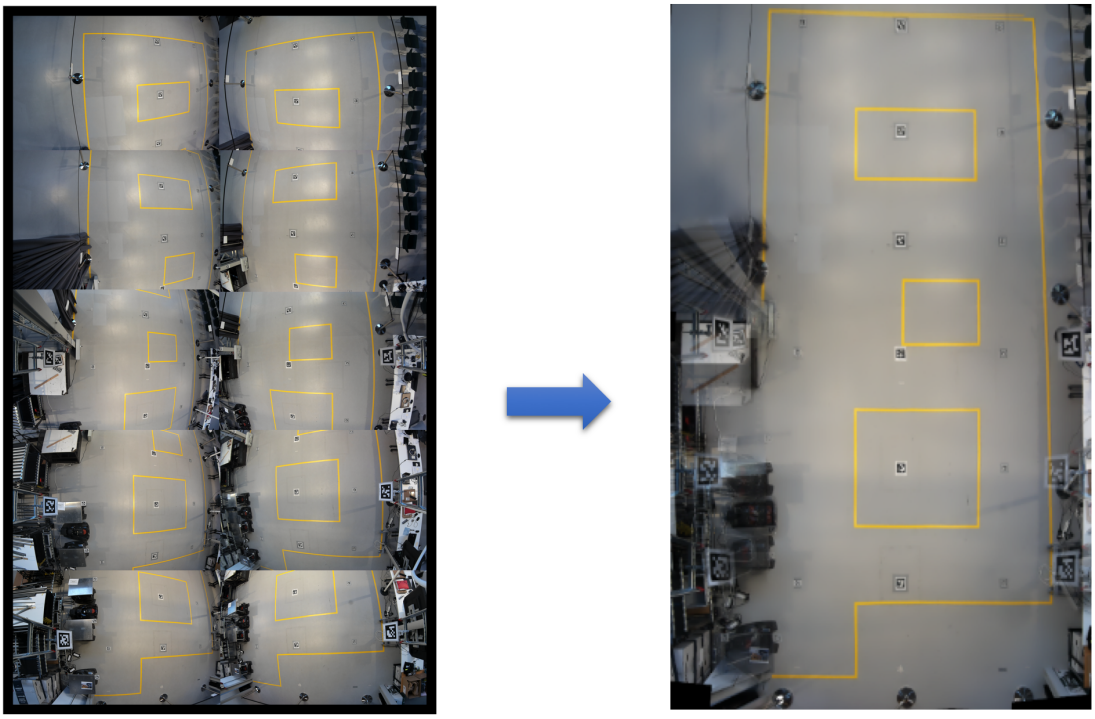
CHALMERS
UNIVERSITY OF TECHNOLOGY



Indoor 2D map generation using projective transformation

Master's thesis in Systems, Control and Mechatronics

Admir Alihodza
Jonas Hejderup

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020:EENX30

Indoor 2D map generation using projective transformation

Admir Alihodza Jonas Hejderup

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
Gothenburg, Sweden 2020

Indoor 2D map generation using projective transformation
ADMIR ALIHODZA JONAS HEJDERUP

Company Supervisor: Per-Lage Götvall, Volvo Group Trucks Operations
Academic Supervisor & Examiner: Torsten Sattler, Department of Electrical Engineering

Master's Thesis 2020:EENX30
Department of Electrical Engineering
Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: The figure shows all the used input images and the final generated map using the pipeline outlined in this thesis.

Typeset in LaTeX
Gothenburg, Sweden 2020

Indoor 2D map generation using projective transformation
ADMIR ALIHODZA JONAS HEJDERUP
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Indoor navigation and localization within complex environments is a critical component for many systems. With increased computation power and cheaper cameras, it is becoming more viable to use computer vision-based techniques to generate large-scale indoor maps using images. In recent years, Structure of motion has been an approach for these types of problems, where the 3D reconstruction of the indoor scene forms the basis of the map. An alternative approach is to estimate projective planar transformations between images and stitch them together into a map. This approach is better suited than Structure of motion since most indoor scenes contain planar surfaces.

The aim of the thesis is to develop a pipeline for generating a 2D indoor map using images over a planer scene. The primary approach is to stitch images together into a single image using planar projective transformations. The pipeline's performance is evaluated by constructing a map over an indoor scene and measuring the overall distance accuracy of the map. The evaluation shows that the proposed pipeline can generate an accurate map over an indoor scene that resembles an industrial factory environment.

Keywords: Map generation, Computer vision, Homography, Image stitching, Bundle adjustment, Multi-band blending.

Acknowledgements

Firstly, we would like to thank our supervisor and examiner Torsten Sattler for his extremely valuable support and guidance throughout the thesis. We were constantly amazed by his vast knowledge within computer vision and he kept us encouraged throughout the thesis. Thank you Torsten!

Secondly, we would like to thank Per-Lage Götvall at Volvo Trucks Operations for giving us the opportunity to carry out our thesis at AB Volvo. We would also like to thank him for his insights and support during the thesis.

Lastly, we would like to thank our families and friends for the support given to us during our pursuit of an engineering degree at Chalmers. You guys kept using going the early mornings and late nights. Without your support and love, we would have never come this far.

The last dance!

Admir Alihodza & Jonas Hejderup, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1
Introduction

An indoor digital map aims to create an accurate digital two-dimensional representation of an indoor scene. These maps form the foundation of many systems that rely on precise localization and navigation within complex indoor environments. An example of these types of environments are factories where determining and navigating to different locations in the environment is highly critical for many different systems. These maps are either created manually or using some of measuring device such as LiDAR that is mounted on a robot which maps the entire indoor scene. With the introduction of cheaper cameras and increased computing power, it is becoming a more viable option for generating maps by using computer vision-based techniques. Volvo Group Trucks Operations wants to investigate the possibility of automatically generate a 2D map using images over an indoor scene.

A possible vision-based method of generating an indoor map is to use Structure from Motion (SfM). This is a technique used to reconstruct a 3D scene using a series of images from different viewpoints by estimating both the camera pose and the 3D structure [1]. SfM is an extensively studied area within computer vision which has proven to work quite well with large-scale reconstructions [2, 3, 4, 5, 1]. For the 3D reconstruction to successfully generate an accurate map, the point cloud of the indoor scene needs to be dense, in particular for the floor of the scene. An alternative approach is to consider the indoor scene as a large single plane instead of a 3D structure since the scene mostly consist of a floor. Therefore, it is more suitable to estimate a planar projective transformation that relates a common plane in all images corresponding to a plane in the scene, namely the floor. An existing method that uses this approach is the Micro-GPS system [6]. It works by generating a map over the planar surface by using a robot which takes close-range images over the surface. These images are stitched together to form a 2D map over the planar surface. This process is quite slow since it requires a robot to drive through the entire scene and inflexible to any change in the scene since it would require the robot to remap the entire scene again.

The thesis presents a vision-based pipeline of generating a digital 2D indoor map which is heavily inspired by the automatic image stitching pipeline [7]. It uses images captured from the ceiling of the indoor scene and finds feature correspondences between them. The correspondences are used to estimate a homography [1] that relates the pixels between the images. These homographies are called pairwise homographies and is used to calculate another homography that relates all input images into a common image coordinate system. From these homographies, it is possible to

stitch the images into a single image. The homographies are further optimized by using bundle adjustment [8] to ensure the images are correctly aligned. The stitched image is correctly scaled and the seams between the images are removed by using multi-band blending [2]. The proposed pipeline uses projective planar transformation instead of creating a 3D reconstruction of the indoor scene and therefore does not require a dense point cloud. It also addresses several shortfalls of the Micro-GPS system, mainly that it simplifies the map generation process. Micro-GPS system requires the images to be captured close to the surface while the proposed method uses images captured from the ceiling. Images from the ceiling cover more of the indoor scene, thereby the pipeline needs fewer images to construct the indoor map.

The following contributions made by the thesis:

- Proposed a pipeline for generating a map using images over an indoor scene.
- The proposed pipeline was able to generate an accurate map using images from a real world indoor scene resembling an industrial environment.
- The components of the pipeline were evaluated to study their influence on the generated map.

1.1 Related work

This section presents previous research and work related to vision-based methods for indoor map generation, as well as previous works that will be incorporated into the proposed map generation pipeline.

1.1.1 3D reconstruction

An indoor map can be generated by reconstructing an indoor 3D scene from a set of images by estimating the camera poses and the 3D scene points. Most recent research has focused on SfM and visual Simultaneous Localization and Mapping (SLAM) [9]. SfM is mainly used for offline tasks and has been shown to successfully solving large-scale reconstruction problems such as reconstructing buildings and cities [10]. Visual SLAM is an online method that mainly focuses on the navigation of robots and autonomous cars where the task is to locate and build a map of an unknown environment using measurements from various sensors. More specifically, SLAM tries to estimate the positions of the landmarks and the robot by using the measurement information from the sensors. The landmarks are used to construct a map of the unknown environment. Visual refers to the fact that cameras are used as sensors. Both methods are similar and mostly differs on the type of applications, SfM for offline problems and visual SLAM for online problems.

The general pipeline for SfM consists of extracting and matching features for all of the input images. Commonly Scale-invariant feature transform (SIFT) [11] is used as a feature extractor since it has shown to be quite robust for finding distinct features between the images [12]. For uncalibrated cameras, eight matched feature correspondences are used to calculate the relative pose between the views by using

epipolar geometry [13] while calibrated camera only need five matched feature correspondences to compute the relative pose [14]. The scenes points are estimated by using the computed camera poses and triangulation [1]. Due to noise and drift, the resulting camera poses and scene points needs to be further refined by using bundle adjustment [1] which solves a nonlinear optimization problem that minimizes the reprojection error. As previously discussed, to generate an accurate map using SfM, the 3D reconstruction needs to be dense. Most feature detectors struggle to find distinct features over homogenous surfaces such as a floor. This results in just a few point correspondences are found between the images which lead to a quite spare final 3D reconstruction. Therefore, the sparse point cloud would require further reconstructions to turn it into a dense point cloud by for example using the Patch-based Multi-View Stereo algorithm [15]. The points that correspond to the floor of the indoor scene are not constrained to be located on the same plane which further adds inaccuracies when generating a map.

Early visual SLAM was based on Bayesian framework to reduce computing time where an Extended Kalman filter is used to estimate the landmarks and the camera pose using extracted features from the images [16]. More recently, visual SLAM methods have started to use sparse bundle adjustment which optimizes the 3D landmarks along with the camera poses [17]. This has resulted in higher accuracy for the estimated map given the same constraint on the computation time [17]. This approach has shown to generate a consistent map for large-scale outdoor environments [18].

1.1.2 Image Stitching

Aligning images and stitching them together is a well-established problem within computer vision [19, 20, 21, 22, 23], where the goal is to relate one image's coordinate system into another image's coordinate system.

Within image stitching, there are two main techniques: Direct alignment and Feature-based registration [19]. Earlier image stitching algorithms were based on direct alignment which searches through all possible alignments and picks the alignment with the lowest error [19]. There are different techniques used to speed up the search, but overall this technique is slow and not as robust as the feature-based techniques [19]. More recent image stitching algorithms are using feature-based techniques since they are generally faster and more robust [7]. The feature-based techniques try to find distinct features between the images and estimate a homography to transform one image's coordinate system into another [7]. A general feature-based image-stitching pipeline consists of feature extraction and matching, homography estimation, bundle adjustment, blending and compositing the final image [2]. Crucially, the introduction of bundle adjustment significantly improved the accuracy of the stitched image since it gives a globally consistent solution by optimizing all of the poses in the image simultaneously [19].

It has been shown that it is possible to stitch together images of a small planar surface such as a whiteboard by using homographies [24]. As far as the authors know there has not been any research or use of image stitching when it comes to indoor map generation, except for the Micro-GPS system [6]. The Micro-GPS system works by first constructing an offline database over the planar surface by using a robot that takes close-range images over the surface for the entire indoor scene. The images are stitched together by using the general feature-based stitching pipeline with bundle adjustment. These stitched together images form the map over the indoor scene. The Micro-GPS system solves the problem with the lack of texture on homogeneous surfaces by using images close to the surface for the SIFT detector to find distinct features [6].

There is not any fundamental algorithmic differences between our proposed approach and the Micro-GPS system. Both pipelines use the general image stitching pipeline as the foundation. The difference lies in the type of input images used in the respective pipelines. The Micro-GPS requires images captured close to the surface of the indoor scene which is a cumbersome process especially for larger indoor scenes such as factories or warehouses. Our approach uses images taken from the ceiling which is able to capture more of the scene and thereby needing fewer images. If the cameras are mounted in the ceiling, then our approach could automatically update the map if the layout of the indoor scene changes. However, our approach will struggle more to find point correspondences since the images are not detailed enough to extract the texture of the homogeneous surface. Instead, the approach relies on that other distinct features exist on the floor, such as lane markings.

1.2 Thesis Outline

The thesis consists of four sections. The theory section contains the underlying theory of the different components in the proposed pipeline. Next section describes how the proposed pipeline is able to generate a map given a set of input images along with details about the implementation of the pipeline. The results section presents the evaluation of the pipeline for a real world indoor scene. Also, the different components of the pipeline are evaluated to study their influence on the generated map. Finally, the conclusion summarizes the results of the thesis along with suggestions for potential future work.

2

Theory

Image stitching is a problem that aims to align and join images that cover only parts of a scene into a large image that covers the entire scene. The images are seamlessly and consistently joined together in a way where the result should be an aesthetically pleasing image which accurately portrays the scene. The pipeline proposed by the thesis is heavily influenced by the general image stitching pipeline [19], which can be seen in Figure 2.1.

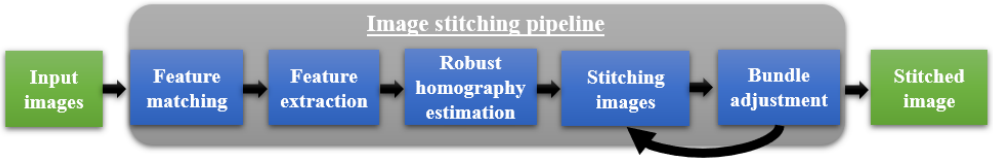


Figure 2.1: The figure shows a flow chart of a general image stitching pipeline [19]. The pipeline shows the different stages required to stitch together the input images into a single image. From the input images, features are extracted and matched. The matched features are used to estimate an homography. The images are stitched using the homography and the result is optimized using bundle adjustment.

The first section of the chapter will introduce the pinhole camera model and projective geometry which forms the mathematical foundation of the proposed pipeline. The section after will introduce and present how homographies can be used to transform images so that they align with each other. Also, methods of estimating homographies using point-correspondences between images will be presented. Afterwards, the next section will present how to extract, describe and match features which form the point-correspondences used to estimate the homographies between the images. The point-correspondences will not always be correctly matched and therefore an algorithm called RANdom SAmple Consensus (RANSAC) [25] will be presented to robustly estimate model parameters using data containing outliers. Bundle adjustment will be explained, which is a method of further refining the homographies by using non-linear optimization. Finally, multi-band blending will be presented, which is a method of removing the seams for the stitched-together image to create a more aesthetically pleasing result.

2.1 Camera Model

Camera models are mathematical models used for mapping 3D points in the world into 2D points in the image. The mathematical model is different between camera models and the difference depends on for example the lens a certain camera uses to focus or disperse light [1]. One of the most commonly used and simplest camera model is the pinhole camera. The mathematical model behind the pinhole camera is illustrated in Figure 2.2.

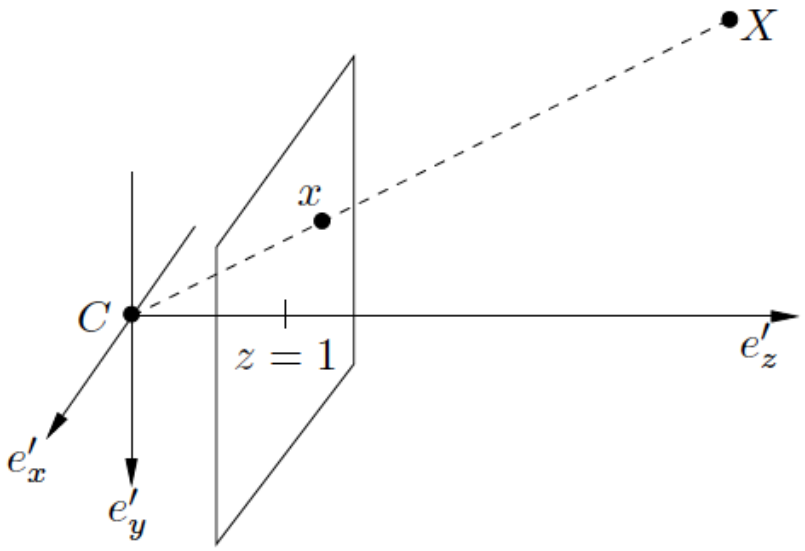


Figure 2.2: Illustration of the mathematical model behind the pinhole camera model. The pinhole camera coordinate system is given by $e_x^{'}, e_y^{'}, e_z^{'}$ where the origin is the camera center C and the plane at $e_z^{'} = 1$ is the image plane. The illustration shows the ray between the scene point X and the camera center C intersects the image plane at point x. The figure is taken from [26].

The origin in the camera model is the camera center C for which all viewing rays associated with the image intersect. A global 3D point is projected to a point on the image plane where a ray connecting the global point to the camera center intersects the image plane. The mapping of a point from 3D world to 2D image coordinates can be computed using a perspective transformation [1]. For the pinhole camera model, the perspective transformation is given by the camera matrix

$$P = K \begin{bmatrix} R & t \end{bmatrix}, \tag{2.1}$$

where $\begin{bmatrix} R & t \end{bmatrix}$ are the external parameters of the camera matrix which relate the position and orientation of the camera frame to the world coordinate frame. The first parameter R is a 3×3 rotation matrix and the second parameter t is a 3×1 translation vector. By applying the rotation R and the translation t it is possible

to transform a point from the world frame into the pinhole camera frame as seen in Figure 2.3. While K is the calibration matrix of the camera matrix and it encodes the internal orientation of the camera. In other words, the calibration matrix K maps transformed world points in the camera frame into points in the image frame which has units pixels as seen in figure 2.3. The calibration matrix K is an upper triangular matrix and is constructed for a pinhole camera as

$$K = \begin{bmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where γ is an aspect ratio parameter and s is a skew parameter. The aspect ratio determines the pixel $x : y$ ratio while the skew corrects for tilted pixels in the image. f is the focal length and corresponds to the distance between the image plane and the camera center. The coordinates (x_0, y_0) are denoted the principal point and are the pixel coordinates in the image of the point where the z-axis intersects the image plane. The focal length re-scales image coordinates into pixels while the principle point translates the origin from the intersection point in the image plane to the upper left corner of the image. Therefore, after applying the calibration matrix, the coordinates will be measured in pixels where the origin is at the top left corner of the image.

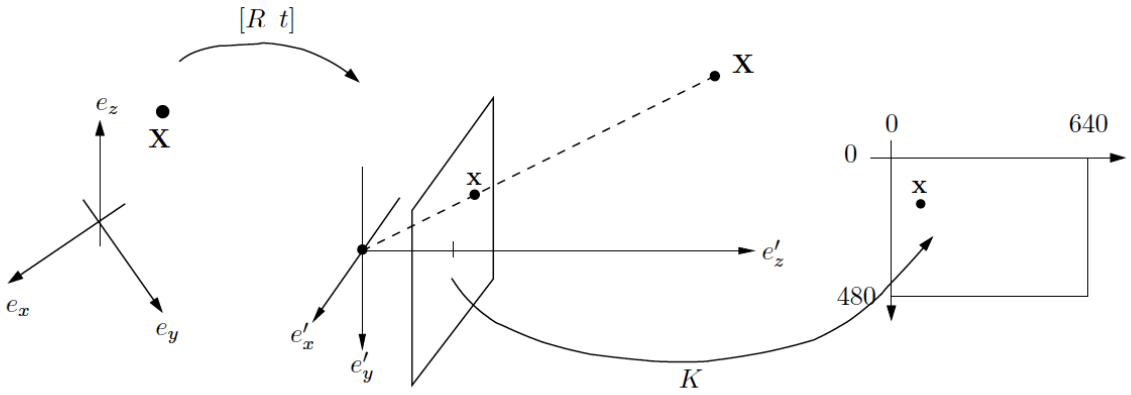


Figure 2.3: An illustration of a point transformed between coordinates frames. The transformation of a point between the left and middle image shows the mapping of a 3D point from the world frame into the pinhole camera frame using the external parameters $[R\ t]$. The transformation of a point between the middle and right image shows the mapping of a point from camera coordinates into pixels coordinates using the calibration matrix K. The figure is taken from [27].

As previously mentioned, 3D world points can directly be mapped to pixel coordinates in the image using the camera matrix P. This is possible since the camera matrix contains the calibration matrix K and the external parameters $\begin{bmatrix} R & t \end{bmatrix}$. However, before applying the camera matrix to a given point in the 3D world, it is necessary to rewrite the points into homogeneous coordinates. Homogeneous coordinates are necessary in order to be able to use matrix multiplication between

the camera matrix and the 3D world point. Applying homogeneous coordinates increases the dimension of the point with one. In other words, a N dimensional point becomes a $N+1$ dimensional point. This means that a 3D world point can directly be projected into pixel coordinates in an image using homogeneous coordinates and the camera matrix. The mapping of a 3D world point into pixel coordinates using homogeneous coordinates can be computed as

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.3}$$

where λ is an arbitrary scaling parameter and $\begin{bmatrix} x & y \end{bmatrix}^T$ is the pixel coordinates in the image of the corresponding 3D world point $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$. Due to the use of homogeneous coordinates, the projected pixel point will have a $N+1$ dimensionality. However, the corresponding pixel point can easily be retrieved by dividing all rows with the last element and removing the last row after the division.

2.2 Homography

As discussed, a perspective transformation is a projection of 3D points in the world frame to 2D points in the image frame. Another transformation that is used for mapping between planes or 2D points to 2D points is a planar projective transformation [1]. A planar projective transformation is also sometimes referred to as a homography and is represented by a non-singular 3×3 matrix as

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{2.4}$$

A homography is an invertible mapping of 2D point to 2D points in 2D projective space $\mathcal{P}^2$ where lines are mapped to lines [1]. In other words, three points that lie on a single straight line in $\mathcal{P}^2$ will be mapped, using the homography, to three new points in $\mathcal{P}^2$ which also lie on a straight line. Therefore, a homography is a point to point mapping which has the property to preserve lines between the mappings [1].

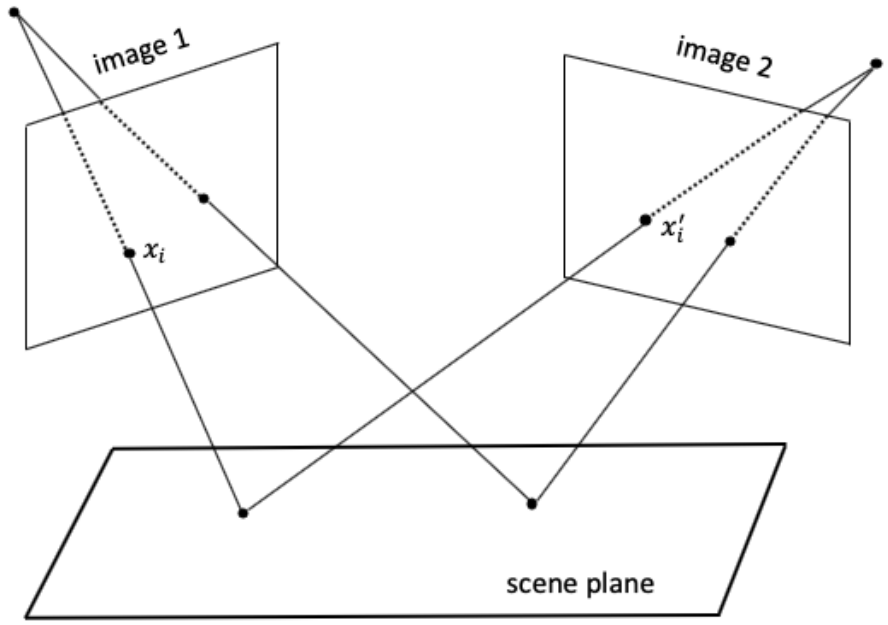


Figure 2.4: Illustration of how a homography maps point correspondences between two images with common world points belonging to the same scene plane.

Typically, homographies are used to transform 2D points from one image into 2D points in another image coordinate system. The transformation is valid under the assumption that both images have a set of common world points that belong to the same scene plane, as seen in Figure 2.4. If the assumption holds, there is a homography corresponding to a scene plane that maps points between images belonging to the same image plane and corresponding to the same point. The mapping of image points between images with a common image plane and common points can be expressed as

$$\underbrace{\begin{bmatrix} x_1' \\ y_2' \\ 1 \end{bmatrix}}_{\boldsymbol{x}_i'} = H \underbrace{\begin{bmatrix} x_1 \\ y_2 \\ 1 \end{bmatrix}}_{\boldsymbol{x}_i}, \tag{2.5}$$

where $\boldsymbol{x}_i$ is a set of 2D points in one image and $\boldsymbol{x}_i'$ are the corresponding set of mapped points in the other image. As in the case with the camera matrix, 2D image points are represented as 3D image points using homogeneous coordinates. The use of homogeneous coordinates makes the transformation linear and gives a linear transformation of homogeneous points between image planes in 2D projective space $\mathcal{P}^2$ [1].

Another essential property of a homography is that consecutive transformations between multiple images are applicable. For example, lets assume there exists a homography H_{AB} mapping image points from I_B to I_A and another homography

H_{BC} mapping image points from I_C to I_B. Then it is possible to map image points from I_C to I_A by consecutively multiplying homographies as

$$H_{AC} = H_{AB}H_{BC}, \tag{2.6}$$

where H_{AC} is the homography mapping points from image I_C to I_A.

2.3 Homography Estimation

If two images are capturing the same planar scene from different angles, it is possible to estimate the homography relating 2D pixel points from one image to another. The estimation is performed by extracting feature correspondences between two images and ensuring that the correspondences are image points belonging to the same planar scene. If the correspondences between the images are representing the same points in a common planar scene, it is possible to estimate a homography that maps points from one image plane to another. There are many different linear algorithms for determining the homography given point correspondences between images. One of the simplest algorithms is the direct linear transformation (DLT) algorithm [1].

The DLT algorithm is a linear algorithm that uses a set of point correspondences to solve a homogeneous linear system in order to estimate the homography matrix. More specifically, a homography is a 3×3 matrix that contains nine elements. However, only eight elements are solved as equality is up to an arbitrary scale [28]. The reason for only eight unknown elements comes from the fact that homogeneous coordinates are used and a homography can be multiplied with a nonzero scale factor but still represent the same projective transformation [2]. In other words, the homography is homogeneous matrix with eight degrees of freedom. Eight equations are needed to determine eight unknowns. Each pair of corresponding points yields three equations when inserted to Equation (2.5). But only the two first equations are linearly independent since the third equations is a linear combination of the two other [1]. Therefore, each pair of corresponding points accounts only for two equations that can be used to solve for the unknown elements. The minimal amount of point correspondences n required to solve for the homography is given by

$$2n \geq 8 \Leftrightarrow n \geq 4. \tag{2.7}$$

Therefore, at least four point correspondences are needed for the DLT to be able to solve the homogeneous linear system of equations. The DLT algorithm in [1] solves the generated system of linear equations by representing everything in matrix form and estimates an approximate solution by finding the nullspace of the system matrix. The DLT algorithm below is derived in the same way as the derivation presented in [1]. Firstly, the expression in Equation (2.5) can be rewritten in terms of a vector cross product as

$$\boldsymbol{x}_i^{'} \times H\boldsymbol{x}_i = 0, \tag{2.8}$$

where $\boldsymbol{x}_i^{'}$ and $\boldsymbol{x}_i$ are the image point correspondences related by the homography H. It is possible to formulate the transformation as a cross-product because the vectors $\boldsymbol{x}_i^{'}$ and $H\boldsymbol{x}_i$ have the same direction even though they may differ in magnitude since homogeneous coordinates are used [1]. Having the same direction implies that the vectors are parallel and therefore their cross-product is zero. To simplify the derivation, one can rewrite $H\boldsymbol{x}_i$ as

$$H\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{h}_1^T \boldsymbol{x}_i \\ \boldsymbol{h}_2^T \boldsymbol{x}_i \\ \boldsymbol{h}_3^T \boldsymbol{x}_i \end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{h}_j^T$ is a 1×3 vector containing the j-th row of H. Substituting $H\boldsymbol{x}_i$ into Equation (2.8) and performing the cross-product yields

$$\boldsymbol{x}_i^{'} \times H\boldsymbol{x}_i = \begin{bmatrix} y_i^{'}\boldsymbol{h}_3^T x_i - \boldsymbol{h}_2^T x_i \\ \boldsymbol{h}_1^T x_i - x_i^{'}\boldsymbol{h}_3^T x_i \\ x_i^{'}\boldsymbol{h}_2^T x_i - y_i^{'}\boldsymbol{h}_1^T x_i \end{bmatrix}, \tag{2.10}$$

where $\boldsymbol{x}_i^{'} = \begin{bmatrix} x_i^{'} & y_i^{'} & 1 \end{bmatrix}^T$ and $\boldsymbol{x}_i = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$ are the i-th pair of corresponding points. The result from the cross product in Equation (2.10) can be expressed in terms of matrix multiplication as

$$\begin{bmatrix} \boldsymbol{0}^T & -\boldsymbol{x}_i^T & y_i^{'}\boldsymbol{x}_i^T \\ \boldsymbol{x}_i^T & \boldsymbol{0}^T & -x_i^{'}\boldsymbol{x}_i^T \\ -y_i^{'}\boldsymbol{x}_i^T & x_i^{'}\boldsymbol{x}_i^T & \boldsymbol{0}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.11}$$

which in a more compact way can be written as

$$\boldsymbol{A}_i \boldsymbol{h} = \boldsymbol{0}, \tag{2.12}$$

where $\boldsymbol{A}_i$ is a 3×9 matrix for the i-th pair of corresponding points and $\boldsymbol{h}$ is a 9×1 vector containing the unknown elements of the homography matrix H. Each $\boldsymbol{A}_i$ matrix is constructed using only one point correspondence while four correspondences are needed to determine the homography. In other words, four $\boldsymbol{A}_i$ matrices are needed to fully determine the homography. This can be constructed by assembling each $\boldsymbol{A}_i$ into a single matrix as

$$\underbrace{\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \\ \boldsymbol{A}_4 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix}}_{\boldsymbol{h}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \tag{2.13}$$

where $\boldsymbol{A}$ is a 12×9 matrix containing four $\boldsymbol{A}_i$ matrices constructed using four, $i = 1, 2, 3, 4$, pair of corresponding points. The system in Equation (2.13) can be solved by finding a non-zero vector in the nullspace of the system matrix $\boldsymbol{A}$. However, the system will not always have an exact solution since the correspondences are usually

exposed to noise. Therefore, a least squares solution is found by reformulating the problem as a homogeneous least-squares problem. Furthermore, the constraint $\|h\|^2 = 1$ is also added since the homography is only determined up to an arbitrary scale. A least-squares solution is obtained by minimizing the following

$$\min_{\|\boldsymbol{h}\|^2=1} \|\boldsymbol{A}\boldsymbol{h}\|^2 . \tag{2.14}$$

In some cases, the linear system is numerically unstable and causes the solution to diverge from the correct estimation. The reason for instability is due to the highly varying magnitude values between the entries in the A_i matrix. Image correspondences are measured in pixels and each point can have values that are in the thousands. Therefore, the magnitude between terms in the A_i matrix can vary highly since some terms are squared coordinates while other coordinates are constant. To prevent divergence and obtain a more stable system, [1] proposes that the point correspondences are normalized before performing DLT. The points are normalized by translating them such that their centroid is at the origin and their average distance from the origin is equal to $\sqrt{2}$.

2.4 Feature detection and matching

The first step in the stitching pipeline is to extract a set of image features and matching them between the images. Matching identical features between images make it possible to establish point correspondences between images that can be used to estimate the homography. The process can be divided into three main parts: feature extraction, creating feature descriptors and matching features. The feature extraction part is simply an algorithm that extracts unique features present in the image. The second part consists of representing each detected feature using a feature descriptor which is used to match features between each other. Features are matched by comparing descriptors between two images and a match is found if two descriptors are similar.

2.4.1 Feature detector

A feature detector algorithm extracts unique features present in an image. Several feature extraction algorithms exist, where the most apparent difference between them can be found in what kind of distinct feature they extract. Two commonly used algorithms are Harris corner detector [29] and SIFT [11]. Harris corner detector extracts corners from the image while SIFT extracts blobs. Selecting between these two detectors depends on the problem at hand and what features are available in the image.

2.4.1.1 Harris corner detector

Harris corner detector is a feature detector algorithm that aims to detect corner in an image [29]. Corners are identified by checking if their gradient changes in all directions. This makes corners distinct and easily distinguishable compared to

edges and flat regions where the gradient change either in one direction or not at all. Corners are detected by placing a small window that is centred around each pixel in the image. The window measures the intensity level of all the pixels located in the window. Then the squared difference in intensity level is computed by shifting the window by small amounts in every direction. This is can be expressed as

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2, \tag{2.15}$$

where $w(x,y)$ is the window function and u, v are the small shift in x and y direction in the image. The change in intensity should be large for corners, therefore Equation (2.15) should be maximized. This is achieved by firstly approximating Equation (2.15) using 2D Taylor-approximation and in [29] the equation is rewritten into matrix form as

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)}_{\mathrm{M}} \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.16}$$

where I_x and I_y are the partial derivative of the intensity with respect to x and y. Edges will also contribute to a change in the gradient, therefore it is important to ensure there is a large change in all directions. The eigenvalues of the M matrix are proportional to the principal curvatures from Equation (2.16) [29]. The size of these eigenvalues indicates whether a corner, edge or a flat region is present in the window [29]. There are three possible cases:

- If both eigenvalues λ_1 and λ_2 are large: A shift in either direction of the window gives a large change of the intensity, therefore this corresponds to a corner. As discussed before, the gradient should change in all directions for corners.
- If one eigenvalue is large and the other eigenvalue is small: A shift in one direction of the window gives a large change in intensity while a shift in the other direction gives a small change in intensity. This corresponds to an edge since the gradient does not change along the edge but changes when crossing the edge.
- If both eigenvalues λ_1 and λ_2 are small: A shift in either direction of the window gives a small change of the intensity, therefore this corresponds to a flat region. As discussed before, the gradient should not change in any directions for flat regions.

Figure 2.5 shows a graph over the eigenvalues along with the type of feature they correspond to. Previously, computing eigenvalues was considered as too computationally expensive, therefore [29] proposed a response function to efficiently determine if a corner is present in the window without the need to explicitly compute the eigenvalues. The response function is given by

$$R = det(M) - kTr(M)^2 \quad \text{where} \quad det(M) = \lambda_1 \lambda_2 \quad Tr(M) = \lambda_1 + \lambda_2. \tag{2.17}$$

The response R is given by the determinant and trace of the M matrix and k is an empirically determined constant. For corners, the response is positive while for edges the response is negative [29]. The response for flat regions has a low magnitude [29]. Contour curves for the response of different eigenvalues can be seen in Figure 2.5.

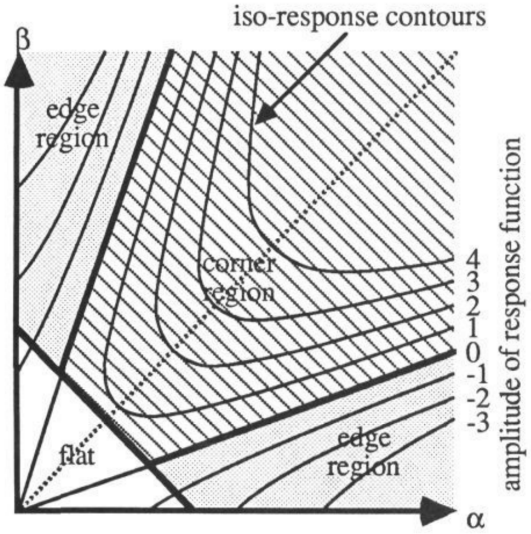


Figure 2.5: The graphs x and y-axis are the values of the two eigenvalues from the M matrix, here α corresponds to λ_1 and β to λ_2. It shows the contour plots for Equation (2.17) and how they varies with the values of λ_1, λ_2. The graph is divided into different regions corresponding if the eigenvalues indicate whether it is a corner, edge or flat region. The graph is from [29].

2.4.1.2 SIFT

The SIFT feature extractor is a blob detector that extracts distinctive features that are invariant for both scale and rotation and partially invariant to viewpoint and illumination [11]. A blob is a region of the image that share common properties such as intensity level or colour that is significantly different compared to neighbouring regions. Blobs are detected by filtering the image using Laplacian of Gaussian (LoG) filter to find local extrema in the image. For the feature extractor to be scale-invariant, the LoG filter needs to apply different Gaussian blur with varying variance σ and for different scales of the image [11]. However, the Laplacian of Gaussian (LoG) filter requires the calculation of the second derivative which is computationally expensive, therefore the Difference of Gaussian (DoG) filter is used as an approximation of LoG to find the local extrema [11]. The DoG filtering is by done smoothing the image with two Gaussian blurs. The first Gaussian blur has the variance σ while the second Gaussian blur has the variance $k\sigma$ where k is a multiplication factor. Then these two blurred images are subtracted shown as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{2.18}$$

where $L(x, y, \sigma)$ is the Gaussian blurred image with the variance σ and is defined by

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad \text{where} \quad G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}. \tag{2.19}$$

The DoG simplifies the computation since it simply involves applying Gaussian blurs to images and subtract them. The different scale and the DoG filtering process are illustrated in Figure 2.6.

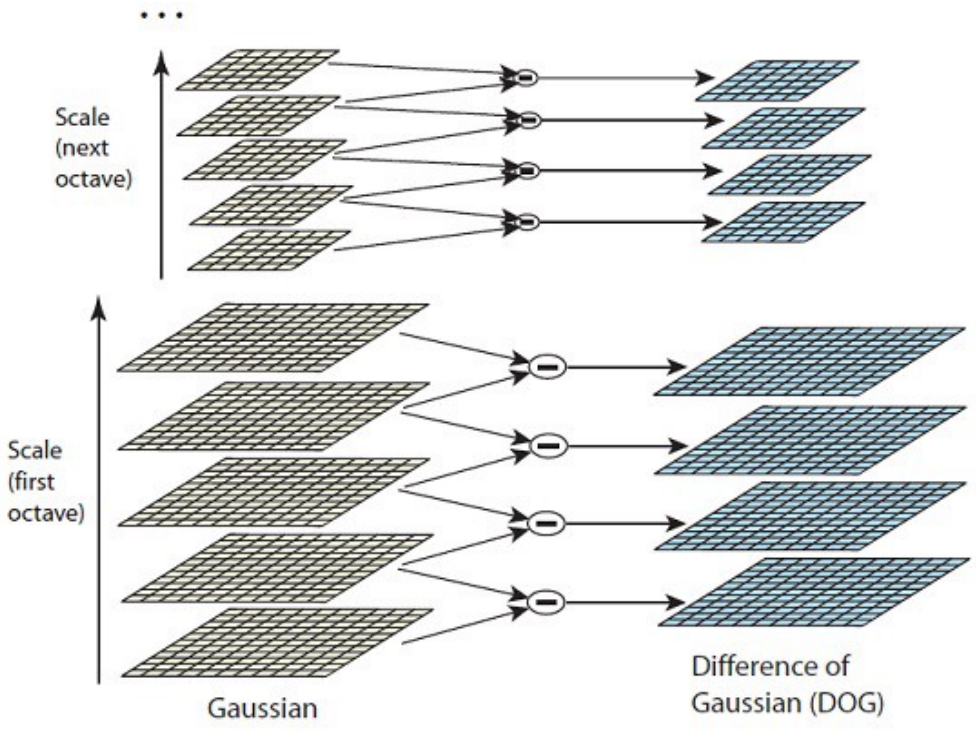


Figure 2.6: The figure shows for each octave the image is filtered with Gaussian blur with a varying σ. The DoG is calculated by subtracting the blurred images with each other. The resulting image is then used to find any local extrema. The figure is from [11].

The features are identified by checking the maximum and minimum for each pixel after the DoG filtering. This is accomplished by comparing the values at neighbouring pixels at the same variance σ, but also comparing the pixels for the scale above and below it [11]. However, further refinement is required to find the sub-pixel locations of the local extrema. This is achieved by using Taylor expansion of the scale-space up to the quadratic term around the potential feature and interpolate to find the sub-pixel location of the local extrema [11]. At this stage, local extrema with low contrast are removed by checking if the intensity level is below a certain threshold. Similar to the Harris corner detector, the edges are removed as potential features by analyzing the eigenvalues of the hessian matrix given by the previous Taylor expansion [11]. Here, the criterion is given by the ratio between the trace and the determinant of the hessian matrix instead of Equation (2.17). Rotation invariant is achieved by computing an orientation histogram for a region around the extracted feature. The histogram contains 36 bins where each sample is weighed

with its gradient magnitude. The highest peak and any peak within 80 percent of the highest peak will form the orientation of the feature.

2.4.2 Feature descriptor

Feature descriptors are used to describe the extracted features and are used to match common features that are present in different images. There are several different approaches to creating descriptors such as ORB [30] and BRIEF [31] descriptors. A commonly used descriptor is the SIFT descriptor [11], where the descriptor is calculated by creating a 16x16 grid centred around the extracted feature. The image gradient is computed for each cell in the grid and from these gradients an orientation histogram with eight bins are formed in each quadrant of the grid. This process can be seen in Figure 2.7, note that the figure only shows an 8x8 grid instead of a 16x16 grid. The four different histograms are stacked into a vector which becomes the resulting descriptor which is a vector with a length of 128 elements.

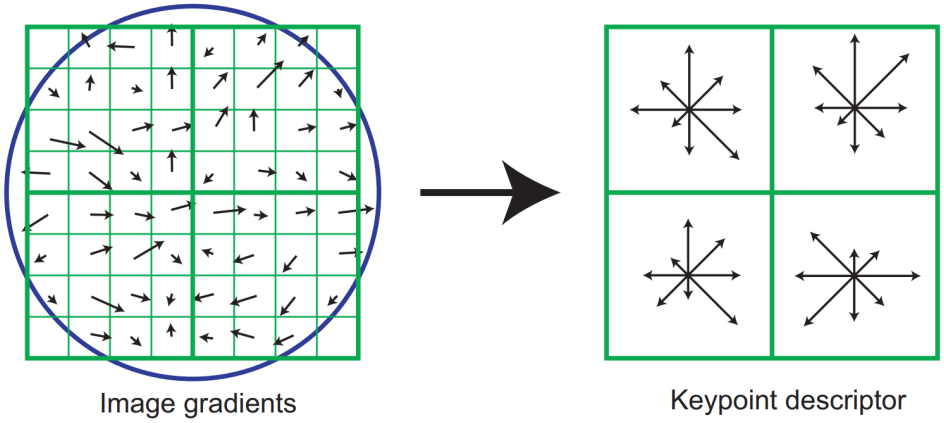


Figure 2.7: The figure to the left shows the grid that is centred around the detected feature. The arrows in the cells represent the magnitude and direction of the image gradients. The figure to the right shows the orientation histogram for each quadrant which is later stacked into a vector to form the descriptor. The figure is from [11].

2.4.3 Feature matching

Feature matching aims to match common features between images. The idea is to take one feature from one image and find the feature in the other image which has the lowest Euclidean distance between their descriptor vectors. To search through all the extracted features is quite time-consuming. Therefore, the search is approximated by using Best-Bin-First algorithm [32] to speed up the search process [11]. The high dimensionality of the descriptor vector causes many matches being to close to each other which potentially leads to incorrectly matched features [11]. A simple test to identify incorrectly matched features is to find the ratio between the match and the second-closest match. If they are above a certain pre-determined threshold between 0.6-0.9, they are discarded. The logic is that an incorrectly matched feature

will likely have other falsely matched feature that is similar to each other [11]. An example of matched features can be seen in Figure 2.8.

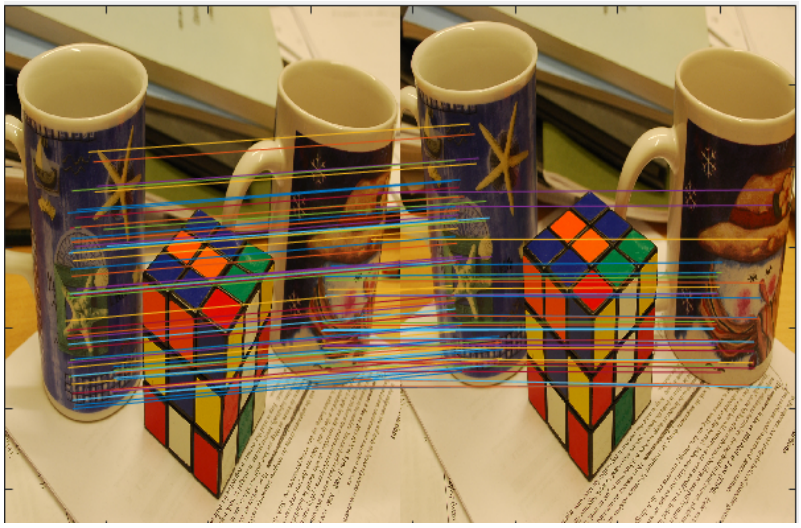


Figure 2.8: The figure shows matched SIFT features between two images that covers the same scene with two different view-points. The images are from [33].

2.5 Random sample consensus

Random sample consensus [25] (RANSAC) is an algorithm used to robustly estimate model parameters when using a data set containing outliers. These are data points that cannot be explained by the model parameters. The algorithm tries to filter out all the outliers from the data set by estimating the model parameters using the minimal amount of data points which is randomly selected. Each data point from the data set is verified whether it is an inlier or an outlier. The verification can be done in multiple ways depending on which type of model the algorithm tries to estimate the parameters for. For homography estimation reprojection error is commonly used [1] and if the error is within a certain pre-determined threshold then the point is counted as an inlier, otherwise it will be counted as an outlier. The process is repeated with new randomly selected data points to re-estimate the model parameters. The final solution is the parameters with the largest number of inliers.

In [25] presents an equation to compute how many iterations the algorithm needs to run to probabilistically guarantee that one sample will be outlier free which is given by

$$N = \frac{log(1 - p_{success})}{log(1 - (1 - p_{outlier})^s))}, \tag{2.20}$$

where $p_{success}$ is the probability of successfully finding the model parameter without using any outlier data and $p_{outlier}$ is the probability of an outlier in the data set and s is the number of data points needed to estimate the model parameters. The $p_{outlier}$ value is difficult to know before running the RANSAC algorithm, therefore it is estimated during the run-time of the RANSAC algorithm by finding the ratio

of the number of inlier points for a solution and the total number of data points. Then, a new N is computed by using Equation (2.20) and the newly estimated $p_{outlier}$ value.

2.6 Bundle adjustment

Stitching together a large set of images require consecutive multiplications of the pairwise homographies to transform all images into a common coordinate system. These consecutive multiplications result in an accumulation of error between the images and constraints between the images will be disregarded [7]. Bundle adjustment is a method used to reduce the accumulated error between images and ensure consistent global alignment between the images by solving a non-linear optimization problem [2]. More specifically, Bundle adjustment aims to find a set of parameters that minimize the re-projection error. The error is given by

$$\boldsymbol{r_{ij}^k} = \boldsymbol{\hat{u}_i^k} - \boldsymbol{\tilde{u}_{ij}^k}, \tag{2.21}$$

where $\boldsymbol{\hat{u}_i^k}$ is the k-th feature in image i and $\boldsymbol{\tilde{u}_{ij}^k}$ is the same feature that have been projected from image j into image i. The loss function for the optimization problem is the sum of all the squared residual errors [7] and is given by

$$e_{reproj} = \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in F(i,j)} \left\| \boldsymbol{r_{ij}^k} \right\|^2, \tag{2.22}$$

where $I(i)$ is the images that have matches with image i and $F(i, j)$ is the matched features between image i and image j. In image stitching application, Bundle adjustment tries to find the optimal homographies and feature points that result in the lowest total reprojection error by solving non-linear least squares problem. These problems are typically solved by using the Levenberg-Marquardt algorithm which incorporates properties from both Gradient descent and Gauss-Newton algorithms [1]. Gradient descent is a first order optimization method that computes the local minimum by iteratively taking small steps in directions where the cost function has the largest decrease which is the negative gradient. The update step is given by

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} - \alpha 2 \boldsymbol{J(\theta_k)^T r(\theta_k)}. \tag{2.23}$$

The Jacobian $\boldsymbol{J}$ is with respect to all parameters $\boldsymbol{\theta}$, namely the homographies and features points we want to optimize. With small enough step size α, the Gradient descent algorithm is always able to find a local minimum, but the convergence rate of the algorithm is quite slow when approaching the solution. In contrast, the Gauss-Newton algorithm has a faster convergence rate compared to the Gradient descent algorithm. The Gauss-Newton algorithm achieves the faster convergence rate by first approximate the Hessian matrix by

$$\boldsymbol{H} \approx \boldsymbol{J^T J}. \tag{2.24}$$

This is under the assumption that the residual is quite small or if it is approximately linear. The update step is therefore given by

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} - (\boldsymbol{J}(\boldsymbol{\theta_k})^T \boldsymbol{J}(\boldsymbol{\theta_k}))^{-1} \boldsymbol{J}(\boldsymbol{\theta_k})^T \boldsymbol{r}(\boldsymbol{\theta_k}). \tag{2.25}$$

This leads to a faster convergence than the Gradient descent algorithm and without the computational expense of calculating the second derivative of the residuals. However, this algorithm is quite unstable and fails if the initial values are far from the solution. Levenberg-Marquardt is an algorithm that is more stable algorithm than the Gauss-Newton algorithm and has a faster convergence rate than the Gradient descent algorithm [34]. The update step for the Levenberg-Marquadt is given by

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} - (\boldsymbol{J}(\boldsymbol{\theta_k})^T \boldsymbol{J}(\boldsymbol{\theta_k}) + \boldsymbol{\lambda I})^{-1} \boldsymbol{J}(\boldsymbol{\theta_k})^T \boldsymbol{r}(\boldsymbol{\theta_k}). \tag{2.26}$$

For large values of λ, the solver behaves as the Gradient descent algorithm and it guaranteed to find a lower value for the cost function. For smaller values of λ, the solver behaves as the Gauss-Newton algorithm and is able to converge to the solution quite fast. The general strategy for these solvers is to start with a large λ in the beginning when the solution is far from the minimum and decrease λ when it starts reaching the minimum [34].

2.7 Multi-band blending

Blending is a technique that is used in image stitching to remove seams caused by exposure differences and pixel miss-alignments between two or more images with overlapping regions [2]. Multi-band blending presented by [35] is a blending algorithm that uses image pyramids to blend images. The algorithm ensures that most seams are removed and no ghosting or blurring is present in the resulting image. The multi-band blending algorithm presented in [35] works by constructing a blending mask M for the region which is to be blended. Values are assigned to the pixels in the blending mask to indicated if a pixel is either from image I_A or I_B. The pixel value of 1 indicates that the pixel comes from image I_A while the pixel value of 0 implies that it comes from image I_B. Furthermore, Laplacian pyramids L_A and L_B are constructed from both images and a Gaussian pyramid G_M is created from the blending mask M. Creating a Gaussian pyramid makes it possible to weight each level in the Laplacian pyramids correctly by the corresponding level in the Gaussian pyramid [36]. The blended pyramid L_S is constructed using the weighted levels from the Laplacian pyramids where each single pyramid level in L_S is given by

$$L_S^l = G_M^l L_A^l + (1 - G_M^l) L_B^l, \tag{2.27}$$

where l is the pyramid level. The final blended image I_s can be obtained by interpolating and summing up the blended images in every layer of the L_S pyramid [19]. After the operations, image I_s will correspond to the image at the lowest level $l = 0$ in the pyramid.

3
Method

The goal of the thesis is to construct a digital 2D map over a planar indoor scene by only using images. The main idea is to stitch the images together to form the indoor map. An overview of the proposed pipeline can be seen in Figure 3.1.

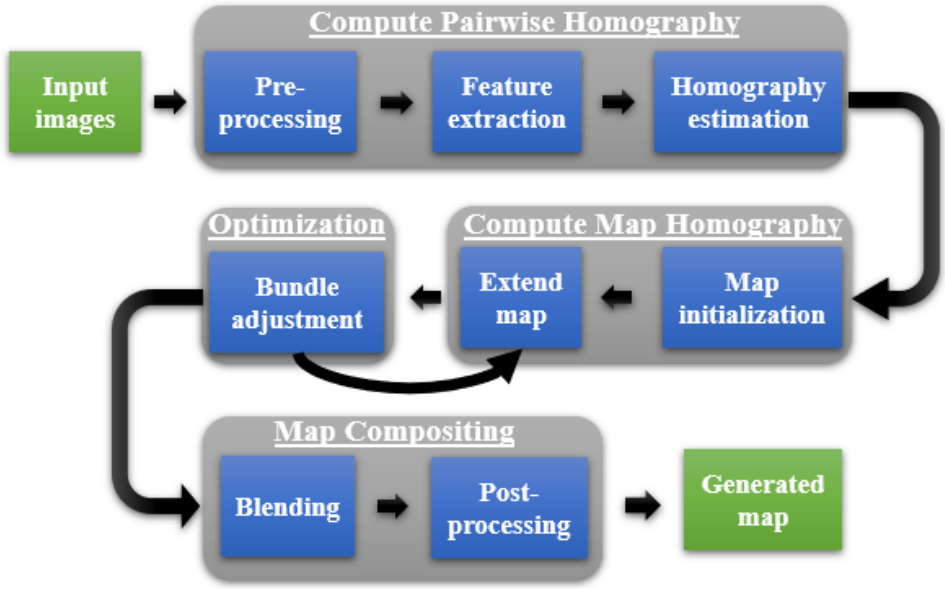


Figure 3.1: The figure shows the different stages from turning the input images into a 2D map. The first stage is to estimate pairwise homographies between images by finding point correspondences between them. The second stage consists of building the map by calculating homographies which transforms each image into the maps coordinate system. These homographies are called map homographies. The third stage is to optimize the map homographies and point correspondences for each image using bundle adjustment. The final stage is compositing the map by stitching all the images together using the calculated map homographies and converting the unit of the map from pixels into centimetres.

The first stage of the pipeline consists of estimating the pairwise homographies between overlapping images to relate the pixel coordinates from one image to another. Pairwise homographies are estimated by finding point correspondences between the images using feature detectors. The second stage consists of computing homographies that transform each image into a common coordinate system which will serve

as the coordinate system of the map. A coordinate system of a reference image is selected as the common coordinate system and all remaining images are consequently transformed into the reference image. Homographies transforming images into the reference image are called map homographies and are obtained by consecutively multiplying pairwise homographies between multiple images. Bundle adjustment is used to optimize all the map homographies and point correspondences after each image is transformed to the reference image. This is done to ensure global alignment between the images when they are stitched together. The final stage of the pipeline is to generate the map by using the optimized map homographies and multi-band blending to stitch the images together. Multi-band blending removes the seams between the images and a final post-processing step is needed to accurately convert the map units from pixels to centimetres.

This section will go into more details about the four stages of the pipeline.

3.1 Pairwise homography estimation

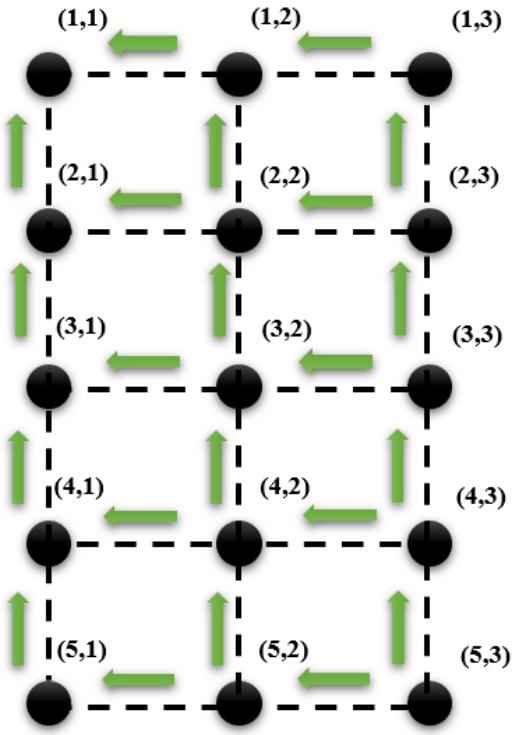


Figure 3.2: The figure shows an example of a birdseye view over the camera layout. Each black dot represents a camera and they are identified by the position in the grid system. The green arrows show the images each camera is matched with. For example, camera $(1, 1)$ image is matched with cameras $(1, 2)$ and $(2, 1)$ images.

The main idea of constructing the map is to stitch images together, therefore it is essential to estimate homographies that relate overlapping image pixel coordinates

from one image to another. Homographies that transforms between overlapping images are called pairwise homographies.

The first step is to pre-process the images to remove any radial distortion by using the radial distortions parameters since any distortion may lead to an incorrect estimation of the pairwise homography. The pipeline assumes the cameras are calibrated beforehand and that the radial distortion parameters are estimated during the calibration process for a pinhole camera model. Also, cameras are assumed to be positioned in a grid system, an example is shown in Figure 3.2. Images are identified according to the position of the camera in the grid system. Some additional pre-processing is required with images that have other highly texture planar surfaces except for the floor. For example, pallets with lids that are highly textured. These highly textured surfaces might lead to more point correspondences that are located between these planar surfaces instead of the floor. This causes the RANSAC algorithm [25] to estimate homographies between these planes and not the floor of the scene since there are more point correspondences between these planes. The problem is solved by letting the user manually add a black mask over these surfaces before inputting the images to the algorithm. In this way, the feature extractor is not able to detect features over these surfaces.

The next step is to form the point correspondences between overlapping images to estimate the pairwise homography using the DLT algorithm [1], as explained in Section 2.3. Point correspondences are formed by extracting features from each image by either using a SIFT feature detector [11] or a Harris corner detector algorithm [29]. Extracted features are matched by calculating a SIFT descriptor for each feature and searching in the descriptor space using nearest neighbour search to find the closet match. Each image is matched to the image below and to the left according to the arrows shown in Figure 3.2. Images located on the last row or column are only matched either to the image to the left or the image below it respectively. This reduces the computation time since the features for a single image is at most matched with features from two other images. The SIFT detector and descriptor along with the Harris corner detector and nearest neighbour search is implemented in C++ using the VLFeat library [37].

The matching process does not guarantee that all features will be correctly matched and any incorrectly matched feature will result in a poorly estimated homography. Therefore, the RANSAC algorithm [25] along with the DLT algorithm is used to robustly estimate the pairwise homographies. The DLT algorithm is implemented in C++ using the Eigen library [38]. The pseudo-code for how DLT and RANSAC algorithms are integrated together can be seen in Algorithm 1. The homography *BestH* is saved as the solution along with its associated inliers points.

3.2 Compute Map homography

All images need to be transformed into a common coordinate system to stitch them together into a single image. Pairwise homgraphies only describe the transformation

Algorithm 1 Homography estimation with DLT and RANSAC

```
for A pre-determined number of iterations do
    Randomly sample four point correspondences
    Compute H using DLT and the four point correspondences
    NumInliers = 0
    for All point correspondences do
        Compute reprojection error
        if reprojectionerror ≤ threshold then
            NumInliers ← NumInliers + 1
    if NumInliers > BestNumInliers then
        BestNumInliers ← NumInliers
        BestH ← H
```

between two image coordinate systems, therefore a homography that relates each image into a common coordinate system needs to be computed. These homographies are called map homographies.

The first step is to initialize the common coordinate system by selecting an image that will serve as a reference for the coordinate system. The top left image in the grid system is selected as the reference image. Then, the map homography for each image is computed where the map homography for the reference image is simply an identity matrix. Furthermore, the map homographies for images below and to the left of the reference image is their respective pairwise homography. For the rest of the images, their map homographies are computed by consequently multiplying the pairwise homographies. For example, the map homography for image j to the reference image i is obtained by

$$H_{ij} = H_{ik}H_{kj}, \tag{3.1}$$

where H_{ik} is the pairwise homography that relates images i with image k and H_{kj} is the pairwise homography that relates images k with image j. If there exist several different paths to obtain the map homography, then the path that has the pairwise homographies with the largest number of inliers is selected. The reason is that a homography with a large number of inliers is more correctly estimated and will more accurately transform the images.

3.3 Optimization

Multiple map homographies are computed by consequently multiplying several pairwise homographies. These multiplications lead to an accumulation of error and the constraints from the pairwise homographies are disregarded. The accumulated error results in a stitched image that is not globally consistent. The idea here is to use bundle adjustment to reduce the error and enforcing the constraints for the homographies. This ensures that the stitched image is globally consistent and ultimately results in a more accurate map.

The bundle adjustment problem is initialized using the reference image and a nearby image that it has the largest number of inliers matches with. All matched points from the reference image are added to a 2D map where the coordinate system is the same as the reference image. Each point in the 2D map is associated with its corresponding point in the reference image and the nearby image. The homographies from both images are initialized by using the computed map homography from each image. The bundle adjustment optimizes the map homographies and map points by minimizing the reprojection error. The error is given by converting the 2D map points into homogeneous coordinates and projecting them into both images using the inverse map homography for each image. Before the residual error is computed, the projected points are converted back into inhomogeneous coordinates. The residual error is given by

$$\boldsymbol{r}_{\boldsymbol{i}}^{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{k}} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{k}}, \tag{3.2}$$

where $\boldsymbol{r}_{\boldsymbol{i}}^{\boldsymbol{k}}$ is the residual error between the actual image point $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{k}}$ and the projected point $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{k}}$ for the k-th map point in image i. The error function for the bundle adjustment is given by

$$e_{reproj} = \sum_{i=1}^{n} \sum_{k \in F(i)} h(\boldsymbol{r}_{\boldsymbol{i}}^{\boldsymbol{k}}), \tag{3.3}$$

where n is the number of images and $F(i)$ is the number of map points that is visible in image i. In this case n is only two images since the map is being intiliazed. Furthermore, a Huber loss function is used to make bundle adjustment more robust towards outliers [7] since some of the 2D point correspondences are incorrectly matched. The Huber loss function is defined as in [39] and is given by

$$h(\boldsymbol{r}) = \begin{cases} |\boldsymbol{r}|^2 & \text{if } |\boldsymbol{r}| < \delta \\ 2\delta|\boldsymbol{r}| - \delta^2 & \text{otherwise} \end{cases}, \tag{3.4}$$

where δ is a hyperparameter that decides the transition between a quadratic and a linear loss function.

After the two first images have been optimized, a new image is added that has point matches with one of the already optimized images in the 2D map. If the feature of the matched points does not already exist in the 2D map then the map is extended by projecting the matched inlier points of the existing image. The newly extended map points and the map homography of the newly added image along with the previously added map homographies are optimized by minimizing the reprojection error. This process repeated until all images have been added and all map homographies have been optimized. The bundle adjustment is implemented in C++ using the Ceres library [40].

3.4 Map Compositing

The final stage in the pipeline is to generate the map by stitching all of the images together by using the calculated map homographies. The generated map needs to

be rescaled to ensure that each pixel corresponds to one centimetre.

The main stage in the map compositing is to warp each image using the calculated map homography. This process will create seams between the warped images since the exposure differs between each image. Therefore, multi-band blending [35] is used to remove the seams. The images are warped using the map homography and before they are stitched together, a binary mask is created for each image. The binary masks have the same size as the final stitched image and in each mask the value 1 is assigned to the pixels where the corresponding warped image will be located while all other pixels are assigned the value 0. The masks encode the overlapping regions of each image and makes it possible to distinguish the origin of each pixel in the stitched image. Using the masks, the stitched image is blended. The warping and blending process is implemented in C++ using the OpenCV library [41].

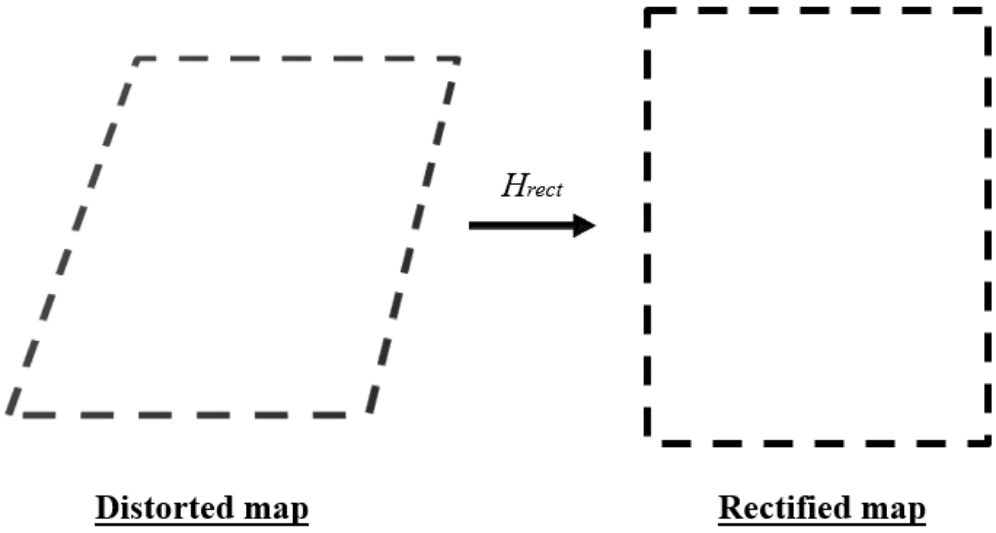


Figure 3.3: The figure shows an illustration of a distorted map to the left and the rectified map after applying the estimated homography.

The stitched image is from the perspective of the reference image. If the camera of the reference image is not pointed straight downwards, the generated map will have some distortions and needs to be rectified as illustrated in Figure 3.3. The rectification is done by selecting four or more points from the stitched image where the world positions are known. The pipeline assumes that the positions are given in centimetres. From the selected points, a homography can be estimated using the DLT algorithm [1]. With the estimated homography it is possible to rectify the stitched image. Additionally, the homography scales the image so one pixel corresponds to one centimetre since real world points used to estimate the homography are given in centimetres. The procedure is implemented in MATLAB since it offers a function called *ginput* to manully select points on the image.

4

Results

In this section, the results for the map generation pipeline will be presented. The map is evaluated accordingly to how accurately the map represents the real world indoor scene. Details of how the accuracy is measured along with the experimental setup and implementation settings will be presented.

4.1 Experimental Setup

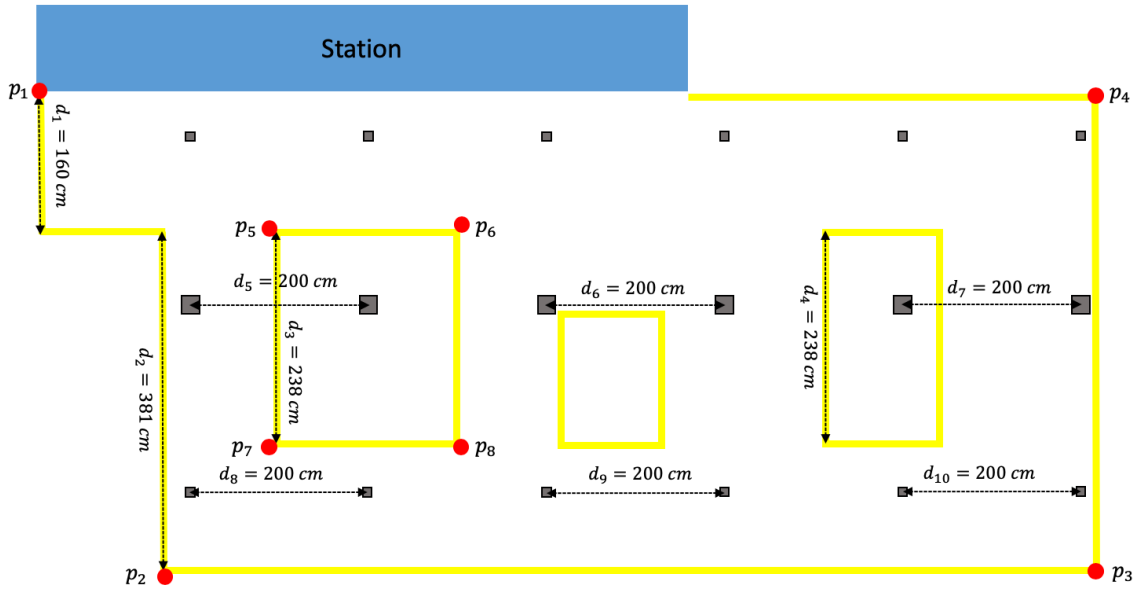


Figure 4.1: The figure shows a schematic overview of the indoor scene used to evaluate the map generation pipeline. The ten black arrows indicate distances between points that are used to evaluate the accuracy of the map. The red circles are points use to estimate the rectification homography. All the distances in the figure are measured in centimeters.

Two sets of ten images over an indoor scene are used to evaluate the performance of the proposed pipeline. The images are collected using ten stationary cameras mounted in the ceiling over an indoor scene in Assar arena in Skövde. The first set of images obtained from the cameras contain an indoor scene with yellow lane markings and a homogenous floor which contributes to a scene similar to an environment in an industrial factory. Therefore, the indoor scene gives a strong indication of how

well the pipeline would perform in a real world industrial setting. The second set of images is from the same indoor scene but with industrial objects spread out through the scene. An overview of the indoor scene without objects can be seen in Figure 4.1. Furthermore, all cameras are calibrated beforehand and are positioned in a grid system with five rows where each row has two cameras. All images in the set have the same resolution of 2304×1536 pixels and are captured simultaneously. The two sets of images used in the evaluation can be seen in Appendix A.

4.2 Evaluation Metric

The main evaluation metric is the accuracy of the generated map since it measures how well the map depicts the indoor scene. The accuracy is computed by having prior knowledge of the real world distances between several points in the generated map. From these points, the error is computed between the real world distance and the distance measured in the map. A more accurate measurement of the overall map accuracy is computed by selecting several distances that are spread out through the map and finding their distance error. For example, only computing the error for a single distance in the map will only show the accuracy of that specific section in the map. Another reason for using multiple distances is to reduce the influence of the error that occurs when manually selecting points in the map. The error is computed by using the ten distances that are represented as black arrows in Figure 4.1. Using these errors, the Root Mean Square (RMS) error is computed by

$$e_{RMS} = \sqrt{\sum_{i=1}^{N} \frac{(d_i - \hat{d}_i)^2}{N}}, \tag{4.1}$$

where d_i is the measured distance in the map and $\hat{d}_i$ is the real world distance. The RMS error is the performance metric used to evaluate the generated maps.

Furthermore, the robustness of the map generation is also evaluated since RANSAC introduces randomness in the pipeline and the generated maps should consistently be accurate. The metric used to measure the robustness is the spread of the data. More specifically, it is the spread of the mean accuracy for ten different generated maps and the spread of the mean distance error for each distance. A low spread corresponds to a robust map generation, while a high spread corresponds to a less robust map generation.

Additionally, other metrics are used to evaluate the impact of the feature extractors on the pipeline. Firstly, the ability of the feature detector to extract features from the input images and form point correspondences are studied. This is measured by computing three averages: Number of extracted features per image, matched features per image and number of inliers per image. Furthermore, the quality of the point correspondences are studied by measuring the average track length which indicates how well the feature extractors are able to extract distinctive features that are present in multiple images. The average cost per map point gives a good

indication of how accurately the pairwise homographies are estimated using different feature extractors.

4.3 Implementation details

In this section, the different parameters and settings used for evaluating the map generation pipeline are presented. The SIFT extractor uses 10 octaves with 5 scale levels at each octave since using these parameters the SIFT extractor was able to extract many features from the input images. A boundary margin of 1 pixel is used for the Harris corner detector since the corner of the image should not be detected as a corner. The peak and edge threshold was manually selected to 1 and 10000 respectively. With these parameters, the detector was able to extract many features while keeping the run-time at a reasonable level. The RANSAC loop is run for 30000 fixed iterations and the error threshold is 5 pixels. These values are manually chosen when tuning the pipeline as these values gave robustly estimated homographies that were quite accurate. During the matching process, at most 1000 leaf nodes comparisons will be made per query during the k-d tree search in order to the keep the run-time fairly low. The ratio threshold used to test the two closet matches is selected to be 0.7 as it will keep most of the correctly matched features while removing most of the incorrectly matched features [11]. The δ parameter for the Huber loss is 5. Initially, the value was larger but through trial and error it was finally selected to by 5 since it gave the best results. A summary of all the different parameters can be found in Table 4.1.

Function	**Parameter**	**Value**
SIFT	*Number of octaves*	10
SIFT	*Number of scale levels*	3
Harris	*Peak threshold*	1
Harris	*Edge threshold*	10000
Harris	*Boundary margin*	1 pixel
RANSAC	*Threshold*	5 pixels
Matching	*Max number of visited leaf nodes*	1000
Matching	*ratio threshold*	0.7
Bundle adjustment	*Boundary transition (δ)*	5

Table 4.1: The table shows the different parameter values used for evaluating the map generation pipeline.

4.4 Evaluation

In this section, the map generation is evaluated by conducting six experiments using the experimental setup described in Section 4.1 and the evaluation metrics described in Section 4.2. Two of the experiments aim to investigate how the pipeline performs on input images with and without masked objects in them. The other four experiments aim to investigate how different components of the pipeline affect the outcome

for the map generation. In particular, two components are singled out as highly interesting, namely the feature extraction and bundle adjustment. Matched features between the images are the point correspondences used to estimate the pairwise homographies. If the features are incorrectly matched or if they are not located on the floor of the indoor scene then the pairwise homography will be poorly estimated and lead to an inaccurate map. Therefore, the feature extractor needs to extract enough features that are both correctly matched and located on the floor of the indoor scene. The SIFT feature detector [11] along with the Harris corner detector [29] is evaluated accordingly to the accuracy of the map and the robustness of the map generation. Furthermore, the influence of bundle adjustment on the generated map is also evaluated. In particular, to study how much the optimized map homographies further improves the accuracy of the map and if the robustness of the map generation improves.

4.4.1 Performance of bundle adjustment

This section studies the impact bundle adjustment has on the proposed pipeline. In particular, to what extent does the map accuracy and robustness increase when optimizing the initial map homographies using bundle adjustment. The impact of bundle adjustment is investigated by generating four different sets of maps. The first two sets of maps are generated using Harris corner detector and SIFT feature extractor while bundle adjustment is omitted from the pipeline. While the remaining two sets of maps are generated using the same feature extractors but this time bundle adjustment is included in the pipeline. Two different feature extracts are used to investigate the consistency of the bundle adjustment since they might yield different initial map homographies. This shows the impact of bundle adjustment regardless of switching between feature extractors in the pipeline.

Additionally, each set of maps contains 10 generated maps using the same input images. Therefore, in total 40 maps are evaluated by computing distance errors for the marked distances in Figure 4.1. Bar charts for the two sets of maps using Harris corner detector with and without bundle adjustment can be seen in Figure 4.2. Each chart visualizes the mean error for each distance respectively, where the mean value is derived through ten different generated maps. Also, error bars are shown to indicates the variability of the measurements for each distance. Maps generated with bundle adjustment have overall a lower error for nearly all the distance when compared to the maps generated without bundle adjustment. This clearly shows that bundle adjustment is able to improve the map homographies by using nonlinear optimization. The only exception is the distance d_9 which has a slightly higher error when bundle adjustment is included in the pipeline. The reason why it did increase could be that the error is already fairly low if compared to all other distances and bundle adjustment could not significantly further improve the result. Therefore, the small increase could possibly come from when manually choosing the points on the map when measuring the error for distance d_9.

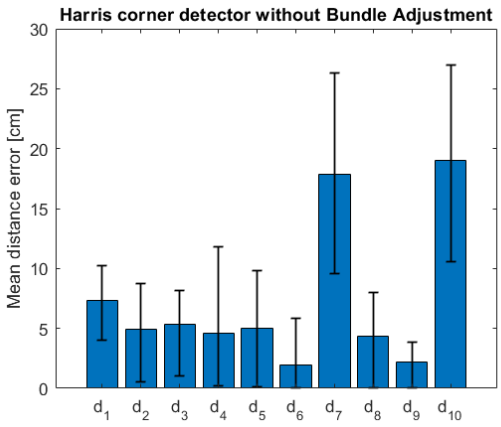
Another observation that can be made is that the error distances of d_7 and d_{10}

are considerably larger compared to the rest of the distances in Figure 4.2. This is mainly due to the poor quality of the feature correspondences between the input images where distance d_7 and d_{10} are located in. Therefore, the pairwise homographies between these images are poorly estimated which later in the pipeline contributes to a poorly computed initial map homography. However, bundle adjustment is still able to bring down the mean error of the measurements for both distances even though poorly initial map homographies are provided. By having better initial map homographies the mean distance error can be further decrease using bundle adjustment since the cost function used is non-convex. A better initial map homography leads to the optimizer finding a solution located at a lower local minimum.

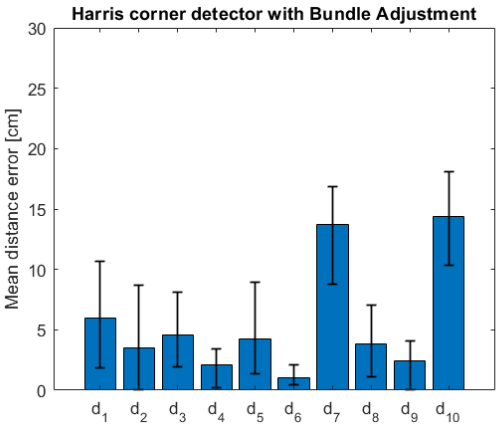
Results for the second set of maps using SIFT feature extractor instead of Harris corner detector can be seen in Figure 4.3. The result follows a similar pattern as the result from the Harris corner detector, mainly that bundle adjustment lowers the mean distance error for each measured distance . The impact of bundle adjustment is more visible in this case since it reduces the error considerably more compared to the equivalent results for the Harris feature detector. This further demonstrates bundle adjustment ability to improve the accuracy even if the initial map homographies are poorly estimated.

Overall, results in Figure 4.2 and Figure 4.3 shows a clear improvement for the map accuracy when including bundle adjustment regardless of feature extractor. Furthermore, this behaviour can be confirmed by looking at the final generated maps for each feature extractor. Figure 4.4 shows the final maps when using Harris corner detector and Figure 4.5 shows the map for the SIFT feature extractor. Both figures show that bundle adjustment significantly improves the alignment of the images, specifically for the lanes at the top of the maps.

When it comes to the robustness, using bundle adjustment yields a pipeline which is a more robust regardless of what feature detector is applied. This can be seen in both Figure 4.2 and Figure 4.3 where the variability for most distances is either lower or similar after bundle adjustment has been included in the pipeline. More specifically, the impact on the robustness can especially be seen in the SIFT case where the variability of distance d_7 and d_{10} is drastically reduced after bundle adjustment is applied. In other words, bundle adjustment manages in the SIFT case to drastically improve the poorly estimated initial map homographies caused by poor quality features and randomness from RANSAC. However, the variability for distance d_1 increases even if the mean error distance decreases for both the Harris and SIFT case after bundle adjustment is applied. This increase in variability is likely caused when computing the measured distance manually in the map since the starting point of d_1 is not always as clear as the starting points for other distances.

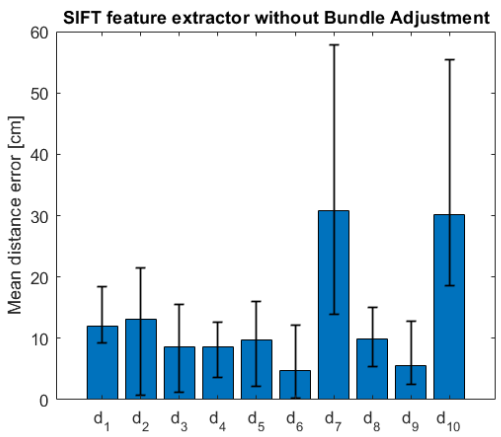


(a) The figure shows the results without using bundle adjustment.

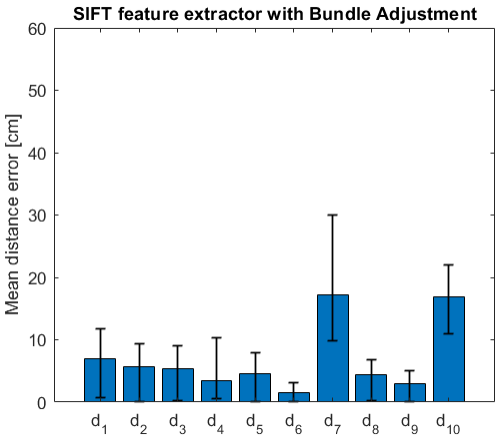


(b) The figure shows the results with bundle adjustment.

Figure 4.2: The figure shows the mean distance for all specified distances in Figure 4.1 when using Harris corner detector. The results for not using bundle adjustment can be in Figure 4.2a and the results for using bundle adjustment can be seen in Figure 4.2b. These figures clearly show that bundle adjustment reduces the distance error for all distances except for d_1. Also, the variability for the distance errors decreases which indicates that bundle adjustment increases the robustness of the map generation.

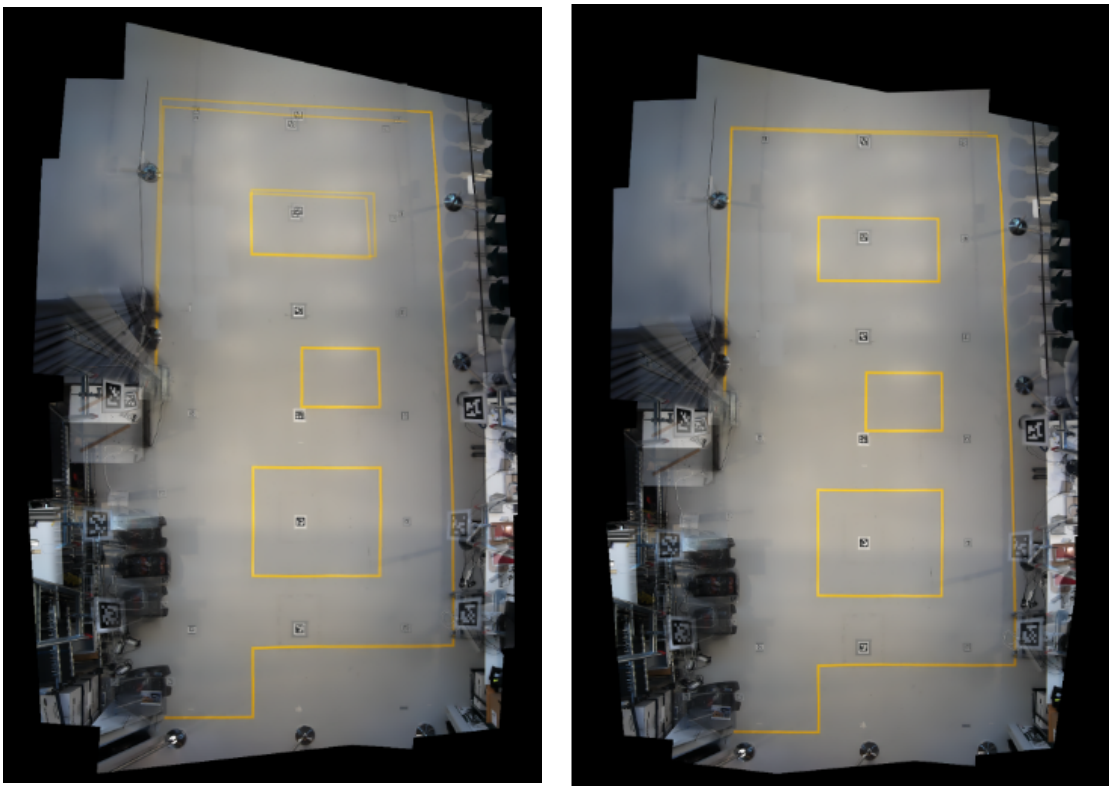


(a) The figure shows the results without using bundle adjustment.



(b) The figure shows the results with bundle adjustment.

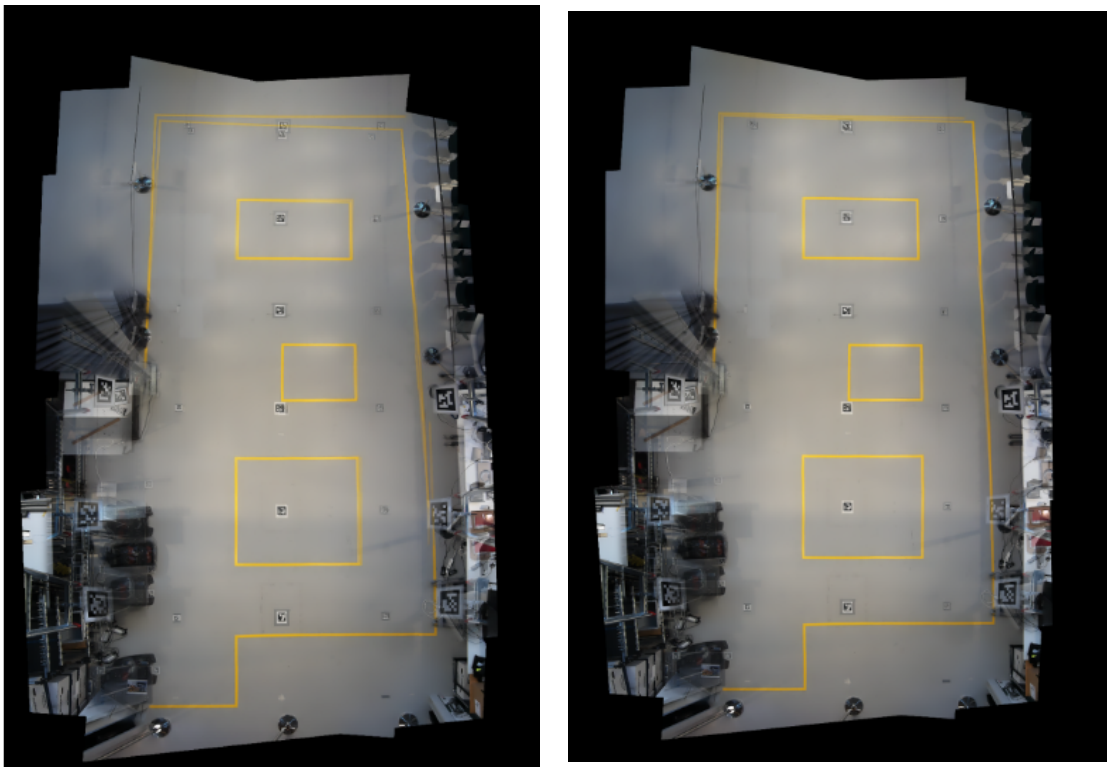
Figure 4.3: The figure shows the mean distance for all specified distances in Figure 4.1 when using SIFT feature extractor. The results for not using bundle adjustment can be in Figure 4.3a and the results for using bundle adjustment can be seen in Figure 4.3b. These figures clearly show that bundle adjustment reduces the distance error for all distances except for d_1. Also, the variability for the distance errors decreases which indicates that bundle adjustment increases the robustness of the map generation.



(a) The figure shows the generated map without using bundle adjustment.

(b) The figure shows the generated map using bundle adjustment.

Figure 4.4: The figure shows the effect of bundle adjustment when using Harris corner detector. Figure 4.4a shows the generated map without using bundle adjustment and Figure 4.4b shows the result with bundle adjustment. The figures clearly show that the map generated with bundle adjustment more accurately portrays the indoor scene. This can clearly be seen at the top of the map where the bundle adjustment ensures the lanes are more aligned.



(a) The figure shows the generated map without using bundle adjustment.

(b) The figure shows the generated map using bundle adjustment.

Figure 4.5: The figure shows the effect of bundle adjustment when using SIFT feature extractor. Figure 4.5a shows the generated map without using bundle adjustment and Figure 4.5b shows the result with bundle adjustment. The figures clearly show that the map generated with bundle adjustment more accurately portrays the indoor scene. This can clearly be seen at the top of the map where the bundle adjustment ensures the lanes are more aligned.

4.4.2 Comparasion of feature extractor

The previous subsection concluded that bundle adjustment improves the accuracy and robustness of the pipeline regardless of the feature extractor. In this section, the influence of feature extractor on the proposed pipeline is further investigated in detail. In particular, the performance of two different features extractor is investigated, namely Harris corner detector and SIFT feature extractor. The performance of the extractor will be evaluated according to the overall accuracy of the generated maps and the robustness of the map generation.

The evaluation is based on the two map sets that include bundle adjustment in its pipeline from the bundle adjustment experiments in Section 4.4.1. However, instead of looking at the individual distance errors, an RMS error is computed using all the distances in each map. The RMS error computed for each map in a set is used to compute an average RMS error for the entire set. In this way, a single metric gives us the overall accuracy of the maps outputted from the pipeline given that different

feature extractors are used. Additionally, to further study the effect that the feature extractors have on the map accuracy, seven different metrics have been computed for one map generation as seen in Table 4.2.

Results show that Harris corner detector achieves a higher map accuracy compared to SIFT feature extractor. This can be seen in Figure 4.6 that shows the average RMS error for an entire map set. The average RMS error is around 20% lower when using Harris corner detector compared to using SIFT feature extractor. A lower RMS implies that Harris corner detector contributes to a pipeline that generates more accurate maps. The reason for the improved accuracy is possible due to the fact the features from the Harris corner detector are able to form more correct point correspondences between images resulting in a more accurate estimated pairwise homography.

Analyzing Figure 4.6 further shows that the spread of the average distance RMS error is lower for Harris corner detector compared to SIFT feature extractor. This indicates that having Harris corner detector in the pipeline contributes to a more robust map generation in terms of accuracy. The reason for a more consistent map generation could be explained by the number of features and matched features per image seen in Table 4.2. On average Harris corner detector extracts and matches four times more features per image then SIFT feature extractor. Furthermore, Table 4.2 shows the average total number of inliers per image. Harris corner detector has approximately seven times more inliers per image. This suggests that Harris corner detector extracts considerably more quality features between images that contribute to better pairwise homographies.

Additionally, the cost per map point before and after bundle adjustment suggests there are no significant differences between the two feature extractor as seen in Table 4.2. However, the results from Figure 4.6 clearly shows a difference in accuracy between the feature extractors. A possible reason for this behavior is that Harris corner detector extracts features that are more distributed throughout the indoor scene in each image. While the SIFT feature extractor finds more features at specific regions in the indoor scene in each image. Having features correspondences at specific regions leads to homographies that are overfitting the map to the regions where the features are extracted from. Therefore, in the SIFT case the homographies are optimized for a certain region of the map resulting in a similar cost per map point as in the Harris case but the overall map accuracy is lower. In other words, Harris corner detector has a better overall map accuracy as the inliers used to compute the homographies are distributed throughout the map and not at specific regions of the map. However, both feature extractors have problems with finding common distinct features that are present in more than two images as seen from the average track length in Table 4.2. Both Harris corner detector and SIFT feature extractor have on average only one distinct feature that is present two images.

Overall, the results show that Harris corner detector performance is better at extracting features from an indoor scene resembling an industrial factory. Extracting

features that a more distributed throughout the indoor scene results in an estimated homography that is not overfitted for certain regions of the map. Improved pairwise homographies yield more accurate initial map homographies that results in a more accurate map since bundle adjustment receives a better initial solution. Also, Harris corner detector has considerably more map points that are distributed throughout the map which contributes against overfitting the estimated homographies. Therefore, Harris corner detector is a better choice of feature extractor in the proposed pipeline compared to SIFT feature extractor.

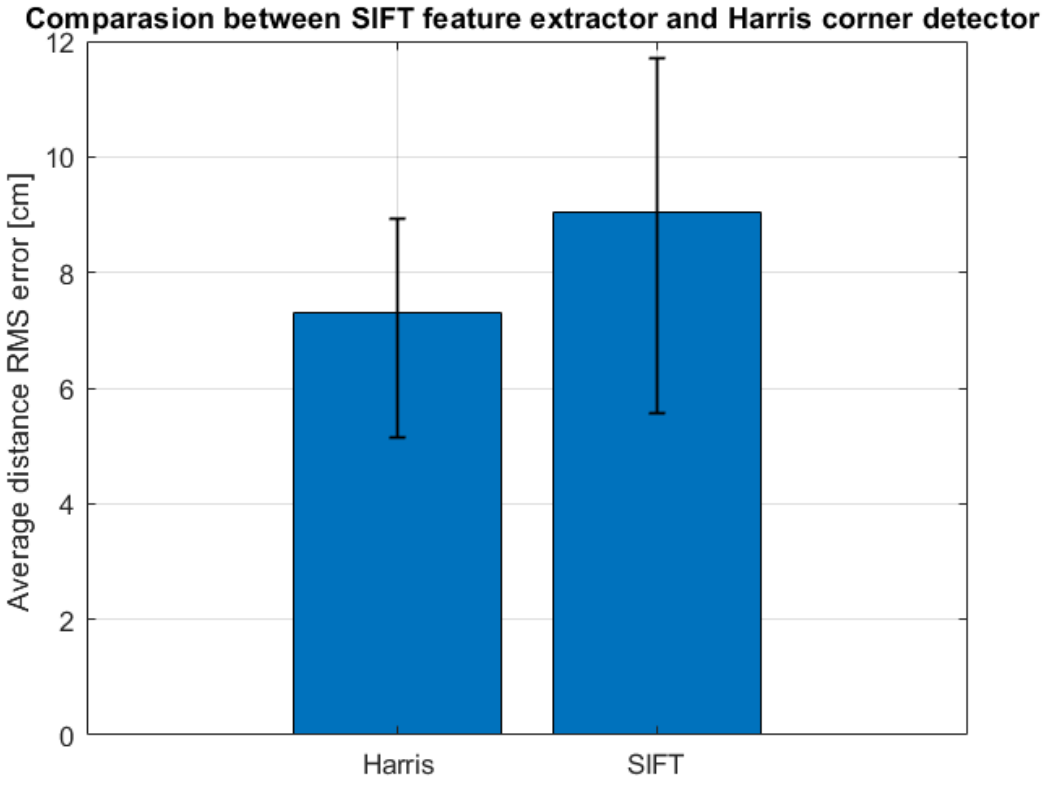


Figure 4.6: The figure shows a bar chart over the average RMS distance error when either using SIFT feature extractor or Harris corner detector in the pipeline. The result shows that the Harris corner detector has lower error compared to SIFT feature extractor. Therefore Harris corner detector generates more accurate maps. Also, Harris corner detector is more robust since the variability of the data is lower.

Metric	SIFT	Harris
Avg. extracted features per image	19130	73091
Avg. matches per image	182	719
Avg. inliers per image	30	216
Total map points	312	2199
Avg. track length	2.076	2.062
Cost per map point before B.A.	3.867	3.247
Cost per map point after B.A.	1.348	1.211

Table 4.2: The table shows seven metrics used for comparing the different feature extractors used in the pipeline for one generated map. The first three metrics shows the average extracted features, matched features and inliers points per image. In these metrics, Harris corner detector extracts more features than the SIFT feature extractor. The last four metrics are the total map points, average track length and the cost per map point before and after bundle adjustment. Both feature detectors perform similarly for average track length and cost per map point. SIFT feature detector has a lower total map points since it has fewer inliers points per image compared to Harris corner detector.

4.4.3 Indoor scene with objects

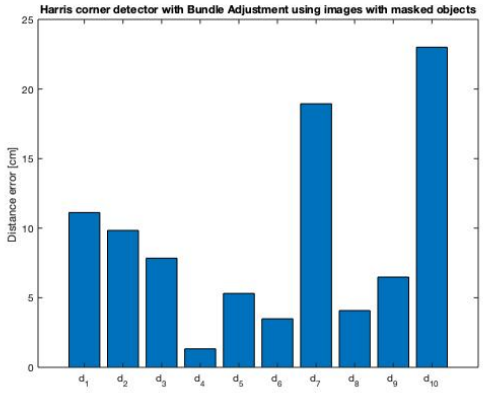
The proposed pipeline is also evaluated on the set of images containing industrial objects in the scene in order to see how the overall accuracy of the map is affected when objects are present. Additionally, the masking step in preprocessing described in Section 3.1 is also evaluated to illustrate how it affects the overall map accuracy. Two pallet objects in the scene are masked for this experiment. However, the performance will only be evaluated when Harris corner detector and bundle adjustment is included in the pipeline. This is the case since it was concluded in Section 4.4.1 and Section 4.4.2 that the pipeline had the highest accuracy and robustness when both bundle adjustment and Harris corner detector were used.

The overall RMS error for generating one map using images that have been masked during the preprocessing can be seen in Table 4.3. The RMS error for the generated map is as before computed using all ten distances on the map and the error for each distance can be seen in Figure 4.7a. The RMS error of the map that gave the highest accuracy in the map set using bundle adjustment and Harris corner detector in Section 4.4.2 can also be seen in Table 4.3. Each distance error used to compute the RMS error for the generated map without objects can also be seen in Figure 4.7b. It is clear from Table 4.3 that the overall accuracy of the map decreases with approximately 50 % when objects are present in the scene. The difference is even more clear when comparing the distance error for each distance in figure 4.8a. For each distance, the error increases when objects are present in the scene even if masking is used. This decrease in accuracy was expected since the feature extractor manages to detect and match a few distinct features located on some of the objects. The RANSAC algorithm does not always manage to filter out these outliers which contribute to poorly estimated pairwise homographies between a few images. This

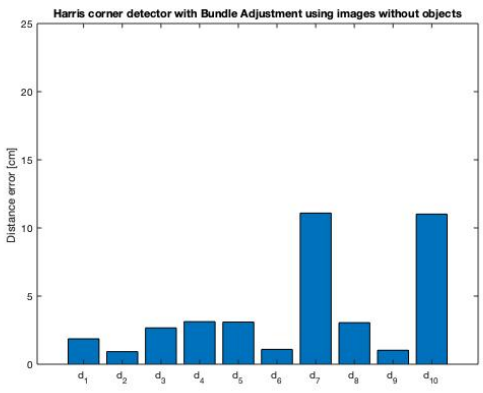
further contributes to an error that stacks up when map homographies are computed by consecutively multiplying pairwise homographies.

Furthermore, generating a map with and without masking the pallets in the scene can be seen in Figure 4.8 . Looking closer at the map where masking was not used it is clear that all distances are not clearly defined which makes it hard to compute a valid RMS error. Therefore it is not possible to compare the overall accuracy using the RMS between these two cases. However, analyzing both images in Figure 4.8 visually, it can be seen that the map generated without masking does not accurately portray the indoor scene. The reason for failure when not masking the pallets is that the RANSAC algorithm [25] finds more inliers for the pairwise homographies transforming points between the planes given by the lid of the pallets and not the plane given by the floor. Therefore in Figure 4.8a objects are less blurry since the map homography aligns the plane given by the objects and not the floor. While in Figure 4.8b the objects are more blurry since the map homography aligns the plane given by the floor and not the objects.

The results indicate that objects with a large surface representing a plane apart from the floor surface need to be masked if one wants to compute a map that accurately portrays an indoor scene with objects in it. Even with masked objects, the generated map is less accurate compared to the case where no objects are present in the scene.

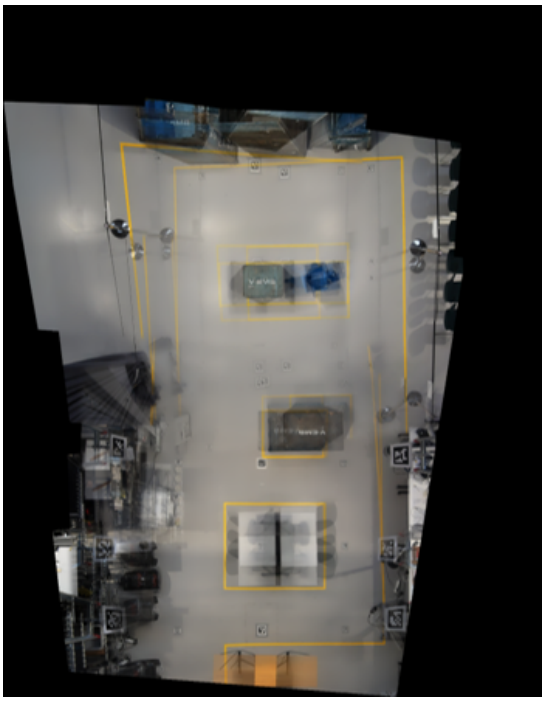


(a) The figure shows the result when using images with masked objects.

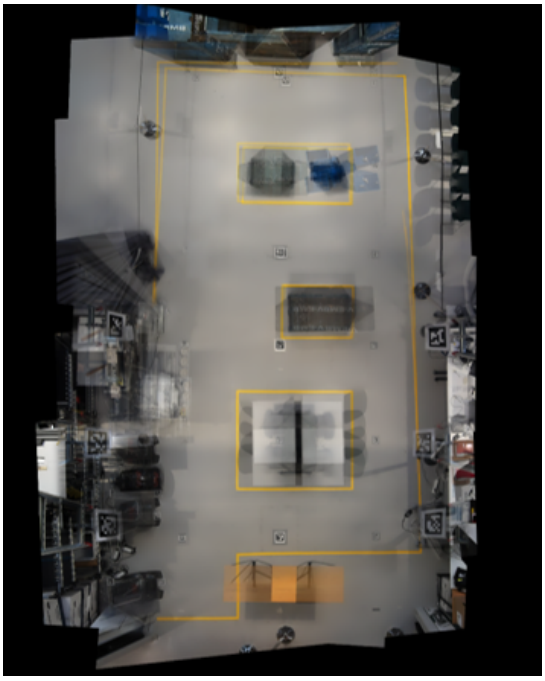


(b) The figure shows the result when using images without objects.

Figure 4.7: The figure shows the ten distance errors of two different generated maps using images with and without industrial objects placed in the scene. Both maps are generated using the pipeline that includes Harris corner detector and bundle adjustment. Two blue pallets in the input images are masked during preprocessing when generating the map with objects. The results show that the distance error for all distances increases for the generated map when objects are present in the scene.



(a) The figure shows the generated map without masking input images.



(b) The figure shows the generated map with masked input images.

Figure 4.8: The figure shows two maps generated using images with industrial objects placed in the scene. Both maps are generated using the pipeline that includes Harris corner detector and bundle adjustment. Figure 4.8a shows the generated map when the two blue pallets in the input images are not masked during preprocessing. Figure 4.8b shows the generated map when the two blue pallets in the input images are masked during preprocessing. The figures clearly show that masking the two pallets contributes to a map that accurately portrays the indoor scene. The overall accuracy for Figure 4.8b can be seen in Table 4.3.

Indoor scene	**RMS error [cm]**
With objects + masking	11.27
Without objects	5.36

Table 4.3: The table shows the RMS error of two different generated maps using images with and without industrial objects placed in the scene. Both maps are generated using the pipeline that includes Harris corner detector and bundle adjustment. Two blue pallets in the input images are masked during preprocessing when generating the map with objects. The results show that the RMS error increases for the generated map when objects are present in the scene. This indicates that the accuracy of the generated maps decreases using images with objects in them.

4.5 Summary

The results clearly show the pipeline is able to generate a highly accurate map using images over an indoor as seen in Figure 4.6. From the results, it is clear the best performing pipeline included bundle adjustment and had Harris corner detector as a feature extractor, as seen in Figure 4.2. However, the result also shows that the pipeline fails to generate a map when the indoor scene contains objects with highly textured planar surfaces. The pipeline is able to generate a map if these objects are masked but the overall accuracy of the map decreases when compared to a map that was generated for a scene without any objects. Overall, the pipeline is able to generate an accurate map over an indoor scene if the scene does not contain too many planar textured objects.

The run-time for the pipeline can be seen in Table 4.4, which clearly shows the pipeline that uses the SIFT feature extractor is considerably faster than the pipeline with Harris corner detector. This expected since the Harris corner detector extracts four times more features than the SIFT feature extractor, as seen in Table 4.2. Therefore, it will increase the run-time for the other stages since there will be considerably more points. However, the runtime of the map generation was not considered when designing the pipeline and therefore there are many potential improvements to decrease the run-time.

Pipeline stage	**Harris** [s]	**SIFT** [s]
Feature extraction	1569.34	89.67
Matching + RANSAC	1818.00	538.75
Bundle adjustment	4.99	0.62
Multi-band blending	5.47	2.58
Total time	3406.18	637.031

Table 4.4: The table shows the overall run-time along with individual stages of the pipeline using either SIFT or Harris corner detector as a feature extractor. The pipeline with Harris corner detector is considerably slower than the pipeline with SIFT feature extractor. Note that the matching process and the RANSAC are performed simultaneously and could not be measured separately.

5
Conclusion

This thesis has investigated a new pipeline that generates a 2D map that could be used for indoor navigation and localization within complex environments. The pipeline utilizes mainly projective planar transformations that relate pixel coordinates from one image to another. These transformations are used to stitch all images of the indoor scene into a single image that captures the entire indoor scene. Different components of the pipeline were evaluated to study their influence on the generated map. The results show that the maps generated by the pipeline increase in accuracy and robustness when bundle adjustment is included. Furthermore, the highest accuracy and robustness was achieved when combining Harris corner detector with bundle adjustment in the pipeline. These results show the importance of finding feature correspondences that are correctly matched and are evenly distributed throughout the floor. The latter is more important since it will contribute to a map that is overall accurate and not only at specific regions of the map.

The pipeline was further evaluated on an indoor scene with objects present on the floor. Overall, the pipeline's accuracy was lower compared to the scene without objects. Also, the results show that the input images needed to be masked to successfully generate a map. This is probably due to the RANSAC algorithm estimating homographies for other planar surfaces caused by the objects. This thesis has shown it is possible to generate an accurate map for an indoor scene using images captured from the ceiling of the scene.

5.1 Future work

The feature extractor stage in the pipeline should be investigated further to see how well different feature extractors perform in different indoor environments. In this thesis, only two feature extractor were studied and only one type of indoor scene was used. Therefore, it could be interesting to see how well different feature extractors perform in different scenes, especially how well they extract features from the floor. In this case Harris corner detector performed quite well since there where many corners present on the floor of the scene.

Furthermore, the robustness of the homography estimation could be further studied. In particular, how to avoid estimating homographies that does not relate the plane given by the floor. One solution could be to develop an additional component in the pipeline that checks if all estimated homographies actually maps to the same

plane. If a homography maps the wrong plane, then the corresponding inliers points are removed and RANSAC is run again. This is done until all pairwise homographies map to the same plane. This could improve the map generation if there are objects present on the scene floor and potentially remove the masking of objects in the preprocessing stage. Lastly, it might be worth to investigate how non-invasive modification of the scene floor can improve the homography estimation. For example, adding several tags on the floor where the feature extractor can detect features from.

Bibliography

[1] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521540518.

[2] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 1848829345.

[3] Noah Snavely, Steven M Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM Siggraph 2006 Papers*. 2006, pp. 835–846.

[4] Noah Snavely, Steven M Seitz, and Richard Szeliski. "Modeling the world from internet photo collections". In: *International journal of computer vision* 80.2 (2008), pp. 189–210.

[5] Changchang Wu. "Towards linear-time incremental structure from motion". In: *2013 International Conference on 3D Vision-3DV 2013*. IEEE. 2013, pp. 127–134.

[6] Linguang Zhang, Adam Finkelstein, and Szymon Rusinkiewicz. "High-precision localization using ground texture". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6381–6387.

[7] Matthew Brown and David G Lowe. "Automatic panoramic image stitching using invariant features". In: *International journal of computer vision* 74.1 (2007), pp. 59–73.

[8] B. Triggs et al. "Bundle adjustment - A modern synthesis". In: *ICCV '99 Proceedings of the International Workshop on Vision Algorithms: Theory and Practice* (Jan. 2000), pp. 198–372.

[9] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. "Visual SLAM and structure from motion in dynamic environments: A survey". In: *ACM Computing Surveys (CSUR)* 51.2 (2018), pp. 1–36.

[10] Sameer Agarwal et al. "Building rome in a day". In: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[11] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[12] Johannes L. Schonberger and Jan-Michael Frahm. “Structure-From-Motion Revisited”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[13] H Christopher Longuet-Higgins. “A computer algorithm for reconstructing a scene from two projections”. In: *Nature* 293.5828 (1981), pp. 133–135.

[14] F.Kahl. *Lecture 6: Camera Computation and the Essential Matrix*. 2019.

[15] Yasutaka Furukawa and Jean Ponce. “Accurate, dense, and robust multiview stereopsis”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.8 (2009), pp. 1362–1376.

[16] Andrew J Davison. “Real-time simultaneous localisation and mapping with a single camera”. In: *null*. IEEE. 2003, p. 1403.

[17] Hauke Strasdat, José MM Montiel, and Andrew J Davison. “Visual SLAM: why filter?” In: *Image and Vision Computing* 30.2 (2012), pp. 65–77.

[18] Hyon Lim, Jongwoo Lim, and H Jin Kim. “Real-time 6-DOF monocular visual SLAM in a large-scale environment”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 1532–1539.

[19] Richard Szeliski et al. “Image alignment and stitching: A tutorial”. In: *Foundations and Trends® in Computer Graphics and Vision* 2.1 (2007), pp. 1–104.

[20] Matthew Brown, David G Lowe, et al. “Recognising panoramas.” In: *ICCV*. Vol. 3. 2003, p. 1218.

[21] David L Milgram. “Computer methods for creating photomosaics”. In: *IEEE Transactions on Computers* 100.11 (1975), pp. 1113–1119.

[22] Chung-Ching Lin et al. “Adaptive as-natural-as-possible image stitching”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1155–1163.

[23] Julio Zaragoza et al. “As-projective-as-possible image stitching with moving DLT”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2339–2346.

[24] Zhengyou Zhang and Li-Wei He. “Whiteboard scanning and image enhancement”. In: *Digital Signal Processing* 17.2 (2007), pp. 414–432.

[25] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[26] F.Kahl. *Lecture 1: The Pinhole Camera Model*. 2019.

[27] F.Kahl. *Lecture 3: Camera Calibration, DLT, SVD*. 2019.

[28] Etienne Vincent and Robert Laganiére. “Detecting planar homographies in an image pair”. In: *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat.* IEEE. 2001, pp. 182–187.

[29] Christopher G Harris and Mike Stephens. “A combined corner and edge detector.” In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[30] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[31] Michael Calonder et al. “BRIEF: Computing a local binary descriptor very fast”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2011), pp. 1281–1298.

[32] Jeffrey S Beis and David G Lowe. “Shape indexing using approximate nearest-neighbour search in high-dimensional spaces”. In: *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE. 1997, pp. 1000–1006.

[33] F.Kahl. *Computer Vision, Assignment 2 Calibration and DLT*. 2019.

[34] F.Kahl. *Lecture 9: Local Optimization*. 2019.

[35] Peter J Burt and Edward H Adelson. “A multiresolution spline with application to image mosaics”. In: *ACM Transactions on Graphics (TOG)* 2.4 (1983), pp. 217–236.

[36] Alexander Behrens et al. “A non-linear multi-scale blending algorithm for fluorescence bladder images”. In: *Computer Science-Research and Development* 26.1-2 (2011), pp. 125–134.

[37] Andrea Vedaldi and Brian Fulkerson. “VLFeat: An open and portable library of computer vision algorithms”. In: *Proceedings of the 18th ACM international conference on Multimedia*. 2010, pp. 1469–1472.

[38] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[39] Peter J. Huber. "Robust estimation of a location parameter". In: *Annals of Mathematical Statistics* 35.1 (Mar. 1964), pp. 73–101.

[40] Sameer Agarwal, Keir Mierle, et al. *Ceres Solver*. `http://ceres-solver.org`.

[41] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

A

Appendix: Input images

Figures A.1-A.3 shows all the inputs images used to evaluate the pipeline.

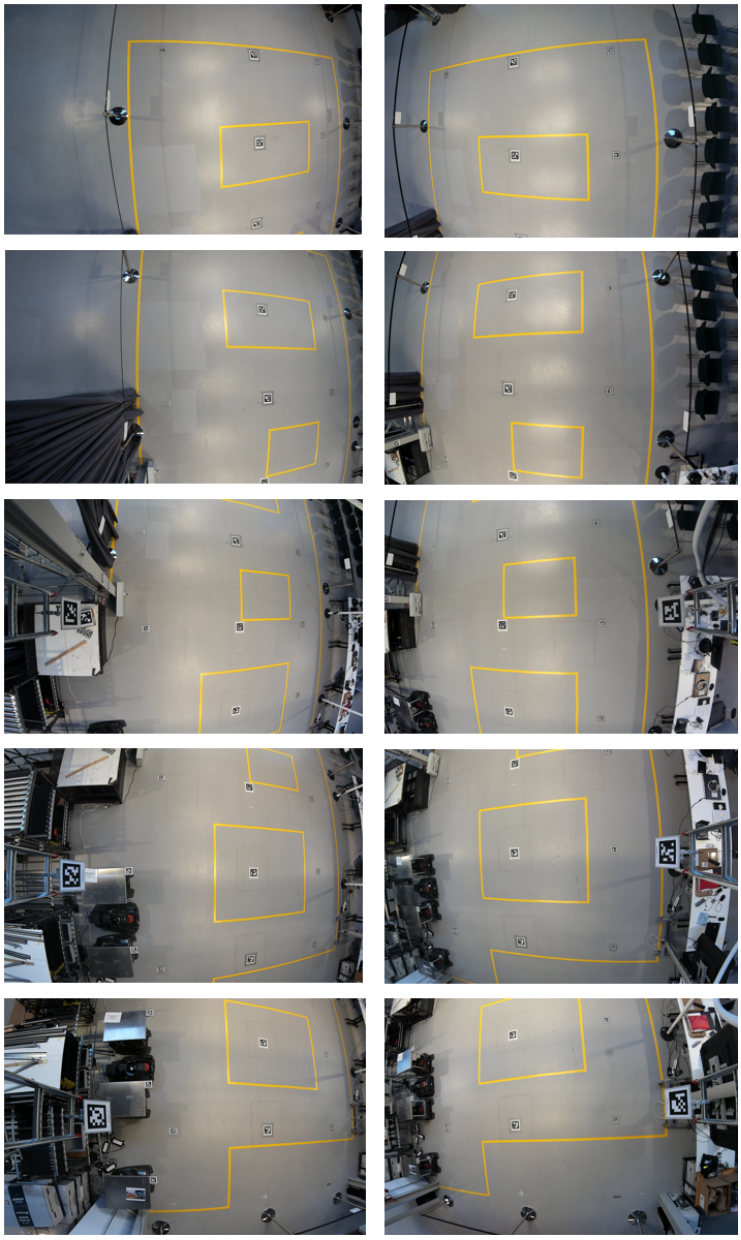


Figure A.1: The figure shows all the input images for a scene without objects.

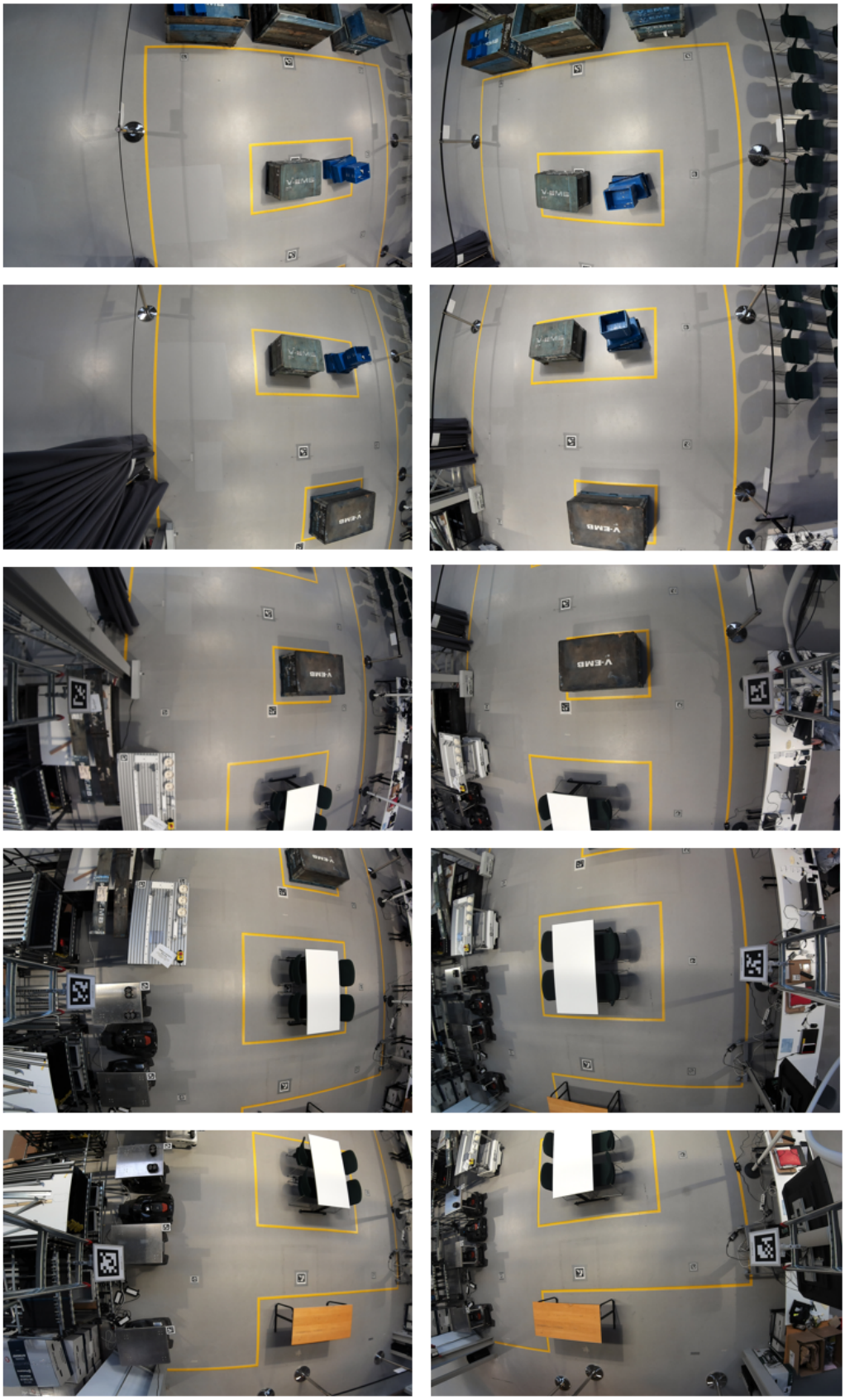


Figure A.2: The figure shows all the input images for a scene with objects.

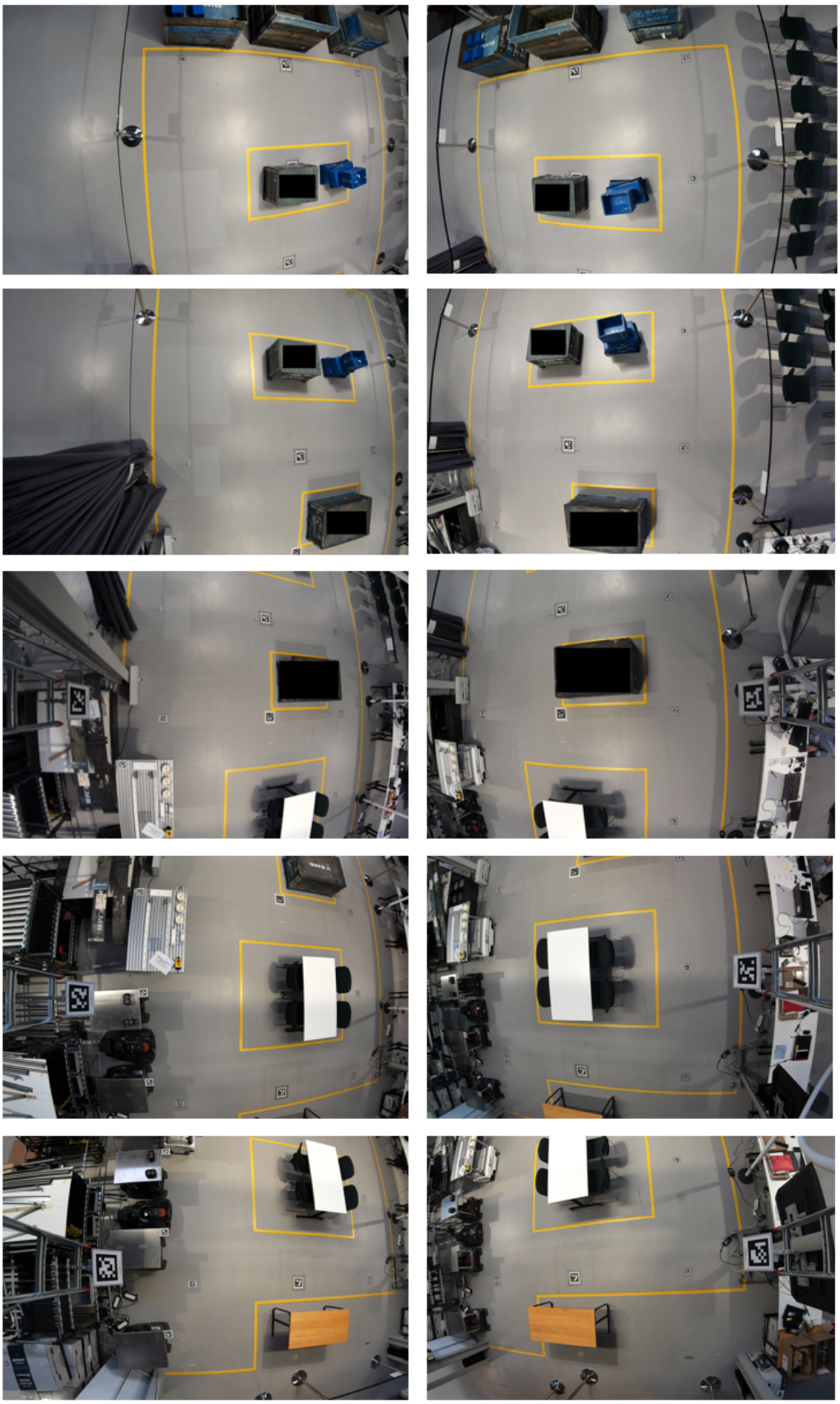


Figure A.3: The figure shows all the input images for a scene with objects were two pallets have been masked.