MSc thesis in Geomatics

Dynamic Seamless Oblique Image Mosaics for Aerial Visualization

Xiaoluo Gong

2025

TU Delft

MSc thesis in Geomatics

Dynamic Seamless Oblique Image Mosaics for Aerial Visualization

Xiaoluo Gong

June 2025

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

Geo-Database Management Centre
Delft University of Technology

Ingenieursbureau Geodelta B.V.

Supervisors:	Dr.ir. Martijn Meijers
	Ir. Edward Verbree
Exterior Supervisor:	Ir. Annemieke Verbraeck
Co-reader:	Dr. Azarakhsh Rafiee

Abstract

High-resolution image mosaicking plays a critical role in geomatics and remote sensing applications, allowing efficient visualization, measurement, and analysis of large-scale environments. Although existing commercial tools provide standard stitching capabilities, they often lack mathematical transparency and real-time customization, limiting their utility in research and professional analysis.

This thesis introduces a systematic approach to dynamic image stitching and visualization within a C# environment. The method uses homography transformations to achieve accurate image alignment while integrating an optimal seam-finding algorithm to improve visual coherence in overlapping regions. An exportable homography matrix supports coordinate traceability, enabling users to perform metric evaluations on stitched images. The implementation focuses on creating a lightweight, interactive stitching prototype capable of processing two to three aerial images with high geometric fidelity and run-time efficiency.

Experimental validation confirms that the system delivers precise stitching results and supports visual exploration for measurement tasks. By combining mathematical clarity, dynamic responsiveness, and user adaptability, this research contributes to a modular and extensible foundation for image mosaicking in the context of geomatics, with practical relevance for aerial inspection, photogrammetry, and spatial data visualization.

Acknowledgements

I would like to acknowledge all those who have supported me in completing this thesis.

First and foremost, I would like to express my sincere gratitude to my company mentor and external supervisor, **Annemieke Verbraeck**, for her invaluable guidance, continuous support, and heartfelt encouragement throughout this thesis. More than a mentor, Annemieke has also been a cheerful and understanding friend whose advice has supported both my academic and personal growth.

I am also grateful to the supervisory team at TU Delft, especially **Dr. Martijn Meijers** and **Edward Verbree**. Martijn's technical expertise and practical support helped me tackle many challenges, while Edward's creative insights consistently pushed my research in new and exciting directions. Also, thanks my co-reader **Dr. Azarakhsh Rafiee** for giving feedback during her busy schedule.

Many thanks to my colleagues at Geodelta for their daily support. Lunchtimes and game parties with them brought joy and balance to my routine. I am especially grateful to the director of Geodelta, **Martin Kodde**, who followed my progress closely and offered thoughtful suggestions that significantly improved my work despite his busy schedule.

Heartfelt thanks to my old friends in China for staying connected across the distance, and to my new friends in the Netherlands, especially **Jiaqi, Zhuoyue, and Marieke** for their kindness, support, and for making me feel at home away from home.

Lastly, I would like to express my deepest gratitude to my family, especially my parents, for their unwavering love, support, and belief in me throughout this journey. Although we have been separated by distance, their constant encouragement and understanding have been a source of strength during the challenging and rewarding moments of this thesis. Their sacrifices and faith in my abilities have played an essential role in helping me reach this milestone, and for that, I am endlessly thankful.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

Glossary

blending a step in image stitching, mix the value of pixels in different images to get an integrate stitching result.. 19, 20, 53

feature point the point which can be easily detected from an image and are usually invariant to scale or time. For example, the corners of buildings are usually good feature points.. 23, 39, 55

homography matrix a 3×3 invertible matrix that describes a projective transformation between two planes in projective geometry.. 12, 14, 16, 17, 22, 27, 30–35, 40, 41, 44–46, 48, 57, 58, 61, 62, 65

morphing the process of smoothly transition between two images, especially between the original image and warped image.. 4, 5, 16, 17, 20, 27

stitching the process of combining two or more images into a single and integrate image.. xi, 2–4, 8, 10, 12, 16–19, 22, 23, 27, 28, 31, 32, 35, 39, 45, 48, 50, 53, 55, 56, 58, 61–63

tie-point the feature point pair that represent the same object on two or more images.. 3, 4, 10, 11, 17, 29, 30, 32, 35, 39, 41, 42, 53, 61, 62

warping digitally manipulate an image to distort its shape to a certain target.. 2–5, 8, 11, 12, 16, 17, 20, 31, 35, 37, 42, 48, 53, 57, 58

1. Introduction

1.1. Background and Motivation

Oblique aerial images are images taken by survey aircraft with an inclined camera angle, offering multiple perspectives of the areas of interest (Verykokou and Ioannidis [2024]). Oblique aerial images provide more intuitive and stereoscopic views of the structures and landscapes than traditional nadir images, revealing details that are often obscured from the vertical view. In addition, compared to panoramas such as Google street view, which can also offer a 3D-like visualization experience and is usually collected by cars or handheld devices, oblique aerial images can cover a larger range with better time efficiency and lower financial cost. Meanwhile, with a larger view field than the panorama, oblique aerial images have less chance of causing dizziness after viewing (Armstrong et al. [2016]). As moving through panoramas offers users the view of walking or driving in the street, the continuous exploration of oblique images offers users the view of flying across the city. With these advantages, oblique aerial images are particularly valuable in fields such as urban planning, heritage protection, and disaster management (Zhang et al. [2023]; Höhle [2013]; Tang et al. [2024]).

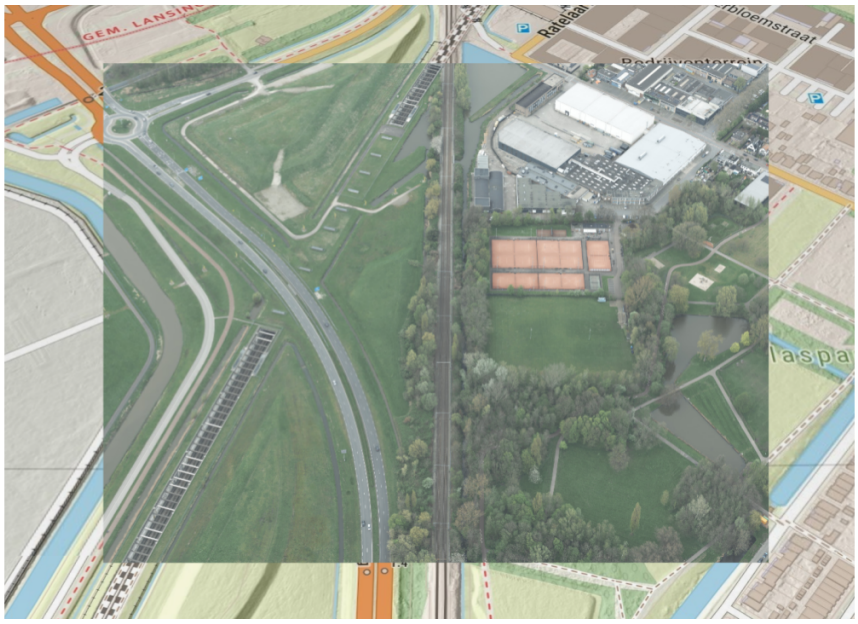

Figure 1.1.: An oblique aerial image accurately superimposed on the base map, source: Omnibase;

Geodelta has developed a comprehensive platform, Omnibase, which offers combined visualization and easy measurements of point cloud, panorama, nadir images, oblique images, and 3D meshes (Geodelta [2024]). Omnibase presents the oblique images by assigning them separately to the accurate locations and range on the map (see Figure 1.1). To give users a smoother and continuous experience when navigating through consecutive oblique aerial images, seamless Stitching of the images is essential to improve image visualization as well as continuous measurement experience.

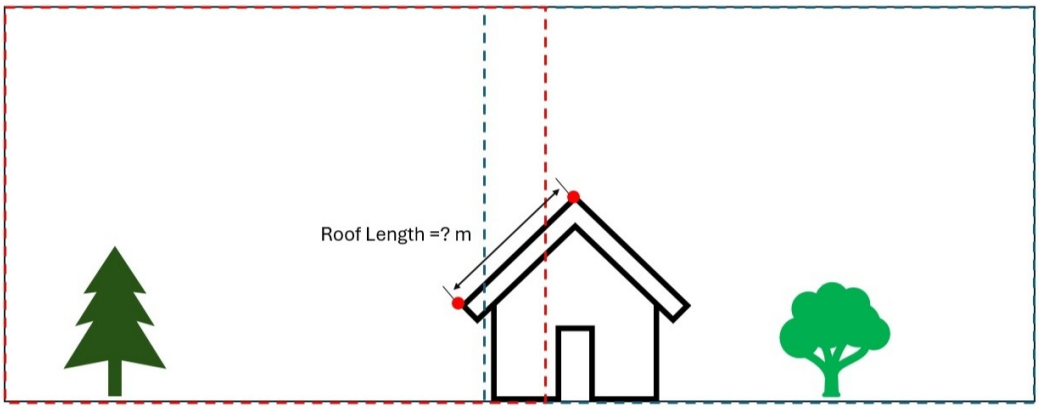

Figure 1.2.: Stitched images enable measuring objects that locate across the boundaries of the image frames

1.2. Research Scope

For general applications, image Stitching is the process that can combine different images by their overlapped areas to obtain an integrated image with a wide view (Wang and Yang [2020]). Generally, there are two main categories of image Stitching, mosaic Stitching, and panoramic Stitching (Fu et al. [2023]). Table 1.1 shows the differences between those two types of image Stitching in various aspects. Taking into account those features of the two Stitching categories, this thesis project will focus on mosaic Stitching to realize the Stitching of aerial images.

In the case of aerial oblique images, each individual image already covers a large area and has a significant file size. Stitching all oblique images in a region into a single integrated map is impractical due to both computational and visualization challenges. Computationally, such a map would be difficult to process because of its massive size. From a visualization perspective, Warping all images to fit a single reference frame can result in significant distortions.

This M.Sc. thesis focuses on a different approach: Stitching aerial images to provide smooth and continuous transitions when switching between adjacent oblique images. This dynamic process ensures seamless navigation between images rather than creating a static large-scale integrated map. At least two images with the same view direction must be stitched together to ensure a smooth transition between images. More images in the same column or row may also be stitched together to create a wider, belt-like view. Users can stop at any

Aspect	Mosaic Stitching	Panorama Stitching
Purpose	Creating a large-scale plane images.	Generating a wide-angle view of a scene (panoramic images)
Viewpoint	Multiple or moving cameras (e.g. drone, aircraft, satellite) with fixed view angle.	Single fixed viewpoint, camera rotates around its axis.
Geometric Correction	Requires significant geometric corrections for perspective, scale, rotation, and alignment due to varying viewpoints.	Primarily focuses on aligning images with minimal geometric correction, mainly dealing with rotation and lens distortion.
Image Overlap	Moderate to large overlap between adjacent images.	Large overlap is required.
Scale and Application	Usually larger-scale (e.g. satellite mapping, aerial images) and approximately plane scene.	Usually smaller-scale, artistic or photographic applications (e.g. indoor photos, street views).

Table 1.1.: Comparison between Mosaic Stitching and Panorama Stitching

intermediate stage of their moving actions to do the measurement, especially by measuring those objects that are divided into two parts by different image frames. Pre-calculations of some parameters like the transformation matrix between two images will be done to achieve a fast real-time performance.

Stitching aerial oblique images is a challenge in several aspects. First, oblique images have perspective distortion depending on the varying distance from the camera to the ground objects (see Figure 1.3). Second, camera motions will cause parallax and alignment errors (Azzari et al. [2008]). Third, varying elevation in the terrain can cause visible artifacts in the stitched images if not handled properly (Steedly et al. [2010]).

To solve these problems, image processing technologies including Tie-point matching and Warping are needed. This research also incorporates camera parameters and camera motion model in photogrammetry. Optimal seam algorithm (Ai and Kan [2020]) is also applied to preserve the building structure as much as possible. The process only applies perspective distortion to the calibrated aerial images ,thus the final results will preserve the feature of straight lines, which can help to apply precise measurement on the images. With a combination of those technologies, the main goal of this research is to create seamless mosaics from oblique aerial images, allowing for a continuous and immersive aerial view experience, as well as minimum errors in measurements through the oblique aerial images.

1.3. Research Questions

The research questions addressed in this M.Sc. thesis, along with the expected outcomes, are focused on processing existing oblique aerial image data of city-scale in Omnibase to achieve seamless Stitching and smooth transitions. The main research question is as follows:

How can seamless oblique image mosaics be created dynamically from aerial photographs to enhance continuous visualization and minimize measurement errors?

Figure 1.3.: Oblique aerial images of the same area in Utrecht with different perspective distortion, source: Omnibase

The subquestions under this main questions include:

1. To what extent can the Stitching achieve a continuous transition between oblique aerial images? And how is continuity measured?
2. How does the area of the intersection or the number of detected Tie-points between two images influence the Stitching quality?
3. How does the seam in the stitched images influence the visualization effect?
4. What is the robust method for oblique aerial image Stitching when the intersection area is small?
5. How does the Stitching process impact the usability of images for measurement tasks?
6. How is Stitching quality assessed from both visualization and measurement aspects?

1.4. Thesis Outline

Chapter 1 gives an overview of this thesis, including motivation, research background, research scope, and research questions.

Chapter 2 outlines the foundational concepts required for the study, including the features of oblique aerial images and the techniques involved in image Stitching, such as key point detection and image Morphing.

Chapter 3 presents a review of the relevant literature, including existing visualization platforms for oblique aerial images, measurement techniques, and panoramas.

Chapter 4 describes the methodology employed in the study, detailing preprocessing, image Warping, and measurement transformation techniques.

Chapter 5 covers the implementation details, including query operations, lens distortion calibration, image Warping, seam finding, dynamic Morphing processes, and how to apply measurements on stitched images.

Chapter 6 presents the experimental results and the methodology evaluation, including an overview of the stitched images, an evaluation of the geometric accuracy, and the result of the real-time stitch performance test.

Chapter 7 summarizes the study's contributions and answers the research questions. The chapter concludes with insights and suggestions for future work to improve the methodologies explored in this thesis.

2. Theories and Concepts

2.1. Features of Oblique Aerial Images

Oblique aerial imagery has gained attention in the field of photogrammetry and remote sensing due to its ability to capture detailed perspectives of buildings, terrain, and other surface features from an inclined angle. Images with an optical axis that deviates from the vertical more than five degrees are usually considered oblique images (Verykokou and Ioannidis [2024]; Höhle [2008]). Contemporary oblique systems typically employ multicamera setups to capture images from several angles. The most frequently used setup is called Maltese-cross configuration, which consists of a vertical camera facing downward and four oblique cameras angled towards the cardinal directions. This configuration ensures symmetrical coverage of ground objects and maximizes the likelihood that any vertical surface, such as the sides of a building, will be captured from at least one camera. In this thesis project, the oblique image data are also collected from Maltese-cross cameras (see Figure 2.1).

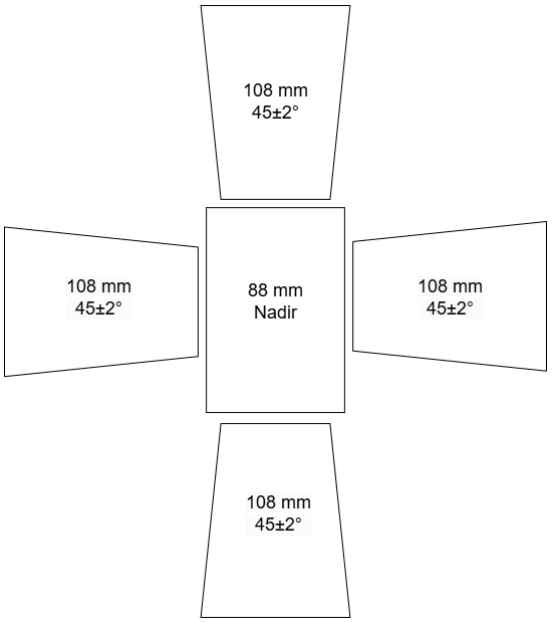

Figure 2.1.: Focal length, camera inclined angle, and camera frame in this thesis project.

Despite the advantage of façade visibility, oblique aerial images also have more perspective distortion than nadir images caused by the varying ground sample distance. The lower side of the image, which is closer to the camera, appears to be larger and more stretched than the upper side. When the camera moves, the upper side of the previous image will be

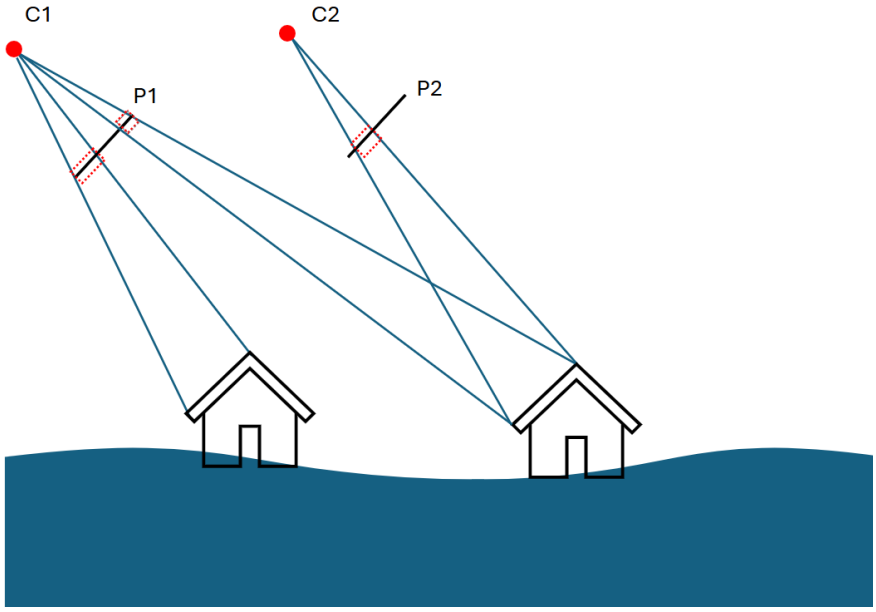

Figure 2.2.: Perspective distortion in oblique aerial images

placed in the lower side in the next. Since the position of the camera center which is also the perspective center has changed, a simple 2D-motion perspective model cannot fully describe this change. Thus, a more complicated motion model should be employed to stimulate the change of camera centers and the perspective relationships between different images to the same ground objects. Figure 2.2 illustrates the perspective distortion. Two roofs of the same size can look different on the same camera, and the same roof can also have different sizes on different cameras.

2.2. Image Stitching Technique

Multiple image Stitching techniques are applied to different scenarios, including merging images taken by daily mobile devices (Laraqui et al. [2017]), generating panoramic images (Brown and Lowe [2007]), Stitching large-scale nadir aerial images (Pham et al. [2021]), etc. In addition to the specific techniques used in different scenarios, there are some common basic steps in image Stitching, including detecting key points, matching the corresponding points, and image Warping.

2.2.1. Key Point Detection and Matching

Distinctive invariant features are considered the key factors that can be used to decide the correspondence between different images. Many algorithms have attempted to detect and match key points that are invariant in scale and rotation. Some widely used algorithms are presented and discussed in this section.

Key Point Detection

- **Scale Invarint Feature Transform (SIFT) by Lowe [2004]:** First, the Gaussian filter is applied to the input image to find the scale space of the image. The local maxima and minima are calculated based on the Difference of Gaussians (DoG) and are considered as the potential key points that are invariant to the scale. After selecting the potential key points. Finally, a localization algorithm refines the key point selection by fitting them more precisely to nearby data. Every remaining high-contrast key point is assigned one orientation based on local image gradient directions. Finally, each output key point would have a highly distinctive descriptor. The descriptors are used to match the corresponding key points in different images.
- **Speed-up robust features (SURF) by Bay et al. [2008]:** Partly inspired by SIFT, SURF has steps similar to SIFT, and some adjustments are applied in each step of feature detection. First, it uses box Gaussian filters to create the scale space and a Hessian matrix as a measure of local change around a single point. The point will be chosen when its Hessian determinant is maximal. To make sure the descriptor of the key point is rotation invariant, SURF calculates the Haar wavelet responses in the x and y directions within a circular neighborhood around the point of interest. SURF will construct a square region around the detected key point and divide it into 4×4 subregions. Then, the Haar wavelet responses are summed up over each subregion and form a feature vector as descriptor. According to the author of SURF, the SURF detector has a faster speed, which is the main improvement and makes it suitable for real-time computation.
- **Oriented FAST and Rotated BRIEF (ORB) by Rublee et al. [2011]:** This method is built on the FAST detector (Rosten and Drummond [2006]) and the BRIEF descriptor (Calonder et al. [2010]). The original Features from Accelerated Segment Test (FAST) detector used machine learning to find key points, but did not include the orientation component. The ORB measured the orientation by intensity centroid and added this attribute to the key points detected by FAST. Binary Robust Independent elementary features (BRIEF), as a binary string key-point descriptor, is computationally efficient but performs badly with rotation. The ORB steered the BRIEF according to the orientation of the key points, thus improving the BRIEF performance in rotated images. ORB is much faster than SURF and SIFT, but it still has problems with scale invariance.
- **DIScrete Keypoints (DISK) by Tyszkiewicz et al. [2020]:** DISK used reinforcement learning and achieved end-to-end feature matching for a large amount of points. It first used a U-Net (Ronneberger et al. [2015]) based architecture to extract a detection heatmap and then divided this heatmap into grid cells. The key points are sampled within each cell using softmax normalization and a sigmoid-based quality filter. Using reinforcement learning, DISK is trainable and flexible for specific tasks, compared to constructing hand-crafted features such as SIFT and SURF.

Detected Feature Matching

After detecting the key points, matching them with the corresponding images is the next step. The descriptors generated when detecting the features are essential for the matching. For example, a simple use case is to calculate the pairwise distance of all descriptors and find the nearest neighbor as the best match. However, in practice, the real-world images

are very noisy, leading to false matches if only the distance of the descriptors is used as a matching reference. Furthermore, comparing all descriptor distances from all key points has a computational time complexity of $O(n^2)$ with a quadratic in the expected number of features. Thus, some more sophisticated techniques are used for feature matching.

- **Random sample consensus (RANSAC):** RANSAC is an iterative algorithm used to estimate parameters of a mathematical model from a dataset that may contain significant outliers (Fischler and Bolles [1981]) and is widely applied in computer vision (Stewart [1999]). It starts by entering key points using a similarity measure, such as the distance for the SIFT descriptors. Then RANSAC will randomly select a minimal subset of correspondences and compute a linear estimate that has the physical meaning of the camera motion. RANSAC will compute the residuals between the real points locations and the estimated locations and will consider those with residuals smaller than the predefined threshold as inliers. The random selection, model fitting, and inlier counting process will repeat until the estimate gets the maximum number of inliers. RANSAC is efficient when processing noisy data and can produce a high-accuracy alighment.

- **Transformer used in feature matching:** The transformer, a well-known deep learning architecture, has been widely applied to feature matching, due to its self and cross attention mechanisms to consider the relationship between all input key points (Carion et al. [2020]). Two of the most famous models for feature matching are SuperGlue (Sarlin et al. [2020]) and LightGlue (Lindenberger et al. [2023]). The transformer as the backbone of these models consists of self-attention layers to aggregate information from other features within the same image, as well as cross-attention layers to aggregate the information from features in the other image. However, built upon SuperGlue, LightGlue also has some differences and improvements compared to SuperGlue. First, LightGlue uses relative positional encoding, which can better capture long-range dependencies. For the scoring system, SuperGlue uses the Sinkhorn algorithm, an iterative method for optimal transport, to compute the correspondence points between two points. In LightGlue, a simpler matchability and similarity scoring system is designed to compute a pairwise score matrix between the points of both images. Additionally, LightGlue also introduces a pruning mechanism. At the end of each layer, a confidence classifier determines if the matching predictions are reliable. If it is confident, the program will stop early to save computation time. Points that are classified as unmatchable are pruned early to focus computation on relevant features. In the experiments by the authors of LightGlue, DISK + LightGlue is proved to be the combination of feature detector and matcher with the highest accuracy. In their experiments, LightGlue shrinks 35% of the running time compared to SuperGlue.

Geodelta has already built the database for aerial images of Utrecht with matched tie points by applying a deep neural network, LightGlue (Lindenberger et al. [2023]). LightGlue matches local sparse features with high efficiency and robustness in accuracy. LightGlue performs best for adjacent images with the same view direction although it can also find a few corresponding points between those images that have little intersection areas or with different view directions. The thesis project will focus on the adjacent images with the same view direction for image Stitching. The two images on the left of figure 2.3 shows the overview of the matching result, there are 588 pairs of Tie-points in total. The two images on the right of figure 2.3 show some details of the matching results of the two images on the right which are mostly correct but still have some errors. In practice, the key points detected in more than two images will be selected as reliable tie points. Most of the matching results

for adjacent image pairs have numbers of Tie-points between 200 and 600, which is sufficient for building the warping transformation.

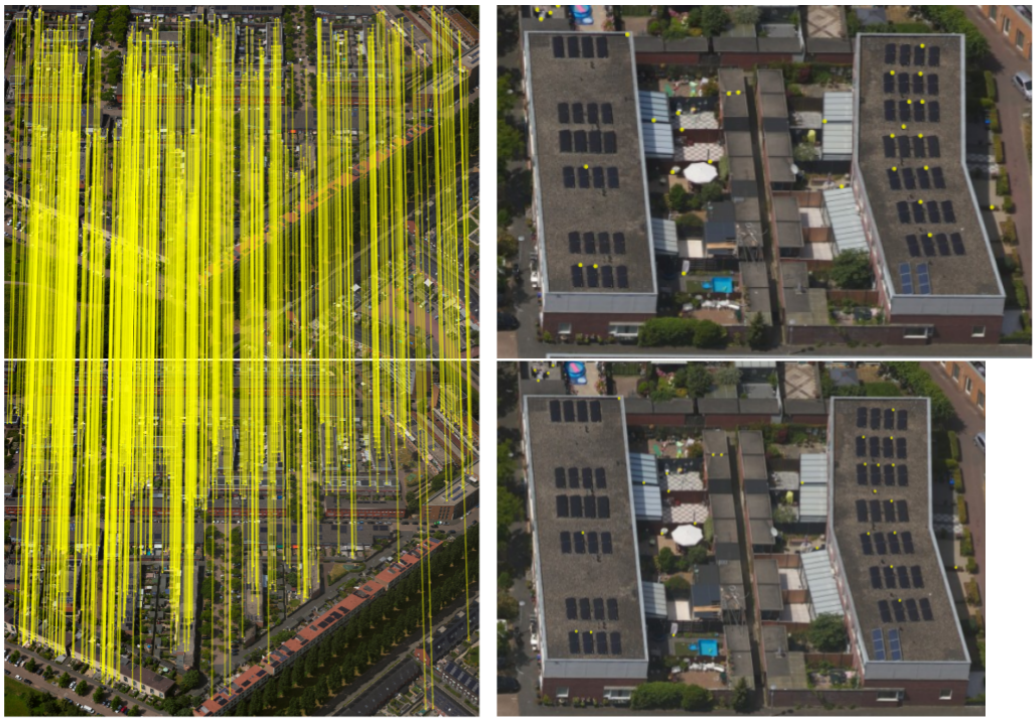

Figure 2.3.: Tie-points matching results for two adjacent oblique aerial images with the same view direction, source: Phoxy

2.2.2. Image Warping

Image Warping is a technique that is used to distort images. The goal of Warping in this thesis is to apply geometric transformation to map a source image onto a target coordinate system by spatial deformation. According to Wolberg [1990], digital Warping consists of two primary steps: coordinate transformation and intensity interpolation. Glasbey and Mardia [1998] further categorize Warping methods into parametric and non-parametric approaches, emphasizing their applications in image registration, motion correction, and shape analysis. Parametric Warping uses mathematical functions with a finite set of parameters to define the transformation, while the non-parametric transformation uses data-driven methods without an explicit parameter equation. Parametric transformations are usually applied globally and ensure the global smoothness, while nonparametric transformations are more flexible for local variation or non-rigid motion. From the computational cost aspects, parametric Warping is more efficient due to its reliance on mathematical equations. Considering the features of parametric and nonparametric Warping, parametric Warping fits better in processing large-scale aerial images used for measuring.

Warping based on 2D transformation

The 2D transformation model shows how a flat plane in 3D space is projected into two different 2D images. Global Warping based on 2D motions can be achieved through global linear transformations of the image. These transformations are represented by a transformation matrix, which can have Degrees of Freedom (DoF) ranging from 2 to 8. This corresponds to the minimum need for only 1 to 4 pairs of corresponding points to define the transformation. The DoF determines the type of transformation, allowing various operations such as translation, scaling, rotation, shearing, or a combination of these. Figure 2.4 illustrates the basic types of 2D transformations, including rigid transformations (translation and rotation), similarity transformations (scaling and rotation), affine transformations, and projective transformations. Each type provides progressively more flexibility in Warping the image, with projective transformations allowing the highest degree of non-linear adjustment. These global transformations are particularly useful for scenarios where the deformation is uniform across the entire image.

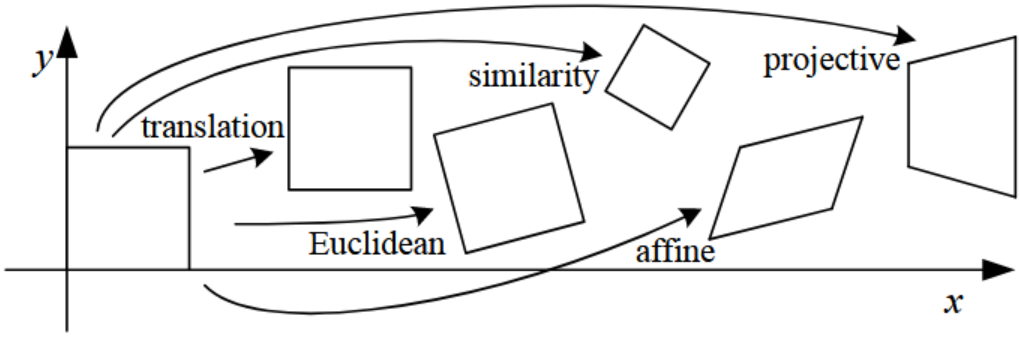

Figure 2.4.: Basic types of 2D transformations (Szeliski et al. [2007])

One of the most important 2D transformations in image Stitching is the projective transformation. Projective transformation, also known as perspective transformation or homography, is a type of 2D transformation that maintains perspective relationships between points. It illustrates the effect that the near objects are big and far objects are small. The straight lines remain, but the angles, distance and parallel relationship are not preserved. The perspective transformation can map any quadrilateral to another quadrilateral, making it very flexible for Warping and alignment. Thus, perspective transformation is widely used in image rectification, Stitching, and projection. A projective transformation can be represented mathematically using a **3×3 Homography matrix** 2.1:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2.1}$$

The matrix H is defined up to scale (i.e., the matrix is normalized such that $h_{33} = 1$ or another value), there are **8 independent unknowns** to solve for: $h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}$.

The relationship between a point (x, y) in the source image and the transformed point (x', y') in the target image is given by 2.2:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.2}$$

Where:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.3}$$

Cross-multiplying to eliminate the denominators gives two linear equations for each correspondence:

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} - h_{31}x'x - h_{32}x'y - h_{33}x' = 0, \\ h_{21}x + h_{22}y + h_{23} - h_{31}y'x - h_{32}y'y - h_{33}y' = 0. \end{aligned} \tag{2.4}$$

Thus, each pair of points contributes 2 equations to the linear system, and we can get a $2n \times 9$ matrix A (2.5):

$$A = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x'_1x_1 & x'_1y_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & y'_1x_1 & y'_1y_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x'_2x_2 & x'_2y_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & y'_2x_2 & y'_2y_2 & y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{2.5}$$

For the unknown elements in H (2.1), we can write them into a vector $\mathbf{h}$ (2.6) and the equation to solve can be written as 2.7:

$$\mathbf{h} = \begin{bmatrix} h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33} \end{bmatrix}^T \tag{2.6}$$

$$A\mathbf{h} = 0 \tag{2.7}$$

Since there are 8 unknowns in the homogeneous equation 2.7, at least 4 pairs of non-collinearity points are needed to solve them. The system will be solved with the Singular Value Decomposition (SVD).

Warping based on 3D transformation

In real world applications, the camera can undergo translation as well as rotations. Also, scenes like terrains and buildings are usually not planar. 3D transformation has a hierarchy similar to that of the 2D transformation and can be denoted using 4×4 transformation matrices which describe rigid body motion and affine transformations (Szeliski et al. [2007]).

In photogrammetry, the intrinsic matrix K (2.8) describes the internal parameters of the camera, including the focal length (f_x, f_y), the skew distortion (s) and the center of the image (c_x, c_y).

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

In the 3D perspective model, a 4×4 projection matrix P (2.9) is constructed based on the intrinsic matrix K (2.8). This matrix maps the homogeneous 4-vector $p = (X, Y, Z, 1)$ to a special kind of homogeneous screen vector $\tilde{q} = (x, y, 1, d)$. Here, d represents the inverse depth $(\frac{1}{Z})$, also known as the disparity in stereo vision.

$$\tilde{q} \sim \begin{bmatrix} K & 0 \\ 0^{\mathrm{T}} & 1 \end{bmatrix} p = Pp, \tag{2.9}$$

Despite the intrinsic parameters, the motion of the camera also influences the position of the projected point in the image plane. Thus, the 3D rigid body motion E (2.10) is also a part of the final Homography matrix.

$$\mathbf{q} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{p} = \mathbf{E}\mathbf{p}, \quad R = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.10}$$

So, considering both the perspective projection P and the camera motion, the relation between the camera coordinates to the 3D coordinates can be written as 2.11

$$\tilde{q}_0 \sim P_0 E_0 p. \tag{2.11}$$

In contrast, if we know the depth of a pixel in an image, we can map it back to the 3D coordinate p using 2.12

$$p \sim E_0^{-1} P_0^{-1} \tilde{q}_0 \tag{2.12}$$

Thus, by projecting a pixel back to the 3D world and projecting it to another image plane, we can discover the relationship between the pixels in two different images. 2.13 describes this projection process.

$$\tilde{q}_1 \sim P_1 E_1 p = P_1 E_1 E_0^{-1} P_0^{-1} \tilde{q}_0 = M_{10} \tilde{q}_0. \tag{2.13}$$

M_{10} can be explicitly written with intrinsic and extrinsic matrix as:

$$M_{10} = \begin{bmatrix} K_1 R_1 R_0^{-1} K_0^{-1} & K_1 \left(t_1 - R_1 R_0^{-1} t_0 \right) \\ 0^T & 1 \end{bmatrix} \tag{2.14}$$

If the camera only has rotation without transformation($t_1 = 0$), the matrix can be simplified as equation 2.15. Figure 2.5 illustrates this scenario.

$$\tilde{M_{10}} = K_1 R_1 R_0^{-1} K^{-1} \tag{2.15}$$

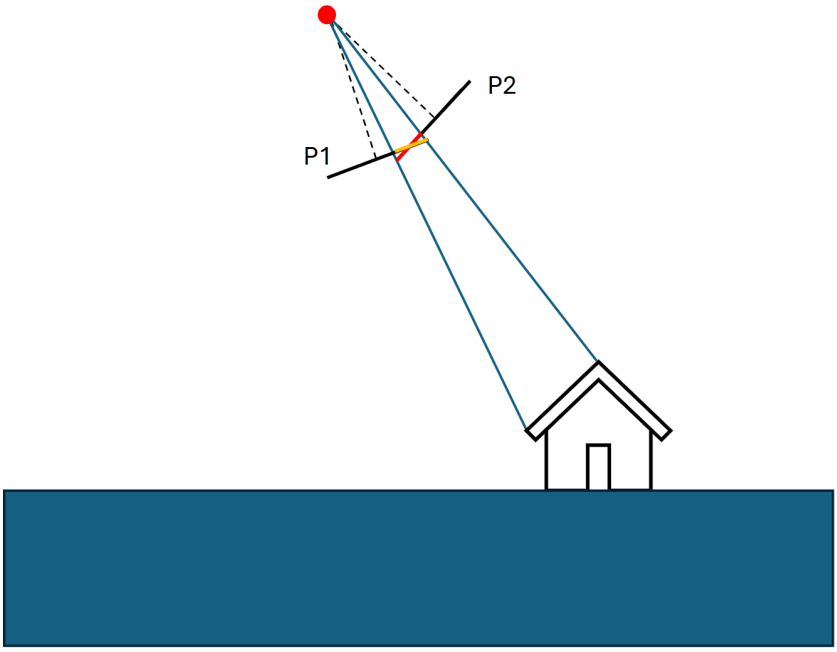

Figure 2.5.: The same object on different images taken by a fixed camera with rotation

Back to the universal scenario, M_{10} can be explicitly written as:

$$M_{10} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

In 2.16, the elements m_{13}, m_{23}, m_{33} represent how changes in depth ($Z\_direction$) of a point in image 0 affect its coordinates in the plane of image 1. However, it is hard to get the height of the terrain objects for every pixel in a high-resolution image. The matrix is usually simplified by assuming a planar scene for the image range in practical (Szeliski et al. [2007]). The depth-dependent column (m_{13}, m_{23}, m_{33}) is ignored as the last row of matrix M_{10} which is used as a mathematical placeholder for homogeneous coordinate transformations in 3D space. This matrix without depth also simulates a special situation in which the sights the camera is observing are almost flat (see Figure 2.6).

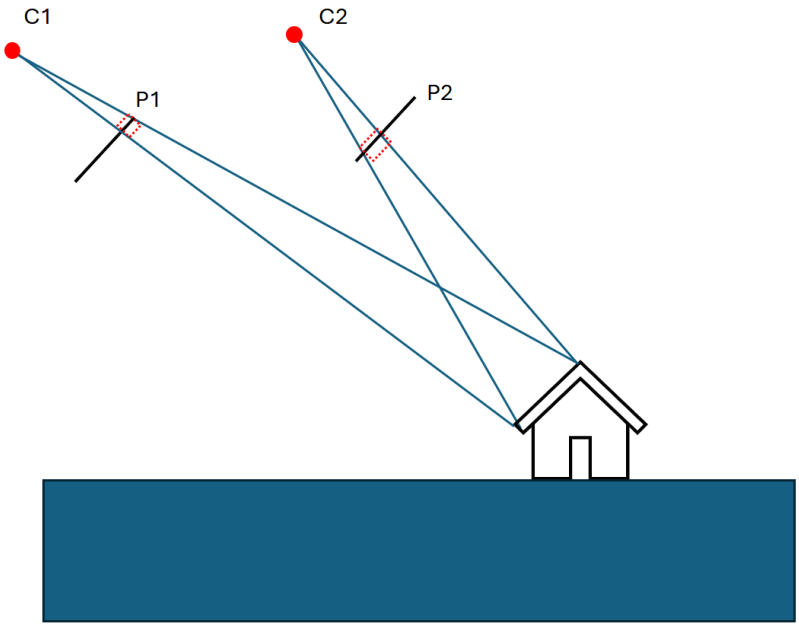

Figure 2.6.: The same object on different images taken by moved camera in a flat terrain scenario

The reduced mapping equation between image 0 and image 1 is a 3×3 Homography matrix which can be written as:

$$H_{10} = \begin{bmatrix} m_{11} & m_{12} & m_{14} \\ m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \end{bmatrix} \tag{2.17}$$

Mathematically, the matrix H(2.1) and H_{10}(2.17) have similar forms and exactly the same 8 degree of freedom. However, they are composited in totally different ways. The H is generated by solving the linear system 2.7 which is constructed by the corresponding pair of points. The H_{10} is derived from known camera parameters. Warping quality by using H highly depend on the matching point data quality including their geometric accuracy, matching accuracy, and distribution on the image. For Warping with H_{10}, the results rely mainly on the quality of camera parameters, which is more stable than using H if those parameters are precisely calibrated.

Other Warping Methods

Despite the linear methods above which use a single Homography matrix, there are various more complex Warping methods for different purposes such as image Stitching or image Morphing. The following mainly introduces the Warping method applied for image Stitching.

- **Dual-Homography warping:** Dual-Homography is used for panoramic Stitching when the camera undergoes a slight transformation instead of rotating precisely around its optical center(Gao et al. [2011]). This method uses K-means clustering to divide the Tie-points into two groups that represent the ground and the distant plane. Then, it estimate two separate homography for each plane and generates a per-pixel weighting map to blend these two homography into a nonlinear warp for seamless alignment.The approach is particularly suitable for panoramic scenes commonly captured at tourist locations, where there are clear distinctions between a distant background and a closer ground plane. However, Dual-Homography will also cause an obvious folding trace on the warped image, and it will fail when the image contains large structures like high buildings that cannot be divided to any of the ground or distant plane.

- **Local warping:** local Warping is commonly used in image Morphing especially when the object on the image has complex and non-linear boundaries like human face (Efros [2016]). In image Stitching, it has also become popular to deal with the parallax. The basic idea is to divide the image into meshes such as grids or triangle nets. Then, it calculates the local projective matrix for each mesh and warps the image mesh by mesh. One of the most used techniques is called As-Projective-As-Possible (APAP) (Zaragoza et al. [2013]). The key technique of APAP is Moving Direct linear transformation (DLT), which computes a local homogeneous matrix for every pixel. The resulting APAP warp smoothly varies across the image, allowing localized adjustments and flexibility. APAP has a relatively high time complexity and will take tens of seconds for an image with 1500×2000 pixels with some C accelerations. Also, to obtain a perfect Stitching result, fine-tuning of the parameters in APAP and other local Warping methods is necessary.

- **Other non-linear warping:** Adaptive As-Natural-As-Possible (AANAP) (Lin et al. [2015]) Stitching addresses distortions similarly to APAP by employing spatially varying local homographies combined with global similarity transformations. AANAP emphasizes a smoother transition between local and global transformations to minimize perspective distortion. Similarly to APAP, AANAP also have relatively high computational complexity. Quasi-homography (Li et al. [2017]) warps propose a balance between projective and perspective distortions by slightly adjusting a global homography warp, using linearized scaling factors. Quasi-homography can obviously improve the visual effect when the overlap rate between images is small and remain stable for multiple image Stitching. This approach retains simplicity, avoiding APAP's high computational complexity and parameter sensitivity while still effectively reducing visible distortions in image Stitching.

2.2.3. Image Composition

Optimal Seams

Optimal seam finding is a critical stage in image Stitching, which significantly impacts the quality and seamlessness of the final mosaics. As illustrated in Figures 2.5 and 2.6, there are only few cases in which the 3×3 Homography matrix can fit the transformation between images very well. However, the real-world scenarios can be so complex that a single homography could not cover, especially the rugged terrain or abrupt high buildings. Figure 2.7 illustrates the image Stitching with elevation changes, which is common in the real world. The defects of the Homography matrix will lead to duplicated or broken objects in

the stitched images. To reduce the influence of elevation change by avoiding going to certain objects such as the rooftop, optimal seam finding is necessary for single homography Stitching with camera movement. Despite its advantage in reducing the error caused by elevation changes, it preserves the structure of buildings or even microstructure such as solar panels or some textures.

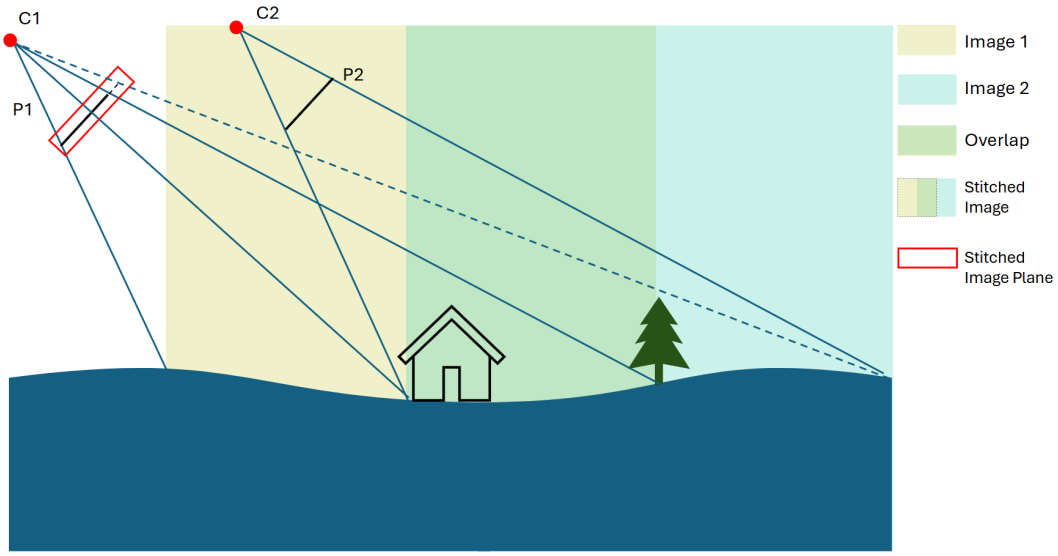

Figure 2.7.: The image Stitching process considering the elevation changes

Various algorithms have been proposed and developed over the years, aiming to efficiently identify seams that minimize visible artifacts. Although we obtain the optimal seam by different methods, the expected outcomes have the same feature, which is to allow the seam to lay as much as possible on the uniform textures. Figure 2.8 illustrates the concept of optimal seams by avoiding seams on the building.

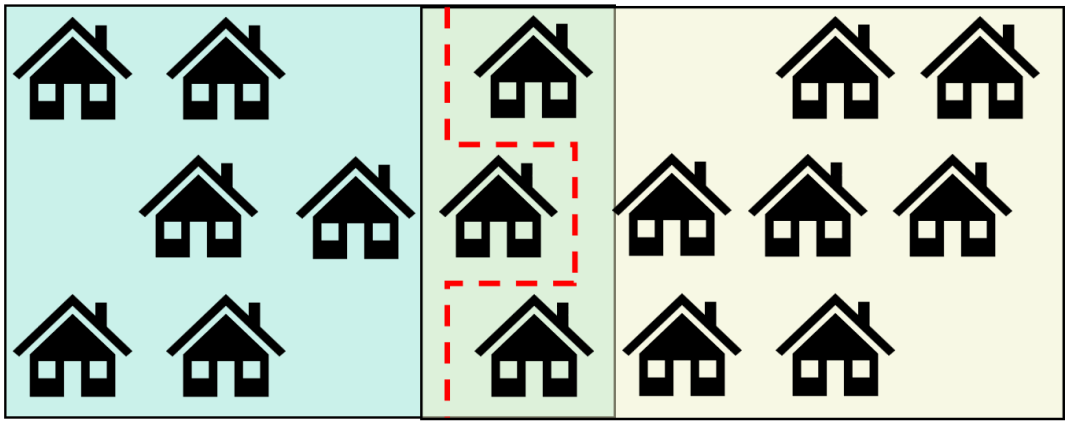

Figure 2.8.: An optimal seam (the red dashed line) across the overlapping area

Graph cut is a classic image segmentation algorithm that models a problem as a graph, where the nodes represent pixels, and the edges represent the costs of labeling (Greig et al. [1989]). The goal is to divide the graph into two or more segments by finding the cut with the minimal cost, which corresponds to the optimal solution. It divides the graph by finding the minimal-cost cut, yielding an optimal solution. In image Stitching, the overlap region is treated as a labeling problem, assigning each pixel to a source image. A weighted graph is constructed, where the edge weights reflect the pixel dissimilarity. The Min-Cut/Max-Flow algorithm (Goldberg & Tarjan [1988]) optimizes the cut, ensuring that seams minimize visual artifacts. The global optimality makes graph-cut also The complexity of the algorithm ranges from $O(VE^2)$ (Dinic [1970]) to $O(VE)$(Greig et al. [1989]), where V refers to the number of nodes (pixels of the image) and E refers to the number of edges (connections between neighboring pixels). For high-resolution images, the complexity of the graph-cut algorithm could lead to a long operation time.

Unlike graph-cut methods, which solve a global optimization problem, dynamic programming simplifies seam finding by decomposing it into subproblems through sequential computations. Dynamic programming considers seams throughout the overlap area along a single primary dimension, either horizontally or vertically, making it particularly suitable for linear image Stitching (Zeng et al. [2014]). By transforming the seam finding problem to a shortest path finding problem, the dynamic programming algorithm uses an energy map as input and gives the results of the path with the lowest accumulating energy. As dynamic programming typically works on the energy map of 2-dimensional image, which is usually a gray-scale image, the complexity is $O(W \times H)$. The W and H are the width and height of the overlap image.

Image Blending and Interpolation

Blending is the process by which the pixel values are mixed between two or more images. Linear Blending, also known as cross-dissolve, is a simple and widely applied method in video editing. It generates the new image by adjusting the opacity of one image over the other. The formula below describes the simple scenario when two images are linearly blended.

$$I_{\text{blend}} = \alpha \cdot I_1 + (1 - \alpha) \cdot I_2 \tag{2.18}$$

I_1 and I_2 are the pixel values of the two images and I_{blend} represents the blended image. α is the weight factor and its value is between 0 and 1, which controls the contribution of each image and produces a gradual transition between the two images.

When the two images to blend have significant differences in color, texture, or brightness, an obvious seam or ghosting can occur. The Laplacian pyramid blend decomposes the images into multiple frequency levels, creates Laplacian pyramids for each image based on smoothing levels, and blends the pyramids at each level (Burt and Adelson [1983]). This will fill the invalid and edge pixels with neighboring values and minimizes abrupt transitions. The initial pyramids Blending operates in the intensity domain, which can struggle with global inconsistencies, such as differences in illumination. Pyramids that blend in the gradient domain minimize the seam by optimizing the Blending of image gradients, and have a significantly improved performance (Levin et al. [2004]). Gradient domain Blending can also be applied to other classic Blending methods such as feathering.

Image interpolation is another technique that is similar and related to image Blending in image processing. Interpolation refers to the calculation of intermediate pixel coordinates or values between known positions (Beier and Neely [1992]). Thus, it not only changes the pixel value but also changes the shape of the images. The implementation of Warping mentioned above also contains the interpolation processing. After operations like perspective Warping or more complex non-linear transformations, the pixel coordinates often become fractional. Thus, interpolation is required to determine the pixel intensities at these fractional positions to ensure visual smoothness and continuity. Common interpolation methods in Warping operation include nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation (OpenCV [2024]).

To smoothly transfer from the original image to the warped image, we can also use linear interpolation to calculate the intermediate images in the Morphing process. The basic linear interpolation of images can also be written mathematically with a weight factor α, which controls the weighted linear combination of the original coordinates (px, py) and their corresponding coordinates (x_{warped}, y_{warped}) in the warped images. The interpolation equations for calculating the intermediate positions $(x_{inbetween}, y_{inbetween})$ for each pixel are as follows:

$$\begin{aligned} x_{inbetween} &= (1-\alpha) \times px + \alpha \times x_{warped} \\ y_{inbetween} &= (1-\alpha) \times py + \alpha \times y_{warped} \end{aligned} \tag{2.19}$$

Here, α ranges from 0 to 1, indicating the degree of transition from the original image to the warped image. When α equals 0, the pixels remain in their initial positions, and when α is 1, the pixels reach their fully warped positions. By controlling the variation of α, a smooth variation between the original image and the warp image can be achieved through animation or user interface.

3. Related Work

3.1. Oblique Aerial Image Visualization Platforms

With an increasing number of oblique images made today and the growing need to view and analyze those data, several platforms have emerged to visualize and interact with these images. Here are the features and functions of some existing platforms.

- **United States Geological Survey (USGS) Oblique Aerial Photography Viewer (U.S. Geological Survey [2018]):** It is a web-based platform designed to provide access to oblique aerial imagery captured across the United States. As an interactive viewer, it allows users to explore oblique aerial images within a map interface as a reference to the location where the images are taken, and it is equipped with basic functions for the user to explore with panning and zooming (see figure 3.1).

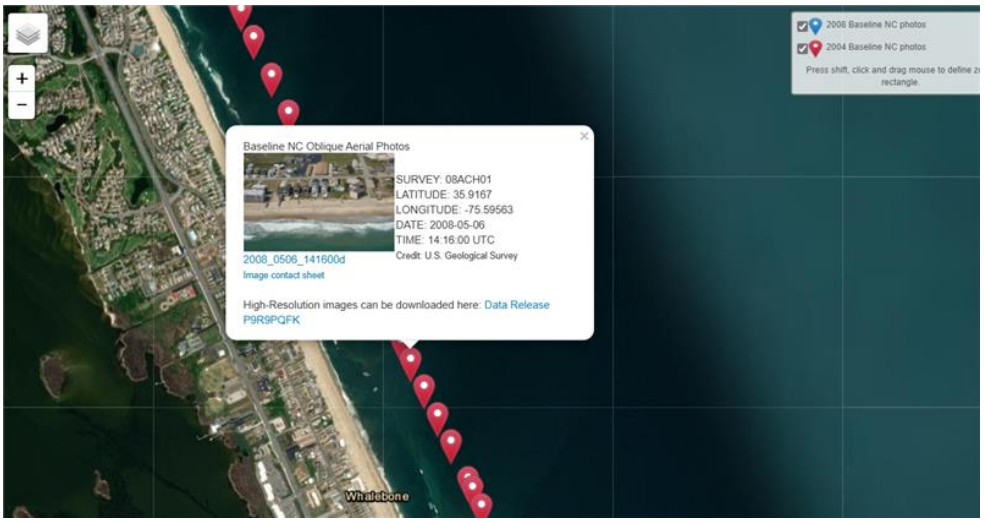

Figure 3.1.: User Interface of USGS Oblique Aerial Photography Viewer

  In addition, as a research platform, it offers users access to the meta data and easy download of the images. However, this platform stopped updating after 2018 and it does not stitch or perform any other process on the images for a smoother visualization experience.

- **Bing Maps Bird's Eye (Bing Maps Team [2022]):** Bing Maps released high-resolution oblique aerial images as well as nadir satellite images on its online map platform (see figure 3.2). They get oblique aerial images georeferred to the basic map and present oblique images when the mouse is moved over the nadir images of the corresponding

areas. This platform offers a very continuous experience when switching from a nadir image to its corresponding oblique image. However, it does not create a smooth transition between oblique images. Each time the user can only browse one oblique image. Also, as a platform for the general public, it does not have functions to make precise measurements from the images.

Figure 3.2.: Nadir and oblique images of the same area around Holland Spoor train station in Den Haag, source: Bing Maps

- **XMAP:** is a cloud-based GIS platform and the visualization of oblique aerial images is one of its modules. XMAP has built-in cascade and height measurement tools, which can apply accurate measurements for multiple scenarios. In addition, it can change the orientation of the oblique images taken in the same area. Like most of oblique aerial image visualization platforms, it aligns every oblique image with the base map. Although with a few seconds of delay, XMAP still realizes a relatively smooth transition from one oblique image to another. As a commercial software and platform, XMAP is not open source, so we cannot quantitatively assess its image transition or stitch quality.

3.2. Existing Softwares for Image Stitching

As image Stitching is widely used in photography, many existing softwares have developed the Stitching functions with a GUI and comprehensive pipelines. By just inputting two or more overlapped images into those GUI stitchers without pre-processing or precise camera parameters, users can obtain an integrated panorama. Some slight adjustments and fine-tuning can also be performed through those GUI to make the outcome better in visualization. The process of those image Stitching GUI is not very transparent because they only provide static stitched images without the intermediate parameters like the Homography matrix. Thus, these softwares cannot be directly used in this thesis project to generate

dynamic and measurable stitched images. However, studying the features and comparing the outcomes of them can still help to better understand the advantages and disadvantages between different stitching-related algorithms as well as users' needs. Here introduces four of the most commonly used Stitching softwares and their Stitching outcomes. Comparisons between the stitched results from existing softwares and this thesis project will be made in Chapter 6.

- **Adobe Photoshop (Adobe Inc. [2025])** : is a professional photo editing software that also supports image Stitching through its 'Photomerge' feature. It allows users to combine multiple photos into panoramas with control over blending, layout, and perspective correction. It is not open-source software and a paid license is required for use.
- **PTGui (New House Internet Services B.V. [2025])**: is a commercial panorama Stitching software. It is a complete pipeline for image Stitching with different purpose like creating spherical panoramas, Stitching large number of pictures into a huge integrate panorama, and projection image into a small planet.
- **AutoStitch(Brown [2025])**: is an automatic panorama Stitching tool developed as a research project. It integrates the Feature point finding and warping technique to stitch images together without requiring manual input or prior knowledge of camera parameters. Although Autostitch was originally developed as a research prototype and is freely available for noncommercial use, the source code has not been publicly released.
- **Hugin(Hugin Project [2025])**: is an open source panorama photo stitcher. It offers a powerful and flexible interface that supports manual and automatic control points, lens calibration, and various projection types for creating high-quality stitched images from overlapping photos. Since it is mainly used for research or art purposes, it is not very optimized in running speed.

3.3. Measurement on Oblique Aerial Images

One of the motivations for photogrammetry is to make precise measurements from photographs instead of field work on the spot. Measurement of oblique aerial images is an efficient way of landscape survey, especially for areas that are difficult for humans and instruments to directly access. In addition, it can save lots of time and money comparing to the on-site measurements.

The measurement from oblique aerial images is mainly realized by applying the reconstruction technique in photogrammetry. After constructing the projection matrix for the camera (M), the 3D coordinates of the corresponding 2D points can be calculated using triangulation methods. Figure 3.3 shows how the 3D coordinates align with the 2D image points in different image planes. The nonlinear triangulation method is mathematically used to solve a minimization problem and is widely used in the real world.

$$\min_{\hat{P}} \left\| M\hat{X} - x \right\|^2 + \left\| M'\hat{X} - x' \right\|^2 \tag{3.1}$$

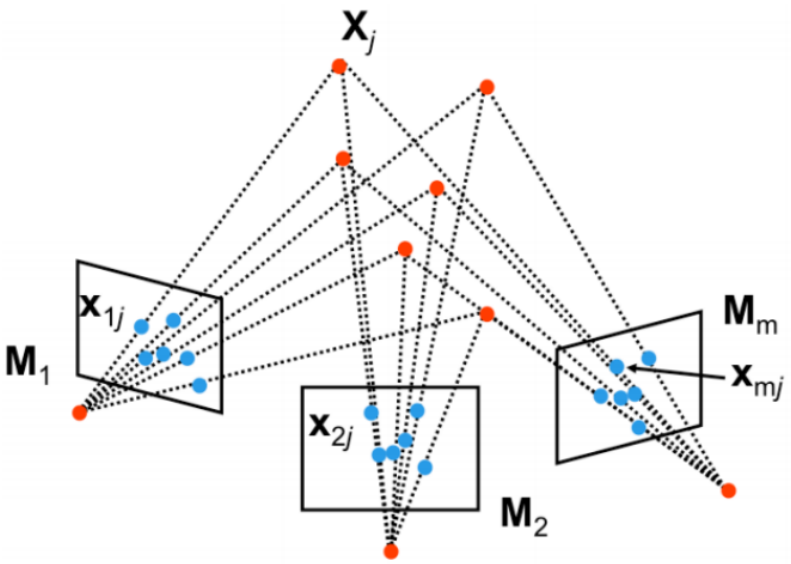

Figure 3.3.: Correspoinding points in 2D and 3D for reconstruction (3D Geoinformation, TU Delft [2025])

3.1 represents the sum of errors between the reprojected points and their corresponding points in two images of different views. The 3D point (X) can be estimated by finding the best least-squares estimation of $\hat{X}$ in both images. In practice, the sum of the reprojection error can be calculated from more than two images which have overlap areas.

Omnibase has already realized the measurement from oblique aerial images by using multiview overlapped images. Once a user clicked on a point on an image, it will detect other images which have the same point in real time to calculate the 3D terrain coordinates of the certain point. Then the distances between different points are calculated based on those 3D coordinates to measure the lengths or areas of the real-world objects. Figure 3.4 shows the measurement applied in Omnibase to measure the width of a standard football field. The left image is where the user is doing the measurement, and the right windows show the same point in other images. The top right shows the 3D coordinate of the current measuring point. The regulated width of a standard football field is 68 meters, and the Omnibase measurement result is 68.712 meters, which is a relatively accurate result.

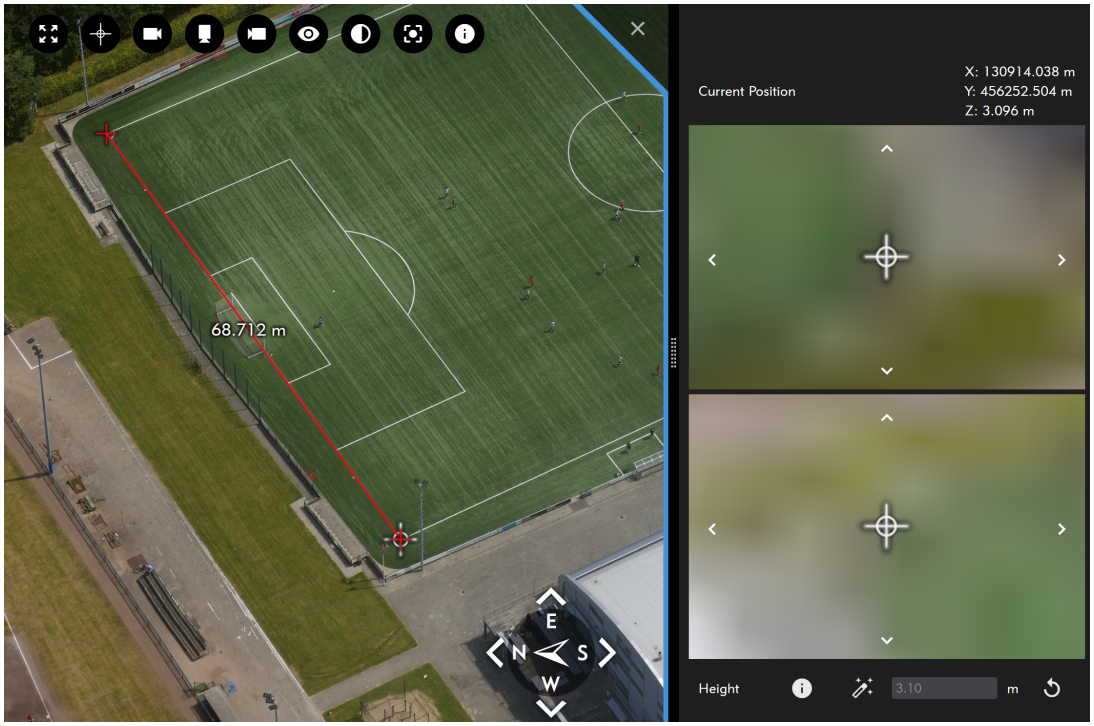

Figure 3.4.: Measurement on unwarped oblique aerial images in Omnibase

4. Methodology

This chapter gives an overview of the process of dynamic and measurable image Stitching. Figure 4.1 presents a general methodology and a processing flow chart of the dynamic Stitching program. The whole process can be divided into two main stages: preprocessing and image Morphing, and these two parts are introduced in Sections 4.1 and 4.2. The pre-calculation and adjustment of Homography matrix will be included in Section 4.2. The measurement is introduced as a separate part in Section 4.3.

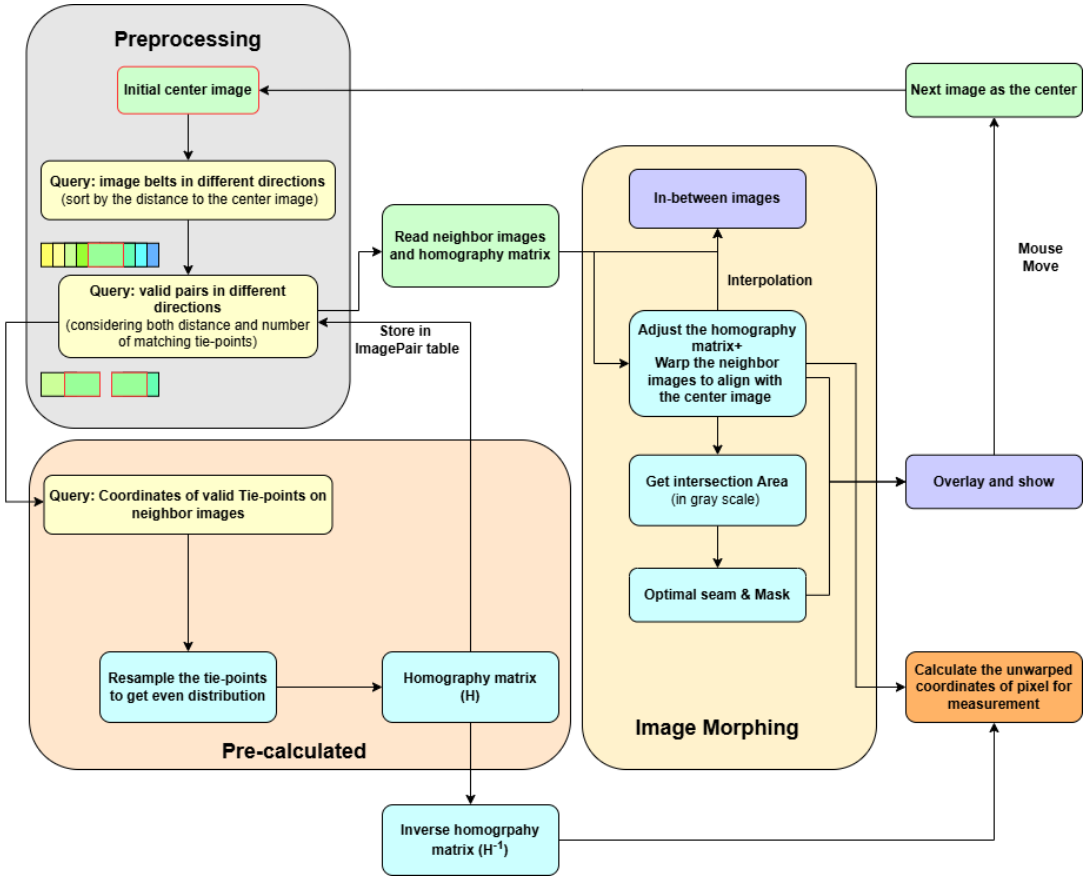

Figure 4.1.: General methodology flowchart

4.1. Preprocessing

Since Geodelta has applied the DISK-LightGlue algorithm to find the tie points and the results are already stored in Phoxy database, the preprocessing is mainly to query and get the valid image pairs and matching points from the image files and database.

4.1.1. Optical Aerial Oblique Imagery

The current dataset of this thesis project covers the municipality of Utrecht. The images were taken in 2024 with 45 degrees from four directions down to the earth's surface by a survey company, Kavel 10. The approximate positions of the images are recorded with the Global Navigation Satellite System (GNSS) system, and then a bundle-block adjustment is performed to obtain the adjusted positions. The images were taken with 50 meters interval on the same flightline, and the distance between flightline is around 150 meters. Some images that largely consist of water are labeled as unsuitable mapping due to the difficulty of finding the connection ground points for accurate adjustment. Table 4.1 shows some essential parameters of the images and cameras in this data set. In practice, the size and flight height of every image slightly varies, the table below only shows an average value of these two parameters. According to the flight height, focal length and sensor pixel size in the table, **gsd!** (**gsd!**) is around 1.75 cm around the principal point.

Image Width	10652(pixels)
Image Height	14204(pixels)
Size on Disk	∼75 (MB)
Focal Length	108 (mm)
Flight Height	∼420 (m)
Sensor Width	40.05152 (mm)
Sensor Height	53.40704 (mm)
Sensor Pixel Width	3.76 (µm)
Sensor Pixel Height	3.76 (µm)

Table 4.1.: General image properties and sensor parameters of the used data set

The lens distortion for this data set is mainly a radial distortion (see equation 4.1, r is the pixel distance from the principal point), which is primarily caused by imperfections in the lens design. The radial lens distortion coefficients are shown in the table 4.2. Comparing with universal cameras (Choi and Kim [2021]) in daily life, the professional cameras using to generate the aerial images have relatively low distortion. Since lens distortion does not have a significant influence on the Stitching results, lens distortion calibration is not included in preprocessing.

$$r_d = r(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{4.1}$$

Coefficient	Value (Approximate)
k1	-5×10^{-6}
k2	9.5×10^{-10}
k3	$\pm 3 \times 10^{-13}$

Table 4.2.: Estimated Radial Distortion Coefficients from Calibration for a camera

4.1.2. Phoxy Database

Phoxy is a platform now under development by Geodelta to show the Tie-points between two images (see figure 2.3). However, Phoxy is still in development, especially for its interface. In this thesis project, the Phoxy database and some subsets of it are mainly used. Phoxy database provides the records including the interior and exterior camera parameters of every image and the Tie-points coordinates.

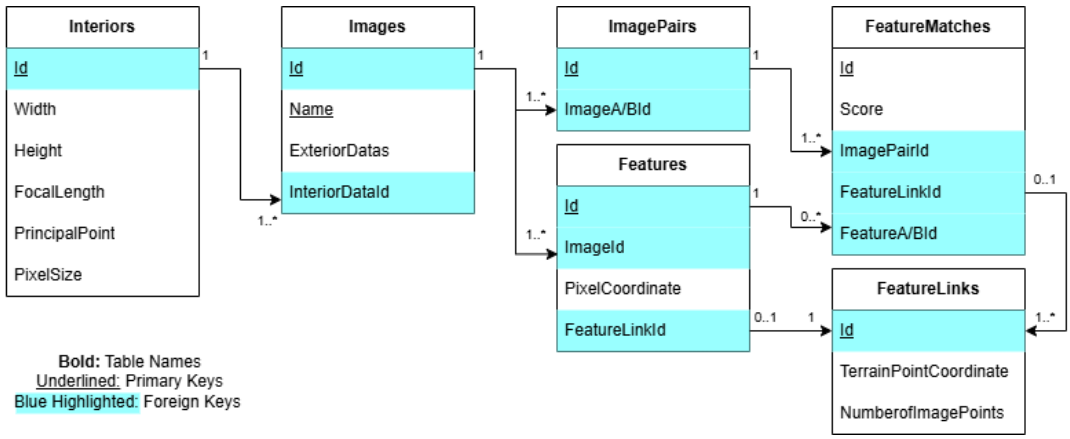

Figure 4.2.: Database structure of used Phoxy tables in this thesis

Figure 4.2 shows the tables and their relationships in the database that are used in this thesis. The following gives a more detailed introduction to the tables used:

- **Interiors:** Records of interior parameters. The images taken with the same camera share the same record in the interior tables.
- **Images:** Basic information about every image. Unique IDs are created randomly.
- **ImagePairs:** Matched pairs of overlapped images. Each pair record has a unique pair Id and Ids of two matched images.
- **Features:** Detected key points in a single image. The pixel coordinates of every feature on the image plane are recorded. The same feature that is detected on more than three images has a FeatureLinkId.
- **FeatureMatches:** Matched pairs of Tie-points. If the pair of points are also detected in other images, the FeatureLinkId record is not null. The score given by LightGlue is also recorded for matched features. The higher the score, the more convincing the pair of matched points is.

- **FeatureLinks:** Linked key-point record. Since the point is detected by more than three images, it's terrain coordinate can be precisely calculated with the triangulation method in photogrammetry.

The original Phoxy database of Utrecht contains 2860 image records, 144378 image pairs, and more than two millions of feature matchings and features. Subsets of the original database are made for different experiments and the basic structure of the database is maintained.

4.1.3. Data Cleaning and Resampling

Despite query and data retrieval, data cleaning is also important for Tie-points because of errors in detection and matching points. In principle, only the key points with FeaturelinkId will be selected as input points for image warp due to high confidence. However, for some image pairs that don't have enough Tie-points with Featurelinks, the score of Tie-points is also set as a selection standard. If the key point has a score greater than 0.8, it will also be selected as a valid input for future steps.

Because of the scene features, some of the images have detected key points with uneven distribution (see figure 4.3). These unevenly distributed points within the overlap area will lead to errors in Homography matrix calculation.

Figure 4.3.: Uneven distribution of key points in overlapped area

In figure 4.3, the upper left buildings have too many detected key points, which is unnecessary. Thus, resampling the points to decrease the weights of upper left corner can be a

solution to this problem. K-means clustering algorithm is used to detect different grouping of points. For each group, the central point is selected as the representative point for Homography matrix fitting. The resampling processing not only reduces the influence of uneven distribution but also decreases the number of input points to Homography matrix fitting and improves the runtime performance. Collinearity checking is also applied to grant proper Homography matrix generation in extreme cases. Detailed implementation will be introduced in Chapter 5.

4.2. Image Stitching

4.2.1. Image Warp and Transformation

As mentioned in Chapter 2, parametric Warping has multiple advantages, including its low computation cost and explicit expression. Although parametric Warping can also be applied with mesh-based techniques to achieve local Warping (Efros [2016]), the oblique aerial images usually only have global perspective distortion caused by viewing angels and camera movements. Thus, using global Warping with one single Homography matrix can already achieve a good Stitching result which has minimal errors like duplicate or broken objects and keeps all the edges straight. In addition, the simplicity of the single homography transformation ensures fast computation performance and low memory costs, which makes it possible for dynamical or even real-time Stitching. Global transformation is used to warp the image to stitch to the reference image.

In Chapter 2, two different global Warping methods (2D and 3D) are introduced. Although they have different physical meanings, they still give the transformation expressions in the same mathematical form, a 3×3 Homography matrix. The following describes how to generate the Homography matrix with these two methods:

- **2D:** All the valid matching point pairs are input into the a planar projective transformation solver to estimate the Homography matrix. The Open Source Computer Vision Library (OpenCV) findHomography function (OpenCV [2024]) is chosen to solve the Homography matrix.
- **3D:** The Homography matrix can be formed directly by the camera parameters. Equation 4.2 shows the projection matrix (H_i) from the ground to the image. Thus, the inverse matrix (H_i^{-1}) is the projection matrix from the image to the ground. The projection matrix from image1 to image2 can be formed by multiplying the project matrix to image1 plane to ground (H_1^{-1}) and the projection matrix from the ground to the image2 plane (H_2). H_{12} in Equation 4.3 is the projection matrix from image1 to image2. Here, $K \in \mathbb{R}^{3\times3}$ is the intrinsic matrix of the camera, $R_i \in \mathbb{R}^{3\times3}$ is the rotation matrix representing the orientation of the camera, and $t_i \in \mathbb{R}^3$ is the translation vector representing the position of the camera in space.

$$H_i = K \begin{bmatrix} R_i^{[1:2]} & t_i \end{bmatrix} \in \mathbb{R}^{3\times3}, \quad \text{only using first two columns of } R_i. \tag{4.2}$$

$$H_{12} = H_2 \cdot H_1^{-1} = \left(K \begin{bmatrix} R_2^{[1:2]} & t_2 \end{bmatrix} \right) \cdot \left(K \begin{bmatrix} R_1^{[1:2]} & t_1 \end{bmatrix} \right)^{-1} \tag{4.3}$$

Although the 3D method has the advantage that camera movement is considered, lack of consideration in terrain elevation can still cause errors to this method. Thus, the points warped by H_{12} will be entered again into the projective transformation solver with their matching points to obtain a Homography matrix H'. The final transformation matrix is calculated by equation 4.4:

$$H = H_{12} \cdot H' \tag{4.4}$$

The homography calculated by the Tie-points from the matching point pairs will be stored in the same table as the matched image pair results. When Stitching processing starts, the corresponding Homography matrix will be loaded as the best pair matching is selected. Thus, the time for homography calculation is saved for the dynamical Stitching process. Also, by pre-calculation of Homography matrix, the mismatched pairs can be detected and repaired to avoid bugs and large errors in the dynamical Stitching process.

Despite the homography transformation, there are two other kinds of transformations that occur in the image Stitching process, the scale and the translation. Because the size of the raw oblique images is very large (4.1), it is more efficient to scale them to achieve good real-time performance. The Homography matrix is calculated on the basis of the original image, it cannot be directly applied to the scaled images. Scaling to the Homography matrix is a must if the images are scaled. The matrix 4.5 is the scale matrix and s is the scale factor. The Homography matrix is scaled by equation 4.6.

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$H_s = S \cdot H \cdot S^{-1} \tag{4.6}$$

In the warp transformation, at least one image is selected as the reference image, and other images are warped to stitch to it. Since the image coordinate system sets the upper left corner as the original coordinate, part of the warped image on the left side of the reference image will have negative coordinates. The mainstream computer vision and matrix computation libraries like OpenCV, Scikit-image, and Tensorflow do not support the negative index. Thus, Stitching the image on the left or up side, the negative direction of x and y axis respectively, needs translation of the reference image's coordinate system to give enough space to the left and up images (see figure 4.4).

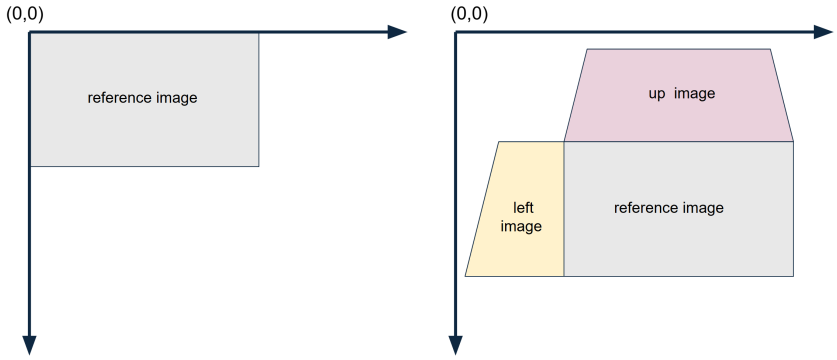

Figure 4.4.: Positive translation in both x and y axis of the reference image

The translation matrix can be written as matrix 4.7. tx and ty are the translations needed from the x and y axes. In most cases, both scale and translation are applied and 4.8 shows the calculation of the final Homography matrix.

$$T = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$H_{final} = T \cdot S \cdot H \cdot S^{-1} \tag{4.8}$$

4.2.2. Optimal Seam Finding

In this project, the dynamic programming algorithm is chosen for optimal seam finding. The algorithm operates by first calculating a cost or energy function based on the differences of pixels in overlapping regions. This cost typically reflects intensity or gradient differences, which quantify visual disparity at potential seam positions. Then, the algorithm will initialize two matrices that have the same shape as the energy map, the cost matrix, and the backtrack matrix (see figure 4.5). The cost will be computed and accumulated from one boundary of the overlapping region to the opposite side. At each pixel location, the algorithm evaluates the minimal cumulative cost of adjacent pixels, effectively tracing the path of the least visual difference. After filling the cumulative cost matrix, the algorithm identifies the optimal seam by following backward from the position of the minimum accumulated cost at the opposite boundary, following the path that contributed to this minimal total. Consequently, this produces an optimal seam with minimal cumulative visual discrepancy, ensuring a smooth transition between stitched images.

Figure 4.5 shows the input energy map and the calculated result of the cost matrix and the backtrack matrix in a horizontal seam carving process. The energy map (left matrix) represents the importance or disruption cost per pixel, typically calculated using an image gradient magnitude. High values correspond to visually significant regions, while low values indicate visually uniform areas suitable for removal. Each element $E(i, j)$ denotes the energy of pixel in row i, column j. The NaN value in the input will be considered infinite in the cost calculation to avoid having to go through those pixels. The final optimal seam is

labeled blue in the backtrack matrix. The cost matrix (middle) stores the cumulative minimum energy required to reach each pixel from the top row. This matrix is initialized by copying the first row of the energy map, since no previous pixels influence it. For all other pixels (i, j), the cumulative cost $M(i, j)$ is computed as:

$$M(i,j) = E(i,j) + min(M(i+1,j-1), M(i+1,j), M(i+1,j+1))$$

where only valid neighboring positions are considered. If a value is inf, it typically indicates an impassable or undefined path due to NaN in the energy map.

The red arrows on the energy map (left image in Figure 4.5) visualize this recursive minimization process, showing how the minimum path propagates down the matrix. While the cost matrix(C) stores the minimum cumulative energy required to reach each pixel, the backtrack matrix(B) preserves the origin of that minimal cost, allowing for seam path reconstruction via backward traversal. For the horizontal seam, the backtrack matrix stores the row indices of the element in the $j-1$ column in the element j column. Here, in the example of figure 4.5, the minimum cumulative energy pixel is at $C(2,4)$, and the minimum value is 16. The recorded front-row indice at $B(2,4)$ is 3, which means the last energy pixel is at $C(3,3)$. Repetition of this backtrack processing will finally find the optimal seam in the image, which is labeled blue in the backtrack matrix of Figure 4.5.

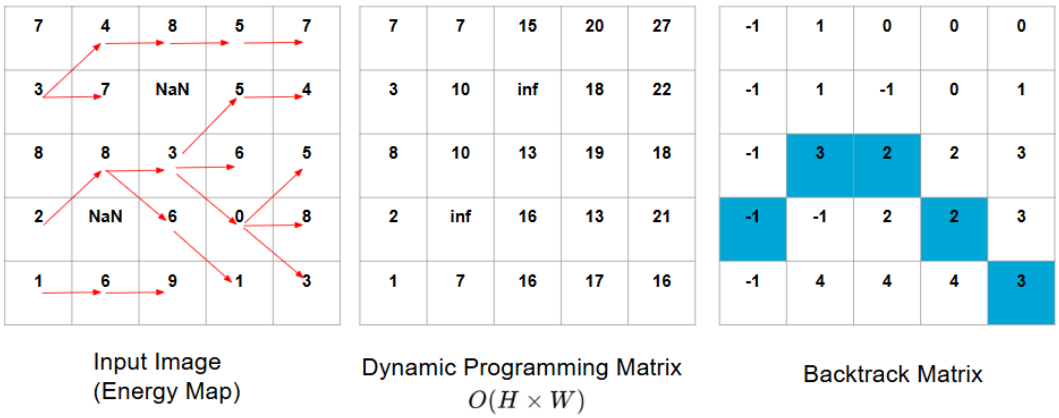

Figure 4.5.: Left: input energy map; middle: cost matrix; right: backtrack matrix.

4.2.3. Interpolation

The goal of interpolation is to generate the intermediate images between the orginal images and the warped images to achieve a smooth transformation. Since the Homography matrix is a nonlinear, reversible, singular matrix, directly interpolation between different transformation matrices is a practical solution to generate the in-between image.

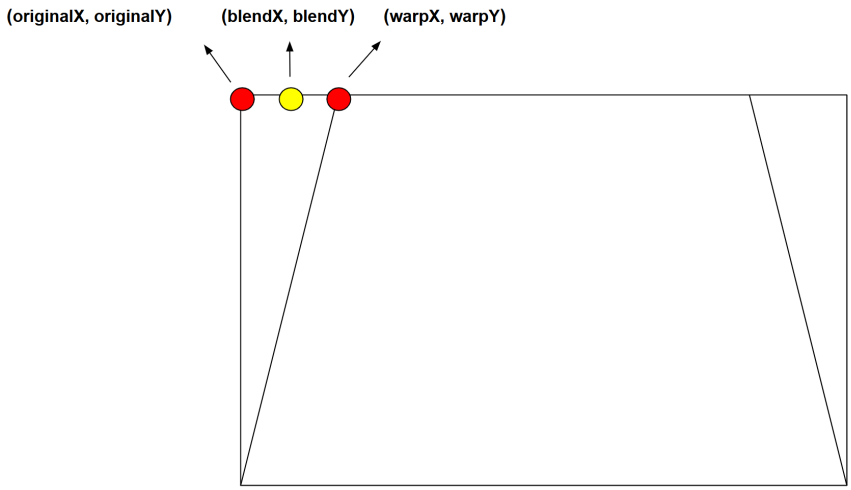

Figure 4.6.: Example of a blend pixel generated by the original pixel and the warped pixel

Figure 4.6 shows an example of where the interpolated pixel should be. The interpolation process calculates the coordinates of every blend pixel and remaps these pixel to the coordinate. Equation 4.10 presents the calculation of the pixel coordinates of a simple linear interpolation.

$$\text{blendX} = (1 - \alpha) \times OriginalX + \alpha \times warpX, \tag{4.9}$$
$$\text{blendY} = (1 - \alpha) \times OriginalY + \alpha \times warpY \tag{4.10}$$

4.3. Measurement Transformation

As mentioned in Chapter 3, Geodelta has developed an online measurement tool for aerial oblique images based on a nonlinear triangulation method. However, the code for the measuring tool is embedded in the back-end code of Omnibase instead of acting as an independent application programming interface (API), so it is hard to integrate the current Warping or Stitching images into the measurement system directly. The main goal of this part is to prove the measurability of the warped and stitched image.

To measure on the warped image, there is just one more step to do than measuring on the original image, that is, finding the original image pixel coordinates out of the warped ones. Since the input Tie-points are not collinear, the Homography matrix must be full-rank and invertable (Bernstein [2009]). Thus, simply applying the inverse Homography matrix (H^{-1}) to the point clicked by the user on the warped images, the coordinates of the original points on the unwarped image can be calculated. Then the original coordinates can be put into the existing triangulation pipeline for measuring.

5. Implementation

This chapter presents the implementation details of the interactive image Warping system using EmguCV and C#. It begins with the architectural layout of the system and design rationale behind key components. Furthermore, the chapter illustrates the step-by-step integration of user interactions, matrix operations, and rendering processes, emphasizing both technical and visual accuracy. Practical challenges encountered during development are also addressed, along with the solutions adopted. This chapter aims to provide a comprehensive understanding of the implementation mechanics, laying the groundwork for evaluation in the subsequent section.

5.1. Preprocessing

5.1.1. Query Operations

SQLite is a lightweight, serverless, self-contained Structured Query Language (SQL) database engine (SQLite Consortium [2025]). Unlike traditional database systems that require a separate server process, SQLite integrates directly into applications, using simple function calls to manage and query data. Thus, SQLite is a simple and efficient data management tool for this project.

To optimize query speed, indexes are created for every table in the database based on the primary keys and foreign keys. In the following, the query process is introduced step by step.

- **Read the initial image:** The whole query process starts with the table **Image** to read the parameters and the image file of the center reference image.

- **Read the neighbor image belt:** The images that are valid to stitch to the current center images have to satisfy two conditions. First, it is in the same sampling row or column as the center image. Second, it has the same direction of view as the center image. To filter qualified images, the optical center coordinates and orientation parameters are taken into consideration.

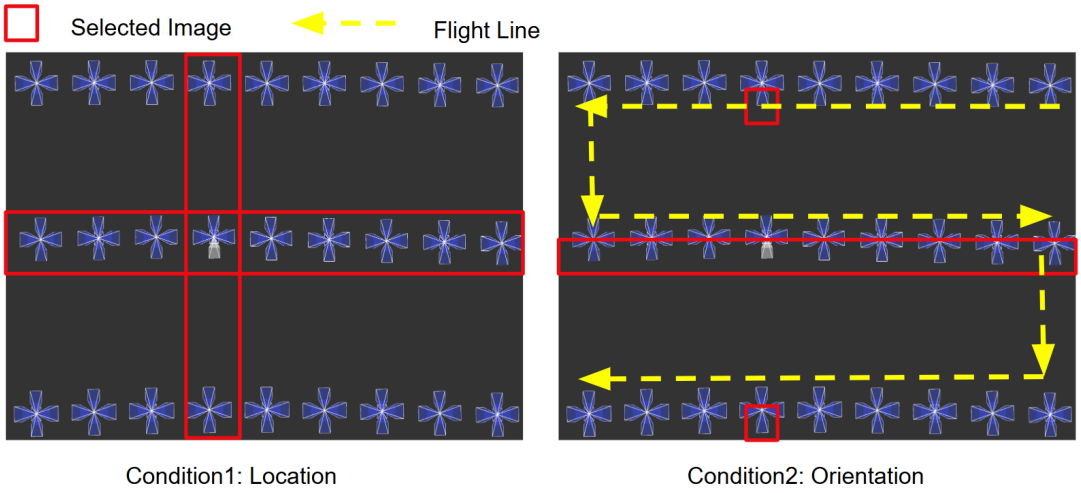

Figure 5.1.: Selected images on the image belt for stitching to the selected image in white frame

For images in the same column or row, the distances between their X or Y coordinates are less than 10 meters. So, the condition for images with qualified location in the column or row is:

$$CenterImage.X - 10 < SelectedImage.X < CenterImage.X + 10$$

or

$$CenterImage.Y - 10 < SelectedImage.Y < CenterImage.Y + 10$$

As images with the location on the same row or column are found, the next step is to make sure that the image is looking from the same direction to the ground. The camera rotation parameters, also called the orientation parameters, are recorded for every image. Since this is a dataset with images taken by 45° angle from a Maltese-cross camera system (see figure 2.1), there are four possible combination of ω and ϕ for x and y axis orientation.

Orientation	ω	ϕ
1	-45	0
2	45	0
3	0	45
4	0	-45

Table 5.1.: Four possible combination of ω and ϕ parameters (unit: degree)

All the images are reclassified into those four orientation categories for the view direction checking in queries.

- **Select the best image pairs to stitch:** Although the closest image to the center image on the image belt is most likely to be the best match to stitch, there are sometimes not

enough Tie-points detected or no point between the closest pairs due to errors in the Tie-point matching results. However, the closest image is still most likely to be the best image for Stitching. Thus, the images on the image belt are sorted by the distance to the center image from shortest to longest for a loop query. The pseudo-code of the best pair selection is shown in 5.1.

Algorithm 5.1: GetBestPair Algorithm

```
Input: Image belt B, image info I, database connection C
Output: Valid image pairs P
Initialize empty list P ;
foreach image b in B do
    Retrieve image pair p for (I_0, b) from C ;
    if p exists and has > 10 feature matches then
        Create new pair r with correct A, B order ;
        Append r to P ;
return P
```

- **Get the Tie-points:** After deciding on the image pair to stitch, the record for Tie-point matching results can be retrieved from table **FeatureMatch** which also records the Id for the corresponding points on two images. By transferring the Ids into a list, SQLite will execute a very fast batch query to get the coordinates of the Tie-points on both image planes.

5.1.2. Tie-point Data Cleaning and resample

As mentioned in chapter 2, the mismatched pairs and uneven distribution of point pairs may influence the final homography calculation. The cleaning step is applied while querying for the feature-match records.

Feature links consist of three or more matched features that have been forward-intersected to create a ground object point. Thus, Feature points that are part of a feature link are thus considered reliable, and their field of "Score" is higher than 0.8.

To get enough reliable point pairs, a threshold of a minimum of point pairs is set. If the number of points with "FeatureLink" is greater than the threshold, then the point pair list contains only the linked points. If there are not enough linked points, the point pair list will get other point pairs with high matching score until the number of point pairs in the list is greater than the threshold. Here, the threshold of points is set to 100 according to the results of the geometric accuracy experiment in Chapter 6.

To solve the uneven distribution problem, the k-means clustering algorithm is applied to divide the Feature points into different clusters and select the center point as the final input. The pseudocode for the clustering of K-means is shown in 5.1 and Figure 6.11 visualizes the selection result. The clustering number is set to 20.

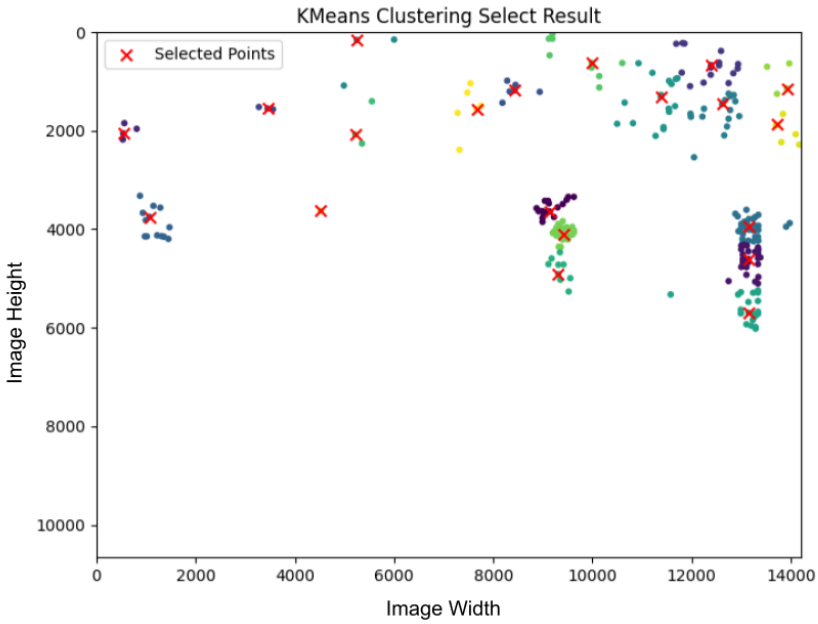

Figure 5.2.: K-means clustering and selection results of an example point pair dataset

Algorithm 5.2: Cluster-Based Point Selection

Input: Point sets P_A, P_B; number of clusters k
Output: Selected points P'_A, P'_B

Apply KMeans clustering with k clusters on P_A ;
Initialize empty index list I ;
for $i \leftarrow 0$ **to** $k-1$ **do**
    Find cluster points C_i where label = i ;
    Compute distances to cluster center C_i^{center} ;
    Let j be index of point in C_i closest to C_i^{center} ;
    Append j to I ;
Set $P'_A \leftarrow P_A[I]$, $P'_B \leftarrow P_B[I]$;
Visualize clustering and selected points ;
return P'_A, P'_B

Although not encountered in practice, there is still the possibility that in some extreme cases the selected 20 points are collinear. Collinearity checking will be done applied before input those points into Homography matrix construction. If those points are collinear, the clustering progress will restart, and the selection condition from each clustering will be two random points in every cluster in a clustering will be taken until the group of points passes the collinearity checking (see Figure 5.3). If after 1000 iterations the points are still collinear, then the program will try to input all points to collinearity checking. If they are still collinear, the program throws warnings.

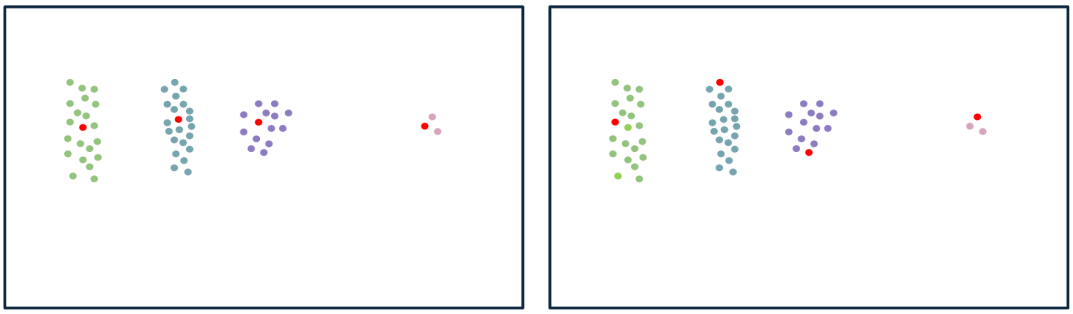

Figure 5.3.: Left: Collinear points selected (marked in red) under clustering-center condition. Right: Non-collinear points selected by random iterations. Note: This is an artifically generated example instead of real data

5.2. Image Stitching

Most of the image processing is performed by the OpenCV library (OpenCV [2025]). OpenCV is an open-source software library primarily aimed at real-time computer vision and image processing tasks. OpenCV provides a comprehensive collection of optimized algorithms and functions written in C/C++, with interfaces available for languages such as Python and Java. EmguCV is a cross-platform .NET wrapper for the OpenCV library, allowing the use of OpenCV functions in.NET languages such as C#. The following context mainly introduces the functions referring to OpenCV library and would mention its corresponding function in Emgu CV as the project is done mainly with C#.

5.2.1. Image Warp

After the data is cleaned, the tie points are entered into the OpenCV **findHomography** function (CvInvoke.FindHomography in EmguCV) to calculate the Homography matrix (OpenCV [2024]). A RANSAC algorithm (Fischler and Bolles [1981]) is used to minimize the influence of outliers. It repeatedly selects a random subset of point pairs from the tTie-point dataset to estimate a tentative homography. Then, it projects points using the estimated homography and checks which point pairs agree within a certain error threshold (inliers). The default threshold here set by OpenCV is 3. After many iterations, the model with the highest number of inliers is selected as the final homography.

With the calculated Homography matrix, the **warpPerspective** function of OpenCV (CvInvoke.WarpPerspective in EmguCV) is used to generate the warped image (OpenCV [2024]). This function integrates two main operations, namely, calculating the warped coordinates (see Equation 5.1) and interpolating. The default interpolation method is bilinear interpolation.

$$\mathrm{dst}(x,y) = \mathrm{src}\left(\frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}}, \frac{H_{21}x + H_{22}y + H_{23}}{H_{31}x + H_{32}y + H_{33}}\right) \tag{5.1}$$

For the 3D warp method, as shown in Chapter 4, it can consider camera movement without the influence of the terrain. In practice, the camera parameters, especially the transformation parameters, and the spatial position of the camera center when taking the image are recorded in geographic coordinates. However, the cameras are facing four different directions, and the axis of the image coordinate may not be in the same orientation as the geographic coordinates. Thus, before the projection, rotating the image and Tie-points to make the coordinate system in the same direction as the geographic coordination system are must to solve the orientation issues.

The 3D warp method only implies as an experiment in Python code, and the final C# demo only includes the 2D warp considering the balance between accuracy and run-time. The comparison between those two Warping methods is shown in Chapter 6. Thus, here the implementation of the 2D methods is mainly introduced. The implementation with 3D methods is also based on this and the camera projection matrix is formed based on the query results of camera parameters.

5.2.2. Optimal Seam Finding

The Optimal Seam Finder algorithm employs dynamic programming to efficiently identify a seam path with the minimum cumulative energy through an energy map, which represents pixel discrepancies or gradients.

The energy map is generated by the grayscale image of the overlapped area. To improve the run-time and get a smoother seam, the input image is downsampled by a factor of 20, which means the pixel number of both length and width is divided by 20 from that of the original image.

A Sobel operator (Sobel [2014]) is applied to the image to calculate the energy map. The Sobel operator detects gradients in an image; specifically, it calculates the first-order derivatives in the horizontal (x) and vertical (y) directions. When the value calculated by the Sobel operator is high, it indicates that the area could be edges or full of texture details, where the seam should avoid being blown. Thus, these areas are of high value on the energy map (see Figure 5.4). In particular, the areas with NaN values in the image are set to infinite on the energy map.

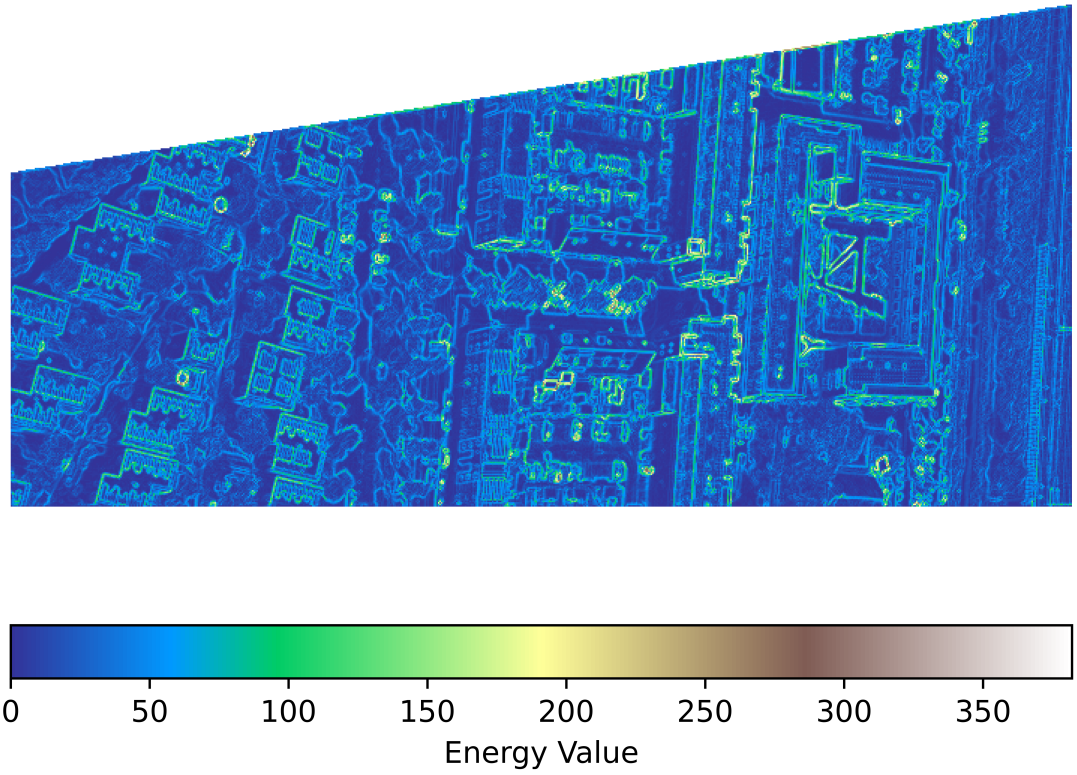

Figure 5.4.: Example of energy map of between certain images

As described in Chapter 4 and Figure 4.5, the optimal seam is calculated based on the energy map with the dynamic programming algorithm. Pseudocode 3 shows the implementation of the algorithm in the horizontal direction. The direction of the seam is decided by comparing the aspect ratio of the overlapped area before the seam calculation. If the width of the overlapped area is longer than its height, the horizontal seam is applied. Otherwise, it will try to find a vertical seam. The vertical seam finding analogy is basically the same as the horizontal one except for the replacement of row indices by column indices. Since the energy map is calculated by a downsampled image, the final seam is also resampled to match the original image and generate the mask.

Algorithm 5.3: Seam Finding via Dynamic Programming in Horizontal Direction

Input: Energy map E of size $H \times W$
Output: Minimum-cost horizontal seam path $\mathcal{S}$ and its cost C

Convert E to a NumPy array of type `object`;
Initialize $M \leftarrow$ matrix of size $H \times W$ with values $+\infty$;
Initialize $B \leftarrow$ matrix of size $H \times W$ with values -1;
for $r \leftarrow 0$ **to** $H-1$ **do**
    if $E[r,0]$ *is valid* **then**
        $M[r,0] \leftarrow E[r,0]$;
for $c \leftarrow 1$ **to** $W-1$ **do**
    for $r \leftarrow 0$ **to** $H-1$ **do**
        if $E[r,c]$ *is invalid* **then**
            continue;
        $best\_cost \leftarrow M[r,c-1]$, $best\_row \leftarrow r$;
        if $r > 0$ *and* $M[r-1,c-1] < best\_cost$ **then**
            $best\_cost \leftarrow M[r-1,c-1]$, $best\_row \leftarrow r-1$;
        if $r < H-1$ *and* $M[r+1,c-1] < best\_cost$ **then**
            $best\_cost \leftarrow M[r+1,c-1]$, $best\_row \leftarrow r+1$;
        if $best\_cost < \infty$ **then**
            $M[r,c] \leftarrow E[r,c] + best\_cost$;
            $B[r,c] \leftarrow best\_row$;
$min\_cost \leftarrow \infty$, $min\_row \leftarrow -1$;
for $r \leftarrow 0$ **to** $H-1$ **do**
    if $M[r,W-1] < min\_cost$ **then**
        $min\_cost \leftarrow M[r,W-1]$, $min\_row \leftarrow r$;
if $min\_row = -1$ **then**
    return $(\emptyset, \infty)$;
$\mathcal{S} \leftarrow []$, $curr \leftarrow min\_row$;
for $c \leftarrow W-1$ **to** 0 -1 **do**
    Append $(curr, c)$ to $\mathcal{S}$;
    $curr \leftarrow B[curr, c]$;
    if $curr < 0$ **then**
        break;
Reverse $\mathcal{S}$;
return $(\mathcal{S}, min\_cost)$;

5.2.3. Dynamic Interpolation

The dynamic interpolation algorithm is designed to provide the user with a smooth transformation between images. Algorithm 4 shows the dynamic interpolation process to continuously warp an image with mouse-drag control. In an ideal output, this algorithm should be applied to three images at the same time. However, due to the limitation of time and the difficulties in calculating the post-warp Homography matrix, the final demo does not include

the implementation of this interpolation algorithm. However, the interpolation algorithm is still documented for future software development.

Algorithm 5.4: Drag-based Interpolation Algorithm

Input: Original image I, homographies H_{21}, H_{23}, blending factor α
Output: Controllable warped image displayed via `pictureBox`
$w \leftarrow \text{width}(I) + 800$
$h \leftarrow \text{height}(I)$
$\text{maxOffset} \leftarrow 200$
if $\alpha < 0.5$ **then** $\text{interpH} \leftarrow \text{InterpolateMatrix}(H_{21}, I_d, 2\alpha)$
$tx \leftarrow (1 - 2\alpha) \cdot \text{maxOffset}$
else $\text{interpH} \leftarrow \text{InterpolateMatrix}(I_d, H_{23}, 2(\alpha - 0.5))$
$tx \leftarrow -2(\alpha - 0.5) \cdot \text{maxOffset}$
$T \leftarrow \text{CreateTranslation}(tx, 0)$
$\text{centerT} \leftarrow \text{CreateTranslation}\left(\frac{\text{width}(I)-w}{2}, 0\right)$
$\text{temp} \leftarrow T \cdot \text{interpH}$
$\text{totalH} \leftarrow \text{centerT} \cdot \text{temp}$
foreach $y \in [0, h)$ *in parallel* **do**
    for $x \leftarrow 0$ **to** $w - 1$ **do**
        $\text{denom} \leftarrow H_{2,0}x + H_{2,1}y + H_{2,2}$
        $xx \leftarrow (H_{0,0}x + H_{0,1}y + H_{0,2})/\text{denom}$
        $yy \leftarrow (H_{1,0}x + H_{1,1}y + H_{1,2})/\text{denom}$
        $\text{mapX}[y, x] \leftarrow xx$
        $\text{mapY}[y, x] \leftarrow yy$
$\text{displayImage} \leftarrow \text{Remap}(I, \text{mapX}, \text{mapY})$
`pictureBox.Image` $\leftarrow \text{ToBitmap}(\text{displayImage})$

5.3. Runtime Optimization

Figure 4.1 represents the basic process of the dynamic image Stitching program. However, if progress is made simply with a single thread following the arrows in Figure 4.1, the real-time performance will be poor due to the waste of computational resources. Thus, run-time optimization has been applied to improve real-time performance mainly by using parallel computation and precalculation. The runtime optimization and the demo run with the original images scaled with a factor of 0.2 to achieve a balance between visual effects and runtime performance.

Pre-calculation of the scaled image and Homography matrix can save much computation time according to the results in Chapter 6. In addition, by parallelizing synchronous processes, real-time Stitching performance is also improved. Figure 5.5 shows the optimized threads in the dynamic Stitching demo. Image loading and processing are performed in parallel to improve runtime performance.

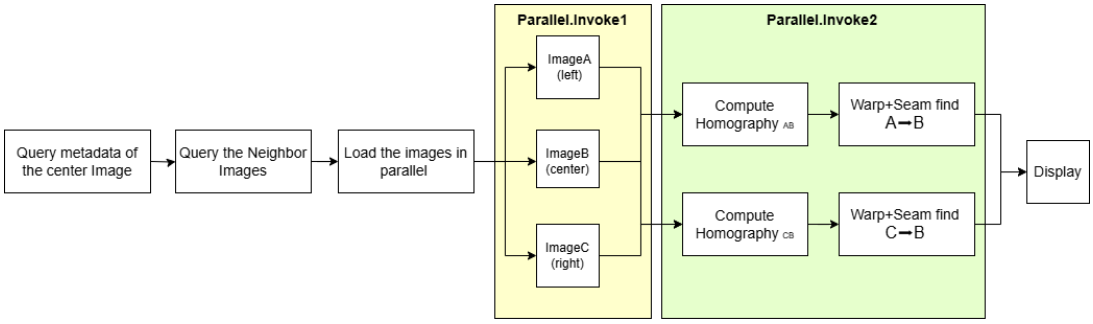

Figure 5.5.: Threads in real-time image stitching

5.4. Reprojection for Measurements

The inverse Homography matrix is calculated using the **CvInvoke.Invert()** function in C# with EmguCV. Because the pixel coordinates are always integers, the reprojected points by the inverse Homography matrix have to be rounded.

6. Results and Assessment

6.1. Visualization of Stitched Images

In this chapter, static stitched images in different intersecting scenarios are shown in 6.1.1. Since the original aerial images are very large in file size, the static stitched images here are all downsampled. To show the details of the stitched image, some zoom-in batches are cut and shown in 6.1.2. Comparison between the result stitched by this project and some existing software are shown in 6.1.3. This part mainly shows the visualization result of the stitched images and the geometric accuracy with quantitative calculation are shown in 6.2.

6.1.1. Overview of Static Stitched Images

Figure 6.1.: Three stitched oblique aerial images in the same flightline without optimal seam in a GUI view

Figure 6.2.: Three stitched oblique aerial images in the same flightline with optimal seams in a GUI view (Scenario 1)

Figures 6.1 and 6.2 show the same group of images. The results in Figure 6.1 confirm the effectiveness of the Homography matrix in aligning images with high geomatric precision. Figure 6.2 applied the optimal seam, where the seam in 6.2 can still be detected by human eyes, but it is not as obvious as the seam in 6.1. The seam adapts to image content, preserving high-frequency features such as edges while avoiding high-gradient areas that would otherwise be visually disruptive. This confirms that our approach achieves high-quality Stitching in terms of both visual coherence and geometric alignment.

This part also shows the images stitched in four different intersection scenarios. Table 6.1 gives the description and intersection rate of the image pairs shown in Figure 6.2, 6.3, 6.4, and 6.5, respectively. The intersection rate is calculated based on the pixel number of warped images divided by the pixel number of the intersection area after Warping.

Stitching Scenario	Scenario Description	Intersection Rate (%)
1	Same flightline + High intersection	62.3
2	Same flightline + Low intersection	19.1
3	Across flightline + Same Column	50.9
4	Across flightline + Diagonal Intersection	33.3

Table 6.1.: Stitching scenarios and the intersection information for the shown image pairs

Figure 6.3.: Two oblique aerial images in the same flightline with low intersection rate stitched (Scenario 2)

Figure 6.4.: Two oblique aerial images across the flightline stitched (Scenario 3)

Figure 6.5.: Two diagonal intersected oblique aerial images stitched (Scenario 4)

6.1.2. Details of Static Stitched Images

This part mainly shows the details of the stitched images, especially in the seam and intersected areas. Figure 6.6 mainly shows the details on the optimal seam and Figure 6.7 compares the details on the seam of directly overlaid results generated by different methods.

Figure 6.6 shows the optimal seam location based on the intersection area between one of the pairs in Figure 6.2. Some windows along the seam are chosen to zoom in for showing the details of the optimal seams. From Figures 6.6 (a), (b), and (c), we can see that the optimal seam would maintain the roof as integrated as possible and also tries to maintain the integration of features such as solar panels. However, wall objects such as windows can still be broken due to perspective differences. This situation also occurs for the nadir image Stitching with optimal seam. Figures 6.6 (d), (e), and (f) show how the seam is placed on the ground. It can avoid going through big objects like buildings but still cannot completely avoid cutting some small obstacles like cars.

Figure 6.6.: Optimal seam (marked in red line) for the intersection area and Zoom-in views of the area inside the yellow frame.

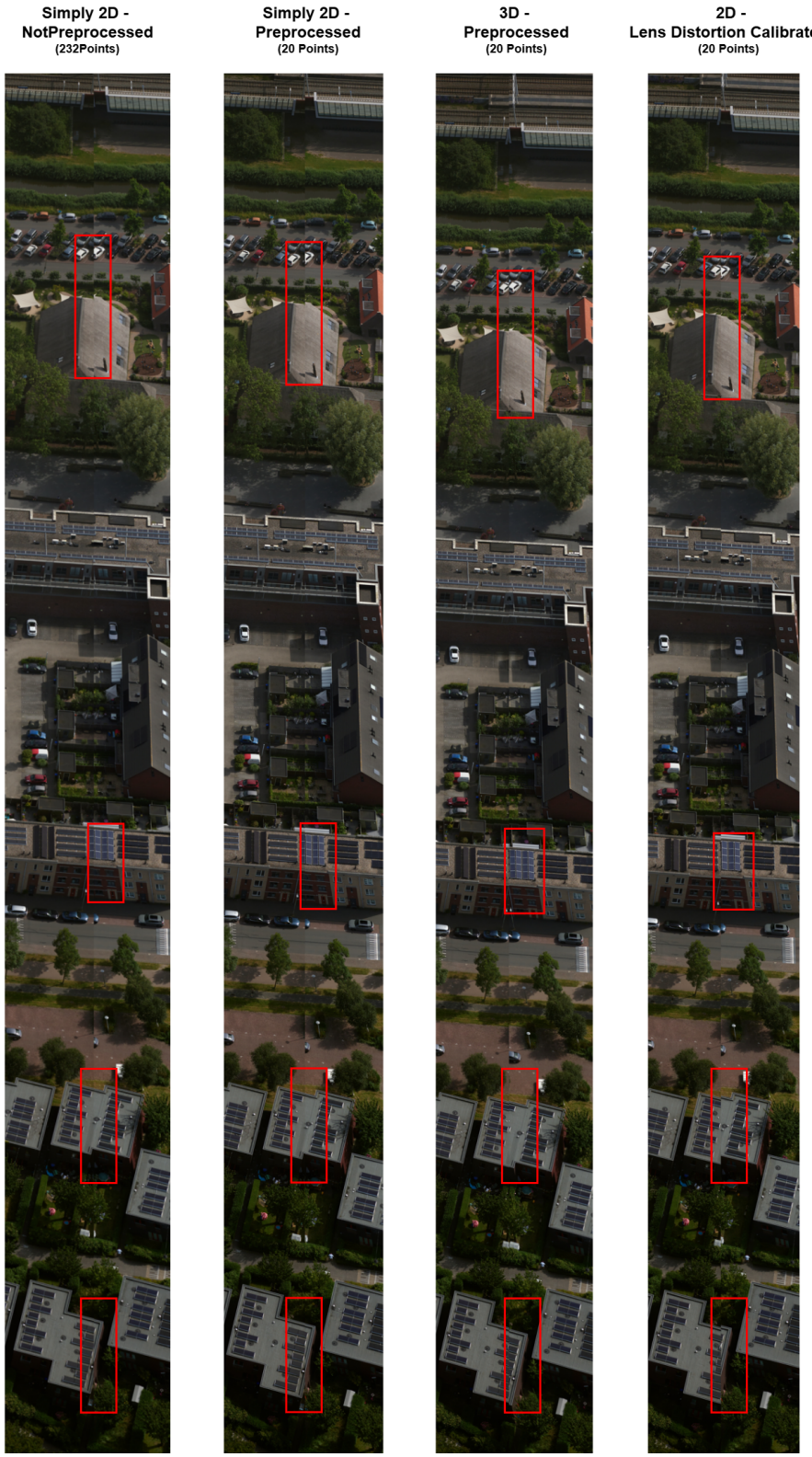

Figure 6.7.: Comparison between the directly overlayed results generated by different preprocessing and stitching methods

Figure 6.7 presents the results of the directly overlaid stitching using different preprocessing or Stitching methods. For this group of images, the Tie-points are almost evenly distributed on the images. Thus, the results generated by the clustering-selected algorithm are almost the same as those without preprocessing. The third image on the left uses the 3D method to stitch, which slightly decreases the duplicate items on the result. The last image shows the Stitching results after the lens distortion calibrate of both images, which also decreases the duplicate items but sometimes too much and leads to a small loss of information. For example, a column of the solar panel disappears in the last image. The quantitative comparison between those Warping methods is shown in table 6.4.

6.1.3. Comparison to Image Stitching Software

The comparison between image Stitching results generated by this thesis project and existing image Stitching software mentioned in Chapter 3 is shown here. For the Stitching software, we only give the results by those with free use license including Hugin, AutoStitch, and demo version of PTGui. Table 6.2 shows a detailed comparison between different technical aspects and Figure 6.8 shows the visual overview of the results.The geometric distortion present in AutoStitch and Hugin outputs causes straight lines to bend unnaturally, especially near the image borders. In contrast, the result produced by this thesis preserves rectilinear structures and overall proportions more effectively, owing to the use of global homography estimation and feature match filtering. Additionally, seam visibility is significantly reduced in our results through dynamic seam optimization rather than Blending to avoid ghosting artifacts that are evident in the overlapping areas of Hugin and PTGui outputs.

Table 6.2.: Comparison of Stitching Methods in Terms of Key Technical Aspects

Aspect	**AutoStitch**	**PTGui**	**Hugin**	**This Thesis**
Feature Matching	SIFT	SIFT / Manual Control Points	SIFT / Manual Control Points	DISK + LightGlue
Warp Type	Homography Warping	Dual homography or Custom Projection Warping	Dual Homography or Non-Linear Warping	Homography Warping
Global Consistency	Low	High	Moderate	Very High
Geometric Distortion	High	Low to Moderate	Moderate	Low
Adjustable Parameters	None	Extensive and GUI-based for control points, projection models, blending options	Extensive with scripting and GUI like Hugui	Fully adjustable at code level
Exportable Outputs	None	None	Project file (.pto) and control point	Any intermediate result if needed
Seam Optimization and Blending	Multi-band blending over overlap regions	Multi-band blending and user-controllable seams	Multi-band blending and optional seam mask specification	No blending but seam finder based on dynamic programming
Runtime	Very snow	Fast	Moderate	Very Fast

AutoStitch

PTGui

Hugin

Result of the Thesis

Figure 6.8.: Visual Comparison of the stitching results using different pipelines

6.2. Geometric Accuracy

6.2.1. Difference between Warped Points and Corresponding Points

The differences between the warped Feature points and their corresponding points are a natural indicator of the geometric accuracy of the image Stitching result. The smaller the difference, the better the image Stitching accuracy is. Figure 6.9 shows the spatial position of the Feature points in the images to stitch and Figure 6.10 shows the change in position of PointsA to stitch to the target points (PointsB). As shown in Figure 6.10, the target points (PointsB) and the warped points almost overlap each other, which means that the homography transformation has high accuracy. The RMSE between the positions of those points is 15.82 (pixel).

Figure 6.9.: Detected Feature Points on the images for stitching

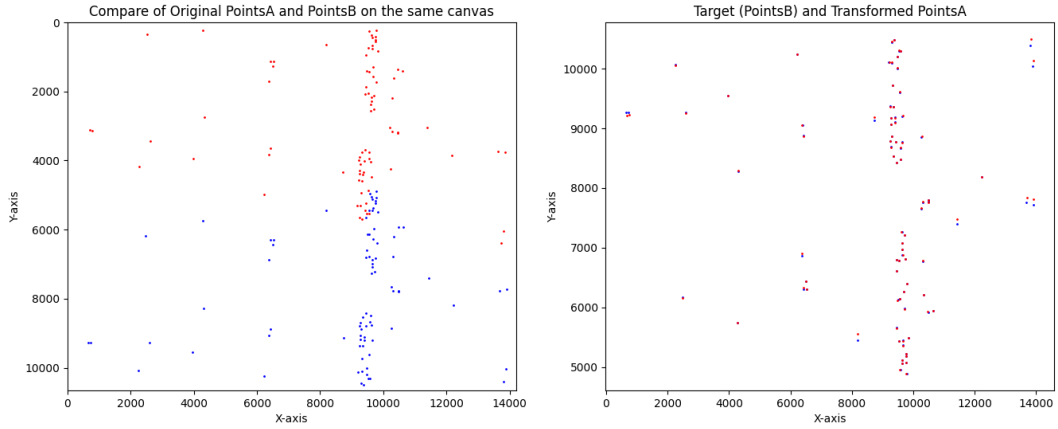

Figure 6.10.: Original and warped points on the same canva with target points

To find the robust condition for image Stitching, the RMSE test is performed for more pairs of

images. 2182 image pairs that meet the basic Stitching conditions in Chapter 5 are selected from the Phoxy database to warp and compare RMSE. Mapping the intersection ration with the matched feature numbers as in Figure 5.2, it presents an approximate positive correlation. Also, because the images are taken with a certain gap, the intersection rate of the images is distributed. The shape of the Gaussian mixture clustering can well describe this kind of distribution (Scikit-learn [2024]). Thus, the image pairs are classified by these two factors,the intersection rate and the number of matching points. Outliers labeled "O" and colored pink are mostly image pairs with less than 20 matching points, which are not suitable for Stitching. And "N" represents the pairs that cannot be classified into any types. From cluster "1" to "6", the intersection rate and the number of feature matching are getting higher. Figure 6.12 shows the average warped RMSE for each cluster. The RMSE shows an obvious decrease between clusters 2 and 3 and tends to be at a stable low value in clusters 4, 5, and 6. Considering that the size of the image is 10652 × 14204, the RMSE error of cluster 3 is around 100 pixels, which is still acceptable. The cluster center for cluster 3 is 60.57% intersection rates and 187.89 tie points, which can be considered approximately the robust Stitching condition for this data set. The warping error for image pairs shown in Section 6.1 is shown in table 6.3. Although scenario 2 has a very low intersection rate, it still has a relatively low warping error.

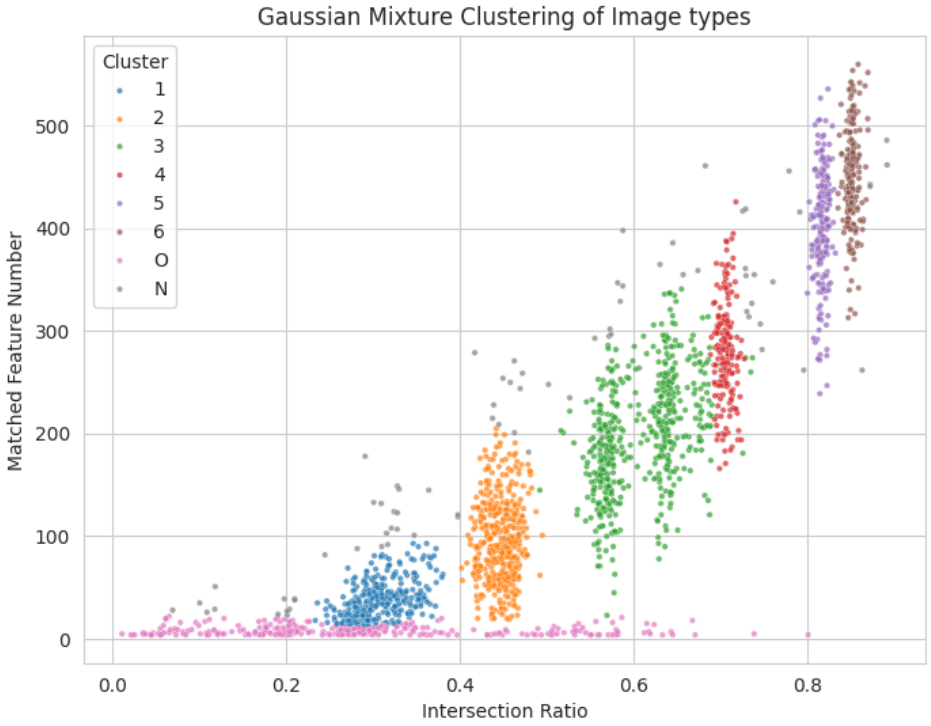

Figure 6.11.: Classification of Images based on Intersection-Tiepoints Number Clustering

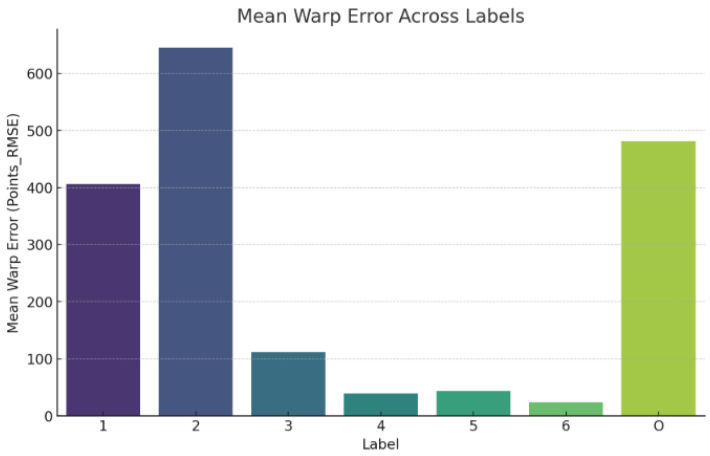

Figure 6.12.: The Average Warping RMSE in Every Image Pair Clustering

Stitching Scenario	RMSE (pixel)	Intersection Rate (%)
1	15.82	62.3
2	63.32	19.1
3	26.28	50.9
4	86.91	33.3

Table 6.3.: Stitching Scenarios in 6.1 and Their Warping Errors

Table 6.3 shows the warped errors for the same pair of images using different Warping methods. The visualized comparison result is also shown in Figure 6.7. From table 6.3, the 3D Warping method has the lowest RMSE, which means the best alignment accuracy. However, it is only 1.32 pixels lower than the normal 2D Warping method. The 2D Warping with unselected points has the highest RMSE, while the lens distortion calibrated somehow makes the RMSE increase a bit.

Methods	RMSE (pixel)
2D-Preprocessed	15.82
2D-NotPreprocessed	37.51
3D-Preprocessed	14.50
2D-Lens Distortion Calibrated	23.32

Table 6.4.: Stitching Methods in Figure 6.7 and Their Warping Errors

6.2.2. Difference between the Reprojected Points and Original Points

The difference between the reprojected points and the original points is calculated by applying the inverse Homography matrix to any pixel in the warped image to get its image coordinates on the original image. In theory, the difference between the reprojected points

and the original points should be zero if the inverse Homography matrix is correctly calculated. However, due to the precision caused by floating point numbers (Microsoft Learn [2024]), it can still have errors between 1 and 2 pixels.

Table 6.5 shows some reprojection results from the manually selected points on the warped image (Transformed_x and Transformed_y) to the original image (Original_x and Original_y). Since the pixel coordinates have to be integer, the rounded coordinates are also compared. When comparing the object type of the points, the points on the flowerbed have slightly higher reprojection errors than other types of objects, which is mainly due to the difficulty of clicking on the exact point on the flowerbed.

Transformed_x	Transformed_y	Original_x	Original_y	Type	Reversed_x	Reversed_y	Round_x	Round_y	Distance	Distance_Round
1003	6930	1037	902	RoofCorner	1036.92	902.11	1037	902	0.13	0.00
7661	7004	7657	1999	Solar Panel	7656.89	1998.87	7657	1999	0.17	0.00
12487	6142	12449	1886	RoofCorner	12449.01	1885.74	12449	1886	0.26	0.00
571	16422	635	10167	RoofCorner	634.76	10166.84	635	10167	0.29	0.00
6088	12636	6078	7303	RoofCorner	6077.68	7303.28	6078	7303	0.43	0.00
6037	13306	6025	7951	RoofCorner	6025.31	7950.74	6025	7951	0.41	0.00
5946	14132	5933	8743	Window	5933.14	8743.43	5933	8743	0.46	0.00
7473	6985	7470	1952	Solar Panel	7470.39	1951.21	7470	1951	0.88	1.00
7424	7071	7420	2029	Solar Panel	7421.36	2028.85	7421	2029	1.36	1.00
7612	7092	7608	2079	Solar Panel	7607.84	2078.48	7608	2078	0.54	1.00
6516	12604	6501	7337	RoofCorner	6500.23	7336.63	6500	7337	0.85	1.00
5961	13936	5948	8554	Window	5948.50	8554.47	5949	8554	0.69	1.00
5864	14131	5852	8729	Window	5852.33	8730.11	5852	8730	1.17	1.00
12195	11955	12102	7560	RoofCorner	12102.77	7559.35	12103	7559	1.00	1.41
6468	13267	6450	7977	RoofCorner	6450.55	7977.63	6451	7978	0.84	1.41
5877	13939	5867	8546	Window	5865.69	8544.75	5866	8545	1.81	1.41
8448	13855	8398	8849	Flowerbed	8398.95	8849.65	8399	8850	1.15	1.41
9184	13769	9123	8877	Flowerbed	9123.98	8876.48	9124	8876	1.11	1.41
9166	14044	9104	9141	Flowerbed	9104.50	9141.76	9105	9142	0.92	1.41
8432	14133	8380	9119	Flowerbed	8381.64	9118.19	8382	9118	1.82	2.24

Table 6.5.: Reprojected Corresponding Points

6.3. Dynamic Stitching Run-time Performance

The dynamic Stitching performance is assessed by placing timers inside the demo code to record the time cost at each step of image Stitching. Table 6.6 records the average running time of 100 rounds in each stage of Stitching processing before and after optimization. As summarized in Table 6.6, the execution time dropped from 8.63 seconds to 0.59 seconds. This confirms that the seam cost computation step benefits substantially from parallelism due to its inherently independent row-wise operations. Taking a look at the run time of each stage, the image loads and resizes have the best performance in optimization with parallel processing, the Warping and optimal seam finding are only sightly improved. Also, the precalculation of the Homography matrix saves around 0.5 seconds in the Stitching process, which also significantly improves the run-time performance.

Stage	Original	Optimized
Create Canva	16.00	15.00
DB Read Center Image Information	0.00	0.00
Image Load and Resize	6673.75	74.25
DB Find Neighbors	0.00	0.00
DB Read Image Pairs	115.50	106.75
Homography Calculation	554.75	0.00
Homography Scale and Offset	1.25	0.00
Image Warping	12.00	12.00
Optimal Seam Finding	802.25	382.00
Image Display	471.00	8.50
Total Time	8630.5	585.50

Table 6.6.: Comparison of Average Running Times (Units: ms)

7. Summary and Discussion

7.1. Summary and Contributions

This thesis aimed to design and implement a pipeline to dynamically Stitching images to provide a continuous user experience in aerial oblique image view. This part includes answers to the research questions proposed in Chapter 1, a brief summary, and future perspectives of this thesis project.

7.1.1. Answers to Research Questions

The main question of this thesis project and the answer to it are as follows:

How can seamless oblique image mosaics be created dynamically from aerial photographs to enhance continuous visualization and minimize measurement errors?

The dynamical seamless oblique image mosaics are created mainly by using corresponding Tie-point pairs to calculate the Homography matrix and applying global transformation to the target images to stitch. This process is optimized by precalculation and storage of Tie-point pairs and homography. The explicit global Homography matrix can be used to precisely revert the original image before Stitching, which minimizes the measurement errors.

1. **To what extent can the Stitching achieve a continuous transition between oblique aerial images? And how is continuity measured?**

   Currently, this research achieves a quick hard transition between images on the same flight controlled by mouse click in east-west direction. When a mouse click to the selected direction, an immediate transition to the next central image and its two stitched images will be done within 1 second. For smooth transition, some attempts like the interpolation have been done, but have not really been applied to the final demo. The continuity is mainly measured by the runtime of the integrated dynamic Stitching demo.

2. **How does the area of the intersection or the number of detected Tie-points between two images influence the Stitching quality?**

   Since the Homography matrix is mainly calculated by the Tie-points, the direct influence factor is the number and distribution of the Tie-points. However, the larger the intersection area, the higher the possibility that more Tie-points can be detected. According to the results in Chapter 6, the intersection rate of 60% is a reference value for reliable Stitching.

3. **How does the seam in the stitched images influence the visualization effect?**

   Seam is an unavoidable issue in the image Stitching results of two images with camera transformation and depth variation. Choosing a seam which crosses the urban structure like buildings as much as possible not only decreases the parallex caused by depth variation of the image but also keeps the integrate structures of user-interested objects like buildings.

4. **What is the robust method for oblique aerial image Stitching when the intersection area is small?**

   Like the answer in Question 2, the intersection area is not the true reason for bad Stitching results. If there are enough precise and well-distributed Tie-points, even the oblique aerial image with low intersection can still be nicely stitched by Homography matrix.

5. **How does the Stitching process impact the usability of images for measurement tasks?**

   Since the Stitching process has explicit expression of a Homography matrix, it only adds one more step for the measurement. Calculating the inverse matrix of the Homography matrix does not need much computation time and space and only causes 1-2 pixel errors because of the precision loss caused by float-number operations. Thus, the Stitching process has little impact on the usability of images for measurement tasks.

6. **How is Stitching quality assessed from both visualization and measurement aspects?** The quality of the Stitching is mainly assessed by visual presentation of static stitched images for visualization aspect. And for measurement, the reprojected geometric accuracy shown in Chapter 6 is mainly for assessment.

7.1.2. Conclusions

In this thesis, an effective and systematic approach to dynamical image Stitching for visualization in a C# environment was successfully developed and validated. The key innovation lies in applying robust homography transformations for accurate image alignment, combined with an implementation of optimal seam finding to enhance the visual quality of stitched images.

The proposed method demonstrated high precision and explicit math expression compared to existing Stitching software. The exportable Homography matrix makes it possible to track the original image coordinates and measure on the stitched images. The experimental results clearly indicated that the developed method provides high-quality Stitching outcomes and a fast run-time, which is suitable for applications in the visualization and measurement of large-scale aerial images.

In general, this research advances the field of geomatics and computer vision by providing practical solutions and insights into interactive image mosaicking and visualization, setting the foundation for further innovations and technological advancements.

7.2. Limitations

The demo and results of this thesis are still prototypes for dynamical image Stitching. Several limitations are still existing.

First, for the Stitching results itself, the single homography does not expand well to multiple image Stitching scenarios, like Stitching all the images within a small range. The current method can only stitch the two to three images on the same row or column well, which can only provide a relatively small field of view.

Second, for a smooth transition between images, although some experiments have been performed to realize the interpolation between the warped and original images, the interpolations for multiple images are still too complex to achieve during the thesis progress. The current transition implementation is still directly switch to the next image in the center, which is hard and discontinuous.

Third, there are still spaces for computational efficiency to be optimized using techniques like cache and Graphics Processing Unit (GPU). The current implementation almost makes full use of Central Processing Unit (CPU) and bottlenecks when processing high-resolution images, impacting real-time responsiveness. However, there is sometimes still memory waste because some images are repeated loaded due to the lack of efficient use of caches.

Other essential issues that have not been explored, such as scalability and integration with the Omnibase measurement system, are also limitations of this research.

7.3. Future Work and Suggestions

Building on the current implementation and acknowledging the limitations outlined in this thesis, several suggested directions for future work are proposed to improve the scalability, performance, and usability of the system in real-world applications.

7.3.1. Advanced Multi-Image Stitching

Future work should focus on extending the current method beyond Stitching two or three images in a single row or column. This would involve developing more flexible strategies for handling multidirectional mosaicking (e.g., grid-based arrangements) and improving the robustness of homography chaining to support more complex spatial relationships and larger-scale datasets. Figure 7.1 shows an example of multi-image mosaic Stitching.

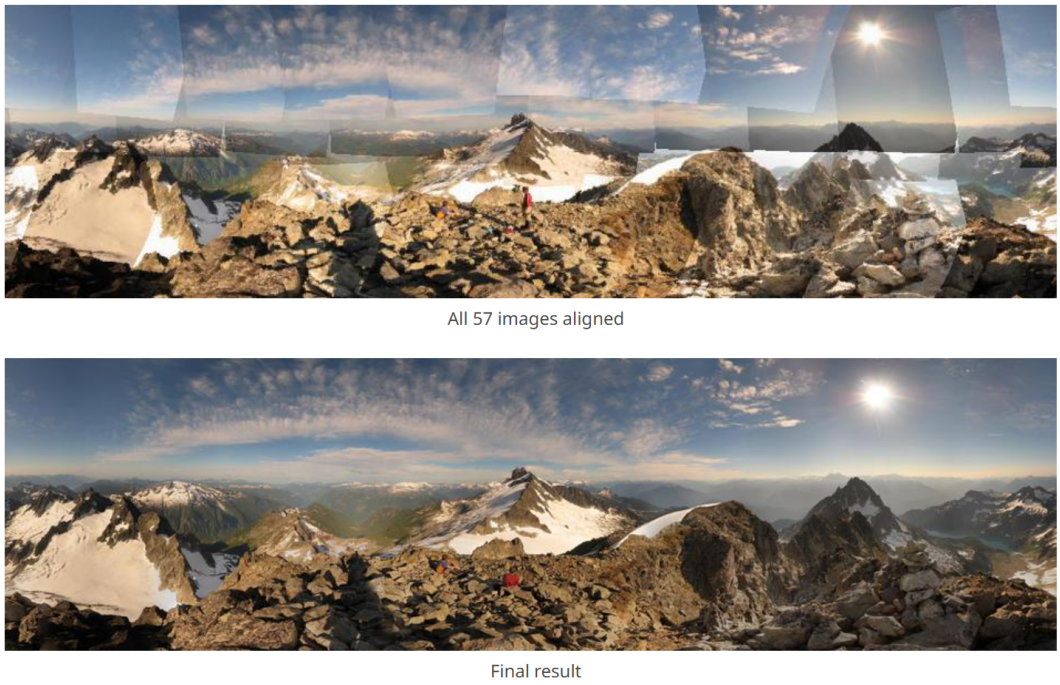

Figure 7.1.: Multi-image Stitching Example, Adopted from AutoStitch Website (Brown [2025])

7.3.2. Smooth Transition and Scrolling

The current image transition mechanism uses abrupt switching centered on the viewport. A smoother and more perceptually pleasing experience could be achieved by implementing advanced interpolation techniques across multiple images. Figure 7.2 shows the structure of the image frame of the hard and soft transition. Interpolation is needed between every hard transition of different center images.

An interactive scrolling application could be developed to replace the static or hard-switched transitions currently in use. Such a system would allow users to pan continuously across a virtual canvas composed of stitched images, enhancing the usability for large-scale aerial data.

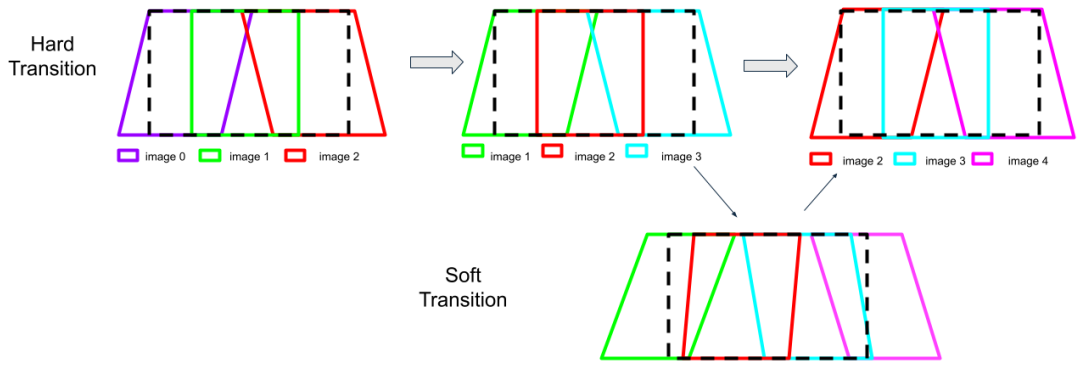

Figure 7.2.: Image Frame Structure of Hard and Soft Transition

7.3.3. Integration with Omnibase

As mentioned in Chapter 1, integration with Omnibase would make the system practically applicable to real-world surveying and inspection tasks. Future work should establish a two-way communication interface for seamless data exchange and enable users to take accurate measurements directly on the stitched images with the coordinate fidelity ensured by Homography matrix.

7.3.4. Runtime Optimization

To improve performance, future work should integrate GPU-based processing and implement efficient caching mechanisms. This would reduce memory waste and increase the responsiveness of the system when dealing with high-resolution images.

A. Reproducibility self-assessment

A.1. Marks for each of the criteria

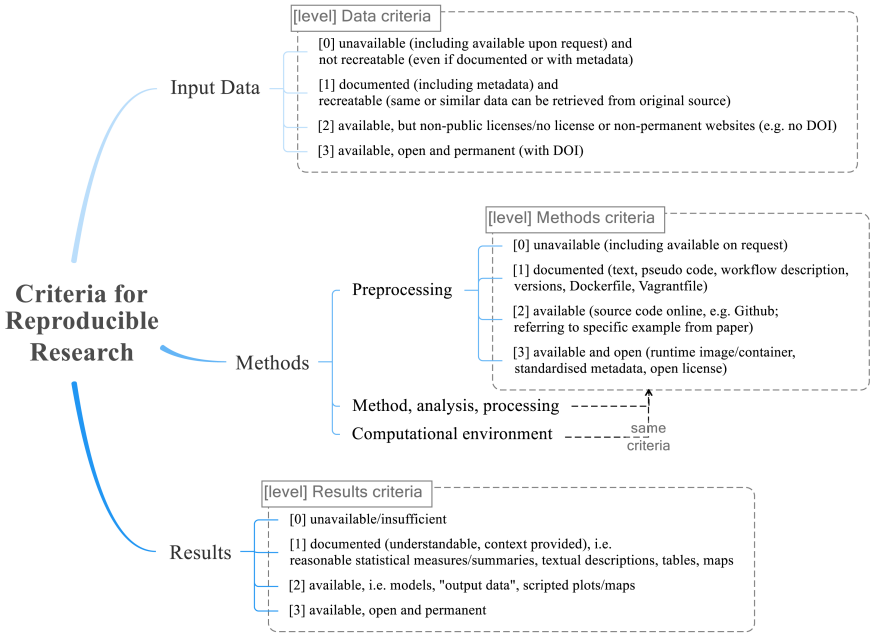

Figure A.1.: Reproducibility criteria to be assessed.

category	criteria	grade
1. Input data	Oblique aerial imagery	1
	Tie-point pairs	1
2. Methods	Pre-processing	1
	Image Stitching processing	1
	Analysis	1
	Computational environment	3
3. Results		1

Table A.1.: Self-reflection of reproducibility criteria in each part of the research according to Figure A.1

A.2. Self-reflection on reproducibility

As indicated in the table A.1, the reproducibility of different parts in this thesis program varies. Generally speaking, this thesis is done as the form of thesis internship at Geodelta B.V., which named the topic and gave the original data and many supports. Thus, reproducibility is limited from several aspects to protect the interests of the company.

- **Input data:** The oblique aerial imagery dataset is offered by Geodelta. To obtain these images, Geodelta has paid for the survey company, Kavel 10, which executes the flight missions to take the images. Thus, these oblique aerial images are not open data. Table 4.1 provides some of the essential properties and meta-data of these images.

  The tie-point pairs are generated by Geodelta's internal algorithm based on the DISK-LightGlue. Since these two algorithms are published in academic papers (Tyszkiewicz et al. [2020]; Lindenberger et al. [2023]), the tie-points data set is partly reproducible.

- **Methods:** All of the implementation of this research is described in detail in this thesis, and the pseudocode is available in Chapter 5. As agreed at the beginning of this thesis program, providing pseudo-code instead of the source code is enough. Thus, no code is available online.

  The implementation of the method relies mainly on the open source computer vision library OpenCV, which is available in different programming languages, including Python and C#. So, the computational environment has a high score in reproducibility.

- **Results:** The dynamic demos still highly rely on Geodelta's image data and database. However, the static version of the stitched images is shown in Chapter 6 and can be reproduced.

B. Reflection

This part includes reflection of this thesis research in three aspects.

- **The relationship between the methodical line of approach of the Master Geomatics and the method chosen by the student in this framework.**

  The MSc Geomatics programme provided a strong methodological foundation that directly informed the thesis. Core courses such as Sensing Technologies and Photogrammetry provide essential knowledge for this research. The integration of practical programming skills from Python Programming for Geomatics and modelling insights from 3D Modelling of the Built Environment supported the technical development of the drag-interpolation and seam-finding algorithms.

- **The relationship between the conducted research and application of the field geomatics**

  The project contributes to geomatics by enhancing oblique aerial image navigation and measurement. It addresses practical challenges in visualization platforms like Omnibase, offering improved stitching, dynamic warping, and reproducible seams. The solution supports urban analysis, infrastructure inspection, and interactive mapping, key application areas in the geomatics domain. The use of real-world datasets and the collaboration with Geodelta reinforced its practical relevance. By transforming academic techniques into an operational prototype, the research illustrates how MSc Geomatics methodologies can be applied to improve real-world engineering tools.

- **The relationship between the project and the wider social context.**

  This thesis supports the societal needs by making aerial imagery more accessible, measurable, and interactive. In applications like urban planning, infrastructure monitoring, and emergency response, better image continuity and transparency improve decision making. The project's focus on dynamic visual transition enhances the usability of oblique imagery for both experts and public stakeholders.

Bibliography

3D Geoinformation, TU Delft (2025). Reconstructing 3d geometry. https://3d.bk.tudelft.nl/courses/geo1016/handouts/04-reconstruct_3D_geometry.pdf. Lecture handout for GEO1016: Introduction to 3D Modelling.

Adobe Inc. (2025). Photo stitching - adobe creative cloud. Accessed: 2025-03-25.

Ai, Y. and Kan, J. (2020). Image mosaicing based on improved optimal seam-cutting. *IEEE Access*, 8:181526–181533.

Armstrong, D., Charlesworth, E., Alderson, A. J., and Elliott, D. B. (2016). Is there a link between dizziness and vision? a systematic review. *Ophthalmic and Physiological Optics*, 36(4):477–486.

Azzari, P., Di Stefano, L., and Mattoccia, S. (2008). An evaluation methodology for image mosaicing algorithms. In *Advanced Concepts for Intelligent Vision Systems: 10th International Conference, ACIVS 2008, Juan-les-Pins, France, October 20-24, 2008. Proceedings 10*, pages 89–100. Springer.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.

Beier, T. and Neely, S. (1992). Feature-based image metamorphosis. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 35–42, New York, NY, USA. ACM.

Bernstein, D. S. (2009). *Matrix mathematics: theory, facts, and formulas*. Princeton university press.

Bing Maps Team (2022). Get bird's eye views in your next great maps app. https://blogs.bing.com/maps/2022-07/Get-Bird-s-Eye-Views-in-your-next-great-maps-app. Accessed: 2025-01-02.

Brown, M. (2025). Autostitch - image stitching demo. Accessed: 2025-03-25.

Brown, M. and Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74:59–73.

Burt, P. J. and Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)*, 2(4):217–236.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 778–792. Springer.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Choi, K. H. and Kim, C. (2021). Proposed new av-type test-bed for accurate and reliable fish-eye lens camera self-calibration. *Sensors*, 21(8):2776.

Dinic, E. A. (1970). Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280.

Efros, A. A. (2016). Image morphing. https://people.eecs.berkeley.edu/~job/Classes/CS184/Fall-2016-Slides/efros-morphing-cs184.pdf. Lecture slides, accessed: 2025-01-02.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fu, M., Liang, H., Zhu, C., Dong, Z., Sun, R., Yue, Y., and Yang, Y. (2023). Image stitching techniques applied to plane or 3-d models: a review. *IEEE Sensors Journal*, 23(8):8060–8079.

Gao, J., Kim, S. J., and Brown, M. S. (2011). Constructing image panoramas using dual-homography warping. In *CVPR 2011*, pages 49–56. IEEE.

Geodelta (2024). OmniBase Software. https://geodelta.com/software/omnibase. Accessed: 2024-11-27.

Glasbey, C. A. and Mardia, K. V. (1998). A review of image-warping methods. *Journal of applied statistics*, 25(2):155–171.

Greig, D. M., Porteous, B. T., and Seheult, A. H. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 51(2):271–279.

Höhle, J. (2008). Photogrammetric measurements in oblique aerial images. *Photogrammetrie, Fernerkundung, Geoinformation*, (1):7–14.

Höhle, J. (2013). Oblique aerial images and their use in cultural heritage documentation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40:349–354.

Hugin Project (2025). Hugin - panorama photo stitcher. Accessed: 2025-03-25.

Laraqui, A., Baataoui, A., Saaidi, A., Jarrar, A., Masrar, M., and Satori, K. (2017). Image mosaicing using voronoi diagram. *Multimedia Tools and Applications*, 76:8803–8829.

Levin, A., Zomet, A., Peleg, S., and Weiss, Y. (2004). Seamless image stitching in the gradient domain. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV 8*, pages 377–389. Springer.

Li, N., Xu, Y., and Wang, C. (2017). Quasi-homography warps in image stitching. *IEEE transactions on multimedia*, 20(6):1365–1375.

Lin, C.-C., Pankanti, S. U., Natesan Ramamurthy, K., and Aravkin, A. Y. (2015). Adaptive as-natural-as-possible image stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1163.

Lindenberger, P., Sarlin, P.-E., and Pollefeys, M. (2023). Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110.

Microsoft Learn (2024). Why floating-point numbers may lose precision. Accessed: 2025-05-12.

New House Internet Services B.V. (2025). Ptgui - panorama photo stitching software. Accessed: 2025-03-25.

OpenCV (2024). Feature matching + homography to find a known object. https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html. Accessed: 2025-04-30.

OpenCV (2025). OpenCV. Accessed: 2025-05-07.

OpenCV, D. (2024). Geometric transformations of images. https://docs.opencv.org/3.4/da/d6e/tutorial_py_geometric_transformations.html. Accessed: 2025-04-04.

Pham, N. T., Park, S., and Park, C.-S. (2021). Fast and efficient method for large-scale aerial image stitching. *IEEE Access*, 9:127852–127865.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947.

Scikit-learn (2024). *Gaussian mixture models — scikit-learn 1.4.2 documentation*. Accessed: 2025-05-13.

Sobel, I. (2014). History and definition of the sobel operator. *Retrieved from the World Wide Web*, 1505.

SQLite Consortium (2025). Sqlite - embedded sql database engine. https://www.sqlite.org/. Accessed: 2025-05-01.

Steedly, D., Szeliski, R., Uyttendaele, M., and Cohen, M. (2010). Oblique image stitching. United States Patent.

Stewart, C. V. (1999). Robust parameter estimation in computer vision. *SIAM review*, 41(3):513–537.

Szeliski, R. et al. (2007). Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104.

Tang, M., Mei, X., Li, Y., Chen, C., Liu, X., and Lu, H. (2024). Application of oblique photogrammetry technique in geological hazard identification and decision management. *Earthquake Research Advances*, 4(3):100269.

Tyszkiewicz, M., Fua, P., and Trulls, E. (2020). Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33:14254–14265.

U.S. Geological Survey (2018). Oblique aerial photography viewer. https://www.usgs.gov/tools/oblique-aerial-photography-viewer. Accessed: 2025-01-02.

Verykokou, S. and Ioannidis, C. (2024). Oblique aerial images: Geometric principles, relationships and definitions. *Encyclopedia*, 4(1):234–255.

Wang, Z. and Yang, Z. (2020). Review on image-stitching techniques. *Multimedia Systems*, 26(4):413–430.

Wolberg, G. (1990). Digital image warping.

Zaragoza, J., Chin, T.-J., Brown, M. S., and Suter, D. (2013). As-projective-as-possible image stitching with moving dlt. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2339–2346.

Zeng, L., Zhang, S., Zhang, J., and Zhang, Y. (2014). Dynamic image mosaic via sift and dynamic programming. *Machine vision and applications*, 25:1271–1282.

Zhang, Z., Cheng, L., Feng, X., Wang, X., Zhai, G., Tang, K., and Sun, S. (2023). Application of oblique photography technology in airport site selection. In *Civil Engineering and Disaster Prevention*, pages 413–419. CRC Press.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.