UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

THIAGO DE AZEVEDO DORNELLES

Online frame-to-model pipeline to 3D reconstruction with depth cameras using RGB-D information

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Claudio Jung

Porto Alegre
May 2020

CIP — CATALOGING-IN-PUBLICATION

Dornelles, Thiago de Azevedo

Online frame-to-model pipeline to 3D reconstruction with depth cameras using RGB-D information / Thiago de Azevedo Dornelles. – Porto Alegre: PPGC da UFRGS, 2020.

70 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Claudio Jung.

1. 3D reconstruction. 2. Visual odometry. 3. RGB-D Cameras. 4. Frame-to-model. I. Jung, Claudio. II. Título.

“Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won’t come in.

— Isaac Asimov

ACKNOWLEDGMENTS

First of all, I would like to thank my dear parents, everything so far has only been possible thanks to their enormous love in my life.

To my future wife, always my girlfriend, for the resilient patience and support in the most stressful and difficult hours of the days. You made the challenges a little calmer with a hug.

To Professor Dr. Claudio for always being so helpful with providing good insights and for his interest in the knowledge and techniques related to this work.

To my closest friends who always invited me for a beer and I often had to refuse.

To the public university and its workers who carry out most of the scientific research produced in Brazil, despite many difficulties.

This master thesis is for all of you and me.

ABSTRACT

Several challenges in computer vision and robotics involve developing algorithms capable of using partial spatial information to generate a reliable 3D perception of the world. Various breakthrough applicable technologies such as Mixed Reality, Autonomous Robotics, Autonomous Driving, Reverse Engineering, 3D Printing, among others, depend on this research topic to move forward. In order to implement a complete application to build 3D reconstructions for individual objects, this master's thesis presents an online pipeline for incremental 3D reconstruction and 6-DoF camera pose estimation based on colored point clouds captured by consumer RGB-D cameras. The proposed approach combines geometric and photometric matching of data provided by both depth and color sensors through an adaptive weighting scheme that copes with eventual misalignment errors between RGB and depth data. Our experimental results indicate that the 3D reconstructions achieved by the proposed scheme are visually better than competitive approaches.

Keywords: 3D reconstruction. visual odometry. RGB-D Cameras. frame-to-model.

Modelo online para reconstrução 3D usando informação RGB-D

RESUMO

Muitos dos desafios da visão computacional e da robótica envolvem o desenvolvimento de algoritmos capazes de usar informação espacial parcial para a geração de uma percepção 3D confiável do mundo. Várias tecnologias inovadoras aplicadas como Realidade Mista, Robótica Autônoma, Veículos Autônomos, Engenharia Reversa, Impressão 3D, entre outras, dependem desta área de pesquisa para seguirem evoluindo. Para implementar uma aplicação completa, que seja capaz de realizar reconstruções 3D para objetos individualmente, esta dissertação apresenta um *pipeline* de reconstrução 3D incremental e de estimativa de pose de câmera baseado em nuvens de pontos coloridas capturadas por câmeras RGB-D de uso doméstico. A abordagem proposta combina casamento geométrico e fotométrico de dados fornecidos pelos sensores de profundidade e cor através de um esquema adaptativo de ponderação que lida com eventuais desalinhamentos entre os dados RGB e de profundidade. Os resultados experimentais indicam que as reconstruções 3D obtidas com esquema proposto são visualmente melhores que a abordagem competidora.

Palavras-chave: Reconstrução 3D, odometria visual, câmeras RGB-D, frame-to-model.

RESUMO EXPANDIDO

Muitos dos desafios da visão computacional e da robótica envolvem o desenvolvimento de algoritmos capazes de usar informação espacial parcial para a geração de uma percepção 3D confiável do mundo. Várias tecnologias inovadoras aplicadas como Realidade Mista, Robótica Autônoma, Veículos Autônomos, Engenharia Reversa, Impressão 3D, entre outras, dependem desta área de pesquisa para seguirem evoluindo. Para implementar uma aplicação completa, que seja capaz de realizar reconstruções 3D para objetos individualmente, esta dissertação apresenta um *pipeline* de reconstrução 3D incremental e de estimativa de pose de câmera baseado em nuvens de pontos coloridas capturadas por câmeras RGB-D de uso doméstico. A abordagem proposta combina casamento geométrico e fotométrico de dados fornecidos pelos sensores de profundidade e cor através de um esquema adaptativo de ponderação que lida com eventuais desalinhamentos entre os dados RGB e de profundidade.

No Capítulo 2, apresentamos os fundamentos teóricos envolvidos em cada etapa do pipeline. Explicamos a matemática utilizada na modelagem de uma câmera *pinhole*, o papel da álgebra de Lie na busca de soluções no espaço de transformações rígidas, o funcionamento do algoritmo de mínimos quadrados não-linear empregado na otimização do problema associado ao alinhamento das nuvens de pontos, apresentamos as diferentes abordagens sobre o problema *Perpective-n-Point* e a obtenção de estimativas robustas contra outliers utilizando o método RANSAC e finalmente conceituamos odometria visual.

Posteriormente, no Capítulo 3, apresentamos os trabalhos relacionados à área de reconstrução 3D e suas aplicações no contexto da implementação proposta. Apresentamos uma divisão de classes de técnica de reconstrução entre técnicas locais e técnicas globais e explicamos como este trabalho se localiza na classe das técnicas locais. Também revisamos técnicas que usam nuvens de pontos com e sem informação de cor associada.

A técnica proposta nesta dissertação é apresentada no Capítulo 4. O procedimento de reconstrução 3D proposto permite que o usuário reconstrua um objeto a partir de imagens obtidas sequencialmente de uma câmera RGB-D qualquer. Alinhamos as nuvens de pontos coloridas em um modelo do objeto que vai sendo consolidado incrementalmente, conforme novos pontos de vista do objeto vão sendo capturados.

Inicialmente filtramos as imagens a fim de remover possíveis ruídos e melhorar o desempenho do alinhamento fotométrico utilizando de um filtro bilateral. Para obter

uma boa estimativa de pose de camera inicial para o sistema não-linear utilizado neste trabalho, utilizamos o algoritmo EPnP dentro de um método RANSAC, a fim de descartar features ORB que tenham casamentos espúrios. Este casamentos de features ORB são feitos entre cada dois pares de frames consecutivos das imagens de entrada da câmera.

A partir da estimativa de pose inicial refinamos o alinhamento das nuvens de pontos minimizando uma função de erro que leva em conta tanto a função de erro fotométrico quando o erro geométrico gerado pelo alinhamento. Estas duas funções de erro são combinadas em uma função que nomeamos de função de erro combinado. O erro combinado usa um fator de ponderação que busca tanto aproximar a posição dos pontos casados no espaço 3D quanto considera as semelhanças entre as intensidades de cores que estes possuem. Essa característica permite que o algoritmo seja mais robusto em situações onde o objeto reconstruído possui pouca informação discriminativa de cor, como por exemplo um objeto de cor única, ou com pouca informação discriminativa geométrica, como por exemplo uma esfera.

O erro combinado é minimizado no espaço dos 6 graus de liberdade permitindo a reconstrução com liberdade total de movimento da câmera (translação e rotação). A função de erro combinado é minimizada mudando os parâmetros do movimento de câmera de forma iterativa com o algoritmo de Gauss-Newton. Para diminuirmos o espaço de parâmetros necessários para minimização utilizamos o que chamamos de coordenadas exponenciais e para isso utilizamos uma aproximação do espaço das rotações usando a álgebra de Lie. As transformações finais no espaço de transformações rígidas são obtidas pela exponenciação de matrizes.

O somatório das diferenças pixel a pixel entre os frames consecutivos são ponderadas por uma função que leva em consideração a confiabilidade da informação geométrica obtida pela câmera, minimizando ou mesmo removendo a influência de pixels de profundidade errôneos. O modelo é incrementado com novas vistas a cada frame de entrada utilizando uma estrutura de dados volumétricos chamada TSDF (*Truncated Signed Distance Function*) o que permite criar uma superfície de malha de triângulos do objeto sempre atualizada a cada imagem nova obtida. O modelo é renderizado e seu buffer de profundidade é utilizado como a imagem de profundidade anterior à mais recentemente obtida pela câmera. Desta forma o modelo serve como referência no processo de alinhamento com a nova nuvem de pontos que chega da câmera o que chamamos de abordagem frame-to-model.

A avaliação experimental da técnica proposta é descrita no Capítulo 5. Foram

feitas análises quantitativas utilizando métricas de trajetória de câmera tradicionais como *Absolute Trajectory Error* (ATE) e *Relative Pose Error* (RPE), bem como métrica sobre o modelo final gerado chamada de *Signed Cloud to Mesh Difference* (C2M). Ainda foi feita uma análise qualitativa que buscou evidenciar as diferenças entre os modelos de objetos reconstruídos com a técnica proposta e com a técnica concorrente.

Nossos resultados experimentais indicaram que as reconstruções 3D obtidas pelo esquema proposto são visualmente melhores que a implementação que usamos como padrão de comparação. Adicionalmente, verificamos que as métricas de odometria visual também são melhores utilizando a abordagem proposta, especialmente no contexto de cenas capturadas com a câmera executando uma trajetória mais errática, como por exemplo quando esta é carregada pelas mãos do operador.

Finalmente, o Capítulo 6 apresenta as conclusões desta dissertação de mestrado. O capítulo também apresenta problemas em aberto e possíveis direções para trabalhos futuros.

LIST OF ABBREVIATIONS AND ACRONYMS

2D	2-Dimensional
3D	3-Dimensional
6-DoF	6 Degrees of Freedom
ATE	Absolute Trajectory Error
BRIEF	Binary Robust Independent Elementary Features
C2M	Cloud to Mesh Distance
CPU	Central Processing Unit
ePnP	Efficient Perspective-n-Point
FPFH	Fast Point Feature Histograms
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
NICP	Normal Iterative Closest Point
ORB	Oriented FAST and rotated BRIEF
PFH	Point Feature Histograms
PnP	Perspective-n-Point
RANSAC	RANdom SAmple Consensus
RGB	Red, Green and Blue
RGB-D	Red, Green, Blue and Depth
RPE	Relative Pose Error
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SfM	Structure from Motion
SHOT	Signature of Histograms of OrienTations
ToF	Time of Flight
TSDF	Truncate Signed Distance Function

LIST OF FIGURES

LIST OF TABLES

CONTENTS

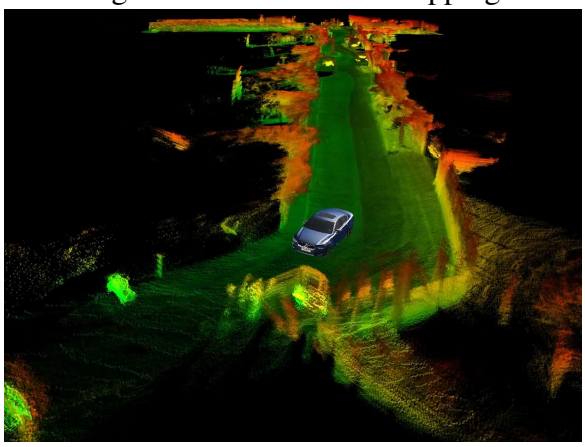
1 INTRODUCTION

1.1 Motivation

In computer vision, 3D reconstruction is the process of estimating the 3D structure (shape and appearance) of objects. This research topic is the core of many applications that use 3D spatial information, such as Mixed Reality (NEWCOMBE et al., 2011; IZADI et al., 2011; Rematas et al., 2018), Autonomous Robotics (Mur-Artal; Montiel; Tardós, 2015; ENGEL; SCHÖPS; CREMERS, 2014), Reverse Engineering (NEWCOMBE et al., 2011; KERL; STURM; CREMERS, 2013), and Additive Manufacturing (STURM et al., 2013).

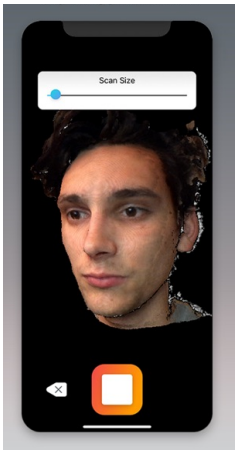
Many good examples of commercial applications that involve 3D reconstruction are already applied in our daily lives. For example, we can cite some autonomous vehicles, such as some BMW models (Figure 1.1) or reconstruction apps to smartphones reconstruct small 3D objects, such as Scandy (Figure 1.2), or large-scale 3D reconstruction, such as Matterport solutions (Figure 1.3).

Figure 1.1: BMW lidar mapping



Source:<https://spectrum.ieee.org/cars-that-think/transportation/sensors/embargoed-until-7am-edt-thursday-25-april-bmw-will-use-solidstate-lidar-from-innoviz>

Figure 1.2: Scandy Pro App



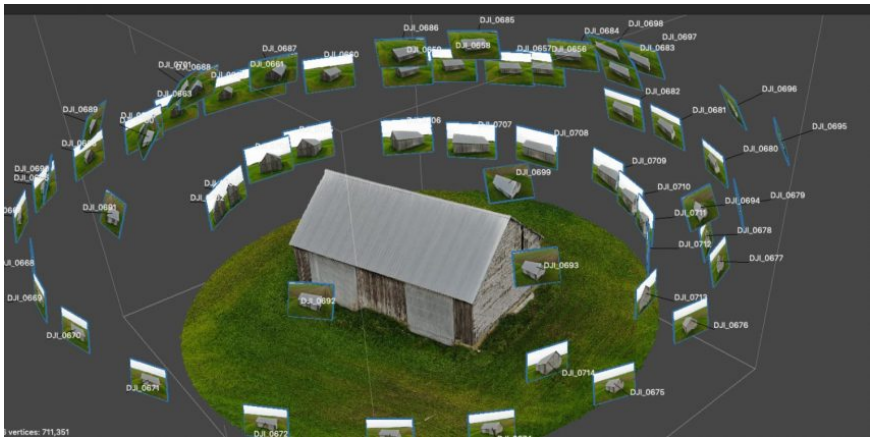
Source: <https://prmac.com/release/82969>

Figure 1.3: Matterport Scanning Example

Source: <https://matterport.com/industries/gallery/insta360-vacation-rental-corfe-castle-purbecks>

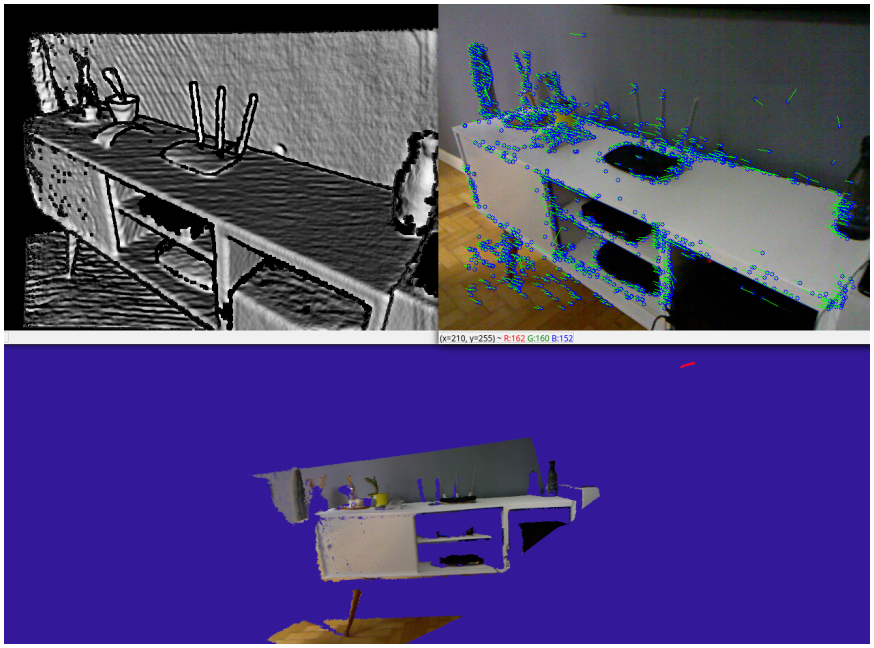
Several techniques can be used to obtain a 3D model of an object, such as set of color captures of the object from different views (which is the core of Structure-from-Motion – SfM – in Computer Vision, as illustrated in Figure 1.4, or a set of point clouds acquired from different views (using a laser scan or time of flight camera, for instance), as shown in Figure 1.5 .

Figure 1.4: Structure from motion based reconstruction



Source:<https://www.fxphd.com/fxblog/new-course-photogrammetry-with-metashape-and-nuke>

Figure 1.5: RGB-D 3D Reconstruction



Source: Authors

Nowadays, we have a large availability of low-cost RGB-D sensors, making 3D scanning software a popular application for the maker culture. These RGB-D sensors capture simultaneously color (RBG) and depth (D) information, generating a colored 3D point cloud as output. Although using color as an additional cue to depth information

might seem straightforward, there might be some complicating factors. In general, color and depth data are captured by different sensors (sometimes with different Field-of-Views – FoVs) that are aligned to generate a colored point cloud. This alignment is based on the intrinsic and extrinsic parameters of the two sensors, and small errors might produce mis-aligned colored point clouds.

Another issue related to RGB-D cameras is that some of these sensors focus on near-range applications, being useful to build natural user interfaces, 3D object scanning and face recognition, for example. On the other hand, other methods are able to capture longer-range depth values, being designed to build applications that needs a wider perception of the environment like autonomous cars, robot navigation and scene reconstruction, to name a few applications. Although the mathematical formulation for shorter- or longer-range applications are the same, the nature of captured scenes are considerably different. For example, algorithms for 3D alignment that use mid-range sensors (2 to 4 meters) focus on the reconstruction of larger scenes (e.g. full indoor environments), such as (DAI et al., 2017). In these cases, the sensor is able to capture several 3D objects at different locations and orientations (e.g. walls, chairs, tables), so that many point and depth correspondences across different frames can be obtained. Also, the full FoV of the sensor might contain scene objects, and it was recently shown in (SILVEIRA; JUNG, 2019) that using features that are spatially spread tends to produce more accurate results in the context of epipolar matrix estimation (and hence, pose estimation). On the other hand, scanning smaller objects requires near-range sensors (0.5 meters to 1 meter) to obtain enough geometric and textural details. In this scenario, the captured image contains mostly the object (typically in the central part of the camera FoV) and some background, leading to limited geometry/color/texture variability. As a consequence, finding correspondences across different views is a more challenging task, which compromises both camera pose estimation and 3D reconstruction.

A common problem to both short- or long-range applications is the temporal continuity of the input data. For instance, depth information can suffer variations even when no motion has been applied to the camera with respect to the object due to sensor noise or errors in pose estimation. When using frame-to-frame alignments, this temporal discontinuity can lead to worse alignments because of the influence of noise or outlier measurements. In frame-to-model schemes, depth data is accumulated and filtered in order to generate a smooth model that cope with noise and outliers in depth data.

1.2 Goals and contributions

Our main goal in this work is to develop a suitable pipeline to process sequences of RGB-D images in order to reconstruct 3D models of rigid objects in near range using RGB-D cameras like Kinect. To obtain a 3D model, the user captures images around the object continuously, and the object is reconstructed online while the user records the images. To improve the results we address a set possible sources of alignment problems when 3D scanning is performed: noise in the input images, large displacements between adjacent camera poses, and objects with low descriptive geometric or texture information.

1.2.1 Contributions

The main contributions proposed in this thesis are:

- Development of a complete frame-to-model pipeline that uses combined color and depth information, obtaining low drift estimates;
- Proposal of a coarse initial alignment using a Perspective-n-Point (PnP) approach to cope with large displacement between adjacent frames;
- Introduction of a weighting scheme to avoid alignment problems when the depth image have pixels with low confidence, which relates to depth discontinuities (and affects the alignment with the color sensor).

1.3 Chapters organization

In Chapter 2 we introduce and reference some of the fundamental concepts of camera modeling and mathematical optimization techniques that are extensively used in this work. chapter 3 revises important papers in the 3D reconstruction area, these works are divided by techniques that rely on local minimization techniques and global minimization techniques. At the end of chapter is introduced the competitor technique to this work.

In chapter 4 we describe the pipeline used to obtain robust alignment of continuous incoming RGB-D data from a depth camera. All stages of the pipeline involving image filtering, initial pose estimation, pose refinement, and model integration.

The experimental results are presented in Chapter 5, where we also define the used

datasets, as well as the qualitative and quantitative metrics to assess the performance of the system in both camera pose estimation and final model geometric/appearance fidelity. Finally, Chapter 6 presents the final conclusions and directions for future work.

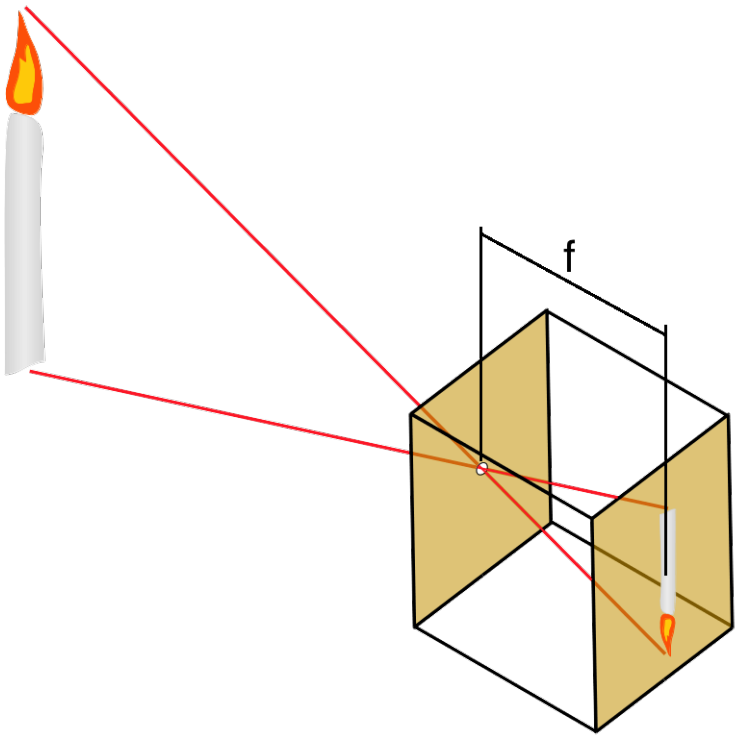
2 FUNDAMENTALS

This chapter presents mathematical/geometrical concepts and underlying numeric techniques necessary to understand the implementation of the algorithm to solve the 3D reconstruction problem.

2.1 Pinhole Camera Model

The acquisition of images by a standard camera can be modeled as the Pinhole Camera Model described in the work of Hartley and Zisserman (2000). This model describes a camera based on a box that presents a very small aperture in the center of one of its sides. The rays of light pass through the aperture and are projected at the opposite side of the box. By being an abstraction to a real camera this model ignores geometric distortion and other effects, with the advantage of being easier to understand and work.

Figure 2.1: A Pinhole Camera diagram



Source: The authors

The process of transforming a point in the world coordinate frame to the image plane is described by the camera matrix. Camera matrix is composed of two other matrices: Intrinsics camera parameters matrix or camera calibration matrix; and Extrinsics

camera parameters matrix.

The intrinsics matrix contains parameters concerning to focal length and principal point, and for digital cameras it also encodes information about the discrete geometry of pixels. The extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ describes the transformation between 3D world coordinates and camera coordinates (rotation and translation).

2.1.1 Intrinsic parameters

As noted in Figure 2.1, the projection of a 3D object based on the pinhole camera model generates similar triangles, from which we can devise the projection equations. These projections are encoded by a matrix called calibration matrix or intrinsics camera parameters matrix. This is a 3×3 upper-triangular matrix given by

$$\mathbf{K}_p = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

where f is the focal length (distance between the image plane and aperture), p_x and p_y are the principal point coordinates (central coordinates where a ray perpendicular to the image plane converges).

In a digital camera, the photosensitive sensor is discrete (leading the the notion of *pixels*). As such, it is modeled by an extension of the pinhole camera model called "Finite Projective Camera Model" that also takes into account the discrete sensor. It is given by

$$\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where α_x and α_y are the focal lengths in both spatial axis (given in pixels), and (u_0, v_0) are the principal point coordinates in pixels. Parameter γ represents the skew coefficient between the x and the y axis, which often is 0.

2.1.2 Extrinsic parameters

The extrinsic parameters matrix stores the coordinate system transformations from the 3D world coordinates to 3D camera coordinates. It also encodes the position of the camera center and the camera's direction in world coordinates. The extrinsic parameters are fundamental in visual odometry algorithms, since we can find the camera position in world coordinates $\mathbf{C}$ as $\mathbf{C} = -\mathbf{R}^{-1}\mathbf{t} = -\mathbf{R}^{\top}\mathbf{t}$, noting that $\mathbf{R}$ is a rotation matrix (and hence, orthogonal). The extrinsic parameters matrix is typically represented in the form below

$$\Big[\mathbf{R}|\mathbf{t}\Big] = \Big[\mathbf{R}_{3\times3} \quad \mathbf{t}_{3\times1}\Big] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right] \tag{2.3}$$

2.1.3 Central Projection Using Homogeneous Coordinates

A convenient way to transform 3D points from world coordinates to camera coordinates and after to image plane coordinates is using homogeneous coordinates.

A point in world coordinates $[x_w, y_w, z_w, 1]^{\top}$ in the homogeneous form is transformed to the camera coordinate space by the multiplication with the extrinsics matrix $[\mathbf{R}|\mathbf{t}]$ (2.3). After that, the point – now in the camera space – is projected into a image plane by multiplication with the intrinsics matrix $\mathbf{K}$ (2.2). The process of mapping from 3D space to camera image plane space is summarized as

$$\begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix} \sim \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{K}\Big[\mathbf{R}|\mathbf{t}\Big] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \tag{2.4}$$

where $[u/w, w/v]^{\top}$ are the final image coordinates expressed in pixel units, and $\sim$ denotes that the two points are in the same line of projection.

2.2 Rigid Body Motion using Lie Algebra of Twists

A twist in its vector form (exponential coordinates) is are very useful because they can represent a rigid body motion matrix $\mathbf{T} \in$ SE(3) using a minimal vector representation $\boldsymbol{\xi} \in \Re^6$ in exponential coordinates. It is given by

$$\boldsymbol{\xi} = (\xi_1, ..., \xi_6)^\top = (u_1, u_2, u_3, \omega_1, \omega_2, \omega_3)^\top = (\mathbf{u}^\top, \boldsymbol{\omega}^\top)^\top, \tag{2.5}$$

where $\mathbf{u} \in \Re^3$ and $\boldsymbol{\omega} \in \Re^3$ represent the translational and rotational velocities, respectively.

Let us also define the operator $[.]_x$ given by

$$[.]_x : \Re^3 \rightarrow \text{so(3)} \subset \Re^{3\times3}; [\boldsymbol{\omega}]_x = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \tag{2.6}$$

which operates on a 3D vector and produces a 3×3 matrix that corresponds to the vectorial cross-product, i.e., $\mathbf{p} \times \mathbf{q} = [\mathbf{p}]_x \mathbf{q}$.

Let us also define the hat operator ($\wedge$) that combines rotation and translation parameters into an intermediate matrix form, which can be mapped to SE(3) Lie group matrix:

$$\wedge : \boldsymbol{\Re}^6 \rightarrow \text{se(3)} \subset \Re^{4\times4};\ \hat{\boldsymbol{\xi}} = \boldsymbol{\xi}^\wedge = \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\omega} \end{bmatrix}^\wedge = \begin{bmatrix} [\boldsymbol{\omega}]_x & \mathbf{u} \\ 0 & 0 \end{bmatrix} \tag{2.7}$$

The mapping between the twist skew symmetric matrix in se(3) and the SE(3) Lie group is done by matrix exponentiation:

$$\exp : \text{se(3)} \rightarrow \text{SE(3)};\ \exp(\hat{\boldsymbol{\xi}}) = \exp\left(\begin{bmatrix} [\boldsymbol{\omega}]_x & \mathbf{u} \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} e^{[\boldsymbol{\omega}]_x} & \mathbf{V}\mathbf{u} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \tag{2.8}$$

where the exponential of a square matrix is defined as the Taylor series expansion applied to the matrix. In fact, we can obtain a closed-form for $e^{[\boldsymbol{\omega}]_x}$ and $\mathbf{V}$ by simplyfing the Taylor series, which is called the Rodrigues rotation formula:

$$\phi = \|\boldsymbol{\omega}\|; \mathbf{n} = \frac{\boldsymbol{\omega}}{\phi} \tag{2.9}$$

$$e^{[\boldsymbol{\omega}]_x} = e^{[\boldsymbol{n}]_{x\phi}} = \mathbf{I} + \sin\phi[\mathbf{n}]_x + (1-\cos\phi)[\mathbf{n}]_x^2 \tag{2.10}$$

$$\mathbf{V} = \mathbf{I} + \frac{1-\cos\phi}{\phi}[\mathbf{n}]_x + \frac{\phi-\sin\phi}{\phi}[\mathbf{n}]_x^2 \tag{2.11}$$

These operators are applied in order as below:

$$\boldsymbol{\xi} \in \Re^6 \underset{\vee(vee)}{\overset{\wedge(hat)}{\rightleftarrows}} \hat{\boldsymbol{\xi}} \in se(3) \underset{\log}{\overset{\exp}{\rightleftarrows}} \mathbf{T} \in \text{SE(3)}, \tag{2.12}$$

and the backward order is possible by using the inverse operators $\log$ and $\vee$ operator. These operators are not used in this work as we only need to apply these transformations in the forward direction.

Note that for 3D reconstruction and pose estimation we need to obtain the transformation parameters (rotation, translation) based on observed data. Since these parameters appear in a non-linear form, linearization is typically applied. One important feature of the motion parametrization based on twist coordinates is that the Jacobian of twists of each 3D point $\mathbf{X}$ is very simple to compute, as it does not involve any trigonometric operations and depends only on the point coordinates $\mathbf{x_i}$ as provided next:

$$J_\xi = \frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} = \begin{bmatrix} 1 & 0 & 0 & 0 & x_3 & -x_2 \\ 0 & 1 & 0 & -x_3 & 0 & x_1 \\ 0 & 0 & 1 & x_2 & -x_1 & 0 \end{bmatrix} \tag{2.13}$$

2.3 Non-Linear Least squares estimation

Least squares estimation is a common technique to compute approximated solutions of overdetermined systems. There are two classes of least-squares problems: Linear and non-linear. In general, most of the problems have non-linear formulations, and thus non-linear estimation algorithms are useful in a wider set of problems.

The technique aims to minimize the sum S of the squares of the residual terms r_i:

$$S = \sum_{i=1}^{n} {r_i}^2, \tag{2.14}$$

where residual term r_i is given by

$$r_i = y_i - f(x_i, \boldsymbol{\beta}). \tag{2.15}$$

In Equation (2.15), x_i and y_i are the observed values for the independent and dependent variables, respectively. Also, f is a fitting function that depends on a set of parameters given by vector $\boldsymbol{\beta}$, such that y_i should be close to $f(x_i, \boldsymbol{\beta})$.

Assuming that S is convex, its minimum is obtained by setting the n gradient equations to zero:

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial \boldsymbol{\beta}_j} = 0, \text{ for } j = 1, \ldots, n, \tag{2.16}$$

which leads to a linear system when f is linear w.r.t. $\boldsymbol{\beta}$. For non-linear relationships, Equation (2.16) leads to system of non-linear equations, and S is typically minimized using an iterative numerical scheme.

Most of non-linear formulations strongly depend on good initial values for parameters, is this case "good" means parameters close enough to the global optimal solution. Assuming that we have a good initial state $\boldsymbol{\beta}_j^0$ the parameters are refined iteratively by the successive the approximation rule:

$$\boldsymbol{\beta}_j^{k+1} = \boldsymbol{\beta}_j^k + \Delta \boldsymbol{\beta}_j, \text{ for } k = 1, 2, 3, \ldots \tag{2.17}$$

To find the shift vector $\Delta \boldsymbol{\beta}_j$ at each iteration, the model is linearized by a first-order Taylor expansion about $\boldsymbol{\beta}^k$ where k is the current iteration number:

$$f(x_i, \boldsymbol{\beta}) \approx f(x_i, \boldsymbol{\beta}^k) + \sum_j \frac{\partial f(x_i, \boldsymbol{\beta}^k)}{\partial \boldsymbol{\beta}_j} \left(\boldsymbol{\beta}_j - \boldsymbol{\beta}_j^k \right) = \tag{2.18}$$

$$f(x_i, \boldsymbol{\beta}^k) + \sum_j \mathbf{J}_{ij} \, \Delta \boldsymbol{\beta}_j, \tag{2.19}$$

noting that the jacobian $\mathbf{J}$ changes at each iteration. If we define

$$\Delta y_i = y_i - f(x_i, \boldsymbol{\beta}^k), \tag{2.20}$$

we can write

$$r_i = y_i - f(x_i, \boldsymbol{\beta}) = \big(y_i - f(x_i, \boldsymbol{\beta}^k)\big) + \big(f(x_i, \boldsymbol{\beta}^k) - f(x_i, \boldsymbol{\beta})\big) \approx \Delta y_i - \sum_{s=1}^{n} J_{is} \Delta \beta_s. \tag{2.21}$$

Substituting the above equations into (2.16) yields:

$$-2 \sum_{i=1}^{m} \mathbf{J}_{ij} \left(\Delta y_i - \sum_{s=1}^{n} \mathbf{J}_{is} \, \Delta \boldsymbol{\beta}_s \right) = 0, \tag{2.22}$$

which can be reorganized to get the n normal equations:

$$\sum_{i=1}^{m} \sum_{s=1}^{n} \mathbf{J}_{ij} \mathbf{J}_{is} \, \Delta \boldsymbol{\beta}_s = \sum_{i=1}^{m} \mathbf{J}_{ij} \, \Delta y_i \qquad \text{for } j = 1, \ldots, n, \tag{2.23}$$

and finally can be expressed in matrix notation as:

$$\left(\mathbf{J}^\top \mathbf{J}\right) \boldsymbol{\Delta\beta} = \mathbf{J}^\top \boldsymbol{\Delta y}. \tag{2.24}$$

A weighted version of the sum of the squared could be used in the case different confidence levels of the parameters of the model is presented below:

$$S = \sum_{i=1}^{m} W_{ii} r_i^2, \tag{2.25}$$

and the corresponding normal equations in this case are:

$$\left(\mathbf{J}^\top \mathbf{W} \mathbf{J}\right) \boldsymbol{\Delta\beta} = \mathbf{J}^\top \mathbf{W} \boldsymbol{\Delta y}, \tag{2.26}$$

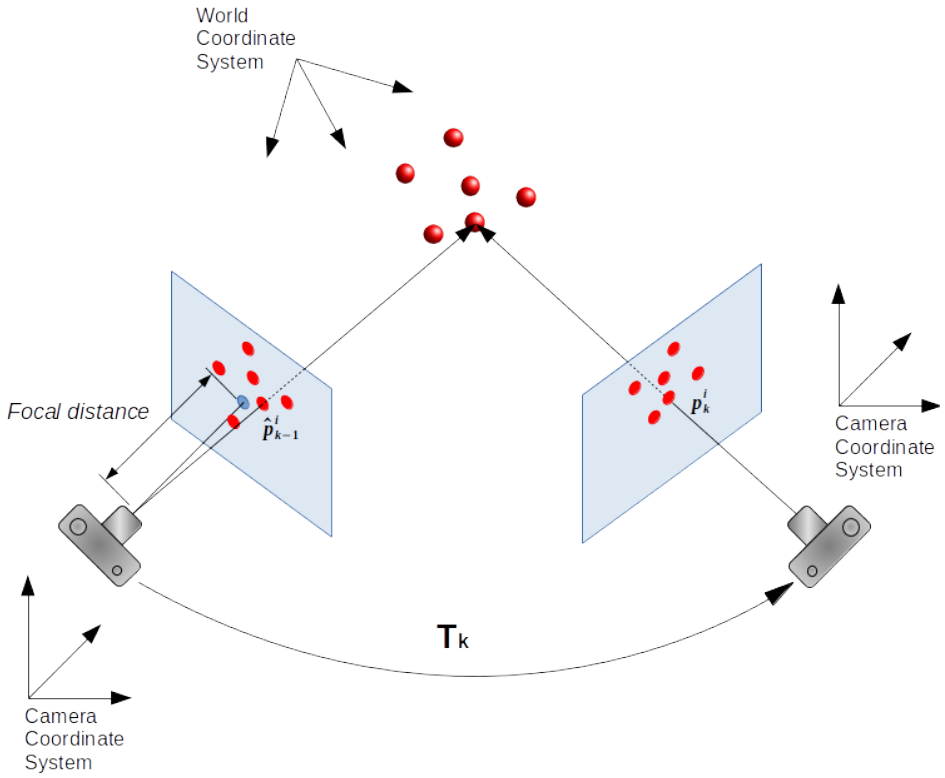
where $\mathbf{W}$ is a diagonal matrix with the weighting values for each parameter.

2.4 Perspective-n-Point

Perspective-n-Point is well-known problem that aims to recover the relative camera pose between a calibrated camera (i.e., intrinsic parameters are known) and the origin of a certain coordinate system, given a set of n points in 3D and their corresponding 2D projections in the image (MARCHAND; UCHIYAMA; SPINDLER, 2016). As mentioned before, the camera pose presents six degrees-of-freedom (6-DoF), which are re-

lated to the translational (x,y,z) and the rotational motion (roll, pitch, yaw). These 6-DoF parameters describes the relative motion (World coordinates to Camera coordinates) between two points of view of the same scene as shown in Figure 2.2.

Figure 2.2: Pose transform between two points of view of the same point cloud



Source: The authors

Although the Direct Linear Transformation (DLT) is the straightforward solution for the case where the number of points is $n > 5$ (HARTLEY; ZISSERMAN, 2000) many authors proposed solutions to this error function by using fewer points, for example, $n = 3$ which is called the P3P problem. Different algorithms were developed to solve the P3P problem as for example the work of Gao et al. (2003) and even for the cases when $n \geqslant 4$ as ePnP (LEPETIT; MORENO-NOGUER; FUA, 2009).

The formulation considered the "gold standard" error for PNP problem (HARTLEY; ZISSERMAN, 2000; MARCHAND; UCHIYAMA; SPINDLER, 2016) consists on finding the 6-DoF transformation between the camera and world frames ${}^c\mathbf{T}_w \in \mathrm{SE}(3)$ that minimizes the Euclidean distance error of the projected points, as follows:

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}} \sum_{i=1}^{N} d(\mathbf{x}_i, \mathbf{\Pi}\, \mathbf{T}_w\, {}^w\mathbf{X}_i)^2. \tag{2.27}$$

In equation (2.27) the motion is denoted by $\mathbf{q} \in$ se(3) as $\mathbf{q} = [{}^{c}\mathbf{t}_w, \theta\mathbf{u}]^\top$ where θ and $\mathbf{u}$ are the angle and the axis of rotation and ${}^{c}\mathbf{t}_w$ represents the translation motion. The error is obtained by calculating the Euclidean distance between 2D point $\mathbf{x}_i$ and the 3D point $\mathbf{X}_i$ transformed and projected by $\mathbf{T}_w$ and $\mathbf{\Pi}$ respectively.

The solution of the problem relies on iterative methods like Gauss-Newton algorithm. An exponential mapping of $\mathbf{q}$ is applied to obtain the ${}^{c}\mathbf{T}_w$ transformation (MA et al., 2003).

2.5 RANSAC - Random Sample Consensus

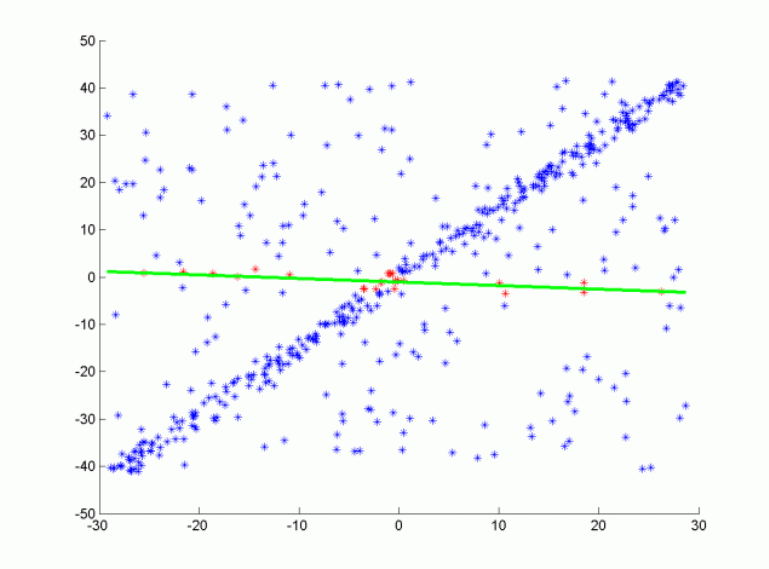
RANdom SAmple Consensus, or RANSAC, is an iterative method to estimate parameters of mathematical models when the data contains outliers, where an outlier is a point that is "different" from the expected observations. In the case of images, outliers can be erroneous pixels generated by the various sources of noise in the digital photography process, or a wrong depth estimate due to object characteristics (e.g., dark objects do not reflect much light, which affect time-of-flight cameras) or illumination conditions.

A classical application of RANSAC is line fitting based on a set of roughly aligned 2D points containing outliers. In this case, the result obtained with conventional methods like Linear Regression will be affected by the most distant points from the expected observation, since the squared error is large at outliers. An example of (wrong) linear regression is shown in Figure 2.3, while a desirable result would the case is shown in figure 2.4.

To avoid outlier influence in the final result, RANSAC chooses randomly n data points in order to estimate the desired model, then it checks how many points of the full dataset are well approximated by this guess. If the random selection leads to "good" points, the model will be adequate and the other points will be well explained by the fitted model. Otherwise, the process keeps going until it reaches a maximum number of iterations or it provides a good enough model. A general explanation is shown in Algorithm 1.

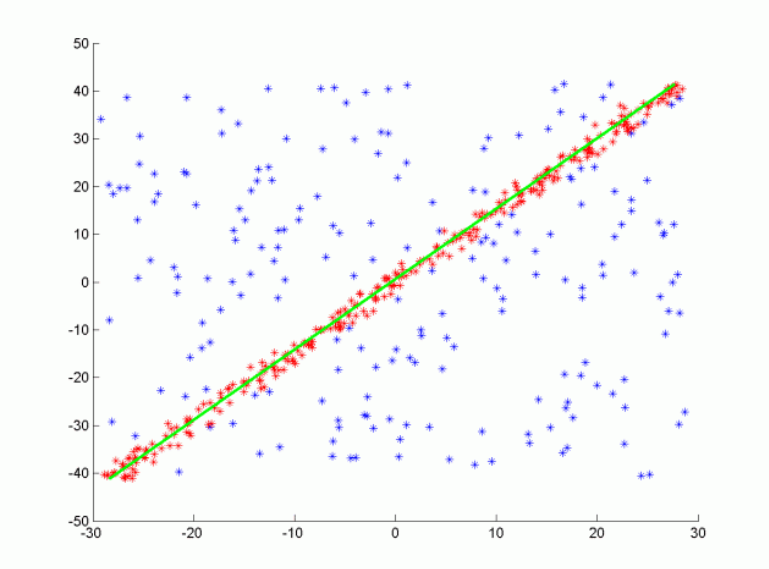
To avoid problems with noisy estimates of depth and color RANSAC when using ePNP we need to put ePNP estimates inside of RANSAC iterative process as the fitting model step of the algorithm. Fortunately we have this already implemented in the OpenCV Library (BRADSKI, 2000).

Figure 2.3: Bad linear model estimate



Public Domain Image - Retrieved from <https://commons.wikimedia.org/wiki/File:RANSAC_LINIE_Animiert.gif>

Figure 2.4: Good linear model estimate



Public Domain Image - Retrieved from <https://commons.wikimedia.org/wiki/File:RANSAC_LINIE_Animiert.gif>

2.6 Visual Odometry

Visual odometry is the process of determining the pose and orientation of a moving camera based on visual information. This is a key task in robotics, and it is useful to keep track of the robot's position and orientation in time. A famous example of application to this technique is the Mars Exploration Rover, since it operates in a GPS denied environment (MAIMONE; CHENG; MATTHIES, 2007). For more information on visual

Algorithm 1: Random Sample Consensus

```
Data: data – A set of observations.
model – A model to explain observed data points.
n – Minimum number of data points required to estimate model parameters.
k – Maximum number of iterations allowed in the algorithm.
t – Threshold value to determine data points that are fit well by model.
d – Number of close data points required to assert that a model fits well to
data.
Result: bestFit – model parameters which best fit the data (or nul if no good
        model is found)
iterations := 0;
bestFit := nul;
bestErr := something really large;
while iterations < k do
    maybeInliers := n randomly selected values from data;
    maybeModel := model parameters fitted to maybeInliers;
    alsoInliers := empty set;
    for every point in data not in maybeInliers do
        if point fits maybeModel with an error smaller than t then
            add point to alsoInliers;
        end
    end
    if the number of elements in alsoInliers is > d then
        betterModel := model parameters fitted to all points in maybeInliers
         and alsoInliers;
        thisErr := a measure of how well betterModel fits these points;
        if thisErr < bestErr then
            bestFit := betterModel;
            bestErr := thisErr;
        end
    end
    iterations++;
end
```

odometry we redirect the reador to the work of Scaramuzza and Fraundorfer (2011), who introduce the main concepts, present and analyzes different algorithms.

2.7 Conclusions of Chapter

This chapter presented the main concepts and basic techniques typically used for 3D camera pose estimation and 3D reconstruction. It indicates that there are several challenges and steps involved in the aforementioned problems, and some of them will be tackled in the next chapters.

3 RELATED WORK

Various approaches have been proposed to address the 3D reconstruction problem using RGB-D cameras. Some of them depend on reliable point matching between point clouds to estimate a rigid transformation that describes camera movement between these frames, others do not need explicit correspondences but work well only if the points clouds are close to each other. This review will focus on methods that explore point-cloud data (with or without associated color information). To explain how a 3D model can be generated from point cloud data we direct the reader to (BERGER et al., 2017) for a survey on surface reconstruction from point clouds.

3.1 Global Registration

Global registration techniques aims to find the transformation matrix that aligns two point clouds without any assumption on the initial pose of the cameras that captured the point clouds. There are two main approaches to this end: Voting Methods and Geometric Descriptors.

In the case of voting method algorithms, an exhaustive search is performed by varying the rigid transformation parameters. To speed up the process, many techniques were developed like Generalized 3D Hough Transform (KHOSHELHAM, 2007) and Pose Clustering (OLSON, 1999). As described in the work of Gelfand et al. (2005), although voting methods can guarantee to find optimal solutions, these techniques have a very high computational cost making implementation impractical to be used in interactive applications.

Geometric (RUSU et al., 2008; TOMBARI; SALTI; STEFANO, 2010) and image (RUBLEE et al., 2011; LOWE, 2004; BAY et al., 2008; CALONDER et al., 2010) descriptors can be used to establish sufficiently reliable correspondences between point clouds or images, aiming to obtain acceptable results at lower running times. By finding these correspondences we can estimate the transformation that aligns point clouds by using various algorithms as SAC-IA (RUSU; BLODOW; BEETZ, 2009), 3D-NDT (MAGNUSSON, 2009) or Extended Gaussian Images (MAKADIA A. PATTERSON, 2006) alone or combining with outlier robustification techniques like RANSAC (FISCHLER; BOLLES, 1981).

Usually, faster global registration algorithms rely on point matching. Errors on

point matching lead to an inaccurate alignment, needing the use of local methods such as ICP (BESL; MCKAY, 1992) to refine the transformations. In spite of this, Zhou, Park and Koltun (2016) claim that their global registration method is even better than ICP.

Next we present some depth or color descriptors that can be used for feature matching, and then possibilities for PnP that explore such matched features to estimate the desired camera pose.

3.1.1 Features and Descriptors

To get correspondences between point clouds, one must detect points of interest that are salient and repeatable in depth/geometry or color in order to generate information that describes these points. This descriptive information is used later to find matchings between point clouds or images.

A well-known descriptor and feature detector to work with geometric data is PFH (RUSU et al., 2008), and its faster version FPFH (RUSU; BLODOW; BEETZ, 2009) is implemented in the PCL Library (RUSU; COUSINS, 2011). When using these types of descriptors, the neighborhood of 3D points is used to generate a signature of the region around each point. Similarly, SHOT features (TOMBARI; SALTI; STEFANO, 2010) is another option of 3D features that offers a faster alternative.

In the case of 2D RGB images, similar approaches like SURF (BAY et al., 2008) and ORB (RUBLEE et al., 2011) are available, with the advantage of allowing lower complexity implementations while keeping good accuracy. The descriptors called LIFT (YI et al., 2016) and SuperPoint (DETONE; MALISIEWICZ; RABINOVICH, 2018) use machine learning techniques to find discriminative keypoints and descriptors, achieving state-of-the-art results. However, the dependency of the training sets used in machine learning approaches and the test images is still a point to be investigated.

As one can observe, an extensive variety of feature detectors and descriptors are in constant development, some of them are less robust to variability in data, while others may try to be invariant to scale, rotation, or affine transformation. In this work, we chose to use ORB, which is fast and showed to be robust enough (has rotational invariance) to obtain the coarse approximations that initialize our pose refinement step of the pipeline. As mentioned above, geometric (Depth) and image features (RGB) work with different characteristics of RGB-D information. Depth-based features can be useful when only homogeneous or repetitive color information is available; on the other hand, color-based

features may be useful when depth data is mostly planar or with low geometric variation, leading to an ambiguous situation to find matching correspondences.

3.1.2 PnP Algorithm Variants

The Perspective-n-Point problem presented in Section 2.4 has many proposed solutions. A common classification the proposed solutions is the minimum number of points n needed to find the pose of camera e.g. $n = 3$ (P3P), $n = 4$ (P4P) and so on.

Several solutions to the P3P problem were proposed (MERRITT, 1988; Xiao-Shan Gao et al., 2003; WANG et al., 2018), most of them is based on the law of cosines to generate three quadric equations about the lengths between the three 3D points and the camera origin (Zhou; Kaess, 2019). A detailed review of the P3P algorithms can be seen in the work of Haralick et al. (1994).

To solve PnP where $n \geqslant 4$, many other solutions were proposed as well (Lu; Hager; Mjolsness, 2000; SCHWEIGHOFER; PINZ, 2008; LI; XU; XIE, 2012; Zhou; Kaess, 2019). The work of Lepetit, Moreno-Noguer and Fua (2009) presents the first $\mathcal{O}(n)$ algorithm to solve PnP. To achieve this, they introduce the notion of control points. More precisely, the authors define four 3D virtual control points that have their coordinates estimated by computing weights of the null space eigenvectors by solving a fixed and small number of quadratic equations. The final transformation is refined obtaining the weights of the control points using Gauss Newton iterative optimization in the error of distances between control points.

3.2 Local Registration

Local registration methods are techniques that need a good initial pose to estimate accurate transformation between point clouds. This limitation is mostly true to all techniques that work optimizing non-linear functions. The most popular technique of this kind is the Iterative Closest Points (ICP) introduced works of Besl and McKay (1992) and Chen and Medioni (1992).

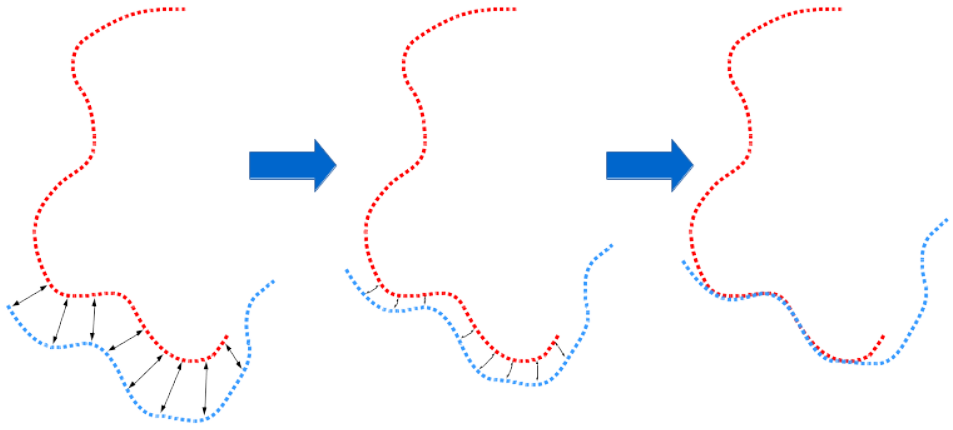
Iterative Closest Points is a very explored algorithm with various variants by the tuning of many steps of the classical implementation. The most conventional types of formulation are the point-to-point distance (CHEN; MEDIONI, 1992; BESL; MCKAY,

1992) and point-to-plane distance (LOW, 2004). An overview of the variants of this class of algorithms is presented in (RUSINKIEWICZ; LEVOY, 2001).

The objective of ICP, as indicated by its name, is to approach the nearest points of two-point clouds by keeping a reference point cloud fixed, while the other one is moved in 6-DoF space to match as much as possible with the reference cloud. To achieve this goal, a generic list of steps is executed:

- Selection of the matching points of interest on each point cloud. Typical choices are: using all or most of the data (dense information), or selecting discriminative sparse points. A measurement of which is closest point to a specific point varies: Euclidian distance, projective association, normal shooting, etc.
- Weighting or rejection of the selected points by confidence measurement: Weighting or rejection of some erroneous point gives robustness to the system.
- Minimization of a chosen error metric: Minimization can be done by classical algorithms like gradient descent, Gauss-Newton or even closed-form solutions depending of the chosen error metric.
- Iterate (repeat all steps above)

Figure 3.1: Point cloud iterative alignment. Black arrows indicate point selection and local motion. Thick blue arrows indicate different iteration steps



Source: The authors

The overall algorithm can be summarized as the loop of these steps above with each new iteration updating the transformation of the new cloud with respect with reference one. A graphical illustration of the method is shown in Figure 3.1.

In the work of Newcombe et al. (2011) called KinectFusion is presented a 3D real-time reconstruction pipeline. The main contributions of their work are the implementation of ICP using the point-to-plane error metric with projective association and a

Truncated Signed Distance Function (TSDF) which continuously smooths the obtained surfaces while running on GPU yielding real-time performance. Since the implementation can perform integration of depth data at fast frame rates, it allows tracking based on the generated map by aligning the incoming point clouds to the smooth accumulated map, this is called frame-to-model alignment.

A more robust ICP variation is the Dense Normal Based Point Cloud Registration (NICP) (SERAFIN; GRISETTI, 2015), which introduces a minimization function that as KinectFusion, uses only geometric data. It generated extended measurements by using normal and tangent information of the points. Its error function formulation aims to minimize the Mahalanobis distance of the extended geometric information together with the usual point distance error. To prune bad correspondences normals, distance and curvature are used to improve the robustness of the system. The system is able to run real-time using only CPU.

Color information is coupled to the ICP point-to-plane cloud alignment problem in the paper of Park, Zhou and Koltun (2017). They use colors converted to the grayscale for solving a photometric error together with depth data in order to obtain better alignment between colored point clouds. However, they do not deal with the color-depth sensor misalignment that is present in consumer cameras.

3.3 Conclusion of chapter

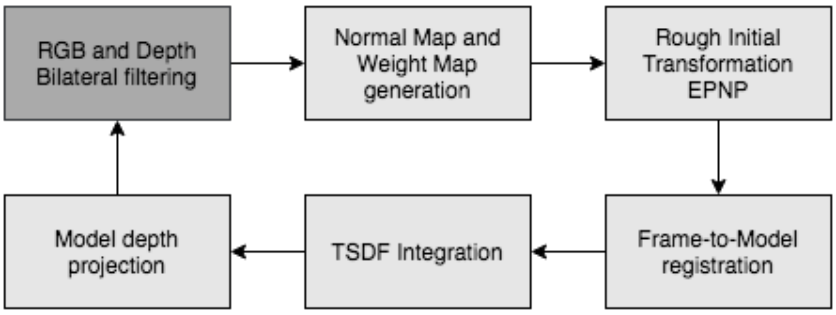
This chapter presented different possibilities and approaches used by researchers to tackle the problems of pose estimation and 3D reconstruction. In general, global registration methods need more time to compute global optimal solutions, but yields better results. Some of the global techniques use point matching in order to reduce the computation time, with the tradeoff of being less accurate. It also indicates that the use of joint color and depth is used by some authors, but existing approaches do not account for the potential mis-alignment between these two sensors.

4 THE PROPOSED APPROACH

In this thesis, we propose a pipeline that can build 3D models on-the-fly, fusing the acquired point cloud into a consolidated refined model, considering that at first the consolidated model is just the first acquired point cloud. We use the current pose to project the model, creating a depth map at time t that will be used to find the next camera pose at time $t + 1$ using a weighted combination of photometric and geometric features.

The first step of our method is to filter the input depth data, aiming to reduce noise. A normal map is obtained with the depth data and a weighting value is computed at each pixel. A rough initial camera pose is then obtained using a PnP algorithm, and a non-linear optimization method that jointly explores color and depth is used to refine the camera pose estimate. Depth data of the current frame is integrated to the model using Truncate Signed Distance Fields (TSDF), and each new depth image is aligned to the renderization of the model from the last pose. An overview of the proposed approach is illustrated in Fig. 4.1, and each step is detailed next.

Figure 4.1: Proposed iterative pipeline



4.1 Color and Depth Bilateral filtering

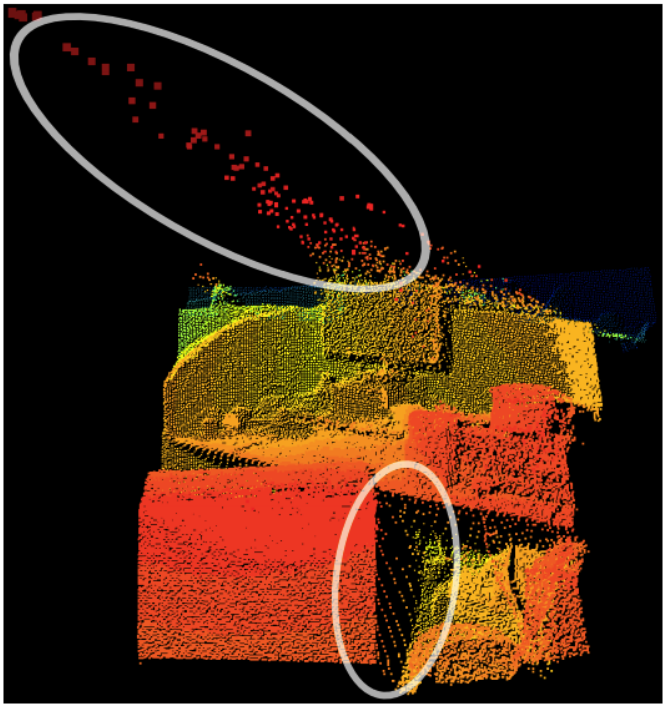
Cheap RGB-D Cameras have very limited image quality, particularly when it works under low illumination conditions. In the case of depth data, color and reflectance properties of the scene often cause pixels to be obtained with bad depth estimates, which are undesirable both to the registration process and model integration process. To improve the quality of the final model, normal map estimation and get better results at photometric/depth alignment, we perform a fast bilateral filtering (PARIS; DURAND, 2009) on

input RGB and depth images. This filter reduces noise of depth measurements and RGB pixels preserving the edges of the original image based on two parameters: σ_s to control spatial smoothing and σ_r controls range smoothing.

4.2 Normal Map and Weight Map Estimation

Although the pipeline presented in this work is generic for any point cloud capture device, we note that devices that explore Time of Flight (ToF) or structured light present limitations when the scanned surface is locally orthogonal (or close to orthogonal) to the emitting rays. In these cases, the amount of reflected light back to the sensor (in the case of ToF cameras) is small, so that the depth estimate (and consequently the corresponding 3D camera coordinates) might be inaccurate or generating "flying pixels" that are disconnected from the object. An example of such artifacts is shown in Figure 4.2, highlighted by gray ellipses.

Figure 4.2: Flying pixels generated by ToF Camera



Source: (Reynolds et al., 2011)

To avoid generating point cloud models with flying pixels or similar artifacts, we first prune the acquired point cloud using a pre-processing step based on surface normals. More precisely, we compute the angular distance $\theta(\boldsymbol{n}_c, \boldsymbol{n}_p)$ between the local normal vector $\boldsymbol{n}_p$ at each point of the cloud and an inverted virtual camera axis vector $\boldsymbol{n}_c =$

$(0, 0, -1)^\top$ (normals with negative z coordinate points towards the camera) at the same coordinate frame origin given by

$$\theta(\boldsymbol{n}_c, \boldsymbol{n}_p) = \arccos < \boldsymbol{n}_p, \boldsymbol{n}_c >, \tag{4.1}$$

where $< \cdot, \cdot >$ denotes the inner product. If the angle is sufficiently large (based on a threshold T_θ, in this case $70°$), the point is removed from the point cloud.

4.3 Rough Initial Registration based on ePnP

Although we assume a relatively temporally continuous capture of the object, rapid camera (or object) motion might generate adjacent point clouds that are significantly apart from each other. Since our fine alignment scheme (that will be described next) might not converge in such cases, we first perform a rough alignment of the point clouds.

A key aspect in PnP algorithms is obtaining point correspondences, which can be done based on geometric (point cloud) or textural (color) information. When analyzing objects at a near distance, which is the case of this work, our tests with geometric descriptors like PFH (RUSU et al., 2008), FPFH (RUSU; BLODOW; BEETZ, 2009), or SHOT (TOMBARI; SALTI; STEFANO, 2010) did not show very stable matching results. Hence, we opted to use image-based features instead, assuming that objects to be scanned provide textural information. More precisely, we selected ORB image features (RUBLEE et al., 2011), due to the good compromise between quality and execution time. To estimate the 3D position of extracted keypoints from the reference view, we unproject the keypoints to 3D space using the reference depth map frame. Assuming that the depth camera is calibrated (which is the case, since this information is used to fuse RGB and depth values), we end up with the classical Perspective-n-Point (PnP) problem. From the several existing solutions, we chose the EPnP algorithm (LEPETIT; MORENO-NOGUER; FUA, 2009) due to its speed. We also include a RANSAC-based outlier removal scheme over EPnP.

To prune bad point matchings, for each detected ORB feature in the actual frame we list only the top-2 best ORB feature matchings from the previous frame, where the matching score is done with Hamming distance between the binary descriptors. We define the point matching as valid when the Euclidian distance on the image domain between two

matched points is below 100 pixels (since we are using images with 640×480 resolution this distance allows wide motion between frames). Moreover, the Hamming distance of best matched descriptor must be smaller than 80% of the Hamming distance of the second-best match, to avoid selecting ambiguous matches.

In cases where RANSAC can not obtain 90% of inliers (as inlier we mean points with reprojection error below 1-pixel distance), we coniseder that PnP did not produce a valid pose. In that case, we choose the identity matrix as the initial transformation for the non-linear refinement.

The parameters/thresholds presented above were obtained during the empirical tests, assuming that camera motion is mostly smooth and the matching refinement criteria remove most of outliers from data.

4.4 Refined Registration

After the initial pose estimation provided by EPnP, we fine-tune the estimate by solving a convex optimization problem that aims to minimize intensity and depth differences between pixels of each pair of the RGB-D images in two consecutive video frames. The goal of the optimization problem is to obtain optimal parameters of a rigid transformation matrix $\boldsymbol{T}$ that describes the 6-DoF synthetic camera pose that renders the best alignment between two colored point clouds: the first one is the current model, rendered by a synthetic RGB-D camera at the location of the last camera pose, and the second one is the RGB-D frame currently being captured by the sensor. We find the optimal rigid transformation between these two frames using a fine-tuned Gauss-Newton algorithm over a joint photometric and geometric error function.

Following the approach proposed in (PARK; ZHOU; KOLTUN, 2017), we have the photometric and geometric errors combined in the same objective function. As in their work, our algorithm receives as input a pair of registered RGB-D images $(\boldsymbol{I}_i, \boldsymbol{D}_i)$, $(\boldsymbol{I}_j, \boldsymbol{D}_j)$ and a transformation $\boldsymbol{T}$ that roughly aligns the images, obtained from ePnP.

4.4.1 Photometric Error

The Photometric Error E_P is formulated as the sum of squared intensity differences, as introduced in works of Park, Zhou and Koltun (2017), Steinbrucker, Sturm

and Cremers (2012), Kerl, Sturm and Cremers (2013). As argued by these authors, the choice of combining RGB channels in one gray scale value reduces the computational cost without significant loss of accuracy. In this work, we consider findings of Nguyen, Izadi and Lovell (2012), Khoshelham and Elberink (2012), Mallick, Das and Majumdar (2014) and introduce a sensor independent weighting function $w_P(\boldsymbol{p})$ for each pixel at the location $\boldsymbol{p}$ that aims to reduce the importance of residuals which the depth estimate might be degraded. As this depth accuracy degradation might produce erroneous RGB to Depth matching, we prefer to give progressively less weight to the color pixels that are aligned with poor depth estimates.

The photometric error E_P is given by

$$E_P(\boldsymbol{T}) = \sum_{\boldsymbol{p},\boldsymbol{q}} w_P(\boldsymbol{p}) \left(\boldsymbol{I}_i(\boldsymbol{p}) - \boldsymbol{I}_j(\boldsymbol{q})\right)^2, \tag{4.2}$$

where $\boldsymbol{p} = (u, v)^T$ is a pixel in the registered $(\boldsymbol{I}_i, \boldsymbol{D}_i)$ image pair and $\boldsymbol{q} = (u', v')^T$ is its correspondent pixel in the $(\boldsymbol{I}_j, \boldsymbol{D}_j)$ image pair. This correspondence is found by unprojecting $\boldsymbol{q}$ back to the 3D space (using pixel location, depth estimate and camera intrinsic parameters), applying the transformation $\boldsymbol{T}$ and re-projecting it into pixel coordinates of the $(\boldsymbol{I}_j, \boldsymbol{D}_j)$ pair:

$$\boldsymbol{p} = \sigma_{uv}(s(\pi^{-1}(\boldsymbol{D}_j(\boldsymbol{q})), \boldsymbol{T})), \tag{4.3}$$

Where s is the function that applies rigid transformation to the homogeneous points according to:

$$s(\boldsymbol{x}, \boldsymbol{T}) = \boldsymbol{T}\boldsymbol{x}. \tag{4.4}$$

Also, the inverse point projection π^{-1} from depth map to homogeneous 3D point is defined as:

$$\pi^{-1}(u, v, d) = \left(\frac{d(u - c_x)}{f_x}, \frac{d(v - c_y)}{f_y}, d, 1 \right)^\top, \tag{4.5}$$

where f_x and f_y are focal lengths and (c_x, c_y) is the principal point coordinate of the camera. The function π maps a 3D point $\boldsymbol{w} = (w_x, w_y, y_z, 1)^\top$ into image coordinates u, v plus an additional coordinate d to store depth information based on how much a point is away from the sensor.

$$\pi(w_x, w_y, w_z) = \left(\frac{w_x f_x}{w_z} + c_x, \frac{w_y f_y}{w_z} + c_y, w_z \right)^\top , \tag{4.6}$$

To extract information from images we defined two convenient functions, namely σ_{uv} and σ_d. Function σ_{uv} returns the first two coordinates of π, which are the pixel coordinate on the image plane $(\boldsymbol{I_i}, \boldsymbol{D_i})$, while function σ_d returns only the last coordinate which stores the depth of the point.

4.4.2 Geometric Error

The Geometric Error E_G is obtained in a similar way to the ICP point-to-plane formulation of (LOW, 2004), except that we choose using only the third component from normals $\boldsymbol{N}_i(p)$ and 3D points of depth maps $\boldsymbol{D}_i$ and $\boldsymbol{D}_j$.

We are first intending to align confident points from planes that are parallel to the camera's image plane while avoiding alignment of noisy points that lies on planes that are almost orthogonal to the image plane. As $n_z = cos(\theta)$, we are giving greater weighting to the fronto-parallel tangent planes of the incoming point cloud.

Differently from the proposed photometric error modelling, here we can easily measure how much the values of the correspondent points change as the camera moves and the difference must be calculated using values obtained from σ_d composed with $\boldsymbol{D}_i(p)$ in order to get the modified depth values of $\boldsymbol{W}_j(q)$ at each pose step. As in the photometric error we also use a weighting scheme, and define

$$\boldsymbol{W}_j(q) = s(\pi^{-1}(\boldsymbol{D}_j(\boldsymbol{q})), \boldsymbol{T})) \tag{4.7}$$

$$E_G(\boldsymbol{T}) = \sum_{\boldsymbol{p},\boldsymbol{q}} w_G(\boldsymbol{p}) \boldsymbol{N}_i(p) \left(\boldsymbol{D}_i(\boldsymbol{p}) - \sigma_d(\boldsymbol{W}_j(q)) \right)^2 , \tag{4.8}$$

where $w_G(\boldsymbol{p})$ is a weighting function for the geometric error. In this work we chose set $w_G = w_P$, and this common weighting function will be detailed in Section 4.4.3.

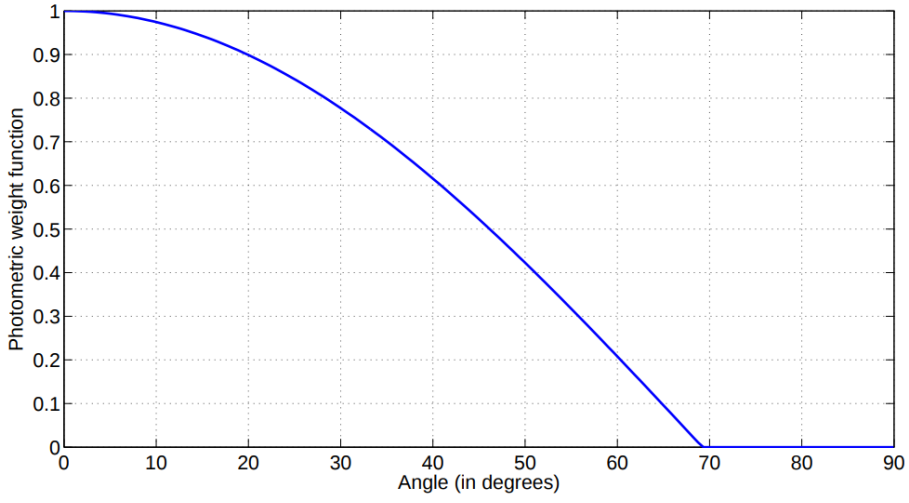
4.4.3 Point weighting

As discussed in Section 4.1, time of flight and structured light depth cameras are more accurate when scanning fronto-parallel regions, since the amount of reflected/captured light is larger. The previous statement is confirmed in the case of Kinect v1 and v2 sensors as presented in the works of Nguyen, Izadi and Lovell (2012) and Grossmann (2017). Besides using this information for removing potentially bad points, we also explore it for weighting the global photometric and geometric distances. More precisely, the largest weight should be assigned to points that present tangent planes fronto-parallel to the camera sensor, and progressively less weight as the tangent planes become closer to fronto-orthogonal. This behavior is captured by the angle $\theta(\boldsymbol{n}_c, \boldsymbol{n}_p)$ between the camera viewing direction $\boldsymbol{n}_p$ and the normal vector $\boldsymbol{n}_p$ to the tangent plane of the point $\boldsymbol{p}$ under consideration. Our weighting function w_P is given by

$$w_P(\boldsymbol{p}) = \max\{0, \cos\left(\tau\theta(\boldsymbol{n}_c, \boldsymbol{n}_p)\right)\}, \tag{4.9}$$

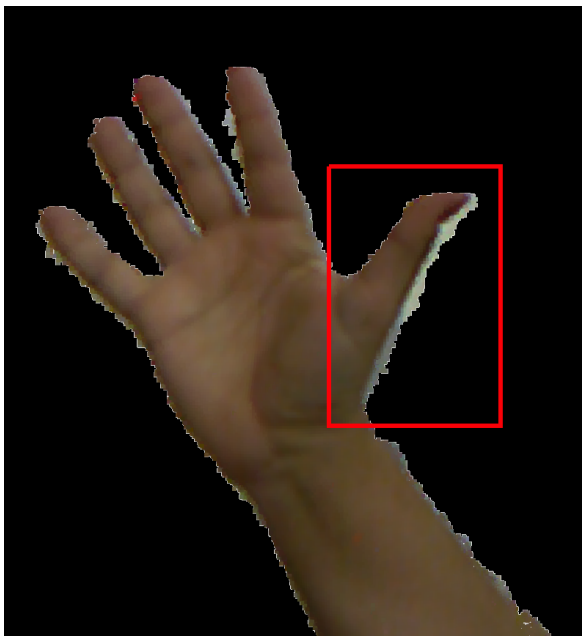
where τ is a parameter that controls how fast the weight decreases as the angle θ increases, so that θ values above $\pi/(2\tau)$ lead to null weights (i.e., the corresponding points are discarded from the analysis). We selected $\tau = 1.3$ based on experiments and by the strong indication in the works of Nguyen, Izadi and Lovell (2012), Grossmann (2017) that most of Z-axial error (depth measurement) occurs when tangent plane angle varies from $70°$ to $90°$ with respect to the camera pointing vector. The plot of w_P for $0° \leq \theta \leq 90°$ is provided in Fig. 4.3, and angular differences above $\approx 67°$ generate null weights.

Figure 4.3: Plot of the photometric weight function w_P.



It is important to note that using $w_P(\boldsymbol{p})$ weighting function also helps with another common issue when dealing with RGB-D cameras: "color bleeding". Note that depth and color information are captured by different calibrated sensors, and later fused together based on the (intrinsic and extrinsic) parameters of each sensor. Since these parameters typically present some error, the registration of the two sensors is not perfect. In particular, the mis-alignment is more noticeable along depth discontinuities (which typically arise on the boundaries between two objects). At these locations, the colors of one object might be wrongly projected to the other object, as illustrated in Fig. 4.4, which could lead to inconsistencies between E_P and E_G. Since depth discontinuities tend to generate a front-orthogonal local tangent plane, its effect is implicitly alleviated by our weighting function.

Figure 4.4: Color bleeding caused by wrong RGB and depth registration at depth discontinuities.



4.4.4 Joint Error Optimization

We previously presented individual photometric and geometric errors, but we seek for a solution where both objective functions are minimized as much as possible. For that purpose we build a joint objective function that combines these errors through a weighted sum:

$$E(\boldsymbol{T}) = E_P(\boldsymbol{T}) + \lambda E_G(\boldsymbol{T}), \tag{4.10}$$

where $\lambda \geq 0$ controls the relative weight of the geometric error. Note that if we set $\lambda > 1$, we are favoring geometric over photometric alignment.

To represent the 6-DoF with a minimal representaion we use the twist vector $\boldsymbol{\xi} = [\omega_x, \omega_y, \omega_z, t_x, t_y, t_z]^\top$ that combines rotational and translational motion parameters to describe a rigid motion. This twist vector has its Lie algebra correspondence in matrix form $\hat{\boldsymbol{\xi}} \in$ se(3) as follows:

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} 0 & -\omega_z & \omega_y & t_x \\ \omega_z & 0 & -\omega_x & t_y \\ -\omega_y & \omega_x & 0 & t_z \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.11}$$

From this matrix we can obtain the 6-DoF transformation $\boldsymbol{T} \in$ SE(3) matrix using the matrix exponential function $\boldsymbol{T} = \exp(\hat{\boldsymbol{\xi}})$. To find the best transformation to this non-linear objective function, we use the Gauss-Newton method since it provides fast convergence. To calculate the Jacobian for residuals of the projected RGB and Depth pixels $\boldsymbol{r}_P$, $\boldsymbol{r}_G$ we need to compute $\boldsymbol{J}r_P$ and $\boldsymbol{J}r_G$ as shown below:

$$\boldsymbol{J}r_P = \frac{\partial \boldsymbol{r}_P}{\partial \boldsymbol{\xi}} = \frac{\partial \boldsymbol{r}_P}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}, \tag{4.12}$$

$$\boldsymbol{J}r_G = \frac{\partial \boldsymbol{r}_G}{\partial \boldsymbol{\xi}} = \lambda \begin{bmatrix} \boldsymbol{N}_r \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}} \end{bmatrix}, \tag{4.13}$$

where $\frac{\partial \boldsymbol{r}_P}{\partial \boldsymbol{x}}$ are the image gradients from the gray intensity image obtained from the original RGB data processed with the Scharr operator, $\boldsymbol{N}_r$ are the pixel-associated normal vectors obtained from the depth map using the technique described on paper of Badino et al. (2011), $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}$ is the 6-DoF motion derivative of points in 3D with respect to motion parameters in Eq. (2.13), and $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}$ are the perspective projection derivatives as follows:

$$\frac{\partial \mathbf{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\alpha_x}{Z} & 0 & \frac{-\alpha_x X}{Z^2} \\ 0 & \frac{\alpha_y}{Z} & \frac{-\alpha_y Y}{Z^2} \end{bmatrix}. \tag{4.14}$$

Based on the computed jacobians, we define:

$$\boldsymbol{J}_{\boldsymbol{\xi}} = \left(\boldsymbol{J}r_P, \boldsymbol{J}r_G\right)^\top, \tag{4.15}$$

$$\boldsymbol{r} = \left(\boldsymbol{r}_P, \boldsymbol{r}_G\right)^\top, \tag{4.16}$$

and we use these variables to solve the following linear system in order to obtain motion increments $\Delta\boldsymbol{\xi}$:

$$\boldsymbol{J}_{\boldsymbol{\xi}}^\top \boldsymbol{J}_{\boldsymbol{\xi}} \Delta\boldsymbol{\xi} = -\boldsymbol{J}_{\boldsymbol{\xi}}\boldsymbol{r}. \tag{4.17}$$

To check if we have found a new motion increment that minimizes the objective function, we update the matrix $\hat{\boldsymbol{\xi}}$ to calculate the rigid motion matrix $\boldsymbol{T}_t = \exp(\hat{\boldsymbol{\xi}})$ that aligns the model with the current frame. This process is repeated until the objective function no longer decreases its value or has reached to the maximum of iterations allowed. The pipeline keeps an updated matrix $\tilde{\boldsymbol{T}}_{t-1}$ containing the last pose of camera until this last estimate. By concatenating the last pose matrix and the inverse of the last found transformation $\boldsymbol{T}_t^{-1}$ we get the new actual camera pose matrix:

$$\tilde{\boldsymbol{T}}_t = \tilde{\boldsymbol{T}}_{t-1}\boldsymbol{T}_t^{-1} \tag{4.18}$$

4.5 TSDF Integration and Model generation

To incrementally build the mesh and use it as a smooth reference model, we use a type of Signed Distance Function (CURLESS; LEVOY, 1996) and integrate point cloud data into a Truncated Signed Distance Function (TSDF) as in KinectFusion (NEWCOMBE et al., 2011). The core idea of this step is to accumulate the measured distances between points that fall into the same voxel space with a weighted running average given by:

$$D_{i+1}(\boldsymbol{p}) = \frac{W_i(\boldsymbol{p})D_i(\boldsymbol{p}) + w_{i+1}(\boldsymbol{p})d_{i+1}(\boldsymbol{p})}{W_i(\boldsymbol{p}) + w_{i+1}(\boldsymbol{p})}, \tag{4.19}$$

$$W_{i+1}(\boldsymbol{p}) = W_i(\boldsymbol{p}) + w_{i+1}(\boldsymbol{p}), \tag{4.20}$$

where $D_i(\boldsymbol{p})$ is the accumulated average of signed distance of the point $\boldsymbol{p}$ to the centroid $\boldsymbol{c}_i$ of the voxel that contains the point, $W_i(\boldsymbol{p})$ is a function that returns the previous voxel accumulated weight, $w_{i+1}(\boldsymbol{p})$ is defines the incremental weight (set to 1). The term $d_{i+1}(\boldsymbol{p})$,is the new point distance to surface scaled by the truncation value μ that will be accumulated into the same voxel, given by:

$$d_{i+1}(\mathbf{p}) = \begin{cases} \text{sgn}(\phi_i(\mathbf{p})) & \text{if } |\phi_i(\mathbf{p})| > \mu \\ \phi_i(\mathbf{p})/\mu & \text{otherwise} \end{cases} \tag{4.21}$$

where $\phi_i(\mathbf{p})$ denotes signed distance between point $\mathbf{p}$ and the voxel centroid, given by

$$\phi_i(\boldsymbol{p}) = \pi(\boldsymbol{c}_i) - \pi(\boldsymbol{p}) \tag{4.22}$$

At the very first integration of a point cloud to the TSDF structure we initialize all the with voxels with $W_i(\boldsymbol{p}) = 0$ and $W_{i+1}(\boldsymbol{p}) = 1$ and the update voxels where the incoming points falls with $d_{i+1}(\boldsymbol{p})$ function. Finally to extract the surface from the TSDF voxels at every iteration, the classical Marching Cubes algorithm (LORENSEN; CLINE, 1987) is used.

4.6 Model projection

The core of the proposed approach described in Section 4.4 requires a pair of color and a pair of depth images (at adjacent frames). Although all these four images are provided by the RGB-D camera, we adopted a frame-to-model alignment. More precisely, we synthetically generate the depth image $\boldsymbol{D}_i$ by projecting the model from the last tracked pose of the camera, rendering only the closest visible points of the model concerning the current camera position. This is easily obtained from the depth buffer obtained from Open3D's visualization module.

The synthetic generated depth image is smoother than the depth image produced by the sensor, as it is rendered from a mesh obtained from the accumulated depth values of TSDF. This improves the alignment process, reducing measurement errors. On the other hand, a RGB image obtained by synthesizing a view from the current 3D model is

not suitable because it tends to accumulate errors due to color bleeding when RGB and depth present any misalignment. Hence, we use a hybrid approach: the color image pair is obtained directly from the sensors; for the depth image pair, one image is obtained from the sensor (current frame), and the other (previous frame) obtained by rendering the consolidated model from the last camera pose.

5 EXPERIMENTAL RESULTS

We implemented our algorithm in C++11, with the help of the libraries `OpenCV` for image manipulation and `Open3D` for 3D visualization and TSDF data structures. All tests were run on a PC running Linux Mint 18.3 in an Intel i5-7400 processor with 8 GB of RAM. All the pipeline runs only on CPU, except for visualization tasks that explore GPU processing.

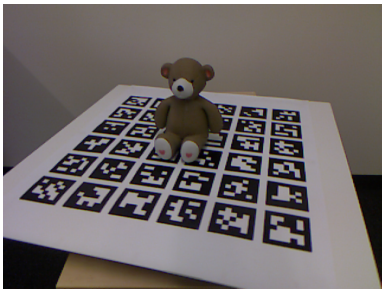
To evaluate the accuracy of our reconstruction approach, we performed tests using the 3D-Printed RGB-D Object Dataset introduced in (SLAVCHEVA et al., 2016). This dataset consists of a selection of five different 3D-printed objects that were scanned using RGB-D sensors of different quality and with two types of scanning camera motions: stopped camera pointing to the object on a turntable and a handheld camera motion (more erratic) around the object. Here we use only the Kinect recorded scenes of the dataset because only these can emulate an application using off-the-shelf cameras, noting that the customized phase shift camera and synthetic data available in this dataset are not suitable to the scope of this work. In this dataset, the objects lies over a textured table with fiducial markers in order to obtain the ground truth poses, as we can see in Figure 5.1. Since the ORB features used in our method might artificially benefit from these fiducial markers, we have also captured other datasets with a Kinect sensor to evaluate our approach.

Ground truth camera motion data is provided with the dataset, which enables us to quantitatively evaluate the performance of the camera pose estimation of the algorithm using the Relative Pose Error (RPE) and Absolute Trajectory Error (ATE), which were presented in (Sturm et al., 2012).

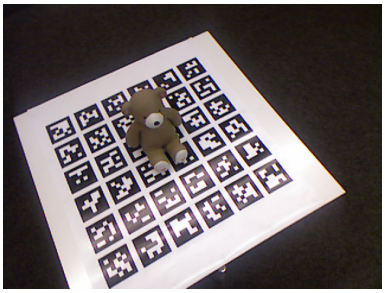
To evaluate the quality of the final reconstruction, which is the main goal of this work, we compare each model created using the ground truth data with the models created by Open3D (PARK; ZHOU; KOLTUN, 2017), our proposed approach, and our implementation without any weighting scheme. To this end, we use CloudCompare (GIRARDEAU-MONTAUT, 2003) software using the C2M (Cloud to Mesh) Distance, which provides a list of distances between each point of a point cloud to a (ground-truth) 3D mesh.

Figure 5.1: Frames of 3D Printed Dataset

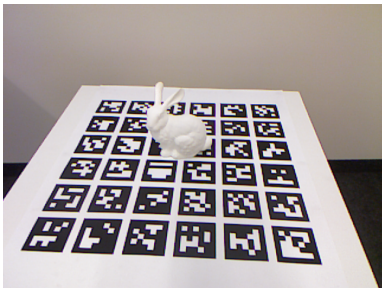
(a) Teddy Turntable



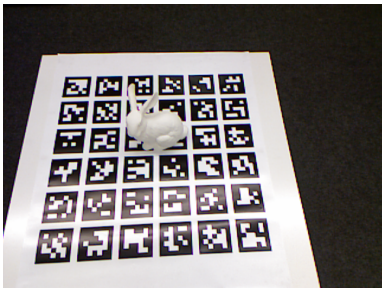
(b) Teddy Handheld



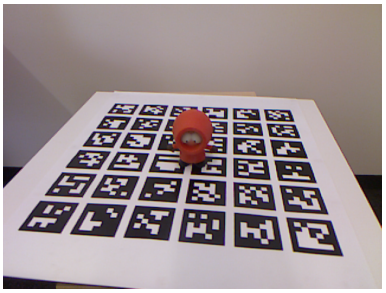
(c) Bunny Turntable



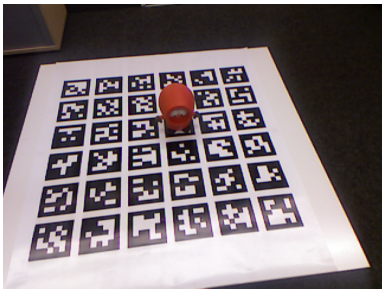
(d) Bunny Handheld



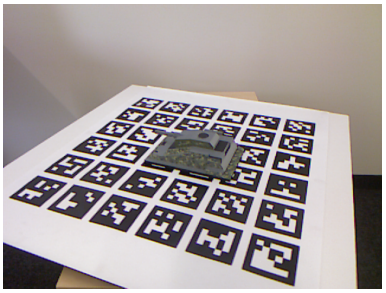
(e) Kenny Turntable



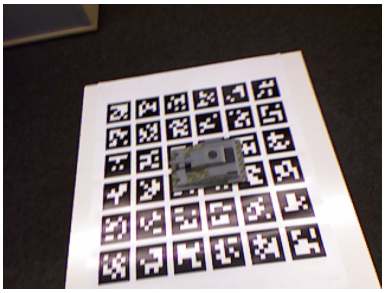
(f) Kenny Handheld



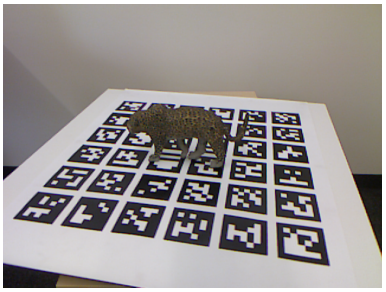
(g) Tank Turntable



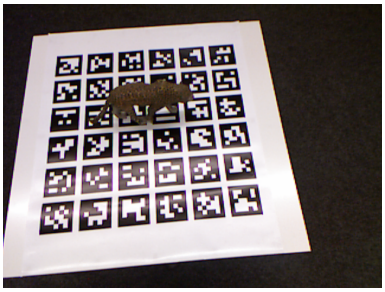
(h) Tank Handheld



(i) Leopard Turntable



(j) Leopard Handheld



5.1 Open3D Library and Colored Point Cloud Registration

`Open3D` is a recent open source library that support development of software that works with 3D data. This library includes modules to provide visualization, 3D point cloud registration, visual odometry, cloud filtering and many other tasks (ZHOU; PARK; KOLTUN, 2018).

In this work we use its 3D visualization module to visualize our results and generate depth model frames. To integrate depth data we use its TSDF data structure implementation and to compare performance we use its colored point cloud alignment algorithm (PARK; ZHOU; KOLTUN, 2017) as baseline.

5.2 Parameter setting

We set $\lambda = 5$ in Equation (4.10) favoring a tight depth alignment while using color information to disambiguate point matchings on planar regions of depth data. Larger values of λ tend to make the error function behave like the common ICP distance error. Conversely, lower λ values tend to preserve photometric errors at lower levels but not necessarily with a proper depth alignment. This value for λ was found based on empirical tests covering all the scenes of the used dataset.

For the edge-aware smoothing using the bilateral filter, we selected $\sigma_s = 3$ and $\sigma_r = 5$, also based on visual inspection of the images. We look for parameters that lead homogeneous areas to be almost equal in local ranging of values while areas with edges being preserved.

To remove possible flying points we use a conservative pruning angle value $T_\theta = 70^\circ$ was chosen for Equation (4.1), which might also remove some good points. However, we noted that removing a few good points at some frames is better than keeping bad estimates, since missed good points tend to appear again (at a better angle) during other frames of the capture.

5.3 Quantitative evaluation of camera trajectory (pose)

We used the translational RPE and ATE metrics to compare our results with state-of-art work by Park, Zhou and Koltun (2017), which is implemented in the Open3D li-

brary (from this point we will refer to Open3D as the Colored Point Cloud registration module of the library). For a fair comparison, Open3D is used as a substitute for the "Frame-to-Model Registration" step of our algorithm presented in Figure 4.1.

5.3.1 Relative Pose Error (RPE)

This evaluation metric error measures local accuracy of the camera trajectory estimate, being used as a methodology to measure the drift that occurs at each pair of consecutive frames. In this evaluation, we use the well-known root-mean-square error (RMSE) measure, and compare results using the percentage difference between results putting in evidence relative gain of our technique over Open3D.

Table 5.1: RPE RMSE results, scaled by a factor of 10^3

Dataset	**Open3D**	**Proposed**	**W/o weight**
Teddy Turntable	1.66	**0.60**	0.61
Bunny Turntable	9.58	**0.57**	0.58
Kenny Turntable	2.88	**0.60**	0.62
Tank Turntable	3.61	**0.56**	0.57
Leopard Turntable	2.98	**0.62**	0.65
Teddy Handheld	12.05	**7.34**	7.44
Bunny Handheld	10.34	**4.79**	7.50
Kenny Handheld	6.75	**1.60**	2.20
Tank Handheld	10.82	6.83	**5.38**
Leopard Handheld	10.57	**5.76**	7.40
Average	7.12	**2.93**	3.29

As shown in Table 5.1, our algorithm consistently yields better results than Open3D in all scenes. When comparing the results with and without the weighting scheme, it is interesting to note that both approaches present similar errors for the datasets acquired with a turntable. However, the difference is noticeable when using the less constrained handheld capture mode, for which the weighing scheme yields improvements up to 63%.

5.3.2 Absolute Trajectory Error (ATE)

This evaluation metric measures the global accuracy of the camera pose estimates comparing the ground truth camera trajectory and the estimated camera trajectory. For a 3D reconstruction system, it can be an important quantity as the error in global trajectory

can lead to gross errors in the final generated model.

Table 5.2: ATE RMSE results scaled by a factor of 10^2

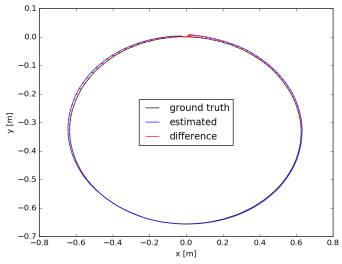
Dataset	**Open3D**	**Proposed**	**W/o weight**
Teddy Turntable	1.39	1.29	**1.13**
Bunny Turntable	2.80	0.81	**0.79**
Kenny Turntable	1.39	**1.15**	1.16
Tank Turntable	1.22	**1.12**	1.13
Leopard Turntable	1.24	1.11	**1.10**
Teddy Handheld	1.83	**1.16**	1.30
Bunny Handheld	1.52	**0.99**	1.25
Kenny Handheld	0.90	**0.75**	1.02
Tank Handheld	2.21	1.49	**1.41**
Leopard Handheld	1.87	**0.92**	1.21
Average	1.64	**1.08**	1.15

Observing the results in Table 5.2, again our technique presents better results than Open3D. The differences are smaller for some datasets, such as for “Teddy Turntable”, but in others, the results produced by our approach presents a considerably smaller error. For instance, our error for the “Bunny Turntable” dataset was one-third of the error obtained by Open3D. We can also observe that using the weighting scheme increases the ATE RMSE error for some datasets, but on average it is better than the version without weighting.

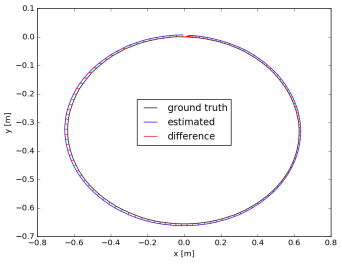
A full visual comparison of all the obtained camera trajectories is shown in Figures 5.2 and 5.3. The black line represent the ground truth motion, the blue is the estimated motion and the red lines connects ground truth and estimated poses to improve visualization of the difference between them. As one can see our approach was better in all scenarios in both motion types (Handheld and Turntable).

Figure 5.2: Trajectory Turntable: Visual comparison

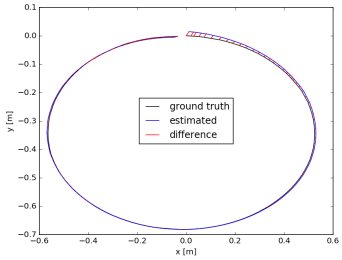
(a) Teddy Turntable Proposed



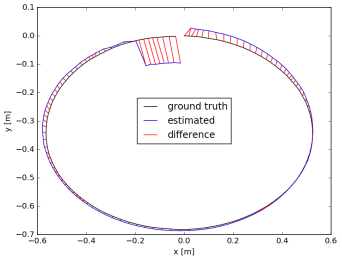
(b) Teddy Handheld Open3D



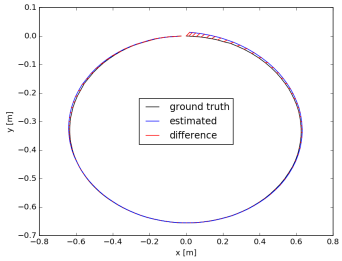
(c) Bunny Turntable Proposed



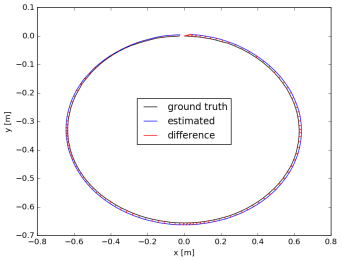
(d) Bunny Turntable Open3D



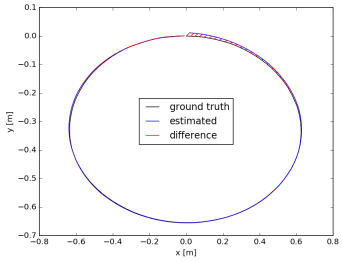
(e) Kenny Turntable Proposed



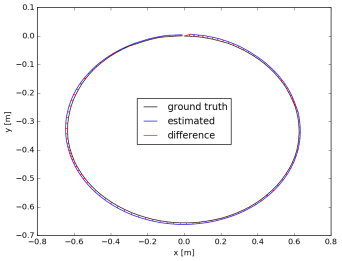
(f) Kenny Turntable Open3D



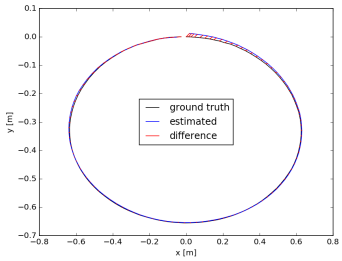
(g) Tank Turntable Proposed



(h) Tank Turntable Open3D



(i) Leopard Turntable Proposed



(j) Leopard Turntable Open3D

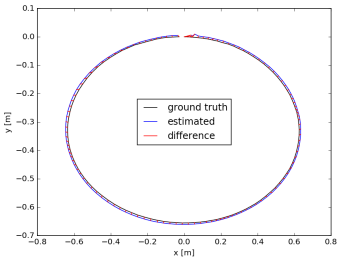
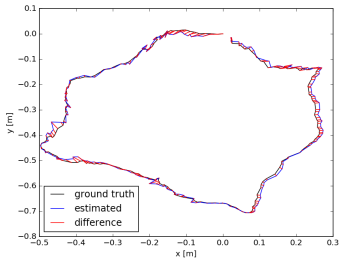
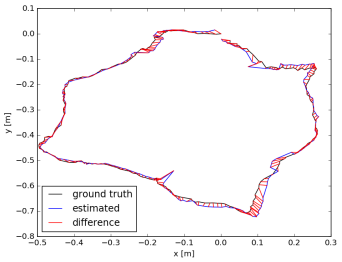


Figure 5.3: Trajectory Handheld: Visual comparison

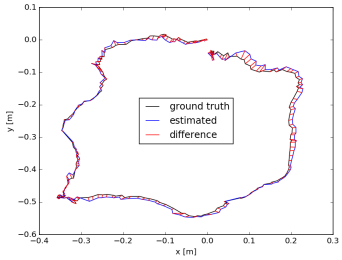
(a) Teddy Handheld Proposed



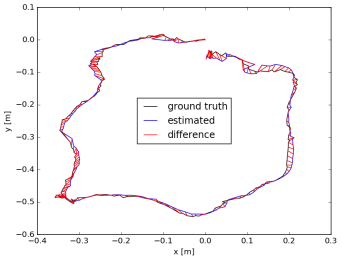
(b) Teddy Handheld Open3D



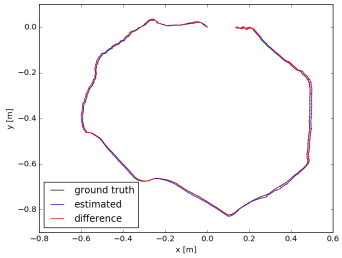
(c) Bunny Handheld Proposed



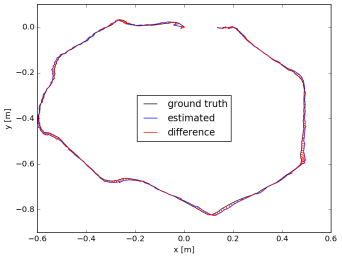
(d) Bunny Handheld Open3D



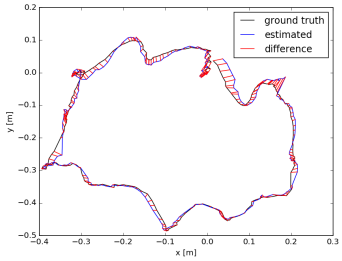
(e) Kenny Handheld Proposed



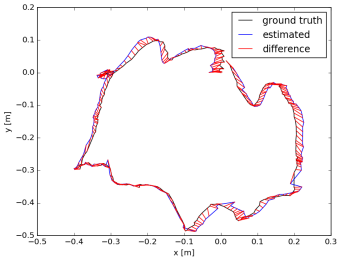
(f) Kenny Handheld Open3D



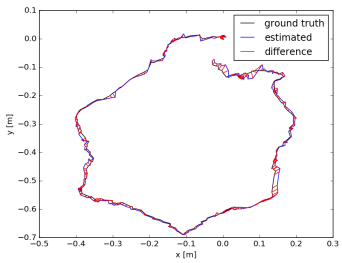
(g) Tank Handheld Proposed



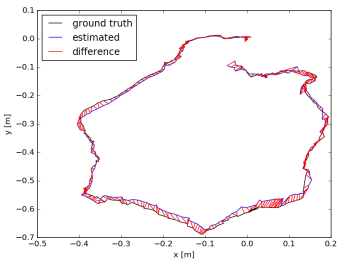
(h) Tank Handheld Open3D



(i) Leopard Handheld Proposed



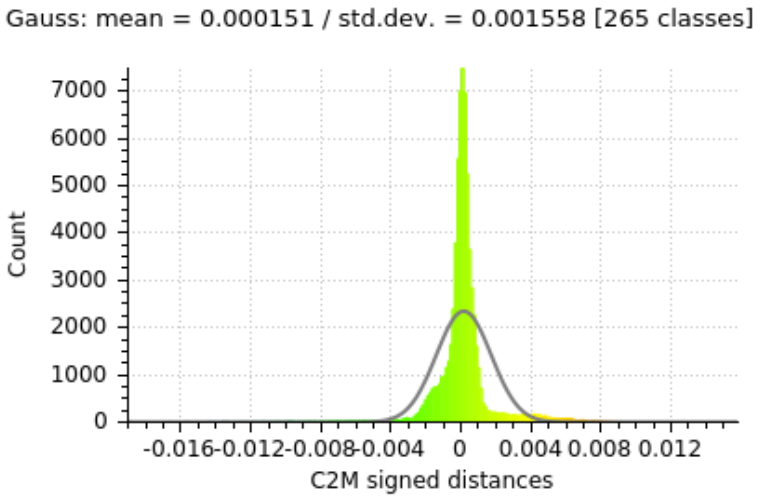
(j) Leopard Handheld Open3D



5.4 Quantitative comparison of the 3D models

Although smaller pose estimation errors are expected to generate better 3D models, that might not be always the case. To quantitatively evaluate the quality of the produced 3D models, we compare the 3D point distances of the obtained models with the corresponding "ground-truth" model generated using ground truth pose data. By using the Cloud-to-Mesh (C2M) tool provided by CloudCompare (GIRARDEAU-MONTAUT, 2003), we can obtain a histogram of distance differences, which is calculated with a point cloud and a mesh: one is the ground truth mesh, and the other is a point cloud sampled from the mesh created with our pipeline. Each of the points of the sampled cloud has its distance calculated with respect to the closest triangle from the ground truth mesh, and these distances are accumulated in a histogram as in Figure 5.4. We can obtain a measure of the mesh fidelity by computing the mean value and standard deviation of the error histograms. Although both the mean and variance can be strongly affected by outlier data, our visual inspection of the error histogram showed no signs of bad outliers.

Figure 5.4: Example of histogram of C2M error



Source: The authors

We summarize the results in Table 5.3, using the average of absolute distance error (note that distances might be positive or negative, depending if a given point is inside or outside the mesh) and standard deviation of the signed distance error. The first one is used to verify if the generated mesh presents its points close to the ground truth mesh, which might indicate if the mesh has as global shift with respect to the geometry of the ground truth mesh. The standard deviation with the signed distance error measures the deviation from the error average: if it is large, we can expect more deformations in the

final generated mesh. All of these measurements assume that two meshes are sufficiently well-registered by the native ICP of CloudCompare software. A posterior inspection is applied with manual adjustment if necessary to initialize a new ICP process until we get the minimal distance between the correspondence points.

Table 5.3: C2M Mean $\pm$ Std. Dev. scaled by a factor of 10^4

Dataset	Open3D	Proposed	W/o weight
Teddy T.	13.30 ± 21.53	$\mathbf{11.54 \pm 18.71}$	12.50 ± 19.85
Bunny T.	90.39 ± 119.07	$\mathbf{11.66 \pm 17.56}$	11.97 ± 18.40
Kenny T.	$\mathbf{2.64} \pm 14.47$	$8.64 \pm \mathbf{14.15}$	10.14 ± 17.66
Tank T.	$\mathbf{5.65 \pm 11.01}$	10.20 ± 17.87	10.58 ± 19.37
Leopard T.	$\mathbf{7.85 \pm 14.27}$	11.49 ± 18.70	12.07 ± 20.04
Teddy H.	33.68 ± 65.37	$\mathbf{9.48 \pm 18.05}$	9.75 ± 19.11
Bunny H.	12.58 ± 30.53	$\mathbf{6.20 \pm 9.71}$	7.22 ± 11.55
Kenny H.	$\mathbf{7.17} \pm 17.98$	12.30 ± 20.61	$7.72 \pm \mathbf{14.21}$
Tank H.	16.03 ± 30.90	$\mathbf{4.93 \pm 8.60}$	7.90 ± 12.90
Leopard H.	16.94 ± 30.69	$\mathbf{8.35 \pm 16.88}$	18.78 ± 44.66
Average	20.62 ± 35.58	$\mathbf{9.48 \pm 16.08}$	10.86 ± 19.77

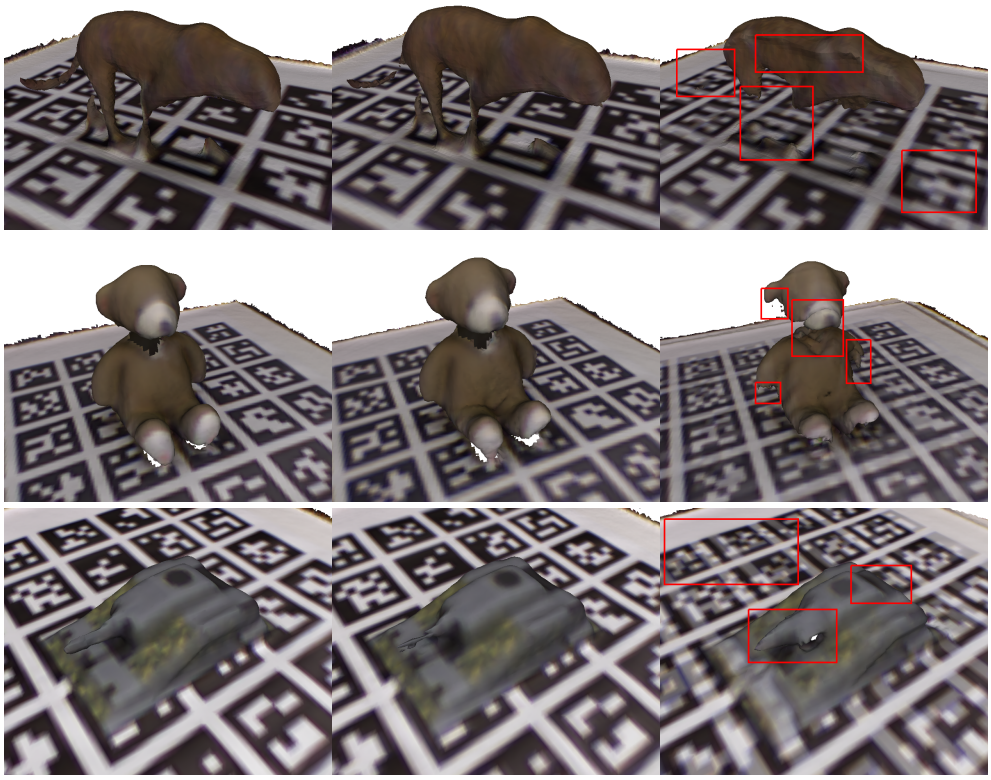
Table 5.3 indicates that the use of the weighting scheme provides consistently better results than not using it (both in terms of the mean and standard deviation). Also, our full approach (with weighting) presented smaller both average error and standard deviation than the baseline considering all datasets (see last row of the table). In particular, the improvement is more noticeable in the handheld sequences, which indicates that our methods can better handle erratic camera motion.

5.5 Qualitative evaluation

Although C2M presents a quantitative way of comparing 3D models, visual inspection is paramount for identifying the introduction of possible artifacts. In particular, the preservation of geometric details and thin structures of the generated models is essential for evaluating if the reconstruction succeeded. Figure 5.5 shows the GT model and the results produced by our approach and Open3D for three datasets, where the red rectangles highlight regions for which Open3D generated artifacts. Some artifacts are errors in geometry reconstruction, while others are observed as wrong color blending. It is also interesting to note that our reconstruction results were very similar to the GT model.

To get a better feel of the results, we also show a visualization of the scalar field

Figure 5.5: Top to Bottom: Leopard Handheld, Teddy Handheld and Tank Handheld datasets. From left to right: GT model, Proposed and Open3D.



generated by CloudCompare in Figures 5.6 and 5.7. These scalar fields represent the C2M distance in logarithm scale to better visualize very small differences. In the figures, red/orange areas represent larger errors, while green/blue areas represent smaller differences to the ground truth values. Looking at these results, one can observe that there are more red/orange areas (larger error) when Open3D alignment is applied than when our proposed approach is used.

As additional visual results, we have also created datasets related to 3D body scans and compared our results with Open3D. Figure 5.8 shows two views of a partial body scan produced by the two methods using 220 RGB-D frames to reconstruct these meshes. These frames were captured within a distance of 1 meter from the subject and the capture process took about 12 seconds. Although the results of both methods look coherent, a closer inspection indicates that the shirt texture on the shoulder of the person in Figure 5.8 (top) was somewhat deformed by Open3D, and the nose of the person in Figure 5.8 (bottom) was better preserved by our approach than Open3D. We have also performed a full 360 degree scanning of the same subject shown in Figure 5.9 using both methods. In this process, Open3D was not able to provide good alignments in the back of the person, and we aborted the execution. Our approach, on the other hand, did not show such drifts

Figure 5.6: Teddy T. Log C2M Difference. Top: Proposed. Bottom: Open3D

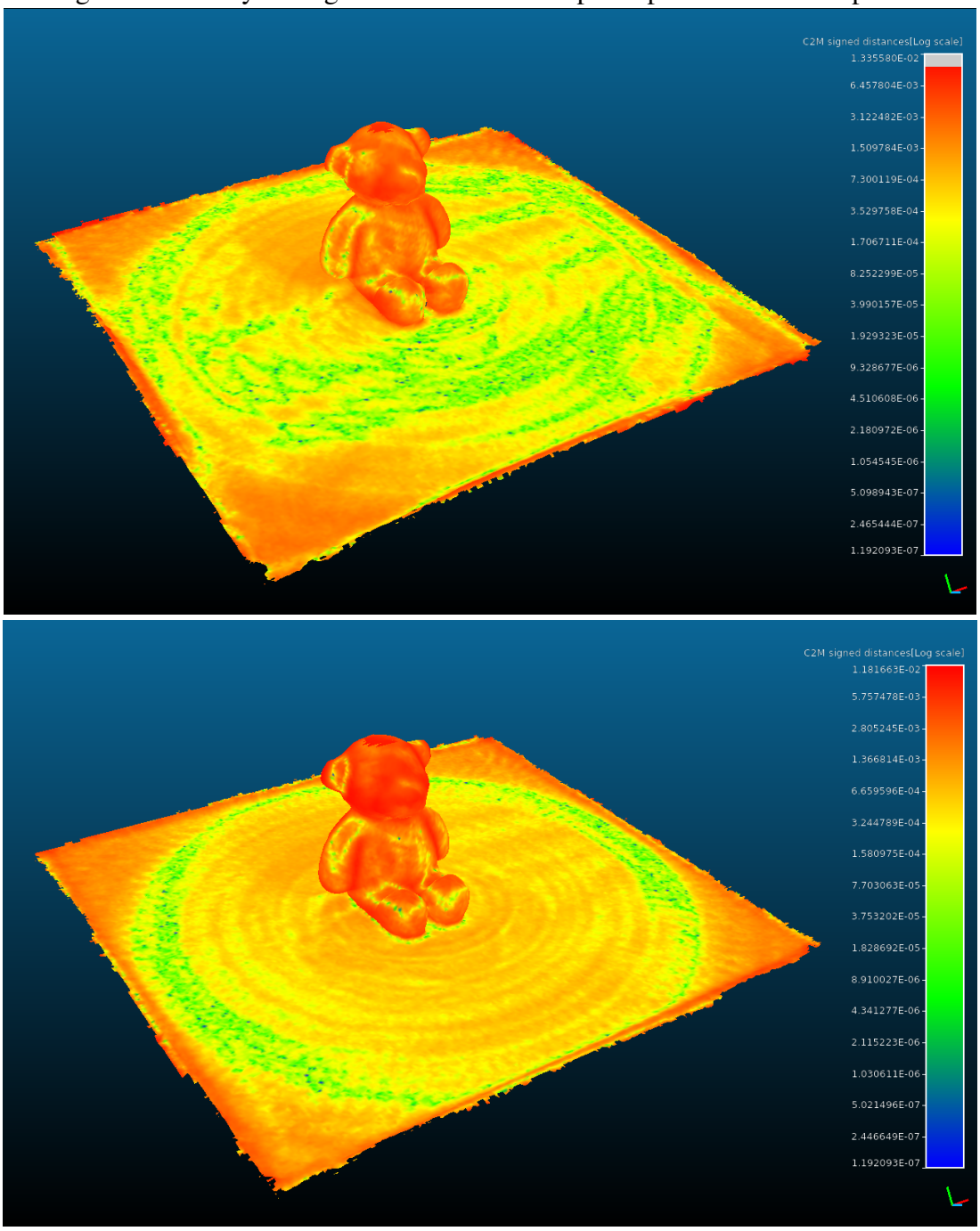
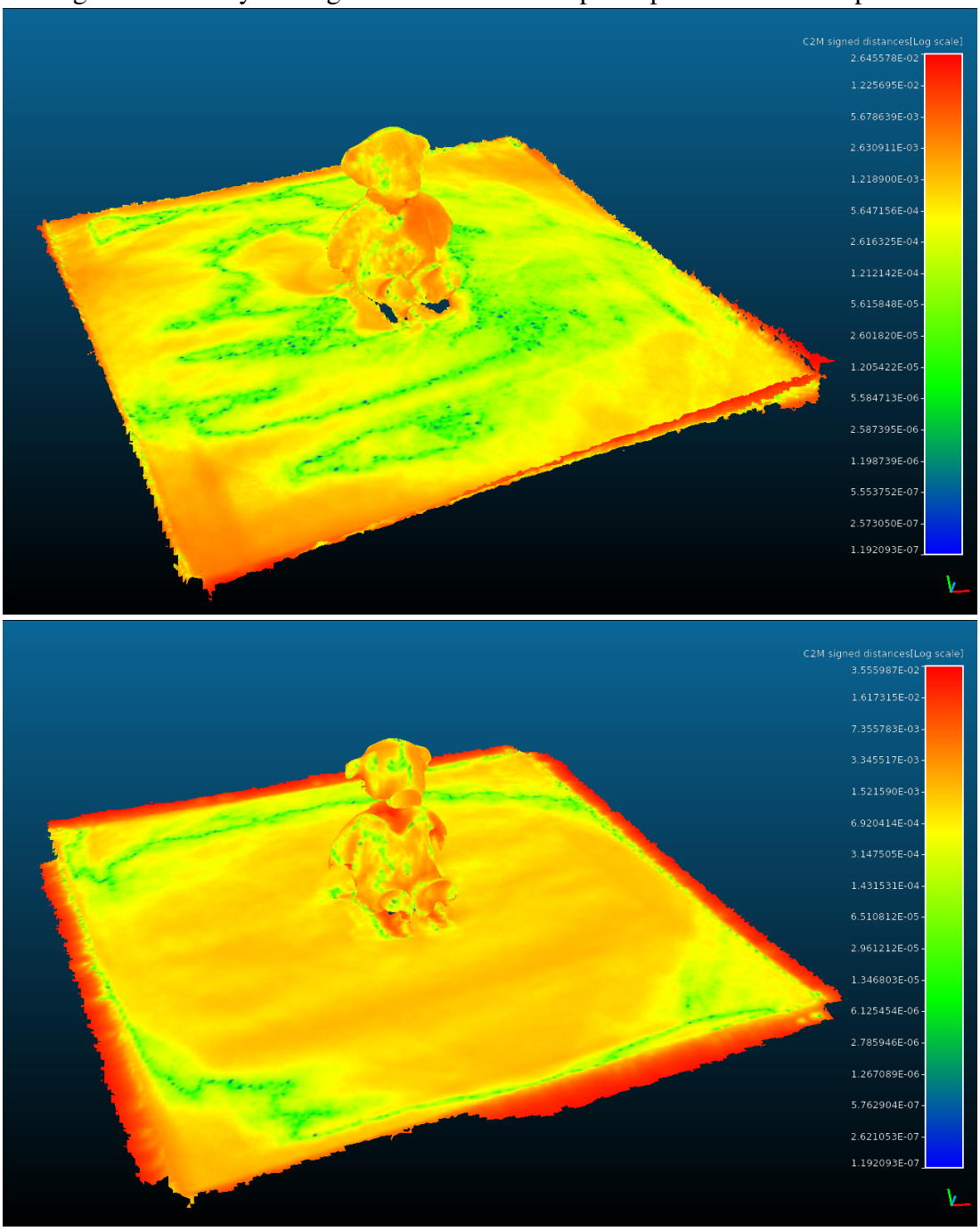


Figure 5.7: Teddy H. Log C2M Difference. Top: Proposed. Bottom: Open3D



(results shown in Figure 5.9). For this full scan, we captured 438 RGB-D images with distance to subject about 1 meter and total time of scene was captured in 23 seconds. The Figure 5.9 (bottom) shows a specific view when the first and last views meet. As it can be observed, the visual error in the loop closure was small.

Figure 5.8: Two views (top and bottom) of a partial body Scanning. Left: Proposed, Right: Open3D

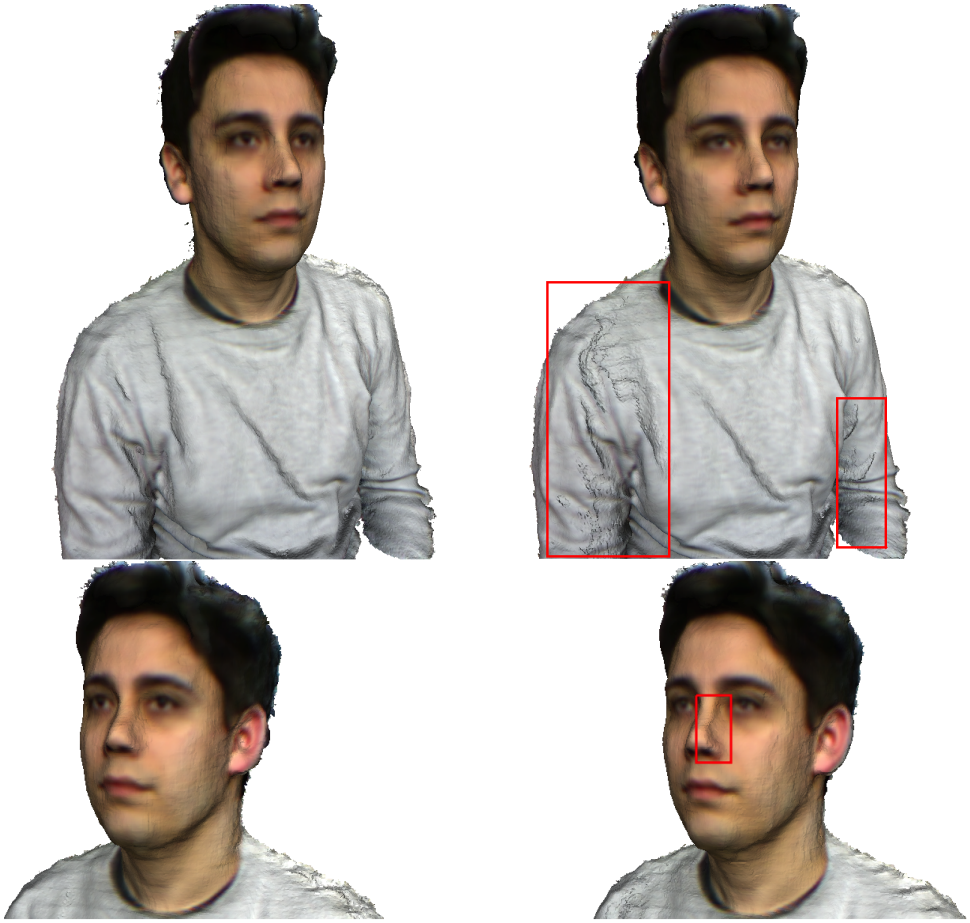


Figure 5.9: Full 360 body scanning. Top Proposed and Open3D. Bottom: view showing loop closure error using our method.

Smaller objects with low geometric details like boxes may be scanned as well by rotating the object over a texturized planar surface to take advantage of color alignment, as shown in Figure 5.10. This helps EPnP and refinement steps to give better appearance to the objects. In this case, one can observe a trade-off of geometry accuracy versus appearance accuracy of each approach. Another handheld scanned scene shown in Figure 5.11. Although both results are visually good, Open3d generates small artifacts in some geometrical structures, as highlighted by the red rectangles.

Figure 5.10: Full 360 small object scanning: Proposed vs Open3d

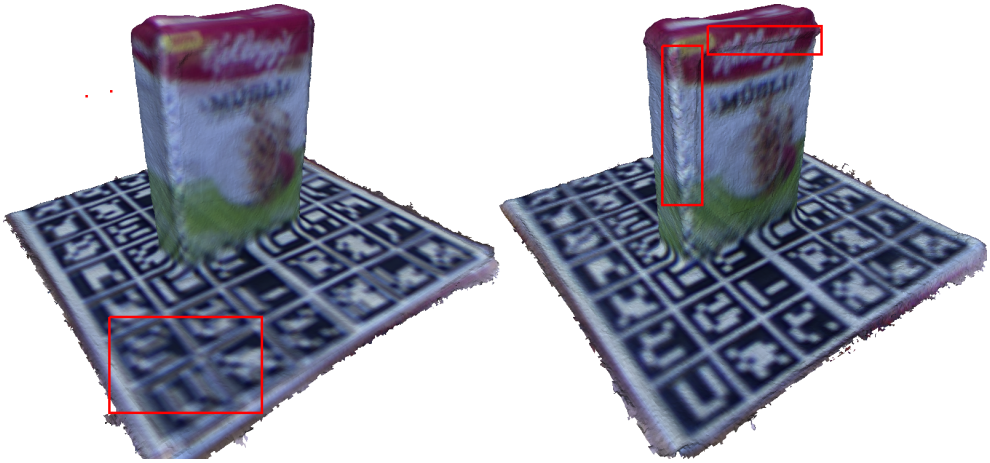
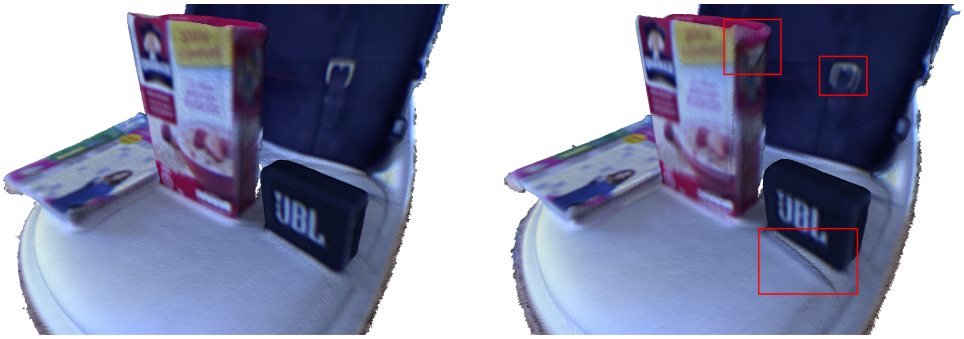


Figure 5.11: Handheld motion scanning: Proposed vs Open3d



5.6 Execution speed

Although the tested implementation is not optimized for parallel processing and did not explore GPU acceleration, we check the execution times for each pipeline step in order to check for possible processing bottlenecks. Running thought several frames we compute the average time per frame (in of the milliseconds) for each of steps of the proposed pipeline.

We confirm by measurements that time performance was affected mainly by the two components that had major complexity, Bilateral Filtering, and RGB-D Alignment as shown in Table 5.4. These two components are fully parallelizable, but for now, our implementation did not have this concern in mind. Open3D alignment step have the average of 856 milliseconds per iteration working with 640×480 pixels on depth and RGB images.

Table 5.4: Average in milliseconds per frame

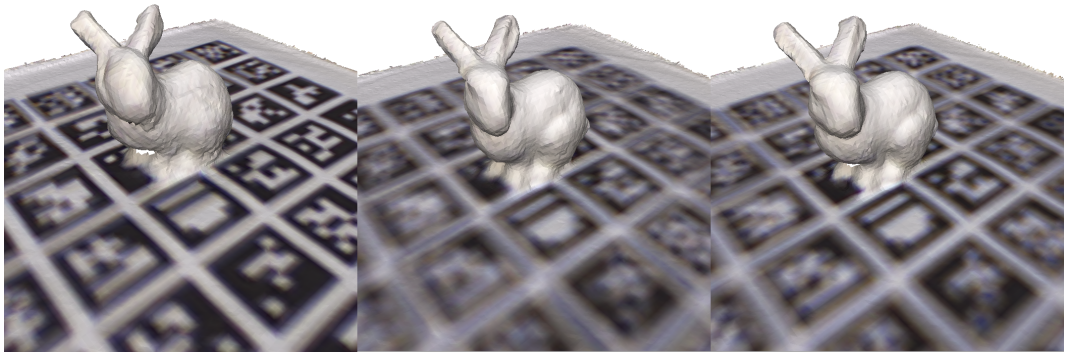
Bilateral Filter	**Normal Map**	**Rough Initialization**	**RGB-D Alignment**	**TSDF Integration**
1004	168	68	1045	251

5.7 Pose Graph Optimization Results

As a final experiment, we evaluated the effect of adding a post-processing scheme based on loop closure, which is adequate when performing full 360° scans which the initial frame and the final frame are obtained from the same camera pose. More precisely, we coupled the pose graph optimization algorithm (Sungjoon Choi; Zhou; Koltun, 2015) implemented in the `Open3D` library (ZHOU; PARK; KOLTUN, 2018) to the obtained camera trajectory with the constraint that initial and final camera poses are the same. Figures 5.12 and 5.13 illustrate some visual results, and they indicate that the overall geometry of the meshes is improved with the post-processing step. Interestingly, the texture on the planar surface was blurred after the optimization step, which corroborates the hypothesis of mis-alignment of color and depth sensors. This final step takes a few seconds to optimize the poses graph (usually around 2 or 3 seconds). To reintegrate all frames to TSDF again, it might take a few minutes depending of the size of the dataset. As shown before, each TSDF integration takes 251 milisseconds to process.

Figure 5.12: Teddy Turntable: Proposed, Proposed + Pose Graph and GT

Figure 5.13: Bunny Turntable: Proposed, Proposed + Pose Graph and GT



6 CONCLUSIONS

We have presented an online 3D reconstruction pipeline based on RGB-D video sequences focused on near-range captures. The core of the proposed approach is an iterative pose alignment procedure that considers a weighted combination of color and depth images aiming to reduce the inherent noise of depth sensors that also accounts for possible misalignment between the color and depth sensors. As additional contributions, we perform a rough alignment based on PnP that is able to handle larger motion between adjacent frames of the sequence, and the use of a frame-to-model registration scheme that further reduces the influence of noise in the depth image,

Our experimental results indicate that both the pose and the obtained 3D models with our method are comparable to or better than a state-of-the-art method. A qualitative analysis (visual inspection) indicates that the proposed model is capable of keeping geometric texture and thin structures, and at the same time, avoids color bleeding artifacts that arise due to the misalignment of depth and color sensors. The use of a pose graph optimization step based on loop closure as a post-processing step can refine the online camera poses, leading also to better 3D models.

As future work, the most immediate issue to address is code optimization to accelerate the bilateral filter and the RGB-D alignment performance. In addition, the idea of weighting the correspondences could be improved through the use of machine learning techniques, by recognizing image areas where we could set larger weights in order to ensure the best alignment of point clouds. As an alternative to the projective correspondence implemented in this work, the nearest neighbor correspondence could be used as in Open3D technique, possibly resulting in smoother alignments. A promising experiment would be using machine learning techniques in order to find a model that is able to recognize good and repeatable points of interest in RGB-D data. This would enable the presented pipeline use weighting on regions of images to improve results.

REFERENCES

BADINO, H. et al. Fast and accurate computation of surface normals from range images. In: **2011 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 2011. p. 3084–3091. ISSN 1050-4729.

BAY, H. et al. Speeded-up robust features (surf). **Comput. Vis. Image Underst.**, Elsevier Science Inc., USA, v. 110, n. 3, p. 346–359, jun. 2008. ISSN 1077-3142. Available from Internet: <https://doi.org/10.1016/j.cviu.2007.09.014>.

BERGER, M. et al. A survey of surface reconstruction from point clouds. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2017. v. 36, n. 1, p. 301–329.

BESL, P. J.; MCKAY, N. D. A method for registration of 3-d shapes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 14, n. 2, p. 239–256, Feb 1992. ISSN 0162-8828.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

CALONDER, M. et al. Brief: Binary robust independent elementary features. In: . [S.l.: s.n.], 2010. v. 6314, p. 778–792.

CHEN, Y.; MEDIONI, G. Object modeling by registration of multiple range images. **Image Vision Comput.**, v. 10, p. 145–155, 01 1992.

CURLESS, B.; LEVOY, M. A volumetric method for building complex models from range images. In: FUJII, J. (Ed.). **SIGGRAPH**. ACM, 1996. p. 303–312. ISBN 0-89791-746-4. Available from Internet: <http://dblp.uni-trier.de/db/conf/siggraph/siggraph1996.html#CurlessL96>.

DAI, A. et al. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. **ACM Transactions on Graphics (ToG)**, ACM, v. 36, n. 4, p. 76a, 2017.

DETONE, D.; MALISIEWICZ, T.; RABINOVICH, A. Superpoint: Self-supervised interest point detection and description. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)**. [S.l.: s.n.], 2018. p. 337–33712. ISSN 2160-7508.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: FLEET, D. et al. (Ed.). **Computer Vision – ECCV 2014**. Cham: Springer International Publishing, 2014. p. 834–849. ISBN 978-3-319-10605-2.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 24, n. 6, p. 381–395, jun. 1981. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/358669.358692>.

GAO, X.-S. et al. Complete solution classification for the perspective-three-point problem. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 25, p. 930– 943, 09 2003.

GELFAND, N. et al. Robust global registration. In: . [S.l.: s.n.], 2005.

GIRARDEAU-MONTAUT, D. **CloudCompare: 3D Point Cloud and Mesh Processing Software**. 2003. <http://www.danielgm.net/cc/>. Accessed: December 10, 2019.

GROSSMANN, N.

Extracting Sensor Specific Noise Models — Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, aug 2017. 1. Available from Internet: <https://www.cg.tuwien.ac.at/research/publications/2017/grossmann-2016-baa/>.

HARALICK, R. et al. Review and analysis of solutions of the three point perspective pose estimation problem. **International Journal of Computer Vision**, v. 13, p. 331–356, 12 1994.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. [S.l.]: Cambridge University Press, ISBN: 0521623049, 2000.

IZADI, S. et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In: **Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology**. New York, NY, USA: ACM, 2011. (UIST '11), p. 559–568. ISBN 978-1-4503-0716-1. Available from Internet: <http://doi.acm.org/10.1145/2047196.2047270>.

KERL, C.; STURM, J.; CREMERS, D. Robust odometry estimation for rgb-d cameras. In: **ICRA**. [S.l.]: IEEE, 2013. p. 3748–3754. ISBN 978-1-4673-5641-1.

KHOSHELHAM, K. Extending generalized hough transform to detect 3d objects in laser range data. In: . [S.l.: s.n.], 2007. XXXVI.

KHOSHELHAM, K.; ELBERINK, E. O. Accuracy and resolution of kinect depth data for indoor mapping applications. In: **Sensors 2012, 12, 1437–1454. 2013**. [S.l.: s.n.], 2012. p. 8238.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o(n) solution to the pnp problem. **International Journal of Computer Vision**, v. 81, 02 2009.

LI, S.; XU, C.; XIE, M. A robust o(n) solution to the perspective-n-point problem. **IEEE transactions on pattern analysis and machine intelligence**, v. 34, 01 2012.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In: STONE, M. C. (Ed.). **SIGGRAPH**. ACM, 1987. p. 163–169. ISBN 0-89791-227-6. Available from Internet: <http://dblp.uni-trier.de/db/conf/siggraph/siggraph1987.html#LorensenC87>.

LOW, K. lim. **Linear least-squares optimization for point-to-plane ICP surface registration**. [S.l.], 2004.

LOWE, D. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, p. 91–, 11 2004.

Lu, C. .; Hager, G. D.; Mjolsness, E. Fast and globally convergent pose estimation from video images. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 6, p. 610–622, June 2000. ISSN 1939-3539.

MA, Y. et al. **An Invitation to 3-D Vision: From Images to Geometric Models**. [S.l.]: SpringerVerlag, 2003. ISBN 0387008934.

MAGNUSSON, M. **The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection**. Thesis (PhD), 12 2009.

MAIMONE, M.; CHENG, Y.; MATTHIES, L. Two years of visual odometry on the mars exploration rovers. **J. Field Robotics**, v. 24, p. 169–186, 03 2007.

MAKADIA A. PATTERSON, K. D. A. Fully automatic registration of 3d point clouds. In: **2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)**. [S.l.: s.n.], 2006. v. 1, p. 1297–1304. ISSN 1063-6919.

MALLICK, T.; DAS, P. P.; MAJUMDAR, A. K. Characterizations of noise in kinect depth images: A review. **IEEE Sensors Journal**, v. 14, n. 6, p. 1731–1740, June 2014. ISSN 2379-9153.

MARCHAND, E.; UCHIYAMA, H.; SPINDLER, F. Pose estimation for augmented reality: A hands-on survey. **IEEE Transactions on Visualization and Computer Graphics**, v. 22, n. 12, p. 2633–2651, Dec 2016. ISSN 2160-9306.

MERRITT, E. Explicit three-point resection in space. **Photogrammetric Engineering, vol. 15, no. 4, pp. 649–655, 1949**, 1988.

Mur-Artal, R.; Montiel, J. M. M.; Tardós, J. D. Orb-slam: A versatile and accurate monocular slam system. **IEEE Transactions on Robotics**, v. 31, n. 5, p. 1147–1163, Oct 2015. ISSN 1552-3098.

NEWCOMBE, R. A. et al. Kinectfusion: Real-time dense surface mapping and tracking. In: **2011 10th IEEE International Symposium on Mixed and Augmented Reality**. [S.l.: s.n.], 2011. p. 127–136.

NGUYEN, C.; IZADI, S.; LOVELL, D. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In: . [S.l.: s.n.], 2012.

OLSON, C. Efficient pose clustering using a randomized algorithm. **International Journal of Computer Vision**, v. 23, 03 1999.

PARIS, S.; DURAND, F. A fast approximation of the bilateral filter using a signal processing approach. **International Journal of Computer Vision**, v. 81, n. 1, p. 24–52, Jan 2009. ISSN 1573-1405. Available from Internet: <https://doi.org/10.1007/s11263-007-0110-8>.

PARK, J.; ZHOU, Q.; KOLTUN, V. Colored point cloud registration revisited. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. p. 143–152. ISSN 2380-7504.

Rematas, K. et al. Soccer on your tabletop. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 4738–4747. ISSN 2575-7075.

Reynolds, M. et al. Capturing time-of-flight data with confidence. In: **CVPR 2011**. [S.l.: s.n.], 2011. p. 945–952.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2564–2571.

RUSINKIEWICZ, S.; LEVOY, M. Efficient variants of the icp algorithm. In: . [S.l.: s.n.], 2001. p. 145–152. ISBN 0-7695-0984-3.

RUSU, R.; COUSINS, S. 3d is here: Point cloud library (pcl). In: . [S.l.: s.n.], 2011.

RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast point feature histograms (fpfh) for 3d registration. In: **2009 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 2009. p. 3212–3217. ISSN 1050-4729.

RUSU, R. B. et al. Aligning point cloud views using persistent feature histograms. In: **2008 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2008. p. 3384–3391. ISSN 2153-0866.

SCARAMUZZA, D.; FRAUNDORFER, F. Visual odometry [tutorial]. **IEEE Robot. Automat. Mag.**, v. 18, p. 80–92, 12 2011.

SCHWEIGHOFER, G.; PINZ, A. Globally optimal o(n) solution to the pnp problem for general camera models. In: . [S.l.: s.n.], 2008.

SERAFIN, J.; GRISETTI, G. Nicp: Dense normal based point cloud registration. In: **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2015. p. 742–749.

SILVEIRA, T. L. T. d.; JUNG, C. R. Perturbation analysis of the 8-point algorithm: A case study for wide fov cameras. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019.

SLAVCHEVA, M. et al. SDF-2-SDF: Highly Accurate 3D Object Reconstruction. In: **European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2016.

STEINBRUCKER, F.; STURM, J.; CREMERS, D. Real-time visual odometry from dense rgb-d images. In: **2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)(ICCVW)**. [s.n.], 2012. v. 00, p. 719–722. Available from Internet: <doi.ieeecomputersociety.org/10.1109/ICCVW.2011.6130321>.

STURM, J. et al. Copyme3d: Scanning and printing persons in 3d. In: WEICKERT, J.; HEIN, M.; SCHIELE, B. (Ed.). **Pattern Recognition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 405–414. ISBN 978-3-642-40602-7.

Sturm, J. et al. A benchmark for the evaluation of rgb-d slam systems. In: **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2012. p. 573–580.

Sungjoon Choi; Zhou, Q.; Koltun, V. Robust reconstruction of indoor scenes. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 5556–5565. ISSN 1063-6919.

TOMBARI, F.; SALTI, S.; STEFANO, L. D. Unique signatures of histograms for local surface description. In: **Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III**. Berlin, Heidelberg: Springer-Verlag, 2010. (ECCV'10), p. 356–369. ISBN 3-642-15557-X, 978-3-642-15557-4. Available from Internet: <http://dl.acm.org/citation.cfm?id=1927006.1927035>.

WANG, P. et al. An efficient solution to the perspective-three-point pose problem. **Comput. Vis. Image Underst.**, Elsevier Science Inc., USA, v. 166, n. C, p. 81–87, jan. 2018. ISSN 1077-3142. Available from Internet: <https://doi.org/10.1016/j.cviu.2017.10.005>.

Xiao-Shan Gao et al. Complete solution classification for the perspective-three-point problem. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 25, n. 8, p. 930–943, Aug 2003. ISSN 1939-3539.

YI, K. M. et al. LIFT: learned invariant feature transform. **CoRR**, abs/1603.09114, 2016. Available from Internet: <http://arxiv.org/abs/1603.09114>.

Zhou, L.; Kaess, M. An efficient and accurate algorithm for the perspective-n-point problem. In: **2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2019. p. 6245–6252. ISSN 2153-0858.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Fast global registration. In: . [S.l.: s.n.], 2016. v. 9906.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Open3D: A modern library for 3D data processing. **arXiv:1801.09847**, 2018.