DOCTORAL THESIS

Change Detection of 3D Scene with 3D and 2D Information for Environment Checking

(3次元と2次元の情報を用いた環境検査のための3次元変化検出手法)

Author:
Baowei Lin (林 宝尉)

Supervisor:
Dr. Toru Tamaki
Dr. Kazufumi Kaneda
Dr. Takio Kurita
Dr. Koichi Harada
Dr. Koji Ichii

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Intelligent System and Modeling Lab.
Department of Information Engineering
Graduate School of Engineering
HIROSHIMA UNIVERSITY

September 2013

Declaration of Authorship

“Education is a admirable thing, but it is well to remember from time to time that nothing worth knowing can be taught.”

Oscar Wilde
British dramatist

This thesis is dedicated to my parents.
For their endless love, support and encouragement.

Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude to my supervisor Dr. Toru Tamaki, whose sincerity and encouragement I will never forget. Dr. Toru Tamaki has been my inspiration as I hurdle all the obstacles in the completion of my PhD life.

Also, my sincere gratitude to my sub-supervisors, Dr. Kazufumi Kaneda, Dr. Takio Kurita, Dr. Koichi Harada and Dr. Koji Ichii, who provide me many useful and important suggestions on my research and this dissertation. Their unselfish and unfailing support will help me all my research life.

And then, my sincere gratitude to Dr. Tsukasa Hirashima and Dr. Bisser Raytchev, who let me know what research is and always give me help in technical paper writing. My gratitude to my laboratory members. Thanks Dr. Marcos Slomp and Mr. Kouhei Sakai for sharing variant new idears with me. Thanks Mr. Bingzhi Yuan for the discussion of my research and the contribution of section 4.2.2. Thanks Mr. Yuji Ueno for helping me of my researches in early stage. Also, thanks Mr. Haoming Wang for creating two figures for making the introduction chapter.

I also would like to thank the open source initiatives of **PCL** (Point Cloud Library)[1], **OpenCV** (Open Source Computer Vision)[2], **Eigen** (Linear Algebra Library)[3], **ANN** (Approximate Nearest Neighbor Searching Library)[4], **rply** (Read/Write PLY Files Library)[5], **siftGPU** (A GPU Implementation of Scale Invariant Feature Transform)[6], **boost**[7], **KDevelop**[8], **Meshlab**[9], **OpenOffice**[10], **MiKTeX**[11] and **TeXnicCenter**[12] for their generosity and excellence in their open source contributions that accelerated the development of the research projects contained in this thesis, as well as the writing of this document.

Last but not the least, my family and my friends, for all your kindness understanding and supports, I have confidence to go abroad and finish my study. Thank you very much.

[1]http://pointclouds.org/
[2]http://opencv.willowgarage.com/wiki/
[3]http://eigen.tuxfamily.org
[4]http://www.cs.umd.edu/~mount/ANN/
[5]http://w3.impa.br/~diego/software/rply/
[6]http://cs.unc.edu/~ccwu/siftgpu/
[7]http://www.boost.org/
[8]http://www.kdevelop.org/
[9]http://meshlab.sourceforge.net/
[10]http://www.openoffice.org/
[11]http://miktex.org/
[12]http://www.texniccenter.org/

Abstract

Recent surveillance technology developments motivate a regular observation of places such as coast, slopes and highways, where disasters and accidents can happen with high probability. Hence, any obvious change must be alerted. However, this is not practical for a wide area. Therefore, an automatic detection of unusual change seems to be useful and important.

In this thesis, we describe some methods which divided into online and offline approaches to detect temporal change. The online step is a 3D-2D detection using a 3D scene geometry at time A (reference) and the image of the same scene at time B (query or test). The offline step is a 3D-3D registration of 3D scenes for different times which could provide more accurate results.

To quickly detect changes and visualize the change taking place for operators at regular observation, our online step method uses the combination of 3D-2D and 2D-2D matching. We assume that a 3D scene geometry is given but only at time A as a reference 3D scene, hence no 3D range finder is necessary. At time B (query), a hand-held camera is used to take images of the same place, and the reference 3D geometry is used for 3D-2D matching for camera pose estimation. To find a region of change in the query images, first, the nearest image of a query image is selected from the training images. Then, the matching between the query image and the nearest images is computed for finding a set of non-matching points as a region of change. These change regions are visualized by projecting 3D points back only to those regions. Experimental results show that our method can detect change areas correctly.

In order to perform 3D-2D matching for camera pose estimation, we represent the 3D keypoints by their corresponding 2D keypoints. A 3D keypoint is assumed to have corresponding 2D keypoints in two or more of the training images. These 2D keypoints have 2D feature descriptors that can be used to characterize the 3D keypoints. Therefore, we use the 2D feature descriptors as a feature descriptor of the corresponding 3D keypoint. After detecting the 3D keypoints, we perform feature matching and register the newly taken query images. Experimental results with 3D point clouds of indoor and outdoor scenes show that the extracted 3D keypoints can be used for matching with 2D keypoints in query images.

Once we find the change areas from the 3D-2D method, what we need to do is to improve the accuracy of the detection result. We propose a method to register 3D point clouds offline. The alignment result could provide the users much better result than just using image based rough detection result. Our offline method is divided into two methods. The first method estimates the scale of each point cloud separately: each point cloud has its own scale that is something like the size of a scene. We call it a keyscale, which is a representative scale and is defined for a given 3D point cloud as the minimum of the cumulative contribution rates of PCA of descriptors over different scales. Our second method directly estimates the ratio of scales (scale ratio) of two point clouds. Instead of finding the minimum, this approach registers the two sets of curves of the cumulative contribution rate of PCA by assuming that those differ only in scale. Experimental results with simulated and real scene point clouds demonstrate that the scale alignment of 3D point clouds can be effectively accomplished by our scale ratio estimation.

Keywords: Change Detection, Surveillance System, Online Approach, Offline Approach, 3D-2D, 3D-3D, Feature Matching, Camera Pose Estimation, Scale Ratio ICP

Contents

List of Figures

List of Tables

Abbreviations

PCB	**P**rinted **C**ircuit **B**oard
SfM	**S**tructure **f**rom **M**otion
NARF	**N**ormal **A**ligned **R**adial **F**eature
PCA	**P**rincipal **C**omponent **A**nalysis
ICP	**I**terative **C**losest **P**oint
RANSAC	**RAN**dom **SA**mple **C**onsensus
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
DoG	**D**ifference **o**f **G**aussians
SVD	**S**ingular **V**alue **D**ecomposition
PMVS2	**P**atch-based **M**ulti **V**iew **S**tereo2
GMM	**G**aussian **M**ixture **M**odel
MRF	**M**arkov **R**andom **F**ield
ROC	**R**eceiver **O**perating **C**haracteristic
TP	**T**rue **P**ositive
FP	**F**alse **P**ositive
TN	**T**rue **N**egtive
FN	**F**alse **N**egtive
FOV	**F**ields **o**f **V**iew
TOF	**T**ime **o**f **F**light

Symbols

$\mathbb{R}$	the set of all real numbers
$\mathbb{R}^n$	the n-dimenstional real linear space
$\mathbb{E}^n$	the n-dimenstional Euclidean space
$\mathbf{p} \in E^3$	a point in space
C	point cloud
C_{key}	set of 3D keypoints
$\boldsymbol{X^i}(i = 1, 2, 3, \ldots, n)$	3D point
$\boldsymbol{Y^i}(i = 1, 2, 3, \ldots, n)$	3D point
$\bar{\boldsymbol{X}}$	center point of X
$\bar{\boldsymbol{Y}}$	center point of Y
X'	deviation of X
Y'	deviation of Y
$I_j(j = 1, 2, 3, \ldots, m)$	training images
$\boldsymbol{x_j^i}$	2D points: the ith point in jth image
$S(\boldsymbol{x_j^i})$	feature of 2D keypoint
$S(\boldsymbol{X^i})$	feature of 3D keypoint
$g = (R, \boldsymbol{t})$	rigid body transformation
R	rotation matrix
$\boldsymbol{t}$	translation vector
P	projection matrix for 2D image
$\boldsymbol{n}$	normal vector
$\boldsymbol{c}$	viewing vector of a camera
th_v	threshold of visible image when select 3D points which have corresponding 2D points
th_d	threshold of distance (distance unit: pixel)
n_1^i	numbers of training images in which a 3D point is projected

n_2^i	numbers of images in which 3D points appear as 2D keypoints
$K \in R^{3\times3}$	the camera calibration matrix
$\Pi_0 \in R^{3\times4}$	the standard projection matrix $[I, 0]$ from R^3 to R^2
$e(R_1, R_2)$	the distance of rotation matrices R_1 and R_2
$\boldsymbol{S_i}$	spin iamge
w	spin image width
(α, β)	2D coordinate in spin image
c_d^w	contribution rate for keyscale, d: dimension of spin image vector, w: width of spin image
t	scale ratio of two point clouds
$\boldsymbol{Spin_i}(\alpha, \beta, w)$	spin image with parameters
$\mathrm{tr}(M)$	the trace of a square matrix M
$\det(M)$	the determinant of a square matrix M
f	camera focus length
λ	a scalar real number
I	identity matrix

Chapter 1

Introduction

Recent surveillance technology developments motivate a regular observation of places such as coast, slopes and highways, where disasters and accidents can happen with high probability (Figure 1.1). At coastal areas, a lot of wave dissipating blocks are placed to decrease the power of the waves. If the configuration of the blocks is altered due to erosion, the blocks might collapse which would diminish their effectiveness. A mountain slope with a face exposed after a heavy rain might cause a landslide. Hence, any obvious change must be real-time alerted. The filling beneath a highway can collapse due to earthquakes or aging, which can result in cracks and subsidence. Such places should

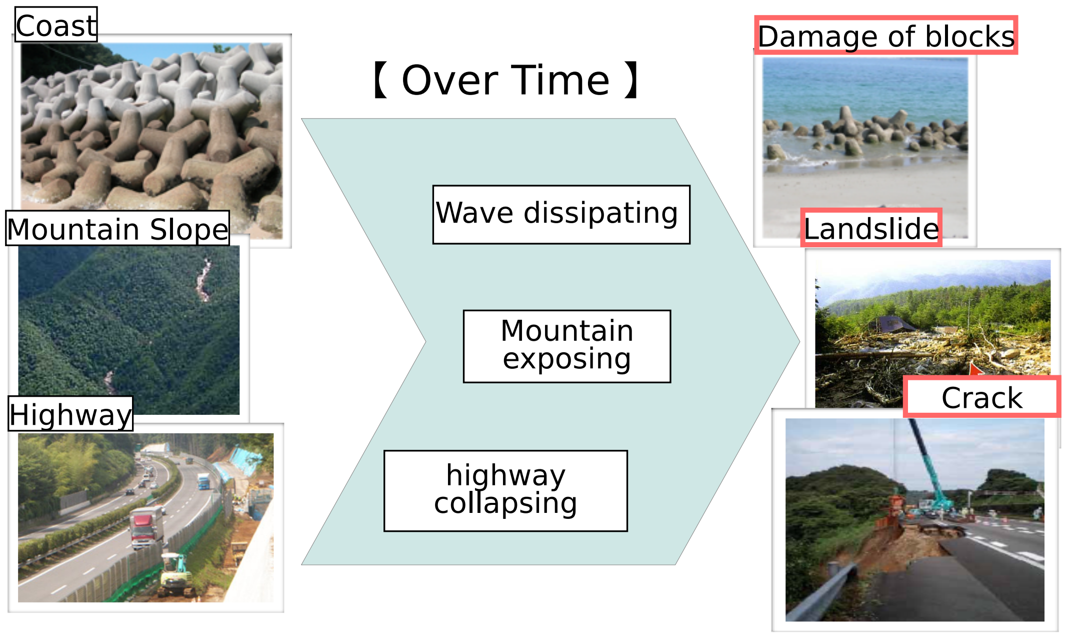


FIGURE 1.1: Surveillance research targets.

be observed all the time, however, this is not practical for a wide area. Therefore, a real-time automatic detection of unusual change seems to be useful and important.

Surveillance technologies are very popular and useful for monitoring of behavior, activities, or other information of the environment, and surveillance systems are widely installed by governments, factories, schools and so on. However, surveillance systems can NOT be easily used in all fields. For example, in our research target, we want to alert any obvious change in a wide area. So we need a surveillance system with follow functions:

- It is a change detection system in a wide area.
- It does not request static cameras.
- It could provide real-time change area.
- It could ensure the accuracy of detection result.

Actually, they are very challenging and difficult tasks, therefore, we analyze the possible approaches of surveillance systems related to the task of change detection and categorize them into three.

The first one is a simple way which monitors some places by cameras and detects security problems by human operators. Such systems have been widely used in market malls or parking areas over the last decades. It doesn't request high computational cost. However, the hand-held cameras can not be used and the detection accuracy is not good.

The second one is an online real-time client-server systems. Oberti et al. [1] proposed a combination surveillance system to detect change with a mobile camera. In the client, the system captures and processes some images. In the server, the system analyzes the images and returns the detection results to the client. This kind of systems are more intelligent than the first kind of surveillance systems and can reduce the heavy task of human operators. But there is a problem that the detection results are not accurate.

The third one is an offline surveillance system. Change results are detected in a long term offline time, therefore, high accurate detection results could be provided. However, powerful equipments and long processing time are necessary.

In this thesis, we design systems based on the second and the third kinds of surveillance systems. The second kind of surveillance system, online real-time client-server system, can be used in our particularly environment. Besides, hand-held camera used real-time change detection could be performed. Then the third kind of surveillance system, offline system, is used for ensuring accuracy of detection results. In other words, we divide our research goal into two sub-goals:

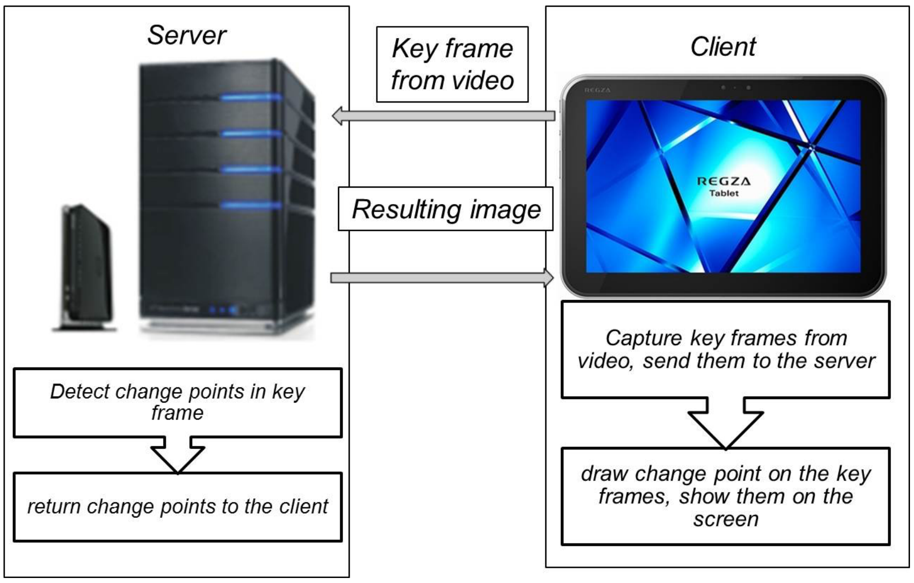

Server
Key frame from video
Client
Resulting image
Capture key frames from video, send them to the server
Detect change points in key frame
return change points to the client
draw change point on the key frames, show them on the screen

FIGURE 1.2: Module overview of the online client-server system.

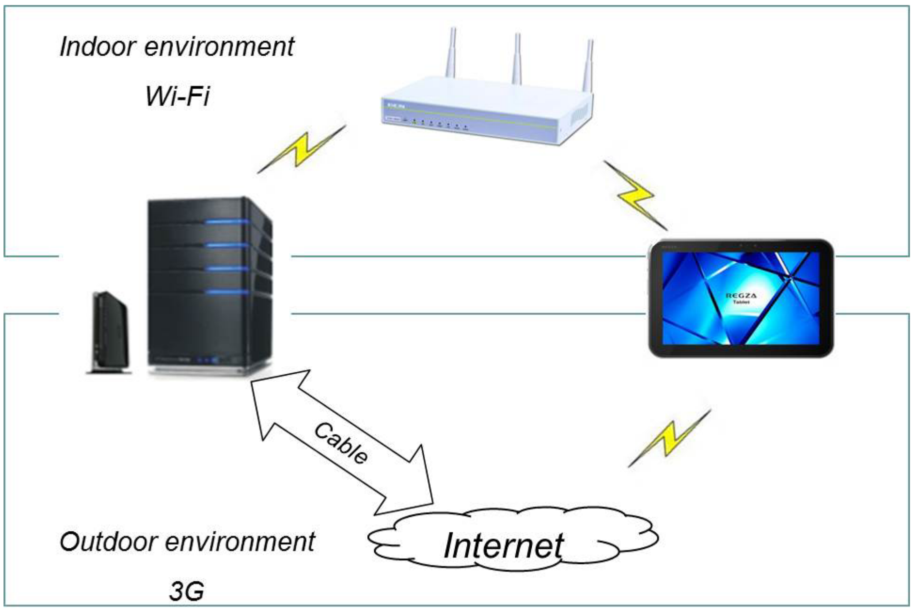


FIGURE 1.3: Connection between clients and server.

The first sub-goal pursued in this thesis is an ***online*** approach (corresponding to the second category), used for a regular (e.g., monthly or bi-monthly, etc.) inspection of a target scene (e.g., blocks on a coast), to assist users by quickly visualizing potential candidate parts of the scene in order to identify which part of the scene could have changed during a particular time period. In order to achieve this sub-goal, we propose a client-server system for change detection shown in Figure 1.2. In our system, the regular inspection of a target scene is done by using the clients, such as mobile phones and tablets, which capture images and display candidate change results. The data are transferred to and stored in a powerful server computer via wireless techniques (WiFi) (see Figure 1.3).

The second sub-goal is an ***offline*** approach, used for improving the accuracy of change detection results. In our online approach, we have estimate a rough change detection result. However, rough result is not enough for the users. So, in order to achieve high accuracy, powerful equipment and long time processing are provided for the offline approach. In this approach, we assume that we have gotten the potential change areas by using the client-server system, and 3D scene data of the areas. Then the details of the change areas are estimated by aligning the 3D scene data.

We have described our sub-goals, but what is the definition of change have not been introduced yet. In the following sub-sections, we will describe the definition and possible methods of change detection.

1.1 Change Detection Methods

1.1.1 Definition of Change

As we mentioned before, a change exists in a wide environment, which can cause unusual accidents. But what is a common definition of change? We say that the original scene is captured at *time A*, which means there is not yet any changes occurred. Then, after a certain period of time, at *time B*, we capture the same scene again. So we define that *a change is a difference of objects in the scene at time A and at time B.*

To build a change detection system in terms of the definition above, three possible directions based on existing methods will be described in next section.

1.1.2 Categories of Detection Methods

2D-2D method The first most used way uses two input images taken by a fixed camera to output a binary image in which change areas are visualized as red pixels. We call this category a 2D-2D based change detection. An output example is shown in Figure 1.4. A huge amount of research on change detection or moving object detection has been studied for surveillance [2–7]. These 2D-2D based methods usually use pixel intensity, color, and texture. It is however not appropriate for our purpose because a

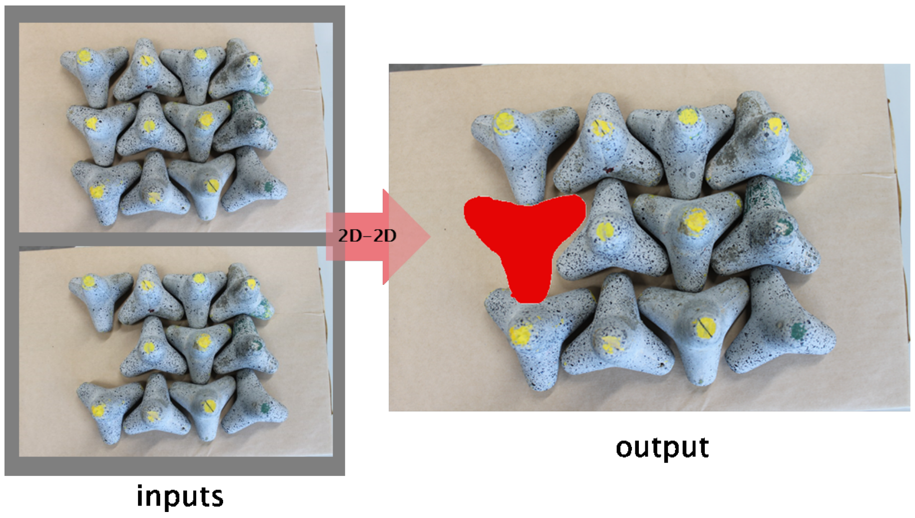


FIGURE 1.4: 2D-2D matching based change detection.

number of fixed cameras are needed to cover the whole scene of the observed place. Hence, it is impractical if in particular the scene is wide and large.

3D-2D method The second possible direction is one that, instead of only 2D images for input, uses 3D data and 2D images to output change detection results as an image. Suppose we have a 3D point cloud at time A and a *query* image at time B. The output detection results can be visualized by red color pixels in 2D query image as shown in Figure 1.5. A fixed camera is no longer needed because the camera pose can be

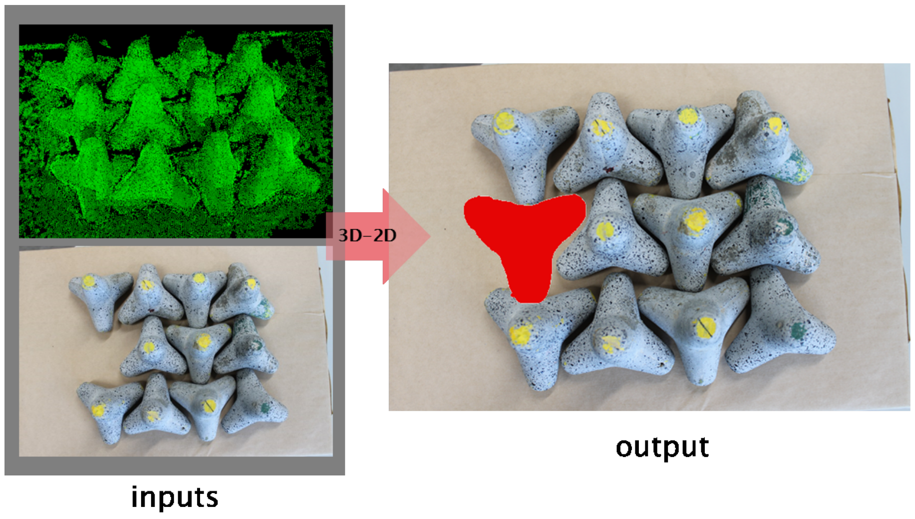


FIGURE 1.5: 3D-2D matching based change detection.

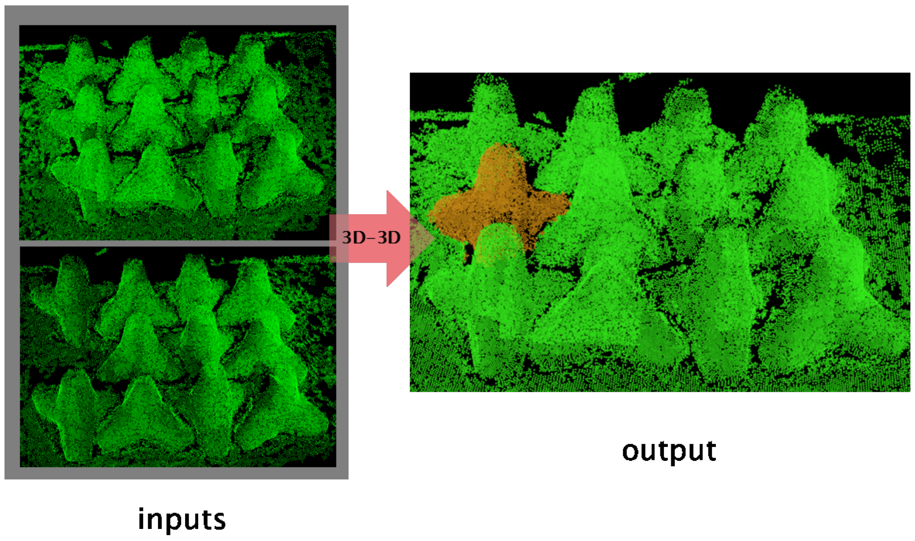


FIGURE 1.6: 3D-3D matching based change detection.

estimated by using 3D-2D matching [8–11] and we can compare the 2D query image with 3D information of the scene. This direction is the best choice for our online approach because: 1) the client in online approach captures the 2D query image, 2) 3D information of the scene are stored in the server, and used to campared to 2D query image. But it is usually difficult to perform 3D-2D matching, because 3D information and 2D images can not be directly compared. So we propose a camera pose estimation method and a change detection method.

3D-3D method The third method uses two 3D point clouds at time A and time B as shown in Figure 1.6. 3D point clouds are aligned to find the difference which is visualized by different color of 3D points. There are a lot of existing methods [12–20] for aligning point clouds. The problem is that the alignment is not easy for point clouds different in size. In our offline approach, the 3D point clouds are reconstructed from different sets of images taken at different points in time. Those are different in size. Therefore, we propose methods to deal with point clouds in different size.

Till here, we have described the methods currently used for change detection. Based on these methods, in next section, we will introduce our approaches contributions and the thesis structures.

1.2 Thesis Overview

This thesis presents a set of methods to perform detection of unusual 3D change for two sub-goals: online and offline.

In the online approach, our method for camera pose estimation uses 3D-2D matching between a 3D reference scene and a 2D query image followed by 2D-2D feature comparison between 2D training and query images. We assume that a 3D scene geometry is given but only at time A as a reference. At time B, a hand-held camera is used to take query images of the same place, and the reference 3D geometry is used for 3D-2D matching for camera pose estimation.

For the 3D-2D matching, detecting keypoints both in the 3D point set and in the 2D images and matching them with their local features seems to be a promising way. In other words, in order to be able to match 3D and 2D keypoints, they should have similar features. Existing methods [21, 22] typically utilize edges, silhouettes or texture, as information for registering 3D data to 2D images, however these features are not same format for 3D data and 2D images. Therefore, we represent the 3D keypoints by their corresponding 2D keypoints. A 3D keypoint is assumed to have corresponding 2D keypoints in two or more of the training images. These 2D keypoints have 2D feature descriptors that can be used to characterize the 3D keypoints. Therefore, we use the 2D feature descriptors as a feature descriptor of the corresponding 3D keypoint. After detecting the 3D keypoints, we perform feature matching and camera pose estimation.

For the change detection, utilizing 2D query images with 3D scene geometry is implemented firstly. Then the nearest image of query image is selected. The matching between the query and the nearest images is performed to find 2D keypoints with correspondences and detect a region of change. The detected region of change is visualized by projecting the 3D points back to the region.

In the offline approach, we divides the task of change detection into two steps: scale estimation and registration followed by detection. One possible way is to estimate the scale of each point cloud separately. In this case, each point cloud has its own scale that is something like the size of a scene. Then we can re-scale one point cloud to another by using the estimated scales. However, this method is computational expensive and unstable. As an alternative, we propose a method for directly estimating the ratio of

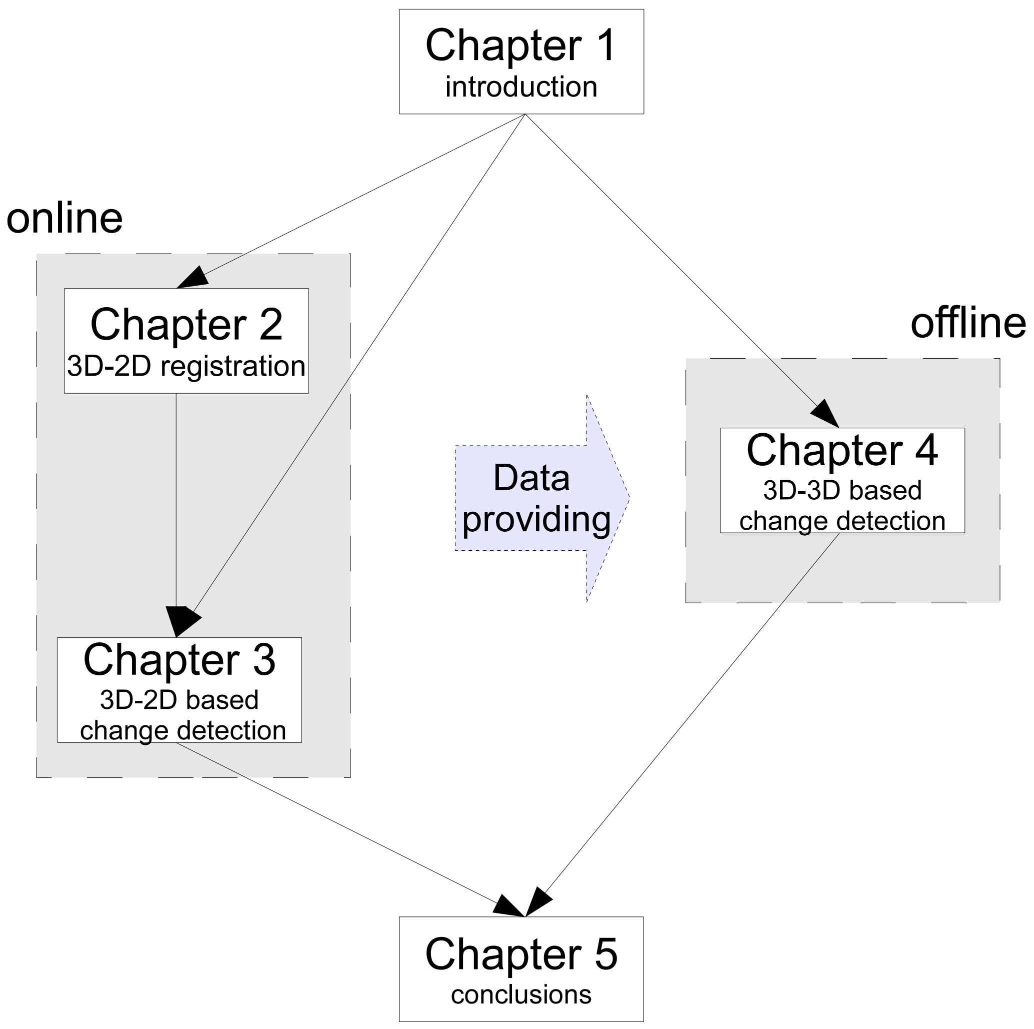


FIGURE 1.7: Thesis chapters' structure.

scales of two point clouds. Once the scale ratio has been estimated, we can use existing alignment methods and perform a simple change detection method.

Instead of placing related work in a single separate section, this thesis describes related work in its context in the thesis. Consequently, multiple related work sections appear throughout the thesis. The breakdown by chapter is as follows (The flow chart is shown in Figure 1.7):

In Chapter 1, we have introduced the motivation of our research. We have also introduced the basic view of '3D-2D matching', '3D-2D based change detection' and '3D-3D alignment' and their relations.

In Chapter 2, we will describe what the 3D keypoints exactly are, how to detect them and how robust they are. This chapter is important for chapter 3. The work described in Chapter 2 was published in the paper “3D keypoints detection from a 3D point cloud for real-time camera tracking” [20].

Chapter 3 describes the details of 3D-2D based change detection. Why would this method work and how to implement it. Chapter 3 also provides sufficient evaluations to prove that the image based change detection method can detect change areas correctly. The use of 3D-2D detection work presented in this thesis was described in “image based detection of 3D scene change” [23] and “camera position estimation for detecting 3D scene change” [24].

Scale ratio ICP which used for 3D-3D alignment will be explained in Chapter 4. We will introduce the related works about 3D scales alignment. The details of implementations are described too. Some of the scale ratio ICP works presented in this thesis was introduced in the paper “scale ratio ICP for 3D point clouds with difference scales” [25].

Chapter 5 presents our conclusions.

Chapter 2

3D Keypoints Detection for 3D-2D Matching

As we introduced in Chapter 1, 3D-2D matching method is important for 3D-2D based change detection. In this chapter, we propose a method for detecting 3D keypoints in a 3D point cloud for robust real-time camera tracking. The method detects keypoints in 3D point clouds which are suitable for matching to 2D keypoints in query images. Recently, it has become easy to obtain 3D point clouds by using a range finder or 3D reconstruction techniques [26–29]. To analyze 3D data with 2D images, 3D-2D registration is usually used. Existing methods [21, 22] typically utilize edges, silhouettes or texture, as information for registering 3D data to 2D images. In addition, detecting keypoints both in the 3D point set and in the 2D images and matching them with their local features seem to be a promising way for a fast 3D-2D registration.

We assume that the 3D point cloud is obtained by some 3D reconstruction techniques which use 2D keypoints in images (we call them *training/reference images*). Therefore, the camera positions of all training images and the 3D scene geometry (as a 3D point cloud) are assumed to be obtained in advance. The key idea is to fully utilize the relationship between the 3D point cloud and the training images. More specifically, we define *a 3D keypoint as a 3D point that has corresponding distinctive 2D keypoints in the training images*. These 3D keypoints are expected to appear with high probability as 2D keypoints in newly taken query images.

In order to match 3D and 2D keypoints, they should have similar features. For this purpose, we represent the 3D keypoints by their corresponding 2D keypoints. A 3D keypoint is assumed to have corresponding 2D keypoints in two or more training images. These 2D keypoints have 2D feature descriptors that can be used to characterize the 3D keypoints. Therefore, we use the 2D feature descriptors as a feature descriptor of the corresponding 3D keypoint. After detecting the 3D keypoints, we perform feature matching and register the newly taken query images.

In the following sections, related work of keypoints detection is given in section 2.1. The main idea concerning 3D keypoints detection will be introduced in section 2.2. We will show the experimental results in section 2.3 and then conclude this chapter.

2.1 Related work of Keypoints Detection

Here we discuss several methods that related to keypoints detection.

There are however not much work has been done on detecting 3D keypoints in a 3D point cloud. Despite of recent progress in keypoint detection in 2D images [30, 31], only a few works on 3D keypoint detection have been reported. Most of this work [32–36] has been proposed for range data on a grid (a depth value z is given at each (x, y) coordinates) usually obtained by range finders. These methods use a grid to characterize keypoints and local features, therefore, they would not work on a point cloud which is an arbitrary set of 3D points. Research works [37, 38] have proposed 3D keypoints detection methods which work on 3D point clouds or mesh surfaces. But it is difficult to find 3D-2D correspondences to perform 2D image registration.

On the other hand, several camera tracking methods which work by matching 2D frames and 3D point clouds have been proposed in the past few years. Schindler et al. [8] presented a method for large-scale location recognition based on video streams and specific trained vocabulary trees. Eade and Drummond used frame-to-frame matching and confining-search-region strategies for rapid feature matching [9]. Irshara et al. [10] proposed a fast location recognition with structure-from-motion (SfM) point clouds by creating synthetic views. Dong et al. [11] also propose a method based on SfM point clouds by selecting a set of optimal keyframes.

In contrast, our approach focuses on the detection of a sparse set of 3D keypoint from SfM point clouds, which allows camera pose estimation to be performed reasonably fast.

2.2 3D Keypoints Detection Algorithm

In this section, we will explain the main idea of 3D keypoints detection. Firstly, we will give a brief overview of our approach. In our approach, there are two modules: the 3D keypoints extraction module and the camera tracking module.

The task of the 3D keypoints extraction module is to extract 3D keypoints (section 2.2.1) and then to compute feature descriptors (section 2.2.2). For each 3D keypoint, corresponding 2D features in the training images are used to compute a 3D feature of the 3D keypoint.

After 3D keypoints and features are prepared, we go to the camera tracking module (section 2.2.3). For each query image, 2D keypoints and their features descriptors are extracted. Then the camera pose of the query image is estimated by using a RANSAC based method [39] by using the matching between the 2D and 3D keypoints.

2.2.1 3D Keypoints Detection

Algorithm 1 Proposed algorithm for 3D keypoints detection.

```
1: $C_{key} = \phi$, is2DKeypoint(i) $= \phi$ /* empty sets */
2: for $i = 1, \ldots, m$ do
3:   $n_1^i = n_2^i = 0$
4:   for $j = 1, \ldots, n$ do
5:     project a 3D point $\boldsymbol{X}^i$ back to a training image $I_j$, $\boldsymbol{x}_j^i = P_j \boldsymbol{X}^i$
6:     if $\boldsymbol{x}_j^i$ is inside of $I_j$ then
7:       if $\boldsymbol{X}^i$ is visible from $\boldsymbol{x}_j^i$ then
8:         $n_1^i$ ++
9:         $count = \#\{\| \boldsymbol{x}_j^i - \boldsymbol{x} \| < th\_d \ \ | \ \ \boldsymbol{x} \in \{\text{all 2D keypoints in } \mathrm{I_j}\}\}$
10:        if count = 1 /* $\boldsymbol{x}_j^i$ is a 2D keypoint */ then
11:          $n_2^i$ ++
12:          add $j$ to is2DKeypoint(i)
13:        end if
14:      end if
15:    end if
16:  end for
17:  if $n_2^i > th\_v$ then
18:    add $\boldsymbol{X}^i$ to $C_{key}$
19:  end if
20: end for
```

In this sub-section, we describe the proposed method for detecting 3D keypoints from a 3D point cloud as shown in Algorithm 1.

From the 3D reconstruction process, we suppose to have the following data:

- 3D points $\boldsymbol{X}^i (i = 1, 2, \ldots, n)$, of a point cloud C, and associated normal vectors $\boldsymbol{n}^i$.
- Training images $I_j (j = 1, 2, \ldots, m)$, used for reconstruction.
- External camera parameters as a projection matrix P_j for each image I_j.

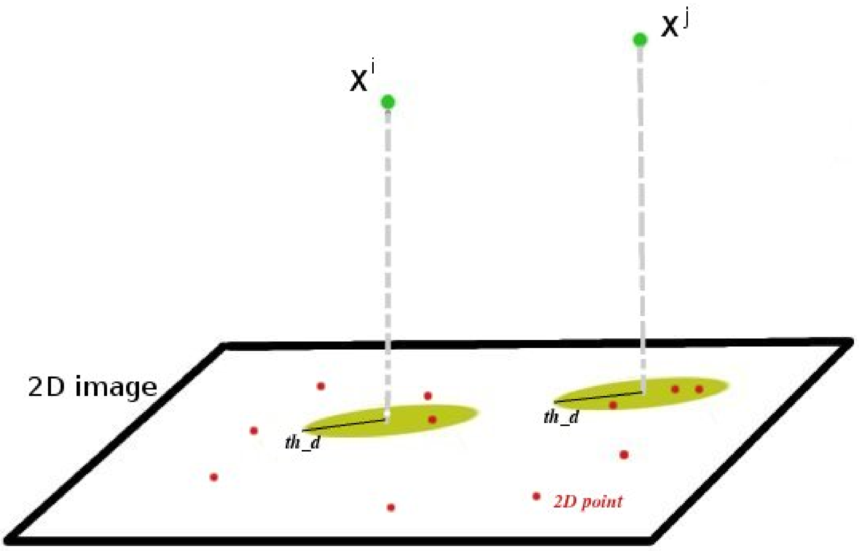


FIGURE 2.1: The use of distance threshold to exclude unreliable 3D keypoints. Red points are 2D keypoints in an image and green points are 3D points. $\boldsymbol{X}^i$ is a candidate of a 3D keypoint, but $\boldsymbol{X}^j$ is not because there are several 2D keypoints around its projection onto the image.

We want to find a 3D keypoint to appear as a 2D keypoint in many images for establishing the 3D-2D keypoint correspondences. Therefore, we eliminate 3D points which have corresponding 2D points only in fewer than *th_v* images, which is a visibility threshold for 3D points.

Those 3D keypoints should be visible from many training images, however, merely projecting 3D points to images is not enough to check the visibility. Since points in a 3D point cloud can be projected to inner-side of a surface, we need to use normal vectors to ensure that the projection is performed outward from of a surface. This procedure, usually called *backface culling* in graphics, is described as follows. The 3D points $\boldsymbol{X}^i$ are projected back to all images I_j to obtain the corresponding 2D points $\boldsymbol{x}_j^i = P_j \boldsymbol{X}^i$. If $\boldsymbol{x}_j^i$ is outside of I_j, then it is excluded. Then the normal vectors of $\boldsymbol{X}^i$ are used for

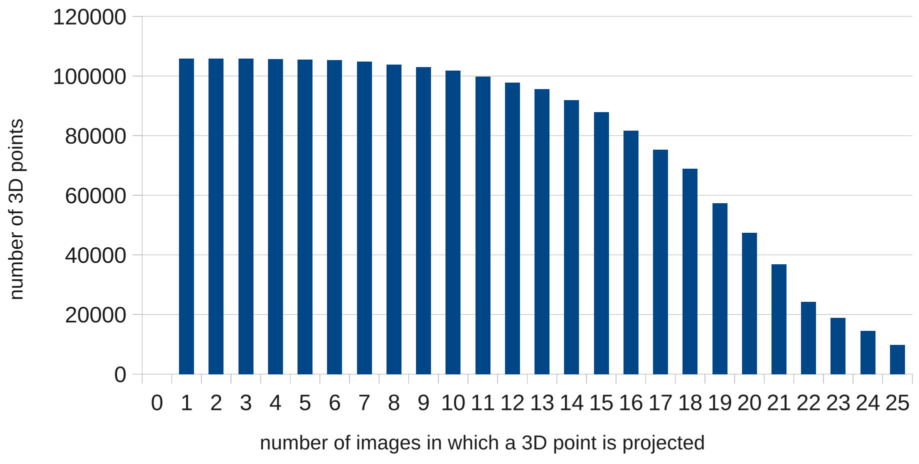


FIGURE 2.2: Histogram of n_1^i, the number of images to which 3D points are projected.

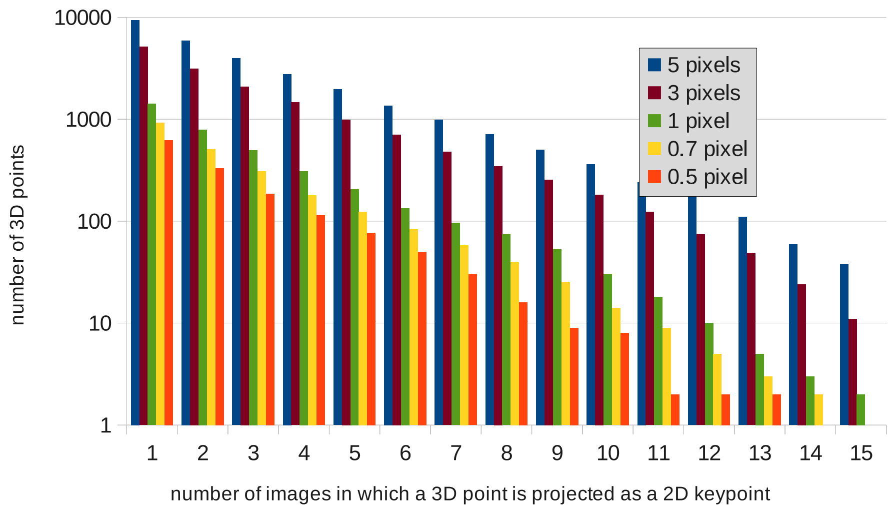


FIGURE 2.3: Histogram of n_2^i, the number of images in which 3D points appear as 2D keypoints for different visibility thresholds.

backface-culling.

$$\boldsymbol{n}(\boldsymbol{X}^i) \cdot (-\boldsymbol{c}) < \cos 60^\circ. \tag{2.1}$$

When the angle between the normal vector $\boldsymbol{n}(\boldsymbol{X}^i)$ and the viewing vector of a camera $\boldsymbol{c}$ is less than 60 degree in this case, $\boldsymbol{X}^i$ is not able to be projected.

When there is a 2D keypoint near to $\boldsymbol{x}_j^i$ as shown in Figure 2.1, i.e., the distance between the 2D keypoint and $\boldsymbol{x}_j^i$ is smaller than th_d, a distance threshold (5 pixels in this case), then $\boldsymbol{X}^i$ is said to appear as (and corresponds to) the 2D keypoint. If the distance is

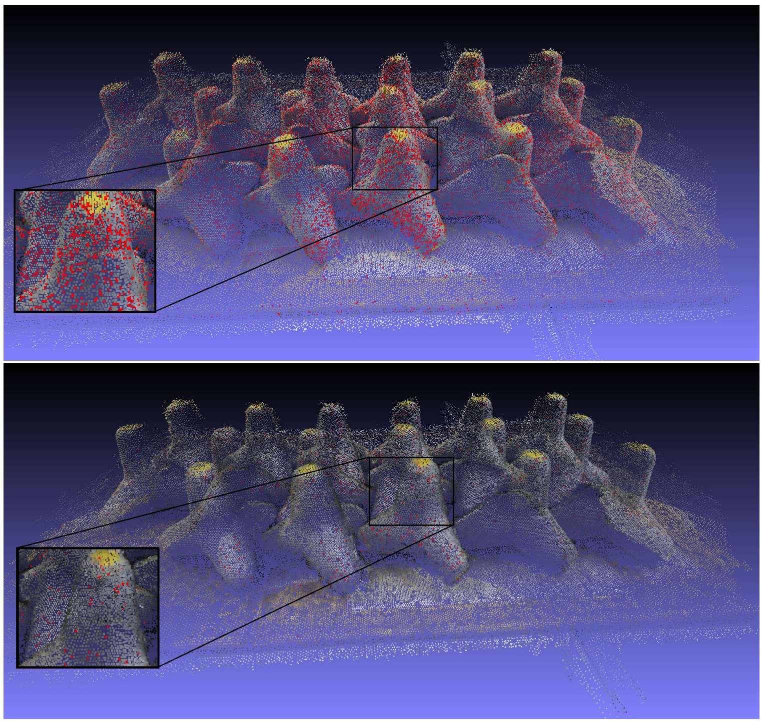


Figure 2.4: Detected 3D keypoints in 3D point cloud with (top) $n_2^i \geq 1$ and (bottom) $n_2^i \geq 7$.

larger than *th_d*, then it is not a keypoint. Often there could be more than one 2D points within *th_d*, however, those 3D points ($\boldsymbol{X}^j$ in Figure 2.1) are considered as outliers and hence excluded.

Finally, we count the number n_2^i of images in which 3D points appear as 2D keypoints. When n_2^i is larger than the visibility threshold (7 in this case), we set this 3D point as a 3D keypoint.

Figures 2.2 and 2.3 show results for a 3D point cloud of 105,779 points reconstructed from 25 training images (detailed in the experiments). Figure 2.2 shows how many training images there are in which a 3D point is projected (n_1^i in the algorithm). Every 3D point appears in at least 1 training image. This figure demonstrates the need for reducing the number of 3D points. Figure 2.3 shows the number of images in which a 3D point appears as a 2D keypoint (n_2^i in the algorithm) for different distance thresholds over different visibility thresholds. Now, 91% of the 3D points (80,262 points) are not 2D keypoints in any images, while only 0.9% (984 points) are 2D keypoints (*th_d*=5 pixels, *th_v*=7). These 3D keypoints are a reasonably small set (about 1000 points) compared to the original 3D points, and expected to appear with high probability as 2D keypoints again in any newly taken images. They are shown in Figure 2.4 as red points superimposed in 3D point cloud. There are too many points if the visibility threshold is 1, while a moderate number of points is obtained when visibility threshold is 7. In this 3D point cloud, we set the visibility threshold to 7 because we have sufficiently sparse 3D keypoints.

To see the effects of th_d and th_v, we counted how many 3D points appear in each training image for each th_d. Figure 2.5, 2.6, 2.7, 2.8 and 2.9 show n_2^i of the 3D points projected onto each training image. Each bar shows the number of all projected points (light gray), points corresponding to $th\_v = 1, \ldots, 6$ (dark gray), and points obtained for $th\_v > 6$ (black). Figure 2.5 shows that about 1000 3D keypoints detected by the proposed method appear in all training images. Therefore, we can expect that about 1000 3D keypoints would also appear in newly taken images.

Till here, we have described that how to detect 3D keypoints. In next subsections, how to generate 3D keypoint features and how to use them will be introduced.

2.2.2 Embedding 2D Features into 3D Keypoints

In order to generate 3D features, we embed 2D features into 3D keypoints. The 2D features are actually SIFT features which will be introduced from now.

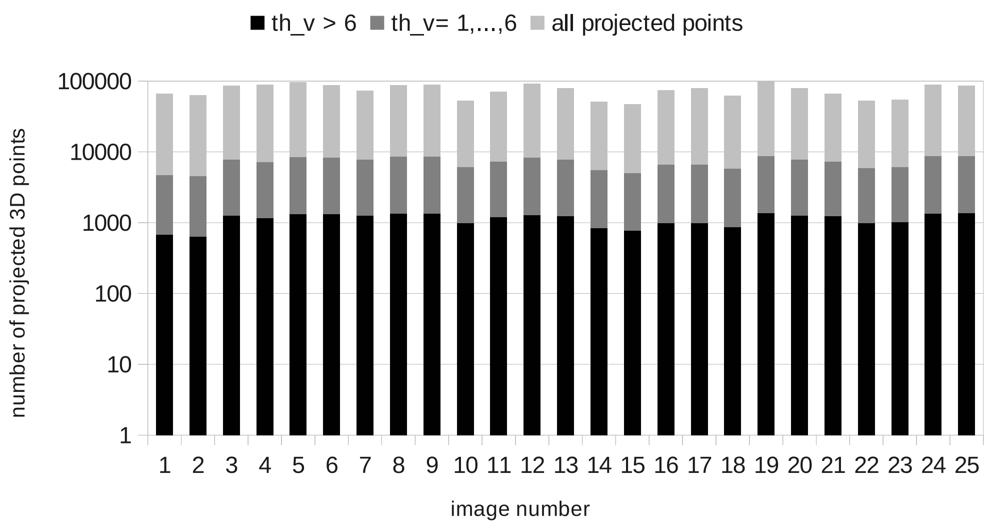


Figure 2.5: Histogram of projected points in each images when $th\_d = 5$.

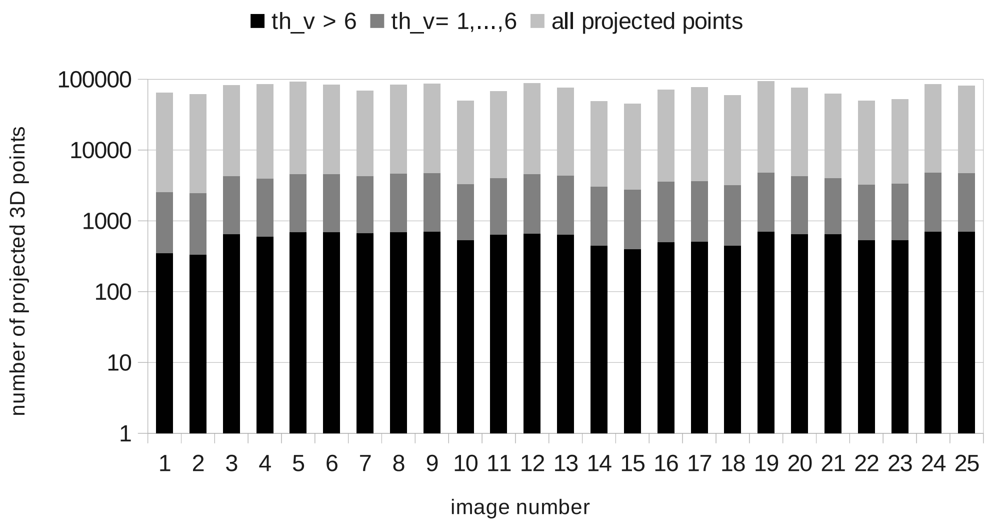


FIGURE 2.6: Histogram of projected points in each images when $th\_d = 3$.

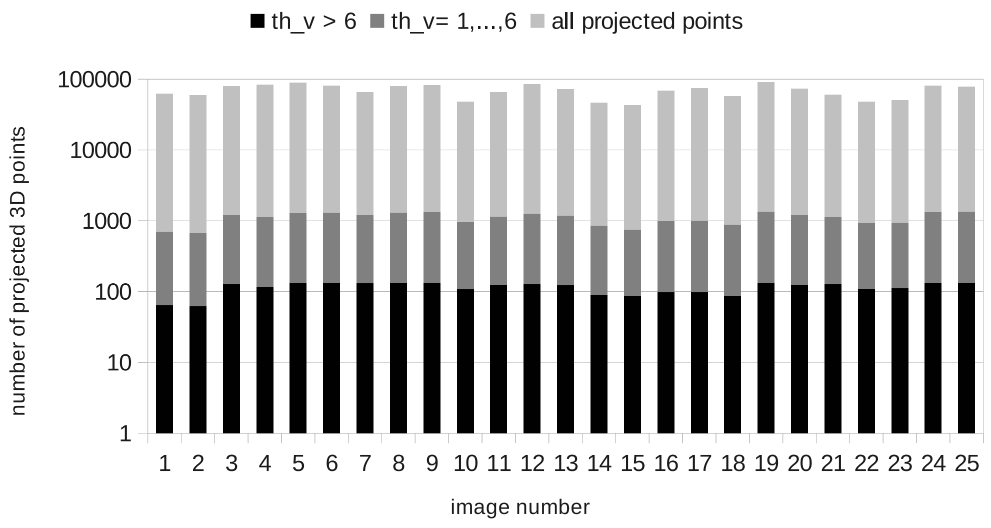


FIGURE 2.7: Histogram of projected points in each images when $th\_d = 1$.

2.2.2.1 SIFT

First, we introduce the concept of SIFT (Scale-invariant feature transform) which first introduced in [30]. SIFT is an algorithm to detect and describe local features in images.

For an image, SIFT is a “feature description” used for describing objects. If one scene appears in two images, the description extracted from the two images, can then be used

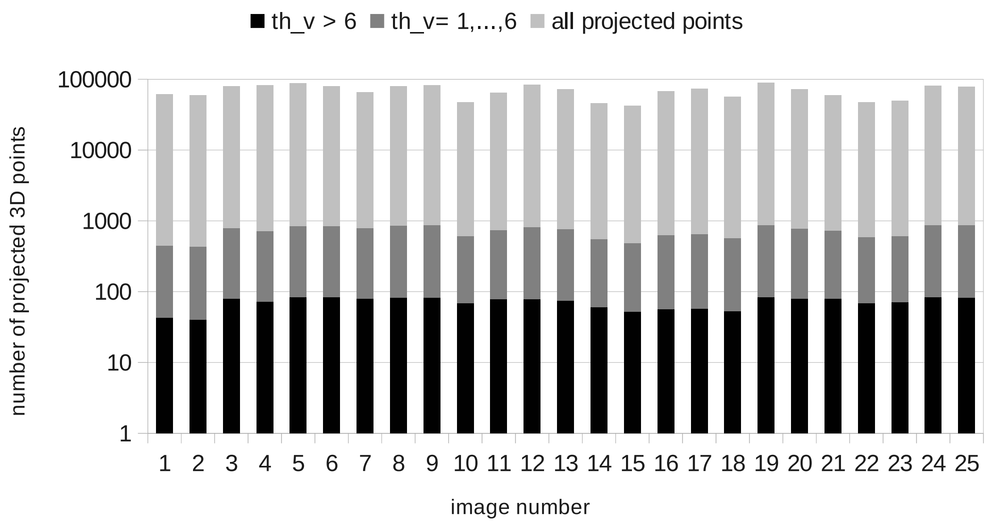


FIGURE 2.8: Histogram of projected points in each images when $th\_d = 0.7$.

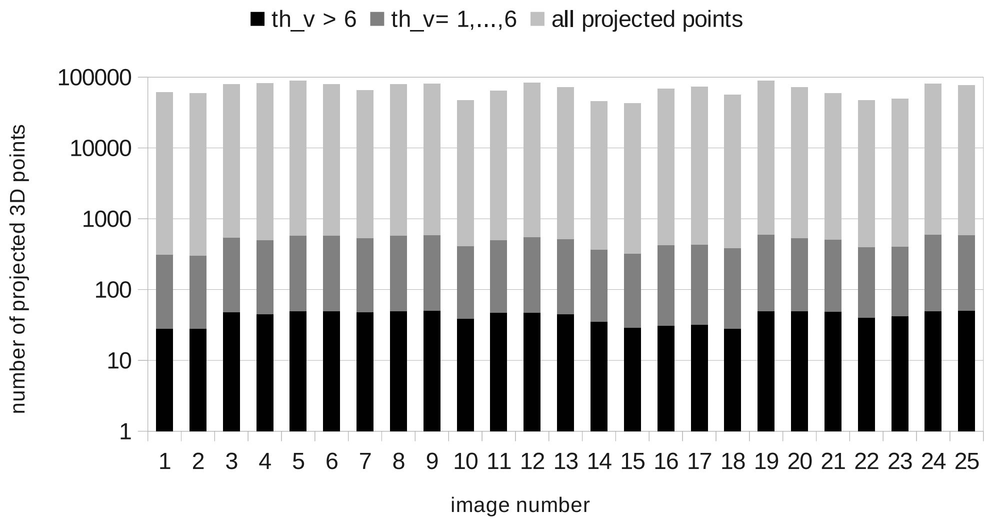


FIGURE 2.9: Histogram of projected points in each images when $th\_d = 0.5$.

to identify the objects. It is easy to find the location of the objects in both images. In order to perform a reliable recognition, changes in image scale, noise, rotation and illumination should not affect the interesting points extracted result. Such extracted feature points usually lie on high-contrast regions of images, such as object edges, corners or shadows. Another important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another. For example, if only the corners of desk are used as features, they would work no matter what is the desk's position. But if points on the desk were also used, the recognition would fail if the desk is rotated in the image. Similarly, if there are flexible objects exist in two images, the features located in such objects will not work if the object changed in the two images. However, SIFT can detect much larger number of features from the images. It can reduces the contribution of the errors caused by these variations flexible objects.

There are two important stage for generating SIFT as follows:

detector This is the first stage which detect the interesting points. The interesting points are also called keypoints in SIFT framework. The keypoints detection is also the detector for features. In order to detect SIFT keypoints, we need to implement two steps as follows:

- DoG image generation

  In this step, the image is convolved with Gaussian filters at different scales and then the differences of Gaussian images are performed. Concretely, suppose we have the original image $I(x, y)$. Then we have a convolution image $L(x, y, k\sigma)$ of $I(x, y)$ with Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$:

  $$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y).$$

  Then, a DoG image $D(x, y, \sigma)$ is given by

  $$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma).$$

  Because a DoG image between scales $k_i\sigma$ and $k_j\sigma$ is just the difference of the Gaussian-blurred images at scales $k_i\sigma$ and $k_j\sigma$. For scale space extrema detection in the SIFT algorithm, the image is first convolved with Gaussian-blurs at different scales. The convolved images are grouped by octave (an octave corresponds to the double value of σ), and the value of k_i is selected so that we obtain a fixed number of convolved images per octave. Then the Difference-of-Gaussian images are taken from adjacent Gaussian-blurred images per octave.

- Maxima/Minima selection

In this step, keypoints are selected as the maxima/minima of the Difference of Gaussian (DoG) that occur at different scales. Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images by considering the scales. This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. If the pixel value is the maximum or minimum among all compared pixels, it is the potential selection of keypoint.

descriptor This is the second stage which called descriptor. It has two parts of computation: orientation and description.

- Orientation

  In this step, each keypoint is assigned some orientations based on the gradient directions of images. Once we can get the orientation of image, we can easily achieve the rotation invariance purpose. Because the rotation invariance is actually the keypoint descriptor represented by considering the image orientation.

  Broadly, the original image $I(x, y)$ is convoluted by Gaussian blur at scale σ to get Gaussian-Smoothed image $L(x, y, \sigma)$. So that, all computations are performed in a scale-invariant manner. For an image sample $L(x, y)$ at scale σ, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are precomputed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \arctan(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y)).$$

  The gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$ are computed for every pixel in a neighboring region around the keypoint in the Gaussian-Blurred image L. An orientation histogram with 36 bins is formed, with each bin covering 10 degrees. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within

80% of the highest peaks are assigned to the keypoint. In the case of multiple orientations are going to be assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.

- Description

  Now, we want to compute the keypoints descriptors. The descriptors are highly distinctive for different keypoints. They are also invariant to the images which are variations such different illumination and different viewpoints. Concretely, a set of orientation histograms are created on 4×4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16×16 region around the keypoint such that each histogram contains samples from a 4×4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are $4 \times 4 = 16$ histograms, and each histogram has 8 bins, so we can get the vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination.

We have already finished the introduction of SIFT. So it is helpful to understand the next section: embedding 2D features into 3D keypoints.

2.2.2.2 Embedding 2D Features into 3D keypoints

Here we describe a 3D keypoint feature descriptor for 3D-2D matching. To perform 3D-2D matching with features, 2D and 3D keypoints should have the same or similar features.

Let the 2D feature descriptor (e.g., SIFT [30]) of a point $\boldsymbol{x}_j^i$ be $S(\boldsymbol{x}_j^i)$. Then the feature descriptor $S(\boldsymbol{X}^i)$ of a 3D keypoint $\boldsymbol{X}^i \in C_{key}$ is defined by the average of the corresponding 2D feature descriptors as follows:

$$S(\boldsymbol{X}^i) = \frac{1}{n_2^i} \sum_{j \in \text{is2DKeypoint(i)}} S(\boldsymbol{x}_j^i). \tag{2.2}$$

2.2.3 Feature Matching and Camera Pose Estimation

After generating 3D keypoints, we need to match 3D keypoints and 2D keypoints of newly taken query images, and estimate the camera poses of those images. Estimation is done by a well-known method for estimation of camera pose with RANSAC [39]. In the following two sub-sections, the projection matrix, estimation of R and $\mathbf{t}$, and RANSAC algorithm will be introduced.

2.2.3.1 Projection Matrix

Projection matrix[1] is the camera pose of image which used to transform 3D points onto 2D image's coordinate. Suppose we have a point $\mathbf{p}$, with coordinates $\boldsymbol{X}^0 = [X^0, Y^0, Z^0]^T \in \mathbb{R}^3$ relative to the world reference frame. As we know, the coordinates $\boldsymbol{X}^1 = [X^1, Y^1, Z^1]^T \in \mathbb{R}^3$ of the same point $\mathbf{p}$ relative to the camera frame are given by a rigid-body transformation $g = (R, \mathbf{t})$ of $\boldsymbol{X}^0$:

$$\boldsymbol{X}^1 = R\boldsymbol{X}^0 + \mathbf{t} \in \mathbb{R}^3. \tag{2.3}$$

So, we can get

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z^1} \begin{bmatrix} X^1 \\ Y^1 \end{bmatrix}. \tag{2.4}$$

where the point $\boldsymbol{X}^1$ is projected onto the image plane at. In homogeneous coordinates, the projection can be re-written as:

$$Z^1 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^1 \\ Y^1 \\ Z^1 \\ 1 \end{bmatrix}. \tag{2.5}$$

[1]This section is inspired by the textbook ‘An Invitation to 3D Vision’ [40].

We set $\boldsymbol{X}^1 = [X^1, Y^1, Z^1, 1]^T$ and $\mathbf{x} = [x, y, 1]^T$ in homogeneous coordinates, so, we can rewrite the above equation equal to

$$Z^1\mathbf{x} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{X}^1, \tag{2.6}$$

Since the coordinate Z^1 (the depth of the point $\mathbf{p}$) is usually unknown, we can simply write it as an arbitrary scalar $\lambda \in \mathbb{R}$.

In the above equation we can decompose the matrix into

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.7}$$

Then we define that,

$$K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in R^{3\times3}, \Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in R^{3\times4}. \tag{2.8}$$

So we have

$$\lambda x = K_f \Pi_0 \boldsymbol{X}. \tag{2.9}$$

From the coordinate transformation we have $\boldsymbol{X}^1 = [X^1, Y^1, Z^1, 1]^T$, which is

$$\begin{bmatrix} X^1 \\ Y^1 \\ Z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X^0 \\ Y^0 \\ Z^0 \\ 1 \end{bmatrix}. \tag{2.10}$$

In simple way, it is $\boldsymbol{X}^1 = g\boldsymbol{X}^0$.

To summarize, by using the above notation, the overall geometric model can be described as:

$$\lambda x = K_f \Pi_0 \boldsymbol{X} = K_f \Pi_0 g \boldsymbol{X}^0. \tag{2.11}$$

Here, K_f is the intrinsic camera matrix and Π_0 is the extrinsic camera matrix (without calibration information). We call $K_f \Pi_0 g$ the standard *projection matrix*.

Next, we will introduce the estimation of R and $\mathbf{t}$

2.2.3.2 Estimation of R and t

Assuming that we have two sets of corresponding points $X = \{\boldsymbol{X^1}, \boldsymbol{X^2}, \ldots, \boldsymbol{X^n}\}$ and $Y = \{\boldsymbol{Y^1}, \boldsymbol{Y^2}, \ldots, \boldsymbol{Y^n}\}$. Their center points are $\bar{\boldsymbol{X}} = \frac{1}{n}\sum \boldsymbol{X}^i$ and $\bar{\boldsymbol{Y}} = \frac{1}{n}\sum \boldsymbol{Y}^i$ where $i = 1, 2, \ldots, n$. Then we can get deviations $X' = \{\boldsymbol{X^1} - \bar{\boldsymbol{X}}, \boldsymbol{X^2} - \bar{\boldsymbol{X}}, \ldots, \boldsymbol{X^n} - \bar{\boldsymbol{X}}\}$ and $Y' = \{\boldsymbol{Y^1} - \bar{\boldsymbol{Y}}, \boldsymbol{Y^2} - \bar{\boldsymbol{Y}}, \ldots, \boldsymbol{Y^n} - \bar{\boldsymbol{Y}}\}$. In order to estimate R and $\boldsymbol{t}$[2], the objective function which we want to solve is:

$$\min \sum_{i=1}^{n} \left\| \boldsymbol{Y}^i - (R\boldsymbol{X}^i + \boldsymbol{t}) \right\|^2. \tag{2.12}$$

It is the same form of

$$\min \|Y - (RX + \boldsymbol{t})\|_F^2. \tag{2.13}$$

- Estimation of R

  There are three methods to estimate R which are Lagrange multiplier [42], quaternions [43, 44] and SVD (Singular Value Decomposition) [42, 45–47]. In this section, we introduce the concept of SVD.

  Substitute Equation 2.13 for X' and Y', we can get the function (just consider about R)

$$\min \left\|Y' - RX'\right\|_F^2. \tag{2.14}$$

  It is the same form of

$$\min \operatorname{tr}((Y' - RX')^T (Y' - RX')). \tag{2.15}$$

  The simplify form is

$$\max \operatorname{tr}(Y'^T R X'). \tag{2.16}$$

[2]This section is inspired by the paper ‘Pose Estimation and Rotation Matrices’ [41].

Just consider about $\mathrm{tr}(Y'^T RX')$, it is the same form of $\mathrm{tr}(RX'Y'^T)$. Because $X'Y'^T = U\Sigma V^T$ (SVD), we can get

$$\mathrm{tr}(Y'^T RX') = \mathrm{tr}(RU\Sigma V^T) = \mathrm{tr}(V^T RU\Sigma) \leq \mathrm{tr}(\Sigma). \tag{2.17}$$

So when we maximize Equation 2.16, we have

$$V^T RU = I, \tag{2.18}$$

which means

$$\hat{R} = VU^T. \tag{2.19}$$

But the determinants of Equation 2.19 are +1 and -1. So we calculate the rotation R as

$$\hat{R} = VHU^T, \tag{2.20}$$

where $H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & |VU^T| \end{bmatrix}$.

- Estimation of **t**

  As we know that

$$Y = RX + \boldsymbol{t}, \tag{2.21}$$

  So,

$$\bar{Y} = R\bar{X} + \boldsymbol{t}, \tag{2.22}$$

  which means that we can get

$$\boldsymbol{t} = \bar{Y} - R\bar{X}. \tag{2.23}$$

Till here, we have known projection matrix and how to estimate it. But it is important to find matches for the estimation. In next sub-section, the RANSAC algorithm will be described to do robust inliners extraction.

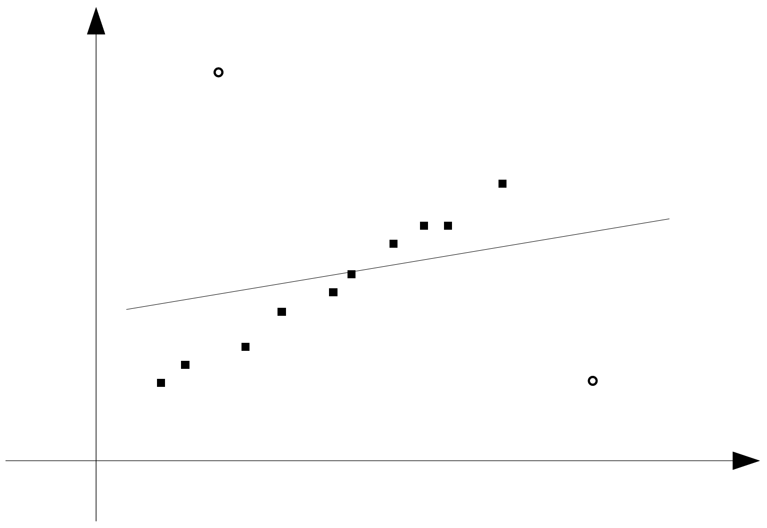


Figure 2.10: A least squares (orthogonal regression) fit to the point data is severely affected by the outliers (The solid points are inliers, the open points are outliers).

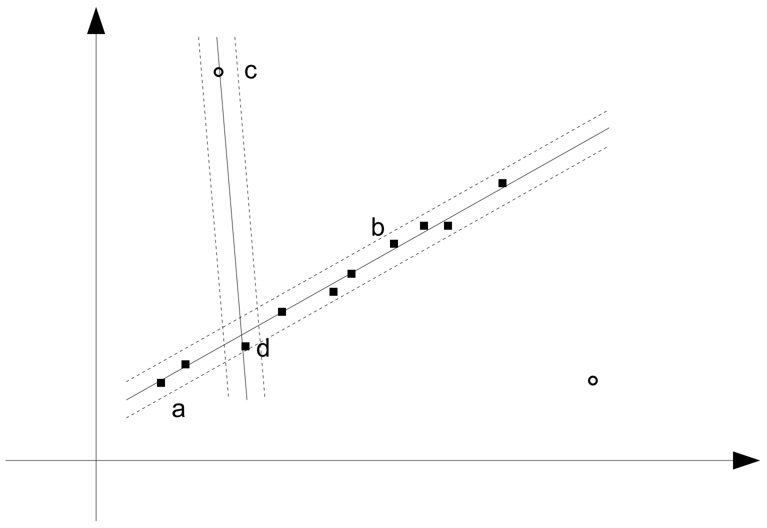


Figure 2.11: A RANSAC fitted problem (The solid points are inliers, the open points are outliers). The dotted lines indicate the threshold distance. For the lines shown the support is 10 for line $< a, b >$ where both of a and b are inliers; and 2 for line $< c, d >$ where the point c is an outlier.

2.2.3.3 RANSAC Algorithm

We will introduce the RANSAC algorithm in this section[3]. Suppose we have already get a set of corresponding points, which are $\{X, Y\}$, used for camera pose estimation. But because of the accuracy of the correspondences generation, there are some mismatched points exist in $\{X, Y\}$ which called *outliers*. Usually, this kind of outliers are followed Gaussian error distribution which means most of the data are correct. So we need a

[3]This section is inspired by the textbook 'An Invitation to 3D Vision' [40].

Algorithm 2 RANSAC algorithm.

```
1. selects $N$ pairs of corresponding points randomly
2. estimates camera pose $P$ by using the selected points
3. finds how many corresponding point fit $P$ within a given tolerance. Call this $K$.
4. if $K$ is big enough, accept fit and exit with success.
5. repeat $1 \dots 4$ $L$ times
6. fail if you get here
```

How big K has to be depends on what percentage of the data you think belongs to the structure being fit and how many structures you have in the image. If there are multiple structures then, after a successful fit, remove the fit data and redo RANSAC.

robust method to reject the outliers. We call the method outlier rejection algorithm RANSAC (the RANdom SAmple Consensus) [39].

In order to simplify the problem above, we just take the 2-Dimension data as an example. We use 2D example to visualize the outlier rejection problem. The problem is as follow:

Suppose we have a set of 2D data points in 2D coordinate. We want to take a line to fit these 2D points. Which means that the summation of the distances when project 2D points to the line is a minimized value. The problem is illustrated in Figure 2.10 shows that the generated line without considering about outliers is incorrect. This is actually two problems: a line fit to the data; and a classification of the data into inliers and outliers.

The procedure of RANSAC is: first, random select two points in the 2D coordinate. A line can be defined by using the selected two points. Then the number of points that lie within a distance threshold is used to support the line. After some iterations, select the most biggest support for the line fit. Now, the points within the threshold distance are the inliers. The example can be seen in Figure 2.11. We can get a solid line $< a, b >$ in the example with the maximum support points number as 10. However, if the result is estimated as the solid line $< c, d >$, the support is just 2 points. So we select line $< a, b >$ as the result. The points lie on or near line $< a, b >$ are said to be the inliers.

The RANSAC algorithm of the camera pose estimation is summarized in Algorithm 2.

Till here, we can use RANSAC to estimate the projection matrix. In next section, we will introduce our evaluation experiments.

2.3 Evaluation

2.3.1 Indoor Scene with Small Blocks

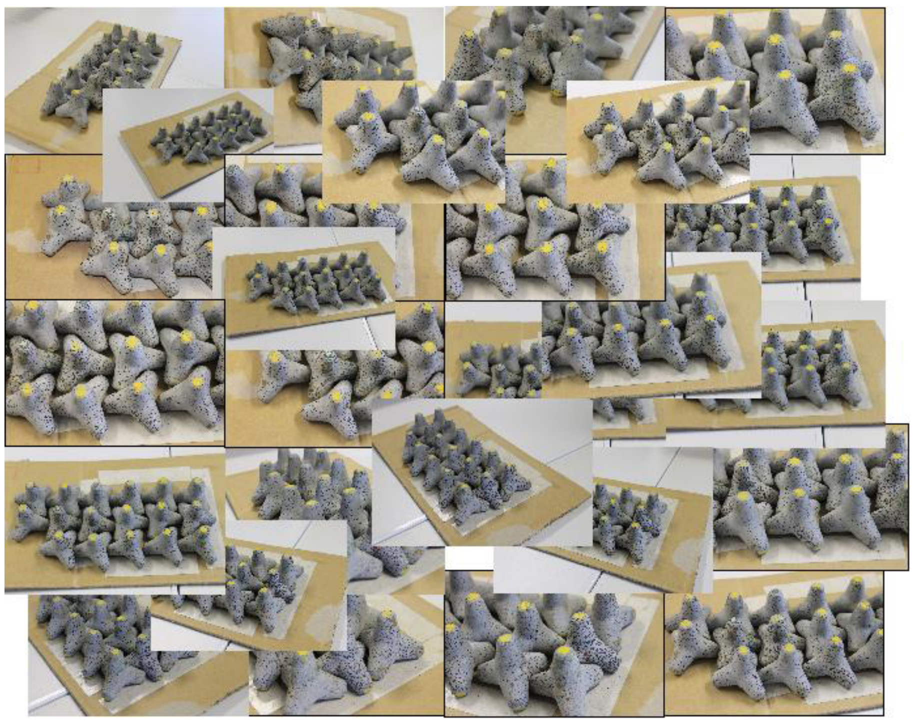
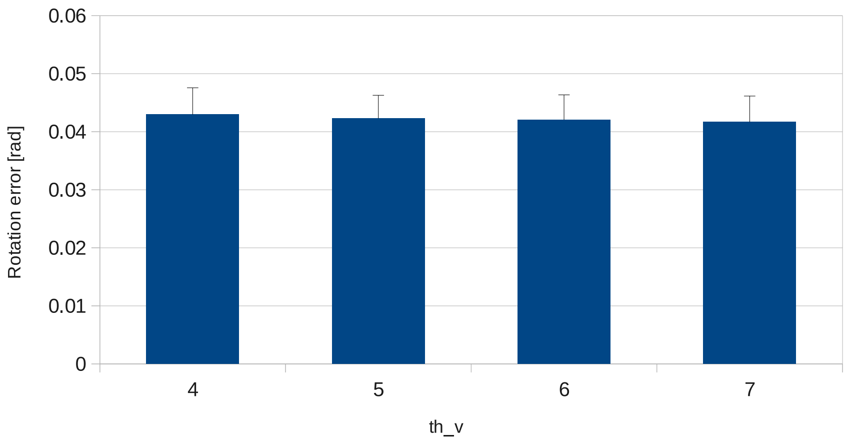
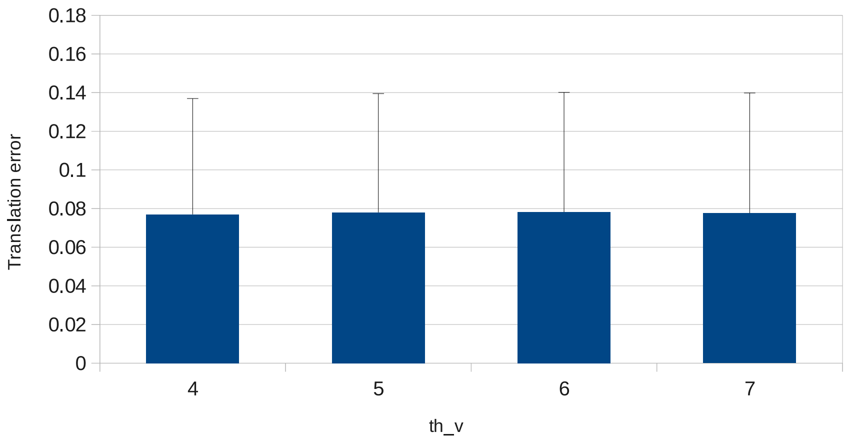


FIGURE 2.12: 27 training images of the indoor scene with small blocks.

The 3D point cloud of small blocks on a table is used to evaluate the accuracy of the proposed method. The 3D point cloud was obtained by performing 3D reconstruction with Bundler [28] followed by Patch-based Multi-view Stereo (PMVS2) [26, 27]. 27 images (2256×1504 in size) of the small block scene were used (shown in Figure 2.12). The scene contains a lot of small blocks that are similar to each other in color and shape; therefore it is a difficult and challenging task. After all 27 projection matrices were obtained; the first 25 images were used as a training set to reconstruct a 3D point cloud of 105,779 points. Then, the 3D keypoints were computed based on this 3D point cloud. The remaining 2 images were used for evaluation. For image 26 and image 27, we estimated projection matrices 30 times for averaging estimates. Let the original projection matrices be P_t (obtained by using all 27 images). For evaluating errors, each estimated projection matrix $\hat{P}$ and corresponding true projection matrix P_t were decomposed into rotation matrices $\hat{R}$ and R_t, and translation vectors $\hat{\boldsymbol{t}}$, $\boldsymbol{t}_t$. Translation



(a)



(b)

FIGURE 2.13: Estimation results for the indoor scene. (a) Rotation matrices evaluation for *th_v*. (b) Translation vectors evaluation for *th_v*.

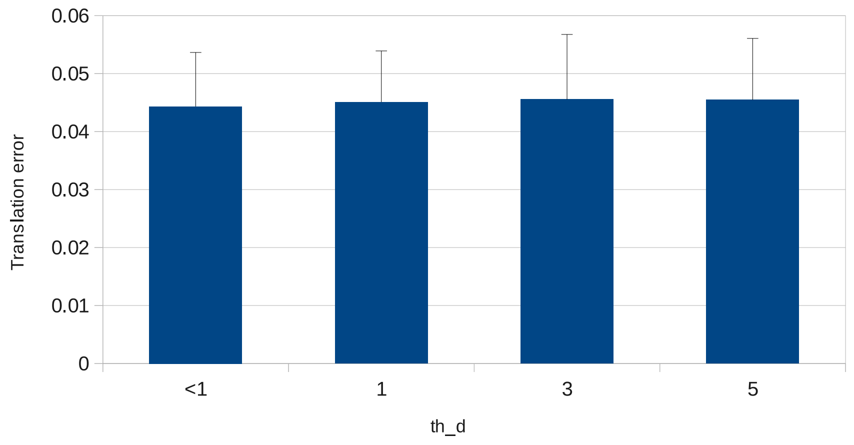
error is evaluated by the relative error as follows:

$$e(\boldsymbol{t}_t, \hat{\boldsymbol{t}}) = \frac{\| \boldsymbol{t}_t - \hat{\boldsymbol{t}} \|}{\| \hat{\boldsymbol{t}} \|}. \tag{2.24}$$

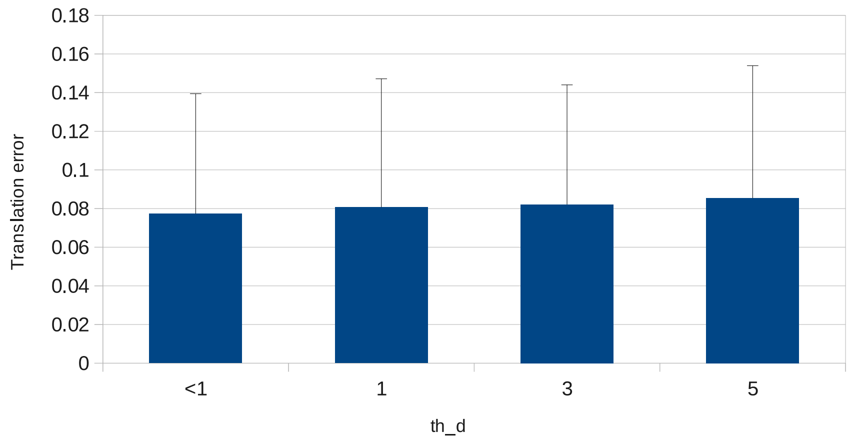
Rotation error is evaluated by the distance of two rotation matrices[40] as follows:

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(R_t \hat{R}^T) - 1}{2}\right). \tag{2.25}$$

The results are shown in Figures 2.13(a) and (b) for different visibility thresholds. We



(a)



(b)

FIGURE 2.14: Estimation results for the indoor scene. (a) Rotation matrices evaluation for th_d when $th\_v = 0$. (b) Translation vectors evaluation for th_d when $th\_v = 0$.

evaluated errors for $th\_v = 4, \ldots, 7$ and the smallest error is obtained when $th\_v = 5$. In this case, the rotation error is about 0.04 [rad], while the translation error is up to 8%. The accuracy can be improved if we use a better alternative method for camera pose estimation, instead of the one described in section 2.2.3.

We compare the errors with the case where simply all 3D points were used, i.e., 3D keypoints detected by our proposed method were not used. Figures 2.14 (a) and (b) show the results for fixing $th\_v = 0$ (the visibility threshold is disabled) and different th_d (distance threshold is disabled when $th\_d < 1$). In this case, the 3D points are simply used when they appear in the images. As shown in the figures, the average errors

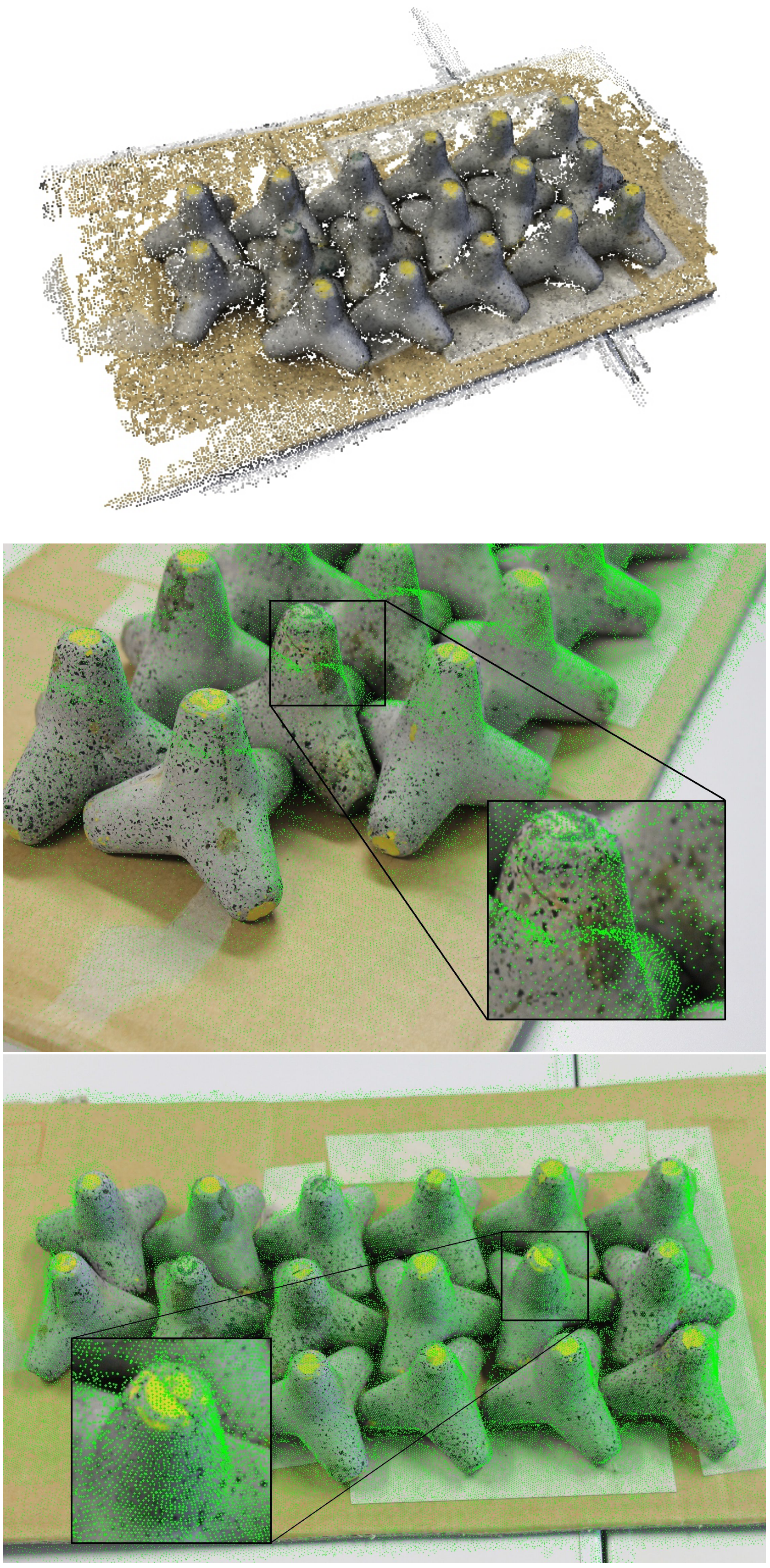


FIGURE 2.15: Estimation results. (top) 3D point cloud. (middle and bottom) Projection of the 3D point cloud with estimated camera poses onto image 26 and 27.

and standard deviations are slightly larger than those in Figure 2.13 (a)(b). This means that the proposed method keeps the same accuracy with fewer keypoints being used, which is one of the advantages of our method.

In this experiment, all of the 2D keypoints are extracted by SiftGPU [48]. The 3D features descriptors have the same form of SIFT[30] features of 128 dimensions.

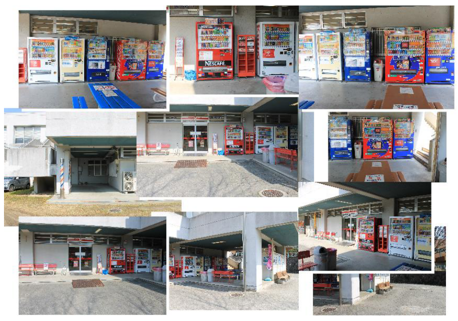
The projected 3D points (green points) on query images 26 and 27 are shown in Figure 2.15. When we have an accurate estimation of the camera pose, the 3D points are projected onto the images so that the projected points and the objects in the scene appear exactly the same shape and place. In particular, the projected points in Figure 2.15 are aligned along the edges of the object, which shows that the estimation is accurate, because small error in the estimation usually leads to misalignment along the edges.

To demonstrate the robustness of the proposed method, we also applied the proposed method to a video sequence (1280×720 in size) taken by a hand-held HD camera. For most of the frames, the camera pose is estimated well in real time even with non-optimized experimental code (about 3 fps). Because some frames are blurred where the camera moves fast and enough number of 2D keypoints are not obtained, the camera pose estimation fails in such few frames.

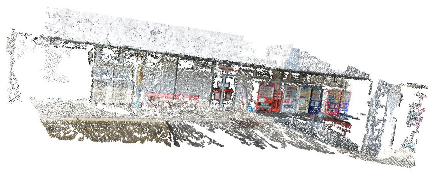
The full video sequence is available on YouTube (`http://youtu.be/Wx_JGpAJTwI`)

2.3.2 Outdoor Real Scene

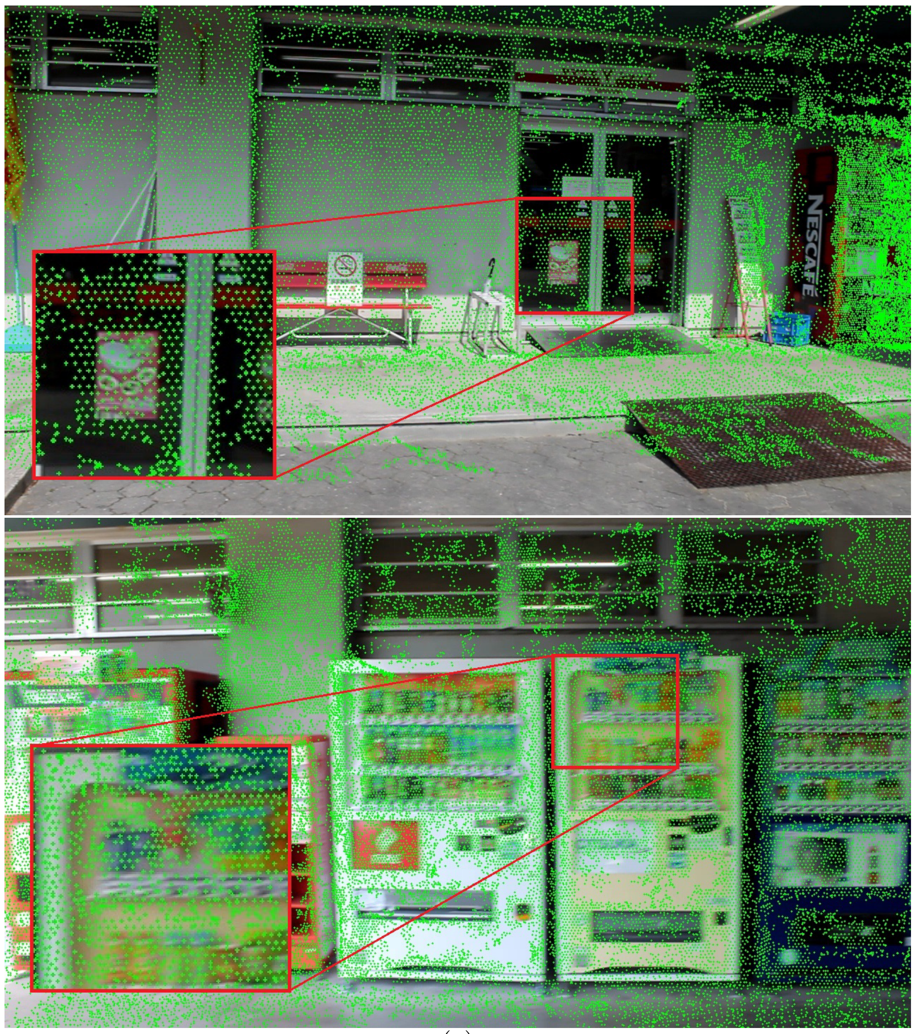
Figure 2.16 shows another example of an outdoor scene of more complex environment and different scale compared to the previous indoor scene. In this scene there are objects of different size (i.e., different scales), under different lighting conditions, and with different appearances, while the indoor scene only has small objects under controlled lighting condition. With this experiment, we demonstrate that the proposed method works well on different scenes. The total number of training images are 26 (2256×1504 in size). Figure 2.16(a) shows some training images for 3D reconstruction. Figure 2.16(b) shows the reconstructed 125,874 3D points. We set th_v as 4 in this case. Finally, we get 410 3D keypoints. The query videos are taken in different resolutions (1280×720 and 640×480). Some snapshots of the tracking query videos are shown in Figure 2.16(c). These results are also contained in our video on YouTube. Of course, the accuracy of



(a)



(b)



(c)

FIGURE 2.16: Estimation results for the outdoor scene. (a) Some training images. (b) Reconstructed 3D point cloud. (c) Two snapshots of query videos with 3D points superimposed.

the proposed method depends on the specific scenes. The experimental results shown here, however, demonstrate that the method works in two completely different situations: scenes of different scales (from desktop to walk space), lighting conditions (artificial lights and sunlight), and object appearance. Therefore, we expect that the proposed method would also work well for other scenes. One of the factors affecting the performance of the method is the number of training images. The proposed 3D keypoint detection will fail if only a few and sparse training images are available, while if the number of training images is huge, the method could become unfeasible because most of the 3D points will be chosen as keypoints. Our future work includes finding the optimal balance between the size and goodness of the training image dataset.

2.4 Summary

In our research, we would like to build a surveillance system for change detection. Change detection is important because that disasters and accidents can happen with high probability if the wide areas have temporary change. But it is very different for operators to install expensive and heavy equipments in a widely outdoor area. So the detection goal is divided to two sub-goals: online approach and offline approach. The online approach is a kind of 3D-2D method used for a regular observation of a target scene to quickly detect changed areas.

However, 3D-2D detection need to do 3D-2D matching first. In this chapter, we have proposed a method for detecting 3D keypoints in a 3D point cloud for robust real-time camera tracking. By detecting a sparse set of 3D keypoints from a large number of 3D points, the proposed method enables us to use 3D-2D matching for a fast camera pose estimation. A 3D keypoint is defined as a point that has corresponding 2D keypoints in many training images. The descriptors of these 2D keypoints are used as a descriptor of the 3D keypoint. After detecting 3D keypoints, matching between them and 2D keypoints extracted from query images is performed and camera projection matrices of the query images are estimated. Experiments for two 3D point clouds from two real scenes show promising results for the proposed method.

Our method still has some limitations. First, if the camera takes a scene from a very different viewpoint from the training images, the camera pose would not be estimated accurately. In our experiments, we captured sufficient training images to cover the real space. Second, if the number of training images is huge, we need to set the thresholds carefully. Third, if the 3D keypoints are too many, matching takes much longer time. Fourth, conversely, sometimes there are too few 3D keypoints to register query images. In our future research, we will find a more systematic way for choosing parameters to improve our method.

3D keypoints is the robust way for 3D scene and 2D image to estimate camera pose. It is also important when implementing 3D-2D matching. In next chapter, we will focus on our online change detection approach. In our online detection approach, a client-server system is presented. A quickly change detection result is expected by using 3D point cloud and 2D images which captured by using mobile phones and tablets. Therefore, 3D-2D matching followed 3D keypoints detection and description will be one of the most important parts of the online sub-goal.

Chapter 3

3D-2D based 3D Scene Change Detection

As we know, change detection embedded in a surveillance system has became more popular and important. Any obvious change must be alerted to protect wide environment. However, it is not practical to observe the wide areas all the time. Therefore, a reasonable sub-goal pursued in this thesis is a system, used for a regular (e.g., monthly or bi-monthly, etc.) inspection of a target scene (e.g., blocks on a coast), to assist users by visualizing candidate parts of the scene in order to identify which part of the scene could have changed during a particular time period.

In this chapter, we describe an online approach to detect temporal change by using a 3D scene geometry at time A (*reference*) and the image of the same scene at time B (*query* or *test*). To quickly detect changes and visualize the change taking place for operators at regular observation, our method uses 3D-2D matching between a 3D reference scene and 2D query images followed by 2D-2D feature comparison between 2D training and query images, which is superior to either of 3D-3D or 2D-2D matching. In the case of 3D-3D matching [49–51], two 3D scenes are registered for change detection. This may be able to provide a detail comparison between reference and query 3D scenes, however, this approach is not suitable because online and real-time 3D reconstruction is very difficult and still a challenging problem. In the case of 2D-2D matching, images taken by a fixed camera are compared to detect changes. A huge amount of research on change detection or moving object detection has been carried out mainly for surveillance cameras [5–7].

This approach is also not appropriate for our purpose: a number of fixed cameras are needed to cover the whole scene of the observed place, hence, it is impractical if in particular the scene is wide and large. If the scene is observed by a hand-held camera for taking a number of 2D images, the problem above may not arise. However, reference and query 2D images are very difficult to register unless they have a large overlapping area in each pair of the images. In contrast, our approach uses the combination of 3D-2D and 2D-2D matching. We assume that a 3D scene geometry is given but only at time A as a reference 3D scene, hence no 3D range finder is necessary. At time B (query), a hand-held camera is used to take images of the same place, and the reference 3D geometry is used for 3D-2D matching for camera pose estimation (see Chapter 2). This enables us to perform a robust 2D-2D matching.

In the following sections, related work on scene change detection is reviewed in section 3.1. Section 3.2 outlines the concept of the proposed method. The algorithm of online method for change detection is shown in section 3.3. Experimental results are given in section 3.4.

3.1 Related Work of 3D Change Detection

Recently, several 3D change detection methods with a laser range finder have been proposed for environment monitoring and robot navigation. Goncalves et al. [49] proposed a method to visualize 3D displacement map of two 3D laser-scanned scene point clouds. Ryle et al. [50] presented a detection method of fast background subtraction of 3D geometry with voxel segmentation. Neuman et al. [51] also proposed a method for online segmentation-based change detection for mobile robots. These methods need a 3D range finger to obtain 3D geometries both for reference and query, hence it may not be used near seaside coasts or mountain slopes.

With a monocular video, Eden et al. [52] proposed a method for matching 3D line segments by using computational complexity reduction with incidence relations among lines to detect changes. But this method is not applicable to round-shape or tapered tube objects such as a wave dissipating block, or natural scenes where straight lines rarely happen.

For omni-directional videos, a method for frame alignment of two video sequences followed by change detection with color differences was proposed by Sato et al. [53]. Because there is no 3D information in the panoramic videos, it fails when scene objects are relatively close to the camera and then images have large parallax.

A method proposed by Pollard et al. [54] updates probabilistic voxel models by using a new image and models changes by a probability of seeing observed pixel color. This method can deal images with arbitrary (but known) cameras, however, it strongly relies on GMM color models and hence is not adequate to objects with similar colors.

Taneja et al. [55] proposed a method for computing 2D inconsistency maps combined with MRF and project color difference of images back to 3D surfaces for 3D scene change detection. The irrelevant changes such as walking people, cars and vegetation are excluded by classifying those changes. This method uses color information of images which is not stable for illumination changes.

Our approach is similar to the work of Taneja et al. [55]: to utilize 2D images with 3D scene geometry. Our key contribution is to use local feature descriptors [30] for detecting changes instead of color information, because those are well known to be robust to illumination changes, and invariant to many transformations including rotation, translation, and scaling.

3.2 Change Detection by using Feature UnMatching

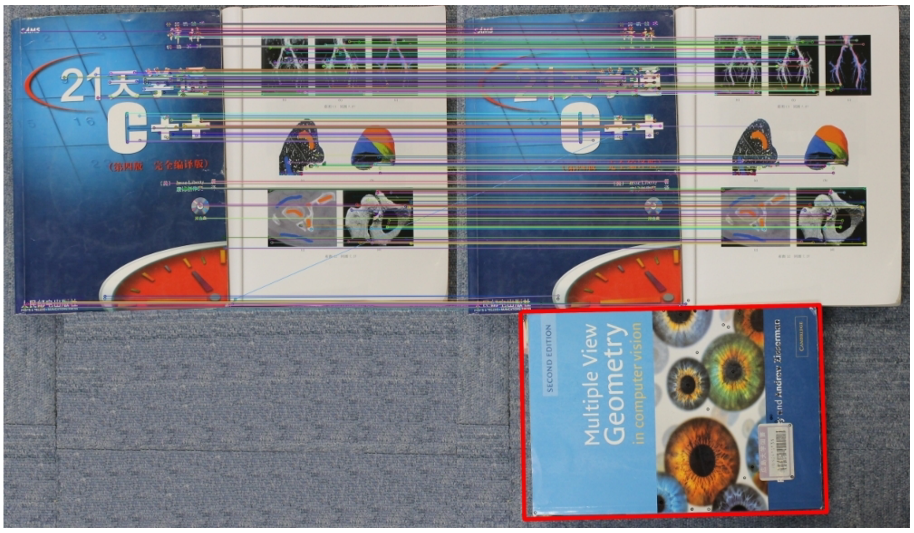


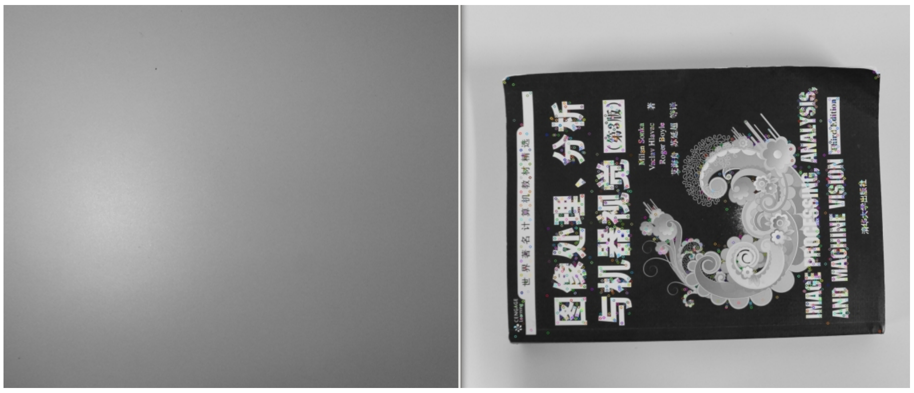
FIGURE 3.1: An intuitive and illustrative example of change detection. The left image is a reference and the right image is a query with a change (a book appears). Keypoints with correspondences (colored lines) are in a region without changes, while points with no correspondences are in a region of change.

The proposed online method consists of two parts: camera pose estimation by 3D-2D matching [56], and change detection by 2D-2D matching. Since our online method focuses on the second part, here in this section we explain the concept how we detect changes by using 2D image feature descriptors in the training (reference) and query images. Now we assume to have a reference 3D point cloud (at time A) and associated 2D (*training* or *reference*) images (at time A) used for 3D reconstruction. The problem is to find changes in a *query* image (at time B) .

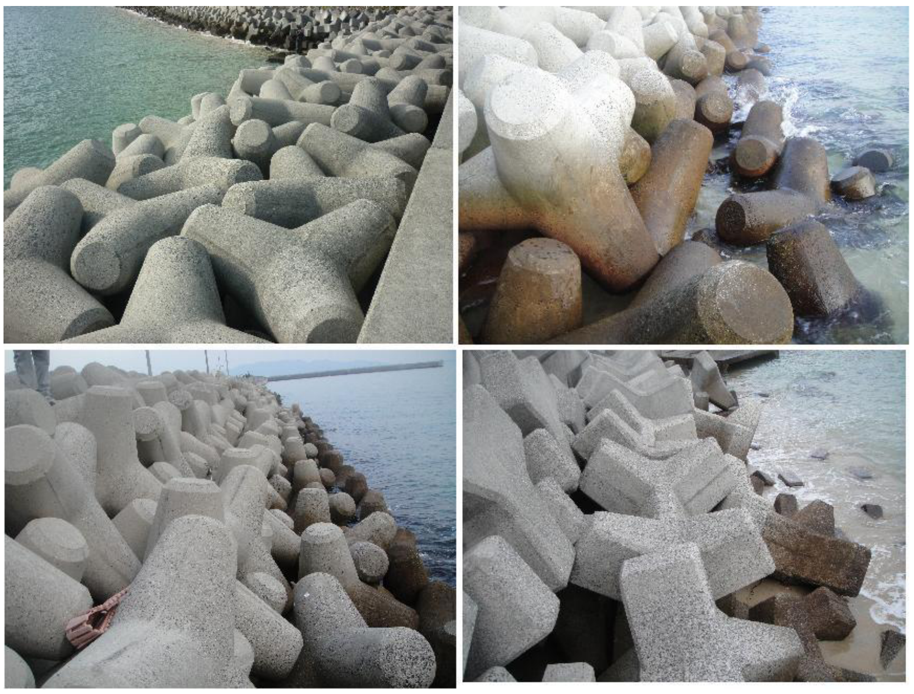
A region of change is defined in this chapter as *a region in the query image in which feature points* (e.g., SIFT keypoints) *do not correspond to points in the training images.* Figure 3.1 illustrates this concept with a very simple situation where on the left is shown a training image (reference) and on the right a query image with a change, in this case a book appears. Here, the feature points on the book have no corresponding points in the reference image, hence those are identified as a change in the scene. Even if the reference and query are interchanged (i.e., the book disappears), the region is still going to be detected.

This idea may look to be so limited that only appearing and disappearing objects can be detected, however, this is not true. Any movement of a rigid object would cause the appearance/disappearance of part of the surface of the object or the appearance of part of the background that has been occluded by the object. Therefore, the proposed concept, although quite simple, is still expected to work well s situations where objects move as well as when those appear or disappear.

To detect those points as changes, we need to have some assumptions. The first assumption is that the scene has rich texture to ensure that many feature points are extracted. If the background has less texture, no change can be detected when an object disappears

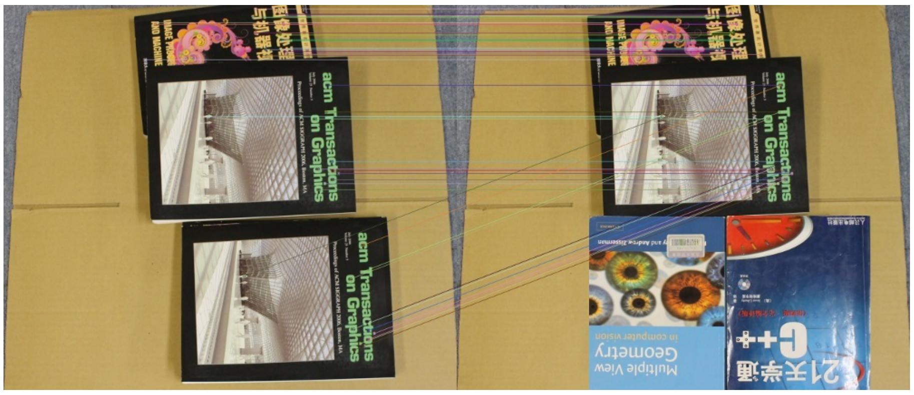


(a)

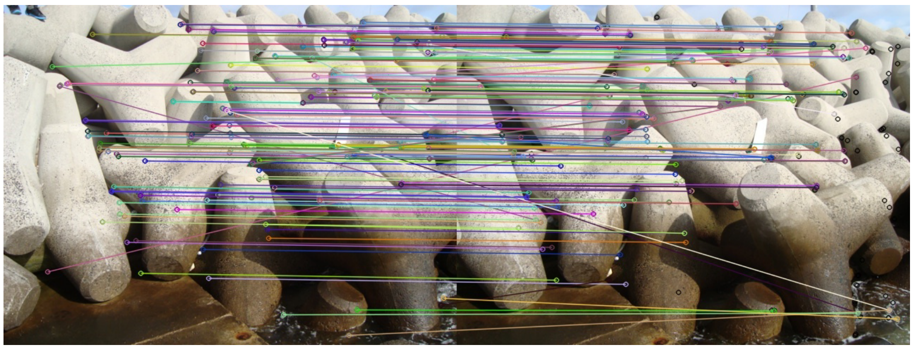


(b)

FIGURE 3.2: (a) An example with less textured background. (b) A real scene example with rich texture.



(a)



(b)

FIGURE 3.3: (a) An example of repeated texture from similar objects. (b) A real scene where keypoints between similar images are well matched.

(see an example in Figure 3.2(a)). However, this case rarely happens in our task because real scenes like Figure 3.2(b) have always rich texture.

The second assumption is that the scene does not have repeated texture. If a feature descriptor matches to other place in the scene, the proposed method fails (as illustrated in Figure 3.3(a)). Another common example of repeated texture is the windows on the walls of buildings, however, in practice those are not the target of change detection, and real scenes like Figure 3.2(b) always have non-repeated texture. Even if those images look to be similar to each other and repeated at first glance, standard feature descriptors like SIFT [30] can be matched very well with few outliers (see Figure 3.3(b)).

3.3 Change Detection and Visualization

In this section, we describe the proposed change detection and visualization method. The main idea is to find 2D keypoints in a query image with no correspondence in the training images . To this end, we first estimate the camera pose (i.e., the projection matrix) of a query image by using 3D-2D matching (details in Chapter 2 and [20, 56]). Then, we find the *nearest* image among the training images: the closest training image to the query image. Using the nearest image, 2D points with no correspondence are detected by performing 2D-2D matching. Finally, we visualize a region of change with those 2D points by projecting the 3D point cloud of the scene.

3.3.1 Finding the Nearest Image

As shown in Figure 3.4, we want to find the closest training image to the query in terms of the camera position and orientation. However, just comparing the camera positions is not enough for our task because often two cameras with different positions and orientations can capture very similar images of the same scene. In our method, we estimate the nearest image in two steps.

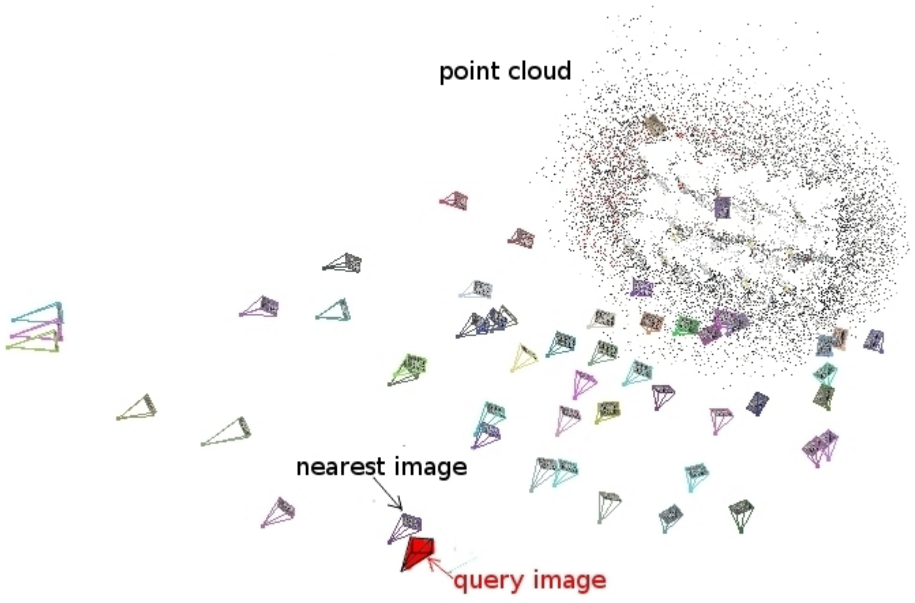


FIGURE 3.4: An example of a nearest image to the query image.

- First, we define and calculate the error between two (reference and query) camera positions and orientations. Let $\hat{P}$ be the camera projection matrix for the query image and P for the training image. Those projection matrices are decomposed into rotation matrices $\hat{R}$ and R, and translation vectors $\hat{\boldsymbol{t}}$, $\boldsymbol{t}$. Translation change is defined as the relative error as follows:

$$e(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \frac{\| \boldsymbol{t} - \hat{\boldsymbol{t}} \|}{\| \hat{\boldsymbol{t}} \|}. \tag{3.1}$$

  Rotation change is defined as the distance of two rotation matrices [40] as follows:

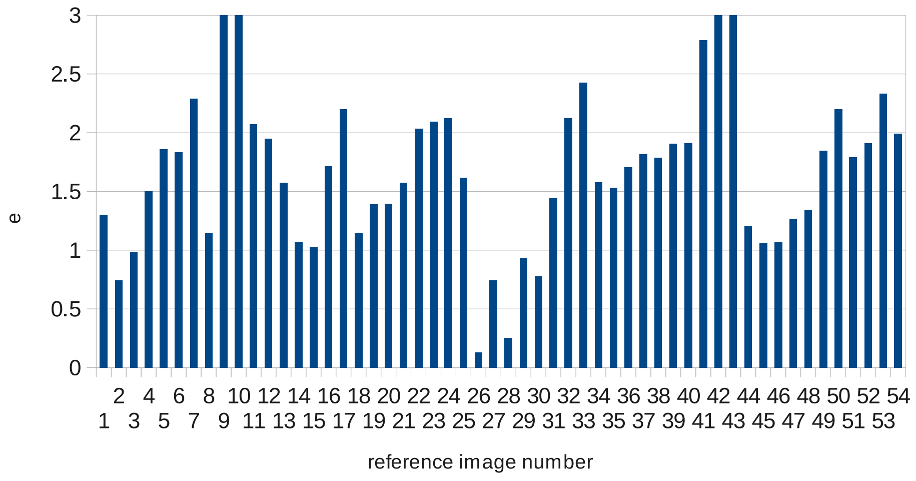
$$e(R, \hat{R}) = \cos^{-1}\left(\frac{\mathrm{tr}(R\hat{R}^T) - 1}{2}\right). \tag{3.2}$$

  As the change of camera projection matrices involves both rotation and translation changes, we define the distance (change) of two camera projection matrices by:

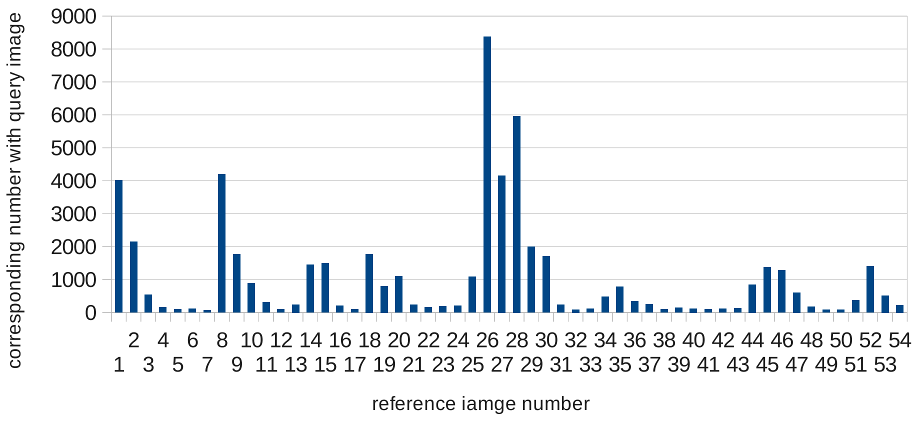
$$e(P, \hat{P}) = e(\boldsymbol{t}, \hat{\boldsymbol{t}}) + e(R, \hat{R}). \tag{3.3}$$

- This distance, however, provides just rough candidates for the nearest image. Therefore, next we count the number of corresponding points between the query image and each of the training images (not for all training images but only for the several "closest" training images in terms of the distance Equation 3.3. Then the image which has the largest number of corresponding points is selected as the nearest image.

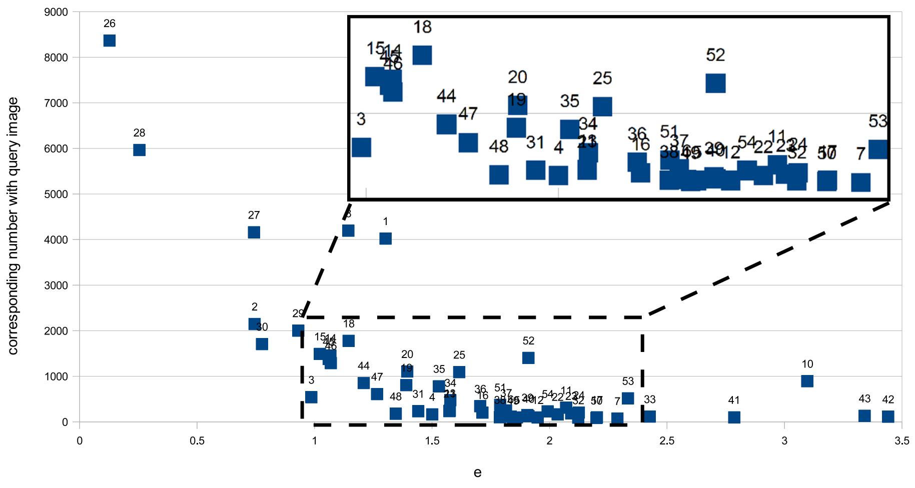
Figure 3.5 illustrates this concept. The distances $e(P, \hat{P})$ between a query and each of 54 training images are shown in Figure 3.5(a). Figure 3.5(b) shows the number of correspondences between the query image and each of 54 training images. In Figure 3.5(c), the scatter plot is made based on Figure 3.5(a) and (b). The label at each point shows the training image number. In this figure, $e(P, \hat{P})$ should be as small as possible. If two or more reference images have the same distance (in this example, image 27 and image 2), we select one which has more corresponding points (i.e., image 27). We expect that the minimum number of correspondences is larger than about 2000 (it needs to be large enough to cover the query image), and in fact the closest 4 images 26, 28, 27 and 2 have more than 2000 correspondences.



(a)



(b)



(c)

FIGURE 3.5: (a) Distances e between the query and each of the training images. (b) The number of corresponding points between the query and each of the training images. (c) Scatter plot of the distance and the number of correspondences.

This result might suggest that the proposed two steps are not necessary and just either of them is enough. However, the combination of the two steps is much better than either of them because the first step (checking the camera projection matrices distances) greatly reduces the computation time in the second step (counting matching points), and the second step allows us to find the training image with the most similar appearance to the query.

3.3.2 Change Area Detection

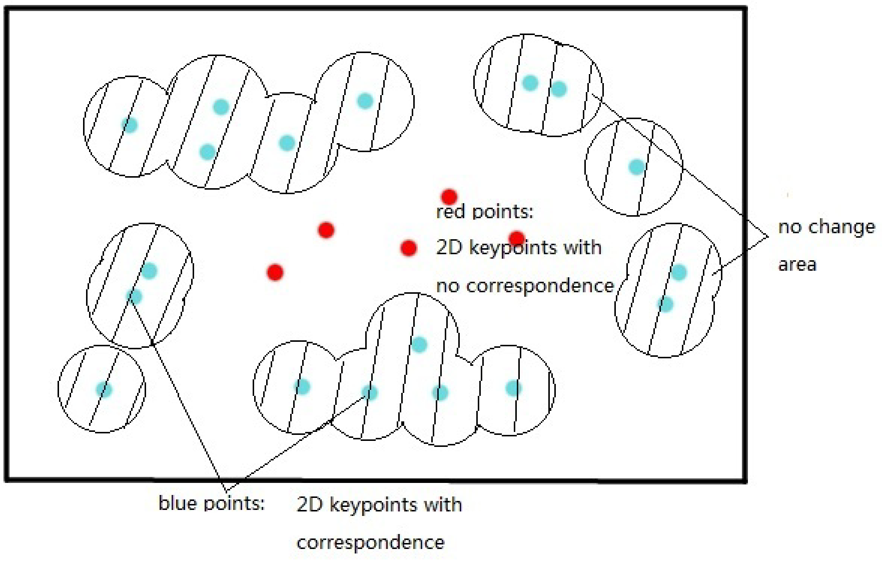


FIGURE 3.6: Change area detection. Blue points are 2D keypoints with correspondences, and a “no change” area is defined around those points.

As explained in section 3.3.1, a region of change is detected by comparing the query and the nearest images. Figure 3.7 shows an example for change detection. Figure 3.7(a) shows the correspondence between the query and the nearest image. In Figure 3.7(b), the 2D keypoints in the query image that have corresponding points in the nearest image are shown in blue. Those are considered not to be changed, therefore, we mark the corresponding areas as “no change”. In other words, we define that a pixel is in a region of change when the distance from the pixel to the closest “no change” 2D keypoint is larger than a threshold (see Figure 3.6). 2D keypoints that have no correspondences are shown in red, and those are considered to be in a region of change. Because not the whole query image could be covered by the nearest image, the top and left sides of the query image in Figure 3.7(b) are also detected as a region of change. This problem

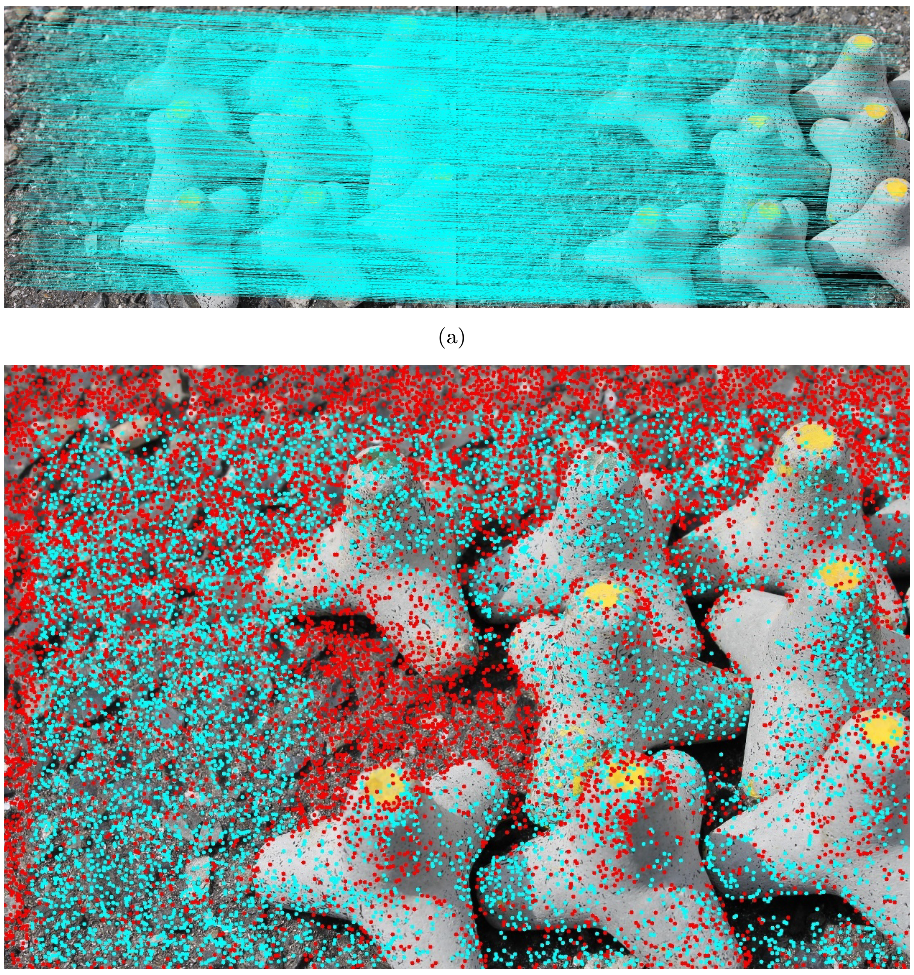


FIGURE 3.7: Matching between the query image and nearest image. (a) Corresponding points are marked by blue lines. (b) 2D keypoints that have correspondence (blue) and no correspondences (red) in the query image.

is not so serious because an observer can easily recognize that it is an artefact of the method, hence, it is left for future work.

3.3.3 Visualization

Once we have found the change detection area, we visualize the region by projecting the 3D point cloud of the scene only inside the region. In Figure 3.8, we shows a comparison with different estimated camera poses. The top image shows the detected change area, the three images in the left column are pose estimation results (visualized by projecting the 3D point cloud). The top image uses the estimated (correct) camera pose, while in the bottom image the pose was rotated by 30 degree about the y axis (for simulating a

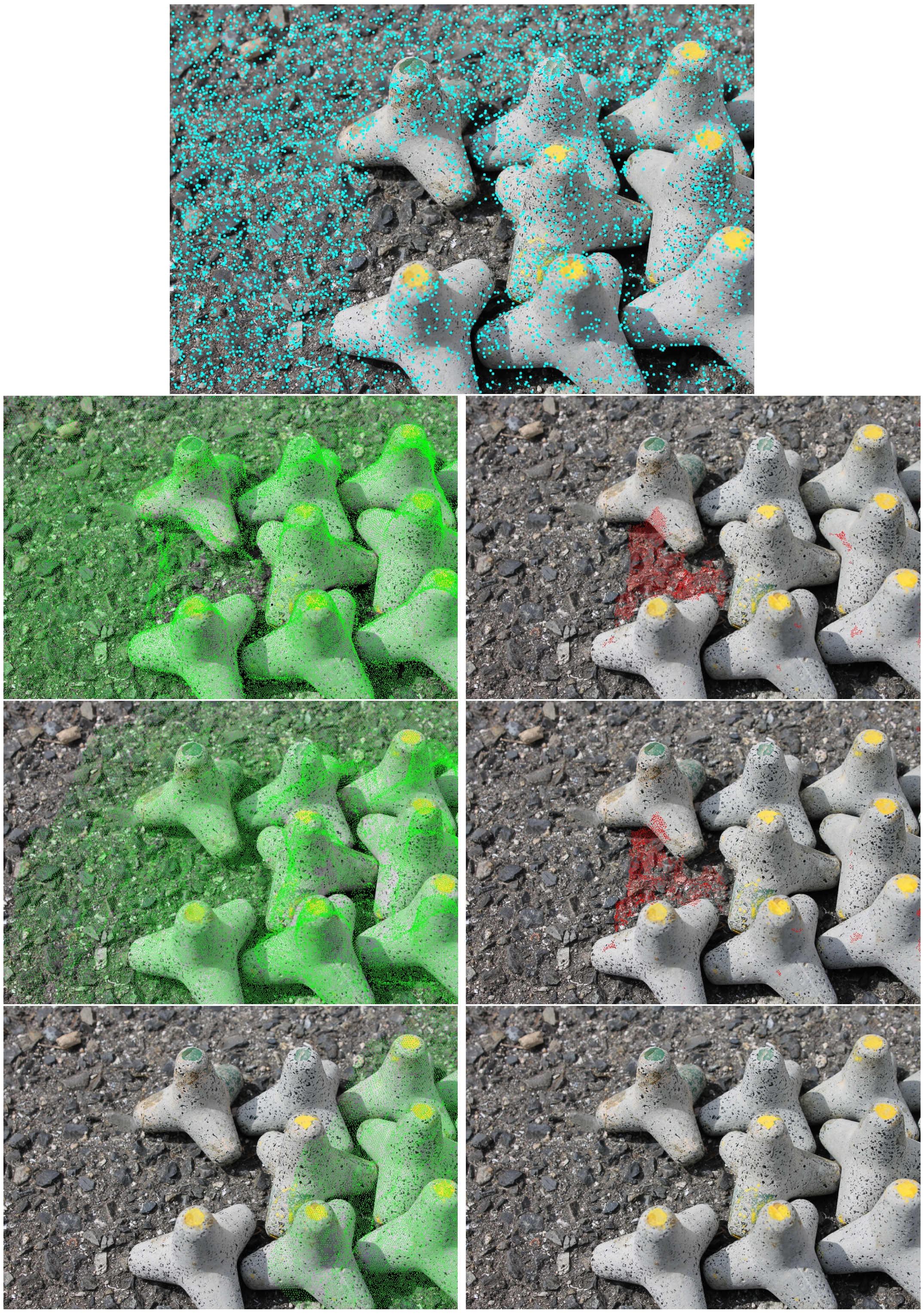


FIGURE 3.8: The relationship between camera pose and change. (top) detected change area, (left column) correct, 10 degree error and 30 degree error camera poses, (right column) correct and incorrect visualization results.

very inaccurate camera pose). Notice that for the middle image with 10 degree rotated camera pose (simulating small error), the changed area is visualized correctly, while the 3D shape may not appear correctly. The corresponding visualization of the detection results (images in the bottom row) show that the correct camera pose is important for better visualization. We have seen in all experiments with the dataset used in this chapter that estimated camera poses were correct enough as shown in the left images of Figure 3.8.

3.4 Evaluation

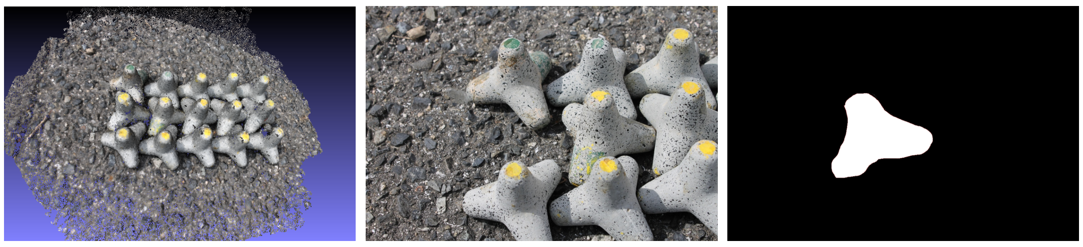
3.4.1 Small Blocks

To evaluate the proposed method, first we show quantitative results for experiments with small blocks (about $5cm \times 5cm \times 5cm$) set on the ground (the area size where the blocks are set is about $35cm \times 15cm$). We took 54 images of the scene with a camera about 30cm away from the blocks with a variety of viewpoints. Then these 54 training images (2256×1504 in size) at time A were used to reconstruct a 3D point cloud (Figure 3.9(a)). A query image (of the same size with the training images) at time B taken by different viewpoint is used to detect change. In this dataset, a block was taken away from the scene to represent the change. A hand-marked ground truth is used for calculating true positive and false positive rates of projected 3D points. Figure 3.9(b) shows the detection and visualization results for different thresholds.

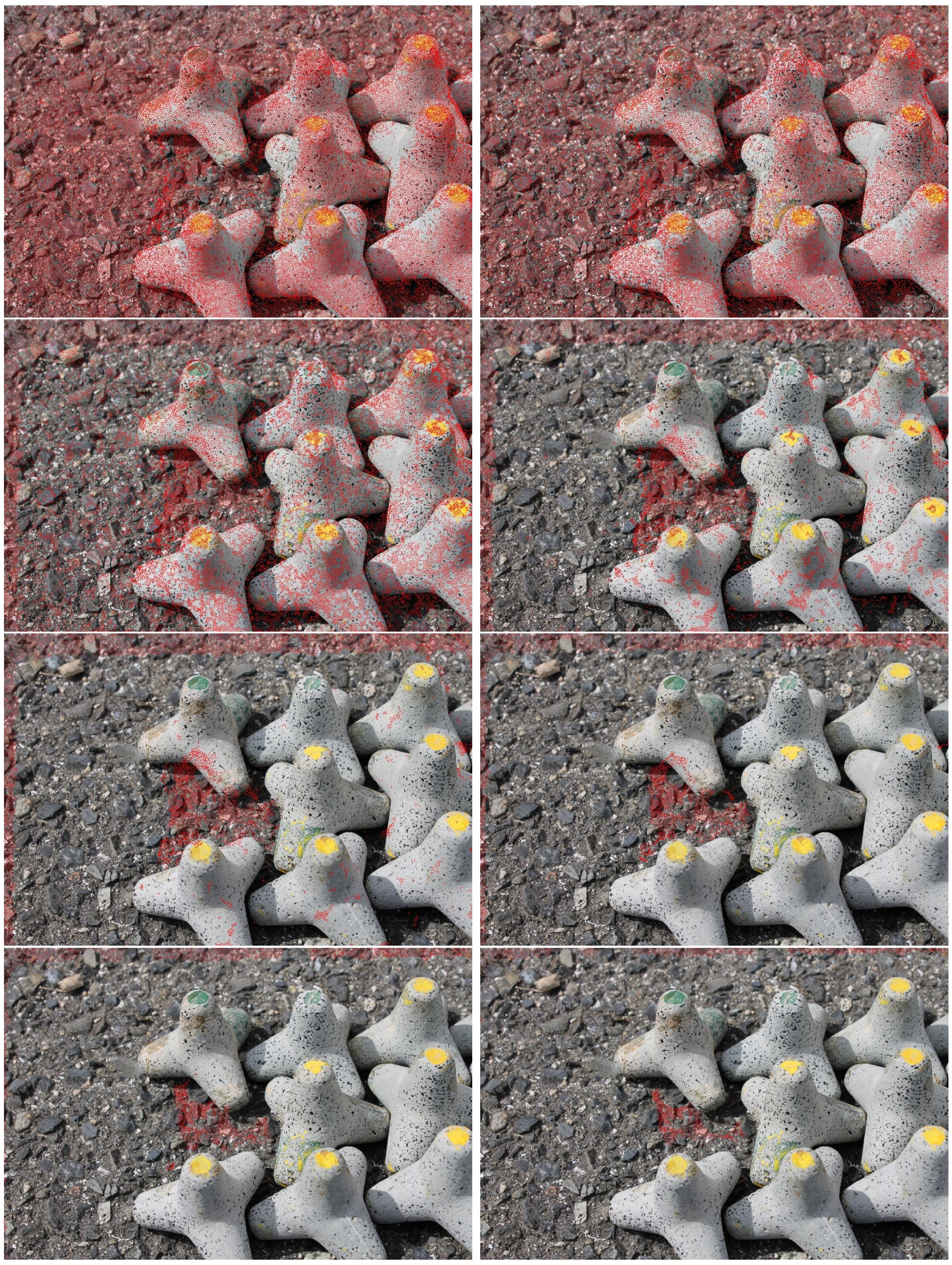
We plot receiver operating characteristic (ROC) curves which show the fraction of correctly detected changes, the True Positive (TP) rate, against the fraction of falsely detected (projected 3D) points, the False Positive (FP) rate. In our case, the number of points projected inside the white area of the ground truth image is denoted by P, and the number of points projected outside the white area is denoted by N. Similarly, the number of points detected by our method inside the correct white region is denoted by TP, and the number of detected points outside the correct white region is denoted by FP. Then the TP rate is defined as the ratio $\frac{TP}{P}$ and FP rate as $\frac{FP}{N}$. Each point in Figure 3.10(a) corresponds to a different threshold (90, 70, 50, 30, 20, 10, 5 and 0). Based on the ROC curves, or this dataset it is best to set the threshold between 20 and 30.

The figure also shows results when the 1st nearest image is not available but the kth ($k > 1$) nearest images is used instead. This illustrates how sensitive the proposed method is to the selection of the nearest image and how the detection result degrades when the “nearest” image is far from the ideal one. This shows that even when the 4th nearest image is used, the proposed method still works as well as when the 1st nearest image is used.

Notice that in Figure 3.10(a) the 2nd nearest image performed better than the 1st nearest image. The reason is that the query image cannot be always covered by the

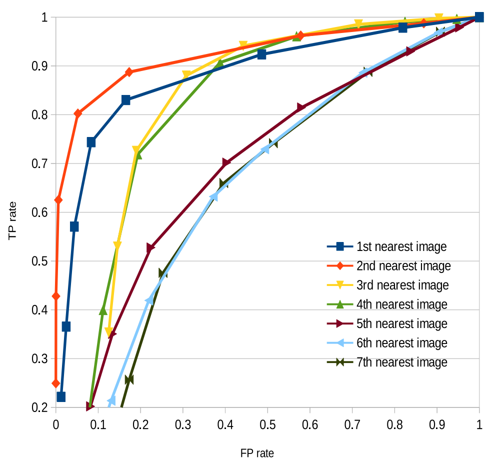


(a)

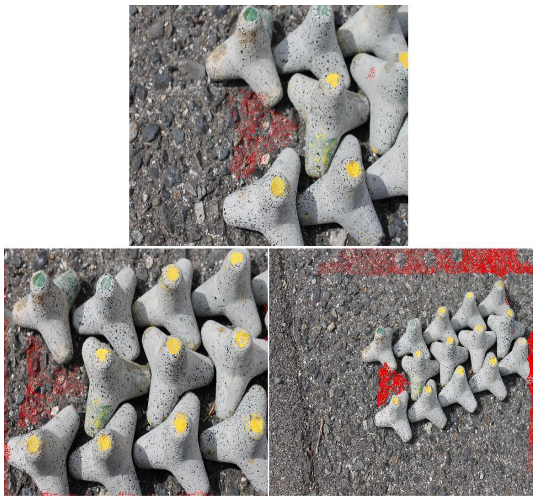


(b)

Figure 3.9: Experimental results for a small block dataset. (a) Point cloud, a query image and its hand-marked ground truth. (b) Results for different thresholds (from left to right: 0, 5, 10, 20, 30, 50, 70 and 90).

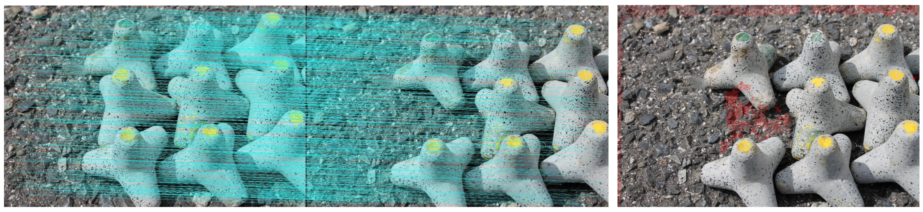


(a)

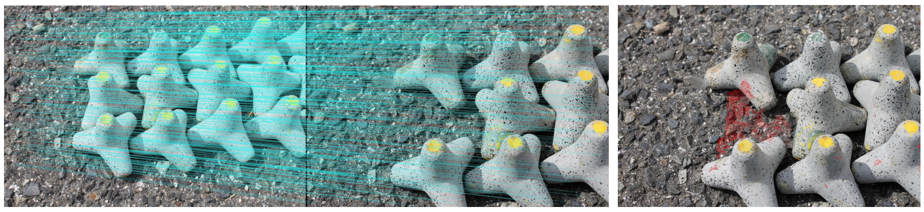


(b)

FIGURE 3.10: Experimental results for a small block dataset. (a) ROC curves. (b) Detection results for other query images.



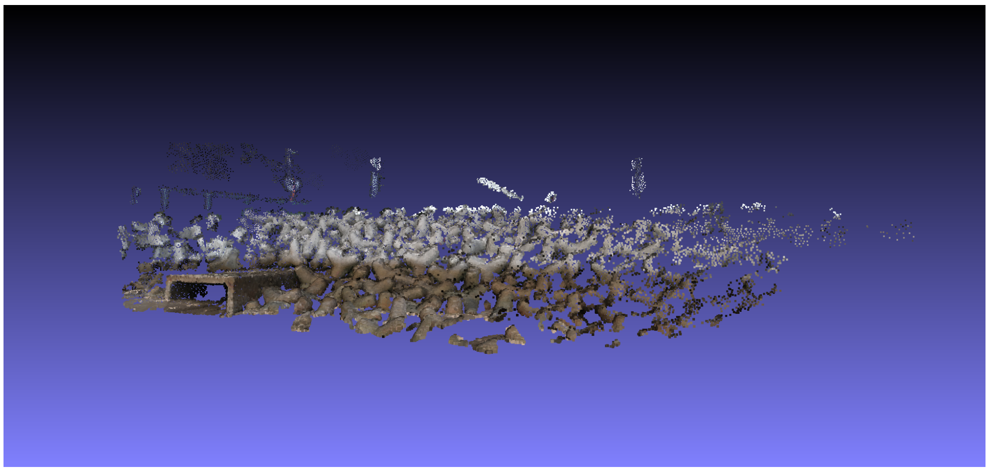
(a)



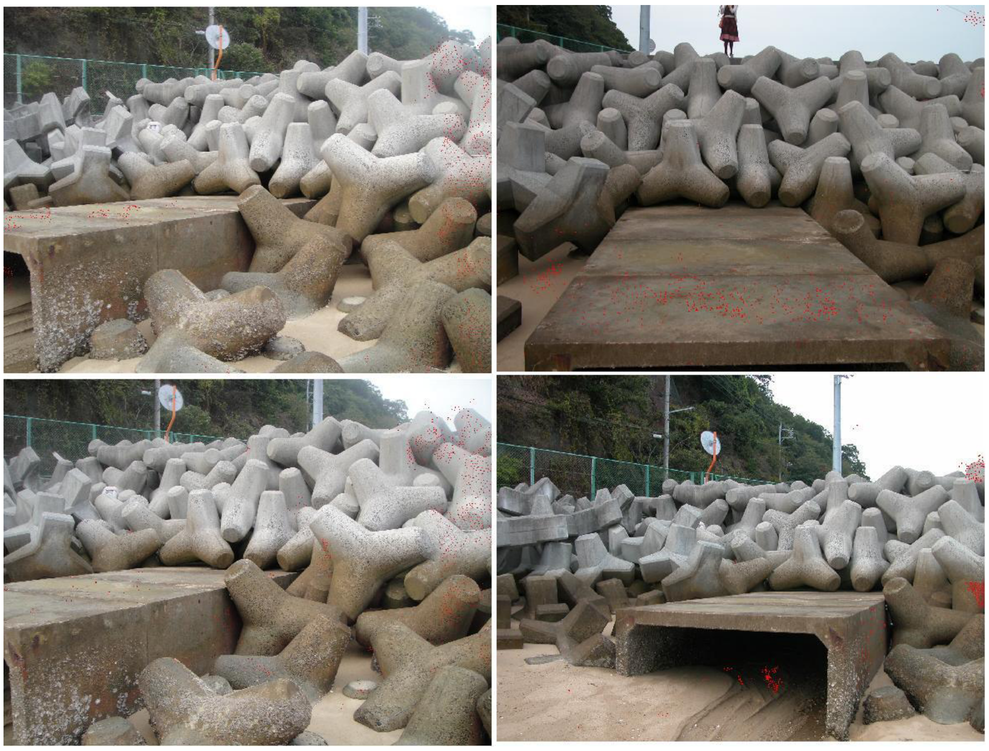
(b)

FIGURE 3.11: The relationship between nearest images and query image, (a) the matching and change detection results by using 1st nearest image, (b) the matching and change detection results by using 2nd nearest image.

nearest image due to the difference of the fields of view (FOVs) as shown in Figure 3.11. There exist a kind of "border effect" shown as the red points at the top border in the query image when the 1st nearest image is used (Figure 3.11(a)). This effect does not appear for the 2nd nearest image (Figure 3.11(b)) because it has much larger FOV than the query image. We have planned to reduce the border effect in order to further improve the performance.



(a)



(b)

Figure 3.12: Results for outdoor blocks. (a) 3D scene point cloud. (b) Change detection results for different query images.

3.4.2 Outdoor Real Blocks

Our research target is to detect change in a real outdoor environment. In this experiment, a set of real block images were used for detection. 23 training images were taken in October 2009 and query images were taken in January 2010. The results are shown in Figure 3.12. Because no disasters or accidents had occurred in this time period, no change should be detected. Apart form a few wrong detections, the proposed method works reasonably well on this real dataset. We will continue to observe this site for finding any suspicious changes.

3.4.3 Discussion of the Evaluation

The proposed method takes 20 seconds for detection of a single query image, while Taneja's method [55] takes about 60 seconds. This is because our method uses 3D points mainly for visualization shown on a 2D query image. However, Taneja's method focuses both on 2D images and 3D points and shows detection results in the 3D point cloud, which takes much longer time to process. Therefore, our method is reasonably faster than 3D point-based detection methods.

3.5 Summary

In order to detect unusual temporary change in a widely area, we set our research goal to be a change detection surveillance system development. But consider to the complicate situation of target surveillance environment, such as coastal areas, mountain slopes and highways, it is necessary to divide the research goal into two sub-goals: online approach and offline approach. The online approach is used for real-time change detection. The users can capture the query image of a changed scene by using mobile phones and tablets. Then the detection result is computed and returned by a powerful server via WiFi which can ensure the quickly processing.

In this chapter, we have focused on a quickly change detection method which is the online real-time change detection approach. Therefore, we have proposed a method for 3D-2D based change detection. To find a region of change in a query images, first, the nearest image is selected. Then, the matching between the query and the nearest images is performed to find 2D keypoints with correspondences and detect a region of change. The detected region of change is visualized by projecting the 3D points back to the region. Our future work includes reducing wrong detection by using spatial information such as smoothness or coherency, and implementations on mobile devices.

Our online approach is very useful in a widely area because it is portable for operators and fast for computation. However, the online approach returns potential changed area which are not always high accuracy. Therefore, in next chapter, we will introduce our offline detection approach. Since we can have the 3D point cloud at time A and the reference images at time B, we can provide such data to the offline approach. Then, a 3D-3D based high accurate point clouds alignment will be implemented.

Chapter 4

Scale Alignment of 3D Point Clouds

As we introduced in Chapter 1, our offline model need a method to improve the detection accuracy of change areas results. In this chapter, we propose a method for estimating the scales of point clouds in order to align them. This is important because once point clouds are transformed into the same scale, feature descriptors that are not scale invariant such as spin image [13, 14] and NARF [15] can be used for 3D registration. Our proposed method performs Principal Component Analysis (PCA) to generate two sets of cumulative contribution rate curves. Then the *scale ratio* is defined as the scale factor that is used to re-scale one point cloud to the other, which can be found by registration of two sets of curves, instead by registration of the point clouds themselves.

Actually, in many cases we do not need to estimate the scales of the point clouds because absolute scales can be obtained by using 3D capturing devices such as laser range finders or time-of-flight (TOF) cameras or even consumer level cheap depth sensor devices like Kinect [29]. Such devices provide a range image or a depth map, in which a depth value z is given at each (x, y) coordinates with absolute metric [32–35]. Alternatively, we can put calibration objects into a 3D scene to obtain the absolute length in the 3D point cloud.

However, there are many practical situations where it is difficult to apply the strategies above because of the following reasons.

Research Challenge In order to detect unusual changes quickly, we have developed a method which registers a 3D point cloud of a past scene with 2D images of a current scene (see Chapter 1 and [23]). These 3D point clouds are generated by structure-from-motion (SfM) [26–28] with a lot of 2D images taken by human operators walking around the coast. Although a quick change detection has been achieved by the developed system, a more accurate and detailed analysis of the change in the scene requires the registration of 3D point clouds from the past and current scenes.

To this end, we have to deal with two problems: how to capture 3D point clouds in target scenes, and how to align them.

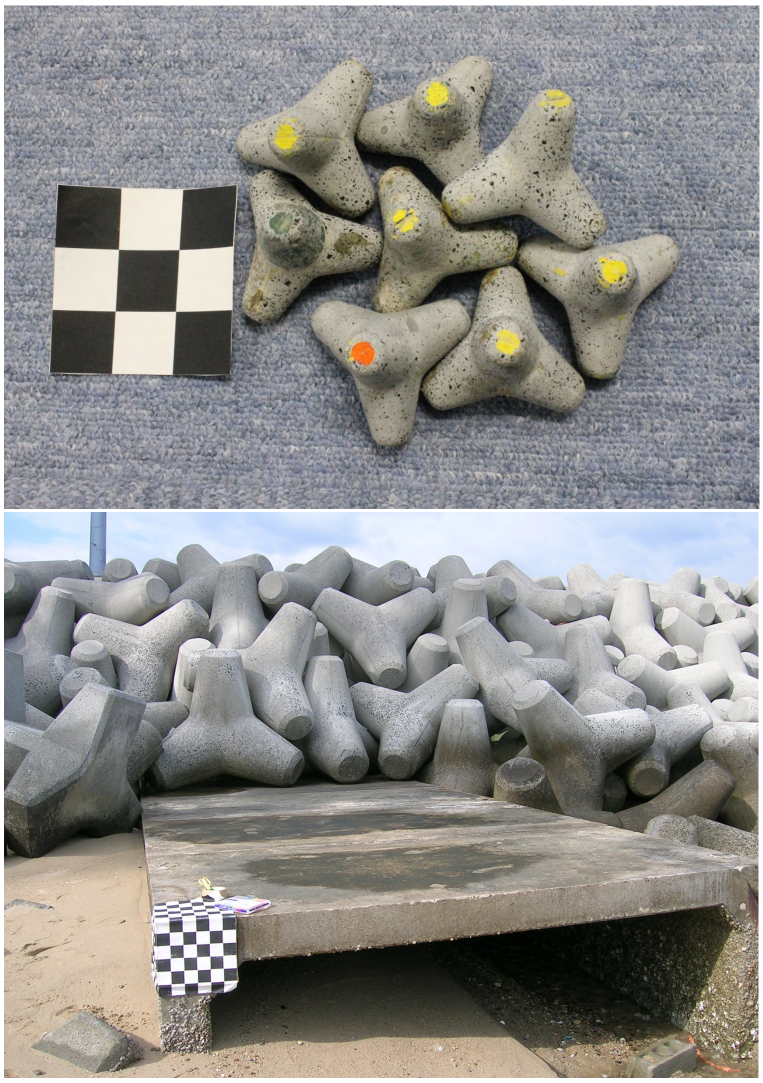


FIGURE 4.1: Examples of target scenes and calibration objects. (Top) Small objects on a desktop. (Bottom) Large blocks on a coast.

Target scenes The devices mentioned above, that can capture the 3D shape of a scene, are not appropriate for our task. This is because the observation places are very specific, like coasts or seashores, where the waves roll into the sand and the ground becomes slippery. Expensive devices such as range finders are therefore not expected to be used. Consumer level devices (like Kinect, for example) are cheap but not useful for large scale scenes or under strong sunlight. Since the target objects we consider are usually larger than three meters in height and the operators would capture the 3D scene during the daytime, such devices are not a good choice.

The best choice is therefore to use cheap and hand-held digital cameras to reconstruct the target scene with SfM. However, is this case we need to estimate the scales of the 3D scenes because SfM does not provide the scales directly.

A calibration object can be used for obtaining the scale of a scene, however, this is not enough. Figure 4.1 shows examples of commonly used calibration objects, black-and-white checker boards, set in target scenes. These checker boards are useful in particular for a desktop-size small scenes like the one shown in the top image of Figure 4.1. As the target object size becomes large, as in the bottom image of Figure 4.1, the error in the estimated scale would increase linearly, which would significantly affect the result of the registration.

Our approach Many methods have been proposed to align two 3D point clouds, which include Iterative Closest Point (ICP) based methods and descriptor-based methods. Our target scenes are however difficult to be registered by those methods. Figure 4.24 shows an example of 3D point clouds of one of the target scene. ICP-based methods would not work well for this dataset as it does for a simple and clean dataset such as the Stanford Bunny [57]. Descriptor-based methods also do not work as expected because the scene has lots of clutter that makes it difficult to extract invariant descriptors stably. Furthermore, both types of methods have the same drawback, that point clouds of different scales are difficult to register, which will be discussed in the next section.

The approach presented in this chapter divides the task into two steps: scale estimation and registration. One possible way is to estimate the scale of each point cloud separately. In this case, each point cloud has its own scale that is something like the size of a scene. We call it a *keyscale* [58], which is a representative scale and is defined for a given 3D point cloud as the minimum of the cumulative contribution rates of PCA of descriptors (spin images [13, 14]) over different scales. Then we can re-scale one point cloud to another by using the estimated keyscales. However, this method has two weak points. First, finding the minimum might not be stable due to noise. Second, we need a very small step size for an accurate keyscale estimation, which might be very expensive to compute.

As an alternative, we propose a method for directly estimating the *scale ratio*, the ratio of scales, of two point clouds [25, 58]. We assume that the two sets of curves of the cumulative contribution rate of PCA differ only in scale. With a given initial search range of the scale, the proposed method aligns the two sets of curves by using a concept similar to ICP in order to estimate the scale ratio. Once the scale ratio has been estimated, we can use existing ICP-based or descriptor-based alignment methods.

4.1 Related Work of 3D Scale Alignment

Here we discuss several methods that could be used to estimate the scales of or align 3D point clouds.

Some simple ways Possible simple ways might be the use of one of the sides of the bounding box of a 3D point cloud, or the standard deviation of the point cloud distribution. These values can be used to align the scales of two point clouds, however this would work only when the 3D point clouds contain a single object or the same part of a scene. Yet another way is to use mesh-resolution [13, 14], which is the median of the distances of all edges between neighboring points in a point cloud. Obviously, this would fail when the densities of the point distributions in the two point clouds are different.

ICP-based methods The Iterative Closest Point (ICP) method, proposed by Besl et al. [12], is a widely used method to align two 3D point clouds. The algorithm iterates two steps: in each iteration, closest points are selected, and a rigid transformation between them is estimated. This estimation is usually called the orthogonal Procrustes problem [59, 60] which has been well studied since the 1960s. Also, a similarity transformation can be estimated [60–62] instead of the rigid transformation. ICP could provide a scale ratio of two point clouds as a result of alignment when similarity transformation is estimated. However, it is well known that ICP requires a good initial pose and also an initial guess of the scale. ICP usually fails if the two point clouds differ in pose and scale.

Descriptor-based methods In contrast to ICP, descriptor-based methods do not require any initial guess of transformation. There are several well-known 3D descriptors such as spin images [13, 14], NARF [15], and Shape context [63].

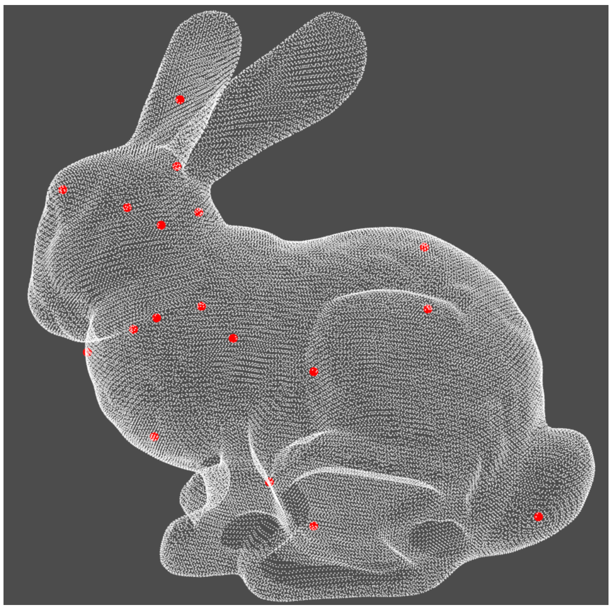
These feature descriptors are computed at keypoints by using associated normal vectors and a local neighborhood size. Those descriptors are then used for finding correspondences between two point clouds in order to register them. One disadvantage of these methods is the requirement of a fixed local neighborhood size to compute the descriptors. It is difficult to set up the appropriate neighborhood size if the scales of the point clouds are different.

For 3D data, there are some scale invariant features. Some extensions of 2D features have been proposed such as 3D SIFT [16] and nD SIFT [17], but they just describe features of 3D volumes or n-dimensional data, not a sparse set of 3D points. Instead of directly extending those 2D features, some researchers have proposed to combine 2D features with 3D point clouds [18–20]. Those are effective when 2D images are available, but not applicable to characterize the scale of 3D point clouds themselves.

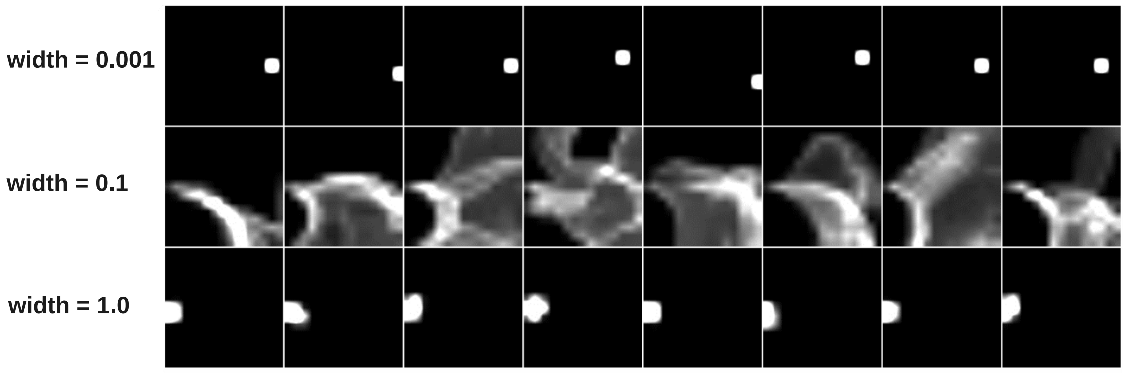
Our contribution The proposed methods described in the following sections estimate the scale ratio of two point clouds by using only local 3D shape characteristics. Since we have separated the alignment step from the scale estimation step, we can use existing alignment methods such as ICP or spin images once the scale ratio has been estimated by our methods.

4.2 Scale Estimation of a Single Point Cloud

In this section, we describe our first method for estimating the scale of a given point cloud. This method finds an appropriate neighborhood size, which we call keyscale. We use spin images [13, 14] in our work, however, we can use any other feature descriptors which are not scale invariant: such as NARF [15] or Shape context [63].



(a)



(b)

FIGURE 4.2: A point cloud (a) and its spin images (b) obtained for different widths $w = 0.001$, 0.1 and 1.0.

The main idea is that spin images require an "appropriate size" of neighborhood. Figure 4.2 illustrates how the size affects the descriptor. Descriptors like spin images are computed at keypoints (red points on the object in Figure 4.2 (a)) with a given neighborhood size. The size here is called a width w of the spin images.

When the width is too small, as shown in the top row of Figure 4.2 (b), spin images do not represent correctly the local geometry because usually any object surface can be seen as flat in extremely small locality. In this case, all spin images would represent just a plane, being the same or very similar to each other as shown in the top row of Figure 4.2 (b).

Similarly, spin images do not represent geometry correctly if the width is too large, as shown in the bottom row of Figure 4.2 (b). When the width is say 10 times larger than the size of the object in consideration, any 3D point would tend to fall in the same bin (or in small number of bins) of a histogram computed for a spin image. Again, spin images may become very similar to each other.

Therefore, we can say that the similarity between spin images has a minimum as the width changes. At the minimum, the spin images are most different from each other as shown in the middle row of Figure 4.2 (b). That is the width of spin images, or the keyscale , that characterizes the scale of a point cloud.

We define the similarity between spin images as the cumulative contribution rate of a PCA of the spin images at a specific dimension of eigenspace. We will explain this idea more formally in the following subsections. The key idea is that if spin images are not approximated well by a small number of eigenvectors, then the spin images are not similar to each other and the cumulative contribution rate decreases.

In order to well understand our method, firstly, we will introduced spin images and PCA in next two sub-sections.

4.2.1 Spin Images

The spin image is a surface representation technique that was initially introduced by Johnson et al. [13, 14] and is used for surface matching and object recognition in 3D scenes. Spin images ensure the global properties of any surface in an object-oriented coordinate system rather than in a viewer-oriented coordinate system. Object-oriented coordinate systems are coordinate systems fixed on a surface or an object while viewer-oriented coordinate systems are based on the viewpoint of the observer of the surface. By using object-oriented coordinate systems, the description of a surface or an object is view-independent and it does not change while the viewpoint changes.

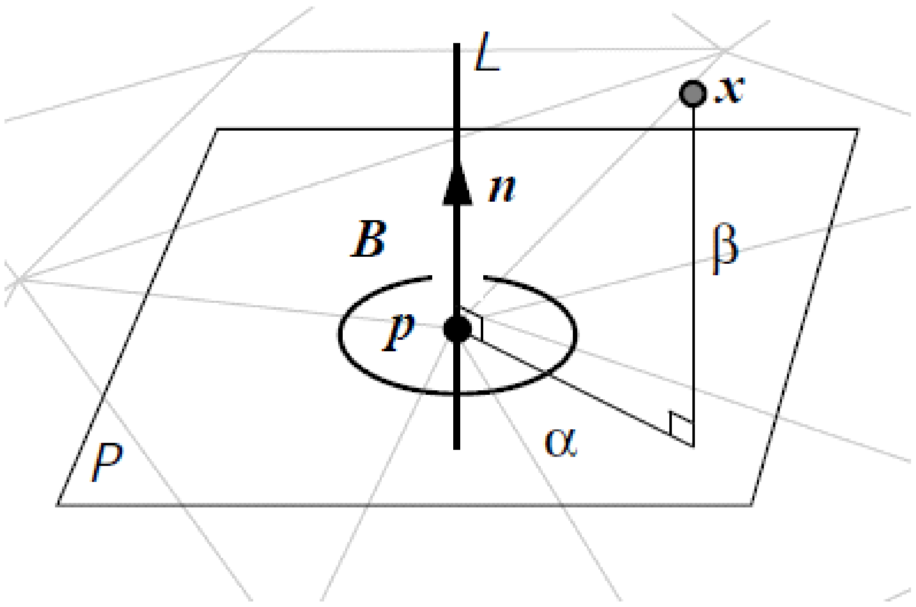


FIGURE 4.3: The cylindrical coordinate system and its($\mathbf{p}, \mathbf{n}$) 2D basis (Taken from [13, 14])

.

The key element of spin images generation is the oriented points. The oriented points (denote as O) are 3D surface points which have an associated direction normal vector to them. The surface which these points correspond to is defined as a polygonal mesh M with vertices. An oriented point O at a surface mesh vertex is defined by the 3D position of the surface vertex (denoted as $\boldsymbol{p}$) and a reference normal vector (denoted as $\boldsymbol{n}$). Then we can formulate a 2D basis $(\boldsymbol{p}, \boldsymbol{n})$ which will correspond to a local coordinate system. To achieve this we use the tangent plane P through $\boldsymbol{p}$ oriented perpendicularly to $\boldsymbol{n}$ and the line L through $\boldsymbol{p}$ parallel to $\boldsymbol{n}$. This will cumulated in a (α, β) cylindrical coordinate system where α is the (non-negative) perpendicular distance to L and β is the signed (positive or negative) perpendicular distance to P. The cylindrical coordinate system is shown in Figure 4.3.

The oriented point basis is then used to generate a spin map, $\boldsymbol{S}_0$. $\boldsymbol{S}_0$ can be expressed as a projection function of the 3D points $\boldsymbol{x}$ of an object to 2D coordinates (α, β) associated with the 2D basis $(\boldsymbol{p}, \boldsymbol{n})$ that corresponds to the oriented point O. The projection function can be seen below:

$$S_0 : \mathbb{R}^3 \to \mathbb{R}^2, \tag{4.1}$$

$$S_0(x) \to (\alpha, \beta) = (\sqrt{\| \boldsymbol{x} - \boldsymbol{p} \|^2 - (\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p}))^2}, \boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p})). \tag{4.2}$$

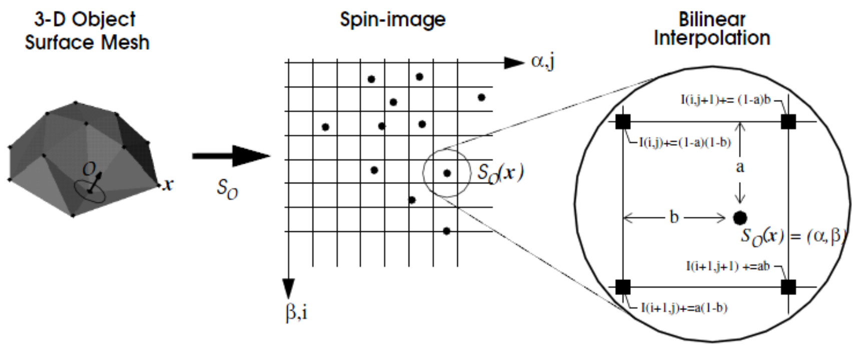


FIGURE 4.4: Creation of the 2D array representation of a spin image (Taken from [13, 14])

.

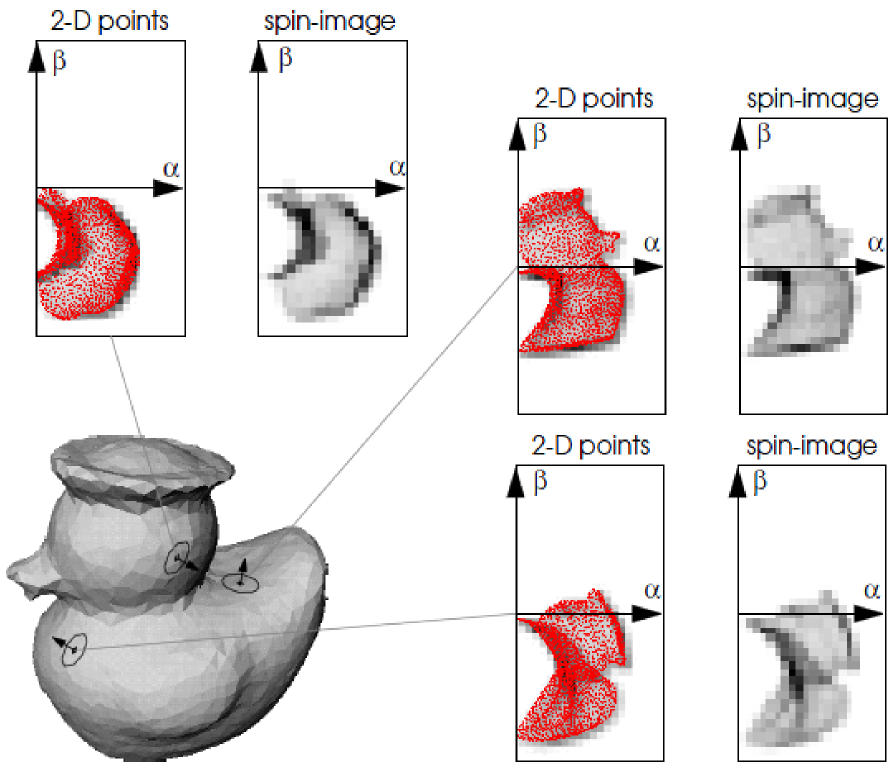


FIGURE 4.5: Spin image creation for three oriented points in the surface of an object (Taken from [13, 14])

.

To better understand the above procedure, it is necessary to recall that the surface of an object is represented as a mesh M with vertices. By applying the projection Equations 4.2 to all the vertices, we create a set of points with (α, β) 2D coordinates. Note that to create a spin image, we only need one oriented point. The other 3D points near the oriented point are used to expressed to cumulative density. The next step is to create a 2D array that is going to represent the spin image generated by using the

2D basis. In order to achieve this (shown in Figure 4.4), we accumulate the 2D points (α, β) into discrete bins. Each time a 2D point is cumulated into a bin, we update the 2D array by incrementing the surrounding bins in the table. At this point we need to consider the noise in the data. So the contribution of the 2D point is spread to the four surrounding bins of the 2D array by using bilinear interpolation. Some examples are shown in Figure 4.5. By using different O points, the generated spin images are very close to the 2D points distribution.

4.2.2 PCA

Principal Component Analysis (PCA) [64] is a variable reduction method and it's very famous in the area of statistics learning and data analysis. Observing data (or samples) on a number of variables is very useful when machine learning is applied, and it's believed that some redundancy existed in those observed data. In this case, redundancy means that some of the variables are correlated with one another, possibly because they are measured in the same construct. Since this kind of redundancy existed, it's reasonable to believe that there is a way to reform the observed data and remove the redundancy by using a smaller number of artificial variables (principal components) that will account for the great majority of the variance in the observed variables. Or in another way, there maybe a method to find a transformation matrix which can represent the observed data in a better way.

Consider a case with a set of vectors $\mathcal{X}$. Then we have a vector $\boldsymbol{x}$ and calculating the average of the covariance matrices of all samples.

$$\boldsymbol{X} = \frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{X}} (\boldsymbol{x} - \boldsymbol{m}) (\boldsymbol{x} - \boldsymbol{m})^T \tag{4.3}$$

where n is the number of the samples.

An optimal transformation matrix $\boldsymbol{P}$ is consisted of the generalize eigenvectors which correspond to the largest d' eigenvalues of $\boldsymbol{X}$. d' is decided by the contribution rate of the cumulated eigenvalues.

PCA can be done by using the singular value decomposition (SVD) of the data matrix.

Suppose there is a $n \times p$ data matrix $\boldsymbol{X}$, n and p is the number of observations and the predictors, respectively. $\boldsymbol{X}$ is centered that the column means are 0. Then the SVD of $\boldsymbol{X}$ is

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^T \tag{4.4}$$

$\boldsymbol{U}$ is a $n \times n$ matrix, its columns are the eigenvector of $\boldsymbol{XX^T}$. $\boldsymbol{\Sigma}$ is a $n \times p$ diagonal matrix with the eigenvalues of $\boldsymbol{XX^T}$ on its diagonal line. $\boldsymbol{V}$ is $p \times p$ matrix, its columns are the eigenvector of $\boldsymbol{X^TX}$. The optimal transformation matrix $\boldsymbol{P}$ is consists of the first q ($q \ll p$) columns in $\boldsymbol{U}$. Now combine $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ together to get a new coefficient matrix $\boldsymbol{C}$=$\boldsymbol{\Sigma V^T}$, then $\boldsymbol{X}$ can be represented by $\boldsymbol{U}$ and $\boldsymbol{C}$:

$$\boldsymbol{X} = \boldsymbol{UC} \tag{4.5}$$

Then column $\boldsymbol{x_i}$ in $\boldsymbol{X}$ can be represented as

$$\boldsymbol{x_i} = \sum_{j=1}^{n} \boldsymbol{u_j} c_{ij} \tag{4.6}$$

Since $\boldsymbol{P}$ is made of first q columns in $\boldsymbol{U}$ and the left columns in $\boldsymbol{U}$ has very small eigenvalues, $\boldsymbol{x_i}$ can be approximated as:

$$\boldsymbol{x_i} \approx \sum_{j=1}^{q} \boldsymbol{p_j} c_{ij} \tag{4.7}$$

Turn Equation 4.7 into matrix form

$$\begin{aligned} \boldsymbol{X} &\approx \boldsymbol{PC} \qquad (4.8) \\ &= \boldsymbol{DC} \qquad (4.9) \end{aligned}$$

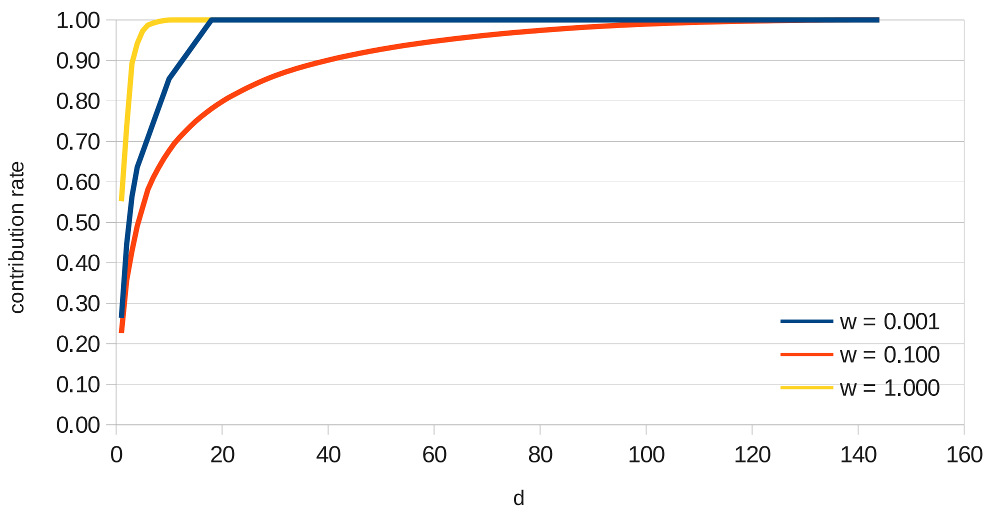
Here $\boldsymbol{C}$ is reshaped to a $q \times p$ matrix by discarding the left part to make $\boldsymbol{P}$ and $\boldsymbol{C}$ multipliable. $\boldsymbol{D}$ is dictionary and just the same with $\boldsymbol{P}$.

Now basing on Equation 4.9, PCA can be treated as using a linear combination of eigenvectors (principal components, short as PCs) to represent the data. It's the same interpretation topic appears in linear regression which the response is a linear combination of the predictors.

Till here, we have introduced the concepts of spin images and PCA. The contribution rate and the keyscale will be described in next two sub-sections.

4.2.3 Contribution Rate

Next, we describe the cumulative contribution rate. We denote spin images $Spin_i(\alpha, \beta, w)$ of $m \times m$ bins as vectors $\boldsymbol{s}_i^w$ of m^2 dimensions. By performing PCA for a set of spin images $\{\boldsymbol{s}_i^w\}$, we have m^2 eigenvectors $e_1^w, \ldots, e_{m^2}^w$ of m^2 dimensions and corresponding real eigenvalues $\lambda_1^w \geq \cdots \geq \lambda_{m^2}^w \geq 0$.



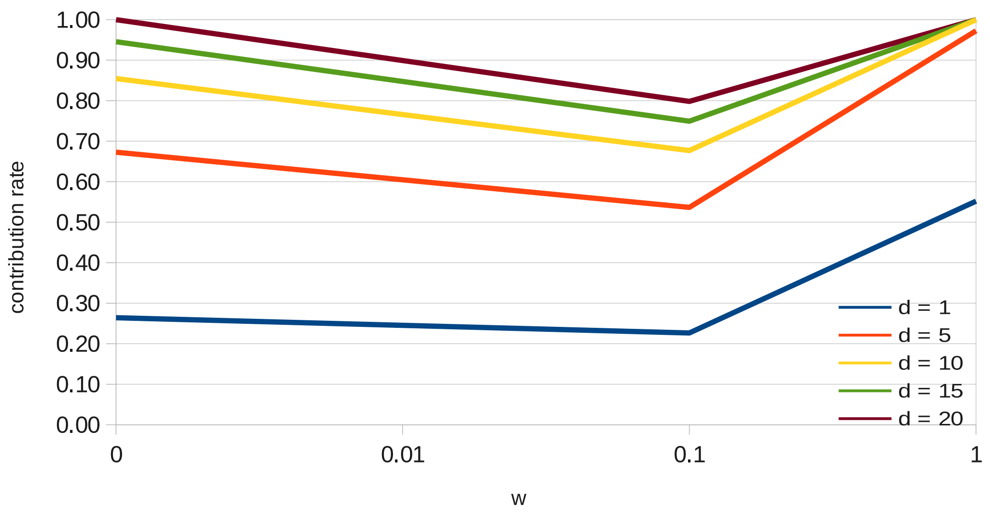


FIGURE 4.6: Examples of contribution rate curves over different dimensions d (top) for fixed widths w, and (bottom) different width w for fixed dimensions d.

The cumulative contribution rate is defined as

$$c_d^w = \frac{\sum_{i=1}^{d} \lambda_i^w}{\sum_{i=1}^{m^2} \lambda_i^w}, \tag{4.10}$$

where $d = 1, 2, \ldots, m^2$. Figure 4.6 (top) gives some typical curves showing how the cumulative contribution rate c_d^w increases as the dimension d increases. As we explained above, when the width w is too small or large compared to the object, the spin images become similar to each other. In Figure 4.6 (top), we can see that the curves approach quickly 1 when $w = 0.001$ (too small) and $w = 1.0$ (too large): fewer dimensions are enough to approximate the spin images if they are similar to each other. On the other hand, the curve of $w = 0.1$ increases more slowly, which means that the spin images are quite different from each other.

4.2.4 Finding the Keyscale

As described above, the idea of a keyscale is to find the appropriate spin image width w in terms of the minimum of cumulative contribution rate. We can see this minimum in Figure 4.6 (top), for example, at dimension $d = 10$. The values of cumulative contribution rate are large for both w =0.001 and 1.0, however the value for $w = 0.1$ is much smaller than those values.

This can be seen more clearly in Figure 4.6 (bottom) which is a plot of the cumulative contribution rate values over different w. In this plot, the curve for $d = 10$ has the minimum at $w = 0.1$, which is the keyscale of this sample point cloud.

4.2.5 Limitations

Our first method, explained above, can find a particular scale of point clouds as a keyscale by determining w which gives the minimum of the curves in Figure 4.6 (bottom). This method however has the following three problems. First, we need to choose the dimension d to decide which curve is used. In the description above we chose $d = 10$ as an example, but there is no specific reason for this. Fortunately, every curve in Figure 4.6 (bottom) has the same minimum, but this might not be the case always. Second, the minimum is not stable: it may change against small amounts of noise, and there might be more than one local minima. In the latter case, it is difficult to determine which local minima is more appropriate as a keyscale. If there is a heavy clutter in the 3D scene, the situation can become even worse. Third, the minimum is found at discrete steps of the width, which is neither accurate nor efficient. There is a trade-off between these problems because we may find a stable minimum with a larger and sparser discrete step size, but the minimum is less accurate. A more accurate estimate may be obtained with a dense step size, while the computational cost is more expensive and the scale ratio result is less stable.

4.3 Scale Ratio Estimation of Two Point Clouds

In this section, we introduce our second method that estimates the ratio of the scales of two point clouds instead of estimating the scales of each point cloud separately. This method is an extension of the first one: here we use all curves of all dimensions, while in the first approach we used only a single curve of a specific dimension to find a minimum.

We will explain below how this approach works and how it solves the problems that the first approach had.

4.3.1 Formulation

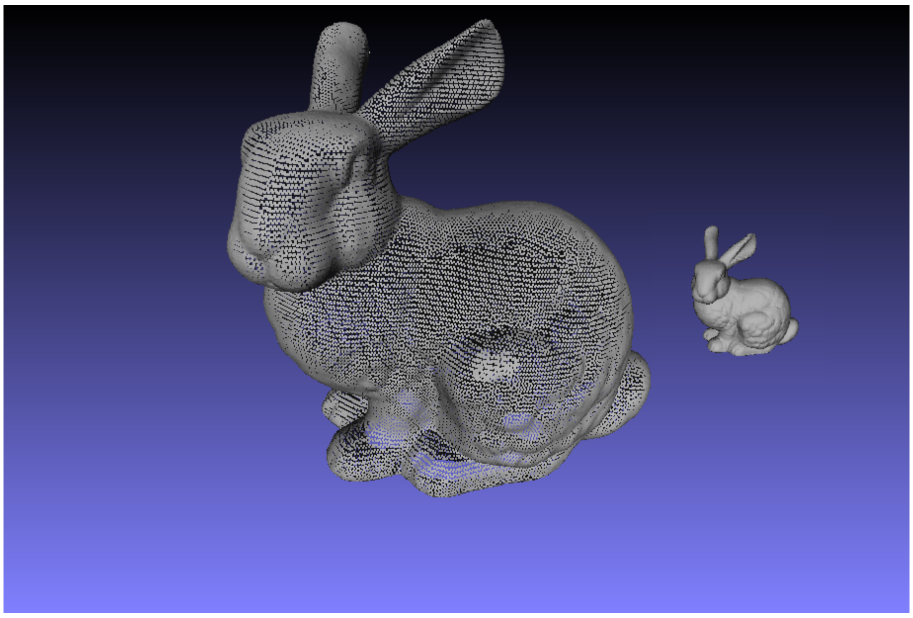
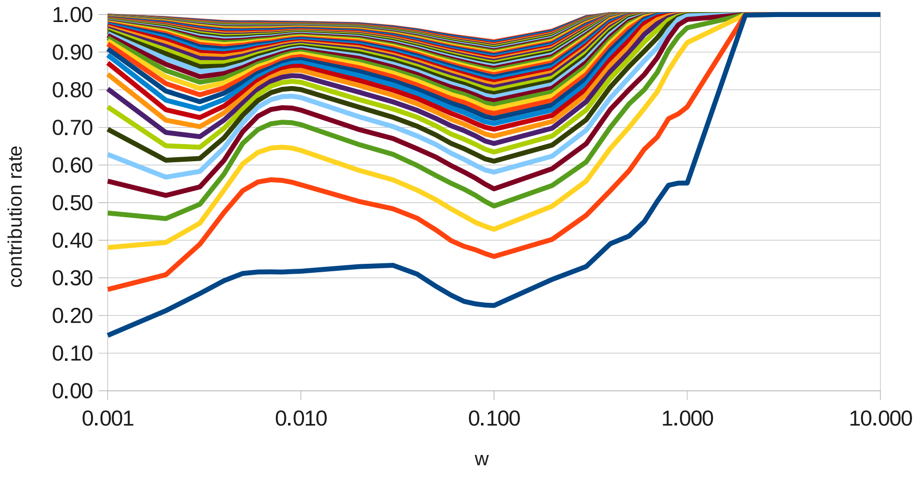


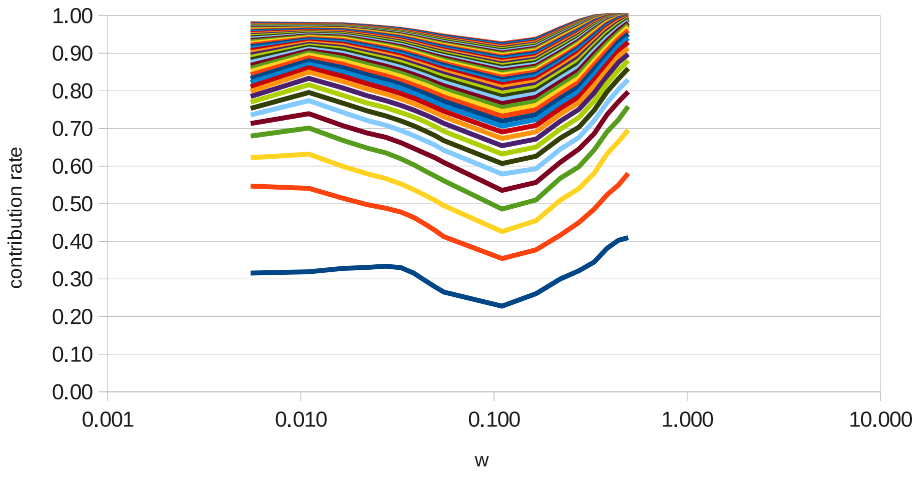
Figure 4.7: Point clouds from the bunny simulation.

The idea is that we can use all curves of all dimensions to find the scale ratio robustly. An example of two sets of curves is shown in Figure 4.8 (a) and Figure 4.9 (a). The point cloud is shown in Figure 4.7. We assume that the two sets of curves differ only in scale — this is reasonable as the point clouds corresponding to these two sets of curves should share a similar overlapping part of a 3D scene.

Of course there are some parts where the sets of curves are not similar (for example, the left parts of Figure 4.8 (a) and Figure 4.9 (a) differ from each other). If we can remove those parts from the sets of curves, we can “register” the two sets of curves.



(a)

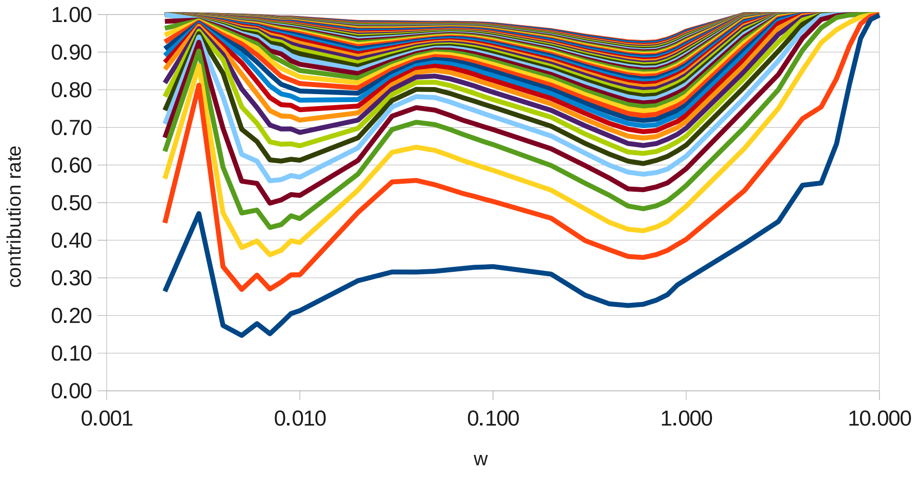


(b)

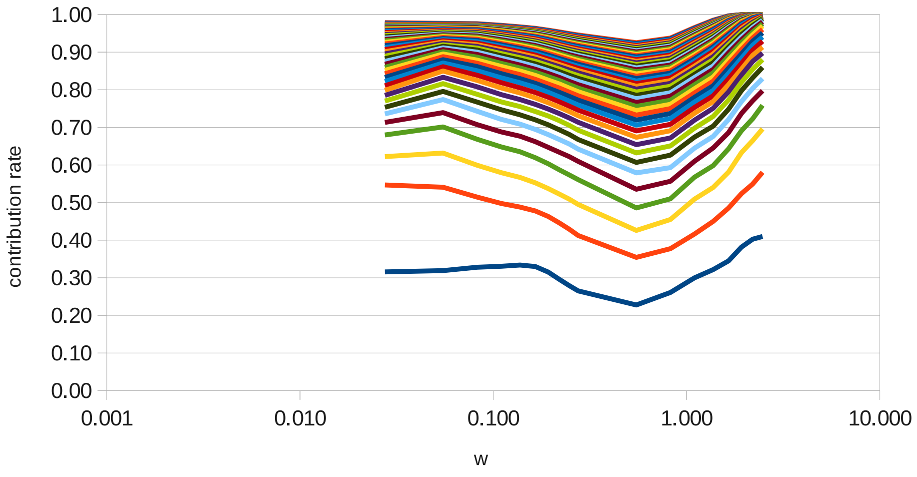
FIGURE 4.8: c_d^w curves from the original bunny simulation. (a) Original scale and (b) the curves shown only in the initial search range found by the initialization step.

In other words, we estimate a scale difference (i.e., a *scale ratio*) between two sets of curves by registering those curves. This idea simplifies the problem: before we had to register two 3D point clouds for finding scales, but now we have a 1D registration problem along the scale (w) axis in Figure 4.8 and Figure 4.9. We call this approach *scale ratio ICP* and describe it in more detail below.

Suppose there are two sets of curves $c1_d^w = \{x_{w_i}^d, w_i\}$ and $c2_d^w = \{y_{w_i'}^d, w_i'\}$ where $1 \leq d \leq m^2$. Notice that here we represent the curves as sets of 2D points. The horizontal axis



(a)



(b)

FIGURE 4.9: c_d^w curves from the 5 times larger bunny simulation. (a) 5 times larger scale. (b) the curves above shown only in the initial search range found by the initialization step.

is the width w of the spin images, and the vertical axis is the cumulative contribution rate.

The objective function we want to minimize is

$$E(t) = \sum_d \sum_i \left\| \begin{pmatrix} y_{w_i'}^d \\ w_i' \end{pmatrix} - \begin{pmatrix} x_{w_i}^d \\ t w_i \end{pmatrix} \right\|^2, \tag{4.11}$$

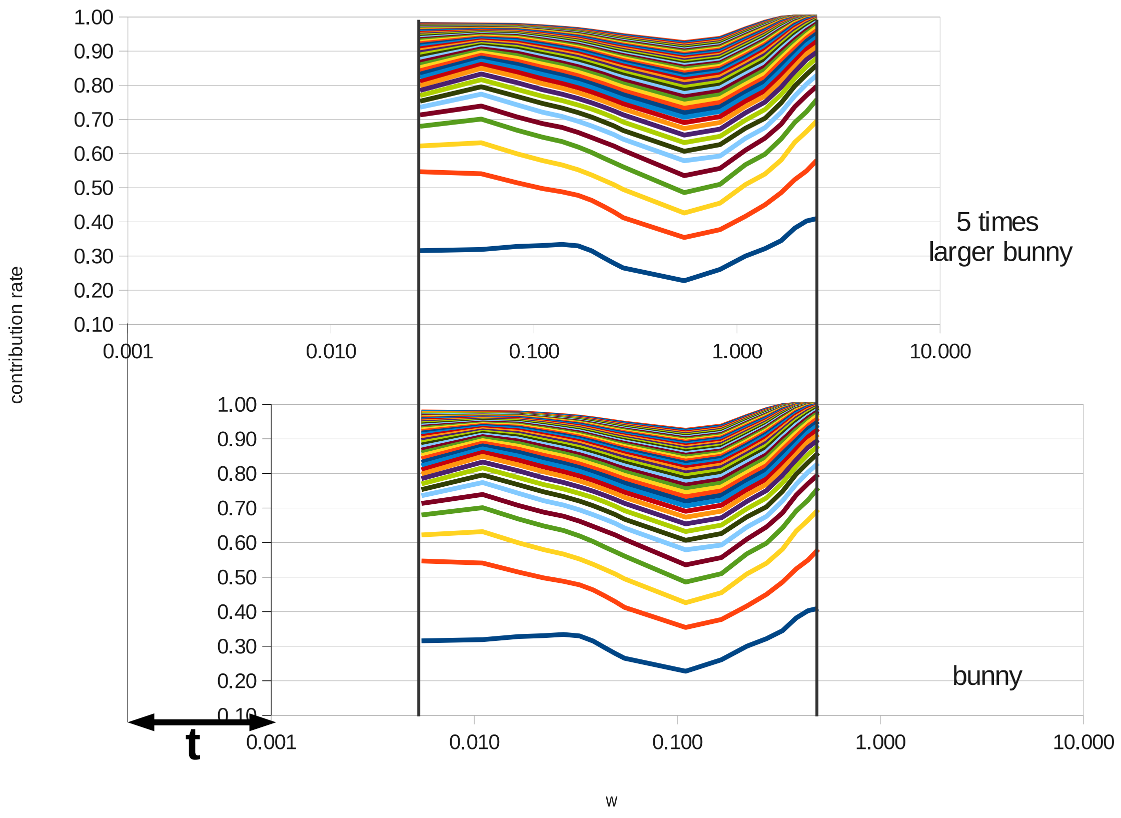


FIGURE 4.10: Estimated scale ratio t by curve registration using the overlapping parts of the curves shown in Figure 4.8 (b) and Figure 4.9 (b).

where t is the unknown scale ratio to be estimated. The solution is obtained in a closed-form. The objective function can be written as:

$$E(t) = \sum_d \sum_i \left(\left(y^d_{w'_i} - x^d_{w_i} \right)^2 + \left(w'_i - t w_i \right)^2 \right).$$

By taking the derivative of $E(t)$ with respect to t, we have:

$$\frac{\partial E(t)}{\partial t} = -2 \sum_d \sum_i w_i \left(w'_i - t w_i \right),$$

and by setting it to 0:

$$\sum_i \left(-w_i w'_i + t w_i^2 \right) = 0,$$

then we have the solution,

$$t = \frac{\sum_i w_i w'_i}{\sum_i w_i^2}. \tag{4.12}$$

So far we have assumed that the points on each curve have corresponding points on the other curve, however, in fact we do not know the correspondences in advance. Therefore, we use the strategy of the standard ICP: finding the closest points iteratively.

4.3.2 Scale Ratio ICP Algorithm

In this section, we will introduce the proposed scale ratio ICP algorithm. Because scale ratio ICP is a variation of standard ICP. So, we will describe the standard ICP first.

4.3.2.1 ICP

The Iterative Closed Points (ICP) [12] is a widely used method to align two 3D point clouds. It is an iterative algorithm that works in two phases: in each iteration, closest points are selected, and a rigid transformation between them is estimated. In the alignment procedure, the corresponding points are not needed. Suppose we have two 3D point clouds X and Y to do alignment. The details are shown in Algorithm 3.

Algorithm 3 ICP algorithm.

```
1. temporary corresponding: find the closest points for $X$ from $Y$.
2. parameter: estimate transformation parameter $R$ and $\mathbf{t}$.
3. temporary matching: find the closest points for $RX + \mathbf{t}$ from $Y$.
4. iterate steps 2 and 3 until the estimate converges.
```

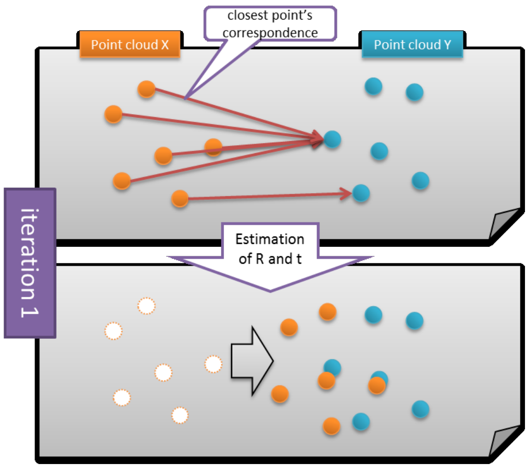
Figure 4.11 and Figure 4.12 show an example for a 4 iterations ICP alignment procedure. The ways to estimate R and $\mathbf{t}$ are introduced in section 2.2.3.2. The details of scale ratio ICP outlined above are given in next sub-section.

4.3.2.2 Scale Ratio ICP

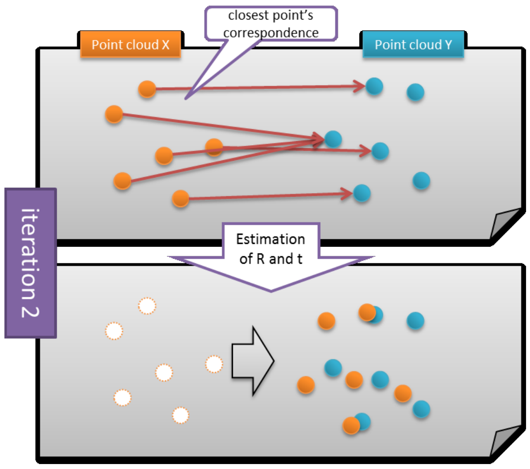
The details of the scale ratio ICP algorithm are as follow:

1. Initialization

   An exhaustive search is used to find an initial rough estimate of t. First, mesh-resolutions [13, 14] w_{mesh1} and w_{mesh2} of the two sets of point clouds are used to

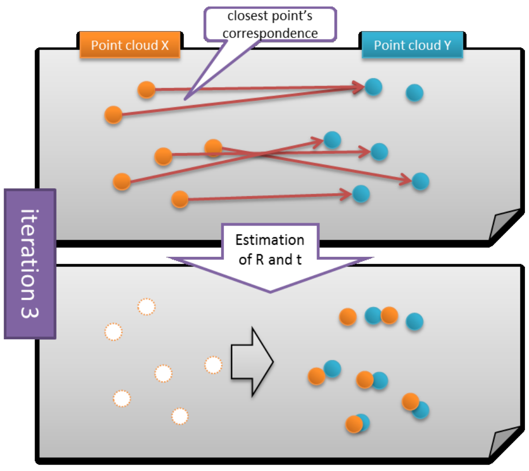


(a)

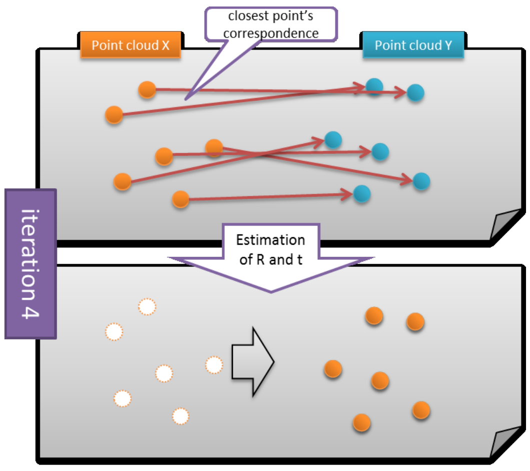


(b)

FIGURE 4.11: An example of ICP which contain 4 iterations, (a) Iteration 1 and (b) iteration 2.



(a)



(b)

FIGURE 4.12: An example of ICP which contain 4 iterations, (a) Iteration 3 and (b) iteration 4.

set the minimums $t_{m1} = 5w_{mesh1}, t_{m2} = 5w_{mesh2}$ and maximums $100t_{m1}, 100t_{m2}$ of spin image width w to find the overlapping part as shown in Figure 4.8 (b) and Figure 4.9 (b). The overlapping parts are $c1_d^w = \{x_{w_i}^d, w_i\}$ where $t_{m1} \leq w_i \leq 100t_{m1}$ and $c2_d^w = \{y_{w_i'}^d, w_i'\}$ where $t_{m2} \leq w_i' \leq 100t_{m2}$. Then we find the minimum in the range at discrete steps as the initial estimate t_{init}:

$$t_{init} = \underset{0<t\leq 100\frac{t_{m2}}{t_{m1}}}{\operatorname{argmin}} \; E(t). \tag{4.13}$$

Then $t \leftarrow t_{init}$.

2. Find putative correspondences

   For each point in the set corresponding to curve $c1_d^w$, find the closest point on the curve $c2_d^w$ with the current estimate of t. Note that this process is performed for different d independently.

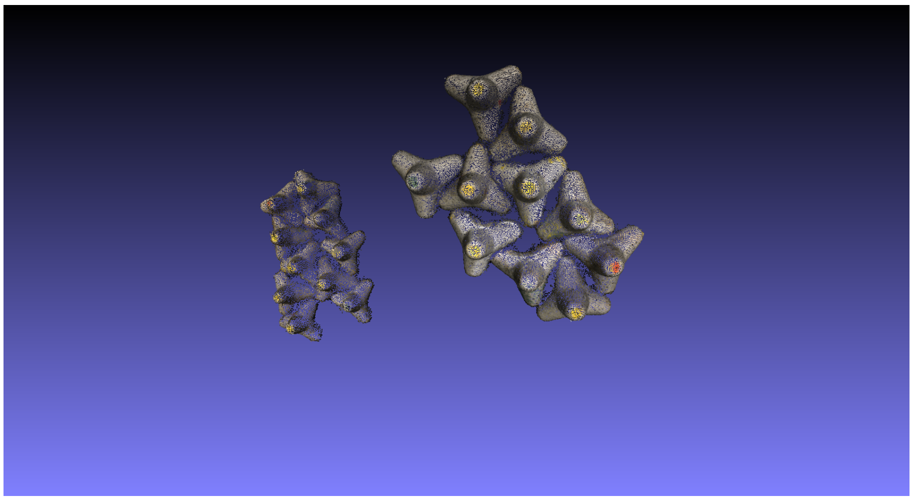
3. Estimate t

   Estimate t based on the current correspondences.

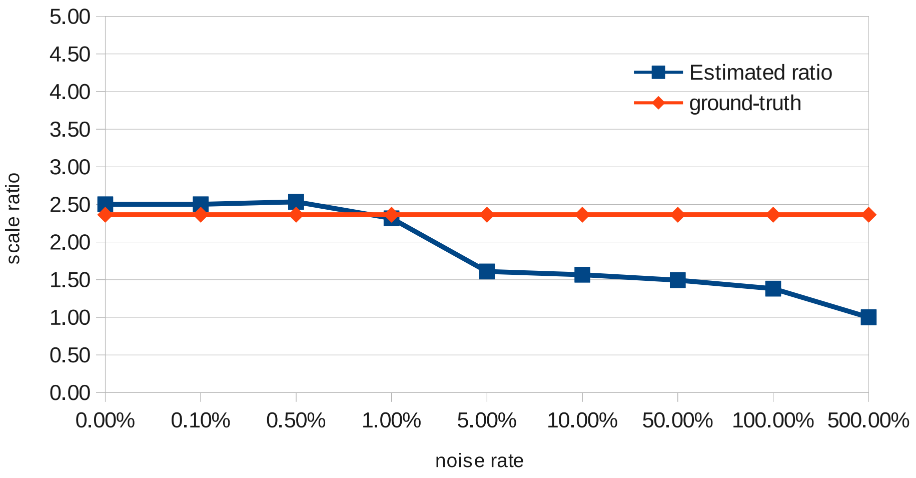
4. Iterate

   Iterate steps 2 and 3 to update t until the estimate converges.

4.4 Evaluation



(a)



(b)

FIGURE 4.13: Experimental results for the small blocks dataset. (a) Original point clouds. (b) Estimated scale ratios over different amount of noise added to the point clouds.

We demonstrate that the proposed method works effectively on 3D point clouds both in simulations, and in artificial and real data experiments.

4.4.1 Simulation

For demonstrating the scale ratio approach, we generated two synthetic 3D point clouds from the Stanford bunny [57]. One point cloud with 69,451 points was created from the

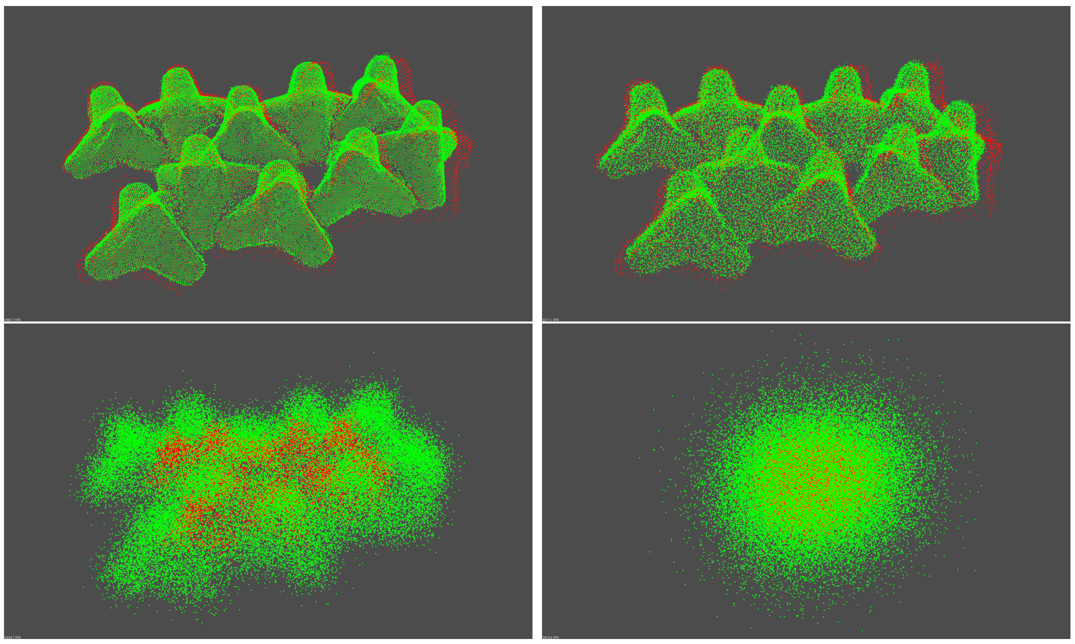


FIGURE 4.14: Registration results (small blocks without change) with the estimated scale ratio for different noise levels (0%, 0.5%, 5%, 50%).

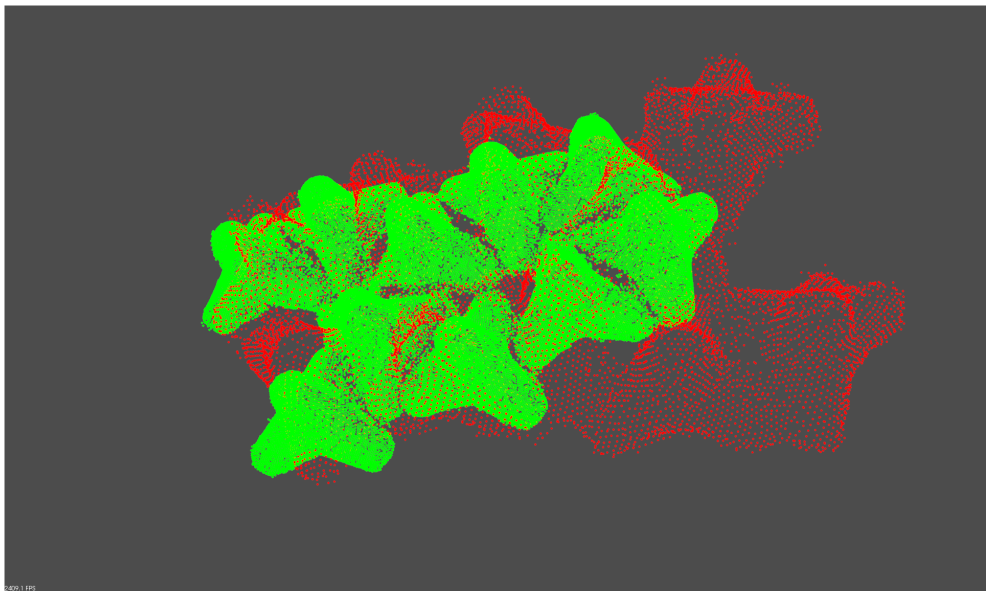


FIGURE 4.15: The standard ICP fails to align the point clouds (small blocks without change), while the scale ratio is correctly estimated.

bunny. Then, the point cloud was scaled by a factor of 5 to create the other point cloud (both are shown in Figure 4.7). Two sets of cumulative contribution rate curves are shown in Figure 4.8 (a), the original scale; and Figure 4.9 (a), 5 times larger scale. To plot these curves, we did not use the initialization described in section 4.3. Therefore, the two sets of curves are difficult to register without appropriately finding the overlapping parts. Figure 4.8 (b) and Figure 4.9 (b) show the curves only in the range determined

Table 4.1: Performance evaluation of several different methods on four different datasets. Numbers in parenthesis are relative errors from the ground truth.

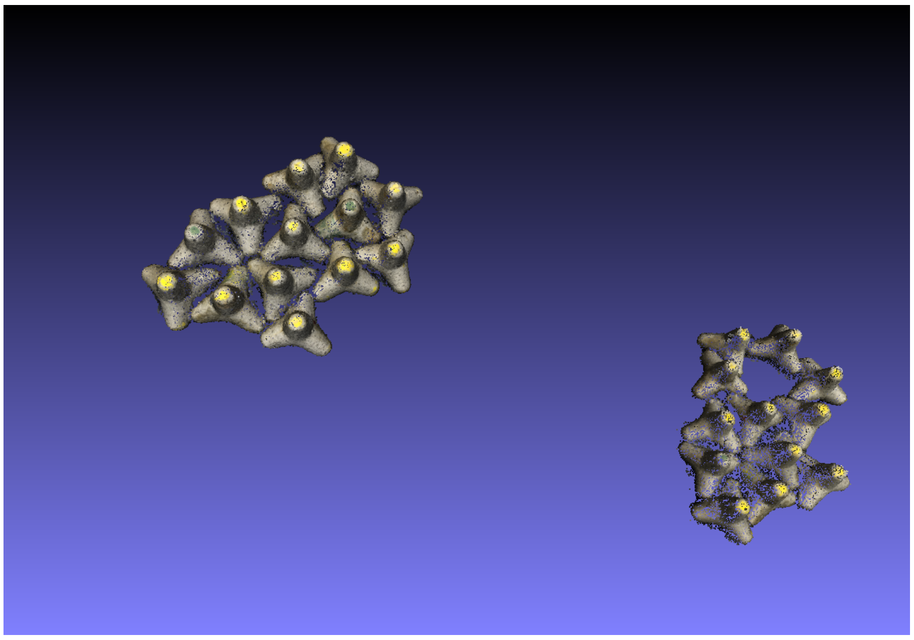
dataset	bunny	small blocks (no change)	small blocks (with change)	real blocks
GROUND TRUTH	5.000	2.364	2.424	1.696
STANDARD DEVIATION	5.000	4.855 (105.37%)	3.833 (58.13%)	1.593 (6.07%)
MESH RESOLUTION [13, 14]	5.000	1.162 (50.85%)	1.684 (30.53%)	1.607 (5.25%)
KEYSCALE [58]	5.000	1.400 (40.78%)	2.250 (7.18%)	1.500 (11.56%)
STANDARD ICP [12]	5.000	3.029 (28.13%)	2.561 (5.65%)	**1.767 (4.19%)**
SCALE RATIO ICP	5.000	**2.502 (5.84%)**	**2.543 (4.91%)**	1.607 (5.25%)

by the initialization step for finding the initial estimate of t_{init}. The ratio found by the proposed scale ratio ICP is exactly the same as the ground truth. Our method worked well for the simple dataset, but so did all other methods compared in the first column of Table 4.1.

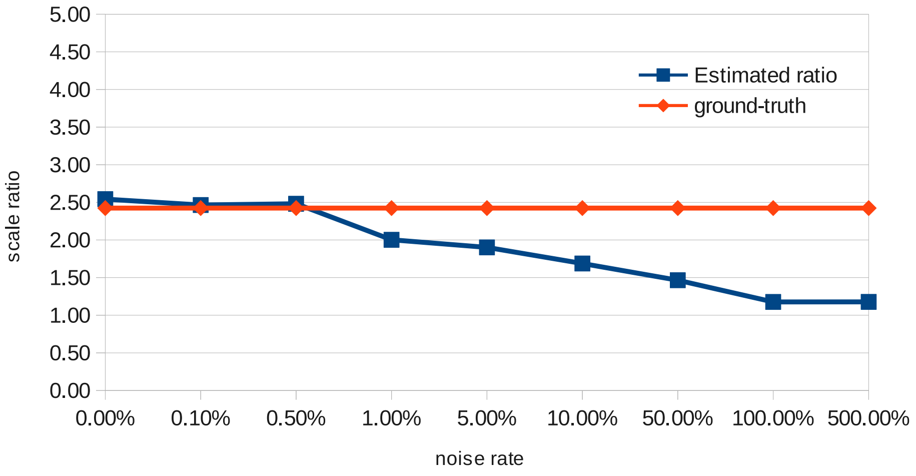
4.4.2 Small Blocks

Now we show experimental results on datasets of miniature blocks, each of which is about 5 cm in height. We used the standard ICP [12] to obtain the ground truth for the scale ratio by carefully providing the initial poses of the point clouds by hand. Then, in order to demonstrate the robustness of the proposed method, we created noisy versions of these datasets by adding uniform noise to each x, y, and z coordinate of every point in the small blocks. The magnitudes of the noise were set in the range between 0.1% and 500% of the maximum of the three sides of the bounding box of the point clouds.

The first dataset consists of a number of small blocks, shown in Figure 4.13 (a). Two point clouds of 73,224 and 101,859 points, together with their normal vectors were computed by 3D reconstruction from 26 and 29 images respectively, using the Bundler [28] followed by Patch-based Multi-view Stereo (PMVS2) [26, 27]. The ground truth of the scale ratio in Figure 4.13 (b) is 2.364. The proposed method provided acceptable results when the noise magnitude is smaller than 1.0%. Some registration results are



(a)



(b)

Figure 4.16: Experimental results for the small blocks dataset with change. (a) Original point clouds. (b) Estimated scale ratios over different amounts of noise added to the point clouds.

shown in Figure 4.14. Results for other methods including standard deviation, mesh-resolution, keyscale and the standard ICP for noise free small blocks are shown in the second column of Table 4.1. Our method achieved the lowest error and best performance.

The second dataset consists of the same small blocks, but one block was removed from the scene while the others are left unchanged, as shown in Figure 4.16 (a). The ground truth for the scale ratio is 2.424 and is shown in Figure 4.16 (b). The accuracy of

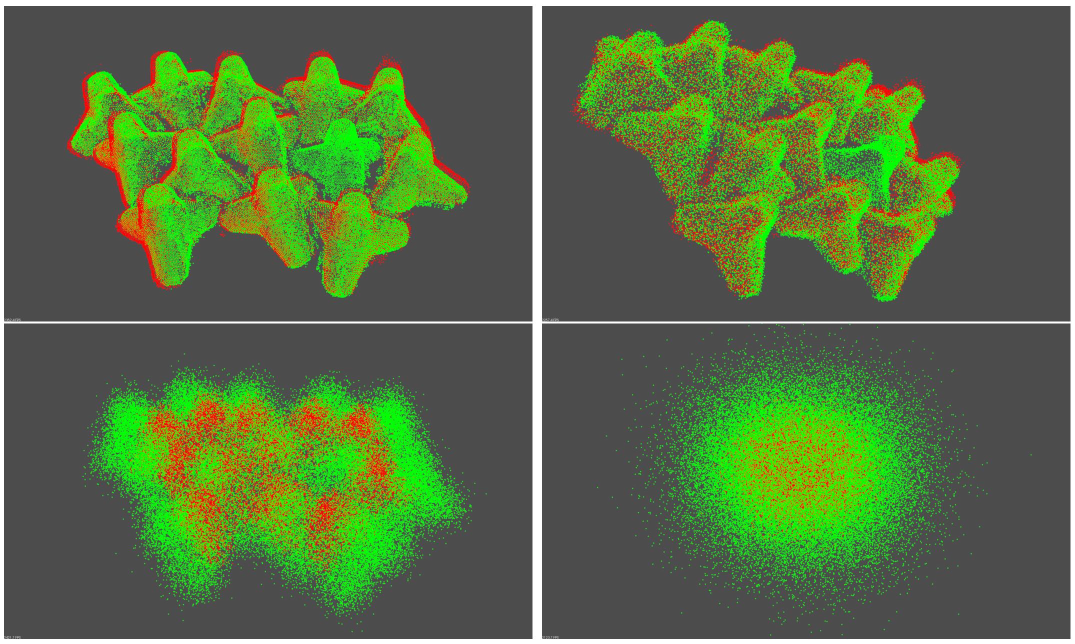


FIGURE 4.17: Registration results (small blocks with change) with estimated scale ratio for different noise levels (0%, 0.5%, 5%, and 50%).

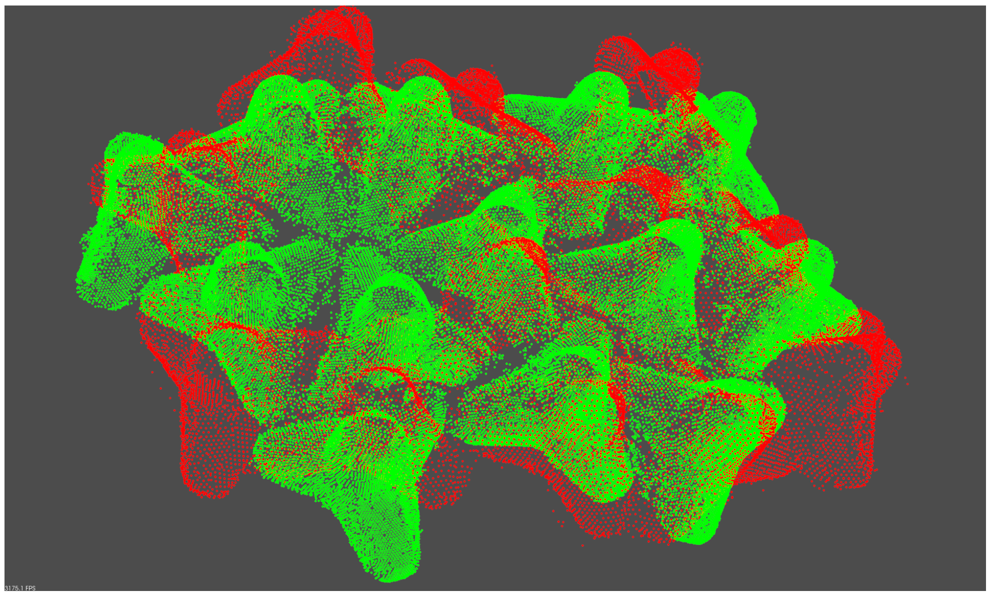
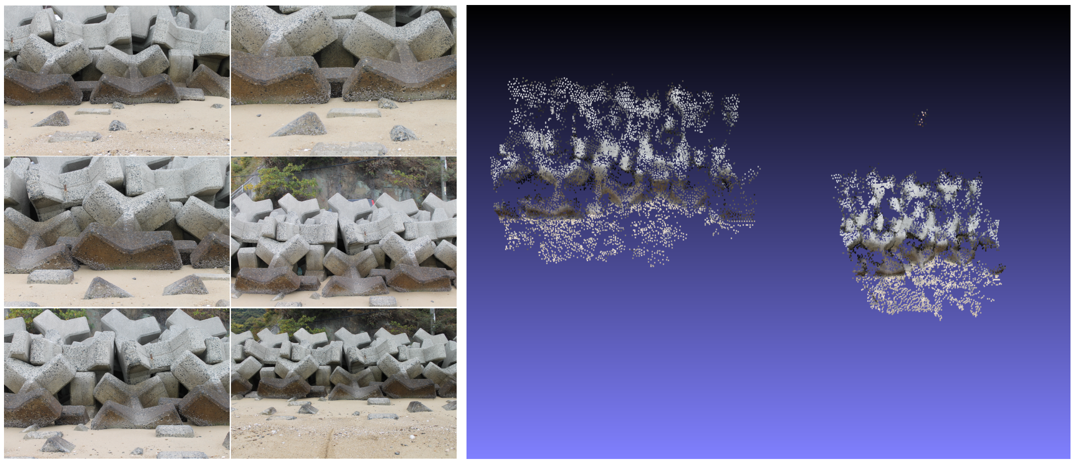


FIGURE 4.18: The standard ICP fails to align the point clouds (small blocks with change), while the scale ratio is correctly estimated.

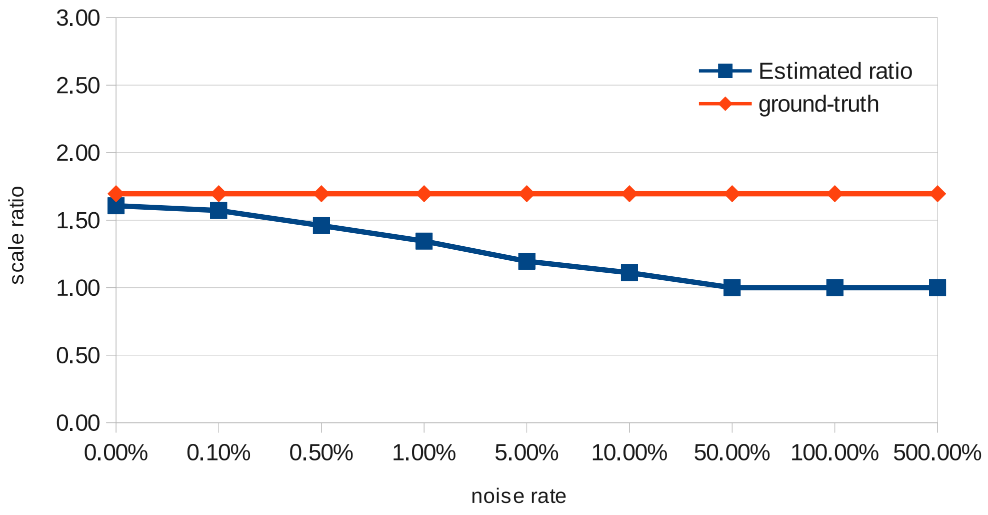
the proposed method gradually decreases as the noise magnitude grows. However, our method achieved again the best performance as shown in the third column of Table 4.1. Figure 4.17 shows some 3D registration results with scale ratio estimated by scale ratio ICP. The alignment process can be seen in our video on YouTube [1].

[1]http://www.youtube.com/watch?v=ZNIkZ5Dd0EU

4.4.3 Outdoor Real Blocks



(a)



(b)

FIGURE 4.19: Experimental results with a real block dataset. (a) Original point clouds and some of the images used for the 3D reconstruction. (b) Estimated scale ratios over different amount of noise added to the point clouds.

Here we show experimental results on two real datasets to demonstrate the robustness of the proposed method. The first dataset consists of two point clouds of 10,325 and 9,343 points computed from 24 images of real wave dissipating blocks of 5 m in height at a coast. Some of the images used for reconstruction and the point clouds are shown in Figure 4.19 (a). As shown in Figure 4.19 (b), the estimated scale ratio decreases gradually from the ground truth (1.607) as the noise increases. Some registration results are shown in Figure 4.20. Results for this dataset are shown in the forth column of

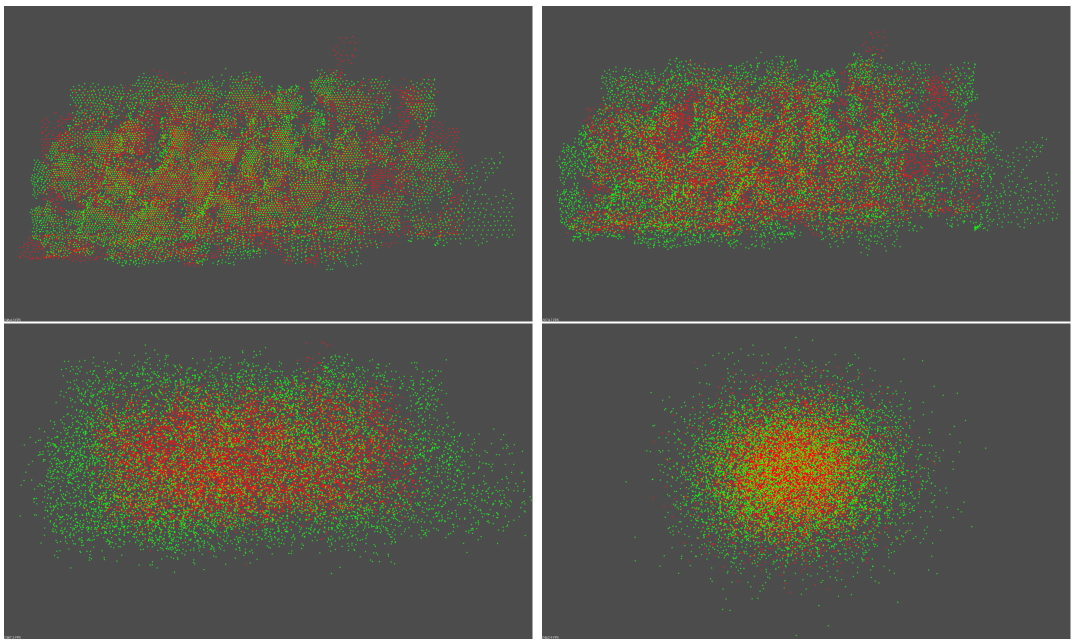


FIGURE 4.20: Registration results (real blocks) with estimated scale ratio for different noise amounts (0%, 0.5%, 5%, and 50%).

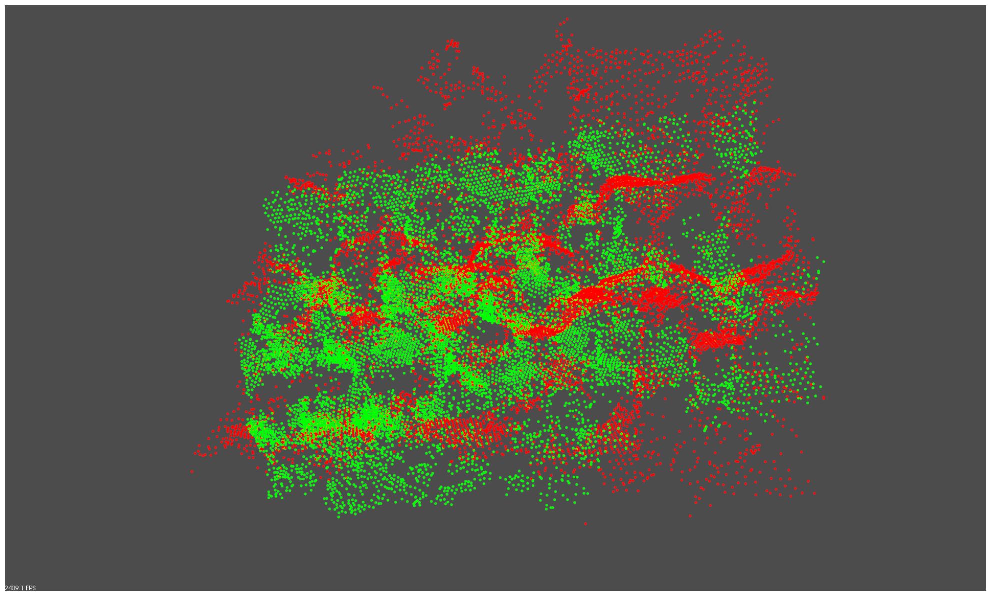
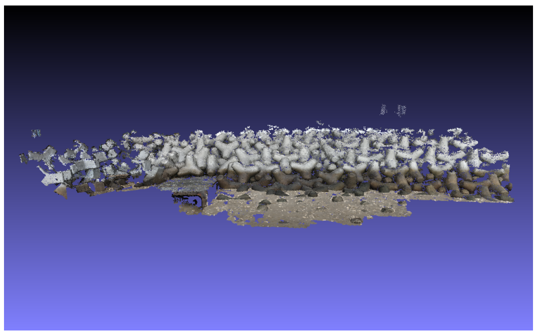


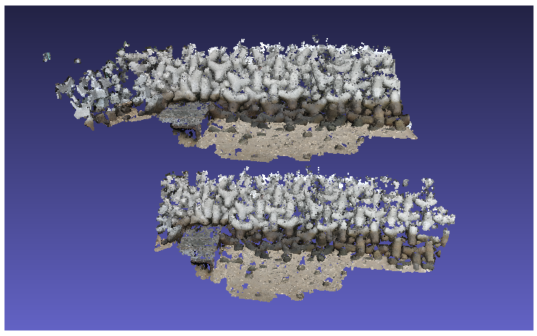
FIGURE 4.21: The standard ICP fails to align point clouds (real blocks) while the scale ratio is correctly estimated.

Table 4.1. Although the result for ICP was the best one, the alignment is completely wrong, as shown in Figure 4.21.

The second dataset is used to evaluate the effect of changing the degree of overlap between the two point clouds. If the two point clouds overlap completely, the scale ratio can be estimated accurately. However usually this is not the case. Therefore in this experiment we evaluate our method in a more realistic scenario, where the two point

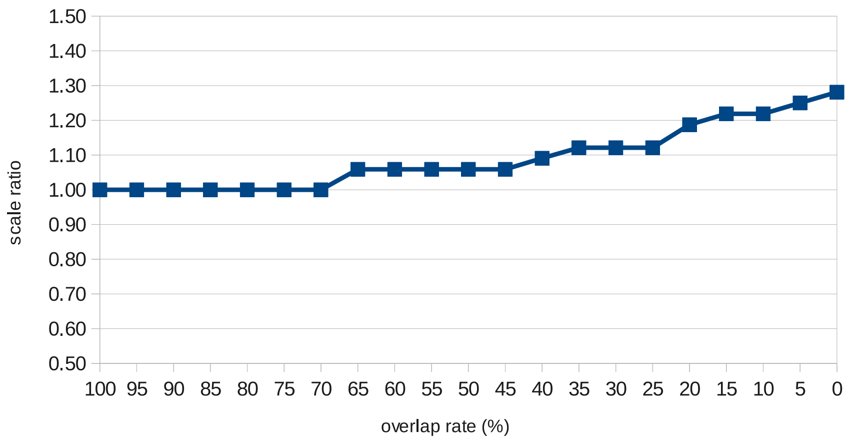


(a)

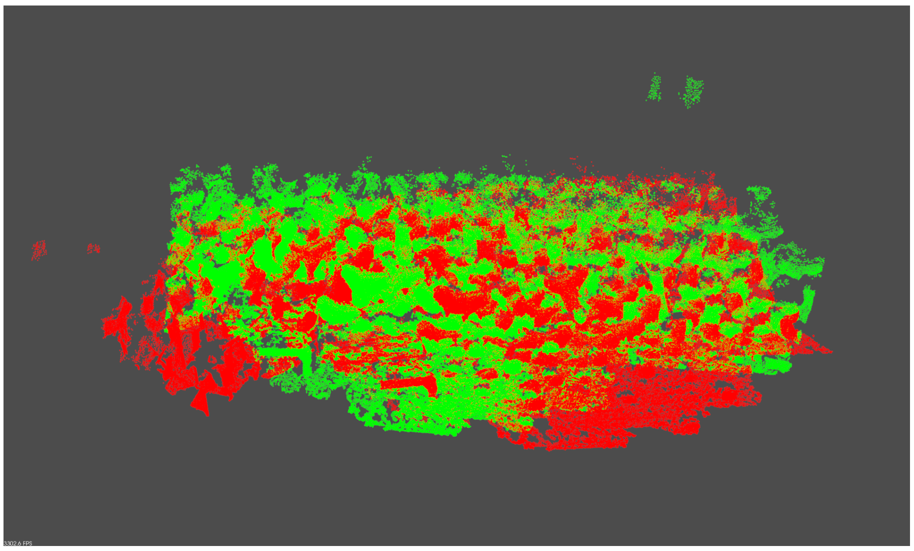


(b)

FIGURE 4.22: Evaluation for different levels of overlap. (a) The original point cloud, and (b) the partially cut-out point clouds with an overlap rate of 70%.

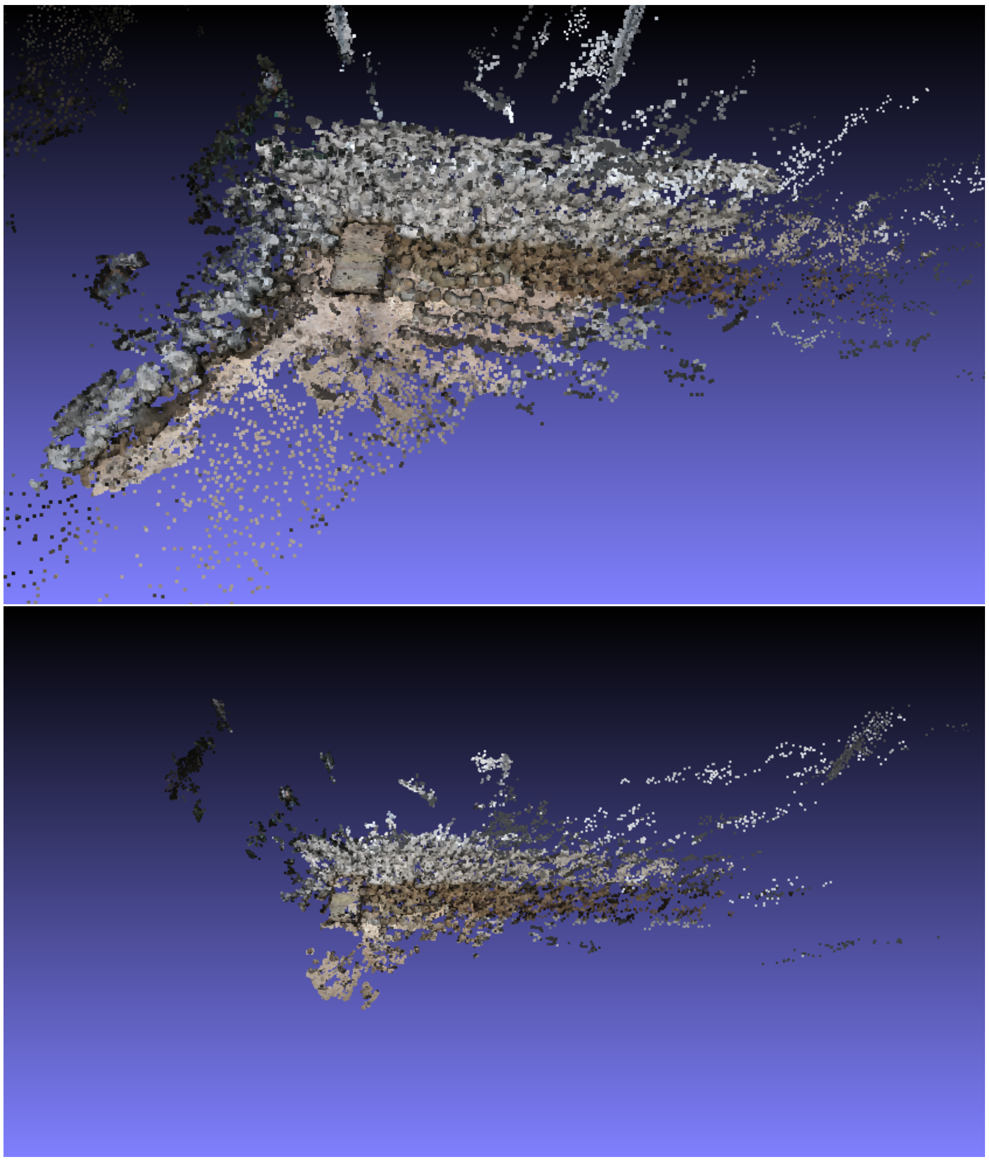


(a)

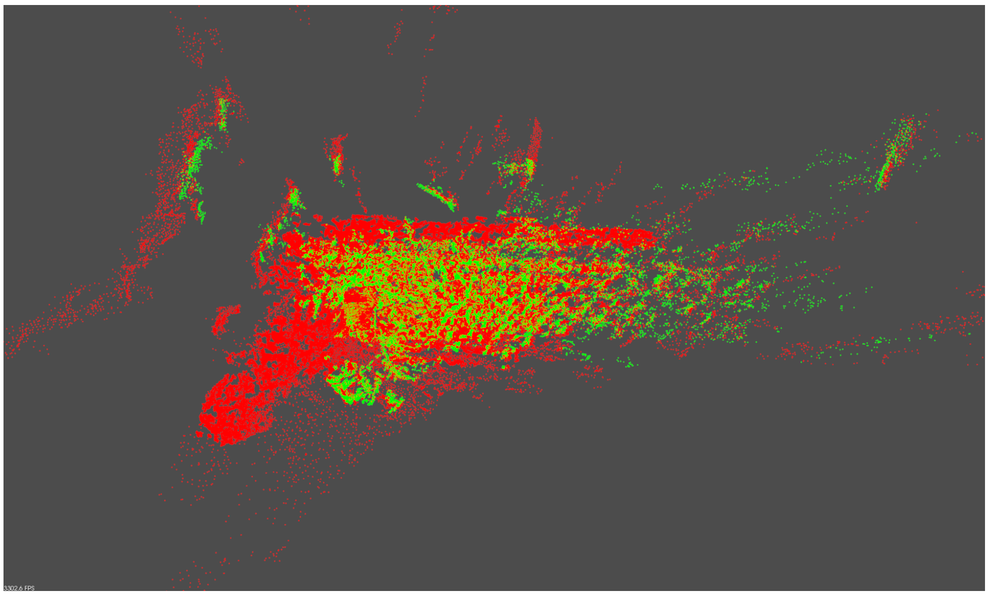


(b)

FIGURE 4.23: Evaluation for different levels of overlap. (a) Estimated scale ratios over different levels of overlap, and (b) The standard ICP fails to align the two point clods and to estimate the scale.



(a)



(b)

FIGURE 4.24: Example of scale alignment in a real-world scene. (a) Original point clouds. (b) Manually registered point clouds with estimated scale ratio.

clouds have different levels of overlap. First we reconstructed a 3D point cloud of 207,583 points. Then we manually remove some points from the left and right sides of the point cloud by a specified percentage to generate point clouds with different levels of overlap. Two point clouds with an overlap rate of 70% are shown in Figure 4.22. As the level of overlap decreases, the estimated scale ratio goes away from the ground truth (1.0 in this case), as shown in Figure 4.23 (a). We have observed that a scale ratio can be estimated reasonably well when the overlap rate is larger than 70%. In contrast, the standard ICP fails to estimate both the scale ratio and the alignment, as shown in Figure 4.23 (b). These results are also available in our video on YouTube.

4.4.4 Real-World Application

Here we demonstrate a motivating example of scale estimation between quiet different point clouds of the same scene of showing real wave-dissipating blocks on a coast. The point clouds consist of 88,021 and 222,781 points reconstructed from 200 and 440 images taken at different times. The proposed method estimated the scale ratio to be 4.3. Figure 4.24 (a) shows the point clouds, and (c) shows the result of manual alignment with the estimated scale ratio. Obviously, the scale aligned point clouds are well registered, which demonstrates that the proposed method works efficiently even on complex real datasets. The full video sequence is also available on YouTube.

4.5 Summary

We have presented some methods for change detection in this thesis. These methods are combined as a surveillance system. Since the surveillance system motivate a regular observation of places such as coast, slopes and highways, where disasters and accidents can happen. Therefore, to detect any obvious temporary changes is very important. However, it is not practical in such situation. So, in this thesis, we divide the detection goal into two sub-goals. The first sub-goal is an online approach described in previous chapters. Then, the second sub-goal is designed as an offline approach, which is used to achieve high accurate detection result.

In this chapter, we have focused on our offline approach. We have proposed a method for estimating the scales of point clouds in order to align them. We defined a keyscale and a scale ratio, which can be used to re-scale one point cloud to the other. By performing PCA of spin images to generate two sets of cumulative contribution rate curves, the proposed method estimates scales by finding the minimum of the curves, or by registering the two sets of curves. Experimental results demonstrated that the proposed method works well both for simple and difficult point cloud datasets.

Future work includes reducing the computational cost for generating spin images from the point clouds and performing PCA. Our current implementation takes several minutes to perform scale estimation for point clouds of the order of 100,000 points. Although a fast response of the system is not necessary for the task we deal with, still we have to accelerate the computation in order to handle much larger number of points.

Chapter 5

Conclusions

In this thesis, we have developed a surveillance system for change detection. This surveillance system can be used in wide areas, such as coast, slopes and highways. Because disasters and accidents can happen with high probability at these wide areas if there are unusual changes exist. For example, at coastal areas, many wave dissipating blocks are installed in order to decrease the power of the waves. If the configuration of the wave blocks is changed by the seawater corrosion, the blocks might be collapsed which is very dangerous when there is earthquake or heavy surge. Therefore any obvious temporary change should be alerted for the users. However, in the complicate surveillance environment, it is very difficult to observe the places all the time. The static electronic camera is very damageable in the outdoor wet and windy conditions. Thus, a real-time automatic unusual change detection surveillance system is useful and important.

In our surveillance system, we need to detect change in a wide area with un-fixed cameras. Obviously, it is very challenging and difficult tasks. Therefore, we analyze the possible implementations and divide the research goal into two sub-goals.

The first sub-goal, which we call online approach, is used in the wide area directly for change detection. Therefore, a fixed camera is not request and the detection should be real-time processing. In order to achieve this sub-goal, we have proposed a client-server system. The client side captures images for server and displays potential change area for operators, and the server side computes and returns estimated results. In order to find a region of change in a query image, first, the nearest image is searched. Then the matching between the nearest image and query image is performed to find 2D keypoints, so that a region of change could be detected by using the correspondences. At last, 3D points are projected onto the changed region on query image.

In order to find a robust way to search nearest image and to do projection, we have proposed a 3D keypoint detection and description method. We also call it 3D-2D matching method. By detecting a sparse set of 3D keypoints from a large number of 3D points, the proposed method enables us to use 3D-2D matching for a fast camera pose estimation. A 3D keypoint is defined as a point that has corresponding 2D keypoints in many training images. The descriptors of these 2D keypoints are used as a descriptor of the 3D keypoint. After detecting 3D keypoints, matching between them and 2D keypoints extracted from query images is performed and camera projection matrices of the query images are estimated.

The second sub-goal, which we call offline approach, is used for improving the accuracy of change detection results. The online approach have provided potential candidate changed areas, however, rough result is not enough for users. So for our offline approach, we have proposed a 3D-3D based method for estimating the scales of point clouds in order to align them. We defined a keyscale and a scale ratio, which can be used to re-scale one point cloud to the other. By performing PCA of spin images to generate two sets of cumulative contribution rate curves, the proposed method estimates scales by finding the minimum of the curves, or by registering the two sets of curves.

In our future work, the separated online and offline approaches should be combined together. So that the users can obtain accurate detection results in time. There are two directions: 1) reducing the computation time while performing 3D-3D alignments, and 2) optimizing 3D-2D algorithm for image based change detection and embedding it into cheap devices such as mobile phones and tablets.

Bibliography

[1] F. Oberti, L. Marcenaro, and C. S. Regazzoni. Real-time change detection methods for video surveillance systems with mobile camera. *European Signal Processing Conference*, September 2002.

[2] C. S. Regazzoni, A. Teschioni, and E. Stringa. A long term change detection method for surveillance applications. *Image Analysis and Processing*, 1311:485–492, 1997.

[3] S. Huwer and H. Niemann. Adaptive change detection for real-time surveillance application. *Third IEEE International Workshop on Visual Surveillance*, pages 37–45, July 2000.

[4] J. L. A. Samatelo and E. O. T. Salles. A new change detection algorithm for visual surveillance system. *Latin America Transactions IEEE*, 10:1221–1226, February 2012.

[5] B. Micusik. Trajectory reconstruction from non-overlapping surveillance cameras with relative depth ordering constraints. *ICCV*, 2011.

[6] Y. Kameda, T. Takemasa, and Y. Ohta. Outdoor see-through vision utilizing surveillance cameras. *ISMAR*, 2004.

[7] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor. Viewpoint independent detection of vehicle trajectories and lane geometry from uncalibrated traffic surveillance cameras. *ICIAR*, 2004.

[8] G. Schindler, M. Brown, and R. Szelisk. City-scale location recognition. *CVPR*, 2007.

[9] E. D. Eade and T. W. Drummond. Unified loop closing and recovery for real time monocular slam. *BMVC*, 2008.

[10] A. Irschara, C. Zach, J. M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. *CVPR*, 2009.

[11] Z. Dong, G. Zhang, J. Jia, and H. Bao. Keyframe-based real-time camera tracking. *ICCV*, 2009.

[12] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *PAMI*, 14 (2):239–256, 1991.

[13] A. E. Johnson and M. Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16:635–651, 1998.

[14] A. E. Johnson and M. Hebert. Image based detection of 3d scene change. *PAMI*, 21:433–449, 1999.

[15] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Narf: 3d range image features for object recognition. *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at IROS*, 2010.

[16] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. *ACM Multimedia*, pages 357–360, 2007.

[17] cheung. n-sift: n-dimensional scale invariant feature transform. *Trans. IP*, 18(9): 2012–2021, 2009.

[18] C. Wu, B. Clipp, X. Li, J. M. Frahm, and M. Pollefeys. 3d model matching with viewpoint invariant patches (vips). *CVPR*, 2008.

[19] G. Baatz, K. Koser, D. Chen, R. Grzeszczuk, and M. Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *IJCV*, 96(3):315–334, 2012.

[20] B. Lin, T. Tamaki, M. Slomp, B. Raytchev, K. Kaneda, and K. Ichii. 3d keypoints detection from a 3d point cloud for real-time camera tracking. *IEEJ Transactions on Electronics, Information and Systems*, 133(1):84–90, 2013.

[21] I. Stamos and P. K. Allen. Geometry and texture recovery of scenes of large scale. *CVIU*, 88(2), 2002.

[22] D. G. Lowe. Fitting parametrized three-dimensional models to images. *PAMI*, 13 (5):441–450, 1991.

[23] B. Lin, T. Tamaki, M. Slomp, B. Raytchev, K. Kaneda, and K. Ichii. Image based detection of 3d scene change. *IEEJ Transactions on Electronics, Information and Systems*, 133(1):103–110, 2013.

[24] Y. Ueno, B. Lin, K. Sakai, T. Tamaki, B. Raytchev, and K. Kaneda. Camera position estimation for detecting 3d scene change. *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, pages 344–350, 2012.

[25] B. Lin, T. Tamaki, B. Raytchev, K. Kaneda, and K. Ichii. Scale ratio icp for 3d point clouds with different scales. *ICIP*, 2013.

[26] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopisi. *CVPR*, pages 1362–1376, 2007.

[27] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopisi. *PAMI*, 32(8):1362–1376, 2010.

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *IJCV*, 80(5):189–210, 2008.

[29] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. *Proceedings of the ACM symposium on User interface software and technology (UIST)*, pages 559–568, 2011.

[30] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60 (2):91–110, 2004.

[31] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[32] E. Akagunduz and I. Ulusoy. Extraction of 3d transform and scale invariant patches from range scans. *CVPR*, 2007.

[33] M. Haker, M. Bohme, T. Martinetz, and E. Barth. Scale-invariant range features for time-of-flight camera applications. *CVPRW*, 2008.

[34] A. S. Mian, M. Bennamoun, and R. Owens. Keypoint detection and local feature matching for textured 3d face recognition. *IJCV*, 79(1):1–12, 2008.

[35] C. Conde and A. Serrano. 3d facial normalization with spin images and influence of range data calculation over face verification. *CVPRW*, 2005.

[36] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3d range scans taking into account object boundaries. *ICRA*, pages 2601–2608, 2011.

[37] R. Unnikrishnan and M. Hebert. Multi-scale interest regions from unorganized point clouds. *CVPR*, 2008.

[38] A. Zabarescu, E. Boyer, and K. Varanasi. Surface feature detection and description with applications to mesh matching. *CVPR*, 2009.

[39] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Second Edition Cambridge University Press (2003)*, 2003.

[40] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. An invitation to 3d vision: From images to geometric models. page 27, 2006.

[41] T. Tamaki. Pose estimation and rotation matrices. *IEICE Technical Report*, 109 (202):59–64, 2009.

[42] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *PAMI*, 13(4):376–380, 1991.

[43] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642, 1987.

[44] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solutions of absolute orientation using orthonorimal matrices. *Journal of the Optical Society of America*, 5:1127–1135, 1988.

[45] P. H. Schonemman. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[46] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3d point sets. *PAMI*, 9(5):698–700, 1987.

[47] K. Kanatani. Analysis of 3d rotation fitting. *PAMI*, 16(5):543–549, 1994.

[48] C. Wu. Siftgpu: A gpu implementation of scale invariant feature transform(sift). URL http://cs.unc.edu/~ccwu/siftgpu/.

[49] J. G. M. Goncaluves. 3d laser-based scene change detection for basic technical characteristics/design information verification. *ESARDA BULLETIN*, 2010.

[50] J. Ryle and N. Hillier. Alignment and 3d scene change detection for segmentation in autonomous earth moving. *ICRA*, 2011.

[51] B. Neuman, B. Sofman, A. Stentz, and J. A. Bagnell. Segmentation-based online change detection for mobile robots. *ICRA*, 2011.

[52] I. Eden and D. B. Cooper. Using 3d line segments for robust and efficient change detection from multiple noisy images. *ECCV*, pages 172–185, 2008.

[53] J. Sato, T. Takahashi, I. Ide, and H. Murase. Change detection in streetscapes from gps coordinated omni-directional image sequences. *ICPR*, pages 935–938, 2006.

[54] T. Pollard and J. Mundy. Change detection in a 3-d world. *CVPR*, 2007.

[55] A. Taneja, L.Ballon, and M. Pollefeys. Image based detection of geometric changes in urban environments. *ICCV*, 2011.

[56] T. Tamaki, Y. Ueno, S. Tanigawa, B. Raytchev, and K. Kaneda. 3d keypoint detection by embedding 2d features for 3d-2d matching. *FCV*, 2012.

[57] The stanford 3d scanning repository. [online]. URL http://www.graphics.stanford.edu/data/3Dscanrep/.

[58] T. Tamaki, S. Tanigawa, Y. Ueno, B. Raytchev, and K. Kaneda. Scale matching of 3d point clouds by finding keyscales with spin images. *ICPR*, pages 3480–3483, 2010.

[59] J. R. Hurley and R. B. Cattell. Producing direct rotation to test a hypothesized factor structure. *Behavioral Science*, pages 258–262, 1962.

[60] L. L. Dyden and K. V. Mardia. Statistical shape analysis. *Wiley*, 1998.

[61] S. Du, N. Zheng, S. Ying, Q. You, and Y. Wu. An extension of the icp algorithm considering scale factor. *ICIP*, 5:193–196, 2007.

[62] T. Zinßer, J. Schmidt, and H. Niemann. Point set registration with integrated scale estimation. *Proc. of International Conference on Pattern Recognition and Image Processing*, 5:116–119, 2005.

[63] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002.

[64] I. T. Jolliffe. Principal component analysis. *Springer-Verlag*, (1), 1986.

Publication List

Journal Papers

- **Baowei Lin**, Toru Tamaki, Marcos Slomp, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "3D Keypoints Detection from a 3D Point Cloud for Real-Time Camera Tracking," IEEJ Transactions on Electronics, Information and Systems, Vol. 133, No. 1, pp. 84-90 (2013).
  http://dx.doi.org/10.1541/ieejeiss.133.84

- **Baowei Lin**, Yuji Ueno, Kouhei Sakai, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "Image Based Detection of 3D Scene Change," IEEJ Transactions on Electronics, Information and Systems, Vol. 133, No. 1, pp. 103-110 (2013).
  http://dx.doi.org/10.1541/ieejeiss.133.103

- **Baowei Lin**, Kouhei Sakai, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "Semi-Rigid Registration of 3D points," Journal of Signal and Information Processing, Vol. 4, No. 3B, pp. 25-29 (2013).
  http://dx.doi.org/10.4236/jsip.2013.43B005

- Haoming Wang, **Baowei Lin**, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "Development of a Client-Server System for 3D Scene Change Detection," Journal of Software Engineering and Applications, Vol. 6, No. 7B, pp. 17-21 (2013).
  http://dx.doi.org/10.4236/jsea.2013.67B004

International Conference Papers

- **Baowei Lin**, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "Scale Ratio ICP for 3D Point Clouds with Different Scales," 2013 IEEE International Conference on Image Processing (ICIP2013), September 15-18, 2013, Melbourne, Australia. (accepted)
- Yuji Ueno, **Baowei Lin**, Kouhei Sakai, Toru Tamaki, Bisser Raytchev and Kazufumi Kaneda: "Camera Position Estimation for Detecting 3D Scene Change," 18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV2012), pp. 344-350, 2012. Kawasaki International Center, Kanagawa, Japan, Feb. 2-4, 2012.

Submitted Journal Paper

- **Baowei Lin**, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "Scale Alignment of 3D Point Clouds with Different Scales," Machine Vision and Applications, Springer. (April 2013 submitted)

Domestic presentations

- Kouhei Sakai, Yuji Ueno, **Baowei Lin**, Toru Tamaki, Bisser Raytchev and Kazufumi Kaneda: "Real-time system of 3D scene change detection," Meeting on Image Recognition and Understanding (MIRU 2011), pp. 1697-1698, 2011, Yokohama, Japan. (in Japanese)
- Kouhei Sakai, Yuji Ueno, **Baowei Lin**, Toru Tamaki, Bisser Raytchev and Kazufumi Kaneda: "Real-time detection of change in a 3D scene," The 15th SANKEN International Symposium, January 12-13, 2012, Osaka, Japan.
- Kouhei Sakai, **Baowei Lin**, Toru Tamaki, Bisser Raytchev, Kazufumi Kaneda and Koji Ichii: "3D Point Cloud Matching based on Energy Minimization of Point Cloud Configuration," the 19th Symposium on Sensing via Image Information (SSII 2013), June 12-14, 2013, Yokohama, Japan. (in Japanese)

Previous Papers

- **Baowei Lin**, Tsukasa Hirashima and Hidenobu Kunichika: "Interactive Environment for Learning by Question-Posing for Beginner's English Learning," The Journal of Information and Systems in Education, Vol. 9, No. 1, pp. 15-24 (2011).
- **Baowei Lin**, Tsukasa Hirashima and Hidenobu Kunichika: "Interactive Question-Posing Environment for Beginners' English Learning," the 18th International Conference on Computers in Education (ICCE2010), pp. 17-24 (2010), Putrajaya, Malaysia.
- **Baowei Lin**, Tsukasa Hirashima and Hidenobu Kunichika: "An Implementation of Interactive Environment for Learning by Question-Posing," Workshop Proc. of the 18th International Conference on Computers in Education (ICCE2010), The 4th International Workshop of Modeling, Management and Generation of Problems/Questions in Technology-Enhanced Learning, pp. 88-90 (2010), Putrajaya, Malaysia.
- **Baowei Lin**, Tsukasa Hirashima and Hidenobu Kunichika:"Interactive English Learning Environment based on Question-Posing for Beginners," the 10th symposium of JSiSE chugoku, pp. 23-28 (2010), Yamaguchi, Japan.