## Background Knowledge

In this section, we first provide the basic notations for visual odometry algorithms and explain how to calculate rigid transformation for describing the translational and rotational relationship between two different views. Then, the process of optical flow based on the Lucas-Kanade method [21] follows in the next section.

### 2.1 Rigid transformation

---

The proposed algorithm is based upon the property that the three-dimensional velocities of the three-dimensional points which belong to the same rigid object spatially have the same rigid motion,  $\mathbf{H}$ , temporally.

**Definition 1** Let  $\mathbf{X}_j$  and  $\mathbf{X}_k$  denote the corresponding  $m$ -points set of the  $j$ -th and  $k$ -th images, respectively, where  $\mathbf{X}_l = [\mathbf{x}_1^{(l)}, \mathbf{x}_2^{(l)}, \dots, \mathbf{x}_m^{(l)}] \in \mathbb{R}^{3 \times m}$  is a three-dimensional point set, and  $\mathbf{x}_*^{(l)} = [x, y, z]^T \in \mathbb{R}^{3 \times 1}$  is a three-dimensional point in the  $l$ -th frame represented by the Cartesian coordinates. Then, the rigid rotation and translation matrix from the  $j$ -th coordinates to the  $k$ -th coordinates are defined as follows:

$$\mathbf{R} = \mathbf{M}(\mathbf{M}^T \mathbf{M})^{-1/2} \quad (2.1)$$

$$\mathbf{t} = -\mathbf{R}\mathbf{P}_j + \mathbf{P}_k, \quad (2.2)$$

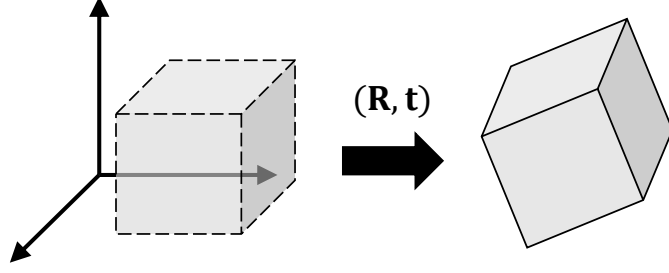


Figure 2.1: The Euclidean transformation in the three-dimensional space. It preserves distances between every pair of points allowing only rotation and translation.

where

$$\mathbf{P}_j = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^{(j)} \quad \text{and} \quad \mathbf{P}_k = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^{(k)}$$

$$\mathbf{M} = (\mathbf{X}_k - \mathbf{P}_k) (\mathbf{X}_j - \mathbf{P}_j)^T.$$

Consequently, the rigid transformation matrix is defined as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \quad (2.3)$$

which satisfies

$$\mathbf{H} \cdot \begin{bmatrix} \mathbf{X}_j \\ \mathbf{1}^T \end{bmatrix} = \begin{bmatrix} \mathbf{X}_k \\ \mathbf{1}^T \end{bmatrix}, \quad (2.4)$$

where  $\mathbf{0} = [0, \dots, 0]^T \in \mathbb{R}^{m \times 1}$  and  $\mathbf{1} = [1, \dots, 1]^T \in \mathbb{R}^{m \times 1}$ .

**Proof 1** The above problem is equivalent to finding the optimal  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  and  $\mathbf{t} \in \mathbb{R}^3$  that minimize the following:

$$f(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^m \left\| \mathbf{R} \mathbf{x}_i^{(j)} + \mathbf{t} - \mathbf{x}_i^{(k)} \right\|^2 \quad (2.5)$$

subject to

$$\mathbf{R}^T \mathbf{R} = \mathbf{I}, \quad (2.6)$$

where  $\|\cdot\|$  represents the 2-norm. For simplicity, as mentioned previously, we denote  $[\mathbf{x}_1^{(j)}, \mathbf{x}_2^{(j)}, \dots, \mathbf{x}_m^{(j)}]$  as  $\mathbf{X}_j$ , and  $[\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}, \dots, \mathbf{x}_m^{(k)}]$  as  $\mathbf{X}_k$ . With a Lagrange multiplier symmetric matrix  $\Lambda \in \mathbb{R}^{3 \times 3}$  and  $\mathbf{1} := [1, \dots, 1]^T \in \mathbb{R}^{m \times 1}$ , augmented cost  $\mathbf{J}$  is

$$\mathbf{J} = \text{trace} \left( (\mathbf{R} \mathbf{X}_j + \mathbf{t} \cdot \mathbf{1}^T - \mathbf{X}_k)^T (\mathbf{R} \mathbf{X}_j + \mathbf{t} \cdot \mathbf{1}^T - \mathbf{X}_k) + \Lambda (\mathbf{R}^T \mathbf{R} - \mathbf{I}) \right). \quad (2.7)$$



Then, necessary conditions for optimality are

$$\frac{\partial \mathbf{J}}{\partial \mathbf{R}} = 2\mathbf{X}_j \mathbf{X}_j^T \mathbf{R}^T + 2\mathbf{X}_j \cdot \mathbf{1} \cdot \mathbf{t}^T - 2\mathbf{X}_j \mathbf{X}_k^T + 2\Lambda \mathbf{R}^T = 0 \quad (2.8)$$

$$\frac{\partial \mathbf{J}}{\partial \mathbf{t}} = 2 \cdot \mathbf{1}^T \mathbf{X}_j^T \mathbf{R}^T + 2 \cdot \mathbf{1}^T \mathbf{1} \cdot \mathbf{t}^T - 2 \cdot \mathbf{1}^T \cdot \mathbf{X}_k^T = 0 \quad (2.9)$$

$$\frac{\partial \mathbf{J}}{\partial \Lambda} = \mathbf{R}^T \mathbf{R} - \mathbf{I} = 0. \quad (2.10)$$

Note that

$$\mathbf{X}_j \cdot \mathbf{1} = m\mathbf{P}_j \quad \text{and} \quad \mathbf{X}_k \cdot \mathbf{1} = m\mathbf{P}_k, \quad (2.11)$$

where

$$\mathbf{P}_j = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^{(j)} \quad \text{and} \quad \mathbf{P}_k = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i^{(k)}. \quad (2.12)$$

By substitution Eq. (2.9) into Eq. (2.8) eliminating  $\mathbf{t}$ , we obtain

$$\mathbf{X}_j \mathbf{X}_j^T \mathbf{R}^T + \mathbf{X}_j \cdot \mathbf{1} \cdot (\mathbf{1}^T \mathbf{X}_k^T - \mathbf{1}^T \mathbf{X}_j^T \mathbf{R}^T) \frac{1}{m} - \mathbf{X}_j \mathbf{X}_k^T + \Lambda \mathbf{R}^T = 0. \quad (2.13)$$

Then, we can solve the rotation matrix by

$$\mathbf{R} = \left( \mathbf{X}_k \mathbf{X}_j^T - \mathbf{X}_k \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T \right) \left( \mathbf{X}_j \mathbf{X}_j^T - \mathbf{X}_j \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T + \Lambda \right)^{-1}. \quad (2.14)$$

From Eq. (2.10), we have

$$\begin{aligned} & \left( \mathbf{X}_j \mathbf{X}_j^T - \mathbf{X}_j \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T + \Lambda \right) = \\ & \left( \left( \mathbf{X}_k \mathbf{X}_j^T - \mathbf{X}_k \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T \right)^T \left( \mathbf{X}_k \mathbf{X}_j^T - \mathbf{X}_k \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T \right) \right)^{1/2}. \end{aligned} \quad (2.15)$$

Therefore, from Eqs. (2.9), (2.14) and (2.15)

$$\mathbf{R} = \mathbf{M}(\mathbf{M}^T \mathbf{M})^{-1/2} \quad (2.16)$$

$$\mathbf{t} = \frac{1}{m} \cdot (\mathbf{X}_k \cdot \mathbf{1} - \mathbf{R} \cdot \mathbf{X}_j \cdot \mathbf{1}) = \mathbf{P}_k - \mathbf{R} \mathbf{P}_j, \quad (2.17)$$

where

$$\mathbf{M} = \left( \mathbf{X}_k \mathbf{X}_j^T - \mathbf{X}_k \cdot \frac{\mathbf{1} \cdot \mathbf{1}^T}{m} \cdot \mathbf{X}_j^T \right) = (\mathbf{X}_k - \mathbf{P}_k) (\mathbf{X}_j - \mathbf{P}_j)^T. \quad (2.18)$$

■

Moreover, for the evaluation of the generated motion hypothesis, we define the rigid transformation error as below:

$$\mathbf{E}(\mathbf{H}, \tilde{\mathbf{X}}_j, \tilde{\mathbf{X}}_k) := \text{diag} \left( \left( \mathbf{H} \tilde{\mathbf{X}}_j - \tilde{\mathbf{X}}_k \right)^T \left( \mathbf{H} \tilde{\mathbf{X}}_j - \tilde{\mathbf{X}}_k \right) \right) \in \mathbb{R}^{1 \times n}, \quad (2.19)$$

where  $\tilde{\mathbf{X}}_l = [\tilde{\mathbf{x}}_1^{(l)}, \tilde{\mathbf{x}}_2^{(l)}, \dots, \tilde{\mathbf{x}}_n^{(l)}] \in \mathbb{R}^{4 \times n}$  is a three-dimensional point set, and  $\tilde{\mathbf{x}}_*^{(l)} = [x, y, z, 1]^T \in \mathbb{R}^{4 \times 1}$  is a three-dimensional point in the  $l$ -th frame represented by homogeneous coordinates. Note that sum of elements in Eq. (2.19) is equal to Eq. (2.5).

## 2.2 Grid-based optical flow

---

We utilize the naïve Lucas-Kanade optical flow method implemented in mexopencv [22] to extract a three-dimensional motion of specific pixels basically. However, there are some issues for using the naïve optical flow method as it is. Under the situations such as the appearance of motion blur and the existence of repetitive or texture-less patterns, the performance of optical flow tends to degenerate. To resolve these issues, we check the validity of optical flow by calculating a bi-directional error and a similarity. The bi-directional error is the distance from the original point to forward-backward tracked point, and the similarity is calculated from the two-dimensional correlation coefficient between image patches surrounding original and point tracked by optical flow. Both validations are effective methods to eliminate unreliable optical flow vectors.

For estimating the depth direction of a velocity, we refine the quality of the depth image in advance. A pixel whose depth value is invalid is filled with a similar value in the vicinity, and a pixel whose depth value shows abnormally large change is eliminated. After refining the depth image, we utilize the two-dimensional interpolation to obtain the depth of tracked points.

# 3

## Motion Spatial Segmentation

Algorithm 1 shows a pseudocode which explains the motion spatial segmentation procedure. A total of  $n$  grid-based optical flow vectors is fetched to the algorithm, and segmented label and motion are derived. Some of parameters used are the number of hypothesis generations,  $H$ , the number of points,  $m$ , to estimate motion hypothesis, i.e., position variation of  $m$ -points, and the threshold value to decide whether  $m$ -points belong to the same object. Motion spatial segmentation procedure is roughly divided into three kinds of processes: motion hypothesis search, refinement, and clustering.

### 3.1 Motion hypothesis search

---

The algorithm consists of searching rigid motion hypotheses, refining hypotheses, and clustering hypotheses. The first two processes, i.e., searching and refining hypotheses, are executed iteratively until it finds a total of  $H$  hypotheses. First, it chooses 1-point randomly with regarding an entropy,  $\mathbf{S}$ , as a sampling weight. As will be explained in details in the next section, the entropy is a measure of how well the estimate of a hypothesis is done in the corresponding pixels. As a result, a point of which entropy is high has a high probability to be chosen. Then it selects  $(m - 1)$ -points within the radius of the specified size around the chosen 1-point so that the probability of choosing  $m$ -points which belong to the same object is high. By using the selected  $m$ -points, it is possible to

---

**Algorithm 1** Motion Spatial Segmentation

---

**Input:**  $\{\mathbf{V}_i\}; i = 1, \dots, n$  $\triangleright n$  Grid-based points**Output:**  $\mathbf{G}$  $\triangleright$  Group label-indexed data

```
1:  $\mathbf{G} \leftarrow \emptyset$ 
2:  $h \leftarrow 0$   $\triangleright$  Where  $h$  is the number of hypotheses
3: while  $h < H$  and  $\text{mean}(\mathbf{S}) > \mathbf{S}_{\min}$  do
4:   Select 1-point in  $\{\mathbf{V}_i\}$  randomly with regarding entropy,  $\mathbf{S}$ , as a weight
5:   Select  $(m - 1)$ -points randomly near to the 1-point
6:   Estimate  $\mathbf{H}$  using the selected  $m$ -points
7:   Evaluate  $\mathbf{H}$  within  $m$ -points (see Section 3.2)
8:   if  $m$ -points belong to the same motion then
9:      $N_{\max} \leftarrow m$ 
10:    repeat
11:       $N \leftarrow$  Evaluate  $\mathbf{H}$  for all  $n$ -points  $\triangleright$  Where  $N$  is the number of inliers
12:      if  $N > N_{\max}$  then
13:         $N_{\max} \leftarrow N$ 
14:        Estimate  $\mathbf{H}$  using the inliered  $N$ -points
15:      end if
16:    until  $N = N_{\max}$ 
17:     $\mathbf{H}_h \leftarrow \mathbf{H}$ 
18:    Evaluate  $\mathbf{H}_h$  for all  $n$ -points
19:    Calculate  $\mathbf{S}$  using evaluation error
20:     $h \leftarrow h + 1$ 
21:  end if
22: end while
23:  $\mathbf{G} \leftarrow$  Cluster  $\{\mathbf{H}_{h=1, \dots, H}\}$  (see Section 3.3)
```

---

estimate the rigid motion hypothesis,  $\mathbf{H}$ . In this process, we can choose high values of  $m$  or the radius size for robustness to the noise of optical flow vector. On the other hand, we can increase the resolution for detecting the small rigid object by decreasing the searching radius for  $m$ -points

or the number of points,  $m$ .

## 3.2 Motion hypothesis refinement

---

In the process of refining motion hypotheses, the proposed algorithm calculates rigid transformation error of the  $m$ -points in order to check whether they constitute one rigid object. If they belong to the same object, the algorithm evaluates the hypothesis for all  $n$  grid-based points. This procedure is designed with consideration for situations where a rigid object could appear in multiple parts on the image, so we regard multiple distinct areas that have the same motion as the same object. If the number of inliers whose motions are almost the same,  $N$ , is larger than  $m$  or the maximum number of inliers,  $N_{\max}$ , in the previous iteration, the algorithm sets the value of  $N_{\max}$  as  $N$ , and recalculate hypothesis of the inliers for the better preciseness.

When it converges, this hypothesis becomes an element of the motion hypothesis set. At the same time, the algorithm calculates an entropy by evaluating rigid transformation error for all  $n$  grid-based points and elements of the hypothesis set. The entropy,  $\mathbf{S}$ , for all  $n$  grid-based points is defined as follows:

$$\mathbf{S}_i = 1 - \min_h \exp(-\lambda \mathbf{E}_i - \delta), \quad i = 1 \dots n, \quad (3.1)$$

where

$$\mathbf{E}_i = \left[ \mathbf{E}^{(i)} \left( \mathbf{H}_1, \tilde{\mathbf{X}}_j, \tilde{\mathbf{X}}_k \right), \dots, \mathbf{E}^{(i)} \left( \mathbf{H}_h, \tilde{\mathbf{X}}_j, \tilde{\mathbf{X}}_k \right) \right] \in \mathbb{R}^{1 \times h}. \quad (3.2)$$

In the above equations,  $\tilde{\mathbf{X}}_l \in \mathbb{R}^{4 \times n}$  is a three-dimensional point set of  $n$  grid-based points in the  $l$ -th frame represented by homogeneous coordinates, and  $\mathbf{E}^{(i)}(\cdot)$  is the  $i$ -th element in the output of the  $\mathbf{E}(\cdot)$  function.  $h$  is the number of elements in the hypothesis set and  $\lambda$ ,  $\delta$  are parameters for adjusting the characteristics of the entropy. For the moderate increase of the entropy, we can choose a small value of  $\lambda$ . Also, small  $\delta$  results in decreasing the difference between the maximum and minimum of the entropy. In the implemented algorithm, we choose  $10^3$  as the value of  $\lambda$ , and  $10^{-2}$  as the value of  $\delta$ .

---

**Algorithm 2** Motion hypotheses clustering

---

**Input:**  $\{\mathbf{H}_h\}; h = 1, \dots, H$  $\triangleright$  Hypothesis set**Output:**  $\mathbf{G}$  $\triangleright$  Group label-indexed data

```
1:  $\mathbf{G} \leftarrow \emptyset$ 
2:  $\{\mathbf{H}_{h=1,\dots,g}\} \leftarrow$  Cluster motion hypotheses,  $\{\mathbf{H}_{h=1,\dots,H}\}$ 
3: for each hypothesis,  $\mathbf{H} \in \{\mathbf{H}_{h=1,\dots,g}\}$  do
4:    $N_{\max} \leftarrow 0$ 
5:   repeat
6:      $N \leftarrow$  Evaluate  $\mathbf{H}$  for all  $n$ -points  $\triangleright$  Where  $N$  is the number of inliers
7:     if  $N > N_{\max}$  then
8:        $N_{\max} \leftarrow N$ 
9:       Estimate  $\mathbf{H}$  using the inliered  $N$ -points
10:    end if
11:  until  $N = N_{\max}$ 
12:  if  $N_{\max} > N_{\text{threshold}}$  then
13:    Insert  $\mathbf{H}$  into  $\mathbf{G}$ 
14:  end if
15: end for
```

---

### 3.3 Motion hypothesis clustering

---

In order to find several distinct motions among the  $H$  hypotheses, we make use of an existing clustering algorithm, in our case, density-based spatial clustering of applications with noise (DBSCAN) [23]. Hypotheses are refined as the algorithm divides  $H$  hypotheses into  $g$  distinct motions. As shown in the Algorithm 2, clustering algorithm fetches  $H$  hypotheses into DBSCAN in advance. Then it refines the clusters in the same way as explained at the line 9-15 in the Algorithm 1. The process repeats clustering and refining until the number of clusters converges.

# 4

## Motion Temporal Segmentation

In addition to distinguishing objects that move independently from each other in a frame, we propose an algorithm to track the object. Since the proposed motion spatial segmentation algorithm executes in a similar way as two-frame based segmentation, it has no idea which is which. Thus, we implement motion temporal segmentation following the motion spatial segmentation. It provides label matching algorithm and probabilistic approach based on the dual-mode simple Gaussian model [20].

### 4.1 Label matching

---

The label matching algorithm simply calculates a so-called correlation coefficient between segments and finds label pairs whose correlation is the maximum along the column standing for the current segment. The correlation coefficient between two segments is defined as follows:

$$\mathbf{C}_{ij}^{(k,l)} = \frac{N_{i,j}^{(k,l)}}{\sqrt{N_i^{(k)} N_j^{(l)}}}, \quad (4.1)$$

where  $N_i^{(k)}$  is the number of grid-based points which belong to the  $i$ -th segment in the  $k$ -th frame, and  $N_{i,j}^{(k,l)}$  is the number of grid-based points which belong to both the  $i$ -th and  $j$ -th segments in the  $k$ -th and  $l$ -th frame, respectively. Also, a correlation score between the  $(k, l)$ -th frames is

defined as follows:

$$score = \frac{\sum_{i=1}^g \left( \left[ \mathbf{C}_{ii}^{(k,l)} \right]_{i=1\dots g} \circ \left[ N_i^{(k)} \right]_{i=1\dots g} \right)}{\sum_{i=1}^g N_i^{(k)}}, \quad (4.2)$$

where  $\circ$  operator is the Hadamard product,  $g$  is the number of observed label, and

$$[N_i]_{i=1\dots g} = [N_1, \dots, N_g]. \quad (4.3)$$

Through the above equation, the algorithm tracks the object that appeared previously, and puts a new label on the unmatched pairs of the current segments. The pseudocode is shown in the Algorithm 3.

---

**Algorithm 3** Motion Temporal Segmentation

---

**Input:**  $\mathbf{G}^{(l)}; l = j, \dots, k$

▷ Group label-indexed array

**Output:**  $\mathbf{G}^{(k)}$

▷ The current  $k$ -th groups

```

1:  $\mathbf{M} \leftarrow \emptyset$                                 ▷ Where  $M$  is matching pairs for temporal grouping
2:  $l = 1$                                           ▷ Where  $l$  is a frame index for iteration
3:  $\text{maxCorr} = 0$ 
4: for  $l < \min \{ \text{tMinSize}, k \}$  do          ▷ Where tMinSize is pre-defined constant for window size
5:    $\mathbf{M}^{(k,l)} \leftarrow \text{Match pairs between } \mathbf{G}^{(k)}, \mathbf{G}^{(l)}$ 
6:    $\mathbf{R}^{(k)} \leftarrow \text{Unmatched pairs of } \mathbf{G}^{(k)}$ 
7:    $\text{corr} \leftarrow \text{Calculate correlation score of } \mathbf{M}^{(k,l)}$ 
8:   if  $\text{corr} > \text{maxCorr}$  then
9:      $\text{maxCorr} \leftarrow \text{corr}$ 
10:     $\mathbf{M} \leftarrow \mathbf{M}^{(k,l)}$ 
11:     $\mathbf{R} \leftarrow \mathbf{R}^{(k)}$ 
12:   end if
13:    $l \leftarrow l + 1$ 
14: end for
15:  $\mathbf{G}^{(k)} \leftarrow \text{Rearrange group label, } \mathbf{G}^{(k)}$  with  $\mathbf{M}$  and  $\mathbf{R}$ 
16: Compensate dual-mode Gaussian models
17: Update models with  $\mathbf{G}^{(k)}$  as measure

```

---



In the algorithm, we set the maximum number of identified objects in one frame as  $m$  for reducing computational load and not assigning numerous identifier to erroneous label, i.e. noise. Thus, we count the number of the labels being observed during execution time. Then, we choose the identifier which is less observed among the unobserved identifiers in the  $m$  stacks. Moreover, for the robustness, we define a constant, which denotes how long grid retains its identifier. Even though the grid loses the object temporarily, the corresponding identifier is not assigned to a newly observed object. Naturally the region which occupies over half of the image is registered as low identifier preferentially, for this reason, static background should cover a larger area than dynamic objects in the first frame thereby being assigned as “identifier 1”. For reference, our algorithm regards “identifier 1” as static background, and others as dynamic objects.

## 4.2 Dual-mode simple Gaussian model

---

Since the RGB-D camera has a minimum distance of range, it can fail to measure a depth. Moreover, it is reasonable to assume that the static and dynamic elements do not appear or disappear abruptly on a frame by frame basis. Thus, in order to track the static and dynamic parts in the image sequence consistently, we use a simple Gaussian model which has a probability vector and an age as properties. Fig. 4.1 shows a framework of the dual-mode simple Gaussian model. It, first, compensates two models through the previously estimated ego-motion for updating the model with the measurement corresponding to the identical three-dimensional point. Then, we update the probability vector and age of both models based on certain criteria. For instance, when candidate model is updated more frequently than its apparent model, both models are swapped with each other. Finally, temporal labels can be selected as indices which indicate the maximum value in each probability vector.

As mentioned before, the designed simple Gaussian model has the probability vector as a property. The  $i$ -th element of a probability vector means the likelihood of corresponding pixel belonging to the  $i$ -th label. The probability vector has a size of  $m$  which means the maximum number of identified objects in a frame as introduced in Section 4.1. Fig. 4.2 describes the probability vector. In the figure, top-left grid denotes the measured label, and bottom-left one represents updated label as distinct colors. By utilizing the Gaussian model with probability vector, our algorithm

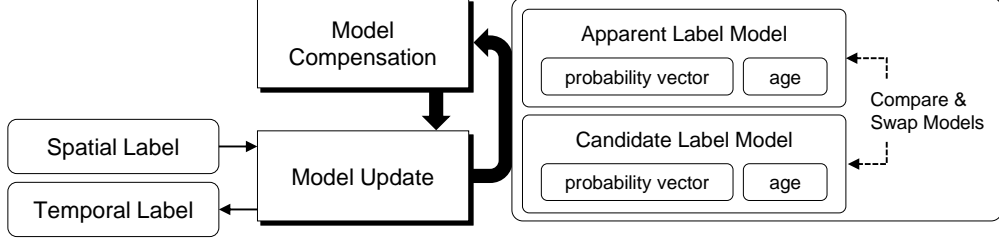


Figure 4.1: The framework of the dual-mode simple Gaussian model. Both apparent and candidate label models (“dual-mode”) are compensated and updated, iteratively.

detects label with robustness to the erroneous label.

#### 4.2.1 Update model

In the designed dual-mode simple Gaussian model,  $n$  grid have its own models, i.e., both apparent and candidate label model. We denote the probability vector of a grid  $i$  as  $P_i^{(k)}$  in the  $k$ -th frame, and the age of the pixel in the  $k$ -th frame as  $\alpha_i^{(k)}$ . Then, the probability vector and age are updated as follows:

$$P_i^{(k)} = \frac{\tilde{\alpha}_i^{(k-1)}}{\tilde{\alpha}_i^{(k-1)} + 1} \tilde{P}_i^{(k-1)} + \frac{1}{\tilde{\alpha}_i^{(k-1)} + 1} G_i^{(k)} \quad (4.4)$$

$$\alpha_i^{(k)} = \tilde{\alpha}_i^{(k-1)} + 1, \quad (4.5)$$

where  $G_i^{(k)}$  is the measured label of grid  $i$  after label matching algorithm in the  $k$ -th frame, and  $\tilde{P}_i^{(k)}$ ,  $\tilde{\alpha}_i^{(k)}$  are compensated parameters of dual-mode simple Gaussian model which will be discussed in Section 4.2.2.

Contrary to the K. Yi *et al.*'s algorithm [20] which measures pixel intensity to update models, our algorithm takes temporarily coupled label as measurements. This label might be invalid if its properly matched pair could not be found by optical flow or its flow considered too fast to be an appropriate result. Thus, we build some criteria for updating both models. First, both models are initialized with measurements directly if each age is zero. Second, an apparent label model will be updated in the case where the age is not zero and the label detected from its probability vector is equal to the measured label. In addition, a candidate label model will be updated when the age is not zero and the corresponding apparent label model is not updated. Each time model

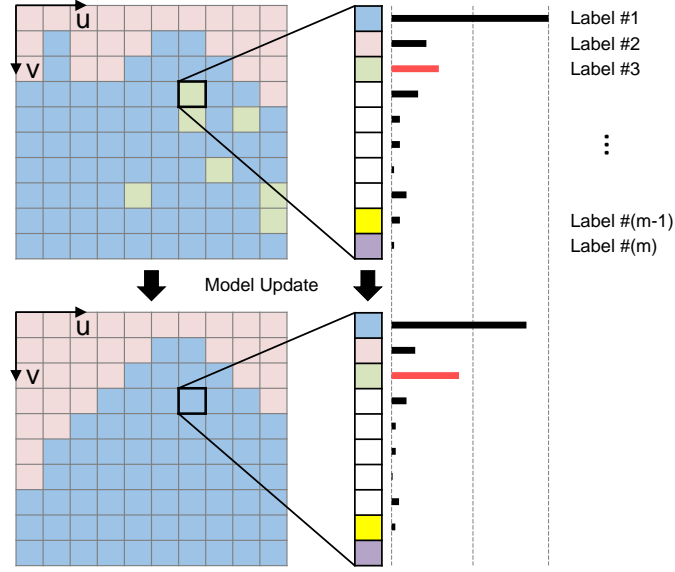


Figure 4.2: The procedure of updating a probability vector in the corresponding grid. Different labels are denoted as different colors. The probability vector makes the algorithm robust to an erroneous result of spatial segmentation based on the statistical approach.

is updated, the age increases by one until the specified maximum value. By setting the maximum value for the age of the model, we prevent the model from being insensitive to the appearance of dynamic objects because of an excessive ageing.

The proposed algorithm treats a foreground object like a static element when the object stops. This perception can be achieved by adopting a candidate model and a swapping function. For example, we could consider the case where one object with “identifier 2” moves around in the static background with “identifier 1” and after a while, the object stops. Since then, the grid belonging to the object is measured as a static element with “identifier 1”, not “identifier 2”. Thus, the corresponding apparent model is not updated whereas the candidate model be. Consequently, only the age of the candidate model increases so both models are swapped with each other when the age of the candidate model is saturated (maximum age) or larger than the one of the apparent model.

A temporal label which is the output of the temporal segmentation is obtained from the probability vector of the apparent label model with several criteria. The temporal label is updated

when the corresponding apparent label model is initialized or updated at that time. By doing so, we prevent the algorithm from prejudging the identifier of the unobserved or unmeasured grid while maintaining the previous identifier of the grid. Finally, the temporal label is extracted from the indices which indicates the maximum value in the probability vector.

### 4.2.2 Compensate model

In the previous section, we used probability vector as a property of the grid to update and classify a label of the corresponding grid. These processes for the grid model assume that each grid represents the specified point in world coordinates consistently. However, in the case of the non-stationary camera, a result of the label matching following the spatial segmentation cannot be used directly for updating the model. Thus, in order to update the grid model, we need a model compensation.

Since each model has simple parameters such as probability vector and age, we use area-weighted interpolation approach. The current grid model is proportionally compensated with models around the previous corresponding grid, which is calculated previously by optical flow in Section 2.2. For grid which has valid optical flow vector, these grid models are corrected individually. We denote a set of grids overlapped with the current grid  $i$  in the  $k$ -th frame as  $\mathbb{S}_i^{(k)}$ , weights for interpolation as  $\omega_j$ , and a region of overlap between the current grid  $i$  and the previous corresponding grid as  $R_j$  where  $j \in \mathbb{S}_i^{(k)}$  in the  $k$ -th frame. Then, the compensated probability vector and age are obtained as follows:

$$\tilde{P}_i^{(k-1)} = \sum_{j \in \mathbb{S}_i^{(k)}} \omega_j P_j^{(k-1)} \quad (4.6)$$

$$\tilde{\alpha}_i^{(k-1)} = \sum_{j \in \mathbb{S}_i^{(k)}} \omega_j \alpha_j^{(k-1)}, \quad (4.7)$$

where

$$\omega_j \propto R_j \quad (4.8)$$

and

$$\sum_j \omega_j = 1. \quad (4.9)$$

# 5

## Evaluation Results

For the evaluation of the proposed algorithm, we use the self-made dataset recorded by ASUS Xtion RGB-D camera and Vicon motion capture system. Each dataset is composed of 16-bit depth images and 8-bit RGB images with a size of  $640 \times 480$ . There are two kinds of datasets which are captured from the stationary camera or non-stationary camera. In the case of datasets using the stationary camera, we regard origin point in the world coordinates as a true location of the camera. On the other hand, for the non-stationary camera, we validate the performance of our visual odometry algorithm with measurements of Vicon system. The proposed algorithm is implemented with unoptimized MATLAB code and runs on 64bit Windows with Intel Core i7-3770@3.4GHz and 8GB memory. In this chapter, we validate the performance of our algorithm by comparing with existing motion segmentation and visual odometry algorithms.

### 5.1 Dataset

---

In the experiment, we use the five kinds of datasets as described in Table 5.1. We describe datasets in terms of travel distance, travel time, invalid depth ratio, camera position, and scene description. The travel distance and time literally mean the distance the camera moved and the total time it moved, respectively. Invalid depth ratio is the percentage of the invalid pixels among all pixels of the whole depth frame. In the foregoing table, the former two datasets were recorded by the

Table 5.1: Description of datasets for evaluating visual odometry algorithm.

Environment	Distance traveled [m]	Time traveled [s]	Invalid depth ratio [%]	Camera position	Description
Fixed Camera 1	0.0	14.00	11.38	Static	One moving object
Fixed Camera 2	0.0	19.95	14.26	Static	Two moving objects
Vicon Room 1	8.6648	26.43	10.86	Hand-held camera	Fast movement
Vicon Room 2	8.0316	50.24	11.69	Hand-held camera	Slow movement
Vicon Room 3	2.0339	23.43	23.11	Hand-held camera	Closely approach

RGB-D camera which is fixed on a certain point, and one or two objects moved around in front of the camera, respectively. On the other hand, the latter three datasets were recorded by the non-stationary camera. By looking at the travel distance and time, we can easily figure out the average speed of the camera during the period. Through the percentage of the invalid depth pixels, we can predict how the object moves along the  $z$ -axis roughly since the depth measurement range of the RGB-D camera is limited to 0.5m to 3m.

## 5.2 Motion segmentation

Fig. 5.1 shows some of internal parameters for one video sequence. In figures, different colors mean that the algorithm recognizes them as different objects. By applying motion spatial and temporal segmentation algorithm, the algorithm classifies objects which have different movement to each other, recognizes the previously observed object, and tracks the object robustly.

Fig. 5.2 shows the qualitative comparison with existing motion segmentation algorithms, MCD5.8ms [20] and randomized voting [17]. In Fig. 5.2 (a), MCD5.8ms detects moving object properly except the second frame. However, in Fig. 5.2 (b), MCD5.8ms fails to detect objects appropriately. Since MCD5.8ms algorithm uses pixel intensity to update the grid model, it is only possible to distinguish the object whose dominant color differs from one of the static background. Randomized voting can classify the object which moves independently when the object exists, but it requires the number of objects in advance. In this respect, we have preset the number of objects, 2 or 3, as a constant value in the randomized voting algorithm. As shown in both Fig. 5.2 (a) and Fig. 5.2 (b), the randomized voting algorithm always tries to divide the image into the pre-defined number of objects, 2 or 3, respectively.

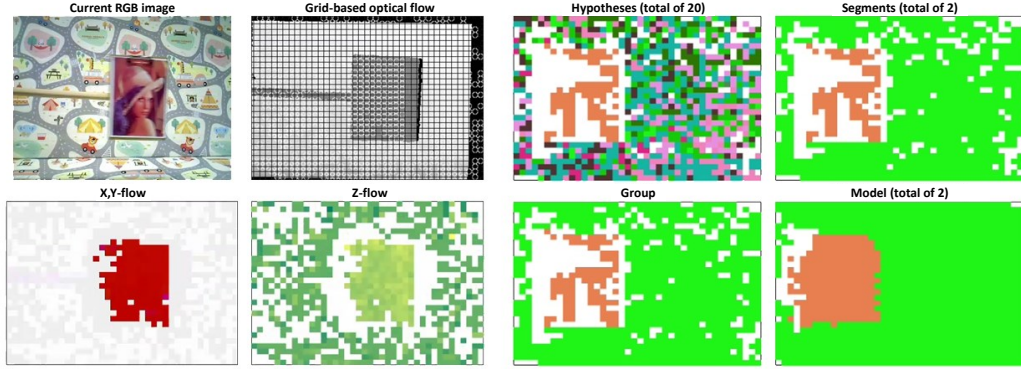


Figure 5.1: Internal parameters of the proposed motion segmentation algorithm. An object moves around as shown in the figure entitled *Current image*. *Grid-based optical flow* figure shows how the previous grid-based points move as denoted by white circle. Figure *x, y-flow* represents the magnitude and the direction of optical flow vectors as saturation and hue, respectively, whereas *z-flow* map only shows the magnitude of the *z*-directional flow, brighter as the object moves away from the camera. The figure entitled *Hypotheses* shows the result of motion spatial segmentation just before clustering motion hypotheses, and *Segments* shows the result after clustering motion hypotheses. Figure *Group* shows the result of the label matching explained in Section 4.1. Finally, figure *Model* is the output of the grid model. Note that different objects are denoted as different colors.

For the quantitative comparison, we measure recall and precision. Both measures are widely used to quantify the performance of binary classification. Recall means a chance of deciding positive among the true data, and precision means a chance of being true among the positive decisions. Table 5.2 shows a definition of true positive, true negative, false positive, and false negative. Using these definitions, recall and precision are defined as follows:

$$recall = \frac{TP}{TP + FN} \quad (5.1)$$

$$precision = \frac{TP}{TP + FP} \quad (5.2)$$

In addition, two measures are dependent on each other that means they are in inverse proportion to each other. Thus, the performance of the motion segmentation algorithm cannot be evaluated by only one measure of them, recall or precision. For this reason, F-measure [24] is a proper measure that combines recall and precision, and can be calculated by harmonic mean as in the

Table 5.2: Definition of true positive, true negative, false positive, and false negative.

Detect \ Exist	yes	no
	yes	True Positive (TP)
no	False Negative (FN)	True Negative (TN)

Table 5.3: Quantitative analysis of the proposed and existing motion segmentation algorithms.

Environment	Recall [%]			Precision [%]			F-measure		
	Proposed	MCD5.8ms	RV	Proposed	MCD5.8ms	RV	Proposed	MCD5.8ms	RV
Fixed Camera 1	<b>92.15</b>	91.32	31.41	95.30	88.05	<b>100.0</b>	<b>0.9370</b>	0.8966	0.4780
Fixed Camera 2	<b>93.15</b>	45.24	65.18	96.90	72.04	<b>99.10</b>	<b>0.9499</b>	0.5558	0.7864
Vicon Room 1	<b>86.68</b>	47.18	39.34	<b>99.46</b>	72.53	94.72	<b>0.9263</b>	0.5717	0.5559
Vicon Room 2	39.17	<b>43.15</b>	31.18	<b>100.0</b>	88.23	95.82	0.5629	<b>0.5795</b>	0.4704
Vicon Room 3	44.91	<b>53.50</b>	48.61	<b>96.11</b>	80.17	96.06	0.6121	0.6418	<b>0.6455</b>

following:

$$F\text{-measure} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} \quad (5.3)$$

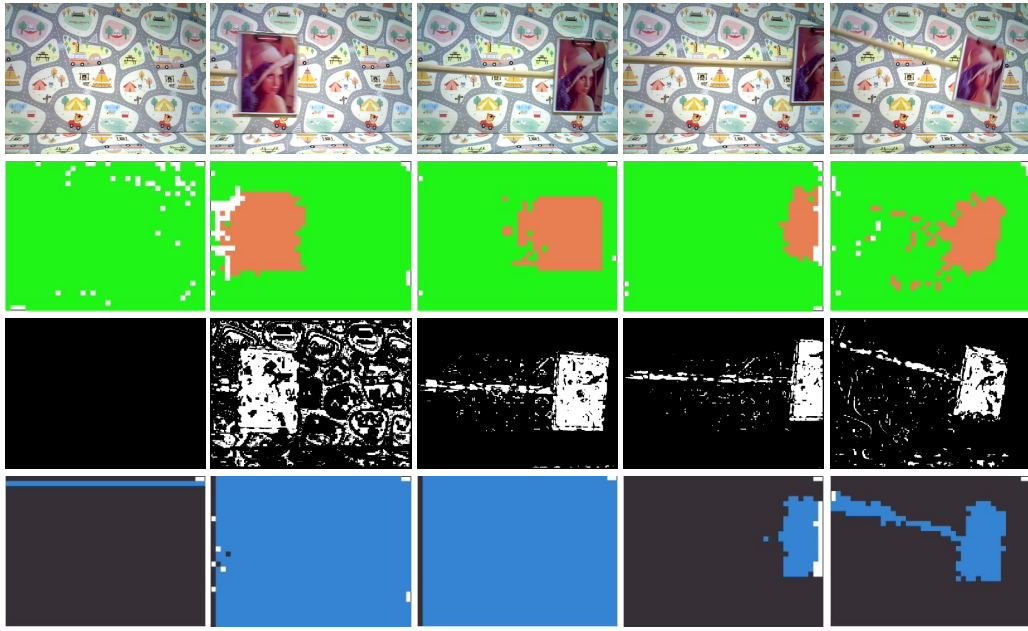
Above recall and precision are calculated in an entire video sequence.

Table 5.3 shows the quantitative results of motion segmentation algorithms. Because our algorithm and randomized voting separate out objects from the static background based on the temporary movement, they judge a static object that moved before to be static parts. In the evaluation, we regard this result as true positive. Additionally, in the case of randomized voting, it fails to segment image for lack of existing objects as mentioned before. Nevertheless, we count the 1st to 3rd frames of 4th row in Fig. 5.2 (a) as the true positive, still regarding 5th frame of 4th row in Fig. 5.2 (b) as the false negative. The proposed algorithm shows relatively high recall values in the *Fixed Camera* datasets and is second place for the precision test. For the F-measure combined recall and precision, our algorithm shows outstanding results compared to other algorithms. For the *Vicon Room* datasets, whereas, the first remarkable thing is that the precision is improved slightly as the moving object in the *Vicon Room* repeats appearing and disappearing less than one of *Fixed Camera*. Also, we can notice that the performance of the motion segmentation algorithms deteriorates when the speed of the camera and the object are

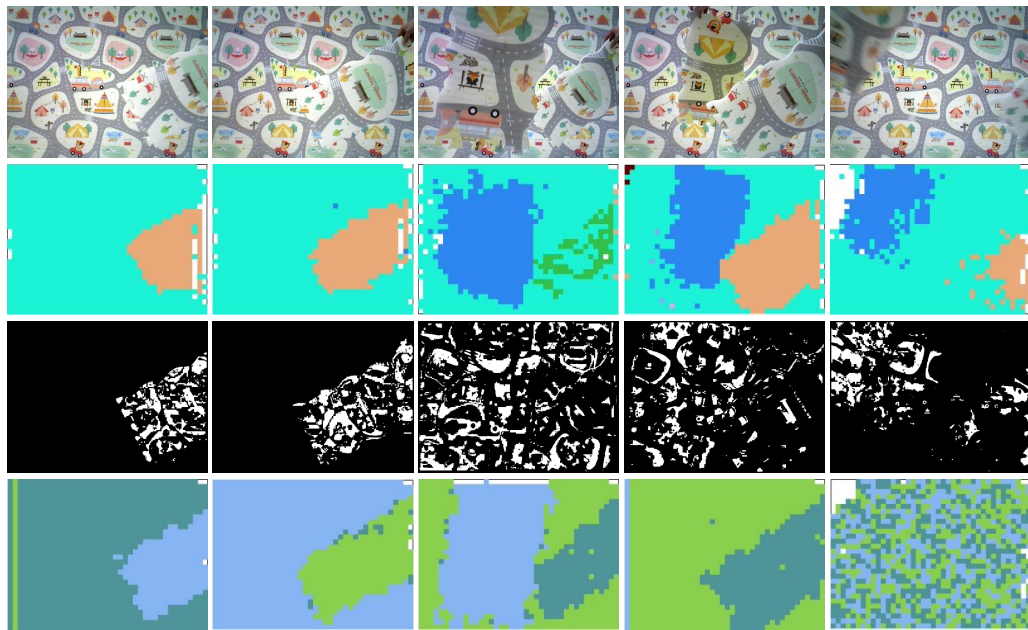


slow, or the object approaches the camera closely.

With regard to implementation details, we use consistent value of parameters to retain objectivity. Especially, the number of hypotheses  $H$  is 20, and the size of the grid is  $16 \times 16$  in pixels; and these are critical parameters which improve the performance or increase the computational load.



(a) Fixed camera 1



(b) Fixed camera 2

Figure 5.2: Qualitative analysis of motion segmentation algorithms. From top to bottom: original image, our segmentation, MCD5.8ms, and randomized voting. The proposed algorithm shows different motions in different colors, and MCD5.8ms detects dynamic pixels. On the other hand, randomized voting algorithm only divides into several parts which have different motions to each other, so label of the grid can be easily changed.

### 5.3 Visual odometry

---

For the quantitative comparison between the proposed algorithm and the current state-of-the-art visual odometry algorithms, we use relative pose error (RPE) and absolute trajectory error (ATE) [3]. RPE and ATE mean how exact the estimated motion is and how exact the trajectory is, respectively. For a sequence of  $n$  camera poses, we denote time interval as  $\Delta$ , the ground truth pose as  $\mathbf{Q}_1, \dots, \mathbf{Q}_n \in SE(3)$ , and the estimated pose as  $\mathbf{P}_1, \dots, \mathbf{P}_n \in SE(3)$ . Then, RPE is defined as follows:

$$RPE := \left( \frac{1}{m} \sum_{i=1}^m \|\text{trans}(\mathbf{E}_i)\|^2 \right)^{\frac{1}{2}}, \quad (5.4)$$

where  $m = n - \Delta$ ,  $\|\cdot\|$  represents the 2-norm,  $\text{trans}(\mathbf{E}_i)$  is the translational components of  $\mathbf{E}_i$ , and

$$\mathbf{E}_i := (\mathbf{Q}_i^{-1} \mathbf{Q}_{i+\Delta})^{-1} (\mathbf{P}_i^{-1} \mathbf{P}_{i+\Delta}). \quad (5.5)$$

Similarly, ATE is defined as follows:

$$ATE := \left( \frac{1}{n} \sum_{i=1}^n \|\text{trans}(\mathbf{F}_i)\|^2 \right)^{\frac{1}{2}}, \quad (5.6)$$

where

$$\mathbf{F}_i := \mathbf{Q}_i^{-1} \mathbf{S} \mathbf{P}_i \quad (5.7)$$

and  $\mathbf{S}$  is rigid transformation matrix from the camera coordinates to the world coordinates. Therefore, for the precisely estimated pose  $\mathbf{P}_i$ , the below equation is satisfied:

$$\mathbf{S} \mathbf{P}_i \approx \mathbf{Q}_i. \quad (5.8)$$

Figs. 5.3 to 5.7 show the evaluation results for each sequence in datasets. The proposed algorithm is compared with the current state-of-the-art visual odometry algorithms, which are DVO (direct dense method) [7], ORB-SLAM2 (indirect sparse method) [9], and DSO (direct sparse method) [10]. In order to verify the performance as a visual odometry, we further test modified version of ORB-SLAM, which is unable to detect and correct loop closure. We refer to this modified ORB-SLAM as ORB-VO. In Fig. 5.3, algorithms show outstanding performance except for DVO. Because DVO is based on direct dense method, it performs optimization process across all pixels, thereby it is seriously influenced by dynamic elements even if small object. Especially,

Table 5.4: Evaluation of visual odometry algorithms.

Environment	Relative Pose Error [m]					Absolute Trajectory Error [m]				
	Proposed	DVO	ORB-SLAM2	ORB-VO	DSO	Proposed	DVO	ORB-SLAM2	ORB-VO	DSO
Fixed Camera 1	0.00156	0.60026	<b>0.00055</b>	0.00082	0.00166	0.00525	0.67494	<b>0.00050</b>	0.00060	0.00117
Fixed Camera 2	<b>0.01384</b>	0.16728	0.05664	0.07248	0.15687	<b>0.04269</b>	0.46060	0.04666	0.20831	0.45773
Vicon Room 1	<b>0.04941</b>	0.61796	0.53376	0.52792	0.22635	<b>0.11287</b>	0.85412	0.93911	0.94335	0.27141
Vicon Room 2	<b>0.09008</b>	0.30968	0.23901	0.23811	0.10167	<b>0.37233</b>	0.77860	0.73921	0.64159	0.45613
Vicon Room 3	<b>0.03852</b>	0.31036	0.11864	0.15391	0.11581	<b>0.06791</b>	0.70322	0.17707	0.31355	0.48000

ORB-SLAM shows the best performance among the comparative group in virtue of loop closure. Next, when there appear two moving objects, the performance of our algorithm is superior to one of other algorithms in terms of RPE and ATE as shown in Fig. 5.4 and Table 5.4. At around 10 seconds, two objects appear and occupy more than half of the image, and this causes other algorithms except ours to lose their pose.

In Figs. 5.5 to 5.7, a measurement of Vicon system is denoted as black solid line. Each of the three sequences was recorded in situations of fast movement and slow movement of the camera, and closely approach of the dynamic object as depicted in Table 5.1. Our algorithm shows superior performance compared to other algorithms. For your information, ORB-SLAM and ORB-VO show a tendency to track features of a dynamic object and DSO undergoes a scale-drift problem frequently if the dynamic object is observed for a long time.

Since the motion segmentation and the estimation parts of our algorithm are not strongly coupled with each other, it is possible for existing visual odometry algorithm to combine with the proposed motion segmentation to improve their robustness in dynamic environments. As shown in Fig. 5.8, we test the combination of existing visual odometry and our motion segmentation. We can see that all three modified algorithms, Proposed $\times$ {DVO, ORB-SLAM2, ORB-VO}, show much better performance compared to the original versions by combining with our motion segmentation.

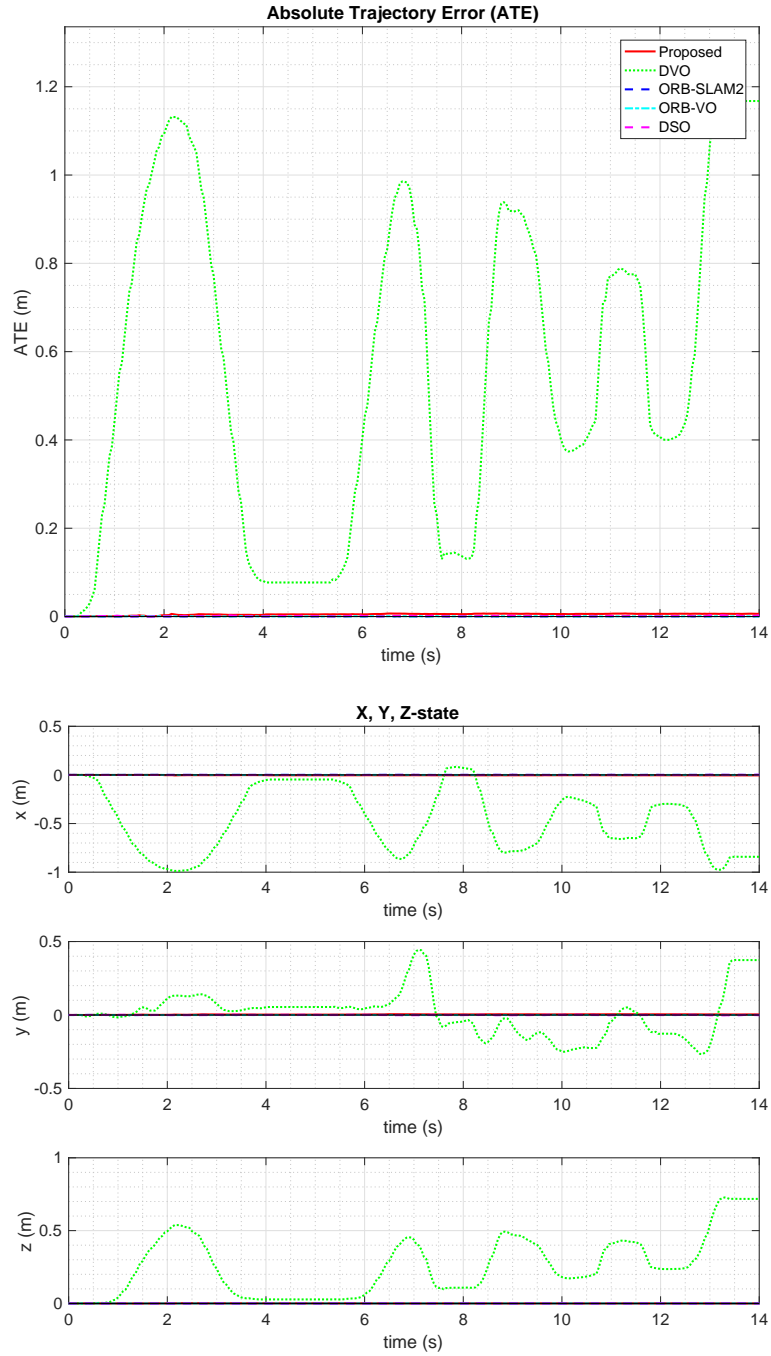


Figure 5.3: Absolute trajectory error and XYZ positions for *Fixed Camera 1* dataset. Absolute trajectory error is defined in Eq. (5.6). XYZ positions denotes the estimated position of the fixed camera in each  $x, y, z$ -axis, and true positions of the camera are consistently zeros.

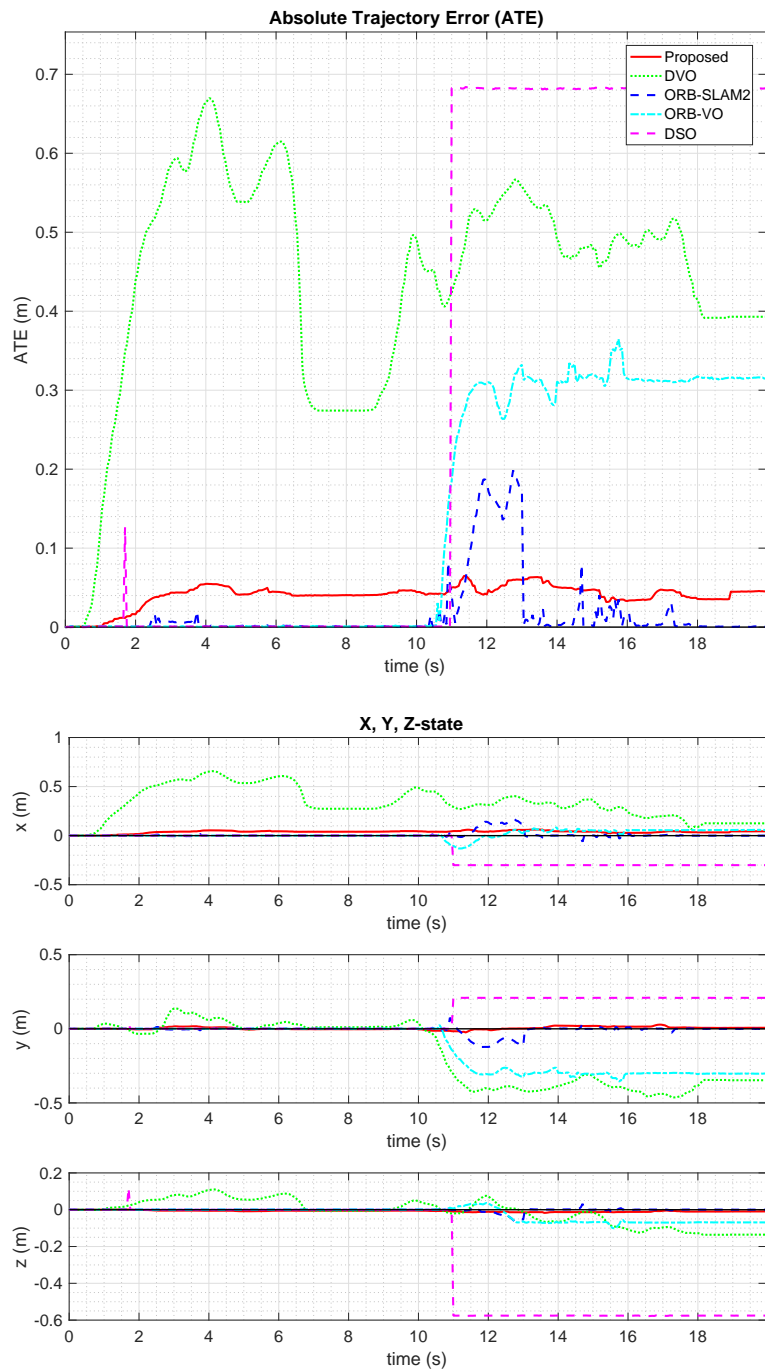


Figure 5.4: Absolute trajectory error and XYZ positions for *Fixed Camera 2* dataset. Please be informed that two objects appear around 10 seconds.

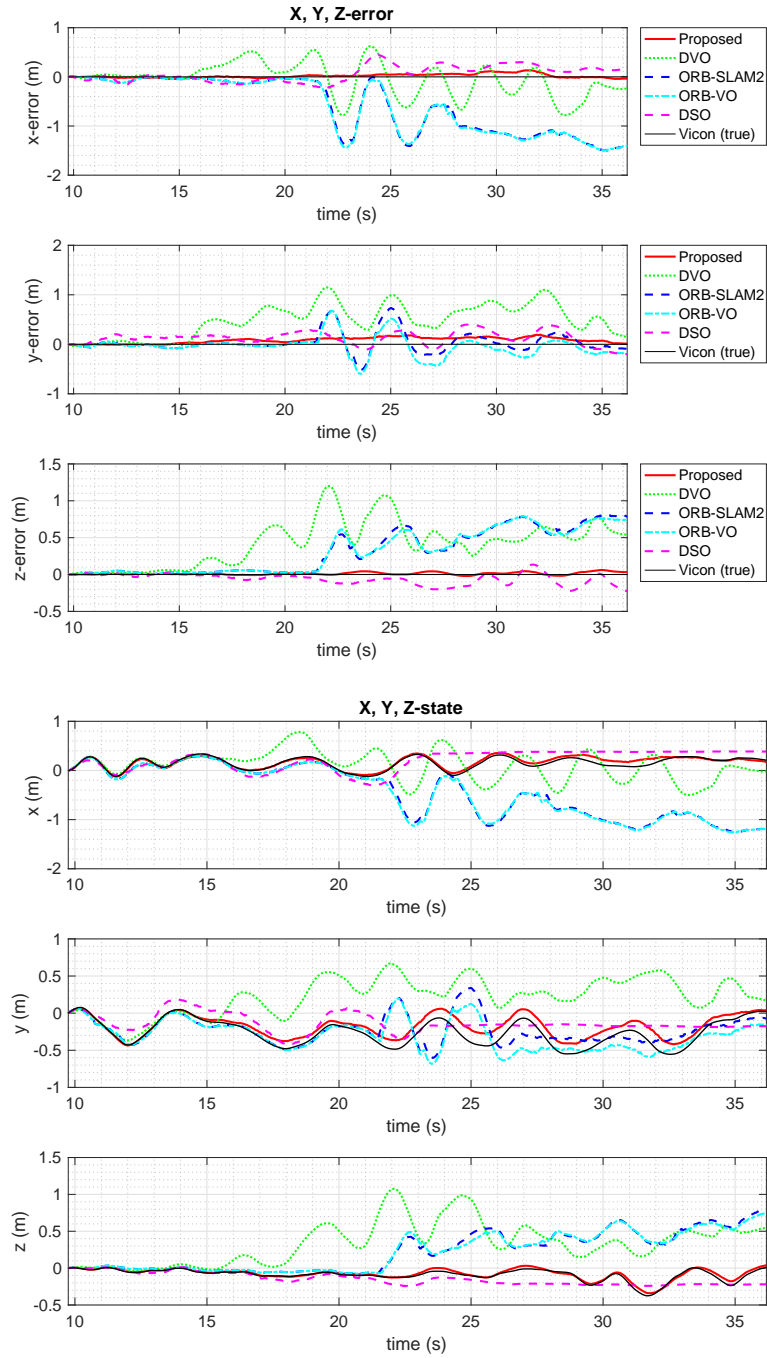


Figure 5.5: XYZ position errors and XYZ positions for *Vicon Room 1* dataset. XYZ errors mean the difference between the estimated position and the true in each  $x, y, z$ -axis, and true values are denoted as black solid line.

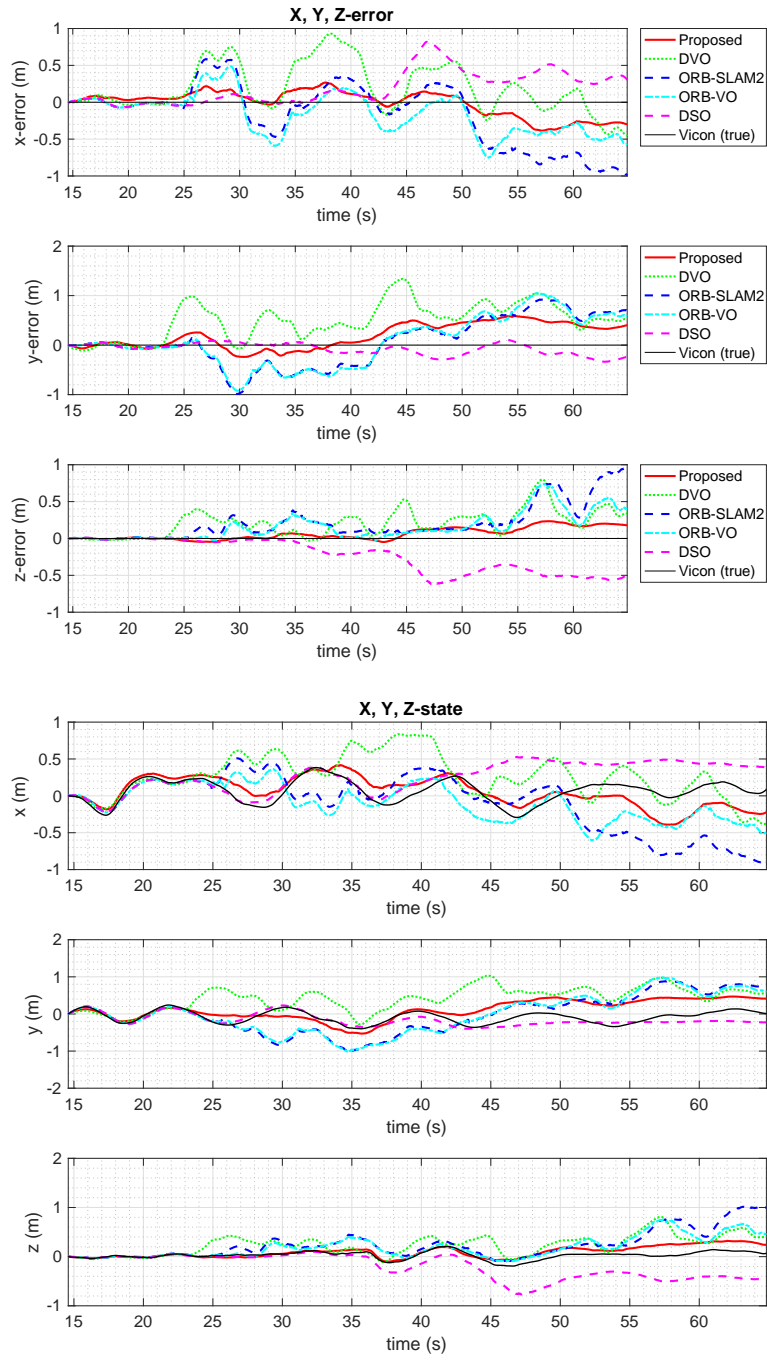


Figure 5.6: XYZ position errors and XYZ positions for *Vicon Room 2* dataset.



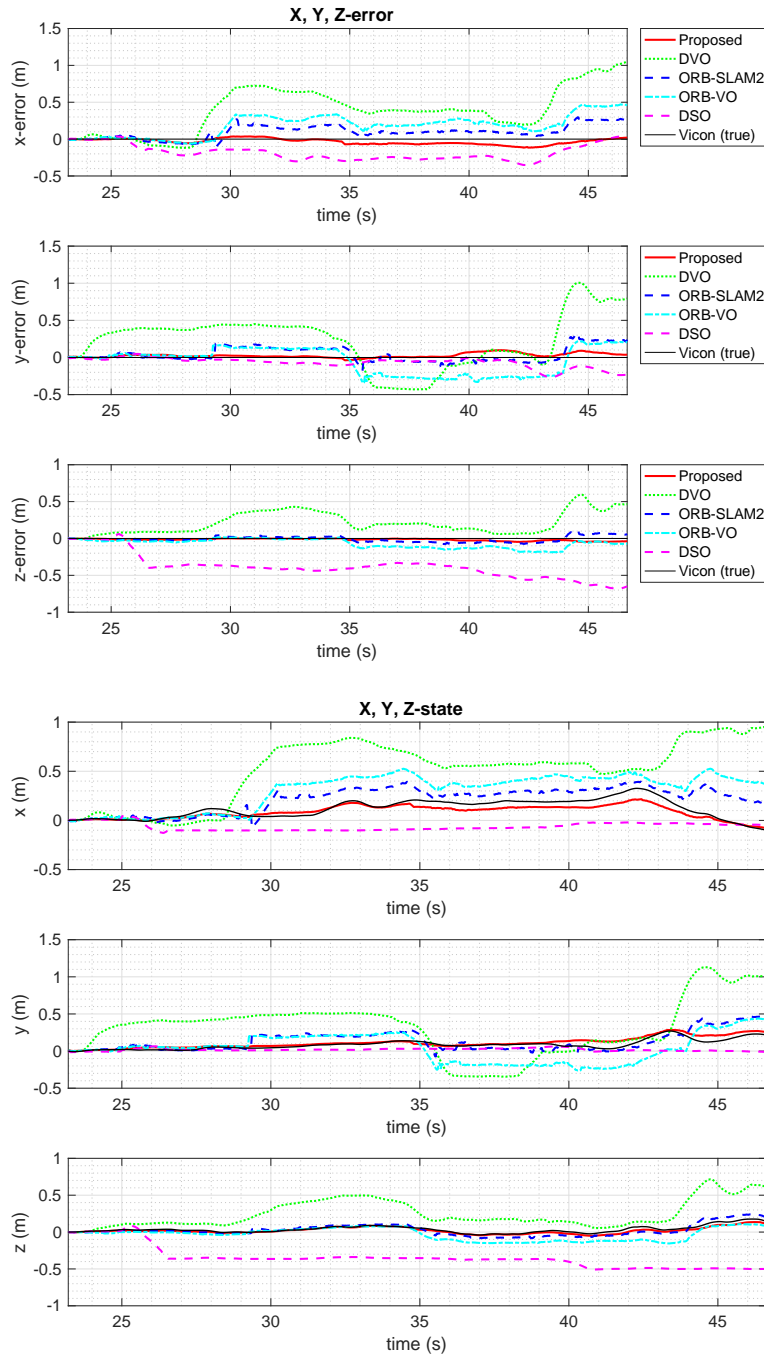


Figure 5.7: XYZ position errors and XYZ positions for *Vicon Room 3* dataset.

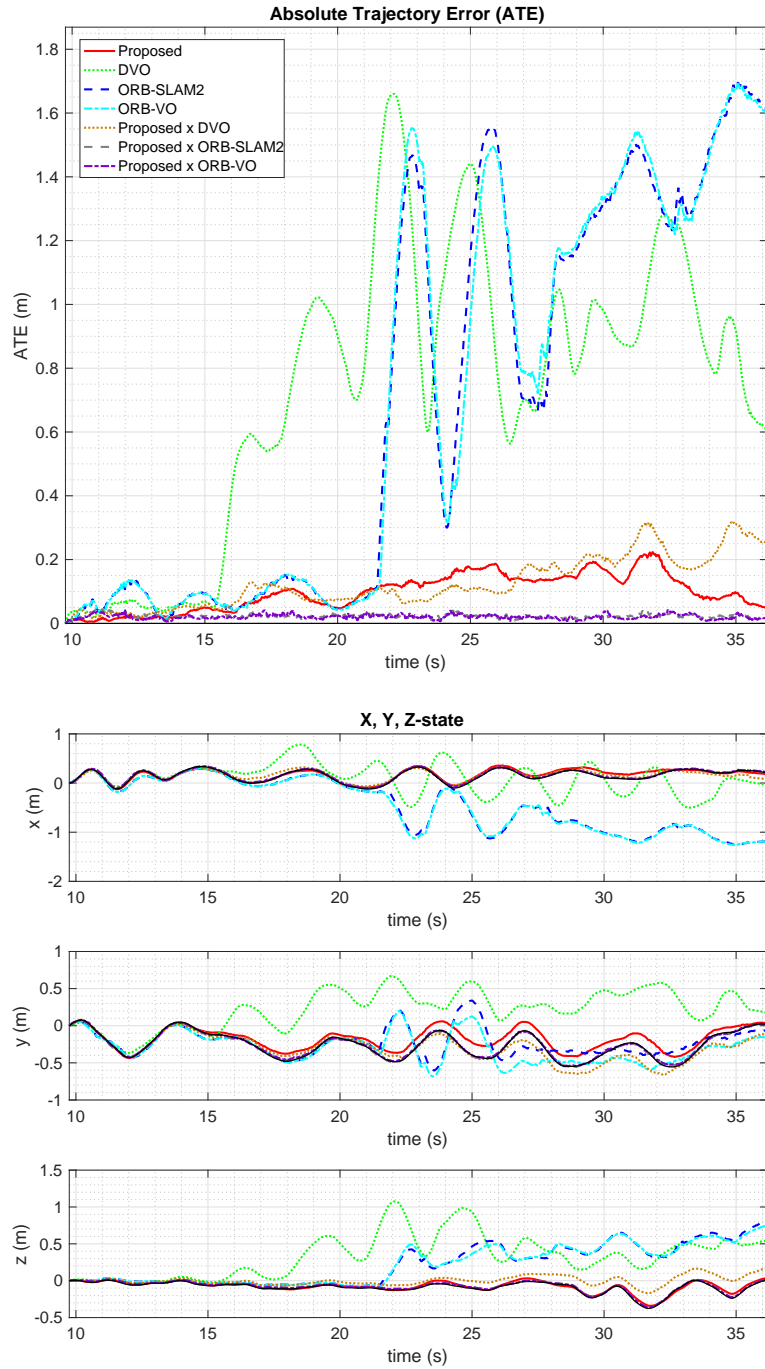


Figure 5.8: XYZ position errors and XYZ positions for *Vicom Room 1* dataset. Proposed $\times$ {DVO, ORB-SLAM2, and ORB-VO} estimate the ego-motion with notable accuracy through combining with the proposed motion segmentation. For algorithms which are based on the same visual odometry technique, they denoted as the same line style.

# 6

## Conclusion

In this paper, we proposed a robust visual odometry algorithm via rigid motion segmentation using grid-based optical flow. The proposed algorithm is considerably more robust than the current state-of-the-art visual odometry algorithms while showing high accuracy. For robustness, the proposed spatial motion segmentation uses the three-dimensional optical flow vectors to generate and search distinct motions with no prior information such as the shape of the number of objects. Besides, temporal segmentation initializes and updates a dual-mode simple Gaussian model of the grid so that our algorithm differentiates the static background and dynamic objects robustly. Finally, the ego-motion is estimated by the use of optical flows belonging to the static background segment. The additional benefit of the proposed algorithm is that it can be combined with existing visual odometry algorithms to improve their robustness in dynamic environments. Further, it can be used as a part of an efficient dynamic obstacle avoidance algorithm by using the kinematic information of moving objects.

# References

- [1] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. I–652.
- [2] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial],” *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.
- [3] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 573–580.
- [4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [5] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, “Dense semi-rigid scene flow estimation from rgb-d images,” in *European Conference on Computer Vision*. Springer, 2014, pp. 567–582.
- [6] M. Jaimez, M. Souiai, J. Stückler, J. Gonzalez-Jimenez, and D. Cremers, “Motion cooperation: Smooth piece-wise rigid scene flow from rgb-d images,” in *3D Vision (3DV), 2015 International Conference on*. IEEE, 2015, pp. 64–72.
- [7] C. Kerl, J. Sturm, and D. Cremers, “Dense visual slam for rgb-d cameras,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2100–2106.
- [8] F. Steinbrücker, J. Sturm, and D. Cremers, “Real-time visual odometry from dense rgb-d images,” in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 719–722.
- [9] R. Mur-Artal and J. D. Tardos, “Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras,” *arXiv preprint arXiv:1610.06475*, 2016.

- [10] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [11] B. Kitt, F. Moosmann, and C. Stiller, “Moving on to dynamic environments: Visual odometry using feature classification,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5551–5556.
- [12] R. Sabzevari and D. Scaramuzza, “Monocular simultaneous multi-body motion segmentation and reconstruction from perspective views,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 23–30.
- [13] —, “Multi-body motion estimation from monocular vehicle-mounted cameras,” *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 638–651, 2016.
- [14] D.-H. Kim and J.-H. Kim, “Effective background model-based rgb-d dense visual odometry in a dynamic environment,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.
- [15] A. Dib and F. Charpillet, “Robust dense visual odometry for rgb-d cameras in a dynamic environment,” in *Advanced Robotics (ICAR), 2015 International Conference on*. IEEE, 2015, pp. 1–7.
- [16] S.-J. Jung, J.-B. Song, and S.-C. Kang, “Stereo vision-based visual odometry using robust visual feature in dynamic environment,” *The Journal of Korea Robotics Society*, vol. 3, no. 4, pp. 263–269, 2008.
- [17] H. Jung, J. Ju, and J. Kim, “Rigid motion segmentation using randomized voting,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1210–1217.
- [18] L. Zappella, E. Provenzi, X. Lladó, and J. Salvi, “Adaptive motion segmentation algorithm based on the principal angles configuration,” *Computer Vision—ACCV 2010*, pp. 15–26, 2011.
- [19] E. Elhamifar and R. Vidal, “Sparse subspace clustering,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2790–2797.

- [20] K. Moo Yi, K. Yun, S. Wan Kim, H. Jin Chang, and J. Young Choi, “Detection of moving objects with non-stationary cameras in 5.8 ms: Bringing motion detection to your mobile device,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 27–34.
- [21] B. D. Lucas, T. Kanade *et al.*, “An iterative image registration technique with an application to stereo vision,” 1981.
- [22] K. Yamaguchi, “Mexopencv,” *Collection and a development kit of matlab mex functions for OpenCV library, available at <http://www.cs.stonybrook.edu/~kyamagu/mexopencv>*, 2013.
- [23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [24] D. M. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” 2011.

# 국 문 초 록

기존 대다수의 영상 항법 알고리즘은 정적인 환경을 가정하여 개발되어 왔으며, 잘 정의된 데이터셋에서 성능이 검증되어 왔다. 하지만 무인 로봇이 영상 항법을 활용하여 임무를 수행하여야 하는 장소는, 실제 사람이나 차량이 왕래하는 등 동적인 환경일 가능성이 크다. 비록 RANSAC을 활용하여 영상 항법을 수행하는 일부 알고리즘들은 프레임 내의 비정상적인 움직임을 위치 추정 과정에서 배제할 수 있지만, 이는 동적 물체가 영상 프레임의 작은 부분을 차지하는 경우에만 적용이 가능하다. 따라서 불확실성이 존재하는 동적 환경에서 자기 위치를 강인하게 추정하기 위해, 본 논문에서는 동적 환경에 강인한 영상 기반 주행 기록계 알고리즘을 제안한다. 제안한 알고리즘은 원활한 수행 속도와 이미지 내에 균일하게 분포된 모션을 계산하기 위해, 격자 기반 옵티컬 플로우를 이용한다. 그리고 격자 단위 그리드의 모션을 통해 단일 프레임 내에서 3차원 공간 모션 분할을 수행하고, 다수의 동적 물체 및 정적 요소를 지속적으로 구분 및 구별하기 위해 시간적 모션 분할을 수행한다. 특히 지속적으로 동적 및 정적 요소를 구별하기 위해, 우리는 이미지 내의 각 그리드에 이중 모드 가우시안 모델을 적용하여 알고리즘이 공간적 모션 분할의 일시적 노이즈에 강인하게 하고, 확률 벡터를 구성하여 그리드가 서로 구별되는 각각의 요소로 발현할 확률을 계산하게 한다. 개발한 알고리즘의 성능 검증을 위해 ASUS Xtion RGB-D 카메라와 Vicon 모션 캡처 시스템을 통해 구성한 데이터셋을 이용하였으며, 기존 모션 분할 알고리즘과의 재현율 (recall), 정밀도 (precision) 비교 및 기존 영상 기반 주행 기록계 알고리즘과의 추정 오차 비교를 통해 타 알고리즘 대비 우수한 모션 검출 및 위치 추정 성능을 확인하였다.

주요어 : 영상 항법, 동적 환경, 모션 분할, 격자 기반 옵티컬 플로우, 깊이 카메라.

학번 : 2015-20784