ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

VISUAL ODOMETRY AND MAPPING USING STEREO VISION AND REGISTRATION ALGORITHMS

Autor: Álvaro Arranz Domingo

Directores: Álvaro Sánchez Miralles

Arturo de la Escalera Hueso

Madrid
Abril 2015

Resumen

Esta tesis propone un sistema completo de odometría visual utilizando únicamente Visión por Computador y análisis estéreo. El documento aborda el problema de localización analizando tres de sus pasos fundamentales: comienza con las etapas iniciales del análisis estéreo, continúa analizando la fase de refinamiento y finaliza con la fase de odometría visual propiamente dicha. De esta manera, el resultado de la tesis es un sistema formado por tres propuestas novedosas para los tres bloques que forman un sistema de localización mediante Visión por Computador.

En primer lugar, el documento realiza un análisis del estado del arte global, cuyo objetivo es ofrecer una visión general del problema, mostrar qué diferentes aproximaciones existen para resolver ciertos problemas y citar los sistemas y algoritmos más importantes encontrados en la literatura. Durante este estudio, se incluye el análisis de las técnicas de *Simultaneous Location and Mapping* (SLAM) debido a su íntima relación con los sistemas de odometría visual. Más allá de ser una simple descripción del estado del arte, se realiza un análisis y clasificación detallado de las diferentes soluciones encontradas, describiendo similitudes y diferencias en los pasos que sigue cada una de ellas. Como resultado se obtienen varias tablas en las que se resume qué pasos siguen y qué técnicas utilizan los algoritmos más importantes de la literatura.

El documento continúa proponiendo un *marco multiresolución de minimización de energía* para el problema de correspondencia estéreo. Este marco aborda el problema del cálculo de disparidades utilizando diferentes resoluciones al comienzo y al final del algoritmo. Iniciar la estimación estéreo con un par de imágenes de alta resolución es muy positivo para obtener una alta definición en el campo de las disparidades, pero aumenta el tamaño del problema. Por el contrario, si deseamos disminuir el tamaño del problema y se decide disminuir la resolución inicial del par estéreo, la definición en la dimensión de las disparidades quedará restringida. Para solucionar este problema, se propone un algoritmo que admita como entrada imágenes de alta resolución, y obtenga como salida imágenes de resolución reducida pero manteniendo la definición en la dimensión de las disparidades como si fuesen de alta resolución. Se demuestra la importante disminución de la complejidad del problema comparado con el caso clásico de cálculo estéreo con imágenes de alta resolución. El análisis se realiza utilizando técnicas estéreo globales basadas en *Modelos Aleatorios de Markov* (MRF).

Después del análisis estéreo, se propone una forma innovadora de abordar el

problema del refinamiento de los mapas de disparidades obtenidos previamente. Para ello, se propone utilizar un algoritmo genético como optimizador de una función de energía representativa de la calidad de las correspondencias. Desde este punto de vista, la solución es similar a otras formas de optimización propuestas en la literatura para minimizar MRFs. A pesar de no garantizar un mínimo global, la utilización de algoritmos genéticos abre la posibilidad de utilizar cualquier tipo de función de energía sin restricciones. Ésta característica es una aportación importante comparada con el resto de algoritmos del estado del arte, que están limitados a ciertas funciones de energía. A lo largo del documento se proponen diferentes operadores genéticos adaptados al problema de análisis estéreo, se proponen diferentes funciones de energía y se comparan entre ellas. Además, se describe su implementación en una unidad gráfica de proceso (GPU) para aprovechar su naturaleza paralelizable y se muestran los beneficios que se pueden obtener en su tiempo de ejecución.

Finalmente se propone un algoritmo de odometría visual diseñado para modelar escenas no estáticas. Basándose en algoritmos clásicos de registrado de nubes de puntos como el *Iterative Closest Points* (ICP), el algoritmo presentado realiza una clasificación de los puntos en tres dimensiones según su coherencia después de una transformación (rotación más traslación). De esta forma, utilizando un sistema de votación se detectan cuáles son las transformaciones más recurrentes y se clasifican los grupos de puntos en consecuencia. Esto permite detectar y eliminar elementos en movimiento para una estimación más precisa del movimiento de la cámara. Esto distingue al algoritmo frente a la gran mayoría de los existentes en el estado del arte, que contemplan los objetos en movimiento como *outliers* y es quizá la aportación más importante del algoritmo. Además, esta eliminación abre la posibilidad de calcular mapas de entorno incluyendo únicamente elementos estáticos evitando así el efecto *sombra*. Por último se analiza el rendimiento de los algoritmos clásicos de registrado como el ICP con nubes de puntos poco precisas como las obtenidas mediante algoritmos estéreo.

Abstract

This thesis proposes a complete visual odometry system using only Computer Vision techniques and stereo analysis. The document addresses the pose estimation problem studying three of its fundamental steps: begins with the early stereo analysis stages, continues with the refinement phase and finally finishes with the analysis of the visual odometry algorithm itself. Therefore, the result of this thesis is a complete system constituted from three novel algorithms for the three stages that forms Computer Vision pose estimation system.

Firstly, the document globally analyses the state of the art of pose estimation algorithms with the aim of offering a general view of the problem, shows the different existing approaches for solving certain problems and cites the most important systems and algorithms found in the literature. This analysis includes the *Simultaneous Location and Mapping* (SLAM) techniques given that they are closely related to the visual odometry systems. However, it is not a mere state of the art description. It also realizes a detailed taxonomy and classification of the different solutions found, describing their differences and similarities for each of their steps. As a result, various tables resuming the steps and techniques of the most important algorithms in the literature are obtained.

The document continues proposing a *multiresolution energy minimization framework* for the stereo correspondence problem. This framework deals with the problem of estimating the disparities using different image resolutions at the beginning and the end of the algorithm. Using high resolution images as stereo pairs is very adequate for obtaining also a high definition in the disparity dimension. However, this configuration leads to an important increment of the problem's size. On the contrary, if the size of the problem is meant to be reduced and a decimation of the input stereo pair is performed, then the disparity resolution is also decimated, which leads to bad disparity estimations. In order to solve this problem, an algorithm using high resolution stereo images as input and obtaining lower resolution disparity maps but maintaining the original depth resolution is proposed. The document shows the important reduction of the problem's size compared to classical high resolution stereo pair inputs. For this analysis, global algorithms based on *Markov Random Fields* (MRF) are used.

After the stereo analysis, a novel disparity map refinement process is proposed.

A genetic algorithm is used for minimizing an energy function modeling the stereo correspondence fitness. This approach is similar to other techniques found in the literature, in the sense that an energy function based on an MRF is meant to be minimized. In spite of finding a global minimum is not guaranteed, the utilization of genetic algorithms gives the opportunity of using any kind of energy function without any restriction. This characteristic is an important contribution compared to most algorithms found in the literature, given that they are usually restricted to some forms of energy functions. During the document, the most important genetic operators are adapted to the stereo matching problem. Moreover, different energy functions dealing with occlusions are proposed and compared. A *Graphics Processing Unit* (GPU) implementation using CUDA is also explained in detail, in order to demonstrate the parallel nature of the algorithm and the performance boost that could be obtained using this hardware units.

Finally, a visual odometry algorithm capable of dealing with non-static scenes is proposed. Based on classic point-cloud registering algorithms found in the literature such as *Iterative Closest Points* (ICP), the proposed algorithm performs a three dimensional point classification based on its coherence after certain transformations (translation and rotation). That way, using a voting system, the most frequent transformations are detected and points are classified forming groups that fulfill rigid bodies constraints. This classification permits to eliminate moving objects from the scene and perform a better camera motion estimation. This classification distinguishes this algorithm from others in the literature and is probably the most important contribution along this final odometry step. Besides, these objects removal enables to map the scene including only the static environment, avoiding the *shadow* effect. Finally, the accuracy of the classic point-cloud registering algorithms using clouds obtained from stereo algorithms is analyzed.

Contents

List of Tables

List of Figures

1 Introduction

The aim of Computer Vision is to understand the color information present in the images captured by any device, such as video-cameras. Although it seems an easy task, mainly because any human can do it instinctively, it has showed up to be a very difficult problem. Depending on the purpose of the visual analysis, many different algorithms have appeared for motion estimation, object recognition, tracking and reconstruction, stereo vision, image filtering, etc. All these fields have received intensive attention of the researchers during the last decades. However, the development of these fields are generally not mature enough and the solutions found cannot compete with the capabilities of a human being. Many problems are still waiting for better solutions, and a lot of contributions can still be made.

This thesis will focus mostly on the *motion retrieval* problem. This problem focuses on obtaining the real movement of a camera by means of only analyzing the images captured. Note that this task is remarkably well done by everyone of us in a daily basis. We can go to work knowing exactly where we are, without bumping into obstacles (mostly) and finally return home at the end of the day. If you analyze it, you will probably conclude that this is an extraordinary complex task: it involves recognition, tracking, depth estimation, navigation, etc. Actually, we can figure out our motion because we can analyze and recognize our environment. Thus, please think about the *motion retrieval* problem as an agglutination of small complex problems, each one affecting the final result.

The *motion retrieval* problem has many applications, for instance, in robotics and in computer generated graphics (CGI) communities. However, the experiments shown along this document will be mostly related to the robotics research field.

This first chapter will describe the scope of the thesis in section 1.1, its main objectives in section 1.2 and the document global outline in section 1.3.

1.1 Scope of the thesis

Many different approaches can be found in the literature in order to solve the *motion retrieval* problem. The title of this thesis was carefully selected to emphasize three important words: *stereo*, *registration* and *mapping*. These three words establish the main research path followed during all the document.

Firstly, this thesis will be focused on using only stereo vision sensors and algorithms. Stereo vision, although being more expensive than monocular vision, enables simple metric reconstruction algorithms. Secondly, stereo cameras will be used as if they where range scanners. The motion estimation algorithm will be based on 3D point-cloud iterative registration algorithms. This decision can be justified by the good results obtained lately by range RGB-D sensors, which arises questions such as if stereo sensors are able to achieve similar results. Finally, the environment information is crucial, so especial efforts in obtaining final scene's metric maps were made.

1.2 Thesis objectives

The aim of this work is to develop a complete SLAM system using only a stereo camera and Computer Vision techniques, designed for real, non-static environments. In order to achieve this goal, several intermediate objectives and tasks have been defined to deal with the problem in several steps:

- Understand the SLAM problem using Computer Vision, both stereo and monocular, and elaborate a *taxonomy* with the most important techniques and works found in the literature. The step will provide a wide view of the main techniques used and the differences between them.
- As the first step for the system, propose an stereo algorithm using global techniques. Given that the global algorithms found in the literature are generally very slow, the main concern of this step is to improve its performance.

- Develop a final refinement step for the stereo algorithm developed in the previous objective. This step is considered to be crucial to improve considerably the quality of the stereo reconstruction and therefore, to boost the quality of the final SLAM solutions.

- Finally, as the core of the system, the last objective is to develop a SLAM algorithm that uses the stereo reconstruction obtained in previous steps to deduce the movement of the camera while calculating the shape of the environment. The main constraint of this step is that it should be able to cope with real environments that contain non-static objects.

1.3 Document outline

In order to address all the challenges stated in the previous section, this thesis is organized in six different chapters, generally, each one focusing on one of the objectives aforementioned. Each chapter has entity on its own and can be read and understood individually. However, to follow the natural design of the complete system, it is recommended to be read in order. Each chapter will focus on one or many of the steps carried out for obtaining the camera's poses from stereo images. The steps considered are described in Figure 1.1.

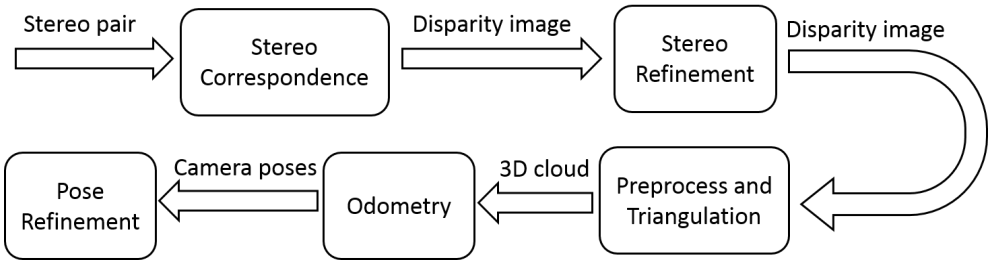

Figure 1.1: Steps for camera pose estimation from stereo input

This section briefly describes each chapter, how it fits into the pose estimation problem and its main contributions.

Chapter 2 contains the most important preliminaries of techniques and algorithms that will be covered around the whole document. The purpose of this chapter is twofold: first, it can be used as an introduction to the Vision-SLAM research area and to establish a starting point for the thesis and the next chapters. Second, it contains an overview and classification of the most popular algorithms, followed

by an analysis of common steps found in similar approaches. Its contribution is the taxonomy, analysis and comparison of different SLAM or pose-estimation algorithms. Following the schema in Figure 1.1, this analysis focuses on the *Odometry* step. As a result, several tables with an overview of each algorithm are shown.

Chapter 3 focuses on the stereo matching problem and on the extraction of good quality disparity maps. Global algorithms are considered to be accurate, and are generally well positioned in the rankings. In this chapter, the complexity of global algorithms using *Markov Random Fields* (MRF) is analyzed. This chapter starts the analysis in the first box of Figure 1.1, the *Stereo Correspondence* step. As a result, a multiresolution energy minimization framework is proposed for reducing the complexity of a *Markov Random Field* for stereo correspondence.

Chapter 4 deals with the refinement process applied after the stereo matching algorithm. This process is crucial to improve the quality of the disparity map and avoid outliers in the next steps. A genetic algorithm is used during this refinement process, evolving a fitness function representing the quality of the reconstruction. In order to improve its performance, a GPU implementation using CUDA language was used to evaluate the improvements that could be obtained. This chapter focuses on the *Stereo Refinement* step in Figure 1.1. Its main contribution is the capability of using any unrestricted energy function.

Chapter 5 proposes a SLAM algorithm for obtaining the movement of the stereo camera while reconstructing the environment. The disparity maps obtained in the previous chapters are triangulated and filtered to obtain a 3D point-cloud representing the environment. This point-cloud is used with iterative algorithms in order to deduce the movement frame by frame. To deal with non-static environments, a segmentation of different moving objects is obtained by means of RANSAC and J-linkage algorithms. As shown in Figure 1.1, this chapter is focused on the three steps in the lower row, the *Preprocess and triangulation*, the *Odometry* and the final *Pose Refinement*. The contributions in this chapter are twofold: a transformation segmentation algorithm able to segment moving objects in the scene is proposed and the analysis of the performance of registration algorithms using state of the art stereo algorithms is presented.

Finally, **Chapter 6** concludes this document, outlining the most important results, conclusions and original contributions obtained during the rest of the chapters. New lines of research are finally suggested.

The main repositories used along this document are the standard Middlebury stereo database (http://vision.middlebury.edu/stereo/), mainly for the chapters related to stereo matching, and the KITTI repository (Geiger et al., 2012) for generally all other chapters.

2 Preliminaries

During the last decades, since the 1980's, Computer Vision has received an important focus of the research community. Several research areas appeared for dealing with different problems: recognition, segmentation, feature detection, stereo analysis, tracking, etc. Regarding robotics and pose recovery, the *simultaneous location and mapping* (SLAM) has become the main problem to solve. The aim of this section is twofold. Firstly, to make a brief introduction to the problem and outline the main tendencies recently found in the literature. Secondly, to classify the most important solutions and to analyze them by finding common structures. Each structure or step is then described in detail, determining the different techniques used and their purpose. As a result, three different tables are elaborated in order to summarize the analysis carried out along this chapter. Please note that this chapter is not meant to be an exhaustive survey, but a general overview.

The rest of this chapter is organized as follows: in section 2.1 a brief overview of the SLAM problem is presented and in section 2.2 a general classification and analysis is detailed.

2.1 Introduction to the SLAM problem

The SLAM problem has been intensively studied in the literature. Several surveys have tried to review the most important solutions presented for this problem. In (Desouza and Kak, 2002), a complete survey of vision-based navigation algorithms aimed for indoor and outdoor robotics is presented. Although the *navigation* term

appears in the title, and indeed is one of the main interests for the authors, the SLAM problem is analyzed in order to obtain the robot's localization. This survey covers the existent algorithms until the 90's decade. More recently, (Bonin-Font et al., 2008) extended the analysis of (Desouza and Kak, 2002) updating the survey with the more recent algorithms.

The literature covered by this document has been centered in the algorithms and solutions that permit the localization and mapping using vision as its main sensor. Many other SLAM algorithms are based on different sensors such as range scanners, sonar sensors, or the more recent RGB-D cameras. Most of them are solutions quite dependent of the sensor used, so this analysis will cover just algorithms that could be used in a system with just a stereo camera sensor.

The SLAM problem has been traditionally studied in different research fields. From the Computer Vision point of view, the problem was stated as the recovery of the three-dimensional structure of the environment while estimating the movement of the camera around it. The problem was denominated *Structure from motion* (SMF) and it was based on some peculiarities. Firstly, the systems were not meant to be run in real time because, from a Computer Vision perspective, it was meant for applications where the video scene was already filmed and it was thought as a part of a post-production step. Therefore, another peculiarity is that these algorithms are manly monocular and they do not use any other sensor (such as inertial, GPS, laser, etc.) for their estimations. Finally, another important characteristic of these algorithms is that they are not casual, i.e they can use future information, such as future recorded frames, for estimating the current state.

All these restrictions for the *Structure from motion* problem lead to solutions based mainly on *bundle adjustment* algorithms. The estimation problem is addressed as an optimization problem where the three-dimensional location of the camera and all the detected features are solved simultaneously for a group of frames in the scene. This technique poses some restrictions such as an static environments and small frame-to-frame displacements. Traditionally, these solutions are interested in the quality of the mapping as much as the quality of the camera's localization. A survey of bundle adjustment techniques can be found in (Triggs et al., 2000). Optimizations for the application of bundle adjustments techniques in the real-time camera pose estimation problem can be found in (Chiuso et al., 2002)(Mouragnon et al., 2006).

Another different research field focused on the SLAM problem is the vision-aided robotics community. Traditionally, the problem was to obtain the pose of the robot

as precisely as possible given the previous poses. Generally, the environment was used just for the localization task and not aimed to obtain a precise map of the environment. The problem was solved performing a feature detection followed by a feature matching step and some kind of Kalman Filter (KF) or any of its extensions. Its main advantages are that, if few features are tracked, it can run in real time and it can fuse measurements of other inertial devices or odometry provided by the robot.

Many algorithms that belong to this SLAM group were proposed in the literature and even nowadays new ones still continue to appear. One of the most famous is (Davison, 1998), where the author proposed a stereo-based system that used the feature detector proposed in (Shi and Tomasi, 1994) and an EKF for filtering. It also used an active approach for maintaining the most robust features. Recently, (Paz et al., 2008) estimated the 6DOF motion of stereo and monocular cameras. The novelty is the combination of 3D points and inverse depth points followed by an EKF and the inclusion of conditionally independent maps created for large-scale maps. In (Alcantarilla et al., 2010) an algorithm that uses visual odometry pose estimation as a prior in the prediction phase in an Extended Kalman Filter EKF-SLAM is proposed. SLAM techniques using EKF were successfully used also for monocular localization and mapping. In (Davison et al., 2007) the authors describe the *MonoSLAM* algorithm, probably one of the most important ones.

The *smoothing and mapping* (SAM) algorithm (Dellaert and Kaess, 2006) has also been used in the literature instead of the KF, e.g. for stereo underwater reconstruction (Beall et al., 2010).

This filter-based approach was traditionally denominated in the literature the *simultaneous localization and mapping problem* ($VisualSLAM$), although the aim of the SFM is basically the same with different constraints. Compared to the SFM, the real-time constraint of these algorithms made impossible to analyze the environment in detail with thousands of features, so a precise localization was prioritized over a complete environment description.

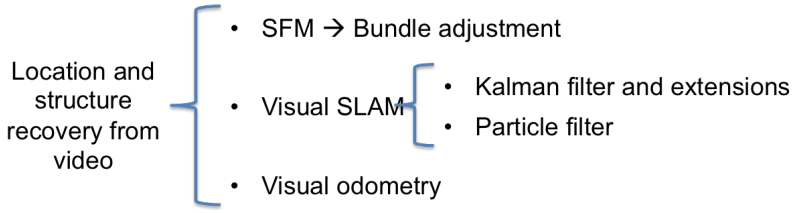

Figure 2.1: General classification of the $SLAM$ algorithms

As it can be seen, the aim of the algorithms is exactly the same in both SFM and $VisualSLAM$. Both families of algorithms perform a per-frame estimation of the camera location and a group of features in the environment. However, the application's restrictions of each problem led to different kind of solutions. Recently, the differences between the SFM and the $VisualSLAM$ have become smaller and more diffuse. Some analysis have been made for establishing the advantages and drawbacks of each one (Strasdat et al., 2010), and some other solutions have even unified both of them (Deans and Hebert, 2000).

In addition to the SFM and the $VisualSLAM$ algorithms, a different approach to the SLAM problem can be also found in the literature, the *visual odometry*. Given an ordered list of frames, its purpose is to find the transformation between each consecutive pair of frames independently. The final pose of the camera could be calculated as the multiplication of the estimated transformations from the beginning until the actual frame. As a dead reckoning algorithm, its main drawback is that it is prone to cumulative errors. For that reason, most algorithms use a final refinement process such as bundle adjustment, KF, loop closing algorithms, etc.

Visual odometry can be solved using any kind of feature detection and tracking. In (Nister et al., 2004) the authors proposed a visual odometry algorithm for stereo and monocular cameras that consisted of feature detection, matching and tracking followed by a preemptive RANSAC (Nister, 2003). In (Kaess et al., 2009), near degenerate cases are addressed by separating the rotation and the translation estimation processes. These cases are specially important because RANSAC-based approaches usually fail. Other visual odometry algorithms are (Geiger et al., 2011b) and (Comport et al., 2007), both performing a final KF filtering stage for pose refinement. Note that most algorithms consider the static-world assumption, dealing with moving objects using a robust estimator with high number of outliers such as RANSAC.

Figure 2.1 shows a general classification of the techniques aforementioned for pose and structure recovery using visual information.

2.2 Taxonomy

Once the general algorithm classification has been exposed in section 2.1, a more in-depth analysis can be performed. The purpose of this section is to classify and

divide the different methods found in the literature, obtaining the different building blocks for each algorithm's step. Each step will be analyzed and the most important algorithms related to this step will be described. This taxonomy turns out to be very useful for understanding the different steps that build up the most well known algorithms and highlighting the main core of each one. The reader is also invited to think about what building blocks could be interchanged and which algorithms fit best for their needs.

Figure 2.2 shows a general classification of the building blocks for a SLAM algorithm.

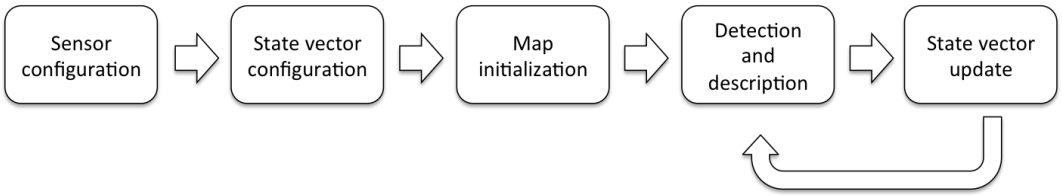

Figure 2.2: *SLAM* steps

The classification starts with the *sensor configuration*, determining if the algorithm is suitable for monocular or stereo cameras. The next step describes the form of the *pose* and *landmark* vectors. Some algorithms, generally monocular ones, need a *map initialization* process, so the next step deals with that task. Finally, the SLAM algorithms start a loop of *detection and description* followed by a *state vector estimation*, which generally is considered to be the very core of the SLAM algorithm.

The following sections will describe each step in greater detail.

2.2.1 Sensor configuration

Deciding the sensor configuration will have a great impact in both the kind of estimation algorithms and in the final cost of the system. Currently, the vision-based systems are:

- **Monocular camera**: This configuration is the simplest, least expensive and more versatile available. However, generally, the algorithms are more complicated and CPU-demanding. They also have the implicit problem of the scale estimation. In this group also appears the monocular wide-angle vision cameras, which facilitate the feature detection and tracking (Davison et al., 2004).

- **Stereo camera**: This configuration is more complex and expensive due to construction constraints and calibration, but it enables the use of stereo algorithms for depth estimation, and therefore simpler state estimation algorithms. The stereo calibration also solves the scale problem of the monocular cameras. Similarly to the monocular wide-angle vision cameras, some authors use the stereo version (Schleicher et al., 2006). The parallel configuration is the most common one, but other non-parallel configurations are also possible.

- **RGB-D camera**: Recently, the RGB-D cameras have appeared to add depth information to the classic color channels. Different underlying sensing mechanisms can be used, being the infra-red projected-light stereo the most famous due to the well-known Kinect sensor. This type of cameras are more precise estimating the depth than the stereo algorithms, but they can only be used in a restricted depth range. In addition, the sunlight and some surfaces can interfere in their estimations, making them impossible to use in outdoor applications.

Another camera configuration, used generally in vehicles for covering a wide view, is the various parallel monocular cameras with overlapping field of view. It is not a stereo configuration due to the overlapping is very little and it does not aims the depth estimation (Pollefeys et al., 2008).

2.2.2 State vector configuration

The state vector is composed of the variables that are to be estimated in the SLAM algorithm. Generally, the state vector is made of a group of variables related to the camera's state and other group related to the state of the landmarks or features. The state vector is defined by

$$x_t^s = \begin{pmatrix} x_t^c \\ x_t^l \end{pmatrix} \tag{2.1}$$

where x_t^s is the vector state, x_t^c is the state related with the camera and x_t^l is the state of the landmarks.

One may initially think that the camera's state vector x_t^c is formed by the position and orientation of the camera at time t. In a two dimensional world, the result is a state vector of three variables: two variables for position and one for orientation

(Equation 2.2). For a three dimensional world, the result is a state vector of six variables: three for position and three for orientation (Equation 2.3)

$$x_t^{c,2D} = \begin{pmatrix} x_t & y_t & \phi_t \end{pmatrix}^T \tag{2.2}$$

$$x_t^{c,3D} = \begin{pmatrix} x_t & y_t & z_t & \alpha_t & \beta_t & \gamma_t \end{pmatrix}^T \tag{2.3}$$

where x,y,z variables are the Cartesian coordinates, ϕ is the angle in two dimensions and α, β and γ are the euler angles.

However, it is usual to also model the linear and angular velocities including the temporal partial derivatives, Equation 2.4 and Equation 2.5.

$$x_t^{c,2D} = \begin{pmatrix} x_t & y_t & \phi_t & \dot{x}_t & \dot{y}_t & \dot{\phi}_t \end{pmatrix}^T \tag{2.4}$$

$$x_t^{c,3D} = \begin{pmatrix} x_t & y_t & z_t & \alpha_t & \beta_t & \gamma_t & \dot{x}_t & \dot{y}_t & \dot{z}_t & \dot{\alpha}_t & \dot{\beta}_t & \dot{\gamma}_t \end{pmatrix}^T \tag{2.5}$$

Regarding the landmark's state vector x^l, the easiest way to represent their state is just with their position (two or three dimensional), as shown in Equation 2.6. Given that the static-environment constraint is usually imposed, there is no need to model their velocities as previously happened with the camera state.

$$x_t^L = \begin{pmatrix} x_t^0 & y_t^0 & z_t^0 & x_t^1 & y_t^1 & z_t^1 & \dots & x_t^N & y_t^N & z_t^N \end{pmatrix}^T \tag{2.6}$$

where N is the number of landmarks analyzed.

A more complex representation of the landmark state is the *inverse depth parametrization* proposed in (Civera et al., 2008). This landmark parametrization is composed of six variables for the three-dimensional space and permits an accurate representation of uncertainty.

Finally, it is important to point out that numerous algorithms, such as filters, model the state vector as a group of stochastic variables assigning a probability distribution to each one. Usually, stochastic variables are approximated with Gaussian distributions and the content of the state vector are filled with their mean. In ad-

dition, these mean values are complemented with the covariance matrices describing their uncertainties.

2.2.3 State vector initialization

The state vector initialization aims to perform an initial estimation of the state variables, enabling the vector update process to have some initial values to work with. This first initialization usually receives an special treatment. The quality of many *SLAM* algorithms directly depends on the accuracy of this first initialization step, so special attention must be paid. This initialization process could be triggered in different situations such as the beginning of the algorithm or after a fail situation where the algorithm must be reset.

The state variables related to the camera location x_t^c are normally considered zero if no a priori position information is known. On the contrary, the state variables are assigned the mean value of the a priori probability distribution and the uncertainty is considered if the algorithm models them as stochastic variables.

Similarly, in the SLAM problem, the state vector corresponding with the landmarks x_t^l is usually unknown. However, in this case the initial value obviously can not be considered zero, and an initial estimation of the landmark's position has to be performed. If a stereo camera is used, an option is to use a stereo algorithm to initially estimate the depth of the landmarks. If required, the uncertainty in the landmark's position can be calculated using the precision achieved during the detection process and the uncertainty of the camera's intrinsic parameters estimation during the calibration process (Davison and Murray, 2002).

If the system has a monocular configuration, an stereo analysis is not possible and other algorithms must be used. In (Davison, 2003), the uncertainty for each landmark for the first frame is distributed over a straight line in the three-dimensional space. Some hypotheses are distributed along this line using a particle filter approach and validated afterwards during the next frames. In (Nister, 2004a) the authors proposed an algorithm based on a five points RANSAC (Fischler and Bolles, 1981) hypotheses. An evolution of this five point initialization algorithm was proposed in (StewÃľnius et al., 2006).

2.2.4 Feature detection and description

Feature detection algorithms aim to find points of interest in the scene, e.g. corners, such that they have a high repeatability and they are robust to scale or rotation transformations. The feature detection step is needed in a wide range of *SLAM* algorithms and it is mainly used for landmark detection and tracking.

The feature descriptors use the image information around the detected point to obtain a description vector. This vector varies from one algorithm to another and is used for searching landmarks that are similar. These algorithms are used mainly for feature tracking and matching between different frames. The most common algorithms used for these steps are:

- Harris (Harris and Stephens, 1988)
- Shi (Shi and Tomasi, 1994)
- Fast-10 (Rosten and Drummond, 2006)
- SIFT (Lowe, 1999)
- SURF (Bay et al., 2006)
- KLT with gain estimation (Pollefeys et al., 2008)

2.2.5 State vector update

The state vector update step is the heart of the *SLAM* algorithms. It is the responsible of, given a new frame and the previous state vector values, estimating the current state vector. Depending on the approach taken to solve this step, they can be classified an filter-based, bundle adjustment and visual odometry. Each method uses a different update algorithm, composed generally of different sub-steps. The work herein proposed tries to classify those steps based on the most important solutions found on the literature.

The next sections will be dedicated to explain in detail the classification proposed for each family.

2.2.5.1 Filter-based state vector update

These methods treat the state vector variables as a group of stochastic variables, assigning them a probability distribution. Generally, the probability distributions are considered Gaussian distributions, so each variable can be defined by its mean value and the covariance matrix. Note that the quadratic relation between the size of the state vector and the size of the covariance matrix will impact directly on the size of the problem.

The algorithms that populate this group are mainly based on the *Kalman Filter* (KF), its extension, the *Extended Kalman Filter* (EKF) or the *Paricle Filter*. After analysing these algorithms, the steps described in Figure 2.3 were found.

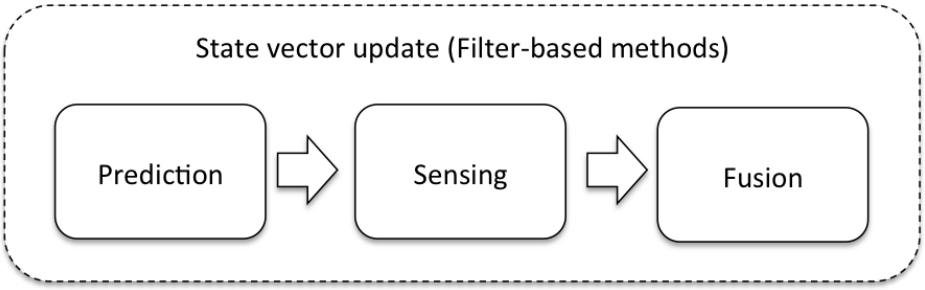

Figure 2.3: Filter-based state vector update steps

The first step for the vector update is the **prediction** step. The main task of this step is to estimate the probability distribution for the state vector variables based on how the robot's actions modifies its state. Here, while the action concept was inherited from the robotics community, it refers to the whole differential sensor's measurements. Traditionally, in robotics, the encoders are used for this task, making it possible to predict where the robot and the landmarks should be. Inertial sensor are also included, specially in cases where the encoders are not available or they just do not make sense, as in free moving camera situation. Mathematically, this concept is expressed in Equation 2.7.

$$P\left(x_t^s | x_{t-1}^s, a_{t-1}\right) \tag{2.7}$$

where x_t^s is the current state vector, x_{t-1}^s is the previous state vector and a_{t-1} is the *action* or *motion model* that describes the transition function.

Note that this is the general formulation of the prediction step. Depending on

the sensor configuration, the a_t function will have different formulations. In the case of general computer vision configuration, with a free moving camera, unless the camera has attached some inertial sensors, the prediction step is reduced to a constant rigid-body velocity model, which reduces the formulation to Equation 2.8.

$$P\left(x_t^s | x_{t-1}^s\right) \tag{2.8}$$

However, the vision system can be used, in addition to landmark detection, for performing visual odometry. These algorithms try to estimate the movement of the camera frame-to-frame by means of feature detection and tracking. As a black box, it behaves similarly to an inertial sensor and therefore, the visual odometry estimations can be incorporated in the a_t function. In this case, the Equation 2.7 is used. RANSAC methods are commonly used for this purpose (Agrawal and Konolige, 2006; Nister, 2003)

It is important to understand the differences between these approaches. While visual odometry uses feature detection and tracking in a deterministic frame-to-frame analysis, probabilistic approaches consider the whole previous state vector and its probability functions for estimating the whole current state.

The next step is the **sensing** step. It is responsible of acquiring absolute measurements with respect to the static environment. These measurements lead to a probability distribution of the measurements obtained given a certain state vector. This is expressed in Equation 2.9.

$$P\left(z_t | x_t^s\right) \tag{2.9}$$

where s_t are the measurements obtained.

There are two different types of measurements that can be obtained for this step: absolute position measurements using GPS or similar technologies or measurements relative to the environment.

The global absolute measurements obtain the sensor's three dimensional geospatial position (latitude, longitude and height values). The main advantage of the GPS technology is its simplicity, the global position information and its relatively good accuracy. However, it is only available for outdoor applications, it has problems in big cities due to satellite visibility issues and its accuracy is adequate for

big displacements but insufficient for small movements, specially for the elevation measurement. Given these problems, usually the GPS sensors are not considered suitable for $SLAM$.

Relative measurements can be obtained with a wide range of sensors, such as laser, sonar, etc. However, $VisualSLAM$ algorithms usually use a landmark detection and correspondence framework. The landmark detection is usually performed in the previous step explained in subsection 2.2.4. For the landmark correspondence, a variety of algorithms can be used:

- **Feature tracking**: based on the property that consecutive frames only imply small changes in the image, the features are searched in a reduced portion of the following frame, making the matching process more reliable and efficient. This method has the main limitation that only consecutive landmarks are matched and landmark re-visiting is generally a problem.
- **Patch matching**: each feature detected is associated with the surrounding intensity information and modelled as a planar patch. This patch can be used for matching new landmarks with already detected ones that are included in the environment map. Considering the perspective transformations, a *warping* can be applied to the patches for a more accurate matching.
- **Descriptors matching**: each feature is associated with a descriptor using some of the methods explained in subsection 2.2.4. When new landmarks are detected, their descriptors are compared with the ones already detected, performing the matching. Although these methods are accurate, they tend to be very computationally expensive.

Finally, the last step is the **fusion**. This step uses the probability distributions obtained previously and performs the estimation of the state vector for the current frame. Let's consider the visible variables, i.e. variables that can be measured, the environment measurements z_t and the action or inertial measurements a_t. The equation that relates the probability distribution of the current state vector, and the measures is shown in Equation 2.12.

$$Z_t = \{z_1, z_2, z_3 ... z_t\} \tag{2.10}$$

$$A_t = \{a_1, a_2, a_3 ... a_t\} \tag{2.11}$$

$$P\left(x_t^s | Z_t, A_{t-1}\right) = \frac{P\left(z_t | x_t^s\right) \sum_{\Phi \in x_{t-1}^s} P\left(x_t^s | x_{t-1}^s, a_{t-1}\right) P\left(x_{t-1}^s | Z_{t-1}, A_{t-1}\right)}{\sum_{\Phi \in x_t^s} P\left(Z_t, x_t^s\right) P\left(x_t^s | Z_{t-1}, A_{t-1}\right)} \tag{2.12}$$

where Z_t and A_t are the visible variables until time t, as described in Equation 2.10 and Equation 2.11. The sets $\Phi \in x_{t-1}^s$ and $\Phi \in x_t^s$ are all the possible states of the stochastic variables x_{t-1}^s and x_t^s respectively.

Although this equation might seem a bit scary at first glance, it is important to understand what each of its terms represent. Firstly, $P\left(x_t^s | Z_t, A_{t-1}\right)$ is the solution of the fusion step, representing the probability of the state vector in the current frame, given all the measurements until now. Note that the A_t variable is not yet known when the fusion process is run. $P\left(z_t | x_t^s\right($ describes how probable is the current state given the current world measurements, i.e. the **sensing** step term.

The next summation represents the probability distribution of the current state given all the previous measurements and actuations, i.e. the **prediction** step. It is composed of two terms; the $P\left(x_t^s | x_{t-1}^s, a_{t-1}\right)$ models the influence of the actuators and $P\left(x_{t-1}^s | Z_{t-1}, A_{t-1}\right)$ models the probability of the state vector in the previous frame. Note that this last term is exactly the solution term of the equation but for the previous frame.

Finally, although it is not obvious, the denominator is a normalization term. Therefore, sometimes Equation 2.12 is represented in a reduced form in Equation 2.13

$$P\left(x_t^s | Z_t, A_{t-1}\right) = \eta_t P\left(z_t | x_t^s\right) \sum_{\Phi \in x_{t-1}^s} P\left(x_t^s | x_{t-1}^s, a_{t-1}\right) P\left(x_{t-1}^s | Z_{t-1}, A_{t-1}\right) \tag{2.13}$$

where η_t is a normalization constant independent from the current state vector x_t^s.

Depending on the way of solving Equation 2.12 and the assumptions considered, several algorithms are used such as *Kalman Filter* (Kalman, 1960), *Extended Kalman Filter* (Maybeck and Siouris, 1980), *Iterated Extended Kalman Filter* (Maybeck and Siouris, 1980), *FastSLAM* (Montemerlo et al., 2002), *FastSLAM2.0* (Montemerlo

et al., 2003) and *Rao-Blackwellised* (Sim et al., 2006).

In (Yan et al., 2009) a review of the Extended Kalman Filter based algorithms recently proposed for navigation purposes is shown.

Algorithm	Sensor configuration	Estate vector		Map initialization	Detection / description	State vector update		
		Position	Landmarks			Prediction	Sensing	Fusion
(Davison, 2003)	Monocular	3D pos+vel	pos + patch	particles	Shi	Const. velocity	bounded search	EKF
(Sim et al., 2006)	Monocular / stereo	3D pos	pos + descriptor	undelayed initialization	SIFT	Any distribution	SIFT	Rao-blackwellised
(Molton et al., 2004)	Monocular	3D pos+vel	pos + patch + norm	Not detailed	Shi	Const. velocity	bounded search with warping	EKF
(Davison et al., 2007)	Monocular	3D pos+vel	pos + patch + norm	Given position	Shi	Const. velocity	bounded search with warping	EKF
(Eade and Drummond, 2006)	Monocular	3D pos+vel	pos + patch + norm	particles	FAST	Const. velocity	bounded search with warping	FastSLAM2.0
(Tomono, 2007)	Monocular	3D pos	3D pos	particles	-	Const. prob	exhaustive hypothesis	Rao-blackwellised
(Davison et al., 2004)	Monocular	3D pos+vel	pos + patch	particles	Shi	Const. velocity	bounded search	EKF
(Pollefeys et al., 2008)	Monocular	3D pos+vel	3D pos	5 points	KLT	Tracking + RANSAC	GPS + INS	EKF
(Davison and Murray, 2002)	Stereo	2D pos	pos + patch	stereo	Shi	Encoders	bounded search	EKF
(Paz et al., 2008)	Stereo	3D pos+vel	inverse depth + patch	stereo	Shi	Const. velocity	bounded search	EKF local maps
(Agrawal and Konolige, 2006)	Stereo	3D pos	3D pos + patch	stereo	Harris in disparity space	Tracking + RANSAC	GPS	KF
(Schleicher et al., 2006)	Stereo	3D pos+vel	pos + patch	stereo	Shi	Const. velocity	bounded search	EKF

Table 2.1: Filter based systems review. (- values represent *not specified*)

2.2.5.2 Bundle adjustment

Bundle adjustment addresses the problem of visual localization and reconstruction as an optimization problem where all the variables along a group of frames are jointly estimated. These methods are very flexible because they allow to include as variables not only the position and orientation of the camera, but also its internal parameters independently for each frame. In addition, different cost functions can be specified for being minimized. For these reasons sometimes bundle adjustment is also used for accurate camera calibration in the refinement step (Furukawa and Ponce, 2008). (Engels et al., 2006) realizes an analysis of the most important real-time bundle adjustment techniques and (Strasdat et al., 2010) confirms that bundle adjustment methods can be used for real-time applications if it is wisely formulated and implemented.

Bundle adjustment is just a large sparse geometric parameter estimation problem, where an optimal solution is found in terms of the cost function selected. Generally, the problem is expressed as a non linear least squares problem. A very simple example is shown in Equation 2.14. The cost function is formulated as the sum of the squared distances between the expected feature position in an image and its real measured position.

$$\min_{x^c,x^l} \sum_{i}^{I} \sum_{t}^{T} v_{it} d\left(H(x_t^c, x_i^l), m_{it}\right)^2 \tag{2.14}$$

where x^c and x^l are the camera-related parameters (including position and orientation) and landmarks parameters respectively, J and T are the number of frames and the number of landmarks, $H(x^c, x^l)$ is the projection function that transforms x^l into the two-dimensional projection under x^c parameters and m_{it} the detected feature i in the frame t. The function $d(a, b)$ calculates the distance between the a and b vectors and v_{it} is a boolean parameter that takes value one only if the landmark i is seen in frame t. Note that the vector x^c is composed of all the camera parameters for each t frame, but, assuming static environment, x^l is time-invariant and referred to, for instance, the starting position at frame $t = 0$.

The optimal solution for Equation 2.14 obtains the parameters that produce the minimum reprojection error in terms of geometric distance between features. However, bundle adjustment can be also used for intensity-based matching of image

patches. Instead of geometric coordinates, the observables are gray values or colors and the error is formulated in terms of intensity residuals. More detailed information for intensity-based bundle adjustment can be found in (Triggs et al., 2000)

In the literature, a more general expression for the projection error is usually found. It is shown in Equation 2.15.

$$\Delta z_k(x) \equiv \underline{z_k} - z_k(x) \tag{2.15}$$

where x is the vector containing all landmarks and camera parameters at any time, $z_k(x)$ the projection function of the landmarks in x for camera parameters at frame k and $\underline{z_k}$ the landmark positions measured in frame k.

This projection error leads to a more general expression of the cost function, Equation 2.16.

$$f(x) = \frac{1}{2}\sum_k \Delta z_k(x)^T W_k \Delta z_k(x) \tag{2.16}$$

where $\Delta z_k(x)$ is the prediction error and W_k is the symmetric positive definite *weight matrix*, which, for Gaussian measurement models, it can be demonstrated that should be chosen to approximate the inverse measurement covariance of $\underline{z_k}$.

The cost function in Equation 2.16 is usually called the *weighted sum of squared errors*, where the weight matrix allows to center the optimization cost in the variables with less variance, while letting the uncertain variables to acquire a wider range of values.

Even with the weighted expression of the cost function, the appearance of outliers might ruin the parameter estimation. These outliers are frequent and are generally caused by a feature mismatch. Therefore, for robustness purposes it is necessary to include a cost function which is able to deal with outliers. The problem with outliers is the high value of the cost function for those features incorrectly matched, so a saturating function is used. Equation 2.17 adds the robust distance error function to the Equation 2.16.

$$f(x) = \frac{1}{2}\sum_k \varphi\left(\Delta z_k(x)^T W_k \Delta z_k(x)\right) \tag{2.17}$$

where φ is a robust function, acquiring a constant value for infinite error values.

Similarly to the filter-based case in subsubsection 2.2.5.1, the bundle adjustment update step can be subdivided into three different substeps. These substeps are shown in Figure 2.4.

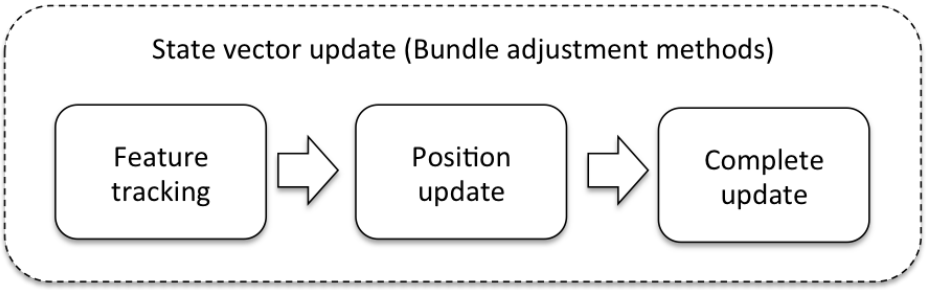

Figure 2.4: Bundle adjustment state vector update steps

The **feature tracking** step is used for establishing feature matches between frames. This step is essential because the bundle adjustment requires knowing which is the projection of each feature in every frame. Although robust cost functions help to overcome mismatching problems, a good quality feature matching can prevent future problems during the bundle adjustment. There are many different ways to deal with the feature tracking/matching problem. In the literature, one of the most popular algorithms used for bundle adjustment is the feature searching in a restricted image area calculated assuming a constant velocity model. The feature is usually modeled as a planar patch and the correlation is calculated after a patch warping for enhanced matching robustness. Many bundle adjustment algorithms are centered on the next two steps, but it is important to emphasize the crucial role of this step.

The next step is the **position update** step. It is responsible for updating just the camera-related parameters, i.e. its intrinsic and extrinsic parameters, using the same bundle adjustment methods. This estimation is achieved assuming that the feature-related parameters are constant since the last complete update, reducing considerably the size of the parameter vector. Therefore, this position update is considerably faster than the complete updating step, and generally achieves a good approximation of the camera parameters. This step is not mandatory, but it is a good approach for fast camera updates.

Finally, the **complete updating** step is run. This step is the core of the bundle adjustment technique and aims to solve the minimization problem described in Equation 2.14. During this process, all the parameters associated with both camera

and landmarks are updated. Due to being a larger problem than the position update step, it is more computationally demanding.

An efficient strategy found in the literature is to let the position update step run in a high frequency thread for obtaining a high rate position estimation, and also run the complete update step in a different thread, with a lower frame-rate depending on video *keyframes*. This allows the landmarks to be updated when enough camera movement has occurred, leading to a more accurate estimate, without reducing the frame-rate for localization. Thus, although Figure 2.4 suggests that the state vector update is a process composed of three consecutive tasks, it can be implemented in many different ways, removing the position update, running the complete update just some frames, running them in parallel, etc.

Algorithm	**Sensor configuration**	**Estate vector**		**Map initialization**	**Detection and description**	**State vector update**		
		Position	**Landmark**			**Tracking**	**Position update**	**Complete update**
(Engels et al., 2006)	Monocular	3D pos	3D pos	5 points alg 2004	-	-	-	single thread
(Newcombe and Davison, 2010)	Monocular	Transf. Matrix	pos + patch + norm	5 points alg 2006	Fast-10	bounded search with warping	high rate thread	keyframes
(McLauchlan and Murray, 1995)	Monocular	3D pos	3D pos	batch VSDF	-	-	VSDF	VSDF
(Manessis et al., 2000)	Monocular	3D pos	3D pos	batch VSDF	-	-	VSDF	VSDF + visibility constraints
(Klein and Murray, 2007)	Monocular	Transf. Matrix	pos + patch + norm	5 points alg 2006	Fast-10	bounded search with warping	high rate thread	keyframes

Table 2.2: Bundle adjustment based systems review. (- values represent *not specified*)

2.2.5.3 Visual odometry

Visual odometry (*VO*) refers to the group of techniques that addresses the problem of finding the egomotion of the camera only considering the information of two frames. The concept is similar to the classic odometry measures realized by encoders or inertial sensors, which permits estimating the camera movement between two frames. Nonetheless, using the computer vision approach is considered to improve the relative position error in comparison to the sensor-based methods. These techniques started to appear during the 1980's decade with the system described in (Moravec, 1980), but the concept of *Visual odometry* was not introduced until (Nister et al., 2004). Some authors consider these techniques a particular case of the well studied *SFM* problem. However, *VO* tackles the problem as a frame by frame estimation, retrieving the global camera positions in an incremental basis. An in-depth *Visual odometry* survey can be found in (Scaramuzza and Fraundorfer, 2011).

The problem can be formulated as follows: the camera movement between two frames can be represented as a rigid-body transformation $T_{t,t-1}$. This transformation matrix has the form shown in Equation 2.18.

$$T_{t,t-1} = \begin{bmatrix} R_{t,t-1} & t_{t,t-1} \\ 0 & 1 \end{bmatrix} \tag{2.18}$$

where $T_{t,t-1} \in \mathbb{R}^{4x4}$, $R_{t,t-1}$ is the rotation matrix, $t_{t,t-1}$ the translation vector and t the frame of interest. The purpose of the *VO* algorithms is to estimate those transformation matrices. Thus, retrieving the camera position at frame t can be achieved by concatenating the successive translation matrices from frame zero to t.

$$C_t = \prod_{i=0}^{t} T_{i,i-1} \tag{2.19}$$

As Equation 2.19 indicate, *VO* recovers the camera's trajectory incrementally pose after pose. Finally, for reducing the drift and enhancing the reconstruction quality, a final iterative refinement process over the last frames can be performed. This final step is called *windowed-bundle adjustment* and, similarly to the *bundle adjustment* techniques mentioned in subsubsection 2.2.5.2, it tries to minimize the squared reprojection error.

As shown in Figure 2.5, *visual odometry* algorithms are composed of two different steps: the feature tracking or matching and the motion estimation. In addition, note that the final *windowed-bundle adjustment* step can also be applied if an enhance accuracy is wanted.

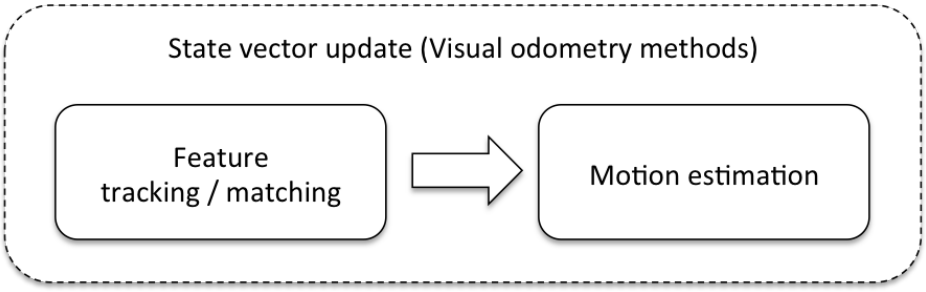

Figure 2.5: Visual odometry state vector update steps

The **feature tracking or matching** step is very similar to the first step described for the *bundle adjustment* methods. While the tracking approach is more suited for applications where the frames are taken consecutively and with small movements between them, the matching approach is efficient when large motions or non-consecutive frames are expected. A complete review of the detection and description algorithms for *VO* can be found in (Fraundorfer and Scaramuzza, 2012).

The **motion estimation** step is the core of every *VO* system. This step is where the transformation matrices from Equation 2.18 are computed. Both monocular and stereo configurations can be used for *VO* algorithms. However, the algorithms for each configuration are quite different. For stereo configurations, the procedure usually exploits the stereo depth reconstruction capability, performing a triangulation step for obtaining the depth of the features. Afterwards, the motion is estimated as a three-dimensional registration problem. If a monocular configuration is used, only bearing information is available and other techniques have to be used. Note that the monocular configuration is also a degenerate case of the stereo one, when the baseline is small compared with the environment distances.

More specifically, the following classification divides the *VO* algorithms depending on how features are specified:

- **3D to 3D**: features in the previous and current frames are considered three-dimensional. They usually corresponds to features detected and triangulated in a stereo system. Conceptually, the 3D point-clouds have to be registered.

Some examples are (Arun et al., 1987; Maimone et al., 2007) and (Comport et al., 2007) which, instead of using triangulation, is based on the quadrifocal constraint.

- **2D to 2D**: in this case, features of both previous and current frames are just detected in the image space. This problem arises during the initialization phase for monocular systems. The five-point algorithm presented in (Nister, 2004a) and the eight-point algorithm (Longuet-Higgins, 1987) are some examples of 2D to 2D algorithms.

- **3D to 2D**: features on the previous frame are specified in the three-dimensional space, but features on the current frame are just detected in the image space. Note that this can be used for both stereo and monocular systems, but for the latter, three frames are needed. Some examples are described in (Nister et al., 2004; Hartley and Zisserman, 2004; Kneip et al., 2011; Nister, 2004b).

A discussion about the accuracy of these three different approaches can be found in the literature. According to (Nister et al., 2004), the image-based methods (2D to 2D and 3D to 2D) generally achieve a better performance than those based only in the three-dimensional information (3D to 3D). This is due to the high uncertainty in the triangulated 3D points have in the depth dimension. Similarly, for monocular systems, 2D to 2D methods are preferable to the 3D to 2D ones.

Independently from the 3D or 2D approach, the feature matching step is suitable of incurring in outliers in the form of mismatches. These outliers can spoil all the motion estimation step, so a robust algorithm must be used. For this task, RANSAC (Fischler and Bolles, 1981) based techniques have become the most popular, being able to deal with a quite big outliers ratio with real-time performance.

Algorithm	**Sensor configuration**	**Estate vector**		**Map initialization**	**Detection and description**	**State vector update**	
		Position	**Landmarks**			**Feature tracking / matching**	**Motion estimation**
(Nister et al., 2004)	Monocular / Stereo	3D pos	3D pos	5 point algorithm / stereo	Harris + non-max suppression	KLT	3D to 2D
(Hirschmüller et al., 2002)	Stereo	3D pos	3D pos	stereo	Harris	Rank / Census	least-squares 3D to 3D
(Howard, 2008)	Stereo	3D pos	3D pos	stereo	Harris / FAST	SAD	least-squares 3D to 3D
(Maimone et al., 2007)	Stereo	3D pos	3D pos	stereo	Harris + non-max suppression	odometry + NCC	Maximum likelihood 3D to 3D
(Comport et al., 2007)	Stereo	3D pos	-	stereo	-	-	Quadrifocal 3D to 3D
(Nister, 2004a)	Monocular	3D pos	2D	-	-	-	2D to 2D
(Kneip et al., 2011)	Monocular / Stereo	3D pos	3D	-	-	-	3D to 2D

Table 2.3: Visual odometry based systems review. (- values represent *not specified*)

3

Multiresolution energy minimization framework

Nowadays, retrieving the three-dimensional structure of a given scene is being widely studied due to its potential applications. Some of these applications are oriented to computer graphics such as view synthesis (Rogmans et al., 2009), monument reconstruction and rendering, urban reconstruction (Pollefeys et al., 2008) (Cornelis et al., 2008) and so on. Other applications are related to sense the environment to feed intelligent systems such as robot navigation and SLAM. Passive stereo vision techniques have suffered a big improvement during the last decade due to the increase of the computational capacity and the appearance of new algorithms.

Dense disparity map estimation is the area of stereo vision which probably has been more intensively studied. Note that all the stereo applications aforementioned are possible thanks to the dense disparity estimation (Scharstein and Szeliski, 2002). Although sparse feature reconstruction algorithms have received the main research focus in the past, especially for real-time applications, dense disparity maps generally provide much more information about the scene structure than sparse features. The main challenge is to obtain accurate dense depth maps while being able to run the algorithms in real-time.

Disparity map estimation is commonly the first step for robot navigation when a vision system is installed in the robot or vehicle. Stereo vision for robot navigation is considered more reliable and efficient than monocular structure-from-motion estimation because of being a simpler problem. Knowing the environment in a certain moment is the first step for being able to estimate the movement without using odometry. As shown in Figure 1.1, it was considered during the Introduction chapter as the first step or pose retrieval. This three-dimensional information will be refined

and used later for visual odometry.

For real-time-oriented applications, several solutions have been proposed in the literature. Traditionally, the most successful ones are those based on local methods, which are generally very computationally efficient. Although a lot of variations have been proposed, these methods have low accuracy as their main drawback. As a modification of the local methods, and in order to reduce their computational cost, several authors proposed to apply multiresolution methods (Zhao and Taubin, 2011) (Yang and Pollefeys, 2003a) (Gong and Yang, 2001). These multiresolution algorithms use information from coarse resolutions to guide the search in the fine ones. Energy minimization algorithms, also known as global algorithms, solved using graph-cuts (Boykov et al., 2001) or belief propagation (Yedidia et al., 2003) have been traditionally excluded for real-time applications (Scharstein and Szeliski, 2002).

This chapter presents a new multiresolution energy minimization framework for real-time robotic applications. It extends the classic energy minimization framework and aims to reduce the optimization problem's size for making it more computationally efficient. The main contributions of this chapter are decoupling the disparity map resolution from the stereo input resolution with no disparity decimation penalization and the reduction of the problem's complexity associated.

The rest of the chapter is organized as follows: section 3.1 reviews the main two-frame stereo algorithms found in the literature, making special emphasis on the multiresolution approaches; section 3.2 is a brief introduction to Markov Random Fields (MRF); section 3.3 shows the classic energy minimization framework while section 3.4 presents the new multiresolution minimization framework; in section 3.5 the impact of the reduction parameter R in the performance is analyzed theoretically; in section 3.6 the impact of R on the MRF parameters is studied; in section 3.7 some experiments are carried out and some results are shown and finally in section 3.8 some conclusions are drawn.

3.1 Dense two-frame stereo algorithms overview

Dense stereo is one of the most researched topics in recent computer vision. Many overviews and surveys that analyze and compare most popular algorithms have been published. One of the most important surveys is the taxonomy and evaluation made in (Scharstein and Szeliski, 2002), where a thorough state of the art of dense two-

frame algorithms and their comparison is presented. According to this taxonomy, dense two-frame stereo algorithms can be classified into two groups: local methods and global optimization methods.

3.1.1 Local and global stereo

Local methods estimate the depth for each pixel independently, only using the color information surrounding them. Typically, they are simple algorithms that perform some kind of correlation based on photometric properties over a support window. They compute a cost for each available disparity and select for each pixel independently the disparity configuration that achieves a lower cost

$$\min_{\overline{d}} C_{all}(\overline{d}) = \min_{\overline{d}} \sum_{p} C\left(p, d_p\right) = \sum_{p} \min_{d_p} C\left(p, d_p\right) \tag{3.1}$$

being $C_{all}(\overline{d})$ the global cost associated to all pixels, $C(p,d)$ the cost associated to pixel p for disparity value d_p and $\overline{d} = \{d_1, d_2, ..., d_K\}$ the vector of all disparities for all K pixels.

The great advantage of these algorithms is that they usually can be computed in parallel and are much faster than the global optimization ones (Rogmans et al., 2009) (Scharstein and Szeliski, 2002) because their costs are independent of other pixel's disparities. In fact, nowadays, local methods continue to be the most popular for real-time applications. The most important algorithms that are included in this group are square window, shiftable window, boundary guided and adaptive weight (Yoon and Kweon, 2006). All of them are reviewed and compared in (Wang et al., 2006a). The comparison is made in terms of accuracy and performance.

Global optimization methods estimate the depth of each pixel of the reference image considering the depth and the cost value of other pixels in the image. Global algorithms can be classified in three different groups: Markov Random Fields (MRF) based, dynamic programming and cooperative algorithms. The MRF based methods (Geman and Geman, 1984) are solved using an energy minimization framework: an energy function that depends on the depth estimation is defined and the minimum energy configuration is searched.

$$\min_{\overline{d}} E_{all}(\overline{d}) = \min_{\overline{d}} \sum_{p} E\left(p, \overline{d}\right) \tag{3.2}$$

where $E_{all}(\overline{d})$ is the total energy and $E\left(p, \overline{d}\right)$ is the energy for pixel p and disparity configuration $\overline{d}$.

The purpose of these algorithms is to obtain a disparity configuration that minimizes the aforementioned energy function. This approach is very popular because it can be justified in terms of maximum a posteriori estimation of a MRF. During the last decade, many algorithms have been proposed to solve this NP-hard optimization problem. Simulated annealing was the first algorithm used for solving them but it was demonstrated to be too computationally demanding and did not perform well in terms of accuracy. Lately, other algorithms have been presented such as Iterated Conditional Modes (ICM) (Besag, 1986), Expansion, Swap (Boykov et al., 2001) and belief propagation (Yedidia et al., 2003). A comparison of their accuracy and performance is studied in (Szeliski et al., 2008) (Tappen and Freeman, 2003). Recently, parallel versions of the graph-cuts and belief propagation have been successfully implemented in GPGPUs (Vineet and Narayanan, 2008) (Yang et al., 2006) (Liang et al., 2009). The main drawback of these algorithms is that generally powerful GPGPUs are not available for real-time applications such as robotic navigation. Modifications to the graph-cuts algorithms for performance enhancement have been lately proposed in (Kohli and Torr, 2007) (Alahari et al., 2010) (Juan and Boykov, 2006) (Wang et al., 2008) (Yu et al., 2007). These techniques are complementary to the framework proposed in this paper, so they can be applied as the optimization algorithm.

The dynamic programming (Bobick and Intille, 1999) (Salmen et al., 2009) (Veksler, 2005) (Wang et al., 2006b) approach solves optimally each scanline independently. For this reason, no coherence is enforced between different scanlines

$$\min_{\overline{d}} E_{all}(\overline{d}) = \min_{\overline{d}} \sum_{p} E\left(p, \overline{d}_l\right) = \sum_{l \in L} \min_{\overline{d}_l} \sum_{p \in l} E\left(p, \overline{d}_l\right) \tag{3.3}$$

where $E_{all}(\overline{d})$ is the total energy, $E\left(p, \overline{d}_l\right)$ is the energy of pixel p given the disparity values of line l, L the set of lines in the reference image and $\overline{d} = \{\overline{d}_1, \overline{d}_2, .., \overline{d}_L\}$ is the vector containing the disparities for each line l in L as each of its elements.

The size of the problem is reduced considerably and near real-time performance

can be achieved. However, the accuracy of the solutions is similar to the ones obtained with local methods (Scharstein and Szeliski, 2002).

Cooperative algorithms (Marr and Poggio, 1976) (Wang and Zheng, 2008) use local non-linear operations that finally result in global optimization behavior. They achieve good accuracy but with high run-time.

Global methods generally obtain better results than local methods especially in the discontinuity areas; on the contrary, global methods require more computational resources than local methods and are not suited for real time applications.

In Middlebury's website (http://vision.middlebury.edu/stereo/) a complete accuracy analysis for many state of the art algorithms is presented. The methods that achieve the best results are mainly based on global algorithms solved using belief propagation (Klaus et al., 2006) (Yang et al., 2009) (Zitnick et al., 2007). Apart from the basic algorithms mentioned earlier, every method presented in the evaluation includes other processing algorithms that improve the disparity accuracy. Usually, some kind of preprocessing algorithm is applied, such as image segmentation, which helps to obtain better disparity estimations.

3.1.2 Multiresolution approaches

Using multiresolution techniques in the two-frame stereo correspondence problem is not novel. The early work in (Zemerly et al., 1992) proposes a coarse-to-fine pyramidal approach applied to airborne imagery. Different resolution levels are created where the analysis in the coarser layers is used to constraint the search in the finer ones. Owing to the reduction of the search space and the parallelization capabilities, they report an enhancement of both accuracy and performance. In (Iocchi and Konolige, 1998) the authors propose to use coarser resolutions for objects that are near the camera and only apply fine resolutions for objects that are far away. This approach has as main drawback that for robotic applications generally is preferable to obtain high disparity resolution for near objects (potential obstacles) than for the rest of the objects in the environment. In (Satorre et al., 2002) a similar approach to (Zemerly et al., 1992) is presented but information about edges is incorporated. Recently, a couple of interesting multiresolution algorithms have been successfully implemented in GPGPUs. In (Yang and Pollefeys, 2003b) some implementation based modifications to the pyramidal approach are made in order to achieve real-time performance on commodity graphics hardware. In (Zhao and

Taubin, 2011) a pyramidal algorithm is also implemented in a GPGPU but using adaptive windows local search and foreground detection.

Every multiresolution algorithm presented in the literature suffers the same problem: coarser pyramid levels leads also to coarser resolutions in the disparity dimension which can lead to bad correspondences that propagate to finer levels. So, if high resolution in the disparity dimension is needed, the whole fine layers must be analyzed. They also have the main drawback that only local methods are used, which lead to less accurate results as stated in (Scharstein and Szeliski, 2002). The only multiresolution algorithm applied to a global optimization algorithm can be found in (Chang and Chatterjee, 1990), but Simulated Annealing is used, which is an outdated approach compared to grap-cuts or belief propagation.

In this chapter a completely different multiresolution approach is presented. While other methods construct a pyramid from coarse to fine resolutions, our framework uses just two different resolutions: one for the input stereo images and other one for the dense disparity depth map. The main contribution of this chapter is developing a framework for reducing the final depth map resolution (width and height) while preserving the disparity resolution (number of disparity labels). The disparity pixels for the low resolution depth map are modelled as a Markov Random Field. This image decimation reduces the number of stochastic variables considered in the MRF, also reducing the problem's complexity. In contrast, the disparity range and the data term are preserved from the high resolution images in order to achieve the same disparity resolution as the high resolution input. The reduction of the depth map's resolution provides a great improvement in performance due to the great reduction in the complexity of the problem.

3.2 Markov Random Fields introduction

A Markov Random Field, also called Markov Network or undirected graph is a graphical representation of the conditional independences that appear between a group of stochastic variables. The variables that appear in the graph should fulfill the Markov property. The main differences between the Markov Random Fields and the Bayesian networks is that the latter are directed graphs with no cycles while MRFs are undirected graphs and might contain cycles.

The Markovian property was first related to the memory-less processes. If a group

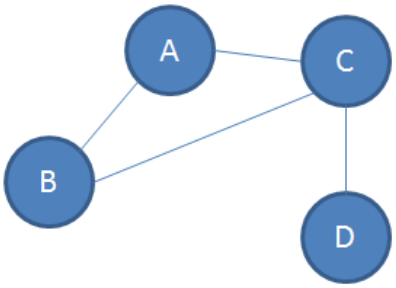

Figure 3.1: Markov Random Field example network

of stochastic variables describe the state of a system in a certain time t, the process is considered to have the Markov property if the state in time $t+1$ only depends on the current state at time t

$$\mathbb{P}\left(X_{t+1} = x_{t+1} | X_t = x_t ... X_0 = x_0\right) = \mathbb{P}\left(X_{t+1} = x_{t+1} | X_t = x_t\right) \tag{3.4}$$

where X_t is the state of the system at time t.

The Markov assumption applied to a MRF represents the conditional independence of all the variables in the graph. If the MRF network is known, a stochastic variable A is conditionally independent to any other stochastic variable in the graph given the subset of variables which are connected to A with an edge in the graph. For instance, for the graph shown in Figure 3.1 the following simplifications can be derived

$$\mathbb{P}\left(X_D = x_D | X_A = x_A, X_B = x_B, X_C = x_C\right) = \mathbb{P}\left(X_D = x_D | X_C = x_C\right) \tag{3.5}$$

$$\mathbb{P}\left(X_A = x_A | X_B = x_B, X_C = x_C, X_D = x_D\right) = \mathbb{P}\left(X_A = x_A | X_B = x_B, X_C = x_C\right) \tag{3.6}$$

A group of potential functions, also called cliques, can be defined over a MRF network

$$\Psi = \{\Phi_1(D_1), \Phi_2(D_2) ... \Phi_K(D_K)\} \tag{3.7}$$

where $\Phi_i(D_i)$ are the potential functions and D_i is the scope of each function, i.e the set of variables that interact in that particular clique. Not every set of cliques is compatible with any MRF network. Cliques containing in their scope variables that are conditional independent in the MRF are not allowed. The joint probability of any state of the variables can be calculated as follows

$$\tilde{\mathbb{P}}_\Phi \left(X_1 = x_1 ... X_N = x_n\right) = \prod_i^K \Phi_i(D_i) \tag{3.8}$$

$$Z_\Phi = \sum_{X_1 ... X_N} \tilde{\mathbb{P}}_\Phi \left(X_1 = x_1 ... X_N = x_n\right) \tag{3.9}$$

$$\mathbb{P}_\Phi \left(X_1 = x_1 ... X_N = x_n\right) = \frac{\tilde{\mathbb{P}}_\Phi \left(X_1 = x_1 ... X_N = x_n\right)}{Z_\Phi} \tag{3.10}$$

where N is the number of stochastic variables, $\tilde{\mathbb{P}}_\Phi$ is the unnormalized probability function, Z_Φ (also called the partition function), sums over all possible states of $\tilde{\mathbb{P}}_\Phi$, and finally $\mathbb{P}_\Phi$ is the joint probability function.

The joint probability can also be represented as a log-linear model by performing the following substitution

$$\Phi_i(D_i) = exp(-w_i f_i(D_i)) \tag{3.11}$$

$$\mathbb{P}_\Phi = \frac{\prod_i^K exp(-w_i f_i(D_i))}{\sum_{X_1 ... X_N} \prod_i^K exp(-w_i f_i(D_i))} = \frac{exp(\sum_i^K -w_i f_i(D_i))}{\sum_{X_1 ... X_N} exp(\sum_i^K -w_i f_i(D_i))} \tag{3.12}$$

where w_i is a vector of weights representing clique i and f_i is the feature function related to the clique i. Feature functions are defined such that they are indicators of the cliques configurations. $\mathbb{P}_\Phi$ is also known as the Gibbs distribution.

Considering previous definitions, the energy function $E(X)$ can be defined as

$$\mathbb{P}_\Phi = \frac{exp(-E(\bar{X}))}{\sum_{X_1 ... X_N} exp(-E(\bar{X}))} \tag{3.13}$$

The energy $E(\bar{X})$ represents the energy configuration of a given state $\bar{X}$ of the stochastic variables. From Equation 3.13 it can be deduced that the problem of finding the state of minimum energy is equivalent to finding the state of maximum probability for the distribution $\mathbb{P}_{\Phi}$.

3.3 Classical energy minimization approach

As described in (Geman and Geman, 1984), the stereo matching problem can be stated as the minimization of an energy function that includes two different terms. The aim of the global optimization algorithms is to find the set of labels $\overline{f} = \left\{f_1, f_2, ..., f_n\right\}$ where n is the number of pixels in the reference image that minimizes the energy function $E(\overline{f})$. In case of stereo correspondence, each label f_n represents the disparity value assigned to pixel n and can take any value of the disparity space $\Lambda = \{1, ..., L\}$, being L the value of the maximum analyzed disparity. Thus, obtaining an estimation of $\overline{f}$ is a solution for the correspondence problem. The equations for the traditionally formulated energy minimization framework are

$$\min_{\overline{f}} E(\overline{f}) \tag{3.14}$$

$$E(\overline{f}) = E_{data}(\overline{f}) + E_{smooth}(\overline{f}) \tag{3.15}$$

where $E_{data}(\overline{f})$ is the data term and $E_{smooth}(\overline{f})$ is the smoothing term. In the following subsections each term will be explained in greater detail.

3.3.1 Data term

$E_{data}(\overline{f})$ indicates the quality of the stereo correspondence between pixels of the solution $\overline{f}$ based on some photometric dissimilarity measure. This term can be generalized as

$$E_{data}(\overline{f}) = \sum_{p \in H} D_p(f_p) \tag{3.16}$$

where H is the set of all pixels in the reference image, f_p is the element p in the f_p vector and $D_p(f_p)$ is a dissimilarity function that describes how good the disparity

value f_p fits for a given stereo pair. This dissimilarity function might appear in different flavors. Some functions consider sub-pixel information, others take into account different illumination biases for left and right images, etc. However, the most simple dissimilarity function can be formulated as

$$D_p(f_p) = \sum_{c \in C} |I_L(x(p), y(p), c) - I_R(x(p) - f_p, y(p), c)| \tag{3.17}$$

where I_L and I_R are the left and right stereo pair respectively, C is the set of channels that form the images and $x(p)$ and $y(p)$ are the x and y coordinates of the pixel p respectively. Given this dissimilarity measure, it is important to note that it is very sensitive to bias difference between the stereo pair. This bias is very common because usually each stereo image is taken by a different camera with different lenses and different sensor and even different illumination conditions. However, this color and shape issues can be largely overcome through a calibration process. If previous calibration between the stereo pair is not possible, other dissimilarity measures have been proposed in the literature. For example, (Klaus et al., 2006) use a combination of both gradient and absolute color difference

$$C_{SAD}(p, f_p) = \sum_{u \in N(p)} I_L(x(u), y(u)) - I_R(x(u) - f_p, y(u)) \tag{3.18}$$

$$\begin{aligned} C_{GRAD}(p, f_p) = & \sum_{u \in N_x(p)} |\nabla_x I_L(x(u), y(u)) - \nabla_x I_R(x(u) - f_p, y(u))| \\ & + \sum_{u \in N_y(p)} |\nabla_y I_L(x(u), y(u)) - \nabla_y I_R(x(u) - f_p, y(u))| \end{aligned} \tag{3.19}$$

$$D_p(f_p) = (1 - \omega) \cdot C_{SAD}(p, f_p) + \omega \cdot C_{GRAD}(p, f_p) \tag{3.20}$$

where $N(p)$ is a 3x3 surrounding window at position $(x(p), y(p))$, N_x a surrounding window without the rightmost column, N_y a surrounding window without the lowest row, ∇_x the forward gradient to the right and ∇_y the forward gradient to the bottom. If color images are used, the dissimilarity measure must be summed up for all different channels. Finally, ω is a parameter for weighting the absolute difference term and the gradient term.

Another important contribution was published in (Birchfield and Tomasi, 1998), where a color similarity measure that is insensible to image sampling is proposed.

$$\overline{d}\left(p, f_p, I_L, I_R\right) = \min_{x(p)-f_p-\frac{1}{2} \leq k \leq x(p)-f_p+\frac{1}{2}} \left| I_L\left(x(p), y(p)\right) - \hat{I}_R(k, y(p)) \right| \tag{3.21}$$

$$\overline{d}\left(p, f_p, I_R, I_L\right) = \min_{x(p)-\frac{1}{2} \leq k \leq x(p)+\frac{1}{2}} \left| \hat{I}_L\left(k, y(p)\right) - I_R(x(p)-f_p, y(p)) \right| \tag{3.22}$$

$$D_p\left(f_p\right) = \min\{\overline{d}\left(p, f_p, I_L, I_R\right), \overline{d}\left(p, f_p, I_R, I_L\right)\} \tag{3.23}$$

where $\hat{I}_L$ and $\hat{I}_R$ are the linearly interpolated functions between the sample points of the left and right scanline respectively.

3.3.2 Smoothing term

The smoothing term E_{smooth} in the minimization problem described in the equation Equation 3.15, determines how smoothly the labels change throughout the disparity image. Generally, it is defined as the summation of the Markov Random Fields pairwise terms considering the neighboring pixels

$$E_{smooth}\left(\overline{f}\right) = \sum_{\{p,q\} \in N} V_{\{p,q\}}\left(f_p, f_q\right) \tag{3.24}$$

where $V_{\{p,q\}}$ is the aforementioned pairwise term and N the set of neighboring pixels defined by

$$p, q \in N \Leftrightarrow |x(p) - x(q)| + |y(p) - y(q)| = 1 \forall p, q \in H \tag{3.25}$$

Each pairwise term measures how probable is for pixel p to have assigned the label f_p, given that its neighboring pixel q has been assigned the label f_q.

As for the dissimilarity function $D_p(f_p)$, many different functions have been proposed for $V_{\{p,q\}}$. As stated in (Kolmogorov and Zabin, 2004), depending on

which optimization algorithm is going to be applied, not any pairwise function can be used. Graph-cuts has demonstrated to be one of the most efficient and accurate minimization algorithms when applied to the Markov Random Field stereo labeling problem. However, as demonstrated in (Kolmogorov and Zabin, 2004), only pairwise functions that satisfy the *regularity* condition are graph-representable, so only *regular* pairwise functions can be used as part of the smoothing term. The most popular ones used in the literature are the absolute difference

$$V_{\{p,q\}}(f_p, f_q) = \lambda \cdot |f_p - f_q| \tag{3.26}$$

and the truncated absolute difference

$$V_{\{p,q\}}(f_p, f_q) = \lambda \cdot \min(K, |f_p - f_q|) \tag{3.27}$$

where K is a threshold for difference of disparities and λ is a weighting parameter for setting the ratio between the smoothing and data term.

3.3.3 Reducing the stereo input resolution

The original problem can also be represented as a probabilistic graphical model. Each stochastic variable is represented as a node (a circle) in the graph and edges between nodes determine the conditional independence between variables. For instance, in Figure 3.2 each circle represents a variable p_i, one for each pixel in the reference image. $C(p, d)$ is the cost volume that determines the cost of assigning a certain label to a given pixel, which is aggregated in the $E_{data}(\overline{f})$ term. The edges between variables are related to the $E_{smooth}(\overline{f})$ term. The graphical representation of the classic energy minimization problem is shown in Figure 3.2.

An option for improving the performance of the algorithm is to reduce the size of the stereo input by a factor of R. This operation diminishes the total number of stochastic variables by a factor of R^2 but also diminishes the resolution in the disparity space, i.e the total number of disparities between the algorithm may choose. The graphical representation of this situation is shown in Figure 3.3. An analysis of the performance improvement incurred by this operation is detailed in section 3.5.

Depending on the configuration of the stereo cameras, the total resolution of the

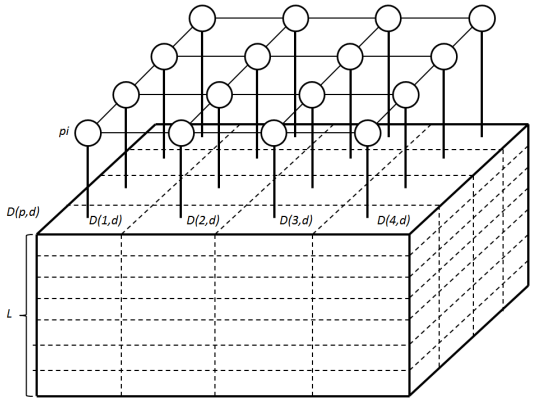

Figure 3.2: Markov Random Field graphical representation

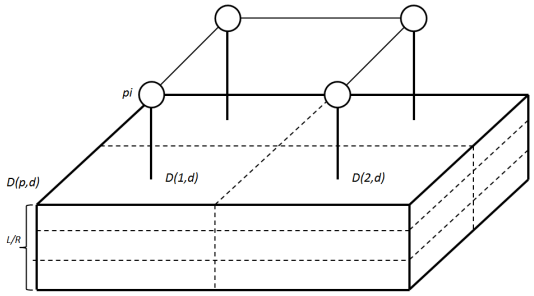

Figure 3.3: Markov Random Field graphical representation reduced by R

stereo input, and the disparity resolution needed for the application, the reduction of the disparity space might be acceptable or not. A classical decimation problem in the disparity space when performing this type of operations is shown in Figure 3.4. The analysis of the decimation that can be performed over the stereo input for a given disparity resolution is detailed in subsubsection 3.4.3.1

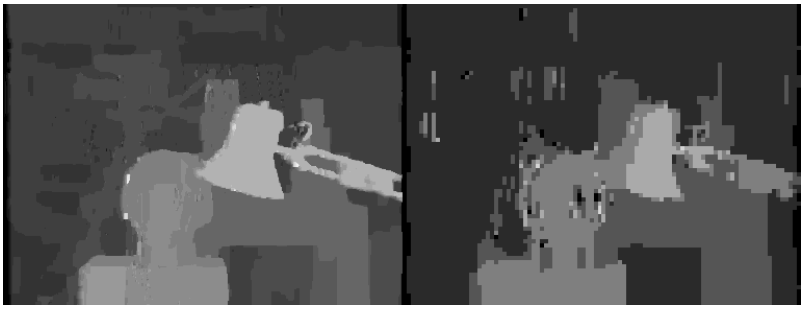

Figure 3.4: Quantization error due to low-resolution image input

3.4 Multiresolution energy minimization approach

The aim of the new formulation presented in this section is to reduce the complexity of the problem in order to enhance the performance of the optimization algorithm while maintaining the accuracy as much as possible. The new formulation is similar to the classical one presented in the previous section, but it reduces the number of hidden stochastic variables that describe the optimization problem. This reduction is achieved by the introduction of a new parameter, R, which will determine the number of stochastic variables considered in the problem and the final resolution of the disparity map obtained. As R increases, the number of variables will decrease and the performance of the algorithm will be improved, but the resolution of the final disparity map will be decreased, involving a less accurate reconstruction.

In the classical energy minimization framework, each pixel in the reference image has associated a disparity estimation. Each of these estimations is represented in the Markov Random Fields model as a discrete stochastic variable. Thus, the size of the MRF is determined in first instance by the size of the disparity map and in the last instance by the size of the reference image. In the multiresolution energy minimization framework proposed in this chapter, the final size of the disparity map is reduced by a factor R and the number of stochastic variables is reduced as a consequence.

The final width wd ad height hd of the disparity image, which will be reduced by a factor of R compared with the reference image, is defined by

$$w_d = floor\left(\frac{w}{R}\right), h_d = floor\left(\frac{h}{R}\right), R \in \mathbb{R}, R \geq 1 \tag{3.28}$$

Given the disparity map final size, a new subset of pixels can be defined for the new resolution coordinate system, one per pixel

$$p \in P \Leftrightarrow p = (x(p), y(p)), \forall x(p), y(p) \in \mathbb{N} / x(p) < w_d, y(p) < h_d \tag{3.29}$$

The purpose of this new resolution for the disparity image is to solve the correspondence problem for a limited subset of pixels and reducing the MRF complexity.

As in the classical energy minimization framework, the energy function used

for the multiresolution minimization framework will be made of a data term and a smoothing term

$$\min_{\overline{f}} E'(\overline{f}) \tag{3.30}$$

$$E'(\overline{f}) = E'_{data}(\overline{f}) + E'_{smooth}(\overline{f}) \tag{3.31}$$

3.4.1 Data term

The new data term E'_{data} preserves its aim to indicate the quality of the stereo correspondence based on a certain dissimilarity measure between left and right pixels. Compared to the original data term, it is only calculated in the subset P and uses interpolated values if necessary

$$E'_{data}(\overline{f}) = \sum_{p \in P} D'_p(f_p) \tag{3.32}$$

$$D'_p(f_p) = \sum_{c \in C} |\psi(I_L, x(p)R, y(p)R, c) - \psi(I_R, x(p)R - f_p, y(p)R, c)| \tag{3.33}$$

where ψ is an interpolation function such as the bilinear or bicubic functions. The most important thing to note in the new data term is that it uses the high-resolution original input images I_L and I_R for generating a cost associated with each disparity level and for each point p of the low-resolution disparity image. This reduction from the H set to the P subset reduces in R^2 times the size of the corresponding MRF. Note that while these calculations are made in the high resolution images, the number of labels is not reduced in this model, maintaining the same resolution as in the classic optimization framework.

As in subsection 3.3.1, other dissimilarity measures can be used in the multiresolution energy minimization framework for calculating E'_{data}. If an analogous dissimilarity function to the one introduced in (Klaus et al., 2006) is wanted to be used, the new expressions of the energy terms are as follows:

$$C_{SAD}\left(p, f_p\right) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \psi\left(I_L, x(u)R+i, y(u)R+j\right) - \psi\left(I_R, x(u)R-f_p+i, y(u)R+j\right) \tag{3.34}$$

$$\begin{aligned} C_{GRAD}\left(p, f_p\right) = & \sum_{i=-1}^{0} \sum_{j=-1}^{1} |\nabla_x I_L(x(u)R+i, y(u)R+j) \\ & - \nabla_x I_R(x(u)R-f_p+i, y(u)R+j)| \\ & + \sum_{i=-1}^{1} \sum_{j=-1}^{0} |\nabla_y I_L(x(u)R+i, y(u)R+j) \\ & - \nabla_y I_R(x(u)R-f_p+i, y(u)R+j)| \end{aligned} \tag{3.35}$$

$$D_p\left(f_p\right) = (1-\omega) * C_{SAD}(p, f_p) + \omega * C_{GRAD}\left(p, f_p\right) \tag{3.36}$$

where ∇_x is the forward gradient to the right, ∇_y the forward gradient to the bottom and ω is a parameter for weighting the absolute difference term and the gradient term. The dissimilarity measure must be summed up for all the different channels that conform the image.

For the sampling insensitive dissimilarity measure presented in (Birchfield and Tomasi, 1998), the new formulation is

$$\overline{d}\left(p, f_p, I_L, I_R\right) = \min_{x(p)R-f_p-\frac{1}{2} \leq k \leq x(p)R-f_p+\frac{1}{2}} \left|\psi\left(I_L, x(p)R, y(p)R\right) - \psi\left(\hat{I}_R(k), y(p)R\right)\right| \tag{3.37}$$

$$\overline{d}\left(p, f_p, I_R, I_L\right) = \min_{x(p)R-\frac{1}{2} \leq k \leq x(p)R+\frac{1}{2}} \left|\psi\left(\hat{I}_L, k, y(p)\right) - \psi\left(I_R, x(p)R-f_p, y(p)R\right)\right| \tag{3.38}$$

$$D_p\left(f_p\right) = \min\{\overline{d}\left(p, f_p, I_L, I_R\right), \overline{d}\left(p, f_p, I_R, I_L\right)\} \tag{3.39}$$

where $\hat{I}_L$ and $\hat{I}_R$ are the linearly interpolated functions between the sample points of the left and right scanline respectively.

3.4.2 Smooth term

The new smoothing term E_{smooth} is formulated in a similar way as in the classical energy minimization framework but accounting for the new coordinate system

$$E'_{smooth}\left(\overline{f}\right) = \sum_{\{p,q\}\in M} V'_{\{p,q\}}\left(f_p, f_q\right) \tag{3.40}$$

Mathematically, the neighboring set of pixels is defined by

$$p, q \in M \Leftrightarrow |x(p) - x(q)| + |y(p) - y(q)| = 1 \forall p, q \in H \tag{3.41}$$

in other words, two pixels p and q are neighbors if they are at one pixel of distance in image reduced by the parameter p.

The definitions of the functions $V_{\{p,q\}}$ in Equation 3.26 and Equation 3.27 for the classic energy minimization framework are still valid for the multiresolution minimization framework.

Similarly to the classical energy minimization framework, the problem can be represented as a graph. The introduction of the R parameter reduces the number of stochastic variables in the model, but the inclusion of the multiresolution minimization framework allows to keep the original resolution in the disparity space. An example is shown in Figure 3.5

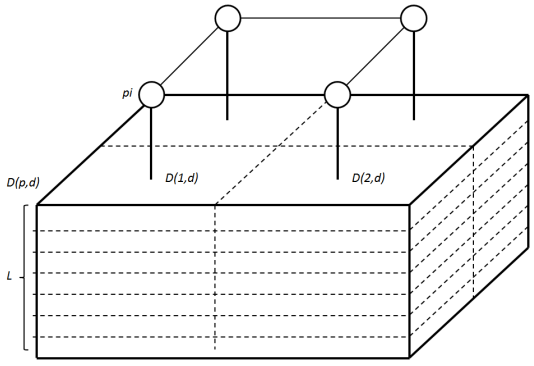

Figure 3.5: MRF graphical representation for the multiresolution energy minimization framework

3.4.3 Subsampling issues

The multiresolution minimization framework proposed in this section is based on the idea that subsampling the stereo input reducing its resolution by a factor of R does not substantially affect the informational content of the image. Obviously, subsampling involves a loss of information and, in certain situations, may absolutely change the image content. The effect of the subsampling in the disparity space and in the image space is herein independently analyzed, always focusing on the real-time robotics application.

3.4.3.1 Disparity space subsampling

For stereo correspondence applications, the resolution in the disparity space is correlated to the initial resolution of the stereo input. A higher resolution in the stereo input permits a higher sensitivity for determining correspondences and disparities between images. For a typical stereo camera with rectified images, the equations that determine the final resolution in the disparity space are

$$Z = \frac{fB}{d} \tag{3.42}$$

where f is the focal length of the pinhole camera, B is the stereo baseline, Z is the distance to the camera and d is the disparity. If a certain depth resolution $\triangle Z$ is required at a certain distance Z, the resolution needed in the image sensor would be

$$(d_1 - d_2) = \frac{fB}{Z} - \frac{fB}{Z + \triangle Z} = \frac{fB\triangle Z}{Z^2 + Z\triangle Z} \tag{3.43}$$

Therefore, if the sensor is known to have a certain pixel size pix_{size}, the decimation Dec that can be applied to the original image can be computed by the equation

$$Dec = \frac{(d_1 - d_2)}{pix_{size}} \tag{3.44}$$

Note that this decimation Dec is the complete decimation that may be applied to the stereo input. The disparity estimation is only affected by the horizontal resolution of the imput images, so the decimation Dec can be applied to both dimensions of the image, just the horizontal dimension or use another decimation factor for the

vertical decimation.

For instance, the decimation Dec for the PointGrey Bumblebee2 stereo camera has been calculated. Given that the PointGrey software performs a radial distortion removal over both left and right images and a rectification process, we can assume that epipolar lines are horizontal and that the camera has the following intrinsic parameters:

- $f = 3.8mm$
- $B = 12cm$
- $pix_{size} = 4.65\mu m$

if $\triangle Z = 3cm$ at $Z = 1m$ is acceptable, for example, for robotic navigation applications, $Dec = 2.86$ is obtained from Equation 3.44 and Equation 3.43. Thus, the horizontal resolution of the stereo input can be modified from 1032 to 361 pixels.

It is important to note that all the analysis has been carried out considering that no sub-pixel algorithms are used. Of course, a sub-pixel algorithm to enhance the precision of the disparity space resolution can also be applied. The results achieved here can be considered a worst-case situation.

3.4.3.2 Image space subsampling

The parameter R of the multiresolution minimization framework establishes the final resolution of the disparity depth map. The relationship between the resolution of an image and the size of the object projected is given by the following equation

$$Obj_{size} = \frac{Zpix_{size}RDec}{f} \tag{3.45}$$

where Obj_{size} is the minimum object size detectable at distance Z with camera parameters pix_{size} and f, being R the reduction factor of the model and Dec the decimation calculated in subsubsection 3.4.3.1 for obtaining a certain disparity resolution.

Therefore, if the same camera of the previous example is used and:

- decimation of $Dec = 2$

- reduction of $R = 3$
- at $Z = 5m$

a minimum object size of $Obj_{size} = 3.67cm$ can be detected in any direction perpendicular to the reference camera.

Thus, the subsampling in the image space is limited by the application, the camera used and the surroundings of the system. Note that performing a subsampling in the main reference image using the multiresolution minimization framework does not modify the resolution of the disparity space, which is very valuable information.

3.5 Impact of R on the performance

In this section a justification for the Markov Random Field problem reduction is described. Moreover, a theoretical quantification of the performance improvement incurred by including the factor R in both frameworks is analyzed. The analysis is going to be made considering that the algorithm used for finding the best disparity configuration is the alpha-expansion described in (Boykov et al., 2001), using the min-cut/max-flow algorithm described in (Boykov and Kolmogorov, 2004).

The alpha-expansion is an iterative algorithm that basically consists on recursively finding the best assignation of the disparity value α from pixels with any other value $\bar{\alpha}$. Firstly, for analyzing the complexity of the MRF minimization problem for different values of R, knowing how the alpha-expansion algorithm works is necessary. The algorithm can be divided into the following steps

- take a disparity level and construct the alpha-expansion graph
- perform a min-cut/max-flow algorithm in order to find the best label assignation for the graph
- repeat the process for all possible disparity levels

For analyzing the complexity of the whole alpha-expansion algorithm, the complexity of each step depending on the parameter R should be analyzed. Usually, the complexity of the first step, the alpha-expansion construction, must not be

ignored. However, the min-cut/max-flow algorithm that we are using (Boykov and Kolmogorov, 2004) reuse the trees constructed before and never start building them from scratch, which was demonstrated to lead to an important performance improvement over other min-cut/max-flow algorithms. Considering this methodology, it is going to be considered that, in the worst case, this step is independent from the parameter R and that no performance improvement is achieved reducing the MRF size. An example of an alpha-expansion graph is shown in Figure 3.6.

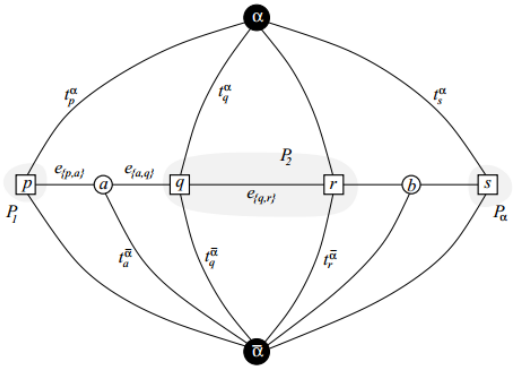

Figure 3.6: alpha-expansion graph extracted from (Boykov et al., 2001)

The number of nodes that appear in the graph can be calculated by

$$n = n_{vars} + n_{neighbors} \tag{3.46}$$

where n_{pix} is the number of stochastic variables, generally pixels for the stereo correspondence problem, $n_{neighbors}$ are the number of variables which have different label than its neighbor and n the number of nodes in the graph. Considering the number of different labels L and a complete random initial state of the stochastic variables, it can be stated that

$$\mathbb{P}\left(f_p \neq f_q\right) = \frac{L-1}{L} \approx 1.0 \tag{3.47}$$

$$n_{neighbors} = 2 \cdot n_{vars} \cdot \mathbb{P}\left(f_p \neq f_q\right) \approx 2 \cdot n_{vars} \tag{3.48}$$

where $\mathbb{P}\left(f_p \neq f_q\right)$ is the probability of a stochastic variable of having different label compared to its neighbors. The total number of nodes in the graph is one per stochastic variable and one per neighboring variable with different label assignation.

This last term is caused by the second order cliques relating one stochastic variable and its vicinity. The factor 2 is due to the fact that there are two second-order cliques per stochastic variable. Finally, the number of nodes in the graph can be calculated as

$$n \approx 3 \cdot n_{vars} \tag{3.49}$$

The general expression for the number of edges that appear in an alpha-expansion graph is

$$m \approx 2 \cdot n_{vars} + 3 \cdot n_{neighbors} \approx 8 \cdot n_{vars} \tag{3.50}$$

where m is the total number of edges in the graph. Of course, depending on the labeling of the stochastic variables, this calculations may differ considerably. However, they represent the worst-case, where the maximum number of nodes and edges might appear.

The second step consists on finding the min-cut on the alpha-expansion graph that performs the best assignation of disparity value α to stochastic variables with other disparity value. As said before, the algorithm used for this step is described in (Boykov and Kolmogorov, 2004), which can be classified in the *push flow* group of algorithms. In the paper, it is stated that the algorithm results in a theoretical worst-case complexity of $O(mn^2\,|C|)$ where $|C|$ is the cost of the minimum cut. However, it is also stated that practically, the algorithm has proven to outperform algorithms with complexity $O(mn^2)$, so it is going to be considered the complexity $O(mn^2)$ as a worst-case complexity. In terms of the number of stochastic variables, the min-cut/max-flow complexity can be expressed as

$$O(mn^2) = O(K \cdot n_{vars}^3) \tag{3.51}$$

where K is a constant.

For the last step, the process of finding a min-cut/max-flow for different graphs is repeated L times, one per each disparity value for completing one alpha-expansion iteration. Thus, the final complexity of one alpha-expansion iteration in terms of the number of variables and the number of labels is

$$O(K \cdot L \cdot n_{vars}^3) \tag{3.52}$$

Considering the reduction factor R for the **classic energy framework**, the number of variables and number of labels are

$$n_{vars,R} = \frac{n_{vars}}{R^2} \tag{3.53}$$

$$L_R = \frac{L}{R} \tag{3.54}$$

which leads to a complexity improvement of

$$O\left(L_R \cdot K \cdot n_{vars,R}^3\right) = O\left(\frac{L \cdot K \cdot n_{vars}^3}{R^7}\right) \tag{3.55}$$

Considering the reduction factor R for the **multiresolution framework**, the number of variables and labels are

$$n_{vars,R} = \frac{n_{vars}}{R^2} \tag{3.56}$$

$$L_R = L \tag{3.57}$$

which leads to a complexity improvement of

$$O\left(L_R \cdot K \cdot n_{vars,R}^3\right) = O\left(\frac{L \cdot K \cdot n_{vars}^3}{R^6}\right) \tag{3.58}$$

Based on these equations, two important conclusions may be drawn. Firstly, a reduction factor R in the Markov Random Field's network will likely have a dramatic impact on its complexity and, thus, on its performance. Stating that the performance is improved by a factor of R^6 might be too daring due to the complexity herein considered is a worst-case complexity. Usually the min-cut/max-flow algorithms perform a lot better. However, in section 3.7 the performance improvement will be determined experimentally. Secondly, and more likely, the performance lost when using R in the multiresolution framework compared to using R in the classic

framework is penalized just by a factor of R.

3.6 Impact of R on the MRF parameters

In this section the impact of the parameter R, which has been introduced in previous sections for both classic and multiresolution, is discussed. The reduction of the number of stochastic variables in both frameworks due to the reduction performed by R, clearly has an impact on both data and smoothing terms. The parameters that will be herein considered are λ and K, which appear in the smoothing term Equation 3.27. Tuning the parameters in the MRF model is crucial to achieve good and accurate results. For instance, incorrect λ values will lead to highly photometric consistent disparity values but also very different values between neighbors, i.e. a low probability prior. Moreover, the accuracy results might be pretty sensitive to little variations of the parameters. Thus, it is very important to study the impact of R on each term that make up the total energy function for properly tuning the parameters.

Firstly, the data term, as described in section 3.3 has the following definition

$$E_{data}(\overline{f}) = \sum_{p \in H} D_p(f_p) \tag{3.59}$$

The main impact of parameter R in the data term is a consequence of the reduction of the subspace of pixels analyzed from $p \in H$ to $p \in P$. Given that the number of pixels analyzed is reduced by a factor of R^2, it can be predicted that also the whole E_{data} term will suffer a reduction of R^2. The reduction will probably lead to a worse energy value, specially in the case of the classic framework. This effect would have an impact on the approximation, but it will not be considered due to its stochastic nature.

$$E'^{R}_{data}(\overline{f}) = E^{R}_{data}(\overline{f}) \approx \frac{E_{data}(\overline{f})}{R^2} \tag{3.60}$$

where $E'^{R}_{data}(\overline{f})$ is the data term obtained using the multiresolution framework, $E'^{R}_{data}(\overline{f})$ is the data term obtained using the classic one and $E_{data}(\overline{f})$ is the data term with no R applied.

Secondly, the smoothing term has the following definition previously described in section 3.3

$$E_{smooth}\left(\overline{f}\right) = \sum_{\{p,q\}\in N} V_{\{p,q\}}\left(f_p, f_q\right) \tag{3.61}$$

Each addend on the Equation 3.61 is related to a second order clique defined in the MRF. The reduction of the number of stochastic variables imply also a reduction on the number of second order cliques that have to be analyzed. The number of cliques depend on the number of stochastic variables in the following manner

$$n_{cliques} = 2 \cdot n_{vars} \tag{3.62}$$

where $n_{cliques}$ is the number of cliques in the MRF and n_{vars} the number of stochastic variables.

Consequently, at first glance, one might think that given that the number of cliques is proportional to the number of variables, the number of cliques when R is applied is reduced also by a factor of R^2, and as a result, also the smooth term. However, although the total number of cliques is reduced, the value of each one is increased. Figure 3.7 represents graphically what happens with the values of the cliques when R is applied. The graph on the left shows two cliques, V_{12} and V_{13} that appear in a classic bidimensional MRF with second order cliques. Each variable is represented by a white circle. On the right, a reduced graph with $R = 2$ is shown with grey circles representing the final stochastic variables.

The value of the cliques on the original-size problem can be calculated as

$$V_{12} = \lambda \cdot |f_1 - f_2| \tag{3.63}$$

$$V_{23} = \lambda \cdot |f_2 - f_3| \tag{3.64}$$

where f_i are the labels estimated for the correspondent stochastic variable i.

The value of the correspondent clique on the reduced-size problem is also

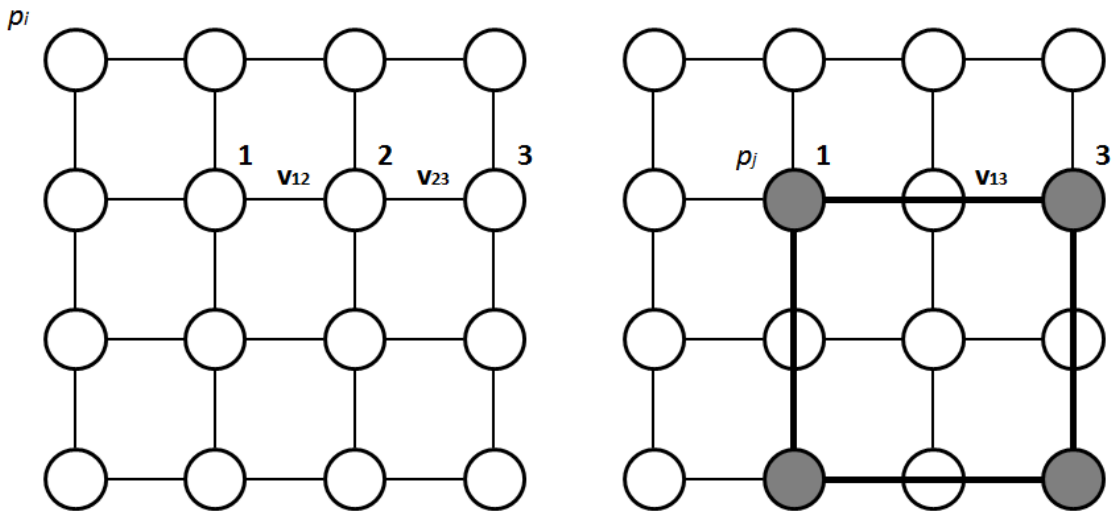

Figure 3.7: Impact of R in the MRF cliques. Left: classic MRF. Right: Reduced MRF by half

$$V_{13} = \lambda \cdot |f_1 - f_3| \tag{3.65}$$

Considering the disparity function from 1 to 3 a monotonic function, then it relates to the previous cliques as follows

$$V_{13} = \lambda \cdot |f_1 - f_2 + f_2 - f_3| \approx \lambda \cdot |f_1 - f_2| + \lambda \cdot |f_2 - f_3| = V_{12} + V_{23} \tag{3.66}$$

which is approximately the sum of the original cliques. Generally speaking, it can be stated that the value of the cliques for the reduced problem are R times greater than the cliques of the original problem. Thus, although the number of total cliques is reduced by the factor R^2, for the final value of the smooth term it has to be considered also this increment of their value. The final relation is

$$E'^{R}_{smooth} = \frac{E_{smooth}}{R} \tag{3.67}$$

where E'^{R}_{smooth} is the smoothing term for the multiresolution case and E_{smooth} the original one. This equation only holds for the multiresolution case. For the classic framework, another effect must be considered. The number of labels L is reduced by

a factor of R when the classic framework is applied. Thus, the absolute difference is reduced also by R, as is shown in

$$V_{13} = \lambda \cdot \left| \frac{f_1 - f_3}{R} \right| \approx \frac{V_{12} + V_{23}}{R} \tag{3.68}$$

which is a reduction of R compared to the multiresolution case. Finally it leads to a smoothing term of

$$E^R_{smooth} \approx \frac{E_{smooth}}{R^2} \tag{3.69}$$

where E^R_{smooth} is the smoothing term for the classic framework when R is applied.

Considering these mathematical relations, the parameters λ and K can be modified to obtain a continuous ratio between the data term and the smoothing term. This is crucial because if these parameters are not modified, applying a reduction R in the multiresolution framework will lead to a decrement of the data-smoothing ratio which leads to solutions which favour a lot more the smooth disparities than in the original problem. Considering Equation 3.60 and Equation 3.69, the values of the parameters for the classic framework are the following

$$\lambda_R = \lambda \tag{3.70}$$

$$K_R = \frac{K}{R} \tag{3.71}$$

Considering Equation 3.60 and Equation 3.67 the final parameters for the multiresolution case are

$$\lambda'_R = \frac{\lambda}{R} \tag{3.72}$$

$$K'_R = K \tag{3.73}$$

In order to support theses mathematical results, several experiments have been carried out. Some images have been used to analyze the evolution of both data and smoothing terms for different values of R. Three images from the Middlebury data

set have been used: Tsukuba, Teddy and Venus. More detailed information about the data set can be found in subsection 3.7.1. The parameters used for these images are shown in Table 3.1, applying Equation 3.72 and Equation 3.73.

	λ	K
Tsukuba	4	8
Teddy	4	30
Venus	4	10

Table 3.1: Parameters for tests

The data-smoothing ratio indicates if both terms are equally weighted while incrementing R. The results for the ratios are shown in Table 3.2. The table shows that the ratios are maintained practically constant with R, so the mathematical results are supported experimentally.

R	1	2	3	4	5	6
Tsukuba	7.9291	8.0114	8.3633	7.8994	8.8919	9.6015
Teddy	7.0972	7.6964	8.0022	8.2548	8.608	8.4096
Venus	20.9144	20.4703	20.6972	20.5071	20.1659	20.8192

Table 3.2: Data-smoothing ratio

3.7 Experimental Results

In this section the new multiresolution energy minimization framework presented in section 3.4 is evaluated and compared to the classical one. For the analysis, two different datasets have been used: Middlebury and KITTI.

3.7.1 Data sets

The first data set is the well-known Middlebury stereo data set (Scharstein and Szeliski, 2002). Their stereo pairs have been widely used for stereo research and they are famous for being a reference set for algorithm comparison and ranking. The stereo set and a table with the rank of the most important stereo algorithms can be found in the Middlebury's stereo website (http://vision.middlebury.edu/stereo/). The set used in theses experiments contains six images, which are shown in Figure 3.8. It is

Figure 3.8: Middlebury data set

a group of test images with known high-quality true disparities. The main drawback of this test set is that images have been taken in ad controlled environment and the disposition of the objects may not be representative for real applications. That is, they have the same camera gain (very important for intensity and color correlation) and the objects in the scene are mainly positioned perpendicular to the camera's main axis.

The second data set used for the analysis is the KITTI Vision Benchmark Suite (Geiger et al., 2012). It consists of 194 training pairs and 195 test pairs captured while driving around the mid-size city of Karlsruhe, in rural areas and in highways. The main purpose of this data set is to validate stereo, optical flow and odometry algorithms for the autonomous driving application. An example of an image of this data set is shown in Figure 3.9.

KITTI data set is very interesting because it represents a real application, with real illumination, occlusions, general non front-to-parallel planes, etc. However, the true disparities were obtained using a Velodyne Laserscanner located on top of a vehicle, so only an sparse ground truth is given for training. An example of the sparse ground truth of the image in Figure 3.9 is shown in Figure 3.10.

In the KITTI web page a group of tools can be downloaded for evaluating own

Figure 3.9: KITTI image example

Figure 3.10: KITTI sparse ground truth

disparity estimations. In these tools there is the option for interpolating the ground truth values for obtaining a dense ground truth. An example is shown in Figure 3.11. The main disadvantage of this data set is that neither the sparse ground truth, nor the interpolated one are precise enough for evaluating high-end disparity estimations.

3.7.2 Implementation

For the multiresolution energy minimization framework described in section 3.4, any energy function proposed in the literature can be used. Moreover, any energy minimization algorithm that is appropriate for solving the classical framework is also appropriate for solving the new energy minimization framework.

Many energy functions have been proposed in the literature. In order to keep the analysis as simple as possible, the simplest data and smoothing terms have been selected, which correspond to Equation 3.33 for the data term and Equation 3.27 for the smoothing term. The parameters of the energy function have been set dependent of R as explained in section 3.6.

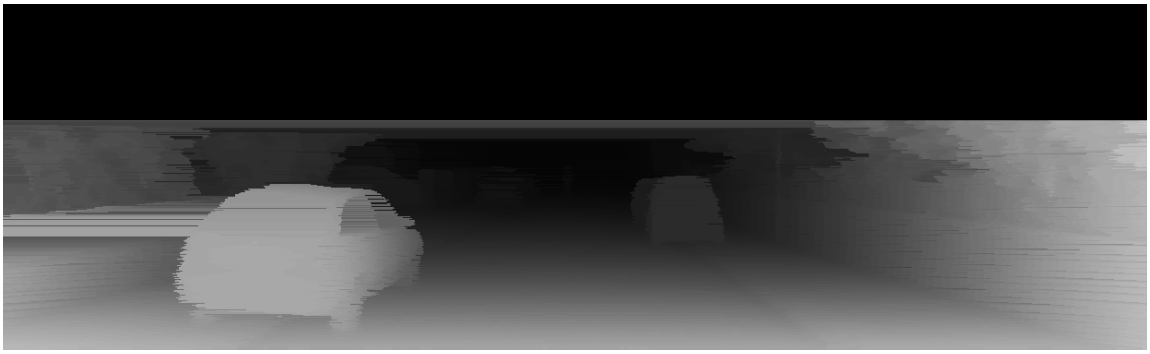

Figure 3.11: KITTI interpolated ground truth

The optimization algorithm was selected based on the comparative study made in (Szeliski et al., 2006). Several energy minimization methods are analyzed and compared, concluding that the alpha-expansion (Boykov et al., 2001) and the TRW-S (Wainwright et al., 2002) algorithms obtain the lowest energies. However, comparing the performance, alpha-expansion seems to be clearly the best between both methods. For these reasons, alpha-expansion has been used in all the experiments carried out in this section. Given that alpha-expansion has been used, the theoretical performance impact deduced in section 3.5 can be studied here empirically.

For the tests, the algorithms used were programmed in C++, using the implementation of (Szeliski et al., 2008). The algorithms were run on an Intel i7-2600 at 3.4 GHz CPU, just in one thread, and 8 GB of RAM. Windows was used as operating system.

3.7.3 Accuracy analysis

In this section both frameworks are evaluated and compared in terms of accuracy. The quality metrics used are the same as the ones described in (Scharstein and Szeliski, 2002), the root-mean-squared error RMS and the bad-pixel percentage B.

$$RMS = \left(\frac{1}{N} \sum_{(x,y)} |d_C(x,y) - d_T(x,y)|^2 \right)^{\frac{1}{2}} \tag{3.74}$$

$$B = \frac{1}{N} \sum_{(x,y)} \left(|d_C(x,y) - d_T(x,y)| > \delta_d \right) \tag{3.75}$$

where N is the total number of pixels, δ_d is a threshold parameter and d_C and d_T are the disparities calculated and the true disparities respectively. All the statistics calculated using these equations were obtained over all occluded, textureless and discontinuity regions. As recommended in (Scharstein and Szeliski, 2002), the δ_d parameter has been set to 1 in all the analyses made in the Middlebury dataset. For the KITTI dataset, generally a $\delta_d = 3$ is used, but for being able to analyze the subsampling effect on the disparity dimension, a more restrictive threshold $\delta_d = 1$ has been used. For evaluating the RMS and the B error measures, the true disparity image has been subsampled when needed using the bilinear interpolation function in order to perform a pixel-by-pixel comparison.

Several benchmarks have been run in order to compare both frameworks. For each framework, several solutions have been obtained by using the implementation proposed in subsection 3.7.2 and through different values of the reduction factor R. The reduction factor R applied to the classical framework will determine the decimation of the resolution performed over the input stereo pair to obtain a disparity map with a lower resolution.

In Figure 3.12 a benchmark analysis of the RMS error for the Middlebury test images is shown. The aim of this figure is to analyze the evolution of the RMS error when the R parameter is increased in both frameworks. These charts show two different trends in the evolution of the RMS. The results on the Tsukuba, Venus and Teddy images show that the error for the classic framework increases rapidly with R, while for the multiresolution case it increases with a considerably lower rate. Thus, it can be stated that in these cases the multiresolution framework outperforms the classic one. However, the behavior of the RMS on the Cones, Aloe and Baby images is quite different. The evolution of the error in the classic framework is much more similar to the multiresolution case compared to the previous images. Moreover, both frameworks show an uncontrolled behavior for certain values of R. Therefore, for this last set of images, one cannot conclude that any framework outperforms the other.

This behavior of the RMS error in certain situations can be explained by the disparity estimation method it is used. Analyzing in more detail the charts in Figure 3.12, one can see that the images where the multiresolution framework outperforms the classic one are also the images where a lower error is obtained. This means that the initial estimation, that is for $R = 1$, of the optimization algorithm is reasonably accurate. Then, when R is increased, the error for the classic framework

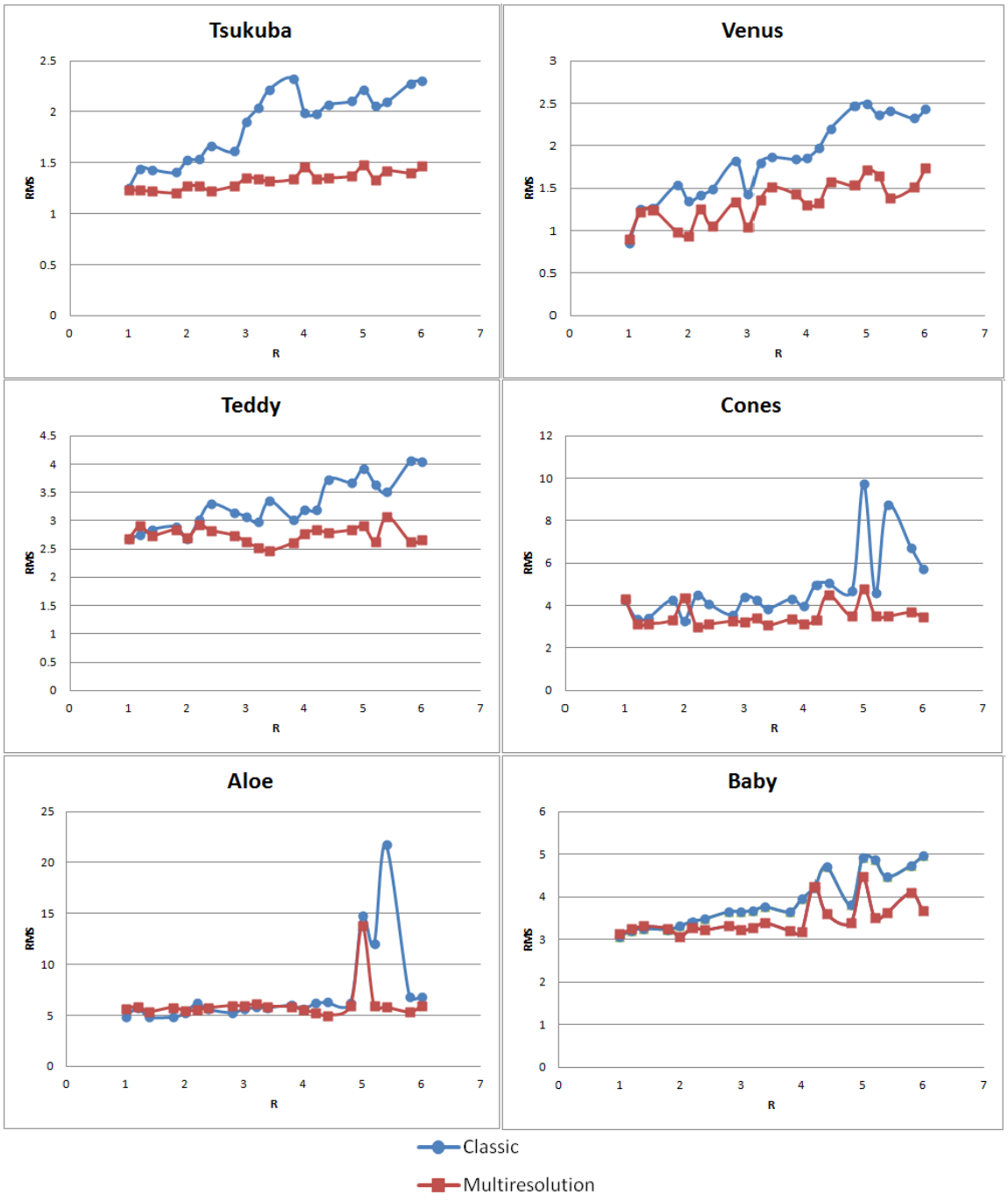

Figure 3.12: Middlebury RMS error

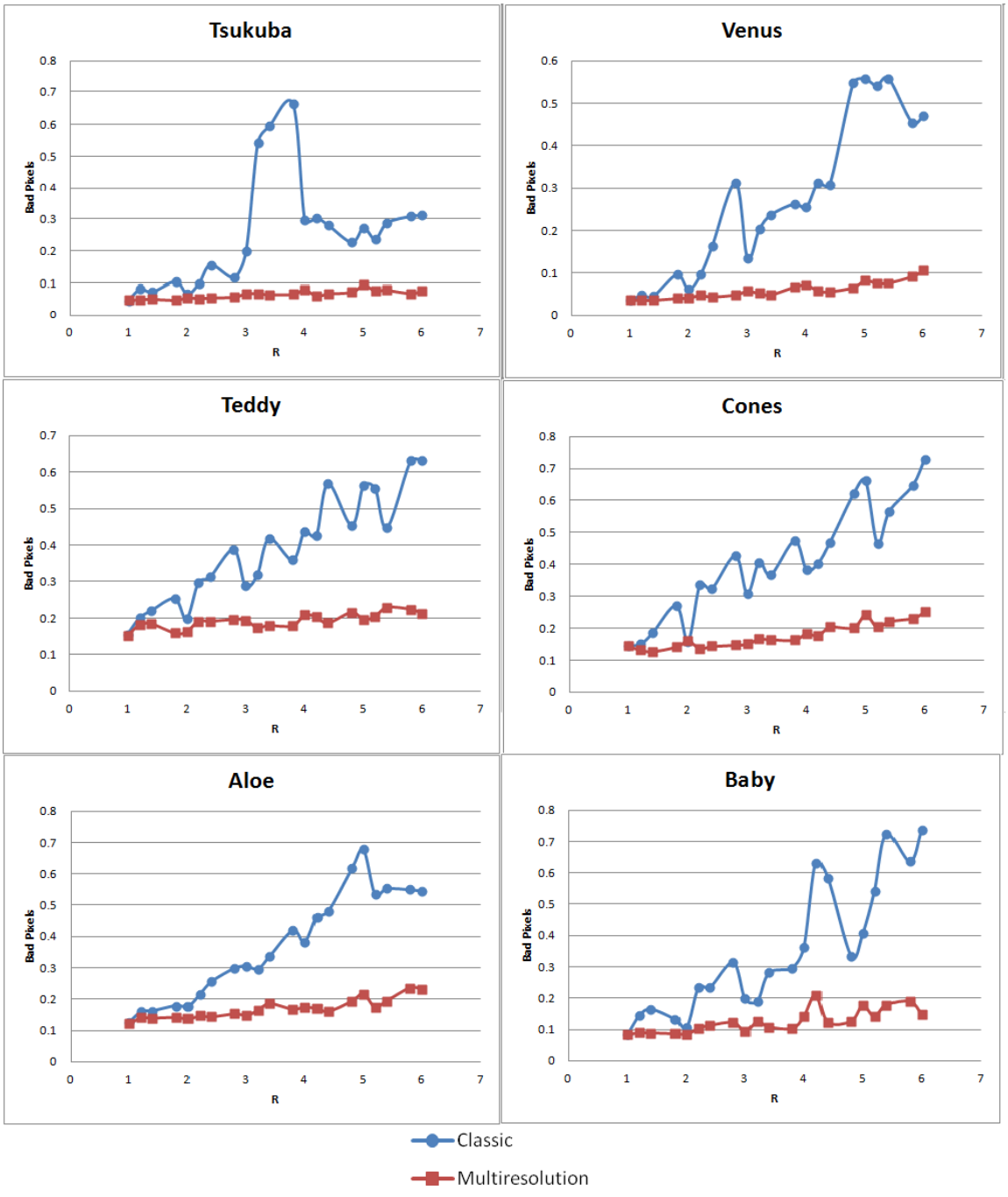

Figure 3.13: Middlebury B error

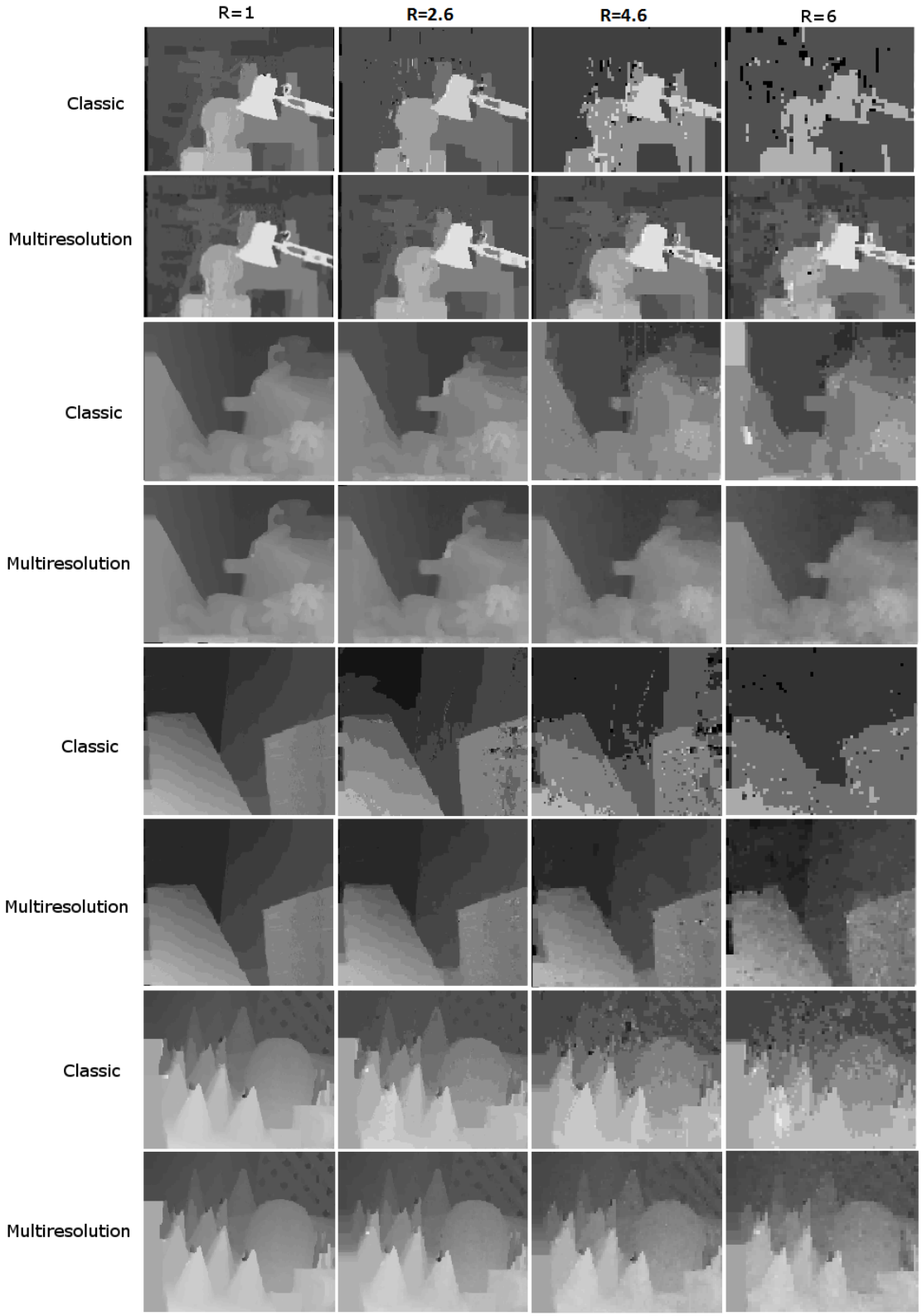

Figure 3.14: Comparison of Middlebury disparity maps obtained by using classic and multiresolution frameworks

increases mainly due to the subsampling effect on the disparity space. Meanwhile, the disparity estimation for $R = 1$ for the Cones, Aloe and Baby images is not as good as the estimations obtained before. This error is mainly related to the general bad performance of the energy minimization algorithms with these parameters. Given that the RMS error calculation is an unbounded error measure that penalizes higher disparity differences, it can be observed the error caused by the bad disparity estimation due to the algorithm itself is predominant compared to the error caused by subsampling issues. Therefore, based on the RMS error, one cannot conclude that the multiresolution framework outperforms the classic one. Moreover, it is suggested that the RMS error might not be a good measure for comparing results when not high accurate estimations are obtained.

The number bad-pixels error measure has been also analyzed and shown in Figure 3.13. Note that the B error is a bounded error measure with a certain threshold δ_d which has been set to one for these experiments. For this error measure, more homogeneous results have been obtained. All the charts shown in Figure 3.13 indicate that disparity estimations suffer a high increment of B proportionally to R when the classic framework is used. Moreover, increasing R generally involves and uncontrolled behavior. Owing to the reduction of the resolution in the disparity space, the error depends on the fortune of having the correct labels between the set of labels available in the model. That is, the cyclical increase and decrease of some of the errors is justified as a subsampling of the disparity space issue. Meanwhile, B for the multiresolution case behaves in a more stable way and clearly outperforms the classic framework.

This result leads to the prevention of using downsampling in the stereo input if good RMS error and also good resolution in the disparity domain are required, as in the case of robotic navigation applications. The Venus test case is probably the most important one given that it is made of non-fronto-parallel planes and thus every disparity value exists in the true disparity image.

The relative error for each image and R value is shown in Figure 3.15 and in Figure 3.16. It has been calculated as the actual error for a certain image, R and framework divided by the best solution found for that image and framework. It shows how the solution is deteriorated when the R parameter is increased. The first row in both figures shows the RMS relative error for both frameworks. As stated before, when the RMS error was analyzed, the charts do not show a clear dominant technique. Although the relative error keeps mostly constant in the multiresolution

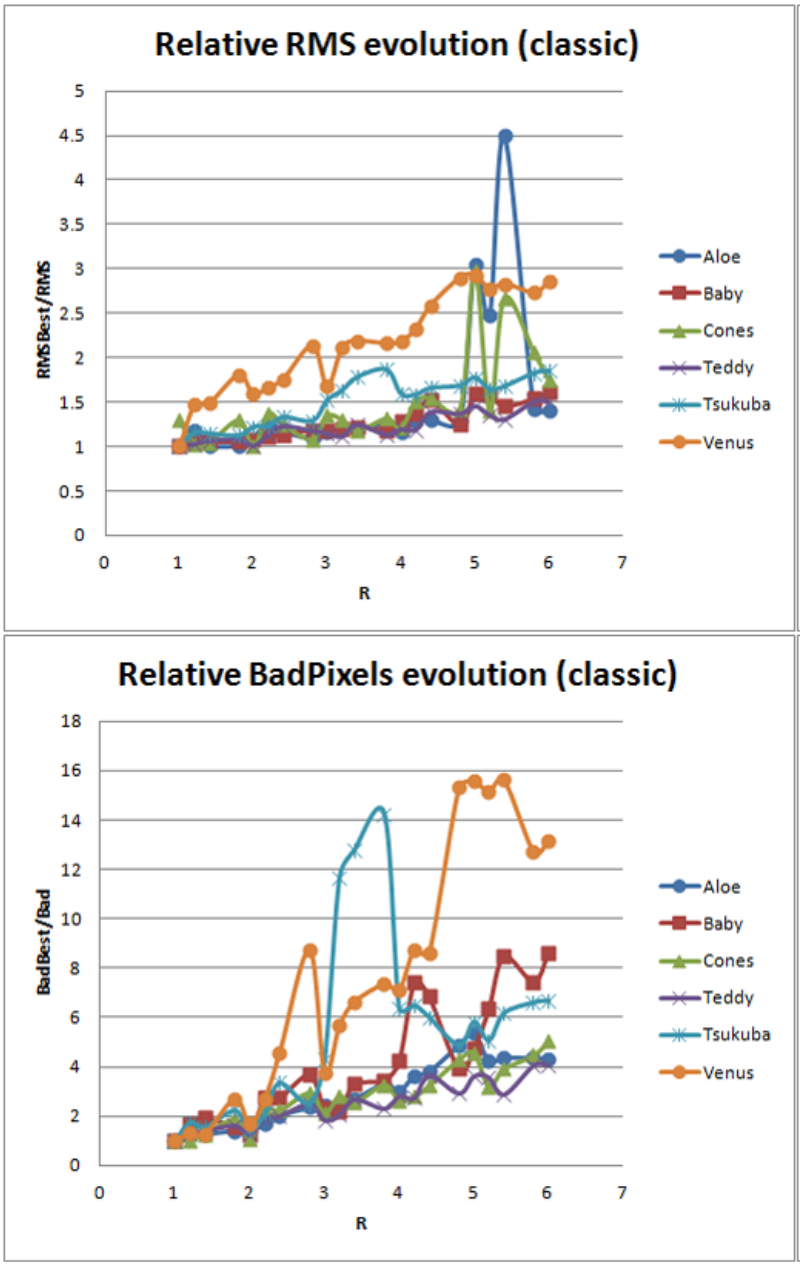

Figure 3.15: Evolution of the RMS error for the classical framework depending on R ($RMSbest1$)

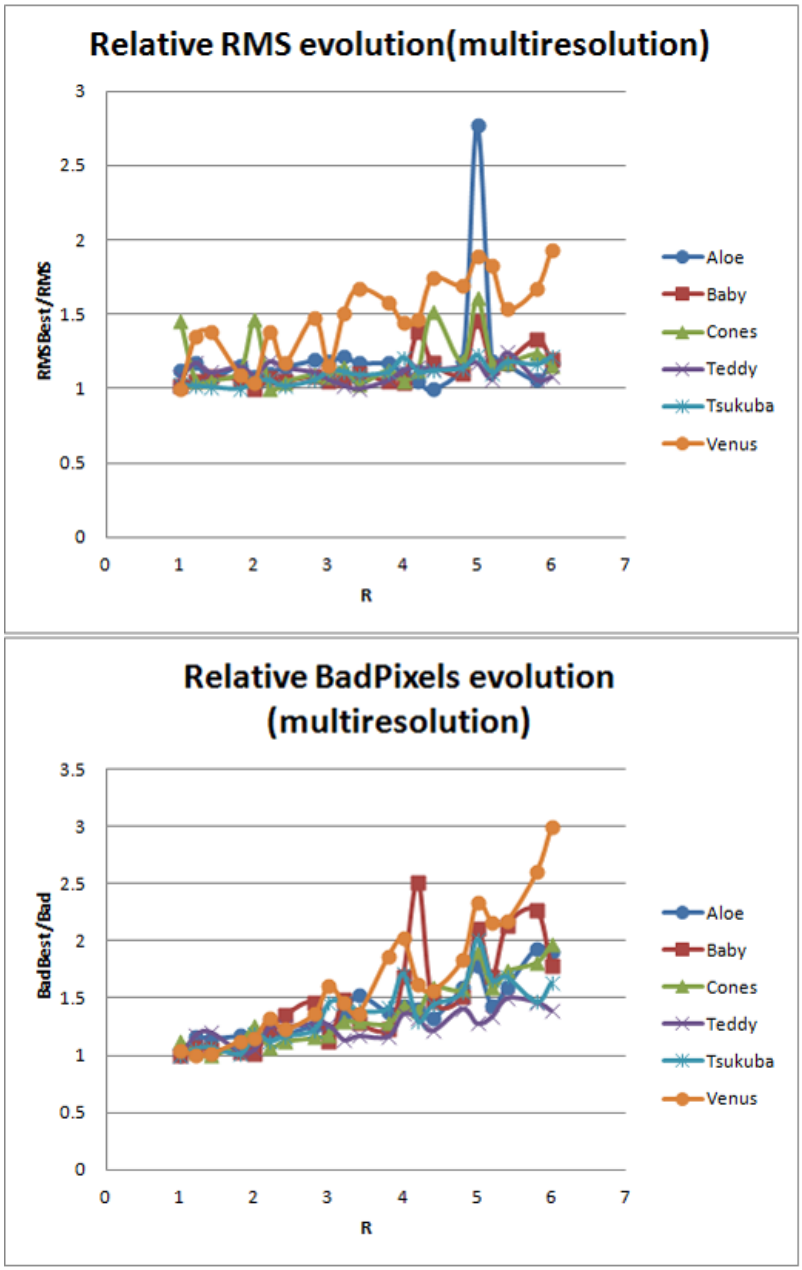

Figure 3.16: Evolution of the RMS error for the multiresolution framework depending on R ($RMSbest1$)

case, it is also maintained under 1.5 for the classic one for most images. In contrast, the second row shows how the relative B measure performs considerably better in the multiresolution case. However, an increment of a 100% of the error have been obtained for values of $R = 6$ and approximately 50% for $R = 4$. Thus, it is important to note that although better accuracy results where obtained using the multiresolution framework, always an increment on the R parameter involves an increment on both RMS and B error, so a trade-off between accuracy and performance has to be considered.

The resulting disparity maps for various Middlebury test cases varying R are shown in Figure 3.14.

Similar results have been obtained for the KITTI data set. The RMS error calculated for the KITTI data set, similarly to the Middlebury experiments, has shown to behave in an uncontrolled manner for some images. Thus, no conclusive results have been obtained considering this error measure. If B is analyzed,(Figure 3.17) several differences must be highlighted. Firstly, it shall be noted that the error measured for all images is considerably higher than those measured for the Middlebury data set. As previously mentioned, this set of images were taken in an uncontrolled environment and are considered a better test for real applications. Thus, one important result to point out is that for real scenarios, stereo algorithms using energy minimization frameworks optimized using alpha-expansion (graph-cuts) perform generally very poorly. Compared to the best algorithm (Yamaguchi et al., 2012) reported in the KITTI web-page, obtaining an average 7.78% of bad-pixels with an error threshold of 2 considered, the error difference is very noticeable. These results can be attributed to the form of the energy function, the parameters used or the algorithm used for minimization. Secondly, the increment ratio of B is also higher for both frameworks compared to the ones obtained in the Middlebury data set. However, comparing both frameworks, the multiresolution outperforms constantly the classic one for each image analyzed.

Finally, the disparities estimated for a pair of KITTI images and various values of R are shown in Figure 3.18

3.7.4 Performance analysis

For the performance analysis presented in this section, five whole alpha-expansion optimizations with five iterations each for both classic and multiresolution frameworks

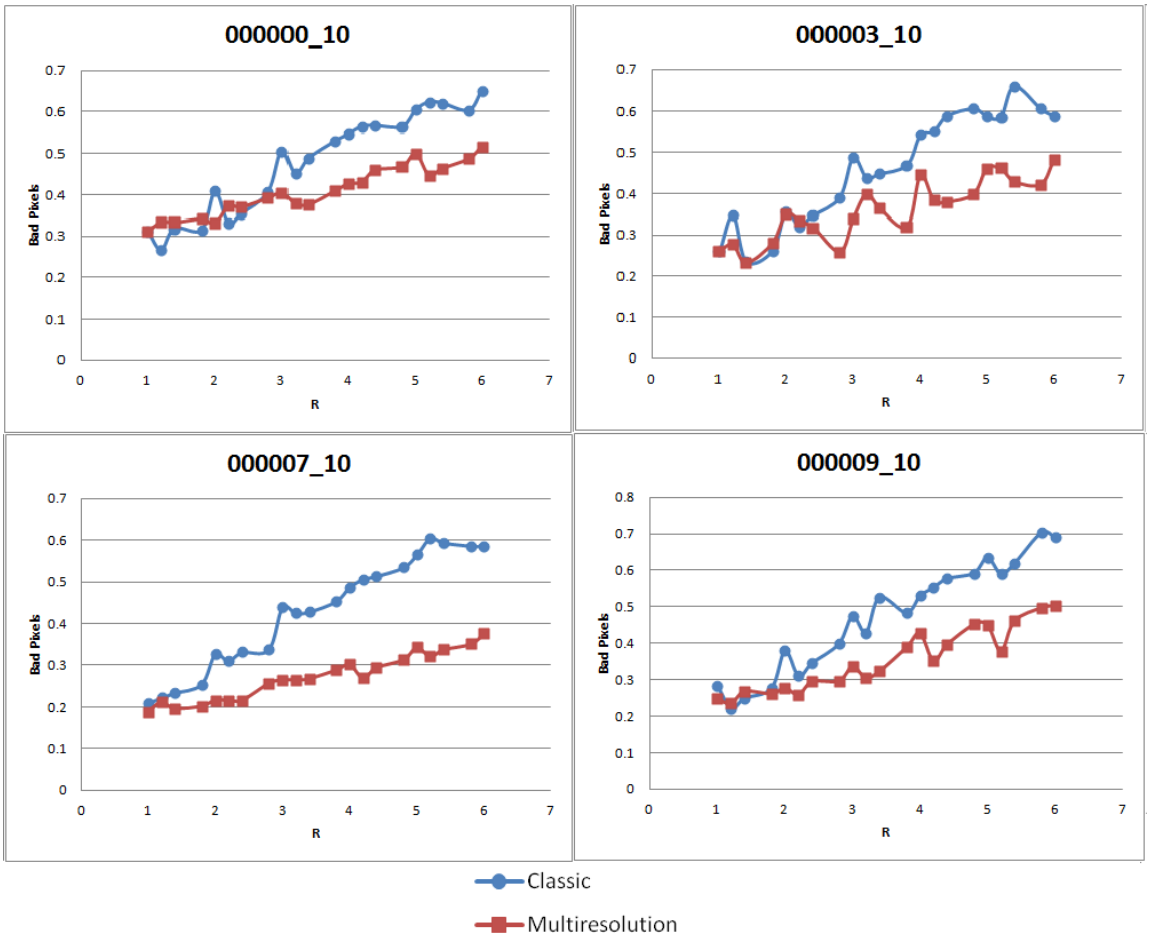

Figure 3.17: Bad pixels obtained for some images of the KITTI data set

have been run. Five individual optimizations where run for obtaining consistent timing results and five iterations each for obtaining reasonably low energy estimations.

Firstly, the impact of R in the alpha-expansion performance is analyzed. As previously mentioned in the accuracy analysis, in the classical framework R is used to perform a subsampling in the input stereo pair. In Figure 3.19 and Figure 3.20 the evolution of the elapsed time for various values of R is presented. Although only one image of each data set is actually shown, the same evolution has been observed in all the other images. Only the computational time for solving the MRF, i.e the alpha-expansion iterations where measured, not considering the computational time elapsed for calculating the cost function. The figures show that even low R values, such as 1.3, can lead to a dramatic impact on the computational time. For both classic and multiresolution frameworks, a hyperbolic evolution of the performance

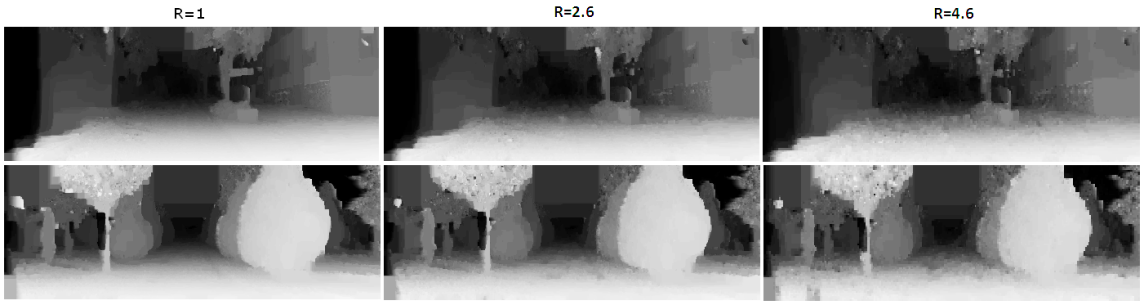

Figure 3.18: Disparity maps obtained for two images of the KITTI data set

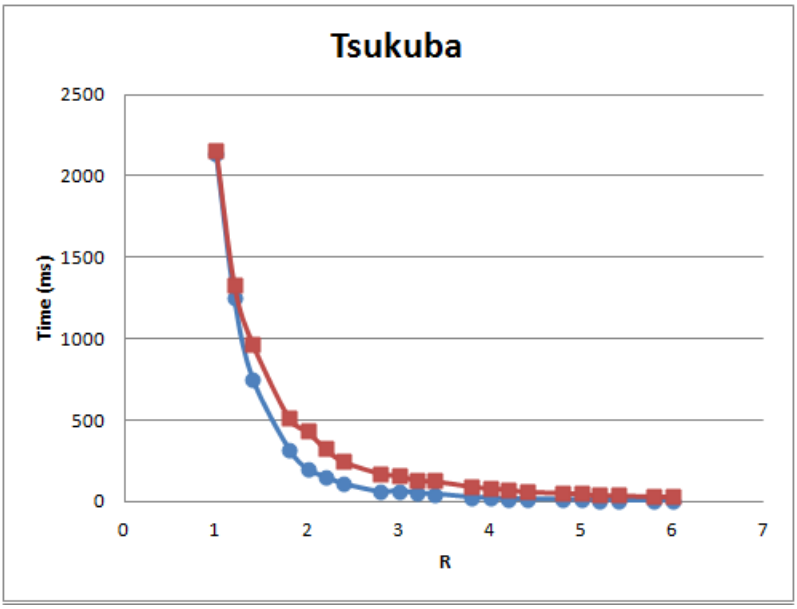

Figure 3.19: Evolution of the computing time depending on R for Tsukuba image

with R can be deduced.

In section 3.5 a theoretical performance analysis was presented to demonstrate the performance improvement of the multiresolution minimization framework compared to the classic framework. Here, some experiments have been carried out to analyze which is the real performance improvement for various values of R. Being tm the time spent for the multiresolution case and tc the time spent for the classic framework using $R = 1$, Figure 3.21 shows the polynomial regression carried out for the Tsukuba image pair and the performance improvement for all the images in the Middlebury data set. The performance increases monotonically with R, obtaining a speedup factor of up to 70x. A parabolic function can be deduced from the data (polynomial regression parameters of second order are shown in Figure 3.21). Therefore, it can be

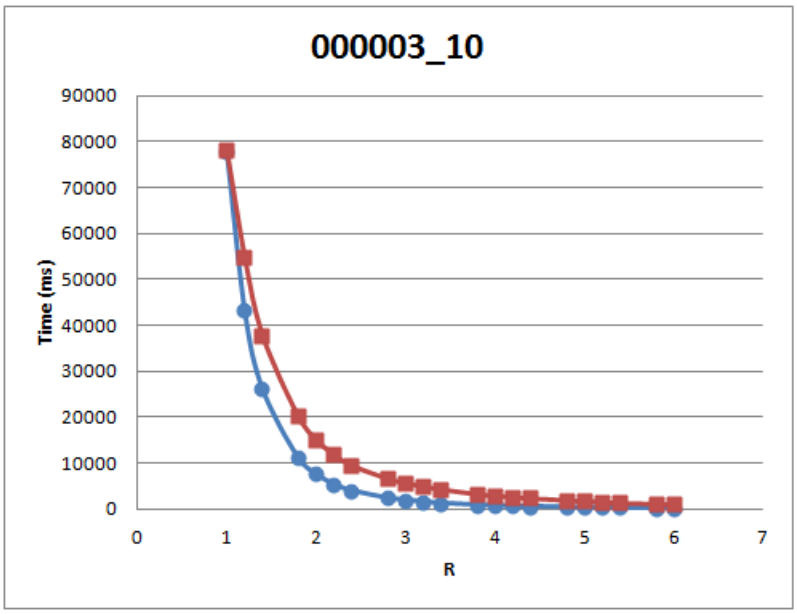

Figure 3.20: Evolution of the computing time depending on R for 000003_10 image

proved empirically that the real reduction on computational complexity is dependent on R^2.

In section 3.5 it was deduced that the performance penalty of using the multiresolution instead of the classic framework is dependent on R. This relation has been also empirically tested and shown in Figure 3.22 and Figure 3.23. In this case, the predicted relation is proved to be correct, multiresolution tests run R times slower than the corresponding classic ones basically because of the increment on the number of disparities analyzed in each problem.

For real-time applications a maximum computing time of milliseconds is available for stereo analysis. Figure 3.24 shows the frames per second that can be obtained for high values of R for four different images. For these experiments, it was also included the computational time spent for the cost calculation. Both Teddy and Cones stereo input images have a higher bit resolution and much more disparity levels to analyze than the other two data sets, hence an impact on the performance is evident. However, nearly five frames per second are achieved in the Teddy and Cones case whereas ten fps are achieved in Venus and approximately 20 in Tsukuba.

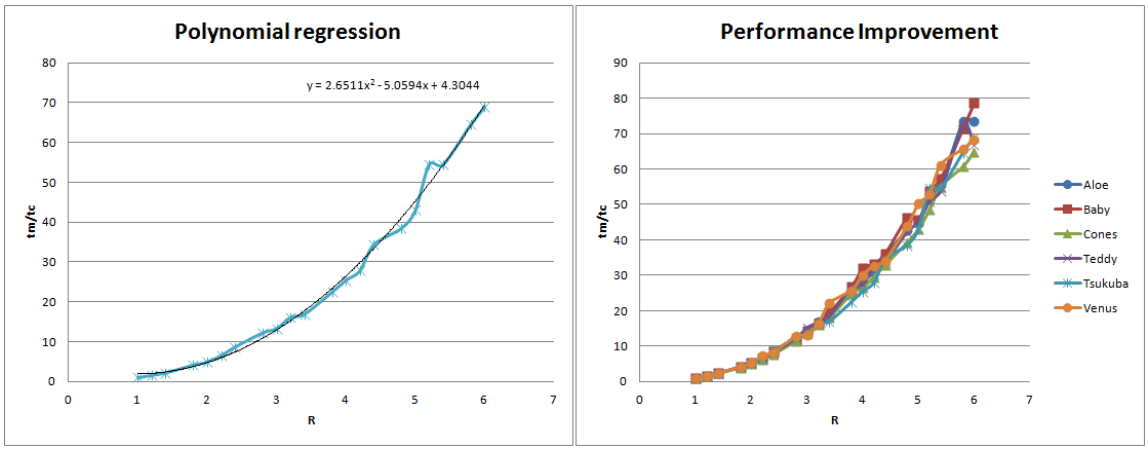

Figure 3.21: Performance enhancement of incrementing R. t_{c1} is the computing time needed for the classical model to find a solution for $R = 1$. T_m is the computing time for multiresolution model to find a soution for each R on the graph

3.8 Conclusions

A new multiresolution energy minimization framework for stereo matching has been presented in this chapter. This framework extends the classic energy minimization framework with the main objective of reducing the size of the MRF problem and, thus, enhancing its performance. New data and smoothing terms have been defined including a new parameter R in the model. The impact of this term on the alpha-expansion performance has been analyzed both theoretically in section 3.5 and empirically in subsection 3.7.4. Moreover, the impact of R in the MRF parameters has been studied in section 3.6. The main advantage of this new framework is the reduction of the computational complexity of the stereo matching proportionally to R^2 while maintaining the resolution in the disparity domain. Moreover, it has been demonstrated that traditional algorithms designed for solving MRFs, such as alpha-expansion, can be used. Note that the framework presented is compatible with new parallel algorithms such as CUDAcuts (Vineet and Narayanan, 2008).

Traditionally, an option for reducing the size of the problem is to reduce the resolution of the stereo input. In order to avoid decimation in the disparity domain (number of disparity labels), a subsampling of the input stereo pair is not recommended. As is shown in section 3.7, performing subsampling operations in the input stereo image will probably lead to uncontrolled behavior of the final disparity error measure. In contrast, using the framework herein proposed, no decimation in the

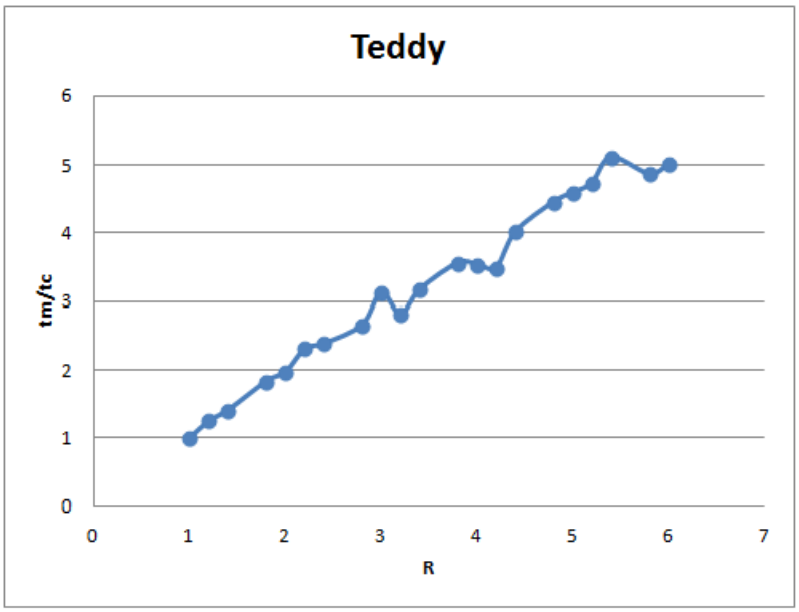

Figure 3.22: Ratio between the elapsed time for multiresolution and classic frameworks for Teddy image

disparity space is performed, although a reduction in the size of the problem is still achieved. However, note that although better accuracy results were obtained using the multiresolution framework, an increment on the R parameter always involves an increment on both RMS and B error, so a trade-off between accuracy and performance has to be considered.

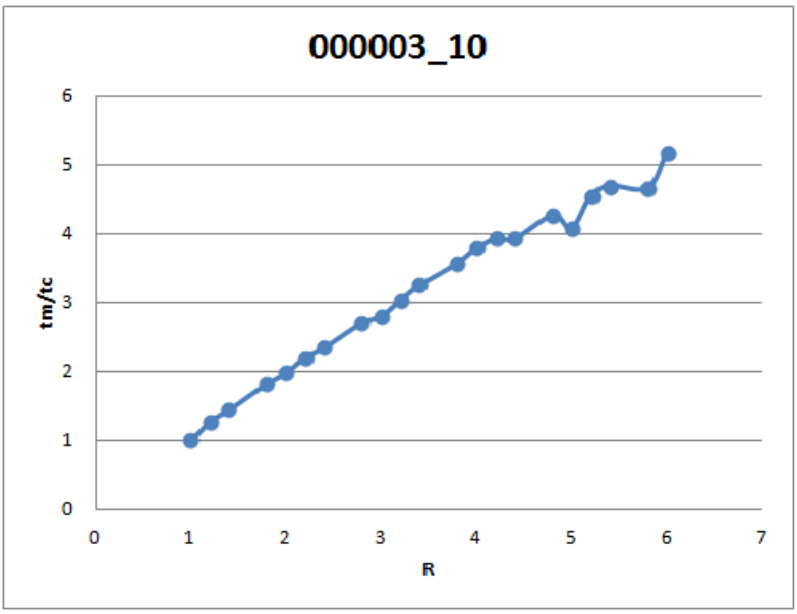

Figure 3.23: Ratio between the elapsed time for multiresolution and classic frameworks for 000003_10 image

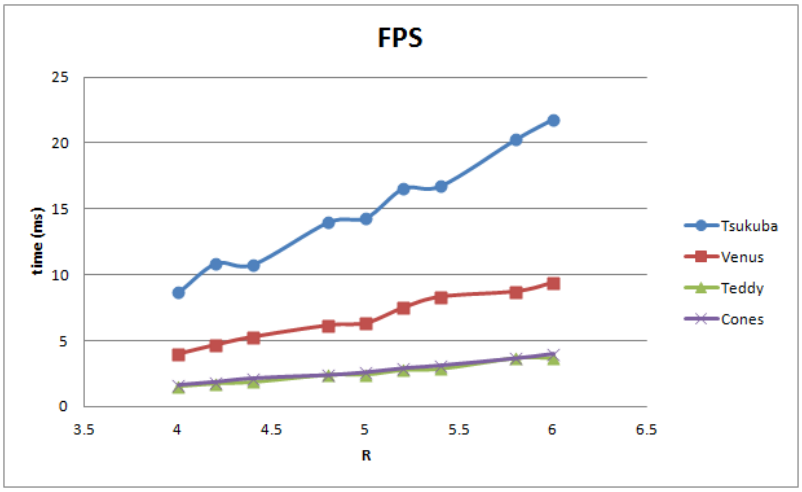

Figure 3.24: Real-time performance analysis for a single iteration of the multiresolution model, varying the R parameter

4

Genetic algorithm for stereo refinement

Previously, a framework for improving the performance of global stereo algorithms based on Markov Random Field (MRF) has been proposed along Chapter 2. In order to test the framework, the alpha-expansion algorithm (Boykov et al., 2001) was used as the optimization algorithm for obtaining a good estimation for the disparity map. However, many other optimization algorithms proposed in the literature for solving MRF problems could be used. In this chapter, the application of a genetic algorithm for minimizing the MRF's energy function related to the stereo correspondence problem is discussed. Applied to stereo vision, genetic algorithms are flexible in the cost function configuration, permit global reasoning and are not label-based. These characteristics allow to test different configurations for the energy functions, consider the disparity map to be continuous compared to the label-based state of the art minimization algorithms and to include the refinement step in the minimization process itself.

This chapter analyses the second step described in Figure 1.1 in the Introduction chapter. Therefore, it receives the disparity map calculated in the previous step and enhances it to maximize the accuracy in the following steps during the pose estimation process. The contributions of the algorithm described in this chapter are the application of evolutionary techniques for disparity refinement that enables any energy function to be used, and the proposal of a new energy function that deals with occlusions.

The chapter is organized as follows: section 4.1 is a brief introduction to the chapter, section 4.2 describes the state of the art of previous genetic algorithms applied to the stereo correspondence problem, section 4.3 explains in detail the for-

mulation of the genetic algorithm proposed, section 4.4 describes how the algorithm can be implemented in GPU using CUDA, section 4.5 gets into the justification of the philosophy of the genetic algorithm applied to stereo, section 4.6 shows some results obtained for the Middlebury and KITTI data sets and finally in section 4.7 some conclusions are drawn.

4.1 Introduction

The stereo correspondence problem has been widely studied in the literature. As previously mentioned in Chapter 2, most stereo algorithms can be classified in one of these five big groups: sparse-feature based, window or block matching, dynamic programming (also known as scan-line optimization), cooperative algorithms and MRF-based. Continuing the analysis of the previous chapter and the whole thesis, this chapter is focused on how to obtain a good disparity map estimation and refinement through the minimization of energy functions in the MRF framework using a genetic algorithm. The energy minimization framework combined with a refinement process has proven to achieve very accurate results (Alahari et al., 2010) (Scharstein and Szeliski, 2002) (Klaus et al., 2006).

MRF-based algorithms establish a relationship between the probability of a certain disparity map given the stereo pair and a specific energy function. Thus, the aim of these algorithms is to obtain a high probability disparity map, which translates into finding a configuration that minimizes the given energy function. The definition of the energy function, its parameters, and the minimization algorithm used will determine the quality of the estimations. It is still under discussion which is the energy function that best represents the reality and which minimization algorithm is capable of finding a global minimum for that energy function. Currently, some algorithms have some kind of unstable behavior (Pearl, 1982) while others impose some restrictions on the structure of the energy to be minimized (Boykov et al., 2001) (Kolmogorov and Zabin, 2004).

Genetic algorithms are non-linear heuristic evolutionary search algorithms that are based on the natural evolution process. They have been traditionally used for successfully solving many different kinds of optimization problems. One of their most useful characteristics is that they are very flexible in their configuration. However, are not guaranteed to find optimal solutions. A genetic algorithm can be used in any

problem if it is possible determine

- How a solution to the problem is represented (genome)
- How good a certain solution is (fitness function)
- How to combine two different solutions (crossover)
- How a solution can be randomly altered (mutation)

Although virtually any optimization problem could be solved using genetic algorithms, the proper definition of these elements, their parameters and the nature of the problem will strongly determine the performance of this type of algorithms.

In this chapter, a stereo algorithm using a genetic optimization approach is described. Due to its flexibility, both stereo matching and refinement are included in the same framework under the MRF energy minimization. Note that this is the first novelty, given that, in the literature, global algorithms first minimize the energy function and later they perform a refinement process, which does not guarantee to have better energy value. Other contributions are the occlusion handling procedure that is included as a part of the genetic algorithm and the proposed fitness function that is demonstrated to outperform other traditionally used energy functions. Finally, it was implemented in CUDA in order to demonstrate its parallelism and to improve the performance significantly.

4.2 Genetic algorithms in stereo

Genetic algorithms are a class of evolutionary algorithms that have been widely used as an heuristic search for optimization problems in many different applications. They have been previously used in stereo correspondence problems, adding global reasoning to the solutions found. In (Saito and Mori, 1995) a genetic algorithm was used to combine solutions of window-based methods with different window sizes while favouring photo-consistency and smoothness. In order to reduce the size of the problem, the authors also proposed to divide the solution into blocks and find optimal disparity maps block-by-block. The authors in (Han et al., 2001) use a region extraction algorithm for dividing the image. Their fitness function is made of an intensity similarity and a smoothing term between regions. A multi-resolution

approach is proposed in (Gong and Yang, 2001) and (Gong and Yang, 2002), where a quad-tree structure is used for representing each individual. A Markov Random Field (MRF)-based fitness function for global reasoning is used. In (Wang et al., 2003) it is proposed to use the whole disparity map as a representation of genomes with no mutation operation. Recently, (Dai et al., 2008) use an adaptive crossover and mutation while their fitness function does not include any smoothing term. Finally, (Zhang et al., 2009) use a pyramidal propagation stratagem for solution representation and (Nie et al., 2009) implement a stereo correspondence genetic algorithm in GPU for performance enhancement. Genetic algorithms have also been used for matching sparse features, for instance in (Issa et al., 2002) a genetic algorithm is employed to match edges. However, evolutionary algorithms have been rarely used in the literature.

Using genetic algorithms in stereo correspondence has some advantages over other global optimization traditional methods. Firstly, it is a very flexible algorithm, which allows to virtually evaluate any kind of fitness function, although getting close to the optimum is not guaranteed. Secondly, unlike most global or local methods, genetic algorithms can provide various local solutions during the optimization process. Defining the genetic operators and setting up their parameters are one of their main drawbacks.

Our approach uses some of the ideas found in the literature and proposes new crossover and mutation operators. One of the main contributions of this chapter is that a method for occlusion handling is included natively in the genetic algorithm, not only as a final refinement process. The disparity estimation is performed on both right and left images. This permits to calculate an occlusion map for both images, and treat differently the occluded pixels than the common ones. The new crossover proposed permits to make a combination of a large number of pixels at the same time while favouring the child to inherit the best regions of both parents. The new mutation operator radically changes the disparity values of some regions to enable large jumps in the solution space. Finally, another contribution is the fitness function proposed, that really achieves to optimize correctly the number of bad-pixels in the image considering occlusions.

As mentioned before, (Middlebury, 2002) has become one of the main resources for evaluation and comparison of stereo correspondence algorithms. Any of the genetic stereo correspondence methods previously cited neither used the standard set of stereo images, nor they compared their results with the state-of-the-art stereo

algorithms. In this chapter a comparison between several stereo methods is shown and results have been uploaded to the ranking system in (Middlebury, 2002) for future comparison.

4.3 Proposed algorithm

Generally, stereo correspondence has been demonstrated to be an ill-posed, NP-hard problem. Moreover, even considering a quite small size disparity image of four hundred pixels of width, three hundred of height and sixty different disparity labels, the number of different solutions is completely overwhelming. A naive implementation of a genetic algorithm, with random disparity assignments to each pixel, does not perform correctly due to the fact that almost any random disparity image does not make sense as an image. Thus, it is very easy to spend most of the time exploring disparity images which are not even feasible as images. It would be interesting not to search over every disparity distribution, but only search in the group of disparity images that actually make sense. Therefore, due to the huge search space involved in stereo correspondence, properly guiding the genetic algorithm towards feasible disparity maps is fundamental to make the algorithm computationally tractable.

This section explains in more detail genetic algorithm approach for stereo correspondence and refinement, including three main contributions. Firstly, occlusion detection and management is included in the algorithm. This process is done by calculating the dense disparity map for both left and right stereo images and deducing which pixels are necessarily occluded. Secondly, for initializing the stereo process, a combination of adaptive support-weight approach (Yoon and Kweon, 2006) and a census window-cost aggregation scheme is presented. Finally, for setting the fitness function, a new energy smooth term is proposed in the framework of the MRF, which performs substantially better than classic formulations in terms of final accuracy.

In the following subsections each operator of the genetic stereo correspondence algorithm proposed in this chapter is explained in greater detail.

4.3.1 Genome representation

As mentioned in section 4.1, many different genome representations have been used in the literature. The most remarkable ones are the quad-tree and the disparity map

representation. Given that, in the method herein proposed, no multi-resolution is used and the disparity map representation makes it easier to compute the fitness function, the whole disparity map representation has been chosen. An important novelty implemented in this algorithm is to include both left and right disparity images in the genome representation. This permits to estimate the occluded regions on one image based on the disparities calculated on the other. This image, describing which pixels are occluded, is called occlusion map. Treating occluded pixels differently compared to regular ones is a key part in the proposed fitness function, so this is absolutely needed for improving the accuracy of the stereo algorithm as if it was a refinement process. Each genome of the genetic algorithm includes both left and right disparity estimations and occlusion maps.

$$\bar{g} \begin{cases} \bar{f_L} = \{f_{L_1}, f_{L_2}, ..., f_{L_N}\} \\ \bar{f_R} = \{f_{R_1}, f_{R_2}, ..., f_{R_N}\} \\ \bar{O_L} \\ \bar{O_R} \end{cases}, \quad f_i \in \Lambda, \Lambda = \{1, 2, ..., L\} \tag{4.1}$$

where $\bar{g}$ is the genome, $\bar{f_L}$ and $\bar{f_R}$ are the representation of the left and right disparity images respectively, $\bar{O_L}$ and $\bar{O_R}$ are the left and right occlusion maps, f_{L_i} and f_{R_i} are the disparities estimated for pixel i on the left and right disparity images respectively, N the total number of pixels in each image, L the total number of labels and Λ the set of labels representing the set of disparities analyzed.

Both left and right occlusion maps are defined as follows

$$O(p) = \begin{cases} 0 & \textit{if not occluded} \\ 1 & \textit{if occluded} \end{cases} \tag{4.2}$$

where $O(p)$ is the occlusion map and p is the pixel. The calculation of the occlusion maps given the disparity maps will be explained in subsection 4.3.5. An example of a genome representation for the Midelbury's *Tsukuba* is shown in Figure 4.1

4.3.2 Initialization

It has been observed that a complete random initialization of the individuals does not work properly due to the huge search space, so two different approaches have been

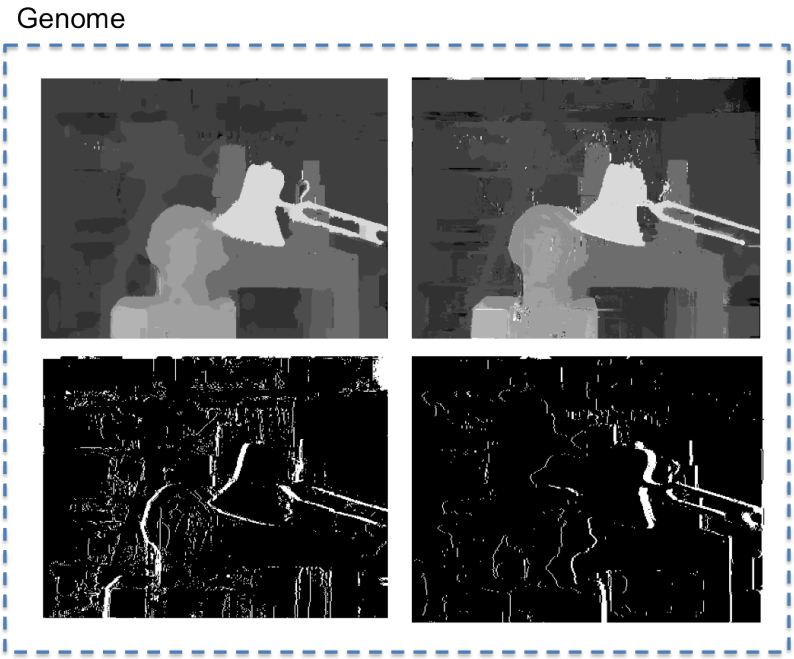

Figure 4.1: Genome representation with left and right disparity and occlusion maps

used in the literature. Some of them use random sampling for their initialization process. The probability of each pixel having a certain disparity value is based on a photo-consistency measurement. For disparities which imply the correspondence of two pixels with very different colors this probability measure should be low, whereas in the opposite case the probability should be high.

Others, used a solution of other local window-based algorithm with a random window size. This approach is similar to the one proposed in (Saito and Mori, 1995) with the main difference that the disparity range is not restricted to the range obtained by the local methods.

For the initialization process, two different window-based algorithms with different window sizes have been used, the adaptive support-weight approach (Yoon and Kweon, 2006) with random parameters, and the census based with window-cost aggregation. For the census transform, a constant window size of 9x7 was used as suggested in (Mei et al., 2011). During the initialization process, each pixel is sampled with a probability proportional to the times it has appeared in the local window-based algorithms. It is important to notice that either algorithm alone performs fine in occluded, discontinuities or untextured areas.

Other stereo algorithms could be used for the initialization process, such as classic

Figure 4.2: Adaptive weight local initialization for *Tsukuba*

global stereo algorithms using graph-cuts or belief propagation. They could be solving a different energy function and they show their benefit in places such as untextured areas. If these algorithms are used for initialization, notice that they could cause the genetic algorithm to get stuck in local minima quite fast. It is recommended to perform an individual analysis for selecting which initialization algorithms are more suitable for each application. As shown in section 4.6, we found useful to add global algorithms in the initialization process in the Middlebury dataset, while harmful in the KITTI dataset.

This initialization process is very flexible because it permits to add suggested disparities for pixels using any other stereo method and afterwards include them in the genetic algorithm. In this sense the genetic algorithm behaves like a refinement process merging the solutions obtained through other methods. Examples of genome initialization using adaptive weight and census are shown in Figure 4.2 and Figure 4.3 respectively. It can be seen that each algorithm performs better in different regions, so the main aim is to keep the best parts of each algorithm based on a MRF energy function.

4.3.3 Fitness function

As it is widely known, the stereo correspondence optimization problem can be formulated as a MRF. The fitness function is responsible for evaluating the energy function given an individual. The less energy associated with the individual, the more likely that it will survive in the next generation. As the aim of the algorithm is to get the best left disparity estimation, note that the fitness function will evaluate only

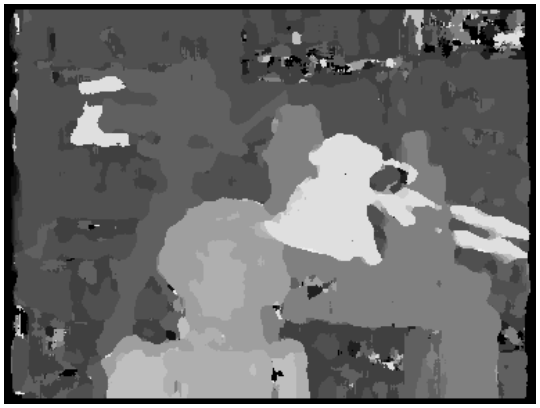

Figure 4.3: Census local initialization *Tsukuba*

the energy on the left disparity image. One possible option for the fitness function, very similar to the classical MRF approach, is given by the following equations

$$E_{classic}\left(\bar{g}\right) = E_{data-classic}\left(\bar{f}_L\right) + E_{smooth-classic}\left(\bar{f}_L\right) \tag{4.3}$$

$$E_{data-classic}\left(\bar{f}_L\right) = \sum_{p \in H} |I_L(x(p), y(p)) - I_R(x(p) - f_p, y(p)))| \tag{4.4}$$

$$E_{smooth-classic}\left(\bar{f}_L\right) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(f_p, f_q) \tag{4.5}$$

where g is a certain individual, f_L is the left disparity image, I_l and I_r stand for the left and right stereo pair, H is the set of all pixels in the left image, $x(p)$ and $y(p)$ are the image coordinates of pixel p, and $V_{\{p,q\}}$ is a smoothing function favouring the neighboring pixels having the same disparity.

This energy configuration has been widely used in the literature by global stereo algorithms. One reason for this energy function to be so common is because it can be optimized by heuristic algorithms such as graph-cuts (Kolmogorov and Zabin, 2004). As stated in (Kolmogorov and Zabin, 2004), not any function can be used for this purpose.

The classic energy function might present some problems in the disparity estimation along the discontinuities and the occluded areas. It also favours the fronto-parallel surfaces, which can be problematic in environments with predominant slanted planes. Due to the flexible nature of the genetic algorithms, other functions more complicated can be used as fitness functions. Herein it is proposed an energy function

that considers discontinuities and occlusions for the energy evaluation:

$$E\left(\bar{g}\right) = E_{data}\left(\bar{f_L}\right) + E_{smooth}\left(\bar{f_L}\right) \tag{4.6}$$

$$E_{data}\left(\bar{f_L}\right) = \sum_{p \in H} \min\left(|I_L(x(p), y(p)) - I_R(x(p) - f_p, y(p)))| \cdot (1 - O(p)) + \lambda_d \cdot O(p), \lambda_{dt}\right) \tag{4.7}$$

The main modification in the E_{data} term is that occluded pixels, as they do not have a correspondent pixel on the right image, contribute to the energy with a constant value λ_d. This characteristic enables low energy configurations where the occluded pixels are matched correctly. Besides, it has been added a threshold parameter λ_{dt} in order to determine a maximum value of the term.

$$E_{smooth-fronto}\left(\bar{f_L}\right) = \sum_{\{p,q\} \in N} \min\left(\frac{max(\lambda_s, \gamma_s - |I_L(p) - I_L(q)|)}{\varphi_s}|f_p - f_q|, \lambda_{st}\right) \tag{4.8}$$

where λ_s, γ_s and φ_s are constant parameters for every pixel. λ_t is a truncation parameter for clamping the value of the smooth term.

This new smooth function establishes a relation between the color consistency of the neighbors and the associated weight of their disparity difference. For neighboring pixels that are very different in color, low weight is assigned to their disparity difference, while neighboring pixels that are very similar are forced to have the same disparity.

As in the classic smooth term, Equation 4.8 favour fronto-parallel surfaces. The formulation for general planar surfaces would have the following shape

$$E_{smooth-planar}\left(\bar{f_L}\right) = \sum_{\{p,q,r\} \in N} \min\left(\frac{max(\lambda_s, \gamma_s - |I_L(p) - I_L(q)|)}{\varphi_s}|f_p - 2f_q + f_r|, \lambda_t\right) \tag{4.9}$$

where the smooth function penalizes changes in the normal values of the surface, not changes in disparity.

At this point it is very important to emphasize that in any case the genetic algorithm, neither using the classic energy function nor the proposed one, is guaranteed to find the optimum energy configuration. Even more, how close one can get to the optimum will depend on the fitness function, the crossover and mutator operators, their parameters, the population size, etc. Probably, in any case the genetic algorithm will get close to the optimum, but the experiments carried out and shown in section 4.6, suggest that the proposed energy function is more adequate than the classic one for both discontinuity and occlusion management.

4.3.4 Crossover

It is difficult to define an efficient crossover between disparity images that explores new solutions and at the same time that succeeds on improving the fitness of the final individuals in log term. As suggested in (Wang et al., 2003), traditionally used crossovers operators, which are totally random, such as one-point crossover or uniform crossover generally will not transmit good genetic material from parents to sons.

On first place, our algorithm used as crossover an operator very similar to the uniform crossover. The operator was used in the left and right disparity images only, leaving both occlusion maps out. Given that the objective is to merge two different disparity images, which are two dimensional data, several random crossover points for x and y dimensions were applied. For each disparity image, a random block size was calculated representing a region of each disparity image f_L and f_R. Then, a random assignation of each parent block to the sons was performed with different variations for the mixing ratio. The crossover is graphically shown in Figure 4.4.

While this stochastic approach to the crossover operation is inherent to the genetic algorithms, some tests with a deterministic crossover were also carried out. A new crossover was defined that, instead of randomly assigning the blocks to each child with a mixing ratio, e.g. of 0.5, first it evaluates the fitness function on each parent block and then combines the blocks with the best fitness function on the same child. In this sense, this approach contradicts the stochastic nature of the genetic algorithm and might involve getting stuck in local minima. However, after testing both approaches, the deterministic crossover achieved a considerably better fitness function than the stochastic one. The results that justifies these approach are shown in subsection 4.6.1. Therefore this approach was used for the final tests.

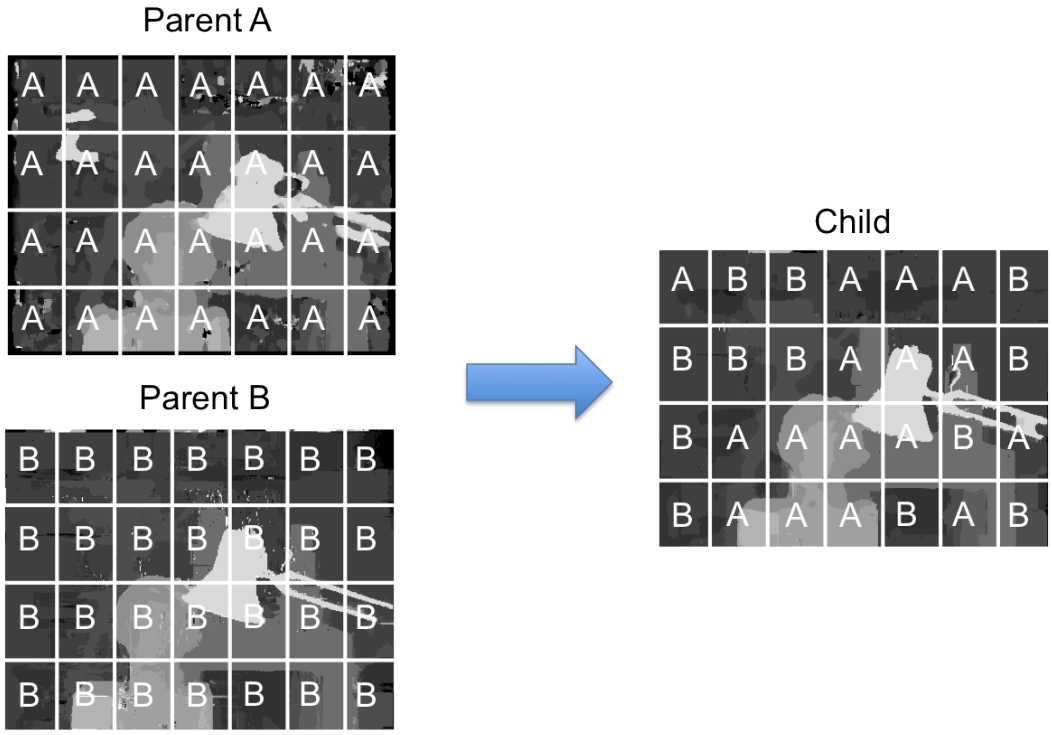

Figure 4.4: Crossover operator for *Tsukuba*

4.3.5 Mutation

In the genetic algorithm, the mutation operation has a fundamental role. Several mutation operations are herein proposed:

- Initialize again with other stereo methods
- Apply a median filter
- Apply the erode morphological operation
- Apply a dilation operation
- Apply a open operation
- Apply a close operation
- Occlusion handling

These operations are not exclusive and each one has its own probability associated. Firstly, one possible mutation operation is to initialize again some pixels of one of the left or right images following the steps explained in subsection 4.3.2. That is, the disparity map is randomly changed with a probability proportional to those suggested by local methods. This mutation operation may happen with a probability P_{Mini}.

Secondly, a median filter operation with a random window size is also performed as a mutation function. It is not any novelty, but sometimes it is effective for managing some sparse outliers. This median filter operation is performed with a probability P_{Mmed}. Several types of morphological operations may also happen. These are erode, dilate, open and close, each one having a probability of P_{Mmorph}.

Finally, an occlusion detection and handling is also included as a mutation with a probability P_{Mocc}. This step is the most important mutation operation and, combined with the new fitness function explained in subsection 4.3.3, it has a great impact on the final accuracy of the algorithm as it will be demonstrated on the section 4.6. This mutation is a two step operation: an occlusion detection followed by an occlusion management.

4.3.5.1 Occlusion estimation

Some top-performance algorithms like (Mei et al., 2011) use a right-left consistency check as an outlier detection and then classify them into occluded and mismatches. Our occlusion classification algorithm is based on a different approach. Given that both left and right disparity images are being estimated by our algorithm, we can use the right image disparities to estimate which pixels cannot have possible matches on the left one and vice-versa. The following operations are defined for calculating left occlusion map:

$$O_L(p) = \begin{cases} 0 & \exists i / \begin{pmatrix} x(i) + \bar{g_R}(i) \\ y(i) \end{pmatrix} = \begin{pmatrix} x(p) \\ y(p) \end{pmatrix} \quad p, i \in P \\ 1 & \text{other case} \end{cases} \tag{4.10}$$

being O_L the left occlusion map, $x(p)$ and $y(p)$ the x and y coordinates of point p respectively and P the set of disparity image points. In other words, a warp from the right image to the left it is performed using the right disparities estimated.

The points on the left image that have no wrapping correspondent, are considered occluded, while the rest are considered non-occluded.

Similarly, an expression for the right occlusion map for the right image is:

$$O_R(p) = \begin{cases} 0 & \exists i / \begin{pmatrix} x(i) - \bar{g_L}(i) \\ y(i) \end{pmatrix} = \begin{pmatrix} x(p) \\ y(p) \end{pmatrix} \quad p, i \in P \\ 1 & \text{other case} \end{cases} \tag{4.11}$$

being O_R the right occlusion map. An example of a left occlusion map obtained for the *Tsukuba* stereo pair is shown Figure 4.5.

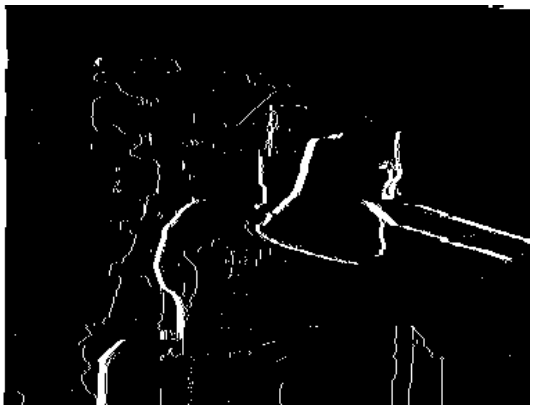

Figure 4.5: Left occlusion map for *Tsukuba* stereo pair

4.3.5.2 Occlusion management

Once the occlusion maps are calculated for both images, a very simple but effective occlusion management procedure is applied. Other top-performance algorithms use segmentation procedures or histogram-based algorithms for the occlusion management, but here it was meant to keep it as simple as possible.

Two quite similar algorithms were implemented. The first one, used the color information surrounding the occluded pixels to perform the calculations. Some testing proved this algorithm to be more accurate but also more computationally expensive. It is based on an iterative process where each occluded pixel is assigned the disparity value of the most photo-consistent non-occluded neighbor (in the left disparity map case it is run from left to right). If no non-occluded neighbors exist, it maintains its occluded status for the next iteration. Special status is assigned

to the occluded pixels whose $x(p)$ coordinate is less than the number of disparities analyzed. In this case the filling is made in the opposite direction (from right to left if it is the left disparity map) and from bottom to up. The iteration is finished when no occluded pixels are left on the left occluded map. Throughout this document this algorithm will be named **photometric iterative occlusion filling** algorithm.

For the right image it is similarly done but vice-versa (right to left for common pixels and left to right for pixel whose $x(p)$ is at a distance of the number of disparities analyzed from the right image border). This simple algorithm demonstrates to be effective in section 4.6.

In order to reduce the computational complexity, a faster and parallelizable algorithm for occlusion filling was implemented. This greedy algorithm iterates for every occluded pixel, assigning the disparity value of the closest horizontal non-occluded pixel, searching in the left direction if it is the left disparity map and in the right direction if it is the right disparity map. As happened in the previous algorithm, the search direction for the occluded pixels caused by image displacement (in the left-most or in the right-most area) is reverted. This algorithm will be called **horizontal fast occlusion filling**.

4.3.6 Selection

The selection method used for this genetic algorithm was a very standard one. It was used the famous roulette wheel selector preceded by a normalization of the fitness of each individual. The procedure is *elitist*, which means that the best individual of each generation is always maintained to the next generation.

4.4 Parallelization using CUDA

Recently, CUDA has become one of the most popular tools for increasing the efficiency of parallel algorithms thanks to the computational capacity of the *Graphics Processing Unit* (GPU) compared to serial CPU programs. Traditionally, GPUs appeared in the computer market as hardware specialized on rendering tasks and more specifically for improving the gaming experience. Given that most steps of the rendering pipeline are highly parallel, they rapidly evolved to hardware capable of efficiently running parallel algorithms in contrast to the CPUs. In the last few

years, the flexibilization of the GPU hardware and tools enabled the use of these parallel processing units for research purposes. CUDA consists of a compiler and a group of tools developed by Nvidia and since 2007 it can be used on the latest GPUs manufactured by this company. Many algorithms and libraries are currently being ported from their traditional serial versions to new parallel ones implemented generally in CUDA language. Some of them are *Thrust*, a parallel version of the C++ standard library, *cuBlas* and *cuSparse*, GPU-accelerated versions of the popular Basic Linear Algebra Subroutines (BLAS) and SPARSE matrix library, Matlab GPU computing support, and algorithms described in (Vineet and Narayanan, 2008) (Mei et al., 2011).

Note that the only reason for using CUDA is to boost the programs' performance. Generally, most parallel libraries aforementioned claim to obtain a boost from 40x to 100x using GPU compared to the CPU version. However, this improvement in the computational time, due to the increase of the computational capacity, highly depends on the nature of the algorithm itself. The following questions have to be asked in order to evaluate if the algorithm should be implemented in GPU:

- Which parts of the program are actually spending most of the computational time?
- Are these parts naturally parallel or is there any parallel version of them?
- Can I use an existing library or should I program it from scratch?
- Is it necessary to continuously transfer data from the GPU to the CPU or vice-versa?
- By how much do we expect the computational time to be reduced?

These are some fundamental questions that are part of the APOD methodology (Analyze, parallelize, optimize, deploy) that will help you understand which are the expectations you must have if your program is parallelized in CUDA. The following section will briefly cover the basics of the CUDA programming model in order to understand the rest of the section.

4.4.1 Genetics model in CUDA

After analyzing the performance of the serial version of the genetic algorithm, it is easy to conclude that the functions that are more computationally demanding are the genetic operators and not the genetic algorithm itself. This result is straightforward because each genome includes a lot of data and information inside (whole four images: two disparity maps and two occlusion maps). For example, each genome evaluation implies evaluating the energy function for each pixel and neighborhood individually. Besides, each genome operator is naturally parallel, which suggests that implementing these operators in CUDA will have a dramatic impact on the genetic algorithm performance.

In Figure 4.6 it is shown where is computed each genetic operation.

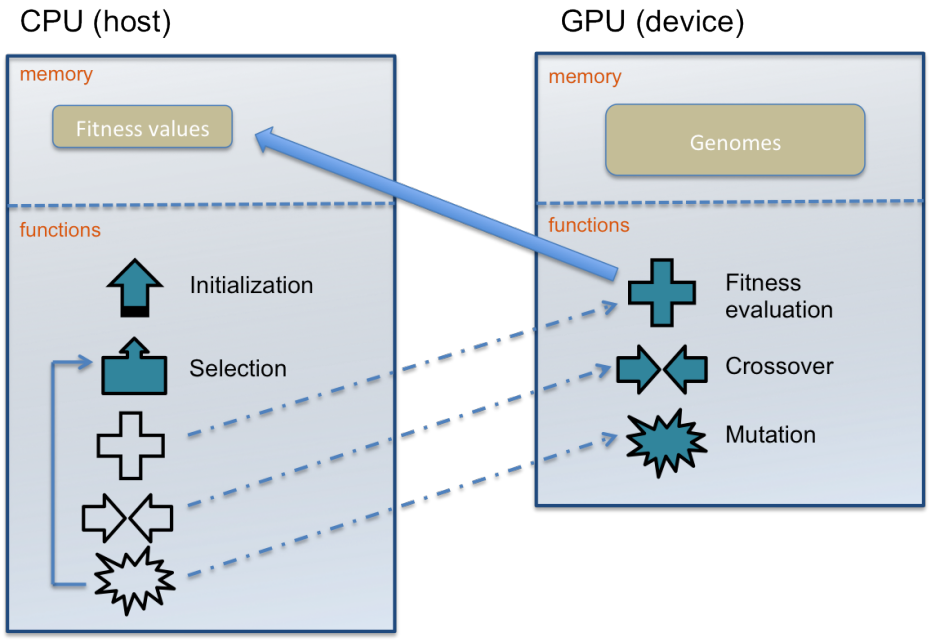

Figure 4.6: Assignment of genetic operators to GPU and CPU

The left side of Figure 4.6 shows where the data is allocated and which functions are implemented and executed in the CPU. The fitness values are stored in the CPU because they are needed for the selection operator in order to decide which individuals of the actual population will survive to the next one. The right side of the diagram represent which information is stored and which functions are evaluated in the GPU. All the genomes are stored in the GPU in order to enable fast access to the data from the functions evaluated in the device. The only memory transaction between the

CPU and GPU needed is the copy of the fitness value of each individual from device to host and is represented by the big blue arrow from the *fitness evaluation* function icon to the fitnes value memory in the CPU. Remember that these device-host and vice-versa transactions are very costly and must be minimized for achieving the best performance.

The genetic algorithm has been implemented in the CPU using the GAlib library. For the image processing and allocation it has been used the OpenCV library, specifically the GPU module, which facilitates the memory allocation and transaction and has quite a lot processing algorithms built-in in the GPU already. Finally, evaluation, crossover and mutation operators have been implemented in CUDA in several kernels. The next sections describe in detail the strategy used for implementing efficiently each operator in CUDA language.

4.4.1.1 CUDA evaluation kernel

Although the title may suggest that the evaluation of a genome is carried out just by one kernel, the reality is that it is a process composed by three steps. The first two are solved using a single kernel each, while the third required two kernels. The first two steps could be executed in parallel by two different CUDA *streams* but the last one must be executed after the firsts have finished. This parallel capability has not been implemented and all four kernels have been programmed to run in the same *stream*.

The first step in the evaluation process is the **data** term evaluation of the energy function. The result is one value for each pixel and its calculation is independent from the values of the neighboring pixels. Thus, the relation is one to one and its parallel implementation is very efficient and straight forward. This type of operation is also called MAP, and it has been implemented using one thread per pixel in the disparity image. A simple diagram of MAP is shown in Figure 4.7.

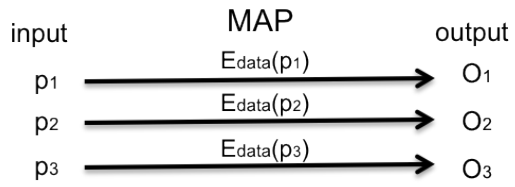

Figure 4.7: Parallel MAP operation

As explained in subsection 4.3.3, the data term only depends on the values of the left and right stereo images and on the disparity image evaluated. Left and right stereo images have been allocated in the device as *2D textures*, which are very efficient for interpolation. Note that in this case, using *shared* memory does not make much sense because the number of memory accesses needed per thread would not be minimized. The result is saved in a floating point structure of the same size as the original image, and here will be referred as *mem_ data*.

The second step is the evaluation of the **smoothing** term. If one thread per pixel is used, it requires to access its own disparity value and the neighboring disparities. This operation can be considered a type of *Stencil* operation, in which many reads are needed as input while only one write is performed. An illustrative example is shown in Figure 4.8

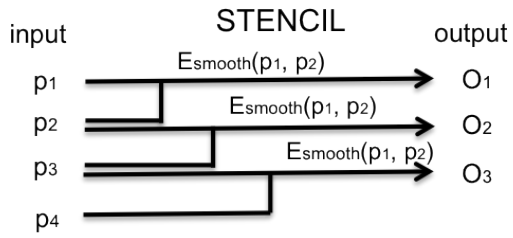

Figure 4.8: Parallel STENCIL operation

In order to maximize the performance, *shared memory* is used for initially loading all the disparities in a block and then using that shared memory in all the threads of the same block. Remember that the access to *shared* memory is much faster than the access to *global* memory. Each thread is related to each pixel in the disparity image and it is in charge of evaluating the smoothing function that relates itself with the right and bottom neighbors. As happened in the data kernel, the result is saved into a floating point vector with the same size of the disparity image and here will be called *mem_ smooth*.

The third and last step is composed of two kernels, one executed after the other. It is in charge of performing the **summation** over all pixels of the *mem_ data* and *mem_ smooth* structures calculated in the two first steps. This type of summation is an operation also known as *Reduce*. Although at first glance this operation might seem difficult to parallelize, actually it is fairly simple. Figure 4.9 shows the two step reduction implemented. Besides, in this step sums of *mem_ data* and *mem_ smooth* individually for each pixel and saving it in *mem_ total* in order to facilitate the crossover task explained in subsubsection 4.4.1.2.

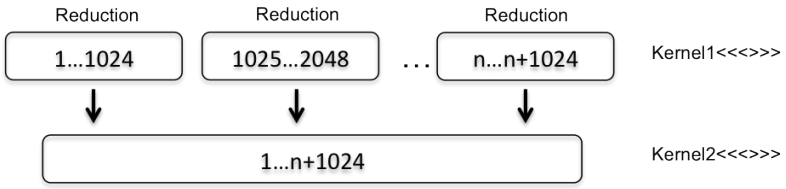

Figure 4.9: Reduction operation performed in two kernels

For enhancing the performance of the *Reduce* operation, the data has been divided in groups of 1024 addends, each being processed by a CUDA block. All the data in each group is loaded in *shared* memory to improve its read and write speed. The first kernel performs the summation over each group, obtaining one result per group. Finally, the last kernel performs the last summation over all the results of the previous kernel, and obtains the final value for the fitness function.

Finally, an asynchronous memory copy is performed from device to host in order to copy the final fitness value calculated for that genome. This process is asynchronous because the memory copy will be performed at the same time as other kernels are executed in other streams, instead of waiting until the memory copy has finished.

Note that in spite of running all the evaluation kernels in the same *stream*, different individuals are able to run their evaluation in different streams, which enables copying memory from device to host at the same time as other kernels and a higher level of parallel exploitation.

The evaluation process is performed over the left and right disparity map, but for the right one, it is not necessary to perform the final *Reduction* and memory copy. This optimization can be achieved because the fitness function of each genome is just the fitness function of its left disparity image.

4.4.1.2 CUDA crossover kernel

As explained in subsection 4.3.4, the crossover operation consists of three steps:

- Divide the two disparity maps into blocks. In this case, the number of pixel of each block will not be greater than 1024.
- Sum up the *mem_total* for each pixel inside the blocks, compare them by pairs (one for each parent) and keep the block with the best fitness function.

- Copy the best block to the children.

The limit of 1024 pixels per block is related to the maximum number of threads per CUDA block available by the GPU. All the pixels in a block must be part of the same CUDA block because the summation can be performed using *shared* memory, which is much more efficient than *global* memory. Therefore, the first step is realized by the CPU, and the following two are done by the GPU, the first one as a *Reduction* operation very similar to the one in subsubsection 4.4.1.1 and the second one as a very simple copy operation as a MAP.

4.4.1.3 CUDA occlusion handling kernel

Occlusion handling encompasses two different tasks: occlusion estimation and occlusion management. The **occlusion estimation** is calculated through an image *warping*, where each pixel of the other disparity image is displaced a number of pixels equal to its disparity level. Pixels left without any assignation are considered to be occluded pixels. Thus, each pixel operation is independent from the rest, but several threads can output their result to the same piece of memory. This operation is also known as *Scatter* and can be solved using, for example, atomic operations. In our case it is not necessary because the function aims only to output a boolean value, more precisely a zero to indicate that the pixel is not occluded.

The second task is the **occlusion management**, where the main objective is to re-estimate the disparity value for the pixels that where labeled as occluded. For this parallel implementation the *horizontal fast occlusion filling* algorithm explained in subsubsection 4.3.5.2 was used. Given that a horizontal search for the closest non-occluded pixel has to be performed, the occlusion information was loaded in *shared* memory, being each block responsible for each independent scan-line. Each thread is in charge of estimating the new depth for each occluded pixel. Figure 4.10 shows the per-thread operations and the memory accesses incurred.

4.5 Why using genetics in stereo makes sense

As mentioned in section 4.1, genetic algorithms can virtually be used in any type of optimization problem. However, their performance will depend on the nature of the problem. In this section, a brief explanation of the particularities of genetic

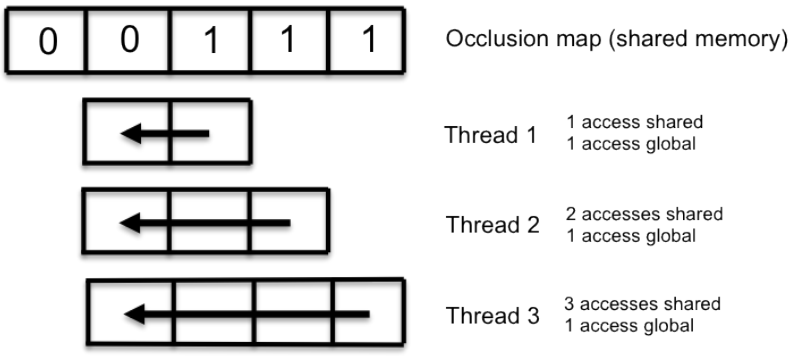

Figure 4.10: Horizontal fast occlusion filling implementation example

algorithms applied to stereo correspondence problem and how it relates to other state of the art methods is presented.

As a brief reminder, lets analyze the general structure of the most important graph-cuts algorithms in the literature. Both *swap* and *alpha-expansion* (Boykov et al., 2001) start by a random disparity solution and through several iterations, they perform some kind of *movement* which is guaranteed to enhance the cost function. *Swap* and *alpha-expansion* differ only on the type of movement applied throughout the iterations. These movements are usually optimal in a particular sense, e.g. the optimal swap of pixels labeled with two given α and β labels. Notice that although these movements are optimal, the global optimization algorithm does not guarantee to find the optimal solution of the energy function. This structure is represented in Figure 4.11

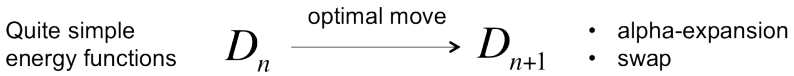

Figure 4.11: Graph cuts general optimization structure

Basically, the algorithm starts by a random solution and new solutions, which are optimal in a sense, are proposed iteratively. The main problem of this approach is that guaranteeing this optimality imposes some restrictions on the energy functions, which only permits to use these algorithms with simple standard energy functions.

Recently, new algorithms with different approaches have been proposed in the literature. In (Woodford et al., 2009), another structure for the optimization algorithm is proposed to enable the use of more complex energy functions, such as functions with second order priors. As in the *swap* and *alpha-expansion* cases, the algorithm proposes new disparity estimations based on previous disparities. The

main difference is that, given that no optimal proposes can be achieved due to the energy functions nature, suboptimal proposals are created. This is represented in Figure 4.12.

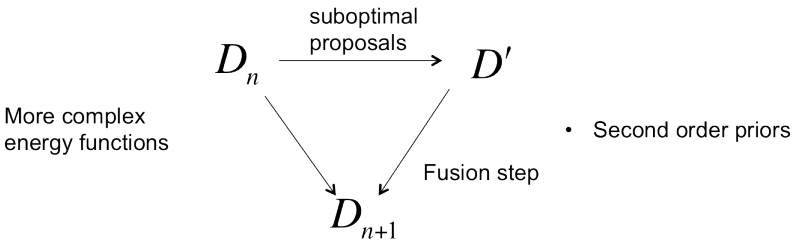

Figure 4.12: Second order priors optimization structure

In (Woodford et al., 2009), the proposals are followed by a fusion step, which tries to merge the original disparity map and the proposed one in order to obtain the best of both solutions. This fusion step is not guaranteed to obtain a better solution than the original ones. The main advantage of this algorithm is that, although the movements are not optimal, it is able to optimize complex energy functions which can represent more realistic priors.

Thus, now the question is if the genetic algorithm optimization structure resembles to any of the aforementioned algorithms. Similarly, the *proposal* function and the *fusion* function have their equivalent in the genetic operators in the genetic algorithm. For instance, the mutation operation can be considered a proposal operation because a modification is performed over the genome based on its original configuration and a new original solution is added. The initialization operator is also a *proposal* function that enables the introduction of solutions achieved by other stereo algorithms. Finally, the fitness, selection and crossover operators work like a big fusion step where several proposals are combined and only the best ones survive to the next generation. Figure 4.13 shows this classification.

Now, the main problem is to define which operators are best suited for this particular optimization problem. The main characteristics of the energy minimization stereo correspondence problem are:

- The search space is huge
- Only a small subspace of the huge search space is actually possible as a image
- Evaluating the fitness of each genome is very expensive

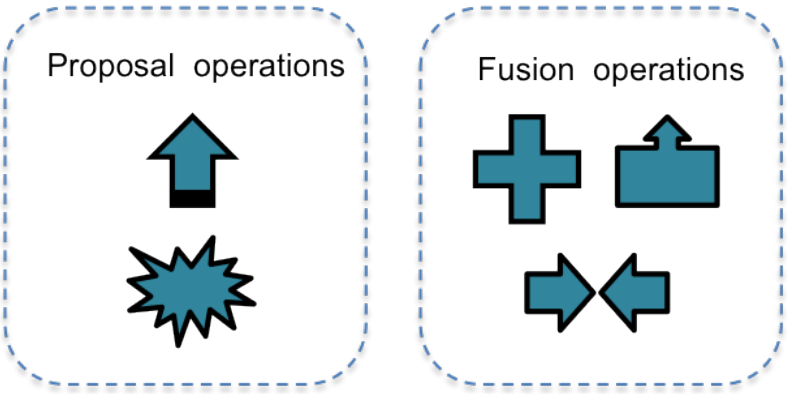

Figure 4.13: Genetics algorithm optimization structure

The first two characteristics explain why a guided search is preferred. Although having the risk of getting stuck in local minima, guiding the search is the only way to make the search tractable. All the possible mutation and initialization operations were designed to make disparity proposals that actually fall in this *"feasible as an image"* search space. The last characteristic implies that probably the algorithm will not be able to manage big population sizes or lots of generations. Therefore, it justifies why a greedy crossover was implemented in order to obtain good results early and why an optimized implementation in CUDA was necessary.

4.6 Experimental results

In this section, accuracy analysis for the Middlebury and KITTI datasets will be presented. Finally, a speed improvement analysis of the CUDA implementation versus the CPU implementation will be detailed.

4.6.1 Accuracy in Middlebury dataset

The genetic algorithm proposed has been applied to solve the Middlebury standard stereo dataset in (Middlebury, 2002). The set consists of four images: Tsukuba, Venus, Teddy, and Cones. Quite a lot of parameters for tweaking have appeared along this chapter. The parameters related with the new energy function proposed are shown in Table 4.1, while the parameters related with the genetic algorithm are shown in Table 4.2. For the local methods used during the initialization process of

the genetic algorithm, window sizes between 3 and 45 have been used and random values for the adaptive-weight parameters. The energy function used was the new proposed fronto-parallel smooth term described in Equation 4.8. All the test-cases were run using the same parameters.

λ_d	λ_{dt}	λ_s	γ_s	φ_s	λ_{st}
10.0	255	50.0	2.0	10.0	$ndisp/2$

Table 4.1: Parameters for the new energy function

Population	Generations	P_{cross}	P_{Mini}	P_{Mmed}	P_{Mmorph}	P_{Mocc}
50.0	1000	0.9	0.02	0.016	0.016	0.9

Table 4.2: Parameters for the genetic algorithm

The resulting left disparity images with their BadPixel image representation are shown in Figure 4.14 and Figure 4.15. As it can be seen, all the disparity images look pretty similar to the true disparities. Looking to the bad-pixel images, it is clear that the Tsukuba and Venus images obtain the more accurate results. The algorithms perform quite well all along the discontinuity regions, while there are other regions such as the top-right corner in the Tsukuba image or the untextured region around the teddy bear in the Teddy image where the algorithm fails substantially. This can be attributed to the fact that the local algorithms used for the initialization of the algorithm also fail in these untextured regions, therefore the genetic algorithm is unable to generate individuals with proper disparities on those regions.

All four images were uploaded and evaluated using the Middlebury stereo webpage via their submit procedure. The results are shown in Figure 4.16 together with other near-rank algorithms. The algorithm is situated on the middle area of the rank with an average rank of 38.5 and an average percentage of bad-pixels of 5.81. This is a considerable high improvement over, e.g. the adaptive-weight algorithm used for its initialization step which has an average rank of 61.4 and an average percentage of bad-pixels of 6.67. Moreover, the proposed genetic algorithm has achieved the best rank in the discontinuity areas of the Tsukuba image. Comparing the proposed algorithm to the best reported one (Mei et al., 2011), its average percentage of bad-pixels is 3.97, which implies that our algorithm performs 1.84% worse. This can be attributed mainly to the texture-less regions talked about before where the local methods fail considerably.

As it was detailed in subsection 4.3.3, our genetic algorithm uses a new energy

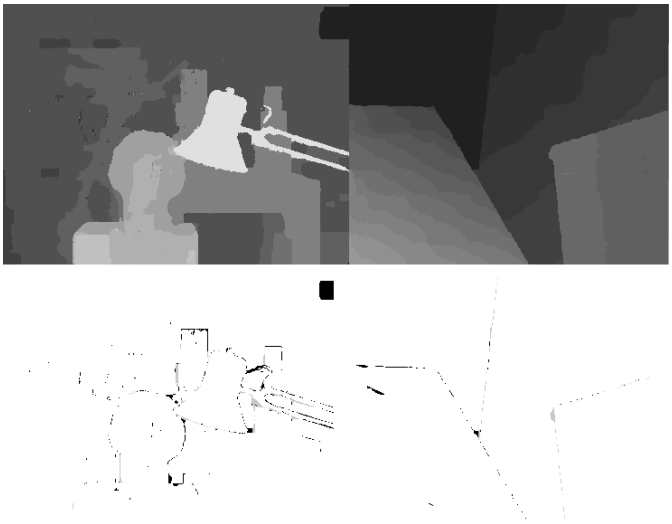

Figure 4.14: Tsukuba and Venus results. Disparity images (first row) and bad-pixels (second row)

function that accounts for disparity discontinuities and occlusions compared to the classic energy function used in the classic MRF minimization framework. Some tests were carried out using exactly the same genetic algorithm but changing the energy function for the classic one. The truncated linear function was used for the smoothing function with a cost of 1.0 and a truncation value of 10.0.

The results were uploaded to the Middlebury stereo web-page, following the same steps as in the previous case. The results are shown in Figure 4.17. The average percent of bad-pixels increases from 5.81 to 8.56, which is near a 3% more bad-pixel error if the classic energy function is used.

The evolution of each energy function during the optimization process compared to the bad-pixels error measurement is shown in Figure 4.18 and Figure 4.19. Both energy functions are compared, being the blue one the new energy function proposed and the red one the classic energy function. Each figure represents the evolution of different populations: the first one using the Tsukuba stereo pair and the other one using Venus.

The first row on both images shows the evolution of the whole energy function which is minimized during the first 500 generations. Both algorithms in both tests follow a similar descendant curve. However, it shall not be compared both algorithms in terms of minimum energy achieved given that different functions and parameters are used.

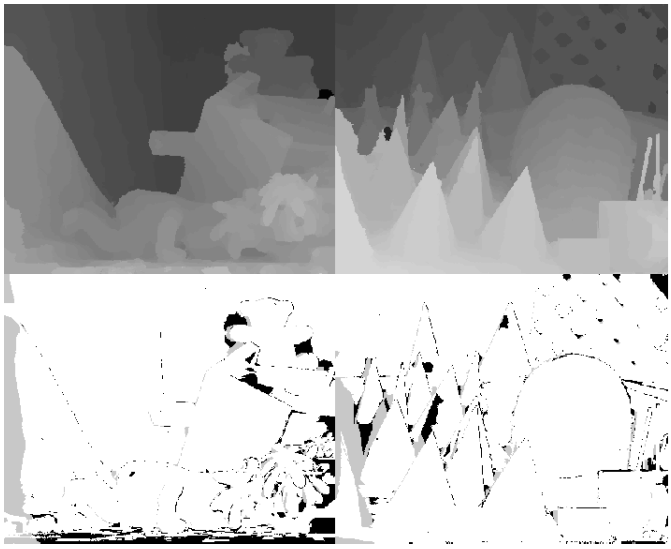

Figure 4.15: Teddy and Cones results. Disparity images (first row) and bad-pixels (second row)

Error Threshold = 1 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc									
Algorithm	Avg. Rank	Tsukuba ground truth			Venus ground truth			Teddy ground truth			Cones ground truth			Average Percent Bad Pixels			
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc				
PlaneFitBP [32]	37.6	0.97 12	1.83 36	5.26 13	0.17 16	0.51 34	1.71 9	6.65 46	12.1 51	14.7 24	4.17 72	10.7 72	10.6 66	5.78			
SymBP+occ [7]	37.7	0.97 10	1.75 30	5.09 12	0.16 13	0.33 15	2.19 22	6.47 40	10.7 27	17.0 56	4.79 82	10.7 74	10.9 71	5.92			
YOUR METHOD	38.5	1.26 29	1.76 31	**4.31** 1	0.20 21	0.42 25	2.03 18	7.52 65	12.3 54	18.1 71	3.94 65	8.59 29	9.32 53	5.81			
BSM [124]	38.8	3.08 97	3.38 79	7.80 60	0.26 38	0.70 50	2.40 25	5.74 24	8.95 17	14.8 27	2.34 5	8.79 37	6.80 6	5.42			
AdaptLocalSeg [125]	39.8	1.33 38	1.82 34	7.19 49	0.32 46	0.79 54	4.50 66	5.32 19	11.9 47	14.5 23	2.73 17	9.69 60	7.91 24	5.67			

Figure 4.16: Middlebury rank position for the genome algorithm

The second row shows the evolution of the bad-pixels measurement of the best individual for each population. These charts were selected because they show empirically how the real disparity error evolves when each energy function is minimized. In both Tsukuba and Venus tests it is shown that a reduction of the classic energy function not always implies a reduction of the bad-pixels. Actually, in the Tsukuba case produces some kind of unstable behavior and in the Venus case it produces an error increase. Meanwhile, using the energy function herein proposed, it is obtained a much more stable behavior and a much better final error for all the tests carried out.

However, it is important to notice that it cannot be stated that the energy function proposed better represents the real disparity images, i.e the true disparity images obtain lower energy values than others. Genetic algorithms work fine for finding good

Error Threshold = 1 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc			
Algorithm	**Avg. Rank**	**Tsukuba** ground truth nonocc	all	disc	**Venus** ground truth nonocc	all	disc	**Teddy** ground truth nonocc	all	disc	**Cones** ground truth nonocc	all	disc	Average Percent Bad Pixels
AdaptPolygon [43]	85.7	2.29 81	2.88 74	8.94 69	0.80 78	1.11 71	3.41 55	10.5 104	15.9 100	21.3 100	6.13 102	13.2 99	13.3 95	8.32
RealtimeVar [72]	86.2	3.33 102	5.48 109	16.8 116	1.15 91	2.35 97	12.8 101	6.18 32	13.1 60	17.3 65	4.66 79	11.7 84	13.7 98	9.05
YOUR METHOD	87.1	2.15 75	4.02 90	9.23 72	0.88 80	1.72 88	9.02 90	8.84 91	15.1 90	22.4 103	5.52 99	11.2 79	12.6 90	8.56
ConvexTV [46]	87.3	3.61 105	5.72 112	18.0 118	1.16 92	2.50 100	12.4 99	6.10 29	15.7 97	16.8 53	3.88 60	14.4 105	11.5 77	9.30
GenModel [20]	90.3	2.57 86	4.74 100	13.0 97	1.72 104	3.08 104	16.9 109	6.86 52	15.0 88	19.2 82	4.64 78	14.9 109	11.4 75	9.50

Figure 4.17: Middlebury rank position for the genome algorithm using the classic energy function

approximations to real optimum values only when all the genetic operators are well set. It cannot be guaranteed that the genetic algorithm will perform better using the proposed energy function for any genetic configuration. Neither can be guaranteed that the optimum in one case has less error than the optimum in the other case. However, this trend has appeared in all the tests carried out.

The proposed genetic algorithm has demonstrated that using these kind of optimization techniques can obtain good results in the stereo correspondence problem. Given an initial set of approximate disparities during the initialization process, it is able to substantially improve the final disparity error. Moreover, the genetic algorithms makes it easier to try other energy functions, which have a great impact on the result.

Finally, in order to justify the deterministic crossover final implementation, some tests with different mixing factors in the Cones stereo pair were carried out. An implementation with a mixing factor of 1.0 represents a deterministic implementation where all good blocks are merged in the same individual while a mixing factor of 0.5 represents a complete random assignation. The analysis has been carried out in terms of the bad-pixels (Figure 4.20) and the energy evolution(Figure 4.21). Both charts show that randomizing the crossover with mixing factors closer to 0.5 implies a poorer performance in both bad-pixels and energy achieved my the algorithm.

4.6.2 Accuracy in KITTI dataset

The genetic algorithm was also tested using the KITTI dataset, which is considered a more appropriate test for real applications. Given that the dataset is formed by high resolution stereo pairs, the images in the initialization step of the genetic

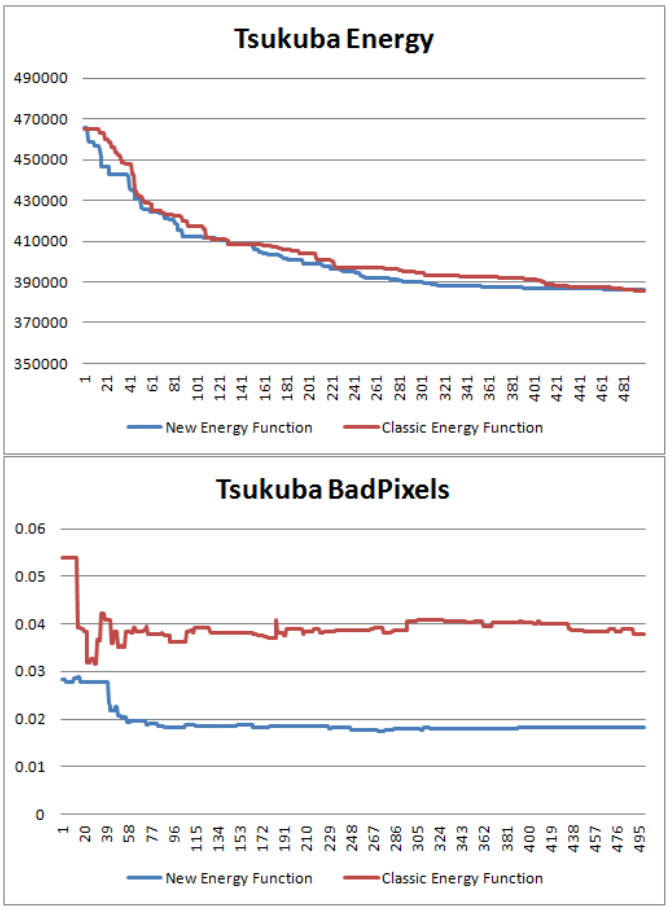

Figure 4.18: Evolution of the energy functions and the bad-pixels in Tsukuba

algorithm were reduced at half their size. The effect of this reduction is similar to the multiresolution effect studied in the previous chapter. The output of the genetic algorithm is the disparity map with half of the original size but with the disparity resolution unaltered. The main reason for this reduction is to optimize the performance of the algorithm using the CUDA implementation.

The parameters used for the genetic algorithm were the same as the ones for the Middlebury dataset described in Table 4.2. Given the nature of the environment in the dataset, the smooth term with second order priors detailed in Equation 4.9 was used. For establishing the values of the parameters, the gradient-descent based *twiddle* algorithm was run. Given a stereo pair with its ground truth and starting point for the parameters, the algorithm increments and decrements iteratively the value of each parameter trying to find out which configuration obtains the best solution. Note that it is a quite difficult task given the stochastic nature of the

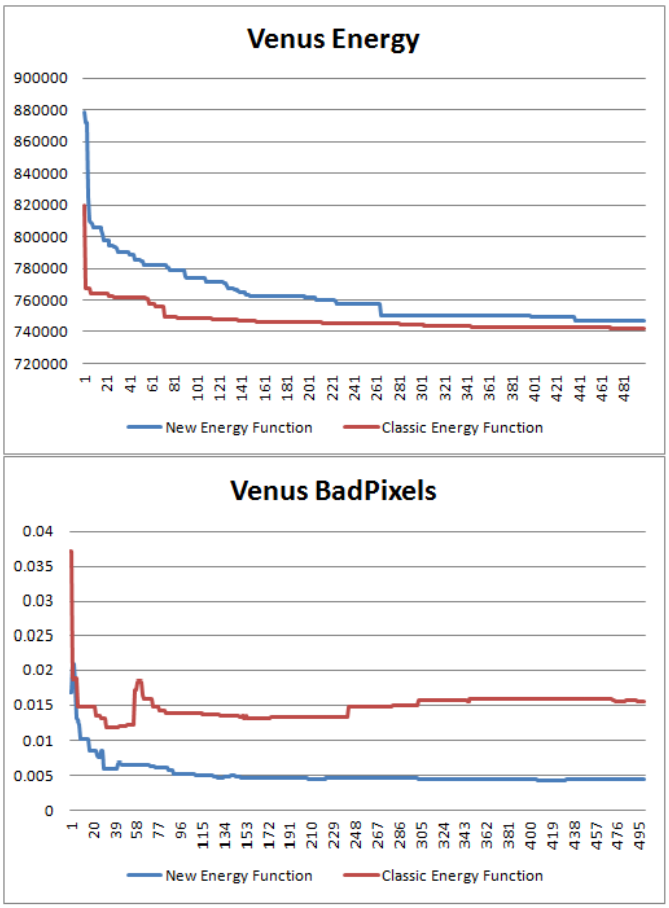

Figure 4.19: Evolution of the energy functions and the bad-pixels in Venus

genetic algorithm, and the parameters obtained might not be the best solution. However, the values obtained for the parameters are shown in Table 4.3. During the initialization process only the *census* local algorithm with different window sizes was considered after concluding that adding the other local methods or even global methods generally provoked worse solutions.

λ_d	λ_{dt}	λ_s	γ_s	φ_s	λ_{st}
13	100	18.5	1.86	15.88	11.47

Table 4.3: Parameters for the new energy function with second order priors

As a result, the disparity maps shown Figure 4.22 has been obtained using the genetic algorithm with the parameters detailed in Table 4.3. Note that the algorithm achieves good results in a quite textureless areas such as the road. The second order prior smooth term in the energy function is important in these environments given

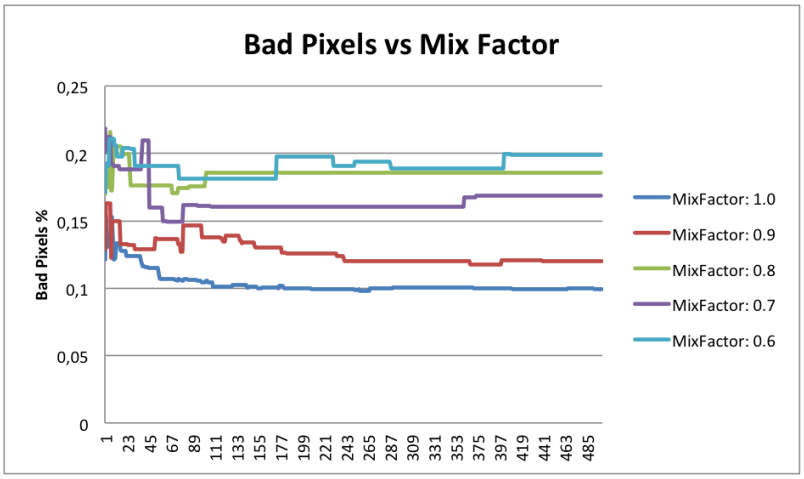

Figure 4.20: Evolution of the bad-pixels measure for different crossover mix factors

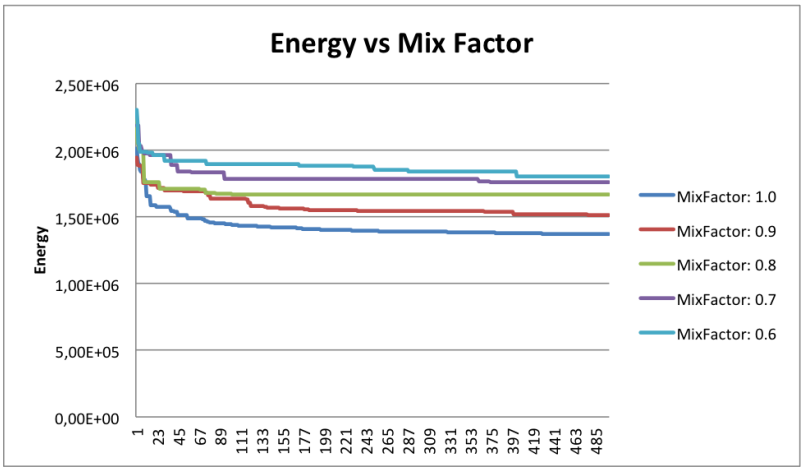

Figure 4.21: Evolution of the energy for different crossover mix factors

that the predominant planes are not fronto-parallel.

The disparity images for the test stereo pair were calculated and uploaded to the KITTI website. Thanks to their evaluation framework, the bad-pixels measure is evaluated for all the images and some global statistics of the performance of the algorithm are returned. These statistics are shown in Table 4.4.

In order to analyze the performance of the genetic algorithm given their inputs, a comparison between the bad-pixels error measure ($\psi = 3$) of some of the inputs and the final disparity map obtained is shown in Figure 4.23. The algorithm receives all the local inputs without knowing which input is better than the other. Then, based on the fitness function, the algorithm merge the local solutions and keep the best

Figure 4.22: Disparity maps obtained for some KITTI stereo pairs

Error	**Out-Noc**	**Out-All**	**Avg-Noc**	**Avg-All**
2 pixels	22.56 %	23.75 %	4.1 px	4.3 px
3 pixels	17.22 %	18.26 %	4.1 px	4.3 px
4 pixels	14.19 %	15.10 %	4.1 px	4.3 px
5 pixels	12.20 %	13.00 %	4.1 px	4.3 px

Table 4.4: Results obtained for the KITTI test images

ones.

Another important characteristic to notice is how local methods such as Census outperform global methods, e.g graph-based ones, in the KITTI dataset. This result contrasts considerably with the results obtained for the Middlebury dataset, where global methods are mostly predominant. This result can be partially justified by the mostly fronto-parallel environments, and low disparity labels present in the Middlebury data set. However, Venus image is mostly non fronto-parallel.

Finally, for visualization purposes, a three-dimensional representation of the disparity map obtained for the first image in the training set is shown in Figure 4.24.

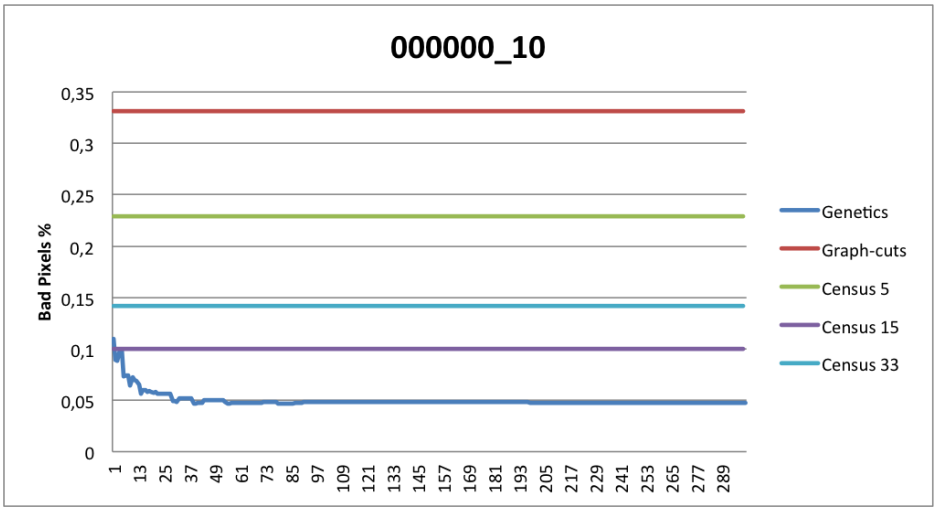

Figure 4.23: Bad pixels for different input algorithms

4.6.3 CUDA implementation analysis

In this section the parallel capabilities of the genetic algorithm are discussed. Both the serial implementation and the parallel one using CUDA, described in section 4.4, are compared. Given the stochastic nature of the algorithm and the various types of mutations that are likely to happen, the algorithm was run for different images during one hundred iterations and an average per individual and generations was calculated. The results are shown in Table 4.5.

	CPU (ms)	CUDA(ms)	Speedup
Tsukuba	20.84	0.602	34.6
Venus	31.6	0.939	33.65
Teddy	45.63	1.029	44.34
Cones	46.77	1.037	45.1

Table 4.5: Time spent for each individual and generation in CPU and CUDA implementation

The speedup column in Table 4.5 describes how many times the computational time is reduced when CUDA is used. As a conclusion, it can be stated that approximately a 40x speedup can be achieved using a parallel implementation of the algorithm. For the tests, an Intel i7-2600 at 3.4 GHz CPU and an Nvidia GeForce GTX 590 (single processor used).

Figure 4.24: 3D representation of the stereo reconstruction achieved for image 000000_10 of the KTTI dataset

4.7 Conclusions

A new genetic algorithm has been proposed for the stereo correspondence and refinement problem. Global reasoning and not restricted energy function compared to other global methods are some benefits that genetic algorithms provide. The contributions of this chapter are two-fold. Firstly, compared to other genetic algorithms previously proposed, it uses a new crossover and mutation operators that accounts for occlusion handling. Both left and right disparity images are estimated in order to manage occlusions adequately. Secondly, two new energy functions that include occluded pixels handling in the formulation and enables depth discontinuities on pixels with high photometric derivatives have been proposed and analyzed. The first one with a fronto-parallel smoothing term formulation and the second one with a second order prior formulation. Note that these energy functions can not be used with other refinement techniques. To prove the parallel nature of the genetic algorithm proposed, a GPU implementation on CUDA device has been described and analyzed.

The genetic algorithm has been evaluated using the standard Middlebury stereo dataset using both classic and proposed energy functions. The implementation using the herein proposed fronto-parallel energy function outperformed the classic one in 2.75% of bad-pixels on average, which represents a 32% of error reduction using the new energy function. Moreover, analyzing the evolution of the bad-pixels error

measurement compared to the evolution of each energy function, suggests that the new formulation is more adequate for representing real disparities. It has also been tested using the KITTI dataset for real applications, however best results have been obtained for the Middlebury set.

The algorithm proposed was rated with an average rank of 38.5 in the Middlebury ranking and as far as we know, it is the first evolutionary algorithm included on this table.

5

Transformation segmentation for visual odometry

Previous chapters have studied the stereo correspondence problem and its refinement process in order to obtain high-quality disparity images. This chapter is focused on how to use these disparity images to estimate the camera pose from a visual odometry point of view. Based on 3D registration algorithms, a transformation segmentation algorithm is herein proposed. The contribution of this work is threefold. Firstly, aiming to deal with non-static environments, it separates objects with different trajectories and velocities from the static environment. This segmentation facilitates dealing with inconsistencies found in the disparity images, improving the accuracy of the registration algorithms. Secondly, registration algorithms such as *Iterative Closest Points* (ICP) or the *Normal Distribution Transform* (NDT) are tested for viability using point-clouds obtained from stereo algorithms. Note that these algorithms has been traditionally used with laser-range data. Finally, a *shadow-free* map is demonstrated to be obtained thanks to the segmentation of the moving objects in the scene.

This chapter focuses on the last three steps described in Figure 1.1 shown in the Introduction chapter. That way, this module receives the refined dense disparity image obtained from the previous stereo modules and estimates how the camera has changed its position. It is organized as follows: in section 5.1 a brief overview of visual odometry and SLAM algorithms is presented, section 5.2 describes the transformation segmentation algorithm proposed, in section 5.3 some experimental results are shown with comparisons between the transformation segmentation and traditional registration algorithms, and finally in section 5.4 some conclusions are drawn.

5.1 Visual odometry in the literature

As explained in the *Preliminaries* chapter of this thesis, the purpose of the *visual odometry* is to find the transformation between two consecutive frames, independently from the others. If this differential calculation is repeated along a list of consecutive frames, the final position can be estimated as an aggregation of each individual transformations. Being a dead reckoning algorithm, its incremental nature makes it prone to cumulative errors. For solving this problem, most *visual odometry* algorithms use a final refinement process such as bundle adjustment, KF, loop closing algorithms, etc. More details about *visual odometry*, its most representative algorithms and its relation with other SLAM techniques are available in the *Preliminaries* chapter of this thesis.

Visual odometry algorithms can be classified into three different groups depending on how features are specified. This chapter is going to be focused on the **3D to 3D** algorithms, based on the triangulated disparity maps obtained using stereo techniques. Therefore, these algorithms are closely related to traditional registration algorithms found in other research areas.

Recently, a new type of cameras that provide color and depth information have appeared: the RGB-D cameras. Being the Microsoft's Kinect the most popular one, RGB-D cameras join the accuracy of the laser-scanners with the color information of the traditional monocular cameras. These cameras enabled the usage of traditional registration iterative algorithms, previously used only with laser-scanners, in the *visual odometry* field. In (Izadi et al., 2011) and (Newcombe et al., 2011) a quite impressive algorithm is proposed for real-time stereo reconstruction using a RGB-D camera, iterative closest points ICP (Besl and McKay, 1992) and TSDF surfaces (Curless and Levoy, 1996). Real-time is achieved by GPU computation and an efficient ICP and TSDF implementation. A more traditional approach is followed in (Fiala and Ufkes, 2011), where a feature based algorithm is proposed using SIFT combined with the depth information of the camera.

The core of these algorithms is how to register two point-clouds captured from two different positions. Traditionally, the ICP algorithm has been used for this task. It is based on a point matching step followed by an optimal transformation calculation repeated iteratively. Depending on how the matches, the distances and the transformations are calculated, several different ICP flavours appeared in the literature. Various surveys can be found in (Rusinkiewicz and Levoy, 2001),(Salvi

et al., 2007)and (Liying and WeiDong, 2009). In (Chen and Medioni, 1991) ICP with point-to-plane alignment was proposed for the first time. The generalized-ICP or GICP was proposed in (Segal et al., 2009) merging the point-to-plane ICP and the classic ICP into one probabilistic framework. A performance improvement using a hierarchical multi-resolution approach was proposed in (Li et al., 2010) and a combination of ICP and occupancy map registering is proposed in (Milstein et al., 2011). Other authors tried to include the color information of the point-cloud in the registration process (Godin et al., 1994), (Druon et al., 2006) and (Joung et al., 2009).

Recently, a new registration technique has gained interest in the registration research community, the *3D Normal-Distributions Transform* or NDT. Originally proposed in (Biber and Strasser, 2003) as a solution for the 2D registration problem, it was modified in (Magnusson et al., 2007) and (Magnusson, 2009) for the 3D case. The main difference between the ICP and the NDT algorithms is that while the first is point based, the latter is surface-based. The NDT firstly transforms the point cloud into a smooth surface representation by estimating a set of local probability density functions, each one describing a section of the surface. Then, a likelihood minimization process is solved using the Newton's algorithm by gradient and hessian calculation. A modification to the NDT that incorporates color information was presented in (Huhle et al., 2008).

A comparison between ICP and NDT was carried out in (Magnusson et al., 2009). As a conclusion, the study stated that NDT performed faster than ICP and converged from larger range initial poses, but it showed some unpredictability in the failures compared with ICP. In the failure cases, a worst estimate was achieved by the NDT compared with ICP.

The purpose of the investigation presented along this chapter is threefold. Firstly, the behavior of the ICP and NDT algorithms will be analyzed using point-cloud data obtained from stereo cameras. Notice that ICP and NDT algorithms have been traditionally used only with point-clouds obtained from laser-scanners data or RGB-D cameras. Using ICP, (Newcombe et al., 2011) obtained great results for small environments, so it is important to determine if actual stereo algorithms are accurate enough to permit these cloud-based registration algorithms.

Secondly, a registration algorithm for non-static environments based on ICP or NDT is proposed in this chapter. The aim of the algorithm is to segment the scene due to the different rigid-body transformations along time. This offers some

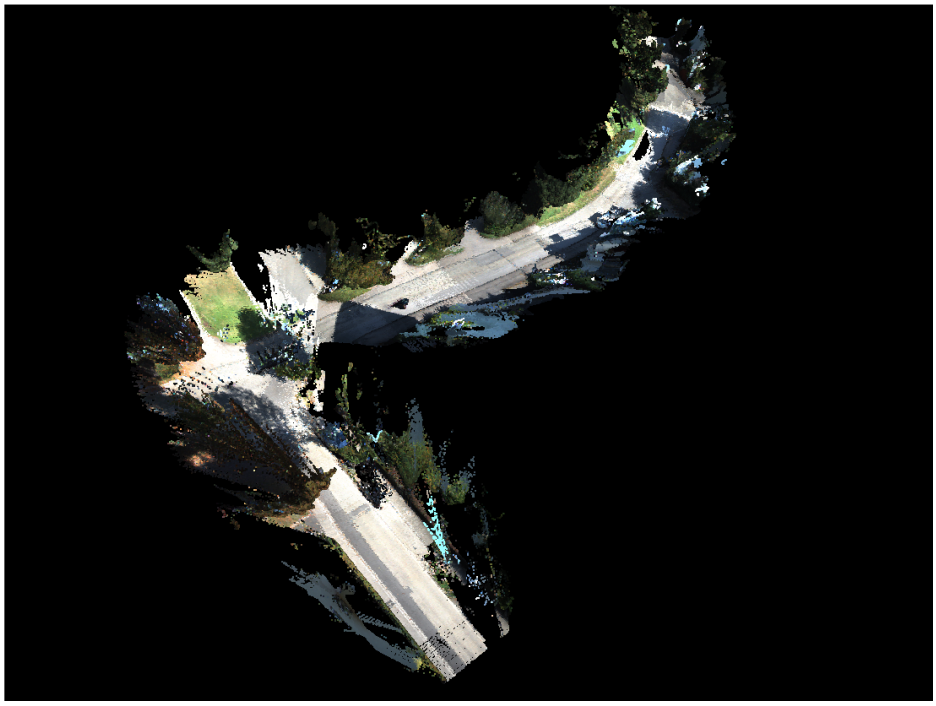

Figure 5.1: Final environment mapping of a KITTI dataset using the visual odometry algorithm herein proposed

advantages:

- Rigid bodies moving at different velocities are segmented
- Stereo errors that do not satisfy the euclidean transformation are eliminated
- ICP or NDT registration algorithms are only applied to a couple of clouds that originally demonstrated to fulfill the euclidean transformation constraints. This prevents errors caused by non-static environments.

Finally, an accurate mapping of the environment with no "shadow effect" can be obtained from the support point-clouds obtained during the registration algorithm. An example of the final map obtained using the algorithm is shown in Figure 5.1.

5.2 Transformation segmentation algorithm

The aim of the *Transformation segmentation algorithm* is to calculate the transformation that the environment has suffered when comparing two different frames

captured by the stereo camera. This transformation can be produced either by the movement of the camera or by the movement of the whole environment. In real applications, the environment may have several rigid-bodies moving at different speeds, directions and rotations, which commonly impact negatively in the camera pose estimation. Therefore, in order to explain how the whole environment has changed, it is necessary to estimate different transformation matrices, one for each moving object in the environment. Most of the algorithms cited in section 5.1 are based on a static-environment assumption and are only interested in finding the movement of the camera. Therefore, most algorithms in the literature usually treat the rest of the transformations as mere outliers.

The algorithm receives as inputs an ordered collection of consecutive stereo captures and their correspondent disparity maps. As an assumption and for simplicity, two consecutive disparity maps are considered to have quite similar visibility constraints. Transformations between two disparity maps obtained by two very different poses is also possible, but problems arise when different areas of the environment are perceived only by one of the poses.

The *Transformation segmentation algorithm* can be divided into the following tasks:

- Pre-processing
- Color region-growing segmentation of both point-clouds
- Transformation segmentation based on the j-linkage algorithm
- Transformation tracking using Belief Propagation
- Final pose refinement by means of a Kalman filter

Firstly, the algorithm starts a preprocessing stage by applying a bilateral filter to the disparity images obtained by the stereo algorithm. This step is essential because it helps to overcome the quantification error in the disparity domain smoothing surfaces while preserving discontinuities. Then, a traditional per-pixel triangulation process using the disparities and the camera calibration information is performed in order to obtain a colored point-cloud. Finally, the last step of the preprocessing stage consists of a statistical noise removal in the 3D space. Given the low accuracy of the depth dimension that is achieved using stereo techniques compared to RGB-D or laser-scanners, this first preprocessing stage helps considerably to remove the noise of the original point-cloud and to obtain a uniform cloud.

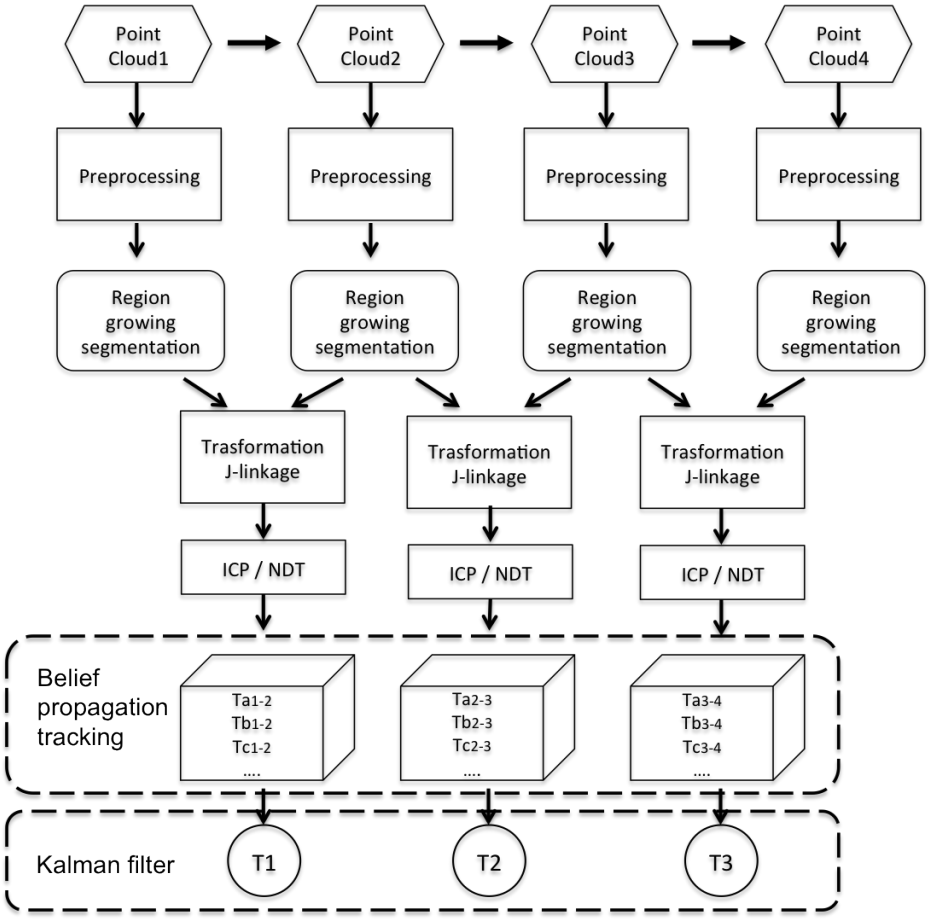

Figure 5.2: Odometry algorithm workflow

After the preprocessing stage, the core registering process is run. Its first step is a color segmentation algorithm run in both point-clouds. The clustering is achieved considering color and normal information with a color region-growing segmentation algorithm. Then, a J-linkage based algorithm is applied to the clusters between frames in order to obtain a transformation segmentation. This step is the main step of the algorithm and proposes several transformations with their associated supporting clusters. Then, a final registration algorithm is performed considering just the clusters associated to each transformation obtained in the previous step. Finally, in order to track the transformations and, thus, the elements in the environment, a simple tracking algorithm based on belief propagation is run on several frames followed by a Kalman Filter. In the following subsections each step is explained in greater detail. The global work-flow of the algorithm is represented in Figure 5.2.

5.2.1 Preprocessing

The preprocessing step receives the disparity map obtained by any stereo algorithm and it aims to produce a three-dimensional point-cloud with as little noise as possible. This step is composed by the following algorithms: Bilateral filtering, triangulation and color region growing segmentation.

5.2.1.1 Bilateral filter and triangulation

Most stereo algorithms obtain a discrete disparity map, i.e. the disparity dimension is usually represented as an integer number, producing a quantization effect. Although the disparity map looks great as a two-dimensional image, when transformed to a point-cloud using triangulation, the quantization is very noticeable. Geometry exists only at certain depth levels in the 3D space, creating fake geometry. Taking care of this effect is absolutely essential and has a deep impact on the performance of the registration algorithms applied in the following steps.

Therefore, a 2D smoothing filter is applied to the disparity map to obtain continuous disparity values and to reduce the noise. For this task, the bilateral filter (Tomasi and Manduchi, 1998), which is a classic edge-preserving smoothing algorithm commonly used for noise removal applications, was used.

Once a smooth disparity map is obtained, a classical triangulation process, which transforms disparity values into three-dimensional points using the stereo camera intrinsic parameters is performed. Figure 5.3 shows the effect of the bilateral prefiltering comparing the point-clouds obtained with and without filtering.

5.2.1.2 Clipping and statistical point-cloud noise removal

The noise removal process comprises a depth-clipping step and a statistical outlier removal. The depth-clipping or Z-clipping phase is needed due to the lack of precision inherent to the stereo algorithms at large depths (small disparities). Similarly to the laser scanners, which are designed with a range limit, stereo has also a range limit related to the resolution of the stereo pair, the lenses, how the stereo rig is positioned, etc. The main problem with stereo is that this limit has to be tested and is application dependent rather than being simply specified as a characteristic of the machine, as occurs with the laser scanners. The clipping process is a very simple operation that

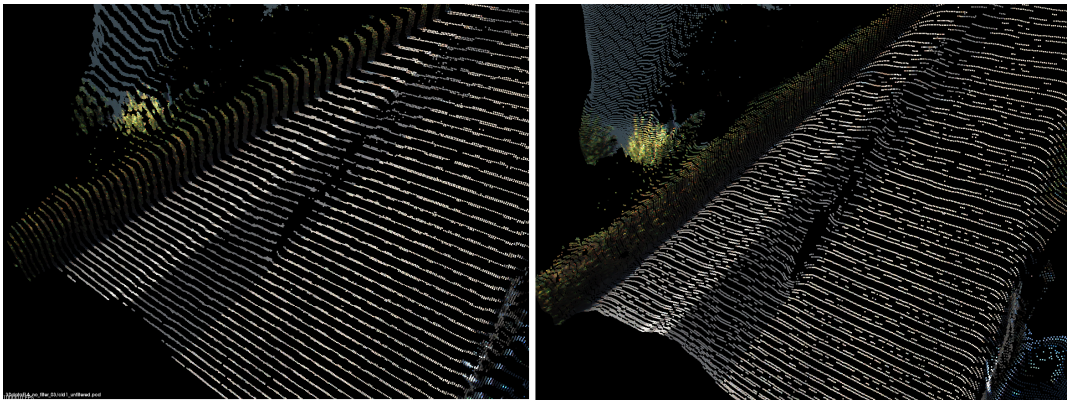

Figure 5.3: Bilateral filter effect after triangulation; left: point-cloud obtained with no previous bilateral filter, right: point-cloud obtained with triangulation after bilateral filter

removes the points that lie farther than a certain clipping distance in the Z axis. Besides the Z-clipping plane, other planes can be used if certain regions of the stereo pair are not trustworthy, i.e. the occluded left most side of the left disparity images.

For noise removal, the statistical outlier removal algorithm described in (Rusu et al., 2008) is used. The algorithm first calculates the mean and standard deviation in the neighborhood of each point and then it classifies the points in inliers or outliers depending on their distance to the mean value. Figure 5.4 shows the effect of this step on a point-cloud example.

5.2.2 Color region growing segmentation

This step aims to perform a segmentation over each point-cloud. Quite a lot of algorithms have been published that deal with this problem. Thanks to stereo cameras are being used, besides the topological information of the point-cloud, the color information is also available. Therefore, proximity, normal and photometric information are used for segmentation purposes. Remember that traditional point-cloud segmentation algorithms only use topological information due to the lack of color information when using traditional laser-scanners.

A region growing segmentation algorithm which considers the color information was finally used to solve this task (Zhan et al., 2009). The algorithm starts by performing a region growing step followed by region merging and refinement. It uses the K nearest neighbors algorithm described in (Bentley, 1975) for efficient

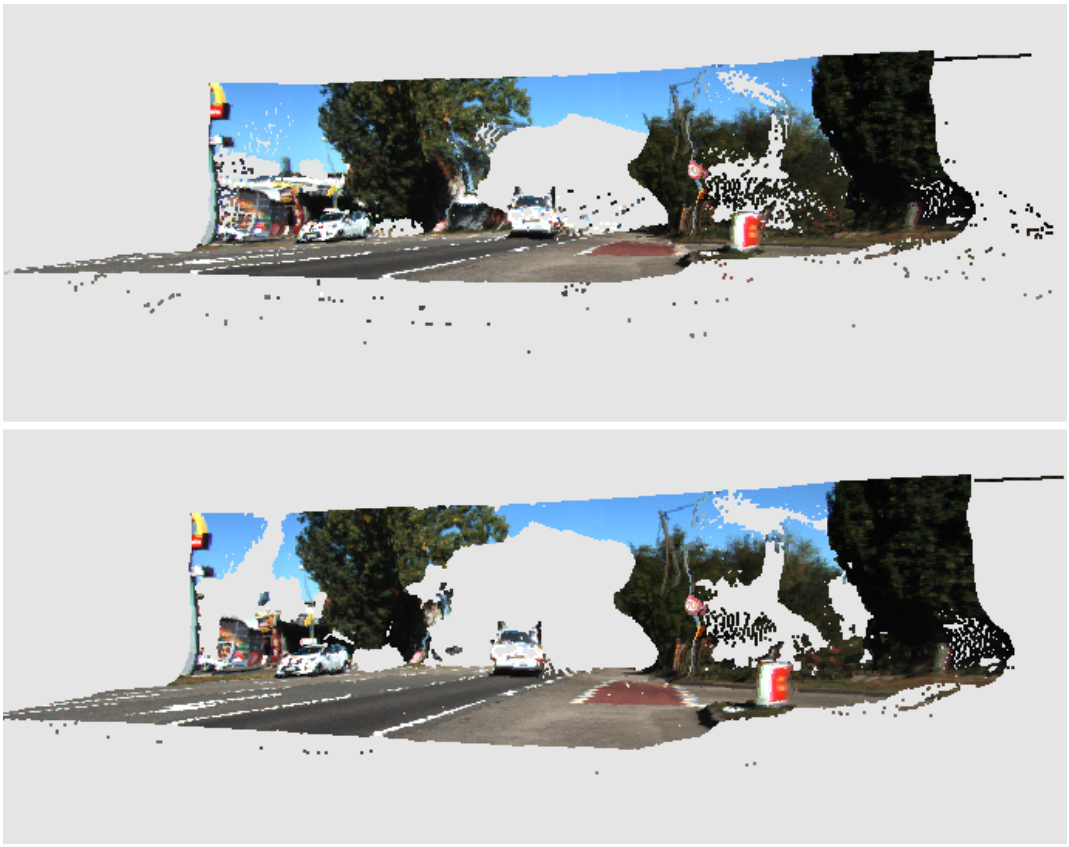

Figure 5.4: Statistical noise removal effect; left: point-cloud before statistical noise removal, righ: point-cloud after statistical noise removal

region growing calculation. For the shake of brevity, a look to the original paper is recommended for greater detail of the algorithm. For a good segmentation, several parameters have to be tuned, such as the *point to point colorimetrical similarity threshold* TPP, the *size of neighborhood* TNN, the *point to point distance threshold* TD, the *region-region colorimetrical similarity threshold* TRR and *minimal region size* Min. An example of the segmentation obtained using this technique can be seen in Figure 5.5

Performing a correct segmentation in this step is fundamental for the overall performance of the algorithm. Each segment obtained as an output by this step will be considered as a rigid body in the following steps. Several clusters may be associated with one common transformation, forming for example the static environment, but all the inner points in a cluster must have a common transformation. Otherwise, the cluster will probably be considered an outlier and it will not fit in any transformation.

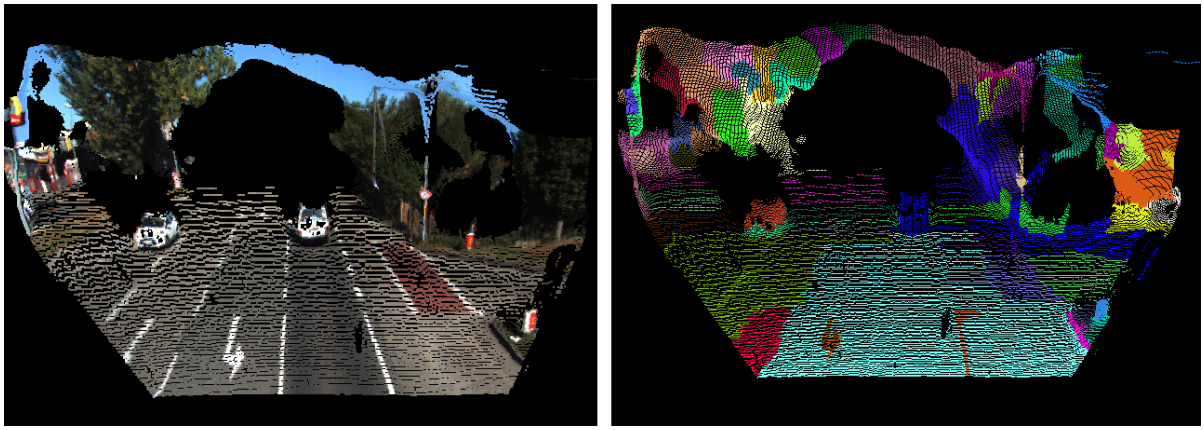

Figure 5.5: color region growing segmentation example. Left: original color point-cloud, right: segmented point-cloud

5.2.3 J-linkage applied to visual odometry

The main objective of the J-linkage step is to obtain, given the clustered input point-clouds, several transformation proposals that explain how the clusters have moved and rotated during the two frames. One important characteristic is that several transformations are proposed in order to explain different moving objects in the environment.

For this task, an algorithm based on the J-linkage multimodal model fitting, firstly proposed in (Toldo and Fusiello, 2008) and later refined in (Toldo and Fusiello, 2010), has been developed. Based on RANSAC (Fischler and Bolles, 1981), the algorithm is capable of obtaining multiple instances of a model, classifying data into inliers and outliers referred to each of the models obtained. This classification also permits to obtain a better estimation of the parameters of each model because only the inliers are considered for the estimation of the final parameters.

One of the main contributions of this work is applying the J-linkage algorithm to the visual odometry problem for multimodal transformation estimation. This section describes in great detail this process. First lets take a brief overview of how the J-linkage algorithm works. The original J-linkage algorithm consists of three steps:

- Random sampling
- J-linkage clustering
- Model refinement

Firstly, the random sampling is in charge of proposing different models given small subsets of the data. Each iteration of this process is meant to be fast and lots of erroneous models will be probably proposed. The data is then classified into inliers (consensus set) and outliers for each proposed model. A matrix containing this information is built, giving as a result the *preference set* (PS) for each data input. The clustering step reduces the size of the matrix by merging data points with similar preference set. Finally, the largest preference sets are considered models appearing in the data. Each cluster is considered free from outliers and a final parameter estimation process is performed.

Now that the general structure of the original algorithm has been briefly explained, the application of J-linkage to the visual odometry problem is now detailed. Keep in mind that the algorithm starts with the cluster classification previously obtained. The **random sampling** is an iterative process that is meant to propose possible models existing in the data. In the visual odometry problem, the models are the transformation matrices that explain the changes from frame to frame. Therefore, for each iteration of the random sampling, a transformation (model) is proposed. The frame earlier in time will be called the *source frame* and the following frame *target frame* and their related point clouds *sourcPCL* and *targetPCL* respectively.

The algorithm 1 shows the pseudocode for the random sampling algorithm. For each iteration, a cluster from the *sourcePCL* is selected and its correspondent cluster is searched through a bunch of possible candidates from the *targetPCL* point cloud. The candidates are chosen based on their cluster size and their mean-color characteristics. For each candidate, an initial registering algorithm such as the *color generalized iterative closest points* (color GICP (Segal et al., 2009)) is run in order to estimate the transformation from the source cluster to the target candidate one. It also permits to determine if both point clouds fit together and decide if they where sampled from the same real object. For that purpose, the fitness between point-clouds is calculated after the color GICP is run and, if it is higher than a certain threshold δ_c, the transformation is proposed as a possible model for the J-linkage step. The fitness function between point-clouds is detailed in subsubsection 5.2.3.1.

In this sense, algorithm 1 has the same structure as the original J-linkage algorithm. Several models, spatial transformations in this case, are proposed using a small data subset, the clusters. Then, the rest of the clusters are considered inliers or outliers given that transformation, i.e. if the clusters also satisfy that transformation. The random sampling step calculates the preference set for each cluster, i.e. which

Input: $sourcePCL$, $targetPCL$
Output: M
$M \leftarrow \varnothing$
$ListT_{proposals} \leftarrow \varnothing$
$n = 0$
while $n < N_{iter}$ **do**
$cluster_{source} = RandomClusterFromSource()$
$target_{candidates} = SearchSimilarClusters(cluster_{source})$
for all $cluster_{tg}$ **in** $target_{candidates}$ **do**
$T_{proposed} = colorGICP(cluster_{source}, cluster_{tg})$
$cluster_{srcTransf} = ApplyTransformationToPCL(T_{proposed}, cluster_{source})$
$fit = FitnessBetweenPCLs(cluster_{srcTransf}, cluster_{tg})$
if $fit > \delta_c$ **then**
$ListT_{proposals} \leftarrow T_{proposed}$
$n \leftarrow n + 1$
end if
end for
end while
for all T **in** $ListT_{proposals}$ **do**
for all $cluster_{src}$ **in** $sourcePCL$ **do**
$cluster_{srcTransf} = ApplyTransformationToPCL(T, cluster_{src})$
$fit = FitnessBetweenPCLs(cluster_{srcTransf}, targetPCL)$
if $fit > (\delta_i - 0.1)$ **then**
$M(cluster_{src}, T) \leftarrow 1$
else
$M(cluster_{src}, T) \leftarrow 0$
end if
end for
end for

Algorithm 1: Random sampling for visual odometry

transformations explain the movement of each cluster from one frame to the next, Equation 5.1.

$$M(cluster, T) \begin{cases} 1 & \textit{if cluster supports the transformation } T \\ 0 & \textit{otherwise} \end{cases} \tag{5.1}$$

Notice that the M matrix is not square, having the number of clusters in the source point-cloud $sourcePCL$ a rows and the number of $ListT_{proposals}$ as columns.

A lot of transformations that appear in the M matrix as columns are actually repeated. This is caused by the difference in the parameters and the inliers that appear in the estimated transformations due to the different clusters used in the GICP algorithm. Thus, an algorithm for merging different preference sets is now required. This task is performed by the next step, the **J-linkage clustering**. Based only on the clusters preference set, a hierarchical agglomerative algorithm is run. For this task, the *Jaccard distance* between different preference sets is used (Equation 5.2).

$$d_J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{5.2}$$

being A and B two binary strings and the operators $\cup$ and $\cap$ are the *or* and *and* bitwise operators. It measures the degree of overlap between two sets, where 0 represents totally equal and 1 totally different. All the *Jaccard distances* between clusters can be expressed in a compact manner in matrix form D_J, where the element $D_J(i, j)$ is the distance $d_J(i, j)$ between the preference sets of cluster i and cluster j.

The pseudocode for the J-linkage agglomerative clustering is detailed in algorithm 2. In each iteration, the cluster pair with lowest d_J distance is selected and merged until the minimum distance between clusters is higher than a threshold δ_J. Generally, the parameter $\delta_J = 1.0$ is used. The merge procedure between the cluster's preference sets is done using the *intersection* operator $\cap$. This guarantees that the resulting merged cluster supports only the transformations that are supported by both the original clusters. The output of this step is the row-reduced M matrix where

Finally, the precise parameters of each transformation are calculated using only the clusters supporting it. This estimation is achieved using any registration algorithm, such as *color GICP* or *NDT*, again but with considerable better results compared to the ones performed in the random sampling step for two reasons:

- The transformation is estimated only using inlier clusters
- Using several clusters in the registration increases the accuracy

By the end of this process, several transformations with their related clusters are

Input: M
Output: M
$dist \leftarrow \varnothing$
while $dist < \delta_J$ **do**
$D \leftarrow JaccardDistances(M)$
$dist, row1, row2 \leftarrow \min(D)$
if $dist < \delta_i$ **then**
$M \leftarrow mergeRowsIntersection(M, row1, row2)$
end if
end while

Algorithm 2: J-Linkage clustering algorithm

proposed. This step is fundamental for both location and mapping processes. An example of the segmentation obtained by this process is shown in Figure 5.6

5.2.3.1 Fitness function between point-clouds

Knowing whether two point-clouds correctly fit together or not after a registration process is basic to know if the registration was successful and if those point-clouds were originally obtained from a sampling process from the same surface. Establishing the fitness function as the nearest-neighbor distance for each point is a possibility, but some sampling error is inevitable and might produce erroneous results. Thus, a point-to-surface approach is used. Moreover, color information shall be also included in the fitness function, providing more information about the correctness of the registration.

The mean point-to-surface distance is defined as

$$\mu_d = \frac{1}{N} \sum_{i}^{N} (\bar{p_i} - \bar{p_{nni}}) \cdot \bar{N_{NNi}} \tag{5.3}$$

where N is the number of points in the reference point-cloud, $\bar{p_i}$ are the coordinates of the reference points, $\bar{p_{nni}}$ are the coordinates of the nearest-neighbor of point i belonging to the target point-cloud, $\bar{N_{NNi}}$ is the normal vector corresponding to the nearest-neighbor of i in the target point-cloud and $\cdot$ is the dot product. Note that a previous normal estimation of the target point-cloud is needed.

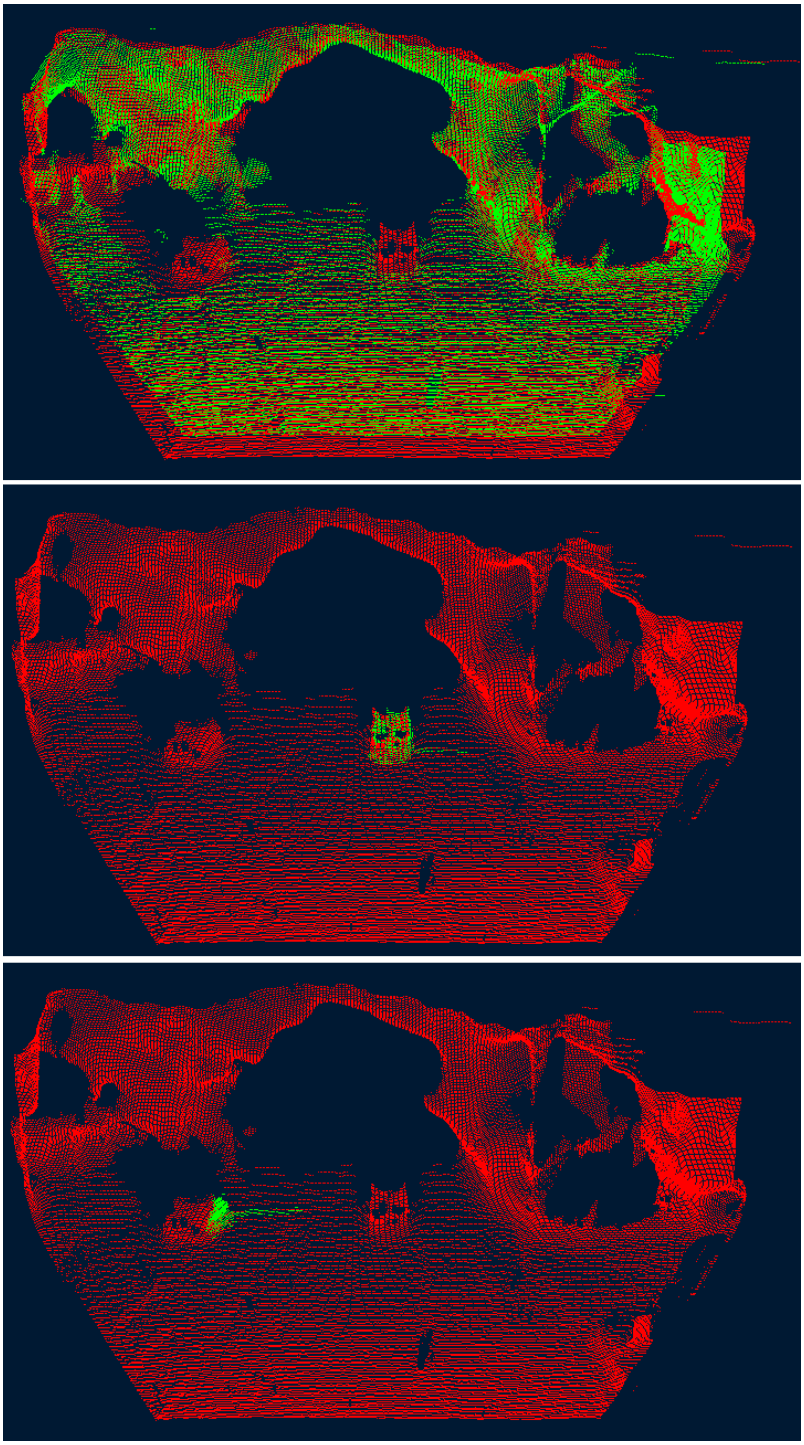

Figure 5.6: Transformation segmentation after the J-linkage step. Each green point-cloud has been detected as a rigid body with a different velocity

The mean color distance is defined as

$$\mu_c = \frac{1}{N}\sum_{i}^{N} |\bar{c_i} - \bar{c_{NNi}}| \tag{5.4}$$

where $\bar{c_i}$ is the color vector in unsigned char units (0-255) of the point i and $\bar{c_{NNi}}$ the color vector of the nearest-neighbor from the target point-cloud.

Finally, a global $fitness$ measure can be calculated as

$$fitness = e^{-\left(\frac{\mu_d}{0.3}+\frac{\mu_c}{4.5}\right)} \tag{5.5}$$

where the 0.3 and 4.5 factors were calculated for weighting properly both distance and distance fitness. The exponential function provides a decaying curve for high distance values, so high values of $fitness$ indicates that the mean point-to-surface and the color distances are low, and that both point-clouds fit well.

5.2.4 Motion tracking using belief propagation

Once several transformations per frame have been proposed, a method for linking them along the time is needed. For this task, a stochastic framework is used for describing how likely a group of transformations actually corresponds to the movement of the same static object in the environment. Note that there will be a group of transformations that will correspond to the movement of the camera relative to the static environment. The environment, being the *static world*, is considered to appear in every frame, and its transformations are used for calculating the absolute movement of the stereo pair. Although the method herein described will only focus on calculating the per frame transformations of the cameras, it could also be used for finding other static objects with different transformations.

The problem consists of selecting one transformation per frame. Therefore, each frame will be related to a stochastic variable S_i, indicating which transformation has been considered for frame i to explain the transformation of the environment. The likelihood of each stochastic state will depend on some characteristics of the chosen transformation itself and on the state of the next and previous stochastic variables. The graphical model of these stochastic relations is shown in Figure 5.7.

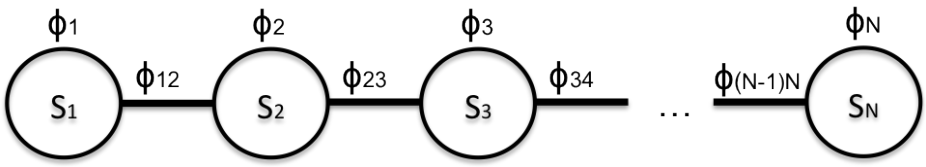

Figure 5.7: Graphical model for the camera tracking problem

The probability of each combination of stochastic states will be determined by the *factors* of the graphical model. Two different kinds of factors are proposed, the individual and the pairwise factors. The individual factors only depend on the state of one stochastic variable and it is represented as Φ_i in Figure 5.7 being i the related frame. On the contrary, the pairwise factors depend on two contiguous states and indicate the probability of that stochastic configuration. Remember that the values of the factors are not probabilities by themselves and they can take any positive real number.

Estimating the movement between two consecutive frames is equivalent to estimating the relative speed of the objects considered. Both linear and angular velocities can be calculated from the transformation matrix, first obtaining the relative displacement and angular change and then dividing it by the time spent from frame to frame. These speeds will be used for linking the transformations from frame to frame using a constant velocity model prior.

The equation describing the value of the individual factors is

$$\Phi_i(T_i^k) = \beta_\Phi \cdot cloudSize(T_i^k) \cdot fitness(T_i^k) \tag{5.6}$$

where Φ_i is the value of the factor on state i, T_i^k is the k transformation proposed for state i, β_Φ is a constant weighting parameter, $cloudSize(T)$ is a function that returns the size of the point-cloud that supports the transformation T and the $fitness(T)$ function returns the fitness of the point-cloud associated with transformation T, one indicating perfect fitness and 0 very low fitness. Both $cloudSize()$ and $fitness()$ functions need the segmented point-clouds and are pre-calculated for each transformation in the previous J-linkage step. This factor basically favour the transformations supported by big point-clouds (static environment is usually the biggest rigid body in the scene) and penalizes the transformations that obtained a high error rate during the color GICP step. Note that the equation Equation 5.6 only depends on the state's transformations and it can be calculated independently

of the rest of the stochastic variables.

The pairwise factors are calculated as follows

$$\Phi_{ij}(T_i^k, T_j^k) = \frac{1}{e^{1.155 \cdot \left(\left|v(T_i^k) - v(T_j^k)\right| + 20\left|w(T_i^k) - w(T_j^k)\right|\right)}} \tag{5.7}$$

where $\Phi_{ij}(T_i^k, T_j^k)$ is the pairwise factor, T_i^k is the proposed transformation k in state i, T_j^f is the proposed transformation f in state j, $v(T)$ is the function that returns the linear velocity of input transformation T and $w(T)$ is the function that returns the angular velocity of T. Note that i and j in this case are two contiguous states. It basically favours a constant speed and angular velocity from frame to frame. The 20 multiplier is a weighting constant between rotation and translation measures and the 1.155 multiplier is to obtain a proper exponential decay. Note that the pairwise factors only depend on the transformations analyzed and the structure is constant over the all the states.

Now that the factors have been determined, the probability for each state configuration can be calculated as a product of factors followed by a normalization in a Gibbs distribution. However, the aim of this step is not to know how probable is one certain state configuration, but to find out which is the most likely state configuration. This problem is called the *max product* and can be solved using belief propagation. For this task, an implementation of the algorithms described in (Mooij, 2010) was used.

5.2.5 Pose refinement

The final step of the visual odometry algorithm is the pose refinement using a standard Kalman filter. Given that the dead reckoning is an intrinsic characteristic of an odometry algorithm and small errors in one estimation affects the result of the following ones, a Kalman filter that minimizes the error and accounts for noise in each transformation estimation is rather useful.

Remember that the Kalman filter is a prediction-correction process where the knowledge of the natural evolution of a system is combined with the measurements evidences for estimating the system's state. The state vector considered for the three-dimensional rotation and translation case studied in this chapter includes both velocity and acceleration

$$x_t = \begin{pmatrix} v_t \\ a_t \end{pmatrix} \tag{5.8}$$

being v_t and a_t the velocity and acceleration vectors at time t, formed by both the rotation and translation velocities and accelerations respectively. Note that in three-dimensional space v and a are vectors of 6 dimensions each.

The measurement vector is the rotation and translation speed measured from one frame to another

$$y_t = \frac{1}{\Delta t} \begin{pmatrix} r_t \\ t_t \end{pmatrix} \tag{5.9}$$

being r_t and t_t the rotation and translation vectors for time t obtained in the previous *motion tracking* step.

The dynamic model explains how the system evolves only considering the information of the previous system's state. For our implementation, a constant acceleration model was used.

$$x_t = A_{t-1} x_{t-1} + q_{k-1} \tag{5.10}$$

$$A_t = \begin{pmatrix} I & \Delta I \\ 0 & I \end{pmatrix} \tag{5.11}$$

being q_t the error in the dynamic model, following a distribution function

$$q_t \sim N(0, Q_k) \tag{5.12}$$

being Q_k the covariance matrix of the q_t error. The definition of this covariance matrix will determine how probable the system's actual state can be predicted efficiently by knowing the previous step. Its definition is very application-specific.

The measurement-correction process is represented by the following equations

$$y_t = H_t x_t + r_t \tag{5.13}$$

$$H_t = \begin{pmatrix} I & 0 \end{pmatrix} \tag{5.14}$$

being r_t the measurement error represented by the normal distribution

$$r_t \sim N(0, R_t) \tag{5.15}$$

being R_t the covariance matrix of the measurement error. The definition of this matrix will indicate how trustworthy the measurements are and it is usually application specific. For example, registration algorithms used in point-clouds obtained with stereo algorithms are naturally more accurate determining the rotation than the translation. Moreover, the translation on the Z axis is usually considered to have a higher error variance.

However, regardless of the R_t definition, it is important to make it dependent on the fitness obtained by the registration algorithm. This fitness will determine how nicely the two point-clouds overlap and is calculated as subsubsection 5.2.3.1 detailed. The factor used for the R_t covariance matrix in order to include the measure uncertainty was calculated as

$$K_{Ri} = \frac{1}{1 + e^{50(fitness(T_i) - 0.77)}} \tag{5.16}$$

where T_i is the transformation obtained from registration algorithm.

Just as a reminder, the *prediction equations* for updating the state vector are

$$\begin{aligned} x_t^- &= A_{t-1} x_{t-1} \\ P_t^- &= A_{t-1} P_{t-1} A_{t-1}^T + Q_{t-1} \end{aligned} \tag{5.17}$$

being P_t the covariance matrix representing the uncertainty of the actual state vector. Note that for the initialization process a P_0 representing the uncertainty of the initial state vector has to be supplied. Finally, the *correction equations* for updating the state vector with some measurements are

$$
\begin{aligned}
v_t &= y_t - H_t x_t \\
S_t &= H_t P_t^- H_t^T + R_t \\
K_t &= P_t^- H_t^T S_t^{-1} \\
x_t &= x_t^- + K_t v_t \\
P_t &= P_t^- - K_t S_t K_t^T
\end{aligned}
\tag{5.18}
$$

5.3 Experimental results

The proposed algorithm has been tested using the KITTI dataset (Geiger et al., 2012) similarly to the previous chapters. Here, the high resolution stereo video sequences with accurate ground truth information are used to evaluate the accuracy of the algorithm herein proposed. The video sequences are taken from a stereo camera mounted on a car while circulating through several environments. Thus, most environments are semi-structured. However, note that the general algorithm is independent of the structure of the environment.

For the accuracy analysis, three different cases were drawn from the dataset. The selection was made based on the quantity of movement present on the environment. Given that the strength of the algorithm here proposed is that it is capable of dealing with non-static scenes, the *scene 04* was selected from the training set. Having quite a lot of movement all around the scene, with a static vehicle constantly in front of the camera and several vehicles passing by on the other direction, it is a scene quite difficult to deal with for traditional methods. It will be treated as the main scene for comparison. Another case with some movement is the *scene 03*, where another car is in front of the camera during half the scene. Finally, the *scene 05* was also selected as an example of a static environment.

Firstly, a description of the error metrics used for the evaluation is detailed in subsection 5.3.1. Then, some tests are carried out in order to analyze the behavior of the algorithm, when the different stereo algorithms, different maximum distance parameters and different registration methods are used. Finally, a comparison with traditional registration algorithms is performed at the end of subsection 5.3.3.

5.3.1 Evaluation metrics

Several error metrics have been proposed for odometry accuracy evaluation in the literature. In this section, the error metrics proposed by the KITTI dataset (Geiger et al., 2012), extending the metrics proposed in the previous study (KÃijmmerle et al., 2009), are used. The rotation and translation errors are separated in different metrics, being defined by

$$E_{trans}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \|(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\| \tag{5.19}$$

$$E_{rot}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{(i,j)\in\mathcal{F}} \angle\left[(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\right] \tag{5.20}$$

where $\mathcal{F}$ is the set of consecutive frames in the stereo capture, $\hat{p}_i$ is the position and rotation estimated by the visual odometry algorithm at frame i, p_i is the ground truth position and rotation at frame i and $\angle$ is the function that returns the angle of a transformation. $\ominus$ is the inverse composition operator, in comparison to the $\oplus$ operator which is the composition operator.

5.3.2 Parameters

Table 5.1 shows the constant parameters used for every simulation run along this section.

δ_c	δ_i	N_{iter}	β_Φ
0.8	30	1.0	0.0001

Table 5.1: Test Parameters

Just as a reminder, δ_c is the acceptance fitness threshold in the J-linkage step, δ_i is the Jaccard maximum clustering distance, N_{iter} is the maximum number of iterations in the J-linkage step and β_Φ is the weighting parameter for the individual factors in the motion tracking step.

5.3.3 Accuracy analysis

The accuracy of the odometry transformation segmentation algorithm depending on the stereo algorithm used and the maximum clipping distance is analyzed. A comparison between classic registration algorithms and the transformation segmentation is shown and the performance of GICP compared to NDT is evaluated.

5.3.3.1 Stereo algorithm impact

Selecting the proper stereo algorithm for performing the registration process is crucial. In this section, the sensitivity of the visual odometry algorithm to the stereo algorithm is analyzed. Four different stereo algorithms were used as inputs:

- Local with window based with consistency test
- Global MRF solved using graph cuts (chapter 2)
- Genetics (chapter 3)
- Recient high-end local stereo algorithm (Geiger et al., 2011a), ELAS

The list has been sorted by increasing accuracy, based on standard datasets such as (Scharstein and Szeliski, 2002) and (Geiger et al., 2012), being ELAS the most accurate.

For this study, the scene with the most quantity of moving objects, the *scene 04*, was used. For evaluating the impact of the stereo algorithms on the odometry results, our algorithm was used in two different configurations. For the first configuration, the GICP algorithm was used for the registration operations performed inside our algorithm. This configuration will be called TS_GICP along this section. The results are shown in Figure 5.8. The second configuration uses the NDT registration algorithm instead of the GICP. This configuration will be designed as TS_NDT. Results using this configuration are shown in Figure 5.9. These analysis permit to evaluate the impact of the registration algorithm in addition to the stereo algorithm.

Firstly, Figure 5.8 shows the analysis for the TS_GICP case. The first figure shows the path obtained by the visual odometry algorithm combined with each of the stereo algorithms studied. For this case study, the ground truth is a straight line, so deviations from it are considered to be erroneous estimations. By watching the

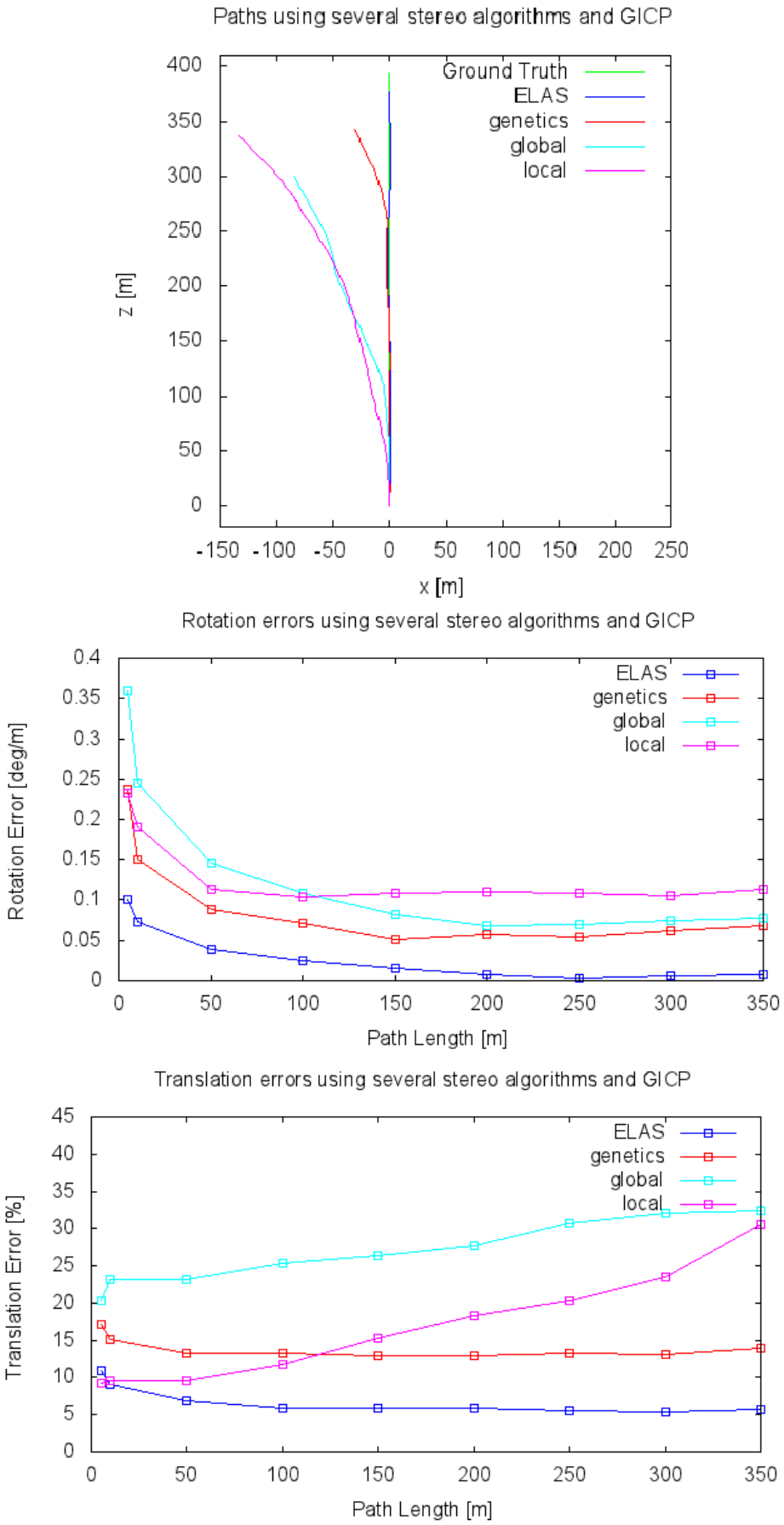

Figure 5.8: Errors obtained for different stereo algorithms using TS_GICP algorithm

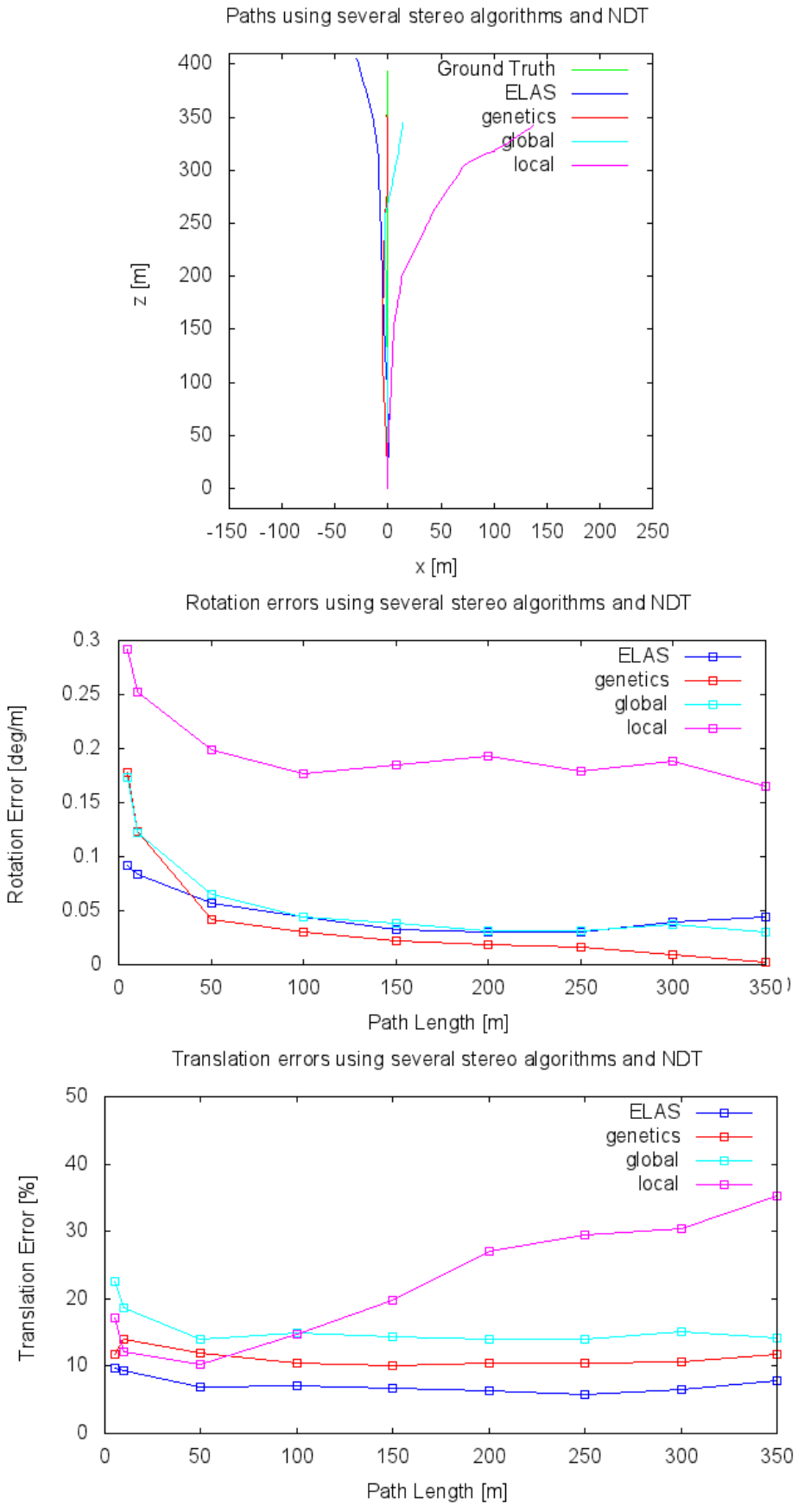

Figure 5.9: Errors obtained for different stereo algorithms using TS_NDT algorithm

trajectories, it is rather clear that the accuracy of the odometry results is correlated with the accuracy of the stereo algorithm. The most accurate stereo algorithm, ELAS, clearly obtains the most accurate path, while the genetic algorithm obtains the second place and the other two obtain quite bad results.

This analysis is supported by the error metrics used. Firstly, the rotation error is shown in the chart in the middle of Figure 5.8. It establishes the same accuracy order as the one viewed in the path analysis. Clearly, the ELAS algorithm obtains the best result with practically zero error, while the genetics obtains a second place followed closely by the global algorithm. Note that the error difference between the ELAS algorithm and the rest is quite large.

Finally, the last chart in Figure 5.8 shows the translation error. In this case, the same expected results are achieved. ELAS achieves the most accurate results with a 5.6% translation error, followed by the genetics with an approximate 13.9% at 350m. The case of the local algorithm worth a special mention, obtaining good results at the start but finding its error increased linearly with the path length.

These results are expected because the registration algorithms are obviously very dependent on the accuracy of the point-clouds used. Therefore, more accurate stereo algorithms are prone to obtain better odometry results using registration algorithms. However, these results were obtained for the TS_GICP registering algorithm case, the behavior of the TS_NDT case is studied in the following paragraphs.

The upper chart of Figure 5.9 represents the paths obtained for the TS_NDT registering case. The trajectory accuracy obtained by the algorithms is now not as evident as in the TS_GICP case. The genetic algorithm obtains the most straight line, but also a shorter one compared to ELAS. On the contrary, ELAS obtain a better distance estimation, but the trajectory is quite deviated. Only by watching the path, no conclusions can be drawn.

The rotation error chart supports the visual information shown by the path chart. In this case, the genetic algorithm obtains the best solution, followed by the global and ELAS. Finally, the translation error chart shows that ELAS is a bit more accurate than genetics, having a 7.7% error the former and approximately a 11.7% the latter.

Analysing the impact of the stereo algorithm in the TS_NDT registration case is not as obvious as in the TS_GICP one. Although a better translation error is achieved by the ELAS algorithm, it is not clear if it obtains better global results compared to the genetic.

The traditional *local algorithm* are the worst performing algorithm and shall not be used for this purpose, at least for autonomous driving application. This can be explained by the general worse performance of this algorithm and specially the texture-less areas appearing in the video captures such as the road. The *global* algorithm performs a bit better than the local one, specially if the TS_NDT registering algorithm is used. The *genetic* algorithm obtains a good accuracy if used also with the TS_NDT algorithm. Finally, the best results considering every configuration tested were obtained using the ELAS stereo algorithm and the TS_GICP registering case.

5.3.3.2 Clipping maximum distance analysis

The visual odometry algorithm comprises a clipping procedure described in subsubsection 5.2.1.2 as part of the preprocessing stage. The clipping function is defined by a maximum clipping distance and determines the maximum local Z coordinate of the points that remain for the rest of the algorithm. For the laser-scanner case, this distance is usually defined by the range of the laser device, but for the passive stereo case, no theoretical maximum is defined. Keeping points too far from the camera will impact negatively during the registration step due to the lower accuracy of far points while increasing the computational complexity. However, establishing a clipping distance too close to the camera will produce an increasing rotational error.

Figure 5.10 shows both rotation and translation error charts for different maximum clipping distances. For the analysis, the algorithm herein presented with TS_GICP was used. The charts show that both rotation and translation accuracies improve if further clipping distances are used. Therefore, for the rest of the analysis a clipping distance of 150 metres is used.

5.3.3.3 Comparison of registration algorithms

Finally, this section compares the odometry algorithm proposed in this chapter with the registration algorithms traditionally used in the literature. The algorithms analyzed are:

- Traditional Generalized Iterative Closest Points with color information, followed by a KF (GICP)
- Traditional Normal-distributions Transform, followed by a KF (NDT)

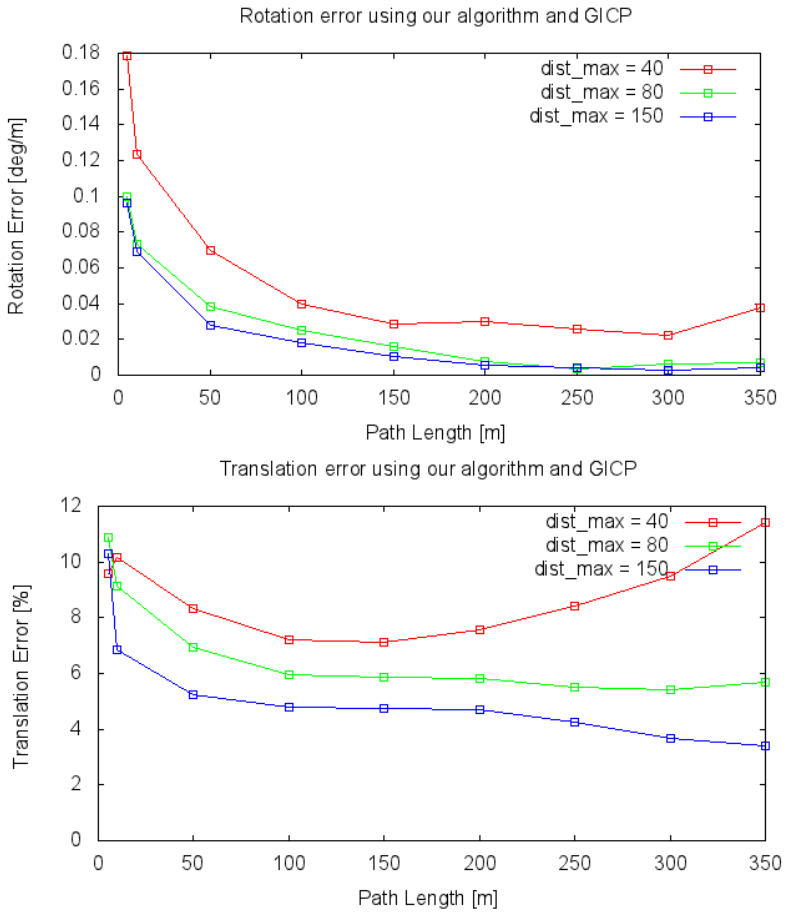

Figure 5.10: Errors obtained for different clipping distances and TS_GICP registration algorithm using *sequence 04* data

- Proposed odometry algorithm with GICP as registration algorithm (TS_GICP)
- Proposed odometry algorithm with NDT as registration algorithm (TS_NDT)

A KF was used for pose filtering after both traditional registration algorithms in order to equally compare them with our algorithm. All four odometry algorithms were tested in the three different sequences previously described: *sequence 04* for lot of environment movement, *sequence 03* for some environment movement, *sequence 05* for static environment. The disparity information was obtained using ELAS, as analyzed in subsubsection 5.3.3.1, and it was common for every algorithm. The maximum clipping distance used was 150, as suggested in subsubsection 5.3.3.2

Figure 5.11 shows the three charts describing the accuracy of each algorithm for the *sequence 04*. The first one, displaying the paths obtained, shows that the only

algorithm able to maintain the straight line is the TS_GICP. It also obtains a very low rotation error, virtually zero, and low translation error of approximately 3.4%. Both NDT and TS_NDT performs similarly, one obtaining less rotation error and the other a bit less translation error, with 5.2% and 6.4% respectively. In this case, the NDT algorithm does not benefit much of the algorithm herein proposed. Finally, clearly the worst performer is the GICP by itself, obtaining around 13.4% translation error.

The analysis for the *sequence 03* is shown in Figure 5.12. Again, the algorithm with the best accuracy is the TS_GICP, obtaining a final translation error of 8.6% at 250m. Notice the benefits of using the transformation segmentation in the GICP case, lowering the error a 6.9%. The NDT and TS_NDT cases behave more erratically, TS_NDT performing better on the first part of the trajectory and NDT performing better on the last part. However, both perform worse than the TS_GICP.

Finally, the behavior of the algorithms is tested in a static environment, the *sequence 05*. Results are shown in Figure 5.13. In this case, the contribution of the transformation segmentation is not as important as in the other two previous tests. The translation errors show two different tendencies, the GICP and the NDT, appearing little differences between the TS and traditional versions. GICP algorithms outperform the NDT ones, in rotation and translation errors, having around 8% error the GICP group compared to the 20% of the NDT group. The most accurate is the TS_GICP with a 6.7% error, reducing a 1.84% the translation error. As expected, the usage of the transformation segmentation in the static case is not as relevant as in the scenes with movement.

Generally, the results show that using the transformation segmentation algorithm herein proposed, impacts positively when used with the GICP registration algorithm. This configuration, named TS_GICP obtained the best results in every sequence tested compared to GICP alone and the NDT algorithms. However, using the transformation segmentation with the NDT algorithm has not shown the same results. No real benefit has been obtained using the transformation segmentation with the NDT.

5.3.3.4 Mapping

Finally, the environment global map can be reconstructed using the transformations estimated by the odometry algorithm and their supporting point-cloud clusters. One

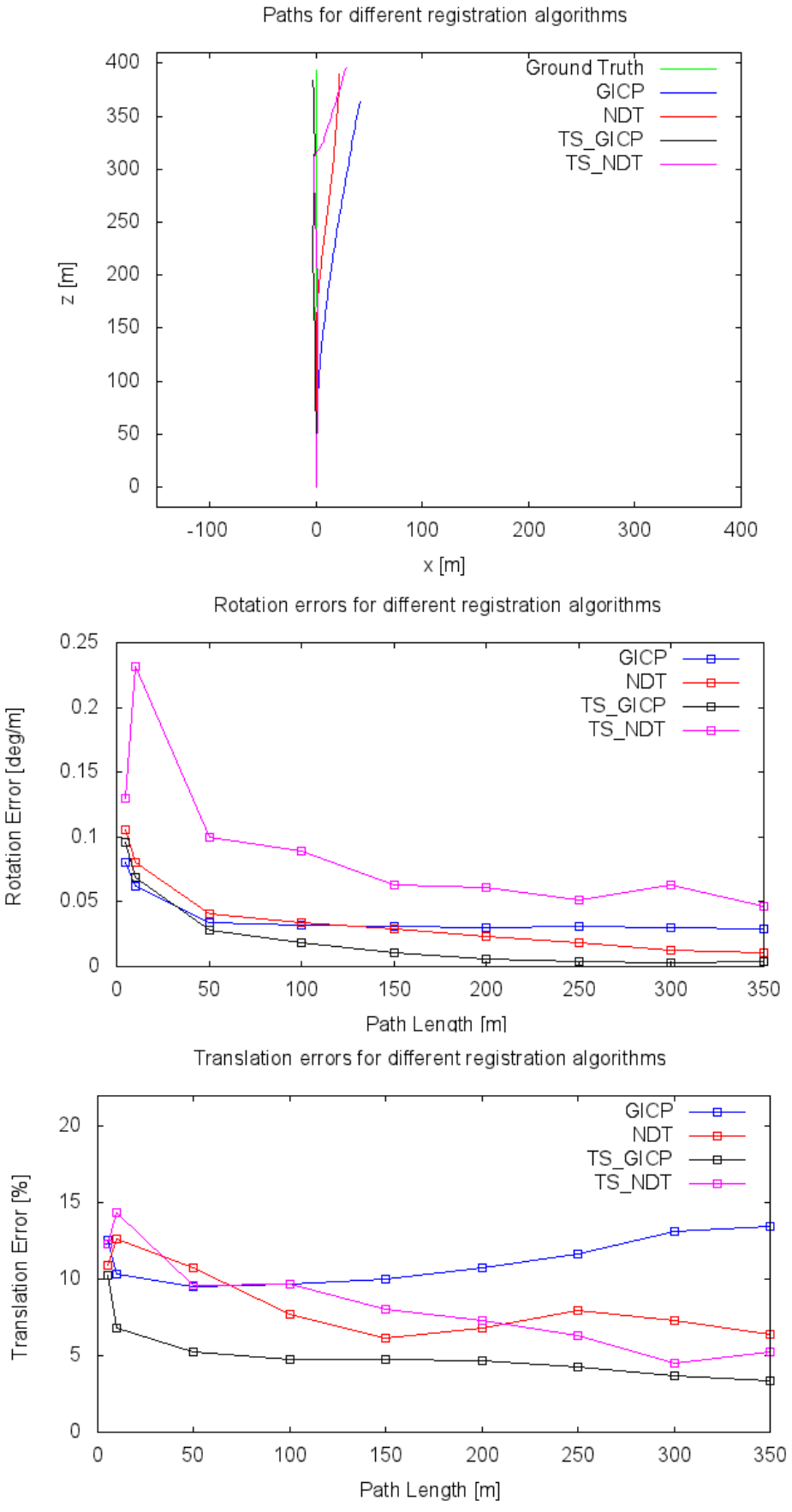

Figure 5.11: Errors comparison using different registration algorithms for the 04 sequence

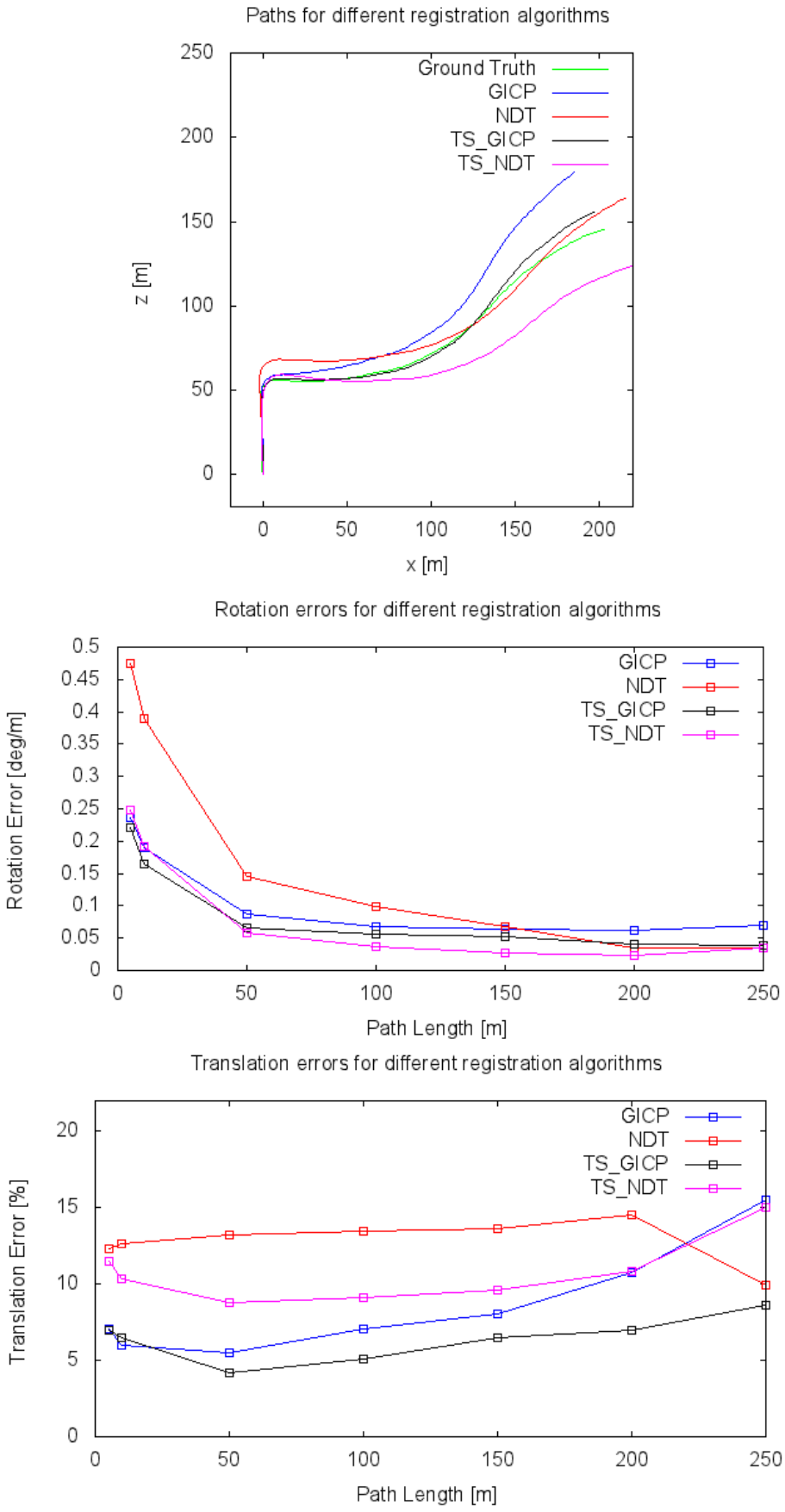

Figure 5.12: Errors comparison using different registration algorithms for the 03 sequence

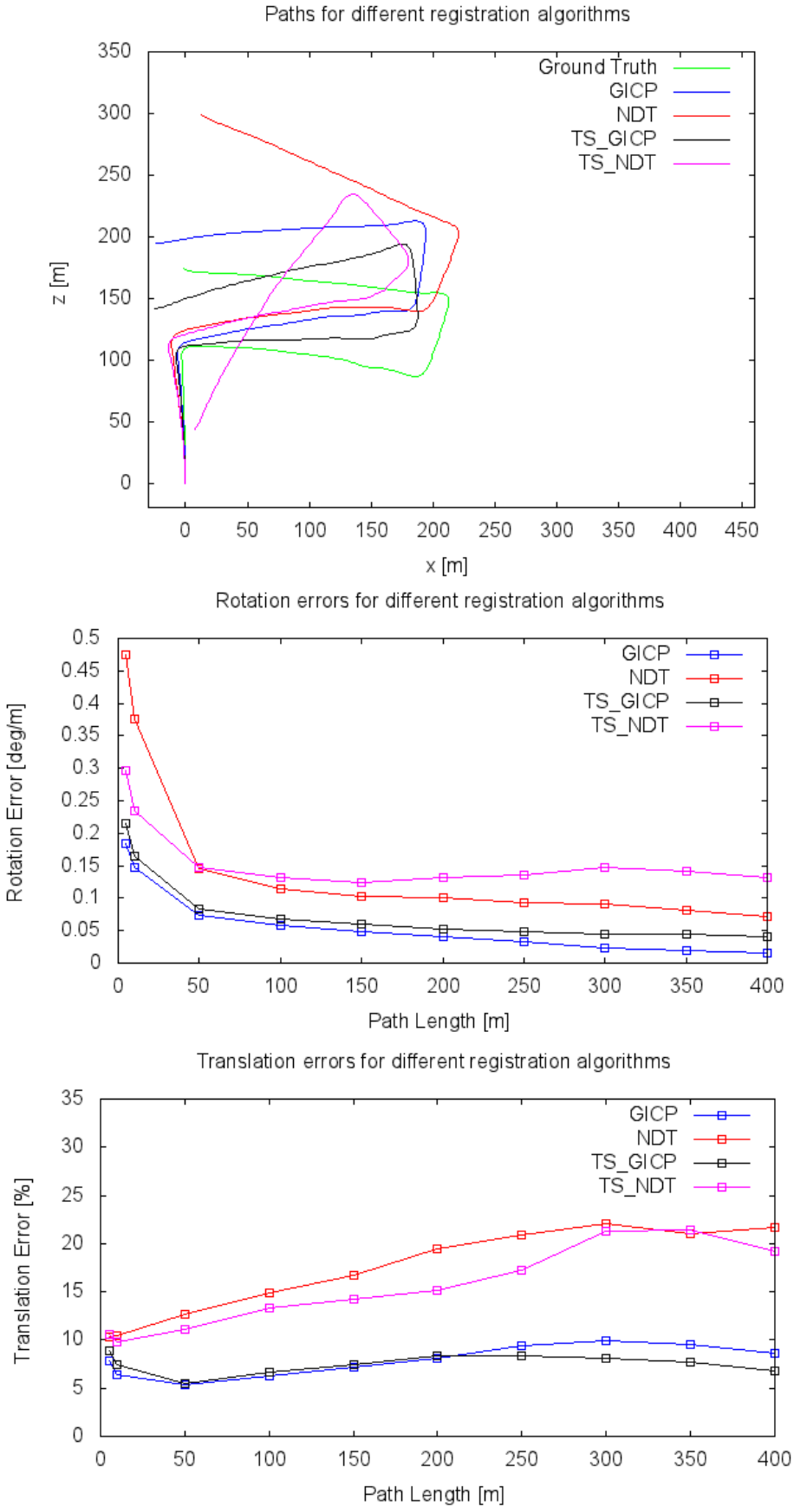

Figure 5.13: Errors comparison using different registration algorithms for the 05 sequence

of the main advantages of using the transformation segmentation algorithm for mapping the environment is that it determines which parts of the original point-clouds belong to the static environment, preventing the inclusion of objects that are actually moving. Therefore, the *shadow effect*, that is present in most of the reconstruction algorithms when non-static environments are captured, is mostly avoided. Given that the closer points to the capturing device are the most accurate, a distance limit of 30m from the camera was used for the mapping.

Figure 5.14 shows the three-dimensional reconstruction achieved for the *sequence 03*. Three different renderings of the global map are presented. Note that the global point-cloud obtained is free of outliers and free of moving objects that originally appear in the scene. By instance, in the original sequence, after the crossing a car continuously drives in front of the camera. If no segmentation of the point-cloud is performed, the car would appear like a shadow in the final global map. In this sense, the algorithm successfully removes the noise that moving objects would provoke in the final global map.

The same results are shown for *sequence 04* in Figure 5.15. The original scene contains lots of cars driving through the street, but the final map removes them all.

5.4 Conclusions

In this chapter a new visual odometry algorithm for stereo cameras has been proposed. Instead of using feature-based tracking odometry techniques, registration algorithms traditionally used for laser-scanner data were used. However, the presence of non-static environments and bad disparity estimations during stereo usually lead to bad pose estimations. A transformation segmentation algorithm has been proposed to deal with separating rigid bodies that appear in the environment with different trajectories and velocities. Using J-linkage for clustering purposes, the segmentation algorithm guarantees that the points belonging to the same cluster were sampled form the same rigid body and a transformation that explains the movement from one frame to another can be estimated.

The algorithm has been tested using the KITTI odometry dataset. Three environments with different quantity of moving objects were selected for testing. Firstly, a comparison of the stereo algorithms used for obtaining the point-clouds was performed. As expected, the most accurate stereo algorithms found in the literature

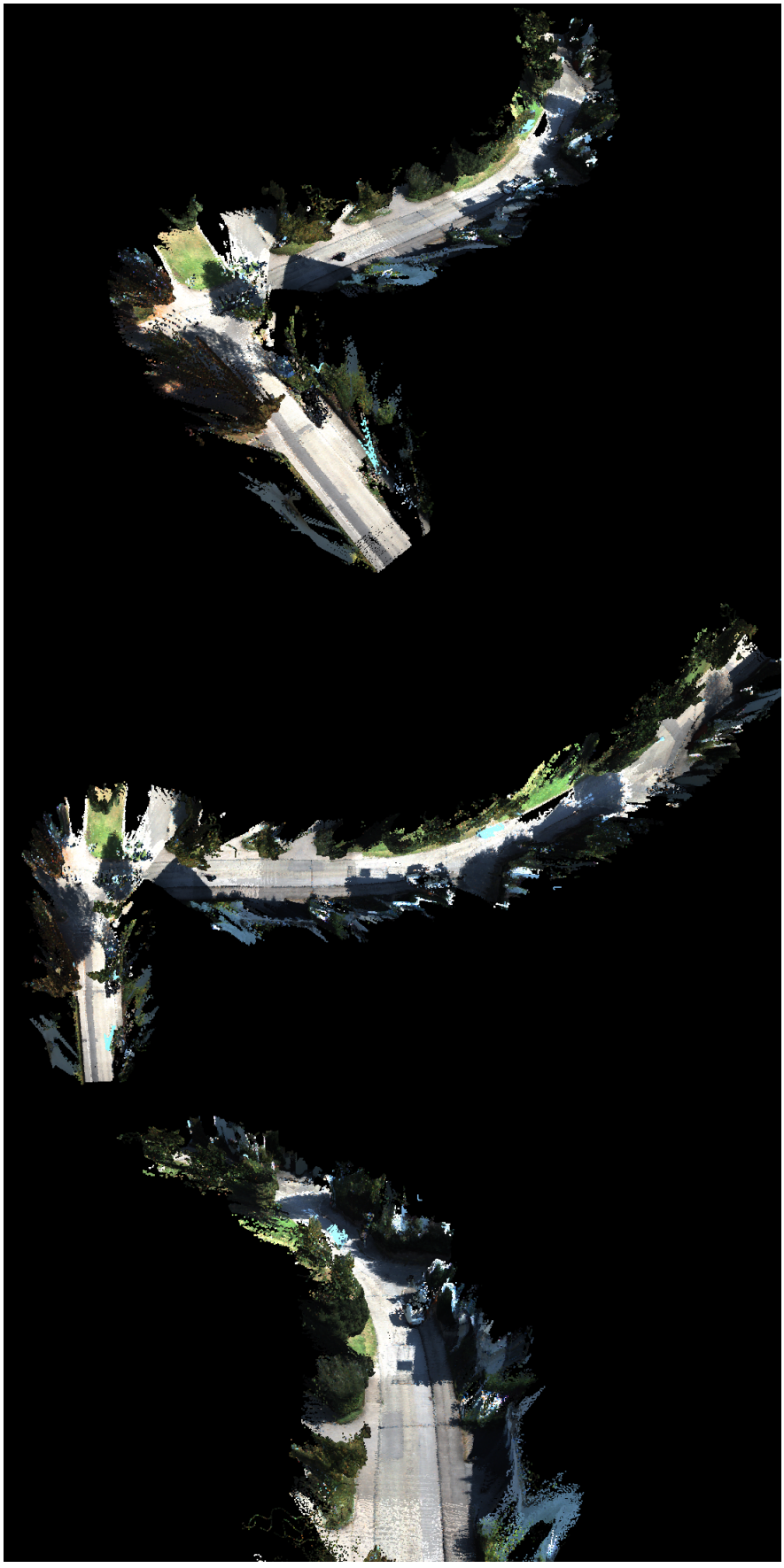

Figure 5.14: 3D Reconstruction of the 03 sequence

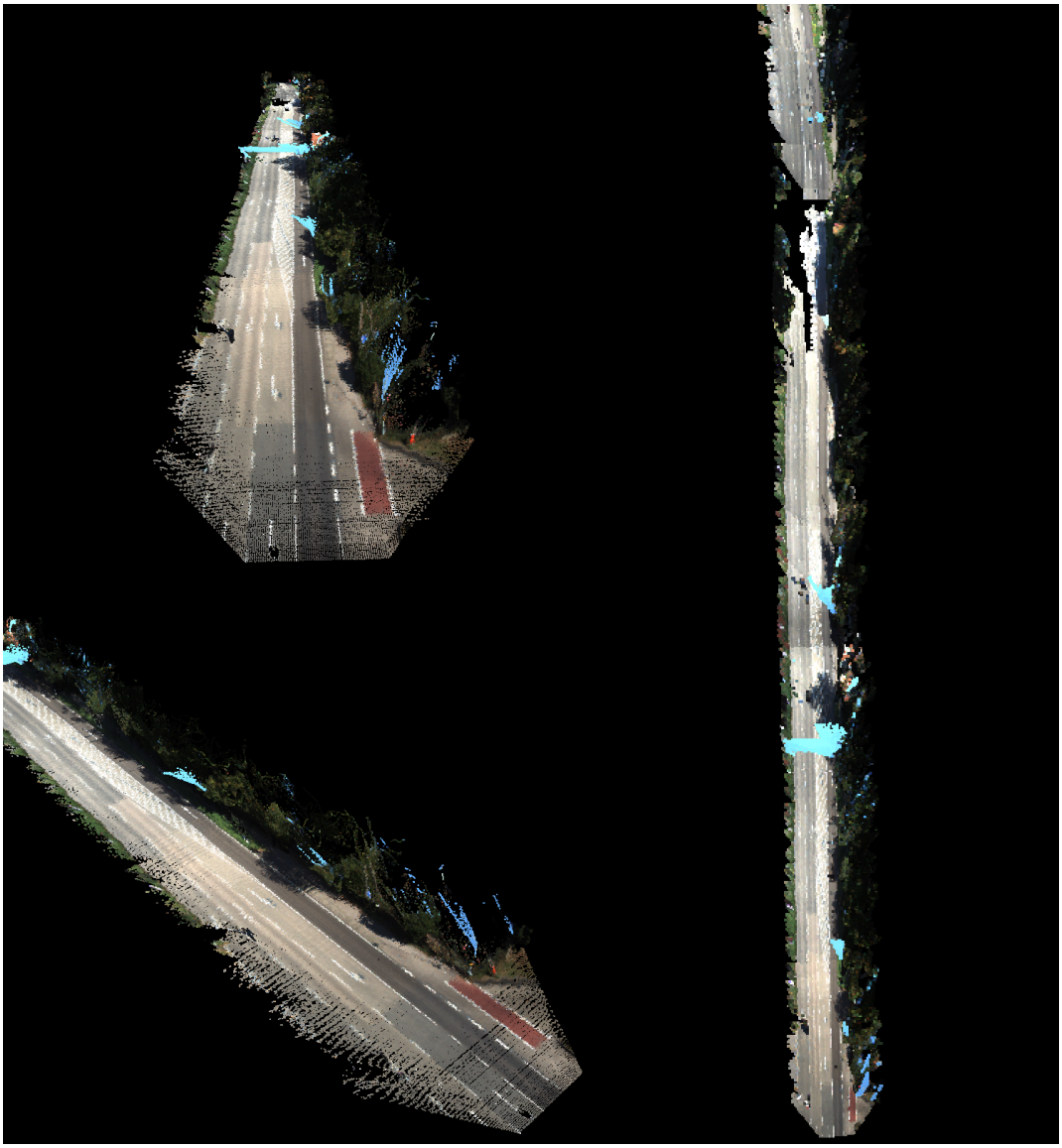

Figure 5.15: 3D Reconstruction of the 04 sequence

obtain the best odometry estimations. Using the ELAS (Geiger et al., 2011a) algorithm resulted in the most accurate motion estimation, followed by the genetic algorithm presented in chapter 4 and the global MRF algorithms. As expected, traditional local stereo solution obtained the worst odometry estimation.

Then, a comparison between the traditional GICP and NDT algorithms with the transformation segmentation ones using GICP and NDT, called TS_GICP and TS_NDT respectively, was made. For the test in non-static environments, the results showed that the transformation segmentation algorithm enhanced the performance of the GICP algorithm, demonstrating that the TS_GICP algorithms obtained the best results compared to any other registration algorithm. This supports the idea that segmenting moving objects and removing them during the motion estimation process has important benefits. However, the TS_NDT algorithm does not outperforms its traditional version, the NDT. This is supported by the idea discussed in (Magnusson et al., 2009), where NDT showed a stronger convergence capability starting from larger initial poses but also showed some unpredictability in the failures.

Finally, analyzing the behavior of the TS_GICP and TS_NDT algorithms in a static environment also showed some benefits. This is explained because wrongly estimated surfaces during the stereo process are segmented and removed given that they do not satisfy the movement restrictions. However, the differences are not so significant compared to the moving environments.

The transformation segmentation algorithm was also used for mapping purposes. Given that a clustering of the environment is performed, excluding moving objects, an accurate mapping without the classical *shadow effect* could be achieved.

6 Conclusions and future works

Along this document, a complete Visual Odometry system has been presented. Starting with the analysis and taxonomy of similar systems found in the literature, a global view of the problem and existing solutions was shown. Afterwards, as the starting point of the *motion estimation* system, an improvement in stereo global algorithms performance was proposed using a multi-resolution energy minimization framework. A refinement process for stereo matching is proposed using a genetic algorithm implemented in a parallel processing unit in CUDA. Finally, a Visual Odometry algorithm capable of segmenting moving objects in non-static environments and perform an accurate metric mapping was proposed.

This final chapter will focus on outlining the most important conclusions drawn and on detailing the original contributions yielded by this thesis. Some further developments are also noted at the end of this chapter.

6.1 Conclusions

The document started by making a brief introduction to the SLAM problem and outlining similar systems and the main tendencies recently found in the literature. A classification of the SLAM systems into three different groups was performed: *filter*, *bundle adjustment* and *visual odometry* based systems. It was shown that some of the basic steps, such as the *map initialization* and the *feature detection and description* steps, were shared by these groups. Three tables were presented, one for each group, as an overview of the most important systems, each one detailing the approaches

taken for each step that builds the *motion estimation* algorithm.

A new multi-resolution energy minimization framework for stereo matching was presented afterwards. This framework provided the option of reducing the size of the MRF problem in order to enhance its performance. New parameters for controlling this complexity reduction were included to the classic energy minimization framework. The impact of these parameters was first discussed theoretically and later confirmed empirically. The main conclusion drawn by this work is that a complexity reduction proportional to R^2, being R the reduction factor on the input image resolution, can be achieved while maintaining the resolution in the disparity domain. Moreover, traditional graph-cuts algorithms, can still be used for solving this MRF problem. However, reducing the size of the MRF was shown to always involve an increment on both RMS and B error, thus a trade-off between accuracy and performance was recommended.

A genetic algorithm was proposed for the stereo refinement problem. The main benefit of using evolutionary algorithms is that the form of energy function is not restricted, in comparison to other global optimizers. This characteristic leads to the possibility of more complex functions for describing the stereo matching problem. On the one hand, the genetic algorithm proposed in this thesis uses a new crossover and mutation operators accounting for the occlusion phenomena. For this purpose, both left and right disparity maps are estimated. On the other hand, two new energy functions with occlusion and discontinuity handling were proposed: one with a fronto-parallel smoothing term formulation and the second one with a second order prior formulation.

The implementation using the fronto-parallel energy function outperformed the classic one in 2.75% of bad-pixels on average, which represents a 32% of error reduction using the new energy function. Besides, the evolution of the bad-pixels error measurement and the evolution of each energy function suggests that the new formulation is more suitable for representing real disparities. As far as we know, is the first evolutionary algorithm included in the Middlebury ranking table.

Finally, a visual odometry and mapping algorithm for stereo cameras was presented. Using registration algorithms, traditionally used for laser-scanner data, a new transformation segmentation algorithm was proposed for dealing with non-static environments. The aim of the transformation segmentation was to separate rigid bodies which appear in the environment with different velocities and trajectories. For this task, RANSAC and the J-linkage clustering algorithm were used. The clustering

process guarantees that the points belonging to the same cluster were sampled from the same rigid body and that a transformation explaining the movement from one frame to another can be estimated. Note that two rigid bodies with same velocity and trajectory will be grouped in the same cluster.

The transformation segmentation algorithm was tested in real scenarios using the KITTI odometry dataset. The scenarios involved images taken in a car while driving through different environments. The system was able to estimate the elements belonging to the static environment such as the road, nearby buildings, trees and other parked cars, and to separate it from other moving elements such as cars driving in the opposite direction and cars driving in the same lane. As a result, given that other moving elements were filtered, a better estimation of the camera's motion and a mapping, only including the static environment, could be achieved.

6.2 Contributions

This thesis has provided several original contributions to the stereo vision and motion estimation research fields. These contributions are detailed below:

i A taxonomy of the different *motion estimation* algorithms found in the literature, and an analysis and classification of each of their steps has been presented. This information is grouped in three different tables, each one describing the steps for each *motion estimation* algorithm group.

ii A new multiresolution energy minimization framework for stereo matching has been proposed. The main characteristic of this framework is the reduction of the computational complexity of the stereo matching problem in an *Markov Random Field* while maintaining the resolution in the disparity domain and the error measures bounded.

iii An evolutionary algorithm has been proposed for the stereo correspondence and refinement problems. The algorithm is used for minimizing a *Markov Random Field* energy function, similarly to other global methods found in the literature. The principal contribution of this work is providing an algorithm able to cope with unrestricted types of energy functions (although it is commonly known, finding minimum configurations is not guaranteed).

iv A visual odometry algorithm, based on registration methods, for stereo cameras has been proposed. The contributions made in this field are twofold. Firstly, registration algorithms have demonstrated to be also usable with stereo cameras (traditionally were used with laser range sensors and RGB-D cameras). Secondly, a transformation segmentation algorithm is proposed for dealing with non-static scenes, separating different moving rigid bodies.

Other contributions made during the development of this thesis are:

i New functions for the *Markov Random Fields* energy minimization method for the stereo matching problem were proposed. Two new energy functions accounting for occluded pixels handling in the formulation and treating depth discontinuities on pixels with high photometric derivatives have been proposed and analyzed. The first one with a fronto-parallel smoothing term formulation and the second one with a second order prior formulation.

ii A parallel implementation of the proposed evolutionary algorithm was described and analyzed. Each operator included in the genetic algorithm had its correspondent parallel kernel operator for parallel processing. As a conclusion, a speedup of around 40x could be achieved using a *CUDA* implementation in a Geforce 590 consumer graphics card.

iii Given that the transformation segmentation algorithm is able to separate moving objects from the static environment, an accurate mapping algorithm for non-static environments was proposed. Mapping a city environment using stereo vision was achieved even though other cars were driving around the capturing device, occluding some areas that afterwards were visible.

6.3 Further developments

This thesis has presented a complete visual odometry system starting with the stereo algorithm and finishing with the visual odometry algorithm itself. However, some possible improvements are pointed out below.

6.3.1 TSDF for surface representation

Along chapter 5, point-clouds were used to perform scan alignment between depth estimations obtained from stereo matching. Once aligned, this depth measures can be used to produce a fused representation of the scene beyond simply overlapping scans. The *Truncated Signed Distance Function* (TSDF) (Curless and Levoy, 1996) was used in other works such as (Izadi et al., 2011) and has demonstrated to be very useful for real-time photo-realistic rendering compared to other probabilistic techniques such as probabilistic occupancy grids. Therefore, it is proposed, similarly to the work in (Izadi et al., 2011), to maintain a dense reconstruction of the scene using TSDF not only for visualization purposes, but also being able to perform frame-to-model tracking instead of pairwise frame-to-frame visual odometry.

6.3.2 Use different stereo algorithms

The visual odometry algorithm presented in chapter 5 demonstrated to be very sensible to the stereo algorithm used. This result is reasonable considering that the registration algorithms used are very sensitive to the initial accuracy of the input point-clouds. Therefore, the use of new and more accurate stereo algorithms can improve the overall system's performance. It is proposed to use algorithms such as (M. Antunes, 2014) (M. Antunes, 2012) to create accurate point-clouds and analyze its impact on the visual odometry accuracy.

6.3.3 Analyze the real-time viability of the Transformation Segmentation algorithm

The Transformation segmentation algorithm presented in this thesis was implemented with no performance optimization in mind. Computing times were very far from being usable for real-time applications. In the literature, it has been demonstrated that registration algorithms can be used in real-time applications if they are optimally implemented by exploiting parallelism in GPU hardware (Izadi et al., 2011). Considering the advances that the GPU hardware computation has attained during the last years, it would be interesting to analyze if the Transformation segmentation algorithm could be ported to GPU and used in real-time applications.

Bibliography

Agrawal, M. and Konolige, K. (2006). *Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS*. author:.

Alahari, K., Kohli, P., and Torr, P. H. S. (2010). Dynamic hybrid algorithms for map inference in discrete mrfs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1846–1857.

Alcantarilla, P. F., Bergasa, L. M., and Dellaert, F. (2010). Visual odometry priors for robust EKF-SLAM. In *ICRA*, page 3501âĂŞ3506, Anchorage, AK, USA.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698âĂŞ700.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *In ECCV*, pages 404–417.

Beall, C., Lawrence, B., Ila, V., and Dellaert, F. (2010). 3D reconstruction of underwater structures. In *IROS*, pages 4418–4423.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, B-48:259–302.

Besl, P. and McKay, N. D. (1992). A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.

Biber, P. and Strasser, W. (2003). The normal distributions transform: a new approach to laser scan matching. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2743–2748 vol.3.

Birchfield, S. and Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20; 20(4):401–406.

Bobick, A. F. and Intille, S. S. (1999). Large occlusion stereo. *International Journal of Computer Vision*, 33:181–200.

Bonin-Font, F., Ortiz, A., and Oliver, G. (2008). Visual navigation for mobile robots: A survey. *Jurnal of Intelligent & Robotic Systems*, 53:263–296.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124 –1137.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 23(11):1222–1239.

Chang, C. and Chatterjee, S. (1990). Multiresolution stereo by simulated annealing. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 885–890 vol.2.

Chen, Y. and Medioni, G. (1991). Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729 vol.3.

Chiuso, A., Favaro, P., Jin, H., and Soatto, S. (2002). Structure from motion causally integrated over time. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):523–535.

Civera, J., Davison, A. J., and Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. *Robotics, IEEE Transactions on*, 24(5):932–945. ID: 1.

Comport, A. I., Malis, E., and Rives, P. (2007). Accurate quadrifocal tracking for robust 3D visual odometry. In *ICRA*.

Cornelis, N., Leibe, B., Cornelis, K., and Gool, L. V. (2008). 3d urban scene modeling integrating recognition and reconstruction. *International Journal Of Computer Vision*, 78(2-3):121–141.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA. ACM.

Dai, C., Wu, X., and Liu, J. (2008). Stereo matching using adaptive genetic algorithm. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1225–1228.

Davison, A. J. (1998). *Mobile Robot Navigation Using Active Vision.* PhD thesis, University of Oxford.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1403, Washington, DC, USA. IEEE Computer Society.

Davison, A. J., Cid, Y. G., and Kita, N. (2004). Real-time 3D SLAM with wide-angle vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon.*

Davison, A. J. and Murray, D. W. (2002). Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):865–880. ID: 1.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: real-time single camera SLAM. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067. ID: 1.

Deans, M. and Hebert, M. (2000). Experimental comparison of techniques for localization and mapping using a bearings only sensor. In *Proc. of the ISER '00 Seventh International Symposium on Experimental Robotics.*

Dellaert, F. and Kaess, M. (2006). Square root SAM: simultaneous localization and mapping via square root information smoothing. *Int. J. Rob. Res.*, 25(12):1181âĂŞ1203.

Desouza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267. ID: 1.

Druon, S., Aldon, M.-J., and Crosnier, A. (2006). Color constrained ICP for registration of large unstructured 3D color data sets. In *Information Acquisition, 2006 IEEE International Conference on*, pages 249–255.

Eade, E. and Drummond, T. (2006). Scalable monocular SLAM. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476. ID: 1.

Engels, C., StewÃľnius, H., and NistÃľr, D. (2006). Bundle adjustment rules. In *In Photogrammetric Computer Vision*.

Fiala, M. and Ufkes, A. (2011). Visual odometry using 3-dimensional video input. In *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pages 86–93.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun.ACM*, 24(6):381–395.

Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry : Part II: matching, robustness, optimization, and applications. *Robotics Automation Magazine, IEEE*, 19(2):78–90.

Furukawa, Y. and Ponce, J. (2008). Accurate camera calibration from multi-view stereo and bundle adjustment. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, Providence, USA.

Geiger, A., Roser, M., and Urtasun, R. (2011a). Efficient large-scale stereo matching. In Kimmel, R., Klette, R., and Sugimoto, A., editors, *Computer Vision âĂŞ ACCV 2010*, volume 6492 of *Lecture Notes in Computer Science*, pages 25–38. Springer Berlin Heidelberg.

Geiger, A., Ziegler, J., and Stiller, C. (2011b). StereoScan: dense 3d reconstruction in real-time. In *IV*, Baden-Baden, Germany.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741.

Godin, G., Rioux, M., and Baribeau, R. (1994). Three-dimensional registration using range and intensity information. pages 279–290.

Gong, M. and Yang, Y.-H. (2001). Multi-resolution stereo matching using genetic algorithm. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 21–29.

Gong, M. and Yang, Y.-H. (2002). Genetic-based stereo algorithm and disparity map evaluation. *International Journal of Computer Vision*, 47(1):63–77.

Han, K.-P., Song, K.-W., Chung, E.-Y., Cho, S.-J., and Ha, Y.-H. (2001). Stereo matching using genetic algorithm with adaptive chromosomes. *Pattern Recognition*, 34(9):1729–1740.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hirschmüller, H., Innocent, P. R., and Garibaldi, J. M. (2002). Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. *International Conference on Control, Automation, Robotics and Vision*.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. ID: 1.

Huhle, B., Magnusson, M., Strasser, W., and Lilienthal, A. (2008). Registration of colored 3D point clouds with a kernel-based extension to the normal distributions transform. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 4025–4030.

Iocchi, L. and Konolige, K. (1998). A multiresolution stereo vision system for mobile robots.

Issa, H., Ruichek, Y., and Postaire, J. G. (2002). Stereo correspondence using a genetic scheme with a new solution encoding. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 6, page 5 pp. vol.6.

Izadi, S., Newcombe, R. A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A. J., and Fitzgibbon, A. (2011). KinectFusion: real-time dynamic 3D surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks*, SIGGRAPH '11, page 23:1âĂŞ23:1, New York, NY, USA. ACM.

Joung, J., An, K. H., Kang, J. W., Chung, M.-J., and Yu, W. (2009). 3D environment reconstruction using modified color ICP algorithm by fusion of a camera and a 3D laser range finder. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3082–3088.

Juan, O. and Boykov, Y. (2006). Active graph cuts. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1023–1029.

Kaess, M., Ni, K., and Dellaert, F. (2009). Flow separation for fast and robust stereo odometry. In *ICRA*, Kobe, Japan.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME âĂŞ Journal of Basic Engineering*, (82 (Series D)):35–45.

Klaus, A., Sormann, M., and Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2969–2976.

Kohli, P. and Torr, P. H. S. (2007). Dynamic graph cuts for efficient inference in markov random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):2079–2088.

Kolmogorov, V. and Zabin, R. (2004). What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159.

KÃijmmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. (2009). On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407.

Li, C., Xue, J., Du, S., and Zheng, N. (2010). A fast multi-resolution iterative closest point algorithm. In *Pattern Recognition (CCPR), 2010 Chinese Conference on*, pages 1–5.

Liang, C.-K., Cheng, C.-C., Lai, Y.-C., Chen, L.-G., and Chen, H. H. (2009). Hardware-efficient belief propagation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 80–87.

Liying, W. and WeiDong, S. (2009). A review of range image registration methods with accuracy evaluation. In *Urban Remote Sensing Event, 2009 Joint*, pages 1–8.

Longuet-Higgins, H. C. (1987). A computer algorithm for reconstructing a scene from two projections. Readings in computer vision: issues, problems, principles, and paradigms, pages 61–62. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157.

M. Antunes, J. B. (2012). Semi-dense piecewise planar stereo reconstruction using SymStereo and PEARL.

M. Antunes, J. B. (2014). SymStereo: Stereo matching using induced symmetry. *International Journal Of Computer Vision (IJCV)*.

Magnusson, M. (2009). *The Three-Dimensional Normal-Distributions Transform âĂŤ an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, ÃŰrebro University. ÃŰrebro Studies in Technology 36.

Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, page 803âĂŞ827.

Magnusson, M., Nuchter, A., Lorken, C., Lilienthal, A., and Hertzberg, J. (2009). Evaluation of 3D registration reliability and speed - a comparison of ICP and NDT. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3907–3912.

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics, Special Issue on Space Robotics*, 24:2007.

Manessis, A., Hilton, A., Palmer, P., McLauchlan, P., and Shen, X. (2000). Reconstruction of scene models from sparse 3D structure. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 666–671 vol.2. ID: 1.

Marr, D. and Poggio, T. (1976). Cooperative computation of stereo disparity. Technical report, Massachusetts Institute of Technology.

Maybeck, P. S. and Siouris, G. M. (1980). Stochastic models, estimation, and control, volume i. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(5):282.

McLauchlan, P. F. and Murray, D. W. (1995). A unifying framework for structure and motion recovery from image sequences. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 314–320. ID: 1.

Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., and Zhang, X. (2011). On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 467 –474.

Middlebury (2002). http://vision.middlebury.edu/stereo/.

Milstein, A., McGill, M., Wiley, T., Salleh, R., and Sammut, C. (2011). Occupancy voxel metric based iterative closest point for position tracking in 3D environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4048–4053.

Molton, N., Davison, A., and Reid, I. (2004). Locally planar patch features for real-time structure from motion. In *BMVC04*, pages –yy.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: a factored solution to the simultaneous localization and mapping problem. In

Eighteenth national conference on Artificial intelligence, pages 593–598, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Montemerlo, M., Thrun, S., Roller, D., and Wegbreit, B. (2003). FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1151–1156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Mooij, J. M. (2010). libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173.

Moravec, H. P. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford, CA, USA. AAI8024717.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real time localization and 3D reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370.

Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. *Conference on Computer Vision and Pattern Recognition, IEEE Computer Society*.

Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., and Fitzgibbon, A. (2011). KinectFusion: real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136.

Nie, D.-H., Han, K.-P., and Lee, H.-S. (2009). Stereo matching algorithm using population-based incremental learning on gpu. In *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, pages 1–4.

Nister, D. (2003). Preemptive RANSAC for live structure and motion estimation. *International Conference on Computer Vision*.

Nister, D. (2004a). An efficient solution to the five-point relative pose problem. *Ieee Transactions On Pattern Analysis And Machine Intelligence*, 26(6):756–777.

Nister, D. (2004b). A minimal solution to the generalised 3-point pose problem. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–560–I–567 Vol.1.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652–I–659 Vol.1.

Paz, L. M., Pinies, P., Tardos, J. D., and Neira, J. (2008). Large-scale 6-DOF SLAM with stereo-in-hand. *Robotics, IEEE Transactions on*, 24(5):946–957.

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, Pittsburgh, PA.

Pollefeys, M., Nister, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal Of Computer Vision*, 78(2-3):143–167.

Rogmans, S., Lu, J., Bekaert, P., and Lafruit, G. (2009). Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity gpus. *Image Commun.*, 24(1-2):49–64.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, volume 1, pages 430–443.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927 – 941. <ce:title>Semantic Knowledge in Robotics</ce:title>.

Saito, H. and Mori, M. (1995). Application of genetic algorithms to stereo matching of images. *Pattern Recognition Letters*, 16(8):815–821.

Salmen, J., Schlipsing, M., Edelbrunner, J., Hegemann, S., and Luke, S. (2009). Real-time stereo vision: Making more out of dynamic programming. In *CAIP*, pages 1096–1103.

Salvi, J., Matabosch, C., Fofi, D., and Forest, J. (2007). A review of recent range image registration methods with accuracy evaluation. *Image Vision Comput.*, 25(5):578âĂŞ596.

Satorre, R., CompaÃś, P., BotÃŋa, A., and Rizo, R. (2002). Multiresolution scheme for stereo correspondence using correlation techniques. In *Proceedings of the 12th Portuguese Conference on Pattern Recognition*.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *Robotics Automation Magazine, IEEE*, 18(4):80–92.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal Of Computer Vision*, 47(1-3):7–42.

Schleicher, D., Bergasa, L. M., Barea, R., Lopez, E., and Ocana, M. (2006). Real-time simultaneous localization and mapping using a wide-angle stereo camera. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, pages 55–60. ID: 1.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-ICP. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600. ID: 1.

Sim, R., Elinas, P., Griffin, M., Shyr, A., and Little, J. J. (2006). Design and analysis of a framework for real-time vision-based SLAM using rao-blackwellised particle filters. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 21–21. ID: 1.

StewÃľnius, H., Engels, C., and NistÃľr, D. (2006). Recent developments on direct relative orientation. *Journal of Photogrammetry and Remote Sensing*.

Strasdat, H., M, M. J. M., and Davison, A. J. (2010). Real-time monocular SLAM: why filter? *IEEE International Conference on Robotics and Automation*.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for markov random fields.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30; 30(6):1068–1080.

Tappen, M. F. and Freeman, W. T. (2003). Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 900, Washington, DC, USA. IEEE Computer Society.

Toldo, R. and Fusiello, A. (2008). Robust multiple structures estimation with j-linkage. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 537–547, Berlin, Heidelberg. Springer-Verlag.

Toldo, R. and Fusiello, A. (2010). Real-time incremental j-linkage for robust multiple structures estimation. In *3DPVT*.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846.

Tomono, M. (2007). Monocular SLAM using a rao-blackwellised particle filter with exhaustive pose space search. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2421–2426. ID: 1.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment âĂŤ a modern synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer Berlin Heidelberg.

Veksler, O. (2005). Stereo correspondence by dynamic programming on a tree.

Vineet, V. and Narayanan, P. J. (2008). Cuda cuts: Fast graph cuts on the gpu. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8.

Wainwright, M., Jaakkola, T., and Willsky, A. (2002). Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *IEEE Transactions on Information Theory*, 51:3697–3717.

Wang, B., Chung, R., and Shen, C.-L. (2003). Genetic algorithm-based stereo vision with no block-partitioning of input images. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 2, pages 830–836 vol.2.

Wang, L., Gong, M., Gong, M., and Yang, R. (2006a). How far can we go with local optimization in real-time stereo matching. *3D Data Processing Visualization and Transmission, International Symposium on*, pages 129–136.

Wang, L., Jin, H., and Yang, R. (2008). Search space reduction for mrf stereo. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 576–588, Berlin, Heidelberg. Springer-Verlag.

Wang, L., Liao, M., Gong, M., Yang, R., and Nister, D. (2006b). High-quality real-time stereo using adaptive cost aggregation and dynamic programming.

Wang, Z.-F. and Zheng, Z.-G. (2008). A region based stereo matching algorithm using cooperative optimization.

Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2009). Global stereo reconstruction under second-order smoothness priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2115–2128.

Yamaguchi, K., Hazan, T., McAllester, D., and Urtasun, R. (2012). Continuous markov random fields for robust stereo estimation. In *ECCV*.

Yan, J., Guorong, L., Shenghua, L., and Lian, Z. (2009). A review on localization and mapping algorithm based on extended kalman filtering. In *Information Technology and Applications, 2009. IFITA '09. International Forum on*, volume 2, pages 435–440. ID: 1.

Yang, Q., 0002, L. W., Yang, R., Wang, S., Liao, M., and NistÃľr, D. (2006). Real-time global stereo matching using hierarchical belief propagation. In Chantler, M. J., Fisher, R. B., and Trucco, E., editors, *BMVC*, pages 989–998. British Machine Vision Association.

Yang, Q. X., Wang, L., Yang, R. G., Stewenius, H., and Nister, D. (2009). Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *Ieee Transactions On Pattern Analysis And Machine Intelligence*, 31(3):492–504.

Yang, R. and Pollefeys, M. (2003a). Multi-resolution real-time stereo on commodity graphics hardware.

Yang, R. and Pollefeys, M. (2003b). Multi-resolution real-time stereo on commodity graphics hardware. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–211–I–217 vol.1.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2003). Understanding belief propagation and its generalizations. pages 239–269.

Yoon, K. J. and Kweon, I. S. (2006). Adaptive support-weight approach for correspondence search. *Ieee Transactions On Pattern Analysis And Machine Intelligence*, 28(4):650–656.

Yu, T., Lin, R.-S., Super, B., and Tang, B. (2007). Efficient message representations for belief propagation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

Zemerly, M. J., Holden, M., and Muller, J.-P. (1992). A multi-resolution approach to parallel stereo matching of airborne imaginery. In *Systems for Data Processing and Analysis*, volume Commission II, pages 350–357.

Zhan, Q., Liang, Y., and Xiao, Y. (2009). Color-based segmentation of point clouds. In *Laserscanning09*, volume XXXVIII, pages 248–252, Paris, France.

Zhang, Z., Hou, C., and Yang, J. (2009). A stereo matching algorithm based on genetic algorithm with propagation stratagem. In *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, pages 1–4.

Zhao, Y. and Taubin, G. (2011). Real-time stereo on gpgpu using progressive multi-resolution adaptive windows. pages 420–432.

Zitnick, Kang, and Sing (2007). Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65.