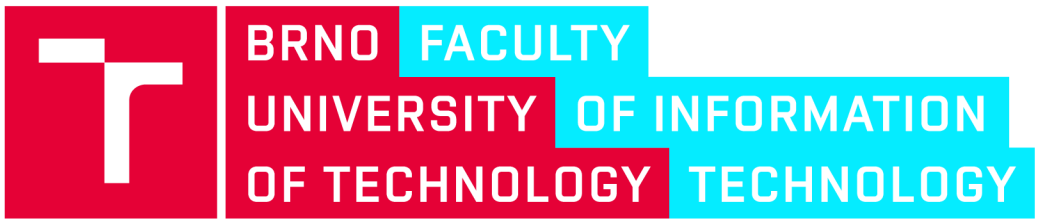


DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

SCALABLE MULTISENSOR 3D RECONSTRUCTION FRAMEWORK

NÁSTROJ PRO 3D REKONSTRUKCI Z DAT Z VÍCE TYPŮ SENZORŮ

PH.D. THESIS
DISERTAČNÍ PRÁCE

AUTHOR — ING. MAREK ŠOLONY
AUTOR PRÁCE

SUPERVISOR — PROF. DR. ING. PAVEL ZEMČÍK
VEDOUCÍ PRÁCE

SUPERVISOR-SPECIALIST — DR. ING. VIORELA ILA
ŠKOLITEL-SPECIALISTA

BRNO 2016

CONTENTS

1 INTRODUCTION 1

2 RELATED WORK 3

3 PRACTICAL APPLICATION 5
- 3.1 Available Sensors . . . 5
- 3.2 Available Datasets . . . 7

4 BACKGROUND 10
- 4.1 Sensor Models . . . 10
- 4.2 Epipolar Geometry . . . 14
- 4.3 Camera Pose Estimation . . . 15
- 4.4 Robust Estimators . . . 19
- 4.5 Structure Triangulation . . . 19
- 4.6 Bundle Adjustment (BA) . . . 20
- 4.7 3D Reconstruction Pipeline . . . 23

5 MULTISENSOR FRONTEND 25
- 5.1 Feature Detection and Descriptor Extraction in Data . . . 25
- 5.2 Multisensor Registration . . . 27
- 5.3 Evaluation of the Front-end Application . . . 32

6 MULTISENSOR 3D RECONSTRUCTION BACK-END 36
- 6.1 SLAM++ . . . 36
- 6.2 System Representation . . . 44
- 6.3 System Building . . . 48

7 EXPERIMENTS AND EVALUATION 49
- 7.1 Evaluation of Stereo Spherical Image Reconstruction . . . 49
- 7.2 Multisensor Reconstruction Accuracy . . . 51

8 CONCLUSION 58

REFERENCES 59

1 INTRODUCTION

The 3D reconstruction problem in the field of computer vision aims for creation of a detailed and accurate model of real-life objects or environments from a set of measurements. Since the introduction of digital photography, the image processing algorithms became one of the most researched topics in the field of computer vision, establishing the basics of recreating 3D structure from camera motion [54]. After four decades of research, the 3D reconstruction topic became a mature area, defining projective camera geometry, statistical inference methods and established techniques for the estimation of sensor pose and 3D structure [27]. Recently, the computational power of modern computers and high performance graphic processing units, have opened the possibilities for the 3D reconstruction algorithms to reconstruct highly detailed 3D representations of large-scale environments in real time. Thus the development effort has focused on the processing of large amount of data from multiple types of sensors to create a consistent 3D model.

The reconstructed 3D models are used in a large variety of applications in fields ranging from computer graphics, virtual reality, architecture to medicine, movie and gaming industry and robotics. The model of an environment offers valuable information for city planning or modification of existing buildings as well as visualization of such modifications. Similarly, the 3D reconstruction can be used as a tool for maintaining the cultural heritage, allowing the virtual presentation of the cultural landmark or artistic object without physical damage to the original object. The non-invasive scene 3D reconstruction finds application in forensics and crime scene investigation, where a crime scene can be scanned to capture all scene details for further interaction and reviewing.

In the film industry, it is advantageous to know the metric 3D information about the environment for Computer Graphics Imagery (CGI) modelling and insertion of special effects, virtual actors or objects into the scene. According to scale of the scene, as well as available budget expenses, different types of on-site scene capture techniques can be utilized. The laser ranging technology provides very precise depth information at the cost of expensive equipment and a need for expert operation. Another option is the processing of images from monocular, stereoscopic or spherical cameras or even from multiple types of sensors simultaneously and constructing the model by fusing partial reconstructions from individual cameras.

For 3D reconstruction of the environment, it is common to scan the scene with one sensor, but using multiple sensor types is more beneficial. Laser scanners or 360° field of view cameras are able to reconstructed whole scene using only few scans, but they are more expensive and require an expert to operate. Other sensors such as monocular cameras are easy to use, but to cover whole scene, large number of photos with satisfactory visual overlap have to be taken. A better 3D model of a scene is the one created by combining the models from different sensors - a model of a whole scene is created

with surrounding scene reconstructed from laser scanners or 360°view cameras and detailed parts of a scene reconstructed from handheld monocular cameras.

The contribution of this thesis is a 3D reconstruction system capable of incorporating data from multiple types of sensors such as monocular, stereoscopic or spherical cameras and laser scanning devices and produce accurate representation of the environment. The focus lies on *unified representation* of different scanning devices, measurements and the spatial relations between them, so one system containing all sensors and measurements is build and optimised to achieve higher accuracy of the reconstruction. The system containing data from multiple types of sensors is optimised using very efficient non-linear graph optimisation library SLAM++[45, 30, 29][1].

To evaluate best data processing approach for multisensor registration we perform an exhaustive analysis of registration of two spherical images and of a registration of spherical and planar image.

1 https://sourceforge.net/projects/slam-plus-plus/

2 RELATED WORK

The techniques for the 3D reconstruction–Structure from Motion (SFM), Bundle Adjustment (BA), Simultaneous Localisation and Mapping (SLAM) have been successfully applied in multiple software systems. *Photo tourism* [49] is able to create 3D models of frequently photographed famous historical buildings or tourist attractions such as Notre Dame from thousands of planar images available on internet services e.g., *Flickr*[1]. Processing an unordered set of images is computationally expensive, so the main focus of the algorithm is the detection of visually similar images, to create reconstruction order that leads to complete model of the scene. The computation usually requires several days of processing on cluster of computers. The software contains image-modelling front-end from large photo collections as well as photo explorer which uses image rendering techniques for smooth translation between images that allows virtual photo tours of famous locations.

Bundler[2] is one of the first SFM software able to process an unordered set of images. Its earlier version was used in *Photo Tourism* project which was later developed into *Photosynth*[3] for Microsoft. *Bundler's* front-end software is able to detect and match feature points across the input image set and to incrementally reconstruct the *sparse* 3D structure of the scene. Modified version of *Sparse Bundle Adjustment* [40] is applied in the process as an underlying optimization engine to refine the reconstruction.

VisualSFM[4] represents an user friendly application for image 3D reconstruction exploiting multicore parallelism [59], fast feature extraction and matching [57] and bundle adjustment [58]. Further, the reconstructed camera and structure information from *VisualSFM* can be used as an input for Patch-based Multi-view Stereo Software (PMVS) by Furukawa et al. [20] to obtain dense 3D reconstruction. PMVS starts with correspondences estimated by SFM algorithm and iteratively expands the depth to surrounding pixels. The false correspondences are filtered out using visibility constraints, by removing patches of depth map that lead to visibility conflict (occlusion) with other patches. The increased set of corresponding points is further used to refine the extrinsic and intrinsic camera parameters in final BA step.

The *OpenMVG*[5] is a library for image processing and multiple view geometry estimation, including algorithms for feature matching of unordered set of images, SFM pipeline, optimisation and visualization tools, as well as simple examples explaining basic functionality. The library also contains a database of intrinsic camera parameters, which can be extracted from image Exchangeable image file format (EXIF) data. The output of the library is a sparse 3D *point cloud* data and camera poses.

The *StereoScan* application [22] allows real-time 3D reconstruction by fusing information from dense depth maps and camera position estimation based on visual odom-

1 https://www.flickr.com/
2 http://www.cs.cornell.edu/ snavely/bundler/
3 https://photosynth.net/
4 http://ccwu.me/vsfm/
5 https://github.com/openMVG/openMVG

etry. The real-time processing is achieved by separating the camera pose estimation process from map building process which links multiple views together and reconstructs reliable point clouds using known camera positions.

Microsoft's Kinect Fusion creates detailed 3D model of the indoor scene using the Kinect device. Only the depth information is used to track the camera position and to reconstruct the 3D model of the scene in real time. The real time, interactive capabilities are possible thanks to the accelerated data processing on the Graphics Processing Unit (GPU), but also non-interactive, offline processing is available. The system finds application in low-cost handheld scanning and geometry-aware and physics-based augmented reality applications.

Commercial software *Capturing Reality*[6] allows 3D reconstruction from multiple sensor types - monocular cameras and CLIDAR device. The multi sensor reconstruction is achieved by transforming coloured 3D point cloud generated by CLIDAR to six planar images by projecting the 3D data to six sides of a cube, and using them for registration to images from monocular cameras.

6 www.capturingreality.com

3 PRACTICAL APPLICATION

The sclable multisensor 3D reconstruction framework was developed for the task of reconstruction of large outdoor scenes for European project IMPART[1] in collaboration with two movie companies, *FilmLight*[2] and *DoubleNegative*[3]. One of the goals was to integrate all measurements acquired by sensors in order to create reconstruction of 3D environment. The available tools, at the time, were too slow for this purpose. The need for in situ visualisation of the 3D reconstructed environment and taking decision on which parts of the scene needs more sampling, motivated the development of a fast and accurate system for 3D reconstruction from multiple sensors.

In this chapter we describe sensors used for 3D reconstruction, their advantages and disadvantages and introduce the datasets captured in the scope of the IMPART project.

3.1 AVAILABLE SENSORS

The first step of 3D reconstruction consists of data acquisition. Two main categories of data acquisition sensors exist - active and passive. Active scanning devices emit some kind of radiation or light and detect its reflection from object to obtain depth map and recreate the object or environment (LIDAR, RADAR, structured light). Passive scanning sensors, on the other hand do not emit light themselves, but rather use reflected natural light instead (CCD cameras).

Monocular Cameras

The conventional cameras are cheap and easy solution to obtain 3D reconstruction. The monocular cameras produce planar 2D images images by projecting 3D scene onto a 2D camera projective plane. The cues from the images, such as silhouettes, shading, texture or motion can be exploited to estimate the 3D geometry of an object or scene. The processing of the video sequences from monocular cameras allows easier detection of corresponding parts of the scene thanks to the big spatial overlap between the consecutive images. The estimation of camera poses and 3D structure of the environment from multiple images of a scene is in literature referred as Structure from Motion (SFM).

1 https://impart.upf.edu/
2 http://www.filmlight.ltd.uk/
3 http://www.dneg.com/

a) b)

Figure 1: a) Spheron[4] camera b) Same part of the scene projected into different (vertical) part of spherical image.

Spherical Cameras

Spherical cameras use *spherical projection* - projecting 3D point of a scene onto a surface of a sphere to create an image capturing whole surrounding scene. Spherical cameras provide images which cover the whole surrounding space, so using spherical images from one or multiple view-points is a feasible way to create 3D models of large environments.

Devices such as *Spheron*[4] capture spherical image by a vertical line-scan camera with wide-angle lens rotating around the centre of projection. The final high-resolution image is created by joining scans into a single image that covers 360° in horizontal and ~ 180° in vertical field of view. For storage purposes the spherical image is stored as rectangular *longitude-latitude* image by mapping from spherical model to 2D dimensions of rectangle. *Spheron* devices are mounted on rigs that allow for precise vertical movement for capturing stereo spherical image pairs with defined vertical baseline.

The main disadvantage of the spherical images and their longitude-latitude representation is the distortion introduced by projection from sphere to rectangular plane. The same parts of the scene can appear very different depending where in the longitude-latitude image they are projected to (Figure 1 b)). This can cause problems when extracting and matching features, especially when the images are captured with wide baseline.

Range sensors

Range detection devices provide information about a depth of the observed object or a scene. Laser scanning devices, also called LIDAR, are often utilized to acquire dense model of a scene. They employ time-of-flight techniques to estimate the distance of a scene point by measuring the time the light beam travels between LIDAR and the point. LIDARs often include rotating mirror that allows to change the angle of the

4 https://www.spheron.com/

Table 1: Dataset details. *The number of monocular planar images for Synthetic dataset is a sum of the images of each scenario.

	CCSR	Cathedral	Atrium	Studio	Synthetic
Spherical Images	3	3	5	4	–
Spherical baseline	~ 6m	~ 23m	~ 3m	~ 1.5m	–
CLIDAR Scans	3	7	–	–	3
CLIDAR baseline	~ 6m	~ 7m	–	–	~ 6m
Planar images	243	92	50	–	30*
Area	250m^2	2500m^2	400m^2	100m^2	250m^2

laser beam and thus scanning area around the device. Specialized 3D LIDARs with added vertical field of view are able to capture dense structured 3D point clouds representing the scene. Some devices such as Faro[5] are capable also to fuse colour information from wide angle lens camera located at the LIDAR sensor with the 3D point cloud data to create the coloured 3D model of the environment.

3.2 AVAILABLE DATASETS

Several datasets containing data from different types of sensors were acquired to evaluate our multisensor processing framework and other applications developed in IMPART project. The planar images have been captured by standard hand-held Canon and Samsung cameras, covering surrounding area of captured scene. The spherical images were acquired with a SpheroCam-HDR[6] system, which captures vertical scan lines by a turning camera with fisheye lenses, synthesises them and provides up to 50 Mpix latitude-longitude image. The CLIDAR data capture was performed using *Faro Focus*^{3D}[7] device providing a 3D point cloud data with assigned colour information for each point. Details about the content of each dataset is shown in table 1.

CCSR dataset

The *CCSR dataset* is an outdoor dataset of an enclosed area of approximately 250m^2. The scene was captured by spherical camera from three positions with the displacement of $5-6$m and three CLIDAR scans are available from approximately same positions as spherical images. Each capture of spherical image was done at two different heights to produce stereo image pairs. The hand-held Canon camera has been used to capture the planar images and covers whole surrounding area. Many subsets of the images are captured with small baseline.

The scene contains visual reflective markers accompanying the CLIDAR *Faro* sensor which serve for the easy correspondence estimation and sensor registration. Using the *Faro* software[7], precise positions of the sensors can be computed. This poses can be

5 http://www.faro.com

6 https://www.spheron.com/

7 http://www.faro.com/en-us/products/faro-software/scene/overview

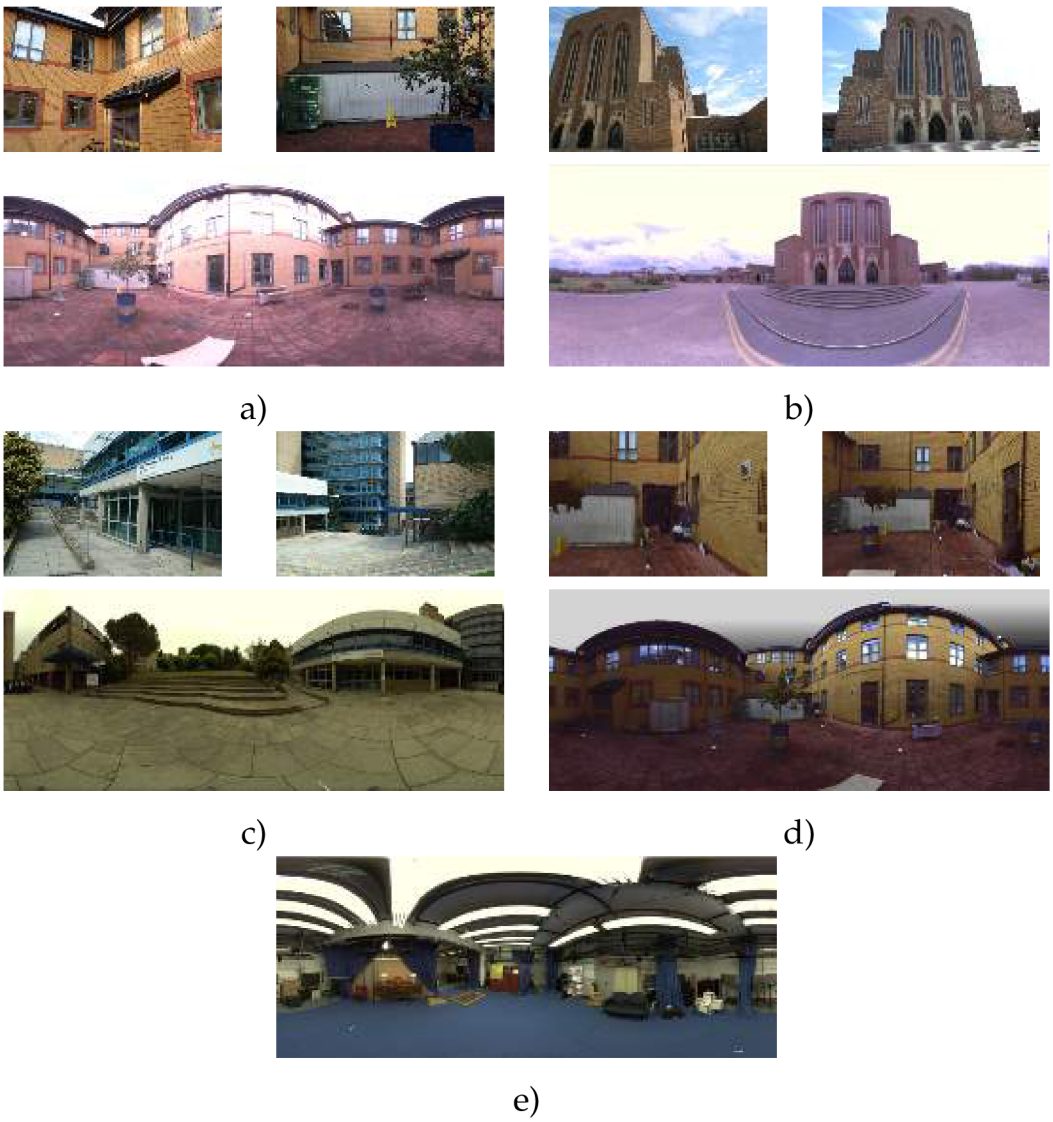


Figure 2: Example data from the datasets, up planar images, down spherical image, a) CCSR, b) Cathedral, c) Atrium, d) Synthetic and e) Studio dataset

used as a reference for comparison of the accuracy of registration of CLIDAR sensor integrated in our system.

Cathedral dataset

The *Cathedral* dataset covers an area of approximately 2500m^2 and captures the scene in front of *Guildford Cathedral* building, surrounding smaller buildings and parking lot. In order to test how the system performs in the case of large sensor displacements, the spherical cameras were placed at positions far apart (approx. 23 m). Seven CLIDAR scans are available for this dataset, which were captured in different day, so lightning conditions and small details in scene may be different than in spherical images. The planar images cover only the cathedral building, images of no other objects were captured.

Atrium dataset

The *Atrium* dataset captures a semi enclosed, outdoor area of approximately 400m^2 using five spherical camera scans. The planar images capture whole surrounding area. The datasets *Cathedral*, *CCSR* and *Atrium* were captured as a part of an European project IMPART[8] and are available upon request[9].

Studio dataset

Studio dataset was captured for the purpose of evaluating the accuracy of spherical image registration. The physical distances between the poses of the spherical cameras were measured as well as the distances to certain distinctive points in the scene. The spherical cameras were precisely placed and aligned to face the same direction. The indoor scene was captured from four spherical camera poses.

Synthetic dataset

For the purpose of evaluating multisensor 3D registration algorithm, especially the registration of CLIDAR/spherical images and planar images, we used dense CLIDAR data to generate artificial views from virtual planar cameras with known calibration and position in the scene. This way we are able to generate images from virtual sensors with known 3D poses which are used as a ground truth for comparison with estimated poses. *Synthetic* dataset contains images generated from CLIDAR data of *CCSR* dataset. The registration of CLIDAR sensors is available from the *Faro* software which utilizes visual reflective markers for the computation of the sensor pose.

Multiple scenarios were considered for the synthetic datasets:

- *Short baseline* - The images were generated from virtual cameras with close distance to each other ($\sim 0.3\text{m}$). These images contain big overlap.
- *Long baseline* - The baseline between virtual sensors was approximately 2.5m and contain bigger change ($\sim 30°$) in rotation compared to small baseline dataset. The images contain smaller overlap.
- *Combined baseline* - This dataset contains images both with small and large baseline and rotations between sensors. This dataset imitates the real scene capturing using a handheld camera.
- *Noise in depth data* - The LIDAR depth data are generally very precise. Therefore to evaluate accuracy of registration of multisensor data in the presence of noise such as in case of stereo spherical image depth map, the depth map available from CLIDAR was artificially perturbed by zero mean Gaussian noise with standard deviation $\sigma = 0.15\text{m}$. This dataset simulates registration of monocular images and stereo spherical images.

8 impart.upf.edu

9 kahlan.eps.surrey.ac.uk/impart/

4 BACKGROUND

This chapter describes image processing, projective geometry, representation of camera models, geometry between cameras observing the scene and the fundamental algorithms for estimation of camera pose and 3D structure.

4.1 SENSOR MODELS

In the following sections we describe the models of different sensors and the details of the imaging process of cameras.

Pinhole Camera Model

The simplest model of describing a camera is called *pinhole camera model*. Pinhole camera model is a specialization of the general projective camera model. This model utilizes *central projection* which assumes a line passing through 3D world point and centre of projection, intersecting image plane Π in point where the image is formed as shown in Figure 3. The projection of the 3D point ${}^{\{c\}}\mathfrak{m} = [X, Y, Z, 1]^\top$ to the camera plane is performed by applying a series of matrix transformation operations specified by a *camera model*. Assuming that the camera centre of projection lies in the centre of *world coordinate frame*, its optical axis is oriented along the $z-$axis and the distance of

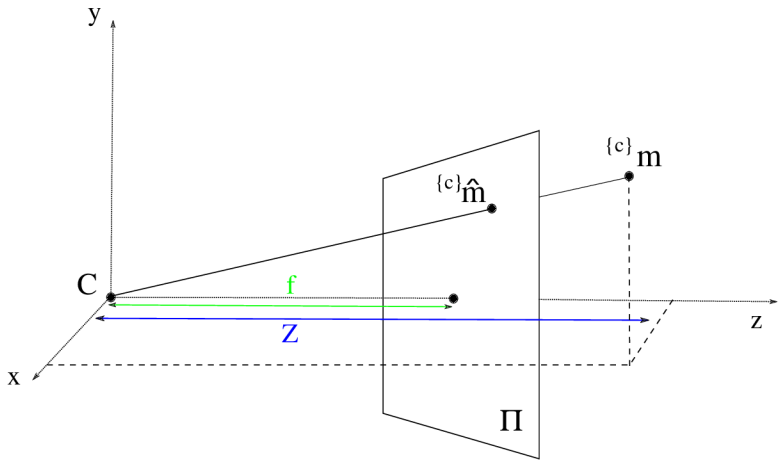


Figure 3: Pinhole camera model. It can be seen that given focal length f, the position of the projected point ${}^{\{c\}}\mathfrak{m}$ in the projection plane Π is ${}^{\{c\}}\hat{\mathfrak{m}} = [f\frac{X}{Z}, f\frac{Y}{Z}, f]^\top$.

the camera projection plane from centre of projection, called *focal length* f, is equal to one, the homogeneous representation of the projection can be described by equation:

$$ {}^{\{i\}}m = \begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \\ 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} . \tag{1} $$

In the terms of geometric relations, this projection transforms the 3D point ${}^{\{c\}}m$ from camera coordinate frame to camera projection plane coordinate frame. Please note that we are using notation ${}^{\{c\}}m$ for a 3D point in the coordinate frame of camera, notation ${}^{\{i\}}m$ for a point in coordinate frame of image (pixel coordinates), and notation ${}^{\{c\}}\hat{m}$ for a point in the coordinate frame of projection surface of the camera, also known as *normalized image coordinates*.

The equation (1) assumes that the 3D point coordinates are in camera coordinate frame, i.e., coordinate frame with origin in the centre of projection. This is not usually valid in real scenarios where camera pose and 3D points are defined in *world coordinate frame*. Therefore to project the 3D point ${}^{\{w\}}m$ to camera projection plane, first it must be transformed from world coordinate frame into the camera coordinate frame. This is achieved by using a rigid transformation $[R \mid t]$, where R is the rotation of the camera coordinate frame and $t = -RC$, C being position of the camera centre in the world coordinate frame:

$$ {}^{\{c\}}m = [R \mid t]\,{}^{\{w\}}m . \tag{2} $$

Rotation matrix R is a 3×3 matrix, element of Special Orthogonal group $SO3$, which is a group of all valid rotations around the origin in 3D Euclidean space. The matrix $[R \mid t]$ represents *extrinsic* camera parameters.

The focal length of the real world cameras is generally different than one, therefore to transform the point ${}^{\{c\}}m$ from camera coordinate frame to point ${}^{\{i\}}m$ in the image coordinate frame the projection has to be scaled to take this into account. Also the principal point $c = [c_x, c_y, 1]$ is introduced which defines the coordinates of centre of projection plane in a coordinate frame of the image. Focal length and principal point are called *intrinsic camera parameters*. They are independent from the structure of the scene or camera position or rotation and can be estimated by camera calibration [48]. Upper triangular matrix K:

$$ K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} , \tag{3} $$

containing intrinsic parameters f and c, and defining central projection is called *camera calibration matrix*. We can write equations (1) and (2) as:

$$ \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \approx \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} , \tag{4} $$

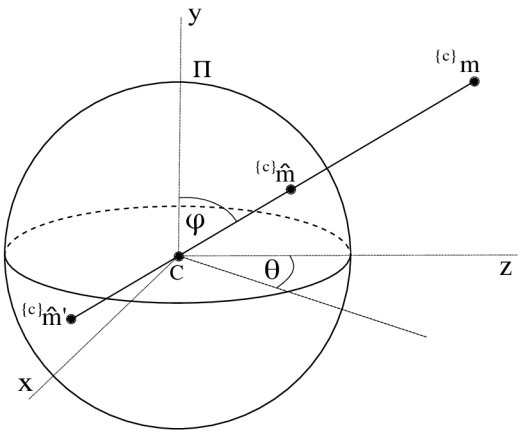


Figure 4: Model of spherical camera.

or shortly as:

$$ {}^{\{i\}}\mathrm{m} \approx \mathrm{K}[\mathrm{R} \mid \mathrm{t}]\,{}^{\{w\}}\mathrm{m}\,. \tag{5} $$

If the calibration matrix K of the camera is known, the normalized coordinates ${}^{\{c\}}\hat{\mathrm{m}}$ can be computed using equation:

$$ {}^{\{c\}}\hat{\mathrm{m}} = \mathrm{K}^{-1}\,{}^{\{i\}}\mathrm{m}\,. \tag{6} $$

The extrinsic camera parameters together with camera calibration matrix K form the camera projection matrix P, a 3×4 matrix which defines a projection of a 3D point form a world coordinate frame to 2D image coordinate frame:

$$ \mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathrm{t}]\,. \tag{7} $$

Spherical Camera Model

Central panoramic cameras [52], unlike the pinhole cameras, use the imaging surface of a sphere instead of a planar one. In the projective geometry, the projection of a 3D projective space onto a spherical surface is topologically equivalent to the projection onto a projective plane.

Figure 4 shows the model of a spherical camera with a centre of projection C and an unit sphere with centre in the centre of projection is defined. The line passing through the 3D point ${}^{\{c\}}\mathrm{m}$ and the camera centre C intersects the spherical surface Π in two points, so it is necessary to assume only half-lines to remove the projection ambiguity. The set of all projections of visible 3D points captured by spherical camera is called *spherical image*, and the spherical projection is defined by a map from 3D space to a surface of a sphere.

The 3D point ${}^{\{c\}}\hat{\mathrm{m}}$ on the surface of unit sphere can be computed as:

$$ {}^{\{c\}}\hat{\mathrm{m}} = \frac{{}^{\{c\}}\mathrm{m}}{\|{}^{\{c\}}\mathrm{m}\|}\,, \tag{8} $$

where $\|{}^{\{c\}}\mathrm{m}\| = \sqrt{X^2 + Y^2 + Z^2}$ is a L_2 norm of a vector ${}^{\{c\}}\mathrm{m}$.

Similar to the pinhole camera model, the pose of spherical camera in the world coordinate frame is defined by transformation matrix $[\mathrm{R} \mid \mathrm{t}]$, composed of relative rotation

R and translation t, which transforms the 3D point ${}^{\{w\}}\mathrm{m}$ from the world coordinate frame into the local coordinate frame of the spherical camera:

$$ {}^{\{c\}}\mathrm{m} = [\mathrm{R}\,|\,\mathrm{t}]\,{}^{\{w\}}\mathrm{m}\,. \tag{9} $$

The spherical coordinates are often expressed with angle parameters $[\theta, \varphi]$ (Figure 4), longitude θ describing the angle between z axis and projection of vector $\mathrm{C}^{\{w\}}\mathrm{m}$ to plane defined by axis xz, and latitude φ being the angle of vector $\mathrm{C}^{\{w\}}\mathrm{m}$ and axis y. Assuming that the radius of the sphere is one, the mathematical transformation between spherical coordinates and angular coordinates is given by equations:

$$ \begin{aligned} {}^{\{c\}}\hat{\mathrm{m}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \begin{bmatrix} \sin\theta\,\sin\varphi \\ \cos\varphi \\ \cos\theta\,\sin\varphi \end{bmatrix}, \\ \begin{bmatrix} \theta \\ \varphi \end{bmatrix} &= \begin{bmatrix} \arctan\left(\frac{x}{z}\right) \\ \arccos y \end{bmatrix}. \end{aligned} \tag{10} $$

Multiple formats to store spherical image are used depending on the application. Full panoramatic image stores spherical image as a 2D rectangular image with x axis representing longitude and y axis representing latitude. The range along the x axis is $u_i \in [-\pi, \pi]$ and axis y $v_i \in [-\pi/2, \pi/2]$ and the mapping between longitude-latitude and pixel coordinates is given by equation:

$$ {}^{\{i\}}\mathrm{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\theta+\pi}{2\pi}(M-1)+1 \\ N - \frac{\varphi+\pi/2}{\pi}(N-1)+1 \\ 1 \end{bmatrix}, \tag{11} $$

where M and N are dimensions of the image horizontally and vertically. Other possible format is a cubic panorama [18] consisting of six images representing projection of spherical image onto unit cube.

LIDAR Model

All LIDAR devices work on the principle of measuring time between optical pulse generation and its receiving. A laser pulse is generated in certain direction, reflects upon interaction with an object and returns to the device. High speed counter measures the time of flight between generation of the pulse and its return.

In this thesis we model LIDAR devices as a sensor with a pose $[\mathrm{R}\,|\,\mathrm{t}]$ in world coordinate frame, similar to pinhole or spherical camera model, and expect the data to be a cloud of 3D points in the coordinate frame of sensor with intensity or colour information. For detailed information about processing of LIDAR signal and computation of the point cloud we refer reader to [39].

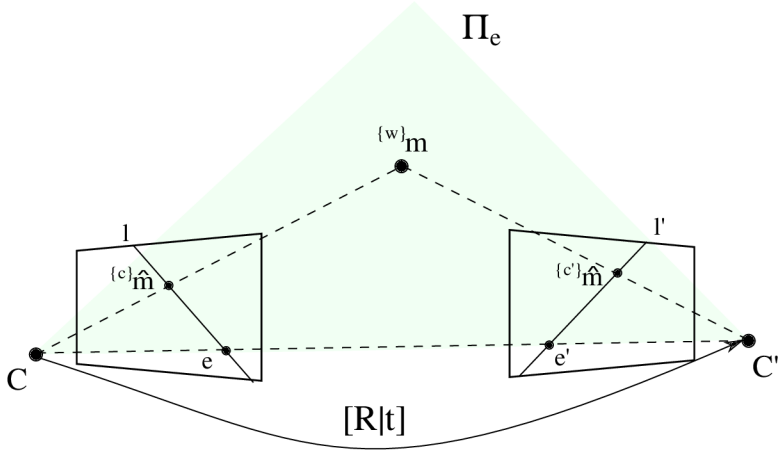


Figure 5: Epipolar geometry between two planar cameras.

4.2 EPIPOLAR GEOMETRY

Based on the projective camera model, two cameras capturing a scene from different positions are constrained by geometric relations between camera centres, 3D points and their 2D images defined by *epipolar geometry*.

Figure 5 shows two cameras are observing same scene. The 3D point ${}^{\{w\}}\mathfrak{m}$, the camera centres C and C' and the corresponding points in the projection planes of cameras ${}^{\{c\}}\hat{\mathfrak{m}}, {}^{\{c'\}}\hat{\mathfrak{m}}$ are coplanar i.e., lie on the same plane Π_e, called *epipolar plane*. Epipolar plane intersects the camera projection plane in epipolar lines l, l' which contain the images of 3D point. The epipole e - a distinct point in the camera image plane is formed by projection of other's camera centre point as if was considered as a point in space. Epipoles will always lie on the epipolar plane and epipolar lines, independent of the position of 3D point. Epipolar points may lie in infinity if the camera projection planes are coincident.

According to epipolar geometry, to mathematically describe the relation between the images ${}^{\{c\}}\hat{\mathfrak{m}}, {}^{\{c'\}}\hat{\mathfrak{m}}$ of 3D point ${}^{\{w\}}\mathfrak{m}$, without loss of generality, we can assume that the centre of first camera lies in the origin of world coordinate system and its rotation matrix is identity. The second camera is positioned according to rigid transformation $[R \mid t]$. If the points ${}^{\{c\}}\mathfrak{m}$ and ${}^{\{c'\}}\mathfrak{m}$ are the coordinates of the images of 3D point ${}^{\{w\}}\mathfrak{m}$ in the coordinate system of cameras C and C' respectively, the points are related by rigid transformation:

$$ {}^{\{c'\}}\mathfrak{m} = R\,{}^{\{c\}}\mathfrak{m} + t\,. \tag{12} $$

And in the terms of images ${}^{\{c\}}\hat{\mathfrak{m}}, {}^{\{c'\}}\hat{\mathfrak{m}}$ and their scales λ and λ':

$$ \lambda' {}^{\{c'\}}\hat{\mathfrak{m}} = R\lambda\,{}^{\{c\}}\hat{\mathfrak{m}} + t\,. \tag{13} $$

This equation relates the vectors ${}^{\{c\}}\hat{\mathfrak{m}}, {}^{\{c'\}}\hat{\mathfrak{m}}$ through the rigid transformation $[R \mid t]$. In order to eliminate scales, both sides can be pre-multiplied by skew-symmetric matrix $[t]_\times$:

$$ \lambda' [t]_\times\, {}^{\{c'\}}\hat{\mathfrak{m}} = [t]_\times R\lambda\, {}^{\{c\}}\hat{\mathfrak{m}}\,. \tag{14} $$

Another pre-multiplying with ${}^{\{c'\}}\hat{m}^{\top}$ yields left side of equation to be equal to zero, since the vector $[t]_x\,{}^{\{c'\}}\hat{m}$ is perpendicular to vector ${}^{\{c'\}}\hat{m}^{\top}$ and thus its inner product ${}^{\{c'\}}\hat{m}^{\top}[t]_x\,{}^{\{c'\}}\hat{m} = 0$ is zero. Right side of equation is thus equal to zero, and the scale λ can be eliminated because it is non-zero, non-negative variable:

$$ {}^{\{c'\}}\hat{m}^{\top}\,[t]_x R\,{}^{\{c\}}\hat{m} = 0\,. \tag{15} $$

The Equation 15 describes the principle of epipolar geometry and the 3x3 matrix

$$ E = [t]_x R \tag{16} $$

is the algebraic representation of epipolar geometry and describes the relative transformation between two cameras and is called the *essential matrix* [27].

4.2.1 *Epipolar Geometry for Guided matching*

The guided matching reduces the number of outliers in the set of corresponding image pairs computed by matching algorithm by introducing matching constraints derived from epipolar geometry relations between the cameras. Assume only image ${}^{\{c\}}\hat{m}$ (Figure 5) is known and we want to know how the corresponding point ${}^{\{c'\}}\hat{m}$ is constrained. The epipolar plane Π_e defined by camera centres and vector ${}^{\{c\}}\hat{m}$ intersects projection plane of second camera in epipolar line $l' = E\,{}^{\{c\}}\hat{m}$. The corresponding image ${}^{\{c'\}}\hat{m}$ of 3D point ${}^{\{w\}}m$ lies on this line, satisfying equation $l'\,{}^{\{c'\}}\hat{m} = 0$, so in the terms of stereo correspondence algorithm the search is restricted to 1D space.

4.3 CAMERA POSE ESTIMATION

Camera registration algorithms estimate the relative transformation between two cameras based on visual information from the camera images. We assume that the intrinsic camera parameters are known for both cameras and that the cameras capture overlapping parts of the scene. In the initialization phase, the areas of the scene that are observed by both cameras are detected by extracting the 2D feature points and matching against feature points of other images, creating a set of 2D-2D corresponding points. Depending on the available information, three situations may arise:

- The 3D depth information in the coordinate frame of the camera is known for the 2D correspondences in both images (from depth map or previous camera registration). In this case, the relative camera position can be estimated from alignment of the 3D structure from one camera to other.

- The 3D depth information is available for one camera, but from 2D-2D correspondences we can establish the relations between 3D points and their 2D images in second camera. From those correspondences the pose of second camera can be estimated using Perspective-n-Point (PnP) algorithm.

- No 3D information is available, only 2D-2D correspondences between cameras without known poses. In this case we can perform the initialisation - estimation of the relative pose between cameras only from 2D-2D correspondences. It is

important to find the best pair of images for the initialization of system. The images from nearby cameras suffer from large triangulation errors due to small baseline. On the other hand, images captured by cameras with large baseline tend to contain little or no overlap between the images thus failing to detect enough good corresponding points.

In following sections we will look at these situations in more detail.

4.3.1 *Pose from 3D structure alignment*

If the 3D object points corresponding to 2D image points are known for both cameras, the problem of the estimation of the relative transformation between cameras can be formulated as finding transformation between two sets of 3D points. The transformation estimation between two sets of 3D corresponding points is addressed in [3]. The optimal transformation $[R|t]$ relates corresponding 3D points in sets $s = [s_0, s_1, \ldots, s_n]$ and $d = [d_0, d_1, \ldots, d_n]$ by:

$$s_i = Rd_i + t\,, \tag{17}$$

where R is a 3×3 rotation matrix and t is a 3×1 translation vector. The solution to the optimal transformation can be found by minimizing *least squares error*:

$$E_R(R, t) = \sum_i^n \|s_i - (Rd_i + t)\|^2\,. \tag{18}$$

By finding the centroids $\hat{s}, \hat{d}$ of the 3D point sets and transforming the points the coordinate frame so the centroid of new point sets s^c, d^c lie in the origin of this coordinate frame removes the translation component from the error term (18) and the equation can be rewritten to:

$$E_R(R) = \sum_{i=0}^n s_i^{c\top} s_i^c + d_i^{c\top} d_i^c - 2 s_i^{c\top} R d_i^c\,. \tag{19}$$

The error is minimized when the term $s_i^{c\top} R d_i^c$ is maximised which equals to maximising $\mathrm{tr}(R, H)$, where H is a correlation matrix [3]:

$$H = \sum_{i=0}^n d_i^c s_i^{c\top}\,. \tag{20}$$

Operation tr denotes *trace*, a sum of diagonal elements of square matrix. The solution is found by singular value decomposition (SVD) which decomposes the matrix $H = USV^\top$ to product of matrices - two unitary matrices U and V and a diagonal matrix S. The optimal rotation matrix R is:

$$R = VU^\top\,. \tag{21}$$

The optimal translation can be obtained from the translation that aligns centroids $\hat{s}, \hat{d}$ of the point sets:

$$t = \hat{s} - R\hat{d}\,. \tag{22}$$

4.3.2 *Iterative Closest Point (ICP) for 3D Point Cloud Registration*

If the 3D data is available for each camera, the relative pose can be estimated without prior detection of point correspondences by performing 3D point-cloud registration. The 3D points can be obtained from the depth map and registered using ICP algorithm.

The ICP algorithm has been widely adopted to align two given point sets [5, 46]. It finds a rigid 3D transformation (rotation R and translation t) between two overlapping clouds of points by alternating between closest point computation for correspondence estimation and iteratively minimising squared-error of registration between the corresponding points from one set to the other:

$$E_R(R,t) = \sum_{i}^{n_s} \sum_{j}^{n_d} \lambda_{i,j} \|s_i - (Rd_j + t)\|^2 , \tag{23}$$

where n_s and n_d are the number of points in the model set s and reference set d, respectively, and $\lambda_{i,j}$ are the weights for a point match.

In each ICP iteration, the rigid 3D transformation can be efficiently calculated by singular value decomposition (SVD) [27].

The disadvantage of ICP algorithm is that it requires good initialisation and when applied to point cloud registration, the ICP algorithm can become very slow with large number of 3D points.

4.3.3 *Pose from 3D-2D correspondences*

The camera pose estimation algorithm, or the Perspective-n-Point (PnP) algorithm, computes the 6DOF pose of the camera given the correspondences between 3D points in the world coordinate frame and their 2D projections in the camera image and camera calibration matrix. The *P3P* algorithm [21] solves the minimal form of the PnP algorithm, requiring minimum of $n = 3$ point correspondences. The camera pose estimation problem can be formulated as a geometric problem based on the reprojection equation of a camera (1). The relations between the 3D and 2D points are used to build a system of equations (Figure 6), based on the law of cosines: given the three 3D points ${}^{\{w\}}m_{0,1,2}$, their corresponding points ${}^{\{c\}}\hat{m}_{0,1,2}$ in the camera projection surface, camera centre C, distances $w_0 = \|C{}^{\{w\}}m_0\|, w_1 = \|C{}^{\{w\}}m_1\|, w_2 = \|C{}^{\{w\}}m_2\|$, angles $\alpha = \angle{}^{\{c\}}\hat{m}_1 C{}^{\{c\}}\hat{m}_2, \beta = \angle{}^{\{c\}}\hat{m}_0 C{}^{\{c\}}\hat{m}_2, \gamma = \angle{}^{\{c\}}\hat{m}_0 C{}^{\{c\}}\hat{m}_1$, distances $d_0 = \|{}^{\{w\}}m_0{}^{\{w\}}m_1\|$, $d_1 = \|{}^{\{w\}}m_1{}^{\{w\}}m_2\|$, $d_2 = \|{}^{\{w\}}m_0{}^{\{w\}}m_2\|$. We form the following system:

$$\begin{aligned} w_1^2 + w_2^2 - w_1 w_2\, 2\cos\alpha - d_0^2 &= 0\,, \\ w_2^2 + w_0^2 - w_0 w_2\, 2\cos\beta - d_1^2 &= 0\,, \\ w_0^2 + w_1^2 - w_0 w_1\, 2\cos\gamma - d_0^2 &= 0\,. \end{aligned} \tag{24}$$

By solving the set of linear equations in (24) the distances d_0, d_1, d_2 can be obtained and from that the coordinates of 3D points ${}^{\{c\}}m_{0,1,2}$ in the coordinate frame of the

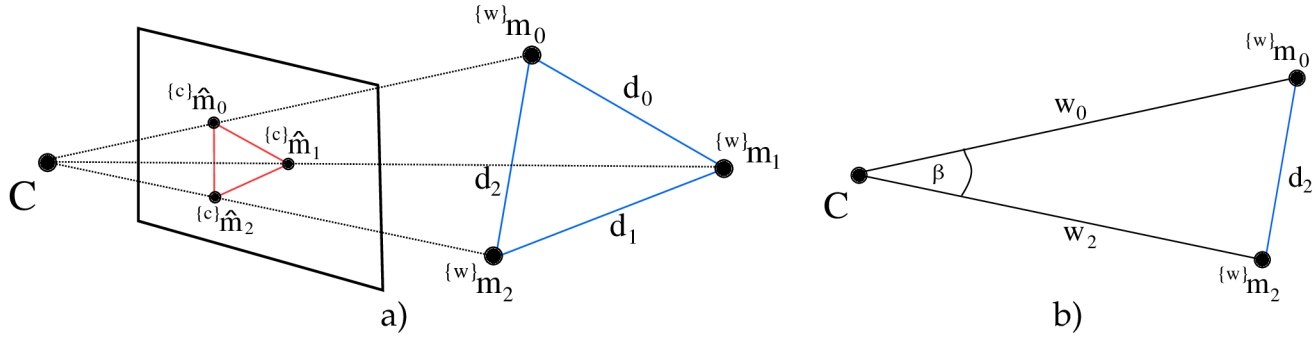


Figure 6: Illustration of P3P problem; a) Relations between world points ${}^{\{w\}}m_{0,1,2}$ and corresponding points ${}^{\{c\}}\hat{m}_{0,1,2}$ in camera projection surface. b) One of triangles used for building equations (24) by applying cosine law.

camera computed. The camera pose is estimated by finding the rigid transformation between the world 3D points ${}^{\{w\}}m_{0,1,2}$ and local 3D points ${}^{\{c\}}m_{0,1,2}$. This algorithm produces up to four solutions for the pose estimation problem, but using fourth point removes the ambiguity.

Another approach for solving the PnP problem has been presented in [38]. The *Efficient PnP* algorithm solves the problem for $n \geqslant 4$ corresponding points in linear time complexity. This method expresses each 3D point as a weighted sum of four virtual control points and the coordinates of those control points are unknowns of the problem.

4.3.4 *Pose from 2D-2D correspondences*

Without any prior 3D information, the relative pose between cameras can be estimated directly from epipolar geometry. To estimate the relative pose of the cameras, without loss of generality we can assume the position of first camera in the centre of coordinate frame with zero rotation along coordinate axis: $[I \,|\, \mathbf{0}]$. The second camera pose can be expressed relative to the first in terms of rotation and translation $[R \,|\, t]$. From (16) we can observe that the essential matrix E is a product of a relative rotation R and a skew-symmetric translation matrix $[t]_\times$. Factorizing the essential matrix using the SVD algorithm [27], $E = USV^T$, decomposes the Essential matrix to three matrices, two unitary matrices U and V and a diagonal matrix S. We can obtain up to four possible solutions for relative transformation between the cameras:

$$\begin{aligned} P' &= [UWV^T \,|\, \pm u_3], [UW^TV^T \,|\, \pm u_3]\,, \\ W &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\,, \end{aligned} \tag{25}$$

where u_3 is a last column of U, and using the *cheirality* [56] constraint, the correct solution can be identified. The concept of cheirality has been introduced in [27]. The sign of the cheirality value indicates whether the 3D point lies in front of camera or behind it. For the estimated camera poses the cheirality of the corresponding points

has to be positive. The obtained relative transformation is computed up to an arbitrary scale. From the relative transformation the camera projection matrices are $P = K[I \mid \mathbf{0}]$ and $P' = K[R \mid t]$ according to (7).

4.4 ROBUST ESTIMATORS

The pose estimation algorithms are sensitive to outliers [25]. In geometry estimation, such problems are typically solved with the help of robust estimators. M-Estimators [60, 50] reduce the effect of the outliers by applying weighting function, reducing the problem to weighted least-squares estimation. M-Estimators require a good initial guess and works best for low presence of outliers.

RANSAC [19] applies a hypothesise-and-test framework on small, randomly selected sets of correspondences. For the model hypothesis generation a small subset of the data is used. The validity of such hypothesis is evaluated on the rest of the data and the hypothesis with the highest number of inlier data is stored to be challenged by next hypothesis. RANSAC terminates when it is confident that a better solution is unlikely [11], returning initial pose estimate and the correspondence set supporting the hypothesis.

The modification of RANSAC - *MLESAC* [53] evaluates the quality of the consensus set by computing its likelihood, improving the accuracy through better hypothesis assessment. The locally optimised (LO) RANSAC [12] performs an optimisation of the solution using inlying data to further improve the estimate accuracy. Biased sampling [10] steers the hypothesis generation towards samples with a better likelihood of being inliers (as indicated by the correspondence ranking). WaldSAC [11] allows the rejection of poor hypotheses without testing the entire correspondence set, and therefore, provides significant computational savings.

4.5 STRUCTURE TRIANGULATION

Assuming known camera poses, the 3D points corresponding to the point pair computed by matching algorithm can be estimated by triangulation. The aim of triangulation algorithm is to find the intersection of the lines defined by the camera centres of projection C, C' and image coordinates ${}^{\{c\}}\hat{m}, {}^{\{c'\}}\hat{m}$ of 3D point (Figure 7). In real world scenarios, due to the presence of the noise, the lines in 3D space will not usually intersect. Therefore multiple methods such as *mid-point algorithm* [4], Direct Linear Transform (DLT) [27] or *optimal triangulation* [26] have been presented to find the closest point to both lines. The disadvantage of the *mid-point* and *dlt* methods is that the reconstruction is not invariant to affine nor projective transformation because perpendicularity is not preserved under those transformations.

Given the corresponding pair ${}^{\{c\}}\hat{m}, {}^{\{c'\}}\hat{m}$, the key idea of the optimal triangulation algorithm (Figure 7) is to find a pair of points ${}^{\{c\}}\dot{m}, {}^{\{c'\}}\dot{m}$ that best satisfies the epipolar constraint ${}^{\{c'\}}\dot{m}^{\top} E {}^{\{c\}}\dot{m} = 0$. The points satisfying epipolar constraint must lie on the corresponding epipolar lines, e.g. the point ${}^{\{c'\}}\dot{m}$ lies on the epipolar line

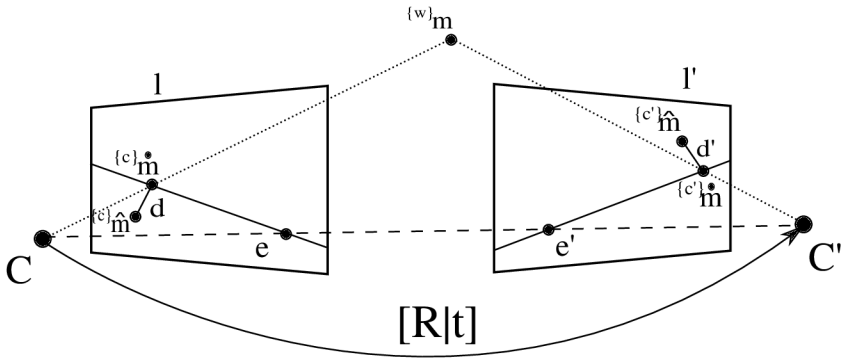


Figure 7: Optimal triangulation problem. The optimal image points ${}^{\{c\}}\dot{m}, {}^{\{c'\}}\dot{m}$ lie on the corresponding epipolar lines, closest to the measured points ${}^{\{c\}}\hat{m}, {}^{\{c'\}}\hat{m}$.

$l = E^{\top\{c'\}}\dot{m}$ and vice versa. At the same time these points should lie as close as possible to the original points ${}^{\{c\}}\hat{m}, {}^{\{c'\}}\hat{m}$. Therefore we seek to minimize:

$$d({}^{\{c\}}\hat{m}, {}^{\{c\}}\dot{m})^2 + d({}^{\{c'\}}\hat{m}, {}^{\{c'\}}\dot{m})^2 , \tag{26}$$

where the function $d({}^{\{c\}}\hat{m}, {}^{\{c\}}\dot{m})$ computes distance between parameter points. Solution to this triangulation problem can be found using iterative minimization methods or by applying non-iterative polynomial method presented in [26]. The advantage of the optimal triangulation is the affine and projective invariance.

4.6 Bundle Adjustment (BA)

The sensor measurements inherently contain noise which propagates to the estimation of sensor poses and computation of the 3D structure. Multiple measurements of the same variable allow to find optimal configuration of sensor poses and 3D points that minimises the measurement error. This refinement process is usually performed as a final step of reconstruction pipeline by applying optimisation algorithm. The measurement error functions are generally non-linear, so non-linear approaches have to be used to find the solution.

4.6.1 *Graph Representation*

We model the static environment and parametrise it as positions of the structure points together with the poses and parameters of sensors by state variables $\theta = [\theta_1 \dots \theta_n]$. The sensors observe the environment indirectly by measurements $\mathbf{z} = [z_1 \dots z_m]$.

For simple and flexible representation highlighting the structure of such a complex optimisation problem, we adopt a *graph* representation. Graph model is a graph containing *vertices* defining the system variables, such as sensor or point positions, connected by *edges*, representing spatial constraints between the variables derived from measurements or prior knowledge. The cardinality of the factors define how many variables the edge connects e.g., unary factors define constraints for a single variable, binary relate two or ternary three variables of the system.

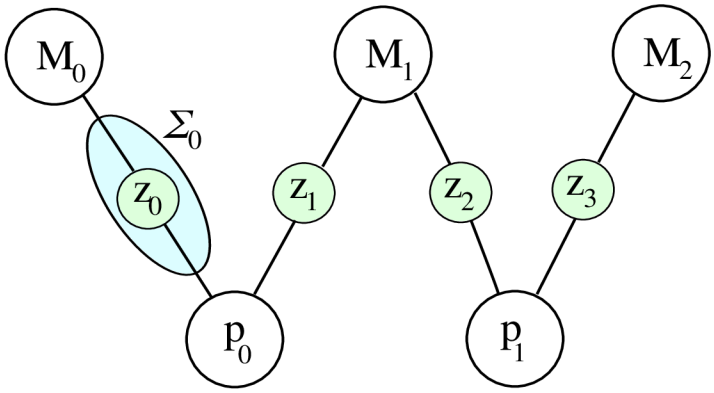


Figure 8: Graph representation of two sensors p_0, p_1 observing points M_0, M_1, M_2 with measurements z_k with covariances Σ_k (for simplicity only covariance of measurement z_0 is shown).

The goal of the BA is to obtain the *Maximum Likelihood Estimation* (MLE) of a set of variables θ, containing the state variables e.g., sensor poses, environment information, given the set of relative measurements $\mathbf{z}$:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ P(\theta \mid \mathbf{z}) = \underset{\theta}{\operatorname{argmin}}\ \left(-\log(P(\theta \mid \mathbf{z}))\right). \tag{27}$$

Due to the sensor noise, the measurements are also affected by noise:

$$z_k = h(\theta_{ik}, \theta_{jk}) - v_k\,, \tag{28}$$

where the sensor model function $h(\theta_{ik}, \theta_{jk})$ computes zero noise measurement according to the actual configuration of variables θ_{ik}, θ_{jk} and v_k is normally distributed zero-mean noise with covariance Σ_k:

$$P(z_k \mid \theta_{ik}, \theta_{jk}) \propto \exp\left(-\frac{1}{2} \parallel z_k - h(\theta_{ik}, \theta_{jk}) \parallel^2_{\Sigma_k}\right). \tag{29}$$

Finding the MLE from (27) is done by solving the following non-linear least squares problem:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{k=1}^{m} \left\| z_k - h(\theta_{ik}, \theta_{jk}) \right\|^2_{\Sigma_k}\right). \tag{30}$$

4.6.2 *Non-linear Solving*

To find the solution of the NLS, iterative methods such as Gauss-Newton (GN) or Levenberg-Marquard (LM) can be applied. These iterative approaches start with an initial configuration point θ^0 and, at each step, a correction δ towards the solution is computed. For small $\|\delta\|$, a Taylor series expansion leads to linear approximations in the neighbourhood of θ^0

$$\tilde{\mathbf{e}}(\theta^0 + \delta) \approx \mathbf{e}(\theta^0) + J\delta\,, \tag{31}$$

where $\mathbf{e} = [e_1, \ldots, e_m]^\top$ is the set of all nonlinear errors, called *residuals*, between the estimated and the actual measurement:

$$e_k(z_k, \theta) = z_k - h_k(\theta_{i_k}, \theta_{j_k})\,, \tag{32}$$

and furthermore J is the Jacobian matrix which gathers the derivatives of the components of $\mathbf{e}$ with respect to the state. Thus, at each iteration q, a linear LS problem is solved:

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \frac{1}{2} \|A\,\delta - \mathbf{b}\|^2 \,, \tag{33}$$

where $A = \boldsymbol{\Sigma}^{-\top\backslash 2} J(\theta^q)$ is the system matrix, $\mathbf{b} = -\mathbf{e}(\theta^q)$ the right hand side (r.h.s.) and $\delta = (\theta - \theta^q)$ the correction to be calculated [17]. The the minimum is attained where the first derivative equals zero:

$$A^\top A\,\delta - A^\top \mathbf{b} = 0 \quad \text{or} \quad \Lambda\delta - \boldsymbol{\eta} = 0\,, \tag{34}$$

with $\Lambda = A^\top A$, the square symmetric system matrix, called the *information matrix* and $\boldsymbol{\eta} = A^\top \mathbf{b}$, the right hand side. This is commonly referred to as the *normal equation*.

4.6.3 *Linear Solving*

The linearised version of the problem introduced above can be efficiently solved using sparse direct optimization methods, either performing Cholesky or QR factorizations, followed by backsubstitution. *Cholesky factorisation* yields $\Lambda = R^\top R$, where $R^\top$ is the *Cholesky factor* and a forward and back substitutions on $R^\top \mathbf{d} = A^\top \mathbf{b}$ and $R\delta = \mathbf{d}$, first recovers $\mathbf{d}$ and then the actual solution δ.

Alternatively, the normal equation in (34) can be skipped and *QR factorisation* can be applied directly to matrix A in (33), yielding $A = QR$, where Q is orthogonal and R is upper triangular, similar to R of Cholesky factorization up to the sign (Cholesky will always have positive entries on the diagonal). The solution δ can be directly obtained by backsubstitution in $R\delta = \mathbf{d}$ where $\mathbf{d} = R^{-\top} A^\top \mathbf{b}$. Note, that Q is not explicitly formed. instead $\mathbf{b}$ is modified during factorisation to obtain $\mathbf{d}$.

After computing δ, the new linearisation point becomes

$$\theta^{q+1} = \theta^q \oplus \delta\,, \tag{35}$$

where the operator $\oplus$ is a corresponding composition operator depending on the type of the variables.

4.6.4 *Structure of Linearised system*

The system information matrix Λ contains approximations of second derivatives of error functions e_{ij} (28). Because the error function e_{ij} is dependent only on the state variables θ_i and θ_j, it will affect the structure of the Jacobian to be non-zero only in the rows corresponding to θ_i and θ_j:

$$J_{ij} = \frac{\delta e_{ij}(\theta)}{\delta \theta} = \left[0 \ldots \frac{\delta e_{ij}(\theta_i)}{\delta \theta_i} \ldots 0 \ldots \frac{\delta e_{ij}(\theta_j)}{\delta \theta_j} \ldots 0\right] . \tag{36}$$

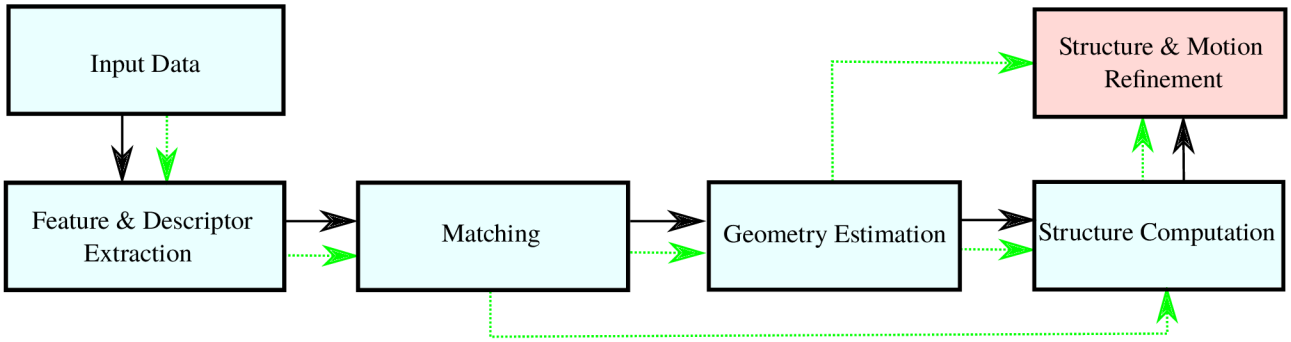


Figure 9: Pipeline of the reconstruction. Full lines represent order of processing blocks, dotted lines (green) describe data dependencies of each block. Blue blocks are part of *front-end*, estimating initial sensor poses and 3D structure. Red block represents *back-end* and is responsible for refinement of the initial estimations. (best seen in colour)

Each measurement produces one row in the Jacobian matrix with non-zero elements on the corresponding column positions. The system information matrix Λ and the coefficient vector η are computed according to:

$$\begin{aligned} \Lambda &= \sum_{<i,j>\in S} J_{ij}^T \Sigma_{ij}^{-1} J_{ij} \,, \\ \eta &= \sum_{<i,j>\in S} e_{ij}^T \Sigma_{ij}^{-1} J_{ij} \,, \end{aligned} \tag{37}$$

where S is a set of indices of variables that the measurements relate.

In practice, it is advantageous to keep the information matrix Λ as the system representation because its size depends only on the number of variables, whereas the Jacobian matrix A dimensions grow also with measurement count. Augmenting the system with a new variable involves increase of the system matrix size. Updating with corresponding measurement is an additive operation on the system matrix. Given the initial configuration set of the variables and a set of constraints, the optimal configuration of variables can be found following the MLE described in Section 4.6.2.

4.7 3D RECONSTRUCTION PIPELINE

Figure 9 illustrates the flow of the visual 3D reconstruction algorithm. The algorithm can be divided into two parts–*front-end* part responsible for initial estimation of the sensor positions and 3D structure, and *back-end* part that refines this initial estimate by applying a non-linear optimization algorithm.

1. First step of the 3D reconstruction is the data acquisition and selection of input data. The set of images should contain overlapping parts of the scene and depict a static scene.

2. The processing continues with detecting feature points in the input images and extracting their descriptors.

3. The descriptors are used by a matching algorithm to establish the correspondence pairs between sets of feature points from images, assuming the images contain an overlap. False correspondence pairs are filtered out using RANSAC algorithm and Epipolar geometry model of the cameras.

4. Once the corresponding pairs are established the pose of the camera can be computed, depending on the available information, by one of the 3D pose estimation algorithms (Section 4.3). If no 3D points are associated with the 2D feature points, which is typical for processing the first pair of cameras, the poses of the cameras is computed by decomposition of the Essential matrix. Otherwise if the 3D information is available for some of the feature points, the camera pose is estimated using PnP algorithm.

5. The estimated camera poses and corresponding pairs are used as an input for triangulation algorithm to compute the 3D structure.

Due to the noise in the measurements, the camera poses and structure points are also subject to error. Therefore it is necessary to apply BA algorithm to refine the camera poses and 3D structure. BA applies non-linear optimisation algorithms to find optimal solution for camera poses and structure positions that minimizes the reconstruction error.

5 MULTISENSOR FRONTEND

The multisensor reconstruction algorithm consists of two main parts - *multisensor front-end* and *multisensor back-end*. The multisensor front-end is responsible for processing the data from sensors and estimation of the positions and rotations of sensors in the scene, the spatial relations between them and initial computation of 3D structure. The multisensor back-end builds internal representation of the system and further refines the *front-end* estimation in a process called *optimisation* (Chapter 6). The front-end processing follows the reconstruction pipeline (Figure 9) - feature and descriptors extraction from data, matching, geometry estimation and 3D structure triangulation. In this chapter we describe specific approaches applied in multisensor front-end.

5.1 FEATURE DETECTION AND DESCRIPTOR EXTRACTION IN DATA

The relations between the sensors are estimated from a sparse set of corresponding data points. Using sparse sets of correspondences is computationally efficient and reliable for wide baseline registration. Finding the correspondences between two sparse sets of feature points is based on matching algorithms which compare the descriptors of the feature points and according to a similarity function choose the point pairs with highest scores. When working under wide baseline, the features corresponding to the same 3D point can visually differ due to the projective transformations of camera models. To cope with the visual difference, robust feature descriptors and matching methods have to be utilized to detect corresponding image points.

One of the two image pre-processing algorithms can be applied - projecting the spherical image onto a cube [18], creating six images with reduced spherical distortion and using them for descriptor extraction, or a projection of the spherical image around the feature point to plane tangent to sphere [9]. Comparison of the matching quality of different methods is described further in this chapter.

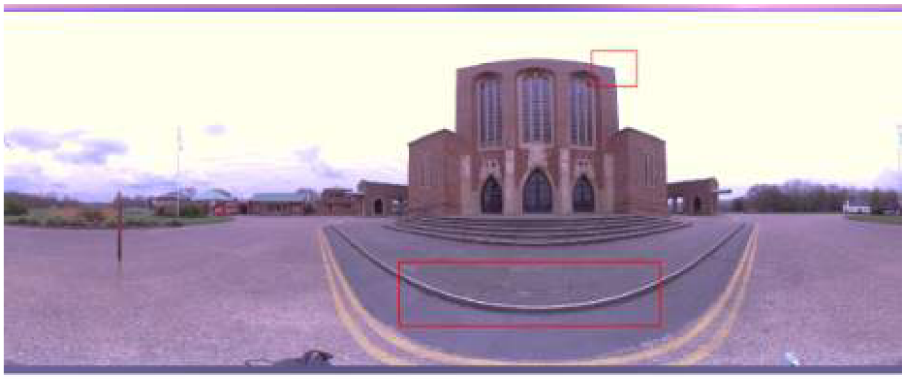


Figure 10: Distortion of the lines in longitude-latitude image.

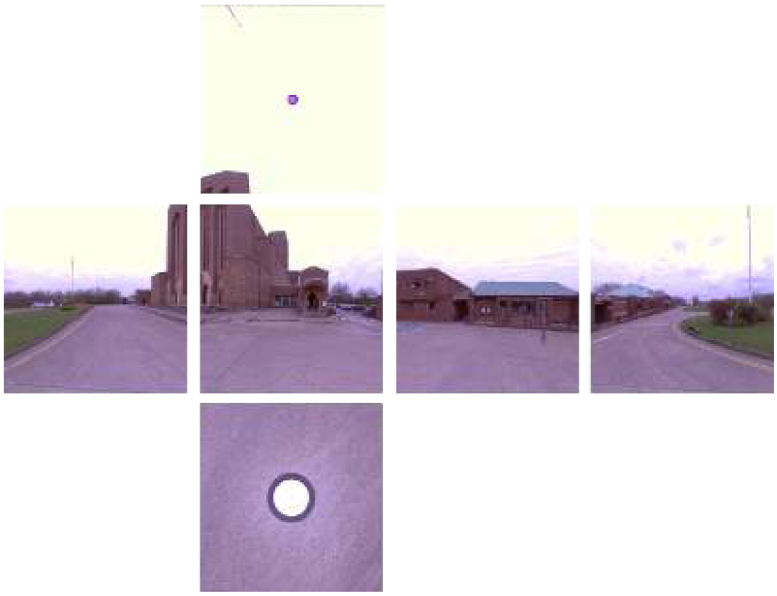


Figure 11: Six cubic images generated from spherical image by projecting the data onto six sides of a cube.

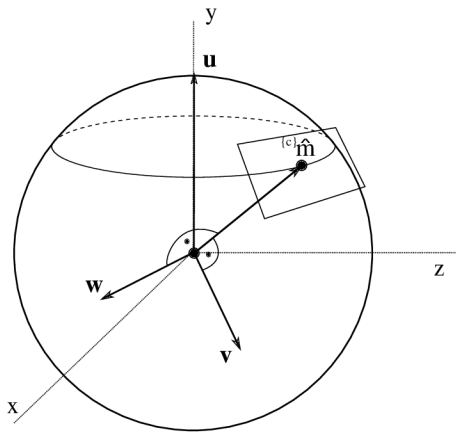


Figure 12: Tangential space.

Cubic Projection

By projecting the spherical image to the six sides of a unit cube co-centric with the sphere, it is possible to create six planar images with reduced distortion present in longitude-latitude image [18]. Using these six cubic images (Figure 11), standard algorithms for processing of projective images can be applied. Disadvantage of this method is that partitioning the image to six images causes that the descriptors of feature points detected near the borders of the image lose some information.

Tangential Projection

The reduction of the spherical distortion as well as preservation the continuity of the spherical image along left and right border can be achieved by projecting the spherical image onto a plane tangent to the sphere at the feature point. This approach extracts a patch around the feature point and performs the descriptor extraction on this image patch.

Figure 13: Correction of an extracted patch - From spherical latitude-longitude image (left) the corrected patch (right bottom) is computed using tangential projection.

The patch is extracted around the feature point, in a coordinate system of a plane tangent to the sphere at the feature point. The basis of the coordinate frame are determined as shown in Figure 12. The coordinates of the feature point ${}^{\{c\}}\hat{\mathfrak{m}}$ are computed using (10). Vector $\mathbf{u} = [0, 1, 0]^\top$ is chosen to correspond with the direction of the $y - axis$ of the spherical camera. The vectors $\mathbf{v}, \mathbf{w}$ are computed to form the orthogonal basis for the local coordinate system around feature point ${}^{\{c\}}\hat{\mathfrak{m}}$ using equations:

$$\begin{aligned} \mathbf{w} &= {}^{\{c\}}\hat{\mathfrak{m}} \times \mathbf{u}\,, \\ \mathbf{v} &= \mathbf{w} \times {}^{\{c\}}\hat{\mathfrak{m}}\,, \end{aligned} \tag{38}$$

therefore the corners of the tangent patch can be computed as:

$$\mathbf{a}_i = {}^{\{c\}}\hat{\mathfrak{m}} + \left[\pm\lambda \frac{\mathbf{v}}{2\,\|\mathbf{v}\|}, \pm\lambda \frac{\mathbf{w}}{2\,\|\mathbf{w}\|} \right], \tag{39}$$

where λ is a scale that defines the size of the patch N in pixel units, the scale can be computed from the knowledge of the pixel width M of the source longitude-latitude image:

$$\begin{aligned} \alpha &= 2\,\pi \frac{N}{M}\,, \\ \lambda &= 2 \tan\left(\frac{\alpha}{2}\right). \end{aligned} \tag{40}$$

By applying the inverse transformation from points on the tangent patch to the spherical image, the image can be sampled and colour information of the patch pixels computed (Figure 13).

5.2 MULTISENSOR REGISTRATION

In a multisensor scenario, where the image data is captured by different types of sensors, it is desirable to process all available information to create a 3D model of a scene and to use the relations between all sensors to achieve better accuracy and coverage of the scene. We have defined the epipolar geometry in Section 4.2 and the relations and geometry estimation between planar images in Section 4.3. In this section we will analogously describe the relations between different sensors - two spherical cameras,

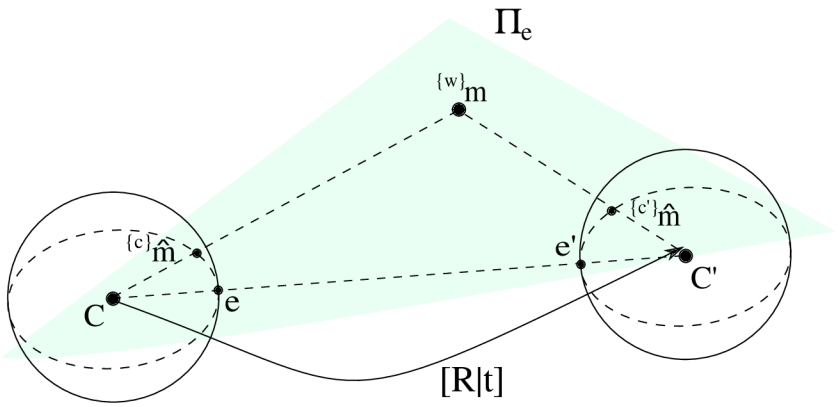


Figure 14: Epipolar geometry between two spherical cameras.

spherical and planar camera and CLIDAR scan and spherical camera. These sensors are often used for large-scale scene reconstruction, each with advantages and disadvantages. Monocular cameras can capture small details and obstructed parts of scene but cover small field of view, whereas spherical cameras and CLIDAR devices cover large parts of the scene but may not cover all details.

5.2.1 *Epipolar geometry of Spherical Camera*

Compared to pinhole camera projection, spherical projection is geometrically equivalent, but in the case of spherical camera, the scene is projected onto a unit sphere instead of projective plane [41]. The epipolar geometry is valid also between two spherical cameras, if the data normalization to unit vectors is performed. This normalization transforms the pixel coordinates, or latitude-longitude coordinates to a unit vector on a sphere according to (10) and (11). The following epipolar relations are defined assuming normalized coordinates of the images of 3D point ${}^{\{w\}}\mathbf{m}$ - ${}^{\{c\}}\hat{\mathbf{m}}$, ${}^{\{c'\}}\hat{\mathbf{m}}$ which together with the camera centres $\mathbf{C}, \mathbf{C}'$ define the epipolar plane.

In Figure 14, we can observe that the 3D point ${}^{\{w\}}\mathbf{m}$ is projected into spherical imaging surfaces, creating point images ${}^{\{c\}}\hat{\mathbf{m}}$ and ${}^{\{c'\}}\hat{\mathbf{m}}$, and together with camera centres $\mathbf{C}, \mathbf{C}'$ are coplanar. The epipolar plane Π_e intersects the spherical surfaces in epipolar circles with their centres in the camera projection centre. The line coinciding with camera centres $\mathbf{C}, \mathbf{C}'$ intersects the spherical surfaces in epipoles $\mathbf{e}, \mathbf{e}'$. If the points ${}^{\{c\}}\hat{\mathbf{m}}$ and ${}^{\{c'\}}\hat{\mathbf{m}}$ are corresponding points in this stereo system, then essential matrix relates them by:

$$ {}^{\{c'\}}\hat{\mathbf{m}}^{\top}\mathbf{E}\,{}^{\{c\}}\hat{\mathbf{m}} = 0\,. \tag{41} $$

Note that according to (16), the first part, $\mathbf{n}' = {}^{\{c'\}}\hat{\mathbf{m}}^{\top}\mathbf{E} = {}^{\{c'\}}\hat{\mathbf{m}}^{\top}[\mathbf{t}]_{\times}\mathbf{R}$, creates a vector perpendicular to translation vector $[\mathbf{t}]_{\times}\mathbf{R}$ between camera centres $\mathbf{C}, \mathbf{C}'$ and to vector ${}^{\{c'\}}\hat{\mathbf{m}}$, therefore defining a normal to the epipolar plane Π_e instead of general representation of a line as in case of pinhole cameras. The inner product of this normal vector $\mathbf{n}'$ and vector ${}^{\{c\}}\hat{\mathbf{m}}$ is equal to zero:

$$ \mathbf{n}'\,{}^{\{c\}}\hat{\mathbf{m}} = 0\,, \tag{42} $$

i.e. the point ${}^{\{c\}}\hat{\mathfrak{m}}$ lies in the epipolar plane Π_e. Analogously this relation is valid for normal vector $\mathfrak{n} = \mathrm{E}\,{}^{\{c\}}\hat{\mathfrak{m}}$ and vector ${}^{\{c'\}}\hat{\mathfrak{m}}^{\top}$.

5.2.2 *Spherical - Spherical Camera Registration*

Accurate registration of the spherical images is an important step in the multisensor 3D reconstruction process. Spherical images, compared to traditional cameras, capture large portion of a scene and therefore only few stereo image pairs are needed to reconstruct whole scene. Each spherical stereo pair yields a 3D point cloud model of a scene with respect to centre of the stereo spherical camera. To acquire consistent model of a entire scene, these models have to be correctly aligned using one of the alignment methods.

Two methods can be applied for the registration of stereo spherical image pairs - ICP or 3D pose alignment with correspondence estimation. The ICP registration uses 3D point cloud data from each sensor and iteratively finds the alignment of the point clouds that minimizes distance between closest 3D points. This approach requires good initialisation and generally larger amount of 3D points, especially when registering data captured with wide baseline. Another disadvantage is the computational complexity of ICP methods when using large amount of 3D points.

The 3D pose alignment with correspondence estimation approach estimates the relative transformation between sensors by guided matching with the geometry described in Section 4.3.1 as a model. The descriptors from the 2D features are assigned to their corresponding 3D points for each sensor, and the matching is performed between the 3D points. The feature matching stage seeks for nearest neighbours, by comparing the associated descriptors. The correspondences are ranked by the MR-Rayleigh metric [55]. However, the 3D reconstruction framework often operates under wide-baseline conditions, which significantly reduces the number of viable matchings. Therefore, the implementation often resorts to a compromise between ambiguity and quantity, and considers the multiple nearest neighbours, instead of the best. Each candidate is verified for *reciprocity*, *i.e.* whether the points are in each other's neighbourhoods. Excessively ambiguous matches are rejected by truncating the neighbourhoods so that, the ratio of the similarity scores for the worst candidate within the neighbourhood and the best candidate outside is above a threshold.

In our reconstruction pipeline, we prefer the latter method because the pose estimation using only sparse subset of corresponding 3D points followed by refinement achieves similar accuracy results to ICP method but with better time efficiency. The comparison of the methods is shown in Section 7.1.

In the case of registration of monocular spherical image, the PnP algorithm (Section 4.3.3) can be applied to find the relative transformation using the 2D-2D correspondences between spherical images to create 3D-2D correspondences. Assuming that we use the normalized unit vectors to represent the points correspondences and that at least four correspondences are available to estimate the pose of the new registered spherical camera.

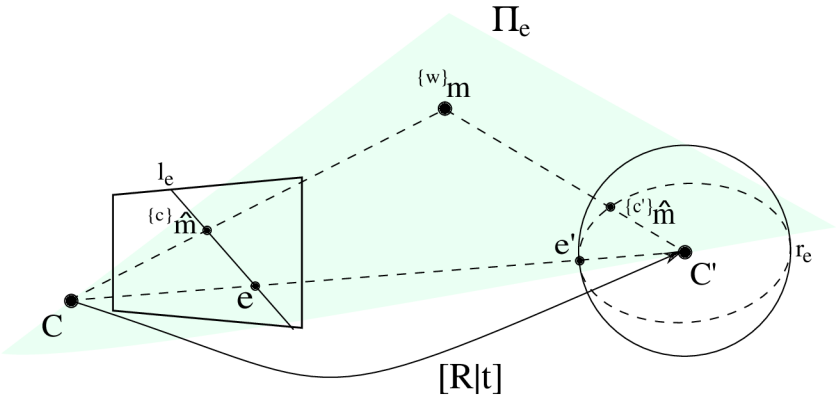


Figure 15: Epipolar geometry between spherical and planar camera.

5.2.3 *Spherical - Planar Camera Registration*

Although the 3D structure of the environment reconstructed from stereo spherical image pairs provides dense scene structure it may contain noise and inaccuracy due to the mismatches during disparity map estimation caused by insufficient illumination or lack of texture in the parts of scene. The information from planar images can recreate more details of the scene or improve the accuracy of reconstruction by estimating the structure from multiple registered planar cameras. Also the images from hand-held camera are easy to obtain to cover the areas obscured by objects in the scene.

The registration of planar and spherical cameras is based on visual correspondences. The camera models of the spherical and monocular cameras are both projective models, but with different projection surfaces. Due to the fact that the spherical cameras capture complete scene around the camera, the overlap between spherical and monocular image is usually present but small in the spherical image. This can lead to small number of corresponding points and a large number of outliers, therefore a robust algorithm is required to determine the relative transformations between the cameras. Also the distortion in the longitude-latitude images has to be taken into account (Section 5.1).

The epipolar relations between monocular planar image and a spherical image projected onto unit cube has been researched in [8]. We define the relations with the spherical image in its spherical form, because it is a convenient format for internal representation directly produced by industrial cameras.

Following the notation of Figure 15, we assume geometry of planar and spherical camera, where ${}^{\{c\}}\hat{\mathfrak{m}}$ is a vector of image point in planar camera imaging surface and ${}^{\{c'\}}\hat{\mathfrak{m}}$ a vector of image point on unit sphere of imaging surface of spherical camera. The camera centres C, C', 3D point ${}^{\{w\}}\mathfrak{m}$ and its images define epipolar plane Π_e which intersects the projection surface of the camera in epipolar line l_e and the projection surface of the spherical camera in epipolar circle r_e. Assuming known essential matrix E, (41) will be valid also for this scenario, because the $l_e = {}^{\{c'\}}\hat{\mathfrak{m}}^\top E$ defines epipolar line in the planar image and the image ${}^{\{c\}}\hat{\mathfrak{m}}$ lies on the line, as well as equation $n = {}^{\{c\}}\hat{\mathfrak{m}}E^\top$ defines normal of a epipolar plane which the point ${}^{\{c'\}}\hat{\mathfrak{m}}$ contains.

a) b)

Figure 16: Correspondences estimation: a) Guided matching using spherical-planar epipolar constraint. b) Guided matching using spherical-planar constraint and 3D-2D registration scheme described in Section 4.3.3.

In the Figure 16, the registration of the longitude-latitude and planar image is shown. The guided matching algorithm applying the epipolar geometry described in this section finds set of corresponding matches between the images, but still some outliers are present because the spherical-planar epipolar constraint restricts the corresponding point to lie on epipolar plane or line and therefore any point lying on those will satisfy the constraint. Therefore these matches are further filtered using the 3D-2D registration model (Section 4.3.3) to obtain reliable set of corresponding points and relative transformation between the sensors.

5.2.4 *CLIDAR Registration*

CLIDAR scans provide accurate dense 3D structure of the scene in the form of point cloud with assigned colour. Often the reconstruction using only few CLIDAR scans is sufficient for many applications, but in a large-scale scenario it is advantageous to extend the 3D model with data from other sensors such as spherical cameras or handheld cameras to achieve better range, more detailed reconstruction or to cover obstructed parts of the scene. For this purpose the relative transformations between sensors have to be estimated.

The CLIDAR devices such as FARO[1] are composed of multiple sensors, a range measuring laser scanner and camera capable of capturing colour information. The precise calibration allows for mapping between 3D points and colour information. The devices also provide tools to extract the longitude-latitude image from the colour information captured by camera and the 3D point cloud provides depth for each element of longitude-latitude image. So this data is equivalent to the data from stereo spherical image pair and can be used for the estimation of relative pose of sensors.

In the case where the longitude-latitude image is not available from CLIDAR device and only the coloured 3D point cloud is provided, coloured 3D point cloud can be transformed to the from of spherical (and depth) image by projecting the 3D points

1 www.faro.com

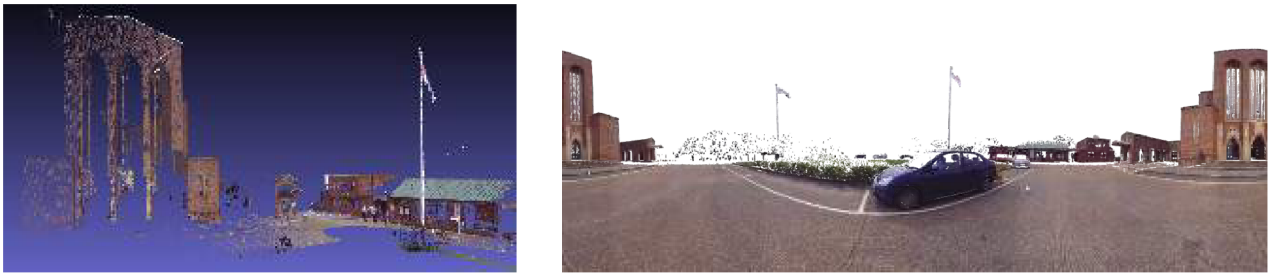


Figure 17: Coloured 3D point cloud from CLIDAR device (top). Generated longitude-latitude image from point cloud data (bottom).

onto unit sphere with the centre in the frame origin of the point cloud using (8). For each such projected 3D point the pixel position is found by computing the longitude-latitude coordinates and applying (11). The source 3D point cloud and a longitude-latitude image created from CLIDAR scan from *Cathedral* dataset using this procedure are shown in the Figure 17.

The generated longitude-latitude image and its corresponding depth information can be used for registration either with other longitude-latitude images (Section 5.2.2) or with monocular planar images (Section 5.2.3).

5.3 EVALUATION OF THE FRONT-END APPLICATION

We evaluate the quality of correspondence estimation between two images and accuracy of registration with respect to the used descriptor type (SIFT, KAZE) and a method of image distortion correction. We compare the number, quality of matches and the accuracy of image registration using the *cubic projection* method and *tangent projection* method compared to the basic method - descriptor extraction directly from longitude-latitude image. Note that the evaluation in this section involves poses estimated by front-end application, without system optimisation.

5.3.1 *Spherical-Spherical image registration*

To evaluate the spherical-to-spherical image registration, we use the *Studio* dataset spherical images which contain *ground truth* measurements of the distances between the centres of spherical camera positions as well as distances to distinctive points in the scene (Table 2). For each method *(longitude-latitude image, cubic images, tangent space)* and descriptor type (SIFT, KAZE), we perform the registration of spherical images, and measure the number of valid correspondence matches (using RANSAC with geometry estimation constraint) used for the estimation of the relative position, and compute the error in the measured distances between spherical cameras and known *ground truth* information. To achieve the fair comparison of descriptors, the feature point set was extracted individually and the descriptors (SIFT, KAZE) were extracted for those feature points. We were not able to apply this to the ASIFT approach due to the different extraction process.

Another dataset that we used for spherical registration experiments is the *Synthetic* dataset, containing spherical images generated from CLIDAR data (Table 2). Further

evaluation has been performed on datasets *CCSR, Atrium, and Cathedral* to compare the number of inlying matches used for relative pose estimation depending on the used descriptor extraction method in different baseline settings between capture poses ~ 3m, 6m, 23m for *Atrium, CCSR, Cathedral* respectively (Table 3).

Table 2: Correspondence pairs counts and accuracy of the registration of spherical images for every descriptor type (*d* - directly from longitude-latitude image, *c* - projection to 6 cubic images, *t* - projection of the image to tangent plane) for *Studio* and *Synthetic* dataset. Multiple numbers in each column represents measurements between consecutive spherical images, e.g. first number in *Matches* column represents number of correspondence matches between first and second longitude-latitude image.

	Studio		
	Matches	Error [mm]	Error [°]
SIFT d	2114/2064/3120	1/24/13	1.2/2.4/0.2
SIFT c	2571/2023/2868	1/28/8	1.2/2.4/0.2
SIFT t	2615/2044/3070	1/26/9	1.2/2.5/0.2
ASIFT d	4541/2987/4801	6/41/8	1.3/2.5/0.2
ASIFT c	1321/1806/3387	2/35/10	1.2/2.5/0.1
KAZE d	2426/1972/2887	1/25/10	1.2/2.5/0.1
KAZE c	2345/1930/2718	1/26/11	1.3/2.3/0.2
KAZE t	2435/1986/2945	1/21/11	1.2/2.5/0.2
	Synthetic		
	Matches	Error [mm]	Error [°]
SIFT s	1448/2300	46/87	1.7/1.3
SIFT c	1354/1982	39/79	1.7/1.2
SIFT t	1423/2235	32/80	1.7/1.3
ASIFT s	1666/2129	37/81	1.7/1.3
ASIFT c	1226/1262	36/81	1.6/1.5
KAZE s	1456/2189	55/90	1.7/1.3
KAZE c	1392/1908	55/88	1.7/1.2
KAZE t	1411/2176	52/83	1.7/1.3

Summary

The relative transformation could be estimated using all three types of descriptors with similar number of estimated correspondence pairs, see Table 2. For the registration of images from sensors with large baseline (*Cathedral*), ASIFT feature and descriptor extractor provided highest number of estimated correspondences. This is due to the extraction of the descriptors also from affine transformed longitude-latitude images and therefore achieving affine invariability. On the other hand, ASIFT detector produces very high amount of feature points which leads to more time expensive processing.

Table 3: Correspondence pairs counts of the registration of spherical images depending on the descriptor type (*d* - directly from longitude-latitude image, *c* - projection to 6 cubic images, *t* - projection of the image to tangent plane) for *Atrium, CCSR and Cathedral* datasets. Multiple numbers in each column represents measurements between consecutive spherical images, e.g. first number in *Matches* column represents number of correspondence matches between first and second longitude-latitude image.

	Atrium	CCSR	Cathedral
SIFT d	2555/1924/1838/2130	1390/1145	334/165
SIFT c	2221/1980/1780/1858	1158/964	317/161
SIFT t	2334/1949/1731/1902	1316/1082	315/261
ASIFT d	2638/1909/1698/1897	1804/1864	717/597
ASIFT c	1938/1566/1565/1802	1172/1310	564/638
KAZE d	2091/1785/1543/1445	1204/1012	274/147
KAZE c	1789/1609/1481/1427	1106/927	289/214
KAZE t	2177/1769/1608/1703	1173/1025	274/254

Comparing the feature and descriptor extraction *directly* from longitude-latitude images and extraction from six generated *cubic* images, the number of established correspondences is lower for the *cubic* method, mostly due to the image borders in six generated images removing information for descriptors compared to longitude-latitude image. The overall translation error is similar or slightly lower for all descriptor types using the *cubic* method compared to the extraction directly from longitude-latitude image. The approach utilizing *tangent* projection for descriptor extraction provided similar number or more correspondence pairs as *direct* method but resulted mostly in slightly lower translation error than the other two methods.

All methods and descriptor types proved to be feasible for the registration of stereo spherical image pairs, with *tangent* projection method achieving lowest errors in most of the datasets while maintaining high number of correspondence pairs. For the processing of datasets with long baseline (more than 15m), using ASIFT features and descriptors assures highest amount of correspondence pairs. For datasets with smaller baseline, SIFT or *KAZE* extractor provides sufficient amount of correspondence pairs with the advantage of lower computation time compared to the *ASIFT* extractor.

5.3.2 *Spherical-Planar image registration*

To create a consistent 3D reconstruction from spherical and planar images the relative poses of the sensors have to be estimated. For this task a sufficient number of corresponding features in both type of images has to be determined. Generally, the spherical images capture surrounding area on much bigger scale than the planar images which always capture only small portion of the scene, therefore the descriptors have to be scale invariant. The distortion in the longitude-latitude images also plays important role in finding correspondences. Using the best descriptor type and distortion correction method (Section 5.1) can lead to more established correspondences and therefore to more accurate pose estimation. In this section, we evaluate the accuracy

Table 4: Spherical-Planar image registration results for different types of descriptors and distortion correction methods used (*d* - directly from longitude-latitude image, *c* - projection to 6 cubic images, *t* - projection of the image to tangent plane). The values in the parenthesis represent variance.

	Synthetic			CCSR
	Matched [%]	Error [mm]	Error [°]	Matched [%]
SIFT d	100%	80(0.003)	0.521(0.089)	52%
SIFT c	100%	51(0.001)	0.368(0.019)	50%
SIFT t	100%	44(0.001)	0.380(0.020)	55%
ASIFT d	100%	67(0.002)	0.39(0.017)	60%
ASIFT c	100%	60(0.003)	0.46(0.038)	58%
KAZE d	90%	84(0.008)	0.54(0.176)	24%
KAZE c	90%	82(0.007)	0.54(0.086)	23%
KAZE t	90%	65(0.002)	0.42(0.116)	24%

of registration of planar images to the spherical image depending on the descriptor type and the method of spherical image distortion correction.

We evaluate the registration algorithm on *Synthetic* dataset, with ground truth information about poses of spherical camera and virtual planar cameras. The results in the Table 4 show percentage of correctly registered cameras and the mean pose error and variance compared to the ground truth for each descriptor type and distortion correction method.

Summary

For the *Synthetic* dataset, the registration algorithm was able to register all planar images to the spherical image using SIFT and ASIFT descriptors. *KAZE* descriptors failed to register two images from the *Synthetic* dataset for each correction method.

The ASIFT descriptors performed comparably in both combinations with *direct* extraction and *cubic* projection method, but did not achieve the accuracy of SIFT descriptors with *cubic* or *tangential* projection method.

Overall, the *cubic* projection method managed to lower the error for all types of descriptors. Furthermore, using the *tangent* projection method proved to be most accurate of the correction methods.

Regarding the *CCSR* dataset, many planar images could not be registered due to the camera capturing very small part of of the scene or ground, where not enough distinctive features could be found to establish sufficient number of correspondence pairs. Using the *KAZE* features failed for the biggest number of the *CCSR* dataset rendering this method not very suitable for processing of real world dataset. SIFT and ASIFT descriptors with *tangent* correction and *direct* method succeeded in most cases of the planar-spherical image registration. The *cubic* method failed at more images than other two methods due to the borders in six generated images leading to less information in descriptors.

6 MULTISENSOR 3D RECONSTRUCTION BACK-END

The multisensor back-end is tied to the front-end part, and its purpose is to refine the initial sensor poses and structure estimation provided by the front-end algorithm. The internal representation consists of *variables* representing the sensor poses and structure points parameters, and of *edges* derived from the measurement data. The initial configuration of the sensor and structure parameters is provided by the front-end application and it encodes the initial state of the system. Given this state, we can compute the *expectations*–predictions of the measurements. The difference between measurement expectation and actual measurement describes how well the actual configuration of system fits the measurements.

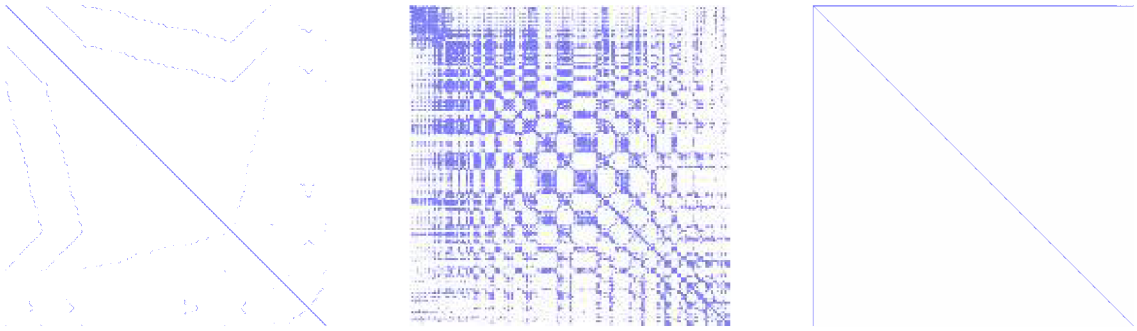
6.1 SLAM++

The joint pose and structure refinement is implemented on our open-source, non-linear graph optimisation library, called SLAM++ [45]. This C++ library is a very efficient implementation of several non-linear least squares solvers, based on fast sparse block matrix manipulation for solving the linearised problems. SLAM++ was primarily developed for efficient solving of SLAM problems in robotics, which can be formulated as a non-linear least squares problem similarly as described in Section 4.6.2, where variables represent robot trajectory and/or landmark positions, and the edges consist of relative measurements of the landmarks from robot positions. SLAM problem is mathematically equivalent to BA. The general implementation allows for definition of variables and edges for solving BA problems as well. SLAM++ produces fast, but accurate estimations, which most of the time outperforms similar state-of-the-art implementations of graph optimisation systems [34, 33, 37].

6.1.1 *Sparse block matrix structure*

Solving the BA, SLAM and SFM non-linear problems involves operations with matrices having a block structure (Section 4.6.4), because the variables usually have more than one degree of freedom (DOF). For example the pose of sensor in 3D is a variable represented by *six* parameters - *three* defining position and *three* rotation of the sensor. The associated system matrix can be interpreted as partitioned into sections corresponding to each variable, called *blocks*, which can be manipulated at once.

The dimensions of the system matrix are usually very large, but only a small number of blocks are non-zero. It is due to the fact that a measurement only affects a few variables, for example the field of view of cameras is limited so they do not observe all 3D points, i.e. not all variables are connected by measurements and therefore only a few blocks in the system matrix are non-zero. Therefore it is necessary to use sparse block structures for memory efficient storage and use sparse algorithms for matrix operations [14, 16].



a) b) c)

Figure 18: Sparse matrix structure, a) SLAM pose and landmark problem. b) BA problem - natural order b) BA problem - reordered. The non-zero blocks are in blue, the b) and c) matrices contain same amount of non-zero blocks.

In the existing state of the art implementations of sparse block matrix schemes [37, 1], the arithmetic efficiency is mostly reduced, compared to element-wise sparse matrices. That can be explained intuitively by the need for two extra nested loops for block rows and block columns that reduce the arithmetics to flow control instruction ratio and thus also computational efficiency. SLAM++ implementation elegantly solves this issue using *metaprogramming* [43, 45].

SLAM++ takes advantage of advanced metaprogramming concepts: *type lists* are employed to represent and manipulate the sets of possible block sizes. Those are used in the matrix operations to generate decision trees that handle all possible loop sizes in a given matrix. This allows for optimization using loop unrolling and vectorization at the block level. It can be easily shown that if $\log_2$ of the number of different block sizes is smaller than the average block size, the resulting code will contain less branching and thus will run faster. Note that in C++, this functionality is accessible using simple and easy to read syntax where the list of block sizes is passed to each individual matrix operation call in angled brackets.

The vectorization and loop unrolling, in addition to other algorithmic and data structure improvements lead to substantial advantages over element-wise sparse implementations, as well as over the other existing sparse block matrix implementations.

Additionally, in the process of solving a linearised system, direct methods are often employed. Some of the other existing implementations such as g2o [37], iSAM [34] or Ceres [1] use some sparse block matrix schemes internally but rely on element-wise sparse factorization [13, 15]. This requires converting the system matrices, leading to reduced efficiency. SLAM++ contains highly efficient sparse block Cholesky factorization and thus avoids this conversion.

The information matrices associated with SLAM problems are usually very sparse (about 0.1–0.25%). Since the odometry is often involved, edges exist between consecutive poses, yielding a block diagonal matrix. Additional edges in the form of loop closures and landmark observations add the off-diagonal non-zeros. In landmark SLAM, the landmarks typically form only a small fraction of the system (Figure 18, a)).

Similarly, the information matrices associated with the BA problems are also very sparse, 0.005–0.025%. Unlike landmark SLAM, however, the landmarks form the major part of the system, e.g. 92/57957 in *Guildford Cathedra*l.On the other hand, in SLAM

datasets 100/10000 in CityTrees10k or 151/6969 in Victoria Park. Additionally, the BA systems typically lack odometry and thus they form *bipartite* graphs. This is often seen as an "arrow shape" (Figure 18, c)) matrix when the sensor pose vertices are ordered before the landmark position vertices.

6.1.2 *Optimisation*

SLAM++ provides two iterative non-linear optimisation methods–Gauss-Newton (GN) and Levenberg-Marquardt (LM). For the BA problems, the LM method provides more reliable results because the initial estimation can be relatively far from the minimum and the GN easily diverges. LM is based on efficient damping strategies which allow convergence even from poor initial solutions. For that, LM solves a slightly modified variant of (34), which involves a damping factor λ:

$$(\Lambda + \lambda \bar{D})\delta = \eta \,, \tag{43}$$

where $\bar{D}$ can be either the identity matrix, $\bar{D} = I$, or the diagonal of the matrix Λ, $\bar{D} = \mathrm{diag}(\Lambda)$.

Special structure of the BA problem can be exploited to achieve more efficient solving of linearised system. Schur complement is employed to solve the linearised problem in (43). The system matrix is split in four blocks separating camera and points variables:

$$\begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \eta_p \\ \eta_m \end{bmatrix} \,. \tag{44}$$

This is a common practice in solving 3D reconstruction problems, where the camera poses are linked only through the points. It results in block diagonal A and C matrices, which can be easily inverted by inverting the individual blocks. If C is invertible, the Schur complement of the submatrix C is $A - BC^{-1}B^\top$, and is used to solve for the camera pose variables first. This is done by solving $Schur(A)\mathbf{p} = \eta_m - BC^{-1}\eta_p$, which is amenable to using both direct or iterative solvers (e.g. [40] used a dense Cholesky solver, [36] used a sparse one). The points can then be obtained by two matrix-vector products $\mathbf{m} = C^{-1}(\eta_m - B^\top \mathbf{p})$.

Performing matrix inversion and multiplication in the Schur complement form brings reduction in computational time compared to performing Cholesky factorisation of the whole system.

6.1.3 *Incremental approach*

For applications that run in real time, augmenting the system with new variables and measurements needs to be performed efficiently every step. In [29], we present an approach that takes advantage of the sparse-block structure of SLAM and BA problems, and avoids the assembly of the linearised system each iteration by incrementally updating the factorised form R of the linear system Λ and changing the linearisation point only when needed. The incremental updates are performed only on the parts of the matrix that are affected by new measurements.

Incrementally updating the system matrix

Updating the system with a new measurement is additive in information form [30]. We denote $\Omega = J_{ij}^{\top}\Sigma_{ij}^{-1}J_{ij}$ and $\omega = -J_{ij}\Sigma_{ij}^{-1/2}e_{ij}$ to be the increments in information, where J_{ij} is the Jacobian of the new measurement. In general, the measurement function $h(\cdot)$ involves only two variables, (θ_i, θ_j). For this reason and for simplicity, the following formulation will be restricted to measurements between two variables but its application remains general. The corresponding Jacobian, J, is very sparse (36) and this translates into a sparse Ω and $\boldsymbol{\omega}$. The update step only partially changes the information matrix Λ and the information vector $\boldsymbol{\eta}$. For simplicity of the notations, in the following formulations, the system matrices are split in parts that change (Λ_{11}, $\boldsymbol{\eta}_1$) and parts that remain unchanged (Λ_{00}, Λ_{10} and $\boldsymbol{\eta}_0$):

$$\tilde{\Lambda} = \begin{bmatrix} \Lambda_{00} & \Lambda_{10}^{\top} \\ \Lambda_{10} & \Lambda_{11} + \Omega \end{bmatrix} \quad \tilde{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{\eta}_0 \\ \boldsymbol{\eta}_1 + \omega \end{bmatrix} . \tag{45}$$

In the formulation above we deliberately considered that the current measurement to be integrated involves the last variable added to the system. This is the situation usually encountered in incremental SLAM problem. Note that this assumption is not necessarily needed, the formulation in (45) stays general.

As shown above, only a small part of the information matrix and the information vector are changed in the update process and the same happens with its factorized form R. The updated $\tilde{R}$ factor and the corresponding r.h.s. $\tilde{\mathbf{d}}$ can be written as:

$$\tilde{R} = \begin{bmatrix} R_{00} & R_{01} \\ 0 & \tilde{R}_{11} \end{bmatrix} \quad \tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d}_0 \\ \tilde{\mathbf{d}}_1 \end{bmatrix} . \tag{46}$$

The updated part of the Cholesky factor and the corresponding right hand side can be computed as:

$$\tilde{R}_{11} = \mathrm{chol}(R_{11}^{\top}R_{11} + \Omega) , \tag{47}$$

$$\tilde{\mathbf{d}}_1 = \tilde{R}_{11}^{\top} \setminus (\tilde{\boldsymbol{\eta}}_1 - R_{01}^{\top}\mathbf{d}_0) . \tag{48}$$

This fast incremental update approach suffers from two important problems. Firstly, without periodic reorderings, the factorized form becomes less and less sparse, slowing down the solving. Another problem is that within an iterative non-linear solver the linearization point can change every iteration, invalidating the entire factorization.

Incremental Ordering

The recently introduced data structure, the Bayes tree [33], offers the possibility to develop incremental algorithms where reordering and re-linearization are performed fluidly, without the need of periodic updates. Inspired by this strategy, SLAM++ proposes an elegant and highly efficient incremental reordering which combines the efficiency of matrix implementation [29].

The order of the rows and columns in the system matrix Λ directly influences the number of non-zero elements, also called *fill-in*, in the factorised matrix R and affects speed of updates. It has been presented [33] that reordering the variables every step

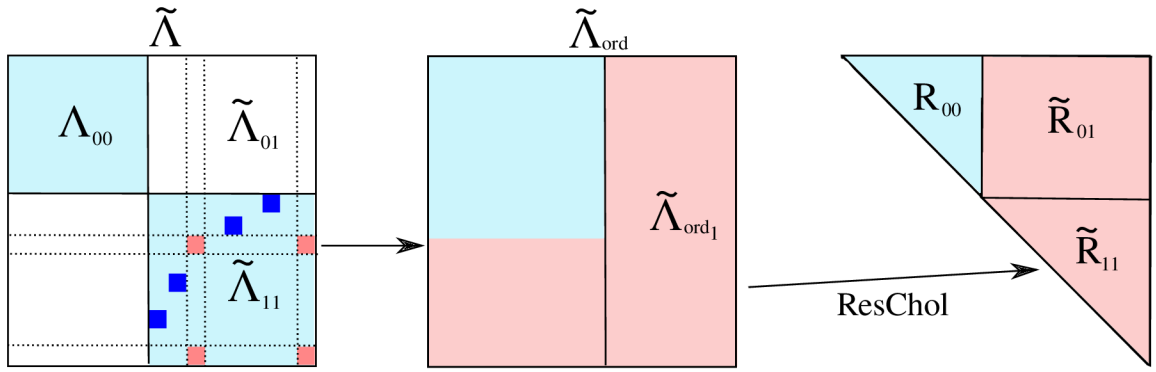


Figure 19: Data flow diagram of incremental block Cholesky factorisation. Light blue parts of matrix do not change, pink are parts that will change, red blocks represent the update and dark blue blocks non-zero elements.

significantly reduces the *fill-in* of the factorised matrix, but performing the full reordering of whole system matrix Λ would be inefficient and would essentially lead to a batch solver. Therefore the partial reordering strategy of the part of the factorised matrix affected by update is facilitated. Whole system matrix reordering and factorisation is performed only when linearisation point changes or when the updated part of factorised matrix is significantly big.

The approach in [45] shows how an efficient incremental ordering can be obtained by considering a partial ordering on a submatrix of $\tilde{\Lambda}$, which is slightly larger than $\tilde{\Lambda}_{11} = \Lambda_{11} + \Omega$ and which satisfies the conditions of being square and not having any non-zero elements above or left of it (Figure 19). This guarantees that the ordering heuristics such as approximate minimum degree [2] will have information about the non-zero entries in $\tilde{\Lambda}_{10} = \tilde{\Lambda}_{01}^{\top}$, which would otherwise cause unwanted fill-in.

The factorisation of the $\tilde{\Lambda}$ matrix can be performed using *Resumed Cholesky* algorithm implemented in SLAM++. This algorithm is able to compute factorisation by columns while only using the calculated values to the left of this column. Therefore it is possible to resume the factorisation of the right part of R while only using the reordered part of Λ and the unchanged part of the factor R_{00}. The advantage of this approach is the overall simplicity of the incremental updates to the factor, while also saving substantial time by avoiding recalculation of R_{00}.

6.1.4 *Covariance Recovery*

In some applications, the estimation of the *covariance* of the variables is necessary to assert the reconstruction or to evaluate mutual information required in active mapping. The calculation of the covariance amounts to inverting the system matrix $\Sigma = \Lambda^{-1}$. For large systems this operation is prohibitive, since it results in a fully dense matrix. Many applications require computation of covariances only for a few elements of the system matrix, usually the covariances of the diagonal elements and of the last column. For example in BA application those covariances of diagonal elements represent uncertainty of camera poses and 3D point positions.

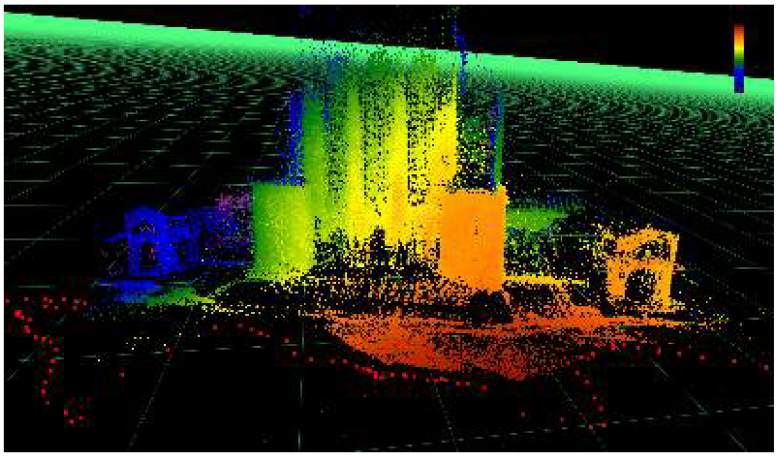


Figure 20: Covariance computed for camera poses and structure points of *Cathedral* dataset.

SLAM++ elaborates on the recursive formula for covariance estimation of [6, 23, 32] which allows computation of covariances for specific elements from factorised matrix R. To compute multiple elements of the covariance matrix, such as the whole block diagonal, these formulas are efficient only if all the intermediate results are stored.

We mentioned that most of the algorithmic speedups can be applied in case the linearisation point is kept the same. As demonstrated in (45), the contribution of new measurements is additive. In [31] we show that the same update of covariance matrix is subtractive, i.e. the new measurement adds information to the system and reduces uncertainty. The proposed scheme allows for incremental calculation of Σ on demand, whenever needed. Calculating the covariances incrementally leads to about two orders of magnitude speed-up, compared to the other state of the art implementations.

6.1.5 *SLAM++ efficiency results*

The SLAM community developed very efficient solvers due to the need of fast processing in robotics. To evaluate the SLAM++ efficiency, we compare the implementation with similar state of the art solvers such as iSAM [34], g2o [37], gtsam implementation of the iSAM2 algorithm [33] and SPA [40]. The evaluation is performed on standard simulated robotic datasets - *Manhattan* [42], *10k* [24], *City10k*, *CityTrees10k* [34], *Sphere* [24], and four real datasets - *Intel* [28], *Killian Court* [7], *Victoria park* [28] and *Parking Garage* [37].

All the tests were performed on an Intel Core i5 CPU 661 with 8 GB of RAM and running at 3.33 GHz. This is a quad-core CPU without hyperthreading and with full SSE instruction set support. During the tests, the computer was not running any time-consuming processes in the background. Each test was run ten times and the average time was calculated in order to avoid measurement errors, especially on smaller datasets.

SPA and g2o are based on similar sparse block matrix scheme which is maintained until the matrix factorisation is performed, then the switch to format to be able to use libraries CSparse [13] and CHOLMOD [15] to perform factorisation, which is a time consuming process. Those are state of the art element-wise implementations of operations on sparse matrices. SPA is optimized for 2D SLAM problem, g2o imple-

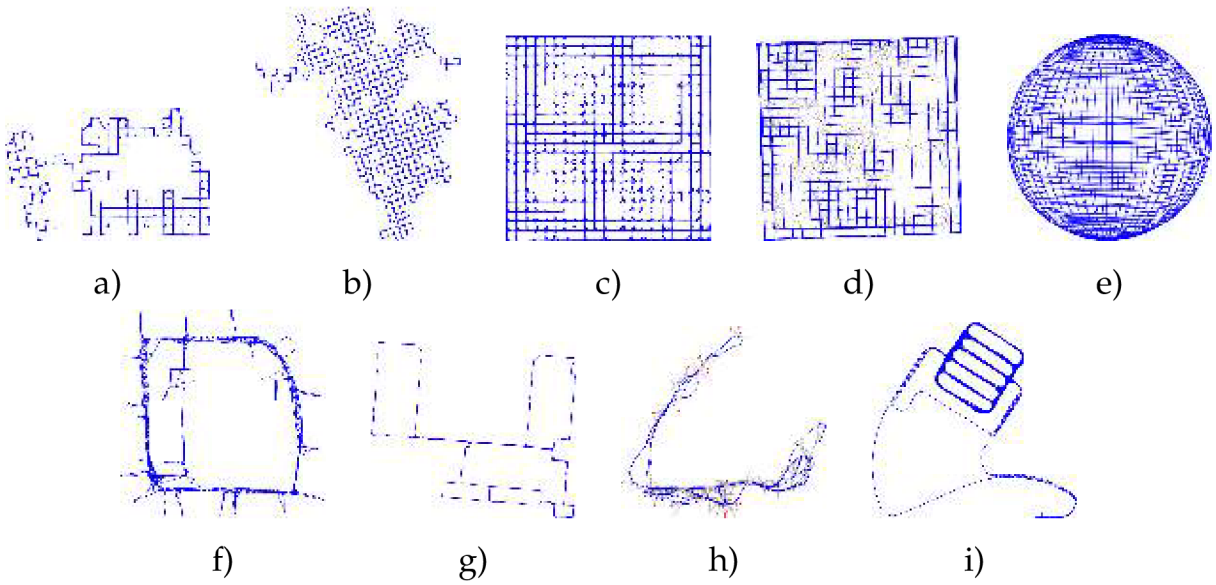


Figure 21: Illustration of SLAM datasets: a) Manhattan, b) 10k, c) City10k, d) CityTrees10k, e) Sphere, f) Intel, g) Killian Court, h) Victoria Park, i) Parking Garage

mentation is general, allowing any type of SLAM, BA or SFM problem. iSAM requires periodic batch steps to reduce the *fill-in*. iSAM2 is based on Bayes tree data structure, allows incremental reordering and fluid relinearisation.

Batch Solving

Timing results for running batch solving are shown in Table 5. The last row reports the values of χ^2 error. We denote $A-SLAM$ an algorithm that builds linear system in (33) and $\Lambda-SLAM$ an algorithm that increments information matrix in (34). The algorithm is also evaluated using factorisation from CSparse (CS) and CHOLMOD (CM) libraries. The comparison in batch mode shows a speed-up of 10% when compared to the fastest implementation which is mainly due to the proposed block matrix scheme. Note that the small speed-up is due to the fact that in this benchmark, the factorization accounts most of the solving time and the compared solvers use the same implementations.

Incremental Solving

Two incremental algorithms, first updating only the system matrix Λ, performing factorisation every step (denoted *Inc*Λ) and second keeping the factorised matrix L up to date (*IncL*), were evaluated using block Cholesky (BC) factorisation proposed in [45], factorisation from CSparse (CS) and CHOLMOD (CM) libraries.

In the Table 6, the execution times of the processing of the datasets are shown. The flags *b100* represent the frequency of batch computations (factorisation of whole system matrix Λ) each 100 vertices inserted. For the results without those flags, the nonlinear system was solved every step in order to obtain the current estimation, or only when needed in the case of our incremental algorithm. The incremental algorithm provides a solution with each new update.

	Manhattan	10K	100K	City10K	Trees10K	Intel	Killian
g2o(CS)	0.061	0.554	10.814	0.486	0.136	0.007	0.008
g2o(CH)	0.060	0.550	9.418	0.449	0.139	0.007	0.009
iSAM(CS)	1.364	2.952	24.958	1.421	0.625	0.036	0.054
A-SLAM(CS)	0.057	0.634	10.479	0.464	0.139	0.013	0.009
A-SLAM(CH)	0.061	0.698	12.009	0.531	0.147	0.008	0.010
Λ-SLAM(CS)	**0.042**	**0.485**	**9.221**	**0.420**	**0.092**	**0.005**	**0.007**
Λ-SLAM(CH)	0.047	0.580	11.056	0.456	0.109	0.006	0.008
χ^2	6112	171545	8685	31931	548	559	$5 \cdot 10^{-6}$

Table 5: Comparison of the batch solvers (CH refers to CHOLMOD and CS to CSparse library).

	Manhattan	10K	City10K	Trees10K	Sphere	Intel	Killian	Victoria	Garage
SPA	24.16	518.34	309.56	N/A	N/A	1.48	5.67	N/A	N/A
g2o	22.51	500.37	302.50	175.12	145.49	1.30	5.02	81.19	20.37
iSAM(b100)	4.83	279.93	77.57	22.93	36.22	1.29	4.21	11.92	52.22
iSAM2	4.93	91.74	60.98	32.69	31.27	0.62	1.19	16.35	3.66
IncΛ CS	8.60	287.70	202.84	19.53	216.49	0.65	1.71	23.16	17.32
IncΛ CH	10.73	236.28	181.14	24.48	71.49	0.79	2.10	28.26	23.93
IncΛ BC	7.21	242.21	188.85	**17.57**	78.37	0.51	1.24	18.71	11.34
IncL BC	**3.05**	**79.65**	**53.95**	19.31	**9.87**	**0.35**	**1.05**	**11.20**	**3.41**
error-iSAM2	6205	171600	31951	794	775	559	$8e-5$	370	1.26
error-IncL BC	6111	171919	31931	12062	727	558	$5e-5$	144	1.31

Table 6: Performance and accuracy tests on multiple datasets. The accuracy is measured as a sum of squared errors. The accuracy for landmark datasets (Trees10K, Victoria Park) are different because of different landmark parametrisation and therefore incomparable.

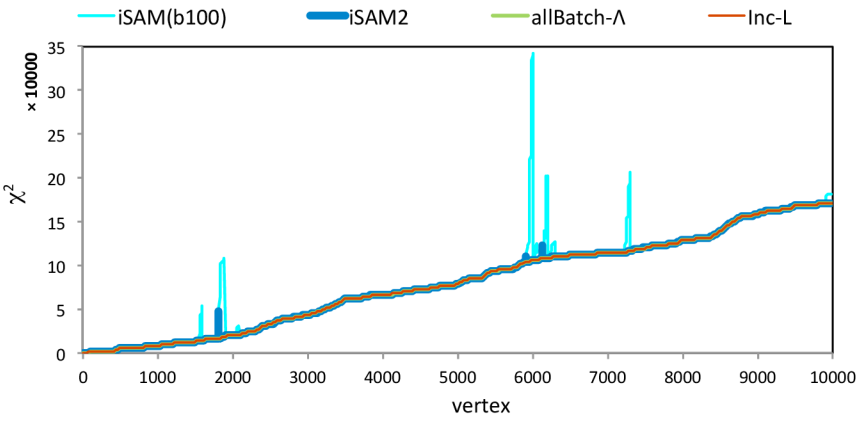


Figure 22: Quality of the estimations on *10k* dataset.

The incremental algorithm is different from the algorithms of g2o and SPA, where a batch step is performed every n variables inserted into the system and no solutions

are available in-between, so the results are comparable only to *Inc*Λ for $n = 1$. iSAM and iSAM2 provide solution every step, with iSAM requiring periodic batch step (by default every 100 steps). Keeping the same linearisation point for long time leads to error increases and decreases between those steps (seen in Figure 22).

The incremental implementation achieves the fastest results on all datasets except *CityTrees10k* dataset, which is caused by dense structure of the problem. In this case reordering is advantageous over incremental reordering. The closest results to *IncL* algorithm are from iSAM2. The difference between those algorithms is that *IncL* relinearizes affected variables only when needed.

The block Cholesky factorisation algorithm was tested on full system matrices in the incremental algorithm and compared with CSparse and CHOLMOD algorithms. The fastest results were achieved using the block Cholesky algorithm for all tested datasets.

Covariance Recovery

Table 7 shows the time performance of SLAM++ incremental covariance recovery strategy compared with g2o and iSAM implementations. The block-diagonal and the last block column of the covariance matrix are recovered at every step in all the cases. These are the only elements of the covariance matrix required for taking active decisions based on the current estimation and efficient search for data association in an online SLAM application. The SLAM++ covariance computation for BA datasets were performed in [44].

	Manhattan	10K	City10K	Trees10K	Sphere	Intel	Killian	Victoria	Garage
iSAM	206.58	6712.03	4585.15	1009.91	6051.73	6.23	19.27	310.57	237.13
g2o	18.42	5902.46	3742.66	938.97	5536.48	6.92	21.59	293.09	216.28
SLAM++	**4.37**	**179.69**	**55.87**	**30.98**	**24.64**	**0.54**	**1.43**	**13.89**	**10.77**
SLAM++ Total	13.88	388.67	219.43	60.41	105.35	1.11	2.99	37.11	27.08

Table 7: Time performance in seconds for the covariance recovery method on multiple SLAM datasets. Last row reports total processing time–solving the SLAM problem and covariance computation.

In conclusion, the proposed implementation significantly outperforms all the existing implementations due to the proposed incremental covariance update algorithm and the blockwise implementation of the recursive formula.

6.2 SYSTEM REPRESENTATION

We utilize *hyper-graph* structure to represent the optimisation problem (Section 4.6.1). SLAM++ implements variables structures to define sensors poses and points in 2D or 3D space and edge structures to impose constraints between the variables. In this section we describe the internal representation of the variable and edge structures used in mutisensor SLAM++ application.

6.2.1 *Variables*

The configuration of the system consists of variables such as sensor poses and structure points. All variables extend the implementation class CSEBaseVertexImpl which models the parameter block used for representation of vertex with specified degree of freedom. The variable classes also implement the update (35), which needs to be handled differently for each variable. For example, whereas the update of the 3D point is per-element addition, the update of 6DOF position variable is an operation on $se(3)$ which is the Lie algebra [51] of the special Euclidean group $SE(3)$.

3D Point

Code 1: Implementation of a 3D point variable.

```
1 class CVertexXYZ : public CSEBaseVertexImpl<CVertexXYZ, 3>
```

The reconstructed environment is represented by 3D points computed from sensor measurements e.g., triangulation algorithm from corresponding points between cameras or from depth information from stereo cameras or LIDAR. Due to the presence of noise in the measurements and camera positions, the computed position of the 3D points is also perturbed by noise, therefore it is necessary to define the 3D points as variables to be able to refine the structure by optimising the system. The 3D structure point is represented by a vector $M = [x, y, z]^\top \in \mathbb{R}$ describing the position of the point in the world coordinate frame.

Monocular camera

Code 2: Implementation of a monocular camera variable.

```
class CVertexCam : public CSEBaseVertexImpl<CVertexCam, 6>
protected:
    Eigen::Matrix<double, 5, 1, Eigen::DontAlign> m_v_intrinsics;
```

The camera pose consists of position and orientation. The position is defined by *three* parameters representing the position of the sensor in the world coordinate frame and the rotation is represented by *three axis-angle* parameters. The axis-angle representation, in the form of $\alpha \boldsymbol{e}$, compared to the rotation matrix representation uses only *three* quantities to describe the rotation. Unit vector $\boldsymbol{e} = [e_0, e_1, e_2]$ indicates the axis of rotation and the angle α describes magnitude of rotation.

The camera pose variable has *six* degrees of freedom and is represented by a vector $p = [x, y, z, \alpha e_0, \alpha e_1, \alpha e_2]$, element of $se(3)$, defining the rigid transformation of the camera in the world coordinate frame.

Furthermore the monocular camera is parametrised by intrinsic camera parameters - focal length $\mathbf{f}$, principal point $\mathbf{c}$ and a first order radial distortion coefficient d of the monocular camera. Having the intrinsic camera parameters as a variable allows for calibration refinement.

There are two options for modelling the intrinsic camera parameters - as a part of the monocular camera variable or as a separate variable. The former option extends the camera variable by a five parameter vector $\tau = [f_x, f_y, c_x, c_y, d]$ and then the optimisation refines the parameters specifically for this camera variable. The option involving separate intrinsic variable allows for sharing of intrinsic camera parameters between multiple cameras, optimising the separate variable linked to multiple monocular camera variables. This is achieved via *ternary* reprojection edges described in Section 6.2.2.

Intrinsic Camera Parameters

Code 3: Implementation of an intrinsic parameters variable.

```
class CVertexIntrinsics : public CSEBaseVertexImpl<CVertexIntrinsics, 5>
```

For modelling of shared camera calibration, for example, when multiple images were captured by the same camera, the variable for intrinsic parameters is introduced. Intrinsic camera parameters variable contains information about focal length $\mathbf{f}$, principal point $\mathbf{c}$ and a first order radial distortion coefficient d of the monocular camera. This variable is represented by *five* parameter vector $\tau = [f_x, f_y, c_x, c_y, d]$.

Stereo Spherical Camera and CLIDAR

Code 4: Implementation of spherical camera and LIDAR variable.

```
class CVertexSpheron : public CSEBaseVertexImpl<CVertexSpheron, 6>
```

In Section 5.2.2 we show that the CLIDAR data can be represented and processed similar to the stereo spherical cameras. Therefore the variable representation of spherical camera and CLIDAR device is the same. This variable is used to represent the 6DOF pose of these sensors in the world coordinate frame. Similar to the monocular camera variable, the position and orientation is represented by a vector $p = [x, y, z, \alpha e_0, \alpha e_1, \alpha e_2]$, element of special Euclidean group $SE(3)$, defining the rigid transformation of the sensor in the world coordinate frame.

6.2.2 *Edges*

Code 5: Implementation of a base edge type.

```
template <class CDerivedEdge, class CVertexTypeList, int
    _n_residual_dimension, int _n_storage_dimension = -1>
class CBaseEdgeImpl : public CBaseEdge
```

The measurements impose relations between variables, represented by *edges* connecting the variables involved in the measurement. Furthermore we assume independent

Gaussian measurement noise, for each measurement z_k, represented by covariance matrix Σ_k. Each edge gives rise to residual (32) and the goal of the back-end is to find the configuration of the variables θ that minimize the sum of squared residuals by solving the non-linear least squares problem (30).

The implementation class `CBaseEdgeImpl` (Code 5) is templated by list of vertex types. This edge contains dimension of the residual vector and a dimension of measurement vector. Based on the number of variables that the edge connects, we differentiate between *unary*, *binary* and *hyper-edges*.

Unary Edge

The unary edge constraints only one variable and its purpose is to provide a prior information. In the context of BA and SLAM applications, the unary edge is used to fix the position of the first camera p_0 to world coordinates. The residual of unary edge has form of:

$$e(p_0) = 0 \ominus p_0 \,, \tag{49}$$

where vector 0 defines desired fixed camera pose and operand $\ominus$ is an inverse pose composition of SE3 group.

Reprojection Edge

Code 6: Implementation of reprojection edge without shared intrinsic parameters.

```
class CEdgeP2C3D : public CBaseEdgeImpl<CEdgeP2C3D, MakeTypelist(
    CVertexCam, CVertexXYZ), 2>
```

Code 7: Implementation of reprojection edge with shared intrinsic parameters–note the definition of third vertex type `CVertexIntrinsics` that the edge connects.

```
class CEdgeP2CI3D : public CBaseEdgeImpl<CEdgeP2CI3D, MakeTypelist(
    CVertexCam, CVertexXYZ, CVertexIntrinsics), 2>
```

Reprojection constraint describes the process of projecting a 3D structure point into the 2D image. Reprojection edge can have *binary* or *ternary* cardinality. The *binary* reprojection edge (Code 6) connects camera pose variable extended with camera intrinsic parameters and a 3D point. The *ternary* reprojection edge (Code 7) connects the variables of 3D point, sensor pose and camera parameters.

This edge is established from measurements of a feature point positions in the image of a camera. The reprojection residual function is defined as a difference between the observed 2D point measurement and expected 2D position computed as a function of 3D point ${}^{\{w\}}m_i$, camera pose p_j and the vector containing camera intrinsic parameters τ_k:

$$e_{ijk}(z_{ij}, {}^{\{w\}}m_i, p_j, \tau_k) = z_{ij} - h_{\text{reprojection}}({}^{\{w\}}m_i, p_j, \tau_k) \,. \tag{50}$$

The reprojection function $h_{\text{reprojection}}$ computes the expected position of the image of 3D point in the camera projection plane using (5) in Section 4.1.

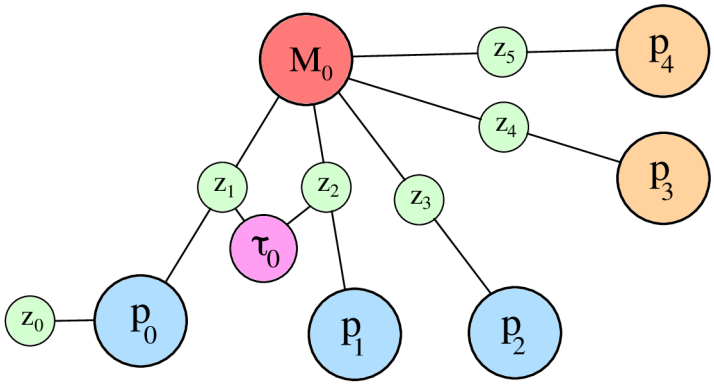


Figure 23: Graph representation of multisensor optimisation problem. Graph contains *seven* variables, *three* monocular camera variables (blue), *one* variable representing shared intrinsic camera parameters (pink), *two* variables for stereo spherical camera and CLIDAR device (orange) and *one* variable representing observed 3D point (red). Unary edge $e_0(p_0)$ defines prior measuremen

3D Point Edge

Code 8: Implementation of 3D point edge.

```
class CEdgeSpheronXYZ : public CBaseEdgeImpl<CEdgeSpheronXYZ, MakeTypelist
    (CVertexSpheron, CVertexXYZ), 3>
```

The 3D point edge defines the constraint between sensor position and a measured 3D point. This *binary* edge connects variables of 6DOF pose (spherical camera or LIDAR) and 3D point. The residual function is the displacement between position of predicted 3D point and the measurement of the point 3D position:

$$e_{ij}(z_{ij}, {}^{\{w\}}m_i, p_j) = z_{ij} - ({}^{\{w\}}m_i \ominus p_j)\,, \tag{51}$$

where the operation $\ominus$ is an inverse pose composition of SE3 group i.e., transforms the coordinates of point ${}^{\{w\}}m_i$ from world frame to the coordinate frame of sensor.

6.3 SYSTEM BUILDING

The initial configuration of the sensor poses and 3D points is provided by the front-end application using one of the pose estimation and triangulation algorithms (Section 5.2). The system integrates one by one the camera/sensor poses and corresponding 3D points observed from it. As the data are processed, the measurements between the sensors or between the sensors and 3D points are added as edges. Each edge is linearised and added to the system matrix Λ by building the update matrix Ω (45), calculated from Jacobian of the measurement function, and following the incremental strategy described in previous section. The constraints can be inserted into the system in any order. This way a large connected graph is built with edges interconnecting different variables. Figure 23 shows graph representation of simple multisensor system.

The initial configuration of the system is refined by the optimisation procedure (Section 6.1), finding the solution that minimizes the error functions of the system.

7 EXPERIMENTS AND EVALUATION

In this chapter we aim to experimentally evaluate several aspects of the multisensor 3D reconstruction application. We focus on the evaluation in terms of accuracy for different sensor combinations. First we evaluate the 3D reconstruction from stereo longitude-latitude images, which is the most challenging sensors to integrate. Then we add integration of monocular cameras and CLIDAR sensors and evaluate multisensor scenarios.

7.1 EVALUATION OF STEREO SPHERICAL IMAGE RECONSTRUCTION

We first evaluate the 3D reconstruction from spherical stereo images only. Dense registration using ICP, described in Section 4.3.2, has been successfully used in the literature for the 3D reconstruction from spherical images [35]. Therefore, ICP is used as a reference in the time and accuracy evaluations of the refinement method. We refer to the refinement by ICP method as ICP. To calculate the initial estimate of the camera poses and the 3D structure, the SURF descriptors were extracted in the longitude-latitude images using *tangential projection* to reduce the spherical distortion effect, and guided matching (Section 4.2.1) with geometry model described in Section 4.3.1 was performed to estimate the relative pose.

We use dense ICP to define a ground truth for testing the accuracy of our method in the outdoor datasets where there are no manual measurements available. For that, manually matched sparse features are used to calculate an initial estimate for the ICP registration, and it will be further referred as GT-ICP.

Accuracy evaluation of Stereo Spherical image registration

In our pipeline we can identify two sources of errors that can affect the final reconstruction, a) the error of the depth map and b) the camera pose estimation error. To analyse the accuracy of the stereo spherical registration, ground truth data were measured for all three datasets. Smaller sensor displacement and flat ground surface of the *Studio* dataset allowed for precise positioning of spherical cameras, and manual measurements of distances from the spherical camera positions to several objects in the scene as well as distances between camera poses. For the outdoor datasets, *Cathedral* and *CCSR*, the ground truth data was generated by manually matching sparse features to create an initialisation for the dense ICP (GT-ICP). For the ICP registration a standard implementation provided by the PCL library [47] was used. The *Studio* dataset contains 4 pairs of stereo longitude-latitude images with 2m and 1m distance between the spherical camera positions. The 3 stereo pairs of *CCSR* dataset was captured from positions ~ 6m apart, and 3 stereo pairs of *Cathedral* dataset share baseline ~ 23m apart.

Table 8: Depth map accuracy results: Differences between GT and measurements in the depth map. Each row represents the error between measured and ground truth distance for actual spherical camera position. Certain distances were not measured for ground truth, those cells are marked by N/A symbol.

	Error [mm]			
	Object 1.1	Object 1.2	Object 2	Object 3
P1	18	12	2	3
P2	N/A	N/A	7	8
P3	N/A	N/A	78	1
P4	17	32	13	3

The error of the depth map was evaluated for the Studio dataset by comparing the calculated depth from the dense stereo processing with the measured ground truth. In this dataset, the cameras were placed in four different positions with a measured distances in between, and distance to objects in the scene were also measured. Table 8 shows the errors between the manually measured and the estimated 3D positions. The depth map error is, in average, of 1.6 cm for the Studio dataset. We can say that is a very good depth calculation from stereo longitude-latitude images for indoor scenes, nevertheless, we should expect larger errors in the outdoor scenes.

In order to evaluate the joint camera and structure estimation, two types of errors are evaluated, a) camera pose estimation error and b) structure error. To compute the pose estimation error, the transformations between the GT-ICP and the estimated poses are calculated. The errors in translation and rotation are reported separately, by computing the norm of the translation and the angle of rotation, respectively. For each dataset, pair-wise spherical camera registrations are evaluated. The structure error is computed in Studio dataset as an average error of distances to known objects in the scene. In the case of Cathedral and CCSR datasets, the structure error is given by the average euclidean distance between two *dense* point clouds–one from GT-ICP and second from optimized solution.

Table 9 confirms our expectations that both, ICP and SLAM++ have similar accuracy, and that larger errors in pose estimation correlate with errors in structure estimation. Note that for longer baselines, the SLAM++ copes better with the errors in the initial estimation compared to ICP which requires very good initialisations. This is due to the fact that unlike ICP which relies only on matches between consecutive spherical cameras for each registration, SLAM++ also considers matches over multiple spherical images.

Time evaluation

The disadvantage of applying ICP for image registration is its processing time. The proposed approach offers much faster solutions in this direction. Table 1, bottom, shows that SLAM++ is, for all datasets, almost three orders of magnitude faster than the ICP algorithm. The good time performance stems from the fact that it optimises

Table 9: Accuracy results: Top: Structure Error. Bottom: Camera pose error evaluated separately for the rotation and translation.

Criteria	**Method**	Studio			Cathedral		CCSR	
		S1-S2	S2-S3	S3-S4	S1-S2	S2-S3	S1-S2	S2-S3
Pose err.	SLAM++ [mm]	4.6	7.9	11.2	708.7	371.4	374.9	119.3
	ICP [mm]	10.7	36.1	50.8	678.3	740.5	261.1	149.9
	SLAM++ [°]	1.14	0.57	0.89	5.48	3.91	0.81	1.66
	ICP [°]	5.03	0.81	1.38	4.85	4.83	0.52	2.71
Structure err.	SLAM++ [mm]	16.1			1120.2		488.9	
	ICP [mm]	35.4			1765.6		394.7	

Table 10: Time processing evaluation for refinement using SLAM++ and ICP.

Processing			
Feat. & desc. extract [s]	8.15	7.02	6.32
Initial estimation [s]	6.99	11.41	25.65
Refinement			
ICP [s]	146.057	366.024	995.43
SLAM++[s]	**0.120**	**0.091**	**0.134**

for a sparse set of points and from the actual efficient implementation of non-linear least squares solver SLAM++.

By analysing the processing time of each step of the pipeline in Table 1, we see that the time of optimising the camera poses is now very small compared to the other processing times in the pipeline, while using ICP, the registration time would have been the predominant time and would have constituted a bottleneck in large applications.

7.2 MULTISENSOR RECONSTRUCTION ACCURACY

We processed several multisensor datasets using the proposed multisensor 3D reconstruction pipeline. The accuracy evaluation of the reconstruction from monocular and CLIDAR sensors is performed on *Synthetic* dataset with known ground truth described in Section 3.2. The computed error is per-pose *all-to-all relative pose error* (RPE) obtained as a sum of differences between all estimated and all ground truth camera relative poses divided by number of cameras n:

$$e_{RPE} = \frac{1}{n} \sum_{ij} |p_{ij} \ominus p_{ij}^{GT}|, \tag{52}$$

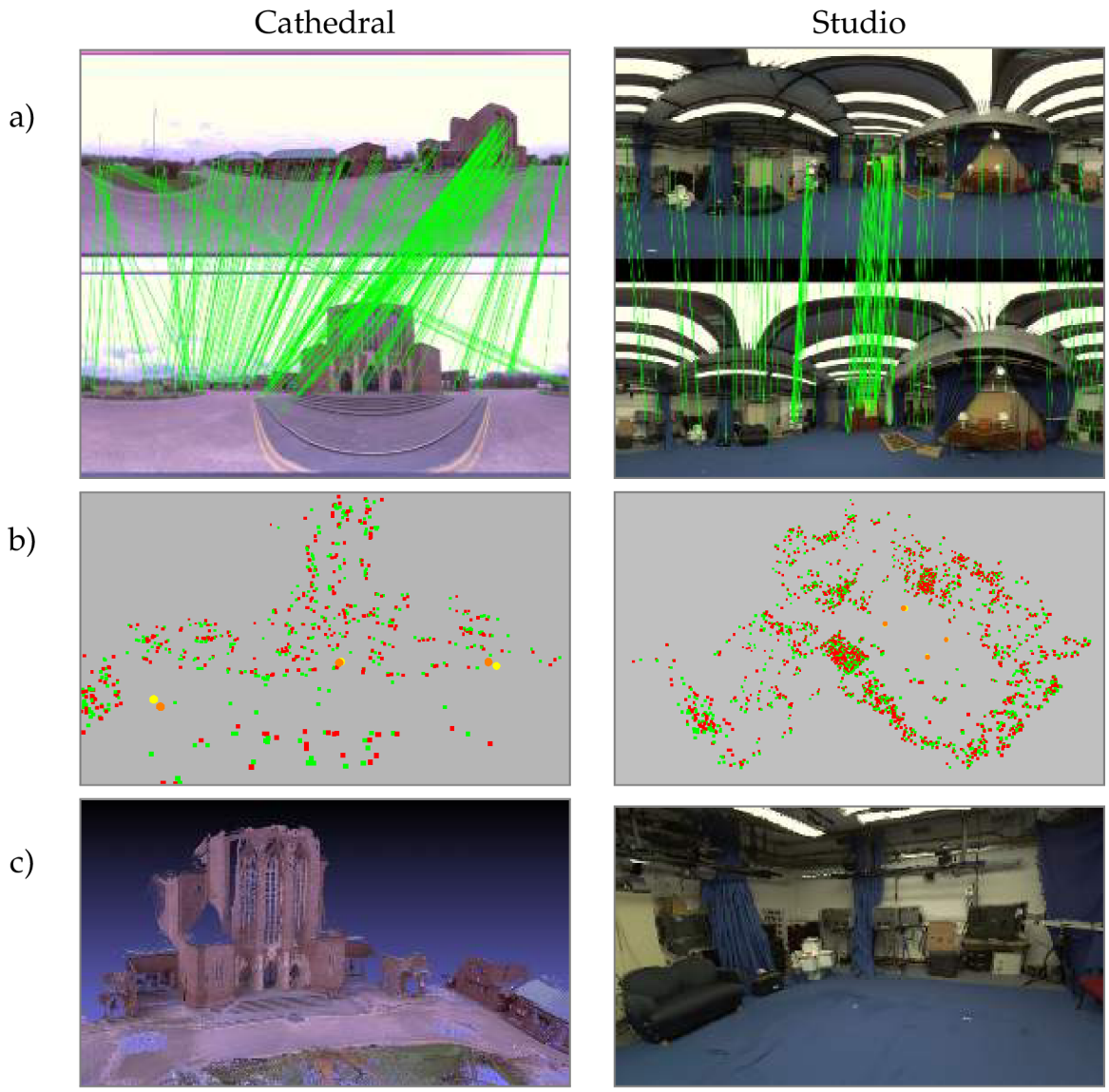


Figure 24: 3D reconstruction from stereo spherical images. a) Inliers after matching with RANSAC algorithm (for better visibility only a fraction of matches is shown for Studio dataset). Please note that the crossing lines in the left column are not outliers, the image is spherical so the left part of the image continues on the right. b) Initial 3D points (red) and camera poses (orange) and optimised 3D points (green) and camera poses (yellow). d) Final dense 3D reconstruction created by integration points from depth maps.

where the p_{ij} and p_{ij}^{GT} is a relative transformation between two estimated camera positions and *ground truth* camera positions respectively and operation $\ominus$ performs inverse pose composition. The results are also compared with the commercial software *CapturingReality*[1] for which the RPE is computed as well.

The initial sensor poses are estimated using multisensor pipeline. CLIDAR coloured 3D point cloud is transformed to the form of longitude-latitude image by process described in Section 5.2.4. SIFT features and descriptors are extracted from image data and the *tangential projection* is applied to longitude-latitude images to reduce the effect of spherical distortion. The correspondences are found using guided matching

1 www.capturingrality.com

Table 11: Per-pose all-to-all RPE error of our approach and CapturingReality software compared to ground truth of Synthetic dataset. The evaluation of *CapturingReality* in the presence of noise could not be performed due to different handling of input CLIDAR data.

		Our approach	CapturingReality
Synthetic-short	RPE [mm]	2.1	3.1
	RPE [°]	0.034	0.043
Synthetic-long	RPE [mm]	4.3	8.3
	RPE [°]	0.016	0.132
Synthetic-combined	RPE [mm]	4.3	23.9
	RPE [°]	0.034	0.312
Synthetic-short-noise	RPE [mm]	2.3	N/A
	RPE [°]	0.034	N/A
Synthetic-long-noise	RPE [mm]	4.5	N/A
	RPE [°]	0.021	N/A
Synthetic-combined-noise	RPE [mm]	6.1	N/A
	RPE [°]	0.037	N/A

Table 12: Average reprojection error in pixels of 3D reconstructions from monocular, monocular+spherical and monocular+lidar configurations.

	Monocular	Monocular + Sph	Monocular + CLIDAR	All
CCSR [px]	0.371	0.357	0.354	0.354
Cathedral [px]	0.236	0.226	0.222	0.224
Atrium [px]	0.342	0.312	–	–
Synthetic-combined [px]	0.260	–	0.259	–

(Section 4.2.1) with geometry model depending on the registered sensors. For stereo longitude-latitude images the 3D-3D registration model (Section 4.3.1) is used, and for longitude-latitude and planar image the spherical-planar epipolar geometry (Section 5.2.3) is applied.

Table 11 shows the per-pose all-to-all registration RPE error of registration of multiple scenarios of *Synthetic dataset* consisting of 3 CLIDAR scans and 10 generated planar images per scenario–containing images from virtual cameras with short baseline (*Synthetic-short*), long baseline (*Synthetic-long*) and combination of the long and short (*Synthetic-combined*). These datasets were evaluated with two different noise levels in CLIDAR depth data. First configuration uses depth data directly from CLIDAR device, which according to manufacturer, has standard deviation of depth error 2mm. For second experiment, the depth data was perturbed by a normal distributed noise with standard deviation of 150mm to evaluate the effect of depth map noise on reconstruction accuracy.

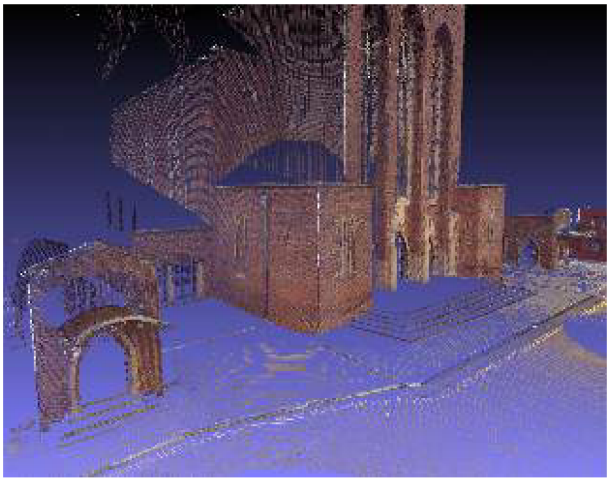
The input for both algorithms, our and *CapturingReality* consists of CLIDAR 3D point clouds and a set of synthetic planar images. Initial camera parameters were provided for both applications to assure the same initial conditions.

For short baseline scenario both algorithms achieve similar accuracy results, our approach being slightly more accurate. For long and combined baseline our approach achieves better results with accuracy of $\sim$ 4mm RPE per pose. This is because of more non-linear iterations ($\sim$ 25) of BA solver. I was not possible to specify or check for number of iterations for *CapturingReality*. Even in the presence of noise in depth data our algorithm achieves accurate results. The evaluation of *CapturingReality* in the presence of noise could not be performed due to different handling of input CLIDAR data.

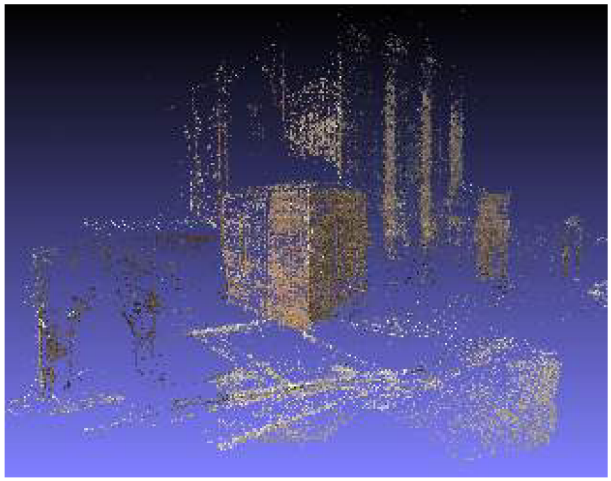
Further, we compute the reprojection error of the structure, computed as the average of differences of projected the structure points and their measured positions. Figures 27, 26 and 28 show the 3D reconstructions from different types of sensors are shown for *Cathedral*, *CCSR* and *Atrium* datasets, introduced in Section 3.2. Images *a), b), c)* show separate reconstruction for CLIDAR, monocular cameras and spherical cameras respectively. The reconstruction from spherical cameras suffers from artefacts caused by inaccuracies in disparity map. In both Figures 27 and 26, the images *d), e)* show reconstruction from spherical cameras, and monocular cameras superimposed with green colour and with colour information from the images. Image *f)* shows reconstruction using all sensors. Only sparse structure from longitude-latitude images is shown, i.e. the points for which the correspondence was established with points from other sensors. The coverage of obstructed area by structure from monocular cameras can be seen in Figure 26, f).

The table 12 displays accompanying reprojection errors for each sensor combination. In the visualizations of results (Figure 27,26 c), 28 b)) it is visible that for the spherical reconstruction the whole reprojected disparity map contains big distortions. But when this spherical data is used in the combination with monocular images, the reprojection error drops from 0.371 to 0.357 for *CCSR* and 0.236 to 0.226 for *Cathedral* compared to reprojection error of reconstruction only from monocular images. Lowest reprojection error is achieved using monocular and CLIDAR sensors.

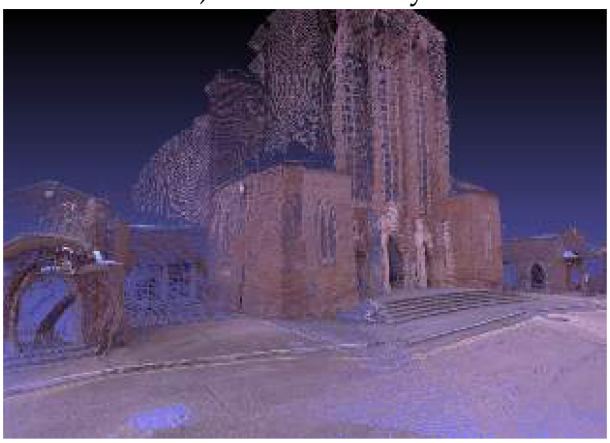
According to the evaluation of *Synthetic* dataset, the presented multisensor 3D reconstruction pipeline compared to the *CapturingReality* achieves more accurate results. The accuracy stems from the quality of established corresponding points and joint optimisation by SLAM++. The joint processing of stereo spherical and monocular data improves the reprojection error of monocular reconstruction and structure from monocular reconstruction improves the noisy stereo spherical depth map. The accurate depth data from CLIDAR allows for easy integration.



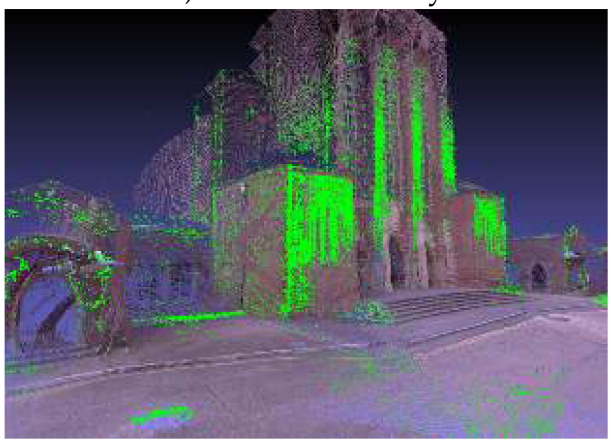
a) CLIDAR only



b) Monocular only

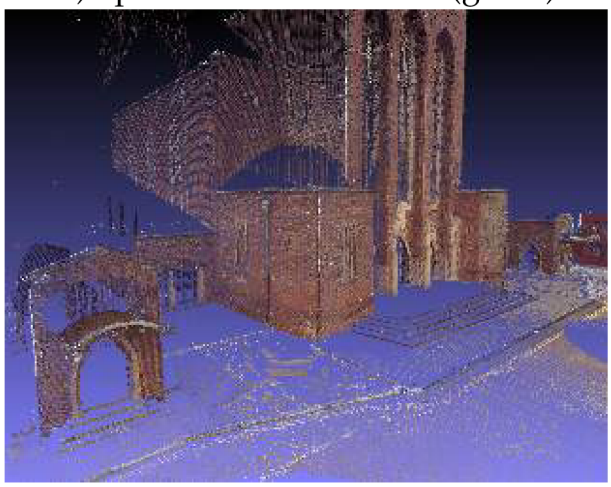


c) Spherical only



d) Spherical + Monocular (green)

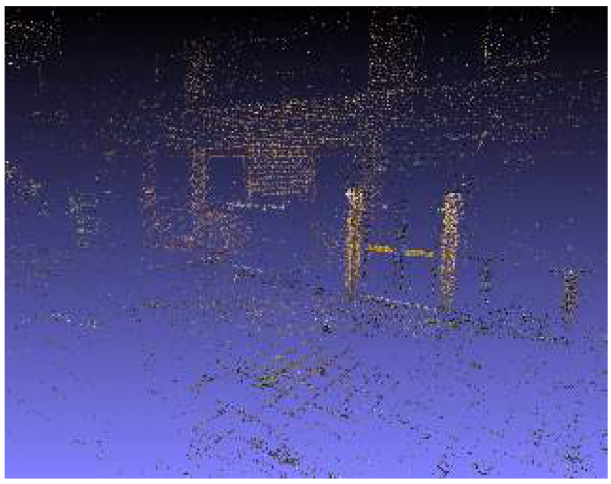
e) Spherical + Monocular



f) CLIDAR + Spherical (sparse) + Monocular

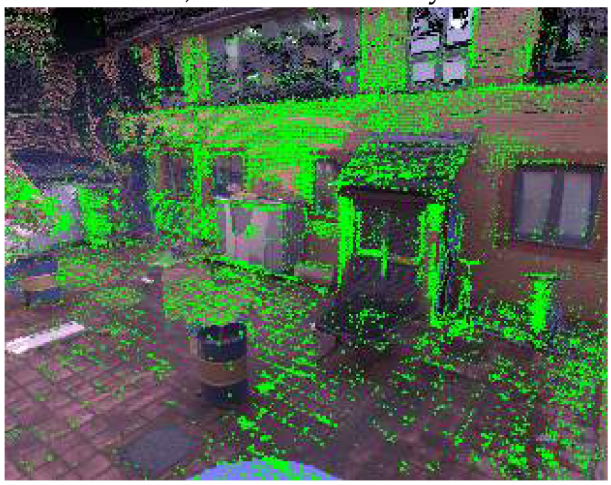
Figure 25: Reconstructions of the Cathedral dataset

a) CLIDAR only



b) Monocular only

c) Spherical only



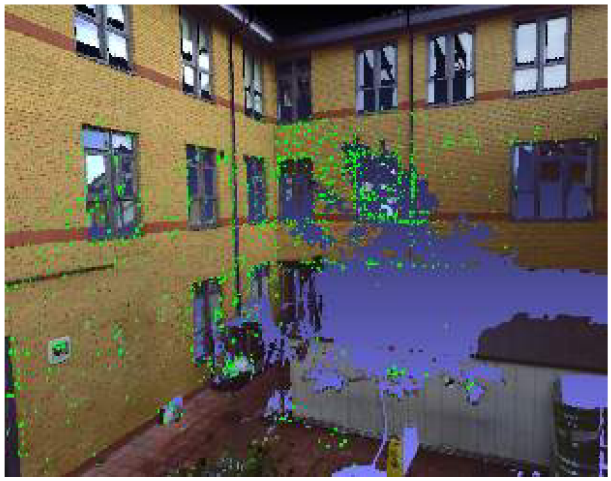
d) Spherical + Monocular (green)

e) Spherical + Monocular

f) CLIDAR + Spherical (sparse) + Monocular

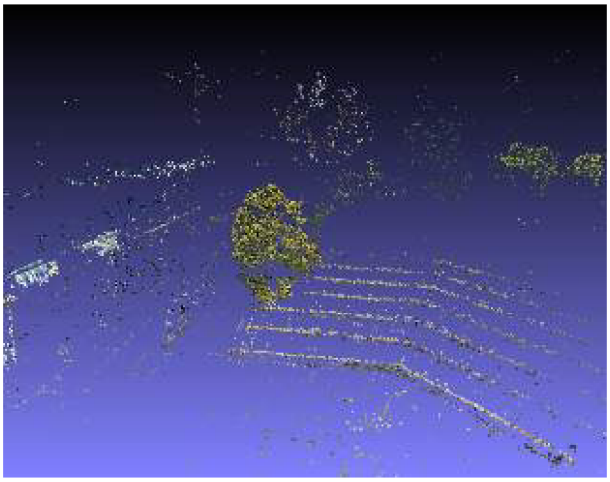
Figure 26: Reconstructions of the CCSR dataset

a) CLIDAR only

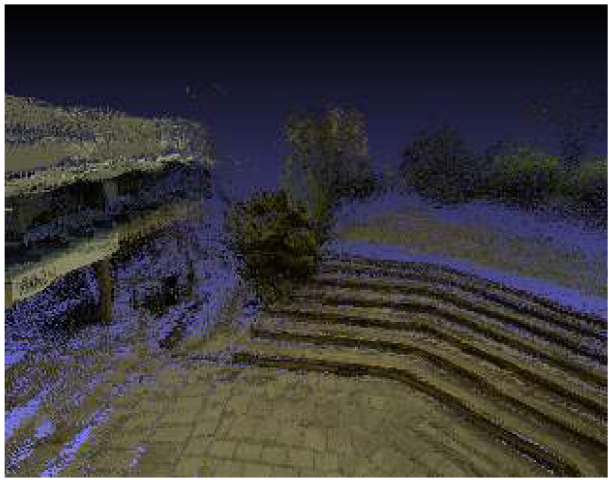


b) CLIDAR + Monocular (green)

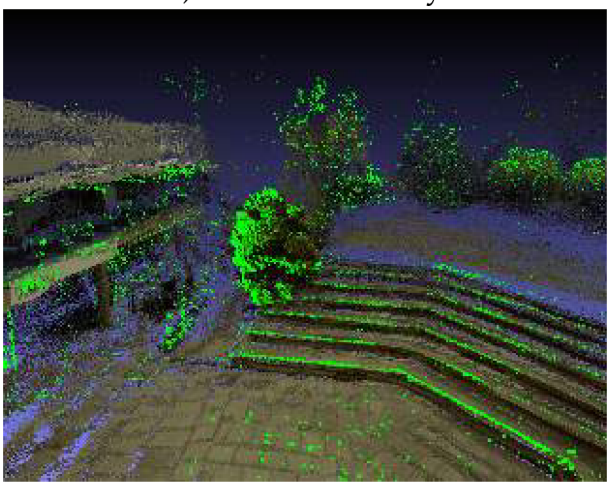
Figure 27: Reconstruction of Synthetic dataset.



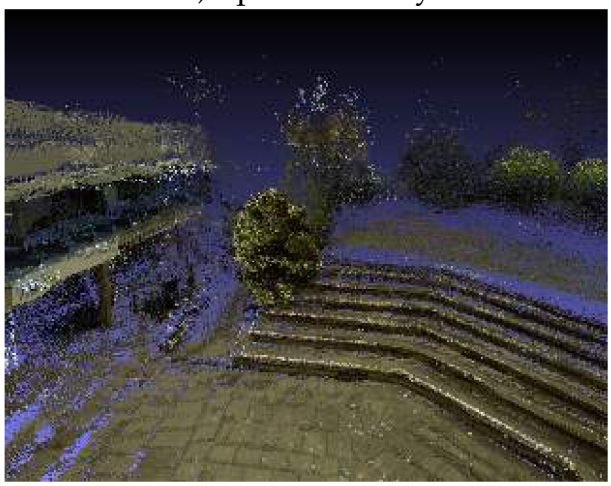
a) Monocular only



b) Spherical only



c) Spherical only + Monocular (green)



d) Spherical + Monocular

Figure 28: Reconstructions of the Atrium dataset

8 CONCLUSION

The contribution of this thesis is the formulation of the multisensor 3D reconstruction using unified representation for different sensors and measurements in terms of sparse BA and based on that, obtaining complete solution from all available data without need of manual alignment of models created by single sensor reconstruction algorithms. The representation consists of *variables* defining the poses of the sensors and structure points and *edges* encoding the relations between variables.

A sparse 3D reconstruction pipeline consists of a front-end which processes the sensor data and provides an initial estimate for the sensor position and the 3D structures, which is further optimized by the back-end. In this thesis we analysed algorithms for reduction of spherical distortion in images from spherical cameras and generated from CLIDAR devices to achieve higher initial registration accuracy. We evaluated multiple feature extractors, matching and registration accuracy of longitude-latitude images and planar images. This thesis proposes an algorithm that computes the *tengential* projection which reduces the effect of spherical distortion in longitude-latitude images and achieves better accuracy compared to registration using the longitude-latitude images in uncorrected form.

After the initialisation, the unified system built from measurements of multisensor data is refined by joint sensor pose and structure optimisation. This offers a robust estimation capable of exploiting relationships between multiple sensors and refining the solution according to those constraints. This is formulated as an optimization on graphs where the vertices represent the variables and the edges of the graph are derived from the measurements. The graph optimization is implemented in the SLAM++ non-linear least squares optimisation library developed in collaboration with my colleagues L. Polok and V. Ila. The SLAM++ is a very efficient library based on fast sparse block matrix manipulation.

The future work will include integration of the incremental optimisation approach of SLAM++ for time efficient incremental data processing. Furthermore the processing of data from additional sensors will be implemented as well as support for processing of videos from monocular and spherical cameras, including key-frame selection. Another area of the interest is the estimation of the dense depth map from the spherical images more accurately using the depth information from other sensors.

BIBLIOGRAPHY

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

[2] P. Amestoy, T. A. Davis, and I. S. Duff. Amd, an approximate minimum degree ordering algorithm). *ACM Transactions on Mathematical Software*, 30(3):381–388, 2004.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, Sept 1987.

[4] P. A. Beardsley, A. Zisserman, and D. W. Murray. *Navigation using affine structure from motion*, pages 85–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

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

[6] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[7] M.C. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. 23(12):1113–1139, Dec 2004.

[8] G. Carmichael. Matching spherical panoramas and planar photographs, 2009.

[9] J. L. De Carufel and R. Laganière. Matching cylindrical panorama sequences using planar reprojections. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 320–327, Nov 2011.

[10] O. Chum and J. Matas. Matching with PROSAC - Progressive Sample Consensus. In *Proc. CVPR*, pages 220–226, 2005.

[11] O. Chum and J. Matas. Optimal Randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008.

[12] O. Chum, J. Matas, and J. Kittler. Locally Optimized RANSAC. In *Lecture Notes in Computer Science*, volume 2781, pages 236–243. Springer, 2003.

[13] T. Davis. Csparse. http://www.cise.ufl.edu/research/sparse/CSparse/, 2006.

[14] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[15] T. A. Davis and W. W. Hager. Modifying a sparse cholesky factorization, 1997.

[16] T. A. Davis and W. W. Hager. Modifying a sparse cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 20(3):606–627, May 1999.

[17] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research, IJRR*, 25(12):1181–1204, December 2006.

[18] M. Fiala and G. Roth. Automatic alignment and graph map building of panoramas. *IEEE Int. Workshop on Haptic Audio Visual Environments and their Applications*, pages 103–108, 2005.

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

[20] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, Aug 2010.

[21] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(8):930–943, Au-

gust 2003.

[22] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968, June 2011.

[23] G. H. Golub and R. J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, pages 34:3–38, 1980.

[24] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *In Proc. of Robotics: Science and Systems (RSS*, 2007.

[25] A. R. Hanson and R. Kumar. Robust methods for estimating pose and a sensitivity analysis, 1994.

[26] R. I. Hartley and P. Sturm. *Computer Analysis of Images and Patterns: 6th International Conference, CAIP '95 Prague, Czech Republic, September 6–8, 1995 Proceedings*, chapter Triangulation, pages 190–197. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.

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

[28] A. Howard and N. Roy. The robotics data set repository (Radish), 2003.

[29] V. Ila, L. Polok, P. Smrž, M. Šolony, and P. Zemčík. Incremental block cholesky factorization for nonlinear least squares in robotics. In *In proceedings of The Robotics: Science and Systems 2013 Conference*, pages 1–8. MIT Press, 2013.

[30] V. Ila, L. Polok, P. Smrž, M. Šolony, and P. Zemčík. Incremental cholesky factorization for least squares problems in robotics. In *Proceedings of The 2013 IFAC Intelligent Autonomous Vehicles Symposium*, pages 1–8. IEEE Computer Society, 2013.

[31] V. Ila, L. Polok, M. Šolony, P. Zemčík, and P. Smrž. Fast covariance recovery in incremental nonlinear least square solvers. In *Proceedings of IEEE International Conference on robotics and Automation*, pages 1–8. IEEE Computer Society, 2015.

[32] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Robot. Auton. Syst.*, 57(12):1198–1210, December 2009.

[33] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[34] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. pages 1670–1677, Rome, Italy, April 2007.

[35] H. Kim and A. Hilton. 3d scene reconstruction from multiple spherical stereo pairs. *International Journal of Computer Vision*, 104(1):94–116, 2013.

[36] K. Konolige. Sparse sparse bundle adjustment. In *British Machine Vision Conference*, Aberystwyth, Wales, 08/2010 2010.

[37] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613, May 2011.

[38] V. Lepetit, F.Moreno-Noguer, and P.Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal Computer Vision*, 81(2), 2009.

[39] B. Lohani. Airborne altimetric lidar: Principle data collection processing and applications. 2008.

[40] M. I. A. Lourakis and A. A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw.*, 36(1):2:1–2:30, March 2009.

[41] B. Micusik and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–485–I–490 vol.1, June 2003.

[42] E. Olson. *Robust and Efficient Robot Mapping*. PhD thesis, Massachusetts Institute of Technology, 2008.

[43] L. Polok, V. Ila, and P. Smrž. Cache efficient implementation for block matrix operations. pages 698–706. ACM, 2013.

[44] L. Polok, V. Ila, and P. Smrž. 3d reconstruction quality analysis and its acceleration on gpu clusters. In *Proceedings of European Signal Processing Conference 2016*, pages 1–8. Institute of Electrical and Electronics Engineers, 2016.

[45] L. Polok, V. Ila, M. Šolony, P. Zemčík, and P. Smrž. Efficient implementation for block matrix operations nonlinear least squares problems for robotic applications. In *Proceedings of 2013 IEEE International Conference on Robotics and Automation*, pages 123–131. IEEE Computer Society, 2013.

[46] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *International Conference on 3-D Imaging and Modeling*, pages 145–152, 2001.

[47] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[48] J. Salvi, X. Armangué, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35(7):1617 – 1635, 2002.

[49] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[50] Ch. V. Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, 1999.

[51] H. Strasdat, A. Davison, and E. Edwards. *Local Accuracy and Global Consistency for Efficient SLAM*. Imperial College London, 2012.

[52] T. Svoboda, T. Pajdla, and V. Hlaváč. *Computer Vision — ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June, 2–6, 1998 Proceedings, Volume I*, chapter Epipolar geometry for panoramic cameras, pages 218–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[53] P. H. S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[54] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London B: Biological Sciences*, 203(1153):405–426, 1979.

[55] V.Fragoso and M. Turk. SWIGS: A Swift Guided Sampling Method. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Portland, Oregon, June 2013.

[56] T. Werner and T. Pajdla. Cheirality in epipolar geometry. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 548–553 vol.1, 2001.

[57] Ch. Wu. Siftgpu: A gpu implementation of scale invariant feature transform. 2007.

[58] Ch. Wu. Towards linear-time incremental structure from motion. In *Proceedings of the 2013 International Conference on 3D Vision*, 3DV '13, pages 127–134, Washington, DC, USA, 2013. IEEE Computer Society.

[59] Ch. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3057–3064, Washington, DC, USA, 2011. IEEE Computer Society.

[60] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artif. Intell.*, 78(1-2):87–119, October 1995.