

A Thesis for the Degree of Ph.D. in Engineering

**Relative Pose Estimation
with RGB Camera:
Improvements and Applications**

February 2021

Graduate School of Science and Technology,
Keio University

Masahiro Yamaguchi

Abstract

With the development of automatic robot control and smartphone-based augmented reality applications, methods for camera pose estimation from image sequences taken by an RGB camera have attracted numerous attention recently. To estimate the camera pose from an image sequence, it is necessary to obtain the 3D structure of the environment and the 3D shape of the target object in the image sequence. The method of estimating both the 3D structure and the camera pose simultaneously is called Simultaneous Localization and Mapping (SLAM) and has been extensively studied recently. On the other hand, it is known that more robust camera poses can be obtained by using a previously obtained 3D structure.

This thesis proposes a method for improving the accuracy of camera pose estimation by obtaining a 3D structure of the target in advance and a novel method of visualization using the pre-measured 3D shape. First, a robust estimation of the relative camera poses is proposed using a pre-built 3D structure of the environment. The proposed method describes a solution to the problem of scale-mismatch in 3D space caused by online estimation of camera trajectories on a pre-built 3D structure. Next, this thesis proposes a novel method that focuses on extending visualization techniques by applying pre-measured 3D shapes to the estimation of relative camera poses. The second method shows a method of superimposing physical quantity distributions acquired by a camera with different modalities on a 3D map of the environment as estimated by a normal visible light camera. This allows users to visualize the 3D structure of the temperature distribution. The third method proposes a novel interactive visualization technique to present assembly tutorials in 3D space from an RGB image sequence. This approach utilizes connections between the parts and the relative pose from the camera to a given 3D object.

In this manner, this thesis describes how to apply pre-measured data to the estimated relative RGB camera poses in three different situations and

uses the results to improve the accuracy of camera pose estimation and apply them to visualization.

Acknowledgments

First and foremost, I would like to express the deepest appreciation to my main supervisor, Prof. Hideo Saito for his extraordinary mentoring. Prof. Saito not only gave me a lot of advice, but also supported my research activities by respecting my will. I learned a lot of things from Prof. Saito, including how to do research. I also appreciate him for providing me with many opportunities for research topics and collaborations.

I am also thankful for Prof. Issei Fujishiro, Assoc. Prof. Yuta Sugiura, and Assoc. Prof. Hideaki Uchiyama for being on my reading and defense committee. I greatly appreciate their all communications and their insightful suggestions and comments on my thesis.

I am very lucky and grateful to have worked with fantastic collaborators Assoc. Prof. Denis Kalkofen and Prof. Dieter Schmalstieg during my stays at Graz University of Technology. During my stay in Graz, Assoc. Prof. Kalkofen took care not only of my research but also of my living. This made it possible for me to spend a valuable and wonderful time doing research. I learned so much from Assoc. Prof. Kalkofen on research taste, paper writing, and presentation.

It has been a great experience to work with my amazing co-authors (in chronological order): Shohei Mori, Vincent Nozick, Trong Phuc Truong, Peter Mohr, Markus Tatzgern, Ana Stanescu. I am especially grateful to Dr. Mori for giving me a lot of advice on my research. I would like to express my sincere thanks to Prof. Kenji Kono for the grant of CREST-JPMJCR1683 during the term of my Ph.D. student. In addition, I appreciate Shoji Yachida and the other members of NEC for the six years of collaborative research.

I am also grateful to my lab mates for sharing brilliant time with me by working together and by talking about things other than just our research.

I also give sincere thanks to my family and my parents.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Overview	1
1.2 Thesis Outline	4
2 Camera Trajectory Estimation in Pre-built Global Map	8
2.1 Introduction	8
2.2 Related Works	11
2.3 Method	13
2.3.1 Method Overview	13
2.3.2 Global Scene Map Generation Using SfM	14
2.3.3 Pose Estimation of VO on Global Map	15
2.3.4 Keyframe Pose Refinement	17
2.4 Experiments	19
2.4.1 Implementation	19
2.4.2 Datasets	20
2.4.3 Results of ATE with Sim(3)-alignment	20
2.4.4 Results of ATE Without Sim(3)-Alignment	22
2.5 Summary	24

3	Pose Estimation between Different Modality Cameras	26
3.1	Introduction	26
3.2	Related Works	28
3.3	Method	29
3.3.1	Camera Calibration	31
3.3.2	Relative Scale Between the Point Cloud	40
3.3.3	Generating RGB-T Image from the Point Cloud	47
3.4	Experiment	49
3.4.1	Experimental Setups	49
3.4.2	Calibration Refinement	49
3.4.3	Thermal Maps and RGB-T Images	50
3.4.4	Evaluation of the Estimated Scale	51
3.5	Summary	53
4	Object Pose Estimation toward AR Applications	56
4.1	Introduction	56
4.2	Related Works	61
4.3	Method	63
4.3.1	Determining the Assembly Sequence	63
4.3.2	Visualization	74
4.4	Experiment	80
4.5	Summary	89
5	Conclusion and Outlook	92

List of Figures

1.1	Outline	2
2.1	Registering a locally growing VO	9
2.2	Method Overview	13
2.3	Graph optimization	19
2.4	Registering a VO map	20
2.5	Registering a VO map in the global map generated by LIDAR	22
2.6	Result of all sequences in the EuRoC Machine Hall dataset . .	23
2.7	The trajectory error accumulation	24
3.1	Overview of the proposed method	30
3.2	Blur on the thermal-infrared camera image	33
3.3	Estimation of distortion parameter.	34
3.4	Initial detection and refining the points.	35
3.5	Constraints of cost function.	39
3.6	Variation of back-projection by different scales	42
3.7	Result of back-projection to thermal-infrared images using dif- ferent scales	44
3.8	Generating a depth map using a multi-view stereo (MVS) al- gorithm	45
3.9	Variation of the MI score with scale	47
3.10	RGB-T image	48
3.11	Camera rig	49
3.12	Thermal maps	51
3.13	RGB-T images from various scenes	52
3.14	Evaluation of the estimated scale	54

4.1	Video annotated assembly tutorial	58
4.2	Animating the 3D assembly model	60
4.3	Registration	66
4.4	Tracking and removal detection	69
4.5	Sequence refinement	72
4.6	AR visualization	76
4.7	Texture filling and label extraction	78
4.8	Label placement	80
4.9	Apparatus	82
5.1	Object detection using global map	94

List of Tables

2.1	ATE in the EuRoC dataset (RMSE)	21
3.1	Calibration RMS in pixel	50
3.2	Mean and standard deviation of the relative error between the estimated scale and the ground truth.	53

Chapter 1

Introduction

1.1 Overview

Recently, many devices, such as robots, cars, and smartphones, have been equipped with various devices, such as GPS, Lidar, and IMU in addition to an RGB camera. There are many types of RGB cameras: stereo, wide-angle, and fisheye cameras. By measuring the environment, these sensors can be used to estimate the self-pose of devices. Estimating self-pose is useful for efficient robot movement, applications such as augmented reality (AR), and obstacle detection in the environment. However, the more sensors a device has, the more expensive it becomes, thus the number of sensors and the type of cameras vary depending on the cost. When considering sensors' selection for applications, an RGB camera is among the preferred candidates for selection. This is because they are mounted on many devices and provide numerous pieces of information. Using only a monocular RGB camera not

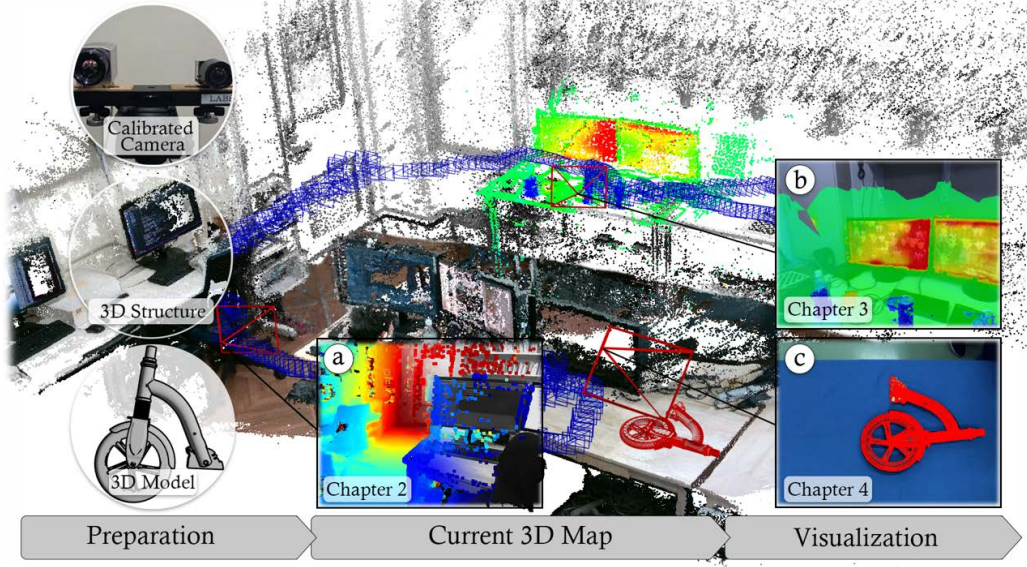


Figure 1.1: **Outline of this thesis.** This represents the 3D space assumed by this thesis. (a) performs the camera pose estimation on a pre-built 3D map. In addition, (b) and (c) are the proposed visualization methods using relative camera poses.

only reduces the cost and eliminates the need for sensor synchronization, but it also reduces accuracy and limits what can be done.

Estimating the camera poses from an RGB image sequence requires obtaining the 3D structure of the environment and the 3D shape of the target object. Visual Odometry (VO) and visual Simultaneous Localization and Mapping (vSLAM) [20, 16, 17, 51], which are methods for simultaneously estimating the camera poses while measuring the 3D structure, are one of these methods and have been studied extensively. These methods use only a series of images for camera pose estimation to maintain real-time. On the other hand, it is known that a more accurate estimation of camera pose

can be achieved by combining a previously obtained 3D shape. Therefore, this thesis proposes a method for improving the accuracy of camera pose estimation by obtaining a 3D structure of the target in advance and a novel method of visualization using the pre-measured 3D shape. In addition, this thesis also solves the scale problem that arises when using pre-measured 3D shapes to estimate the current camera pose.

Figure 1.1 represents the three main categories of using the relative camera poses in this thesis. Using the obtained 3D shapes in advance, this thesis proposes a method for improving the accuracy of camera pose estimation and a novel method of information visualization. First, Figure 1.1(a) describes a method to improve the accuracy of camera pose estimation using the pre-built 3D structure of the environment. The relative poses from the static environment to the camera are estimated here. This is described in Chapter 2. While Figure 1.1(a) is concerned with improving the accuracy of camera pose estimation, Figures 1.1(b) and 1.1(c) represent a novel method focusing on extending the visualization technique by applying pre-measured 3D shapes to the relative camera pose. Chapter 3 describes a method of superimposing physical quantity distributions acquired by cameras with different modalities on a 3D map of the environment as estimated by ordinary visible light cameras. Consequently, the temperature distribution is visualized on the RGB image using a 3D structure (Figure 1.1(b)). The relative poses from the camera to the static environment are estimated here. Finally, Figure 1.1(c) estimates the relative pose from the camera to a given 3D ob-

ject. Using this approach, Chapter 4 proposes a new visualization method to present instructions in 3D space from RGB image sequences. The relative poses from the camera to the dynamic objects are estimated here. In this thesis, the moving objects excluding the camera and the fixed objects are defined in the world coordinate system as dynamic and static objects, respectively.

1.2 Thesis Outline

This thesis comprises the following chapters. In this thesis, since the relative camera poses are required in all chapters, each chapter does not refer to the poses of the original object.

Chapter 2 describes the trajectory estimation of the current RGB camera in a pre-built 3D map. By estimating the camera pose in the pre-built 3D space, more robust poses can be estimated than ordinary VO and SLAM, and the data registered in the 3D space can be obtained using the camera position. In general, VO and SLAM are used to estimate the camera pose in real-time from a continuous image sequence. However, when estimating the camera pose using VO and SLAM, the scale of the estimated 3D structure always differs because the estimated 3D space depends on the initialization step size. Thus, the scale of the estimated space mismatches that of the pre-built space. Therefore, this chapter proposes a method for estimating the camera trajectory using VO under a 3D space (the global map) generated in

advance using SfM. The proposed method re-estimates the camera poses on the pre-built global map using the input frame of a VO with high similarity to the images for the environment map. Based on the re-estimated camera poses, the trajectory estimated by the VO is recomputed on the scale of the global map. The proposed method shows that it is possible to estimate camera trajectories using the VO on an environment map generated by a priori observations and also shows that it is possible to estimate the camera pose more robustly than ordinary VO and SLAM. The content of this chapter is based primarily on [92].

Chapter 3 proposes a method to visualize different physical quantity distributions on a 3D structure obtained from an RGB camera image sequence using pre-measured positional relationships between cameras of different modalities. When calibrating the relative poses between cameras, the length of one side of the calibration board can be used to obtain the real scale of the relative poses between cameras. On the other hand, as mentioned earlier, when only a monocular RGB camera is used for camera tracking, the resulting camera poses do not correspond to the real-world scale because they are relative. In general, when using cameras of the same modality, the tracking scale can be fixed to the real world by initializing the tracking from the distance images obtained by stereo matching between cameras. On the other hand, when using cameras with different modalities, it is not possible to relate the relative poses of the cameras to the obtained 3D space because it is not possible to estimate the distance images by stereo matching. Therefore,

this chapter proposes a method to solve this problem for RGB and thermal-infrared cameras, and to superimpose the temperature distribution obtained by a thermal-infrared camera on the 3D structure obtained by VO. Using a group of images obtained from both RGB and thermal-infrared cameras, depth images are generated by multi-baseline stereo with poses obtained from the VO, respectively in advance. Temperature distribution can be superimposed on the 3D structure estimated from VO using the mutual information of the obtained depth images. In this manner, this chapter shows a method to visualize the temperature distribution superimposed on a 3D structure. The content of this chapter is based primarily on [93].

Chapter 4 proposes a novel interactive visualization technique to present assembly tutorials in 3D space from an RGB image sequence. This approach utilizes connections between the assembly parts and the relative pose from the camera to a given 3D object. This method enables interactive visualization by estimating the relative pose from the camera to an object with a given 3D shape. As an example of visualization using such a given object pose, this chapter proposes a method to visualize the assembly instructions of an object in 3D space. These assembly instructions are usually written in a manual or with instructional videos uploaded to YouTube. However, by estimating the relative poses of objects from the camera, such tutorials can be visualized in 3D space and presented clearly. This chapter proposes to generate 3D AR tutorials from 2D video input by estimating the relative pose of the RGB camera to an object. The video input is analyzed

using a 3D model with an assembly graph. The usefulness of this method was demonstrated by comparing the tutorials with videos. The content of this chapter is based primarily on [91].

Finally, **Chapter 5** provides conclusions of this thesis and some suggestions for future work.

Chapter 2

Camera Trajectory Estimation in Pre-built Global Map

2.1 Introduction

Real-time camera pose estimation is an essential function for AR systems in registering 3D content to the scene. A scene size can vary from a desktop to a city scale, and depending on the scale, the feasible hardware for camera pose estimation also changes. Since outside-in tracking becomes impractical in wide areas, AR systems with wide scalability rely on inside-out tracking.

Stand-alone inside-out tracking frameworks, such as VO and vSLAM, use vision sensors, i.e., a camera, to achieve pixel-wise registration in the user’s view. However, VO accumulates errors over time and drifts from the original location. Although SLAM can mitigate this error by detecting re-visits in a scene and attempting to cancel the accumulated errors, it suffers from the drift before the loop close detection. This makes location-based

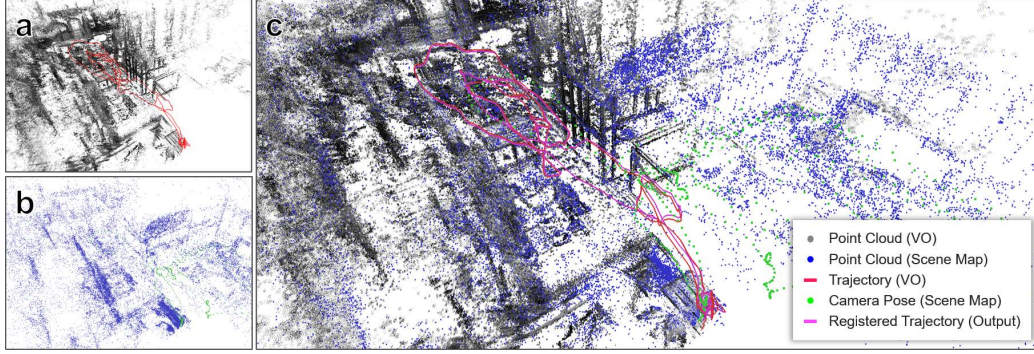


Figure 2.1: **Registering a locally growing VO.** (a) to a globally optimized scene map (b). Since both maps are in individual scales and individual coordinate systems, the proposed method registers the VO to the scene map by reducing such differences at run time (c). This allows the location-based AR system using the VO to retrieve AR contents registered in the scene map.

AR error-prone, especially in wider areas, since the drifted position triggers unrelated AR content.

Since VO and SLAM provide only temporal personalized scene tracking to AR, scene registered content can be created only at the runtime and will be paused in the next trial. Therefore, to enable a consistent AR experience on a daily basis, AR developers need to register their content to pre-built common-scene maps, and AR systems are required to match their executing VO or SLAM to the scene map to access pre-built content. Consequently, the scene map creation must be done in a stage earlier than the user's AR experience.

To satisfy these AR-specific needs, this chapter proposes a new camera pose registration method using VO with a pre-built scene map. The proposed method enables feeding a pre-built scene map to a VO. In this

method, a locally running VO can refer to the preserved scene map's information immediately after the execution. This means that this tracking framework can bootstrap the VO within the scene map scale and update the current camera pose with pose graph optimization without closing the VO's trajectory loop by itself. Figure 2.1 shows snapshots of a globally optimized scene map (Figure 2.1(a)) and a locally growing VO map on different scales (Figure 2.1(b)). The proposed method recalculates the scale difference of the VO and the scene map on the fly and continues updating the VO map when the scene map is available (Figure 2.1(c)). Contributions of the proposed method are summarized as follows:

- A camera tracking framework that automatically registers the local user's VO map to a pre-built scene map relying only on a color camera. With this, the user can receive AR content in the scene map within the adjusted scale immediately after the method finds a matching between the current undergoing VO map and the world map. Additionally, this can mitigate drift errors that would be accumulated over time with solely the VO.
- An approach to match the scale between the VO map and a scene map to provide scale-consistent content in the AR space.
- The results of quantitative evaluations, demonstrating the superiority and the limitations of the proposed method.

One can find several similar approaches that combine VO and a pre-

built scene map [59, 65]. The major difference is that such approaches rely on the inertial measurement unit (IMU), i.e., visual-inertial odometry (VIO) [60], for stability and the absolute scale factor, whereas the proposed method does not, i.e., VO receives only a video stream.

2.2 Related Works

Camera pose estimation methods for AR using a color camera are divided into three major approaches: VO, SLAM, and pre-built map-based tracking.

VO and SLAM: VO [20, 16] is a camera tracker that gives pixel-wise content registration in the AR view. As VO is designed to optimize the poses and the map regarding several latest frames, it suffers from drift errors over time. SLAM [17, 23, 51] is an alternative designed to reduce drift errors with a global optimization process such as Bundle Adjustment (BA) and a loop closure scheme [75].

Regardless of the global optimization process, both approaches use temporally built maps to track the scene. The reason behind this is that VO and SLAM provide different scale factors in every trial depending on how the user moves the camera. This prevents AR applications from fetching pre-built content. VIO is one of the choices used to overcome the scale difference issue, as it provides a real-scale map. Several approaches [59, 65] have already proposed such methods in the last few years. GPS can also be a tool to obtain a real-scale map in SLAM [80]. In contrast to these sensor

fusion approaches, only a monocular camera is relied on to minimize the hardware required for AR systems in this method. To this end, this method uses a pre-built map and estimates a scale from the locally running VO and the pre-built scene map.

Pre-built Map-Based Tracking: Location-based AR applications must have an interface to link the camera pose and the location to trigger location-specific AR content. One popular approach to achieve this is to register the camera within a preserved scene map to have access to the map-registered content.

Landmark database-based approaches use maps built with Structure from Motion (SfM) to estimate camera poses in the map by linking observed feature points and those in the database map [74, 81, 37]. Thereby lacking feature point matching results in the tracking failures. This approach uses VO which keeps tracking the camera using its online local map. PTAMM can relocalize the camera in multiple local maps distributed in a scene [12]. This approach is only applicable to room-scale scenes, where no loop closure scheme is required, due to the limited scalability of the core SLAM method. This method can scale from a desktop environment to a city-scale environment with the empowerment of the state-of-the-art VO.

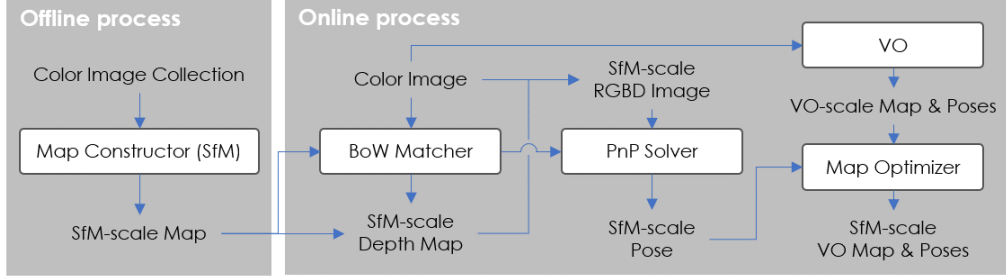


Figure 2.2: Method Overview.

2.3 Method

This chapter proposes a method capable of registering and refining a trajectory of a locally running VO using a scene map optimized in advance, potentially with higher accuracy than what naïve VO and SLAM can provide. To this end, this chapter proposes a framework that provides an SfM scale in ongoing VO and proposes to match and optimize the VO trajectories in an SfM-scale map.

2.3.1 Method Overview

Figure 2.1 shows an overview of the proposed method. Given a global map of scene $\mathcal{G}$ that contains frame depth maps in the scale s^{SfM} , poses, and Bags of Words (BoW) database [22], a camera starts exploring the scene, and VO estimates the trajectory in its own scale map s^{VO} . When the framework detects the best match of the incoming frame to a frame in $\mathcal{G}$, it calculates the corresponding pose in the SfM scale. Given a collection of such poses,

the proposed method optimizes the current VO trajectory through graph optimization. Although this approach best fits VO, VO could be replaced with SLAM without losing generality. SLAM is a framework that includes map optimization by itself, so VO is the minimum configuration for the proposed method.

2.3.2 Global Scene Map Generation Using SfM

Given M images, a map $\mathcal{G}$ is constructed using SfM before actual VO tracking starts. As the maps generated by SfM [66] are known to be accurate compared to the ones created by SLAM and VO due to their global optimization nature, the global map $\mathcal{G}$ is not updated during VO tracking. On the other hand, the VO map is optimized at runtime to match the map to the stable global map. Such a global map comprises color frames $\mathcal{I}^{\text{SfM}}$, depth maps at each frame $\mathcal{D}^{\text{SfM}}$, and associated frame poses $\mathcal{T}^{\text{SfM}}$. Hereafter, this chapter denotes the $i_{th}(< M)$ color frame, depth frame, and their pose as $I_i^{\text{SfM}} \in \mathcal{I}^{\text{SfM}}$, $D_i^{\text{SfM}} \in \mathcal{D}^{\text{SfM}}$, and $\mathbf{T}_i^{\text{SfM}} \in \mathcal{T}^{\text{SfM}}$, respectively. In addition, the proposed method uses BoW with ORB features [63], $F_i^{\text{SfM}} \in \mathcal{F}^{\text{SfM}}$, detected at each frame I_m to relate the frames in the global map $\mathcal{I}^{\text{SfM}}$ with the frames given to VO, i.e., the global map is defined as $\mathcal{G} \in \{\mathcal{I}^{\text{SfM}}, \mathcal{D}^{\text{SfM}}, \mathcal{T}^{\text{SfM}}, \mathcal{F}^{\text{SfM}}\}$. Note that $\mathbf{T}_i$ represents the 4×4 pose matrix consisting of a 3×3 rotation matrix and a 3×1 translation vector.

2.3.3 Pose Estimation of VO on Global Map

As the baseline length at the initialization of a monocular VO is unknown in most cases, such a VO estimates a camera trajectory and a corresponding map in an arbitrary scale given at a bootstrapping stage based on the step width [16]. Although a stereo VO [87] can use a calibrated baseline length between the two cameras to obtain a scale, fitting the scale to that for a global map is another issue, unless these scales are calibrated in the real unit [60, 85]. Instead of bootstrapping VO from scratch, $\mathcal{D}^{\text{SfM}}$ is used to feed the scale of $\mathcal{G}$, i.e., s^{SfM} , to VO. Given a VO keyframe $I^{\text{KF}} \subset \mathcal{I}^{\text{VO}}$ and its BoW vector F^{KF} , a depth frame D_i^{SfM} that has a frame index i is searched, satisfying the following condition:

$$1 - \frac{1}{2} \left| \frac{F_i^{\text{SfM}}}{|F_i^{\text{SfM}}|} - \frac{F^{\text{KF}}}{|F^{\text{KF}}|} \right| > t_{\text{BoW}}, \quad (2.1)$$

where t_{BoW} is a user-given threshold.

If multiple frames are obtained, the top several frames are used in the order of the highest score. Once such a frame index is found, 3D points are calculated from the depth map D_i^{SfM} . Detecting and matching feature points in I_i^{SfM} and I^{KF} give their 2D–2D correspondences, and the unprojected 3D points at such feature points in I_i^{SfM} give 3D–2D correspondences between I_i^{SfM} and I^{KF} . Solving the perspective-n-point (PnP) problem with a RANSAC robust estimator gives the pose of the keyframe I^{KF} , T^{KF} , in scale s^{SfM} .

Among the 2D points from which 2D–2D matching is obtained, the points where the depth image $\mathcal{D}^{\text{VO}}$ obtained from the VO has a value is used to obtain the relative scale $s^{\text{VO} \rightarrow \text{SfM}}$ of the VO and the global map $\mathcal{G}$. When VO uses the localization function, the same feature points obtained from the global map and VO are used. However, this method matches the SIFT feature points used for SfM with the point of direct reference to the pixel value used for VO. The 3D point clouds created from SfM and VO cannot be treated as the same points. Therefore, it is necessary to recalculate the pose using such a matching-based depth image and scale correction to reduce the drift error.

In the frames processed with matching, let $\hat{\mathbf{T}}^{\text{KF}}$ be the camera pose obtained by solving the PnP problem and $\mathbf{T}^{\text{KF}}$ be the camera pose estimated by VO. With the depth value d estimated by the VO, the image coordinate $\mathbf{i} = (u, v, 1)^\top$ of the VO can be expressed as $\mathbf{t}_{\text{VO}} = d\mathbf{K}^{\text{VO}^{-1}}\mathbf{i}$ for the corresponding VO camera coordinate system. With this, the transformation to the camera coordinate system of SfM can be expressed using the following equation:

$$\begin{pmatrix} \mathbf{t}_{\text{SfM}} \\ 1 \end{pmatrix} \simeq \mathbf{T}^{\text{SfM}^{-1}} \hat{\mathbf{T}}^{\text{KF}} \mathbf{T}^{\text{KF}^{-1}} \begin{pmatrix} \mathbf{t}_{\text{VO}} \\ 1 \end{pmatrix}, \quad (2.2)$$

where the third component of $\mathbf{t}_{\text{SfM}}$ represents the depth of each pixel in the SfM image coordinate system, which is denoted by d_p^{VO} . Let be the depth value obtained by SfM for the corresponding pixel and the relative scale can be calculated using the following equation:

$$s^{\text{VO} \rightarrow \text{SfM}} = \frac{1}{n} \sum_{p \in \Omega} \frac{d_p^{\text{SfM}}}{d_p^{\text{VO}}}, \quad (2.3)$$

where $p \in \Omega$ represents a pixel p in the two-dimensional plane Ω of the image. Here, the pixels p are only those pixels that have obtained 2D–2D matching in solving the PnP problem and have a value of $\mathcal{D}^{\text{VO}}$. The total number of pixels satisfying the condition is n . $\mathcal{D}^{\text{VO}}$ is the depth image obtained by the VO and is sparser than the one obtained by the depth sensor.

2.3.4 Keyframe Pose Refinement

In the keyframe matched with $I^{\text{SfM}} \in \mathcal{I}^{\text{SfM}}$ by BoW, solving the PnP problem with 3D–2D enables the camera pose to be obtained in scale s^{SfM} . Furthermore, $s^{\text{VO} \rightarrow \text{SfM}}$ is used to convert from the camera pose of VO to the global map for frames without a camera pose corresponding to the scale s^{SfM} from the 3D–2D correspondences. Frames that are corrected for poses are divided into two groups: the frames from the current frame to the latest matching frame with similar frames (active frames) and the frames before that (inactive frames). The following equation is applied to the active frames to convert them from the VO coordinate system to the corresponding global map coordinate system. Note that j is the index of the frame to be transformed, and i is the index of the frame for which matching is obtained.

$$\mathbf{T}_j^{\text{VO} \rightarrow \text{SfM}} := s^{\text{VO} \rightarrow \text{SfM}} \hat{\mathbf{T}}_i^{\text{KF}} \mathbf{T}_i^{\text{KF}^{-1}} \mathbf{T}_j^{\text{VO}}, \quad (2.4)$$

where $\mathbf{T}^{\text{VO} \rightarrow \text{SfM}}$ represents the camera pose in the global map coordinate system and $\mathbf{T}^{\text{VO}}$ represents the camera pose in the VO coordinate system. The depth value obtained in frame j is also corrected to the scale of the global map by $s^{\text{VO} \rightarrow \text{SfM}}$.

In addition, graph optimization is used to correct the camera poses for inactive frames. The graph structure is constructed by connecting adjacent inactive frames. Consequently, an edge is constructed as $\mathbf{e}_{i,i-1}$, for the i_{th} keyframe, as follows:

$$\mathbf{e}_{i,i-1} = ((\mathbf{T}_{i-1}^{\text{KF}})^{-1} \mathbf{T}_i^{\text{KF}})((\hat{\mathbf{T}}_{i-1}^{\text{KF}})^{-1} \hat{\mathbf{T}}_i^{\text{KF}}), \quad (2.5)$$

where $\hat{\mathbf{T}}_i^{\text{KF}}$ represents the i_{th} estimated pose from the PnP solver and $\mathbf{T}_i^{\text{KF}}$ represents the i_{th} estimated pose from VO. The following pose graph is optimized using the g2o algorithm [38]:

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i \rangle \in N} \mathbf{e}_{i,i-1}^T \Omega_{i,i-1} \mathbf{e}_{i,i-1}, \quad (2.6)$$

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathbf{F}(\mathbf{x}), \quad (2.7)$$

where N is the last matched keyframe index, $\mathbf{x}^* = (\mathbf{T}'_0, \dots, \mathbf{T}'_N)$ is an array of refined poses, and $\Omega_{i,i-1}$ is an information matrix. Equation (2.7) is optimized with the Levenberg Marquardt algorithm. Note that the first keyframe pose, i.e., $\mathbf{T}_0$, is fixed, whereas the other poses are not.

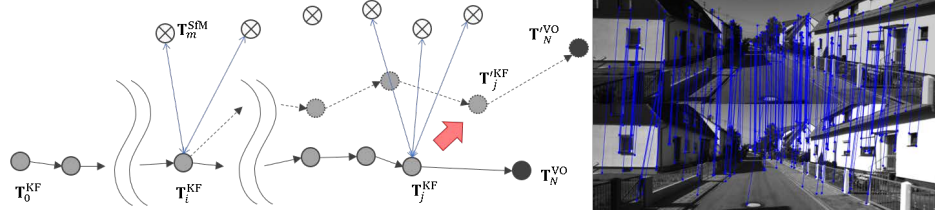


Figure 2.3: **Graph optimization.** Graph optimization using an environment map (left) and a matching result (right).

2.4 Experiments

The proposed method was validated using the public datasets of two different kinds, and the results were compared to those of state-of-the-art approach Direct Sparse Odometry (DSO) [16]. Through the evaluations, it is demonstrated how the proposed approach is capable of mitigating accumulative drift errors in VO with the help of pose updates using a scene map.

2.4.1 Implementation

SfM is performed using COLMAP [66] to generate an environmental map $\mathcal{G}$. There are two ways to generate depth images of COLMAP: photometric and geometric. This method uses the geometric depth image [67]. In addition, ORB features were computed for each frame and created an image database using DBoW3. In order to reduce the drift error against the camera pose at the time of the shooting, the captured images were thinned out from the consecutive image sequences by an amount of one out of every five images to be inputted to COLMAP.

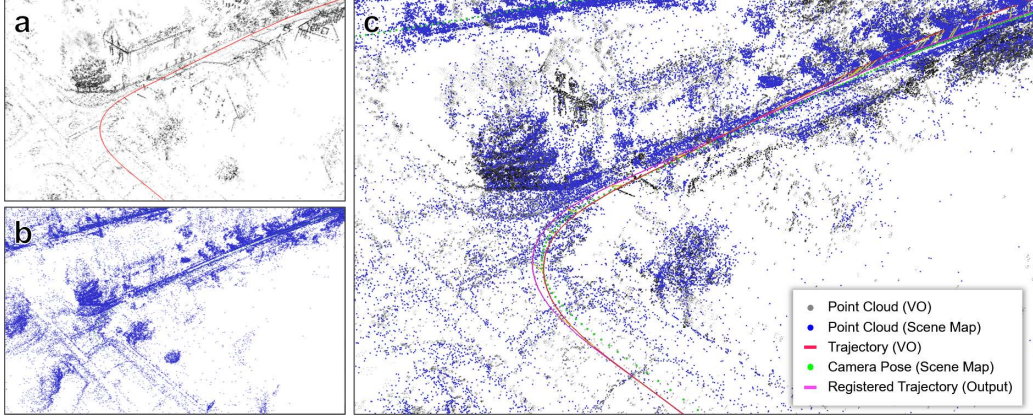


Figure 2.4: **Registering a VO map.** (a) to a scene map (b) in the KITTI dataset sequence 05. (c) Both maps are matched regarding the coordinate systems and the scales.

2.4.2 Datasets

The KITTI dataset is a large dataset of in-vehicle stereo cameras driving through a city. Eleven image sequences are provided in the EuRoC MAV dataset, including stereo images, IMU data, and ground truth of the camera trajectory. The image sequences were taken in three different environments; Machine Hall provides five different sequences, Vehicon Room 1 provides three different sequences, and Vehicon Room 2 provides three different sequences.

2.4.3 Results of ATE with Sim(3)-alignment

The proposed method was validated by measuring the absolute trajectory error (ATE) in the EuRoC MAV dataset. The ATE contains only VO

Table 2.1: **APE in the EuRoC dataset (RMSE)**

Sequence	Proposed Method					DSO	(Colmap)
VO\G	MH01	MH02	MH03	MH04	MH05	–	–
MH01	–	0.046	0.069	0.058	0.069	0.280	(0.031)
MH02	0.029	–	0.049	0.074	0.054	0.270	(0.017)
MH03	0.124	0.173	–	0.254	0.185	0.630	(0.036)
MH04	0.249	0.258	0.344	–	0.068	0.572	(0.049)
MH05	0.104	0.150	0.272	0.085	–	0.509	(0.045)
VO\G	V101	V102	V103	–	–	–	–
V101	–	0.148	0.200	–	–	0.344	(0.141)
V102	0.080	–	0.094	–	–	0.631	(0.049)
V103	0.830	1.242	–	–	–	1.009	(0.063)
VO\G	V201	V202	V203	–	–	–	–
V201	–	0.105	×	–	–	0.262	(0.030)
V202	0.142	–	0.052	–	–	0.589	(0.025)
V203	×	1.544	–	–	–	1.611	(0.029)

keyframes since such keyframes are the optimized ones. Note that the resultant trajectories were optimized with respect to the ground truth using Sim(3)-alignment, i.e., Umeyama Alignment [79]. Table 2.1 shows the quantitative results of Machine Hall and Vehicon Rooms 1 and 2, respectively. The cross marks in the tables show there were not enough matching points between the input and global map frames and this method failed to bootstrap. Figure 2.1 and Figure 2.4 show qualitative comparisons of a global scene map, a VO map, and a registered VO map in the global scene map in EuRoC Machine Hall and in KITTI sequence.

Instead of using the SfM results in Figure 2.5, the global map was generated from ground truth camera pose using GPS and depth images using LIDAR. Figure 2.5(b) is a distance image created from LIDAR, and Fig-

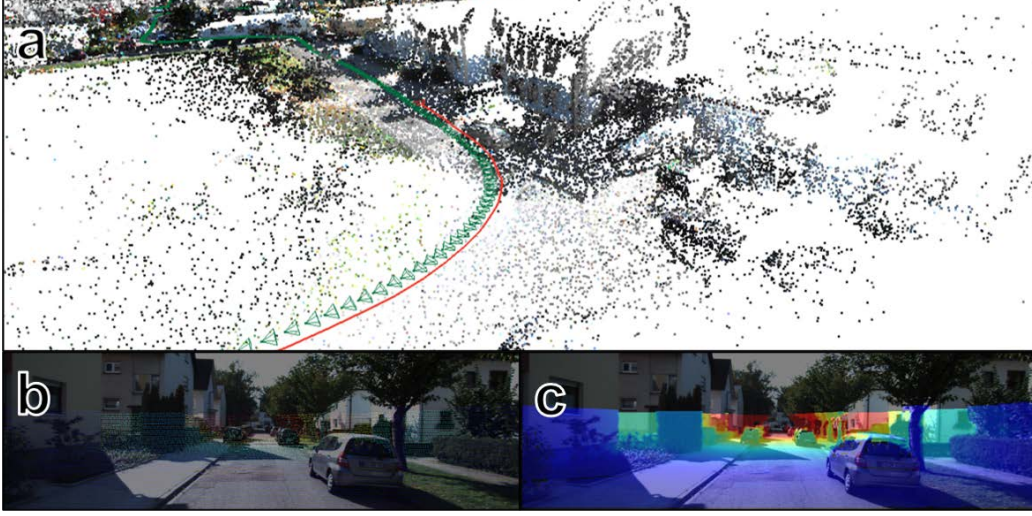


Figure 2.5: **Registering a VO map in the global map generated by LIDAR.** (a) Output results for VO in the global maps created by LIDAR in the KITTI dataset. (b) Depth image obtained from LIDAR. (c) Dense depth image by filtering.

Figure 2.5(c) was created by making it dense using a bilateral filter. The result of the proposed method is shown in Figure 2.5(a).

Moreover, Figure 2.6 shows the results of alignment and scaling for all five Machine Hall sequences using this method with MH01easy as the environmental map.

2.4.4 Results of ATE Without Sim(3)-Alignment

One of the advantages of this approach is that it can fit the scale of VO to the global scene map automatically. This section finally demonstrates how the trajectory errors accumulate in naïve DSO and how this approach can reduce the errors. VO, in the proposed framework, initializes the scale

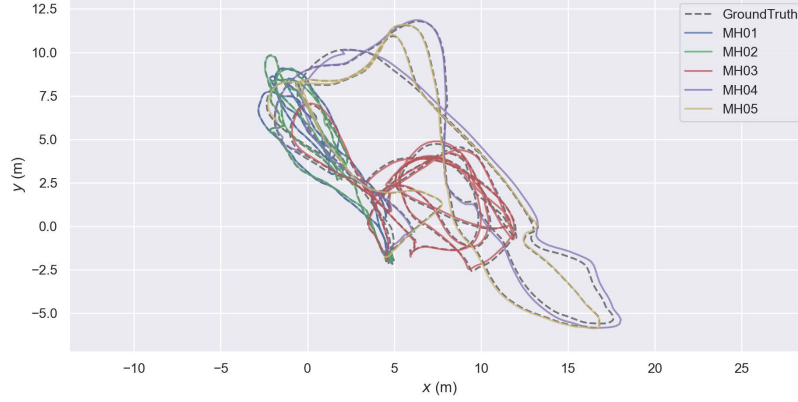


Figure 2.6: **Result of all sequences in the EuRoC Machine Hall dataset.** Results of trajectory estimation using the same environment map for all sequences of the EuRoC Machine Hall dataset

factor using an SfM depth map and updates it when the framework finds matching between an observed frame and preserved frame in the scene map. For fair comparisons, the naïve DSO is bootstrapped with the SfM depth map but does not update the scale in the later frames. Figure 2.7 shows the frame-by-frame differences in ATE of each tracking.

One interesting observation from Figure 2.7 is that the color gradients of this approach show some spikes even though it achieves lower errors across the frames. These spikes were observed as jumps right after the pose updates when VO is pulled back to the global scene map every time VO matches one of the global scene map frames. However, through this process, this approach significantly reduces the number of accumulative errors of VO. This should be troublesome for AR applications. For example, annotations registered in the map will always shift in the AR view when the matching happens. Thus,

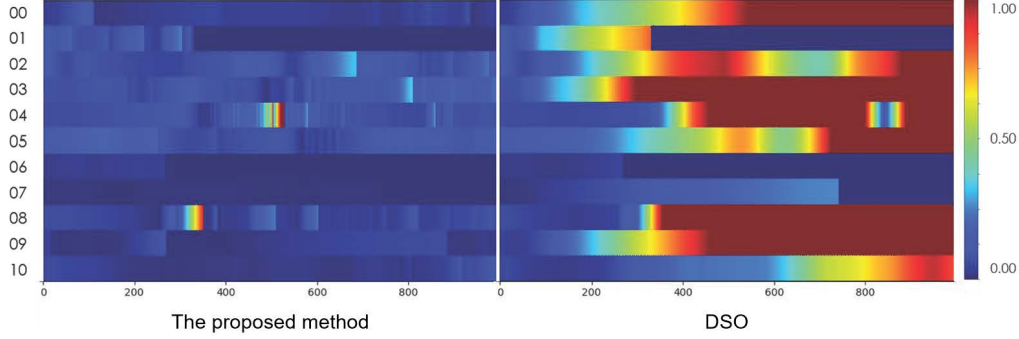


Figure 2.7: **The trajectory error accumulation.** The error accumulation with the proposed method and DSO in the KITTI dataset. ATE was calculated between the estimated poses and the ground truth poses at runtime. While this approach can keep the ATE significantly lower than the naïve DSO, some spikes occur before the pose update.

new pose update rules should be designed to change these system-oriented pose updates to the user-oriented pose updates, to reduce the user’s mental load.

2.5 Summary

This chapter proposed a method to register a locally running VO to an existing scene map. The proposed method refines upcoming camera poses by referring to a prepared global scene map from SfM, and registers the estimated VO trajectory to the scene map using pose graph optimization on the fly. This approach enables the reuse of a pre-built map, which potentially contains AR contents, to solve the scale ambiguity problem of VO and to reduce accumulative errors of VO. The results using public datasets demon-

strated that this approach could reduce the accumulative errors of VO.

Limitations

Since this method corrects the VO trajectories by referring to the measurement results of the pre-built global map, the errors of the pre-built global map propagate to the accuracy of the corrected VO trajectories. In other words, the method assumes that the pre-built global map is accurate, and therefore the method is not designed to compensate for errors in the pre-built global map. Furthermore, it is assumed that the images that are inputted to VO are relatively similar to the ones used to create the pre-built global map in terms of the lighting environment and capturing angle. If the lighting conditions differ, BoW-based similar image retrieval cannot produce accurate results. In case the capturing angles differ, the matching process does not provide enough corresponding points or may detect wrong correspondences, which may cause errors.

Chapter 3

Pose Estimation between Different Modality Cameras

3.1 Introduction

Thermal-infrared cameras measure infrared rays emitted from any object and can use this ray information to estimate the object's temperature. This temperature, which cannot be retrieved from RGB cameras, is valuable information that induces many applications in industrial domains and academic research. Indeed, thermal-infrared cameras can be used to detect gas leaks, fires, abnormalities in electronic apparatus, and so on. Most of these issues are difficult to be detected by visible light; thus, thermal-infrared cameras constitute a precious tool in these cases. However, thermal-infrared cameras also have some drawbacks. In practice, the materials used to produce the thermal-infrared camera lenses usually make their field of view narrow compared to the standard RGB camera lenses. Moreover, textures perceptible

in the visible spectral domain (i.e. captured by RGB cameras) are likely lost in invisible light. Since the human eye deals with visible light, it is often difficult for humans to understand thermal-infrared images due to this texture alteration.

These issues would be easily solved if it was possible to perfectly superpose a thermal-infrared image with its corresponding RGB image. This is feasible only if both RGB and thermal-infrared cameras share the same optical center (i.e., the same position). Some specific cameras, such as the one used in [94], can simultaneously capture RGB and thermal-infrared images. However, in standard situations, RGB and thermal-infrared images are captured using different cameras. This chapter addresses the problem of effectively associating thermal data with its corresponding RGB counterpart for visualization purposes. Contributions of the proposed method are summarized as follow:

- An accurate thermal-infrared camera calibration process.
- The overall process to superimpose thermal-infrared data on 3D reconstruction from the RGB images, including the specific problem of finding the unknown projective scale between RGB and thermal-infrared cameras.

3.2 Related Works

This chapter considers the situation where the acquisition device comprises at least an RGB and a thermal-infrared camera. In a standard situation, two or more cameras cannot share the same position. Thus, the matching of these data requires some geometric computation.

A feasible way to proceed comprises adding a 3D sensor to the thermal-infrared and RGB cameras to handle these geometric constraints. Borrmann et al. [7] used a laser scanner and an RGB camera. Due to its heavyweight, the laser scanner is attached to the RGB camera and mounted on a wheeled robot. The robot moves around the scene and performs laser scans repeatedly to generate a 3D RGB map. For each scan of the scene, the robot also acquires the corresponding 3D thermal map. While this method performs well, both the laser scanner and the robot are very expensive. Moreover, the need for a wheeled robot prevents the acquisition process from being conducted in scenes comprising stairs or other low-quality paths.

A cheaper alternative is to use an RGB-D sensor to generate a 3D model of the scene. Vidas et al. [83] and Matsumoto et al. [45] mounted a Kinect with a thermal-infrared camera on a hand-held stereo rig. They obtained a 3D structure of the scene using KinectFusion [52]. Although this method performs well in indoor situations, low-cost depth sensors, such as Kinect, usually fail in outdoor environments where sunlight affects the depth value acquisition [32, 3]. Moreover, the size of the acquired scene is usually

limited to small areas using KinectFusion.

Ham et al. [26] computed the geometry of the scene using the SfM technique [1] and superimposed the thermal information on the resulting 3D structure. This system uses only an RGB camera and a thermal-infrared camera, and thus handles both indoor and outdoor scenes. However, the calculation cost of the 3D reconstruction from the SfM is significant. Indeed, SfM combines all images of the considered sequence to match them, meaning that the generated 3D point cloud is accurate; however, this process induces a significant computational cost.

This chapter proposes a method combining an RGB camera with a thermal-infrared camera. As discussed in [26], this setup does not include any depth sensor and thus supports outdoor scenes. VO is used instead of an SfM approach, therefore the calculation cost is lower than in [26]. Moreover, this chapter proposes a robust calibration process for thermal-infrared cameras and an accurate camera relative pose for a more efficient thermal to RGB mapping.

3.3 Method

This chapter presents a method to generate RGB-T images (RGB plus Thermal) by superimposing thermal information obtained from the thermal-infrared camera onto the RGB images.

More specifically, the proposed system comprises four stages: first, an

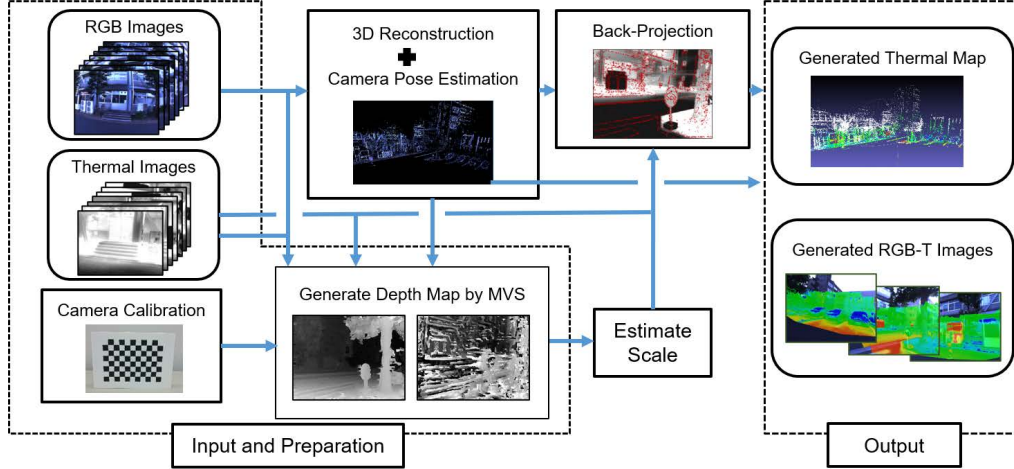


Figure 3.1: **Overview of the proposed method.**

RGB camera and a thermal-infrared camera are attached to a stereo rig and calibrated using a common calibration board which can be simultaneously detected from both cameras. Second, the RGB camera captures a set of images of the scene with which a 3D structure of the environment is generated at runtime. In the third step, the projective scale is estimated between the 3D structure and the calibration board using depth images generated from both RGB and thermal-infrared camera sequences. Finally, the thermal map and the RGB-T images are generated using the 3D structure computed from the RGB images and the estimated scale. An overview of the whole process is depicted in Figure 3.1.

3.3.1 Camera Calibration

(1) Calibration Board Issues

Camera calibration is a well-known process for cameras. This process, which usually requires point correspondences between images, has been widely studied for RGB images but appeared more challenging for thermal-infrared cameras due to the lack of textures. Assuming that the calibration process is performed using a calibration board, the thermal-infrared camera hardly detects the calibration pattern. Indeed, a thermal-infrared camera measures the temperature from the infrared rays emitted by objects. The locally emitted infrared light intensity depends more on the object temperature than the object color. Although it is true that black objects absorb and emit enormous infrared energy than brighter objects, in practice, a thermal-infrared camera cannot detect a checker pattern if the calibration board temperature is uniform across the black and white parts.

A general way to make the board detectable by the thermal-infrared camera is to heat the calibration board to make the re-emitted light significantly hotter than the intrinsic object temperature. Matsumoto et al. [45] used a hairdryer to heat the calibration board, while Ham et al. and Weinmann et al. [88, 26] used a set of lamps. Some other methods proposed by Prakash et al. and Saponaro et al. [57, 64] used a flood lamp to heat the calibration board. However, Vidas et al. [82] suggested that the use of a flood lamp is comparatively inaccurate.

In this method, the calibration board was designed from a printed chessboard pattern and the black plastic tape was placed on the black parts of the board. Then, a freezer was used to cool the calibration board. Heating the board with a lamp is also a good way to proceed; however, using a freezer is more convenient in our case.

The resulting image obtained after heating or cooling still presents some very blurred textures, as shown in Figure 3.2. This blur occurs because the board temperature quickly tends to diffuse from one cell to the next. Thus, after the heating or cooling process, the measured temperature gradually diffuses from a black part to a white one.

This blur is a serious issue for the calibration process not only for accuracy, but also for any automatic chessboard detection. The proposed method first estimates the camera lens distortion parameter using the plumb line method. Next, an initial estimation of the checker pattern corners is performed. These estimated points are significantly refined using a non-linear process involving constraints such as cross-ratio and vanishing point consistency. Note that the proposed method uses the pinhole camera model [28] for both the RGB and thermal-infrared cameras.

(2) Distortion Correction

The camera lens distortion correction is performed only once per camera unless the lenses are changed or manipulated (zoom or strong refocus). The plumb line method described by Devernay and Faugeras [15] is used to correct

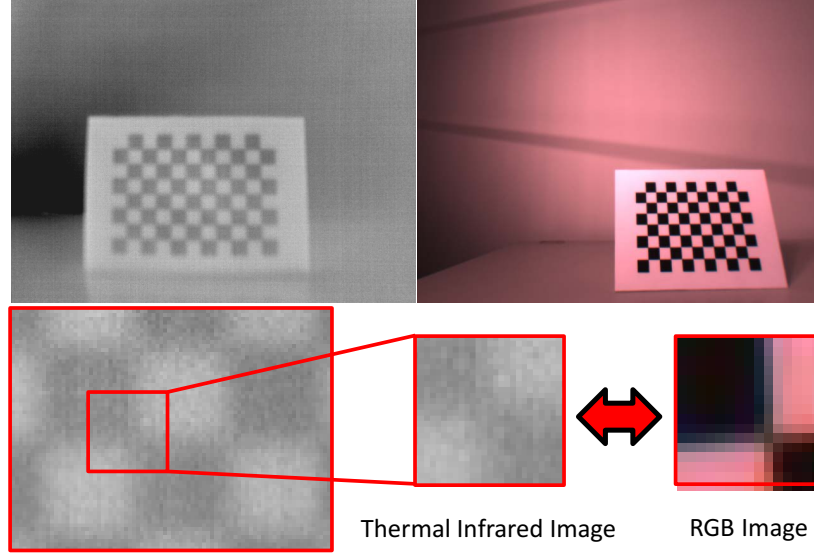


Figure 3.2: **Blur on the thermal-infrared camera image.** The top row shows images of a calibration board captured by both a thermal-infrared camera and an RGB camera. At the bottom row, a zoomed corner on a thermal-infrared image and an RGB image are compared. It is more difficult to detect a thermal-infrared image than an RGB image because of the blur.

the lens distortion by making straight lines of the scene also straight in the image. This method requires a set of points assumed to be aligned in the real world. The proposed method manually extracts these points from building images that contain good straight lines with strong thermal contrasts with the sky. The method defined in [15] non-linearly computes the fine radial distortion parameters such that the selected points become aligned. The resulting images present an excellent correction, as depicted in Figure 3.3. This radial distortion correction procedure is strongly required for the next steps of the camera calibration process.

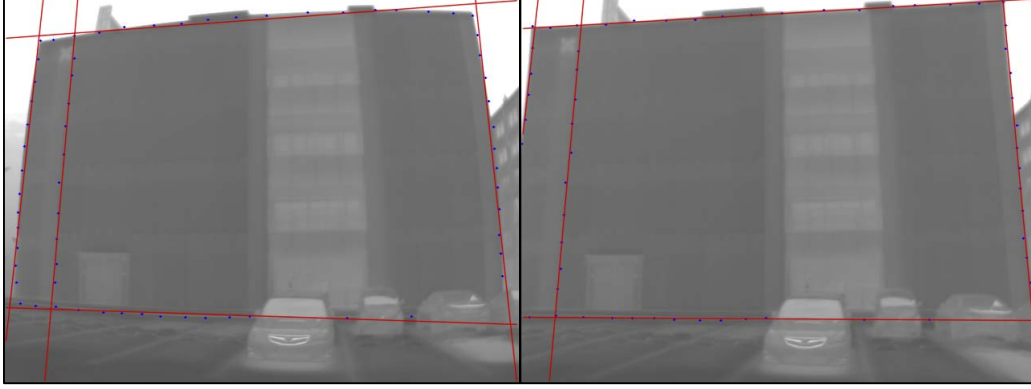


Figure 3.3: **Estimation of distortion parameter.** (Left) A set of points supposed to be aligned in the real scene and their least-square fitted line. (Right) Corresponding undistorted image.

(3) Initial Detection for Corner Refinement

A calibration chessboard is easily detected using automatic tools provided by OpenCV [8]. This tool automatically detects the corner of the chessboard and is widely used to calibrate RGB cameras. However, for the thermal-infrared camera, the inherent blur (Figure 3.2) drastically decreases the corner detection accuracy. The corner detection is clearly inaccurate and messy for thermal images. Since a good accuracy in camera calibration is required for the thermal to RGB mapping, the next section describes a chessboard detection refinement process.

(4) Corner Refinement Based on Line Intersection

This process starts from an undistorted image of the calibration chessboard obtained from the earlier described process. Thus, every corner of the checker-

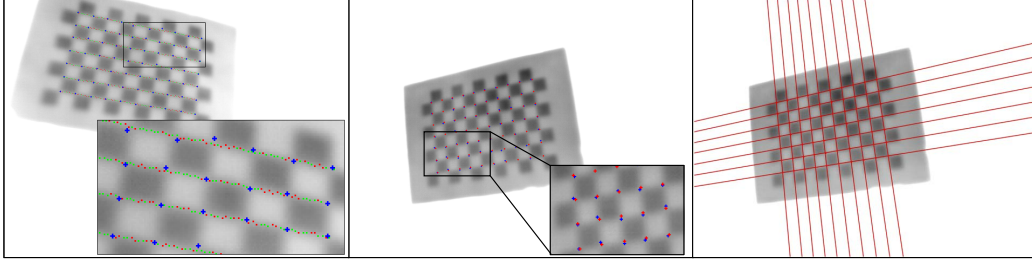


Figure 3.4: **Initial detection and refining the points.** Refined corner points should lie along straight lines in the left image. The center image shows the refinement of the points. Blue points are the points detected by OpenCV, red points are the refined points, and the right image shows the lines drawn connecting the corners.

board pattern should be detected such that each point lying on the same row (respectively column) should be aligned. The corner points are refined using line intersection optimization inspired by the method proposed by De la Escalera and Armingol [13]. This method performs the least square fitting of the lines from the “inaccurate” detected corners instead of using the Hough transform on the image.

Naturally, these first estimated lines are inaccurate due to the blur on the thermal-infrared images, see Figure 3.4 (left). However, they can be considered an initial guess for the first step of this refinement process. These lines are transformed to fit the neighboring maximum gradient of the image intensity, computed only in the orthogonal direction of the lines. In other words, the lines are transformed to lie in the middle of the blur between the black and the white cells of the chessboard. Since the maximum gradient point set may include outliers, the line fitting is performed with RANSAC

line fitting. Finally, the corner points are estimated based on the intersection between each row and column.

(5) Corner Refinement Based on Geometric Constraints

First, let $\mathcal{P}$ denote the set of corners initially detected on the calibration board and refined in the previous section. The next step of the proposed refinement involves a non-linear process. Indeed, the proposed method also optimizes the corners' position according to both the cross-ratio extracted from each chessboard cell and the vanishing points of the chessboard rows and columns, respectively.

Cross Ratio. Corner points are placed at constant intervals on the chessboard and thus should satisfy some constraints of cross-ratio. The cross-ratio is a perspective invariant defined in the invariance theory of cross-ratio [95, 28]. If the points A , B , C , and D are collinear in space, the four corresponding projected image points A' , B' , C' , and D' are also collinear. The cross-ratio of two sets of any such consecutive four points from any row (or column) of the chessboard are identical and satisfies the following equation:

$$\hat{\rho} = \frac{AC}{CB} / \frac{AD}{DB} = \frac{A'C'}{C'B'} / \frac{A'D'}{D'B'}. \quad (3.1)$$

In practice, it is possible to measure how the detected points satisfy this cross-ratio constraint by a cost function $C_{\text{cross}}(\mathcal{P})$ inspired by a method proposed

by Ricolfe-Viala et al. [61]. This cross-ratio cost is equally divided into two costs dedicated correspondingly to rows and columns constrains:

$$C_{\text{cross}}(\mathcal{P}) = C_{\text{cross}}^{\text{rows}}(\mathcal{P}) + C_{\text{cross}}^{\text{cols}}(\mathcal{P}),$$

where $C_{\text{cross}}^{\text{rows}}(\mathcal{P})$ and $C_{\text{cross}}^{\text{cols}}(\mathcal{P})$ are computed in the same way. For brevity, let us explain only the computation of $C_{\text{cross}}^{\text{rows}}(\mathcal{P})$. According to the cross-ratio constraint, each four successive points of a row should lead to the theoretical cross-ratio $\hat{\rho}$ defined by any consecutive points on the chessboard. Let Q_{rows} be the set of all combinations of four consecutive points $q = A, B, C, D$ of $\mathcal{P}$, extracted only from the rows of the calibration board. The cross-ratio $\rho(q_i)$ of each $q_i \in Q_{\text{rows}}$ is defined as:

$$\rho(q_i) = \frac{A_i C_i}{C_i B_i} / \frac{A_i D_i}{D_i B_i}.$$

Then, the cross-ratio cost function dedicated to the row points can be defined as:

$$C_{\text{cross}}^{\text{rows}}(\mathcal{P}) = \frac{1}{\text{card}(Q_{\text{rows}})} \sum_{q_i \in Q_{\text{rows}}} \left(1 - \frac{\rho(q_i)}{\hat{\rho}} \right) \quad . \quad (3.2)$$

The same procedure holds for Q_{cols} with:

$$C_{\text{cross}}^{\text{cols}}(\mathcal{P}) = \frac{1}{\text{card}(Q_{\text{cols}})} \sum_{q_i \in Q_{\text{cols}}} \left(1 - \frac{\rho(q_i)}{\hat{\rho}} \right) \quad . \quad (3.3)$$

Vanishing Points. As for the cross-ratio cost function, the vanishing point cost function is partitioned into two costs since a calibration board contains two main vanishing points:

$$C_{\text{vanish}}(\mathcal{P}) = C_{\text{vanish}}^{\text{rows}}(\mathcal{P}) + C_{\text{vanish}}^{\text{cols}}(\mathcal{P}).$$

Here, the chessboard row vanishing point is considered. The cost function procedure starts with estimating the vanishing point (x_v, y_v) computed as the least square intersection of all lines extending each row. Then, the vanishing point cost $C_{\text{vanish}}^{\text{rows}}(\mathcal{P})$ is defined as the average distance from this vanishing point to these lines. Since each of those lines should pass through the vanishing point, the average distance from the lines to the vanishing point should tend to zero if the corner points are accurately positioned. More specifically, let $\mathbf{r}_i = (a_i, b_i, c_i)^\top$ be the Hessian form of the i^{st} row and $\mathcal{V}$ the set of rows of $\mathcal{P}$, then the cost to minimize can be expressed as:

$$C_{\text{vanish}}(\mathcal{P}) = \frac{1}{\text{card}(\mathcal{V})} \sum_{\mathbf{r}_i \in \mathcal{V}} \frac{|a_i x_v + b_i y_v + c_i|}{\sqrt{a_i^2 + b_i^2}}. \quad (3.4)$$

The denominator transforms the line equations in their normalized Hessian form. Moreover, note that this formulation is not used when the calibration board is orthogonal to the principal ray of the camera since, in this situation, the vanishing points would lie at infinity, which is incompatible with our vanishing point scoring method. Figure 3.5 depicts both the cross-ratio and the vanishing points' constraints.

Data Fidelity Term. Using only the two first terms would lead to a set of points perfectly aligned and with the perfect distances from one to the next, but not necessarily consistent with the calibration board since the refined points could freely move anywhere. Thus, to maintain a certain consistency between the points and the calibration board, a data fidelity term $C_{\text{dist}}(\mathcal{P})$ is added. This cost simply defines the ℓ_2 distance $d(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ from any original point $\hat{\mathbf{x}}_i$ computed in the earlier process to its corresponding refined points

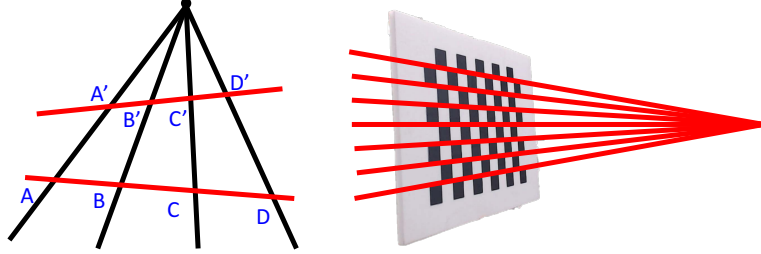


Figure 3.5: **Constraints of cost function.** The left figure describes the cross-ratio constraint satisfied by Equation (3.1). The right figure shows the vanishing point of the calibration board.

$\mathbf{x}_i \in \mathcal{P}$:

$$C_{\text{dist}}(\mathcal{P}) = \frac{1}{\text{card}(\mathcal{P})} \sum_{\mathbf{x} \in \mathcal{P}} d(\mathbf{x}_i, \hat{\mathbf{x}}_i). \quad (3.5)$$

Global Cost Function. The final cost function to optimize the chessboard corner position estimation can be described as the weighted sum of the three first cost functions:

$$C(\mathcal{P}) = \arg \min_{\mathcal{P}} (C_{\text{vanish}}(\mathcal{P}) + \alpha C_{\text{cross}}(\mathcal{P}) + \beta C_{\text{dist}}(\mathcal{P})), \quad (3.6)$$

where α and β are given parameters. Once the corners are refined by minimizing the weighted sum of Equation (3.6), these refined points are used to calibrate the camera parameters using Zhang's method [96].

(6) Deciding the Weights α and β

The weights α and β of Equation (3.6) were decided experimentally. A virtual chessboard comprising a set of points regularly arranged was generated and transformed using a random but realistic perspective transformation (homography). Then, Gaussian noise was added to these points. For a large set of values for α and β , many noised virtual chessboards were generated and this algorithm was run. The pair (α, β) that reconstructed the best the original virtual chessboard was selected on average over all virtual noised chessboards.

3.3.2 Relative Scale Between the Point Cloud

Any 3D models generated by a monocular RGB camera are built and defined at an unknown scale. Since the thermal-infrared images are back-projected on the 3D model obtained by DSO, this unknown scale should be estimated. In practice, this scale is also the scale that relates to a depth map generated from the RGB image to the corresponding depth map generated from the thermal-infrared images. This chapter defines how to estimate the scale of the DSO point cloud in the stereo rig coordinate system and how to back-project the thermal-infrared images on this point cloud.

DSO divides frames into two types: keyframes for which a depth map is computed, and the other frames just used to refine the depth map generated by the keyframes. These depth maps are used to superimpose thermal

information on the 3D structure. For a given keyframe, the depth value of each pixel $\mathbf{x}$ refers to a 3D point that can be projected on the infrared thermal image on $\mathbf{x}'$. This projected point $\mathbf{x}'$ corresponds to a thermal value that can be associated with the pixel $\mathbf{x}$ in the RGB image.

More precisely, let I_{rgb} express an RGB image and I_{ir} a thermal-infrared image. The depth maps of DSO are sparse so that some pixels obtain values and others do not. Image coordinates are denoted by $\mathbf{x} = (x, y, 1)^\top$. A general back-projection function [28] to convert the RGB image coordinate to a 3D coordinate according to the depth d in the DSO depth map is given by:

$$\mathbf{X}(d) = \begin{pmatrix} (\mathbf{K}\mathbf{R})^{-1}(d\mathbf{x} - \mathbf{K}\mathbf{t}) \\ 1 \end{pmatrix}. \quad (3.7)$$

Considering that $\mathbf{P}_{rgb}$ defines the coordinate system of the stereo rig, $\mathbf{P}_{rgb} = \mathbf{K}_{rgb}[\mathbf{Id}|\mathbf{0}]$ and $\mathbf{P}_{ir} = \mathbf{K}_{ir}[\mathbf{R}_{ir}|\mathbf{t}_{ir}]$, where $[\mathbf{R}_{ir}|\mathbf{t}_{ir}]$ is the relative pose between the two cameras, usually with the arbitrary constraint $\|\mathbf{t}_{ir}\|^2 = 1$. Since it is a back projection from $\mathbf{P}_{rgb} = \mathbf{K}_{rgb}[\mathbf{Id}|\mathbf{0}]$, the back-projection function simplifies to :

$$\mathbf{P}'_{rgb}(\mathbf{x}, d) = \begin{pmatrix} d\mathbf{K}_{rgb}^{-1}\mathbf{x} \\ 1 \end{pmatrix}, \quad (3.8)$$

where $\mathbf{K}_{rgb}$ is the intrinsic matrix of the RGB camera. Then, the following equation is used to project the DSO depth map onto the thermal-infrared image to read its thermal intensity value:

$$\mathbf{x}' = \mathbf{M}_{rgb.ir}\mathbf{P}'_{rgb}(\mathbf{x}, d), \quad (3.9)$$

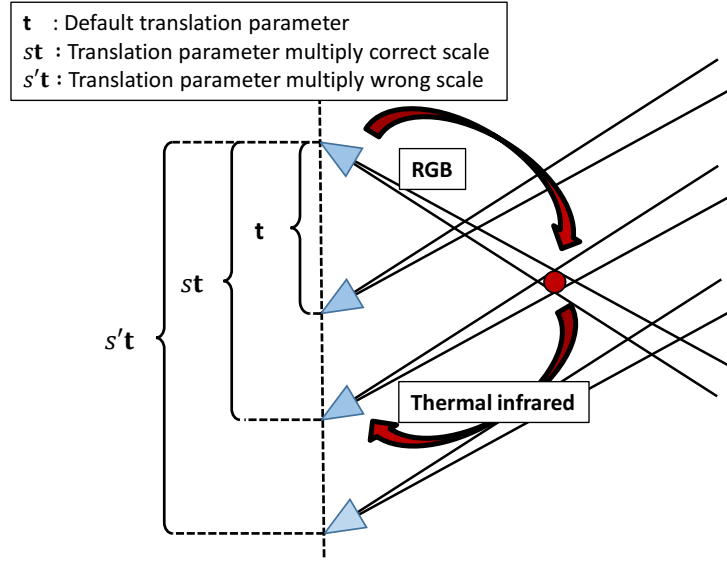


Figure 3.6: **Variation of back-projection by different scales.** The red circle represents a 3D object of the scene with a certain arbitrary scale. The translation $\mathbf{t}$ between the two cameras initially subject to $\|\mathbf{t}_{ir}\|^2 = 1$ can be scaled with the fine-scale factor s to fit the 3D reconstruction from the first camera.

such that the pixels of the RGB image can be associated with the corresponding pixels in the thermal-infrared image:

$$I_{rgb}(\mathbf{x}) \leftrightarrow I_{ir}(\mathbf{x}'). \quad (3.10)$$

However, since the relative pose of the thermal-infrared camera in the RGB camera coordinate system must be defined taking the scale into account, the relative pose transformation $\mathbf{M}_{rgb_ir}$ from $\mathbf{P}_{rgb}$ at scale s is defined as follows:

$$\mathbf{M}_{rgb_ir} = \mathbf{K}_{ir} [\mathbf{R}_{ir} \mid s\mathbf{t}_{ir}]. \quad (3.11)$$

Here, $\mathbf{t}_{ir}$ is a translation vector corresponding to the position of the thermal-

infrared camera in the stereo rig coordinate system. Since $\mathbf{t}_{ir}$ is defined at an arbitrary scale (here $\|\mathbf{t}_{ir}\|^2 = 1$), it can also be defined at any other scale with the scale factor s . Moreover, the 3D structure and thus the depth values provided by any monocular RGB SLAM are also defined at an unknown scale; thus, the DSO point cloud cannot be projected to thermal-infrared camera images unless a common scale factor can be determined between the 3D structure and stereo rig coordinate systems. In this process, the 3D structure scale factor can be fixed to its default value and the scale factor s of the stereo rig that fits the best to the 3D structure is explored, as depicted in Figure 3.6.

Figure 3.7 shows projection results from the DSO point cloud to thermal-infrared images using different scales s in Equation (3.11), varying the parameter s . In this method, the scale of the point cloud generated by a monocular RGB camera is estimated, and the point cloud is accurately projected onto thermal-infrared images using the scale.

Thermal and RGB values have different modalities, and thus cannot be compared directly. The proposed method compares RGB and thermal-infrared images with depth values, since depth is a common modality. In this method, a depth map is generated from both the RGB and thermal-infrared images using a multi-view stereo (MVS) algorithm [54, 56]. A depth map is generated using several images near focus frames and the trajectory of the camera obtained from DSO. A patch for each pixel is generated, then the score is obtained by comparing the patch and other frame patches using

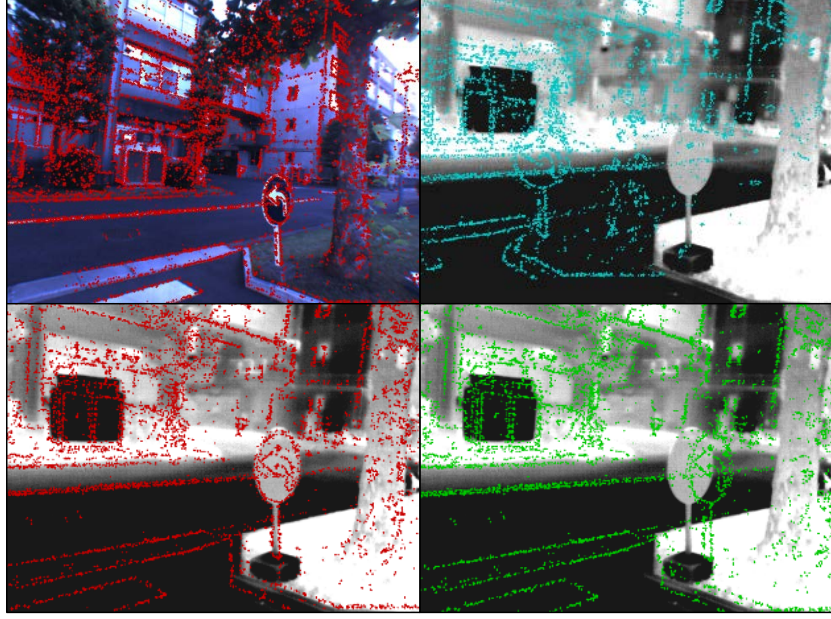


Figure 3.7: **Result of back-projection to thermal-infrared images using different scales.** The top left is a reference RGB camera image. The other images are generated by back-projection of the DSO point cloud using different scales. The bottom left image shows the accurate scale.

zero-mean normalized cross correlation (ZNCC). The depth value d_i of pixel i is determined from all possible depth values D using the following equation:

$$d_i = \arg \min_{d \in D} C(i, d). \quad (3.12)$$

Equation (3.13) describes the score of the ZNCC where C expresses the ZNCC score of the patch centered in pixel i with a certain depth value d .

$$C(i, d) = - \sum_{j \in I_p} \frac{(I_p(j) - \overline{I_p}) (I'_p(j) - \overline{I'_p})}{\sigma(I_p) \sigma(I'_p)}. \quad (3.13)$$

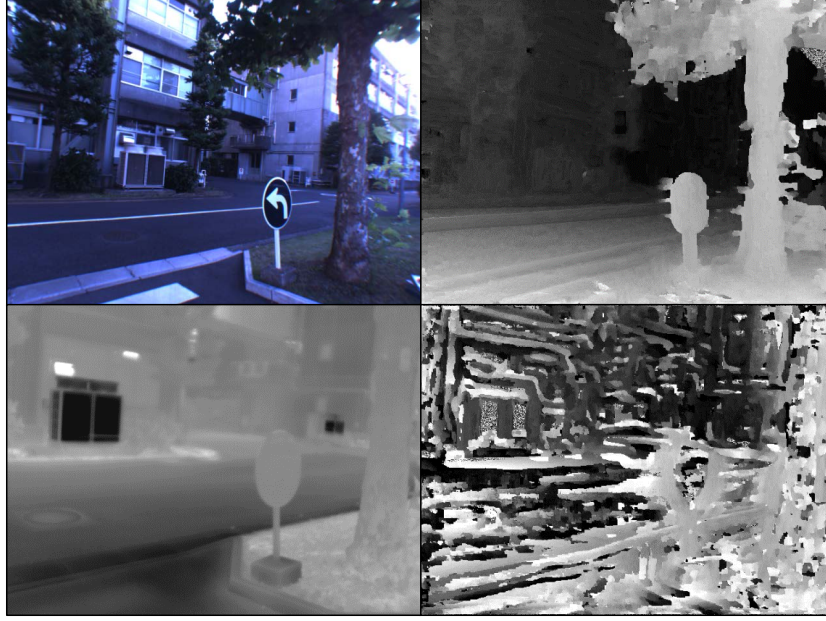


Figure 3.8: **Generating a depth map using a multi-view stereo (MVS) algorithm.** The top left shows a reference RGB camera image, and the bottom left shows a reference thermal-infrared camera image. The right images show the result of MVS using each image.

This score is calculated by comparing a square window I_p in depth image I and a square window I'_p in depth image I' . $I_p(\mathbf{j})$ and $I'_p(\mathbf{j})$ describe the intensity value at pixel $\mathbf{j}$. Finally, $\overline{I_p}$ and $\sigma(I_p)$ refer to the mean and the standard deviation of the window I_p . Figure 3.8 shows some results of depth maps generated from both RGB and thermal-infrared images.

The depth map generated from the RGB image sequence is successively translated to thermal-infrared camera coordinates by Equation (3.11) by iterating on $\mathbf{t}$. The translated RGB depth map and the thermal-infrared depth map are compared using the mutual information defined in [71, 84].

This mutual information is often used to compare images of different modalities. In this case, this modality variety can be expressed as the difference in accuracy between the RGB and thermal camera depth map computation. Indeed, thermal-infrared depth maps are always significantly worse than the depth maps built from RGB images. The translation $\mathbf{t}$ leading to the best depth map overlap defines the scale s . Equation (3.14) describes the mutual information score:

$$MI(I_{rgb}, I_{ir}) = \eta \left(H(I_{rgb}) + H(I_{ir}) - H(I_{rgb}, I_{ir}) \right). \quad (3.14)$$

In Equation (3.14), I_{rgb} is the patch converted by Equation (3.10) using given scale s , and η describes the ratio of appearance of the RGB depth map on the thermal-infrared camera coordinate after converting. $H(I)$ describes the appearance ratio of depth i on the depth image I , and $H(I, I')$ describes the two dimensional appearance ratio of depth i on the depth image I and depth j on the depth image I' :

$$H(I) = - \sum_{i \in \Omega} p_I(i) \log(p_I(i)), \quad (3.15)$$

$$H(I, I') = - \sum_{i \in \Omega} \sum_{j \in \Omega} p_{II'}(i, j) \log(p_{II'}(i, j)), \quad (3.16)$$

where $i, j \in \Omega$ represents pixels i and j in the two-dimensional plane Ω of the image. Figure 3.9 shows the variation of the mutual information scores with different scales s . The green curve is the average score for each scale. The optimal scale can be estimated uniquely as a global maximum from this

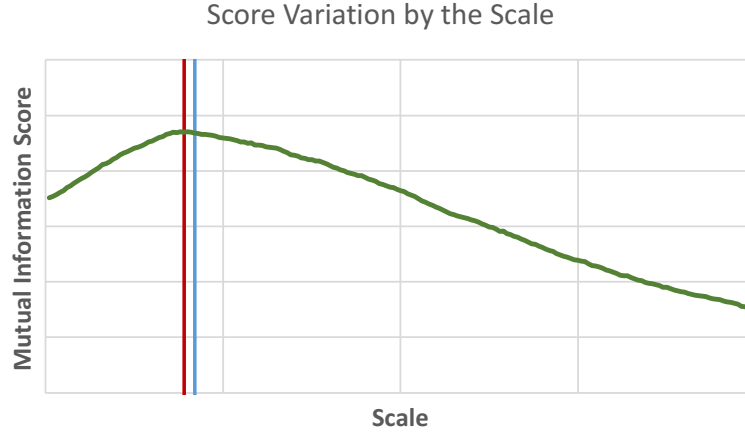


Figure 3.9: **Variation of the MI score with scale.** The green curve is the average score for each scale. Some samples are used to estimate the scale, and the average of the score is obtained. The red vertical line is the place where the score is maximum, and the blue vertical line is the ground truth obtained by the experiment.

graph. This scale is a valuable information to correctly superimpose the temperature data on the point cloud.

3.3.3 Generating RGB-T Image from the Point Cloud

At that stage, a point cloud computed by the RGB camera and the correct scale between this point cloud and the stereo rig referential is obtained. Thus, it is possible to project associating with each point an RGB value as well as a thermal value.

In practice, the point cloud generated from DSO is sparse; thus, the

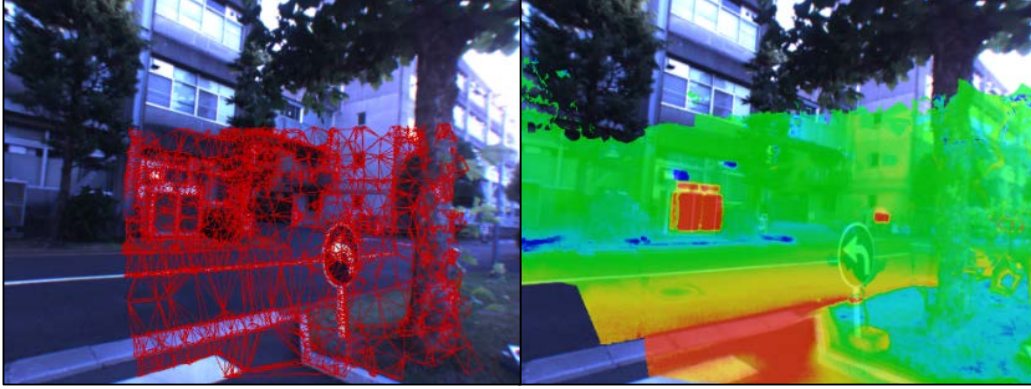


Figure 3.10: **RGB-T image.** The left shows the result of overlaying a 2D mesh on the image, and the right shows the result of generating an RGB-T image.

thermal information may have to be interpolated, where the point cloud is not dense enough. Thus, the 3D point cloud is first back-projected on the targeted RGB image using the camera pose provided by DSO. Second, triangle meshes are generated from the projected point cloud using Delaunay triangle division [36, 40]. Then, each triangle vertex is back-projected on the thermal image using Equation (3.9) to compute the right texture coordinates of each triangle in the thermal image. The final rendering involves superimposing the dense thermal textured mesh back-projected onto the corresponding RGB image. Figure 3.10 shows an example of an RGB-T image built with this process.



Figure 3.11: **Camera rig.** Left: thermal-infrared camera. Right: RGB camera.

3.4 Experiment

3.4.1 Experimental Setups

In these experiments, a Flea3 in Point Grey monocular RGB camera and a PI640 Optris thermal-infrared camera were mounted on a hand-held stereo rig, as depicted in Figure 3.11.

3.4.2 Calibration Refinement

The calibration refinement process was evaluated as follows. At first, the cameras were calibrated with the non-optimized chessboard corners detected by OpenCV, then the reprojection error on each camera is computed; i.e. the average distance from each detected corner in the image and the projected

Table 3.1: **Calibration RMS in pixel.**

	Initial	Refined
Thermal	1.498	0.631
Stereo	1.235	0.759
RGB	0.172	-

corner with the projection matrix. Second, this process was repeated with the refined chessboard corners. Table 3.1 shows these reprojection errors. Although the reprojection error of the thermal camera is still not as accurate as for the RGB camera, the calibration reprojection error of a thermal camera is noticeably improved, roughly by a factor of two, in the reprojection error of the thermal camera calibration.

3.4.3 Thermal Maps and RGB-T Images

The final RGB-T images are computed by superimposing thermal information on the RGB-images. The thermal data are rendered with a color map with red, green, and blue gradation, where red indicates high temperature and blue indicates lower temperature. For this experiment, thermal maps from both outdoor and indoor sequences were generated. Figure 3.12 shows the resulting generated thermal maps.

In the outdoor sequence, the building equipment and the asphalt exposed to the sunlight got a high temperature, and thus appeared in red. The cars parked in the shadows of buildings had a lower temperature and were

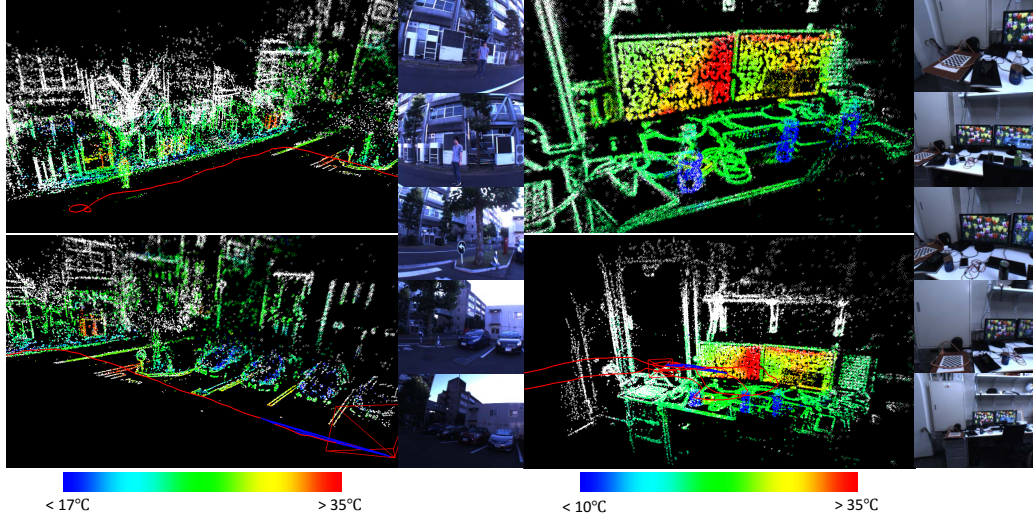


Figure 3.12: **Thermal maps.** Left: outdoor sequence generated by 1,330 pairs of RGB and thermal-infrared images. Right: indoor sequence with 1,264 pairs of RGB and thermal-infrared images.

rendered in blue. The indoor sequence shows the high temperature of the displays in red and the low temperature of the beverage rendered in blue.

RGB-T images were also generated from various scenes. Fig. 3.13 shows the resulting RGB-T images and reference RGB images.

3.4.4 Evaluation of the Estimated Scale

This chapter describes the evaluation of our scale estimation method and its accuracy. Figure 3.14 shows the environment used to evaluate the scale. In this experiment, a calibration board of known length was reconstructed. Using the scale estimated by this method, the estimated size of the calibration

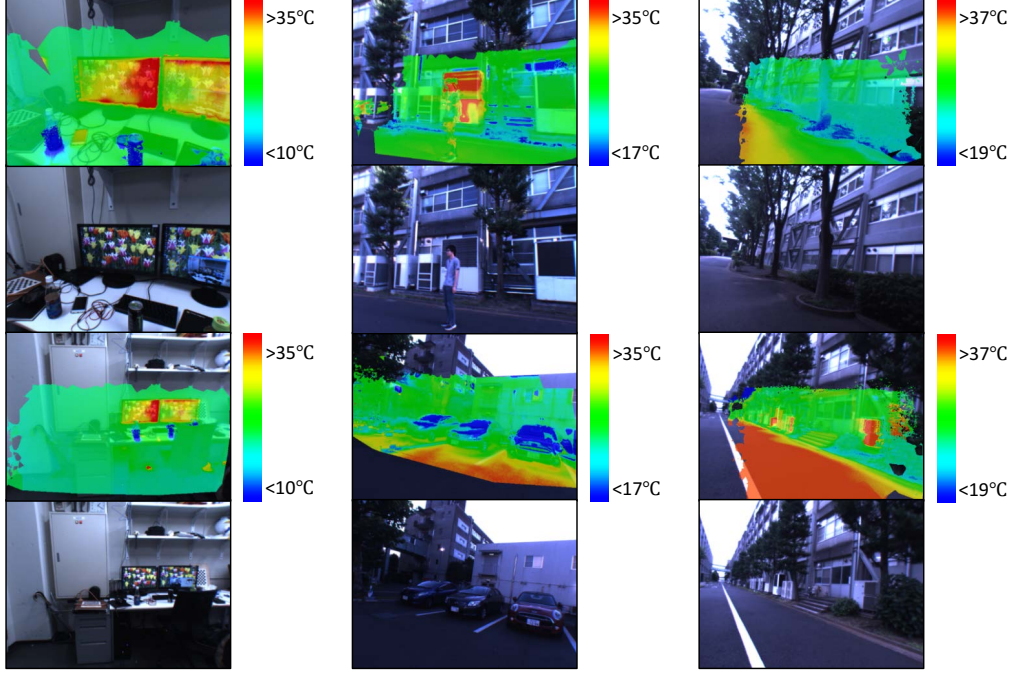


Figure 3.13: **RGB-T images from various scenes.** These images are generated from three different sequences.

board was compared with the actual size.

First, a thermal map was reconstructed as usual, as shown on the left of Figure 3.14, and then the calibration board was reconstructed in the right part of the figure. Second, the point cloud was reprojected to the image capturing the calibration board, and the points on the corners of each cell of the calibration board were picked up. The points were very noisy, therefore the outliers that were not on the plane of the calibration board were removed by RANSAC, and an estimated length $\hat{L}$ of each cell side was computed:

Table 3.2: Mean and standard deviation of the relative error between the estimated scale and the ground truth.

	Average	Standard deviation
Scene1	-0.1026	0.0048
Scene2	0.0719	0.0386

$$\hat{L} = s\|\mathbf{t}\|. \quad (3.17)$$

Next, the average length of the cell on the scale of the DSO point cloud was estimated. Using Equation (3.17), the scale was converted to the actual size. The mean relative error against the actual size L was calculated using Equation (3.18):

$$\epsilon = \frac{\hat{L} - L}{L}. \quad (3.18)$$

Considering the randomness of RANSAC, the process was repeated 100 times. Table 3.2 tabulates the mean and the standard deviation of the relative error. The indoor and outdoor scenes were evaluated in two sequences.

3.5 Summary

This chapter visualizes temperature more effectively by generating a thermal map and RGB-T images using 3D structures obtained from DSO. Furthermore, the results of this method are demonstrated using indoor and outdoor

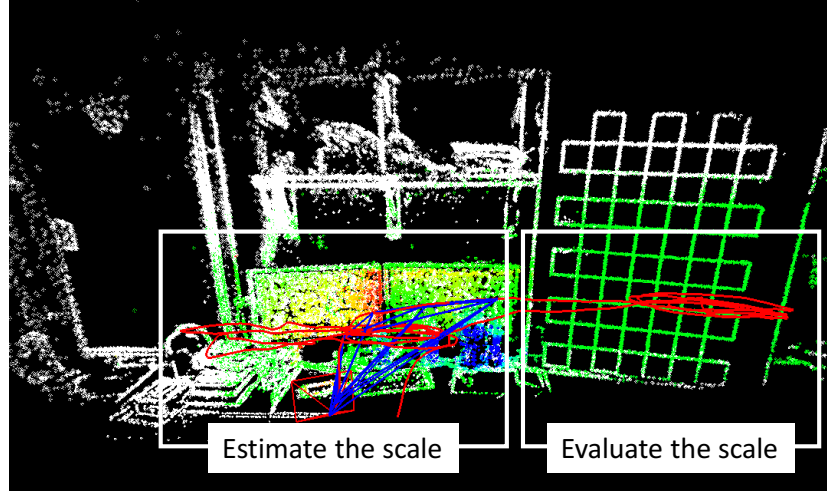


Figure 3.14: **Evaluation of the estimated scale.** The reconstructed environment includes a calibration board. First, the scale of the point cloud is estimated. Second, the estimated scale and the actual scale are compared using the side of the calibration board.

scenes.

The proposed method, first, calibrates an RGB and a thermal-infrared camera using a calibration board that can be detected by the thermal-infrared camera. Then, temperature information is superimposed onto the generated 3D structure using the extrinsic parameter between both cameras. Since a scale in 3D space is needed, this method uses the depth images obtained by MVS to estimate the scale. Moreover, an RGB-T image is generated using Delaunay triangulation to generate an RGB-T image that can superimpose temperature on an RGB image that yields the insufficient number of 3D points from the DSO to create a triangular mesh. Thermal maps and RGB-T images can be generated using the proposed method.

Limitations

If the distance between the light axes of the cameras is small, the method may be unnecessary. For example, in the case of using cameras whose light axes are even closer than those used in this experiment, the error in visualization may not be a problem even without using this method. However, this method can be effective when more accurate results are required or when results are required for objects close to the camera. In this method, the camera is aligned offline using the 3D structure obtained from the captured images. This is an implementation limitation, and depending on the implementation, this part could be done online.

Chapter 4

Object Pose Estimation toward AR Applications

4.1 Introduction

In AR technology, recognizing each object and estimating the relative pose from the camera to the object is as important as estimating the pose of the camera in the environment. It is necessary to estimate the pose of an object from a camera in a dynamic environment where the pose of each object changes in the world coordinate system. By tracking the relative pose from the camera to the dynamic object, it can be used to add AR effects to dynamic objects and improve the accuracy of camera pose estimation in dynamic environments. In this chapter, dynamic object tracking is applied to assembly and disassembly procedures as one of the use cases of relative pose estimation of objects in dynamic environments.

Traditionally, assembly and disassembly procedures are communicated

using textual descriptions that are often combined with a series of images for illustrating motions [46]. As complex manipulations are difficult to visualize in a series of images, illustrations are often complemented with video instructions [48]. Such video tutorials are widely available on the Internet and an abundant source of information.

Following video tutorials may be challenging, when the user's current viewpoint is not aligned with the one in the video [9]. Interactive 3D tutorials utilize object tracking to overcome the issue of mismatching viewpoints [25] by presenting 3D instructions that are aligned with the user's current view of the object. This enables users to perceive both the real-world object and the associated instructions, from a single viewpoint. However, creating 3D animations for this kind of interactive tutorial is often difficult and, thus, quickly becomes time-consuming and expensive.

This chapter proposes a novel approach for retargeting assembly video tutorials, which allows us to obtain a complete and detailed instructional 3D tutorial that also encompasses small and occluded parts. The extracted instructions are presented in an AR magic mirror setup [19], as shown in Figure 4.1. The method of instruction extraction is based on the principle of assembly-by-disassembly [14], which is commonly used in assembly planning [34]. Assembly-by-disassembly determines the assembly steps by disassembling the object of interest and reverting the calculated sequence. This method follows a similar approach and parses the input video in reverse, starting with the completely assembled object and analyzing its disassembly.

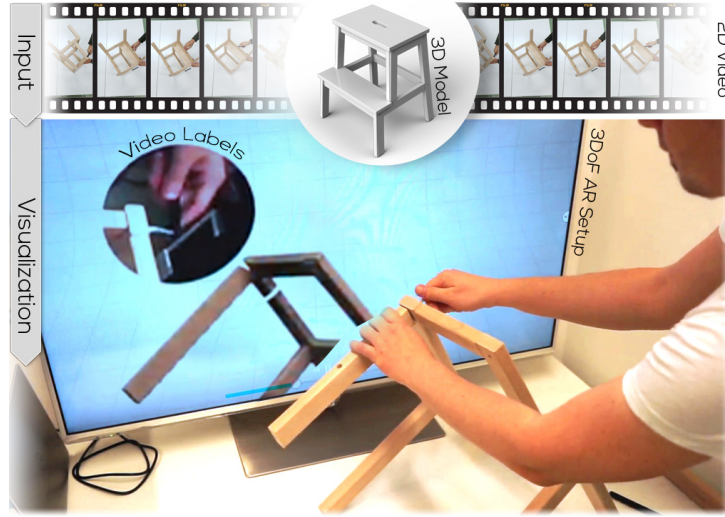


Figure 4.1: **Video annotated assembly tutorial.** The proposed system converts an assembly video tutorial into a 3D animation of a video textured 3D model and annotates it with the corresponding video sequence. These video labels provide missing details, such as tools and complex motions that are required to perform the actions.

Therefore, by reverting the determined removal sequence, the actual assembly steps which are used to generate the proposed instruction visualization are retrieved. This simple but effective idea allows tracking the removal of parts instead of having to detect the presence of new parts, which provides significant benefits to object detection and tracking. A major advantage is that direct observations are used in the input video for object tracking, whereas previous approaches had to rely on object detection using synthetic 3D renderings [86].

The proposed approach extracts complete assembly instructions including manipulations of visually undetectable parts by tracking the target objects and presents them as animated 3D instructions. The proposed

method achieves this by combining 2D segmentation-based 3D object tracking with queries of an assembly graph [35] of the target object that encodes blocking relationships between parts and allows us to identify small or occluded parts in the video. The presentation of 3D instructions is enhanced by annotating the object with video sequences from the input video tutorial to preserve details, such as complex assembly motions, tool usage or non-rigid object deformations. To make interactive 3D AR tutorials widely available to users, the proposed method also aims for a practical setup that uses commonly available hardware such as laptops and TVs for presenting assembly instructions. In addition, a typical smartphone is utilized for tracking the assembly pose, to align the viewpoint shown in the AR mirror with that of the user.

The proposed method is the first to automatically generate comprehensible and complete 3D visualizations from video tutorials containing difficult to-detect parts and complex object manipulations. In summary, the proposed method makes the following contributions:

- A novel approach for extracting complete 3D assembly instructions from a video tutorial utilizing the 3D object tracking and the object's assembly graph.
- A novel visualization technique for assembly instructions combining 3D animations with video information to encompass all information from input video tutorials. Video annotations provide information on otherwise undetectable manipulations.

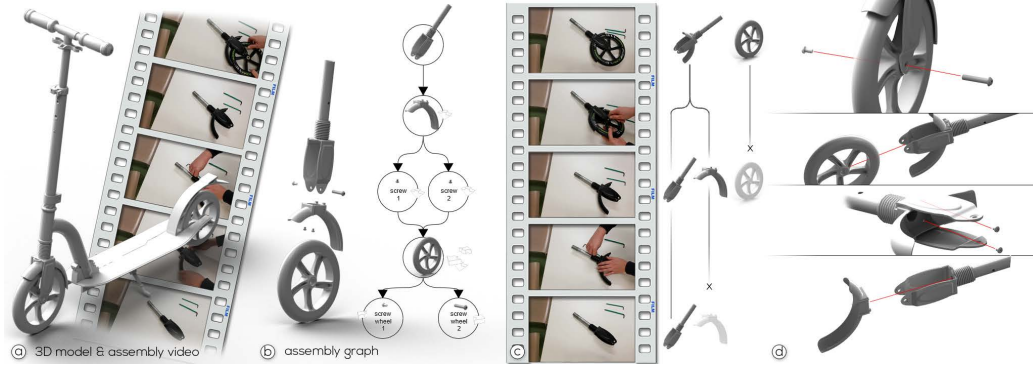


Figure 4.2: **Animating the 3D assembly model.** (a) Given a 3D model of the object of interest, (b) its assembly graph is constructed, (c) and object parts which are large enough to be detected are identified. (d) After detecting the removal of large parts, the blocking information is used in the assembly graph to infer the removal of small parts, which cannot be detected otherwise. The detected order of removal is used to generate a 3D animation that encodes the removal actions as depicted in the video.

- A practical approach for an AR tutorial setup using commonly available hardware by utilizing a smartphone for object tracking and a monitor as an AR mirror.
- The completeness of the extraction and visualization approach by performing a user study comparing against typical video tutorials.

This approach relies on model-based tracking of parts in a video frame. Hence, it relies on video data of certain quality, e.g. with large enough depth of field and little motion blur.

4.2 Related Works

Due to their spatial anchoring, assembly instructions are ideally suited for AR applications. In comparison to traditional media, AR systems have been shown to reduce mental workload, improve task performance [76] and localization [30]. Users following AR instructions not only benefit from an improved spatial understanding, but also from comprehensible instruction visualization [33].

Advances in 3D pose estimation and tracking [58, 90] have opened up new possibilities for scene understanding, and enabled the design of sophisticated AR assembly tutorial systems. With this motivation in mind, this method introduces an end-to-end approach to automatically generate AR tutorials from input videos and apply them to real-world assembly applications.

AR assembly tutorials are traditionally realized using rule-based systems [18] and scripting languages [11, 39]. However, assembly instructions can also be inferred from a 3D CAD model directly [47, 44, 2, 35], for instance, by calculating blocking relationships and creating an assembly graph containing information about the assembly sequence of all parts. This method integrates assembly graphs into the proposed system to support the extraction of assembly sequences from input videos and to identify small or occluded parts.

More recent approaches extract assembly instructions from existing

sources, for instance, by analyzing written manuals [48, 69, 68] or video tutorials that demonstrate the procedure. Gupta et al. [25] investigated the creation of assembly tutorials for Duplo blocks by allowing a user to demonstrate assembly instructions via a semi-automatic authoring system. The proposed approach is related to this category of demonstration-based authoring and uses video tutorials as input. Using video as a source expands the applicability of the proposed approach due to the wide availability of such content in online platforms. In a recent work, Mohr et al. [49] presented a similar system that extracts AR tutorials for tools with surface contact. Similarly, the proposed method also extracts instructions from unmodified input videos, but this section broadens the scope to support general instructions for the assembly of complex objects.

This system has the goal to reduce extraneous cognitive load by automatically creating effective assembly instructions from input videos. Therefore, the design principles suggested by cognitive load theory (CLT) research is adopted to create effective assembly instructions. Instructions are segmented into steps, so that, in each step, only a single major part and its fasteners are attached. Using segmented 3D animations allows users to follow instructions at their own pace. Free viewpoint changes are allowed to avoid occlusions, improve part visibility and enable users to align the viewpoint of the 3D instructions with their own viewpoint. To improve realism, inpainting techniques apply the texture of the input video to the CAD parts. Additional details are provided by registering the original instructional videos

in 3D. This method visualizes the video viewpoint relative to the 3D assembly using a 3D view frustum indicating the view direction of the recorded video. Such viewpoint visualizations support the spatial orientation of users [77] and communicate viewpoint locations [72].

4.3 Method

4.3.1 Determining the Assembly Sequence

This section provides an overview of the system for generating a free-viewpoint animation. Input to the proposed system is a video tutorial and a 3D model of the object of interest (Figure 4.2 (a)). The 3D model must be structured such that individually movable parts can be distinguished. The proposed system animates each of the parts following the assembly procedure that is demonstrated in the input video.

Step 1 - Computing the Assembly Graph. The assembly graph is used to support the detection of the removal of assembly parts in the video. To identify unblocked parts that can be removed, blocking relations are computed between all parts of the 3D model. The blocking information is stored in a directed acyclic assembly graph (Figure 4.2 (b)), which represents the parts of the object as nodes, and blocking relations between parts as edges. The assembly graph is computed using the approach presented by Kerbl et al. [35], which detects blocking relations by analyzing the 3D.

Step 2 - Extracting the Assembly Sequence. The assembly sequence is

computed from the input video by analyzing the video frames in reverse order, thus, observing the disassembly of the object and detecting the removal of object parts. Hence, inverting the observed disassembly sequence provides us with the actual shown assembly sequence. For each disassembly step, the assembly graph is used to identify all currently removable parts in the 3D model (Figure 4.2 (c)). Then, the six degrees of freedom (6DOF) pose of the object is computed in order to project all of its parts into the corresponding video frame. These projections are used to generate a 2D footprint of each part and to initialize a segmentation-based tracker. To detect a part's removal, the current removable parts are projected while tracking the assembly and the probability of each part's existence is calculated at the location of its expected 2D footprint. Once a part has been removed, the probability in the corresponding 2D footprint decreases significantly, providing a reliable measure for removal detection. When a part is detected as deleted, add it to the assembly sequence and remove it from the 3D model and assembly graph.

This process iterates, until the assembly graph is empty. However, since the proposed method relies on the 2D footprint based on projection, only the presence of a sufficiently large, object part that provides a detectable footprint can be monitored. Therefore, in each frame, the detectability of all detachable parts is tested by first measuring the size of the 2D footprint in image space. Undetectable parts are marked as potentially removed in the assembly graph, so that blocked, but detectable parts become removable.

Note that an undetectable part is added to the assembly sequence only after the removal of a larger part that it blocks is detected.

Since multiple undetectable parts may block a single detectable part, the correct removal order of the undetected part must be determined, as shown in the video. Their removal order is inferred by segmenting the user's hands and tools from the video and analyzing the positions relative to assembly parts, assuming they are removed sequentially using hands and tools.

Step 3 - Generating Step-by-Step Animations. Once a complete disassembly sequence has been computed, visual instructions are generated to communicate the assembly actions. Commonly used assembly visualizations such as motion lines and 3D animations [53] are generated by translating each part in the direction where no other parts block the removal. When multiple directions are possible, this method follows the approach of Li et al. [42] and selects the direction that allows the part to escape the assembly bounding box the fastest (Figure 4.2 (d)).

(1) Model Registration

This approach is based on the registration and tracking of an existing 3D model of the assembly in the video. This approach aims at a wide range of assembly tutorials. Therefore, it focuses on common RGB video formats. An image-based tracker [58] that tracks the 2D segmentation and the 6DOF pose of a 3D object simultaneously is employed.

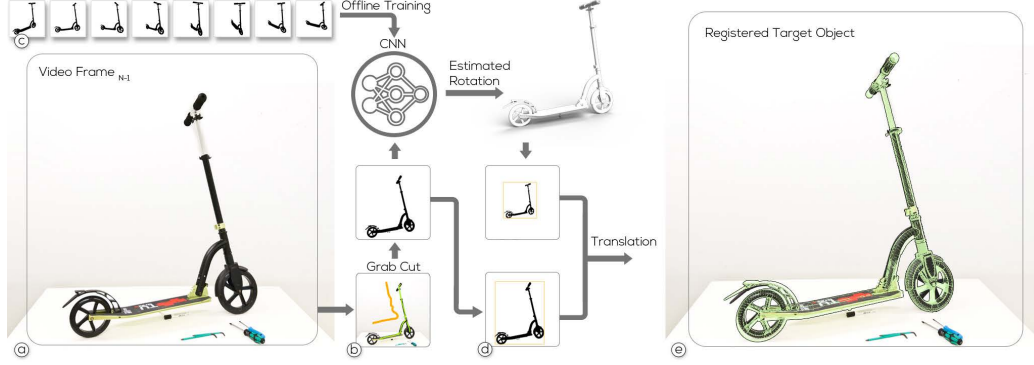


Figure 4.3: **Registration.** (a) The assembly is registered to the last frame of the input video. (b) This approach uses a segmentation of the object, (c) and a neural network to derive the camera orientation from the segmentation. (d) Using the estimated orientation the object of interest is rendered, and subsequently detected its bounding rectangle. The bounding rectangle of the segmentation is also computed in the input frame, and the distance of the camera is derived from the ratio of the sizes of the rectangle. (e) The resulting translation and rotation are employed to register the assembly.

Initially, using GrabCut [62], an object mask M_N of the fully assembled object $\mathbf{O}$ is generated at the last frame N of the input video I_N (Figure 4.3 (b)). Subsequently, the initial 6DOF pose of $\mathbf{O}$ is derived from the mask M_N . For pose estimation, a convolutional neural network (CNN) is trained with renderings of $\mathbf{O}$ (Figure 4.3 (c)). In order to remove misleading color information, both the training renderings and the runtime segmentation are converted to binary image masks. Note that this is similar to the approach of Wu et al. [90]. However, in contrast to Wu et al., the translation is estimated separately, using only the bounding rectangle of the mask M_N . The translation can be detected separately since it is possible to rely on the very precise mask M_N generated by GrabCut. Hence, the network is focused

on the estimation of the rotation only, which significantly reduces the size of the network and the amount of data required for training. This 3DOF pose was trained with approximately 10% of the training data required for state-of-the-art 6DOF pose estimation techniques [90].

Rotation. The training data of the network is generated by rendering the object’s 3D model from various camera positions on its bounding sphere (Figure 4.3 (c)). Each training image is cropped using the bounding rectangle that encloses the rendering and subsequently resizes it to 224×224 pixels to match the size of the CNN input structure. The separation of rotation and translation allows us to set up a small network. The proposed network consists of only five convolution-and-pooling layers with Rectified Linear Unit (ReLU) activation and one fully connected layer. The first convolution layer uses a 5×5 kernel, while other convolution layers use a 3×3 kernel size. After every convolution, a max-pooling operation down-samples the image to the half of its size. The loss function makes use of a standard mean squared error in quaternion space.

Translation. For the full pose, the translation that aligns the segmented object mask M_N with the mask M_R of the rendered object is calculated. M_R is generated by rendering $\mathbf{O}$, using the estimated rotation R_N and an initial distance d from the center of $\mathbf{O}$. An initial distance d is chosen so that the rendering contains the entire object. Both masks represent the projection of the object under the same rotation and thus show very similar shapes, while differing in size. To align both masks, the ratio of the longest edges of both

bounding rectangles is computed, and the translation is adjusted accordingly (Figure 4.3 (d)).

Evaluation. This pose estimator requires only a small set of images for training and generates results that are sufficiently accurate to initialize incremental tracking. This is demonstrated on the LINEMOD dataset [31], from which 10,000 images were used for training and 1,000 for testing. Ten objects were used from the dataset. Symmetric objects that make the proposed pose estimation ambiguous were excluded. The overlap of the ground truth masks and estimated masks of a 3D model was measured. The overlap and the standard deviation were $81\%(\pm 12)$, which is substantially larger than what has been reported for initializing image based pose tracking [58].

(2) Removable and Detectable Part Selection

After the 3D model of the object to the video is registered, monitoring its removable parts is started. Removable parts are identified by searching the assembly graph for nodes without outgoing edges. The resulting list contains all parts that can potentially be removed from the assembly at the current point in time. However, some parts contained in this list may not be detectable in the camera view due to their small size in image space. Therefore, the detectability of the parts is determined under observation by computing their footprint in image space. Parts are removed from the list which are too small to be detected (a part is considered too small if its footprint is smaller than 25×25 pixels). Afterwards, the list of removable

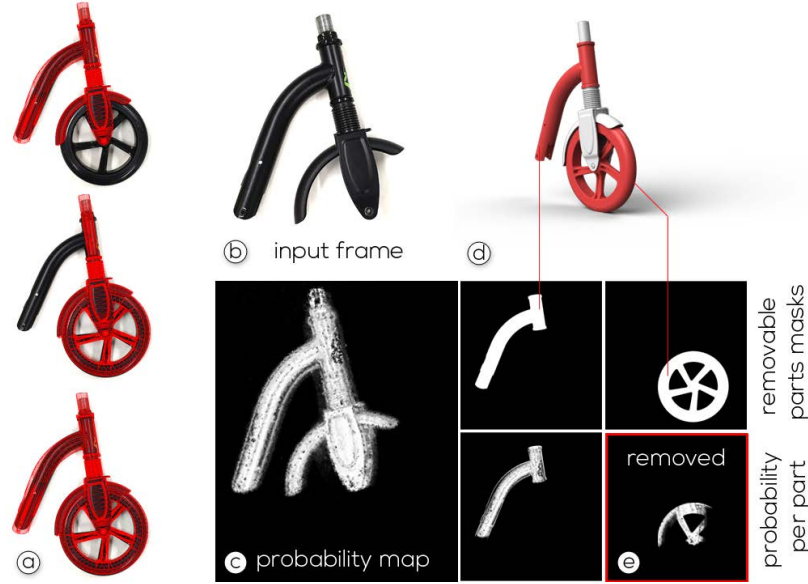


Figure 4.4: Tracking and removal detection. Masks are generated for a large object used for tracking by configuring each tracker with all but the removable part, in addition to the entire assembly. (a) The example shows the resulting three trackers for two removable parts. (b) From the input frame, (c) the probability of all tracker is computed and the one with the highest value is used, i.e. the one without the removed part, for updating the camera pose. (d) The camera pose is used to project removable parts (illustrated in red), (e) and the probability of their existence in corresponding footprints is computed.

parts is updated by searching for parts that have now become removable. This process is repeated, until no footprint in the list of removable parts is considered being too small. The size test of the part under consideration is repeated and they are re-categorized when their detectability state changes. The rest of this chapter discusses the tracking and detection of removable parts that have a sufficiently large footprint. Smaller parts are inferred from the removal detection of these larger trackable parts.

(3) Tracking

After the assembly is registered and its initial 6DOF pose is determined, the proposed system tracks the rigid 3D assembly using the approach of Prisacariu and Reid [58]. The methodology is a probabilistic framework for simultaneous region-based 2D segmentation and 2D-to-3D pose tracking using a known 3D model (here, a rigid 3D assembly). Given such a model, the relative pose from the camera to the object is estimated by maximizing the discrimination between the statistical foreground and background appearance models based on color histograms via direct optimization of the 3D pose parameters. However, since the observed object is disassembled in a video tutorial, its 3D shape changes over time. This will eventually lead to tracking failure. To improve tracking stability, the previously determined removable parts list is used. Multiple trackers are employed, one for each removable part, and the tracker is used with the highest confidence for updating the pose of the entire assembly.

Since the stability of each tracker improves with the number of pixels covered by the observed object, the aim is to maximize the footprint of the tracked parts. Therefore, the current assembly is tracked without a removable part, instead of tracking a single removable part in isolation. This approach is applied for each removable part of a frame, thus, running multiple object tracker instances in parallel, each ignoring a different removable part (Figure 4.4 (a)). One of these trackers will not be affected by the removal of a part, because the tracked assembly configuration matches the real

configuration after the part has been removed, thereby improving its tracking stability. To further improve stability, this method copes with hands occluding the assembly by performing hand and tool segmentation using the approach of Li et al. [41] and excluding such pixels from the input frame of the trackers.

Among all trackers, the one which is finally used for updating the pose of the entire assembly is identified by comparing their confidence. The confidence is derived from their energy terms, after pose optimization has been completed for the current frame. This method follows the formulation of the energy term by Prisacariu and Reid [58], which uses a level set embedding of the pose, and the likelihood models P_f and P_b . The likelihood models use color histograms, for calculating the probability of a pixel x to fall within the foreground region f and the background b . Note that the smaller the energy gets, the more confident the estimated pose becomes.

(4) Removal Detection

The removal of a part is inferred from the probability of its 2D projection to cover the part in the video frame. Therefore, each removable part O_k is projected to image space, using the current pose of the assembly, and generates the footprints $\Omega(O_k)$ (Figure 4.4 (d)). To compensate for hand occlusions, pixels from $\Omega(O_k)$ that intersect with the hand segmentation $\Omega(H)$ are removed, i.e., $\Omega'(O_k) \equiv \Omega(O_k) \setminus \Omega(H)$. A frame for a part O_k is skipped if more than n percent of the pixels in $\Omega(O_k)$ are occluded by

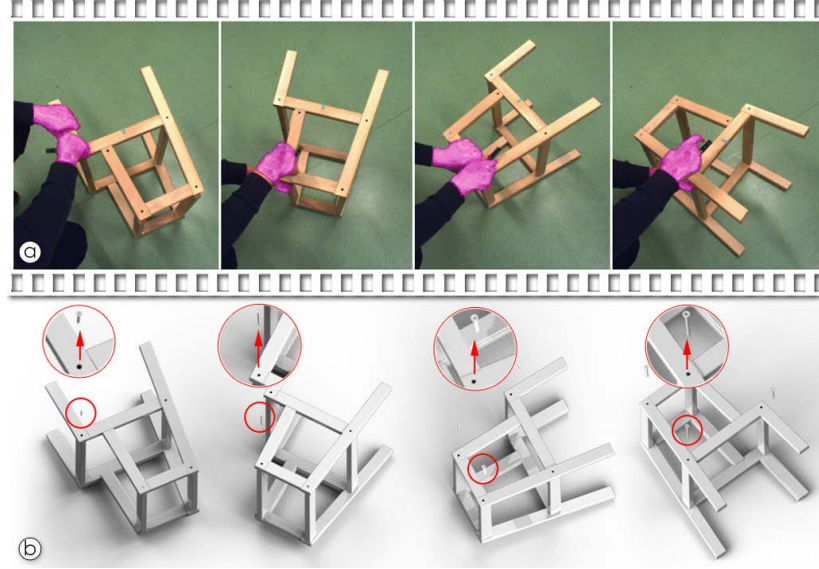


Figure 4.5: **Sequence refinement.** If multiple small parts block a large part, their order of removal is unknown. To refine the removal order for such parts, this method additionally searches for the user’s hands and tools in the corresponding video sequence. (a) Scaled hand masks are used to compute the overlap projection of small parts, and (b) the order of their removal is inferred from the order in which the hands overlap the parts. This example shows four screws that have been ordered based on the overlap of the hand masks with the projected footprint of the screws.

the user’s hands. The probability $P(O_k)$ is calculated over all pixels in a footprint of a removable part (Equation (4.1)):

$$P(O_k) = 1/|\Omega'(O_k)| \sum_{\mathbf{x} \in \Omega'(O_k)} (P_f(\mathbf{x}) - P_b(\mathbf{x})). \quad (4.1)$$

It is considered a part to be removed, if its probability $P(O_k)$ falls below a threshold $t_{removed}$. To compute $P(O_k)$ the current foreground likelihood model P_f and background likelihood models P_b is needed for a pixel at $\mathbf{x}$ in

the region $\Omega'(O_k)$. Since the likelihood models change over time, they are kept updating for a tracker if its confidence value is higher than a threshold t_{track} . The results in this chapter were generated using $t_{track} = 0.70$, $t_{removed} = 0.15$, and $n = 50$. This current implementation processes input videos at approximately 3 frames per second (fps) on a system using an Intel Core i7-7700 CPU, 32 GB RAM and an NVIDIA GeForce GTX 1060 GPU. Optimizations such as multi-threading can further improve the performance of the multi-objects tracking and the removal detection.

(5) Sequence Refinement

After detecting the removal of a larger part O_{large} , the smaller parts that may have not been detected are searched for. Since only unblocked parts can be removed from the assembly, it is assumed that any part that blocked O_{large} must have been removed before. Thus, the assembly graph is used to identify all parts O_{small} that block O_{large} and that have not been removed yet.

Since multiple parts O_{small} may block O_{large} , the removal order of all O_{small} must be determined as shown in the video. This is derived from analyzing the user's hands movement and the tools they are using. It is assumed that the user is removing a part using a tool or with their hands directly. This allows the user to infer the removal sequence from the locations of the user's hands and tools relative to the parts of the object. The hand and tool segmentation [41], and the footprint of all O_{small} are computed in the frames between the one where the removal of the current large part O_{large} is

detected and the one where the previous large part is detected. If no previous part exist, it starts from the last frame of the input video.

The footprint of a part O_{small} is computed by projecting its corresponding 3D model into the video frame using the estimated camera pose. Then, the overlap between a projected part and the hand and tools masks is computed and is assumed that the user is removing the part O_{small} with the largest overlap. If two or more parts O_{small} overlap with the same mask, they are grouped and shown their instruction simultaneously, instead of consecutively. Figure 4.5 illustrates this approach.

4.3.2 Visualization

Generating virtual 3D animations from the input video allows us to create free-viewpoint instruction visualizations. An interactive AR mirror is used to align the user’s viewpoint of the physical object with the virtual 3D instruction view shown in the mirror. The proposed setup requires a monitor and a 3D object tracker attached to the physical assembly to update the virtual camera during user interaction. Instead of relying on 2D segmentation-based 3D object tracking that is typically error-prone, an additional tracking device is used. Inspired by the approach of Mohr et al. [50], an off-the-shelf smartphone is employed, which this method attaches to the assembly to enable its tracking (see Figure 4.6).

(1) Texturing

To minimize cognitive load, the appearance of the 3D model is matched to the one in the video by using pixel colors from the video and applying them to the model texture. A texture atlas for each part is created and it is filled by projecting the pixel from a video frame onto the object using the estimated camera pose. Since the user's hands often occlude the object during the interaction, hand pixels are excluded before a frame is used. Video tutorials usually show the object of interest from multiple viewpoints. Since the camera pose is tracked, video frames can be projected onto the 3D assembly to fill the texture atlas. Only video frames where an object has been identified are projected. Some parts may not be visible, in which case an inpainting solution is additionally used to estimate missing pixel colors from those within the texture atlas [5] (Figure 4.7 (a)).

(2) Video Label Extraction

A video label is a cropped video excerpt focusing on the interaction with a part. The label is especially useful to communicate actions that are otherwise challenging to visualize in a 3D animation, such as tool usage or non-rigid object deformations. In such cases, the video label is able to present direct visual explanations of interactions. In order to focus the content of the video label, frames which do not show any actions are removed, and the remaining frames are cropped to the area that shows the assembly part in addition to the user's hands and tools that have been used to manipulate it. In particular, the

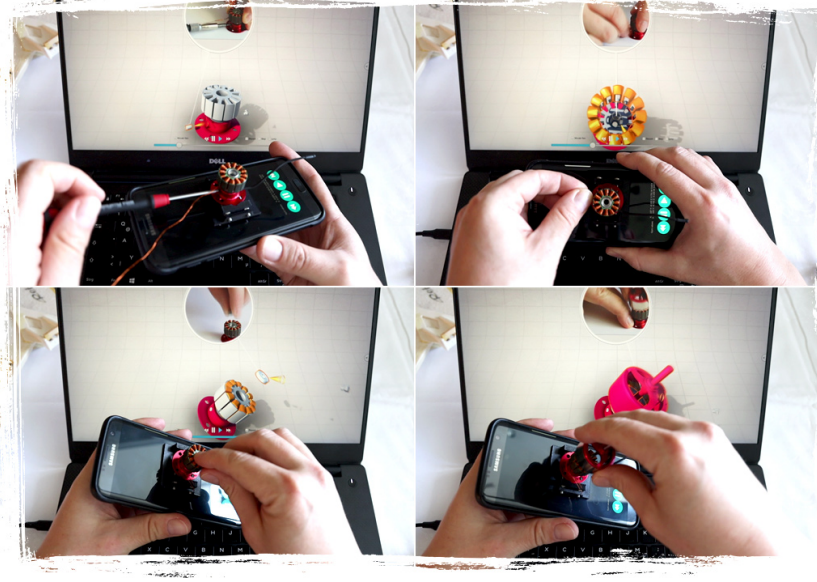


Figure 4.6: **AR visualization.** A common smartphone can be utilized for tracking the object-of-interest, and widely available displays can also be utilized as an AR mirror. In this example, the user follows a video-annotated AR tutorial that has been generated from a video tutorial showing the assembly of a motor on a laptop screen. The tutorial includes fastening small screws and coiling of a wire. These operations are presented as video annotations, as they cannot easily be extracted with current computer vision techniques.

video is searched for hands and tools using the feature descriptor presented by Li et al. [41] that is already used for sequence refinement. The result is used to produce a pixel mask, which is referred to as the action mask and, consequently, frames are removed where no action masks appear.

To focus on the content of a video label, the center $\mathbf{c}_l$ and the radius r_l of the most important area are calculated. First, the contours of parts, hands, and tools [73] are detected. When a tool and a hand, or two hands have been detected, their masks are combined and treated as one. Subsequently, the

minimal enclosing circles are computed on each contour. The centers of the part and the hand regions are obtained as $\mathbf{c}_p$ and $\mathbf{c}_h$, and the radii of the circles, which enclose the masks as r_p and r_h , respectively. Finally, the video label center $\mathbf{c}_l$ and its radius r_l are calculated using Equation (4.2) and (4.3). To stabilize video label regions over time, rolling average filtering is used for $\mathbf{c}_l$ and the median of r_l is computed throughout the video sequence. See Figure 4.7 (b), (c) for an example. This current implementation calculates video labels at approximately 37 fps (see above for PC details).

$$\mathbf{c}_l = (r_l - r_p) \frac{\mathbf{c}_h - \mathbf{c}_p}{\|\mathbf{c}_h - \mathbf{c}_p\|} + \mathbf{c}_p, \quad (4.2)$$

$$r_l = (\|\mathbf{c}_h - \mathbf{c}_p\| + r_h + r_p)/2. \quad (4.3)$$

(3) Video Label Placement

Labels are visualized as annotations of the object of interest. They consist of a circular representation of the video and a thin anchor line which connects the video with the object.

Dynamic placement. Labels are initially placed on a plane in 3D space [78], at the point where the corresponding video frame was captured from (Figure 4.8 (a)). However, as a 3D video label disappears when the user explores the object from the side or the back, the video is presented on a 2D billboard, which is placed on the bounding circle of the object, when the 3D label leaves a predefined viewing cone (55°) (Figure 4.8 (b)).

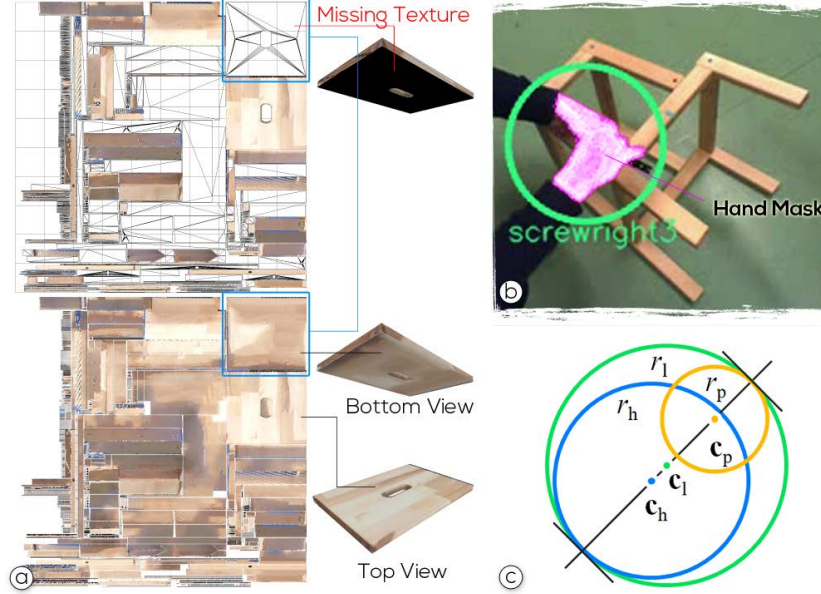


Figure 4.7: **Texture filling and label extraction.** (a) To visually align the 3D animation with the video label, the texture of the 3D model is filled by projecting video pixels from the tracked camera onto the object. (b) The information is focused on video labels by cropping the input video to the circle that encloses the user’s hand, tools, and the part that is assembled. (c) The center and radii of the detected part (yellow) and hand-and-tool (blue) are used to focus and stabilize the video label region (green).

Thus, depending on the current viewpoint, it automatically switches between the 3D and the billboard placement. However, based on user feedback in a pilot study with three participants, the placement strategy is to be updated. All users raised the issue of overlap of the video and the object, when the virtual camera is aligned with the one in the video in the 3D presentation. This eventually obstructs their view of the object. Even when videos show more information than the 3D animation, all users mentioned they felt insecure about the instruction when the 3D animation was not visible any-

more. In addition, all of them commented on the distracting motion that is introduced by the moving label. This is in line with the findings of Madsen et al. [43], who study the impact of label motion on user performance. It was initially wrongly assumed that the 3D placement would allow users to anticipate the movement of the label and therefore not be distracted.

Static placement. To compensate for label motion and object occlusion, the placement strategy was refined. The label was first placed in the corner of the display (Figure 4.9) and disabled positional updates. In addition to video labels, a 3D cone visualization and a 3D camera icon are presented, which indicate the 3D camera pose and the field of view that was used for capturing the video (Figure 4.8 (c) shows a screenshot that includes the camera icon and the view cone).

While the static placement was well received in a second pilot with the three users, the position was criticized as being too far away from the object. The users reported that the distance made it difficult to look at both, the 3D animation and the video, simultaneously. Therefore, the static placement was kept but the label center was moved to the point where the object's up vector intersects its scaled bounding circle. The proposed final placement strategy avoids motion and occlusion, and the placement on the up vector enforces alignment between the object and the label, therefore bringing the whole content to the user's view.

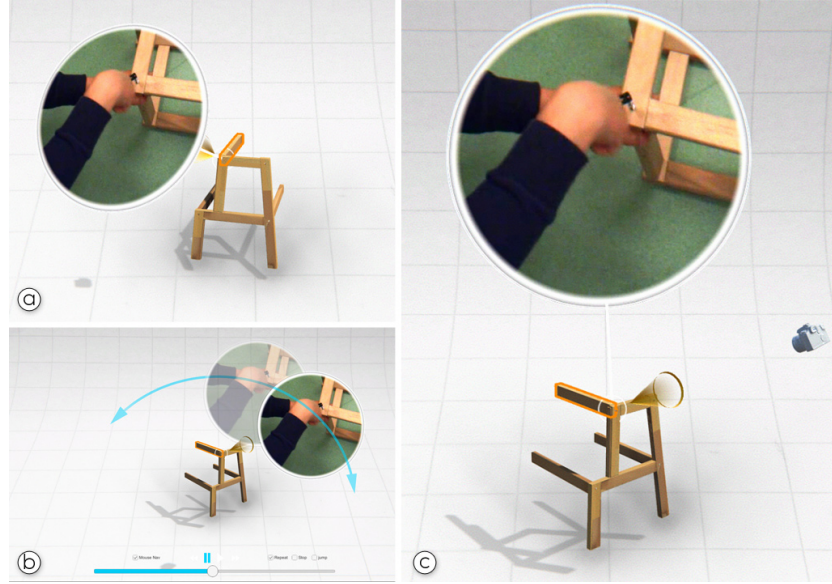


Figure 4.8: **Label placement.** (a) The video is placed on a plane in 3D space using the estimated pose of the video camera. In addition, a 3D cone is rendered to indicate a predefined field of view and to connect the label with the object. (b) A 3D label is displayed only when the virtual camera is in the field of view. For all other camera positions, the center of the label is projected to the scaled bounding circle of the object. This causes the label to move on a circle around the object when the camera is outside the viewing cone. (c) Based on user feedback, the placement strategy was refined. To avoid label motion and occlusions, the label is placed on the object’s up vector, at a fixed distance outside its bounding circle. It is placed close enough to observe both, the 3D animation and the video. To indicate the camera pose a 3D camera icon is added.

4.4 Experiment

The results of this approach were validated to automatically create step-by-step AR instructions from an input video in a user study where participants followed these instructions to assemble an object. The extracted AR instructions was compared to the baseline method of using traditional video tutorials

to investigate if this approach generated a complete set of instructions from the input video. The evaluation also allowed us to collect subjective feedback regarding the usability and cognitive load of the system. This AR visualization follows established guidelines for optimal instruction design and, thus, uses step-by-step instructions, animations and free viewpoint changes to reduce a user's cognitive load. Due to these design guidelines, it was expected the AR condition to be superior to a video tutorial in a typical assembly task.

Design. A within-subject study was designed with an independent variable “instruction method” with two conditions: video and AR. The video condition corresponded to a video tutorial without zoom effects or cuts. The video showed the assembly from a viewpoint defined by the video's author. As is typical with current online video tutorials, the video did not automatically pause playback between assembly steps. However, users could pause the video at any time. In AR, users worked with the presented AR mirror system. The system paused automatically at each segmented instructional step, and users could manually forward by pressing a button on a keyboard. Furthermore, users could freely change the viewpoint.

Task completion time (TCT) was measured, i.e., time to finish the assembly, the total number of assembly errors, cognitive load as mental effort using the commonly used rating scale of Paas [55], usability with the System Usability Scale (SUS) [10], task load using the NASA TLX questionnaire [27], and overall preference. If not indicated otherwise, numerical values in

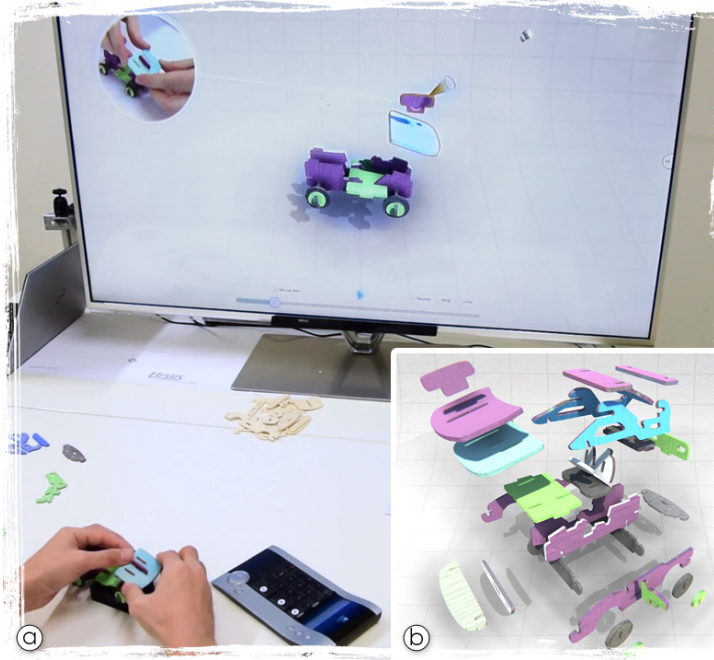


Figure 4.9: **Apparatus.** (a) In the study, a 50 inch screen was employed in front of the user. (b) Exploded view of the assembly used in the study.

the text of this chapter are reported in the format “mean (sd).” For the proposed study, 16 participants (2 female, $\bar{X} = 27.8$ (3.85) years) volunteered. On a scale from one to five (best), the mean of self-rated AR experience was 3 (sd=1.03, median=3), and the mean of the handicraft work experience was 2.93 (sd=0.99, median=3). 6.25% of participants rated AR experience as 1, 31.5% as 2, 18.75% as 3, 43.75% as 4. Hence, the majority of those participants had at least intermediate AR experience. All the participants except one had a computer science background. One participant had an engineering background and self-rated AR experience of 3, thus, also experience with AR.

Apparatus. Video and AR conditions were displayed on a 50 inch screen. The screen was located in front of the assembly area. Participants were seated in front of the screen and followed the presented instructions. Users had to assemble a car model with 34 parts attached to a wireless IMU for tracking. In the video condition, they could interact with the video and navigate between assembly steps in the AR condition using a computer mouse and a keyboard with labeled buttons for video controls. Figure 4.9 shows the apparatus.

Procedure. After filling out an informed consent form and demographics questionnaire, the participants familiarized themselves with the system using a trial assembly. To avoid learning effects, the object used in the trial task was different from the one used in the main task. If they were familiar with the interfaces of both methods, they started the task with one instructional method. The participants were instructed to be fast and accurate. After completing a condition, they filled in the mental effort, SUS, and NASA TLX questionnaires and continued with the remaining instructional method. After completing the final task, they filled out the preference questionnaire.

The starting order of the instruction method was counterbalanced. After assembling half of the car model, the participants continued assembling the car using the other instruction method. This procedure was followed, because using the same toy car allows us to generate comparable quantitative data between both instruction methods. In this study, both halves of the car have been assembled using both methods an equal number of times in order

to compensate for differences in assembly complexity. The overall mean of these measurements is calculated for each instruction method.

Hypotheses. It was expected that the AR condition would outperform the video condition with respect to TCT (**H1**) and the total number of errors (**H2**) due to the design of the system that takes into account research from instruction design and CLT. Consequently, it was also expected to see better usability (SUS) (**H3**), less mental effort (**H4**), less task load (NASA TLX) (**H5**) and overall preference (**H6**) for the AR condition.

Results. The data were evaluated using a significance level of 0.05. As the data did not fulfill the normality requirements, it was analyzed using Wilcoxon signed-rank tests; effect sizes are calculated as $r = \frac{Z}{\sqrt{N}}$, as proposed by Fritz et al. [21]. The analysis was performed using the statistics software *R*.

Wilcoxon signed-rank tests revealed statistically significant differences in mental effort (video 3.9 (1.9); AR 3.0 (1.0); $Z=2.36$, $p<0.05$, $r=0.42$), SUS (video 70.5 (14.3); AR 86.3 (8.2); $Z=2.9$, $p<0.01$, $r=0.51$) and the factor temporal demand of the NASA TLX (video 48.1 (24.3); AR 28.4 (22.1); $Z=2.42$, $p<0.05$, $r=0.43$). There were no significant differences in TCT (video 183 (43); AR 182 (51)), error (video 0.6 (1.5); AR 0.5 (0.8)) or raw NASA TLX (video 37 (18); AR 31 (13)).

Differences in assembly complexity between the two halves of the car were checked by calculating Wilcoxon signed-rank tests for each instruction

method and assembly half. The tests did not reveal statistically significant differences.

Discussion. There was no statistically significant difference in TCT and the number of errors. Therefore, H1 (TCT) and H2 (error) are rejected. However, the subjective results clearly demonstrate the advantages of this AR system over traditional video tutorials. H6 (preference) is accepted, as 100% of the participants preferred the AR system over the traditional video tutorial. 62.5% of participants had a self-rated AR experience above 3. Therefore, this strong preference for the AR condition cannot be fully explained by a novelty effect, but is an indicator of the perceived superiority of the AR condition over video only. Based on the result of the statistical evaluation, hypothesis H3 (SUS) and H4 (mental effort) are accepted. The superior mean SUS score of 87 for this AR system indicates a usability rating above the average score of 70, that according to the analysis of Bangor et al. [4] translates into the adjective “excellent.” The participants also had to invest significantly less mental effort in solving the tasks using the proposed AR system.

These positive results for the AR condition are underlined by the qualitative feedback of the participants. As unsolicited feedback in an unstructured after-study interview, 37.5% of the participants stated that the AR visualization was easy to understand and very clear. Furthermore, 50% of the participants positively remarked on the additional interactivity and viewpoint control in the AR condition that it allowed them to better un-

derstand the task and assembly shapes. 19% of the participants explicitly mentioned the video annotations and stated that they were especially useful to understand the details of the instructions, when the 3D rendering was not sufficiently clear. These observations indicate that the proposed algorithm for extracting assembly steps and the visualizations work as expected.

The results confirmed observed that the video inpainting of the 3D parts shown in the instruction animation supported the participants in identifying the corresponding real-world parts during the assembly task. In the interview, only a few participants (12.5%) explicitly pointed out the advantages of visualizing assembly parts with the same texture as the real-world parts. However, throughout the assembly tasks, 37.5% of the participants mixed up two similarly colored parts, thereby making the same assembly errors. This clearly indicates that the participants used the parts' visual appearance to identify the subsequent part and highlights the value of using inpainting for the 3D visualization.

While a statistically significant difference was not found in the NASA TLX, H5 is accepted partially as the test revealed a significant difference in temporal demand. This difference may be explained by the design of the instructions. 25% of the participants stated that they were less pressured by the AR condition due to the presentation as step-by-step instructions compared to the video condition that had to be paused manually. In the video condition, the participants usually tried to work at the speed of the video. Hence, the video was subjectively found to be putting more pressure

on the participants than the step-by-step instructions of the AR condition.

The difference of temporal demand between AR and video instructions may also explain the lack of a significant difference in TCT. According to the proposed observations and the participants' feedback, in the video condition, the participants tried to work at the speed of the video without pausing it, while they took their time in the AR condition and investigated the instructions from different viewpoints. Hence, the participants sped up their interaction in the video condition, while they slowed down in the AR condition, thus, influencing the TCT. Furthermore, the lower mental effort for the AR condition indicates that understanding and following step-by-step AR instructions was easier than following the video tutorial, even though the latter could have also been paused on demand.

Overall, the evaluation showed that the users were able to complete the tasks with both video and AR instructions. This indicates that the extracted instructions and generated visualizations of the proposed system are complete so that users can follow the generated instructions without difficulty. One major advantage of the AR mirror approach lies in the interactive visualization of assembly steps most notably of small and occluded parts. In such cases, video tutorials require zooming in on small parts, or a change of camera angles, which leads to a mismatch between the user's view of the real-world object and the video's viewpoint. Videos are also often edited to cut between viewpoints, which requires additional mental rotations from the viewer. The interactive viewpoints of AR facilitate this reorientation as is

indicated by the user feedback and the decreased mental effort. In addition, the 3D animation of the assembly instruction naturally avoids occlusions, e.g., by hands or tools. A future study should isolate the impact of these factors by controlling occlusions in the video and by controlling viewpoint discrepancies between the video and the real-world object.

The usability of the video condition was rated as 70.5 (SUS), which corresponds to average usability and the adjective “good” [4]. This shows that the video condition was perceived positively despite its limitations. Usability could likely be improved by segmenting the video like in the AR condition. It was noticed that the participants rarely used the pause button on the video. This is in line with the results of Biard et al. [6], who additionally showed that users performed better when videos were automatically pausing between steps. Interestingly, the participants did not remark upon such a feature for videos in the proposed unstructured after-study interview but requested additional AR features such as more animation control (e.g., speed), or a preview of the next steps. However, although segmented video instructions may improve in terms of temporal demand, the participants would still be required to mentally align the video viewpoint with the real object and the video would suffer from occlusions (e.g., by the author’s hands), both of which may contribute to increased mental effort. A future study should isolate the effect of instruction segmentation for videos on the task performance when compared to the AR mirror system.

4.5 Summary

This chapter shows that a method for estimating the relative pose of a camera to the target object in a dynamic environment is applied to generate AR content and this approach successfully extracts assembly instructions from video tutorials. Given only a video recording and the object's 3D model, the object's assembly steps are inferred in their correct order by estimating the 3D assembly poses from the camera. This method takes inspiration from the technique of assembly-by-disassembly and parses input videos from end to beginning to facilitate object extraction. Furthermore, the object's assembly graph is utilized to identify the manipulation of small or occluded parts. The completeness of the extracted instructions was evaluated in a user study that demonstrated that users were indeed able to follow the generated instructions and to complete the assembly tutorial.

In the following, a series of design recommendations for visual instructions of retargeted videos are enumerated. Generally, this AR mirror instruction visualization caused less mental effort and, thus, cognitive load than a common video tutorial. This is a result of grounding the proposed design in findings from current state-of-the-art: (1) this chapter shows assembly instructions segmented into step-by-step animations that are shown at a speed chosen by the user [6, 70]; (2) this chapter allows users to control the viewpoint of the instructions so that they are always aligned with the corresponding real-world object [24] and can be used to resolve occlusions or

to reorient during the task [29]. This chapter adds to the current research, as this method is the first to introduce (3) video annotated AR instructions combining free viewpoint navigation and detailed instruction visualization. Video annotations are essential to capture assembly steps that cannot easily be extracted with current computer vision algorithms (e.g., tool usage, complex assembly motion, non-rigid object deformations), while AR instructions enable aligning the viewpoint of the real-world assembly with the visualization. Furthermore, the proposed user study clearly showed that color is a strong cue during users' visual search for the next assembly part [89]. This aspect needs further investigation in the context of video retargeting, as (4) video texturing is a useful tool to visually align the virtual 3D instructions with the real-world scene and, thus, facilitates visual search. However, a future system must consider visually similar assembly parts and provide automatic guidance to resolve similarities of color and/or shape that could induce assembly errors.

Limitations

This approach processes detailed assembly videos. Currently, this system can process videos that show objects that are continuously manipulated in a series of video frames without interruptions. While cuts in the video can be handled by reinitializing the object tracking, stop-motion sequences or slideshows will require frequent reinitialization, which decreases usability. Furthermore, while this system can detect small parts, it relies on the trackability of larger

parts. For optimal detection results, content providers should take care of appropriate lighting, improving contrast to the background and calibrating the recording camera. This system currently does not use audio information from the video. However, a combination of extracted video annotations and voice overlay could further enhance the presented information. Also, this experiment was conducted on a car model in the environment shown in Figure 4.9. However, it is necessary to experiment with various kinds of objects and environments to make the method more applicable.

Chapter 5

Conclusion and Outlook

In these studies, three types of relative poses of a camera to the environment were categorized and their problems and solutions were described for each of them.

Chapter 2 described a robust estimation of the relative camera poses using a pre-built 3D structure of the environment. When the camera pose was estimated in a pre-created 3D space, a problem occurred since the scale of the current camera pose differed from the original 3D space. Therefore, in Chapter 2, the scale was recalculated from the camera poses obtained by VO. In addition, the camera poses were recalculated in all frames by graph optimization so that the camera trajectory could be estimated on the pre-built 3D structure.

Chapter 3 showed a method of superimposing physical quantity distributions acquired by cameras with different modalities on a 3D map of the environment as estimated by a normal visible light camera. This allowed

users to visualize the 3D structure of the temperature distribution.

Chapter 4 proposed a novel interactive visualization technique to present assembly tutorials in 3D space from an RGB image sequence. Chapter 4 also proposed a method for generating a 3D AR tutorial from a 2D video input by knowing the relative poses of a color camera to an object. The video input was analyzed using a 3D model with an assembly graph. The usefulness of this method was demonstrated by comparing the proposed tutorials with video.

Throughout the three chapter, a method for improving the estimation accuracy of camera pose and a new visualization method were proposed using pre-measured 3D shapes.

Future Work

These methods can be further developed to measure 3D space with higher accuracy and richer data. Such developments are expected to improve the accuracy of automatic robot driving and expand the range of AR applications. This chapter discusses two possible applications based on the proposed method.

One direction is to expand the elements comprising a pre-built map as described in Chapter 2. Figure 5.1 shows a specific implementation. The appearance and disappearance of an object are detected from the difference obtained by comparing the label on the current camera image with the previously obtained label on the 3D structure. Instance segmentation is used

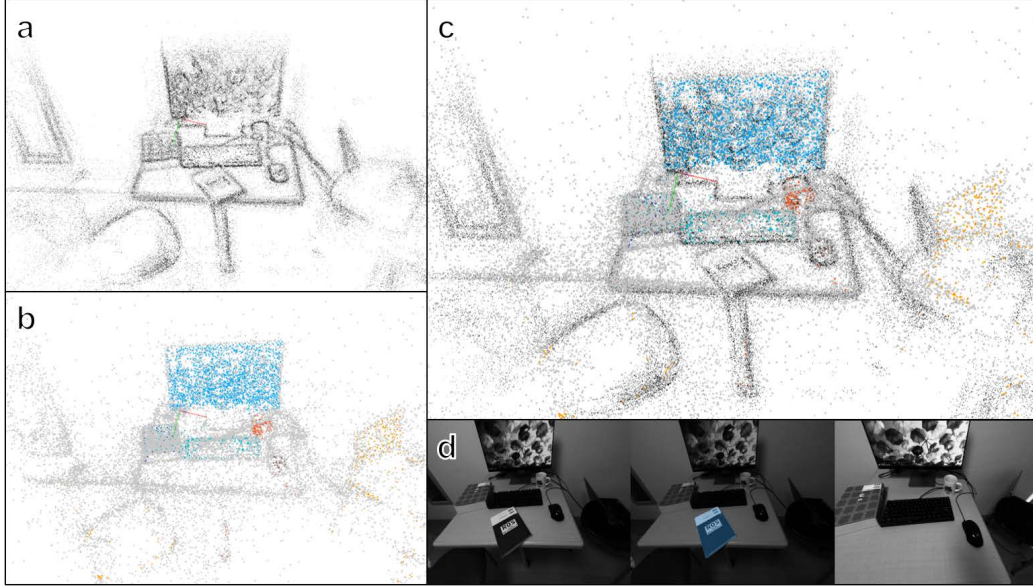


Figure 5.1: **Object detection using global map.** (a) 3D reconstruction using VO, (b) The global map with object labels, (c) Output registration result of VO on the global map, (d) (left) VO input, (middle) Object detection result, (right) Global map input image used for comparison.

to associate an instance label for each object with a 3D point in the pre-built global map. There are many possible applications of using a pre-built global map in such a manner. For example, when visualizing the temperature distribution as a three-dimensional structure in Chapter 3, it is possible to identify the abnormal parts of the temperature distribution more efficiently by combining the method of Chapter 2 with a pre-map that includes the locations to be observed in advance. Also, when assembling objects in Chapter 4, the assembly tutorials can be performed more efficiently by preserving the locations of parts and other objects on the pre-built global map in advance. One of the most important points is how to update the pre-built map.

As mentioned in Chapter 2, this thesis does not describe how to update the pre-built map. However, a method to update the pre-built map with each observation and add new data is mandatory when considering such an application. How to update and apply the data will be an issue that needs to be considered further in the use of maps. In addition, considering the applications for smartphones, the issue is how to reduce the data size of the pre-built map.

The other direction is to combine the method of estimating the relative poses from the environment to the camera in Chapter 2 and the method of estimating the relative poses from the camera to the object in Chapter 4 to measure 3D space. It is possible to combine the pre-built map created in Chapter 2 with the dynamic object pose estimation method in Chapter 4. This method is considered effective in estimating the camera poses and the object poses even under dynamic environments. In a dynamic environment, estimating the camera poses from a series of images using an RGB camera is expected to be more accurate than the previous method of estimating the camera poses using an RGB-D camera, which ignores dynamic objects. Moreover, the application of camera position and pose estimation can be expanded by registering objects incrementally through Dynamic Scene Understanding, as opposed to the existing label-based methods such as Chapter 4 and Figure 5.1.

References

- [1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building Rome in a Day. *Communications of the ACM*, 54(10):105–112, October 2011.
- [2] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing Effective Step-by-Step Assembly Instructions. *ACM Transactions on Graphics (TOG)*, 22(3):828–837, July 2003.
- [3] Guillem Alenyà, Sergi Foix, and Carme Torras. Using ToF and RGBD Cameras for 3D Robot Perception and Manipulation in Human Environments. *Intelligent Service Robotics*, 7(4):211–220, September 2014.
- [4] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies archive*, 4(3):114–123, May 2009.
- [5] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (TOG)*, 28(3), July

- 2009.
- [6] Nicolas Biard, Salomé Cojean, and Eric Jamet. Effects of Segmentation and Pacing on Procedural Learning by Video. *Computers in Human Behavior*, 89:411–417, December 2018.
 - [7] Dorit Borrmann, Jan Elseberg, and Andreas Nüchter. Thermal 3D Mapping of Building Façades. *Intelligent Autonomous Systems 12*, pages 173–182, 2013.
 - [8] Gary Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
 - [9] Paul Breedveld. Observation, Manipulation, and Eye-Hand Coordination Problems in Minimally Invasive Surgery. In *European Conference on Human Decision Making and Manual Control*, pages 219–231, December 1997.
 - [10] John Brooke. SUS: A Quick and Dirty Usability Scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.
 - [11] Andreas Butz. BETTY: Planning and Generating Animations for the Visualization of Movements and Spatial Relations. In *Proceedings of the Workshop on Advanced Visual Interfaces*, pages 53–58, June 1994.
 - [12] Robert Castle, Georg Klein, and David W. Murray. Video-Rate Localization in Multiple Maps for Wearable Augmented Reality. In *IEEE International Symposium on Wearable Computers*, pages 15–22, September 2008.

-
- [13] Arturo De la Escalera and Jose María Armingol. Automatic Chessboard Detection for Intrinsic and Extrinsic Camera Parameter Calibration. *Sensors*, 10(3):2027–2044, 2010.
 - [14] L.S. Homem De Mello and Arthur C. Sanderson. A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. *IEEE Transactions on Robotics and Automation*, 7(2):228–240, April 1991.
 - [15] Frederic Devernay and Olivier Faugeras. Straight Lines have to be Straight. *Machine Vision and Applications*, 13(1):14–24, August 2001.
 - [16] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(3):611–625, March 2017.
 - [17] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale Direct Monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849, September 2014.
 - [18] Steven Feiner, Blair Macintyre, and Dorée Seligmann. Knowledge-Based Augmented Reality. *Communications of the ACM*, 36(7):53–62, July 1993.
 - [19] Mark Fiala. Magic Mirror System with Hand-held and Wearable Augmentations. *IEEE Virtual Reality Conference*, pages 251–254, March 2007.

-
- [20] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014.
 - [21] Catherine O. Fritz, Peter E. Morris, and Jennifer J. Richler. Effect Size Estimates: Current Use, Calculations, and Interpretation. *Journal of Experimental Psychology: General*, 141(1):2–18, February 2012.
 - [22] Dorian Gálvez-López and Juan D Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.
 - [23] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct Sparse Odometry with Loop Closure. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204, October 2018.
 - [24] T. B. Garland and Christopher A. Sanchez. Rotational Perspective and Learning Procedural Tasks from Dynamic Media. *Computers and Education*, 69:31–37, November 2013.
 - [25] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. Duplo-Track: A Real-Time System for Authoring and Guiding Duplo Block Assembly. In *Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 389–402, October 2012.
 - [26] Youngjib Ham and Mani Golparvar-Fard. An Automated Vision-Based Method for Rapid 3D Energy Performance Modeling of Existing Build-

- ings using Thermal and Digital Imagery. *Advanced Engineering Informatics*, 27(3):395–409, August 2013.
- [27] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology*, 52:139–183, 1988.
- [28] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
- [29] Julie Heiser, Doantam Phan, Maneesh Agrawala, Barbara Tversky, and Pat Hanrahan. Identification and Validation of Cognitive Design Principles for Automated Generation of Assembly Instructions. In *the Working Conference on Advanced Visual Interfaces*, pages 311–319, May 2004.
- [30] Steven Henderson and Steven Feiner. Exploring the Benefits of Augmented Reality Documentation for Maintenance and Repair. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(10):1355–1368, October 2011.
- [31] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562, November 2012.
- [32] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An Overview of Depth Cameras and Range Scanners Based

- on Time-of-Flight Technologies. *Machine Vision and Applications*, 27(7):1005–1020, June 2016.
- [33] Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Comprehensible Visualization for Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(2):193–204, March 2009.
- [34] Denis Kalkofen, Markus Tatzgern, and Dieter Schmalstieg. Explosion Diagrams in Augmented Reality. In *IEEE Virtual Reality Conference*, pages 71–78, March 2009.
- [35] Bernhard Kerbl, Denis Kalkofen, Markus Steinberger, and Dieter Schmalstieg. Interactive Disassembly Planning for Complex Objects. *Computer Graphics Forum*, 34(2):287–297, May 2015.
- [36] Bernhard Kornberger. Fade2D Delaunay Triangulation Library. <https://www.geom.at/contact/publications/>.
- [37] Hideyuki Kume, Arne Suppé, and Takeo Kanade. Vehicle Localization along a Previously Driven Route Using an Image Database. In *IAPR International Conference on Machine Vision Applications (MVA)*, pages 177–180, May 2013.
- [38] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A General Framework for Graph Optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, June 2011.

-
- [39] Florian Ledermann and Dieter Schmalstieg. APRIL A High-Level Framework for Creating Augmented Reality Presentations. In *IEEE Virtual Reality Conference*, pages 187–194, March 2005.
 - [40] Der-Tsai Lee and Bruce J Schachter. Two Algorithms for Constructing a Delaunay Triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242, June 1980.
 - [41] Cheng Li and Kris M. Kitani. Pixel-Level Hand Detection in Ego-Centric Videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3570–3577, June 2013.
 - [42] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated Generation of Interactive 3D Exploded View Diagrams. *ACM Transactions on Graphics (TOG)*, 27(3):1–7, August 2008.
 - [43] Jacob Boesen Madsen, Markus Tatzgern, Claus B. Madsen, Dieter Schmalstieg, and Denis Kalkofen. Temporal Coherence Strategies for Augmented Reality Labeling. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(4):1415–1423, April 2016.
 - [44] Sotiris Makris, George Pintzos, Loukas Rentzos, and George Chrysosouris. Assembly Support Using AR Technology Based on Automatic Sequence Generation. *Cirp Annals-manufacturing Technology*, 62(1):9–12, 2013.
 - [45] Kazuki Matsumoto, Wataru Nakagawa, Hideo Saito, Maki Sugimoto, Takashi Shibata, and Shoji Yachida. AR Visualization of Thermal 3D

- Model by Hand-held Cameras. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 480–487, March 2015.
- [46] Paul Mijksenaar and Piet Westendorp. *Open Here. The Art of Instructional Design*. Thames & Hudson, 1999.
- [47] Niloy J. Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. Illustrating How Mechanical Assemblies Work. *ACM Transactions on Graphics (TOG)*, 29(4):58:1–58:12, July 2010.
- [48] Peter Mohr, Bernhard Kerbl, Michael Donoser, Dieter Schmalstieg, and Denis Kalkofen. Retargeting Technical Documentation to Augmented Reality. In *Annual ACM Conference on Human Factors in Computing Systems (CHI)*, pages 3337–3346, April 2015.
- [49] Peter Mohr, David Mandl, Markus Tatzgern, Eduardo E. Veas, Dieter Schmalstieg, and Denis Kalkofen. Retargeting Video Tutorials Showing Tools with Surface Contact to Augmented Reality. In *Annual ACM Conference on Human Factors in Computing Systems (CHI)*, pages 6547–6558, May 2017.
- [50] Peter Mohr, Markus Tatzgern, Tobias Langlotz, Andreas Lang, Dieter Schmalstieg, and Denis Kalkofen. TrackCap: Enabling Smartphones for 3D Interaction on Mobile Head-Mounted Displays. In *Annual ACM Conference on Human Factors in Computing Systems (CHI)*, pages 1–11, May 2019.
- [51] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D cameras. *IEEE*

- Transactions on Robotics*, 33(5):1255–1262, June 2017.
- [52] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, October 2011.
- [53] Marc Nienhaus and Jürgen Döllner. Dynamic Glyphs - Depicting Dynamics in Images of 3D Scenes. In *International Symposium on Smart Graphics*, pages 102–111, June 2003.
- [54] Masatoshi Okutomi and Takeo Kanade. A Multiple-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 15(4):353–363, April 1993.
- [55] Fred G. Paas. Training Strategies for Attaining Transfer of Problem Solving Skills in Statistics : a Cognitive-load Approach. *Journal of Educational Psychology*, 84(4):429–434, December 1992.
- [56] Vivek Pradeep, Christoph Rhemann, Shahram Izadi, Christopher Zach, Michael Bleyer, and Steven Bathiche. MonoFusion: Real-time 3D Reconstruction of Small Scenes with a Single Web Camera. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–88, October 2013.
- [57] Surya Prakash, Pei Yean Lee, Terry Caelli, and Tim Raupach. Robust Thermal Camera Calibration and 3D Mapping of Object Surface

- Temperatures. *Thermosense XXVIII*, 6205(1):62050J–62050J–8, April 2006.
- [58] Victor A. Prisacariu and Ian D. Reid. PWP3D: Real-time Segmentation and Tracking of 3D Objects. *International Journal of Computer Vision*, 98(3):335–354, January 2012.
- [59] Tong Qin, Peiliang Li, and Shaojie Shen. Relocalization, Global Optimization and Map Merging for Monocular Visual-Inertial SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1197–1204, May 2018.
- [60] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, July 2018.
- [61] Carlos Ricolfe-Viala and Antonio-José Sánchez-Salmerón. Robust Metric Calibration of Non-linear Camera Lens Distortion. *Pattern Recognition*, 43(4):1688–1699, April 2010.
- [62] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, August 2004.
- [63] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. ORB: an Efficient Alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, November 2011.

-
- [64] Philip Saponaro, Scott Sorensen, Stephen Rhein, and Chandra Kambhamettu. Improving Calibration of Thermal Stereo Cameras using Heated Calibration Board. In *IEEE International Conference on Image Processing (ICIP)*, pages 4718–4722, September 2015.
 - [65] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, January 2018.
 - [66] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, June 2016.
 - [67] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, pages 501–518, September 2016.
 - [68] Tianjia Shao, Dongping Li, Yuliang Rong, Changxi Zheng, and Kun Zhou. Dynamic Furniture Modeling Through Assembly Instructions. *ACM Transactions on Graphics (TOG)*, 35(6), November 2016.
 - [69] Tianjia Shao, Wilmot Li, Kun Zhou, Weiwei Xu, Baining Guo, and Niloy J. Mitra. Interpreting Concept Sketches. *ACM Transactions on Graphics (TOG)*, 32(4), July 2013.
 - [70] Ingrid A.E. Spanjers, Tamara Van Gog, Pieter Wouters, and Jeroen J.G. Van Merriënboer. Explaining the Segmentation Effect in Learning from

- Animations: The Role of Pausing and Temporal Cueing. *Computers and Education*, 59(2):274–280, September 2012.
- [71] Colin Studholme, Derek LG Hill, and David J Hawkes. An Overlap Invariant Entropy Measure of 3D Medical Image Alignment. *Pattern Recognition*, 32(1):71–86, January 1999.
- [72] Mengü Sukan, Carmine Elvezio, Ohan Oda, Steven Feiner, and Barbara Tversky. Parafrustum: Visualization Techniques for Guiding a User to a Constrained Set of Viewing Positions and Orientations. In *Annual ACM Symposium on User Interface Software and Technology (UIST)*, page 331–340, October 2014.
- [73] Satoshi Suzuki and Keiichi Abe. Topological Structural Analysis of Digitized Binary Images by Border Following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, April 1985.
- [74] Takafumi Taketomi, Tomokazu Sato, and Naokazu Yokoya. Real-Time and Accurate Extrinsic Camera Parameter Estimation using Feature Landmark Database for Augmented Reality. *Computers and Graphics*, 35(4):768–777, August 2011.
- [75] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM Algorithms: a Survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, June 2017.
- [76] Arthur Tang, Charles Owen, Frank Biocca, and Weimin Mou. Comparative Effectiveness of Augmented Reality in Object Assembly. In *An-*

- nual ACM Conference on Human Factors in Computing Systems (CHI)*, pages 73–80, April 2003.
- [77] Markus Tatzgern, Raphael Grasset, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, and Dieter Schmalstieg. Exploring Real World Points of Interest: Design and Evaluation of Object-centric Exploration Techniques for Augmented Reality. *Pervasive and Mobile Computing*, 18:55–70, April 2015.
- [78] Markus Tatzgern, Denis Kalkofen, Raphael Grasset, and Dieter Schmalstieg. Hedgehog Labeling: View Management Techniques for External Labels in 3D Space. In *IEEE Virtual Reality*, pages 1–6, April 2014.
- [79] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.
- [80] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. Global Localization from Monocular SLAM on a Mobile Phone. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 20(4):531–539, March 2014.
- [81] Jonathan Ventura and Tobias Höllerer. Wide-area Scene Mapping for Mobile Visual Tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 3–12, November 2012.
- [82] Stephen Vidas, Ruan Lakemond, Simon Denman, Clinton Fookes, Sridha Sridharan, and Tim Wark. A Mask-Based Approach for the Geometric Calibration of Thermal-Infrared Cameras. *IEEE Transac-*

- tions on Instrumentation and Measurement*, 61(6):1625–1635, February 2012.
- [83] Stephen Vidas, Peyman Moghadam, and Michael Bosse. 3D Thermal Mapping of Building Interiors using an RGB-D and Thermal Camera. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2311–2318, May 2013.
- [84] Paul Viola and William M Wells III. Alignment by Maximization of Mutual Information. *International Journal of Computer Vision*, 24(2):137–154, September 1997.
- [85] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517, May 2018.
- [86] Bin Wang, Guofeng Wang, Andrei Sharf, Yangyan Li, Fan Zhong, Xueying Qin, Daniel CohenOr, and Baoquan Chen. Active Assembly Guidance with Online Video Parsing. In *IEEE Conference on Virtual Reality and 3D User Interfaces*, pages 459–466, March 2018.
- [87] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo DSO: Large-Scale Direct Sparse Visual Odometry With Stereo Cameras. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3903–3911, October 2017.
- [88] Martin Weinmann, Jens Leitloff, Ludwig Hoegner, Boris Jutzi, Uwe Stilla, and Stefan Hinz. Thermal 3D Mapping for Object Detection

- in Dynamic Scenes. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(1):53–60, November 2014.
- [89] Jeremy M. Wolfe. Guided Search 2.0 a Revised Model of Visual Search. *Psychonomic Bulletin & Review*, 1:202–238, June 1994.
- [90] Jimmy Wu, Bolei Zhou, Rebecca Russell, Vincent Kee, Syler Wagner, Mitchell Hebert, Antonio Torralba, and David MS Johnson. Real-Time Object Pose Estimation with Pose Interpreter Networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6798–6805, October 2018.
- [91] Masahiro Yamaguchi, Shohei Mori, Peter Mohr, Markus Tatzgern, Ana Stanescu, Hideo Saito, and Denis Kalkofen. Video-Annotated Augmented Reality Assembly Tutorials. In *Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 1010–1022, October 2020.
- [92] Masahiro Yamaguchi, Shohei Mori, Hideo Saito, Shoji Yachida, and Takashi Shibata. Global-Map-Registered Local Visual Odometry Using On-the-Fly Pose Graph Updates. In *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 299–311. Springer, August 2020.
- [93] Masahiro Yamaguchi, Trong Phuc Truong, Shohei Mori, Vincent Nozick, Hideo Saito, Shoji Yachida, and Hideaki Sato. Superimposing Thermal-Infrared Data on 3D Structure Reconstructed by RGB Visual Odometry. *IEICE Transactions on Information and Systems*, 101(5):1296–1307, 2018.

-
- [94] Kazutaka Yasuda, Takeshi Naemura, and Hiroshi Harashima. Thermo-Key: Human Region Segmentation from Video. *IEEE Computer Graphics and Applications*, 24(1):26–30, January 2004.
 - [95] Guangjun Zhang, Junji He, and Xianming Yang. Calibrating Camera Radial Distortion with Cross-Ratio Invariability. *Optics & Laser Technology*, 35(6):457–461, September 2003.
 - [96] Zhengyou Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.