Are the Latest SLAM Pipelines Improving the Pose Accuracy Remarkably?

José Miguel de Sousa Mota
jose.sousa.mota@ist.utl.pt
Instituto Superior Técnico, Universidade de Lisboa, Portugal
October 2020

Abstract

Simultaneous localization and mapping (SLAM) is a field in constant investigation, due to its diverse applications. The main objective of this work is the development of an implementation of a SLAM method that serves as a comparison and assessment baseline for state of the art, public domain, implementations. This work is concerned with the specific cases where the input data is based just on video cameras, acquiring 2D images, or is based on colour-depth cameras, acquiring 2D images and depth maps. The developed solution is compared with *COLMAP* and VSFM, using the ETH3D and TUM datasets. Results show the proposed implementation and the alternatives have comparable accuracies, with just some advantage for VSFM. This similarity indicates SLAM may have reached a high level of maturity in R&D.

I. Introduction

Computer vision contains a vast variety of applicability, ranging from fully autonomous (unmanned) vehicles to vehicles where computer-vision-based systems support a driver.

At the beginning topographic and engineering technologies were connected, so that the construction of digital elevation models (DEMs) were possible using photogrammetry, differential global positioning station (GPS) and total station 3D positioning data. More recently, laser scanning systems were in the demand of providing an high quality/resolution data in form of 3D point clouds similar to LiDAR. Since both this approach's were very expensive and required a lot of advanced technology. Simultaneous localization and mapping (SLAM) started to become more popular since it was a very low cost alternative that provided high resolution data with reduced user supervision. This technology became possible through advancing of computers, digital cameras and unmanned aerial systems (UAS). The rapid growth and research interests using this technology, further demonstrated the actual potential and the applicability's to it.

SLAM is also used to create orthophotograph mosaics, 3-D point clouds, and digital elevation models. All this can be used in mapping and photogrammetry, to determine tree biomass, analyze geology, detect topographic changes to monitor glacial, fluvial, coastal, hill slopes environments and also in robotics and self-driving vehicles.

Autonomous vehicles typically use computer vision for navigation by producing a map of its environment (SLAM), for detecting obstacles and detecting certain tasks. It is already applicable in Space exploration in autonomous vehicles, for example NASA's Curiosity and CNSA's Yutu-2 rover[5].

Several different softwares were developed in the scope of SLAM, as Visual SfM, COLMAP, Regard3D and OpenSfM. There functional principal starts by extracting features from the set of 2D images, compute the camera pose and assemble the complete 3D scene point cloud. In order to find correspondence between images, features such as points from corners or objects are extracted. Then they are matched with the following image, in order to find the relative camera pose.

A common feature detector is the SURF (speeded-up robust features), which instead of evaluating the gradient histograms, it computes the sums of gradient components and the sums of their absolute values.

scale-invariant feature transform (SIFT) is another feature detector that uses the maxima from the difference-of-Gaussians (DOG) pyramid as features.

SURF proved to be a better extractor since it was much faster in extracting features but had the disadvantage of being less accurate on acquiring their positioning.

After extracting features from an image, they will be matched using an matching algorithms that track features from one image to another. Fischler and Bolles, provided an algorithm using RANSAC to solve the location determination problem (LDP),to determine the 3D position of the extracted features. Then the features positioning are used to reconstruct their 3D positions.

If the camera position is computed each time an new camera is added to the system, the method is called incremental SLAM. The global SLAM, computes all the camera poses at the same time. The out-of-core SLAM, is an intermediate approach where the solution is computed partially globally and partially relative.

From the above reasons and as Matlab is one of the most promising tools being used, the main objective of this thesis was settled to be the development of an alternative software of SLAM using Matlab, in order to compare our results with the most recent SLAM softwares developed. This comparison was made to investigate if the current SLAM softwares have been improving their results.

II. Background and Related Work

Navigation is present in several fields; land, marine, aeronautic, and space. On Aerospace, a great example is the

Global positioning satellites(GPS) that used this concept of monitoring trajectories and estimation of positions of objects.

Furthermore, navigation is a field that focuses on the process of monitoring the movement of an object. When applied to the area of simultaneous localization and mapping (SLAM), this concept remains the same but applied to the estimation of the camera pose along the time. In Aerospace this can be observed on satellite trajectory monitoring [3], GPS modeling for designing aerospace vehicle [4], etc.

In this chapter, is presented some background knowledge related to the construction of our SLAM software. The different image models, some methods to acquire features, different approaches to improve the reconstructed scene, various optimization methods, and state-of-the-art alternative softwares.

A. Projection Model

Computer Vision plays a very important role in our daily day life. On modern times the best way to help the computer to "see", is by providing all necessary tools to full fill the objective that we established. The best way for this to happen is through images interpretation. A single scene image is a rich source of information on the three-dimensional and four-dimensional system. The primary approach to interpreted those images is to handle them in a two-dimensional reference system.

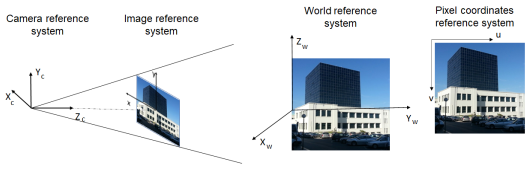

Fig. 1. This image shows us the different reference system. The first is the camera reference system; The second is the 2D image reference system; The third image is the world reference system centered on the north tower of IST; The last image corresponds to the 2D camera reference system in pixel coordinates.

The camera projection model describes the relation between the coordinates in the 3D world reference system and the 2D camera reference system. The camera opening can be described as a point, where no lenses are used to focus light and does not include geometric distortions or blurring of unfocused objects.

The camera projection model can be expressed by different mathematical relations,depending on which reference system is in demand. X, Y, Z are the coordinates in the world coordinate system, x,y,z are the coordinates in the camera reference system.

In order to go from the world reference system to the camera reference system, it is necessary to know the rotation and translation of the camera, in relation to the world reference. The projection matrix give us this relation, which can be written mathematical by:

$$P = [RT] \tag{1}$$

In conclusion, in the case where we have a 3D point(X,Y,Z) and we want to find its projection in pixel coordinates (u,v), we have to multiply the homogeneous coordinates of the 3D point by the matrix P, K.

$$\begin{bmatrix} \frac{u}{w} \\ \frac{v}{w} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-f}{s_x} & 0 & u_0 \\ 0 & \frac{-f}{s_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} [R] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

In the case of back-projection, the process is the opposite approach of the mentioned above. It's necessary to find the position of the 3D point using his 2D correspondence. This process is prevalent in SLAM software since it's necessary to find the 3D coordinates of features extracted from images. When a feature is present in the view of multiple cameras, the process to compute the 3D world coordinates is called triangulation.

Each feature in an image always has a corresponding line in 3D space. Suppose a feature is detected in different images. In that case, those lines will intercept in a common point, being this point the correspondent 3D world point P. Formulating an algebric relation the coordinates of P (3D point) can be computed. From this relation and the epipolar constrain, we obtain the coordinates of P.

B. Epipolar constrain

When a point is present in different cameras views, it is possible to obtain its 3D position by performing a triangulation. The epipolar constrain allows us to express this problem geometrically.

When two cameras view a 3D point(P) from two different perspectives, it is possible to withdraw a geometric relation between the 3D point and its projection onto the 2D images.

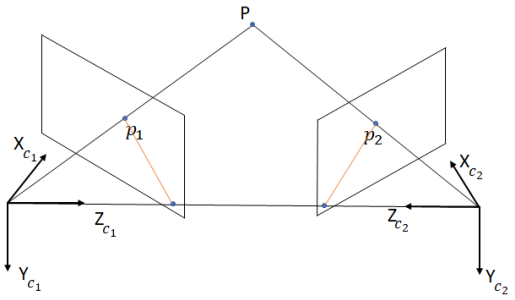

Fig. 2. Projection matrix scenario;C_1 is the camera center of the first camera and C_2 the camera center of camera 2. p_1 and p_2 are the epipolar points; P is the 3D point present in the two camera views.

The resulting equation is the following:

$$\vec{p_1}.(t \times R\vec{p_2}) = 0 \tag{3}$$

Where $\vec{t}$ is the translation vector from the camera C_1 to C_2 and $R_{C_0}^{C_1}$ is the rotation of the camera C_2 in relation to C_1

C. Essential Matrix

Given a set of corresponding image points it is possible to compute an essential matrix. This matrix satisfies the defining

epipolar constraint for all the points. However, if they are subject to noise, it won't be possible to find an essential matrix. The essential matrix can only be used in relation to calibrated cameras since the camera parameters must be known for the the normalization. E is the essential matrix, and $[t]_x$ the skew matrix of the translation t.

$$E = R \times [t]_x \tag{4}$$

D. Feature detectors

In computer vision, the operation of selecting image features according to geographic features such as corners or surfaces present in an image is called feature extraction. This is accomplished in several ways, but the more common methods will be presented here.

A feature is an "interesting" part of an image, and that is used as a starting point for many computer vision algorithms. They can be classified as, edges, corners, blobs and ridges. Edges are features extracted in boundaries between two image regions.

For SLAM, the most crucial characteristic of a feature is its repeatability, because it is necessary to be able to compute the relative pose estimation in between pictures.

Feature detection is usually performed as the first operation on an image. Examines every pixel to see if there is a feature on that pixel to use on future operations.

The most used features detectors are: Scale-invariant feature transform(SIFT)[8],Speeded up robust features (SURF)[11], Features from accelerated segment test (FAST)[10] and Binary Robust Invariant Scalable Keypoints(BRISK)[6].

After investigating the different methods for detecting features, the software chosen was SURF feature detector. The reason for this choice was based on the fast computation of the method, and overall good results when compared with the other methods.

E. Depth map

An RGB-D dataset, as seen before, its a group of RGB images with their correspondent depth image. From the depth images is possible to retrieve the correspondent depth map. A depth map is an image that contains the information related to the distance to the surface to the scene object.

The term "Depth" is linked to the coordinate of the z- axis, which is the central axis of the camera reference system.

F. Pose estimation

Camera Pose estimation is one of the most significant problems in "simultaneous localization and mapping" (SLAM). To be able to construct an accurate 3D reconstruction, it's necessary to treat the features extracted, and use them to estimate the relative pose of the image. In this chapter, different types of pose estimation algorithms will be presented, according to the inputs received.

In order to estimate the pose, it can be used three algorithms:

The Perspective-n-Point (PnP) problem has a main objective of retrieving the position and orientation of the camera, based on n points

The Perspective-n-Line (PnL) algorithms, use lines has an input to estimate the camera pose.

The Perpective n Lines and n Points algorith, that uses both lines and points to estimate the camera pose.

a) RANSAC optimisation algorithm: To improve the features used by the software, it necessary to filter our results so it fits better within desired model .

Since part of the data used for the camera pose estimation contains outliers, it's applied an algorithm to eliminate the outliers so our pose estimations are more precise. This algorithm is called RANSAC.

The RANSAC algorithm estimates parameters of a model with random sampling. Given a data contain both inliers and outliers, RANSAC finds the optimal fitting result.

At first, a sample of the data is restrained randomly selected from the input data. A fitting model and the corresponding model parameters are computed using the sample.

After this, the algorithm checks which elements of the entire data are consistent with the model started. If a data element does not fit the model, it will be considered as an outlier. In this case, an outlier is an element that is greater than the error threshold defined as the maximum deviation. The set of inliers obtained for the fitting model is called the consensus set. The procedure is repeated until it reached the best model with a greater number of inliers.

b) Bundle adjustment: Bundle adjustment is defined as a refinement tool for solving the problem of localization and mapping simultaneous. The name refers to the 'bundle' of light rays leaving each 3D feature and converging on the different camera centres, which are optimized for both feature and camera positions.When using this algorithm, is always used as the last step of every 3D reconstruction algorithm, to obtain an optimal reconstruction.

The main objective behind Bundle adjustment is to minimize the re-projection error between the image locations of observed and predicted image points, which is represented as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms. It estimates the parameters that improve upon the previous and the resulting series of iterates, which converge to a local minimum in the objective function.

c) Simultaneous localization and mapping Pipelines: Simultaneous localization and mapping (SLAM) is the process of reconstructing a 3D scene using a number of images previously taken. This type of software mainly use two approaches, the incremental and global SLAM.

The incremental SLAM approach is the standard, which consists in adding one image at a time to grow the complete reconstruction. The advantages of this method is that it is robust, but on the bad note, is not scalable since it requires to repeat all the operations every time one image is added.

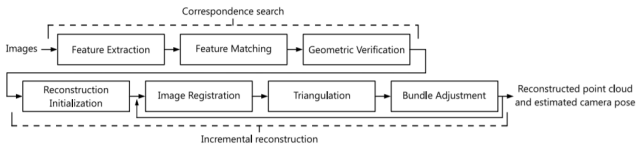

Fig. 3. An example of a incremental SLAM software. The process of optimization is performed each time an image is registered [13]

Global SLAM uses a different method, the entire view graph is computed at same time. Global SLAM methods have proven to be faster and more accurate, in comparison to the incremental approach.

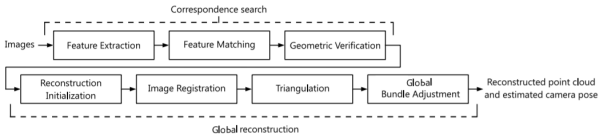

Fig. 4. An example of a global SLAM software. The process of optimization is only performed after registering all images available [13].

For both methods, after computing the camera poses, a triangulation method is performed to obtain a correct 3D reconstruction. Then, from the previous results, it's used a Bundle Adjustment to refine the pose of all cameras.

In the incremental method, the bundle adjustment is performed every time an image is added to the data, on the other hand the global SLAM only performs this action one time.

d) Visual SfM: VisualSFM[15] is an open-source implementation of incremental SLAM pipeline Compared to COLMAP, this software is less flexible, since it only uses one set of algorithms to make the complete 3D reconstruction. This SLAM software was developed in C/c++ and allows SLAM configuration.

VisualSFM is one of the fastest SLAM softwares, as it exploids the multicore parallelism for feature detection, feature matching, and bundle adjustment. The reconstructions can be exported in VisualSFM's NVM file format.

The feature detector/matcher uses SIFT on GPU(SiftGPU), with utilizes the GPU to perform this action faster. Afterwards, the features are sorted in a descending order according to their scale. The top-scale subset, has an higher chance acquiring matches according to the number of images on this subset. This structure grants an decreasing of computation time, without decreasing accuracy.

After obtaining the matched features, an triangulation is performed. Later, if an image fails to match features, VSFm trys to match them again using re-triangulate (RT).

Bundle adjustment [2] uses Levenberg-Marquardt (LM) has the method of choice, guaranteeing good convergence for most problems and reducing the number of necessary iterations.

VSFM, in order to decrease time of computation performs a full bundle adjustment, and a partial Bundle adjustment.

Each time an image is added, instead of performing an full bundle adjustment, it will generate an small group of recently added images and then it performs a partial bundle adjustment on top of that group. Therefore, the time spent on performing the Ba is decreased, improving performance issues.

The images with large re-projection errors or small triangulation angles are filtered, to improve accuracy. The resulted 3D reconstruction is enhanced using RANSAC.

e) COLMAP: COLMAP[12] is an open-source implementation of incremental SLAM pipeline that provides a general-purpose solution usable to reconstruct any scene.

This SLAM software was developed in C++ and allows configuration of some pipelines parameters and to export the sparse reconstruction.

The feature detector/matcher used is RootSIFT.Analogous to VisualSfM, the triangulation is performed taking into account the drift effects prior to the global Bundle Adjustment(BA). COLMAP uses Direct Linear Transform method(DLT) to perform the triangulation to estimate the location of the 3D points. However, BA has better results improving the camera and point parameters.

Usually a significant part of the data introduced in the bundle adjustment are outliers, and it can affect it's result. To perform the bundle adjustment COLMAP uses Ceres Solver [1] and Multicore BA then filters the observations with a large re projection error.

Afterwards,COLMAP filters the resulted data from the BA using several different types of RANSAC.

III. Vision Based Navigation

This chapter details the vision based navigation pipeline as implemented in this thesis. Each block is introduced in an input-output point of view, in order to present its function. Two types of input are considered, namely a sequence of colour (RGB) images and a sequence of colour-depth (RGB-D) images, acquired by moving cameras.

In the first section is detailed the pipeline for the colour image sequences. Then is presented the pipeline for the colour-depth (RGB-D) sequences of images.

A. Navigation systems

After analysing all the content of the previous chapters, it was possible to decide some of the necessary methods to be implemented in the developed software. Methods such as feature detector/extractor, matching features, triangulation and bundle adjustmentII. Furthermore, how the depth images will be integrated into our image-based navigation system.

In conclusion, all of these choices allows us achieving the main objective of this thesis, which is analyse the trajectory and movement of the camera along the time.

B. Colour Images based Navigation

In this section it is detailed 3D reconstruction of a scene using only the images as inputs, knowing the intrinsic parameters of the camera. The software developed uses an incremental approach. Each time an image is received to be processed, the software will perform the optimization operation again, to adjust the complete scene. The bundle adjustment and triangulation are repeated, improving some of the overall results. The figure 5 describes the 3D reconstruction as a cascade of processing blocks, the arrow re-entering the pose

estimation block means that the model is being reinserted to our partial model.

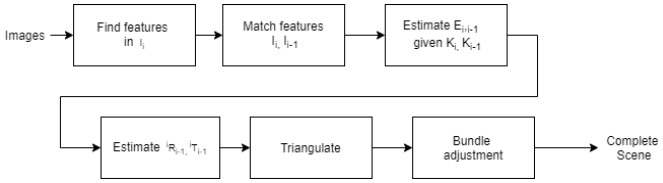

Fig. 5. Pipeline applicable for the SLAM software developed using colour images, where $i = 1, 2, 3...n$ corresponds to the number of images. $^iR_{i-1}$),$^iT_{i-1}$ corresponds to the relative pose of the camera between image i and $i-1$; K_i, K_{i-1} is the intrinsic parameters of the camera.

1) Features Extraction: Upon reading all images and converting them to grey-scale, its necessary to interpreted each image individually (I_i). After registering one (I_i), the extraction of features starts according to the region of interest (ROI) established, in the camera reference system.

The feature detector here considered as, a baseline, is the SURF feature detector, well known for its robustness and fast computation properties.

After, it is necessary to extract features location according to the output computed. This procedure results in two arrays of points that correspond to the features coming from each image, (f_i) and (f_{i-1}). All extracted features are invariant under radiometric and geometric changes so that our software can uniquely recognize them in multiple images. Moreover, when an image is registered, new features are obtained to the scene, increasing the coverage by extending the amount of points present.

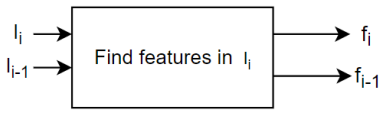

Fig. 6. The input I_i corresponds to the image i of the current dataset, and f_i,f_{i-1} are the resulting features. Where i=1,2,3..N are the number of images. f_i,f_{i-1} dimensions correspond to the number of features extracted between images I_i and I_{i-1}

2) Matching Features: Feature matching is the process of discovering correspondences between two images of the same scene. The most common approach is after detecting the features associated with the image descriptor from image data, it established the features that matched between those images. After receiving all extracted features from images I_{i-1} and I_i, it is necessary to compare both of this array and find the matching points. The matcher will search all features on f_{i-1} and find the correspondence on f_i. The final result will be two arrays containing all matched elements according to the image; F_i and F_{i-1} are the arrays of matched features on the respective pictures. The method used for matching features was FLANN Matcher [9] (Fast Library for Approximate Nearest Neighbors), which is much faster in comparison to other matchers like for example, Brute Force matcher (BFMatcher).

Fig. 7. F_i and F_{i-1} are the features that matched between picture i and $i-1$. This block receives the features location f_{i-1}, f_i and computes two arrays with all the matched features F_i and F_{i-1}.f_{i-1} and f_i have the same dimension between them, since it corresponds to the number of features that are extracted from each image. $F_{i-1}.f_{i-1}$ also have the same dimension between them, since it corresponds to the number of matched points.

3) Geometric Verification based in the Essential matrix: From the matched features location F_{i-1}, F_i and knowing the intrinsic parameters $K_{i-1,i}$, the essential matrix $E_{i-1,i}$ can be estimated using different algorithms. The method chosen to estimate $E_{i-1,i}$ was the five points algorithm in a RANSAC framework (M-SAC) [14]. M-SAC uses the same strategy as RANSAC but is aimed at assessing the essential/fundamental matrix. It uses a P5P algorithm to estimate the essential matrix using the epipolar constraint but removing outliers using MSAC.

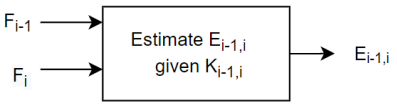

Fig. 8. F_i and F_{i-1} correspond to the 2D matched features location between images i and $i-1$. Their dimensions are linked to the number of features that matched on the previous block (block III-B2). The output is the essential matrix ($E_{i,i-1}$) between images i and $i-1$.

4) Estimation of the incremental Pose Change: Since the cameras are assumed has calibrated, the relative position and orientation can be computed using the essential matrix and the position of corresponding matched points. Using the essential matrix previously computed $E_{i,i-1}$, the matched features F_{i-1}, F_i and intrinsic parameters $K_{i,i-1}$, the camera pose is estimated by performing the Hartley and Zisserman's SVD [16], a decomposition method II-C. The resulting output is the relative camera pose ($^iR_{i-1}$,$^iT_{i-1}$) between images I_i and I_{i-1}.

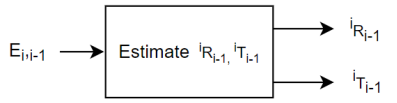

Fig. 9. The input is the essential matrix already estimated $E_{i,i-1}$, resulting in a relative rotation ($^iR_{i-1}$) and translation ($^iT_{i-1}$) between image i and $i-1$

5) Stereo based Reconstruction: From all the data retrieved from the new images (I_{i-1}, I_i), new scene points can be triangulated and added the current view. Some of those reconstructed points may cover a new scene part, from a different perspective. Triangulation is a crucial step as it helps to improve the 3D representation of the 3D scene, enabling to identify the 3D position of each extracted feature. This operation triangulates the 3D locations of 2D matched

points(F_{i-1},F_i) across multiple views (I_{i-1}, I_i) and, it returns the 3D world points position (P_i) corresponding to points matched across various images taken with a calibrated camera.

The process mentioned uses the epipolar constrain (section II-B) to acquire the 3D coordinates of all features using 2D matched features location, the intrinsic parameters and the camera poses (equation 3).

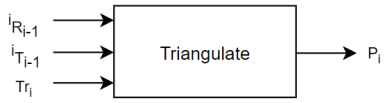

Fig. 10. The inputs received by this block are, the camera pose estimated ($^iR_{i-1}$,$^iT_{i-1}$) and the track of 2D point present in the current view (F_{i-1},F_i). The output is an array with all the 3D location of the matched features(P_i).

6) Bundle adjustment: Bundle adjustment is the process of optimizing the scenario to be rebuilt, as mentioned in section II-F0b. Incremental structure reconstruction and integration into a global model is known to have drifting problems caused by the accumulated errors on the estimated relative camera poses.

Initially estimated poses might not be accurate enough, so it is necessary to provide a function that improves the overall results. This procedure is always performed as the last step of every 3D reconstruction algorithms to solve problems such as the 3D structure and viewing parameters (camera pose, 3D features location).

Each time an image is registered, the operation of Bundle adjustment (BA) is repeated to adjust the complete scenario. Bundle adjustment receives as inputs, the new camera poses ($^iR_{i,i-1}$ $^iT_{i-1}$), 3D points (P_i) and 2D matched points (F_{i-1}, F_i). Resulting in a complete reconstruction which is optimal regarding the noise pertaining to the observed image features. The final result after the bundle adjustment optimization is the points and camera poses that minimize the re-projection errors with respect to the same reference system.

The algorithm chosen for performing the bundle adjustment is a variant of Levenberg-Marquardt [7], which has proven to be the most successful. It requires the solution of linear systems termed the normal equations. When applied to the minimization problems in the framework of bundle adjustment, the normal equations are a sparse block structure.

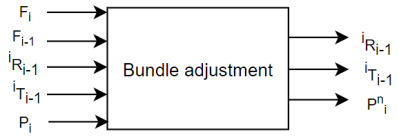

Fig. 11. Th inputs of this block are, all 2D points, the camera pose estimated ($^iR_{i,i-1}$ $^iT_{i-1}$) and the 3D points that resulted from the triangulation block(P_i). It results in a new camera pose and new 3D features location (P_i^n).

C. Colour-Depth Images based Navigation

Colour-depth datasets (RGB-D), adds an extra layer of input data when compared to RGB datasets, namely the depth corresponding to each pixel. Colour-depth images are captured using a microsoft Kinect camera.

The approach to colour-depth datasets images based navigation is different. While in a just iamges approach to be able to build a complete 3D model it is necessary to create a depth map in parallel with the RGB images analysis, when the sensor provides depth the reconstruction is not necessary. There exists still the work of integrating the depth map and the RGB point cloud into a complete point cloud, all point clouds are then merged together according to the rotation estimated, creating a complete representation of the 3D scenery.

Overall, the pipeline used for colour-depth based navigation is the same as the colour navigation, with the construction of point clouds from depth maps running in parallel.

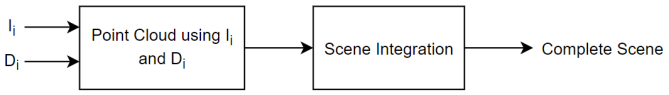

Fig. 12. Pipeline applicable for the SLAM software developed using colour-depth images, where $i = 1, 2, 3...n$ corresponds to the number of images, I_i the colour image and D_i the depth image.

1) Point Cloud Generation: After performing the whole process explained in the previous section, in parallel, the global depth map is being built using a function developed. This function converts all depth images D_i to a depth map using the intrinsic parameters and a scale factor provided by the microsoft Kinect camera. Later, the depth map is merged with the output coming from the RGB images I_i analysis generating a complete point cloud of a pair of images (Pc_i).

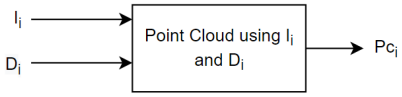

Fig. 13. Point cloud generation. This block receives has inputs the RGB image (I_i) and the correspondent depth image (D_i), the resulting output is a point cloud using images i (P_{c_i}).

2) Scene Reconstruction Estimation: After obtaining the point cloud Pc_i of the image I_i and D_i, it is necessary to merge it with the previous point cloud Pc_{i-1} that contains all point clouds so far. This intersection is made according to the camera pose estimated during the analysis of the RGB images ($^iR_{i,i-1}$ $^iT_{i-1}$). Therefore, the output result will be a point cloud containing all the information merged together.

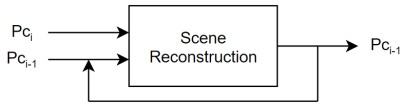

Fig. 14. This block receives two consecutive point clouds (P_{i-1}, P_i). The output is a new point cloud with both P_{i-1} and P_i. This process is done until the last registered image.

IV. Reconstruction error assessment

There are several ways to assess the results of the 3D reconstruction made by the proposed baseline method and

public domains method. The selected assessment aspects were: pairwise error, last pose error and computation time. These are important indicators that allow comparing various reconstruction methodologies.

a) Last pose error: checks if the incremental error between poses has reached a value that is only "noise" and the pose is not viable. This error is computed using the last valid camera pose ($P_{est}(R_f, T_f)$) and comparing it to the correspondent ground truth pose ($P_{Gt}(R_f, T_f)$):

$$e_{Last\ Pose} = ||P_{est}(R_f, T_f) - P_{Gt}(R_f, T_f)|| \quad (5)$$

b) Pair-wise error: is calculated between the estimated pose and the ground truth pose. It gives the average distance/rotation error between the estimated pose and ground truth for each position of the camera (n=1,2,...,N). This is computed by :

$$e_{pairwise} = \frac{\sum_{n=1}^{N} ||P_{est}(R_n, T_n) - P_{Gt}(R_n, T_n)||}{N} \quad (6)$$

c) Computation Time: The measured time is the difference between the time of image registration and the 3D representation of the scene. Allows comparing the performance of the software developed with state of the art alternatives. Computation time also allows us to conclude if the chosen methods are the most optimal to our image-based navigation software.

V. Experiments and Results

This chapter describes the experiments performed to validate the developed methodologies, already discussed in the previous chapters.

On these experiments nine datasets were chosen, six being from ETH3D and three from TUM, containing a total of 11 202 images. The algorithm developed was also applied to the nine datasets of different sizes. These datasets are high-resolution images of different environments where the predominant scenes are interiors and exteriors/interiors of buildings. All these datasets have available the ground-truth camera locations. Throughout all experiments, the results from the software developed in Matlab were compared to the outputs of COLMAP and Visual SFM software.

The first step to make a comparison with the other software was to run each one of them and generate a file with the results acquired. Subsequently, a script was created that centralized all the data in the same reference origin and compared all different errors. The parameters used as comparison factors are: Pair-wise error, last pose error and computation time. The pairwise and last pose rotation errors were represent according to the Euler's angles, where α represents yaw, β represents the pitch, γ represents the roll.

A. Colour images Navigation

Firstly, several colour images were analysed using ETH3D dataset. The approach consists of reading all images in the SLAM software developed in Matlab. Subsequently, the results were compared with data that resulted from COLMAP and VSFM.

1) Last-Pose error: The error of the last position was calculated using the files generated by the different software. These results were taken from the .txt files and centralised in origin [0,0,0], to be compared later on. After adjusting the reference system, a script was created to calculate the different scale factors on each software results. At the end of the data processing, these were the results obtained comparing to the latest valid position of our software.

	Facade	Courtyard	Relif	Eletro	Terrace	Delivery area
Software developed	3.3869	0.3241	1.5524	2.4223	1.6189	2.2176
Visual SFM	2.3856	0.1287	1.0584	2.1913	1.2327	1.3787
COLMAP	3.6019	0.8127	2.0039	2.9317	1.6329	2.4871

TABLE I
TABLE WITH THE DIFFERENT VALUES OBTAINED OF THE LAST POSE ERROR ACCORDING TO EACH DATASET

From the results obtained (table I), the first conclusions we draw are that Visual SfM is the one that produces the best results, with the lowest last pose error on the location of the camera.

Visual SfM presents better result do a process of re-triangulation that occurs after the bundle adjustment, that helps increase the number of matches between pairwise images. The features that failed to match are again inserted in the matcher to try improving the number of matched features, resulting in more accurate results.

The results between COLMAP and the software developed are slightly different, originating from the use of a SIFT detector feature with lower accuracy than SURF. Another reason for this discrepancy is the difficulty in determining matches between pairwise images, which leads an increment of the error between poses.

The results obtained concerning the rotation error in the last position can be seen in the table below. Overall, Visual SfM produces the best results, with an average lower than the other alternative softwares (table II).

	COLMAP			VSFM			Software developed		
	α	β	γ	α	β	γ	α	β	γ
Fascade	1,6382	0,3094	1,1938	1,4016	0,6398	1,3769	1,2122	0,3773	0,3062
Courtyard	1,7327	0,3095	1,8594	1,3844	0,0345	1,8886	2,4004	0,8177	1,0309
Relif	1,4894	0,6382	1,2122	1,2521	0,6323	1,1712	1,7444	0,7923	1,2032
Eletro	0,8812	0,4789	1,3943	0,9828	0,4512	1,274	1,0248	0,4683	1,2982
Terrace	0,2411	0,3155	0,6394	1,3090	0,5448	1,7567	0,2034	0,5251	1,3492
Delivery area	1,5215	0,8551	0,6423	2,2577	0,3456	1,9606	2,3833	0,4868	0,4391

TABLE II
TABLE PRESENTING THE ROTATION ERROR IN THE LAST POSE

Fig. 15. Graphic of the error variation according to each of the Euler angles. a) Graphic of Yaw error on the last pose α. b) Graphic of pitch error on the last pose β. c) Graphic of roll error on the last pose γ.

From both translation and rotation error, it possible to observe (graphic 15) that Visual SfM is the software with the best results. But the software developed in Matlab has proven to be more accurate then COLMAP. In some cases,

the software developed in Matlab reached values with 20% more accuracy then COLMAP.

2) Pairwise error: Starting by analyzing the pairwise error it was necessary to do the same data treatment as for the last position error (section IV), in which it was necessary to change the reference system and find the scale factor between the different data. After this procedure, the calculations were made. The results obtained were the following (table III):

	Facade	Courtyard	Relif	Eletro	Terrace	Delivery area
Software developed	5.8460	4.1596	2.4137	1.5891	1.4236	4.4343
Visual SfM	4.6750	4.0970	2.1240	1.2310	1.1579	4.3897
COLMAP	5.2151	4.0640	2.6694	1.8296	1.6844	4.6799

TABLE III
PAIRWISE ERROR OBTAINS FROM THE DIFFERENT DATASETS USING COLMAP AND VISUAL SFM

The results presented in the table III shows that the pairwise error is very similar between all state-of-the-art software. However, it is visible that Visual SfM is the SLAM software with better results since the extra optimizations, guarantee that the failed matched features are re triangulated, leading to an increase of accuracy. Between COLMAP and our software, the difference is not significant enough to be highlighted.

All software present a low pairwise translation errors. This data shows that the error present on the calculation of each pose is very small. However, Visual SfM is able to improve his accuracy by an average of 25% in relation to COLMAP. Matlab improved his precision of only 10% in relation to COMLAP.

	COLMAP			VSFM			Software developed		
	α	β	γ	α	β	γ	α	β	γ
Fascade	1,7139	0,3156	1,9523	1,3701	0,4049	1,1018	1,9449	0,8797	0,8135
Courtyard	1,7227	0,1658	1,4400	1,4372	0,0075	1,4666	0,6197	0,8499	0,1589
Relif	1,1265	0,3632	0,7256	1,1286	0,9712	0,6219	1,3510	0,7775	0,7579
Eletro	0,8671	0,4689	1,3621	0,9671	0,4478	1,2801	1,0241	0,4571	1,3212
Terrace	0,4753	0,1964	1,2537	1,3184	0,0963	0,3782	0,6949	0,1855	0,3496
Delivery area	1,6886	0,0518	1,7994	1,9813	0,5587	1,7038	1,6330	0,6597	0,8298

TABLE IV
TABLE CONTAINING ALL PAIRWISE ROTATION ERRORS ACCORDING TO EACH OF EULER ANGLES

The pairwise rotation error is shown in table IV. It is shown that the behaviour changes, but in general Visual SfM is the software with the lower error, presenting an accuracy greater than 20% in comparison to Matlab.

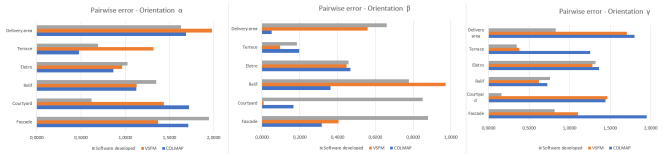

Fig. 16. Graphics containing all pairwise rotation errors. Each of these graphics correspond to the average error pairwise of each of the Euler angles. a) Graphic of the pairwise yaw error α. b)Graphic of the pairwise pitch error β. c) Graphic of the pairwise roll error γ.

In conclusion, Visual SfM is the best software with the lowest pairwise rotation error and pairwise location error, in comparison to COLMAP and our Matlab software.

3) Computation time: To evaluate the performance of the software developed, it was necessary to measure the computation time. This time was determined from the moment the software read the images until the estimation of the last valid camera pose. The results obtained were as following:

	Facade	Courtyard	Relif	Eletro	Terrace	Delivery area
Software developed	2182.1 s	681.9 s	449.9 s	881.9 s	521,1 s	922.3 s
Visual SFM	1317,1 s	292,6 s	155,1 s	352,3 s	203,5 s	301,3 s
COLMAP	1817,7 s	786,4 s	1232,4 s	881,3 s	1102,4 s	1257,6 s

TABLE V
TIME NECESSARY TO GENERATE N CAMERA POSES FROM SEVERAL DATASETS

After analysing the data retrieved and correlating it according to the inputs used, the following conclusions were drawn. In general, performance-wise Visual SfM is the most optimised software due to the fact that a large part of the computation time is used on optimising all results. In the case of VSFM, this is done innovatively, since it uses partial bundle adjustments instead of totals, leading to ”savings” of computation time. Another feature present in Visual SfM that improves the performance is the use of multi-core parallelism feature detection and Bundle adjustment [2], which allows running several feature detector threads at the same time.

When comparing COLMAP with the developed software, the measured times are not very distinct. Since scene optimisation is similar between the two software, the performance difference is due to the fact that COLMAP uses SIFT feature detectors and the software developed uses SURF feature detectors. As already seen, SURF is a more optimised and faster version of the SIFT detector. Moreover, this difference can be consolidated when analysing the results of both software.

Overall, the most optimised software is Visual SfM, and the results in some case are very different, as shown on the graphics presented. The difference sometimes approaches the 300% in comparison to the worst result on ETH3D Dataset Terrace.

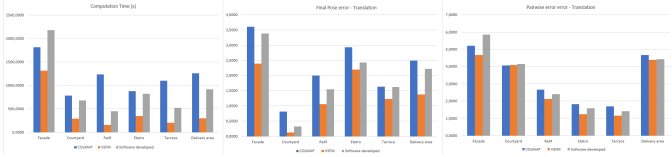

Fig. 17. The reconstruction time in seconds measured until the camera N.

Overall, the most optimised software is Visual SfM, and the results in some case are very different, as shown on the graphics presented. The difference sometimes approaches the 300% in comparison to the worst result on ETH3D Dataset Terrace.

B. Colour-Depth image Navigation

The second part of the experiment was to test the colour-depth datasets and find out what the behaviour of other SLAM software.

For this experience, it was used TUM datasets that contain both colour images, depth images, and the ground truth camera pose.

1) Last-Pose error: In this section, we analyze the existing error of the last pose of the camera, present in both location and rotation.

TUM's datasets are quite large, with some of them having about 2900 photographs. The approach to evaluating the last pose error was similar to the colour datasets. It is necessary to find the scale factor and change the reference system. Subsequently, the last camera pose was analysed using the calculations required to allow the comparison of all results.

The results obtained for the the last pose error present on the location are the following (V-B1):

	COLMAP	VSFM	Software developed
Rgbd freiburg1_360	1,4028	1,1312	1,2093
Rgbd freiburg1	1,5372	1,2751	1,3352
Freiburg1 rpy	2,8710	2,1320	2,4320

It is possible to observe that in all cases, Visual SfM provided more accurate results. Sometimes it achieves results with an accuracy greater than 30% compared to COLMAP, has seen in Freiburg1 rpy. However, when compared to the software developed in Matlab, the difference is not so accentuated, remaining only at 14%.

Regarding the last pose rotation error, sometimes the difference between results is close to null, in case of yaw (α). This is due to the fact that the movement on this axis is not significantly high enough to be evaluated. But also because there is a high number of images, leading to a close to uniform rotation movement.

In the case of the pitch angle, Visual SFM shows an improvement of 80% compared to COLMAP and the software in Matlab. On the rotation roll (γ), the difference becomes minimal, remaining in an improvement of only 12%.

	COLMAP			VSFM			Software developed		
	α	β	γ	α	β	γ	α	β	γ
Rgbd freiburg1_360	1,4213	1,2095	1,3517	1,2261	0,8781	1,1897	1,3567	1,0812	1,3412
Rgbd freiburg1	1,5176	0,8289	1,4594	1,2516	0,4376	1,2114	1,2516	0,8376	1,3387
Freiburg1 rpy	1,8315	0,8263	1,8594	1,5152	0,6268	1,4123	1,5152	0,6268	1,6725

TABLE VI

TABLE CONTAINING ALL ERRORS ROTATION ERRORS PRESENT IN THE LAST POSE. THE EXPERIMENT WAS CARRIED OUT ON 3 TUM DATASETS, IN WHICH THE ERROR VALUES WERE RECORDED FOR EACH OF THE 3 EULER ANGLES.

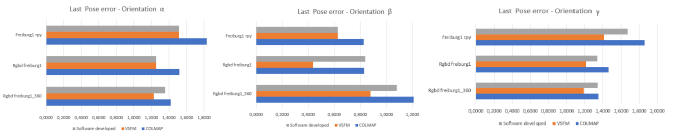

Fig. 18. Graphic of all rotation errors in the last pose of the camera. a) Graphic of the last pose yaw error α . b) Graphic of the last pose pitch error β . c) Graphic of the last pose roll error γ.

Visual SFM had a 14% enhancement over Matlab when calculating the last camera position. However, its rotation error was improved by 80% in one case. Since for SLAM, the position is the most critical factor, it can be concluded that the accuracy of other states-of-the-art softwares evolved on this aspect.

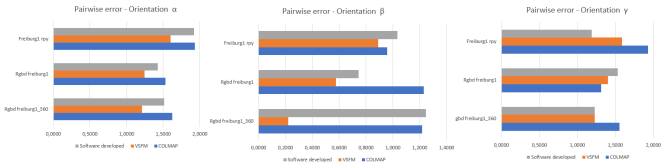

Fig. 19. Graphic containing an average of the rotation errors present in all three Euler angles present along the camera poses. a) Graphic of the pairwise yaw error α. b)Graphic of the pairwise pitch error β. c) Graphic of the pairwise roll error γ.

2) Pairwise error: In this section, it is shown the pairwise error on translation and rotation for colour-depth datasets.

The pairwise calculation was made using the same approach has the colour datasets, the average between the difference of the estimated poses with the ground truth. The results obtained are shown in the table VII:

	COLMAP	VSFM	Software developed
Rgbd freiburg1_360	1,5156	1,0471	1,3128
Rgbd freiburg1	1,3716	0,9794	1,2761
Freiburg1 rpy	1,5250	1,4198	1,6130

TABLE VII

TABLE CONTAINING THE PAIRWISE ERROR IN THE RESULTS OF ALL SOFTWARES

The error present in the camera's location continues to show that VSFM has higher accuracy than the other two software. Meaning that it has a lower cumulative error than other softwares, showing an improvement of 30% in relation to the other two. As already mentioned, this difference is due to the presence of more sophisticated mechanisms to optimize the scene. COLMAP continues to have very low accuracy, producing results with an accuracy of less than 15% compared to Matlab.

Regarding rotation, this behaviour is maintained, having one of the VSFM an improvement in precision of 80% in relation to Matlab and COLMAP.

	COLMAP			VSFM			Software developed		
	α	β	γ	α	β	γ	α	β	γ
Rgbd freiburg1_360	1,6211	1,2187	1,5523	1,2122	0,2211	1,2308	1,5132	1,2450	1,2308
Rgbd freiburg1	1,5285	1,2289	1,3123	1,2471	0,5769	1,4015	1,4249	0,7430	1,5310
Freiburg1 rpy	1,9329	0,9561	1,9281	1,6020	0,8912	1,5892	1,9212	1,0320	1,1897

TABLE VIII

TABLE CONTAINING ALL PAIRWISE ERRORS PRESENT IN THE ROTATION OF THE ESTIMATED CAMERA POSES.

In conclusion, the pairwise error is higher in MATLAB and COLMAP. Visual SfM showed the best results, with accuracy over 30% in the camera location and 80% in the rotation.

3) Computation Time: In this section, the time needed to estimate results will be analyzed. This procedure was performed in the same way as for colour datasets. The used RGB-D datasets have a very high number of images, compared to ETH3D. This difference is visible in the computation time, as it takes longer to compute N camera poses. The data obtained were as follows:

	COLMAP	VSFM	Software developed
Rgbd freiburg1_360	526,08 s	2146,70 s	1231,55 s
Rgbd freiburg1	543,96 s	1992,00 s	869,18 s
Freiburg1 rpy	2820,00 s	3280,00 s	3123,42 s

TABLE IX
TABLE WITH THE COMPUTATION TIME USING COLOUR-DEPTH IMAGES

Fig. 20. Graphic with the different computation times in seconds using colour depth images

As can be seen, COLMAP obtained much better results compared to the other two, with a performance improvement of up to 300% compared to VSFM. This difference is do to the fact of the difficulty to find matches on the other two softwares. Between VSFM and the software developed in Matlab, Matlab showed better results than VSFM with a 70% reduction on the computation time.

C. Summary

One of the main objectives of this thesis was to determine if the accuracy of SLAM softwares has improved with time.One way to verifying this type of evolution was to use Matlab tools to build SLAM software, and compare to others such as COLMAP and Visual SfM.

The table X shows the relative improvement of our software developed in Matlab in comparison to Visual SfM. All computed values are concerning the Matlab software since it had better results then COLMAP.

	Relative error - Time	Relative error - Last - Pose	Relative error - Pair-wise
Facade	-0,6568	-0,4197	-0,2504
Courtyard	-1,3356	-1,5182	-0,0152
Relif	-1,9023	-0,4667	-0,1363
Eletro	-1,3332	-0,1054	-0,2909
Terrace	-1,5672	-0,3132	-0,2294
Delivery area	-2,0640	-0,6084	-0,0101
Rgbd freiburg1_360	0,4263	-0,0690	-0,2537
Rgbd freiburg1	0,5636	-0,0471	-0,3029
Freiburg1 rpy	0,04773	-0,1407	-0,1360
Mean Value	-0,8690	-0,4098	-0,1806

TABLE X
IN THIS TABLE, IT IS PRESENTED THE RELATIVE ERROR ON THE LOCATION OF THE CAMERA, IN RELATION TO THE SOFTWARE DEVELOPED IN MATLAB AND VISUAL SFM. MOST OF THE VISUAL SFM RESULTS ARE BETTER THEN THE ONES OBTAINED IN MATLAB

Analyzing each of the errors of the SLAM softwares, it is possible to conclude that this evolution brought an improvement to the results.

Looking at the errors related to the computation time, Visual SfM improved significantly, by achieving more accurate results using only half of the time used by Matlab.

Regarding the accuracy of the results, this difference is significant when compared to Matlab since the accuracy in most cases improved by 40%. Performance-wise, this evolution is better since it takes less time to run a complete dataset on VSFM then Matlab. In many cases, Visual SfM was able to compute the better results using in some cases less 150% time then Matlab. Overall, the results from VSFM are good, but improved in all aspects. COLMAP is also a recent software, but most of the results were worst then Matlab.

From the study performed, it can be concluded that the evolution of SLAM softwares is significant since the result have been improved overall. Considering Matlab tools have been available for a long time, by comparing it to Visual SfM, it is possible to infer that this evolution is significant. Visual SfM was able to introduce some new techniques to this area, improving the accuracy by an average of 40% and computation time by 100%. This way ”pushing forward” SLAM softwares in terms of accuracy and performance.

VI. CONCLUSIONS

The main goal of this work is to provide a SLAM software able to estimate a complete 3D scene using ETH3D and TUM datasets. The proposed SLAM implementation yielded results close to COLMAP and maintained a comparable accuracy. Visual SfM was the method that provided the best results within the three options.

The observation that an implementation based on standard Matlab toolboxes obtained results with an accuracy comparable to the state of the art, indicates that SLAM may have already reached a high R&D maturity level.

REFERENCES

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] Changchang Wu Sameer Agarwal Brian Curless and Steven M. Seitz. Multicore bundle adjustment. *CVPR*, 2011.

[3] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[4] J. J. Dougherty, H. El-Sherief, D. J. Simon, and G. A. Whitmer. Gps modeling for designing aerospace vehicle navigation systems. *IEEE Transactions on Aerospace and Electronic Systems*, 31(2):695–705, 1995.

[5] Lai J. Xu Y. Bugiolacchi R. et al. First look by the yutu-2 rover at the deep subsurface structure at the lunar farside. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume Nat Commun 11, page 3426, 2020.

[6] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011.

[7] Kenneth Levenberg. *A Method for the Solution of Certain Non-Linear Problems in Least Squares*. Quarterly of Applied Mathematics, 1944.

[8] David G. Lowe. Object recognition from local scale-invariant features. *ICCV*, page 1150–1157, 1999.

[9] M. Muja and D. G. Lowe. Fast matching of binary features. In *2012 Ninth Conference on Computer and Robot Vision*, pages 404–410, 2012.

[10] Tom Rosten, Edward; Drummond. Machine learning for high-speed corner detection. *ECCV*, page 430–443, 2006.

[11] Hiromichi Yanagihara Ryuji Funayama. Robust interest point detector and descriptor. *US8165401B2*, 2009.

[12] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Gianluigi Ciocca Simone Bianco and Davide Marelli. Evaluating the performance of structure from motion pipelines. *Journal of Imaging*, 2018.

[14] Philip Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 08 2000.

[15] Changchang Wu. Visualsfm: A visual structure from motion system. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1, 14,, 2011.

[16] R. Hartley A. Zisserman. Fast matching of binary features. In *Multiple View Geometry in Computer Vision*, 2003.