Universidad Politécnica de Madrid
Departamento De Automática, Ingeniería Electrónica e Informática
Industrial

Escuela Técnica Superior de Ingenieros Industriales

***GRAPH SEARCH AND ITS APPLICATION
TO IMAGE MATCHING***

Tesis Doctoral

Autor: Jorge Artieda Trigueros

Ingeniero Industrial

Tutor: Pablo San Segundo Carrillo

Dr. Ingeniero Industrial

2017

Tribunal nombrado por el Magnífico y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día _____ de _____ de 2017

Presidente: _____

Vocal: _____

Vocal: _____

Vocal: _____

Secretario: _____

Suplente: _____

Suplente: _____

Realizado el acto de defensa de la Tesis el día _____ de _____ de 2017 en la E.T.S.I.I.

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

To my family because each sleepless hour has been worth it

ABSTRACT

The main innovation presented in this thesis is a novel filtering technique for visual feature matching that takes into account multiple hypothesis. This algorithm, named CCMM, uses graph techniques to efficiently evaluate feasibility and joint compatibility of a set of possible matches. Specifically, the problem is reduced to a maximum clique search over an association graph and state-of-the-art algorithms are applied to solve it. Another contribution of this work is the tailoring of an exact clique solver for this application; the new algorithm is denoted BBMCW. BBMCW uses a bitstring formulation with *sentinel* bits to limit the boundaries of the semi-sparse bit-blocks that arise from the specific structure of the CCMM graphs.

The CCMM algorithm is then applied to the Structure from Motion problem. There, it improves the reconstruction results and helps to maintain overall stability. In the context of the Structure from Motion problem, this thesis presents a further contribution concerning the initialization of the bundle adjustment optimization step. In the first place, it shows that initialization is critical for a successful reconstruction of the scene. Moreover, this work also describes several novel strategies for initialization and compares the scene reconstructions with those obtained by known strategies. The reported results show that ordering frames according to the minimum distance to a central point in an error graph improves state-of-the-art bundle adjustment.

RESUMEN

La principal innovación presentada en esta tesis es una innovadora técnica de filtrado para la asociación de puntos característicos entre pares de imágenes. Esta técnica tiene en cuenta múltiples hipótesis en la asociación de puntos. Este algoritmo, al que hemos llamado CCMM usa técnicas de grafos para evaluar de forma eficiente la compatibilidad entre un conjunto de posibles parejas. En concreto el problema se reduce a la búsqueda de un máximo clique en un grafo donde se utilizan la últimas y más eficientes técnicas para su resolución. Otra innovación que se presenta en este trabajo es un algoritmo hecho a medida para la búsqueda exacta de cliques en los grafos generados por el algoritmo CCMM. Este algoritmo recibe el nombre de BBMCW y utiliza una formulación del problema basada en cadenas de bits a las que se les asigna unos centinelas que marcan el comienzo y el fin del bloque de bits. Estos bloques tienen una estructura semi-dispersa que surge de la forma en que se generan los grafos en el algoritmo CCMM.

Se aplica el algoritmo CCMM a la reconstrucción tridimensional de escenas a partir de imágenes monoculares obteniendo una mejora en los resultados y mejorando la estabilidad del algoritmo. En el marco de la reconstrucción tridimensional esta tesis presenta otra innovación en un paso concreto de la reconstrucción. Este paso es la inicialización del algoritmo de optimización que está demostrado ser un paso crítico en la reconstrucción. Este trabajo presenta varias estrategias compararemos con el algoritmo actualmente usado en el estado del arte. La estrategia de ordenado basada en la mínima distancia calculada a lo largo de un grafo creado a partir de los errores ha demostrado mejorar el rendimiento de la estrategia utilizada actualmente por el estado del arte.

ACKNOWLEDGEMENTS

I would like to thank my thesis tutor Pablo who has taught me to look for order into bits and nodes. Without his guidance and help this work could have never succeeded.

Many thanks also to my family for keeping me smiling every day and to my parents for supporting me on finishing this thesis.

Thanks to all my partners in the ETSIDI, they have given me the opportunity to focus on this project. Special thanks to the Paella lovers group for one of the best hours of the day.

Thanks to all my final grade project students, especially to Kunshen Zhang and Nerea Béjar. They have suffered my code and have provided useful insight for this work.

Finally, I would like to show my gratitude to Jesús Artieda who introduced me into the image processing world and provided me with endless challenges.

CONTENTS

1	INTRODUCTION	18
2	IMAGE REGISTRATION	19
2.1	CAMERA MODELS	20
2.1.1	<i>The Pinhole model</i>	21
2.1.2	<i>Push-broom models</i>	23
2.1.3	<i>Fisheye cameras</i>	24
2.2	FEATURE DESCRIPTORS	25
2.2.1	<i>SIFT</i>	26
2.2.2	<i>SURF</i>	28
2.2.3	<i>AKAZE</i>	29
2.3	FEATURE-MATCHING TECHNIQUES	33
2.3.1	<i>K-Nearest Neighbor (KNN)</i>	34
2.3.2	<i>K-d tree</i>	35
2.3.3	<i>Bag of Words approximation (BoW)</i>	36
2.4	FILTERING TECHNIQUES	38
2.4.1	<i>RANSAC</i>	40
3	CLIQUE MATCHING	45
3.1	THE CLIQUE MODEL FOR IMAGE ASSOCIATION	47
3.2	AN EXACT ALGORITHM FOR THE MAXIMUM CLIQUE PROBLEM	49
3.3	THE NEW CCMM ALGORITHM	55
3.4	METRICS AND CONFIGURATION	60
3.5	WATCHING SUBGRAPHS	64
3.5.1	<i>The sentinel encoding</i>	66
3.5.2	<i>The new BBMCW algorithm</i>	68
3.5.3	<i>Experiments</i>	71
3.6	RESULTS	75
3.6.1	<i>Initialization of the clique solver with a large initial clique</i>	75
3.6.2	<i>CCMM and RANSAC</i>	77
3.6.3	<i>BBMCW</i>	83
4	RECONSTRUCTION TECHNIQUES	87
4.1	FILTERING TECHNIQUES	88

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

4.2	OPTIMIZATION TECHNIQUES	95
4.2.1	<i>Levenberg-Marquardt</i>	96
4.2.2	<i>Bundle adjustment implementation</i>	97
5	BUNDLE ADJUSTMENT INITIALIZATION	100
5.1	GENERAL CONSIDERATIONS FOR FRAME ORDERING	106
5.2	MAXIMUM NUMBER OF POINTS IN COMMON	110
5.3	FRAME ORDERING BASED ON MINIMUM DISTANCE	112
5.4	FRAME ORDERING BASED ON THE MST	114
5.5	THE TRAVELLING SALESMAN STRATEGY	117
5.6	EXPERIMENTS AND RESULTS	119
6	THE DATA SETS	129
7	CONCLUSIONS	137
8	BIBLIOGRAPHY	140
9	APPENDICES	148
	APPENDIX I INITIALIZATION ALGORITHMS EVALUATION	149
	APPENDIX II BBMCW EVALUATION	160

LIST OF TABLES

TABLE 1 FEATURE EXTRACTION TIMES (SECONDS) BETWEEN SURF AND AKAZE.....	31
TABLE 2 AKAZE AND SURF QUALITATIVE COMPARISON. DATA SETS ARE DESCRIBED IN DETAIL IN CHAPTER 6.	32
TABLE 3. AN EXAMPLE OF THE MAXIMUM COMMON SUBGRAPH PROBLEM AS MAXIMUM CLIQUE	47
TABLE 4 METRICS USED FOR THE TESTS	63
TABLE 5 IMAGE DATA SETS USED IN THIS WORK. PARAMETER K REFERS TO THE MAXIMUM NUMBER OF KEYPOINTS FROM THE ORIGINAL IMAGE, WHICH ARE CANDIDATES TO MATCH A SINGLE KEYPOINT IN THE TARGET IMAGE. PARAMETER T IS A NORMALIZED THRESHOLD OF DISTANCE SIMILARITY BETWEEN PAIRS OF KEYPOINTS.	74
TABLE 6 ANALYSIS OF INITIAL SOLUTION ESTIMATION USING AMTS. G COLUMN REFERS TO THE NUMBER OF ASSOCIATION GRAPHS (PROBLEMS) STUDIED FOR EACH DATA SET, N IS THE AVERAGE NUMBER OF VERTICES AND D THE AVERAGE DENSITY. AMTS+ COLUMN REFERS TO THE NUMBER OF CASES IN WHICH TOTAL TIME IMPROVED.	75
TABLE 7.RESULTS FOR RANSAC-H WHICH USES A HOMOGRAPHY MODEL.....	78
TABLE 8.RESULTS FOR RANSAC-F WHICH USES A PROJECTIVE MODEL.	78
TABLE 9. RESULTS FOR CCMM.	79
TABLE 10 COMPARISON BETWEEN DIFFERENT EXACT CLIQUE SOLVERS FOR DIFFERENT DATA SETS. TIMES IN SECONDS ARE SHOWN ON THE FIRST TO THIRD COLUMNS AND COLUMNS FOUR AND FIVE CONTAINS THE RELATIVE IMPROVEMENT OF BBMCW OVER INCMaxCLQ AND BBMCi RESPECTIVELY.	84
TABLE 11 COMPARISON BETWEEN DIFFERENT EXACT CLIQUE SOLVING ALGORITHMS FOR DIFFERENT DATASETS. TIMES ARE NORMALIZED AGAINST THE BBMC ALGORITHM AND AVERAGED FOR THE COMPLETE DATASET. BBMCW IS THE ONLY ALGORITHM THAT IMPROVES BBMC TIMES.	86

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

TABLE 12. A TAXONOMY FOR INITIALIZATION STRATEGIES	108
TABLE 13. RECONSTRUCTION DATA FOLLOWING A MAXIMUM-COMMON-POINT ORDERING WITH TWO DIFFERENT BASELINES. #PNTS IS THE NUMBER OF 3D POINTS ON THE SCENE. #FRMS THE NUMBER OF FRAMES INSERTED IN THE SOLUTION. #PROJ IS THE TOTAL NUMBER OF PROJECTIONS. RMSE IS THE ROOT-MEAN-SQUARE ERROR OF THE PROJECTED POINTS.	110
TABLE 14. RECONSTRUCTION DATA FOLLOWING A MINIMUM DISTANCE BASED ORDERING OF FRAMES WITH DIFFERENT BASELINES. PARAMETERS ARE THE SAME AS IN PREVIOUS TABLES.	113
TABLE 15. MINIMUM SPANNING TREE-BASED FRAME ORDERING RESULTS. THE COLUMN MARKED AS <i>DEPTH FIRST</i> REFERS TO TRAVERSING EACH BRANCH OF THE MST IN ORDER. <i>MIN DIST</i> REFERS TO THE HEURISTIC REPORTED IN THE PREVIOUS SECTION (TABLE 14). FINALLY, MIN DIST MST SORTS THE NODES ACCORDING TO NON-DECREASING DISTANCE ALONG THE MST.	116
TABLE 16. TSP FRAME ORDERING RESULTS. ROWS WITH HYPHENS REFER TO DATA SETS WHERE THE ALGORITHM WAS NOT ABLE TO PRODUCE A VALID INITIAL SOLUTION.	118
TABLE 17. EXPERIMENTAL RESULTS FOR THE NUMBER OF 3D POINTS MATCHED CORRECTLY ACCORDING TO THE DIFFERENT SORTING CRITERIA. RESULTS ARE NORMALIZED ACCORDING TO THE BEST RESULT FOR EACH DATASET. AN INDEX OF 1 INDICATES THE HIGHEST NUMBER OF POINTS WAS MATCHED.	119
TABLE 17. COMPARISON BETWEEN REFERENCE ORDERING AND THE METHOD WITH BEST RESULTS. BEST OF THE TWO METHODS IS HIGHLIGHTED ON BOLD FOR EACH ROW.	121
TABLE 19. RECONSTRUCTION RESULTS FROM THE CORRIDOR DATASET. FIRST ROW SHOWS THE DATASET IMAGES AND A MATCHING SAMPLE. SECOND ROW SHOWS THE RECONSTRUCTION RESULT USING THE MINIMUM DISTANCE ORDERING (LEFT) AND MAXIMUM COMMON POINTS STARTING ON THE MINIMUM ERROR PAIR (RIGHT)	122
TABLE 20. RECONSTRUCTION RESULTS FROM THE FOUNTAIN-R25 DATASET. FIRST ROW SHOWS THE DATASET IMAGES AND A MATCHING SAMPLE. SECOND ROW SHOWS THE RECONSTRUCTION RESULT USING THE MINIMUM DISTANCE ORDERING (LEFT) AND MAXIMUM COMMON POINTS STARTING ON THE MINIMUM ERROR PAIR (RIGHT)	123

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

TABLE 18 PROPERTIES OF THE DATA SETS.....	130
TABLE19. IMAGE CHARACTERISTICS OF THE DATA SETS	132
TABLE 20. AERIAL SCENES	133
TABLE21. OUTDOOR BUILDINGS.....	133
TABLE 22. INDOOR.....	135
TABLE 23 OBJECTS	135
TABLE 24. OTHER DATASETS	136
TABLE 25 MAXIMUM POINTS STARTING ON A MINIMUM ECCENTRICITY PAIR	149
TABLE 26 MAXIMUM COMMON POINTS STARTING FROM THE MINIMUM ERROR PAIR	150
TABLE 27 MINIMUM SPANNING TREE TRAVERSED ON A DEPTH FIRST ORDER.....	151
TABLE 28 MINIMUM DISTANCE CALCULATED ALONG THE FULL GRAPH STARTING ON THE MINIMUM ECCENTRICITY PAIR	153
TABLE 29 MINIMUM DISTANCE ALONG THE FULL GRAPH STARTING ON THE MINIMUM ERROR PAIR.	154
TABLE 30 MINIMUM DISTANCE ALONG A MINIMUM SPANNING TREE	155
TABLE 31 MINIMUM DISTANCE ALONG THE FULL GRAPH STARTING FROM THE MINIMUM ECCENTRICITY POINT CALCULATED FROM THE MINIM SPANNING TREE.	157
TABLE 32 SEQUENTIAL ALGORITHM USING FILENAMES.	158
TABLE 33. TIME PERFORMANCE OVER A NUMBER OF REAL CORRESPONDENCE GRAPHS TAKEN FROM SIX IMAGE DATA SETS. HEADER <i>P</i> SHOWS AVERAGE DENSITY. TIMES ARE MEASURED IN SECONDS. COLUMNS CONTAINING <i>IMP</i> . SHOW THE RATIO BETWEEN THE ALGORITHM IN THE CORRESPONDING HEADER AND BBMCW.	161

LIST OF FIGURES

FIGURE 1. IMAGE REGISTRATION AS PART OF A 3D RECONSTRUCTION AND MOSAICING CHAIN	20
FIGURE 2. PINHOLE CAMERA MODEL	21
FIGURE 3. GRAPHICAL DERIVATION OF THE PINHOLE CAMERA MODEL	22
FIGURE 4. PUSH-BROOM CAMERA MODEL	23
FIGURE 5. FISHEYE CAMERA MODELS.....	24
FIGURE 6. DIFFERENCE OF GAUSSIAN IMAGE (DOG) TAKEN FROM THE US PATENT: SYSTEM AND METHOD FOR SIFT IMPLEMENTATION AND OPTIMIZATION US 8441489 B2	27
FIGURE 7. AN EXAMPLE OF BOX FILTER APPROXIMATION. THE IMAGE IS TAKEN FROM OPENCV DOCUMENTATION.	29
FIGURE 8. GAUSSIAN BLUR (TOP ROW) AND NON-LINEAR DIFFUSION (BOTTOM ROW).	30
FIGURE 9. AN EXAMPLE OF THE K-NEAREST NEIGHBOR HEURISTIC. FEATURE F IS MATCHED TO THE NEAREST FEATURE F2.	35
FIGURE 10. K-D TREE EXAMPLE. ON THE LEFT THE FEATURE SPACE SUBDIVIDED IN BINARY NESTED CONTAINER. ON THE RIGHT THE CORRESPONDING TREE STRUCTURE TO THE LEFT SUBDIVISION.....	36
FIGURE 11. EXAMPLE OF THE BAG OF WORDS (BOW) ALGORITHM	37
FIGURE 12. EPIPOLAR GEOMETRY. A POINT (IN RED) PROJECTED ON THE LEFT IMAGE (XL) DEFINES A LINE (EPIPOLAR LINE) ON THE RIGHT IMAGE (XR) CREATED BY THE DIFFERENT POSSIBLE DEPTHS OF THE TRIDIMENSIONAL POINT PROJECTED ON THE RIGHT IMAGE.....	42
FIGURE 13. THE ERROR USING THE FUNDAMENTAL MATRIX IS DEFINED AS THE DISTANCE D FROM THE MEASUREMENT (IN RED) ON THE RIGHT FRAME (XR) TO THE EPIPOLAR LINE DETERMINED BY THE POINT PROJECTION ON THE LEFT FRAME (XL).	43

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

FIGURE 14. IMAGE PROCESSING CHAIN FOR MOSAICING A 3D RECONSTRUCTION	46
FIGURE 15. AN EXAMPLE OF THE NEW CLIQUE ASSOCIATION MODEL.....	60
FIGURE 16. EXAMPLE FOR DEMONSTRATION PURPOSES. TO THE LEFT A SIMPLE UNDIRECTED GRAPH. TO THE RIGHT, THE BITSTRING ENCODING OF ITS VERTEX SET.	65
FIGURE 17. WATCHED BITBLOCKS	66
FIGURE 18. PERFORMANCE OF THE NEW ALGORITHM OVER THE <i>DINO</i> BENCHMARK. ON THE RIGHT, THE HEURISTIC ALGORITHM AMTS IS USED TO PRE-COMPUTE A LARGE INITIAL CLIQUE. ON THE LEFT, NO HEURISTIC IS EMPLOYED.....	76
FIGURE 19. PERFORMANCE OF THE NEW ALGORITHM OVER THE <i>HELIGRANDE</i> BENCHMARK. ON THE RIGHT, THE HEURISTIC ALGORITHM AMTS IS USED TO PRECOMPUTE A LARGE INITIAL CLIQUE. ON THE LEFT, NO HEURISTIC IS EMPLOYED.	77
FIGURE 20. MEAN NUMBER OF THREE OR MORE CONSECUTIVE FRAMES IN WHICH A POINT IS TRACKED CONSISTENTLY IN THE 6 DATASETS CONSIDERED.	80
FIGURE 21. TOTAL RUNNING TIMES IN MILLISECONDS (Y-AXIS) FOR ALL ALGORITHMS OVER THE <i>REDUCED</i> DATASET.	81
FIGURE 22. A SAMPLE OF FEATURE MATCHING FROM EACH OF THE REPORTED DATA SETS. IN GREEN (TOP) ASSOCIATIONS USING RANSAC-F; IN RED (BOTTOM) ASSOCIATIONS GIVEN BY THE NEW CCMM ALGORITHM.	83
FIGURE 23. BUNDLE ADJUSTMENT IMAGE PROCESSING CHAIN.....	100
FIGURE 24. FOUR POSSIBLE SOLUTIONS FOR CAMERA POSITIONS DERIVED FROM THE ESSENTIAL MATRIX.	103
FIGURE 25. PROJECTED POINTS ARE OBSERVED IN COORDINATES X AND X' , AND DUE TO MEASUREMENT ERRORS THEY DEFINE RAYS THAT DO NOT INTERSECT IN SPACE. AS A RESULT, THE MEASUREMENTS DO NOT SATISFY THE EPIPOLAR RESTRICTION. THIS HARDENS THE TRIANGULATION PROBLEM USING THE FUNDAMENTAL MATRIX.	104
FIG. 26. MST (HIGHLIGHTED EDGES) OF THE DEMONSTRATION GRAPH.....	116
FIG. 27. ORDERING ALONG MST, FOLLOWING A DEPTH-FIRST TRAVERSAL.....	116

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

FIG. 28. MINIMUM DISTANCE ORDERING ALONG THE MST	116
FIG. 29. MINIMUM DISTANCE ORDERING ALONG THE FULL GRAPH	116
FIGURE 30. ORDERING USING TSP STRATEGY	118
FIGURE 31 REDUCED DATASET RECONSTRUCTION	124
FIGURE 33 TEACUP DATASET RECONSTRUCTION	125
FIGURE 34 VUELO6 DATASET RECONSTRUCTION	126
FIGURE 35 FOUNTAIN-R25 DATASET.....	126
FIGURE 36 MERTON COLLEGE 1	127
FIGURE 37 MODEL HOUSE DATASET	127

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

1 INTRODUCTION

This thesis describes the use of graph analysis techniques to create 3D reconstructions from monocular imagery. This document is divided into 7 sections. First we describe the problem of Image Registration. Second we present the one off the innovations described in this thesis: the Clique Matching algorithm (CCMM). In the Chapter 4 we present one use case for CCMM, 3D reconstruction. Chapter 5 describes the second contribution of this thesis, which is using graph-based techniques for bundle adjustment optimization initialization. Chapter 6 describes all the image benchmarks used in this work. Chapter 7 summarizes conclusions and describes open research lines related to this work.

2 IMAGE REGISTRATION

Image registration is the process of transforming different sets of data into one reference coordinate system. In computer vision, data refers to sets of images from multiple viewpoints. Registration is necessary to be able to compare and make inferences over the data. We can find several examples of this problem. For example, aerial images must be converted to a common world coordinate reference frame before being used. This reference frame is usually on a geodesic projection, such as Mercator or Geographical. Medical Images constitute another example. This problem is similar to aerial photogrammetry, as the images need to be transformed into a common reference space, but with the added difficulty that areas, lines and other features, must be preserved with precision.

There are two important sides to image registration. The first one is the model, and the second one is how the model is adjusted to the observations. The model determines how an image is transformed into the common space. It also determines the number of parameters and, therefore, the minimum number of measurements to estimate it. The way the measures are done is also a very important aspect of image registration. Until the last decade, most of the measurements were done by hand by expert operators. This lead to tedious processing of the captured image to obtain the final corrected image. Recent

developments on image recognition have provided tools to create sets of point matches between two images. These tools have allowed to automate the processing chain but have created new challenges. One of them is that most of the adjustment algorithms only show good performance when the errors have a Gaussian distribution. However, the matching errors of the automated tools are usually of two types. The first one is related to the precision with which the algorithm is able to locate the points. This error does follow a normal distribution. The second one deals with the possibility that a pair of points are incorrectly matched. A wrong match creates a large error measurement that can greatly disturb the results of the overall adjustment. Points wrongly matched are called *outliers*, as opposed to *inliers* which are correctly matched points with a certain Gaussian error.

This chapter is divided as follows: first, we describe some typical models used for camera projection; we then introduce the feature descriptors that allow for automatic point matching; finally, we describe the state-of-the-art techniques to filter outliers from the matched set of points.

Figure 1. Image registration as part of a 3D reconstruction and mosaicing chain

2.1 Camera models

Camera models relate a point in the 3D space with its 2D projection in the image space. A camera model can be a generic mathematical function, such as a polynomial. Alternatively, it can be a specific function that physically models the image acquisition process. Any such model uses a set of parameters that define the position and orientation in the coordinate space (exterior parameters), as

well as the optics and projection (the *interior* parameters). In the following subsections a subset of common camera models will be briefly described. The list presented is by no means exhaustive, as there are many imaging systems that failed to be captured. They cover, however, most of the applications concerning this work.

2.1.1 The Pinhole model

The most common specific model for conventional cameras is called the *pinhole model*.

Figure 2. Pinhole camera model

This model (see Figure 2) is very useful to represent conventional cameras. The corresponding equations are derived from the focal length f parameter, which represents the distance of the projection plane to the optical center of the camera.

Figure 3. Graphical derivation of the Pinhole camera model

Figure 3 shows how the a light ray that goes from an object to the optical center of the camera C , crosses the projection plane in a coordinate v . v can be formulated in terms of the 3D coordinates of the point and the focal length. We can express this relationship in a linear form using homogeneous coordinates as follows:

$$\begin{bmatrix} w \cdot u \\ w \cdot v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

To take into consideration that the frame coordinates are not in the middle of the sensor, the center of the image(c_x, c_y) the above expression is changed to:

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Moreover, the model can further be extended to the case when the position of the camera (x_c, y_c, z_c) is not at the coordinates origin:

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} [R \quad T] \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (1)$$

Equation 1 shows the complete pinhole camera model. The model has two types of parameters: *intrinsic*, and *extrinsic*. The former, such as the focal point f , or the image center (c_x, c_y), remain constant during image acquisition, while the

latter (e.g. rotation and translation) are dynamic with respect to the camera movement.

2.1.2 Push-broom models

The so called push-broom models [1] are used in systems where the sensor is a line camera that shoots images as it moves along. This sensors are very common for satellite and medical imagery applications.

Figure 4. Push-broom camera model

The model derivation starts with a pinhole model of the linear array sensor:

$$\begin{bmatrix} w \cdot v \\ w \end{bmatrix} = \begin{bmatrix} f & pv \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}$$

which, builds the image as it moves along. Suppose that $u = t$

$$\begin{bmatrix} u \\ wv \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & f & pv \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t \\ y \\ z \end{bmatrix} \quad (2)$$

Then

$$\vec{x}(t) = \vec{x}_o - t\vec{V} = (x_o, y_o, z_o)^T - t(v_x, v_y, v_z)^T \quad (3)$$

combining (2 and 3) we arrive to the following model:

$$\begin{bmatrix} u \\ wv \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & f & pv \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/v_x & 0 & 0 \\ -v_y/v_x & 1 & 0 \\ -v_z/v_x & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$$

Alternatively, when the input points are referred to an external reference system, the model becomes:

$$\begin{bmatrix} u \\ wv \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & f & pv \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/v_x & 0 & 0 \\ -v_y/v_x & 1 & 0 \\ -v_z/v_x & 0 & 1 \end{bmatrix} (R| - R\vec{T}) \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$$

2.1.3 Fisheye cameras

Fisheye cameras are wide angle cameras that allow capturing panoramic or even hemispherical images. These type of cameras have highly complex optics difficult to model.

Figure 5. Fisheye camera models

The image projection usually is approximated as a function of the distance to the optical center, enhanced in some cases by other distortion functions. The simplest fisheye camera model is described in [2] by the following equations:

$$\begin{aligned}a &= x/z \\b &= y/z \\r^2 &= a^2 + b^2 \\ \theta &= \text{atan}(r)\end{aligned}$$

where θ is the angle between the main axis and the incoming ray. Several functions can be used to model the relationship between the incident angle and the distance to the optical center. Some of the most common ones are:

- *stereographic projection*: $r = 2f \tan(\theta/2)$
- *equidistance projection*: $r = f\theta$
- *equisolid angle projection*: $r = 2f \sin(\theta/2)$
- *orthogonal projection* $r = f \sin(\theta)$

Finally, $r(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + \dots$ is a widely accepted generic parametric model.

The projection is then estimated as a correction over the pinhole model with the following formulas.

$$\begin{aligned}x' &= (\theta_d/r)a \\ y' &= (\theta_d/r)b\end{aligned} \tag{4}$$

It follows from (4), when classical calibration is applied, that the (u, v) image coordinates can be estimated by the following equations:

$$\begin{aligned}u &= f_x(x' + \alpha y') + c_x \\ v &= f_y y' + c_y\end{aligned}$$

2.2 Feature descriptors

This section describes different feature descriptors that may be used to model features captured from images. Without loss of generality, we will henceforth use the term point, keypoint and feature indistinctly, since the points are the only features that concern this work.

Although commonly described together, we differentiate two stages in the process of describing features: I. keypoint generation and II. computation of

descriptors. The first stage outputs a list of interesting points from the image and the second assigns a descriptor value to each point.

The most primitive approximations used a small patch of the image to describe the feature. Afterwards, successive patches are matched against the first one using a correlation metric. This technique, however, encounters problems due to illumination, scale and movement changes. Examples in literature for this basic form of keypoint detection are the Harris corner detector [3] and the Shi-Tomasi Variant [4].

The latter approximations are typically supervised by detectors that compute keypoints in different scale spaces and assign a rotation to each point. This produces invariance to scale and rotation changes. Examples of these improved detectors and descriptors are ORB, BRIEF, BRISK, SIFT, SURF and AKAZE. The algorithms SIFT, SURF and AKAZE will be described in detail in the following subsections.

2.2.1 SIFT

The SIFT algorithm stands for Scale Invariant Feature Transform and was first described by Lowe [5]. It is based on the construction of a Difference of Gaussians (also DoG) stack. Local maxima and minima between levels of this stack are selected as the actual keypoints on the image where a descriptor will be calculated (see Figure 6). The DoG function $D(x, y, \sigma)$ is calculated as:

$$D(x, y, \sigma)_{ij} = L(x, y, k_i \sigma) - L(x, y, k_j \sigma)$$

Where $L(x, y, k_i \sigma)$ is the convolution of the image with a gaussian blur kernel.

Figure 6. Difference of Gaussian Image (DoG) taken from the US PATENT: System and method for SIFT implementation and optimization US 8441489 B2

Moreover, to achieve rotation invariance, each keypoint is assigned an orientation, estimated by calculating a gradient for each pixel and then creating a histogram with all the gradient orientations. The most frequent bin in the histogram is assigned to the keypoint as orientation. The gradient module and its orientation is estimated using equation (5):

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (5)$$

$$\theta(x, y) = \text{atan2}(L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y))$$

where $L(x, y)$ is the convolution of the image with a Gaussian kernel at pixel (x, y) , and $m(x, y)$ and $\theta(x, y)$ are the module and orientation of the estimated gradient respectively.

To assign a descriptor to the keypoint, a neighborhood of the keypoint is defined using the scale and orientation previously assigned to it. This neighborhood is

then divided into a 4x4 grid. Inside each cell of the grid, a histogram of magnitude and orientation is calculated in a similar way as the previously described histogram that determined orientation. Each histogram has 8 bins. Since we have 4x4=16 cells, each one typically with an 8 bin histogram, we can create a 128 element vector with all the values of the bins from all the histograms. This vector is further normalized to avoid the effects of illumination changes. The resulting unit vector is a robust descriptor of the keypoint, presenting a great level of distinctiveness.

2.2.2 SURF

Speeded-up robust features [6] is a faster version of the previously described descriptor and keypoint extractor SIFT. Similarly to SIFT, it is currently patented (under the US2009238460 patent) and may not be used without the authors' explicit consent.

SURF uses the Hessian matrix H instead of the Difference of Gaussians, which is defined as:

$$H(p, \sigma) = \begin{pmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{yx}(p, \sigma) & L_{yy}(p, \sigma) \end{pmatrix}$$

where $L_{xx}(x, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ of the image I in point x , and the same for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$.

The SURF algorithm approximates the second order Gaussian derivatives with box filters. This allows the algorithm to employ integral images thus reducing computational costs.

Figure 7. An example of Box filter approximation. The image is taken from OpenCV documentation.

Orientation assignment is performed applying the Haar-wavelet to the image both on the x and y directions in a circular neighborhood of the interest point. The most frequent orientation is the one finally assigned to the keypoint. Haar-wavelets are approximated by box filters which are faster to compute (see Figure 7).

Wavelet responses are used also to create the vector descriptor. The neighborhood of the keypoint is divided, as in the SIFT algorithm, in a 4×4 grid. A four dimensional vector for each cell is calculated as $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ and therefore a 64 length vector can be created joining all the vectors for each of the 4×4 grid. Also as in the SIFT algorithm, descriptors are normalized to a unit vector.

2.2.3 AKAZE

AKAZE algorithm [7] uses a diffusion equation to build a scale space that preserves boundaries and respects details. The classic non-linear diffusion equation is:

$$\frac{\partial L}{\partial t} = (c(x, y, t) \cdot \nabla L)$$

where div and ∇ are the divergence and gradient operators respectively, and L is the image luminance (a measure of the luminous intensity). Thanks to the

introduction of a conductivity function (6) in the diffusion equation, it is possible to make diffusion sensitive to the local image structure (see Figure 8).

Figure 8. Gaussian blur (top row) and non-linear diffusion (bottom row)

$$c(x, y, t) = \frac{1}{1 + \frac{|\nabla L_{\sigma}(x, y, t)|^2}{\lambda^2}} \quad (6)$$

In (6), the parameter λ is the contrast factor that controls the level of diffusion. It determines which edges will be preserved or will disappear.

The nonlinear scale space is built by discretizing the diffusion at different times t_i , $i = \{1, 2, \dots\}$. This creates a mapping $\sigma_i \rightarrow t_i$ and allows the creation of a Hessian matrix for each of the filtered images.

The actual feature descriptor proposed by the AKAZE author is known as Modified-Local Difference Binary (M-LDB), although SIFT or SURF descriptors, may also be used.

One of the weak points of the AKAZE predecessor, the KAZE algorithm, was its high computational cost. Note this is a critical issue, since these kind of algorithms are often required by real time applications, or applied over large image databases. In this work, we have also compared the new AKAZE algorithm with the SURF algorithm, both implementations were taken from the OpenCV library, version 3. This comparison is of a great interest, as AKAZE is released into the open domain and may be used freely, while SIFT and SURF are both patented. We now proceed with a brief summary of the results obtained.

To allow a fair comparison between AKAZE and SURF, we first adjusted the test configuration so as to obtain a similar number of keypoints. The number of extracted points has a large impact in the ensuing descriptor extraction stage.

Comparison results between SURF and AKAZE over 13 datasets (used for benchmarking throughout this work) are shown in Table 1.

Table 1 Feature extraction times (seconds) between SURF and AKAZE

<i>Data Sets</i>	<i>AKAZE</i>	<i>SURF</i>
<i>airview</i>	5.06	12.45
<i>albert</i>	2.68	3.27
<i>cantera</i>	62.52	168.33
<i>castle</i>	7.20	18.31
<i>dino</i>	4.55	4.61
<i>fountain</i>	7.78	12.58
<i>fpv1</i>	13.58	85.79
<i>heliGrande</i>	195.96	369.42
<i>herzjesu</i>	22.47	13.21
<i>kapel</i>	61.54	17.26
<i>piedra</i>	14.00	11.03
<i>reduced</i>	4.08	8.99
<i>ualberta</i>	390.27	118.91

As shown in Table 1, AKAZE has comparable speed performance with SURF and is occasionally faster in some of the tested instances.

Moreover, we also compare the quality of the descriptors obtained by both algorithms using our new image-matching-pair algorithm CCMM, contribution of this work. We proceed to report the results obtained in the comparison, omitting CCMM. For a detailed explanation of the latter we refer the reader to Chapter 3 of this work. To evaluate the descriptors we match image pairs of our data sets and consider the following 4 metrics:

- Total number of features detected: for each data set, the total number of keypoints extracted and correctly matched.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

- Total number of features after filtering: same as before but having removed outliers / points matched incorrectly.
- Number of features per pair: average number of points matched for each pair of frames.
- Number of frames with more than 100 correctly matched points, a threshold considered as a consistent match between frame pairs.

Table 2 AKAZE and SURF qualitative comparison. Data sets are described in detail in Chapter 6.

	<i>AKAZE</i>				<i>SURF</i>			
<i>Data Sets</i>	<i>Total</i>	<i>Filtered</i>	<i>Num</i> <i>/frame</i>	<i>Nº</i> <i>>100</i>	<i>Total</i>	<i>Filtered</i>	<i>Num</i> <i>/frame</i>	<i>Nº</i> <i>>100</i>
<i>airview</i>	1259	326	55.62	142	187	55	10.44	0
<i>albert</i>	1322	494	182.25	1461	488	136	64.98	907
<i>cantera</i>	1348	135	8	242	1047	411	11.1	348
<i>Castle</i>	1224	253	23.72	7	1259	648	28.88	8
<i>Dino</i>	1046	321	18.8	25	469	71	9.82	1
<i>fountain</i>	1246	205	32.76	28	866	311	28.37	22
<i>fpv1</i>	1437	472	116.61	2562	2936	1345	222.93	3343
<i>heliGrande</i>	1280	384	18.43	36710	854	367	14.67	25159
<i>herzjesu</i>	1186	342	41.38	32	1049	376	36.57	23
<i>kapel</i>	1355	312	13.9	48	773	153	9.94	24
<i>piedra</i>	1219	337	14.24	0	611	229	10.4	0
<i>reduced</i>	1291	284	59	9	1114	588	58.27	8
<i>ualberta</i>	1340	407	15.05	2666	312	67	6.08	314

Table 2 reports the results of the comparison. The data sets that appear in the table are described in detail in Chapter 6. In the table, we can see that AKAZE performs better than SURF in most of the data sets according to the total number of points extracted. However after filtering the results are more equilibrated and SURF results on more points correctly matched on *cantera*, *castle*, *fountain*, *fpv1*, *herzjesu* and *reduced* datasets. Our conclusion is that

both methods have similar performance according to the selected metrics. This is an important result for this work as one of the contributions of this thesis, the algorithm CCMM (chapter 3), relies heavily on good quality feature descriptors and considers multiple hypothesis for matches. SIFT and SURF are also good candidates, but since they are restricted by patents, they find limited use in practical applications. We have thus established that AKAZE is an open domain algorithm with similar performance than SURF and SIFT for the metrics that are of interesting to this work. Moreover, it allows our algorithm CCMM to be used in a broad range of applications in practice.

Descriptors provide a similarity metric that serves as matching criteria for pairs of observations in two images. This similarity metric, however, suffers from the differences between observations due to the effects of noise, light, scale etc. Moreover, the matching algorithm must also detect situations where a feature is occluded, or simply not present, in some of the frames. Another difficult problem is performing image associations in a structured environment where repeated patterns are present and features cannot be matched solely by their individual aspect. In all these situations, additional geometrical constraints can help to discard outliers (filtering) and may be used as heuristics to drive the search.

In this thesis we are differentiate between the concrete matching process and the filtering process. This will not only allow for a clearer exposition, but also help us to establish a better comparison between one of the contributions of this work and the current alternatives. Following this approach, we proceed to describe in the next subsection some common feature-matching techniques. State-of-the-art filtering techniques will be presented in subsection 2.4.

2.3 Feature-matching techniques

This section covers some common strategies used in state-of-the-art feature association between two images. Specifically, we consider the following:

- K-Nearest Neighbor (usually referred to by the acronym KNN)
- K-d tree

- Bag of words

We now proceed to describe each one of them in detail.

2.3.1 K-Nearest Neighbor (KNN)

One of the most common techniques to pair features is to select the nearest neighbor according to the descriptors (see Figure 9). Before describing the details of KNN in our application, we must first introduce the concept of *search* and *reference spaces*. They are named due to the influence of photogrammetry, where a reference photograph was first determined. This reference photograph was processed so that its position could be estimated in a so called global reference space. Whenever a fresh image needed registration, points from this *search image* were determined in the reference image. In conclusion, search proceeded by iterating over the features of the search image to see if a correct match could be established with those in the reference one. Note that this does not preserve symmetry, and some features in the reference image may never get be considered.

In the problem presented in this work none of our images are pre-processed first (to become reference images). However when matching a pair of images we do often need to assign reference and search labels arbitrarily. As explained later on in this section, reverting the labels can lead to different keypoint associations.

K-Nearest Neighbors first computes the distance from a descriptor to every other descriptor in the search space. The reference feature to be matched is then associated to the one nearest in the search image space.

Figure 9. An example of the k-nearest neighbor heuristic. Feature f is matched to the nearest feature f_2 .

There are several disadvantages to this approach. One problem is a potentially high computational cost: this can be a problem in big search spaces, when performed for all combinations of candidate and reference keypoints. A second important problem is that a feature may be linked to more than one reference feature. An interesting observation is that, if the search is performed inverting the search and reference spaces, the outcome of KNN can differ. This property can be used to filter outliers: matches that are different when the search is performed in both directions are removed from the matched list.

To end this discussion, we note that the KNN method requires a metric of similarity between the descriptors. The most common ones are L1, L2 and Hamming. L1 and L2 are used by SIFT, SURF and AKAZE. Hamming is used by ORB, BRIEF, BRISK and others. In the Hamming case, descriptors must be encoded as bitstrings.

2.3.2 K-d tree

To speed up the nearest neighbor search, an indexing technique called k-d tree may be used. A k-d tree is a space partitioning structure to index points in a k-dimensional space. In a k-d tree, each node represents a division of the search space in two halves and each division is done by a plane perpendicular to one of the axis. There are many possible ways to construct a k-d tree by choosing axis aligned to splitting planes. Typically, the axis chosen in each iteration is the

dimension for which the data exhibits the greatest variance. Interestingly, it has been proved [8] that this selection strategy can degenerate in some situations outputting non-optimal indexes. Figure 10 presents an example of k-d tree for a binary-nested container.

Figure 10. K-d tree example. On the left the feature space subdivided in binary nested container. On the right the corresponding tree structure to the left subdivision

[8] employed a modification of the k-d algorithm using randomized trees, called the *Best-bin-first* search method. Randomized trees are built by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance. This technique avoids index degeneration.

Another variation [9] is the hierarchical k-means tree. These type of trees are constructed by splitting the data points at each level into K distinct regions using a k -means clustering, and then applying the same method recursively to the points in each region. Recursion stops when the number of points in a region is smaller than K . The results of this algorithm are approximate due to the nature of the k -means tree. The technique, however, speeds up the search and returns a good candidate, although optimality is not guaranteed.

2.3.3 Bag of Words approximation (BoW)

When working with large image datasets, performing image matching between all the possible pairs of images may become impractical. Moreover, most of the combinations could well be non-overlapping pairs and attempting to match such pairs introduces a high number of incorrect matches. To avoid this problem, a method to evaluate the global aspect of the image and determine if

two images overlap is desirable. The Bag of Words (BoW), also called Bag of Features, technique is one such method.

Figure 11. Example of the Bag of Words (BoW) algorithm

Bag of Words is constructed as the following two-step process:

1. Select a vocabulary of features representative of the scenery.
2. Calculate a bag of words for each image as a histogram of the scene vocabulary (set of relevant features) that appear.

The first stage uses a subset of all the features found on the complete data set of images. The key idea is using an unsupervised classifier to generate a vocabulary of features with enough eloquence to describe our scenes. *K*-means is a good alternative to create this vocabulary. The result is a set of feature descriptors that will be used as categories to create our bags (see Figure 11).

The second stage creates a histogram for every image taking into account all its feature descriptors. Each features is classified into its nearest word from the vocabulary and added to the corresponding bin. The resulting histogram is thus made up of the number of descriptors that fit into each word from the vocabulary.

Another important issue is that BoW also requires a measurement of similarity between bags. In most disciplines, bags of words are employed to classify objects and machine learning classifiers are often used to assign a bag of words to a category. Typical examples of such algorithms are Bayesian classifiers and Support Vector Machines (SVM). In our specific application, however, a softer metric of similarity is sufficient. Specifically in this work, we chose the correlation metric c_{ij} between two images i and j defined as:

$$c_{ij} = \sum_k BoW_i[k] \cdot BoW_j[k]$$

where k refers to each of the bins of the Bag of Words and BoW_i , BoW_j , refer to the corresponding Bag of Words of each image.

The advantage of this metric over, for example, L1 or L2, becomes clear in the case of frames with partial overlapping. When this occurs, say, for example, a 50% overlap, the areas that do not overlap introduce a set of features in the histogram, which make them very different to the case when L1 or L2 metrics are employed as a measure of non-similarity. Note, however, that the similarity information between them is still recognizable by the correlation measurement c_{ij} described above.

2.4 Filtering techniques

As previously explained, model adjustment methods are usually very sensitive to non-Gaussian errors. The presence of outliers, even in a small proportion, may lead to invalid results on the adjustment. Matching techniques are very prone to introduce outliers, into the reconstruction problem. Such errors have a non-normal distribution and can lead to large distortions or even avoid convergence altogether. Consequently, filtering the initial matching results from outliers is a critical step in robust image reconstruction.

The state-of-the-art filtering technique is the RANSAC algorithm, which will be explained in detail in the following section. The algorithm falls into the category of parametric filtering, which means that it requires a prior model to adjust. This model can be as simple as a linear transform or as complex as a rational

polynomial projection. Literature also describes non-parametric filters, a good example of which is *vector field consensus* [10]. Vector field consensus creates a vector field representing the deformation between one frame and the other. It is similar to RANSAC in the sense that it uses a set of matches to estimate the field and then employs the field to check the consistency of the other set. Vector field consensus has demonstrated to be superior to RANSAC when a great number of outliers are present. However, in the presence of a common ratio of outliers it performs similarly to RANSAC and due to its high computational complexity has not become popular.

One of the problems of common matching techniques is that they usually cannot consider multiple candidate options from the reference image for a match. Feature descriptors are typically associated to their nearest neighbor in the reference image, and if the ensuing filtering stage rejects the match other neighbors are not considered. Interestingly, in [5] it is recommended only to accept a match if the second nearest neighbor has a feature descriptor distance that is at least 20% larger than the first. This rejects all the matches that are suspicious of ambiguity even if they are good matches. On the other hand, this strategy can be a problem in structured scenarios with a lot of repetitive patterns, such as bricks, windows, tiles and others.

A contribution of this thesis is a new filter technique compatible with RANSAC that allows considering a set of k best candidates for each feature. The importance of this new filter is better understood over structured scenes. Consider, for example, a building with regular patterns of windows or bricks, all of which have similar descriptors (i.e. two windows are almost identical). When finding an association for a window in another image, it is very plausible that the best candidate is matched to a different window instead of the correct one. By considering the best k matches it becomes possible to pair the correct window in the building.

Our new algorithm CCMM uses a graph model (denoted *association graph*) to encode the association information of two images including multi-hypothesis; finding a maximum clique in the graph provides a good compatible subset of

features while, at the same time, discarding outliers. As mentioned in subsection 2.2.3, CCMM is described in detail in Chapter 3.

We end this Chapter with an overview of the RANSAC algorithm and its application to image association.

2.4.1 RANSAC

Most model adjustment methods, such as least squares, don't have internal mechanisms for detecting and rejecting measurements with gross errors. In this work, the term *measurement* refers to a point detection in an image. It must be understood that measurements include a detection error that may be Gaussian, i.e. the point is located correctly on the image but its center has a little offset, or non-Gaussian, i.e. the point is simply incorrectly matched. In the following discussion we assume any measurement includes, not only the feature, but also the observation error.

The RANdom Sample Consensus paradigm [11] tries to estimate the model parameters with the minimum possible number of measurements n (see Listing 1). Initially, RANSAC will select randomly n measurements and estimate the parameters of the model. In a further step, the model is tested against the remaining measurements. If the error for a certain measurement is above a certain threshold T_ERR , related to the model, then the measurement is considered an outlier. If the model has a number of inliers, called consensus, above a certain threshold T_IN , then a new model is adjusted using all the inliers (step 4) and the procedure ends. If the cardinality of the consensus is lower than T_IN then a fresh set of n random points are selected and the process is repeated until an adequate consensus is found or a maximum number of iterations is reached. In the latter case, the model which had the greatest consensus is returned.

Listing 1. The RANSAC algorithm

RANSAC(*MODEL*, *measures*, T_IN , T_ERR , MAX_ITER)

Input: *MODEL*: mathematical model to adjust, *measures*: point detections,
 T_IN : desired inlier number, T_ERR : inlier maximum error

Output: updated parameters vector.

```

1. INLIER := {}
2. while count(INLIER) < T_IN OR iter < MAX_ITER
3.   m := selectn RANDOM data points from DATA_SET
4.   ADJUST (MODEL, params, m)
5.   for measure in measures
6.     err = |MODEL(params, data_point) – measure |
7.     if (err < T_ERR) then
8.       INLIER = INLIER ∪ {data_point}
9.     else
10.      OUTLIER = OUTLIER ∪ {data_point}
11.   next data_point
12.   iter := iter + 1
13. end while
14. return params

```

We consider two variants of this algorithm: one uses a homography transformation and the other a fundamental matrix. Note that affine transformations are a special case of homography transformations.

A Homography H is the matrix that represents the projective deformation that a plane can suffer, and may be expressed by the following equation:

$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where (x', y') and (x, y) are projections of the same point in two different images and s is an arbitrary scalar. Although the matrix H has 9 entries, in fact the homography has only 8 free parameters. This is explained by equations (7) and (8), which show two matrixes representing the same homography when all their terms are multiplied by a scalar. The key idea is that the scalar s is arbitrary.

$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = h_{33} * \begin{bmatrix} h_{11}/h_{33} & h_{12}/h_{33} & h_{13}/h_{33} \\ h_{21}/h_{33} & h_{22}/h_{33} & h_{23}/h_{33} \\ h_{31}/h_{33} & h_{32}/h_{33} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} (s/h_{33})x' \\ (s/h_{33})y' \\ (s/h_{33}) \end{bmatrix} = \begin{bmatrix} h'_{11} & h'_{12} & h'_{13} \\ h'_{21} & h'_{22} & h'_{23} \\ h'_{31} & h'_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (8)$$

Note that if all the features were selected from a plane in the 3D space, then the homography equations would hold for all the projections of the points.

When using a homography to filter the incorrect matches, we consider that a point is an inlier if the adjusted homography matrix predicts that a point from the search image is *near* the measured point in the reference image. The distance threshold is expressed in pixels.

Instead of using a homography for the RANSAC model, it is also possible to consider the epipolar relations between the two frames. The fundamental matrix F is the matrix that satisfies the epipolar geometry restriction shown on Figure 12

Figure 12. Epipolar geometry. A point (in red) projected on the left image (X_L) defines a line (Epipolar line) on the right image (X_R) created by the different possible depths of the tridimensional point projected on the right image.

Mathematically this restriction is expressed as equation (9)

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} F \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = 0 \quad (9)$$

where (x_1, y_1) and (x_2, y_2) are corresponding points on images 1 and 2 and F is the 3×3 fundamental matrix. F is estimated using a minimum of 8 points [12]. RANSAC uses the 8-point algorithm from [12] which avoids the ill conditioned nature of the basic equations by normalizing the coordinates of the points:

$$\hat{y} = Ty$$

$$\hat{y}' = T'y'$$

$$\mathbf{0} = (\hat{y}')^T ((T')^T)^{-1} F T^{-1} \hat{y} = (\hat{y}')^T \hat{F} \hat{y} \quad (10)$$

From (10), F can be obtained by reverting the initial transformation as follows:

$$F = (T')^T \hat{F} T$$

RANSAC considers a point to be an inlier if its projection in one of the images is nearer than a maximum distance threshold to the epipolar line determined by the point projection on the other image.

Figure 13. The error using the fundamental matrix is defined as the distance d from the measurement (in red) on the right frame (X_R) to the epipolar line determined by the point projection on the left frame (X_L).

It is important to note that using the fundamental matrix F as a model for RANSAC enforces a weaker constraint than using the homography H . This is because it allows the projected points to lie in a band along an the epipolar line that crosses the image, instead of a circle around a point as in the homography. However, using the fundamental matrix doesn't make any assumption about the

scene or the movement of the camera. A key insight is that using a homography as projective model implies that the perspective differences are small enough that the 3D points can be considered to be on the same plane. This assumption doesn't usually hold against real datasets.

But there is another greater problem related to homography or linear transformation models in RANSAC. When using a strict filter, it will reject all the points that do not comply with the assumptions, *even if they are correctly matched*. Therefore, the filter tends to select only the points that lie on a plane, rejecting all the good matches out of this plane. At first glance, this might be seen as a minor problem, but as we will explain in this work, the reconstruction equations tend to form an ill-conditioned system when all the points lie on the same plane. Therefore, a homography/based model for RANSAC during image registration is more likely to lead to an inconsistent reconstruction than when the fundamental matrix is applied.

Interestingly, homography and fundamental matrix models may be used simultaneously during 3D reconstruction. Homography is the most frequently used model to reject outliers, as it has a better rejection capabilities, so two RANSACs can be used during image registration: one RANSAC (RANSAC- F) uses F to build the 3D model and the other RANSAC (RANSAC- H) uses H to filter outliers.

One of the contributions of this work is a new filtering method CCMM that considers multiple hypothesis and replaces RANSAC as filter for outliers. As will be shown in the next chapter, our method performs better than RANSAC (Listing 1) both with F and H models. Moreover, after image registration using CCMM, it is also possible to use a RANSAC- F for the later steps of the 3D reconstruction.

3 CLIQUE MATCHING

One of the contributions of this thesis is a new algorithm CCMM that considers multiple hypotheses when matching visual feature descriptors with a clique model (also referred to as *clique matching*). Essentially, CCMM first builds a graph G that encodes association information between pairs of keypoints from two different images. This is done in such a way that a maximum clique in G defines an optimal association between the two images.

Determining if a hidden clique of a fixed size k exists is a well-known and deeply studied NP-complete problem referred to as k -clique [13]. Given a simple undirected graph $G = (V, E)$, a clique is any subgraph such that all its vertices are pairwise adjacent. The corresponding optimization problem is known as the *maximum clique problem* (MCP), which looks for the largest possible clique. The size of the solution to the MCP is known as the clique number of the graph $\omega(G)$. Besides its theoretical relevance as an NP-hard problem, MCP is known to have many applications in different fields of science, such as bioinformatics and computational biology [14], computer vision [15], robotics [16] [17], stock-market analysis, general network analysis and others.

Exact maximum clique has been studied from many perspectives, but the most successful family of current algorithms are branch and bound solvers which use vertex coloring heuristics as upper bound [18] [19] [20] [21] [22] [23] [24] [25]

[26]. In the comparative study by Prosser [27], MCS [19] and bit-parallel BBMC [21] [22] [26] stand out as leading solvers. In particular, Prosser indicates a lower bound rate of improvement of $2x$ in favor of BBMC in a number of graphs of middle and high densities from public benchmarks. Noteworthy is a recent pruning strategy based on a MaxSAT encoding of colored subgraphs of the original problem graph [24] [23]. Also noteworthy are the recent infra-chromatic bounds described in BBMCX [28] and BBMCXR [29].

Literature also records strong heuristic algorithms for MCP, such as iterated local search ILS [30] or adaptive multi-start tabu search AMTS [31]. Also in [25] Batsyn et al. report good results when feeding an exact solver with an initial solution produced by ILS.

Figure 14. Image processing chain for mosaicing a 3D reconstruction

We describe in this Chapter the new algorithm CCMM (acronym for image Clique Matching) for feature association between two images, a critical step in automated image processing (see Figure 14, association rejection). As mentioned, the algorithm first builds a graph model to capture pairwise multi-hypothesis information for each keypoint to be matched in the original image. Robust joint correspondence between two images is achieved in a second step by searching for a maximum clique in the graph with an efficient state-of-the-art exact procedure.

Moreover, we show in this work that worst case intractability for maximum clique is not a problem in practice for reasonable image sizes, partly because the order of the graph is linear with the number of interest points and the number of (multi) hypotheses considered. Reported results also show that CCMM is able

to track a keypoint along more frames better than previous RANSAC-based alternatives, something very useful in the later steps of the image processing chain (see Figure 14).

3.1 The clique model for image association

One of the most general forms of data association model is known as relational structure association. A relational structure is a 3-tuple $S = \{U, P, R\}$ in which U denotes the elements (units) of the structure, P is a set of properties defined for these elements and $R \subseteq U \times U$ a set of relations between elements, function of P , which are assumed binary. Let S' and S'' be two such structures; two elements $u' \in S', u'' \in S''$ are considered feasible if all properties that hold for u' also hold for u'' . Furthermore, two pairs of units $(u_1', u_1''), (u_2', u_2'')$, each pair comprising one element from each structure, are said to be compatible if all relations that hold for (u_1', u_2') also hold for (u_1'', u_2'') . The relational structure association problem consists in finding a pairing of units from S' and S'' such that feasibility and compatibility are maximized. When binary relations are quantified, as is usually the case in real applications, relational structures may be represented as graphs with weighted edges and the association problem is equivalent to the maximum common (weighted) subgraph problem (MCS).

Table 3. An example of the maximum common subgraph problem as maximum clique

A) A weighted graph $G_1 = (V_1, E_1)$

B) A weighted graph $G_2 = (V_2, E_2)$

C) Association graph $G = (V_1 \times V_2, E)$ considering a 20% similarity threshold between edges (i.e. $15 = 13, 13 \neq 10$)

Label notation is (v/w) , $v \in V_1, w \in V_2$

Colored edges refer to a maximum clique in G which solves joint compatibility $\{(2,1), (3,2), (4,3)\}$.

The dotted edges are incorrect associations that fall outside the compatibility threshold.

A clique model to solve this problem requires an *association or correspondence graph*. Given two relational structures (S', S'') , its association graph $G = (V, E)$ is a simple undirected graph such that nodes refer to pairs of feasible elements, one of each structure, and there is an edge between two nodes if the corresponding pairs of elements are also compatible. More formally:

$$V = \{(u', u'') \subseteq U' \times U'' \mid u' \text{ and } u'' \text{ are feasible}\}$$

$$E = \{(u'_1, u''_1), (u'_2, u''_2) \subseteq V \times V \mid (u'_1, u'_2) \text{ and } (u''_1, u''_2) \text{ are compatible}\}$$

The key idea behind the association graph G is that optimal joint compatibility is reduced to finding a maximum clique in G . Note that any vertex in G represents a pair of feasible elements and that all vertices in a clique are pairwise compatible. Table 3 presents an example of association graph for two relational structures G_1 and G_2 with weighted edges. In the example, two edges are considered compatible if the difference between weights is below a 20% threshold (i.e. 13 is compatible with 15, but not with 10). Examples of clique models for association are frequent in different branches of science, as in [14] [15] [16] [17].

In the case of image registration, a structure is an image, units are points of interest (pixels), properties of units are vector descriptors computed by SIFT,

SURF or AKAZE algorithms and relations between units are Euclidean distances between the corresponding pixel pairs.

3.2 An exact algorithm for the maximum clique problem

This section outlines the reference exact algorithm for MCP, denoted REF_CLIQUE, used by CCMM. We start by introducing some definitions and notation. A simple undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$, consists of a finite set of vertices $V = \{v_1, v_2, \dots, v_n\}$ and edges $E \subseteq V \times V$ that pair distinct vertices.

Notation used in this section concerning graphs includes the following:

- $N(v)$: The neighbor set of a vertex v in a given graph.
- $G[U]$: The subgraph induced by a set of vertices $U \subseteq V$ in G .
- $C(G) = \{C_1, C_2, \dots, C_k\}$: Any feasible (vertex) coloring of G with k colors (also *size* or *cardinality* of the coloring). C_i denotes the i -th independent subset (also i -th *color class*) that contains all the vertices assigned the i -th color number.
- $c(v)$: the color number assigned to v by $C(G)$, i.e. $v \in C_i \Leftrightarrow c(v) = i$.
- $\chi(G)$: The chromatic number of the graph, the minimum size of any feasible coloring.

The following well-known propositions relate approximate vertex coloring to exact maximum clique branch-and-bound search:

Proposition 1 [Balas and Yu, 1986] *The chromatic number of an arbitrary graph is never lower than its clique number: $\chi(G) \geq \omega(G)$.*

Proposition 2. *Let k be the size of any feasible coloring of G ; then $k \geq \omega(G)$.*

REF_CLIQUE is essentially BBMC [21] (with the improvements described in [22]), a branch and bound bit-parallel algorithm which uses approximate vertex coloring to prune the search tree. It encodes the adjacency matrix of the input graph as rows of bit vectors and uses bit masks to compute child nodes and color bounds efficiently. We refer the reader to [21] for all implementation details not included in this overview. We note that the current outline has been

tailored for clique matching and includes the heuristic computation of a very large initial clique which is not present in [21].

Before continuing with an explanation of REF_CLIQUE, we include additional notation concerning variables and symbols that appear in the listings of algorithms in this section:

- U : The list of vertices which make part of the current subproblem (node).
- U_v : The list of vertices of the new child subproblem induced by vertex v .
- S : The set of vertices of the clique being enlarged in the current branch and also the path from the root to the current node.
- S_{max} : The set of vertices of the incumbent clique, that is, the largest clique found at any moment during search.
- S_0 : An initial large clique found by a heuristic at the beginning of the search.

Listing 2, Listing 3 and Listing 4 describe REF_CLIQUE. The actual NP-hard step is outlined in Listing 3 and Listing 4, while Listing 2 is a pre-processing step. For a clearer exposition we will skip the order of the calls and first explain the search routines. A brief comment on pre-processing will follow afterwards.

Listing 2. The reference maximum clique procedure

```

REF_CLIQUE ()
1.  INIT_SORT(V)
2.  C ← INIT_COLOR(V)
3.   $S_0 \leftarrow AMTS(V, E)$  //heuristic algorithm seed
4.   $S \leftarrow \phi$ 
5.   $S_{max} \leftarrow \text{FIND\_MAX\_CLIQUE}(V, C, S, S_0)$ 
6.  return  $S_{max}$ 
    
```

Listing 3. The branch and bound algorithm

```

FIND_MAX_CLIQUE (U, C, S,  $S_{max}$ )
1.  repeat until  $U = \phi$ 
2.      select a vertex  $v \in U$  with maximum color number in C
3.      if ( $|S| + c(v) \leq |S_{max}|$ ) then return
    
```

```

4.       $s \leftarrow s \cup \{v\}$ 
5.       $U_v \leftarrow U \cap N_u(v)$ 
6.      if ( $U_v \neq \emptyset$ ) then
7.           $C_v \leftarrow \text{REF\_COLOR}(G[U_v])$ 
8.           $\text{FIND\_MAX\_CLIQUE}(U_v, C_v, S, S_{max})$ 
9.      elseif ( $|S| > |S_{max}|$ ) then  $S_{max} \leftarrow S$ 
10.     endif
11.      $S \leftarrow S / \{v\}$ 
12.      $U \leftarrow U / \{v\}$ 
13. endrepeat
14. return  $S_{max}$ 

```

FIND_MAX_CLIQUE (Listing 3) describes a branch and bound algorithm, implemented by recursion so that a fresh step in the search is a new call to the algorithm. U is the set of candidate vertices that can improve the current clique in S and is initialized to V at the root node. A new node in the search tree is formed by enlarging the clique S with any candidate vertex from U . Full enumeration (possibly with repetitions) is achieved by enlarging the clique in S until it is maximal and then backtracking to the previous node to continue with another candidate in U . All maximal cliques are compared with the incumbent solution in S_{max} (step 9), which is updated when required a bigger clique has been found. It is interesting to note that the following two invariants hold during tree traversal: *invariant 1*- $S \cap U = \emptyset$, *invariant 2*-every vertex in U is pairwise adjacent to all vertices in S , $\forall S, U$.

A more efficient strategy is to prune the search when the current growing clique S cannot possibly improve the clique S_{max} . For each new subproblem, a minimum improvement threshold k_{min} [5] is $|S_{max}| - |S|$. As explained previously, the majority of efficient recent maximum clique exact solvers use approximate (vertex) coloring to establish an upper bound on $\omega(G)$, according to proposition 2.

In FIND_MAX_CLIQUE the coloring is computed by a call to procedure REF_COLOR (step 7), which outputs a feasible coloring the new child subproblem $G[U]$. Let $c(v)$ refer to the color number assigned to vertex $v \in G[U]$

by REF_COLOR. Then, according to the definition of k_{min} , the subproblem hanging from v can be pruned when $c(v) \leq k_{min}$ (step 3 of FIND_MAX_CLIQUE). Also in modern solvers, coloring does not only establish a bound, but also serves as a pivoting heuristic. Specifically, FIND_MAX_CLIQUE branches on candidates with *maximum color first* (step 2). This idea was first described in algorithm MCQ by Tomita and Seki [3] and has since then been taken up by the vast majority of exact MCP approximate coloring algorithms.

Listing 4. The sequential independent set coloring procedure

```

REF_COLOR ( $G[U_v] = (V', E')$ )

Input: An induced subgraph  $G[U_v]$ 
Output: Greedy independent set coloring  $C(G[U_v])$ 

i.1.  $C \leftarrow \phi, U \leftarrow V$ 

1.  repeat until  $U = \phi$ 
2.       $C_i \leftarrow U$ 
3.      repeat until  $C_i = \phi$                                 //computes color set i
4.          pick the first vertex  $v$  from  $C_i$  not previously selected
5.           $C_i \leftarrow C_i / N(v)$ 
6.           $U \leftarrow U / \{v\}$ 
7.      endrepeat
8.       $C \leftarrow C \cup C_i$ 
9.       $i \leftarrow i + 1$ 
10. endrepeat
11. return  $C$           //returns the new vertex coloring
    
```

REF_COLOR is outlined in Listing 4. It is a sequential independent set coloring heuristic where color sets are obtained in turn (steps 3-7), as opposed to typical sequential coloring where vertices are colored in the order they are presented to the algorithm. One advantage is that vertices can be sorted efficiently by non-decreasing color (step 4 of REF_COLOR), the pivoting strategy required by FIND_MAX_CLIQUE. The disadvantage is that it operates with the full set of unlabeled vertices at the start of each new color iteration. Reported results in literature for bit-parallel algorithm BBMC and its variants has shown that this overhead may be reduced by an appropriate bit encoding of the domain. Again

for additional implementation details on the bit-parallel framework we refer the reader to [21] [22].

PRE-PROCESSING (Listing 2)

The algorithm starts with an initial sorting of vertices of the input graph $G = (V, E)$ (INIT_SORT in Listing 2, step 1), which is well known to have an impact in reducing the search tree. REF_CLIQUE uses the ordering first described in [32] in the context of vertex coloring. The vertices are ordered (and selected at the root node) by non-increasing degree (i.e. smallest-degree first) which is well known to reduce the size of the search tree in exact procedures. However instead of using an absolute ordering based on degree, a degenerate one is used (i.e. at each step of the sorting procedure the vertex with smallest degree is taken and removed from the original set. Degrees are then recomputed for the remaining vertices before the next sorting step).

For practical reasons concerning data structures, vertices are actually placed in reverse order (i.e degenerate smallest-degree-last) and picked at the root node from last to first. More formally, vertices $V = \{v_1, v_2, \dots, v_n\}$ are placed initially such that $\forall i = 1, \dots, n$, the degree of v_i in the induced graph $G[\{v_1, \dots, v_{i-1}\}]$ is minimal.

A simple initial color label $C(G)$ is also provided for each vertex (INIT_COLOR in Listing 2, step 2) such that $\forall i = 1, \dots, n$, $c(v_i) = \min\{i, \Delta G + 1\}$. Note that $C(G)$ will be unfeasible in the general case for vertices with labels $\Delta G + 1$, but is still valid for the problem since $\Delta G + 1$ is an upper bound for the clique number of the graph.

Once the vertices are sorted and labeled, global variables S and S_{max} are then assigned initial values (Listing 2, steps 4 and 5). REF_CLIQUE uses an initial clique S_0 as seed to the main procedure, which is computed by state-of-the-art adaptative multistart_tabu search heuristic algorithm AMTS [31] (any other strong approximate algorithm such as ILS [30], would be perfectly valid as well). AMTS in particular includes features such as constrained neighborhood, a tabu

mechanism and a long term memory-based restart strategy as explained by the authors. This approach of feeding the exact solver an initial strong solution does not make part of the original BBMC, but has recently proven particularly effective in the general case as mentioned in the introductory section. The framework described here computes an initial “good” solution by default. When the search procedure is called S_{max} is assigned S_0 .

THE SPECIFIC APPLICATION

An association graph resulting from image feature association tends to be large and relatively sparse. Its sparse nature comes from the fact that edges represent Euclidean distance compatibility between pixels, which is a hard local constraint. The majority of optimizations described recently for the general purpose MCP exact solvers are connected with producing tighter bounds. Noteworthy ideas are *recoloring*, first described in MCS [19], relaxed coloring [26], computation of bounds based on a reduction to partial maximum satisfiability MaxCLQ and IncMaxCLQ [23] [24] and the so called *infra-chromatic* bounds [28] [29]. All these techniques work well over middle and high density graphs but are not specially suited for the less dense graphs, where the impact of local pruning is much lower. In general, the sparser the problem graph the simpler the algorithm should be. This is the reason why we have chosen as implementation for REF_CLIQUE BBMCI [33] and not BBMCR [33], BBMCL [26], incMaxCLQ [23], BBMCX [29] or BBMCRX [34]

Specifically, we note that solvers which use SAT-based bounds, and also infra-chromatic solvers, are known to perform well in the denser graphs, because they capture structure better. In this particular problem, however, the densities of the association graph are never higher than 0.1 (see Table 5). Tests with graphs generated from the CCMM using incMaxCLQ confirms this idea (see Table 10). MCS and BBMCR are similar to BBMCI but include a recoloring procedure which is fruitful mainly in the denser graphs as explained in [22]. Finally, tests with the selective coloring scheme proposed in BBMCL [26] proved no advantage over the reference solver (see Table 11).

In section 3.5 a new encoding tailored for semi-sparse graphs will be introduced. This graph encoding, named watched graphs, can be used with the reference algorithm BBMC1 to create a new exact clique search algorithm (BBMCW). This new algorithm is targeted to efficiently solve the graphs result of the CCMM algorithms. Results of the performance of this new technique will be described on section 3.6.3.

It is common knowledge that seeding a branch and bound exact algorithm with a tight initial lower bound is good practice. However literature records only recently this idea in relation with finding an exact maximum clique. In [25] heuristic ILS is employed to feed an exact solver with an initial good solution under bounded time. The new algorithm outperforms the original one in a number of instances taken from public benchmarks (see results on section 3.6). Our default REF_CLIQUE employs this idea (note that ILS has been substituted by AMTS).

3.3 The new CCMM algorithm

This section describes CCMM (acronym for Clique Matching), a new multi-hypothesis algorithm for the image registration step in automated image processing. In contrast with state-of-the-art RANSAC, CCMM is much more robust since it considers the first k best candidates in the reference (also target) image for each interest point in the search (also original) image. It does so by building an association graph G and then looking for a largest clique in the graph.

The new procedure is applicable between image pairs subject to the following properties:

- *Uniqueness*: For each keypoint in the original image there is only one correct point in the target image (or none at all if the images do not overlap).
- *Low deformation*: The relative distance (in pixels) between two correctly matched points in the original image should be similar (but not necessarily the same) to the distance between the corresponding pair in the target image.

The low deformation requirement holds (depending on the notion of similarity) for consecutive (or almost consecutive) shots taken by a moving camera. In practice, this occurs for example in aerial imagery (i.e. almost zenithal shots taken at a constant height) or in visual odometry for robot navigation. Moreover, it is also valid in applications in which the optical or projective deformation is not very high.

Another setback of general clique models in real life applications is that the graph order is quadratic in the number of features if no a priori feasibility relation between single features exists. In our specific application, feasibility is provided by SURF/SIFT algorithms in the form of vector descriptors for each keypoint. Proposition 3 provides the main result for the applicability of clique models in practice for this problem. We start with a preliminary definition to present the result.

Definition: A model for keypoint association is k -hypotheses if for each point in the original image a set of (at most) k keypoints are considered in the target image according to a given feasibility relation.

Proposition 3. Consider a keypoint association problem with n points in the original image and an indeterminate number of points in the target image. Then, the order of the association graph in the clique k -hypotheses model is at most $n \times k$.

Intuitively since at most k points in the target image are considered for each point to match in the original image, only these combinations appear as vertices in the association graph. Note that this result is independent of the actual number of keypoints in the target image.

CCMM appears in Listing 5. It receives as input two images I_A (original) and I_B (target) as well parameters K , the number of hypotheses encoded by the clique model, and T , a threshold which captures similarity between pairs of points in both images as a percentage of the distance between the pair in the original image. Other symbols used in the listing are:

- $KP_X \in \{I_A, I_B\}$: The set of interest points for image X .

- p_a, p_b : A keypoint from image I_A or I_B respectively
- d_a, d_b : An Euclidean distance between two points (pixels) from image I_A or I_B respectively.

Listing 5. Outline of the new algorithm

```

CCMM ( $I_A, I_B, K, T$ )
Input: Two images  $I_A$  (original) and  $I_B$ (target),  $K$ :number of hypotheses,
      T: distance threshold
Output: A consistent set of keypoints (as pixels) between the two images.

1. Determine SURF keypoints from  $I_A$  ( $KP_A$ ),  $|KP_A| = N$ 
2. Determine SURF keypoints from  $I_B$  ( $KP_B$ )
3. Extract SURF descriptors from  $KP_A$  and  $KP_B$ 
4. Select  $K$  best matches in  $KP_B$  for each point in  $KP_A$ 
6. Allocate space for a empty graph  $G(V, E)$  with  $|V| = n \leftarrow N \times K$ 
7. for  $i = 1$  to  $n-1$ 
8.     for  $j = i$  to  $n$ 
9.          $d_a \leftarrow |p_{a_i} - p_{a_j}|, p_{a_i}, p_{a_j} \in KP_A$ 
10.         $d_b \leftarrow |p_{b_i} - p_{b_j}|, p_{b_i}, p_{b_j} \in KP_B$ 
11.        if ( $d_a \neq 0$  and  $d_b \neq 0$ ) then
12.            if ( $|d_a - d_b| < T \times d_a$ ) then
13.                add edge  $(i, j)$  to  $G$ 
14.            endif
15.        endif
16.    next  $j$ 
17. next  $i$ 
18.  $r \leftarrow \text{REF\_CLIQUE}(G)$ 
19. return  $r$ 
    
```

The general outline of CCMM has the following steps:

1. Computation of keypoints and descriptors for each image using SURF.
2. Computation of the best K matches in the target image for each interest point in the original image (the vertices of the association graph).
3. Building an association graph G .
4. Finding a maximum clique in G .

First SURF keypoint sets KP_A and KP_B and descriptors are computed for each image (lines 1-3). Then, for each interest point $p_a \in KP_A$ in the original image, the algorithm selects the K closest interest points in the target image according to

the descriptors (line 4). In this work K has been set to 3 (i.e. only the best three options have been considered in the clique model).

One of the most expensive operations when building the association graph is to compute distances (measured in pixel units) between pairs of points. Attempts at caching relevant distances have not improved performance in our tests, so the current implementation calculates the distance between point pairs on demand during graph generation.

Each pair $(p_a \in KP_A, p_b \in KP_B)$ of similar points according to the SURF descriptors becomes a vertex in the association graph. For $N = |KP_A|$ keypoints in the original image, and the chosen default value $K=3$, the order of the association graph is $3 \times N$. Descriptors enable the clique model to grow linearly (according to proposition 3), which makes CCMM practical even for big high resolution images with a dense cloud of points.

The complexity of graph generation is $O(N^2)$. The complexity of robust joint compatibility using CCMM is exponential, since we use an exact algorithm for the maximum clique problem. In spite of this, the experiment section reveals that the polynomial order of the association graph ($K=3$) keeps complexity within reasonable bounds in practice.

The association graph is built with edges representing valid associations between the pairs of interest points referred by its vertices (i.e. there is an edge between (p_{a1}, p_{b1}) and (p_{a2}, p_{b2}) if $(p_{a1}, p_{a2}) \approx (p_{b1}, p_{b2})$ (steps 7-17). Let $v_1 = (p_{a1}, p_{b1})$ and $v_2 = (p_{a2}, p_{b2})$ be any two vertices in the association graph $G = (V, E)$. Then there is an edge (v_1, v_2) in E if the following two conditions hold:

- I. The same keypoint is not associated with more than one point in the other image, i.e. $p_{a1} \neq p_{a2}$ and $p_{b1} \neq p_{b2}$ (step 11).
- II. The distance in pixel units between points is below a certain threshold, i.e. $|p_{a1} - p_{a2}| \approx |p_{b1} - p_{b2}|$ (step 12).

Condition I avoids a point being associated with more than one point in the target image. Condition II captures geometric constraints.

Regarding the threshold of similarity T , we initially chose an absolute value but the resulting clique models did not capture locality appropriately and the corresponding MCPs were hard to solve. Much better results are obtained by using a relative measure for T , i.e. a percentage of the size of $|p_{a1} - p_{a2}|$. In this way, the similarity test is harder to pass for pairs of points close in the original image compared with those that are more separated.

A possible explanation for this is that, in practice, distortions between consecutive images also scale with pixel distance, i.e. bigger objects are very often more distorted than smaller ones. Absolute thresholds tend to produce clique models with many connected components of similar size and it takes longer for the exact solver to escape from these local minima. In all the experiments reported in this paper, T has been set to 0.1. This threshold has been selected empirically to reduce the reprojection error while keeping the amount of information encoded in the correspondence graph within reasonable bounds.

Figure 15 depicts an example of how the association graph is built. Symbols $A\langle N \rangle$ and $B\langle M \rangle$ refer to keypoints N and M in the original and target images respectively. The top left picture shows vertices in the clique model as pairs of feasible points $(A\langle N \rangle, B\langle N \rangle)$ according to descriptors. The red edges in the top right image are invalid associations according to condition I and will not make part of the association graph. Note that, in all cases, the two vertices connected by the edge share a common point. At the bottom left corner the label underneath the dotted edge is a geometric constraint (condition II). It will make part of the clique model if distances d_1 and d_2 are comparable according to T . Finally the bottom right image depicts the association graph. Maximum clique A_1B_1, A_2B_2, A_3B_3 is the solution to the problem.

Figure 15. An example of the new clique association model.

3.4 Metrics and configuration

We have compared the new algorithm CCMM with the two RANSAC variants commonly used for this problem: RANSAC-H, which uses a homography model, and RANSAC-F which uses a fundamental matrix F model (see Chapter 2). We have employed OpenCV implementations both for RANSAC and F estimation. For descriptors we have employed the SURF OpenCV implementation with adaptive parameters to obtain a number of features (keypoints) between 100

and 400. The implementation varies the SURF threshold between 500 and 4000 with a starting value of 1600.

A qualitative (and fair) comparison between CCMM (which captures uncertainty in the graph model and makes exhaustive search) and a non-deterministic algorithm such as RANSAC is not an easy task. Moreover, it has not been possible to find a standard procedure in literature. We have, therefore, devised a number of specific metrics for this problem which can be computed over any data set. All of them are also useful indicators for later steps in the automated visual processing chain such as 3D reconstruction, visual odometry or mosaicing. We are also aware that ground truth comparisons for feature association techniques are usually performed over synthetic images. However, in this work we are trying to compare outlier rejection techniques, and synthetic images are often not representative for outliers.

The metrics chosen are the following (see also Table 4):

- Total number of points in frames
- Mean number of frames per point
- Mean number of more than two frames per point
- Number of unique points
- Number of unique points in more than two frames
- Reprojection error

The reprojection error is commonly used in visual matching problems. It is the mean root minimum square (RMS) error over the reprojection computed by the fundamental matrix F , estimated with correctly matched points between image pairs.

To understand the remaining metrics, we will start by describing the main data structures employed and the algorithm used to extract the data from our data sets. We will then proceed with the mathematical formulation for each one. We use a linked list of matched physical points, each point containing a list of frame objects and each frame object containing the two coordinates of the corresponding pixel and the frame identifier.

For a point to be added to the list, it must be found in at least two frames. Listing 6 outlines the algorithm POINT_TRACK which processes pairs of keypoints. It receives as input the current list of physical points P and two correctly matched keypoints (in 2D pixel coordinates) together with the corresponding frame identifiers. On output, it updates the list of physical points conveniently.

Each time POINT_TRACK is called, it checks for an identical image point already in P (line 3) by comparing the input keypoint pixel coordinates and frame numbers against all previously stored points (lines 3, 7). If the pixel coordinates and frame ids are the same, this means that the physical point p_i which maps to the keypoint is already in P , and p_i is added the missing frame (lines 4, 8). However, if the point cannot be found, it is added to P as a new physical point together with the information of both input frames (line 13).

POINT_TRACK is called for every scene point correctly matched in every pair of images in each data set. The procedure outputs the updated set of points P , each point $p_i \in P$ containing a set of frames F_i . Once all the calls to POINT_TRACK are done, P is evaluated according to the metrics listed previously, which are formally defined in Table 4.

Listing 6. Outline of the algorithm which updates a list of physical points P given a pair of keypoints correctly matched in two frames. Notation $p_i.f_j$ refers to the j -th frame of the i -th point in P .

```

POINT_TRACK ( $P, k_1, k_2, I1, I2$ )
Input: A list of scene points  $P$ , keypoints  $k_1$  y  $k_2$  correctly matched in frames
      I1 and I2 respectively.
Output: Updated list of scene points

1. for  $i = 1$  to  $|P|$ 
2.   for  $j = 1$  to the number of frames in  $p_i \in P$ 
3.     if ((  $k_1.x = p_i.f_j.x$  ) and(  $k_1.y = p_i.f_j.y$  ) and
         (  $I1.id = p_i.f_j.id$  ) then
4.        $p_i.f_j.add(new\ frame(k_2.x, k_2.pt.y, I2));$ 
5.       return
6.     end if
7.     if ((  $k_2.x = p_i.f_j.x$  ) and(  $k_2.y = p_i.f_j.y$  ) and
         (  $I2.id = p_i.f_j.id$  ) then

```

```

8.           $p_i.f_j.add(\text{new frame}(k_1.x, k_1.y, I_1));$ 
9.          return
10         end if
11.       next j
12. next i
13.  $P.add(\text{new point}(\text{new frame}(k_1.x, k_1.y, I_1), \text{new frame}(k_2.x, k_2.y, I_2));$ 
14. return

```

Configuration parameters for CCMM are the similarity threshold $T=0.1$ and the number of candidates $K=3$. The implementation used for the search routine in REF_CLIQUE (Listing 2) is the same as in BBMCI [33]. We consider variants with and without approximate clique algorithm AMTS to compute an initial clique. In the latter case, step 3 in Listing 2 is substituted by a simple initial clique computation described in [11]. We use default configuration parameters for AMTS as provided by the authors in [15].

The tests have been performed on a 20 core Intel Xeon with 64Gb RAM. The operative system used is Linux Ubuntu Server 12.04. The binaries have been compiled with the GNU C Compiler tool-chain.

Table 4 Metrics used for the tests

Metric	definition
<i>total number of points in frames</i> Sum of matched points over all the frames.	$\sum_{p_i \in P} F_i $
<i>mean number of frames per point</i> Average number of frames in which it has been possible to track a point.	$\frac{\sum_{p_i \in P} F_i }{P}$
<i>mean number of more than 2 frames per point</i> Average number of three or more frames in which it has been possible to track a point.	$\frac{\sum_{p_i \in P} F_i }{ P }, \forall p_i \in P \vee F_i > 2$
<i>number of unique points</i> Number of points in the real world derived from the matching. For example, a keypoint tracked along three frames computes only as one.	$ P $

<i>number of unique points in >2 frames</i> Number of unique points that have been detected in more than two frames.	$\sum_i P_i, \forall p_i \in P \vee F_i > 2$
--	--

3.5 Watching Subgraphs

Graphs generated by the CCMM algorithm show a common structure and density within a certain range. This allows us to consider tailoring REF_CLIQUE (described in section 3.2) in a way that best suits the problem graphs. This section describes the on such tailoring, the new algorithm BBMCW also contribution of this work.

As mentioned previously, REF_CLIQUE is well suited for small and middle sized graphs. However, in a recent work [35] it is shown that it does not scale well in the case of massive real graphs. Amongst other things, storing the full adjacency matrix in memory for such case is impractical, and some form of bit-compression is required to allow sparse information to be fitted into a more compact representation. The algorithm BBMCSP [35] tailors REF_CLIQUE for massive graphs. A key idea is a novel sparse bitstring encoding, which does not allocate memory for non-adjacent nodes but still allows for efficient bit-masking.

In our tests, however, graphs derived from CCMM range typically between 100 and 5000 nodes with a density between 0.2 and 0.5. Such graphs do not fall under the sparse category, but even so, some of the biggest instances cannot be solved efficiently by REF_CLIQUE. Note that, in contrast with the assumptions made in BBMCSP, these graphs can be encoded explicitly in memory without a heavy penalty in performance. There are, however, still a significant number of spurious empty bitblocks in the encoding. In this type of scenario, *efficiency depends very much on the number and management of empty bitblocks that are processed during search.*

Figure 16. Example for demonstration purposes. To the left a simple undirected graph. To the right, the bitstring encoding of its vertex set.

To help understand the problem we will use the sparse bipartite graph G depicted in Figure 16, a. Figure 16, b, shows the bitset encoding of its vertex set V , which may also be viewed as the root node subproblem $U = V$ of REF_CLIQUE. As can be seen, two of its four B_{64} bitblocks, B_U^2 and B_U^3 , are empty. The existence of these empty bitblocks quickly degrades the benefits of bitmask operations in REF_CLIQUE. Here are some examples which justify this assertion:

- *Enumeration of child subproblems* (Listing 3 at step 5 of REF_CLIQUE): REF_CLIQUE first branches on vertex $\{255\}$ which has minimum degree. Set intersection $B_{U_{255}} = B_U \cap B_{N(255)}$ requires four bitmask operations, whereas actually $B_{N(255)}$ contains just the single non-empty bitblock $B_{N(255)}^1$ (corresponding to vertex $\{2\}$). Therefore, only the bitmask concerning the first block of both bitsets is actually meaningful.
- *Empty set checking* (as in Listing 3 at step 6 of REF_CLIQUE): We now consider a further branching on vertex $\{2\}$, $B_{U_2} = B_{U_{255}} \cap B_{N(2)}$. To check that B_{U_2} is empty, REF_CLIQUE requires examining its four bitblocks, but $B_{U_{255}}$ contained only the single non-empty bitblock $B_{U_{255}}^1$, so it is pointless to check the other three.

Motivated by the above considerations we present a novel encoding which introduces two auxiliary pointers to reduce the impact of empty bitblocks in REF_CLIQUE. This new encoding makes use of these two pointers to skip bit-

masking operations on a subset of (empty) bitblocks. These pointers are denoted *sentinels* in the new bitstring encoding.

3.5.1 The *sentinel* encoding

To formalize the new bitstring encoding, we introduce the notion of *sentinel*. A *sentinel* is a subset $U \subseteq [n]$, that refers to one of the *non-empty* blocks of bitset U . We denote the specific bitblock by B_U^s . Normal values for sentinels are block indexes, i.e. $\{1, 2, 3, \dots, 1 + \lfloor n/w \rfloor\}$, but, in practice, we also consider negative values -1 or -2 to denote the empty set. Special mention deserve sentinels l and h , which point to the first (lowest) and last (highest) non-empty blocks of a bitstring respectively. More formally:

$$l = \min\{i | B_{n,U}^i \neq 0\} \text{ or } -1 \text{ if } B_{n,U} = 0$$

$$h = \max\{i | B_{n,U}^i \neq 0\} \text{ or } -2 \text{ if } B_{n,U} = 0$$

Furthermore, we say a bitstring is *watched*, and denote it by BW_n , if it contains the additional sentinels (l, h) . Thus, in a watched bitstring, BW_n^l always refers to first (lowest) non empty block, and BW_n^h to the last (highest) non empty block. Figure 17 illustrates these concepts.

Figure 17. Watched Bitblocks

The term *watched*, as used in this work, is inspired by the notion of a *watched literal* employed in efficient SAT solvers. There, an unsatisfied clause permanently *watches* two of its open literals to detect when it becomes unit. In the case of maximum clique, watched subsets BW_U^l, BW_U^h of an induced subproblem $G[U]$, may also be seen as induced subgraphs $G[BW_U^l], G[BW_U^h]$ themselves, and their order is limited to the size of the register word W . For this reason, we denote the proposed encoding a *watched subgraph encoding*.

Listing 7 outlines the basic functionality to be expected in watched bitstrings: procedures INIT_WATCH, UPDATE and IS_EMPTY. INIT_WATCH initializes both sentinels and UPDATE reassigns them in case the current watched subgraphs have become empty. Note that UPDATE is conveniently non symmetrical for maximum efficiency: steps 2 to 8 of UPDATE set the lower sentinel to a non-empty bitblock or the empty state, whereas the upper sentinel is only updated if the lower one is in a non-empty state. UPDATE_LOW is a useful specialization of UPDATE for just the lower sentinel.

Listing 7. Basic procedures INIT_WATCH, UPDATE, UPDATE_LOW and IS_EMPTY to operate with watched sets.

INIT_WATCH(BW_n)

1. s_l : **if** ($BW_n \neq \phi$) **then**
2. $l \leftarrow$ point to the lowest non-empty biblock
3. **else** $l \leftarrow -1$
4. s_h : **if** ($BW_n \neq \phi$) **then**
5. $h \leftarrow$ point to the highest non-empty bitblock
6. **else** $h \leftarrow -2$

UPDATE(BW_n)

1. **if** ($l = -1 \parallel h = -2$) **return**
2. **while** ($BW_n^l = \phi$)
3. $l \leftarrow l + 1$
4. **if** ($l > h$)
5. $l \leftarrow -1, h \leftarrow -2$
6. **return**
7. **endif**
8. **endwhile**
9. **while** ($BW_n^h = \phi$)
10. $h \leftarrow h - 1$
11. **endwhile**

UPDATE_LOW(BW_n)

1. **if** ($l = -1 \parallel h = -2$) **return**
2. **while** ($BW_n^l = \phi$)
3. $l \leftarrow l + 1$
4. **if** ($l > h$)
5. $l \leftarrow -1, h \leftarrow -2$

```

6.         return
7.     endif
8. endwhile

```

```

IS_EMPTY( $BW_n$ )
1.  if( $l > h$ ) return TRUE else return FALSE

```

Finally, we consider a SET_SENTINELS ($BW_n, (l_1, h_1)$) operation which sets the sentinels of BW_n , i.e. $l = l_1, h = h_1, (l, h) \in BW_n$, independent of the actual state of the watched subgraphs $BW_n^{l_1}, BW_n^{h_1}$. This means that the operation is still executed even if the sentinels end up pointing to an empty block. SET_SENTINELS is helpful to *manually* circumscribe meaningful bitmask operations to a certain range, but leaves the bitstring in a potentially dangerous state, i.e. IS_EMPTY should never be called afterwards without a proper UPDATE of the watched bitstring.

3.5.2 The new BBMCW algorithm

This section describes the new BBMCW bit-parallel exact maximum clique algorithm tailored for semi-sparse graphs. BBMCW employs the watched bitstring encoding outlined in the previous subsection.

A first task to consider when inserting watched bitstrings into the REF_CLIQUE algorithm is to adapt the COLOR procedure (Listing 4) to the new encoding. Listing 8 describes COLORW, an effective way to implement COLOR using watched bitstrings. COLORW achieves a reasonable tradeoff between the overhead of sentinel up-keeping and the reduction in the number of superfluous bitmasking operations. In the algorithm, auxiliary sets COL ($BW_{|V|,COL}$) and UNCOL ($BW_{|V|,UNCOL}$) are now watched. Since in this application the allocated size of the encoding for all bitsets is $|V|$, it will be removed from subscripts in the bitstring notation hereafter.

In the initial step, BW_{UNCOL} receives the set of vertices to be colored U_v and updates the sentinels accordingly. The idea is that, if U_v contains vertices in just the i -th block BW_{UNCOL}^i , then all bitwise operations in COLORW will be

circumscribed to the single bitmask between the i -th blocks of whatever two sets are involved.

Once a color subset is computed, the sentinels in BW_{UNCOL} are again updated (step 13) to further restrict spurious operations. Note that sentinel information in BW_{UNCOL} is copied to BW_{COL} before the start of the inner loop (step 2). An optimization detail is that, inside the inner loop, only the lower sentinel of BW_{COL} is updated (step 7). Checking the upper sentinel is unnecessary because vertices in BW_{COL} are selected in order in step 4.

Listing 8. The new COLORW procedure, which implements COLOR in Listing 4 using *watched* bitstrings. The boxes mark the critical operations which benefit from bit-parallelism.

COLORW(BW_{U_v}, k_{min})

Input: 1) An induced subproblem graph $G[U_v]$
 2) A color number threshold $k_{min} = |S_{max}| - |S|k_{min}$

Output: 1) A (partial) vertex coloring of $G[U_v]$
 2) A set L of candidate vertices for branching in $G[U_v]$

i.1. INIT_WATCH($BW_{UNCOL} \leftarrow BW_{U_v}$)

i.2. $BW_{COL} \leftarrow \phi, L \leftarrow \phi, k \leftarrow 1, k_{min} \leftarrow \max(k_{min}, 1)$

1. **repeatuntil** IS_EMPTY(BW_{UNCOL})

2. $BW_{COL} \leftarrow BW_{UNCOL}$

3. **repeatuntil** IS_EMPTY(BW_{COL})

4. select the first vertex v from BW_{COL}^l

5. $BW_{COL} \leftarrow BW_{COL}/N(v)$ in range (BW_{COL}^l, BW_{COL}^h)

6. $BW_{UNCOL} \leftarrow BW_{UNCOL}/\{v\}$

7. UPDATE_LOW(BW_{COL})

8. **if** $k \geq k_{min}$ **then**

9. $C_k \leftarrow v$

10. $L \leftarrow L \cup \{v\}$

11. **endif**

12. **endrepeat**

13. UPDATE(BW_{UNCOL})

14. $k \leftarrow k + 1$

15. **endrepeat**

16. **return**($\{C_{k_{min}}, C_{k_{min}+1}, \dots, C_{k-1}\}, L$)

Finally, Listing 9 shows pseudocode for the new exact maximum clique algorithm BBMCW proposed in this work. The algorithm encodes the current subproblem U as a watched bitstring BW_U and, similar to watched COLOR procedures, updates its sentinels in the initial step. The advantages of the encoding with respect to REF_CLIQUE appear in the computation of child nodes (step 7) and in the evaluation of empty sets (step 9). In the first case, bitmasks are restricted to the sentinel range of $BW_U \setminus \{v\}$. In the second case, checking that a bitset is empty can now be computed in constant time.

Listing 9. The new exact maximum clique procedure BBMCW. The box marks the critical operation that benefits from bit-parallelism.

```

BBMCW( $U, S, S_{max}, C, L$ )

Input: A simple graph  $G = (V, E)$ 
Output: A maximum clique in  $S_{max}$ 

i.1.  $S \leftarrow \phi, S_{max} \leftarrow \phi, c(v_i) \leftarrow \min\{i, \Delta G + 1\}, L \leftarrow V$ 
i.2. INIT_WATCH( $BW_U \leftarrow U$ )
1.  repeat until  $L = \phi$ 
2.      select a vertex  $v$  from  $L$  in reverse order
3.      if  $(|S| + c(v) \leq |S_{max}|)$  then return
4.       $BW_U \leftarrow BW_U / \{v\}$ 
5.      UPDATE_SENTINELS( $BW_U$ )
6.       $S \leftarrow S \cup \{v\}$ 
7.       $BW_{U_v} \leftarrow BW_U \cap N(v)$  in range  $(BW_U^l, BW_U^h)$ 
8.      UPDATE_SENTINELS( $BW_{U_v}$ )
9.      if IS_EMPTY( $BW_{U_v}$ ) then
10.         if  $|S| > |S_{max}|$  then  $S_{max} \leftarrow S$ 
11.          $S \leftarrow S \setminus \{v\}$ 
12.         continue in step 2 with a fresh vertex
13.     endif
14.     if  $|V| < 5000$  and  $p \geq 0.1$  then //  $p$  is the average density of  $G$ 
15.          $\{C^v, L\} \leftarrow COLORW(BW_{U_v}, |S_{max}| - |S|)$ 
16.     else  $\{C^v, L\} \leftarrow COLORTW(BW_{U_v}, |S_{max}| - |S|)$ 
17.     endif
18.      $BBMCW(BW_{U_v}, S, S_{max}, C^v, L_v)$ 
19.      $S \leftarrow S \setminus \{v\}$ 
20. endrepeat
    
```

We further consider the following implementation details concerning the new encoding:

- *Neighbor bitsets $N(v)$, for all $v \in V$, do not need to be watched*: these bitsets encode each row of the adjacency matrix of the input graph and are therefore static, so sentinels would never change position. The current implementation of BBMCW uses the same encoding of the adjacency matrix as REF_CLIQUE.
- $\text{INIT_WATCH}(BW_{UNCOL} \leftarrow BW_{U_v})$, in the initial steps of COLORW, may be implemented in constant time by simply copying the sentinel information.
- *Every IS_EMPTY operation must be preceded by an adequate UPDATE*: it is imperative that sentinels are updated properly, else IS_EMPTY will fail. In the case of COLORW in Listing 8, watched bitstring BW_{UNCOL} is updated in step 13. COLORW checks BW_{UNCOL} to see if it is empty in the main outer loop. Also, in BBMCW (Listing 9), the empty set check in step 9 occurs after the corresponding UPDATE in step 8.

3.5.3 Experiments

This section first describes our test configuration and the different algorithms considered to validate CCMM. The end of the section provides extensive computational results. We have considered the following algorithms to validate the contribution presented in this work:

- I. BBMCI [33]: A state-of-the-art bit-parallel algorithm for small and middle-size graphs. It implements REF_CLIQUE described in section 3.2.
- II. BBMCW: The new algorithm described in this section. Compared with BBMCI, BBMCW employs the new watched bitstring encoding.
- III. IncMaxCLQ [36]: A state-of-the-art solver for exact maximum clique, which has reported very good performance for hard small and middle-size graphs.

- IV. BBMCSP [35]: The state-of-the-art exact maximum clique algorithm for massive sparse graphs. BBMCSP is a bit-parallel algorithm, but uses a novel compressed encoding, different from the one proposed in this work, to store the input graph in memory. It also includes a strong pre-processing step to reduce the scale of the problem, based on cores.

Algorithms I and III are representative MCP solvers for common graphs. As mentioned at the beginning of this chapter, the reason why BBMCSP has been selected for this report, instead of more sophisticated published variants with the same kernel is that BBMCSP performs better, on average, over the kind of (semi) sparse graphs which are the target of CCMM. The main reason for this is that, in the latter case, capturing structure is less relevant for overall performance.

All the experiments were performed on a *single core* of a Linux Intel(R) Xeon(R) CPU E5-2690 v2 multi-core workstation with 128GB of main memory. The algorithms tested are all implemented in C++ and compiled using *gcc4.8.1 -O2*. Only the user time taken for searching the graph is reported, i.e. the time spent reading the graph from a file and computing an initial solution is excluded. Time is measured in seconds in all tables.

Graph models arising from the CCMM algorithm have two parameters that determine the graph size and density. These parameters are the similarity threshold T and the number of points K in the second image that are considered valid pairwise matches of the same point in the first image. Thus, by varying K and T it is possible to generate correspondence graphs of different sizes and densities. Finally we also consider a threshold KP_{MAX} for the maximum number of keypoints in each image. This parameter is introduced in the keypoint detector algorithm (in our case SURF [6]).

Each row of Table 5 includes a demonstration keypoint association between two images of the corresponding benchmark. A complete description of the images used on this benchmark can be found on section 6. A suffix b added to the name of original the benchmark refers to a parameter configuration that leads to more vertices (*herzjesu* instances, for example, were too easy to solve for $KP_{MAX}=500$;

it was then set to 2000 keypoints in *herzjesu_b*). The particular instances reported have been selected from non-trivial problems, i.e. the two images matched are *reasonably* similar (and thus have many keypoints in common and a big maximum clique). The table also provides the configuration triplets (K, T, KP_{MAX}) for all the data sets used in this work.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Table 5 Image data sets used in this work. Parameter K refers to the maximum number of keypoints from the original image, which are candidates to match a single keypoint in the target image. Parameter T is a normalized threshold of distance similarity between pairs of keypoints.

<i>Name</i>	<i>K</i>	<i>T</i>	<i>KP_{MAX}</i>	<i>Example</i>
<i>herzjesu_b</i>	5	0.1	2000
<i>fpv1</i>	5	0.2	500
<i>fountain</i>	5	0.2	500
<i>castle</i>	5	0.2	500
<i>heliGrande</i>	5	0.2	500
<i>castle_b</i>	5	0.1	2000

3.6 Results

This section reports results obtained by different tests by the algorithms. We start by analyzing the impact of determining an initial large clique before starting the search step in the association graphs that arise in this application. Secondly, the algorithm CCMM is evaluated against the state-of-the-art filtering techniques. Lastly, we test the new solver BBMCW against state-the-art exact clique solvers for association graphs derived from our image data sets. All the data sets use in this work for validation are described in detail in Chapter 6.

3.6.1 Initialization of the clique solver with a large initial clique

We first consider the impact of using the heuristic algorithm AMTS to compute a good initial solution with which to feed the CCMM clique solver. Table 6 summarizes the general characteristics of the graph models obtained for all data sets, including the number of graphs (header *g*), the average number of vertices (header *n*) and the average density (header *d*). The table shows that graphs have low densities, but are not really sparse. We also note that concrete average densities present high variance (but never over 0.2).

Table 6 shows that AMTS improves speed in a small percentage of the graphs only (header AMTS+), the most relevant case being the *Heligrande* dataset. The reason is that this dataset presents good properties (i.e. images with low distortions, high overlap etc.) which accounts for denser and harder to solve graph models.

Table 6 Analysis of initial solution estimation using AMTS. *g* column refers to the number of association graphs (problems) studied for each data set, *n* is the average number of vertices and *d* the average density. AMTS+ column refers to the number of cases in which total time improved.

	g	n	d	AMTS+	%
Dino	513	615	0.05	4	0.77%
Heligrande	1691	1263	0.09	375	22.17%
Kapel	1273	1131	0.03	11	0.86%
Reduced	55	771	0.05	3	5.45%

Skycam	1411	770	0.05	14	0.99%
Ualberta	1710	306	0.03	4	0.23%

A more qualitative evaluation of the impact of AMTS is given in Figure 18 for the *Dino* data set (see Chapter 6). The histograms depict the time-to-solution distribution with (right) and without (left) AMTS. The overall conclusion is that AMTS should not be used by default but only in *good* scenarios, i.e. in which many similarities between pairs of interest points are to be expected. Note that the maximum time for any of the image matches is reduced using the heuristic from 1034ms to 532ms. However, the total average time is, for the full set, increased from 82ms, *without* heuristic, to 215ms *with* heuristic. Similar results are obtained in the *Heligrande* data set (Figure 19). In this case the maximum time is reduced from 1612ms to 1427ms. However average total time for the set increases slightly from 570ms to 599ms.

Figure 18. Performance of the new algorithm over the *Dino* benchmark. On the right, the heuristic algorithm AMTS is used to pre-compute a large initial clique. On the left, no heuristic is employed.

Figure 19. Performance of the new algorithm over the *Heligrande* benchmark. On the right, the heuristic algorithm AMTS is used to precompute a large initial clique. On the left, no heuristic is employed.

3.6.2 CCMM and RANSAC

Table 7, Table 8 and Table 9 report results for all the metrics defined in section 3.4 over representative data sets for RANSAC-H, RANSAC-F and new CCMM respectively. A first conclusion is that RANSAC-H is outperformed by the more sophisticated model employed by RANSAC-F in all datasets, so we will focus on the latter when comparing with CCMM.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Table 7.Results for RANSAC-H which uses a homography model.

	<i>Heligrande</i>	<i>Reduced</i>	<i>Ualberta</i>	<i>Kapel</i>	<i>Dino</i>	<i>Skycam</i>
Total number of points in frames	171,976	1,502	16,711	13,606	4,580	12,488
Mean number of frames per point	12	2	4	2	2	2
Mean number of >2 frames per point	16	3	6	3	3	3
Number of unique points	13,670	693	3,501	5,315	1,804	4,518
Number of unique points in >2 frames	9,862	96	2,023	1,638	573	1,744
Reprojection error	977,920	2,565	34,938	33,980	22,214	8,130

Table 8.Results for RANSAC-F which uses a projective model.

	<i>Heligrande</i>	<i>Reduced</i>	<i>Ualberta</i>	<i>Kapel</i>	<i>Dino</i>	<i>Skycam</i>
Total number of points in frames	312,930	3,929	39,467	25,401	9,442	22,539
Mean number of frames per point	25	2	9	3	3	3
Mean number of >2 frames per point	32	3	13	4	4	4
Number of unique points	12,285	1,547	4,188	7,981	2,987	6,575
Number of unique points in >2 frames	9,340	519	2,820	3,676	1,442	3,303
Reprojection error	996,589	13,925	63,988	33,980	25,511	28,422

Table 9. Results for CCMM.

	<i>Heligrande</i>	<i>Reduced</i>	<i>Ualberta</i>	<i>Kapel</i>	<i>Dino</i>	<i>Skycam</i>
Total number of points in frames	322,178	4,919	29,530	22,753	8,409	16,516
Mean number of frames per point	158	4	18	3	4	3
Mean number of >2 frames per point	187	6	25	6	6	5
Number of unique points	2,037	1,178	1,569	5,732	1,929	4,680
Number of unique points in >2 frames	1,719	607	1,145	2,774	1,043	2,275
Reprojection error	747,104	15,505	60,788	91,397	47,998	28,837

A first obvious conclusion is that the total number of points (first row in the tables) matched by RANSAC-F is greater in 4 out of 6 datasets (all except *Heligrande* and *Reduced*) than CCMM. Also both are better than RANSAC-L in all cases.

However, in order to correctly evaluate CCMM it is necessary to analyze the *quality* of the points. A more qualitative measure is given by the second and third rows in the tables, which consider the number of frames in which the same point is tracked consistently. CCMM has outstanding results in both metrics. Specifically, it is known that a point tracked in 3 or more frames is an important source of robustness for the final steps of the image processing chain (such as 3D reconstructions or mosaics). Figure 20 depicts this last result as a histogram for all datasets. This additional perspective has been included because of the relevance of the contribution.

Alternative qualitative metrics result from focusing on points instead of frames, in particular the number of unique (real) points tracked in two or more frames (row 4) and three or more frames (row 5). These metrics attempt to capture the information synthesis capability of the different methods. If a method has similar total number of points matched (row 1) and less unique point values, it

indicates that the features have been tracked along more frames, i.e. the association is more robust. CCMM performs comparably with RANSAC-F in *Ualberta*, *Kapel*, *Dino* and *Skycam*. In *Heligrande*, it outperforms RANSAC_F (note the total number of points is comparable, but with fewer unique point metrics). Finally it has to be said that CCMM is slightly outperformed by RANSAC_F in *Reduced* data set.

Figure 20. Mean number of three or more consecutive frames in which a point is tracked consistently in the 6 datasets considered.

The last metric considered is the reprojection error (row 6). To obtain this result, we have estimated the fundamental matrix F using the associations provided by each algorithm and computed the reprojection error with this matrix. As expected, RANSAC-F is the best method using these criteria because it attempts to minimize precisely this error. Noteworthy is that CCMM results are competitive even though the method doesn't use projective metrics. Worst performance for CCMM occurs in *Dino* and *Kapel* data sets, which also have the highest projective distortions. Also noteworthy is that CCMM performs comparably to RANSAC-F in *Skycam* which presents a lot of distortion. A possible explanation for this is that the nature of the distortion is not projective, and therefore not modeled well by F .

Figure 21. Total running times in milliseconds (Y-axis) for all algorithms over the *Reduced* dataset.

Concerning running times, RANSAC variants are notably quicker since CCMM is actually searching over a set of candidates and has to build the model for each association. However, we showed in proposition 3 that the order of the model is bounded linearly by the number of keypoint candidates in the target image considered in the association. When this number is low (not greater than 4) total computation time is also kept within reasonable bounds in practice.

Figure 21 shows total running times for all algorithms over the *Reduced* dataset. CCMM (without heuristic) computes the set around 4 times slower than RANSAC variants, but times are still reasonable for many applications.

Figure 22 presents an association example from each dataset. In all image pairs the green lines show the computed associations using RANSAC-F (from now on referred to simply as RANSAC) and the red lines show CCMM associations. We highlight the following points:

- *UAlberta* (A): In this structured environment the quality of the association is comparable in both algorithms.
- *Reduced* (B): The clique model produces more robust associations due to low overlapping between images and the presence of repetitive patterns, which make it more difficult for RANSAC

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

- *Skycamera* (C): These images of the sky are very hard for both algorithms due to cloud dynamics. In this type of difficult scenarios the clique model provides more consistent matches on average.
- *Kapel* (D): The RANSAC algorithm is outperformed on this set as well due to abundance of repetitive patterns. However distortions are big because of the different viewpoints and there are some mismatches in both cases.
- *Heligrande* (E): Both algorithms perform similarly because the images present good characteristics (i.e. very high overlap and low distortions).
- *Dino* (F): RANSAC is defeated here because of the repetitive patterns.

A) A sample from UAlberta	B) A sample from Reduced

C) A sample from Skycamera	D) A sample from Kapel

E) A sample from Heligrande	F) A sample from Dino
<p>Figure 22. A sample of feature matching from each of the reported data sets. In green (top) associations using RANSAC-F; in red (bottom) associations given by the new CCMM algorithm.</p>	

3.6.3 BBMCW

BBMCW proves to be consistently better than the other state-of-the-art algorithms reported over image correspondence graphs (see Table 10). For instance, it averages a two order of magnitude improvement with respect to IncMaxCLQ. Also, the difference in performance between BBMCW and BBMC I is more acute as the size of the graph increases.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Table 10 Comparison between different exact clique solvers for different data sets. Times in seconds are shown on the first to third columns and columns four and five contains the relative improvement of BBMCW over IncMaxCLQ and BBMCI respectively.

	<i>IncMaxCLQ</i>	<i>BBMCI</i>	<i>BBMCW</i>	<i>BBMCW/Inc</i>	<i>BBMCW/BBMCI</i>
name	time(s)	time(s)	time(s)		
<i>herzjesu_b_1</i>	218	58.6	16.9	12.9	3.47
<i>herzjesu_b_2</i>	319	4.42	1.15	278	3.85
<i>herzjesu_b_3</i>	57	2.18	0.557	102	3.92
<i>herzjesu_b_4</i>	191	7	1.72	111	4.08
<i>herzjesu_b_5</i>	267	2.5	0.629	424	3.97
<i>herzjesu_b_6</i>	108	2.37	0.575	187	4.12
<i>herzjesu_b_7</i>	100	38.6	10.2	9.81	3.78
<i>fpv1_1</i>	574	3.69	0.925	620	3.99
<i>fpv1_2</i>	644	7.49	1.91	337	3.92
<i>fpv1_3</i>	552	4.18	1.05	525	3.98
<i>fpv1_4</i>	519	3.75	0.955	544	3.92
<i>fpv1_5</i>	760	5.31	1.35	562	3.93
<i>fpv1_6</i>	554	3.41	0.862	643	3.95
<i>fpv1_7</i>	352	5.08	1.28	275	3.96
<i>fpv1_8</i>	704	5.76	1.45	487	3.98
<i>fpv1_9</i>	457	3.93	0.991	461	3.97
<i>fpv1_10</i>	390	3.05	0.759	514	4.01
<i>fountain_1</i>	20.8	0.268	0.114	182	2.35
<i>fountain_2</i>	7.18	0.156	0.081	88.7	1.93
<i>castle_1</i>	15.4	0.204	0.103	149	1.98
<i>castle_2</i>	8.39	0.262	0.139	60.3	1.88
<i>castle_3</i>	12.5	30.2	16.1	0.777	1.87

<i>castle_4</i>	10.5	2.2	1.6	6.52	1.37
<i>castle_5</i>	7.91	0.191	0.089	88.9	2.15
<i>castle_6</i>	9.52	34.3	11.8	0.804	2.89
<i>castle_7</i>	6.56	0.793	0.622	10.5	1.27
<i>heliGrande_1</i>	7.47	0.09	0.047	159	1.91
<i>heliGrande_2</i>	12.8	0.095	0.052	245	1.83
<i>heliGrande_3</i>	7.66	0.089	0.054	142	1.65
<i>heliGrande_4</i>	8.04	0.109	0.058	139	1.88
<i>heliGrande_5</i>	10.7	0.084	0.046	233	1.83
<i>castle_b_1</i>	17	0.502	0.209	81.5	2.4
<i>castle_b_2</i>	6.55	6.55	1.66	3.94	3.94
<i>castle_b_3</i>	7.07	3.88	1.18	6.01	3.29
<i>castle_b_4</i>	5.18	16.2	11.2	0.461	1.44

To describe the overall effects of the clique search algorithm on a complete data set we present aggregated results for several ones. Results show that BBMCW is faster than BBMC, which is consistent with the results shown on the previous table. On the other hand, other variants do not provide any speedup; on the contrary, more often than not, they can reduce speeds up to 4 times. This confirms the ideas presented on section 3.2 about the semi-sparse nature of CCMM graphs. Also interesting are the bad results obtained by BBMCSP, the sparse version of the BBMC algorithm. The penalty of the sparse representation is too high for the type of graphs concerning this work, which are all denser than 0.1 (Table 5).

Table 11 Comparison between different exact clique solving algorithms for different datasets. Times are normalized against the BBMC algorithm and averaged for the complete dataset. BBMCW is the only algorithm that improves BBMC times.

	reduced	fpv1	fountain	herzjesu
<i>BBMC</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>
<i>BBMC T</i>	<i>2,0231</i>	<i>3,895</i>	<i>1,7954</i>	<i>2,1179</i>
<i>BBMCR</i>	<i>1,2861</i>	<i>2,47</i>	<i>1,7223</i>	<i>1,8487</i>
<i>BBMCL</i>	<i>0,9133</i>	<i>1,2415</i>	<i>1,3263</i>	<i>1,2657</i>
<i>BBMCL T</i>	<i>1,7735</i>	<i>3,9851</i>	<i>1,9954</i>	<i>2,4841</i>
<i>BBMCL R</i>	<i>1,1392</i>	<i>2,4234</i>	<i>1,7711</i>	<i>1,9174</i>
<i>BBMC W</i>	<i>0,8223</i>	<i>0,671</i>	<i>0,9963</i>	<i>0,9879</i>
<i>BBMC WT</i>	<i>1,9867</i>	<i>4,1722</i>	<i>2,2231</i>	<i>2,5145</i>
<i>BBMCSP</i>	<i>27,9332</i>	<i>>2h</i>	<i>> 2h</i>	<i>>2h</i>

4 RECONSTRUCTION TECHNIQUES

One relevant application of feature matching is the Structure from Motion (SFM) problem. SFM is a relevant part of satellite earth observation, aerial photogrammetry, robot navigation, medical imaging and other related fields. In this section, the main reconstruction techniques used in SFM will be outlined. For a clearer exposition, we have divided the reconstruction techniques into two groups: I. constructive algorithms which improve an initial reconstruction incrementally, and II. reconstruction based on global optimization criteria.

The first type, also called *filtered-based* or *filtered*, adds, at each step, fresh observations to a current solution, thus incrementally improving reconstruction. The second type, computes a solution taking into account all the observations at the same time. Although the latter type provides more accurate reconstructions, in practice it is only used for off-line applications. For fast real-time reconstruction, filtered techniques should be preferred.

Current state-of-the-art techniques usually employ ideas from both categories. Thus, filtered techniques usually create submaps that are then integrated in a single bundle adjustment, as in the so called *keyframe methods*. Also, global

optimization-based implementations can insert frame by frame into an initial solution, which is then optimized.

The results for image reconstruction obtained in this thesis have been more successful using the optimization algorithms compared to the filtering algorithms. This point of view was also corroborated in [37], where key frame optimizations showed a better cost/accuracy ratio than their counterparts. For this reason, this thesis has focused mainly on global optimization techniques.

Before describing a second important contribution of this work, we briefly describe in section 4.1 state-of-the-art filtering techniques and, in section 4.2, current optimization techniques. The contribution of this work to the latter includes a number of new initialization strategies which are presented in Chapter 5.

4.1 Filtering techniques

The SFM problem has some variants depending on the availability of different sensors. For example, there are slight differences if a monocular camera is considered compared to a stereo one. Especially relevant to this work is the problem in which only a monocular camera is employed, because a reliable feature matching step is critical for robust reconstruction. In this scenario, the CCMM algorithm can have a big impact in the quality of the final solution.

There are two main filtered-based implementations of SFM algorithms for monocular camera systems. One uses an extended Kalman filter (EKF) and the other is based on the Rao-Blackwellised particle filter. The former stores the information of the observations into both a state and a covariance matrix, and this representation is refined with each observation. In contrast, the particle filter approach uses a set of particles to store the current solution. Successive observations adjust the particle positions which represent possible hypothesis of the scene parameters and points.

State-of-the-art has primed EKF solutions over particle-based ones because the Rao-Blackwellised filters were published later on time and there are less examples of successful monocular SFM which use them. Note that these last

filters have lower computational requirements than EKF-based ones, when the size of the reconstruction exceeds a certain threshold. On the other hand, the different randomness nature of the observations and the camera movement model usually leads the particle filter solution to an unstable state which makes it difficult to use and adjust. Moreover, the particle filter's inherent random nature makes the results non-repeatable making the experimental analysis more difficult. We now briefly describe the details of the EKF implementation. The details of the particle filter are described in depth in, for example, [38].

The Extended Kalman filter for SFM is formulated using a process matrix described by the equations that model the movement of the camera and a set of keypoints. The observation matrix applied is based on the specific projection equations given by the camera model. The state vector is composed by the location of the features and camera. The algorithm uses a solid moving at constant speed as model for the system, although alternative models are possible if a priori knowledge is available. Additionally, it employs the pinhole camera as measurement model for the camera sensor. Other assumptions include that the intrinsic parameters of the pinhole model are assumed invariants and that the features observed by the images are fixed elements in the scene. The difficulty of the algorithm relies on the accuracy of the system and sensor models, the adjustment of the filter parameters and the initialization of the algorithm.

EKF requires an initialization stage where the following information must be provided:

- Intrinsic parameters of the camera.
- Extrinsic parameters (initial) of the camera.
- The three-dimensional position of a first set of features.
- The Bi-dimensional position, in the image space, of the features on two consecutive frames.

The above set of parameters describes the state of the system. It is noteworthy that the set includes information about two consecutive moments: i.e. most of the values of the elements of the state concern two successive positions, for

example, the position of the camera and the relevant features in the image space. Relevant to initialization is that it determines the scale of the future reconstructed scene.

After initialization, EKF iterates over the following two phases:

- Computes the Kalman filter
- Maintains consistency of the current scene and refines it.

The Kalman filter initialized in the previous phase evolves with the models and the measurements of the system. Here, the term *system* refers to the state of the camera and the state of the features that appear in the scene. *Measurements* refer to feature observations in the image space.

The upkeeping of the scene includes the following tasks:

- Identifying the features that disappear from the field of vision and remove them from the model of the sensor that uses the Kalman filter.
- Adding fresh features to the system in such a way that they stay within the field of vision for a minimum number of frames.
- Recover the features that re-enter the field of vision from the outside.

We now briefly discuss the mathematical models use by EKF and how they are used in practice by the algorithm.

On EKF the state of the system is defined by two variables:

- $\hat{X}_{k|k}$, the state considered in time k given the observations up to the time k
- $P_{k|k}$, the error covariance matrix, which represents an estimation of the precision of the considered state of the system considered in time k given the observations up to the time k

The state the system, in our case, is formed by the state of the camera and the state of the characteristics:

$$X = \begin{bmatrix} X_{\text{camera}} \\ X_{\text{characteristics}} \end{bmatrix}$$

Assuming that the camera moves with six degrees of freedom with uniform movement, its model will be:

$$\begin{aligned}x &= x_o + v_x \cdot \Delta t + w_x \\y &= y_o + v_y \Delta t + w_y \\z &= z_o + v_z \Delta t + w_z \\r &= r_o + \omega_r \Delta t + v_r \\p &= p_o + \omega \Delta t + v_p \\q &= q_o + \omega \Delta t + v_q\end{aligned}$$

where (r, p, q) are the components of the rotation vector.

Approximating the speeds by differences we have:

$$\begin{aligned}v_k &= x_k - x_{k-1} / \Delta t \\ \omega_k &= \varphi_k - \varphi_k / \Delta t\end{aligned}$$

and solving for x on time k :

$$x_k = x_{k-1} + x_{k-1} - x_{k-2} = 2x_{k-1} + x_{k-2} \quad (11)$$

we can remove the angular velocity from the above expressions. Following (11) we can now express the state of one of the variables as:

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}$$

We assume that 3D observed points are static therefore its model will be an identity matrix.

The complete model will have $12 + 3 \times n$ state variables as follows:

$$\begin{bmatrix} x_k \\ x_{k-1} \\ y_k \\ y_{k-1} \\ z_k \\ z_{k-1} \\ \vdots \\ x_{ci} \\ y_{ci} \\ z_{ci} \end{bmatrix} = \begin{bmatrix} 2 & -1 & & & & & & & & \\ 1 & 0 & & & & & & & & \\ & & 2 & -1 & & & & & & \\ & & 1 & 0 & & & & & & \\ & & & & 2 & -1 & & & & \\ & & & & 1 & 0 & & & & \\ & & & & & & \ddots & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ y_{k-1} \\ y_{k-2} \\ z_{k-1} \\ z_{k-2} \\ \vdots \\ x_{ci-1} \\ y_{ci-1} \\ z_{ci-1} \end{bmatrix} + [0] \quad (12)$$

Where x_k is the camera position on the instant k and x_{ci} is the feature c position at the instant i .

The covariance of the model is initialized to a first diagonal. The terms of the diagonal depend on the variable related to the model of the sensor.

As stated previously, the model of the sensor is a pinhole model. It is therefore nonlinear, and, because of this, requires the use of an Extended Kalman Filter (notice this has the added difficulty of computing the Jacobian). The model is a function of the form $f(x, y, z|X, \theta) = (x', y')$. This function is defined in homogenous coordinates as follows:

$$\begin{bmatrix} n \cdot x' \\ n \cdot y' \\ n \end{bmatrix} = \begin{bmatrix} f & c_x \\ & f & c_y \\ & & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

where f is the focal length and (c_x, c_y) is the optical center of the image, r_{ij} and t_x, t_y, t_z are the elements of the rotation and translation matrix of the camera position, (x', y') are the image coordinates of the projected point and finally (x_c, y_c, z_c) are the point coordinates in the 3D space.

The Kalman filter has two phases, the phase of prediction and the phase of correction.

In the phase of prediction, the state is estimated using the model of the system and its corresponding matrix of covariance of the error calculation.

$$\begin{aligned} \hat{X}_{k|k-1} &= F_k \hat{X}_{k-1|k-1} + B_k u_k \\ P_{k|k-1} &= F_k P_{k-1|k-1} F_k^T + Q_k \end{aligned}$$

where F_k and B_k are the matrices that define the evolution of the state of the system, based on the previous state $X_{k-1|k-1}$ and u_k is the control input (see equation (11)). Note that, in our specific application u_k does not exist. However, if there is a priori knowledge about the camera movement, as in the case of a robot, the control input can be used to add this information to the filter. Finally, Q_k is the error covariance matrix of the system model.

Concerning the correction stage, measurements are employed to repair the predicted state of the system. This stage may be implemented by the following steps:

- Calculation of the innovation or remainder of the measurement

$$y_k = z_k - H_k \hat{X}_{k|k-1}$$
- Calculation of the covariance of the innovation $S_k = H_k P_{k|k-1} H_k^T + R_k$
- Calculation of the Kalman gain $K_k = P_{k|k-1} H_k^T S_k^{-1}$
- Update of the state $\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k y_k$
- Update of the covariance $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

In the extended Kalman filter (EKF), the matrix of state transitions and the observation model do not need to be linear functions; in fact, they can even be non-differentiable functions of the form:

$$\begin{aligned} x_k &= f(x_{k-1}, u_k, w_k) \\ z_k &= h(x_k, v_k) \end{aligned} \tag{13}$$

Functions f and h are used to calculate the predicted states using the previous estimation. To update the covariance matrix the Jacobian matrix, computed from the partial derivatives of (13) is employed. This is equivalent to the linearization of a nonlinear function around the current state.

When the points leave the field of vision, it is necessary to modify the matrix of measurement of the Kalman filter. This is the simplest case, since it is only necessary to eliminate the row and corresponding column of the measurement matrix.

The easiest way to detect a point that leaves the field of vision is to remove it when it is near the edge. This strategy is not optimal for the following reasons:

- A point can return back to the field of vision before leaving
- A point that leaves the field of vision without crossing through the edge might produce a mistaken identification.

When the number of visible points is below a minimum threshold, it is necessary to add a fresh point. The difficulty of this stage lies in an adequate initialization. A bad initialization can cause the divergence of the Kalman filter or a non-recoverable error in the mapping.

The initialization of a point is a process that must be made throughout several cycles a sufficiently ample movement is necessary to be able to triangulate the point accurately. The initialization algorithm has three phases:

- Generation of possible points
- Track the points throughout matching along several frames
- Triangulate the points and filter possible invalid matches

The first step towards point generation is to find a point in the image that has good characteristics to be followed. This can be obtained by means of feature extraction algorithms such as SIFT, SURF or AKAZE as shown in section 2.2. When we have a candidate list of points, it is necessary to verify that they are not near to points already considered, and attempt to select points in those sections of the field of view with less features.

The selected points are tracked throughout several frames using the matching techniques described in section 2.3. The goal is to obtain a three-dimensional estimation of these points by triangulating the observations in several views. In order to make this estimation, we used an adjustment of square minimums on the simplified model of the image presented in (14).

$$\begin{bmatrix} n \cdot x' \\ n \cdot y' \\ n \end{bmatrix} = \begin{bmatrix} f & c_x \\ f & c_y \\ 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (14)$$

$$\begin{bmatrix} n \cdot x' \\ n \cdot y' \\ n \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (15)$$

solving (15)

$$\begin{aligned} (p_{31}x_w + p_{32}y_w + p_{33}z_w + p_{34})x' &= (p_{11}x_w + p_{12}y_w + p_{13}z_w + p_{14}) \\ (p_{31}x_w + p_{32}y_w + p_{33}z_w + p_{34})y' &= (p_{21}x_w + p_{22}y_w + p_{23}z_w + p_{24}) \end{aligned} \quad (16)$$

Simplifying (16) and applying it to a pair of views of the same point we have

$$\begin{bmatrix} p_{31}x'_1 - p_{11} & p_{32}x'_1 - p_{12} & p_{33}x'_1 - p_{13} \\ p_{31}y'_1 - p_{21} & p_{32}y'_1 - p_{22} & p_{33}y'_1 - p_{23} \\ p'_{31}x'_2 - p'_{11} & p'_{32}x'_2 - p'_{12} & p'_{33}x'_2 - p'_{13} \\ p'_{31}y'_2 - p'_{21} & p'_{32}y'_2 - p'_{22} & p'_{33}y'_2 - p'_{23} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} p_{34}x' - p_{14} \\ p_{34}y' - p_{24} \\ p'_{34}x'_2 - p'_{14} \\ p'_{34}y'_2 - p'_{24} \end{bmatrix}$$

$$AX = B$$

The estimated position of the feature is solved by minimum squares:

$$\bar{X} = A^T(AA^T)^{-1}B$$

The detection of points that re-enter the field of vision is made by projecting each 3D point to the estimated camera parameters, and verifying if it falls within the field of vision. If this is so, a matching step is performed around the projected point. This algorithm is not robust if there is an important deviation of the reconstruction before closing the loop.

4.2 Optimization techniques

The alternative to filtering techniques is describing the structure from motion problem (SFM) in the form of an optimization problem. The goal is to minimize the projection error of the reconstructed features.

The projected point may be described by the following general model:

$$x = F(X, I, E)$$

where X is the feature position in the tri-dimensional space, I the intrinsic parameter vector and E the extrinsic parameter vector. Intrinsic parameters remain constant all through the optimization problem while extrinsic parameters change in each frame. Examples of both types are: focal and image centers, considered intrinsic, and rotation and translation which fall in the extrinsic category. A cost function for the above general model can be defined as:

$$J = \sum x_{obs} - x = \sum x_{obs} - F(X, I, E)$$

The projection model may be any of the models described in section 2.1, such as pinhole or fisheye. Consequently, the previous optimization function will be strongly non-linear. This implies that, although the problem is expressed as a least squares problem, nonlinear optimization techniques to estimate the

position of the tri-dimensional points are required. The most common solution for this specific problem is the Levenberg-Marquardt algorithm. It employs a parametrizable step that allows the algorithm to switch between a gradient-descent and a Newton step depending on the improvement obtained in the previous iteration. In the general case, this reduces the total number of iterations while being robust to non-linearities.

The problem dimension depends on the number of features and observations present on the scene. When the scene is composed by hundreds of points and thousands of observations, a generic implementation of the algorithm can lead to excessive memory and computation requirements. To sidestep this situation, the structure of the problem can sometimes be exploited.

We now first proceed to describe the standard Levenberg-Marquardt algorithm, and later describe the improved variant, which takes into account the sparse nature of the problem.

4.2.1 Levenberg-Marquardt

We present here the mathematical foundations of this optimization algorithm. For specific implementation details we refer the reader to [39].

Given a function $\hat{x} = f(p)$ that estimates a measurement vector x using a set of parameters p , our objective is to find the vector p that minimizes $\sum (x - \hat{x})^2$. The Levenberg-Marquardt (L-M) algorithm is a least squares optimization algorithm. Its model function is approximated by its Taylor approximation:

$$f(p + \delta_p) = f(p) + J\delta_p$$

where J is the jacobian matrix $\frac{\partial f(p)}{\partial p}$. From the above approximation it can be inferred that the value of δ_p that minimizes the error can be obtained solving the so-called normal equations:

$$J^T J \delta_p = J^T \epsilon \tag{17}$$

where $\epsilon = (x - \hat{x})$ is the error vector for the current estimation of the measurement. The matrix $J^T J$ is an approximation to the hessian matrix. In

practice, the L-M algorithm employs an extended version of this matrix called the normal matrix N such that:

$$N\delta_p = J^T \epsilon$$

where the elements of N are the same as the ones of $J^T J$, except for the main diagonal which is substituted by $J^T J_{ii} + \mu$. The parameter μ is called the *damping term* which allows the algorithm to adjust the length of the step according to the evolution of the optimization error. Specifically, in those steps where the reduction of the error is big, the damping term is reduced. Conversely, a high error increases the damping term and the algorithm takes a steepest-descent direction. Moreover, the damping term has the effect of avoiding situations where the normal matrix can be singular. We note that this can be of great importance when traversing a non-linear optimization field.

If a covariance matrix is available for the measurements then it can be incorporated to the L-M algorithm by using a weighted version of the equation (17):

$$J^T \sum J \delta_p = J^T \sum \epsilon$$

The algorithm terminates either when the magnitude of the gradient is low, the error evolution is low, the step is too small or a maximum number of iterations has been reached.

4.2.2 Bundle adjustment implementation

When solving the optimization problems that arise from bundle adjustment, the Normal matrix mentioned in the previous section is typically sparse. In [40], the authors take advantage of this fact to reduce the total computation cost very significantly. Because of its practical importance, we describe the improved normal equations (17) for the sparse case.

We assume that n 3D points are seen in m views and use notation x_{ij} to denote the projection of the i -th point on image j . Also, each camera j is parameterized by a vector a_j , and each 3D point i by a vector b_i . With this setup, the goal is to minimize the reprojection error given by:

$$e = \sum_{i=1}^n \sum_{j=1}^m (f(a_j, b_i) - x_{ij})^2$$

Let k and λ define the dimension of vectors a and b respectively; then the dimension of the complete parameter vector space can be given by $m\kappa + n\lambda$, which can be very large even for small bundle adjustment problems. If we denote A_{ij} and B_{ij} the derivatives $\frac{\partial x_{ij}}{\partial a_j}$ and $\frac{\partial x_{ij}}{\partial b_i}$ respectively, then the Jacobian matrix can be expressed in the following form, for a given a set of 4 images and 3 points:

$$J = \frac{\partial x}{\partial p} = \begin{bmatrix} A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\ 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\ 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\ A_{21} & 0 & 0 & 0 & B_{21} & 0 & 0 \\ 0 & A_{22} & 0 & 0 & B_{22} & 0 & 0 \\ 0 & 0 & A_{23} & 0 & B_{23} & 0 & 0 \\ A_{31} & 0 & 0 & 0 & 0 & B_{31} & 0 \\ 0 & A_{32} & 0 & 0 & 0 & B_{32} & 0 \\ 0 & 0 & A_{33} & 0 & 0 & B_{33} & 0 \\ A_{41} & 0 & 0 & 0 & 0 & 0 & B_{41} \\ 0 & A_{42} & 0 & 0 & 0 & 0 & B_{42} \\ 0 & 0 & A_{43} & 0 & 0 & 0 & B_{43} \end{bmatrix}$$

As shown in [40], the structure of the normal equations is the following:

$$\begin{pmatrix} U & W \\ W & V \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \epsilon_a \\ \epsilon_b \end{pmatrix}$$

where U and V are diagonal block matrix defined by:

$$\begin{aligned} U_j &= \sum_i A_{ij}^T \Sigma_{x_{ij}}^{-1} A_{ij} \\ V_i &= \sum_j B_{ij}^T \Sigma_{x_{ij}}^{-1} B_{ij} \\ W_{ij} &= A_{ij}^T \Sigma_{x_{ij}}^{-1} B_{ij} \\ \epsilon_{a_j} &= \sum_i A_{ij}^T \Sigma_{x_{ij}}^{-1} \epsilon_{ij} \\ \epsilon_{b_i} &= \sum_j B_{ij}^T \Sigma_{x_{ij}}^{-1} \epsilon_{ij} \end{aligned}$$

U and V are augmented by adding μ to yield U_j^* and V_i^* . Compute δ_a from $S\delta_a^T = e^T$ where $S_{jk} = \delta_{jk}U_j^* - \sum_i Y_{ij} W_{ik}^T$, given that δ_{jk} is Kronecker's delta, $Y_{ij} = W_{ij}V_i^{*-1}$ and $e_j = \epsilon_{a_j} - \sum_i Y_{ij} \epsilon_{b_i}$.

δ_b is computed from the equation $\delta_b = V_i^{*-1} \left(\epsilon_{b_i} - \sum_j W_{ij}^T \delta_{a_j} \right)$

This formulation allows problems with thousands of variables to be solved two orders of magnitude faster than when using a generic version of the algorithm.

To wrap up this discussion, this algorithm needs an initial estimation of its parameters, cameras and points positions, to begin the optimization. If this initial estimation is not close to the final solution, then the optimization algorithm can end into local minima or even diverge to an inconsistent reconstruction. In the next chapter we will propose different heuristics that improve state-of-the-art bundle adjustment with an innovative graph based approach.

5 BUNDLE ADJUSTMENT

INITIALIZATION

This chapter deals with the initialization step of the bundle adjustment problem derived from image reconstruction. As shown in Figure 23, this step is prior to the optimization algorithm and takes as input the set of filtered image observations correctly matched for all image pairs.

Figure 23. Bundle Adjustment image processing chain

The initialization step starts with the estimation of all the relative displacements between image pairs. This estimation is a costly operation so if the number of combinations is high, techniques such as bag of words [41] [42] can be

implemented to reduce the analysis to the most likely ones. To compute the relative displacement we first require the estimation of the fundamental matrix F using the RANSAC algorithm. The algorithm uses the 8-point algorithm from [12] and is a good robust estimator for F at this stage of the reconstruction (see section 2.4.1).

The next step is creating an absolute estimation of the positions both of the camera and the 3D features, i.e. an initial solution. This absolute estimation starts by a pair or a triplet of images [43] and is then extended incrementally by adding a fresh image at each step. Bundle adjustment is known to be very sensitive to this initial solution and, rather surprisingly, there is no systematic study of this step in literature to the best of our knowledge [44] [45]. Note that, due to error accumulation, a complete initial solution that uses all the frames may not even be possible to obtain. A good example is the case of a long sequence of images taken along a line starting in A and ending in B. If we start adding images from A to B then the accumulated error in B will be big. If we add from B to A then the error will be bigger in the frames near to A. If we start from a point in the middle of both, the error will be shared between A and B. This last situation is much more desirable in the case of bundle adjustment, because it has less deviation from the real solution and is, therefore, less susceptible to fall in a local minima.

Initially, baseline positions are inferred using the decomposition described in [12]. Tridimensional camera positions for the baseline are extracted from the *essential matrix* E . This matrix relates the coordinates of two views of the same scene in a similar way as the fundamental matrix. The difference is that the former works in a normalized coordinate space, whereas the fundamental matrix works in the original image space. An estimation of the essential matrix can be obtained from the fundamental matrix F according to the equation:

$$E = K^T F K$$

where K is the calibration matrix as explained in section 2.1.1.

From here, taking the Singular Value Decomposition (SVD) of the essential matrix and assuming that a first camera is placed at the coordinates origin

$P=[I|o]$, there are four possible solutions $\{P_1, P_2, P_3, P_4\}$ for the matrix model of the second camera as follows:

$$P_1 = \left[U \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^T \middle| U_3 \right]$$

$$P_2 = \left[U \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^T \middle| -U_3 \right]$$

$$P_3 = \left[U \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^T \middle| U_3 \right]$$

$$P_4 = \left[U \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} V^T \middle| -U_3 \right]$$

where U, V are the results from the SVD

The geometrical explanation of this ambiguity is shown in Figure 24. The four depicted figures ((a), (b),(c), (d)) differ with respect to the positions of the cameras along the baseline. The difference between pairs ((a),(b)), as well as ((c), (d)), is a 180° rotation of the camera. It should be noted that only solution (a) has the observation point in front of both cameras.

Figure 24. Four possible solutions for camera positions derived from the essential matrix.

We solve the ambiguity by selecting the solution with most points in front of both cameras (option (a) in the example) as described in [44]. Moreover, it is customary to apply a filter to the feature points at this stage, so that those which fall behind the cameras are removed.

Iterative Linear LS triangulation methods are used to estimate the position of the points in the 3D space [45]. Suppose that $u = Px$ where u are the homogenous coordinates of the observed point. If we decompose the matrix notation by rows we have:

$$\begin{aligned} wu &= p_1^T x \\ wv &= p_2^T x \\ w &= p_3^T x \end{aligned} \tag{18}$$

where p_i is the i -th row of the projection matrix, P . By substituting w in equation (18) on the other two we have:

$$\begin{aligned} up_3^T x - p_1^T x &= 0 \\ vp_3^T x - p_2^T x &= 0 \end{aligned}$$

This defines a set of four linear equations that can be written on the form $Ax = 0$ where A is a 4×4 matrix. By setting $x = (x, y, z, 1)^T$ we can use the pseudoinverse or SVD of A to obtain a least-squares solution for the position of point x .

The problem of this approach is that the value being minimized is $\|Ax\|$ instead of the projection error shown in Figure 25. To overcome this, we appropriately introduce fresh weights w, w' in the expressions $w = p_3^T x$ and $w' = p_3'^T x$, that compensate possible errors in the measures.

Figure 25. Projected points are observed in coordinates x and x' , and due to measurement errors they define rays that do not intersect in space. As a result, the measurements do not satisfy the epipolar restriction. This hardens the triangulation problem using the fundamental matrix.

However, since the weights depend on x they cannot be determined a priori. We therefore employ an iterative algorithm where the initial estimation of x is done by using the *unweighted* equations, and weights are estimated *a posteriori* with the new value of x that results from the previous iteration. This process may be stopped when the changes in the weights are small between consecutive steps.

In spite of all the previous considerations, it has been reported in [45] that the initialization algorithm described can still lead to unstable situations where it

diverges. Consequently, a convergence test should be run at the end of each step to filter out problematic points that could potentially be inserted in the final solution.

Another important issue during the initialization step of bundle adjustment is that it is necessary to estimate the absolute position of the camera shot for every fresh frame considered by the algorithm, To this avail, the set of points that have already been reconstructed are again detected in the new frame. This step can be described as the problem to obtain the pose of a camera from a set of n point 3D coordinates and its 2D projections, also called the PnP problem. In this work, we apply the EPnP algorithm described in [46], which solves the PnP problem in one step without the need of a costly iterative procedure. Once the absolute position of the fresh frame has been estimated, more points are added to the 3D reconstructed set by triangulating the new observations. This process continues for each remaining frame: the algorithm iterates over the set of already reconstructed frames estimating the frame's absolute position and triangulating new points. Listing 10 outlines the INSERT_FRAME procedure just described. It essentially updates a list of 3D reconstructed points from a list of keypoints and a list of matches of keypoints.

Listing 10. The frame insertion procedure called by the initialization algorithm

```

INSERT_FRAME ( $R, first\_flag, F_{01}, matches, keypoints, n$  )
Input:  $R$ : reconstructed points list,  $first\_flag$ : Boolean to indicate if this is
      the first pair of the reconstruction,  $F_{01}$  fundamental matrix of the first
      pair,  $matches$ : matches list,  $keypoints$ : 2D point list,  $n$ : id of the frame to
      be inserted
Output: Updated list of reconstructed points in  $R$ , Absolute camera position
        $P_n$ 

1. if ( $first\_flag = \text{TRUE}$ ) then
2.    $first\_flag \leftarrow \text{FALSE}$ 
3.    $P_0 \leftarrow I$ 
4.    $P_1 \leftarrow \text{DECOMPOSE} ( F_{01}, matches )$ 
5. else
6.   FIND points FROM keypoints VIEWED IN  $n$  and  $points \in R$ 
7.    $P_n \leftarrow \text{EPnP}(R, keypoints)$ 
8. endif

```

9. $X \leftarrow \text{TRIANGULATE}(\text{matches})$
10. $R \leftarrow R \cup X$

5.1 General considerations for frame ordering

An important contribution of this work is using common graph techniques and metrics, such as minimum spanning tree, eccentricity, minimum distance path and others, to determine the order in which frames are presented to the initialization algorithm of bundle adjustment. To the best of our knowledge, this is a novel idea which leads to different sorting heuristics. We consider in this work a relevant set of 6, some of which are included for validation purposes. Note that, for each selected frame in order, the `INSERT_FRAME` procedure in Listing 10 is called to insert it in the current solution.

We encode association information between pairs of frames as a weighted undirected graph $G = (V, E)$, where nodes represent frames and the weight of an edge $w(e)$, $e \in E$, $w(e) \in \mathcal{R}^*$, represents the pairwise adjustment error between the two endpoint nodes (frames). If the adjustment between any two frames is not possible due to lack of common points, the corresponding nodes have no edge. Conversely, an edge of weight 0 implies a perfect overlap. This error graph is the starting point to determine the way frames are presented to the initialization step of the bundle adjustment.

Also relevant to this work is a measure of graph centrality. The eccentricity $\epsilon(v)$ of a graph vertex v in a connected graph G , is defined as the maximum distance between v and any other vertex $u \in G$. The maximum eccentricity is the *graph diameter* and the minimum graph eccentricity is the *graph radius*. A vertex with minimum eccentricity is therefore in a central position in the graph, since, from there, any other node is within range of, at most, the graph radius. Nodes in central positions in our error graph makes the corresponding frames good candidates as starting baseline for our initialization algorithm of the bundle adjustment.

The pairwise reconstruction error (also each weight of our error graph) is defined by the following equation:

$$err = \sum_{i \in S} [P_{1x}^i, P_{1y}^i, 1]^T F [P_{2x}^i, P_{2y}^i, 1] \quad (19)$$

where F is the fundamental matrix estimated using the RANSAC algorithm described in section 2.4.1, and (P_{1x}^i, P_{1y}^i) and (P_{2x}^i, P_{2y}^i) , are the projections of the i -th 3D point (in image coordinates) in the corresponding frames. In all the experiments, RANSAC is configured to consider a point to be an outlier when the distance from its projection to the epipolar line is greater than a 3 pixel threshold.

Besides graph-based criteria, which will be defined later on in the section, we further consider the following 2 methods to determine frame order:

- *Maximum common points*: the state-of-the-art procedure where the next frame inserted is the one with more points in common with the set of current 3D reconstructed points. The starting frame pair is selected as the one with minimum error according to the fundamental matrix F . This heuristic is described in [47] and [48].
- *Lexicographical order*: frames are sorted according to their filename. In many data sets, filenames of frames also encode the original order in which the images were taken. We have selected this method, whenever possible, instead of a random order, because it is more robust than the latter and usually leads to the convergence of the algorithm.

Combining the two above methods with the new graph-based ones we have studied and tested a total of 9 different sorting heuristics for the frames. The listing is as follows:

- Maximum number of points in common starting from the pair of frames with minimum error (MAX PTS MIN ERR).
- Maximum number of points in common starting from a single frame with minimum eccentricity (MAX PTS MIN ECC).
- Minimum spanning tree (MST) of the error graph, traversed in a depth-first fashion (DEPTH FIRST).
- Minimum spanning tree (MST) of the error graph, traversed according to minimum distance (MIN DIST MST).

- Minimum distance starting from the pair of frames with minimum error (MIN DIST MIN ERR).
- Minimum distance from the node with minimum eccentricity of the error graph (MIN DIST MIN ECC).
- Minimum distance from the node with minimum eccentricity when circumscribed to the MST of the error graph (MIN DIST CENT MST).
- Minimum path as determined by solving a Travelling Salesman Problem (TSP).
- Sequential, i.e. frames are selected in the order they were shot if the information is available (SEQ).

A more detailed explanation for each case appears later on in the section. Thus, subsection 5.2 is concerned with heuristics based on the maximum number of points in common with the current solution, subsection 5.3 with heuristics based on minimum distance, subsection 5.4 deals with minimum spanning tree-based ones and 5.5 describes the TSP-based frame order.

A taxonomy for the different frame orderings is presented in Table 12. Strategies are classified according to different combinations of baseline selection (the initial set of frames), the graph considered and the actual sorting criteria for the rest of the frames.

Table 12. A taxonomy for initialization strategies

Baseline Selection	Graph	Ordering
Minimum Eccentricity	MST	Depth First
	Full Graph	Minimum Distance
Minimum Error	None	Max. Common Points
Virtual Node	Full Graph	TSP
Image #0	None	Sequential

The *Minimum Error* baseline strategy selects the pair of frames with minimum error as defined by equation (19), i.e. the two endpoints of the edge with minimum weight of our error graph. The *Minimum Eccentricity* baseline criterion selects as single initial frame a node v in the error graph with

minimum eccentricity ϵ selecting the minimum error in case of having the same eccentricity. *Virtual node* refers to adding a fresh node to the error graph such that it is connected to all other nodes by zero weighted paths. This node will be the starting and ending node of a TSP. Finally, *Image #0* determines the first image in the sequence according to its filename, which, as explained previously, usually contains a temporal code.

The *Graph* column in Table 12 contains the following three categories: methods that are not graph-based (labeled as *None*), methods that use the error graph (labeled as *Full Graph*) and methods that use the minimum spanning tree of the error graph as reference for some other metric (label MST).

Finally, the last column of Table 12 classifies five sorting heuristics that are related to the initial baseline and error graph by the values of the corresponding cells in the same row. *Depth First* refers to the traversal of the MST of the error graph from a (central) starting point to its leaves, selecting nodes in a depth-first fashion. When a branch is inserted up to the leave next branch ordered by error is started. *Minimum Distance* refers to the traversal of the error graph by selecting, at each step, the node which is nearest to a central node (i.e. with minimum eccentricity). The strategy referred to as *Maximum Common Points* is not graph-based, as explained previously. Instead, it takes the frames which contain the maximum number of points that are already reconstructed at the moment of insertion. The *TSP* strategy determines frame order as the route solution of a Travelling Salesman Problem over the error graph, when a fresh virtual node is added as starting and ending city of the route. The idea behind this is to obtain a frame ordering with minimum accumulated error. Finally, the *Sequential* category simply sorts the frames according to the timestamp in the filenames whenever possible.

To wrap up this introductory section, we consider three main different types of baseline selection: *minimum error*, *minimum eccentricity* and *virtual node*. The sequential algorithm, which is presented here as contrast method, takes images in the shooting order (and its baseline is denoted by *Image#0* in the table). Moreover, and according to how the error is propagated to determine

frame order, we further classify strategies in terms of using the proposed error graph G , its minimum spanning tree (MST) and not considering G at all.

In the following sections, the different heuristics contribution of this work are examined in detail.

5.2 Maximum number of points in common

The state-of-the-art approach reported in the literature [48] [47] for the initialization step of a bundle adjustment is to select a starting baseline amongst all the frame pairs using a metric of geometrical *goodness*. After that, the rest of the frames are inserted one by one according to the maximum number of points they have in common with the set of already reconstructed (3D) points. We denote this heuristic MAX_COMMON_POINT (Listing 11) and take it as our reference procedure for validation.

Listing 11 Initial reconstruction by selecting frames according to non-increasing number of points in common with the current solution

```

MAX_COMMON_POINT(error)
Input: a list of pairs of frames (pairs)

1.  $fp \leftarrow \text{MIN}(\textit{error})$  //find pair with minimum error
2.  $F \leftarrow \text{FindFundamental}(fp)$ 
3.  $\text{INSERT\_FRAME}(R, \text{true}, F, \text{matches}, \text{kpoints}, fp)$  //Listing 10
4.  $\textit{pairs} \leftarrow \textit{pairs} \setminus \{fp\}$ 
5. while ( $\textit{pairs} \neq \emptyset$ )
6.      $np \leftarrow \max(\text{findmatchesOf}(\text{3D\_positions}, \textit{pairs}))$  //next pair
7.      $\text{INSERT\_FRAME}(R, \text{false}, \text{null}, \text{matches}, \text{kpoints}, np)$ 
8.      $\textit{pairs} \leftarrow \textit{pairs} \setminus \{fp\}$ 
12 endwhile
    
```

The initialization's starting baseline is usually selected as the pair of frames with minimum error. Then, the algorithm inserts the images according to a non-increasing number of 3D points in common (step 6). To evaluate the sensitivity of the algorithm with respect to different starting points we have carried out a number of tests which are reported in Table 13.

Table 13. Reconstruction data following a maximum-common-point ordering with two different baselines. #pnts is the number of 3d points on the scene. #frms the number of frames inserted in the solution. #proj is the total number of projections.

RMSE is the root-mean-square error of the projected points. In bold best strategy for each dataset

	MIN. ECCECNTRICITY				MIN. ERROR			
	#pts	#frms	#proj	RMSE	#pts	#frms	#proj	RMSE
<i>reduced</i>	1	2	2	0.02	151	7	330	106108
<i>aerial_view_1</i>	133	6	334	0.26	111	5	263	0.12
<i>aerial_view_2</i>	98	6	204	0.71	6	2	12	0.00
<i>dinosaur</i>	11	2	22	0.00	889	36	2011	299.41
<i>merton_college1</i>	220	3	467	0.10	220	3	467	0.10
<i>model_house</i>	1	2	2	0.01	121	7	290	0.38
<i>wadhamcollege</i>	417	5	930	0.10	0	0	0	0.00
<i>kapel</i>	1	2	2	0.05	1	2	2	0.00
<i>corridor</i>	297	11	676	0.15	575	10	1838	0.35
<i>library</i>	22	2	44	0.21	34	2	68	0.21
<i>merton_college3</i>	92	3	205	0.18	12	2	24	0.09
<i>castle_p19</i>	6	2	12	0.00	180	6	397	0.08
<i>castle_p30</i>	178	7	439	0.33	2	2	4	0.00
<i>entry-p10</i>	114	3	253	0.05	56	2	112	0.04
<i>fountain-p11</i>	0	0	0	0.00	92	6	225	0.33
<i>herz-jeus-p25</i>	0	0	0	0.00	3	2	6	0.00
<i>desktop</i>	132	4	269	0.12	164	3	330	0.13
<i>teacup</i>	281	6	701	0.32	243	5	551	0.32
<i>herzjesu-r23</i>	3	2	6	0.00	9	2	18	0.00
<i>fpv1_r</i>	4416	108	11758	0.31	4048	107	11313	0.32
<i>airview</i>	892	34	2643	0.20	1021	33	2762	0.28
<i>albert-laser-vision</i>	1368	72	6090	0.10	1299	70	7197	1.74
<i>castle-r20</i>	39	4	97	0.10	201	8	468	0.47
<i>dinoSparseRing</i>	22	2	44	0.12	0	0	0	0.00
<i>templeSparseRing</i>	321	4	656	0.03	28	2	56	0.12
<i>fountain-r25</i>	720	21	1734	0.16	507	19	1256	0.31
<i>vuelo6</i>	1	2	2	0.01	1132	36	2967	0.38

The table shows results for MAX_COMMON_POINT with a different starting frame pair, based either on minimum eccentricity or on minimum error. In the table, *#pnts* refers to the number of 3D points of the scene successfully reconstructed, *#frms* the number of frames inserted on the solution, *#proj* the total number of projections and RMSE the root-mean-square error of the projected points.

Results show that minimum eccentricity provides a slightly better starting point than the minimum error point for MAX_COMMON_POINT. The former is better or comparable in 15 of the 27 datasets, against 13 of the latter. However, differences are small and the performance of both strategies is overall comparable.

A more qualitative analysis considers data set types (see Chapter 6 for a detailed description of each dataset). Here, we see that both algorithms behave similarly over structured (*merton_college*, *castle*) and non-structured (*reduced*, *airview*) environments as well as with different geometrical distributions. We note that similar data sets like the *herz-jesu-p5* and *herz-jesu-p28* can lead to different results with the same starting point heuristic.

5.3 Frame ordering based on minimum distance

By a minimum distance criterion we refer to a procedure that starts from a central node (frame) in a graph, as determined by eccentricity, and sorts the remaining nodes according to the distance to this central point. We use the well-known Dijkstra algorithm to obtain the minimum distances of all the nodes to this central node. Listing 12 outlines the algorithm for the sake of completeness.

Listing 12. The Dijkstra algorithm

```

DIJKSTRA ( $G = (V, E)$ ,  $source \in V$ )
Outputs: the distance of all the other nodes in  $V$  to  $source$ 

1.  $dist[source] \leftarrow 0$ 
2.  $seen[source] \leftarrow 1$ 
3. create vertex set  $Q$ 
4. for each vertex  $v$  in  $G$ :
5.     if  $v$  is neighbor( $source$ ) then
6.          $dist[v] \leftarrow length(source, v)$ 
7.     else
8.          $dist[v] \leftarrow INFINITY$ 
9.     endif
10. next  $v$ 
11. while ( $seen[] = \emptyset$ )
12.      $v \leftarrow \text{minimum } dist[] \text{ where } seen[] = \emptyset$ 
13.      $seen[v] \leftarrow 1$ ;
14.     for each  $w$  in neighbors ( $v$ ):
15.         if  $dist[w] > dist[v] + length(v, w)$ 
16.              $dist[w] \leftarrow dist[v] + length(v, w)$ 
17.         endif
18.     next  $w$ 
19. endwhile

```


20. **return** *dist*[]

As in the previous case, we can compare the performance of the proposed frame ordering using different starting points.

Table 14 reports results for three cases: using as baseline the node with minimum eccentricity in the error graph G (first column), the pair of nodes with a minimum error (second column) and the node with minimum eccentricity considering only the edges of the minimum spanning tree of G (third and final column).

Table 14. Reconstruction data following a minimum distance based ordering of frames with different baselines. Parameters are the same as in previous tables.

	<i>min. eccen.</i>				<i>min. err</i>				<i>MST min. eccen.</i>			
	#pts	#frms	#proj	RMSE	#pts	#frms	#proj	RMSE	#pts	#frms	#proj	RMSE
<i>reduced</i>	42	4	92	0.18	328	11	750	60.86	393	10	873	0.28
<i>aerial_view_1</i>	108	5	229	0.20	57	6	119	0.21	103	5	230	0.27
<i>aerial_view_2</i>	99	6	233	0.76	65	5	128	0.20	10	2	20	0.00
<i>dinosaur</i>	566	12	1383	0.49	379	13	904	0.21	368	11	893	0.36
<i>merton_college1</i>	154	3	327	0.10	158	3	331	0.09	154	3	327	0.10
<i>model_house</i>	94	5	216	0.21	43	7	88	0.06	0	0	0	0.00
<i>kapel</i>	14	2	28	0.01	140	4	278	0.02	145	8	346	108.90
<i>corridor</i>	592	11	1845	0.31	317	11	770	0.10	618	11	1831	0.16
<i>library</i>	5	2	10	0.00	5	2	10	0.00	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09	144	3	311	0.12	101	3	213	0.09
<i>castle_p19</i>	29	2	58	0.19	51	4	112	0.22	6	2	12	0.00
<i>castle_p30</i>	224	9	531	53.49	370	14	861	0.32	122	8	284	0.22
<i>entry-p10</i>	173	3	356	0.29	188	3	389	0.48	173	3	356	0.29
<i>fountain-p11</i>	6	2	12	0.00	85	9	177	0.18	68	5	158	0.30
<i>herz-jeus-p25</i>	38	2	76	0.04	122	13	266	0.42	22	5	50	0.11
<i>herz-jesu-p8</i>	12	2	24	0.16	26	3	55	0.24	43	4	93	0.53
<i>desktop</i>	240	4	506	0.14	21	2	42	0.34	13	2	26	0.03
<i>teacup</i>	152	3	322	0.50	92	4	194	0.07	373	6	855	0.05
<i>herzjesu-r23</i>	71	10	158	0.12	110	6	216	0.04	10	2	20	0.00
<i>dolmen</i>	180	6	375	0.10	241	2	482	0.03	3	2	6	0.00
<i>fpv1_r</i>	4002	108	10666	0.37	3754	107	9384	0.45	3845	108	11065	0.36
<i>Airview</i>	896	31	2872	0.38	869	29	2139	0.20	0	0	0	0.00
<i>albert-l-vision</i>	1821	72	7968	0.18	1657	59	5105	0.15	1403	65	7985	11.59
<i>castle-r20</i>	34	4	80	0.27	98	7	226	0.43	13	2	26	0.01
<i>dinoSparseRing</i>	167	6	367	0.27	200	8	430	0.33	22	2	44	0.12
<i>templeSparseRing</i>	380	4	776	0.02	338	2	668	0.00	334	3	676	0.03
<i>fountain-r25</i>	674	24	1738	0.45	989	24	2317	0.17	443	22	1006	0.37
<i>vuelo6</i>	1541	37	4199	0.31	1140	36	2478	0.21	1207	36	3197	0.32

The table shows that performances obtained when choosing as origin of distances the frame with minimum error are better on average than with the

other methods. This start point is better in 12 datasets against 10 when using the minimum eccentricity of the full graph and 5 when using the min. eccentricity along the MST of the graph.

Minimum eccentricity along the MST appears to be the worse strategy to choose the starting node and only provides slightly better results on degenerate reconstructions where the results of the three methods are bad, as in *teacup* or *kapel*

5.4 Frame ordering based on the MST

Instead of employing the error graph G as basis for the different sorting criteria, it is also possible to selectively consider non-induced subgraphs $G' = (V', E')$, where $V' = V$ since all the frames have to be sorted, but $E' \subset E$, i.e. some of the original edges are removed. Specifically, we are interested in the edges E' determined by the minimum spanning tree (MST) of G . This set of edges has the property of connecting all the nodes without any cycles and with minimum total edge weight. We consider in this section graph-based heuristics over the subgraph $G' \subset G$ determined by the edges of the MST of G . We compute G' using the well-known Kruskal algorithm (outlined in Listing 13 for completeness).

Listing 13 Kruskal algorithm pseudocode

```

KRUSKAL ( $G$ )
Output:  $A \leftarrow$  minimum spanning tree

1.  $A \leftarrow \emptyset$ 
2. foreach  $v \in G$ 
3.    $SET_v \leftarrow \emptyset$ 
4. next  $v$ 
5. foreach  $(u, v)$  ordered by non-decreasing weight( $u, v$ )
6.   if  $SET_v \neq SET_u$  then
7.      $A \leftarrow A \cup \{(u, v)\}$ 
8.      $SET_{uv} \leftarrow SET_u \cup SET_v$ 
9.   endif
10. next  $(u, v)$ 
11. return  $A$ 

```

In all figures the node labeled o (and the first node in the output ordering) is the central node (its eccentricity equals the graph radius). Fig. 27 presents the node ordering obtained by a depth-first traversal of the MST: starting from the minimum eccentricity along the tree node, branches are taken according to minimum distance until the leaf node is reached. This heuristic is considered mainly for validation purposes. In Fig. 29, the nodes are sorted according to the Dijkstra algorithm, which takes as source the central node o (see section 5.3). Conversely, in Fig. 28, the nodes are sorted considering only distances along the MST (highlighted in black). Table 15 compares the performance of the 3 heuristics.

The minimum distance frame ordering of Fig. 29 was described in the previous section (table 14, MIN_ECCEN.). Note however that the ordering depicted in Fig. 28, which is also minimum distance-based, is new. Here, it is the distance to the central node o, measured along the MST, which determines the ordering.

Fig. 26. A demonstration graph with MST highlighted	Fig. 27. Ordering along MST, following a depth-first traversal

Fig. 28. Minimum distance ordering along the MST	Fig. 29. Minimum distance ordering along the full graph

Table 15. Minimum Spanning Tree-based frame ordering results. The column marked as *Depth First* refers to traversing each branch of the MST in order. *MIN DIST* refers to the heuristic reported in the previous section (Table 14). Finally, *MIN DIST MST* sorts the nodes according to non-decreasing distance along the MST.

	<i>DEPTH FIRST</i>				<i>MIN DIST MIN ECC</i>				<i>MIN DIST MST</i>			
	#pts	#frms	#proj	RMSE	#pts	#frms	#proj	RMSE	#pts	#frms	#proj	RMSE
<i>reduced</i>	393	7	882	0.03	42	4	92	0.18	422	10	991	0.05
<i>aerial_view_1</i>	109	5	232	0.20	108	5	229	0.20	108	5	229	0.20
<i>aerial_view_2</i>	10	2	20	0.00	99	6	233	0.76	10	2	20	0.00
<i>dinosaur</i>	351	10	856	0.31	566	12	1383	0.49	368	11	893	0.36
<i>merton_college1</i>	154	3	327	0.10	154	3	327	0.10	154	3	327	0.10
<i>model_house</i>	0	0	0	0.00	94	5	216	0.21	0	0	0	0.00
<i>kapel</i>	219	9	466	33.60	14	2	28	0.01	0	0	0	0.00
<i>corridor</i>	485	11	1381	0.19	592	11	1845	0.31	553	11	1667	0.16
<i>library</i>	5	2	10	0.00	5	2	10	0.00	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09	101	3	213	0.09	101	3	213	0.09
<i>castle_p19</i>	6	2	12	0.00	29	2	58	0.19	6	2	12	0.00
<i>castle_p30</i>	216	9	491	0.17	224	9	531	53.49	96	6	220	0.22
<i>entry-p10</i>	173	3	356	0.29	173	3	356	0.29	173	3	356	0.29
<i>fountain-p11</i>	62	4	137	0.25	6	2	12	0.00	107	6	243	0.24
<i>herz-jeus-p25</i>	88	6	191	0.27	38	2	76	0.04	20	3	43	0.63
<i>herz-jesu-p8</i>	54	4	115	0.14	12	2	24	0.16	43	4	93	0.53
<i>desktop</i>	13	2	26	0.03	240	4	506	0.14	13	2	26	0.03
<i>teacup</i>	393	6	964	0.07	152	3	322	0.50	439	6	1074	0.06
<i>herzjesu-r23</i>	10	2	20	0.00	71	10	158	0.12	10	2	20	0.00
<i>dolmen</i>	3	2	6	0.00	180	6	375	0.10	3	2	6	0.00
<i>fpv1_r</i>	3844	108	11606	0.40	4002	108	10666	0.37	3594	108	9849	0.33
<i>airview</i>	612	20	2002	0.40	896	31	2872	0.38	545	30	1527	0.20
<i>albert-l-vision</i>	1454	73	7622	0.12	1821	72	7968	0.18	0	0	0	0.00
<i>castle-r20</i>	29	3	65	0.15	34	4	80	0.27	13	2	26	0.01
<i>dinoSparseRing</i>	22	2	44	0.12	167	6	367	0.27	22	2	44	0.12
<i>templeSparseRing</i>	330	2	660	0.00	380	4	776	0.02	334	3	676	0.03

<i>fountain-r25</i>	604	20	1561	0.43	674	24	1738	0.45	973	21	2302	0.16
<i>vuelo6</i>	1057	36	2912	0.45	1541	37	4199	0.31	1260	37	3338	0.42

The Depth First heuristic was expected to perform worse than the other methods, as it should intuitively lead to a greater accumulated error that is not compensated by those neighboring frames waiting to be inserted. The results clearly confirm this behavior as the only instances where depth first reconstruct more points, kapel and herzjesu, are partial reconstructions.

Concerning minimum distance along the MST, the results contradicted our initial expectations. By circumscribing the distance criterion to a subgraph that minimized the global error metric, we expected an improvement in performance. From the experiments we conclude that the stabilizing effect of neighbor frames is more beneficial than attempting to reduce the overall aggregate error of every pair of frames (i.e. the total sum of the weights of the graph). Note that in most of the data sets reported, MIN DIST MST has worst, or at most not better, performance than MIN DIST. The only notable exceptions are reduced and teacup datasets that are almost complete and using MIN DIST MST and quickly diverge with MIN DIST. Whoever those instances are not enough to infer a usage pattern.

Overall, the minimum distance strategy when applied over the error graph, outperforms MST-based ones.

5.5 The Travelling Salesman strategy

This heuristic finds the shortest path between all the nodes in the error graph. Such a path minimizes the sum of the errors (weights) computed for each pair of frames, and thus was considered a potentially good sorting criteria

To find this shortest path we formulate the problem as a Travelling Salesman Problem (TSP), a well-known problem in graph theory that determines the shortest path to visit a list of cities exactly once, and return to the origin city. Here the cities are nodes (also frames), and each edge weight of the error graph determines the distance between the corresponding two cities.

The TSP formulation requires selecting an origin node, which is the starting and ending point of the tour. In our case we avoid this selection by adding a fresh *zero node* with null distance to every other node, and take it as origin (see Figure 30). Finally, we solve the TSP with the very efficient Concorde implementation [49].

Figure 30. Ordering using TSP strategy

The results of this strategy were very different to what we had expected initially (see Table 16). In the table, hyphens in cells indicate that the algorithm failed to provide a valid initial solution, i.e. the fundamental matrix could not be estimated. This might be explained by the fact that, although the sorting strategy minimizes a certain total error, it does not capture properly the local structure, i.e. does not take into account the impact of the neighbors already inserted in the model in previous iterations.

Table 16. TSP frame ordering results. Rows with hyphens refer to data sets where the algorithm was not able to produce a valid initial solution.

	TSP					TSP			
	#pts	#frms	#proj	RMS		#pts	#frms	#proj	RMS
<i>reduced</i>	1	2	2	0,00	<i>fountain-p11</i>	0	2	0	--
<i>aerial_view_1</i>	--	--	--	--	<i>herz-jesus-p25</i>	0	2	0	--
<i>aerial_view_2</i>	--	--	--	--	<i>herz-jesu-p8</i>	2	2	4	0,00
<i>dinosaur</i>	1352	18	3491	0,51	<i>Teacup</i>	283	8	723	1,31
<i>merton_college1</i>	--	--	--	--	<i>herzjesu-r23</i>	6	2	12	0,00
<i>model_house</i>	0	2	0	--	<i>dolmen</i>	0	2	0	--
<i>wadhamcollege</i>	--	--	--	--	<i>airview</i>	0	2	0	--
<i>merton_college2</i>	--	--	--	--	<i>albert-laser-vision</i>	1099	73	4605	0,14
<i>corridor</i>	876	11	3190	0,54	<i>castle-r20</i>	0	2	0	--

<i>Library</i>	--	--	--	--	<i>dinoSparseRing</i>	0	2	0	--
<i>merton_college3</i>	--	--	--	--	<i>templeSparseRing</i>	11	2	22	0,00
<i>castle_p19</i>	138	7	345	1,69	<i>fountain-r25</i>	6	2	12	0,00
<i>castle_p30</i>	0	2	0	--	<i>Piedra</i>	1	2	0	--
<i>entry-p10</i>	285	8	655	0,69	<i>vuelo6</i>	5	2	10	0,00

5.6 Experiments and Results

In this section we analyze the impact of the proposed heuristics, over the reconstructed scene output of the bundle adjustment. All the experiments were performed on a Linux Intel(R) Xeon(R) CPU E5-2690 v2 multi-core workstation with 128GB of main memory. The algorithms tested are all implemented in C++ and compiled using gcc 4.8.1 -O2. The test architecture is based on Jenkin tasks. A complete task for all datasets takes between 12 and 24 hours. Each dataset execution is limited to 2 hours. For a detailed description of every dataset we refer the reader to chapter 6.

To evaluate the quality of the reconstructed image we use the number of points inserted in the final solution as metric. Points are not inserted in the solution if they have a root-mean-square error (RMSE) greater than 5 pixels. Using the adjustment's total RMSE is not representative, since, for example, degraded solutions with very few points usually have a very good RMSE value. Moreover, algorithms that identify a similar number of points and frames also tend to have a similar RMSE

Table 17 shows a compact representation of the number of points reconstructed for each data set, in which cell values are normalized according to the best result in each row, i.e. an index of 1 means that the method reconstructed the highest number of points. For the full report, including number of reconstructed points, matched frames and RMSE error we refer the reader to Appendix I.

Table 17. Experimental Results for the number of 3D points matched correctly according to the different sorting criteria. Results are normalized according to the best result for each dataset. An index of 1 indicates the highest number of points was matched.

<i>max points</i>	<i>MAX PTS</i>	<i>MAX PTS</i>	<i>DEPTH FIRST</i>	<i>MIN DIST</i>	<i>MIN DIST</i>	<i>MIN DIST</i>	<i>MIN DIST</i>	<i>SEQ</i>
-------------------	--------------------	--------------------	------------------------	---------------------	---------------------	---------------------	---------------------	------------

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

	<i>MIN ECC</i>	<i>MIN ERR</i>		<i>MIN ECC</i>	<i>MIN ERR</i>	<i>MST</i>	<i>CENTER MST</i>	
<i>reduced</i>	0.00	0.36	0.93	0.10	0.78	1.00	0.93	0.03
<i>aerial_view_1</i>	1.00	0.83	0.82	0.81	0.43	0.81	0.77	0.01
<i>aerial_view_2</i>	0.96	0.06	0.10	0.97	0.64	0.10	0.10	1.00
<i>dinosaur</i>	0.01	1.00	0.39	0.64	0.43	0.41	0.41	0.01
<i>merton_college1</i>	1.00	1.00	0.70	0.70	0.72	0.70	0.70	0.85
<i>model_house</i>	0.01	0.63	0.00	0.49	0.23	0.00	0.00	1.00
<i>kapel</i>	0.00	0.00	1.00	0.06	0.64	0.00	0.66	0.23
<i>corridor</i>	0.48	0.93	0.78	0.96	0.51	0.89	1.00	0.74
<i>library</i>	0.65	1.00	0.15	0.15	0.15	0.15	0.15	0.00
<i>merton_college3</i>	0.64	0.08	0.70	0.70	1.00	0.70	0.70	0.64
<i>castle_p19</i>	0.03	1.00	0.03	0.16	0.28	0.03	0.03	0.23
<i>castle_p30</i>	0.48	0.01	0.58	0.61	1.00	0.26	0.33	0.00
<i>entry-p10</i>	0.61	0.30	0.92	0.92	1.00	0.92	0.92	0.92
<i>fountain-p11</i>	0.00	0.52	0.35	0.03	0.48	0.60	0.38	1.00
<i>herz-jeus-p25</i>	0.00	0.02	0.72	0.31	1.00	0.16	0.18	0.01
<i>herz-jeus-p8</i>	0.00	0.00	1.00	0.22	0.48	0.80	0.80	0.22
<i>desktop</i>	0.55	0.68	0.05	1.00	0.09	0.05	0.05	0.00
<i>teacup</i>	0.64	0.55	0.90	0.35	0.21	1.00	0.85	0.67
<i>herzjesu-r23</i>	0.02	0.07	0.08	0.57	0.88	0.08	0.08	1.00
<i>dolmen</i>	0.00	0.00	0.01	0.75	1.00	0.01	0.01	0.00
<i>fpv1_r</i>	1.00	0.92	0.87	0.91	0.85	0.81	0.87	0.90
<i>airview</i>	0.83	0.96	0.57	0.84	0.81	0.51	0.00	1.00
<i>albert-l.-vision</i>	0.75	0.71	0.80	1.00	0.91	0.00	0.77	0.70
<i>castle-r20</i>	0.19	1.00	0.14	0.17	0.49	0.06	0.06	0.47
<i>dinoSparseRing</i>	0.11	0.00	0.11	0.84	1.00	0.11	0.11	0.11
<i>templeSparseRing</i>	0.84	0.07	0.87	1.00	0.89	0.88	0.88	0.92
<i>fountain-r25</i>	0.73	0.51	0.61	0.68	1.00	0.98	0.45	0.60
<i>vuelo6</i>	0.00	0.73	0.69	1.00	0.74	0.82	0.78	0.72
media	0.412	0.499	0.532	0.605	0.665	0.460	0.464	0.500
>0.7	8	10	12	12	14	10	10	11
>0.95	4	6	2	5	7	3	1	5

The reported results for 28 datasets shows that minimum distance ordering along the full graph with origin the pair of frames with minimum error, MIN DIST MIN ERR (highlighted in grey) provides better initial solutions for bundle adjustment than the other algorithms. It averages 0.665 over the normalized number of points index. The reference method, MAX PTS MIN ERR, has 0.499. Also MIN DIST MIN ERR obtains more than 0,7 in 14 of the datasets and more than 0.95 in 7.

Table 18. Comparison between the reference ordering of frames and the best graph-based method. In bold, the best performance for each row

	MAX PTS MIN ERR				MIN DIST MIN ERR			
	#Pts	#frms	#proj	RMSE	#Pts	#frms	#proj	RMSE
<i>reduced</i>	151	7	330	106108	328	11	750	60.8
<i>aerial_view_1</i>	111	5	263	0.12	57	6	119	0.21
<i>aerial_view_2</i>	6	2	12	0.00	65	5	128	0.20
<i>dinosaur</i>	889	36	2011	299.41	379	13	904	0.21
<i>merton_college1</i>	220	3	467	0.10	158	3	331	0.09
<i>model_house</i>	121	7	290	0.38	43	7	88	0.06
<i>wadhamcollege</i>	0	0	0	0.00	0	0	0	0.00
<i>kapel</i>	1	2	2	0.00	140	4	278	0.02
<i>corridor</i>	575	10	1838	0.35	317	11	770	0.10
<i>library</i>	34	2	68	0.21	5	2	10	0.00
<i>merton_college3</i>	12	2	24	0.09	144	3	311	0.12
<i>castle_p19</i>	180	6	397	0.08	51	4	112	0.22
<i>castle_p30</i>	2	2	4	0.00	370	14	861	0.32
<i>entry-p10</i>	56	2	112	0.04	188	3	389	0.48
<i>fountain-p11</i>	92	6	225	0.33	85	9	177	0.18
<i>herz-jeus-p25</i>	3	2	6	0.00	122	13	266	0.42
<i>herz-jesu-p8</i>	0	0	0	0.00	26	3	55	0.24
<i>desktop</i>	164	3	330	0.13	21	2	42	0.34
<i>teacup</i>	243	5	551	0.32	92	4	194	0.07
<i>herzjesu-r23</i>	9	2	18	0.00	110	6	216	0.04
<i>dolmen</i>	0	0	0	0.00	241	2	482	0.03
<i>fpv1_r</i>	4048	107	11313	0.32	3754	107	9384	0.45
<i>airview</i>	1021	33	2762	0.28	869	29	2139	0.20
<i>albert-laser-vision</i>	1299	70	7197	1.74	1657	59	5105	0.15
<i>castle-r20</i>	201	8	468	0.47	98	7	226	0.43
<i>dinoSparseRing</i>	0	0	0	0.00	200	8	430	0.33
<i>templeSparseRing</i>	28	2	56	0.12	338	2	668	0.00
<i>fountain-r25</i>	507	19	1256	0.31	989	24	2317	0.17
<i>pedra</i>	0	0	0	0.00	0	0	0	0.00
<i>vuelo6</i>	1132	36	2967	0.38	1140	36	2478	0.21

Table 18 compares the detailed results of the reference method, MAX PTS MIN ERR, and the new proposed ordering, MIN DIST MIN ERR. It can be seen that our proposed method is better in most of the instances. Also its results are comparable in the instances where MAX PTS MIN ERR ordering is better with the exception of the dinosaur dataset. The opposite is not true, in some of the datasets where the new ordering outperforms the reference method the latter has poor results, as seen on *dinoSparseRing*, *templeSparseRing*, *herz-jesu*, *fountain-p11*, *castle_p30*.

We end this section with a brief number scene reconstructions for a selection of data sets. In the ensuing figures, the black dots are the reconstructed tridimensional positions of the observed keypoints. Relevant reconstructed

camera positions and orientations are shown as blue pyramids, which represent the projection frame of a pinhole model.

Table 19. Reconstruction results from the *corridor* dataset. The top part shows a matching example of two images using CCMM. The bottom part depicts reconstruction results using the minimum distance ordering (left) and the ordering based on maximum common points starting with the pair of frames with minimum error (right).

Table 19 shows an image registration example of a pair of images from the *corridor* data set using our CCMM algorithm (top). The bottom part is the final reconstructed scene when bundle adjustment uses the best ordering heuristic (MIN DIST MIN ERR, left) and the state-of-the-art ordering (MAX PTS MIN

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

ERR). The latter figure presents a slight distortion and less points. Moreover, the straight lines of the reconstructed corridor are less defined than in the case of MIN DIST MIN ERR.

Table 20. Reconstruction results from the *fountain-R25* data set. The top part shows a matching sample of two images using CCMM. The bottom part depicts reconstruction results using the minimum distance ordering (left) and the ordering based on maximum common points with the pair of frames with minimum error (right).

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Figure 31 Reduced Reconstruction

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Figure 32 Teacup reconstruction

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Figure 33 Vuelo6 reconstruction

Figure 34 Fountain-R25 Dataset

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Figure 35 Merton College 1 reconstruction

Figure 36 Model House reconstruction

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

6 THE DATA SETS

In this work we have used a large image benchmark to evaluate and validate the proposed algorithms in different situations and environments. Bearing this in mind, we have complemented well known public benchmarks, such as EPFL [50], Oxford [51] or the KITTI dataset [52] with images from other sources to generate a test body with enough variability.

There data sets include structured environments, in which several areas on the image are very similar (e.g. bricks walls, offices, windows etc.). Images with varying depth ranges are also reported to evaluate the robustness of the algorithm w.r.t. projective deformations. Another factor considered is the percentage of overlap in the images to match; examples of both low and high overlap are included. Finally we have used a dataset of images captured with 185° fish-eye optics to analyze performance in a scenario where projective geometry is not valid.

Table 21 summarizes the properties of the datasets used in the experiments. Header n refers to the number of images in each data set, *Structured* refers to scenes with repetitive elements (bricks, windows, etc.) and *Flat* is set to *yes* if the scene features are mainly over a single plane. This is a problematic configuration that commonly appears on aerial imagery. The *Distortion* column categorizes image distortion in very low, low, medium, high and very high.

Finally, column *Overlap* refers to the amount of overlapping motives between frames.

Table 21 Properties of the data sets.

Dataset	N	Structured	Flat	Distortion	Overlap
<i>Heligrande</i>	<i>100</i>	<i>No</i>	<i>Yes</i>	<i>Low</i>	<i>Very High</i>
<i>Reduced</i>	<i>11</i>	<i>No</i>	<i>Yes</i>	<i>Low</i>	<i>Very low</i>
<i>Ualberta</i>	<i>100</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>Medium</i>
<i>Kapel</i>	<i>77</i>	<i>Yes</i>	<i>No</i>	<i>High</i>	<i>Low</i>
<i>Dino</i>	<i>37</i>	<i>No</i>	<i>No</i>	<i>Medium</i>	<i>Medium</i>
<i>Skycam</i>	<i>85</i>	<i>No</i>	<i>No</i>	<i>Very high</i>	<i>Low</i>
<i>castle-P19</i>	<i>19</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>castle-p30</i>	<i>30</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>entry-P10</i>	<i>10</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>fountain_P11</i>	<i>11</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>Herz_Jesu_P25</i>	<i>25</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>Herz_Jesu_P8</i>	<i>8</i>	<i>Yes</i>	<i>No</i>	<i>Very Low</i>	<i>High</i>
<i>desktop</i>	<i>9</i>	<i>Yes</i>	<i>Yes</i>	<i>Low</i>	<i>High</i>
<i>teacup</i>	<i>8</i>	<i>Yes</i>	<i>Yes</i>	<i>Low</i>	<i>High</i>
<i>Herzjesu-R23</i>	<i>23</i>	<i>Yes</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>dolmen</i>	<i>8</i>	<i>Yes</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>albert-laser-vision</i>	<i>73</i>	<i>Yes</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>Castle-R20</i>	<i>20</i>	<i>Yes</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>Dinosaur-Oxford</i>	<i>37</i>	<i>No</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>dinoSparseRing</i>	<i>16</i>	<i>No</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>templeSparseRing</i>	<i>16</i>	<i>No</i>	<i>No</i>	<i>Low</i>	<i>High</i>
<i>Fountain-R25</i>	<i>25</i>	<i>Yes</i>	<i>No</i>	<i>Low</i>	<i>High</i>

Ualberta [53] images were recorded at the Computing Science Centre of the University of Alberta. They comprise a loop around the third floor of the building. Images were captured with a Dragonfly IEEE1394 digital camera from Point Grey Research, placed in a iRobot Magellan Pro robot. An image was taken after an approximate translation of 15 cm or a 5 degree rotation, whichever came first.

EPFL multi-view-evaluation benchmark is a public image dataset made by Christoph Stencha [50] and contains *Fountain-R25*, *Herz-Jesu-R23*, *castle-R20*, *Fountain-P11*, *Herz-Jesu-P8*, *entry-P10*, *castle-p19*, *Herz-Jesu-P25*, *Castle-P30* among others. Datasets with name suffixed by RXX are raw images with a slight distortion. Those with suffixes PXX image datasets that have been modified to present very low distortion. These datasets motives are historic building and circular or semicircular trajectories. They have good overlap but they are strongly structured and motives repeat into patterns which make matching difficult with only visual descriptors alone.

Oxford computer vision group benchmark [51] is composed by data sets: *aerial_view_1*, *dinosaur*, *merton_college*, *model_house*, *wadham-college*, *aerial_view_2*, *kapel*, *corridor*, *library* and *valbonne_church*. This benchmark is very complete in the different motives and trajectories.

DinoSparseRing and *templeSparseRing* are public datasets from [54]. They images are taken using a robotized gantry to obtain exact camera positions.

Heligrande images were shot from a manned helicopter and used for inspection of civil infrastructures. Images are frames from video recordings.

Reduced dataset was taken by an UAV surveying a quarry to estimate a terrain elevation model.

Skycam images were taken from a meteorological station equipped with a zenithal 185° fish-eye camera and are currently being used for cloud observation and weather forecast.

Fpv_1 images were taken from video recordings used for promotional purposes. They have strong fisheye distortions and structured urban scene motives.

Table 22. Image characteristics of the data sets

<i>Dataset name</i>	<i># img</i>	<i>Resolution (pixel)</i>	<i>focal (mm)</i>	<i>image type</i>
<i>Reduced</i>	11	1200 × 900	40mm	<i>Aerial</i>
<i>Heligrande</i>	100	720 × 576	35mm	<i>Aerial</i>
<i>Ualberta</i>	100	640 × 480	35mm	<i>Robot</i>
<i>Skycam</i>	85	1280 × 960	185°	<i>Panoramic</i>
<i>castle-P19</i>	19	3072 × 2048	20mm	<i>Ground</i>
<i>castle-p30</i>	30	3072 × 2048	20mm	<i>Ground</i>
<i>entry-P10</i>	10	3072 × 2048	20mm	<i>Ground</i>
<i>fountain_P11</i>	11	3072 × 2048	20mm	<i>Ground</i>
<i>Herz_Jesu_P25</i>	25	3072 × 2048	20mm	<i>Ground</i>
<i>Herz_Jesu_P8</i>	8	3072 × 2048	20mm	<i>Ground</i>
<i>Desktop</i>	9	3200 × 2368	30mm	<i>small scene</i>
<i>Teacup</i>	8	3200 × 2368	30mm	<i>small scene</i>
<i>Herzjesu-R23</i>	23	3072 × 2048	20mm	<i>Ground</i>
<i>Dolmen</i>	8	3200 × 2368	30mm	<i>Ground</i>
<i>albert-laser-vision</i>	73	320 × 240	30mm	<i>Robot</i>
<i>Castle-R20</i>	20	3072 × 2048	20mm	<i>Ground</i>
<i>Dinosaur-Oxford</i>	37	720 × 576	N/A	<i>Object</i>
<i>dinoSparseRing</i>	16	640 × 480	N/A	<i>Object</i>
<i>templeSparseRing</i>	16	640 × 480	N/A	<i>Object</i>
<i>Fountain-R25</i>	25	3200 × 2368	30mm	<i>Ground</i>
<i>Kapel</i>	77	786 × 512	N/A	<i>Ground</i>
<i>Kitty</i>	4541	1241 × 376	N/A	<i>Car Nav</i>
<i>Characters</i>	35	640 × 480	N/A	<i>Synthetic</i>
<i>Fpv-1</i>	108	1920 × 1080	N/A	<i>Aerial/fisheye</i>

Table 22 shows the number of frames that compose each dataset and also its resolution and focal length. Column *image type* tries to describe the motives shown on the scene. Table 23, Table 24, Table 25, Table 26 and Table 27 contain sample images from each data set.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Table 23. Aerial scenes

<i>Reduced</i>	<i>Airview</i>

<i>Fpv_1</i>	<i>Heligrande</i>

<i>Vuelo6</i>	<i>Aerial_view_1</i>

Table 24. Outdoor buildings

---	--

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>Herz_Jesu_XX</i>	<i>Castle_XX</i>

<i>Entry_XX</i>	<i>Fountain_XX</i>

<i>Kapel</i>	<i>Library</i>

<i>Merton_college_X</i>	<i>Wadham college</i>

<i>valbonne_church</i>	

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

Table 25. Indoor

<i>Corridor</i>	<i>ualberta</i>

<i>albert-B-laser-vision-dataset</i>	

Table 26 Objects

<i>Desktop</i>	<i>Teacup</i>

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>Dino</i>	<i>DinoSparseRing</i>

<i>Piedra</i>	<i>TempleSparseRing</i>

Table 27. Other datasets

<i>Kitti</i>

<i>Characters</i>	<i>Skycam</i>

7 CONCLUSIONS

This thesis describes CCMM, a new algorithm designed to improve the feature association step between image pairs with low deformations, in the automated image processing chain. Unlike state-of-the-art procedures such as RANSAC, which only consider the best match for each keypoint (according to SIFT/SURF descriptors), CCMM builds a multi-hypotheses graph model considering the k best candidates and reduces joint compatibility to a maximum clique search in the graph.

Moreover, the work shows that the size of the graph grows linearly both with the number of keypoints (in the search image) and the number of candidates (in the reference image), whereas the underlying relational matching model is quadratic in the general case. This makes it suitable for a broad range of real problems such as 3D reconstruction, Visual Odometry, Image Mosaicing and others.

The thesis is validated using extensive tests against a broad set of image benchmarks which challenge the underlying graph model restrictions (i.e. high deformations, repetitive patterns or low overlap). Results are, in general, more robust than previous methods, while keeping time complexity under reasonable bounds for real applications.

This research also presents BBMCW, a new efficient exact maximum clique algorithm tailored for large sparse graphs but not massive. This type of graphs appears in practice when some form of locality can be exploited to reduce the scale of the problem. One such case are the correspondence graphs derived from image association problems, target of this work.

BBMCW is based in prior bit-parallel algorithm BBMC [21], and its improved variant BBMCi [33]. It achieves its efficiency mainly from a new sparse bit encoding which effectively reduces the number of empty bitblock operations.

The proposed bit encoding uses two new pointers, denoted as *sentinels*, which *watch* two particular bitblocks of the bitstring. This work shows that the two sentinels effectively circumscribe set-operations to a reduced number of vertices, thus lowering the number of spurious bitmasks. Moreover, empty-set detection is reduced to a constant time operation.

In addition to the above results, the algorithms have been applied to the complete reconstruction chain using bundle adjustment. This research also contributes to this step by considering graph-based techniques to initialize the optimization algorithm. Note that initialization can be extremely important in bundle adjustment: by starting from a good stable initial solution, the adjustment can avoid a number of pitfalls. Results show that establishing an initial scene by selecting frames according to a minimum distance ordering along an error graph, starting from the frame of minimum eccentricity, obtains better scene reconstructions than previously known initialization procedures.

FUTURE WORK

One problem of the CCMM algorithm is that it relies on an arbitrary threshold for rejecting non-compatible matches. Ongoing work is concerned with considering other similarity metrics between feature pairs, besides pixel distance, to build the association graph. Examples of these are non-linear thresholds or reducing the compatibility check to neighbor matches only.

To scale our techniques to large and very large scene reconstructions with big image collections, improvements must be found to keep the computational

effort within reasonable bounds. A simple implementation of the Bag of Words (BoW) technique has been used in this research using correlation as distance metric between BoW pairs. Another open line of research is the study of alternative distance metrics for the case of images that present a low overlap.

Worth considering also are more complex compatibility constraints for CCMM, such as, for example, matching pairs or triplets of features instead of single features.

8 BIBLIOGRAPHY

- [1] R. Gupta and R. I. Hartley., "Linear pushbroom cameras.," *IEEE transactions on pattern analysis and machine intelligence*, pp. 963-975, 1997.
- [2] J. Kannala and S. S. Brand, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses.," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1335-1340, 2006.
- [3] C. Harris and M. Stephens, "A combined edge detector," in *Vision Conference*, Alvey, 1988.
- [4] T. C. Shi J, "Good Featrues to Track," in *Computer Vision and Pattern Recognition*, 1994.
- [5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, pp. 60(2), 91-110., 2004.
- [6] L. Van Gool, H. Bay, A. Ess and T. Tuytelaars, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, pp. 110(3), 346-359., 2008.

- [7] P. Alcantarilla, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Match. Intell.*, pp. 34(7), 1281-1298, 2011.
- [8] J. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces"," *Conference on Computer Vision and Pattern Recognition, Puerto Rico*, p. 1000–1006, 1997.
- [9] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP*, pp. 331-340, 2009.
- [10] J. Ma and e. al., "Robust point matching via vector field consensus.," *IEEE Transactions on Image Processing*, pp. 1706-1721, 2014.
- [11] M. A. Fischler and R. C. Bolles., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.," *Communications of the ACM*, pp. 381-395, 1981.
- [12] R. Hartley and A. Zisserman., *Multiple view geometry in computer vision.*, Cambridge university press, 2003.
- [13] R. Karp., *Reducibility among Combinatorial Problems*, Editors, R.E. Miller, J. W. Thatcher, New York, 1972.
- [14] S. Butenko and W. Wilhelm, "Clique-detection models in computational biochemistry and genomics.," *European Journal of Operational Research*, pp. 173, 1-17, 2006.
- [15] K. Hotta, E. Tomita and H. Takahashi, "A view invariant human FACE detection method based on maximum cliques.," *Trans. IPSJ, 44, SIG14 (TOM9)*, pp. 57-70, 2003.
- [16] P. San Segundo, D. Rodriguez-Losada, F. Matia and R. Galan, "Fast exact feature based data correspondence search with an efficient bit-parallel MCP solver.," *Applied Intelligence*, pp. 311-329, 2010.
- [17] P. San Segundo and D. Rodriguez-Losada, "Robust Global Feature Based

Data Association With a Sparse Bit Optimized Maximum Clique Algorithm.," *IEEE Transactions on Robotics*, 29:5, no. 29:5, pp. 1332-1339, 2013.

- [18] S. T. Tomita E., "An efficient branch and bound algorithm for finding a maximum clique.," *Discrete Mathematics and Theoretical Computer Science. LNCS*, pp. 278-289, 2003.
- [19] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi and M. Wakatsuki, "A Simple and Faster Branch-and-Bound Algorithm for Finding a Maximum Clique," *In Lecture Notes in Computer Science*, pp. 191-203, 2010.
- [20] J. Konc and D. Janečič, "An improved branch and bound algorithm for the maximum clique problem," . *MATCH Commun. Math. Comput. Chem.*, pp. 569-590, 2007.
- [21] P. San Segundo, D. Rodriguez-Losada and A. Jimenez, "An exact bit-parallel algorithm for the maximum clique problem.," *Computers & Operations Research*, pp. 571-581, 2011.
- [22] P. San Segundo, F. Matia, D. Rodriguez-Losada and M. Hernando, "An improved bit parallel exact maximum clique algorithm.," *Optimization Letters.*, pp. 467-479, 2011.
- [23] C. Q. Z. Li, "An Efficient Branch-and-Bound Algorithm based on MaxSAT for the Maximum Clique Problem.," *AAAI*, p. 128–133, 2010.
- [24] C. Li, Z. Fang and K. Xu, "Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem.," *Tools with Artificial Intelligence (ICTAI)*, pp. 939-946., 2013.
- [25] M. Batsyn, .. B. Goldengorin, M. E. and P. Pardalos, "Improvements to MCS algorithm for the maximum clique problem.," *Journal of Combinatorial Optimization*, pp. 397-416, 2014.
- [26] P. San Segundo and C. Tapia, "Relaxed approximate coloring in exact maximum clique search," *Computers & Operations Research*, pp. 185-192,

2014.

- [27] P. Prosser, "Exact Algorithms for Maximum Clique: A Computational Study.," *Algorithms*, pp. 545-587, 2012.
- [28] P. San Segundo, A. Nikolaev and M. Batsyn, "Infra-chromatic bound for exact maximum clique search.," *Computers & Operations Research*, pp. 293-303, 2015.
- [29] P. San Segundo, J. Artieda, R. Leon and C. Tapia, "An Enhanced Infra-Chromatic Bound for the Maximum Clique Problem.," in *International Workshop on Machine Learning, Optimization and Big Data.*, 2016.
- [30] D. Andrade, M. Resende and R. Werneck, "Fast local search for the maximum independent set problem.," *J. Heuristics*, pp. 525-547, 2012.
- [31] Q. Wu and J. Hao, "An adaptive multistart tabu search approach to solve the maximum clique problem.," *Journal of Combinatorial Optimization*, pp. 86-108, 2013.
- [32] D. Matula and L. Beck, "Smallest-Last ordering and clustering and graph coloring algorithms.," *J. Assoc. Comput. Mach.*, p. 417-427, 1983.
- [33] P. San Segundo, F. Matia, D. Rodriguez-Losada and M. Hernando, "An improved bit parallel exact maximum clique algorithm," *Optimization Letters*, vol. 7(3), pp. 467-479, 2013.
- [34] P. San Segundo, A. Nikolaev, M. Batsyn and P. M. Pardalos, "Improved Infra-Chromatic Bound for Exact Maximum Clique Search.," *Informatica*, pp. 463-487, 2016.
- [35] P. San Segundo, A. Lopez and P. M. Pardalos, "A new exact maximum clique algorithm for large and massive sparse graphs.," *Computers & Operations Research*, no. 66, pp. 81-94, 2016.
- [36] L. Chu-Min, F. Zhiwen and X. Ke, "Combining MaxSAT Reasoning and Incremental Upper Bound for the Maximum Clique Problem.," *Proc. Tools with Artificial Intelligence (ICTAI)*, pp. 939-946, 2013.

- [37] H. Strasdat, J. M. M. Montiel and A. J. Davison, "Real-time monocular SLAM: Why filter?," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
- [38] E. Eade and T. Drummond, "Scalable monocular SLAM," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.
- [39] M. I. Lourakis, "A brief description of the Levenberg-Marquardt algorithm implemented by levmar," *Foundation of Research and Technology* , 2005.
- [40] M. Lourakis and A. .. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," FORTH, Crete, 2004.
- [41] H. Strasdat, J. M. M. Montiel and A. J. Davison, "Scale drift-aware large scale monocular SLAM," *Robotics: Science and Systems VI*, 2010.
- [42] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit and P. Fua, ""View-based maps.,"" *The International Journal of Robotics Research* 29,, no. 8, pp. 941-957, 2010.
- [43] P. H. Torr and A. .. Zisserman, "Robust parameterization and computation of the trifocal tensor.," *Image and Vision Computing* 15,, no. 8, pp. 591-605, 1997.
- [44] D. Nistér, "An efficient solution to the five-point relative pose problem.," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, p. 756–770, 2004.
- [45] R. I. a. P. S. Hartley, "Triangulation.," *Computer vision and image understanding* , pp. 146-157, 1997.
- [46] V. Lepetit, F. Moreno-Noguer and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem.," *International journal of computer vision*, p. 155, 2009.

- [47] N. Snavely, S. M. Seitz and R. Szeliski, "Modeling the world from inter-net photo collections.," *International Journal of Computer Vision*, pp. 189-210, 2008.
- [48] M. Brown and D. G. Lowe., "Unsupervised 3D object recognition and reconstruction in unordered datasets.," *3-D Digital Imaging and Modeling*, 2005.
- [49] W. Cook, 2005. [Online]. Available: <http://www.tsp.gatech.edu/concorde.html>.
- [50] C. Strecha, W. von Hansen, L. Van Gool, P. Fua and U. Thoennessen, "On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery," *CVPR*, 2008.
- [51] O. V. G. Group, "Multi-view and Oxford Colleges building reconstruction," August 2009. [Online]. Available: <http://www.robots.ox.ac.uk/~vgg/>.
- [52] A. e. a. Geiger, "Vision meets robotics: The KITTI dataset.," *The International Journal of Robotics Research*, pp. 1231-1237, 2013.
- [53] J. Klippenstein. [Online]. Available: <http://cres.usc.edu/radishrepository/view-one.php?name=ualberta-csc-flr3-vision>.
- [54] S. Seitz, B. Curless, J. Diebel, D. Scharstein and R. Szeliski, 2005. [Online]. Available: <http://grail.cs.washington.edu/projects/mview/>.

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

9 APPENDICES

APPENDIX I INITIALIZATION ALGORITHMS EVALUATION

Table 28 Maximum Points Starting on a minimum eccentricity pair. #pnts is the number of 3d points on the scene. #frms the number of frames inserted in the solution. #proj is the total number of projections. RMSE is the root-mean-square error of the projected points.

	<i>MAX PTS MIN ECC</i>			
	#pts	#frms	#proj	RMSE
<i>reduced</i>	1	2	2	-0.02
<i>aerial_view_1</i>	133	6	334	0.26
<i>aerial_view_2</i>	98	6	204	0.71
<i>dinosaur</i>	11	2	22	0.00
<i>merton_college1</i>	220	3	467	0.10
<i>model_house</i>	1	2	2	0.01
<i>wadhamcollege</i>	417	5	930	0.10
<i>kapel</i>	1	2	2	-0.05
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	297	11	676	0.15
<i>library</i>	22	2	44	0.21
<i>merton_college3</i>	92	3	205	0.18
<i>castle_p19</i>	6	2	12	0.00
<i>castle_p30</i>	178	7	439	0.33
<i>entry-p10</i>	114	3	253	0.05
<i>fountain-p11</i>	0	0	0	0.00
<i>herz-jeus-p25</i>	0	0	0	0.00
<i>herz-jesu-p8</i>	0	0	0	0.00
<i>desktop</i>	132	4	269	0.12
<i>teacup</i>	281	6	701	0.32
<i>herzjesu-r23</i>	3	2	6	0.00

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>dolmen</i>	0	0	0	0.00
<i>fpv1_r</i>	4416	108	11758	0.31
<i>airview</i>	892	34	2643	0.20
<i>albert-laser-vision</i>	1368	72	6090	0.10
<i>castle-r20</i>	39	4	97	0.10
<i>dinoSparseRing</i>	22	2	44	0.12
<i>templeSparseRing</i>	321	4	656	0.03
<i>fountain-r25</i>	720	21	1734	0.16
<i>piedra</i>	0	0	0	0.00

Table 29 Maximum Common Points starting from the minimum error pair

	<i>MAX PTS MIN ERR</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	151	7	330	106,108.00
<i>aerial_view_1</i>	111	5	263	0.12
<i>aerial_view_2</i>	6	2	12	0.00
<i>dinosaur</i>	889	36	2011	299.41
<i>merton_college1</i>	220	3	467	0.10
<i>model_house</i>	121	7	290	0.38
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	1	2	2	0.00
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	575	10	1838	0.35
<i>library</i>	34	2	68	0.21
<i>merton_college3</i>	12	2	24	0.09
<i>castle_p19</i>	180	6	397	0.08

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>castle_p30</i>	2	2	4	0.00
<i>entry-p10</i>	56	2	112	0.04
<i>fountain-p11</i>	92	6	225	0.33
<i>herz-jeus-p25</i>	3	2	6	0.00
<i>herz-jesu-p8</i>	0	0	0	0.00
<i>desktop</i>	164	3	330	0.13
<i>teacup</i>	243	5	551	0.32
<i>herzjesu-r23</i>	9	2	18	0.00
<i>dolmen</i>	0	0	0	0.00
<i>fpv1_r</i>	4048	107	11313	0.32
<i>airview</i>	1021	33	2762	0.28
<i>albert-laser-vision</i>	1299	70	7197	1.74
<i>castle-r20</i>	201	8	468	0.47
<i>dinoSparseRing</i>	0	0	0	0.00
<i>templeSparseRing</i>	28	2	56	0.12
<i>fountain-r25</i>	507	19	1256	0.31
<i>piedra</i>	0	0	0	0.00

Table 30 Minimum Spanning Tree traversed on a depth first order

	<i>Depth First</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	393	7	882	0.03
<i>aerial_view_1</i>	109	5	232	0.20
<i>aerial_view_2</i>	10	2	20	0.00
<i>dinosaur</i>	351	10	856	0.31
<i>merton_college1</i>	154	3	327	0.10

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>model_house</i>	0	0	0	0.00
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	219	9	466	33.60
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	485	11	1381	0.19
<i>library</i>	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09
<i>castle_p19</i>	6	2	12	0.00
<i>castle_p30</i>	216	9	491	0.17
<i>entry-p10</i>	173	3	356	0.29
<i>fountain-p11</i>	62	4	137	0.25
<i>herz-jeus-p25</i>	88	6	191	0.27
<i>herz-jesu-p8</i>	54	4	115	0.14
<i>desktop</i>	13	2	26	0.03
<i>teacup</i>	393	6	964	0.07
<i>herzjesu-r23</i>	10	2	20	0.00
<i>dolmen</i>	3	2	6	0.00
<i>fpv1_r</i>	3844	108	11606	0.40
<i>airview</i>	612	20	2002	0.40
<i>albert-laser-vision</i>	1454	73	7622	0.12
<i>castle-r20</i>	29	3	65	0.15
<i>dinoSparseRing</i>	22	2	44	0.12
<i>templeSparseRing</i>	330	2	660	0.00
<i>fountain-r25</i>	604	20	1561	0.43
<i>piedra</i>	0	0	0	0.00

Table 31 Minimum Distance calculated along the full graph starting on the minimum eccentricity pair

	<i>MIN DIST</i>			
	#pts	#frms	#proj	RMSE
<i>reduced</i>	42	4	92	0.18
<i>aerial_view_1</i>	108	5	229	0.20
<i>aerial_view_2</i>	99	6	233	0.76
<i>dinosaur</i>	566	12	1383	0.49
<i>merton_college1</i>	154	3	327	0.10
<i>model_house</i>	94	5	216	0.21
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	14	2	28	0.01
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	592	11	1845	0.31
<i>library</i>	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09
<i>castle_p19</i>	29	2	58	0.19
<i>castle_p30</i>	224	9	531	53.49
<i>entry-p10</i>	173	3	356	0.29
<i>fountain-p11</i>	6	2	12	0.00
<i>herz-jeus-p25</i>	38	2	76	0.04
<i>herz-jesu-p8</i>	12	2	24	0.16
<i>desktop</i>	240	4	506	0.14
<i>teacup</i>	152	3	322	0.50
<i>herzjesu-r23</i>	71	10	158	0.12
<i>dolmen</i>	180	6	375	0.10
<i>fpv1_r</i>	4002	108	10666	0.37

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>airview</i>	896	31	2872	0.38
<i>albert-laser-vision</i>	1821	72	7968	0.18
<i>castle-r20</i>	34	4	80	0.27
<i>dinoSparseRing</i>	167	6	367	0.27
<i>templeSparseRing</i>	380	4	776	0.02
<i>fountain-r25</i>	674	24	1738	0.45
<i>piedra</i>	0	0	0	0.00

Table 32 Minimum distance along the full graph starting on the minimum error pair.

	<i>MIN DIST MIN ERR</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	328	11	750	60.86
<i>aerial_view_1</i>	57	6	119	0.21
<i>aerial_view_2</i>	65	5	128	0.20
<i>dinosaur</i>	379	13	904	0.21
<i>merton_college1</i>	158	3	331	0.09
<i>model_house</i>	43	7	88	0.06
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	140	4	278	0.02
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	317	11	770	0.10
<i>library</i>	5	2	10	0.00
<i>merton_college3</i>	144	3	311	0.12
<i>castle_p19</i>	51	4	112	0.22
<i>castle_p30</i>	370	14	861	0.32
<i>entry-p10</i>	188	3	389	0.48

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>fountain-p11</i>	85	9	177	0.18
<i>herz-jeus-p25</i>	122	13	266	0.42
<i>herz-jesu-p8</i>	26	3	55	0.24
<i>desktop</i>	21	2	42	0.34
<i>teacup</i>	92	4	194	0.07
<i>herzjesu-r23</i>	110	6	216	0.04
<i>dolmen</i>	241	2	482	0.03
<i>fpv1_r</i>	3754	107	9384	0.45
<i>airview</i>	869	29	2139	0.20
<i>albert-laser-vision</i>	1657	59	5105	0.15
<i>castle-r20</i>	98	7	226	0.43
<i>dinoSparseRing</i>	200	8	430	0.33
<i>templeSparseRing</i>	338	2	668	0.00
<i>fountain-r25</i>	989	24	2317	0.17
<i>piedra</i>	0	0	0	0.00

Table 33 Minimum Distance along a minimum spanning tree

	<i>MIN DIST MST</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	422	10	991	0.05
<i>aerial_view_1</i>	108	5	229	0.20
<i>aerial_view_2</i>	10	2	20	0.00
<i>dinosaur</i>	368	11	893	0.36
<i>merton_college1</i>	154	3	327	0.10
<i>model_house</i>	0	0	0	0.00
<i>wadhamcollege</i>	0	0	0	0.00

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>kapel</i>	0	0	0	0.00
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	553	11	1667	0.16
<i>library</i>	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09
<i>castle_p19</i>	6	2	12	0.00
<i>castle_p30</i>	96	6	220	0.22
<i>entry-p10</i>	173	3	356	0.29
<i>fountain-p11</i>	107	6	243	0.24
<i>herz-jeus-p25</i>	20	3	43	0.63
<i>herz-jesu-p8</i>	43	4	93	0.53
<i>desktop</i>	13	2	26	0.03
<i>teacup</i>	439	6	1074	0.06
<i>herzjesu-r23</i>	10	2	20	0.00
<i>dolmen</i>	3	2	6	0.00
<i>fpv1_r</i>	3594	108	9849	0.33
<i>airview</i>	545	30	1527	0.20
<i>albert-laser-vision</i>	0	0	0	0.00
<i>castle-r20</i>	13	2	26	0.01
<i>dinoSparseRing</i>	22	2	44	0.12
<i>templeSparseRing</i>	334	3	676	0.03
<i>fountain-r25</i>	973	21	2302	0.16
<i>piedra</i>	0	0	0	0.00
<i>vuelo6</i>	1260	37	3338	0.42

Table 34 Minimum distance along the full graph starting from the minimum eccentricity point calculated from the minim spanning tree.

	<i>MIN DIST CENTER MST</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	393	10	873	0.28
<i>aerial_view_1</i>	103	5	230	0.27
<i>aerial_view_2</i>	10	2	20	0.00
<i>dinosaur</i>	368	11	893	0.36
<i>merton_college1</i>	154	3	327	0.10
<i>model_house</i>	0	0	0	0.00
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	145	8	346	108.90
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	618	11	1831	0.16
<i>library</i>	5	2	10	0.00
<i>merton_college3</i>	101	3	213	0.09
<i>castle_p19</i>	6	2	12	0.00
<i>castle_p30</i>	122	8	284	0.22
<i>entry-p10</i>	173	3	356	0.29
<i>fountain-p11</i>	68	5	158	0.30
<i>herz-jeus-p25</i>	22	5	50	0.11
<i>herz-jesu-p8</i>	43	4	93	0.53
<i>desktop</i>	13	2	26	0.03
<i>teacup</i>	373	6	855	0.05
<i>herzjesu-r23</i>	10	2	20	0.00
<i>dolmen</i>	3	2	6	0.00
<i>fpv1_r</i>	3845	108	11065	0.36

<i>airview</i>	0	0	0	0.00
<i>albert-laser-vision</i>	1403	65	7985	11.59
<i>castle-r20</i>	13	2	26	0.01
<i>dinoSparseRing</i>	22	2	44	0.12
<i>templeSparseRing</i>	334	3	676	0.03
<i>fountain-r25</i>	443	22	1006	0.37
<i>piedra</i>	0	0	0	0.00

Table 35 Sequential algorithm using filenames.

	<i>sequential</i>			
	#Pts	#frms	#proj	RMSE
<i>reduced</i>	11	2	22	0.00
<i>aerial_view_1</i>	1	2	2	0.01
<i>aerial_view_2</i>	102	6	238	0.89
<i>dinosaur</i>	11	2	22	0.00
<i>merton_college1</i>	187	3	389	0.10
<i>model_house</i>	191	7	469	0.16
<i>wadhamcollege</i>	0	0	0	0.00
<i>kapel</i>	51	2	102	0.02
<i>merton_college2</i>	0	0	0	0.00
<i>corridor</i>	458	11	1304	0.33
<i>library</i>	0	0	0	0.00
<i>merton_college3</i>	92	3	205	0.18
<i>castle_p19</i>	42	4	94	0.25
<i>castle_p30</i>	0	0	0	0.00
<i>entry-p10</i>	173	3	356	0.29

GRAPH SEARCH AND ITS APPLICATION TO IMAGE MATCHING

– Jorge Artieda– May 2017

<i>fountain-p11</i>	177	9	393	0.54
<i>herz-jeus-p25</i>	1	2	2	0.02
<i>herz-jesu-p8</i>	12	2	24	0.16
<i>desktop</i>	1	2	2	0.01
<i>teacup</i>	293	6	684	0.06
<i>herzjesu-r23</i>	125	7	272	0.04
<i>dolmen</i>	0	0	0	0.00
<i>fpv1_r</i>	3991	108	10711	0.34
<i>airview</i>	1069	37	3108	0.26
<i>albert-laser-vision</i>	1282	73	5982	0.12
<i>castle-r20</i>	95	6	215	0.34
<i>dinoSparseRing</i>	22	2	44	0.12
<i>templeSparseRing</i>	349	3	706	0.01
<i>fountain-r25</i>	594	25	1409	0.38
<i>piedra</i>	0	0	0	0.00

APPENDIX II BBMCW EVALUATION

Table 36. Time performance over a number of real correspondence graphs taken from six image data sets. Header *p* shows average density. Times are measured in seconds. Columns containing *imp.* show the ratio between the algorithm in the corresponding header and BBMCW.

						IncMaxCLQ	BBMCI	BBMCW		
name	V	E	p	ω	ω_o	time (s)	time(s)	time(s)	IncMaxCLQ imp.	BBMCI imp.
herzjesu_b_1	9675	8873275	0.19	760	719	218	58.6	16.9	12.9	3.47
herzjesu_b_2	9541	8578844	0.19	872	820	319	4.42	1.15	278	3.85
herzjesu_b_3	9436	6335802	0.14	505	472	57.0	2.18	0.557	102	3.92
herzjesu_b_4	9730	7960327	0.17	762	725	191	7.00	1.72	111	4.08
herzjesu_b_5	8745	7910168	0.21	1174	1162	267	2.50	0.629	424	3.97
herzjesu_b_6	9784	7713350	0.16	649	611	108	2.37	0.575	187	4.12
herzjesu_b_7	10649	9002418	0.16	659	578	100	38.6	10.2	9.81	3.78
fpv1_1	10206	8866422	0.17	1179	1167	574	3.69	0.925	620	3.99
fpv1_2	10104	8759028	0.17	1278	1254	644	7.49	1.91	337	3.92
fpv1_3	10212	8599757	0.16	1283	1270	552	4.18	1.05	525	3.98
fpv1_4	9946	8026449	0.16	1237	1226	519	3.75	0.955	544	3.92
fpv1_5	10401	8714730	0.16	1314	1300	760	5.31	1.35	562	3.93
fpv1_6	10209	8799150	0.17	1285	1277	554	3.41	0.862	643	3.95
fpv1_7	10190	8524605	0.16	1238	1221	352	5.08	1.28	275	3.96
fpv1_8	10128	8465909	0.17	1245	1223	704	5.76	1.45	487	3.98
fpv1_9	10146	8291706	0.16	1204	1185	457	3.93	0.991	461	3.97
fpv1_10	9638	7345533	0.16	1164	1155	390	3.05	0.759	514	4.01
fountain_1	3958	1852762	0.24	253	242	20.8	0.268	0.114	182	2.35
fountain_2	3595	1904085	0.29	393	382	7.18	0.156	0.081	88.7	1.93
castle_1	2701	1086305	0.30	308	295	15.4	0.204	0.103	149	1.98
castle_2	2643	1111405	0.32	116	91	8.39	0.262	0.139	60.3	1.88
castle_3	2747	1205769	0.32	52	34	12.5	30.2	16.1	0.777	1.87

castle_4	3959	2388974	0.30	30	22	10.5	2.20	1.60	6.52	1.37
castle_5	3625	1749309	0.27	212	154	7.91	0.191	0.089	88.9	2.15
castle_6	3633	1714761	0.26	116	54	9.52	34.3	11.8	0.804	2.89
castle_7	3871	1288845	0.17	41	34	6.56	0.793	0.622	10.5	1.27
heliGrande_1	3234	1476360	0.28	439	439	7.47	0.090	0.047	159	1.91
heliGrande_2	3275	1472250	0.27	415	414	12.8	0.095	0.052	245	1.83
heliGrande_3	3175	1419946	0.28	422	421	7.66	0.089	0.054	142	1.65
heliGrande_4	3224	1494127	0.29	447	445	8.04	0.109	0.058	139	1.88
heliGrande_5	3220	1438513	0.28	416	416	10.7	0.084	0.046	233	1.83
castle_b_1	4943	3562268	0.29	321	303	17.0	0.502	0.209	81.5	2.40
castle_b_2	4539	2636689	0.26	130	63	6.55	6.55	1.66	3.94	3.94
castle_b_3	4852	2247405	0.19	46	35	7.07	3.88	1.18	6.01	3.29
castle_b_4	4601	2089877	0.20	34	25	5.18	16.2	11.2	0.461	1.44