Bournemouth University

Generalised Discriminative Optimisation Algorithms for Augmented Reality: Theory and Practice

Yan Zhao

Supervisor: Prof. Wen Tang
Prof. Jun Feng
Dr. TaoRuan Wan

Faculty of Science & Technology
Bournemouth University

A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

July 2022

Abstract

Augmented Reality (AR) technology achieves the seamless registration between virtual scenes and the real world. Three-dimensional registration is a core method for integrating virtual information and the real world. Registration has always been treated as an optimisation process, involving the design of objective functions and the estimation of gradient directions. Learning-based optimisation methods learn the gradient directions via the least-square methods acting on a parameter vector space and extracted features from data. Learning-based optimisation methods do not require the design of objective functions or the calculation of derivations. This advantage reduces the high complexity and the large storage requirement for inverse Hessian approximation, which is common when using traditional optimisation methods.

This thesis explores learning-based optimisation methods for point cloud registration with the aim for augmented reality applications. Three methods and a computational framework have been proposed: (1) A General Discriminative Optimisation method (GDO) has been proposed to reduce the effect of perturbations on updating gradient directions. The existing learning-based optimisation methods have several drawbacks, one of which is using a single feature to learn gradient paths, which makes the learning vulnerable to perturbations; (2) A Reweighted Discriminative Optimisation method (RDO) has been put forward to explore the asymmetrical contributions of each component of parameter vectors on registration to capture the influence of the component to improve the registration accuracy; (3) A Graph-based Discriminative Optimisation method (GRDO) has been proposed to reduce the storage requirement and computational cost; (4) Finally, a computational framework, SGRTmreg, has been devised to achieve multiple point clouds registration, which is a step forward in the state-of-the-art, since previous learning-based optimisation algorithms have mainly focused on single point cloud registration. Each of the new algorithms has been comprehensively compared with several state-of-the-art traditional registration algorithms as well as recently developed deep learning-based algorithms using public point cloud data sets in order to demonstrate key features (robustness, accuracy, efficiency, stability) of algorithms and its registration performance.

In this thesis, theoretical convergence proofs for the proposed algorithms are also provided in appendixA. The potential of GDO, RDO, and GRDO for 3D point cloud registration is demonstrated through applications of 3D registration in real scenes and object tracking.

Table of contents

List of figures

List of tables

Acknowledgments

My doctoral research is coming to an end. I am full of gratitude for these four years. My achievement is not limited to the research outcomes and experience, but more importantly, the personal growth. The global covid-19 pandemic has had a significant impact on my research, my life, my emotion, and my spirits, but it has made me strong, built my resilience, and given me the courage whenever I face any difficulty in the future. I would like to take this opportunity to express my sincere gratitude to all those who have cared for and helped me.

First of all, I would like to thank my first supervisor, Prof. Wen Tang, for her constant support and invaluable advice when my research came to a standstill and for her encouragement and meticulous care when my emotion and health were poor. She is not just a professor full of profound knowledge and insightful ideas but also a friend sharing your happiness and sadness.

I want to thank my second supervisor, Prof. Jun Feng, who has provided me with the opportunity to study abroad at Bournemouth University. I will never forget the care she had given me during the Covid-19 pandemic. I would also like to thank Prof. TaoRuan Wan at the University of Bradford, who always gives me valuable advice when I have no idea which journal is a good fit for my manuscript.

Besides, I would like to thank Long Xi and QingHong Gao for their research collaborations on point clouds. Without their help, the experimental part of my project could not have been completed. They have been very patient with me and provided detailed guidance on coding. The spark of our discussion has always inspired novel ideas, making me more knowledgeable on various research topics.

Outside academic circle, I have shared memorable moments with many awesome people: Chi Zhang, WeiLai Xu, Ge Zheng, Hua Zheng, Long Xi, QingHong Gao, roommates, and my enthusiastic landlady. We often have meals and trips around Bournemouth, do exercises, and chat together. Their care and help made me feel at home. Thanks to Ying Chen, Hao Tang and Pei Su for your accompany and tolerance. Our 10 years of friendship will not stop.

Lastly but not least, I express my sincere gratitude to my parents. Without your love, support, and encouragement, my PhD study will not be successful, especially through the tough times of overcoming the immense pressure on research and life obstacles during the Covid-19 epidemic.

Thanks to all the people working hard in scientific research. Be brave no matter what life is presenting to you.

Declaration

I, Yan Zhao, hereby declare that this thesis represents my own work which has been done after registration for the degree of P.h.D at Bournemouth University and has not been submitted to any other institute for any award or qualifications. I also confirm that this work fully acknowledges the contributions from the work of others.

Yan Zhao
July 2022

List of Abbreviations & Notations

Abbreviations

$GRDO_{NTF}$ Graph-based Reweighted Discriminative Optimisation without transfer learning

$GRDO_{TF}$ Graph-based Reweighted Discriminative Optimisation with transfer learning

3DSC 3D Shape Contexts

3D Three Dimensions

Adagrad Adaptive Gradient Algorithm

APSR Articulated Point Set Registration

AR Augmented Reality

BFGS Broyden–Fletcher–Goldfarb–Shanno Algorithm

BnB Branch and Bound

CFP Closed-form Projections

CONReg Convex Programs on Registration

CPD Coherent Point Drift

D3Feat Dense Detection and Description of 3D local Features

DCP Deep Closest Point

DLD Dependent Landmark Drift

DNN Deep Neural Network

DO Discriminative Optimisation

DPGMM Dirichlet Process Gaussian Mixture Mode

EA Evolutionary Algorithm

EM Expectation-Maximisation

FGR Fast Global Registration

FPFH Fast Point Feature Histogram

GDO General Discriminative Optimisation

GentleBoost Gentle Adaptive Boosting

Geo-CNN Geometric-induced Convolution Neural Network

GLR Graph-Laplacian Regularisation

GMM Gaussian Mixture Model

GOGMA Globally-Optimal Gaussian Mixture Alignment

GO Globally optimal

GRDO Graph-based Reweighted Discriminative Optimisation

HGMR Hierarchical Gaussian Mixture Registration

HOG Histogram of Oriented Gradients

ICP Iterative Closest Point

IEBM Iterative Error Bound Minimisation

IRLS Iteratively Reweighted Least Squares

JRMPS Joint Registration of Multiple Point Sets

JS Jensen-Shannon

KKT Karush–Kuhn–Tucker

L2E L2-Minimising Estimate

LK Lucas & Kanade Algorithm

LLE Local Linear Embedding

LSR Least Square Registration

MCC Maximum Correntropy Criterion

MLMD Maximum Likelihood Mixture Decoupling

MR Manifold Regularisation

MSE Mean Square Error

MVCNN Multiview Convolution Neural Network

MVDesc Multi-View Local Descriptor

NAG Nesterov Accelerated Gradient

NDT Normal Distributions Transform

NN Neural Network

PDA Probabilistic Data Association

PM Probabilistic Model

PRNet Partial Registration Network

QPCCP A Quadratic Programming based Cluster Correspondence Projection Algorithm

RDO Reweighted Discriminative Optimisation

RelativeNet Relative Pose Estimation Network

RGBD RGB plus Depth

RMBP Robust Matching Using Belief Propagation

RMSProp Root Mean Square Propagation

RPM Robust Point Matching

SA Simulated Annealing

SCIF Split Covariance Intersection Filter

SDM Supervised Descent Method

SDTM Schrodinger Distance Transform

SGO Stochastic Global Optimisation

SIFT Scale-invariant Feature Transform Descriptor

SPSR Stochastic Point Set Registration

SSFR Structured Scene Feature-based Registration

SSU Supervised Sequential Update Methods

SVD Singular Value Decomposition

SVGM Support Vector–parametrised Gaussian Mixture

SVMs Support Vector Machines

SWS Soft-weighted Selection

TEASER Truncated Least Squares Estimation And Semi-definite Relaxation

TLS Truncated Least-squares

Notations

β the parameter measures the similarity of graph structure

μ_m the mean value of points in a box

σ_m^2 the covariance value of points in a box

$\breve{S}$ the collection of training point clouds

δ Dirac delta function

γ_i the coefficient of penalty function φ_i

$\hat{\mathbf{x}}_k$ the k-th column of the diagonal matrix $\mathbf{A}$

$\hat{\sigma}$ controls the width of the exp function

λ the hyperparameter for learning $\mathbf{D}_{t+1}$

$[Y]_k$ the k_{th} row of Y

$[y]_k$ the k_{th} element of y

$[\Delta\mathbf{x}]_l$ the l_{th} element of $\frac{\partial\Phi(\mathbf{x})}{\partial\mathbf{x}}$

$[\mathbf{E}]_{:,k}$ the k-th column of matrix $\mathbf{E}$

$[\mathbf{E}]_k$ the k-th row of matrix $\mathbf{E}$

$[\mathbf{h}(x;\mathbf{S})]^c$ the coordinates-based feature

$[\mathbf{h}(x;\mathbf{S})]^d$ the density-based feature

$\mathbb{R}^3$ finite-dimensional real vector space

$\hat{\mathbf{S}}$ the solution set

$\mathbf{A_d}$ adjacency matrix

$\mathbf{A}$ the diagonal matrix with diagonal elements of 0.1

$\mathbf{Coordinates_M}$ the coordinates of target point cloud

$\mathbf{Coordinates_S^i}$ the coordinates of extracted points in candidate training point cloud

$\mathbf{C}$ the diagonal matrix whose diagonal element is a constant c

$\mathbf{C}_M^k$ the coordinates of test points in cluster $\mathbf{C}_k$

$\mathbf{C}_S^k$ the coordinates of training points in cluster $\mathbf{C}_k$

$\mathbf{C}_k$ the $k-th$ cluster of the mixture $\mathbf{P}$

$\mathbf{c}_k$ the k-th column of matrix $\mathbf{C}$

$\mathbf{c}_M$ represents the counted number of the key points fallen into each uniform grid

$\mathbf{Degree_M}$ the degree list of target point cloud

$\mathbf{Degree_S}$ the degree list of training point cloud

$\mathbf{D}_{t+1}$ the sequence of regressors in learning-based optimisation algorithms

$\mathbf{E_r}$ a $N \times p$ matrix made up of vector $\mathbf{Err}^i$

$\mathbf{Err}^i$ the normalization of vector $\mathbf{err}^i$

$\mathbf{err}^i$ the fitting error of the $\mathbf{Q}_i$ on the whole parameter vector space

$\mathbf{E}$ distance matrix

$\mathbf{E}^c$ counts the similarity of the normal vectors of point clouds.

$\mathbf{f}(\mathbf{x})$ objective function in general optimisation algorithms

$\mathbf{F}(\mathbf{s}_b; x)$ the function that applies rigid transformation with parameter x to target point $\mathbf{s}_b$

$\mathbf{f}_M$ the feature of M

$\mathbf{f}_S$ the feature of S

$\mathbf{F}_{t+1}$ the sequence of regressors mapping the extracted features

$\mathbf{G}$ uniform grid

$\mathbf{g}^+$ the set of grids on the 'front' of s_1

$\mathbf{g}^-$ the set of grids on the 'back' of s_1.

$\mathbf{g}_j$ residual function

$\mathbf{h}$ the function extracting features

$\mathbf{H}_f$ the *collaboration* of different features

$\mathbf{H}f(\mathbf{x}_t)$ Hessian matrix

$\mathbf{M}$ source model

$\mathbf{m}$ the point in model point cloud $\mathbf{M}$

$\mathbf{NormalVector}_S$ the normal vectors set including the normal vectors of the collection of the candidate similar training point clouds

$\mathbf{n}_a$ the normal vector of the source model point $\mathbf{m}_a$

$\mathbf{N}_m$ the number of points in source point cloud $\mathbf{M}$

$\mathbf{N}_s$ the number of points in target point cloud $\mathbf{S}$

$\mathbf{P}$ the coordinates of the mixture of point clouds

$\mathbf{P}_m$ the probability of measuring each point s_j of $\mathbf{S}$ in a box

$\mathbf{q}_j^i$ the j-th elements of the i-th sample $\mathbf{Q}_i$

$\mathbf{s_m}$ the point in target model$\mathbf{S}$ corresponding to $\mathbf{m}$

$\mathbf{S_p}$ sparse matrix

$\mathbf{Scene_C}$ the color vector of medical scene

$\mathbf{Scene_P}$ the coordinates vector of the cluster grouped through the $Color_\beta$

$\mathbf{S}$ target model

$\mathbf{S}_a^+$ the set of target points on the 'front' of source model point $\mathbf{m}_a$

$\mathbf{S}_a^-$ the set of target points on the 'back' of source model point $\mathbf{m}_a$

$\mathbf{T}^*$ optimal transformation matrix

$\mathbf{T}$ transformation matrix

$\mathbf{W}_t$ weighting diagonal matrix

$\mathbf{w}_t$ weighting vector

$\mathbf{x_0}$ the initial parameter vector

$\mathbf{x}_*$ optimal solution / ground truth

$\mathbf{x_{t+1}}$ the parameter vector at the next iteration

$\mathcal{N}\left(\mu_m, \sigma_m^2\right)$ normal distribution

C an operator

$F\left(\mathbf{y};\mathbf{x}\right)$ applies rigid transformation with parameter $\mathbf{x}$ to vector $\mathbf{y}$

$\nabla f\left(\mathbf{x}_t\right)$ the first order derivation of function

$\odot$ Hadamard product

ϕ_i the derivative of φ_i

Φ the objective function in the proposed GDO method

τ_M the index of the cluster that has the largest number of test points

τ_S the index of the cluster that has the largest number of training points

L loss function

R regularisation term

$\tilde{\mathbf{x}}_{t_k}$ the 'detector' exploring the 'structure' of ground truth

φ penalty function

$Color_\beta$ the color information of medical scene

d_E Euclidean distance

d_H Hamming distance

err_k^i the fitting error of the i-th sample on the k-th dimension of the parameter vector space

L the length of the degree list of target point cloud

N the number of problem instances

N_S^k the number of test points in $\mathbf{C}_k$

N_S^k the number of training points in $\mathbf{C}_k$

p the index of the normal vector of M

q the index of $\mathbf{NormalVector}_S$

R_M the proportion of $\mathbf{C}_M^k$ in $\mathbf{Coordinates_M}$

R_S the proportion of $\mathbf{C}_S^k$ in $\mathbf{Coordinates_S^i}$

s_1 the key point in S

z normalizes $\mathbf{h}$ to sum to 1

TO my parents, HaiLong Zhao, Ying Jia.

Chapter 1

Introduction

1.1 Background

Augmented Reality (AR) technology integrates our physical world with the virtual scene to enhance the human perception of and interaction with the real world. AR has become more widely used in education [1], entertainment [2], medicine [3] and other fields. The core components of Augmented Reality technology include display technology, interaction technology, and object recognition and tracking technology. With the development of imaging equipment and hardware computing power, the current object tracking technology based on computer vision has made significant progress. Optical tracking as a relatively low-cost and accurate solution can be classified into two categories: marker-based tracking and marker-less tracking. Marker-based tracking technique is unfeasible to be applied in real scenes because it relies on artificial patterns placed in the scene to estimate the camera pose. Therefore, most recent research has focused on developing efficient marker-less tracking algorithms. Unlike marker-based tracking, the marker-less tracking method relies not on artificial patterns but on natural objects found in the scene [4]. As the critical technique in the marker-less tracking method, model-base tracking can be divided into three stages: modelling, visual information processing and tracking. Tracking involves the objects matching or point clouds registration. AR is defined as a combination of virtual and real information, real-time interaction, and three-dimensional registration and summarises that the registration problem is one of the most basic problems limiting AR applications [5]. Objects in the real world and the virtual world must be properly aligned with each other, or the coexistence between the information in the two worlds will be compromised. Without accurate registration, AR will not be viable in many applications, such as medicine and navigation [6].

Three-dimensional registration, as a fundamental problem in AR, is to find a transformation $\mathbf{T}$ to be applied to a model point cloud $\mathbf{M}$ such that the difference between $\mathbf{M}$ and scene point cloud $\mathbf{S}$ is minimised. Therefore, the registration problem can be cast as an optimisation problem, where the transformation parameters in $\mathbf{T}$ are adjusted to obtain an optimal solution of the matching problem.

Formulating registration as an optimisation problem faces two main challenges: *1)* designing an objective function that has an optimal solution and is robust to various perturbations; *2)* selecting an efficient and suitable search method to search the optimal solution for the designed objective function.

The objective function designed for registration is usually the combination of a least-square method and various regular terms. The least-square method is always devised based on different data spaces (coordinates, features, or density). Due to its conceptual simplicity and high usability, the least-square method has been widely applied to model objective functions for numerous tasks. Nevertheless, in the presence of noise, missing data, outliers, and other perturbations, the least-square method will cause over-fitting and lack robustness. In this case, regularisation or other constraints are added to the least-square method to improve the robustness of algorithms and avoid over-fitting. However, different regularisation has its strength and weakness. Thus, it is a challenge to design suitable regularisation for various least-square functions and give full play to the maximum utility of the 'cooperation' of regularisation and least-square functions for avoiding over-fitting or improving robustness.

The selection of search methods determines whether the optimal solution of the objective function can be found. The widely used search methods for optimisation tasks are Gradient-based methods, which updates search directions according to the gradient information of objective functions during iterations. Newton's method [7], one of the gradient-based optimisation algorithms, is a powerful technique due to its quadratic convergence. Nevertheless, Newton's method requires cost functions to be twice differentiable, and the Hessian matrix needs to be positive definite, limiting its applications in many cases. As the alternative to Newton's method, the Quasi-Newton method [8] generates an estimation of the inverse Hession matrix for finding the local maxima or minima of objective functions, which is often used when the Jacobian or Hessian matrix is unavailable or is too expensive to compute. However, the lack of precision in the Hessian estimation may lead to slow convergence. Another potential disadvantage is the need to store inverse Hessian approximation, which will require a large amount of memory. The calculation and storage of the gradient information pose another challenge because of the large number of parameters and high complexity of Hessian matrix inversion in many visualisation tasks

Learning-based optimisation is proven to be efficient in overcoming the mentioned challenges. It learns gradient direction and updates gradient paths according to data features without directly calculating the Jacobian matrix or the Hessian matrix of objective functions. Supervised Descent Method (SDM) [9] [10] learns a sequence of linear maps as gradient directions through minimising nonlinear least-squares functions in a feature space, which avoids the expensive computation of the Jacobian and Hessian metrics. The Discriminative optimisation (DO) method [11] extracts the data features and mimics gradient descent based on the extracted features without the explicit modelling of the objective function of registration.

Specifically, learning-based optimisation acts on the parameters space and learns the updating gradients by making the currently estimated parameter vectors approximate the

ground truth. The updating maps are learned from data features, which are more robust to perturbations than the gradient calculated from the devised specific objective functions. The learned updating map can also be directly used to estimate the transformation parameters of the same point cloud under different transformations or different perturbations without re-calculating the transformation of each point in the point cloud, and the whole process avoids calculating the Hessian matrix inversion or the Jacobian matrix and achieves less computational cost.

Although learning-based optimisation algorithms can learn gradient paths without designing objective functions and calculating the Hessian inversion, they are primarily used in computer image processing. In terms of point cloud registration, there is still room to improve: (1) The existing learning-based optimisation algorithms use a single feature to learn the gradient path, making the gradient path vulnerable to various perturbations. (2) Learning-based optimisation algorithms estimate the final parameters by approaching the currently estimated parameter vector to ground truth, ignoring the different influence of each component of the vector on the registration results. (3) The existing learning-based optimisation algorithms have limited ability to achieve multiple point clouds registration. This research aims to improve the performance of the existing learning-based optimisation algorithms on registration in terms of robustness, accuracy and efficiency and further explore the potential of learning-based optimisation algorithms to make them able to achieve multiple point clouds registration like deep learning algorithms.

1.2 Research Contributions

The existing learning-based optimisation algorithms utilise a single data feature to learn the gradient path, causing the searching paths to fall into a local minimum. This thesis extracts different features from point clouds and learns gradient paths based on the *collaboration* of the extracted features to reduce the effect of perturbations on the search paths and make the path converge to an optimal, further improving the robustness of learning-based optimisation algorithms on registration. Besides, the existing learning-based optimisation algorithms learn the updating map through a least-square method that makes a residual vector approach to zeros to approximate the solution. Specifically, they directly minimise the least-square function in parameter space to get the updating maps, which ignores the influence of each component of parameter vectors on fitting errors, attaining the undesirable local optima. In this case, the impact of each component and the asymmetrical contributions of the components on fitting results are explored and an asymmetrical parameter treatment scheme to improve the accuracy of parameter estimation in least-square problems is proposed in this thesis. Moreover, the existing learning-based optimisation algorithms have mainly focused on the single point cloud registration with limited ability to handle a large number of multiple point clouds registration efficiently compared with deep learning-based algorithms. To extend the existing learning-based optimisation algorithms on multiple point cloud registration, a framework called SGRTmreg composed of search scheme, a learning-based optimisation method called graph-based reweighted discriminative optimisation

(GRDO), and the transfer learning technique is devised in this study. Due to the high storage requirement and computational cost of the existing learning-based optimisation algorithms on the registration of dense point cloud, a graph-based optimisation method is put forward to make the learning-based optimisation algorithms efficient and less computation and memory requirement. Each method is comprehensively compared with several state-of-the-art traditional registration algorithms as well as recently developed deep learning-based algorithms on public point cloud data sets. To evaluate the performance of the proposed methods, comparative experiments on real scenes are conducted, such as point cloud registration and object tracking. The experimental results show the better performances of the proposed algorithms in terms of efficiency, accuracy, robustness, and stability than other comparative registration algorithms.

More specifically, the main contributions of this work are:

1 **General Discriminative optimisation for point cloud Registration**
The existing learning-based optimisation algorithms learn the gradient maps via a single feature extracted from data sets, causing the learned updating paths to be vulnerable to perturbations, thus falling into a bad stationary point. The General Discriminative optimisation method (GDO) extracts different features from point clouds and adjusts updating paths in line with the *collaboration* of these features to reduce the effect of perturbations on updating directions. Compared with the state-of-the-art learning-based optimisation method (DO) [12], traditional registration algorithms, and deep-learning-based algorithms, the proposed General Discriminative optimisation method is prominent in dealing with registration under various perturbations with higher robustness. Experiments show that the registration accuracy of GDO is almost 4.00% higher than that of DO on the Bunny [13], Chef [14], and Dancing Children models. The registration accuracy of GDO is almost 25.00% higher than that of DO on the Indoor Scene models [15]. And the successful Rate of GDO is almost 14% higher than that of DO. GDO has higher stability than conventional algorithms and deep-learning-based algorithms on the ModelNet40 data set [16]. The decline of the registration accuracy of GDO on the registration under higher rotations is 6.91%, and the decline is as high as 78.49 % for PointNetLK [17]. The registration accuracy of GDO is 26.00% higher than that of DO on the registration under different noises. Meanwhile, the stability of GDO is 7.91% higher than that of PointNetLK.

2 **Reweighted Discriminative optimisation for Least-Squares Problems with Point Cloud Registration**
The learning process of the gradient paths in learning-based optimisation algorithms is based on the least-square method, which directly learns the gradients paths by approaching the residual parameter vectors to zeros, ignoring the influence of each component of parameter vectors on final fitting results, causing the estimated parameter vectors not accurate. The reweighted Discriminative optimisation method (RDO) explores the influence of parameter components on final fitting results and devises an

asymmetrical parameter treatment scheme to improve parameter estimation accuracy in least-squares problems. Experimental results illustrate the higher robustness, accuracy, and efficiency of RDO than the advanced learning-based optimisation algorithms. The registration accuracy of RDO is almost 40.00% higher than that of DO on the Happy [13] and Hand [18] models. RDO has 2.11% higher accuracy than DO when achieving the registration on the Dancing Children model. And the successful Rate of RDO is almost 5% higher than that of DO. In addition, the computation time of RDO is almost 20% lower than that of DO. In addition, in the registration experiment on the ModelNet40 data set, the registration success rate of RDO is nearly 30% higher than that of PointNetLK.

3 **SGRTmreg: A Learning Based optimisation Framework for Multiple Point Clouds Registration**
Although learning-based optimisation algorithms learn gradients from data without designing objective functions to avoid calculating the inversion Hessian matrix, it has mainly focused on the single point cloud registration with limited ability to handle a large number of multiple point clouds registration efficiently compared with deep learning-based algorithms. A framework, SGRTmreg, composed of a search scheme, a learning-based optimisation method called graph-based reweighted discriminative optimisation (GRDO), and transfer learning, is proposed to achieve multiple point clouds registration, which maintains the high accuracy and robustness of GRDO also develops the ability of GRDO for multiple point clouds registration. Experimental results illustrate the higher robustness, accuracy, and efficiency of GRDO than the conventional algorithms and other learning-based optimisation algorithms. The registration accuracy of GRDO is almost 41.93% higher than that of BCPD on the registration of synthetic data sets under various rotations. The accuracy of GRDO when registering point clouds under various noises is almost twice that of BCPD and is 15.33% higher than that of DO. The accuracy of GRDO is similar to the accuracy of BCPD when achieving the registration under various outliers. They both have almost 25.00% higher accuracy than DO. When handling the registration with small rotations on the ModelNet40 data set, PointNetLK has higher accuracy than other algorithms. When handling the registration with large rotations, GRDO has 4.78% higher accuracy than BCPD and 15.06% higher accuracy than PointNetLK. In addition, GRDO has the greatest accuracy and stability than other algorithms when achieving the registration with multiple perturbations.

1.3 Thesis Outline

Chapter 1: Introduction of research background and main contributions.
Chapter 2: Literature review on point cloud registration, and introductions of traditional optimisation algorithms and learning-based optimisations for solving point cloud registration computer vision tasks, in terms of establishing the optimisation model and the solution

of the optimisation model.
Chapter 3: A General Discriminative optimisation algorithm (GDO). This chapter proposed a general learning-based optimisation method to improve the robustness of the registration method through learning update directions via the *collaboration* of multiple extracted features.
Chapter 4: A Reweighted Discriminative optimisation algorithm (RDO). This chapter devised an asymmetrical parameter treatment scheme to improve the accuracy of parameter estimation in least-squares problems.
Chapter 5: A framework called SGRTmreg for multiple point cloud registration. This chapter described the framework SGRTmreg composed of search scheme, a learning-based optimisation method called graph-based reweighted discriminative optimisation (GRDO), and the transfer learning technique to achieve multiple point clouds registration.
Chapter 6: Experiments and applications of point clouds registration in real scenes and object tracking.
Chapter 7: Conclusion and Discussions.
Appendix: The convergence proofs and codes.

1.4 List of Publications

Accepted Paper

1) **Yan Zhao**, Wen Tang, Jun Feng, TaoRuan Wan, Long Xi. General Discriminative optimisation for point cloud Registration. *Computers & Graphics (2021)*

2) **Yan Zhao**, Wen Tang, Jun Feng, TaoRuan Wan, Long Xi. Reweighted Discriminative optimisation for Least-squares Problems with Point Cloud Registration. *Neurocomputing (2021)*

3) Long Xi,**Yan Zhao**, Long Chen, QingHong Gao, Wen Tang, TaoRuan Wan, Tao Xue. Recovering Dense 3D Point Clouds from Single Endoscopic Image. *Computer Methods and Programs in Biomedicine (2021)*

Paper Under Review

1) **Yan Zhao**, Wen Tang, Jun Feng, TaoRuan Wan. SGRTmreg: A Learning Based optimisation Framework for Multiple Point Clouds Registration. (submitted to *Pattern Recognition*)

2) QingHong Gao, **Yan Zhao**, Wen Tang, TaoRuan Wan, Long Xi. Break and Splice: A Statistical Method for Non-rigid Point Cloud Registration. (submitted to *Pattern Recognition*)

Chapter 2

Related Works

This chapter reviews several works on point cloud registration and numerical optimisation related to our research's aims and core technique. First of all, a thorough classification and review of 3D point cloud registration methods are summarised according to the recent fast development of registration methods (1992-2021), including both conventional algorithms and current deep learning methods. Moreover, the registration problem is analysed in terms of optimisation. After that, traditional numerical optimisation algorithms are discussed in two parts: the establishment of the optimisation model and the solution of the optimisation model. Finally, learning-based optimisation methods are introduced, which learn the appropriate cost function for a given task and the search direction without first-order and second-order information. In particular, the Discriminative optimisation method [19] is the basis for the proposed algorithms in this thesis.

2.1 Point Cloud Registration

Point cloud registration is the process of finding a spatial transformation that aligns two point clouds, which can be cast as the problem of minimising the difference between two point clouds. The problem can be formulated as follows:

Let $\{\mathbf{M},\mathbf{S}\}$ be two finite-size point cloud in a finite-dimensional real vector space $\mathbb{R}^3$, which contains $\mathbf{N}_m$ and $\mathbf{N}_s$ points, respectively. The problem is to find a desirable transformation mapping $\mathbf{T}^*$ to make the difference between the moving model $\mathbf{M}$ and the target model $\mathbf{S}$ minimised. The output of the registration problem is therefore the optimal transformation $\mathbf{T}^*$ such that $\mathbf{M}$ is best aligned to $\mathbf{S}$.

$$\mathbf{T}^* = \arg\min_{\mathbf{T}} dist\left(\mathbf{T}\left(\mathbf{M}\right),\mathbf{S}\right) \tag{2.1}$$

Where $\mathbf{T}$ denotes the set of all possible transformations, and $\mathbf{T}\left(\mathbf{M}\right)$ represents the transformed model. The *dist* shows the distance function acting on different spatial measurements. Euclidean distance is widely used to measure spatial distance.

$$dist\left(\mathbf{T}\left(\mathbf{M}\right),\mathbf{S}\right) = \sum_{\mathbf{m}\in\mathbf{T}(\mathbf{M})} \|\mathbf{m}-\mathbf{s}_m\|_2^2 \tag{2.2}$$

$$\mathbf{s}_m = \arg\min_{\mathbf{s}\in\mathbf{S}} \|\mathbf{s}-\mathbf{m}\|_2^2 \tag{2.3}$$

Where $\|\cdot\|_2$ denotes the l_2 norm, $\mathbf{s}_m$ is the corresponding point in $\mathbf{S}$ that is closest to a given point $\mathbf{m}$ in $\mathbf{T}(\mathbf{M})$.

The above equations formulate the point cloud registration process and involve four significant factors: the way to transform models- $\mathbf{T}$, the correspondences- $\mathbf{s}_m$, the spatial measurement- *dist* and the methods searching for the optimal transformation $\mathbf{T}^*$. To learn the different roles of these factors in point cloud registration, 3D registration methods proposed within 20 years are analysed and classified, as shown in Figure.2.1. In addition, these significant factors constitute the optimisation modelling process for point cloud registration. 3D registration will be cast into an optimisation problem and be discussed in the subsequent chapter from the perspective of the optimisation modelling.

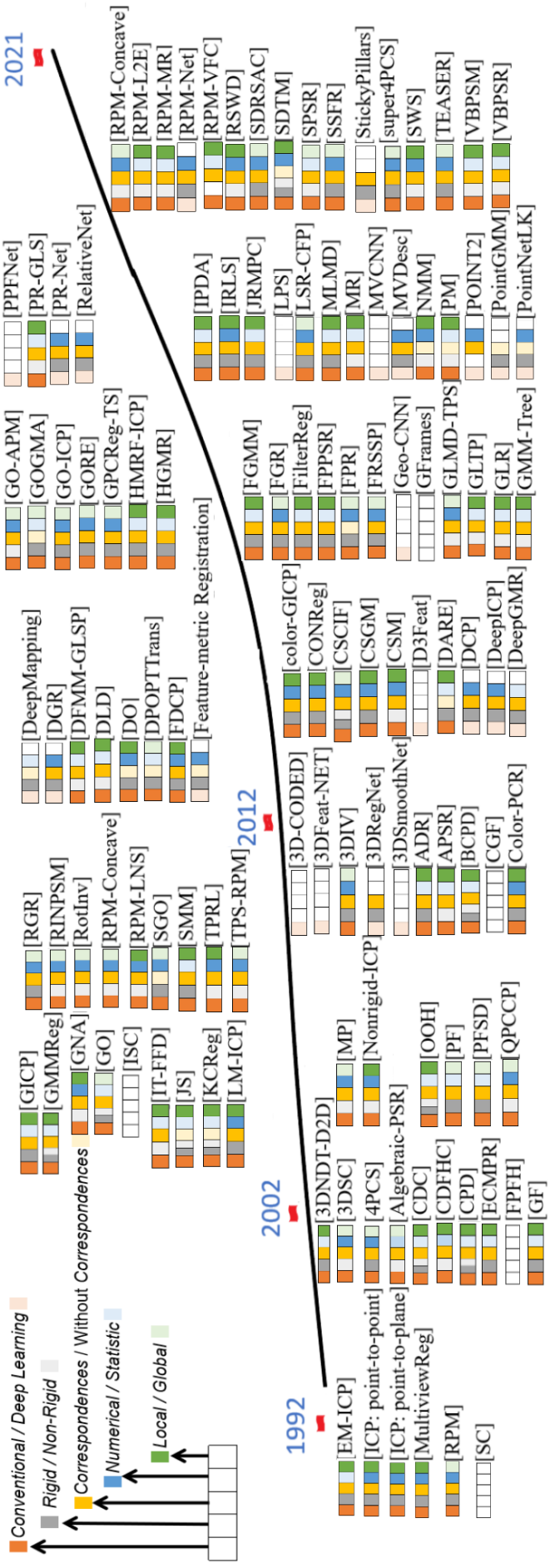


Fig. 2.1 Overview of the most 3D point cloud registration methods

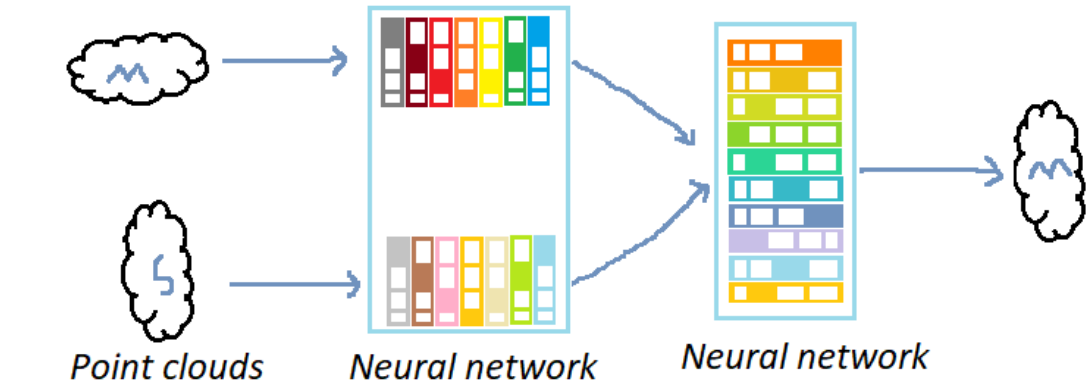
The way to transform models- **T**. The registration involving a transformation that only consists of translation and rotation is called rigid registration; otherwise, it is called non-rigid registration. The latter is usually applied to transform deformable point clouds (e.g. [3DSC] [20], [Nonrigid-ICP] [21], [APSR] [22] and [GMM-Tree] [23] ,etc.).

Correspondences- $\boldsymbol{s}_m$. Most of the registration algorithms achieve the registration via the given correspondences or searching for the correspondences before optimisation, few methods estimate the transformation mapping without correspondences (e.g. [DO] [19], [FGR] [24], [Gogma] [25] and [SGO] [26], etc.).

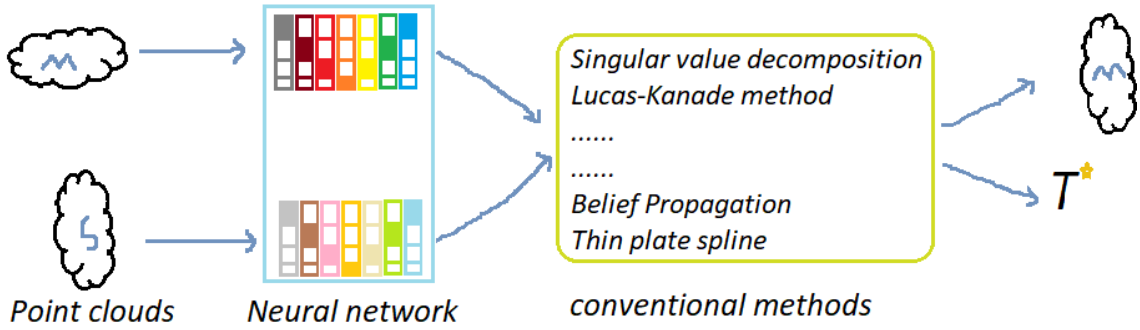
Spatial measurement- dist. If the element in the distance function *dist* is represented by the statistical distribution or the formulation of the distance function is designed as a statistical model, such as Gaussian mixture model, Student's t-distribution, and so on, these registration methods are classified as statistic registration approaches (e.g. [GMMReg] [27], [DLD] [28], [HGMR] [29] and [PM] [30],etc.). The Least-Square regression and Thin plate spline algorithm are categorised into the numerical registration approaches (e.g. [ICP] [31], [CONReg] [32], [IRLS] [33] and [SSFR] [34],etc.).

The methods searching for the optimal transformation $\mathbf{T}^*$. In addition to designing appropriate objective functions for registration, the way to search for the optimal transformation $\mathbf{T}^*$ or the way to attain the solution of the eq.2.1 also determines the performance of the registration methods. [SPSR] [35], [RPM-concave] [36], [LSR-CFP] [37] and [CSCIF] [38] etc. achieve global point cloud alignment via global optimisation techniques, such as Branch and bound (BnB) algorithm, Simulated annealing (SA) algorithm and Evolutionary algorithm (EA). Due to the high computational cost for global optimisation, gradient-based optimisation as the local optimisation technique is widely used to search for the optimal transformation for most registration methods, such as Gradient-Descent method used in [QPCCP] [39], Gauss-Newton method applied in [SDTM] [40] [RPM-L2E] [41] [SWS] [42] and Levenberg–Marquardt method adopted in [PDA] [43].

Recently, the success of deep learning techniques in image processing has been extended to the 3D point clouds. Various deep learning models have been proposed to achieve 3D point cloud registration. The way of deep learning to solve the 3D point cloud registration is similar to that of the mentioned conventional algorithms. Its essence is also to make the difference between point clouds minimised while estimating the transformation information. Deep learning algorithms can be classified into two categories in terms of the structures of the deep learning framework in 3D point cloud registration, as shown in Figure.2.2.



(a) The transformation is estimated through a deep learning model.



(b) The transformation is estimated via conventional methods.

Fig. 2.2 The structures of deep learning methods used for 3D point cloud registration

Figure.2.2 shows the structures of deep learning methods used to address the 3D point cloud registration task. Both structures use neural networks to extract 3D point cloud features and estimate the transformation parameters in a global/local feature space. The difference between the two structures mainly focuses on the pose estimation module. (a) displays a rough registration structure. The second network module in (a) can be regarded as a regressor acting on the feature space, and the output of this structure is often the transformed point cloud, which is always used in non-rigid registration, such as [PRNet] [44], [3DRegNet] [45], [RelativeNet] [46], [D3feat] [47], [Geo-CNN] [48] and [MVCNN] [49]. (b) shows a fine registration structure. Conventional registration methods replace the second network module in (a). For example, singular value decomposition (SVD) is used to solve the pose estimation based on the given correspondences in [RPM-Net] [50], [DCP] [51] and [DeepICP] [52]. Lucas–Kanade method acts on global feature space in [Pointnet-LK] [53] to address the registration task. [MVDesc-RMBP] [54] leverages Belief Propagation to attain robust point cloud matching. The outputs of this structure include the transformed point cloud and the transformation parameters. The registration accuracy of this structure relies not only on the robustness of the extracted features from the first network but also on the performance of the conventional registration methods.

The critical steps in deep-learning-based registration methods are the feature extracting for the regression-based pose estimation module and the correspondences seeking for the pose estimation module based on conventional registration methods. Both significantly

determine the accuracy and robustness of the registration. However, the latter not only depends on the robustness of the features extracted but also hangs on the outlier rejections, which often be converted as the selection of the inner points. [DCP] [51] and [StickyPillars] [55] extract reliable correspondences through adding the attention module. Adding the attention module in deep-learning-based registration is similar to regularisation or constraints to the objective functions in conventional registration methods.

Overall, the point cloud registration method based on deep learning is similar to the traditional point cloud registration method in structure, which can be summed up to feature extraction and pose estimation. This thesis treats deep-learning-based registration only in passing and focuses instead on the conventional registration methods. The subsequent part will mainly focus on analysing conventional registration methods in terms of optimisation modelling.

The design of the distance function *dist* and the selection of the metric space (coordinates,features,etc.) in Equation.2.1 constitute the modelling process of registration problems. The objective of the modelling is to find an optimal transformation $\mathbf{T}^*$ to make the difference between the source model $\mathbf{M}$ and the target model $\mathbf{S}$ minimised. Once the registration model has been formulated, an optimisation algorithm can be used to find its solution. Local optimisation algorithms seek a local solution at which the objective function is smaller than at all other feasible nearby points. Global optimisation algorithms search for the global solution, the point with the lowest function value among all feasible points. The modelling process and the subsequent seeking for optimal solutions build the optimisation modelling of registration tasks.

In short, in optimisation modelling, establishing an optimisation model is the process of identifying objectives, variables, and constraints for a given problem. The solution of the constructed model is to search for the optimal solution of objective functions in the solution space. The construction of an appropriate model sometimes is the most critical step in the optimisation modelling process. If the model is overly simple, it will miss the insightful information on the practical problems. If it is excessively complex, it may be challenging to search for its solution. Besides, the selection of search methods is also essential, as it may determine whether the objective function is solved rapidly or slowly and whether the effective solution can be found. The above issues greatly impact the performance of traditional optimisation in handling practical problems. By contrast, learning-based optimisation will overcome these challenges and achieve optimisation modelling more concisely and efficiently.

In the ensuing sections, traditional optimisation and learning-based optimisation will be discussed in terms of the establishment and the solution of the optimisation model.

2.2 Traditional Optimisation

A natural approach to address computer vision or graphics problems is to devise an objective function $f : \hat{\mathbf{S}} \rightarrow \mathbf{R}$ which models the phenomena of interest and then uses a

suitable search method to find the best solution $\mathbf{x}_*$:

$$\mathbf{x}_* = \min_{\mathbf{x}\in\hat{\mathbf{S}}} f(\mathbf{x}) \tag{2.4}$$

$\hat{\mathbf{S}}$ is the solution set including all possible solutions for the optimisation problem. For traditional optimisation, the most common form of objective functions f is the summation of several penalty functions:

$$f(\mathbf{x}) = \sum_{j=1}^{J} \varphi\left(\mathbf{g}_j(\mathbf{x})\right) \tag{2.5}$$

$$\mathbf{g}_j(\mathbf{x}) = \mathbf{0}_d, j = 1, \cdots J, \tag{2.6}$$

where $\varphi : \mathbf{R}^d \to \mathbf{R}$ is a penalty function and $\mathbf{g}_j$ is the residual function which models tasks of interest. Residual functions typically measure the difference between the observed data and the expected value of the task model.

2.2.1 Optimisation Models in Point Cloud Registration

In point cloud registration, residual function $\mathbf{g}_j$ measures the difference between the target model $\mathbf{S}$ and the transformed source model $\mathbf{T}(\mathbf{M})$ in various metric space. Iterative closest point (ICP) algorithm [31] and its variants [56] [57] [58] [59] aim at finding the best transformation parameters to minimise the difference between two point clouds in coordinates space. [JS] [60] and [NDT] [61] model point cloud by a probability density function, then minimise the distance between these probability densities to achieve point cloud registration. [FGR] [24] utilises the Fast Point Feature Histogram (FPFH) [62] descriptor to extract features, seek correspondences, and then optimise transformation pose based on the distance between correspondences. [GMM-Tree] [23] builds trees of Gaussian mixtures to represent the feature of points, then matches points by finding the most appropriate level of geometric detail based on the feature. Compared with coordinates-based registration (ICP), feature-based registration (FGR) and density-based registration (NDT, GMM-Tree) are more robust because they register point clouds using local and global structure information rather than the single information between point pairs, reducing the impact of perturbations on registration accuracy. However, the assumption guaranteeing the feature-based registration's better performance is that the feature must be robust to various perturbations and be lower-dimensional, which will improve the registration accuracy while reducing the computation cost.

Least-square regression is the most commonly used penalty function φ due to its conceptual simplicity and high usability, the objective of which is to adjust the parameters of a model function $\mathbf{T}(\mathbf{M})$ to fit the data set $\mathbf{S}$ best. Nevertheless, least-square regression is sensitive to outliers because of the Euclidean norm formulation, which will be exposed when dealing with the registration on the coordinates space. [RPM] [63] introduces slack variables to the euclidean norm in least-square regression, which handles outliers in a statistically robust manner. [IRLS] [33] utilises M-estimation to replace the least-squares

function with a robust penalty function that is less sensitive to outliers. [TEASER] [64] employs a truncated least-squares (TLS) estimator and introduces a pre-defined constant that determines inliers and outliers to achieve robust registration. Gaussian kernel functions are also used to represent the penalty function φ, which is most common in statistical modelling of registration. Gaussian mixture models (GMM) are usually constructed in this case, and EM (Expectation-maximisation) algorithm is used to update the parameters in the mixture models. All point clouds are represented by a Gaussian mixture and the registration is cast into a clustering problem in [JRMPS] [65]. [MLMD] [66] proposes a point cloud registration algorithm that utilises GMM and a mixture decoupling technique to achieve robust and accurate 3D point cloud registration. [HGMR] [29] develops a registration algorithm using a Hierarchical Gaussian Mixture to efficiently perform point-to-model association. Compared to the least-square regression, the statistical model can provide better geometric matching. However, the typical shortcomings of this model are the high computation and slow speeds. The more points there are, the longer it takes.

Regularisation terms or other constraints commonly cooperate with penalty functions φ to enhance the robustness of algorithms, which improves the registration accuracy in the presence of various perturbations(e.g., noise, occlusion, outliers). [RPM] [63] regularises the affine transformation by penalising large values of the scale and shear components. [RPM-L2E] [41] imposes a smooth constraint on transformation by adding a regularisation term to the objective function. [MR] [67] introduces a manifold regularisation to penalise transformation. [CPD] [68] formulates a regularisation on a displacement, which comes from a prior displacement field. [GLR] [69] proposes a Graph-Laplacian regularisation to preserve the intrinsic geometry of the point cloud to be displaced. [Nonrigid-ICP] [21] utilises a stiffness term to regularise deformation and penalises the weighted difference of the transformations of neighbouring vertices under the Frobenius norm using a weighting matrix. [APSR] [22] integrates a local linear embedding (LLE)-based topology constraint along with the CPD-based regularisation to encourage the global coherent motion and the local deformation coherence of points. Most regularisation terms are used to impose motion constraints to transformation to preserve the geometric structure of points in motion, improving perturbation robustness.

Overall, the optimisation model in conventional registration methods involves designing objective functions, which refers to three factors: the choice of metric space, the schemes to devise penalty functions, and regularisation terms. Different combinations between these factors will attain various transformation parameters.

2.2.2 Search Methods for Optimisation

Except for constructing an appropriate optimisation model, the selection of search methods also determines whether the optimal transformation $\mathbf{T}^*$ can be found. The typical search methods (numerical optimisation methods) are gradient-based methods, which make extensive use of the gradients information of objective functions during iterations. Gradient-based algorithms have been broadly used for solving the optimisation models in the

applications of registration, such as gradient descent for multi-view reconstruction [70], Gauss-Newton for face alignment [71] and surface fitting [72], Levenberg-Marquardt for structure from motion [73], Conjugate Gradient for surface reconstruction [74].

$$\mathbf{x}_{t+1} = \mathbf{x}_t - [\mathbf{H}f(\mathbf{x}_t)]^{-1} \nabla f(\mathbf{x}_t) \tag{2.7}$$

Newton's algorithm (2.7) is a popular gradient-based optimisation method due to its quadratic convergence. Nevertheless, Newton's algorithm requires cost functions to be twice differentiable and the Hessian matrix $\mathbf{H}f(\mathbf{x}_t)$ needs to be positive definite. Since the Hessian matrix $\mathbf{H}f(\mathbf{x}_t)$ may not always be positive definite, the direction $[\mathbf{H}f(\mathbf{x}_t)]^{-1} \nabla f(\mathbf{x}_t)$ may not always be a descent direction, which will lead the solution for optimisation model to converge to a bad local minimum. Moreover, although the quadratic convergence rate of this method is desirable, computing the second derivative requires a lot of computation. The computational cost for obtaining the second-order gradient information for the Hessian matrix makes the method not feasible in many cases, especially for the case with a large number of parameters. Therefore, to achieve fast computation for large and complicated problems, many researchers [75] [76] have been using Quasi-Newton methods to estimate the inverse Hession matrix for finding the local maxima or minima of functions. However, the lack of precision in the Hessian estimation may lead to slow convergence. Another potential disadvantage of the Quasi-Newton method (such as the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)) is that storing the inverse Hessian approximation takes a large memory space, which could be detrimental for solving large complex tasks. Limited-memory BFGS (LBFGS) as the variant of the BFGS method, only stores a set of vectors and calculates a reduced rank approximation to the Hessian approximation, which needs much less memory to operate. Nevertheless, the amount of storage required by LBFGS depends on the parameter setting which determines the number of BFGS corrections saved.

Gradient-based searching methods are also applied to deep learning networks to search for the optimal parameters of networks, such as gradient descent with momentum [77], Nesterov accelerated gradient (NAG) [78], Adaptive Gradient Algorithm (Adagrad) [79], Adadelta [80], etc. By comparison, the most widely used for training deep learning model is Adam [81], which combines the advantages of Root Mean Square Propagation (RMSProp) [82] and Adagrad, and computes individual adaptive learning rates for different parameters. Despite the widespread popularity of Adam, some research papers [83] [84] have noted that it can fail to converge to an optimal solution under specific settings.

2.3 Learning-based Optimisation

Considering the limitations of the design of optimisation models and the selection of appropriate search methods for the optimal transformation $\mathbf{T}^*$, some works propose to learn the cost function and search path from the available training data. The former is

achieved by using machine learning techniques, and the latter involves the supervised sequential update (SSU) methods.

2.3.1 Learning Cost Functions

Several works propose to learn cost functions from existing training data through machine learning rather than manually designing objective functions based on the feature of data. [85] integrated the Support vector machines (SVMs) classifier and the optical-flow-based tracker for object tracking. The cost function maximises the SVMs classification score instead of minimising the intensity difference function between successive frames. However, the cost function works on prior information- the initial image must be transformed into an edge-based image. [86] proposed a data-driven approach to learn a metric for image alignment to reduce the effect of local minimal, where the metric learning is posed as a convex quadratic programming problem. [87] put forward a component-based discriminative approach for face alignment with initialisation. This approach regards the iterative face alignment as the process of maximising the score of a trained two-class classifier used to distinguish the correct and incorrect alignment. It is worth noting that this work trains nine classifiers for each facial component, which means that the optimisation model works based on the manually designed data features. [88] used random forest regressors trained from data to detect features of deformable models. [89] employed regression analysis and random forests with image features to learn the cost function for hand pose estimation in RGBD images, where the solution of the learned optimisation model is obtained by particle swarm search method.

A major downside of these approaches is that they need to impose the form of the cost function, e.g., [86] requires the cost function to be quadratic. In addition, these optimisation models are based on some prior information, such as the edge-based image [85] and the nine facial components [87]. It can be found these learning-based optimisation models are primarily used in the application of images, such as detecting image features and tracking image objects. They work on the image patches or image features, limiting the kinds of problems they can solve. It is not clear how to extend these models to other applications, such as point cloud registration. [90] develops a continuous data representation for point cloud registration, Support Vector–parameterised Gaussian Mixture (SVGM), which is created by training a one-class Support Vector Machine and mapping it to a Gaussian Mixture model, but the cost function of registration is still a least-square regression based on Gaussian Mixture Models.

2.3.2 Learning Search Directions

Apart from learning cost functions, recent works also propose to use learning techniques to compute the search direction of cost functions. This is done by learning a sequence of regressors to replace gradient directions of cost functions. [91], [92] regarded weak learner as functional gradient descent to update the parameter vector. These methods perform updates through learning a fixed linear sequence of weak regressors, which does

not take into account the feature of the data. [93] presented cascaded pose regression for computing the 2D pose of objects in images. This regression builds different regressors via the pose-indexed features, which depend on both the image data and the current estimate of the pose. One of the drawbacks of this method is that designing weakly invariant pose-indexed features can be quite challenging when applying this method to other applications. [94] proposed a cascaded regression approach derived from a Gauss-Newton solution to a non-linear least-squares problem, where the descent direction is replaced by a sequence of averaged Jacobian and Hessian matrices learned from data. [95] presented an "Explicit Shape Regression" approach for face alignment. This work learns a vectorial regression function to infer the whole facial shape from the image and explicitly minimise the alignment errors over the training data. [96] proposed an accurate learning-based tracking algorithm combined with object detection, where the descent direction is replaced by a linear regression function. [97] developed Iterative Error Bound Minimisation (IEBM), where a support vector regression is used to learn a sequence of linear maps to update transformation parameters for nonrigid image alignment. [98] used GentleBoost as the regression function to map image intensity to an update vector to fit a set of local feature models to an image. [9] learned a sequence of regressors matrices to update the shape parameters based on the image features per iteration.

The mentioned works either learn a fixed regression or use the features of data to gain a sequence of regressors, where the features are often available for image processing or pose estimation, such as intensity difference, Scale-invariant feature transform descriptor (SIFT), Histogram of oriented gradients (HOG), etc. Extending these methods to other problems requires designing new features. [11] [19] [12] proposed discriminative optimisation methods (DO) to address 3D registration and other computer vision tasks. The framework proposed by DO can learn the gradient directions from training data without calculating the Hessian matrix or Jacobian matrix. And experimental results illustrate the better robustness and efficiency of DO in 3D point cloud registration than other conventional registration methods.

The learning process for search direction in these methods is based on the supervised sequential update (SSU) methods, which can be formulated as follows:

$$\mathbf{x}_{t+1} = C\left(\mathbf{x}_t, \mathbf{F}_{t+1} \circ \mathbf{h}(\mathbf{x}_t)\right) \tag{2.8}$$

Where $\mathbf{x}_t \in \mathbf{R}^p$ is the estimated parameters at time t, $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$ extracts some features from the input data, and $\mathbf{F}_{t+1} : \mathbf{R}^p \to \mathbf{R}^d$ is a regressor that maps the feature $\mathbf{h}(\mathbf{x}_t)$ to an update vector, and $C : \mathbf{R}^p \times \mathbf{R}^d \to \mathbf{R}^p$ is the operator working on the parameter $\mathbf{x}_t$ and the update map $\mathbf{F}_{t+1} \circ \mathbf{h}$.

Given a training set $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{h}^i(\mathbf{x}_0^i)\right)\right\}_{i=1}^N$, including N problem instances, each instance has its ground truth parameter $\mathbf{x}_*^i$, the initial parameter $\mathbf{x}_0^i$, and the extracted features $\mathbf{h}^i(\mathbf{x}_0^i)$. The sequence of regressors $\mathbf{F}_{t+1}$ can be learned through the following formulation:

$$\mathbf{F}_{t+1} = \min_{\mathbf{F} \in \mathfrak{F}} \sum_{i=1}^{N} L\left(\mathbf{x}_*^i, C\left(\mathbf{x}_t^i, \mathbf{F} \circ \mathbf{h}^i(\mathbf{x}_t^i)\right)\right) + R(\mathbf{F}) \tag{2.9}$$

Where $L : \mathbf{R}^p \times \mathbf{R}^p \rightarrow \mathbf{R}$ is a loss function, and $R : \mathbf{R}^d \rightarrow \mathbf{R}$ is a regularisation term.

The learned regressor sequence $\mathbf{F}_{t+1}$ will be applied to estimate the transformation parameters of test data. A summary of the details of the above algorithms is provided in Table 2.1. This table shows that the mechanisms of the above algorithms are based on the supervised sequential update methods, and most of them are applied to computer vision via image features and fixed regressors. Thus, it is potential to extend these methods to point cloud registration.

2.4 Discussion

Here, the similarities and differences between the supervised sequential update (SSU) method and the neural network (NN) method are discussed and the issues in the DO method for 3D point cloud registration are pointed out.

The most apparent similarity of the SSUs and NNs is that they have a similar structure-multiple layers of regressors. However, their layers are very much different in the following aspects: *Training*: NNs can be trained "end-to-end" through backpropagation; specifically, all layers of the network can be affected by minimising the loss function at once. SSUs need to be trained layer-by-layer, and the loss function in each layer is different, but the essence is in common- make the currently estimated parameters approach to the ground truth. *Flexibility*: NNs are more flexible, and the loss function can be different for different applications, while the loss function in SSUs is limited to parameter estimation. Trained NNs can be transferred to various applications, while regression in SSUs can only be used for the same application or even the same data set. In addition, NNs are almost purely data-driven, and SSUs are more model-driven, where model means the feature. *Computational Requirement*: The number of parameters in each layer of SSUs is fixed, which is much less than that of each layer in NNS, so NNs are more computationally expensive on the same size of training data.

Although the advanced learning-based algorithm DO has shown superior results on 3D registration, some limitations remain to be solved. Firstly, DO learns the gradient directions utilising a single feature, making the learned updating map vulnerable to perturbations (e.g., noises, outliers, occlusions). Secondly, the loss functions in DO are based on the least-square regression, which ignores the influence of each component of parameter vectors on the fitting results. Rotation and translation play different roles in 3D point cloud registration actually. Namely, the contributions of the components of the transformation parameter vectors are unequally for 3D registration. Finally, the learned map $\mathbf{F}_{t+1}$ in DO extensively depends on the extracted features $\mathbf{h}(\mathbf{x}_t)$, and the dimension of $\mathbf{F}_{t+1}$ is relevant to that of the extracted feature $\mathbf{h}(\mathbf{x}_t)$, which causes that the learned map $\mathbf{F}_{t+1}$ can only estimate the transformation parameters of the same cloud. This issue makes the learning-based optimisation method DO unable to deal with multiple point cloud registration as the deep learning-based methods.

Therefore, it is necessary to address these limitations and explore more robust and efficient registration methods. The General Discriminative Optimisation (GDO) method is

Table 2.1 A summary of the learning-based search methods

Approach	Parameter x	Feature h	Composition C	Map F
CPR	Pose parameters	pose-indexed feature	Inversible composition	Random ferns
POCR	Multiple 2D image coor	Intensity diff	Summation	Linear map
ESR	Multiple 2D image coor	shape-indexed feature	Summation	boosting regression
IEBM	Transformation parameters	Intensity diff	Summation	Support Vector regression
GentleBoost	Multiple 2D image coor	Haar wavelets	Summation	GentleBoost
SDM	Pose parameters	SIFT	Summation	Linear map
DO	Pose parameters	devised feature	Summation	Linear map

proposed in Chapter.3 to improve the robustness of the learned gradient directions through the collaboration of different features; the Reweighted Discriminative Optimisation (RDO) method is put forward to improve the accuracy of estimating parameters in Chapter.4 through emphasising the different contribution of components of the parameter vector on final fitting results. Although both GDO and RDO improve the robustness and accuracy of registration by changing the way to learn the updating gradients, they have different motivations and mechanisms and they can be extended to solve different tasks. Due to GDO and RDO can only register point clouds under different perturbations, the framework called SGRTmreg is developed to achieve multiple point clouds registration in Chapter.5. Besides, the feature extraction approaches in GDO and RDO limit their abilities to deal with dense point clouds, in this case, the Graph-based Reweighted Discriminative Optimisation (GRDO) method is also explored in Chapter.5.

Chapter 3

General Discriminative Optimisation for Point Set Registration

3.1 Introduction

Existing supervised sequential update (SSU) methods [19] use a single feature $\mathbf{h}$ of data to gain a sequence of updating maps $\mathbf{F}_{t+1}$. One single feature will make the learned gradient path vulnerable significantly to the perturbations around data, thus falling into a bad stationary point, especially when the single feature lacks robustness. In this case, a General Discriminative Optimisation (GDO) is proposed to improve the robustness of SSUs on registration. GDO learns the gradient updating map $\mathbf{F}_{t+1}$ through different features of the point sets, reducing the impact of perturbations on gradient directions and, as a result, increasing the accuracy of the parameter estimation for registration. By balancing the contribution of different extracted features on the updating gradient, a sequence of gradient updating maps can be learned directly from training data sets while making the gradient path converge to the optimal point as closely as possible.

3.2 Motivation

Discriminative Optimisation (DO) updates gradient directions according to the feature of input data without calculating the Jacobian or Hessian matrix. More specifically, DO splits gradient information as the updating map $D \in \mathbf{R}^{p\times f}$ and the feature $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$, and updates the map D through approaching the current estimated parameter vector $\mathbf{x}_t$ to ground truth $\mathbf{x}_*$.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - D_{t+1}\mathbf{h}\left(\mathbf{x}_t\right) \tag{3.1}$$

$$D_{t+1} = \min_{\tilde{D}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \tilde{D}\mathbf{h}\left(\mathbf{x}_t^i\right)\right\|_2^2 + \frac{\lambda}{2}\left\|\tilde{D}\right\|_F^2. \tag{3.2}$$

Where $\|\cdot\|_F$ is the Frobenius norm, and λ is a hyperparameter. Equation.3.1 and Equation.3.2 illustrate the way to update estimated parameters and the way to learn gradient paths respectively.

Despite not calculating the Jacobian or Hessian matrix, DO still has several issues theoretically. Equation.3.2 shows that DO uses a single feature $\mathbf{h}$ of data to gain a sequence of updating maps D_{t+1}. The lack of other features of data increases the risk of perturbations around data on the update of gradient directions. In this case, GDO explores the *collaboration* of different features $\mathbf{H}_f$ to reduce the impact of perturbations on the gradient direction.

Another theoretical issue of DO is that the constraint for the convergence of DO requires each $D\mathbf{h}(\mathbf{x})$ to be strictly monotone at ground truth for all samples. Actually, not all features are able to fulfil this constraint. In other words, the convergence constraint limits the selection of feature function $\mathbf{h}$. We provide a weaker constraint for the convergence of the learning-based optimisation.

3.3 Methodology

Optimisation problems can be formulated as follows.

$$\min_{\mathbf{x}} \quad \Phi(\mathbf{x}) = \sum_{i=1}^{I} \gamma_i \frac{1}{J_i} \sum_{j_i=1}^{J_i} \varphi_i\left(\mathbf{g}_{j_i}(\mathbf{x})\right). \tag{3.3}$$

Equation.3.3 formulates the optimisation problems. Where $\varphi_i : \mathbf{R}^d \to \mathbf{R}$ is a penalty function, $\mathbf{g}_{j_i} : \mathbf{R}^p \to \mathbf{R}^d$ models the problem of interest, J_i is the number of $\mathbf{g}_{j_i}$ corresponding to the penalty function φ_i, γ_i is the coefficient of penalty function φ_i. The following framework shows how to use the feature information of data to mimic the gradient descent of unknown functions Φ.

For simplicity, just two different unknown penalty functions are considered, $I = 2$ and assume $\Phi(\mathbf{x})$ is differentiable.

$$\Phi(\mathbf{x}) = \frac{\gamma_1}{J_1} \sum_{j_1=1}^{J_1} \varphi_1\left(\mathbf{g}_{j_1}(\mathbf{x})\right) + \frac{\gamma_2}{J_2} \sum_{j_2=1}^{J_2} \varphi_2\left(\mathbf{g}_{j_2}(\mathbf{x})\right). \tag{3.4}$$

Let us observe its derivative:

$$\begin{aligned}
\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}} &= \sum_{i=1}^{2} \gamma_i \frac{1}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}}\right]^{\mathrm{T}} \left[\frac{\partial \varphi_i}{\partial \mathbf{g}_{j_i}}\right] \\
&= \sum_{i=1}^{2} \gamma_i \frac{1}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}}\right]^{\mathrm{T}} \phi_i\left(\mathbf{g}_{j_i}\right) \\
&= \sum_{k=1}^{d} \sum_{i=1}^{2} \gamma_i \frac{1}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}}\right]^{\mathrm{T}}_{k,:} \left[\phi_i\left(\mathbf{g}_{j_i}\right)\right]_k .
\end{aligned} \tag{3.5}$$

Where $\mathbf{g}_{j_i}(\mathbf{x})$ is denoted as $\mathbf{g}_{j_i}$ and $\varphi_i\left(\mathbf{g}_{j_i}(\mathbf{x})\right)$ is denoted as φ_i to reduce notation clutter. $[Y]_k$ is the k_{th} row of Y ,and $[y]_k$ means the k_{th} element of y. Y means in general any matrix. y generally refers to any vector. Then the derivative of φ_i is replaced with a generic function ϕ_i.

For simplicity, the l_{th} element of $\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}}$ is denoted as $[\Delta \mathbf{x}]_l$. Then Equation (3.5) can be rewritten as the following format:

$$
\begin{aligned}
[\Delta \mathbf{x}]_l &= \sum_{k=1}^{d} \sum_{i=1}^{2} \gamma_i \frac{1}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}} \right]_{k,l} \left[\phi_i \left(\mathbf{g}_{j_i} \right) \right]_k \\
&= \sum_{k=1}^{d} \sum_{i=1}^{2} \frac{\gamma_i}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}} \right]_{k,l} \int_{\mathbf{R}^d} \left[\phi_i (\mathbf{v}) \right]_k \delta \left(\mathbf{v} - \mathbf{g}_{j_i} \right) d\mathbf{v} \\
&= \sum_{k=1}^{d} \sum_{i=1}^{2} \int_{\mathbf{R}^d} \left[\phi_i (\mathbf{v}) \right]_k \frac{\gamma_i}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_i} \right) d\mathbf{v} \\
&= \sum_{k=1}^{d} \int_{\mathbf{R}^d} \left[\phi_1 (\mathbf{v}) \right]_k \frac{\gamma_1}{J_1} \sum_{j_1=1}^{J_1} \left[\frac{\partial \mathbf{g}_{j_1}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_1} \right) \\
&+ \left[\phi_2 (\mathbf{v}) \right]_k \frac{\gamma_2}{J_2} \sum_{j_1=2}^{J_2} \left[\frac{\partial \mathbf{g}_{j_2}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_2} \right) d\mathbf{v} \\
&= \sum_{k=1}^{d} \int_{\mathbf{R}^d} \left[\phi_1 (\mathbf{x}) \quad \phi_2 (\mathbf{x}) \right]_k \times \begin{bmatrix} \frac{\gamma_1}{J_1} \sum_{j_1=1}^{J_1} \left[\frac{\partial \mathbf{g}_{j_1}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_1} \right) \\ \frac{\gamma_2}{J_2} \sum_{j_2=1}^{J_2} \left[\frac{\partial \mathbf{g}_{j_2}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_2} \right) \end{bmatrix} d\mathbf{v}
\end{aligned}
\tag{3.6}
$$

Where $\delta (\mathbf{x})$ is the Dirac function. Then

$$
\left[\frac{\partial \Phi (\mathbf{x})}{\partial \mathbf{x}} \right]_l = \sum_{k=1}^{d} \int_{\mathbf{R}^d} D (\mathbf{v}, k) \, \mathbf{H}_f (\mathbf{v}, k, l; x) \, d\mathbf{v}. \tag{3.7}
$$

$$
D (\mathbf{v}, k) = \left[\phi_1 (\mathbf{x}) \quad \phi_2 (\mathbf{x}) \right]_k. \tag{3.8}
$$

$$
\mathbf{H}_f (\mathbf{v}, k, l; \mathbf{x}) = \begin{bmatrix} \gamma_1 \mathbf{h}_1 \\ \gamma_2 \mathbf{h}_2 \end{bmatrix}. \tag{3.9}
$$

Where $\mathbf{h}_i$ is the representation of the i_{th} feature of data, and the coefficient γ_i means the contribution of each feature on updating direction.

$$
\mathbf{h}_i (\mathbf{v}, k, l; \mathbf{x}) = \frac{1}{J_i} \sum_{j_i=1}^{J_i} \left[\frac{\partial \mathbf{g}_{j_i}}{\partial \mathbf{x}} \right]_{k,l} \delta \left(\mathbf{v} - \mathbf{g}_{j_i} \right). \tag{3.10}
$$

$$
\gamma_i = \frac{Tr \left(\mathbf{Cov} \left(\mathbf{h}_i \right) \right)}{\sum_{i=1}^{I} Tr \left(\mathbf{Cov} \left(\mathbf{h}_i \right) \right)}. \tag{3.11}
$$

$Tr \left(\mathbf{Cov} \left(\mathbf{h}_i \right) \right)$ is the trace of the covariance matrix of $\mathbf{h}_i$.

3.4 GDO Framework

3.4.1 Learning for GDO

Given a set of training data $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{H}_f(\mathbf{x}_0^i)\right)\right\}_{i=1}^N$, including N problem instances, each instance has its ground truth parameter $\mathbf{x}_*^i$, the initial parameter $\mathbf{x}_0^i$, and the extracted features $\mathbf{H}_f(\mathbf{x}_0^i)$. For simplicity, $\mathbf{H}_f(\mathbf{x}_t^i)$ is denoted as $\mathbf{H}_{ft}^i$ to represent the feature of the i-th sample at the t-th iteration. The goal of GDO is to learn a sequence of maps D_{t+1} that update $\mathbf{x}_0^i$ to $\mathbf{x}_*^i$.

$$D_{t+1} = \min_{\tilde{D}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{x}_*^i - \mathbf{x}_t^i + \tilde{D}\mathbf{H}_{ft}^i \right\|_2^2 + \frac{\lambda}{2} \left\| \tilde{D} \right\|_F^2. \tag{3.12}$$

Where $\|\cdot\|_F$ is the Frobenius norm, and λ is a hyperparameter.

The initial training data $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{H}_{f0}^i\right)\right\}_{i=1}^N$ is applied to Equation (3.12) to learn map D_1 at first. Then, D_1 will be applied to Equation (2.1) to get the current estimation $\mathbf{x}_1$. At each step, a new parameter vector can be created by recursively applying the update rule in Equation (2.1). The learning process is repeated until some termination criteria are met, such as until the error does not decrease much or the maximum number of iterations T is reached. The pseudocode for training GDO is shown in Alg.1.

Algorithm 1 Training a sequence of update maps

Input $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{H}_{f0}^i\right)\right\}_{i=1}^N$, T, λ
Output $\{D_t\}_{t=1}^T$

```
for t = 0 to T − 1 do
    Compute D_{t+1} with (3.12)
    for i = 1 to N do
        Update x_{t+1}^i := x_t^i − D_{t+1} H_{ft}^i
    end for
end for
```

3.4.2 Convergence Analysis of GDO

Theorem 3.4.1 (Convergence of GDO's training error)
Given a training set $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{H}_{f0}^i\right)\right\}_{i=1}^N$, if there exists a linear map $\hat{D} \in \mathbf{R}^{p \times f}$ where $\hat{D}\mathbf{H}_f$ meets the condition $\sum_{i=1}^N \left(\mathbf{x}_*^i - \mathbf{x}_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_{ft}^i > 0$ at $\mathbf{x}_*^i$ for all i, and if there exists an i where $\mathbf{x}_t^i \neq \mathbf{x}_*^i$, then the update rule:

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i - D_t \mathbf{H}_{ft}^i. \tag{3.13}$$

with $D_t \subset \mathbf{R}^{p\times f}$ obtained from Equation (3.12), guarantees that the training error strictly decreases in each iteration:

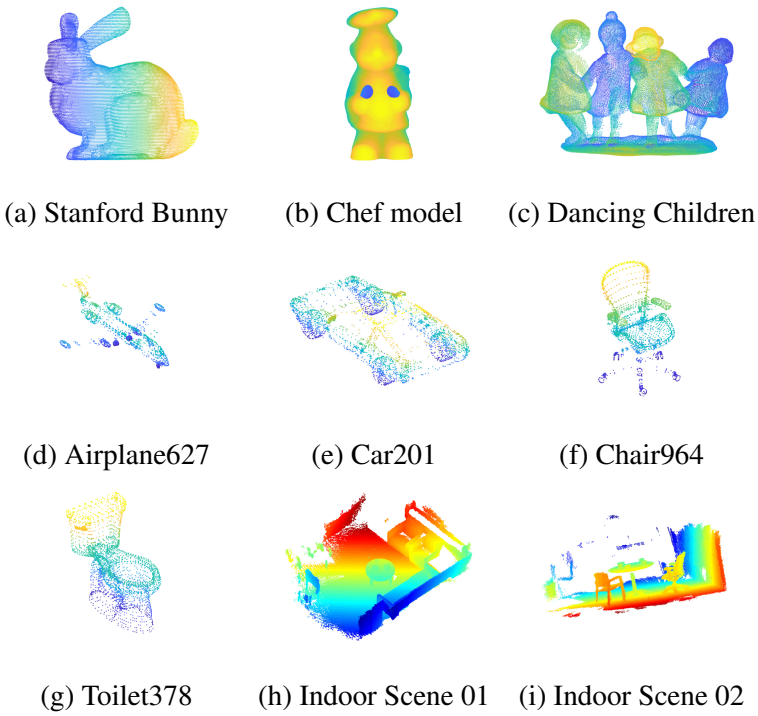
$$\sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_{t+1}^i\right\|_2^2 < \sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2. \tag{3.14}$$

If $\hat{D}\mathbf{H}_f$ is strongly monotone, and if there exist $H > 0$, $M > 0$ such that $\left\|\hat{D}\mathbf{H}_f^i\right\|_2^2 \leq H + M\left\|\mathbf{x}_*^i - \mathbf{x}^i\right\|_2^2$ for all i, then the training error converges to zero.

Thm.3.4.1 says that for all instances, if $\hat{D}\mathbf{H}_f$ meets the condition $\sum_{i=1}^{N}\left(\mathbf{x}_*^i - \mathbf{x}_t^i\right)^{\mathrm{T}}\hat{D}\mathbf{H}_{ft}^i > 0$, then the average training error will decrease in each iteration; if $\hat{D}\mathbf{H}_f$ is strongly monotone at $\mathbf{x}_*^i$, the average training error will converge to zero. Note that $\mathbf{H}_f$ can be not only a single function but also a combination of different functions of $\mathbf{x}^i$. DO also presents a similar convergence result for a update rule, but it requires $\hat{D}\mathbf{H}_{ft}^i$to be strictly monotone at $\mathbf{x}_*^i$ for all i. Besides, different from the single feature $\mathbf{h}$ in DO, as the combination composed of several feature functions, $\mathbf{H}_f$ takes into account more features of data. The detailed proof for convergence is provided in Appendix A.

3.5 Experimentation

This section describes how to apply GDO to the 3D point set registration with various perturbations. We compare GDO with other classical registration methods on various data sets. The data sets are shown as Figure.3.1.



(a) Stanford Bunny (b) Chef model (c) Dancing Children

(d) Airplane627 (e) Car201 (f) Chair964

(g) Toilet378 (h) Indoor Scene 01 (i) Indoor Scene 02

Fig. 3.1 Experimental data sets

3.5.1 3D Point Set Registration

Let $\{\mathbf{M},\mathbf{S}\}$ be two point sets in a finite-dimensional real vector space $\mathbf{R}^3$, which contains $\mathbf{N}_m$ and $\mathbf{N}_s$ points, respectively. Our goal is to find a rigid transformation T to be applied to scene set $\mathbf{S}$ such that the difference between $\mathbf{S}$ and model set $\mathbf{M}$ is minimised. The transformation matrix T is posed as the Lie algebra $x \in \mathbf{R}^6$ in our optimisation problem.

Feature for Registration

The feature $\mathbf{H}_f$ for registration is combined by two different features: the coordinates-based feature $[\mathbf{h}(x;\mathbf{S})]^c$ and the density-based feature $[\mathbf{h}(x;\mathbf{S})]^d$.

We use the feature extraction method in [11] to extract the features $[\mathbf{h}(x;\mathbf{S})]^c$ and $[\mathbf{h}(x;\mathbf{S})]^d$, where $\mathbf{h}$ is devised to be a histogram indicating the weights of scene points on the 'front' and the 'back' sides of each model point. As shown in Figure.3.2.

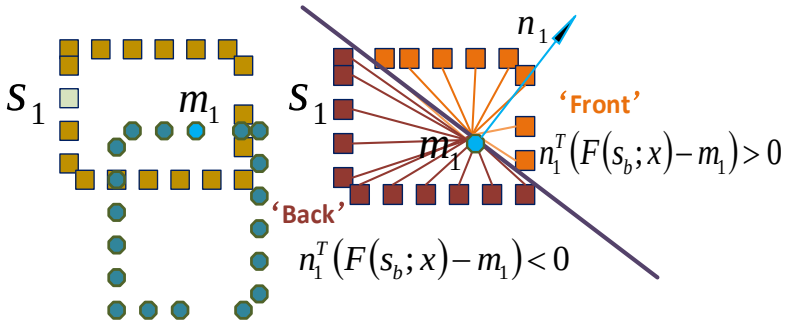


Fig. 3.2 The positional relationship between scene points (square) $\mathbf{s}_1$ and model point (hexagon) $\mathbf{m}_1$.

$$\mathbf{S}_a^+ = \left\{\mathbf{s}_b : \mathbf{n}_a^{\mathrm{T}}\left(\mathbf{F}\left(\mathbf{s}_b;x\right)-\mathbf{m}_a\right) > 0\right\} \tag{3.15}$$

$\mathbf{S}_a^+$ indicates the set of scene points on the 'front' of model point $\mathbf{m}_a$, and $\mathbf{S}_a^-$ contains the remaining scene points.; $\mathbf{n}_a \in \mathbf{R}^3$ is the normal vector of the model point $\mathbf{m}_a$; $\mathbf{F}(\mathbf{s}_b;x)$ is the function that applies rigid transformation with parameter x to scene point $\mathbf{s}_b$.

Then the feature $[\mathbf{h}(x;\mathbf{S})]^c$ can be calculated through the following formulas:

$$[\mathbf{h}(x;\mathbf{S})]_{a^+}^c = \frac{1}{z}\sum_{\mathbf{s}_b\in\mathbf{S}_a^+}\exp\left(\frac{-1}{\hat{\sigma}^2}\left\|\mathbf{F}(\mathbf{s}_b;x)-\mathbf{m}_a\right\|^2\right). \tag{3.16}$$

$$[\mathbf{h}(x;\mathbf{S})]_{a^-}^c = \frac{1}{z}\sum_{\mathbf{s}_b\in\mathbf{S}_a^-}\exp\left(\frac{-1}{\hat{\sigma}^2}\left\|\mathbf{F}(\mathbf{s}_b;x)-\mathbf{m}_a\right\|^2\right). \tag{3.17}$$

Where z normalises $\mathbf{h}$ to sum to 1, and $\hat{\sigma}$ controls the width of the exp function. The design of the feature $[\mathbf{h}(x;\mathbf{S})]^d$ can be divided into two stages. The first stage is to calculate the

probability of measuring each point of **S** in the boxes of **M**, and the probability of each point of **M** in the boxes of **S**, as shown in Figure.3.3. The second stage is to apply the calculated probability to (3.16),(3.17) to extract the density feature $[\mathbf{h}(x;\mathbf{S})]^d$.

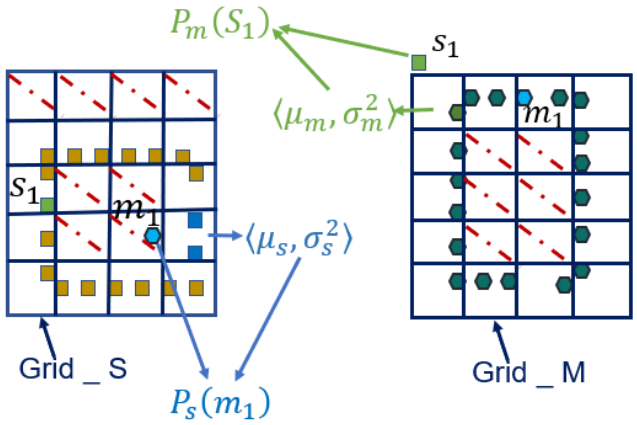


Fig. 3.3 The first stage for designing the density feature $[\mathbf{h}(x;\mathbf{S})]^d$.

Figure.3.3 shows the first stage for designing the density feature $[\mathbf{h}(x;\mathbf{S})]^d$. The Grid_S represents the grids around the model **S**. The Grid_M represents the grids around the model **M**. The grid marked by the red dotted line represents the grid where is no point, which will be removed when calculating the mean μ and covariance σ^2.

The probability of measuring each point of **S** in the boxes of **M**, $\mathbf{P}_m(s_j)$, can be calculated as follows, and the probability of measuring the points of **M** in the boxes of **S**, $\mathbf{P}_s(m_a)$, can be calculated in a similar way.

1. The 3D space around the point set **M** is subdivided regularly into boxes with constant size (e.g. the Grid_S, Grid_M in Figure.3.3).

2. For each box, the following is done:

   - Collect all 3D points $m_{i=1,2,\cdots,N_m}$ in **M** contained in this box. If there is no point in a box, the box will be removed (e.g. the grids marked by the red dotted line in Figure.3.3).
   - Calculate the mean

$$\mu_m = \frac{1}{N_m}\sum_{i=1}^{N_m} m_i.$$

   - Calculate the covariance matrix

$$\sigma_m^2 = \frac{1}{N_m}\sum_{i=1}^{N_m} (m_i - \mu_m)(m_i - \mu_m)^{\mathrm{T}}.$$

3. The probability of measuring each point s_j of $\mathbf{S}$ in this box is now modeled by the normal distribution $\mathscr{N}\left(\mu_m, \sigma_m^2\right)$.

$$\mathbf{P}_m\left(s_j\right) \sim \exp\left(\frac{-(s_j-\mu_m)^{\mathrm{T}}(s_j-\mu_m)}{2\sigma}\right).$$

$$[\mathbf{h}(x;\mathbf{S})]_{a^+}^d = \frac{1}{z}\sum_{\mathbf{s}_b\in\mathbf{S}_a^+}\exp\left(\frac{-1}{\hat{\sigma}^2}\left\|\mathbf{P}_m(\mathbf{F}(\mathbf{s}_b;x))-\mathbf{P}_s(\mathbf{m}_a)\right\|^2\right). \tag{3.18}$$

$$[\mathbf{h}(x;\mathbf{S})]_{a^-}^d = \frac{1}{z}\sum_{\mathbf{s}_b\in\mathbf{S}_a^-}\exp\left(\frac{-1}{\hat{\sigma}^2}\left\|\mathbf{P}_m(\mathbf{F}(\mathbf{s}_b;x))-\mathbf{P}_s(\mathbf{m}_a)\right\|^2\right). \tag{3.19}$$

The final feature $\mathbf{H}_f$ can be posed as:

$$\mathbf{H}_f = \begin{bmatrix}\gamma_1[\mathbf{h}\left(x;\mathbf{S}\right)]^c\\ \gamma_2[\mathbf{h}\left(x;\mathbf{S}\right)]^d\end{bmatrix}. \tag{3.20}$$

We get the coefficients γ_1, γ_2 using (3.11), which represent the contributions of features on updating gradient direction.

3.5.2 GDO Training Settings

The parameters in the GDO training process are the same as those in the code provided in the Github of DO [11] for the comparison experiments on the synthetic data sets. We normalise a given model shape $\mathbf{M}$ to $[-1,1]^3$ and uniformly sample from $\mathbf{M}$ with the replacement of 400 to 700 points to generate a scene model. Then we apply the following perturbations to the scene model: *(i) Rotation and translation:* The rotation is within 60 degrees and the translation is in $[-0.3,0.3]^3$, which represents the ground truth $\mathbf{x}_*$; *(ii) Noise and Outliers:* Gaussian noise with the standard deviation 0.05 is added to the scene model. 0 to 300 points within $[-1.5,1.5]^3$ are added as the sparse outliers. Besides, a Gaussian ball of 0 to 200 points with the standard deviation of 0.1 to 0.25 is used to simulate the structured outliers; *(iii) incomplete shape:* We remove 40% to 90% points from the scene model to simulate occlusions, the detailed removing approach can be found in [11]. For all experiments, we generated 30000 training samples, set up iterations $T=30$ and set λ as 2×10^{-4}, β^2 as 0.03, and the initial transformation $\mathbf{x}_0$ is $\mathbf{0}^6$. For the second feature $[\mathbf{h}(x;\mathbf{S})]^d$, we build the uniform grid in the range $[-2,2]$ with 81 points in each dimension.

For the comparison experiments on the Modelnet40 data set, we design three modes for GDO training. (i) $mode_1$: The rotation is within 45 degrees and the translations is in $[-0.5,0.5]^3$; (ii) $mode_2$: The rotation is within 90 degrees and the translations is in $[-0.5,0.5]^3$; (iii) $mode_3$: The rotation is within 90 degrees, the translations is in $[-0.5,0.5]^3$ and Gaussian noise with the standard deviation 0.05 is also applied. The first two modes aim to compare the registration of all methods in terms of varying degrees of rotation, named *single-class training*. The latter is to compare the performance of different

methods on the registration with multiple perturbations, named *multi-class training*. We generated 30000 training samples for all modes, and the training sample will be normalised to $[-1,1]^3$ without downsampling. The number of points of all samples is 5120.

3.5.3 Performance Metrics

Baselines. We compared GDO with the advanced learning-based approach DO [11], two point-based approaches (ICP and IRLS), two density-based approaches (CPD and NDT) and the feature-based approach (FGR).
We used the successful registration rate, average MSE and computation time as performance metrics.
Successful Registration Rate. A registration is successful when the mean ℓ_2 is less than 0.05 of the model's largest dimension.
Average MSE. It is worth noting that the MSE is the mean ℓ_2 error between the model and scene sets, and the Average MSE is the average for MSE for all test sets.

In order to make the experimental results more clear, we use log_{10} MSE and log_{10} computing time to describe the accuracy and efficiency of the registration of all registration methods on the ModelNet40 data set.

3.5.4 Parameters Settings

The maximum number of iterations of all registration methods was set to 30. For *DO* and *GDO*, we set λ as 2×10^{-4}, β^2 as 0.03. The value of the tolerance of absolute difference between current estimation and ground truth in iterations is 1e-4; For *ICP*, the tolerance of absolute difference in translation and rotation is 0.01 and 0.5 respectively; For *IRLS*, we used *Huber criterion function* as the regression function, the remaining parameters were set as the same as the setting of *ICP*. For *CPD*, the type of transformation is set to *rigid*, and the expected percentage of outliers with respect to a normal distribution is 0.1, the tolerance value is the same as that in DO. For *NDT*, the value of the expected percentage of outliers is set to 0.55, and the tolerance value is set as the same as that in ICP; For *FGR*, the value of the division factor used for graduated non-convexity is 1.4, the maximum correspondence distance is 0.025, the value of the similarity measure used for tuples of feature points is 0.95, the value of the maximum tuple numbers for trading off between speed and accuracy is set to 1000.

For *BCPD*, the expected percentage of outliers is set to 0.1, the parameter in the Gaussian kernel is 2.0 and the expected length of the displacement vector is 400. All deep-learning-based registration networks are trained on an Nvidia Geforce 2080Ti GPU with 12G memory. For *PCRNet*, the kernel sizes are 64, 64, 64, 128, 1024, 1024, 512, 512, 256 and 7. The iteration for rotation and translation is set to 8. Adam optimiser with an initial learning rate of 0,1, 300 epochs and a batch size of 32 is used for the training process. For *PointnetLK*, the kernel sizes are 64, 64, 64, 128, 1024. The maximum iteration for rotation and translation is set to 30. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 10 is used for the training process. For *DCP*, the kernel

sizes are 64, 64, 128, 256, 512, 1024, 256, 128, 64, 32 and 7. The iteration for rotation and translation is set to 1. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 32 is used for the training process. For *ICPMCC*, the error threshold is set to 10^{-7}, the iteration number is 30, and the number of nearest points for calculating normal vectors is set to 10.Please note that the values of these parameters are the default values set in their official codes.

3.5.5 Registration Experiments

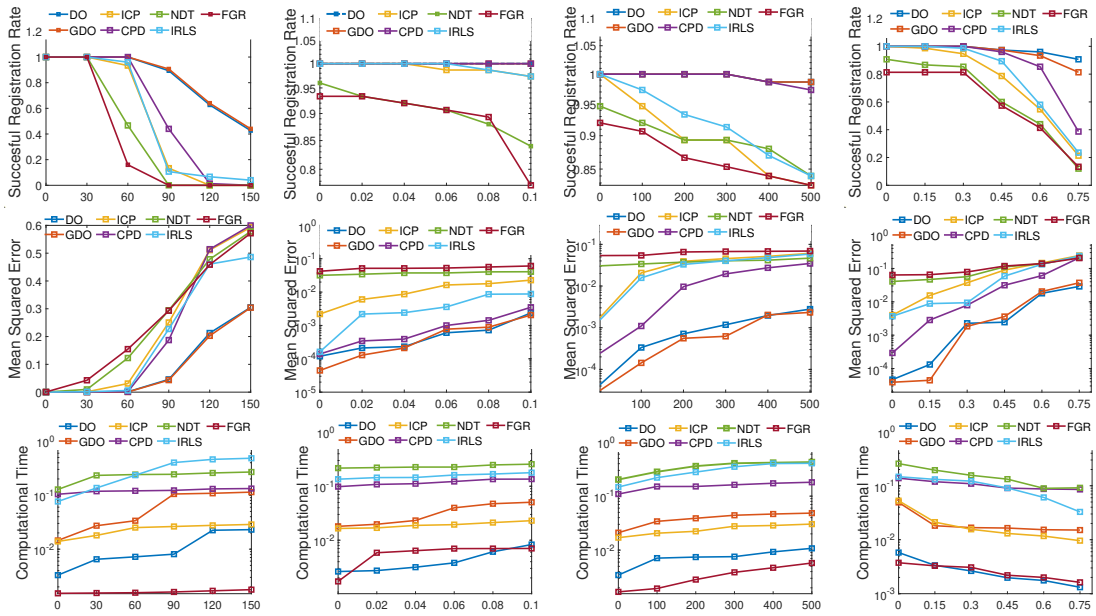
We have used the Stanford Bunny model [13], Chef model [14], Dancing Children, Indoor Scene [15] as the data sets for experiments Figure.3.1. Dancing Children are available at the AIM@SHAPE shape repository http://visionair.ge.imati.cnr.it/ontologies/shapes/. The model set **M** is generated by using the grid average downsample method in MATLAB to select 477 points from the original model.

The performance of algorithms are evaluated by comparing the evaluation metrics in the case of various perturbations: *(1) rotation:* We compare the performance metrics when the initial angle is 0°, 30°, 60°, 90°, 120° and 150°[default=0°to60°]; *(2) noise:* The standard deviation of the noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]; *(3) outliers:* We set the number of outliers to 0, 100, 200, 300, 400 and 500 respectively [default=0]; *(4) incomplete ratio:* The ratio of incomplete scene shape is set to 0, 0.15, 0.3, 0.45, 0.6 and 0.75 [default=0]. The random translation of all generated scenes is within $[-0.3, 0.3]^3$. When one parameter is changed, the values of other parameters are fixed to default values. In addition, the scene points are sampled from the original model, not from **M**. We will test 750 testing samples in each variable setting. It is noteworthy that the training samples are generated by adding various perturbations to the model **M** and assigning random parameters for the translation and rotation of the model **M**. The testing samples are generated similarly, but the degree of perturbation and the parameters for transformation are different, and the down-sampled model is the original model instead of the model **M**.

We also conduct comparative registration experiments on the ModelNet40 data set [16] with traditional methods (BCPD [99], FPFH-ICP [62] and ICPMCC [100]) and other advanced deep-learning-based registration methods, such as PCRNet [101], PointnetLK [17], and DCP [102] (as shown in $(d) \sim (g)$ of Figure.3.1). There are three kinds of comparison settings corresponding to the training modes in 3.5.2: for $mode_1$: the initial angle is 0°, 15°, 30°, 45°, 60° and 75°; for $mode_2$: the initial angle is 0°, 30°, 60°, 90°, 120° and 150°; for $mode_3$: the initial angle is 0°, 30°, 60°, 90°, 120° and 150° [default=0° to 90°] and the standard deviation of Gaussian noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]. It is worth noting that when we change one parameter, the values of other parameters are fixed to the default value. We will test 100 test samples in each variable setting.

Experimental Results

Registration results. Figure.3.4 and Figure.3.5 show the 3D registration results on Bunny model and Chef model with various perturbations. The top is the Successful Registration Rate (SRR). The middle is the Average MSE (AMSE). The bottom is the Computation Time. It can be seen that in the presence of arbitrary perturbation, learning-based registration algorithms (DO, GDO) can achieve more accurate registration results than the traditional registration methods (ICP, CPD, NDT, IRLS, FGR). Compared with DO, the performance of GDO is slightly better than DO. However, GDO is more time-consuming, the reason for which is that the second feature $[\mathbf{h}(x;\mathbf{S})]^d$ calculates the density probability of each point in point sets, which involves the search of the closest box. Also, the calculation way of the second feature determines that the running time of GDO and the size of the point set are positively correlated.



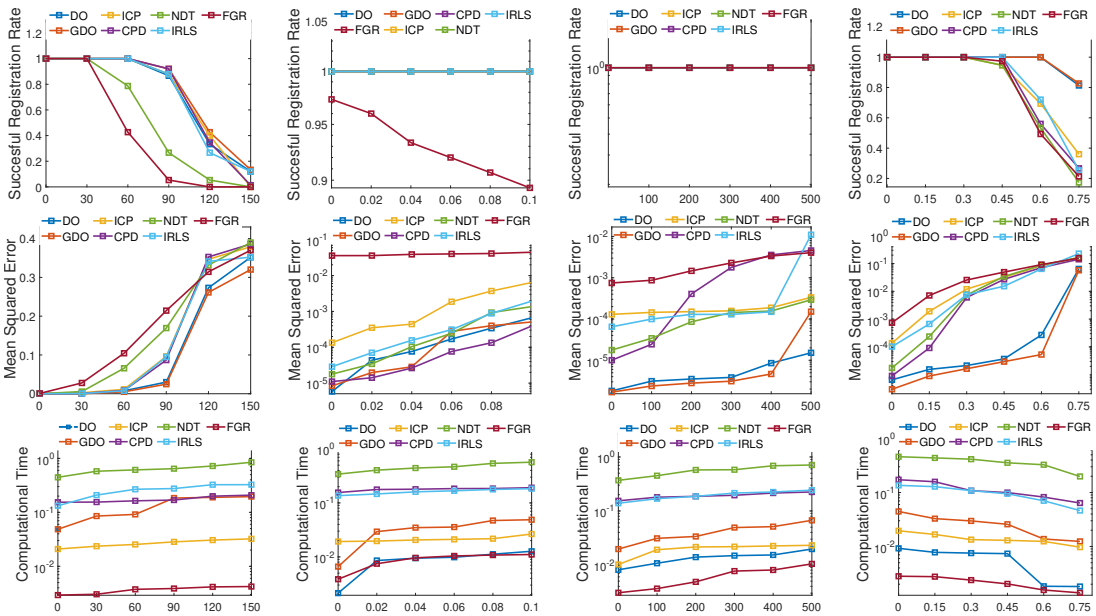
(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

Fig. 3.4 Results of 3D registration with Bunny model under different perturbations.

Table.3.1 shows the quantitative results of the registration on the Bunny model. B means the baseline method - Discriminative Optimisation method (DO); P is the proposed method in this chapter - General Discriminative Optimisation method (GDO); C is the outstanding conventional method in this registration experiments - Coherent Point Drift method (CPD). We set the Successful Rate of DO, Mean Square Error of DO, and the Computation Time of DO as the references of comparison. The value of the Successful Rate is higher, the stability of the method is higher; the value of the Mean Square Error is lower, the registration accuracy is higher; the shorter the computation time, the real-time capability is better. The Bold in this table shows that the stability and registration accuracy of GDO is better than DO and CPD. The computing time of GDO is almost six times that of DO.

Table 3.1 The quantitative results of the registration on the Bunny model (B-Baseline DO; P-Proposed GDO; C-Conventional method CPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.00**	1.00	1.00	**0.90**	20.0	1.00	6.00	80.0
(R)90	1.00	**1.02**	0.45	1.00	**0.67**	66.7	1.00	17.30	115.38
(R)120	1.00	**1.03**	0.06	1.00	**0.95**	2.15	1.00	4.50	35.00
(R)150	1.00	**1.02**	0.00	1.00	1.00	1.67	1.00	4.50	40.00
(N)0.04	1.00	1.00	1.00	1.00	**0.80**	2.00	1.00	4.00	20.00
(N)0.06	1.00	1.00	1.00	1.00	1.50	2.50	1.00	6.67	20.00
(N)0.08	1.00	1.00	1.00	1.00	**0.75**	2.25	1.00	6.00	28.57
(N)0.1	1.00	1.00	1.00	1.00	**0.90**	2.00	1.00	4.67	22.22
(O)200	1.00	1.00	1.00	1.00	**0.71**	12.9	1.00	6.67	33.33
(O)300	1.00	1.00	1.00	1.00	**0.56**	22.2	1.00	1.17	36.67
(O)400	1.00	1.00	1.00	1.00	1.00	40.0	1.00	5.63	28.75
(O)500	1.00	1.00	0.97	1.00	**0.33**	16.7	1.00	5.00	24.00
(I)0.30	1.00	1.00	1.00	1.00	**0.33**	2.67	1.00	6.00	20.00
(I)0.45	1.00	1.00	0.95	1.00	1.67	13.30	1.00	7.00	20.00
(I)0.60	1.00	0.98	0.94	1.00	1.12	6.25	1.00	9.00	23.33
(I)0.75	1.00	0.96	0.44	1.00	1.50	4.50	1.00	13.25	25.00



(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

Fig. 3.5 Results of 3D registration with Chef model under different perturbations

Table.3.2 shows the quantitative results of the registration on the Chef model. C is the outstanding conventional method in this registration experiments - Iteratively Reweighted Least Squares method (IRLS). The Bold in this table shows that the stability and registration accuracy of GDO is better than DO and IRLS. The registration accuracy of GDO is 4.7% higher than that of DO.

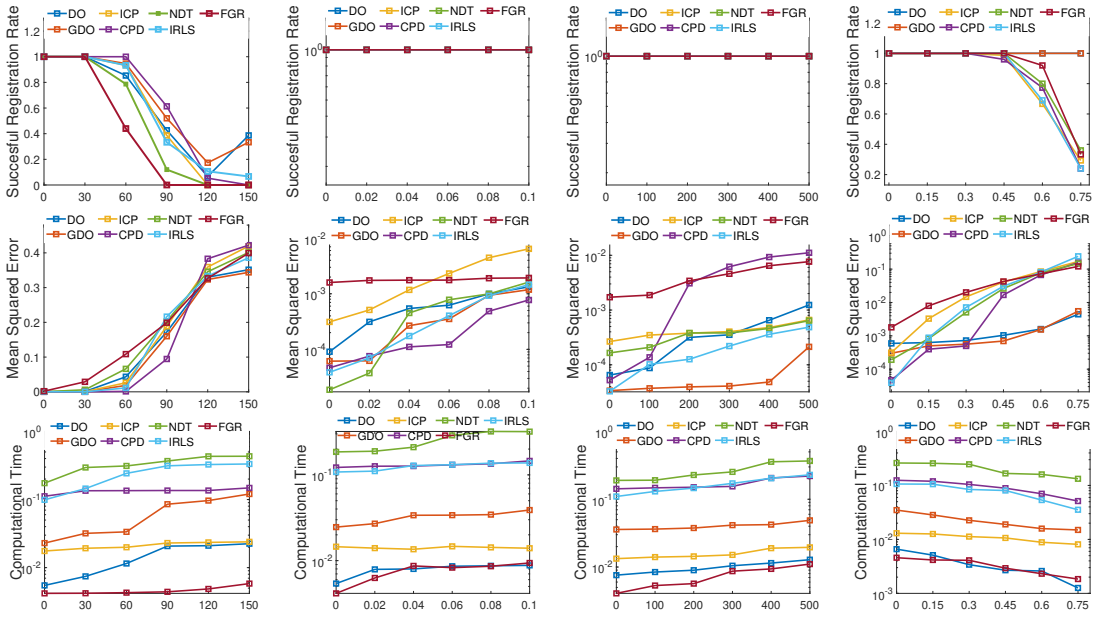
Table 3.2 The quantitative results of the registration on the Chef model (B-Baseline DO; P-Proposed GDO; C-Conventional method IRLS)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.00**	1.00	1.00	**0.50**	1.00	1.00	5.00	37.50
(R)90	1.00	**1.08**	1.08	1.00	**0.86**	2.28	1.00	10.97	48.78
(R)120	1.00	**1.25**	1.05	1.00	**0.93**	1.22	1.00	3.00	16.67
(R)150	1.00	**1.11**	0.11	1.00	**0.94**	1.00	1.00	3.00	20.00
(N)0.04	1.00	1.00	1.00	1.00	**0.60**	1.80	1.00	4.44	11.11
(N)0.06	1.00	1.00	1.00	1.00	2.50	2.50	1.00	5.44	13.04
(N)0.08	1.00	1.00	1.00	1.00	1.05	2.27	1.00	5.16	12.90
(N)0.1	1.00	1.00	1.00	1.00	**0.83**	3.33	1.00	5.00	12.00
(O)200	1.00	1.00	1.00	1.00	**0.70**	13.30	1.00	4.44	14.44
(O)300	1.00	1.00	1.00	1.00	**0.80**	13.10	1.00	5.43	16.30
(O)400	1.00	1.00	1.00	1.00	**0.90**	11.40	1.00	5.16	17.20
(O)500	1.00	1.00	1.00	1.00	8.00	100.00	1.00	6.00	16.00
(I)0.30	1.00	1.00	1.00	1.00	**0.94**	941.17	1.00	5.56	11.11
(I)0.45	1.00	1.00	1.00	1.00	**0.86**	476.19	1.00	4.70	10.59
(I)0.60	1.00	1.00	0.75	1.00	**0.20**	600.00	1.00	7.50	15.00
(I)0.75	1.00	**1.01**	0.35	1.00	**0.93**	6.67	1.00	5.56	11.11

Figure.3.6 shows the registration results on the Dancing Children model. The trend and distribution of the running time of all algorithms on the Dancing Children model are the same as that on the Bunny or Chef models. GDO is more capable when dealing with the complex model than registering simple models (Bunny, Chef), which can be illustrated by the Mean Square Error criteria.

Table.3.3 shows the quantitative results of the registration on the Dancing Children model. C is the outstanding conventional method in this registration experiment - the Coherent Point Drift method (CPD). We set the Successful Rate of DO, Mean Square Error of DO, and the Computation Time of DO as the references of comparison. The Bold in this table shows that the stability and registration accuracy of GDO is better than DO and CPD. The registration accuracy of GDO is 3.86% higher than that of DO. The Successful Rate of GDO is 0.86% higher than that of DO.

Figure.3.7 and Figure.3.8 show the results of 3D registration on Indoor Scenes. The performances of NDT and GDO are prominent when registering real scene models.

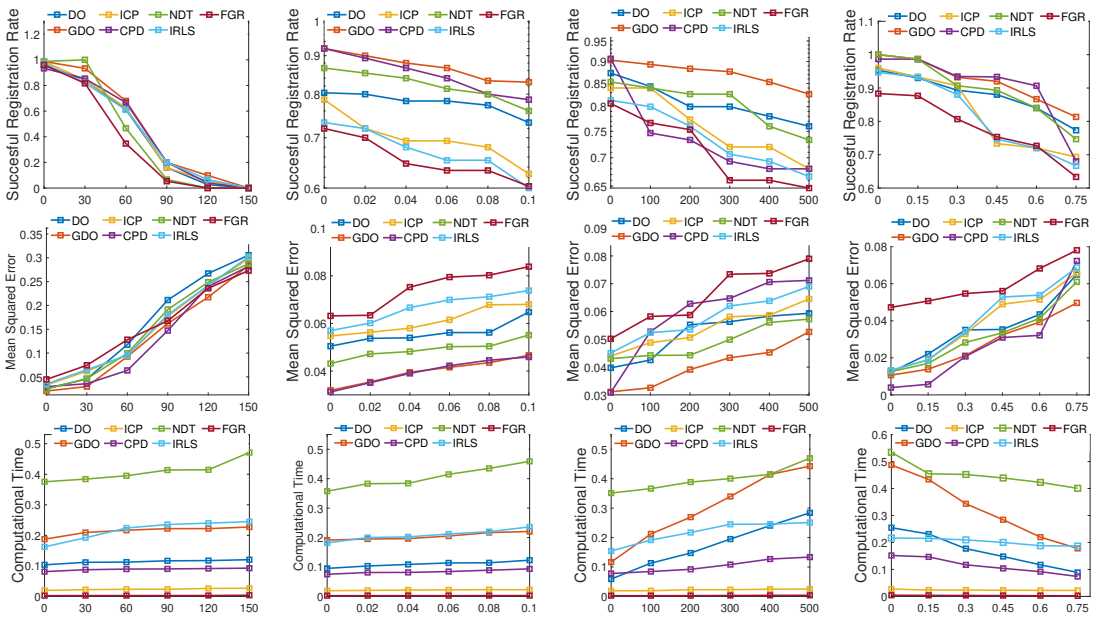


(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

Fig. 3.6 Results of 3D registration with Dancing Children model under different perturbations

Table 3.3 The quantitative results of the registration on the Dancing Children model (B-Baseline DO; P-Proposed GDO; C-Conventional method CPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.17**	1.22	1.00	**0.50**	0.00	1.00	4.00	10.00
(R)90	1.00	**1.33**	1.70	1.00	**0.88**	0.47	1.00	3.08	3.85
(R)120	1.00	0.95	0.71	1.00	**0.94**	1.22	1.00	3.33	5.19
(R)150	1.00	0.93	0.00	1.00	**0.97**	1.27	1.00	3.33	6.67
(N)0.04	1.00	1.00	1.00	1.00	**0.43**	0.11	1.00	10.00	24.00
(N)0.06	1.00	1.00	1.00	1.00	**0.56**	1.19	1.00	8.67	20.00
(N)0.08	1.00	1.00	1.00	1.00	**0.99**	0.64	1.00	7.43	17.14
(N)0.1	1.00	1.00	1.00	1.00	**0.98**	0.92	1.00	8.57	18.57
(O)200	1.00	1.00	1.00	1.00	**0.13**	5.00	1.00	6.25	15.00
(O)300	1.00	1.00	1.00	1.00	**0.13**	14.28	1.00	5.78	13.30
(O)400	1.00	1.00	1.00	1.00	**0.14**	16.00	1.00	5.59	21.51
(O)500	1.00	1.00	1.00	1.00	**0.22**	11.11	1.00	5.80	23.00
(I)0.30	1.00	1.00	1.00	1.00	**0.63**	0.50	1.00	6.00	20.00
(I)0.45	1.00	1.00	0.93	1.00	**0.61**	11.63	1.00	5.00	22.50
(I)0.60	1.00	1.00	0.79	1.00	1.00	33.33	1.00	4.50	17.50
(I)0.75	1.00	1.00	0.22	1.00	1.50	50.00	1.00	5.00	30.00



(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

Fig. 3.7 Results of 3D registration with Indoor Scene01 model under different perturbations

Table.3.4 shows the quantitative results of the registration on the Indoor Scene01 model. C is the outstanding conventional method in this registration experiment - the Coherent Point Drift method (CPD). We set the Successful Rate of DO, Mean Square Error of DO, and the Computation Time of DO as the references of comparison. The Bold in this table shows that the stability and registration accuracy of GDO is better than DO and CPD. The registration accuracy of GDO is 24.43% higher than that of DO and 11.31% higher than that of CPD. The Successful Rate of GDO is 9.3% higher than that of DO.

Table.3.5 shows the quantitative results of the registration on the Indoor Scene02 model. C is the outstanding conventional method in this registration experiment - the Coherent Point Drift method (CPD). We set the Successful Rate of DO, Mean Square Error of DO, and the Computation Time of DO as the references of comparison. The Bold in this table shows that the stability and registration accuracy of GDO is better than DO and CPD. The registration accuracy of GDO is 26.36% higher than that of DO and 12.63% higher than that of CPD. The Successful Rate of GDO is 13.43% higher than that of DO.

While FGR and ICP required low computation time for all cases, they had low success rates when the perturbations were high. CPD performed well in all cases except when the number of outliers was high. The running time of IRLS was similar to that of CPD when dealing with the registration of simple models (Bunny, Chef); it did not perform well when the model was highly incomplete. NDT achieved more accurate registration of real scenes than other algorithms; it was the most time-consuming for all cases. For the learning-based algorithms, DO and GDO outperformed the baselines when registering simple models. When dealing with the complex model (Dancing Children) and real large scene models, GDO performed better than DO. This is because DO just considers one single feature that does not consider the internal topology or density distribution of points, which makes it lack the robustness than GDO.

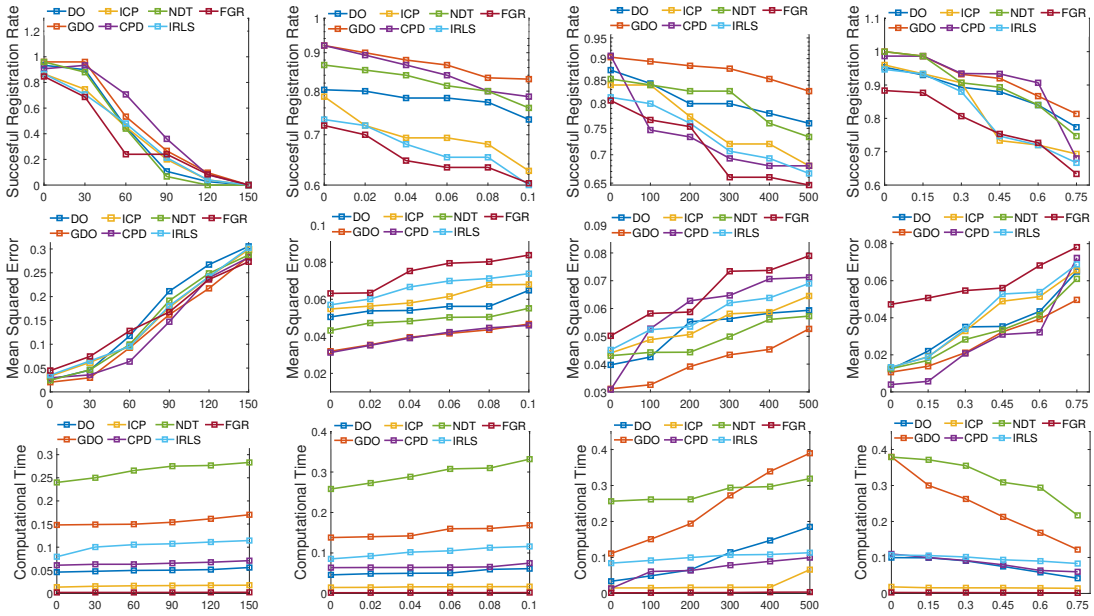
Table 3.4 The quantitative results of the registration on the Indoor Scene01 model (B-Baseline DO; P-Proposed GDO; C-Conventional method CPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.20**	1.12	1.00	0.67	**0.50**	1.00	1.67	0.67
(R)90	1.00	**1.22**	1.22	1.00	0.71	**0.62**	1.00	1.54	0.62
(R)120	1.00	**1.38**	1.15	1.00	**0.78**	0.85	1.00	2.00	0.80
(R)150	1.00	1.00	1.00	1.00	**0.86**	0.88	1.00	1.50	0.57
(N)0.04	1.00	**1.13**	1.09	1.00	**0.76**	0.76	1.00	1.67	0.67
(N)0.06	1.00	**1.09**	1.04	1.00	**0.77**	0.79	1.00	1.62	0.62
(N)0.08	1.00	**1.06**	1.04	1.00	**0.79**	0.81	1.00	2.20	0.90
(N)0.1	1.00	**1.09**	1.07	1.00	0.70	**0.69**	1.00	1.57	0.71
(O)200	1.00	**1.05**	0.87	1.00	**0.75**	1.20	1.00	2.08	0.70
(O)300	1.00	**1.04**	0.84	1.00	**0.79**	1.21	1.00	2.13	0.60
(O)400	1.00	**1.11**	0.88	1.00	**0.78**	1.26	1.00	1.90	0.50
(O)500	1.00	**1.12**	0.91	1.00	**0.87**	1.24	1.00	1.86	0.50
(I)0.30	1.00	**1.06**	1.06	1.00	**0.51**	0.51	1.00	1.78	0.67
(I)0.45	1.00	1.07	**1.09**	1.00	0.67	**0.65**	1.00	1.87	0.73
(I)0.60	1.00	1.05	**1.08**	1.00	0.95	**0.55**	1.00	1.69	0.77
(I)0.75	1.00	**1.06**	0.88	1.00	**0.68**	1.17	1.00	1.82	0.73

Figure.3.9 displays the performance of all methods on registration with $mode_1$. The top is the computational time for registration. The bottom is the $log10MSE$ of all the comparison methods. It can be seen that the accuracy of BCPD is higher than other methods, but BCPD takes almost dozens of times as long as other algorithms. DCP

Table 3.5 The quantitative results of the registration on the Indoor Scene02 model (B-Baseline DO; P-Proposed GDO; C-Conventional method CPD)

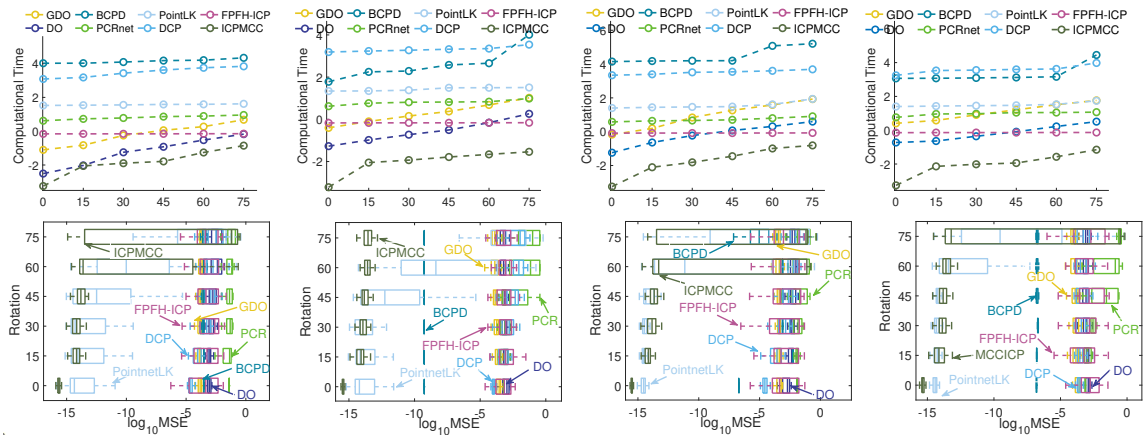
	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	1.18	**1.56**	1.00	0.64	**0.45**	1.00	3.00	1.20
(R)90	1.00	1.88	**2.50**	1.00	0.71	**0.62**	1.00	3.04	1.20
(R)120	1.00	**1.64**	1.64	1.00	**0.72**	0.84	1.00	3.06	1.20
(R)150	1.00	1.00	1.00	1.00	**0.77**	0.83	1.00	3.20	1.20
(N)0.04	1.00	**1.14**	1.12	1.00	**0.76**	0.76	1.00	2.80	1.40
(N)0.06	1.00	**1.13**	1.09	1.00	**0.71**	0.73	1.00	3.00	1.40
(N)0.08	1.00	**1.10**	1.06	1.00	**0.77**	0.81	1.00	2.14	1.00
(N)0.1	1.00	**1.12**	1.05	1.00	0.68	**0.67**	1.00	2.17	1.14
(O)200	1.00	**1.08**	0.89	1.00	**0.73**	1.19	1.00	3.60	1.00
(O)300	1.00	**1.07**	0.85	1.00	**0.75**	1.14	1.00	2.40	0.70
(O)400	1.00	**1.09**	0.86	1.00	**0.72**	1.17	1.00	2.46	0.62
(O)500	1.00	**1.12**	0.91	1.00	**0.80**	1.13	1.00	2.11	0.50
(I)0.30	1.00	**1.09**	1.09	1.00	**0.47**	0.53	1.00	3.11	1.00
(I)0.45	1.00	1.08	**1.11**	1.00	0.83	**0.72**	1.00	3.14	1.14
(I)0.60	1.00	1.03	**1.10**	1.00	0.90	**0.68**	1.00	3.60	1.20
(I)0.75	1.00	**1.01**	0.85	1.00	**0.72**	1.17	1.00	3.00	1.50



(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

Fig. 3.8 Results of 3D registration with Indoor Scene02 data set under different perturbations

takes about the same time as BCPD, but its accuracy and stability are poor. The poor performance also occurs on the PCRnet method. By contrast, PointnetLK can keep higher stability and accuracy when dealing with the registration not over 60^o. Compared with the deep-learning methods, as the traditional learning-based method, DO and GDO can achieve the registration with higher accuracy and stability. FPFH-ICP also performs well. The stability of ICPMCC has a sharp decrease when ICPMCC registers the registration over 60^o.



(a) Airplane627 (b) Car201 (c) Chair964 (d) Toilet378

Fig. 3.9 The registration results on Modelnet40 with perturbation setting $mode_1$.

Table.3.6 shows the log_{10} Mean Square Error of registration results on Modelnet40 with perturbation setting $mode_1$. B means the baseline method - Discriminative Optimisation method (DO); P is the proposed method in this chapter - General Discriminative Optimisation method (GDO); C is the outstanding conventional method in this registration

experiments - Iterative Closest Point algorithm based on maximum correntropy criterion (ICPMCC). We set the absolute value of log_{10} Mean Square Error of DO as the reference of comparison. The value of the absolute value of the log_{10} Mean Square Error is higher, and the registration accuracy is higher. Q_0, Q_4 and IQR aim to measure data distribution in boxplot figures. Q_0 and Q_4 are the minimum and maximum respectively. And IQR is the interquartile range, which measures the distance between the upper and lower quartiles; the shorter the distance, the more stability of the method. It can be seen that GDO, ICPMCC and PointNetLK all have higher accuracy than DO, as shown the bold in this table. Compared with ICPMACC and PointNetLK, GDO has higher stability.

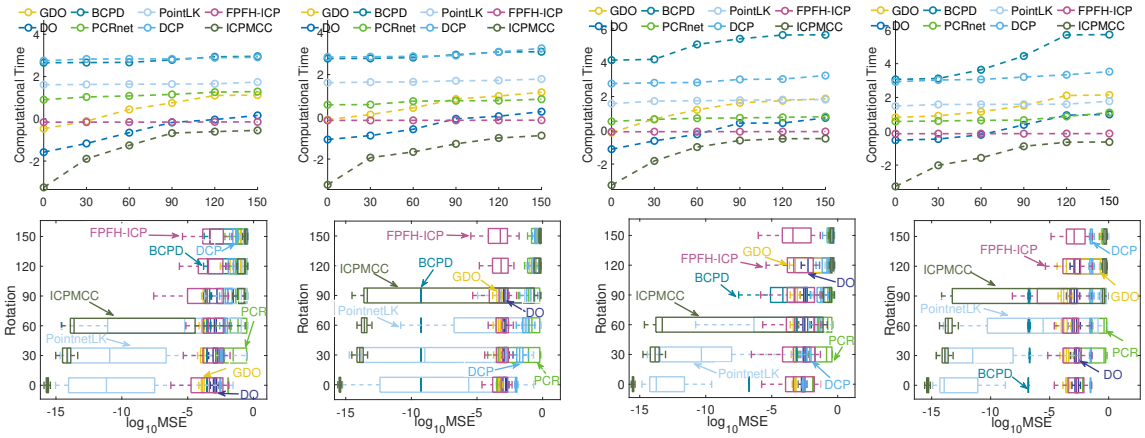
Table 3.6 The log_{10} Mean Square Error of registration results on Modelnet40 with perturbation setting $mode_1$ (B-Baseline DO; P-Proposed GDO; C-Conventional method ICPMCC; D-Deep learning method PointNetLK; Q_0 - Minimum; Q_4 - Maximum; IQR - Interquartile range)

	Q_0				Q_4				IQR			
	B	P	C	D	B	P	C	D	B	P	C	D
627(45)	1.00	1.15	3.67	**3.73**	1.00	2.00	**6.35**	6.25	1.00	**0.29**	0.71	6.86
627(60)	1.00	1.33	**3.94**	3.94	1.00	**2.35**	0.59	0.71	1.00	**0.33**	9.50	5.50
627(75)	1.00	1.15	**4.17**	2.78	1.00	**2.71**	0.57	0.57	1.00	**0.21**	9.43	3.07
201(45)	1.00	1.21	3.82	**3.92**	1.00	1.39	**6.08**	2.61	1.00	**0.29**	0.71	6.14
201(60)	1.00	1.25	3.55	3.55	1.00	1.30	**5.65**	0.43	1.00	0.60	**0.30**	7.20
201(75)	1.00	1.20	**3.75**	1.75	1.00	1.30	**5.48**	0.35	1.00	**1.00**	2.00	4.00
964(45)	1.00	1.33	4.83	**4.97**	1.00	1.60	**6.70**	6.50	1.00	**0.14**	1.43	1.86
964(60)	1.00	1.72	**6.00**	6.00	1.00	**1.78**	0.56	0.44	1.00	**0.71**	17.14	12.86
964(75)	1.00	1.54	5.11	**5.36**	1.00	**2.92**	0.23	0.23	1.00	1.50	58.00	33.00
378(45)	1.00	1.29	3.89	3.89	1.00	1.40	5.20	**5.60**	1.00	1.33	2.33	**1.00**
378(60)	1.00	1.29	3.76	**3.95**	1.00	1.39	**5.57**	3.04	1.00	**3.00**	3.00	15.00
378(75)	1.00	1.29	**3.95**	3.95	1.00	**1.52**	0.43	0.43	1.00	**1.50**	62.50	35.00

Figure.3.10 shows the registration results on the perturbation of larger rotations $mode_2$. The stability and accuracy of ICPMCC and PointnetLK are worse when ICPMCC and PointnetLK handle the registration over 60^o. The performance of DCP and PCR is unstable as ever. DO, GDO, and BCPD can keep the high accuracy and stability until they register point sets with large rotations (over 120^o). Nevertheless, the accuracy of GDO is higher than that of BCPD and DO when dealing with the registration over 120^o. FPFH-ICP still keeps its high stability and accuracy, and the performance of ICPMCC is poor once it is used to achieve the registration with larger rotations.

Table.3.7 shows the log_{10} Mean Square Error of registration results on Modelnet40 with perturbation setting $mode_2$. C is the outstanding conventional method in this registration experiments - Iterative Closest Point based on Fast point feature histogram (FPFH-ICP). We set the absolute value of log_{10} Mean Square Error of DO as the reference of comparison. It can be seen that GDO can keep higher stability than other methods when handling the registration with higher rotations. Compared with Table.3.6, we can find that the accuracy of GDO on the registration with large rotations is 6.91% lower than that of the registration with small rotations. This decline is as high as 78.49 % for PointNetLK.

Figure.3.11 illustrates the registration results on Modelnet40 data set with multiple perturbations $mode_3$. DO and GDO can keep the higher stability and accuracy on the



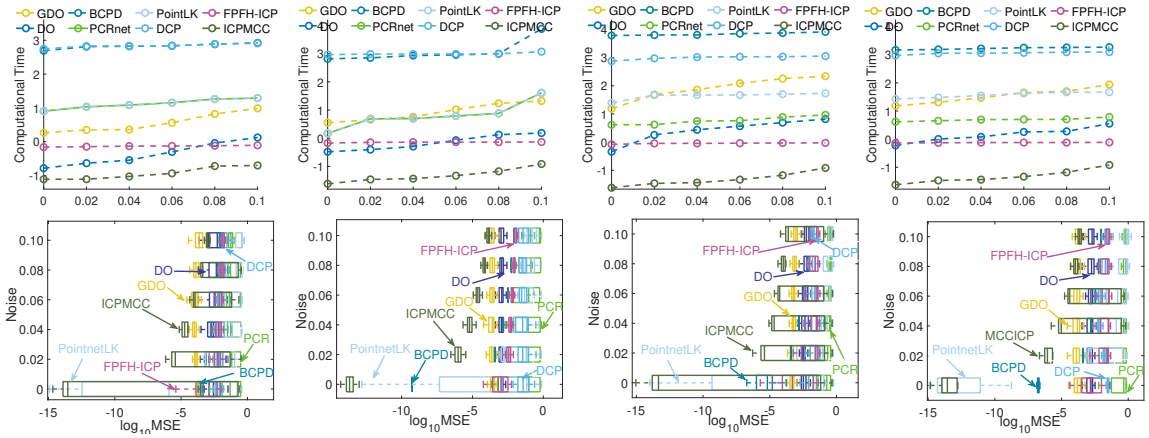
(a) Airplane627 (b) Car201 (c) Chair964 (d) Toilet378

Fig. 3.10 The registration results on Modelnet40 with perturbation setting $mode_2$.

Table 3.7 The log_{10} Mean Square Error of registration results on Modelnet40 with perturbation setting $mode_2$ (B-Baseline DO; P-Proposed GDO; C-Conventional method FPFH-ICP; D-Deep learning method PointnetLK; Q_0 - Minimum; Q_4 - Maximum; IQR - Interquartile range)

	Q_0				Q_4				IQR			
	B	P	C	D	B	P	C	D	B	P	C	D
627(90)	1.00	1.20	**2.29**	1.14	1.00	**2.67**	1.33	0.53	1.00	**0.30**	2.50	1.10
627(120)	1.00	1.09	**2.61**	0.87	1.00	1.00	**1.67**	0.58	1.00	1.20	3.00	**1.00**
627(150)	1.00	1.17	**2.94**	0.83	1.00	1.25	**1.50**	0.50	1.00	**0.17**	2.00	0.33
201(90)	1.00	1.40	**1.77**	1.70	1.00	**1.20**	0.72	0.48	1.00	**0.80**	3.40	2.60
201(120)	1.00	1.11	**2.67**	1.11	1.00	1.50	**1.67**	0.42	1.00	**0.50**	3.50	0.50
201(150)	1.00	1.38	**4.15**	1.15	1.00	1.00	**1.50**	0.42	1.00	2.00	18.00	2.00
964(90)	1.00	1.20	**1.83**	0.70	1.00	**1.20**	0.64	0.48	1.00	2.00	11.00	**1.50**
964(120)	1.00	1.36	**1.86**	0.57	1.00	0.92	**1.42**	0.42	1.00	**0.29**	2.71	1.00
964(150)	1.00	1.09	**5.09**	0.91	1.00	0.92	**1.33**	0.42	1.00	**2.00**	21.00	2.00
378(90)	1.00	1.15	**2.08**	0.28	1.00	**1.20**	0.64	0.48	1.00	2.00	16.50	**0.50**
378(120)	1.00	1.11	**1.50**	0.28	1.00	0.80	**1.00**	0.33	1.00	9.33	7.00	**0.67**
378(150)	1.00	1.09	**4.73**	0.73	1.00	0.92	**1.33**	0.42	1.00	**1.00**	23.00	2.00

registration with multiple perturbations, compared with other methods. The ability of deep-learning methods to handle the registration with multiple perturbations is poor than that of the traditional methods. The performance of FPFH-ICP is still stable, but the accuracy is not high.



(a) Airplane627 (b) Car201 (c) Chair964 (d) Toilet378

Fig. 3.11 The registration results on Modelnet40 with perturbation setting $mode_3$.

Table.3.8 shows the log_{10} Mean Square Error of registration results on Modelnet40 with perturbation setting $mode_3$. C is the outstanding conventional method in this registration experiments - Iterative Closest Point algorithm based on maximum correntropy criterion (ICPMCC). We set the absolute value of log_{10} Mean Square Error of DO as the reference of comparison. It can be seen that the registration accuracy of GDO is 26% higher than that of DO, 3.64% higher than that of ICPMCC and 81.99% higher than that of PointNetLK. The stability of GDO is 7.91% higher than that of PointNetLK.

Table 3.8 The log_{10} MEan Square Error of registration results on Modelnet40 with perturbation setting $mode_3$ (B-Baseline DO; P-Proposed GDO; C-Conventional method ICPMCC; D-Deep learning method DCP; Q_0 - Minimum; Q_4 - Maximum; IQR - Interquartile range)

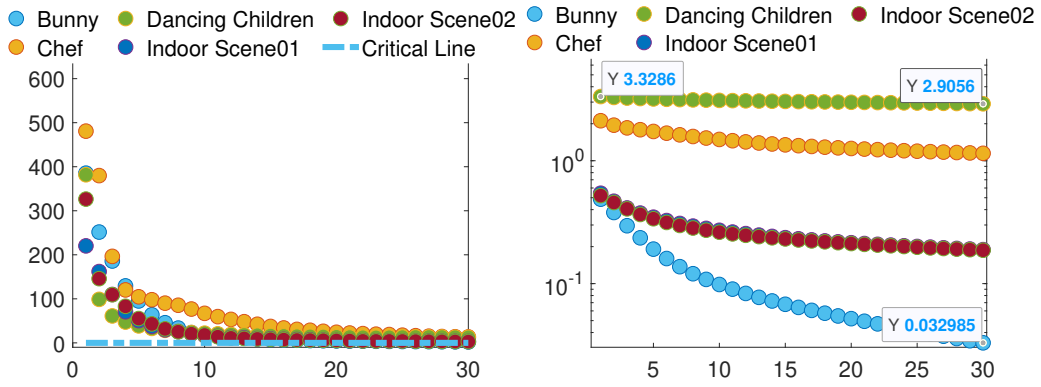
	Q_0				Q_4				IQR			
	B	P	C	D	B	P	C	D	B	P	C	D
627(0.06)	1.00	**1.26**	0.66	0.66	1.00	**1.73**	0.91	0.82	1.00	**0.33**	0.56	0.56
627(0.08)	1.00	**1.21**	0.63	0.61	1.00	**1.73**	0.82	0.82	1.00	**0.11**	0.50	0.44
627(0.10)	1.00	**1.25**	0.55	0.53	1.00	**1.77**	0.82	0.59	1.00	**0.38**	0.40	0.40
201(0.06)	1.00	1.26	**1.34**	0.66	1.00	1.36	**1.71**	0.38	1.00	0.33	**0.11**	1.11
201(0.08)	1.00	1.39	**1.48**	0.79	1.00	1.36	**1.64**	0.46	1.00	**0.11**	0.22	0.33
201(0.10)	1.00	1.28	**1.28**	0.64	1.00	1.39	**1.54**	0.36	1.00	0.21	**0.11**	0.56
964(0.06)	1.00	1.19	**1.59**	0.78	1.00	**1.48**	0.22	0.74	1.00	**0.33**	5.67	0.33
964(0.08)	1.00	1.19	**1.53**	0.75	1.00	**1.48**	1.04	0.74	1.00	**0.28**	0.29	0.29
964(0.10)	1.00	1.19	**1.56**	0.75	1.00	**1.48**	0.22	0.87	1.00	1.00	15.50	**0.50**
378(0.06)	1.00	1.50	**1.59**	0.78	1.00	**1.68**	0.20	0.80	1.00	3.00	20.50	**1.50**
378(0.08)	1.00	1.20	**1.23**	0.60	1.00	1.68	**1.88**	0.72	1.00	1.25	**0.25**	1.25
378(0.10)	1.00	1.20	**1.15**	0.60	1.00	1.64	**1.72**	0.68	1.00	1.25	**0.50**	0.50

In summary, the learning-based methods (DO and GDO) have higher stability and robustness compared with deep-learning methods (PCRnet, PointnetLK, and DCP) and other traditional methods. FPFH-ICP performs well even on the registration with larger rotations, but the accuracy of FPFH-ICP is not better, which may be caused by the fewer

iterations for FPFH to find correspondences. The ability to achieve more accurate and stable registration on larger rotations or multiple perturbations for the deep-learning methods and ICPMCC is limited.

The benefit of the FPFH-ICP is the ability to handle registration with larger rotations while maintaining higher stability. Comparing the registration results on $mode_2$ and $mode_3$, it can be seen that the only drawback of the traditional learning-based methods (DO and GDO) is the less ability to register point clouds over 120^o, which illustrates that the learning-based methods are more vulnerable on rotations, not noises. In addition, the features in GDO can be replaced by any features extracted by 3D feature descriptors such as Fast Point Feature Histograms (FPFH) descriptors, Signature of Histogram of Orientations (SHOT), and so on. The potential issue of the usage of various descriptors is whether it will increase the degree of over-fitting of the learning-based methods.

Verify Convergence. Figure.3.12 shows the Convergence Criteria and Training Error of our method on different data sets. (a) shows the value of $\sum_{i=1}^{N}\left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_f\left(x_t^i\right)$on different data sets. (b) illustrates the training error of our method on different data sets. We can find that the $\hat{D}\mathbf{H}_f$ in our method meets the convergence condition $\sum_{i=1}^{N}\left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_f\left(x_t^i\right) > 0$ for all data sets, and the training error of our method decreases in each iteration.



(a) Convergence Criteria (b) Training Error

Fig. 3.12 The Convergence Criteria and Training Error of our method on different data sets.

3.6 Conclusion

This chapter proposes the general discriminative optimisation (GDO) method to solve the transformation parameter estimation in point set registration by learning update directions from different features of training samples. Specifically, GDO designs an approach to achieve the *collaboration* of the different extracted features from point sets to reduce the effect of perturbations on updating directions. GDO combines a coordinates-based feature and a density-based feature to update the gradient map to improve the accuracy and robustness of transformation estimation. GDO outperformed state-of-the-art registration approaches on different data sets. The registration accuracy of GDO is almost 4.00% higher than that of DO on the Bunny, Chef, and Dancing Children models. The registration

accuracy of GDO is almost 25.00% higher than that of DO on the Indoor Scene models. And the successful Rate of GDO is almost 14% higher than that of DO. GDO has higher stability than conventional methods and deep-learning-based methods on the ModelNet40 data set. The decline of the registration accuracy of GDO on the registration under higher rotations is 6.91%, and the decline is as high as 78.49 % for PointNetLK. The registration accuracy of GDO is 26.00% higher than that of DO on the registration under different noises. Meanwhile, the stability of GDO is 7.91% higher than that of PointNetLK. The major advantage of GDO over traditional registration methods and learning-based registration methods includes robustness to outliers and other perturbations, which is more prominent when dealing with complex 3D models and real scene model registration. The limitation of GDO is that the training point cloud and the test point cloud are highly relevant, limiting its ability to train many point clouds and achieve multiple point clouds registration like the registration methods based on deep learning. In addition, the feature extraction approach of GDO takes longer as the number of points increases. Future works of interest are to design a feature function that is more robust to perturbations and more efficient and to design a registration framework to enable GDO to achieve multiple point clouds registration. The strong theoretical foundation and good registration performance of GDO suggest its usefulness as a general-purpose registration technique.

Chapter 4

Reweighted Discriminative Optimisation for Least-Squares Problems with Point Cloud Registration

4.1 Introduction

The General Discriminative Optimisation (GDO) method achieves the robust point cloud registration via the collaboration of different features, and it learns the sequence of regressions in the same way as supervised sequential update (SSU) methods. They all utilise the least-square regression to estimate parameter vectors, that is, the updating gradients are learned by approaching the currently estimated parameter vectors towards the ground truth. However, that the $\ell 2$ norm of the residual vector approaches to 0 does not represent the best fitting between the estimated and real models, because different components of parameter vectors have various impacts on fitting results. Thus, GDO and other learning-based optimisation methods ignore the fact that different components of parameter vectors play different roles at registration.

In this case, Reweighted Discriminative Optimisation (RDO) is proposed to address the issue of asymmetrical contributions of the components of the parameter vector on fitting accuracy. The new method assigns different weights to components of parameter vectors in each iteration to emphasise the impact of each component on the fitting results. It is worth noting that RDO is different from the Iterative Reweighted Least Squares (IRLS) [103]. IRLS attains weights through an M-estimation criterion to emphasise specific components or the range of equations, where the weighting matrix of $\ell 2$ norm is an identity matrix. When IRLS is applied to point cloud registrations to reduce the influence of outliers [33], the M-estimator of IRLS acts on the coordinates of points directly. The RDO method, however, utilises the characteristics of fitting errors to adjust the weights which are assigned to the components of parameter vectors.

More specifically, the aim in this chapter is the accurate estimations of parameter vectors through reweighting the different components of parameter vectors in the learning process. The potential of RDO is demonstrated in handling challenging visualisation

tasks including 3D point cloud registration, object recognition and multi-view stitching. Experimental results show that RDO outperforms the state-of-the-art algorithms in terms of robustness and accuracy on both synthetic data sets and range-scan data sets.

4.2 Motivation

4.2.1 Motivation from SSU Methods

Supervised sequential update (SSU) algorithms predict a set of parameters by sequentially refining previously estimated parameters. The refinement is performed by establishing a sequence of regressors to update the parameters. The specific refinement process can be cast as follows:

$$\mathbf{x}_{t+1} = C\left(\mathbf{x}_t, \mathbf{F}_{t+1} \circ \mathbf{h}(\mathbf{x}_t)\right) \tag{4.1}$$

Where $\mathbf{x}_t \in \mathbf{R}^p$ is the estimated parameter vector at time t, $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$ extracts features from the input data, $\mathbf{F}_{t+1} : \mathbf{R}^f \to \mathbf{R}^d$ is a regressor that maps the feature $\mathbf{h}\left(\mathbf{x}_t\right)$ to an update vector, $C : \mathbf{R}^p \times \mathbf{R}^d \to \mathbf{R}^p$ is the operator working on the parameter $\mathbf{x}_t$ and the update map $\mathbf{F}_{t+1} \circ \mathbf{h}$.

The regressor $\mathbf{F}_{t+1}$ can be attained through minimising the residual vector of the estimated vector $\mathbf{x}_{t+1}$ and the ground truth $\mathbf{x}_*$.

$$\begin{aligned} \mathbf{F}_{t+1} &= \min_{\hat{\mathbf{F}}} \sum_{i=1}^{N} \left\| \mathbf{x}_{t+1}^i - \mathbf{x}_*^i \right\|_2^2 \\ &= \min_{\hat{\mathbf{F}}} \sum_{i=1}^{N} \left\| C\left(\mathbf{x}_t^i, \hat{\mathbf{F}} \circ \mathbf{h}(\mathbf{x}_t^i)\right) - \mathbf{x}_*^i \right\|_2^2 \end{aligned} \tag{4.2}$$

Where N is the number of samples.

4.2.2 Motivation from Point Cloud Registration

Let $\mathbf{M}, \mathbf{S}$ be two point sets in a finite-dimensional real vector space $\mathbf{R}^3$, which contains N_m and N_s points respectively. Point cloud registration is to find a spatial transformation $\mathbf{T}$ to be applied to the scene set $\mathbf{S}$ such that the difference between $\mathbf{S}$ and the model set $\mathbf{M}$ is minimised. The rigid rotation matrix is not closed with respect to addition, which limits the direct derivation of the objective function to the rotation matrix. To overcome this challenge, Lie groups are used to represent the transformations in 2D and 3D spaces [104] so that the point cloud registration can be turned into an optimisation problem over the Lie group, where parameter vectors $\mathbf{x}_{t+1}, \mathbf{x}_* \in \mathbf{R}^6$.

Assume $\mathbf{x}_* = \mathbf{0}_6$, $\mathbf{x}_{t+1} = \hat{\mathbf{x}}_k$, $\hat{\mathbf{x}}_k$ is the k-th column of the diagonal matrix $\mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$
$$= [\hat{\mathbf{x}}_1, \quad \hat{\mathbf{x}}_2, \quad \hat{\mathbf{x}}_3, \quad \hat{\mathbf{x}}_4, \quad \hat{\mathbf{x}}_5, \quad \hat{\mathbf{x}}_6]$$

For any $\hat{\mathbf{x}}_k, k \in \{1,2\cdots 6\}$, the $\ell 2$ norm of the residual vector $\|\mathbf{x}_* - \hat{\mathbf{x}}_k\|_2^2$ is 0.01. Each $\hat{\mathbf{x}}_k$ as the parameter vector is applied to register the hand model, and the registration results are shown in Figure.4.1.

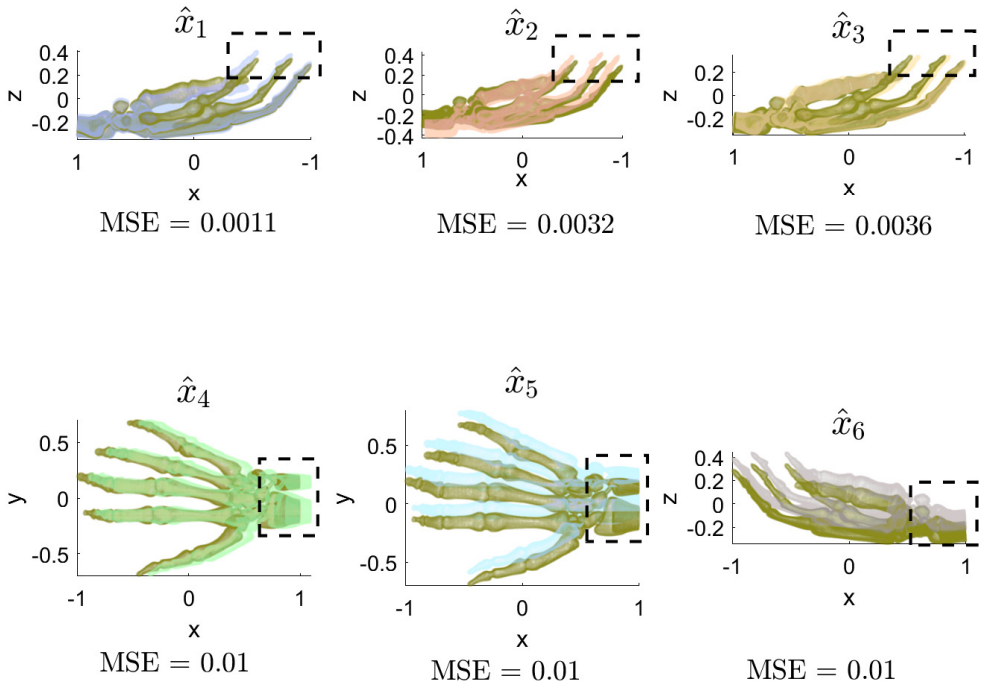


Fig. 4.1 Registration results with different transformation parameter vectors $\hat{\mathbf{x}}_k, k \in \{1,2\cdots 6\}$.

Figure.4.1 shows that although the norms of residual vectors $\|\hat{\mathbf{x}}_k - \mathbf{x}_*\|_2^2$ reach the same value 0.01, the mean squared errors of the registration results are different, which means that the components of a parameter vector have different impact on the registration results.

Figure.4.2 illustrates the impact of each component of parameter vectors on transformation. $\hat{\mathbf{x}}_k, k \in \{1,2,3\}$ represent the rotations along with the x-axis, y-axis, and z-axis. $\hat{\mathbf{x}}_k, k \in \{4,5,6\}$ show the translation. It can be seen that each component plays a different role in transforming the dark-green plane. Combining with Figure.4.1, we can find that

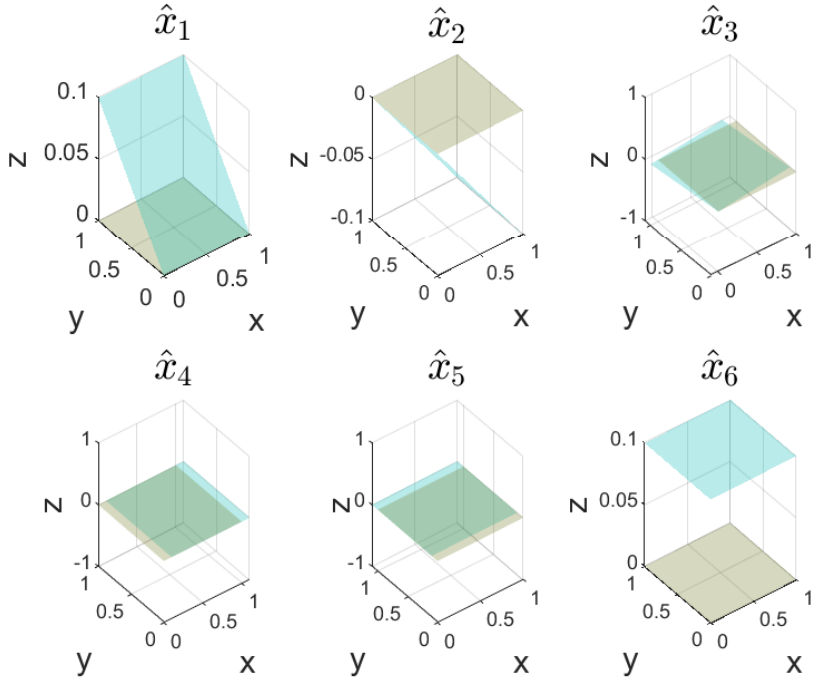


Fig. 4.2 The transformation with $\hat{\mathbf{x}}_k, k \in \{1,2\cdots 6\}$

although the components of parameter vectors have changed on the same scale 0.1, the registration error is different, and the $\hat{\mathbf{x}}_k, k \in \{1,2,3\}$ is more robust with regard to perturbations than translation. Therefore, the goal of RDO is to improve the robustness of translation to perturbations and maintain the robustness of rotation.

To reduce the registration error and improve robustness while the estimated parameter vector $\mathbf{x}_{t+1}$ is approaching the ground truth $\mathbf{x}_*$, RDO assigns different weights to the components of the parameter vectors to emphasise the influence of each component of the vectors on the final registration results.

4.2.3 Sequence of Update Maps

Let $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$ be a function that encodes features of a data set, and $\mathbf{D}_{t+1} \in \mathbf{R}^{p\times f}$ be a matrix that maps the feature $\mathbf{h}$ to a update vector. Given an initial parameter vector $\mathbf{x}_0 \in \mathbf{R}^p$, the iterative updating process can be defined as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{D}_{t+1}\mathbf{h}(\mathbf{x}_t) \tag{4.3}$$

The update process ends until $\mathbf{x}_{t+1}$ converges to a stationary point. And the sequence of matrices $\mathbf{D}_{t+1}, t = 0,1\cdots$ are learned through approximating estimated parameter vector

$\mathbf{x}_{t+1}^i$ to the ground truth $\mathbf{x}_*^i$.

$$\begin{aligned}\mathbf{D}_{t+1} &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{x}_{t+1}^i - \mathbf{x}_*^i\right\|_2^2 \\ &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{x}_t^i - \hat{\mathbf{D}}\mathbf{h}\left(\mathbf{x}_t^i\right) - \mathbf{x}_*^i\right\|_2^2\end{aligned} \tag{4.4}$$

Where N is the number of samples, $\mathbf{x}_t^i$ is the parameter vector of i-th sample at the t -th iteration. For simplicity, we denote $\mathbf{x}_t^i$ as $\mathbf{x}_t$ when the formula/function applies to all samples.

Considering that each component of the parameter vector $\mathbf{x}_t$ can have a different impact on the fitting error, RDO explores a weighting vector $\mathbf{w}_t \in \mathbf{R}^p$ to emphasise the impact of each component of the parameter vector on fitting results.

$$\mathbf{w}_t = [w_1,\ w_2,\ w_3,\ \cdots\ w_p]^{\mathrm{T}} \tag{4.5}$$

$$\begin{aligned}\mathbf{D}_{t+1} &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{w}_t \odot \left(\mathbf{x}_{t+1}^i - \mathbf{x}_*^i\right)\right\|_2^2 \\ &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{w}_t \odot \left(\mathbf{x}_t^i - \hat{\mathbf{D}}\mathbf{h}\left(\mathbf{x}_t^i\right) - \mathbf{x}_*^i\right)\right\|_2^2\end{aligned} \tag{4.6}$$

Where $\odot$ represents Hadamard product. This weighting vector $\mathbf{w}_t$ can be transformed into a weighting diagonal matrix $\mathbf{W}_t \in \mathbf{R}^{p\times p}$.

$$\mathbf{W}_t = \begin{bmatrix} w_1 & 0\ldots & 0 \\ 0 & w_2\ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & w_p \end{bmatrix}_{p\times p}$$

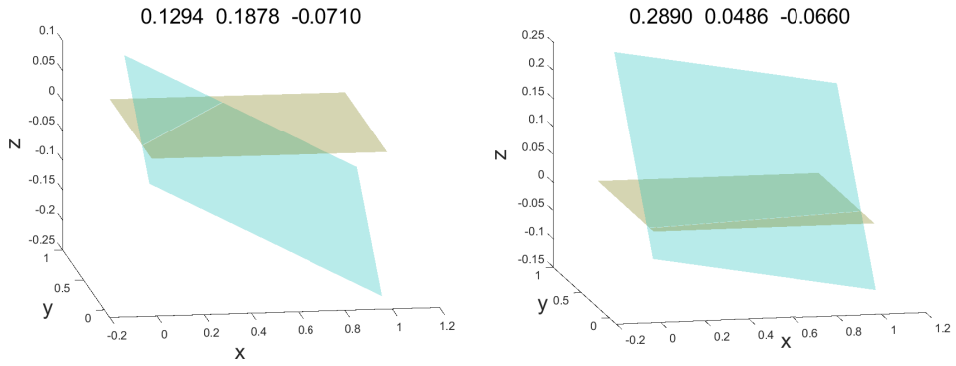
Then, we get

$$\begin{aligned}\mathbf{D}_{t+1} &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{W}_t \left(\mathbf{x}_{t+1}^i - \mathbf{x}_*^i\right)\right\|_2^2 \\ &= \min_{\hat{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{W}_t \left(\mathbf{x}_t^i - \hat{\mathbf{D}}\mathbf{h}\left(\mathbf{x}_t^i\right) - \mathbf{x}_*^i\right)\right\|_2^2\end{aligned} \tag{4.7}$$

4.2.4 Design Weighting Matrix $\mathbf{W}_t$

Before designing weights for components of parameter vectors, we explored the function of weights on transformation, as shown in Figure.4.3. Figure.4.3 shows the transformation with different weights. (a) shows the transformation process from the dark-green to green when $\mathbf{w}_t = [0.2, 0.6, 0.1, 1/30, 1/30, 1/30]^{\mathrm{T}}$. The element of the vector at the top of the sub-figure $[0.1294, 0.1878, -0.0710]$ corresponds to the rotation degree along with the X, Y, Z axes respectively. (b) illustrates the rotation information with the weights

$[0.6, 0.2, 0.1, 1/30, 1/30, 1/30]$. It can be seen that the weights control the scale of transformation. The greater the weight, the larger the scale of the transformation.



(a) (b)

Fig. 4.3 The transformation with different weights.

Figure.4.3 shows that the weights assigned to the components of parameter vectors have the ability to control the scale of transformation, and the greater the weight, the larger the scale of the transformation. To determine which component should be allocated a larger weight, we introduce a 'detector' $\tilde{\mathbf{x}}_{t_k}$ to 'probe' the difference between the component of the current estimation $\mathbf{x}_t$ and that of $\mathbf{x}_*$.

$$\tilde{\mathbf{x}}_{t_k} = \mathbf{x}_t - \mathbf{c}_k,\ k \in \{1, 2, \cdots, p\} \tag{4.8}$$

Where $\tilde{\mathbf{x}}_{t_k}$ is the changed parameter vector. $\mathbf{c}_k \in \mathbf{R}^{p\times 1}$ is a vector with the length of p, and the k-th element in this vector is a constant, and other elements are zero. Except for the k-th element, $\tilde{\mathbf{x}}_{t_k}$ is the same as $\mathbf{x}_t$.

The matching error between the 'detector' and ground-truth is the basis to assign weights. The greater error means the larger difference between the k_{th} component of the current estimation $\mathbf{x}_t$ and that of ground-truth $\mathbf{x}_*$. We need to assign a greater weight to reduce the difference. That is to say that the matching error determines the weights. And the weights act on the component of transformation vectors, then influence the transformation scale that determines the matching error. The relationship between weights and matching error is provided in Figure.4.4. The principle to design the weights is that the weight monotonically increases as the matching error increases, which guarantees that larger weights corresponding to the greater matching error.

The weight w_k depends on err_k^i, which is the fitting error of the i-th 'detector' on the k-th dimension of the parameter vector space. err_k^i involves the parameter vector pair $\left\langle \tilde{\mathbf{x}}_{t_k}^i, \mathbf{x}_*^i \right\rangle$.

$$err_k^i = \frac{1}{N_{q_i}} \sum_{j=1}^{N_{q_i}} \left\| F\left(\tilde{\mathbf{x}}_{t_k}^i, \mathbf{q}_j^i\right) - F\left(\mathbf{x}_*^i, \mathbf{q}_j^i\right) \right\|_2^2 \tag{4.9}$$

Where $\mathbf{q}_j^i$ represents the j-th elements of the i-th sample $\mathbf{Q}_i$, which can be the j-th point of the i-th point cloud $\mathbf{Q}_i$; F is a function that applies the parameter vector to $\mathbf{q}_j^i$.

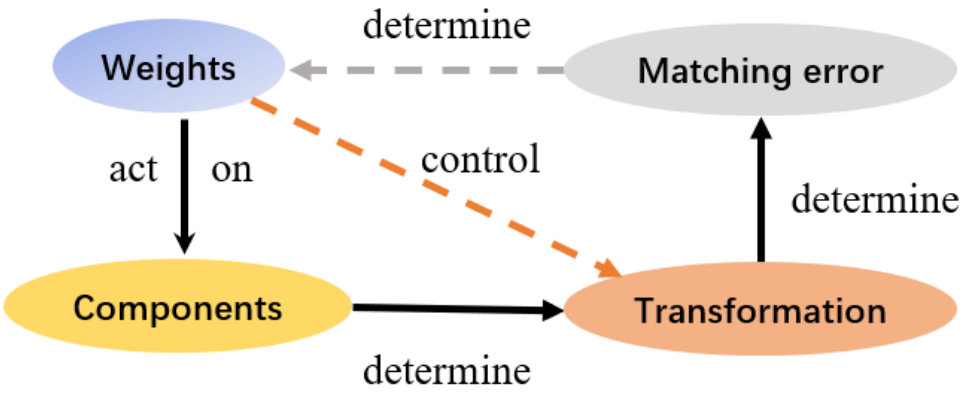


Fig. 4.4 The relationship between weights and matching error

The fitting error of the $\mathbf{Q}_i$ on parameter vector space is $\mathbf{err}^i = \left[err_1^i,\ err_2^i,\ \cdots\ err_p^i\right]^{\mathrm{T}}$.

$$Err_k^i = \frac{err_k^i - \min(\mathbf{err}^i)}{\max(\mathbf{err}^i) - \min(\mathbf{err}^i)} \tag{4.10}$$

$\mathbf{Err}^i = \left[Err_1^i,\ Err_2^i,\ \cdots\ Err_p^i\right]^{\mathrm{T}}$ refers to the normalisation of vector $\mathbf{err}^i$. $\mathbf{E}$ is a $N \times p$ matrix made up of vector $\mathbf{Err}^i$. N is the number of samples. We express $[\mathbf{E}]_k$ as the k-th row of matrix $\mathbf{E}$, $[\mathbf{E}]_{:,k}$ as the k-th column of matrix $\mathbf{E}$.

Statistical skills are used to get the normal distributions of the registration errors of point clouds on each dimension of the parameter vector space, as shown in Figure.4.5. Figure.4.5 shows the normal distributions of registration errors with parameter vectors pairs $\langle \tilde{\mathbf{x}}_{t_k}, \mathbf{x}_* \rangle$, $k \in \{1, 2, \cdots, p\}$. Curves with different colours represent different distributions. The normal distribution is plotted by fitting the histogram of the fitting errors $[\mathbf{E}]_{:,k}, k = 1, 2 \cdots p$. p is 6 for Lie-algebra of transformation matrices. It can be seen that the distributions corresponding to the different parameter vector pair $\langle \tilde{\mathbf{x}}_{t_k}, \mathbf{x}_* \rangle$ are different.

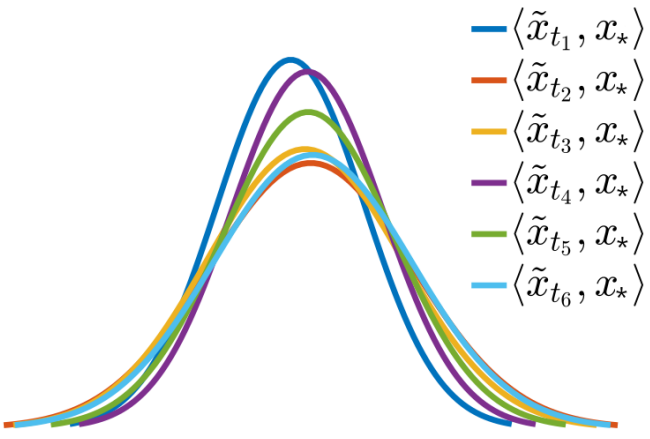


Fig. 4.5 Normal distributions of registration errors with parameter vectors pairs $\langle \tilde{\mathbf{x}}_{t_k}, \mathbf{x}_* \rangle$, $k \in \{1, 2, \cdots, p\}$.

Weights w_k is calculated according to the characteristics of normal distributions of errors.

$$\begin{aligned} \mu_k &= \frac{\sum_{i=1}^{N}[\mathbf{E}]_{i,k}}{N} \\ \sigma_k^2 &= \frac{\sum_{i=1}^{N}\left([\mathbf{E}]_{i,k}-\mu_k\right)^2}{N} \end{aligned} \tag{4.11}$$

$$f(\mu_k) = \frac{1}{\sigma_l\sqrt{2\pi}}\exp(-\frac{1}{2\sigma_l^2}(\mu_k-\mu_l)^2) \tag{4.12}$$

Where μ_k and σ_k^2 are the mean and variance of the fitting errors of all samples on the k-th dimension of the parameter vector. $f(\mu_k)$ depicts the probability density of the normal distribution $\mathcal{N}(\mu_l,\sigma_l)$ on μ_k. μ_l is the maximum or the minimum of μ_k, and the corresponding σ_l is the minimum. Details to select μ_l and σ_l will be explained in the following.

Figure.4.6 shows the criteria for designing weights, $\mu_1<\mu_2<\mu_3$, $\delta_3<\delta_1<\delta_2$. The orange circles are the $f(\mu_1)$ and $f(\mu_3)$, where the $\mu_l=\mu_2$. It can be seen that although $\mu_1<\mu_3$, the $f(\mu_1)\approx f(\mu_3)$, which does not satisfy the principle of designing weights. The blue plus signs illustrate the $f(\mu_2)$ and $f(\mu_3)$, where the $\mu_l=\mu_1$. And the green squares represent the $f(\mu_1)$ and $f(\mu_2)$, where the $\mu_l=\mu_3$. The latter two cases ensure the monotonicity of weight assignment. The black lines show the difference between $f(\mu_2)$ and $f(\mu_3)$, $f(\mu_2)$ and $f(\mu_1)$. It can be seen that the difference between $f(\mu_2)$ and $f(\mu_1)$ is larger than the difference between $f(\mu_2)$ and $f(\mu_3)$. Assume $\mu_1<\mu_2<\mu_3<\mu_4<\mu_5<\mu_6$,

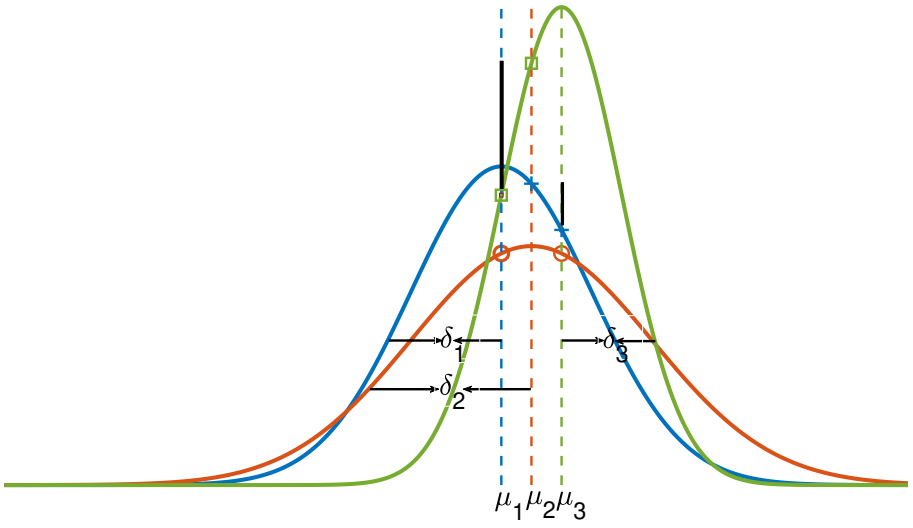


Fig. 4.6 The criteria for designing weights

we choose μ_1 and μ_6 as the candidates for μ_l due to the monotonicity of weight assignment.

if $\delta_1<\delta_6$, $\mu_l=\mu_1$

$$w_k = \exp(\frac{1}{2}(1-f(\mu_k))^t) \tag{4.13}$$

if $\delta_1>\delta_6$, $\mu_l=\mu_6$

$$w_k = \exp(\frac{1}{2}(f(\mu_k))^t) \tag{4.14}$$

Designing weights in this way guarantees monotonicity and makes sure that the difference between weights is sufficiently to reflect the difference between matching errors.

4.2.5 Learning a SUM

Suppose we are given a set of training data $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{h}(\mathbf{x}_0^i), \mathbf{Q}_i\right)\right\}_{i=1}^N$, including N samples $\mathbf{Q}_i$, where $\mathbf{x}_0^i \in \mathbf{R}^p$ is the initial parameter vector of the i-th sample (*e.g.* the i-th points cloud), $\mathbf{x}_*^i \in \mathbf{R}^p$ is the ground truth (*e.g.* the transformation parameters), $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$ provides feature information of samples. For simplicity, we denote $\mathbf{h}(\mathbf{x}_t^i)$ as $\mathbf{h}_t^i$ to represent the feature of the i-th sample $\mathbf{Q}_i$ at the t-th iteration.

The goal of RDO is to learn a sequence of maps $\mathbf{D}_{t+1}$ that update $\mathbf{x}_0^i$ to $\mathbf{x}_*^i$.

$$\mathbf{D}_{t+1} = \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_*^i - \mathbf{x}_t^i + \hat{\mathbf{D}} \mathbf{h}_t^i \right) \right\|_2^2 \tag{4.15}$$

In practice, to prevent over fitting, ridge regression is used to learn the maps:

$$\mathbf{D}_{t+1} = \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_*^i - \mathbf{x}_t^i + \hat{\mathbf{D}} \mathbf{h}_t^i \right) \right\|_2^2 + \lambda \left\| \hat{\mathbf{D}} \right\|_F^2 \tag{4.16}$$

Where $\|\cdot\|_F$ is the Frobenius norm, and λ is a hyperparameter.

The initial training data $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{h}_0^i, \mathbf{Q}_i\right)\right\}_{i=1}^N$ is applied to (4.8), (4.9), (4.10), (4.11), (4.12) and (4.13) to get an initial weighting matrix $\mathbf{W}_0$ at first. Next, an initial update map $\mathbf{D}_1$ can be learned by applying the initial training data $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{h}_0^i, \mathbf{Q}_i\right)\right\}_{i=1}^N$ and the weighting matrix $\mathbf{W}_0$ to (4.16), then $\mathbf{D}_1$ will be applied to (5.7) to get the current estimation $\mathbf{x}_1$. At each step, a new parameter vector can be created by recursively applying the update rule (4.16) to (5.7). The learning process is repeated until some termination criteria are met, such as until the fitting error does not decrease much or the maximum number of iterations is reached. The pseudocode for training a sequence of update maps is shown in Alg.2.

Algorithm 2 Training a sequence of update maps

Input $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{h}_0^i, \mathbf{M}_i\right)\right\}_{i=1}^N$, T, λ, c
Output $\{\mathbf{D}_{t+1}\}_{t=0}^{T-1}$

for $t = 0$ to $T-1$ **do**
  Compute $\mathbf{W}_t$ with (4.8), (4.9), (4.10), (4.11), (4.12), (4.13)
  Compute $\mathbf{D}_{t+1}$ with (4.16)
  for $i = 1$ to N **do**
    Update $\mathbf{x}_{t+1}^i := \mathbf{x}_t^i - \mathbf{D}_{t+1} \mathbf{h}_t^i$
  end for
end for

4.2.6 Convergence of Training Error

Theorem 4.2.1 (Convergence of RDO's training error) Given training set $\left\{\left(x_0^i, x_*^i, \mathbf{h}_0^i\right)\right\}_{i=1}^N$, if there exists a linear map $\hat{D} \in \mathbf{R}^{p\times f}$ where $\hat{D}\mathbf{h}_t^i$ meets the condition $\sum_{i=1}^N \left(x_t^i - x_*^i\right)^{\mathrm{T}} \hat{D}\mathbf{h}_t^i > 0$ at x_*^i for all i where $\left[\mathbf{h}_t^i\right]_{j,:} \in (0,1)$, and if there exists an i where $\mathbf{x}_t^i \neq \mathbf{x}_*^i$, then the update rule:

$$\mathbf{D}_{t+1} = \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_*^i - \mathbf{x}_t^i + \hat{\mathbf{D}}\mathbf{h}_t^i\right) \right\|_2^2 + \lambda \left\|\hat{\mathbf{D}}\right\|_F^2 \tag{4.17}$$

where $\left[\mathbf{W}_t\right]_{j,j} > 1$

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i - \mathbf{D}_{t+1}\mathbf{h}_t^i \tag{4.18}$$

guarantees that the training error strictly decreases in each iteration:

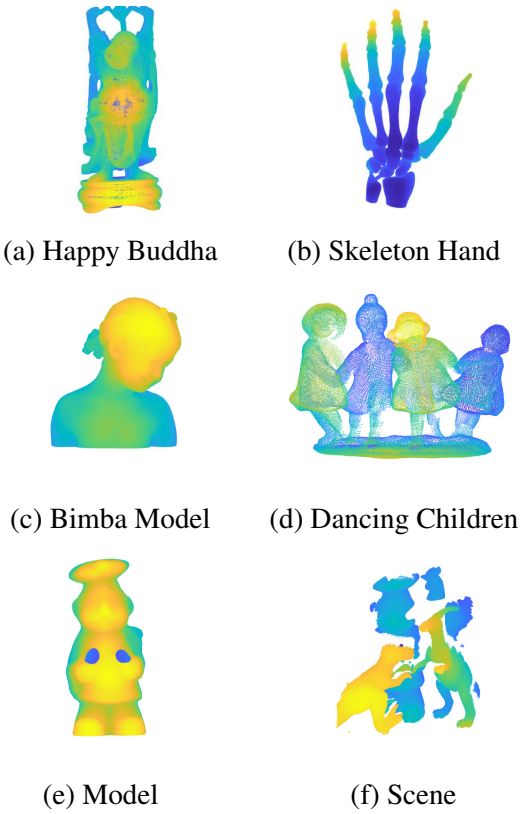
$$\sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_{t+1}^i\right\|_2^2 < \sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2 \tag{4.19}$$

If $\hat{\mathbf{D}}\mathbf{h}_t^i$ is strongly monotone at $\mathbf{x}_*^i$, and if there exist $H > 0$, $M > 0$ such that $\left\|\hat{\mathbf{D}}\mathbf{h}_t^i\right\|_2^2 \leq H + M\left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2$ for all i, then the training error converges to zero.

Thm.4.2.1 says that for all samples , if $\hat{D}\mathbf{h}_t^i$ meets the condition $\sum_{i=1}^N \left(x_t^i - x_*^i\right)^{\mathrm{T}} \hat{D}\mathbf{h}_t^i > 0$, then the average training error will decrease in each iteration; if $\hat{D}\mathbf{h}_t^i$ is strongly monotone at x_*^i, the average training error will converge to zero. [11] also presents a similar convergence result for an update rule, but it requires $\hat{D}\mathbf{h}_t^i$to be strictly monotone at x_*^i for all i.The detailed proof for convergence is provided in Appendix A.

4.3 Experimentation

This section describes how to apply RDO to 3D point cloud registration (Synthetic data and Range-scan data) and stitching. We provide a comparative study of RDO with other classical registration methods on different data sets. The data sets are shown as Figure.4.7, Figure.4.8 and Figure.4.9.

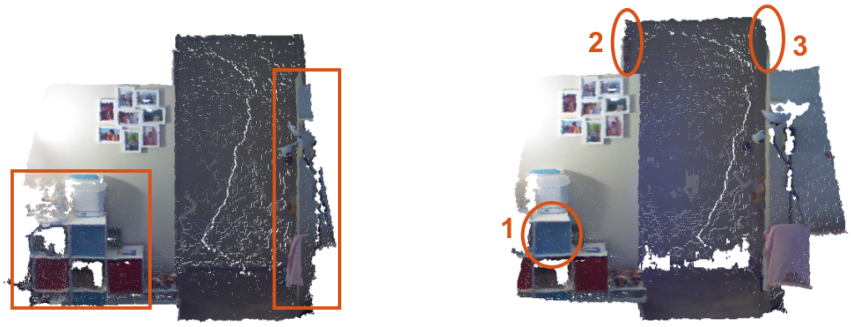


(a) Happy Buddha (b) Skeleton Hand

(c) Bimba Model (d) Dancing Children

(e) Model (f) Scene

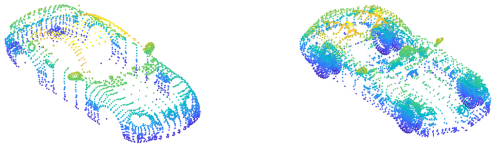
Fig. 4.7 3D registration data sets. $(a) \sim (d)$ are the Synthetic data, and $(e), (f)$ are the Range-scan data



(a) ptCloudRef (b) ptCloudCurrent

Fig. 4.8 Point Cloud stitching experiments data sets. The rectangles show the major differences between two different views. The ellipses marked by 1,2,3 show the obvious differences in stitching experimental results.

(a) Airplane0001 (b) Airplane0144



(c) Car0007 (d) Car0026

Fig. 4.9 3D point clouds for comparing the PointnetLK and RDO registration methods.

4.3.1 Experimental Design

Data sets

We performed 3D points cloud registration experiments on synthetic data sets: Happy Buddha model [13], Skeleton Hand model [18], Dancing Children and Bimba Model http://visionair.ge.imati.cnr/ (as shown in $(a) \sim (d)$ of Figure.4.7). And the registration experiment on Range-scan data is conducted on the UWA data set [105] (as shown in $(e), (f)$ of Figure.4.7). Multi-view points clouds for the stitching experiment are as shown in Figure.4.8. We also compare the performance of RDO and the advanced deep-learning-based registration method PointnetLK [17] on the ModelNet40 data set [16], as shown in Figure.4.9.

Design h

Let $\mathbf{M} \in \mathbf{R}^{3\times N_M}$ be a matrix containing the 3D coordinates of the original model and $\mathbf{S} \in \mathbf{R}^{3\times N_S}$ for the scene model. Our method is applied to register $\mathbf{S}$ and $\mathbf{M}$, where the transformation matrix is represented by Lie algebra $\mathfrak{se}(3)$ $\mathbf{x} \in \mathbf{R}^6$. We use the feature extraction method in [11] to extract the feature of the data sets, which designs $\mathbf{h}$ to be a histogram indicating the weights of scene points on the 'front' and the 'back' sides of each model point according to the coordinates in Figure.4.10.

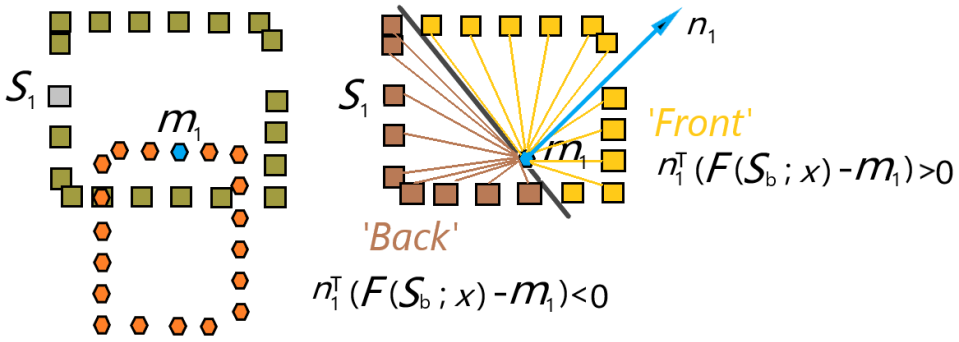


Fig. 4.10 The positional relationship between scene points (square) and model point (hexagon) $\mathbf{m}_1$.

$$\mathbf{S}_a^+ = \left\{ \mathbf{s}_b : \mathbf{n}_a^T \left(F\left(\mathbf{s}_b; \mathbf{x}\right) - \mathbf{m}_a \right) > 0 \right\} \tag{4.20}$$

$\mathbf{S}_a^+$ indicates the set of scene points on the 'front' of model point $\mathbf{m}_a$, and $\mathbf{S}_a^-$ contains the remaining scene points; $\mathbf{n}_a \in \mathbf{R}^3$ is the normal vector of the model point $\mathbf{m}_a$; $F(\mathbf{s}_b; \mathbf{x})$ is the function that applies rigid transformation with parameter $\mathbf{x}$ to scene point $\mathbf{s}_b$. Then the features in different divided areas of the model point $\mathbf{m}_a$ can be calculated through the following formulas:

$$\left[\mathbf{h}(\mathbf{x}; \mathbf{S})\right]_{a^+} = \frac{1}{z} \sum_{\mathbf{s}_b \in \mathbf{S}_a^+} \exp\left(\frac{-1}{\beta^2} \left\| F(\mathbf{s}_b; \mathbf{x}) - \mathbf{m}_a \right\|^2 \right) \tag{4.21}$$

$$\left[\mathbf{h}(\mathbf{x}; \mathbf{S})\right]_{a^-} = \frac{1}{z} \sum_{\mathbf{s}_b \in \mathbf{S}_a^-} \exp\left(\frac{-1}{\beta^2} \left\| F(\mathbf{s}_b; \mathbf{x}) - \mathbf{m}_a \right\|^2 \right) \tag{4.22}$$

z normalises $\mathbf{h}$ to sum to 1, and β controls the width of the exp function. $\mathbf{h}$ is specific to model $\mathbf{M}$, and it has a fixed size $2\mathbf{N}_M$.

4.3.2 RDO Training

The parameters in the RDO training process are the same as those in the code provided in the Github of DO [11]. We normalise a given model shape $\mathbf{M}$ to $[-1,1]^3$ and uniformly sample from $\mathbf{M}$ with the replacement 400 to 700 points to generate a scene model. Then we apply the following perturbations to the scene model: *(i) Rotation and translation:* The rotation is within 60 degrees, and the translation is in $[-0.3, 0.3]^3$, which represents ground truth $\mathbf{x}_*$; *(ii) Noise and Outliers:* Gaussian noise with the standard deviation 0.05 is added to the scene model. 0 to 300 points within $[-1.5, 1.5]^3$ are added as the sparse outliers. Besides, a Gaussian ball of 0 to 200 points with the standard deviation of 0.1 to 0.25 is used to simulate the structured outliers; *(iii) incomplete shape:* We remove 40% to 90% points from the scene model to simulate occlusions, the detailed removing approach

can be found in [11]. For all experiments, we generated 30000 training samples, set up iterations $T = 30$ and set λ as 2×10^{-4}, β^2 as 0.03, and the initial transformation $\mathbf{x}_0$ is $\mathbf{0}^6$.

4.3.3 Experiments Metrics

Baselines

We compared RDO with the advanced learning-based approach DO [11] and deep-learning-based method PointnetLK [17], two point-based approaches (ICP [31] and IRLS [33]) and three density-based approaches (CPD [68], BCPD [99] and NDT [61]). The codes for all methods were downloaded from the authors' websites, except for ICP, where we used MATLAB's implementation. For IRLS, the Huber cost function was used. The code of RDO was implemented in MATLAB.

Evaluation Metrics

We use the registration success rate, the average MSE, and the computation time as performance metrics for comparing the performance of RDO with registration methods (DO, ICP, NDT, CPD, BCPD, and IRLS). And registration success rate, the MSE, and the log_{10} average MSE are used for RDO and PoinnetLK comparison.

Registration Success Rate. Registration is successful when the mean ℓ_2 is less than 0.05 of the model's largest dimension.

Average MSE. It is worth noting that the MSE is the mean ℓ_2 error between the model and the scene sets, and the Average MSE is the average for MSE for all test sets.

4.3.4 Parameters Settings

The maximum number of iterations of all registration methods was set to 30. For *DO* and *RDO*, we set λ as 2×10^{-4}, β^2 as 0.03. The value of the tolerance of absolute difference between current estimation and ground truth in iterations is 1e-4; For *ICP*, the tolerance of absolute difference in translation and rotation is 0.01 and 0.5, respectively; For *IRLS*, we used *Huber criterion function* as the regression function, the remaining parameters were set as the same as the setting of *ICP*. For *CPD*, the type of transformation is set to *rigid*, and the expected percentage of outliers with respect to a normal distribution is 0.1; the tolerance value is the same as that in *DO*. For *NDT*, the value of the expected percentage of outliers is set to 0.55, and the tolerance value is set as the same as that in *ICP*. For *BCPD*, the expected percentage of outliers is set to 0.3, the parameter in the Gaussian kernel is 0.2, and the expected length of the displacement vector is 1e9. For *PointnetLK*, the PointnetLK network is trained on an Nvidia Geforce 2080Ti GPU with 12G memory. The kernel sizes of the PointnetLK are 64, 64, 64, 128, 1024. The maximum iteration for rotation and translation is set to 30. Adam optimiser with an initial learning rate of 0.001,

250 epochs and a batch size of 10 are used for the training process. Please note that the values of these parameters are the default values set in their official codes.

4.3.5 Registration Experiments

Synthetic Data

We perform registration experiments on Happy Buddha model, Skeleton Hand model, Dancing Children and Bimba Model in Figure.4.7. The models are downsampled by selecting 477 points from the original model as the model set $\mathbf{M}$. The performance of algorithms are evaluated by comparing the evaluation metrics in the case of various perturbations: *(1) rotation:* The initial angle is 0°, 30°, 60°, 90°, 120° and 150° [default=0° to 60°]; *(2) noise:* The standard deviation of Gaussian noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]; *(3) outlier:* We set the number of outliers to 0, 100, 200, 300, 400 and 500, respectively [default=0]; *(4) incomplete ratio:* The ratio of incomplete scene shape is set to 0, 0.12, 0.243, 0.36, 0.48 and 0.6 [default=0]. The random translation of all generated scenes is within $[-0.3, 0.3]^3$. It is worth noting that when we change one parameter, the values of other parameters are fixed to the default value. In addition, the scene points are sampled from the original model, not from $\mathbf{M}$. We will test 750 test samples in each variable setting.

Range-scan Data

We perform a registration experiment on the UWA data set in Figure.4.7. This data set contains 50 cluttered scenes with five objects taken with the Minolta Vivid 910 scanner in various configurations. All objects are heavily occluded (60% to 90%). From the original model, ~ 400 points were sampled using *pcdownsample* and used as model $\mathbf{M}$. We also downsampled the scene to ~ 1000 points. We initialised the model from 0 to 60 degrees from the ground truth orientation with random translation within $[-0.3, 0.3]^3$ (points can move within $[-0.3, 0.3]$ on each dimension). We ran 75 initialisation for each parameter setting, and we set the inlier ratio of ICP to 50% as an estimate for self-occlusion.

ModelNet40 Data set

We perform registration experiments on ModelNet40 Data set in Figure.4.9. For *PointnetLK training*, We train the PointnetLK network on 20 categories of public ModelNet40 data set, and all 3D point clouds are downsampled to 1024 points during training. For *RDO training*, there are two training schemes: single-class training and multi-class training. The former is to train RDO on each point cloud (such as Airplane0001), and the latter is to train RDO on all data sets (four point clouds) in Figure.4.9. The following perturbation setting is adaptable for RDO training (single-class and multi-class) and PointnetLK training. There are three kinds of perturbation setting modes. (i)$mode_1$: The rotation is within 45 degrees and the translations is in $[-0.3, 0.3]^3$; (ii) $mode_2$: The rotation is within 60 degrees and the translations is in $[-0.3, 0.3]^3$; (iii) $mode_3$: The rotation is within 45 degrees, the

translations is in $[-0.3, 0.3]^3$ and Gaussian noise with the standard deviation 0.05 is also applied.

The performance of algorithms are evaluated by comparing the evaluation metrics in the case of various perturbations: for $mode_1$: the initial angle is 0°, 15°, 30°, 45°, 60° and 75°; for $mode_2$: the initial angle is 0°, 30°, 60°, 90°, 120° and 150°; for $mode_3$: the initial angle is 0°, 15°, 30°, 45°, 60° and 75° [default=0° to 45°] and the standard deviation of Gaussian noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]. It is worth noting that when we change one parameter, the values of other parameters are fixed to the default value. We will test 100 test samples in each variable setting.

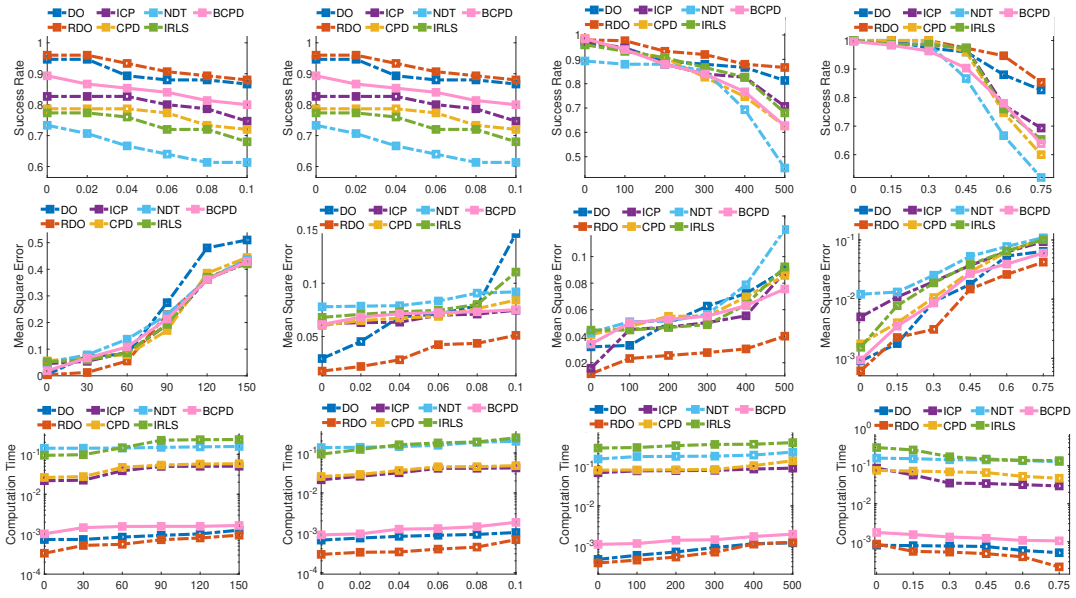
4.3.6 Stitching Experiments

We perform a multi-view points cloud stitching experiment on the data set in Figure.4.8, which stitches together a collection of point clouds that were captured with a Kinect to construct a larger 3D view of the scene. To align the two point clouds, we regard the first point cloud as the reference and apply the transformation parameters to the second point cloud. ~ 400 points of the reference model were sampled using *pcdownsample* and used as the model $\mathbf{M}$. We also downsampled the second point cloud to ~ 1000 points. We initialise the reference model from 0 to 15 degrees with random translations within $[-0.1, 0.1]^3$. It is worth noting that after attaining the estimated parameters, we use it to transform the second point cloud to the reference coordinate system defined by the first point cloud.

4.3.7 Experimental Results and Discussion

Registration Results

Figure.4.11, Figure.4.12 represent the 3D registration results on Happy Buddha model and Skeleton Hand model with various perturbations through Bar graphs. Top is the Success Rate (SR). Middle is the Average MSE (AMSE). Bottom is the Computation Time. It can be seen that the learning-based registration algorithms (DO, RDO) have better performance than the traditional registration methods (ICP, CPD, BCPD, NDT, IRLS) on all evaluation metrics, i.e., Successful Registration Rate, Average MSE, and Computation Time. Specifically, in the presence of various perturbations, the Successful Registration Rate of RDO is higher and more stable compared with other algorithms. And the values of Average MSE illustrate that RDO can achieve more accurate and stable registration on different data sets (Happy Buddha and Skeleton Hand) than DO. While ICP required less computation time in all cases, it showed low success rates in high perturbations cases, and the performance of CPD was similar to that of ICP. As another statistic-based algorithm, NDT was more time-consuming and had higher registration errors and lower success rates than CPD. BCPD had better performance and took less time dealing with registration under all kinds of rotation and noise. However, when there were different degrees of outliers and occlusion, the performance of the algorithm was not ideal. IRLS required a high computation time and did not perform well when the model was highly incomplete.



(a) Initial Angles (b) Noise SD (c) Outlier (d) Incompleteness

Fig. 4.11 Results of 3D registration with Happy model under different perturbations.

Table.4.1 shows the quantitative results of the registration on the Happy model. B means the baseline method - Discriminative Optimisation method (DO); P is the proposed method in this chapter - Reweighted Discriminative Optimisation method (RDO); C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift method (BCPD). We set the Successful Rate of DO, Mean Square Error of DO, and the Computation Time of DO as the references of comparison. The value of the Successful Rate is higher, the stability of the method is higher; the value of the Mean Square Error is lower, the registration accuracy is higher; the shorter the computation time, the real-time capability is better. We can find that the Successful Rate of RDO is 12.62% higher than that of DO, and the registration accuracy of RDO is almost 45.56% higher than that of DO. The computation time of RDO is 21.73% lower than that of DO. By contrast, the successful rate and registration accuracy of BCPD are not as good as DO, and BCPD takes more time to achieve registration.

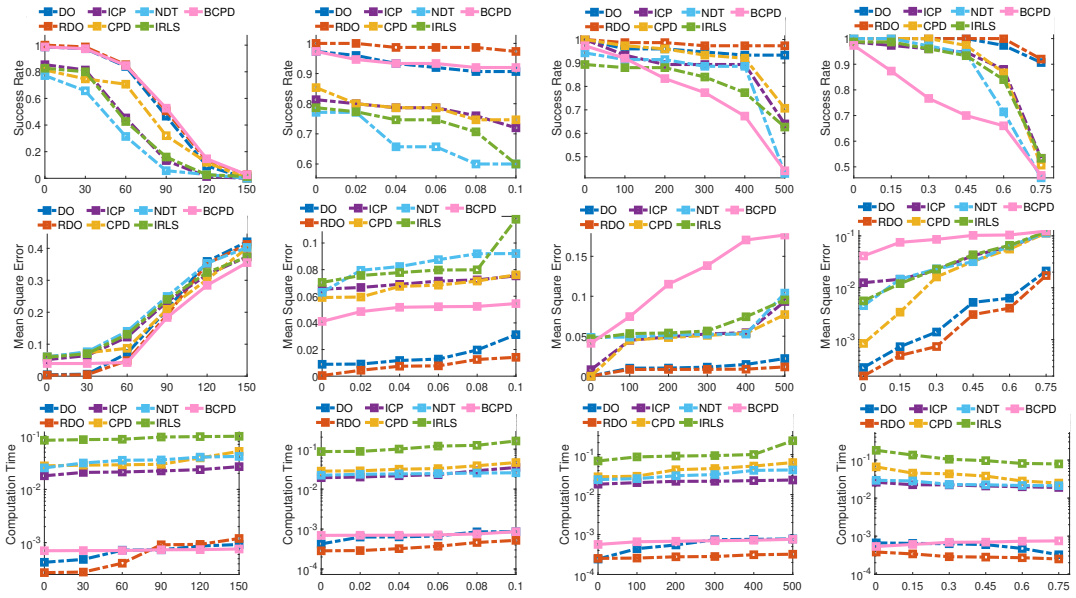
Table.4.2 shows the quantitative results of the registration on the Hand model. C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift method (BCPD). We can find that the Successful Rate of RDO is 5.31% higher than that of DO, and the registration accuracy of RDO is almost 38.38% higher than that of DO. The computation time of RDO is 24.31% lower than that of DO. BCPD takes 2.3 times as much time as RDO.

Figure.4.13 illustrates that the better performance of RDO in dealing with the registration with 60^o rotation compared with other algorithms. Different colours show the registration results of different registration algorithms. The rectangles illustrate the obvious difference between the registration results.

Figure.4.14 and Figure.4.15 show the statistical registration results on the Bimba model and Dancing children model with various perturbations by Box plots. Each column shows the registration results in the presence of various perturbations with different degrees. The

Table 4.1 The quantitative results of the registration on Happy model (B-Baseline DO; P-Proposed RDO; C-Conventional method BCPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.01**	0.88	1.00	**0.50**	1.00	1.00	**0.71**	2.86
(R)90	1.00	**1.13**	1.11	1.00	**0.76**	0.80	1.00	**0.80**	2.67
(R)120	1.00	**2.22**	1.00	1.00	**0.75**	0.75	1.00	**0.79**	2.56
(R)150	1.00	**1.25**	0.63	1.00	**0.80**	0.80	1.00	**0.91**	2.22
(N)0.04	1.00	**1.03**	0.95	1.00	**0.46**	1.04	1.00	**0.71**	1.43
(N)0.06	1.00	**1.02**	0.93	1.00	**0.40**	0.88	1.00	**0.86**	1.43
(N)0.08	1.00	**1.03**	0.92	1.00	**0.37**	0.85	1.00	**0.90**	2.31
(N)0.10	1.00	**1.03**	0.92	1.00	**0.34**	0.80	1.00	**0.90**	2.50
(O)200	1.00	**1.03**	0.97	1.00	**0.46**	1.04	1.00	**0.80**	2.00
(O)300	1.00	**1.03**	0.95	1.00	**0.40**	0.88	1.00	**0.75**	1.25
(O)400	1.00	**1.01**	0.87	1.00	**0.40**	0.92	1.00	**0.99**	1.75
(O)500	1.00	**1.06**	0.72	1.00	**0.43**	0.94	**1.00**	1.01	2.47
(I)0.30	1.00	**1.03**	0.97	1.00	**0.54**	1.00	1.00	**0.80**	1.00
(I)0.45	1.00	**1.02**	0.94	1.00	**0.90**	2.00	1.00	**0.75**	1.12
(I)0.60	1.00	**1.07**	0.89	1.00	**0.50**	0.75	1.00	**0.64**	1.03
(I)0.75	1.00	**1.05**	0.81	1.00	**0.70**	0.93	1.00	**0.43**	1.14

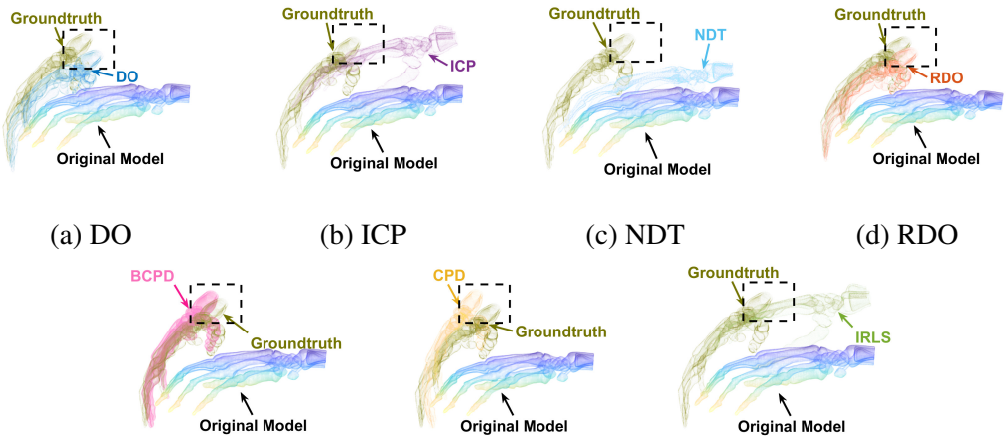


(a) Initial Angles (b) Noise SD (c) Outlier (d) Incompleteness

Fig. 4.12 Results of 3D registration with Hand model under different perturbations.

Table 4.2 The quantitative results of the registration on Hand model (B-Baseline DO; P-Proposed RDO; C-Conventional method BCPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)60	1.00	**1.05**	0.97	1.00	**0.67**	0.94	1.00	**0.85**	1.79
(R)90	1.00	**1.10**	0.96	1.00	**0.60**	1.23	1.00	**0.86**	1.55
(R)120	1.00	**1.01**	0.72	1.00	**0.50**	0.97	1.00	**0.60**	1.95
(R)150	1.00	**1.04**	0.71	1.00	**0.67**	1.04	1.00	**0.85**	1.94
(N)0.04	1.00	**1.01**	0.83	1.00	**0.50**	0.81	1.00	**0.89**	1.91
(N)0.06	1.00	**1.10**	0.98	1.00	**0.44**	0.98	1.00	**0.73**	1.63
(N)0.08	1.00	**1.00**	0.93	1.00	**0.69**	0.90	1.00	**0.64**	1.80
(N)0.10	1.00	**1.08**	0.89	1.00	**0.74**	1.07	1.00	**0.70**	1.51
(O)200	1.00	**1.08**	0.75	1.00	**0.65**	1.08	1.00	**0.84**	1.71
(O)300	1.00	**1.09**	0.95	1.00	**0.64**	0.89	1.00	**0.79**	1.66
(O)400	1.00	**1.01**	0.99	1.00	**0.59**	1.26	1.00	**0.76**	1.58
(O)500	1.00	**1.04**	0.95	1.00	**0.55**	1.23	1.00	**0.61**	1.59
(I)0.30	1.00	**1.03**	0.81	1.00	**0.76**	1.21	1.00	**0.85**	1.71
(I)0.45	1.00	**1.08**	0.78	1.00	**0.67**	1.10	1.00	**0.64**	1.54
(I)0.60	1.00	**1.04**	0.81	1.00	**0.40**	0.88	1.00	**0.71**	1.80
(I)0.75	1.00	**1.09**	0.86	1.00	**0.79**	0.90	1.00	**0.79**	1.74

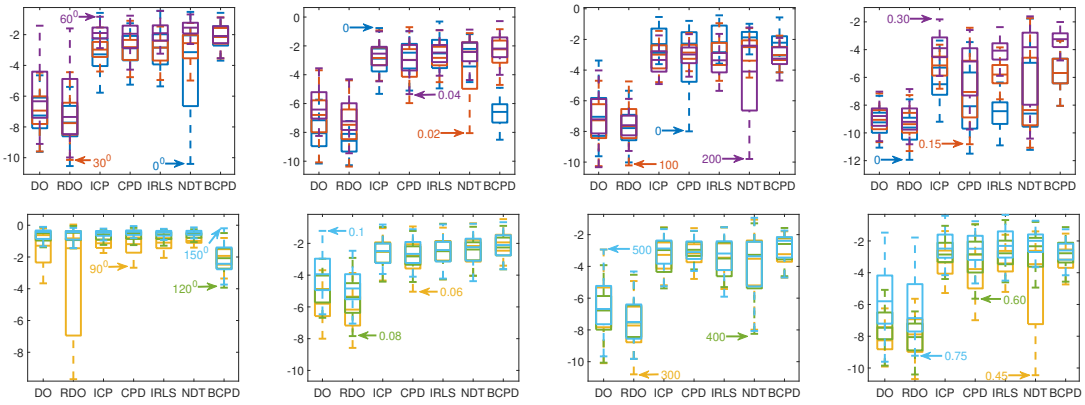


(a) DO (b) ICP (c) NDT (d) RDO

(e) BCPD (f) CPD (g) IRLS

Fig. 4.13 Registration results of Hand model with 60^o rotation.

X-axis represents different registration algorithms. The y-axis shows the $\log_{10}$ value of the registration error of samples. The values with different colours represent different degrees of perturbations. The $\log_{10}$ value of the registration error corresponding to the inter-quartile ranges in Box Plots illustrates that the learning-based algorithms (DO and RDO) are more robust than the traditional algorithms (ICP, CPD, BCPD, NDT, IRLS). ICP, IRLS, and BCPD did not perform well when the model was highly incomplete. And CPD and BCPD were less able to deal with registration with outliers, especially when the number of outliers was high. DO and RDO outperformed the baselines in almost all test scenarios. However, RDO achieved more accurate registration than DO as shown in the positions of minimum, maximum, quartiles, and the skewness in Box plots.



(a) Initial Angles (b) Noise SD (c) Outliers (d) Incompleteness

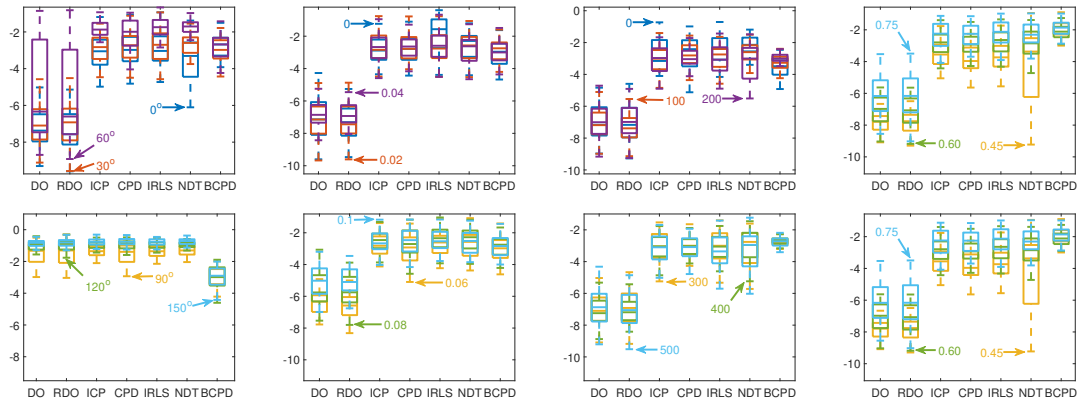
Fig. 4.14 Statistical results of 3D registration with the Bimba Model under different perturbations.

Table.4.3 shows the log_{10} Mean Square Error of registration results on Bimba model. B means the baseline method - Discriminative Optimisation method (DO); P is the proposed method in this chapter - Reweighted Discriminative Optimisation method (RDO); C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift (BCPD). We set the absolute value of log_{10} Mean Square Error of DO as the reference of comparison. The value of the absolute value of the log_{10} Mean Square Error is higher, the registration accuracy is higher. Q_0, Q_4 and IQR aim to measure data distribution in boxplot figures. Q_0 and Q_4 are the minimum and maximum respectively. And IQR is the interquartile range, which measures the distance between the upper and lower quartiles; the shorter the distance, the more stability of the method. It can be seen that RDO has higher accuracy than DO, as shown the bold in this table. Compared with RDO and DO, BCPD has higher stability.

Table.4.4 shows the log_{10} Mean Square Error of registration results on Dancing Children model. C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift (BCPD). It can be seen that BCPD has higher accuracy than RDO and DO on the registration with larger rotations. In the case of registration with other perturbations (Noises, Outliers and Occlusions), the registration accuracy of RDO is 2.11% higher than that of DO and almost twice that of BCPD.

Table 4.3 The quantitative registration results on Bimba model (B-Baseline DO; P-Proposed RDO; C-Conventional method BCPD; Q_0 - Minimum; Q_4 - Maximum; IQR - Interquartile range)

	Q_0			Q_4			IQR		
	B	P	C	B	P	C	B	P	C
(R)90	1.00	**2.21**	1.05	1.00	0.90	**10.00**	1.00	3.14	**0.64**
(R)120	1.00	0.92	1.58	1.00	1.00	**5.00**	1.00	**1.00**	1.88
(R)150	1.00	0.92	1.50	1.00	**3.00**	1.00	1.00	**1.00**	1.99
(N)0.06	1.00	**1.06**	0.48	1.00	**1.05**	0.13	1.00	0.60	**0.16**
(N)0.08	1.00	**1.23**	0.62	1.00	**1.07**	0.36	1.00	1.00	**0.17**
(N)0.10	1.00	**1.03**	0.59	1.00	**3.50**	0.70	1.00	0.85	**0.50**
(O)300	1.00	**1.10**	0.48	1.00	**1.79**	0.64	1.00	0.77	**0.65**
(O)400	1.00	**1.00**	0.48	1.00	**1.20**	0.50	1.00	0.65	**0.62**
(O)500	1.00	**1.04**	0.50	1.00	**1.50**	0.71	1.00	**0.54**	0.57
(I)0.45	1.00	**1.10**	0.48	1.00	0.94	0.27	1.00	1.67	0.83
(I)0.60	1.00	**1.05**	0.48	1.00	**1.30**	0.34	1.00	0.63	**0.63**
(I)0.75	1.00	**1.00**	0.41	1.00	**1.18**	1.18	1.00	1.07	**0.34**



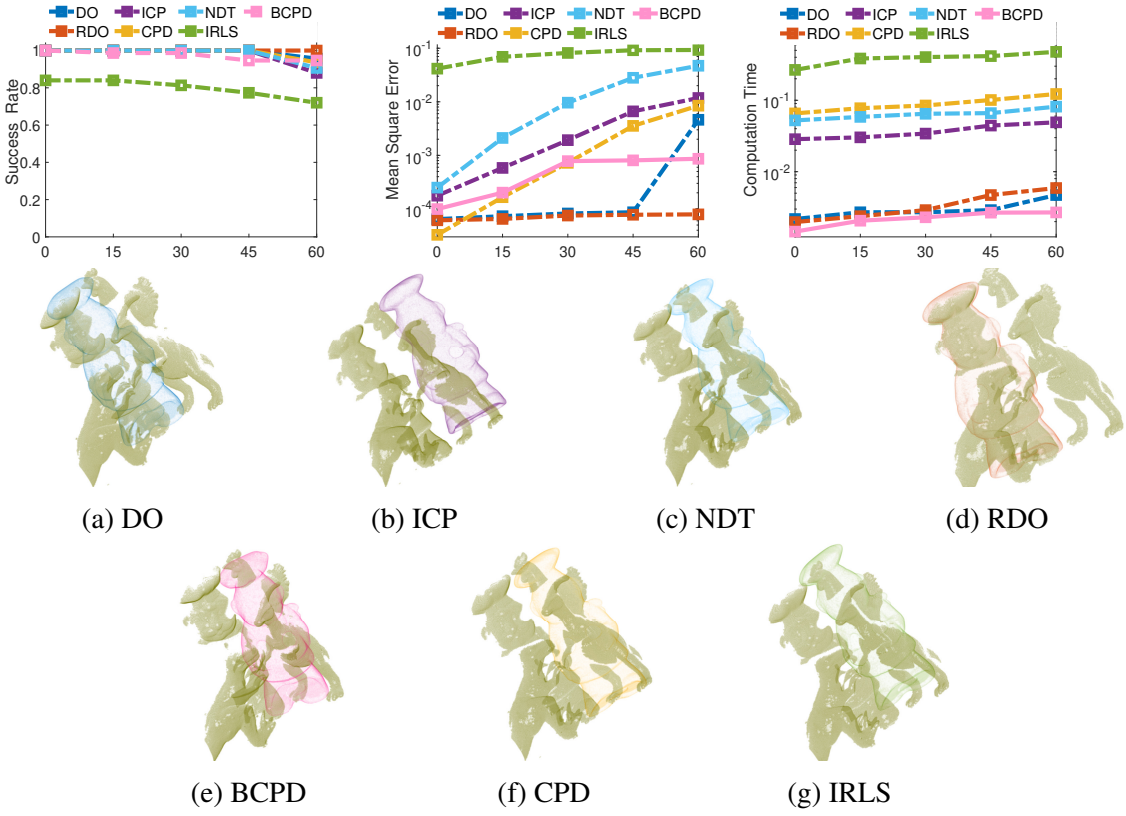
(a) Initial Angles (b) Noise SD (c) Outlier (d) Incompleteness

Fig. 4.15 Statistical results of 3D registration with Dancing Children model under different perturbations.

Table 4.4 The quantitative registration results on Dancing Children model (B-Baseline DO; P-Proposed RDO; C-Conventional method BCPD; Q_0 - Minimum; Q_4 - Maximum; IQR - Interquartile range)

	Q_0			Q_4			IQR		
	B	P	C	B	P	C	B	P	C
(R)90	1.00	1.00	**1.50**	1.00	0.50	**5.00**	1.00	**1.08**	1.16
(R)120	1.00	1.11	**2.78**	1.00	0.50	**5.00**	1.00	**1.00**	3.50
(R)150	1.00	1.00	**4.00**	1.00	0.50	**5.00**	1.00	**1.00**	7.00
(N)0.06	1.00	**1.05**	0.58	1.00	**1.04**	0.40	1.00	1.20	1.40
(N)0.08	1.00	**1.03**	0.54	1.00	**1.33**	0.67	1.00	0.82	**0.71**
(N)0.10	1.00	**0.97**	0.54	1.00	**1.27**	0.60	1.00	0.88	**0.75**
(O)300	1.00	**1.00**	0.40	1.00	**0.90**	0.44	1.00	1.25	**0.38**
(O)400	1.00	**0.94**	0.40	1.00	**1.00**	0.42	1.00	0.87	**0.40**
(O)500	1.00	**1.04**	0.39	1.00	**1.14**	0.52	1.00	0.94	**0.33**
(I)0.45	1.00	**1.04**	0.36	1.00	1.00	0.22	1.00	1.33	**0.33**
(I)0.60	1.00	**1.04**	0.33	1.00	**1.00**	0.31	1.00	0.93	**0.40**
(I)0.75	1.00	**1.08**	0.35	1.00	**1.00**	0.33	1.00	1.19	**0.19**

Figure.4.16 shows the results of the registration with the UWA data set. The first row shows the evaluation results of the performance of various methods on the UWA data set . And the second and the third rows display the registration results of the UWA data set with 60^o rotation; different colours show the registration results of different registration algorithms. While DO and RDO have outperformed other traditional registration algorithms in terms of the Success Rate, Average MSE, and Computation Errors, RDO has shown improvements over DO and maintained low computation time. RDO produces more accurate results than DO when registering point clouds with large rotations.



(a) DO (b) ICP (c) NDT (d) RDO

(e) BCPD (f) CPD (g) IRLS

Fig. 4.16 Results of the registration on the UWA data set .

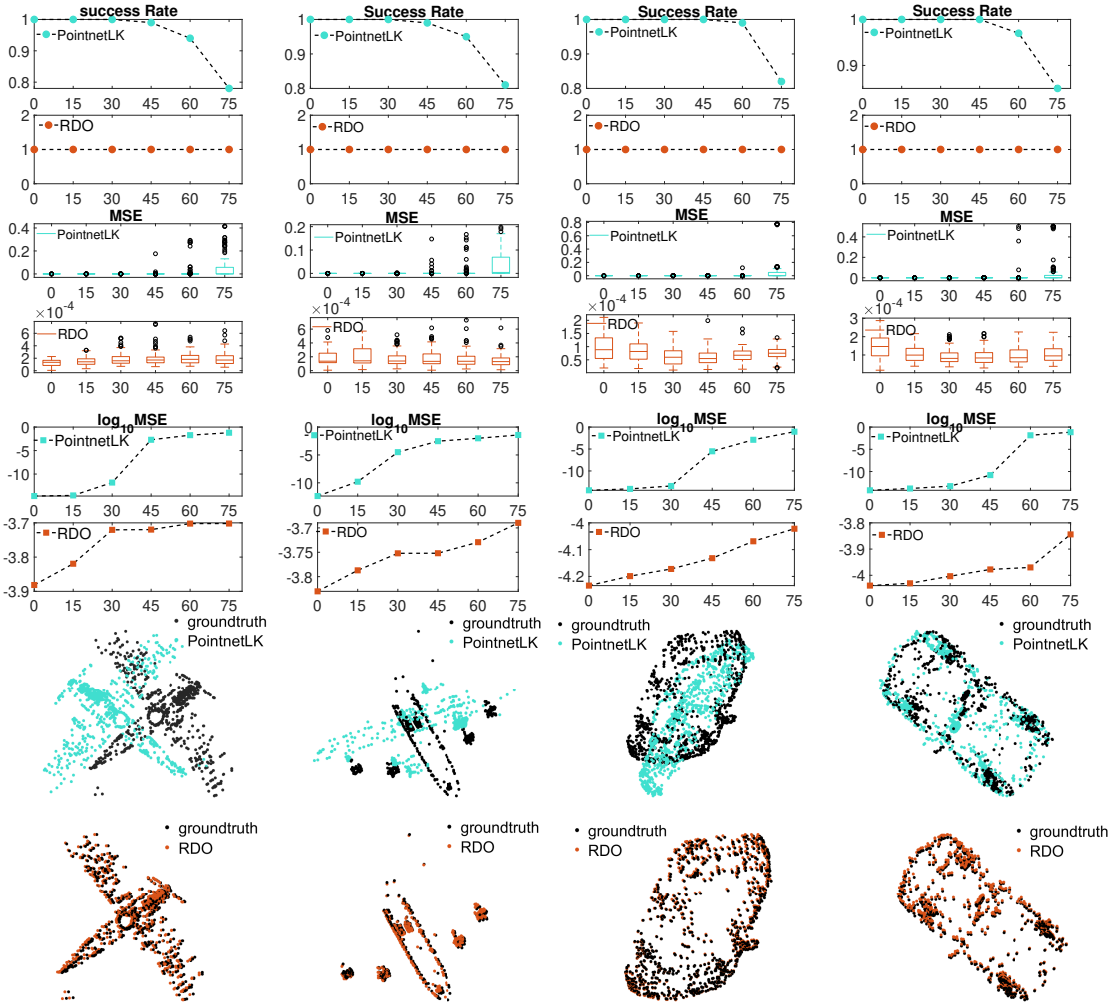
Table.4.5 shows the quantitative results of the registration on the UWA data set . C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift method (BCPD). We can find that RDO has higher registration accuracy and successful registration rate than DO and BCPD.

Table 4.5 The quantitative registration results on UWA data set (B-Baseline DO; P-Proposed RDO; C-Conventional method BCPD)

	Successful Rate			Mean Square Error			Computation Time		
	B	P	C	B	P	C	B	P	C
(R)30	1.00	**1.00**	1.00	1.00	**1.00**	10.00	1.00	**1.00**	0.95
(R)45	1.00	**1.00**	0.92	1.00	**1.00**	10.00	1.00	1.25	**1.00**
(R)60	1.00	**1.09**	1.00	1.00	**0.02**	0.23	1.00	1.03	**0.57**

Figure.4.17, Figure.4.18, Figure.4.19 and Figure.4.20 illustrate the registration results of RDO and PointnetLK on ModelNet40 with the single-class training scheme. Top shows the Success Rate. The second shows the Mean Square Error (MSE). The third shows the log_{10} average MSE. Bottom displays the registration results of ModelNet40 with 60^o rotation, and the turquoise shows the registration result of PointnetLK, and the dark orange illustrates the registration results of RDO. The first two correspond to the evaluation for perturbation setting modes $mode_1$ and $mode_2$, and the last two show the registration results corresponding to the perturbation setting mode $mode_3$. Figure.4.17 shows the registration results when rotation is within 45^o during the training process. It can be seen that, compared with PointnetLK, RDO had a higher and more stable *Success Rate* and

lower *Mean Square Error (MSE)*. The number of outliers in box-plots of *MSE* illustrates that the performance of PointnetLK is not stable when dealing with the registration with rotation degree over 45^o, which also can be verified by the registration accuracy showed in log_{10} *average MSE*.



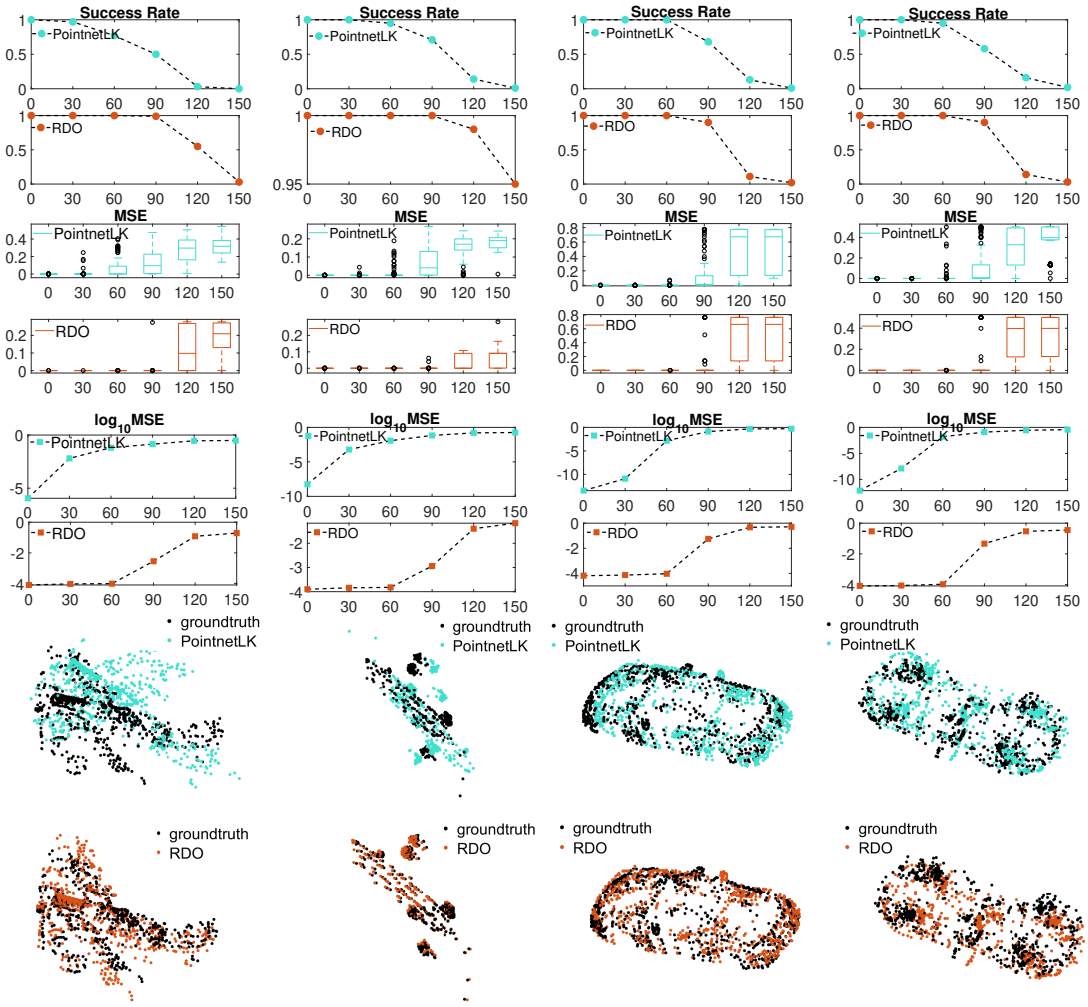
(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.17 The registration results of the single-class training scheme with perturbation setting $mode_1$.

Figure.4.18 displays the registration results with the training rotation within 60^o. It can be seen that PointnetLK did not perform well when registering the point cloud with a larger rotation, even it had low registration accuracy when dealing with registration with 60^o.

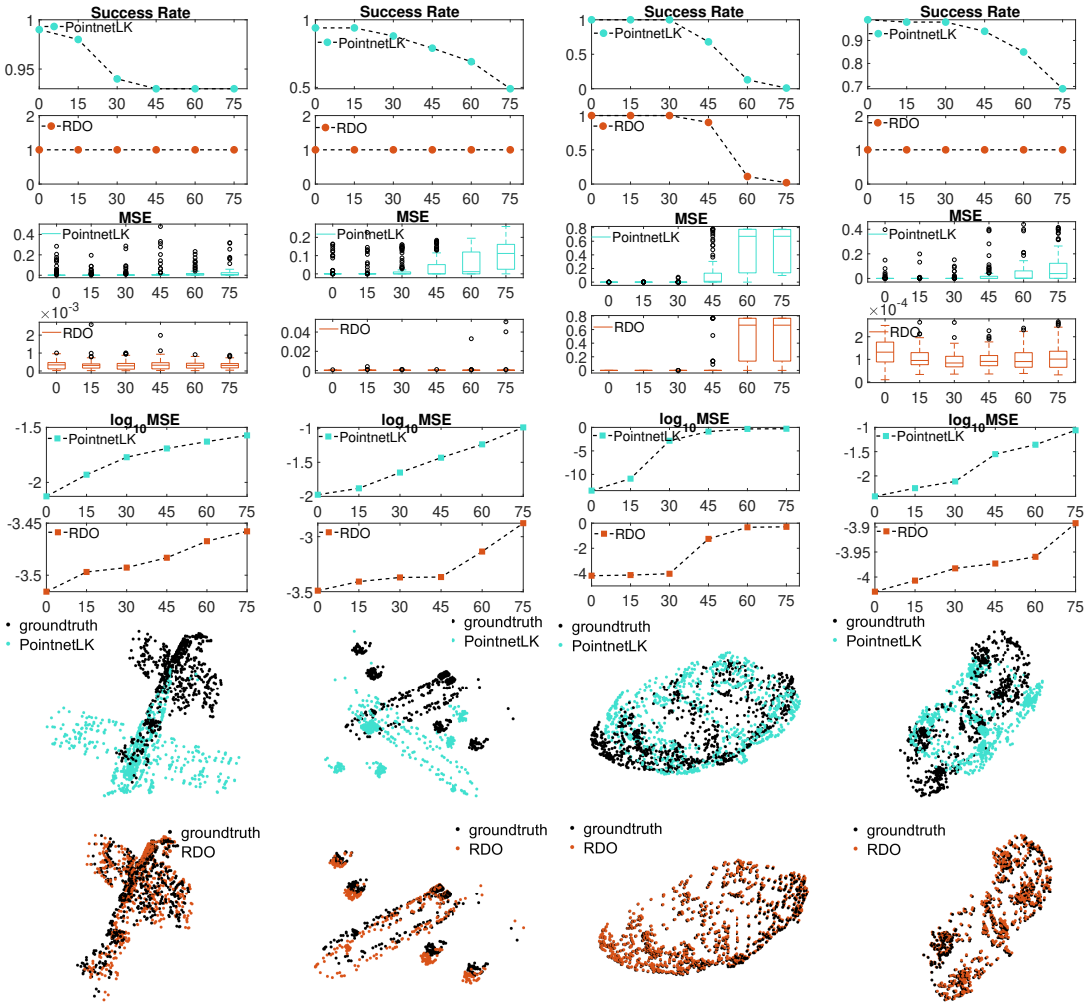
Figure.4.19 and Figure.4.20 illustrate the registration results when the rotation is within 45^o and the standard deviation of Gaussian noise is 0.05 during the training process. Figure.4.19 also exposes the low robustness and instability of the PointnetLK method when registering point clouds with rotation degrees over 45^o.

Figure.4.20 shows that PointnetLK did not perform well when dealing with registration with various degrees of noise.



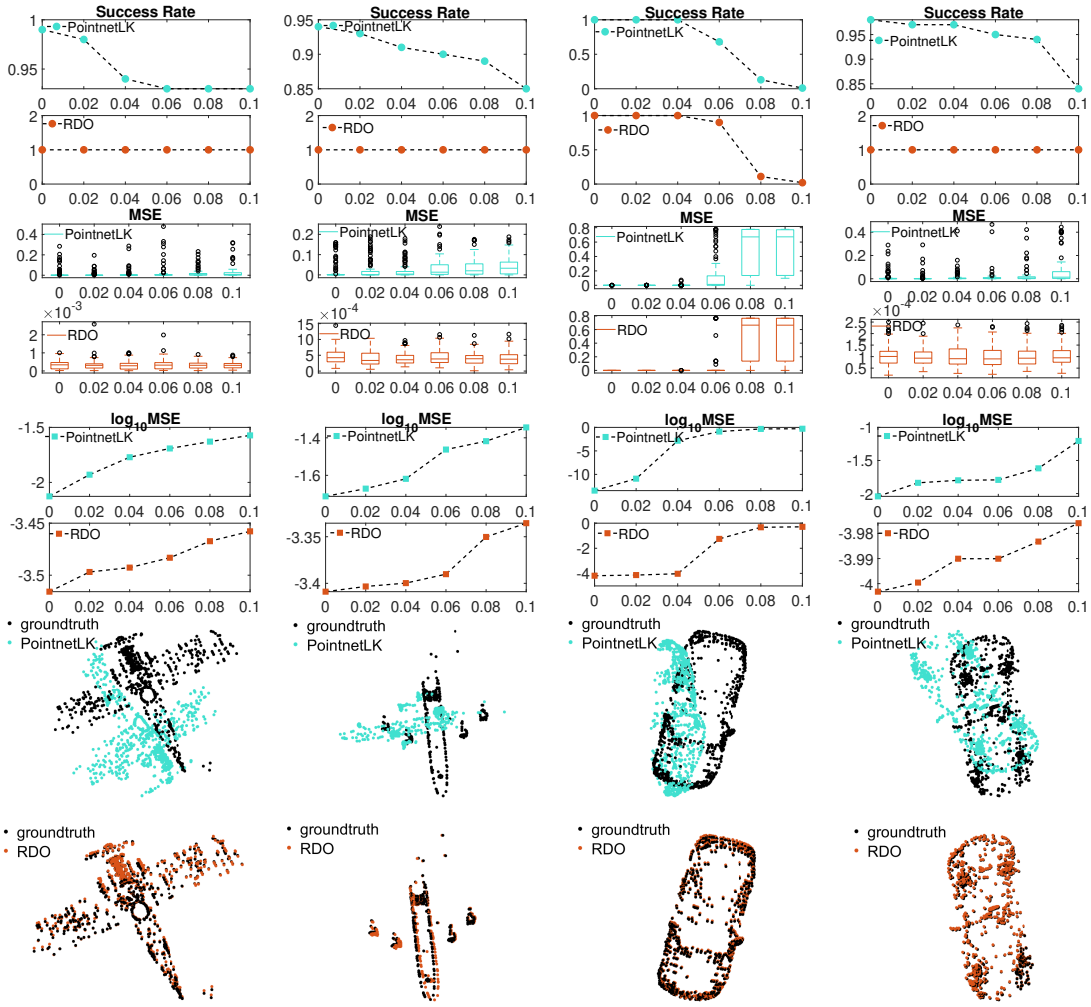
(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.18 The registration results of the single-class training scheme with perturbation setting $mode_2$.



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

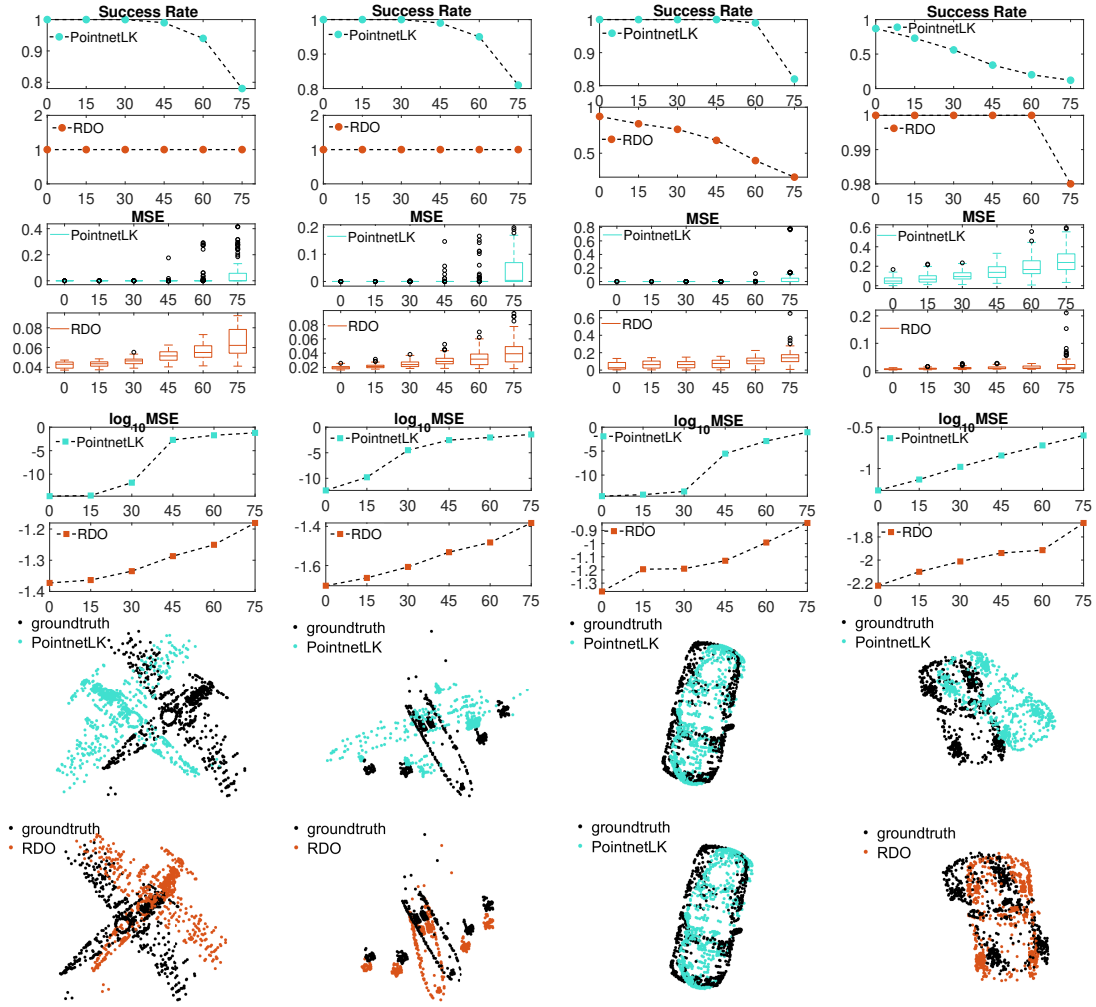
Fig. 4.19 The registration results of the single-class training scheme with perturbation setting $mode_3$.



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.20 The registration results of the single-class training scheme with perturbation setting $mode_3$.

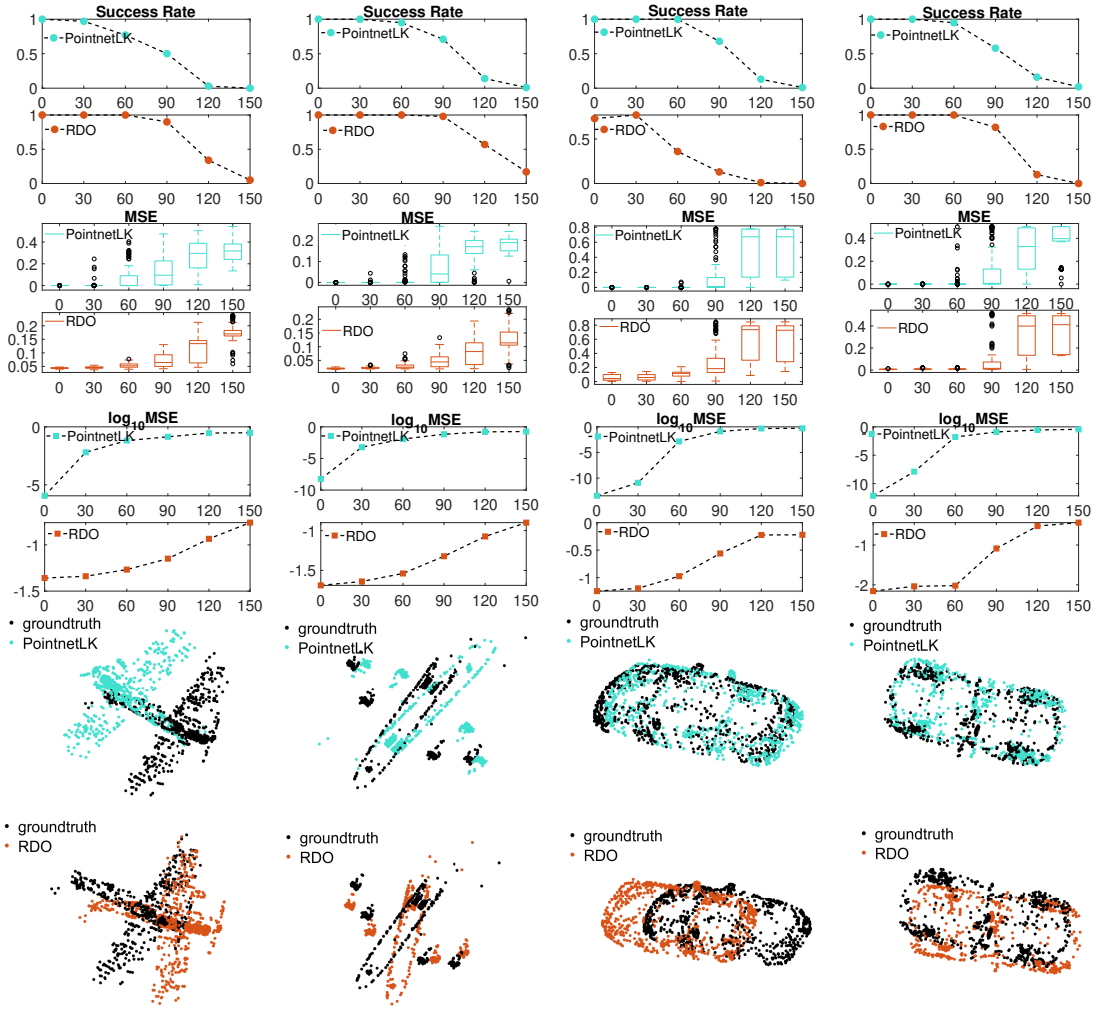
Figure.4.21, Figure.4.22, Figure.4.23 and Figure.4.24 illustrate the registration results of RDO and PointnetLK on ModelNet40 with the multi-class training scheme. The first two correspond to the evaluation for perturbation setting modes $mode_1$ and $mode_2$, and the last two show the registration results corresponding to the perturbation setting mode $mode_3$. Figure.4.21 shows the registration results when rotation is within 45^o during the training process. It can be seen that, compared with PointnetLK, the performance of RDO is stable but has lower accuracy, which also can be illustrated by Figure.4.22.



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.21 The registration results of the multi-class training scheme with perturbation setting $mode_1$.

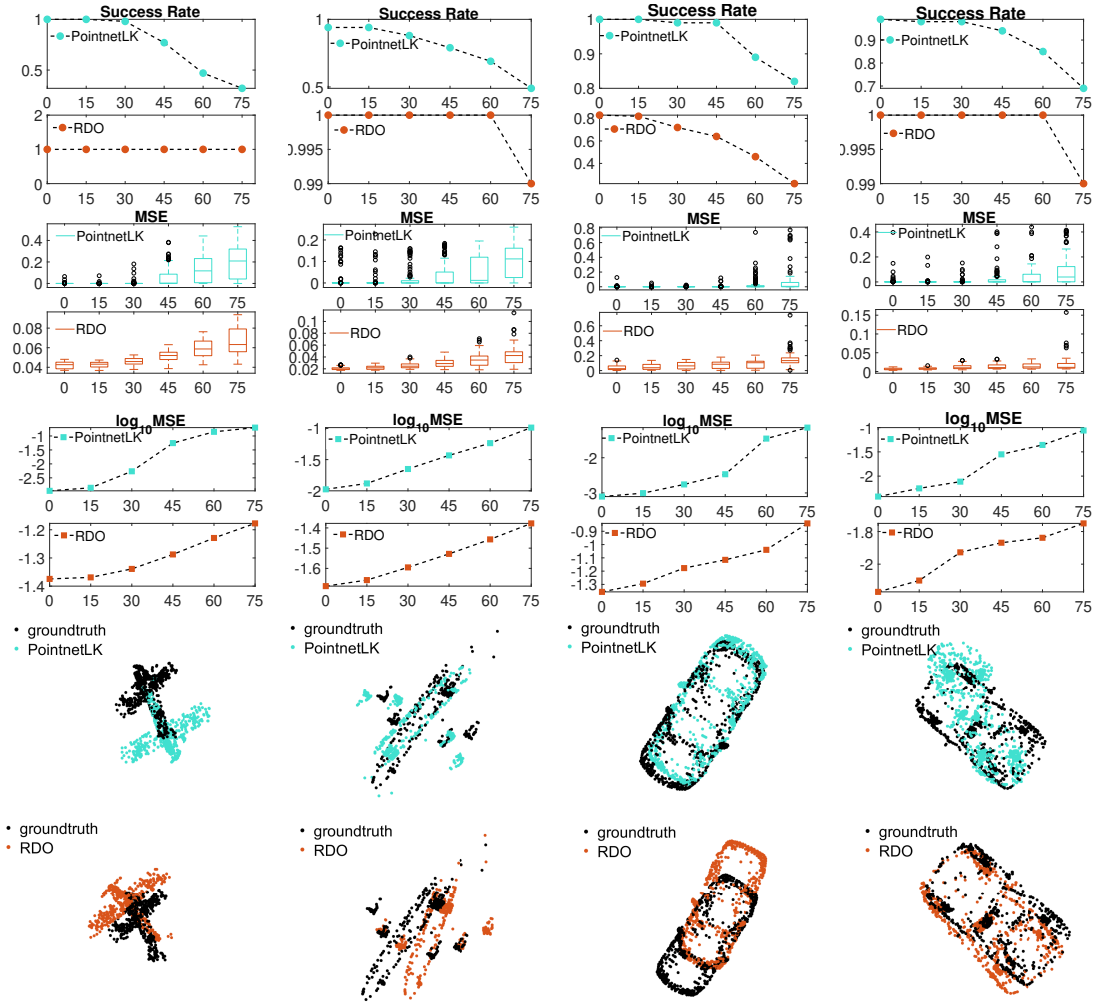
However, in the case of multi-class and multi-perturbation training, the stability of RDO and the accuracy of RDO are better, especially for registration when the perturbation scale exceeds the prescribed range setting in the training process, as shown in Figure.4.23 and Figure.4.24. Besides, Figure.4.21 and Figure.4.23 show that the stability of PointnetLK and the accuracy of PointnetLK are reduced in the case of multi-perturbations training. By contrast, the performance of RDO is more stable. In Figure.4.23, the Success Rate shows higher stability of RDO, and the MSE illustrates the higher accuracy of RDO when dealing



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.22 The registration results of the multi-class training scheme with perturbation setting $mode_2$.

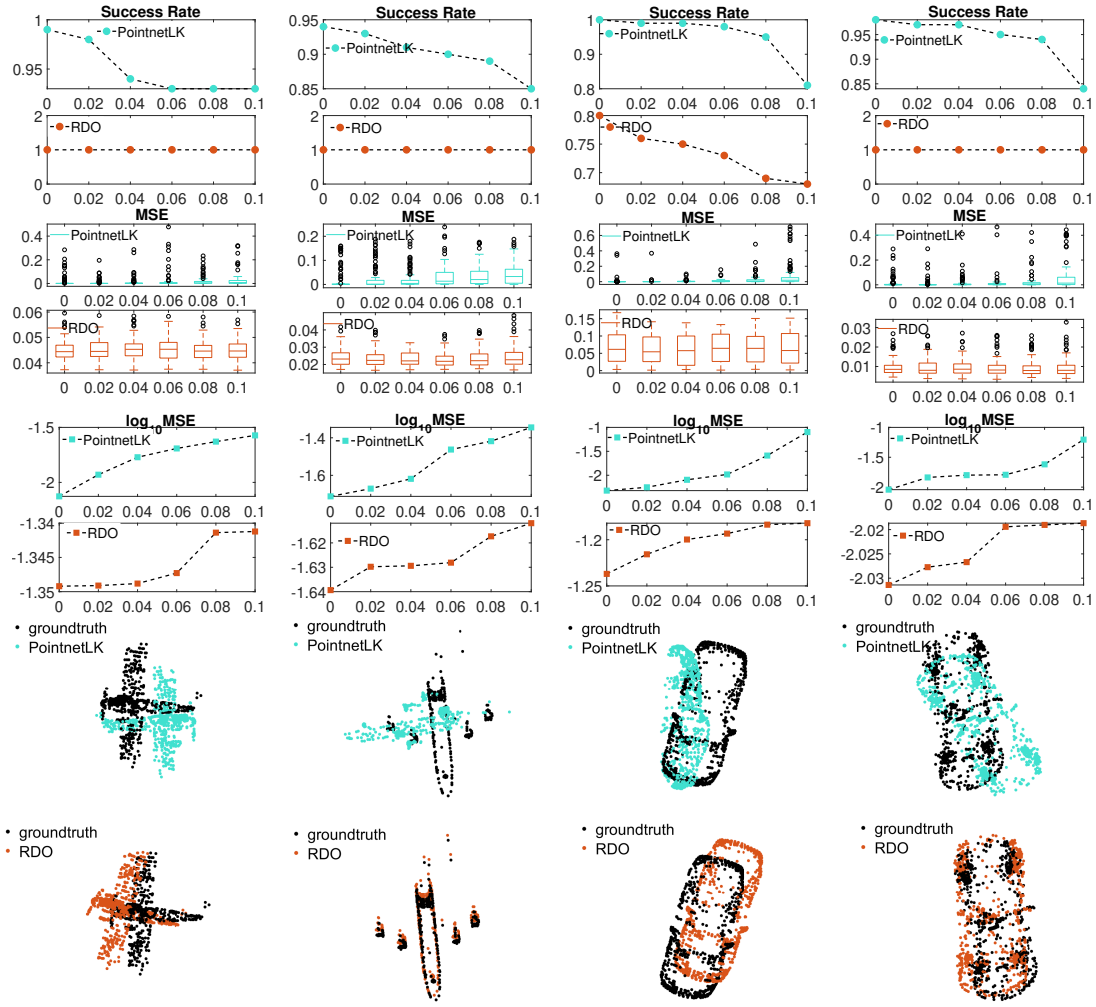
with registration with larger perturbations. In Figure.4.24, It can be seen that RDO can keep its higher stability in the case of muti-class and multi-perturbations training.



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.23 The registration results of the multi-class training scheme with perturbation setting $mode_3$.

Table.4.6 shows the quantitative results of the registration on the ModelNet40 data set . SR means the Successful Rate. This table compares the registration results of RDO with PointNetLK on different models (0001, 0144, 0007,0026) under various perturbations ($mode_1$,$mode_2$,$mode_3$) after the different training process (single class and Multiple classes). After training the model using a single data set (0001), RDO has 31.70% higher successful rate than PointNetLK. And the registration accuracy of RDO is almost triple as PointNetLK. However, the registration accuracy of RDO when dealing with large rotation (120,150) is 30% of that when dealing with small rotation (60,90). There is a similar situation in PointNetLK. After training RDO using multiple data sets (0001, 0144, 0007 and 0026), RDO still can keep its higher stability, but its accuracy has dropped by almost 1/3. In short, RDO has higher robustness than PointNetLK when dealing with the registration under various perturbations, which will be outstanding, especially after using the single data set to train the RDO method.



(a) Airplane0001 (b) Airplane0144 (c) Car0007 (d) Car0026

Fig. 4.24 The registration results of the multi-class training scheme with perturbation setting $mode_3$.

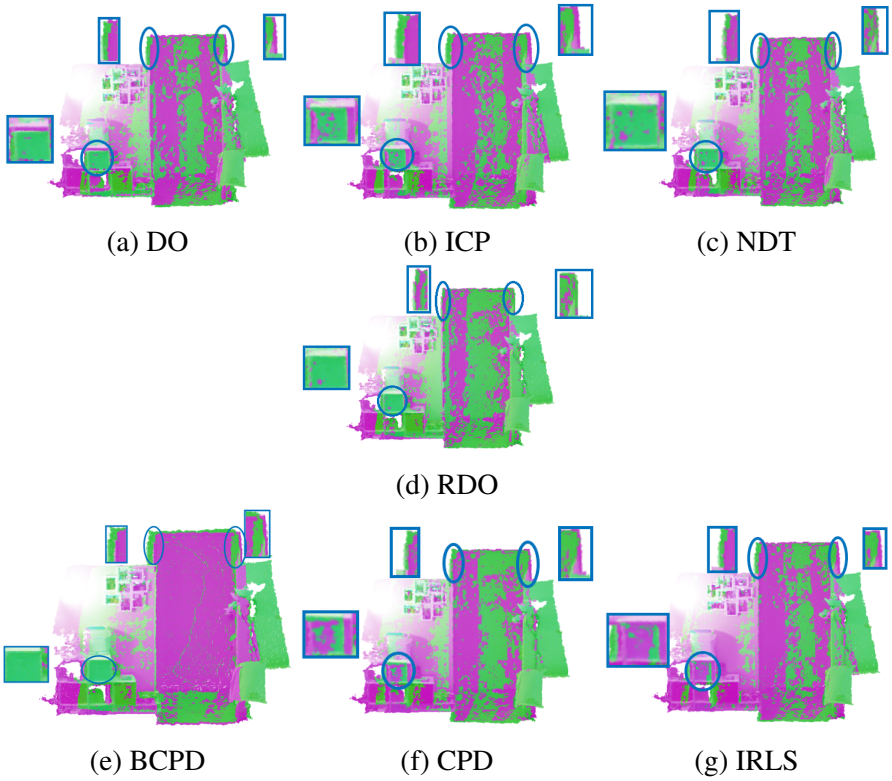
Table 4.6 The quantitative results of the registration on the ModelNet40 data set (SR-Successful Rate)

			Single Class				Multiple Classes			
			RDO		PointNetLK		RDO		PointNetLK	
			SR	log_{10}MSE	SR	log_{10}MSE	SR	log_{10}MSE	SR	log_{10}MSE
0001	$mode_1$	(R)60	**1.00**	**-3.70**	0.93	-2.50	**1.00**	-1.25	0.95	**-2.50**
		(R)75	**1.00**	**-3.70**	0.78	-1.30	**1.00**	-1.10	0.78	**-2.00**
	$mode_2$	(R)120	**0.50**	**-1.00**	0.00	-0.50	**0.25**	**-1.00**	0.00	-0.80
		(R)150	0.00	**-1.00**	0.00	-0.40	0.00	-0.75	0.00	-0.80
	$mode_3$	(R)60	**1.00**	**-3.47**	0.93	-1.60	**1.00**	**-1.25**	0.50	-0.85
		(R)75	**1.00**	**-3.46**	0.93	-1.50	**1.00**	**-1.10**	0.30	-0.80
		(N)0.08	**1.00**	**-3.48**	0.93	-1.60	**1.00**	-1.34	0.93	**-1.70**
		(N)0.100	**1.00**	**-3.46**	0.93	-1.46	**1.00**	-1.34	0.93	**-1.60**
0144	$mode_1$	(R)60	**1.00**	**-3.74**	0.95	-2.50	**1.00**	-1.50	0.95	**-2.50**
		(R)75	**1.00**	**-3.70**	0.80	-2.30	**1.00**	-1.40	0.80	**-2.40**
	$mode_2$	(R)120	**0.99**	**-1.20**	0.20	-0.80	**0.50**	**-1.10**	0.20	-0.80
		(R)150	**0.95**	**-1.00**	0.00	-0.80	**0.30**	-0.80	0.00	-0.80
	$mode_3$	(R)60	**1.00**	**-3.20**	0.70	-1.25	**1.00**	**-1.50**	0.65	-1.25
		(R)75	**1.00**	**-2.80**	0.50	-1.00	**0.99**	**-1.40**	0.50	-1.00
		(N)0.08	**1.00**	**-3.35**	0.89	-1.40	**1.00**	-1.40	0.90	**-1.62**
		(N)0.100	**1.00**	**-3.30**	0.85	-1.37	**1.00**	-1.35	0.85	**-1.60**
0007	$mode_1$	(R)60	**1.00**	**-4.05**	1.00	-2.50	0.30	-1.10	**1.00**	**-2.50**
		(R)75	**1.00**	**-3.95**	0.81	-1.00	0.25	-0.90	**0.80**	**-1.00**
	$mode_2$	(R)120	0.15	**-0.30**	0.15	0.00	0.00	**-0.25**	0.20	0.00
		(R)150	0.00	**-0.30**	0.00	0.00	0.00	**-0.25**	0.00	0.00
	$mode_3$	(R)60	0.20	**-0.10**	0.20	0.00	0.50	-1.45	**0.90**	**-1.70**
		(R)75	0.00	**-0.10**	0.00	0.00	0.20	-1.40	**0.82**	**-1.50**
		(N)0.08	0.20	-0.10	0.20	0.00	0.68	-1.18	**0.95**	**-1.50**
		(N)0.100	0.00	-0.10	0.00	0.00	0.67	**-1.15**	**0.80**	-1.00
0026	$mode_1$	(R)60	**1.00**	**-3.95**	0.98	-1.30	**1.00**	**-2.00**	0.20	-0.70
		(R)75	**1.00**	**-3.85**	0.85	-1.00	**0.98**	**-1.60**	0.10	-0.60
	$mode_2$	(R)120	**0.30**	**-0.80**	0.30	-0.50	0.20	-0.50	0.20	-0.50
		(R)150	0.00	**-0.70**	0.00	-0.10	0.00	0.00	0.00	0.00
	$mode_3$	(R)60	**1.00**	**-3.95**	0.85	-1.50	**1.00**	**-1.90**	0.85	-1.50
		(R)75	**1.00**	**-3.90**	0.70	-1.00	**0.99**	**-1.78**	0.70	-1.00
		(N)0.08	**1.00**	**-3.99**	0.95	-1.60	**1.00**	**-2.00**	0.95	-1.70
		(N)0.100	**1.00**	**-3.97**	0.85	-1.30	**1.00**	**-2.00**	0.85	-1.20

The comparative study demonstrates not only the ability of learning-based algorithms in dealing with complex point cloud registration tasks in the presence of noise and the incomplete data sets or occlusions but also shows the marked performance improvements of our proposed Reweighted Discriminative Optimisation (RDO) over the learning-based algorithm Discriminative Optimisation (DO) for registration tasks. And RDO maintains high accuracy, robustness, and stability when dealing with point cloud registration with large perturbations. Compared with the deep-learning-based method PointnetLK, RDO has better stability and robustness in dealing with registration problems with large and multiple perturbations.

Stitching Results

Figure.4.25 shows the 3D stitching results of the two-view point clouds. (a) is the value of $\sum_{i=1}^{N}\left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{h}_t^i$on different data sets. (b) is the training error of our method on different data sets. (c) and (d) are the rates of Convergence of DO and RDO on different data sets. The ellipses show the visible difference of the constructed 3D scenes, and the obvious detailed information is exhibited in the rectangles. The labels (DC, BM, etc.) are the abbreviations for the names of data sets; the rates of the convergence are marked by squares for DO and circles for RDO, respectively. The ellipses show the visible difference of reconstructed 3D scenes, and the obvious detailed information is exhibited in the rectangles. It can be seen that RDO and NDT performed well than other algorithms. RDO adjusts the degrees of rotation and translation by assigning weighting coefficients to transformation matrices, which causes the difference between the stitching results of DO and RDO.



(a) DO (b) ICP (c) NDT

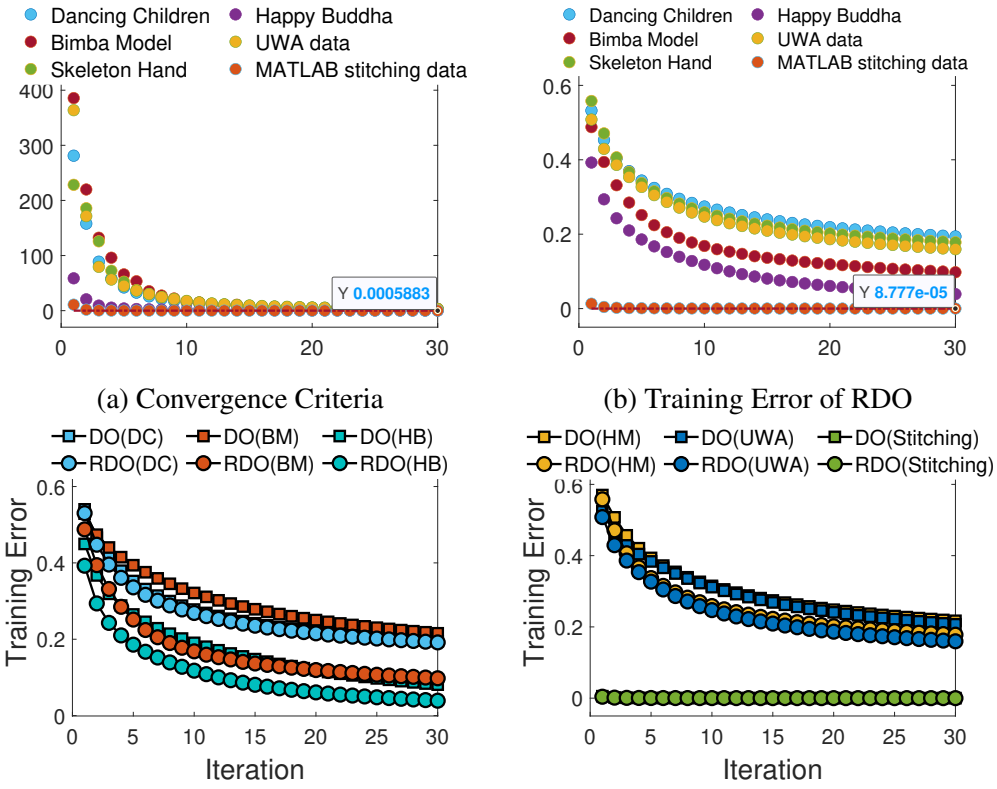
(d) RDO

(e) BCPD (f) CPD (g) IRLS

Fig. 4.25 Results of Stitching experiment on Matlab data set .

Proof of Convergence

Figure.4.26 shows the Convergence Criteria and Training Errors of RDO and DO on different data sets.



(a) Convergence Criteria

(b) Training Error of RDO

(c) Rate of Convergence of DO and RDO

(d) Rate of Convergence of DO and RDO

Fig. 4.26 The Convergence Criteria and Training Errors of RDO and DO on different data sets.

We can find that our method meets the convergence condition $\sum_{i=1}^{N}\left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{h}_t^i > 0$ for all data sets (as shown in (a)), and the training error of RDO decreases in each iteration (as shown in(b)). (c) and (d) illustrate that DO and RDO converges sub-linearly and logarithmically. It can be seen that the addition of weighting coefficients does not have impact on the rate of convergence.

Space Complexity Analysis

The memory requirement of RDO per iteration is determined by the variable storage. Calculating weighting matrix $\mathbf{W}_t$ needs maximum storage $O(c_1 \times N \times N_M)$; Calculating feature $\mathbf{h}$ needs maximum storage $O(c_2 \times N \times (N_M + N_S))$; Calculating the regressor $\mathbf{D}$ needs maximum storage $O(c_3 N_M \times N_M)$. Therefore, the memory requirement of RDO per iteration is $O\left(N\left(((c_1+c_2)N_M + c_2 N_S)\right) + c_3 N_M^2\right)$, where N_M, N_S are the size of target model $\mathbf{M}$ and the size of moving model $\mathbf{S}$, $c_i, i = 1,2,3$ are constant, and N is the number of training samples.

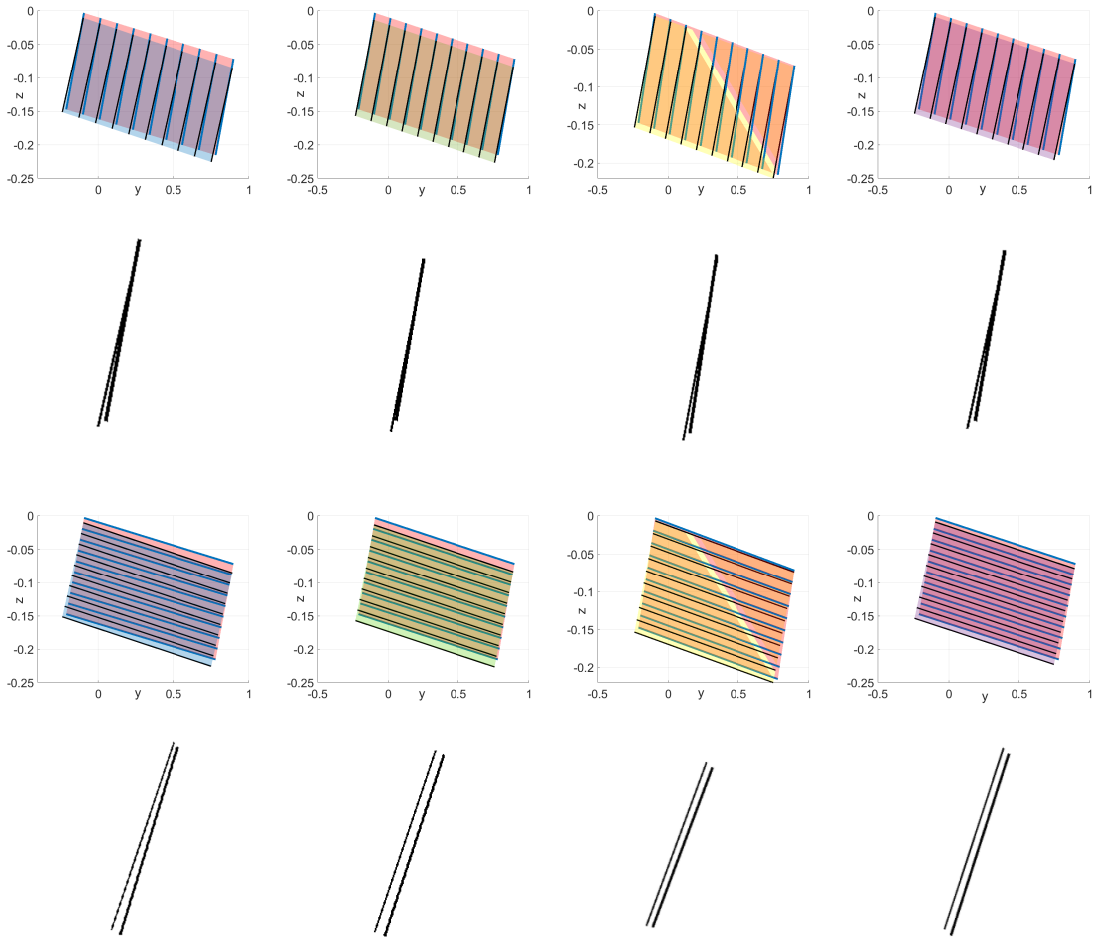
4.4 Discussion for DO and RDO

DO, as a classical Supervised sequential update (SSU) algorithm, utilises least-squares to solve parameter estimation in computer vision by learning update directions from training samples. DO is more robust to noise and outliers and other perturbations and efficient than other traditional methods. RDO is highly inspired by the DO algorithm. As an asymmetrical parameter treatment scheme, RDO aims to improve the robustness of parameter estimation in least-squares problems. RDO considers the different impact of each component of the parameter vector on the final fitting error, which is different from DO. Section 4.3 shows the comparison of the experimental results of RDO, DO, and other traditional registration methods and illustrates that RDO is more robust than DO.

This section discusses the substantial difference between DO and RDO, which focuses on the training stage. The computational efficiency of DO and RDO is different in the training stage and the same in the testing stage. The major difference in the training stage is the calculation of the weighting matrix. The computation of the weighting matrix is $O(6N)$, where N is the number of training samples. We compared the transformation status of DO and RDO at the end of the training process and when the iteration increases (as shown in Figure.4.27 and Figure.4.28). In addition, the registration performance under different perturbations is also discussed when the iteration number T increases (as shown in Figure.4.29).

Figure.4.27 illustrates the transformation differences of DO, RDO, and *RDO without* t at the end of the training process. The t is from the Eq.4.13 and Eq.4.14. The pink plane in the first row and third row represent the ground truth, where the lines drawn along rows and columns reflect the transformation difference. The lines in the second column correspond with the leftmost lines of each plane in the first column. The lines at the bottom are obtained by rotating the bottommost lines of the planes in the third column 90° clockwise. (a), (b) and (c) show the transformation of DO, RDO and *RDO without t* at the 30_{th} iteration respectively. (d) represents the transformation of *RDO without t* at the 29_{th} iteration. Due to the higher robustness of rotation than translation shown in Figure.**??**, the rotation plays a significant role in the transformation when handling perturbations. The rotation difference between ground truth and the transformation is reflected by the two adjacent edges of a plane. When the two adjacent edges of the plane are parallel or coincident with that of the ground truth plane (pink plane), the rotation angle is the same as that of the ground truth.

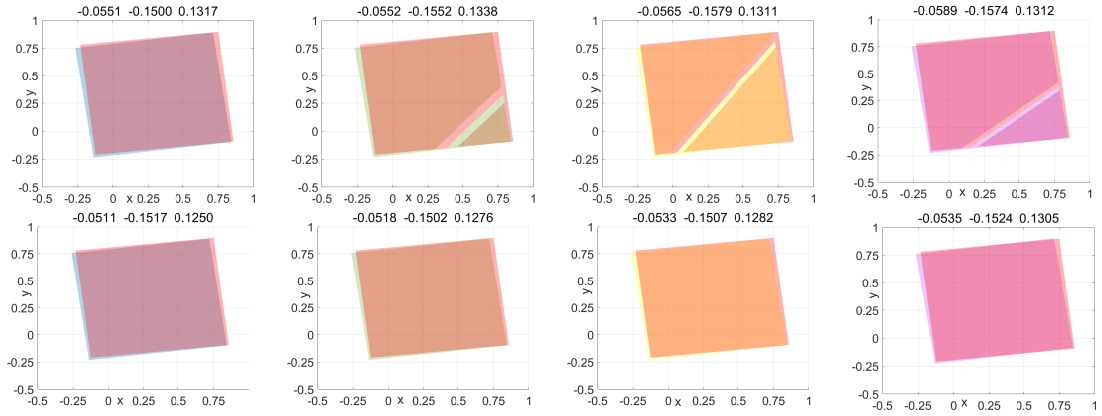
From (a) and (b), it can be seen that the rotation of RDO in the training process is closer to the rotation of ground truth, compared with that of DO. (c) and (d) illustrate that although allocating weights without t restriction can also guide the rotation to close to the ground truth (compared with DO), the weight without t will guide the plane to rotate by a large margin when the rotation has been close to the ground truth. (c) The shades in the first and third columns indicate that the two planes intersect, which means that after the 29_{th} iteration (d), the plane still rotates with a large degree.



(a) DO Iteration=30 (b) RDO Iteration=30 (c) RDO without t Iteration=30 (d) RDO without t Iteration=29

Fig. 4.27 The transformation differences of DO, RDO, and *RDO without t*.

Figure.4.28 shows the transformation at different iterations in the training stage. The first row show the transformations of DO at the 30_{th}, 34_{th}, 37_{th} and 40_{th} respectively. The values above each sub-figure correspond to sequential rotations about the X, Y, and Z axes. The rotation of ground truth (pink plane) is expressed as a vector [-0.0518 -0.1513 0.1099]. The second row displays the transformation of RDO. The shades on the figures show the intersection of the planes.



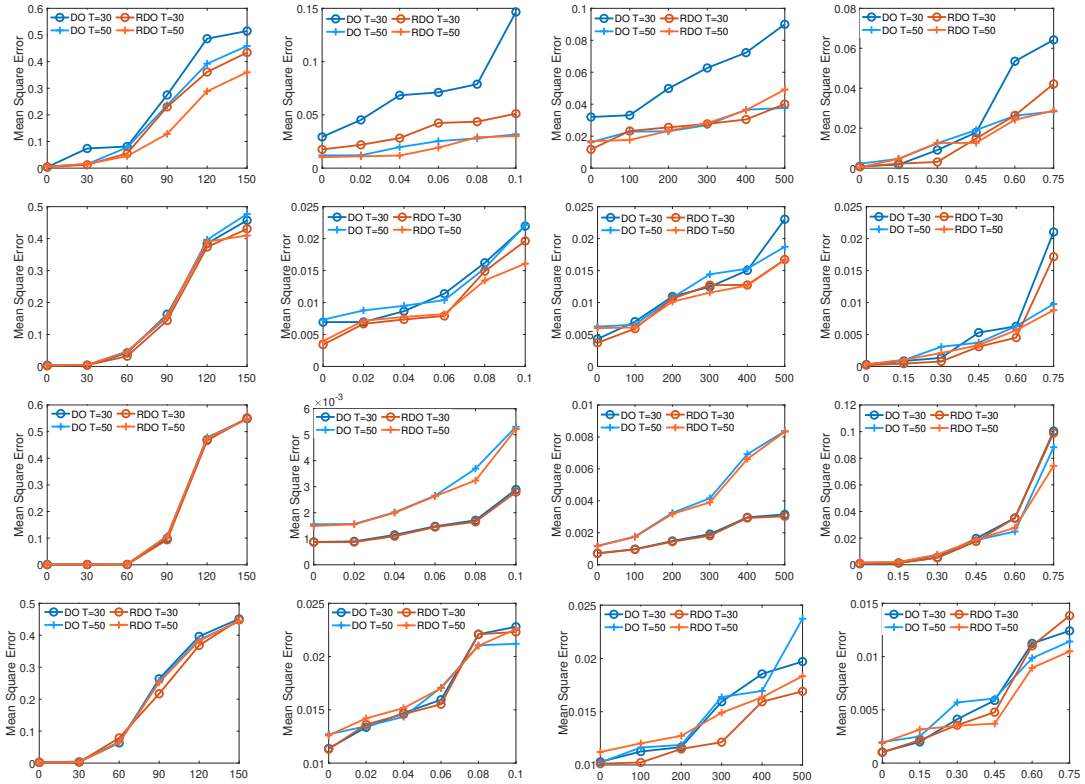
(a) Iteration=30 (b) Iteration=34 (c) Iteration=37 (d) Iteration=40

Fig. 4.28 The transformation with large t.

It can be seen that as the iteration number increases, the rotation of DO is far more different from that of the ground truth [-0.0518 -0.1513 0.1099]. Compared with DO, RDO's transformation changed a little. And there is no intersection between the transformation plane and the ground truth plane (pink).

Figure.4.29 shows the Mean Square Error of registration with Synthetic data under different perturbation when $T = 30$ and $T = 50$. Top involves the Happy Buddha model. The second row is about Skeleton Hand. The third row is about Bimba Model. Bottom is the result on the Dancing Children model. The plus and circle represent the MSE when $T = 50$ and $T = 30$, respectively. It can be seen that in most cases, the MSE of RDO registration is less than that of DO registration when $T = 50$, except for the registration of the Bimba Model with different initial rotation and noises. Besides, we can find that if the MSE of RDO registration when $T = 30$ is higher than the MSE of RDO registration when $T = 50$, such as the MSE of Happy Buddha with different noises, the MSE of RDO registration is still the least. Likewise, if the MSE of RDO registration when $T = 30$ is lower than that of RDO registration when $T = 50$, the MSE of RDO registration is the smallest compared with the MSE of DO registration. In short, compared with DO, RDO estimates a transformation vector closer to the ground-truth, when DO and RDO registration errors are both smaller.

Besides, we also compared the decreasing rate of rotation matching error and translation matching error between our algorithm and DO algorithm in each iteration. The rotation matching error is calculated through fixing $\hat{\mathbf{x}}_k = 0, k \in \{1,2,3\}$ and the translation matching error is the mean square error of all samples when $\hat{\mathbf{x}}_k = 0, k \in \{4,5,6\}$. Figure 4.30 shows the decreasing rate of matching error on rotation and translation. It can be seen that the



(a) Initial Angles (b) Noise SD (c) Outlier (d) Incompleteness

Fig. 4.29 The Mean Square Error of 3D registration with Synthetic data under different perturbations with different iteration numbers.

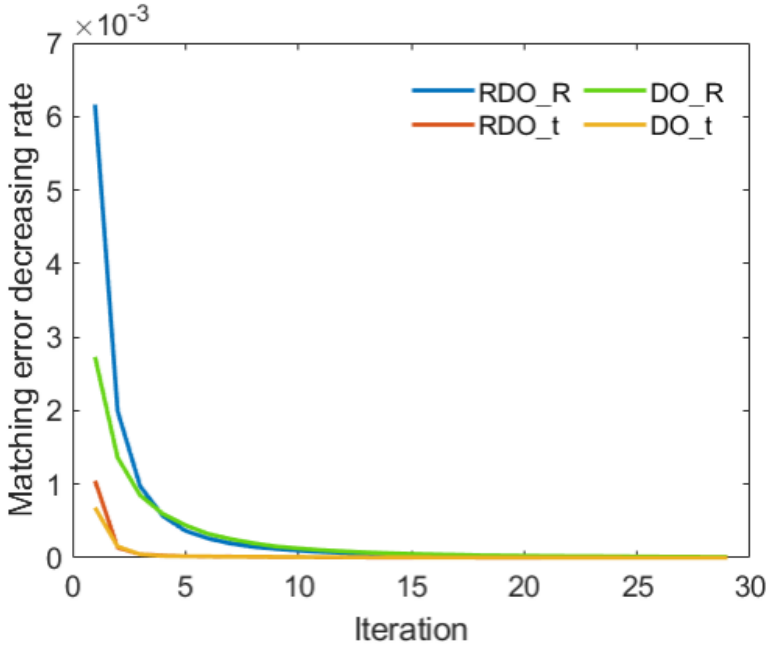


Fig. 4.30 The decreasing rate of rotation matching error and translation matching error.

weighting scheme in RDO accelerates the iterative process of convergence.

Summary

1) The asymmetrical parameter treatment scheme in RDO is able to adjust the scales of transformation in registration and make the rotation closer to the ground truth in the training process (as shown in (b), (c), (d) of Figure.4.27) and the weighting matrix can accelerate the convergence process.

2) The t in the weights assignments Eq.4.13, Eq.4.14 controls the scale of transformation in each iteration (compared (b), (c), (d) in Figure.4.27, as shown in Figure.4.28). As the iteration number increases, DO's transformation with the uniform weighting scheme has been far more different from the ground truth in terms of rotation, and the difference is more and more obvious. In contrast, RDO's transformation has only changed a little bit at each iteration. Besides, the accumulation of the little change of RDO's transformation still has successfully avoided the larger difference from the ground truth.

3) Increasing the number of iterations will influence the matching error (as shown in Figure.4.29). In most cases, the benefit of RDO will remain when the iteration number T increases. Besides, whatever the iteration number, when DO and RDO registration errors are both smaller, we can find that the RDO always estimates the transformation vector closer to the ground truth.

4.5 Discussion for RDO and PointnetLK

RDO and PointnetLK are both learning-based methods, and they have both similarities and differences. The most obvious similarity of the RDO and PointnetLK is that they have a similar structure- multiple layers of regressors. *Training*: PointnetLK can be trained "end-to-end" through back-propagation; specifically, all layers can be affected by minimizing the loss function at once. RDO needs to be trained layer-by-layer, and the loss function in each layer is different, but the essence is in common- make the currently estimated parameters approach to the ground truth. *Flexibility*: PointnetLK is more flexible. It can deal with different kinds of data sets (such as ModelNet40) at the same time. The memory requirement for learning $\mathbf{D_{t+1}}$ is $O\left(N\left(((c_1+c_2)N_M+c_2N_S)\right)+c_3N_M^2\right)$, which does not apply to dealing with many data sets at once like deep learning. Namely, PointnetLK is almost purely data-driven, and RDO is more model-driven, where model means the feature. Although PointnetLK can process a variety of data simultaneously, its stability is far less than that of RDO. Even in the case of dealing with large perturbations, RDO can still maintain stability, and the registration error is lower than that of PoinnetLK. Besides, multi-perturbations training will affect the performance of PoinnetLK, but RDO will maintain stability and robustness under the same conditions.

4.6 Conclusion

A novel Reweighted Discriminative Optimisation (RDO) is proposed, an asymmetrical parameter treatment scheme to improve the accuracy of parameter estimation in least-squares problems. Specifically, RDO assigns different weights to components of parameter vectors according to the characteristics of the fitting errors over parameter vectors space to emphasise certain components of parameter vectors. We provide theoretical proof on the convergence of RDO under mild conditions. We demonstrate the potential of RDO in computer graphics and visualisation applications through the problems of 3D point cloud registration and multi-view stitching. Our comparative study with state-of-the-art algorithms illustrates that RDO produces more accurate and stable results. The registration accuracy of RDO is almost 40.00% higher than that of DO on the Happy and Hand models. RDO has 2.11% higher accuracy than DO when achieving the registration on the Dancing Children model. And the successful Rate of RDO is almost 5% higher than that of DO. The computation time of RDO is almost 20% lower than that of DO. In addition, in the registration experiment on the ModelNet40 data set, the registration success rate of RDO is nearly 30% higher than that of PointNetLK. Training with multiple data sets or with a single data set will not impact the stability of RDO. However, the registration accuracy of RDO is higher after the training with a single data set than after multiple data sets. Nevertheless, RDO and PointNetLK both are unable to achieve accurate registration under large rotations (such as 120° and 150°). Future work is to design a generalised representation of parameter vectors that is suitable for computer vision and graphics applications other than those specific to the Lie Algebra for a rigid transformation matrix. On this basis, we believe RDO can be applied to a much wider range of problems in computer graphics and computer vision, such as non-rigid registration, image denoising, and so on.

Chapter 5

SGRTmreg: A learning-based optimisation Framework for Multiple Point Clouds Registration

5.1 Introduction

Although GDO and RDO can achieve the point cloud registration with more robustness and accuracy, GDO and RDO have only been applied to train single data set at any one time and test its relevant data accordingly. In contrast, deep-learning-based methods have shown their flexibility to simultaneously train a large amount of data and test any relevant data sets. This is since deep-learning-based methods can be considered data-driven, whereas learning-based optimisation methods are more model- or feature-driven.

In this chapter, the issue of single point cloud registration of learning-based optimisation methods is addressed via the devised computational framework called SGRTmreg to achieve multiple point clouds registrations while maintaining the high accuracy and robustness of the methods in the case of various perturbations. Given a collection of training point clouds and a target point cloud, there are three steps for multiple point clouds registration in this framework: (1) Finding a training point cloud similar to the target point cloud via a searching scheme; (2) Learning a sequence of regressors from the selected training point cloud through a learning-based optimisation method called Graph-based Reweighted Discriminative Optimisation (GRDO); (3) Applying the learned sequence of regressors in estimating the transformation parameters of the target point cloud via transfer learning. It is worth noting that the learned sequence of regressors can not only be used to register the selected training point cloud but also be applied to the registration of any target point cloud similar to the selected training point cloud.A similar training point cloud is selected via the devised searching scheme in terms of the graph structure of point clouds, the similarity of the coordinates, the similarity between the importance of graph nodes, and the similarity of the normal vectors of point clouds.

The way to extract features in GDO and RDO requires a large amount of memory storage and computational time to deal with the registration of dense point clouds. The

proposed Graph-based Reweighted Discriminative Optimisation (GRDO) method extract features from the key points selected based on the graph structure of the selected training point cloud to learn the updating gradient path, reducing the memory storage and computational cost for learning the sequence of regressors.

The potential of SGRTmreg on multiple point clouds registration is demonstrated on Modelnet40 data sets and the high performance of GRDO in point clouds registration under various perturbations is also evaluated. Experimental results show that SGRTmreg achieves accurate and efficient multiple points clouds registration and illustrate that the GRDO method outperforms the advanced registration methods, including deep learning-based methods in terms of robustness, accuracy, and computational time.

5.2 Motivation

The memory requirement for learning $\mathbf{D_{t+1}}$ of RDO and DO is $O\left(N\left(\left(\left(c_1+c_2\right)N_M+c_2N_S\right)\right)+c_3N_M^2\right)$, meaning that the larger the number of points, the longer it takes to extract feature points, making it infeasible to achieve the registration of dense point clouds. Besides, they can only train a single data set at any one time and test its relevant data accordingly, which is determined by the criteria of updating regressors.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{D}_{t+1}\mathbf{h}\left(\mathbf{x}_t\right) \tag{5.1}$$

Where $\mathbf{h} : \mathbf{R}^p \rightarrow \mathbf{R}^f$ is a function that encodes a feature of a point cloud, and $\mathbf{D}_{t+1} \in \mathbf{R}^{p\times f}$ is a matrix that maps the feature to an update vector, $\mathbf{x}_{t+1}$ is the updating parameter vector. It can be seen that the prerequisite for the learned regressor $\mathbf{D}_{t+1}$ being used to estimate the parameter vector $\mathbf{x}_{t+1}$ of the target point cloud is that features of the training and target point clouds must be similar and have the same dimension. Thus, the learning-based optimisation methods are commonly used to achieve identical point set registration with various perturbations.

Graph-based Discriminative Optimisation method is proposed in this paper to achieve point clouds registration with less storage requirement and less computational time. A framework called SGRTmreg is also devised for learning-based optimisation methods to register multiple point clouds via a single learned sequence of regressors $\mathbf{D}_{t+1}$.

5.3 Framework SGRTmreg

SGRTmreg aims to register multiple point clouds via a single sequence of learned regressor $\mathbf{D}_{t+1}$. The critical steps in this framework are: *(1)* Search for a training point cloud $\mathbf{S}$ which is similar to the target point cloud $\mathbf{M}$ (Section 5.3.2); *(2)* Learn the sequence of regressors $\mathbf{D}_{t+1}$ via GRDO (Section 5.3.3); *(3)* Apply the sequence of the learned regressors $\mathbf{D}_{t+1}$ to estimate the transformation parameters of the target point cloud $\mathbf{M}$ via transfer learning (Section 5.3.4). To make the registration accurate, stable, efficient, and with fewer storage requirements, key points are extracted according to the structural information of the graph

that is transformed from a point cloud. Then a feature f is designed based on the extracted key points. The feature f of $\mathbf{S}$ will be used to learn the sequence of regressors $\mathbf{D}_{t+1}$, and the sequence of learned regressors $\mathbf{D}_{t+1}$ will act on the feature f of $\mathbf{M}$ to estimate the transformation parameters. The scheme of SGRTmreg is shown in Figure.5.1. The process for multiple point clouds registration can be divided into three stages: (1) Extract key points of the collection of training point clouds $\check{S}$ and the target point cloud $\mathbf{M}$ respectively. At the same time, search for a similar training point cloud $\mathbf{S}$ for the target point cloud $\mathbf{M}$ via the Searching scheme based on the Graph structure, Coordinates information, Importance of nodes and Normal Vector information of point clouds (as shown in Figure.5.4). (2) Utilise feature f of $\mathbf{S}$ to learn the gradient direction $\mathbf{D}_{t+1}$ via GRDO. (3) The learned gradient direction $\mathbf{D}_{t+1}$ of $\mathbf{S}$ will be employed to the registration of $\mathbf{M}$ via transfer learning.

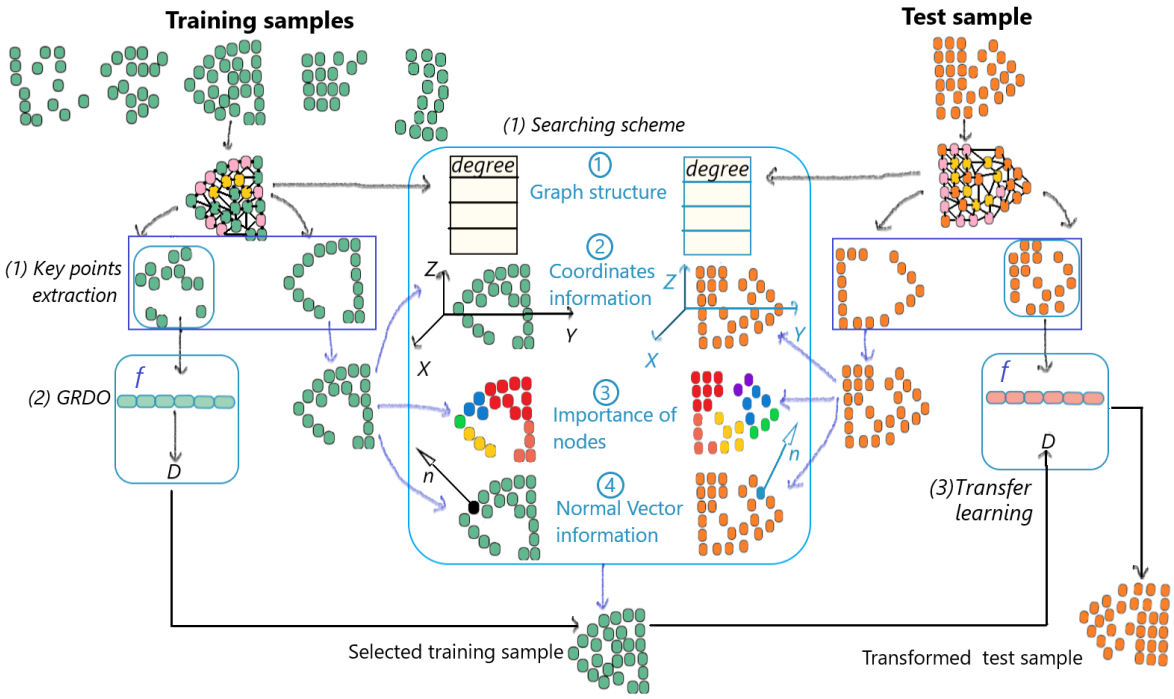


Fig. 5.1 The framework for multiple point clouds registration.

5.3.1 Key Points Extraction

Extracting key points aims to reduce the storage requirement for designing the feature f and learning the updating map $\mathbf{D}_{t+1}$, which will make the GRDO method less computational cost. The process of extracting key points is shown in Figure.5.2. Triangulation is used to transform point cloud $\mathbf{M}$ to a graph, then select the key points and the boundary of $\mathbf{M}$ based on the degree of nodes in the graph and the edges connected with nodes. The nodes whose degree has the most or the second most occurrence number will be selected as the key points, such as the pink and the orange. The nodes connected by the edge not shared by two triangles will be extracted as the boundary of $\mathbf{M}$.

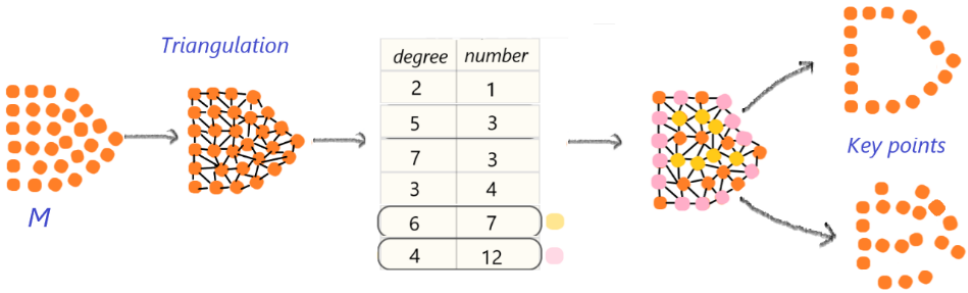


Fig. 5.2 The process of key points extraction.

Delaunay triangulation acts on the point cloud **M**. One of the triangle sides along the periphery of **M** is associated with only one triangle, all nodes connected by such edges on that side form the boundary. Triangulation post the point cloud **M** as a graph, the degrees of all nodes in which are counted. The degree of a node is the number of connections that it has to other nodes in the graph. The node whose degree is the most and the second most will be identified as the key point. The key points extraction can be cast as a downsample problem, which can reduce the point number while keeping the detailed information of the point cloud, as shown in Figure.5.3. Figure.5.3 shows that the proposed approach for key points extraction is able to keep the detailed information of models compared with the random and uniform downsample methods in MATLAB.

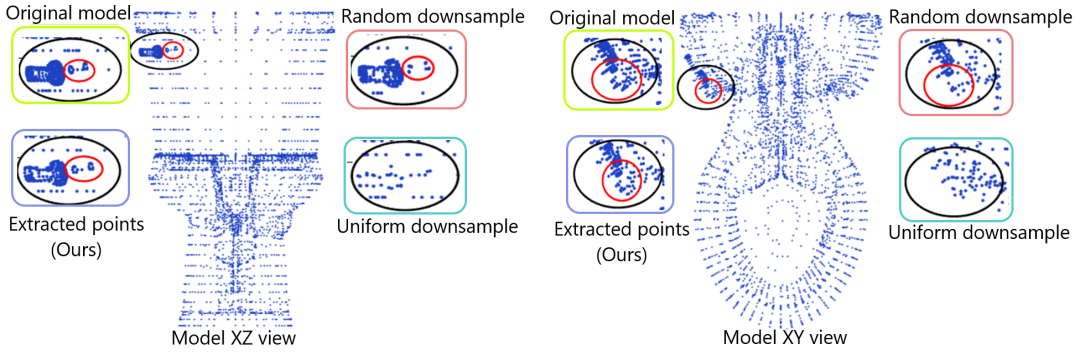


Fig. 5.3 The comparison of the proposed downsample approach with the random and uniform downsample methods in MATLAB.

5.3.2 Searching Scheme

The searching scheme aims to find a similar training point cloud for a given target point cloud, as shown in Figure.5.4.

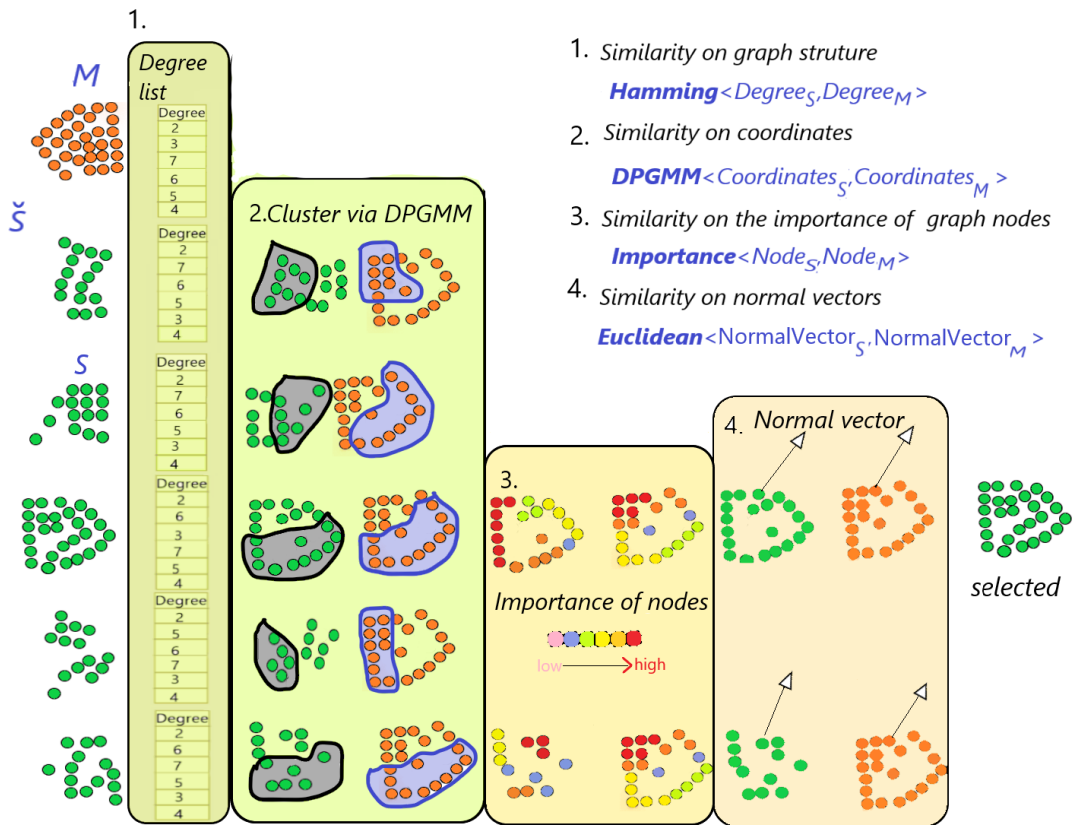


Fig. 5.4 The structure of searching scheme.

The searching scheme aims at searching for a similar training point cloud **S** for a given target point cloud **M** from the collection of training point clouds $\check{S}$. Specifically, the selection mechanism is based on four information sources: the graph structure of the point clouds, the similarity of the coordinates of the point clouds, the similarity between the importance of graph nodes, and the similarity of the normal vectors of the point clouds. Training point cloud that does not meet the requirements of the selection mechanism will be removed from the candidate list of similar samples. All training point clouds will be selected based on different criteria: the Hamming distance between the graph structure $<\mathbf{Degree_S}, \mathbf{Degree_M}>$, the clustering results based on the coordinates of points $<\mathbf{Coordinates_S}, \mathbf{Coordinates_M}>$ via Dirichlet Process Gaussian Mixture Model, the difference between the importance of graph nodes $<\mathbf{Node_S}, \mathbf{Node_M}>$ and the similarity of normal vectors $<\mathbf{NormalVector_S}, \mathbf{NormalVector_M}>$. The collection of training point clouds is denoted as $\check{\mathbf{S}}$, the training point cloud as **S**, and the target point cloud as **M**. i is the index of **S** in $\check{S}$. The number of points in **M** is N_M, and the number of points in **S** is N_S.

Similarity on graph structure

After the Delaunay triangulation of the point cloud, the degree of the nodes in the graph is used to scan similar training point clouds initially. $\mathbf{Degree_S} = \left[de_S^1, \cdots, de_S^k, \cdots\right]$ and $\mathbf{Degree_M} = \left[de_M^1, \cdots, de_M^k, \cdots\right]$ represent the degree list of training point cloud **S** and the target point cloud **M**, where k is the index of degree list and each value in the degree list represents the degree of nodes. The degrees are sorted according to the occurrences,

respectively. The degree lists of **S** and **M** have the same length. If the length of $\mathbf{Degree_S}$ is larger than that of $\mathbf{Degree_M}$, the degree with less occurrence in $\mathbf{Degree_S}$ will be removed. If it is shorter, the list $\mathbf{Degree_S}$ will be filled with 0 until the length of $\mathbf{Degree_S}$ is the same as that of $\mathbf{Degree_M}$.

$$S_{De}^{i} = 1 - \frac{d_H\left(\mathbf{Degree_S^i}, \mathbf{Degree_M}\right)}{L} \tag{5.2}$$

Where d_H represents the *Hamming distance*, which shows the number of the corresponding elements at the same position in $\mathbf{Degree_S^i}$ and $\mathbf{Degree_M}$ that are different. The length of the degree list of target point cloud $\mathbf{Degree_M}$ is noted as L. A training point cloud will be passed to the second round as a candidate similar point cloud if the value of S_{De}^{i} is larger than β. $\beta \in (0.5, 1)$ always be set manually.

Similarity on coordinates

The selection based on the similarity of the graph structure eliminates a large number of training point clouds that do not belong to the same category as the target point cloud. The similarity of the distribution of point coordinates is used as a screening criterion to select a similar training point cloud from the remaining collection of candidate training point clouds. The similarity of the coordinates distribution is measured through the Dirichlet Process Gaussian Mixture Model (DPGMM) [106], which clusters the mixture of extracted points of similar point clouds candidate $\mathbf{Coordinates_S^i}$ and that of the target point cloud $\mathbf{Coordinates_M}$.

Suppose $\mathbf{P} = [\mathbf{Coordinates_S^i}; \mathbf{Coordinates_M}]$ represents the coordinates of the mixture for clustering. And assuming the mixed set **P** has been divided into K clusters via DPGMM. $\mathbf{P} = \{\mathbf{C}_1, \cdots, \mathbf{C}_K\}$ and $\mathbf{C}_k = \left\{\mathbf{C}_S^k, \mathbf{C}_M^k\right\}$, $k \in \{1, \cdots, K\}$. $\mathbf{C}_S^k$ and $\mathbf{C}_M^k$ are the coordinates of training points and test points in cluster $\mathbf{C}_k$ respectively. The number of training points in $\mathbf{C}_k$ is N_S^k. The number of test points $\mathbf{C}_k$ is N_M^k.

$$R_S = \left[\frac{N_S^1}{N_S}, \cdots, \frac{N_S^K}{N_S}\right]$$
$$R_M = \left[\frac{N_M^1}{N_M}, \cdots, \frac{N_M^K}{N_M}\right]$$

R_S illustrates the proportion of $\mathbf{C}_S^k$ in $\mathbf{Coordinates_S^i}$. R_M is the proportion of $\mathbf{C}_M^k$ in $\mathbf{Coordinates_M}$.

$$\tau_S = \sum_{k=1}^{K}\left(k \times \delta\left(\frac{N_S^k}{N_S} - max\left(R_S\right)\right)\right) \tag{5.3}$$

$$\tau_M = \sum_{k=1}^{K}\left(k \times \delta\left(\frac{N_M^k}{N_M} - max\left(R_M\right)\right)\right) \tag{5.4}$$

Where δ is the *Dirac delta function* [107]. τ_S represents that the cluster $\mathbf{C}_{\tau_S}$ has the largest number of training points (as shown the cluster circled by the black in Figure.5.4). The cluster $\mathbf{C}_{\tau_M}$ groups the largest number of test points (as shown the cluster circled by the blue in Figure.5.4). $\tau_S^i = \tau_M$ means that the coordinates distribution of S^i is similar to the coordinates distribution of $\mathbf{M}$, thus the S^i will be passed to the next round as the candidate similar training point cloud. Please note that if $\mathbf{Coordinates_M}$ has been divided equally, the training point cloud S^i will also be regarded as the candidate similar training point cloud.

Similarity on the importance of graph nodes

The similarity of the importance of graph nodes is also used as the screen criteria to narrow the range of candidates chosen in the previous round. Eigenvector centrality [108] is used to measure the importance of a node in a graph, which depicts the closeness of points in a point cloud. For a given graph $G := (V,E)$ with $|V|$ number of nodes and $|E|$ number of edges, let $\mathbf{A_d} = (\mathbf{a}_{v,t})$ be the adjacency matrix, i.e. $\mathbf{a}_{v,t} = 1$ if node v is linked to node t, and $\mathbf{a}_{v,t} = 0$ otherwise. The v_{th} component of the largest eigenvector of the adjacency matrix gives the relative centrality score of the node v in the graph [108]. The scores are normalised such that the sum of all centrality scores is 1. The scores of the nodes with the same background colour in Figure.5.4 are close to each other. S^i will be the final selected similar training point cloud if the average score of all nodes of S^i is the closet to that of the target point cloud $\mathbf{M}$. If there is more than one point cloud whose average score is closest to that of $\mathbf{M}$, these training point clouds will participate in the final selection process based the similarity of normal vectors of the point clouds.

Similarity on normal vectors

The similarity of the normal vectors of extracted points is employed to find the training point cloud that is most similar to the target point cloud. $\mathbf{NormalVector}_S = \left[\mathbf{n}_S^1, \mathbf{n}_S^2, \cdots \mathbf{n}_S^n\right]$ is the normal vectors set including the normal vectors of the collection of the candidate similar training point clouds. n is the size of the candidate training collection in the current round. Euclidean distance between each normal of $\mathbf{M}$ and $\mathbf{NormalVector}_S$ is calculated, which will generate a distance matrix $\mathbf{E}$ with the size of $N_M \times n \times N_S$. $\mathbf{E}_{p,q} = d_\mathbf{E} < \mathbf{n}_M^p, \mathbf{n}_S^q >$, p is the index of the normal vector of $\mathbf{M}$, q is the index of $\mathbf{NormalVector}_S$. d_E is the *Euclidean distance*. $\mathbf{E}_{p,:}$ represents the the p_{th} row of the matrix $\mathbf{E}$. $\mathbf{E}_{:,q}$ represents the the q_{th} column of the matrix $\mathbf{E}$. Another matrix $\mathbf{E}^c$ with the size of $N_M \times n \times N_S$ is used to count the similarity of the normal vectors of point clouds.

$$\mathbf{E^c}_{p,q} = \begin{cases} 1 & \mathbf{E}_{p,q} = min(\mathbf{E}_{p,:}) \\ 0 & \mathbf{E}_{p,q} \neq min(\mathbf{E}_{p,:}) \end{cases} \tag{5.5}$$

$$N_s^i = \sum_{j=N_S\times(i-1)+1}^{N_S\times i} \mathbf{E^c}_{:,j}, i \in \left\{1, 2 \cdots n^{'}\right\} \tag{5.6}$$

The i_{th} training point cloud with the maximal value of N_s^i is the final selected similar training point cloud $\mathbf{S}$. The updating map of $\mathbf{S}$ will be applied to the transformation parameter estimation of the target point cloud $\mathbf{M}$.

5.3.3 Graph-based Reweighted Discriminative Optimisation

The graph-based reweighted discriminative optimisation method aims to achieve registration with less computational time and storage requirement, which involves learning the updating maps and the design of features. The latter needs to make the dimensions of the features of various point clouds are same.

Sequence of Update Maps

Let $\mathbf{h} : \mathbf{R}^p \to \mathbf{R}^f$ be a function encoding a feature of a point cloud, and $\mathbf{D}_{t+1} \in \mathbf{R}^{p \times f}$ be a matrix mapping the feature to an update vector. Given an initial parameter vector $\mathbf{x}_0 \in \mathbf{R}^p$, the iterative updating process can be defined as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{D}_{t+1}\mathbf{h}\left(\mathbf{x}_t\right) \tag{5.7}$$

The update process ends until $\mathbf{x}_{t+1}$ converges to a stationary point. And the sequence of matrices $\mathbf{D}_{t+1}, t = 0, 1 \cdots$ are learned through approximating the estimated parameter vector $\mathbf{x}_{t+1}^i$ to the ground truth $\mathbf{x}_*^i$.

$$\begin{aligned} \mathbf{D}_{t+1} &= \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_{t+1}^i - \mathbf{x}_*^i \right) \right\|_2^2 \\ &= \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_t^i - \hat{\mathbf{D}}\mathbf{h}\left(\mathbf{x}_t^i\right) - \mathbf{x}_*^i \right) \right\|_2^2 \end{aligned} \tag{5.8}$$

Where N is the number of point clouds which participate in the training process, $\mathbf{x}_t^i$ is the parameter vector of i-th point cloud at the t -th iteration. $\mathbf{W}_t \in \mathbf{R}^{p \times p}$ is a weighting diagonal matrix. A detailed explanation of the weighting matrix and Equation.5.8 has been provided in the previous work [109]. For simplicity, $\mathbf{x}_t^i$ is denoted as $\mathbf{x}_t$ for any point cloud. The learned sequence of update maps $\mathbf{D}_{t+1}$ will be utilised to estimate the transformation parameters of $\mathbf{M}$.

Design the feature function h

The function $\mathbf{h}$ in Equation.5.7 is used to extract the feature $\mathbf{f}_S$ of $\mathbf{S}$ and the feature $\mathbf{f}_M$ of $\mathbf{M}$. Equation.5.7 shows that the dimension of $\mathbf{f}_S$ is related to the dimension of the learned updating map $\mathbf{D}_{t+1}$, which illustrates that the dimension of $\mathbf{f}_M$ needs to be the same as the dimension of $\mathbf{f}_S$. A sparse matrix $\mathbf{S_p}$ is designed to make the dimensions of both features the same, as shown in Figure.5.5. $\mathbf{h}$ is a histogram indicating the position information of each key point s_1 in $\mathbf{S}$. The space around the $\mathbf{S}$ is divided into the uniform grid $\mathbf{G}$ in the

range [-2,2] in each dimension, and each grid stores the value of **h** evaluated at the centre of each grid.

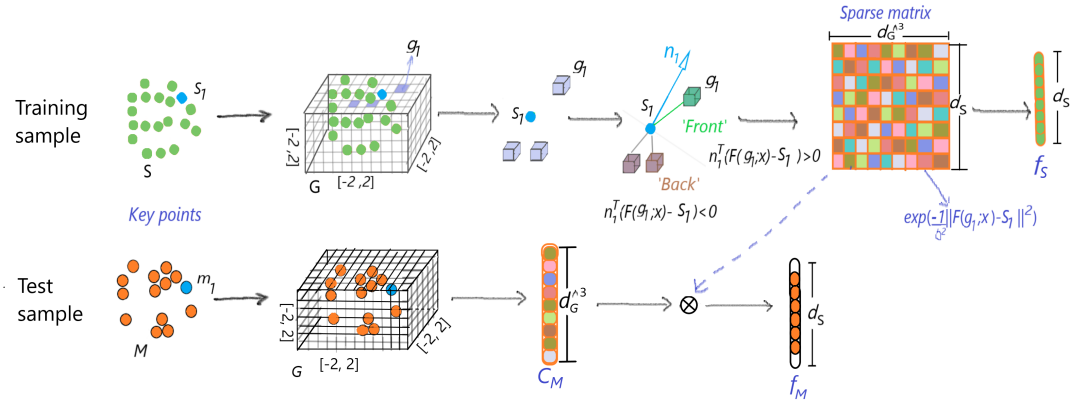


Fig. 5.5 The process of feature extraction.

A grid is denoted as g_1, then the feature $\mathbf{f}_S$ of $\mathbf{S}$ can be calculated as follows: Let $\mathbf{n}_1$ be a normal vector of s_1 computed from its neighbouring points; $F(\mathbf{y};\mathbf{x})$ applies rigid transformation with parameter $\mathbf{x}$ to vector $\mathbf{y}$; $\mathbf{g}^+ = \{\mathbf{g}_1 : \mathbf{n}_1^T (F(\mathbf{g}_1;\mathbf{x}) - S_1) > 0\}$ is the set of grids on the 'front' of s_1 ; and $\mathbf{g}^- = \{\mathbf{g}_1 : \mathbf{n}_1^T (F(\mathbf{g}_1;\mathbf{x}) - S_1) < 0\}$ is the set of grids on the 'back' of s_1.

$$[\mathbf{S_p}]_{i,j} = exp\left(-\frac{1}{\hat{\sigma}^2}\left\|F(\mathbf{g}_j;\mathbf{x}) - S_i\right\|^2\right) i = 1,\cdots d_S \quad j = 1,\cdots d_G^3 \tag{5.9}$$

$$[\mathbf{h_S}(\mathbf{x};\mathbf{G})]^+ = \frac{1}{z}\sum_{\mathbf{g}_1 \in \mathbf{g}^+} exp\left(-\frac{1}{\hat{\sigma}^2}\left\|F(\mathbf{g}_1;\mathbf{x}) - S_1\right\|^2\right) \tag{5.10}$$

$$[\mathbf{h_S}(\mathbf{x};\mathbf{G})]^- = \frac{1}{z}\sum_{\mathbf{g}_1 \in \mathbf{g}^-} exp\left(-\frac{1}{\hat{\sigma}^2}\left\|F(\mathbf{g}_1;\mathbf{x}) - S_1\right\|^2\right) \tag{5.11}$$

$$\mathbf{f}_S = \left[[\mathbf{h_S}(\mathbf{x};\mathbf{G})]^+; [\mathbf{h_S}(\mathbf{x};\mathbf{G})]\right] \tag{5.12}$$

Where z as the normalisation factor normalises $\mathbf{f}_S$ to sum to 1, and $\hat{\sigma}$ as the inner scale controls the width of the *exp* function. The element of the feature $\mathbf{f}_S$ less than 10^{-6} is set to 0. Then the feature $\mathbf{f}_S$ will be used to learn the updating map $\mathbf{D}_{t+1}$ via the Equation.5.8.

5.3.4 Transfer Learning

After learning the updating map $\mathbf{D}_{t+1}$, the learned map $\mathbf{D}_{t+1}$ will be utilised to estimate the transformation of the target point cloud $\mathbf{M}$ via Equation.5.7. The counted number of the key points falling into each uniform grid forms the vector $\mathbf{c}_M$. Then the feature $\mathbf{f}_M$ can be calculated as follows:

$$\mathbf{f}_M = \mathbf{c}_M \times S_p \tag{5.13}$$

Then the transformation parameters of the target point cloud **M** can be attained through the following iterative formula:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{D}_{t+1} \times \mathbf{f}_M \tag{5.14}$$

Where $\mathbf{D}_{t+1}$ is the learned updating map in the training stage of the selected training point cloud **S**, which is reused as the gradient map to guide the transformation estimation of the target point cloud **M** .

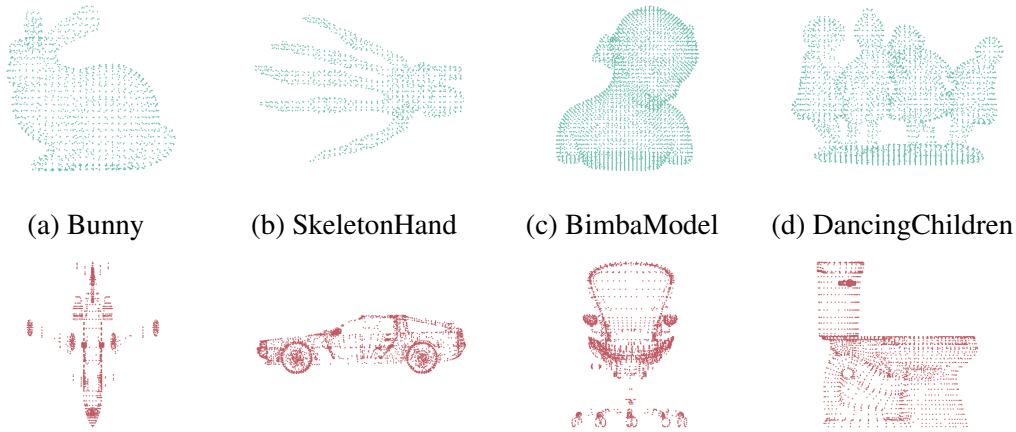
5.4 Experimentation

This section describes how to apply the proposed framework SGRTmreg for multiple 3D point clouds registration. Three kinds of experiments are conducted: one is the comparison experiments with the traditional registration methods- *Iterative Closest Point* (ICP), *Coherent Point Drift* (CPD), *Normal Distributions Transform* (NDT), and *Bayesian Coherent Point Drift* (BCPD) and learning-based optimisation methods- *Discriminative optimisation method* (DO), *Reweighted Discriminative optimisation method* (RDO) on synthetic data sets to show the accuracy and efficiency of GRDO. Another is the comparison with deep learning-based registration methods on the ModelNet40 data sets, which involves the selection of training point clouds for target point clouds and the estimation of the parameters via transfer learning. An experiment to verify the validity of transfer learning on multiple point clouds registration is also conducted.

5.4.1 Experimental Design

Data sets

GRDO is compared with the registration methods (DO, RDO, ICP, CPD, NDT and BCPD) on synthetic data sets ($(a) \sim (d)$ of Figure.5.6): Stanford Bunny model [13], Skeleton Hand model [18], Bimba Model and Dancing Children (http://visionair.ge.imati.cnr/).Comparative registration experiments are also conducted on the ModelNet40 data set[16] ($(e) \sim (h)$ of Figure.5.6) with traditional methods and other advanced deep-learning-based registration methods, such as PCRNet [101], PointnetLK [17], and DCP [51].



(a) Bunny (b) SkeletonHand (c) BimbaModel (d) DancingChildren

(e) Airplane627 (f) Car201 (g) Chair964 (h) Toilet378

Fig. 5.6 3D registration data sets.

GRDO Training

Comparison on synthetic data sets

The parameters in the GRDO training process are similar to those in DO [11]. A given model shape $\mathbf{S}$ is normalised to $[-1,1]^3$ and a scene model is generated through uniformly sampling from $\mathbf{S}$ with the replacement of almost 1500 points. Then the following perturbations are applied to the scene model: *(i) Rotation and translation:* The rotation is within 60 degrees and the translation is in $[-0.3,0.3]^3$, which represents the ground truth $\mathbf{x}_*$; *(ii) Noise and Outliers:* Gaussian noise with the standard deviation 0.05 is added to the scene model. 0 to 300 points within $[-1.5,1.5]^3$ are added as the sparse outliers. Besides, a Gaussian ball of 0 to 200 points with a standard deviation of 0.1 to 0.25 is used to simulate the structured outliers. For all experiments, 30000 training samples are generated, the number of iterations is set to $T = 30$, and the initial transformation $\mathbf{x}_0$ is $\mathbf{0}^6$.

Comparison on the ModelNet40 data set

Three modes for GRDO training are designed in this chapter. (i) $mode_1$: The rotation is within 45 degrees and the translations is in $[-0.5,0.5]^3$; (ii) $mode_2$: The rotation is within 90 degrees and the translations is in $[-0.5,0.5]^3$; (iii) $mode_3$: The rotation is within 90 degrees, the translations is in $[-0.5,0.5]^3$ and Gaussian noise with the standard deviation 0.05 is also applied. The first two modes aim to compare the registration of all methods in terms of varying degrees of rotation, named *single-class training*. The latter is to compare the performance of different methods on the registration with multiple perturbations, named *multi-class training*. 30000 training samples are generated for all modes, and the training samples are generated based on the selected point cloud according to the searching scheme in Section5.3.2. Please note that the selected training point cloud will be normalised to $[-1,1]^3$ without downsampling. The number of points of all samples is 5120.

Experiments Metrics

Baselines GRDO is compared with the advanced learning-based approaches (DO and RDO), the classical point-based approach ICP and three density-based approaches (CP, BCPD and NDT) and the deep-learning-based methods (PCRNet, PointnetLK and DCP). The codes for all methods were downloaded from the authors' websites, except for ICP where MATLAB's implementation is used. The code of GRDO was implemented in MATLAB.

Evaluation Metrics log_{10} MSE, and computation time are the performance metrics for comparing the performance of GRDO with registration methods (DO, RDO, ICP, NDT, CPD, BCPD, PCRNet, PointnetLK and DCP).

Parameters settings

The maximum number of iterations of the learning-based optimisation registration methods were set to 30. For *DO* and *RDO*, $\hat{\sigma}^2$ is set as 0.03. The value of the tolerance of absolute difference between current estimation and ground truth in iterations is 1e-4; For *ICP*, the tolerance of absolute difference in translation and rotation is 0.01 and 0.5 respectively; For *CPD*, the type of transformation is set to *rigid*, and the expected percentage of outliers with respect to a normal distribution is 0.1, the tolerance value is the same of that in DO. For *NDT*, the value of the expected percentage of outliers is set to 0.55, and the tolerance value is set as the same as that in ICP. For *BCPD*, the expected percentage of outliers is set to 0.1, the parameter in the Gaussian kernel is 2.0 and the expected length of the displacement vector is 400. All deep-learning-based registration networks are trained on an Nvidia Geforce 2080Ti GPU with 12G memory. For *PCRNet*, the kernel sizes are 64, 64, 64, 128, 1024, 1024, 512, 512, 256 and 7. The iteration for rotation and translation is set to 8. Adam optimiser with an initial learning rate of 0,1, 300 epochs and a batch size of 32 is used for the training process. For *PointnetLK*, the kernel sizes are 64, 64, 64, 128, 1024. The maximum iteration for rotation and translation is set to 30. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 10 is used for the training process. For *DCP*, the kernel sizes are 64, 64, 128, 256, 512, 1024, 256, 128, 64, 32 and 7. The iteration for rotation and translation is set to 1. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 32 is used for the training process.Please note that the values of these parameters are the default values set in their official codes.

Registration Experiments

Comparison on synthetic data sets

3D registration experiments are executed on the Standford Bunny model, Skeleton Hand model, Bimba model and Dancing Children in Figure.5.6. The models are downsampled by selecting almost 1500 points from the original model as the model **S**. The performance of methods are evaluated by comparing the evaluation metrics in the case of various perturbations: *(1) rotation:* The initial angle is 0°, 30°, 60°, 90°, 120° and 150° [default=0°

to 60°]; *(2) noise:* The standard deviation of Gaussian noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]; *(3) outliers:* the number of outliers is set to 0, 100, 200, 300, 400 and 500, respectively [default=0]. The random translation of all generated scenes is within $[-0.3, 0.3]^3$. It is worth noting that when one parameter is changed, the values of other parameters are fixed to the default value. In addition, the scene points are sampled from the original model, not from $\mathbf{S}$. 750 test samples in each variable setting are tested.

Comparison on the ModelNet40 data set

There are three kinds of comparison settings corresponding to the training modes in 5.4.1: for $mode_1$: the initial angle is 0°, 15°, 30°, 45°, 60° and 75°; for $mode_2$: the initial angle is 0°, 30°, 60°, 90°, 120° and 150°; for $mode_3$: the initial angle is 0°, 30°, 60°, 90°, 120° and 150° [default=0° to 90°] and the standard deviation of Gaussian noise is set to 0, 0.02, 0.04, 0.06, 0.08 and 0.1 [default=0]. It is worth noting that when one parameter is changed, the values of other parameters are fixed to the default value. 100 test samples in each variable setting are tested.

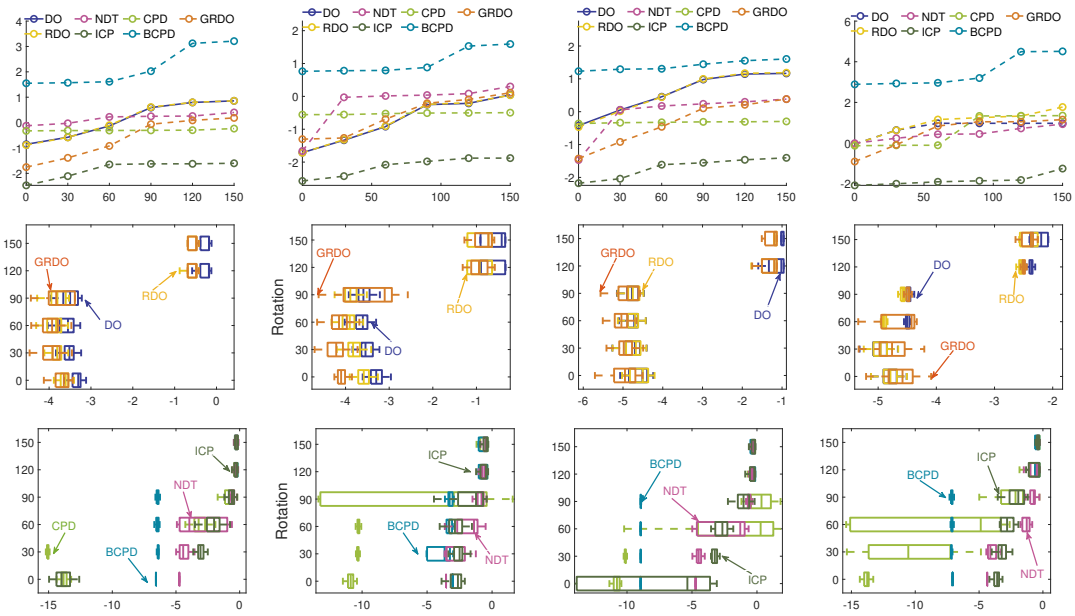
To verify the validity of the transfer learning of the learned updating map $\mathbf{D}_{t+1}$, a comparison experiment with the above settings is also conducted on the Modelnet40 data set. The difference from the above comparison experiments is that the updating map $\mathbf{D}_{t+1}$ is learned from the target point cloud, not the selected training point cloud. This experiment aims to compare the accuracy and robustness of the registration with transfer learning and the registration without transfer learning.

5.4.2 Experimental Results and Discussion

Comparison on synthetic data sets

Figure.5.7 $\sim$ Figure.5.9 show the registration results on Bunny, Skeleton Hand, Bimba Model, and Dancing Children data sets under various perturbations. The top shows the computational time of registration in the presence of rotation. The middle shows the log_{10} MSE of learning-based optimisation registration methods. The bottom shows the log_{10} MSE of other traditional registration methods. It can be seen that BCPD and the learning-based optimisation methods (DO, RDO, GRDO) have the ability to handle the registration with rotation of 90°, other methods fail to register point clouds with the rotation over 60°. Figure.5.7 shows that BCPD and the learning-based optimisation methods (DO, RDO, GRDO) can handle the registration with rotation of 90°, other methods fail to register point clouds with the rotation over 60°. The top rows in Figure.5.7 $\sim$ Figure.5.9 display the log_{10} computational time of all methods to register point clouds under various perturbations. It can be seen that the computational time of BCPD is ten or even hundreds of times that of other methods, although the log_{10} MSE of BCPD is the minimum. The taken computational time and the accuracy of CPD are second only to BCPD. As can be seen from Figure.5.9, the stability of CPD when it handles registration with outliers is poor. By contrast, the learning-based optimisation registration methods (DO, RDO,

and GRDO) can achieve registration with higher stability, and the registration results of GRDO are more accurate. Although ICP takes the least computational time to achieve the registration, the performance of ICP is poor no matter which perturbation the registration with. Overall, GRDO can register point clouds with higher accuracy and more stability within less computational time.



(a) Bunny (b) SkeletonHand (c) BimbaModel (d) DancingChildren

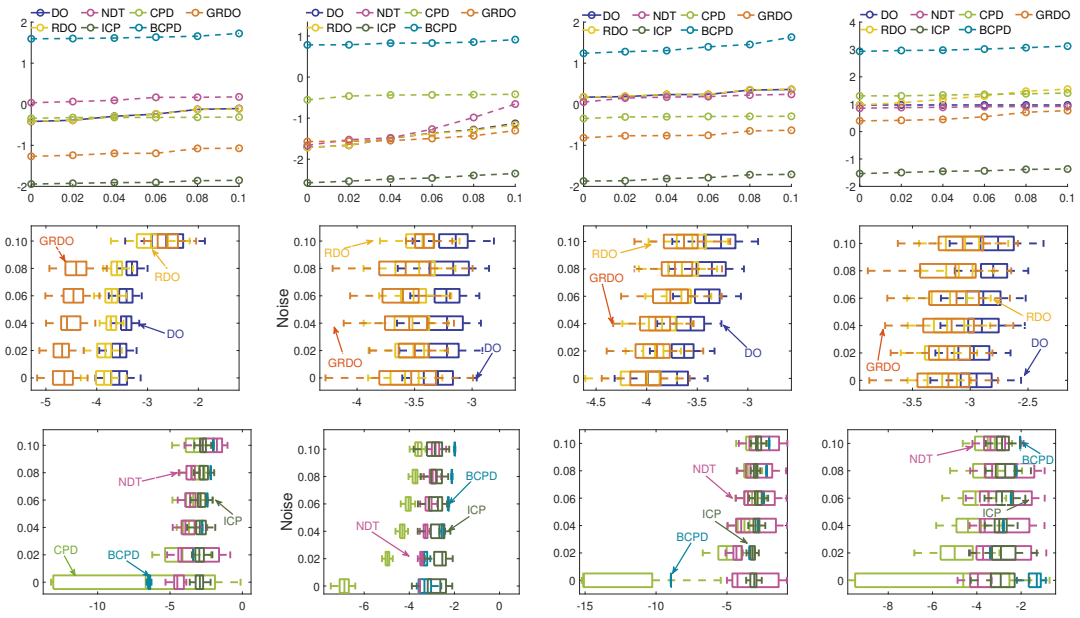
Fig. 5.7 The registration results on different data sets under various rotations.

Table.5.1 shows the log_{10} Mean Square Error of registration under various rotations on synthetic data sets (Bunny model, Skeleton Hand model, Bimba model and Dancing Children model). B means the baseline method - Discriminative Optimisation method (DO); P is the proposed method in this chapter - Graph-based Reweighted Discriminative Optimisation method (GRDO); C is the outstanding conventional method in this registration experiments - Bayesian Coherent Point Drift (BCPD). We set the absolute value of log_{10} Mean Square Error of DO as the reference of comparison. The value of the absolute value of the log_{10} Mean Square Error is higher, and the registration accuracy is higher. Q_0, Q_4 and IQR aim to measure data distribution in boxplot figures. Q_0 and Q_4 are the minimum and maximum respectively. And IQR is the interquartile range, which measures the distance between the upper and lower quartiles; the shorter the distance, the more stability of the method. It can be seen that GRDO has higher accuracy than DO, as shown the bold in this table. GRDO has 41.93% higher accuracy than BCPD. And the performance of GRDO is better than that of BCPD when handling the registration with large rotations (120° and 150°). However, the stability of BCPD is 21.03% higher than that of GRDO when achieving the registration under various rotations.

Table.5.2 shows the log_{10} Mean Square Error of registration under various noises on synthetic data sets (Bunny model, Skeleton Hand model, Bimba model and Dancing Children model). C is the outstanding conventional method in this registration experiment - Bayesian Coherent Point Drift (BCPD). It can be seen that GRDO has higher accuracy

Table 5.1 The quantitative registration results under various rotations. (B-Baseline method DO; P-Proposed method GRDO; C-Conventional method BCPD)

		Q_0			Q_4			IQR		
		B	P	C	B	P	C	B	P	C
Bunny	(R)90	1.00	1.10	**1.59**	1.00	1.03	**1.97**	1.00	1.33	**0.33**
	(R)120	1.00	**1.43**	0.86	1.00	**5.00**	4.00	1.00	1.00	1.00
	(R)150	1.00	**2.00**	1.20	1.00	**5.00**	4.00	1.00	1.00	1.00
Skeleton Hand	(R)90	1.00	**1.26**	1.03	1.00	0.84	**0.94**	1.00	3.33	**0.67**
	(R)120	1.00	**1.60**	1.00	1.00	**4.00**	2.00	1.00	**0.57**	0.86
	(R)150	1.00	**1.36**	0.91	1.00	**4.50**	2.00	1.00	**0.57**	0.86
Bimba Model	(R)90	1.00	1.12	**1.54**	1.00	1.09	**1.81**	1.00	1.40	**0.40**
	(R)120	1.00	**1.11**	0.33	1.00	1.30	0.40	1.00	0.80	**0.40**
	(R)150	1.00	**1.50**	0.50	1.00	**1.30**	0.40	1.00	1.50	**1.00**
Dancing Children	(R)90	1.00	0.98	**1.74**	1.00	0.98	**1.73**	1.00	**1.00**	2.00
	(R)120	1.00	**1.08**	0.24	1.00	**1.14**	0.23	1.00	**1.00**	2.00
	(R)150	1.00	**1.20**	0.25	1.00	**1.20**	0.25	1.00	**0.67**	0.67



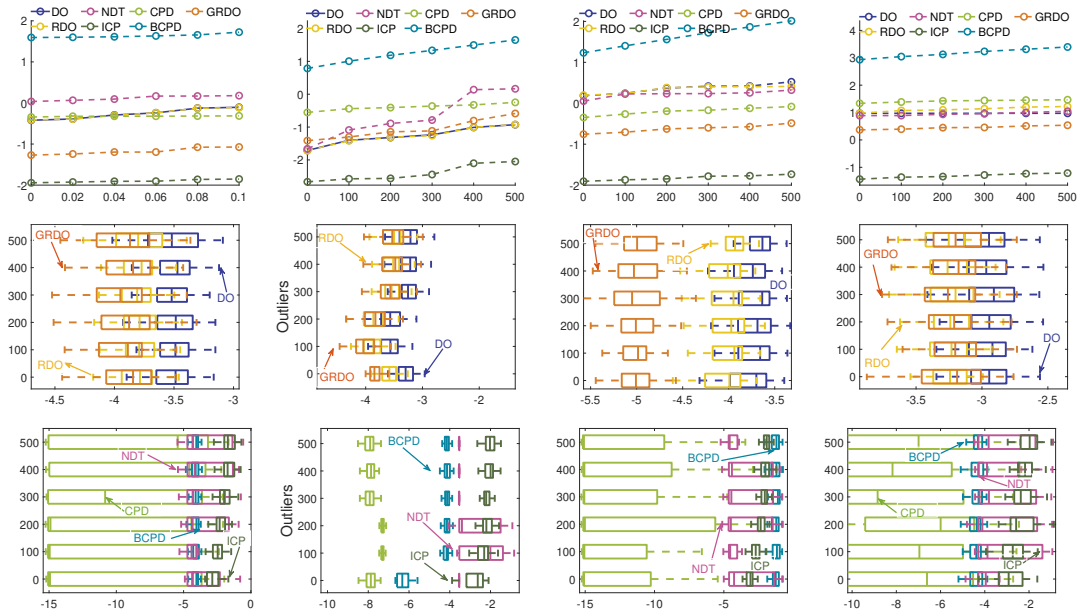
(a) Bunny (b) SkeletonHand (c) BimbaModel (d) DancingChildren

Fig. 5.8 The registration results on different data sets under varying degrees of noise.

than DO, as shown the bold in this table. The accuracy of GRDO when registering point clouds under various noises is almost twice that of BCPD and is 15.33% higher than that of DO. However, the stability of BCPD is the best among these three methods.

Table 5.2 The quantitative registration results under various Noises. (B-Baseline method DO; P-Proposed method GRDO; C-Conventional method BCPD)

		Q_0			Q_4			IQR		
		B	P	C	B	P	C	B	P	C
Bunny	(N)0.06	1.00	**1.33**	0.72	1.00	**1.30**	0.76	1.00	1.67	**0.33**
	(N)0.08	1.00	**1.38**	0.74	1.00	**1.35**	0.77	1.00	1.67	**0.33**
	(N)0.10	1.00	**0.93**	0.80	1.00	**1.09**	1.00	1.00	0.43	**0.14**
Skeleton Hand	(N)0.06	1.00	**1.14**	0.64	1.00	**1.12**	0.76	1.00	2.00	**1.00**
	(N)0.08	1.00	**1.22**	0.57	1.00	**1.07**	0.71	1.00	1.33	**0.33**
	(N)0.10	1.00	**1.03**	0.56	1.00	**1.20**	0.70	1.00	**0.50**	1.00
Bimba Model	(N)0.06	1.00	**1.13**	0.64	1.00	**1.03**	0.79	1.00	1.33	**0.33**
	(N)0.08	1.00	**1.10**	0.56	1.00	**1.08**	0.71	1.00	1.00	**0.33**
	(N)0.10	1.00	**1.11**	0.56	1.00	**1.20**	0.71	1.00	1.00	**0.25**
Dancing Children	(N)0.06	1.00	**1.12**	0.76	1.00	**1.04**	0.92	1.00	1.22	**0.22**
	(N)0.08	1.00	**1.25**	0.75	1.00	**1.08**	0.92	1.00	2.00	**0.33**
	(N)0.10	1.00	**1.10**	0.71	1.00	**1.08**	0.89	1.00	1.18	**0.22**



(a) Bunny (b) SkeletonHand (c) BimbaModel (d) DancingChildren

Fig. 5.9 The registration results on different data sets under varying numbers of outliers.

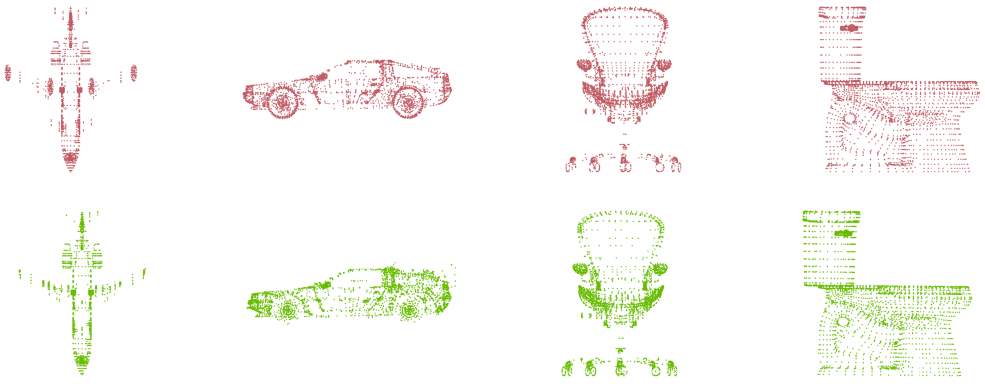
Table.5.3 shows the log_{10} Mean Square Error of registration under various outliers on synthetic data sets (Bunny model, Skeleton Hand model, Bimba model and Dancing Children model). C is the outstanding conventional method in this registration experiment - Bayesian Coherent Point Drift (BCPD). It can be seen that the accuracy of GRDO is similar to the accuracy of BCPD when achieving the registration under various outliers. They both have almost 25.00% higher accuracy than DO. Nevertheless, the stability of BCPD is the best among these three methods.

Table 5.3 The quantitative registration results under various Outliers. (B-Baseline method DO; P-Proposed method GRDO; C-Conventional method BCPD)

		Q_0			Q_4			IQR		
		B	P	C	B	P	C	B	P	C
Bunny	(O)300	1.00	1.39	**1.51**	1.00	1.31	**1.54**	1.00	1.60	**0.40**
	(O)400	1.00	1.38	**1.59**	1.00	1.38	**1.60**	1.00	1.60	**0.40**
	(O)500	1.00	1.38	**1.54**	1.00	1.38	**1.62**	1.00	1.50	**0.25**
Skeleton Hand	(O)300	1.00	1.24	**1.29**	1.00	1.21	**1.43**	1.00	2.33	**0.67**
	(O)400	1.00	1.18	**1.26**	1.00	1.28	**1.48**	1.00	1.40	**0.40**
	(O)500	1.00	1.18	**1.25**	1.00	1.26	**1.48**	1.00	1.14	**0.29**
Bimba Model	(O)300	1.00	**1.43**	0.71	1.00	**1.27**	0.29	1.00	1.75	**0.50**
	(O)400	1.00	**1.29**	0.66	1.00	**1.27**	0.29	1.00	0.80	**0.40**
	(O)500	1.00	**1.38**	0.70	1.00	**1.30**	0.29	1.00	1.14	**0.57**
Dancing Children	(O)300	1.00	1.10	**1.47**	1.00	1.06	**1.54**	1.00	1.00	**0.40**
	(O)400	1.00	1.09	**1.55**	1.00	1.12	**1.60**	1.00	1.13	**0.50**
	(O)500	1.00	1.10	**1.47**	1.00	1.06	**1.54**	1.00	1.25	**0.50**

Comparison on the ModelNet40 data set

Figure.5.10 shows the search results for a given point cloud. The top shows the given target point clouds. The bottom shows the selected similar training point clouds for the given models. This figure shows that the selected training point cloud is similar to the given point cloud, which illustrates the effectiveness of the proposed search framework.



(a) Airplane (b) Car (c) Chair (d) Toilet

Fig. 5.10 The searching results for the given target models.

The registration results of the target point cloud via the learned regressors of the selected training point cloud are shown in Figure.5.11 $\sim$ Figure.5.13. Please note that, for the learning-based optimisation methods (DO, RDO, and GRDO), the registration results here are obtained through the learned regressors of the selected training point clouds, which are different from the settings of the comparative experiments with traditional methods.

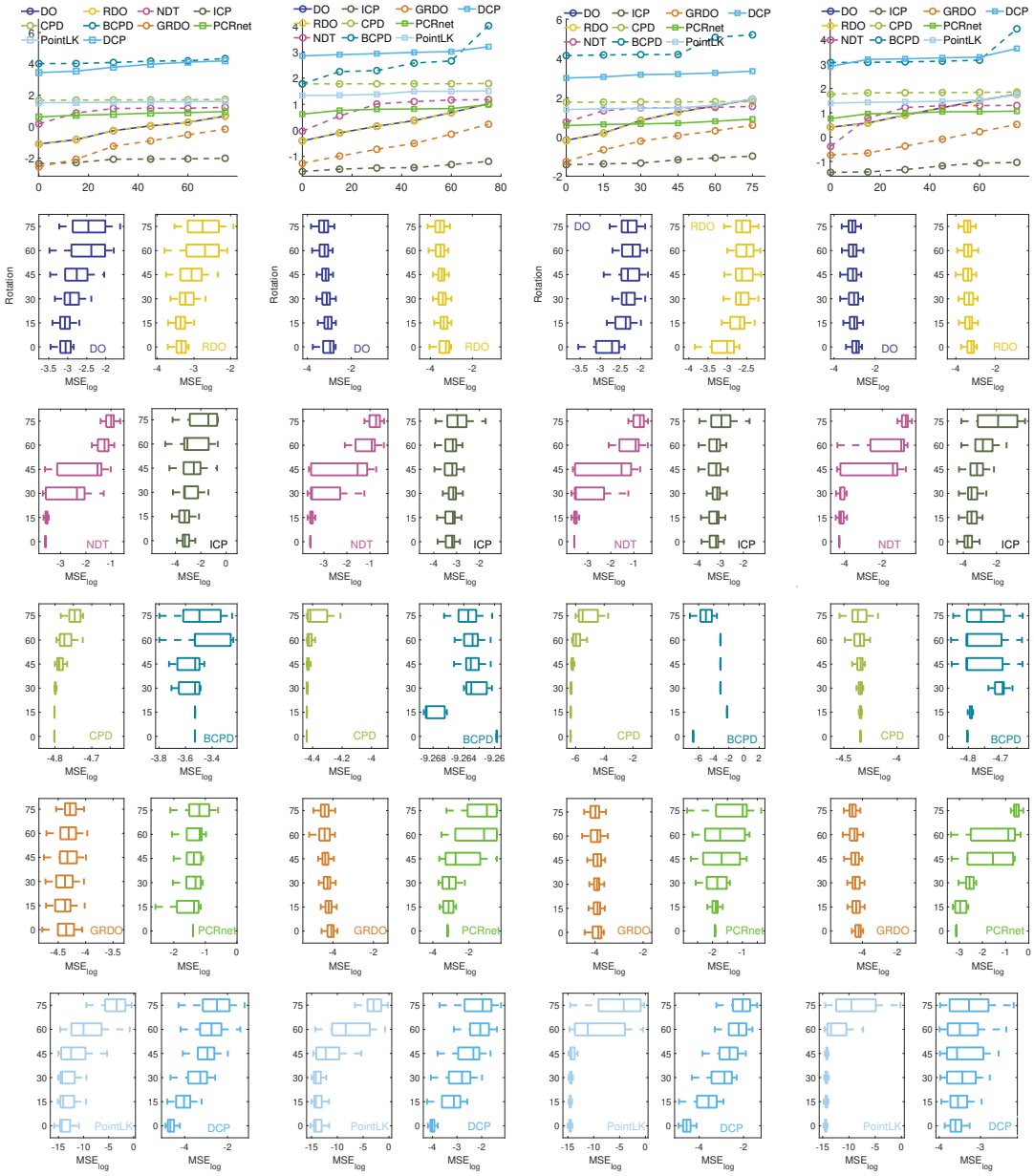
Figure.5.11 displays the performance of all methods on registration with $mode_1$. $mode_1$ depicts the point clouds registration with small rotations. Top shows the log_{10} compu-

tational time of all comparative methods; Second $\sim$ Bottom illustrate the log_{10} MSE of all registration methods. It can be seen that BCPD and DCP, CPD, and PointnetLK take about the same amount of time to register point clouds, respectively. The registration accuracy of DCP is poor than that of BCPD. The stability of CPD is better than that of PointnetLK, although the registration result of PointnetLK is more accurate when the rotation angle is small (not over 60°). The computational time of PCRnet is close to that of the learning-based optimisation methods (DO and RDO), the registration accuracy of PCRnet is poor by contrast. Also as a learning-based optimisation method, GRDO can register point clouds with higher accuracy within less computational time while maintaining high stability. It is challenging for NDT to handle registration over 60°, and the performance of ICP on registration with the rotation of 75° is poor. In summary, compared with the deep learning methods, the traditional methods can achieve more accurate registration when the rotation is 75°; compared with the traditional methods, the benefits of the learning-based optimisation methods (DO, RDO, and GRDO) are higher stability and less computation time.

Figure.5.12 shows the registration results of all methods with the $mode_2$ settings, reflecting the performance of all methods on the point clouds registration under large rotations. BCPD, CPD and other learning-based optimisation methods can handle the registration with 90°, and even the registration results of them on 120° are more accurate. PCRnet, PointnetLK, and DCP are unable to keep higher accuracy when dealing with registration with the rotation over 60°, which also is reflected by the registration results of NDT and ICP. A similar situation also occurs in the learning-based optimisation methods. The difference is that when DO, RDO, and GRDO handle the registration with the rotation over 120°, the registration accuracy will fall off a cliff. The reason is that the learning-based optimisation methods have not learned the feature regarding the relative position when the point cloud rotates at a large angle. The rotation parameter set during the training process is only 90 degrees. Nonetheless, the registration accuracy of GRDO is still higher than DO and RDO.

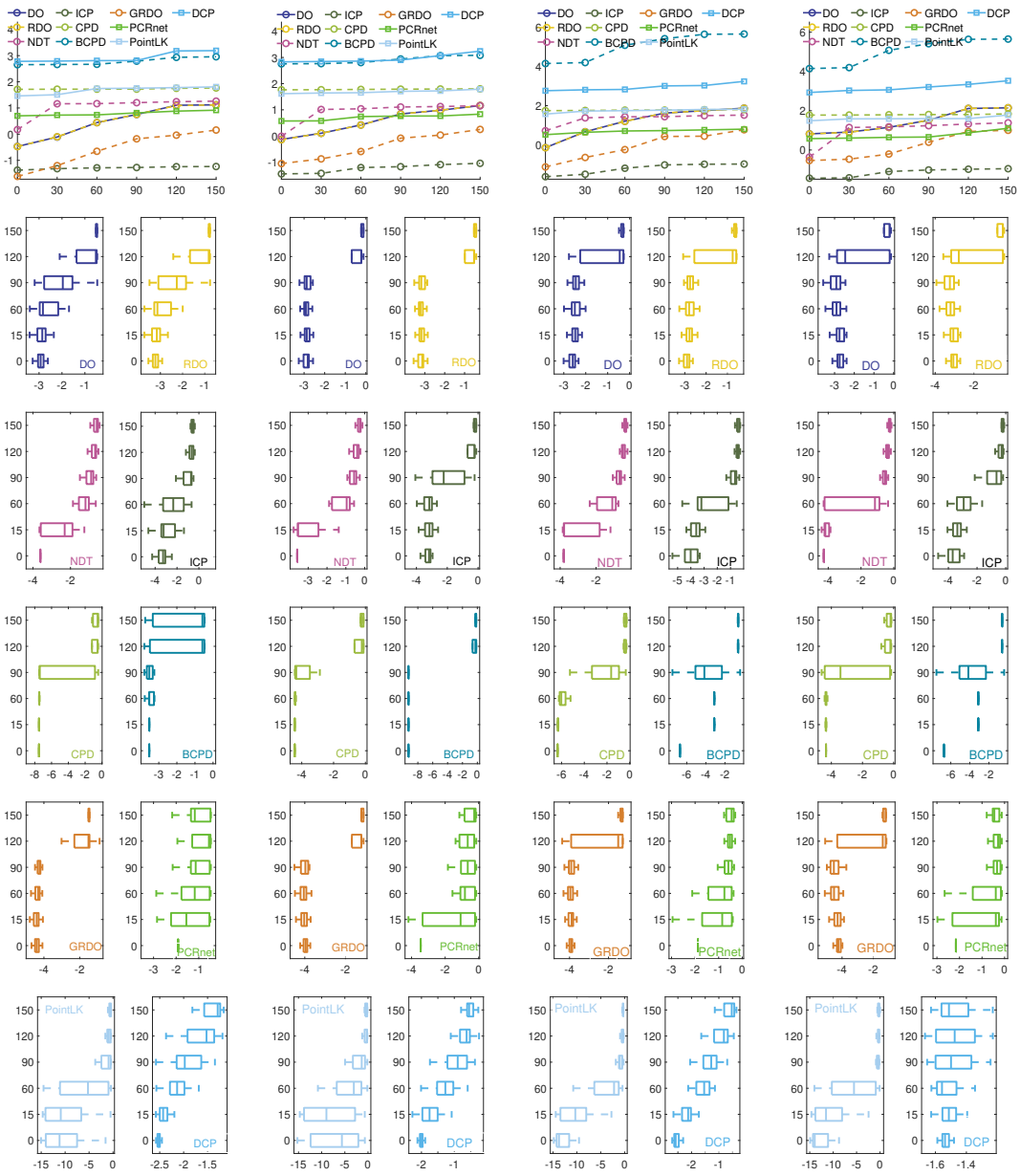
Figure.5.13 shows the performance of all methods on the registration with $mode_3$ (multiple perturbations). Specifically, the point clouds rotate arbitrarily in the range of 90° with varying extents of noise perturbation. DO, RDO, and GRDO are more robust in terms of noise compared with other methods, and the existence of noise has little effect on their performance. GRDO still keeps its higher accuracy, higher stability, and less computational time. Comparing the registration results of them under the setting of $mode_2$ and $mode_3$, it can be found that the performance of CPD and BCPD is highly impacted by noise. The accuracy and the stability of NDT and ICP are more vulnerable under multiple perturbations. Besides, for PCRnet, PointnetLK and DCP, the higher the extent of the noise perturbation, the worse the registration performance.

Table.5.4 shows the log_{10} Mean Square Error of registration on the ModelNet40 data set. C is the outstanding conventional method in this registration experiment - Bayesian Coherent Point Drift (BCPD). D is the deep learning-based method with better performance - PointNetLK. When handling the registration with small rotations ($mode_1$), PointNetLK has



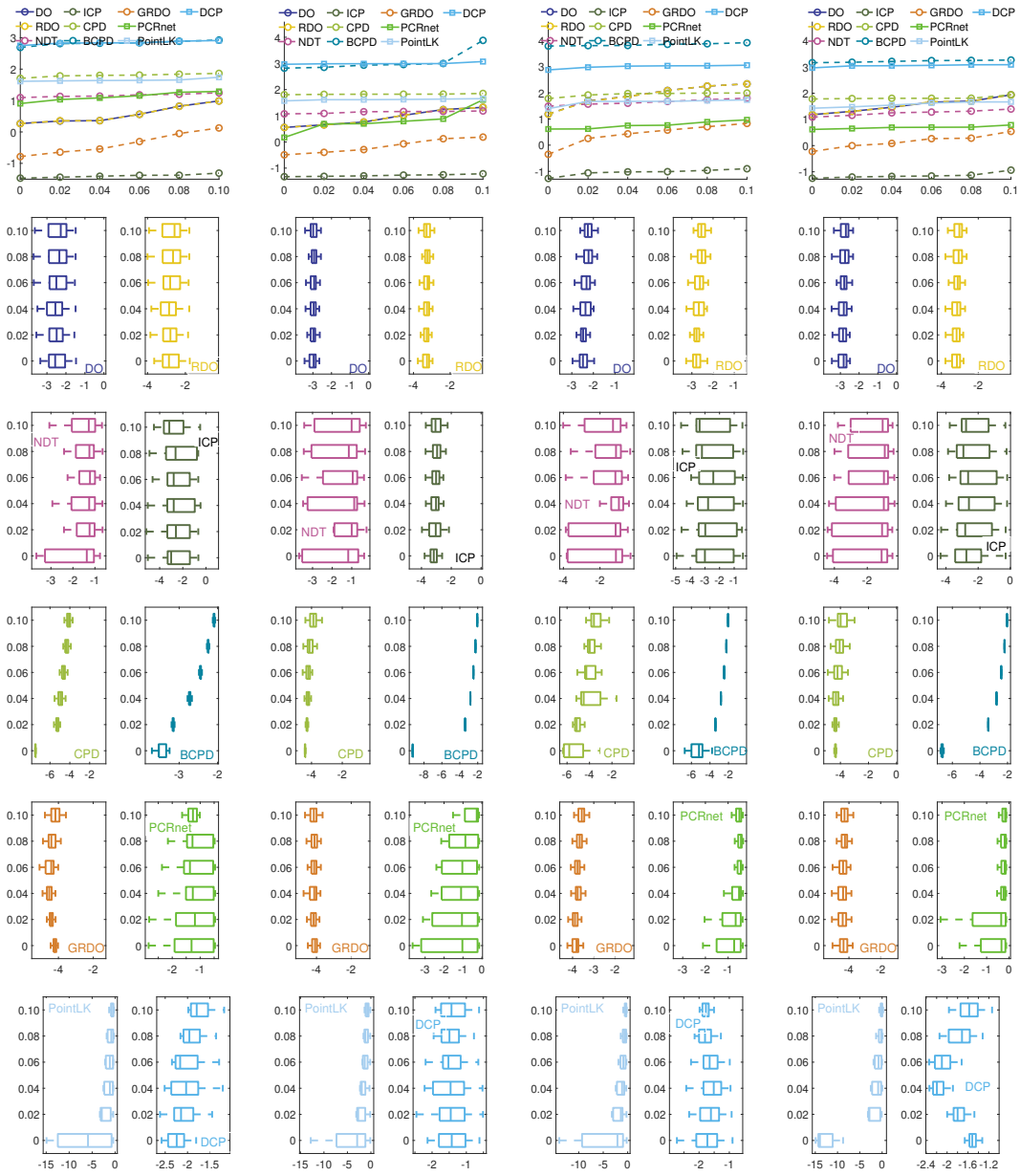
(a) Airplane (b) Car (c) Chair (d) Toilet

Fig. 5.11 The registration results of different data sets with perturbation setting $mode_1$.



(a) Airplane (b) Car (c) Chair (d) Toilet

Fig. 5.12 The registration results of different data sets with perturbation setting $mode_2$.



(a) Airplane (b) Car (c) Chair (d) Toilet

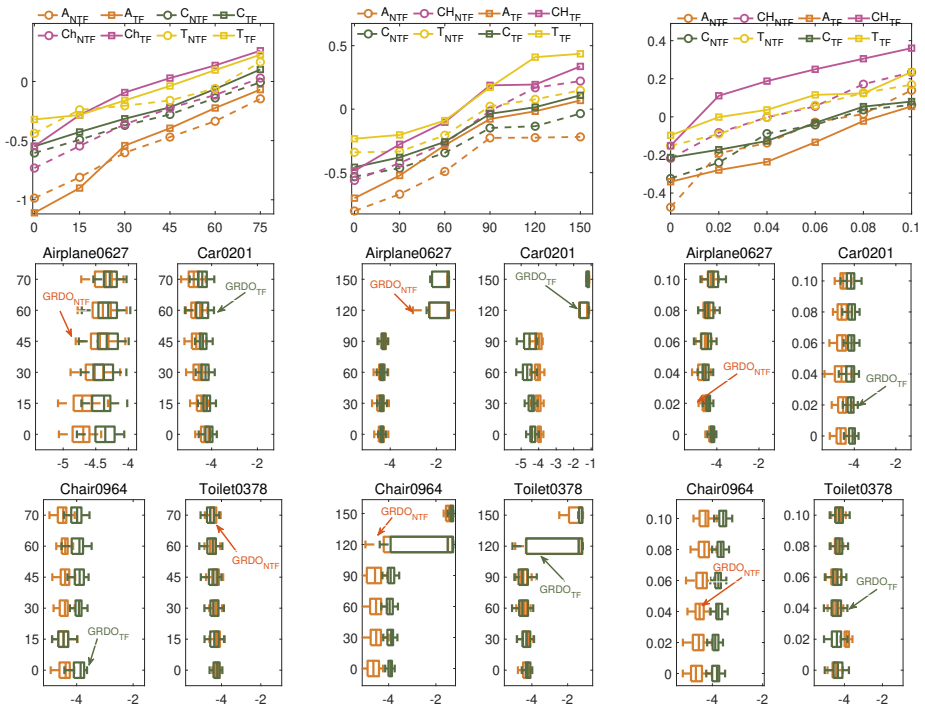
Fig. 5.13 The registration results of different data sets with perturbation setting $mode_3$.

higher accuracy than other methods, but its stability is the lowest among these comparative methods. Although the accuracy of DRDO is similar to that of BCPD, its stability is not good as BCPD. When handling the registration with large rotations ($mode_2$), GRDO has 4.78% higher accuracy than BCPD and 15.06% higher accuracy than PointNetLK. And the stability of GRDO is 31.02% higher than that of BCPD. GRDO has the greatest accuracy and stability than other methods when achieving the registration with multiple perturbations ($mode_3$: rotation and noises).

Table 5.4 The quantitative registration results on the ModelNet40 data set. (B-Baseline method DO; P-Proposed method GRDO; C-Conventional method BCPD; D- Deep-learning method PointnetLK)

			Q_0				Q_4				IQR			
			B	P	C	D	B	P	C	D	B	P	C	D
$mode_1$	Airplane	(R)45	1.00	1.37	1.06	**4.29**	1.00	2.10	1.75	**2.50**	1.00	0.60	**0.40**	8.00
		(R)60	1.00	1.37	1.09	**4.29**	1.00	**2.22**	1.78	0.28	1.00	**0.25**	0.38	7.50
		(R)75	1.00	1.44	1.19	**3.13**	1.00	**2.67**	2.20	0.33	1.00	**0.19**	0.38	5.00
	Car	(R)45	1.00	1.32	2.44	**3.95**	1.00	1.43	**3.31**	1.79	1.00	1.00	**0.004**	10.00
		(R)60	1.00	1.38	2.32	**3.75**	1.00	1.43	**3.31**	0.18	1.00	0.63	**0.003**	6.25
		(R)75	1.00	1.33	**2.32**	1.75	1.00	1.43	**3.31**	0.18	1.00	0.63	**0.005**	3.75
	Chair	(R)45	1.00	1.37	1.00	**5.00**	1.00	2.53	1.87	**8.67**	1.00	0.40	**0.40**	2.00
		(R)60	1.00	1.53	1.09	**5.45**	1.00	**2.53**	1.87	0.67	1.00	0.60	**0.40**	18.00
		(R)75	1.00	1.53	2.00	**5.45**	1.00	1.90	**2.00**	0.50	1.00	**0.50**	2.00	22.50
	Toilet	(R)45	1.00	1.25	1.21	**3.75**	1.00	1.33	1.54	**4.33**	1.00	0.40	**0.20**	2.00
		(R)60	1.00	1.25	1.21	**3.75**	1.00	1.43	1.65	**2.68**	1.00	0.60	**0.20**	10.00
		(R)75	1.00	1.25	1.21	**3.75**	1.00	1.33	**1.54**	0.33	1.00	0.50	**0.25**	18.75
$mode_2$	Airplane	(R)90	1.00	**1.36**	1.15	1.52	1.00	**7.17**	5.33	1.67	1.00	**0.07**	0.67	1.67
		(R)120	1.00	1.36	**1.73**	1.36	1.00	**1.71**	0.71	1.43	1.00	**0.10**	0.10	0.60
		(R)150	1.00	2.50	**4.75**	1.25	1.00	**3.00**	0.71	1.14	1.00	1.00	1.00	4.00
	Car	(R)90	1.00	1.36	**2.73**	1.52	1.00	1.56	**3.56**	0.40	1.00	**0.07**	0.07	2.00
		(R)120	1.00	**2.00**	1.00	2.00	1.00	**5.00**	4.50	5.00	1.00	0.80	**0.20**	1.00
		(R)150	1.00	**3.36**	3.33	3.67	1.00	**4.50**	4.50	5.00	1.00	1.00	1.00	10.00
	Chair	(R)90	1.00	1.33	**2.67**	0.83	1.00	**1.90**	0.10	0.85	1.00	**0.50**	0.50	2.50
		(R)120	1.00	**1.40**	0.07	0.67	1.00	2.50	0.25	**4.25**	1.00	0.40	**0.10**	0.15
		(R)150	1.00	1.71	0.29	**2.57**	1.00	2.50	0.25	**4.25**	1.00	1.00	1.00	1.00
	Toilet	(R)90	1.00	1.26	**1.84**	0.53	1.00	**1.74**	0.09	0.74	1.00	0.75	5.00	**0.50**
		(R)120	1.00	**1.43**	0.23	0.57	1.00	**10.00**	7.00	17.00	1.00	0.93	**0.07**	0.11
		(R)150	1.00	**2.86**	1.14	2.57	1.00	**10.00**	7.00	17.00	1.00	0.25	**0.25**	0.25
$mode_3$	Airplane	(N)0.06	1.00	**1.28**	0.72	0.64	1.00	**2.67**	1.67	0.67	1.00	0.50	**0.33**	1.33
		(N)0.08	1.00	**1.23**	0.62	0.64	1.00	**2.53**	1.53	0.67	1.00	0.50	**0.33**	0.83
		(N)0.10	1.00	**1.23**	0.56	0.51	1.00	**2.40**	1.40	0.67	1.00	0.43	**0.14**	0.29
	Car	(N)0.06	1.00	**1.15**	0.77	0.64	1.00	**1.36**	0.68	0.36	1.00	0.75	**0.25**	1.00
		(N)0.08	1.00	**1.29**	0.80	0.57	1.00	**1.36**	0.68	0.36	1.00	1.50	**0.50**	1.50
		(N)0.10	1.00	**1.23**	0.69	0.51	1.00	**1.36**	0.68	0.36	1.00	0.75	**0.25**	0.50
	Chair	(N)0.06	1.00	**1.40**	1.00	0.83	1.00	**1.71**	1.38	0.62	1.00	0.60	**0.20**	0.60
		(N)0.08	1.00	**1.40**	0.93	0.73	1.00	**1.75**	1.35	0.65	1.00	0.75	**0.25**	0.50
		(N)0.10	1.00	**1.50**	0.96	0.68	1.00	**1.75**	1.30	0.65	1.00	0.60	**0.20**	0.20
	Toilet	(N)0.06	1.00	**1.67**	0.93	0.83	1.00	**1.73**	1.32	0.59	1.00	1.50	**0.50**	1.50
		(N)0.08	1.00	**1.37**	0.77	0.63	1.00	**1.73**	1.27	0.59	1.00	1.50	**0.50**	1.25
		(N)0.10	1.00	**1.37**	0.71	0.57	1.00	**1.73**	1.18	0.59	1.00	3.00	**1.00**	2.00

Figure.5.14 shows the registration results of $GRDO_{TF}$ (GRDO with transfer learning) and $GRDO_{NTF}$ (GRDO without transfer learning) on ModelNet40 data sets. Left shows the registration results with the setting $mode_1$. Middle depicts the registration results with the setting $mode_2$. Right presents the comparison results with the setting $mode_3$. Top displays the comparison of computation time. The computation time for registration via $GRDO_{TF}$ is presented by the solid line with squares. The dashed line with the circle shows the computation time of $GRDO_{NTF}$. The log_{10} MSE of the $GRDO_{TF}$ and $GRDO_{NTF}$ are presented by the orange and the dark green respectively. It can be seen that the registration accuracy of $GRDO_{NTF}$ is better, and the robustness and the stability of both are similar. The computational time of $GRDO_{NTF}$ is less than that of $GRDO_{TF}$ in most cases.



(a) $mode_1$ (b) $mode_2$ (c) $mode_3$

Fig. 5.14 The comparison of GRDO with transfer learning ($GRDO_{TF}$) and GRDO without transfer learning ($GRDO_{NTF}$) under different perturbation settings.

Discussion. Compared with the deep-learning methods, the learning-based optimisation methods (DO, RDO, and GRDO) can achieve registration with more stability and robustness, especially under multiple perturbations. This is because the deep-learning methods largely belong to the data-driven model. The addition of perturbations (such as noises, and outliers) makes it challenging for networks to converge to an optimal. The learning-based optimisation methods are model-driven, and the performance depends on the designed feature. Besides, the learning-based optimisation methods and the deep-learning methods can hardly handle the registration with large rotations, which may be due to the local optimisation or the over-fitting issues.

Transfer learning is the critical technique to achieve multiple point clouds registration for the learning-based optimisation methods. Although the accuracy of GRDO with transfer learning is less than that of the GRDO without transfer learning, the GRDO with transfer learning still maintains high robustness and stability. And the accuracy of GRDO registration based on transfer learning is better than that of most comparative registration methods.

5.5 Discussion

Although learning-based optimisation methods can not deal with the multiple point clouds registration within a limited time, SGRTmreg provides a new perspective for them to achieve multiple point clouds registration. Maybe a more general feature is a breakthrough to increase the number and the kinds of samples to be registered within a limited period.

Compared with DO and RDO, the storage requirement of GRDO for learning the updating map is reduced. The memory requirement for learning $\mathbf{D_{t+1}}$ in DO and RDO is $O\left(N\left(((c_1+c_2)N_M+c_2N_S)\right)+c_3N_M^2\right)$ [109], which largely depends on the number of points in a point cloud. The feature in this framework SGRTmreg is designed based on the extracted key points, and the way to extract key points keeps the details of the point cloud. The usage of key points greatly reduces the requirement for storage space.

5.6 Conclusion

This chapter describes a framework SGRTmreg composed of a novel learning-based optimisation algorithm that shares the same principles with deep learning for learning regressors, a searching scheme for selecting point clouds, and transfer learning for applying the learned regressors. Specifically, given a target point cloud and a collection of training point clouds, a search scheme is to find a similar training point cloud for the target point cloud from the collection; the Graph-based Reweighted Discriminative optimisation method (GRDO) is used to learn a sequence of regressors from the selected similar training point cloud; transfer learning focuses on storing the learned sequence of regressors while applying it to the transformation estimation of the target point cloud. The key point in SGRTmreg is that the learned gradient direction is from a selected training point cloud that is different but similar to the target point cloud. The potential of the framework SGRTmreg is demonstrated in 3D multiple point clouds registration with higher accuracy and robustness, and efficiency than the state-of-the-art deep learning-based methods and traditional optimisation-based registration methods. The registration accuracy of GRDO is almost 41.93% higher than that of BCPD on the registration of synthetic data sets under various rotations. The accuracy of GRDO when registering point clouds under various noises is almost twice that of BCPD and is 15.33% higher than that of DO. GRDO has almost 25.00% higher accuracy than DO when achieving the registration under various outliers. When handling the registration with large rotations, GRDO has 4.78% higher accuracy than BCPD and 15.06% higher accuracy than PointNetLK. In addition, GRDO has the greatest accuracy and stability than other methods when achieving the registration with multiple perturbations, and the computation time of GRDO is 1/10 of that of DO. Future work is to design a generalised representation of parameter vectors that is suitable for computer vision and graphics applications other than those specific to the Lie Algebra for a rigid transformation matrix. On this basis, the learning-based optimisation algorithm (GRDO) together with the search scheme and transfer learning can be applied to a wider range of problems in computer graphics and computer vision, such as non-rigid registration, and image denoising, and more.

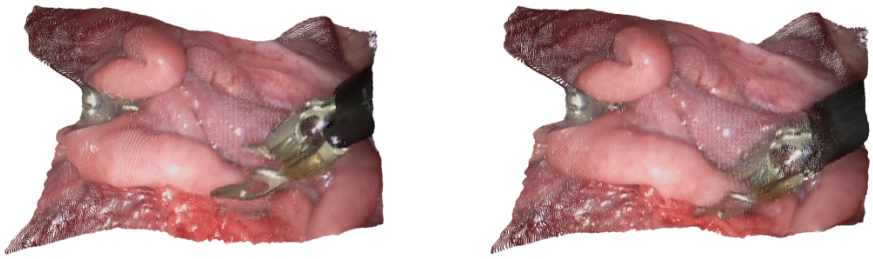
Chapter 6

Application on Computer Vision

The data sets for conducting the point clouds registration experiments in the previous three chapters are most of the public, synthetic data sets, which are not convincing to evaluate the performance of these proposed methods on augmented reality applications. Because the augmented reality technique involves object tracking in real scenes. And the object tracking is achieved via the point cloud registration. In this case, this chapter applies the previously proposed algorithms (GDO, RDO, and GRDO) to achieve the point clouds registration and object tracking in real scenes. Specifically, the rigid registration of medical instruments in the intraoperative scene is accomplished. The registration between models and scenes reconstructed through the images captured by Kinect is also conducted. And the experiment on object tracking is also carried out in this chapter.

6.1 Registration of Medical Instruments

The point clouds of the medical intraoperative scene can be attained from the previous work [110], as shown in Figure.6.1. Specifically, the intraoperative medical scene is reconstructed from the adjacent frames of the public available Hamlyn Centre Laparoscopic/Endoscopic video data sets [111]. It can be seen that the organ and the instrument are merged. And the organ has obscured the partial instrument (as shown in Figure.6.1(b)). Due to the proposed methods focusing on rigid registration, it is necessary to segment the instruments from the reconstructed scenes. Therefore, a segmentation scheme to separate the instruments is proposed. The scheme can be divided into two stages: rough segmentation and fine segmentation. The former is to separate the instruments from the organs roughly via Dirichlet Process Gaussian Mixture Models (DPGMM) based on the colour and the coordinates of point clouds, and the latter is to remove the remaining points in the segmented area.



(a) (b)

Fig. 6.1 The point clouds of the medical intraoperative scene.

6.1.1 Point Cloud Segmentation

Rough Segmentation

There are three key steps to achieve the rough segmentation: (1) The medical scene will be clustered via the colour information $colour_\beta = \frac{median(Scene_C)}{max(Scene_C)}$, where $Scene_C$ is the colour vector of the medical scene, and the number of clusters depends on the number of different values of $colour_\beta$. (2) DPGMM (Dirichlet Process Gaussian Mixture Models) is used for the partition of the clusters. Please note that DPGMM acts on the combination of colour and coordinates information $\alpha \times Scene_C + \rho \times Scene_P$, where $Scene_P$ is the coordinates vector of the cluster grouped through the $colour_\beta$. And α and ρ are coefficients, $\alpha + \rho = 1$. The criteria for assigning weights are to keep the size of the combination the same as that of the coordinates in the cluster. The partition results will be manually merged as the organ and the instrument. (3) The denoising technique [112] in Matlab will be applied to remove the noise in merged results. As shown in Figure.6.2.

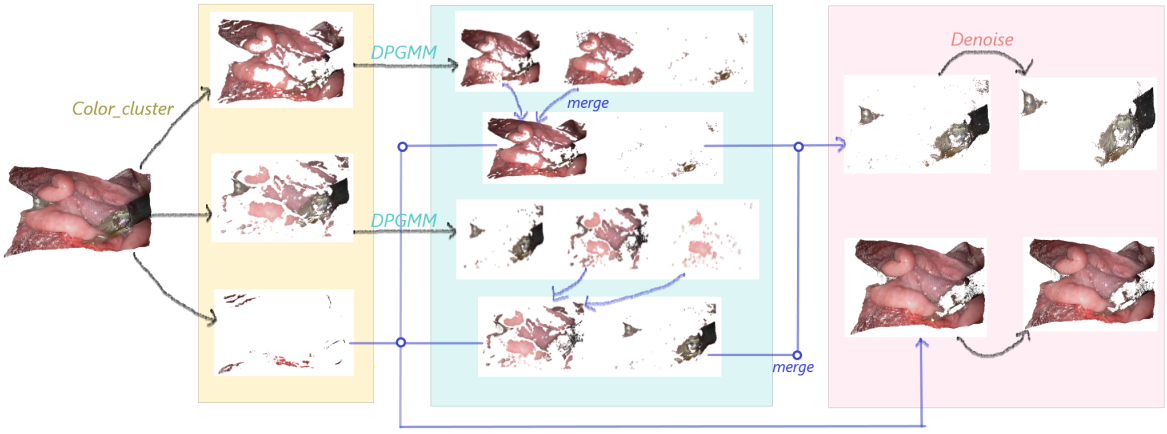


Fig. 6.2 The framework for rough segmentation.

Fine Segmentation

To completely segment the instruments from the medical scene, a fine segmentation scheme is designed to remove the remaining points in the blank partitioned area. The function

fitgmdist in Matlab is used to fit Gaussian mixture models to the organ data. Then the boundary of each component of the Gaussian mixture models will be extracted [113], and the boundary points will be selected if the distribution of their normals is irregular. Univariate linear regression will be applied to the selected boundary points to generate a linear model to screen those points to be removed. As shown in Figure.6.3.

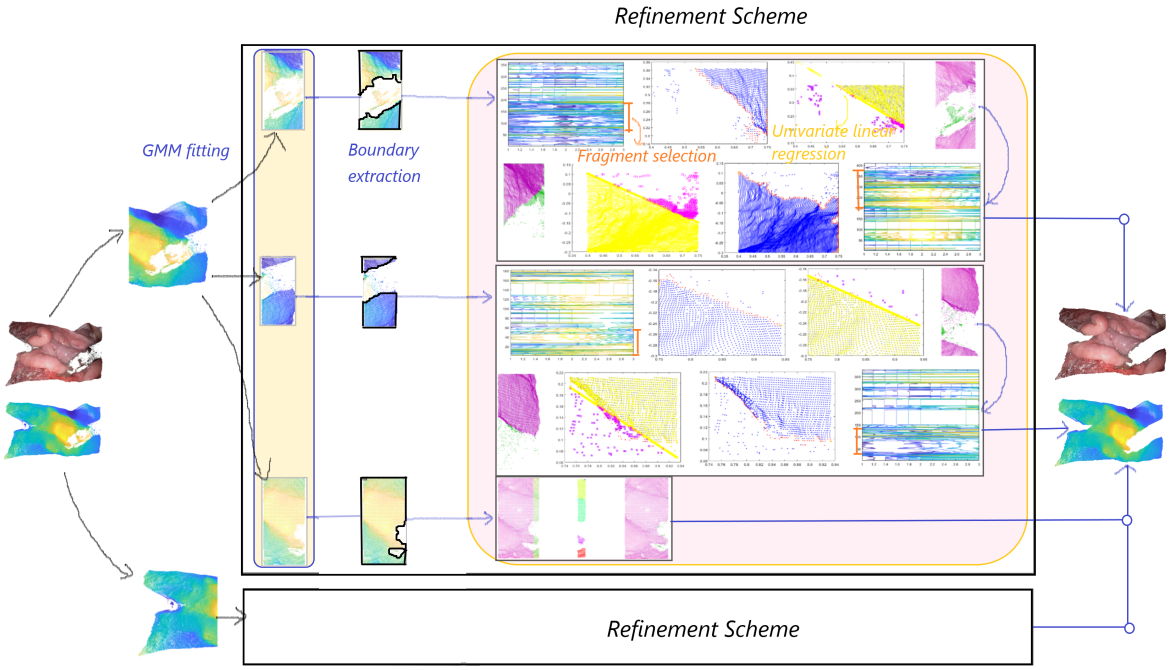
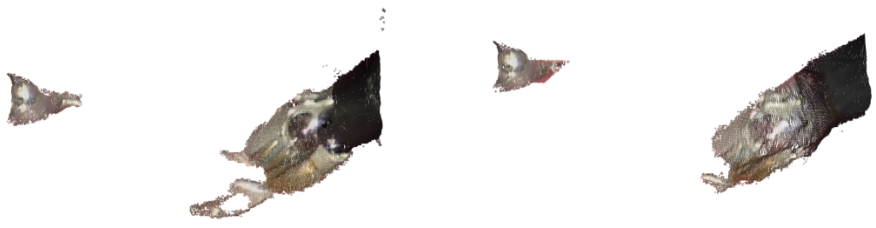


Fig. 6.3 The framework for fine segmentation.

Segmentation Results

Figure.6.4 displays that the segmented instrument is incomplete and has outliers. The reason causing the incompleteness of the instrument is that the colour of the instrument's tip in Figure.6.1 is similar to that of the organ near the tip, and the colour is darker. The difference between the colours of the tip and the organ is not apparent, which leads to the poor performance of the cluster that is only based on colour information.



(a) (b)

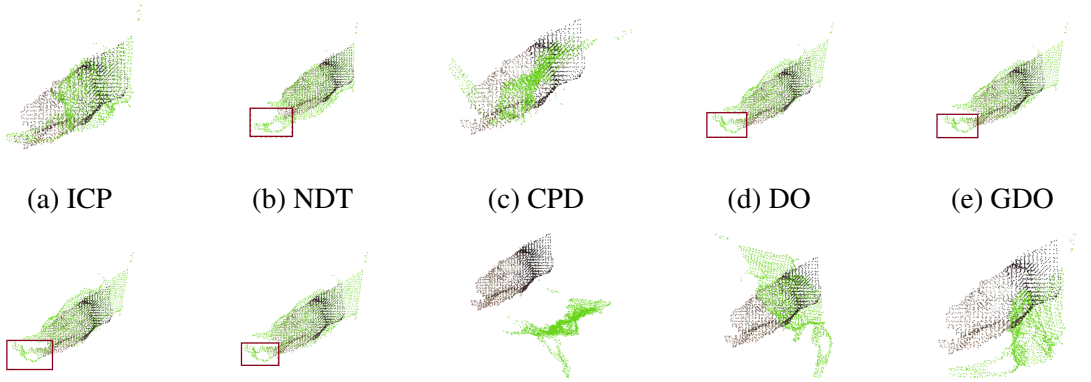
Fig. 6.4 The segmentation results of instruments.

6.1.2 Parameter Settings

The parameters in the training processes of *DO*, *GDO*, *GDO* and *GRDO* are the same in the instruments registration experiment. A given model shape $\mathbf{M}$ is normalised to $[-1,1]^3$ and a scene model is generated through uniformly sampling from $\mathbf{M}$ with the replacement of almost 1500 points. Then the following perturbations are applied to the scene model: *(i) Rotation and translation:* The rotation is within 60 degrees and the translation is in $[-0.2,0.2]^3$; *(ii) Noise and Outliers:* Gaussian noise with the standard deviation 0.05 is added to the scene model. 0 to 300 points within $[-1.5,1.5]^3$ are added as the sparse outliers. For all experiments, 30000 training samples are generated, the number of iterations is set to $T = 30$, and the initial transformation $\mathbf{x}_0$ is $\mathbf{0}^6$. And $\hat{\sigma}^2$ is set as 0.03. The value of the tolerance of absolute difference between current estimation and ground truth in iterations is 1e-4. For *ICP*, the tolerance of absolute difference in translation and rotation is 0.01 and 0.5 respectively; For *CPD*, the type of transformation is set to *rigid*, and the expected percentage of outliers with respect to a normal distribution is 0.1, the tolerance value is the same of that in DO. For *NDT*, the value of the expected percentage of outliers is set to 0.55, and the tolerance value is set as the same as that in ICP. All deep-learning-based registration networks are trained on an Nvidia Geforce 2080Ti GPU with 12G memory. For *PCRNet*, the kernel sizes are 64, 64, 64, 128, 1024, 1024, 512, 512, 256 and 7. The iteration for rotation and translation is set to 8. Adam optimiser with an initial learning rate of 0,1, 300 epochs and a batch size of 32 is used for the training process. For *PointnetLK*, the kernel sizes are 64, 64, 64, 128, 1024. The maximum iteration for rotation and translation is set to 30. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 10 is used for the training process. For *DCP*, the kernel sizes are 64, 64, 128, 256, 512, 1024, 256, 128, 64, 32 and 7. The iteration for rotation and translation is set to 1. Adam optimiser with an initial learning rate of 0.001, 250 epochs and a batch size of 32 is used for the training process.

6.1.3 The Registration Results

Figure.6.5 shows the registration results of instruments on the mentioned perturbations (occlusion and outliers). The rectangles show the overlap at the tips of the instruments. The overlap parts (rectangles) illustrate that the optimisation-based registration methods (DO, GDO, RDO, and GRDO) have better performance than other registration methods. The deep learning-based methods (DCP, PCR, and PointnetLK) cannot register the real data sets under various perturbations.



(a) ICP (b) NDT (c) CPD (d) DO (e) GDO

(f) RDO (g) GRDO (h) DCP (i) PCR (j) PointnetLK

Fig. 6.5 The registration results of instruments.

Table.6.1 shows the quantitative results of the registration on instruments. MSE represents the mean square error. It can be seen that the registration accuracy of the learned optimisation methods is higher than that of other methods.

Table 6.1 The quantitative results of the registration on instruments. (MSE-Mean Square Error)

	ICP	NDT	CPD	DO	GDO	RDO	GRDO	DCP	PCR	PointNetLK
MSE	0.0097	0.0054	0.0151	0.0037	0.0023	0.0028	0.0045	0.4888	0.0381	0.1056

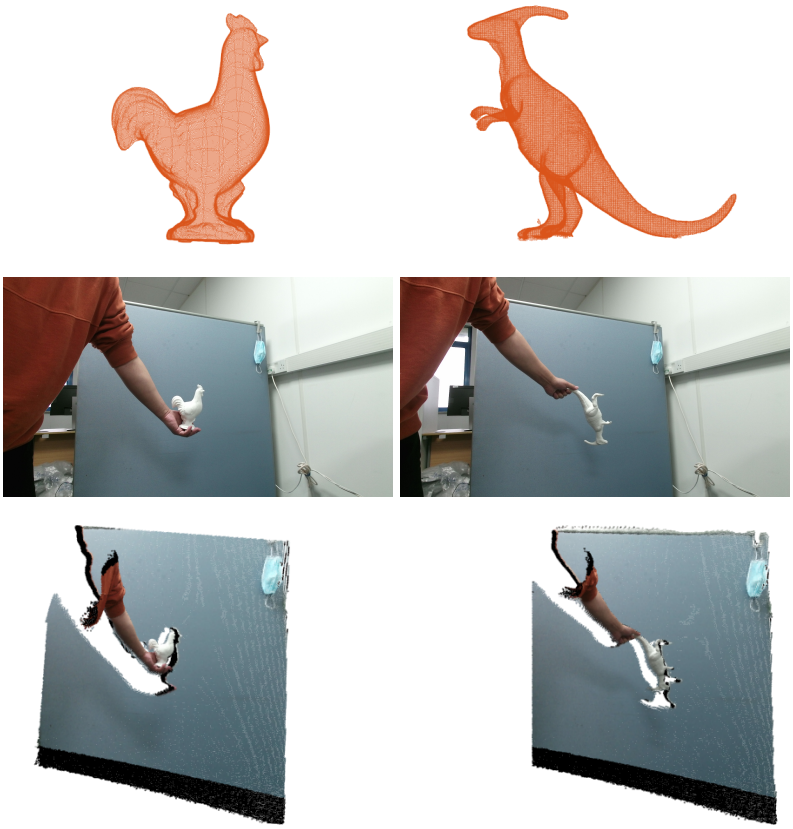
6.2 Registration between Model and Scene

In this section, 3D registration experiments are conducted on the real scenes (A person is holding a chicken or parasaurolophus), which are captured by Microsoft KinectV2, as shown in Figure.6.6.Top shows the 3D model chicken and parasaurolophus. Bottom shows the 3D scene captured by KinectV2.

The parameter settings in this section are similar to the settings in Section6.1.2, the only difference is that the rotation range is extended to 90 degrees for *DO*, *GDO*, *GDO* and *GRDO*.

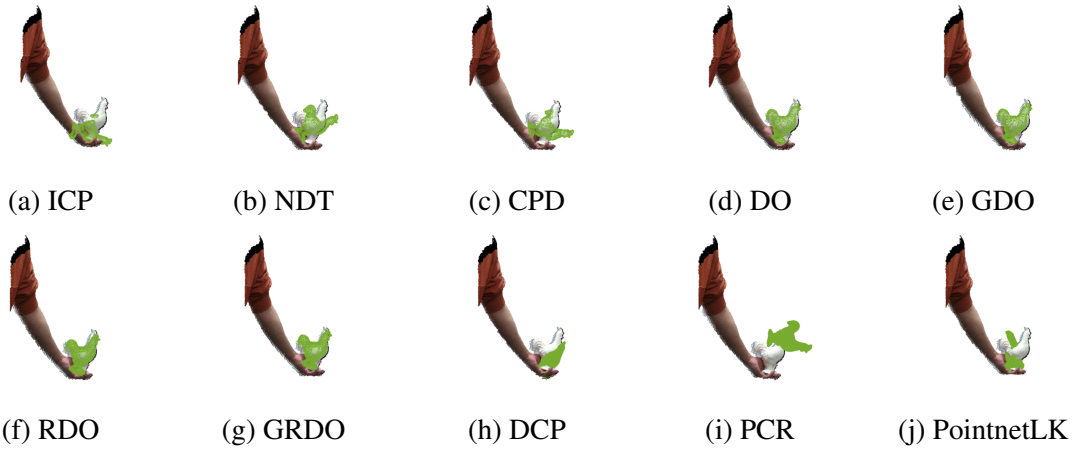
Registration Results

Figure.6.7 and Figure.6.8 show the registration results on the chicken and parasaurolophus models respectively. It can be seen that although the optimisation-based methods (DO, GDO, RDO, and GRDO) can register models and scenes with higher accuracy than deep-learning-based registration methods (DCP, PCR, and PointnetLK) and traditional methods (ICP, CPD, and NDT), they all failed to achieve registration with larger rotation. Besides, it will be a huge challenge for these methods to register models in the case of self-rotation, as shown in Figure.6.8, where the direction of the parasaurolophus' head is opposite.



(a) Chicken (b) Parasaurolophus

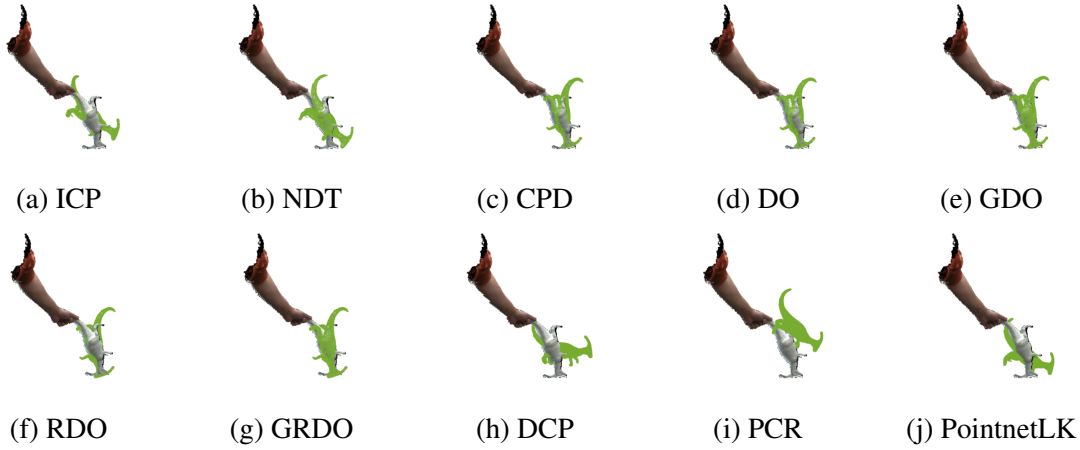
Fig. 6.6 The models and scenes for registration.



(a) ICP (b) NDT (c) CPD (d) DO (e) GDO

(f) RDO (g) GRDO (h) DCP (i) PCR (j) PointnetLK

Fig. 6.7 The registration results of chicken model.



(a) ICP (b) NDT (c) CPD (d) DO (e) GDO

(f) RDO (g) GRDO (h) DCP (i) PCR (j) PointnetLK

Fig. 6.8 The registration results of parasaurolophus model.

6.3 Object Tracking

In this section, 3D registration methods are applied to 3D object tracking, as shown in Figure.6.9. Microsoft KinectV2 is used to capture RGBD videos at 20fps, which is similar to the setting of DO in [11], then a 3D point cloud will be reconstructed from depth images and colour images. In this video, the objects moving is recorded from different orientations. For GDO and GRDO, the approach to extract their features determines that the point clouds to be matched should be in the same space and not far away from each other; otherwise, it needs a high requirement for memory space that may be beyond the scope of the memory for the coding software. Therefore, the first frame is manually initialised to make the point clouds in the same space, and the subsequent frames were initialised via the pose of the previous frames. In addition, the depth image will be downsampled to reduce the computational cost. The parameter settings in this section are similar to the settings in 6.2. Besides, the ratio of incomplete scene shape is set between 0.3 and 0.8 in the training process for *DO*, *GDO*, *GDO* and *GRDO*.

Fig. 6.9 The scenes for object tracking.

Tracking Results

Figure.6.10 and Figure.6.11 show the object tracking results of model bunny. Each column shows the same frame. Each row displays the tracking results of each method. Odd

columns illustrate the reprojection on RGB images, and even columns show the registered 3D point clouds in scene. Please note that the orientations of the bunny model in different columns of Figure.6.10 and Figure.6.11 are different. Compared with other methods, *DO*, *RDO*, *GDO* and *GRDO* can seek the position and the orientation of 3D model in scenes roughly. And if more areas of the models are exposed in the scene, the performance of these methods will be better, which can be found by comparing the 3D point clouds in the second column and the fourth column in Figure.6.11. Nevertheless, *RDO* still has a better performance than other learning-based methods, significantly better than *DO* and *GRDO*, as shown in the black rectangles in Figure.6.10. The reason for the worse performance of *GRDO* is that the exposed small area only provides limited information to *GRDO* to construct the graph and to extract key points. The tracking results of model chicken with different orientations, as shown in Figure.6.12 and Figure.6.13, also expose the above issues. The difference between Figure.6.12 and Figure.6.13 is that the orientations of the chicken model in these two figures are different.

The deep-learning-based registration methods (DCP, PCR, and PointnetLK) and traditional methods (ICP, CPD, and NDT) failed to handle the registration with structured outliers and could not track rotation well. The optimisation-based registration methods (DO, GDO, RDO, GRDO) could robustly track the 3D object under various perturbations but failed to follow the object with more missing areas. Because the approach extracting features for GRDO is based on the graph information, and it lacks sufficient graph information to track objects when dealing with the registration with large missing areas, the performance of GRDO is worse than other optimisation-based methods (DO, GDO, and RDO). The optimisation-based registration methods (DO, GDO, RDO, and GRDO) have better performance than other methods in tracking targets. But the accuracy is not good, which will be the focus of the follow-up research.

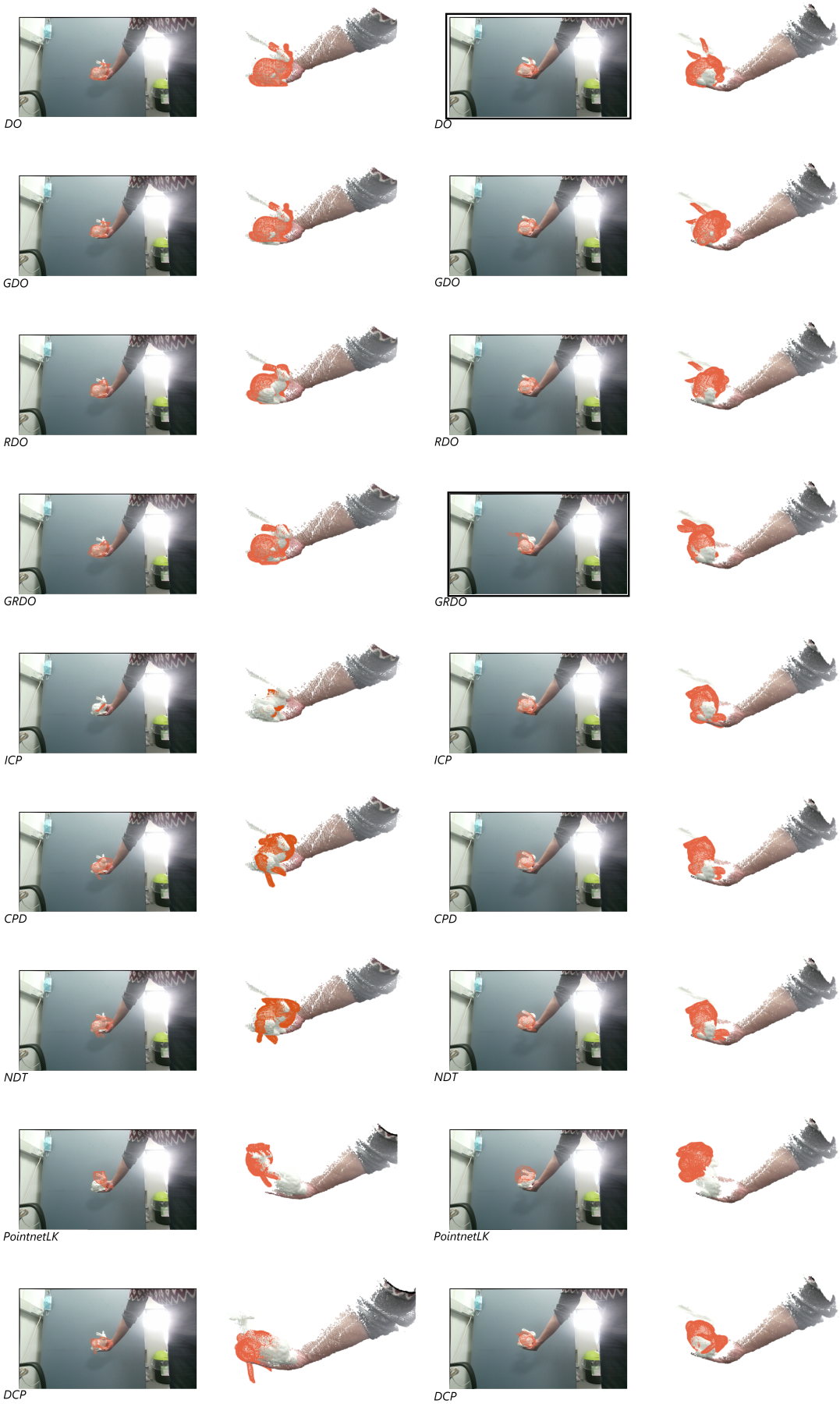


Fig. 6.10 Results for object tracking of model bunny (1).

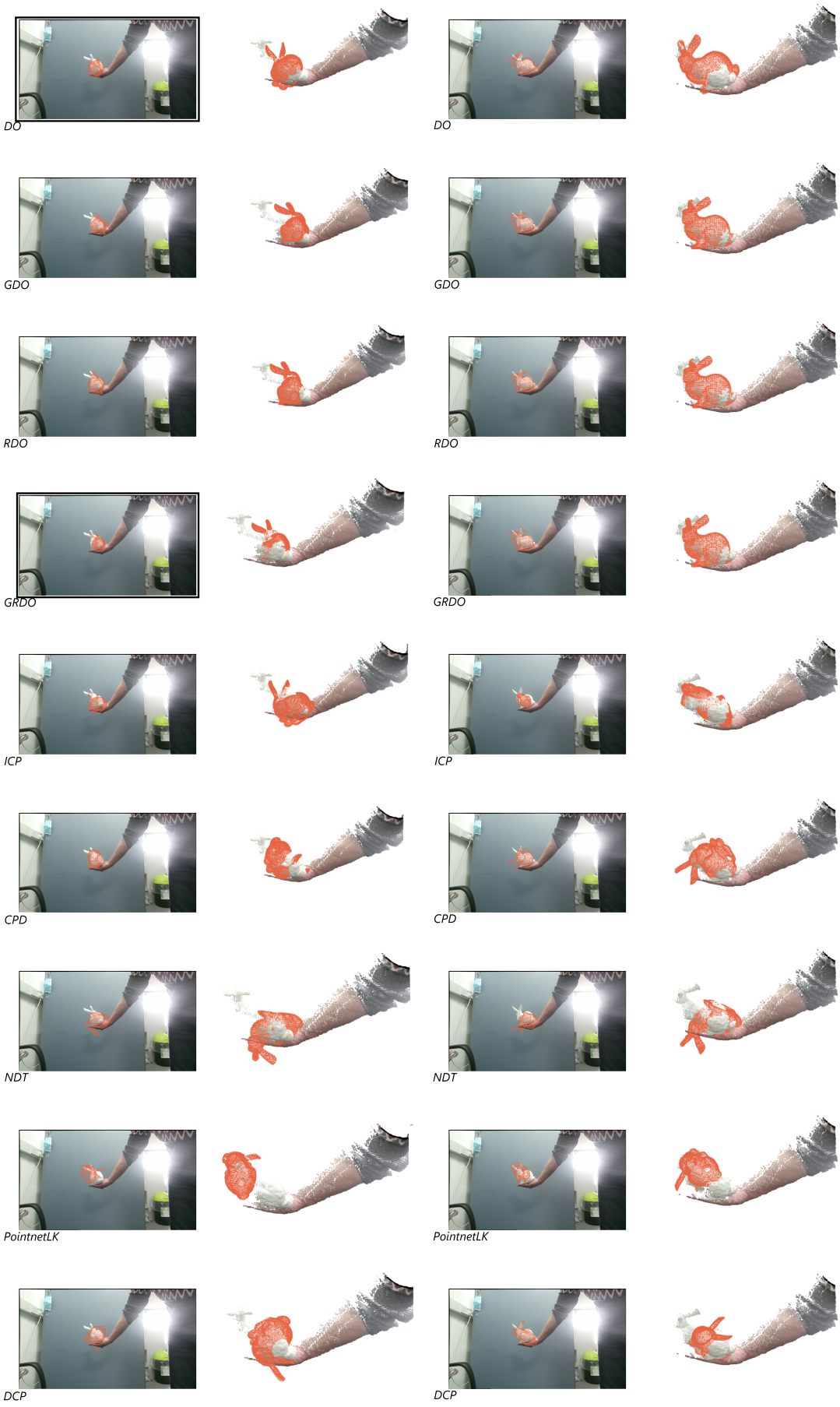


Fig. 6.11 Results for object tracking of model bunny (2).

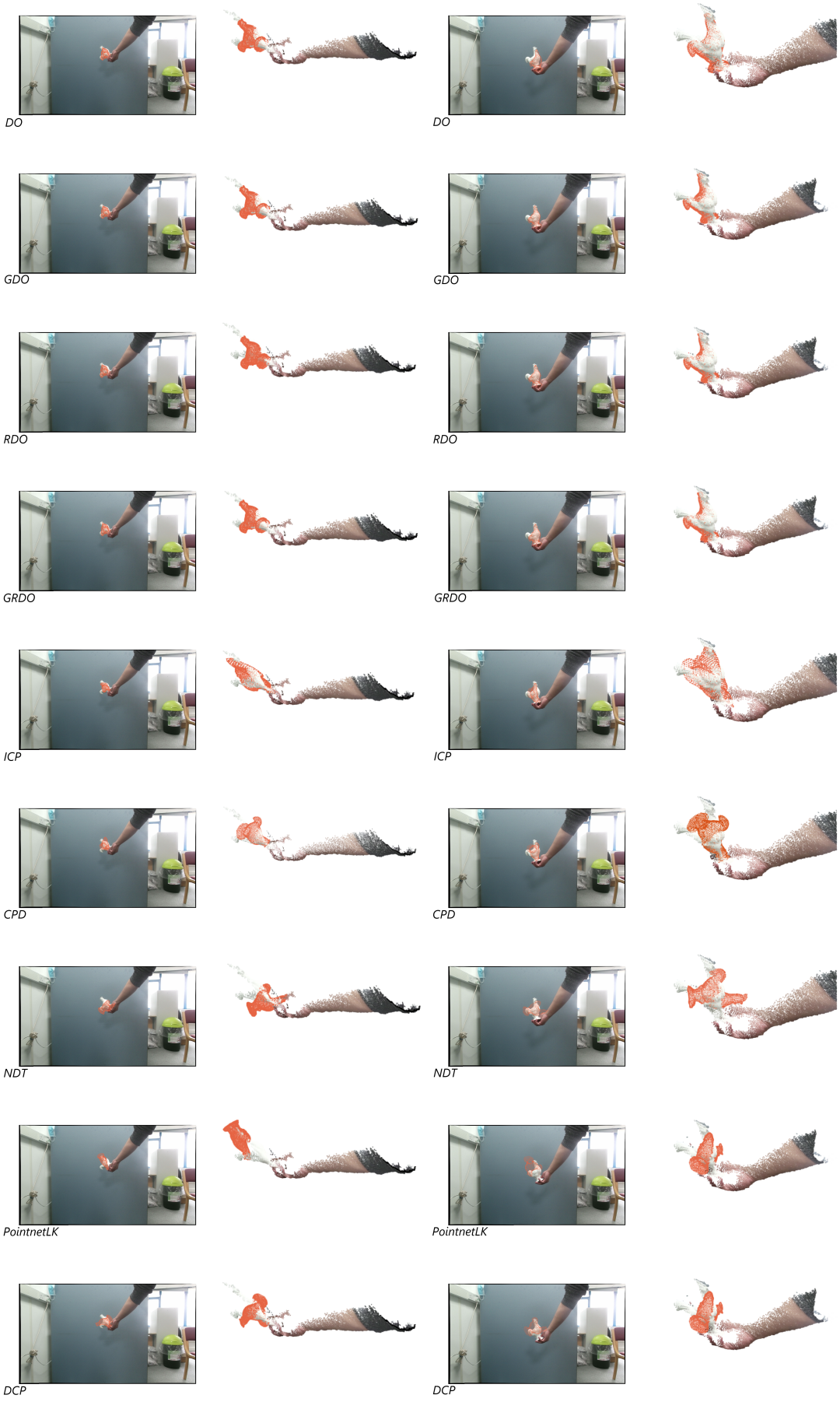


Fig. 6.12 Results for object tracking of model chicken (1).

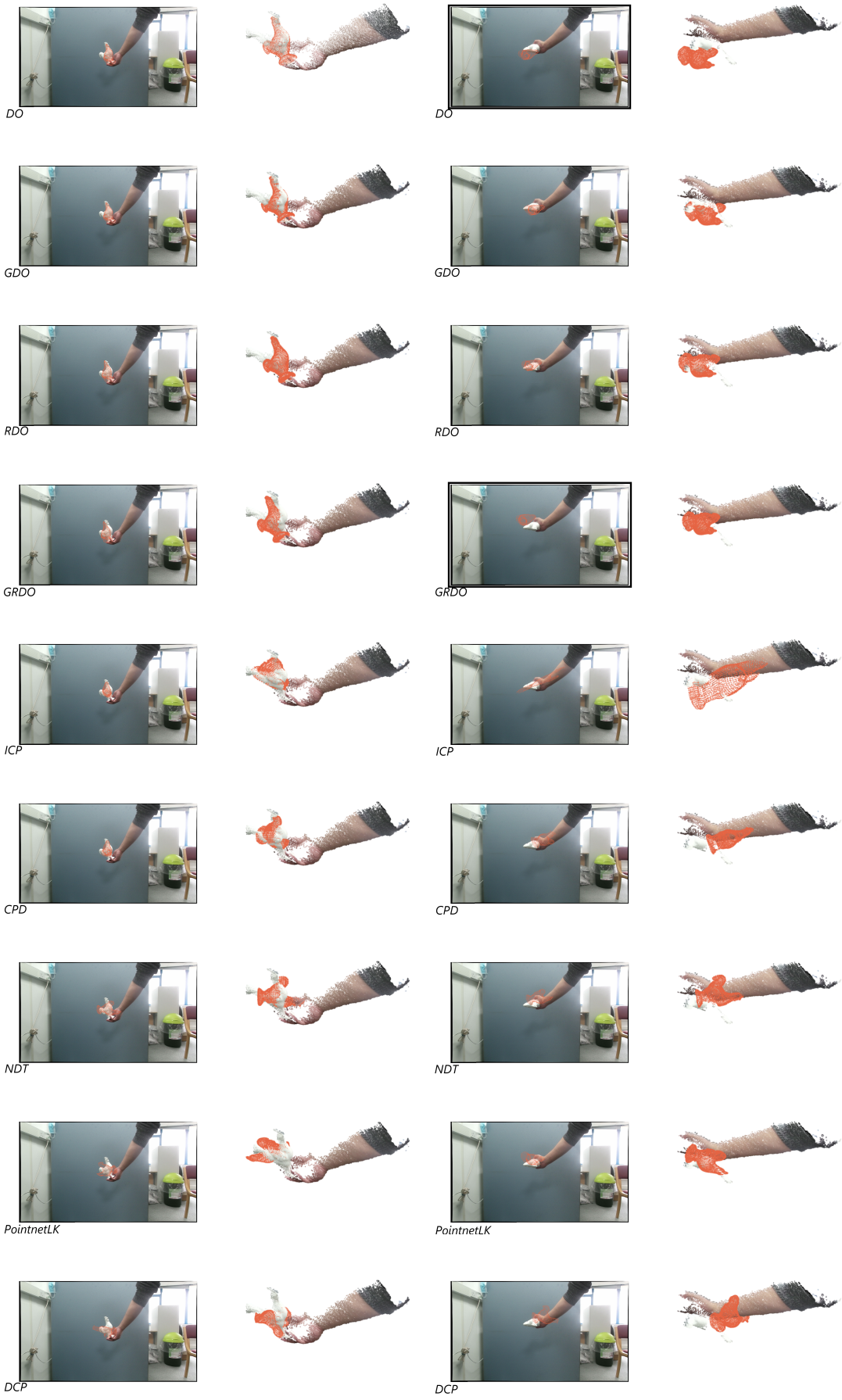


Fig. 6.13 Results for object tracking of model chicken (2).

Chapter 7

Conclusions and Future Works

7.1 Conclusions

Three learning-based optimisation methods and a framework called SGRTmreg are proposed in this thesis to improve the accuracy and robustness of point cloud registration which is the basic problem in augmented reality technology.

Specifically, General Discriminant optimisation (GDO) proposed in Chapter 3 makes use of the *cooperation* of different features to reduce the influence of various perturbations on the learning gradient path. The experimental results show the higher robustness and accuracy of GDO. The registration accuracy of GDO is almost 4.00% higher than that of DO on the Bunny, Chef, and Dancing Children models. For the registration on the complex and real scene, the accuracy of GDO is 25.00% higher than that of DO. The registration accuracy of GDO is 26.00% higher than that of DO on the registration under different noises. In addition, the stability of GDO is 7.91% higher than that of PointNetLK regarding the registration on the ModelNet40 data set. Nevertheless, the computation time of GDO is about triple that of DO.

Reweighted Discriminative optimisation (RDO) proposed in Chapter 4 designs an asymmetrical parameter treatment scheme to assign different weights to components of vectors according to the characteristics of fitting errors over parameter vectors space. The experimental results illustrate the higher accuracy and efficiency of RDO. Specifically, the registration accuracy of RDO is almost 40.00% higher than that of DO on the Happy and Hand models and 2.11% higher than that of DO when achieving the registration on the Dancing Children model. And the computation time of RDO is almost 20% lower than that of DO. The registration success rate of RDO is nearly 30% higher than that of PointNetLK in the registration on the ModelNet40 data set. However, RDO is unable to handle the dense point clouds registration, because the memory requirement for learning $\mathbf{D_{t+1}}$ is $O\left(N\left(((c_1+c_2)N_M+c_2N_S)\right)+c_3N_M^2\right)$.

Graph-based Reweighted Discriminative optimisation (GRDO) proposed in Chapter 5 aims to achieve more accurate and robust registration with less computational cost and less memory requirement. Also, a framework, called SGRTmreg, is designed in Chapter 5 for optimisation-based methods to achieve multiple point clouds registration while maintaining

high accuracy and robustness when dealing with the registration in the case of various perturbations. Experimental results display the higher accuracy and robustness of GRDO. The registration accuracy of GRDO is almost 41.93% higher than that of BCPD on the registration of synthetic data sets under various rotations. The accuracy of GRDO when registering point clouds under various noises is almost twice that of BCPD and is 15.33% higher than that of DO. When handling the registration with large rotations, GRDO has 4.78% higher accuracy than BCPD and 15.06% higher accuracy than PointNetLK. In addition, GRDO has the greatest accuracy and stability than other methods when achieving the registration with multiple perturbations, and the computation time of GRDO is 1/10 of that of DO.

Experimental results in Chapter 6 show that the proposed three methods perform better than traditional and deep learning-based registration methods when applied to real applications. Nevertheless, the experimental results also expose the defects of the above methods, which will be explored in the next section.

7.2 Future Works

Although the proposed approaches have shown better performance than other registration methods (traditional methods and deep learning-based methods), there are still some issues to be solved, which will provide promising research directions for future work.

- *Improvement on the registration with large rotations.*
  The registration results in Chapter 3 show that learning-based optimisation methods are not good at handling the registration with larger rotations (over 90 degrees). In comparison, the ICP method based on the FPFH descriptor shows better performance when registering point clouds with large rotations. Except for the optimisation-based methods, the registration with larger rotation also poses a challenge for deep learning-based methods. Therefore, how to improve the registration accuracy in the case of large rotations will be a research direction. Designing a feature that extracts correspondences while avoiding falling into the local mnimum or preventing overfitting is the breakthrough to achieving the registration with large rotations.

- *Improvement on the registration with longer distance.*
  The features of *GDO* and *GRDO* are all based on the grids dividing the space of 3D point clouds, which is not suitable for the registration with a longer distance. In this thesis, the distance between the two models has to be manually adjusted before registration; otherwise, the memory storage requirement is a challenge for software. Thus, it is necessary to address this issue to improve registration accuracy under longer distances.

- *Registration with self-rotation.*
  The registration results of the parasaurolophus model in Chapter 6 show that learning-based optimisation methods and the other comparison methods are not able to handle

the registration with self-rotation. The process for registration with self-rotation can be cast into two steps: 1) approaching the positions of point clouds; 2) rotating the source point cloud around its centring axis. How to combine these two steps, formulating the process is critical to exploring the potential of registration methods for the achievement of self-rotation registration.

- *Extension to non-rigid Registration.*
  The parameter vector $\mathbf{X}_t$ in these learning-based optimisation methods represents the transformation parameter and is adopted in rigid registration. There is no fixed parameter formulation for non-rigid registration, so extending these learning-based optimisation methods to non-rigid registration is a significant challenge. If applying rigid registration to each point to accomplish the non-rigid registration, finding the substitutes for the ground truth $\mathbf{X}^*$ is still a difficulty. Dealing with non-rigid registration via learning-based optimisation methods could be a potential direction in the future.

- *Improvement on multiple point clouds registration.*
  Although learning-based optimisation methods can achieve multiple point clouds registration via the framework SGRTmreg proposed in Chapter 5, it works on the premise that the point clouds to be matched must be similar, which still limits the flexibility of the learning-based optimisation methods to deal with multiple point clouds registration like deep learning-based methods. One of the reasons for this is that the designed features are not general enough. Therefore, to handle this case, it is vital to develop a general feature to lessen the recline of the sequence of learned regressors $\mathbf{D}_{t+1}$ on features.

References

[1] S. N. Kundu, N. Muhammad, and F. Sattar, "Using the augmented reality sandbox for advanced learning in geoscience education," in *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 2017, pp. 13–17.

[2] M. Y. Ni, R. W. Hui, T. K. Li, A. H. Tam, L. L. Choy, K. K. Ma, F. Cheung, and G. M. Leung, "Augmented reality games as a new class of physical activity interventions? the impact of pokémon go use and gaming intensity on physical activity," *Games for health journal*, vol. 8, no. 1, pp. 1–6, 2019.

[3] L. Chen, W. Tang, N. W. John, T. R. Wan, and J. J. Zhang, "Slam-based dense surface reconstruction in monocular minimally invasive surgery and its application to augmented reality," *Computer methods and programs in biomedicine*, vol. 158, pp. 135–146, 2018.

[4] M. Lowney and A. S. Raj, "Model based tracking for augmented reality on mobile devices," *Computer Science*, pp. 1–5, 2016.

[5] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[6] W. Narzt, G. Pomberger, A. Ferscha, D. Kolb, R. Müller, J. Wieghardt, H. Hörtner, and C. Lindinger, "Augmented reality navigation systems," *Universal Access in the Information Society*, vol. 4, no. 3, pp. 177–187, 2006.

[7] A. Fischer, "A special newton-type optimization method," *Optimization*, vol. 24, no. 3-4, pp. 269–284, 1992.

[8] H. Qinghui, W. Shiwei, L. Zhiyuan, and L. Xiaogang, "Quasi-newton method for lp multiple kernel learning," *Neurocomputing*, vol. 194, pp. 218–226, 2016.

[9] X. Xiong and F. De la Torre, "Global supervised descent method," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2664–2673.

[10] ——, "Supervised descent method and its applications to face alignment," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 532–539.

[11] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira, "Discriminative optimization: Theory and applications to point cloud registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4104–4112.

[12] ——, "Discriminative optimization: Theory and applications to point cloud registration," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] M. Levoy, J. Gerth, B. Curless, and K. Pull, "The stanford 3d scanning repository," *URL http://www-graphics. stanford. edu/data/3dscanrep*, vol. 5, 2005.

[14] A. S. Mian, M. Bennamoun, and R. A. Owens, "A novel representation and feature matching algorithm for automatic pairwise registration of range images," *International Journal of Computer Vision*, vol. 66, no. 1, pp. 19–40, 2006.

[15] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3050–3057.

[16] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[17] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7163–7172.

[18] G. Turk and B. Mullins, "Large geometric models archive, georgia institute of technology," 1998.

[19] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira, "Discriminative optimization: Theory and applications to computer vision," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 4, pp. 829–843, 2019.

[20] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *European conference on computer vision*. Springer, 2004, pp. 224–237.

[21] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid icp algorithms for surface registration," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[22] S. Ge and G. Fan, "Non-rigid articulated point set registration for human pose estimation," in *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2015, pp. 94–101.

[23] B. Eckart, K. Kim, and J. Kautz, "Fast and accurate point cloud registration using trees of gaussian mixtures," *arXiv preprint arXiv:1807.02587*, 2018.

[24] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*. Springer, 2016, pp. 766–782.

[25] D. Campbell and L. v. Petersson, "Gogma: Globally-optimal gaussian mixture alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] C. Papazov and D. Burschka, "Stochastic global optimization for robust point set registration," *Computer Vision and Image Understanding*, vol. 115, no. 12, pp. 1598–1609, 2011.

[27] B. Jian and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1633–1645, 2010.

[28] O. Hirose, "Dependent landmark drift: robust point set registration with a gaussian mixture model and a statistical shape model," *arXiv preprint arXiv:1711.06588*, 2017.

[29] B. Eckart, K. Kim, and J. Kautz, "Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 705–721.

[30] H.-B. Qu, J.-Q. Wang, B. Li, and M. Yu, "Probabilistic model for robust affine and non-rigid point set matching," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 2, pp. 371–384, 2016.

[31] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[32] Y. Khoo and A. Kapoor, "Non-iterative rigid 2d/3d point-set registration using semidefinite programming," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 2956–2970, 2016.

[33] P. Bergström and O. Edlund, "Robust registration of point sets using iteratively reweighted least squares," *Computational Optimization and Applications*, vol. 58, no. 3, pp. 543–561, 2014.

[34] J. Sanchez, F. Denis, P. Checchin, F. Dupont, and L. Trassoudaine, "Global registration of 3d lidar point clouds based on scene features: Application to structured environments," *Remote Sensing*, vol. 9, no. 10, p. 1014, 2017.

[35] I. Kolesov, J. Lee, G. Sharp, P. Vela, and A. Tannenbaum, "A stochastic approach to diffeomorphic point set registration with landmark constraints," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 238–251, 2015.

[36] W. Lian and L. Zhang, "Robust point matching revisited: a concave optimization approach," in *European conference on computer vision*. Springer, 2012, pp. 259–272.

[37] S. M. Ahmed, N. R. Das, and K. N. Chaudhury, "Least-squares registration of point sets over se (d) using closed-form projections," *Computer Vision and Image Understanding*, vol. 183, pp. 20–32, 2019.

[38] L. Li, M. Yang, C. Wang, and B. Wang, "Cubature split covariance intersection filter-based point set registration," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3942–3953, 2018.

[39] W. Lian, L. Zhang, Y. Liang, and Q. Pan, "A quadratic programming based cluster correspondence projection algorithm for fast point matching," *Computer Vision and Image Understanding*, vol. 114, no. 3, pp. 322–333, 2010.

[40] Y. Deng, A. Rangarajan, S. Eisenschenk, and B. C. Vemuri, "A riemannian framework for matching point clouds represented by the schrodinger distance transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3756–3761.

[41] J. Ma, J. Zhao, J. Tian, Z. Tu, and A. L. Yuille, "Robust estimation of nonrigid transformation for point set registration," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2147–2154.

[42] X. Li, J. Yang, Q. Wang *et al.*, "Nonrigid points alignment with soft-weighted selection." in *IJCAI*, 2018, pp. 800–806.

[43] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti, "Point clouds registration with probabilistic data association," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4092–4098.

[44] L. Wang, J. Chen, X. Li, and Y. Fang, "Non-rigid point set registration networks," *arXiv preprint arXiv:1904.01428*, 2019.

[45] G. D. Pais, S. Ramalingam, V. M. Govindu, J. C. Nascimento, R. Chellappa, and P. Miraldo, "3dregnet: A deep neural network for 3d point registration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7193–7203.

[46] H. Deng, T. Birdal, and S. Ilic, "3d local features for direct pairwise registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3244–3253.

[47] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6359–6367.

[48] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3d point clouds using geo-cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 998–1008.

[49] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer, "Learning local shape descriptors from part correspondences with multiview convolutional networks," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 1, pp. 1–14, 2017.

[50] Z. J. Yew and G. H. Lee, "Rpm-net: Robust point matching using learned features," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 824–11 833.

[51] Y. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3522–3531.

[52] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "Deepicp: An end-to-end deep neural network for 3d point cloud registration," *arXiv preprint arXiv:1905.04153*, 2019.

[53] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust amp; efficient point cloud registration using pointnet," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7156–7165.

[54] L. Zhou, S. Zhu, Z. Luo, T. Shen, R. Zhang, M. Zhen, T. Fang, and L. Quan, "Learning and matching multi-view descriptors for registration of point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 505–522.

[55] M. Simon, K. Fischer, S. Milz, C. T. Witt, and H.-M. Gross, "Stickypillars: Robust feature matching on point clouds using graph neural networks," *arXiv preprint arXiv:2002.03983*, 2020.

[56] L. Zhu, J. Barhak, V. Srivatsan, and R. Katz, "Efficient registration for precision inspection of free-form surfaces," *The International Journal of Advanced Manufacturing Technology*, vol. 32, no. 5-6, pp. 505–515, 2007.

[57] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4, 2009, p. 435.

[58] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking." in *ISMAR*, vol. 11, no. 2011, 2011, pp. 127–136.

[59] F. Pfeuffer, M. Stiglmayr, and K. Klamroth, "Discrete and geometric branch and bound algorithms for medical image registration," *Annals of Operations Research*, vol. 196, no. 1, pp. 737–765, 2012.

[60] F. Wang, B. C. Vemuri, A. Rangarajan, and S. J. Eisenschenk, "Simultaneous nonrigid registration of multiple point sets and atlas construction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 2011–2022, 2008.

[61] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2743–2748.

[62] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3212–3217.

[63] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness, "New algorithms for 2d and 3d point matching: Pose estimation and correspondence," *Pattern recognition*, vol. 31, no. 8, pp. 1019–1031, 1998.

[64] H. Yang and L. Carlone, "A polynomial-time solution for robust registration with extreme outlier rates," *arXiv preprint arXiv:1903.08588*, 2019.

[65] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis, "A generative model for the joint registration of multiple point sets," in *European Conference on Computer Vision*. Springer, 2014, pp. 109–122.

[66] B. Eckart, K. Kim, A. Troccoli, A. Kelly, and J. Kautz, "Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration," in *2015 International Conference on 3D Vision*. IEEE, 2015, pp. 241–249.

[67] J. Ma, J. Zhao, J. Jiang, and H. Zhou, "Non-rigid point set registration with robust transformation estimation under manifold regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[68] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[69] V. Panaganti and R. Aravind, "Robust nonrigid point set registration using graph-laplacian regularization," in *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2015, pp. 1137–1144.

[70] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers, "A super-resolution framework for high-accuracy multiview reconstruction," *International journal of computer vision*, vol. 106, no. 2, pp. 172–191, 2014.

[71] G. Tzimiropoulos and M. Pantic, "Gauss-newton deformable part models for face alignment in-the-wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1851–1858.

[72] M. Aigner and B. Juttler, "Gauss-newton-type techniques for robustly fitting implicitly defined curves and surfaces to unorganized data points," in *2008 IEEE International Conference on Shape Modeling and Applications*. IEEE, 2008, pp. 121–130.

[73] A. M. Buchanan and A. W. Fitzgibbon, "Damped newton algorithms for matrix factorization with missing data," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 316–322.

[74] R. R. Paulsen, J. A. Baerentzen, and R. Larsen, "Markov random field surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 4, pp. 636–646, 2009.

[75] W. Xu, Z. Miao, X.-P. Zhang, and Y. Tian, "A hierarchical spatio-temporal model for human activity recognition," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1494–1509, 2017.

[76] H. Hadizadeh and I. V. Bajić, "Soft video multicasting using adaptive compressed sensing," *IEEE Transactions on Multimedia*, vol. 23, pp. 12–25, 2020.

[77] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[78] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady AN USSR*, vol. 269, 1983, pp. 543–547.

[79] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[80] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[82] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[83] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[84] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4148–4158.

[85] S. Avidan, "Support vector tracking," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[86] M. H. Nguyen and F. De la Torre, "Metric learning for image alignment," *International Journal of Computer Vision*, vol. 88, no. 1, pp. 69–84, 2010.

[87] L. Liang, R. Xiao, F. Wen, and J. Sun, "Face alignment via component-based discriminative search," in *European conference on computer vision*. Springer, 2008, pp. 72–85.

[88] T. F. Cootes, M. C. Ionita, C. Lindner, and P. Sauer, "Robust and accurate shape model fitting using random forest regression voting," in *European Conference on Computer Vision*. Springer, 2012, pp. 278–291.

[89] K. Paliouras and A. A. Argyros, "Towards the automatic definition of the objective function for model-based 3d hand tracking," in *ICMMI*, 2015.

[90] D. Campbell and L. Petersson, "An adaptive data representation for robust point-set registration and merging," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4292–4300.

[91] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Advances in neural information processing systems*, 2000, pp. 512–518.

[92] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[93] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 1078–1085.

[94] G. Tzimiropoulos, "Project-out cascaded regression with an application to face alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3659–3667.

[95] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[96] O. Tuzel, F. M. Porikli, P. Meer *et al.*, "Learning on lie groups for invariant detection and tracking." in *CVPR*. Citeseer, 2008, pp. 8–9.

[97] J. Saragih and R. Goecke, "Iterative error bound minimisation for aam alignment," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2. IEEE, 2006, pp. 1196–1195.

[98] D. Cristinacce and T. F. Cootes, "Boosted regression active shape models." in *BMVC*, vol. 2. Citeseer, 2007, pp. 880–889.

[99] O. Hirose, "A bayesian formulation of coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[100] S. Du, G. Xu, S. Zhang, X. Zhang, Y. Gao, and B. Chen, "Robust rigid registration algorithm based on pointwise correspondence and correntropy," *Pattern Recognition Letters*, vol. 132, pp. 91–98, 2020.

[101] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset, "Pcrnet: Point cloud registration network using pointnet encoding," *arXiv preprint arXiv:1908.07906*, 2019.

[102] Y. Wang and J. M. Solomon, "ns for point cloud registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3523–3532.

[103] C. S. Burrus, "Iterative reweighted least squares," *OpenStax CNX. Available online: http://cnx. org/contents/92b90377-2b34-49e4-b26f-7fe572db78a1*, vol. 12, 2012.

[104] E. Eade, "Lie groups for 2d and 3d transformations," *URL http://ethaneade. com/lie. pdf, revised Dec*, 2013.

[105] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1584–1601, 2006.

[106] D. Görür and C. E. Rasmussen, "Dirichlet process gaussian mixture models: Choice of the base distribution," *Journal of Computer Science and Technology*, vol. 25, no. 4, pp. 653–664, 2010.

[107] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*. McGraw-Hill New York, 1986, vol. 31999.

[108] M. E. Newman, "The mathematics of networks," *The new palgrave encyclopedia of economics*, vol. 2, no. 2008, pp. 1–12, 2008.

[109] Y. Zhao, W. Tang, J. Feng, T. Wan, and L. Xi, "Reweighted discriminative optimization for least-squares problems with point cloud registration," *Neurocomputing*, 2021.

[110] L. Xi, Y. Zhao, L. Chen, Q. H. Gao, W. Tang, T. R. Wan, and T. Xue, "Recovering dense 3d point clouds from single endoscopic image," *Computer Methods and Programs in Biomedicine*, vol. 205, p. 106077, 2021.

[111] P. Mountney, D. Stoyanov, and G.-Z. Yang, "Three-dimensional tissue deformation recovery and tracking," *IEEE Signal Processing Magazine*, vol. 27, no. 4, pp. 14–24, 2010.

[112] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[113] M. Awrangjeb, "Using point cloud data to identify, trace, and regularize the outlines of buildings," *International Journal of Remote Sensing*, vol. 37, no. 3, pp. 551–579, 2016.

Appendix A

Convergence proofs

The convergence proofs of GDO and RDO are provided here. The learning process of updating maps in GRDO is the same as that of RDO, so there is no convergence proof for GRDO.

A.1 Convergence Proof for GDO

Theorem A.1.1 (Convergence of GDO's training error)
Given a training set $\left\{\left(\mathbf{x}_0^i, \mathbf{x}_*^i, \mathbf{H}_{f0}^i\right)\right\}_{i=1}^N$, if there exists a linear map $\hat{D} \in \mathbf{R}^{p\times f}$ where $\hat{D}\mathbf{H}_f$ meets the condition $\sum_{i=1}^N \left(\mathbf{x}_*^i - \mathbf{x}_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_{ft}^i > 0$ at $\mathbf{x}_*^i$ for all i, and if there exists an i where $\mathbf{x}_t^i \neq \mathbf{x}_*^i$, then the update rule:

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i - D_t \mathbf{H}_{ft}^i. \tag{A.1}$$

$$D_{t+1} = \min_{\tilde{D}} \frac{1}{N} \sum_{i=1}^N \left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \tilde{D}\mathbf{H}_{ft}^i\right\|_2^2 + \frac{\lambda}{2}\left\|\tilde{D}\right\|_F^2. \tag{A.2}$$

with $D_t \subset \mathbf{R}^{p\times f}$ obtained from (A.2), guarantees that the training error strictly decreases in each iteration:

$$\sum_{i=1}^N \left\|\mathbf{x}_*^i - \mathbf{x}_{t+1}^i\right\|_2^2 < \sum_{i=1}^N \left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2. \tag{A.3}$$

If $\hat{D}\mathbf{H}_f$ is strongly monotone, and if there exist $H > 0$, $M > 0$ such that $\left\|\hat{D}\mathbf{H}_{ft}^i\right\|_2^2 \leq H + M\left\|\mathbf{x}_*^i - \mathbf{x}^i\right\|_2^2$ for all i, then the training error converges to zero.

Proof.
First, the proof in this case that $\hat{D}\mathbf{H}_f$ meets the condition $\sum_{i=1}^N \left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_{ft}^i > 0$ is illustarted.

According to Equation.(A.2), the following equation can be attained:

$$\sum_{i=1}^N \left\|x_*^i - x_t^i + \mathbf{D}_{t+1}\mathbf{H}_{ft}^i\right\|_2^2 \leq \sum_{i=1}^N \left\|x_*^i - x_t^i + \bar{\mathbf{D}}\mathbf{H}_{ft}^i\right\|_2^2 \tag{A.4}$$

Let $\bar{\mathbf{D}} = \alpha\hat{\mathbf{D}}$

$$\alpha = \frac{\beta}{\eta} \tag{A.5}$$

$$\beta = \sum_{i=1}^{N} \left(x_t^i - x_*^i\right)^T \hat{\mathbf{D}}\mathbf{H}_{ft}^i) \tag{A.6}$$

$$\eta = \sum_{i=1}^{N} \left\|\hat{\mathbf{D}}\mathbf{H}_{ft}^i\right\|_2^2 \tag{A.7}$$

Since $\sum_{i=1}^{N} \left(x_*^i - x_t^i\right)^{\mathrm{T}} \hat{D}\mathbf{H}_{ft}^i > 0$, both β and η are both positive, and thus α is also positive.

$$\begin{aligned}
\sum_{i=1}^{N} \left\|x_{t+1}^i - x_*^i\right\|_2^2 &= \sum_{i=1}^{N} \left\|x_*^i - x_t^i + \mathbf{D}_{t+1}\mathbf{H}_{ft}^i\right\|_2^2 \\
&\leq \sum_{i=1}^{N} \left\|x_*^i - x_t^i + \bar{\mathbf{D}}\mathbf{H}_{ft}^i\right\|_2^2 \\
&= \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|_2^2 + \sum_{i=1}^{N} \left\|\alpha\hat{\mathbf{D}}\mathbf{H}_{ft}^i\right\|_2^2 + \\
&+ 2\alpha \sum_{i=1}^{N} \left(x_t^i - x_*^i\right)^T \hat{\mathbf{D}}\mathbf{H}_{ft}^i \\
&= \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|_2^2 + \alpha^2\eta - 2\alpha\beta \\
&= \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|_2^2 + \frac{\beta^2}{\eta} - 2\frac{\beta^2}{\eta} \\
&= \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|_2^2 - \frac{\beta^2}{\eta} \\
&< \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|.
\end{aligned} \tag{A.8}$$

Then :

$$\sum_{i=1}^{N} \left\|x_*^i - x_{t+1}^i\right\|_2^2 < \sum_{i=1}^{N} \left\|x_*^i - x_t^i\right\|_2^2.$$

Based on the definition of strongly monotone at a point and the assumption in this theorem, the following equation is attained

$$\beta = \sum_{i=1}^{N} \left(x_t^i - x_*^i\right)^T \hat{\mathbf{D}}\mathbf{H}_{ft}^i \geq m \sum_{i=1}^{N} \left\|x_t^i - x_*^i\right\|_2^2 \tag{A.9}$$

$$\eta = \sum_{i=1}^{N} \left\| \hat{\mathbf{D}} \mathbf{H}_f^i \right\|_2^2 \leq NH + M \sum_{i=1}^{N} \left\| x_*^i - x_t^i \right\|_2^2 \tag{A.10}$$

Also, let us denote the training error in the t^{th} iteration as E_t:

$$E_t = \sum_{i=1}^{N} \left\| x_*^i - x_t^i \right\|_2^2 \tag{A.11}$$

Form Equation.(A.9),

$$\begin{aligned} E_{t+1} &= E_t - \frac{\beta^2}{\eta} \\ &\leq E_t - \frac{m^2 E_t^2}{NH + ME_t} \\ &= \left(1 - \frac{m^2 E_t}{NH + ME_t}\right) E_t \end{aligned} \tag{A.12}$$

Recursively applying the above inequality, it can be found that

$$E_{t+1} \leq E_0 \prod_{l=0}^{t} \left(1 - \frac{m^2 E_l}{NH + ME_l}\right) \tag{A.13}$$

Since $E_t \geq 0$ for all t, and

$$0 < \left(1 - \frac{m^2 E_l}{NH + ME_l}\right) < 1 \tag{A.14}$$

thus

$$\lim_{t \to +\infty} E_{t+1} = 0 \tag{A.15}$$

Next, the case where $H = 0$ is considered. In this case,

$$0 \leq E_{t+1} \leq E_t \left(1 - \frac{m^2}{M}\right) = E_0 \left(1 - \frac{m^2}{M}\right)^{t+1} \tag{A.16}$$

This proves that the training error converges linearly to zero.

A.2 Convergence Proof for RDO

Theorem A.2.1 (Convergence of RDO's training error)
Given a training set $\left\{ \left(x_0^i, x_*^i, \mathbf{h}_0^i\right) \right\}_{i=1}^{N}$, if there exists a linear map $\hat{D} \in \mathbf{R}^{p \times f}$ where $\hat{D}\mathbf{h}_t^i$ meets the condition $\sum_{i=1}^{N} \left(x_t^i - x_*^i\right)^{\mathrm{T}} \hat{D}\mathbf{h}_t^i > 0$ at x_*^i for all i where $\left[\mathbf{h}_t^i\right]_{j,:} \in (0,1)$, and if there exists an i where $\mathbf{x}_t^i \neq \mathbf{x}_*^i$, then the update rule:

$$\mathbf{D}_{t+1} = \min_{\hat{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{W}_t \left(\mathbf{x}_*^i - \mathbf{x}_t^i + \hat{\mathbf{D}}\mathbf{h}_t^i\right) \right\|_2^2 + \lambda \left\| \hat{\mathbf{D}} \right\|_F^2 \tag{A.17}$$

where $[\mathbf{W}_t]_{j,j} > 1$

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i - \mathbf{D}_{t+1}\mathbf{h}_t^i \tag{A.18}$$

guarantees that the training error strictly decreases in each iteration:

$$\sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_{t+1}^i\right\|_2^2 < \sum_{i=1}^{N} \left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2 \tag{A.19}$$

If $\hat{\mathbf{D}}\mathbf{h}_t^i$ is strongly monotone at $\mathbf{x}_*^i$, and if there exist $H > 0$, $M > 0$ such that $\left\|\hat{\mathbf{D}}\mathbf{h}_t^i\right\|_2^2 \leq H + M\left\|\mathbf{x}_*^i - \mathbf{x}_t^i\right\|_2^2$ for all i, then the training error converges to zero.

Proof.
First, the proof in this case that $\hat{D}\mathbf{h}_t^i$ meets the condition $\sum_{i=1}^{N}\left(x_t^i - x_*^i\right)^{\mathrm{T}}\hat{D}\mathbf{h}_t^i > 0$ is illustrated. Assume that not all $\mathbf{x}_*^i = \mathbf{x}_t^i$, otherwise all $\mathbf{x}_*^i$ are already at their stationary points. Let us denote

$$f(\mathbf{D}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{W}_t\left(\mathbf{x}_*^i - \mathbf{x}_t^i + \mathbf{D}\mathbf{h}_t^i\right)\right\|_2^2 + \lambda\left\|\mathbf{D}\right\|_F^2 \tag{A.20}$$

$$g(\mathbf{D}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \mathbf{D}\mathbf{h}_t^i\right\|_2^2 + \lambda\left\|\mathbf{W}_t^{-1}\mathbf{D}\right\|_F^2 \tag{A.21}$$

$$q(\mathbf{D}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \mathbf{D}\mathbf{h}_t^i\right\|_2^2 + \lambda\left\|\mathbf{D}\right\|_F^2 \tag{A.22}$$

Derive $f(\mathbf{D})$, $g(\mathbf{D})$ and $q(\mathbf{D})$ with respect to $\mathbf{D}$, the following equation can be attained

$$\mathbf{D}_{t+1} = \min_{\mathbf{D}} f(\mathbf{D}) = \min_{\mathbf{D}} g(\mathbf{D}) \tag{A.23}$$

$$\bar{\mathbf{D}}_{t+1} = \min_{\mathbf{D}} q(\mathbf{D}) \tag{A.24}$$

Where $\bar{\mathbf{D}}_{t+1} \neq \mathbf{D}_{t+1}$. From Equation A.22, A.23 and A.24, it can be seen that

$$\begin{aligned}
&\lambda N\left\|\bar{\mathbf{D}}_{\mathbf{t+1}}\right\|_F^2 - \lambda N\left\|\mathbf{D}_{t+1}\right\|_F^2 \\
&\leq \sum_{i=1}^{N}\left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \mathbf{D}_{\mathbf{t+1}}\mathbf{h}_t^i\right\|_2^2 - \sum_{i=1}^{N}\left\|\mathbf{x}_*^i - \mathbf{x}_t^i + \bar{\mathbf{D}}_{\mathbf{t+1}}\mathbf{h}_t^i\right\|_2^2 \\
&\leq \lambda N\left\|\mathbf{W}_t{}^{-1}\bar{\mathbf{D}}_{\mathbf{t+1}}\right\|_F^2 - \lambda N\left\|\mathbf{W}_t{}^{-1}\mathbf{D}_{t+1}\right\|_F^2
\end{aligned} \tag{A.25}$$

Solve Equation A.23 and Equation A.24,

$$\mathbf{D}_{t+1}\sum_{i=1}^{N}\mathbf{h}_t^i\mathbf{h}_t^{iT} + \left(\mathbf{W}_t\mathbf{W}_t\right)^{-1}\lambda N\mathbf{D}_{t+1} = \sum_{i=1}^{N}\left(\mathbf{x}_t^i - \mathbf{x}_*^i\right)\mathbf{h}_t^{iT} \tag{A.26}$$

$$\bar{\mathbf{D}}_{t+1}\sum_{i=1}^{N}\mathbf{h}_t^i\mathbf{h}_t^{iT}+\lambda N\bar{\mathbf{D}}_{t+1}=\sum_{i=1}^{N}\left(\mathbf{x}_t^i-\mathbf{x}_*^i\right)\mathbf{h}_t^{iT} \tag{A.27}$$

Due to $\left[\mathbf{h}_t^i\right]_{j,:}\in(0,1)$, $\mathbf{h}_t^i$ and $[\mathbf{W}_t]_{j,j}>1$, where $j\in\{1,2,\cdots p\}$. Thus

$$\|\mathbf{D}_{t+1}\|_F^2\geq\left\|\bar{\mathbf{D}}_{t+1}\right\|_F^2 \tag{A.28}$$

Similarly,

$$\lambda N\left\|\mathbf{W}_t{}^{-1}\mathbf{D_{t+1}}\right\|_F^2\geq\lambda N\left\|\mathbf{W}_t{}^{-1}\bar{\mathbf{D}}_{t+1}\right\|_F^2 \tag{A.29}$$

Then

$$\sum_{i=1}^{N}\left\|\mathbf{x}_*^i-\mathbf{x}_t^i+\mathbf{D_{t+1}}\mathbf{h}_t^i\right\|_2^2\leq\sum_{i=1}^{N}\left\|\mathbf{x}_*^i-\mathbf{x}_t^i+\bar{\mathbf{D}}_{\mathbf{t+1}}\mathbf{h}_t^i\right\|_2^2 \tag{A.30}$$

Let $\hat{\mathbf{D}}=\frac{1}{\alpha}\bar{\mathbf{D}}_{t+1}$

$$\alpha=\frac{\eta}{\gamma} \tag{A.31}$$

$$\eta=\sum_{i=1}^{N}\left(\mathbf{x}_t^i-\mathbf{x}_*^i\right)^T\hat{\mathbf{D}}\mathbf{h}_t^i \tag{A.32}$$

$$\gamma=\sum_{i=1}^{N}\left\|\hat{\mathbf{D}}\mathbf{h}_t^i\right\|_2^2 \tag{A.33}$$

Since $\sum_{i=1}^{N}\left(x_t^i-x_*^i\right)^{\mathrm{T}}\hat{D}\mathbf{h}_t^i>0$, both η and γ are both positive, and thus α is also positive.

$$\begin{aligned}
\sum_{i=1}^{N}\left\|\mathbf{x}_{t+1}^{i}-\mathbf{x}_{*}^{i}\right\|_{2}^{2} &= \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}+\mathbf{D}_{t+1}\mathbf{h}_{t}^{i}\right\|_{2}^{2} \\
&\leq \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}+\bar{\mathbf{D}}_{t+1}\mathbf{h}_{t}^{i}\right\|_{2}^{2} \\
&= \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2}+\sum_{i=1}^{N}\left\|\alpha\hat{\mathbf{D}}\mathbf{h}_{t}^{i}\right\|_{2}^{2}+ \\
&+2\alpha\sum_{i=1}^{N}\left(\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right)^{T}\hat{\mathbf{D}}\mathbf{h}_{t}^{i} \\
&= \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2}+\alpha^{2}\gamma-2\alpha\eta \\
&= \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2}+\frac{\eta^{2}}{\gamma}-2\frac{\eta^{2}}{\gamma} \\
&= \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2}-\frac{\eta^{2}}{\gamma} \\
&< \sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2}.
\end{aligned} \tag{A.34}$$

Based on the definition of strongly monotone at a point and the assumption in this theorem, the following equation can be attained

$$\eta=\sum_{i=1}^{N}\left(\mathbf{x}_{t}^{i}-\mathbf{x}_{*}^{i}\right)^{T}\hat{\mathbf{D}}\mathbf{h}_{t}^{i}\geq m\sum_{i=1}^{N}\left\|\mathbf{x}_{t}^{i}-\mathbf{x}_{*}^{i}\right\|_{2}^{2} \tag{A.35}$$

$$\gamma=\sum_{i=1}^{N}\left\|\hat{\mathbf{D}}\mathbf{h}_{t}^{i}\right\|_{2}^{2}\leq NH+M\sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2} \tag{A.36}$$

Also, let us denote the training error in the t^{th} iteration as E_t:

$$E_{t}=\sum_{i=1}^{N}\left\|\mathbf{x}_{*}^{i}-\mathbf{x}_{t}^{i}\right\|_{2}^{2} \tag{A.37}$$

According to Equation.(A.34),

$$\begin{aligned}
E_{t+1} &\leq E_{t}-\frac{\eta^{2}}{\gamma} \\
&\leq E_{t}-\frac{m^{2}E_{t}^{2}}{NH+ME_{t}} \\
&= \left(1-\frac{m^{2}E_{t}}{NH+ME_{t}}\right)E_{t}
\end{aligned} \tag{A.38}$$

Recursively applying the above inequality, then

$$E_{t+1} \leq E_0 \prod_{l=0}^{t} \left(1 - \frac{m^2 E_l}{NH + ME_l}\right) \tag{A.39}$$

Since $E_t \geq 0$ for all t, and

$$0 < \left(1 - \frac{m^2 E_l}{NH + ME_l}\right) < 1 \tag{A.40}$$

thus

$$\lim_{t \to +\infty} E_{t+1} = 0 \tag{A.41}$$

Next, the case where $H = 0$ is considered. In this case,

$$0 \leq E_{t+1} \leq E_t \left(1 - \frac{m^2}{M}\right) = E_0 \left(1 - \frac{m^2}{M}\right)^{t+1} \tag{A.42}$$

This proves that the training error converges linearly to zero.

Appendix B

Codes

B.1 General Discriminative Optimisation Method

Learning regressors

```
function [DM, error4] = learnGDO(Xmodel,X,Y,nMap,sigmaSq,
    gridStep)
% Input: Xmodel,X,sigmaSq,gridStep are used to calculate the feature Hf
% Xmodel - orinigal model
% X - the model with perturbations
% Y - groundtruth (parameter vector)
% Output: The learned regressor

fprintf('Training GDO with %d maps\n',nMap)
fprintf('#data: %d \n',length(X))

D = cell(1,nMap);
error4 = inf(nMap,1);
Xori = X;
Yori = Y;

[pcmptFeat,normals] = precomputeFeature(Xmodel,sigmaSq,
    gridStep);
Ygoal = transMatToParam(Yori);
Yinit = zeros(6,length(X));% the parameters are initialized to 0
Y = Yinit;
fprintf('It: %d, err: %f\n',0,norm(Ygoal-Y,'fro').^2/length
    (X))

gridStep=0.05;
x = -2:gridStep:2;
[X1,Y1,Z1] = meshgrid(x,x,x);
```

```
D1 = [X1(:) Y1(:) Z1(:)]';
[F,Mdl] = PDFnorm(Xmodel',D1); %calculate the probability density function

for itMap =1:nMap
 featX = extFeat(X,pcmptFeat); %extract the coordinate-based feature
 featXF = computeFeatureturpdf01(Xmodel',X,sigmaSq,normals,F,
    Mdl);%extract the density-based
    feature
 [lameda1,lameda2] = FEATXFXGDO( featX,featXF );%calculate the
    weights for features
 FeatX=[lameda2*featX;lameda1*featXF];%achieve the cooperation of
    different features
 featY = Y - Ygoal;
 D{itMap} = (featY*FeatX')/(FeatX*FeatX'/length(X)+1e-4*(
    eye(size(FeatX,1))))/length(X);%attain the learned
    regressor
 clear featX FeatX featXF featY

 rotMat = paramToTransMatnew(Y);
 X = transCellnew(rotMat,Xori,0);
 error4(itMap) = norm(Ygoal-Y,'fro')^2/length(X);
 fprintf('It: %d, err: %f\n',itMap,error4(itMap))

 s=0;
 if ~(itMap==1)
    deltaD=D{itMap}-D{itMap-1};
    for j=1:size(deltaD,2)
     s = s+ norm(deltaD(:,j));
     DD=s/size(deltaD,2);
    end
    if DD<10^(-4)
       break;
    end
end
end

%save data
DM = struct();
DM.Dmap = D;
DM.runOrder = 1:length(D);
DM.trainErr =error4;
DM.Xmodel = Xmodel;
DM.pcmptFeat = pcmptFeat;
end
```

Calculating Coordinate-based feature

```
function [pcmptFeat,normals] = precomputeFeature(Xmodel,
    sigmaSq,gridStep)
%PRECOMPUTEFEATURE

%set parameters
sparseMapThreshold = 1e-6;
x = -2:gridStep:2;
[X,Y,Z] = meshgrid(x,x,x);
D = [X(:) Y(:) Z(:)]';

%Compute the grids by finding all pairwise distance between all points in
%the model and the grids. This could cause error if memory is not large
%enough. If it happens, reduce the grid size or write a loop instead.
pcmptFeat.map = exp(-pwSqDist(Xmodel,D)/sigmaSq);
pcmptFeat.minVal = -2;
pcmptFeat.size = length(x)*[1 1 1];
pcmptFeat.maxSize = pcmptFeat.size(1);
pcmptFeat.numStepPerUnit = 1/gridStep;

%calculate normal vector
[ normals ] = findPointNormals(Xmodel',6,[0 0 0]);
dirMat = bsxfun(@gt,normals*D,sum(normals.*Xmodel',2));

clear D

%compute grid in front and in the back of normal vectors
mapTmp1 = pcmptFeat.map;
mapTmp1(dirMat) = 0;%mapTmp1 means the grid is on the front of the model
    point
mapTmp2 = pcmptFeat.map;
mapTmp2(~dirMat) = 0;
pcmptFeat.map = [mapTmp1;mapTmp2];

clear mapTmp1
clear mapTmp2
%set small values to zero and make the matrix sparse
if sparseMapThreshold > 0
    spPcmptFeatTmp = pcmptFeat.map;
    spPcmptFeatTmp(spPcmptFeatTmp < sparseMapThreshold) =
        0;
```

```
% pcmptFeat.map = sparse(spPcmptFeatTmp);
    pcmptFeat.map = sparse(spPcmptFeatTmp);
end

clear spExpMapTmp
end
```

```
function [featX] = extFeat(X, pcmptFeat)
%EXTFEAT extract features from precomputed map
    sizeMapVec = [size(pcmptFeat.map,2) 1];

    featX = zeros(size(pcmptFeat.map,1),length(X));

    % loop over each shape in X
    for i = 1:length(X)
        tmp = X{i};

        % remove points too far
        tmp(:,any(abs(tmp) > 2,1)) = [];

        % compute index of each point in the map
        tmp = round((tmp-pcmptFeat.minVal)*pcmptFeat.
            numStepPerUnit+1);
        tmp = tmp(2,:)+(tmp(1,:)-1)*pcmptFeat.size(2)+(tmp
            (3,:)-1)*pcmptFeat.size(2)*pcmptFeat.size(1);

        % compute the feature
        featX(:,i) = pcmptFeat.map*accumarray(tmp',1,
            sizeMapVec,[],[],1);
    end

    % normalise
    if length(X) > 2
        featX = bsxfun(@rdivide,featX,sum(featX,1));
    else
        featX = featX/sum(featX);
    end
    featX(~isfinite(featX)) = 0;

end
```

Calculating density-based feature

```
function [F,Mdl] = PDFnorm(modelSmp,D)
    Xmodel=modelSmp';
    Mdl = createns(D','Distance','euclidean');
    IdxNN = knnsearch(Mdl,Xmodel','K',1);% for each point in Q, to
        find the nearest k points in Mdl;
    I1=IdxNN(:,1);
    I1U=unique(I1);
    for i=1:length(I1U)
        [M{i,1},n] = find(I1==I1U(i,1));
        MC{i,1}=Xmodel(:,M{i,1}(1))';
        sigma=zeros(3);
        for k=1:size(MC{i,1},2)
            sigma=((MC{i,1}(:,k)-mean(MC{i,1},2))*(MC{i
                ,1}(:,k)-mean(MC{i,1},2))'+sigma);
        end
        Mmu{i,1}=mean(MC{i,1},2);
        Sigmamatrix{i,1}=sigma/k;
    end
    for i=1:length(Xmodel)
        a=find(I1U==I1(i,1));
        F(i,1)= exp(-1*(Xmodel(:,i)-Mmu{a,1})'*pinv(
            Sigmamatrix{a,1})*(Xmodel(:,i)-Mmu{a,1})/2);
    end
end
```

```
function [featX] = computeFeatureretpdf01(model,X,sigmaSq,
    normals,F,Mdl)
  for i=1:length(X) % the X is the cell
     X{i}=X{i}(1:3,:);
     tmp=X{i};
     [F1] = PDFnorm02(tmp,Mdl);
     criteria = exp(-pwSqDistfeature(F,F1)/sigmaSq);
     dirMat = bsxfun(@gt,normals*tmp,sum(normals.*model',2)
         );
     mapXmodel21 = criteria;
     mapXmodel21(dirMat) =0;% scene is in the front of the object
     mapXmodel22 = criteria;
     mapXmodel22(~dirMat) = 0;% scene is in the back of the object
     mapXmodel21=mapXmodel21/sum(mapXmodel21);
     mapXmodel22=mapXmodel22/sum(mapXmodel22);
     featX(:,i) = [mapXmodel21;mapXmodel22];
     featX(~isfinite(featX)) = 0;
```

```
        for j=1:length(featX(:,i))
            if featX(j,i)<10^(-6)
                featX(j,i)=0;
            end
        end
    end
end
```

Calculating weights of features

```
function [ lameda1,lameda2 ] = FEATXFXGDO( featX,featXF )
    A=cov(featXF');
    B=cov(featX');
    a=trace(A);
    b=trace(B);
    lameda1=a/(a+b);
    lameda2=b/(a+b);
end
```

Estimating transformation

```
function [Y,Ystep,Xstep] = inferGDO(D,X,maxIter,stopThres,
    sigmaSq,normals,F,Mdl)
%INFER Infer the transformation
%
% INPUT
% D object contains the precomputed features
% X : 3 x n matrix - 3D scene shape
% maxIter : 1 x 1 matrix - maximum number of iterations
% stopThres : 1 x 1 matrix - stop threshold for terminating update
%
% OUTPUT
% Y : 3 x 4 matrix - output transformation
% Ystep : cell of 3 x 4 matrices - transformation in each step
% Xstep : cell of 3 x n matrices - transformed scene in each step
%
Y = zeros(6,length(X));
Xmodel = D.Xmodel;
id = cellfun('length',D.Dmap);
D.Dmap(id==0)=[];
n = length(D.Dmap);
Xout = cell(n,1);
Ystep = cell(n,1);
Xstep = cell(n,1);
```

```
Xori = X;

for i = 1:length(X)
    X{i}(:,any(abs(X{i}) > 3)) = [];
end
for itMap = 1:n
 featX = extFeat(X,D.pcmptFeat);
 featXF = computeFeaturetpdf01(Xmodel',X,sigmaSq,normals,F,
    Mdl);
 [lameda1,lameda2] = FEATXFXGDO( featX,featXF );
 FeatX=[lameda2*featX;lameda1*featXF];
     update = D.Dmap{itMap}*FeatX;
     Y = Y - update;
     rotMat = paramToTransMat(Y);
     X = transCell(rotMat,Xori,0);

     %save transformation in each step
     Ystep{itMap} = {reshape(paramToTransMat(Y),3,4)};

     %save shape in each step
     Xout{itMap} = X;

     %save update
     updateList(:,itMap) = update;
end

%continue loop iwith last map f update is still large
while norm(update,2) > stopThres && itMap<maxIter
     itMap = itMap + 1;
     featX = extFeat(X,D.pcmptFeat);
     featXF = computeFeaturetpdf01(Xmodel',X,sigmaSq,normals
        ,F,Mdl);
    [lameda1,lameda2] = FEATXFXGDO( featX,featXF );
    FeatX=[lameda2*featX;lameda1*featXF];
     update = D.Dmap{end}*FeatX;
     Y = Y - update;
     %transform X
     rotMat = paramToTransMatnew(Y);
     X = transCellnew(rotMat,Xori,0);


    Ystep{itMap} = {reshape(paramToTransMat(Y),3,4)};
```

```
63      Xout{itMap} = X;
64      updateList(:,itMap) = update;
65  end
66
67  % last step
68  % Yout = rotMat;
69  Yout = {reshape(paramToTransMat(Y),3,4)};
70
71  Xstep = Xout;
72  % if size(Xmodel,2)>3
73  [Y,Ystep,Xstep] = invertInferDOCell(Yout,Ystep,Xmodel');
74  else
75  [Y,Ystep,Xstep] = invertInferDOCell(Yout,Ystep,Xmodel);
76  end
77  end
78
79  function [Y,Ystep,Xstep] = invertInferDOCell(Y,Ystep,X)
80  %INVERTINFERDOCELL Invert the result of DO to get transformation that
81  %transforms model to scene
82  %
83  % Input
84  % Y : 3 x 4 matrix - transformation matrix that is output from INFER, i.e.,
85  % transform scene to 3D model
86  % Ystep : cell of 3 x 4 matrices - transformation matrix that is output from INFER, i.e.,
87  % transform scene to 3D model in each step
88  % X : 3 x n matrix - 3D model of object
89  %
90  % Output
91  % same as input
92  if ~exist('Ystep','var')
93      Ystep = [];
94  end
95
96  if ~exist('X','var')
97      X = [];
98  end
99
100 % invert Y
101 for i = 1:length(Y)
102     Y{i} = [Y{i}(1:3,1:3)' -Y{i}(1:3,1:3)'*Y{i}(1:3,4)];
103 end
104
```

```
105 Xstep = [];
106
107 %invert Ystep
108 if ~isempty(Ystep)
109     for itMap = 1:length(Ystep)
110         for i = 1:length(Ystep{itMap})
111             Ystep{itMap}{i} = [Ystep{itMap}{i}(1:3,1:3)' -
                    Ystep{itMap}{i}(1:3,1:3)'*Ystep{itMap}{i
                    }(1:3,4)];
112         end
113     end
114
115     %invert X back
116     if ~isempty(X)
117         Xstep = cell(length(Ystep),1);
118
119         for itMap = 1:length(Ystep)
120
121             Xstep{itMap} = cell(length(Ystep{itMap}),1);
122
123             for i = 1:length(Ystep{itMap})
124                 Xstep{itMap}{i} = X;
125                 Xstep{itMap}{i}(:,4) = 1;
126                 Xstep{itMap}{i} = Ystep{itMap}{i}*Xstep{
                        itMap}{i}';
127             end
128
129         end
130
131     end
132
133
134 end
135 end
136
137 %
```

B.2 Rewighted Discriminative Optimisation Method

Learning regressors

```
function [DM,error] = learnRDO(Xmodel,X,Y,nMap,pcmptFeat,
    normals)

    fprintf('Training DO with %d maps\n',nMap)
    fprintf('#data: %d \n',length(X))

    D1 = cell(1,nMap);
    error = inf(nMap,1);
    Xori = X;
    Yori = Y;
    Ygoal = transMatToParam(Yori);
    Yinit = zeros(6,length(X));
    Y = Yinit;
    R=cell(nMap,1);
    fprintf('It: %d, err: %f\n',0,norm(Ygoal-Y,'fro').^2/
       length(X))
    Ykm=0.1*eye(6);
    ykm=zeros(6);
    lameda=1e-4;
    for itMap = 1:nMap
        for k=1:length(X)
            YY1=Ygoal(:,k);%groundtruth
            YY2=Y(:,k); %estimated
            rotMat22 = paramToTransMatnew(YY2);
            XX2=transCellnew(rotMat22,Xmodel,0);
                for j=1:6
                    ykm(:,j)=YY1(:,1)-Ykm(:,j);
                    rotMatkm = paramToTransMatnew(ykm(:,j));
                    Xkm = transCellnew(rotMatkm,Xmodel,0);
                    MSEkm2(k,j)=immseNAN(XX2',Xkm');
                end
        end
          N = normalize(MSEkm2,2);
        for i=1:6
            [mu(i,1),sigma(i,1)]=normfit(N(:,i));
        end
            Mu=min(mu);
            Index=find(mu==Mu);
        for p=1:6
            Yy(p,1)=normpdf(mu(p,1),mu(Index,1),sigma(Index
               ,1));
        end
```

```
            N2ory=0.5*(ones(6,1)-Yy).^itMap;
            MSEMSE=exp(N2ory);
            A=pinv(diag(MSEMSE.^2))*lameda*length(X);
            featX = extFeat(X,pcmptFeat);
            B=featX*featX';
            featY = Y - Ygoal;
            C=featY*featX';
            D1{itMap} = sylvester(A,B,C);
            Y = Y - D1{itMap}*featX;
            rotMat = paramToTransMatnew(Y);
            R{itMap,1}=rotMat;
            X = transCellnew(rotMat,Xori,0);
            error(itMap) = norm(Ygoal-Y,'fro')^2/length(X);
        end

DM = struct();
DM.Dmap = D1;
DM.runOrder = 1:length(D1);
DM.trainErr = error;
DM.Xmodel = Xmodel;
DM.pcmptFeat = pcmptFeat;
DM.normals = normals;
end
```

Estimating transformation

```
function [Y,Ystep,Xstep,error,Ygoal] = inferJX(D,X,Yori,
    maxIter,stopThres)
%INFER Infer the transformation
%
% INPUT
% D object contains the precomputed features
% X : 3 x n matrix - 3D scene shape
% maxIter : 1 x 1 matrix - maximum number of iterations
% stopThres : 1 x 1 matrix - stop threshold for terminating update
%
% OUTPUT
% Y : 3 x 4 matrix - output transformation
% Ystep : cell of 3 x 4 matrices - transformation in each step
% Xstep : cell of 3 x n matrices - transformed scene in each step
%
Y = zeros(6,length(X));
Xmodel = D.Xmodel;
```

```
n = length(D.Dmap);
Xout = cell(n,1);
Ystep = cell(n,1);
Xori = X;
Ygoal = transMatToParam(Yori);
% for i = 1:length(X)
% Xi(:,any(abs(Xi) > 3)) = [];
% end

for itMap = 1:n

    % compute ture
    featX = extFeat(X,D.pcmptFeat);

    % update
    update = D.Dmap{itMap}*featX;
    Y = Y - update;


    % convert to tansformation and transform X
    rotMat = paramToTransMat(Y);
    X = transCell(rotMat,Xori,0);

    % save transformation in each step
    Ystep{itMap} = {reshape(paramToTransMat(Y),3,4)};

    % save shape in each step
    Xout{itMap} = X;

% end

% continue loop iwith last map f update is still large
while norm(update) > stopThres && itMap < maxIter
    itMap = itMap + 1;

    % extract feature
    featX = extFeat(X,D.pcmptFeat);

    % update
    update = D.Dmap{end}*featX;
    Y = Y - update;

```

```
59      %transform X
60      rotMat = paramToTransMat(Y);
61      X = transCell(rotMat,Xori,0);
62
63      %save param
64      Ystep{itMap} = {reshape(paramToTransMat(Y),3,4)};
65      Xout{itMap} = X;
66  %updateList(:,itMap) = update;
67  end
68  Ystep1=Ystep;
69  %last step
70  Yout = {reshape(paramToTransMat(Y),3,4)};
71  error= norm(Ygoal-Y,'fro')^2/length(X);
72  %invert transformation
73  [Y,Ystep,Xstep] = invertInferDOCell(Yout,Ystep,Xmodel);
74  %[YCHANG,Ystep1,XstepCHANG] = invertInferDOCell(Yout,Ystep1,modelFull);
75  end
76
77  function [Y,Ystep,Xstep] = invertInferDOCell(Y,Ystep,X)
78  %INVERTINFERDOCELL Invert the result of DO to get transformation that
79  %transforms model to scene
80  %
81  %Input
82  %Y : 3 x 4 matrix - transformation matrix that is output from INFER, i.e.,
83  %transform scene to 3D model
84  %Ystep : cell of 3 x 4 matrices - transformation matrix that is output from INFER, i.e.,
85  %transform scene to 3D model in each step
86  %X : 3 x n matrix - 3D model of object
87  %
88  %Output
89  %same as input
90  if ~exist('Ystep','var')
91      Ystep = [];
92  end
93
94  if ~exist('X','var')
95      X = [];
96  end
97
98  %invert Y
99  for i = 1:length(Y)
100     Y{i} = [Y{i}(1:3,1:3)' -Y{i}(1:3,1:3)'*Y{i}(1:3,4)];
```

```
101 end
102
103 Xstep = [];
104
105 %invert Ystep
106 if ~isempty(Ystep)
107     for itMap = 1:length(Ystep)
108         for i = 1:length(Ystep{itMap})
109             Ystep{itMap}{i} = [Ystep{itMap}{i}(1:3,1:3)' -
                    Ystep{itMap}{i}(1:3,1:3)'*Ystep{itMap}{i
                    }(1:3,4)];
110         end
111     end
112
113     %invert X back
114     if ~isempty(X)
115         Xstep = cell(length(Ystep),1);
116
117         for itMap = 1:length(Ystep)
118
119             Xstep{itMap} = cell(length(Ystep{itMap}),1);
120
121             for i = 1:length(Ystep{itMap})
122                 Xstep{itMap}{i} = X;
123                 Xstep{itMap}{i}(4,:) = 1;
124                 Xstep{itMap}{i} = Ystep{itMap}{i}*Xstep{
                        itMap}{i};
125             end
126
127         end
128
129     end
130
131
132 end
133 end
```

B.3 SGRTmreg

Extracting key points

```
[G1,SH1,UW1,TBindextrain,BOttrain,DT1] = Graphdraw02(double
    (modelSmp));
indexex3t= find((UW1==TBindextrain(end))|(UW1==TBindextrain
    (end-1)));
indextrain = unique([BOttrain;indexex3t]);
ModeltExtractedO = modelSmp(:,indextrain);
```

```
function [G1,SH,DSH,TBindextest,indexboundary,DT,modeltP1]
    = Graphdraw02(modeltP)
modeltP2=unique(modeltP','rows');
modeltP1=modeltP2';
DT = delaunayTriangulation(modeltP1(1:2,:)');
Dtmatrix=zeros(length(modeltP),length(modeltP));
DTC=DT.ConnectivityList;
for i=1:length(DTC)
    Dtmatrix(DTC(i,1),DTC(i,2))=Dtmatrix(DTC(i,1),DTC(i,2))
        +1;
    Dtmatrix(DTC(i,1),DTC(i,3))=Dtmatrix(DTC(i,1),DTC(i,3))
        +1;
    Dtmatrix(DTC(i,2),DTC(i,3))=Dtmatrix(DTC(i,2),DTC(i,3))
        +1;
    Dtmatrix(DTC(i,2),DTC(i,1))=Dtmatrix(DTC(i,2),DTC(i,1))
        +1;
    Dtmatrix(DTC(i,3),DTC(i,1))=Dtmatrix(DTC(i,3),DTC(i,1))
        +1;
    Dtmatrix(DTC(i,3),DTC(i,2))=Dtmatrix(DTC(i,3),DTC(i,2))
        +1;
end
tf = issymmetric(Dtmatrix);
G1=graph(Dtmatrix);
UW=degree(G1);
TB=sortrows(tabulate(UW),2);
a1=DT.ConnectivityList;
SH=subgraph(G1,unique(a1));
DSH=degree(SH);
TBDSH=sortrows(tabulate(DSH),2);
if length(TBDSH)<10
    a=10-length(TBDSH);
    TBindextest=[zeros(a,1);TBDSH(:,1)];
else
    TBindextest=TBDSH(end-9:end,1);
end
```

```
[bidstX2t, EtX2t, NetX2t] = findtdelaunaytboundary03fig1(
    double(modeltP1'),0.4);
indexboundary=[];
for i=1:length(bidstX2t)
indexboundary=[bidstX2t{1,i};indexboundary];
end
close;
end
```

Searching scheme

```
clc;clear;
load('E:\BaiduNetdiskDownload\Dataset\
    MOdelNet40tplytdataset\ModelNet40tplytdownsamplingt1024\
    car\matlab.mat')
%the key points in a Car model
ptclou01=modeltPtCar{1,2};
for k=1:length(modeltPtCar)-1
    if k==1
        ptCloud01tScreen01{k,1}=modeltPtCar{1,k};
    else
        ptCloud01tScreen01{k,1}=modeltPtCar{1,k+1};
    end
end
%similarity on degree list
[C,B] = Screen01(TBindexCar);
cs = C{2,1};
for j=1:length(cs)
ptCloud02tScreen02{j,1}=modeltPtCar{1,j};%Candidates
end
%DPGMM
for i=1:length(ptCloud02tScreen02)
    ptCloud03{i,1}=[ptCloud02tScreen02{i,1},ptclou01]';
    [n,d] = size(ptCloud03{i,1});
      ss = 0.1;
      sm = 30;
%if niw1=niw2=niw,partition of the final transformed model groups one cluster
%if niw1 =niw2 =niw, partition groups two clusters
      [niw1] = Gausstinititp01(d,ptCloud02tScreen02{i,1}',
          sm,ss); %the initial guass information for the fixed
          model
      [niw2] = Gausstinititp01(d,ptclou01',sm,ss); %the initial
          guass information for the moving model
      [niw] = Gauss01(d,ss,sm); %the intial guass for unknown cluster
```

```
29          [dpmm, dpmmtposterior, dpmmttime] = DPMMtgauss01(
                ptCloud03{i,1},niw1,niw2,niw,50);
30          dpmm1 = dpmm.z(1:1024);
31          dpmm2 = dpmm.z(1025:end);
32          c1ttraining = length(find(dpmm1==1));
33          c2ttraining = 1024-c1ttraining;
34          c1treference = length(find(dpmm2==1));
35          c2treference = 1024-c1treference;
36          Rtraining(i,1) =  c1ttraining/1024;
37          Rtraining(i,2) =  c2ttraining/1024;
38          Rreference(1,1) = c1treference/1024;
39          Rreference(1,2) = c2treference/1024;
40  end
41  Rtraining01 = Rtraining(:,1)- Rtraining(:,2);
42  Rreference01 = Rreference(1,1)- Rreference(1,2);
43  indexscreen02 = find(Rtraining01*Rreference01>0);
44  for j=1:length(indexscreen02)
45      ptCloud03tScreen03{j,1} = ptCloud02tScreen02{
            indexscreen02(j,1),1};%
            Candidates
46      [bidstX2t, EtX2t, NetX2t] =
            findtdelaunaytboundary03fig1(double(
            ptCloud03tScreen03{j,1}'),0.1);
47      G1{1,j}=graph(NetX2t+NetX2t');
48      Score1(:,j) =  centrality(G1{1,j},'eigenvector');
49  end
50  %similarity on closeness
51  [bidstX22t, EtX22t, NetX22t] = findtdelaunaytboundary03fig1
        (double(ptclou01)',0.1);
52  G2=graph(NetX22t+NetX22t');
53  Score2 = centrality(G2,'eigenvector');
54  Sm2=mean(Score2);
55  Sm1=mean(Score1);
56  indexscreen03=find(Sm1==Sm2);
57  for j=1:length(indexscreen03)
58      ptCloud04tScreen04{j,1}= ptCloud03tScreen03{
            indexscreen03(j),1};%
            Candidates
59  end
60  %similarity on normal vector
61  PTcloud01=pointCloud(ptclou01');
62  normal01=pcnormals(PTcloud01);
63  for k=1:length(ptCloud04tScreen04)
```

```
64      PTcloud02t1{k,1}=pointCloud(ptCloud04tScreen04{k,1}');
65      normal02t1{k,1}=pcnormals(PTcloud02t1{k,1});
66  end
67  NormalVectors = [normal02t1{1,1};normal02t1{2,1}];
68  NE=zeros(length(NormalVectors),length(normal01));
69  for i=1:size(NE,1)
70       for j=1: size(NE,2)
71           NE(i,j)= sqrt(sum((NormalVectors(i,1)-normal01(j,1)
                 ).^2));
72       end
73  end
74  for k=1:size(NE,2)
75      a=find(NE(:,k)==min(NE(:,k)));
76      indexfinal(k,1) = a(1);
77  end
78  indexf01=sort(indexfinal);
79  a1=length(find(indexf01<=1024));
80  a2=length(find(indexf01>1024));
81  if a1>a2
82      selectedtptcloud = ptCloud04tScreen04{1,1};
83  else
84      selectedtptcloud = ptCloud04tScreen04{2,1};
85  end
86
87  figure;
88  subplot(1,2,1)
89  pcshow(ptclou01')
90  subplot(1,2,2)
91  pcshow(selectedtptcloud')
```

```
1   function [C,B,Correctiontmatrix] = Screen01(TBindexairplane
        )
2       TBindex1=TBindexairplane;
3       TBindex=TBindex1(4:end,:);
4       Correctiontmatrix=zeros(length(TBindex),length(TBindex)
            );
5       for i=1:length(TBindex)
6           for j=1:length(TBindex)
7               Correctiontmatrix(i,j)=1-pdist2(TBindex(:,i)',
                    TBindex(:,j)','hamming');%
                    similarity
8           end
```

```
    end
    B=Correctiontmatrix;
    B(boolean(eye(length(Correctiontmatrix))))=-100;
    C=cell(length(B),1);
    for i=1:length(C)
        C{i,1}=find(B(i,:)>0.8);
    end
end
```

```
function [niw1] = Gausstinititp01(d,X1,sm,ss)
            niw1.d   = d;                          %the dimensions of data
            niw1.mu0 = (mean(X1,1))';              %prior mean
            niw1.k0  = 1;                          %how strongly we believe in m0
                prior
            niw1.S0 =diag(diag(cov(X1)));
            niw1.nu0 = d+1; %how strongly we believe in S0 prior (dof)
                nu0>D-1
            niw1.ss  = (sm*sm)/(ss*ss);    %cluster spread
            niw1.rr  = 1/niw1.ss;          %1/cluster spread
            niw1.SS  = niw1.nu0*niw1.S0; %nu0 * S0
end
```

```
function [niw] = Gauss01(d,ss,sm)
          %Dtermine the parameters of initial guass
          niw.d   = d;                    %the dimensions of data
          niw.mu0 = zeros(d,1);           %prior mean
          niw.k0  = 1;                    %how strongly we believe in m0 prior
          niw.S0  = 2*ss*ss*eye(d);    %prior mean for Sigma
          niw.nu0 = d+1;   %how strongly we believe in S0 prior (dof) nu0>D-1
          niw.ss  = (sm*sm)/(ss*ss);          %cluster spread
          niw.rr  = 1/niw.ss;             %1/cluster spread
          niw.SS  = niw.nu0*niw.S0; %nu0 * S0
end
```

```
function [dpmm, dpmmtposterior, dpmmttime] = DPMMtgauss01(X
    ,niw1,niw2,niw,numgibbs)
%
%clear all; close all;
rng('default');
options.generateplots = 1;

%
[n,d] = size(X);
```

```
9
10 trueK=4;       %true number of clusters
11 trueZ=ceil(trueK*(1:n)/n); %true labels
12
13 K = 2;         %init number of clusters
14 trueKfake=K;
15 alpha=0.01;      %concentration parameter
16 z=ceil(trueKfake*(1:n)/n); %true labels
17
18 %init dpmm
19 gauss1 = gaussinit(niw1); %init gauss struct
20 gauss2 = gaussinit(niw2); %init gauss struct
21 gauss = gaussinit(niw); %init gauss struct
22 Gauss=cell(3,1);
23 Gauss{1,1}=gauss1;
24 Gauss{2,1}=gauss2;
25 Gauss{3,1}=gauss;
26 [dpmm] = dpmminit01(K,alpha,Gauss, X, z);
27 recordK = zeros(numgibbs,1);
28 dpmmttime = zeros(numgibbs,1);
29 Ns = [1 2 3 10 50 100 150 numgibbs]; cnt = 1;
30
31 for iter=1:numgibbs
32
33     recordK(iter) = dpmm.K;
34     if (iter == Ns(cnt) && options.generateplots == 1)
35         cnt = cnt + 1;
36     end
37     fprintf('gibbs iter: %d\n', iter);
38  %if (iter == numgibbs)
39         numK = dpmm.K+1;
40         dpmmtposterior = zeros(n,numK);
41     end
42
43     tic;
44     for ii = 1:n
45         k = dpmm.z(ii);
46         %remove xi statistics from component zi
47         dpmm.nk(k) = dpmm.nk(k) - 1;
48         dpmm.gm{k} = delitem(dpmm.gm{k}, dpmm.X(ii,:));
49
50         if (dpmm.nk(k) == 0)
```

```
51              %delete empty component
52              dpmm.gm(k) = [];
53              dpmm.K = dpmm.K - 1;
54              dpmm.nk(k) = [];
55              idx = find(dpmm.z>k);
56              dpmm.z(idx) = dpmm.z(idx) - 1;   %re-label component
57          end
58          logpzi = log([dpmm.nk; dpmm.alpha]);
59          for kk=1:dpmm.K+1
60              logpzi(kk) = logpzi(kk) + logpredictive(dpmm.gm{
                    kk}, dpmm.X(ii,:));
61          end
62          pzi = exp(logpzi - max(logpzi)); %numerical stability
63          pzi = pzi/sum(pzi);
64          if (iter == numgibbs), dpmmtposterior(ii,1:numK) =
                pzi(1:numK); end
65
66          u = rand;
67          k = find(u < cumsum(pzi),1,'first');
68
69          if (k==dpmm.K+1)
70              %create a new cluster
71              dpmm.gm{k+1} = dpmm.gm{k}; %warm init
72              dpmm.K = dpmm.K + 1;
73              dpmm.nk= [dpmm.nk; 0];
74          end
75          dpmm.z(ii) = k;
76          dpmm.nk(k) = dpmm.nk(k) + 1;
77          dpmm.gm{k} = additem(dpmm.gm{k}, dpmm.X(ii,:));
78
79      end
80      dpmmttime(iter) = toc;
81  end
82
83  SAVEtPATH = './';
84  save([SAVEtPATH,'dpmm.mat']);
85
86  %if (options.generateplots)
87  figure; plot(recordK)
88  end
89
90  end
```

```
91
92  function [dpmm] = dpmminit(K,alpha,gauss, X, z)
93
94  dpmm.K = K;
95  dpmm.N = size(X,1);
96  dpmm.alpha = alpha;
97  dpmm.gm = cell(1,K+1);   %cell of structs!
98  dpmm.X = X;
99  dpmm.z = z;
100 dpmm.nk = zeros(K,1);
101
102 %init mixture components
103 for kk = 1:K+1
104     dpmm.gm{kk} = gauss;
105 end
106
107 %add data items to mixture components
108 for i = 1:dpmm.N
109     k = z(i);
110     dpmm.gm{k} = additem(dpmm.gm{k},X(i,:));
111     dpmm.nk(k) = dpmm.nk(k) + 1;
112 end
113
114 end
115
116 function [gauss] = additem(gauss, xi)
117
118 xi = xi(:);
119 gauss.n = gauss.n + 1;
120 gauss.rr = gauss.rr + 1;
121 gauss.nu0 = gauss.nu0 + 1;
122 gauss.Sigma = cholupdate(gauss.Sigma,xi);
123 gauss.mu = gauss.mu + xi;
124
125 end
126
127 function [gauss] = delitem(gauss, xi)
128
129 xi = xi(:);
130 gauss.n = gauss.n - 1;
131 gauss.rr = gauss.rr - 1;
132 gauss.nu0 = gauss.nu0 - 1;
```

```
133 gauss.Sigma = cholupdate(gauss.Sigma,xi,'-');
134 gauss.mu = gauss.mu - xi;
135
136 end
137
138 function [gauss] = gaussinit(niw)
139 %create a gaussian struct with no data items
140
141 gauss.d = niw.d;
142 gauss.n = 0;                    % init number of items.
143 gauss.rr = niw.rr;
144 gauss.nu0 = niw.nu0;
145 gauss.Sigma = chol(niw.SS + niw.rr*niw.mu0*niw.mu0'); %
      Cholesky factorization
146 gauss.mu = niw.rr*niw.mu0;
147 gauss.Z0 = ZZ(niw.d,gauss.n,gauss.rr,gauss.nu0,gauss.Sigma,
      gauss.mu);
148
149 end
150
151 function ll = logpredictive(gauss,xi)
152 xi = xi(:);
153 ll =   ZZ(gauss.d,gauss.n+1,gauss.rr+1,gauss.nu0+1,
      cholupdate(gauss.Sigma,xi),gauss.mu+xi) ...
154     - ZZ(gauss.d,gauss.n   ,gauss.rr   ,gauss.nu0   ,
                  gauss.Sigma      ,gauss.mu);
155 end
156
157 function zz = ZZ(dd,nn,rr,vv,CC,XX)
158
159 zz = - nn*dd/2*log(pi) ...
160      - dd/2*log(rr) ...
161      - vv*sum(log(diag( cholupdate(CC,XX/sqrt(rr),'-')  )))
           ...
162      + sum(gammaln((vv-(0:dd-1))/2));
163
164 end
165
166 function y = logdet(A)
167 % Compute log(det(A)) where A is positive-definite
168 % This is faster and more stable than using log(det(A)).
169 try
```

```
170     U = chol(A);
171     y = 2*sum(log(diag(U)));
172 catch
173     y = 0;
174     warning('logdet:posdef', 'Matrix is not positive
            definite');
175 end
176
177 end
178
179 function [pipdf]=GEM(K, alpha)
180 %stick-breaking construction
181 beta=zeros(K,1);
182 pipdf=zeros(K,1);
183
184 for k=1:K
185     beta(k)=betarnd(1,alpha);
186     pipdf(k)=beta(k)*prod(1-beta);
187 end
188
189 end
190
191 function [] = dpmmplot(dpmm)
192
193 cmap = hsv(dpmm.K);
194
195 for kk = 1:dpmm.K
196   [mu sigma] = randsample(dpmm.gm{kk});
197   plotellipse01(mu,sigma,'color',cmap(kk,:),'linewidth',2);
198   hold on
199   ii = find(dpmm.z==kk);
200   d=length(mu);
201   if (d > 2)
202     [xcoeff, xscore] = pca(dpmm.X);
203     xpca = xscore(:,1:3);
204     xx = xpca(ii,:);
205   else
206     xx = dpmm.X(ii,:);
207   end
208 end
209
210 end
```

```
211
212 function el = plotellipse(xx,var,varargin)
213 % plot ellipse with centre xx and shape given by positive definite matrix var
214 [v,d] = eig(var);
215 d = sqrt(d);
216 theta = (0:.05:1)*2*pi;
217 xy = repmat(xx,1,length(theta))+d(1,1)*v(:,1)*sin(theta)+d
    (2,2)*v(:,2)*cos(theta);
218 dim = size(v,1);
219 if (dim <= 3)
220     el = plot3(xy(1,:),xy(2,:),xy(3,:),varargin{:});
221 end
222 end
223
224 function [mu,sigma] = randsample(gauss)
225 % [mu, sigma] = rand(qq)
226 % generates random mean mu and covariance sigma from the posterior of
227 % component parameters given data items in component.
228
229 CC = cholupdate(gauss.Sigma,gauss.mu/sqrt(gauss.rr),'-')\
    eye(gauss.d);
230
231 sigma = iwishrnd(CC,gauss.nu0,CC);
232 mu = mvnrnd(gauss.mu'/gauss.rr, sigma/gauss.rr)';
233
234 end
235
236
237 %
```

```
1 function [bids, E, Ne] = findtdelaunaytboundary03fig1(X,Fd)
2 %inputs:
3 %X = n by 3 array (Easting, Northing, Height) or (X,Y,Z)
4 %Fd = dmax (max point to point distance
5 %outputs
6 %bids: IDs to X that presents the sequence of points in the extracted
7 %boundary
8 %Please refer the paper: M. Awrangjeb, "Using point cloud data to identify,
9 %trace, and regularize the outlines of buildings" International Journal of Remote Sensing,
    Volume 37, Issue 3, February 2016, pages 551-579
10 %Open access at: http://www.tandfonline.com/doi/pdf/10.1080/01431161.2015.1131868
11
```

```
%just identification of boundary, no tracking (blocked/commented)
aThresh = 22.5/2; %standard 45 degree
dFd = 2*Fd;
msd = dFd*dFd;
msd1 = Fd*Fd;
TRI = delaunay(X(:,1),X(:,2));
TRI = sort(TRI,2);
%draw
figure;
triplot(TRI,X(:,1),X(:,2),'-c'); hold on;
xlim([-30,-9.5]); ylim([-32,+32]);%for A-shape

%for X
mnX = min(X(:,1));
mxX = max(X(:,1));
mnY = min(X(:,2));
mxY = max(X(:,2));
bb = 0.5; xlim([mnX-bb,mxX+bb]); ylim([mnY-bb,mxY+bb]);%for
    noisy shape


nP = size(X,1);
E = zeros(nP,nP) == 1;
Ne = zeros(nP,nP);%number of times an edge is shared by triangles (max 2, min 0)
Xr = [];%removed edges
for i = 1:size(TRI,1)
    T = TRI(i,:);
    Ne(T(1,1),T(1,2)) = Ne(T(1,1),T(1,2)) + 1;
    Ne(T(1,2),T(1,3)) = Ne(T(1,2),T(1,3)) + 1;
    Ne(T(1,1),T(1,3)) = Ne(T(1,1),T(1,3)) + 1;

    E(T(1,1),T(1,2)) = 1;
    E(T(1,2),T(1,1)) = 1;
    E(T(1,2),T(1,3)) = 1;
    E(T(1,3),T(1,2)) = 1;
    E(T(1,1),T(1,3)) = 1;
    E(T(1,3),T(1,1)) = 1;
end

%show boundary edges
for i = 1:nP
    for j = 1:nP
```

```
53          if Ne(i,j) == 1
54            plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-m'); hold
                  on;
55          end
56      end
57  end
58  plot(X(:,1), X(:,2), '.k', 'markersize', 10); hold on;
59  tv = [294; 295; 326; 327];
60  Q = [];
61  for i = 1:nP
62      for j = 1:nP
63          if (i < j && E(i,j) == 1 && Ne(i,j) == 1)
64                  sd = (X(i,1)-X(j,1))*(X(i,1)-X(j,1)) + (X(i
                        ,2)-X(j,2))*(X(i,2)-X(j,2));
65              if sd > msd
66                  out1 = [i j]
67                  if i == 2986 || j == 2986
68                      here = 1;
69                  end
70                  %remove this
71                  plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-r')
                        ; hold on;
72                  Xr = [Xr; [X(i,1:2), X(j,1:2)]];
73                  E(i,j) = 0;
74                  E(j,i) = 0;
75                  Ne(i,j) = 0;
76
77                  if sum(i == tv) == 1 || sum(j == tv) == 1
78                      here = 1;
79                  end
80                  %find other vertex k of triangle ijk and add edge ik and jk
81                  %as suspects
82                  k = find(E(i,:) & E(j,:) == 1);
83                  if size(k,2)>1
84                      here = 1;
85                      P1 = X(i,:);
86                      P2 = X(j,:);
87                      if (P2(1,1) - P1(1,1)) == 0
88                          P1(1,1) = P1(1,1)+0.001;
89                          X(i,1) = P1(1,1);
90                      end
```

```
91                          m = (P2(1,2) - P1(1,2))/(P2(1,1) - P1
                                (1,1));
92                          c = P1(1,2) - m*P1(1,1);
93                          Pks = X(k',:);
94                          num = sqrt(1 + m*m);
95                          den = m*Pks(:,1) - Pks(:,2) + c;
96                          dks = abs(den/num);
97                          [mn id] = min(dks);
98                          k = k(1,id);
99                      end
100                     if size(k,2) == 1%found
101                         %for edge (i,k)
102                         if i < k
103                             Ne(i,k) = Ne(i,k)-1;
104                             Q = [Q; [i k]];
105                         else
106                             Ne(k,i) = Ne(k,i)-1;
107                             Q = [Q; [k i]];
108                         end
109                         %for edge (j,k)
110                         if j < k
111                             Ne(j,k) = Ne(j,k)-1;
112                             Q = [Q; [j k]];
113                         else
114                             Ne(k,j) = Ne(k,j)-1;
115                             Q = [Q; [k j]];
116                         end
117                     else%not found, error! check if happen
118                     end
119                 else
120
121                 end
122             end
123         end
124 end
125
126 while size(Q,1)>0
127     i = Q(1,1); j = Q(1,2);
128     if E(i,j) == 1
129         %plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-y'); hold on;
130         sd = (X(i,1)-X(j,1))*(X(i,1)-X(j,1)) + (X(i,2)-X(j
                ,2))*(X(i,2)-X(j,2));
```

```
131             if sd > msd
132                 out2 = [i j]
133                 if i == 2986 || j == 2986
134                     here = 1;
135                 end
136                 %remove this
137                 plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-r');
                        hold on;
138                 Xr = [Xr; [X(i,1:2), X(j,1:2)]];
139                 E(i,j) = 0;
140                 E(j,i) = 0;
141                 Ne(i,j) = 0;
142                 %find other vertex k of triangle ijk and add edge ik and jk
143                 %as suspects
144                 if sum(i == tv) == 1 || sum(j == tv) == 1
145                     here = 1;
146                 end
147                 k = find(E(i,:) & E(j,:) == 1);
148                 if size(k,2)>1
149                     here = 1;
150                     P1 = X(i,:);
151                     P2 = X(j,:);
152                     if (P2(1,1) - P1(1,1)) == 0
153                         P1(1,1) = P1(1,1)+0.001;
154                         X(i,1) = P1(1,1);
155                     end
156                     m = (P2(1,2) - P1(1,2))/(P2(1,1) - P1(1,1))
                            ;
157                     c = P1(1,2) - m*P1(1,1);
158                     Pks = X(k',:);
159                     num = sqrt(1 + m*m);
160                     den = m*Pks(:,1) - Pks(:,2) + c;
161                     dks = abs(den/num);
162                     [mn id] = min(dks);
163                     k = k(1,id);
164                 end
165                 if size(k,2) == 1%found
166                     %for edge (i,k)
167                     if i < k
168                         Ne(i,k) = Ne(i,k)-1;
169                         Q = [Q; [i k]];
170                     else
```

```
171                         Ne(k,i) = Ne(k,i)-1;
172                         Q = [Q; [k i]];
173                     end
174                     %for edge (j,k)
175                     if j < k
176                         Ne(j,k) = Ne(j,k)-1;
177                         Q = [Q; [j k]];
178                     else
179                         Ne(k,j) = Ne(k,j)-1;
180                         Q = [Q; [k j]];
181                     end
182                 else%not found, error! check if happen
183                 end
184             else
185
186             end
187         end
188         %elseif size(Q,1)>1
189                 Q = Q(2:end,:);
190             else
191                 Q = [];
192             end
193         %end
194 end
195
196
197 %make removed edges white
198 for ii = 1:size(Xr,1)
199     plot([Xr(ii,1) Xr(ii,3)], [Xr(ii,2) Xr(ii,4)],'-w');
        hold on;
200 end
201 %show boundary edges
202 for i = 1:nP
203     for j = 1:nP
204         if Ne(i,j) == 1
205           plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-m'); hold
              on;
206         end
207     end
208 end
209 plot(X(:,1), X(:,2), '.k', 'markersize', 10); hold on;
210
```

```
211
212 Se = [];
213 F = zeros(nP,nP) == 1;%F(i,j) = 0: not checked before or decided to remove, 1:
        checked before and decided not to remove
214 while(1)
215     dE = 0;
216     for i = 1:nP
217         for j = 1:nP
218             if i == 2935 && j == 2986
219                 here = 1;
220             end
221             if (i < j && E(i,j) == 1 && Ne(i,j) == 1 && F(i
                    ,j) == 0)
222                 %plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-k'); hold on;
223                 sd = (X(i,1)-X(j,1))*(X(i,1)-X(j,1)) + (X(i
                        ,2)-X(j,2))*(X(i,2)-X(j,2));
224                 k = find(E(i,:) & E(j,:) == 1);
225                 if size(k,2)>1
226                     here = 1;
227                     P1 = X(i,:);
228                     P2 = X(j,:);
229                     m = (P2(1,2) - P1(1,2))/(P2(1,1) - P1
                            (1,1));
230                     c = P1(1,2) - m*P1(1,1);
231                     Pks = X(k',:);
232                     num = sqrt(1 + m*m);
233                     den = m*Pks(:,1) - Pks(:,2) + c;
234                     dks = abs(den/num);
235                     [mn id] = min(dks);
236                     k = k(1,id);
237                 end
238                 if size(k,2) == 1%found
239                     %find angle at vertex k for triangle ijk
240                     A = X(i,:);
241                     B = X(j,:);
242                     C = X(k,:);
243                     asq = (B(1,1)-C(1,1))*(B(1,1)-C(1,1)) +
                            (B(1,2)-C(1,2))*(B(1,2)-C(1,2));
244                     a = sqrt(asq);
245                     bsq = (A(1,1)-C(1,1))*(A(1,1)-C(1,1)) +
                            (A(1,2)-C(1,2))*(A(1,2)-C(1,2));
246                     b = sqrt(bsq);
```

```
247                     csq = (B(1,1)-A(1,1))*(B(1,1)-A(1,1)) +
                            (B(1,2)-A(1,2))*(B(1,2)-A(1,2));
248
249                     gamma = 180*acos((asq + bsq - csq)/(2*a
                        *b))/pi; %angle at
                        C
250                     if (gamma > 90 && sd > msd1) || abs
                        (180-gamma) <= aThresh %abtuse
                        angle
251                         dE = dE + 1;
252                         plot([X(i,1) X(j,1)], [X(i,2) X(j
                                ,2)],'-r'); hold on;
253                         Xr = [Xr; [X(i,1:2), X(j,1:2)]];
254                         E(i,j) = 0;
255                         E(j,i) = 0;
256                         Ne(i,j) = 0;
257                         %for edge (i,k)
258                         if i < k
259                             Ne(i,k) = Ne(i,k)-1;
260                             %Q = [Q; [i k]];
261                         else
262                             Ne(k,i) = Ne(k,i)-1;
263                             %Q = [Q; [k i]];
264                         end
265                         %for edge (j,k)
266                         if j < k
267                             Ne(j,k) = Ne(j,k)-1;
268                             %Q = [Q; [j k]];
269                         else
270                             Ne(k,j) = Ne(k,j)-1;
271                             %Q = [Q; [k j]];
272                         end
273                     else
274                         F(i,j) = 1;
275                     end
276                 else%not found, error! check if happen
277                     %mark this edge as a straight edge without forming a
                        triangle
278                     Se = [Se; [i j]];
279                 end
280             end
281         end
282     end
```

```
283     if dE == 0
284         break
285     end
286 end%while
287
288 %make removed edges white
289 for ii = 1:size(Xr,1)
290     plot([Xr(ii,1) Xr(ii,3)], [Xr(ii,2) Xr(ii,4)],'-w');
        hold on;
291 end
292 %show boundary edges
293 for i = 1:nP
294     for j = 1:nP
295         if Ne(i,j) == 1
296             plot([X(i,1) X(j,1)], [X(i,2) X(j,2)],'-m'); hold
                on;
297         end
298     end
299 end
300 plot(X(:,1), X(:,2), '.k', 'markersize', 10); hold on;
301
302 chkV = E;
303 fv = 0;PVs = [];
304 for i = 1:nP
305     for j = 1:nP
306         if i < j && E(i,j) == 1 && Ne(i,j) == 1
307             s = struct('f0', [], 'f1', [i j], 'f2', [i;j],
                'f3', []);%f0: parent struct, f1: edges, f2: vertices list, f3:
                children structs
308             fv = i;%first vertex
309             cv = j;%current vertex
310             pv = i;%previous vertex
311             plot([X(pv,1) X(cv,1)], [X(pv,2) X(cv,2)],'-c',
                'linewidth', 2); hold on;
312             break;
313         end
314         chkV(i,j) = 0;
315         chkV(j,i) = 0;
316     end
317     if fv > 0
318         break;
319     end
```

```
end
%i = 9;
j = 13;
s = struct('f0', [], 'f1', [i j], 'f2', [i;j], 'f3', []);
%if fv>0
    S{1} = s;
    El = [i j 1 1];
    chkV(i,j) = 0;
    chkV(j,i) = 0;
    count = 1;
    Q = [count];%queue for structs to explore
    is = i;
    js = j;
    Qactive = 1;
else
    Q = [];
    Qactive = [];
end
%B = []; B1 = [];
ccount = 1;PVall ={};
InsertLoop = [];
while size(Q,1)>0
    found = 0;
    prevSQ = size(Q,1);
    for qc = prevSQ:-1:1
        q = Q(qc,1);%struct number
        if Qactive(qc,1) == 1%if this sturct is not marked inactive
            found = 1;
            break;
        end
    end
    if found == 0
        Q = [];
        Qactive = [];
        break;
    end
    e = S{q}.f1;%starting edge
    pv = e(1,1);%previous and current vertices
    cv = e(1,2);
    plot([X(pv,1) X(cv,1)], [X(pv,2) X(cv,2)],'-g', '
       linewidth', 2); hold on;
    if size(PVs,1) == 0
```

```
361         PVs = [[pv q];[cv q]];
362         stq = q;%starting q of this connected commponent of the graph
363     else
364         PVs = [PVs; [cv q]];%previous vertics, which are left column of
                Sq.f1
365     end
366     %explore the current struct
367     while(1)
368         vs = find(E(cv,:) == 1); %all vertices that are connected to cv
369         %find all vertices, other than pv, which are along boundary (remove non
370         %boundary vertices
371         bv = [];
372         if cv == 1628 || cv == 156
373             here = 1;
374         end
375         for m = 1:size(vs,2)
376             k = vs(1,m);
377             if q == 13 && k == 42 && cv == 1
378                 here = 1;
379             end
380             chk = (El(:,1) == cv & El(:,2) == k) | (El(:,1)
                    == k & El(:,2) == cv); %check if edge (cv,k) or (k,cv)
                    is already visited
381             if sum(chk) > 0 && k ~= pv%the current edge is already
                    visited, so a close boundary is found and mark it
382                 el = El(chk,:);
383                 S{stq}.f0 = q;
384                     InsertLoop(ccount,:) = [stq q];
385                 end
386                 if el(1,3) < q
387                     %this edge was added in the Q and S to be explored
388                     %latter, but it has now come to be an edge of contour
389                     %found from q, so make this Q and S entry inactive so
390                     %that it is not used latter
391                     chkq = Q == el(1,3);
392                     if sum(chkq)>0
393                     Qactive(chkq,1) = 0;
394
395                     %and add this to be a nextly visited edge
396                     bv = [bv; k];
397                     %plot(X(bv,1), X(bv,2), 'dk'); hold on;
398                     plot([X(k,1) X(cv,1)], [X(k,2) X(cv,2)
                            ],'-y', 'linewidth', 2); hold on;
```

```
399                     else
400                         plot([X(k,1) X(cv,1)], [X(k,2) X(cv
                                ,2)],'-w', 'linewidth', 2); hold
                                 on;%ignore already visited
                                edge
401                     end
402                 else
403                     plot([X(k,1) X(cv,1)], [X(k,2) X(cv,2)
                            ],'-k', 'linewidth', 2); hold on;
404                     continue;
405                 end
406             end
407             if k ~= pv              && sum(chk) == 0 && (Ne(cv
                    ,k) == 1 || Ne(k,cv) == 1)
408                 bv = [bv; k];
409                 plot([X(k,1) X(cv,1)], [X(k,2) X(cv,2)],'-
                        y', 'linewidth', 2); hold on;
410             end
411         end
412
413         if size(bv,1) == 0
414             break;
415         elseif size(bv,1) == 1
416             %only 1 option, so go forward tracking boundary
417             k = bv(1,1);
418
419             chkV(cv,k) = 0;
420             chkV(k,cv) = 0;
421
422             S{q}.f1 = [S{q}.f1; [cv k]];%update edge list for Sq
423             S{q}.f2 = [S{q}.f2; k];%update vertices list for Sq
424             El = [El; [cv k q size(S{q}.f1,1)]];
425             pv = cv;
426             cv = k;
427             plot([X(pv,1) X(cv,1)], [X(pv,2) X(cv,2)],'-m',
                    'linewidth', 2); hold on;
428
429             if cv == 2935
430                 here = 1;
431             end
432
433             PVs = [PVs;[cv q]];
```

```
434
435         else%more than one option, generate new structs to explore latter
436             clockwiseids = clockwisetchoice(X([cv;pv;bv],:)
                    );
437             bvclockwise = bv(clockwiseids,1);
438             for l = size(bvclockwise,1):-1:1
439                 k = bvclockwise(l,1);
440
441                 chkV(cv,k) = 0;
442                 chkV(k,cv) = 0;
443
444                 count = count+1;
445                 %create a new struct and set q as a parent of new, and new
446                 %as a child of q
447                 S{count} = struct('f0', q, 'f1', [cv k], '
                        f2', [cv;k], 'f3', []);
448                 El = [El; [cv k count 1]];
449                 S{q}.f3 = [S{q}.f3; count];
450                 Q = [Q; count];
451                 Qactive = [Qactive; 1];
452             end
453             break;
454         end
455     end
456
457     %if prevSQ > qc
458
459         Q1 = Q(1:qc-1,:);
460         Qa1 = Qactive(1:qc-1,:);
461
462         Q2 = Q(prevSQ+1:end,:);
463         Qa2 = Qactive(prevSQ+1:end,:);
464
465         Q = [Q1;Q2];
466         Qactive = [Qa1;Qa2];
467     %end
468
469     if sum(Qactive,1) == 0 || size(Q,1) == 0
470         Q = [];
471         Qactive = [];
472         PVall{ccount} = PVs;
473         if size(InsertLoop,1)<ccount
```

```
474             InsertLoop(ccount,:) = [0  0];
475         end
476
477         ccount = ccount+1;
478     %end
479
480     %if size(Q,1) <= qc || size(Q,1) == 0
481     % Q = [];
482     % Qactive = [];
483
484         %look for another boundary edge, if any
485             fv = 0;
486             bm = (E & chkV) & (Ne == 1);
487             hasB = sum(sum(bm));
488             if hasB > 0
489                 for i = is:nP
490                     for j = js:nP
491                         if i < j && bm(i,j)
492                             s = struct('f0', [], 'f1', [i j
                                ], 'f2', [i;j], 'f3', []);%
                                f0: parent struct, f1: edges, f2: vertices list,
                                f3: children structs
493                             fv = i;%first vertex
494                             cv = j;%current vertex
495                             pv = i;%previous vertex
496
497                             %show
498                             %plot(X(fv,1), X(fv,2), 'og'); hold on;
499                             %plot(X(pv,1), X(pv,2), 'sy'); hold on;
500                             %plot(X(cv,1), X(cv,2), 'db'); hold on;
501                             plot([X(pv,1) X(cv,1)], [X(pv
                                ,2) X(cv,2)],'-c', '
                                linewidth', 2); hold on;
502                             break;
503                         end
504                         chkV(i,j) = 0;
505                         chkV(j,i) = 0;
506                     end
507                     if fv > 0
508                         break;
509                     end
510                 end
```

```
511                     if fv > 0% a new edge is found
512                         count = count+1;
513                         S{count} = s;
514                         El = [El;[i j count 1]];
515                         chkV(i,j) = 0;
516                         chkV(j,i) = 0;
517                         is = i;
518                         js = j;
519                         Q = [Q;count];%queue for structs to explore
520                         Qactive = [Qactive; 1];
521                         PVs = [];
522                     end
523                 end
524             if size(Q,1) == 0
525                 break;
526             end
527         %else
528         % Q = Q(qc+1:end,:);
529         % Qactive = Qactive(qc+1:end,:);
530         end
531 end
532
533 for pv=1:size(PVall,2)
534     %find loops
535     PV1 = PVall{pv};
536     loops{pv} = [];
537     for nid = 1:size(PV1,1)-2
538         node = PV1(nid,1);
539         chk = PV1(nid+1:end,1) == node;
540         if sum(chk) == 1
541             %a loop found
542             loops{pv} = [loops{pv}; [nid find(chk==1)+nid]];
543         end
544     end
545 end
546
547 %for each loop try to find two segment loops
548 bids = {};
549 bcount = 0;
550 for pv=1:size(PVall,2)
551     %find loops
552     lcount = 0; loopids = {};
```

```
553     lentries = loops{pv};
554     insloop = InsertLoop(pv,:);
555     PVs = PVall{pv};
556     for i = 1:size(lentries,1)
557         %try to find a loop starting from one end to another end. so, maximum loops
                will be 2 for this entry
558         lp = lentries(i,:);
559         stq = PVs(lp(1,1),2);
560         enq = PVs(lp(1,2),2);
561         if (insloop(1,1) == stq && insloop(1,2) == enq) ||
               (insloop(1,1) == enq && insloop(1,2) == stq)
562             %inserted loop, call only once
563             [loopids1 flg1] = checkloop01(PVs, S, lp(1,1),
                   lp(1,2));
564
565             if flg1 == 1
566                 if lcount > 0
567                     [idr flgr] = checkSubset(loopids,
                           loopids1);
568                     %flgr: 0 new, so add; -1 loopids1 is a superset of an existing, so
                           replace by loopids1;
569                     %1 loopids1 is a subset of an existing, so no replace
570                     if flgr == 0%new
571                         lcount = lcount+1;
572                         loopids{lcount} = loopids1;
573                         plot(X(loopids1,1),X(loopids1,2),'-m',
                               'linewidth', 2); hold on;
574                     elseif flgr == -1%replace
575                         loopids{idr} = loopids1;
576                         plot(X(loopids1,1),X(loopids1,2),'-m',
                               'linewidth', 2); hold on;
577                     end
578                     else
579                         lcount = lcount+1;
580                         loopids{lcount} = loopids1;
581                         plot(X(loopids1,1),X(loopids1,2),'-g',
                               'linewidth', 2); hold on;
582                     end
583                 end
584             else%call twice
585                 [loopids1 flg1] = checkloop01(PVs, S, lp(1,1),
                       lp(1,2));
586
```

```
587             if flg1 == 1
588                 if lcount > 0
589                     [idr flgr] = checkSubset(loopids,
                            loopids1);
590                 %flgr: 0 new, so add; -1 loopids1 is a superset of an existing, so
                        replace by loopids1;
591                 %1 loopids1 is a subset of an existing, so no replace
592                 if flgr == 0%new
593                     lcount = lcount+1;
594                     loopids{lcount} = loopids1;
595                     plot(X(loopids1,1),X(loopids1,2),'-m',
                            'linewidth', 2); hold on;
596                 elseif flgr == -1%replace
597                     loopids{idr} = loopids1;
598                     plot(X(loopids1,1),X(loopids1,2),'-m',
                            'linewidth', 2); hold on;
599                 end
600                 else
601                     lcount = lcount+1;
602                     loopids{lcount} = loopids1;
603                     plot(X(loopids1,1),X(loopids1,2),'-g',
                            'linewidth', 2); hold on;
604                 end
605             end
606             [loopids2 flg2] = checkloop01(PVs, S, lp(1,2),
                    lp(1,1));
607
608             if flg2 == 1
609                 if lcount > 0
610                     [idr flgr] = checkSubset(loopids,
                            loopids2);
611                 %flgr: 0 new, so add; -1 loopids1 is a superset of an existing, so
                        replace by loopids1;
612                 %1 loopids1 is a subset of an existing, so no replace
613                 if flgr == 0%new
614                     lcount = lcount+1;
615                     loopids{lcount} = loopids2;
616                     plot(X(loopids2,1),X(loopids2,2),'-m',
                            'linewidth', 2); hold on;
617                 elseif flgr == -1%replace
618                     loopids{idr} = loopids2;
619                     plot(X(loopids2,1),X(loopids2,2),'-m',
                            'linewidth', 2); hold on;
```

```
                end
                else
                    lcount = lcount+1;
                    loopids{lcount} = loopids2;
                    plot(X(loopids2,1),X(loopids2,2),'-g',
                        'linewidth', 2); hold on;
                end
            end
        end
    end

    for j = 1:size(loopids,2)
        bcount = bcount + 1;
        bids{bcount} = loopids{j};
    end
end
here = 1;
end
```

B.4 Segmentation

```
clc;clear;
ptCloud=pcread('C:\Users\zy080\Documents\MATLAB\Medicaldata
    \dataset\1177tDataset9.ply');
figure
pcshow(ptCloud); %123200
%modelFull =ptCloud.Location';
modelFull = bsxfun(@minus,modelFull,min(modelFull,[],2));
modelFull = bsxfun(@minus,modelFull,max(modelFull,[],2)/2);
modelFull = modelFull / max(abs(modelFull(:)));
%LocationtO=modelFull';
ColortO=ptCloud.Color;
iindex=find(ColortO(:,3)==0);
LocationtO(iindex,:)=[];
ColortO(iindex,:)=[];
pt=pointCloud(LocationtO);
pt.Color=ColortO;
%pcshow(pt);
%[ColortO1,I1,J1]=unique(ColortO,'rows');
[r2,r3,r4,RtC1tkeep] = Colorf(ColortO);
[rc1,ci,cj]=unique(RtC1tkeep);
```

```
for k=1:length(rc1)
        IJC{k,1} = find(rc1(k,1)==RtC1tkeep(:,1));
end
Par=[];
Ptindex=[];
for p=1:length(rc1)
    if length(IJC{p,1})<10
        Par=[Par;IJC{p,1}];
        Ptindex=[Ptindex;p];
    end
end
l2=linspace(1,12,12);
l2(1,Ptindex)=100;
Ptindextleft=find(l2(:)~=100);
for i=1:length(Ptindextleft)
         flg(i,1)=length(IJC{Ptindextleft(i,1),1});
         Locationti=LocationtO(IJC{Ptindextleft(i,1),1},:);
         Colorti=ColortO(IJC{Ptindextleft(i,1),1},:);
         ptcloudti=pointCloud(Locationti);
         ptcloudti.Color=Colorti;
         PT{i,1}=ptcloudti;
        subplot(1,length(Ptindextleft),i);
        pcshow(ptcloudti);
end
%seperate into two groups
 [m1,n1]=sort(flg); %less to more
 pttn1=PT{n1(end),1};
 pttn2=PT{n1(end-1),1};
 PT{n1(end),1}={};
 PT(cellfun(@isempty,PT))=[];
 PT{end,1}={};
 PT(cellfun(@isempty,PT))=[];
 Locationtn3= LocationtO(Par,:);
 Colortn3= ColortO(Par,:);
 for i=1:length(PT)
     Locationtn3=[PT{i,1}.Location;Locationtn3];
     Colortn3 =[PT{i,1}.Color;Colortn3];
 end
 pttn3=pointCloud(Locationtn3);
 pttn3.Color=Colortn3;
%figure
%subplot(1,3,1)
```

```
%pcshow(pttn1); Location11=0.8*single(pttn1.Color)/256;+0.2*(pttn1.Location);
[nnp1,ip1,jp1]=unique(Location11,'rows');
[nnc1,ic1,jc1]=unique(pttn1.Color,'rows');
[ptn1t01,ptn1t02,ptn1t03,indxtP1] = PositiontSplit(nnp1,
    nnc1,30,5);
for k=1:length(indxtP1)
for j =1:length(indxtP1{k,1})
        IJCtnn1{k,1}{j,1} = find(indxtP1{k,1}(j,1)==jp1(:,1)
            );
end
end
for k=1:length(IJCtnn1)
    IJCtNN1{k,1}=cell2mat( IJCtnn1{k,1});
end
figure
for i=1:length(IJCtNN1)
    subplot(1,length(IJCtNN1),i);
    ptn1t{i,1}=pointCloud(pttn1.Location(IJCtNN1{i,1},:));
    ptn1t{i,1}.Color=pttn1.Color(IJCtNN1{i,1},:);
    [rn1t1{i,1},rn2t1{i,1},rn3t1{i,1}, RtC1tkeept1{i,1}] =
        Colorf(ptn1t{i,1}.Color);
    mergetcri1(i,1)=mean(rn2t1{i,1});
    pcshow(ptn1t{i,1});
end
    [a1,b1]=max(mergetcri1);
    ptn1tInstr=ptn1t{b1,1};
    ptn1t{b1,1}=[];
    ptn1t(cellfun(@isempty,ptn1t))=[];
    orlocation=[];
    orcolor=[];
for i=1:length(ptn1t)
    orlocation=[ptn1t{i,1}.Location;orlocation];
    orcolor=[ptn1t{i,1}.Color;orcolor];
end
ptn1torgan=pointCloud(orlocation);
ptn1torgan.Color=orcolor;
figure
subplot(1,2,1)
pcshow(ptn1torgan);%79368
subplot(1,2,2)
pcshow(ptn1tInstr);%1509
%pttn2
```

```
Location12=0.3*single(pttn2.Color)/256;+0.7*(pttn2.Location
    );
[nnp2,ip2,jp2]=unique(Location12,'rows');
[nnc2,ic2,jc2]=unique(pttn2.Color,'rows');
[ptn2t01,ptn2t02,ptn2t03,indxtP2] = PositiontSplit(nnp2,
    nnc2,50,20);
for k=1:length(indxtP2)
for j =1:length(indxtP2{k,1})
        IJCtnn2{k,1}{j,1} = find(indxtP2{k,1}(j,1)==jp2(:,1)
            );
end
end
for k=1:length(IJCtnn2)
    IJCtNN2{k,1}=cell2mat( IJCtnn2{k,1});
end
figure
for i=1:length(IJCtNN2)
    subplot(1,length(IJCtNN2),i);
    ptn2t{i,1}=pointCloud(pttn2.Location(IJCtNN2{i,1},:));
    ptn2t{i,1}.Color=pttn2.Color(IJCtNN2{i,1},:);
    [rn1t2{i,1},rn2t2{i,1},rn3t2{i,1}, RtC1tkeept2{i,1}] =
        Colorf(ptn2t{i,1}.Color);
    mergetcri2(i,1)=mean(rn2t2{i,1});
    pcshow(ptn2t{i,1});
end
 [a2,b2]=max(mergetcri2);
 ptn2tInstr=ptn2t{b2,1};
 ptn2t{b2,1}=[];
 ptn2t(cellfun(@isempty,ptn2t))=[];
 orlocation=[];
 orcolor=[];
for i=1:length(ptn2t)
    orlocation=[ptn2t{i,1}.Location;orlocation];
    orcolor=[ptn2t{i,1}.Color;orcolor];
end
ptn2torgan=pointCloud(orlocation);
ptn2torgan.Color=orcolor;
figure
subplot(1,2,1)
pcshow(ptn2torgan);%79368
subplot(1,2,2)
pcshow(ptn2tInstr);%1509
```

```
139
140 Organ=[ptn1torgan.Location;ptn2torgan.Location;pttn3.
        Location];
141 pttOrgan=pointCloud(Organ);
142 pttOrgan.Color=[ptn1torgan.Color;ptn2torgan.Color;pttn3.
        Color];
143 pcshow(pttOrgan); %103968
144
145 Instr=[ptn1tInstr.Location;ptn2tInstr.Location];
146 pttInstr=pointCloud(Instr);
147 pttInstr.Color=[ptn1tInstr.Color;ptn2tInstr.Color];
148 pcshow(pttInstr) %19226
149 pcshowpair(pttOrgan,pttInstr);
150
151 [labels, numClusters] = pcsegdist(pttInstr, 0.06);
152 pcshow(pttInstr.Location,labels)
153 colormap(hsv(numClusters))
154 ttlabels=tabulate(labels);
155 [t1,t2]=sort(ttlabels(:,2),'descend'); %t2 is label, t1 is number
156 indextinstr1=find((labels==t2(1,1))|(labels==t2(2,1)));
157 ptinstrument=pttInstr.Location;
158 ptinstrument(indextinstr1,:)=[];
159 ctinstrument=pttInstr.Color;
160 ctinstrument(indextinstr1,:)=[];
161 pttInstr01=pointCloud(pttInstr.Location(indextinstr1,:));
162 pttInstr01.Color=pttInstr.Color(indextinstr1,:);
163 pcshow(pttInstr01);%18004
164
165 pttOrgan01=pointCloud([ptinstrument;pttOrgan.Location]);
166 pttOrgan01.Color=[ctinstrument;pttOrgan.Color];
167 pcshow(pttOrgan01);%105190
168
169
170
171
172
173 %[bidstX2t, EtX2t, NetX2t] =
        findtdelaunaytboundary03tfig1(double(pttInstr01.Location),0.008);
174 [Bid1,Bid2,Bid1t] = boundt(bidstX2t);
175 [Instrtindextret,p0tInstr] = boundt2(pttInstr01,Bid1t);
176 Bidd0=unique([Instrtindextret;cell2mat(Bid1t')]);%96
177 ptitlo=pttInstr01.Location;
```

```
ctitlo=pttInstr01.Color;
ptitlo(Bidd0,:)=[];
ctitlo(Bidd0,:)=[];
PTtinstr=pointCloud(ptitlo);
PTtinstr.Color=ctitlo;
 pcshow(PTtinstr) %18000 flag=1;
  %PtitLo = PTtinstr;
while flag
        if length(bidstX2t)~=2
            flag=1;
        [PtitLo,bidstX2t,Bid3,Bid4] = partitiontrepeat(
            PtitLo,0.004);
        else
            flag=0;
        end
end
ppti1=PtitLo.Location(Bid3,:);
ppti2=PtitLo.Location(Bid4,:);
[Instrtn] = boundt3(ppti1,ppti2,PtitLo.Location);
Ptinstr=pointCloud(PtitLo.Location(Instrtn,:));
Ptinstr.Color=PtitLo.Color(Instrtn,:);
pcshow(Ptinstr); %17855

[~,indextinstr]=ismember(Ptinstr.Location,pttInstr01.
    Location,'rows');
ptt=pttInstr01.Location;
ptt(indextinstr,:)=[];
ctt=pttInstr01.Color;
ctt(indextinstr,:)=[];
Organ=[pttOrgan01.Location;ptt];
pttOrgan=pointCloud(Organ);
pttOrgan.Color=[pttOrgan01.Color;ctt];
pcshow(pttOrgan);%105339

%PTtO=pttOrgan.Location;
    xtO=PTtO(:,1);
    xmtO=mean(xtO);
    indextO1=find(xtO>xmtO);
    indextO2=find(xtO<=xmtO);
    pttO1=pttOrgan.Location(indextO1,:);
    cttO1=pttOrgan.Color(indextO1,:);
    pttO2=pttOrgan.Location(indextO2,:);
```

```
218       cttO2=pttOrgan.Color(indextO2,:);
219       pcshow(pointCloud(pttO1))
220       pcshow(pointCloud(pttO2))
221       pttO1ty=pttO1(:,2);
222 %pttO2ty=pttO2(:,2);
223       GMModeltO1= fitgmdist(pttO1ty,2);
224       MmO1=GMModeltO1.mu;
225       indextO1ty=find(((min(MmO1)+sqrt(GMModeltO1.Sigma
              (:,:,2)))<=pttO1ty)&(pttO1ty<=(max(MmO1)-sqrt(
              GMModeltO1.Sigma(:,:,1)))));
226       pttO1ty1=pttO1(indextO1ty,:);
227       cttO1ty1=cttO1(indextO1ty,:);
228       PTtO1tY1=pointCloud(pttO1ty1);
229       PTtO1tY1.Color=cttO1ty1;
230      pcshow(PTtO1tY1);
231       pttO2ty=pttO2(:,2);
232       GMModeltO2= fitgmdist(pttO2ty,2);
233       MmO2=GMModeltO2.mu;
234       indextO2ty=find(((min(MmO2)+sqrt(GMModeltO2.Sigma
              (:,:,2)))<=pttO2ty)&(pttO2ty<=(max(MmO2)-sqrt(
              GMModeltO2.Sigma(:,:,1)))));
235       pttO2ty1=pttO2(indextO2ty,:);
236       cttO2ty1=cttO2(indextO2ty,:);
237       PTtO2tY1=pointCloud(pttO2ty1);
238       PTtO2tY1.Color=cttO2ty1;
239 %pcshow(PTtO2tY1);
240 %[ptto1ty1,Bid,OU1] = partitiontrepeat02(PTtO1tY1,0.004);
241       [PTtO1tY2,Outliers1,Ftmatrix3] =BoundtPTO1Y1(ptto1ty1,
              Bid);
242       figure;
243       pcshowpair(PTtO1tY2,pointCloud([Outliers1]));
244 %[ptto1ty2,Bidt,OU1t] = partitiontrepeat02(PTtO1tY2,0.004);
245        [PTtO1tY3,Outliers1t,Ftmatrix1t,p0] = BoundtPTO1Y2(
              ptto1ty2,Bidt);
246        figure;
247         pcshowpair(ptto1ty2,pointCloud([Outliers1t;p0;OU1t]))
              ;
248 %[ptto1ty3,Bidt2,OU1t2] = partitiontrepeat02(PTtO1tY3,0.004);
249       [Outliers1t2,Ftmatrix1t2] =BoundtPTO1Y3(ptto1ty3,Bidt2
              );
250       figure;
251       pcshowpair(ptto1ty3,pointCloud(Outliers1t2));
```

```
252       Otf01=[OU1; Outliers1 ;OU1t; Outliers1t ;p0;OU1t2;
              Outliers1t2 ];
253       figure ;
254       pcshowpair(pointCloud(Otf01),PTtO1tY1);
255
256
257 %[ptto2ty1,Bid2,OU2] = partitiontrepeat02(PTtO2tY1,0.005);
258 pt1tInstrLocation=ptto2ty1.Location;
259 pttO2ty=pt1tInstrLocation(:,2);
260 GMModeltO1= fitgmdist(pt1tInstrLocation(:,2),2);
261  MmO1=GMModeltO1.mu;
262 indextO1ty=find(((min(MmO1)+sqrt(GMModeltO1.Sigma(:,:,2)))
      <=pttO2ty)&(pttO2ty<=(max(MmO1)-sqrt(GMModeltO1.Sigma
      (:,:,1)))));
263  pttO2ty1=pt1tInstrLocation(indextO1ty,:);
264  figure
265  subplot(1,2,1)
266  pcshowpair(pointCloud(pttO2ty1),pointCloud(
        pt1tInstrLocation));
267  subplot(1,2,2)
268  pcshow(pointCloud(pttO2ty1));
269
270  %[pttO2ty1t,Bid2t,OU2t] = partitiontrepeat02(pointCloud(pttO2ty1),0.003);
271 figure
272 pcshowpair(pttO2ty1t,pointCloud(OU2t));
273 [Ftmatrix2,Outliers2] = BoundtPTO2Y1(pttO2ty1t,Bid2t);
274 pt11=pt1tInstrLocation;
275 pt11(indextO1ty,:)=[];
276 pttO2ty2=pointCloud(pt11);
277 figure
278 pcshow(pt11);
279 [labels, numClusters] = pcsegdist(pttO2ty2, 0.05);
280 pcshow(pt11,labels)
281 colormap(hsv(numClusters))
282 t=tabulate(labels);
283 [at,bt]=min(t(:,2));
284 outliersttindex=find(labels==bt);
285 outlierst=pt11(outliersttindex,:);
286 Ftmatrix3 =pt11;
287 Ftmatrix3(outliersttindex,:)=[];
288
289 Oft02=[OU2;OU2t;Outliers2;outlierst];
```

```
290 pcshowpair(pointCloud(Oft02),PTtO2tY1);
291
292 Organtoutliers=[Oft02;Otf01];
293 %figure;
294 pcshowpair(pointCloud(Organtoutliers),pttOrgan);
295 [~,indextOo]=ismember(Organtoutliers,pttOrgan.Location,'
      rows');
296 pttInstr=[Ptinstr.Location;pttOrgan.Location(indextOo,:)];
297 pttInstrc=[Ptinstr.Color;pttOrgan.Color(indextOo,:)];
298 Ptcloudtinstr =pointCloud(pttInstr);
299 Ptcloudtinstr.Color=pttInstrc;
300
301 pttOrganf0=pttOrgan.Location;
302 pttOrganc0=pttOrgan.Color;
303 pttOrganf0(indextOo,:)=[];
304 pttOrganc0(indextOo,:)=[];
305
306 Ptcloudtorgan=pointCloud(pttOrganf0);
307 Ptcloudtorgan.Color=pttOrganc0;
308 pcshow(Ptcloudtorgan);
309 %pcshow(pttOrganc0);
310 pttOrganc01=single(pttOrganc0)/255;
311 [rn2,rn3,rn4,R01] = Colorf(pttOrganc01);
312 index1=find((rn4<0.06)&(round(pttOrganc01(:,1))==0));
313 hold on
314 pcshow(Ptcloudtorgan.Location(index1,:))
315 pttInstr=[pttInstr;Ptcloudtorgan.Location(index1,:)];
316 pttInstrc=[pttInstrc;Ptcloudtorgan.Color(index1,:)];
317 Ptcloudtinstr =pointCloud(pttInstr);
318 Ptcloudtinstr.Color=pttInstrc;
319
320 ptorgan=Ptcloudtorgan.Location;
321 ptorgan(index1,:)=[];
322 ctorgan=Ptcloudtorgan.Color;
323 ctorgan(index1,:)=[];
324
325 pttInstr=[pttInstr;Ptcloudtorgan.Location(index1,:)];
326 pttInstrc=[pttInstrc;Ptcloudtorgan.Color(index1,:)];
327 Ptcloudtinstr =pointCloud(pttInstr);
328 Ptcloudtinstr.Color=pttInstrc;
329
330 Ptcloudtorgan=pointCloud(ptorgan);
```

```
Ptcloudtorgan.Color=ctorgan;

%figure
%subplot(1,2,1)
%pcshow(Ptcloudtorgan);
%subplot(1,2,2)
%pcshow(Ptcloudtinstr);

ptCloudOut = pcdenoise(Ptcloudtinstr);
[ptCloudInstr,inlierIndices,outlierIndices] = pcdenoise(
    Ptcloudtinstr);
ptCloudOrgan=pointCloud([Ptcloudtinstr.Location(
    outlierIndices,:);Ptcloudtorgan.Location]);
ptCloudOrgan.Color=[Ptcloudtinstr.Color(outlierIndices,:);
    Ptcloudtorgan.Color];
figure
subplot(1,3,1)
pcshow(ptCloudOrgan);
subplot(1,3,2)
pcshow(ptCloudInstr);
subplot(1,3,3)
pcshow(ptCloud);
```

```
function [r2,r3,r4,RtC1tkeep] = Colorf(ColortO)
    for i=1:size(ColortO,1)
            c01=single(ColortO(i,:));
            c01tmin=min(c01)/256;
            c01tmax=max(c01)/256;
            c01tmedian=median(c01)/256;
            r2(i,:)=c01tmax/c01tmin;
            r3(i,:)=c01tmedian/c01tmin;
            r4(i,:)=ColortO(i,1)-ColortO(i,2);
            RtC1tkeep(i,:)=round(r3(i,:))/round(r2(i,:));
    end
end
```

```
function [pt01,pt02,pt03,indxP] = PositiontSplit(LocationtO
    ,ColortO,Sm,Number)
%Position
    [dpmmP,XRP] = Partitiontsingle(LocationtO,Sm,Number);
    for i=1:length(XRP)
```

```
5               [~,indxP{i,1}]=ismember(XRP{i,1},LocationtO,'
                    rows'); %
                    repeat
6               LP(i,1)=length(indxP{i,1});
7       end
8       if  length(XRP)==2
9               LocationtOP1=LocationtO(indxP{1,1},:);
10              pt01=pointCloud(LocationtOP1);
11              ColortOP1=ColortO(indxP{1,1},:); %60555
12              pt01.Color=ColortOP1;
13              LocationtOP2=LocationtO(indxP{2,1},:);
14              ColortOP2=ColortO(indxP{2,1},:); %62645
15              pt02=pointCloud(LocationtOP2);
16              pt02.Color=ColortOP2;
17              figure
18              subplot(1,2,1)
19              pcshow(pt01);
20              subplot(1,2,2)
21              pcshow(pt02);
22              figure
23              pcshowpair(pt01,pt02);
24              pt03=[];
25      elseif  length(XRP)==3
26              LocationtOP1=LocationtO(indxP{1,1},:);
27              pt01=pointCloud(LocationtOP1);
28              ColortOP1=ColortO(indxP{1,1},:); %60555
29              pt01.Color=ColortOP1;
30              LocationtOP2=LocationtO(indxP{2,1},:);
31              ColortOP2=ColortO(indxP{2,1},:); %62645
32              pt02=pointCloud(LocationtOP2);
33              pt02.Color=ColortOP2;
34              LocationtOP3=LocationtO(indxP{3,1},:);
35              ColortOP3=ColortO(indxP{3,1},:); %62645
36              pt03=pointCloud(LocationtOP3);
37              pt03.Color=ColortOP3;
38              figure
39              subplot(1,3,1)
40              pcshow(pt01);
41              subplot(1,3,2)
42              pcshow(pt02);
43              subplot(1,3,3)
44              pcshow(pt03);
```

```
45              figure
46              pcshow(pt01);
47              hold on
48              pcshow(pt02);
49              hold on
50              pcshow(pt03);
51      else
52              LocationtOP1=LocationtO(indxP{1,1},:);
53              pt01=pointCloud(LocationtOP1);
54              ColortOP1=ColortO(indxP{1,1},:); %60555
55              pt01.Color=ColortOP1;
56              LocationtOP2=LocationtO(indxP{2,1},:);
57              ColortOP2=ColortO(indxP{2,1},:); %62645
58              pt02=pointCloud(LocationtOP2);
59              pt02.Color=ColortOP2;
60              LocationtOP3=LocationtO(indxP{3,1},:);
61              ColortOP3=ColortO(indxP{3,1},:); %62645
62              pt03=pointCloud(LocationtOP3);
63              pt03.Color=ColortOP3;
64              LocationtOP4=LocationtO(indxP{4,1},:);
65              ColortOP4=ColortO(indxP{4,1},:); %62645
66              pt04=pointCloud(LocationtOP4);
67              pt04.Color=ColortOP4;
68              figure
69              subplot(1,4,1)
70              pcshow(pt01);
71              subplot(1,4,2)
72              pcshow(pt02);
73              subplot(1,4,3)
74              pcshow(pt03);
75              subplot(1,4,4)
76              pcshow(pt04);
77              figure
78              pcshow(pt01);
79              hold on
80              pcshow(pt02);
81              hold on
82              pcshow(pt03);
83              hold on
84              pcshow(pt04);
85
86      end
```

```
87
88 end
```

```
1  function [dpmm,XR] = Partitiontsingle(X1,sm,iteration)
2      t=size(X1,2);
3      X=X1(:,1:t);
4      X2=X1(:,1:t);
5      [n,d] = size(X);
6      a1=diag(cov(X2));
7      ss = sqrt((median(a1))/2);
8      [niw1] = Gausstinititp(d,X2,sm,ss); %the initial guass
           information for the fixed model
9      [niw2] = Gausstinititp(d,X2,sm,ss); %the initial guass
           information for the moving model
10     [niw] = Gauss01(d,ss,sm); %the intial guass for unknown cluster
11     [dpmm, dpmmtposterior, dpmmttime] = DPMMtgauss01(X,
           niw1,niw2,niw,iteration);
12     [XR] = PartitontSingle01(dpmm,X1);
13 end
```

```
1  function [ Bid1,Bid2,A] = boundt(bidstX2t)
2  %select the longest boundary Bid1 and the second longer boundary Bid2
3  %A is the left boundary points
4   for i=1:length(bidstX2t)
5          Dtinstr(i,1)=length(bidstX2t{1,i});
6   end
7    [Di,Dj]=sort(Dtinstr);
8    Bid1=[bidstX2t{1,Dj(end)}];
9    if Di(end-1)>50
10   Bid2=[bidstX2t{1,Dj(end-1)}];
11   else
12   Bid2=[];
13   end
14 bidstX2t{1,Dj(end)}=[];
15 if Di(end-1)>50
16 bidstX2t{1,Dj(end-1)}=[];
17 end
18   A=bidstX2t;
19   A=A(~cellfun('isempty',A));
20 end
```

```
1  function [Instrtindextre,p0] = boundt2(pttInstr,Bidt)
2      pt1tInstrLocation=pttInstr.Location;
```

```
    Instrtindextre=[];
    for i=1:length(Bidt)
        PT1=pt1tInstrLocation(Bidt{1,i},:);
        [pt1txyz] =Ffeature(PT1);
        Instr1tindex= find((pt1txyz.xm<=pt1tInstrLocation
            (:,1))&(pt1tInstrLocation(:,1)<=pt1txyz.xM)&(
            pt1txyz.ym<=pt1tInstrLocation(:,2))&(
            pt1tInstrLocation(:,2)<pt1txyz.yM));
        Instrtindextre=[ Instr1tindex;Instrtindextre];
    end
    p0=pt1tInstrLocation;
    p0(Instrtindextre,:)=[];
end
```

```
function [PtitLo,bidstX3t,Bid3,Bid4] = partitiontrepeat(
    PTtinstr,s)

[bidstX3t, EtX2t, NetX2t] = findtdelaunaytboundary03tfig1(
    double(PTtinstr.Location),s);
[Bid3,Bid4,Bidt] = boundt(bidstX3t);
[Instrtindextre] = boundt2(PTtinstr,Bidt);
Biddt=unique([Instrtindextre;cell2mat(Bidt')]);
ptitlo=PTtinstr.Location;
ctitlo=PTtinstr.Color;
ptitlo(Biddt,:)=[];
ctitlo(Biddt,:)=[];
PtitLo=pointCloud(ptitlo);
PtitLo.Color=ctitlo;
figure;
pcshow(PtitLo)
end
```

```
function [ptto1ty1,Bid,OU,bidstX] = partitiontrepeat02(
    PTtO1tY1t2,s)
        PTtO1=PTtO1tY1t2;
        OU=[];
        flag=1;
        while(flag)
            [ptto1ty1,Outlier0,Bid3,Bid4,bidstX] =
                RepeattP2(PTtO1tY1t2,s);%8804
            OU=[OU;Outlier0];
            if (length(bidstX)~=2)&&(length(bidstX)~=1)
```

```
                    flag=1;
                    PTtO1tY1t2=ptto1ty1;
                else
                    flag=0;
                end
            end

            if length(Bid3)<=length(Bid4)
                Bid={Bid3};
            else
                Bid={Bid4};
            end
end
```

```
function [PTtO1tY2,Outliers1,Ftmatrix]=BoundtPTO1Y1(
    ptto1ty1,Bid)
        pto1y1tInstrLocation=ptto1ty1.Location;
        for i=1:length(Bid)
            PT1=pto1y1tInstrLocation(Bid{1,i},:);  %
                boundary data
        end
        [Instrtindextre,p0] = boundt2(ptto1ty1,Bid);
        re=[];
        PT2=[pto1y1tInstrLocation(Instrtindextre,:);re];
        [ normals, curvature ] = findPointNormals(PT1, 3);
        contour(normals);
        disp('Please select the key points for normals !');
        prompt = 'The key point will be %d: ';
        ra = input(prompt);
        disp('Please select the key boundary !');
        prompt = 'The beta will be %d: ';
        beta= input(prompt);
        if beta==1
        pt2=PT1(1:ra,:);%contour normals
        else
        pt2=PT1(ra:end,:);
        end
    %pt2=PT1(1:ra,:);pcshowpair(pointCloud(pt2),pointCloud(PT2))
        Idx= find((PT2(:,1)<=max(pt2(:,1)))&(PT2(:,2)<=max(
            pt2(:,2))));  %narrow the
            partition
        pcshowpair(pointCloud(PT2(Idx,:)),pointCloud(pt2));
        pm=PT2(Idx,:);
```

```
26          pm1=PT2;
27          pm1(Idx,:)=[];  %1040 keep Organ 1
28          Pm=pointCloud(pm);
29      %cm=CT2(Idx,:);
30      %Pm.Color=cm;
31          [a01,a02]=max(pt2(:,1));
32          [b01,b02]=min(pt2(:,1));
33          plot(pm(:,1),pm(:,2),'b.');
34          hold on
35          plot(pt2(:,1),pt2(:,2),'r.');
36          hold on
37          plot(pt2(a02,1),pt2(a02,2),'yo');
38          hold on
39          plot(pt2(b02,1),pt2(b02,2),'ys');
40          disp('Please select the way of fitting !');
41          prompt = 'The note1 value will be %d: ';
42          note1= input(prompt);
43          if note1==1
44          fitobject = fit(pt2(:,1),pt2(:,2),'poly1');
45          p1=fitobject.p1;
46          p2=fitobject.p2;
47          else
48          p1=(pt2(b02,2)-pt2(a02,2))/(pt2(b02,1)-pt2(a02,1));
49          p2= pt2(a02,2)-p1*pt2(a02,1);
50          end
51          Atmatrix=zeros(length(pm),6);
52          Atmatrix(:,1:3)=pm;
53        [~,index2]= ismember(pt2,pm,'rows');
54        Atmatrix(index2,4)=1;
55        Atmatrix(:,5)=p1*Atmatrix(:,1)+p2;
56        hold on
57       plot(Atmatrix(:,1),Atmatrix(:,5),'yo');
58        disp('Please select the range of non-outliers !');
59          prompt = 'The note2 value will be %d: ';
60          note2= input(prompt);
61        if note2==1
62              rr=1;
63        else
64              rr=-1;
65        end
66       Atmatrix(:,6)=rr*sign(Atmatrix(:,2)-Atmatrix(:,5));
67       index=find(Atmatrix(:,6)==-1);
```

```
        hold on
        plot(Atmatrix(index,1),Atmatrix(index,2),'y+');
        Ftmatrix=unique([Atmatrix(index,1:3);pt2;pm1],'rows')
            ; %1431
        [~, index1]=ismember(Ftmatrix,PT2,'rows');
        Outliers1=PT2;
        Outliers1(index1,:)=[]; %24
        hold on
        plot(Outliers1(:,1),Outliers1(:,2),'ms');
        [~,index1]=ismember(PT2,ptto1ty1.Location,'rows');
        ptl=ptto1ty1.Location;
        ptl(index1,:)=[];
        PTtO1tY2=pointCloud(ptl);
end
```

```
function [PTtO1tY3,Outliers1,Ftmatrix,p0] = BoundtPTO1Y2(
    ptto1ty2,Bidt)
    pto1y1tInstrLocation=ptto1ty2.Location;
     for i=1:length(Bidt)
          PT1=pto1y1tInstrLocation(Bidt{1,i},:);   %boundary
              data
     end
     [Instrtindextre,p0] = boundt2(ptto1ty2,Bidt);
     pcshowpair(pointCloud(p0),ptto1ty2)
     re=[];
     PT2=[pto1y1tInstrLocation(Instrtindextre,:);re];
     [ normals, curvature ] = findPointNormals(PT1, 3);
     contour(normals);
       disp('Please select the key boundary !');
     prompt = 'The beta will be %d: ';
     beta= input(prompt);
     if beta==1
     disp('Please select the key points for normals !');
     prompt = 'The key point will be %d: ';
     ra = input(prompt);
     pt2=PT1(1:ra,:);%contour normals
     elseif beta==0
     disp('Please select the key points for normals !');
     prompt = 'The key point will be %d: ';
     ra = input(prompt);
     pt2=PT1(ra:end,:);
     else
     disp('Please select the key points for normals !');
```

```
27      prompt = 'The ra will be %d: ';
28      ra = input(prompt);
29      prompt = 'The rb will be %d: ';
30      rb = input(prompt);
31      pt2=PT1(ra:rb,:);
32      end
33  %pt2=PT1(1:ra,:);pcshowpair(pointCloud(pt2),pointCloud(PT2))
34      Idx= find((PT2(:,1)<=max(pt2(:,1)))&(PT2(:,2)<=max(pt2
            (:,2)))); %narrow the
            partition
35      pcshowpair(pointCloud(PT2(Idx,:)),pointCloud(pt2));
36      pm=PT2(Idx,:);
37      pm1=PT2;
38      pm1(Idx,:)=[]; %1040 keep Organ 1
39      Pm=pointCloud(pm);
40  %cm=CT2(Idx,:);
41  %Pm.Color=cm;
42      [a01,a02]=max(pt2(:,1));
43      [b01,b02]=min(pt2(:,1));
44      plot(pm(:,1),pm(:,2),'b.');
45      hold on
46      plot(pt2(:,1),pt2(:,2),'r.');
47      hold on
48      plot(pt2(a02,1),pt2(a02,2),'yo');
49      hold on
50      plot(pt2(b02,1),pt2(b02,2),'ys');
51      disp('Please select the way of fitting !');
52      prompt = 'The note1 value will be %d: ';
53      note1= input(prompt);
54      if note1==1
55      fitobject = fit(pt2(:,1),pt2(:,2),'poly1');
56      p1=fitobject.p1;
57      p2=fitobject.p2;
58      else
59      p1=(pt2(b02,2)-pt2(a02,2))/(pt2(b02,1)-pt2(a02,1));
60      p2= pt2(a02,2)-p1*pt2(a02,1);
61      end
62      Atmatrix=zeros(length(pm),6);
63      Atmatrix(:,1:3)=pm;
64     [~,index2]= ismember(pt2,pm,'rows');
65     Atmatrix(index2,4)=1;
66     Atmatrix(:,5)=p1*Atmatrix(:,1)+p2;
```

```
    hold on
   plot(Atmatrix(:,1),Atmatrix(:,5),'yo');
    disp('Please select the range of non-outliers !');
     prompt = 'The note2 value will be %d: ';
     note2= input(prompt);
    if note2==1
        rr=1;
    else
        rr=-1;
    end
   Atmatrix(:,6)=rr*sign(Atmatrix(:,2)-Atmatrix(:,5));
   index=find(Atmatrix(:,6)==-1);
   hold on
   plot(Atmatrix(index,1),Atmatrix(index,2),'y+');
   Ftmatrix=unique([Atmatrix(index,1:3);pt2],'rows'); %1431
   [~, index1]=ismember(Ftmatrix,Pm.Location,'rows');
   Outliers1=Pm.Location;
   Outliers1(index1,:)=[]; %24
   hold on
   plot(Outliers1(:,1),Outliers1(:,2),'ms');
%[,index1]=ismember(Outliers1,PT2,'rows');
%ptl=ptto1ty2.Location;
%ptl(index1,:)=[];
 PTtO1tY3=pointCloud(pm1);
end
```

```
function [Outliersf,Ftmatrixf]=BoundtPTO1Y3(ptto1ty3,Bidt2)
    pto1y1tInstrLocation=ptto1ty3.Location;
    for i=1:length(Bidt2)
        PT1=pto1y1tInstrLocation(Bidt2{1,i},:);   %boundary data
    end
    [Instrtindextre,p0] = boundt2(ptto1ty3,Bidt2);
    pcshowpair(pointCloud(p0),ptto1ty3)
    re=[];
    PT2=[pto1y1tInstrLocation(Instrtindextre,:);re];
    [ normals, curvature ] = findPointNormals(PT1, 3);
    contour(normals);
    disp('Please select the key boundary !');
    prompt = 'The beta will be %d: ';
    beta= input(prompt);
    if beta==1
```

```
        disp('Please select the key points for normals !');
        prompt = 'The key point will be %d: ';
        ra = input(prompt);
        pt2=PT1(1:ra,:);% contour normals
    elseif beta==0
        disp('Please select the key points for normals !');
        prompt = 'The key point will be %d: ';
        ra = input(prompt);
        pt2=PT1(ra:end,:);
    else
        disp('Please select the key points for normals !');
        prompt = 'The ra will be %d: ';
        ra = input(prompt);
        prompt = 'The rb will be %d: ';
        rb = input(prompt);
        pt2=PT1(ra:rb,:);
    end
    % pt2=PT1(1:ra,:);pcshowpair(pointCloud(pt2),pointCloud(PT2))
    Idx= find((PT2(:,1)<=max(pt2(:,1)))&(PT2(:,2)<=max(pt2
        (:,2)))); % narrow the
        partition
    pcshowpair(pointCloud(PT2(Idx,:)),pointCloud(pt2));
    pm=PT2(Idx,:);
    pm1=PT2;
    pm1(Idx,:)=[]; %1040 keep Organ 1
    Pm=pointCloud(pm);
    % cm=CT2(Idx,:);
    % Pm.Color=cm;
    [a01,a02]=max(pt2(:,1));
    [b01,b02]=min(pt2(:,1));
    plot(pm(:,1),pm(:,2),'b.');
    hold on
    plot(pt2(:,1),pt2(:,2),'r.');
    hold on
    plot(pt2(a02,1),pt2(a02,2),'yo');
    hold on
    plot(pt2(b02,1),pt2(b02,2),'ys');
    disp('Please select the way of fitting !');
    prompt = 'The note1 value will be %d: ';
    note1= input(prompt);
    if note1==1
        fitobject = fit(pt2(:,1),pt2(:,2),'poly1');
```

```
56        p1=fitobject.p1;
57        p2=fitobject.p2;
58    else
59        p1=(pt2(b02,2)-pt2(a02,2))/(pt2(b02,1)-pt2(a02,1));
60        p2= pt2(a02,2)-p1*pt2(a02,1);
61    end
62    Atmatrix=zeros(length(pm),6);
63    Atmatrix(:,1:3)=pm;
64    [~,index2]= ismember(pt2,pm,'rows');
65    Atmatrix(index2,4)=1;
66    Atmatrix(:,5)=p1*Atmatrix(:,1)+p2;
67    hold on
68    plot(Atmatrix(:,1),Atmatrix(:,5),'yo');
69    disp('Please select the range of non-outliers !');
70    prompt = 'The note2 value will be %d: ';
71    note2= input(prompt);
72    if note2==1
73        rr=1;
74    else
75        rr=-1;
76    end
77    Atmatrix(:,6)=rr*sign(Atmatrix(:,2)-Atmatrix(:,5));
78    index=find(Atmatrix(:,6)==-1);
79    hold on
80    plot(Atmatrix(index,1),Atmatrix(index,2),'y+');
81    Ftmatrix=unique([Atmatrix(index,1:3);pt2],'rows'); %
          1431
82    [~, index1]=ismember(Ftmatrix,Pm.Location,'rows');
83    Outliers1=Pm.Location;
84    Outliers1(index1,:)=[]; %24
85    hold on
86    plot(Outliers1(:,1),Outliers1(:,2),'ms');
87    [bidstX3t, EtX2t, NetX2t] =
          findtdelaunaytboundary03tfig1(double(Outliers1)
          ,0.004);
88    [Instrtindextret,p0t] = boundt2(pointCloud(Outliers1),
          bidstX3t);
89    Outliersf=[p0t;p0];
90    figure;
91    pcshowpair(ptto1ty3,pointCloud(Outliersf));
92    [~,index01]=ismember(Outliersf,pto1y1tInstrLocation,'
          rows');
```

```
    pto1y1tInstrLocation(index01,:)=[];
    Ftmatrixf=pto1y1tInstrLocation;
end
```

```
function [Ftmatrix,Outliers1] = BoundtPTO2Y1(pttO2ty1t,
   Bid2t)
pt112tInstrLocation=pttO2ty1t.Location;
pm=pt112tInstrLocation;
for i=1:length(Bid2t)
     PT112=pt112tInstrLocation(Bid2t{1,i},:);   %boundary data
end
%pcshow(PT112)
[normals, curvature] = findPointNormals(PT112, 4);
contour(normals);
disp('Please select the key boundary !');
    prompt = 'The beta will be %d: ';
    beta= input(prompt);
    if beta==1
    disp('Please select the key points for normals !');
    prompt = 'The key point will be %d: ';
    ra = input(prompt);
    pt2=PT112(1:ra,:);%contour normals
    elseif beta==0
    disp('Please select the key points for normals !');
    prompt = 'The key point will be %d: ';
    ra = input(prompt);
    pt2=PT112(ra:end,:);
    else
    disp('Please select the key points for normals !');
    prompt = 'The ra will be %d: ';
    ra = input(prompt);
    prompt = 'The rb will be %d: ';
    rb = input(prompt);
    pt2=PT112(ra:rb,:);
    end
    figure
pcshowpair(pointCloud(pt2),pttO2ty1t);
figure
    [a01,a02]=max(pt2(:,1));
    [b01,b02]=min(pt2(:,1));
    plot(pm(:,1),pm(:,2),'b.');
    hold on
```

```
38      plot(pt2(:,1),pt2(:,2),'r.');
39      hold on
40      plot(pt2(a02,1),pt2(a02,2),'yo');
41      hold on
42      plot(pt2(b02,1),pt2(b02,2),'ys');
43      disp('Please select the way of fitting !');
44      prompt = 'The note1 value will be %d: ';
45      note1= input(prompt);
46      if note1==1
47      fitobject = fit(pt2(:,1),pt2(:,2),'poly1');
48      p1=fitobject.p1;
49      p2=fitobject.p2;
50      else
51      p1=(pt2(b02,2)-pt2(a02,2))/(pt2(b02,1)-pt2(a02,1));
52      p2= pt2(a02,2)-p1*pt2(a02,1);
53      end
54  Atmatrix=zeros(length(pm),6);
55  Atmatrix(:,1:3)=pm;
56  [~,index2]= ismember(pt2,pm,'rows');
57  Atmatrix(index2,4)=1;
58  Atmatrix(:,5)=p1*Atmatrix(:,1)+p2;
59  hold on
60  plot(Atmatrix(:,1),Atmatrix(:,5),'yo');
61  disp('Please select the range of non-outliers !');
62  prompt = 'The note2 value will be %d: ';
63  note2= input(prompt);
64  if note2==1
65          rr=1;
66  else
67          rr=-1;
68  end
69  Atmatrix(:,6)=rr*sign(Atmatrix(:,2)-Atmatrix(:,5));
70  index=find(Atmatrix(:,6)==-1);
71  hold on
72  plot(Atmatrix(index,1),Atmatrix(index,2),'y+');
73  Ftmatrix=unique([Atmatrix(index,1:3);pt2],'rows'); %1431
74  [~, index1]=ismember(Ftmatrix,pm,'rows');
75  Outliers1=pm;
76  Outliers1(index1,:)=[]; %24
77  figure;
78  pcshowpair(pointCloud(Outliers1),pointCloud(pm));
79  end
```