

Real Time Dense Tracking and Mapping Using a Monocular Camera

A thesis presented
by
Andrés Alejandro Díaz Toro

Supervised by
Prof. Lina María Paz Pérez
Prof. Eduardo Francisco Caicedo Bravo

In partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of
Electrical and Electronic Engineering

UNIVERSIDAD DEL VALLE
Department of Engineering
Cali, Valle del Cauca
September, 2018

This thesis is dedicated to
my loved mother
for been my pillar and motivation

Acknowledgements

I thank my tutors, Prof. Lina Paz and Prof. Eduardo Caicedo, for all the advices, discussions, and valuable time, devoted to the development of this thesis. Their ideas have been of significant importance and transcendence in this work. I thank Prof. Pedro Piniés that enriched the contributions of this thesis with fruitful hours of teaching and discussions. I also thank my family that supported me with advices, love and permanent motivation. Finally, I thank Perception and Smart Systems Group PSI of Universidad del Valle and the Mobile Robotics Group MRG of Oxford, for all the experiences that I lived with them and Fundación CEIBA for the financial support that made this work possible.

© 2018 - *Andrés Alejandro Díaz Toro*
ALL RIGHTS RESERVED.

Abstract

Algorithms for localization and map building (*Simultaneous Localization and Mapping*, SLAM) estimate the robot pose when it wanders an unknown environment and at the same time build a map of this environment in an incremental way. The initial objective of SLAM systems was getting autonomous robots [83]. However, vision-based SLAM systems have a great potential outside the field of mobile robotics, such as augmented reality, wearable devices, user interfaces, surgical procedures, among others, because they give a camera the capability to work as a sensor of 3D position of general purpose.

Most of the visual SLAM systems use sparse features such as corners [16], [58], edges [45], [81] or planes [78], and have as principal goal to localize the camera. Dense techniques for localization and mapping do not use sparse features of the environment but all the pixels of an image, getting high-quality maps. For achieving real time performance concepts such as keyframes, bundle adjustment and parallel computing are used. Depth maps built with monocular cameras are affected by conditions commonly found in real environments as low-textured surfaces, very reflective or translucent objects and changes in light conditions. Even when a depth sensor is used, depth maps can be noisy and incomplete due to dark, very reflective and translucent objects, out of range surfaces and occlusions. This situation affects negatively the dense reconstruction got by fusing raw depth maps.

In this doctoral thesis we propose a novel technique for enhancing depth maps built with a monocular camera or obtained from a depth sensor, by integrating depth data of an optimal model (shape prior) using a variational technique that besides merging depth data of two sources, makes inpainting and denoising, generating a significant impact in the appearance and in accuracy of the dense reconstruction, specially in the region of the object of interest. We present two approaches for dense localization: frame-to-frame and frame-to-model approaches, the algorithm for building a depth map with a monocular camera, the optimization of the model w.r.t. pose, scale and shape, the proposed technique for integrating the shape prior and the fusion of enhanced depth maps for creating a dense reconstruction of the scene. We also present the conclusions in each chapter, the general conclusions and future work.

Keywords. Dense information, localization, mapping, SLAM, variational techniques, monocular camera, shape priors, 3D level-set embedding function, Gaussian process latent variable models.

Resumen. Los algoritmos de localización y construcción de mapas (Simultaneous Localization and Mapping, SLAM) estiman la posición de un robot que navega en un entorno desconocido y al mismo tiempo construye un mapa en forma incremental. El objetivo inicial de los sistemas de SLAM fue obtener robots autónomos [83]. Sin embargo, los sistemas de SLAM tienen gran potencial por fuera de la robótica móvil, como en realidad aumentada, dispositivos portables, interfaces de usuario, procedimientos quirúrgicos, entre otros, debido a que dan a la cámara la capacidad de trabajar como sensor de posicionamiento 3D de propósito general.

La mayoría de los sistemas de SLAM visual usan características dispersas como esquinas [16], [58], bordes [45], [81] o planos [78], y tienen como principal objetivo localizar la cámara. Las técnicas que utilizan información densa para localización y mapeo, no utilizan características dispersas sino todos los píxeles de las imágenes, obteniendo mapas de alta calidad. Conceptos como marcos clave, ajuste de rayos, y computación en paralelo, son usados con el fin de lograr desempeño en tiempo real. Los mapas de profundidad construidos con cámaras monoculares son afectados por condiciones comúnmente encontradas en ambientes reales como superficies con poca textura, objetos muy reflectivos o traslúcidos y cambios en la iluminación. Incluso cuando se emplea un sensor de profundidad, la información puede ser ruidosa e incompleta debido a objetos opacos, muy reflectivos o traslúcidos, superficies fuera del rango de medida y oclusiones. Esta situación afecta negativamente la reconstrucción densa obtenida al fusionar los mapas de profundidad sin procesar.

En esta tesis doctoral proponemos una técnica novedosa para mejorar los mapas de profundidad construidos con una cámara monocular u obtenidos con una cámara de profundidad, integrando información de un modelo óptimo (*shape prior*) usando una técnica variacional que además de unir datos de profundidad de dos fuentes, hace *inpainting* y *denosing*, generando un impacto importante en la apariencia visual y en la exactitud de la reconstrucción densa, especialmente en la región del objeto de interés. Presentamos dos enfoques para localización usando información densa: marco-marco y marco-modelo, el algoritmo para construir mapas de profundidad con una cámara monocular, la optimización del modelo con respecto a la posición, escala y forma, la técnica propuesta para integrar el *shape prior* y la fusión de mapas de profundidad mejorados para crear una reconstrucción densa. También presentamos las conclusiones en cada capítulo, las conclusiones generales y trabajo futuro.

Palabras clave. Información densa, localización, mapeo, SLAM, técnicas variacionales, cámara monocular, *shape priors*, 3D *level-set*, modelos de procesos Gaussianos y variable latente.

CONTENTS

1	Introduction	19
1.1	Evolution of the SLAM systems	20
1.2	Visual SLAM	21
1.2.1	Filter-based SLAM	21
1.2.2	Keyframe-based SLAM	22
1.2.3	Dense SLAM	23
1.3	Definition of the problem	24
1.4	Importance of this thesis	24
1.5	Objectives	26
1.6	Thesis outline	26
2	Dense localization Techniques	28
2.1	Introduction	28
2.2	Related Work	29
2.3	Dense Localization with Frame-to-Frame Approach	31
2.3.1	Keyframe Selection	31
2.3.2	Initial Pose Estimation	32
2.3.3	Pose Refinement	33
2.4	Dense Localization with Frame-to-Model Approach	37
2.4.1	Rasterization	38
2.4.2	Dense ICP	39
2.5	Results	42
2.5.1	Results with the Frame-to-Frame Approach	42
2.5.2	Results with Frame-to-Model Approach	47
2.6	Conclusions	50

3	Depth Map building using a Monocular Camera	52
3.1	Introduction	52
3.2	Related Work	52
3.3	Building an Initial Depth Map	53
3.4	Refining the initial depth map with variational methods	55
3.4.1	The decoupling approach	56
3.5	Results	58
3.5.1	Results with synthetic data	58
3.5.2	Results with real data	62
3.6	Conclusions	66
4	Shape Prior Estimation	68
4.1	Introduction	68
4.2	Related work	70
4.3	Alignment and TSDF estimation of the 3D Models	71
4.4	Compression of the Embedding Function	73
4.5	Dimensionality Reduction	74
4.6	Shape Optimization	76
4.7	Pose and Scale optimization	78
4.8	Results	80
4.8.1	Results in Shape optimization with synthetic data	81
4.8.2	Results in Pose optimization with synthetic data	82
4.8.3	Results in Scale Optimization with synthetic data	83
4.8.4	Results in Pose, Scale and Shape with synthetic data . .	84
4.8.5	Results in Pose, Scale and Shape with Real Data (Kinect)	86
4.8.6	Results with depth map built with real data (monocular camera)	91
4.9	Conclusions	95
5	Shape Prior Integration	97
5.1	Introduction	97
5.2	Related work	98
5.3	Merging, Inpainting and Denoising of Depth maps	100
5.3.1	Primal-Dual algorithm	100
5.3.2	Putting together depth map estimation and shape prior integration	102

5.4	Results	104
5.4.1	Results with a depth camera	104
5.4.2	Results with monocular camera	107
5.4.3	Results with Synthetic Depth maps	111
5.5	Conclusions	114
6	Dense Reconstruction with Shape Prior	116
6.1	Introduction	116
6.2	Related work	117
6.3	Volumetric Fusion	118
6.3.1	Volumetric structure	119
6.3.2	Voxel update	119
6.3.3	Marching cubes	121
6.4	Results	121
6.4.1	Results of Dense Localization and Mapping	121
6.4.2	Results using Shape Prior Data	122
6.5	Conclusions	134
7	Conclusions	135
7.1	Results, Contributions and Future Work	137
7.2	Papers, academic events and short-term research abroad	140

LIST OF FIGURES

1.1	Proposed system for dense localization and mapping using shape priors. On the left, the system works with a monocular camera. On the right, the system works with a depth camera.	27
2.1	The camera trajectory is drawn in cyan. Keyframes r_j are drawn with their local coordinate systems. They are selected based on a time criteria, e.g. for every nine images a new keyframe is initialized.	32
2.2	(a) Perfect ray-traced RGB Image and (b) depth map from the dataset of A. Handa [36]. They are used for computing the initial pose estimation and for the posterior refinement.	33
2.3	Current image $\mathbf{I}_c$ and reference image $\mathbf{I}_r$ with their corresponding local and global coordinate systems. The transformations that relate coordinate systems are drawn in red.	34
2.4	Alignment between reference and warped images. The residual depends on the estimation in motion.	34
2.5	(a) 3D model and its visualization on a screen. (b) depth map obtained from Z-buffer of OpenGL. Dark regions correspond to space outside the grid of voxels.	38
2.6	Diagram of the frame-model based tracking. The vertex map in $k - 1$, obtained from Z-buffer of OpenGL, is aligned with the vertex map, computed from the measured depth map.	39
2.7	Graphical interpretation of the point-plane optimization. The vertex $\mathbf{V}_k$ (in red), obtained from the measurement, is moved in order to reduce the distance to the tangent plane of vertex $\hat{\mathbf{V}}_{k-1}$ obtained from the model.	40
2.8	(a) Trajectory and (b) orientation of the camera for the 60 fps dataset. The positions of keyframes are drawn as blue points in (a).	43
2.9	Error in camera pose (position and orientation). The keyframe positions are drawn as blue points in (a).	44
2.10	Dense Map built using depth maps of keyframes. The trajectory is drawn in orange and keyframes are blue spheres.	45

2.11	Comparison between using only one image and a four-level pyramid. (a) requires 76 iterations with an image of 480×640 pixels for converging. (b) requires 2 iterations in the lower level (image of 480×640 pixels) for converging.	46
2.12	3D reconstruction resulting from fusing synthetic depth maps and color of the scene, from the estimated camera pose.	48
2.13	Error in camera pose. (a) Error in camera position. (b) Error in camera orientation. The errors remain small with low drift after completing the whole trajectory.	48
2.14	Dense reconstruction for the sequence fr1/desk of TUM, with color from the scene.	49
3.1	Projection of a 3D point referenced to the coordinate system r into the image plane c , using the transformation T_{cr}	54
3.2	(a) Pixel in a high-textured region produces a clear minimum, while (b) pixel in a low-textured region produces a flat cost.	55
3.3	(a) Gray image from reference frame, (b) ground truth of the depth map from reference frame, (c) initial depth estimation and (d) solution in depth after 200 iterations of primal-dual algorithm. Red represents high depth while blue represents low depth.	59
3.4	Performance of the algorithm for an x-slice ((a)-(d)) and y-slice ((e)-(h)), when just the data term is used and when both the data term and the regularisation term are used.	60
3.5	Absolute error in depth [m] got (a) using just the data term and (b) using both the data term and the regularisation term (clear regions represent high errors). (c) Error distribution in both cases.	61
3.6	(a) Gray image from reference frame (real data), (b) depth map taken with kinect sensor for reference frame (used for estimating depth accuracy of solution), (c) initial depth map built using the photometric error and (d) refined depth map after 200 iterations of primal-dual algorithm. Red represents high depth while blue represents low depth.	63
3.7	Performance of the algorithm for an x-slice ((a)-(d)) and y-slice ((e)-(h)), when just the data term is used and when both the data term and the regularisation term are used.	64
3.8	Absolute error in depth [m] got (a) using just the data term and (b) using both the data term and the regularisation term (clear regions represent high errors). (c) Error distribution in both cases.	65

4.1	Main pipeline of the system for shape prior estimation using built depth maps.	69
4.2	Illustrative description of the dimensionality reduction between the high dimensional space of the compressed 3D models and the space of the latent variables.	69
4.3	Alignment and TSDF estimation. (a) Base model. (b) Model i . (c) Aligned base model and model i . (d) Model i and its corresponding explicit representation got by applying marching cubes to the TSDF values of the volumetric grid.	72
4.4	3D models of cars with different shapes.	73
4.5	3D model of fig. 4.4(b) using different number of coefficients in each dimension (degree of compression).	74
4.6	(a) Energy and (b) latent space for embedding functions of cars. Low variance regions are drawn in blue while high variance regions are drawn in red. More likely valid shapes are obtained in blue areas.	76
4.7	Initial coordinate system q , object coordinate system o and world coordinate system w	79
4.8	Reference model. (a) Downsampled 3D point cloud. (b) Rendered model (ground truth in shape).	81
4.9	Evolution of the shape in the optimization process. The search is done in the latent space and the model is obtained with the inverse DCT. The point cloud is drawn in red.	81
4.10	(a) Evolution of the energy. (b) Search of the latent variable. The initial latent variable and for iterations from 1 to 14 are drawn in clear orange. The latent variable for iteration 15 is drawn in dark orange. The ground truth in latent variable is drawn in green.	82
4.11	Evolution of the pose of the embedding function in the optimization process. The point cloud is drawn in red.	83
4.12	Evolution of the energy for pose optimization using Levenberg-Marquardt.	83
4.13	Evolution of the position and orientation with respect to the ground truth.	83
4.14	Evolution of the scale of the embedding function in the optimization process. The point cloud is drawn in red.	84
4.15	(a) Evolution of the energy and (b) the scale of the embedding function in the optimization process. The ground truth in scale is $s = 1$	84

4.16	Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process. The point cloud is drawn in red.	85
4.17	Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt.	86
4.18	(a) RGB and (b) depth images of a toy car over a desk (dark red represents high depth). The infrared light emitted by the Kinect is absorbed by black surfaces, producing regions without data.	87
4.19	(a) Mask of the segmented car. (b) Point cloud of region associated to the segmented area (lateral side of the toy car). . . .	87
4.20	World (in blue) and camera (in red) coordinate systems. . . .	87
4.21	(a) Point cloud of the toy car with the normal to the supporting plane. (b) Angles that move the embedding function to the flat surface.	88
4.22	Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process, for real data using the Kinect. The point cloud is drawn in red.	90
4.23	(a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.	90
4.24	Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt. The data comes from the Kinect 1.0.	91
4.25	(a) Reference image in gray, (b) depth for reference image coming from sensor, (c) initial depth got by solving the data term and (d) solution in depth got by using the primal-dual algorithm. Red represents high depth while blue represents low depth.	92
4.26	(a) Mask of the segmented car. (b) Point cloud of region associated to the segmented area and norm to the supporting plane. . . .	92
4.27	Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process, for real data coming from the built depth map using a monocular camera. The point cloud is drawn in red.	94
4.28	(a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.	94
4.29	Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt. The data comes from a depth map built with a monocular camera. . . .	95
5.1	(a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.	104

5.2	Depth maps obtained from (a) the sensor and (b) the model. Red represents high depth while green represents low depth. .	105
5.3	Masks that define the inpainting domain for depth maps obtained from (a) the sensor and (b) the model.	105
5.4	(a) Initial depth map $D_s(\mu)$. Sought depth map $D_f(\mu)$ for iterations (b) 40, (c) 100 and (d) 200 of the primal-dual algorithm. Red represents high depth while green represents low depth.	106
5.5	Depth for x-slice, row 370. The smoother and complete solution is drawn in cyan while the incomplete depth from the sensor is drawn in dark blue. The two vertical lines encompass the car.	107
5.6	(a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.	107
5.7	Depth maps (a) built with a monocular camera and (b) obtained from the model. Red represents high depth while blue represents low depth.	108
5.8	(a) Depth map built with a monocular camera. Sought depth map $D_f(\mu)$ for iterations (b) 40, (c) 70 and (d) 100 of the primal-dual algorithm. Red represents high depth while green represents low depth.	108
5.9	Depth for row 210; the data of the built depth map is drawn in dark blue, the smoother and complete solution resulting from merging data is drawn in cyan and the incomplete depth from the sensor is drawn in green. The vertical lines encompass the car.	109
5.10	(a) Initial depth map computed by solving eq. (3.7), (b) Resulting depth map after 200 iterations of the algorithm for refining and shape prior integration, (c) Depth for row 210; data from the built depth map (dark blue), the smoother and complete solution resulting from merging the shape prior (cyan) and the incomplete depth from the sensor (green).	110
5.11	Synthetic data used for estimating the accuracy of the algorithm that makes denoising, inpainting and merging. (a) RGB image with noise, (b) Depth map (ground truth), (c) Depth map with noise and missing data (23.65%), and (d) Depth map with noise and more missing data (40.02%).	112
5.12	Results of the alignment process between a 3D-level set and the point cloud. (a) For depth map with missing data of 23.65% and (b) For depth map with missing data of 40.02%. Synthetic depth maps obtained from models for (c) 23.65% and (d) 40.02% of missing data.	113

5.13	The first row contains images for the depth map with missing data of 23.65% while the second row for the depth map with missind data of 40.02%. From left to right: ground truth, depth map with noise and missing data, our results, results using [67].	114
6.1	Notation for applying marching cubes to the volumetric structure.	119
6.2	Ray that passes through the voxel i (dark blue), range of uncertainty of the measured depth (yellow), surface been scanned (light blue) and difference between the measured depth and the z-coordinate for a voxel i (in camera frame). The last value is used for computing the TSDF.	120
6.3	Data used for quantifying the accuracy in geometry.	123
6.4	Point Cloud (front and back views) with color representing the distances between the point cloud and the reference model (scale in centimeters).	124
6.5	Comparison between dense reconstructions got with a depth sensor using and not using shape prior. (a) Colored reconstruction using shape prior, (b) Colored reconstruction without shape prior, (c) Uncolored reconstruction using shape prior, (d) Uncolored reconstruction without shape prior, (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.	127
6.6	Comparison between dense reconstructions got with a monocular camera, using and not using shape prior. (a) Colored reconstruction using shape prior, (b) Colored reconstruction without shape prior, (c) Uncolored reconstruction using shape prior, (d) Uncolored reconstruction without shape prior, (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.	128
6.7	Comparison between the reference reconstruction (depth sensor and shape prior) and the dense reconstruction got with the monocular camera and without shape prior. (a) Colored reconstruction used as reference, (b) Colored reconstruction got with monocular camera and without shape prior, (c) Uncolored reconstruction (reference), (d) Uncolored reconstruction (monocular and without shape prior), (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.	129

6.8	Comparison between the reference reconstruction (depth sensor and shape prior) and the dense reconstruction got with the monocular camera and with shape prior. (a) Colored reconstruction used as reference, (b) Colored reconstruction got with monocular camera and with shape prior, (c) Uncolored reconstruction (reference), (d) Uncolored reconstruction (monocular and with shape prior), (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view), and (g) histogram of distances.	130
6.9	Comparison between segmented car (from dense reconstruction got with monocular camera and no shape prior) and reference reconstruction (with depth sensor and shape prior). (a) Region segmented from the reconstruction got with monocular camera and no shape prior, (b) Resulting segmented car, (c) Reference reconstruction and segmented car, (d) Segmented car aligned to the reference reconstruction, (e) Another view of aligned car, (f) segmented car colored with data associated to the distances and (g) histogram of distances.	131
6.10	Comparison between segmented car (from dense reconstruction got with monocular camera and shape prior) and reference reconstruction (with depth sensor and shape prior). (a) Region segmented from the reconstruction got with monocular camera and shape prior, (b) Resulting segmented car, (c) Reference reconstruction and segmented car, (d) Segmented car aligned to the reference reconstruction, (e) Another view of segmented car aligned to the reference car and (f) segmented car colored with data associated to the distances to the reference reconstruction.	132
6.11	Comparison between histograms of distances between the reference reconstruction and the aligned and segmented car 1. from reconstruction with monocular camera and NO shape prior (red) and 2. from reconstruction with monocular camera and shape prior (blue).	133

LIST OF TABLES

2.1	Summary of accuracy in camera position	44
2.2	Summary of accuracy in camera orientation	45
2.3	Information about convergence using a coarse-to-fine scheme	46
2.4	Summary of accuracy in camera pose	49
2.5	TUM RGBD dataset. Comparison of translation RMSE (m)	49
2.6	Processing time of the main modules	50
3.1	Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for x and y slices.	61
3.2	Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for the whole image.	61
3.3	Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for x and y slices. The reference is the depth map from the sensor.	65
3.4	Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for the whole image. The reference is the depth map from the sensor.	66
3.5	Processing time for building a depth map with a monocular camera.	66
5.1	Comparison of accuracy in depth, in the car area, for the depth map built with the monocular camera DMB and the depth maps resulting from merging the shape prior data in sequential SPI and simultaneous way DMB-SPI.	110
5.2	Processing time for depth map building DMB, shape prior integration SPI and both depth map building and shape prior integration simultaneously DMB-SPI.	111
5.3	Comparison of RMSE [m] of our algorithm and [67].	113

6.1	Summary of accuracy in geometric data (taken from Cloud-Compare software [33]).	122
-----	---	-----

CHAPTER 1

INTRODUCTION

The problem of building a map and simultaneously localize a vehicle (*Localization and Mapping*, SLAM) was considered at the end of the 90s as the task of giving a mobile robot, located in an unknown environment, the ability to build a map in incremental way and at the same time estimate its location w.r.t. this map [23].

A probabilistic approach was proposed initially [83], [82], and since then, it has been used in a wide variety of systems, presenting through time some variations (the kind of sensor, the kind of filter, the features or descriptors, the motion model, the measurement model, the data association, the matching validation, the kind of environment, among others). This approach, coming from the field of mobile robotics, is based on filters such as the EKF-SLAM algorithm that uses the Extended Kalman Filter EKF, the Fast-SLAM algorithm based on particle filter PF and the Sparse Extended Information Filter SEIF.

Other approach, coming from the field of computer vision, is based on concepts as key-frames, bundle adjustment, projective geometry, techniques of structure from motion SFM and visual odometry VO, for building a representation of the environment and estimate the camera pose, with global consistency. With the increase in computing power, the second approach has presented considerable advances and impressive results, that overcome in accuracy and in quality the maps got with filter-based systems [89].

In this chapter we present the evolution of the SLAM systems, general properties of visual SLAM systems, filter-based systems, keyframe-based systems, and dense systems. Next, we present the definition of the problem of study, the importance of this thesis, the objectives and the thesis outline.

1.1 EVOLUTION OF THE SLAM SYSTEMS

The autonomy of mobile robots was the main goal of the systems developed from the concept of SLAM that emanated in the IEEE conference on Robotics and Automation, hold in San Francisco in 1986. In this meeting, probabilistic methods were proposed for tackling the SLAM problem. The most outstanding researches were Peter Cheeseman, Jim Crowley, Hugh Durrant-Whyte, Raja Chatila, Oliver Faugeras and Randal Smith.

The first SLAM systems used ring of sonars, lasers and stereo cameras, coupled to terrestrial robots, and the Kalman Filter as estimator of the state vector, makes up of the robot pose and the features position. The sonar has big lobules that constraint the measurements due to its high uncertainty in the orientation of the data [51], [92]. The systems with the best performance employed lasers but the maps were in 2D [35], [34], [63] .

The increase in computing power and the advances in computer vision (features detection, data association, active search) made the camera the main sensor in SLAM. The authors of [15] implemented the first SLAM system with stereo vision based on the EKF filter and processing data at 5 Hz, been able to build a map and control a mobile robot. The authors of [39] developed a system for mapping terrains using stereo vision as unique sensor and the EKF. This implementation was sequential but did not run in real time. An outstanding monocular SLAM system that uses the EKF is [14] and its posterior work [16]. An outstanding particle filter-based SLAM system that uses a laser is [56], and its enhanced version [57] that was implemented for first time in a monocular system in [24]. Other systems have excelled for using novel techniques such as Gaussian sum approximation [84], SIFT features and FastSLAM [79], patch matching and planes-based maps [78], parametrization with the inverse depth [8], large scale SLAM [66], bio-inspired techniques [54], based on graphs [48], among others.

The first monocular SLAM systems, successfully developed as [15], [16] and [8], have as main objective the accurate estimation of the camera pose for navigation tasks and not a high quality reconstruction of the environment. In these systems, few and sparse features of the scene are used. The sparse monocular system of [16] enables applications of augmented reality but the interaction with virtual objects is poor since most data of the images is not used. The work [44] (PTAM) is the first one to propose a new approach, different to filter-based systems, based on bundle adjustment of key-frames (sparse frame-to-frame tracking approach), where tracking and mapping are separated (run in two parallel threads). This work, with up to 6000 points and 120 keyframes, defined a transition to dense representations of the scene, like [60], [90], and [62]. Other kind of sensor, the RGB-D camera that delivers both intensity and depth data at high frame rates, has been extensively used

for building dense reconstructions and/or estimating the camera pose with dense techniques in real time, as [61], [40] and [87]. These systems will be analysed in section 1.2.3 and in several chapters of the thesis.

1.2 VISUAL SLAM

Most SLAM systems employ visual sensors, as a unique sensor or in combination with other sensors as depth cameras or inertial ones, due to its appealing properties; these sensors are not limited to motions in a plane but it can cope with 6 DOF, appropriate for working coupled to terrestrial, aerial [31] and aquatic robots [43], and even can be moved by hand by a person [16]. Moreover, they have low weight, small size, and low cost, and provide more data than lasers and sonars (e.g. the texture can be used successfully for recognition and data association). For an appropriate performance, they require constant light conditions, rich texture and that consecutive images have common elements (overlap).

Visual systems can use a perspective monocular camera [16], omnidirectional monocular camera [53] or a multi-camera array [65]. Monocular systems provide accurate information about the orientation of the points and an indirect measurement of depth, which does not have scale and requires multiple observations with the sensor in motion. The scale must be defined by observing a pattern of known size or integrating other sensors [77]. The advantages of monocular systems are: unlike stereo cameras, can work with close and distant objects and they are available in mobile devices widely used as cellphones and tablets.

1.2.1 FILTER-BASED SLAM

The SLAM problem can be represented in probabilistic way as a density of a joint probability, comprised of the robot state $\mathbf{X}$ in time k and of the features position $\mathbf{Y}$, given all the observations $\mathbf{z}_{1:k}$ and all the commands of control $\mathbf{u}_{1:k}$

$$p(\mathbf{X}_k, \mathbf{Y} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}). \quad (1.1)$$

The robot state $\mathbf{X}_k$ and the environment state $\mathbf{Y}$, make up the state vector due to its high correlation (the error in the robot pose produces a common error in the estimations of the features position). This correlation grows monotonically as new observations are done, improving through time the knowledge of the relative position of features, in spite of the robot motion [12]. The covariance matrix is in charge of summarizing and propagating the uncertainty

of the robot and map information.

In order to describe the effects of the control and observations, a transition model $P(\mathbf{X}_k|\mathbf{X}_{k-1}, \mathbf{u}_k)$ and an observation model $P(\mathbf{Z}_k|\mathbf{X}_k, \mathbf{Y})$ are required, respectively. The most common way to represent these models is through a model in state space with additive Gaussian noise, known as the Extended Kalman Filter EKF, like is done in [8] with a monocular camera and inverse depth. An alternative representation consists in describing the transition model as a set of samples of a more general Non-Gaussian probability distribution. This is done with the Rao-blackwellised particle filter or Fast-SLAM, as in [24]. Other option is the Extended Information Filter [93]. In the remaining of this section, we will focus on the EKF filter.

The main steps of the EKF are prediction and update. The prediction is based on the transition model and the update is based on the observation model. When the state vector and covariance matrix are updated, the camera pose at time k and the features position are also updated (due to the high correlation between all the features).

The computational cost of filter-based systems for propagating the joint probability distribution (defined by the state vector and the covariance matrix) every time a new observation is done and the current state is updated, increases considerably due to its quadratic dependence with the number of features. This fact is the main problem of the EKF-based systems since the number of features n managed in the map is limited drastically, like in [16] and in ours [21], [19], making these maps inappropriate for applications that require high quality representations of the environment as scene understanding in mobile robotics or augmented reality where a rich interaction with the virtual objects is needed. In [1] four methods for tackling the computational problem are described: state augmentation [99], partitioned updates [47], [92], sparsification [17], [27], [30] and sub-maps [66], [71]. Although good results have been obtained, the focus keeps been the accuracy in the camera trajectory and not the quality of the map.

1.2.2 KEYFRAME-BASED SLAM

SLAM methods that do not use filters carry out in parallel way (in two or more CPU cores or in graphics processing units) the creation of the sparse [44], semi-dense [25] or dense maps [62] and the estimation of the pose of keyframes using techniques of visual odometry VO. When the frame-to-frame approach for tracking the camera is used, either with sparse features or with dense information, the consistency of the map is achieved by bundle adjusting the data of the keyframes. On the other hand, when the frame-to-model ap-

proach is used, bundle adjustment is not needed for simple trajectories (short trajectories with the camera looking at the same scene) since the model, with gathered data of the scene, is continuously observed. This fact will be analysed in chapter 2.

The advantage of keyframes is that discarding certain images and optimizing just the information of keyframes, which can be done at lower frame rates, eliminate the need of propagating the joint probability distribution, speeding up the SLAM process, achieving performance in real time. With this approach, just information of keyframes is included into the map [44]; the other images are used for estimating the relative motion between them but this information at the end is discarded.

1.2.3 DENSE SLAM

Systems that use features, either filter-based or keyframe-based, have an important limitation: only information of these features can be used, discarding most of the image data. By contrast, dense techniques use all the image data, optimizing directly the intensity of the whole image (all pixels), achieving high accuracy and robustness, specially in environments with few high-quality features. Moreover, dense maps provide much more information than the sparse ones, which is very useful in robotics and in augmented reality applications.

The concept of keyframes, initially applied to build sparse maps, has been used for dense systems by incorporating equipments with high computing power. The monocular system [60] is the first one in creating dense representations based on techniques of SFM of PTAM [44] (sparse map with thousand of points and sparse frame-to-frame tracking approach using bundle adjusted keyframes). An initial base surface that approximates the real scene is fitted to the 3D point cloud created with SFM. This model permits to make predictions (synthetic images) that are compared with observed images (keyframe and a set of comparison views). The vertices of the base surface are updated (deforming the surface) according to the correspondences between these images, computed with variational optical flow. The system [90] creates in straight way depth maps for selected keyframes using variational techniques and PTAM [44] for tracking a monocular camera. Depth maps are computed by sequential convex optimization by means of a primal-dual algorithm, a decoupling term and a thresholding scheme. DTAM (Dense Tracking and Mapping) [62] is the first monocular system that builds depth maps for selected keyframes and at the same time estimates the camera pose by whole image alignment against the entire dense model (frame-to-model tracking approach). The depth maps are computed by solving an energy functional using a non-convex optimization, with a decoupling term, the primal-dual algorithm and exhaustive search.

There are also systems that create dense reconstructions and/or estimate the camera motion with dense information, using a depth camera. The system [40], an extension of the work [87] estimates the motion of an RGBD camera by aligning two consecutive RGBD images (dense frame-to-frame approach). A photometric error between the observed and a synthesized image is minimized using a coarse-to-fine scheme. It includes a probabilistic formulation with a constant velocity model and a robust sensor model based on the t-distribution. KinectFusion [61] is the first system that fuses depth maps streamed from the Kinect 1.0 into a volumetric grid and estimates the camera pose using a coarse-to-fine iterative closest point ICP algorithm (frame-to-model tracking approach) at real time, getting limited drift and high accuracy.

1.3 DEFINITION OF THE PROBLEM

For building a depth map with a monocular camera its location for a set of frames must be known, photo-consistency must be satisfied and the images must have texture. However, real environments present changes in light conditions and low-textured, very reflective or translucent objects. This fact breaks the Lambertian condition (photo-consistency) affecting the estimated depth maps. The regularisation term, like the Total Variation term in a variational framework, tackles this problem to some extent, but under difficult conditions, the estimations have low accuracy.

On the other hand, a depth sensor like the Kinect 1.0 estimates the depth of a scene by projecting a pattern of infrared light and capturing the modified pattern after reflecting on the objects. Dark, very reflective and translucent objects, out of range objects and occlusions produce noise and black areas in the depth map (missing data).

These noisy and/or incomplete depth maps, whether coming from a monocular or a depth camera, are fused for building a dense reconstruction of the scene, affecting drastically the appearance and accuracy of the reconstruction, specially in regions with certain particularities, such as low-textured, very reflective or translucent objects that are common in indoors as walls, windows, glasses, screen of monitors, among others.

1.4 IMPORTANCE OF THIS THESIS

In this section we present the importance of the subject of study, of the kind of sensors, of the specific techniques, and finally, of the contributions developed along this work.

Dense localization and mapping systems do not use sparse features of the environment as corners or edges but all the pixels of an image, achieving to build high-quality reconstructions and high accuracy in camera pose, allowing the system to carry out more complex tasks in mobile robotics such as path planing, object detection and manipulation based on both visual appearance and geometry, scene understanding, and also in other fields such as augmented reality, wearable devices, video games, among others.

The monocular systems do not require neither heavy nor bulky sensors, just a low-cost camera that can be easily coupled to aerial, terrestrial and aquatic vehicles, and even be moved by hand by a person. Moreover, monocular cameras can work in both indoor and outdoor environments and come with popular mobile devices that most people use and with an increasing computational power. Moreover, depth cameras based on infrared light are manufactured with smaller size, lower weight and high frame rate, competing with monocular cameras in wearable applications in indoor environments. Finally, we have to mention a third sensor, the stereo camera, that is specially used in dense mapping systems in outdoors, although it behaves as a monocular camera for distant objects.

We can also mention advantages intrinsic to dense techniques. 1. Image alignment is less prone to errors in camera pose than matching sparse features. 2. The performance of image alignment for estimating camera pose is better than sparse techniques when the images have blur. 3. The depth map creation does not require to estimate the covariance matrix, unlike filter-based techniques, and can be implemented on graphics processing units.

All these properties of dense techniques, of monocular and depth cameras and the wide spectrum of applications, make them appealing. Our goal is to contribute in the solution of an identified problem in the field of dense localization and mapping using a monocular camera: the low accuracy of the built depth maps in regions with low texture, and with very reflective or translucent objects, in indoor environments where the light conditions are uncontrolled. Some of these difficulties are common to depth maps obtained from a depth sensor based on structured light so we extend the solution to this kind of data.

Our proposed solution, instead of working with alternative norms applied to the standard data term and regularisation term, adds depth data from an optimal model (object shape prior) that appears in the scene and constraints the solution got with variational techniques to be similar to both the raw depth map from the sensor and the depth data associated to the optimal model and at the same time, constraints the solution to be smooth, getting finally an inpainted, denoised and smoothed solution. Fusing enhanced depth maps in the reconstruction process is important since the surface of the reconstructed

model will be smoother and more accurate, especially in the region associated to the object of interest.

1.5 OBJECTIVES

The main objective of this doctoral thesis is to make the monocular system for dense localization and mapping more robust to difficult conditions such as low-textured, very reflective and translucent objects, and changes in light conditions, that produce depth maps with low accuracy in affected regions and therefore low-quality dense reconstructions when these depth maps are fused.

The specific objectives are:

- Make a review in the state of the art of monocular algorithms for dense localization and mapping.
- Develop an algorithm for estimating the pose of a monocular camera using dense techniques, robust to low texture and changes in light conditions.
- Develop an algorithm for building depth maps using a monocular camera, robust to low texture and changes in light conditions.
- Implement the algorithms for dense localization and mapping in graphics processing unit for improving their performance with respect to time.

1.6 THESIS OUTLINE

This document was structured in chapters associated to the proposed objectives. Each chapter has an introduction, related work, the methodology carried out, experiments, results, and conclusions. Figure 1.1 shows the constituent parts of the system (for a monocular camera and for a depth sensor) and the connections that define the interactions and dependences between modules. In chapter 2 two dense localization approaches are described: frame-to-frame and frame-to-model-vertices approach. In chapter 3 the problem that emerges under certain circumstances in the process of depth map building, using the decoupling approach with a variational framework, is described and quantified through the experiments. In chapter 4 the processes for model alignment, truncated signed distance function TSDF estimation, model compression, dimensionality reduction and finding the optimal model (shape prior) w.r.t. pose, scale and shape is described in detail, quantifying its accuracy with synthetic and real data. In chapter 5 the shape prior is integrated into the raw depth maps using a novel framework based on variational techniques and the primal-dual algorithm, that besides merging depth data from two sources, makes denoising and inpainting. Chapter 6 describes the fusion of enhanced depth maps and quantify the improvements achieved by merging depth data

of the shape prior. The final chapter summarizes the general conclusions, the main contributions and describes the future work.

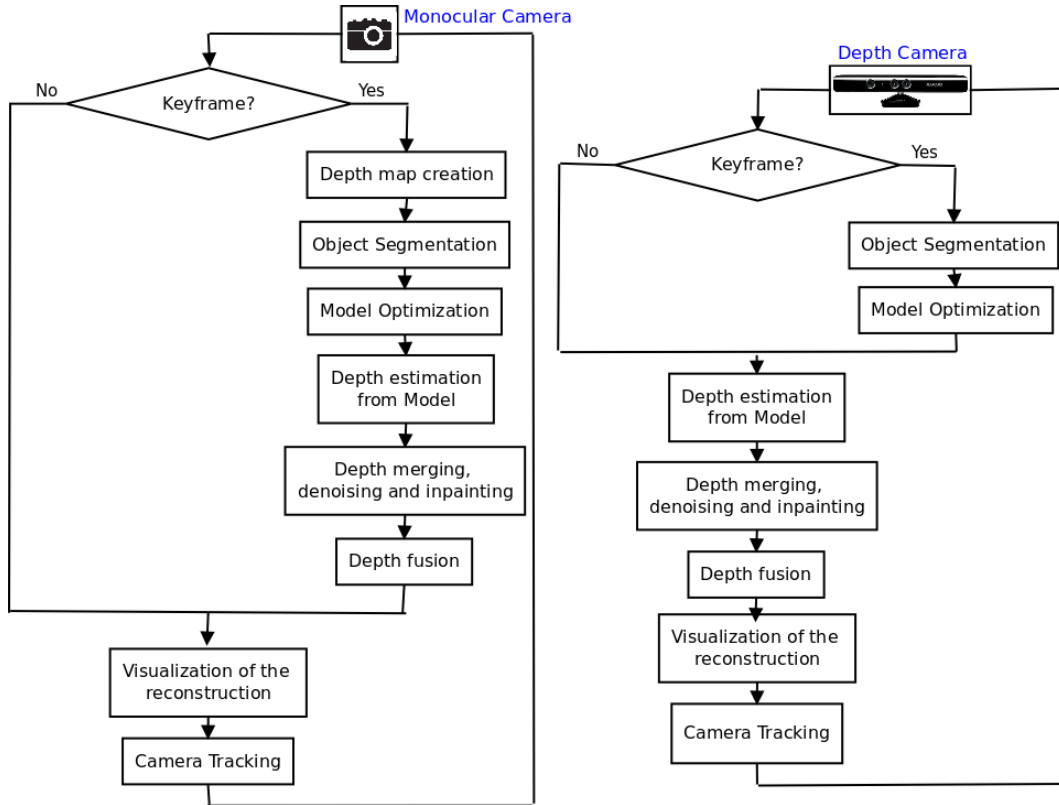


Figure 1.1: Proposed system for dense localization and mapping using shape priors. On the left, the system works with a monocular camera. On the right, the system works with a depth camera.

CHAPTER 2

DENSE LOCALIZATION TECHNIQUES

2.1 INTRODUCTION

There are three main frameworks in the literature about tracking the position of a camera using dense information: frame-to-frame [40], frame-to-model-vertices [61], and frame-to-model-TSDF [4]. In frame-to-frame approaches, we can compute the relative transformation between the current image $\mathbf{I}_c$ and a reference image $\mathbf{I}_r$, given its corresponding depth map $\mathbf{D}_r$, as follows. We use an initial estimation of the relative position between the current and the reference image T_{cr} to create a warp image $\mathbf{I}_{warp}$ that should resemble the reference image $\mathbf{I}_r$. We can then calculate the differences and correct the transformation in order to reduce them. In a frame-to-model-vertices approach, the idea is to estimate the pose of the current camera with respect to a continuously built 3D dense model. Given the current depth map at time k , a set of vertices and normals can be calculated from it $(\mathbf{V}_k, \mathbf{N}_k)$, respectively. From the model and the current estimation of the camera pose w.r.t. the model we can make a prediction of these elements $(\hat{\mathbf{V}}_k, \hat{\mathbf{N}}_k)$. Again, comparing the observations with the predictions we can correct the current pose estimation. Finally, in frame-to-model-TSDF approaches the camera is also localized with respect to the model but instead of predicting vertices and normals what is compared are the values of the truncated signed distance function (TSDF) used to represent the model.

The advantage of the model-based approach is that drift in the camera pose is low if the model is continuously observed. In the case of the frame-to-frame approach, the advantage is that a model is not required to compute the trajectory of the camera. In that sense, model-based approaches are more similar to Bundle Adjustment techniques whereas frame-to-frame ones to Visual Odometry algorithms.

In this chapter, we focus on two of the three frameworks described above for localizing a camera that moves freely with 6DOF: the frame-to-frame approach and the frame-to-model-vertices approach. Our contributions in the tracking

algorithm with the frame-to-frame approach are: 1. the use of a set of depth maps available only for selected frames or "keyframes". In this way, we can overcome the computational payload that commonly whips other visual localization systems; 2. A two-steps process to localize the camera with respect to the closest keyframe: 3D to 2D visual odometry followed by the image alignment. It results in high accuracy estimations even when using blurred input images. The description of this approach can also be found in our chapter of book [22] and in proceedings of LARS 2015 [18]. Our contributions in the algorithm with the frame-to-model-vertices approach are: 1. A color management algorithm, that allows the system to store color information around the whole trajectory and get a model with full color data (this color management algorithm will be described in more detail in chapter 6). 2. An algorithm for propagating to the model information of detected objects and for managing multiple instances of an object class, computing its location, a 3D bounding box, and its probability. This algorithm for scene labeling is explained in our paper [20].

In section 2.2 we analyse the most important sparse, semi-dense and dense localization systems. In sections 2.3 and 2.4 we focus on the proposed techniques for dense localization of a camera based on frame-to-frame and frame-to-model-vertices approaches, respectively. Our experiments and the performance of the algorithms implemented on a graphics processing unit are explained in section 2.5. Finally, we draw our conclusions in section 2.6.

2.2 RELATED WORK

The system presented in [44] radically changed the filtering paradigm used until that moment to non-linear optimisation techniques using bundle adjustment over keyframes. This system is pioneer in balancing the computations between two CPU cores, one for localization and the other for mapping, respectively. As we mentioned before, keyframes are selected to reduce the computational complexity of the optimization. It uses a Five-Point algorithm [88] to initialise the system. Camera tracking is still carried out on the intermediate camera frames by re-projecting the estimated 3D points and minimizing a photo-consistency error. Despite this approach is able to create very accurate maps in real time, it represents the environment sparsely. The map can only contain up to 6000 points and 120 keyframes. Nonetheless, this work represents a transition between sparse and dense systems.

Stuhmer et al. [90] proposed a mapping approach able to compute depth maps from monocular images. The localization is based on bundle adjusting a set of keyframes [44] on a CPU. The system can reach 24 fps for a depth map resolution of 480×360 pixels. Similarly, the work in [62] builds high-quality dense reconstructions of small environments using a GPU and a monocular camera in real time. The camera is localized by aligning the current observed frame to

a synthetic one coming from the model. The camera motion corresponds then to the relative transformation that produces the best alignment between the actual and the synthetic images. A pyramid of images is built to accelerate convergence allowing faster motions of the camera. The initialization of the system is similar to [44].

In [87] an energy-based approach is presented to estimate the rigid body motion of a handheld RGB-D camera for a static scene. The reprojection error is defined such that the warped second image matches the first image, using consecutive images of the trajectory. The authors minimize this non-convex reprojection error by a sequence of convex optimization problems obtained by linearizing the data term and solving the resulting normal equations in a coarse-to-fine manner. This system is able to perform at 12.5 Hz on CPU. The large scale direct monocular SLAM system LSD-SLAM [25] uses direct image alignment and filtering-based estimation of semi-dense depth maps. Moreover, the effect of noisy depth values is included probabilistically into tracking. The global map is represented as a pose graph consisting of keyframes as vertices with 3D similarity transforms as edges. The system detects loop closures and scale-drift, and runs on real-time on a CPU.

KinectFusion [61] is the first system that builds a dense reconstruction of room-sized scenes and simultaneously localize the camera at 30Hz, using the Kinect 1.0, and one GPU. The model-to-frame tracking is based on coarse-to-fine ICP. It minimizes the point-to-plane distance of points matched using projective association rather than feature extraction and matching. The three-level pyramid is built with vertex and normal maps from depth maps obtained from the sensor and from the reconstructed model. Once the camera pose is estimated, the depth maps captured with the sensor are fused and the model is updated. Kintinuous [95] is an extension of KinectFusion [61] for working in both room-sized environments and large-scale paths. The cost function integrates the cost of the RGB-D frame-to-frame tracking of [87] (implemented on GPU), and the cost of the original model-to-frame ICP tracking of [61], in order to tackle poor performance when the camera points to a flat wall or corridors with no significant 3D features. They use a visual odometry system called FOVIS [38], that relies on FAST feature correspondences in the RGB images, and that automatically switches when a planar environment appears. Unlike [61], Kintinuous integrates color information, getting full colored models. The color volume is updated based on KinFu [76], but with real time surface coloring and rejection of unreliable color measurements. The next version of Kintinuous [97] includes loop closure with pose-graph optimization and non-rigid space deformation.

ElasticFusion [98] is a map-centric approach that builds a surfel-based map of room-sized environments and tracks the camera with a dense frame-to-model method that combines point-to-plane error and dense photometric error. It

performs dense model-to-model local surface loop closures applying a non-rigid space deformation of the map, instead of a standard pose-graph optimization. Global loop closure is achieved with an appearance-based place recognition method. ORB-SLAM2 [59] builds globally consistent sparse reconstructions for long-term trajectories with either monocular, stereo or RGB-D inputs, performing in real time on standard CPUs and including loop closure, map reuse and relocalization. This system has three main parallel threads: the tracking with motion only bundle adjustment (BA), the local mapping with local BA, and the loop closing with pose graph optimization and full BA. It uses ORB features for tracking, mapping and place recognition tasks. With BA and keyframes, it achieves more accuracy in localization than state-of-the-art methods based on ICP or photometric and depth error minimization.

2.3 DENSE LOCALIZATION WITH FRAME-TO-FRAME APPROACH

In this section we are going to explain the main processes developed for estimating the camera pose using the frame-to-frame approach. Without loss of generality, consider a sequence of images $\mathbf{I}_{c_i}$, for $i = 1 : m$, gathered by the camera. Our aim is to estimate the relative transformation $T_{r_{near}c_i}$, at each frame $\mathbf{I}_{c_i}$, where r_{near} is the sub-index of the closest keyframe.

Next, we will describe the criteria for keyframe selection, the calculation of an initial pose estimation, how to refine this estimation using a photometric error obtained by aligning the warped and observed images, and finally, its minimization using Levenberg-Marquardt.

2.3.1 KEYFRAME SELECTION

The mapping and global optimization is performed only for certain frames of reference in order to achieve higher performance. These frames are known as keyframes and are selected according to certain temporal and spatial criteria [44]. For the sake of simplicity, fig. 2.1 shows an example of keyframes selected based just on a time criteria. Each keyframe r_j has a transformation matrix T_{wr} , an RGB image $\mathbf{I}_r$ and a depth map $\mathbf{D}_r$. The system saves all keyframes with its corresponding information, which allows it to initialize new keyframes, estimate the relative pose of the current image $\mathbf{I}_c$ with respect to the nearest keyframe and perform optimization for getting a consistent map and trajectory.

We set thresholds to the displacement and to the change in orientation (no time criteria): our system initializes a new keyframe when the displacement with respect to any other keyframe is greater than 0.13 m or when the sum

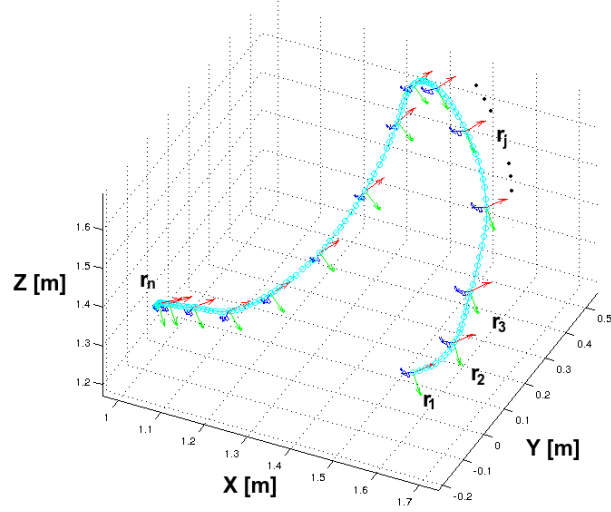


Figure 2.1: The camera trajectory is drawn in cyan. Keyframes r_j are drawn with their local coordinate systems. They are selected based on a time criteria, e.g. for every nine images a new keyframe is initialized.

of changes in orientation with respect to the last keyframe (around X, Y and Z axis) is greater than 30° . Thus, if the displacement and change in orientation of the camera is smaller than these thresholds, no new keyframes will be initialized. In this way, the map is compact in the sense that it does not grow arbitrarily with time due to little or no motion, even with repeated motion in restricted areas as we will see in the results section.

The camera trajectory is obtained concatenating relative transformations, e.g. $T_{wr1}T_{r1r2}T_{r2r3}\dots T_{r_{j-1}r_j}T_{r_jc_i}$, where the world coordinate system corresponds to the coordinate system of the first keyframe, therefore $T_{wr1} = [I \mid \mathbf{0}]$. For the following images $\mathbf{I}_{c_i}$, the position of the previous image is used to define the nearest keyframe. The estimation of relative motion is performed considering the correspondence between the RGB image associated to the nearest keyframe and the current image. The pose of the new image is defined as $T_{wr_{near}}T_{r_{near}c_i}$.

2.3.2 INITIAL POSE ESTIMATION

We compute an initial estimation of the camera motion with a sparse technique. We are interested in 3D to 2D methods where the transformation T_{cr} is computed from correspondences between 3D points $\mathbf{X}_r$ and 2D points $\mathbf{p}_c$. $\mathbf{X}_r$ can be obtained from the depth map $\mathbf{D}_r$ associated to the reference image $\mathbf{I}_r$, it means, the image taken from the keyframe. We use the depth map loaded from the dataset of A. Handa [36]. In Fig. 2.2 we can see the RGB image $\mathbf{I}_r$ and its corresponding depth map $\mathbf{D}_r$.

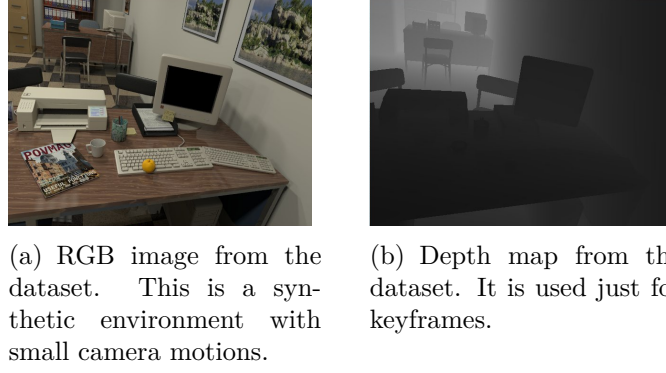


Figure 2.2: (a) Perfect ray-traced RGB Image and (b) depth map from the dataset of A. Handa [36]. They are used for computing the initial pose estimation and for the posterior refinement.

We apply a SURF feature detector and a SURF descriptor extractor to $\mathbf{I}_r$ and to $\mathbf{I}_c$. Then a matcher based on the FLANN algorithm computes the matches between SURF descriptors of both images. The matches are validated if they fulfil three conditions. The first one is that the distance between the two matched descriptors is smaller than a threshold. The second one is that the correlation index between a patch around the descriptor in $\mathbf{I}_r$ and the descriptor in $\mathbf{I}_c$ is greater than a threshold. The final one is that the matches are inliers when the fundamental matrix is computed using RANSAC. This means that the descriptor in $\mathbf{I}_c$ must lie on the corresponding epipolar line. We set the threshold in distance to five times the minimum distance between all matches and the correlation index threshold to 0.7. Finally, as the goal is to find T_{cr} that minimizes the image reprojection error, we apply a solver based on PNP algorithm that uses RANSAC for outlier rejection

$$\arg \min_{T_{cr}} \sum_h \|\mathbf{p}_{c_i}^h - \hat{\mathbf{p}}_{r_j}^h\|^2, \quad (2.1)$$

where $\hat{\mathbf{p}}_{r_j}^h$ is the reprojection of the h^{th} 3D point $\mathbf{X}_{r_j}^h$ into image $\mathbf{I}_{c_i}$ according to the transformation T_{cr} . The transformation of a feature $\mathbf{p}_{r_j}^h$ with 2D coordinates $\boldsymbol{\mu} = (u, v)$ to a spacial coordinate $\mathbf{X}_r^h$ and the transformation of a spacial point $\mathbf{X}_c^h = T_{cr} \mathbf{X}_r^h$ to a point in the image plane $\mathbf{c}_i$, $\hat{\mathbf{p}}_{r_j}^h$, are described in subsection 2.3.3.

2.3.3 POSE REFINEMENT

The goal of the localization algorithm is to estimate the relative transformation matrix between two camera poses T_{cr} (fig. 2.3) using all the depth data $\mathbf{D}_r$ of the associated keyframe and all the intensity data of $\mathbf{I}_r$ and $\mathbf{I}_c$.

The current image is warped with the transformation T_{cr} in order to compare it with the reference image. In case the transformation T_{cr} is the true 3D

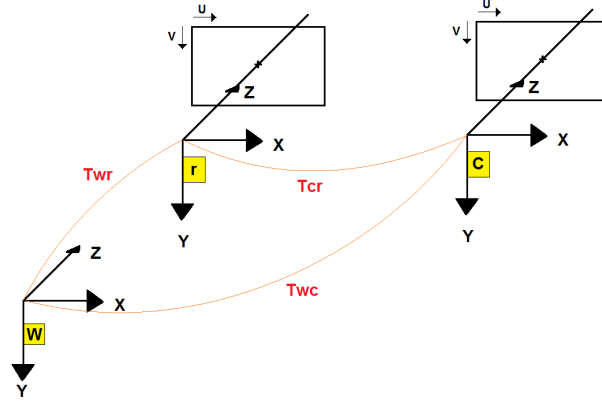


Figure 2.3: Current image I_c and reference image I_r with their corresponding local and global coordinate systems. The transformations that relate coordinate systems are drawn in red.

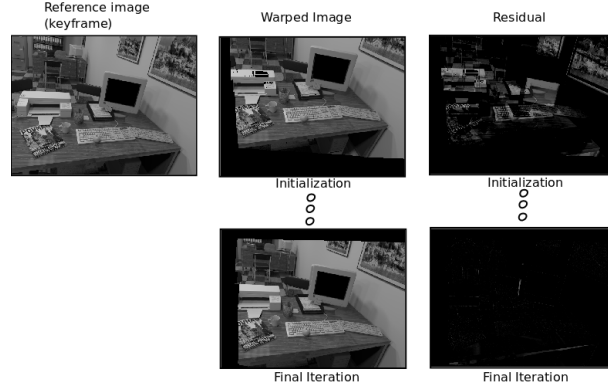


Figure 2.4: Alignment between reference and warped images. The residual depends on the estimation in motion.

transformation between the two cameras, the warped images I_w and I_r should be identical and the residual should be zero [2]. Figure 2.4 shows how the residual decreases for a more refined estimation in motion.

The warping consists in projecting the reference depth map D_r into 3D space, transforming by T_{cr} and projecting back into the current image I_c . Algorithm 1 summarizes the steps of this process. K is the intrinsic calibration matrix, π represents the function that computes normalized homogeneous coordinates and $\dot{\mu}$ is the homogeneous version of the coordinate in the image plane. A residual function is defined as the difference between $I_r(\mu)$ and $I_w(\mu)$,

$$r(T_{cr}, \mu) = I_r(\mu) - I_c(K\pi(T_{cr}D_r(\mu)K^{-1}\dot{\mu})) \quad (2.2)$$

Since the depth $D_r(\mu)$ is known, T_{cr} is defined as the argument that minimizes the residual function. T_{cr} is a homogeneous transformation matrix defined by a rotation matrix R_{cr} and a translation vector t ,

Data: Current Intensity Image $\mathbf{I}_c$, Reference Depth Image $\mathbf{D}_r$, Transformation T_{cr} and Intrinsic Matrix K .

Result: Warped Image $\mathbf{I}_w$.

- 1 Transform coordinates in the image plane $\mathbf{I}_r$ to spacial coordinates:
 $\mathbf{X}_r(\boldsymbol{\mu}) = \mathbf{D}_r(\boldsymbol{\mu})K^{-1}\dot{\boldsymbol{\mu}}$
- 2 Change of coordinate system: $\mathbf{X}_c(\boldsymbol{\mu}) = T_{cr}\mathbf{X}_r(\boldsymbol{\mu})$
- 3 Transform spatial coordinates to coordinates in the image plane: $\dot{\boldsymbol{\mu}} = K\pi(\mathbf{X}_c)$,
 where $\pi(\mathbf{X}_c) = (X_n, Y_n, 1)^T = \left(\frac{X_c}{Z_c}, \frac{Y_c}{Z_c}, 1\right)^T$
- 4 Compute the warped Image: $\mathbf{I}_w(\boldsymbol{\mu}) = \mathbf{I}_c(K\pi(T_{cr}\mathbf{D}_r(\boldsymbol{\mu})K^{-1}\dot{\boldsymbol{\mu}})) = \mathbf{I}_c(\dot{\boldsymbol{\mu}})$

Algorithm 1: Warping an Image

$$T_{cr} = \begin{bmatrix} R_{cr} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} . \quad (2.3)$$

The vector $\boldsymbol{\xi} = (\mathbf{w}, \mathbf{v})$ represents the relative transformation matrix T_{cr} with the smallest possible number of parameters. The orientation of the vector $\mathbf{w} = (w_1, w_2, w_3)$ defines the axis of rotation and the magnitude of the vector defines the amount of rotation. The vector $\mathbf{v} = (v_1, v_2, v_3)$ corresponds to the translation $\mathbf{t}$ rotated over $\mathbf{w}$.

An exponential mapping between the 6-element vector $\boldsymbol{\xi}$ in the Lie algebra $\mathfrak{se}(3)$ [94] to members of the group $SE(3)$ is made, like in [44] with a motion model and feature association (sparse techniques) or in [62] using a dense estimation of rotation [53] and image alignment. Computing the exponential mapping involves the operator $[\cdot]_x$, which maps a vector to its *skew* matrix,

$$\exp_{SE3}([\boldsymbol{\xi}]_x) = T_{cr} , \quad (2.4)$$

where $[\boldsymbol{\xi}]_x$ is

$$[\boldsymbol{\xi}]_x = \begin{bmatrix} 0 & -w_z & w_y & v_x \\ w_z & 0 & -w_x & v_y \\ -w_y & w_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix} . \quad (2.5)$$

The updated transformation T_{cr} is expressed as the product of an initial transformation T_0 and an update in the motion vector $d\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}_0$, that is,

$$T_{cr} = d\mathbf{T} * \mathbf{T}_0 = \exp_{SE3}([d\boldsymbol{\xi}]_x) * \mathbf{T}_0 . \quad (2.6)$$

In this way, the singularities are avoided since the incremental update is calculated in the tangent space around the identity (i.e. it is always very small) and mapped back onto the manifold $SE(3)$. With this transformation, the residual defined in eq. (2.2) is

$$\mathbf{r}(\boldsymbol{\xi}, \boldsymbol{\mu}) = \mathbf{I}_r(\boldsymbol{\mu}) - \mathbf{I}_c(K\pi(\exp_{SE3}([d\boldsymbol{\xi}]_x)\mathbf{T}_0\mathbf{D}_r(\boldsymbol{\mu})K^{-1}\dot{\boldsymbol{\mu}})) . \quad (2.7)$$

Finally, the localization problem consist in finding $\boldsymbol{\xi}$ that minimizes the following cost function:

$$\min_{\boldsymbol{\xi}} \frac{1}{2} \int [\mathbf{r}(\boldsymbol{\xi}, \boldsymbol{\mu})]^2 d\boldsymbol{\mu} . \quad (2.8)$$

This is a non-linear minimization problem that we solve with Levenberg Marquardt [37]. Since $\mathbf{r}$ is non-linear and non-convex in $\boldsymbol{\xi}$ we linearize it using a first order Taylor expansion:

$$\mathbf{r}(\boldsymbol{\xi}) \approx \mathbf{r}(\boldsymbol{\xi}_0) + \left. \frac{d\mathbf{r}(\boldsymbol{\xi})}{d\boldsymbol{\xi}} \right|_{\boldsymbol{\xi}_0} (\boldsymbol{\xi} - \boldsymbol{\xi}_0) . \quad (2.9)$$

The quantity $\frac{d\mathbf{r}(\boldsymbol{\xi})}{d\boldsymbol{\xi}}$ is the Jacobian of $\mathbf{r}$ and is given by

$$J = \begin{pmatrix} \nabla r_1(\boldsymbol{\xi})^T \\ \nabla r_2(\boldsymbol{\xi})^T \\ \vdots \\ \nabla r_i(\boldsymbol{\xi})^T \end{pmatrix} , \quad (2.10)$$

where $\nabla r_i(\boldsymbol{\xi})$ is the derivative of the i^{th} component of the residual vector.

COMPUTATION OF THE JACOBIAN

Lets define a function W for the warping process

$$W(\boldsymbol{\xi}, \boldsymbol{\mu}) = \pi(\overbrace{K\exp_{SE(3)}([d\boldsymbol{\xi}]_x)\mathbf{T}_0\mathbf{D}_r(\boldsymbol{\mu})K^{-1}\dot{\boldsymbol{\mu}}}^{\mathbf{p}_{1n}}) , \quad (2.11)$$

$\mathbf{p}_1$

where $\pi(\mathbf{X}) = \left(\frac{x}{z}, \frac{y}{z}, 1\right)^T$, $\mathbf{p}_1$ denotes the transformed point and $\mathbf{p}_{1n}$ is the transformed back-projected (normalized) point. The residual can be written as

$$\mathbf{r}(\boldsymbol{\xi}, \boldsymbol{\mu}) = \mathbf{I}_c(W(\boldsymbol{\mu}, \boldsymbol{\xi})) - \mathbf{I}_r(\boldsymbol{\mu}) . \quad (2.12)$$

The derivative of the residual can be written using the chain rule,

$$\frac{d\mathbf{r}(\boldsymbol{\xi})}{d\boldsymbol{\xi}} = \left. \frac{d\mathbf{I}_c(W(\boldsymbol{\xi}, \boldsymbol{\mu}))}{dW(\boldsymbol{\xi}, \boldsymbol{\mu})} \right|_{W(\boldsymbol{\xi}_0, \boldsymbol{\mu})} \frac{dW(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\pi} \Big|_{\mathbf{p}_{1n}} \frac{d\pi(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\boldsymbol{\xi}} \Big|_{\boldsymbol{\xi}_0} . \quad (2.13)$$

The first term $\left. \frac{d\mathbf{I}_c(W(\boldsymbol{\xi}, \boldsymbol{\mu}))}{dW(\boldsymbol{\xi}, \boldsymbol{\mu})} \right|_{W(\boldsymbol{\xi}_0, \boldsymbol{\mu})}$ is just the image gradient of the warped image $\nabla \mathbf{I}_w(\boldsymbol{\mu})$. The second term involves the projection $\pi(\cdot)$ and therefore

$$\left. \frac{dW(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\pi} \right|_{\mathbf{p}_{1n}} = \begin{bmatrix} \frac{1}{z} & 0 & \frac{-x}{z^2} \\ 0 & \frac{1}{z} & \frac{-y}{z^2} \end{bmatrix}. \quad (2.14)$$

The third term $\left. \frac{d\pi(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\boldsymbol{\xi}} \right|_{\boldsymbol{\xi}_0} = \left[\frac{d\pi}{dw_x} \frac{d\pi}{dw_y} \frac{d\pi}{dw_z} \frac{d\pi}{dv_x} \frac{d\pi}{dv_y} \frac{d\pi}{dv_z} \right]$. Putting together all these derivatives, it becomes

$$\left. \frac{d\pi(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\boldsymbol{\xi}} \right|_{\boldsymbol{\xi}_0} = K \begin{bmatrix} 0 & p_z & -p_y & 1 & 0 & 0 \\ -p_z & 0 & p_x & 0 & 1 & 0 \\ p_y & -p_x & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.15)$$

The first 3×3 submatrix is the negative skew matrix of the vector $\mathbf{p}_1$ and the second 3×3 submatrix is the identity matrix. Equation (2.15) simplifies to

$$\left. \frac{d\pi(\boldsymbol{\xi}, \boldsymbol{\mu})}{d\boldsymbol{\xi}} \right|_{\boldsymbol{\xi}_0} = K [-[\mathbf{p}_1]_x I_3]. \quad (2.16)$$

Now that the Jacobian was defined, the variation in the motion vector $d\boldsymbol{\xi}$ can be obtained by solving eq. (2.17).

$$J^T J d\boldsymbol{\xi} = -J^T \mathbf{r}. \quad (2.17)$$

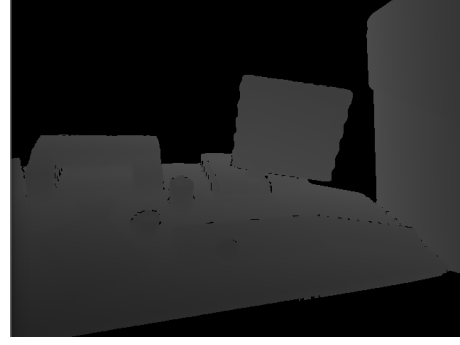
For the Levenberg-Marquardt method the Hessian can be approximated by $J^T J + \alpha I$. For a large α the method approaches the Steepest Descent method, while for a small α the method approaches the Gauss-Newton method. Generally, the Steepest Descent works better further away from the local minima and the Gauss-Newton approximation works better close the local minima where the quadratic approximation is good [2]. We start with a small initial value of α , $\alpha = 1e - 3$. After each iteration, the new cost is compared with the previous cost. If the cost has decreased, α is reduced by a factor of 10 and the parameters are updated. If the cost has increased, the parameters are not updated and α is increased by a factor of 10. Solving eq. (2.17) is efficient because we just need to invert a 6×6 matrix and use $d\boldsymbol{\xi}$ in eq. (2.6).

2.4 DENSE LOCALIZATION WITH FRAME-TO-MODEL APPROACH

In this approach, the Iterative Closest Point ICP algorithm, based on the point-to-plane distance, is used for estimating the relative transformation between $\mathbf{I}_c$ and $\mathbf{I}_r$. For using ICP we need to compute at time k a predicted depth map from the 3D model updated in time $k - 1$. Next, the computation of the predicted depth map and the ICP algorithm are explained.



(a) Dense reconstruction with color of the scene, rasterized for getting a 2D representation.



(b) Depth map obtained from Z-buffer of OpenGL, which is used for frame-to-model ICP in the tracking process.

Figure 2.5: (a) 3D model and its visualization on a screen. (b) depth map obtained from Z-buffer of OpenGL. Dark regions correspond to space outside the grid of voxels.

2.4.1 RASTERIZATION

Our system does not perform ray casting for getting a depth map from the model like [61] but marching cubes and rasterization [68], taking advantage of a depth buffer available in OpenGL and the estimations in camera pose. Rasterization is a technique used for displaying 3D models, represented as a set of polygons, on a screen (see fig. 2.5(a)). The vertices of the polygons are projected on a 2D plane (viewer's monitor), knowing the relative transformation between the model and the camera $T_{cam,model}$. OpenGL overrides the depth of the pixel with the depth of the closest object (see fig. 2.5(b)). No information exists outside the volumetric structure defined by the user for reconstructing it.

The frustrum is defined by Z_{near} , Z_{far} and vertical field of view VFOV. The range of depths $\hat{Z}$ stored on the z-buffer is $[0, 1]$. Objects outside the frustrum will not be rendered. After a perspective projection,

$$\hat{Z} = \frac{Z_{far} + Z_{near}}{2(Z_{far} - Z_{near})} + \frac{1}{Z} \left(\frac{-Z_{far}Z_{near}}{Z_{far} - Z_{near}} \right) + \frac{1}{Z} \quad (2.18)$$

where Z is the real depth. Solving for the real depth, we get:

$$Z = \frac{2Z_{far}Z_{near}}{(Z_{far} + Z_{near}) - 2(\hat{Z} - 1/2)(Z_{far} - Z_{near})} \quad (2.19)$$

We defined Z_{near} as the focal length (in metric units), $Z_{far} = 100\text{m}$ and the vertical field of view as

$$\text{VFOV} = 2\text{atan}\left(\frac{0.5\text{Height}}{f_y}\right) \quad (2.20)$$

2.4.2 DENSE ICP

Dense ICP is appropriate due to two reasons. First, the camera motion at frame rate is small, so we can use fast projective data association to obtain correspondences and point-to-plane metric for pose optimisation, like in [61]. This fact makes dense ICP compatible with depth cameras since these devices deliver depth maps at high frame rates. Second, these algorithms are highly parallelizable so an important speed up is obtained when implemented on GPU.

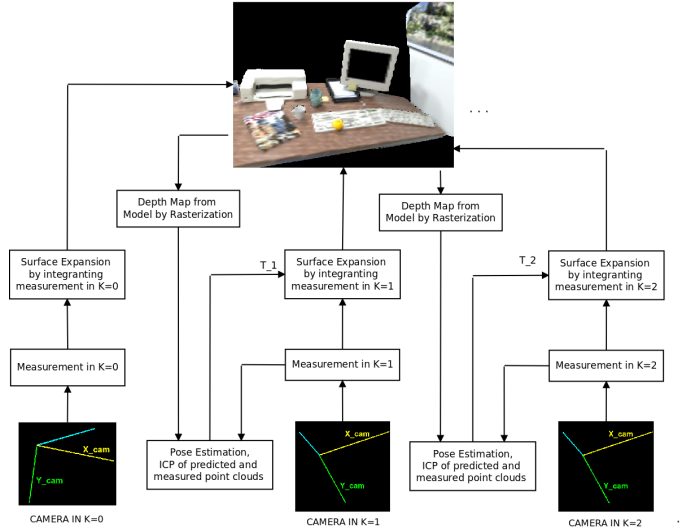


Figure 2.6: Diagram of the frame-model based tracking. The vertex map in $k - 1$, obtained from Z-buffer of OpenGL, is aligned with the vertex map, computed from the measured depth map.

The general diagram of the frame-to-model tracking is shown in fig. 2.6. For each depth map, whether estimated from the model, $\hat{D}_{k-1}(\mu)$, or acquired from the sensor, $D_k(\mu)$, we compute a vertex map and a normal map, $\hat{V}_{k-1}(\mu), \hat{N}_{k-1}(\mu)$ and $V_k(\mu), N_k(\mu)$, respectively. The vertex map is computed as,

$$V_k(\mu) = D_k(\mu)K^{-1}\dot{\mu}, \quad (2.21)$$

where the dot over a vector denotes homogeneous vector. In this case $\dot{\mu}$ is the homogeneous version of the coordinate in the image plane. The normal map is computed as the cross product between two vectors obtained from the neighbours of the current vertex in μ ,

$$N_k(\mu) = v[(V_k(u+1, v) - V_k(u, v)) \times (V_k(u, v+1) - V_k(u, v))] \quad (2.22)$$

where $v[\mathbf{x}] = \frac{\mathbf{x}}{\|\mathbf{x}\|^2}$, it means, a unit normal vector. Next, we transform the vertex from a local coordinate system to a global one

$$\mathbf{V}_k^w(\boldsymbol{\mu}) = T_{wc_k} \dot{\mathbf{V}}_k(\boldsymbol{\mu}) \quad (2.23)$$

ENERGY FUNCTION

$$E(T_{wc_k}) = \sum_{\boldsymbol{\mu} \in \Omega} \|(T_{wc_k} \dot{\mathbf{V}}_k(\boldsymbol{\mu}) - \hat{\mathbf{V}}_{k-1}^w(\hat{\boldsymbol{\mu}}))^T \hat{\mathbf{N}}_{k-1}^w(\hat{\boldsymbol{\mu}})\|_2 \quad (2.24)$$

Equation (2.24) is a global point-to-plane energy, that uses global coordinates for vertices and normals, and a L2 norm of the error, having as argument to minimize the transformation T_{wc_k} . This energy aligns, using Levenberg-Marquardt, a live surface $(\mathbf{V}_k, \mathbf{N}_k)$ with a predicted surface from the model in the previous step $(\hat{\mathbf{V}}_{k-1}, \hat{\mathbf{N}}_{k-1})$ by penalizing long distances between the vertex $\mathbf{V}_k$ and the tangent plane to vertex $\hat{\mathbf{V}}_{k-1}$ (see fig. 2.7). The correspondences are computed by projecting the vertices obtained from the depth sensor in time k to the image plane of camera in time $k-1$, using an estimation for the frame-to-frame transformation $T_{k-1,k}^{\text{iter}} = T_{wc_{k-1}}^{-1} T_{wc_k}^{\text{iter}}$,

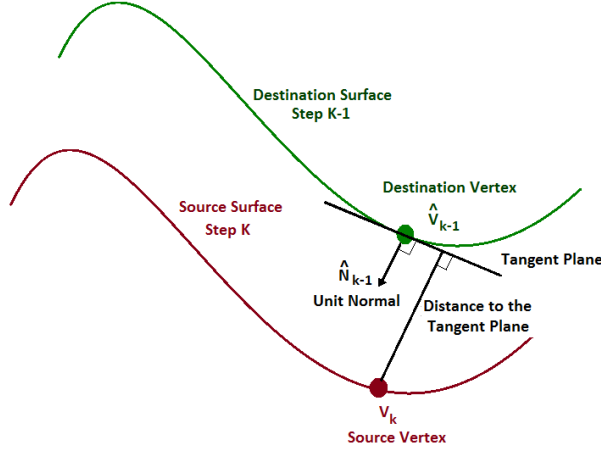


Figure 2.7: Graphical interpretation of the point-plane optimization. The vertex $\mathbf{V}_k$ (in red), obtained from the measurement, is moved in order to reduce the distance to the tangent plane of vertex $\hat{\mathbf{V}}_{k-1}$ obtained from the model.

$$\hat{\mathbf{u}} = K\pi(T_{k-1,k} \dot{\mathbf{V}}_k). \quad (2.25)$$

TRANSFORMATION UPDATE

We initialize the iterative optimization with $T_{wc_k}^0 = T_{wc_{k-1}}$. The global camera transformation is updated, assuming a small change in position and orientation, by pre-multiplying an incremental transformation to the current estimation,

$$T_{wc_k}^{\text{iter}} = T_{\text{inc}}^{\text{iter}} T_{wc_k}^{\text{iter}-1} \quad (2.26)$$

where

$$T_{\text{inc}}^{\text{iter}} = [R^{\text{iter}} | \mathbf{t}^{\text{iter}}] = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix} \quad (2.27)$$

This rigid body transformation can be written as a 6-elements vector,

$$\mathbf{x} = [\alpha, \beta, \gamma, t_x, t_y, t_z] \quad (2.28)$$

With this new transformation, a new vertex map, that reduces the alignment error, is computed,

$$\mathbf{V}_k^w = T_{wc_k}^{\text{iter}} \dot{\mathbf{V}}_k(\boldsymbol{\mu}) = R^{\text{iter}} \mathbf{V}_k^w + \mathbf{t}^{\text{iter}}, \quad (2.29)$$

Expressing eq. (2.29) using the motion vector $\mathbf{x}$ (eq. (2.28)) we get:

$$\mathbf{V}_k^w = G(\boldsymbol{\mu})\mathbf{x} + \mathbf{V}_k^w \quad (2.30)$$

where $G(\boldsymbol{\mu})$ consist of the skew-symmetric representation of $\mathbf{V}_k^w$ and a 3×3 identity matrix,

$$G(\boldsymbol{\mu}) = [[\mathbf{V}_k^w]_{\times} | I] \quad (2.31)$$

The energy of a linearised version of eq. (2.24) can be expressed in terms of the motion vector $\mathbf{x}$, and the minimisation problem results

$$\min_{T_{wc_k}} E(T_{wc_k}) = \sum_{\boldsymbol{\mu} \in \Omega} \|(G(\boldsymbol{\mu})\mathbf{x} + \mathbf{V}_k^w - \hat{\mathbf{V}}_{k-1}^w(\hat{\boldsymbol{\mu}}))^T \hat{\mathbf{N}}_{k-1}^w(\hat{\boldsymbol{\mu}})\|_2^2. \quad (2.32)$$

Derivating the objective function with respect to the motion vector $\mathbf{x}$ and setting to 0, a symmetric linear system of 6×6 , for each correspondence, is obtained (see [61] for more details):

$$\sum_{\Omega(\boldsymbol{\mu}) \neq \text{null}} (\mathbf{A}^T \mathbf{A})\mathbf{x} = \sum \mathbf{A}^T \mathbf{b} \quad (2.33)$$

$$\mathbf{A}^T = G(\boldsymbol{\mu})^T \hat{\mathbf{N}}_{k-1}^w(\hat{\boldsymbol{\mu}}) \quad (2.34)$$

$$\mathbf{b} = \hat{\mathbf{N}}_{k-1}^w(\hat{\mathbf{V}}_{k-1}^w(\hat{\boldsymbol{\mu}}) - \mathbf{V}_k^w(\boldsymbol{\mu})) \quad (2.35)$$

We built a 3-level pyramid, with a scale of 0.5 between levels. The data association based on perspective projection and the optimization based on the

point-plane metric is performed in each level, and the result in the coarsest level is used as initial value in the iterative optimization on the following finest level, using Levenberg-Marquardt. With this scheme we achieve a faster convergence in the finest levels (where the computations take more time), and better estimations when faster camera motion occurs.

2.5 RESULTS

In this section we present the results for both frame-to-frame approach and frame-to-model approach, considering the accuracy and the performance with respect to time. For the first approach we also evaluate the convergence of the algorithm when the coarse-to-fine scheme is used.

2.5.1 RESULTS WITH THE FRAME-TO-FRAME APPROACH

The dense localization algorithm runs on a laptop Hewlett Packard with a processor Intel Core *i7* - 2.2 GHz, 11.7 GB of RAM memory, a graphics processing unit NVIDIA GEFORCE 755M and Ubuntu 14.04 as operating system. OpenCV is employed for processing images, OpenGL for drawing the camera trajectory and the keyframes, and Cuda 6.0 for getting a better performance using a GPU. The following analysis is made using the synthetic RGB-D dataset of A. Handa [36] and is based on three criteria; accuracy, convergence and time.

ACCURACY

The accuracy in the camera position is quantified using three datasets; two of them consist of perfect ray-traced images taken at 20 fps and 60 fps and the last one consists of images with motion blur and noise, taken at 60 fps (60 fps-blur). We define the error as the Euclidean distance between the ground truth in trajectory and the estimated one. Figure 2.8(a) shows the estimated trajectory (in orange) and the ground truth (in green) for the 60 fps dataset, obtained in OpenGL and drawn incrementally as the relative transformations are computed and concatenated. The positions of keyframes are drawn as blue points. The analysis in orientation was made for each axis, comparing the ground truth in orientation with the estimated one along the time. The ground truth in camera rotation around the axis X, Y and Z are shown in Fig. 2.8(b).

In Fig. 2.9(a) we draw the error in camera position for the 60 fps dataset (no blur) as a solid red line and the error in camera position for the 60 fps-blur dataset as a dashed blue line, and highlight the errors of keyframes by drawing them as points. Figure 2.9(b) shows the error in rotations around X, Y, and Z for the 60 fps dataset and Fig. 2.9(c) shows the corresponding errors for the

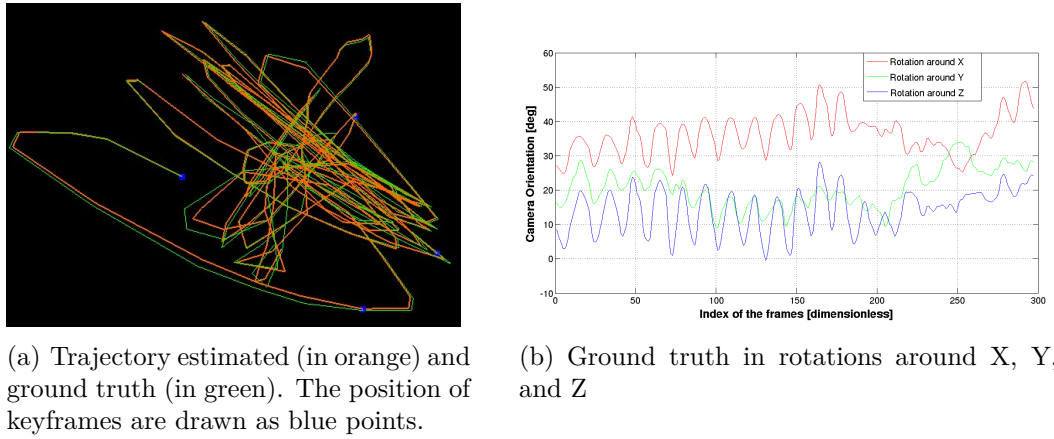


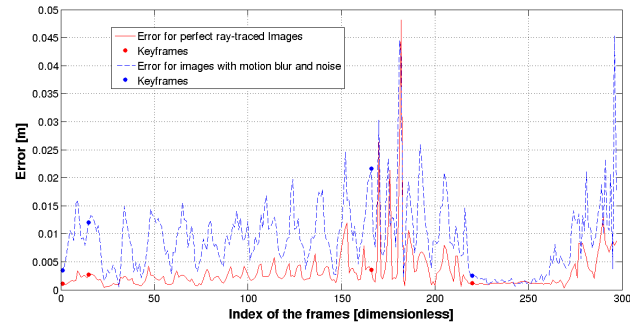
Figure 2.8: (a) Trajectory and (b) orientation of the camera for the 60 fps dataset. The positions of keyframes are drawn as blue points in (a).

60 fps-blur dataset. Note that the errors in both position and orientation for the images with motion blur and noise are greater than the errors for perfect ray-traced images. Errors obtained with the 20 fps dataset are very similar to the error of the 60 fps dataset (solid red line in Fig. 2.9(a) for camera position and errors in Fig. 2.9(b) for camera orientation) so they are not shown.

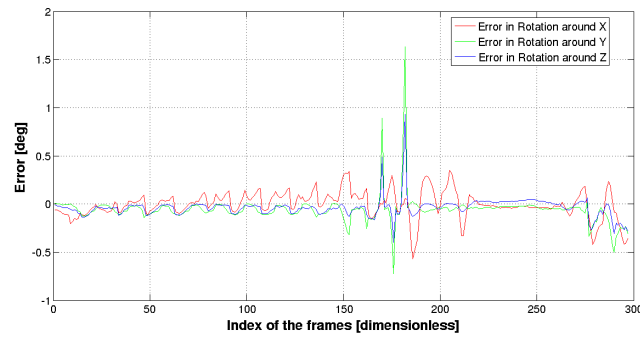
Table 2.1 summarizes the accuracy in camera position for the three datasets. With the 60 fps dataset (no blur) the RMSE and standard deviation decrease although the maximum error increases, compared to the ones obtained with the 20 fps dataset. When the 60 fps-blur dataset is used, the RMSE and the standard deviation increases due to difficulties for getting a good initial estimation and for aligning images with blur and noise. Despite these difficulties, the algorithm converges to a good estimation. We can say that the system has high accuracy in position for both frame rates, and for both perfect ray-traced images and images with blur and noise; in the worst case, the error is 0.16 % of the distance navigated by the camera that corresponds to 6.93 m.

Table 2.2 summarizes the accuracy in the estimated rotations for the three datasets. The RMSE and the standard deviation decrease with the 60 fps dataset although the maximum error increases for rotations around Y and Z. When the 60 fps-blur dataset is used, the RMSE and the standard deviation increase due to the reasons given for the errors in camera position when the images have blur and noise. Note that the maximum error is 1.63° and the minimum error is -1.90° so we finally can claim that the system has high accuracy in both camera position and camera orientation for different frame rates and even with motion blur and noise.

For comparison purposes, we use the information given in [25] about the RMSE in camera position for several localization systems. In it, four datasets were



(a) Error in camera position for 60 fps and 60 fps-blur datasets



(b) Error in rotations around X, Y, and Z for perfect ray-traced images (60 fps dataset)



(c) Error in rotations around X, Y, and Z for images with motion blur and noise (60 fps-blur dataset)

Figure 2.9: Error in camera pose (position and orientation). The keyframe positions are drawn as blue points in (a).

Table 2.1: Summary of accuracy in camera position

Quantity \ Dataset	20 fps	60 fps	60 fps-blur
RMSE [cm]	0.74	0.52	1.12
Max. error [cm]	4.2	4.82	4.54
Mean error [cm]	0.44	0.33	0.92
Median error [cm]	0.25	0.22	0.87
Standard dev. [cm]	0.59	0.41	0.65

Table 2.2: Summary of accuracy in camera orientation

Quantity \ Dataset	20 fps			60 fps			60 fps-blur		
	X	Y	Z	X	Y	Z	X	Y	Z
RMSE [deg]	0.24	0.21	0.12	0.14	0.16	0.10	0.31	0.34	0.43
Max. error [deg]	0.74	1.52	0.75	0.35	1.63	0.93	1.06	1.34	0.70
Min. error [deg]	-0.64	-0.44	-0.24	-0.57	-0.72	0.40	-1.66	-1.15	-1.90
Mean error [deg]	-0.11	-0.02	-0.04	-4e-3	-0.06	-0.04	-0.02	-0.23	-0.34
Median Error [deg]	-0.13	-0.04	-0.06	-0.02	-0.05	-0.03	-2.9e-3	-0.22	-0.30
Standard dev. [deg]	0.21	0.21	0.11	0.14	0.15	0.10	0.31	0.25	0.27



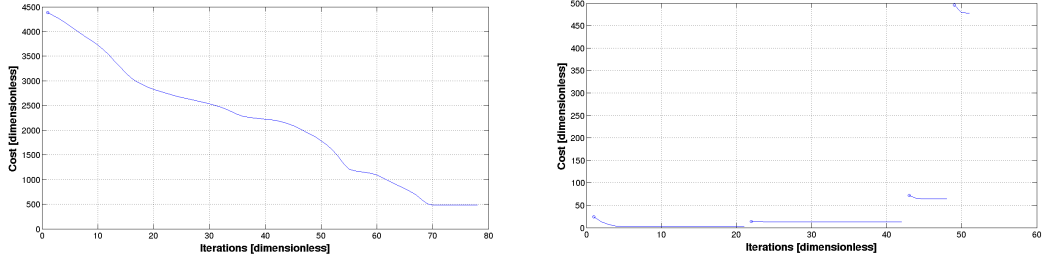
Figure 2.10: Dense Map built using depth maps of keyframes. The trajectory is drawn in orange and keyframes are blue spheres.

used; two real sequences from the TUM RGB-D benchmark [91] and two simulated sequences from [36]. These two synthetic datasets, called sim/desk and sim/slowmo, and the ones that we use in this work, 20 fps, 60 fps and 60 fps-blur, have the same author, A. Handa, so we compare the accuracy using the information of sim/slowmo. Our system, with a RMSE of 0.52 cm for the 60 fps dataset, outperforms the semi-dense mono VO [26] that has a RMSE of 2.21 cm. However, our system has inferior accuracy w.r.t. LSD-SLAM [25] and direct RGB-D SLAM that have 0.35 cm and 0.13 cm, respectively. It may be due to the use of a probabilistic approach that incorporates information about noise on the depth maps into tracking and the use of a framework for graph optimization.

The dense map shown in Fig. 2.10 was built offline with point cloud library 1.7.0 and using depth maps of keyframes. Note that all the local clouds, after being referenced to a global coordinate system, fit tightly.

CONVERGENCE

We analyse the performance of the algorithm when it uses a coarse-to-fine scheme. We present the number of iterations required by the algorithm for



(a) Convergence without using a coarse-to-fine scheme. (b) Convergence using a four level pyramid.

Figure 2.11: Comparison between using only one image and a four-level pyramid. (a) requires 76 iterations with an image of 480×640 pixels for converging. (b) requires 2 iterations in the lower level (image of 480×640 pixels) for converging.

Table 2.3: Information about convergence using a coarse-to-fine scheme

Level	Size [pix.]	Initial Cost	Final Cost	Max. Iterations
4	60×80	24.27	2.47	20
3	120×160	13.89	12.67	20
2	240×320	72.00	64.20	5
1	480×640	495.44	477.72	2

converging before and after implementing this scheme. In both cases the initial relative transformation is $T_{cr} = \begin{bmatrix} R_{cr} & 0.6\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$, where R_{cr} and $\mathbf{t}$ are respectively the ground truth in rotation and translation, that relates frame r and c . The images 1 and 10 from the 20 fps dataset were used as $\mathbf{I}_r$ and $\mathbf{I}_c$, respectively. For the case in which a coarse-to-fine scheme is not used, the initial cost is 4386.8. In the iteration 76, the change in the motion vector $d\boldsymbol{\xi}$ is less than $1e - 6$ and the algorithm stops, converging to a cost of 481.9.

When a four level pyramid with a scale of 0.5 is built, the maximum number of iterations in each level is set to: 20, 20, 5 and 2, from higher to lower levels. The initial and final cost for each level is presented in table 2.3.

Note that the initial cost in the lowest level (bigger image) is very close to the final cost, unlike in the first case. For this reason only two iterations are required in the lowest level when a coarse-to-fine scheme is implemented.

TIME

We have implemented the most important algorithms of the dense localization module on GPU. The Jacobian is computed in 11.1 ms in the lowest level (the full image, 640×480 pixels). The warped image and the residual is computed in 3.9 ms for the same level. The update in the camera position and orientation is made in 5.3 ms. The other processes are implemented on CPU. The building

of the pyramid is made in 1.6 ms. The load of images I_r and I_{c_i} is made in 3.5 ms. The initial estimation is computed in 31.3 ms. The system processes 10 fps, for two iterations of the Levenberg-Marquardt algorithm in the lowest level of a four-level pyramid.

2.5.2 RESULTS WITH FRAME-TO-MODEL APPROACH

Our dense tracking algorithm, based on frame-to-model approach, makes part of a simultaneous dense tracking and mapping system that runs on a laptop ASUS G551VW with a processor Intel Core i7-2.6GHz, 8GB of RAM memory, a graphic processor NVIDIA GEFORCE 960M and Ubuntu 14.04 as operating system. We carried out experiments with three sequences from the TUM RGB-D dataset [91] that provides color and depth images from a Kinect sensor, with camera pose estimated with a high-accuracy motion-capture system. We also use both a synthetic dataset of A. Handa [36] which provides the ground truth in camera pose (perfect ray-traced images taken at 100 fps) and our own data from the Kinect 1.0 moved by hand in a room-sized scene. OpenCV is employed for processing images, Eigen3 for some matrix operations, OpenGL for drawing the 3D model, and for getting depth and color information from the model, Cuda 7.5 and Thrust library for improving the performance using commodity graphics hardware and OpenNI for working with the Kinect. Experiments are made for computing the accuracy of the tracking algorithm. We will also evaluate its performance with respect to time.

ACCURACY IN CAMERA POSE

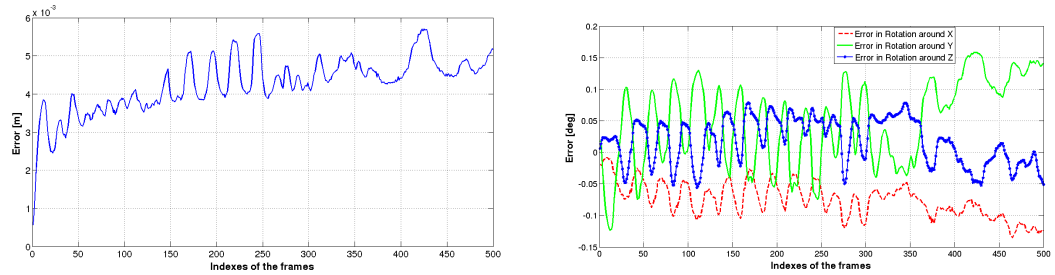
For these experiments, we use the synthetic and the TUM RGB-D dataset. The voxel grid has $128 \times 128 \times 128$ voxels in each dimension, covering a volume of $1.8 \times 1.5 \times 1$ m. The volumetric structure is placed in $(-1.2, -0.8, 0.7)$ m with respect to the first coordinate system of the camera (initialized by the user). The final reconstruction has 788373 vertices. Figure 2.12 shows the resulting 3D reconstruction with color data coming from the scene. More details about the fusion of depth maps and the volumetric grid will be given in chapter 6.

The maximum number of iterations for camera tracking in each level is 4, 8, 10, from fine to coarse level. We define the error in camera position as the Euclidean distance between the ground truth and the estimations. The behavior of the error in camera position around the whole trajectory is shown in fig. 2.13(a). The analysis in orientation is made in each axis, as the difference between the ground truth and the estimated values. These errors are shown in fig. 2.13(b). Table 2.4 summarizes the quantization of the errors in camera position and orientation. Note that the root mean square errors RMSE in camera position (less than 1cm) and in camera orientation (less than one tenth



Figure 2.12: 3D reconstruction resulting from fusing synthetic depth maps and color of the scene, from the estimated camera pose.

of a degree) show the high accuracy of the tracking algorithm. Moreover, the error in camera position remains stable after 500 iterations which indicates low drift in camera tracking. The RMSE in camera position is 0.59% of the distance navigated by the camera that corresponds to 7.27m.



(a) Error in camera position for synthetic data.

(b) Error in camera rotation for synthetic data.

Figure 2.13: Error in camera pose. (a) Error in camera position. (b) Error in camera orientation. The errors remain small with low drift after completing the whole trajectory.

For the TUM RGB-D dataset, we chose three sequences that were taken in room-sized environments, the fr1/desk, fr1/desk2 and fr1/xyz. For the first sequence (similar configuration for the remaining two sequences), the voxel grid has $128 \times 128 \times 128$ voxels, covering a volume of $3 \times 2 \times 3$ m. The volumetric structure is placed in $(-2.5, -1.1, 0.3)$ m with respect to the first coordinate system of the camera. The final reconstruction has 763086 vertices. The maximum number of iterations for camera tracking in each level is 4,8,10, from fine to coarse level. Figure 2.14 shows the resulting 3D reconstruction with color data coming from the scene.

In table 2.5 we can see that KinectFusion [61] and our system has similar accuracy (except for fr1/desk2 where [61] has a high error). It is due to similarities

Table 2.4: Summary of accuracy in camera pose

Quantity \ Variable	Loc.[cm]	Rot. X[deg]	Rot. Y[deg]	Rot. Z[deg]
RMSE	<i>0.4314</i>	0.0802	0.0820	0.0402
Max. error	0.5701	-0.0080	0.1587	0.0785
Min. error	0.0559	-0.1352	-0.1238	-0.0561
Mean error	0.4256	-0.0754	0.0476	0.0181
Median error	0.4325	-0.0745	0.0593	0.0229
Standard dev.	0.0701	0.0274	0.0668	0.0359

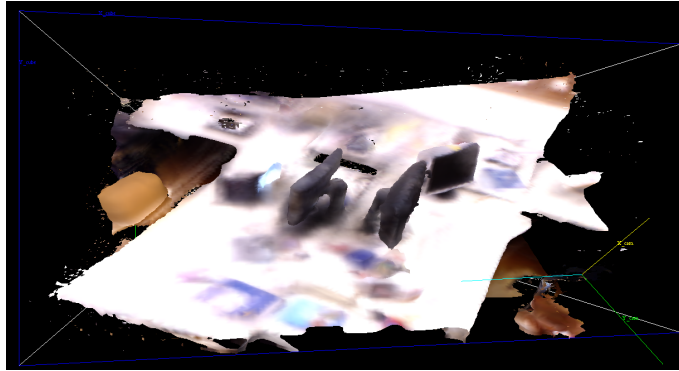


Figure 2.14: Dense reconstruction for the sequence fr1/desk of TUM, with color from the scene.

in the design: dense ICP for tracking and volumetric structure with TSDF values for the model. More recent systems [97],[98], [59], [9], use both photometric and geometric error for estimating the camera pose. Moreover, they include modules for achieving loop closure detection and global consistency with techniques such as non-rigid deformation, pose-graph optimization and bundle adjustment. These systems are more accurate and robust than [61] and our system, especially when performing in complex trajectories (see table 2.5). Note that ORB-SLAM2 [59] is the most accurate system for tracking.

Table 2.5: TUM RGBD dataset. Comparison of translation RMSE (m)

Sequence	Ours	Kinect Fusion [61]	Kintinuou [97]	Elastic-Fusion [98]	ORB-SLAM2 [59]	RGBD TAM [9]
fr1/desk	0.052	0.057	0.037	0.020	0.016	0.027
fr1/desk2	0.083	0.420	0.071	0.048	0.022	0.042
fr1/xyz	0.036	0.026	0.017	0.011	—	0.010

TIME

All the main processes that make up the proposed system were implemented on commodity graphics hardware. Table 2.6 shows the average on processing time for marching cubes, fusion of depth maps, rasterization, and tracking of

the depth camera. With this performance, the system can reconstruct the scene and estimate the camera pose at 10 fps. Note that tracking is the process with more time consumption. It works with three pyramid levels and $\{10, 8, 4\}$ iterations, from coarse to fine levels.

Table 2.6: Processing time of the main modules

Process	Time[ms]
Marching cubes	8.322
Fusion	17.891
Rasterization	8.657
Tracking	68.417
TOTAL	103.297

2.6 CONCLUSIONS

We have developed two algorithms that employ all the data of the intensity and depth images for estimating the pose of a camera that is moved with 6DOF in a small environment. The first algorithm is based on a frame-to-frame approach and the second one is based on a frame-to-model approach.

The first algorithm (frame-to-frame approach) uses a 3D-2D visual odometry technique for computing an initial estimation that is later refined using image alignment. In order to reduce the computational cost, the current camera pose is computed with respect to the closest keyframe stored in the system. The results with the frame-to-frame approach have been satisfactory. First, using a coarse-to-fine scheme we have improved the convergence of the system, requiring just two iterations of the Levenberg-Marquardt algorithm at the lowest level of our four-level pyramid. As a result, we have boosted the performance of the system achieving to process 10 fps on a GPU. Second, using the proposed two-steps process to localize the camera with respect to a keyframe -3D to 2D followed by the image alignment- results in high-accuracy estimations, even when using blurred input images. In the worst case, with blur and noise, the RMSE is 1.12 cm in translation and 0.31° , 0.34° , and 0.43° in the rotations around X, Y and Z axis respectively. Our system, with a RMSE of 0.52 cm for the 60 fps dataset, outperforms the semi-dense mono VO [26] that has a RMSE of 2.21 cm. However, our system has inferior accuracy w.r.t. LSD-SLAM [25] and direct RGB-D SLAM that have 0.35 cm and 0.13 cm, respectively. For future work, we will concentrate on the implementation of techniques for loop closure and global consistency that make the system more robust to complex trajectories.

The second algorithm (frame-to-model approach) uses coarse-to-fine iterative closest point (ICP), achieving low drift, robustness to fast camera motion and

to variable light conditions. The results with this approach also have been satisfactory. First, the ICP algorithm with a coarse-to-fine scheme produces high-accuracy estimations in camera tracking. For the experiment with synthetic dataset, the RMSE is 0.431cm in translation (just 0.59% of the distance navigated by the camera) and 0.080° , 0.082° , and 0.040° , in rotation around the X, Y, and Z axis, respectively. For the experiment with the TUM RGB-D dataset, we got 5.2cm, 8.3cm and 3.6cm for translation RMSE. The accuracy of our system is on par with [61] but lower than systems such as [97],[98], [59], [9] that integrate techniques for achieving global consistency such as appearance-based place recognition for loop closure detection, non-rigid space deformation, pose-graph optimization and bundle adjustment. Second, the implementation was done in commodity graphics hardware, achieving to fuse depth maps and track the camera at 10 fps (with color management), which is appropriate for the data association carried out for dense ICP. For future work, we will integrate the RGB data in the objective function for tracking the sensor and implement techniques for loop closure and global consistency that make the system more robust to complex trajectories.

CHAPTER 3

DEPTH MAP BUILDING USING A MONOCULAR CAMERA

3.1 INTRODUCTION

For building a depth map with a monocular camera its location for a set of frames must be known, photo-consistency must be satisfied and the images must have texture. However, real environments present changes in light conditions and low-texture, very reflective or translucent objects. This fact breaks the Lambertian condition (photo-consistency) affecting the estimated depth maps. The regularisation term in a variational framework tackles this problem to some extent, but under difficult conditions, the accuracy is low.

This chapter has as main objectives to present the difficulties when the Lambertian condition is not fulfilled and quantify the accuracy of the built depth maps. In section 3.2 similar works for building depth maps with a monocular camera are presented. In section 3.3 we explain the difficulties for minimizing the photometric error when complex conditions occur. The refinement of the initial estimation of the depth map is detailed in 3.4. Finally, in sections 3.5 and 3.6 we present the accuracy of the estimated depth maps with real and synthetic data and the conclusions, respectively.

3.2 RELATED WORK

Two techniques for minimizing the energy functional in the process of building a depth map with a monocular camera stand out. The technique of sequential convex optimization linearises the photometric error, as is explained in [101], so the camera motion must be small. A coarse-to-fine scheme with a pyramid of several levels is built to copy with fast camera motion. This technique was successfully implemented in the work of dense mapping of [90]. The non-convex variational technique, based on optical flow for long displacements [86] uses an auxiliary variable that decouples the regularisation term and the data

term in the cost function. The regularisation term is solved with the *primal-dual* algorithm [5], [104], and the data term is solved by an exhaustive search over a finite range of discrete values of the inverse depth. This technique was implemented in the work of dense location and dense mapping of [62].

Different norms applied to the regularisation term have been analyzed, such as the total variation norm, the Huber norm, the total generalized variation norm and the non-convex logarithmic norm, as is done in [70]. The photometric norm can have L1 norm, L2 norm squared, truncated linear penalizer of the intensity, L1 norm on a patch, normalized cross correlation NCC on patches, among others [85]. These variational techniques are highly parallelizable, so it can be accelerated using modern graphics processing units.

3.3 BUILDING AN INITIAL DEPTH MAP

A 3D point in the reference frame r is represented as $\mathbf{X}_r = (X_r, Y_r, Z_r)$. This 3D coordinate can be computed as

$$\mathbf{X}_r(\boldsymbol{\mu}) = \zeta_r^{-1}(\boldsymbol{\mu}) K^{-1} \dot{\boldsymbol{\mu}}, \quad (3.1)$$

where $\dot{\boldsymbol{\mu}}$ is the homogeneous version of the coordinate in the image plane, that is $\dot{\boldsymbol{\mu}} = (u_r, v_r, 1)^T$, K is the intrinsic camera matrix, and ζ_r is the inverse depth. We use inverse depth instead of depth because an uniform sampling of the inverse depth corresponds to a uniform sampling in epipolar lines in the image, allowing the system to make exhaustive search to solve the photometric error and the decoupling term in the refinement process. The 3D point $\mathbf{X}_r$ can be referenced to the camera frame c , as follows:

$$\mathbf{X}_c(\boldsymbol{\mu}) = T_{cr} \mathbf{X}_r(\boldsymbol{\mu}). \quad (3.2)$$

This 3D point is projected into the image plane $\mathbf{I}_c$ for getting the coordinate $\boldsymbol{\mu}_c = (u_c, v_c) = \dot{\boldsymbol{\mu}} = K\pi(\mathbf{X}_c)$ that corresponds to the estimated observation of the 3D point from the new camera pose (see fig. 3.1), where π represents the function that computes normalized homogeneous coordinates

$$\pi(\mathbf{X}_c) = (X_n, Y_n, 1)^T = \left(\frac{X_c}{Z_c}, \frac{Y_c}{Z_c}, 1 \right)^T. \quad (3.3)$$

If the same process is made for all the pixels of the reference image, a synthetic image that represents the scene observed from the pose defined by the transformation matrix T_{cr} , can be estimated. However, the inverse depth of the pixel $\zeta_r(\boldsymbol{\mu})$, required in eq. (3.1), is unknown. To estimate the inverse depth of each pixel of the image $\mathbf{I}_r$, the photometric error is defined as the difference in intensity between a pixel of the reference image $\mathbf{I}_r(\boldsymbol{\mu})$ and a pixel of the current image in the projected coordinate $\mathbf{I}_c(\dot{\boldsymbol{\mu}})$, that is,

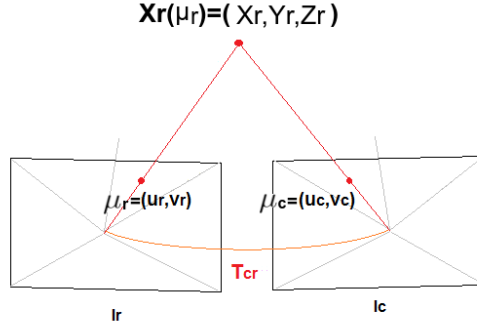


Figure 3.1: Projection of a 3D point referenced to the coordinate system r into the image plane c , using the transformation T_{cr} .

$$\rho_r(\mathbf{I}_c, \boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu})) = \mathbf{I}_r(\boldsymbol{\mu}) - \mathbf{I}_c(\dot{\boldsymbol{\mu}}) = \mathbf{I}_r(\boldsymbol{\mu}) - \mathbf{I}_w(\boldsymbol{\mu}) \quad (3.4)$$

where the warped image $\mathbf{I}_w(\boldsymbol{\mu})$ is

$$\mathbf{I}_w(\boldsymbol{\mu}) = \mathbf{I}_c(K\pi(T_{cr}\zeta_r^{-1}(\boldsymbol{\mu})K^{-1}\dot{\boldsymbol{\mu}})). \quad (3.5)$$

as T_{cr} is supposed to be known, the true value of $\zeta_r(\boldsymbol{\mu})$ minimizes the photometric error. Now, this process is extended to a set of consecutive images $\mathbf{I}_{c_i} : i \in [1, n]$ that share common elements of the scene. An average photometric error is defined as a function of the inverse depth $\zeta_r(\boldsymbol{\mu})$:

$$\mathbf{C}_r(\boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu})) = \frac{1}{n} \sum_{i=1}^n \|\rho_r(\mathbf{I}_{c_i}, \boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu}))\|_1, \quad (3.6)$$

where ρ was defined in the eq. (3.4). Finally, the problem of finding the inverse depth of a pixel $\zeta_r(\boldsymbol{\mu})$ is equivalent to solve

$$\min_{\zeta_r(\boldsymbol{\mu})} \mathbf{C}_r(\boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu})). \quad (3.7)$$

Assuming photo-consistency, the intensity of the observation taken by a camera of a 3D point from two poses must be equal, it means, the change in intensity of the pixels must be just due to the camera motion. However, in real environments there are changes in light conditions, which break the assumption of photo-consistency and the estimations are affected drastically. Besides, images of real scenes present regions with low texture that generate depth estimations in the most dominated by noise [10]. Note in figure 2.2(a) that regions with low texture in the scene like the top of the printer, the screen of the monitor, or the wall, are prone to false minima (there is not a clear minimum as can be seen in fig. 3.2(b) for a pixel of the printer) while regions with rich texture like the pictures in the wall, or the magazine over the desk, have a clear minimum (seen 3.2(a) for a pixel of the magazine). For

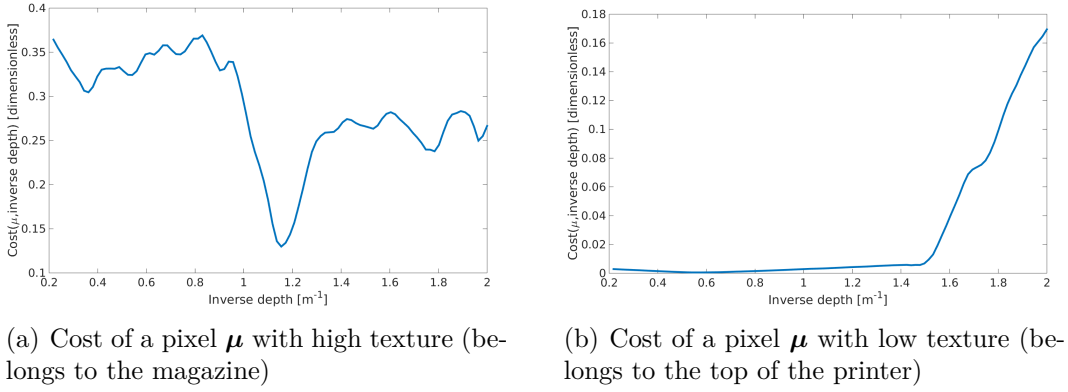


Figure 3.2: (a) Pixel in a high-textured region produces a clear minimum, while (b) pixel in a low-textured region produces a flat cost.

this reason, a regularisation term is used in a variational framework with the result of eq. (3.6) as initial depth map in an iterative process.

3.4 REFINING THE INITIAL DEPTH MAP WITH VARIATIONAL METHODS

A variational approach [80] is adopted for smoothing the depth map, preserving discontinuities and increasing the robustness of the algorithm against illumination changes, occlusions and noise. It was proposed first by Ruding, Osher and Fatemi ROF to consider the “Total Variation” as a regulariser $\int_{\Omega} |\nabla h(\mu)| d\mu$, for functions $h(\mu)$ in the Sobolev space $W^{1,1}$. The big advantage is that it is convex in the variable h , so this problem has a unique solution. For the pure denoising case [75] it is

$$\min_h \int_{\Omega} \underbrace{\|\nabla h(\mu)\|_1}_{\text{regularisation term}} d\mu + \frac{\lambda}{2} \int_{\Omega} \underbrace{\|h(\mu) - g(\mu)\|_2^2}_{\text{data term}} d\mu \quad (3.8)$$

where h is the sought solution and g is the noisy input image. The parameter λ defines the tradeoff between regularisation and data fitting. In our context, the data term measures the photo-consistency between images, the regularisation term smooths surfaces preserving discontinuities and λ plays the same role as in eq. (3.8). The energy functional is

$$E(\mu, \zeta_r(\mu)) = \int \left(\underbrace{w(\mu) \|\nabla \zeta(\mu)\|_{\epsilon}}_{\text{regularisation term}} + \underbrace{\lambda C(\mu, \zeta_r(\mu))}_{\text{data term}} \right) d\mu, \quad (3.9)$$

where $w(\mu)$ is a weighting function, $C(\mu, \zeta(\mu))$ is the photometric error, and $\|\cdot\|_{\epsilon}$ is the Huber norm over the gradient of the inverse depth map, with

$$\|\mathbf{x}\|_\epsilon = \begin{cases} \frac{\|\mathbf{x}\|_2^2}{2\epsilon} & \text{if } \|\mathbf{x}\|_2 \leq \epsilon, \\ \|\mathbf{x}\|_1 - \frac{\epsilon}{2} & \text{otherwise} \end{cases} \quad (3.10)$$

The L_2^2 norm promotes smooth reconstructions while the L_1 norm (total variation regulariser) allows discontinuities at depth edges. As depth discontinuities often coincide with edges in the reference image, the per pixel weight $w(\boldsymbol{\mu})$ is

$$w(\boldsymbol{\mu}) = e^{-\alpha \|\nabla \mathbf{I}_r(\boldsymbol{\mu})\|_2^\beta}, \quad (3.11)$$

Reducing the regularity strength where the edge magnitude is high, thereby limiting solution smoothing across region boundaries. The problem of computing the inverse depth map becomes

$$\min_{\zeta_r(\boldsymbol{\mu})} E(\boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu})) \quad (3.12)$$

where E is the energy defined in eq. (3.9). It is a non-convex problem: the regularisation term is convex and the photometric error is not convex. Next, we describe how to solve it with a decoupling term.

3.4.1 THE DECOUPLING APPROACH

In order to solve eq. (3.12), we use the iterative primal-dual algorithm described in [62] for depth map building. This algorithm requires both the regulariser and the data term to be convex. However, the last term is not a convex function. One solution to this problem is to decouple both terms and solve the decoupled version instead of the original one. The advantage of the decoupling approach is that it allows us to independently solve for the regularisation term using convex optimisation methods and for the data term using a simple exhaustive search. The decoupling approach is based on eliminating the constraint $\zeta(\boldsymbol{\mu}) = \eta(\boldsymbol{\mu})$ of the problem

$$\begin{aligned} \min_{\zeta, \eta} & E_{\text{reg}}(\zeta(\boldsymbol{\mu})) + E_{\text{data}}(\eta(\boldsymbol{\mu})) \\ \text{s.t.} & \quad \zeta(\boldsymbol{\mu}) = \eta(\boldsymbol{\mu}), \end{aligned} \quad (3.13)$$

where $E_{\text{reg}}(\zeta(\boldsymbol{\mu})) = w(\boldsymbol{\mu}) \|\nabla \zeta_r(\boldsymbol{\mu})\|_\epsilon$, $E_{\text{data}}(\eta(\boldsymbol{\mu})) = \lambda \mathbf{C}(\boldsymbol{\mu}, \eta(\boldsymbol{\mu}))$ and $\eta(\boldsymbol{\mu})$ is an auxiliary variable, through the use of a penalty function. Using this approach, eq. (3.13) is minimised by sequentially solving an unconstrained minimisation problem of the form

$$\min_{\zeta(\boldsymbol{\mu}), \eta(\boldsymbol{\mu})} E_{\text{reg}}(\zeta(\boldsymbol{\mu})) + \frac{1}{2\theta} \|\zeta(\boldsymbol{\mu}) - \eta(\boldsymbol{\mu})\|_2^2 + E_{\text{data}}(\eta(\boldsymbol{\mu})) \quad (3.14)$$

enforcing $\zeta(\boldsymbol{\mu}) = \eta(\boldsymbol{\mu})$ as $\theta \rightarrow 0$ and therefore $E(\zeta(\boldsymbol{\mu}), \eta(\boldsymbol{\mu})) \rightarrow E(\zeta(\boldsymbol{\mu}))$. This new energy functional allows us to split the minimisation into two different

problems that are alternately solved until convergence. The regularisation term and the decoupling term,

$$w(\boldsymbol{\mu})\|\nabla\zeta_r(\boldsymbol{\mu})\|_\epsilon + \frac{1}{2\theta}\|\zeta(\boldsymbol{\mu}) - \eta(\boldsymbol{\mu})\|_2^2, \quad (3.15)$$

correspond to the TV-ROF denoising problem, defined in eq. (3.8). It is convex in $\zeta(\boldsymbol{\mu})$ and can be solved using a primal-dual algorithm. Doing an analogy with image denoising, $\eta(\boldsymbol{\mu})$ represents a noisy image whereas $\zeta(\boldsymbol{\mu})$ is the searched denoised result. The non convex in the auxiliary variable $\eta(\boldsymbol{\mu})$, $\frac{1}{2\theta}\|\zeta(\boldsymbol{\mu}) - \eta(\boldsymbol{\mu})\|_2^2 + \lambda\mathbf{C}(\boldsymbol{\mu}, \eta(\boldsymbol{\mu}))$ is point-wise optimisable and the solution is exhaustively searched over a finite range of discretely sampled inverse depth values. The energy with the decoupling term of eq. (3.14) can be written as follows:

$$\min_{\mathbf{y}, \mathbf{z}} \underbrace{\|AW\mathbf{y}\|_\epsilon}_{\text{regularisation term}} + \underbrace{\frac{1}{2\theta}\|\mathbf{y} - \mathbf{z}\|_2^2}_{\text{decoupling term}} + \underbrace{\lambda\mathbf{C}(\mathbf{z})}_{\text{data term}}, \quad (3.16)$$

where $A = \nabla$ is the gradient operator, W is the element-wise weighting matrix, $\mathbf{y}$ and $\mathbf{z}$ are row-wise vector versions of the sought solution $\zeta(\boldsymbol{\mu})$ and auxiliary variable $\eta(\boldsymbol{\mu})$ respectively. The regulariser and the decoupling term of eq. (3.16) for a fixed auxiliary variable $\mathbf{z}$ have the more general form

$$\min_{\mathbf{y}} F(A\mathbf{y}) + G(\mathbf{y}), \quad (3.17)$$

In our case, the convex functions are $F(A\mathbf{y}) = \|AW\mathbf{y}\|_\epsilon$, $G(\mathbf{y}) = \frac{1}{2\theta}\|\mathbf{y} - \mathbf{z}\|_2^2$. We do a Legendre-Fenchel transform:

$$F(A\mathbf{y}) = \max_{\boldsymbol{\varrho}, \|\boldsymbol{\varrho}\|_2 \leq 1} \langle AW\mathbf{y}, \boldsymbol{\varrho} \rangle - F^*(\boldsymbol{\varrho}) \quad (3.18)$$

where $\boldsymbol{\varrho}$ is the dual of $\mathbf{y}$ and $F^*(\boldsymbol{\varrho})$ is the conjugate of $F(A\mathbf{y})$:

$$F^*(\boldsymbol{\varrho}) = \delta(\boldsymbol{\varrho}) + \frac{\epsilon}{2}\|\boldsymbol{\varrho}\|_2^2 \quad (3.19)$$

$$\delta(\boldsymbol{\varrho}) = \begin{cases} 0 & \text{if } \|\boldsymbol{\varrho}\|_1 \leq 1 \\ \infty & \text{if otherwise} \end{cases} \quad (3.20)$$

Replacing $F(A\mathbf{y})$ and $G(\mathbf{y})$ in (3.17), we get the primal-dual formulation of this problem. It is a generic saddle-point problem

$$\min_{\mathbf{y}, \mathbf{z}} \max_{\boldsymbol{\varrho}, \|\boldsymbol{\varrho}\|_2 \leq 1} E = \langle AW\mathbf{y}, \boldsymbol{\varrho} \rangle + \frac{1}{2\theta}\|\mathbf{y} - \mathbf{z}\|_2^2 - \delta(\boldsymbol{\varrho}) - \frac{\epsilon}{2}\|\boldsymbol{\varrho}\|_2^2 \quad (3.21)$$

We do a step of projected gradient ascent (maximization problem) for the dual variable $\boldsymbol{\varrho}$ and one step of gradient descent (minimization problem) for the primal variable $\mathbf{y}$, resulting the updates

$$\begin{cases} \boldsymbol{\varrho}^{n+1} = \text{proj}_{\boldsymbol{\varrho}}((\boldsymbol{\varrho}^n + \sigma W A \mathbf{y}^n)/(1 + \sigma\epsilon)) \\ \mathbf{y}^{n+1} = \frac{\mathbf{y}^n - \tau(W A^* \boldsymbol{\varrho}^{n+1} - \frac{1}{\theta^n} \mathbf{z}^n)}{1 + \frac{\tau}{\theta^n}} \end{cases} \quad (3.22)$$

where A^* is the adjoint operator of the gradient operator and corresponds to the negative divergence operator, σ and τ are the step size for the dual variable $\boldsymbol{\varrho}$ and primal variable $\mathbf{y}$ respectively, $\text{proj}_{\boldsymbol{\varrho}}(x) = x / \max(1, \|x\|_2)$ projects the gradient ascent step back onto the constraint $\|\boldsymbol{\varrho}\|_1 \leq 1$. Finally, for a fixed (and updated) $\mathbf{y}$ we use a point-wise search to solve the remaining non-convex function

$$\arg \min_{\mathbf{z}^{n+1}} E^{aux} = \frac{1}{2\theta^n} (\mathbf{y}^{n+1} - \mathbf{z}^{n+1})^2 + \lambda \mathbf{C}(\mathbf{z}^{n+1}) \quad (3.23)$$

The primal-dual algorithm and the exhaustive search are alternated and θ^n is decreased in each step until convergence, it means, until $\theta^n \leq \theta_{th}$, where θ_{th} is a predefined threshold.

3.5 RESULTS

We carry out two experiments for computing the accuracy of the estimated depth maps: one with synthetic data and the other one with real data. The last one will be enhanced with shape prior data using variational methods in chapter 5. Next, we present the parameter settings and the results.

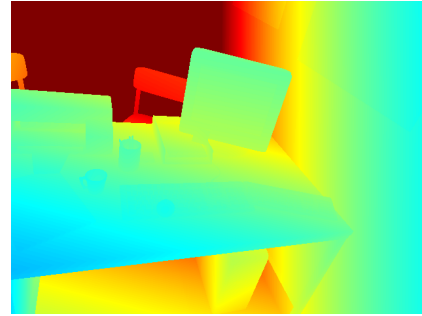
3.5.1 RESULTS WITH SYNTHETIC DATA

For the first experiment we use 40 images ($n = 40$) and one reference image from the dataset of Ankur Handa [36]. The range of depth is defined as $[0.5 \ 5]\text{m}$ that corresponds to an inverse depth range of $[2 \ 0.2]\text{m}^{-1}$. The number of samples (linear sampling in inverse depth) is $n_s = 100$, the balance λ is 1, the threshold for the Huber norm ϵ is 0.01, α and β of eq. (3.11) are 0.4 and 2.4 respectively. Figures 3.3(a), 3.3(b) and 3.3(c) show the image of reference in gray, the ground truth in depth and the initial depth map obtained by minimising eq. (3.7), respectively. Figure 3.3(d) shows the refined depth map after 200 iterations. Figures 3.4(a) presents the depth data for row 240 using just the data term while fig. 3.4(b) shows the smoother version of data got with the regularisation term. Figures 3.4(c) and 3.4(d) show the absolute error and the error distribution for the solution when the regularisation term is included. The same analysis is done for data around column 280 (figs. 3.4(e)-3.4(h)).

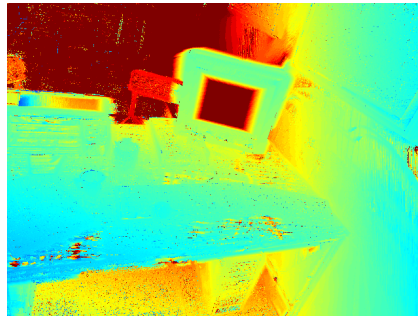
Figures 3.5(a) and 3.5(b) show the absolute error for the whole depth map using just the data term and both the data term and the regularisation term, respectively, while fig. 3.5(c) presents the error distribution. Note that the solution with regularisation term is smoother and preserves discontinuities. The error diminishes considerably: the mean error in depth diminishes from 0.1685m (standard deviation of 0.4483) to 0.0953m (standard deviation of 0.2397) when the regularisation term is used in the optimization problem. The RMSE, the maximum value, minimum value, mean, median and standard deviation for the slices and for the whole depth map are presented in tables 3.1 and 3.2, respectively.



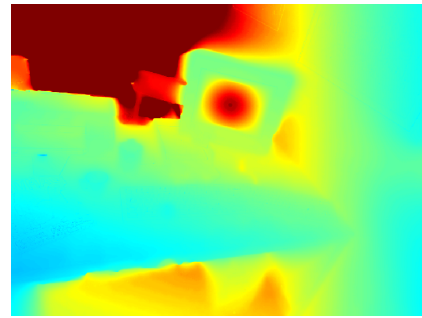
(a) Gray image from reference frame $I_r(\mu)$



(b) Ground truth of the depth map from reference frame $D_r(\mu)$

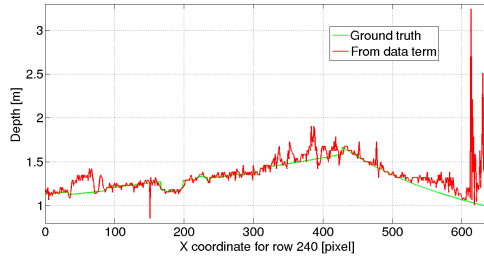


(c) Initial Depth map. It is noisy due to regions of low texture.

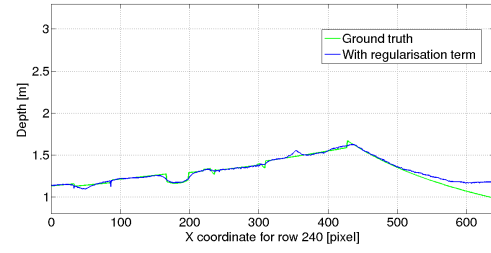


(d) Depth map after 200 iterations of primal-dual algorithm.

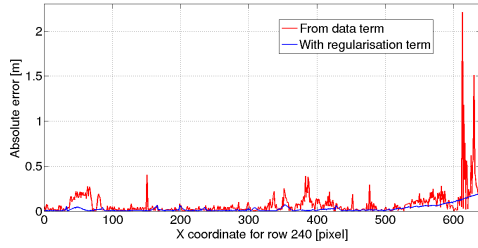
Figure 3.3: (a) Gray image from reference frame, (b) ground truth of the depth map from reference frame, (c) initial depth estimation and (d) solution in depth after 200 iterations of primal-dual algorithm. Red represents high depth while blue represents low depth.



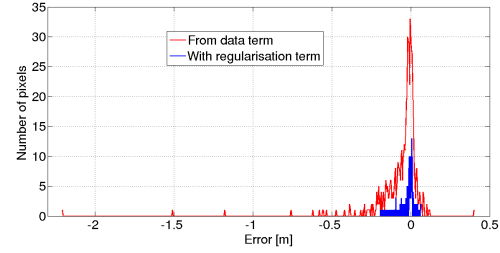
(a) Depth for x-slice, row 240, using just the data term (in red).



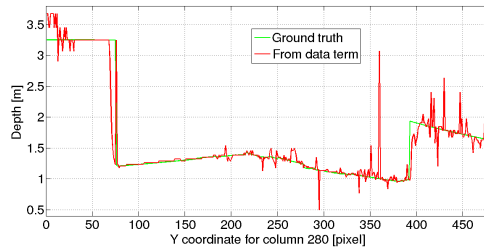
(b) Depth for x-slice, row 240, using the data term and the regularisation term (in blue).



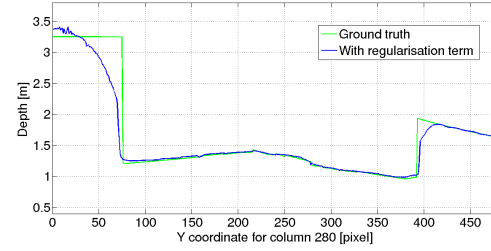
(c) Absolute error of the x-slice.



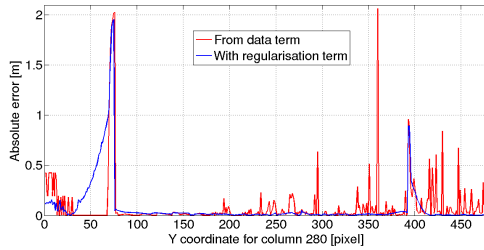
(d) Error distribution of x-slice.



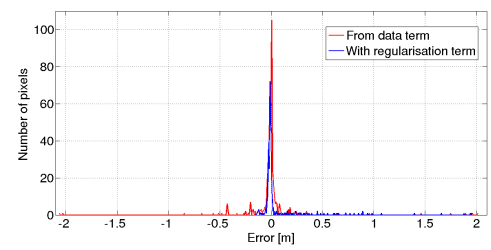
(e) Depth for y-slice, column 280, using just the data term (in red).



(f) Depth for y-slice, column 280, using the data term and the regularisation term (in blue).

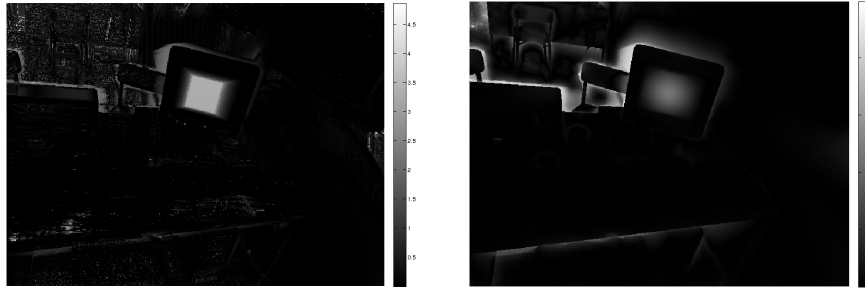


(g) Absolute error of the y-slice.



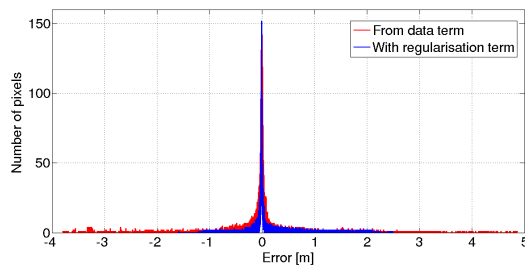
(h) Error distribution of y-slice.

Figure 3.4: Performance of the algorithm for an x-slice ((a)-(d)) and y-slice ((e)-(h)), when just the data term is used and when both the data term and the regularisation term are used.



(a) Absolute error in depth of the whole image using just the data term.

(b) Absolute error in depth of the whole image using both the data term and the regularisation term.



(c) Error distribution of the whole image.

Figure 3.5: Absolute error in depth [m] got (a) using just the data term and (b) using both the data term and the regularisation term (clear regions represent high errors). (c) Error distribution in both cases.

Table 3.1: Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for x and y slices.

Quantity \ Slice	x-slice DT	x-slice DT-RT	y-slice DT	y-slice DT-RT
RMSE [m]	0.1595	0.0466	0.2677	0.2463
Max. error [m]	2.2094	0.1946	2.0651	1.9547
Min. error [m]	4.2837e-05	5.5449e-05	4.8842e-05	5.8159e-05
Mean error [m]	0.0719	0.0265	0.0857	0.0795
Median error [m]	0.0310	0.0098	0.0147	0.0144
Standard dev. [m]	0.1425	0.0384	0.2539	0.2334

Table 3.2: Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for the whole image.

Quantity \ Depth map	whole image DT	whole image DT-RT
RMSE [m]	0.4789	0.2579
Max. error [m]	4.8614	2.4922
Min. error [m]	7.8578e-09	5.4907e-08
Mean error [m]	0.1685	0.0953
Median error [m]	0.0283	0.0174
Standard dev. [m]	0.4483	0.2397

3.5.2 RESULTS WITH REAL DATA

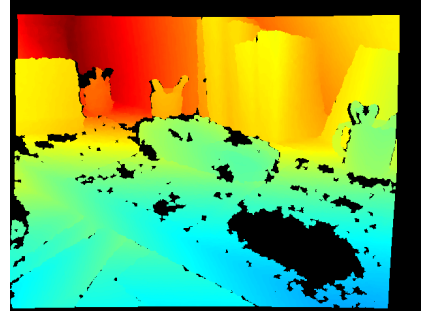
For the second experiment the Kinect 1.0 is employed: the RGB images for creating a depth map and the depth map obtained from the sensor as reference for estimating the accuracy of the solution. The camera pose is estimated with ORB-SLAM2 [59], set for working with RGBD data. We use the RGBD module instead of the monocular module of ORB-SLAM2 because it delivers the whole trajectory of the sensor (no just the pose of selected keyframes as with the monocular camera). This continuous trajectory is essential since the photometric error is computed for a set of images taken from a close view point, helping to fulfill the Lambertian assumption to some extent.

We use 40 images ($n = 40$) and one reference image. The range of depth is $[0.3 \ 1.5]\text{m}$ that corresponds to an inverse depth range of $[3.33 \ 0.66]\text{m}^{-1}$. The number of samples (linear sampling in inverse depth) is $n_s = 100$, the balance λ is 0.7, the threshold for the Huber norm ϵ is 0.01, α and β of eq. (3.11) are 0.4 and 2.4 respectively. Figures 3.6(a) and 3.6(b) show the image of reference in gray and the depth map from the sensor. Figures 3.6(c) and 3.6(d) show the initial depth map obtained by minimising eq. (3.7) and the refined depth map after 200 iterations, respectively. Figure 3.7(a) presents the depth data for row 210 using just the data term while fig. 3.7(b) shows the smoother version of data got with the regularisation term. Figure 3.7(c) and 3.7(d) show the absolute error and the error distribution for the solution when the regularisation term is included. The same analysis is done for data around column 320 (figs. 3.7(e)-3.7(h)).

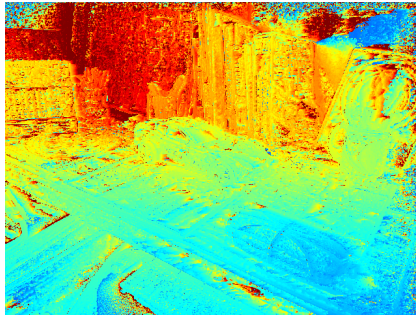
Figures 3.8(a) and 3.8(b) show the absolute error for the whole depth map using just the data term and both the data term and the regularisation term, respectively, while fig. 3.8(c) presents the error distribution. Note that the solution with regularisation term, like in the case of synthetic data, is smoother and preserves discontinuities. The mean error in depth diminishes from 0.0841m (standard deviation of 0.1161) to 0.0474m (standard deviation of 0.0964) when the regularisation term is used in the optimization problem. The RMSE, the maximum value, minimum value, mean, median and standard deviation for the slices and for the whole depth map are presented in tables 3.3 and 3.4, respectively.



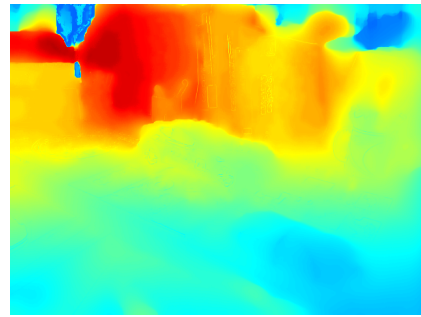
(a) Gray image for the reference frame $I_r(\mu)$, real data.



(b) Depth map from kinect $D_r(\mu)$. Used for estimating accuracy of solution.

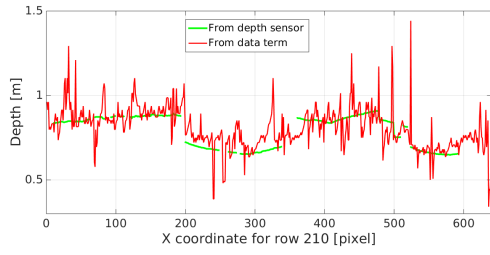


(c) Initial Depth map. It is noisy due to regions of low texture.

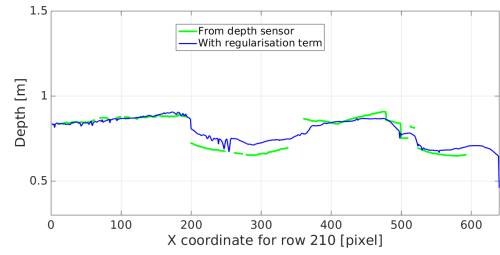


(d) Depth map after 200 iterations of primal-dual algorithm.

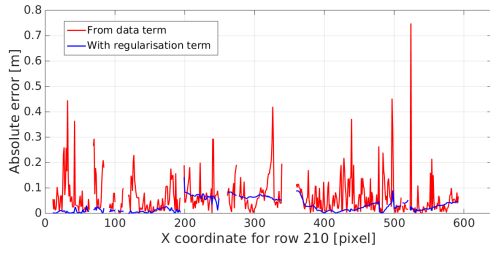
Figure 3.6: (a) Gray image from reference frame (real data), (b) depth map taken with kinect sensor for reference frame (used for estimating depth accuracy of solution), (c) initial depth map built using the photometric error and (d) refined depth map after 200 iterations of primal-dual algorithm. Red represents high depth while blue represents low depth.



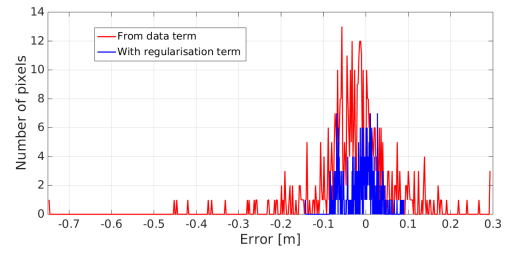
(a) Depth for x-slice, row 210, using just the data term (in red).



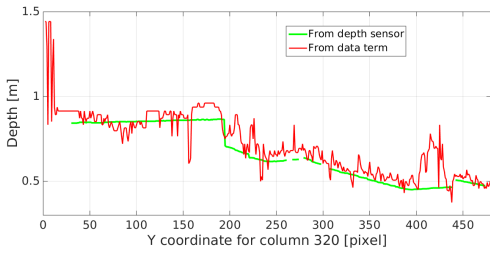
(b) Depth for x-slice, row 210, using the data term and the regularisation term (in blue).



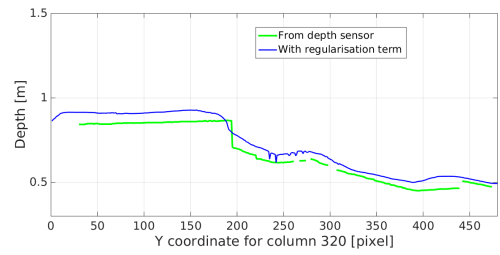
(c) Absolute error of the x-slice.



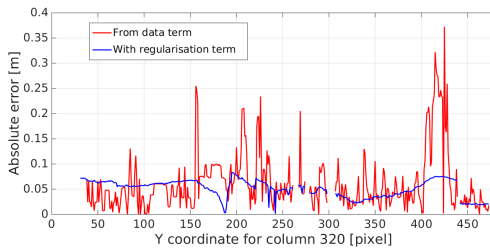
(d) Error distribution of x-slice.



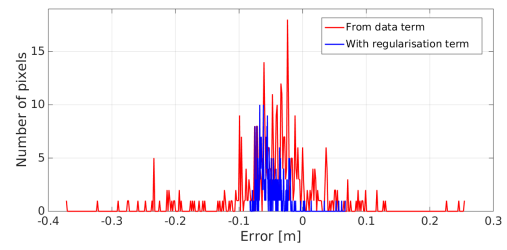
(e) Depth for y-slice, column 320, using just the data term (in red).



(f) Depth for y-slice, column 320, using the data term and the regularisation term (in blue).



(g) Absolute error of the y-slice.



(h) Error distribution of y-slice.

Figure 3.7: Performance of the algorithm for an x-slice ((a)-(d)) and y-slice ((e)-(h)), when just the data term is used and when both the data term and the regularisation term are used.

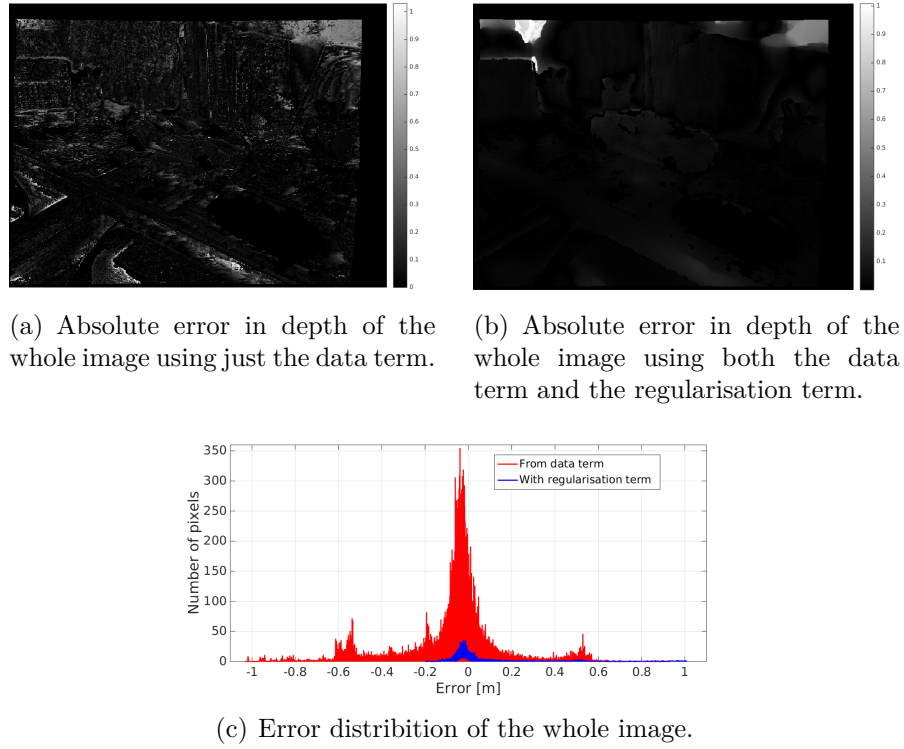


Figure 3.8: Absolute error in depth [m] got (a) using just the data term and (b) using both the data term and the regularisation term (clear regions represent high errors). (c) Error distribution in both cases.

Table 3.3: Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for x and y slices. The reference is the depth map from the sensor.

Quantity \ Slice	x-slice DT	x-slice DT-RT	y-slice DT	y-slice DT-RT
RMSE [m]	0.0987	0.0398	0.0842	0.0540
Max. error [m]	0.7483	0.1424	0.3723	0.0833
Min. error [m]	3.5714e-04	6.5296e-05	1.2987e-05	0.0013
Mean error [m]	0.0659	0.0304	0.0616	0.0512
Median error [m]	0.0444	0.0218	0.0464	0.0555
Standard dev. [m]	0.0736	0.0257	0.0575	0.0170

Table 3.4: Comparison of accuracy in depth using just the data term (DT) and both the data term and regularisation term (DT-RT), for the whole image. The reference is the depth map from the sensor.

Quantity \ Depth map	whole image DT	whole image DT-RT
RMSE [m]	0.1433	0.1074
Max. error [m]	1.0303	1.0089
Min. error [m]	0	2.4951e-07
Mean error [m]	0.0841	0.0474
Median error [m]	0.0449	0.0276
Standard dev. [m]	0.1161	0.0964

TIME

Our system for depth map building with a monocular camera runs on a laptop Hewlett Packard with a processor Intel Core i7-2.2GHz, 11.7GB of RAM memory, a graphic processor NVIDIA GEFORCE 755M and Ubuntu 14.04 as operating system. OpenCV is used for processing images, Eigen3 for matrix operations and Cuda 6.5 for speeding up the primal-dual algorithm. Table 3.5 shows the average on processing time for the main processes.

Table 3.5: Processing time for building a depth map with a monocular camera.

Process	Time[ms]
Creation of initial depth map. Eq. (3.7)	1062.5
Update of g . First line of eq. (3.22)	6.627
Update of y . Second line of eq. (3.22)	3.350
Update of z . Eq. (3.23)	6.293
Remaining processes	2.182
TOTAL ITERATION	18.452

The primal-dual algorithm takes 200 iterations for refining the initial depth map. Considering the time for building the initial depth map of 1062.5ms, the total time for building a refined depth map is 4752.9ms.

3.6 CONCLUSIONS

We have implemented an algorithm that builds depth maps by minimizing a data term and a regularisation term with a decoupling approach, alternating the primal-dual algorithm and an exhaustive search. In this chapter we could evidence two very important facts for defining the main contribution of this thesis. 1. When objects in the scene are very reflective, translucent or low-textured, or when the light conditions are uncontrolled, as occur in most real scenarios, the minimization of the data term, that depends on the average photometric error, is affected drastically, and therefore the estimated depth map is very noisy. 2. The regularisation term tackles this problem to some extent,

reducing the mean and standard deviation when it is used inside a variational approach, as was probed in both experiments: the mean error in depth diminishes from 0.1685m (standard deviation of 0.4483) to 0.0953m (standard deviation of 0.2397) and from 0.0841m (standard deviation of 0.1161) to 0.0474m (standard deviation of 0.0964) when the regularisation term is used in the optimization problem, for synthetic and real data, respectively.

Modifying the energy functional by varying the norm of the data term or the norm of the regularisation term in order to improve the performance of the algorithm when complex conditions occur has been probed in some works, like in [70] and [85]. However, the accuracy is improved to some extent. For getting a significant improvement in the accuracy of built depth maps, additional data must be integrated to the reconstruction process, as shape prior data that we will present in the next chapters.

Finally, the processing time in commodity graphics hardware of our algorithm is 4752.9ms which is appropriate for real time applications that build depth maps with a monocular camera for selected keyframes.

CHAPTER 4

SHAPE PRIOR ESTIMATION

4.1 INTRODUCTION

In this chapter we present a model-based approach that measures the discrepancy between a model hypothesis, represented as a 3D level-set embedding function, and observed depth data, defined by a point cloud obtained from a segmented region associated to the object of interest. This discrepancy defines an objective function, that is minimized with Levenberg-Marquardt w.r.t. pose, scale and shape.

The main pipeline of the system is shown in fig. 4.1. The set of reference models (predefined set of cars) are aligned using ICP (each one with respect to one base model) for getting models with the same scale, translation and rotation. A volumetric grid with TSDF values is computed and a compression is done using DCT, for maintaining just the low-frequency components of the reference models. A continuous, nonlinear, probabilistic and low-dimensional latent shape space that captures the prior knowledge on the 3D shapes that an object can take is found with Gaussian Process Latent Variable Models GPLVM. Figure 4.2 shows an illustrative description of the dimensionality reduction between the high dimensional space of the compressed 3D models and the space of the latent variables. Scaled conjugate gradient SCG is employed for learning the mapping between the latent variables and the low-frequency components (coefficients). Next, the latent variable that best fits to depth data is searched and the coefficients associated to it are estimated. The 3D level-set embedding function encoded in the coefficients is computed with the inverse discrete cosine transform IDCT.

We describe in detail the entire process and carry out experiments for optimizing pose, scale and shape independently (when two of the three data types are known) and alternately (when no one is known), with synthetic and real data, respectively. We can outline three main contributions: 1. The pose optimization is solved using Lie algebra $se(3)$ instead of the group of rigid transformations in 3D space $SE(3)$; 2. We propose an alternating method for

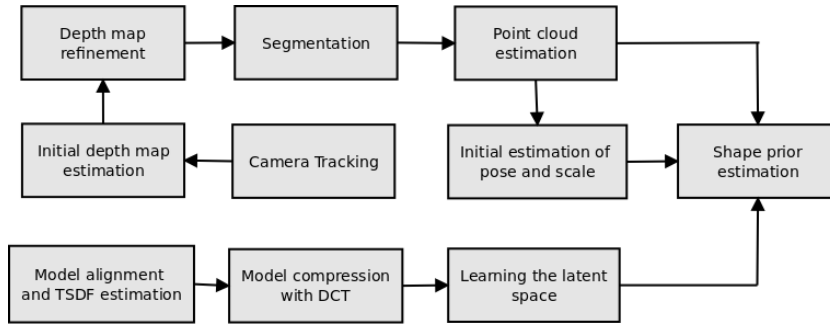


Figure 4.1: Main pipeline of the system for shape prior estimation using built depth maps.

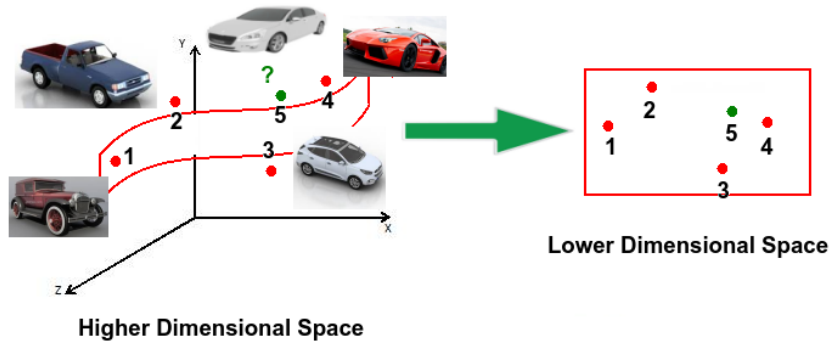


Figure 4.2: Illustrative description of the dimensionality reduction between the high dimensional space of the compressed 3D models and the space of the latent variables.

computing the optimal model. An initial estimation in object pose and scale, based on the centroid, the supporting plane and the average distance of the points to the centroid, is refined in alternating way. Once the pose and scale are refined, the initial shape is found with exhaustive search and posteriorly refined, alternating pose, scale and shape. 3. We quantify the accuracy of the pose, shape and scale optimization with synthetic data where the ground truth in these parameters is known.

This chapter is structured as follows. In section 4.2 similar works about shape priors are presented. In sections 4.3, 4.4, 4.5, 4.6 and 4.7 we explain the modules for model alignment and truncated signed distance function TSDF estimation, model compression, dimensionality reduction, search of the optimal latent variable, and estimation of the optimal pose and scale, respectively. In section 4.8 we show the results and finally, in section 4.9 we present the conclusions.

4.2 RELATED WORK

The system [3] learns a semantic prior comprised of a mean shape for a category (common aspects in shape of a category) and a set of weighted anchor points for instances of the category (specific details). The experiments show that holes and outliers are eliminated and details are recovered, even for multiple cars in a cluttered scene. The system [73] is the first one that back projects depth images and evaluates the resulting point cloud in a 3D level-set embedding function that represents an object model implicitly, instead of projecting the model to the image plane and measuring the discrepancy between expected image cues and the observed ones. Moreover, no point correspondence is required, unlike Iterative Closest Point ICP, since the alignment consist in evaluating the closeness of the points to the zero-level of the embedding function. The authors in [49] remove point cloud artifacts like noisy points, missing data and outliers using a learned shape prior of an object of interest. They use the discrete cosine transform DCT for compressing the SDF values and GPLVM for dimensionality reduction.

The monocular system [72] uses DCT for compressing the 3D level-set embedding functions and GPLVM for nonlinear dimensionality reduction to capture shape variance. The energy function measures the discrepancy between the projected 3D model into the image plane and the probabilistic 2D occupancy map that defines the foreground of the observed object in the image (image-based energy). The minimization is done w.r.t. pose and shape of the 3D model. A 2D segmentation results automatically after convergence. The system [13] combines shape priors and live dense reconstruction using a monocular camera. Photo-consistency and a variational approach are used for building depth maps [62] that are fused for creating a dense reconstruction, PTAM [44] for tracking the camera, a part-based object detector [28], and GPLVM to represent shape priors. The energy function, besides depending on the evaluation of the 3D points on the embedding function like [73] and [49], depends on the matching between the 2D object segmentation, defined by the foreground and background, and the projected 3D SDF (depth and intensity-based energy).

The system [69] estimates depth maps using a monocular camera in workspaces with large plain structures like floors, walls or ceilings. Good depth data is propagated to an interior pixel (inpainting) from the closest valid pixels along the main 8 star directions by using a non-local higher-order regularisation term, in a variational approach, that favours solutions with affine surfaces (prior). The system [11] includes a term, besides the data term and regularisation term, that depends on three scene priors: planarity of homogeneous color regions (using superpixels), the repeating geometry primitives of the scene (data-driven 3D primitives learned from RGBD data), and the Manhattan structure of indoor rooms (layout estimation and classification of box-like

structures).

Our system is not based on scene shape priors like [69], [11] but on object shape priors like [73], [49], [3], [72] and [13]. However, it does not use neither anchor points like [3], nor intensity-based energy function like [72], nor a depth and intensity-based energy function like [13], but GPLVM like [49], [13], [72] and a depth-based energy function like [73] and [49]. We use a variational technique for integrating object shape prior data in the solution, like is done in [69] and [11] for scene shape priors. This integration will be explained in detail in the next chapter.

4.3 ALIGNMENT AND TSDF ESTIMATION OF THE 3D MODELS

We downloaded 49 models of cars from Turbosquid [100], a vast online catalog of 3D models. We aligned each model with respect to a base model (chosen arbitrarily) using just the vertices of the triangular mesh, read from an .obj file. Initially, the point cloud of the base car is translated from their center of gravity to the origin. Then, it is scaled for fitting it to the volumetric grid, and the average distance of each point to the new origin is computed (see [6] for more details),

$$s_{\text{base}} = \frac{\sum_{i=1}^{N_{\text{base}}} \sqrt{X_i^2 + Y_i^2 + Z_i^2}}{N_{\text{base}}} \quad (4.1)$$

where N_{base} is the number of points of the base model. The same process is carried out (translation and average distance computation) for the point cloud of the model i for getting s_i . The scale is computed as,

$$s = \frac{s_{\text{base}}}{s_i} \quad (4.2)$$

This scale is applied to model i in order to get models of the same size. Then, the models are manually oriented in such a way that their frontal side point to the same direction, standing over a parallel supporting plane. This initial alignment is refined using Iterative Closest Point, implemented in CloudCompare [33].

Once the models have been aligned, the model i is loaded in OpenGL using the vertices and facets data (just geometry data), creating a continuous surface. The virtual camera is moved in such a way that circular and vertical scanning at different longitudes (changes in azimuth and elevation of 10°) are performed. The optical axis of the virtual camera is always pointing to the

origin that coincides with the center of gravity of the loaded model. Depth images, obtained by reading the depth buffer from OpenGL, are fused into a volumetric structure, storing a truncated signed distance function TSDF value in each voxel (see chapter 6 for details about fusion). It is a truncated version of the SDF, used for representing surface interfaces as zero, free space as positive values that increases with the distance to the closest surface and (possible) occupied space as a negative value. Figure 4.3 shows the alignment and the resulting model after applying marching cubes to the volumetric grid.

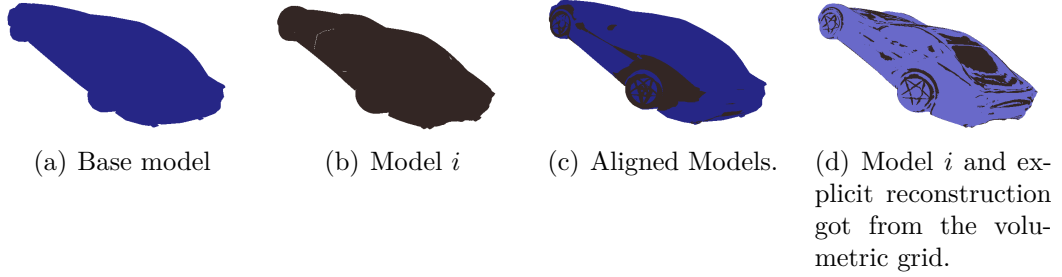


Figure 4.3: Alignment and TSDF estimation. (a) Base model. (b) Model i . (c) Aligned base model and model i . (d) Model i and its corresponding explicit representation got by applying marching cubes to the TSDF values of the volumetric grid.

We carry out this process for $N_m = 49$ 3D models of cars with different shapes (see 12 of the 49 models in fig. 4.4), obtaining 3D level-set embedding functions defined by a volumetric grid of $N_g \times N_g \times N_g$ voxels, with $N_g = 128$. The matrix $M \in \mathbb{R}^{N_m \times N_g^3}$ stores these models as row vectors. However, searching for a model that best fits to data in this high-dimensional space is too complex, so we compress the data using DCT. Then, we find a low-dimensional space and the mapping parameters for passing from this latent space to the space of DCT coefficients.

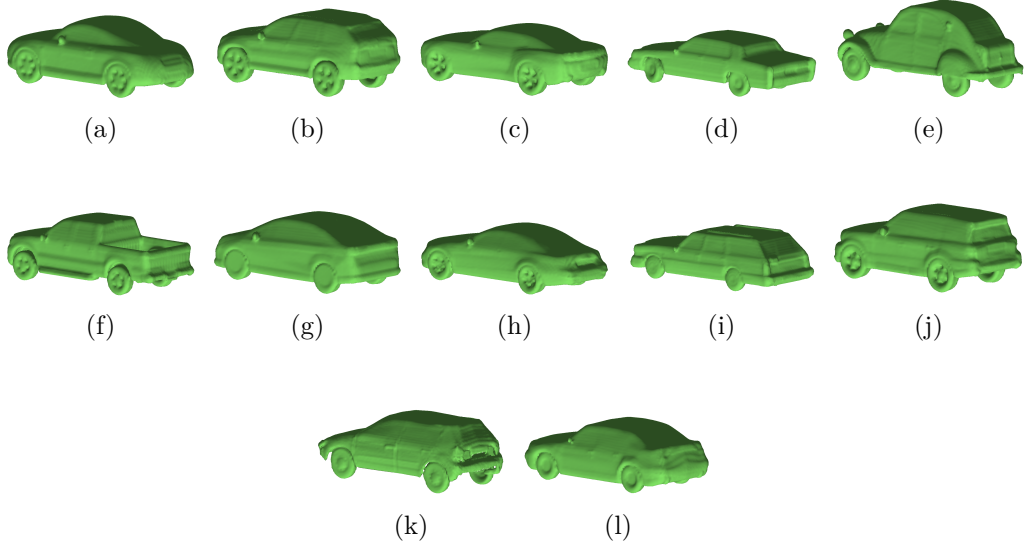


Figure 4.4: 3D models of cars with different shapes.

4.4 COMPRESSION OF THE EMBEDDING FUNCTION

The Discrete Cosine Transform (DCT) transforms a time domain signal to a frequency domain signal. It is used for compression of audio (e.g. MP3) and images (e.g. JPEG) where small high-frequency components can be discarded. We use here this technique for compressing the TSDF values of the volumetric grid. Let $\phi(u, v, w) \in \mathbb{R}^{N_g \times N_g \times N_g}$ be the volumetric grid that stores the TSDF values of a 3D model. The two-dimensional discrete cosine transform for slices ϕ_i in direction u of ϕ , it means, $c_i(v, w) = DCT(\phi_i(v, w))$ is given by

$$c_i(v, w) = \alpha(v, w) \sum_{y=0}^{N_g-1} \cos\left(\frac{\pi(2y+1)v}{2N_g}\right) \sum_{z=0}^{N_g-1} \phi_i(y, z) \cos\left(\frac{\pi(2z+1)w}{2N_g}\right) \quad (4.3)$$

where $\alpha(v, w) = \alpha_v(v) * \alpha_w(w)$, with

$$\alpha_v(v) = \alpha_w(w) = \begin{cases} \frac{1}{\sqrt{N_g}} & \text{if } v, w = 0 \\ \frac{2}{\sqrt{N_g}} & \text{if } v, w \neq 0 \end{cases} \quad (4.4)$$

Since we have $N_g = 128$ slices, the size of ϕ_i is (128×128) . Equation (4.3) can be expressed as matrix and array multiplications

$$c_i = \alpha. * (A\phi_i A^T). \quad (4.5)$$

where $A \in \mathbb{R}^{N_c \times N_g}$ with N_c the number of DCT coefficients. Each row of A stores the cosine $\cos\left(\frac{\pi(2x+1)u}{2N_g}\right)$ for a common u , with $u = 1, 2, \dots, N_c$. We extend it to the three-dimensional discrete cosine transform and apply the property of separability (see [42] for more details), computing the 3D-DCT in two steps with 2D transformations over the slices for each i along dimension u and a final 1D operation. Figure 4.5 shows the 3D models obtained using 10, 25, 35, 50, 75 and 100 coefficients of the model visualized in 4.4(b). Here we can see that the information about the general shape is stored in the low-frequency components and the fine details are stored in the high-frequency components. Note that for 50, 75 and 100 coefficients there are not significant changes in appearance. We consider that $25 \times 25 \times 25$ coefficients are enough for our purposes (like in [49]).

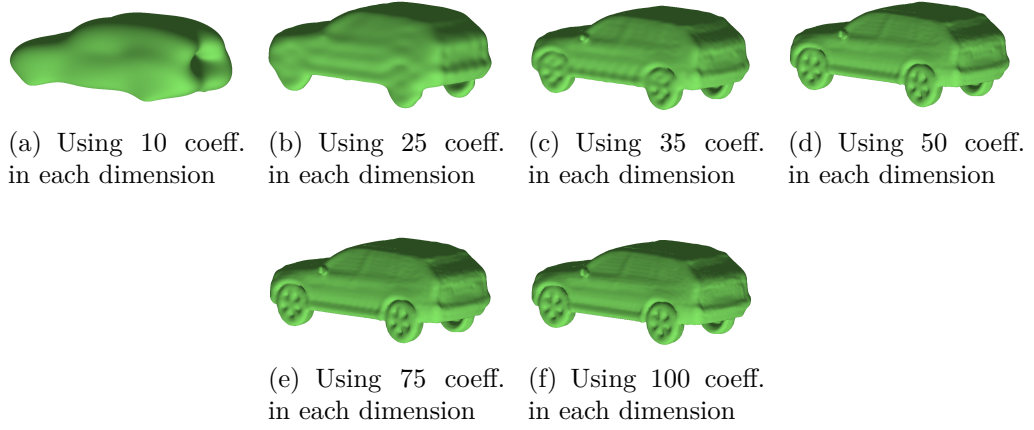


Figure 4.5: 3D model of fig. 4.4(b) using different number of coefficients in each dimension (degree of compression).

4.5 DIMENSIONALITY REDUCTION

The Latent variable Model LVM is used for dimensionality reduction, to capture the shape variance as low-dimensional latent shape spaces, such that the resulting latent variables have less dimensions than the original observed data. Let $Y \in \mathbb{R}^{N_m \times N_c^3}$ be the matrix that stores in row $\mathbf{Y}_j \in \mathbb{R}^{1 \times N_c^3}$ the $N_c = 25$ low-frequency coefficients in each dimension of the volumetric grid, and for each model j , with $j = 1 : 1 : N_m$. Let $\chi \in \mathbb{R}^{N_m \times N_x}$, with $N_c^3 \gg N_x$, be the matrix that stores in each row the latent variable $\chi_j \in \mathbb{R}^{1 \times N_x}$ that represents the coefficients of model j . Now, we learn from the coefficients the parameters $(\theta_1, \theta_2, \theta_3, \theta_4)$ that map any latent variable χ_j to the corresponding row vector of coefficients $\mathbf{Y}_j$, where its relationship is non-linear. Since the mapping $\chi_j \rightarrow \mathbf{Y}_j$ is modelled using a Gaussian process, we can define areas of high probability, it means, areas where there are high certainty of getting a valid shape. Moreover, this technique allows to map coefficients

from a continuous latent space (not just the ones used for learning), creating a continuous search of the latent variable that best fits to data. We define the vector $\mathbf{W} \in R^{1 \times (N_m * N_x + 4)}$ as

$$\mathbf{W} = \{\chi_{1,1} \quad \dots \quad \chi_{1,N_m} \quad \dots \quad \chi_{N_x,1} \quad \dots \quad \chi_{N_x,N_m} \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4\} \quad (4.6)$$

It is the vector of parameters that solves the following optimization problem

$$\min_{\mathbf{W}} E(\mathbf{W}) = \frac{N_m}{2} \ln(2\pi) + \frac{1}{2} \ln|K| + \frac{1}{2} \text{tr}(K^{-1}\mathcal{S}) \quad (4.7)$$

where N_m is the number of models, K is the covariance matrix which is estimated with a Kernel function, and $\mathcal{S} = \mathcal{D}^{-1}YY^T$, with $\mathcal{D}$ the number of coefficients for a model (N_c^3). The components of K , denoted as $k_{ij} = K(\chi_i, \chi_j)$, for $N_x = 2$ and a Radial Basis Function kernel, are

$$k_{ij} = \theta_1 \exp\left(\frac{-\theta_2}{2}((\chi_{1,i} - \chi_{1,j})^2 + (\chi_{2,i} - \chi_{2,j})^2)\right) + \theta_3 + \delta_{ij}\theta_4 \quad (4.8)$$

where δ_{ij} is the Kronecker delta function. We initialize the latent variables of the vector of parameters $\mathbf{W}$ with the estimation got with Dual Probabilistic PCA. Then, we use the scaled conjugate gradient SCG algorithm for refining the initial estimation. It combines the model trust region approach, known from the Levenberg-Marquardt algorithm, with the conjugate gradient approach. The pseudocode and more details about SCG can be found in [55]. The parameters for the kernel are set according to [50]; $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = \exp(-1)$, and $\theta_4 = \exp(-1)$. With these values we can set the initial kernel and continue the iterative process until convergence, as is shown in fig. 4.6(a). The resulting latent space and the corresponding latent variables are shown in fig. 4.6(b). Note that besides the mean value, each point χ_p in the latent space has a variance which is computed as

$$\sigma^2 = K(\chi_p, \chi_p) - K(\chi, \chi_p)K^{-1}K(\chi, \chi_p)^T, \quad (4.9)$$

The parameters for mapping converged to $\theta_1 = 10.4195$, $\theta_2 = 1.6784$, $\theta_3 = 84.5378$, and $\theta_4 = 1.1679$.

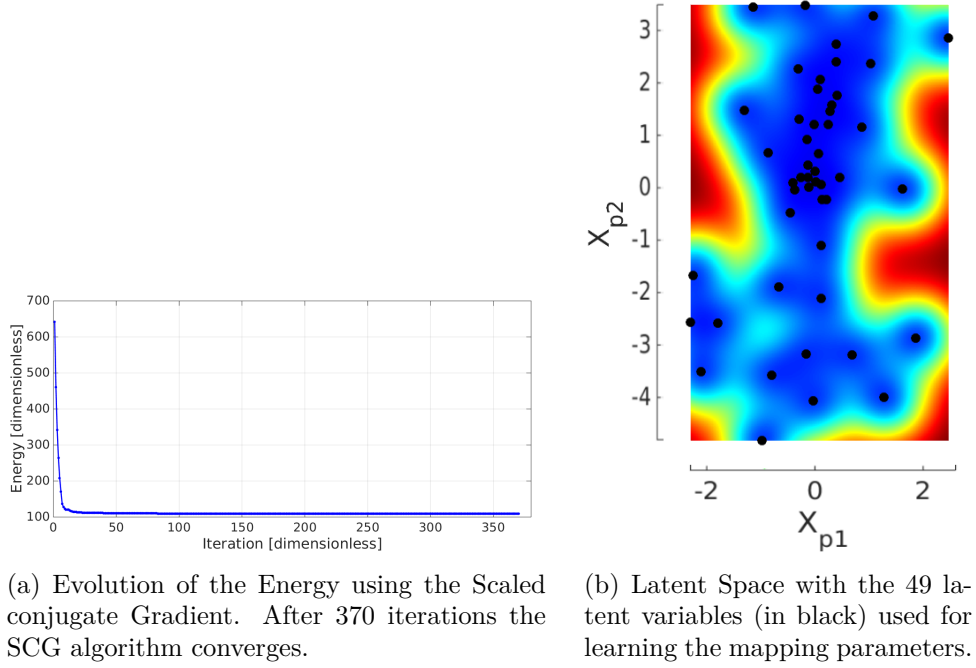


Figure 4.6: (a) Energy and (b) latent space for embedding functions of cars. Low variance regions are drawn in blue while high variance regions are drawn in red. More likely valid shapes are obtained in blue areas.

4.6 SHAPE OPTIMIZATION

We define an energy that depends on the sum of squared residuals,

$$E(\Phi) = \frac{1}{2} \sum (\Phi)^2 \quad (4.10)$$

where the residual vector Φ is defined with the German-McClure function, which is robust to outliers:

$$\Phi = \frac{\phi(\mathbf{X}_q)^2}{\phi(\mathbf{X}_q)^2 + \sigma}. \quad (4.11)$$

where σ is a constant parameter, $\mathbf{X}_q = [X_q \ Y_q \ Z_q]$ is one of the N_p 3D points that are evaluated (3D interpolation) in the embedding function ϕ . Points that are back-projected outside the volumetric grid are assigned a large value. The derivative of the energy w.r.t. the latent variable χ_p is computed through the chain rule,

$$\frac{\delta E}{\delta \chi_p} = \frac{\delta E}{\delta \Phi} * \frac{\delta \Phi}{\delta \chi_p} \quad (4.12)$$

The first term on the right is the residual vector Φ . The variations of the residual vector Φ due to changes in the latent variable χ_p corresponds to the Jacobian,

$$J = \frac{\delta \Phi}{\delta \chi_p} = \frac{\delta \Phi}{\delta \phi} * \frac{\delta \phi}{\delta \chi_p} \quad (4.13)$$

where

$$\frac{\delta \Phi}{\delta \phi} = \frac{2\sigma\phi(\mathbf{X}_q)}{(\phi(\mathbf{X}_q)^2 + \sigma)^2} \quad (4.14)$$

For defining the second term on the right of eq. (4.13) we express the embedding function ϕ in terms of the latent variable,

$$\phi = \text{IDCT}(\nu(\chi_p)) \quad (4.15)$$

The function $\text{IDCT}(\cdot)$ is the inverse discrete cosine transform applied to the mean ν of the coefficients associated with the latent variable χ_p through a Gaussian process,

$$\nu(\chi_p) = K(\chi, \chi_p)K^{-1}Y \quad (4.16)$$

Considering that the derivative of the IDCT of a variable is the IDCT of the derivative of the variable, we have

$$\frac{\delta \phi}{\delta \chi_p} = \frac{\delta(\text{IDCT}(\nu))}{\delta \chi_p} = \text{IDCT}\left(\frac{\delta \nu}{\delta \chi_p}\right) \quad (4.17)$$

where

$$\frac{\delta \nu}{\delta \chi_p} = \frac{\delta K(\chi, \chi_p)}{\delta \chi_p} K^{-1}Y \quad (4.18)$$

$K(\chi, \chi_p) \in R^{1 \times N_m}$ is the row kernel for the latent variable χ_p . It depends on the squared euclidean distance between the latent variable χ_p and each one of the N_m latent variables used for learning the parameters for mapping. Each element of the row kernel for χ_p , with $N_x = 2$, is found using eq. (4.8), replacing χ_j by χ_p . The derivative of $K(\chi, \chi_p)$ w.r.t each component of the latent variable is computed in a straight way. The latent variable is updated by addition, $\chi_p^{k+1} = \delta \chi_p + \chi_p^k$, where $\delta \chi_p$ is found with Levenberg-Marquardt,

$$\delta \chi_p = -(J^T J + \alpha \text{diag}(J^T J)I)(J^T \Phi). \quad (4.19)$$

We start with a small initial value of α , $\alpha = 1e - 4$. After each iteration, the new energy is compared with the previous one. If the energy has decreased,

α is reduced by a factor of 10 and the parameters are updated. If the energy has increased, the parameters are not updated and α is increased by a factor of 10.

4.7 POSE AND SCALE OPTIMIZATION

We now differentiate the energy of eq. (4.10) w.r.t. the pose and scale, optimizing them in separated and alternating way. The pose is minimally parametrized with a vector $\Upsilon = [w_x, w_y, w_z, v_x, v_y, v_z]$, where the first three elements define the axis of rotation and its norm defines the magnitude of the rotation. The remaining elements define the translation after carrying out the rotation. The derivative of the energy w.r.t. Υ is

$$\frac{\delta E}{\delta \Upsilon} = \frac{\delta E}{\delta \Phi} * \frac{\delta \Phi}{\delta \Upsilon} \quad (4.20)$$

Again, the first term on the right is the residual Φ . The variations of the residual vector Φ due to changes in pose Υ correspond to the Jacobian for this problem,

$$J = \frac{\delta \Phi}{\delta \Upsilon} = \frac{\delta \Phi}{\delta \phi} * \frac{\delta \phi}{\delta \Upsilon} \quad (4.21)$$

The first term on the right was defined in equation (4.14). The second term on the right is

$$\frac{\delta \phi}{\delta \Upsilon} = \frac{\delta \phi}{\delta \mathbf{X}_q} * \frac{\delta \mathbf{X}_q}{\delta \Upsilon} \quad (4.22)$$

$\frac{\delta \phi}{\delta \mathbf{X}_q}$ is computed numerically, using the central difference formula,

$$\frac{\delta f(x)}{\delta x} = \frac{f(x+h) - f(x-h)}{2h} \quad (4.23)$$

Since a point $\mathbf{X}_q = [X_q \ Y_q \ Z_q]$ (coordinates of voxels referenced to the initial coordinate system q), we get,

$$\frac{\delta \phi}{\delta \mathbf{X}_q} = \begin{bmatrix} \frac{\delta \phi}{\delta X_q} & \frac{\delta \phi}{\delta Y_q} & \frac{\delta \phi}{\delta Z_q} & 1 \end{bmatrix} \quad (4.24)$$

with $h = 10^{-8}$ for all the components. For defining $\frac{\delta \mathbf{X}_q}{\delta \Upsilon}$ we first express a point $\mathbf{X}_q$ in terms of $\mathbf{X}_w$. We have that

$$\mathbf{X}_w = M_{wo} S M_{oq} \mathbf{X}_q \quad (4.25)$$

M_{wo} is the transformation matrix between the object and the world system. M_{oq} is the transformation matrix between the initial and the object system. It corresponds to a translation to the center of the volumetric grid. Figure 4.7 shows the initial coordinate system, the object system, and the world system. Since the point cloud is referenced to the world coordinate system, the 3D model is also referenced to this system and then, aligned with the point cloud.

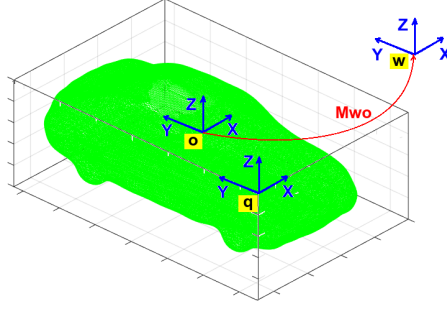


Figure 4.7: Initial coordinate system q , object coordinate system o and world coordinate system w .

S scales the points $\mathbf{X}_o$ and is defined as

$$S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.26)$$

Solving for the point $\mathbf{X}_q$, we get

$$\mathbf{X}_q = (M_{oq})^{-1} S^{-1} M_{ow} \mathbf{X}_w \quad (4.27)$$

The transformation matrix M_{ow} is updated as

$$M_{ow}^{k+1} = \Delta M_{ow} M_{ow}^k \quad (4.28)$$

where the incremental change in the transformation is expressed using the exponential mapping and the Lie algebra, $\Delta M_{ow} = \exp([\Delta \Upsilon]_x)$. The operator $[\cdot]_x$ transforms a six-elements vector to the skew matrix

$$[\Delta \Upsilon]_x = \begin{bmatrix} 0 & -w_z & w_y & v_x \\ w_z & 0 & -w_x & v_y \\ -w_y & w_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.29)$$

$\Delta \Upsilon$ is computed in similar way as $\delta \chi_p$ in eq. (4.19). Replacing M_{ow}^{k+1} of eq. (4.28) and ΔM_{ow} into eq. (4.27), we get

$$\mathbf{X}_q = (M_{oq})^{-1} S^{-1} \exp([\Delta \Upsilon]_x) M_{ow}^k \mathbf{X}_w \quad (4.30)$$

$M_{ow}^k \mathbf{X}_w$ is the 3D point $\mathbf{X}_o^k$ previous to update the transformation matrix M_{ow} . Now, $\frac{\delta \mathbf{X}_q}{\delta \Upsilon}$, considering the scale constant, is

$$\frac{\delta \mathbf{X}_q}{\delta \Upsilon} = M_{oq}^{-1} S^{-1} \frac{\delta(\exp([\delta \Upsilon]_x))}{\delta \Upsilon} \mathbf{X}_o^k \quad (4.31)$$

where the derivative of the exponential w.r.t the first element of the vector Υ is

$$\frac{\delta(\exp([\delta \Upsilon]_x))}{\delta w_x} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.32)$$

This derivative is a generator for rotations in x-axis in the Lie algebra. The same process is carried out for the remaining five parameters. Multiplying by the homogeneous form of $\mathbf{X}_o$ and organizing, we get a 4×6 matrix.

$$\frac{\delta(\exp([\delta \Upsilon]_x))}{\delta \Upsilon} \mathbf{X}_o^k = \begin{bmatrix} 0 & X_{oz} & -X_{oy} & 1 & 0 & 0 \\ -X_{oz} & 0 & X_{ox} & 0 & 1 & 0 \\ X_{oy} & -X_{ox} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.33)$$

Following a similar process for scale optimization, we consider the parameters of pose constant and derive $\mathbf{X}_q$ from eq. (4.27) w.r.t the scale parameter.

$$\frac{\delta \mathbf{X}_q}{\delta S} = -M_{oq}^{-1} S^{-1} \frac{\delta S}{\delta S} S^{-1} \mathbf{X}_o^k \quad (4.34)$$

where $\frac{\delta S}{\delta s}$ is easily computed.

4.8 RESULTS

For testing our optimization algorithms when one of the three variables is unknown (shape, pose or scale) and when the three variables are unknown, we use a down-sampled point cloud (fig. 4.8(a)) coming from the vertices of the triangles computed with the marching cubes algorithm for the compressed version of the base model (fig. 4.8(b)). Moreover, we will use real data: 1. depth maps coming from the Kinect 1.0 and 2. depth maps built with a monocular camera.

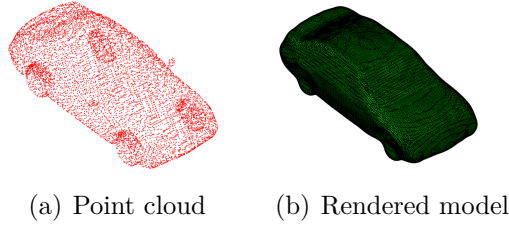


Figure 4.8: Reference model. (a) Downsampled 3D point cloud. (b) Rendered model (ground truth in shape).

4.8.1 RESULTS IN SHAPE OPTIMIZATION WITH SYNTHETIC DATA

The goal is to refine the shape associated to an initial latent variable in order to reduce the residual computed by evaluating the current embedding function in the point cloud (eq. (4.10)). In this experiment, the pose and scale are considered known. Figure 4.9(a) shows an explicit representation of the model for the initial latent variable. Figures 4.9(b), 4.9(c), and 4.9(d) show the model for iterations 1, 2, and 8, respectively. Figure 4.10(a) shows the evolution of the energy through time. Figure 4.10(b) shows the path of the iterative search in the latent space. It starts with a latent variable $\chi = [-0.2723 \quad 2.3048]$ and in iteration 15 χ has evolved to $[0.500114 \quad 1.63515]$, with a ground truth of $\chi = [0.3895 \quad 1.6162]$. Note that after iteration 8 ($\chi = [0.582774 \quad 1.62765]$) the energy diminishes slowly. The Euclidean distance between the estimated latent variable and the ground truth decreased from 0.9551 in the initialization to 0.1122 in iteration 15, getting a reduction in error of 88.25%.

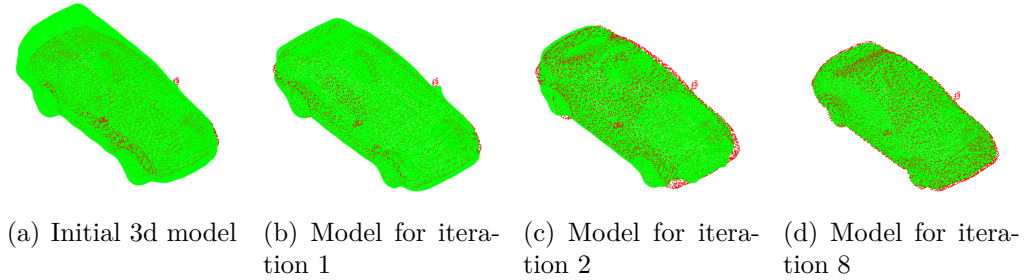


Figure 4.9: Evolution of the shape in the optimization process. The search is done in the latent space and the model is obtained with the inverse DCT. The point cloud is drawn in red.

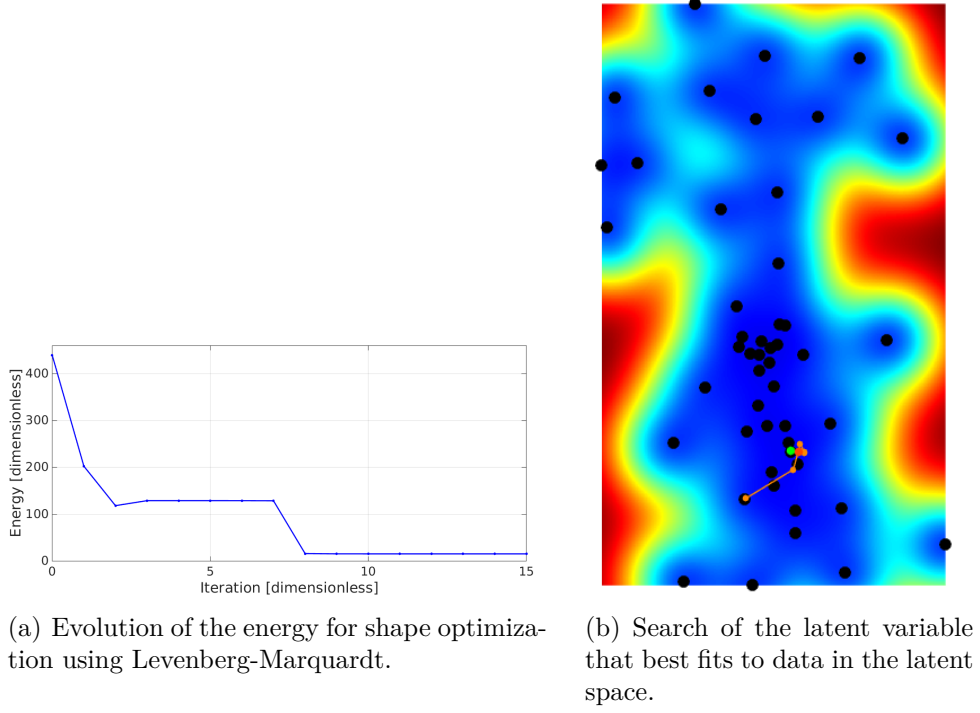


Figure 4.10: (a) Evolution of the energy. (b) Search of the latent variable. The initial latent variable and for iterations from 1 to 14 are drawn in clear orange. The latent variable for iteration 15 is drawn in dark orange. The ground truth in latent variable is drawn in green.

4.8.2 RESULTS IN POSE OPTIMIZATION WITH SYNTHETIC DATA

In this experiment the shape and scale are considered known and an initial pose is refined in order to align the embedding function with the point cloud. The ground truth in position is the origin and in orientation is the identity matrix. The initial position is $\mathbf{t}_{wo} = [-0.2 \ 0.2 \ 0.2]$ with an Euclidean distance to the origin of 0.3464. The initial orientation corresponds to a rotation of $\alpha_z = -30$ around z-axis w.r.t the ground truth. Figure 4.11(a) shows the initial pose, while figs. 4.11(b), 4.11(c) and 4.11(d) show the pose for iterations 20, 40 and 50, respectively. Figure 4.12 shows the energy for pose optimization. Note that it converges at iteration 59. The evolution of the Euclidean distance from the model to the origin and the evolution of the model orientation w.r.t the world coordinate system (ground truth) are presented in fig. 4.13(a) and 4.13(b), respectively. The orientation R_{wo} is represented with rotations over the fixed axis x , y , and z (in this order). The final position is $\mathbf{t}_{wo} = [0.0032 \ -0.0104 \ 0.0035]$, with an Euclidean distance to the origin of 0.0114. The final orientation is $\alpha_x = -0.0017^\circ$, $\alpha_y = -0.0663^\circ$, and $\alpha_z = -0.0209^\circ$. The reduction in position error is 96.70% and in rotation around z-axis is 99.93%.

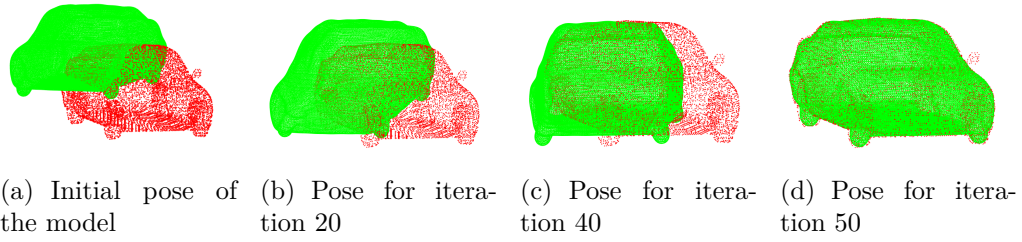


Figure 4.11: Evolution of the pose of the embedding function in the optimization process. The point cloud is drawn in red.

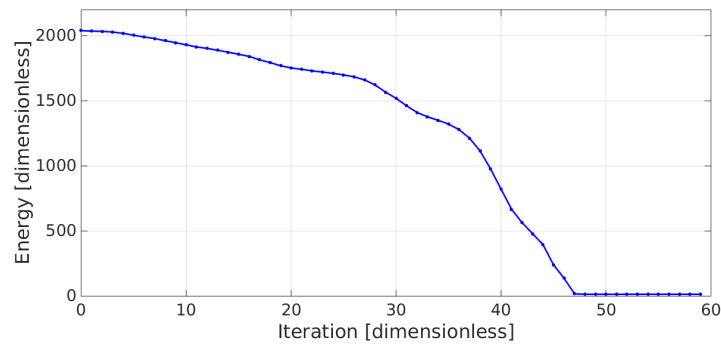


Figure 4.12: Evolution of the energy for pose optimization using Levenberg-Marquardt.

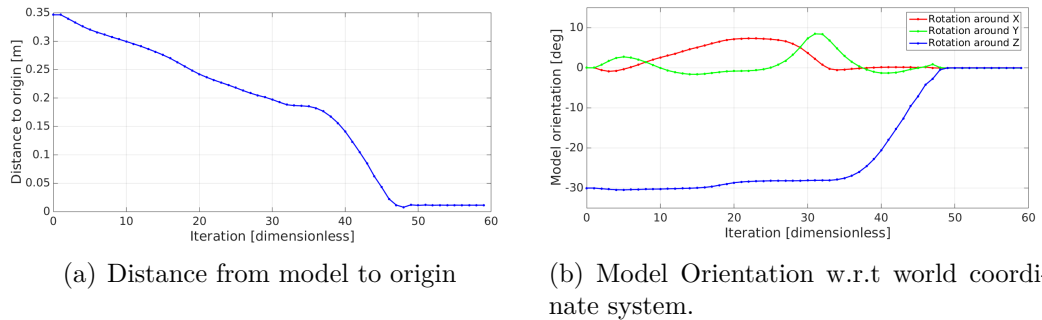


Figure 4.13: Evolution of the position and orientation with respect to the ground truth.

4.8.3 RESULTS IN SCALE OPTIMIZATION WITH SYNTHETIC DATA

For scale optimization, the shape and pose are considered known. Recall that the same scale is used in the three dimensions. The initial value in scale is $s = 0.4$. The ground truth in scale is $s = 1.0$. Figure 4.14(a) shows the initial model scale ($s = 0.4$), meanwhile figs. 4.14(b) ($s = 1.5855$), 4.14(c) ($s = 1.1204$), and 4.14(d) ($s = 1.0053$) show the model scale for iterations

1, 2, and 10, respectively. Note that the algorithm takes 10 iterations for converging. The final error is 0.53% of the ground truth. The reduction in error is 99.12%. Figure 4.15(a) depicts the evolution for the energy and fig. 4.15(b) shows how the scale gets close to the ground truth ($s = 1.0$).

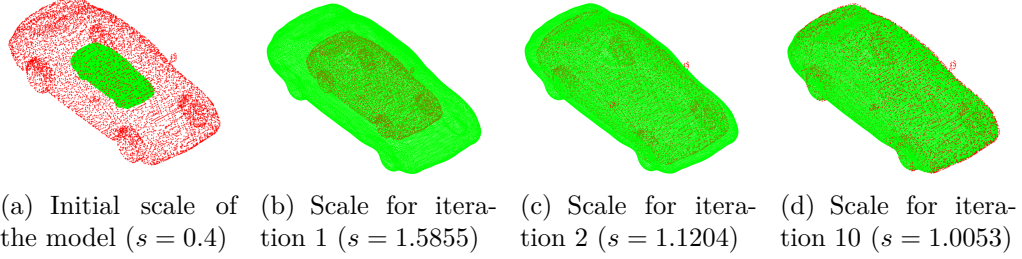


Figure 4.14: Evolution of the scale of the embedding function in the optimization process. The point cloud is drawn in red.

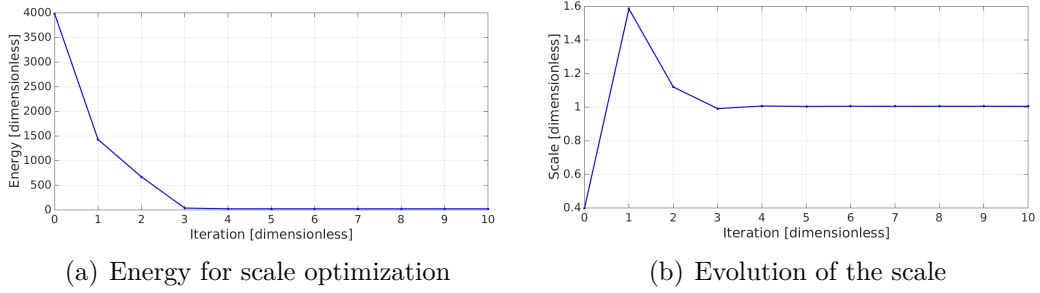


Figure 4.15: (a) Evolution of the energy and (b) the scale of the embedding function in the optimization process. The ground truth in scale is $s = 1$.

4.8.4 RESULTS IN POSE, SCALE AND SHAPE WITH SYNTHETIC DATA

The main optimization problem occurs when the pose, scale and shape are unknown, just an approximation is available. In this case, we alternate the algorithms for optimizing pose, scale, and shape. The ground truth in position is the origin, in orientation is the identity matrix, in scale is $s = 1$, and in shape is $\chi = [0.3895 \quad 1.6162]$. The initial position is $\mathbf{t}_{wo} = [-0.2 \quad 0.2 \quad 0.1]$, with an Euclidean distance to the origin of 0.3. The initial orientation corresponds to a rotation of $\alpha_z = -20^\circ$ around z-axis w.r.t the ground truth. The initial scale is $s = 0.8$, and the initial shape is $\chi = [-0.0828 \quad 0.9363]$. Figures 4.16(a) and 4.16(b) show the initial conditions for the model. For this test, we carry out two cycles with the sequence: 20 iterations for pose and 10 iterations for scale. At the end of this sequence, 60 iterations have been done and a close pose and scale are obtained (see 4.16(d)). Then, we carry out four cycles with the sequence: 20 iterations for pose, 5 iterations for scale, and 10 iterations for shape, getting a refinement in pose and scale for a more approximated shape

(see fig. 4.16(f)). The final scale is $s = 1.0029$, with a reduction in scale error of 98.55%. The final latent variable is $\chi = [0.3921 \ 1.6836]$. The Euclidean distance between the estimated latent variable and the ground truth decreased from 0.8278 in the initialization to 0.0675 in iteration 200, getting a reduction in error of 91.85%. The final position is $t_{wo} = [0.0033 \ -0.0105 \ 0.0028]$, with an Euclidean distance to the origin of 0.0114. The final orientation is $\alpha_x = -0.0665^\circ$, $\alpha_y = -0.0908^\circ$, and $\alpha_z = -0.0221^\circ$. The reduction in position error is 96.20% and in rotation around z-axis is 99.89%. Figure 4.17 shows the evolution of the energy, drawing with red, green, and blue colors for identifying the type of optimization done (pose, scale, shape, respectively). In this figure, the energy is drawn even when it increases. However, the Levenberg-Marquardt algorithm just updates the variables when the energy diminishes, discarding new data in other case.

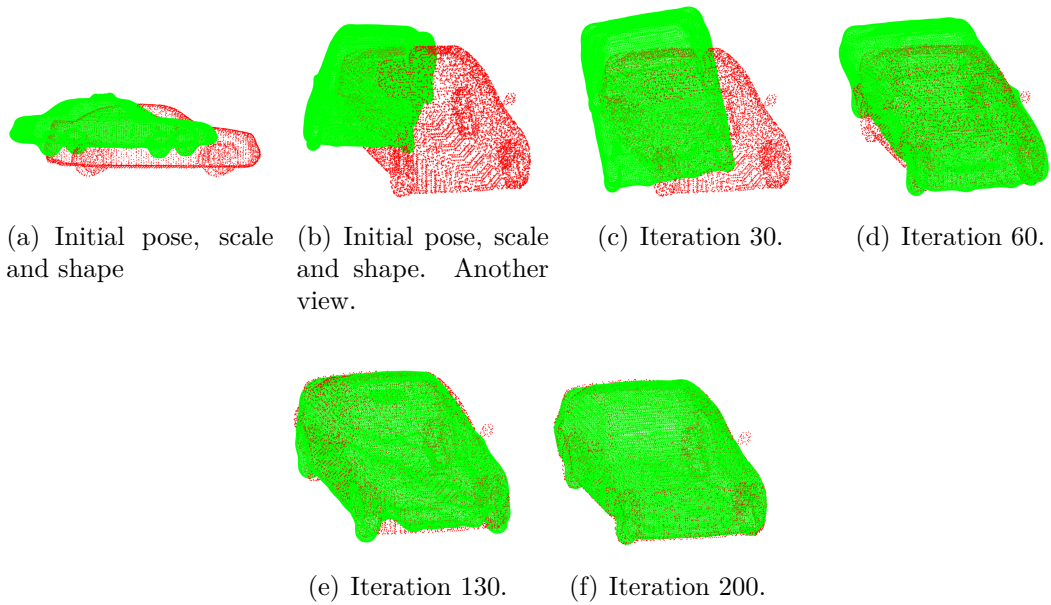


Figure 4.16: Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process. The point cloud is drawn in red.

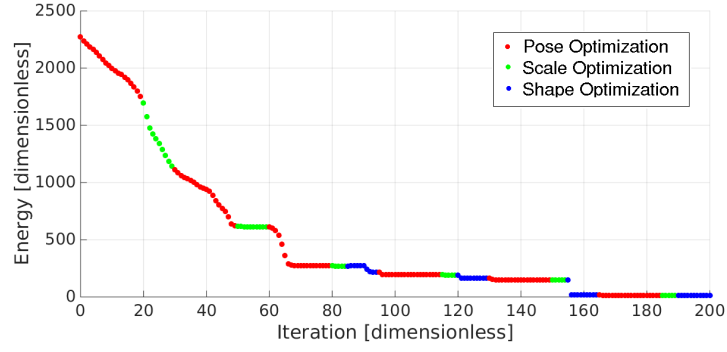


Figure 4.17: Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt.

4.8.5 RESULTS IN POSE, SCALE AND SHAPE WITH REAL DATA (KINECT)

Now, instead of using a point cloud coming from the vertices of the triangles that make up the surface of a 3D model, we use a point cloud coming from a depth map taken with the Kinect 1.0. The Kinect was moved by hand in front of a toy car while VGA images were captured at 30fps. We selected one depth and intensity image (see fig. 4.18) and manually segmented the toy car, getting the mask of fig. 4.19(a). The resulting 3D points X_r associated to the segmented car (red points in fig. 4.19(b)) are referenced to the camera coordinate system, where the z -axis is perpendicular to the image plane (see these coordinate systems in fig. 4.20). We compute 3D points referenced to the world coordinate system, $X_w = M_{wr}X_r$, where M_{wr} is a rotation of 90° around the y axis followed by a rotation of -90° around the new z axis. As was done for model alignment in section 4.3, the point cloud of the toy car referenced to the world system X_w is translated from its center of gravity to the origin $X_w^c = X_w - \bar{X}_w$, and the average distance of each point to the new origin s_{pc} is computed using eq. (4.1). The same process is carried out for the vertices of the 3D model for getting s_m . The initial scale is computed with eq. (4.2).

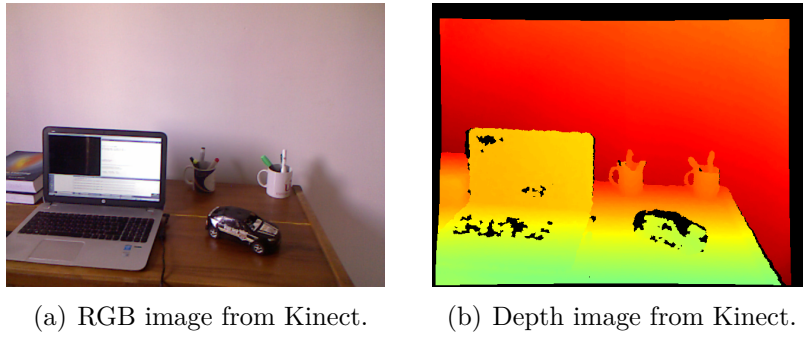


Figure 4.18: (a) RGB and (b) depth images of a toy car over a desk (dark red represents high depth). The infrared light emitted by the Kinect is absorbed by black surfaces, producing regions without data.

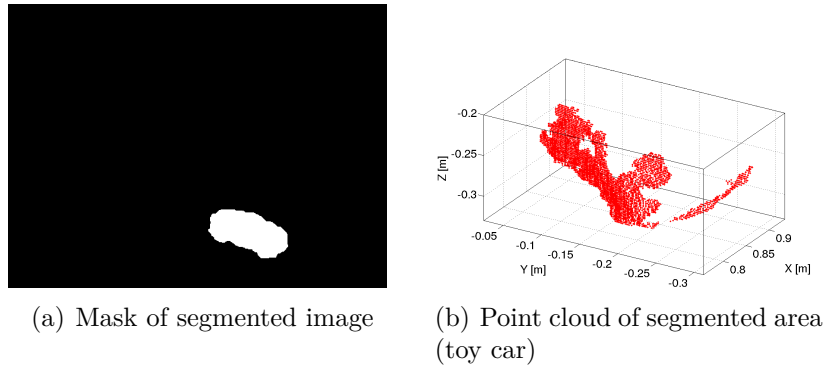


Figure 4.19: (a) Mask of the segmented car. (b) Point cloud of region associated to the segmented area (lateral side of the toy car).

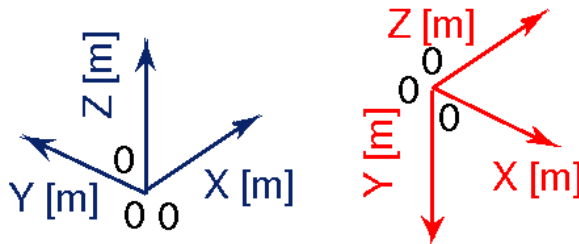


Figure 4.20: World (in blue) and camera (in red) coordinate systems.

The initial position of the car is defined as the center of gravity of the point cloud referenced to the world coordinate system, adding 4% of the x component (z axis in the camera coordinate system) to itself since the data belongs just to a side of the whole car.

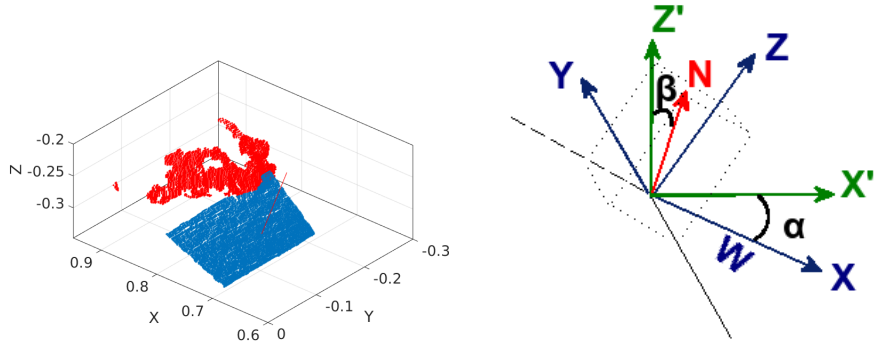
$$\mathbf{t}_{wo} = [1.04\bar{X}_{wx} \quad \bar{X}_{wy} \quad \bar{X}_{wz}]' \quad (4.35)$$

As in [13], we assume that the toy car is over a flat surface, so computing the normal of this surface allows us to estimate an initial orientation of the car. This process begins by calculating 3D points for pixels in a region below the segmented area which belong to the supporting plane. Then, RANSAC is applied: three points are randomly selected and the parameters that define a plane are computed. The points that fulfill with the computed plane (under a threshold) are counted. The random selection process was repeated 100 times and we chose the parameters that generated the largest number of inliers (points under a threshold when the plane is evaluated). Figure 4.21(a) shows the point cloud of the car (in red), the point cloud of the supporting plane (in blue) and the normal $\mathbf{n}$ to the supporting plane (red line). This normal is used for computing two angles α and β that moves the embedding function to the supporting plane,

$$\alpha = \text{acos} \left(\frac{n_z}{\sqrt{n_x^2 + n_z^2}} \right) \quad (4.36)$$

$$\beta = \text{acos}(\sqrt{n_x^2 + n_z^2}) \quad (4.37)$$

The transformation consists of a rotation of $-\alpha$ over the y axis followed by a rotation of β over the new x axis. These angles are shown in fig. 4.21(b).



(a) Point cloud of the toy car (in red), point cloud of the supporting plane (in blue) and normal to the supporting plane (red line) (b) Angles α and β used for computing the initial orientation

Figure 4.21: (a) Point cloud of the toy car with the normal to the supporting plane. (b) Angles that move the embedding function to the flat surface.

The third angle γ needed for totally defining the orientation of the embedding function is the rotation around the unit normal (new z axis). This angle is found through exhaustive search with α between 10° and 360° and with an increment of 10° ; the selected α is the one that produces the minimum energy when eq. (4.10) is evaluated. Finally, the rotation matrix R_{wo} is

$$R_{wo} = \text{Rot}(Y, -\alpha) \text{Rot}(X', \beta) \text{Rot}(Z'', \gamma) \quad (4.38)$$

and the initial transformation matrix M_{wo} is

$$M_{wo} = \begin{bmatrix} R_{wo} & \mathbf{t}_{wo} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \quad (4.39)$$

Summarizing, the initial position is $\mathbf{t}_{wo} = [0.8459 \quad -0.1524 \quad -0.2598]$, the initial orientation corresponds to $\alpha = 20.0017^\circ$, $\beta = 2.18^\circ$ and $\gamma = -20^\circ$. For comparing this orientation with the final one, we use rotations over fixed axis with $\alpha_x = 9.1136^\circ$, $\alpha_y = -18.0095^\circ$, and $\alpha_z = -21.0621^\circ$. The initial scale is $s = 0.1571$, and the initial shape is the one associated to the reference model employed in the model alignment process $\chi = [0.3895 \quad 1.6162]$. Figures 4.22(a) and 4.22(b) show the initial conditions for the model. For this test, we carry out two cycles with the sequence: 20 iterations for pose and 5 iterations for scale. At the end of this sequence, 50 iterations have been done and very close pose and scale estimations are obtained (see 4.22(d)). With these estimations we can perform exhaustive search over the $N_m = 49$ models of cars used for learning the latent space. The latent variable with the 3D level set that produces the minimum energy $\chi = [0.4560 \quad 2.4675]$ is used as initial value in the following refinement process. Finally, we carry out three cycles with the sequence: 5 iterations for shape, 10 iterations for pose and 5 iterations for scale, getting a refinement in pose and scale for a more approximated shape (see fig. 4.22(f)). The final scale is $s = 0.1602$, and the final latent variable is $\chi = [0.5858 \quad 2.6147]$. The final position is $\mathbf{t}_{wo} = [0.8499 \quad -0.1377 \quad -0.2530]$ and the final orientation is $\alpha_x = 5.7633^\circ$, $\alpha_y = -9.0734^\circ$ and $\alpha_z = -26.0871^\circ$.

Figure 4.23 shows the projection of the 3D model into the image plane. Note that the projection superimposes tightly the region of the real car in the RGB image. Figure 4.24 shows the evolution of the energy, drawing with red, green, and blue colors for identifying the type of optimization done (pose, scale, shape, respectively). In this figure, the energy is drawn even when it increases. However, the Levenberg-Marquardt algorithm just updates the variables when the energy diminishes and discards data in other case.

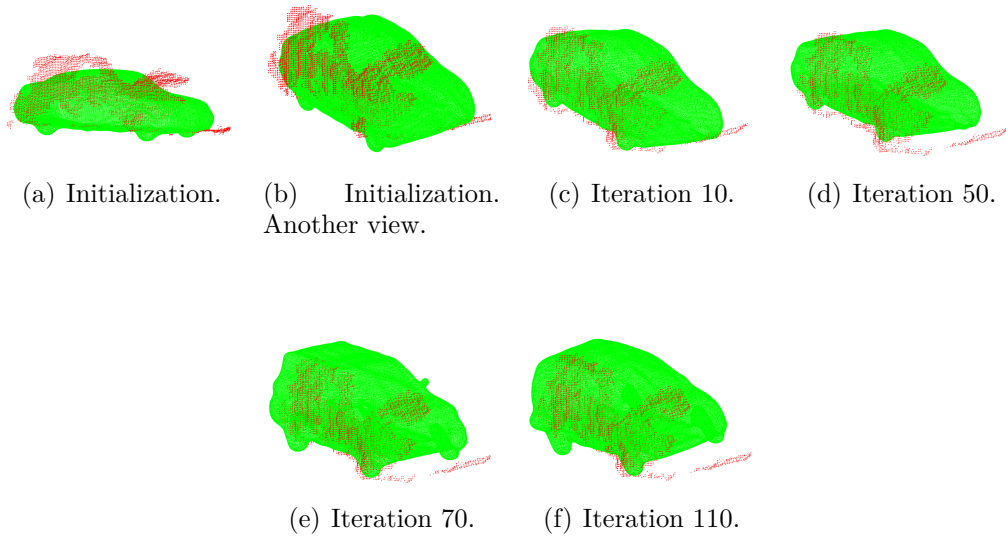


Figure 4.22: Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process, for real data using the Kinect. The point cloud is drawn in red.

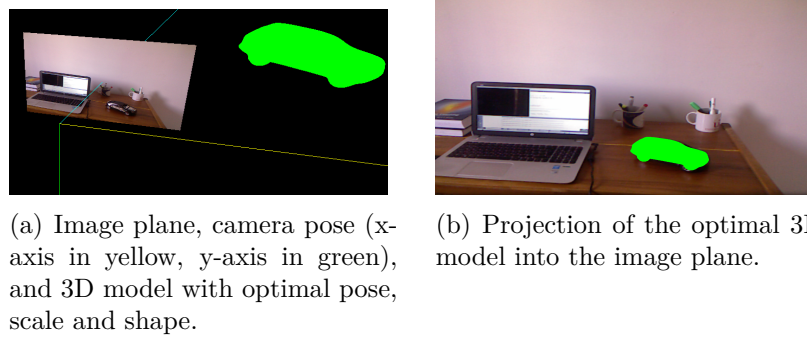


Figure 4.23: (a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.

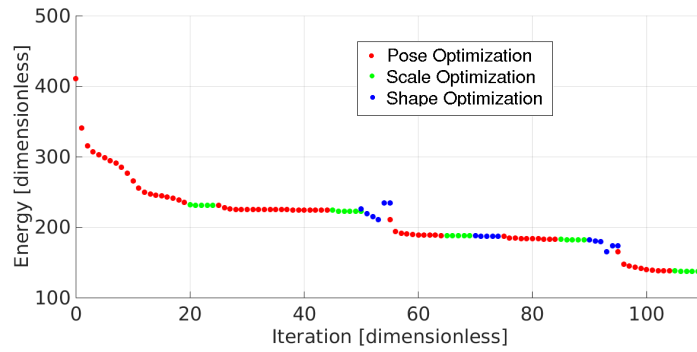


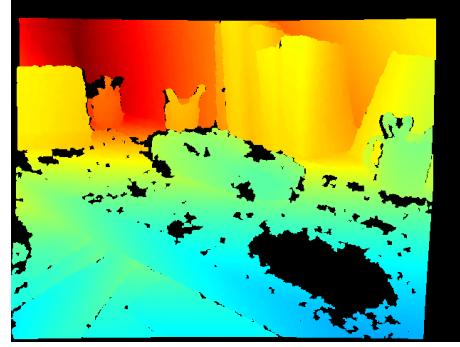
Figure 4.24: Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt. The data comes from the Kinect 1.0.

4.8.6 RESULTS WITH DEPTH MAP BUILT WITH REAL DATA (MONOCULAR CAMERA)

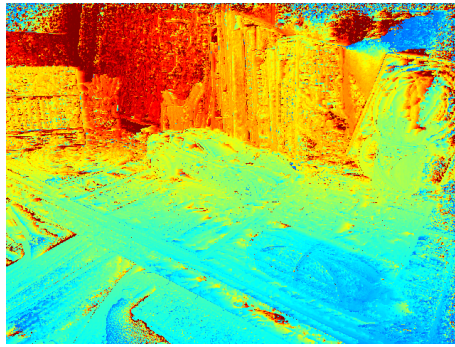
For this experiment we use a depth map built with intensity images. These images were taken with the monocular camera integrated in the Kinect 1.0 (a gray image got from a RGB image is shown in fig. 4.25(a)). The kinect was moved by hand in front of a toy car while VGA images were captured at 30fps. Figure 4.25(b) shows the depth map from the Kinect that is used for computing the accuracy of the built depth map (see the initial solution got with the data term in fig. 4.25(c) and the refined one got with the regularisation term in fig. 4.25(d)). The toy car is manually segmented, getting the mask of fig. 4.26(a). The resulting point cloud (red points in fig. 4.26(b)) is aligned with a 3D model by minimizing the energy function of eq. (4.10) w.r.t. pose, scale and shape. In this figure we also can see the supporting plane (in blue) and the normal vector (red line).



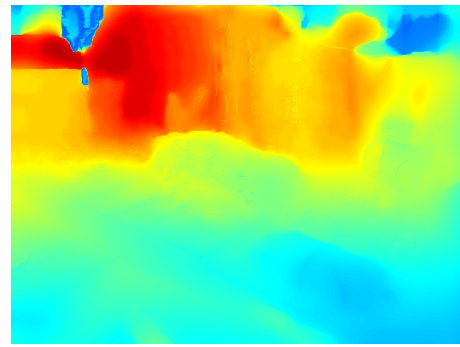
(a) Reference image $I_r(\mu)$.



(b) Depth map for reference image $D_r(\mu)$.



(c) Depth map got by solving just the data term.

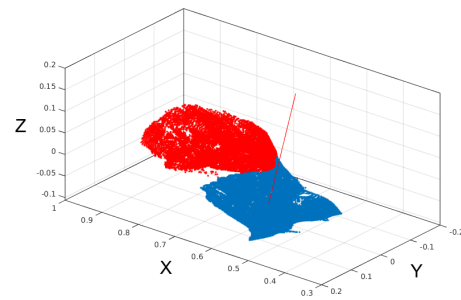


(d) Solution in depth using the primal-dual algorithm.

Figure 4.25: (a) Reference image in gray, (b) depth for reference image coming from sensor, (c) initial depth got by solving the data term and (d) solution in depth got by using the primal-dual algorithm. Red represents high depth while blue represents low depth.



(a) Mask of segmented image



(b) Point cloud of segmented area (in red), point cloud of the supporting plane (in blue) and normal to supporting plane (red line).

Figure 4.26: (a) Mask of the segmented car. (b) Point cloud of region associated to the segmented area and norm to the supporting plane.

Then, they are referenced to the world coordinate system and an initial scale, position and orientation is computed in similar way as was described in section 4.8.5, for the experiment with the RGBD sensor.

Summarizing, the initial position is $\mathbf{t}_{wo} = [0.7433 \ 0.0065 \ 0.0264]$, the initial orientation corresponds to (rotations over fixed axis) $\alpha_x = 14.3506^\circ$, $\alpha_y = -17.8619^\circ$, and $\alpha_z = -31.6360^\circ$. The initial scale is $s = 0.2222$, and the initial shape is the one associated to the reference model employed in the model alignment process $\chi = [0.3895 \ 1.6162]$. Figures 4.27(a) and 4.27(b) show the initial conditions for the model. For refining the initial estimation we carry out two cycles with the sequence: 15 iterations for pose and 5 iterations for scale. At the end of this sequence, 40 iterations have been done and very close pose and scale estimations are obtained (see fig. 4.27(d)). With these estimations we can perform exhaustive search over the $N_m = 49$ models of cars used for learning the latent space. The latent variable with the 3D level set that produces the minimum energy $\chi = [-0.0659 \ 0.2060]$ is used as initial value in the following refinement process. Finally, we carry out three cycles with the sequence: 10 iterations for shape, 10 iterations for pose and 5 iterations for scale, getting a refinement in pose and scale for a more approximated shape (see fig. 4.27(f)). The final scale is $s = 0.2325$, and the final latent variable is $\chi = [-0.0763 \ 0.1037]$. The final position is $\mathbf{t}_{wo} = [0.7614 \ 0.0173 \ 0.0393]$ and the final orientation is $\alpha_x = 13.3060^\circ$, $\alpha_y = -19.9214^\circ$ and $\alpha_z = -37.1394^\circ$.

Figure 4.28 shows the projection of the 3D model into the image plane. Note that, like in the previous case, the projection superimposes tightly the region of the real car in the RGB image. Figure 4.29 shows the evolution of the energy, drawing with red, green, and blue colors for identifying the type of optimization done (pose, scale, shape, respectively). In this figure, the energy is drawn even when it increases. However, the Levenberg-Marquardt algorithm just updates the variables when the energy diminishes and discards data in other case.

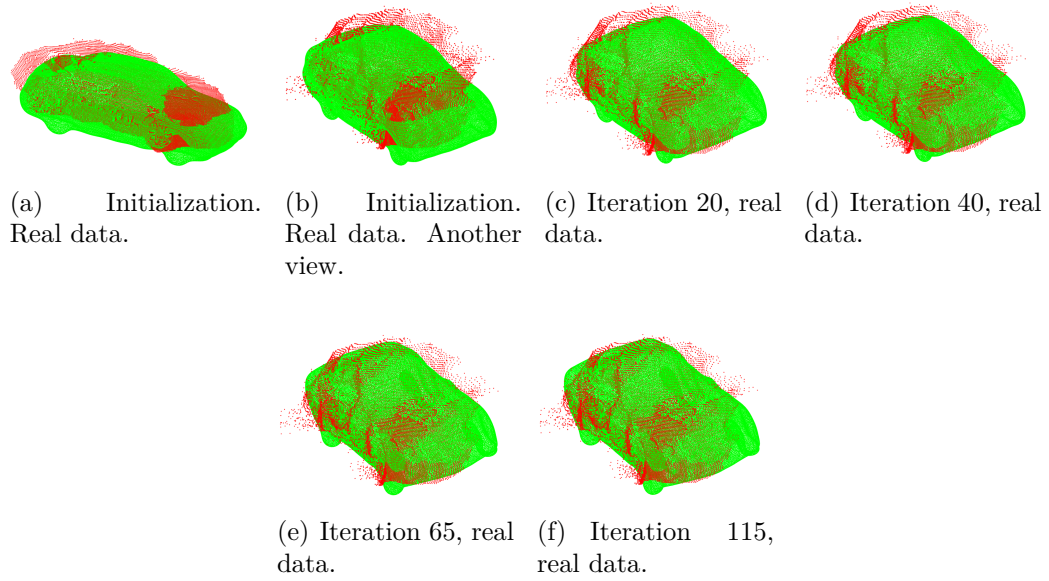
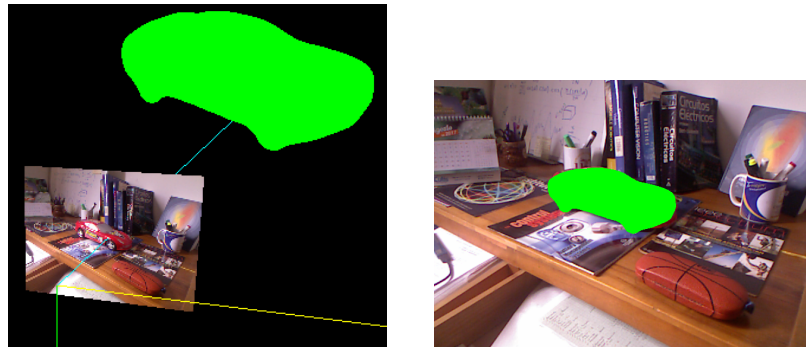


Figure 4.27: Evolution of the pose, scale, and shape of the embedding function in the alternating optimization process, for real data coming from the built depth map using a monocular camera. The point cloud is drawn in red.



(a) Image plane, camera pose (x-axis in yellow, y-axis in green), and 3D model with optimal pose, scale and shape.

(b) Projection of the optimal 3D model into the image plane

Figure 4.28: (a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.

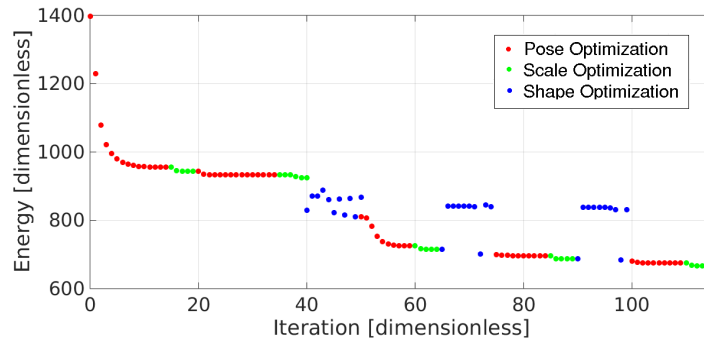


Figure 4.29: Evolution of the energy for pose, scale, and shape optimization, in alternating way, using Levenberg-Marquardt. The data comes from a depth map built with a monocular camera.

4.9 CONCLUSIONS

We have developed a system that optimizes a 3D model, represented as a 3D level-set embedding function, w.r.t. pose, scale and shape, by aligning the embedding function with depth data associated to the object of interest. The embedding function was successfully compressed from 128^3 to 25^3 DCT coefficients (compression ration of 134,22) and reduced from 25^3 to 2 dimensions (bi-dimensional latent space) using GPLVM, for a more efficient search of the optimal model.

The optimization was outstanding. For synthetic data and knowing two of the three kind of data (pose, scale and shape) we have: the shape optimization converges in 15 iterations with a reduction in error of 88.25% w.r.t. the initial estimation; the pose optimization converges in 59 iterations achieving a final error in position of 1.14cm and less that one tenth of degree in rotation around each axis (reduction in position error of 96.70% and in rotation error around z-axis of 99.93%) validating our proposed technique based on Lie algebra; the scale optimization converges in 10 iterations, getting a final error of 0.53% of the ground truth (reduction in scale error of 99.12%). For synthetic data and if no one of the three kind of data (pose, scale and shape) is known, we need to alternate the optimization algorithms for refining an initial estimation close to the solution. We begin alternating the pose and scale optimization and once this data is refined (pose and scale closer to the solution), we alternate them with the shape optimization for refining the shape of the model. This process took 200 iterations and the reduction of the errors are similar to the ones obtained when two of the three kind of data are know: reduction in shape of 91.85%, in position error of 96.20%, in rotation error around z-axis of 99.89% and in scale error of 98.55%.

For real data, we need to estimate an initial pose, scale and shape that are

refined in a posterior iterative process with Levenberg-Marquardt. For an initial estimation of position we use the centroid of the point cloud, adding 4% of the x component to itself since the data belongs just to a side of the whole car. For the scale we consider the average distance of each point to the centroid. We suppose a supporting plane in order to estimate two of the three angles that define the initial orientation of the car. The third angle is found with exhaustive search. This initial estimation is refined with respect to pose and scale. Next, exhaustive search is carried out over the 49 models used for learning the latent space. The resulting latent space that best fits to the data is refined in a posterior optimization that besides pose and scale, includes shape optimization. The optimal model is obtained after 110 and 115 iterations, for depth maps coming from the depth sensor and built with a monocular camera, respectively. The projections of the models into the image planes of real images show a tight overlapping with the car.

We propose as future work to use a depth and intensity-based energy function when the depth data is not significant for computing an approximated initial estimation in pose, scale and shape. We also leave as future work the automatic segmentation of the car in the intensity image and the implementation in commodity graphics hardware of the algorithm for estimating the optimal model (object shape prior) described in this chapter (implemented currently in Matlab).

CHAPTER 5

SHAPE PRIOR INTEGRATION

5.1 INTRODUCTION

High-quality depth information is important for many applications, such as 3D scene reconstruction [97], visual odometry [41], action recognition [7], [64], scene understanding [52], [103], 3D segmentation [32], [29], among others. However, depth maps generally are noisy and incomplete. On one hand, a depth sensor like the Kinect 1.0 estimates the depth of a scene by projecting a pattern of infrared light and capturing the modified pattern after reflecting on the objects. Specular or light absorbing materials, out of range regions and occlusions produce corrupted depth maps with missing information. The most challenging situation arises when unknown regions become large and irregular, affecting considerably the results of the applications. On the other hand, real environments also present changes in light conditions and low-textured objects. These facts break the Lambertian condition (photo-consistency) and affect the photometric error estimation reducing the accuracy of the depth maps built with a monocular camera. The regularisation term like the Total Variation term, in a variational framework, tackles this problem to some extent, but under difficult conditions, the estimations still have low accuracy. One alternative is to include information of a known object in the scene (object shape prior).

We take into account the outstanding performance of variational techniques applied to fusion of aerial images [46], where redundant data is used for enhancing the image with fine details and reducing noise, outliers and problems due to illumination changes by performing joint inpainting and denoising. We propose a novel way for merging two sources of depth data; one that is noisy and incomplete (obtained from the sensor), and the other one obtained from the optimal 3D model (shape prior), achieving a denoised and enhanced depth map, especially in the region of the object of interest.

The main goal in this chapter is to improve the quality of a noisy and incomplete depth map, obtained from a depth sensor or built with a monocular

camera, by including information of the 3D model that best fits to depth data of the object of interest. We can outline three main contributions: 1. We propose an approach to enhance depth maps by integrating depth measurements and shape prior data using a novel formulation of an objective function with a variational approach; 2. A novel variational formulation that integrates in one algorithm both the module for dense mapping with monocular camera and the module for depth merging using a shape prior. 3. The experiments carried out in order to compute the completeness and accuracy of inpainted and denoised depth maps.

The structure of this chapter is as follows. In section 5.2 we present similar works. In section 5.3 we present a detailed description of the proposed variational technique for merging, inpainting and denoising depth maps, in sequential (depth map building followed by shape prior integration) and simultaneous way (depth map building and shape prior integration at the same time), using depth data from the sensor and from the shape prior. In section 5.4 we show the experiments carried out with a depth sensor, a monocular camera and synthetic data. Finally, in section 5.5 we present the conclusions.

5.2 RELATED WORK

In this section we consider the systems described in related work of chapter 4 but now we give some details about the techniques used for integrating the shape priors into the depth maps or into the dense reconstructions. We also describe the system [46] that motivates our proposed technique, for merging depth data of the shape prior and the depth map from the sensor, with an additional inpainting and denoising effect.

In order to improve the accuracy of depth maps created with a monocular camera, a shape prior that considers the scene with box-like structures, with extensive low-texture surfaces like walls, ceilings and floors, can be used. This scene shape prior allows to improve the whole scene. For example, the system [69] estimates depth maps using a monocular camera in workspaces with large plain structures like floors, walls or ceilings. The curvature of a second order approximation of the data term at the minimum cost defines the reliability of the initial depth, getting good depth estimations at the borders of bland objects (high curvature). Good depth data is propagated to an interior pixel (inpainting) from the closest valid pixels along the main 8 star directions by defining a geomteric constraint with a variational approach, that favours solutions with affine surfaces (prior). The energy is minimized in straight way with the primal-dual algorithm. The system [11] shows outstanding performance in low-textured image regions and for low-parallax camera motion. It includes a term, besides the data term and regularisation term, that depends on three scene priors: planarity of homogeneous color regions (using superpixels), the

repeating geometric primitives of the scene (data-driven 3D primitives learned from RGBD data), and the Manhattan structure of indoor rooms (layout estimation and classification of box-like structures). The scene prior terms model the distance from every point to its estimated planar prior. The energy is minimized using a variational approach with a coupling term, the primal-dual algorithm and exhaustive search. In contrast to [69], it requires a preprocessing step.

Other kind of shape prior, the object-based one, is also employed for improving the dense reconstruction built with a monocular camera. The system [3] uses a warped mean shape and anchor points to model objects and improve the reconstruction got with structure from motion (SFM) or multi-view stereo (MVS) techniques. The augmented reconstruction consists in matching anchor points (HOG features), warping by anchor points (an extension of thin-plate spline transformations [74]) and refinement. The system [13], incorporates PTAM [44] and a module for building depth maps with a variational approach in order to create a dense reconstruction that is enhanced by adding data of a model optimized w.r.t. pose, scale and shape, using GPLVM to model the shape. The optimal object is directly fused with the volumetric representation (shifting TSDF values of voxels associated to the car), obtaining full scaled 3D reconstructions with the known object segmented from the scene, and some of the unseen parts of the scene reconstructed (robustness to occlusions).

The system [46] computes range images and transforms a large point cloud into a common orthographic aerial view. This noisy color image with undefined areas due to occlusions and non-stationary objects is subjected to denoising and inpainting by integrating redundant observations of the same scene using variational techniques. Since TV-based regularisation produces unnatural inpainting, a wavelet regularisation (dual-tree complex wavelet transform DTCWT) is used, which enables a natural-appearing recovery of fine details in the images. The Huber norm of the data term measures the discrepancy between the input observations and the sought solution. It uses the iterative and alternating primal-dual algorithm for minimizing the proposed energy. The experiments over synthetic data and real aerial images show high quality reconstructions.

We employ object shape priors like [49] and [13], but instead of modifying a point cloud like [49] or a volumetric structure directly like [13], we enhance the built depth maps by merging a synthetic depth map obtained from the shape prior using a novel variational formulation that considers an additional term for the shape prior data, like is done in [69], [11], and exploiting the ideas of [46] where color aerial images are fused considering the redundant information of the scene.

5.3 MERGING, INPAINTING AND DENOISING OF DEPTH MAPS

For texture inpainting TV-based methods produce results that look unnatural near recovered edges so variants like wavelet transform are used for overcoming this problem. TV-based methods are well suited for tasks like depth data integration, as was probed in [102]. In our context, an energy defined by a TV-based regularisation together with a data term that measures the discrepancy between two sources of data (the depth maps obtained from the model $\mathbf{D}_m(\boldsymbol{\mu})$ and from the sensor $\mathbf{D}_s(\boldsymbol{\mu})$) and the sought solution $\mathbf{D}_f(\boldsymbol{\mu})$, is implemented.

$$\min_{\mathbf{D}_f} E(\mathbf{D}_f(\boldsymbol{\mu})) = \int \left(\underbrace{\|\nabla \mathbf{D}_f(\boldsymbol{\mu})\|_1}_{\text{regularisation term}} + \lambda \underbrace{\sum_{k=1}^2 w_k(\boldsymbol{\mu}) \|\mathbf{D}_f(\boldsymbol{\mu}) - \mathbf{D}_k(\boldsymbol{\mu})\|_\epsilon}_{\text{data term}} \right) d\boldsymbol{\mu}, \quad (5.1)$$

where λ defines the balance between the regularisation term and the data term, $\mathbf{D}_1(\boldsymbol{\mu}) = \mathbf{D}_s(\boldsymbol{\mu})$, and $\mathbf{D}_2(\boldsymbol{\mu}) = \mathbf{D}_m(\boldsymbol{\mu})$. We apply the robust Huber norm over the data term where

$$\|\mathbf{x}\|_\epsilon = \begin{cases} \frac{\|\mathbf{x}\|_2^2}{2\epsilon} & \text{if } \|\mathbf{x}\|_2 \leq \epsilon, \\ \|\mathbf{x}\|_1 - \frac{\epsilon}{2} & \text{otherwise} \end{cases} \quad (5.2)$$

The L_2^2 norm works for small values, which is appropriate for handling Gaussian noise, promoting smooth depth regions while the L_1 norm allows discontinuities at depth edges for larger errors. Missing data in depth maps (undefined pixels) defines the inpainting domain,

$$w_k(\boldsymbol{\mu}) \in \{0, 1\}, \quad (5.3)$$

where $w(\boldsymbol{\mu}) = 0$ corresponds to pure inpainting at location $\boldsymbol{\mu}$. The energy defined in eq. (5.1) is minimised using a first-order primal-dual algorithm.

5.3.1 PRIMAL-DUAL ALGORITHM

The regulariser and the data term of eq. (5.1) can be written in a more general form:

$$\min_{\mathbf{y}} F(A\mathbf{y}) + \sum_{k=1}^2 G_k(\mathbf{y}), \quad (5.4)$$

In our case, $A = \nabla$ the gradient operator, $F(A\mathbf{y}) = \|A\mathbf{y}\|_1$, $G_k(\mathbf{y}) = \lambda \boldsymbol{\varphi}_k \|\mathbf{y} - \boldsymbol{\varphi}_k\|_\epsilon$, $\mathbf{y}$, $\boldsymbol{\varphi}_k$ and $\boldsymbol{\varpi}_k$ are row-wise vector versions of the sought solution $\mathbf{D}_f(\boldsymbol{\mu})$, the depth sources $\mathbf{D}_k(\boldsymbol{\mu})$, and the matrix w_k that defines the inpainting domain, respectively. We do Legendre-Fenchel transforms in this way:

$$F(A\mathbf{y}) = \max_{\|\boldsymbol{\varrho}\|_2 \leq 1} \langle A\mathbf{y}, \boldsymbol{\varrho} \rangle - F^*(\boldsymbol{\varrho}) \quad (5.5)$$

$$G_k(\mathbf{y}) = \max_{|\mathbf{r}_k|_1 \leq \lambda \boldsymbol{\varpi}_k} \langle \mathbf{y} - \boldsymbol{\varphi}_k, \mathbf{r}_k \rangle - G_k^*(\mathbf{r}_k) \quad (5.6)$$

where $\boldsymbol{\varrho}$ and $\mathbf{r}_k$ are dual variables associated to the primal variables $\mathbf{y}$ and $\boldsymbol{\varphi}_k$ respectively, and $\langle \cdot, \cdot \rangle$ corresponds to the dot product. Replacing (5.5) and (5.6) in (5.4) we get the primal-dual formulation of this nonlinear problem. It is a generic saddle-point problem

$$\min_{\mathbf{y}} \max_{|\mathbf{r}_k|_1 \leq \lambda \boldsymbol{\varpi}_k, \|\boldsymbol{\varrho}\|_2 \leq 1} \langle A\mathbf{y}, \boldsymbol{\varrho} \rangle - F^*(\boldsymbol{\varrho}) + \sum_{k=1}^2 [\langle \mathbf{y} - \boldsymbol{\varphi}_k, \mathbf{r}_k \rangle - G_k^*(\mathbf{r}_k)] \quad (5.7)$$

where $F^*(\boldsymbol{\varrho})$ and $G_k^*(\mathbf{r}_k)$ are the convex conjugates of $F(A\mathbf{y})$ and $G_k(\mathbf{y})$, respectively, and are defined as

$$F^*(\boldsymbol{\varrho}) = \delta_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}) \quad (5.8)$$

$$G_k^*(\mathbf{r}_k) = \delta_{\mathbf{r}_k}(\mathbf{r}_k) + \frac{\epsilon}{2} \|\mathbf{r}_k\|_2^2 \quad (5.9)$$

where $\delta_{\boldsymbol{\varrho}}$ and $\delta_{\mathbf{r}_k}$ are indicator functions of the convex sets, defined as

$$\delta_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}) = \begin{cases} 0 & \text{if } \|\boldsymbol{\varrho}\|_1 \leq 1 \\ \infty & \text{otherwise} \end{cases} \quad (5.10)$$

$$\delta_{\mathbf{r}_k}(\mathbf{r}_k) = \begin{cases} 0 & \text{if } |\mathbf{r}_k|_1 \leq \lambda \boldsymbol{\varpi}_k \\ \infty & \text{otherwise} \end{cases} \quad (5.11)$$

We propose to use the scheme of [46] (where color images for aerial applications are merged, denoised and inpainted) for merging depth data of two sources, getting an enhanced depth map. The iterative optimization corresponds to perform in alternating way gradient ascent over the dual variables and gradient descent over the primal variable $\mathbf{y}$, projecting the results onto the constraints and updating the primal variable, as is summarized next.

$$\begin{cases} \boldsymbol{\varrho}^{n+1} = \text{proj}_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}^n + \sigma A \bar{\mathbf{y}}^n) \\ \mathbf{r}_k^{n+1} = \text{proj}_{\mathbf{r}_k} \left(\frac{\mathbf{r}_k^n + \sigma (\bar{\mathbf{y}}^n - \boldsymbol{\varphi}_k)}{1 + \sigma \epsilon} \right), & k = 1, 2. \\ \mathbf{y}^{n+1} = \mathbf{y}^n - \tau (A^* \boldsymbol{\varrho}^{n+1} + \sum_{k=1}^2 \mathbf{r}_k^{n+1}) \\ \bar{\mathbf{y}}^{n+1} = \mathbf{y}^{n+1} + \Phi(\mathbf{y}^{n+1} - \mathbf{y}^n) \end{cases} \quad (5.12)$$

where A^* is the adjoint operator of the gradient operator and corresponds to the negative divergence operator, $\Phi = 1$, $\text{proj}_{\boldsymbol{\varrho}}$ and $\text{proj}_{\mathbf{r}_k}$ are projections of

the dual variables $\boldsymbol{\varrho}$ and $\mathbf{r}_k$, respectively, onto convex sets. They are defined for each element of the vectors as

$$\text{proj}_{\boldsymbol{\varrho}}(\tilde{\boldsymbol{\varrho}}) = \frac{\tilde{\boldsymbol{\varrho}}}{\max(1, |\tilde{\boldsymbol{\varrho}}|)} \quad (5.13)$$

$$\text{proj}_{\mathbf{r}_k}(\tilde{\mathbf{r}}_k) = \begin{cases} \tilde{\mathbf{r}}_k & \text{if } |\tilde{\mathbf{r}}_k| < \lambda \boldsymbol{\varpi}_k \\ \lambda \boldsymbol{\varpi}_k & \text{if } \tilde{\mathbf{r}}_k > \lambda \boldsymbol{\varpi}_k \\ -\lambda \boldsymbol{\varpi}_k & \text{if } \tilde{\mathbf{r}}_k < -\lambda \boldsymbol{\varpi}_k \end{cases} \quad (5.14)$$

Following the algorithm 1 of [5] and the parameter settings of [46] we set the primal and dual time steps with $\tau = 0.05$ and $\sigma = 1/(8\tau)$. The Huber norm parameter ϵ and the balance parameter λ are defined in results section for each experiment. We set the initial primal variable as $\bar{\mathbf{y}}_0 = \boldsymbol{\varphi}_s$ since it is the most informative depth source. Moreover, we initialize possible missing data of $\bar{\mathbf{y}}_0$ with the average depth of $\boldsymbol{\varphi}_s$. The dual variables $\boldsymbol{\varrho}$ and $\mathbf{r}_k$ are initialized with zeros.

5.3.2 PUTTING TOGETHER DEPTH MAP ESTIMATION AND SHAPE PRIOR INTEGRATION

The algorithm described in chapter 3 for building a depth map with a monocular camera and the algorithm described previously in this chapter for integrating the shape prior with a depth map are based on variational techniques that are solved with the primal-dual algorithm so they share common modules. Moreover, the object of interest is static and rigid so its pose, scale and shape do not change with time. We exploit these facts for implementing one algorithm that takes a shape prior estimated previously (for example in a previous keyframe) and makes simultaneously depth map building and shape prior integration. Now, we integrate shape prior data into the energy functional (3.9) with an additional term, the shape prior term,

$$E(\boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu})) = \underbrace{\int (w(\boldsymbol{\mu}) \|\nabla \zeta_r(\boldsymbol{\mu})\|_{\epsilon})}_{\text{regularisation term}} + \underbrace{\lambda_m w_m(\boldsymbol{\mu}) \|\zeta_r(\boldsymbol{\mu}) - \zeta_m(\boldsymbol{\mu})\|_{\epsilon_m}}_{\text{shape prior term}} + \underbrace{\lambda \mathbf{C}(\boldsymbol{\mu}, \zeta_r(\boldsymbol{\mu}))}_{\text{data term}} d\boldsymbol{\mu}, \quad (5.15)$$

where λ_m is a balance factor for the shape prior term, $w_m \in \{0, 1\}$ defines the inpainting domain of the inverse depth map obtained from the model $\zeta_m(\boldsymbol{\mu})$. The shape prior term forces the solution $\zeta_r(\boldsymbol{\mu})$ to be similar to the shape prior $\zeta_m(\boldsymbol{\mu})$. Using the decoupling approach and discretizing the energy we have

$$\begin{aligned}
 \min_{\mathbf{y}, \mathbf{z}} \quad & \underbrace{\|AW\mathbf{y}\|_\epsilon}_{\text{regularisation term}} + \underbrace{\lambda_m \boldsymbol{\varpi}_m \|(\mathbf{y} - \boldsymbol{\varphi}_m)\|_{\epsilon_m}}_{\text{shape prior term}} + \\
 & \underbrace{\frac{1}{2\theta} \|\mathbf{y} - \mathbf{z}\|_2^2}_{\text{decoupling term}} + \underbrace{\lambda \mathbf{C}(\mathbf{z})}_{\text{data term}},
 \end{aligned} \tag{5.16}$$

where $A = \nabla$ is the gradient operator, W is the element-wise weighting matrix, $\mathbf{y}$, $\boldsymbol{\varphi}_m$, $\mathbf{z}$ and $\boldsymbol{\varpi}_m$ are row-wise vector versions of the sought solution $\zeta_r(\boldsymbol{\mu})$, the inverse depth map of the model $\zeta_m(\boldsymbol{\mu})$, the auxiliary variable η and the inpainting domain of the inverse depth map of the model $w_m(\boldsymbol{\mu})$, respectively. For a fixed auxiliary variable $\mathbf{z}$, the regularisation term and the shape prior term have the general form of eq. (3.17), where $F(A\mathbf{y}) = \|AW\mathbf{y}\|_\epsilon$, $G_k(\mathbf{y}) = \lambda \boldsymbol{\varpi}_k \|\mathbf{y} - \boldsymbol{\varphi}_m\|_\epsilon$. With Legendre-Fenchel transforms, we get

$$\begin{aligned}
 \min_{\mathbf{y}} \quad & \max_{\|\mathbf{r}_k\|_1 \leq \lambda \boldsymbol{\varpi}_k, \|\boldsymbol{\varrho}\|_2 \leq 1} \langle AW\mathbf{y}, \boldsymbol{\varrho} \rangle - F^*(\boldsymbol{\varrho}) + \\
 & \langle \mathbf{y} - \boldsymbol{\varphi}_m, \mathbf{r}_m \rangle - G_k^*(\mathbf{r}_k) + \frac{1}{2\theta} \|\mathbf{y} - \mathbf{z}\|_2^2
 \end{aligned} \tag{5.17}$$

where $\boldsymbol{\varrho}$ and $\mathbf{r}_k$ are dual variables associated to the primal variables $\mathbf{y}$ and $\boldsymbol{\varphi}_m$ respectively. $F^*(\boldsymbol{\varrho})$ and $G_k^*(\mathbf{r}_k)$ are the convex conjugates of $F(A\mathbf{y})$ and $G_k(\mathbf{y})$, respectively. They are defined as

$$F^*(\boldsymbol{\varrho}) = \delta_{\boldsymbol{\varrho}}(\boldsymbol{\varrho}) + \frac{\epsilon}{2} \|\boldsymbol{\varrho}\|_2^2 \tag{5.18}$$

$$G_k^*(\mathbf{r}_k) = \delta_{\mathbf{r}_k}(\mathbf{r}_k) + \frac{\epsilon_m}{2} \|\mathbf{r}_k\|_2^2 \tag{5.19}$$

$\delta_{\boldsymbol{\varrho}}$ and $\delta_{\mathbf{r}_k}$ are the same as the ones defined in eq. (5.10) and (5.11). The parameter settings is similar to the one used for denoising, inpainting and integration described previously. Considering the auxiliary variable $\mathbf{z}$ fixed, the gradient ascent and gradient descent steps of the primal dual algorithm are

$$\begin{cases} \boldsymbol{\varrho}^{n+1} = \text{proj}_{\boldsymbol{\varrho}} \left(\frac{\boldsymbol{\varrho}^n + \sigma W A \bar{\mathbf{y}}^n}{1 + \epsilon \sigma} \right) \\ \mathbf{r}_m^{n+1} = \text{proj}_{\mathbf{r}_m} \left(\frac{\mathbf{r}_m^n + \sigma (\bar{\mathbf{y}}^n - \boldsymbol{\varphi}_m)}{1 + \sigma \epsilon_m} \right) \\ \mathbf{y}^{n+1} = \frac{\mathbf{y}^n - \tau (W A^* \boldsymbol{\varrho}^{n+1} + \mathbf{r}_m^{n+1} - \frac{1}{\theta^n} \mathbf{z}^n)}{1 + \frac{\tau}{\theta^n}} \\ \bar{\mathbf{y}}^{n+1} = \mathbf{y}^{n+1} + \Phi(\mathbf{y}^{n+1} - \mathbf{y}^n) \end{cases} \tag{5.20}$$

where A^* is the negative divergence operator, σ and τ are the step size for the dual variables $\boldsymbol{\varrho}$, $\mathbf{r}_m$, and for the primal variable $\mathbf{y}$ respectively, $\Phi = 1$, $\text{proj}_{\boldsymbol{\varrho}}$ and $\text{proj}_{\mathbf{r}_m}$ are projections of the dual variables $\boldsymbol{\varrho}$ and $\mathbf{r}_m$, respectively, onto convex sets. They were defined in eq. (5.13) and (5.14). Finally, for a fixed

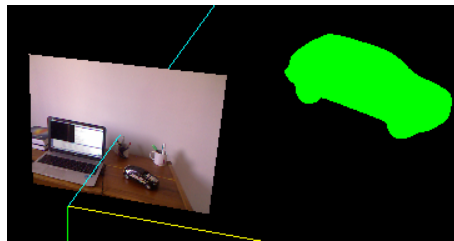
(and updated) $\mathbf{y}$ we use a point-wise search to solve the remaining non-convex function defined in eq. (3.23). The primal-dual algorithm and the exhaustive search are alternated and θ^n is decreased as was done for building the depth map in chapter 3.

5.4 RESULTS

In order to quantify the improvement in accuracy and in completeness when a synthetic depth map obtained from the shape prior is integrated (sequentially or simultaneously) into a noisy and incomplete depth map, we carry out experiments with the Kinect 1.0, with a depth map built using a monocular camera and with synthetic depth maps.

5.4.1 RESULTS WITH A DEPTH CAMERA

We have aligned an embedding function of a car model that best fits to depth data, w.r.t shape, pose and scale. This optimal 3D model (shape prior) was represented as a mesh (see fig. 5.1(a)) and projected into the image plane by deleting the depth buffer of OpenGL (see fig. 5.1(b)). Note how well the projection of the model superimposes the real car.



(a) Image plane, camera pose (x-axis in yellow, y-axis in green), and 3D model with optimal pose, scale and shape.



(b) Projection of the optimal 3D model into the image plane

Figure 5.1: (a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.

Since the main goal is to reduce noise and complete missing data in the depth map, especially in the region of the car, we get depth data of the optimal 3D model seen from the real camera pose. Figure 5.2(a) shows the depth map captured with a depth sensor while fig. 5.2(b) shows the depth map obtained from the optimal 3D model. Note that the former image has missing data due to possible dark, very reflective and translucent objects, out of range objects or occlusions. The last image has defined data just for the 3D model. Figures

5.3(a) and 5.3(b) correspond to the masks $w^s(\mu)$ and $w^m(\mu)$, where 1 represents defined data and 0 missing data (inpainting domain).

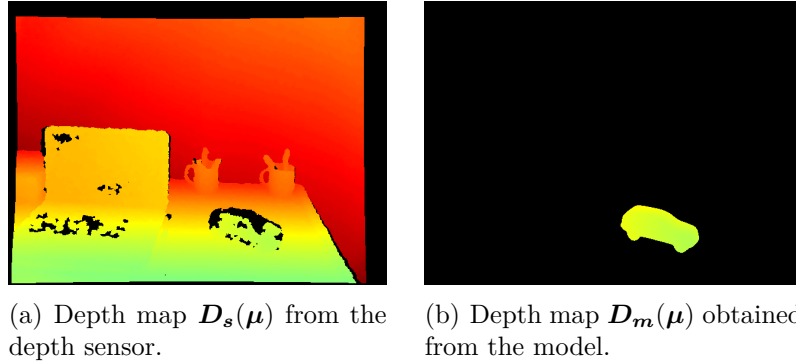


Figure 5.2: Depth maps obtained from (a) the sensor and (b) the model. Red represents high depth while green represents low depth.

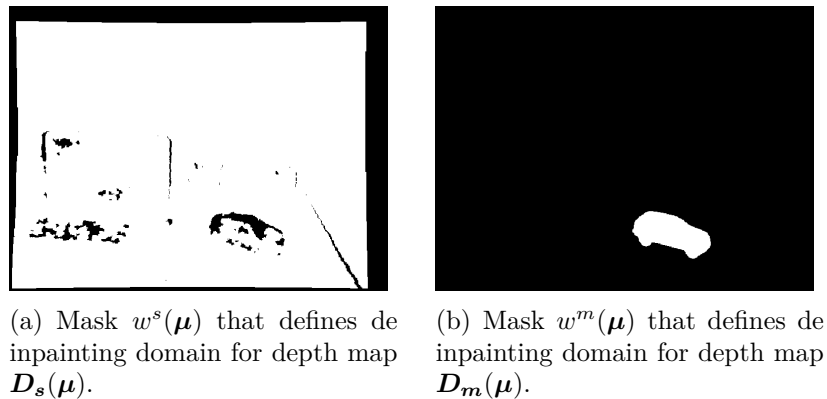


Figure 5.3: Masks that define the inpainting domain for depth maps obtained from (a) the sensor and (b) the model.

The balance λ is 0.8, the threshold for the Huber norm ϵ is 0.1 and the maximum number of iterations is 200. Figure 5.4 shows the evolution of the sought solution through the optimization process, initializing with the depth map from the sensor $D_s(\mu)$ (see fig. 5.4(a)). For iteration 100 (fig. 5.4(c)) the inpainting and denoising process has a good result but we can observe changes in depth in borders of areas where data was filled, specially in the laptop, desk and car. For iteration 200 (5.4(d)) the filled areas are smooth and undetectable, achieving a real appearance. Note that the inpainting in the top and right side of the depth map is not so good since there is not data neither in D_s nor D_m and the missing data covers large areas.

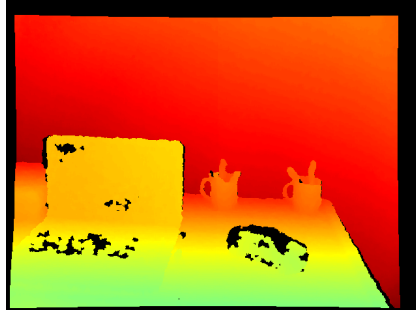
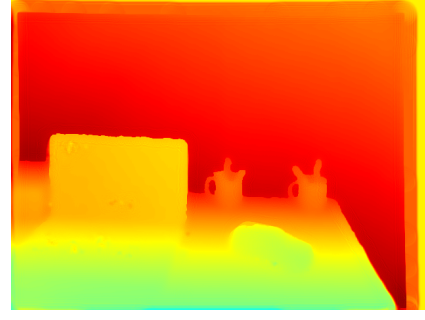
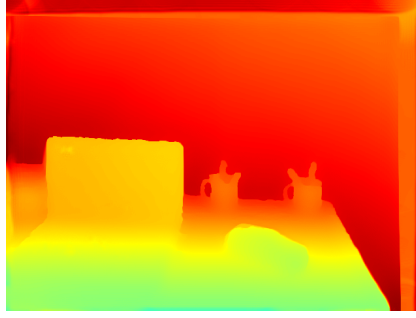
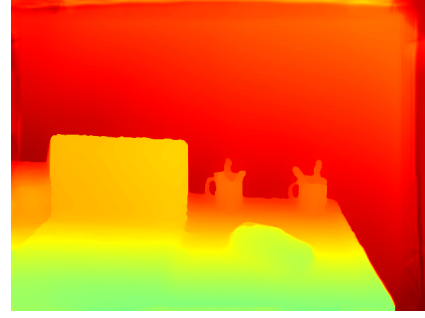
(a) Depth map from sensor $D_s(\mu)$.(b) Sought depth map $D_f(\mu)$ for iteration 40.(c) Sought depth map $D_f(\mu)$ for iteration 100.(d) Sought depth map $D_f(\mu)$ for iteration 200.

Figure 5.4: (a) Initial depth map $D_s(\mu)$. Sought depth map $D_f(\mu)$ for iterations (b) 40, (c) 100 and (d) 200 of the primal-dual algorithm. Red represents high depth while green represents low depth.

We analysed the performance of the inpainting and denoising algorithm using a slice around the x-axis (see fig. 5.5). Note that missing data stays mainly in the image borders, in the laptop area and in the car area (inside the vertical lines in fig. 5.5). It is 22.81% of the total amount of data in the slice. The solution remains similar to the sensor data but it fills missing data, integrates depth data obtained from the model, is smooth and preserves discontinuities. Evaluating the completeness of the car, we can state that the algorithm fills 33.28% of missing data. This estimation was done for pixels belonging to the segmented area of the whole car.

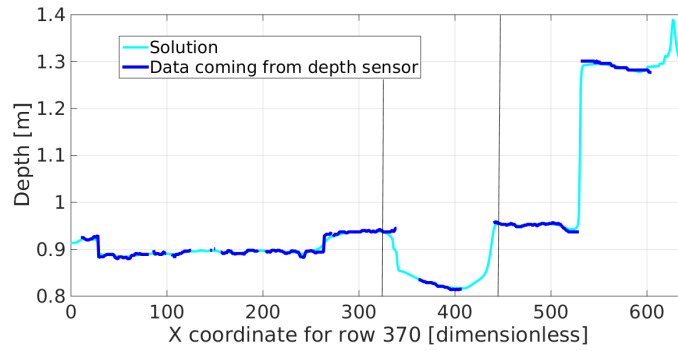


Figure 5.5: Depth for x-slice, row 370. The smoother and complete solution is drawn in cyan while the incomplete depth from the sensor is drawn in dark blue. The two vertical lines encompass the car.

5.4.2 RESULTS WITH MONOCULAR CAMERA

Now we use the depth map built with the monocular camera and the optimal 3D model w.r.t. shape, pose and scale for enhancing the quality of the depth data. Figure 5.6 shows how the projection of the optimal 3D model superimposes the toy car in the intensity image.

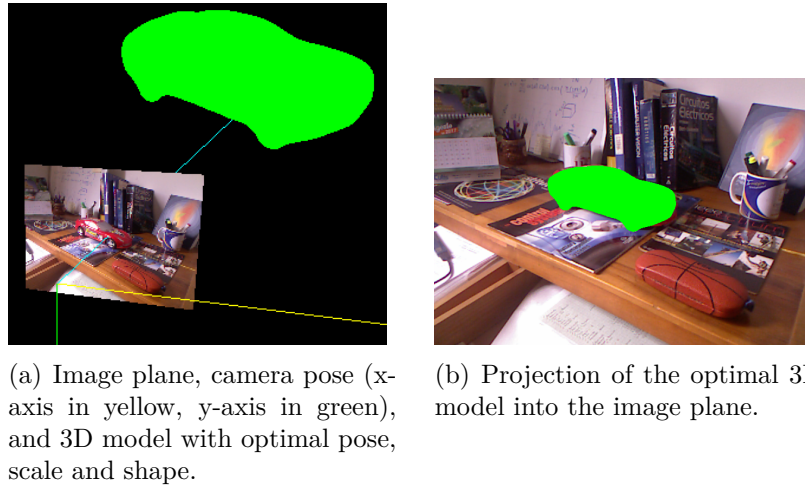


Figure 5.6: (a) Image plane, camera pose and 3D model. (b) Projected 3D model into the image plane.

The depth map built with the monocular camera is presented in fig. 5.7(a). Note that this image does not have missing data unlike the data from depth sensor (see fig. 5.2(a)) so the main goal in this case is to smooth the image and improve the accuracy of depth data for the car with the shape prior. We create a synthetic depth map with the optimal 3D model by reading the depth buffer of OpenGL from the real camera pose. This depth map is presented in fig. 5.7(b).

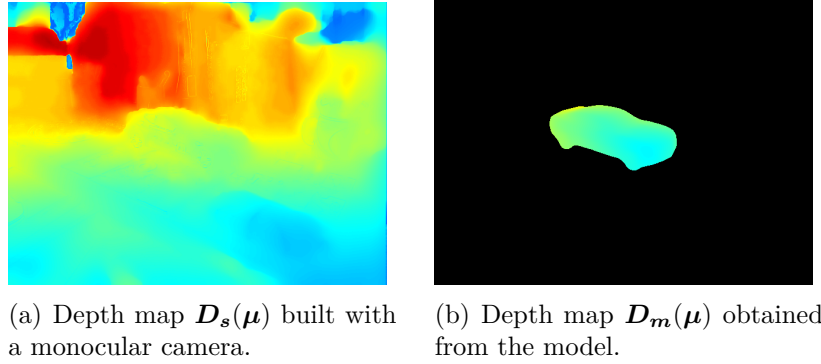


Figure 5.7: Depth maps (a) built with a monocular camera and (b) obtained from the model. Red represents high depth while blue represents low depth.

The balance λ is 0.8, the threshold for the Huber norm ϵ is 0.05 and the maximum number of iterations is 100. Figure 5.8 shows the evolution of the sought solution through the optimization process. Note that the more significant changes occur in the car area where the depth map built with the monocular camera and the depth map from the optimal 3D model interact and integrate. Outside the car area just depth smoothing is carried out.

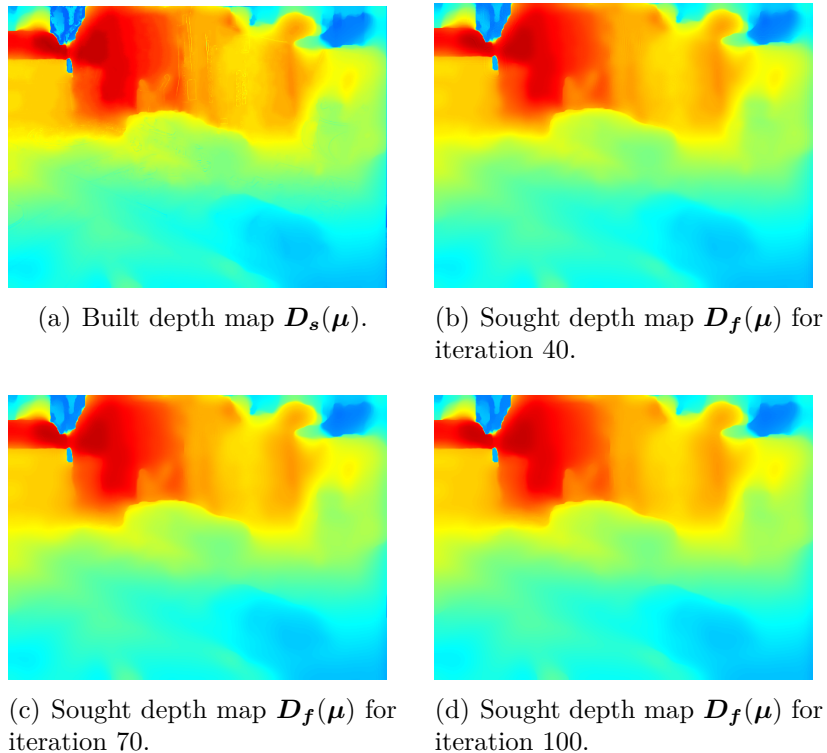


Figure 5.8: (a) Depth map built with a monocular camera. Sought depth map $D_f(\mu)$ for iterations (b) 40, (c) 70 and (d) 100 of the primal-dual algorithm. Red represents high depth while green represents low depth.

Figure 5.9 compares depth data through the x-slice for row 210. In this figure we can see the smoothing effect and the changes in depth inside the car area.

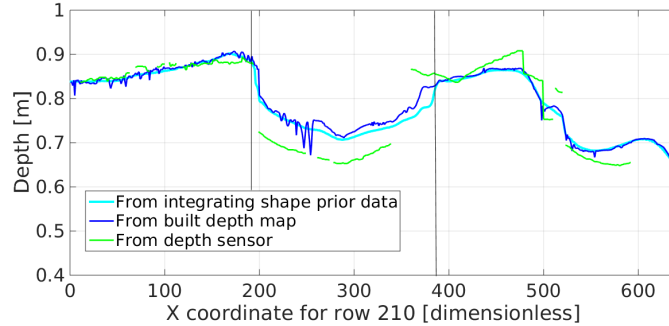
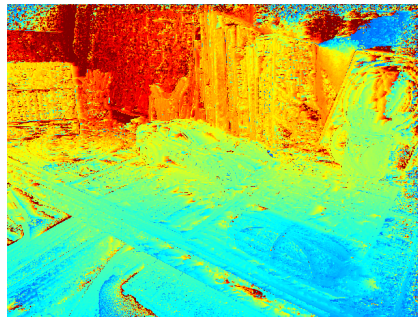


Figure 5.9: Depth for row 210; the data of the built depth map is drawn in dark blue, the smoother and complete solution resulting from merging data is drawn in cyan and the incomplete depth from the sensor is drawn in green. The vertical lines encompass the car.

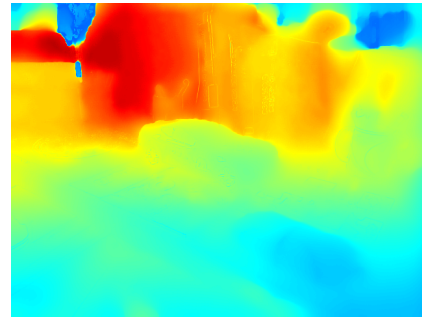
SIMULTANEOUS DEPTH MAP BUILDING AND SHAPE PRIOR INTEGRATION

The alternative approach, that makes simultaneously depth map building and shape prior integration (supposing that we already have a shape prior), produces similar results than running both algorithms sequentially. Figure 5.10(a) shows the initial depth map got by solving the data term (see eq. (3.7)) while fig. 5.10(b) shows the results of the simultaneous depth refinement and shape prior integration defined in eq. (5.20). In fig. 5.10(c) we can see a comparison of depth data through the x-slice for row 210.

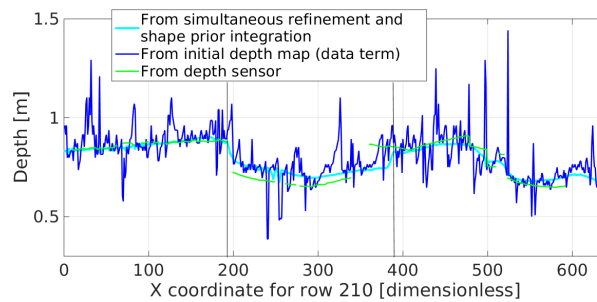
For quantifying these improvements we compute the error in depth of the built depth map, of the depth map estimated with a sequential building and shape prior integration and of the depth map estimated with a simultaneous refinement and shape prior integration, considering only the segmented area (car area) and taking the depth from the Kinect 1.0 as reference. The comparison is summarized in table 5.1. We found that the RMSE, the mean error and the median error diminish when shape prior data is integrated in a sequential or simultaneous way, although the maximum values (in both algorithms) and the standard deviation (in the simultaneous algorithm) increase a little bit due to mismatches in the borders of the car. Moreover, comparing these values we can say that the simultaneous algorithm produces depth maps with higher accuracy in the car area than the sequential one.



(a) Initial depth map computed with eq. (3.7).



(b) Depth map after 200 iterations of the algorithm for simultaneous depth refinement and shape prior integration.



(c) Depth for row 210. The vertical lines encompass the car.

Figure 5.10: (a) Initial depth map computed by solving eq. (3.7), (b) Resulting depth map after 200 iterations of the algorithm for refining and shape prior integration, (c) Depth for row 210; data from the built depth map (dark blue), the smoother and complete solution resulting from merging the shape prior (cyan) and the incomplete depth from the sensor (green).

Table 5.1: Comparison of accuracy in depth, in the car area, for the depth map built with the monocular camera DMB and the depth maps resulting from merging the shape prior data in sequential SPI and simultaneous way DMB-SPI.

Quantity \ Algorithm	DMB	SPI	DMB-SPI
RMSE [m]	0.0624	0.0572	0.0531
Max. error [m]	0.2038	0.2195	0.2316
Min. error [m]	3.4114e-04	1.0117e-04	5.7745e-05
Mean error [m]	0.0596	0.0545	0.0487
Median error [m]	0.0582	0.0537	0.0452
Standard dev. [m]	0.0187	0.0176	0.0212

Finally, we present the processing time for the main steps of the three algorithms: depth map building, shape prior integration and simultaneous depth

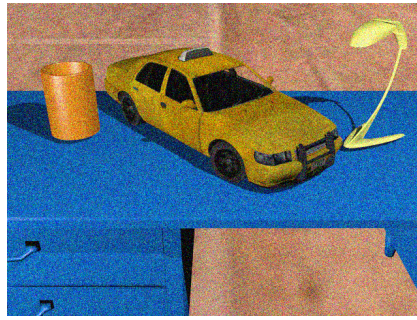
map building and shape prior integration. Since they share the same structure, with the primal-dual algorithm for solving the optimization problems, the processes are similar, as is shown in table 5.2. An iteration for building a depth map takes 18.452ms so the computation of the initial depth map and 200 iterations for refining it takes 4.7529s. An iteration for integrating the shape prior data into the built depth map takes 15.694ms so the total time for 100 iterations is 1.5694s. The time for building and merging shape prior data sequentially is 6.3223s. On the other hand, an iteration of the algorithm that makes simultaneously depth refinement and shape prior integration takes 25.287ms. Considering the time for estimating the initial depth map and 200 iterations of the simultaneous algorithm, the resulting depth map takes 6.1199s.

Table 5.2: Processing time for depth map building DMB, shape prior integration SPI and both depth map building and shape prior integration simultaneously DMB-SPI.

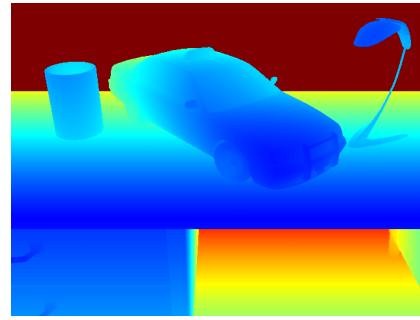
Process \ Algorithm	Time[ms] DMB	Time[ms] SPI	Time[ms] DMB-SPI
Creation initial depth map Eq. (3.7)	1062.5	—	1062.5
Update of g First line of eq. (3.22) First line of eq. (5.12) First line of eq. (5.20)	6.627	6.501	6.648
Update of r Second line of eq. (5.12) Second line of eq. (5.20)	—	5.373	5.415
Update of y Second line of eq. (3.22) Third line of eq. (5.12) Third line of eq. (5.20)	3.350	3.247	3.366
Update of z Eq. (3.23)	6.293	—	6.352
Remaining processes	2.182	0.573	3.506
TOTAL ITERATION	18.452	15.694	25.287

5.4.3 RESULTS WITH SYNTHETIC DEPTH MAPS

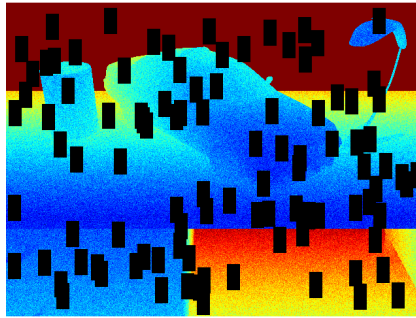
For this experiment we have created in Blender a 3D scene composed of a floor, a desk, a car, a mug and a lamp. A depth map (see fig. 5.11(b)) and a rgb image have been rendered from the virtual camera. The depth data is in the range $[1.2045 \quad 8.9610]m$. We added Gaussian white noise with SNR of 5dB to the rgb image (see fig. 5.11(a)) and 30dB to the depth map. Moreover, we simulated missing data in the depth maps with rectangles of 40×20 pixels containing Nans, randomly placed in the depth maps. The percentages of missing data that we used were 23.65% and 40.02% (see figs. 5.11(c) and 5.11(d), respectively).



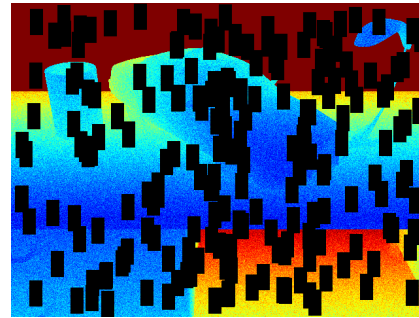
(a) Synthetic RGB image with noise (SNR= 5dB).



(b) Synthetic depth map (ground truth).



(c) Synthetic depth map with noise (SNR= 30dB) and missing data (23.65%).

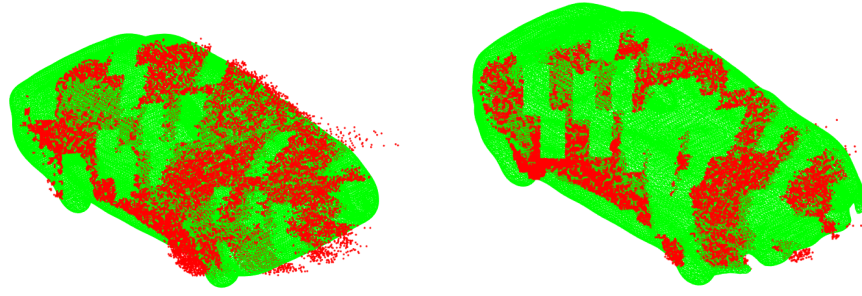


(d) Synthetic depth map with noise (SNR= 30dB) and missing data (40.02%).

Figure 5.11: Synthetic data used for estimating the accuracy of the algorithm that makes denoising, inpainting and merging. (a) RGB image with noise, (b) Depth map (ground truth), (c) Depth map with noise and missing data (23.65%), and (d) Depth map with noise and more missing data (40.02%).

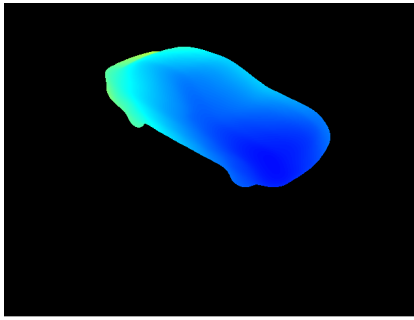
The process for estimating the optimal shape prior (initial estimation and the iterative refinement) and for getting a depth map of this model is similar to the one carried out in chapter 4. Figures 5.12(a) and 5.12(b) show the aligned models with the point clouds for 23.65% and 40.02% of missing data, respectively. Figures 5.12(c) and 5.12(d) show the synthetic depth maps obtained from the optimal models for 23.65% and 40.02% of missing data.

Our algorithm performs 500 iterations for the depth map with 23.65% of missing data and 700 iterations for the one with 40.02%. We compare the results with the ones obtained with the outstanding system of Pertuz and Kamarainen [67], employing their code publicly available. It uses anisotropic diffusion and a region-based approach for inpainting sparse depth maps. Table 5.3 shows that our algorithm outperforms [67] for both percentages of missing data. In fig 5.13 we can see that the inpainted and denoised depth map got with our algorithm is smoother and preserves discontinuities in a better way than [67]. We did not use more percentage of missing data since it affects the alignment

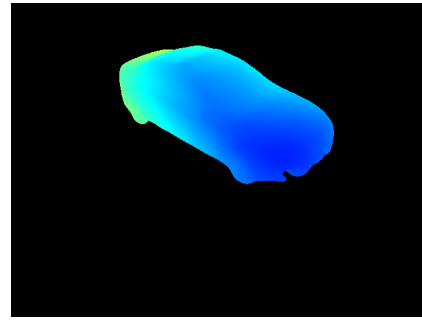


(a) Aligned 3D model with the point cloud for missing data of 23.65%.

(b) Aligned 3D model with the point cloud for missing data of 40.02%.



(c) Depth map from model for 23.65% of missing data.



(d) Depth map from model for 40.02% of missing data.

Figure 5.12: Results of the alignment process between a 3D-level set and the point cloud. (a) For depth map with missing data of 23.65% and (b) For depth map with missing data of 40.02%. Synthetic depth maps obtained from models for (c) 23.65% and (d) 40.02% of missing data.

of the model, producing incorrect estimations of the optimal model.

Table 5.3: Comparison of RMSE [m] of our algorithm and [67].

Missing data \ Algorithm	Ours	[67]
23.65%	0.1498	0.1600
40.02%	0.1869	0.2188

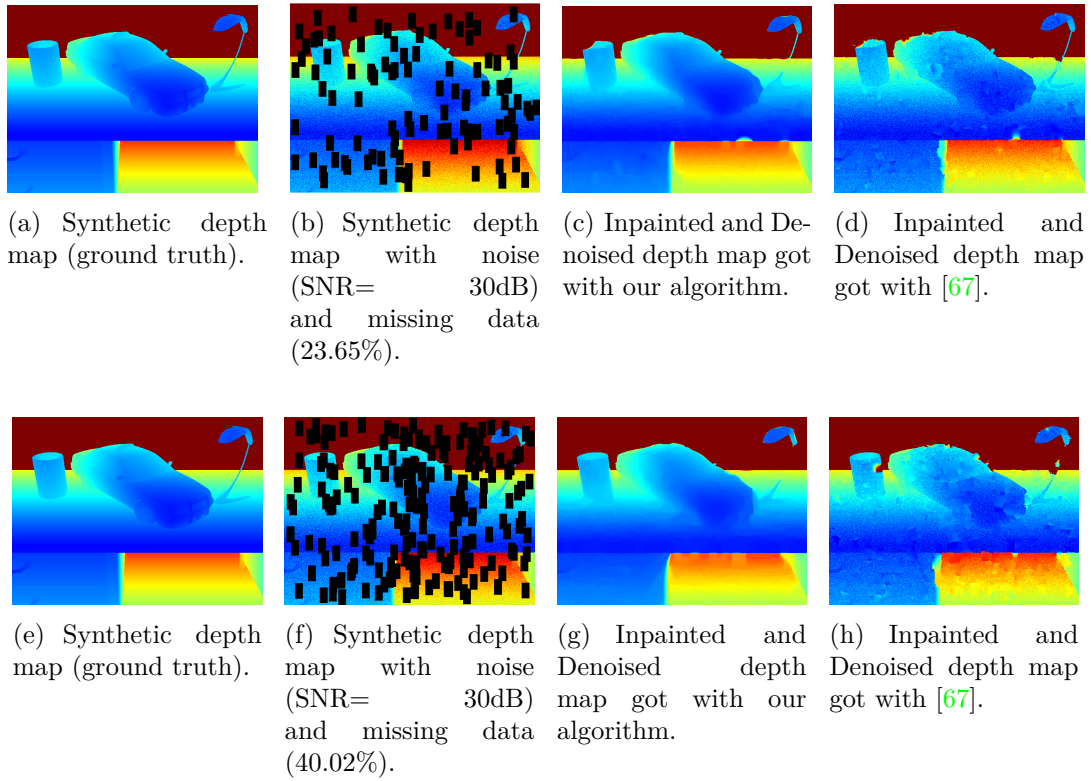


Figure 5.13: The first row contains images for the depth map with missing data of 23.65% while the second row for the depth map with missind data of 40.02%. From left to right: ground truth, depth map with noise and missing data, our results, results using [67].

5.5 CONCLUSIONS

We have developed a novel algorithm -inspired on the fusion of aerial images for enhancing the image with fine details and reducing noise when redundant information is available- that uses depth data of the optimal model (shape prior) for depth merging, denoising and inpainting, enhancing a raw depth map taken with a depth camera or built with a monocular camera. A 3D model, represented as a 3D level-set embedding function, is previously optimized w.r.t. pose, scale and shape in order to align it with a point cloud associated to the object of interest. Once this alignment is done, a synthetic depth map obtained from the optimal model is created and integrated to the built depth map. The integration is done by minimizing an energy functional made up of a regularisation term that smooths the solution and a data term that diminishes the discrepancy between the sough solution and the input depth maps: the raw depth map from the sensor and the depth map obtained from the shape prior. In the simultaneous way, the energy functional for building a depth map is modified by adding a term that constraints the solution to be similar to the synthetic depth map obtained from the shape prior.

The improvement in accuracy is quantified. The results are satisfactory. 1. In completeness of a depth map got with the depth sensor, filling 32.28% of missing data in the segmented region of the car, in 200 iterations, smoothing the data and preserving discontinuities. 2. When the shape prior is integrated into the built depth map the mean error in the segmented area diminishes from 0.0596m to 0.0545m for the sequential algorithm and from 0.0596m to 0.0487m for the simultaneous algorithm, and the data for both cases is smoother, closer to the object's shape and preserves discontinuities. 3. For the experiment with synthetic data, the accuracy of the enhanced depth maps is higher than the one obtained with the outstanding algorithm for depth recovery [67], for both percentages of missing data, 23.65% and 40.02% .

The processing time in commodity graphics hardware is 6.3223s for the sequential algorithm, with 200 iterations for building a depth map and then 100 iterations for merging shape prior data. The processing time is 6.1199s for the simultaneous algorithms with 200 iterations for building a depth map and merging shape prior data at the same time (times do not consider the estimation of the shape prior).

CHAPTER 6

DENSE RECONSTRUCTION WITH SHAPE PRIOR

6.1 INTRODUCTION

We create a dense reconstruction by fusing depth maps taken with a depth sensor or built with a monocular camera. However, depth maps are generally noisy and incomplete. Dark, very reflective and translucent objects, out of range objects and occlusions produce noise and black areas in the depth maps taken with a depth sensor. Real environments also present changes in light conditions and low-textured objects, affecting the depth maps estimated with a monocular camera. As we analysed on chapter 5, including shape information of a known object in the scene improves the accuracy of the depth map. Moreover, since our algorithm besides merging depth data of the shape prior, makes denoising and inpainting, the enhancement achieved in the depth maps is significant.

Instead of including directly into the general reconstruction of the scene the calculated shape prior as an entire model defined also as a volumetric structure, we propose to fuse depth maps enhanced with depth data obtained from the shape prior. In this way, neither the TSDF values of the general volumetric structure should be shifted by the TSDF values of the shape prior nor its weights should be modified with high values but the TSDF values will be adjusted according to the enhanced depth maps and the weights will be incremented by one unit, in a natural way.

The main contributions of this chapter are the next ones. First, a novel scheme (see fig. 1.1) for improving the dense reconstruction, got with a monocular camera or with a depth camera (moved by hand in a small scene), by integrating depth data from a shape prior associated to an object of interest. Second, a color management algorithm, similar to the TSDF management, that allows the system to store color information around the whole trajectory and get a model with full color data. Third, the experiments for quantifying the improvements in accuracy of the dense reconstruction achieved when shape prior

data is merged, specially in the region of the object of interest.

In section 6.2 similar works are presented. In section 6.3 the volumetric structure, the voxel update and the marching cubes algorithm are analysed in detail. The experiments for quantifying the accuracy of the dense reconstruction and the improvements achieved by merging the shape prior are described in section 6.4. Finally, the conclusions are presented in section 6.5.

6.2 RELATED WORK

KinectFusion [61], is the first system that builds a dense reconstruction of room-sized scenes and simultaneously localizes the camera at 30Hz, using a hand-held RGB-D camera from Microsoft, the Kinect 1.0, and one GPU. The model-to-frame tracking is based on coarse-to-fine iterative closest point ICP. It minimises the point to plane distance of points matched using projective association rather than feature extraction and matching. The three-level pyramid is built with vertices and normals maps from depth maps obtained from the sensor and from the reconstructed model. Once the camera pose is estimated, the depth maps captured with the sensor are fused and the model is updated. The truncated signed distance function, TSDF, is used for storing the depth data through the time, and ray casting for explicitly visualizing the model.

Kintinuous [96], based on their previous work [95], is an extension of KinectFusion for working in both room-sized environments and large scale paths. The cost function integrates the cost of the RGB-D frame-to-frame tracking of [87] (implemented on GPU), and the cost of the original model-to-frame ICP tracking of KinectFusion [61], in order to tackle poor performance when the camera points to a flat wall or corridors with no significant 3D features. They use another visual odometry system called FOVIS, [38], that relies on FAST feature correspondences in the RGB images, and that automatically switches when a planar environment appears. Unlike [61], Kintinuous makes use of both dense depth and photometric data. Moreover, it integrates color information in the reconstruction process, getting full colored models. The color volume is updated based on KinFu [76], but with real time surface coloring and rejection of unreliable color measurements by both not updating color close to depth discontinuities and using a weight that depends on the angle between the surface and the camera. It works at 30Hz using one GPU.

The system in [13] combines shape priors and live dense reconstruction using a monocular camera. Photo-consistency and a variational approach are used for building depth maps [62] that are fused for creating a dense representation of the scene, PTAM [44] for tracking the monocular camera, part-based object detector [28] for detecting, segmenting (giving also the posterior probabilities),

and estimating an initial pose of the object in the scene, GPLVM to represent shape priors, and Levenberg Marquadt for the iterative optimization over the pose, scale and shape of the object. The energy function, besides depending on the evaluation of the 3D points on the embedding function (depth-based energy function) as [73] and [49], depends on the matching between the 2D object segmentation, defined by the foreground and background, and the projected 3D SDF (image-based energy function). The optimal object is directly fused with the volumetric representation, obtaining full scaled 3D reconstructions with the known object segmented from the scene, and some of the unseen parts of the scene reconstructed (robustness to occlusions).

6.3 VOLUMETRIC FUSION

Depth data coming from either a depth camera, a data set, or built from images obtained with a monocular or stereo camera, must be represented in a global reference frame. A simple representation could be overlapping scans, for example, drawing local point clouds in a global coordinate system. A better representation is obtained with volumetric fusion of depth maps which generates a surface by computing the zero crossing of the function. The surface is represented as a set of triangles in 3D, obtained from the volumetric structure by applying the marching cubes algorithm. This 3D scene is rasterized in a 2D representation in order to visualize it on a screen. This representation is computationally more efficient, more realistic and allows a richer interaction with real or virtual objects, in robotics or augmented reality.

The constituent element of the volumetric structure is a voxel. We will denote a voxel as $\phi(\mathbf{X})$, where $\mathbf{X} \in R^3$ is the centroid of the voxel. It stores a truncated signed distance function value TSDF, a structure with RGB color data $\mathbf{C}$, and a weight W ,

$$\phi(\mathbf{X}) \mapsto [\text{TSDF}, \mathbf{C}, W]. \quad (6.1)$$

The TSDF, is a truncated version of the SDF, used in computer graphics for representing surface interfaces as zero, free space as positive values that increases with the distance to the closest surface and (possible) occupied space as a negative value. The structure for color data stores the RGB components of the image coming from the scene, in the range $[0,1]$ (in our work [20], we also work with color associated to the category of objects detected in the scene). The weight reflects the confidence of the TSDF value.

6.3.1 VOLUMETRIC STRUCTURE

The user can define the resolution of the reconstructed model through the size of the volumetric structure and the number of voxels in each dimension. The numeration of vertices and edges of an individual voxel is shown in figure 6.1(a). For carrying out marching cubes each vertex must have a TSDF value and a color $\mathbf{C}$. Since each voxel stores just one TSDF and $\mathbf{C}$, we create an extended structure based on the neighbouring voxels, as is shown in figure 6.1(b). The TSDF of voxel $\phi_\ell(\mathbf{X})$ is assigned to Vertex ℓ , for $\ell = 0 : 1 : 7$. The same assignment is done for the RGB color $\mathbf{C}$ of the voxel $\phi_\ell(\mathbf{X})$.

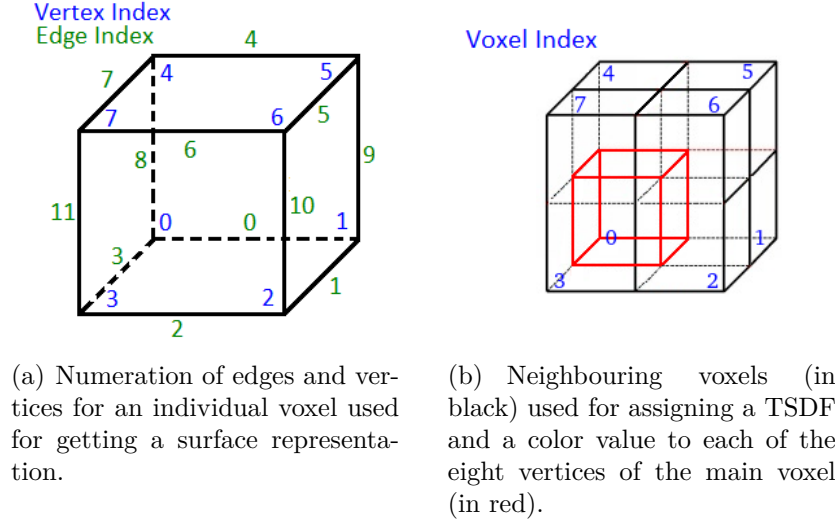


Figure 6.1: Notation for applying marching cubes to the volumetric structure.

6.3.2 VOXEL UPDATE

The depth measurement $\mathbf{D}(\boldsymbol{\mu})$, where $\boldsymbol{\mu} = (u, v)$ is a coordinate in the image plane, has an uncertainty of ς around the true value, so the measure is likely to be in the range $[D(\boldsymbol{\mu}) - \varsigma, D(\boldsymbol{\mu}) + \varsigma]$ over the ray that passes through the voxel $\phi_\ell(\mathbf{X})$ (see fig. 6.2). The free space is in the range $[0, D(\boldsymbol{\mu}) - \varsigma]$ and no information about the surface is obtained farther than $\mathbf{D}(\boldsymbol{\mu}) + \varsigma$ (occupied or free space). With TSDF, points that are in free space within a distance greater than ς from the closest surface will be truncated to a maximum value. Points that are far from the visible surface are not measured.

The TSDF value for a voxel ϕ with center $\mathbf{X}$, given the camera pose T_{wc_k} and a depth map from this pose $\mathbf{D}_{c_k}$, is computed as

$$\text{TSDF}(\mathbf{X}) = \Psi(\mathbf{D}_k(\boldsymbol{\mu}) - X_z) \quad (6.2)$$

where the subindex z corresponds to the z coordinate of the center $\mathbf{X}$ of a voxel ϕ (referenced to the camera frame) and $\mathbf{D}_k(\boldsymbol{\mu})$ is the depth of the 3D point

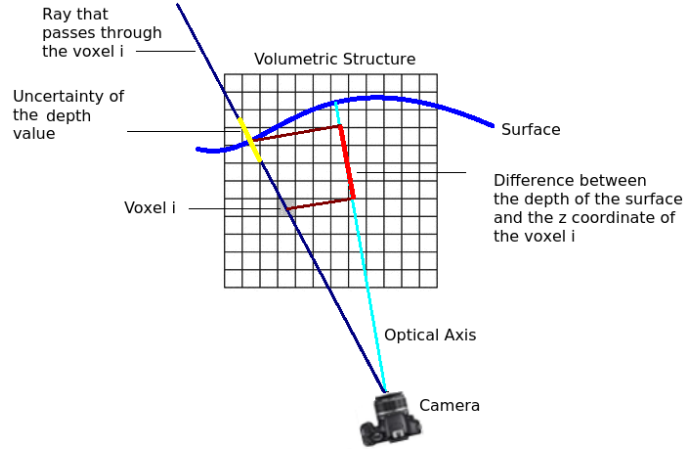


Figure 6.2: Ray that passes through the voxel i (dark blue), range of uncertainty of the measured depth (yellow), surface been scanned (light blue) and difference between the measured depth and the z -coordinate for a voxel i (in camera frame). The last value is used for computing the TSDF.

that lies in the surface and is intersected by the ray. The relation between $\mathbf{X}$ and $\boldsymbol{\mu}$ is,

$$\boldsymbol{\mu} = K\pi(\mathbf{X}), \quad (6.3)$$

where π is the function that computes normalized homogeneous coordinates $(x/z, y/z, 1)$ and K is the intrinsic camera matrix. $\Psi(\cdot)$ performs the SDF truncation

$$\Psi(\eta) = \begin{cases} \min(1, \frac{\eta}{\zeta})\text{sgn}(\eta) & \text{iff } \eta \geq 0 \\ \text{null} & \text{otherwise} \end{cases} \quad (6.4)$$

The fusion of depth maps in the volumetric structure corresponds to the weighted average of all the TSDF computed for each depth map, it means,

$$\text{TSDF}_k(\mathbf{X}) = \frac{W_{k-1}(\mathbf{X})\text{TSDF}_{k-1}(\mathbf{X}) + W(\mathbf{X})\text{TSDF}(\mathbf{X})}{W_{k-1}(\mathbf{X}) + W(\mathbf{X})} \quad (6.5)$$

The update in the color for each channel of the RGB data is an extended version of (6.5):

$$\mathbf{C}_k(\mathbf{X}) = \frac{W_{k-1}(\mathbf{X})\mathbf{C}_{k-1}(\mathbf{X}) + W(\mathbf{X})\mathbf{C}(\mathbf{X})}{W_{k-1}(\mathbf{X}) + W(\mathbf{X})} \quad (6.6)$$

where $\mathbf{C}(\mathbf{X}) = \mathbf{I}_k(\boldsymbol{\mu})$, and $\boldsymbol{\mu}$ was defined in (6.3). Next, the weight is updated as follows.

$$W_k(\mathbf{X}) = W_{k-1}(\mathbf{X}) + W(\mathbf{X}) \quad (6.7)$$

The weight provides information about the uncertainty of a TSDF value, so it can be set according to the depth and angle between the associated pixel ray direction and the surface normal in the local frame. However, we used simply $W(\mathbf{X}) = 1$, that averages the TSDF values, getting good results.

6.3.3 MARCHING CUBES

This algorithm creates a polygonal representation of an isosurface of a 3D scalar field. If one vertex is above the isosurface and an adjacent vertex is below, the isosurface cuts the edge between these two vertices. The position where it cuts the edge will be computed by interpolating the TSDF of the adjacent vertices:

$$\mathbf{X}_{\text{vert}} = \mathbf{X}_1 + m(\mathbf{X}_2 - \mathbf{X}_1) \quad (6.8)$$

where $\mathbf{X}_{\text{vert}}$ is the vertex of a triangle that represents the isosurface, $\mathbf{X}_1$ and $\mathbf{X}_2$ are the position of adjacent vertices of the cube, with local coordinate system, and

$$m = \frac{\text{isovalue} - \text{TSDF}_1}{\text{TSDF}_2 - \text{TSDF}_1}, \quad (6.9)$$

with isovalue = 0. The color is interpolated for each channel using the same ratio m ,

$$\mathbf{C}_{\text{vert}} = \mathbf{C}_1 + m(\mathbf{C}_2 - \mathbf{C}_1). \quad (6.10)$$

6.4 RESULTS

The first experiments are made for computing the accuracy of the reconstructed 3D model using synthetic data. In this case, the error in 3D geometry is computed by comparing a triangular mesh from a loaded model (ground truth) and the point cloud made up of vertices of the reconstructed model. For the second kind of experiments, we use raw depth maps (from a depth sensor or from a monocular camera) and enhanced ones with shape prior data. We quantify the effects that shape priors produce in the dense reconstruction and the improvement in accuracy in shape of the dense reconstruction associated to the object of interest.

6.4.1 RESULTS OF DENSE LOCALIZATION AND MAPPING

Our system for dense tracking and mapping, with color from the scene, runs on a laptop ASUS G551VW with a processor Intel Core i7-2.6GHz, 8GB of

RAM memory, a graphic processor NVIDIA GEFORCE 960M and Ubuntu 14.04 as operating system. We carried out experiments with data obtained from both a depth camera, the Kinect 1.0, and a synthetic dataset of A. Handa [36] which provides the ground truth in camera pose (perfect ray-traced images taken at 100 fps). OpenCV is employed for processing images, Eigen3 for some matrix operations, OpenGL for drawing the 3D model and for getting depth information from the model, Cuda 7.5 and Thrust library for improving the performance using commodity graphics hardware and OpenNI for working with the Kinect. The processing time in commodity graphics hardware for fusing depth maps and for the marching cubes algorithm was presented in chapter 2, in table 2.6.

ACCURACY IN 3D GEOMETRY

We use a 3D model that represents a desktop-like environment (ground truth), downloaded from TurboSquid [100], and the software CloudCompare [33], for performing the comparison.

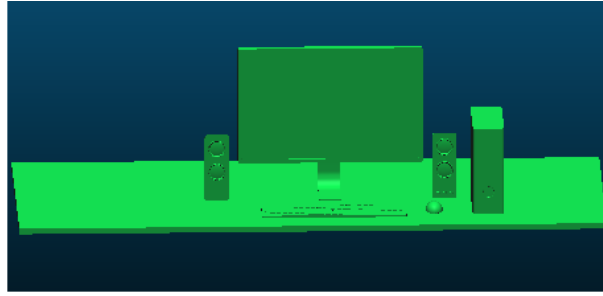
The 3D model is loaded in OpenGL, using just geometric data (fig. 6.3(a)). The virtual camera is located in such a way that circular and vertical scanning at different longitudes (changes in azimuth and elevation of 10°) are performed. The optical axis of the virtual camera is always pointing to the centroid of the loaded model. Depth images, obtained by reading the depth buffer from OpenGL, are fused into a volumetric structure. The vertices of the triangular mesh, computed with marching cubes, are used for the comparison (fig. 6.3(b)). Figure 6.4 shows the resulting comparison as a point cloud with color representing the error. Note that the greater errors are in the farther edges of the desk. Table 6.1 summarizes this comparison.

Table 6.1: Summary of accuracy in geometric data (taken from CloudCompare software [33]).

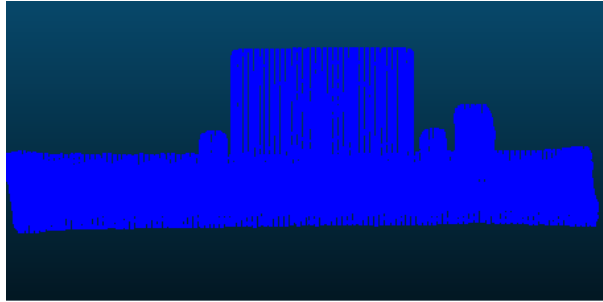
Quantity	Value[cm]
Max. error	6.213
Min. error	0
Mean error	1.4482
Standard dev.	1.1938

6.4.2 RESULTS USING SHAPE PRIOR DATA

The first two experiments quantify the effects in the dense reconstruction of merging the shape prior to depth maps taken with the depth sensor and to depth maps built with a monocular camera, respectively. The next two experiments quantify the accuracy of the dense reconstruction got with a monocular



(a) 3D Model of a desktop-like environment. Downloaded from TurboSquid [100].



(b) Point Cloud from the reconstruction.

Figure 6.3: Data used for quantifying the accuracy in geometry.

camera, without and with shape prior, respectively. The last two experiments also quantify the accuracy but just for the reconstruction associated to the object of interest, without and with shape prior, respectively. The integration of the synthetic depth map obtained from the shape prior was done in sequential way (depth map building followed by shape prior integration), as was described in section 5.3.

EFFECTS OF THE SHAPE PRIOR USING A DEPTH SENSOR

In this experiment we quantify the difference in geometry between the dense reconstruction got by fusing depth maps taken with the depth sensor and the one got by fusing depth maps that integrate depth data from the shape prior and from the depth sensor. We use 360 depth maps in each case and the camera trajectory delivered by ORB-SLAM2. This system is used for tracking since it is the most accurate one, as was registered in table 2.5. The assessment in distance between these two meshes were computed using the software CloudCompare [33]. The difference is unsigned (absolute value of the distance) since it allows us to identify easily the regions where the algorithm produces more changes in the reconstruction. We use two indicators for defining this distance; the root mean square distance RMSD and the upper threshold for 95% of all distances $D_{95\%}$.

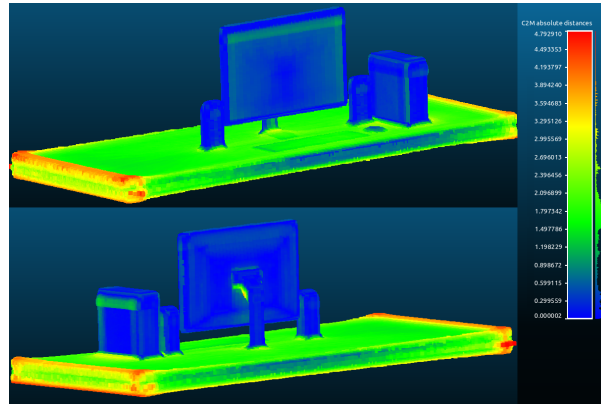


Figure 6.4: Point Cloud (front and back views) with color representing the distances between the point cloud and the reference model (scale in centimeters).

Figures 6.5(a) and 6.5(b) show the dense reconstruction using depth maps enhanced with shape priors and using only raw depth maps from the sensor (Kinect 1.0), respectively. The color is computed as the average of RGB values around the trajectory (see eq. 6.6). Figures 6.5(c) and 6.5(d) show the meshes without color in order to identify more details in geometry of the reconstructions using and not using shape priors, respectively. Note that the first one is smoother and tries to fill holes due to its inpainting property. Figures 6.5(e) (close view) and 6.5(f) (far view) present the reconstruction using shape priors and with color associated to the unsigned distance to the reconstruction built without shape prior. Note that higher distances are in borders (due to the smoothing effect), in the car (due to the shape prior) and in occluded regions (due to inpainting effect). The RMSD is 0.0067m and $D_{95\%}$ is 0.0135m. Figure 6.5(g) shows the histogram of distances. It shows that most of high distances are in a narrow region close to zero.

EFFECTS OF THE SHAPE PRIOR USING A MONOCULAR CAMERA

In this experiment we quantify the distance in geometry between the dense reconstruction got by fusing depth maps built with a monocular camera and the one got by fusing depth maps that integrate the depth data from the shape prior and the built depth maps. Each depth map is built using one reference image and 40 associated images. The reference images are separated by 60 frames. We use 7 built depth maps and the camera trajectory delivered by ORB-SLAM2. We proceed in the same way as in the previous case: we use CloudCompare [33], unsigned distance, the RMSD and $D_{95\%}$ for quantifying accuracy.

Figures 6.6(a) and 6.6(b) show the dense reconstruction using depth maps enhanced with shape prior and using only built depth maps, respectively. The color is computed as the average of RGB values around the trajectory. Figures

6.6(c) and 6.6(d) show the meshes without color in order to identify more details in the geometry of the reconstructions using and not using shape priors, respectively. Note that the first one is smoother and that the inpainting effect is not so evident like in the previous case. Figures 6.6(e) (close view) and 6.6(f) (far view) present the reconstruction using shape prior and with color associated to the unsigned distance to the reconstruction built without shape prior. Note that the higher distances are in noisy regions (not in borders like in the previous case) and in the car (due to the shape prior). The RMSD is 0.0047m and $D_{95\%}$ is 0.0074m. Figure 6.6(g) shows the histogram of distances. It shows that most of the distances are in a narrow region close to zero, even narrower than for the depth sensor case.

ACCURACY OF THE DENSE RECONSTRUCTION USING THE MONOCULAR CAMERA AND WITHOUT SHAPE PRIOR

This experiment was done in order to quantify how close the dense reconstruction got with the monocular camera (without shape prior data) is to the one got with the depth camera (with shape prior data) that is used as reference. Figure 6.7(a) presents the reference reconstruction. It was computed by fusing 360 depth maps from the depth sensor and enhanced with depth data obtained from the shape prior. Figure 6.7(b) shows the dense reconstruction got by fusion 7 depth maps built with the monocular camera and with the trajectory delivered by ORB-SLAM2. No shape prior data is integrated into these depth maps. In figs. 6.7(c) and 6.7(d) we can see the meshes without color for the reference reconstruction and the compared one, respectively. In these figures we can identify more details about the geometry. Finally, in figs 6.7(e) (close view) and 6.7(f) (far view) the reconstructions got with the monocular camera are presented, colored with the associated distances to the reference reconstruction. Note that larger errors exist in low textured and reflective regions, such as the walls, the border of the desk and the board. Figure 6.7(g) shows the histogram for errors between these models. Note that there is meaningful amount of pixels with errors between 0.10m and 0.35m. The mean is 0.0401m, the RMSD is 0.0698m, and $D_{95\%}$ is 0.1785m.

ACCURACY OF THE DENSE RECONSTRUCTION USING THE MONOCULAR CAMERA AND WITH SHAPE PRIOR

Next, we carry out the same process, with the same reference reconstruction (see figs. 6.8(a) and 6.8(c) for colored and uncolored reconstructions, respectively) but compared with the dense reconstruction got with monocular camera and enhanced with shape prior data (see figs. 6.8(b) and 6.8(d) for colored and uncolored reconstructions, respectively). It is evident the improvement got with the shape prior merged using a variational approach, specially for

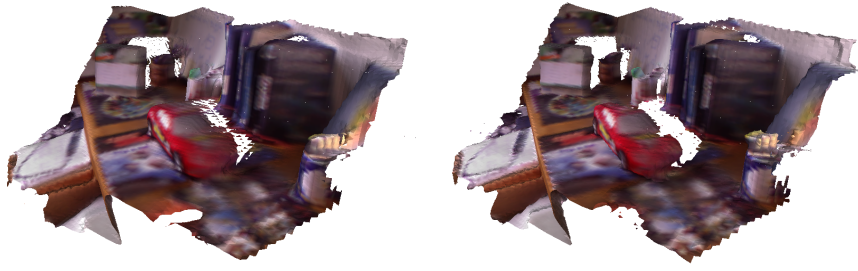
the smoothing effect. Figures 6.8(e) (close view) and 6.8(f) (far view) show the reconstructions got with monocular camera and shape prior, colored with data associated with the distances to the reference reconstruction. Note that these are very similar to the ones presented in figs. 6.7(e) and 6.7(f). The histogram of errors (see fig. 6.8(g)) is also similar to the one got in the previous experiment (see fig. 6.7(g)) since it is dominated by an offset in the direction of the optical axis that makes the reduction in error due to the smoothing effect small. Moreover, the changes due to the shape prior are done just for the car. This fact can be seen in the following values. The mean is 0.0392m, the RMSD is 0.0686m, and $D_{95\%}$ is 0.1739m.

ACCURACY OF THE RECONSTRUCTED CAR USING THE MONOCULAR CAMERA AND WITHOUT SHAPE PRIOR

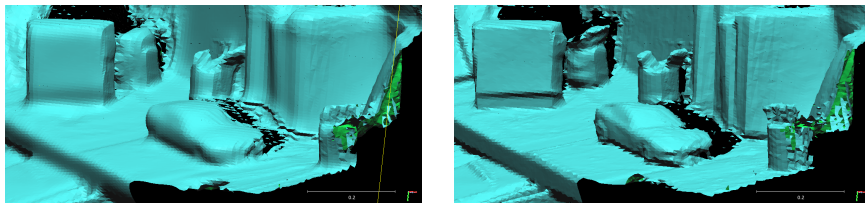
In this experiment we are going to quantify the accuracy in shape of the object of interest (the car). The region associated to the car in the reconstruction (see fig 6.9(a)), got with depth maps built with monocular camera and without shape prior, is manually segmented (see fig 6.9(b)). The reference reconstruction, like in the two previous comparisons, is the one built with depth sensor and shape prior data (see fig 6.9(c)). Since the segmented car and the reference reconstruction present a difference in pose, we align them as is shown in fig. 6.9(d) and 6.9(e). Now, we can estimate the distances between the segmented car and the reference reconstruction and apply color associated to these distances to the segmented car (see fig. 6.9(f)). The histogram of errors is shown in fig. 6.9(g). The mean error is 0.0064m, the RMSD is 0.0083m, and $D_{95\%}$ is 0.0167m.

ACCURACY OF THE RECONSTRUCTED CAR USING THE MONOCULAR CAMERA AND WITH SHAPE PRIOR

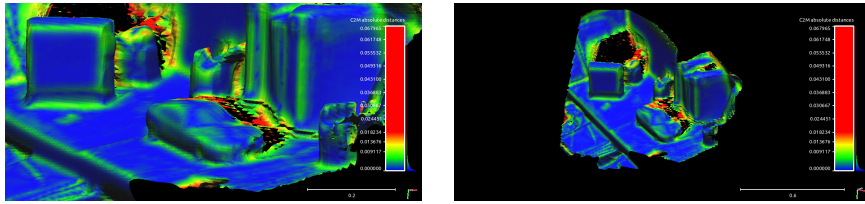
Now, we repeat the comparison process done in the previous experiment, with the same reference reconstruction, but in this case we segment the car from the reconstruction got with monocular camera and shape prior data (see fig. 6.10(a)). The resulting segmented car is shown in fig. 6.10(b). Note that it differs in pose with the reference model (see fig. 6.10(c)) so we align them (see figs. 6.10(d) and 6.10(e)). The segmented car is colored using the distances to the reference reconstruction, as is shown in fig. 6.10(f). Unlike the segmented car in the previous experiment, this car is smooth and with smaller amount of points with errors over 0.013m, as can be verified in the histogram of fig. 6.11 (blue line). The mean error is 0.0053m, the RMSD is 0.0067m, and $D_{95\%}$ is 0.0133m.



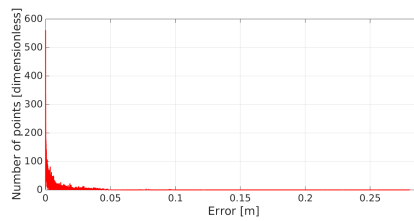
(a) With shape prior, with color, depth sensor case. (b) Without shape prior, with color, depth sensor case.



(c) With shape prior, without color, depth sensor case. (d) Without shape prior, without color, depth sensor case.

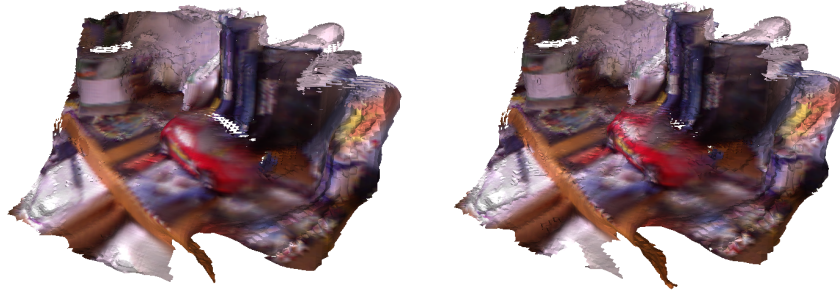


(e) With shape prior, using color associated to the distances between the two models (close view). (f) With shape prior, using color associated to the distances between the two models (far view).

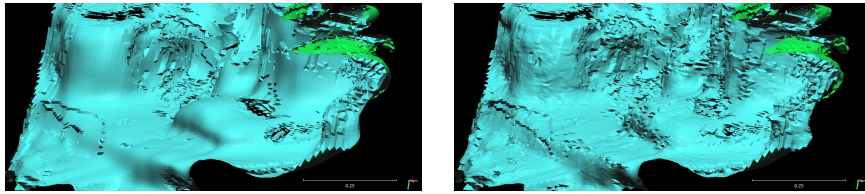


(g) Histogram of distances, depth sensor case.

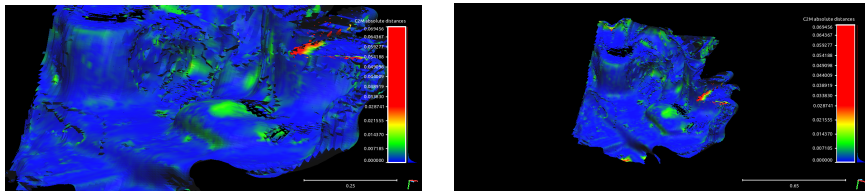
Figure 6.5: Comparison between dense reconstructions got with a depth sensor using and not using shape prior. (a) Colored reconstruction using shape prior, (b) Colored reconstruction without shape prior, (c) Uncolored reconstruction using shape prior, (d) Uncolored reconstruction without shape prior, (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.



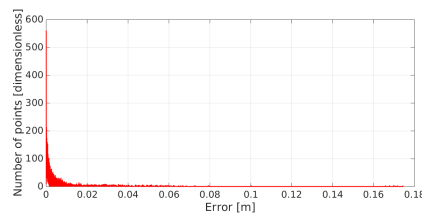
(a) With shape prior, with color, monocular case. (b) Without shape prior, with color, monocular case.



(c) With shape prior, without color, monocular case. (d) Without shape prior, without color, monocular case.

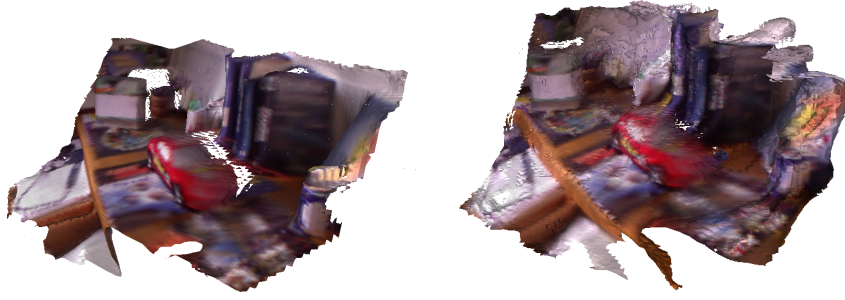


(e) With shape prior using color associated to the distances between the two models (close view). (f) With shape prior using color associated to the distances between the two models (far view).

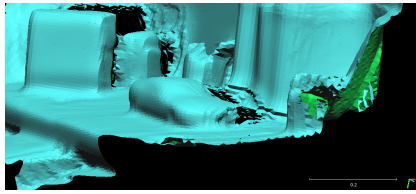


(g) Histogram of distances, monocular case.

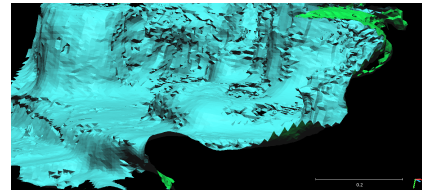
Figure 6.6: Comparison between dense reconstructions got with a monocular camera, using and not using shape prior. (a) Colored reconstruction using shape prior, (b) Colored reconstruction without shape prior, (c) Uncolored reconstruction using shape prior, (d) Uncolored reconstruction without shape prior, (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.



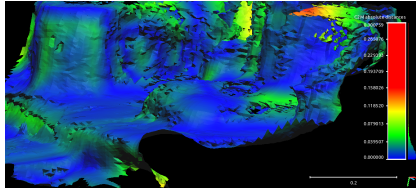
(a) With shape prior, with color, depth sensor case. (b) Without shape prior, with color, monocular case.



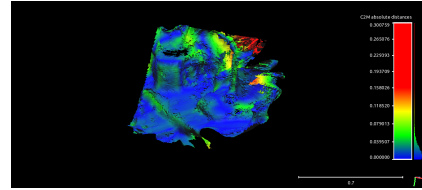
(c) With shape prior, without color, depth sensor case.



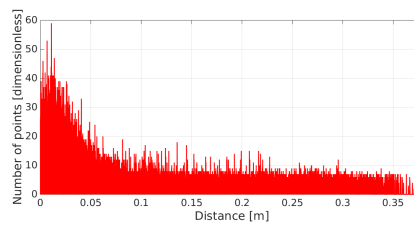
(d) Without shape prior, without color, monocular case.



(e) Without shape prior using color associated to the distances between the two models (close view).

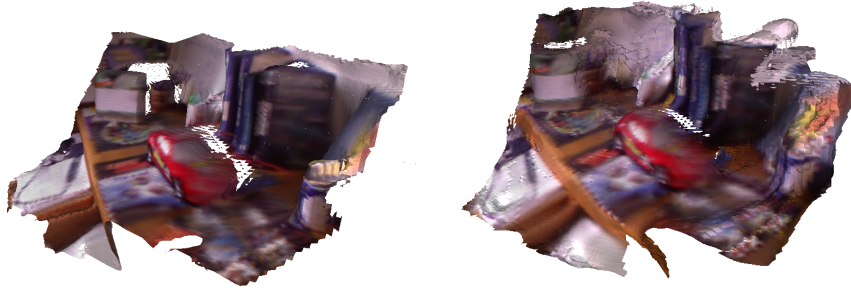


(f) Without shape prior using color associated to the distances between the two models (far view).

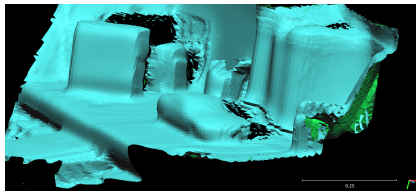


(g) Histogram of distances.

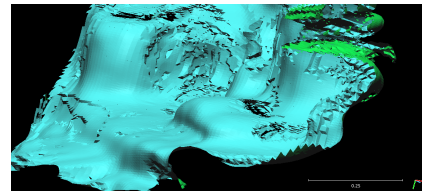
Figure 6.7: Comparison between the reference reconstruction (depth sensor and shape prior) and the dense reconstruction got with the monocular camera and without shape prior. (a) Colored reconstruction used as reference, (b) Colored reconstruction got with monocular camera and without shape prior, (c) Uncolored reconstruction (reference), (d) Uncolored reconstruction (monocular and without shape prior), (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view) and (g) histogram of distances.



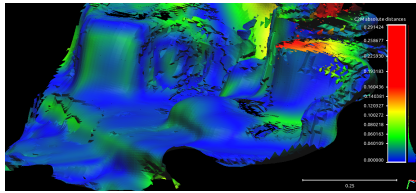
(a) With shape prior, with color, depth sensor case. (b) With shape prior, with color, monocular case.



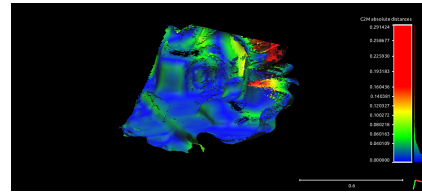
(c) With shape prior, without color, depth sensor case.



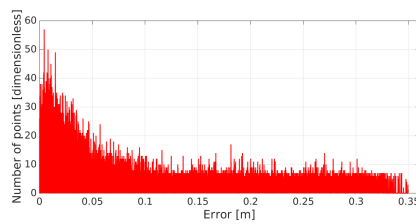
(d) With shape prior, without color, monocular case.



(e) With shape prior using color associated to the distances between the two models (close view).

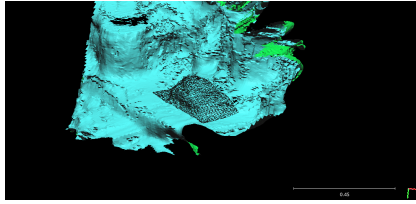


(f) With shape prior using color associated to the distances between the two models (far view).

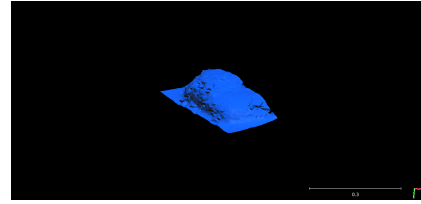


(g) Histogram of distances.

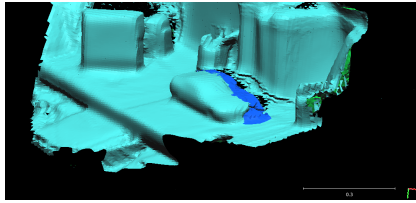
Figure 6.8: Comparison between the reference reconstruction (depth sensor and shape prior) and the dense reconstruction got with the monocular camera and with shape prior. (a) Colored reconstruction used as reference, (b) Colored reconstruction got with monocular camera and with shape prior, (c) Uncolored reconstruction (reference), (d) Uncolored reconstruction (monocular and with shape prior), (e) Colored reconstruction using distances between models, (f) Colored reconstruction using distances between models (far view), and (g) histogram of distances.



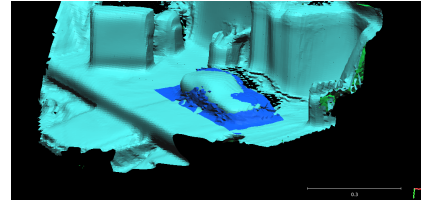
(a) Manual segmentation from the reconstruction got with monocular camera and no shape prior.



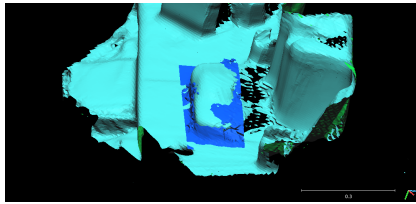
(b) Segmented car.



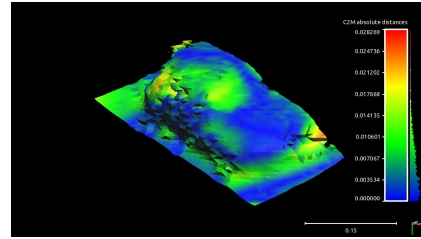
(c) Reference reconstruction (depth sensor and shape prior) and segmented car.



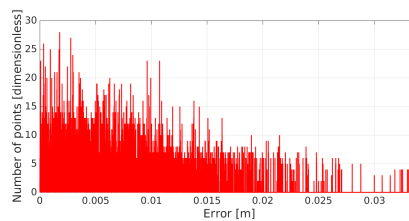
(d) Segmented car aligned to the reference reconstruction (view 1).



(e) Segmented car aligned to the reference reconstruction (view 2).

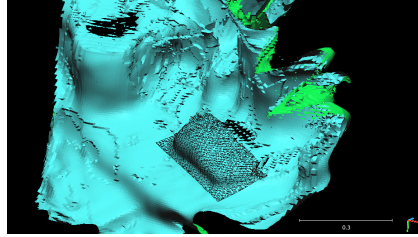


(f) Segmented car using color associated to the distances with the reference reconstruction.

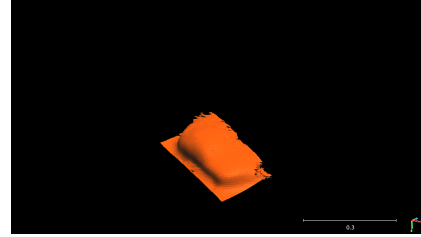


(g) Histogram of distances.

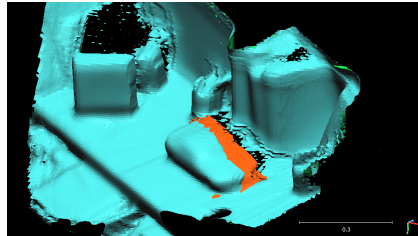
Figure 6.9: Comparison between segmented car (from dense reconstruction got with monocular camera and no shape prior) and reference reconstruction (with depth sensor and shape prior). (a) Region segmented from the reconstruction got with monocular camera and no shape prior, (b) Resulting segmented car, (c) Reference reconstruction and segmented car, (d) Segmented car aligned to the reference reconstruction, (e) Another view of aligned car, (f) segmented car colored with data associated to the distances and (g) histogram of distances.



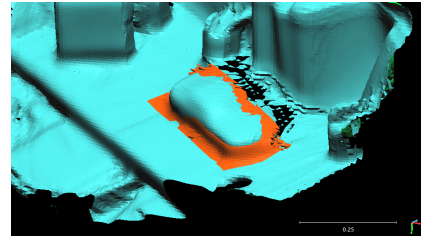
(a) Manual segmentation from the dense reconstruction got with monocular camera and shape prior.



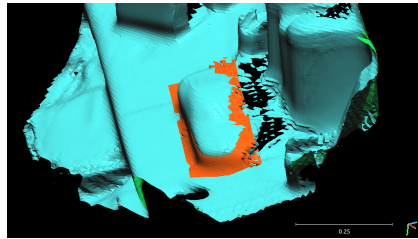
(b) Segmented car.



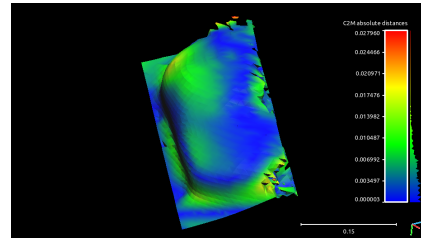
(c) Reference reconstruction (depth sensor and shape prior) and segmented car.



(d) Segmented car aligned to the reference reconstruction (view 1).



(e) Segmented car aligned to the reference reconstruction (view 2).



(f) Segmented car using color associated to the distances with the reference reconstruction.

Figure 6.10: Comparison between segmented car (from dense reconstruction got with monocular camera and shape prior) and reference reconstruction (with depth sensor and shape prior). (a) Region segmented from the reconstruction got with monocular camera and shape prior, (b) Resulting segmented car, (c) Reference reconstruction and segmented car, (d) Segmented car aligned to the reference reconstruction, (e) Another view of segmented car aligned to the reference car and (f) segmented car colored with data associated to the distances to the reference reconstruction.

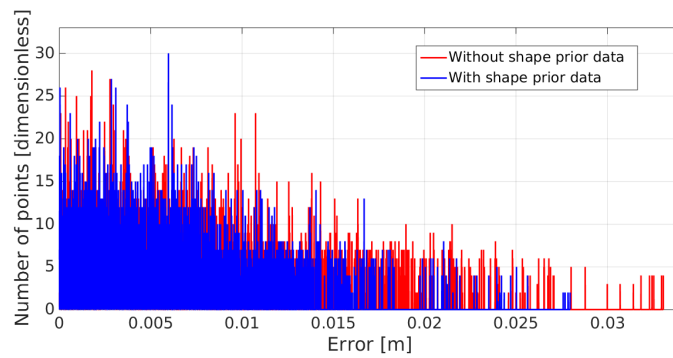


Figure 6.11: Comparison between histograms of distances between the reference reconstruction and the aligned and segmented car 1. from reconstruction with monocular camera and NO shape prior (red) and 2. from reconstruction with monocular camera and shape prior (blue).

6.5 CONCLUSIONS

In this chapter we have fused raw depth maps (obtained from a depth sensor or built with a monocular camera) into a volumetric structure and represented explicitly as a continuous mesh using marching cubes and an average-based process for updating the color of the reconstruction. The results have been satisfactory. First, the fusion and marching cubes algorithms, produce high accuracy estimations in 3D reconstruction (the accuracy in camera tracking was analysed in chapter 2). The accuracy of the models is high: the comparison done for the loaded model of 2m long shows that the mean error in geometry is 1.4482cm. Second, averaging the RGB intensities through time for updating the color of the reconstruction results in a realistic 3D model with color coming from the scene.

We also have enhanced the raw depth maps with depth data obtained from the shape prior, using a variational technique that besides merging both depth sources, makes inpainting and denosing. These depth maps were fused and the resulting dense reconstruction was compared with the one got without shape prior data.

The algorithm for merging two depth sources produces an evident denoising effect in the dense reconstruction when the depth sensor is used, specially in the borders of the objects, and a merging effect in the area of the car due to the shape prior data. When depth maps built with a monocular camera and enhanced with shape prior data are fused, the reconstruction presents considerable changes in the area of the car (merging effect) and a smoothing effect in noisy regions (not in borders).

When the dense reconstruction got with a monocular camera is compared with the reference reconstruction (depth sensor and shape prior) we can see regions of low accuracy that correspond to regions in the intensity images where there are low-textured and very reflective objects. If the shape prior is merged using a variational technique, it adjusts the shape of the car, making it closer to the real shape. The smoothing effect is evident but the improvement in accuracy is not significant since there is an offset in pose between the reconstructions, so a more local comparison is needed. The comparison of the reconstruction only associated to the car and considering the offset in pose shows that the improvement in accuracy of the shape of the object of interest (the car) using shape prior is significant: the RMSD and the $D_{95\%}$ diminish from 0.0083m and 0.0167m to 0.0067m and 0.0133m, respectively.

CHAPTER 7

CONCLUSIONS

In this doctoral thesis we have tackled the problem of low accuracy in some regions of depth maps built with a monocular camera, when it is moved by hand around a room-sized scene. We found in chapter 3 that these regions correspond to surfaces with low texture, low structure, very reflective or translucent, where photo-consistency is not fulfilled and the minimization of the photometric error is dominated by noise. These problems can be faced to some extent with a regularisation term in a variational framework, like is done in [62]. However, the accuracy in problematic regions remains low. We proposed to use object shape priors for improving the accuracy of depth maps, especially in the area of the object of interest. In chapter 4 we presented the methodology for modelling the variation in shape of the object of interest, for computing an initial estimation in pose and scale, and for refining this initial estimation with respect to pose, scale and shape. We proposed to include the shape prior as a synthetic depth map into the built depth map instead of modifying a point cloud like in [46] or the volumetric structure like in [13]. In this sense, we propose in chapter 5 a novel formulation of an objective function, based on the fusion of color aerial images [49], that integrates the synthetic depth map obtained from the shape prior (optimal model) and the built depth map. Moreover, in that chapter we presented a variant in the objective function that simultaneously builds a depth map and integrates the synthetic depth map estimated from the shape prior. The results showed a significant improvement in the accuracy associated to the area of the object of interest together with a denoising and inpainting effect in the whole image. These enhanced depth maps were fused into a volumetric structure and an explicit representation was computed with marching cubes. The comparison carried out in chapter 6 with a reference reconstruction (got with a depth camera and integrating shape priors) shows that the reconstruction got with depth maps built with a monocular camera and shape priors has a higher accuracy than the one obtained without shape priors, specially in the 3D reconstruction associated to the object of interest. Moreover, the denoising effect in the reconstruction for the monocular camera is evident.

We believe that the integration of scene shape priors like in [69], [11], and

object shape priors like in this work could produce depth maps built with a monocular camera with high accuracy in the scene (walls, floor and ceiling) and also in objects of interest (like cars in our work). This integration can be done in a straight way by modifying the objective function in a variational approach. Moreover, we could increase the number of objects of interest by modelling their variation in shape, using 3D models available in internet. This system would allow to segment objects and the structure of the scene (3D segmentation), which could be an important step towards a more complex object and scene understanding, useful in mobile robotics, wearable devices and augmented reality.

These techniques for improving depth maps built with a monocular camera can also be used for enhancing depth maps obtained from a depth sensor since they have noise and missing data due to dark, translucent, very reflective objects, occlusions and objects out of range. This claim was proved in chapter 5, where depth maps captured with the kinect 1.0 were enhanced by including information of the shape prior, besides of denoising and inpainting. Moreover, we carried out experiments with a synthetic depth map with added noise and missing data, showing that the resulting depth map has higher accuracy and is smoother than the one obtained with an outstanding system for depth recovery [67].

Additionally, we presented in chapter 2 some contributions in two approaches for localizing a camera (monocular and depth camera) using dense information. The frame-to-frame approach for localizing a monocular camera uses keyframes selected with a spatial criteria, an initial estimation in camera pose based on 3D-2D visual odometry techniques and a refinement process based on image alignment with a coarse-to-fine scheme. The experiments showed that the algorithm has high accuracy even with blurred images. The other approach, the model-to-frame one, localizes a depth camera using ICP with a coarse-to-fine scheme for aligning two point clouds coming from the model and from the observed depth map. In this work we presented a variant that uses marching cubes and rasterization instead of raycasting like in [61]. Moreover we presented a method for managing the color of the scene with an average-based approach, similar to the one used for updating the TSDF values of the volumetric structure that represents the scene.

Finally, we have to mention the efforts that we did in this work for implementing most of the algorithms in commodity graphics hardware since one of the initial requirements was the performance in real time or near it.

7.1 RESULTS, CONTRIBUTIONS AND FUTURE WORK

An algorithm for tracking a monocular camera using dense information and a frame-to-frame approach. Given a depth map and an intensity image for a reference frame (keyframe) and the current intensity image we can estimate the relative transformation by aligning a synthetic intensity image and the current image. The synthetic image is estimated by back projecting the depth map (for computing a point cloud) and then, projecting it to the image plane of the current image, considering the initial estimation in camera pose got with a 3D-2D visual odometry technique. The algorithm defines keyframes according to spacial constraints and uses the closest one to the current image as reference frame. Since the objective function is linearised, a coarse-to-fine scheme is employed with a pyramid of three levels, allowing the system to copy with faster camera motions and to improve the convergence with Levenberg-Marquardt. The algorithm was implemented in commodity graphics hardware for speeding up its performance, achieving to process 10 fps. The experiments were done with synthetic data [36] for criss-cross motion and with images with blur and noise, showing satisfactory performance. The main contributions of this work are:

- The definition of spacial criteria for initializing new keyframes in order to overcome the computational payload of dense techniques for camera tracking.
- A two-steps process to localize the camera with respect to the closest keyframe: 3D to 2D visual odometry followed by the image alignment. It results in high accuracy estimations even when using blurred input images.

For future work we propose to integrate a module for loop detection and bundle adjustment over keyframes in order to be more robust to long and complex trajectories. Moreover, we propose to build an initial depth map with a monocular camera and use it for tracking the camera in the next steps and simultaneously for building the depth maps of the next keyframes. For this task we can integrate in straight way the frame-to-frame tracking and the algorithm for building depth maps with a monocular camera.

An algorithm for building depth maps using a monocular camera.

Given the camera pose for a set of images that share common elements of the scene, we gather photometric error from these poses. We use a variational technique where this photometric error defines the data term. The regularisation term constraints the solution to be smooth but preserving discontinuities. A third term, that decouples the non-convex data term and the convex regularisation term is used, with the primal-dual algorithm and exhaustive search for solving the optimization problem. The implementation on GPU takes 1.0625s for estimating the initial depth map and 3.6904s for refining the depth map,

which is appropriate for monocular systems that work with keyframes. This algorithm is based on DTAM [62] and its main contribution is

- Put in evidence the problem that we tackle in this doctoral thesis: although the regularisation term improves the quality of the built depth map, its accuracy is low in regions with low texture, in regions with changes in light conditions or with low structure.

We found that an efficient solution is the use of object shape priors in a variational approach. In this sense, we proposed to integrate depth data from an object shape prior using a variational technique as is explained next.

An algorithm for simultaneously tracking a depth camera and building a 3D model of the scene, using dense information and a model-to-frame approach. Given depth maps at high frame rates we can integrate them into a volumetric structure, estimate the relative transformation by aligning a point cloud coming from the model and a point cloud coming from the current depth measurement, and visualize the dense reconstruction with color of the scene. The point cloud alignment is done with Iterative closest point in a coarse-to-fine scheme, with a pyramid of four levels, since the objective function is linearised. The implementation on commodity graphics hardware runs at 10 fps with depth maps of 640x480 pixels, a volumetric structure of 128x128x128 voxels and color management. The experiments were done with synthetic data [36], with a public dataset of TUM [91] and with our own depth maps from the kinect 1.0, achieving satisfactory performance and accuracy in camera pose and geometry of the reconstruction. This work is based on KinectFusion [61] and integrates a module for scene labeling that is described in our paper [20]. The main contributions of this work are:

- A modular system for dense tracking and mapping with a model-to-frame approach that couples an object detection module for labeling the scene and that can easily be expanded by integrating new modules for manipulating objects, for navigation of unmanned aerial vehicles or for augmented reality.
- An algorithm for propagating to the model information of detected objects and for managing multiple instances of an object class, computing its location, a 3D bounding box, and its probability.
- A color management algorithm, similar to the truncated signed distance function (TSDF) management, that allows the system to store color information around the whole trajectory and get a model with full color data.

For future work we propose, like with the algorithm for camera tracking with frame-to-frame approach, to integrate a module for loop detection and bundle adjustment over keyframes or graph optimization in order to be able to

work in more complex environments. Moreover, we propose to integrate color information in the tracking process as is done in [98]. We also propose to use the model-to-frame approach with the monocular camera: instead of align the current image with the synthetic image computed with the depth map of the closest keyframe, align the current image with the synthetic image computed with the updated reconstruction (by reading the depth buffer).

An algorithm for estimating an optimal 3D model, for integrating depth data of the optimal model into the built depth map and for fusing the enhanced depth maps. The optimal 3D model is estimated by aligning a 3D level-set embedding function of a model hypothesis and the point cloud associated to the object of interest. The level sets are computed using a set of 3D models aligned with respect to a base model. Then, they are compressed and a dimensionality reduction is done with GPLVM. We compute an initial estimation of pose, scale and shape. The initial position is based on the centroid of the point cloud associated to the object. The orientation is computed considering that the object is over a supporting plane. The scale is computed based on the average distance of each point to the centroid. The initial latent variable is the one associated to the base model. The refinement is done with Levenberg-Marquardt, initially alternating between pose and scale and after that, adding shape optimization (searching the latent variable in the latent space). A synthetic depth map is computed by reading the depth buffer from the current camera pose. The depth map obtained from the shape prior and from the sensor are integrated using a variational approach that besides merging both depth sources, makes denoising and inpainting. The energy functional is defined by a data term that measures the discrepancy between the depth sources and the sought solution, and the regularisation term that smooths the solution. It is solved using the primal-dual algorithm. Finally, the enhanced depth maps are fused and a 3D reconstruction with higher accuracy is obtained, especially in the region of the object of interest. The depth merging with variational techniques was implemented on commodity graphics hardware, taking 100 iterations 1.5694s. The main contributions of this work are

- An alternating method for computing the optimal model. An initial estimation in object pose and scale, based on the centroid, the supporting plane and the average distance of the points to the centroid, is refined in alternating way. Once the pose and scale are refined, the initial shape is found with exhaustive search and posteriorly refined, alternating pose, scale and shape.
- A novel formulation of an objective function with a variational approach for merging sensor data and shape prior data, with and additional effect of denoising and inpainting.
- A variant in the objective function for performing in one algorithm both the module for dense mapping with monocular camera and the module

for depth merging using a shape prior.

- The experiments carried out in order to quantify accuracy of the pose, shape and scale optimization, the completeness of the inpainted depth maps and the improvements in depth accuracy.

For future work we leave the implementation of the the algorithm for estimating the optimal model in commodity graphics hardware. Moreover we would like to create latent spaces for a set of objects that are common in a particular scene. Finally, for future work we propose to use object shape priors and scene shape priors together.

7.2 PAPERS, ACADEMIC EVENTS AND SHORT-TERM RESEARCH ABROAD

Draft papers.

- A. Díaz, L. Paz, P. Piniés and E. Caicedo. "Depth Map Denoising and Inpainting using Object Shape Priors", 2018.
- A. Díaz, L. Paz, P. Piniés and E. Caicedo. "Depth Map Building and Enhancement using a Monocular Camera, Shape Priors and variational Methods", 2018.

Papers.

- A. Díaz, L. Paz, P. Piniés and E. Caicedo. "Dense Tracking, Mapping and Scene Labeling using a Depth Camera", Revista Facultad de Ingeniería. Universidad de Antioquia, No. 86, pp. 54-69, March 2018. [20].

Chapter of book.

- A. Díaz, L. Paz, P. Piniés and E. Caicedo. "Dense Tracking with Range Cameras Using Key Frames", in Communication in Computer and Information Science (CCIS). Robotics. Extended version of paper presented in LARS 2015/SBR 2015, Uberlandia, Brasil. ISSN: 1865-0929, Springer Publishing Company, 2016, pp. 20-38. [22].

International Event.

- A. Díaz, L. Paz, P. Piniés and E. Caicedo. "Dense Localization of a Monocular Camera using Keyframes", in XII Latin American Robotics Symposium (LARS) and III Brazilian Robotics Symposium (BRS), Uberlandia, Brasil, 2015, pp. 13-18. [18].

National Event.

- A. Díaz, J. Caicedo and E. Caicedo. "Augmented reality without fiducial markers", in 20th Symposium of Signal Processing, Images and Computer Vision (STSIVA), Bogotá, Colombia, 2015, pp. 1-7. [19].

Visiting Researcher.

- Visiting researcher during five months in the mobile robotics group MRG of Oxford University. Oxford. May 2016 - October 2016.

Others.

- A tutorship given to an undergraduate student in his final project: "Desarrollo de un Sistema para Digitalización de Ejemplares de Artesanías (Implementation of a system for digitalizing handicrafts)". The scanner for digitalizing the handicrafts with a Kinect 1.0 together with the algorithm for dense tracking and mapping with a depth sensor moved by hand are products that will be used by craftsmen of Nariño. It is an effort for incorporating new technology in our region, as part of the requirements of the Fundación CEIBA, which sponsored this doctoral thesis.

BIBLIOGRAPHY

- [1] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.
- [2] Simon Baker and Iain Matthews. Lucas-Kanade 20 Years On: A Unifying Framework: Part 1. *International Journal of Computer Vision (IJCV)*, 3(56):221–255, 2004.
- [3] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 1264–1271, Washington, DC, USA, 2013. IEEE Computer Society.
- [4] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems Conference (RSS)*, June 2013.
- [5] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, May 2011.
- [6] M. Chaouch and A. Verroust-Blondet. A novel method for alignment of 3d models. In *2008 IEEE International Conference on Shape Modeling and Applications*, pages 187–195, June 2008.
- [7] C. Chen, R. Jafari, and N. Kehtarnavaz. Improving human action recognition using fusion of depth camera and inertial sensors. *IEEE Transactions on Human-Machine Systems*, 45(1):51–61, Feb 2015.
- [8] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, October 2008.
- [9] Alejo Concha and Javier Civera. RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system, 2017.

- [10] Alejo Concha, Muhammad Wajahat Hussain, Luis Montano, and Javier Civera. Manhattan and piecewise-planar constraints for dense monocular mapping. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.
- [11] Alejo Concha, Wajahat Hussain, Luis Montano, and Javier Civera. Incorporating scene priors to dense monocular mapping. *Autonomous Robots*, 39(3):279–292, Oct 2015.
- [12] Michael Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, Department of Engineering Science, University of Oxford, Oxford, 1997.
- [13] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid. Dense reconstruction using 3d object shape priors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1288–1295, June 2013.
- [14] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. Int. Conf. Computer Vision*, Nice, Oct 2003.
- [15] Andrew J. Davison and David W. Murray. Simultaneous localization and map-building using active vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7):865–880, 2002.
- [16] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Oliver Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.
- [17] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. J. Robotics Research*, 25(12), December 2006.
- [18] A. Díaz, E. Caicedo, L. Paz, and P. Piniés. Dense localization of a monocular camera using keyframes. In *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, pages 13–18, Oct 2015.
- [19] A. Díaz, S. Caicedo, and E. Caicedo. Augmented reality without fiducial markers. In *2015 20th Symposium on Signal Processing, Images and Computer Vision (STSIVA)*, pages 1–7, Sept 2015.
- [20] A. Díaz, L. Paz, P. Piniés, and E. Caicedo. Dense tracking, mapping and scene labeling using a depth camera. *Revista Facultad de Ingeniería*, (86):54–69, March 2018.
- [21] Andrés Díaz, Lina Paz, Eduardo Caicedo, and Pedro Piniés. Simultaneous localization of a monocular camera and mapping of the environment in real time. *DYNA*, 81:7 – 15, 02 2014.

- [22] Andrés Díaz, Lina Paz, Eduardo Caicedo, and Pedro Piniés. Dense tracking with range cameras using key frames. In Fernando Santos Osório and Rogério Sales Gonçalves, editors, *Robotics*, pages 20–38. Springer International Publishing, Cham, 2016.
- [23] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.
- [24] Ethan Eade and Tom Drummond. Scalable monocular slam. In *in IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 469–476. IEEE Computer Society, 2006.
- [25] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Europeion an Conference on Computer Vis(ECCV)*, September 2014.
- [26] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1449–1456, Dec 2013.
- [27] R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: Insights into sparsification. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, August 2005.
- [28] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010.
- [29] J. Feng, B. Price, S. Cohen, and S. F. Chang. Interactive segmentation on rgb-d images via cue selection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 156–164, June 2016.
- [30] J. Folkesson and HI Christensen. Graphical SLAM for outdoor applications. *Journal of Field Robotics*, 23(1):51–70, 2006.
- [31] Christian Forster, Simon Lynen, Laurent Kneip, and Davide Scaramuzza. Collaborative monocular slam with multiple micro aerial vehicles. In *IROS*, pages 3962–3970, 2013.
- [32] H. Fu, D. Xu, and S. Lin. Object-based multiple foreground segmentation in rgb-d video. *IEEE Transactions on Image Processing*, 26(3):1418–1427, March 2017.
- [33] D. Girardeau-Montaut. Cloudcompare (version 2.9) [gpl software], 2017. Accessed 2018-04-03.

- [34] J. Guivant, E. Nebot, and S. Baiker. Localization and map building using laser range sensors in outdoor applications. *Journal of Robotics Systems*, (17):565–583, 2000.
- [35] J. S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation CIRA*, pages 318–325, 1999.
- [36] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. Real-time camera tracking: When is high frame-rate best? In *Proc. of the European Conference on Computer Vision(ECCV)*, Oct. 2012.
- [37] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [38] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *In Proc. of the Intl. Sym. of Robot. Research*, 2011.
- [39] I.K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Proc. of the 9th Int. Conf. on Computer Vision*, pages 946–951, 2003.
- [40] Christian Kerl and Daniel Cremers. Robust Odometry Estimation for RGB-D Cameras. 2012.
- [41] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754, 2013.
- [42] Syed Ali Khayam. The discrete cosine transform (dct): Theory and application., 2003. Department of electrical computing engineering.
- [43] Ayoung Kim and R.M. Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *Robotics, IEEE Transactions on*, 29(3):719–733, June 2013.
- [44] G. Klein and David W. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007.
- [45] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV’08)*, pages 802–815, Marseille, October 2008.
- [46] Stefan Kluckner, Thomas Pock, and Horst Bischof. *Exploiting Redundancy for Aerial Image Fusion Using Convex Optimization*, pages 303–312. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

- [47] J. Knight, A. Davison, and I. Reid. Towards constant time SLAM using postponement. In *Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems*, pages 406–412, Maui, Hawaii, 2001.
- [48] K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *Transactions on Robotics*, 24(5):1066–1077, October 2008.
- [49] Jens Krenzin and Olaf Hellwich. *Reduction of Point Cloud Artifacts Using Shape Priors Estimated with the Gaussian Process Latent Variable Model*, pages 273–284. Springer International Publishing, Cham, 2016.
- [50] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816, December 2005.
- [51] J. J. Leonard, R. Carpenter, and H. J. S. Feder. Stochastic mapping using forward look sonar. *Robotica*, 19:467–480, 2001.
- [52] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgb-d cameras. In *2013 IEEE International Conference on Computer Vision*, pages 1417–1424, Dec 2013.
- [53] Steven Lovegrove and Andrew J Davison. Real-Time Spherical Mosaicing using Whole Image Alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.
- [54] M.J. Milford, G.F. Wyeth, and D. Prasser. Ratslam: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 403–408 Vol.1, April 2004.
- [55] Martin F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *NEURAL NETWORKS*, 6(4):525–533, 1993.
- [56] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.
- [57] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Int. Joint Conf. Artificial Intelligence*, 2003.
- [58] J. M. M. Montiel, Javier Civera, and Andrew J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proc. Robotics: Science and Systems*, Philadelphia, USA, August 2006.

- [59] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [60] Richard a. Newcombe and Andrew J. Davison. Live dense reconstruction with a single moving camera. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1498–1505, June 2010.
- [61] Richard a. Newcombe, Andrew J. Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, October 2011.
- [62] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. DTAM: Dense Tracking and Mapping in Real-Time. (Department of Computing, Imperial College London, UK), 2012.
- [63] P. Newman, J. Leonard, J. D. Tardós, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1802–1809. IEEE, 2002.
- [64] B. Ni, P. Moulin, X. Yang, and S. Yan. Motion part regularization: Improving action recognition via trajectory group selection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3698–3706, June 2015.
- [65] Lina M. Paz, P. Piniés, J.D. Tardós, and J. Neira. Large-scale 6-dof slam with stereo-in-hand. *Robotics, IEEE Transactions on*, 24(5):946–957, Oct 2008.
- [66] Lina M. Paz, Juan D. Tardós, and José Neira. Divide and Conquer: EKF SLAM in $O(n)$. *Accepted in Transactions on Robotics (in Print)*, 24(5), October 2008.
- [67] Said Pertuz and Joni Kamarainen. Region-based depth recovery for highly sparse depth maps. 2017.
- [68] Juan Pineda. A parallel algorithm for polygon rasterization. *SIGGRAPH Comput. Graph.*, 22(4):17–20, June 1988.
- [69] Pedro Piniés, Lina Maria Paz, and Paul Newman. Dense Mono Reconstruction: Living with the Pain of the Plain Plane. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015.

- [70] Pedro Pinies, Lina María Paz, and Paul Newman. Too much tv is bad: Dense reconstruction from sparse laser with non-convex regularisation. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 135–142, 2015.
- [71] Pedro Piniés and Juan D. Tardós. Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision. *Accepted in Transactions on Robotics (in print)*., 24(5), October 2008.
- [72] Victor Adrian Prisacariu, Aleksandr V. Segal, and Ian Reid. Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I, ACCV'12*, pages 593–606, Berlin, Heidelberg, 2013. Springer-Verlag.
- [73] Carl Yuheng Ren and Ian Reid. A unified energy minimization framework for model fitting in depth. In *Proceedings of the 12th International Conference on Computer Vision - Volume 2, ECCV'12*, pages 72–82, Berlin, Heidelberg, 2012. Springer-Verlag.
- [74] K. Rohr, H. S. Stiehl, R. Sprengel, W. Beil, T. M. Buzug, J. Weese, and M. H. Kuhn. *Point-based elastic registration of medical image data using approximating thin-plate splines*, pages 297–306. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.
- [75] L I Rudin, S Osher, and E Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.
- [76] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.
- [77] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry. part i: The first 30 years and fundamentals. (June), 2012.
- [78] Geraldo Silveira, Ezio Malis, and Patrick Rives. An Efficient Direct Method for Improving visual SLAM. (April):10–14, 2007.
- [79] R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based SLAM using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.
- [80] Natalia Slesareva, Andr9s Bruhn, and Joachim Weickert. Optic flow goes stereo: A variational method for estimating discontinuitypreserving dense disparity maps. In *In Proc. 27th DAGM Symposium*, pages 33–40, 2005.
- [81] Paul Smith, Ian D. Reid, and Andrew J. Davison. Real-time monocular slam with straight lines. In *BMVC'06*, pages 17–26, 2006.

- [82] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In J.F. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 435–461. Elsevier Science Pub., 1988.
- [83] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Int. J. Robotics Research*, 5(4):56–68, 1986.
- [84] J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *Proc. of the IEEE/RSJ Int. Con. on Intelligent Robots and Systems, (IROS'05)*, pages 2499–2504, 2005.
- [85] F. Steinbruecker, T. Pock, and D. Cremers. Advanced data terms for variational optic flow estimation. Braunschweig, Germany, 2009.
- [86] F. Steinbruecker, T. Pock, and D. Cremers. Large displacement optical flow computation without warping. In *Proc. Int. Conf. Computer Vision*, Kyoto, Japan, 2009.
- [87] Frank Steinbrucker, Jurgen Sturm, and Daniel Cremers. Real-Time Visual Odometry from Dense RGB-D Images. (3), 2012.
- [88] H. Stewénus, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 284–294, 2006.
- [89] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time Monocular SLAM: Why Filter? In *IEEE International Conference on Robotics and Automation*, May 2010.
- [90] Jan Stuhmer, Stefan Gumhold, and Daniel Cremers. Real-Time Dense Geometry from a Handheld Camera. In *Proceedings of the DAGM Symposium on Pattern Recognition*, pages 11–20, 2010.
- [91] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [92] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21(4):311–330, 2002.
- [93] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Int. J. Robotics Research*, 23(7-8):693–716, 2004.
- [94] V. S. Varadarajan. *Lie groups, Lie algebras, and their representations*, volume 102 of *Graduate Text in Mathematics*. Prentice-Hall, 1974.

- [95] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.
- [96] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J. Leonard, and John McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *In IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [97] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J. Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *Int. J. Rob. Res.*, 34(4-5):598–626, April 2015.
- [98] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion. *Int. J. Rob. Res.*, 35(14):1697–1716, December 2016.
- [99] S. B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, Australian Centre for Field Robotics, University of Sydney, September 2001. Available at <http://www.acfr.usyd.edu.au/>.
- [100] M. Wisdom. Turbosquid. 3d models for professionals, 2000. Accessed 2018-04-03.
- [101] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *In Ann. Symp. German Association Patt. Recogn*, pages 214–223, 2007.
- [102] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.
- [103] H. Zhu, J. B. Weibel, and S. Lu. Discriminative multi-modal feature fusion for rgb-d indoor scene recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976, June 2016.
- [104] Mingqiang Zhu. *Fast Numerical Algorithms for Total Variation Based Image Restoration*. Doctoral thesis, University of California, 2008.