REAL-TIME IMAGE STITCHING FOR AUTOMOTIVE 360°VISION SYSTEMS

A degree's Thesis
Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

By

Alba Pujol Miró

In partial fulfillment
of the requirements for the degree in

AUDIOVISUAL ENGINEERING

Advisor: Javier Ruiz Hidalgo

Barcelona, July 2014

ABSTRACT

This document presents the development of a 360° system adapted to buses. From four cameras located around the vehicle, a bird's eye top view is created to get the driver the vehicle's surroundings information. The development has been focused on implementing a warping algorithm that produces the best alignment between camera images. This project analyses the state-of-the-art warping algorithms. Based on the results obtained, an adapted method to the specific 360° vision system is designed and implemented. In addition, this project also has tested different calibration patterns –both in 3D virtual and real environments– and a calibration pattern for the final application is proposed.

This project has been carried out in a multidisciplinary UPC team. The developments included in this project are part of the work done in the GPI (Image Processing Group) team. This team has been working in a 1-year project commissioned by the Arcol company. This project main goal is to develop a camera-and-display based guidance system for the bus driver.

RESUM

Aquest document presenta el desenvolupament d'un sistema de visió 360° adaptat a autobusos. A partir de quatre càmeres situades al voltant del vehicle es crea una image a vista d'ocell, per donar al conductor la informació del que està succeïnt al voltant del vehicle.El desenvolupament s'ha centrat en la implementació d'un algoritme de deormació que produeixi el millor alineament possible entre càmeres. Aquest projecte analitza els algoritmes aplicats actualment en deformació d'imatges. Basant-se en els resultats obtinguts, un mètode adaptat a aquest sistema de visió 360° és dissenyat i implementat. A més a més, en aquest projecte també s'analitzen diferents patrons de calibració (tant en entorns virtuals 3D com reals), i es proposa un patró de calibració per a l'aplicació final.

Aquest projecte s'ha desevolupat en un equip multidisciplinar de la UPC. Els desenvolupaments inclosos en aquest projecte són una part de la feina feta dins del GPI (Grup de Procesat d'Imatge). Aquest equip ha estat treballant en un projecte d'un any encarregat per l'empresa Arcol, l'objectiu principal del qual és desenvolupar un sistema de guiatge per als conductors d'autobusos basat en una pantalla i un seguit de càmeres.

RESUMEN

Este documento presenta el desenvolupamiento de un sistema de visión 360° adaptado a autobuses. A partir de cuatro cámaras situadas alrededor del vehículo se crea una imagen a vista de pájaro, para dar al conductor la información de lo que está sucediendo alrededor del vehículo.El desenvolupamiento se ha centrado en la implementación de un algoritmo de deormación que produzca el mejor alineamiento posible entre cámaras. Este proyecto analiza los algoritmos aplicados actualmente en deformación de imágenes. Basándose en los resultados obtenidos, un método adaptado a este sistema de visión 360 ° es diseñado e implementado. Además, en este proyecto también se analizan diferentes patrones de calibración (tanto en entornos virtuales 3D como reales), y se propone un patrón de calibración para la aplicación final.

Este proyecto se ha desarrollado en un equipo multidisciplinar de la UPC. Los desarrollos incluidos en este proyecto son una parte del trabajo hecho dentro del GPI (Grupo de Procesado de Imagen). Este equipo ha estado trabajando en un proyecto de un año encargado por la empresa Arcol, el objetivo principal del cual es desarrollar un sistema de guiado para los conductores de autobuses basado en una pantalla y una serie de cámaras.

"En contra de la seva voluntat,
omplo aquest full per incloure
una dedicatòria a Alex Barceló"
Alex Barceló parafrasejant Alba Pujol

ACKNOWLEDGEMENTS

I would like to thank to all the GPI team –Javier Ruiz, J. R. Casas, Albert Gil and Carles Ventura– for their support during this half year of work. Especially, I would like to thank Javier Ruiz for advising this TFG project and J.R. Casas for giving me the opportunity to work in this project.

També voldria agrair als meus pares la seva infinita paciència amb les llargues tardes davant l'ordinador, i la seva ajuda quan més falta feia. A la meva parella pel seu suport, tant matemà tic ccom del meu humor. I als meus companys de classe, amb els qui encara tenim unes braves pendents.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ADAS Advanced Driver Assistance Systems

AHA Advanced Hardware Architecture

CPU Central Processing Unit

CV Computer Vision

DAS Driver Assistance Systems

FPGA Field Programmable Gate Array

GPI Image Processing Group

GUI Graphic User Interface

MLS Moving Least Squares

OS Operating system

RAM Random Acces Memory

ROI Region Of Interest

SIFT Scale-Invariant Feature transform

SoC System-on-Chip

TFG Final Degree Project

CHAPTER 1

INTRODUCTION

Technological developments in Electronics and Signal Processing fields are also being implemented in the nowadays manufactured vehicles. A considerable number of the vehicle functions have moved to be controlled by electronic actuators. This is the case of, for example, fuel valves, brakes, door and windows opening/closing, etc. In addition, most vehicles are including an on-board computer that monitors all the vehicle parameters –speed, remaining fuel, range, etc.–, and inform the driver about any detected potential danger. All those systems that help the driver and made a safer driving are known as DAS, Driving Assistant Systems.

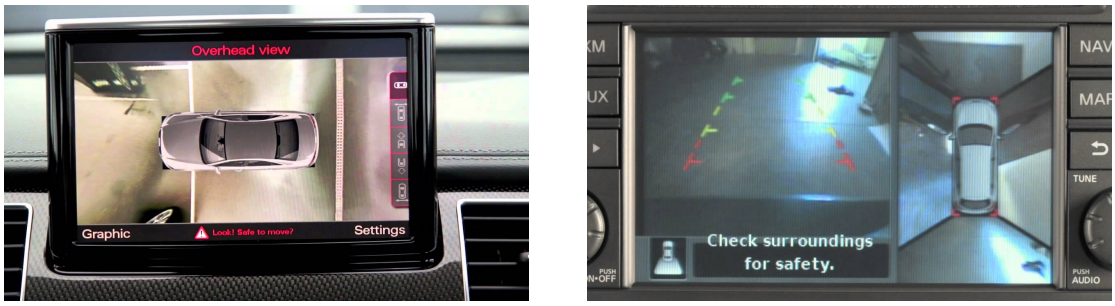
Recently, a new group of technological developments for the automotive industry has sprung: the Advanced Driving Assistance Systems (ADAS). This category embraces those advanced systems that offer the driver a guidance about the environment of the vehicle. Some known examples are parking assistant –ultrasound system that alerts the driver when the rear distance to an object is below a certain limit–, lane departure warning -alerts the driver when any wheel crosses a lane delimiter line-, traffic assistance –the vehicle detects and interprets the traffic signals–, among others.

This project will focus on a specific ADAS: 360° vision system. These device range offer the driver a complete top view –or bird's eye view– of the vehicle surroundings, allowing the driver to see through blind spots around the vehicle. The most common set-up includes several cameras located around the vehicle, and a display where the driver can view a composite image made from the camera data.

360° vision systems, also called bird's eye systems, are currently offered in high-end passenger cars. This systems can be combined with other features such as lane departure warning or pedestrian detection. Figure 1.1 shows two screenshots from different commercial 360° vision systems.

However, 360° vision system for passenger cars can not be straightforwardly applied to all vehicle types. The camera amount, its position, the vehicle length and height and the ROI (Region of Interest) make this algorithms specifically adapted to a vehicle type. Thus, the aim of this project will be to develop an 360° vision systems for long vehicles, specifically buses.

This vision system is an specific part of the work commissioned by the company Arcol to a multidisciplinary UPC team. Arcol is a mirrors and accessories manufacturer for motorhomes, buses and coaches. Currently, this company is working jointly with



(a) Audi S8 – 360 View Camera (b) Nissan Rogue – Around View Monitor

Figure 1.1: Two commercial 360° vision systems display view

the UPC in a 3-year project. This project main goal is to develop a camera-and-display based guidance system for the driver.

This project is carried out jointly in two UPC research groups: The Image Processing Group (GPI) and the Advanced Hardware Architecture (AHA) group. The AHA group main tasks are, on a high profile, to develop all the hardware to obtain and process the signals. The Image Processing Group is responsible of the computer vision algorithms development. During this project first year, the AHA group is developing the recording/display drivers and programming the hardware accelerators needed. Meanwhile, the GPI group is developing the 360° vision system and a pedestrian detection system.

This TFG project contains the part of the work developed in the Image Processing Group developing the 360° vision system, focusing on the image warping algorithms. Thus, this project also develops several parts corresponding to other stitching stages. Stitching algorithm parts developed by other GPI team members, and therefore outside the scope of this project, are only briefly stated for a whole stitching algorithm understanding.

1.1 Project goals

The work done in the Image Processing Group started on February 2014. As stated on the Project Proposal and Workplan delivered on February 14th, this TFG work will end at June 30th, although the work in the ARCOL project will be still in development until December 2014.

This project goals had been defined before the development began, and are stated in the Project Proposal and Workplan mentioned above. The goals of this project can be summed up in the following points:

Representative points automatic detection. To perform the stitching a reference points set had to be highlighted on the image. One of this project goals is to define this points and the model shown to each camera on the calibration step.

Estimate the warping parameters. Study the most suitable transformation and the parameters needed to deform the input images to obtain the output stitched image.

Blending the results on the final stitching. Define an algorithm to merge the warped images on the final composition.

Follow the requirements stated by the Arcol project. All the procedures stated before have to be done according with the Arcol UPC project requirements. This requirements and specifications can be found in Chapter 2.

The work done in this five months has been focused on developing a proper warping algorithm for this specific environment, taking into account all the technical an logistic issues. The specific tasks that had been included in this project scope are explained in Chapter 4.

1.2 Time Plan

The time plan followed during this project development is shown in Figure 1.2 Gantt Diagram. The developed work has been split in 8 work packages. A detailed description of each work package and the internal tasks developed can be found on Appendix A.

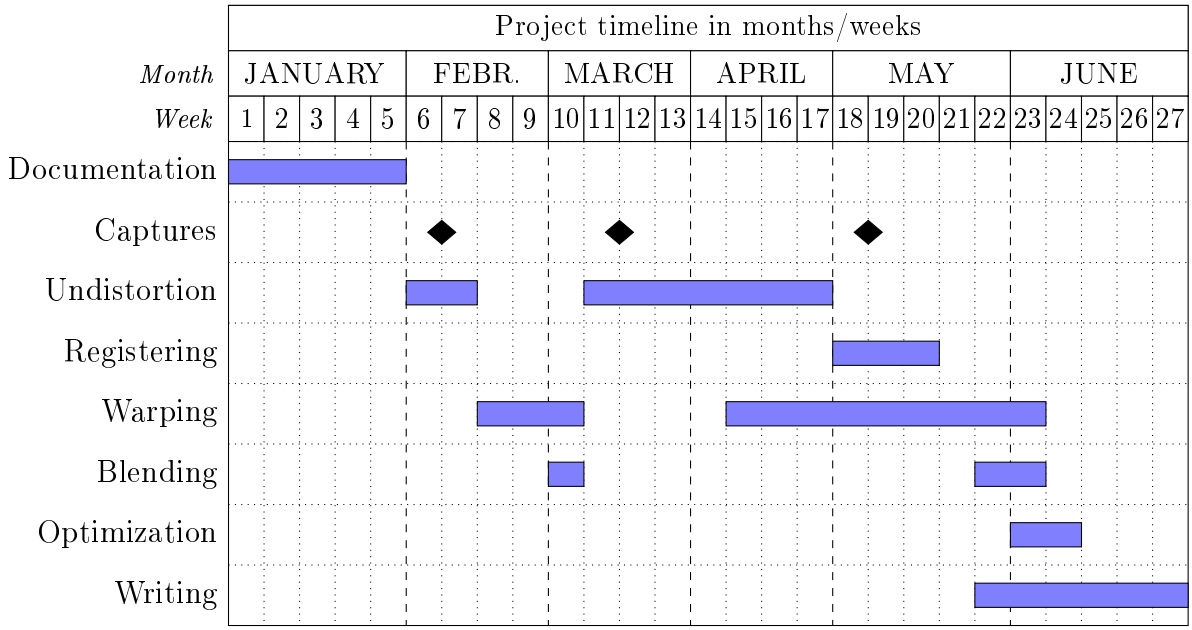


Figure 1.2: Project Gantt Chart

CHAPTER 2

REQUIREMENTS AND SPECIFICATIONS

This project goals include following the requirements stated by the Arcol company to the 360°vision system development. In this chapter both the software and hardware details are stated. The final system will be installed on the client bus using custom hardware. Thus, the designed software will have to adapt to the designed hardware constraints. Moreover, this Chapter also describes the requirements stated by the Arcol company to the final product.

2.1 Hardware and software specifications

The final hardware consists on 4-6 cameras with the corresponding fisheye lenses, a board with a CPU and a FPGA, a TFT display and adapters set to interconnect the different devices to the board. The simplified hardware diagram is the following:

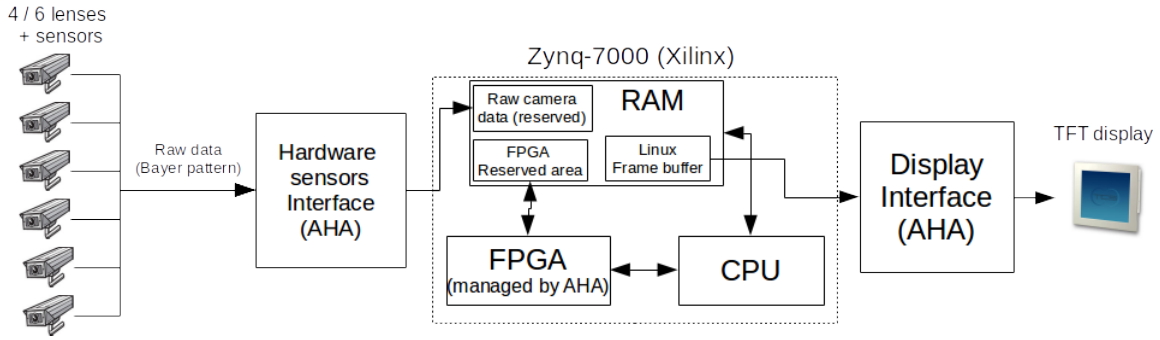


Figure 2.1: Arcol project hardware diagram

The lenses used (Yuntai-optical YT-5547-A1) have 178°(H) / 140°(V) viewing angle. The sensors (PixelPlus PH1100K) have an 1312×816 effective resolution and provide the data in Bayer pattern. For each pixel only one color is provided, following the pattern shown in Figure 2.2. To obtain the full $M \times N$ image the color pixels are interpolated from its neighbours.

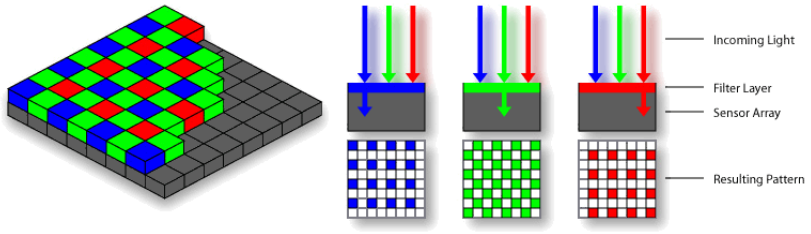


Figure 2.2: Camera sensor Bayer pattern. [1]

This Bayer pattern data is stored in an specific RAM location by the driver and ready to be accessed by the software. The evaluation board used in this project has an 8Mb RAM (Random Acces Memory). This memory is split between the OS (Operating system) kernel, the camera drivers and the FPGA.

The board used (Xilinx Zynq-7000) has an ZC702 SoC (System-on-Chip) with two ARM cores and an FPGA, 1GB RAM DDR2 and a 16MB flash drive. Additionally, it has two FMC connectors used as camera input and display output with the AHA developed drivers.

Although this system can capture data from 6 cameras, only 4 cameras will be used in this TFG project. The remaining 2 cameras are reserved for future implementations –i.e. articulated buses.

All the developed software will not run directly on the Zynq board. Instead, an abstraction layer with a Linux kernel system will be installed. The software will run in the Linux user layer, and proper drivers to access the raw camera data and the FPGA reserved memory will be provided.

The Linux OS will have OpenCV and Boost libraries installed. Thus, the software can use both libraries. Moreover, critical real-time and high parallelizable functions can be implemented on FPGA.

2.2 Required features

Using the specified hardware/software combination stated before a stitching system will be developed. The resulting composite image must have the following properties:

Non-discontinuities on ground lines –i.e. lane lanes. The ground lines must neither be broken nor change its direction between cameras.

Exposure compensation. Perform a light correction between cameras.

Avoid double-images or ghost images. Objects located above ground –i.e. people–, cannot be occluded or seen double in the overlapped areas.

This project will focus on the warping process. Thus, only the first constraint will be taken into account.

Regarding the calibration step, there is a reasonably freedom to select the calibration pattern. However, this system has to be user friendly without decreasing the software performance.

CHAPTER 3

STATE OF THE ART

One of the first car applied camera-based ADAS(Advanced Driving Assistance System) was the rear camera. This system consists on a wide-angle camera on the vehicle rear and a driver display. This system can be found in many large vehicles as a parking guidance, as long as the driver does not have any direct visibility due to the interior mirror absence.

One of this system evaluations is the 360° vision system. This system includes several cameras –usually from 3 to 6– located around the vehicle. The system does not only show the image from the cameras, but also merges them in one virtual top-view image. The first 360° vision systems for cars were available only on high-end vehicles as a luxury accessory. However, nowadays many mid-end vehicles also offer this feature [2].

In this section the most relevant 360° commercial systems are briefly described. As long as the technical details about this systems is not available, the description will focus on technical aspects extracted from promotional and non-official reviews.

Besides the commercial systems this chapter also describes the algorithms used in overhead image stitching. This chapter second part contains the main algorithms description, either those specific for bird's eye view or those generic stitching algorithms relevant on this work.

3.1 Commercial systems

Currently there are several manufacturers that offers an 360° view systems. Most of this systems have similar features, such as the number of cameras or the available viewpoints available. In this section the most relevant ones will be described, focusing on the main features and the relevant technical details.

3.1.1 Nissan: Around View Monitor

The system presented by Nissan in their *Around View Monitor* consists on four cameras located around the vehicle –one at the front, one at each rearview and one

at the rear. The captured images are deformed to match an specific ground pattern[1]. The four deformed images are shown on a single canvas. In some working modes, in this canvas are also overlapped lines showing the vehicle path of the vehicle according to the steering wheel rotation and current speed. [4]

As it can be seen in Figure 3.1, the four images are undistorted –the effects of the fisheye lens are compensated– and aligned to a common ground. However, this screenshot shows that the front camera presents a +10cm misalignment with the lateral cameras. However, this misalignment is acceptable in this application because the four images are not merged. In the application presented in this project, however, the four images are merged in an unique image –not four fragments. Thus, this misalignment level cannot be admitted.

Figure 3.1: Around View Monitor system screenshot on a Nissan Qashqai [5]

This system was first installed in Nissan Elgrand –a luxury vehicle–, but now it can be found in mid-end vehicle as Nissan Note.This system is offered by other brands, such as BMW (Surround view System) or Infinity (Around View Monitor). There are also manufacturers specialized on vehicle accessories that also offers this system, such as Alpine (HCE series). [6] [3]

3.1.2 Audi: 360° View Camera

Audi offers a similar 360° vision system: the *360° View Camera*. This system has the same camera setup than Nissan *Around view monitor*. However, in this case the four images are blended into one image. A smooth transition is performed on the overlapping areas between cameras to make a single camera effect. [7]

As it can be seen in Figure 3.2, the four cameras are effectively merged in one single image, but there are still misalignment between cameras. On the vehicle front right side it cat be seen that a ground line disappears in the transitions between the two cameras. Moreover, on the front left side the white car can only be seen partially.

This project will focus on solving the image misalignment only on ground level, so the ghost image of the white vehicle will still be present with the approaches presented.

[1] [3] shows the calibration process for a car 360° vision system

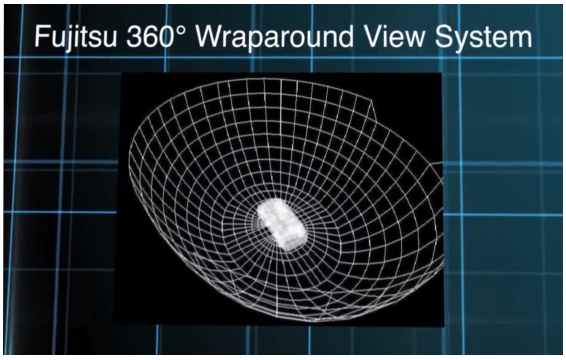
Figure 3.2: 360° View Camera on an Audi S8 screenshot [7]

However, the misalignment –or disappearance– on ground-level lines is an aspect to be improved on this project.

A similar system can be found on some VolksVagen models –such as Touareg–. In this system, however, the overlapped area is marked with thin delimiter lines. [8]

3.1.3 Fujitsu: Multi-Angle Vision

The *Multi-Angle Vision* system by Fujitsu presents a different approach than Nissan and Audi ones. Instead of wrapping the image on a planar surface to obtain an overhead view, this system wraps the four images on a single hemisphere centred on the vehicle, as it can be seen in Figure 3.3a. [9]



(a) Synthetic mesh where the four images are projected

(b) One viewpoint from the images projected on the mesh

Figure 3.3: Mesh and output image of Fujitsu Multi-Angle Vision [9]

This powerful system allows the user to select the viewpoint, and also minimizes the ghost effect on elements above the ground plane. This system can offer many viewpoints to the driver by rendering the mesh from any viewpoint. If it is rendered from above, the image is a deformed bird eye view, with and higher viewed area than the previous systems. The usage of many viewpoints allow the driver to see in more detail different surroundings points –lateral views of the road, frontal view on an intersection of streets, rear view when parking, etc. However, this system has minor

drawbacks. First of all,there are still minor errors on the transitions between images that seem not easy to solve. Another minor drawback is that the mesh system makes more difficult to draw the vehicle path lines on the image, as seen in Nissan approach.

Although a modification based on this system may offer high quality results in this project, this approach is not explored in this TFG. The lack of information about the procedure used and the probably high computational cost left this research path to a future iteration of the project.

3.1.4 Land Rover: Surround Camera System

Land Rover offers a different system than the ones stated before. Instead of using four cameras, the Surround Camera System uses five cameras: two on the front –left and right– two on the sides –left and right– and one on the rear. However, this system does not perform any stitching. Instead, the five images are shown in a grid as seen in Figure 3.4. [10]

Figure 3.4: Screenshot of the Surround Camera System on Land Rover [11]

Although this system can offer a complete view around the vehicle using more cameras, this is not included in this project goals. Placing the cameras on the corners rather than on the sides, however, is an option for future iterations on the Arcol project.

3.2 Technical information

There are two main sources regarding 360° vision stitching system: general information about image stitching –mainly form panorama creation– and specific information about bird's eye systems for passenger vehicles.

Although panorama stitching and bird's eye stitching are different systems, they share a common pipeline. This can be summed up as shown in Figure 3.5. [12]

As it can be seen, the stitching process is divided in main 4 blocks: undistortion, image registration, warping and blending.

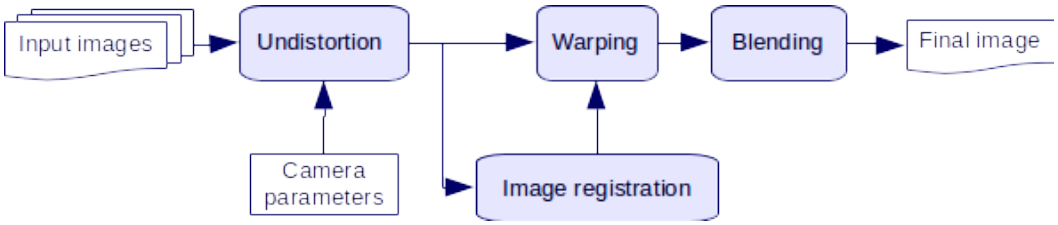


Figure 3.5: Simple pipeline for image stitching

The undistortion step consists on removing all the lens effect, mainly modelled with radial and tangential distortion. Although this step is outside the scope of this project, the quality of the undistortion algorithm will affect the next steps. The resulting image resulting of this step should only present perspective deformation.

Image registration consists on detecting a points set on a image and assign them either to another image or to a common ground plane. In panorama image stitching the different key points are related from one image to another, but in 360° vision systems the key points are related to the ground calibration pattern view. This project will only perform a manual key point registration, leaving the automatic process to a future iteration.

Warping is the technique to deform the undistorted image to match a set of defined key points. This part will be fully developed in this project.

Once the images are deformed, in the blending step the merging parameters and techniques are applied. Although this step is outside this project scope, basic techniques will be developed to obtain a final stitched image.

3.2.1 Image registration

To define the key points or features on the image two main techniques are used: automatic feature detection and pattern-based features.

3.2.1.1 Automatic registration

Automatic feature detection is used when a custom pattern cannot be defined. This is the case of panorama stitching, where the images to be stitched are not previously known and cannot be calibrated. In this case, the image are analyzed to detect relevant points.

Once the automatic features are detected the next step is to perform a two-by-two matching between the features of the different images. As long as the different images are not completely overlapped, only a feature subset from each image will have a correspondence to the other image. Moreover, there can be outliers –correspondences between images that are wrongly established– and valid features that have not been detected.

If the automatic feature detection has been done correctly and there is enough overlapping area, the amount of correctly detected correspondences will be large enough

to discard outliers and establish a pattern matching. Then, the next step, warping, can be performed. Figure 3.6 shows an example image with features correctly matched between two viewpoints.

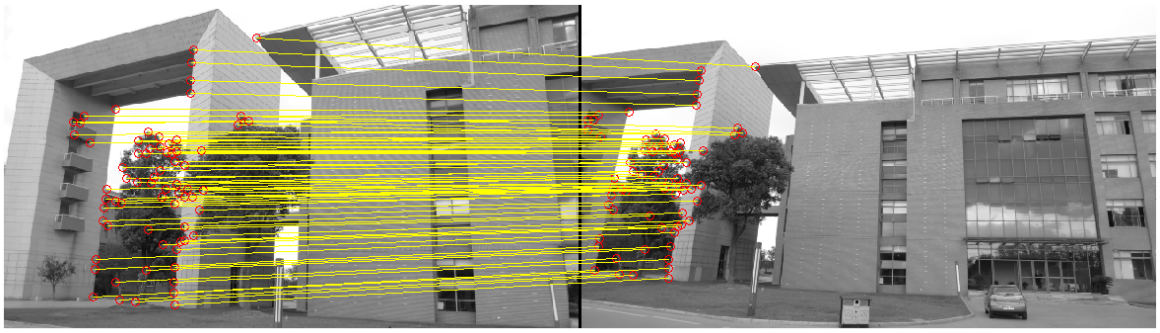


Figure 3.6: Automatic detected matching correspondences between two images [13]

This approach main drawback is that relies on the image information. If the image does not have significant points –contrast changes, distinguishable textures, enough overlapping area, etc.– the stitching process will not obtain good results. However, if the image has enough information the stitching obtained can be very precise.

In the case of 360° vision systems, as it can be seen in the commercial systems, the information and precision in the camera images is significantly low. In addition, the overlapping areas between two cameras is very low or, with fisheye cameras, the distortion on this area is very high. This two factors motivate a pattern-based feature detection system usage.

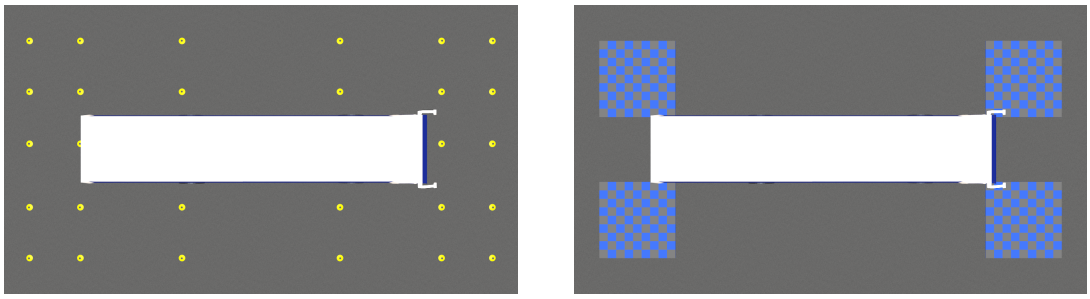
3.2.1.2 Custom pattern registration

A completely different approach for image registration is the use of a ground pattern. In this case, the pattern is placed in a specific position to perform the calibration step. Then, all the cameras whose images are going to be stitched capture a frame showing the pattern in its position and the pattern is detected.

The key points set are obtained from the detection are matched with the actual pattern position. Using this method each image deformation parameters are estimated. In running time, the deformation previously computed is applied to all the images. The patterns used in this project can be seen in Figure 3.7.

This approach is widely used in 360°vision systems. In this case, the cameras are not moved during the different captures, as they are solidly attached to the vehicle and thus a previously calibration step can be done. Thus, a previous calibration step can be done.

The usual configuration is done by putting a pattern on the floor -for simplicity- and capture the image seen from each camera. This pattern can be either centred on the image -the area with better vision- or in the overlapping area between images.



(a) Cone pattern. (b) Chessboard pattern.

Figure 3.7: Virtual view of the patterns used in this project.

This method establishes a relationship between the scene 3D points and the ground. This allows to project the image in a canvas where the reference points can be defined without ambiguity.

For the reasons stated before –non-image dependency, custom canvas, offline calibration step– this approach will be used in this stitching application, rather than using automatic registration.

3.2.2 Warping

The warping algorithms deform the image following a constraints set. In this stitching context, this constraints set can be summed up as a point pairs set, ones referring to points in the original image and ones referring to points on the canvas.

There are several techniques to deform the image following this constraints. Depending on the method complexity degree the procedure aggressiveness , the resulting image presents a different error degree on the key points.

In this specific context, the point pairs set between the image and canvas is from 10 to 32 points, whereas in panoramic image stitching, for example, the key points set can be from hundreds. The small key points set used makes that the most aggressive algorithms, although providing a lower error on the key points, produce an unacceptable deformation on the image.

The most common algorithm and less aggressive algorithm for image deformation is *homography*. To provide a better fitting on the key points two different algorithms –with their variants– can be found: *Minimum Least Squares* and *Polynomial deformation*.

3.2.2.1 Homography

Homography is the most common procedure to deform an image and perform an stitching. It consists on assigning to each point of the destination image $D[k, l]$ a point of the origin image $O[m, n]$. Thus, the deformation is done following the equation that can be described with the matrix H –size 3×3 homography matrix–.

$$D[m', n'] = I[m, n] \tag{3.1}$$

$$\begin{bmatrix} t_i m' \\ t_i n' \\ t_i \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix} \tag{3.2}$$

This projective deformation is the affine transformations contstraintless form. This transformations include rotation, translation and scaling. Each one of this deformations can be described with an H matrix with specific parameters [14]. This transformation is done on the projective space using homogeneous coordinates $[m, n, 1]$. An additional coordinate (t_i) results from the $H \cdot x_{in}$ operation. Therefore, the resulting coordinates had to be normalized with this t_i factor in the back conversion to the Cartesian coordinates $[m, n]$

To compute the H matrix, the OpenCV[2] routine *findHomography* is used. The main parameters, options an algorithms used by its function are described in Section 4.3.1.1.

3.2.2.2 Moving Least Squares

Moving least squares method, as opposed to the homography method, does not make a global deformation in the whole image. Instead of that, this algorithm makes a local deformation near each key point.[15]

This algorithm is mesh-based. A mesh is an image subdivision in small squared *subimages*. Each division contains typically from 3 to 30 pixels by side. The deformation applied by the method is not done individually on each pixel. Instead, each mesh vertex is deformed, and the pixels inside each mesh square are deformed accordingly to the mesh deformation. Figure 3.8 shows the test shape deformation using three different MLS methods.

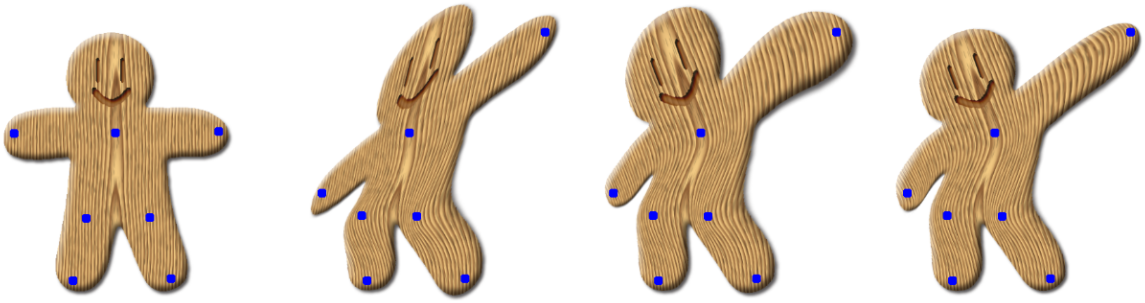


Figure 3.8: Test shape deformation using MLS techniques. From left to right: original image with control points, affine MLS, similarity MLS and rigid MLS.

3.2.2.3 Polynomial deformation

Polynomial deformation –or polynomial distortion– [16] is a mapping-based deformation system. A mapping, also used in the CV (Computer Vision) algorithms, is

[2]OpenCV is an open-source library used for Computer Vision.

matrix with the output image size, where each pixel contains a coordinate referring to an input image point. To obtain the value of each $I_{out}[m,n]$ output pixel, the mapping table is used as a look up table. The position $[m,n]$ of the mapping matrix will contain a coordinate $[m',n']$ from the input image. This coordinate corresponds to the $[m,n]$ output image pixel.

$$M[m,n] = [m',n'] : \quad \begin{bmatrix} m = 0 \cdots M_{out}, & n = 0 \cdots N_{out} \\ m' = 0 \cdots M_{in}, & n' = 0 \cdots N_{in} \end{bmatrix} \tag{3.3}$$

$$I_{out}[m,n] : m = 0 \cdots M_{out}, n = 0 \cdots N_{out} \tag{3.4}$$

$$I_{in}[m',n'] : m' = 0 \cdots M_{in}, n' = 0 \cdots N_{in} \tag{3.5}$$

$$I_{out}[m,n] = I_{in}\left[M[m,n]\right] \tag{3.6}$$

This approach will be fully described in Chapter 4, as it will be an important part of the final stitching algorithm.

Imagemagick is an image deformation tools complete set[16] that provides an environment to perform almost any kind of image processing. One of them is the *polynomial distortion*.

This deformation type assigns each deformation matrix value following a polynomial deformation. As example, Table 3.1 shows the polynomials from order 1 to 2 and its degrees of freedom. X_s, Y_s corresponds to the point on the source image, whereas X_d, Y_d corresponds to the points on the destination image.

Order	Polynomial	DoF
1	$X_d = C_{2x} \cdot X_s + C_{1x} \cdot Y_s + C_0 x$ $Y_d = C_{2y} \cdot X_s + C_{1y} \cdot Y_s + C_0 y$	6
1.5	$X_d = C_{3x} \cdot X_s \cdot Y_s + C_{2x} \cdot X_s + C_{1x} \cdot Y_s + C_0 x$ $Y_d = C_{3y} \cdot X_s \cdot Y_s + C_{2y} \cdot X_s + C_{1y} \cdot Y_s + C_0 y$	8
2	$X_d = C_{5x} \cdot X_s^2 + C_{4x} \cdot X_s \cdot Y_s + C_{3x} \cdot Y_s^2 + C_{2x} \cdot X_s + C_{1x} \cdot Y_s + C_{0x}$ $Y_d = C_{5y} \cdot X_s^2 + C_{4y} \cdot X_s \cdot Y_s + C_{3y} \cdot Y_s^2 + C_{2y} \cdot X_s + C_{1y} \cdot Y_s + C_{0y}$	10

Table 3.1: Deformation polynomials of the *imagemagick* toolbox. Note: DoF=Degrees of Freedom.

The deformation equations C parameters of the deformation equations can be estimated using a points set $[m,n] \longmapsto [m',n']$. This points are the detected points coordinates on the image and the desired placement on the output image. In order to be computed, each pair C_x, C_y needs at least one coordinates pair $[m,n] \longmapsto [m',n']$.

3.2.3 Blending

Blending consists on merging two or more images into a single image. Although this procedure is outside the scope of this project, it is necessary to apply a blending algorithm to obtain visible results. Therefore, several simple and non-definitive procedures will be applied .

There are several algorithms to perform the blending step. The most widely used are the following:

50% blending. This is the simplest procedure. It consists on merging each overlapped pixel in a 50% merging from the two images pixels. This system offers poor results. It is only used to detect errors on the overlapped areas that advanced techniques would obfuscate.

Feathering. This procedure consists on doing a smooth transition between the two images. In the overlapped area, the pixel weight from each camera is assigned based on the proximity to the camera vision area border. Thus, the pixels nearer to its camera border will have an higher weight than the other camera pixels [17]. Figure 3.9a shows an example using this algorithm.

Multiband blending. This is the most advanced approach. Based on the feathering algorithm, this algorithm performs a smooth transition between images. In this case, however, the image is filtered using different band frequencies –from high frequencies to low frequencies–. Instead of applying the same smoothness to all image bands, the low-frequency images have a smoother transition than the high-frequency images. Then, all the images are merged back together. This procedure is useful in panoramic stitching to have a smooth transition in brightness but a sharp transition in contour. In this way, the brightness continuous is preserved but the ghost images are deleted [18]. Figure 3.9b shows an example using this algorithm.

(a) Feathering blending (b) Muliband blending

Figure 3.9: Examples of two different images blended using (a) feathering blending or (b) multiband blending. In the first case, the image preserves many people ghost images, and the text is blurred. Multiband blending produces an image with less people ghosting artifacts and with sharper text.

CHAPTER 4

METHODOLOGY

Taking into account the information described in Chapter 3, this chapter will describe the work done in the development of a seamless stitching algorithm that minimizes the mismatching in the overlapped areas.

As stated before, this project contains the work developed in the scope of a bigger project. Thus, this chapter will focus on the techniques developed in this project: *Blender simulations* and *warping procedure*.

This chapter describes the creation of a complete framework to simulate the camera captures in a virtual environment. This includes the creation of a 3D reconstruction and the different scenarios. In addition, a framework that allows to render the desired scene has been also developed.

This chapter second part develops a warping algorithm with minimum distortion on the overlapped areas. This project presents a modification based on the homography algorithm that minimizes the misalignment on the overlapped areas.

4.1 Blender simulations

As a previous task to the warping process, a complete simulation framework for the ARCOL bus 360°vision system environment. This step is motivated by the technical difficulties to obtain real captures with diverse environments.

The simulation environment will be developed using the Blender software [19]. Blender is a free and open-source 3D computer graphics software used to creating animated films, visual effects, 3D printed models, interactive 3D applications and video games. [20]. This software can model the desired environments and design a complete framework to serialize the captures on this 3D simulated environment. It is a powerful system and has many design tools and features. However, this options and procedures wide range make the software to have a steep learning curve. Most of the development time invested to Blender simulations was spent to the software operation learning[1].

[1] A good reference book for Blender modelling that has been used in this project is [21]

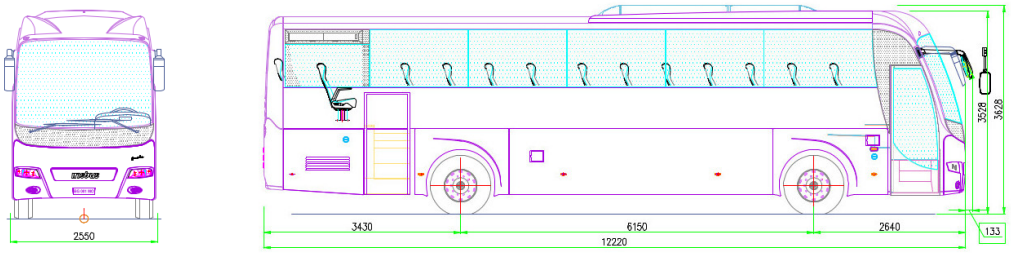
4.1.1 3D design

The different 3D scenes are created using the Blender GUI. This GUI allows to create several meshes and assign then to different layers. In this project, each layer will become a simulation environment –a calibration pattern, a street, people walking, etc.

Blender framework allows to define a real metric (in this case, centimetres) and relate all the measures to this metric. This allows to create accurately all the needed elements.

4.1.1.1 Bus design

The 3D bus is the basic structure on this 3D scene. It is created based on a bus sketch given by ARCOL. This sketch can be seen in Figure 4.1.



(a) Frontal (b) Lateral view

Figure 4.1: Bus sketch (frontal and lateral view) used in Blender modelling

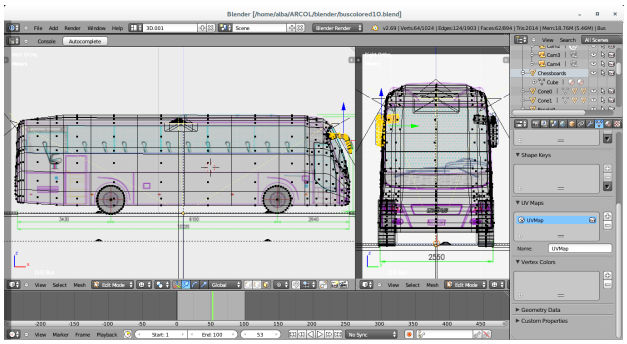
This bus has been modelled using the *Background image* tool. This tool allows to set a background image whose scale is related to the screen zoom. This allows to have a background pattern to model the mesh. [22].

To correctly size the bus, first of all a cube with the bus size ($1222 \times 255 \times 3628$ cm) is created. Then, the background images are moved and scaled to match the bus cube on each view. Afterwards, the bus is modelled by subdividing the cube edges and moving each vertex according to the ground pattern. Blender provides a tools set and options to do the face division and the vertices placement accurately.

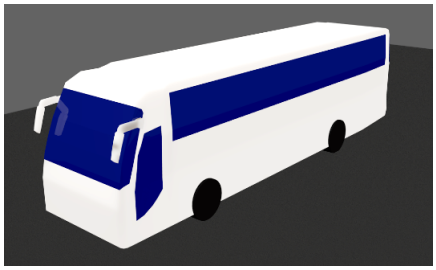
The bus is modelled in 7 meshes: the bus structure, the 4 wheels and the two rear views. Then, this meshes are merged in one single mesh . The result can be observed on Figure 4.2a.

The last aesthetic step is to assign materials on specific bus areas. Different materials with different colors, reflection indexes, roughness, transparencies, etc. are assigned to the different faces. The final result is shown in Figure 4.2b.

Although it is an unrealistic design, this does not affect the quality of the 3D scene. The cameras located on the bus sides only see a small and very distorted bus portion.



(a) Screenshot of bus design procedure with lateral and frontal view.



(b) Rendering of the bus mesh.

Figure 4.2: Bus design procedure

The last step is to place 4 cameras on the bus side centres. This cameras are located at the height where they will be in the real buses (3.30m). They were as close as Blender allows to the bus sides, with a 90°angle to the sides and with an azimuth enough to see the horizon line and the bus side. However, the Blender cameras cannot model lens distortion. A new camera type, *busCamera*, is created to obtain the raw stitching images. This camera type has a 108mm focal lens and a 6.51mm×3.42mm sensor size.

4.1.1.2 Layers design

With this basic bus scene, several layers corresponding to several scenes are designed. This scenes include from the chessboard/cone calibration patterns used in this project to virtual street simulation and people moving in a specific path.

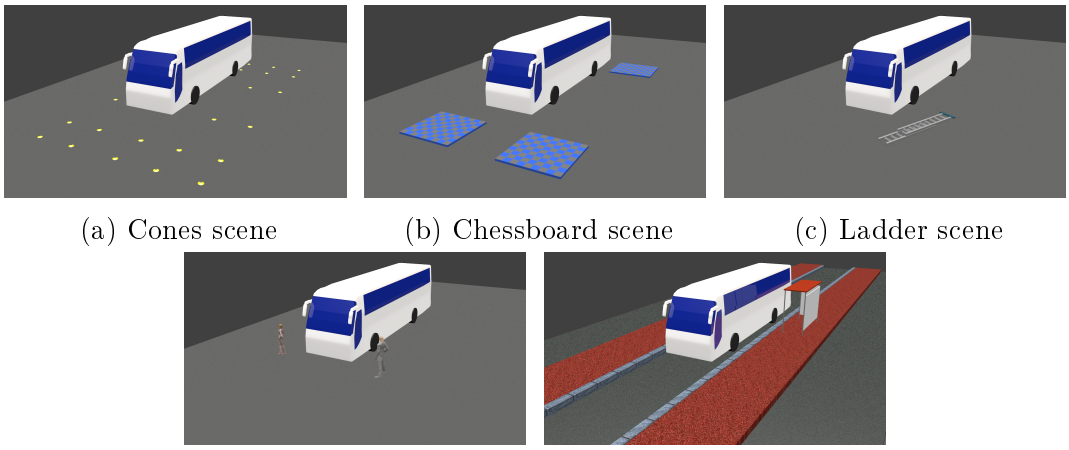
The calibration patterns were carefully designed to match as much as possible the calibration patterns used in the real captures. This includes from the 3D recreation of the yellow cones used in registering to the precise chessboard placement on ground specific positions.

The remaining scenarios are more freely defined. There is, for example, an scenario that tries to imitate the placement of two ladders used as objects on one of the recordings. The ladder was carefully defined –the ladder pieces, the twisted steps, the rubber endings with mat material, etc.– but it was placed approximately using calibration points as reference points.

Another scenarios -a realistic street with a bus stop, two people moving in a defined path- are designed accurately measuring all the elements –road width, bus stop and people height, etc.–. However, the details were artistically decided. Figure 4.3 shows five screenshots of this designs.

4.1.2 Automatic scripting

Obtain the captures from the four cameras in different scenes is a tedious and slow task. Thus, an specific framework to perform this task has been defined. The python-



(a) Cones scene (b) Chessboard scene (c) Ladder scene

(d) People scene (e) Bus stop scene

Figure 4.3: Screenshot of the different scenes that have been generated with Blender.

programmed Blender kernel is exploited to obtain a python-based framework. This framework allows to do camera serialized captures with specific configurations.

To perform this task Blender offers the *bpy* python package [23]. This package offers all the available options in the Blender GUI accessible with Python scripting- However, use python scripting to perform Blender modelling is rarely used, and the documentation is very brief and imprecise.

The first test done was to develop a simple script to capture the data from the four cameras. Code 4.1 shows an example for one camera.

Code 4.1: Example code to perform a simple capture

```
#!BPY
import bpy
front = bpy.data.objects["Cam1"]
bpy.context.scene.render.image_settings.file_format = 'AVI_RAW'
bpy.context.scene.camera=front
bpy.data.scenes['Scene'].render.filepath =
  '/imatge/apujol/ARCOL/blender/output/cam1.avi'
bpy.ops.render.render( animation=True )
```

The use of this scripting eases the four camera captures acquisition. Instead of manually selecting the camera, set the output properties and wait during the rendering, a simple execution of a script can obtain all the data. Moreover, the execution of this script can be done with non-graphic Blender interface. Thus, the data generation –which can be very time consuming for long sequences– can be left on a computing server.

The next step is to automatize all the process with Python scripting. With a Blender file with all the generated objects and relationships and a Python configuration file, a more complex generation can be done. Python offers a configuration-treatment class, *configparser*, which is very useful to read all the configuration. Then, using Blender *bpy* interface, the different parameters are set to the blender scene. Finally, the render process is done as shown above.

4.2 Image warping

The next sections describe the different procedures used in the presented image stitching. Although not all the steps required to perform the stitching are inside the scope of this project, to obtain the results in Chapter 5 all the steps are applied. Therefore, all the procedures done in the required steps will be briefly discussed, focusing on the effects on the warping step.

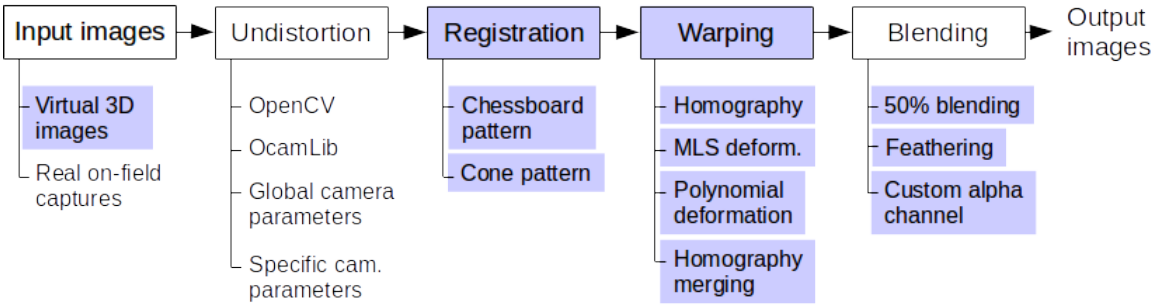


Figure 4.4: Stitching diagram with the tested methods on each step.

Figure 4.4 shows the complete stitching diagram and the algorithms used in this project. The highlighted steps are fully developed in this TFG project. From the other steps, only the highlighted methods had been developed in this project.

4.2.1 Captures

The image sources used in this project are, as stated before, the virtual 3D images created with Blender and the real captures in the Arcol facilities done by other team members.

The on-field captures were not done in real buses. Instead of that, four cameras have been placed in four poles top. This poles have approximately the same height as the definitive bus and are located in the same places where the bus cameras should be. Additionally, four more poles have been placed emulating the bus corners, to have a reference to the overlapping area seen for each camera.

The first capture was done locating several cones forming an specific ground pattern. This ground pattern has been defined beforehand to cover all the reasonable critical stitching positions. However, the precision lack forces a new pattern definition. Therefore, a second capture was done, changing the ground pattern from cones to four chessboards located on the four bus corners.

The first capture, with the cones placed on specific positions, was done to test a reasonable part of the methods stated hereunder. The need of more precision has drove the change to another pattern, the chessboard. This pattern was used to perform the last tests and obtain the results presented.

To follow the development –already outside this project scope– a capture with a van is scheduled by middle July. In the following weeks, captures with real buses driving through roads and streets will be done to continue the development.

4.2.2 Undistortion

Undistortion consists on compensate the non-linear camera lens effects. A simple, non-distorted camera can be modelled as a rectangular panel and a origin point centered and perpendicular to this panel, as shown in Figure 4.5. To obtain the image, a line from the origin point to each rectangular panel point –or pixel– is traced. The pixel value is the color of the object crossed by this virtual line.

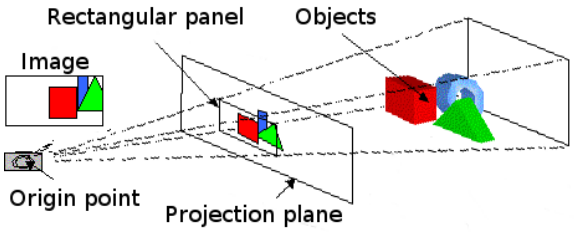


Figure 4.5: Distortion-free camera model

However, when the lens can not be modelled as a rectangular panel the camera distortion arise. In the fisheye cameras case, the ones used in this project, this effect is extremely visible. To compensate this effect two different methods have been tested. This methods analytically model the distortion effects and apply the inverse formula to the distorted image. The methods tested on this project are OpenCV routines[24] –generic routines for all camera kinds, based on Buguet procedure- and OCamLib routines[25]– specifically designed for fisheye cameras.

The first method -(OpenCV) computes the radial distortion using a even polynomial function and the tangential distortion the using the cross-terms multiplication coefficients. The second method (OCamLib) only models the radial distortion using a complete polynomial.

To select the proper undistortion routine another factor has to be taken into account: on the final product, the distortion parameters should be computed once and be valid to all the used cameras. Thus, the appropriate routine would be that which presents less variability in the lens parameters, and produces a less distorted image on average.

The different undistortion algorithms had been explored by another team member. This project only includes the creation of a binding between the selected undistortion algorithm and the remaining steps.

4.3 Registration

The registration step consists on defining a relationship between input image pixels and output image pixels. This project does not cover the automatic pattern detection. Therefore, the matching between pixels has been done manually by the project author. First of all, a pixels set from each image is referenced to a 3D point, as shown in

Equation 4.1. For convention, in this project $[m, n]$ corresponds to pixels and (X, Y, Z) to the 3D scene measures.

$$\begin{array}{lll} I_1[m_1, n_1] \longleftrightarrow (X_{11}, Y_{11}, Z_{11}) & \cdots & I_4[m_2, n_2] \longleftrightarrow (X_{41}, Y_{41}, Z_{41}) \\ I_1[m_2, n_2] \longleftrightarrow (X_{12}, Y_{12}, Z_{12}) & \cdots & I_4[m_2, n_2] \longleftrightarrow (X_{42}, Y_{42}, Z_{42}) \\ \cdots & & \cdots \\ I_1[m_{k_1}, n_{k_1}] \longleftrightarrow (X_{1k_1}, Y_{1k_1}, Z_{1k_1}) & \cdots & I_4[m_{k_4}, n_{k_4}] \longleftrightarrow (X_{4k_4}, Y_{4k_4}, Z_{4k_4}) \end{array} \quad (4.1)$$

In the next step the projection plane is defined. Each point (X, Y, Z) stated above should have a projection point on the output image $I_out[m, n]$. This relationship general expression is stated in Equation 4.2.

$$\begin{aligned} (X, Y, Z) &\longmapsto^{\varphi} [m, n] \\ I_{out}[m, n] &= \varphi(X, Y, Z) \end{aligned} \quad (4.2)$$

The most common plane is a ground parallel plane. In this case, the previous expression can be particularized as shown in Equation 4.3. Offset and scale factors depend on the actual output image size.

$$\varphi(X, Y, Z) = [X \cdot scale + offset_x, Y \cdot scale + offset_y] \quad (4.3)$$

The relation between the input images $I_{1.4}[m, n]$ and the output image $I_{out}[m, n]$ is used on all warping procedures.

4.3.1 Warping algorithms

This project main part is the warping algorithm adapted to the Arcol environment development. To obtain the best results in a uncertain environment –real captures with the bus had not been done yet– different strategies are tested.

The first approach is to use the state-of-the-art algorithm, homography. However, the results obtained are not as good as expected. Therefore, two mesh-based strategies were tested. In those cases, the results obtained are neither as good as expected. Finally, a new approach based on homography merging is tested.

4.3.1.1 Homographies

The first used approach is homography. As explained in Section 3.2.2.1, this approach consists on defining a 3×3 deformation matrix, which maps each destination image point to an origin image point.

First of all the Cartesian coordinates are converted to homogeneous coordinates. This implies translating the $[m, n]$ coordinates to the form $[m, n, 1]$. Then, the following transformation matrix is applied:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} m \\ n \\ 1 \end{bmatrix} \tag{4.4}$$

This transformation returns a 3-coordinate destination image point. This point has to be translated back to Cartesian coordinates. This translation is done by scaling the vector to obtain $z = 1$. Therefore, the $[m', n']$ destination image coordinates can be obtained as follows:

$$x = h_{11}m + h_{12}n + h_{13} \qquad y = h_{21}m + h_{22}n + h_{23} \qquad z = h_{31}m + h_{32}n + h_{33}$$

$$\begin{aligned} m' &= \frac{x}{z} = \frac{h_{11}m + h_{12}n + h_{13}}{h_{31}m + h_{32}n + h_{33}} \\ n' &= \frac{y}{z} = \frac{h_{21}m + h_{22}n + h_{23}}{h_{31}m + h_{32}n + h_{33}} \end{aligned} \tag{4.5}$$

To determine the H matrix, the available information is points pairs in the form $[m, n] \rightarrow [m', n']$. Theoretically the H matrix has 9 degrees of freedom. However, the conversion from homogeneous to Cartesian coordinates deletes a degree of freedom. Therefore, the remaining system 8 degrees of freedom have to be determined by the points pairs. Since all points pair $[m, n] \rightarrow [m', n']$ can be used to determine 2 unknown system terms, 4 points pairs are needed to determine the complete system.

The point pairs available are usually more than 4. This forces an approximation algorithm definition. OpenCV[26] offers three different methods. This methods are tested with the available data, and the results can be found in Chapter 5.

Least squares [27] This is the default method. It consists on finding those parameters that minimize the quadratic error sum on each point [Equation 4.6]. This method is very sensible to outliers, but in good conditions –points correctly detected– offers good results.

$$S = \sum_{i=1}^{n} r_i^2 \qquad r_i = y_i - f(x_i, \boldsymbol{\beta}) \tag{4.6}$$

Least median of squares [28] This method is very similar to the one mentioned above. However, instead of perform the sum of the mean quadratic error, in this method the median is performed [Equation 4.7]. This method is more robust to outliers, but in good conditions obtain little less precision than Least Squares.

$$S = med_i r_i^2 \qquad r_i = y_i - f(x_i, \boldsymbol{\beta}) \tag{4.7}$$

RANSAC [29] Is an iterative method robust to outliers. It consists on selecting a data subset, fit the parameters to this data and test the found fitting to all the dataset. This procedure is iterated until the probability of obtaining a better fitting to a data subset is below a certain limit.

4.3.1.2 MLS deformation

Moving least Squares, as explained in Section3.2.2.2, is a mesh-based deformation. This method obtain a deformation that perfectly matches the control points –the points found in Section 4.3. The neighbour pixels are smoothly deformed to match those constraints. Analytically, it can be defined as follows:

$$min_{l_v(v)} \sum_i w_i |l_v(p_i) - q_i|^2 \qquad \text{where} \qquad \begin{array}{c} w_i = \dfrac{1}{|p_i - v|^{2\alpha}} \\ (p_i, q_i) : \text{detected points pair} \end{array} \tag{4.8}$$

The function $l_v(v)$ has a different expression for each evaluation point v, since w_i depends on v. [15] propose three different methods to compute the Least Square minimization to compute each $l_v(v)$ function. As long as each $l_v(v)$ function is an affine transformation, the operation can be simplified to the M_v matrix finding for each v with specific properties.

Affine deformation Each affine deformation is computed using the classic normal equations solution. In some applications,this unbounded approach can generate undesirable effects, such as non-uniform scaling and shear.

Similarity deformation Similarity deformation is an affine deformation subset that only include translation, rotation and uniform scaling. In some applications, allowing local scaling an also produce undesirable deformations.

Rigid deformation This deformation type only include translation and rotation. Each deformation function $l_v(v)$ can only contain this two factors. When this method is applied to shape deformation it obtains fair good results.

To speedup the computation, instead of finding an $l_v(v)$ for each point, the image is divided into a grid. Only the vertices of the grid are deformed, and each cell content is deformed using bilinear interpolation. The results obtained with this method can be seen in Chapter 5.

4.3.1.3 Polynomial deformation

Polynomial deformation, as explained in Section 3.2.2.3, consists on modelling the image deformation using a polynomial function. To determine this polynomial function, the point pairs found as explained in Section 4.3 are used.

As same as with Homography algorithms, the polynomial parameters computation requires an specific amount of point pairs. In this case, the point pairs needed for each computation depends on the polynomial order, as can be seen in Section 3.2.2.3 Table 3.1.

The results obtained applying this method to the Arcol environment can be found in Chapter 5.

4.3.1.4 Homography merging

The algorithms stated on the previous sections have some major drawbacks. In addition, the previous stitching pipeline steps –undistortion and registration– are not as precise as expected. As it will be seen in Chapter 5, any of the results obtained with the previous methods are satisfactory. This fact arises the need to adapt the previous algorithms to this specific stitching environment.

As it can be seen in Chapter 5, the warped images still present deformations in non-straight lines. The main issue found using the homography algorithm is that cannot model the distortion artifacts still present on this images. Tests with Blender virtual images show that with fully-undistorted images homography obtains excellent results.

The modification presented on this section is based on the homography algorithm. It consists on divide the image in regions with at least 4 control points on each region. Assuming that the error between two pixels increase with the distance, the control points in those regions will have less error between them. This allows to create a more precise 3×3 homography matrix adapted to each subregion. It will be shown that warping the images with each specific homography produces less error on the control points than when using the global homography matrix.

In this specific case, the image division will be done in 2 regions for each camera. Each one will contain the control points corresponding to each $3m \times 3m$ chessboard pattern located on the bus corners, as seen in the generated Blender scenes. Thus, each camera image will have two different homography matrices, one associated with each seen chessboard.

Since the image seen from each camera has large areas without control points, a transition strategy between two different homographies has to be stated. However, this strategy is not defined using directly the homography matrices.

First of all, each homography matrix is converted to a mapping matrix. A mapping, as explained in Section 3.2.2.3, consists on a matrix with the output image size, where each $[m, n]$ element contains a reference to a location on the input image. If this element is not an integer, the resulting pixel will be an interpolation between the input pixels neighbouring the floating point. In addition to the mapping matrix, for each region is also defined a *validity area*. This area can be defined as the area where the homography deformation is fully valid, and other homography deformations should not have any effect.

To define the merging strategy a few considerations are taken into account. First of all, this merging strategy should be continuous. This condition is required to not have any discontinuity between the image areas and avoid discontinuities in ground lines. Another important condition, and directly dependant on the previous one, is that if a pixel is close to a *validity area*, this area should have similar deformation to the named validity area. With this considerations the following merging strategy is defined:

1. Definitions. The following parameters will be used:

$$\begin{aligned} M_i[m,n] &\ \forall m = 0 \cdots M, n = 0 \cdots N : \text{mapping matrix for the i-th region} \\ d([m,n],i) &\ \forall m = 0 \cdots M, n = 0 \cdots N : \text{pixel distance to the i-th } \textit{validity area} \\ W_i[m,n] &\ \forall m = 0 \cdots M, n = 0 \cdots N : \text{Weight mapping.} \end{aligned}$$

   The $W_i[m,n]$ must have the following properties (extracted from [30]):

   (a) $d([m,n],i) = 0 \rightarrow W_i[m,n] = 1, W_j[m,n] = 0, \forall j \neq i$

   (b) $d([m,n],i) > d([m,n],j) \rightarrow W_i[m,n] < W_j[m,n] \ \ \forall i,j$

   (c) $d([m,n],i) = d([m,n],j) \rightarrow W_i[m,n] = W_j[m,n] \ \ \forall i,j$

   (d) $d([m_1,n_1],i) = d([m_2,n_2],i) \rightarrow W_i[m_1,n_1] = W_i[m_2,n_2] \forall m_1,n_1,m_2,n_2$

   (e) $W_i[m,n] > 0 \ \ \forall i \quad \sum_i W_i[m,n] = 1$

2. From the previous weight matrix $W_i[m,n]$, which is a 'proximity measure' to the different validity areas, the $M_i[m,n]$ mapping matrices from each homography can be merged in the following way:

$$M_{out}[m,n] = \sum_i W_i[m,n] M_i[m,n] \tag{4.9}$$

This deformation should produce a smooth transition between areas. In this project specific case, an implementation using two validity areas is proposed, one corresponding to each chessboard seen by each camera. Thus, the merging strategy has only been defined on the area between the two chessboard. Figure 4.6 shows the scenario defined.

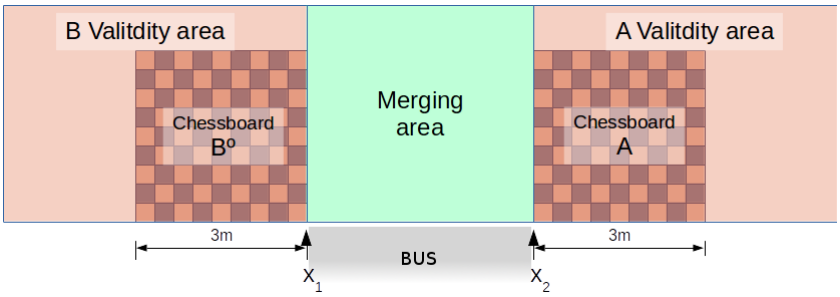


Figure 4.6: Camera 1 (frontal) validity areas and merging area setup.

The previous matrices and distances are defined as follows:

$$\begin{aligned} & d([m,n],A) = (m - x_2) \quad d([m,n],B) = (m - x_1) \\ W_A[m,n] = \frac{d([m,n],B)}{d([m,n],A) + d([m,n],B)} & \quad W_B[m,n] = \frac{d([m,n],A)}{d([m,n],A) + d([m,n],B)} \end{aligned} \tag{4.10}$$

This provides a linear transition between A and B area only within the horizontal coordinate. Using this algorithm, a pixel $[m,n]$ that with A homography will be

obtained from position $[m_1, n_1]$ and with B homography will be obtained from position $[m_2, n_2]$, with this technique will be obtained from the pixel $W_A[m,n] \cdot [m_1, n_1] + W_B[m,n] \cdot [m_2, n_2]$.

This technique has been tested with the chessboard captures and compared with the homography technique. The results obtained can be seen on Chapter 5.

4.3.2 Blending algorithms

Although blending is outside the scope of this project, blender techniques have been used to show the four camera images in one single image. However, exposure compensation algorithms –average the light conditions between cameras– has not been done in any project test.

The simplest method used in this project is *50% blending*. This method consists on simply averaging the images on the overlapped area. The image is divided in 2 image layers: *single-image* where the output pixel takes the RGB value of the corresponding input pixel, and *overlapping*, where the output pixel averages the two input images color-by-color $[R_{out} = (R_{in1} + R_{in2})/2,\ G_{out} = (G_{in1} + G_{in2})/2,\ B_{out} = (B_{in1} + B_{in2})/2]$. This method is only using on development, as it allows to see every artifact and misalignment between cameras.

On release mode, the main used method is *feathering*. This method creates a smooth transition between images. To compute this transition an alpha channel –binary one-channel image– is defined. The pixel image value is 1 when the reference image has information and 0 when does not have any information. Using this alpha channel, the OpenCV feathering function assigns a weight to each pixel. On the overlapping area, the weight assigned to each image pixel depends on the distance of those pixel to the alpha channel 1-zone. This can be stated as follows:

$$w_i[m,n] = \begin{cases} d(I_i[m,n],0) \cdot s & : d(I_i[m,n],0) \cdot s < 1 \\ 1 & : d(I_i[m,n],0) \cdot s \geq 1 \end{cases} \tag{4.11}$$

where
s : Sharpness. Typically 0.02
$d(I_i[m,n],0)$: Distance to the nearest 0-pixel, in pixels. (0 if $I_i[m,n] = 0$)

$$I_o ut = \frac{\sum_i p_i w_i}{\sum_i w_i} + i \cdot W_{EPS} \tag{4.12}$$

where W_{EPS} is an OpenCV scale factor $(1 \cdot 10^{-5})$

Modifying each image alpha channel the weight values can be changed. This allows to select each image influence area, narrowing or widening the overlapping area.

Using all the methods stated in this Chapter, the next chapter will describe the tests done and results obtained in this project development of this project. As it has been seen at this chapter start, in each stitching step several methods had been proposed. Due to the vast combination possibilities, only the most relevant will be described.

CHAPTER 5

RESULTS

In this chapter the results obtained with the warping methods stated in Chapter 4 are described. First of all, the two calibration patterns are stated, and the main advantages and drawbacks are discussed. Then, the results obtained using the different warping techniques –homography, MLS deformation, polynomial deformation and homography merging– are stated. The most successful techniques –homography and homography merging– will be discussed and evaluated both by visual inspection and analytically. Finally, the applied technique will be evaluated in the hardware environment.

5.1 Registration

The first calibration pattern used in this project was built by putting several cones in specific ground positions. This system main advantage is that the pattern shape can be easily changed during the same capture to perform development tests.

However, results show that the precision obtained using this procedure is very low. As it can be seen in Figure 5.1a, the cone shape changes drastically from one viewpoint to another. Thus, there is a not insignificant error in the cone position determination, even with manual registering.

To obtain a better precision a new pattern has been tested. In this case, a chessboard –the same pattern type used in typical camera calibration applications– is used. A 3×3m pattern was placed on the ground on the bus corners. As it can be seen in Figure 5.1b, the precision that can be obtained using this procedure is much larger than when using cones as calibration pattern.

(a) Cone views

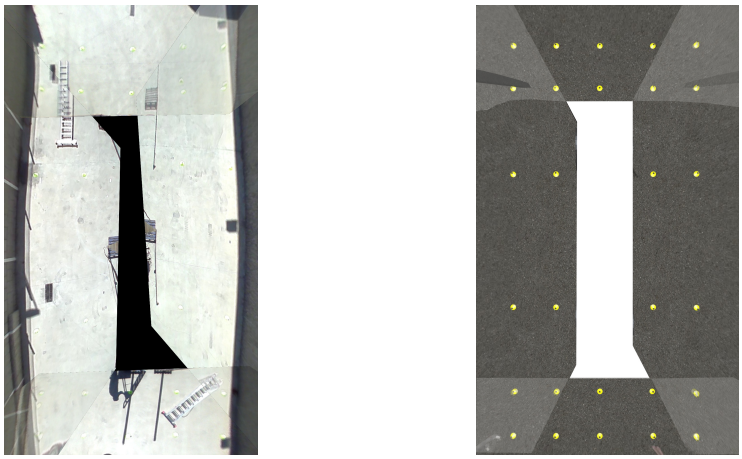
(b) Chessboard view

Figure 5.1: Comparison between cones and chessboard pattern precision

In this project, homography was tested using both methods, as it is the state-of-the-art algorithm in image stitching. MLS deformation and polynomial deformation were only tested using cone pattern The results obtained advise against following this working path. Homography deformation was developed when the chessboard pattern was already available. Thus, all the tests using this method were with the same pattern.

5.2 Warping: Homography

The first warping method tested in this project is homography. Although this is the state of the art method, the results are not satisfactory. As it can be seen in Figure 5.2a, the image is not correctly warped. The image presents serious misalignment between the two image cones used as key points, especially in the bottom right corner. Moreover, it can also be seen that the image does not present straight lines, especially in the left side. This deformations indicate a possible error, either in the undistortion or registration step.



(a) Homography with real images (Blending: 50% merging)

(b) Homography with Blender images (Blending: 50% merging)

Figure 5.2: Homographies using least squares method (default).

To confirm this hypothesis, Figure 5.2b shows a virtual distortion-free image generated with the Blender environment. In this case, the image is precisely warped and present straight lines in the whole image. However, there still can be seen minor misalignment in the cones, possibly caused by the pattern precision error.

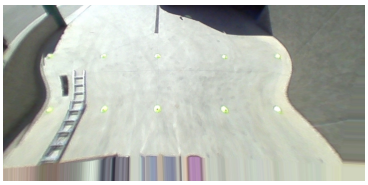
5.3 Warping: MLS deformation and polynomial deformation

The unsatisfactory results obtained with Homograph motivated the usage of a mesh-based warping system that preserves the key point location. Using this constraints two different methods are tested: MLS and polynomial deformation.

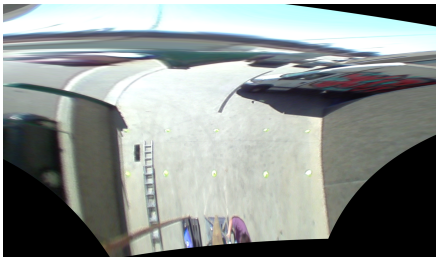
MLS deformation (Figure 5.3a) has shown to be an inadequate algorithm. The deformation is not as smooth as expected, and the image is heavily distorted. Although

the correspondence points are now perfectly aligned, the image distortion is unacceptable. This algorithm could only be applied if there were hundreds of correspondence points. In this case, all the mesh would be defined and the distortion should decrease.

Another method tested is Polynomial deformation (Figure 5.3b). This deformation is smoother than MLS, but the image still became undistorted when leaving the correspondences area ((Figure 5.3b) superior part). This would not be a major issue, but the distortion in the lower part –which corresponds to the bus side– is noticeable. Moreover, tests done with less key points show that the method performance is very affected by the key points detected.



(a) MLS with similarity deformation



(b) Order 1.5 polynomial deformation

Figure 5.3: Mesh-based deformations (frontal camera only)

5.4 Warping: Homography merging

The homography algorithm was also tested on the new chessboard captures. In this captures, the precision at the chessboard corners is higher, but as it can be seen in Figure 5.4, this algorithm presents severe misalignment errors to the pattern –red: theoretical points; green: warped points–. This arises the need of creating an algorithm variant to perform this warping.

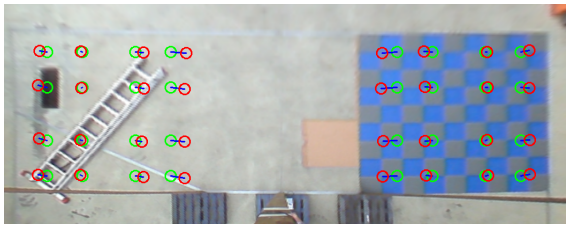
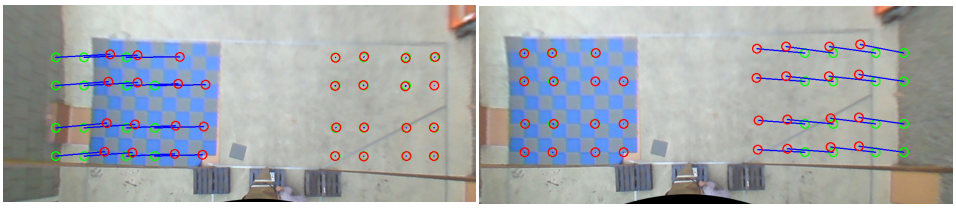


Figure 5.4: Homography error with chessboard pattern - Camera 1

The principal hypothesis for this error is that the image is not fully undistorted. Tests with different homography estimation techniques show that the simple homography algorithm can not adapt to the full image if distortion is still present. Thus, the technique proposed splits the image in fragments and performs an specific homography on each fragment. Taking into account the available data, the key points are split based on the two available chessboards per image.

Figure 5.5 shows the result obtained warping the image with each homography. It can be seen that the image is fully adapted to the chessboard where the homography is computed, whereas the other chessboard is unaligned.



(a) Left corner homography (b) Right corner homography

Figure 5.5: Homography error using only (a) left corner or (b) right corner key points

In the next step a smooth transition is performed, *shrinking* the image to fit both homographies. Using the algorithm seen in Section 4.3.1.4, the warping seen in Figure 5.6 is obtained. This method performs a smooth transition between the two homographies, leaving the two chessboard aligned to the reference points.

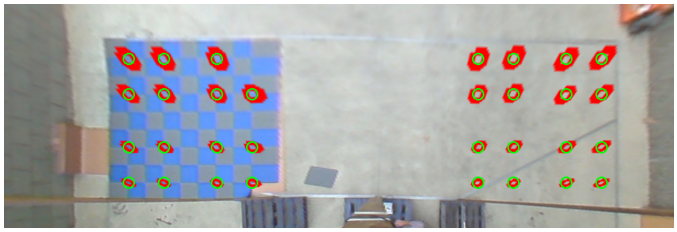


Figure 5.6: Resulting key points matching using homography merging

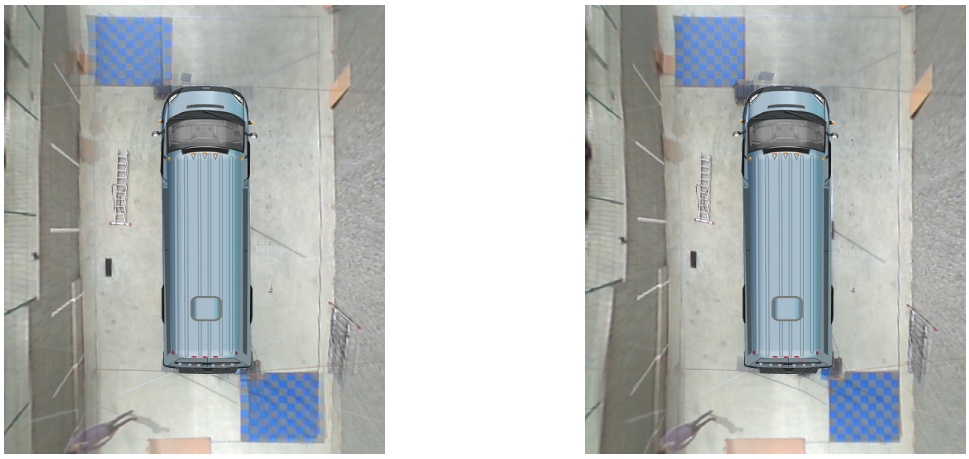
When this method is applied to the full chessboard it shows excellent results. As it can be seen in Figure 5.7, the final image presents indiscernible misalignment errors. The chessboard pattern shown presents no ghosting effect –if the chessboard is misaligned, the two chessboard superposition creates a visible shadow– and the people's feet coincide between cameras. However, the objects located above ground level present misalignment errors on the overlapping areas. In the Figure 5.7 (3rd image; top left corner) it can be seen that the ladder located above ground presents heavy misalignment. However, when the ladder is on ground level (4th image; top left corner) it only presents minor misalignment due to the ladder height.

Figure 5.7: Different sequences using Homography merging (with feathering blending)

Based on the previous results this method seems to be an appropriate warping method for this stitching problem. Future captures using the real system mounted on a bus will determine if this algorithm meets all the Arcol project requirements.

5.4.1 Homography merging performance

Figure 5.8 shows a comparison between homography and homography merging. It can be clearly seen that homography merging offers better results than homography. Using simple homography the chessboards gets blurred and heavily misaligned. However, using homography merging the chessboards are perfectly aligned.



(a) Homography (b) Homography merging

Figure 5.8: Comparison between homography and homography merging

In addition to the image testing, both algorithms also have been tested analytically. The error metric used is the average distance between the theoretical points and the warped points using the different warping methods [Equation 5.1].

$$Error_{avg} = \frac{1}{N} \sum_{i=1}^{n} ||P_{theo}(i) - P_{warp}(i)|| \tag{5.1}$$

Figure 5.9 shows the error comparison between the different used methods –Homography or Homography Merging, each one with the three homography estimation methods:Least squares, Least median of squares and RANSAC.

On first approach it can be seen that Homography presents global poorer results than Homography merging. With Homography, left and right camera are the cameras which presents better results, whereas with Homography Merging front and rear cameras presents better results. Referring to the three estimation methods, all of them present similar performance with Homography Merging, whereas with Homography Least squares presents the best results, and RANSAC offers the poorest results.

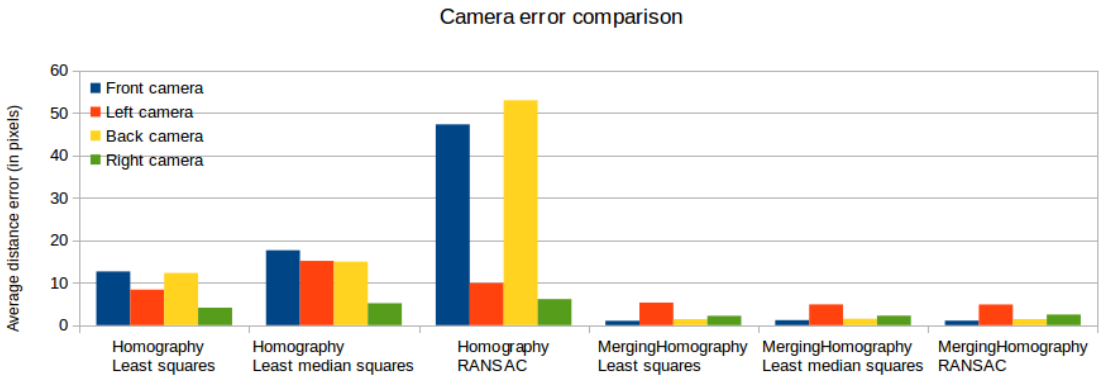


Figure 5.9: Error comparison between methods and cameras

This errors can be caused by two factors: errors on the point acquisition or errors on the chessboard placement. To a better results understanding the two errors are simulated. Point acquisition errors are simulated by adding a Gaussian noise to the detected image point and performing all the warping process. Then, the original image key points are compared to the ground key points. This procedure is done 100 times and averaged to obtain the Gaussian effect approximation. The results obtained can be seen in Figure 5.10.

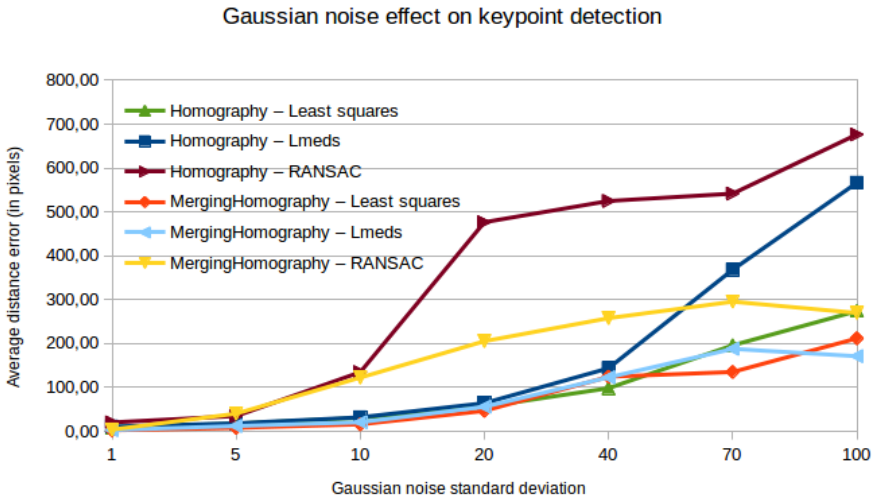


Figure 5.10: Error applying Gaussian noise on the detected key points.

As it can be seen in Figure 5.10, error increases with the Gaussian energy. Homography methods –especially RANSAC and Least Median of Squares– are heavily affected by Gaussian noise. RANSAC presents poor results both with Homography and Homography Merging, whereas Lmeds presents better results with MergingHomography. In conclusion, the MergingHomography method is more robust to noise on the input, but there are no difference with the best Homography method (Least squares).

Another error source that has to be taken into account is the chessboard misplacement on the ground. This error has been modelled by adding a shift to each chessboard point –error modelled as a Gaussian– and test the performance with this moved chessboard. The results obtained can be seen in Figure 5.11.

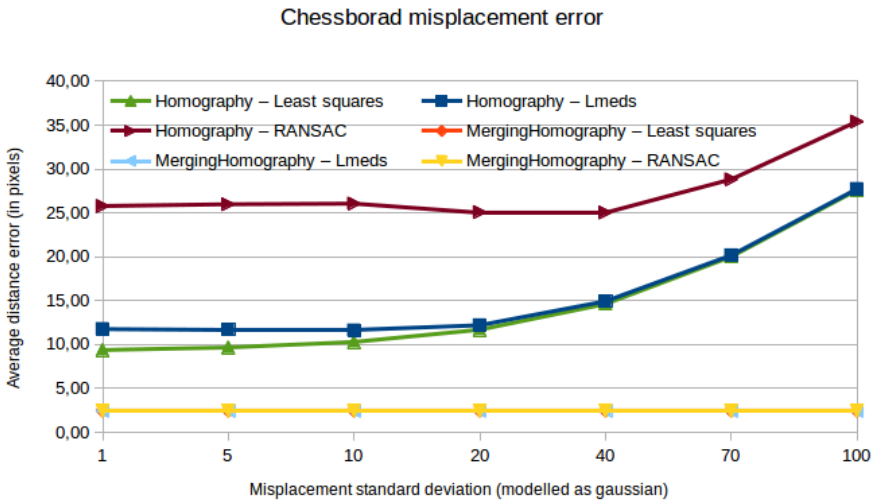


Figure 5.11: Error comparison moving the whole chessboard

It can be clearly seen that Homography Merging, regardless the used homography estimation method, is not affected by chessboard shifting. This is especially useful to minimize the human errors produced during the calibration step. Based on this results, the errors obtained in Figure 5.9 can be analyzed. Greater errors in front and rear cameras can be caused by chessboard misalignment, whereas errors in left and right camera can be caused by noise in the registration process.

5.4.2 Hardware application

In the final application the warping algorithm must be applied in real-time sequences (30fps). Thus, the final warping algorithm –homography merging– is a good candidate to be implemented on FPGA. The homography merging algorithm is a mapping-based algorithm. The mapping scanning is an high-parallelizable procedure. Thus, this method is suitable to be applied in real time.

Moreover, the final undistortion algorithm –outside this project scope– is also a mapping application. Thus, both algorithms could be concatenated in a single step. In this case, the warping step would not add any overhead to the stitching process.

This results show that Homography Merging is an appropriate method to the Arcol stitching pipeline. Moreover, the results show that this method can not only obtain good results with this specific scenario but also can adapt to environments with more error, with a precision higher than the sate-of-the art Homography method.

CHAPTER 6

BUDGET

This TFG project has been developed jointly with the GPI team. The budget presented is divided in Activities. Each Activity corresponds to an specific task, and the division has been done according to the equipment and staff needed to develop each tasks. In this budget are considered not only the activities developed in this project scope, but also the activities that are required to perform this project development.

The Activities that had been taken into account are:

Activity 1: Recording I. Captures done using cones as a pattern.

Activity 2: Recording II. Captures done using a chessboard as a pattern.

Activity 3: Undistortion. Test the different undistortion algorithms.

Activity 4: Registration, warping and blending. This Activity includes all the algorithms and methods tested in this TFG project scope and are directly related to the Arcol project.

Activity 5: Documentation. Additional activity that includes all the tasks done outside the Arcol project scope -TFG writing, additional tests, etc.-.

Each Activity summarized cost is the following:

	Activity	Equipment	Workers	Subtotal
1	Cone recording	197.06 €	290.00 €	487.06 €
2	Chessboard recording	205.83 €	290.00 €	495.83 €
3	Undistortion	46.70 €	1,030.00 €	1,076.70 €
4	Registration. warping and blending	442.00 €	21,000.00 €	21,442.00 €
5	Documentation	66.00 €	3,800.00 €	3,866.00 €
			TOTAL	**27.367,59€**

Each activity detailed costs can be found in Appendix B. This Appendix also contains each equipment detailed amortization and each worker salary.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This project has described the procedure followed to obtain an specific image stitching algorithm. This algorithm had to merge the image from four cameras located around a bus in a single 360°view image. The work described in this project has been done in a multidisciplinary UPC team working in a project commissioned by the Arcol company.

First of all, all the requirements and specifications inherited from the Arcol project have been stated. In basis on these requirements, a research in the current stitching methods has been done, including both commercial systems and high-level algorithms.

The next part included in this project have shown the used methods in the steps inside this TFG scope. From the four different stitching steps –undistortion, registration, warping and blending–, only warping have been fully developed in this project. Table 7.1 shows each step internal tasks and its final status.

Stitching step	*Internal task*	*Status*
Undistortion	Search and test undistortion algorithms	Done by another team member
	Apply the final algorithm	Done by the project author
Registering	Automatic registering	Not done
	Manual registering	Done by the project author
Warping	Search and test warping algorithms	Done by the project author
	Apply the final algorithm	Done by the project author
Blending	Apply provisional algorithm	Done by the project author
	Search and test blending algorithm	Not done
	Apply the final algorithm	Not done
Unghosting	Segment above ground objects	Not done
	Define an algorithm	Not done
	Apply the algorithm to the stitching	Not done

Table 7.1: Internal tasks status

Finally, the last part of this project shows the results obtained in the warping process. It can be extracted from the results that the developed algorithm beats the state-of-the-art algorithms for this specific application. As it can be seen in Chapter 5, the *Homography merging* algorithm developed in this project obtains better results than simple homography in this specific application.

Regarding to the initial goals of this project, the results obtained in each point can be summed up as follows:

Representative points automatic detection. The automatic key points detection has been left outside this project scope, and has been left to a future development.

Estimate the warping parameters. The warping parameters and warping algorithm had been estimated successfully.

Blending the results on the final stitching. Blending algorithms have been tested and the feathering selected offers an acceptable performance for the project.

Follow the requirements stated by the Arcol project. Regarding the Arcol specifications, this project is fully adapted to be used with the specific hardware and software. Moreover, the land lines discontinuity has been fully fulfilled. Exposure compensation and avoiding host images has been left to the Arcol project future development.

The overall project assessment is that, although not completely fulfilling all the goals stated at this project beginning, the results obtained in the developed parts are fully satisfactory. At the project start, the warping process has stated to be a trivial part, by simply applying the homography method. However, the development has arose many issues a priori not taken into account. These difficulties had recommended a change in the project direction, focusing on obtaining a more precise warping algorithm. This development sprung good results –and many discarded algorithms– and establish a base-line for the future development.

This project had left the development in a solid stage, with several work strands to be followed. First of all, an automatic registration algorithm can be implemented on the final product. This algorithm should automatically detect the chessboard corners and automatically do the key points registration. Another opened work strand is the definitive blending algorithm, directly bounded up with the above ground object treatment. In this part, an algorithm has to be defined to manage this objects and obtain the minimum image distortion.

BIBLIOGRAPHY

[1] *RGB Bayer Color and MicroLenses*. URL: http://www.siliconimaging.com/RGB%20Bayer.htm (visited on 06/19/2014).

[2] *Around View Monitor | NISSAN | TECHNOLOGICAL DEVELOPMENT ACTIVITIES*. URL: http://www.nissan-global.com/EN/TECHNOLOGY/OVERVIEW/avm.html (visited on 03/24/2014).

[3] *003 BMW X5 HCE-500 TOPVIEW calibration setup. - YouTube*. URL: https://www.youtube.com/watch?v=Ib1WhZIPsKk (visited on 06/01/2014).

[4] Nissan. *Getting a Virtual Bird's Eye View on Things*. 2014. URL: /EN/TECHNOLOGY/MAGAZINE/around_view_monitor.html (visited on 03/24/2014).

[5] *Nissan Boxenstopp - Around View Monitor*. Apr. 2013. URL: http://www.youtube.com/watch?v=s19dfek9A4U&feature=youtube_gdata_player (visited on 05/26/2014).

[6] *Around ViewÂ® Monitor - Technology - Infiniti Now*. URL: http://www.infinitiusa.com/now/technology/around-view-monitor.html (visited on 06/02/2014).

[7] *Audi S8 360-degree View Camera - YouTube*. URL: https://www.youtube.com/watch?v=LIuUbHSFDlI (visited on 06/01/2014).

[8] *Volskswagen Touareg bird view - YouTube*. URL: https://www.youtube.com/watch?v=tAInPzsUZ3o (visited on 05/31/2014).

[9] *Multi-Angle Vision Full-Perimeter Monitoring System/Ensure Safety and Reliability/ Products & Technology | FUJITSU TEN*. URL: http://www.fujitsu-ten.com/business/safety/multiangle/ (visited on 03/24/2014).

[10] *LR4 | Premium 4x4 | FEATURES | Surround camera system | US (English)*. URL: http://discovery.landrover.com/us/explore/features/surround-camera-system (visited on 06/01/2014).

[11] *Land Rover Discovery 4/ LR4 Surround Cameras Instructional Video - YouTube*. URL: https://www.youtube.com/watch?v=8fZBe5r1Pao (visited on 06/01/2014).

[12] *Stitching Pipeline - OpenCV 2.4.9.0 documentation*. URL: http://docs.opencv.org/modules/stitching/doc/introduction.html (visited on 06/21/2014).

[13] *Computer Vision*. URL: http://sse.tongji.edu.cn/linzhang/CV14/Projects/panorama.htm (visited on 06/20/2014).

[14] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010. URL: http://szeliski.org/Book/ (visited on 04/09/2014).

[15] Scott Schaefer, Travis McPhail, and Joe Warren. "Image deformation using moving least squares". In: *ACM Transactions on Graphics (TOG)*. Vol. 25. ACM, 2006, pp. 533–540. URL: http://dl.acm.org/citation.cfm?id=1141920 (visited on 04/09/2014).

[16] *Distorting – IM v6 Examples*. URL: http://www.imagemagick.org/Usage/distorts/ (visited on 04/08/2014).

[17] *Image Blenders - OpenCV 2.4.9.0 documentation*. URL: http://docs.opencv.org/modules/stitching/doc/blenders.html (visited on 06/21/2014).

[18] Peter J. Burt and Edward H. Adelson. "A multiresolution spline with application to image mosaics". In: *ACM Transactions on Graphics (TOG)* 2.4 (1983), pp. 217–236. URL: http://dl.acm.org/citation.cfm?id=247 (visited on 06/21/2014).

[19] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. URL: http://www.blender.org/ (visited on 06/21/2014).

[20] *Blender (software) - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Blender_(software) (visited on 06/21/2014).

[21] John M Blain. *The Complete Guide to Blender Graphics: Computer Modeling and Animation*. CRC Press, 2012.

[22] *"Fly" - Blender Modeling Timelapse on Vimeo*. URL: http://vimeo.com/2879397 (visited on 06/21/2014).

[23] *Blender Documentation Contents - Blender 2.70.5 - API documentation*. URL: http://www.blender.org/documentation/blender_python_api_2_70_5/ (visited on 06/21/2014).

[24] *Camera calibration With OpenCV - OpenCV 2.4.9.0 documentation*. URL: http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html (visited on 06/29/2014).

[25] *OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab - Davide Scaramuzza*. URL: https://sites.google.com/site/scarabotix/ocamcalib-toolbox (visited on 06/29/2014).

[26] *Camera Calibration and 3D Reconstruction - OpenCV 2.4.9.0 documentation*. URL: http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findhomography#findhomography (visited on 06/24/2014).

[27] Eric W. Weisstein. *Least Squares Fitting – from Wolfram MathWorld*. URL: http://mathworld.wolfram.com/LeastSquaresFitting.html (visited on 06/24/2014).

[28] "Least Median Of Squares Regression". In: *Journal of the American Statistical Association* (Dec. 1984), pp. 871–880. URL: http://spider.ipac.caltech.edu/staff/fmasci/home/statistics_refs/LeastMedianOfSquares.pdf (visited on 06/24/2014).

[29] Marco Zuliani. "RANSAC for Dummies". In: *With examples using the RANSAC toolbox for Matlab and more* (2009).

[30] *Inverse distance weighting - Wikipedia, the free encyclopedia*. URL: http://en.wikipedia.org/wiki/Inverse_distance_weighting (visited on 06/29/2014).

APPENDIX A

TIME PLAN

This Appendix details the work division and the planning for the current TFG, and the first version of the time plan was defined in the Project Proposal and Workplan. Due to changes on the Arcol project schedule and issues found, this time plan has changed during the development of this project.

The first time plan was split in 9 work packages, that can be summed up as follows:

WP#1: *Documentation.* Retrieve documentation about the state-of-the-art in automotive image stitching. Retrieve information about current stitching systems in photography.

WP#2: *Captures.* Capture the camera images to have input data to perform all the tests.

WP#3: *Undistortion.* Undistort the images in order to have straight and parallel lines on the fisheye distorted areas of the image.

WP#4: *Keypoints.* Define an algorithm to auto-detect the keypoints (reference points on the image). The goal is to use cones on the ground as keypoints.

WP#5: *Correspondences.* Once the keypoints are correctly detected, establish a 1-by-1 relation between keypoints of different camera views (front, sides, rear).

WP#6: *Warping.* Define the strategy to deform the image with a common points set. Select the optimal background plane (ground, spherical, cylindrical) to minimize parallax errors.

WP#7: *Blending.* Define the strategy to warp the images on the overlapped area. 50% overlapping (mixture) is only applied for testing purposes. The blending process will be focused on the definitive images, since the image resolution is a significant parameter.

WP#8: *Optimization.* Once all the different parameters for warping have been studied, analyze the results obtained and make the definitive code.

WP#9: *Writing.* Write the final degree project. Summarize the process done through the project and state the main results.

This time plan is oriented to perform a complete stitching, exploring all the fields of this project. However, this project has focused mainly on the warping algorithms, from the simulation in a 3D creation environment to the development of algorithms adapted to the specific stitching problem. Thus, the above presented work package division has been slightly modified to reflect the work done on this project.

First of all, the state of the art in stitching algorithms uses the homography method. This method assumes that the image has been fully undistorted. In this case, however, the undistortion parameters are computed once and the images are not fully undistorted. This has caused that the warping process, which should have been straightforwardly done, was driven to a far more complex algorithm.

Another issue that had changed this project timeplan is the reschedule of the recordings. On February, the definitive recording with the four cameras was scheduled to be done in May. However, this recording has been delayed to July, which made impossible to include the work done with this capture on this project. However, additional captures with provisional cameras had been done during the development of this TFG. These captures have allowed to perform all the non-virtual tests presented on this document.

These changes in the time plan have motivated majors changes in the work packages division. First of all, the work packages *WP#4: Keypoints* and *WP#5: Correspondences* had been merged to one Work Package, *Registering*, that summarizes all the work done on the two Work Packages. In addition to this major change, other minor changes in the internal tasks of the Work Packages have also been done. The complete work division done in this project can be found from Table A.1 to Table A.8.

<table>
<tr><td>Project: Real-time image stitching for 360° vision systems</td><td>WP ref: #1</td></tr>
<tr><td>Major constituent: Documentation</td><td>Sheet 1 of 8</td></tr>
<tr><td rowspan="2">Short description:
Retrieve documentation about the state-of-the-art in automotive image stitching.
Retrieve information about current stitching systems in photography.</td><td>Start week:
1st (1st January)
End week:
5th (5th January)</td></tr>
<tr><td>Duration:
5 weeks</td></tr>
<tr><td colspan="2">Internal task T1: Automotive stitching.
Internal task T2: Cylindrical stitching.
Internal task T3: Planar stitching.
Internal task T4: Spherical stitching.
Internal task T5: Summarize the state-of the-art</td></tr>
<tr><td colspan="2">Deriverables:
Week 05: State-of-the-art summary.</td></tr>
</table>

Table A.1: *Documentation* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #2
Major constituent: **Captures**	*Sheet* 2 *of* 8
Short description: Capture the camera images to have input data to perform all the tests.	*Start week:* – *End week:* –
	Duration: 3 captures
Internal task T1: Capture 1. *Internal task T2:* Capture 2. *Internal task T3:* Capture 3.	
Deriverables:	

Table A.2: *Captures* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #3
Major constituent: **Undistortion**	*Sheet* 3 *of* 8
Short description: Undistort the images in order to have straight and parallel lines on the fisheye distorted areas of the image.	*Start week:* 6th (1st February) *End week:* 17th (4th April)
	Duration: 9 weeks
Internal task T1: Choose the proper undistortion system. *Internal task T2:* Create the binding functions from the selected system to the software.	
Deriverables: Week 17: Binding functions.	

Table A.3: *Undistortion* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #4
Major constituent: **Registering**	*Sheet* 4 *of* 8
Short description: Establish a 1-by-1 relation between characteristic points on the image and reference points on the canvas.	*Start week:* 18th (1st May) *End week:* 20th (3rd may)
	Duration: 3 weeks
Internal task T1: Define the proper calibration system. *Internal task T2:* Create the functions of the registering system.	
Deriverables: Week 20: Registering functions.	

Table A.4: *Documentation* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #5
Major constituent: **Warping**	*Sheet* 5 *of* 8
Short description: Define the strategy to deform the image with a common points set. Select the optimal background planeand warping strategy to minimize misalignments between images.	*Start week:* 8th (2nd February) *End week:* 23th (1st June)
	Duration: 12 weeks
Internal task T1: Cone calibration. *Internal task T2:* Warping with cone images. *Internal task T3:* Search and try other warping algorithms. *Internal task T4:* Chessboard calibration. *Internal task T5:* Definition of the warping algorithm.	
Deriverables: Week 20: Warping algorithm.	

Table A.5: *Warping* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #6
Major constituent: **Blending**	*Sheet* 6 *of* 8
Short description: Define the strategy to warp the images on the overlapped area. 50% overlapping (mixture) is only applied for testing purposes. The blending process will be focused on the definitive images, since the image resolution is a significant parameter.	*Start week:* 10th (1st March) *End week:* 23th (1st June)
	Duration: 3 weeks
Internal task T1: 50% blending. *Internal task T2:* Feathering blending.	
Deriverables:	

Table A.6: *Blending* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #7
Major constituent: **Optimization**	*Sheet* 7 *of* 8
Short description: Once all the different parameters for warping have been studied, analyze the results obtained and make the definitive code.	*Start week:* 23th (1st June) *End week:* 24th (2nd June)
	Duration: 2 weeks
Internal task T1: Optimize warping. *Internal task T2:* Optimize blending.	
Deriverables:	

Table A.7: *Optimization* work package

Project: Real-time image stitching for 360° vision systems	*WP ref:* #8
Major constituent: **Writing**	*Sheet* 8 *of* 8
Short description: Write the final degree project. Summarize the process done through the project and state the main results	*Start week:* 22th (5th March) *End week:* 27th (5th June)
	Duration: 6 weeks
Internal task T1: Collect information. *Internal task T2:* Arrange information. *Internal task T3:* Writing. *Internal task T4:* Coherence and syntax revision. *Internal task T5:* Grammar revision.	
Deriverables: Week 27: Final Degree Project.	

Table A.8: *Writing* work package

APPENDIX B

DETAILED BUDGET

As explained in Chapter 6, this TFG project has been developed jointly with the GPI team. In order to clarify this project budget, the different project parts had been grouped into Activities. An activity can be described as any event done during this project development with specific staff and equipment. From the different activities stated, only a part had been one by the project author. The remaining activities are prerequisite work done during the Arcol project development, and thus must be added to the project cost.

This Appendix will describe the detailed budget project. In the first part, material and staff costs are stated and computed. In the second part, this costs are splitted between the several activities.

B.1 Equipment and staff

First of all, the equipment usage and staff work is described. Table B.1 computes the equipment used amortization cost, based on the overall usage during the Activities. Following the table the exaplanation corresponding to each column can be found.

The staff needs are stated in Table B.2. This table includes the salary per hour corresponding to each worker, as well as the total worked hours and the final salary.

Equipment	C.Price(1)	R.Val(2)	Qty(3)	Us(4)	Ls(5)	C/H(6)	Total cost
Camera	5.00 €	0.50 €	2	28	28	0.16 €	4.50 €
Capture card	200.00 €	20.00 €	2	26	14	12.86 €	334.29 €
Laptop	900.00 €	90.00 €	2	26	2000	0.41 €	10.53 €
Storage device	40.00 €	4.00 €	2	20	50	0.72 €	14.40 €
Structure	100.00 €	10.00 €	1	20	30	3.00 €	60.00 €
Cones	0.50 €	0.05 €	20	120	240	0.00 €	0.23 €
Chessboard	200.00 €	20.00 €	1	6	120	1.50 €	9.00 €
Undistortion pattern	50.00 €	5.00 €	1	2	40	1.13 €	2.25 €
Personal computer	1,000.00 €	100.00 €	1	700	2000	0.45 €	315.00 €
Server cluster	3,000.00 €	300.00 €	1	620	10000	0.27 €	167.40 €
Office material	40.00 €	–	1	–	–	–	40.00 €
						TOTAL	**957.59 €**

Table B.1: Equipment amortization detailed budget

C.Price(1) Commercial price. Original price for each equipment.

R.Val(2) Residual value. Equipment residual value at its lifespan end.

Qty(3) Quantity. Equipment quantity used in this project.

Us(4) Usage (hours). Usage time of the equipment. It is computed as the sum of all the usage hours of the same equipment. For example, if two cameras are used 4 hours each one, the usage will be 8 hours.

Ls(5) Lifespan (hours). Maximum usage time for all equipment unit.

C/H(6) Cost/hour. Computed cost for each usage hour.

Total cost Total cost. Equipment total cost (Usage × Cost/hour).

Workers	Salary/hours	Total hours	Cost
Technician	15.00 €	14	210.00 €
Junior engineer	35.00 €	680	23,800.00 €
Senior engineer	50.00 €	28	1,400.00 €
		TOTAL	**25,410.00 €**

Table B.2: Workers need summary

B.2 Activities

This project budget has been splitted in 5 activities: *Recordings I*, *Recordings II*, *Undistortion*, *Registration, warping & blending* and *Documentation*. This activities are described in the following sections, and their internal budgets are stated.

B.2.1 Activity 1: Cone recording

The first recording was done using cones as ground pattern. For the capture two different cameras were used, changing the positions to obtain the 4 viewpoints. To obtain the camera images an evaluation capture card for each camera was needed –as long as the final camera drivers were still not available–. The images were recorded in an external storage using two laptops.

The recording was done in the Arcol facilities, next to the factory. The structure was mounted with several poles located imitating the camera bus location.

Material	Qty(1)	Us/u(2)	Cost
Camera	2	6	1.93 €
Capture card	2	6	154.29 €
Laptop	2	6	4.86 €
Storage device	2	4	5.76 €
Structure	1	10	30.00 €
Cones	20	6	0.23 €

Workers	Hours	Cost
Technician	6	90.00 €
Senior engineer	4	200.00 €

Total equipment	197.06 €
Total staff	290.00 €
Total activity	487.06 €

Table B.3: Activity 1 detailed budget

Qty(1) Quantity. Equipment quantity used in this Activity.
Us/u(2) Usage (hours). Usage time of each equipment item.
Cost Total cost of the equipment/worker.

B.2.2 Activity 2: Chessboard recording

The second capture was done using a chessboard pattern instead of cones. The setup and remaining equipoment used is the same than *Cone recording*.

Material	Qty(1)	Us/u(2)	Cost
Camera	2	6	1.93 €
Capture card	2	6	154.29 €
Laptop	2	6	4.86 €
Storage device	2	4	5.76 €
Structure	1	10	30.00 €
Chessboard	1	6	9.00 €

Workers	Hours	Cost
Technician	6	90.00 €
Senior engineer	4	200.00 €

Total equipment	205.83 €
Total staff	290.00 €
Total activity	495.83 €

Table B.4: Activity 2 detailed budget

Qty(1) Quantity. Equipment quantity used in this Activity.
Us/u(2) Usage (hours). Usage time of each equipment item.
Cost Total cost of the equipment/worker.

B.2.3 Activity 3: Undistortion

The undistortion procedure consists on recording a calibration pattern in different position. With this images, the undistortion parameters are computed using different algorithms.

Material	Qty(1)	Us/u(2)	Cost
Undistortion pattern	1	2	2.25 €
Camera	2	2	0.64 €
Capture card	1	2	25.71 €
Laptop	1	2	0.81 €
Storage device	2	2	2.88 €
Personal computer	1	20	9.00 €
Server cluster	1	20	5.40 €

Workers	Hours	Cost
Technician	2	30.00 €
Senior engineer	20	1,000.00 €

Total equipment	46.70 €
Total staff	1,030.00 €
Total activity	1,076.70 €

Table B.5: Activity 3 detailed budget

Qty(1) Quantity. Equipment quantity used in this Activity.
Us/u(2) Usage (hours). Usage time of each equipment item.
Cost Total cost of the equipment/worker.

B.2.4 Activity 4: Registering. warping & blending

This activity comprises all the development done inside this project scope. This includes the algorithm research, implementation and testing. For ths application the GPI server cluster was used to perform high computational cost tests.

Material	Qty(1)	Us/u(2)	Cost
Personal computer	1	600	270.00 €
Server cluster	1	600	162.00 €
Office material	1	–	10.00 €

Workers	Hours	Cost
Junior engineer	600	21,000.00 €

Total equipment	442.00 €
Total staff	21,000.00 €
Total activity	21,442.00 €

Table B.6: Activity 4 detailed budget

Qty(1) Quantity. Equipment quantity used in this Activity.
Us/u(2) Usage (hours). Usage time of each equipment item.
Cost Total cost of the equipment/worker.

B.2.5 Activity 5: Documentation

Additional activity that includes all the tasks done outside the Arcol project scope. The current document writing, additional tests performing and minor tasks not directly related are included in this Activity.

Material	Qty(1)	Us/u(2)	Cost
Personal computer	1	80	36.00 €
Office material	1	–	30.00 €

Workers	Hours	Cost
Junior engineer	80	2,800.00 €
Senior engineer	20	1,000.00 €

Total equipment	66.00 €
Total staff	3,800.00 €
Total activity	3,866.00 €

Table B.7: Activity 5 detailed budget

Qty(1) Quantity. Equipment quantity used in this Activity.
Us/u(2) Usage (hours). Usage time of each equipment item.
Cost Total cost of the equipment/worker.

APPENDIX C

PROJECT DOCUMENTATION

The attached document *Refman-Arcol-20140618.pdf* contains the project documentation using the Doxygen tool. This documentation has been automatically generated and contains the functions implemented before June 18th. As long as this project is still in development, there is no final version.

This documentation includes all the generated functions grouped in the namespace *arcolv*. This namespace includes classes and functions used in the undistortion process –undistorter class–, the registering –features and chessfeatures class–, warping –mapping functions– and blending –blending functions–. There are also included many auxiliary functions to draw information on the image and to obtain numerical results.

The namespace *ocamlib* contains the Ocamlib undistorion tools functions. This functions had been downloaded, and only the comments had been adapted to the doxygen environment.

A toy example using the documentation functions could be an stitched image generation with MergeHomography method. The following code shows the procedure done to perform an image warping. Although the code is not functional, it shows the procedure needed to perform the operation.

```
//Feature configuration
arcolcv::ChessFeatures chessFeat;
chessFeat.read(pathfeatures);
chessFeat.setSizes(outsize, border);

//Undistortion configuration
arcolcv::Undistorter und=
    arcolcv::Undistorter(pathcameraparams,sf);

//Registration points acquisition
std::vector< Point2f > obj1A, obj1B, ..., obj4A;
std::vector< Point2f > scene1A, scene1B, ..., scene4A;
chessFeat.getChessFeatures(1,obj1A,scene1A,"A");
...

// Find the Homography Matrices
Mat H1A = findHomography( obj1A, scene1A, homographyMethod );
...
```

```
// Transform homographies to mappings
cv::Mat mapx1A, mapx1B, ..., mapx4A;
cv::Mat mapy1A, mapy1B, ..., mapy4A;
arcolcv::transform_homography(H1A, mapx1A, mapy1A, outsize);
...

//Merge homographies
cv::Mat mapx1,..., mapx4;
cv::Mat mapy1,..., mapy4;
arcolcv::mergeHomographiesX(mapx1B, mapy1B,mapx1A,
    mapy1A,mapx1, mapy1,x1,x2);
...

//Read images
Mat image1, image2, image3, image4;
Mat uimage1, uimage2, uimage3, uimage4;
Mat deform1, deform2, deform3, deform4;
image1=imread(pathcam1);
...

//Undistort images
und.undistortImage(image1,uimage1);
...

//Warp images
cv::remap(uimage1,deform1,mapx1,mapy1, cv::INTER_LINEAR,
    BORDER_CONSTANT,cv::Scalar(0,0,0) );
...

//Blend images
cv::Mat output;
std::vector<cv::Mat> input;
std::vector<cv::Mat> alphas;
cv::Mat alph1, alph2, alph3, alph4;
input.push_back(deform1);
...
arcolcv::calcAlphaImage(deform1,alph1,1);
alphas.push_back(alph1);
...
arcolcv::blendingFeather(input, alphas,output);

//Write warped image
cv::imwrite( "merged.png", output );
```

Code C.1: Tool example

The diferent functions seen in the documentation can be combined in a tool as seen above to perform any of this project stitching. Thus, the framework modularity will allow to include the future developments in a straightforward procedure.

APPENDIX D

ADDITIONAL RESULTS

The additional file *Results_ HomogrpahyMerging.mp4* contains an stitched sequence using the images recorded in the Arcol facilities. The input images are provided by the fish eye cameras. This cameras are located on four poles top simulating the bus sides centres. Additionally, four more poles are located on the bus corners.

The sequence contain the chessboard pattern calibration on the top left and bottom right sides. It has to be taken into account that, due to recording restrictions, only two cameras have been recorded simultaneously. Thus, this sequence only presents synchronism between top and left cameras, and bottom right corners. The overlapping areas between cameras where the chessboards are located are the ones that are being synchronized.

The video shows three different steps of the stitching process. First of all, the four input sequences are shown on a grid. This grid is only shown for the first 200 images from the 1074 images compounding the whole test sequence. As stated before, this input images are synchronized two-by-two. Figure D.3 shows a screenshot of this stage.

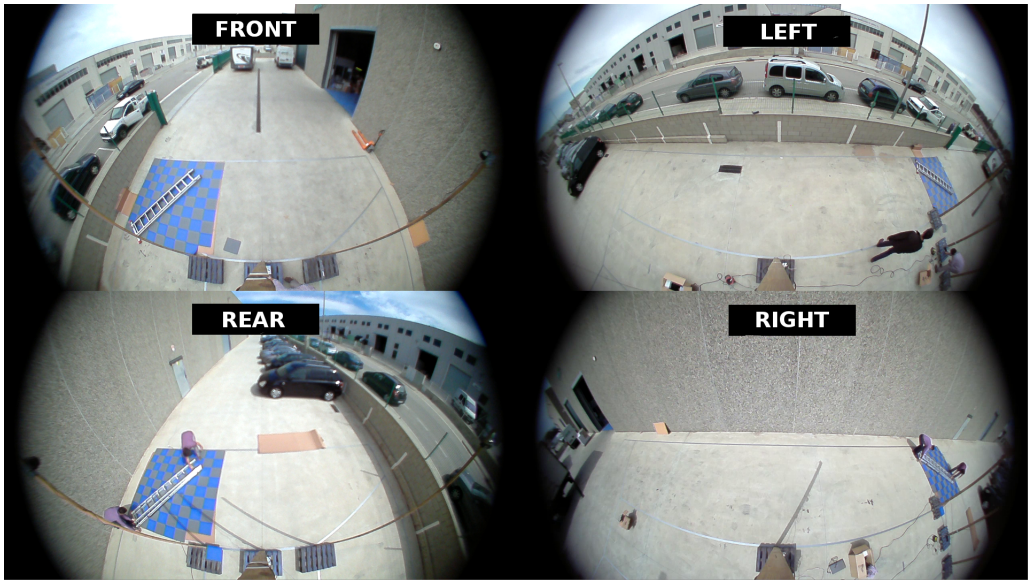


Figure D.1: Original images

The next part shows the undistorted images provided by the OCamLib undistortion algorithm. The camera paremeters are computed two by two (1-3 and 2-4), and with an *sf* parameter 12. An screenshot of the undistorted grid can be seen in Figure **??**.

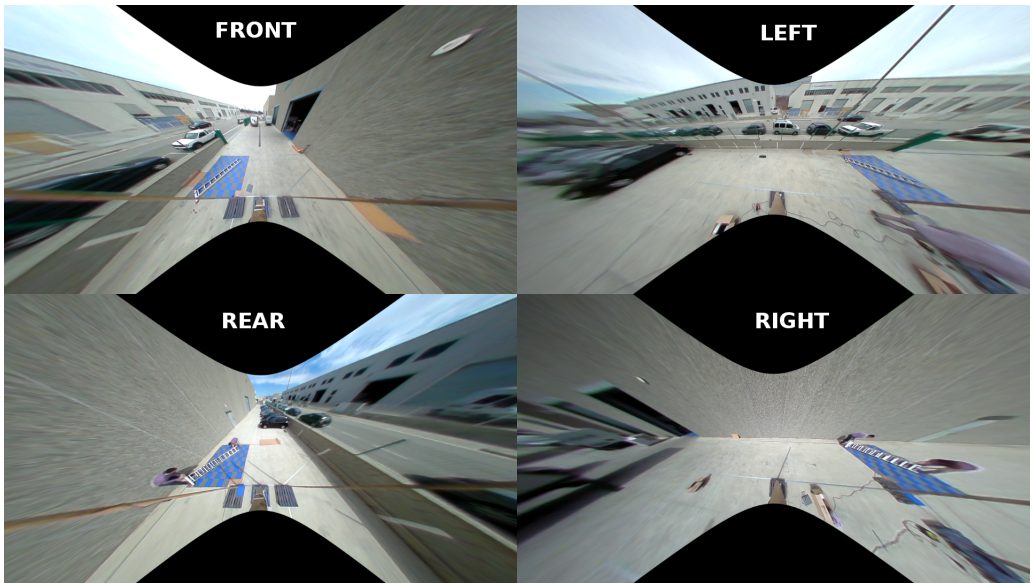


Figure D.2: Undistorted images

Finally, the third video part shows the fully undistorted sequence at 15fps. To obtain this sequence the Homogrpahy Merging algorithm has been used. Form every chessboard, 16 correspondence points have been established. This keypoints are located on the final image corresponding to the four chessboards. Due to the video aspect ratio, the image borders are not equal. The blending strategy used in the final video is Feathering. Figure Figure **??** shows an screenshot of the final stitched image.

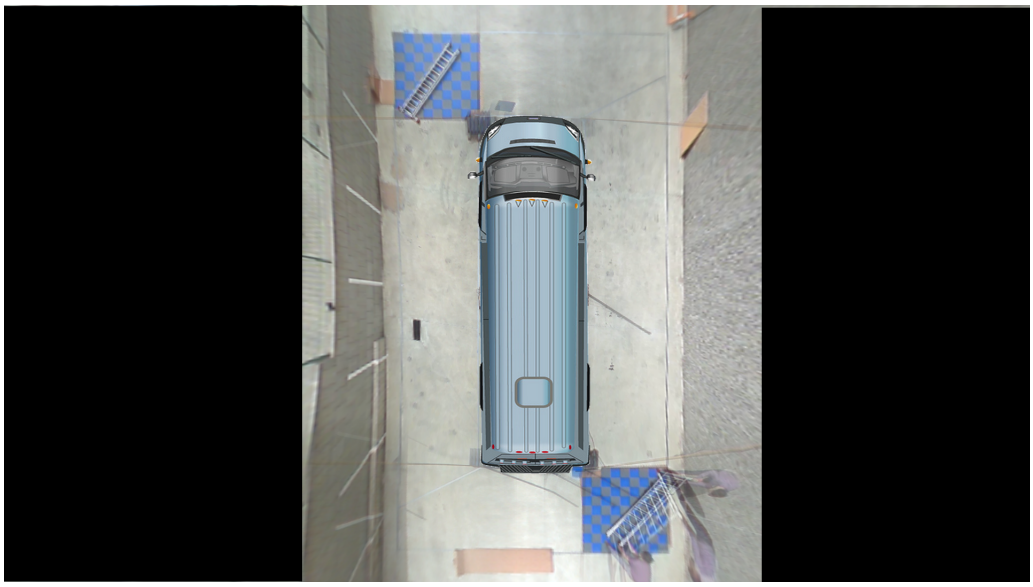


Figure D.3: Final stitched image