Robotic Cameraman for Augmented Reality based Broadcast and Demonstration

PhD Thesis

by

Dingtian Yan dyan@essex.ac.uk

Supervisor: Prof. Huosheng Hu

PHD Thesis
10/01/2020
Word Count: 37814

Abstract

In recent years, a number of large enterprises have gradually begun to use various Augmented Reality technologies to prominently improve the audiences' view of their products. Among them, the creation of an immersive virtual interactive scene through the projection has received extensive attention, and this technique refers to projection SAR, which is short for projection spatial augmented reality. However, as the existing projection-SAR systems have immobility and limited working range, they have a huge difficulty to be accepted and used in human daily life. Therefore, this thesis research has proposed a technically feasible optimization scheme so that it can be practically applied to AR broadcasting and demonstrations.

Based on three main techniques required by state-of-art projection SAR applications, this thesis has created a novel mobile projection SAR cameraman for AR broadcasting and demonstration. Firstly, by combining the CNN scene parsing model and multiple contour extractors, the proposed contour extraction pipeline can always detect the optimal contour information in non-HD or blurred images. This algorithm reduces the dependency on high quality visual sensors and solves the problems of low contour extraction accuracy in motion blurred images. Secondly, a plane-based visual mapping algorithm is introduced to solve the difficulties of visual mapping in these low-texture scenarios. Finally, a complete process of designing the projection SAR cameraman robot is introduced. This part has solved three main problems in mobile projection-SAR applications: (i) a new method for marking contour on projection model is proposed to replace the model rendering process. By combining contour features and geometric features, users can identify objects on colourless model easily. (ii) a camera initial pose estimation method is developed based on visual tracking algorithms, which can register the start pose of robot to the whole scene in Unity3D. (iii) a novel data transmission approach is introduced to

establishes a link between external robot and the robot in Unity3D simulation work-space. This makes the robotic cameraman can simulate its trajectory in Unity3D simulation work-space and project correct virtual content.

Our proposed mobile projection SAR system has made outstanding contributions to the academic value and practicality of the existing projection SAR technique. It firstly solves the problem of limited working range. When the system is running in a large indoor scene, it can follow the user and project dynamic interactive virtual content automatically instead of increasing the number of visual sensors. Then, it creates a more immersive experience for audience since it supports the user has more body gestures and richer virtual-real interactive plays. Lastly, a mobile system does not require up-front frameworks and cheaper and has provided the public an innovative choice for indoor broadcasting and exhibitions.

Acknowledgements

First of all, I would like to extend my sincere gratitude to my supervisor, Professor Huosheng Hu, for his very important advice and useful suggestions on my thesis. In the preparation of the thesis, he has spent lots of time reading each draft and provided me with valuable advice. Without his patient and expert guidance, it is impossible to finish this thesis.

Second, I would like to express my gratitude to my board members, Dr Michael Gardner and Dr Nikolaos Thomos, who has offered my great guidance and valuable suggestions in my academic research. I am deeply grateful for their help in completing this thesis.

I am also deeply appreciated for all the help from the Robin Dowling and Ian Dukes. They gave me great advice and help me building the robotic system.

Contents

1 Introduction **25**

1.1 Background 25

1.2 Research Motivation 29

1.3 Research Problems 30

1.4 Research Methodology 32

1.4.1 System Structure 32

1.4.2 Visual Sensor 33

1.4.3 Scene Reconstruction 34

1.4.4 Sensor Calibration and Pose Initialisation 37

1.4.5 Visual Tracking and Interactive Plays 37

1.4.6 Experimental Applications 38

1.5 Research Contributions 39

2 Literature Review **41**

2.1 **Current AR Broadcasting applications** 41

2.1.1 Monitor-based Display 41

2.1.2 Head-mounted Display 42

2.1.3 Projection-based SAR systems 44

2.2 A review of Sensors 44

2.2.1 Infrared Light Sensor 44

2.2.2 Hybrid Sensors 46

2.3 Sensor Calibration 48
2.3.1 calibration parameters 48
2.3.2 Structured-light projection algorithm 49
2.4 Image Features 51
2.4.1 ORB Extractor and BRIEF Descriptor 53
2.4.2 LSD, Hough and Contour Extractors 53
2.4.3 3D depth Image and Planar data 54
2.5 RGB-D Visual Scene Mapping Approaches 55
2.5.1 Sparse-SLAM Algorithms 56
2.5.2 Dense/Semi-Dense SLAM Algorithms 56
2.6 CNN in Computer Vision 58
2.7 SLAM with CNN 59
2.7.1 Optimising part of SLAM 59
2.7.2 Introducing semantic information to SLAM 60
2.8 Mathematical Analysis 61
2.8.1 Coordinate transformations 61
2.8.2 P-n-P pose transformation algorithm 62
2.8.3 Bundle Adjustment 64
2.8.4 ICP 66
2.9 Summary 68

3 Plane detection and Completion with CNN Segmented Labels 71

3.1 Introduction 71
3.1.1 Features on plane 72
3.1.2 Object description and deep CNN 77
3.2 The proposed line feature extraction pipeline 78
3.2.1 Design idea and proposed structure 80

3.2.2 Line Extraction Pipeline: Process 1 82

3.2.3 Line Extraction Pipeline: Process 2 85

3.2.4 Line Extraction Pipeline: Process 3 87

3.3 Line feature recovery with occlusion 91

3.3.1 Feature line segmentation and endpoint detection 93

3.3.2 Extending feature line and restoring the occluded edge 95

3.4 Depth completion for 3D contour model reconstruction 102

3.5 summary 106

4 Semantic Visual Mapping with Planes 109

4.1 Introduction 109

4.2 System Structure 110

4.3 Contour-to-Contour 3D Regularisation 111

4.3.1 contour-contour registration 112

4.3.2 Limitations of contour-to-contour Registration algorithm . . . 114

4.4 Optimisation with ref model and Ray-Casting 115

4.4.1 A novel approach for computing matching score 115

4.4.2 Constructing Ref Model 118

4.4.3 Updating Ref Model 120

4.5 Lost Recovery 121

4.5.1 Lost Recovery strategy 122

4.5.2 Lost Recovery Implementation 124

4.6 Global Loop Closure Detection 127

4.7 Summary 130

5 Mobile SAR system design and immersive virtual plays 133

5.1 Introduction 133

5.1.1 System design 134

5.1.2 Visual sensors 136

5.1.3 Ground mobile Pedestal 137

5.2 Projection Scene Registration 138

5.3 Autonomous visual sensor calibration 139

5.3.1 Intelligently choosing calibration-area 140

5.3.2 Sensor calibration with structured-light projection 140

5.4 Plane based camera initial Pose estimation 141

5.5 Autonomous Presenter Following Service 147

5.5.1 The Presenter Following service 148

5.5.2 Real-time robot pose tracking 149

5.5.3 Data transmission between mobile robot and Unity3D 149

5.6 Immerse Virtual Plays 149

5.6.1 Interaction with Virtual Model 150

5.6.2 Interaction with Virtual Scene 150

5.6.3 Mobile Interaction with Virtual Background 152

5.7 Summary 153

6 Practical Experiments in Indoor Broadcasting Scene 155

6.1 The proposed robotic platform 155

6.1.1 Structure of robot 155

6.1.2 Working approach 156

6.2 Simulated indoor demonstration Scene and CNN scene parsing 157

6.2.1 Simulated indoor scene 157

6.2.2 CNN scene parsing 159

6.3 Contour Extraction and Modelling 160

6.4 Plane-based Tracking and Mapping 163

6.5 Sensor Calibration and initial pose estimation 167

6.5.1 Sensor Calibration with Structured-light projection 167
6.5.2 Initial pose estimation 168
6.5.3 Scene Building in Unity3D 169
6.6 Trajectory simulation in Unity3D 170
6.7 Designed Interactive Virtual plays 171
6.7.1 Augmentation in Simulation Workplace 171
6.7.2 Experimental results 172
6.8 Comparison between Mobile and fixed SAR 175
6.8.1 Working Area Extension 175
6.8.2 More Immersive Interactive Plays 175
6.8.3 Bottlenecks of purposed SAR system 175
6.9 Summary 176

7 Conclusion and Future Work 179

7.1 Research so far 179
7.2 Research Contribution 182
7.3 A List of Publications 184
7.4 Future Work 184

References 185

List of Figures

1.1 ”Lightform” SAR Projection in ART Demonstration 26

1.2 “RoomAlive” System. (a) describes the Immersive SAR view provided by RoomAlive. (b)describes the scene scanning technique by RoomAlive. 27

1.3 IR markers for weather broadcasting 28

1.4 A brief introduction of the proposed robotic system. 32

1.5 Workflow of the proposed robotic system 33

1.6 Microsoft ”Kinect” Sensor. (a) Kinect V1 RGB-D Camera. (b)Kinect V2 RGB-D Camera. 34

1.7 (a) General projector for education and demonstration. (b) Recent small Scale projector 34

1.8 A brief introduction of the proposed mapping algorithm in case 1. . . 36

1.9 A brief introduction of the proposed mapping algorithm in case 2. . . 36

2.1 Typical concept of Monitor-Displayed AR 42

2.2 Typical concept of Head-mounted AR 43

2.3 Two recent depth scanning techniques 46

2.4 Radial Distortion 48

2.5 Tangential Distortion 49

2.6 A typical Structure-from-Motion pipeline 55

2.7 Classification of CNN-optimised SLAM algorithm 59

2.8 Illustration of coordinate transformation 62

2.9 The description of P-n-P algorithm 63

2.10 The features detected by the same camera in different pose. The coloured lines are the bundles of light. 64

2.11 A graph model of BA. X_n is the spatial point in the scene; C_i is the camera in different positions. If X_n could be detected by C_i, they are connected. 65

2.12 ICP algorithm calculating the transformation matrix to make Blue point cloud overlaps Red point cloud. 67

2.13 Illustration of transformation target of ICP 67

3.1 Comparison between different Plane extraction algorithms: (a) Vector-based; (b) KdTree; (c) RANSAC; (d)Region Growing. 74

3.2 Analysis of recent popular contour extractors 76

3.3 A structure of DeepLabV3+ model. 78

3.4 (a) is the segmentation result from ADE20K-trained model, and (b) is the result from PASCAL-trained model. 79

3.5 Our label dataset for fine-tune 79

3.6 Segmentation failure. Most of pixels in table region are not labelled as 'table' 79

3.7 A description of applied line feature extractor in purposed system . . 81

3.8 A flow chart of our proposed contour extraction pipeline 81

3.9 Segmentation results from 'DeepLabV3+' Model. (A1), (B1), (C1) and (D1) are raw RGB image. (A2), (B2), (C2) and (D2) are labelled images. The dark green regions in labelled images are the detected table surface. 83

3.10 Line extraction results after the 1st process. Note that the original image of each example is shown in figure 3.9 (A1), (B1), (C1) and (D1) respectively. 84

3.11 Line extraction results after the 2nd process. The blue lines along the table edges are detected line features. Note that the closed green contour outside the table region is the false detection generated by Contour Extractor. They are kept in this step since they do not affect the detection results. 86

3.12 Line extraction and filtering pipeline in Process 3. 87

3.13 Contour extraction result with projected line features generated in Progress 2. (A1), (B1), (C1) and (D1) describes all detected closed contours. (A2), (B2), (C1) and (D2) describes the contour describing the table region. We use the label information to select the right contour: if more than 80 percent of each contour region is labelled as 'table', we use this contour to describe the table. 89

3.14 Small edge feature completion with LSD line segments. The yellow lines in (A2) and (B2) are newly added line features from LSD algorithm. The white contour outside table region in (B1) and (B2) are false detected contour connected to table contour. 90

3.15 Line feature clustering result and optimum choosing results. (A1), (B1), (C1) and (D1) has described the different cluster with various colors. (A2), (B2), (C2) and (D2) are the optimal edge in each cluster. 92

3.16 A flowchart of Feature line segmentation and endpoint detection . . . 94

3.17 Endpoint detection result. 94

3.18 A description of line categories. 96

3.19 A flowchart of restoring the occluded edge. 96

3.20 The description of occlusion in condition 2. 97

3.21 The description of occlusion in condition 3. 99

3.22 The description of occlusion in condition 3. 100

3.23 The description of occlusion in condition 3. 100

3.24 Contour detection on table surface in $Scannet - 0705$ dataset (Dai et al., 2017). In (A), the bottom edge and the bottom right corner of table are occluded by the chair. Experimental result shows our algorithm can predict the occluded corner and restore the occluded contour. Image (B) indicates our algorithm will not retain the edge segment if both of its endpoints are not appearing in current perspective. (C) describes the contour detection result when three edges are visible and one of these edges is occluded. (D) shows the result when one long edge (detected by Hough Line Transform) and one small edge (detected by LSD algorithm) is visible. 101

3.25 Contour detection on TV screen. (A) shows the result when entire TV is visible. (B) shows result when only part of TV is visible. Our algorithm always detect the external contour exactly when the object is semantically detected 101

3.26 Contour detection on Whiteboard in the same scene. Image (A), (B) and (C) show that our algorithm can estimate and recover the occluded corner (bottom left) of whiteboard. (B) and (C) describe our algorithm can reconstruct the edge of whiteboard (bottom) occluded by whiteboard eraser. (D) is the experiment result of another whiteboard in the same scene. 102

3.27 A comparison between our algorithm and LSD algorithm on contour detection of large photos in ICL-NUIM dataset. (A1), (B1) and (C1) are the results from LSD algorithm. (A2), (B2) and (C2) are our results. By combining LSD, Hough Transform and Contour Detector, our algorithm always detect the external edges and makes every detected feature meaningful. 103

3.28 Detection error when segmentation result is bad. (A) and (C) are detection results, (B) and (D) are corresponding detection results respectively. In case (A) and (B), since deep learning model cannot detect the whiteboard, our algorithm cannot find the label region and detect edges. In case (C) and (D), since there are more than 20 percent of plane region has been marked as 'whiteboard' by deep learning model, our algorithm works in this situation 104

3.29 Detection error with high illumination, strong motion blur and object boundary is not clear. Image (A) is the result from Contour Detector, and Image (B) is our result. In this case, Contour Detector, LSD and Hough Transform detector provide too many imprecise features, and it is difficult for algorithm to cluster the features and find optimum. . 105

3.30 The flowchart of the proposed depth completion process. 105

3.31 The recovery 3D contour model with depth completion. 106

3.32 The recovery 3D contour and its extending edge with depth completion. 107

4.1 (a) registration result based on the whole scene model, and derived ICP matching score is 0.00673. (b)registration result based on the contour model, and derived ICP matching score is 0.0001093. Since the score is derived by calculating the distances between matched points in two clouds, our contour-contour registration method outperforms scene-scene match because of lower derived ICP score. . . . 113

4.2 (a)3D contour model matching result, the error = 0.0000032. (b) Scene Model after contour-to-contour registration 113

4.3 a)3D contour model matching result, the error = 0.0000018. (b) Scene Model after contour-to-contour registration 113

4.4 Two 'mismatch' cases. (a) its matching score is 0.000523665. (b) its matching score = 0.00034337. 114

4.5 The flowchart of the proposed contour-contour registration process . 116

4.6 A comparison between raw ICP and ray-cast ICP registrations. (a) describes the raw ICP result and its score = 0.000523665. (b) describes the matching result between ray-cast models, and its score = 0.0000513. It shows that using ray-cast algorithm can improve the tracking performance. 117

4.7 Difference between updating ref model and global model 119

4.8 Ref model harsh updating strategy. 119

4.9 The Ref model updating strategy 120

4.10 (a) is the original input RGB image. (b) describes the ref model in the perspective of this frame. 121

4.11 (a) is the ref model updated in current frame. (b) describes the matching result between the next input image (red) and current ray-casting model (grey). 121

4.12 Different Modes in the proposed Mapping algorithm 123

4.13 Work flow of Lost Mode 123

4.14 (a) is the ref model when lost tracking happens. (b) is the fragment saved in Lost Mode 125

4.15 (a) is the registration result in normal mode, which has been decided as a miss matching case. The blue points are describing the global ref model, and the red points outside the blue model is from the mis-matched frame. (b) is the result after lost recovery. The green points are the raycast global ref model, and the red points are from the lost fragment. 125

4.16 (a) is the ref model. (b) is the fragment saved in Lost Mode 125

4.17 (a) is the registration result in normal mode, and this has decided as a missed matching case. (b) is the registration result with the proposed lost recovery algorithm. 126

4.18 (a) is ref model in current frame. (b) is the long-distanced fragment saved in Lost Mode 126

4.19 (a) is the registration result in normal mode, and this has decided as a missed matching case. The blue points are from ref model and the out-ranged red points are from new inserted frame. (b) is the registration result with purposed lost recovery algorithm. The green points are describing the shared view part of both model. 126

4.20 Global Loop closure detection strategy 128

4.21 Updated Landmark model when Global loop closure occurs. 128

4.22 Global optimisation performance. (a) describes the accumulated error after matching 45 key-frames when the optimisation triggers. (b) describes the registration result based on two-way ray-casting theory 129

4.23 Final mapping result. (a) describes the reconstructed contour model. It is different to figure 4.22(a) since we correct the occluded contour model in some frames. (b) describes the reconstructed scene model. . 129

4.24 The extreme cases our tracking algorithm can handle. (A) The red contour is tracking target. (B) The green contour is tracking target. Since our algorithm works when only a small part of contour is visible, it can reconstruct a large range of scenes based on one plane. 130

4.25 An evaluation of our algorithm. (A) is the calculated error between our estimated trajectory and ground truth. (B) is the trajectory difference. The dotted line is from ground truth, and the color line is our trajectory. 130

4.26 The reconstructed model of first picture. (A) is the reconstructed contour model. (B) is the scanned scene around first picture. 131

4.27 The reconstructed model of second picture. (A) is the reconstructed contour model. (B) is the scanned scene around first picture. 131

4.28 The reconstructed model of two pictures. (A) is the reconstructed contour models of two pictures. (B) is the scanned scene around two pictures. 131

5.1 A description of working process of the proposed Mobile Projection-SAR system 134

5.2 A description of the proposed Projection-SAR robot and visual sensor allocation. 135

5.3 KinectV2 RGB-D Camera and its self-calibration process 137

5.4 Small-scale DLP projector 137

5.5 Pioneer 3 – Ground mobile platform 138

5.6 Structured-light projection. (A) describes the projected pattern. (B) describes the calibration result. 141

5.7 A flowchart describes the feature dictionary algorithm 142

5.8 A flowchart describes the proposed pose initialisation algorithm . . . 142

5.9 Explanation of the proposed algorithm. 144

5.10 New added frame and its nearest frame in dataset. (a) describes the first frame in $harvard_c11/hv_c11_2$, and this could be present as the initial perspective of a robot. (b) is the nearest frame in $harvard_c11/hv_c11_1$ matched by point features and DBoW3. 145

5.11 The edge matching result between two images. (a) describes the first frame in $harvard_c11/hv_c11_2$, two edges are drawn in light green and purple. (b) is the nearest frame in $harvard_c11/hv_c11_1$, the matched edges are drawn in light green and purple, and the unmatched edge is drawn in dark green. 146

5.12 3D point cloud of extend edges. (a) is the extend contour model for newly added image. (b) is the extend contour model for its nearest frame. We only display extend model of two edges in (b) to make experimental result clearer 146

5.13 Initial pose estimation result. (a) describes the combined contour model between initial frame after transformation and landmark contour model. (b) describes the combined scene model between initial frame after transformation and the first frame in dataset. 147

5.14 The presenter following service. In both figures, the blue line is the central perpendicular line and the Green lines describes the bounding box and distance threshold. 148

5.15 The moving data transmission process. 150

5.16 To control the object with body gesture, provided by Microsoft unity SDK and RoomAlive SDK. 151

5.17 To contour the object projected on background. The projected virtual content will be adjust its pose while the user is moving in front of the camera. 151

5.18 The reconstructed scene and the attached virtual content. 153

5.19 The different views when the projector is moving. 153

6.1 The whole mobile robotic system. 156

6.2 The connections between computer and sensors. The camera detects human user and analyses body gestures, then it sends these information to computer. The computer orders the robot to move according to detected information, and the robot will feed back the movement information to the computer at the same time. Finally, the computer projects the interactive virtual content on background through the projector according to the distance moved and the human user's body gesture. 156

6.3 Create an environment of broadcasting. 158

6.4 Transferring Raw RGB data to usable RGB data. (a) describes the raw RGB data in 1920*1080 . (b)describes the usable RGB data in 640*480. The cutting position and resizing scale is based on matching corresponding depth data 158

6.5 substation process of captured raw depth data. (a) is the raw depth image. (b) is the usable depth data after substation. 159

6.6 CNN Scene parsing results. (a) is the cropped RGB image and (b) is the segmented result. In (b), both posts are detected but assigned with different labels. 160

6.7 The re-scaled RGB image of two posts 160

6.8 Contour extraction result on figure 6.7(a). (A) BD extractor, (B) Hough Extractor, (C) LSD extractor and (D) Contour extractor. The red line in each image is the confirmed line segments described the label area, and the green lines are the redundant line segments will be ignored in the further process. 162

6.9 Contour extraction results on figure 6.7(b). (A) BD extractor, (B) Hough Extractor, (C) LSD extractor and (D) Contour extractor. The red line in each image is the confirmed line segments described the label area, and the green lines are the redundant line segments will be ignored in the further process 163

6.10 Final contour extraction results on two posts. 163

6.11 Contour model of two posts. (a) is the contour model of 6.7(a). (b) is the contour model of 6.7(b). 164

6.12 Captured images include different posts. (a) describes the frame includes first post. (b) describes the frame include second post. 164

6.13 Scene Reconstruction Result on first post area. (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene. 165

6.14 Scene Reconstruction Result on second post area (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene. 166

6.15 The whole scene mapping result. (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene. 166

6.16 Final scene model in ".obj" format. 166

6.17 Scene Calibration Results. 167

6.18 Initial frame and its nearest neighbour. (a) is the initial frame captured during calibration process. (b) is its nearest neighbour in database. 168

6.19 The initial frame registration result. (a) describes the matched plane in two models. (b) describes the matching result in the front view. . . 169

6.20 Matching scene model and calibrated Projection SAR system in Unity3D.170

6.21 The pose of important elements. (a) describes the pose of calibrated projection scene. (b) is the pose of reconstructed scene model. 170

6.22 Data transmission files. (a) the number in this file indicates the moving direction. (b) describes the list of generated files. 171

6.23 Attaching the virtual content on scene model. (a) describes the scene with contour. (b) describes attaching the virtual content in contour areas. 172

6.24 The experimental results of Augmentation view in $Pose_1$. (a) describes the simulation work-space in Unity3D, (b) describes the augmented view in experimental scene. 173

6.25 The experimental results of Augmentation view in $Pose_2$. (a) describes the simulation work-space in Unity3D, (b) describes the augmented view in experimental scene. 173

6.26 The scene model in experiment2. (a) describes the simulation workspace in Unity3D, (b) describe the error from inaccurate scene model derived in calibration process. 174

6.27 The experimental results of experiment2. (a) describes the user can move virtual texture by body gesture. (b) describes the user can rotate and move the 3D model by body gesture. In both cases, the virtual background (blue and black background) are not moving. . . . 174

List of Tables

2.1 Comparison between display methods for AR broadcasting 41

2.2 A table describes and compares General Sensors in broadcasting systems 45

1

Introduction

1.1 Background

Broadcasting has been the mainstream of entertainment since the end of the World War II. With rapid development of technology in recent years, the audience‘s requirements for broadcasting program has been greatly increased. Therefore, augmented reality (also named as AR) (Kipper and Rampolla, 2012), as the most advanced technique that enhance the real environment with virtual graphic, has begun to be adopted by recent TV broadcast programs and indoor exhibitions. As an optimised alternative to traditional computer post generation applications, augmented reality not only improves the appearance of real environment with virtual content, but also provides a more vivid human-computer interaction experience by letting the human user interact with augmented virtual 3D model and tell stories in more detail. Moreover, augmented reality has better awareness of the scene compare with pose-production computer graphics and virtual reality. Since it creates the virtual space by attaching the virtual content on real environment directly, the audiences feel less unreal and distorted in real applications.

In the past years, videography applications based on augmented reality linked with broadcasting have been developed in various forms, such as head-mounted augmented reality headset and various computer back-end generated monitor-display augmented reality applications. The former is similar to current technology, which requires every user to wear corresponding head device. In fact, this type of augmented reality application has received the most interests in recent years because it produces the most intuitive human-computer interaction experience. However, it is not suitable for indoor multi-people activities as it requires every user wear the expensive device.

Another augmented reality branch is displaying the virtual content through monitor, which makes user enjoy the virtual view through screens. Monitor-displayed augmented reality technique is popular in recent TV broadcasting industry and it supports the user to contact with virtual content through tracking user's movement with 3rd device. However, since this type has the same displaying approach as traditional post-production computer graphics, it is difficult for viewers to distinguish generated virtual effect between these two techniques. Moreover, displaying virtual view through monitors generate less immersive experience.

Different to augmented reality techniques above, the projection-displayed Spatially Augmented Reality (also named as Projection SAR or SAR) (Bimber and Raskar, 2005) projects virtual graphics onto the surface of environment and enables the augmented view directly to the naked eyes. Since it provides interactive augmented experience without the reliance on external equipment, it is widely accepted in recent indoor entertainment and teaching activities, such as exhibitions, the recovery and interpretation of historical relics, indoor entertainment programs and musical shows, which are enjoyed by a large number of viewer. For example, figure 1.1 shows that "Lightform" has used depth sensor to seamlessly merge virtual content with real-world environment.

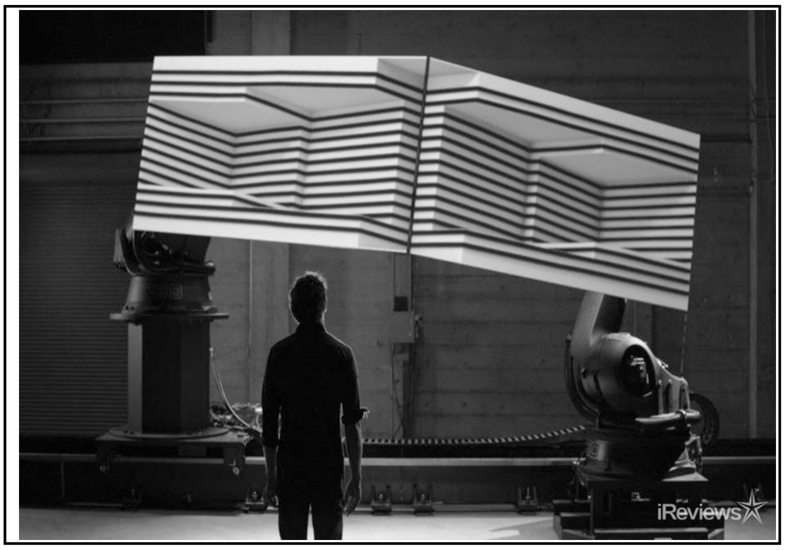
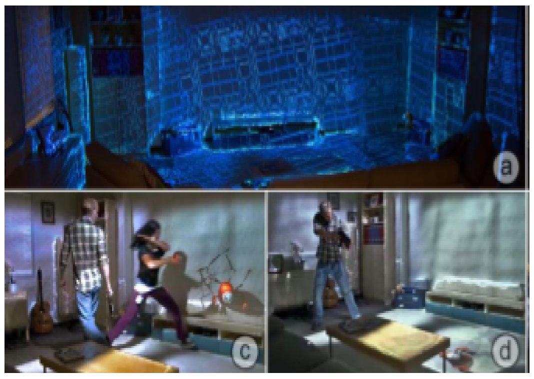


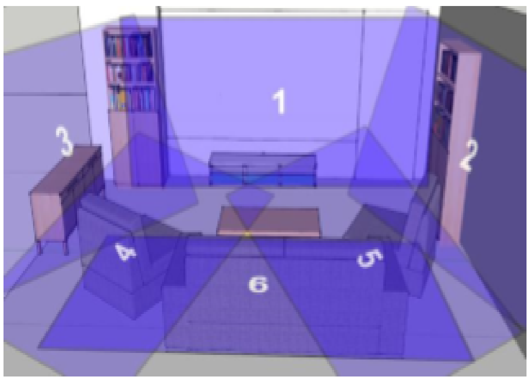
Figure 1.1: "Lightform" SAR Projection in ART Demonstration (STINSON, 2017)

Among three augmented reality branches, projection SAR (projection spatial augmented reality) is the most relevant to modern indoor broadcasting and demonstration applications. It makes the audience on site have the same augmented virtual experience as the audience behind the screen, which is impossible for other branches. However, the technique and applications of existing projection spatial augmented reality is still immature compare with other augmented reality branches. One main limitation is that the current projection-SAR application with single camera-projector has a rigidly limited working range and is unable to imply on a mobile platform. The only solution to expand its working range is increasing the number of camera-projector units, which is more expensive and difficult to build, making current projection SAR systems impractical to marketing.

In the early stage of the projection SAR technology used in broadcasting, TV broadcasting programs utilised AR markers (Bimber and Raskar, 2005) to attach virtual content, such as figure 1.3. Instead of marking the working area with visual 2D marker, others used depth cameras to periodically scan the work-space and re-estimate the projecting parameters. Panasonic has introduced its projection spatial augmented reality system for a luxury suite at a soccer stadium: "Window Augmented Reality Projection"; the dynamic virtual content is projected on a special transparent film attached on windowpane, which permits an augmented view without obscuring. Since projection SAR does not require body-attached equipment, viewers can communicate and enjoy live show with friends together. The most famous application in this field is "RoomAlive" from Microsoft, which could greatly enhance the view of the whole room and interactive playground without physical work. By using projected-structure light technique, RoomAlive could scan the whole room with a projector system with 6 fixed cameras, and then project various designed virtual content over the room appearance, as shown in figure 1.2.



(a)



(b)

Figure 1.2: "RoomAlive" System. (a) describes the Immersive SAR view provided by RoomAlive. (b)describes the scene scanning technique by RoomAlive.
(Jones et al., 2014)

Apart from this, projection SAR (projection spatial augmented reality) is also the best suited to intelligent camera system and robotic cameraman in recent broadcasting and demonstration industries. Since it uses projector to enable an augmented

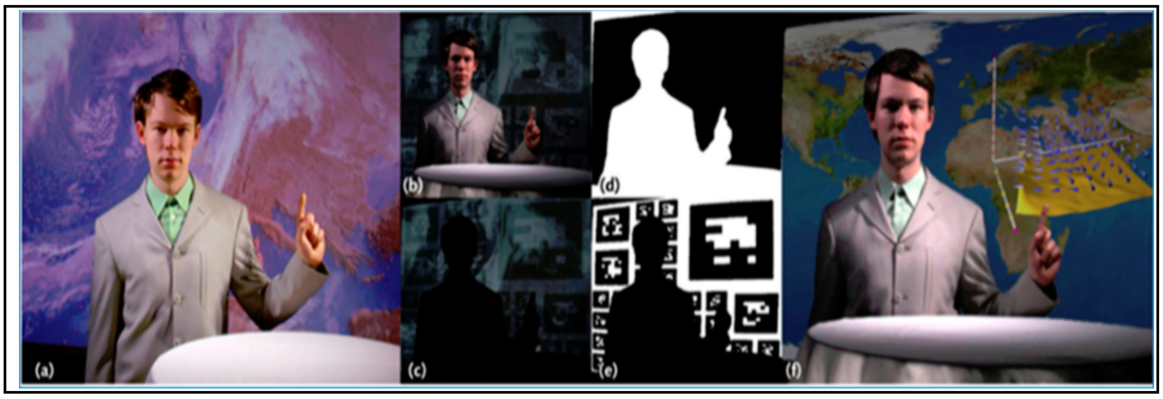


Figure 1.3: IR markers for weather broadcasting
(Grundhöfer et al., 2007)

view on the real scene, its devices can be mounted on intelligent human tracking robot. Recent visual projector has been much smaller, lighter and implementable on a real robot. Moreover, projection SAR tracks the user's movement with a camera, which is similar to cameraman's behavior that tracking the host throughout the whole program. Hence, combining projection SAR with intelligent cameraman robot is mutually beneficial.

On the one hand, projection SAR can make the host's background dynamic and interactive through projection. The host can better describe a story through the interaction of virtual models and virtual backgrounds without post-production. On the other hand, implementing projection SAR on an intelligent mobile robot may be the best solution for "moving" the projector system: state-of-art human tracking service robot includes human detection and limb tracking functions, which are all required by projection SAR; the indoor scene reconstruction and real-time pose tracking included in the advanced visual robot supports projection SAR to estimate its pose simultaneously inside the scene and change the projection content correspondingly.

Based on current status of projection SAR technology and its limitations in applications, this thesis proposes an intelligent mobile SAR system for broadcasting, which consists of a Projection SAR system and a Visual Service robotic pedestal. The projection SAR projects interactive virtual content and 3D models on background, and the robotic pedestal aims at automatically tracking the presenter and feeding back the pose information to the projection SAR system. As an improvement to state-of-art projection-SAR applications, the proposed system could move itself and keep the presenter inside the center of camera. Moreover, the purposed system provides a brand new human-computer interaction AR experience on the existing projection SAR effect.

1.2 Research Motivation

AR technique has been dramatically applied to indoor broadcasting and demonstration industries, namely AR broadcast or AR demonstration. Recent AR applications mainly include three forms based on various displaying methods: (i) AR displayed with wearable glass, (ii) Monitor-displayed AR and (iii) Projection SAR. Among them, the projection SAR is best suited to modern indoor broadcasting and demonstration programs because of two main reasons. One reason is that the presenter could interact with virtual content by projecting virtual content over the real environment instead of computer post generation, which can greatly reduce the errors generated during interacting with virtual model. The 2nd reason is that the augmented view can be directly viewed by the naked eyes and there is no need for external devices. Therefore, it allows more than one audience to enjoy the AR experience at the same time, and also makes the audience on site have the same AR experience as the audiences behind screen.

The concept of applying projection SAR in indoor broadcasting and demonstration has been purposed for years, but has never been realized. One main problem is the existing SAR application is very bulky and has a very limited working range. For example, recent SAR system includes a projector and a depth camera, which can only enhance the virtual scene under one camera‘s view. Some researchers has tried to add a pan-tilt base to the projector, but the working range is still limited. (Wilson et al., 2012) A very representative case is RoomAlive, which gives users a very new idea of using AR to change our life in entertainment and advertisement. However, this system is impractical as it requires too many facilities to enable a whole-room augmented view. (Jones et al., 2014)

Based on this main limitation, we have proposed to combine a projection SAR system with an intelligent robotic cameraman to make existing projection SAR technique have an unlimited working range and a improved interaction AR play. This is novel as there has been relatively little work on improving the mobility of the projection SAR. Instead, recent studies focus more on designing the virtual plays and improving the quality of virtual contents. In general, research in mobile SAR application will not only make great innovations in the technical perspective of existing projection SAR techniques, but also greatly increases its application field and market values.

Combining Projection-SAR with recent robotic Broadcasting system is not simply to mix two concepts together. The best optimisation plan should be created according to their current statues and technical limitations, which could bring the following important benefits:

- Firstly, Projection-SAR is more inclined to demonstration activities or deductive activities. Most of recent broadcasting or exhibition industries use virtual studio to produce augmented experience. Although the rapid development of AR products in recent years, current AR technique is still hard to make users obtain real interactive AR experience as it is produced by computer post production.
- The Projection-SAR enables a group of people enjoying the immersive view together, and also provides the virtual view with naked eyes. Therefore, the audience can experience the similar fun as the viewer behind the screen, which is not available for all the current live broadcasting technology.
- last but not least, an intelligent robotic cameraman will replace human operators in indoor broadcasting and demonstration programs. The preliminary preparations include visual reconstruction of the scene, automatic calibration of sensors, and estimation of the starting pose of the robot. A robot pedestal is more suitable for projection SAR plays as it could provide its motion information with better stability and accuracy. For example, while users are designing the virtual plays, they could plan the path and movement of robotic pedestal so that it can follow the design to make projected virtual graphic correctly match real environment.

1.3 Research Problems

The aim of the proposed system is to enable an interactive and immersive virtual view projected over the indoor environment on a ground mobile visual tracking robot. This system is an automatic intelligent system includes two main processes working in parallel: on the one hand, it tracks human user through visual sensors during operation. When the user in front of the camera moves, it moves itself and keep target appearing in the center of current perspective. On the other hand, system project dynamic virtual graphic and 3D models onto the background. The human user can manipulate the virtual graphic or virtual model by moving its upper limb. Meanwhile, while the user is moving, the projected virtual graphic will change its content to match the background. It has a number of challenge problems to be dealt with in this research, which will be briefly described below:

1. Systematically and completely design of purposed mobile projecting SAR

robot. It includes the selection of sensors, mobile platform and AR designing tools. It also includes the design of data transmission approach between sensors and platforms.

2. Introducing a novel scene mapping algorithm to replace existing scene registration algorithm. In current fixed projection SAR systems, the scene registration process is usually covered by structured-light projection, as a part in sensor calibrating process. Since this approach is designed for fixed application and only generates the 3D projection model in one camera's view, it is no-longer suitable for mobile SAR applications.

   The proposed novel visual mapping algorithm should meet three basic demands. Firstly, this algorithm uses normal RGB-D camera and does not require expensive high-precision sensors. Moreover, since most of broadcasting room and demonstration scenes are empty, low-textured and including empty large planes, like projection screen, monitor and white boards, purposed mapping algorithm can consider one of these planes as a landmark and recover the scene around this plane, and the landmark plane is not required to be completely appeared in every frame. Lastly, the real-time mapping performance is not necessary, purposed mapping algorithm should focus on accuracy and robustness.

3. How to describe the landmark plane robustly and correctly with raw data. In practical, the obtained RGB image and depth data can not maintain high accuracy due to the problem of limited sensor quality and motion blur, and simple edge extraction algorithm can not provide reliable result. Therefore, the proposed plane-based visual mapping algorithm desires an improved contour extraction approach to obtain the optimal edge points and their depth data.

4. Designing an automatic calibration approach and robot initial pose estimation pipeline for mobile projection SAR system. Existing fixed projection SAR application uses structured-light projection for calibration and pose registration. Since this approach is the most efficient calibration algorithm for camera-projector unit, mobile projection SAR can follow this calibration method. But how to register initial pose derived from calibration to the whole reconstructed scene model, and how to build the whole system in Unity3D is the unsolved problem.

5. How to connect external programs with Unity3D. Unity3D is the basic tool for designing the virtual content and interactive plays of projection SAR, but it does not have any port to connect the external mobile robot. Existing projec-

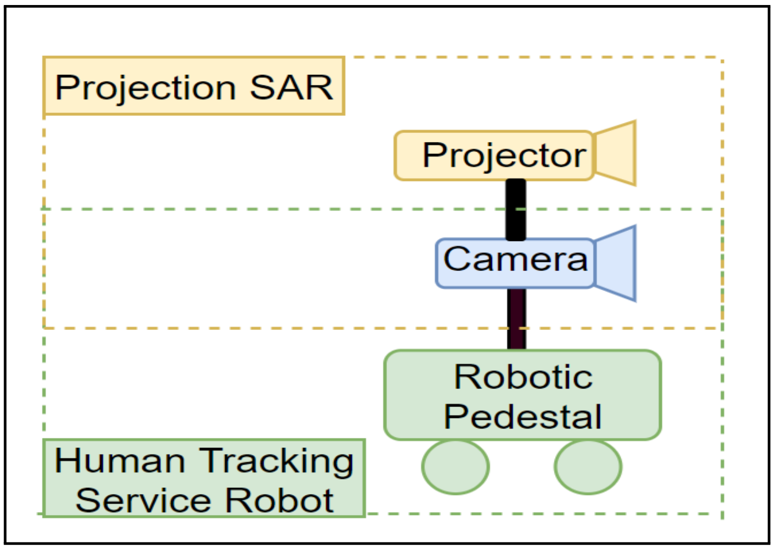


Figure 1.4: A brief introduction of the proposed robotic system.

tion SAR does not need to solve this problem since the camera and projector are fixed. Whereas, mobile projection SAR should solve this problem to make the simulated camera-projector unit in Unity3D simulation work-space resembles the movement of real robot and correctly project corresponding virtual content in changing viewpoints.

1.4 Research Methodology

Based on the research problems mentioned above, this section introduces the structure of the proposed robotic system, sensors and mobile pedestal, as well as the important components.

1.4.1 System Structure

The proposed robotic system is a combination of projection SAR and a robotic pedestal, as shown in figure 1.4. This system has a projector, a RGB-D camera and a ground mobile pedestal. The projector and camera are used in projection SAR, and the same camera and mobile pedestal are used for Human visual tracking service.

The workflow of the proposed system is described in figure 1.5. When a sequence of RGB and depth images is received, the mapping process extracts the features and reconstruct the whole scene. Then, human user can place the robot in the real scenario and start calibration and pose initialisation. At the end of this process,

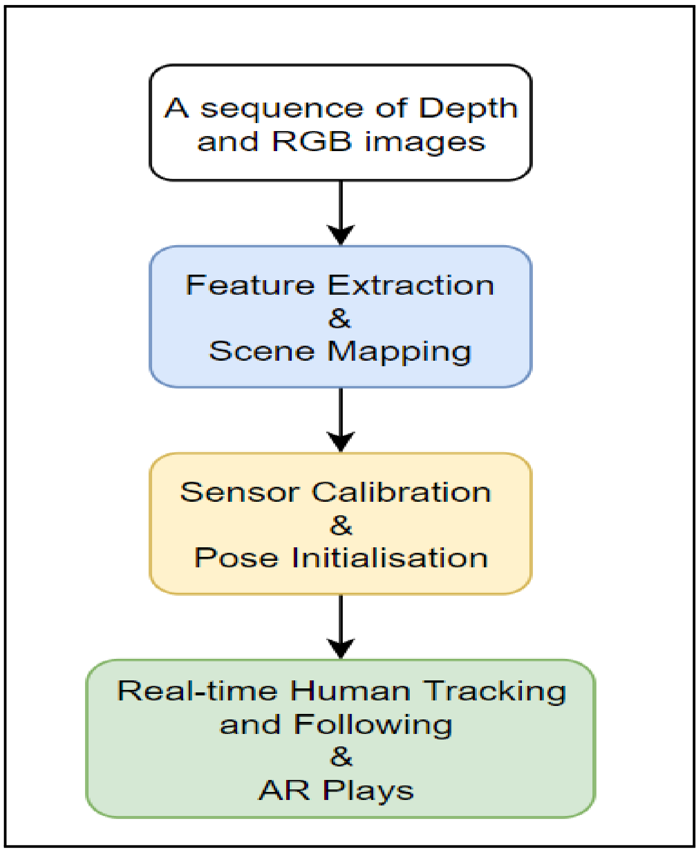

A sequence of Depth and RGB images
Feature Extraction & Scene Mapping
Sensor Calibration & Pose Initialisation
Real-time Human Tracking and Following & AR Plays

Figure 1.5: Workflow of the proposed robotic system

user can build entire SAR system and reconstructed scene in Unity3D simulation work-space. Finally, the system enters running mode and starts tracking human user. The simulated robot in Unity3D work-space simulates the trajectory of real robot and project dynamic virtual content according to user's body gesture and its pose relative to the whole scene.

1.4.2 Visual Sensor

There is a camera and a projector deployed in the proposed robotic system. We use a Kinect RGB-D sensor as the primary camera sensor 1.6. RGB-D camera has played a vital role in existing indoor scene mapping algorithms. Compared with monocular or binocular sensors, a RGB-D sensor is more suitable for indoor featureless environment as it provides depth data directly. Among the state of art RGB-D sensors, Kinect camera has a full detailed toolkit in Unity3D include live image stream and human skeleton tracking, which greatly simplifies the design of the human computer interactive plays in SAR applications.

Recent projectors are small and light to be embedded in a mobile robot, as shown in figure 1.7. The price of these projectors is between 30 pounds to 200 pounds, which is similar to the price of normal projectors.

Figure 1.6: Microsoft "Kinect" Sensor. (a) Kinect V1 RGB-D Camera. (b)Kinect V2 RGB-D Camera.

Figure 1.7: (a) General projector for education and demonstration. (b) Recent small Scale projector

1.4.3 Scene Reconstruction

Since the application scenario is feature-less and empty, we adopt a plane-based semantic mapping algorithm, aiming at reconstructing the scene around a plane by tracking its contour. Compared with existing plane-based visual mapping algorithm, our algorithm relies only one medium or a large plane, such as the surface of table, projection screen and a white empty board, not including the walls and ceiling. Moreover, without prior knowledge, this visual mapping algorithm should be still valid when only part of the plane appears. The proposed scene reconstruction process includes two algorithms: feature extraction and scene mapping loop.

Feature Extraction

To detect the plane, we assume to use CNN scene parsing model for image segmentation and labelling. Although deep learning-based semantic segmentation requires a high-performance GPU and is time-consuming, it is more robust and accurate than feature-based algorithm, especially when the object is partially occluded. In addition, recent deep learning dataset includes most of indoor objects, we only need to fine-tune the model instead of retraining. After plane detection, a novel contour detecting algorithm is required to describe the contour of plane on raw RGB image. It includes two research problems. First, when the RGB image is motion blurred and depth image is not very accurate, it needs to robustly detect the contour and recover the 3D contour model. Second, when the plane is partly occluded because of camera's perspective, the proposed contour detection algorithm can estimate and complete the entire edges. As the proposed contour extraction performance directly affect the final scene reconstruction result, we will use a whole chapter to describe our algorithm and experimental results.

Tracking and Mapping Loop

Since the application scene of projection SAR robot is feature-less, our proposed visual mapping algorithm uses ICP algorithm to densely match the contour models from two frames and estimate the transformation matrix. As ICP algorithm is time-consuming, our mapping algorithm can be used as an optimised back-end optimization algorithm based on existing feature-based SLAM, or a independent off-line visual mapping algorithm when the scene is very empty.

Figure 1.8 introduces the proposed Mapping system in first case. The system is using a RGB-D sensor and the graph based visual SLAM structure, which tracks the inter-frame pose transformation and determines the key frame through images features in the first step. When a key-frame has been detected, the system semantically labels the key frame through the CNN network, and then extracts the edge and contour information of the segmentation part by using the proposed contour extraction algorithm. A data fusion step is proposed to derive contour model. By tracking the contour model, camera pose of each key-frame is further optimised. Finally, the loop closure optimisation triggers when the entire contour is detected.

Figure 1.9 briefly describes the process of the proposed mapping system in second case. This case happens when the visual features is not enough for feature-based visual tracking, and the proposed algorithm works as an independent dense visual mapping system. This system tracks camera frame by frame, and uses global model and ray-casting theory to optimise estimated pose. When the a complete contour of landmark plane is detected, the loop closure optimisation triggers and all previous poses are further optimised. Finally, the scene model is reconstructed and present as a dense point cloud.

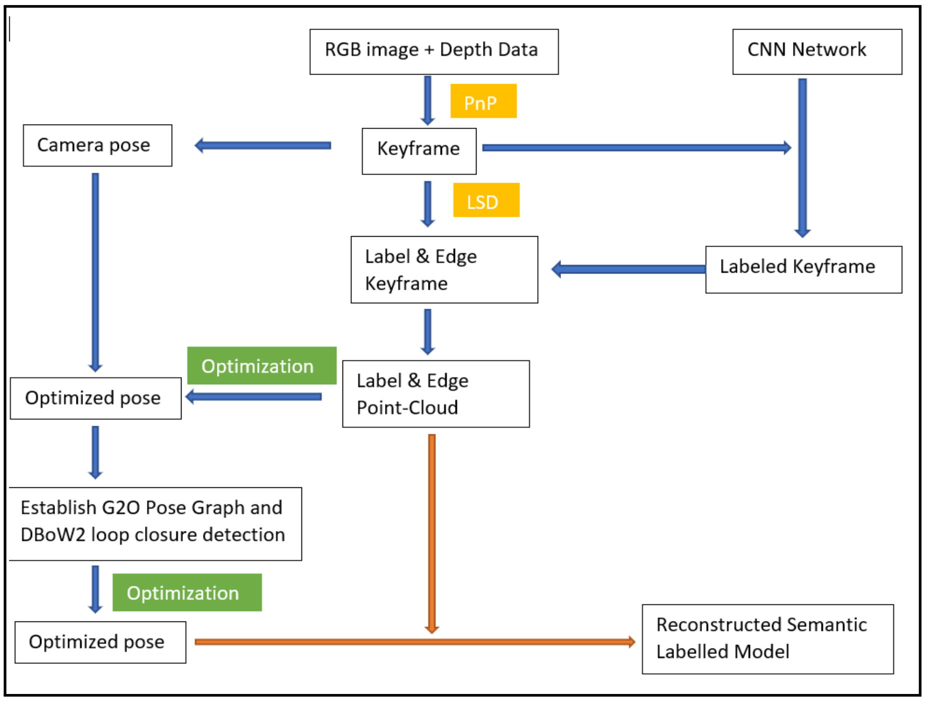


Figure 1.8: A brief introduction of the proposed mapping algorithm in case 1.

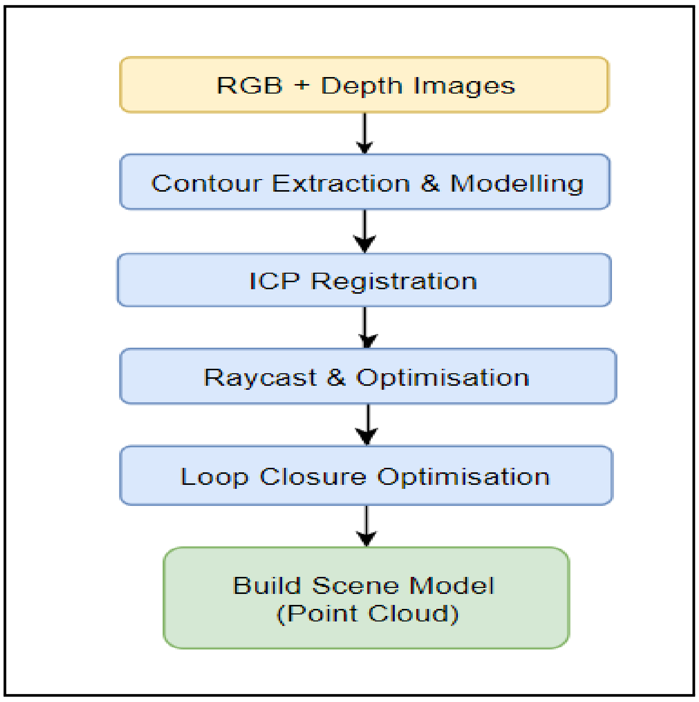

RGB + Depth Images
Contour Extraction & Modelling
ICP Registration
Raycast & Optimisation
Loop Closure Optimisation
Build Scene Model (Point Cloud)

Figure 1.9: A brief introduction of the proposed mapping algorithm in case 2.

1.4.4 Sensor Calibration and Pose Initialisation

After scene mapping process, user can put the robot in the scene and start calibration process that derives the parameters of both visual sensors, RGB image, depth image and the pose of robot relative to current view. By comparing current RGB image and image sequence in mapping step, a rough pose relative to the whole scene is estimated. Then, a more accurate pose is obtained by comparing the contour models in two images. Finally, user can reconstruct entire scene in Unity3D simulation work-space and design AR content based on these estimated parameters.

Sensor Calibration

Similar to existing projection SAR systems, our calibration process is based on structured-light projection. The proposed calibration process is semi-automatic. Firstly, it selects the optimal calibration area in the scene based on the number of visual features, like ORB (Muja and Lowe, 2012) and Line features. And then it requires human user to place the robot in this area and start calibrating. Calibration process derives the camera's pose to calibration scene, parameters of bother sensors, RGB image and depth image.

Pose Initialisation

Since calibration derives the pose relative to the calibration scene instead of the whole environment, a pose initialisation algorithm is required. This algorithm starts with finding the nearest frame of current RGB image in image sequence with DBoW3 library (Wallach, 2006), and then estimates the pose transformation between these two frames. Through transferring the coordinates, we can get the pose of camera relative to the entire scene. At the end of this step, we can build the entire SAR system and scene in Unity3D simulation work-space based on the parameters we calculated, and design the virtual content (Creighton, 2010).

1.4.5 Visual Tracking and Interactive Plays

After calibration, the system enters the running mode and starts following the human user. On the one hand, it detects and follows the human user with RGB-D camera; on the other hand, it projects interactively dynamic virtual content corresponding to user‘s body gesture and current viewpoint.

User Tracking

In live broadcasting and demonstration programs, the host is always in the middle of camera. So, we use human skeleton toolkit provided in Kinect to detect human user and give moving commands to the mobile platform. When the target is moving, the robot should estimate the direction and distance of the movement and keep the target inside its camera's field of view.

AR plays

Due to the fact that most of the SAR applications are fixed and the existing SAR technology development is slow, virtual plays for SAR applications has not made very creative development in recent years. The proposed system supports two types of human-computer interaction experiences.

- Firstly, users can control the projected virtual content by their body gestures, which is similar to the current SAR applications. In this part, the proposed system will use skeleton data to analyse user's movement. The controllable virtual content includes small objects, which allows users to control their rotation, zoom-in and out to let the audience see their three-dimensional information. This can be achieved by several recent open source Unity3D development toolkit.
- Secondly, users can control the virtual background by moving their position inside the scene. The system will define human's relative position inside the room and project the corresponding dynamic virtual background to make the audience feel like the user is moving inside a real scene. This is main innovation for the current SAR technology. The main technical limitation to achieve this is the lack of an interface that links Unity3D and external system. In addition, Unity3D cannot read external dynamic file when it is running. This problem will be figured out in my research.

1.4.6 Experimental Applications

To test the efficiency, an experiment broadcasting scene is designed to test our prototype. This indoor scene is empty and contains several planar objects, such as posts, white board and projection screens. Among them, the empty wall may occupy a very large proportion. The SAR effects designed for live TV scenes are mainly concentrated on the interaction between the broadcaster and the background. During broadcasting, the background behind the presenter will be overlapped with dynamic virtual graphics, and the virtual content will change when the presenter tells another news. In addition, the broadcaster can interact with virtual background and 3D model with simple body gestures, which makes the audience more easily understand.

1.5 Research Contributions

The research contributions of this thesis can be summarised in terms of three aspects, namely academic innovation, application value and Market Value.

- **Academic Innovation** The proposed system provides an innovative solution to the broadcasting combined a robotic pedestal with a projected SAR system. On the one hand, the proposed system can reconstruct a 3D model of Projection area with an innovative Semantic Dense SLAM approach, which could be used in both feature-less environment and rich-texture environment. On the other hand, the system can track human user and estimate its pose during working. Then project interactive and dynamic virtual graphics onto the user‘s background to create a more immersive AR experience.

- **Application Value** By projecting the dynamic virtual graphic alongside the broadcaster, the whole broadcasting scene will turn into a dynamic virtual world. Such design would improve traditional broadcasting scene in three aspects: (i) During news broadcasting, the projected background scene can dynamically display corresponding information and lead to a better understand for audiences; (ii) The projected virtual 3D model can be controlled by broadcaster with designed pose gestures. This will allow the audience to have an intuitive understanding of the described objects. (iii) The proposed system can automatically move and locate itself, keep the broadcaster inside its field-of-view, and project corresponding interactive and dynamic virtual content to the background to enhance the broadcasting scene during broadcasting. (iv) By using Projection SAR technique, the on-site experience will be much improved over existing broadcasting programs, which is the only way to make the on-site AR experience close to experience behind the screen.

- **Market Value** Current AR products on the market are using two external devices: either head-mounted devices or monitor-displayed ones, which make

AR applications difficult to be accepted and used by general public. The proposed system introduces the projection SAR technology into the broadcasting industry and brings a new immersive virtual feeling to the public. It allows audiences to enjoy the virtual content with naked eyes without using external equipment. Moreover, the proposed system is portable and flexible in various types of scene. As a result, a mobile Projection SAR application could be used to wide real-world applications, such as broadcasting studio, indoor demonstration and modern education.

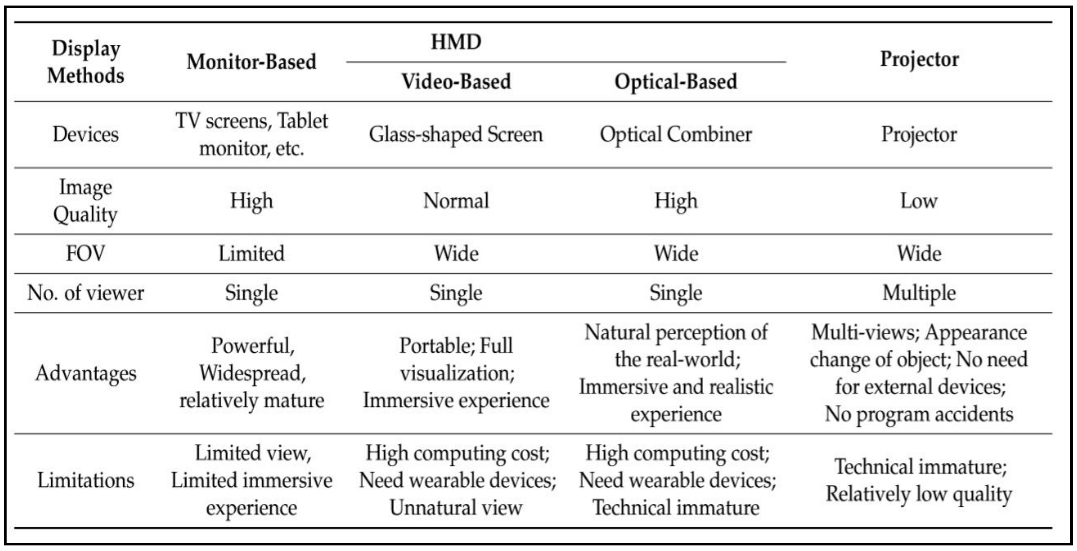
2

Literature Review

2.1 Current AR Broadcasting applications

Recent AR display techniques in broadcasting are still in the developing stage, and are mainly focused on three forms: Monitor-based, HMD (Head-mounted) and Projection-based. A brief comparison between display methods for AR broadcasting is given in table 2.1, and the detailed review will be presented in this section.

Table 2.1: Comparison between display methods for AR broadcasting



Display Methods	Monitor-Based	HMD		Projector
		Video-Based	Optical-Based	
Devices	TV screens, Tablet monitor, etc.	Glass-shaped Screen	Optical Combiner	Projector
Image Quality	High	Normal	High	Low
FOV	Limited	Wide	Wide	Wide
No. of viewer	Single	Single	Single	Multiple
Advantages	Powerful, Widespread, relatively mature	Portable; Full visualization; Immersive experience	Natural perception of the real-world; Immersive and realistic experience	Multi-views; Appearance change of object; No need for external devices; No program accidents
Limitations	Limited view, Limited immersive experience	High computing cost; Need wearable devices; Unnatural view	High computing cost; Need wearable devices; Technical immature	Technical immature; Relatively low quality

2.1.1 Monitor-based Display

Monitor-based display enables augmented view through a monitor or other types of screens, which generally provides the viewers the third-person perspective in front of

TV set, as shown in figure 2.1. The real-world image is firstly captured by camera as inputs. The obtained images are then annotated and merged with computer-generated virtual graphics in a back-stage process. The augmented view is finally displayed through screens. Since this approach requires registering the real-world environment for correctly rendering annotation, the final displayed augmented view is always delayed. AR is firstly used as a virtual annotation on screen-cast of sport competitions, ”Virtual 1st&10 Line” inserted virtual lines to indicate the yardage in American Football broadcasting in 1998. After that, AR annotation has been frequently adopted in basketball, tennis, ice-hockey, etc. (Castle et al., 2011)

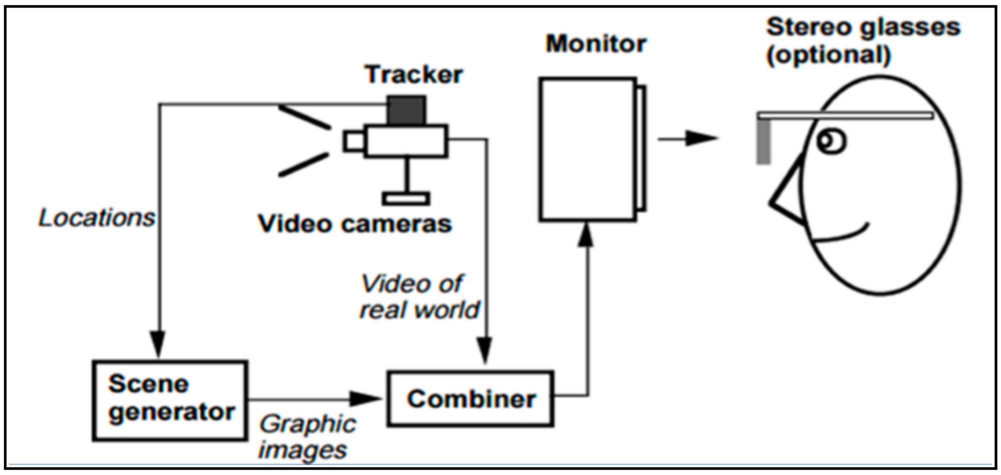


Figure 2.1: Typical concept of Monitor-Displayed AR
(Azuma, 1997)

Recently, this AR technique is widely used to enhance appearance of broadcasting studio. Ericsson introduced their indoor AR systems for visualising the statistical data of sports broadcasting. By augmenting virtual contents at the front of broadcasters, they can interact with 3D virtual models and illustrate sports competition in more detail, e.g., BBC ”Match of the Day” and BT sports. INDE displayed their interactive and touchable AR effect on large screens for education, entertainment, exhibition, etc. As an innovation, some AR systems insert virtual effect on the TV terminal or user side to enable viewers to choose between ordinary image or augmented view, as well as allowed them to design their personal AR effects while watching broadcasts. (Zhang et al., 2016)(Li and Mourikis, 2012)(Hayashi et al., 2010).

2.1.2 Head-mounted Display

Wearable AR glasses are considered to be the most advanced and representative class in recent AR applications(Pavlik, 2015). Unlike recent VR glass that provides a totally virtual environment, AR aims at merging virtual contents with real-world environments. In broadcasting, wearable AR equipment aims at overlapping multiple contents (real or virtual) onto the real world in the audience's view, and its equipment has been designed in two forms: Video See-through (VST) HMD based on video synthesis technology and Optical See-through (OST) HMD based on optical theory, as shown in figure 2.2

Video-based See-Though HMD, VST HMD in short, has the similar working principle as monitor-based AR applications. Since the FOV of human is larger than a normal camera,VST HMD commonly utilised omni-directional cameras. The parallax-free VST HMD system provides a seamless VST Display (Schmalstieg and Höllerer, 2017). Recently, AR Rift has improved the tracking performance with wide FOV cameras on Oculus Rift (Steptoe et al., 2014). Moreover, Optical See-Through HMD (OST HMD) projects virtual contents on special curved optical combiners to merge virtual and real (Cakmakci and Rolland, 2006)(Kress and Starner, 2013). Microsoft has developed its AR headset "Hololens" for the future Sports Broadcasting, which could expand audience view and augmented live 3D virtual models inside the room.

As a comparison, VST HMD has a better compensation for brightness difference between virtual and real and perception delay errors. However, the resolution of its displayed image is limited, and its immersive effect is greatly reduced if the camera is misaligned with user's eyes. OST HMD keeps the resolution of the real world, but it is more difficult for implementation since it needs to face lightness intensity problems. The delay problems are caused by superimposing virtual onto real and resolution matching challenge.

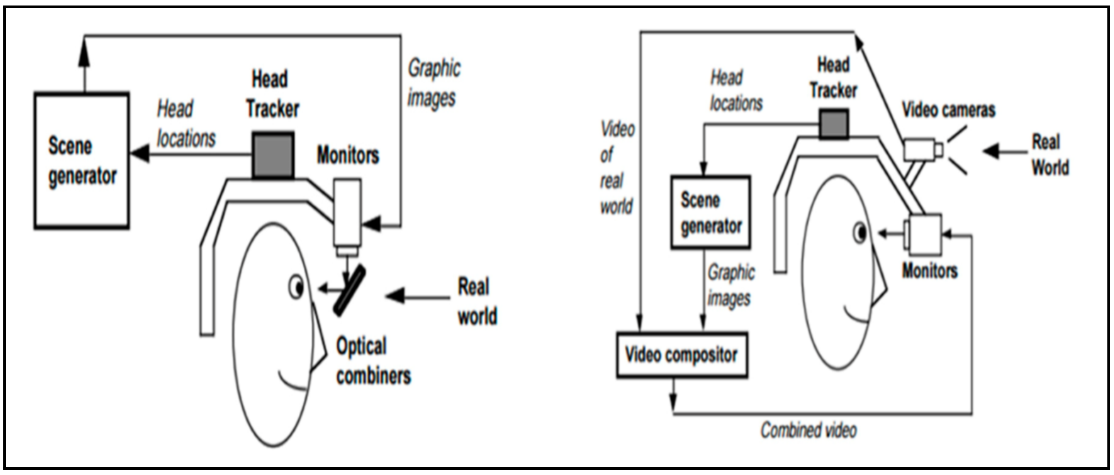


Figure 2.2: Typical concept of Head-mounted AR
(Azuma, 1997)

2.1.3 Projection-based SAR systems

Projector-based Spatial AR technique was developed in AR broadcasting in the past few years(Raskar et al., 1998)(Bimber and Raskar, 2005). In general, projector-based AR systems enable the augmented view directly to the naked eye, and its emergence has eliminated the reliance of external equipment. As an improvement to traditional projectors, SAR allows the projection on the uneven or irregular surface, which creates an interactive, immersive and three-dimensional virtual environment by projecting virtual graphic over objects' surface, 2D or 3D.

Recently, a projector-based AR system was designed to seamlessly merge virtual content with real world. Since it does not require body-attached equipment, viewers can share the experience and communicate while enjoying the AR experience (Kipper and Rampolla, 2012). Panasonic has introduced its concept of projection-based AR system for a luxury suite at a soccer stadium:"Window Augmented Reality Projection". The dynamic virtual content is projected on a special transparent film attached on windowpane, and this special film permits an augmented view without obscuring.

2.2 A review of Sensors

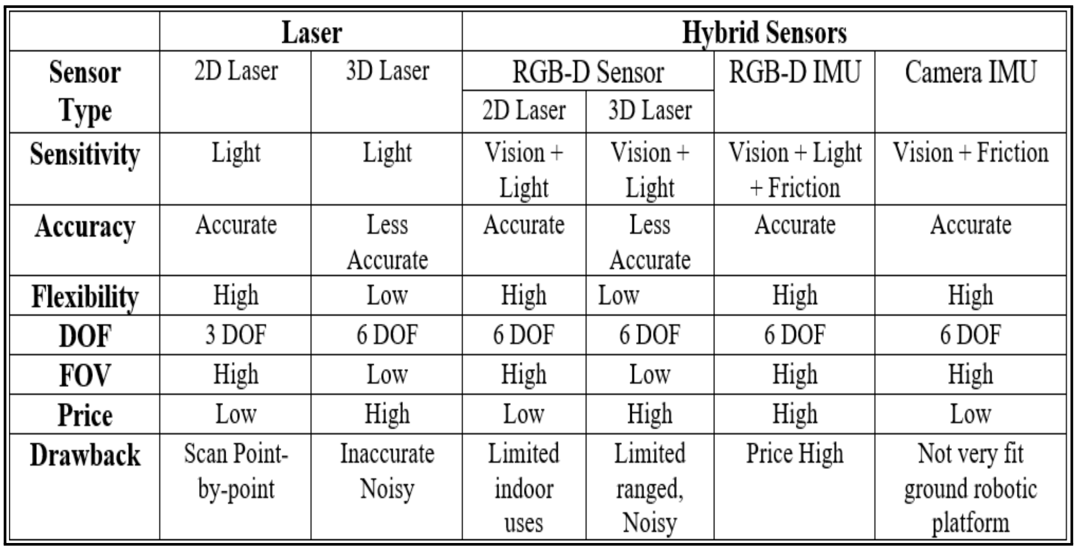
With the recent advancement of sensors in scale and quality, various sensors have been implemented in broadcasting systems. Figure 2.2 shows the most frequently used for indoor SAR, namely the infrared sensors, RGB-D sensor, IMU and their hybrid types. A detailed review on these sensors is given in this section.

2.2.1 Infrared Light Sensor

IR light is a powerful tool for localisation and tracking. Recent IR sensors could be divided into two classes: 2D sensors and 3D sensors. 2D laser sensor is mainly used for measuring distance through TOF theory (Time-of-Flight), which times the round-trip time between emitting infrared light and receive its reflected light. Due to infrared light has a fast transmission speed, its measurement is always accurate and reliable, and not sensitive to external noises. Recently, many indoor intelligent robots or mechanical systems implement this sensor, such as "Pioneer 2" Robot.

Differing from point-by-point scanning, 3D cameras have been introduced as a solid

Table 2.2: A table describes and compares General Sensors in broadcasting systems



	Laser		Hybrid Sensors			
Sensor Type	2D Laser	3D Laser	RGB-D Sensor		RGB-D IMU	Camera IMU
			2D Laser	3D Laser		
Sensitivity	Light	Light	Vision + Light	Vision + Light	Vision + Light + Friction	Vision + Friction
Accuracy	Accurate	Less Accurate	Accurate	Less Accurate	Accurate	Accurate
Flexibility	High	Low	High	Low	High	High
DOF	3 DOF	6 DOF	6 DOF	6 DOF	6 DOF	6 DOF
FOV	High	Low	High	Low	High	High
Price	Low	High	Low	High	High	Low
Drawback	Scan Point-by-point	Inaccurate Noisy	Limited indoor uses	Limited ranged, Noisy	Price High	Not very fit ground robotic platform

competitor, which scans the depth information of the whole FOV at a video frame rate. Recent 3D depth cameras mainly deploy two working principles:

- **Structured-light projection** To obtain depth information, the structure-light projection technique projects light patterns on the object surface by LCD projector or other light source, and then calculates the distance of points by analysing the deformation of projected patterns. Structure light projection is also popular in calibrating intrinsic and extrinsic parameter of camera-projector system (Deglint et al., 2016)(Luo et al., 2014). figure 2.3(a) shows the structured light projection algorithm from Kinect V1.

- **Time of Flight** The TOF based 3D camera projects laser light onto target surface and times the reflection time to measure distances of each point (Kermen et al., 2015). This approach works in a large range with a high accuracy, but it requires higher quality sensors. Currently, its representative applications include Swiss Ranger SR4000/SR4500, and Kinect V2. Figure 2.3(b) shows the Working principle of Time-of-Flying based 3D camera.

Though 3D camera provides a direct depth measurement, it is rigidly limited for indoor applications as infrared light is strongly disturbed by sunlight. In addition, recent consumer-level 3D depth scanners are noisy and low quality, with a working range at around 4 meters to keep depth measurement accurate and reliable.

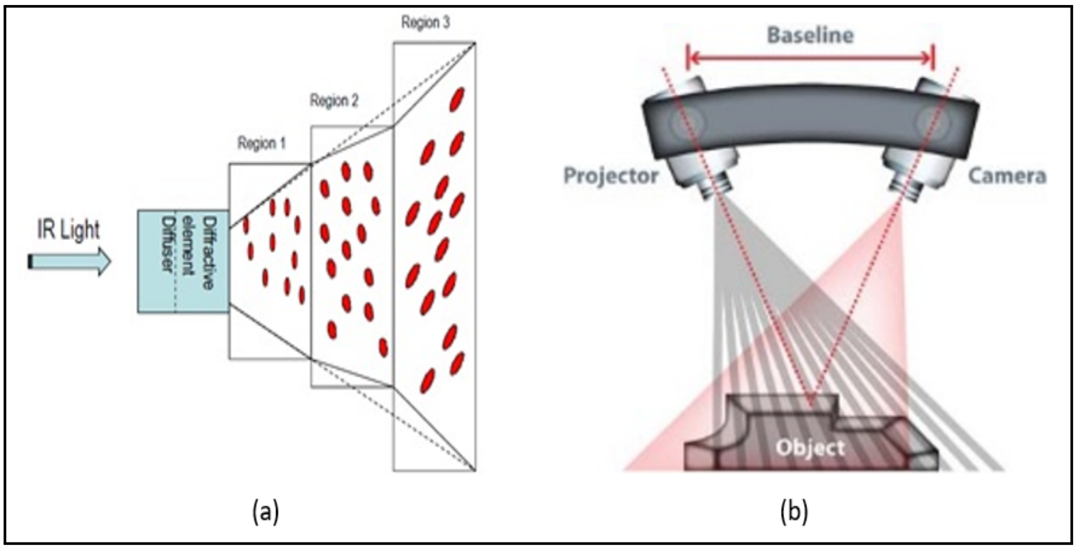


Figure 2.3: Two recent depth scanning techniques
(Sheffield University, 2017)(Perreault, 2016)

2.2.2 Hybrid Sensors

As we know, a pure vision system cannot provide accurate measurement while it is working in a large-scale environment. The current vision-based systems could utilise passive landmarks to calibrate, which is however generally computational expensive and less robust to occlusion problem. As a result, recent auto tracking system frequently combines external sensors to improve tracking performance, such as IMU + Vision sensor for Visual Odometry, and RGB-D camera for indoor SLAM and AR.

The integration of vision and inertial sensor is introduced in early 1990s and its advantages is concluded in (Viéville and Faugeras, 1990) (Corke et al., 2007). The combination could overcome the limitations of both sides, and greatly enhances the overall performance. On the one hand, inertial sensor cannot eliminate accumulation error over time, and loses its efficiency when the sensor moves slowly; on the other hand, vision sensor-based tracking method could track target at a low speed, which causes motion blur while increasing camera‘s moving speed. In addition, land markers recognised by a vision sensor can effectively reduce accumulation errors in inertial sensors. Two methods have been used to fuse vision data and IMU data, namely “Long coupled” and “Tight Coupled”. In “Long coupled”, IMU data is used to optimise feature extraction results, especially when the feature is blurry. In contrast, “Tight Coupled” method puts IMU data and vision data in a statistical filter to estimate target pose. (Kermen et al., 2015)

Similar to inertial-vision sensor, fusing RGB data with laser data greatly enhance tracking performance. Recent laser-camera sensor mainly involves two types of application: One type is similar to common feature-based Visual Odometry: camera position is tracked by sparse visual features from normal RGB images, and the laser sensor is mainly used for measuring the distance to the recognised object. For example, "Vinten" indoor broadcasting robot has implemented a 360-degree 2D laser scanner onto its robotic camera pedestal. Based on triangulation, the robotic camera pedestal could define its position by measuring the distance to three detected objects in the broadcasting studio. Recent AR development platform, Google Tango, combines binocular camera, 2D laser sensor, depth camera and IMU for indoor AR applications. In one of its advising App is called "Measurement", it can measure the distance from the sensor to target, as well as the distance between two spatial points.

The other type is the combination of a 3D laser scanner and a camera, namely RGB-D camera. RGB-D camera captures both RGB image and depth image at the same time and has changed the classic feature-based tracking algorithms. In many cases, depth information is more robust and plays a vital role move than image features. For example, in KinectFusion, while moving Kinect around the scene and extracting all possible data instead of sparse image feature points, the camera pose is continuously tracked at a frame-rate of 30Hz through a coarse-to-fine ICP algorithm for Augmented Reality.(Newcombe et al., 2011a) Similarly, Kintinuous and ElasticFusion has used dense visual tracking approach in a larger scenario.(Whelan et al., 2012) (Whelan et al., 2015) Since dense point-based approach highly relies on powerful hardware, some researchers have introduced semi-dense algorithms, e.g. Semi-dense Visual Odometry has been utilised in mobile AR applications.

Recently, Intel Real Sense camera is popular on robotic systems. It contains two infrared cameras, infrared emitters, RGB camera and IMU four modules (Keselman et al., 2017). As the improvement of technology, its SDK supports hand and finger tracking, facial analysis, voice recognition, background removal, target tracking, augmented reality and 3D scanning. And it is more expensive than other sensors.

Generally, consumer-level RGB-D cameras are sensitive to noise and not very accurate, but frequently used for small AR work-space and academic research. For AR applications in a larger room, existing method is using multiple RGB-D sensors. (Wilson and Benko, 2016)

2.3 Sensor Calibration

Sensor calibration aims at calculating their intrinsic and extrinsic parameters. Intrinsic parameters include focal length, optical centre and skew coefficient, which are the basic parameters of each sensor. Since the working principle of a projector is the opposite of camera, its intrinsic matrix could be illustrated using a similar model.

2.3.1 calibration parameters

The camera intrinsic matrix could be defined as:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where c_x, c_y are the optical centre in pixels relative to image plane; f_x, f_y are the focal length in pixels; S is the skew coefficient, which indicate the deviation from camera lens itself.

The deviation between an ideal point and an image point which is caused by craftsmanship called optical distortion (Simulink, 2017). It includes Radial distortion and tangential distortion. Radial distortion happens when light rays are banded unequally from optical centre to lens edges, and it includes two types, as shown in figure 2.4. Tangential distortion happens while lens and image plane are not parallel to each other. Figure 2.5 shows the basic principle.

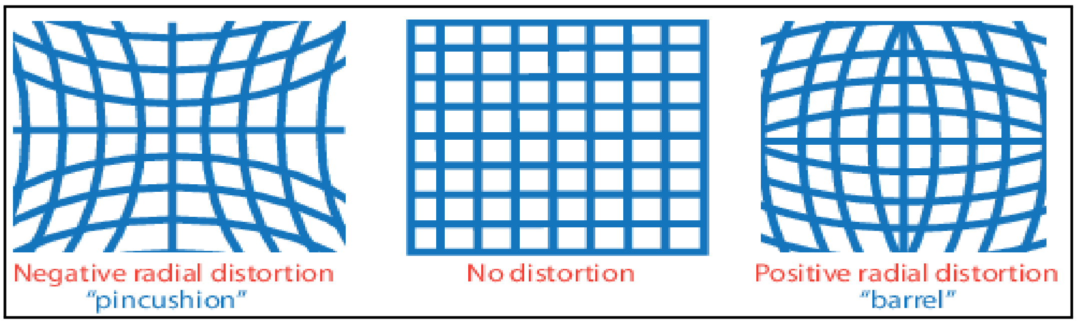


Figure 2.4: Radial Distortion
(Simulink, 2017)

Apart from intrinsic values from sensors, SAR system also requires a precise 3D spatial relationship between individual components. It is included in extrinsic parameter matrix. The extrinsic parameter matrix of an optical sensor could be expressed

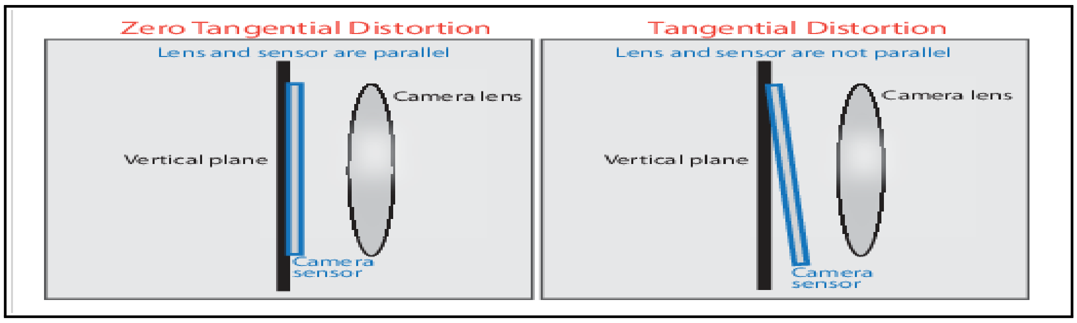


Figure 2.5: Tangential Distortion
(Simulink, 2017)

as a 3 * 3 rotation matrix with a 3 * 1 vector matrix: (Equation 2.2)

$$\begin{bmatrix} R \,|\, T \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,2} & x \\ r_{1,2} & r_{1,2} & r_{1,2} & y \\ r_{1,2} & r_{1,2} & r_{1,2} & z \end{bmatrix} \tag{2.2}$$

This matrix describes how to convert points from the world coordinate system to the camera coordinate system. The translation matrix T describes the position of the space origin in the camera coordinate system. The rotation matrix R describes the direction of the coordinate axes of the world coordinate system relative to the camera coordinate system.

2.3.2 Structured-light projection algorithm

Recently, self-calibration approach has been the mainstream in camera-projector calibration. It has been considered as a powerful replacement to traditional tedious manual calibration method (Martynov et al., 2011). A classic self-calibration approach includes two main steps. Firstly, it projects a pre-encoded 2D pattern, such as ARTag markers and structured light on 2D images or 3D models. Secondly, it uses camera to capture projection images, and decoded the projected patterns to calculate sensor's parameters with bundle adjustment or other algorithms.

Fiola and Mark used self-identify tags for projector calibration with assumption that camera is pre-calibrated (Fiala, 2005). They used three ARTag Patterns instead of 18 Gray code images for simplifying computation process. Similarly, Klose and Stefan projected the numbers of 2D patterns onto a planar surface for calibration, but their algorithm was based on assuming that screen is planar and camera is stationary corresponding to the scene (Klose, 2007). Zhang et al. projected a checker-board pattern on another printed checker-board, and then applied Ray-plane intersection algorithm on each corner to measure camera and projector's 3D position (Zhang et al., 2010a). Alternatively, Wilm, et al. utilised Gaussian RBFs in translating calibration points into projector image coordinates to find optimisation (Wilm et al., 2014). Martynov et al. projected distinguishable points with special color on a checker-board to define parameter, and their calibration was based on in-calibrated camera. However, their approach required users to move the projector frequently in order to provide different projection views (Martynov et al., 2011).

Differing from 2D pattern-based projection, structure light projection based calibration has been widely used in calibration of current room-size SAR systems. Yamazaki and Mochinmaru projected grey encoded structured light patterns on arbitrary surface of target. Their intrinsic and extrinsic parameters of both two devices were estimated by decomposing a radial fundamental matrix into intrinsic and extrinsic parameters established from dense point correspondences between projector and camera images (Yamazaki et al., 2011).

Based on this method, Deglint, et al built a cost function to more accurately estimate in-distorted points during calibration as an improvement(Deglint et al., 2016). Many researchers have used grey code sequence to replace checker-board, which contains richer information in each pattern (Salvi et al., 2004). Since there is no limitation in numbers, e.g. coded or colour points, the points-based method is more flexible. In addition, points-based calibration has similar precision compare with other state-of-art calibration methods (Xiaoyang et al., 2014).

Other researchers have adopted colour information during calibration. Zhang utilised a M-array to define the correspondence between projector and target. The colourful and square encoded patterns were projected on a calibration board, and the captured image was then decoded to obtain the corner feature (Zhang and Zhu, 2009). In order to avoid the pre-calibration of camera, Li defined the parameters of camera and projector at the same time. The correspondence between feature points in camera and project images was calculated with a specific colour-coding scheme (Li et al., 2011a).

To precisely calibrate sensor's parameters, Wilm, et al. used Phase shifting mechanism based on structured light to determine the extrinsic parameters. The basic principle of the phase-shift method is the means of phase shifting profilometry (PSP), and this helps the system find its position in world coordinates. The phase-shifting encode dense scene has smaller number of patterns, comparing with other algorithm (Wilm et al., 2014).

2.4 Image Features

Image features contain the most important messages of each captured images, and the combination of various features tells the information of every real object. Image features include special points, corners, edge, blob, colour and others. Since matching every feature to define camera's pose transformation requires huge computation, and simple image feature cannot tell precise transformation and orientation data, a variety of feature description algorithms have been deployed in recent camera tracking and mapping approaches. Such feature description involves some basic algorithms such as oriented gradient histogram, and modern SIFT, SURF and ORB (Ng and Henikoff, 2003) (Bay et al., 2006) (Rublee et al., 2011)

A good Feature detection algorithm should be able to locate the most significant features accurately in different viewing conditions and at a fast speed. Harris corner detector is a feature detector which locates the corners in every image through calculating and comparing the eigenvalues of Harris matrix between feature points and its neighbours. It is invariant to rotation and scale variance, and is also popular in human face detection or matching works. (Harris et al., 1988) (Derpanis, 2004)

Edge is another salient image feature that is able to effectively describe the real object. Gradient operator is a basic function used to locate potential boundaries of object. To describe the edge features, the orientation and transformation variance of each feature pixel could be described by the direction and magnitude of pixel vector respectively. As an improvement, Canny edge detector firstly smooths each frame with Gaussian convolution, and then uses a 2-D first derivative operation on images to highlight edge area. Finally, only the pixels with top value of each highlight area is kept in the output and is displays as a simple line. All the output lines are evaluated with a threshold in order to eliminate "strong" and "weak" lines (Yitzhaky and Peli, 2003).

FAST (Features from accelerated Segment Test) defines interest points based on corner detection algorithm. Fast detector calculates the grey value of pixels around candidate features point, if the around area has enough number of pixels which has sufficient large difference with candidate, the candidate point is considered as a feature point. FAST detector is very fast, to make it faster, researchers have tested that only 3 of 4 pixels around candidate in every 90-degree angle are enough to locate the feature. The radius of this circle and numbers of comparing pixels are important parameters, and FAST-9 and FAST-12 are most commonly used. (Mair et al., 2010)

Bag-of-Words in image processing is popular feature recognition and clustering technique. A typical BOW algorithm uses k-means clustering method to unsupervised clustering on a large number of extracted features and classifying similar features in various categories. The central of each category is defined as the 'word' of clustered images and the total number of clusters is the size of entire dictionary. Generally, BOW algorithm includes three steps. It firstly extracts representative local stable and distinguishable features from images by using edge-Laplace, Harris-Laplace, hessian-Laplace and others detectors. Secondly the clustering algorithm (e.g. K-Means algorithm) is used to construct the word list (dictionary) from the feature descriptors extracted and the feature descriptors are divided into K clusters. Then similar clusters are fused to ensure the elements in each cluster are similar. The number of clustering categories K is the size of the entire visual dictionary, and it depends on scene complexity. Finally, by extracting and matching features in each frame, an image could be presented as the histogram of code-words. (Zhang et al., 2010b) (Li et al., 2011b)

LBP (Local Binary Pattern) is a texture description method based on binary numbers. Original LBP detector determines the feature pixel through matching its intensity with its surrounding 8 pixels, and each neighbour is represent as '0' and '1' depends on the comparison. Then a binary string is derived by concatenating the results piece-wisely. This description method can be considered as the concatenation of the binary gradient directions, and it refers to 'micropatter' (Ahonen et al., 2006)(Zhang et al., 2009). Recently, 'CoLBP' describes the co-occurrence LBP features on two close points to improve the descriptor on complex scenes (Qi et al., 2014). 'MRELBP' use multiple filters to describe various surrounding regions to make descriptor invariant to noise and sensitive to macro-structure information (Liu et al., 2016).

BRIEF (Binary Robust Independent Elementary Features) is a descriptor that can be quickly calculated and expressed in binary code. The main idea is to randomly select several pairs near the feature points, compare the Gray values of these point pairs with Hamming distance matrix, and then combine results into a binary string. The binary strings are used as descriptor of the feature point. Recently, ORB (Oriented BRIEF) has been popular in localisation and mapping objects, which uses FAST and BRIEF to detect features and describe features respectively (Calonder et al., 2010) (Calonder et al., 2012).

SIFT (Scale Invariant Feature Transform) is a popular feature detector which is not sensitive to scale variance and orientation variance. For scale in-variance, SIFT has introduces 'scale space' which has shifted original images into different scales with Gaussian Convolution kernel and combined these shifted images into an image pyramid. Then it defines a point as a feature point in its layer in the DOG (Difference of Gaussian) scale space if it is the maximum or minimum value among 26 neighbours includes upper and lower layers. For description, the magnitude and orientation of the gradient are calculated, and a histogram of the orientation in the feature area is extracted as a vector (Ng and Henikoff, 2003).

As SIFT is a complex feature extraction algorithm and time consuming, SURF (Speed-Up Robust Feature) has been developed to accelerate feature extraction process without losing quality. SURF uses integrate image theory to describe each frame, then utilises Laplacian-Hessian matrix as a replacement to LOG (Laplace of Gaussian) to represent candidate's value. Similar to SIFT, SURF also use 'scale pyramid' to ensure scale invariant. After the feature point has been defined, x and y direction of Harr wavelet responses of a circle area around feature point with 6S radial (S is the scale value of feature points) are calculated as the orientation vector of feature (Bay et al., 2006).

2.4.1 ORB Extractor and BRIEF Descriptor

ORB feature extractor and descriptor uses FAST algorithm to detect the feature points to improve the extraction performance. The recent published FAST algorithms include FAST-9, FAST-12, FAST-11 and FAST-12. ORB uses BRIEF to describe the FAST features. BRIEF (Binary Robust Independent Elementary Features) is a descriptor that can be quickly calculated and expressed in binary code. The main idea is to randomly select several pairs near the feature points, compare the gray values of these point pairs with Hamming distance matrix, and then combine the results into a binary string used as descriptor of the feature point. Recently, ORB (Oriented BRIEF) has been popular in localisation and mapping objects, which uses FAST and BRIEF to detect features and describe features respectively (Rublee et al., 2011) (Calonder et al., 2012) (Mur-Artal et al., 2015).

2.4.2 LSD, Hough and Contour Extractors

Line extractor in image processing includes two main algorithms: Hough Transformation and LSD detector. As a comparison, LSD is faster as a local line detection algorithm and more popular in recent SLAM algorithms. But it has some drawbacks as well, such as the detected long line is usually segmented into multiple small pieces lines while occlusion or local blur occurs. In LSD working process, when an RGB image is input into the system, it is converted to a gray-scale at the 1st step. Then, LSD calculates the gradient and gradient direction for each pixel. Pixels at the edge will have larger gradient values, and these points will be extracted and saved as edge points (Engel et al., 2014).

After extraction, LBD is introduced as line descriptor. Similar to SIFT and SURF, LBD uses Gaussian scale space pyramid to overcome the scale shift problem. For description, LBD divides points around edge lines (include edge) into multiple parallel bands. By calculating the direction vector (4 directions) between the two bands, the descriptor of the edge line is obtained. This is similar to SIFT descriptor, which generates gradient histogram descriptor by calculating the direction of every pixels around feature point (Zhang and Koch, 2013).

Hough transform is a feature extraction technique in digital image process. It aims to gather the local maximal points in a so-called accumulator space. That is, the points in Cartesian coordinate are the lines in the accumulator space, and the lines in Cartesian coordinate are presented by a point in accumulator space. Line detector based on Hough transformation derives a better detection result when the lines segment to be detected is very representative in the image, such as the longer edge of an object. For non-continuous short segments, Hough line extractor is difficult to keep its efficiency. Therefore, the Hough line extractor is always been used to find the primary lines or primary line direction in an image (Chutatape and Guo, 1999).

2.4.3 3D depth Image and Planar data

Differing from binocular or monocular vision sensors, RGB-D camera provides depth data directly. The captured depth image presents the distance information in current camera's view, and the distance value is stored in 8-bits or 16-bits in every pixel. Furthermore, 16-bits depth image has only one channel and is the standard format for GPU computations. Recent OpenCV and PCL libraries have provided tools to transfer Gray-scale images to RGB images and 3D scene point cloud model respectively (Rusu and Cousins, 2011) (Bradski and Kaehler, 2008). And recent Deep learning scene parsing models can mark the labels semantically on gray-scale or RGB image to make it more understandable. (Chen et al., 2017)

For the other cases, PCL uses the camera intrinsic parameters and depth data to create a 3D point cloud model, and then calculate the normal vector of each point. But comparing the normal vectors with RANSAC and other algorithms, the planar areas of 3D models can be extracted efficiently. The problem for depth data is its captured data from consumer-level sensor is not reliable, and its 3D model always contains errors. In addition, the sensor has a limited working range and will malfunction when facing in-reflected objects (Kurban et al., 2015).

2.5 RGB-D Visual Scene Mapping Approaches

A detailed projection Scene model is the prior requirement of Projection-SAR. With years of development, indoor scene mapping algorithms have been developed in various forms and could be distinguished by processing time: off-line process and on-line process. The most basic visual tracking and mapping algorithm is SFM (Structure-From-Motion), which aims at obtaining 3D information from a sequence of 2D images. It is inspired by human vision system: while a human is moving around an object, he or she can obtain a 3D model of the object through analysing object's 2D images in various views. (Smith et al., 2016) (Schonberger and Frahm, 2016) figure 2.6 shows its typical working pipeline.

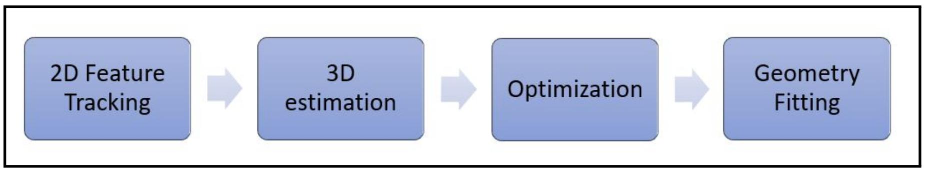


Figure 2.6: A typical Structure-from-Motion pipeline

In its first step, it detects visual features in each image through feature detection and description algorithms, such as SIFT, SURF and others. Then, the correspondence between two frames are defined. In the second step, the motion and structure are estimated by using a projection model, and a normal SFM system could handle 2 or 3 frames in a second. Then, the acquired estimation result is optimised through bundle adjustment or other iterative methods. Finally, the surface of objects or environment will be enriched. Optimisation though bundle adjustment with all the image points is a huge computation process, which is hard to achieve for live visual operations.

Therefore, SFM is frequently used in off-line reconstruction process, e.g. recovering large-scale models, the appearance of building and cities with numbers of pictures. To processing online, the algorithms follows similar pipeline at a much higher speed, such as a variety of SLAM algorithms. These algorithms could be classified into two main aspects based on their visual tracking theory, i.e. Sparse-SLAM and Dense-SLAM.

2.5.1 Sparse-SLAM Algorithms

Sparse-SLAM calculates camera's transformation matrix by extracting and matching 2D image features. As mention earlier, the usable image features include SIFT, SURF, ORB, lines, contours and others. Among them, SIFT and SURF are firstly used in SLAM algorithms, which are replace by ORB and line features in recent studies because of their slow processing speed (Klein and Murray, 2009) (Mur-Artal et al., 2015).

ORB-SLAM is popular in recent SLAM applications. It is named by ORB since is extracts and describes visual features with ORB. ORB-SLAM uses co-visibility graph to simplify tracking and mapping process in a local view-sharing area. Since it abandons many redundant frames and only analysing important key frames, ORB-SLAM is robust and powerful in large-scale mapping and tracking system. Similar to other SLAM algorithm, ORB-SLAM includes two parallel main processes. In tracking process, it extracts ORB features from sequential images, and then performing pose estimation based on last frame or re-initialising the pose through global re-localisation. After that, it tracks the reconstructed map, detect loop closure and optimise pose graphs. The mapping process mainly aims at recovering local map, which inserts the key frames, validates and test the new generated map points through local BA, and then filtering the inserted key frames and removing redundant key-frames. ORB-SLAM is widely used in a variety of visual tracking and mapping tasks such as indoor SLAM and outdoor city-sized SLAM tasks.(Mur-Artal et al., 2015)

Similar to ORB-SLAM, PL-SLAM also detects image features through ORB algorithm. As an improvement, it introduces line features to describe the scene in more detail and improve tracking performance (Pumarola et al., 2017). S-PTAM has been released currently and could be treated as a binocular edition of PTAM. As an improvement, it solves the problem that monocular visual sensor does not have a good initialisation in tracking by solving stereo constraints between stereo images. (Pire et al., 2017).

2.5.2 Dense/Semi-Dense SLAM Algorithms

Alternatively, camera tracking could be achieved by comparing the 3D information of adjacent frames, which refers to Dense tracking algorithm. Dense tracking algorithm relays on 3D point-to-point, point-plane or plane-plane registration, and will keep minimising the point cloud alignment error until it reaches a threshold, which refers to ICP algorithm. Recent Dense-SLAM algorithms include DTAM, KinectFusion, Elastic Fusion, etc. Since their tracking process use all the 3D information, their processing speed highly rely on the powerful computation devices (Kerl et al., 2013) (Whelan et al., 2015) (Newcombe et al., 2011a).

Rather than using image features for tracking, Dense approaches is more flexible in feature-less environments. In recent studies, researchers found not all the 3D data are valuable for ICP registration. In addition, calculating the transformation between key-frames could result in a reliable pose estimation and greatly reduce the processing period. Therefore, the recent light-weighted Dense SLAM algorithms have been introduced as an optimisation, e.g. LSD-SLAM. These algorithms could be called semi-dense SLAM algorithms (Engel et al., 2014).

DTAM tracks camera's pose by solving Lucas-Kanade style non-linear least-squares equation of pose transformation. For dense mapping, it introduces a projective photo-metric cost volume to store inverse depth value of every pixels in the keyframes (Newcombe et al., 2011b). Kinect Fusion decreases computation burden by assuming camera's motion is small between adjacent frames, and uses a fast projective data association algorithm to avoid a dense pose estimation iteratively. In mapping process, it registers each video frame with reconstructed model to enhance its robustness by reducing external noise and camera distortion. Then it recovers 3D model by using a volumetric model represented by SDF value from each voxel grid. However, modelling with voxel grids requires huge memory space, which makes this algorithm inapplicable for large-scale work-space. So, other researchers have designed a volume shifting algorithm to enlarge the working range of voxel-based modelling, e.g. Elastic Fusion (Whelan et al., 2015).

Since handling all images requires a powerful processor and a long calculating time, some AR applications have adopted data less than dense in the tracking and mapping process, such as LSD-SLAM. LSD-SLAM track camera's pose in real-time with a Gaussian-probability distribution of sub-set pixels of images, instead of using image features or the raw data. After estimating the pose transformation, it compares current frame with last frame to check whether the camera has moved far enough. If it moves too far, a new key-frame will be established. To optimize the map, LSD-SLAM inserts the key-frames in the pose graph, and declares a distance to ensure key-frames are inserted suitably (Engel et al., 2014).

2.6 CNN in Computer Vision

Since the great victory in image object classification PASCAL 2012 Challenge, Deep Learning in image processing has attracted increasing research interest and been used in various application fields. Deep learning is the most widely known algorithms using to deal with object recognition and classification. In this field, a commonly used model is CNN (Convolutional Neural Network). The workflow of CNN can be roughly summarised into three steps. Firstly, the image is processed by multiple CNN networks with different convolution kernels so that the original features of the image can be obtained. Then these features are deeply studied and classified. The difference between estimated classification results and ground truth results will form an error function. Finally, by minimising the error function and back-propagation, the parameters in the network are kept updating until the error factor reaches the optimum.

For convolutional Neural Network, the most direct way to improve its accuracy is to deepen the network, and forms DCNN (Deep-CNN). The recent CNN model generally has more than 10 levels: from the initial LeNet (8 levels without final FC), to GoogleNet (22 Levels), and even now ResNet (51, 101, upto 1000 levels) (Canziani et al., 2016) (LeCun et al., 2015) (Szegedy et al., 2017). In fact, experiments have shown that though increasing depth of DL model can improve accuracy and robustness, but also introduces over-fitting problem. Therefore, in recent deep Neural Networks, over-fitting problem is optimized by the dropout layer (ResNet), Inception Module (GoogleNet) and other optimizing strategies. Since outstanding performance in image classification and recognition, CNN is now also applied to robotic visual re-localization problems in (Li et al., 2016) (Li et al., 2018a).

Beside CNN, Recursive Neural Network (RNN) is also introduced in visual tracking and mapping system in recent researches, e.g. DeepVO and UndeepVO. Unlike CNN, RNN pays more attention to the correspondence between adjacent images in an image sequence. Once a new image has been input into neural network, the output from the last frame will be kept and used as the second input parameter. Since RNN can analyze current frame with the result from adjacent frame, it has been developed for visual tracking tasks or camera pose tracking process (Li et al., 2018b).

2.7 SLAM with CNN

Since the first mature framework has been published in 2007, SLAM algorithm has been developed dramatically in the past 10 years and becomes a popular algorithm in the field of computer vision technique (Klein and Murray, 2007). With the development of Deep Learning algorithms, researchers have introduced the benefits of deep learning (DL) to existing SLAM algorithms.

The existing methods of using DL to optimise SLAM pipeline could be classified into two main aspects: In the first aspect, CNN is used to optimise the SLAM structure or a part of SLAM, such as using CNN to create and recognise rigid image features, using CNN to get the depth data (for monocular visual system) and using CNN to track the camera's pose. For the second aspect, researchers use CNN to mark the labels and give the semantic meaning of objects inside the scene. This approach could replace DBoW with better flexibility and accuracy. We have classify recent CNN-SLAM algorithms in figure 2.7.

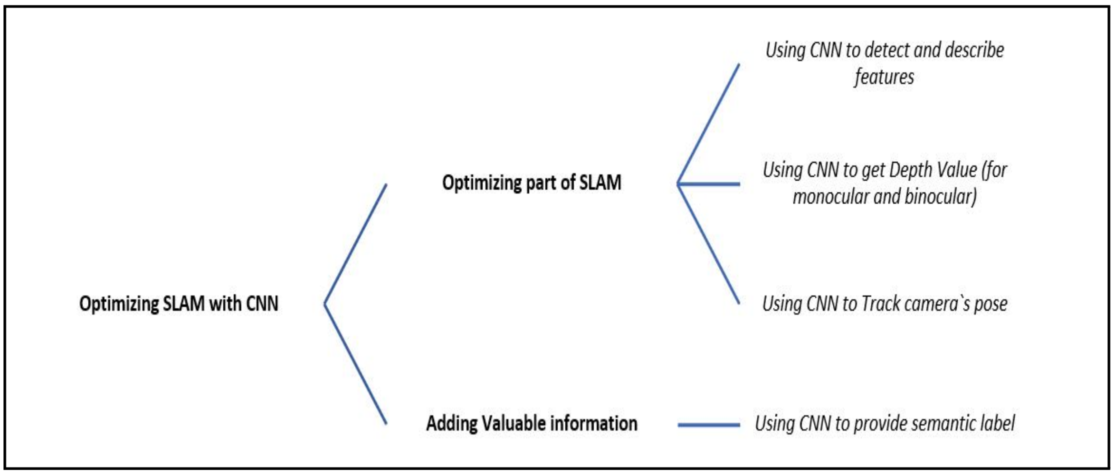

Optimizing SLAM with CNN
Optimizing part of SLAM
Adding Valuable information
Using CNN to detect and describe features
Using CNN to get Depth Value (for monocular and binocular)
Using CNN to Track camera`s pose
Using CNN to provide semantic label

Figure 2.7: Classification of CNN-optimised SLAM algorithm

2.7.1 Optimising part of SLAM

By utilising prominent feature recognition and semantic description capability of CNN, some researchers use CNN to detect and describe image features. For example, two connected deep learning networks were to build a system in the work of DeTone (DeTone et al., 2017). In the 1st deep learning network, the solid 2D image feature points are detected and located. It remembers feature point position in adjacent input images. In the 2nd network, the inputs are two points images and the output is the homography matrix used to estimate camera pose transformation. Yi and his group have introduced LIFT feature for Visual Odometry (Yi et al., 2016). It implements a complete feature point processing pipeline, including detection, direction estimation and characterisation. LIFT generates denser image features, therefore, LIFT is more robust in feature-less environments in comparing with SIFT.

Alternatively, some researcher used CNN to recover the scale factor for monocular visual systems. In the work of Grag, et al, they proposed an unsupervised framework for single-view depth prediction using deep CNN without the need for prior training and annotated ground-truth depths. The network is trained in a way similar to self-encoding, resulting in the prediction of depth information (Garg et al., 2016).

Training Convolutional Networks to estimate pose is another popular research topic. In this field, Pose-Net could be treated as a pioneering work, which is a supervised learning model and uses the basic CNN framework (Kendall et al., 2015). Pose-Net replaces the classification of the fully connected layer with the 6 degree-of-freedom pose estimation regression layer. While training the model, the ground-truth pose information is firstly computed through SFM. Based on Pose-Net strategy, other improved versions are also published recently, e.g. V2V-PoseNet and X-Ray PoseNet (Moon et al., 2018).

Another important Scene reconstruction and camera tracking strategy based on CNN-net is SFM-net, which is the upgraded version of SFM-learner. SFM-net is a geometric aware deep learning model used for estimating pose transformation in a video. This model could be trained with multiple constraints, such as photo-metric error (training in unsupervised method), depth information constraint and camera's motion constraints (Vijayanarasimhan et al., 2017).

2.7.2 Introducing semantic information to SLAM

There are two main reconstruction approaches in recent visual SLAM system, namely geometric mapping and semantic mapping. Comparing with normal reconstruction, semantic mapping appears more in automatic systems, especially in human-computer interactive designs. In a semantic reconstruction system, semantic inflating includes the description of items inside a scene, such as the color of the item, its edge features or three-dimensional information. In current SLAM systems, this information is difficult to be consistently accurate and reliable. For example, the changes of object pose lead to false detection, or errors caused by low-quality sensors.

To make an improvement, researchers use CNN to generate semantic information for

SLAM system. In the work of Li and Belaroussi, a CNN Deep learning model was combined with a LSD-SLAM pipeline to extract precise 3D semantic labels (Li and Belaroussi, 2016). It was made clear that using 2D image and 3D data cannot always provide reliable semantic models, and may lead to false and inaccurate semantic reconstruction. Following a traditional key-frame based reconstruction pipeline, its 2-D semantic information is combined with spatial consistency between connected key-frames firstly, and then create a detailed 3D reconstructed model.

Sünderhauf and his colleagues believe that a reconstructed scene should contain accurate and meaningful semantic information and geometric information in order to enable intelligent human-robot interaction (Sünderhauf et al., 2017). Their system used a RGB-D sensor and followed the ORB-SLAM pipeline. While a keyframe was detected, SSD (Single Shot MultiBox Detector) performed on it to detect objects in the scene. Then, an unsupervised segmentation DL model was used to assign each segmented object with a 3D model. By comparing the Segmented model with the trained 3D model with ICP algorithm, the system will decide whether it should create a new label or associating with an existing label.

McCormac et al. (McCormac et al., 2017) combined CNN with ElasticFusion in their Dense Semantic Reconstruction experiments. They showed that a semantic map can contribute to intelligent robots in robustness, accuracy and practicability. When a frame is inserted, they use dense tracking approach to give each pixel a corresponding label. Their 2D points are then projected into a 3D point cloud. As the camera moves, every point in the point cloud is constantly updated. The main problem of this approach is that it may have different labels when the same point is observed by different frames. Therefore, a probabilistic model is introduced by calculating the transformation of points between frames. The relationship between points and points can be calculated and the label of new point can be inferred.

2.8 Mathematical Analysis

2.8.1 Coordinate transformations

AR applications have a standard pipeline to overlay virtual graphics onto the surface of a real environment, and this pipeline includes three main processes: image capture, image rendering and composited. Each of the main processes has its unique mathematical transformation, which could be described as model transformation, view transformation and projective transformation. Their relationship is described in figure 2.8.

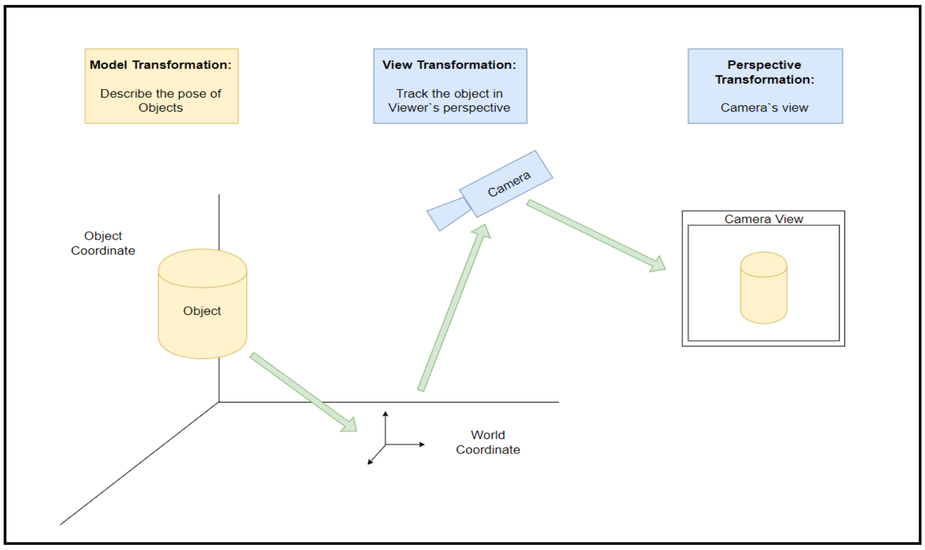


Figure 2.8: Illustration of coordinate transformation

The model transformation is the transformation between the projected objects and the scene coordinate, which means the position of object inside this scene. For the projection-SAR applications, the model transformation could be ignored as the projection object is static. But in some cases, a moving object may be projection target, and the model transformation is necessary to be updated. In contrast, the view transformation describes the relationship between the current camera's view coordinate and the world coordinate. For AR applications, as moving the cameras provides users an immersive view, calculating current camera pose continuously is the most important task.

The perspective transformation includes the information about the camera coordinate and the 2D image display coordinate. For AR applications, it means the transformation between camera's view and user's view. In a spatial-SAR system, the perspective transformation describes the relationship between the camera's coordinate and the projector's coordinate. This transformation is required to be pre-calibrated before operation. If there exist more than one camera-projector system, the sensors should be calibrated separately.

2.8.2 P-n-P pose transformation algorithm

Deriving the 6 DOF pose of camera from n 2D-3D pairs of corresponding feature points refers to a Perspective-n-Point (P-n-P) problem (Gao et al., 2003), where n means the least n pairs of correspondences required. Since each correspondence provides two constraints, PnP generates 2n constraints. To calculate 6 DOF pose transformation, minimum 3 correspondences are required. Figure 2.9 shows its calculation process.

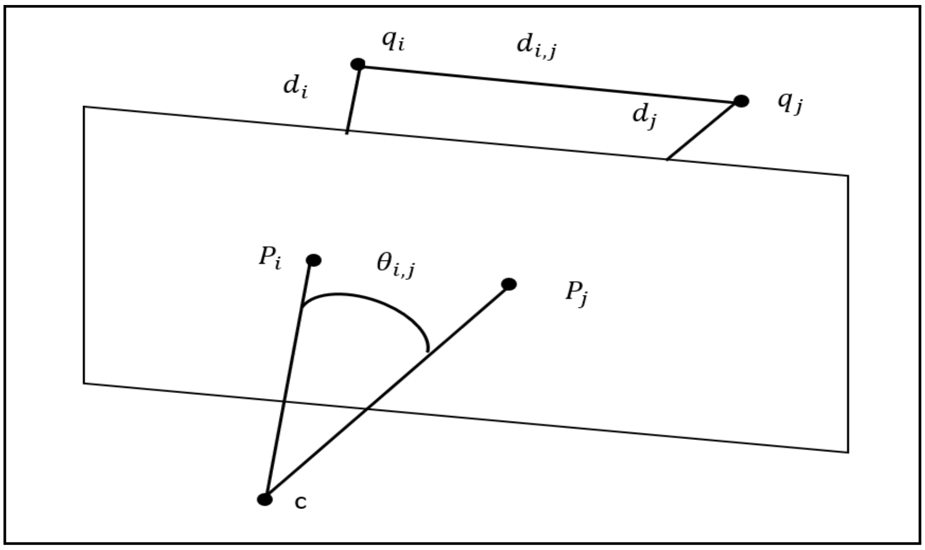


Figure 2.9: The description of P-n-P algorithm

1. select any two points q$_i$ and q$_j$ from given three points;
2. create the triangle which connects c, q$_i$ and q$_j$, where c is the camera center point.
3. calculate $d_{i,j} = abs(q_i - q_j)$ and $cos\theta_{i,j}$, where $cos\theta_{i,j} = Vec(q_i - c)Vec(q_j - c)$
4. an equation (2.3) could be established with two unknowns d$_{i,j}$ = abs(q$_i$ - c) and $d_i, j = abs(q_j - c)$

$$d_{ij}^2 = d_i^2 + d_j^2 - 2d_id_jcos\theta_{i,j} \tag{2.3}$$

   Considering the other two pairs, each pair could form one polynomial equation of degree 4. Three points result in three distances, d_1, d_2 and d_3. The equation set could be solved with at most 4 solutions. Actually, a unique best solution could be derived if we introduce another point, and this could be considered as the common solution.

5. compute the position of q_i in camera coordinate, q_i^c.

$$q_i^c = c + d_i \centerdot Vec(p_i - c) \tag{2.4}$$

6. compute the pose $< R|T >$ by aligning q_i^c with q_i, and performing nonlinear refinement of the pose.

In OpenCV (Bradski and Kaehler, 2008), it provides three P-N-P functions:

1. CV_{P3P}, which could use 4 non-coplanar features to solve pose tracking problems. Error occurs if the features points pair is less than 4
2. $CV_{ITERATIVE}$, which uses 4 co-planar feature points to derive pose parameters.
3. CV_{EPNP}, which solves parameter based on N points and determines the unique correct value only when $N > 3$.

2.8.3 Bundle Adjustment

BA is an optimisation model, aiming at minimising re-projection errors and drift error (Triggs et al., 1999). As introduced, BA optimisation is the last step in SFM. In recent years, the image feature-based SLAM algorithm also uses graph optimisation approaches to replace filter based optimisation methods. Figure 2.10 and figure 2.11 show the graph optimisation based on Bundle Adjustment.

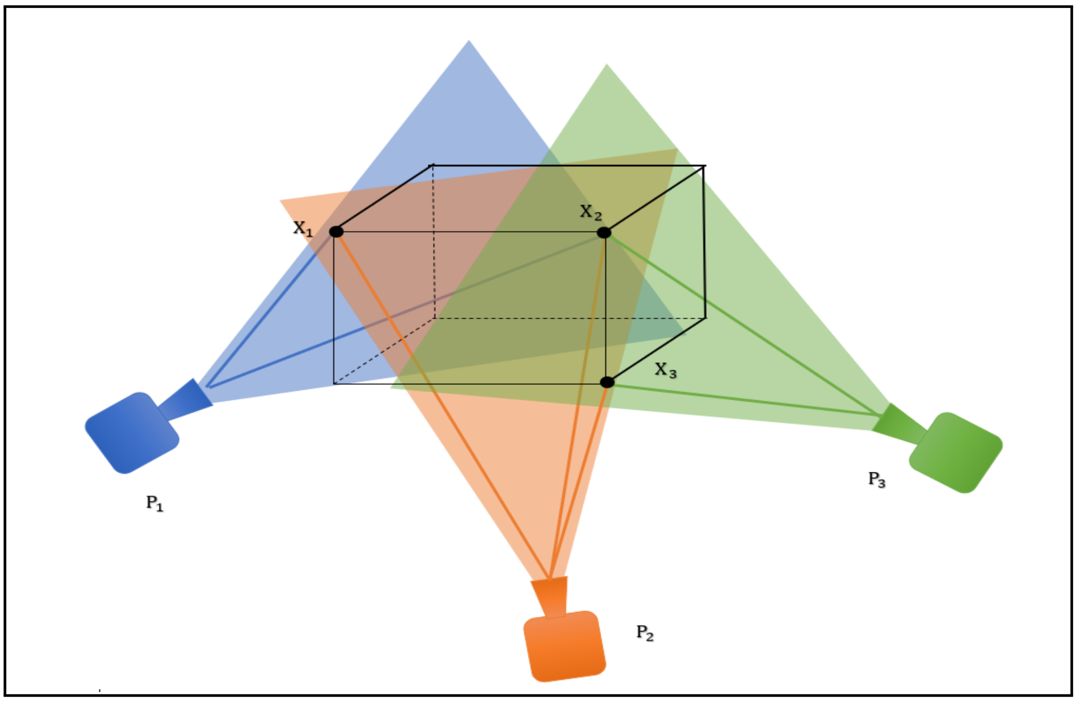


Figure 2.10: The features detected by the same camera in different pose. The coloured lines are the bundles of light.

For a camera with calibration K, let $P_(n, i)$ denote the projection of 3D point X_n to

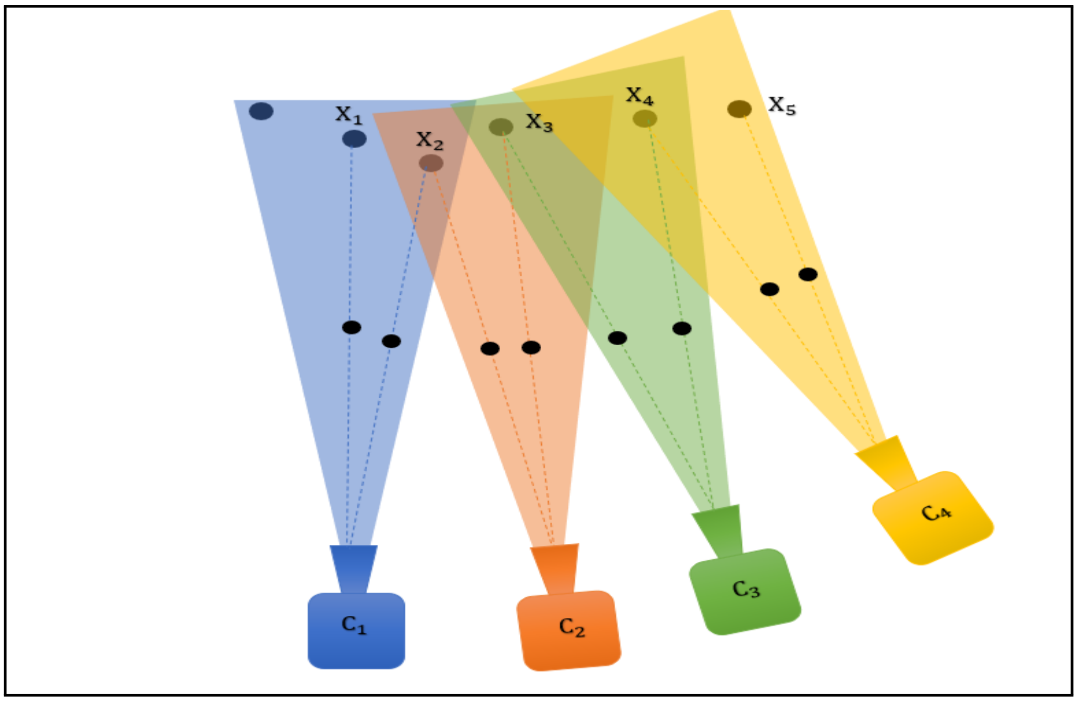


Figure 2.11: A graph model of BA. X_n is the spatial point in the scene; C_i is the camera in different positions. If X_n could be detected by C_i, they are connected.

the frame obtained with camera pose C_i. The re-projection error of a metric Error is minimised:

$$Error = \arg\min \sum_n \sum_i \rho(KC_iX_n - P_{n,i}) \tag{2.5}$$

where ρ is the robust estimator and $P_{(n,i)}$ is a homogeneous vector with three elements.

The common methods yo solve this question include gradient descent approaches, Newton, Gauss-Newton Wedderburn (1974) and Levenberg-Marquardt algorithm Ranganathan (2004). Recent the most popular method is L-M algorithm, which allows the system to switch between gradient descent algorithm and Gauss-Newton algorithm under different conditions. Therefore, L-M algorithm is more robust and efficient than the others. Gradient descent method aims at finding the minimum of a function, and its mathematical expression could be illustrated below:

$$x_k = x_{k-1} - \gamma\nabla(x_{k-1}) \tag{2.6}$$

where γ is the descent length of each iteration.

Gradient descent ensures that the function of each iteration decreases. The initial descent is very fast when the initial point is far away from optimal. However, the descent speed will be quite slow when it nears the optimal because the direction of iteration is polygonal. Gauss-Newton algorithm approximates the function with

Taylor function and then removes the quadratic terms:

$$f(x+\delta x) \approx f(x) + J(x)\delta x \tag{2.7}$$

Since we need to reduce the *Reprojection* Error, We can assume:

$$\varepsilon + J(x)\delta x = 0 \tag{2.8}$$

Then

$$J^T J \delta x = -J^T \epsilon \text{,and } x = x + \delta x \tag{2.9}$$

Thus, x iteration is δx direction until $|\epsilon|$ is the minimum. The benefit of Gauss-Newton method is it ensures a high descent speed in each iteration. However, it cannot ensure every iteration step is descent. As a result of this, L-M approach is introduced to combine the advantages of both. For the equation from Gauss-Newton, L-M changes it to:

$$(J^T J + \mu I)\delta x = -J^T \epsilon \tag{2.10}$$

Through running the value of μ, L-M could switch current applying algorithm in different situations.

2.8.4 ICP

Scanning a scene with a depth camera is usually impossible to scan all the objects in the scene by one acquisition. It can only acquire point clouds at different angles of the scene, and then fuse these points together to get a complete scene. ICP (Iterative Closest Point Iterative Point) algorithm is such a point set to point set registration method figure 2.12, which could be treated as an optimisation process of least square computation (Rusinkiewicz and Levoy, 2001).

Analysing ICP process could be presented as two functions (Li and Belaroussi, 2016),: **Given:** Points group $X = x_1, x_2, ..., x_n$ Points group $P = p_1, p_2, ..., p_n$ **Want:** Translation Matrix T and Rotation Matrix R, which minimises the sum of squared error:

$$E(R,t) = \frac{1}{N_p}\sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2 \tag{2.11}$$

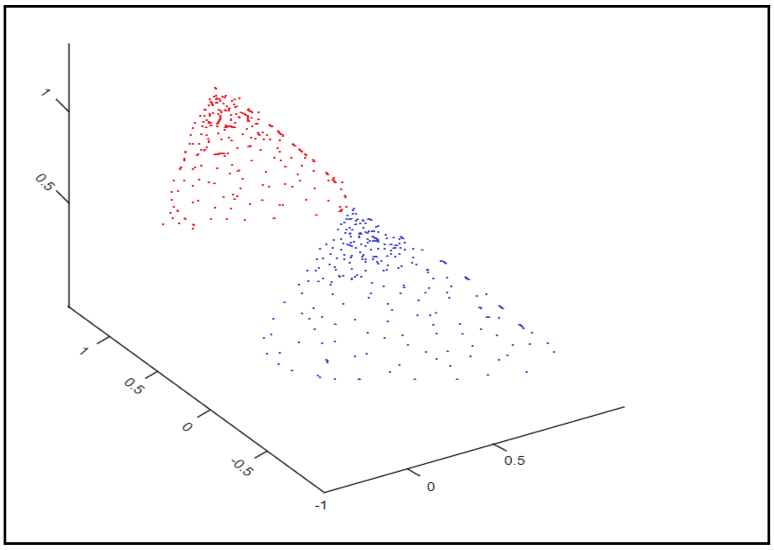


Figure 2.12: ICP algorithm calculating the transformation matrix to make Blue point cloud overlaps Red point cloud.

For ICP algorithm, if correspondences between two point groups could be define in prior, the optimisation equation has a unique solution.

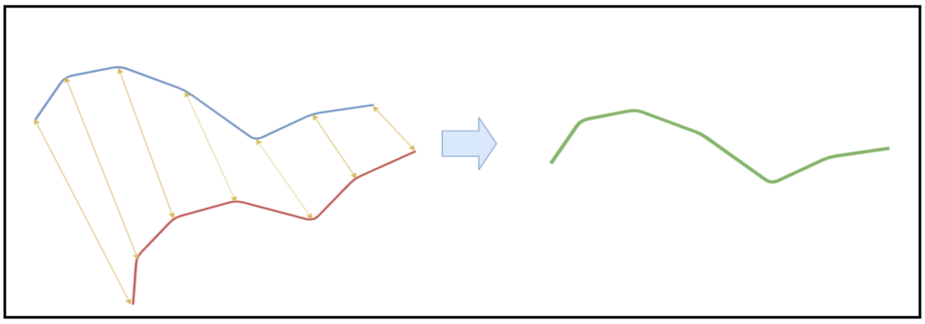


Figure 2.13: Illustration of transformation target of ICP

In the first step, the Centroid of two point clouds μ_x and μ_p is estimated using:

$$\mu_x = \frac{1}{N_x} \sum_{i=1}^{N_x} x_i, \text{ and } \mu_p = \frac{1}{N_p} \sum_{i=1} N_p p_i \tag{2.12}$$

Then the corresponding centre of mass is subtracted from every point in the two point sets before estimation the transformation. For R and T estimation, we can use SVD approach:

$$W = \sum_{i=1}^{N_p} x_i\text{‘} p_i^T \gg W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T \tag{2.13}$$

if $rank(W) = 3$, $E(R, T)$ has a unique optimisation result.

$$R = UV^T, \text{ and } t = \mu_x - R\mu_p \tag{2.14}$$

Comparing with other registration approaches, ICP could return very accurate registration result, and avoid the requirement of feature extraction and description. However, ICP is computational expensive. In addition, While ICP is searching the corresponding points, it uses the Euclidean Distance to match each point pair, which causes matching failure.

As an improvement, recent ICP uses various sampling approaches to find the correspondences, such as uniform sub-sampling, random sub-sampling, feature-based and normal-space sampling. Among them, feature-based sampling will only consider the feature-point inside the points group, which is efficient and popular in recent semi-dense SLAM approaches.

PCL lib (Rusu and Cousins, 2011) also provides other point matching strategy to make ICP flexible in different scene. The most common approach is searching the closet point, this is efficient when camera motion is small, but it runs slowly. Another strategy is searching by normal vectors, which performs better with smaller external noises. The projection approach is utilised in most of current ICP-based tracking programs because of its fast processing speed and good accuracy.It requires small camera‘s motion between frames. Recently, there are many ICP variants, such as point-to-point and point-to-plane that are the most popular and usually performs better in structured environments (Segal et al., 2009).

2.9 Summary

The characteristics of state-of-art AR broadcasting displays are firstly reviewed in this Chapter. In fact, no single display technique could satisfy comprehensive requirements of wide user communities. For example, monitor-based AR broadcasting has more mature applications than the others in recent years, but provides limited immersive experience. HMD has the best immersive experience, whereas it still struggles with latency, focus mechanism, brightness, accuracy and widespread acceptance problems. Projection display enables augmented view with naked eyes, but it has the problem that the resolution of their projected content is hardly to achieve the real-world resolutions.

Then, recent popular visual sensors and their calibration algorithm are introduced. For indoor visual tracking and mapping tasks, the best camera is RGB-D camera since it can provide the RGB and depth data directly. To calibrate the projector, the most popular algorithm is structured-light projection. And existing Projection SAR systems adopt this method. In the fourth sub-section, recent popular visual feature and feature descriptors are described. Then, we summarized and classified the state-of-art indoor visual mapping algorithms based on their camera tracking method in the following part. For feature-less indoor scenes, dense mapping algorithm can provide better mapping performance. Due to deep learning has been increasingly associated with computer vision recently, the application of CNN in computer vision and the application of CNN in SLAM are concluded in the sixth and seventh sub-sections respectively. Finally, the mathematical algorithms related to our research are analysed in the last part.

3

Plane detection and Completion with CNN Segmented Labels

3.1 Introduction

The Projection-SAR systems are usually used in a low-texture environment, such as indoor demonstration rooms, exhibition halls and Broadcasting studios. These environments typically contain large scale planes, such as projection screens, white board, black board and the surface of tables or other objects. A fixed projection-SAR system could register the projection target and obtain enough visual features by using structured-light projection technique. This is however unable to be implemented by a movable Projection-SAR system.

In this chapter, a complete plane detection, plane edge extraction and completion algorithm is introduced. Proposed algorithm aims at optimising the performance of detecting planes and their contours in non-HD images through combining different edge extractors. Compare with state-of-art edge detection algorithms in visual SLAM, our method has made three main improvements: firstly, by clustering and selecting various edge features detected from different feature extraction algorithms, our method improves the accuracy of extracting edges of different sizes in non-HD images. Among them, the Hough line transformation algorithm (Chutatape and Guo, 1999) and LSD (Von Gioi et al., 2010) is used to detect long solid edges and short less-solid edges respectively, then the Contour Detector (Bradski and Kaehler, 2008) is used to filter out the wrong detected line segments. Secondly, by analyzing the end points of line segments and the intersection points between lines, our algorithm supports the restoration of occluded edges in single image without prior knowledge. It should be noted that our algorithm can only be applied to planar objects with regular shapes and straight lines, such as surface of table, whiteboard and pictures. Lastly, we propose a depth value optimisation method based on normal vectors to correct inaccurate depth data captured from consumer-level sensors. And this method can ensure the accuracy of 3D point cloud contour model of labelled plane.

3.1.1 Features on plane

In a 3D scene model, a plane model could provide both 3D and 2D features. A 3D plane contains its normal vector and position in scene coordinate. 2D features on a plane includes its contour, texture and 2d vanishing directions. This section introduces both features and the commonly used feature extractors, as well as their limitations.

3D features

Pose and Normal vector are the most basic 3D features that could be obtained by RGB-D camera and PCL library directly. RGB-D camera estimates the depth of every pixel in each view. After converting 2D image to 3D point cloud model with estimated depth values, we can use various depth extraction algorithms in PCL lib to extract the plane models. Current popular plane extractors in PCL library include KdTree, RANSAC and region growing (Rusu and Cousins, 2011).

KdTree approach is based on euclidean distance between adjacent points in a 3D point cloud (Greenspan and Yurick, 2003). Assuming there is a plane inside current depth camera's view, the 3d point inside this plane will be very close to each other, and the point around the plane's edge will have a significant position difference. According to this assumption, KdTree approach is able to segment the scene and detect the planes. Typically, KdTree is followed by a RANSAC process for further plane extraction and deriving the parameter of normal vector of a plane. In our design, KdTree approach is not used since it cannot provide the normal vector parameter directly.

Plane extraction based on RANSAC can cluster each 3D point by analysing its normal vector direction (Schnabel et al., 2007). In the first step, it defines the direction of each point with its surrounding points. Then, the points with the similar direction in an acceptable distance domain will be clustered, and all the clustered points have the same direction. In this step, we can set the threshold of number of points in each clustered plane and leave the small ones. Finally, all the accepted plane clusters are combined together and derived a final output, which is presented by various planes. Since we have left several small planes, the final output only describes the important planes inside current camera's view. This algorithm is used as plane extractor in our algorithm.

Segmentation with region growing considers both the normal vector and distance between adjacent points (Khaloo and Lattanzi, 2017). This method describes the normal vector difference between two points as a curvature value, and the points with minimum curvature value will be considered as in a flat zone. Assuming this process start with a random point, it firstly compares its neighboured candidate points by the normal vector, and if the normal vector difference is under the threshold, the candidate points will be added to current region; then all the candidate points will be tested with curvature value and only keep the ones below the threshold. This process is repeated until it reaches the edge of the 3D point cloud. Comparing with RANSAC, this approach generates all shapes of 3D data, and is more suitable for large-scale indoor structured scenes, such as corridor and staircase.

In PCL library, normal vector estimation could be accomplished by RANSAC scene segmentation process directly. The average normal vector value will be used to optimise the noise data problem. To get a more correct estimation, it is better to set a smaller normal vector threshold to keep each plane estimation reliable, as shown in figure 3.1, in which only the table area is presented and compared to make the extraction result clearer.

2D features

2D features on plane include the texture features and contour features. The texture describe its appearance, and the contour describe the plane's shape. Contour is one of the most important information in 2D image, which describe the edge of objects. Since the shape of an objects have various types, various contour extraction algorithms have been developed. The Contour Detector (Bradski and Kaehler, 2008), Line detector based on Hough Transformation (Chutatape and Guo, 1999) and LSD extractor (Von Gioi et al., 2010) are the most popular algorithms in this field. As a mature image processing technique, these extractors are open for public and available to use in OpenCV libraries (Bradski and Kaehler, 2008). All three contour detection algorithms are based on the gradient of Gray-scale value, but they provides different functions due to the different feature extraction and description

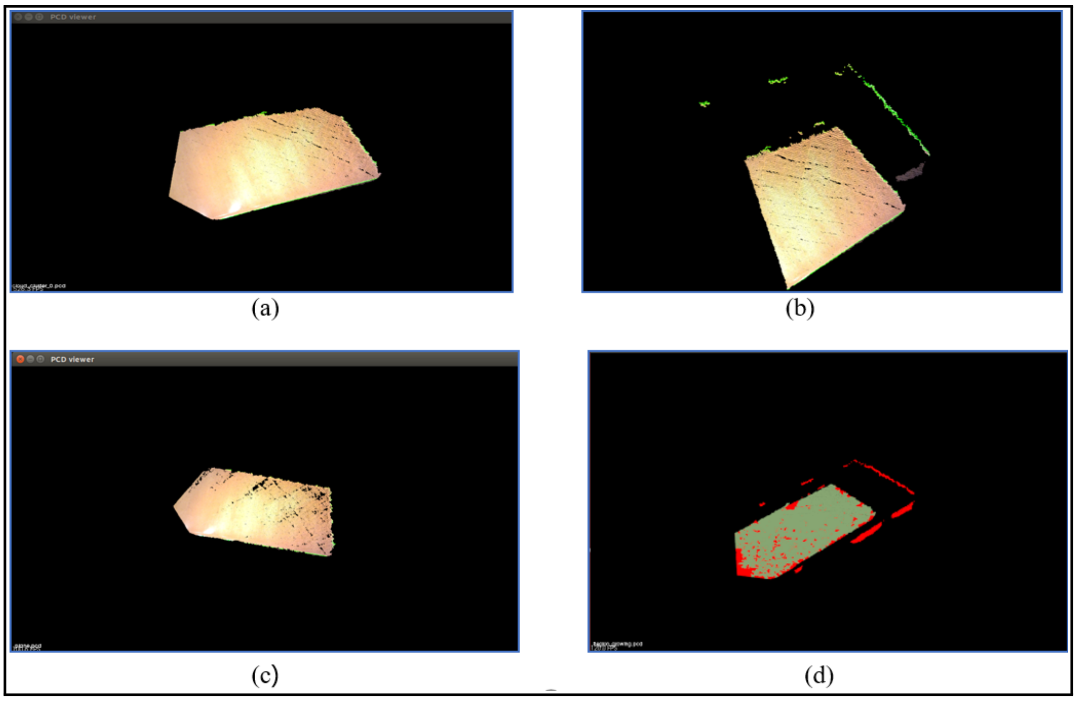


Figure 3.1: Comparison between different Plane extraction algorithms: (a) Vector-based; (b) KdTree; (c) RANSAC; (d)Region Growing.

approaches. In this subsection, the basic theory of each contour extraction will be briefly introduced, and their functional differences is analysed. A more detailed comparison will be concluded in the next section.

Contour Detector is based on the gradient of Gray-scale value and is a possible solution to contribute in detecting the edge when the image has a high resolution. However, it requires the user to set a Gray-scale value threshold manually, and then segmenting the whole Gray-scale image based on this value. Contour Detector generates a closed contour for each object. In addition, it has the ability of deleting the inside false detected line segment, which is difficult to be realized by other line detection theories. Users could choose to fill each closed contour area with different colours as a distinction. In addition, contour Detector is also available to tell a pixel is inside or outside of a closed contour in a very simple process. Furthermore, Contour Detector also provides other functions like calculating the perimeter, detecting the Centroid and locating the corner points of each contour.

However, Contour Detector always retains the outermost contour instead of choosing the optimal result from multiple detection results when the captured image is in low resolution because of visual sensor quality or shooting-jitter problem. It cannot always generate good detection result with shifting illumination since the threshold parameter can not be changed in one image sequence. These two problems could be further optimised by fusing other contour detectors.

Line extractor is another useful tool to detect the contour of an object. it is based on Hough transformation and considers the points onside the same line in Cartesian coordinate could be presented as a point in polar coordinate. Hough Line extractor also detects the edge points and corner point based on gradient change on Gray-scale image. Differing from contour extractor, Hough based detector is able to connect several line segments with the same direction into one line. Moreover, it also has the ability of choosing the optimal line among multiple repeated line segments. However, Hough line extractor is sensitive to image blur and noise though it has the ability of clustering nearby lines and find the optimisation.

LSD extractor is the most popular line segments detector in recent years. Similar to other extractors, it detects the line points and corner points based on the gradient on Gray-scale image. LSD extractor includes three main steps: in the first step, it converts the RGB image into the Gray-scale image and then re-scales generated Gray-scale image with different sizes. Then, the edges points are detected on all scaled Gray-scale image and linked. Finally, all detected lines from multiple scale spaces are projected back to original RGB image and saved as detected line features.

Compare to other line extractors, LSD runs faster and generates richer line segments because it detects the lines in various scales of image. But this algorithm lacks a method of line segments fusion. When the RGB image is clarity or has an uneven illumination intensity, LSD will generates many line segments have same direction and overlapped on each other on the same edge. In addition, LSD will produce multiple line segments with different directions on the same edge when the image is blurred. As a result, LSD extractor could be considered as a feature provider, and these features should be further filtered with other edge detectors to keep each detected line feature correct and reliable.

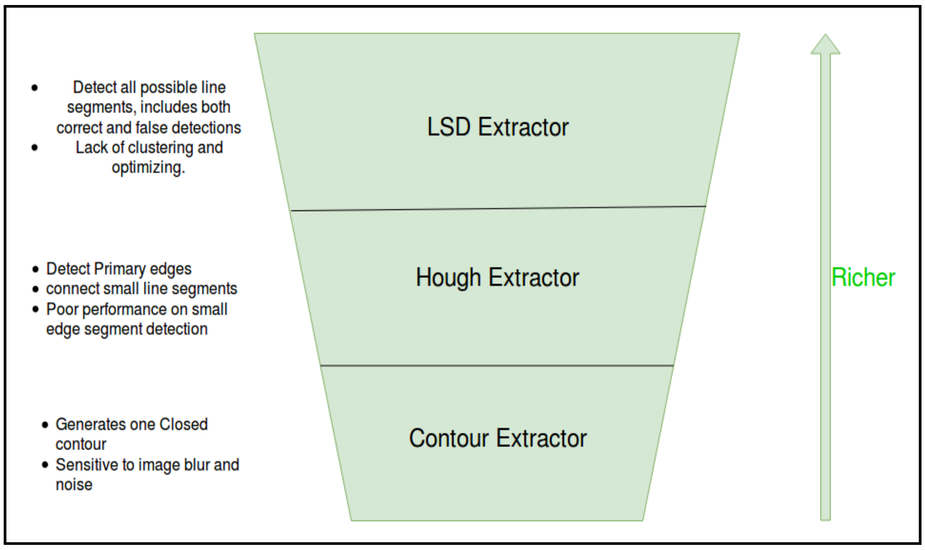
With the basic introduction and analysis of recent popular contour detectors above, we can briefly conclude the fundamental characteristics of each line segment extraction algorithms and build a reverse pyramid structure based on their characteristics, which is shown in figure 3.2.

1. Contour extractor

    (a) Only based on Gray-value threshold.

    (b) Ability to draw a close contour of an object and fill the contour area with different colours.

    (c) Ability to detect the important corner points on contour and Centroid of the contour area.



• Detect all possible line segments, includes both correct and false detections
• Lack of clustering and optimizing.
LSD Extractor
• Detect Primary edges
• connect small line segments
• Poor performance on small edge segment detection
Hough Extractor
• Generates one Closed contour
• Sensitive to image blur and noise
Contour Extractor
Richer

Figure 3.2: Analysis of recent popular contour extractors

(d) Only keeps the outermost contour of an object

(e) It is the most sensitive contour extraction algorithm to image blurring and unbalanced illumination.

2. Hough Extractor

   (a) based on Gray-value gradient and other parameters

   (b) Ability to connect small line segments in same direction.

   (c) Ability to detect the primary line features of an object in RGB image.

   (d) Ability to merge the line feature belongs to the same cluster.

   (e) Could change its parameters to generate richer line features.

3. LSD Extractor

   (a) based on Gray-value gradient and other parameters

   (b) Ability to detect the line features in various image scale.

   (c) Generates most line features compare with other detectors.

   (d) Its detected features include both correct and in-correct features.

   (e) Lack of ability to choose the optimal line among the similar line features

   (f) Lack of ability to merge the lines with similar parameters.

3.1.2 Object description and deep CNN

Currently, there are two main approaches to the description of objects in recent visual tracking and mapping algorithms. The first one is DBoW library (Wallach, 2006) describing the visual features, and the other one is the Deep Convolutional Neural Network that uses the deep learning model to generate the semantic description. Before deep learning became popular, DBoW was the main descriptive algorithm, and is frequently used for loop closure detection and lost tracking recovery.

DBoW algorithm has a training process in which user needs to capture a sequence of images about the experimental scene and then detect the visual features on every image to describe each object. The applied visual features detector and descriptor could be SIFT, SURF, ORB and LSD line features. After the feature detection, DBoW uses k-mean clustering algorithm to classify the feature descriptors and save these descriptors into a dictionary. When a new image is inserted, its feature descriptors will be searched in the dictionary and compared with matched features to get its nearest frame.

Deep learning is popular in recent object recognition, classification and semantically segmentation tasks. Comparing with visual feature based approaches, deep learning is more robust because it is not completely relying on the visual features. In contrast to feature-based algorithms, deep learning models perform better in simple scenes, and has better detection and labelling performance for large-scale empty planes. In our experiment, Deep learning model 'DeepLabV3+' from Google is used for testing planar area detection and extraction.

'DeepLabV3+' is a combination of encoder-decoder and Astrous Convolution Network: encoder-decoder aims at solving boundary information and Astrous Convolution Network computes the feature information (Chen et al., 2017). Figure 3.3 has described the structure of 'DeepLabV3+' model. DeepLabV3+ dataset adopts PASCAL dataset structure, and uses 3 channel RGB data to mark the labels. This is different to Gray-scale label dataset, like NYU dataset. It provides two pre-trained models, ADE20K and PASCAL, which are suitable for indoor experiments in this thesis. ADE20K dataset is a large indoor segmentation dataset with 151 labels, and includes items exist in most of indoor scenes, such as wall, floor, table, blackboard, projection screen, chair, door, etc. PASCAL dataset has 21 labels for both outdoor and indoor objects. Figure 3.4 shows The label images generated by two pre-trained models. More specifically, figure 3.4(a) is the result from ADE20K, and figure 3.4(b) is the result from PASCAL. For fine-tune the 'DeepLabv3+ model', a small dataset

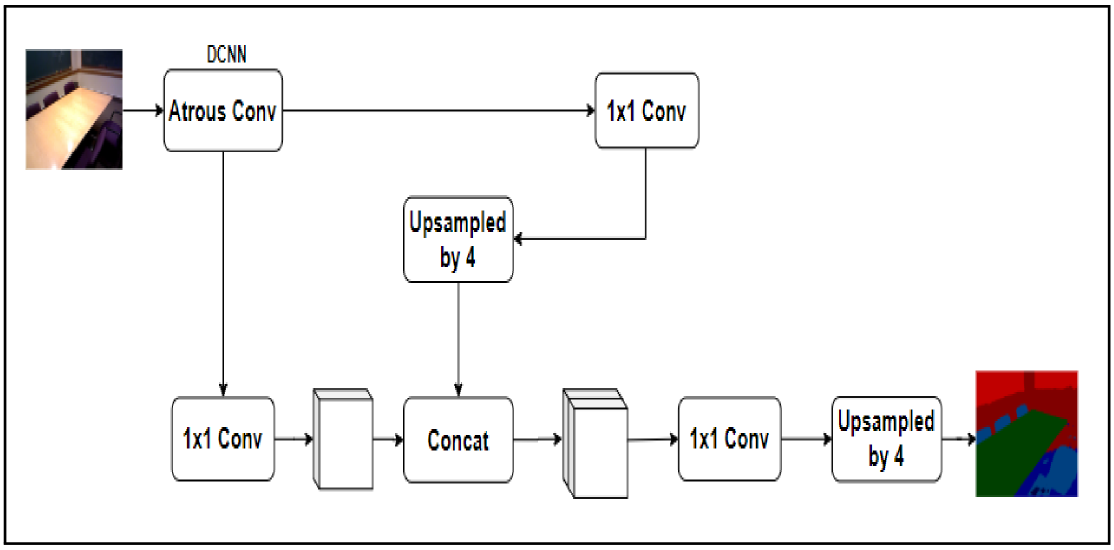


Figure 3.3: A structure of DeepLabV3+ model.
(Chen et al., 2017)

has been created, including 7 objects: table, blackboard, wall, projection screen, photo and ignore label, which are described in figure 3.5. The value of ignore label is equal to 0, which will not be learned in the training process. After the fine-tuning process, I test the model with experiment images and these results are described in figure 3.6. We can conclude the characteristic of deep learning model in scene parsing tasks in three main points:

1. Deep learning performance is better in finding and marking objects with simple structures, such as medium and large scale planes.
2. Deep learning cannot accurately and robustly mark every object, as shown in figure 3.6
3. The scene segmentation result generated by deep learning model is unreliable. We can only believe that it marks the result in a certain area, but can't believe its segmentation result, or contour extraction results in this area, as shown in figure 3.6

3.2 The proposed line feature extraction pipeline

Based on the analysis of recent popular contour extractors and line extractors, it is clear that it is difficult to obtain accurate and robust results by using only one feature extraction algorithm, especially under strong noise interference. This section proposes a novel line features extraction pipeline with two important innovations:

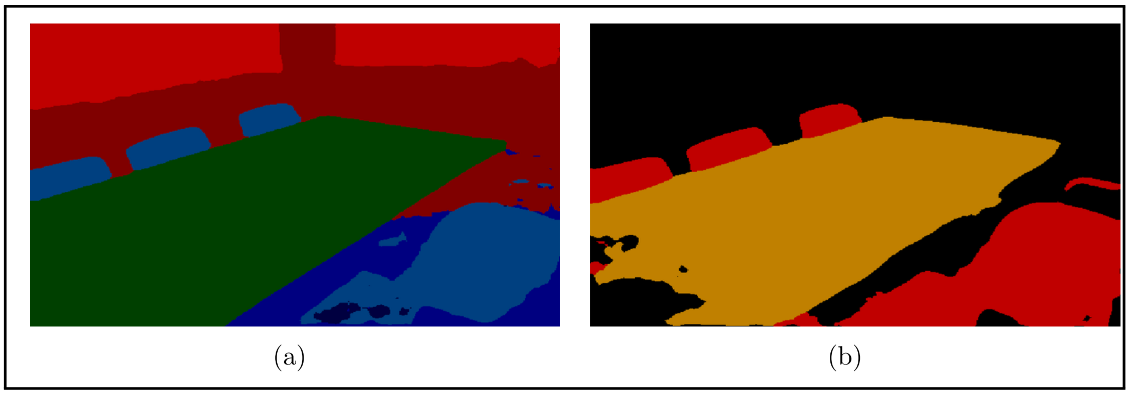


Figure 3.4: (a) is the segmentation result from ADE20K-trained model, and (b) is the result from PASCAL-trained model.

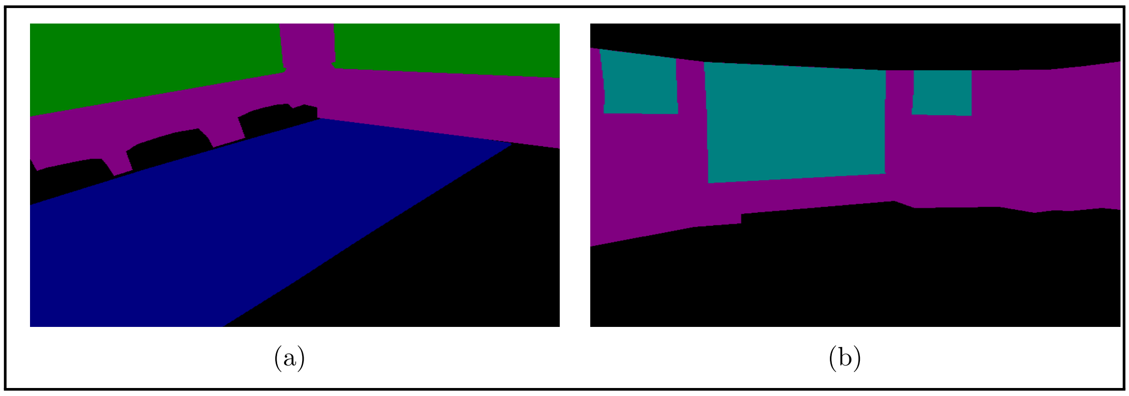


Figure 3.5: Our label dataset for fine-tune

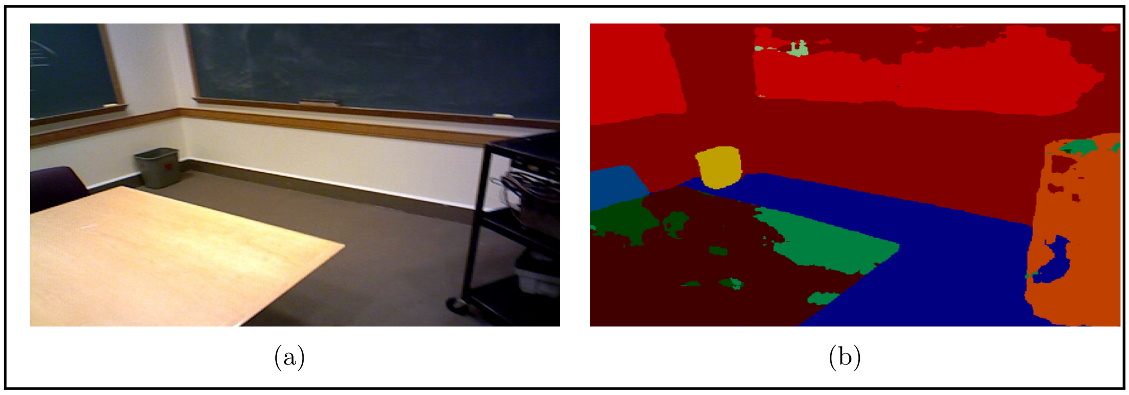


Figure 3.6: Segmentation failure. Most of pixels in table region are not labelled as 'table'

- We combine different contour extractors to detect edges of different sizes in non-HD image. Firstly, the extractor based on Hough Line Transform (Chutatape and Guo, 1999) is applied to detect the long solid edges and connect the small line segments on the same edge. Since Hough Line Transform is difficult to find the small lines, we then use the LSD detector (Von Gioi et al., 2010) as supplement. On the one hand, LSD can detect short line features. On the other hand, when the image is not high definition, LSD is able to provide less-solid line features as a candidate of final edge feature. Finally, we use the Contour Extractor (Bradski and Kaehler, 2008) to delete the false detection not in contour region (like the false detection inside the plane, which is generated by uneven illumination).

- After edge detection, we cluster line features based on their directions. And then define the optimum of each cluster by comparing their direction and length with average values.

Compared to the current feature extraction algorithms, our algorithm can extract the edge segments closest to the real edge in the blurred image, which is not sensitive to uneven illumination. Moreover, after the edge line segmentation, we have introduced another algorithm to define the characteristics of each detected important contour points. By analysing these contour points, our algorithm could extend or connect the contour lines between two defined corner points. This has solved the small occlusion problem for recent edge detection technique. This research will be introduced in more detail in Section 3.3.

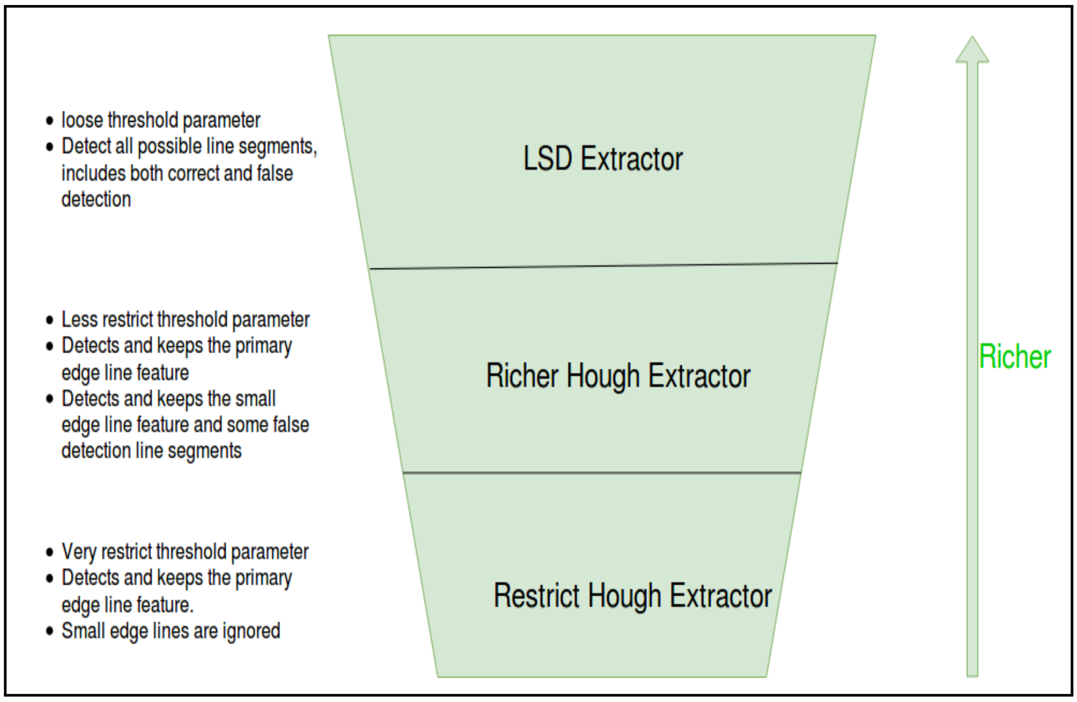
3.2.1 Design idea and proposed structure

This subsection will introduce the design idea and the structure of the proposed contour extraction algorithm. The proposed system uses three contour extractors: Contour Detector (Bradski and Kaehler, 2008), Hough Line Transform (Chutatape and Guo, 1999) and LSD Detector (Von Gioi et al., 2010). Assuming the sensor we are using is not a high-definition sensor, and the captured image has a motion blur problem. That means we can't get the correct edge extraction result from any feature extractor directly, and detection results include both correct and false detection.

We can set appropriate threshold parameters to every feature detector based on their characteristics, and then establish a multi-layer feature filtering framework based on the credibility of obtained features from different feature extractor, as described in figure 3.7.

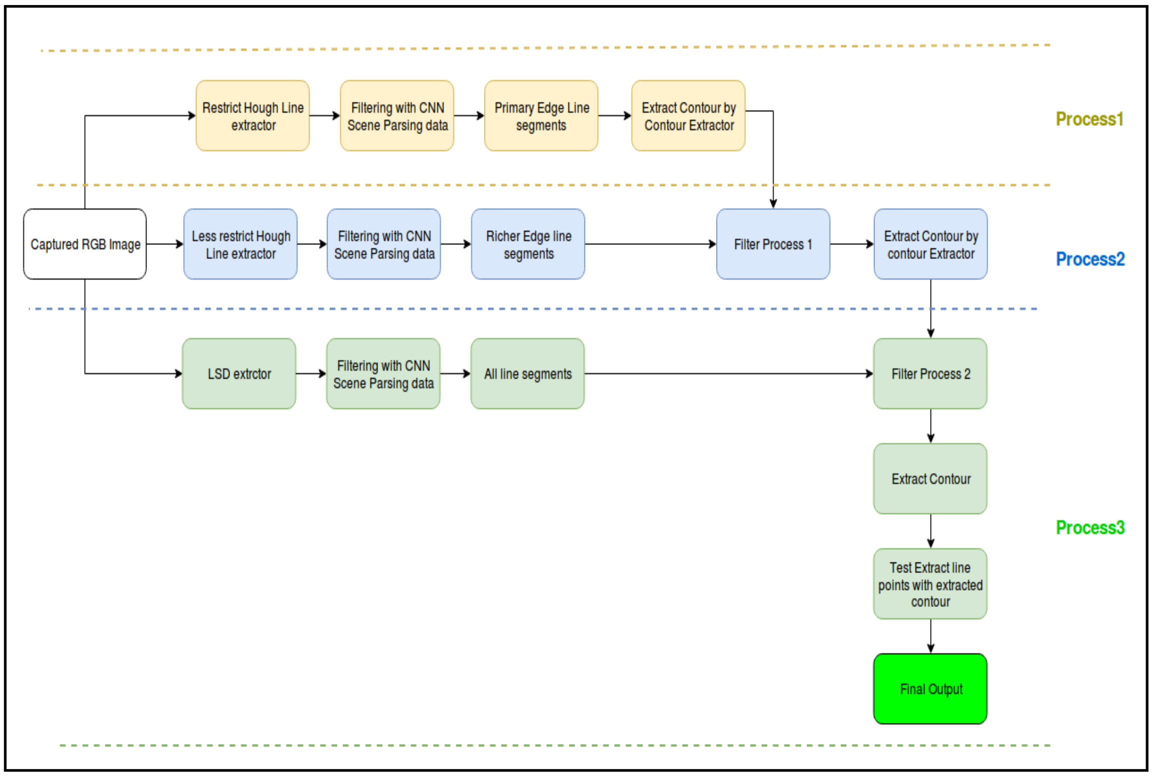
In this structure, the contour extractor is not included. A contour extractor with a restrict threshold parameter can accurately describe the area of an object, but can not accurately detect the contour of the object especially when the image is blurred. We use it as a feature filter in every feature level. This will be introduced in more detailed in the following content in this section.

Figure 3.8 shows a flow chart of the proposed contour extraction pipeline, which is



loose threshold parameter
Detect all possible line segments, includes both correct and false detection
Less restrict threshold parameter
Detects and keeps the primary edge line feature
Detects and keeps the small edge line feature and some false detection line segments
Very restrict threshold parameter
Detects and keeps the primary edge line feature.
Small edge lines are ignored
LSD Extractor
Richer Hough Extractor
Restrict Hough Extractor
Richer

Figure 3.7: A description of applied line feature extractor in purposed system



Restrict Hough Line extractor
Filtering with CNN Scene Parsing data
Primary Edge Line segments
Extract Contour by Contour Extractor
Process1
Captured RGB Image
Less restrict Hough Line extractor
Filtering with CNN Scene Parsing data
Richer Edge line segments
Filter Process 1
Extract Contour by contour Extractor
Process2
LSD extrctor
Filtering with CNN Scene Parsing data
All line segments
Filter Process 2
Extract Contour
Process3
Test Extract line points with extracted contour
Final Output

Figure 3.8: A flow chart of our proposed contour extraction pipeline

based on structure of feature extractor in figure 3.7. There are three processes in our proposed contour extraction pipeline. The detail of each process and experimental results in each process will be introduced clearly in the following subsections.

In this experiment, the table surface in this scenario is set as the landmark plane.

The aim of this experiment is using the proposed line extraction algorithm to successfully extract an optimal contour line of this table robustly and correctly when the input image sequence is not in high-definition and some of the image has very obvious motion blur error. The CNN scene parsing result is not changed throughout the whole process, and we have labelled the table area or part of table in every input image. Figure 3.9 shows the table surface detection result from "DeepLabV3+" Scene parsing model. The dark green area is describing the table. In image 97, the CNN model cannot detect the whole table correctly. This problem could be optimised by our line extraction algorithm in the following process.

3.2.2 Line Extraction Pipeline: Process 1

Process 1 aims at detecting the primary edge lines and measuring their parameters, such as direction. The lines detected in this procedure may not be saved to the final line extraction output, but the parameters of these lines will be kept as an important reference and saved in the line direction dictionary. In Process 2 and Process 3, when the new line segment are detected, it will be compared with the parameters in dictionary firstly, and judged as a new edge feature, a duplicate feature or a wrong feature. This will be introduced in Process 2 and Process 3.

In Process 1, we use Hough line extractor with restrict parameters and scene parsing results from 'DeepLabV3+' model to get the primary edge lines of landmark plane area. In order to ignore the false detection and connect the small line segments, we set the connection threshold to be 45-pixel length, the acceptable length of each detected line segment to be 25-pixel length and acceptable angle to be 1 in Hough line extractor. Meanwhile we make the corner detector in the Hough extractor more demanding, resulting in fewer but more precise line features.

This method has made the contour extraction in process 1 derives only the long and confirmed edge line segments, and the experimental results is described in figure 3.10. The green points on the edge of plane are the detected line segments from Hough line extractor. The contour detector is using the original RGB image as input and the red lines in every image is the results from it (some contour points are overlapped by detected line segments from Hough extractor). The green line outside the table area is generated by the false detection. The false detection is not deleted since it does not affect the filtering result and final output.

In figure 3.10, we can find that only the primary edge features are detected and described in green colour around the table. Since the Hough line extractor in this

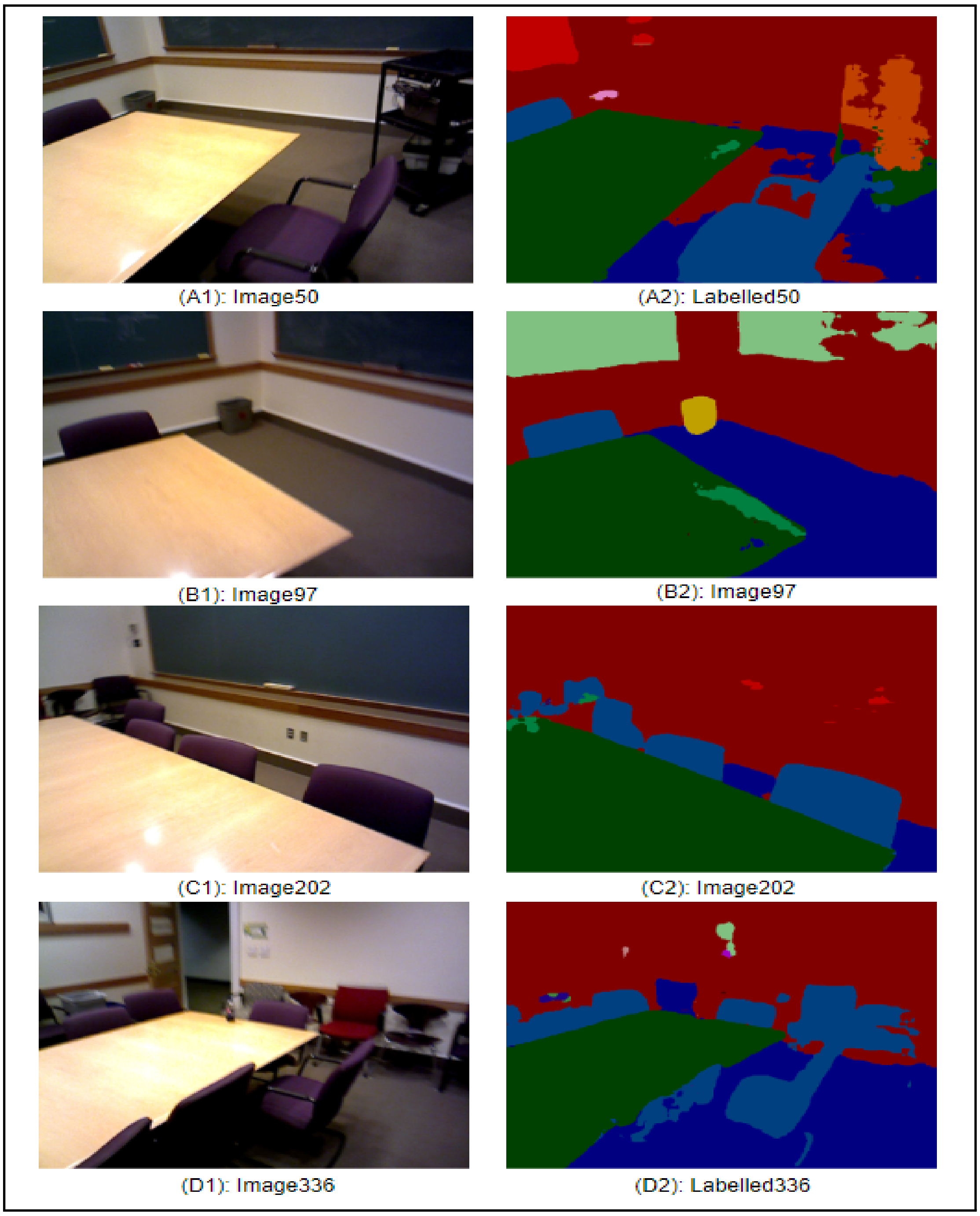


Figure 3.9: Segmentation results from 'DeepLabV3+' Model. (A1), (B1), (C1) and (D1) are raw RGB image. (A2), (B2), (C2) and (D2) are labelled images. The dark green regions in labelled images are the detected table surface.

process has a large connection parameter, we can see our algorithm not only keeps most of primary edge lines and connect the segments along lines, but also solves the small occlusion problem by connecting two line segments on the same edge and forming a complete edge line. This has inspired our algorithm to solve the occlusion problem, which will be introduced in more details in Section 3.3.

Besides these successful extraction, figure 3.10 shows the drawbacks with this pro-

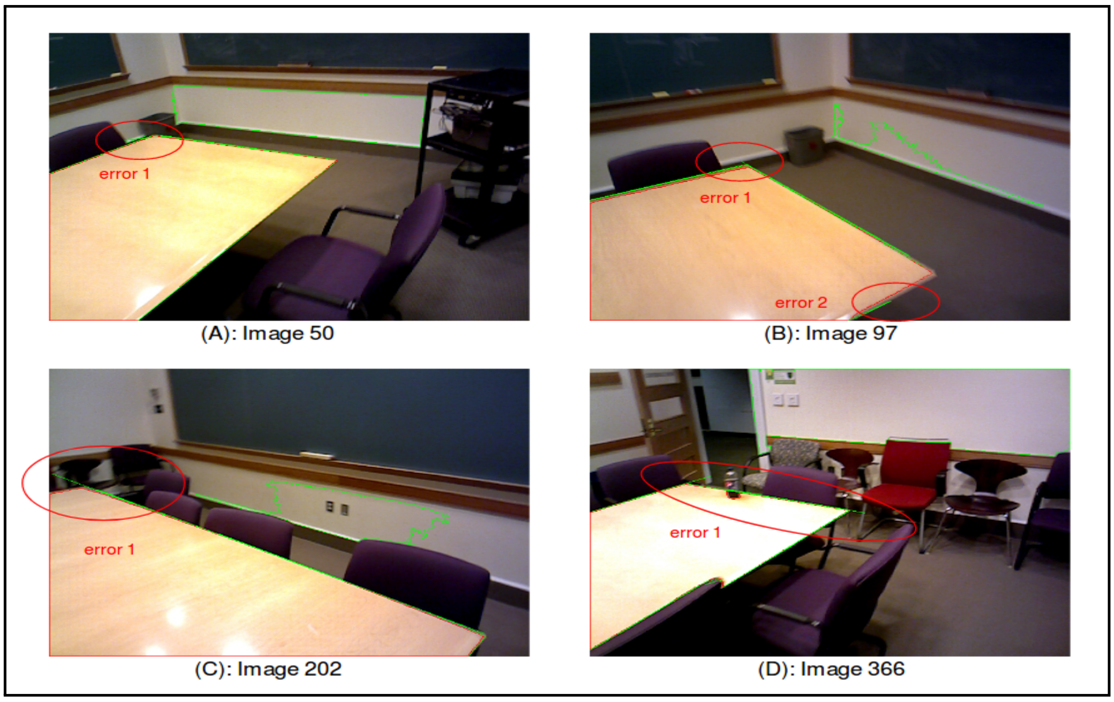


Figure 3.10: Line extraction results after the 1st process. Note that the original image of each example is shown in figure 3.9 (A1), (B1), (C1) and (D1) respectively.

cess, which can be summarized below:

- Since the feature detection algorithm in this process tries to connect the small edge lines, some of the edge lines will be out-ranged the table area, like error 1 in Image 97 and error 1 in Image 366 in figure 3.10.

- Because this feature detector aims at detecting the primary feature but ignore the small features, some true edge lines are ignored, like the error 1 in Image 202 in figure 3.10.

- This process cannot solve the motion blur problem since we use a strict parameter for corner feature point detection, and this error appears in Image 97 as error 2 in figure 3.10. In this situation, only one small line segment is detect and it is not very reliable. So in the following process we will try to detect more feature points and detect more lines, and then make the system automatically choose the optimal line between these detection.

- Though we tried to increase the minimum line pixel length requirements to reduce the false detection, it still leaves some false lines with wrong direction, like error 1 in Image 97 and error 1 in Image 202 in figure 3.10. These false detected lines will be deleted in the next process by considering its credibility.

3.2.3 Line Extraction Pipeline: Process 2

We have established a line segment dictionary which saves the direction of every detected primary lines from Hough line extractor in Process 1. In this step, we continue to enrich the dictionary: on the one hand, we add new certain line segments, and classify the line segments according to their credibility and orientation; on the other hand, we test existing line segments with new extracted edge line segments, the old imprecise line segments will be downgraded or removed from the dictionary. It should be noted that the feature segments obtained in this step may not appear in the final result, but the direction parameters of the these detected line segments are added to the dictionary in different levels.

In Process 2, we use a new Hough line extractor with a smaller connection parameter and a smaller minimum pixel length requirement. This causes the Hough extractor losing the ability to solve small occlusion problems, but producing richer line segments in the table area. Figure 3.11 shows the line extraction results after the 2nd process. To distinguish the result from the 1st process, the blue lines in this figure describes the extraction result from Hough line extractor in Process 2. The contour detector is using the original RGB image as input: the red lines in every image is the correctly detected contour feature (some contour points are overlapped by detected line segments from Hough extractor), and the green line outside the table area is generated by the false detection.

The error 1 in image 366 in figure 3.11 has shown that this detector does not have the ability of connecting the line segments along the same edge. Then, we check the new extracted line segments with dictionary, and it has the following conditions:

- The new line segment has found a matched element in dictionary. In this condition, this line segment will be kept in the dictionary. The credibility of matched element in dictionary will be added with 1, and this element will be upgraded to a higher level. For example, in Image 50 in figure 3.11, the new detect lines described in blue colour is confirmed to have a high credibility and accurately describe the direction of one edge.

- The new line segment has not found any matched data and considered as a new edge. This happens because our search strategy misses some small edge feature segments in the previous layer. When a new segment does not find a match in the dictionary, we will determine if there is a nearby contour segment. If the number of points in this line segments which is near to contour points, is greater than 80 percent of the length of the line segment, we think that this

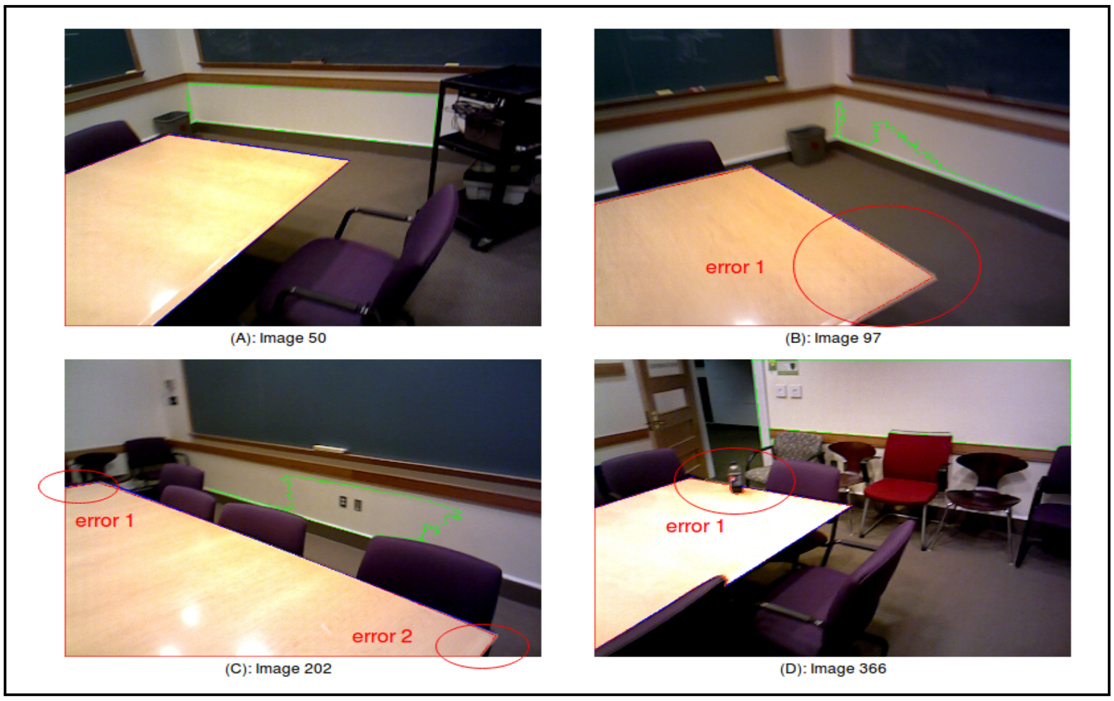


Figure 3.11: Line extraction results after the 2nd process. The blue lines along the table edges are detected line features. Note that the closed green contour outside the table region is the false detection generated by Contour Extractor. They are kept in this step since they do not affect the detection results.

line segment has a similar direction to the nearby segment of the contour, then save it and its direction to the dictionary, as shown in error2 in image 202 in figure 3.11. It should be noted that we do not think This line segment must be correct, so we set its credibility value to 1, all the direction with 1 credibility will be filtered again in Process 3.

- The new line segment has not found any matched data and considered as a false detection. This happens when the new detected line segment is generated by false detection. We use a method similar to condition2 to determine whether this line segment is parallel to the contour segment. If this line segment does not meet the requirements, then we will discard this detected line feature segment.

Besides the process of matching new line segment to the dictionary data, we also test every dictionary element with new detected line feature segments. For the elements which are matched with new features, their credibility will be added with 1 and upgrade to a higher level. For the elements which are not matched, their credibility will be minus with 1. The elements with 0 credit will be tested with extracted contour, if it has a nearby parallel contour, this elements will be kept with 1 credit and tested again in the next progress; otherwise, this element will be removed from

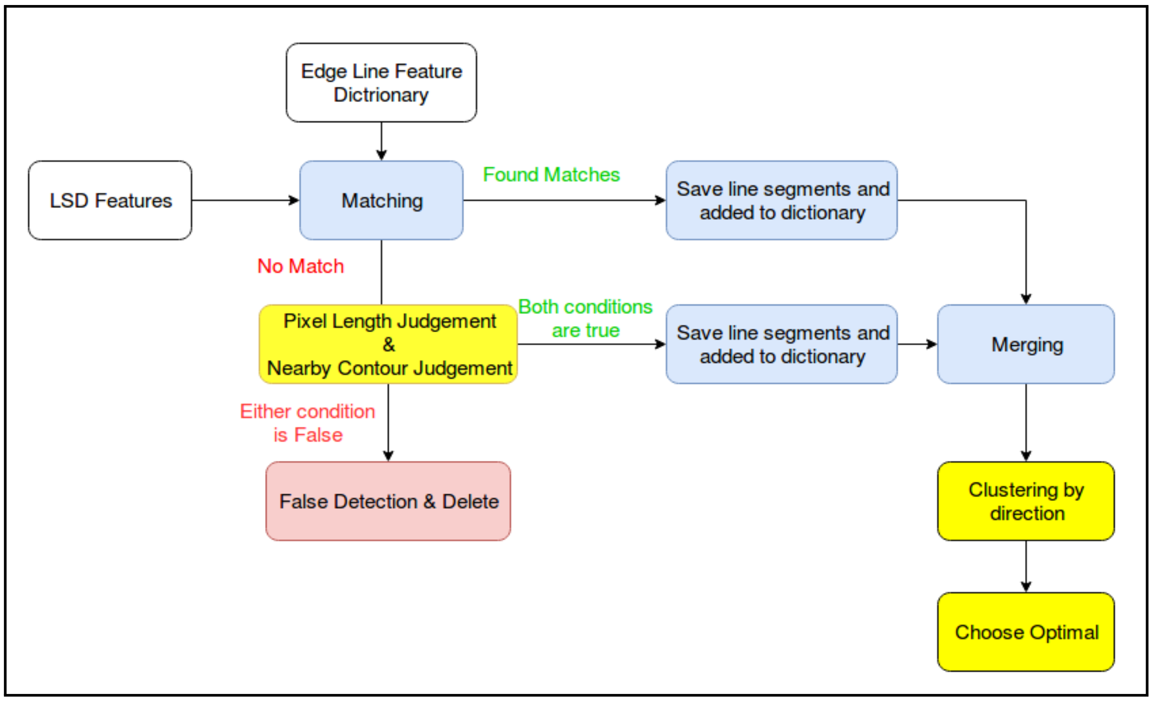

Edge Line Feature Dictrionary
LSD Features
Matching
Found Matches
Save line segments and added to dictionary
No Match
Pixel Length Judgement & Nearby Contour Judgement
Both conditions are true
Save line segments and added to dictionary
Merging
Either condition is False
False Detection & Delete
Clustering by direction
Choose Optimal

Figure 3.12: Line extraction and filtering pipeline in Process 3.

dictionary.

Similar to Process 1, the detection results will be added to dictionary, but may not appear in the final detection output. After Process 2, we can get most of the true edge lines, and the rest of undetected feature will be added in Process 3.

3.2.4 Line Extraction Pipeline: Process 3

After Process 2, the line feature dictionary has been richer and include most of edge features. However, since Hough line extractor does not provide all the line features, and some of them are not very accurate, we add more line features in Process 3 and choose the optimal from each clustered feature group. In current process, we use LSD extractor and contour extractor as the main extractor. LSD extractor is used to detected all possible line features in current camera‘s view, and contour extractor is used to filter out the false detected line segments outside the table area. The whole process in Process 3 could be illustrated in figure 3.12.

When LSD detector has generated the line features in current RGB image, all the LSD line features are compared with Edge line dictionary updated in progress 2. If the LSD line is matched with any elements in the dictionary, this line segment will be saved. For the non-matched LSD lines, they will be decided as false detection or new edge line segments with its pixel length and nearby parallel contour detection.

If the segment satisfy both conditions, this segment will be saved to dictionary. Otherwise, this segment will be considered as false detection and to be deleted.

For the merged new detected line features, they will be clustered by their direction firstly, and then be chosen the optimal results with the average direction. The most important processes in this pipeline are marked with yellow colour. These process will be introduced in more details in the following content. For the processes in blue colour, they won't be described, since they are replicated with earlier processes and less important.

1. **Pixel length judgement and Nearby contour judgement**
   In this process, all the line segments which are not matched with elements in dictionary will go through this process to determine whether they belong to missing feature segments or false detection. This process include two checks, and the line segments will be clustered in two groups: considered as missing edge group and considered as false detection group.

   - **Nearby contour check**. This process is similar to the contour filtering process in Process 2, and it starts with another contour extraction process on the original image with projected line segments from line feature dictionary. Projecting Hough line detection result from progress 2 on original image has forced a correct contour, and leads to a correct contour extraction result. This contour extraction result is described in figure 3.13, in which A1, B1, C1, D1 describe the contour extraction result with projected line features in line feature dictionary, and A2, B2, C2, D2 are the extracted contour describing the table surface with 2 pixel-width. Compare with contour extraction on original raw RGB image, current contour extraction result is closer to real contour in blurred image, and every edge is almost straight, like the real edge of table.

     Then, all the LSD features have not matched with elements in dictionary will be tested by its nearby contour. For every point in LSD segment, if there are more than 2 contour points is in the searching range, this LSD point will be considered as a edge point candidate. If the number of candidates is larger than 80 percent of LSD segment length, this LSD segment will be saved and tested with length check. Otherwise, this segment will be deleted and saved to false detection group.

   - **Pixel length check**. This process will check the length of LSD segment. Since most of edge line features are detected by two-levels Hough line

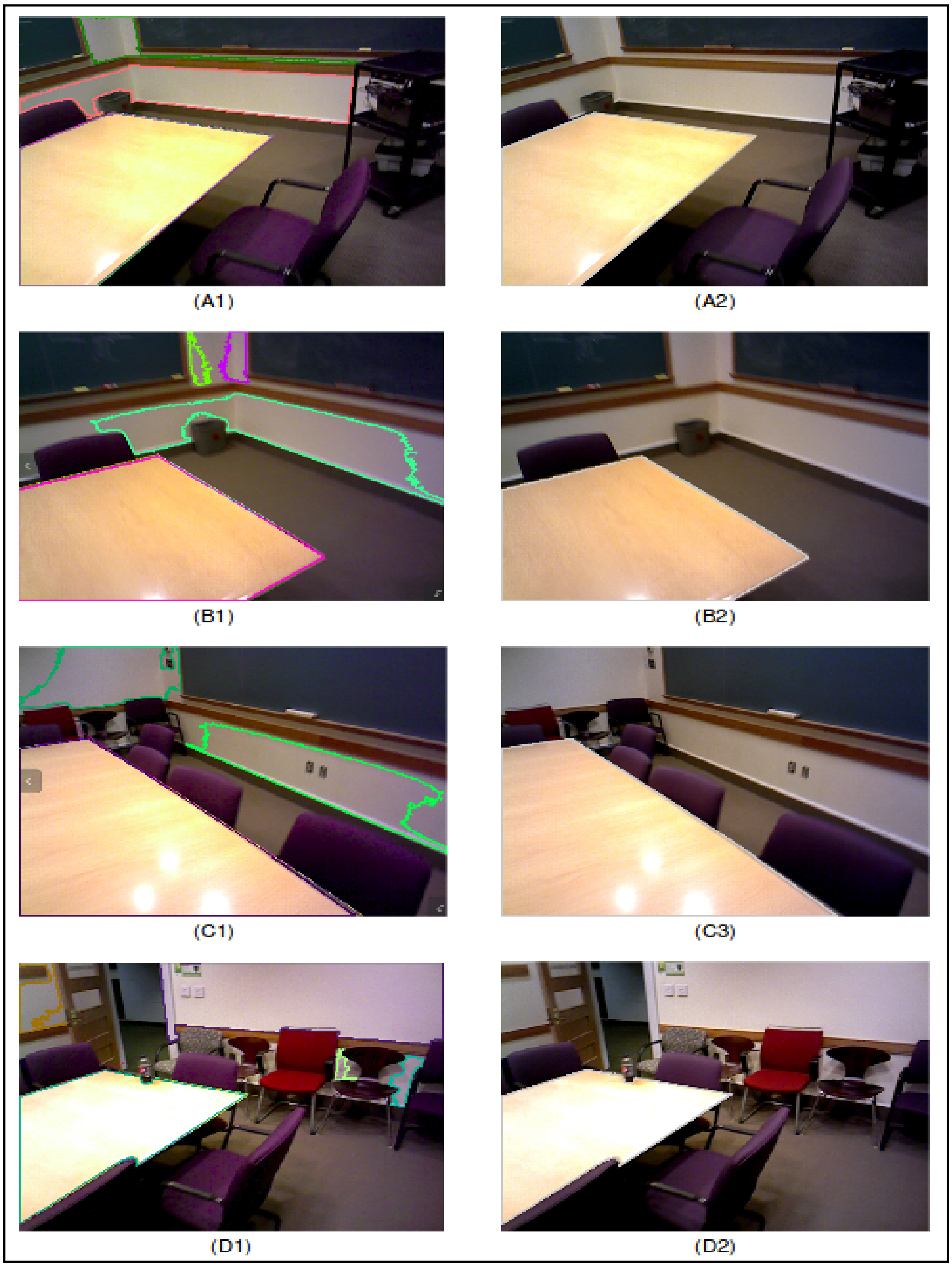


Figure 3.13: Contour extraction result with projected line features generated in Progress 2. (A1), (B1), (C1) and (D1) describes all detected closed contours. (A2), (B2), (C1) and (D2) describes the contour describing the table region. We use the label information to select the right contour: if more than 80 percent of each contour region is labelled as 'table', we use this contour to describe the table.

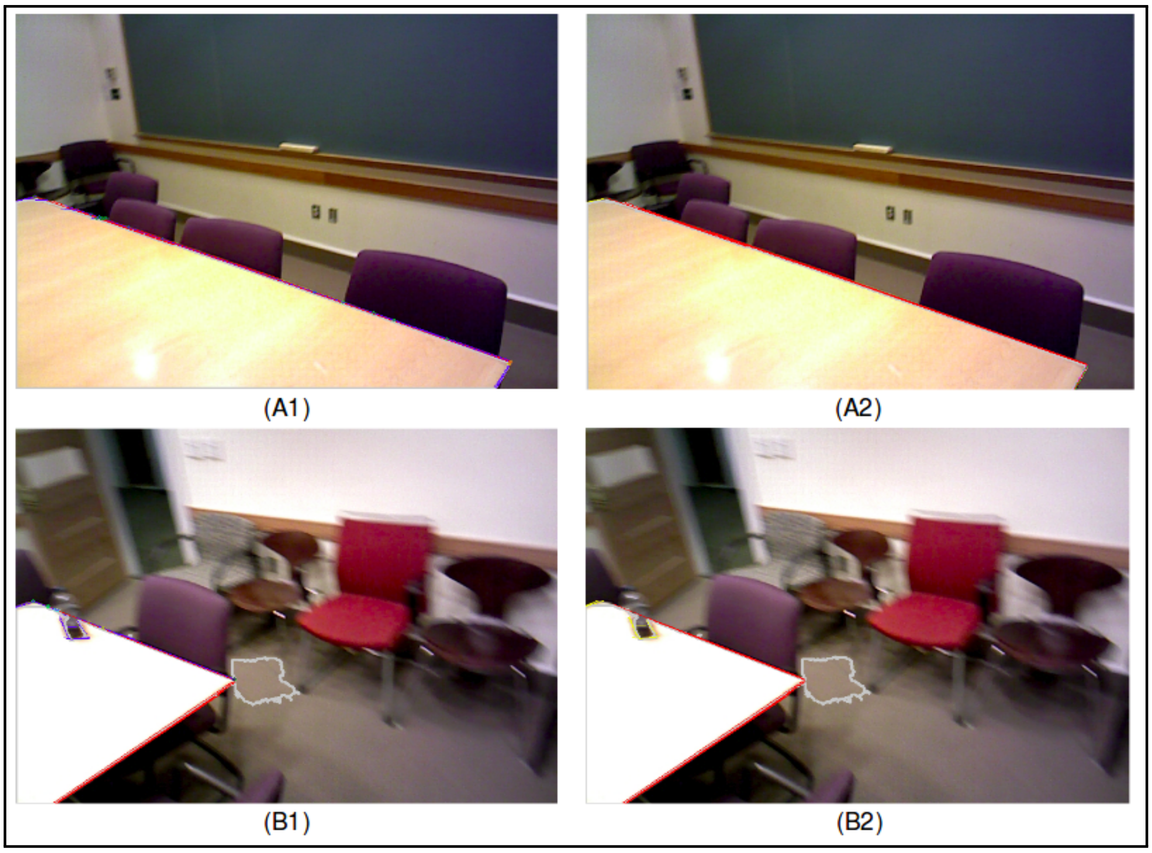


Figure 3.14: Small edge feature completion with LSD line segments. The yellow lines in (A2) and (B2) are newly added line features from LSD algorithm. The white contour outside table region in (B1) and (B2) are false detected contour connected to table contour.

extractors and only the small edges are still missing, we check the length of every segment to find the correct edge line features. For the lines which are too small (smaller than 5 pixel lengths) or too long (larger than 25 pixels) will be deleted and saved to false detection group.

After these two processes, we have a final check based on the direction of line segment. We check the direction of every line segment in missing edge group with elements in false detection group: this line segment will be removed from missing line group if it is found to be parallel with element in false group. This process could remove the last false detected lines caused by noise and image blur. The extraction results are described in figure 3.14, in which (A1) and (B1) describe the features before filtering and the remaining line features are drawn in blue. It is clearly shown that their are still small false edge line segments exists in current line feature dictionary. (A2) and (B2) are the results after filtering process, and the remaining small line features are drawn in yellow colour. After the optimisation, we can find only the line segments which are describing the edge correctly are kept, other false detection lines are removed.

2. **Line feature clustering**

   Before line feature clustering, system get multiple line features describing the same edge line because of applied three level line feature extraction processes. This process will cluster these line segments by their direction and their position in an image. It should be noted that in position check, we are not using the coordinates of each line for line segment classification. Instead, after determining several sets of line segments, we will estimate the distance between the new inserted line segment and the reference line segment in different groups for classification. After forming them into different groups corresponds to different edges. We will choose the optimal line segment in each group in the following process. Since we use three extractors, the group with one element will be considered as an unreliable feature and ignored.

3. **Choosing Optimum**

   Choosing the optimum from each line feature group is based on their directions. This process will sum all the directions and find the average direction value. Then the lines which is nearest to this direction will be considered as the optimal choice. The basis for this is that in a direction vector group, the largest number of directions is closest to the average. And having the most number of direction vectors also means that it is recognised by all three extractors, and is the most reliable among all the features.

   Experimental results in figure 3.15 show this approach always choose the optimal line segment in different image condition, in which A1, B1, C1 and D1 are the clustering results and each line cluster is described in different colour around the table area. A2, B2, C2 and D2 are describing the optimum choosing results from clustered feature groups, which are drawn in red colour. Due to the image blur and noise, some contour lines are not on the table, but they are the best choice from corresponding edge cluster.

3.3 Line feature recovery with occlusion

Occluded line recovery is one of the most important research problem in current image processing applications. When an object in the scene is partially occluded due to the change of the camera's perspective, this occlusion changes the appearance of the object itself and the scene. This is fatal for the current visual tracking and mapping algorithms. The reconstruction result are seriously affected by occlusion, in particular for our plane-based tracking and mapping algorithm. Experiments have

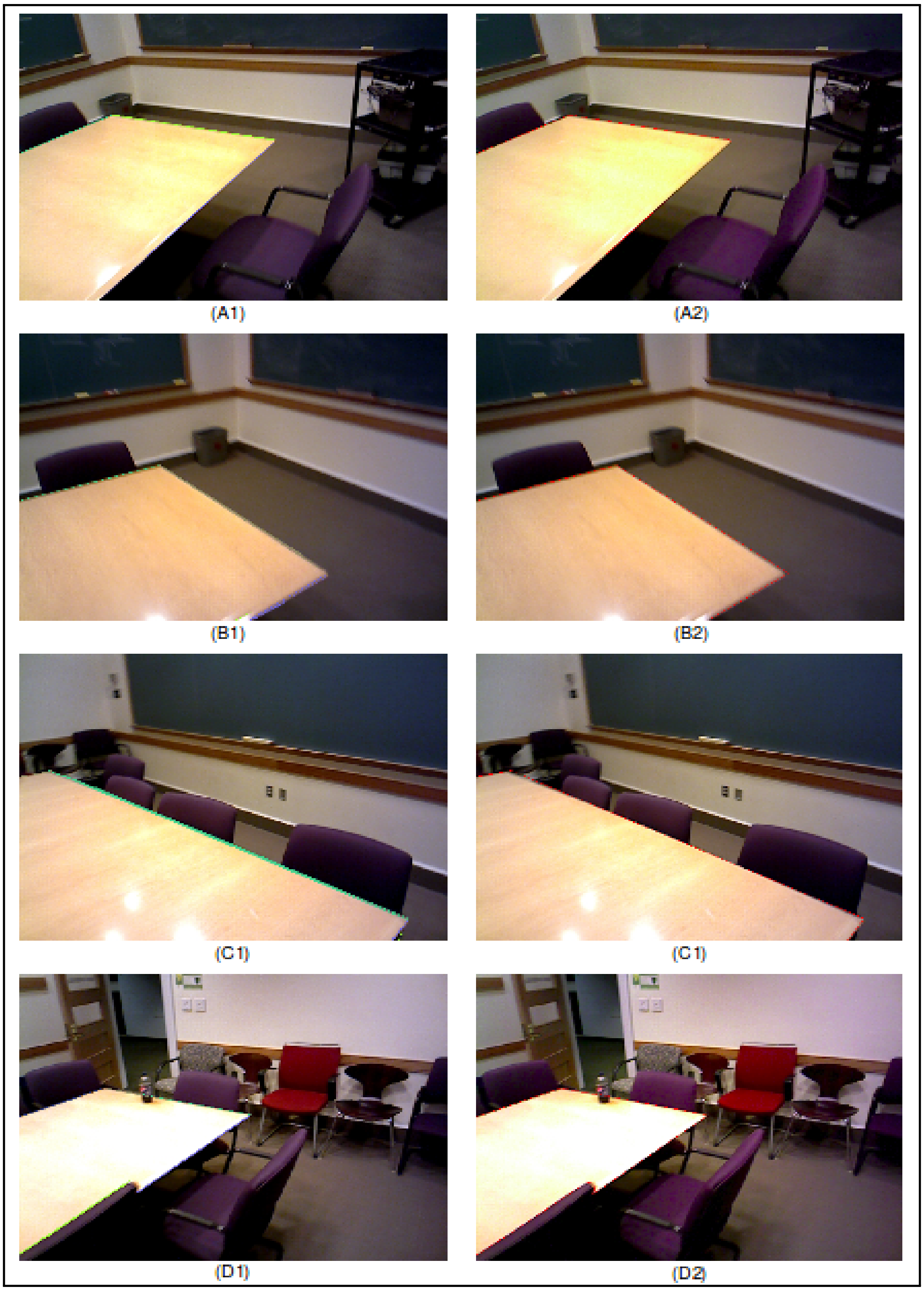


Figure 3.15: Line feature clustering result and optimum choosing results. (A1), (B1), (C1) and (D1) has described the different cluster with various colors. (A2), (B2), (C2) and (D2) are the optimal edge in each cluster.

shown that a slight occlusion can cause lost tracking problem and planar model recovery error.

In response to this problem, a novel line occlusion recovery pipeline is proposed in this thesis research. The experimental results prove that it solves the problem of plane edge recovery when the landmark plane is occluded by other objects from different perspectives. The proposed algorithm does not depend on the prior knowledge of 3D model and the knowledge between adjacent input frames, which is an innovation. By analysing the edge of a planar region with a standard structure in a single image, we can successfully predict its occluded edge line feature in the occluded part. However, when the most area of an object is occluded, our algorithm may be invalid.

Our entire algorithm first finds the intersection between the segments in different directions and saves the point on the line. Then we will define whether each segment is a complete edge segment by its two endpoints. Thus, our algorithm includes two function. The first function aims at finding the intersection between lines with two different directions, then cut the line into pieces with intersection points and saves the part which describes the contour. The next function will consider the condition of each line segment, and it will discipline the lines with complete line and incomplete line. For the incomplete line, it will be check if it has parallel neighbours, and then connect to its neighbour. These two functions will be introduced with more details in the following two subsections.

3.3.1 Feature line segmentation and endpoint detection

Figure 3.16 shows a flowchart of feature line segmentation and endpoint detection algorithm. It starts with finding the intersection between two lines in different directions. In the last line extraction process, we have clustered line segments in different group corresponds to various edges. By finding the intersection between two edge lines, it derives the intersection point either on the line segments or on the vanishing direction of line segment. Then, if detected intersection point is on the line, we will use this intersection to cut the line segment into various pieces, that is Process 2.

In Process 3, the vanishing direction will be calculated from the intersection to endpoint. More specifically, assuming one line has one intersection with the other edge line, this line will be cut into two pieces, i.e. endpoint1 to intersection as piece 1 and intersection to endpoint 2 as piece 2. Both two pieces will be extended in both

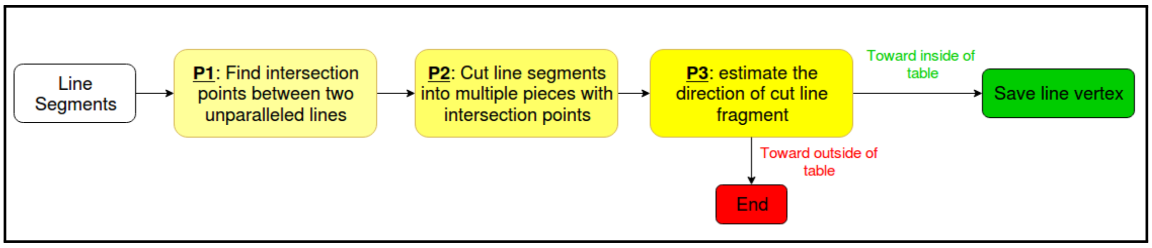

Line Segments
P1: Find intersection points between two unparalleled lines
P2: Cut line segments into multiple pieces with intersection points
P3: estimate the direction of cut line fragment
Toward inside of table
Save line vertex
Toward outside of table
End

Figure 3.16: A flowchart of Feature line segmentation and endpoint detection

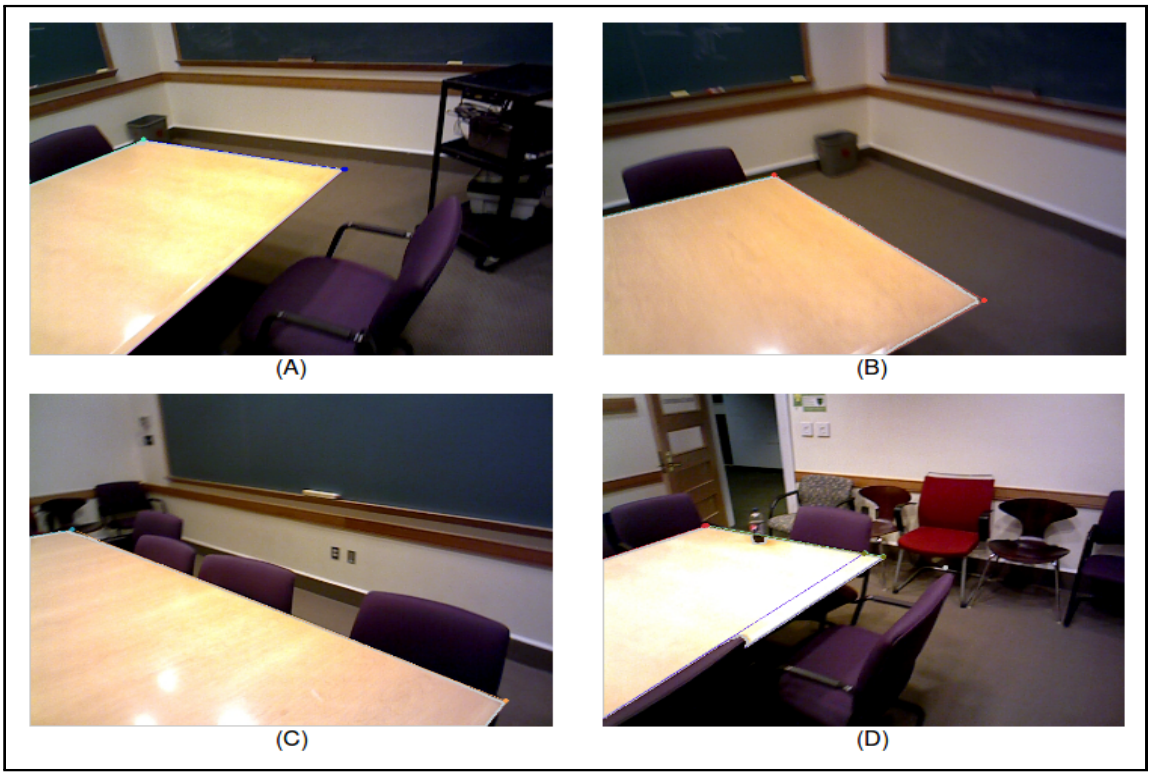


Figure 3.17: Endpoint detection result.

directions starting at the intersection with 5 pixel length and checked if the extended point is landing on the contour line or not. Since the intersection point is on the line, so if one of the pieces is describing the edge, its extension from intersection point to endpoint should be landing on the edge. This algorithm has made two contributions below:

- The method can be used to cut the line segments and save the part inside the contour when some of the extract line segments are over-ranged the contour.
- This method can be used to correctly detect two endpoints of a line segment. The derived endpoints is reliable and available to be used in the following procedure.

The endpoint detection result is described in figure 3.17, in which the endpoints are drawn in different colours corresponding to different extracted line features. The endpoints on the image boundary are not drawn. In Image 366 (Image D), The purple line in the table area is generated because of the occlusion problem, this will be deleted in the following process.

3.3.2 Extending feature line and restoring the occluded edge

In the previous step we got the edge line and its two endpoints. In this step, we will analyse whether each segment is occluded according to the endpoints of a segment. First of all, we classify the detected edge lines describing landmark planar area into two major categories and three subcategories, as shown in figure 3.18. And these categories are:

- Category 1: In this category, the detected line is a complete line without occlusion, as shown in figure 3.18, which is marked in red circle.
- Category 2: In this category, the edge line is partly occluded, and it includes two conditions:

  1. Sub-category 2: In this sub-category, the detected line is partly occluded and it has been split into two line segments, as marked in red circle in figure 3.18.
  2. Sub-category 3: In this subcategory, the detected line is partly occluded and it has one line segment, as marked in blue circle in figure 3.18.

Based on this assumption, we have designed our line occlusion recovery strategy, and this process is illustrated as a flow chart in figure 3.19. In the beginning, we extract the two endpoints of each detected edge line in an image. Then, we will calculate how many lines each endpoint is connecting. Since each line should have two endpoints, if both endpoints are connecting two lines, or either one of the endpoint is connect two lines and the other one is on the boundary of an image, we will consider this line is not occluded and it is a complete edge line feature. And these lines are in condition 1.

The rest of lines are all considered as partly occluded, with two situations. The first one is one of endpoints is connecting two edge lines and the other endpoint is connect one edge line. The second situation is both endpoints are connecting one edge. As our algorithm is based on the surrounding of each endpoint, we don't distinguish between the two cases. The experimental results also prove that our

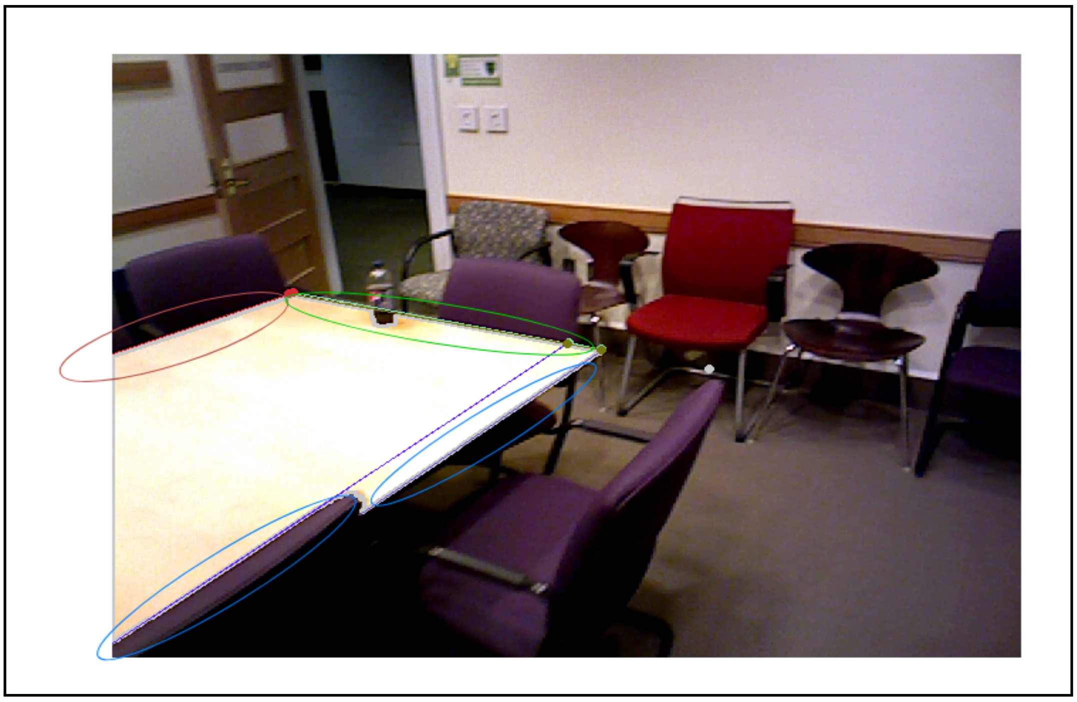


Figure 3.18: A description of line categories.

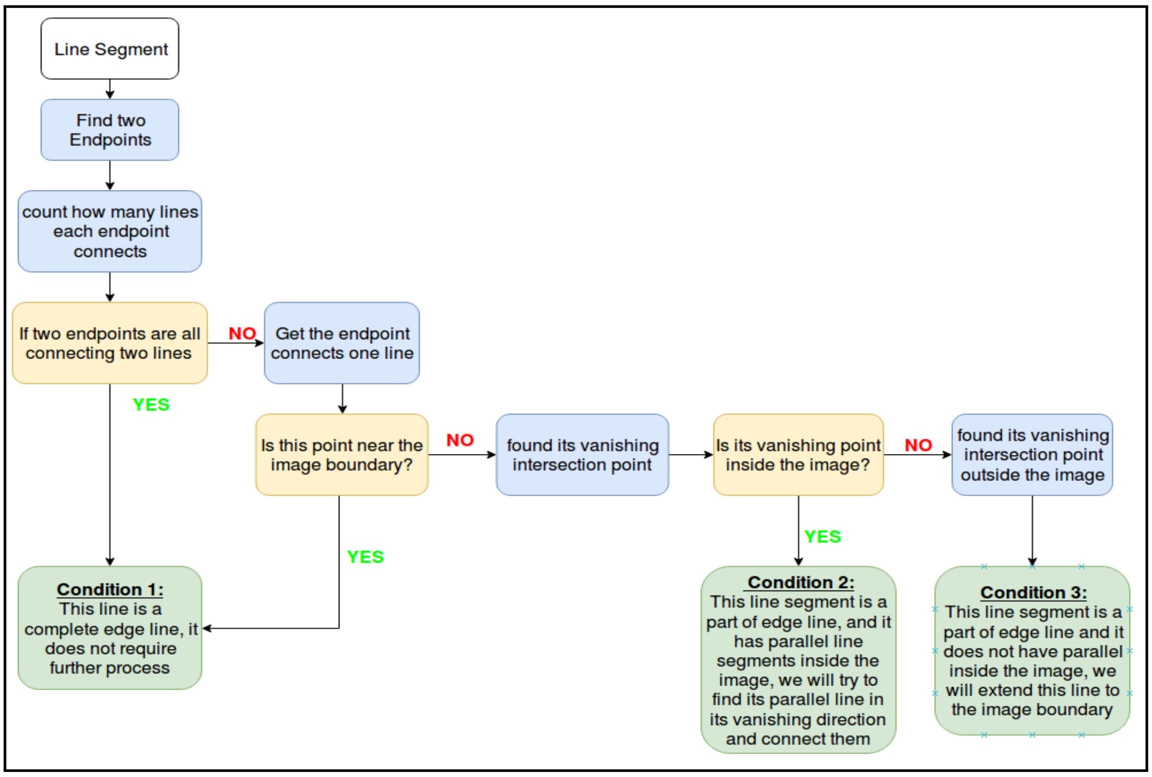

Line Segment
Find two Endpoints
count how many lines each endpoint connects
If two endpoints are all connecting two lines
NO
Get the endpoint connects one line
YES
Is this point near the image boundary?
NO
found its vanishing intersection point
Is its vanishing point inside the image?
NO
found its vanishing intersection point outside the image
YES
YES
Condition 1: This line is a complete edge line, it does not require further process
Condition 2: This line segment is a part of edge line, and it has parallel line segments inside the image, we will try to find its parallel line in its vanishing direction and connect them
Condition 3: This line segment is a part of edge line and it does not have parallel inside the image, we will extend this line to the image boundary

Figure 3.19: A flowchart of restoring the occluded edge.

algorithm can handle both cases. There are two subcategories in this category: (i) partly occluded and cut into multiple pieces; and (ii) partly occluded and appearing in one piece. Distinguishing these two subcategories is based on the endpoint which connects only one line, and the intersection point of current line. Figure 3.20 and figure 3.21 describe various cases in these two subcategories respectively.

In the subcategory (i), a complete edge line is cut into pieces because of occlusion and these pieces are visible in current image, as described in figure 3.20. The line is determined in this subcategory if one of its endpoints connects to one line and its intersection on its vanishing direction is inside this image, or both of its endpoints connect one line with two intersection points on both directions inside the image. Assuming the current image has the condition as described in condition 3 in figure 3.20, edge line L_a is a complete line without occlusion, edge line L_b is segmented into two pieces L_{b1} and L_{b2} because of multiple occlusion and line L_c is partly occluded; L_b has two intersection points $intersection_A$ and $intersection_C$ with L_a and L_c respectively, L_{b1} is the line between $EndPoint_{B1}$ and $EndPoint_{B2}$ and L_{b2} is the line between $EndPoint_{B3}$ and $intersection_C$.

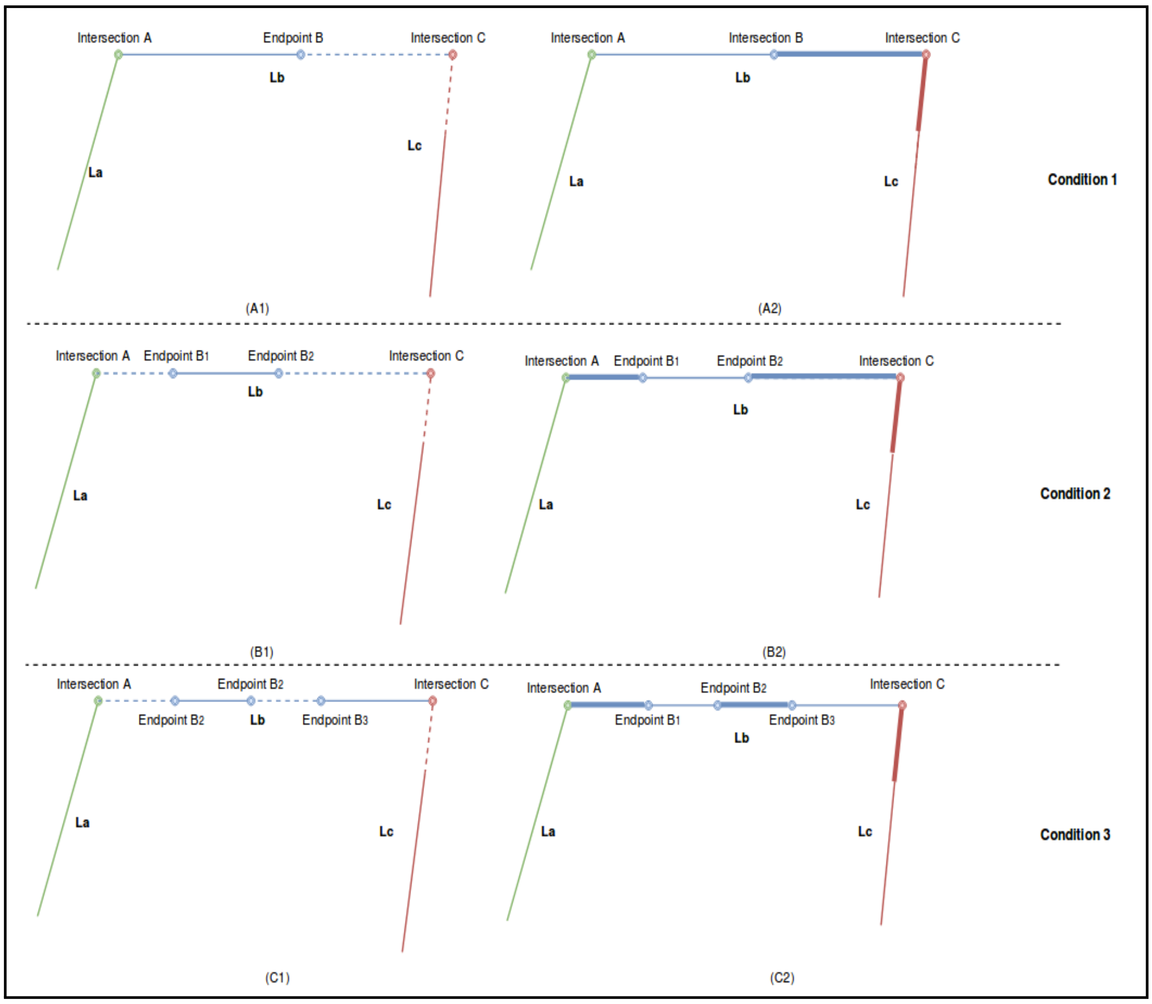


Figure 3.20: The description of occlusion in condition 2.

First of all, we need to confirm that L_{b1} and L_{b2} have the similar direction though they are from different edge line groups. Then, we use either of L_{b1} and L_{b2} to check their intersection on L_a and L_c. Assuming we are using L_{b1}, after the intersection detection. We will connect the $intersection_A$ with $EndPoint_{B1}$, $EndPoint_{B2}$ with $intersection_C$, $intersection_A$ with $EndPoint_{B2}$, $EndPoint_{B1}$ with $intersection_C$

and $intersection_A$ with $intersection_C$ as described as dashed lines in Figure C1. Their pixel length and direction will be calculated and then saved to a line segment vector.

For each line segment, we generate 6 lines between intersection points, including the line segment itself. For the other line segment in the same direction, we do the same process and store the line segments to line segment vector. Finally, we will connect two line segments and generate 4 lines: a line between $EndPoint_{B1}$ and $EndPoint_{B3}$, a line between $EndPoint_{B2}$ and $EndPoint_{B3}$, a line between $EndPoint_{B2}$ and $intersection_C$ and a line between $EndPoint_{B1}$ and $intersection_C$, and save them to segment vector as well.

In this segment vector, we find its maximum length of vector element firstly, and then calculate its average direction. The line segment which is longer than 60 percent of maximum length and closest to average direction will be considered as the feature line on this edge. If the filtered line feature is not from $intersection_A$ to $intersection_C$, we will find the endpoints of this line feature and automatically complete the whole line by connecting the non-intersection point to intersection point. Though this approach may lead to small direction difference between two connected line segments, our line clustering process and line direction check process will handle this line segment, which may consist of multiple segments and is very close to a straight line.

The line is determined in this sub-category if one of its endpoints connects to one line and its intersection on its vanishing direction is outside the image, or it only has intersection point in one vanishing direction, as shown in figure 3.21. In this condition, we extend the line from the corner point to the other point until it reaches the boundary. For example, assuming we are in condition 2 described in figure 3.21. Now we have two line segments L_{b1} and L_{b2} on L_b. More specifically, line L_{b1} is extended from $intersection_A$ to image boundary with direction $intersection_A$ to $EndPoint_A$. $intersection_C$ is connected with $intersection_A$ and $EndPoint_A$ respectively, so that three lines are obtained for each segment, including L_{b1} itself.

We then repeat this process and add all the line segments to line segment vector. Finally, we create the line segment group from L_{b1} and L_{b2} with the similar way in sub-category 2 and add them to line segment vector as well. The maximum length and the average direction are calculated among the elements in this group. We use the line which is closest to average direction and its length is larger than 60 percent of maximum length. If this line is not connecting L_a and image boundary, we will automatically complete this line with the suitable line segment inside the

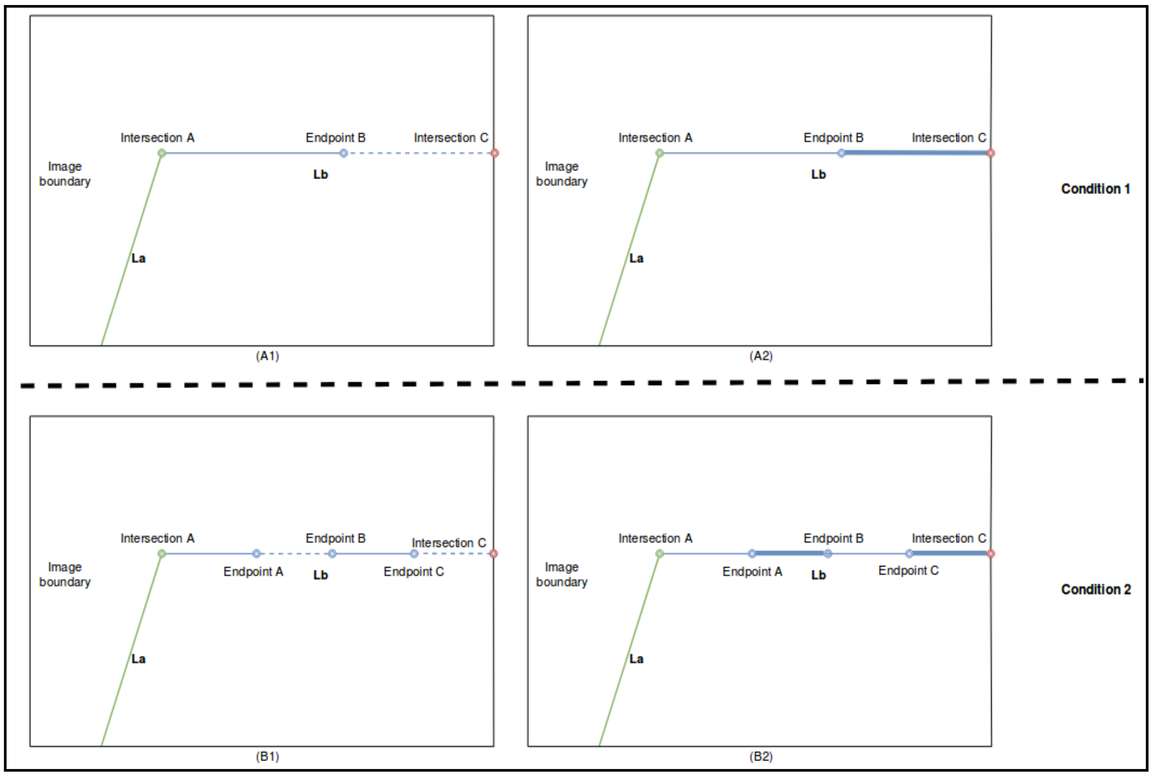


Figure 3.21: The description of occlusion in condition 3.

line segment group.

Finally, we use the contour extraction result again to filter out the outliers, such as the line that is crossing the table and drawn in purple colour in figure 3.18. For the detected edge lines, if more than 60 percent of its points is not near to extracted contour, this line will be considered as false detection and deleted. The final output are described in figure 3.22. Moreover, figure 3.23 presents the extraction results for more complex scenes.

We verify our algorithm in other datasets as well. In $ScanNet - 0705$ Dataset (Dai et al., 2017), we test our algorithm to detect the contour of table surface, white board and TV, and figure 3.24, figure 3.25, figure 3.26 have shown the results respectively. Experiment results show that our algorithm can detect the external and optimal edges of different objects, and recover their occluded corner points and edges in normal quality images. We also compare our algorithm and LSD detector in ICL-NUIM dataset (Handa et al., 2014) in figure 3.27. By combining multiple detectors, our method ignores the inside textures and always detect the outermost edge features. Moreover, our algorithm has a much better performance when the image quality is higher.

Figure 3.28 and figure 3.29 are two failure cases of our algorithm. Figure 3.28 (A) and (B) is generated because of false Deep Learning segmentation result. In our algorithm, if if more than 20 percent of pixel are correctly marked region, it cannot detect the contour of target object. Figure 3.28 (C) and (D) is a comparison when Deep Learning segmentation result is better but still not accurate. In our algorithm, if more than 20 *percent* of pixels in object region is correctly marked, it can locate the object and detect its contour. Note that 20 *percent* threshold value is adjustable based on segmentation accuracy. figure 3.29 is another failure case when the image has a strong illumination and motion blur, and the object boundary is two similar to background. In this situation, all three edge detectors cannot find accurate features, which makes it difficult for our algorithm to cluster and find optimum.

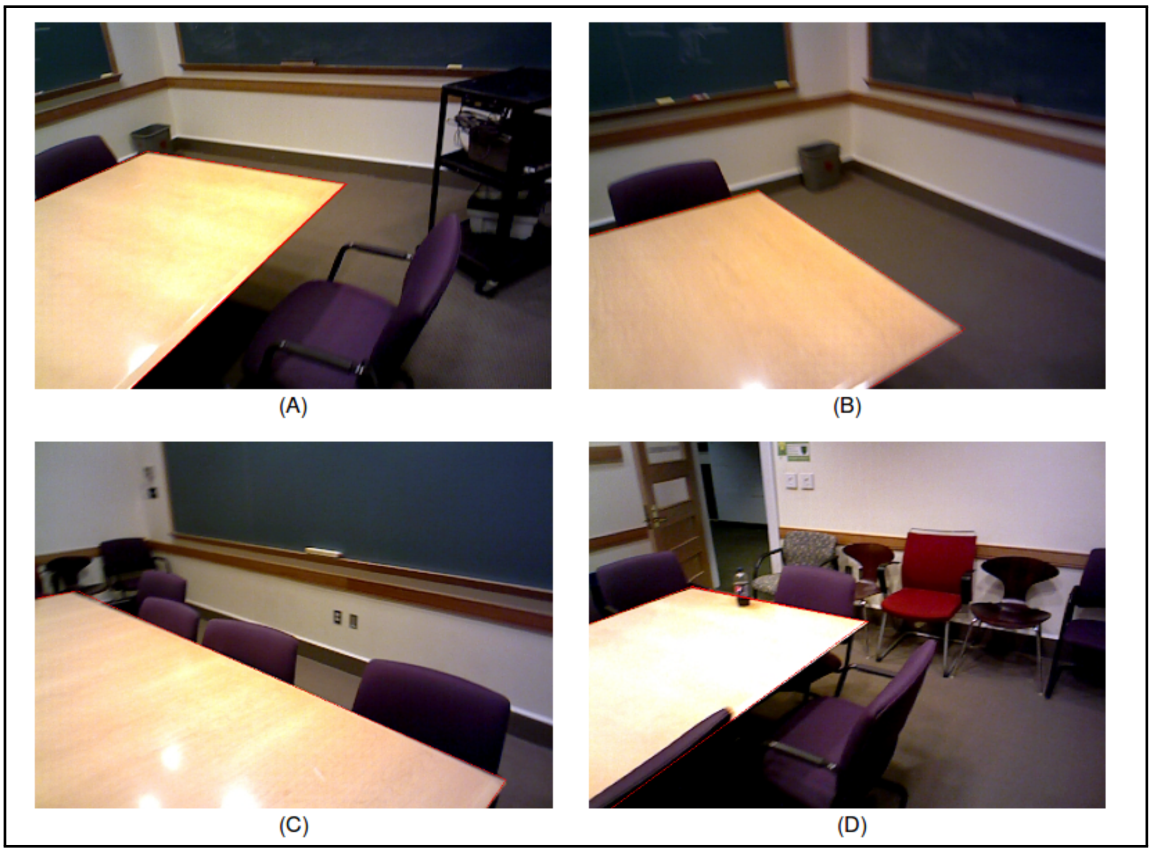


Figure 3.22: The description of occlusion in condition 3.

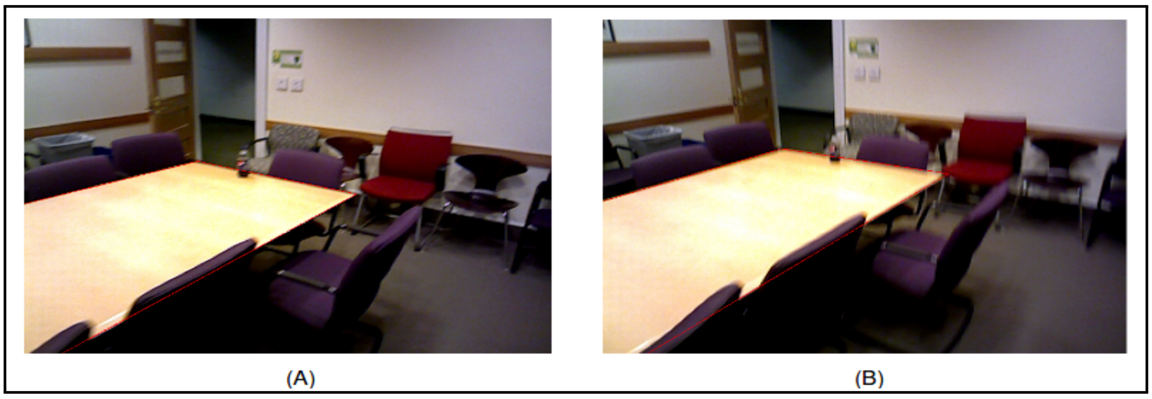


Figure 3.23: The description of occlusion in condition 3.

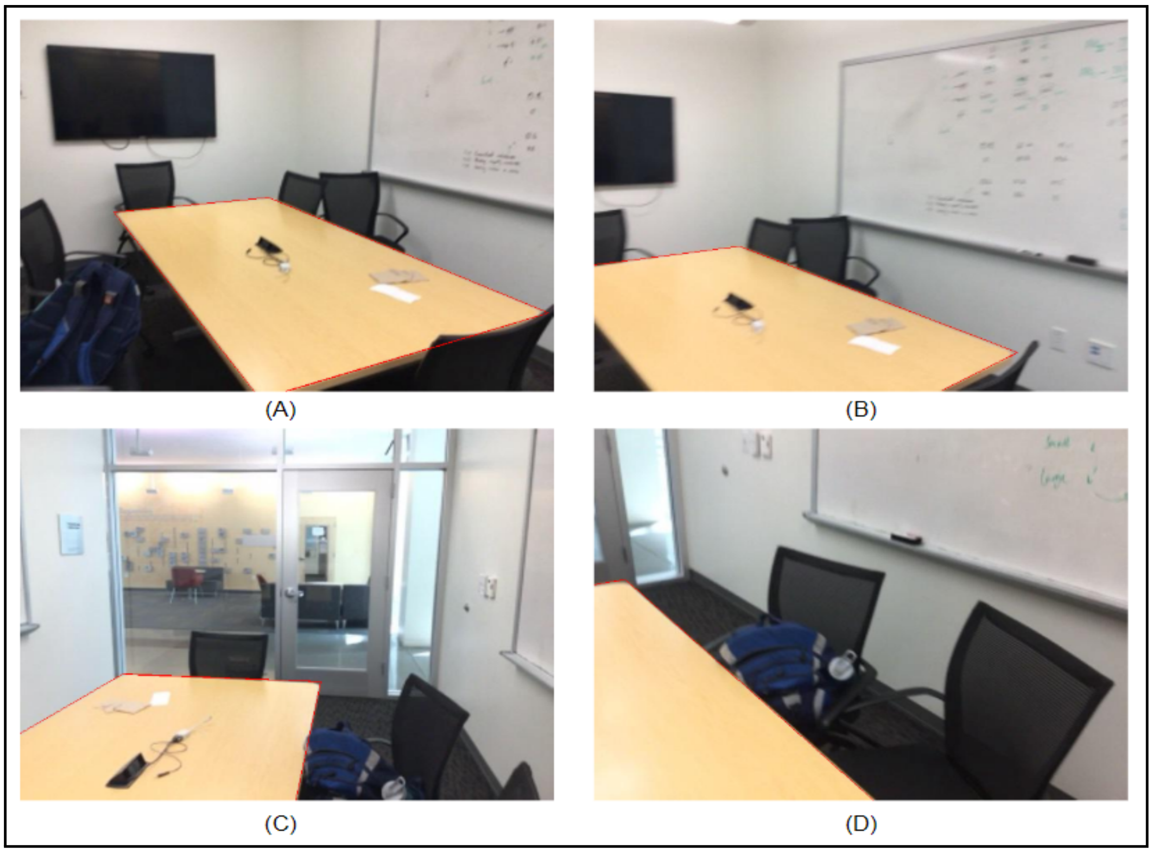


Figure 3.24: Contour detection on table surface in Scannet − 0705 *dataset (Dai et al., 2017). In (A), the bottom edge and the bottom right corner of table are occluded by the chair. Experimental result shows our algorithm can predict the occluded corner and restore the occluded contour. Image (B) indicates our algorithm will not retain the edge segment if both of its endpoints are not appearing in current perspective. (C) describes the contour detection result when three edges are visible and one of these edges is occluded. (D) shows the result when one long edge (detected by Hough Line Transform) and one small edge (detected by LSD algorithm) is visible.*

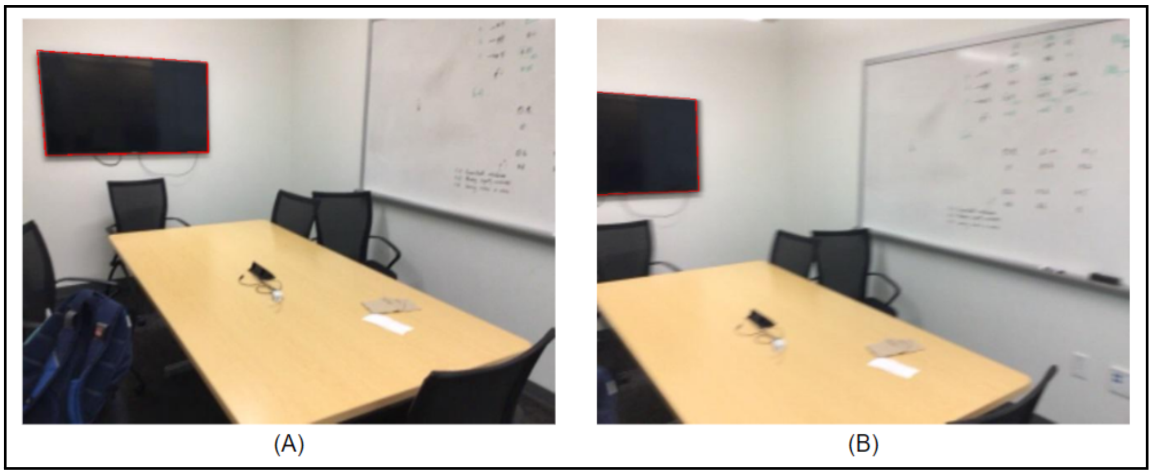


Figure 3.25: Contour detection on TV screen. (A) shows the result when entire TV is visible. (B) shows result when only part of TV is visible. Our algorithm always detect the external contour exactly when the object is semantically detected

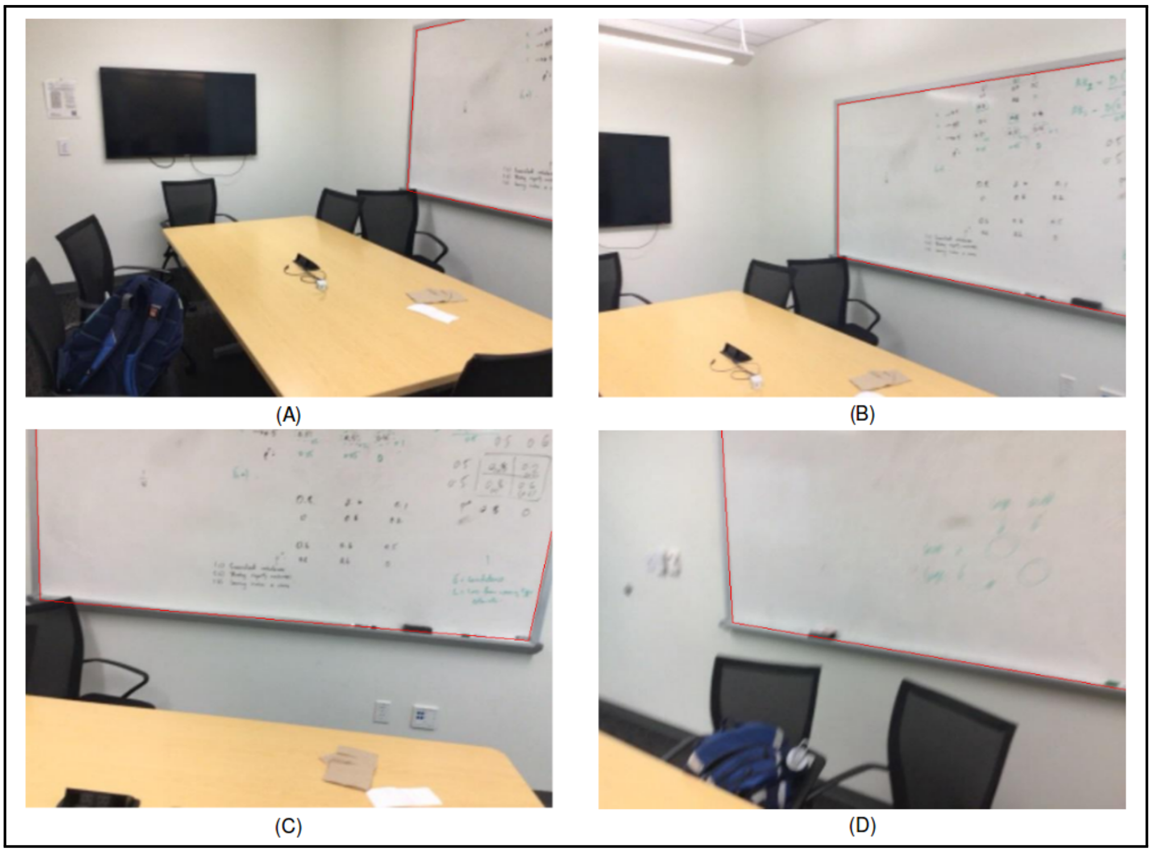


Figure 3.26: Contour detection on Whiteboard in the same scene. Image (A), (B) and (C) show that our algorithm can estimate and recover the occluded corner (bottom left) of whiteboard. (B) and (C) describe our algorithm can reconstruct the edge of whiteboard (bottom) occluded by whiteboard eraser. (D) is the experiment result of another whiteboard in the same scene.

3.4 Depth completion for 3D contour model reconstruction

After extraction the contour line of a table, we try to recovery the 3D plane model from the contour data, RGB data and raw depth data. Due to the serious inaccuracy of captured raw depth data, it is hard to recovery a precise 3D model of plane, or the 3D model of plane's contour. In this section, we introduce a novel plane edge model completion algorithm to recovery a complete 3D contour model of a plane. We only demonstrate the contour model instead of the plane model for proving the efficiency since contour data is more important for the further process in our system.

To overcome the inaccuracy problem of raw depth data, We use the normal vector of the plane and the points in the central area of plane which is more accurate to estimate the 3D coordinates of the edge points. Its workflow could be illustrated as figure 3.30, in which The content in blue frame are the inputs from earlier processes and the yellow frames are the main processes.

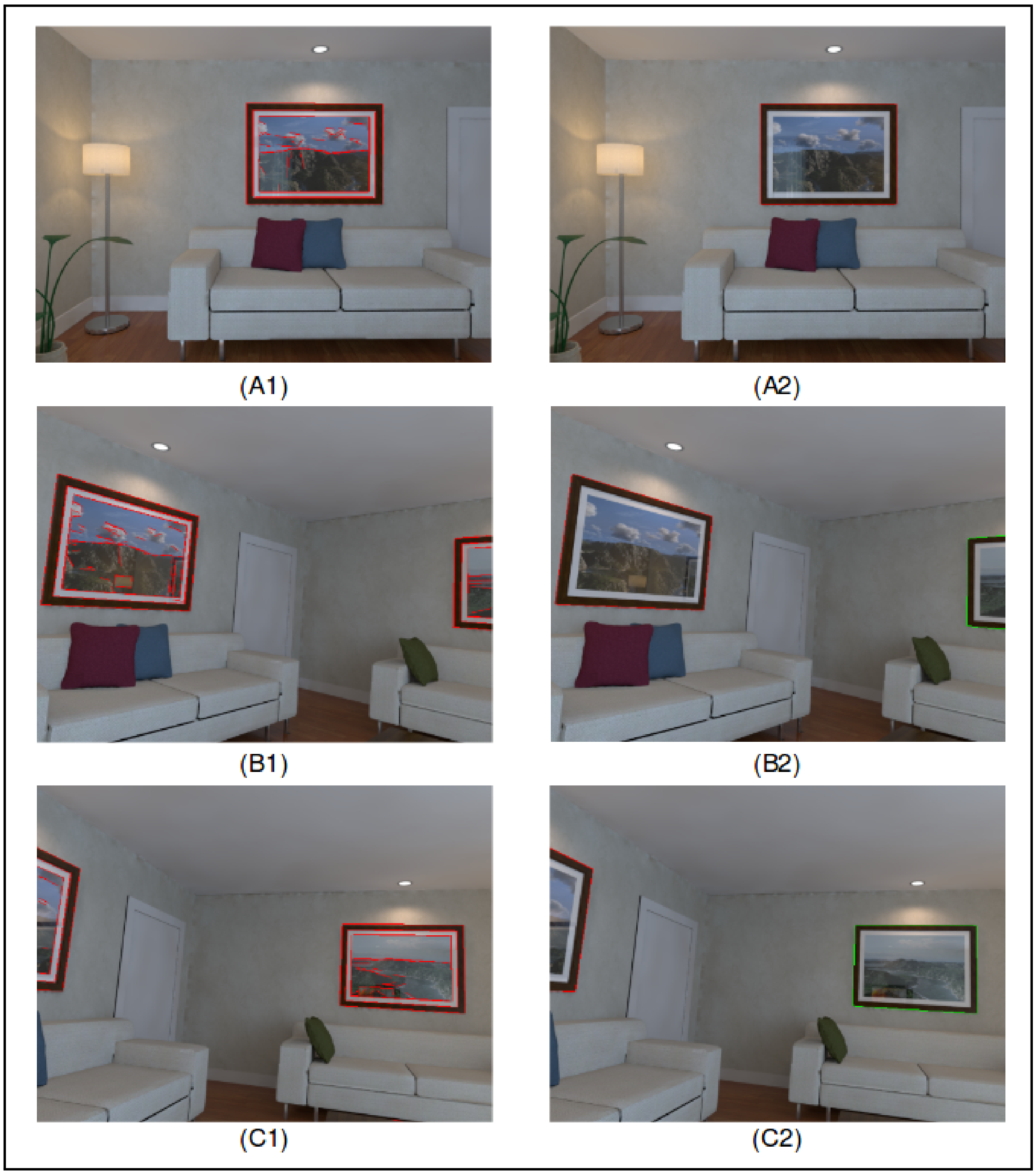


Figure 3.27: A comparison between our algorithm and LSD algorithm on contour detection of large photos in ICL-NUIM dataset. (A1), (B1) and (C1) are the results from LSD algorithm. (A2), (B2) and (C2) are our results. By combining LSD, Hough Transform and Contour Detector, our algorithm always detect the external edges and makes every detected feature meaningful.

The plane edge model completion algorithm includes three main processes:

1. Assuming we get the 3D plane model with raw depth and contour lines as the prior acquired data. In this process, we firstly extend the contour point toward the Centroid of plane for 8 pixel length, and these points are clustered in $Points_{Edge}$, the rest points belong to landmark plane will be classified as $Points_{Inside}$.

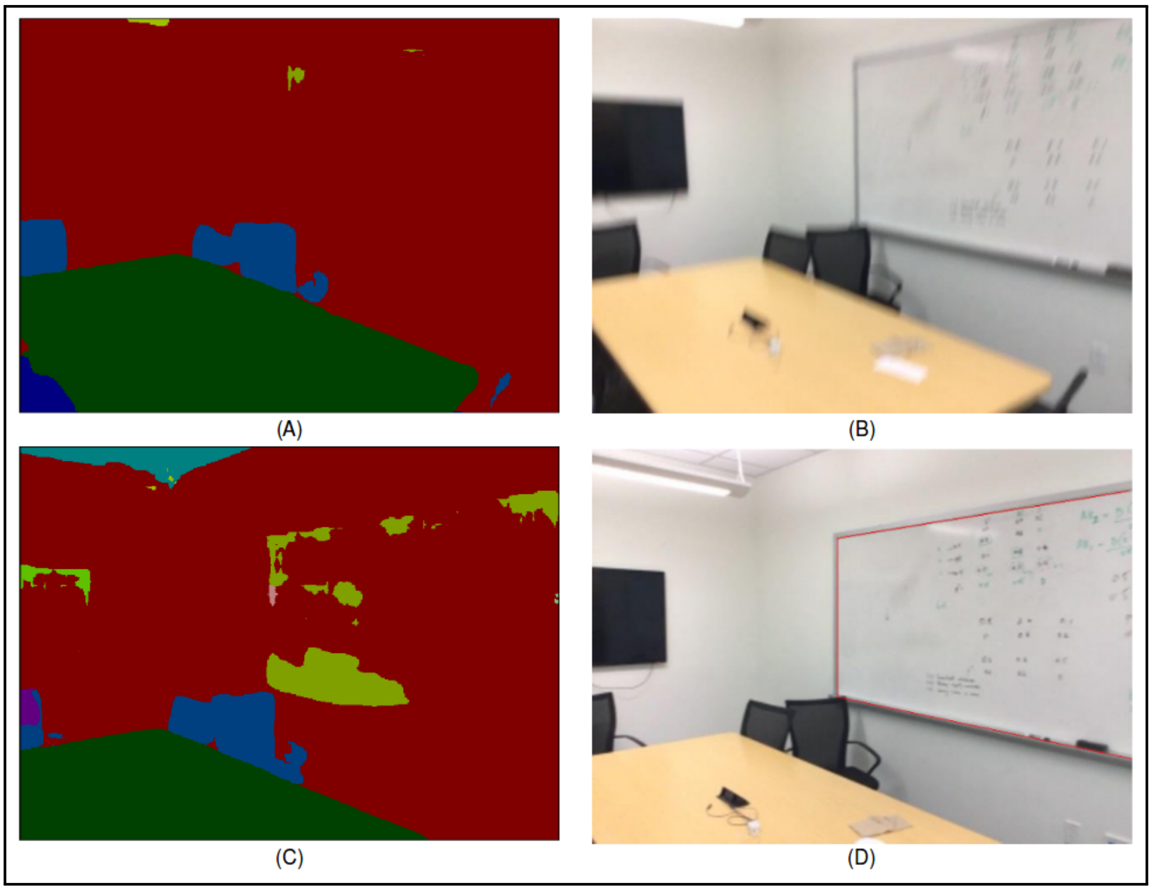


Figure 3.28: Detection error when segmentation result is bad. (A) and (C) are detection results, (B) and (D) are corresponding detection results respectively. In case (A) and (B), since deep learning model cannot detect the whiteboard, our algorithm cannot find the label region and detect edges. In case (C) and (D), since there are more than 20 percent of plane region has been marked as 'whiteboard' by deep learning model, our algorithm works in this situation

2. Recovering the inside plane model with points in $Points_{Inside}$ and raw depth, and then estimate its normal vector with PCL library. Since the inside points in a plane will be more accurate than the edge points, estimating normal vector with $Points_{Inside}$ derives a more accurate normal vector result.

3. In this step, we will estimate the depth data of every contour points with normal vector equation constraints and Ceres lib. We choose two random point P_1 and P_2 in $Points_{Inside}$ and one point P_3 in $Points_{Edge}$, and then we build the normal vector equation:

$$a\vec{x} + b\vec{y} + c\vec{z} + d = 0; \tag{3.1}$$

where a, b, c and d are derived by normal vector extraction, $\vec{x}$, $\vec{y}$ and $\vec{z}$ could be derived by ($P_3.x$ - $P_1.x$), ($P_3.y$ - $P_1.y$), ($P_3.z$ - $P_1.z$) or ($P_3.x$ - $P_2.x$), ($P_3.y$ - $P_2.y$),($P_3.z$ - $P_2.z$). Among them, the x, y and z of P_1 and P_2 are derived

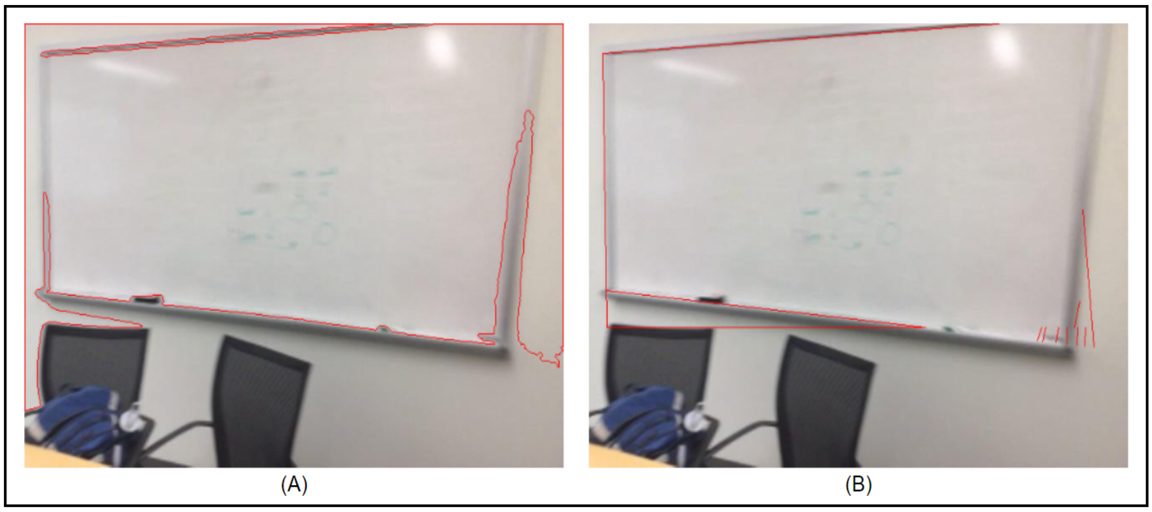


Figure 3.29: Detection error with high illumination, strong motion blur and object boundary is not clear. Image (A) is the result from Contour Detector, and Image (B) is our result. In this case, Contour Detector, LSD and Hough Transform detector provide too many imprecise features, and it is difficult for algorithm to cluster the features and find optimum.

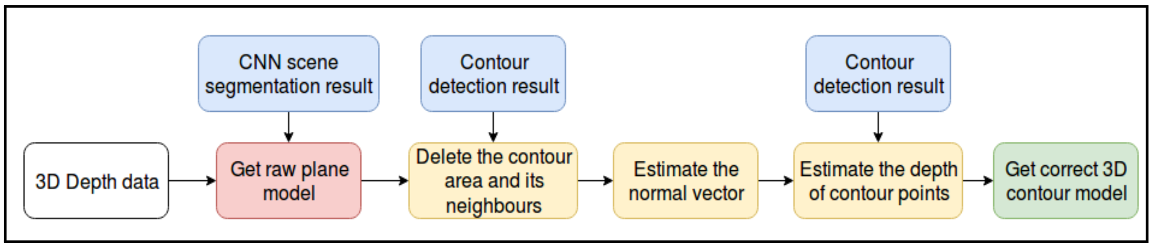

CNN scene segmentation result
Contour detection result
Contour detection result
3D Depth data
Get raw plane model
Delete the contour area and its neighbours
Estimate the normal vector
Estimate the depth of contour points
Get correct 3D contour model

Figure 3.30: The flowchart of the proposed depth completion process.

directly, $P_3.x$ and $P_3.y$ could be derived by equations below:

$$P_3.z = depth/camera_{scale}; \tag{3.2}$$

$$P_3.x = (n - camera_{cx}) * p_3.z/camera_{fx}; \tag{3.3}$$

$$P_3.y = (m - camera_{cy}) * p_3.z/camera_{fy}; \tag{3.4}$$

where m and n are the pixel position of P_3, $camera_{scale}$, $camera_{cx}$, $camera_{cy}$, $camera_{fx}$ and $camera_{fy}$ are camera intrinsic parameters.

Since the normal vector of a plane has two directions, we use the Ceres lib to optimise this problem: inserting these values in normal vector formula and building two residual equations. Then solving these equations to get depth $P_3.z$ of each contour point through Ceres optimisation lib.

$$residual[0] = a * (P_3.x - P_1.x) + b * (P_3.y - P_1.y) + c * (P_3.z - P_1.z) + d \tag{3.5}$$

$$residual[1] = a * (P_3.x - P_2.x) + b * (P_3.y - P_2.y) + c * (P_3.z - P_2.z) + d \tag{3.6}$$

After obtaining the estimate depth value of each point in edge region, we can obtain its 3D model. The proposed algorithm can restore the 3D information of plane in the scene in the most complete way with only ordinary facilities. It has two important meanings: (i) it can provide the precision requirement for projection model of SAR, which is not satisfied by any current visual reconstruction algorithm; (ii) this precise 3D edge model can play the role of a special sign that exists in the scene. By tracking the attitude of this 3D model at different viewing angles, we can locate the camera's position and perspective. This also provides a new strategy for the existence of dense mapping theories.

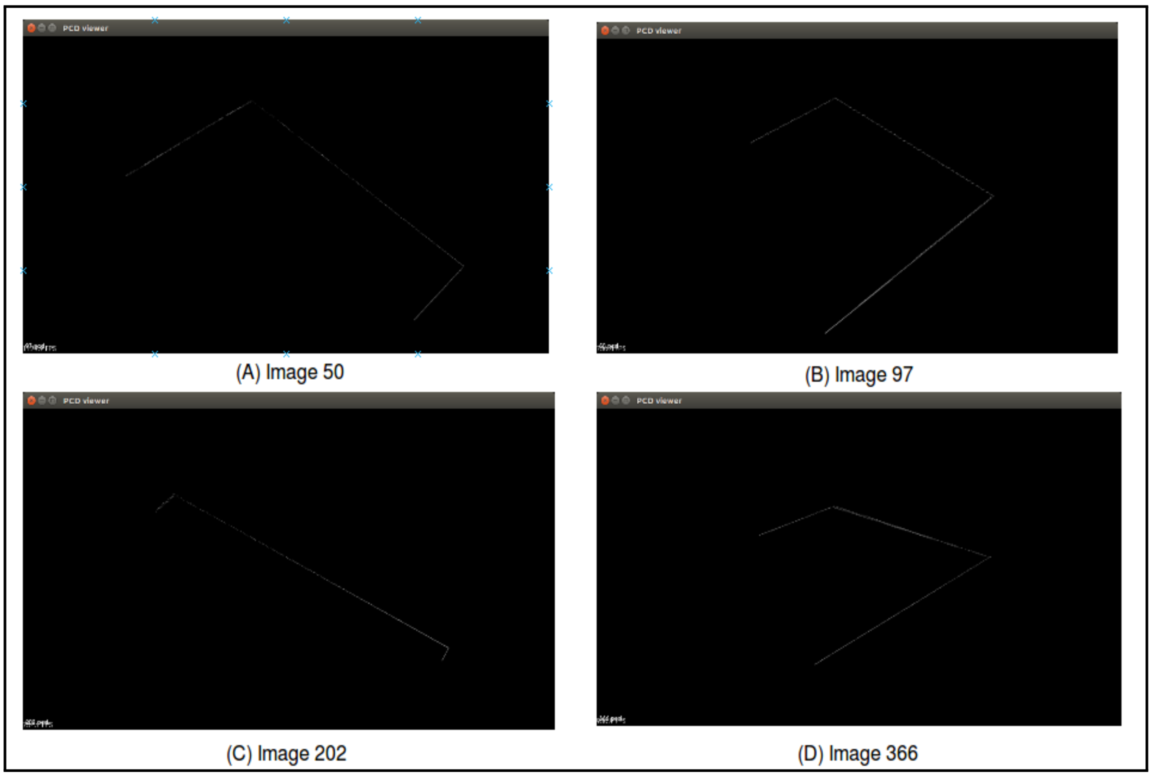


Figure 3.31: The recovery 3D contour model with depth completion.

3.5 summary

In this chapter, we have introduced a novel contour extraction and contour modelling algorithm to optimise the imprecise raw data and small occlusion problems. Our algorithm has solved three fundamental problems for recent visual tracking or visual mapping applications:

1. Contribution 1: Contour feature extraction in normal quality image. In real application, the captured image cannot always be high-resolution or high-

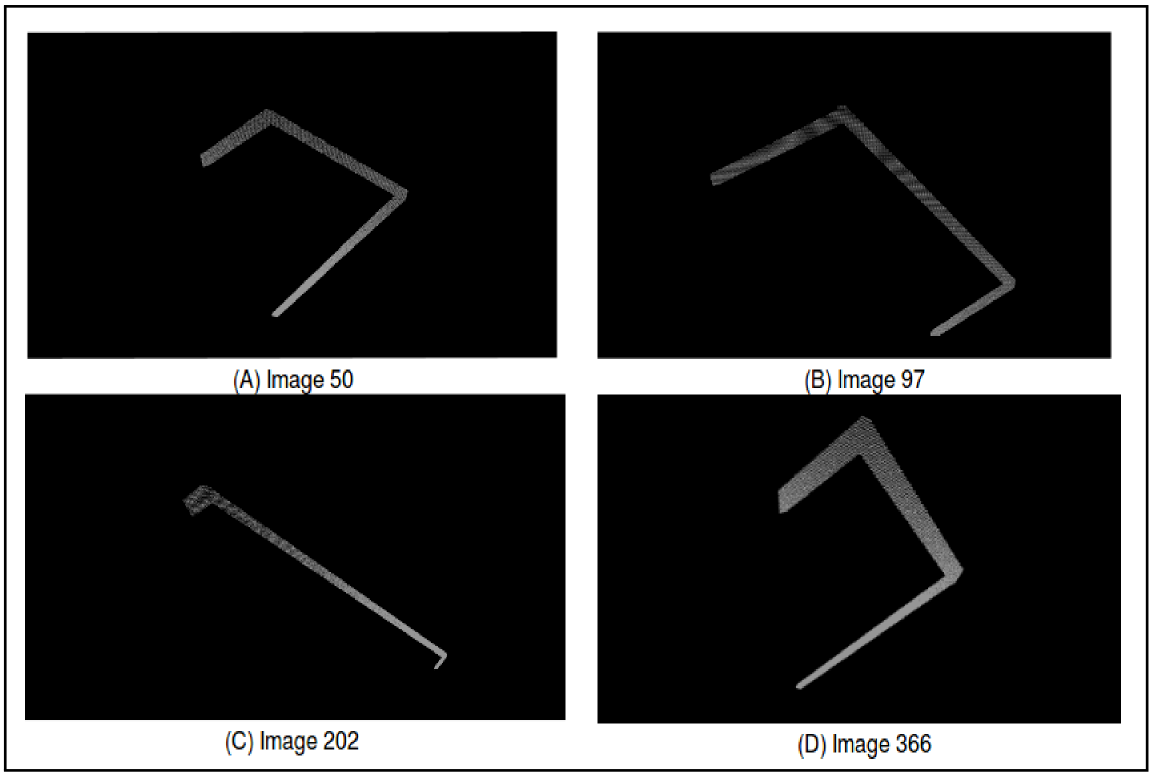


Figure 3.32: The recovery 3D contour and its extending edge with depth completion.

quality due to the quality of consumer-level sensors and motion blur problem. In this situation, it is impractical to extract the edges of a plane are with single contour extractor or contour extractor. Therefore, we have introduced our line extraction pipeline for planar area by combining three popular edge extractors. By analysing the characteristics and disadvantages of each extractor, we find edge features in image from a basic to richer, and finally filtering process. Compare with state-of-art visual tracking or visual mapping algorithm which uses only one extractor, like LSD extractor and Hough line extractor, our algorithm is available to extract the optimal edge features, and automatically choose the best among multiple line features to present each edge of planar area.

2. Contribution 2: We have fixed the problem of line extraction with small occlusion. While camera is moving around the scene, the appearance of an object might be changed due the occlusion in different camera's perspective. A common method to solve this problem is based on the 3D prior knowledge of an object, like the scale of a plane or the 3D model of a plane. However, generate these kind of information is very complex and time consuming. Alternatively, others get rid of this problem by analysing consecutive frames, but this method is highly depending on the quality of image, and it requires the scene has enough visual features outside the occlusion area. As an innovation,

we propose a plane extraction algorithm based on CNN scene parsing model. This algorithm does not depend on the 3D model and continuous frames that were mastered in priority. In the case of only one frame of image, the edge line segment of the occluded plane can be successfully extracted. However, our algorithm also has a disadvantage. When the occlusion object occupies most of the plane, our algorithm will produce incorrect line completion results due to the lack of label information or sufficient line features. In fact, this problem is very difficult to solve without using the prior 3D knowledge.

3. Contribution 3: Our algorithm is using the raw data instead of high-quality data, such as raw CNN-scene parsing result, raw RGB image and raw Depth data. In practical applications, the information captured by the system is difficult to maintain its accuracy, and the precise reconstruction of the plane through normal-quality information is another innovation point of our algorithm. Through the correlation and assistance between the information, our algorithm minimises the impact of each information on the test results due to inaccuracy, but only suitable to offline visual mapping.

There are two main limitations in proposed contour detection algorithm. On the one hand, since we correct the occluded edges through connecting the endpoints of edge segments, our algorithm is more applicable on planar object with straight edges, like surface of table and whiteboard. On the other hand, when the boundary of object is very similar to its background and image is strongly blurred, it is difficult for our algorithm to detect the accurate contour feature. In the future, we may try to use image filtering algorithm to improve the image quality firstly to make our algorithm more practical.

4

Semantic Visual Mapping with Planes

4.1 Introduction

This chapter introduces our plane-based visual mapping algorithm, i.e. a dense visual mapping approach. It aims to reconstruct the scene based on an empty planar area in a feature-less scene by tracking its contour. Unlike the existing approaches, our system has four innovations.

Firstly, the target we are tracking is a medium-size or a large-size empty plane inside indoor scene, such as the surface of table, projection screen and post on the wall. During mapping, this plane could be partially appeared in each frame. Secondly, our algorithm does not require prior knowledge of plane model, which can be completed frame by frame during tracking. Thirdly, the frame to frame matching strategy is only using the 3D contour data of a landmark plane, and the utilized ICP algorithm is specified to point-to-point ICP algorithm. Lastly, a loop closure optimization algorithm is proposed based on contour of landmark plane. The optimization process triggers when the entire contour has been viewed, and then corrects the poses of all previous frames. Compared with global loop closure optimization in existing visual mapping algorithms, this approach simplifies the data recording process and improves practicability.

Visual mapping has three main requirements: the processing speed, mapping accuracy and robustness, which are often not compatible. The algorithms that can be applied to different scenarios (feature-less environment and rich-texture environment) will inevitably increase the processing time. For SAR applications, it requires an accurate and robust mapping algorithm. Therefore, the proposed visual reconstruction method sacrifices the computation time and greatly improves the accuracy and robustness of visual tracking and mapping.

To speed up the system, the proposed algorithm is designed upon ORB-SLAM when some solid visual features exist in the scene. In this situation, the visual mapping process works as a parallel process and only tracks the key-frames generated by ORB-SLAM. When the ORB-SLAM fails in feature-less environment, our mapping process can correct the trajectory through tracking the contour of a detected landmark plane consistently.

When the number of visual features in the scene are insufficient, our mapping algorithm can also be used as an independent visual mapping system. The only difference between two systems is that the independent mapping algorithm tracks camera frame by frame instead of using key-frame and pose graph algorithm in order to maintain tracking accuracy. This chapter will present the details of each important step of our mapping algorithm.

4.2 System Structure

In the beginning, our system will run the ORB-SLAM algorithm to obtain a sequence of key-frames. In this experiment, the surface of a table is labeled as the reference area, and we use 'DeepLabV3+' to label the image semantically, which is described in Chapter 3.1.2. After the image segmentation, the contour of each key-frame will be extracted with proposed contour extraction pipeline described in Chapter 3. A depth compensation algorithm is used to obtain a precise 3D contour model. After this, the system will enter the mapping loop and use a novel optimised contour-to-contour dense ICP matching algorithm to track camera's trajectory. And an optimisation theory based on global ref model and ray-casting is applied to correct the pose estimation result, which are described in Chapter 4.4. The registration performance is determined by the matching score derived by ICP algorithm provided in PCL library (Rusu and Cousins, 2011). We set an acceptable threshold equals to 0.00015 in this experiment. If the score is less than 0.00015, it means two point clouds are correctly matched. And the pose and part of 3D data of new frame will be updated to global poses and global ref model respectively. If the score is larger than 0.00015, the system will decide this frame is the missed match frame. Note that the threshold value '0.00015' is specific to current experiment, and this value is adjustable in other scenarios.

When two consecutive missed match occurs, the system will enter the Lost Mode. In this mode, the global ref model and global pose will not be updated, but new images will keep inserting into the mapping loop to reconstruct the fragment model. When lost tracking happens again, the system will end this mode and enter the lost recovery mode. In the lost recovery mode, the generated fragment in Lost mode will be matched with the last updated ref model. When the fragment is matched, the poses of each frame inside this fragment will be updated to global poses. Finally, the system will return to the Normal mode and continue the mapping process.

The following part in this chapter will introduce the details of each important step of the proposed system apart from the normal ORB-SLAM. In addition, the advantages and creative ideas will be compared with existing approaches and concluded with experimental results. The following content starts with proposed frame-frame registration algorithm. Then, the optimization theory based on global ref model and ray-casting is introduced in the next subsection. The lost recovery method and loop closure detection algorithm are presented in Section 4.5 and Section 4.6 respectively. Finally, the creation and benefits of the proposed mapping algorithm will be summarised.

4.3 Contour-to-Contour 3D Regularisation

Inspired by feature-based SLAM algorithms and recent SLAM algorithms with Manhattan Structure (Coughlan and Yuille, 1999). We estimate the pose transformation through tracking the contour feature in 3D space instead of tracking planes or entire scene. By matching two contour models of same planar object, this method can correctly match two scenes.

Different to point-plane or plane-plane ICP algorithm in existing plane-based SLAM algorithms, proposed algorithm uses point-to-point ICP algorithm (Rusu and Cousins, 2011) to register two contour models, which is described in Chapter 2.8.4. Compare with plane-plane match and point-plane match, our algorithm has less computation and faster speed. And ICP registration with less points is easier to avoid local minimum problem. One main limitation of our method is it relies on the accuracy of detected contour feature. We introduce our contour extraction method in Chapter 3 to solve this problem, and we have evaluated our algorithm in a higher-quality RGB-D dataset in Section 4.6. In the following subsections, the contour-to-contour matching performance will be displayed in the first place. Then, a detailed deficiencies and practical limitations for the proposed contour-to-contour registration algorithm will be summarised.

4.3.1 contour-contour registration

In ICP registration, more complexity of point-cloud brings more accurate registration result, which however is easy to fall into local minimum error at the same time. With the inspiration of using Manhattan Space structure for registration the planar area in corridor, the proposed contour-to-contour registration algorithm has suggested to use the structure of 2D plane as an alternative solution for small mapping scenes.

Differing from the normal ICP algorithm, which aims at finding the transformation between corresponding point pairs, contour-to-contour registration focuses more on computing the transformation of plane 3D structure, though it is achieved by matching 3D points. Similar to feature-based tracking algorithms, each contour could be considered as a very important feature of landmark plane. Instead of describing each contour point with 2D descriptors, the purposed registration method have given the contour in each camera's view a 3D spacial feature. By tracking the pose of contour in 3D, the camera's movement could be easily derived. In other words, the registration using the edge of two plane clouds could be treated as an densely alternative approach to 2D line-line matching. If the edge of two planes are correctly matched, their planes are precisely registered.

In this experiment, the normal point-to-point ICP algorithm is used as the registration method. The contour point clouds are extracted in the data fusion step. For ICP parameters, the iteration time equals to 1500, and the optimum is set to equal to 0. Though this iteration time is very large, the processing time is still much faster than normal approach because of using less points. Figure 4.1 has shown the difference of performance between the scene-scene ICP registration and the proposed registration approach. Figure 4.2 and figure 4.3 show some contour-to-contour registration results. In figure 4.2(a) and figure 4.3(a), the gray line are the contour from first frame, and the red line are the contour from second frame. It is very clear to find contours in both cases are all perfectly matched up.

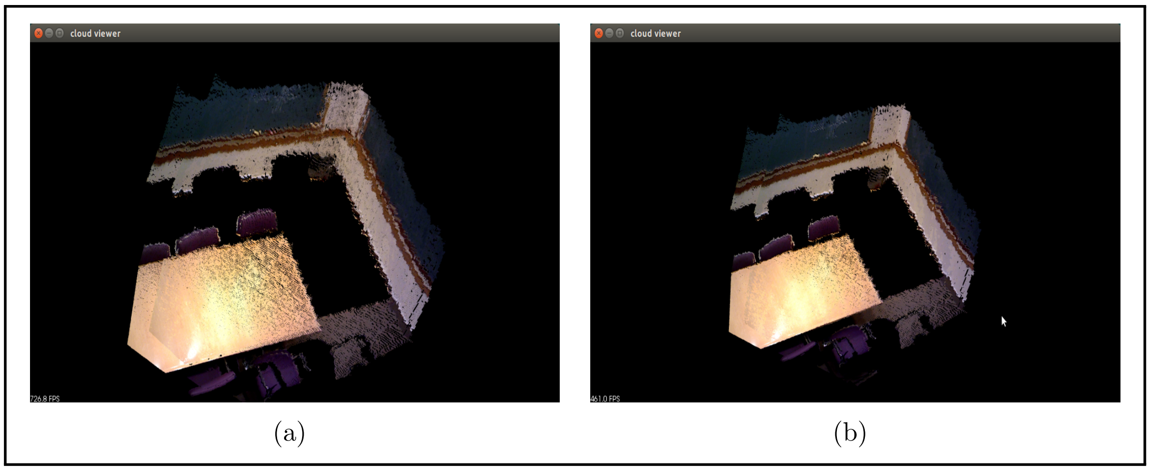


Figure 4.1: (a) registration result based on the whole scene model, and derived ICP matching score is 0.00673. (b)registration result based on the contour model, and derived ICP matching score is 0.0001093. Since the score is derived by calculating the distances between matched points in two clouds, our contour-contour registration method outperforms scene-scene match because of lower derived ICP score.

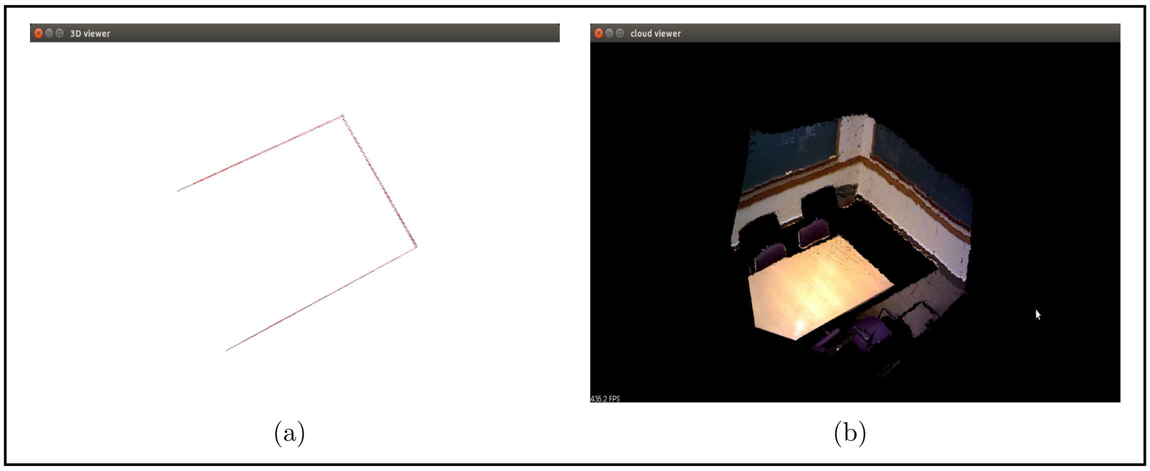


Figure 4.2: (a)3D contour model matching result, the error = 0.0000032. (b) Scene Model after contour-to-contour registration

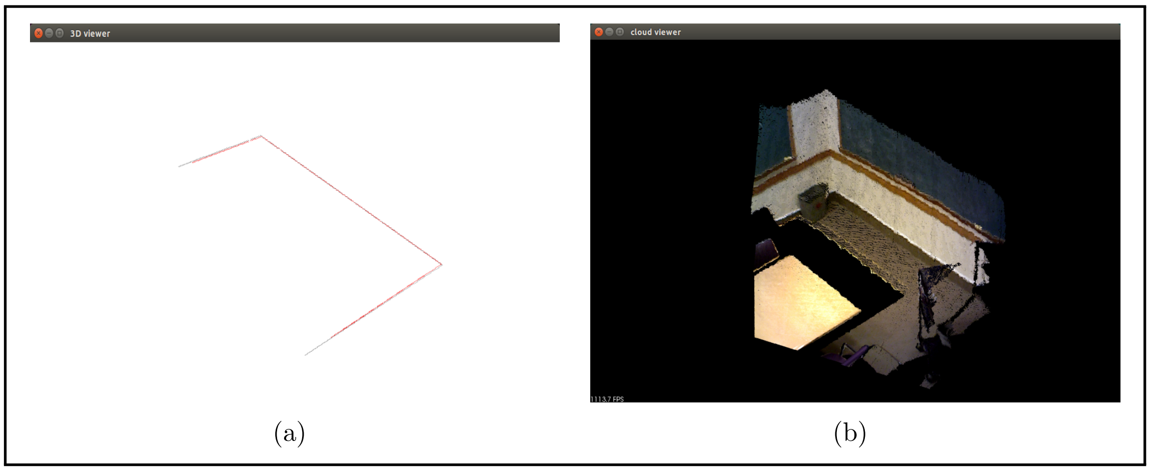


Figure 4.3: a)3D contour model matching result, the error = 0.0000018. (b) Scene Model after contour-to-contour registration

4.3.2 Limitations of contour-to-contour Registration algorithm

The last experiment shows that using purposed contour-to-contour algorithm provides better registration result. However, this rough matching approach has two inevitable problems: (i) The matching score derived from direct frame-frame ICP process is not always reliable when two models have big difference. As this score is determined by the Euclidean distance between points in two models, its score will be high even when the matching result is acceptable. (ii) Similarly, when the difference between two models are large, the proposed registration process is easy to fall into local minimum problem. So, an optimised approach of deciding the matching performance is required.

Figure 4.4 shows two 'mismatch' cases decided by computer. In figure 4.4(a), the gray line describes the contour in first image and red line describes the other. This image has shown that when two clouds have a big difference between each other, the ICP score is still high even if these two models are correctly matched. In figure 4.4(b) describes the false matching caused by local minimum. In this case, ICP has found the smallest matching score but two cloud are not correctly matched.

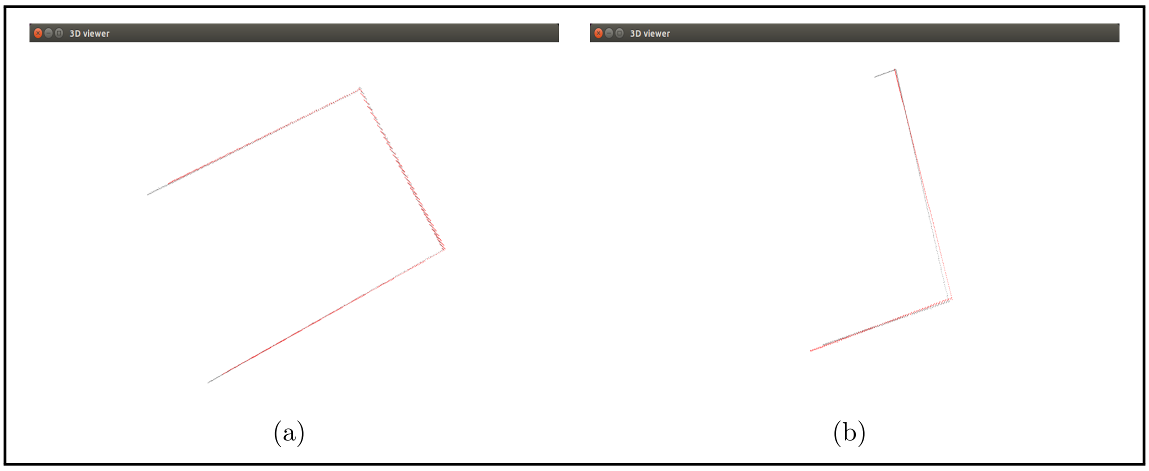


Figure 4.4: Two 'mismatch' cases. (a) its matching score is 0.000523665. (b) its matching score = 0.00034337.

Inspired by KinectFusion (Newcombe et al., 2011a), a novel and practical optimisation approach is introduced in Section 4.4.

4.4 Optimisation with ref model and Ray-Casting

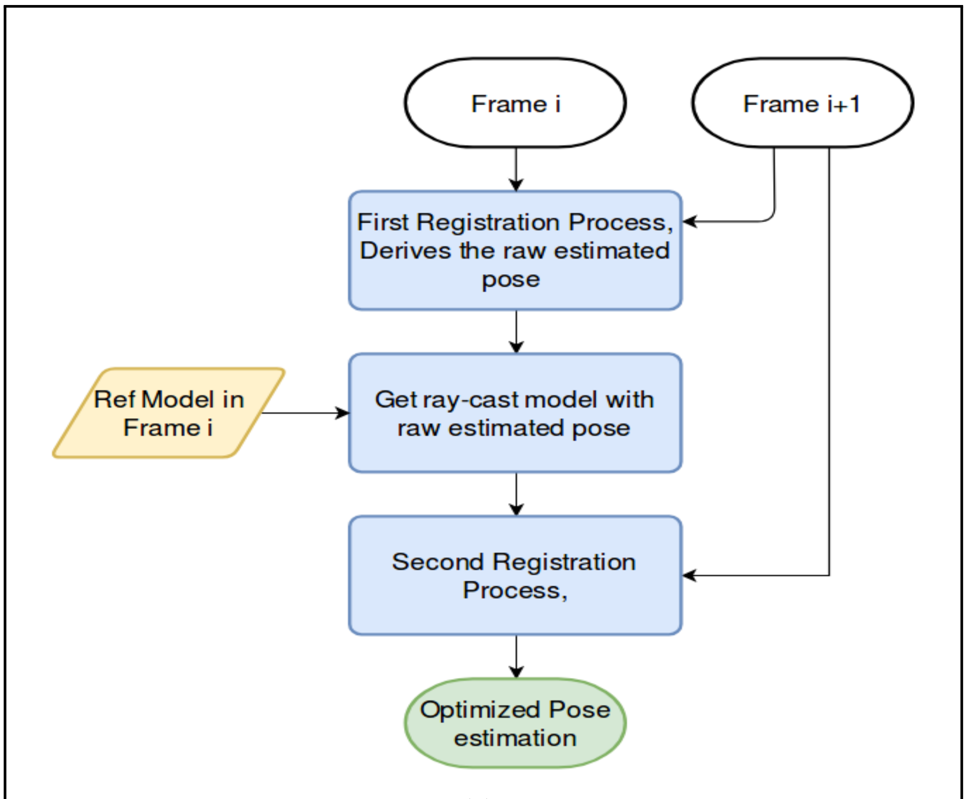
In this section, the detailed Optimisation of contour-to-contour registration is described. This optimisation strategy mainly includes four requirements:

- Firstly, the system needs a very accurate and trustworthy scoring method so that in the complete SLAM process, we can judge the accuracy of the match and determine the lost tracking based on this score.
- Then, as the algorithm we propose is to match the key frames through depth images instead of frame-by-frame matching, we need to truncate the two models before matching so that the two contour models will be similar for matching, which could optimise the local minimum problem.
- Thirdly, similar to the sparse mapping algorithm, the system needs a local loop closure detection algorithm so that it can construct an accurate and complete 3D model even if there is no global loop closure occurs.
- Finally, the system needs an applicable and accurate loss tracking algorithm for emergency needs, which will be introduced in the next section.

With the inspiration from KinectFusion, our mapping algorithm re-calibrate every estimated pose with global model and ray-casting theory. As an innovation, it introduces a ref model besides global model. The ref model is used as a global landmark and the global model describes the final output. During mapping, ref model only save a part of 3D contour points from inserted new frames, and is constantly updated as the camera is moving. In addition, the score derived from this process directly decides whether lost tracking happens or not.

4.4.1 A novel approach for computing matching score

The matching score provided by ICP is an important criteria of estimating the registration accuracy between two point clouds. However, direct frame-frame calculated score is not suitable for the proposed visual and mapping algorithm as the difference between two clouds and local minimum greatly affect its reliability. As a result, a re-calibration algorithm with ray-cast and ref model is introduced here to obtain a better registration performance and a more reliable matching score. Figure 4.5 shows its work flow.


Frame i
Frame i+1
First Registration Process, Derives the raw estimated pose
Ref Model in Frame i
Get ray-cast model with raw estimated pose
Second Registration Process,
Optimized Pose estimation

(a)

Figure 4.5: The flowchart of the proposed contour-contour registration process

Assume that an existing frame f^i, a ref model (this could be the model of the first frame) $\mathcal{M}^i_{ref}$, the corresponding camera pose P^i (this could be the identity pose), a new inserting frame f^{i+1} and its 3D Point Cloud $\mathcal{M}^{i+1}$ are available. In the first ICP process, a raw estimated pose of new inserted frame $P^{\hat{i+1}}$ and a matching score is derived. Because of the ICP limitations, this pose and score is not very accurate. Then, the proposed system conducts the ray-casting on the ref model with estimated $P^{\hat{i+1}}$ to obtain the estimated 3D model of f^{i+1}. This calculation process is described below.

$$\begin{bmatrix} m \\ n \\ 1 \end{bmatrix} = [CameraIntrinsicMatrix] * [Currentview]_{3x4} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.1}$$

$$[CameraIntrinsicMatrix] = \begin{bmatrix} camera_f x & 0 & camera_c x \\ 0 & camera_f y & camera_c y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

$$[Currentview]_{3x4} = \begin{bmatrix} r00 & r01 & r02 & t0 \\ r10 & r11 & r12 & t1 \\ r20 & r21 & r22 & t2 \end{bmatrix} \tag{4.3}$$

Note that figure 4.2 is the camera intrinsic matrix, which can be derived as priority. Figure 4.3 is based on the estimated pose, and all the parameters could be found in Pose Matrix $\hat{P_{i+1}}$. x, y and z are describing each 3D points in ray-cast model. m and n are the estimated pixel position. In figure 4.1, all the elements in the right of equation are known, so the estimated position on 2D image of each 3D point could be derived directly. To define which point are in current camera‘s view, the propose algorithm set the m should be inside $[0 - 640]$ and n is inside $[0 - 480]$. Thus, any points belong to ref model have the correct m and n could be considered as the viewed points of ref model in current camera’s pose.

Since the raw estimated pose $P^{\hat{i+1}}$ is not accurate, the ray-cast model $\mathcal{M}^{i+1}_{Ray}$ can not be as same as the new inserted model $\mathcal{M}^{i+1}$. By matching and computing the transformation between $\mathcal{M}^{i+1}_{Ray}$ and $\mathcal{M}^{i+1}$, an transformation T^{i+1}_{Ray} and an optimised matching score is computed. Then, the optimised pose of f^{i+1} could be derived by $P^{\hat{i+1}} * T^{i+1}_{Ray}.inverse()$. As we used a ray-cast point cloud similar to the second frame to calculate the ICP matching score, this score is much reliable than the previous RAW ICP approach, as shown in figure 4.6.

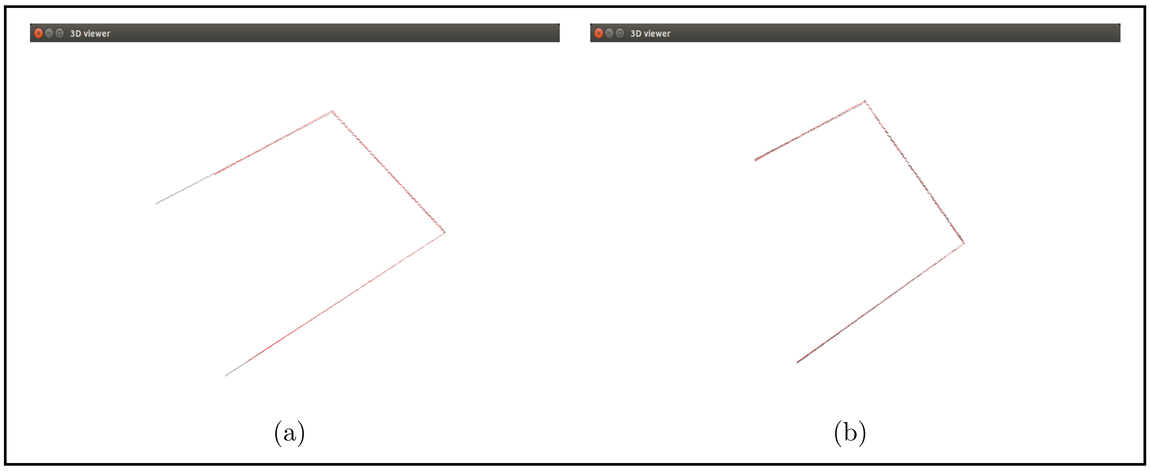


Figure 4.6: A comparison between raw ICP and ray-cast ICP registrations. (a) describes the raw ICP result and its score = 0.000523665. (b) describes the matching result between ray-cast models, and its score = 0.0000513. It shows that using ray-cast algorithm can improve the tracking performance.

4.4.2 Constructing Ref Model

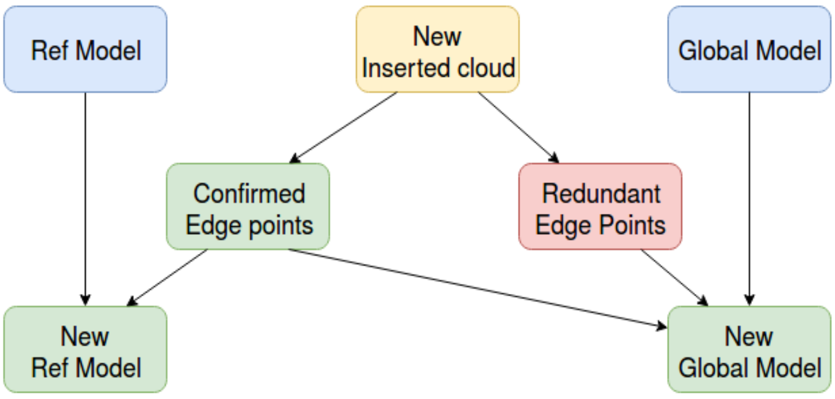
In the existing dense mapping algorithms, the ray-cast theory is often combined with voxel reconstructed model, e.g. SDF and TSDF voxel models. The voxel model saves the current distance from the camera to the surface, which can be updated in real time with GPU. When using the ray-cast algorithm for the voxel model, the resulting point cloud are the viewed surface of model at current camera's perspective. However, the TSDF model does not fit the algorithm we proposed. The main reason is that the number of points in ref model is too small. In addition, deleting and adding voxel in TSDF model is much harder than point cloud. Whenever user wants to update the TSDF model, user needs to convert it to another format and convert back later. Therefore, we still use point cloud to represent the model.

In general, the longer the reconstruction process progresses, the more points are added to the reconstruction model. After a couple of past frames, we can divide the points into three categories:

- The exact edge points - These points accurately describe the edge information of the plane and can be used as a global landmark point.
- The might be accurate points - Since the point cloud model reconstructed from the depth information is sparse, and the same edge has different 3D models at different viewing angles. These points are the points do not accurately describe the edge of the plane with the view changes.
- The error points. The generation of such points is mainly due to errors caused by previous steps, such as the wrong 2D edge, the wrong ray-cast model, and the wrong matching point. These points cannot be added to the global landmark model.

Ref model can be considered as a global model that is cleaner and tidier with less points and errors. Since its restrict filtering method, it can be a long-term landmark model in tracking and mapping process, rather than a more detailed global reconstruction model. Figure 4.7 shows the difference between updating ref model and global model.

To define the point in new inserting cloud is correct or redundant, the K-mean nearest neighbour technique is deployed. When the new cloud $\mathcal{M}_{New}$ is inserted into the ref model $\mathcal{M}_{Ref}$, each point P_{New} are searching for its nearest neighbours P_{Ref} in $\mathcal{M}_{Ref}$ with a distance threshold. If there are more than 3 neighbours exists in the

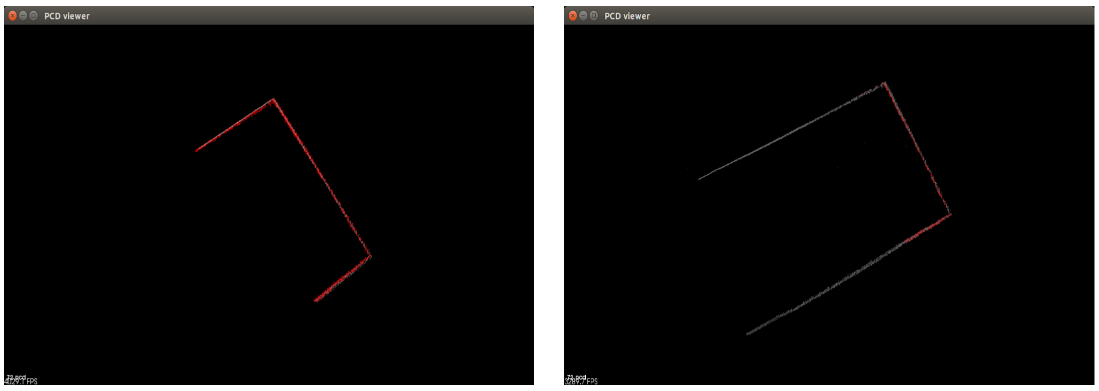


(a)

Figure 4.7: Difference between updating ref model and global model

search domain, we can say that this point belongs to the edge point cloud. Otherwise, this point is a redundant point and ignored. A thing needs to be mentioned is '3 neighbours' is specified to current dataset, user may change this value in other scenarios.

This harsh updating strategy could keep the accuracy of ref model, also filter out useless points. Figure 4.8 shows the ref model harsh updating strategy, in which the grey points are from ref model points and red points are from new inserted frame. (a) describes the registration result from the optimised ICP process. (b) describes only part of new frame points being updated to the ref model.



(a) (b)

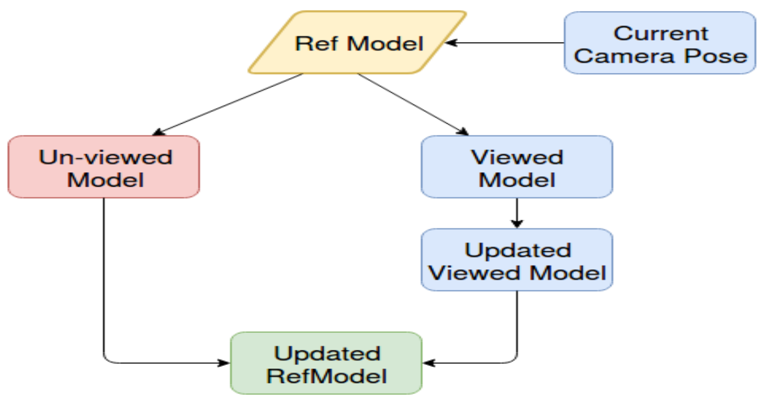
Figure 4.8: Ref model harsh updating strategy.

4.4.3 Updating Ref Model

This section describes a novel model updating strategy specific for purposed mapping algorithm. Since the ref model is presented with point cloud, it requires a specific model updating strategy. The biggest problem of the point based reconstructed model is that the variance of depth on each reconstructed 3D plane is different because of different perspective. So, the proposed system have adopted a more direct way to force updating the ref model.

As mentioned in sub-section 4.4.2, when the new cloud point is received, only the small numbers of points could be inserted in the ref model. This approach keeps the model simple, and makes the points of the ref model gathered into the correct edge area with frame-by-frame update. In more detail, during updating process, the ref model $\mathcal{M}_{Ref}$ could be split to $\mathcal{M}_{nV}$ in which case the points not in current camera's view, and $\mathcal{M}_{V}$ describing the viewed points. When a new frame f^{i+1} is inserted into the mapping loop, its pose will be optimized firstly. After the pose of new inserted frame has been confirmed, all the points in $\mathcal{M}_{V}$ will be projected on the 2D image of f^{i+1}, and they will be extracted and deleted from the ref model $\mathcal{M}_{Ref}$.

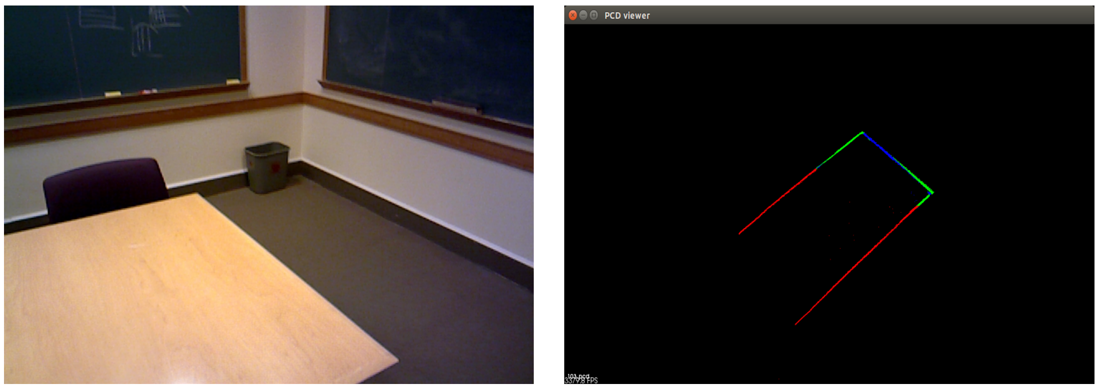
By only keeping the edge points in frame f^{i+1}, our system will reconstruct the contour model with the perspective in this frame and regroup with $\mathcal{M}_{nV}$ to form an updated $\mathcal{M}_{Ref}$. An important point needs to be mentioned here is $\mathcal{M}_{nV}$ not only includes the points that are not in the current camera's view, but also includes the points that do not have correct depth value in f_{i+1}. With this process, the ref model will be kept updating, and makes the ray-casted model as much as similar to the new inserted model. This process could be illustrated in figure 4.9



(a)

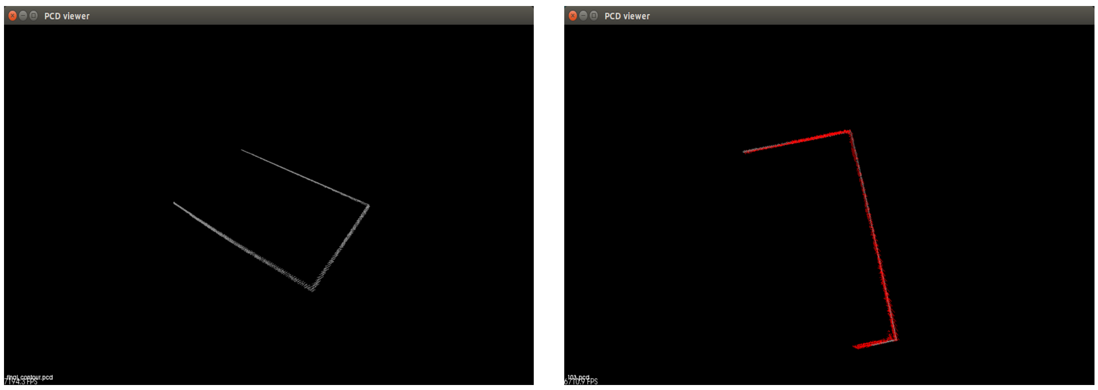
Figure 4.9: The Ref model updating strategy

By using the proposed ref model updating strategy, the ref model always maintains a camera perspective that is consistent with the current frame. The performance of the proposed algorithm is described in figure 4.10 and figure 4.11. It should be noticed that in figure 4.10(b), the red point are the un-viewed points in ref model, the green points are the updated viewed points and the blue points are the viewed points which do not have the correct depth values in current depth image.



(a) (b)

Figure 4.10: (a) is the original input RGB image. (b) describes the ref model in the perspective of this frame.



(a) (b)

Figure 4.11: (a) is the ref model updated in current frame. (b) describes the matching result between the next input image (red) and current ray-casting model (grey).

4.5 Lost Recovery

Lost tracking is one of the basic problems in visual mapping. There are many reasons to trigger lost tracking problems, such as not enough features (include 2D features and 3D geometric features) in current perspective, or a large motion between adjacent frames. Since dense mapping methods have not generated object descriptions, it is more difficult to find the spatial correspondence between past frames and current frame. Based on the existing algorithm, a lost recovery approach is introduced in this part specifically for the proposed contour-to-contour mapping algorithm.

4.5.1 Lost Recovery strategy

When the lost tracking happens, system will break the whole mapping process into pieces, and then tries to find the spatial correspondence between them. There are three modes in the proposed mapping system: Normal mode, Lost Mode and Lost Recovery Mode. The normal tracking mode uses the optimised contour-to-contour matching algorithm to register every successfully matched frames. In this mode, the ref model and the pose of each frame are kept updating until the lost mode is triggered. And triggering the lose mode is determined by the optimised matching score optimisation algorithm proposed in Chapter 4.4.1.

When tracking failure occurs in two consecutive new frames, the system deletes the poses of these two frames in global, and then enters the Lost mode. Assuming the last successfully tracked frame is f_L^{i-1}, and two lost frames are f_L^i and f_L^{i+1}. When lost mode starts, system restarts at f_L^i and continually accepts new frames. The new frame will be matched with its last frame. Since the roughly estimated pose of f_L^i is not accurate, we do not use global ref model to optimise tracking result. Therefore, a normal ICP algorithm is applied to match two frames, and its derived matching score is used to determine the tracking accuracy directly.

To maintain the accuracy, we set a higher accuracy requirement in this mode. All the successfully matched frames will be saved in a fragment. This process ends when the tracking failure occurs again. System firstly stops reading the new image and down-sampling the fragment. Then it enters the Lost Recovery Mode, aiming at finding the correspondences between current fragment and ref model and then updating the pose and 3D data of each frame to global. In the recovery model, the memory of fragment will be released, but their 3D model, Start and End poses will be saved to global. Figure 4.12 shows the different modes in the proposed mapping algorithm, and figure 4.13 shows the work flow of the lost mode.

When the lost tracking problem occurs, the system will save the poses of all the successfully matched frames, the Global model and a global ref model which save as a fragment $\mathcal{M}_F^i$. Since these information are kept updating during the normal tracking mode, it won't cost extra time for saving these information. The ref model and global reconstruction model will not be updated until the lost mode is end. The

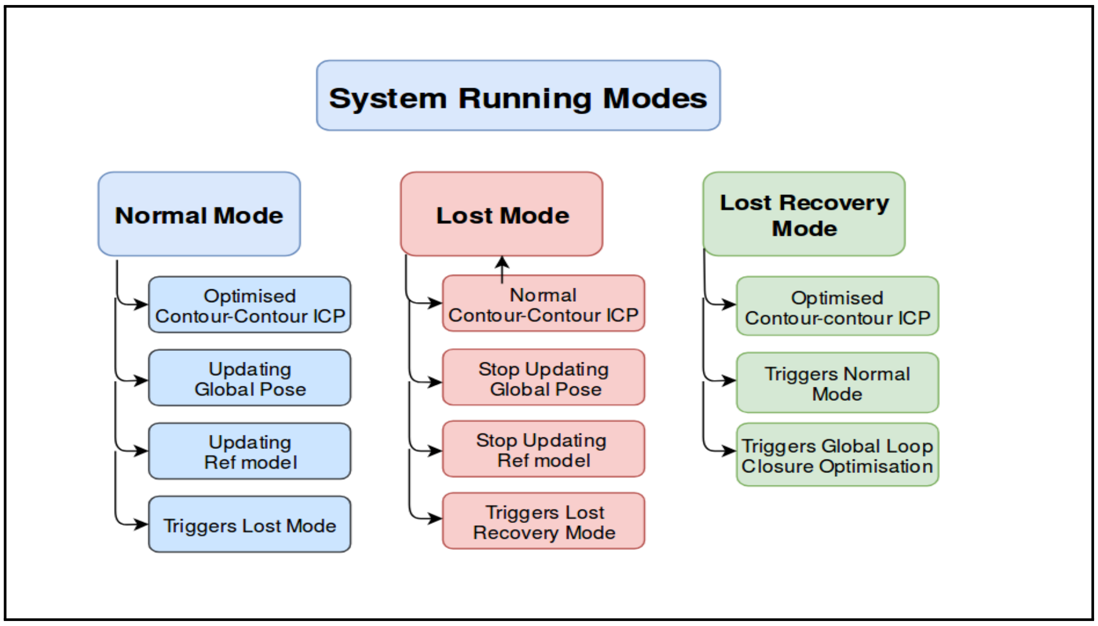

System Running Modes
Normal Mode
Optimised Contour-Contour ICP
Updating Global Pose
Updating Ref model
Triggers Lost Mode
Lost Mode
Normal Contour-Contour ICP
Stop Updating Global Pose
Stop Updating Ref model
Triggers Lost Recovery Mode
Lost Recovery Mode
Optimised Contour-contour ICP
Triggers Normal Mode
Triggers Global Loop Closure Optimisation

Figure 4.12: Different Modes in the proposed Mapping algorithm

start frame of the fragment $\mathcal{M}_F^{i+1}$ is f_L^i, and its start pose is its roughly estimated pose.

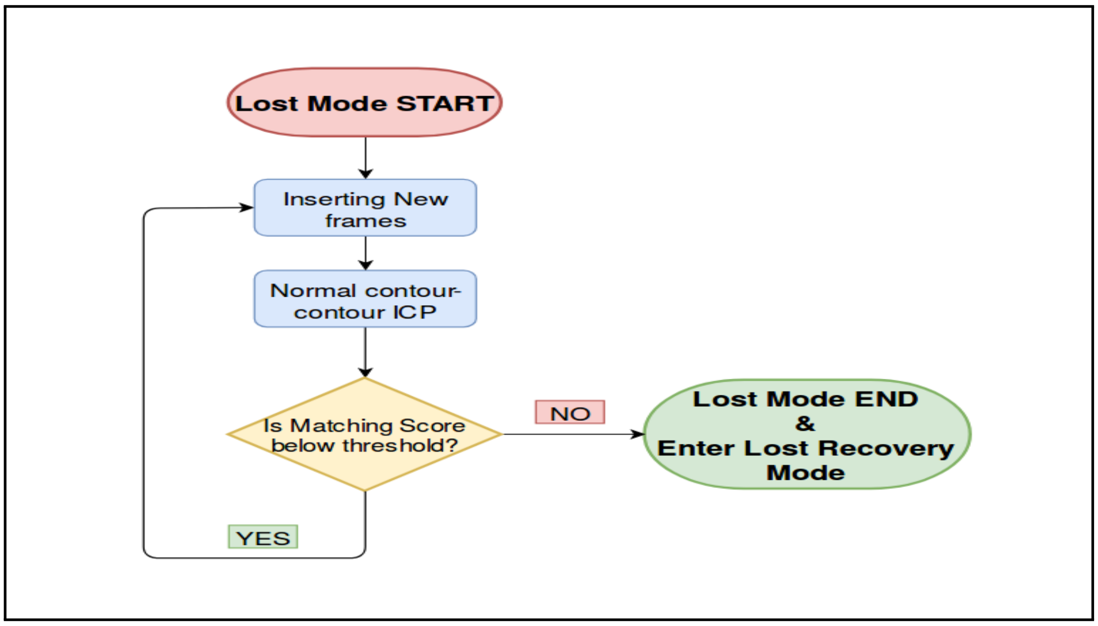

Lost Mode START
Inserting New frames
Normal contour-contour ICP
Is Matching Score below threshold?
NO
YES
Lost Mode END & Enter Lost Recovery Mode

Figure 4.13: Work flow of Lost Mode

To introduce the lost recovery strategy in more detail, we still use $\mathcal{M}_F^i$ as global ref model and $\mathcal{M}_F^{i+1}$ as a lost fragment. In first stage, $\mathcal{M}_F^{i+1}$ will ray-cast its own model based on the endpoint pose in the global ref model $\mathcal{M}_F^i$, and then match it with f_L^i. This will result in a rough registration and a raw pose $\hat{P_{Raw}}$ between the two viewpoints. Then the ref model $\mathcal{M}_F^i$ and $\mathcal{M}_F^{i+1}$ will perform a second ray-cast on their model with P_{Raw}, resulting in two different 3D models which enjoy the same perspective. After the second ray-cast process, the two ray-cast point clouds are much similar than the original two point clouds, resulting in a better registration performance. Note that the matching criteria in this case is loose since we want to continue tracking process. And the trajectory can be further corrected in global optimisation program introduced in Section 4.6. Experiments have shown that this method can completely and accurately match the fragments with the Ref model, even when the fragment is out-ranged the ref model. Finally, the ref model will add the 3D point in $\mathcal{M}_F^{i+1}$ and update itself. The system will end the lost recovery mode and re-enter the normal mapping mode until the next mismatch occurs. Note that system restarts at f_L^i instead of end frame of $\mathcal{M}_F^{i+1}$. Because we want system to re-check the estimated poses in lost mode and re-update the global ref model.

4.5.2 Lost Recovery Implementation

This section aims to verify the efficiency of the proposed Lost Recovery algorithm via the experiments. Figure 4.14 presents the ref model and the fragment. Figure 4.15 shows the new registration results from applying purposed lost recovery algorithm. In figure 4.15(a), the blue points and red points are from adjacent key-frames respectively. its matching score = 0.004763. In figure 4.15(b), the green points are from the raycast ref model, and the red points are from the fragment. Its matching score = 0.00001348.

figure 4.15 has shown that the proposed lost recovery algorithm is very useful. To verify this algorithm is robust to long distance fragments, we compare the ref model with another fragment which has 6 key-frame distance, and the results are shown in figure 4.16 and figure 4.17. In figure 4.17(a), green points and red points are from adjacent key-frames respectively. Its matching score equals to 0.00135. In figure 4.17(b), the green points are from the ref model, the red points are from the fragment. Its matching score equals to 0.00004735.

Another experiment is arranged to test the performance when the new inserted frame is more further away, and it is out-ranged the ref model. In this case, our algorithm will still follow the two-way ray-cast process to register the shared part of two models, and the results are shown in figure 4.18 and figure 4.19. Figure 4.18 has described global ref model in current frame and model of new inserted frame. It clearly shows that new frame is out-ranged ref model. Figure 4.19(a) shows the registration result in normal mode in this case, and its matching score = 0.0020353. Figure 4.19(b) shows the registration result in lost recovery mode, and the matching score = 0.0001964.

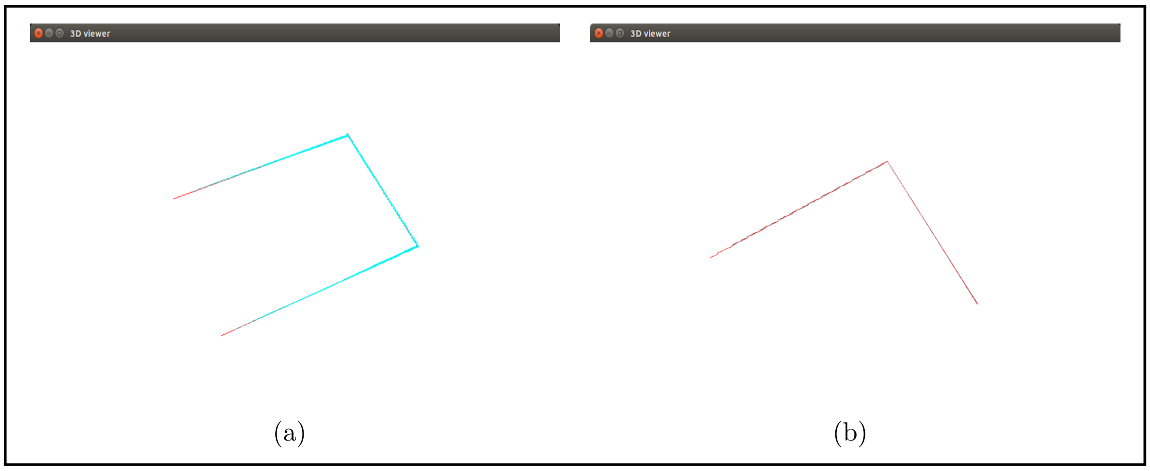


Figure 4.14: (a) is the ref model when lost tracking happens. (b) is the fragment saved in Lost Mode

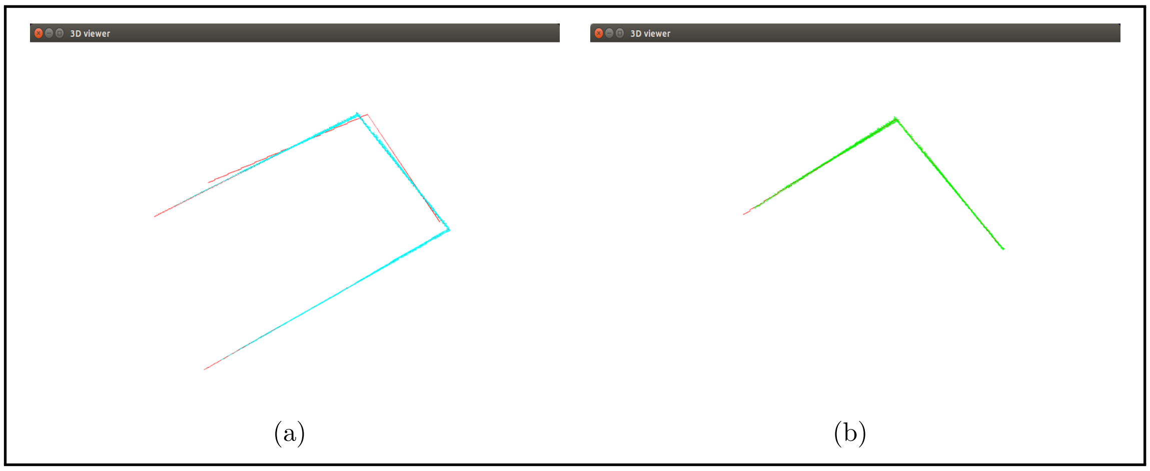


Figure 4.15: (a) is the registration result in normal mode, which has been decided as a miss matching case. The blue points are describing the global ref model, and the red points outside the blue model is from the mis-matched frame. (b) is the result after lost recovery. The green points are the raycast global ref model, and the red points are from the lost fragment.

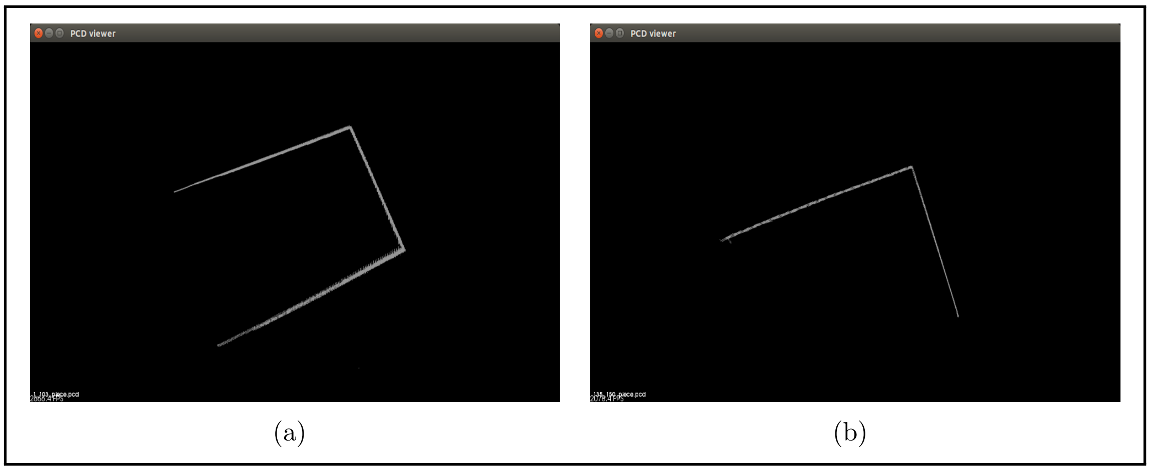


Figure 4.16: (a) is the ref model. (b) is the fragment saved in Lost Mode

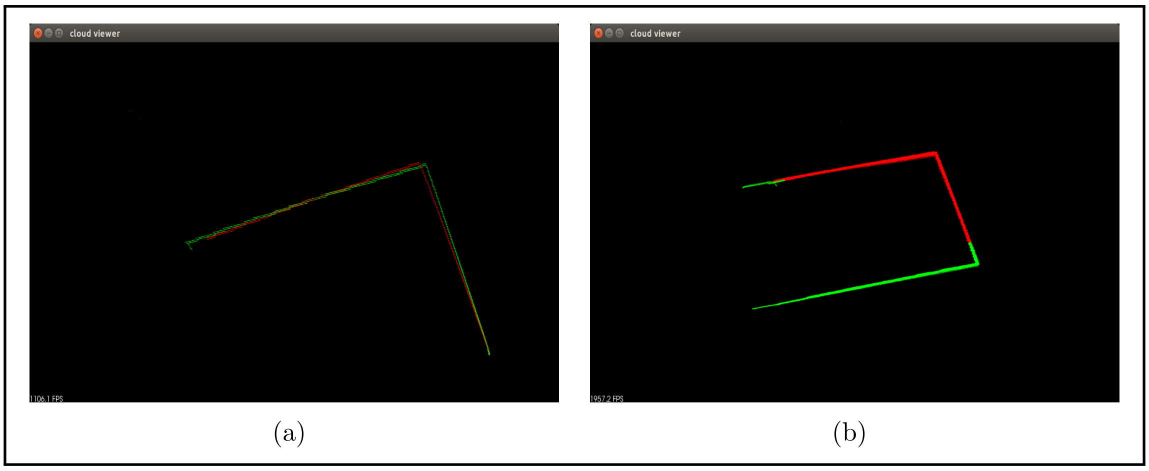


Figure 4.17: (a) is the registration result in normal mode, and this has decided as a missed matching case. (b) is the registration result with the proposed lost recovery algorithm.

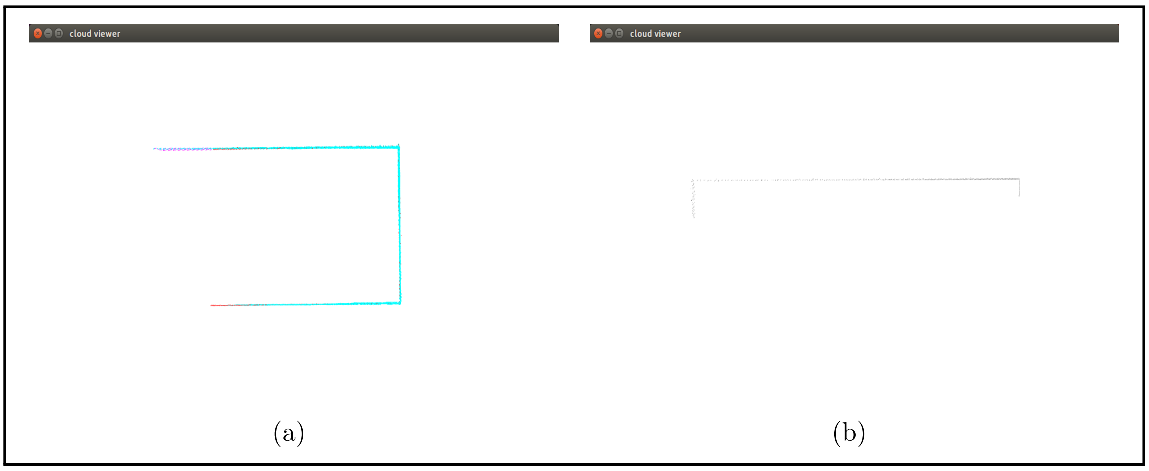


Figure 4.18: (a) is ref model in current frame. (b) is the long-distanced fragment saved in Lost Mode

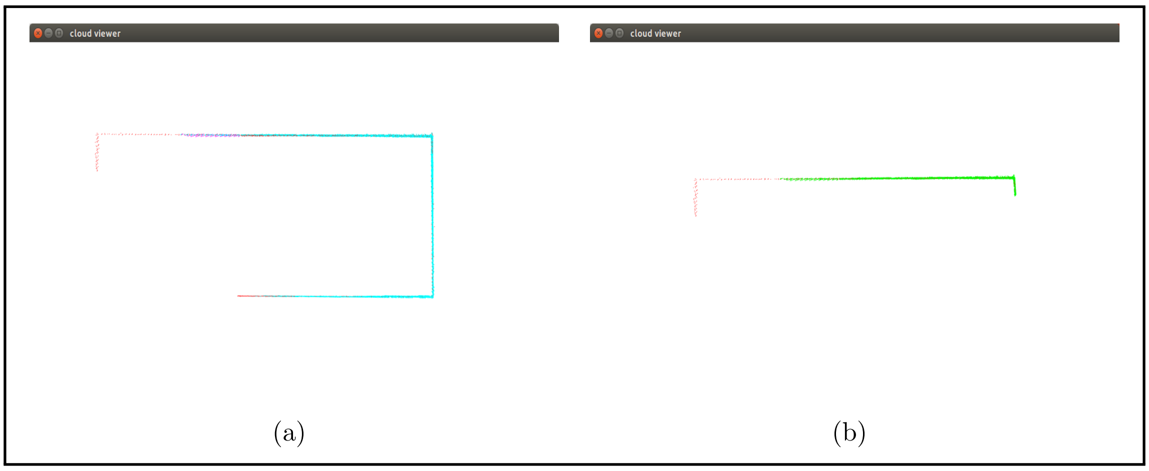


Figure 4.19: (a) is the registration result in normal mode, and this has decided as a missed matching case. The blue points are from ref model and the out-ranged red points are from new inserted frame. (b) is the registration result with purposed lost recovery algorithm. The green points are describing the shared view part of both model.

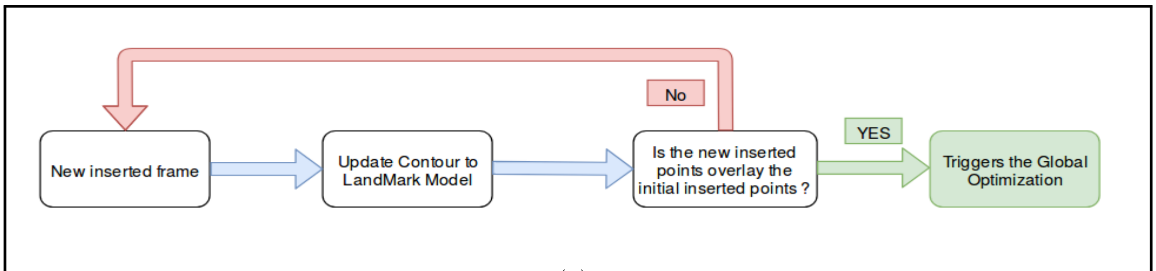
4.6 Global Loop Closure Detection

Modern visual mapping techniques include two main optimisation strategies: local loop closure detection and global loop closure detection. The main difference between local optimization and global optimisation is local method optimises the tracking performance with small numbers of nearest key-frames, and the global method aims at finding the correspondence between current frame and initial frame in the beginning of mapping process. In our system, local loop closure has been been implemented with ref model and ray-casting theory. Meanwhile, an unique global loop closure detection process is triggered in parallel to detect if the contour of target plane is fully appeared or has been entirely viewed.

Proposed Global loop closure optimisation algorithm is based on detecting an entire contour of landmark plane. As mentioned in former sections, global ref model is a simplified contour model, and it is used for consistently local alignment. When this landmark model has been updated with a new inserted frame, it has been disciplined with three parts: the new added points, the viewed points, and un-viewed points. When a number of new added points is closed to the un-viewed points, or overlaying part of the un-viewed points, the system will trigger loop closure detection. This number threshold in this step is set to 30, and this value can be adjusted in other scenario. To save the unnecessary computational cost, the global loop closure detector will not start until the new frame contains the contour not included in ref model.

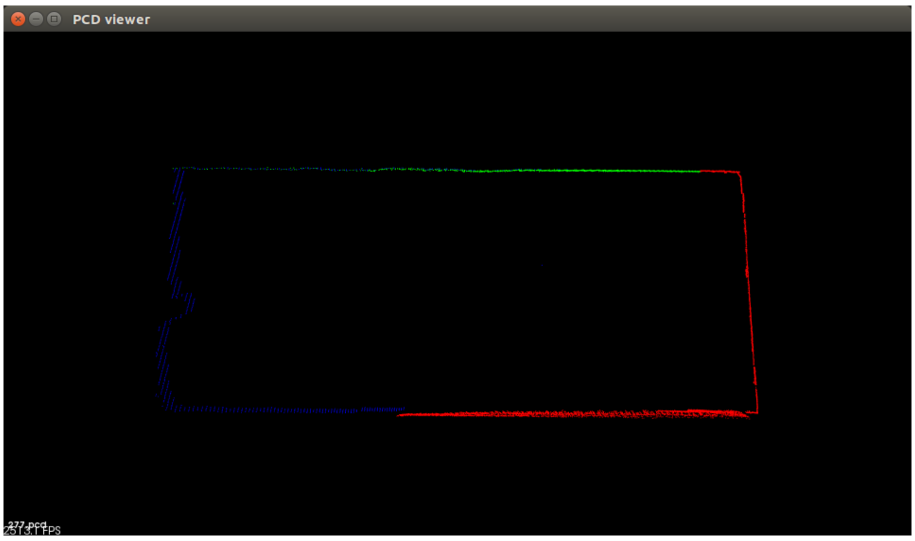
Figure 4.20 illustrates the global loop closure detection strategy, and figure 4.21 describes the updated landmark model when Global loop closure occurs, in which the red points are un-viewed part of ref model in current estimated camera's view; the green points are the viewed points in ref model; the blue points are the new inserted points to ref model model. In figure 4.21, ref model has a large amount of new added point (blue points) because of drift error. Since there are more than 30 blue points has found they are very close to un-viewed ref model (red points), system decides the entire contour has just been viewed and enters the loop closure optimisation process.

When the system has detect the loop closure occurs, it will try to match the current inserted frame and the first frame with a two-way ray-casting theory. Due to the long distance between two frames, the system uses ray-cast theory to find the points in two models have the same shared view. In more detail, the model of the first frame and the current frame will be ray-cast with the current estimated pose and the initial pose respectively, so both models will be cut into two parts: the viewed


No
New inserted frame
Update Contour to LandMark Model
Is the new inserted points overlay the initial inserted points ?
YES
Triggers the Global Optimization

(a)

Figure 4.20: Global Loop closure detection strategy



(a)

Figure 4.21: Updated Landmark model when Global loop closure occurs.

model and un-viewed model. By registering only the viewed parts, the estimated pose of a new inserted frame will be further optimised so that a complete contour data of a reference plane is recovered. The matching result is described in figure 4.22.

The final reconstruction results after optimisation are described in figure 4.23. Figure 4.23(a) describes that our algorithm correctly tracks the contour and reconstructs the contour model after 80 key-frames. Figure 4.23(b) describes the final scene model. Compare with reconstructed contour model, the scene model is less accurate. The main reason of this problem is our purposed contour extraction algorithm may generate errors when some images are very unclear.

In fact, this problem can be optimised. On the one hand, in order to verify the our contour extraction algorithm performs better than existing algorithm, we use an old dataset. Some images included are unclear due to sensor quality and problem of shooting jitter. So, this problem is able to be solved if the camera moves more smoothly while collect the data. On the other hand, the reconstructed contour model

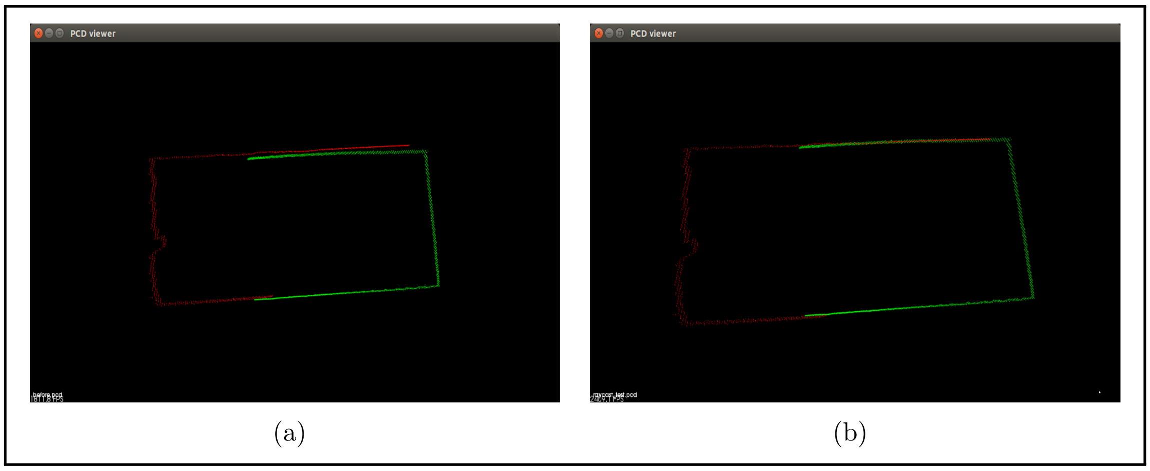


Figure 4.22: Global optimisation performance. (a) describes the accumulated error after matching 45 key-frames when the optimisation triggers. (b) describes the registration result based on two-way ray-casting theory

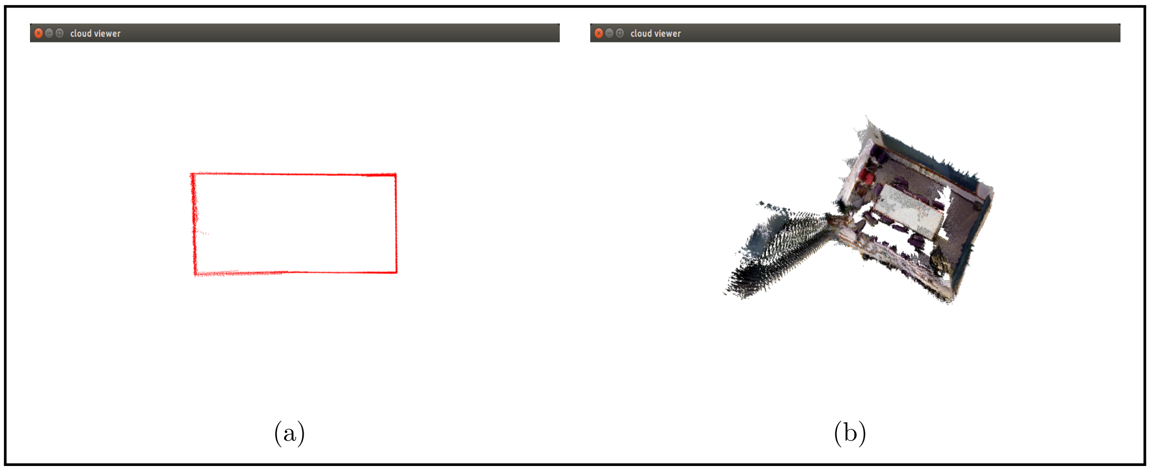


Figure 4.23: Final mapping result. (a) describes the reconstructed contour model. It is different to figure 4.22(a) since we correct the occluded contour model in some frames. (b) describes the reconstructed scene model.

proves that proposed visual mapping algorithm can accurately track the contour of plane. Therefore, our algorithm can get a more accurate reconstructed scene model when the images are in high resolution. We verify this assumption in ICL-NUIM dataset (Handa et al., 2014) experiment and real scene experiment in Section 6.4.

Figure 4.24, figure 4.26, figure 4.27, figure4.28 and figure 4.25 have demonstrated our experimental result in ICL-NUIM dataset. Since the data in ICL-NUIM has a better quality, the reconstruction results are much better than the one described in figure 4.23. In this experiment, two pictures on the wall are tracking objects. Figure 4.24 has described two extreme cases during tracking. Since our algorithm does not requires the tracking target is entirely visible during tracking, it can work in these two extreme conditions and reconstruct a large range of environment based on one planar object. Then, We evaluate our estimated trajectory with ground truth (provided by dataset) in figure 4.25, and it shows the average error is smaller than 0.006. Finally, figure 4.26, figure 4.27 and figure 4.28 have described the reconstructed around two different pictures.

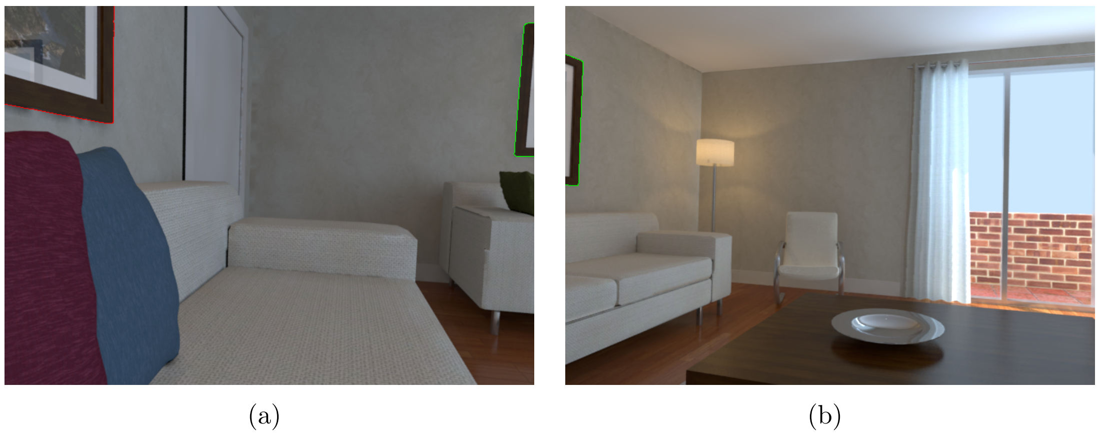


Figure 4.24: The extreme cases our tracking algorithm can handle. (A) The red contour is tracking target. (B) The green contour is tracking target. Since our algorithm works when only a small part of contour is visible, it can reconstruct a large range of scenes based on one plane.

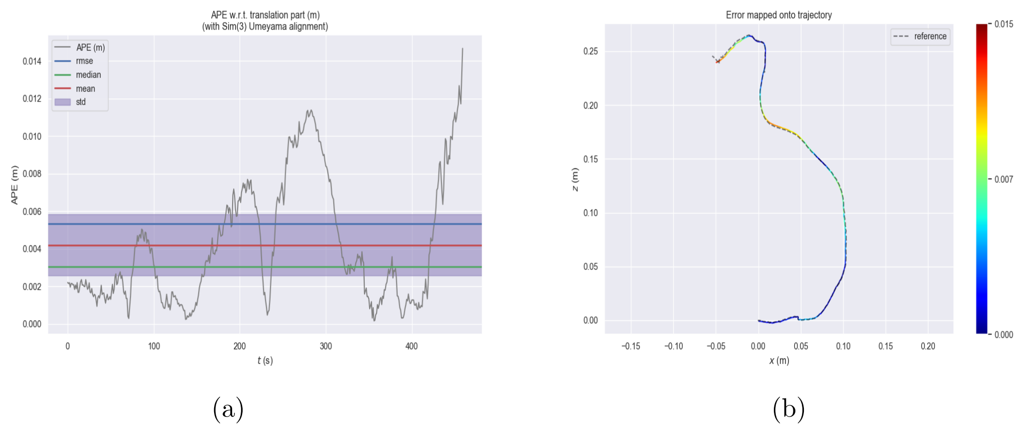


Figure 4.25: An evaluation of our algorithm. (A) is the calculated error between our estimated trajectory and ground truth. (B) is the trajectory difference. The dotted line is from ground truth, and the color line is our trajectory.

4.7 Summary

Most of indoor broadcasting or demonstration scenes is an empty room with large planes, such as broadcasting tables, screens and projection screens. However, these objects do not have enough visual features for the existing visual mapping algorithms to achieve the indoor mapping goal. As a result, a depth information based visual reconstruction algorithm has been introduced in this chapter. The most commonly

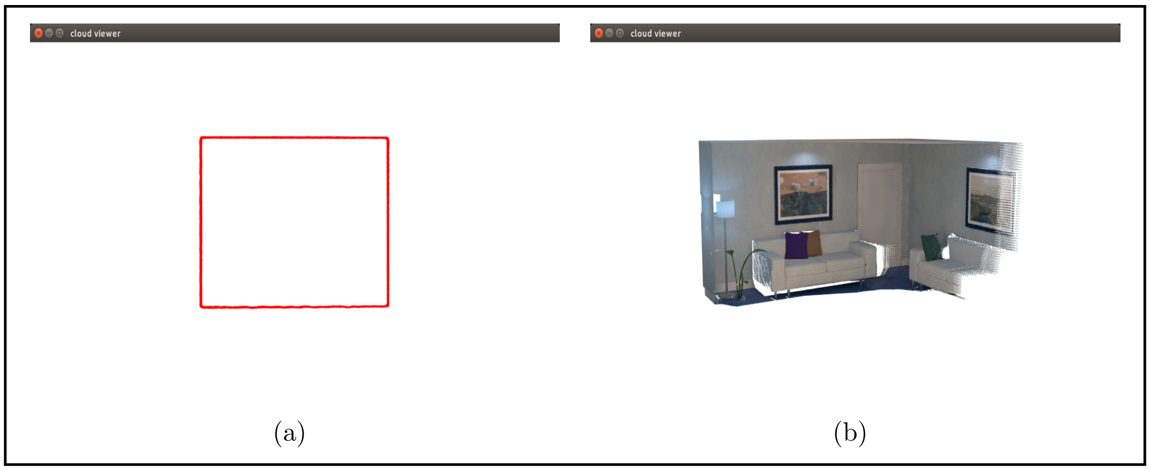


Figure 4.26: The reconstructed model of first picture. (A) is the reconstructed contour model. (B) is the scanned scene around first picture.

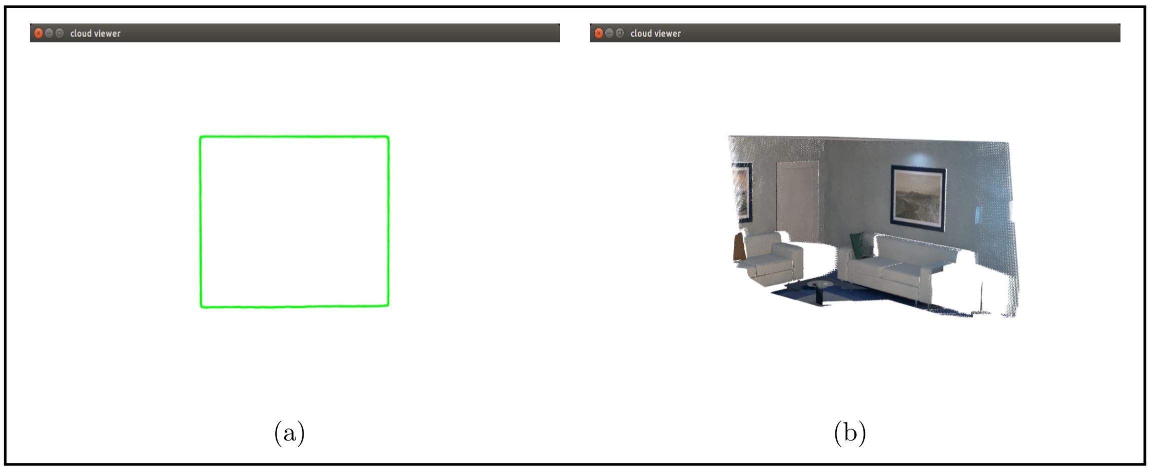


Figure 4.27: The reconstructed model of second picture. (A) is the reconstructed contour model. (B) is the scanned scene around first picture.

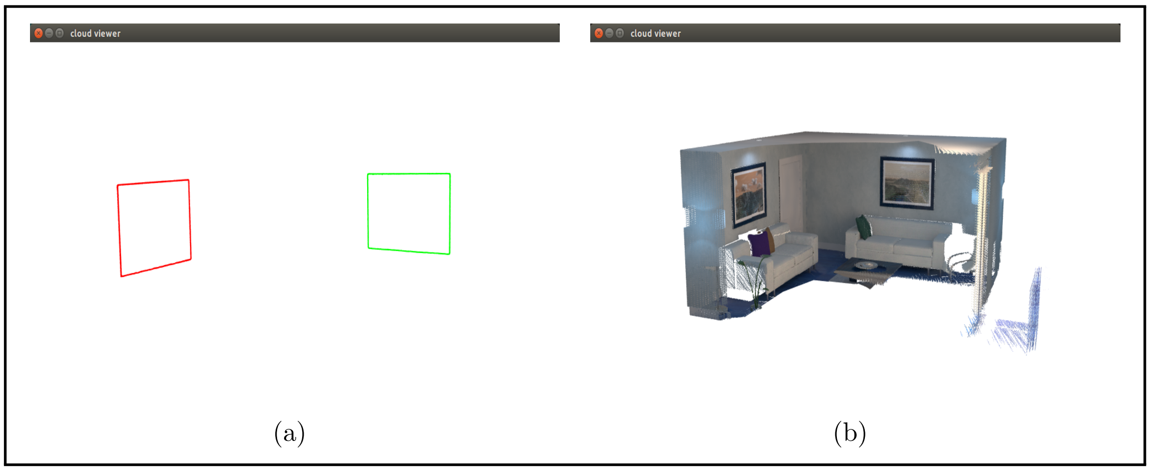


Figure 4.28: The reconstructed model of two pictures. (A) is the reconstructed contour models of two pictures. (B) is the scanned scene around two pictures.

used Kinect-V2 is used as the basic sensor for capturing depth and corresponding RGB images. By tracking one of the large planar surface of an object inside feature-less scene, the proposed system could successfully define the pose of each input frame and correctly register each input 3D scene to a global reconstruction model.

Comparing with recent visual mapping algorithms, the proposed system has greatly improved the accuracy and robustness, but sacrificing the processing time as trade-off. In real applications, this system could be used as a global optimisation process based on existing visual mapping algorithms when the system is running in feature-rich environments. It can also be used as unique mapping process when the environment is feature-less. The advantages of the proposes system include.

- Firstly, it solves the mapping problem in features-less environments. Visual mapping in feature-less scenes is always a challenge task, especially in small indoor environments with less structural information. Compared with the existing algorithms, our algorithm only needs one medium-sized object with a plane as a reference model inside a scene, which is very easy to implement. Moreover, our algorithm does not rely heavily on the global loop closure optimisation, so it can reconstruct the part of the scene instead of forcibly rebuilding the entire scene.

- Secondly, the proposed system has introduced a novel scene-to-scene matching approach based on the contour data of a planar object for registration. The experiment results show that using part of scene for registration is faster and more accurate. Meanwhile, it won't lost important scene data. This approach could be a new dense data registration strategy for the further research.

- In most of existing mapping algorithms, the planar area is useless because of lacking features. By fusing the contour data with CNN segmentation labels, the proposed system uses reconstructed contour model as the 3D description of 2D image feature. When the system is running in a feature-less environment, the system can perform trajectory estimation and optimisation based on the global 3D contour model and the current 3D contour model as long as there is a part of this plane visible in the perspective of the current frame.

5

Mobile SAR system design and immersive virtual plays

5.1 Introduction

State-of-art projection-SAR is a branch of AR technology that covers and changes the appearance of the scene by projecting dynamic virtual contents. Similar to other AR branches, it provides an immersive human-computer interactive experience through analysing human body motion and creating the virtual experience. Existing projection-SAR is fixed and it has a rigidly limited working range. A common method to increase its working range is increasing the number of equipment, which however limits its the application field. The recent researches focuses more on the development and improvement of virtual effect design although a limited working range is still a serious limitation for a projection-SAR system.

The studies on how to work with larger scenes with a single sensor unit is still rare because of two main reasons: (i) The application scenario of a Projection-SAR is always feature-less, and the structured light projection is the most reliable and convenient scene registration approach. Thus, the scene registration approach is very solid for projection-SAR applications and difficult to be replaced. (ii) The projector needs to calibrate itself whenever it changes its pose. In our system, we have optimised the first problem with the previous research and tried to minimise the error brought by the second problem.

This chapter has described a complete design of purposed mobile projection SAR robot. Firstly, the system design is presented, including the applied sensors and robotic mobile pedestal. Then, its workflow and working method are introduced. Three different virtual applications are presented to prove the rationality and practicability of the entire system. Finally, a brief conclusion is given to summarize the human-computer interactive immersive plays designed for projection-SAR system.

5.1.1 System design

The designed system consists of two main parts: (i) a projection-SAR system that provides the user with the virtual interactive AR plays; (ii) a ground mobile pedestal that moves around the scene by following the user. Figure 5.1 shows a complete workflow for the proposed system.

- A projection-SAR system that mainly aims at providing the human-compute virtual interactive plays.
- A ground mobile pedestal that mainly aims at moving the sensors around the whole scene.

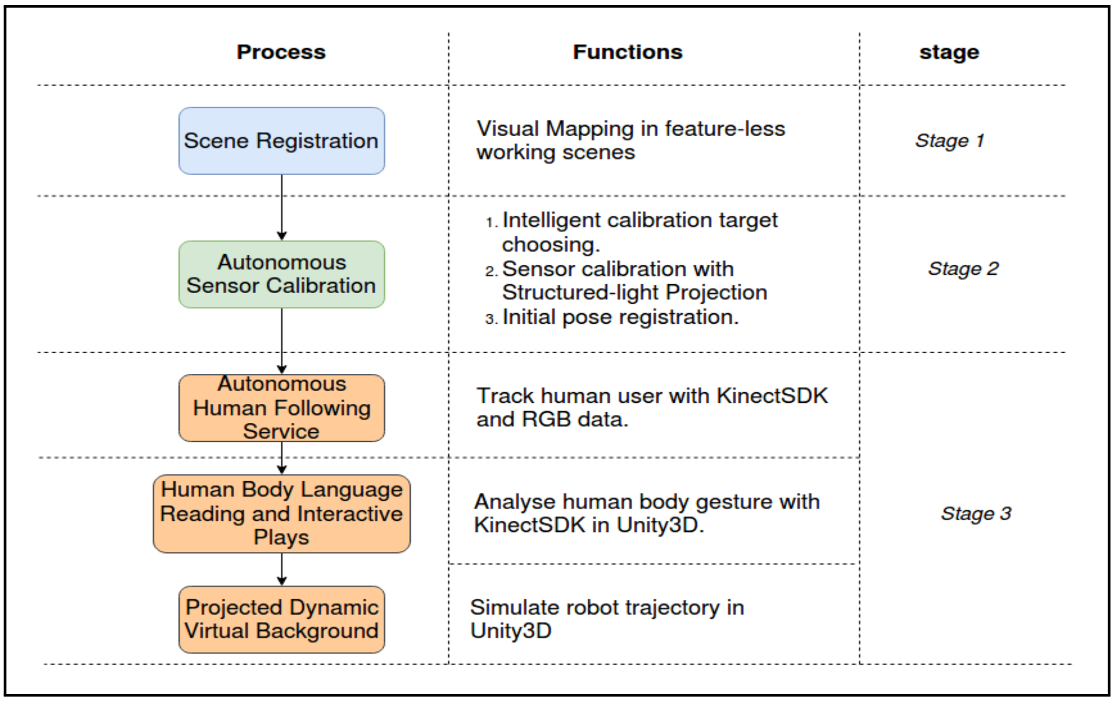

Process
Functions
stage
Scene Registration
Visual Mapping in feature-less working scenes
Stage 1
Autonomous Sensor Calibration
1. Intelligent calibration target choosing.
2. Sensor calibration with Structured-light Projection
3. Initial pose registration.
Stage 2
Autonomous Human Following Service
Track human user with KinectSDK and RGB data.
Human Body Language Reading and Interactive Plays
Analyse human body gesture with KinectSDK in Unity3D.
Projected Dynamic Virtual Background
Simulate robot trajectory in Unity3D
Stage 3

Figure 5.1: A description of working process of the proposed Mobile Projection-SAR system

As described in figure 5.1, the proposed SAR system mainly includes three stages as described below:

- **Stage 1**: the whole working scene is reconstructed using the proposed visual mapping algorithm introduced in Chapter 4, and a point-cloud presented 3D scene model is derived. We need to transfer the '.pcd' point-cloud file to wavefront '.obj' file to use this model in unity3d for further AR design. This process has replaced the commonly used scene registration process with structured-light projection, in a much larger projection area.

- **Stage 2**: the system will calibrate the visual sensors and obtain the initial pose of the system with the structured-light projection. There are three steps in this stage. Firstly, a feature-based algorithm is designed for choosing the best calibration area throughout the whole scene since structured-light projection technique requires solid visual features or geometric features. Then, the user could move the robot to calibration position and make the robot facing calibration scene. The system will calibrate its visual sensors and its initial pose with structure-light projection. Since the initial pose derived by calibration is relative to the calibration area instead of the whole scene, we design a plane-based tracking algorithm to find the initial pose of robot relative to the whole reconstructed scene in the final step.

- **Stage 3**: the robot will enter the working stage. figure 5.2 describes the whole system and sensor allocation. As can be seen, the camera sensor is assigned to the robot pedestal system and projection-SAR to detect human user and analyse user's body gesture; and the projector is used in the projection-SAR system to project dynamic virtual content.

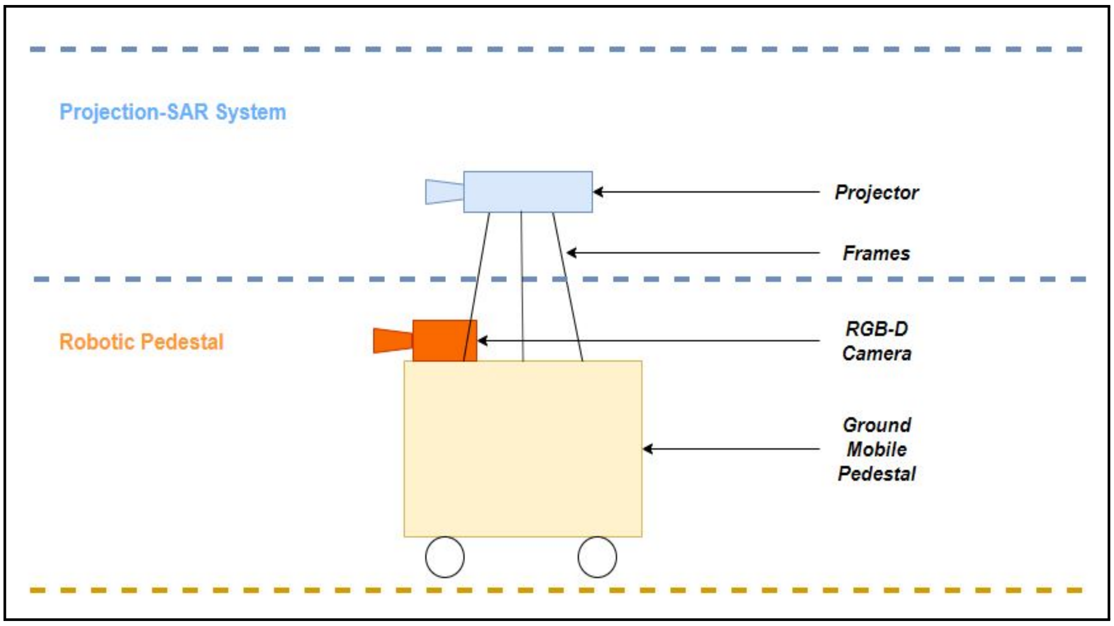


Figure 5.2: A description of the proposed Projection-SAR robot and visual sensor allocation.

A RGB-D camera is deployed on the robotic pedestal to follow the user and keep the user inside its field-of-view. While the user is moving, the camera will recognise user' position in current view and give the corresponding moving order to the robotic pedestal. The wheel movement of the robotic pedestal is used to estimate the moving distance. The projector is used in purposed mobile projection SAR system, and we have designed three AR plays.

- In the first AR play, the user can control the 3D virtual content with its upper limb movements, and the virtual content could be moved or rotated along the Cartesian coordinate. This is the most basic interactive virtual play provided by AR.
- In the second virtual play, the user can control the viewer's focus by his position in front of the camera. While the user is moving around, the virtual content could automatically zoom-in, zoom-out or rotate. This AR plays is specific to projection-SAR application, and it is design for fixed application only. Therefore, in our mobile projection-SAR system, when user chooses this AR play, robot will stop moving immediately. And robot will restart the user tracking service when this AR play is finished.
- In the third virtual play, the robotic pedestal can automatically follow the user and project some dynamic virtual contents on the background corresponding to current user position in the whole scene through estimating its movement. In this play, the wheel movement is the input, and the dynamic projected virtual content is the output.

5.1.2 Visual sensors

The visual sensor for the proposed Mobile-SAR system includes one Kinect-V2 camera and one recent small scale projector. These two devices can be connected through "RoomAlive" software develop kit from Microsoft (Jones et al., 2014), and their intrinsic and extrinsic parameters could be estimated through the calibration tool in RoomAlive.

1. RGB-D Camera in figure 5.3
   Kinect-V2 RGB-D Camera is used in our robotic system. Kinect-V2 provides RGB images and Depth data simultaneously and includes many useful tools, such as human body recognition, body gesture recognition and distance measurement. These tools are opened for public use and will be deployed in Section 5.6. We use Kinect-V2 camera in both data recording process and SAR running process. In scene reconstruction process, Kinect-V2 is used to

record RGB images and depth images. In SAR running process, Kinect-V2 camera is used to detect the user's torch and estimate the moving direction to keep user in camera's field of view.

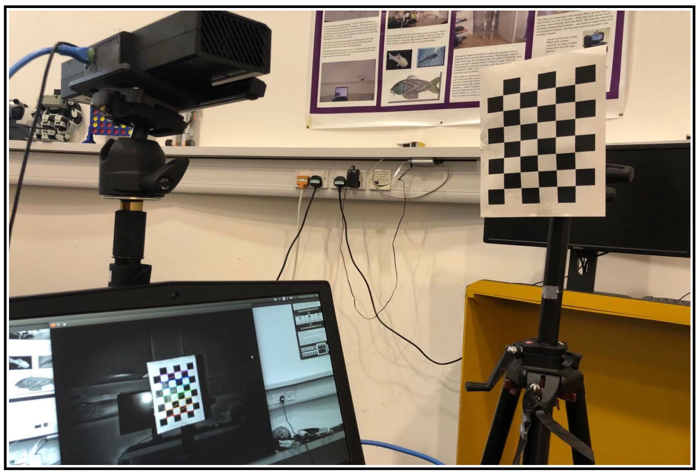


Figure 5.3: KinectV2 RGB-D Camera and its self-calibration process

2. Visual Projector in figure 5.4
   Due to the weight limitation of the robot platform in our robotic lab, a small-scale DLP projector is used on the platform as the main output source. The projector is connected to a computer and can display the image from Unity3D directly.

Figure 5.4: Small-scale DLP projector

5.1.3 Ground mobile Pedestal

Figure 5.5(a) shows a Pioneer 3 mobile robot used as our ground mobile platform, and figure 5.5(b) describes its bottom structure. It consists of two differential driving wheels for its movements and a passive direction wheel for its balance. It also has a 170-degree sonar sensor and a laser sensor with a similar range to detect the distance of surrounding objects. In our designed system, we connect pioneer pedestal to computer and Kinect-V2 camera to provide human user visual following service. We use the RGB data from KinectV2 SDK to detect human user instead of other distance sensors for reducing the cost.

Pioneer robot changes the direction of movement by setting the speed of two driving wheels, which makes the robot cannot move sideways, like described in figure 5.5(b). So when we design mobile SAR plays, the host moving forward and backward will not interact with the virtual content. In fact, the host rarely moves forward or backward compared to moving left and right in broadcasting or demonstration programs. Moreover, this problem can be solved in future mobile projection SAR robots development: When the movement of the mobile platform is not restricted, we can use the depth camera or other cheap sensors to detect the distance between the host and the camera, and design the corresponding interactions, like zoom-in and zoom-out.

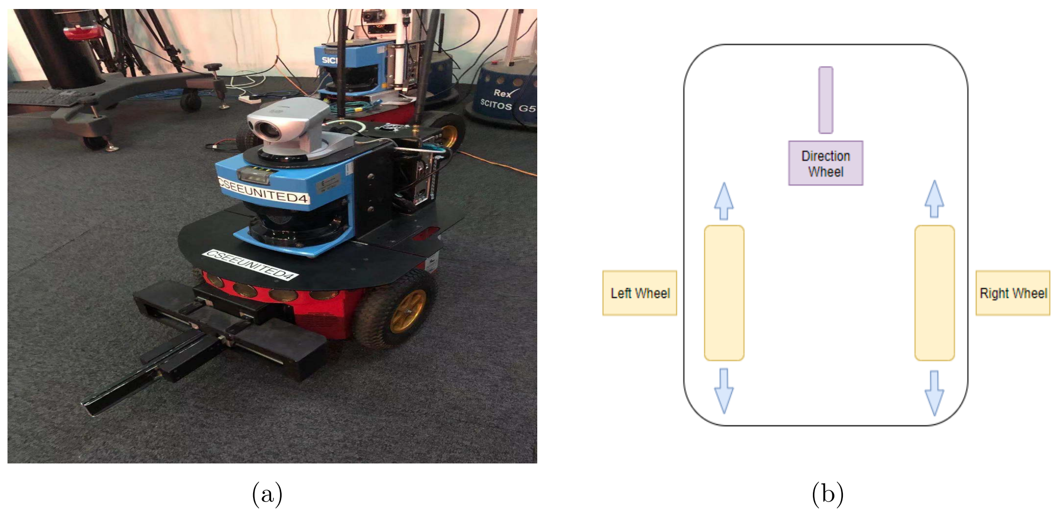


Figure 5.5: Pioneer 3 – Ground mobile platform

5.2 Projection Scene Registration

Indoor broadcasting scene and demonstration scene is always empty but including some large-size empty planes. In existing projection SAR system, scene registration is accomplished by the structured-light projection during calibration process, but this approach is impractical for mobile projection SAR robot. Another problem is that the working environment of projection SAR always contains some visual features and geometric features. State-of-art visual mapping algorithm may be applicable, but the accuracy and robustness of its mapping performance cannot be guaranteed due to uneven distributed features. Based on these two problems, we have proposed a plane-based visual mapping algorithm described in Chapter 4 as an optimisation.

Our algorithm can handle two different situations. On the one hand, when the scene contains enough visual features and these features are evenly distributed, our algorithm can first reconstruct the scene by ORB-SLAM, and then optimise the trajectory and scene model by the planes in the scene. On the other hand, when the number of features in the scene are insufficient, our algorithm can reconstruct the scene around a plane by tracking its contour. If the scene contains multiple planes, the algorithm can divide the entire scene into several regions containing one plane, and then stitch them into a completed scene model.

Our planar-based reconstruction algorithm has another benefit. The scene model in Unity3D is colorless, and the existing method to render multiple images in one large model is very time-consuming and difficult. The proposed mapping approach solves this problem by reconstructing the contour model during mapping and then marking these contours on scene model in Unity3D. Users can distinguish objects by contour and design different virtual contents in different zones.

5.3 Autonomous visual sensor calibration

Structured-light projection is common approach for calibrating the projector-camera unit, resulting in three important results: the intrinsic parameters of camera and projector (principle point and focal length), extrinsic parameters of camera and projector and a registered projection scene model. The first result is describing the parameter of visual sensor itself, like the focal length and lens distortion. The second result is describing the spatial relation between two sensors and the poses of two sensors relative to the scene respectively. The last result is describing the 3D detail of the scene and is replaced with our proposed visual mapping algorithm.

The calibration by using structured-light projection requires the projector facing a non-flat area with some rigid visual features and the camera covering the projected images. It will not work if the visual sensor unit is facing an empty and flat plane. Thus, it is necessary to design an intelligent calibration scene choosing algorithm for the mobile system to find an appropriate area for sensor calibration.

5.3.1 Intelligently choosing calibration-area

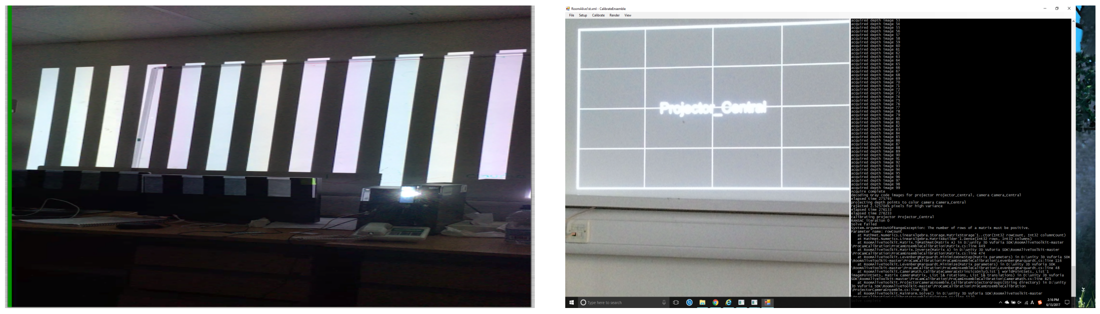
Structured-light projection requires solid visual features in its working view. So, the number of solid ORB features in every key-frame is counted while the system is running in mapping stage. The feature number will be main reference basis for calibration scene selection. Though the visual mapping algorithm is running in feature-less scenes, the point-feature are used for a reference instead of line feature because of two main reasons: one reason is that the point feature is very basic and its number will be more than the number of line features. Another reason is that point features are more reliable, especially when the plane is partly occluded or only part of it appears in images.

While selecting the best calibration target, the entire scene is divided in regions by key-frame and its nearest key-frame neighbours. This is done because when looking for a calibration area, it is easier to face camera to an area and not aim at an image. Moreover, counting the features in a region is more reliable than using a single image. Therefore, in an input image sequence, we count the total amount of solid features on a key-frame and its four key-frames neighbours (2 key-frames before it and 2 key-frames after it), and then estimate their average feature number with 'DBoW3' library. This area is added to the collection of calibration candidates only if both values are reaching the acceptable maximum threshold. For the scene without enough features. Since the mapping process in this scene does not generate key-frames, the whole image sequence will be divided by 7, and each calibration candidate has 7 frames. This number is variable according to the camera moving speed and the scale of scene.

Then, in the calibration candidate group, the number of normal frames between the first and last key-frames in each candidate is used as a reference for secondary filtering, and the region with smallest number of frames will be selected as the best area. This process is designed to minimise the influence of counting the feature between long distance key frames. We also improve the rationality of using feature points as the main reference for calibration scene choosing. To be mentioned, the DBoW3 database is pre-trained with the all the images in image sequence.

5.3.2 Sensor calibration with structured-light projection

After the suitable calibration area is defined, the system will be moved to face this area manually and start the calibration process. In this part we rely on the "RoomAlive" SDK from Microsoft (Jones et al., 2014). RoomAlive provides a complete camera-projector unit calibration process that not only supports calibration of individual sensor units, but also supports collaboration between multiple units. At work, the system generates a series of bar codes from large to small, and then project these image in sequence on the scene with the solid feature. The internal and external parameters of the both visual sensors can be calculated by the distortion of the projected image. The experimental result of calibration is described in figure 5.6.



```
acquired depth image 53
acquired depth image 54
acquired depth image 55
acquired depth image 56
acquired depth image 57
acquired depth image 58
acquired depth image 59
acquired depth image 60
acquired depth image 61
acquired depth image 62
acquired depth image 63
acquired depth image 64
acquired depth image 65
acquired depth image 66
acquired depth image 67
acquired depth image 68
acquired depth image 69
acquired depth image 70
acquired depth image 71
acquired depth image 72
acquired depth image 73
acquired depth image 74
acquired depth image 75
acquired depth image 76
acquired depth image 77
acquired depth image 78
acquired depth image 79
acquired depth image 80
acquired depth image 81
acquired depth image 82
acquired depth image 83
acquired depth image 84
acquired depth image 85
acquired depth image 86
acquired depth image 87
acquired depth image 88
acquired depth image 89
acquired depth image 90
acquired depth image 91
acquired depth image 92
acquired depth image 93
acquired depth image 94
acquired depth image 95
acquired depth image 96
acquired depth image 97
acquired depth image 98
acquired depth image 99
Acquire complete
decoding Gray code images for projector Projector_Central, camera Camera_Central
elapsed time 275793
projecting depth points to color camera Camera_Central
rejected 2.525704% pixels for high variance
elapsed time 276133
elapsed time 276233
calibrating projector Projector_Central
RANSAC iteration 0
Solve failed
System.ArgumentOutOfRangeException: The number of rows of a matrix must be positive.
Parameter name: rowCount
   at MathNet.Numerics.LinearAlgebra.Storage.MatrixStorage`1..ctor(Int32 rowCount, Int32 columnCount)
   at MathNet.Numerics.LinearAlgebra.MatrixBuilder`1.Dense(Int32 rows, Int32 columns)
   at RoomAliveToolkit.Matrix.ToMathNet(Matrix A) in D:\unity 3D Vuforia SDK\RoomAliveToolkit-master
\ProCamCalibration\ProCamEnsembleCalibration\Matrix.cs:line 449
   at RoomAliveToolkit.Matrix.Inverse(Matrix A) in D:\unity 3D Vuforia SDK\RoomAliveToolkit-master
\ProCamCalibration\ProCamEnsembleCalibration\Matrix.cs:line 474
   at RoomAliveToolkit.LevenbergMarquardt.MinimizeOneStep(Matrix parameters) in D:\unity 3D Vuforia SDK
\RoomAliveToolkit-master\ProCamCalibration\ProCamEnsembleCalibration\LevenbergMarquardt.cs:line 116
   at RoomAliveToolkit.LevenbergMarquardt.Minimize(Matrix parameters) in D:\unity 3D Vuforia SDK
\RoomAliveToolkit-master\ProCamCalibration\ProCamEnsembleCalibration\LevenbergMarquardt.cs:line 48
   at RoomAliveToolkit.CameraMath.CalibrateCameraExtrinsicsOnly(List`1 worldPointSets, List`1
imagePointSets, Matrix cameraMatrix, List`1& rotations, List`1& translations) in D:\unity 3D Vuforia
SDK\RoomAliveToolkit-master\ProCamCalibration\ProCamEnsembleCalibration\CameraMath.cs:line 825
   at RoomAliveToolkit.ProjectorCameraEnsemble.CalibrateProjectorGroups(String directory) in D:\unity
3D Vuforia SDK\RoomAliveToolkit-master\ProCamCalibration\ProCamEnsembleCalibration
\ProjectorCameraEnsemble.cs:line 766
   at RoomAliveToolkit.MainForm.Solve() in D:\unity 3D Vuforia SDK\RoomAliveToolkit-master
\ProCamCalibration\CalibrateEnsemble\MainForm.cs:line 1120
Solve complete
```

(a) (b)

Figure 5.6: Structured-light projection. (A) describes the projected pattern. (B) describes the calibration result.

5.4 Plane based camera initial Pose estimation

The initial pose estimation of the proposed mobile system can be considered as a re-localisation problem. Most of re-localisation problems in the state-of-art visual tracking and mapping algorithms are solved through DBoW library, which is based on point feature or line features. This makes the re-localisation problem extremely dependent on the number and quality of visual features. Therefore, the problem of re-localisation in a low-texture environment is more challenge.

Computer vision-based relocation problems can be divided into two continuous processes: finding nearest neighbour and matching the neighbour. More specifically, the finding nearest neighbour process aims at finding the nearest frame through comparing the visual feature of new added frame and visual feature dictionary with 'DBoW3' library. The matching process is estimating the transformation matrix between newly added frame and its nearest neighbour, which is typically achieved through visual tracking algorithms. However, since this algorithm relies on the number and quality of solid visual features, it is difficult to apply in low-texture environments or on images with a low quality.

Based on this limitation, a novel re-localisation algorithm is proposed in this research based on DBoW, CNN scene parsing and point-point ICP algorithm. Similarly, our algorithm has two main steps: (i) feature training and matching with DBoW and (ii) edge-based dense tracking on landmark plane. Two processes is illustrated in figure 5.7 and figure 5.8 respectively. In this experiment, $harvard_c11/hv_c11_1$ is used as the visual mapping dataset and the first frame in $harvard_c11/hv_c11_2$ is assumed as the initial perspective when re-localising the robot. More detailed explanations are given below:

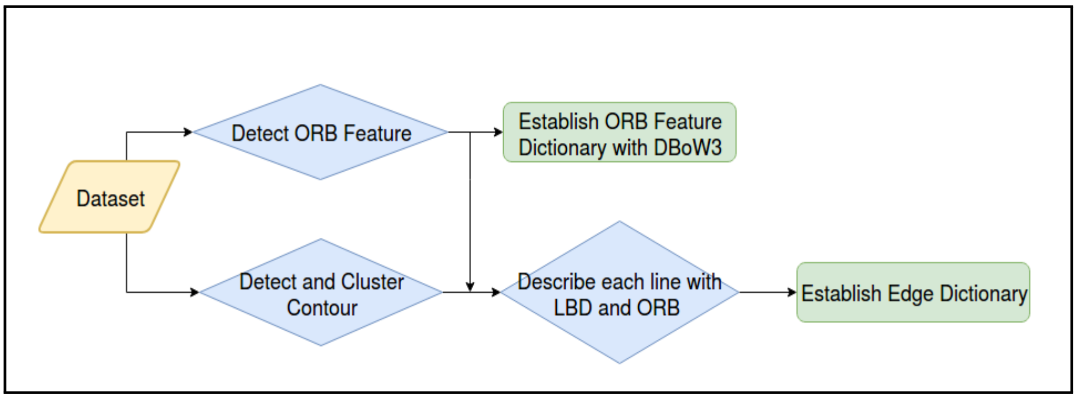

Dataset
Detect ORB Feature
Detect and Cluster Contour
Establish ORB Feature Dictionary with DBoW3
Describe each line with LBD and ORB
Establish Edge Dictionary

Figure 5.7: A flowchart describes the feature dictionary algorithm

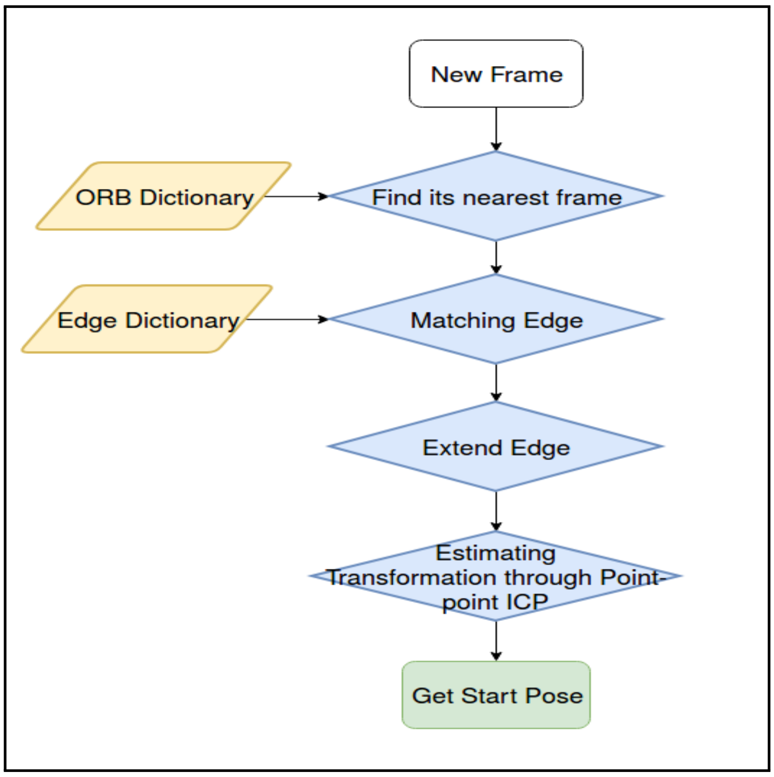


Figure 5.8: A flowchart describes the proposed pose initialisation algorithm

(i) **Feature Training with DBoW3**

We use ORB to detect and describe the point feature, and use LBD to describe and discipline the edges in this step. These two features are saved in different libraries. The point feature library aims at searching the nearest frame in dataset and finding the feature matches, and this process is same to state-of-art feature-based re-localisation algorithm. On the other hand, edge dictionary aims at describing and distinguishing each edge of the detected landmark plane. The establishing process of edge dictionary could be described in three steps:

- When a new frame is inserted during visual mapping process, its landmark plane and contour will be extracted by using our proposed contour extraction pipeline described in Chapter 3. Then each edge will be extinguished and clustered with its direction. In most cases, the two parallel edges will have different directions eventually when viewpoint changes.

- Each edge is described with point and line descriptors. Firstly, ORB descriptor is used to describe two endpoints of an edge, and then the LBD is used to describe each edge. Finally, these information is saved into dictionary.

- In our visual mapping algorithm, the first frame is considered as the global reference, whose detail is saved to dictionary as a reference. When a new frame is inserted and its edges are analysed, we compare these features with the dictionary: if either of point-feature similarity or edge-feature similarity is acceptable, we will update the point features to dictionary. If none of these two features are acceptable, we save the new edge data to a edge feature vector buffer. Then, the new inserted frame will be compared with dictionary and edge feature vector buffer at the same time. If the new inserted frame has the similar edge feature with edge feature in dictionary and vector buffer, we will consider this newly added edge is reliable and insert it to the edge feature dictionary. Note that both ORB features and LBD features are not affected by inaccurate pose estimation result.

This process derives point feature dictionary and edge feature dictionary, which will be applied to find the nearest frame and matching the same edge in two nearest frames in the following step.

(ii)**Edge-based dense matching algorithm**

After defining the nearest frames from the dataset for newly added frame, this step will try to find the spatial transformation between new frame and its neighbour. Different to the existing image feature based approaches, we track the 3D contour feature of landmark planar area as an optimisation. However, due to the small number of contour points and long distance between two contour models, it is impossible to use ICP algorithm to match them directly.

Therefore, we have made three assumptions:

- Assuming that each edge on the contour has the same position and orientation in the world coordinate at different camera angle.
- Assuming that the normal vector of landmark plane is not changed at different viewpoints
- Assuming that each edge is extended along the landmark plane‘s normal vector, and the normal vector of these extended planes are same at different viewpoints in world coordinate.

This is illustrated in figure 5.9 where the $Plane_{Table}$ is the detected landmark plane, and its normal vector is N_{Table}. A is one of the whole edge and a is the visible edge of A in current perspective. The $Plane_A$ is the extended plane along N_{Table} from A. Edge B and C are same as A.

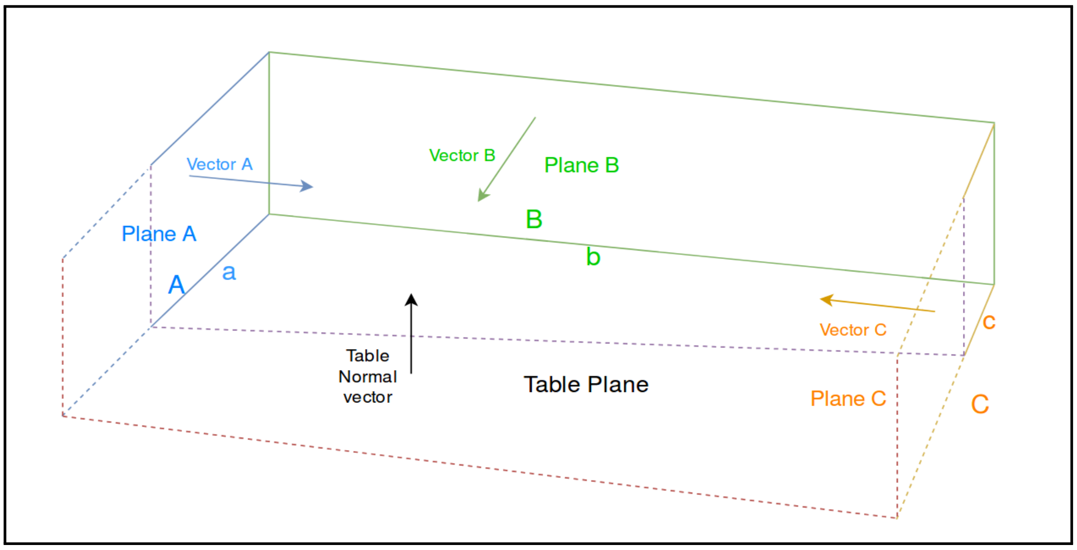


Figure 5.9: Explanation of the proposed algorithm.

As illustrated in figure 5.9, each edge has a visible part and a non-visible part. Since these two parts are from the same line, their extended plane along the normal vector direction should on the same plane. In addition, assume that the extended plane from visible edge is $ExPlane_A$, the normal vector of $ExPlane_A$ should be as same as the normal vector of $PlaneA$. Other edges are same as edge A. Then, we can build three constraint functions based on the normal vector function of two edge plane in different directions and one landmark plane. Therefore, our algorithm can work in the case when only two edges are visible, as shown in figure figure 5.10(a). The structure and main processes of our proposed pose initialisation algorithm are illustrated in figure 5.8. Note that every important process (drawn in blue colour in figure 5.8) is explained in details and the experimental results are presented.

- **Process 1**: Finding the nearest frame
  This process aims at finding the nearest frame for newly added frame with point feature only, which is similar to state-of-art algorithms based on DBoW. The result is described in figure 5.10: figure 5.10(a) is the new added frame and figure 5.10(b) is its matched nearest frame in $harvard_c11/hv_c11_1$.

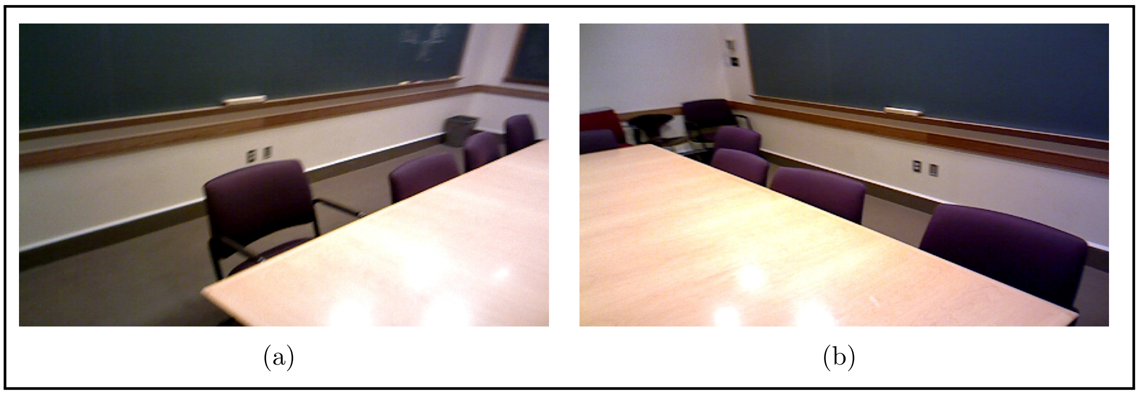


Figure 5.10: New added frame and its nearest frame in dataset. (a) describes the first frame in $harvard_c11/hv_c11_2$, and this could be present as the initial perspective of a robot. (b) is the nearest frame in $harvard_c11/hv_c11_1$ matched by point features and DBoW3.

- **Process 2**: Matching edge between new frame and its neighbour
  This process is to match the edges of the landmark plane displayed in two frames. This step starts with checking the edge feature in edge dictionary for both images, and then match edges with same feature. In this experiment, the landmark plane is the table with 4 edges: three edges are visible in nearest frame from $harvard_c11/hv_c11_1$; and two edges are visible in new added frame from $harvard_c11/hv_c11_2$. Figure 5.11 has described the matching result, and the matched edges are drawn with same color.

- **Process 3**:Construct and Extend edge model with normal vector
  After edge detection and matching, we reconstruct the 3D model of each edge with the algorithm introduced in Chapter 3 to minimise the error caused by inaccurate depth data. Then, we extend each edge along the normal vector direction of landmark plane, which are described in figure 5.12. In this experiment, we set the distance value between 2.5cm to 5cm from contour points and the point in extend model, and the distance interval between extended point is 0.1cm.

- **Process 4**: Establish constraints and estimate transformation
  This step aims at establishing three constraints functions based on normal vec-

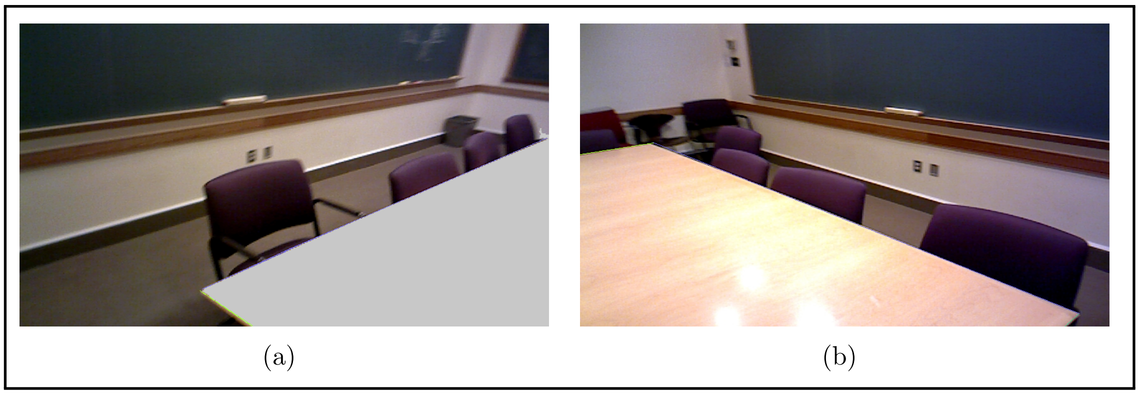


Figure 5.11: The edge matching result between two images. (a) describes the first frame in $harvard_c11/hv_c11_2$*, two edges are drawn in light green and purple. (b) is the nearest frame in* $harvard_c11/hv_c11_1$*, the matched edges are drawn in light green and purple, and the unmatched edge is drawn in dark green.*

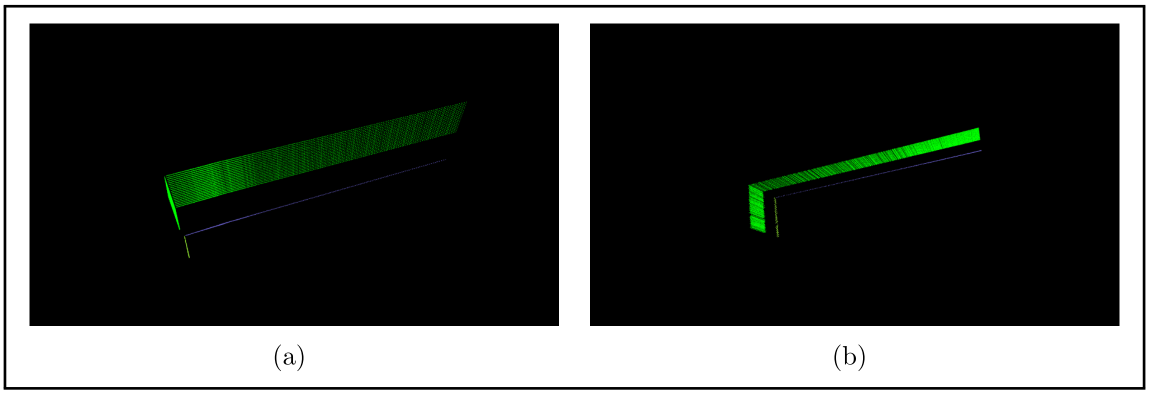


Figure 5.12: 3D point cloud of extend edges. (a) is the extend contour model for newly added image. (b) is the extend contour model for its nearest frame. We only display extend model of two edges in (b) to make experimental result clearer

tor equation and solve these functions. After constructing the extend model, we use RANSAC algorithm provided by PCL lib to estimate the normal vector of each extended plane in the dataset image, and then build three constraint functions based on normal vector function of two extend plane and landmark plane.

For each plane, assuming its normal vector equation is:

$$a\vec{x} + b\vec{y} + c\vec{z} + d = 0; \tag{5.1}$$

$$residual[n] = a * (P1_x - P_x) + b * (P1_y - P2_y) + c * (P1_z - P2_z) + d \tag{5.2}$$

Then the transformation matrix is derived by solving these functions with CERES lib (figure 5.13).

The experimental result shows that our algorithm is practical for solving the re-

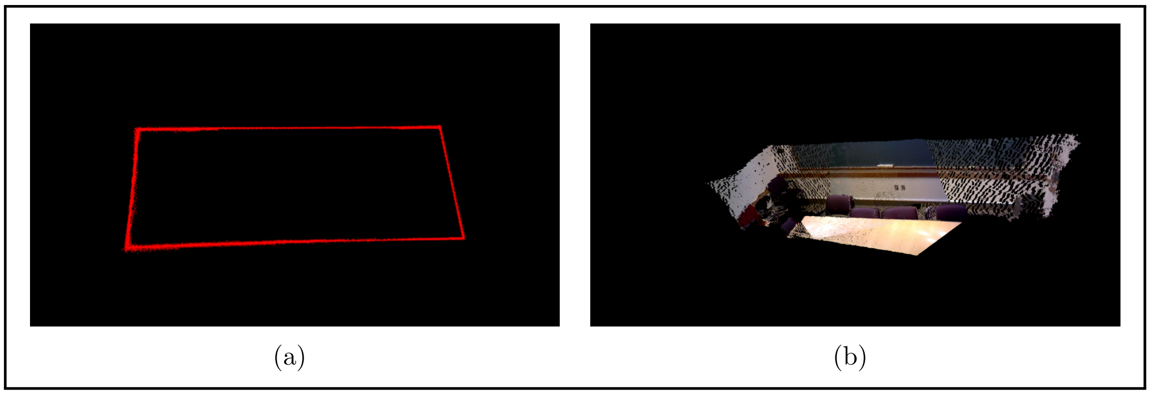


Figure 5.13: *Initial pose estimation result. (a) describes the combined contour model between initial frame after transformation and landmark contour model. (b) describes the combined scene model between initial frame after transformation and the first frame in dataset.*

localisation problem. Compared with the state of art re-localisation algorithm, we have made two main improvements: firstly, our algorithm is not very dependent on the number of solid feature points in the image and applicable in low-texture environment. Secondly, our algorithm is only relying on single global landmark planar area rather than relying on multiple small planes like the existing algorithms. Moreover, the landmark plane could be partly visible. This makes our algorithm more valuable and practical in simple indoor scenarios (broadcasting and demonstration studio). The main bottleneck is our algorithm requires at least two edges are visible in both frames.

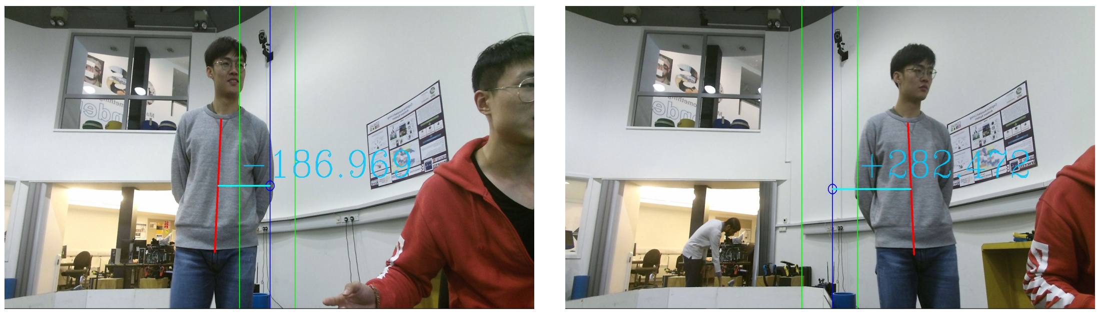
5.5 Autonomous Presenter Following Service

This section introduces our human following service approach. For mobile SAR system, presenter following process is more complex because we need to connect the external robot and Unity3D. While the external robot is moving, the simulated robot in Unity3D should move simultaneously. And this will ensure the robot can project correct virtual content in different areas. In the following content of this section, the human user following approach based on visual sensor is described in the first sub-section. Then, the moving distance estimation of mobile pedestal is suggested in second part. Finally, a creative data transmission method between Unity3D and mobile robot is discussed.

5.5.1 The Presenter Following service

For the presenter following service in our proposed system, we use a Kinect-V2 RGB-D camera to detect and follow the presenter. Since the presenter always stands in front of the projection background, we use the Kinect skeleton detection library to locate and detect the torch of the presenter. While the presenter is moving, a line segment perpendicular to the middle line of the image and connected to the midpoint of detect torch is used to estimate the moving distance. When the distance is larger than the threshold, the system will give moving order to the robotic pedestal to keep the presenter inside central area in camera's view.

It should be noticed that the moving speed is limited with maximum 20 cm/sec to avoid the problem of motion blur and keep the system in balance. We set up a bounding box to allow the presenter to move freely: while the torch of present is inside this box, body gesture will not make the pedestal move. In the experiment described in figure 5.14, the bounding box (described by two green edge) has 100 pixel distance to the central line (blue line). The current camera's view is in 1920 * 1080 scale.



(a) (b)

Figure 5.14: The presenter following service. In both figures, the blue line is the central perpendicular line and the Green lines describes the bounding box and distance threshold.

Note that we have not designed the 'moving forward' and 'moving backward' commands to the pedestal since these operations requires to change projection parameters. So, we set a fixed speed and moving direction (in 'left' and 'right') to make the robot track human user stably and consistently. When the presenter moving left outside the box, the system will return a negative value and make robot moves to left. Otherwise, a positive value will lead to a right movement. if the presenter's torch is inside the box, the system will stop moving.

5.5.2 Real-time robot pose tracking

Since our purposed Projection-SAR robot only has simple actions and the robot moves slowly, we use the mobile platform to measure the moving distance directly. When the robot has moved, it calculates the speed of both wheels and the running period, and then send these values value to unity3D. Since there is no port between the Pioneer robot and unity3D, we have introduced a novel method with '.txt' file as an information transmission medium. This approach will be introduced in detail in the next subsection.

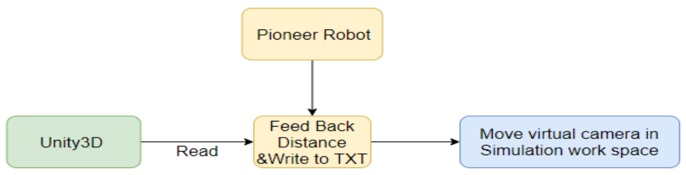
5.5.3 Data transmission between mobile robot and Unity3D

This section aims at solving the problem of simulating trajectory in Unity3D. Since Unity3D does not provide libraries to connect with external robot, and it is impossible to implement the visual tracking or mapping algorithms in Unity3D directly since it is too complex and Unity3D lacks of methods to store dynamic information. We use '.txt' files as a medium for information transmission. Whenever the robot moves, it will store its moving direction and distance in a new '.txt' file. At the same time, Unity3D will read this file and extract the robot's movement information. Then it uses this information to move the virtual camera and projector in the simulation work-space.

By using this method, we can successfully let the camera and projector in the virtual space to simulate the movement of the external devices. Since Unity3D cannot read '.txt' file with changing values, each move of mobile robot generates a new file. After Unity3D starts working, it will wait for a new '.txt' file. When a new file is generated in a specified file directory, it will find the new file and read the moving parameters. If the presenter is static and the robot is not moving, it won't generate the new '.txt' file. The unity will only track user's upper body gesture and project corresponding virtual dynamic contents. The whole data writing and reading process is illustrated in figure 5.15.

5.6 Immerse Virtual Plays

Projection-SAR virtual content is designed with Unity3D, which is the most popular and professional AR developing tool in recent years. Compared to other software, Unity3D has three main benefits. Firstly, it is a mature AR development platform


Pioneer Robot
Unity3D
Read
Feed Back Distance &Write to TXT
Move virtual camera in Simulation work space

(a)

Figure 5.15: The moving data transmission process.

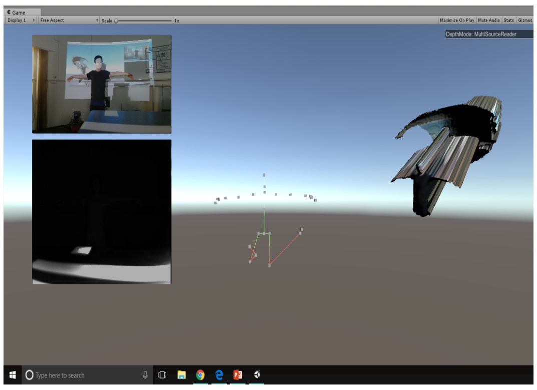
and has a full-featured user interface. Secondly, most of recent popular computer vision algorithms or software have Unity3D software develop tool-kits. Thirdly, Unity3D is opened for public use. In our SAR robot, we use Kinect Unity3D SDK and RoomAlive Unity3D SDK from Microsoft (Jones et al., 2014). These two software packages not only successfully introduce cameras and projectors in unity3D, but also contain almost all of the Unity development software based on computer vision and human body recognition.

5.6.1 Interaction with Virtual Model

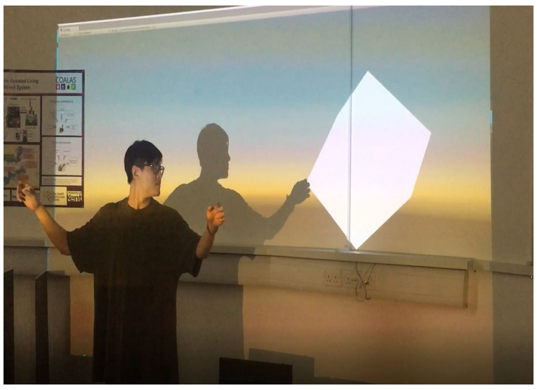
In order to create a human-computer interaction experience for the presenter, we have designed a play to control the virtual object through tracking the human body gesture. In this design, assume the object is landing at the original of Cartesian coordinate. The presenter can control the virtual object to move in X, Y and Z directions by moving right arm, and is also able to control the rotation with X, Y and Z axes with left arm. Figure 5.16 shows the control of the object with body gesture, provided by Microsoft unity SDK and RoomAlive SDK, in which (A) is Human skeleton reading and (B) shows the use of body gesture to control 3D movement of a projected virtual model.

5.6.2 Interaction with Virtual Scene

In this design, we let the presenter to control the movement and scale of virtual object by moving the body in front of the camera. While the presenter is standing in front of the camera, we can detect the distance with depth sensor on KinectV2.



(a)

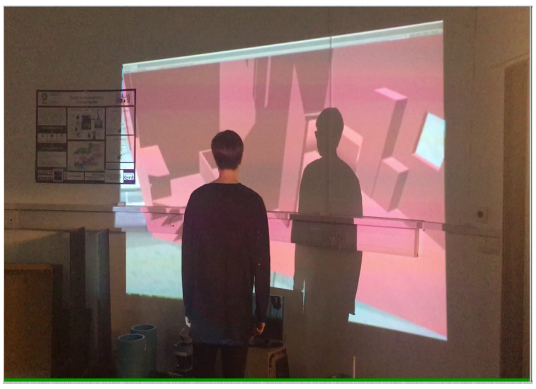


(b)

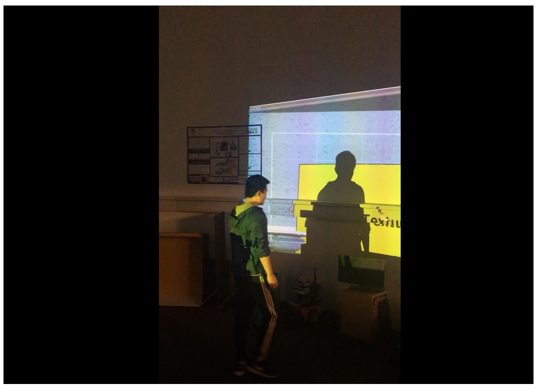
Figure 5.16: To control the object with body gesture, provided by Microsoft unity SDK and RoomAlive SDK.

So, when the presenter is moving to the camera, the distance between camera and presenter will be reduced, which tells the system to zoom in the virtual content. While the presenter is moving away from the sensor, the system will zoom-out the virtual content. While the presenter is moving left or right, the focus of view will concentrate on these directions, and then rotate and zoom in these two different parts in these two directions.

Figure 5.17 shows the contour of the object projected on background. The projected virtual content will be adjusted while the user is moving in front of the camera. More specifically, (A) shows the presenter controls the projected model. (B) shows the presenter controls the projected 3D graph. Note that this AR play requires the user move inside camera's view, so robot will stop presenter following service until user finishes current AR play.



(a)



(b)

Figure 5.17: To contour the object projected on background. The projected virtual content will be adjust its pose while the user is moving in front of the camera.

5.6.3 Mobile Interaction with Virtual Background

As an innovation, we have designed a new virtual interactive play specified for a mobile SAR system. Different to existing SAR application, we will not use the image captured by camera as a part of the projection output.

Before Unity3D runs, user needs to insert registered SAR system and reconstructed scene model into Unity3D simulation work-space. The pose of calibration scene, camera and projector are defined in calibration process, and the pose of reconstructed scene is estimated with pose initialisation process described in Chapter 5.4.2. Note that user needs to adjust the pose parameter of scene model to make it correctly match the calibration scene, this tuning process would not take a long time. Another parameter that needs to be adjusted by the user is the movement trajectory of the virtual projector. Because most of the working environment of mobile projection SAR work are flat, users only need to let the virtual projector horizontally in two directions. By adjusting the moving speed of the virtual projector, the user can make it accurately imitate the trajectory of real robot and project the correct content. This step does not require users to make program by themselves. In Unity3D, we can design a user interface to make user design the path intuitively. The main purpose of this design is to minimize the influence that the changing tangential distortion brings to projected image. The characteristic of tangential distortion is introduced in Section 2.3. When the projector moves, its tangential distortion parameter obtained from calibration process changes and no longer accurate for current viewpoint. Therefore, we let the robot move only in the parallel direction with the projection plane, and let its projection direction always be perpendicular to the projection plane. In this way, errors caused by imprecise tangential distortion can be minimised.

After scene building in Unity3D, user can design the attached virtual content and virtual models in the simulation work-space. For example, we attach green virtual texture around the post area on the wall. Figure 5.18 shows the reconstructed scene and the attached virtual content, in which (a) is the enhanced model in which the green texture area has changed the appearance of picture area, and (b) describes that the presenter can move the virtual camera with Cartesian coordinate to generate different views.

Then, user can run the Unity3D and start the presenter following service. While the presenter is moving left or right, the robot will recognise the presenter's movement and order the robotic pedestal to move the corresponding distances. As soon as the robot has recognised it has entered the region which the post is visible in current camera's perspective, it will project part of the virtual texture over the visible part of post and change its appearance. This design has fundamentally created the mobile nature of projection-SAR system.

Figure 5.19 shows the different views when the projector is moving. In this experiment, we attach the virtual green label on the pose to change its appearance. We can see when we change the position of the projector, different scale of the virtual content has been projected on the post to change its appearance. Note that since we aims at solving the mobility of a Projection-SAR system, We do not focus on improving its virtual effects.

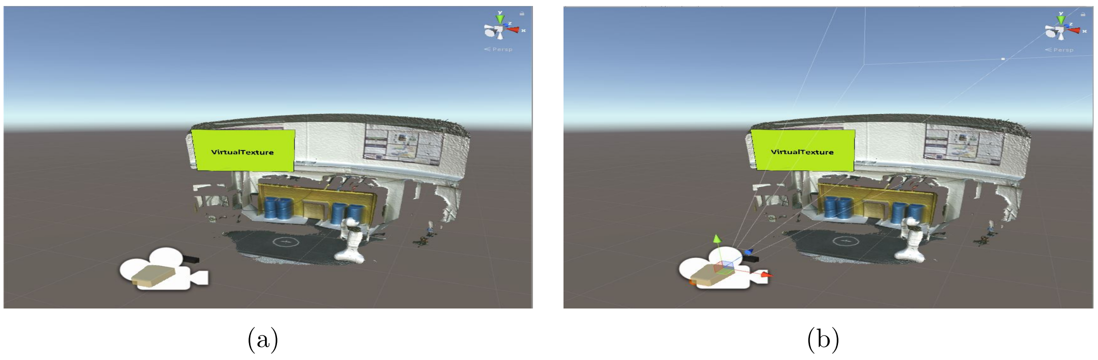


Figure 5.18: The reconstructed scene and the attached virtual content.

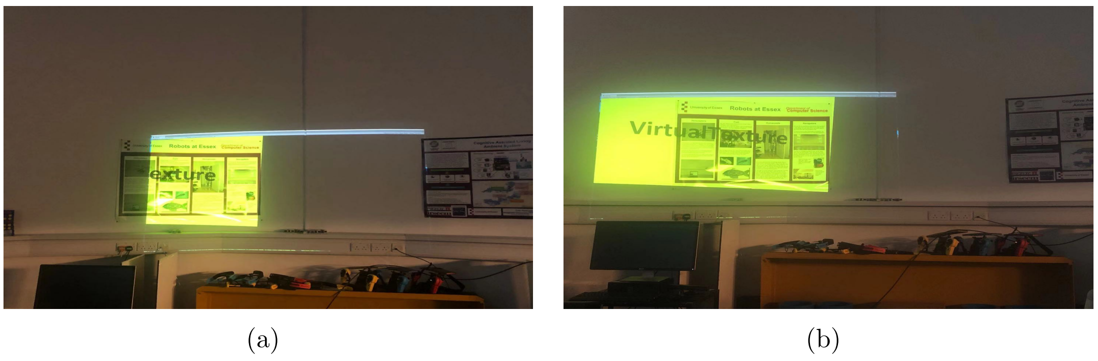


Figure 5.19: The different views when the projector is moving.

5.7 Summary

In this chapter, the entire system and the deployed sensors were firstly introduced. Then the entire workflow was briefly described and different sensors are assigned according to the work division in different parts of the system. Finally, three different virtual applications were designed to prove the rationality and practicability of the entire system. In our proposed system, one RGB-D camera, one projector and a

Pioneer3 ground mobile robotic pedestal were deployed. Before AR design, the entire scene was reconstructed through the plane-based visual mapping algorithm described in Chapter 4 in priority, and the reconstructed model was inserted into Unity3d. Finally, we attach plane model on scene model to make colour-less scene model more readable in Unity3D.

Then the system starts working after AR design and the presenter was distinguished from the background by detecting his torch. The RGB information is applied to estimate the presenter's moving direction. When the robotic system runs, its pose was roughly updated in the scene through the wheel movements feedback from the robotic pedestal and the dynamic virtual image was projected based on its pose corresponding to the scene.

The entire design process solves the current movement problem of the projection-SAR system and minimises the impact of the tangential distortion on the projection images. It should be noted that the RGB image captured by the camera for projection differs from the state-of-art projection-SAR as we want to use projection to change the appearance of the environment. The only problem of the proposed system is it has not solved the problem of the changing tangential distortion of the projector at different projection angles, resulting in our AR play without more free movements. This will be addressed in the future research.

6

Practical Experiments in Indoor Broadcasting Scene

The proposed projection-SAR robotic cameraman will be tested in a simulated indoor broadcasting scene in this chapter, aiming at verifying the applicability of the proposed mobile robotic system and comparing its performance with the existing fixed projection-SAR application. This chapter consists of six sections and is organised as follows. Section 6.1 introduces the basic equipment of the proposed robotic system. The simulated environment and its CNN scene parsing model are introduced in Section 6.2. Section 6.3 describes the contour extraction and modelling. The scene reconstruction and real-time tracking are presented in Section 6.4. Section 6.5 demonstrates the designed interactive virtual plays provided by the proposed mobile robotic system. A comparison between a mobile SAR and a fixed SAR is given in Section 6.6. Finally, a brief summary is given in Section 6.7.

6.1 The proposed robotic platform

6.1.1 Structure of robot

Figure 6.1 shows the whole Robotic platform. A visual camera is placed at top for tracking the presenter. The DLP projector and computer are connected and placed in the middle platform, and the mobile platform is at the bottom. All visual devices and the mobile robot are connecting to a computer, the connections between sensors are described in figure 6.2.

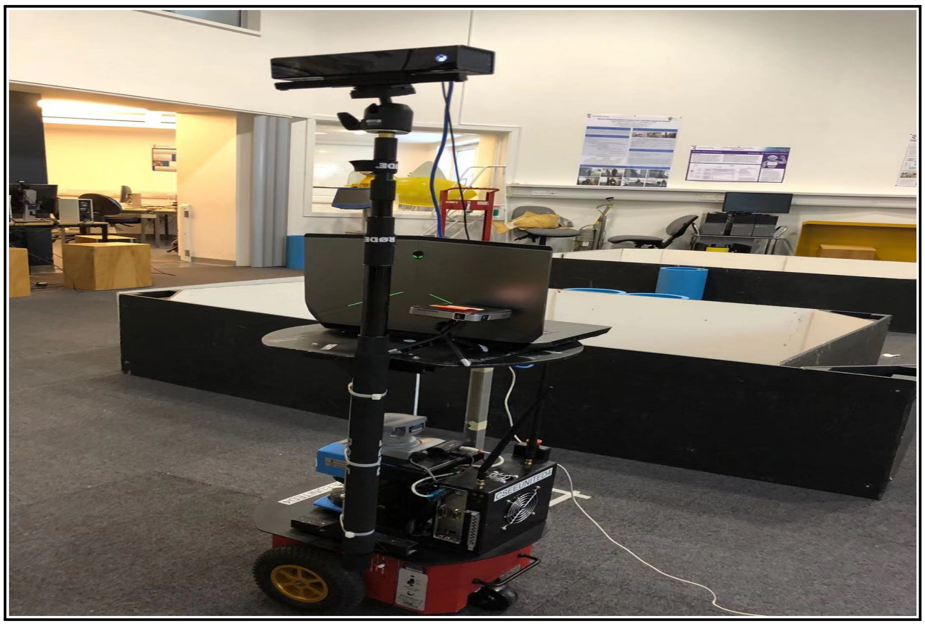


Figure 6.1: The whole mobile robotic system.

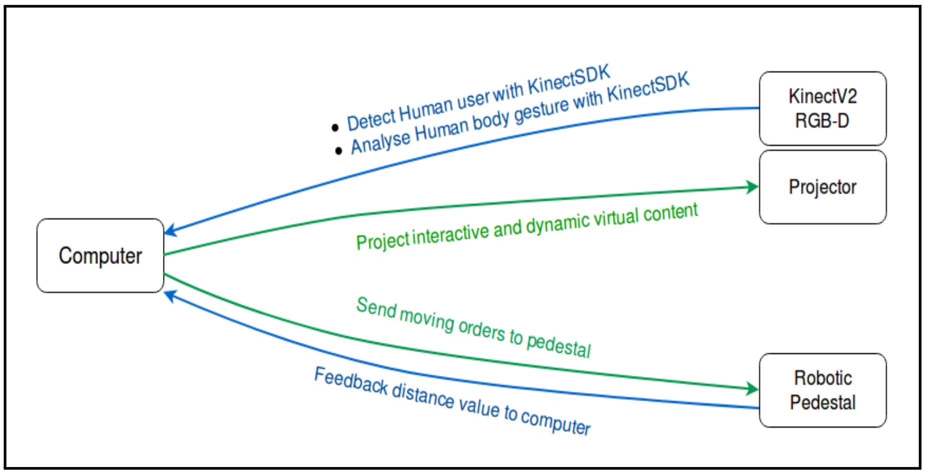

Computer
KinectV2 RGB-D
Projector
Robotic Pedestal
• Detect Human user with KinectSDK
• Analyse Human body gesture with KinectSDK
Project interactive and dynamic virtual content
Send moving orders to pedestal
Feedback distance value to computer

Figure 6.2: The connections between computer and sensors. The camera detects human user and analyses body gestures, then it sends these information to computer. The computer orders the robot to move according to detected information, and the robot will feed back the movement information to the computer at the same time. Finally, the computer projects the interactive virtual content on background through the projector according to the distance moved and the human user's body gesture.

6.1.2 Working approach

The proposed system includes two working stages as follows.

- In the preparation stage, both RGB and Depth image sequences are captured for recovering the 3D model of working scenario. After scene reconstruction, system starts calibration process and pose initialisation process. Then, the calibrated SAR system and scene model are inserted into Unity3D. Note that user needs to set the moving path and speed of virtual robot to make it simulate the trajectory of real robot. Since these value are displayed in an user interface in Unity, this process would not take a long time. Finally, the virtual content and 3D models are attached onto the scene in simulation work-space.

- In the working stage, while the system is moving, the camera captures the presenter's body movements and give orders to Unity Engine to project the dynamic virtual content and to the robotic pedestal to move respectively. As soon as the pedestal moves, it will count the moving distance and feedback the distance value back to Unity Engine. The simulated projector in Unity Engine will move according to the feedback distance, and then change the projected virtual content.

Different to the existing projection-SAR equipment, purposed projection SAR is not required to project current captured RGB image to the scene because it aims at changing the appearance of the scene. In addition, since the tangential distortion of projected image changes after moving the projector, the projector always face the projection target perpendicularly and the moving track of proposed robot is fixed to minimize its influence.

6.2 Simulated indoor demonstration Scene and CNN scene parsing

This section presents the experimental environment and the CNN scene parsing result. In the first subsection, the experimental scene and its features are described. Then, the CNN scene parsing result will be present in the following subsection.

6.2.1 Simulated indoor scene

Figure 6.3 shows a part of our robotic arena that is used as the simulated broadcasting scene in this experiment. The robotic arena is one of the laboratories in School of computer science and Electronic Engineering at University of Essex. It is a large, empty, circular and large-scale indoor space similar to the broadcasting room. Furthermore, the robotic arena is commonly used for demonstration through projecting videos on the wall, we can compare the performance of our proposed mobile system with the existing fixed projector.

The RGB and Depth images captured by KinectV2 do not match each other: the

Figure 6.3: Create an environment of broadcasting.

RGB images are 1920 * 1080 and depth images are 512 * 424, which should be change to 640 * 480 for 'DeepLabV3+' model. More over, the data are always horizontally reverse to the real scene. Thus, for all captured raw RGB images, the system flips them horizontally firstly, and then cut and resize them based on the corresponding depth data to 640*480 scale. This process could be described in figure 6.4

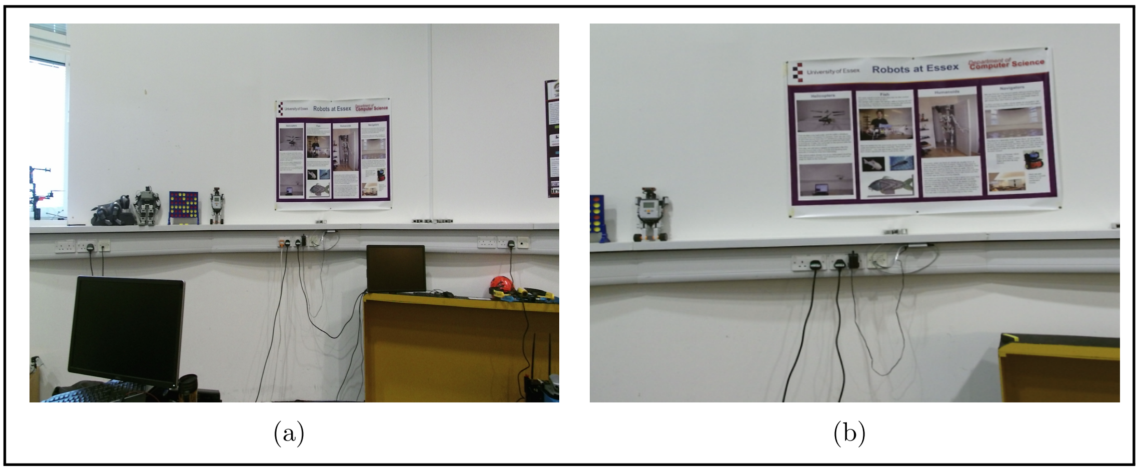


*Figure 6.4: Transferring Raw RGB data to usable RGB data. (a) describes the raw RGB data in 1920*1080 . (b)describes the usable RGB data in 640*480. The cutting position and resizing scale is based on matching corresponding depth data*

The depth data has two problems: (i) its working range is in 512 * 424 and two horizontal edges are blank in the captured raw data; (ii) the captured raw data cannot be directly used in scene mapping due to its inverse pixel value. Therefore, the captured raw depth data is cut firstly, and then each pixel value is subtracted with 65535 to get the usable data. Figure 6.5 has described the one example of subtraction process.

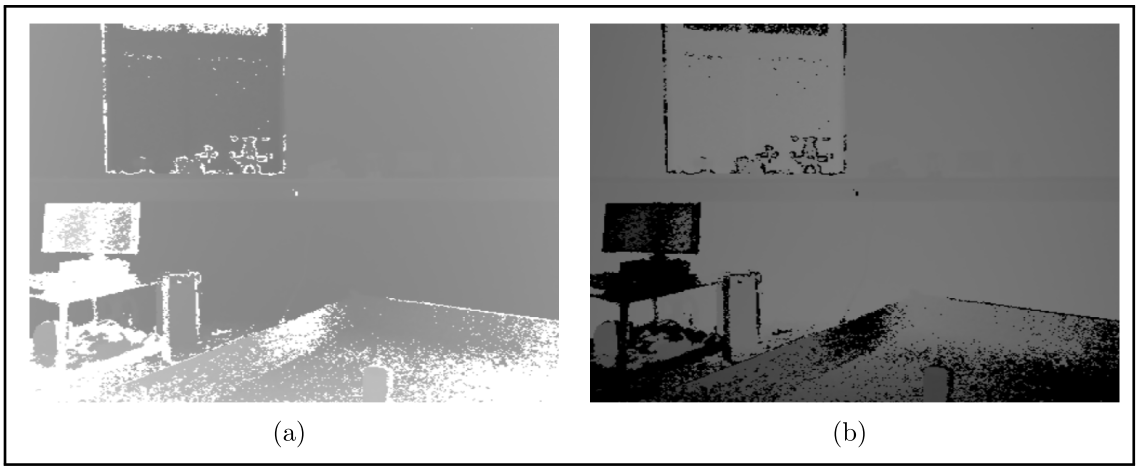


Figure 6.5: substation process of captured raw depth data. (a) is the raw depth image. (b) is the usable depth data after substation.

6.2.2 CNN scene parsing

Scene parsing aims to semantically recognise and segment the element in each captured RGB image. In our system, this is achieved by CNN scene parsing model 'DeepLabV3+' from Google. For the pre-trained model, we choose the model trained with dataset 'ADE20K' that includes most of the objects in indoor scenarios, which is able to detect most of planar objects like screen, blackboard, whiteboard and post.

ADE20K has 151 classes, which makes this CNN model confused between the similar categories. Figure 6.6 describes the CNN model has detected the posts on the wall, but it recognises them with various categories. In figure 6.6(a), the post is described as 'Picture' and described with purple colour. In figure 6.6(b), the post is described as 'Post' and 'Screen', filled with light green and dark yellow colour respectively. Though the label information is not accurate, the CNN scene parsing model still recognise the post on the wall and draw the post area roughly.

Although the CNN model does not provide us with accurate label information and object edge information, the provided segmentation result allows us to split the post in corresponding region and roughly separate it from background wall. Another problem is that the CNN model does not have high segmentation accuracy, especially when there is no obvious Gray-scale difference between the object and its background or the image is blurred. Thus, the segmentation result cannot be directly used to describe an object. We need to fuse all the information and then accurately describe the target object with our contour extraction algorithm.

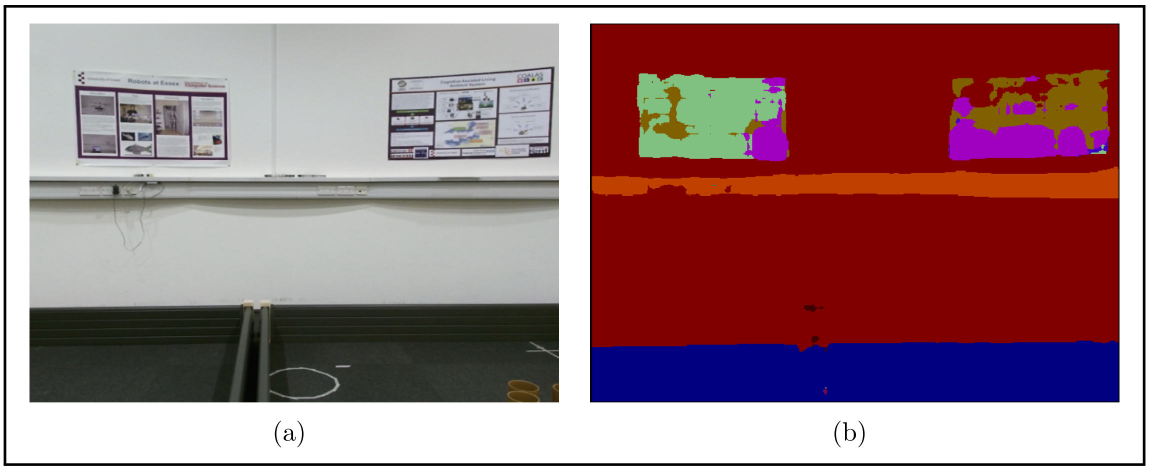


Figure 6.6: CNN Scene parsing results. (a) is the cropped RGB image and (b) is the segmented result. In (b), both posts are detected but assigned with different labels.

6.3 Contour Extraction and Modelling

In this section we will extract important contour information in experimental scene by using the algorithm described in Chapter 3. As mentioned in Chapter 3, we use 4 edge detectors to detect the lines and then fuse these segments to form the accurate contour. In the following content, we will show the results of each contour extraction process and then describe the final result through the figures. We need to mention that the resolution of captured image in this subsection is different to the subsections before, since we need to re-scale the captured RGB image from 1920 * 1080 to 640 * 480 to match the captured depth image. The re-scaled RGB images of two labels are described in figure 6.7.

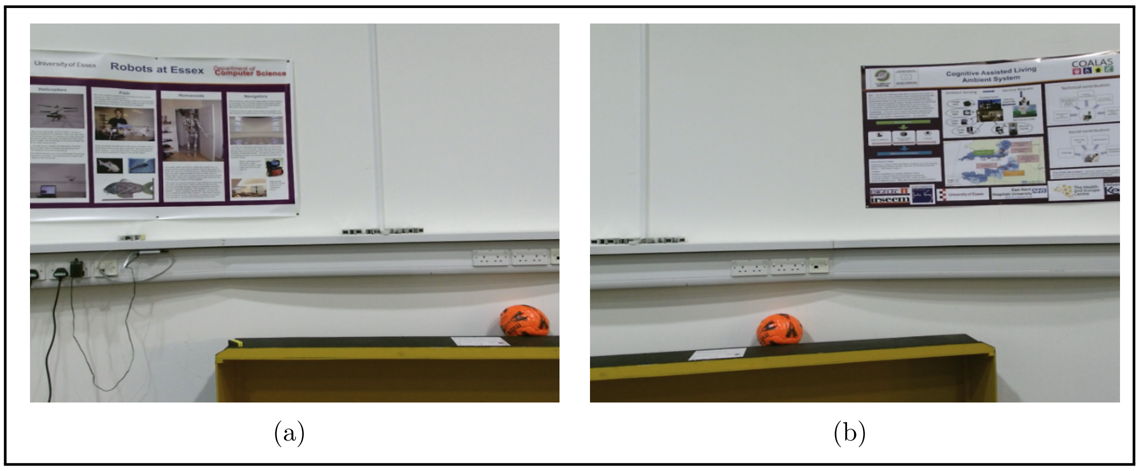


Figure 6.7: The re-scaled RGB image of two posts

1. **BD Extractor** (Bradski and Kaehler, 2008)

   BD Detector works similarly to LSD detector and is provided in OpenCV library. The extraction results from BD extractor is described in figure 6.8(A) and figure 6.9(A). We use the segmentation information from subsection 6.3 to define which line segments are describing the post area. For each line segment, if most of its points are laying beside or on the label area, this line segment will be kept. Otherwise, it is a false description. In figure 6.8(A) and figure 6.9(A), red and green lines are the confirmed and redundant line segments respectively. Due to the unclear boundary between post and wall in figure 6.8(A), BD extractor cannot detect the correct contour of the first post, but it can detect the boundary of the other post described in figure 6.9(A). BD extractor generates many false detection inside each post due to their rich texture. These false segmentation will be deleted in the fusing process.

2. **Hough Extractor** (Chutatape and Guo, 1999)

   The extraction results from Hough extractor is described in figure 6.8(B) and figure 6.9(B). As can be seen, although Hough extractor also failed to detect unclear boundary, it generates less false segmentation inside the post. For the parameters applied, we used a relatively large connection parameter to connect small line segments and set a larger length limit to filter out small line segments.

3. **LSD Extractor** (Von Gioi et al., 2010)

   The extraction results from LSD extractor is described in figure 6.8(C) and figure 6.9(C). LSD extractor generates richer line segments, but it has many false detection in both inside region and contour region.

4. **Contour Extractor** (Bradski and Kaehler, 2008)

   The extraction results from contour extractor is described in figure 6.8(D) and figure 6.9(D). In this experiment, we set the contour extractor to detect the most external contour of each area. However, it generates false detection in both inside and outside post area due to the rich texture and false CNN segmentation information. In order to optimise these errors, we fill the area inside each detected contour and check how many points inside this area belong to the label of post. If more than 85 percent of these points belong to label, we will save this contour as a candidate. For each candidate, we will choose the one with the largest scale and consider it as the optimal external contour.

After extracting 4 contours, we refer to the line feature pyramid mentioned in Chapter 3 to fuse the results obtained by 4 different contour extractors, which are de-

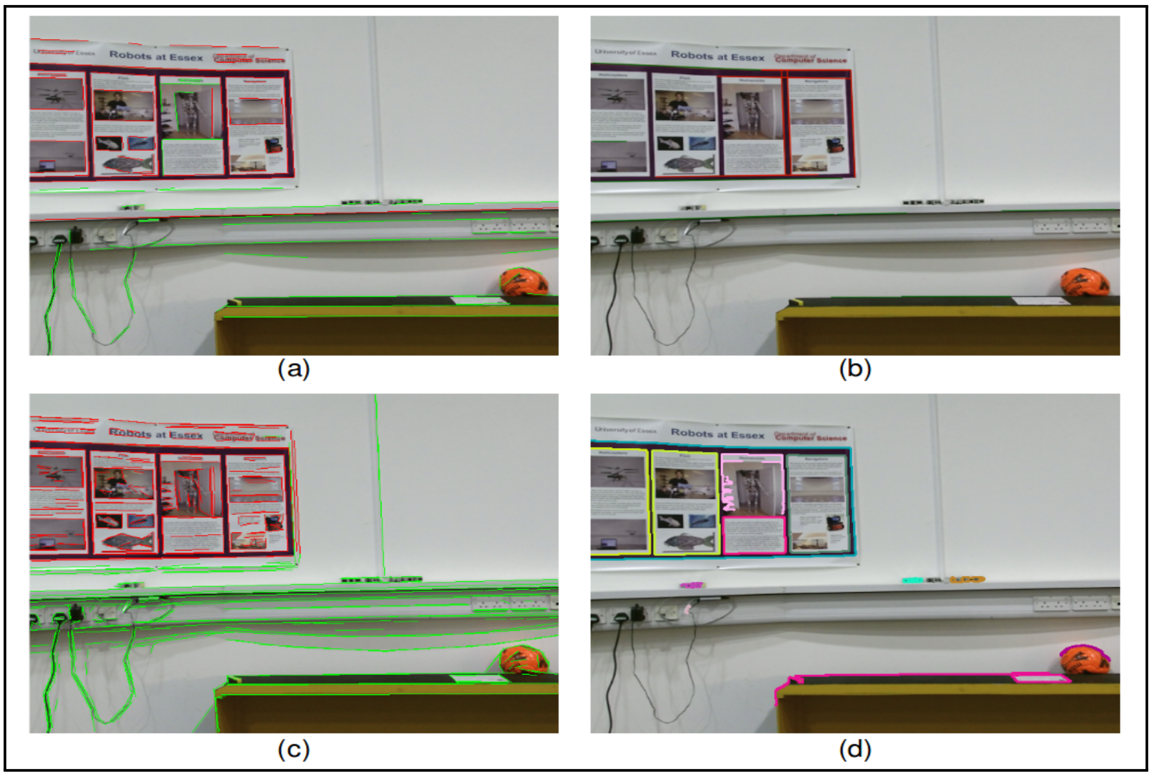


Figure 6.8: Contour extraction result on figure 6.7(a). (A) BD extractor, (B) Hough Extractor, (C) LSD extractor and (D) Contour extractor. The red line in each image is the confirmed line segments described the label area, and the green lines are the redundant line segments will be ignored in the further process.

scribed in figure 6.10. There are two major points found in this step:

1. When the edge and background of the object are not obvious, it is difficult for any contour extractor to find the exact edge of the object. This problem does not occur often in real scenes. When there are other edge textures inside the plane, these edge textures can be used as a substitute.

2. In our contour extraction algorithm we use the Hough extracted line segment and the contour extraction result to remove the extra erroneous edge segments and finally preserve the optimal contour with Contour Detector (Bradski and Kaehler, 2008). This method is still valid in this experimental scenario to delete both inside and outside false line segments. We need to mention that in this experiment results, each edge is not a complete line and divided into several segments due to the wall is curved and not flat.

After the line segmentation, We reconstruct the edge 3D model by combining depth information with edge information. Since the plane where the post is located is curved and the image is high definition, depth completion algorithm is not required to optimise the depth error. Instead, we use a 5 * 30 scale mask to smooth the

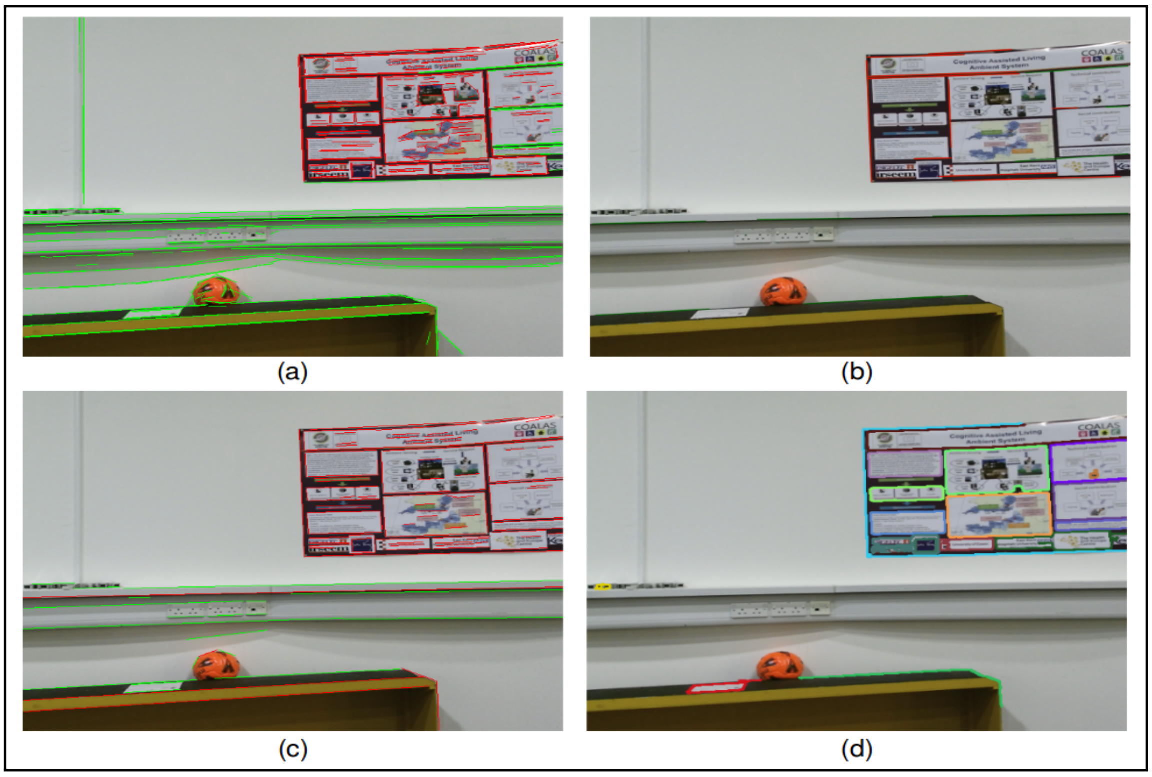


Figure 6.9: Contour extraction results on figure 6.7(b). (A) BD extractor, (B) Hough Extractor, (C) LSD extractor and (D) Contour extractor. The red line in each image is the confirmed line segments described the label area, and the green lines are the redundant line segments will be ignored in the further process

depth image in average, and the contour models are shown in figure 6.11.

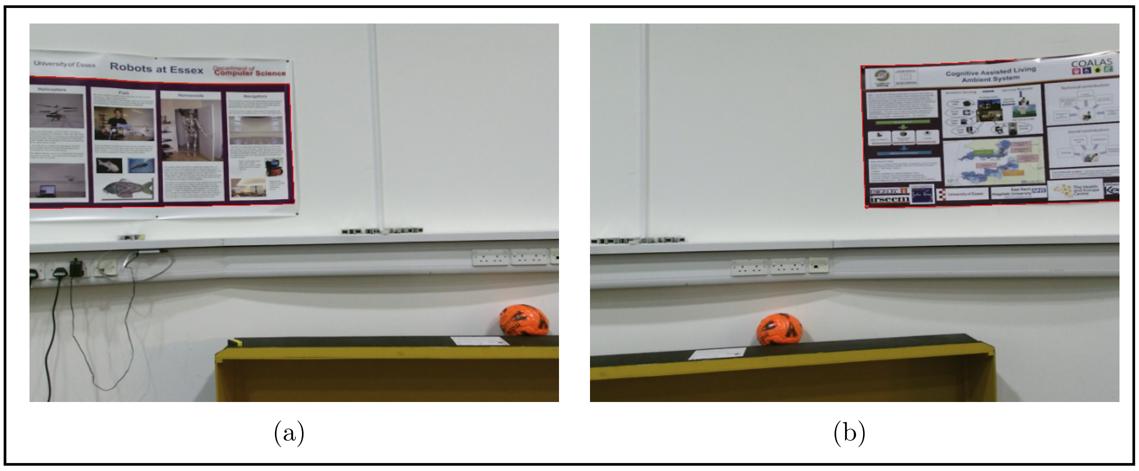


Figure 6.10: Final contour extraction results on two posts.

6.4 Plane-based Tracking and Mapping

In this experimental scenario, due to the lack of solid visual features and the absence of global loop closure optimisation, it is difficult to reconstruct the entire scene using

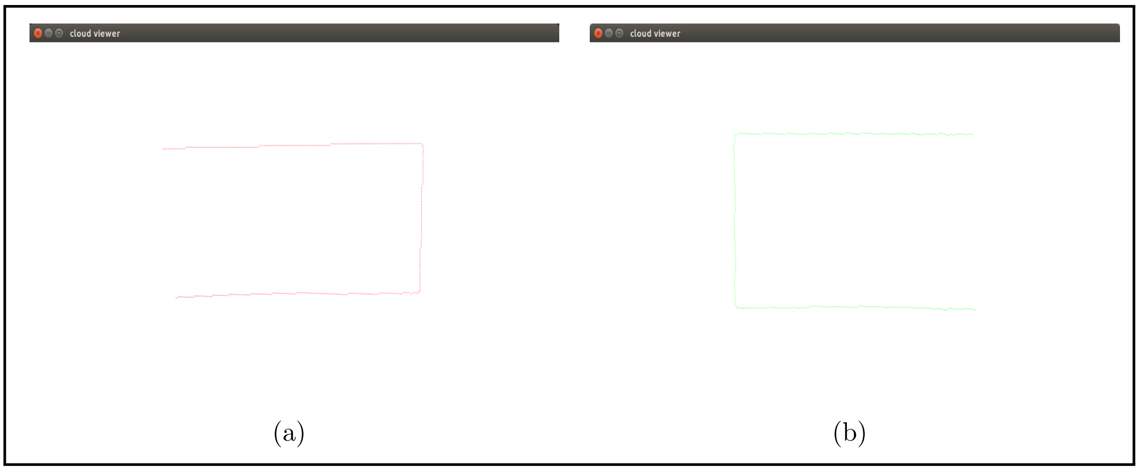


Figure 6.11: Contour model of two posts. (a) is the contour model of 6.7(a). (b) is the contour model of 6.7(b).

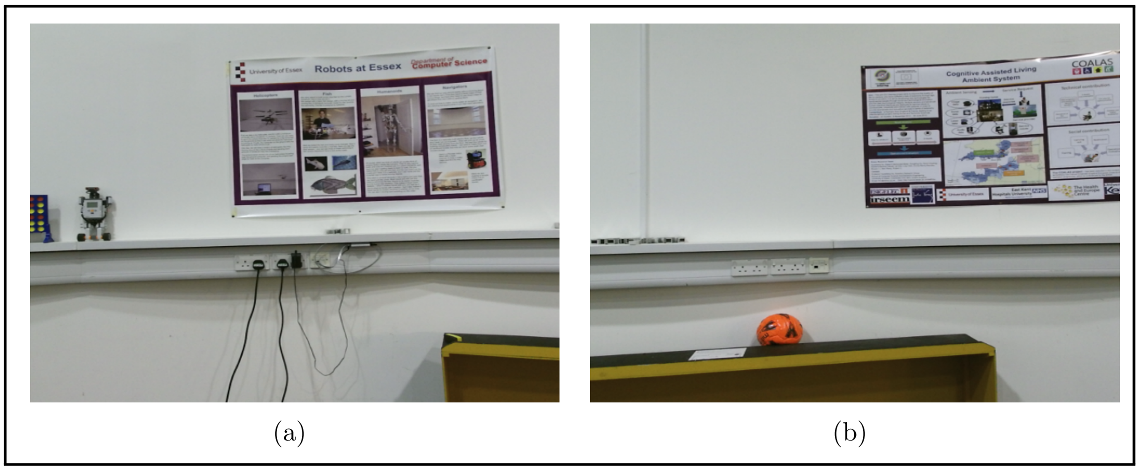


Figure 6.12: Captured images include different posts. (a) describes the frame includes first post. (b) describes the frame include second post.

the traditional feature-based visual mapping algorithm. Moreover, due to the large planes in the scene and the lack of a very obvious geometric feature, the dense visual mapping algorithm based on depth information is also difficult to imply. Thus, we use the plane-based visual mapping algorithm mentioned in the fourth chapter to complete the visual mapping task in this experimental low-textured scene.

There are two planes in this experimental scene described two different posts, and we distinguish these two posts by their textures. Since our purposed plane-based mapping algorithm is based on one planar area, the whole image sequence is divided into two image groups: the first image group $Group_1$ includes images that contain the first Post area only and the second image group $Group_2$ only describes the second Post. And these two conditions are present in figure 6.12(a) and figure 6.12(b) respectively. The mapping process includes three main process:

1. In the first process, we track the camera's movement with the images in

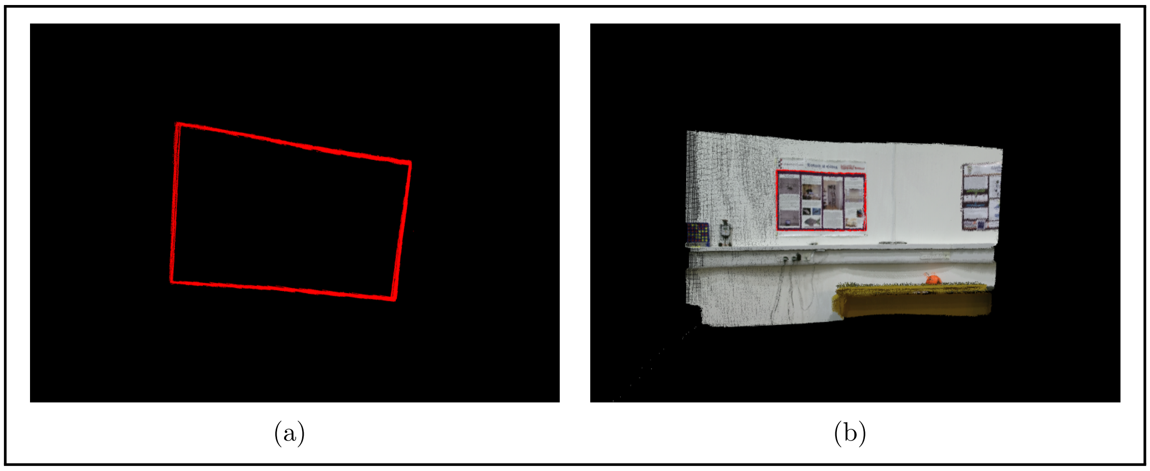


Figure 6.13: Scene Reconstruction Result on first post area. (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene.

$Group_1$, and this process terminates if the contour pixel appeared is less than 100 pixel length threshold value. Note that this threshold value is adjustable in different scenarios. We set a tracking accuracy threshold value to filter out the false tracked frames caused by the imprecise contour detection or tracking performance. And the reconstruction results is described in figure 6.13

2. In the following process, we start tracking the poses of frames in $Group_2$ and reconstruct the scene includes the second post. The way this step ends is similar to the first step. We also set an acceptable threshold value to filter out the imprecise tracking frames, and the poses of every frame are saved in $Pose_2$. The reconstruction result is described in figure 6.14

3. In the final process, we will try to combine two points cloud by their frames in shared view zone. We will look for the two frames with the highest matching score in the last few frames in $Group_1$ and the first few frames in $Group_2$,and then use their transformation matrix to calculate the new pose of each frame saved in $Pose_2$. Finally, all the frames in $Pose_2$ are transferred and added to global point cloud. The performance of this step are described in figure 6.15.

With the purposed mapping process, we can reconstruct a low-texture scene model with less visual features and no reliance on global loop closure optimisation. For the scenes with more planes, we can repeat the process 2 and process 3 to make our visual mapping algorithm work in a larger scenario. Finally, we can transfer the point-cloud presented scene cloud to voxel model in '.obj' format with Meshlab, and the result is described in figure 6.16.

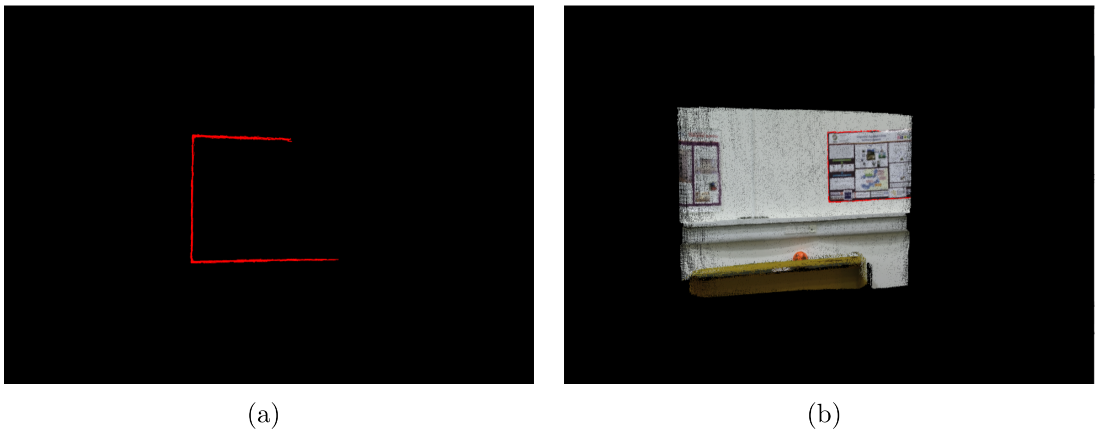


Figure 6.14: Scene Reconstruction Result on second post area (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene.

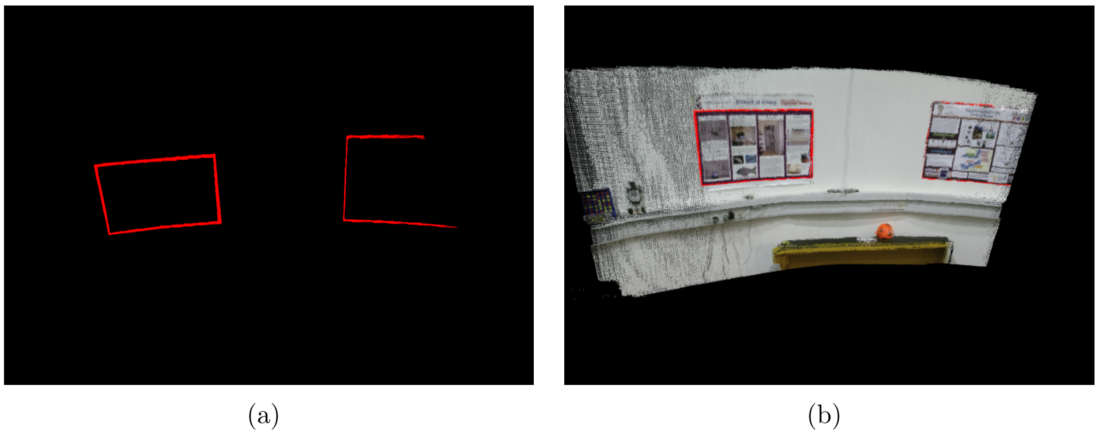
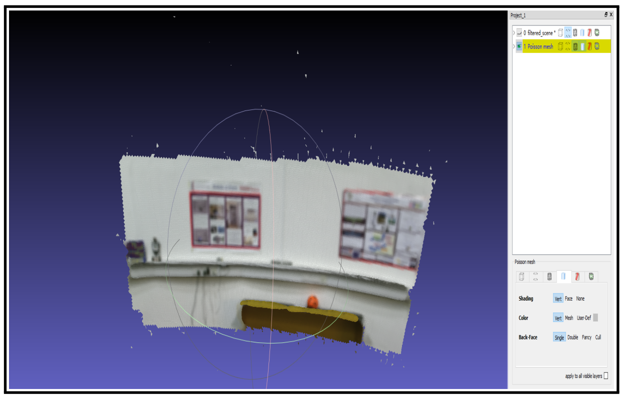


Figure 6.15: The whole scene mapping result. (a) describes the reconstructed contour, (b) describes the reconstructed contour and scene.



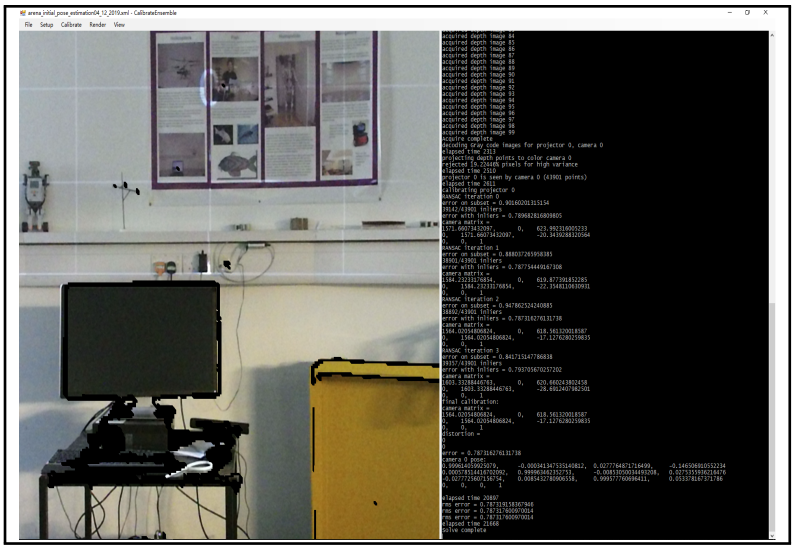
Shading [✓ Vert] Face None
Color [✓ Vert] Mesh User-Def
Back-Face [✓ Single] Double Fancy Cull
apply to all visible layers ☐

Figure 6.16: Final scene model in ".obj" format.

6.5 Sensor Calibration and initial pose estimation

6.5.1 Sensor Calibration with Structured-light projection

Before the SAR application enters the running mode, the system will start with sensor calibration process. In this step, the structured-light projection technique is used to define the internal and external parameters of both visual sensors, and build the calibration scene. Since structured-light projection algorithm requires the projection target to have some obvious geometric and physical structure, We find several candidates from image sequence and choose the optimum based on the number of solid visual features. Note that the calibration target can also be chosen by the presenter's intuitive feeling. After finding the best calibration area inside the scene, we put the robot in a corresponding position and facing this area perpendicularly. Then, the system will start the calibration process. We haven't designed a restrict distance limitation between the robot and the projection surface. As long as it can make the calibration work successfully, this distance is acceptable. The calibration process and result are described in figure 6.17.


arena_initial_pose_estimation04_12_2019.xml - CalibrateEnsemble
File
Setup
Calibrate
Render
View

```
acquired depth image 84
acquired depth image 85
acquired depth image 86
acquired depth image 87
acquired depth image 88
acquired depth image 89
acquired depth image 90
acquired depth image 91
acquired depth image 92
acquired depth image 93
acquired depth image 94
acquired depth image 95
acquired depth image 96
acquired depth image 97
acquired depth image 98
acquired depth image 99
Acquire complete
decoding Gray code images for projector 0, camera 0
elapsed time 2313
projecting depth points to color camera 0
rejected 19.22446% pixels for high variance
elapsed time 2510
projector 0 is seen by camera 0 (43901 points)
elapsed time 2611
calibrating projector 0
RANSAC iteration 0
error on subset = 0.90160201315154
39142/43901 inliers
error with inliers = 0.789682816809805
camera matrix =
1571.66073432097,       0,    623.992316005233
0,    1571.66073432097,       -20.3439288320564
0,    0,    1
RANSAC iteration 1
error on subset = 0.888037265958385
38901/43901 inliers
error with inliers = 0.787754449167308
camera matrix =
1584.23233176854,       0,    619.877391852285
0,    1584.23233176854,       -22.3548110630931
0,    0,    1
RANSAC iteration 2
error on subset = 0.947862524240885
38892/43901 inliers
error with inliers = 0.787316276131738
camera matrix =
1564.02054806824,       0,    618.561320018587
0,    1564.02054806824,       -17.1276280259835
0,    0,    1
RANSAC iteration 3
error on subset = 0.841715147786838
39357/43901 inliers
error with inliers = 0.793705670257202
camera matrix =
1603.33288446763,       0,    620.660243802458
0,    1603.33288446763,       -28.6912407982501
0,    0,    1
final calibration:
camera matrix =
1564.02054806824,       0,    618.561320018587
0,    1564.02054806824,       -17.1276280259835
0,    0,    1
distortion =
0
0
error = 0.787316276131738
camera 0 pose:
0.999614059925079,      -0.000341347535140812,  0.0277764871716499,     -0.146506910552234
0.000578514416702092,   0.999963462352753,      -0.00853050034493208,   0.0275355936216476
-0.027772560715675 4,   0.0085432780906558,     0.999577760696411,      0.053378167371786
0,    0,    0,    1

elapsed time 20897
rms error = 0.787319158367946
rms error = 0.787317600970014
rms error = 0.787317600970014
elapsed time 21668
Solve complete
```

Figure 6.17: Scene Calibration Results.

Structured light projection calibration derives the intrinsic and extrinsic parameters of both visual sensors, as well as the projection scene in camera's initial view. This projection scene is a part of the whole scene. So we will try to register this projection scene to the whole reconstructed scene model in the next step.

6.5.2 Initial pose estimation

This step aims at matching the reconstructed scene and the calibration derived projection scene, and then adding the whole reconstructed scene into Unity3D simulation workplace. This process includes two steps: registering calibration frame to scene model and adding the scene to Unity3d Workplace.

Assuming the pose during calibration refers to the initial pose, and the captured RGB and Depth data refer to initial RGB and initial depth. The first step starts with finding the closest neighbour in the database to the initial frame with visual features and 'DBoW3' library. To ensure the accuracy of the candidate, we repeat this process 10 times and use the most frequent candidate as the nearest neighbour. The initial frame and its neighbour are described in figure 6.18

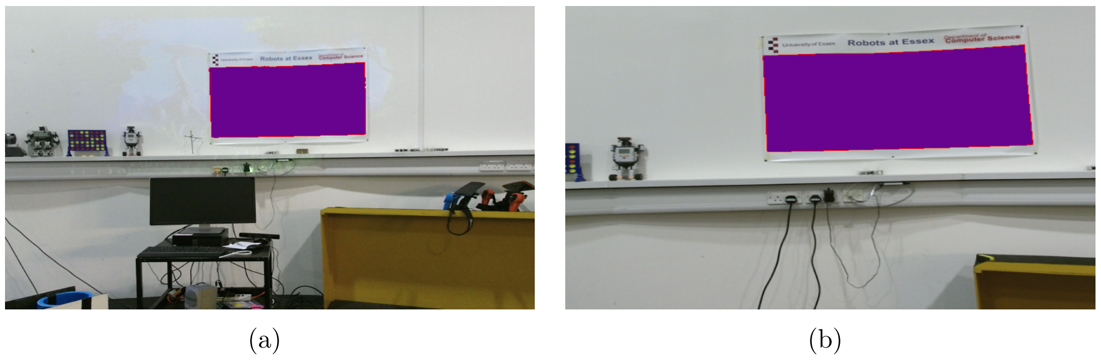


Figure 6.18: Initial frame and its nearest neighbour. (a) is the initial frame captured during calibration process. (b) is its nearest neighbour in database.

Due to the depth variance, it is impossible to imply our plane-based algorithm to find their transformation directly. Therefore, we established a constraint function based on the different normal vectors of the two planes. By only solving the translation parameters only instead of the rotation and translation matrix, this method decrease the distance between the two planes without rotation. Then the spatial relationship of two planes is solvable with the proposed plane-based ICP algorithm. This process can be divided into three steps.

1. We use RANSAC to calculate the normal vector of the two regions. Because the post area is arc-shaped in the experimental scene, RANSAC solution will divide a plane into several small planes due to various normal vectors. So we only consider the plane patch which has the largest number of pixels, and this patch always appear in the middle of plane.

2. In this step, we will build a constraint equation based on the plane's normal vector, and then use 'CERES' lib to solve only the translations in the x, y

and z directions. The reason for this is to prevent the rotation of two contour models. By solving the translation parameter, we can bring the two contour models close enough to perform the point based ICP algorithm.

3. The final step is to find the spatial transformation between two planes based on proposed plane-based registration algorithm. Due to depth, scale and viewing angle differences, we can adjust the scale and match two planes (the same plane in two scenes) in multiple times to make two scene model correctly matched. After transferring the pose of our reconstructed scene model in voxel format and inserting it into Unity3D, user can tune the pose of scene model easily to make virtual models match each other in Unity3D. The result is described in figure 6.20

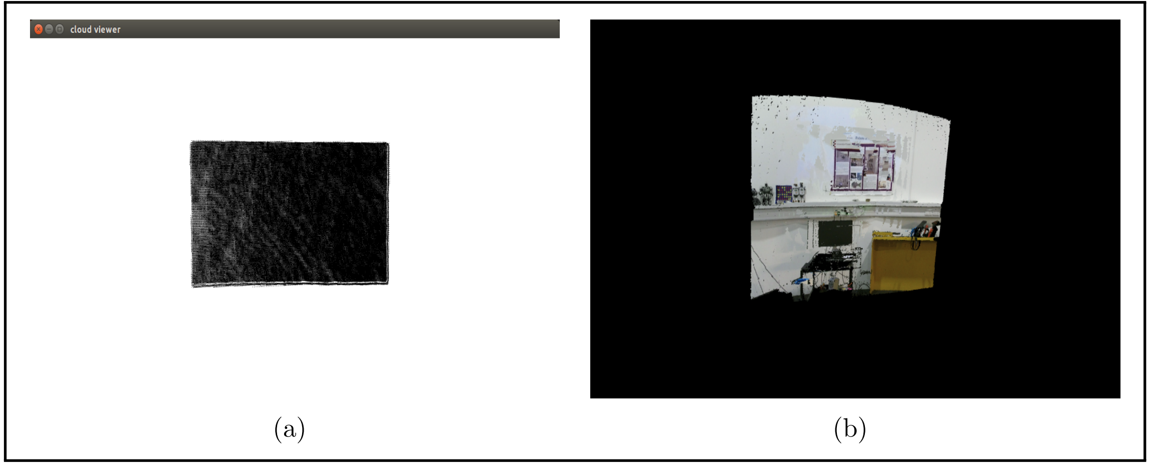


Figure 6.19: The initial frame registration result. (a) describes the matched plane in two models. (b) describes the matching result in the front view.

6.5.3 Scene Building in Unity3D

The matching result derived in the earlier step is not very accurate, but it is available to use as a raw pose of reconstructed scene model in Unity3D. Thus, inserting and adjusting the reconstructed scene model into Unity3D is the main purposed in this step. There are two criteria while user is adjusting the pose of scene model to match the calibration scene in Unity3D:

1. The reconstructed contour model should match the post area.

2. The reconstructed scene model should match the calibration scene.

The final scene and pose parameter of the reconstructed scene are described in figure 6.20 and figure 6.21 respectively.

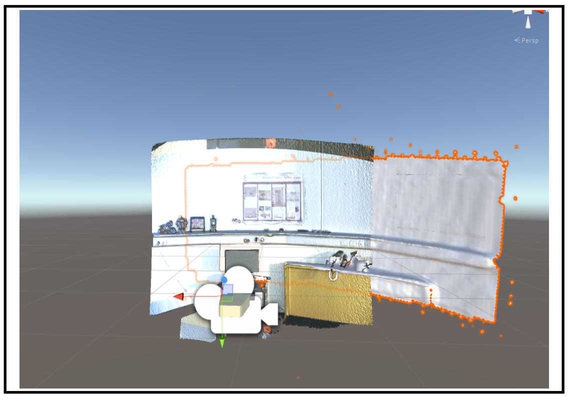
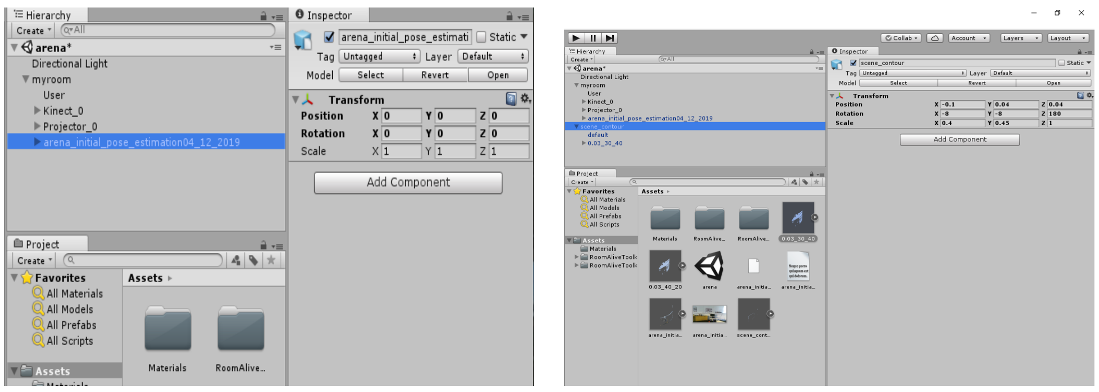


Figure 6.20: Matching scene model and calibrated Projection SAR system in Unity3D.



arena_initial_pose_estimation04_12_2019

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

(a)

scene_contour

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	-0.1	0.04	0.04
Rotation	-8	-8	180
Scale	0.4	0.45	1

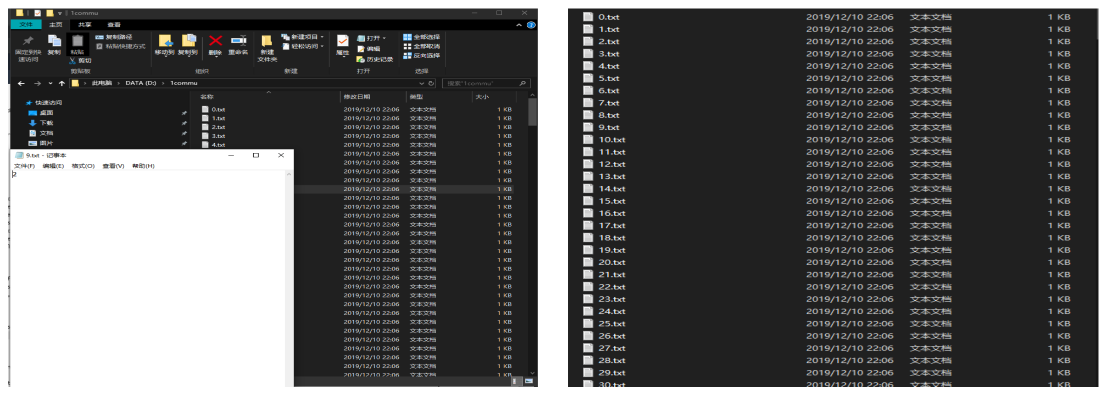
(b)

Figure 6.21: The pose of important elements. (a) describes the pose of calibrated projection scene. (b) is the pose of reconstructed scene model.

6.6 Trajectory simulation in Unity3D

This part is to test the connection between the Pioneer Robot and Unity3D. By connecting Kinect-V2 and the Pioneer robot, we have established a visual presenter following service. The robot always keeps the presenter inside its middle field of view. If the presenter is outside the zone and the distance between the torch and the image centre exceeds a certain threshold, the robot sets moving the direction and send this moving order to mobile platform to track the user automatically.

To connect the real projector and the simulated projector in Unity3D, a '.txt' file is used as a data transmission medium, and the folder where the '.txt' file is stored is considered as a subscribed channel. Unity3D subscribes to this channel. As the robot moves, the robotic platform will write its moving direction in a new '.txt' file. Whenever a new '.txt' file is generated, Unity will immediately read the information in it and move the virtual projector in the simulation work space. In this way, our system can successfully simulate the trajectory of the real sensor in the simulation work space, and project correct dynamic virtual content



```
0.txt     2019/12/10 22:06   文本文档   1 KB
1.txt     2019/12/10 22:06   文本文档   1 KB
2.txt     2019/12/10 22:06   文本文档   1 KB
3.txt     2019/12/10 22:06   文本文档   1 KB
4.txt     2019/12/10 22:06   文本文档   1 KB
5.txt     2019/12/10 22:06   文本文档   1 KB
6.txt     2019/12/10 22:06   文本文档   1 KB
7.txt     2019/12/10 22:06   文本文档   1 KB
8.txt     2019/12/10 22:06   文本文档   1 KB
9.txt     2019/12/10 22:06   文本文档   1 KB
10.txt    2019/12/10 22:06   文本文档   1 KB
11.txt    2019/12/10 22:06   文本文档   1 KB
12.txt    2019/12/10 22:06   文本文档   1 KB
13.txt    2019/12/10 22:06   文本文档   1 KB
14.txt    2019/12/10 22:06   文本文档   1 KB
15.txt    2019/12/10 22:06   文本文档   1 KB
16.txt    2019/12/10 22:06   文本文档   1 KB
17.txt    2019/12/10 22:06   文本文档   1 KB
18.txt    2019/12/10 22:06   文本文档   1 KB
19.txt    2019/12/10 22:06   文本文档   1 KB
20.txt    2019/12/10 22:06   文本文档   1 KB
21.txt    2019/12/10 22:06   文本文档   1 KB
22.txt    2019/12/10 22:06   文本文档   1 KB
23.txt    2019/12/10 22:06   文本文档   1 KB
24.txt    2019/12/10 22:06   文本文档   1 KB
25.txt    2019/12/10 22:06   文本文档   1 KB
26.txt    2019/12/10 22:06   文本文档   1 KB
27.txt    2019/12/10 22:06   文本文档   1 KB
28.txt    2019/12/10 22:06   文本文档   1 KB
29.txt    2019/12/10 22:06   文本文档   1 KB
30.txt    2019/12/10 22:06   文本文档   1 KB
```

(a) (b)

Figure 6.22: Data transmission files. (a) the number in this file indicates the moving direction. (b) describes the list of generated files.

6.7 Designed Interactive Virtual plays

This section describes the detailed design of interactive virtual plays in Unity simulation workplace. More specifically, the first section describes how to design virtual textures in Unity with reconstructed contour information and the necessary of the proposed approach. Then, the experimental results in real application are present in the following subsection.

6.7.1 Augmentation in Simulation Workplace

'.obj' is the main format of scene mesh used in Unity3D, and the most general '.obj' file does not include the colour information. To overcome this limitation, users can add colour information to the scene mesh through third-party software, such as "Blender" and 'MeshMixer'. Alternatively, 'RoomAlive' writes a program to render the model. However, the rendering tools attach only a single image to one-viewed mesh, and is unable to fuse and attach multiple images to a larger multi-viewed scene mesh. Therefore, this subsection introduces a practical approach to attach the virtual content on scene mesh in Unity3D simulation workplace based on the reconstructed geometric and contour data. It includes three main steps:

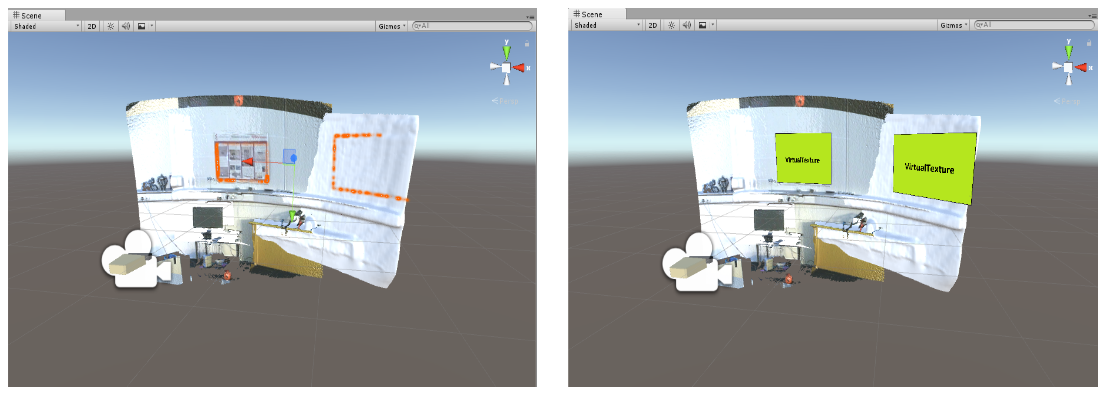
1. For the objects that are not attached on the background and have a clear depth difference on its edge: As they have obvious geometric features in reconstructed scene mesh, users could recognise the object directly and design the overlaying virtual content.

2. For the objects without obvious geometric features: Users could identify the

object in scene mesh through the reconstructed contour model, such as the post on the wall in experimental scene. By combining the structure information of the scene and the contour information of the objects, it is easy to design virtual textures on different regions of scene.

3. The final step aims at registering the virtual content to the scene mesh correctly through matching the contour and geometric feature. After inserting the virtual content into the simulation scene, user needs to adjust its pose to make it attach to the scene mesh and then adjust the view angle to the perspective of projector. Under projector's view, the pose and scale parameters of virtual content should be adjusted to make its edges coincide with reconstructed contour model. In this experiment, since designing the virtual model is not included in our research, we only use basic 3D models for demonstration.

The final representation of augmented scene mesh in Unity3D simulation work-space are described in figure 6.23.



(a) (b)

Figure 6.23: Attaching the virtual content on scene model. (a) describes the scene with contour. (b) describes attaching the virtual content in contour areas.

6.7.2 Experimental results

The mobility and feasibility of proposed mobile projection SAR robot are tested through experiments. The first experiment is started with inserting a virtual texture to cover the post area in simulated scene mesh and overlapping the post in the real scene. Figure 6.24 shows the experimental results.

Assuming current pose of the robotic pedestal refers to $Pose_1$. The system enters the visual tracking mode and follows the presenter. While the robot is moving, the proposed system always project the virtual textures on the post, and does not change the projection position due to the movement of the projector. The simulation work-space and augmented view in $Pose_2$ is demonstrated in figure 6.25.

By comparing figure 6.24(a) and figure 6.25(a), we can find that the projector has a significant displacement in simulation work space. Figure 6.24(b) and figure 6.25(b) show that the projected content always covers the post area and the projection position is not changing while the projector is moving.

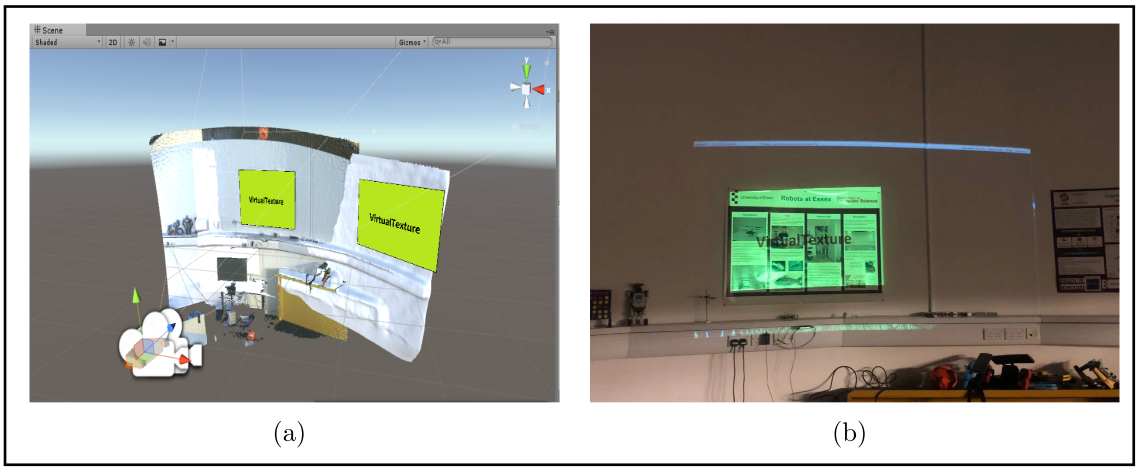


Figure 6.24: The experimental results of Augmentation view in $Pose_1$. (a) describes the simulation work-space in Unity3D, (b) describes the augmented view in experimental scene.

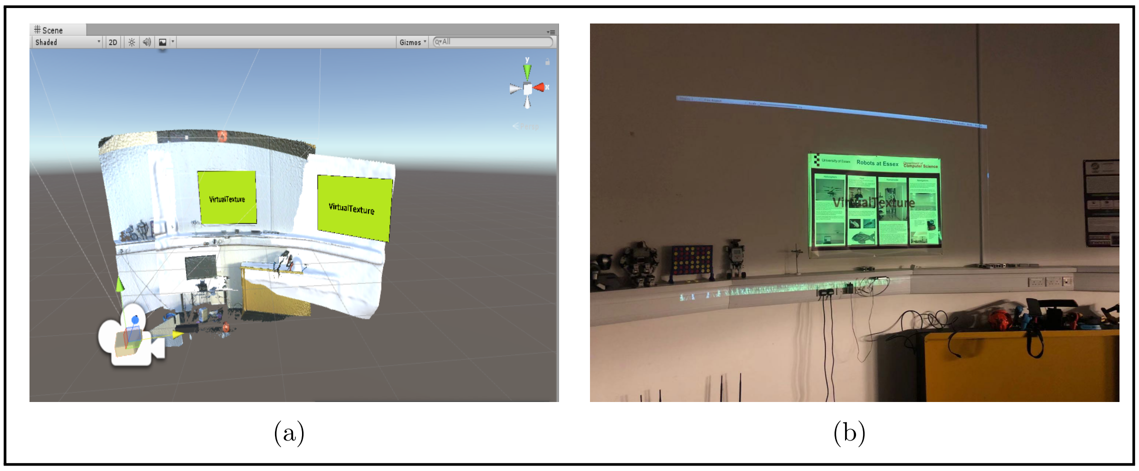


Figure 6.25: The experimental results of Augmentation view in $Pose_2$. (a) describes the simulation work-space in Unity3D, (b) describes the augmented view in experimental scene.

Another experiment is testing the performance of changing the appearance of background and controlling the virtual model with body gesture. Firstly, the virtual backgrounds are designed and attached in the simulation work-space, which is described in figure 6.26(a). Since this experimental scene is curved, two curved planes are designed and attached on the model in two post areas. However, Unity3D does not support attaching dynamic virtual content on curved object. Therefore, in figure 6.27(a), though I have rendered two curved models, both models still have only one colour (blue). Since designing the virtual model is not included in our research, this problem is not optimised in this experiment.

Red circles in figure 6.26(b) has marked the error from the model derived in structured-light calibration process, which is caused by the lack of obvious physical and texture features at the boundary of projection image. In fact, this error will not affect the projection SAR system and it can be optimised in a flat environment. Figure 6.27 has described the virtual interaction performance. In this experiment, two curved planes are designed to be fixed in current position, and two virtual contents (green texture and 3D model) can be controlled with body gesture. User can use left arm to make virtual model move in 4 directions, and use right arm to make it rotate. The experiment results are described in figure 6.27(a) and figure 6.27(b) respectively.

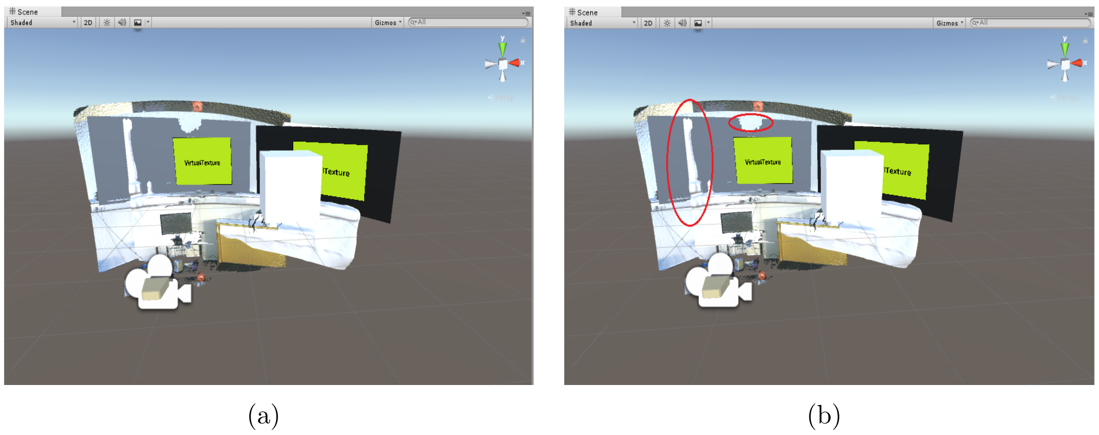


Figure 6.26: The scene model in experiment2. (a) describes the simulation work-space in Unity3D, (b) describe the error from inaccurate scene model derived in calibration process.

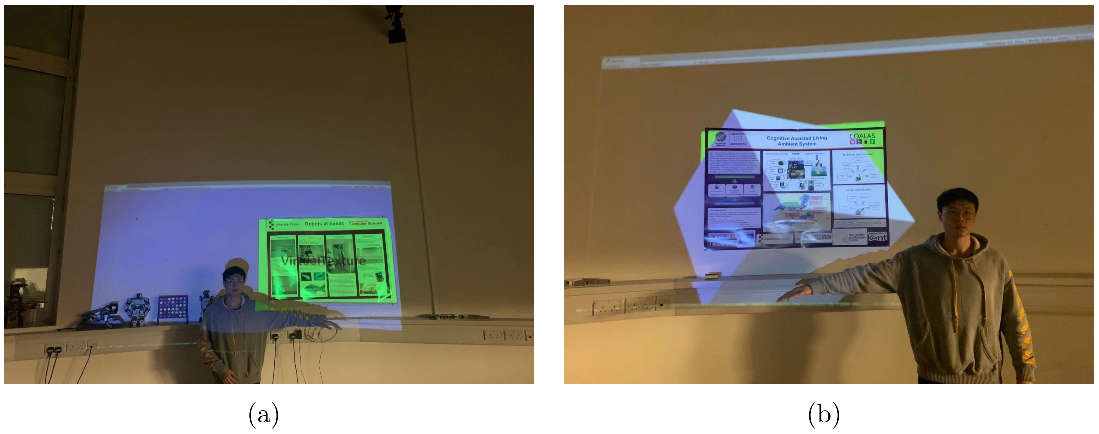


Figure 6.27: The experimental results of experiment2. (a) describes the user can move virtual texture by body gesture. (b) describes the user can rotate and move the 3D model by body gesture. In both cases, the virtual background (blue and black background) are not moving.

6.8 Comparison between Mobile and fixed SAR

This section compares the augmented view between the proposed Mobile projection SAR system and the state-of-art fixed Projection SAR system.

6.8.1 Working Area Extension

Comparing with current fixed Projection SAR system, our mobile application has a larger working area, and permits the presenter to have rich body movements around the scene instead of staying inside a limited working area. In addition, our mobile SAR systems can reduce dependence on high-quality sensors and cut the cost of devices. For example, some state-of-art projection fixed SAR systems require more than one camera-projector to cover a larger working area, and others need to use the fish-eye camera or wide-angle camera to make the camera capture the scene in one view at a high cost.

6.8.2 More Immersive Interactive Plays

The existing fixed SAR systems only support presenters to communicate with virtual images in a fixed area. When the augmented background is more detailed, or its scale is much larger than the projection zone, it cannot provide same interactive experience. The proposed mobile SAR system is a good solution to solve these problems. Assuming the augmented virtual background is very large, user could design and attach the virtual content in Unity3D in advance. When the system is in operation, it could follow the presenter walking around and projecting corresponding dynamic virtual contents on different regions in the scene so that the audience can have a more realistic augmented reality experience.

6.8.3 Bottlenecks of purposed SAR system

Comparing with the existing fixed Projection SAR systems, our proposed system needs the detailed description of scene mesh. The state-of-art fixed projection SAR systems use the structured light projection to obtain a detailed 3D projection scene, which is and only applicable for single-view reconstruction tasks. Our system has introduced a novel plane-based visual mapping algorithm as substitution. To generate descriptions in colour-less scene model in Unity3D, we proposed to use the contour model to mark important objects. This method helps the user to distinguish objects, but still less intuitive than colour information.

The main bottleneck is purposed robotic system has not solved the problems of changing tangential distortion completely, instead, it minimises this error by designing a fixed moving path and making robot facing projection target perpendicularly. This approach is useful when robot is working with a flat background. In general, although the proposed robot has not completely realize the free movement of projection-SAR robot, its research in scene reconstruction, AR design, and interface between Unity3D and external robot systems are very creative and valuable for current and future projection-SAR system.

6.9 Summary

In this chapter, our proposed mobile SAR system was tested in a simulated indoor demonstration scene to test its feasibility and performance. Firstly, our contour extraction pipeline described in Chapter 3 was tested in experimental scene. By comparing results between our algorithm and state-of-art four popular contour extractors, our algorithm is feasible and practical for extracting the external contour of most objects in indoor scenario, especially when the target object is rich in pattern and texture. Then, we demonstrate that our plane-based visual mapping algorithm introduced in Chapter 4 is practical in the scenarios that lack visual and geometric features. Comparing with recent sparse or dense visual mapping algorithms, our algorithm is more suitable for low-texture indoor demonstration and broadcasting scenarios. Moreover, our algorithm does not rely on a global close loop of camera trajectory.

Finally, our designed Projection-SAR cameraman was tested in a simulated scene with two phases: preparation phase and running phase. In the preparation phase, it found the part of the scene that best fits the vision sensor calibration through the visual features, and then calibrated the sensor with structured light projection and saves the RGB image and depth image at the current view. After that, it calculated the spatial relationship between the initial frame and the nearest frame through our designed plane-based inter-frame registration algorithm to obtain the initial pose of the robot relative to the entire scene. Finally, all the models are able to be inserted in Unity3D and tuned by user.

After the preparation phase, the system entered the running phase and started the mobile projection SAR service. In this stage, our designed Projection-SAR cameraman automatically tracked the presenter with RGB and depth information and projected corresponding dynamic virtual contents on the background. Moreover, it can recognise upper body movements of presenter and make interactive reactions. The experimental results were showing that our designed Projection-SAR cameraman can work in a much larger area than the state-of-art SAR systems, and is able to provide a more immersive virtual feeling with single camera-projector unit.

7

Conclusion and Future Work

7.1 Research so far

In recent years, as people's demands for sensory experience have increased, Augmented Reality has been introduced in recent indoor broadcasting and demonstration as an emerging technology. Among all AR branches applied in indoor broadcasting and demonstration, projection SAR (spatial augmented reality) applied in this filed has attracted lots of interest in the early stage since it can provide an immersive interactive experience for a group of viewers, and each viewer can enjoy the AR experience directly with naked eyes instead of wearing external devices. However, since the current projection-SAR system is fixed and its working range is rigidly limited, the entire system needs to be built in advance, state-of-art projection-SAR is impractical for real indoor broadcasting and demonstration applications.

In this thesis, an intelligent projection SAR robotic cameraman has been designed and tested, which is mainly used in the field of indoor broadcasting and demonstration. This system changes the background behind the presenter to provide the viewer with an interactive immersive viewing experience through projection virtual content, and change the virtual images according to the motion of the presenter's upper limb and movements inside the scene. This system mainly includes two technologies: projection SAR and user visual tracking. It has three research objects:

- Firstly, the system needs an accurate and robust edge extraction algorithm to mark the contour of important objects inside the scene since the user needs the important edge information to recognise object in colour-less scene model while designing the attached virtual content.

- Then, our designed robot needs an suitable visual mapping algorithm as the environments for indoor broadcasting and indoor demonstration scenarios contain very few objects with rich textures. Moreover, while reconstruction 3D detailed scene, we must consider there is no global loop closure optimisation occurs.

- Finally, when the projection SAR system is built upon ground mobile platform, we need to consider how the camera can track the human target and interact with the virtual scene effectively.

Based on these research objectives, the thesis research has been conducted in chapters 3, 4, 5 and 6 to show the results of our proposed intelligent projection SAR robotic cameraman respectively.

In chapter 3, a contour extraction pipeline was designed to detect the contour robustly when the RGB, depth information and CNN scene segmentation data are in low resolution or motion blurred. It is a combination of 4 recent popular contour or line extractors. By analysing the advantages and disadvantages of each extractor, we have established a line extractor pyramid based on the quality and quantities of their detected line segments. In order to let the user know the semantic meaning of each contour, we use CNN segmentation data to filter the contour information. In the line extractor pyramid, Hough, BD and LSD extractor were from the top to bottom to represent the number of detected line segments from small to large, and the average accuracy from high to low. In Hough extractor level, because the Hough extractor can join small segments and fuse multiple adjacent parallel segments, a higher threshold parameter was set to make it detect only primary line features inside each image.

Since the BD extractor produces fewer line segments than the LSD extractor and retains the obvious segments, BD extractor and LSD extractor were used as feature extractors for the second and third layers respectively. While detecting the features, our algorithm starts with extracting the contour features from bottom layer to the top layer refers to the pyramid, and then repeatedly clustering detected line segments according to their directions and positions. As contour cannot be accurately described because of blurred image or the edge is too short in current camera's perspective, whenever the image is processing from lower layer to the upper layer, the features detected by upper layer were compared with the clustered features from lower layer and only the similar ones are kept; The rest of features are re-clustered and filtered by their position and length.

Finally, all clustered line segment groups are filtered by the contour extraction result, and only the ones near the contour are kept. In each successfully matched line group, the best result was selected by the average direction of line group and the length of each line segments. Meanwhile, considering the occlusion problem existed in experimental scenes, a line segment completion algorithm was proposed based on the end points of each line segment. By analysing whether the endpoint belongs to the intersection point of contour or boundary of image, our algorithm is able to predict the occluded part of each line and complete each line by connecting segments or extending the segment to the image boundary.

In Chapter 4, a creative plane-based visual mapping algorithm was introduced for indoor low-texture broadcasting and demonstration scene. Our approach relies on only one medium or large size planar area that is not required to be fully appeared in camera's perspective. These planes include projection screens, blackboards, large screens, wall posters and desktops, etc. Another innovative points is we don't need the prior knowledge of this plane, or the structure information of the scene, like Manhattan Indoor Space structure. Similar to state-of-art visual mapping algorithm, our plane-based algorithm has visual tracking and scene mapping process work in parallel. In tracking process, we estimate the pose of camera through tracking the contour of landmark plane. Different to recent line feature based visual tracking algorithm, we project the 2D contour information of the plane into the 3D space and form the 3D contour model. Then the point-point ICP algorithm is used to match the 3D contour models of the two frames and obtain the trajectory of the camera.

By using this inter-frame registration algorithm, we solved the problem of tracking failure caused by too few visual features. The experiments showed that our algorithm fails when only one edge is visible in camera's view. Meanwhile, as the global loop closure optimisation is difficult to implement in practical applications, we track the complete contour of a plane as a replacement. When the contour of landmark plane has been detected completely, our algorithm stops adding new frames and progressively update all previous key-frames from current frame by aligning detected edges. The experimental results proved that our inter-frame registration algorithm can be applied to medium distance key-frames registration, and the whole mapping algorithm can be used as an optimisation program for pose-graph based SLAM algorithm when the scene is lack of visual features, like ORB-SLAM.

In Chapter 5, the design of our mobile projection-SAR cameraman was described. Our robotic cameraman includes two main techniques: a projection-SAR and a ground mobile visual tracking robot. Our system has two main working phases.

In preparation mode, it will read the pre-build scene from data, then calibrate the sensors and estimate its initial pose relative to the projection area. To calibration the sensor, we use state-of-art structured-light projection algorithm to define the parameters of both visual sensors and get the RGB and depth information in its current view. After that, we firstly find the frame closest to the current frame from database through DBoW3, then extend each edge along the direction according to the normal vector of the plane on the 3D contour model of both two frames. Finally, the initial pose of camera relative to the scene was obtained by building and solving the constraints function of matching edges and planes.

To enable the presenter tracking service in the running mode, our system uses KinectV2 camera to locate the presenter with depth information and then track the torch with RGB data. If the presenter is moving away from the centre of camera's view, the system can control the robotic pedestal to keep the presenter inside field of view. Considering the distortion problem of projection images and the flexibility of mobile pedestal, our robot is designed to move left or right only without forward and backward motions. For immersive AR plays, we use the Kinect to track the movement of presenter's upper body, and then project the virtual content accordingly.

In chapter 6, we tested the proposed robot in a simulated indoor demonstration scene. Firstly, our proposed contour extraction algorithm and plane-based scene reconstruction algorithm were tested in the experimental scene. The experimental results show that our algorithm has two practical advantages compared to the existing algorithms. On the one hand, our algorithm can accurately predict the camera's trajectory and reconstruct the scene if there are not enough visual and geometric features, and no global loop closure optimisation while mapping the scene. On the other hand, we have a larger working range than the existing fixed projection-SAR systems, but the lower accuracy of the construction. After that, we tested our AR application in the experimental scenario. Compared to the existing projection-SAR systems, the proposed system can produce a more vivid virtual-real interaction experience in a large environment.

7.2 Research Contribution

This thesis research has made the following contributions:

1. Contribution on Contour Extraction.

- Our algorithm solves the problem that the existing edge extraction algorithms cannot accurately describe edges when images are blurred due to motion blur or shooting jitter. By analysing four different contour extraction algorithms and combining the extraction results according to their characteristics, we can accurately and robustly find the optimal edge in the blurred image.

- Our algorithm also solves the problem that the contour is partially occluded. By analysing whether the endpoint of each edge belongs to the inflection point of the contour or to the edge point of the image, our algorithm can automatically determine whether the edge segment is occluded. Then complete the occluded edges by extending or connecting.

2. Contribution on indoor mapping in extreme environments.

   - The proposed plane-based visual mapping algorithm optimises the mapping performance in low-texture and low geometric feature environment. By tracking a medium or large size planar area inside the scene, our algorithm is able to track the trajectory of camera and then reconstruct the whole scene.

   - The proposed plane-based visual mapping algorithm does not require the landmark plane to be fully appeared in camera's perspective during tracking. The experimental results have shown that our algorithm works successfully when two or more edges are visible.

   - In order to meet the needs of practicability of indoor mapping applications, we replaced loop closure optimisation by tracking the complete contour. This allows our algorithm is applicable for the precise reconstruction of small-scale scenes. Meanwhile our algorithm does not require a prior knowledge of the landmark plane, and does not rely on the indoor Manhattan structure.

3. Contribution on mobility of Projection-SAR applications.

   - Our system combines projection-SAR and ground mobile human following service robot to make it work in a larger working range. By tracking the presenter, the system can project different dynamic virtual images in different background zones, which provides the viewer a richer and more realistic AR experience.

- Our system can provide creative human-computer interaction behaviours and enhance the immersive AR experience for users, on-site audience and audience behind the screens. And this can not be achieved by existing AR systems.
- Compared to the existing SAR system, our system is applicable to different scenarios. Its mobile base can be replaced by different kinds of wheel configuration to meet the needs of various working scenes.
- Our system uses normal sensors and does not required up-front frameworks, which makes projection-SAR has a greater market value.

7.3 A List of Publications

The academic publications that have been achieved during this PhD study are listed below.

- Yan, D. and Hu, H., Application of augmented reality and robotic technology in broadcasting: a survey. Robotics, 6(3), 2017, pages 1-18.
- Yan, D. and Hu, H., A novel plane-based image registration pipeline with CNN scene parsing. Proceedings of the 11th Computer Science and Electronic Engineering Conference, University of Essex, UK, 25-26 Sept. 2019, pages 1-6.
- Yan, D. and Hu, H., Robotic cameraman for AR broadcasting and demonstration, in preparation to be submitted to IEEE Transactions on Systems, Man and Cybernetics: Systems.
- Yan, D. and Hu, H., plane-based visual mapping algorithm in empty indoor scenes. in preparation to be submitted to IEEE Transactions on Industrial Electronics.

7.4 Future Work

Future work will focus more on improving the performance of the mobile projection SAR system, which mainly includes:

1. To work on changing the parameters of projection images.

In recent Projection-SAR applications, the parameters of projection images are defined by structured-light projection algorithm during sensor calibration process, which limits the Projection SAR to a fixed application. When the SAR system moves, the parameter of projector changes. In our system, we have optimised this error but not solved it completely. Therefore, the further research will focus on creating an algorithm for calibrating projection images that is specified for mobile SAR systems. In this algorithm, the project parameters are firstly calibrated by structured-light projection, and then continuously changed according to the relative pose between the projector and the projection target during operation. In addition, this algorithm can also solve the problem of real-time tracking of camera pose in feature-less environment by adding distance measurement sensors or other approaches.

2. To work on improving the flexibility of robotic pedestal.

   Our purposed robotic cameraman is working with a wheeled mobile pedestal with the limited movements. It is only allowed to move left or right with a fixed direction in order to keep the user tracking accuracy for projecting contents in the most indoor SAR applications at present, such as the orbital cameraman in filming and broadcasting. It is clear that future SAR applications require more flexible movement. Thus, it is necessary to adopt the mobile robot with omnidirectional movements for the projection SAR applications, which allows the presenter to move more freely inside the scene and allows the audience to have a more realistic viewing experience.

3. To work on richer interactive plays specific for the projection SAR.

   As the wide applications of AR technology, designing richer interpersonal interaction experiences has been the main research of AR applications in recent years. However, there is no breakthrough being made for improving projection-based AR experience. Therefore, I will try to design some human-computer interaction behaviours suitable for spatial SAR, which can change the appearance of the scene more realistically and recognise more user's actions.

Bibliography

T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

O. Bimber and R. Raskar. *Spatial augmented reality: merging real and virtual worlds.* AK Peters/CRC Press, 2005.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

O. Cakmakci and J. Rolland. Head-worn displays: a review. *Journal of display technology*, 2(3):199–216, 2006.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012.

A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

R. O. Castle, G. Klein, and D. W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding*, 115(6):854–867, 2011.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

O. Chutatape and L. Guo. A modified hough transform for line detection and its performance. *Pattern Recognition*, 32(2):181–192, 1999.

P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing, 2007.

J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 941–947. IEEE, 1999.

R. H. Creighton. Unity 3d game development by example. beginner's guide: a seatof-your-pants manual for building fun, groovy little games quickly. *Packt Pub*, 2010.

A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

J. Deglint, A. Cameron, C. Scharfenberger, H. Sekkati, M. Lamm, A. Wong, and D. A. Clausi. Auto-calibration of a projector–camera stereo system for projection mapping. *Journal of the Society for Information Display*, 24(8):510–520, 2016.

K. G. Derpanis. The harris corner detector. *York University*, 2004.

D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deep slam. *arXiv preprint arXiv:1707.07410*, 2017.

J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

M. Fiala. Automatic projector calibration using self-identifying patterns. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 113–113. IEEE, 2005.

X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

M. Greenspan and M. Yurick. Approximate kd tree search for efficient icp. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 442–448. IEEE, 2003.

A. Grundhöfer, M. Seeger, F. Hantsch, and O. Bimber. Dynamic adaptation of projected imperceptible codes. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531. IEEE, 2014.

C. G. Harris, M. Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

T. Hayashi, H. Uchiyama, J. Pilet, and H. Saito. An augmented reality setup with an omnidirectional camera based on multiple object detection. In *2010 20th International Conference on Pattern Recognition*, pages 3171–3174. IEEE, 2010.

B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira. Roomalive: magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 637–644. ACM, 2014.

A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013.

A. Kermen, T. Aydin, A. O. Ercan, and T. Erdem. A multi-sensor integrated head-mounted display setup for augmented reality applications. In *2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, 2015.

L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

A. Khaloo and D. Lattanzi. Robust normal estimation and region growing segmentation of infrastructure 3d point cloud models. *Advanced Engineering Informatics*, 34:1–16, 2017.

G. Kipper and J. Rampolla. *Augmented Reality: an emerging technologies guide to AR.* Elsevier, 2012.

G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. 2009.

S. Klose. Auto-calibration of multi-projector systems, May 8 2007. US Patent 7,215,362.

B. Kress and T. Starner. A review of head-mounted displays (hmd) technologies and applications for consumer electronics. In *Proc. SPIE*, volume 8720, page 87200A, 2013.

R. Kurban, F. Skuka, and H. Bozpolat. Plane segmentation of kinect point clouds using ransac. In *The 7th international conference on information technology*, pages 545–551, 2015.

Y. LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20, 2015.

M. Li and A. I. Mourikis. Improving the accuracy of ekf-based visual-inertial odometry. In *2012 IEEE International Conference on Robotics and Automation*, pages 828–835. IEEE, 2012.

R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu. Night-time indoor relocalization using depth image with convolutional neural networks. In *2016 22nd International Conference on Automation and Computing (ICAC)*, pages 261–266. IEEE, 2016.

R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Transactions on Automation Science and Engineering*, 15(2):651–662, 2018a.

R. Li, S. Wang, Z. Long, and D. Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE, 2018b.

T. Li, F. Hu, and Z. Geng. Geometric calibration of a camera-projector 3d imaging system. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, pages 187–194. ACM, 2011a.

T. Li, T. Mei, I.-S. Kweon, and X.-S. Hua. Contextual bag-of-words for visual categorization. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):381–392, 2011b.

X. Li and R. Belaroussi. Semi-dense 3d semantic mapping from monocular slam. *arXiv preprint arXiv:1611.04144*, 2016.

L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen. Median robust extended local binary pattern for texture classification. *IEEE Transactions on Image Processing*, 25(3):1368–1381, 2016.

H. Luo, J. Xu, N. H. Binh, S. Liu, C. Zhang, and K. Chen. A simple calibration procedure for structured light system. *Optics and Lasers in Engineering*, 57:6–12, 2014.

E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

I. Martynov, J.-K. Kamarainen, and L. Lensu. Projector calibration by "inverse camera calibration". In *Scandinavian Conference on Image Analysis*, pages 536–544. Springer, 2011.

J. McCormac, A. Handa, A. Davison, and S. Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017.

G. Moon, J. Yong Chang, and K. Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018.

M. Muja and D. G. Lowe. Fast matching of binary features. In *2012 Ninth conference on computer and robot vision*, pages 404–410. IEEE, 2012.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011a.

R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011b.

P. C. Ng and S. Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

J. V. Pavlik. *Digital Technology and the Future of Broadcasting: Global Perspectives.* Routledge, 2015.

A. Perreault. What you need to know about 3d scanning, 2016. URL http://3dscanningservices.net/blog/need-know-3d-scanning/.

T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. J. Berlles. S-ptam: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93:27–42, 2017.

A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508. IEEE, 2017.

X. Qi, R. Xiao, C.-G. Li, Y. Qiao, J. Guo, and X. Tang. Pairwise rotation invariant co-occurrence local binary pattern. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2199–2213, 2014.

A. Ranganathan. The levenberg-marquardt algorithm. *Tutoral on LM algorithm*, 11(1):101–110, 2004.

R. Raskar, G. Welch, and H. Fuchs. Spatially augmented reality. In *First IEEE Workshop on Augmented Reality (IWAR'98)*, pages 11–20, 1998.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. 2011.

S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152, 2001.

R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

J. Salvi, J. Pages, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern recognition*, 37(4):827–849, 2004.

D. Schmalstieg and T. Höllerer. Augmented reality: Principles and practice. In *2017 IEEE Virtual Reality (VR)*, pages 425–426. IEEE, 2017.

R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

A. Sheffield University. How the kinect works depth biomechanics, 2017. URL http://www.depthbiomechanics.co.uk/?p=100/.

M. Simulink. Camera calibrator, 2017. URL https://www.mathworks.com/help/vision/ug/camera-calibration.html.

M. Smith, J. Carrivick, and D. Quincey. Structure from motion photogrammetry in physical geography. *Progress in Physical Geography*, 40(2):247–275, 2016.

W. Steptoe, S. Julier, and A. Steed. Presence and discernability in conventional and non-photorealistic immersive augmented reality. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 213–218. IEEE, 2014.

E. STINSON. Lightform: The magical little device that transforms whole rooms into screens. Web page, 2017. URL https://www.wired.com/2017/04/lightform-magical-little-device-transforms-whole-rooms-screens/. Accessed 2018/04/25.

N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085. IEEE, 2017.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

T. Viéville and O. D. Faugeras. Cooperation of the inertial and visual systems. In *Traditional and non-traditional robotic sensors*, pages 339–350. Springer, 1990.

S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2010.

H. M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984, 2006.

R. W. Wedderburn. Quasi-likelihood functions, generalized linear models, and the gauss—newton method. *Biometrika*, 61(3):439–447, 1974.

T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. 2012.

T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

J. Wilm, O. V. Olesen, and R. Larsen. Accurate and simple calibration of dlp projector systems. In *Emerging Digital Micromirror Device Based Systems and Applications VI*, volume 8979, page 897909. International Society for Optics and Photonics, 2014.

A. Wilson, H. Benko, S. Izadi, and O. Hilliges. Steerable augmented reality with the beamatron. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 413–422. ACM, 2012.

A. D. Wilson and H. Benko. Projected augmented reality with the roomalive toolkit. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, pages 517–520. ACM, 2016.

Y. Xiaoyang, M. Xiaoliang, W. Haibin, Z. Tianjian, and X. Zhenyu. Projector calibration method using orthogonal gray code combined with trapezoid phase shift code. *International Journal of Multimedia and Ubiquitous Engineering*, 1 (5):1–8, 2014.

S. Yamazaki, M. Mochimaru, and T. Kanade. Simultaneous self-calibration of a projector and a camera using structured light. In *CVPR 2011 WORKSHOPS*, pages 60–67. IEEE, 2011.

K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

Y. Yitzhaky and E. Peli. A method for objective edge detection evaluation and detector parameter selection. *IEEE Transactions on pattern analysis and machine intelligence*, 25(8):1027–1033, 2003.

B. Zhang, Y. Gao, S. Zhao, and J. Liu. Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor. *IEEE transactions on image processing*, 19(2):533–544, 2009.

G. Zhang, Z. Liu, J. Sun, and Z. Wei. Novel calibration method for a multi-sensor visual measurement system based on structured light. *Optical Engineering*, 49(4): 043602, 2010a.

L. Zhang and R. Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013.

X. Zhang and L. Zhu. Robust calibration of a color structured light system using color correction. In *International Conference on Intelligent Robotics and Applications*, pages 936–946. Springer, 2009.

Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4): 43–52, 2010b.

Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808. IEEE, 2016.