Diogo Emanuel Ribas Vaz

VSLAM and 3D Reconstruction Using πMatch

Coimbra, September 2017

UNIVERSIDADE DE COIMBRA

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

VSLAM and 3D Reconstruction Using πMatch

Diogo Emanuel Ribas Vaz

Coimbra, September 2017

VSLAM and 3D Reconstruction Using πMatch

Supervisor:

Professor Doutor João Pedro de Almeida Barreto

Jury:

Professor Doutor Hélder de Jesus Araújo

Professor Doutor João Pedro de Almeida Barreto

Professor Doutor Paulo Jorge Carvalho Menezes

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2017

"It is strange that only extraordinary men make the discoveries, which later appear so easy and simple."

— Georg C. Lichtenberg

Acknowledgements

The conclusion of this dissertation marks the end of an enriching, interesting and important journey of my life. A period of my life in which I have contacted with a lot of people whom have made me the person which I am today.

A special acknowledgement to my supervisor, Prof. João Pedro Barreto for the trust he has placed in me to the development of this work, as well as, to PhD Carolina Raposo for her incredible patience, dedication and monitoring. They have shared with me crucial insights and knowledge to the success of this work. During the development of the thesis, the elements of the research group have also contributed with their practical experience to the improvement of my practical skills.

I would like to thank to my colleagues and friends for the support that they have provided to me at the most difficult times. To my parents, I express a profound gratitude for unconditionally being on my side, giving an invaluable stability to my life. At last, thanks to Ana Rodrigues for her patience and emotional support that have made easier this complex and hard journey.

Resumo

O algoritmo πMatch é um método de SLAM visual monocular baseado em pontos salientes, que recorre à informação dos planos da cena, para calcular o movimento da câmara e para fazer uma reconstrução baseada em planos. Contrastando com outros métodos do estado da arte baseados em pontos salientes, este recorre a correspondências afim, em vez de correspondências de pontos, permitindo eficiência na deteção de planos e na estimação de hipóteses de movimento. O movimento da câmara e os planos relevantes são selecionados e refinados com formulações de PEaRL, sendo a reconstrução final obtida a partir de uma segmentação da imagem baseada num Campo Aleatório de Markov (Markov Random Field - MRF). Esta tese propõe uma implementação em C++ do algoritmo πMatch, originalmente implementado em Matlab, e importantes modificações em algumas das suas partes que permitiram acelerações computacionais significativas, preservando (ou mesmo melhorando) a precisão global e qualidade das reconstruções. Como primeira tarefa, foi feita uma tradução direta entre as linguagens Matlab e C++ para identificar os módulos do algoritmo com tempos computacionais inadequados. Assim, como problemas essenciais do algoritmo foram selecionados, o módulo de extração de correspondências afim e o módulo de segmentação da imagem em planos com MRF. Com o intuito de resolver estes problemas, são sugeridas versões alternativas destes módulos. O novo extrator de correspondências afim divide a imagem em blocos, tirando partindo da paralelização para alcançar melhores tempos computacionais. Por outro lado, o método inovador de segmentação baseada em MRF introduz o conceito de superpixel para reduzir a complexidade do problema, alcançando resultados semelhantes com um melhoramento extremo no tempo computacional. Todos os métodos alternativos sugeridos foram validados em testes comparativos com as respetivas versões originais. Por último, foi desenhada uma aplicação para incorporar o algoritmo, de modo a permitir uma fácil experimentação e visualização de resultados, usando o motor de jogo Unity 3D. Para provar o bom desempenho global do algoritmo foram realizadas experiências com o dataset

KITTI, avaliando as estimações do movimento da câmara e da escala relativa da translação. A versão final do algoritmo em C++ processa, aproximadamente, 1 imagem por segundo.

Palavras-chave: SLAM visual monocular, Reconstrução Baseada em Planos, Correspondência Afim, Superpixel, Campo aleatório de Markov

Abstract

The πMatch pipeline is a monocular feature-based VSLAM method that relies on the plane information of the scene to compute the camera motion and to perform a dense Piecewise Planar Reconstruction (PPR). In contrast to other state of the art feature-based methods, this method relies on affine correspondences, instead of point correspondences, allowing an efficient plane detection and estimation of motion hypotheses. The camera motion and the relevant planes of the scene are selected and refined with PEaRL formulations, being the final PPR obtained with an image segmentation based on Markov Random Field (MRF). This thesis proposes a C++ implementation of the πMatch pipeline, originally implemented in Matlab, and important modifications in some parts of the pipeline that enable dramatic computational speed ups while preserving (or even improving) the overall accuracy and quality of reconstructions. As first task, a straight-forward translation between Matlab and C++ languages was performed to identify the pipeline modules with inadequate computational times. Thereby, the extraction of affine correspondences and MRF segmentation of the image into planes were selected as the essential pipeline bottlenecks. In order to overcome the poor computational efficiency of the methods, new alternative versions of these modules are suggested. The novel ACs extractor performs the division of the image in blocks, taking advantages of parallelization to achieve better computational times. Moreover, the new MRF segmentation method introduces the superpixel concept to reduce the complexity of the problem, achieving similar results to the original, with a dramatic improvement in the time performance. All the suggested alternative methods were validated in comparative tests with the respective original versions. At last, an application was designed to incorporate the pipeline, in order to allow an easier experimentation and visualization of the results, using the Unity 3D game engine. Experiments on KITTI dataset were used to prove the good performances of the overall pipeline, evaluating the estimation of the camera motion and the relative scale of translation. The final C++ version of the pipeline processes, approximately,

1 frame per second.

Keywords: Monocular Visual SLAM, Piecewise Planar Reconstruction, Affine Correspondence, Superpixel, Markov Random Field

Contents

List of Figures

List of Tables

1 Introduction

In daily life, humans are able to walk in different environments, avoiding collisions with objects and other people that share the same space. This simple task is only possible by using the eyes and brain to construct an idea of the 3D map of the scene.

In robotics, this capability is one of the most desired, since it allows a correct navigation of robots in unknown environments. In this sense, Simultaneous Localization and Mapping (SLAM) was proposed [1]. SLAM is a generic process to estimate the position of a device, building at the same time a 3D map to represent the surrounding environment. These tasks are only possible by using one or more sensors to acquire data from the real world and a processing unit to process it. There are several SLAM methods which use expensive sensors, like laser sensors or inertial measurement units (IMUs) [2, 3].

In computer vision, another variant of SLAM is studied, called Visual Simultaneous Localization and Mapping (VSLAM) [4], where the sensors used to extract information from the environment are cameras, generally cheaper than other possibilities, achieving similar results. VSLAM is still an open research area due to its many applications in the real world. Nowadays, VSLAM methods can be applied, for example, in autonomous driving [5, 6], autonomous navigation of robots [7] and 3D reconstruction of rooms [8]. In the first application, the estimation of camera motion allows to locate the vehicle in the road and 3D reconstruction can provide information about obstacles, pedestrians and other vehicles to adjust its behaviour to the situation. The navigation of robots can be performed using previously defined maps, however the robots cannot adapt their behaviours in case of modifications in the environment. Since the VSLAM method reconstructs on the fly the surrounding environment, all the modifications are detected and the navigation is adjusted. At last, the 3D reconstruction of rooms is an interesting application of VSLAM, because it relies in the accurate estimation of the camera motions to perform a precise and detailed digitalization of 3D real scene, leading to a richer map of the environment.

1.1 State of the Art and Related Work

Visual Simultaneous Localization and Mapping is a generic process to determine the camera poses (position and orientation) and to simultaneously build 3D maps of the scene based on images. VSLAM architectures should be divided in direct[9, 10, 11] and non-direct[12, 13, 14] methods. The former are based on direct image alignment, or in other words they use full images to track the camera, by minimizing photometric error, and to map the scene, by estimating depth per-pixel. The latter are feature-based, so they need feature extraction and matching to get the data to analyse. In this approach the image analysis is reduced to the matched features between two consecutive images. The computation of camera poses is commonly done by minimizing reprojection errors and the mapping is performed by reconstructing 3D points or using a certain type of mapping-features. Obviously the differences in the approaches lead to different advantages and disadvantages. The following table summarizes them.

Direct Methods / Image Alignment Methods	**Non-direct Methods / Feature-based Methods**
Use more complete information (full image)	Use less complete information (only features)
Dense / semi-dense 3D reconstructions	Sparse 3D reconstructions
Slower (real-time implementation generally in GPUs)	Faster (real-time implementation generally in CPUs)
Less robust to outliers	More robust to outliers (using RANSAC-based methods)
Require small baselines (accomplished by higher frame rates)	Work with small and wide baselines
Highly affected by changes in lighting	Robust to changes in lighting (using appropriate features)

Table 1.1: Comparison between direct and non-direct standard methods. Green and red text represents, respectively, advantages and disadvantages.

After presenting the essential characteristics of each approach of VSLAM, it is relevant to frame the main VSLAM implementations that compose the state of the art architectures. Through the years feature-based methods were considered the best option due to their lower computational times. However, nowadays, the processing power of some devices allows a real-time execution of direct methods, making them a viable alternative to feature-based approaches. Between them, there are:

- **Dense Tracking and Mapping(DTAM)**[9]: DTAM is a monocular VSLAM method and a clear example of a direct method that was presented in 2011. The approach tracks camera motion based on whole images alignment and for each camera pose updates and expands the dense 3D model by estimating detailed and textured depth maps. To perform these tasks in real-time, it relies on GPUs.

- **LSD-SLAM**[10]: LSD-SLAM is a monocular VSLAM method proposed in 2014. It performs direct VSLAM with semi-dense 3D maps and achieves real-time performance in a CPUs. This is possible, because the algorithm does a filtering-based estimation of semi-dense depth maps.

- **Omnidirectional LSD-SLAM**[15]: This method is an extension of LSD-SLAM for wide field-of-view cameras, proposing a new image alignment method.

- **Stereo LSD-SLAM**[16]: This method is an extension of LSD-SLAM for stereo camera rigs. It aligns images directly based on photoconsistency of high-contrast pixels, including corners, edges and high texture areas and estimates depth of them, using both fixed-baseline stereo camera setup and temporal multi-view stereo. This approach also introduces illumination invariance methods.

On the other side, the non-direct methods are:

- **Paralel Tracking and Mapping (PTAM)**[12]: PTAM algorithm was presented in 2007 and is a method that treat tracking and mapping as separate parallel tasks. So, it performs monocular VSLAM using a set of optimization algorithms which are computational expensive and not commonly usable in real-time applications.

- **MonoSLAM**[13]: MonoSLAM was proposed in 2007 and is an implementation of VSLAM that gives more value to the robustness of the tracking task and reconstructs very sparse 3D maps. This is a pioneering method on monocular VSLAM, pretending to show that a single camera can be used to execute SLAM.

- **ORB-SLAM**[14]: ORB-SLAM is a more recent algorithm proposed in 2015. This method uses ORB-features that allow real-time use and ensure good invariance to changes in viewpoint and illumination. It is applicable to large environments and performs tracking, mapping, relocalization (to recover from tracking failure) and loop-closing.

- **ORB-SLAM2**[17]: ORB-SLAM2 extends the application of ORB-SLAM method from monocular cameras to stereo and RGB-D cameras.

The direct and non-direct methods should not always be associated, respectively, to dense or sparse 3D maps. Examples of these are the following works:

- **Semi-direct Visual Odometry (SVO)**[18]: SVO is a monocular VSLAM method published in 2014. Like the name suggests, it is in the boundary between direct and non-direct methods, because the camera tracking is based on image alignment and the mapping requires feature extraction when a certain frame is considered a keyframe.

- **Direct Sparse Odometry(DSO)**[11]: DSO is monocular VSLAM algorithm presented in 2016. This approach consist of a direct method that builds sparse 3D maps.

At last there is another type of VSLAM algorithms designed for RGB-D sensors, like Kinect-Fusion [19], ElasticFusion [20] and others [21, 17]. These achieve very interesting results by taking advantage of the depth image given by RGB-D cameras

After this brief review of the most important VSLAM algorithms for monocular, omni-directional, stereo and RGB-D camera systems, the rest of the discussion is going to centre in Monocular Visual Simultaneous Localization and Mapping algorithms. Although this type of VSLAM is one of the most explored research areas, the current methods still cannot overcome some problems that must be taken into account, like (1) dynamic foregrounds, (2) multiple motions, (3) pure rotation of the camera and (4) scale drift.

The direct methods have poorer performances than the indirect in situations of dynamic foreground and multiple motions. The moving objects in the scene can cause misalignment of the images and hence, a poor estimation of camera motion. Nonetheless the commonly used feature-based methods are affected in similar situations, since they assume that the scene is static/rigid for applying epipolar geometry. For applying the epipolar geometry, it is also assumed that the camera motion is constituted by a significant translation component. This assumption causes poor performances in presence of pure rotation motions. For direct approaches this type of motion does not constitute a problem. The last difficulty and eventually the worst is scale drift. This problem affects equally both direct and non-direct methods and, in the state of the art, is only solved or reduced by using prior information such as height of the camera or loop closures to perform global optimization.

Now, since a significant number of successful algorithms was presented, it is clear that all of them use points to compute the camera motion and to build the 3D map of the scene. Instead of using points, in [22, 23, 24], it has been demonstrated that, for stereo systems, there are advantages of using planes to estimate the camera motion estimation and to perform a Piecewise Planar Reconstruction. However, in monocular VSLAM the planes estimation is a demanding task, as the camera motion is not available. Even so, recently, Raposo and Barreto used a Piecewise Planar Reconstruction approach to construct the 3D

map of the scene in monocular VSLAM. This pipeline is named πMatch[25] and this master thesis is going to deeply study and improve it.

The πMatch is a feature-based monocular VSLAM method with several interesting particularities. The algorithm follows a research work about affine correspondences (ACs)[26] of the same author. Among other conclusions, it is demonstrated that a set of conditions is verified if two affine correspondences belong on the same plane. This property shows the fundamental advantage to feed the pipeline with ACs and not point correspondences (PCs), allowing an efficient detection and computation of planes, even being a monocular VSLAM method. Since the method is based on features, it has the advantages of non-direct methods, but also incorporates some advantages of direct methods.

The πMatch pipeline is robust to outliers, works with small to wide baselines, enables resilience to pure rotation motions, dynamic foreground and multiple motions, has no prior information to deal with scale drift problem and produce dense 3D maps of the scene, using a piecewise planar reconstruction approach. The following table synthesizes the comparison between direct, indirect and πMatch VSLAM methods.

	Direct	**Non-Direct**	**piMatch**
Robust to outliers	✘	✔	✔
Work with wide baselines	✘	✔	✔
Resilience to pure rotation motions, dynamic foregrounds and multiple motions	✘	✘	✔
Use no prior information to deal with scale drift	✘	✘	✔
Produce dense 3D maps of the scene	✔	✘	✔

Table 1.2: Comparison table of direct, non-direct and πMatch monocular VSLAM methods.

The table above clearly shows that πMatch combines the benefits of direct and non-direct approaches, being a relevant contribution to the literature of vSLAM. Thus, the main goal of this thesis is to investigate possible ways of improving its performance.

1.2 Initial Objective of the Thesis

As said above, the thesis is about the πMatch monocular VSLAM pipeline. This VSLAM algorithm was proposed in 2016 and presents computational times that justify more research to achieve a near real-time execution. Based on the interesting results presented in the original work, some objectives were defined:

1. Evolve from a prototype in matlab to a complete C++ application able to acquire

frames (from camera or file) and deliver SFM, and piecewise planar 3D reconstruction.

2. Include in this application a VR environment able to show the 3D reconstruction on-the-fly.
3. Determine main computational bottlenecks through profiling and find ways to overcome them in order to converge to a performance of at least 1 fps for 720p images.
4. Overcome difficulties in scale estimation and back-propagation of the plane info.

1.3 Accomplishments and Contributions

In order to achieve these objectives, a set of tasks were performed, constituting the contributions of this thesis. These tasks are divided into two parts. The first consists in designing and implementing an application with user interface that can work in three acquisition modes: frame on demand, maximum speed automatic acquisition and load images from file. The application is able to provide a piecewise planar reconstruction in an on-the-fly manner. This part implies:

1. Translating several modules of the pipeline from Matlab to C++.
2. Designing a multi-thread architecture.
3. Developing the image acquisition and VR environment using Unity3D.

In the second part, it was done a profiling of the application and improvement of modules with inadequate computational times to achieve a real-time performance. The improvements were done at two main levels:

1. New approach for establishing affine correspondences (ACs) that reduces computational time with respect to initial solution from 3.452 s to 1.097 s (VLFeat Accelerated Method, explained in subsection 3.1.2);
2. New MRF based algorithm for piecewise planar segmentation of images that is 40 times faster than standard implementation while accomplishing almost the same results (MRF formulation explained in subsection 4.1.4).

The outline of this thesis is the following. In chapter 2, the major steps of the πMatch pipeline are identified and described in detail, the efforts of translating the code from Matlab to C++ are reported and a first profiling of the πMatch is performed. This enabled the detection of the two modules with poor computational performances. Therefore, the

subsequent chapters 3 and 4 present possible solutions that greatly improve the pipeline performance. After this, chapter 5 proposes a multi-thread architecture for the algorithm and an application, to incorporate the pipeline, that allows to visualize the 3D model of the scene on-the-fly. In the chapter 6, some experiments on sequences with a large number of images are used to prove the good performances of the overall pipeline. In the last chapter (chapter 7), some conclusions and results are shown to reinforce the qualities of the final C++ version of the πMatch VSLAM.

2 πMatch Pipeline

The present chapter overviews the pipeline, breaking it in main functional modules for better understanding. The efforts done to the translation from matlab to C/C++ code of each module are presented. The last part of this chapter shows a first profiling to identify the main computational bottlenecks.

2.1 Overview of πMatch Pipeline

The πMatch is a feature-based monocular VSLAM method proposed by Raposo and Barreto [25] that takes as input a sequence of images acquired by a monocular camera and outputs the camera motion and a PPR of the scene. Its main functional modules are the following:

- Feature extraction, matching and affine correspondences (ACs) computation.
- Detection of planes by direct and non-iterative clustering of ACs.
- Motion hypotheses estimation.
- Camera motion selection and refinement.
- Scale estimation (only used in applications with more than two frames).
- Plane merging and refinement.
- Discrete optimization of planes (only used in sequences with more than five frames)
- Markov random field to segment the images by planes.

Figure 2.1 summarizes the pipeline structure and presents some intermediate results of certain modules. In the next subsections, the modules functionalities are explained in detail.

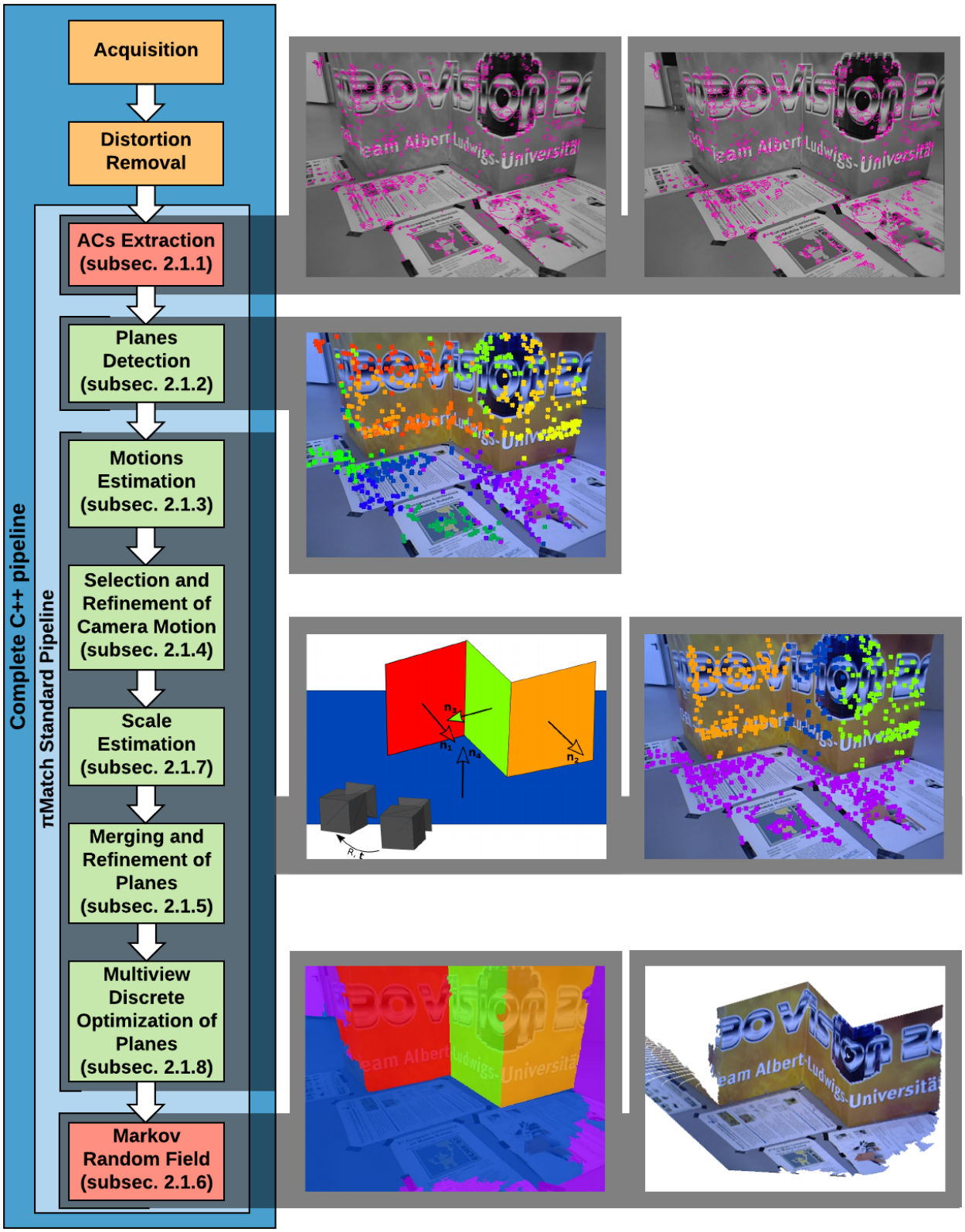


Figure 2.1: Schematic representation of the complete πMatch pipeline structure.

2.1.1 Affine Correspondences (ACs)

Affine correspondences are different from the commonly used point correspondences (PCs) as they include the information about local affine maps. Thus, an AC is defined by a PC

($\mathbf{x}$, $\mathbf{y}$) across views and a 2x2 non-singular affine matrix (A) that maps the neighbourhood of $\mathbf{x}$ into image patch around $\mathbf{y}$. The theory of affine correspondences was researched and deepened in [26].

The affine correspondences computation requires feature extraction, affine shape estimation, matching and the calculation of the affine mapping across views. The affine shape estimation consists of computing an affine matrix that maps an unit oriented circle to an oriented ellipse. Then to perform the first and second tasks, it is used an open-source library, named VLFeat [27]. This library has implementations of affine covariant feature detectors that receive an image as input and returns affine covariant features. These features are defined by a location and an affine matrix containing information of an oriented ellipse (affine shape) estimated based on the brightness of the image pixels around their locations. In this work, the features are extracted with the Difference of Gaussians operator. Using all this information, it computes SIFT descriptors to allow a future matching process. The matching task is done by measuring the squared euclidean distance between each descriptor and all others. During this process the distance to the closest (d_1) and second closest descriptors (d_2) are stored to allow the selection of good matches based on ratio (r_{match}):

$$r_{match} = \frac{d_1}{d_2}. \tag{2.1}$$

If ratio (r_{match}) is lower than a threshold, the match is classified as a good match. The good matches are finally used to compute the affine correspondences. Considering the two matching features across views ($\mathbf{x}$, $\mathsf{A}_\mathbf{x}$) and ($\mathbf{y}$, $\mathsf{A}_\mathbf{y}$), the AC ($\mathbf{x}$, $\mathbf{y}$, A) is defined by the PC and the affine map computed by:

$$\mathsf{A} = \mathsf{A}_\mathbf{y} \times \mathsf{A}_\mathbf{x}^{-1}. \tag{2.2}$$

For each image pair, a set of affine correspondences is extracted and used to feed the plane detection module.

2.1.2 Detection of Planes Using ACs (Π Det)

The detection of planes is a task known to be computationally expensive and difficult to perform in monocular VSLAM algorithms. However the use of affine correspondences can simplify the process, by exploiting their properties. In [26] is proven that, if there are two

ACs, ($\mathbf{x}$, $\mathbf{y}$, A) and ($\mathbf{z}$, $\mathbf{w}$, B), lying on the same plane, the four conditions below are verified.

$$
\begin{gathered}
(\mathbf{w}-\mathbf{y})^T \mathsf{PA}(\mathbf{z}-\mathbf{x}) = 0 \\
(\mathbf{w}-\mathbf{y})^T \mathsf{PB}(\mathbf{z}-\mathbf{x}) = 0 \\
\begin{bmatrix} s + a_2b_3 - a_3b_2 & -(a_1b_3 - a_3b_1) \\ a_2b_4 - a_4b_2 & s - (a_1b_4 - a_4b_1) \end{bmatrix} (\mathbf{w}-\mathbf{y}) = \mathbf{0} \qquad (2.3) \\
\text{with } s = \frac{[-a_2+b_2 \;\; a_1-b_1](\mathbf{w}-\mathbf{y}) - (a_1b_2 - a_2b_1)(x_1 - z_1)}{x_2 - z_2} \text{ and } \mathsf{P} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}
\end{gathered}
$$

The application of this property for plane segmentation is also demonstrated by the author, using the left-side expressions of the conditions as error metrics. So an average of the four metric is used to compute errors and fed to an affinity propagation clustering algorithm, proposed in [28]. The novel approach to segment the scene by planes is a great contribution of the πMatch pipeline. Nonetheless, the affinity propagation has two issues on segmentation: presence of outliers and over-segmentation of the scene into planes. The first occurs because the algorithm assigns each AC to a cluster, even when it does not belong to a plane or the ACs result from a mismatch in the feature matching process. All these problems are solved in the next pipeline stages.

2.1.3 Estimation of Motion Hypotheses (M Hyp)

The motion estimation module uses the clusters of affine correspondences generated by the affinity propagation and, for each one, computes an homography matrix. The estimation of homographies is performed by an MSAC scheme, using the four point correspondences method to compute them. The MSAC gives robustness to outliers, overcoming one of the affinity propagation issues. The motion hypotheses are determined by decomposing the homographies [29]. This decomposition process returns 4 possible motions, for each homography: (R_1, $\mathbf{t}_1$), (R_1, $-\mathbf{t}_1$), (R_2, $\mathbf{t}_2$) and (R_2, $-\mathbf{t}_2$). Despite this, for each rotation, only one translation vector is compatible with an image plane in front of the camera. So the reasonable translation vector can be selected by reconstructing 3D points, using ACs, and verifying if they are reconstructed in front of the camera (positive z-coordinate). Apart from this, if the camera performs a pure rotation movement, the translation vector would be a null vector, but, obviously, in the decomposition process this does not happen. To solve the problem, the homography matrix should be analysed. The homography (H) can be written

as

$$\mathsf{H} = \mathsf{R} + \frac{\hat{\mathbf{n}}\mathbf{t}^T}{d}, \tag{2.4}$$

where R is the rotation component of the motion, $\mathbf{t}$ is the translation, $\hat{\mathbf{n}}$ is the plane normal and d is the distance to the origin. In case of pure rotation ($\mathbf{t} = [0\ 0\ 0]^T$), $\mathsf{H} = \mathsf{R}$ and $\mathsf{H}\mathsf{H}^T = \mathsf{I}$. So, to detect pure rotation motions, the distance between $\mathsf{H}\mathsf{H}^T$ and the identity matrix (I) is computed. For computational efficiency, this distance is computed using the metric Φ_4, proposed in [30]. If the distance is lower than a threshold, the motion is considered a pure rotation and the translation component is nullified.

Summarizing, the module uses each previously computed cluster of ACs to estimate an homography and, per homography, generates two possible motions. Therefore, this module computes a set of camera motion hypotheses, but does not select one. Additionally, it returns a set of outlier-free clusters.

2.1.4 Selection and Refinement of Camera Motion (Cam M)

This module has the function of selecting the dominant motion from the set of hypotheses and afterwards refining the motion. This task is performed by a PEaRL framework, proposed in [31]. A PEaRL formulation is composed by discrete optimization and refinement of the model. Thereby, the selection problem should be treated as a labelling problem where the nodes of the graph ($p \in \mathcal{P}$, being $\mathcal{P}$ the set of nodes) are point correspondences, belonging to the outlier-free clusters, to which a label must be assigned (l_p). The label set $\mathcal{L} = \{\{\mathcal{R}_0, \mathcal{T}_0\}, l_\emptyset\}$ is composed by the motion hypotheses $\{\mathcal{R}_0, \mathcal{T}_0\}$ and the discard label $l_\emptyset$. The solution is computed by minimizing an energy function E.

$$\mathrm{E}(\mathbf{l}) = \underbrace{\sum_{p\in\mathcal{P}} \mathrm{D}_p(l_p)}_{\text{Data Term}} + \underbrace{\lambda_S \sum_{(p,q)\in\mathcal{N}} w_{p,q}\delta(l_p \neq l_q)}_{\text{Smoothness Term}} + \underbrace{\lambda_L|\mathcal{L}_\mathbf{l}|}_{\text{Label Term}} \tag{2.5}$$

The data term is the sum of D_p function value for every node p when the label l_p is assigned to them. D_p function is defined as the Symmetric Transfer Error (STE) [32],

$$\sum_i d(x_i, \mathsf{H}^{-1}x_i')^2 + d(x_i', \mathsf{H}x_i)^2 \tag{2.6}$$

if the label corresponds to a pure rotation motion and as Symmetric Epipolar Distance (SED) [32],

$$\sum_i d(x_i, \mathsf{F}^T x_i')^2 + d(x_i', \mathsf{F}x_i)^2 \tag{2.7}$$

in the other cases ($d(x, y)$ is the euclidean distance between the inhomogeneous points represented by x, y). The smoothness term introduces the neighbourhood concept in the problem, penalising changes in labelling between neighbour nodes. So supposing that p and q are neighbour nodes, the function $\delta(*)$ is 1 if the condition $*$ is verified or 0 if not, $w_{p,q}$ is the degree of neighbourhood between them (in this formulation is constant) and λ_S is a weighting constant. In this case, the nodes p and q are neighbours if they are members of the same cluster returned by the previous module. The label term forces the algorithm to use as few motion hypotheses as possible.

Generally, this discrete optimization step only assigns two labels to the nodes: a label that represents a motion hypothesis and the discard label. However, sometimes it can assign more than one label that represents a motion hypothesis. In this situation, the one that has more clusters associated with it and, in the case of tie, that has more points is selected as the camera motion.

At last, the camera motion is refined using the point correspondences, that were assigned to the corresponding motion label, in a bundle adjustment scheme based on Levenberg-Marquardt Algorithm. In this optimization, if the selected camera motion is a pure rotation, the STE is minimized, otherwise the SED is minimized.

The module returns a camera motion (R,$\mathbf{t}$) and the clusters of ACs only with the points assigned to the camera motion label. In case of selection of a pure rotation camera motion the pipeline cannot compute the planes of the scene neither the PPR, so it skips all the subsequent pipeline modules, passing to the computation of a next frame.

2.1.5 Merging and Refinement of the Planes (π Merge)

In this subsection, the method to generate the final set of planes to reconstruct is explained. This module was thought to solve the over-segmentation of the ACs, presented in subsection 2.1.2, which was ignored in previous modules. The over-segmentation without any process to merge leads to a set of planes with slightly different equation for the same real plane and hence a poor 3D map of the scene.

This part of the pipeline is also based on a PEaRL formulation, since it implies two essential steps: merging and refinement. Therefore, the first step can be cast to a labelling problem where the nodes are the point correspondences, belonging in the initial clusters, and the label set is composed by plane hypotheses and the discard label ($\mathcal{L} = \{\{\mathcal{P}_0\}, l_0\}$). These plane hypotheses are obtained by fitting the point clouds of each cluster associated

with camera motion to a plane, using linear least squares algorithm. The energy function to minimize is similar to the one presented in equation 2.5. In spite of this, there are some differences in the data and smoothness terms. In the first, the D_p function is defined by the STE obtained for the homographies computed using the refined camera motion and the plane hypothesis. In the smoothness term, the notion of neighbourhood is modified. In this case, the definition of neighbour nodes implies a 2D Delaunay Triangulation of the points, two nodes (p and q) are considered neighbours only if they are connected by a line segment in triangulated mesh. The degree of neighbourhood ($w_{p,q}$) is inversely proportional to the distance between two neighbour points, because closer points are more likely to belong to the same plane. Using this neighbourhood concept, possible errors in the first segmentation can be removed.

Finally, the planes are refined by minimizing the STE with Levenberg-Marquardt Algorithm, using the points labelled with the correspondent plane labels.

2.1.6 MRF-based Segmentation of Planes in Images (MRF)

A 3D model of the scene is obtained by performing a dense labelling of the image where each pixel is assigned to a plane hypothesis from the final set of planes (Π_f) or the discard label. In order to perform this task, it is used a Markov Random Field segmentation, proposed in [23]. In this MRF formulation the nodes of the graph are the image pixels ($p \in \mathcal{P}$, where $\mathcal{P}$ is the set of image pixels) and the label set ($\mathcal{L} = \{\{\Pi_f\}, l_0\}$) is composed by the plane hypotheses (Π_f) and the discard label ($l_\emptyset$).

The final labelling should be such that the energy function defined by

$$E(\mathbf{l}) = \underbrace{\sum_{p \in \mathcal{P}} \mathrm{D}_p(l_p)}_{\text{Data Term}} + \underbrace{\sum_{(p,q) \in \mathcal{N}} w_{p,q} \mathrm{V}(p,q)}_{\text{Smoothness Term}} \tag{2.8}$$

is minimized. The data term is based on photo-consistency between the first image of the pair and the second image transformed by the homographies, that are calculated with the estimated camera motion and the final set of planes. As photo-consistency metric, it is computed the normalized cross correlation (NCC) between two windows with same size (W and W^l) centred in pixels with same location in different images:

$$\mathrm{NCC}(p,l) = \frac{\sum\limits_{(x,y)} \left(\mathrm{W}_p(x,y) - \bar{\mathrm{W}}_p\right) \left(\mathrm{W}^l_p(x,y) - \bar{\mathrm{W}}^l_p\right)}{\sqrt{\sum\limits_{(x,y)} \left(\mathrm{W}_p(x,y) - \bar{\mathrm{W}}_p\right)^2 \sum\limits_{(x,y)} \left(\mathrm{W}^l_p(x,y) - \bar{\mathrm{W}}^l_p\right)^2}} \tag{2.9}$$

where p is the pixel (or node) where the NCC is computed and l is the label that represents the plane related to the second image warping. $\mathrm{W}_p(x, y)$ is the graylevel value of the pixel (x, y), in the window around p, of the reference image (first image of the pair) and $\bar{\mathrm{W}}_p$ is the mean of greylevel values of the pixels inside W_p. $\mathrm{W}_p^l(x, y)$ and $\bar{\mathrm{W}}_p^l$ are similar, but related to the second transformed image. The NCC is higher where the pixels are similar and lower otherwise, as opposed to what the energies must be. Since the NCC is inside the range from -1 to 1, the resultant value is inverted and limited between 0 and 1, being calculated the Normalized Cross Correlation Energy (NCCE):

$$\mathrm{NCCE}(p, l) = -0.5(\mathrm{NCC}(p, l) - 1) \tag{2.10}$$

Using equation 2.10, the NCCE images are obtained for each plane equation, and they constitute the D_p function, defining a value for each node and label. So

$$\mathrm{D}_p(l) = \begin{cases} \min\left(\mathrm{NCCE}(p, l), \mathrm{D}_{max}\right) & \text{if } l \neq l_\emptyset \\ \mathrm{D}_\emptyset & \text{if } l = l_\emptyset \end{cases} \tag{2.11}$$

where D_{max} and $\mathrm{D}_\emptyset$ are constants. The smoothness term is constituted by a $\mathrm{V}(p, q)$ function established as

$$\mathrm{V}(p, q) = \begin{cases} 0 & \text{if } l_p = l_q \\ \frac{1}{\lambda_{grad}\nabla I^2 + 1} \cdot \mathrm{T} & \text{if } l_p = l_\emptyset \vee l_q = l_\emptyset \\ \frac{1}{\lambda_{grad}\nabla I^2 + 1} \cdot \min(d(p, q), \mathrm{T}) + t & \text{otherwise} \end{cases} \tag{2.12}$$

where λ_{grad}, T and t are tuning parameters and $d(p, q)$ is the 3D distance between the neighbours belonging to the planes respectively represented by the labels l_p and l_q. $\nabla I = |I(p) - I(q)|$, the gradient between the neighbour pixels p and q. This term is only applied if the pixels p and q are neighbours. The neighbourhood of a pixel is constituted by the four-connected pixels around it.

In the end, the module outputs a labelling image and, based on it, the 3D map can be built with texture.

2.1.7 Scale Estimation

Since the absolute scale cannot be recovered in monocular sequences, a scale estimation module is required so that the relative scale between translation vectors is estimated. This module is only applied to sequences with three or more images, with the norm of the translation between the first two views being set to 1. Thereby, this module, for every two

consecutive motions (R_i, $\mathbf{t}_i$) and (R_{i+1},$\mathbf{t}_{i+1}$), where (R_i, $\mathbf{t}_i$) is the motion between the frames i and i+1, estimates a relative scale (s_i^{i+1}) for the $\mathbf{t}_{i+1}$ by fixing the norm of $\mathbf{t}_i$. The estimated scale is used to modify the second motion (R_{i+1},$\mathbf{t}_{i+1}$) to (R_{i+1},$s_i^{i+1}\mathbf{t}_{i+1}$), ensuring coherency between translation vectors estimated with different image pairs.

In order to perform this task, firstly, the tracklets or, in other words, the point correspondences of different image pairs that tracks the same real point, between views i and $i+2$, are computed and reconstructed in the reference frames i and $i+1$, using the motions (R_i, $\mathbf{t}_i$) and (R_{i+1},$\mathbf{t}_{i+1}$), respectively. Among the 3D points, the inliers are defined, as the points with a reprojection error lower than a threshold, and used in the computation of the scale. This method allows a good selection of inlier due to the accurate estimation of the rotation and direction of translation. The inlier 3D points relative to the first motion ($\mathbf{X}_i$) and the second ($\mathbf{X}_{i+1}$) are represented in the different reference frames, so to use them in the optimization process the points $\mathbf{X}_i$ are transformed by

$$\mathbf{X}_i' = \mathrm{R}_i\mathbf{X}_i + \mathbf{t}_i \tag{2.13}$$

to be referred in the same reference frame than points $\mathbf{X}_{i+1}$. So, the scale (s_i^{i+1}) is set to the median of the element-wise ratio $\frac{\mathbf{X}_i'}{\mathbf{X}_{i+1}}$.

The estimated scale is introduced in an optimization step to minimize the maximum reprojection error of the 3D points $\mathbf{X}_i'$ in frames $i+1$ and $i+2$, computed using the motion (R_{i+1},$\mathbf{t}_{i+1}$). The equation below shows how the optimization process is performed:

$$s_i^{i+1}* = \min_{s_i^{i+1}} \sum_k (\max(d_k^{i+1}, d_k^{i+2}))^2 \tag{2.14}$$

where d_k^i is the reprojection error of point k in frame i.

As said above, the scale estimation module must not be used when the camera motion is a pure rotation, but it must be resumed when the next camera motion with significant translation is estimated.

2.1.8 Multiview Discrete Optimization of Planes

This subsection explains the module thought to allow a more attractive Piecewise Planar Reconstruction by selecting the best planes across multiple views. This module was inspired in [22], where the author proposes to simultaneously refine the camera motion and the PPR in a PEaRL framework. Since, in the πMatch, the camera motion and estimated planes are refined in the previously presented modules, this stage only performs a discrete optimization in a sliding window approach to improve the overall PPR. Performing an optimization

over multiple frames allows the backpropagation of planes estimated in different views, as demonstrated in [22].

This discrete optimization is formulated as a labelling problem, where the goal is to minimize the energy function

$$E(\mathbf{l}) = \underbrace{\sum_i \sum_{p^i \in \mathcal{P}} \mathrm{D}_{p^i}(l_{p^i})}_{\text{Data Term}} + \underbrace{\lambda_S \sum_i \sum_{(p^i,q^i)\in\mathcal{N}} w_{p^i,q^i}\delta(l_p^i \neq l_q^i)}_{\text{Smoothness Term}} + \underbrace{\lambda_L|\mathcal{L}_l|}_{\text{Label Term}} , \tag{2.15}$$

where λ_S and λ_L are constants. In this problem the label set ($\mathcal{L} = \{\{\bigcup_i \mathcal{P}_f^i\}, l_\emptyset\}$) is the union of the planes estimated for each image pair i and the nodes ($p^i \in \mathcal{P}$) are the point correspondences computed for each image pair i. In order to define the data term, the planes in the label set must be represented in all reference frames i using the refined camera motions (R_i, $s_i^{i+1}\mathbf{t}_i$). The transformed planes are used to define the D_p function by computing the STE for each node p and label l.

The neighbourhood $\mathcal{N}$ is defined with two types of neighbours: neighbours in the same image pair i and neighbours between image pairs. The first are computed by performing Delaunay Triangulation of the points of each image i, with the connected points (p^i and q^i) in the triangulated mesh being considered neighbours, with a neighbourhood degree inversely proportional to the distance between them. The neighbours between views are defined by the points p^i and q^i, belonging to different views, that track the same real point (tracklets). In this case, the neighbourhood degree is set with a large constant value, forcing the assignment of the same label to these points.

The label term is used to select as few planes as possible to define the scene, choosing the best planes estimated with the image pairs inside the sliding window.

The discrete optimization is performed with a sliding window of five images with overlap of one. The size of the sliding window was chosen to achieve good results and reasonable computational times.

2.2 Translation Matlab to C/C++

The essential purpose of the thesis is to develop a C/C++ implementation of the πMatch pipeline. The translation from Matlab to C/C++ is a complex task, since the two programming languages are completely different. The main difficulties associated to the translation task were the following:

- functions to allow images and matrices manipulation and to compute algebraic operations are not implemented in the standard C/C++.
- optimizing a code is different in Matlab and in C/C++, so the code structure cannot be the same.
- a correct implementation of the pipeline in C/C++ requires software engineering.

In this way, two open-source libraries were used to facilitate the translation: the Eigen library and the OpenCv library. The former is a library for linear algebra, implementing matrices, vectors, operations between matrices and numerical solvers. The latter is a computer vision library that allows the manipulation of images and implements a large number of computer vision algorithms. Although several useful functions are implemented in the used libraries, certain functions were created, like geometric functions (ex. 3D points reconstruction, reprojection error computation, etc.), specific warping functions and others.

Other libraries used in the C/C++ implementation are the VLFeat library and the GCOptimization library. The first, as explained in subsection 2.1.1, is a feature extraction library with Matlab framework (used in matlab version of πMatch) constructed over a C library. The use of this library in C is more complex than in matlab and the translation required a detailed analysis of the Matlab framework to build the C/C++ code of the feature extraction with similar functionality. The second is a graph cut optimization library that is widely used in the pipeline (subsection 2.1.4, 2.1.5, 2.1.6 and 2.1.8). This library also has a Matlab framework (used in matlab version of πMatch) based on a C++ library that was analysed to adapt to the C++ implementation.

The C/C++ implementation pretends to be a complete VSLAM pipeline. In this regard, modules to perform image acquisition and distortion removal are added to the pipeline. The distortion removal requires a pre-calibration of the camera to obtain its distortion and intrinsic parameters.

2.3 First Profiling

In the thesis, the evaluation tests were performed in a laptop computer with an Intel® Core™ i7-3610QM CPU @ 2.30GHz processor. The tests were done using the same six pairs of images with resolution close to 720p, presented in figure 2.2, between the chapters 2 and 4. These image pairs were acquired with the left and right cameras of a Bumblebee stereo pair, so the ground-truth of camera motion is known.

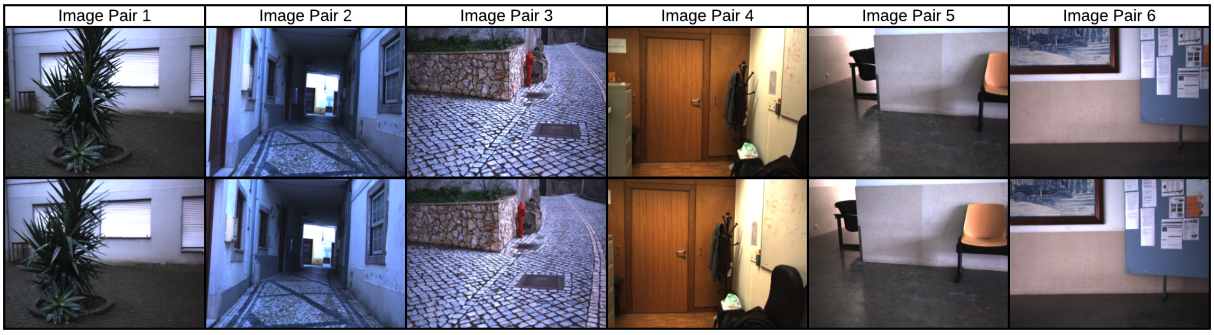


Figure 2.2: The six test image pairs.

After presenting all the modules of the πMatch pipeline, it is the moment to show the computational times of each one and to compare the performances between Matlab and C++ versions. The C++ version used in the comparison was a straightforward translation of the code, without changing any module of the pipeline. In this test, the number of ACs is limited to approximately 650 for computational efficiency. Table 2.1 shows the detailed computational times of matlab and C++ version per module and, in the penultimate column, for the overall pipeline. The last column provide the speed ups, in times, between Matlab and C++ implementations. Since these tests were performed with image pairs, only the modules from 2.1.1 to 2.1.6 were tested. Figure 2.3 presents the distribution of computational times for each pipeline module and for the overall pipeline in Matlab and C++ versions.

Pair	Version	ACs	Π Det	M Hyp	Cam M	Π Merge	MRF	Total	Speed Up
1	Matlab	3.358	0.214	0.538	0.052	0.195	80.317	84.674	1.789
	C++	3.044	0.196	0.297	0.008	0.014	43.779	47.339	
2	Matlab	2.123	0.244	0.663	0.103	0.133	98.764	102.031	1.211
	C++	1.866	0.241	0.072	0.055	0.023	81.973	84.231	
3	Matlab	2.074	0.243	0.550	0.102	0.177	59.248	62.393	1.426
	C++	1.813	0.222	0.045	0.059	0.023	41.606	43.768	
4	Matlab	2.348	0.193	0.375	0.065	0.172	109.643	112.796	1.834
	C++	2.131	0.189	0.068	0.020	0.014	59.086	61.508	
5	Matlab	3.391	0.183	0.482	0.062	0.085	44.033	48.236	1.666
	C++	3.106	0.175	0.097	0.022	0.016	25.533	28.949	
6	Matlab	2.548	0.212	0.528	0.083	0.147	70.014	73.531	5.206
	C++	1.770	0.187	0.056	0.056	0.017	12.039	14.124	
Average	Matlab	**2.640**	0.215	0.523	0.078	0.152	**77.003**	80.610	1.728
	C++	**2.288**	0.202	0.106	0.037	0.018	**44.003**	46.653	

Table 2.1: Computational times per module, in seconds, of the Matlab and C++ versions of the pipeline. The overall speed ups, in times, are presented in the last column of the table.

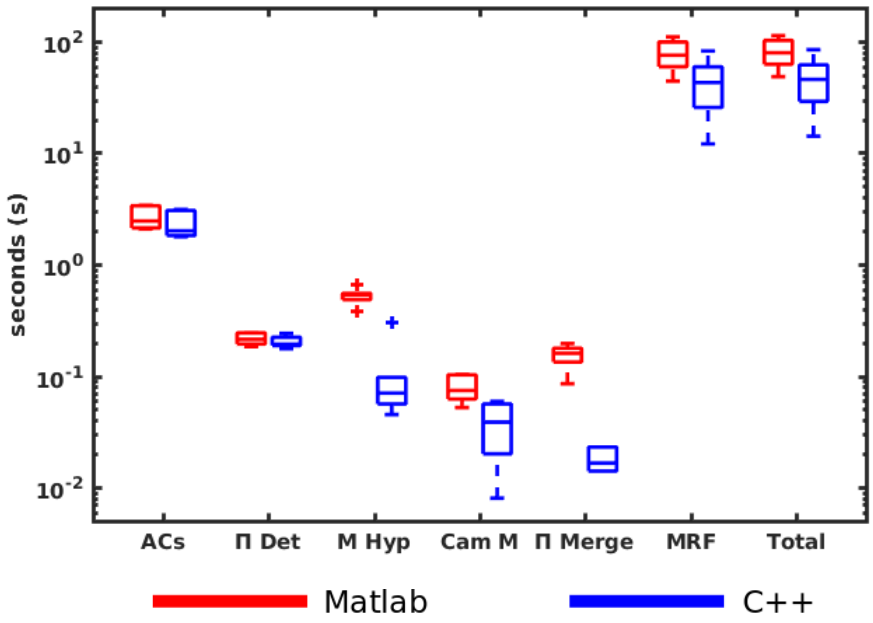


Figure 2.3: Distribution of computational times per module, in seconds, of the Matlab (red) and C++ (blue) versions.

The results presented in the table show an enhancement in the time performance between the Matlab and C++ versions in all modules. This C++ implementation is about 2 times faster than the original matlab version.

Despite the speed ups achieved with the translation between Matlab and C++, the overall computational performance is not reasonable for a VSLAM pipeline. The mean time results of the ACs extractor module (2.1.1) and the MRF segmentation module (2.1.6) demonstrates that they are the essential bottlenecks of the pipeline.

3 Fast Establishment of Affine Correspondences

As reported in section 2.3, the module for establishing ACs is one of the main bottlenecks of the πMatch pipeline as its high computational time hampers real-time performance. Thus, the present section explores alternative solutions for feature extraction and AC computation with the intent of allowing the method to run in real-time while maintaining the accuracy in camera motion estimation and plane detection.

3.1 Proposed Methods

Two different methods were analysed: Maximally Stable Extremal Regions (MSER) [33] feature extraction with SURF description [34] and an accelerated version of VLFeat. This section describes both methods and presents comparative results, allowing the most adequate one to be selected.

3.1.1 MSER Feature Extraction with SURF Description

The first proposed method consists of changing the type of features used, taking advantage of less computationally expensive feature extractors that provide an affine shape around the feature point. The MSER Feature Extractor, proposed in [33], is a new approach to the standard method that accelerates the execution, being two times faster than previously suggested implementations in FPGA. To use this feature extractor in the pipeline, an implementation of this work [35] was chosen.

The MSER feature extractor receives an image as input and computes features, returning the location and the parameters (second order moments of the regions) that encode an ellipse around them. However it does not compute any type of feature description which is fundamental to the matching process. To overcome this problem, the description is done using

a library that implements SURF algorithm [36]. Thus, using the surf descriptors, a simple matching method similar to the one explained in subsection 2.1.1 can be performed. These algorithms allow the definition of point correspondences and not affine correspondences. To define the ACs, the affine mapping should be computed as presented in equation 2.2, but the ellipse information is not encoded into an affine matrix. Hence, the affine matrix (A_i, denoting $\mathsf{A}_\mathbf{x}$ or $\mathsf{A}_\mathbf{y}$ presented in equation 2.2) is computed from the matrix of second order moments (Σ) using

$$\begin{cases} a_{11} = \sqrt{\Sigma_{11}} \\ a_{12} = 0 \\ a_{21} = \frac{\Sigma_{12}}{a_{11}} \\ a_{22} = \sqrt{\Sigma_{22} - a_{21}^2} \end{cases} \quad \text{where } \mathsf{A}_i = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix}. \tag{3.1}$$

Matrix A_i is one of the multiple possibilities of affine matrix that transforms an unit circle to an ellipse, because the ellipse has five degrees of freedom and not six like the affine transformation. In this case, the restriction applied is such that the affine matrix maps the y axis to itself. The matrix can only be unique if the ellipse was oriented, similarly to what happens in the method presented in subsection 2.1.1. Since the affine matrix is estimated the ACs are defined by the point correspondences, resultant from matching, and the affine mapping established as presented in equation 2.2.

3.1.2 VLFeat Accelerated

The good performance of the πMatch pipeline depends on the quality and the spatial spreading of affine correspondences and not only on the number of these correspondences, i.e., the existence of 100 ACs or 1000 ACs on the same plane is relatively indifferent. Therefore, improvements in both the computational performance and accuracy of πMatch pass by limiting the number of ACs while assuring that they properly sample and represent the different planes of the scene.

The VLFeat library performs the detection of salient points in scale space, choosing points whose the derivative along scale is above a certain peak threshold (δ). Moreover, the computational time of the matching process strongly increases with the number of detected features.

The original pipeline module limits the number of features by increasing, δ, but, depending on the texture of the images, this may lead to high concentrations of features in certain

regions of the image and lack of features in others.

In order to solve this problem, the proposed method, VLFeat Accelerated, divides the image into blocks, which enables to speed up detection by parallelizing the process, as well as to limit the number of ACs per block while assuring spreading. The limitation of the number of ACs, in each block, is not performed by increasing the peak threshold δ, since this would lead to no detection in poor textured blocks. In this method, δ is kept low and only a part of the detected features is considered. A straightforward way of limiting the number of features is to randomly select them. However, it was experimentally observed that many features did not provide a match, and thus were discarded, as shown in the left image of Figure 3.1. This problem arose because of the poor quality of the selected features. In order to overcome this, VLFeat accelerated selects the best features as the ones that provide a higher peak value. Figure 3.1 shows that this selection method provides a significantly higher number of matches than random selection, benefiting the subsequent steps of the pipeline. At last the matching process is accelerated by parallelization.

Figure 3.1: Matched features, using a random features selection (left), not limiting the number of features (middle) and selecting the best features (right).

3.2 Comparative Results

In the current section, the original and proposed methods are evaluated and compared. The evaluation is based on the computational time, the quality of the camera motion estimation and the quality of the estimated planes. The test image pairs have ground-truth of the camera motion (R_{GT}, $\mathbf{t}_{GT}$), as said above, then the estimated rotation (R) and translation ($\mathbf{t}$) are easily compared to it. In this experiment, the rotation error (e_{R}) is defined by the angular magnitude of the residual rotation ($\mathsf{R}^T\mathsf{R}_{GT}$) and translation error ($e_{\mathbf{t}}$), by the angle between the vectors $\mathbf{t}$ and $\mathbf{t}_{GT}$. However, the information of the real planes does not exist. Thus, it is considered as pseudo-ground truth the plane equations (normal vector, $\hat{\mathbf{n}}_{GT}$, and

distance to the origin, d_{GT}) obtained with the standard pipeline using a large number of features. After estimating the pseudo ground-truth planes, the test images are manually labelled. The resultant labellings for the six image pairs are presented in the figure 3.2. Thereby, the estimated planes are compared with the pseudo ground-truth planes to infer

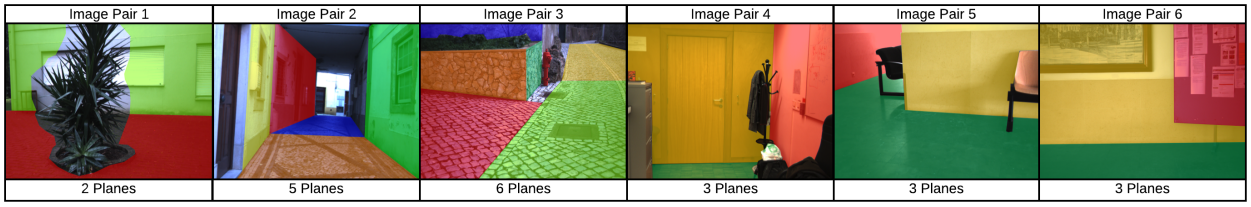


Figure 3.2: The manually labelled the test images, for the pseudo ground-truth planes.

their quality. They are represented by a normal vector ($\hat{\mathbf{n}}$) and a distance to the origin (d) and their evaluation is based on errors in these parameters. The error in the normal vector ($e_{\hat{\mathbf{n}}}$) is quantified by the angle between $\hat{\mathbf{n}}_{GT}$ and $\hat{\mathbf{n}}$, and the error in the distance to the origin (e_d) by the magnitude of the difference between d_{GT} and d. Since the pseudo ground-truth planes are estimated, they have themselves an error associated to their estimation. In this sense, the obtained errors can be higher than expected, as they are, somehow, the sum of errors introduced by the plane estimation in the experiments and in the pseudo ground-truth computation.

The comparative test is performed by running the pipeline fifty times per image pair, with each method. This type of test is used, in order to analyse the repeatability of correct estimations, taking into account the random component of the pipeline given by the MSAC-based motion hypotheses estimation module. Since the output of the AC extraction module affects the performance of the subsequent stages of the pipeline, both in terms of computational time and accuracy, an important characteristic of the algorithm that should be taken into account is the ability to easily limit the number of outputted ACs, without losing too much information. In this regard, the parameters of the algorithms were changed to limit the maximum number of computed ACs to 1000.

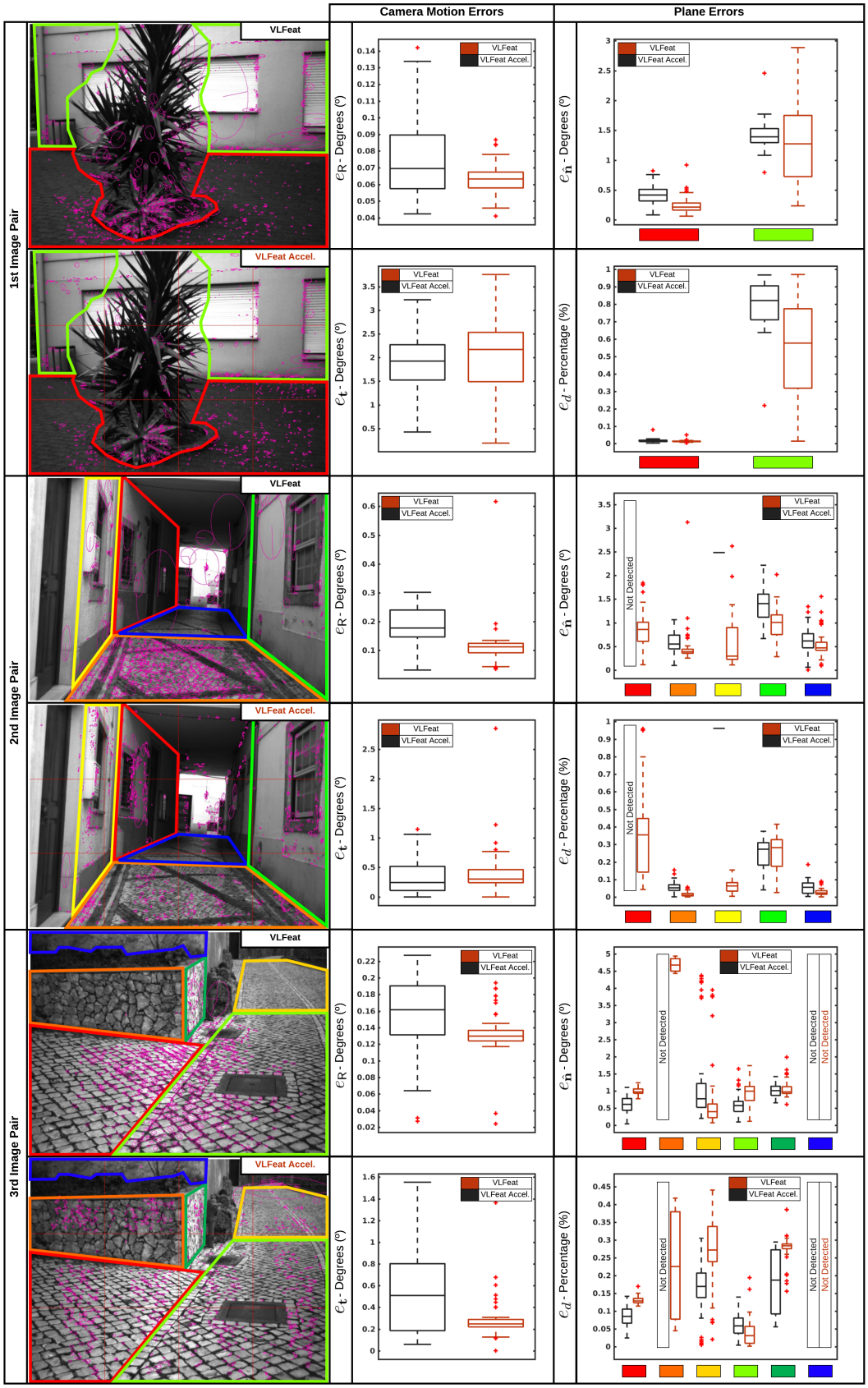
Table 3.1 synthesizes the results of tests and allows to coarsely understand the quality of each method. The time column shows the computational time per pair in seconds, including feature extraction in the two images of the pair, matching and computation of ACs. The table also presents the percentage of correct estimations of the camera motion, showing the robustness and repeatability. The mean and standard deviation of the errors were only computed when the camera motion is correctly estimated. In this case, a correct camera motion is defined as a motion that has errors in rotation (e_{R}) and translation ($e_{\mathbf{t}}$) lower than

5°. In the last column of the table, the average percentage of planes that match with the pseudo ground-truth planes is also shown. A plane is considered correctly estimated if the error in the normal vector ($e_{\hat{\mathbf{n}}}$) is lower than 5° and the relative error in the distance to the origin (e_d) is lower than 1 (corresponding to 100%).

Pair	Method	Time	% Correct Motions	$\bar{e}_{\mathsf{R}}$	$\sigma_{e_{\mathsf{R}}}$	$\bar{e}_{\mathbf{t}}$	$\sigma_{e_{\mathbf{t}}}$	% Planes
1	VLFeat	4.268	100	0.074	0.022	1.902	0.599	61
	MSER	0.480	-	-	-	-	-	-
	VLFeat Accel.	1.095	100	0.063	0.010	2.022	0.667	93
2	VLFeat	2.354	88	0.181	0.072	0.332	0.289	60
	MSER	0.541	40	0.796	0.825	1.490	1.241	48
	VLFeat Accel.	1.093	100	0.115	0.079	0.424	0.416	85
3	VLFeat	2.239	100	0.156	0.048	0.533	0.386	67
	MSER	1.734	100	0.134	0.036	0.469	0.473	68
	VLFeat Accel.	1.106	100	0.133	0.029	0.312	0.252	68
4	VLFeat	2.761	86	0.183	0.065	0.318	0.262	94
	MSER	0.274	-	-	-	-	-	-
	VLFeat Accel.	1.071	82	0.178	0.057	0.244	0.203	98
5	VLFeat	6.621	100	0.102	0.049	0.202	0.134	69
	MSER	0.188	-	-	-	-	-	-
	VLFeat Accel.	1.120	100	0.181	0.045	0.237	0.155	73
6	VLFeat	2.470	80	0.131	0.044	0.901	0.857	67
	MSER	0.328	26	0.348	0.126	1.735	1.414	28
	VLFeat Accel.	1.097	100	0.172	0.028	0.428	0.187	100

Table 3.1: Summarized results of the comparative tests. The camera motion estimation quality is represented by the mean and standard deviation of the error in rotation ($\bar{e}_{\mathsf{R}}$, $\sigma_{e_{\mathsf{R}}}$) and translation ($\bar{e}_{\mathbf{t}}$, $\sigma_{e_{\mathbf{t}}}$) for the fifty runs. The average percentage of matched planes is also shown (% Planes).

A first analysis of the table clearly demonstrates that the MSER method has poorer performances in more than half of the cases and the camera motion estimation fails in general. In this sense, it is discarded in the remaining results presented in the figure 3.3. The figure depicts the performances of camera motion and planes estimation, using the VLFeat and VLFeat Accelerated methods, with more detail.


Camera Motion Errors
Plane Errors
1st Image Pair
2nd Image Pair
3rd Image Pair
VLFeat
VLFeat Accel.
$e_{\mathbf{R}}$ - Degrees (°)
$e_{\mathbf{t}}$ - Degrees (°)
$e_{\mathbf{n}}$ - Degrees (°)
e_d - Percentage (%)
Not Detected

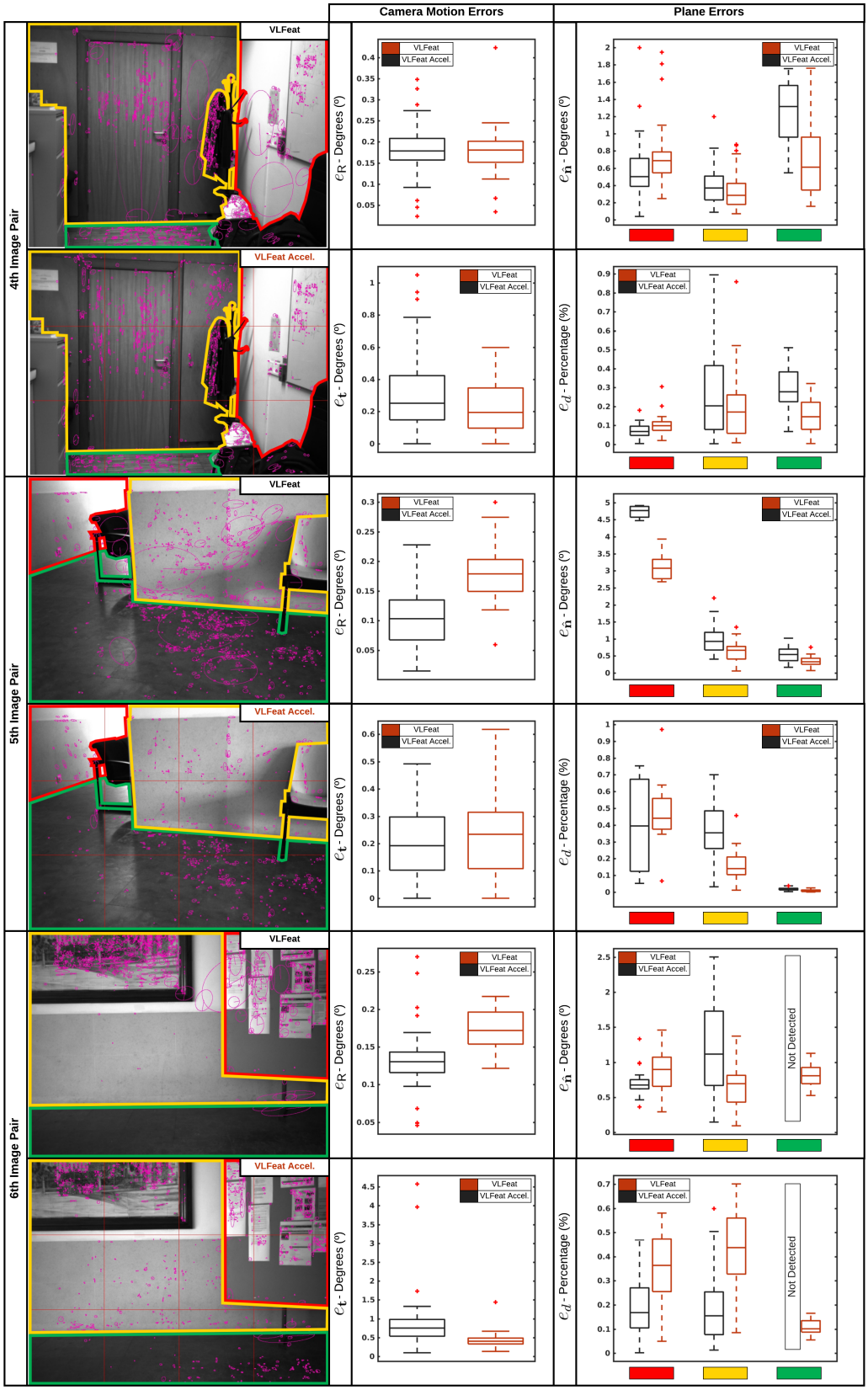


Figure 3.3: The detailed results of VLFeat and VLFeat Accelerated methods. The errors in camera motion ($e_{\mathbf{R}}$ and $e_{\mathbf{t}}$, both in degrees) and in plane estimation ($e_{\hat{\mathbf{n}}}$ and e_d, respectively in degrees and percentage, where 1 corresponds to 100%) are shown to evaluate the methods. The pseudo ground-truth planes are represented in the left images by colours which are used in the box plots on the right hand side to identify the planes.

3.3 Discussion

The comparative results of the ACs extractors has two dimensions that should be analysed: the computational time and the quality of the outputted ACs.

Firstly, the MSER-based method, in general, has low computational times, because it extracts a reduced and insufficient set of features. Thereby, the results demonstrates that the method is not a good alternative to substitute the standard VLFeat method.

On the other hand, VLFeat Accelerated extracts a reasonable number of ACs and simultaneously has lower computational times than standard VLFeat. With respect to camera motion and plane estimations, VLFeat Accelerated has similar or better results, as shown in the box-plots of the figure 3.3. This proposed method frequently estimates more planes than the original, because the original method does not guarantee the dispersion of the features in the image (ex. image pairs 2, 3 and 6) and the planes of certain regions are lost. Other important particularities of the VLFeat Accelerated are the invariance to the feature detection parameters, the more stable computational times in different image types and the easy limitation of the outputted ACs. Using the original VLFeat, the limitation of the number of ACs was performed by tuning the peak threshold in the feature extraction for each image pair. In contrast to this with VLFeat accelerated the number of outputted ACs is limited for all images using always the same parameters, achieving more stable computational times. The standard deviation of the computational time reinforces this idea, being significantly higher for the original method (1,328 s) than for VLFeat Accelerated (0,011 s).

VLFeat Accelerated has equal or better results than the original method, being 2.6 times faster. This evaluation supports the substitution of the ACs extraction module of the pipeline.

4 Fast MRF Segmentation of Images into Planar Regions

The MRF-based method presented in subsection 2.1.6 is a very complex discrete optimization problem due to the high number of nodes (all pixels in the image), thus being extremely slow to apply in a VSLAM pipeline.

This type of problem was studied in several situations and the existing solutions are typically divided into two approaches: local and global approaches. Local approaches are commonly fast, but the segmentation can be poor, in certain cases, because it is based on a small part of the image. In contrast to this, there are the global approaches that generally are time-consuming, but present more robust results. After analysing the method proposed in [37], the idea of using superpixels in a MRF-based segmentation rose. The research evolved in this direction as a way to decrease the number of nodes in the MRF formulation as thus increase speed.

4.1 Proposed Algorithms

The proposed algorithms are light versions of the standard MRF-based method by reducing the number of nodes of the graph to allow a faster convergence. The approach presented by [37] suggests the use of SLIC superpixels [38], in a different situation, with the objective to build a piecewise-planar model of the scene based on sparse 3D point cloud. The SLIC superpixels definition is fast when compared with other type of superpixels of the state of the art, but it is not fast enough. So other methods to define superpixels were evaluated like SEEDS [39] and Preemptive SLIC [40] which have better computational performances than SLIC method.

Since the goal of the proposed MRF methods is to provide an accurate segmentation of the scene into planes, the method that is selected for dividing the image into superpixels

should be able to provide superpixels that correctly delimit planes and do not intersect their limits. Analysing performances and the computational times, the choice was the Preemptive SLIC algorithm. This type of superpixels is an improved and faster version of SLIC that achieves similar results, being less time-consuming. Therefore, the MRF-based segmentation of image is converted to a less complex problem.

The next subsections explain the various methods devised to accelerate the segmentation process that minimizes a typical MRF energy function:

$$E(\mathbf{l}) = \underbrace{\sum_{p \in \mathcal{P}} \mathrm{D}_p(l_p)}_{\text{Data Term}} + \underbrace{\sum_{(p,q) \in \mathcal{N}} w_{p,q} \mathrm{V}(p,q)}_{\text{Smoothness Term}} \tag{4.1}$$

4.1.1 Superpixels in RGB Images (M1)

This first method is a MRF formulation very similar to the explained in the subsection 2.1.6, but with lower complexity. The first image of the pair is fragmented in Preemptive SLIC superpixels and they are used, in the graph, as nodes ($p \in \mathcal{P}$). The label set ($\mathcal{L} = \{\{\Pi_f\}, l_\emptyset\}$) is composed by the final set of planes ($\{\Pi_f\}$) and a discard label ($l_\emptyset$). The modification in the type of nodes leads to a new definition of the terms that compose the energy function in equation 4.1.

The D_p function, in the new approach, implies a pixelwise normalized cross correlation (PNCC) calculation between the first image of a pair and the second warped with the homographies defined with the candidate plane equations, like in the original method. The NCC values computed for each pixel are converted in the NCCE values, as in equation 2.10. However, in the new formulation, the nodes are superpixels and not pixels. In order to determine a superpixel-level metric to establish the new data term function, it is computed an approximated normalised cross correlation energy (ANCCE) to each superpixel (S) equal to the mean value of the NCCE in the N_S pixels inside it ($s_i \in S$):

$$\mathrm{ANCCE}(S, l) = \frac{1}{\mathrm{N}_S} \sum_{s_i \in S} \mathrm{NCCE}(s_i, l) \tag{4.2}$$

Figure 4.1 shows the steps to the computation of ANCCE for each superpixel. Another modification is in the neighbourhood concept. In this method two nodes or superpixels ($p \in \mathcal{P}$ and $q \in \mathcal{P}$) are considered neighbours if they are adjacent, with a constant neighbourhood degree. $\mathrm{V}(p,q)$ does not suffer any type of modification in the structure (2.12). Nonetheless $d(p,q)$ is redefined as the 3D distance between the reconstructed centroids, if belonging to the planes corresponding to labels l_p and l_q. Also the gradient computation is redefined as the

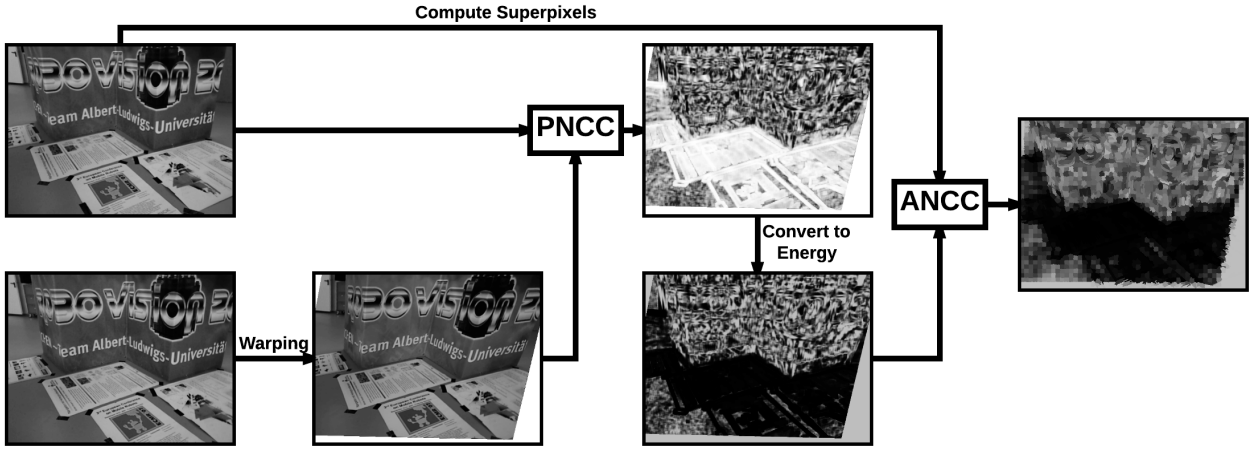


Figure 4.1: Schematic representation of the data term computation for one plane, in the first proposed MRF formulation.

absolute difference between the mean greylevel of the pixels inside the neighbour superpixels p and q. Since the superpixels are broader than pixels, this neighbourhood concept is less restrict and the parameter had to be readjusted.

4.1.2 Superpixels in the Correlation Space (M2)

The second method is an attempt to further reduce the complexity of the MRF by fragmenting the image into as few superpixels as possible. This attempt raises the difficulty of maintaining the quality of plane delimitation. The proposed method was designed to overcome this issue by using the RGB images to form another image that is a coarse plane segmentation of the first. In this image, different planes are differently coloured, allowing a good plane delimitation, even when a reduced number of superpixels is used. In this case, although the superpixels are extracted from the new image, they properly delimit the planes in the original image. Summarizing, this MRF formulation is a very similar approach to the above and only the method that segments the image into superpixels is changed.

In this method, the computation of NCCE images relative to the estimated planes (N planes) is also necessary to subsequently fuse them and to construct an image in the correlation energy space. The creation of a fusion image starts with the binarization of the NCCE images (N images), using the result to define a binary tag for each pixel (2^N possible tags). In the binarization process, a pixel is set to one, if its value is lower than a threshold, or zero otherwise. In order to assign more distinctive colors to more frequent binary tags in the fusion images, an histogram is constructed to count the occurrences of each tag. Using this histogram they are sorted in descending order. The ordinal number of a tag, in a binary

representation, codifies the colour of this tag in the fusion image. Since the tag 0 is only obtained if energies, in all NCCE images, are higher than the threshold, which happens where the correlation is low, it is differently treated and the black colour is assigned. Figure 4.2 depicts the complete process to define fusion images.

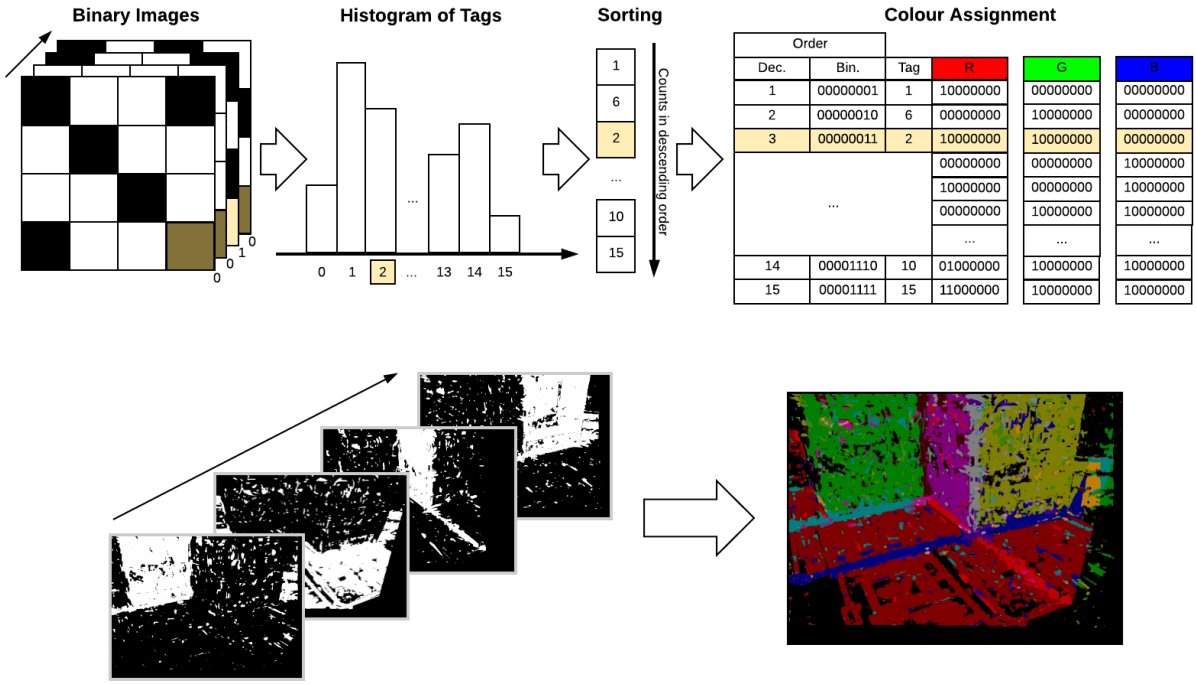


Figure 4.2: Schematic representation of the fusion image formation with four estimated planes (N= 4) (top) and real example (bottom).

The main objective of computing the fusion image is to obtain an image with bigger regions with similar colours segmenting the same planes, as depicted in figure 4.2. Hence, when using the Preemptive SLIC algorithm, a lower number of superpixels should be enough to correctly delimit the planes. In this way the MRF is accelerated by reducing the number of nodes. The formulation of the MRF problem is like the above, but using the superpixel grid obtained with the fusion image.

4.1.3 Superpixels in RGB Images and Superpixel-Level NCC (M3)

The third proposed method arises to resolve other performance problems associated with standard MRF formulation and to improve or refine the usage of superpixels. The low performance of the original MRF is also caused by the data term definition, because the normalized cross correlation is time-consuming when used in a pixelwise approach. Consequently, a new superpixelwise approach is suggested.

In this method, the first image of a pair is fragmented in a grid of Preemptive SLIC superpixels, like in the others. The second image is warped by the homographies associated to the planes, generating a number of transformed images equal to the number of candidate planes, as in the original method. This MRF problem has superpixels as nodes ($p \in \mathcal{P}$) and a label set ($\mathcal{L} = \{\{\Pi_f\}, l_0\}$) composed by the set of planes to reconstruct in the PPR. As described above, the D_p function definition is a slow part in the original MRF segmentation. The NCC, in the first and second proposed MRF methods, is computed for each pixel between the first and the transformed images (PNCC) to allow the definition of ANCC for each superpixel. Differently from both previous methods, in this method the superpixelwise normalized cross correlation (SNCC) was designed to quickly and directly determine, for each superpixel, the photo-consistency between the two images. The photo-consistency measure is the NCC calculated with the superpixels as windows. Using SNCC, the NCC value per superpixel is also converted in NCCE (equation 2.10). Since SNCC is computed without overlay of windows, in contrast to PNCC, the resulting NCCE transitions between neighbour superpixels are abrupt and need to be smoothed with a weighted average between them

$$\mathrm{NCCE}'(S,l) = \alpha_0 \mathrm{NCCE}(S,l) + \sum_{i=1}^{M} \alpha_i \mathrm{NCCE}(\mathrm{N}_i, l) \tag{4.3}$$

where α_0 and α_i $(i = 1 \dots M)$ are the weights, whose sum is one, M is the number of neighbour superpixels and N_i is neighbour superpixel i of S. Figure 4.3 depicts all the steps previously explained.

After the definition of NCCE′, the data term is the following

$$\mathrm{D}_p(l) = \begin{cases} \min(\mathrm{NCCE}'(p,l), \mathrm{D}_{max}) & \text{if } l \neq l_\emptyset \\ \mathrm{D}_\emptyset & \text{if } l = l_\emptyset \end{cases} \tag{4.4}$$

where D_{max} and $\mathrm{D}_\emptyset$ are constants.

The smoothness term has the same structure as in the original method. The V(p,q) function

$$\mathrm{V}(p,q) = \begin{cases} 0 & \text{if } l_p = l_q \\ \frac{1}{\lambda_{grad}\nabla I^2 + 1} \cdot \mathrm{T} & \text{if } l_p = l_\emptyset \vee l_q = l_\emptyset \\ \frac{1}{\lambda_{grad}\nabla I^2 + 1} \cdot \min(d(p,q), \mathrm{T}) + t & \text{otherwise} \end{cases} \tag{4.5}$$

is similar, but the function $d(p,q)$ and ∇I are redefined. First of all, two superpixels p and q are neighbours if they are adjacent. In this approach, the neighbourhood between two superpixels is defined by two representative pixels that are used in $\mathrm{V}(p,q)$ function

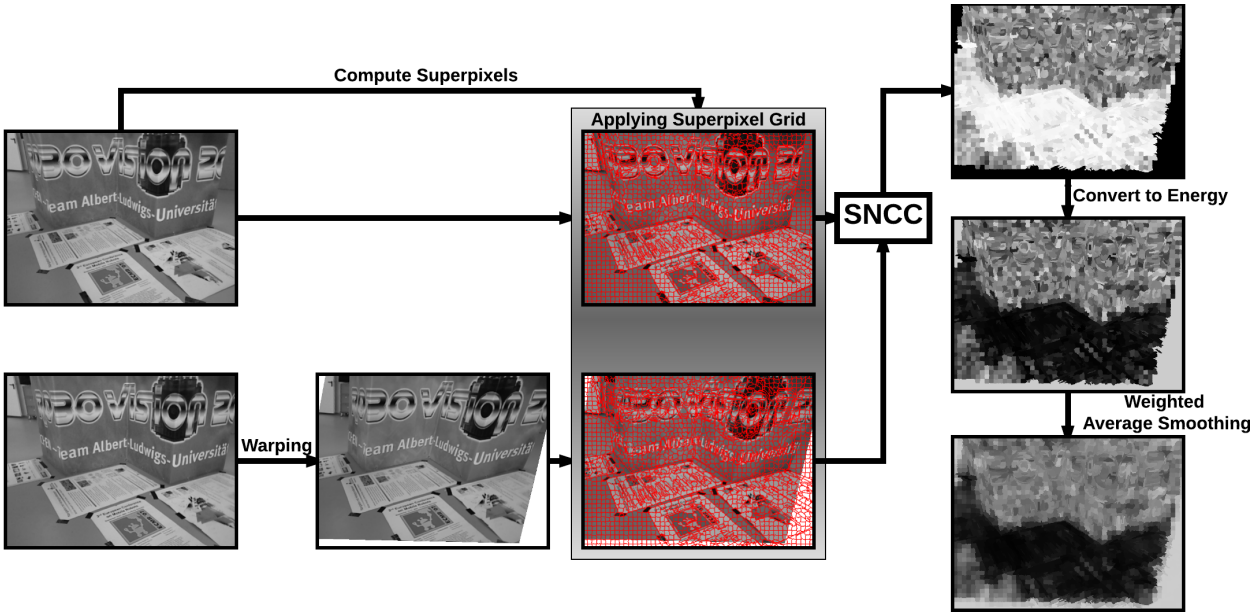


Figure 4.3: Schematic representation of the data term computation for one plane, in the third proposed MRF formulation.

computation. Analysing the grid of superpixels, these pixels are defined by intersecting the line segment, that links the centroids of two neighbour superpixels, with their inner borders. Thereby, $d(p, q)$ is the 3D distance between these two pixels reconstructed on the planes represented by the labels l_p and l_q. In this implementation, the ∇I is established as the distance between their RGB colours.

This last proposed method is faster than the others and extremely faster than the original method. However, the labelling in the transition of planes has faults and a solution to this problem is suggested in the next subsection.

4.1.4 Improvement by Adding Lines (M3 Ext)

The pixelwise approach of the original MRF formulation has high computational times, but produces a labelling in the transitions between planes with small error. On the other hand, the superpixelwise proposed methods have low computational times, but have poor performances in the transitions in certain cases, since the superpixels are coarse approximations to pixels. To solve the problem several possibilities were analysed, like edge detection in the image and the measurement of gradients in the superpixel boundaries to force transitions. However, these alternatives do not produce good results.

In this situation, it was thought how to use the scene information given by the previous pipeline modules to force correct transitions. Therefore, in this improvement, the intersec-

tions of the final estimated planes are computed and the resulting lines are projected to the image. The superpixels that are intersected by these lines are subdivided and the smoothness term is reformulated. The new smoothness term has the following structure:

$$\mathrm{V}(p,q) = \begin{cases} 0 & \text{if } l_p = l_q \\ \frac{1}{\lambda_{grad}\nabla I^2+1} \cdot \mathrm{T} & \text{if } l_p = l_\emptyset \vee l_q = l_\emptyset \\ (\frac{1}{\lambda_{grad}\nabla I^2+1} \cdot \min(d(p,q),\mathrm{T}) + t) f(p,q) & \text{otherwise} \end{cases} \tag{4.6}$$

where the novelty is $f(p,q)$ function. It is equal to a constant, lower than one, if the centroids of the superpixels p and q are separated by one of the lines projected in the image. This constant forces the transition between planes, because if the labels assigned to neighbour superpixels are different and a line separates them, the energy is reduced. The constant can be tuned to force more or less the transitions.

The subdivision of the superpixels, using the lines, allows to better delineate the planes in the image, even when the limits do not present any clear edge in it. This method, used in parallel with superpixels, is a good option because they are obtained with two types of information. While the lines are calculated with 3D information, the superpixels are obtained with 2D image information.

The current improvement can be applied in the three proposed methods, but only the last has a reasonable computational time to the target application. So this improvement is only added to the third method, the fastest.

4.2 Comparative Results

In order to assess the performance and accuracy of all the proposed methods, as well as understand their advantages and disadvantages, comparative tests were performed.

Also, to provide a fair comparison, the camera motion and the planes estimated in the pseudo ground-truth, as described in section 3.2, are used as input to all the MRF approaches.

The comparison is given both in terms of computational times and accuracy of labelling (qualitative and quantitative evaluation). Figure 4.2 shows the obtained labelling, providing a qualitative assessment of the segmentation results as well as a quantitative evaluation using the average Jaccard index computed for all plane labels. The Jaccard index is the ratio between the number of pixels equally labelled (in the ground-truth and in the evaluated labelling) and the number of pixels of the union set. When the index is equal to one, the ground-truth and evaluated labellings are equal. On the other hand, an index equal to zero

means that the labellings are completely different. Table 4.1 presents the computational times per image pair, for all methods, in seconds, and the average speed ups, in times.

Pair	Original	M1	M2	M3	M3 Ext
1	19.500	1.680	1.656	0.586	0.477
2	43.391	3.719	3.969	0.688	0.820
3	24.563	4.273	4.313	0.758	0.836
4	40.453	3.555	3.648	0.688	0.805
5	27.242	2.984	2.891	0.609	0.695
6	14.625	2.273	2.375	0.656	0.609
Average	**28.296**	3.081	3.142	0.664	**0.707**
Speed Up	1.000	9.185	9.006	42.610	**40.020**

Table 4.1: Computational time results per test image pair in seconds. The average speed ups, in times, are shown in the last row of the table.

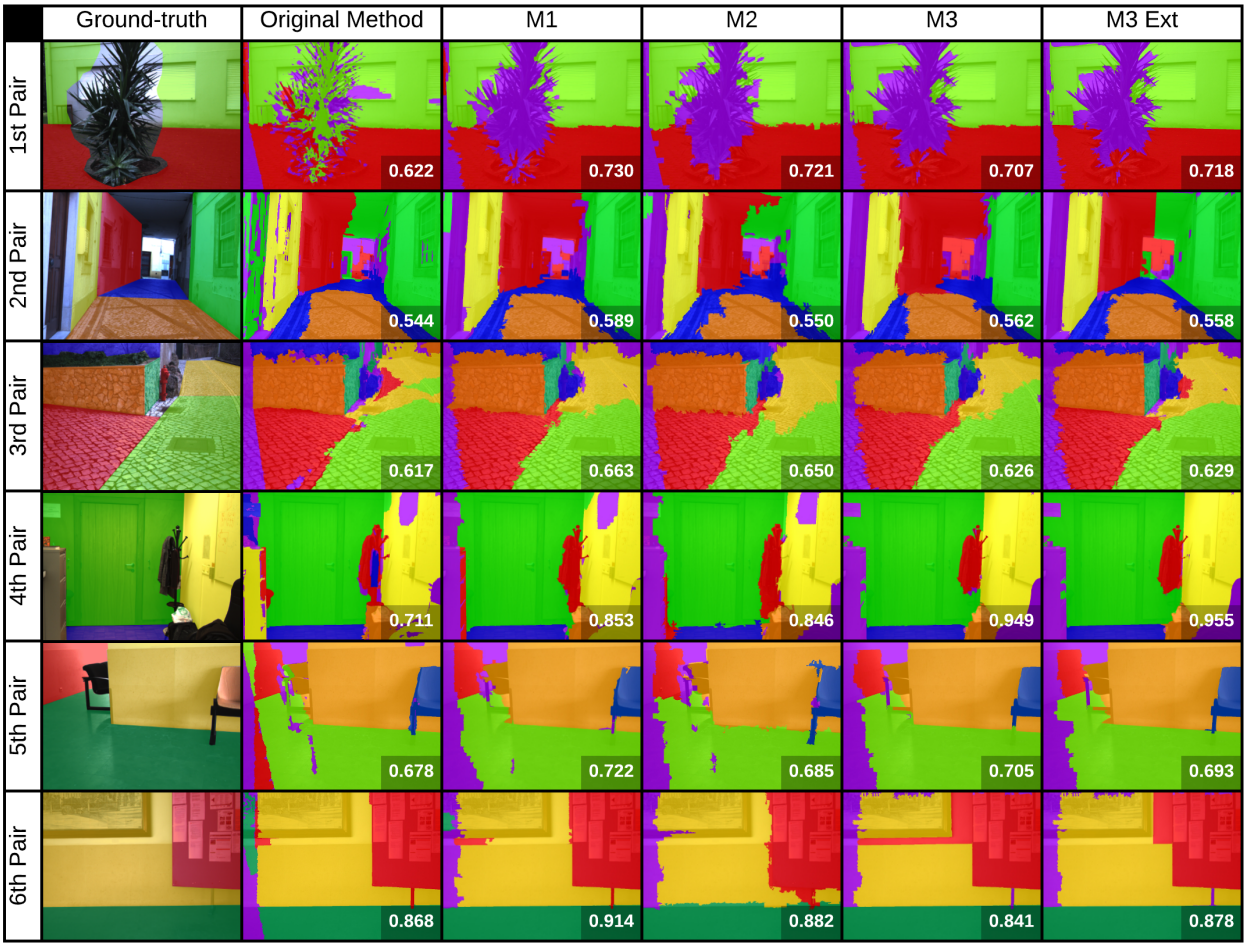


Figure 4.4: Comparative results between the proposed methods and the original.

4.3 Discussion

The discussion of the comparison results is divided into two different fields: the computational time and the quality of the image segmentation in planes.

The Jaccard index provides an information about the similarity between the ground-truth and the evaluated labellings, but the index only suffers significant variations when there are expressive errors in the labelling. This fact can be observed in the results shown in figure 4.4, that contains not only the labelling provided by each method but also its Jaccard index, where similar Jaccard indices are obtained for the labellings provided by the four proposed methods. The original method is the one that typically provides the lowest values for the Jaccard index, demonstrating its lower accurary when compared to the proposed approaches.

The original method has a prohibitive computational time to include in a VSLAM pipeline, taking about 28.296±9.084 s to label all the image pixels. It commonly performs good and fine segmentations, delimiting very well the transitions between planes in the image, but some times the fine treatment of the images causes gaps in poorly textured regions (NCC is not defined) or plane assignments in non-planar surfaces (ex. image pair 1). On the other hand, the proposed superpixelwise MRFs achieve similar or better results than the original, but with significantly lower computational times.

Method M1 (subsection 4.1.1) provides good results in the delimitation of planes. Its main advantages are the reduction of gaps in the labelling, caused by the adaptation of the data cost function (D_p), the discard label assignment to non-planar regions (purple colour in the labelling images) and the computational time reduction. This method completes the label assignment, on average, in 3.081±0.768 s, being about 9 times faster than the original method.

Method M2 (subsection 4.1.2) is a completely novel approach in the usage of superpixels. The idea is interesting, but reveals poor performance in the delimitation of planes in the image. This problem arises due to the inconsistent and imprecise location of the edges in the fusion images. Additionally, the computational time is a little worse than method M1. Thereby, this method performs the labelling task in 3.142±0.835 s, 9 times faster than the original, but with poor quality.

Method M3 (subsection 4.1.3) introduces a new way to compute the data cost function (D_p), that accelerates the optimization process. It has generally a good performance, but in certain cases the transition between planes is incorrect (ex. image pair 2). However, the computational performance is clearly better than the original. This method labels all the

pixels in the image in 0.664±0.047 s, or in other words, is 43 times faster than the standard MRF method.

Method M3 Ext (subsection 4.1.4) is an extension of method M3 created to overcome the incorrect transitions between planes. The problem is solved (image pair 2), but the computational time is a little bit higher than in method M3. Since the quality of segmentation is superior, this is considered the best method. Introducing this improvement, the labelling is realized in 0.707±0.113 s, being 40 times faster than the original MRF.

The last discussed method (method M3 Ext) presents the best segmentation results with the second best computational time, being very close to the fastest method. Therefore it was selected to replace original MRF formulation of the pipeline.

5 Architecture of the πMatch App (ongoing development)

The construction of an application is the best way to allow an easy experimentation of the πMatch pipeline, since it aggregates all modules. This chapter describes the application, detailing the multi-thread and data structure of the program, the visualization of the PPR and the functionalities provided by the application.

5.1 General Architecture

In this work, the application is implemented using the Unity 3D to construct the user interface and visualization of the C++ pipeline results. Thereby, the task requires the communication between C++ and C#, the language used in the Unity 3D. In order to achieve the best performance in the overall pipeline, it is organized in a multi-thread structure implemented in C++:

- **Acquisition Thread**: this thread acquires images with high frame rates to provide a continuous visualization of the scene captured by the camera, with only a small subset of the frames being processed;
- **Distortion Removal and ACs Extraction Thread**: this thread receives frames from the acquisition thread. Since the pipeline takes as input images without distortion, it warps the images to correct them and uses the non-distorted images in the ACs computation module;
- **Plane-based Structure from Motion (SFM) Thread**: the SFM thread receives the previously computed ACs and estimates the camera motion and the planes;
- **PPR Discrete Optimization Thread**: this thread is an optional part of the thread structure that can be enabled or disabled. This thread optimizes the planes received from the SFM thread, using a sliding window relative to every 5 consecutive frames;

- **MRF Segmentation Thread**: the MRF thread performs the segmentation of the image in planes and creates the textures to apply to each one, using the data provided by the SFM thread, if the Discrete Optimization thread is disabled, or by the Discrete Optimization thread, if it is enabled.

In order to correctly transfer the data between threads, the definition of four buffers was needed:

- A LIFO buffer to transfer images between the acquisition and the ACs extraction threads;
- A FIFO buffer to pass the ACs returned by the ACs Extraction thread to the SFM thread;
- A FIFO buffer to accumulate the camera motions and planes estimated in the SFM thread to feed the Discrete Optimization thread;
- A FIFO buffer to pass images and the SFM information from the SFM or Discrete Optimization thread to the MRF thread.

In the unity, it is intended to visualize the computed camera poses and planes with the respective textures and limits. So a data structure to store all this information was defined. The details of the visualization in unity are explained in the next section. The figure 5.1 depicts the global structure of the application.

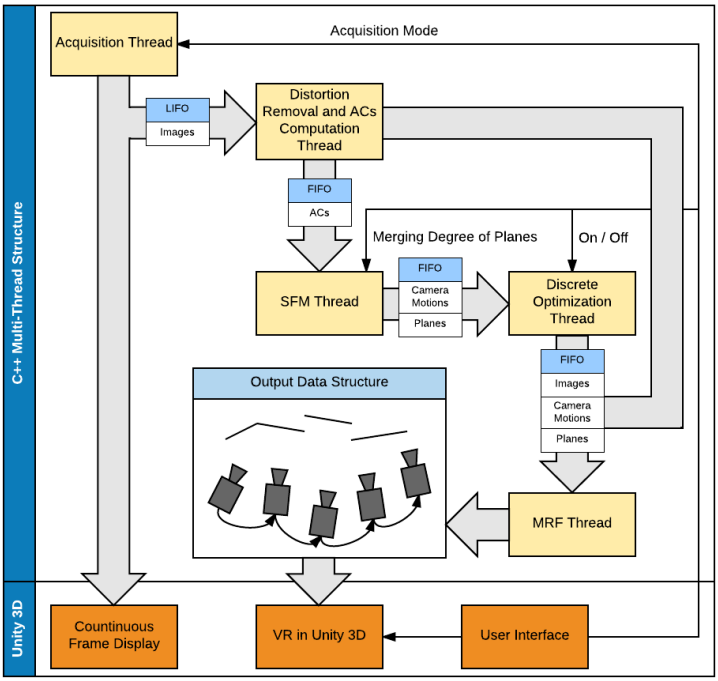


Figure 5.1: Schematic representation of the application architecture.

5.2 Virtual Reality in Unity

The Unity 3D visualizer is the part of the application that performs a 3D graphical representation of the data structure obtained using the pipeline. The data structure was devised to efficiently use the tools provided by the Unity 3D game engine. Thereby, it stores four 3D points to delimit the regions of interest of each plane, an RGBA image per plane to store its texture and the camera poses per image pair.

The computation of the remaining elements, other than the camera poses, is performed as follows. Firstly, the computation of RGBA textures starts with a modification in the initial labelling image returned by the MRF. Initial labelling is used to reconstruct the 3D points and to detect the points with a distance to the camera larger than a threshold. Consequently, the labelling of the corresponding pixels of these points is changed to the discard label. Using the modified labelling, the bounding box of each label is defined and the vertices are reconstructed in the 3D space. The texture images are created with the bounding boxes and the alpha channel is used to force the transparency of the pixels inside them with incorrect labelling.

In the Unity engine, the four points and the texture define a textured quadrilateral in the space (the reconstructed plane) and the camera poses allow the representation of the camera in the reconstruction. The application takes advantages of the Unity engine to perform the visualization.

5.3 Description of the Final Application

The πMatch application incorporates the C++ improved implementation of the original pipeline, allowing an easy experimentation of the algorithm. The application has an user interface to provide a direct utilization of the pipeline with a standard USB camera. This user interface allows to configure the operation of some pipeline modules:

- **Configuration of the acquisition module**: the image acquisition can be done by loading images from file, capturing on demand or capturing as fast as possible. In case of selecting the load from file mode, the user can specify the file where the desired images are stored. On the other hand, if one of the other methods are selected the user has the possibility of choosing the camera and the image size.
- **Configuration of the ACs extraction module**: provides the possibility of defining

the maximum number of outputted ACs.

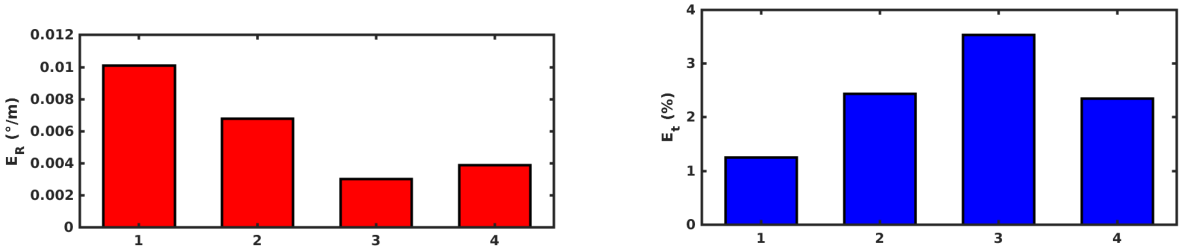
- **Configuration of the plane merging and refinement module**: the plane merging degree is configurable using an integer parameter between 1 and 10.
- **Configuration of the multiview discrete optimization module**: the module can be enabled or disabled by the user.

Apart from the configuration of the pipeline modules, the application also allows to load the camera calibration from a file or to perform the calibration inside the application. During the execution of the pipeline the user can alternate between continuous camera image display, reconstructed model display and a display mode that fuses the previous two. After the conclusion of the reconstruction, the user can freely navigate in the reconstructed environment.

6 πMatch Results

The proposed improvements to the pipeline combined with the developed thread system yield a significantly faster pipeline than the original one. Thus, experiments on four sequences of the KITTI dataset were done to demonstrate the computational time results and the accuracy of the camera motion and scale estimation for large image sequences. In these experiments, the discrete optimization of planes is used to improve the piecewise planar reconstruction and the ACs extractor is configured to divide the image into 16 blocks, limiting maximum number of outputted ACs to 800, for computational efficiency.

In order to numerically evaluate the global performances for the analysed sequences, the average rotation (E_{R}) and translation ($E_{\mathbf{t}}$) errors are computed using error metrics proposed in [41] and used in [25], in the same type of experiments. Figure 6.1 presents these results, for each sequence.



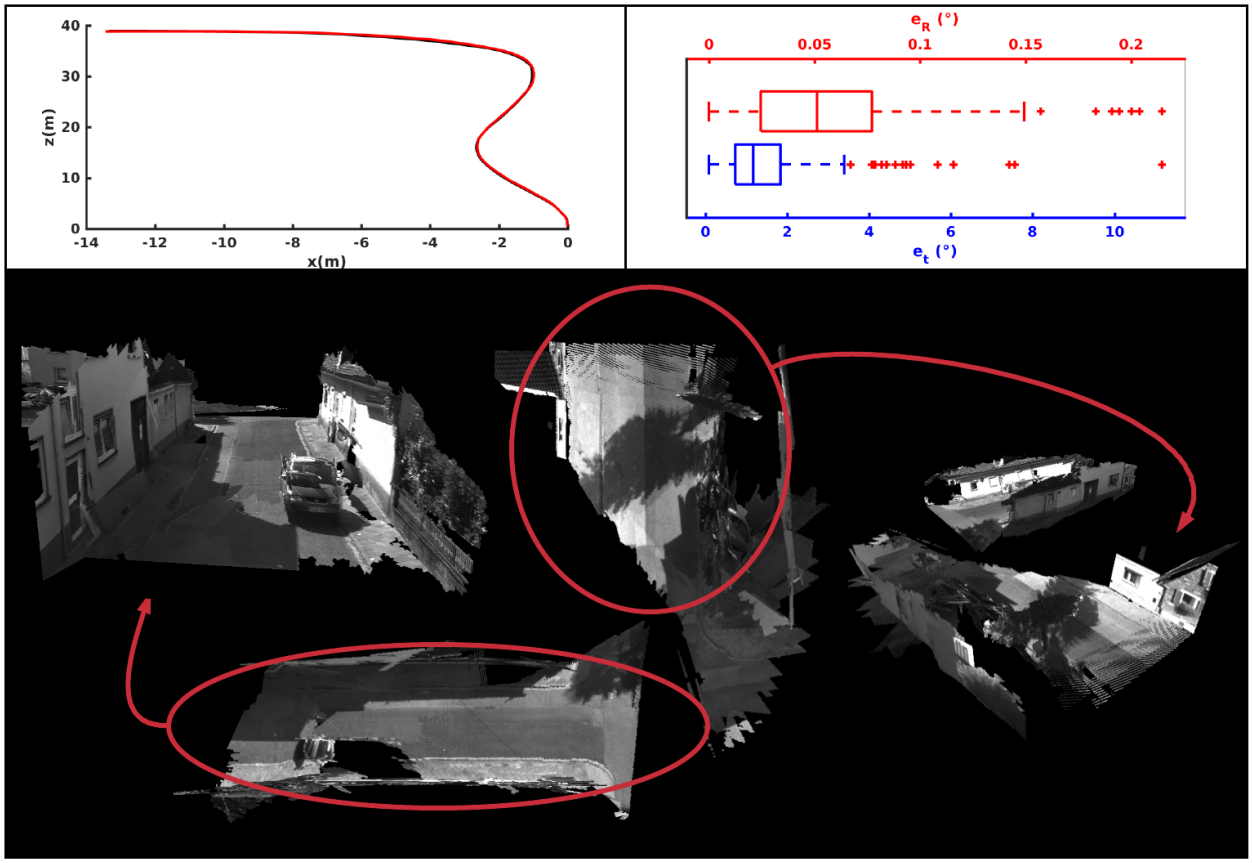
(a) Global rotation errors (E_{R})

(b) Global translation errors ($E_{\mathbf{t}}$)

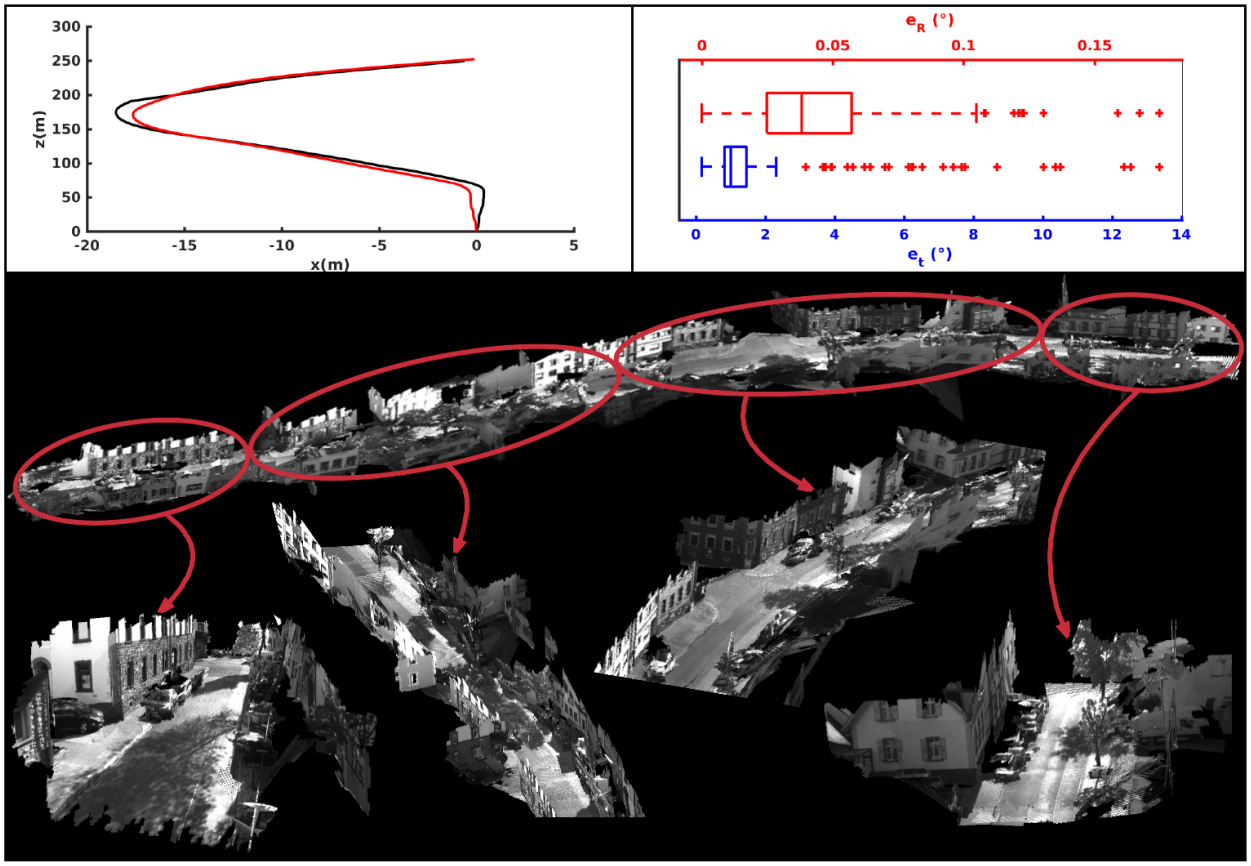
Figure 6.1: Global errors in rotation and translation per sequence.

Figures 6.2a, 6.2b, 6.2c and 6.2d confront the ground-truth and estimated trajectory, present the distribution of errors in translation ($e_{\mathbf{t}}$) and rotation (e_{R}) for each pair, in degrees, and the final reconstruction of the four selected sequences. Using the new version of the pipeline, inserted in a multi-thread architecture, it achieves an average computational times of 1.151 s per frame, being several times faster than the original, presented in section 2.3. If the discrete optimization of planes and the MRF segmentation steps are removed, the method achieves an average computational time of 0.719 s per frame.

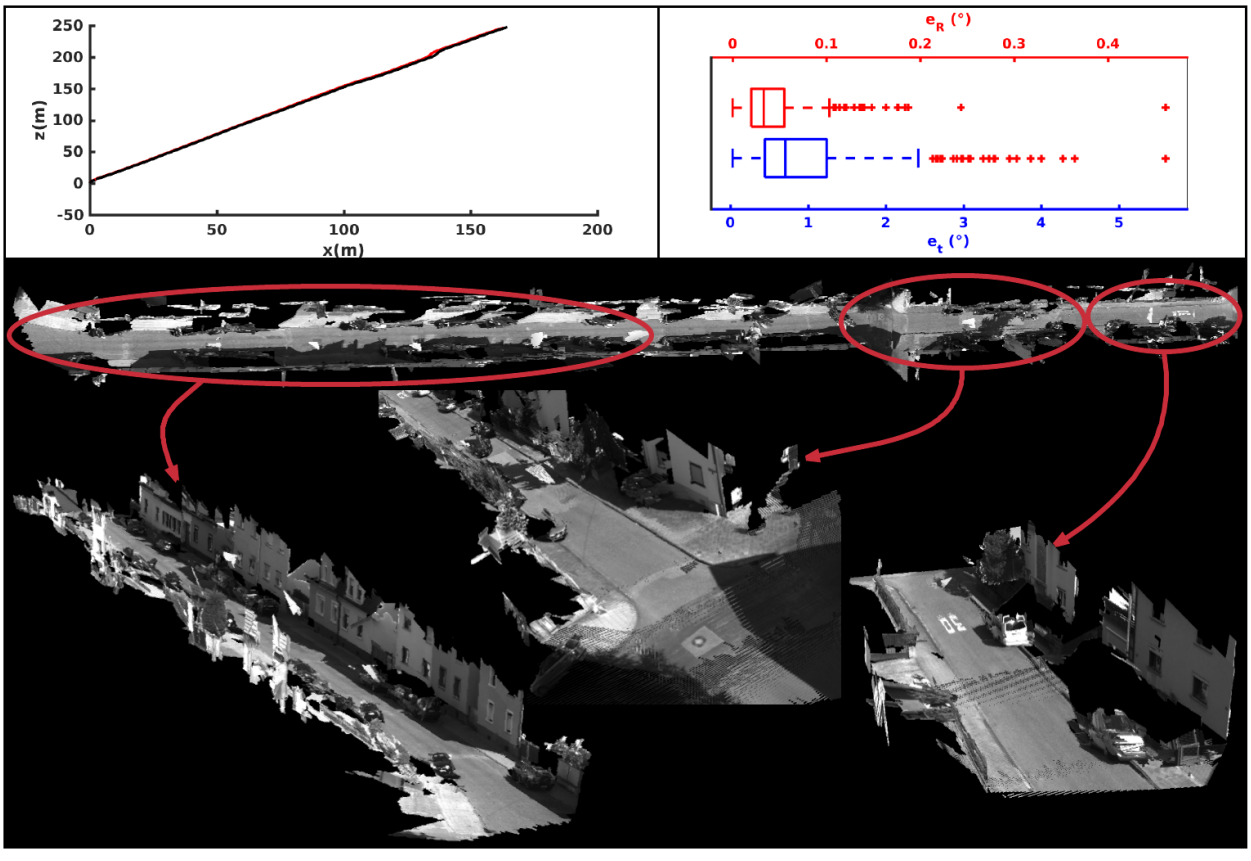
The global motion errors, shown in figure 6.1, the motion errors per pair and trajectory comparisons, presented in figure 6.2, demonstrate that the camera motion and the scale of translation are accurately estimated, being the errors very similar to the reported in [25] for the three first sequences. For the fourth sequence, the new pipeline version achieves significantly better results than the standard πMatch due to a more robust estimation of the relative scale. This indicates that the new AC extractor module, VLFeat Accelerated, is superior to the original one as it causes a significant decrease in the scale drift. Summarizing, the results support the quality of the new pipeline version, being several times faster than the original.



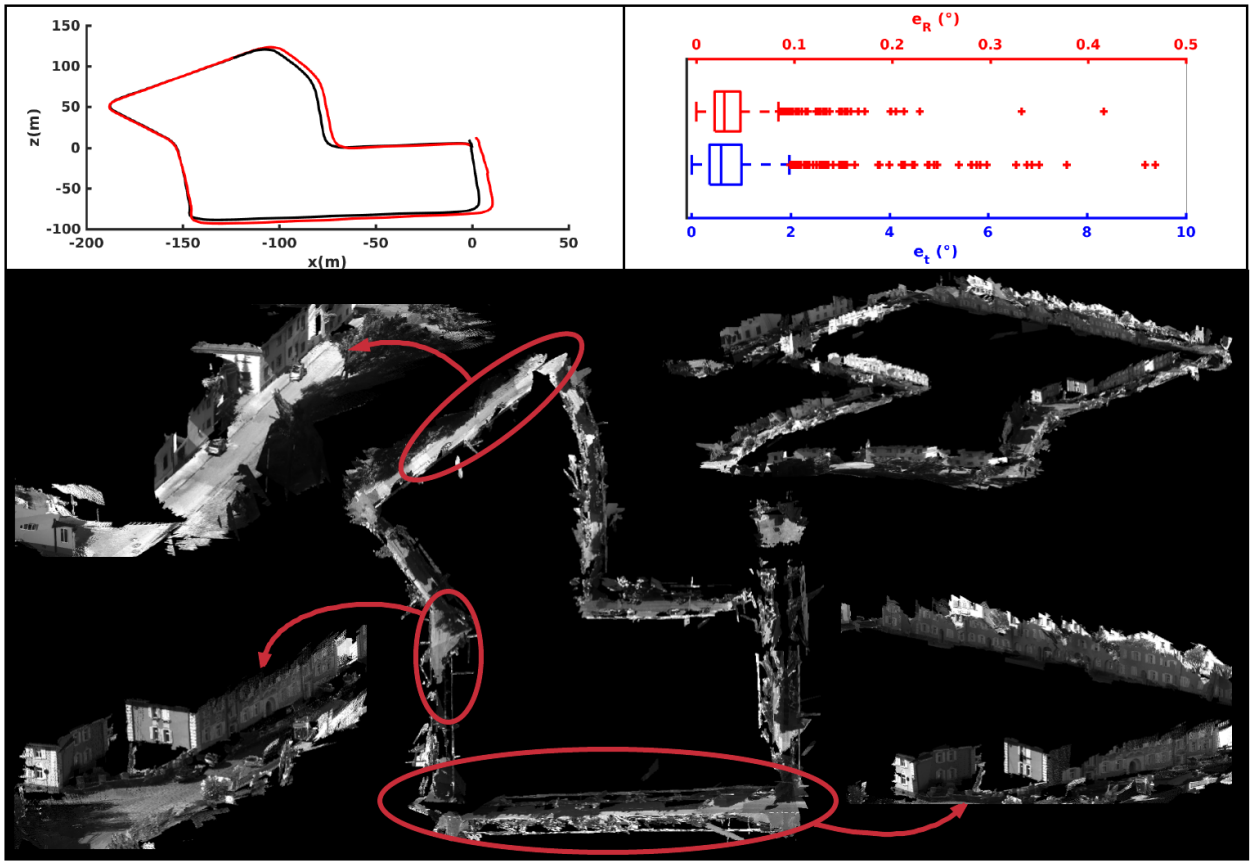
(a) Sequence 1: 125 frames



(b) Sequence 2: 268 frames



(c) Sequence 3: 395 frames



(d) Sequence 4: 1101 frames

Ground-truth πMatch Estimation

Figure 6.2: Comparison between the ground-truth and the estimated trajectory, projected on the ground plane (top left). Boxplots with the distribution of rotation and translation errors per image pair (top right). The reconstruction of the sequences (bottom).

7 Conclusion and Future Work

The thesis proposes a C++ implementation of the πMatch with some improvements in the pipeline modules. The direct translation to C++ halved the computational time, on average, but its time performance (on average, 44.002 s) was completely inappropriate to the target near real-time application. In order to achieve reasonable time performances, the ACs extraction and MRF segmentation modules were identified as the pipeline bottlenecks and some solutions were proposed as hypotheses to substitute them.

The method proposed to substitute the ACs extraction module was denominated VLFeat Accelerated, because it uses the VLFeat library with some adaptations. It performs the establishment of ACs with a lower computational time and simultaneously guarantees the equally scattering of them in the whole image, being more robust to differences in the textures of the scene. The VLFeat Accelerated outperforms the original method in the quality of the extracted ACs, being 2.6 times faster, as demonstrated in section 3.2.

The novel proposed MRF-based piecewise planar segmentation method innovates in the use of superpixels to represent the image in the MRF formulation. This method relies on superpixels to reduce the number of nodes of the labelling problem and on the projection of the intersection lines between planes in 3D space to force the transitions of labelling. The novel method performs a reasonable segmentation of the image in planes, being 40 times faster than the original MRF formulation, as shown in section 4.2.

With these two improvements, the C++ version of the πMatch completely outperformed the original Matlab version, in terms of computational time. However the time performance was insufficient and thus a multi-thread architecture was designed to accelerate the execution of the global pipeline. The thread architecture was tested with experiments on KITTI dataset sequences, proving the good estimations of scale and camera motions, superior to the performances reported in [25]. In the multi-thread execution, the complete pipeline (with discrete optimization of planes and MRF segmentation) approximately process 1 frame per second and without the piecewise planar reconstruction, the pipeline reaches nearly 2 frames

per second.

Finally, an application was designed to introduce the pipeline, in order to allow an easy visualization of its results, using Unity 3D game engine. The πMatch application is currently on development.

In general, all the objectives of the thesis were accomplished, as resumed above. However, they were modified during the execution, since some unexpected problems arose during the code translation, like the inappropriate computational times of the ACs extractor and MRF segmentation modules. In this sense, other problems like improvements in the scale estimation and to operate in multiple motion environments were not analysed.

The proposed C++ implementation of πMatch contributes to an enhancement of the original pipeline, but there are other possible modifications and improvements to be suggested and tested. In this regard, some of them are:

- the usage of KLT feature tracker in the ACs extraction module, avoiding to perform feature extraction in all images;
- the implementation of a linked-list structure for the identification of planes that are shared among views, allowing their back-propagation to previous frames. In particular, when a new plane substitutes an existing plane in the current optimization window, the substitution is performed for all frames the existing plane appears in, even the ones that are outside the discrete optimization window;
- the enhancement of the scale estimation module;
- the implementation of a MRF Multiview to improve the piecewise planar reconstruction.

8 Bibliography

[1] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot, 12 1991.

[2] H Zhao, M Chiba, R Shibasaki, X Shao, J Cui, and H Zha. SLAM in a dynamic large outdoor environment using a laser scanner. In *IEEE International Conference on Robotics and Automation (ICRA)*, number 3, pages 1455–1462, 2008.

[3] Marco Baglietto, Antonio Sgorbissa, Damiano Verda, and Renato Zaccaria. Human navigation and mapping with a 6DOF IMU and a laser scanner. *Robotics and Autonomous Systems*, 59(12):1060–1069, 2011.

[4] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[5] Henning Lategahn, Andreas Geiger, and Bernd Kitt. Visual SLAM for autonomous ground vehicles. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1732–1737, 2011.

[6] German Ros, A Sappa, Daniel Ponsa, and Antonio M Lopez. Visual slam for driverless cars: A brief survey. In *Intelligent Vehicles Symposium (IV) Workshops*, volume 2, 2012.

[7] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual Navigation for Mobile Robots : A Survey. *J. Intell. Robot Syst.*, 53:263–296, 2008.

[8] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[9] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2320–2327, 2011.

[10] Jakob Engel, Thomas Sch, and Daniel Cremers. LSD-SLAM: Direct Monocular SLAM. *European Conference on Computer Vision (ECCV)*, 8690 LNCS(PART 2):834–849, 2014.

[11] Jakob Engel, Vladlen Koltun, and Daniel Cremers. *Direct Sparse Odometry*. Number October. 2016.

[12] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.

[13] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[14] Raul Mur-Artal, J. M.M. Montiel, and Juan D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[15] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct SLAM for omnidirectional cameras. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2015-December, pages 141–148, 2015.

[16] Jakob Engel, Jörg Stückler, Daniel Cremers, Jorg Stuckler, Daniel Cremers, Jörg Stückler, and Daniel Cremers. Large-scale direct SLAM with stereo cameras. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015.

[17] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras, 2017.

[18] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 15–22, 2014.

[19] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 127–136, 2011.

[20] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[21] Christian Kerl, Jorg Stuckler, and Daniel Cremers. Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 11-18-December-2015, pages 2264–2272, 2016.

[22] Carolina Raposo, Michel Antunes, and Joao P. Barreto. Piecewise-planar StereoScan: Structure and motion from plane primitives. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8690 LNCS, pages 48–63, 2014.

[23] Michel Antunes, João P. Barreto, and Urbano Nunes. Piecewise-planar reconstruction using two views. *Image and Vision Computing*, 46:47–63, 2016.

[24] Carolina Raposo, Michel Antunes, and Joao P Barreto. Piecewise-planar stereoscan: Sequential structure and motion using plane primitives. PP:1–1, 08 2017.

[25] Carolina Raposo and João P Barreto. *πMatch: Monocular vSLAM and Piecewise Planar Reconstruction Using Fast Plane Correspondences*, pages 380–395. Springer International Publishing, Cham, 2016.

[26] Carolina Raposo and Joao P. Barreto. Theory and Practice of Structure-From-Motion Using Affine Correspondences. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5478, 2016.

[27] Andrea Vedaldi and Brian Fulkerson. VLFeat - An open and portable library of computer vision algorithms. *Design*, 3(1):1–4, 2010.

[28] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science (New York, N.Y.)*, 315(5814):972–976, 2007.

[29] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. *Sophia*, 6303(6303):90, 2007.

[30] Du Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[31] Andrew Delong, Anton Osokin, Hossam N. Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

[32] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*, volume 2. 2003.

[33] David Nistér and Henrik Stewénius. Linear time maximally stable extremal regions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5303 LNCS, pages 183–196, 2008.

[34] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3951 LNCS, pages 404–417, 2006.

[35] Linear time maximally stable extremal regions implementation. `https://github.com/idiap/mser`. Accessed: 2017-08-17.

[36] Surf. `https://github.com/abhinavgupta/SURF`. Accessed: 2017-08-17.

[37] András Bódis-Szomorú, Hayko Riemenschneider, and Luc Van Gool. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 469–476, 2014.

[38] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2281, 2012.

[39] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc Van Gool. SEEDS: Superpixels Extracted Via Energy-Driven Sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015.

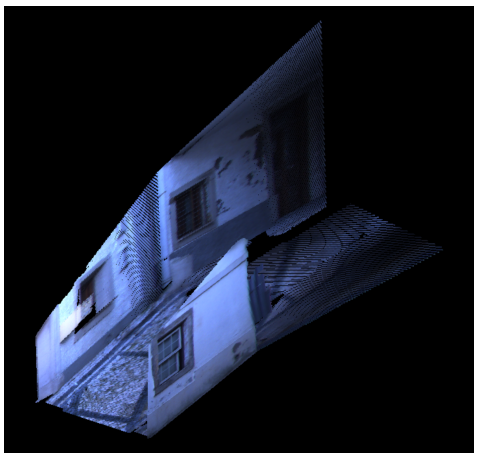
[40] Peer Neubert and Peter Protzel. Compact watershed and preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms. In *Proceedings - International Conference on Pattern Recognition*, pages 996–1001, 2014.

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

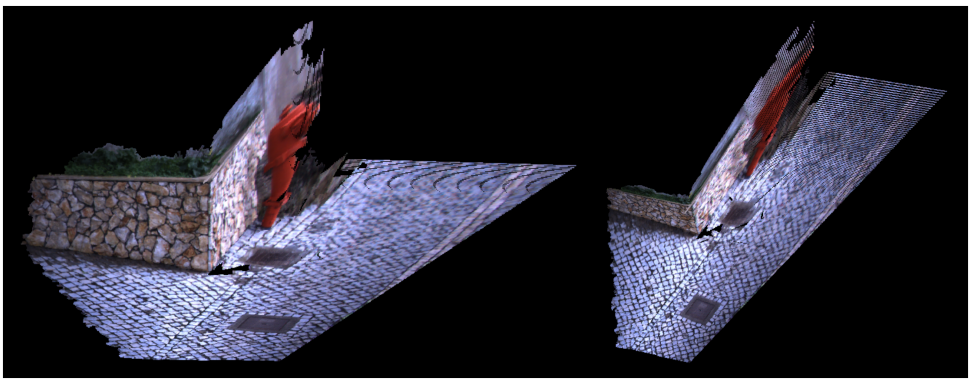
Appendix A

PPR of the Test Image Pairs

(a) Image pair 1.

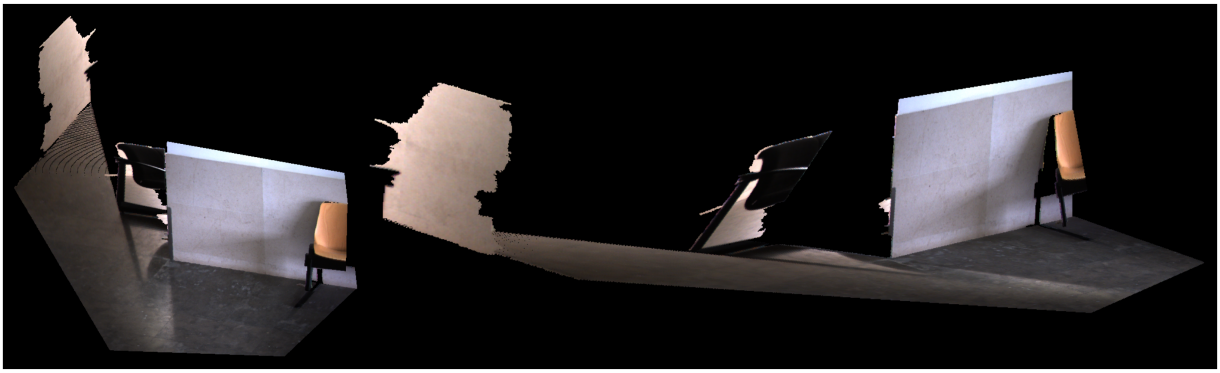


(b) Image pair 2.

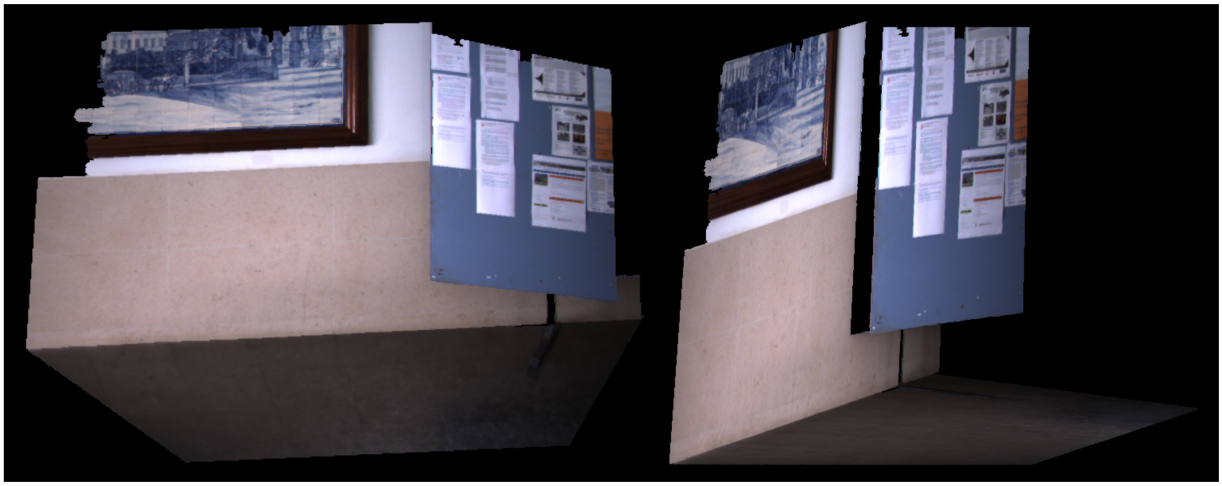


(c) Image pair 3.

(d) Image pair 4.



(e) Image pair 5.



(f) Image pair 6.

Figure A.1: Reconstruction of the six test image pairs.