Tobias Ollmann, BSc

Panorama Stitching in Planetary Environments

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to
Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr. Horst Bischof
ICG - Institute for Computer Graphics and Vision

Dipl.-Ing. Arnold Bauer
Dipl.-Ing. Gerhard Paar
JOANNEUM RESEARCH

Graz, December 2016

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz _____ _____

Date/Datum Signature/Unterschrift

Acknowledgements

Writing a master's thesis cannot be done completely alone. Many people helped and supported me throughout the last year.

First I want to thank Prof. Horst Bischof from the Institute of Computer Graphics and Vision (ICG) at Graz University of Technology. He helped me with useful hints, especially for the writing phase of this thesis.

I am grateful that Gerhard Paar provided me with the opportunity to work at JOANNEUM RESEARCH. It was an interesting, challenging and valuable experience. Thanks to all colleagues for the nice working atmosphere and for answering my countless questions.

Special thanks to Bob Deen from MIPL/JPL/NASA, who helped me orienting with the topic and provided practical hints and ideas.

It is difficult to find words to thank my wife Johanna. With her love and support, she brought a new dimension into my life.

Last but not least I want to thank my parents. They supported me throughout my educational carrier and my whole life.

Abstract

While panorama stitching – the process of combining several images into one – is considered largely solved, the background of planetary environments adds some challenging aspects. One facet is the need of scientifically exploitable panoramas. They provide visual and geometric context for measurement data of other sensors, which makes them the main sense on planetary missions. Some images are taken in different spectral ranges to determine the composition of rocks. These images may also need to be embedded into a wider context for a better understanding.

The process of panorama stitching has to be traceable and documentable like every process in science. Any blending or unconstrained warping could corrupt the result and lead to avoidable uncertainties in interpretation. This scientific integrity of panoramas is paramount for reliable further analysis by geologists.

Another aspect of planetary environments on the other hand are the large data sets of currently up to 1000 images per panorama. In order to reduce the amount of unneeded data which is transmitted, the cameras typically have a very narrow field of view. This allows a high granularity of the mapped area however it increases the required capabilities of the stitching framework.

In this thesis we will present a flexible and scalable pipeline for panorama stitching in planetary environments. The modular design allows individual stages to be modified or completely exchanged without consequent changes on other modules. As the main objective is the flexibility and long-term usability, also for other purposes in the industrial domain, some modules only receive basic functionality. However for the stages directly involved in the stitching process we evaluate state of the art solutions for implementation and extended them.

Keywords: image stitching, planetary environments, panorama weaving, Mars, scientific integrity

Zusammenfassung

Panorama Stitching – das Kombinieren von mehreren Bildern zu einem Panorama – gilt als weitgehend als gelöst. Durch die Anwendung in der Erforschung anderer Planeten ergeben sich allerdings einige neue Anforderungen.

Die wichtigste Eigenschaft der erzeugten Panoramen ist deren wissenschaftliche Integrität. Diese ermöglicht einen fundierten visuellen und geometrischen Kontext für andere Messdaten wie zum Beispiel der Analyse von Gesteinsproben. Für die richtige Interpretation der Messungen ist ein Bezug zur Umgebung wichtig. Auch einzelne Bilder in anderen Lichtspektren können durch Panoramen im Zusammenhang zur Umgebung analysiert werden. Um die wissenschaftliche Integrität zu gewährleisten, muss das Panorama Stitching, wie jeder wissenschaftliche Prozess, nachvollziehbar und dokumentierbar sein. Jegliches Überblenden und lokales Verzerren der Bilder für bessere Übereinstimmung ist ausgeschlossen. Diese Prozesse können vorhandene geometrische Zusammenhänge zerstören und zu Fehlinterpretationen führen.

Ein weiterer Aspekt von planetaren Missionen sind die großen Datensätze von derzeit bis zu 1000 Bildern pro Panorama. Die Kameras haben einen kleinen Bildwinkel, wodurch eine hohe Granularität bezüglich der abgebildeten Szene erreicht wird. Dies reduziert die zu übertragende Datenmenge, erhöht aber die Anforderungen an die Stitching-Software.

In dieser Masterarbeit werden wir ein flexibles und skalierbares System präsentieren, das den Anforderungen in planetaren Missionen gerecht wird. Das modulare Design erlaubt es, einzelne Stufen des Prozesses zu verändern oder ganz auszutauschen ohne nachfolgende Änderungen in anderen Modulen. Eine Hauptanforderung dieser Masterarbeit ist die Flexibilität und langfristige Nutzung der Software. Einzelne Module erhalten daher nur Grundfunktionalität mit der Möglichkeit zur späteren Erweiterung. Für jene Module, die direkt im Stitching involviert sind, werden wir aktuelle Ansätze implementieren, evaluieren und erweitern.

Contents

List of Figures

List of Tables

List of Acronyms

AMASE	Arctic Mars Analog Svalbard Expedition
AUPE	Aberystwyth University Pancam Emulator
ChemCam	Chemistry & Camera
CNSA	China National Space Administration
DBM	depth-based image mosaicing
DSN	Deep Space Network
DTM	Digital Terrain Model
ESA	European Space Agency
GPS	Global Positioning System
HFVM	Hierarchical feature vector matching
ICE	Image Composite Editor
ISRO	Indian Space Research Organisation
JAXA	Japan Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
JR	JOANNEUM RESEARCH
MAHLI	Mars Hand Lens Imager
Mastcam	Mast Camera
MIPL	Multi-mission Instrument Processing Laboratory
MSL	Mars Science Laboratory
NASA	National Aeronautics and Space Administration
NCC	Normalized Cross Correlation

RANSAC Random Sampling Consensus

Roscosmos Roscosmos State Corporation for Space Activities

SAD Sum of Absolute Differences

SfM Structure from Motion

Chapter 1

Introduction

Modern space exploration can not be imagined without the use of cameras. Cameras allow to see worlds where it is too dangerous or (yet) impossible to travel for humans. Even if measurements of other instruments are more significant, cameras provide a visual context for scientists. Geologists who analyse images are used to have a full overview of the surroundings on Earth. Having whole panoramas of sites on Mars or other planets helps them to compare geological features with those on Earth, relying on their intuition.

The goal of this thesis is to evaluate different methods for panorama stitching and to create a flexible and scalable framework for stitching images. The design should be able to easily include future extensions for long-term usability.

Section 1.1 will briefly introduce into the basics of planetary exploration based on current missions to Mars. The link to panorama stitching is included in Section 1.2. This chapter ends with an outline of the rest of this thesis in Section 1.3.

1.1 Planetary Environments

Starting in the sixties and seventies of the last century, space agencies all over the world (China National Space Administration (CNSA), European Space Agency (ESA), Indian Space Research Organisation (ISRO), Japan Aerospace Exploration Agency (JAXA), National Aeronautics and Space Administration (NASA), Roscosmos State Corporation for Space Activities (Roscosmos), SpaceX) started with planetary exploration. After finding the Moon and Venus as not suitable for hosting life, recent missions focus on investigating Mars ([41, 21, 27, 52]). Currently the most promising way for planetary exploration is the use of remote controlled, semi-autonomous rovers.

Scientists can drive around their instruments, search for interesting structures and take and analyse probes of rocks and soil through them. Throughout this thesis we will use NASA's "Curiosity" [39], Mars Science Laboratory (MSL), as a reference for mars rovers and other extraterrestrial robots. Operating since 2012, it is currently the youngest and most sophisticated device on the surface of Mars. The images taken by MSL are made public by NASA and provide a huge set of data to test and validate different panorama stitching algorithms. Along with the image data there is also a lot of meta data linked like position and orientation of the rover and its cameras. This data also includes camera calibration in various camera states. The rover with annotated instruments is shown in Figure 1.1.

On the rover, there is a complete laboratory enabling a subset of the experiments and analyses as geologists would do on Earth. To provide this functionality, it contains a lot of different sensors, including 17 cameras. Most of them are engineering cameras, allowing the operations team to analyse the immediate surroundings, safely navigate on Mars' surface and determine where to perform scientific experiments. In addition to these navigation and hazard avoidance cameras, there are four science cameras on the rover:

Mars Hand Lens Imager (MAHLI) is mounted on the rover's arm and is used for microscopic imaging of rocks as well as self portraits of the rover.

Mast Camera (Mastcam), a high resolution stereo rig, is situated on the rovers mast. The mast has a pan-tilt unit, allowing a 360° view. The two Mastcams have different focal lengths, providing a field of view of 15° (left) and 5.1° (right). With a resolution of 1200x1200 pixels, this yields a spacial resolution of $150\mu m$/pixel at a distance of $2m$ [40] for the right camera. Each camera has several light filters to enable analysis of different spectral ranges from visible light to near infrared.

Chemistry & Camera (ChemCam) is a laser combined with a camera and a spectrograph, used for remote analysis of rocks. Shooting the laser vaporises small parts of the hit rock, creating a radiating plasma. The spectrograph is used to analyse the emitted light to determine the composition of the material. The integrated camera is used to visually analyse the surface of the rock, and to provide a visual context for the spectrograph data.

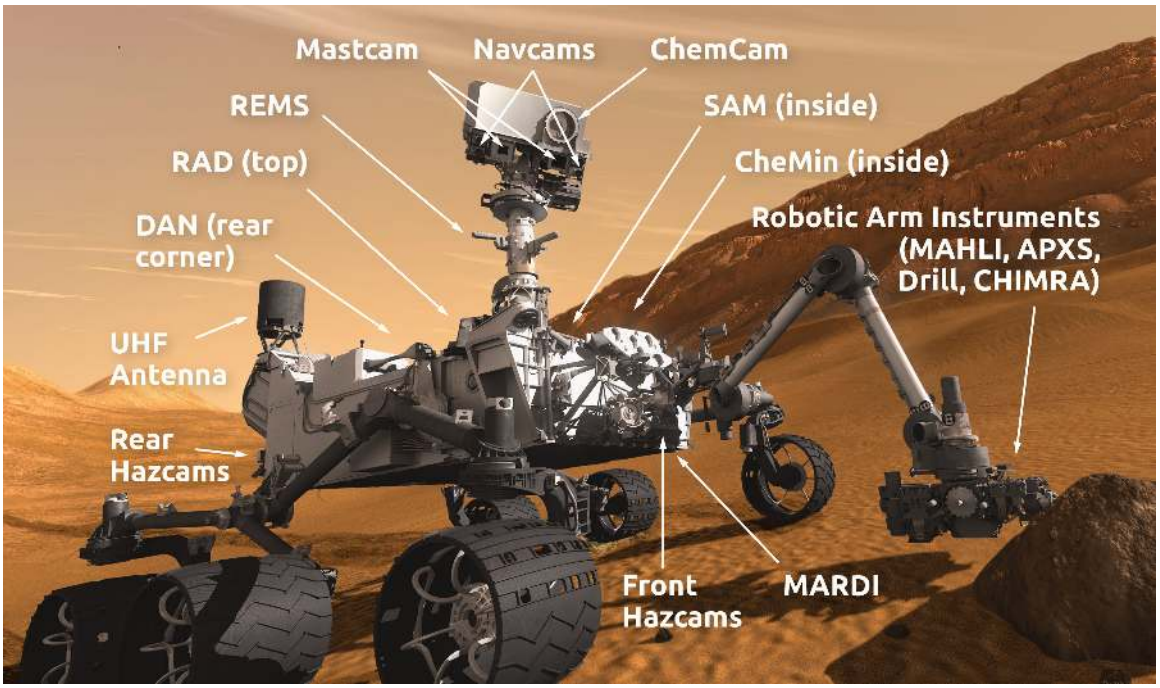


Figure 1.1: MSL Curiosity with instruments annotated Image courtesy by NASA/JPL-Caltech.

On a mission million kilometres away from Earth, every step has to be planned very carefully because recovery possibilities after an incident are very limited. Scientists on Earth have to be able to completely rely on the data as well as their experience as geologists. When they are on field missions on Earth, there is always the possibility for a 360° view of the surroundings. This helps to get an intuitive impression of a site as well as an embedding context to better understand the significance of probes. As mentioned above, Mastcam can help providing this context on Mars.

Depending on the available bandwidth and on the current research topic, not only one image for each Mastcam position (as seen in Figure 1.2) is downloaded, but a stereo pair. In such cases, a pairwise reconstruction is performed, yielding a depth and texture image for each position. In either way, scientist cannot use data like Figure 1.2 directly. For further analysis, the images (RGB, depth or other spectral bands) need to be stitched together.

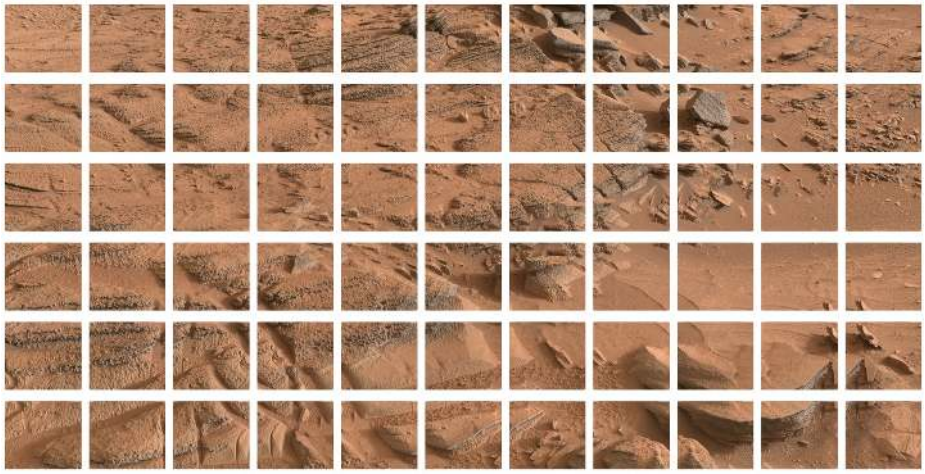


Figure 1.2: Raw images from MSL Rover Image courtesy by Courtesy NASA/JPL-Caltech.

1.2 Panorama Stitching

Panorama stitching is dealing with the task of combining multiple images of the same scene together in one image. Depending on whether the focus is put on the final result or on the source images, some authors use the term *image stitching* for the same task.

The standard case of panorama stitching has been researched and described quite well in multiple papers [55, 58, 46]. Already 10 years ago Richard Szeliski wrote an end-to-end tutorial [55] for combining multiple images into panoramas. It covers image geometry, projection, pixel-based alignment as well as feature-based image registration and different approaches for composing everything into one image. The described process works properly with some requirements on setup and environment.

First and most important assumption is a camera solely rotating around its centre of projection. This concept will be explained later in Section 2.3.1. Any translation of the centre of projection introduces parallax in the image. Parallax is the effect of apparent displacement of foreground objects with respect to the background as shown in Figure 1.3. Depending on the stitching algorithm, the object could appear twice in the result or not at all. To correctly mitigate those errors, knowledge of the 3D structure of the scene is needed.

There are a lot of other factors and constraints to consider for planetary missions, which makes it unfeasible to include such a panorama camera. Instead of one camera rotating around its centre, there is a stereo rig rotating around its common centre. This allows the 3D reconstruction of the surroundings which is more valuable than easing the stitching process.

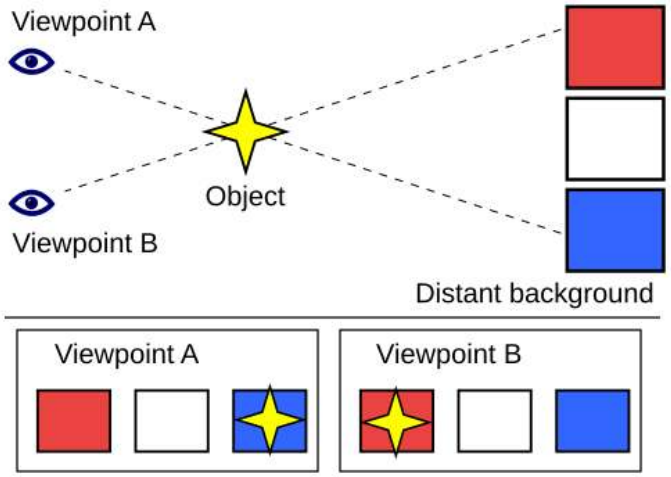


Figure 1.3: Origin of parallax. The images from Viewpoint A and Viewpoint B cannot be simply fused together without introducing unwanted artefacts. Image courtesy by Booyabazooka under CC-BY-SA-3.0 via Wikimedia Commons

A second assumption of most panorama stitching solutions is the absence of motion in the scene during image acquisition. Although the immediate impact on the final panorama is similar, violations of this requirement can be handled better [58]. Any rover movements during image acquisition is known beforehand respectively prevented. The known movement can be handled like parallax errors described above.

Most available panorama stitching frameworks and recent research focus on the visual perception of the final result (e.g. [10, 33]). Existing structures are distorted such that the single images fit better. While this is a reasonable goal in most commercial applications, the planetary environment requires more of a panorama. For further analysis, existing structures and geometric relations must be preserved. It is important to document every step during panorama generation to allow usage in scientific environments. This documentation is called "scientific integrity" in literature [16] and particularly applies in the geometric context. It includes amongst other properties the traceability of each pixel in the final panorama to one position in one source image. A deeper explanation of scientific integrity is given in Section 2.1.

1.3 Outline

After briefly introducing the topic in the previous sections, Chapter 2 will review different approaches in literature. Starting with general planetary data processing in Section 2.1, we will analyse basic mathematics of cameras in Section 2.2. Sections 2.3 to 2.6 contain a summary of panorama stitching in general along with selected subtopics.

We present our framework in Chapter 3, starting with its requirements derived from literature as well as given preconditions from JOANNEUM RESEARCH (JR) in Section 3.1. Section 3.2 contains our proposed design along with a description and some implementation details of the involved modules.

We evaluate our workflow in Chapter 4. The used data sets are shown in Section 4.1. This section also includes reasons why each data set is important for an extensive evaluation. The actual results of our evaluation are presented in Section 4.2. Rounding off evaluation, Section 4.3 contains a comparison with selected state of the art panorama stitching solutions.

Lastly we conclude this thesis in Chapter 5 with a summary as well as an outlook of further work in this topic.

Chapter 2

Related Work

In this chapter we will review the building blocks needed for this thesis. First, we will introduce general planetary data processing schemes according to current literature in Section 2.1. As a basis for the rest of this thesis, we review basic camera geometry in Section 2.2, following the notation of Hartley and Zisserman [25]. The basic image stitching procedure follows in Section 2.3 with a brief overview of current techniques for combining multiple images into one view. Section 2.4 contains the introduction of an alternative stitching approach using manifold projection. Sections 2.5 and 2.6 review selected stages of panorama stitching in more depth. The first handles the calculation or refinement of camera positions and orientations. In the latter we will review different methods of finding a cut line between two or more overlapping images with a special focus on Panorama Weaving in Section 2.6.4.

2.1 Planetary Data Processing

In order to get scientific exploitable panoramas, some specific characteristics and desired properties of planetary data and products have to be considered. On that account there have been workshops recently for handling planetary data [16, 17]. On those workshops Bob Deen, senior software developer at Multi-mission Instrument Processing Laboratory (MIPL) at NASA, defines “scientific integrity” with some basic premises for scientific exploitable panoramas. These principles permit geologists to rely on the final product and derive further knowledge.

Geometric Integrity

One major issue addressed by Bob Deen's work is the geometric integrity of the result. This primarily excludes the usage of unconstrained warping and blending of images in overlapping areas. Both would not be a problem (and not be needed) if the input images are free of parallax and mis-registration. As either occur in general, warping and blending will output unreproducible results.

The above restrictions imply another property of scientific integrity: the "traceability of each pixel to source image" [16]. Every pixel in the panorama should correspond to one position in one source image. This property enables researchers to lookup the source image at a specific location to check the availability of other existing data, e.g. images in different spectral bandwidths.

There are multiple ways to combine images without usage of blending and warping. Bob Deen encourages the use of very simple methods which do not try to hide errors at all. He prefers "a hard-edged, straight seam to something that could be misinterpreted" [17].

Radiometric Integrity

Additionally, Deen proposes to maintain any radiometric correction values. Having this data, it is possible to restore the original pixel value without lookup in the original data, or – as a logical equivalence – allow a 1:1 lookup in the original data.

As blending is not allowed, any radiometric correction must be limited to image-global operations. Typically there is a scale and offset value for each image to account for different illumination conditions. Basis of this correction are colour differences of homologous points in overlapping image areas.

Usage of Existing Pointing Information

In most scientific contexts, especially in planetary environments, there are metadata documenting the image acquisition. Next to time, date and possibly a rover position, this can also include information of turning angle encoders of joints in the camera mount. Such encoders exist on Mars rovers.

In order to provide documented and reproducible results it is important to respect any available pointing information. There exist tools (e.g. AutoStitch [9, 10], Hugin [14]) that solely rely on feature matching to generate relative orientations. On

highly repetitive or empty textures these algorithms can fail. Other systems (e.g. Microsoft Research Image Composition Editor [36]) rely on a structured (e.g. column wise) acquisition of the images.

The existence of camera orientations does not supersede the need to refine it. Due to instable standpoint, hysteresis in joints or too low accuracy of angle encoders, the images can be several pixels off in the final panorama when using the raw angle values.

Precautions have to be taken that the refinement process does not taint the scientific integrity of the panorama. The amount of angle change can e.g. be limited to a certain maximum to stop the optimisation algorithm to drag the whole solution.

2.2 Camera Geometry

The most important subsystem for computer vision applications are cameras. Understanding how cameras map the 3D world onto 2D images is crucial for understanding any algorithm working on images. A widespread model with a huge range of applications is the pinhole camera model, a central projective camera [25]. Notable representatives of other camera systems are fisheye cameras [31] for creating wide angle views. This thesis concentrates on the process of creating images with a wide field of view from multiple images with a narrower field of view. While this is also possible from fisheye images, we will not consider it in this thesis. However, the generality is maintained by the modularity of the approach.

In this section we will review the simple pinhole camera model with its parameters. Following the geometry of a single camera in section 2.2.1, section 2.2.2 contains the mathematical tools to describe two cameras and their relation. While Hartley and Zisserman also describe situations with three and more cameras, we will stick to camera pairs in this thesis. Camera pairs are not necessary for standard panorama stitching, but as we want to be able to stitch depth data, we introduce their mathematical description.

2.2.1 Pinhole Camera Model

The pinhole camera model is the most basic camera model, but still can be applied to the vast bulk of modern cameras. Its name origins from the first primitive pinhole camera, also called camera obscura, as seen in Figure 2.1. The light enters a dark

room or box through a small pinhole. Light travels along straight paths, so every ray reaching the back wall has its origin from a well defined direction. The result is a projection of the outside scene on the back wall.

Modern and generalized variants of this model are called *central projective camera models* [25], as all rays pass through the same point, the projection centre. From the mathematical point of view it makes no difference if the projection is upside down *behind* the projection centre, or upright *in front of* it. For the sake of simpler equations and better intuition, the latter version is used. Figure 2.2 shows the geometric model of central projective cameras.

The algebraic means of calculating the projection from Figure 2.2 is the camera matrix P. A point $\mathbf{X}$ hits the image plane at $(fX/Z, fY/Z)$. Written as homogeneous coordinates (also used by Hartley and Zisserman in [25]) this corresponds to (fX, fY, Z).

The mapping can be written as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} = P \cdot \mathbf{X}. \tag{2.1}$$

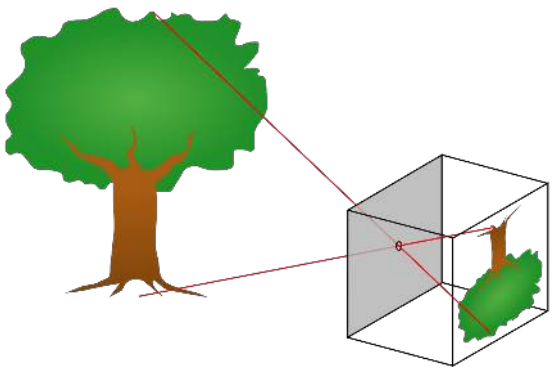


Figure 2.1: Principle of a pinhole camera. Image courtesy by Wikimedia Commons, Public Domain

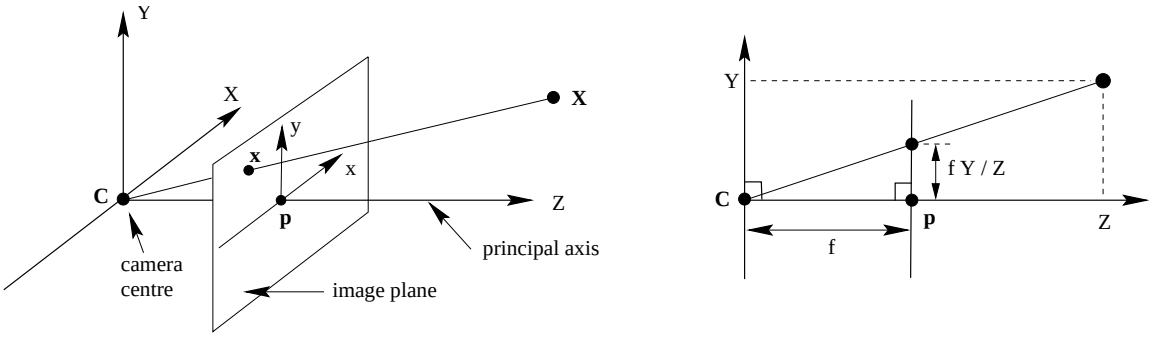


Figure 2.2: Model of a central projective camera. The ray connecting a 3D point **X** and the camera centre **C** is intersected with the image plane. This plane is parallel to the $X - Y$ plane with a distance f from the origin, f being the focal length. Result of the intersection is the image point **x**. The intersection of the principal axis Z with the image plane is called principal point **p**. Image courtesy by Hartley and Zisserman [25]

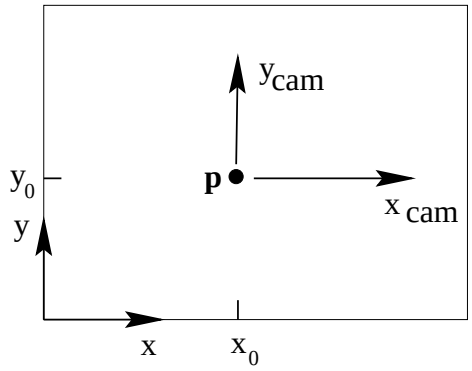


Figure 2.3: p is the principal point of the camera. It is defined as the intersection of the principal axis with the image plane. This image offset can be used to transform between the camera coordinate system (x_{cam}, y_{cam}) and image resp. pixel coordinate system (x, y). The position of p is given as image coordinates (x_0, y_0). Image courtesy by Hartley and Zisserman [25]

As Figure 2.3 shows, the origin of the image plane does not have to be the principal point. To reflect this offset (x_0, y_0) in the projection, the mapping is changed to

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zx_0 \\ fY + Zy_0 \\ Z \end{pmatrix} = \begin{bmatrix} f & & x_0 & 0 \\ & f & y_0 & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.2}$$

P can also be written as

$$P = K[I|\mathbf{0}], \tag{2.3}$$

where

$$K = \begin{bmatrix} f & & x_0 \\ & f & y_0 \\ & & 1 \end{bmatrix} \tag{2.4}$$

is the *camera calibration matrix* [25]. For the sake of generality, K can be extended to model non-square pixels by using

$$K = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{2.5}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$. m_x and m_y represent the respective scaling in x and y direction. Non-square pixels can e.g. occur in some CCD-Cameras.

Additionally, the image axes does not have to be perpendicular. To model a skew in the image, the skew factor s is introduced in Equation 2.6. However, this is 0 for most cameras. The fully generalized camera calibration matrix is

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}. \tag{2.6}$$

In addition to its 5 *intrinsic* parameters in K, a camera can be moved and rotated freely in space relative to some external coordinate system. Suppose the camera is centred at its projection centre C and rotated with respect to the world coordinate

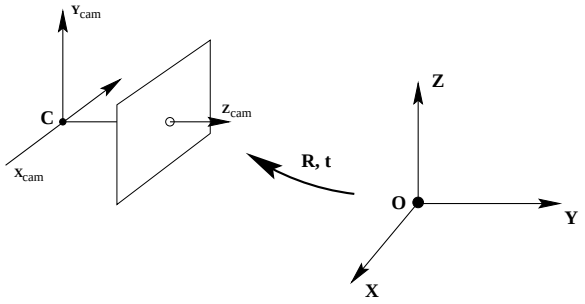


Figure 2.4: Adding rotation and translation parameters $\mathbf{R}$ and $\mathbf{t}$ enables the camera origin to be moved out of the global coordinate system's origin. Any global point X has to be transferred into the camera system as X_{cam}. Image courtesy by Hartley and Zisserman [25]

system by R as seen in Figure 2.4. In euclidean coordinates (marked with a tilde), the relation of a point $\widetilde{\mathbf{X}}$ in world coordinates to its representation in camera coordinates $\widetilde{\mathbf{X}}_{cam}$ is

$$\widetilde{\mathbf{X}}_{cam} = R \cdot (\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}}). \tag{2.7}$$

Equation 2.1 expresses the projection of a homogeneous point in the camera coordinate system. To be able to insert Equation 2.7, it has to be expressed in homogeneous coordinates too:

$$\begin{aligned}
\widetilde{\mathbf{X}}_{cam} &= R \cdot (\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}}) \\
\mathbf{X}_{cam} &= \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -R \cdot \widetilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{X} \\
\mathbf{X}_{cam} &= \begin{bmatrix} R & -R \cdot \widetilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{X}
\end{aligned} \tag{2.8}$$

Inserting Equation 2.8 into

$$\mathbf{x} = K[I|\mathbf{0}]\mathbf{X}_{cam} \tag{2.9}$$

yields

$$\begin{aligned}
\mathbf{x} &= K[R| - R \cdot \widetilde{\mathbf{C}}] \cdot \mathbf{X} \\
\mathbf{x} &= K[R|\mathbf{t}] \cdot \mathbf{X},
\end{aligned} \tag{2.10}$$

$\mathbf{t}$ being $-R \cdot \widetilde{\mathbf{C}}$.

The final form of the camera projection matrix P for a finite camera with 11 degrees of freedom is

$$\begin{aligned} P &= K[R|\mathbf{t}] \text{ or} \\ P &= KR[I| - \widetilde{\mathbf{C}}]. \end{aligned} \tag{2.11}$$

With this projection matrix, every 3D point in front of the camera can be mapped to a pixel on the image plane. As one dimension is lost during projection, we cannot determine the exact 3D point for a pixel, but only a direction in space. With only one camera, there is no possibility to measure depth information.

2.2.2 Multiple Cameras

The previous section described the properties of one camera. There is not much (in terms of 3D reconstruction or image stitching) one can do with one image. To be able to combine information from multiple images, we will review the relevant parts of the multi view geometry of Hartley and Zisserman [25].

The geometric relationship between two images can be expressed in epipolar geometry (see Figure 2.5). It describes a family of planes (Figure 2.6a), which are defined by the base line between the camera centres as fixed parameter. The concrete instance of the plane π is either defined by a 3D point X or an image point x (Figure 2.6b).

For a fixed setup of two cameras there exists a mapping

$$\begin{aligned} x &\mapsto l' \\ l' &= Fx \end{aligned} \tag{2.12}$$

called fundamental matrix F. Any point x in one image *has* to lie on line l' in the other image. The derivation of Equation 2.12 is shown in [25, 60]. F does not depend on the scene, but only on the relative orientation of the cameras. Therefore this relation can be used to restrict the search of a salient point in one image to a line in the other image.

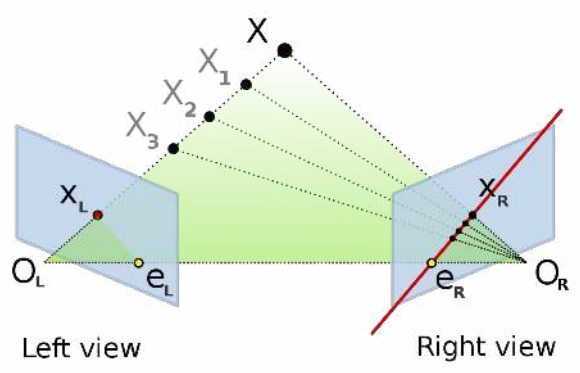
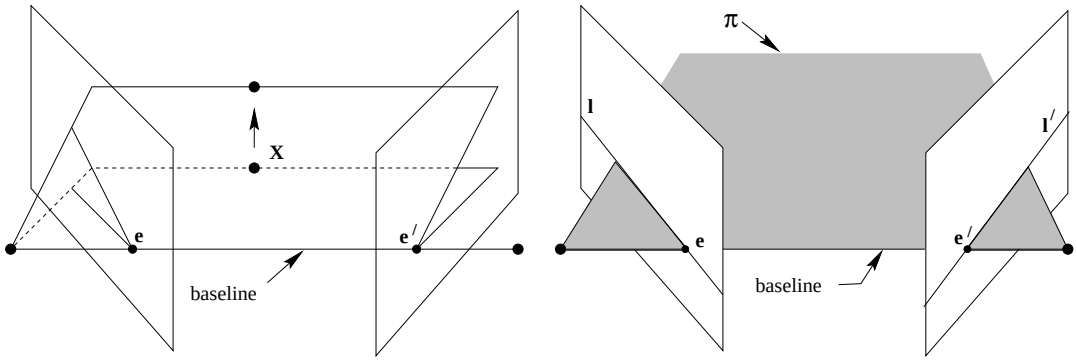


Figure 2.5: Epipolar geometry for two cameras. The line connecting the camera centres O_L and O_R is called *base line*. It can but does not have to intersect the image plane. If it does, the intersection points e_L and e_R are called *epipoles*. Once a point X is projected into the left camera as x_L, its depth cannot be recovered anymore. But as seen in the figure, independent of its distance to the camera $(X, X_1, X_2, \ldots)$, it always will be projected onto the red line in the right image. This line is called *epipolar line*.
Image courtesy by Arne Nordmann under CC-BY-SA-3.0 via Wikimedia Commons



(a) The epipolar geometry defines a pencil - a group of planes having the baseline as common line. The actual epipolar plane is either selected by a 3D point X or an image point x.

(b) An actual instance π of the epipolar pencil (Figure 2.6a). Once defined (e.g. by an image point) it is obvious, that the corresponding point in the other image has to lie on the line l resp. l'.

Figure 2.6: Details of the epipolar geometric relations Image courtesy by Hartley and Zisserman [25]

For calculating F, another property is needed. Given are two corresponding points x and x'. x' has to lie on l', therefore $x'^T \cdot l' = 0$. Inserting this into Equation 2.12, it yields

$$\begin{aligned} x'^T \cdot l' &= x'^T F x \\ x'^T F x &= 0 \end{aligned} \tag{2.13}$$

Given enough point correspondences, F can be calculated for a given, fixed stereo camera setup [25].

Although being a very brief overview, this section should give an idea of camera geometry including multi view setups. In literature [25, 60, 56] these topics are handled in-depth. This includes derivations of formulas, algorithms to calculate F, K and P and a more general view on the topic.

2.3 Image Stitching

The basic image stitching procedure has been described comprehensively by multiple authors, amongst others Richard Szeliski in a tutorial [55] and David Chapel [11]. In this section we will briefly introduce the most important steps of Richard Szeliskis guide to image stitching. He also includes some drawbacks of this approach along with possible solutions. We present those drawbacks and their role in planetary environments in the end of this section.

An overview of the basic process, which we will present on the next pages, is shown in Figure 2.7.

2.3.1 Image Geometry and Motion Models

In order to be able to combine several images into one, it is important to understand how images are formed in cameras and how to describe this. The **image geometry** is the relationship of pixels in the image and global world coordinates. It can be expressed by a simple projective pinhole camera model, a fish eye camera or, in the most general form, an arbitrary mapping of each pixel to a ray of sight [23]. Szeliski focuses on the pinhole camera model using the notation of Hartley and Zisserman [25] as described in Section 2.2.1. Fish eye cameras introduce a lot of distortion, especially towards the image borders in order, to increase the field of view. Stitching such images

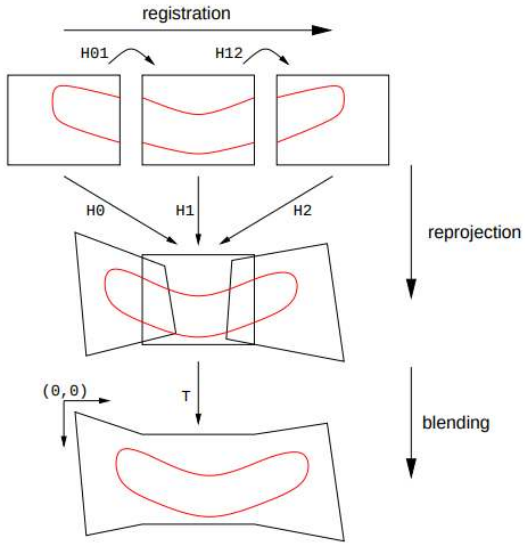


Figure 2.7: Basic steps of image stitching. First, the images are registered relative to each other (already available alignment can be refined). $H01$ and $H12$ is the relative homography (see Figure 2.8) between images 0 and 1 resp. 1 and 2. The next step is the projection onto a common surface and into a common point of view. The needed homographies $H0$, $H1$ and $H2$ are calculated using the common projection and the relative homographies. The middle image is used as reference frame, $H1$ therefore is the identity matrix. Lastly the images are composed into one panoramic image using transformation T. It only shifts the origin of the image. Image courtesy by Capel [11]

yields panoramas with high variations spatial resolution. The majority of these cases can still be handled by rectifying the image beforehand. However some information is always lost during transformations, so using projective cameras directly is preferred.

Motion models are used to map a pixel in one image to a pixel in *another* image. In order to be able to calculate this information, one has to know the relative position and orientation of the cameras, e.g. the epipolar geometry [25]. As described in Section 2.2.1, given the epipolar geometry by the fundamental matrix F and an image point in one image, it is possible to determine the line in the other image on which the point must lie. In order to stitch the images into one seamless panorama it is necessary to know the *exact* position of a pixel in the other image.

The relation of two corresponding point sets on different planes is called a homography H (Figure 2.8). Calculating the homography between two image planes works if the points lie on a plane in 3D space too (Figure 2.9a). If the 3D points which are visible in the images do *not* lie on a plane, they cannot be transferred between the image planes by homographies. Homographies are transitive. If there is a homography H_{01} between planes 0 and 1 and H_{12} between planes 1 and 2, then the homography H_{02} between planes 0 and 2 is

$$\begin{aligned} x_2 &= H_{12} \cdot x_1 \\ x_2 &= H_{12} \cdot H_{01} \cdot x_0 \\ H_{02} &= H_{12} \cdot H_{01}, \end{aligned} \tag{2.14}$$

given that $x_{0,1,2}$ are points on the respective planes.

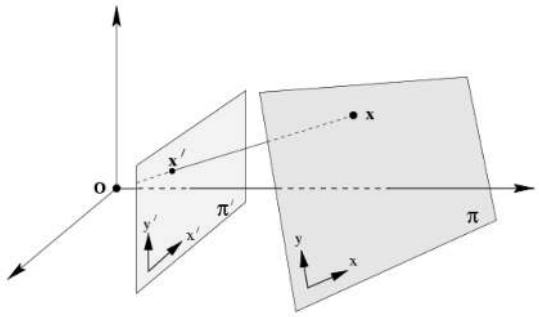


Figure 2.8: A homography H maps points from plane π to plane π' by $\mathbf{x}' = H \cdot \mathbf{x}$
Image courtesy by Hartley and Zisserman [25]

Most scenes do not consist of planes. A more common case is a pure rotational camera. In such situations, the camera's centre of projection is not moved between image acquisitions, just rotated around it (Figure 2.9b). Rays of sight through corresponding points coincide, so according to projective geometry, they intersect at the plane at infinity Π_∞. This plane $\Pi_\infty = (0, 0, 0, 1)$ contains all points at infinity [25]. As in this case all 3D points lie on one plane, a homography between the camera planes can be calculated which only depends on the rotation matrix R.

2.3.2 Cylindrical and spherical projection

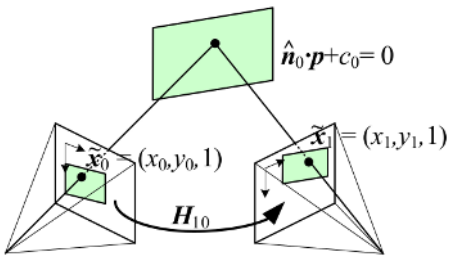
After determining the 3D position of the pixels, the images are transformed into a cylindrical or spherical projection for alignment. As seen in Figure 2.10, each point is projected onto a cylindrical, spherical or more complex surface. Which surface is used, depends on the camera settings. If there is only a panning movement, a cylindrical projection may be used. If both pan and tilt angle change, a spherical projection yields better panoramas [55]. Figure 2.11 shows an image before and after projection to a sphere.

After projecting the image onto a panorama surface, image alignment can be reduced to translation and rotation of the images on this surface. There is no need to take the 3D position and rotation of the cameras into account.

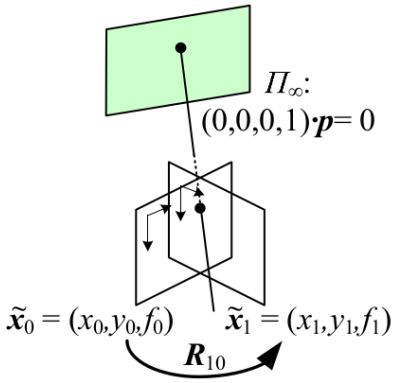
2.3.3 Registration and Alignment

After defining image parameters and projection of the final panorama, the images have to be registered and aligned on this projection surface. In the standard case there is no prior knowledge of where each image will be situated in the panorama. But even if there is pointing information available (e.g. from an angle transmitter in automated tripods), this data can be inaccurate. The goal of this phase of panorama stitching is to correlate images to get their relative position and orientation to each other. The better and more accurate this information is, the less errors will be left to later stages of the stitching process. According to Bob Deen, even some parallax errors can be mitigated by good registration of images [16].

There are two main ways how to determine an accurate position of the images: pixel-based and feature-based alignment.

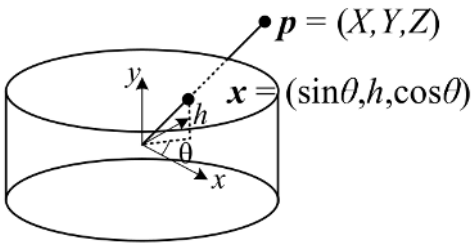


(a) Two cameras take an image of a plane $\hat{\mathbf{n}}_\mathbf{0}$. All points p get projected into camera 0 as $\tilde{\mathbf{x}}_\mathbf{0}$ at pixel position (x_0, y_0) and into camera 1 analogously. As there exists a homography between image plane 0 and $\hat{\mathbf{n}}_\mathbf{0}$, and between plane 1 and $\hat{\mathbf{n}}_\mathbf{0}$, homography H_{10} exists and can be calculated. The rays of sight of points $\tilde{\mathbf{x}}_\mathbf{0}$ and $\tilde{\mathbf{x}}_\mathbf{1}$ intersect at plane $\hat{\mathbf{n}}_\mathbf{0}$ Image courtesy by Szeliski [55]

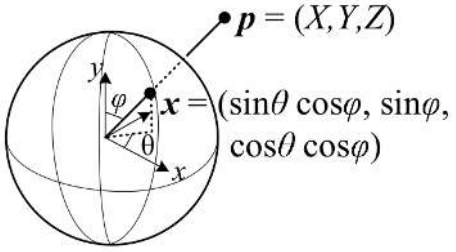


(b) A purely rotational camera setup. All rays of sight of corresponding points coincide, so their intersections lie on Π_∞. A homography between the image planes is therefore possible and only depends on the rotation R_{10} between the cameras Image courtesy by Szeliski [55]

Figure 2.9: Mathematical relation of pixels showing the same 3D point



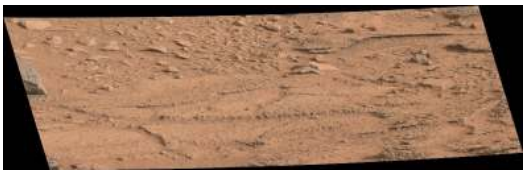
(a) Cylindrical image projection. All points can be described with its azimuth angle θ and height h. The distance of the point to the centre is lost during this transformation.



(b) Spherical image projection. Used for panoramas with different tilt angles. The projected points are defined by azimuth θ and altitude φ. Note that the sphere radius is irrelevant for this projection.

Figure 2.10: Image projections Image courtesy by Szeliski [55]

(a) Raw image of Mastcam Left. It has been cropped before download to Earth to reduce redundancy in the data set and save bandwidth. Mastcam Left has a higher field of view and therefore more (unneeded) overlap than Mastcam Right.
Image courtesy by NASA/JPL



(b) Same image projected onto a given sphere, defined by its origin and an axis. X resp. Y coordinates represent azimuth resp. longitude angle on the sphere. Note that the 3D points for projection were calculated using a plane as a surface model [16]

Figure 2.11: Reprojection of an image to a sphere

Pixel-based alignment uses the raw pixel values and tries to adapt the motion parameters of the images to minimise differences in overlapping areas. The base of this method is to "shift or warp the images relative to each other and to look at how much the pixels agree" [55]. There are a lot of approaches like hierarchical coarse-to-fine techniques [49] or Fourier transformations [42]. Different error measures such as Sum of Absolute Differences (SAD) or correlation measures such as Normalized Cross Correlation (NCC) can be used to determine the quality of a configuration.

Feature-based alignment is a more sparse approach to image registration. It has been in use a long time [24] but is still a research topic (e.g. [12]). Instead of using all pixel values, salient, distinguishable points are extracted from every image. Corresponding points in different images are matched to create a link between camera positions. Depending on the quality and distribution it is much less computational effort to solve the registration problem based on a few points compared to all pixel values. We will cover feature-based alignment more thoroughly in Section 2.5.

Registration strategies (pixel-based and feature-based) have to support sub-pixel accurate image alignment. This means that the pixels of two overlapping images do not have to be congruent with each other. As soon as the camera is slightly rotated during image acquisition, congruent pixels are impossible. However, before the actual stitching process, the images have to be transformed into a common pixel grid. Resampling operators like bilinear, bicubic or Lanczos filters are necessary for this step.

2.3.4 Composition

Once all input images are in place, we have to combine them to one panorama. Each pixel, no matter how many images contribute to it, needs exactly one colour value. If the images are perfectly aligned with exactly the same exposure and illumination conditions, every pixel contributing to one position should have the same colour already. Combining them into one colour value is a trivial task as "any pixel or combination will do" [55]. This scenario assumes an ideal camera without noise and vignetting. Vignetting is a bias in brightness in the image, depending on the location. Often the images gets darker towards the borders due to the construction of the lenses. But also dust on the lens can introduce vignetting effects.

In any real-world situation, at least one of the named preconditions will not hold and we need to put some effort into creating a seamless panorama nevertheless. There are several, more or less intrusive, approaches to handle different errors left in the images.

Seam selection is the least intrusive way to combine multiple images in terms of retaining original information. It only uses already available pixel values for the final panorama. The overlapping area is split in two separate parts along a line, while every part is assigned to the respective images. There are a lot of different approaches on how to calculate this cut line. It can simply be the image border (e.g. [16]), a straight line through the overlapping area or a sophisticated method like the Dijkstra algorithm [19], or a graph-cut minimizer [8]. Seams selection focuses on the mitigation of structural errors like parallax. Radiometric differences between images cannot be handled well.

Blending can be used in addition to or instead of seam selection. Each final pixel is assigned a linear combination of all contributing images. The contribution weight depends on the distance to a cut line, the distance to the image centre or some other measure. In case of the cut line distance, this method is called feathering [11]. The main error addressed by this approach are radiometric differences.

Warping is the umbrella term for a broad range of methods. Despite its contradiction to scientific integrity (see Section 2.1) we describe it for sake of completeness. The only degree of freedom in standard blending is the weighting of each pixel. For warping, the common mechanism is to bend and distort the images locally in

a way that the images agree with each other in the overlapping area. Depending on the kind and magnitude of the errors, visible seams can be effectively removed. But the content of the image is altered in a way that is not always obvious. In scenarios with no direct visual feedback (e.g. by the photographer) or ground truth this can lead to misinterpretations of the result [17]. Most algorithms presented in Section 2.3.5 are in this category.

Figure 2.12 shows the effects of different composition methods on pairs of (misaligned) images.

2.3.5 Open Issues and Recent Developments

As mentioned in the beginning of this section, Richard Szeliski describes some drawbacks and open issues with panorama stitching. Most influential are motion between image acquisition and parallax.

There is not much macroscopic motion on Mars to mitigate. But changing shadows on different times of the day seem like motion on images and have to be handled similarly. On the other hand dust devils sometimes occur, raising dust and thereby changing the appearance of surfaces. In addition, future missions like ExoMars include several independent devices which of course introduces major motion artefacts in images. There are frameworks which can mitigate such artefacts but require multiple acquisitions of the same area [58]. To save bandwidth for download, the images include just as much overlap as needed for reliable matching, eliminating the usage of those tools. Another pragmatic approach to motion mitigation is to handle it as if it was parallax. For sake of simplicity, we will use this approach in this thesis, with the possibility of later adaptions.

We described the origin of parallax in Section 1.2, especially in Figure 1.3. Parallax issues cannot be solved correctly without knowledge of the 3D geometry of the scene. Most parallax aware stitching algorithms roughly use the following procedure:

If feasible, 3D information is calculated from the input images by a Structure from Motion (SfM) or plane sweep [13] approach. In a typical panorama setup it is however not possible to generate a reliable 3D reconstruction. There is often too little overlap for a complete reconstruction or the base line is too small for usable 3D intersections. To be able to process images in those scenarios, the geometry of the scene or its approximation is needed as input for the stitching algorithm. When the structure is known, the images can be projected onto its surface and back-projected

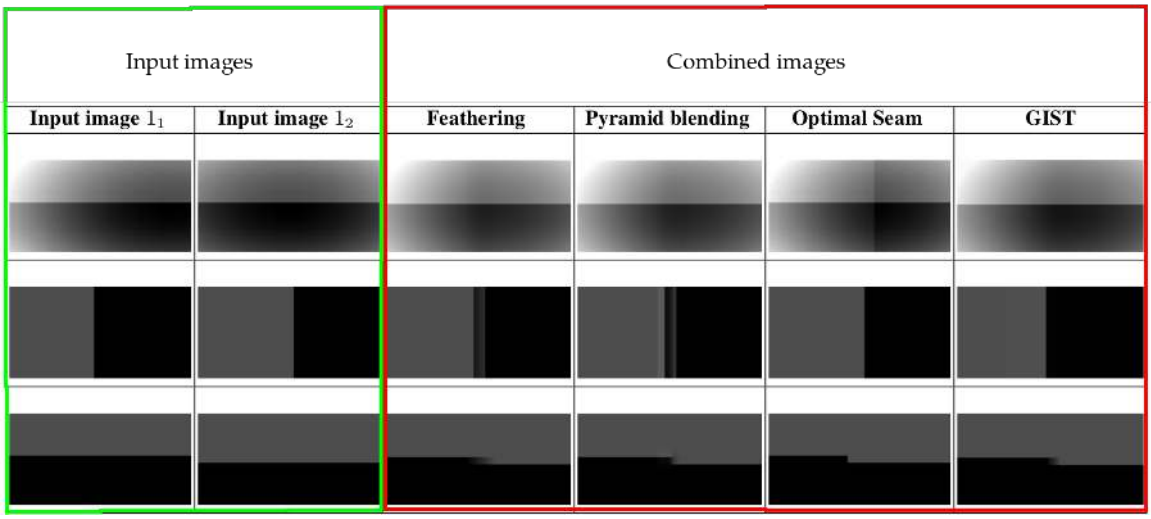
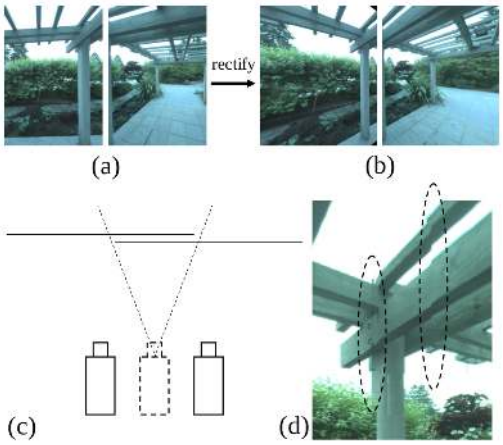


Figure 2.12: Different approaches to composition. The first two columns (green) show different pairs of input images which are not aligned exactly. The rest of the table (red) shows the effect of different merging and blending methods. Feathering [58], pyramid blending [2, 45], optimal seam [15, 20, 37] (mostly based on the Dijkstra algorithm [19]) and GIST (gradient domain fusion) [32] Image courtesy by Levin et al. [32]

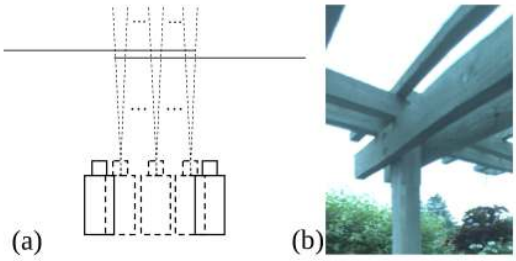
in one common virtual point of view. This point of view is either user-defined or the centre of gravity of all images. The better the surface reflects the actual structure of the surroundings, the less geometric errors are left in the rendered images.

Sing Bing Kang et al. [30] calculate ,,multiple intermediate virtual viewpoints" between cameras not sharing the same point of view (Figure 2.13b). Due to the high density of intermediate viewpoints, the accuracy of the estimated depth information has less influence on the result. For each viewpoint, a plane sweep depth estimation is performed. The parallax error is distributed over the whole overlap area and therefore produces no visible seam. Figures 2.13a and 2.13b show the effect of this approach. However as the view point interpolation blends two (or more) images together, the scientific integrity (see Section 1.1) cannot be satisfied with this approach.

Qi Zhi [61] tries to solve the same problem for dynamic scenes. Her setup consists of multiple fixed video cameras with wide base lines. In addition to hide seams from parallax, movements in the scene and between cameras should be fluent and not disrupted. Zhi proposed depth-based image mosaicing (DBM) to solve this problem. The main idea is to calculate "depth cues" for the image content in overlapping areas. In non-overlapping areas, the depth cues are extrapolated based on colours in the image. One basic assumption is a constant depth value in regions with similar colours. This is of course not always true and leads to geometric incorrect mosaics, but "most



(a) Stitching result using one virtual intermediate point of view. The depth structure of the overlap is estimated with a plain sweep algorithm [13]. Any estimation errors are clearly visible as seam at the borders. Image courtesy by Kang et al. [30]



(b) Stitching result using multiple virtual intermediate points of view. The plain sweep algorithm performed for each column, distributing remaining errors over the whole overlap area. Image courtesy by Kang et al. [30]

importantly, the resulting outputs are perceptually acceptable" [61]. Therefore this method is not suited for panorama stitching in planetary environments as it also cannot fulfil the requirement of scientific integrity.

Agarwala et al. presented a framework for digital photomontage [4]. As a side effect it also can be used for image stitching, but it is designed for combining multiple images of (exactly) the same scene. A classical example would be a group picture where someone looks away in every single image. Using digital photomontage one can create a picture, where everyone is smiling into the camera, without noticeable seams between the persons. The framework uses a graph cut minimiser [8] with a human in the loop to calculate visually pleasing photomontages. They defined a large set of objectives that a user can define as minimising cost function (see also Section 2.6.3). Output of the graph cut minimiser are labels for each pixel, which input image to use for this location. After defining cut lines resp. label regions, they apply "gradient-domain fusion". The main idea of this fusion approach is to stitch gradient images and try to find a colour image whose gradient best matches the stitched gradients. According to Agarwala et al. this eliminates visible seams in colour domain. Up to the blending this method fulfils the requirement of scientific integrity, as it directly uses the values of input pixels without modifying them.

2.4 Dense Sampling

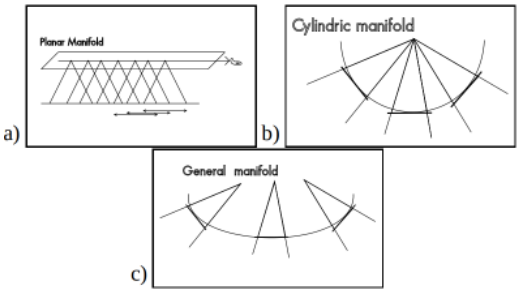
In the last section we presented a workflow for stitching single image frames into one panorama. However, there is a completely different strategy for creating panoramas, namely manifold projection, developed by Shmuel Peleg and Joshua Herman [46]. The input for their panorama pipeline does not consist of single images, but of a video stream covering the view of interest. Due to the high density of input data, alignment and combination is a lot easier. The main principle is to use only one stripe of each frame (see Figure 2.14b) and project it on the used manifold as seen in Figure 2.14a. Changes of the point of view, in illumination and most movements in the images are relatively slow compared to the frame rate. Figure 2.15 shows the output of such process, clearly showing the single stripes on the upper and lower border.

Following this initial idea, there were a lot of extensions and adaptions of manifold projection [29, 44, 47, 51]. Some of these extensions [47, 51] even allow the creation of stereographic panoramas. This methods are used in mobile phones to generate panoramas.

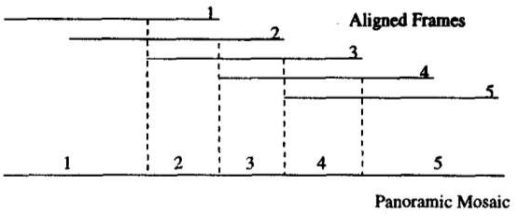
Due to the high amount of data necessary for this class of methods, it is unsuitable for processing of extraterrestrial data. Data transmission rates from outer space are limited. In addition, other deep space projects sharing NASA's Deep Space Network (DSN) reduce the possible size and amount of images taken. Processing the data in-situ and only transmitting the results is also not an option, because scientists need access to the accurate raw data for reliable research.

2.5 Calculation or Refinement of Image Orientations

The process described in this section tries to bring all images into one combined coordinate system. This is paramount for creating seamless panoramas as we need to know where to project which image. As described in Section 2.2, camera parameters contains two types of information: Each camera has its intrinsic parameters which contain at least focal length and principal point. More advanced models also contain vignetting information and lens distortion parameters. Secondly each image needs a description of its position and orientation. This can be either globally in an external coordinate system (like Global Positioning System (GPS) coordinates on Earth) or at least a relative orientation to the other images.



(a) Possible surfaces for manifold projection. Easier manifolds (a) and b)) can be combined arbitrarily for more complex movements and manifolds (c)) Image courtesy by Peleg and Herman [46]



(b) Dense overlap of images for manifold projection. Only the centre stripe of each frame is used for the final mosaic Image courtesy by Peleg and Herman [46]

Figure 2.14: Basic principles of manifold projection

Figure 2.15: Panorama created with manifold projection. The curved border results from unsteady motion of the handheld camera Image courtesy by Peleg and Herman [46]

Section 2.3.3 describes two principal ways to gain or refine such information after image acquisition. Pixel-based methods are potentially more accurate as they use all available data. However with the development of good key-point detectors and descriptors like SIFT [35], feature-based methods gained popularity.

Given the available data structures and algorithms in the used software environment, we will focus the second approach in this thesis. The following subsections describe one method of feature-based alignment, namely *Bundle Adjustment*. First we need to find salient points in each image which can than be matched to create corresponding point sets. A point correspondence contains the locations of a spot in the scene in two or more images. Based on these links, we can then approximate the geometric relation between the images.

2.5.1 Finding and Describing Salient Points

First, one needs to find salient points in each image. Those points are outstanding in a way that they are likely to be re-found in other images from other points of view. Algorithms performing this task are called salient point *detectors* (e.g. [35, 7]). It is important that the position. of the points are accurate, as all following steps build upon them. Good detectors give sub-pixel positions for salient points.

In order to find matching points, we have to *describe* those points mathematically. There are a lot of different approaches on this topic, often tightly linked to the descriptor. They range from naively stacking the surrounding pixels into a vector, up to a scale and rotation-invariant feature vectors based on local gradients as in SIFT [35]. The latter is often used for 3D reconstruction as it can robustly find corresponding points in view points taken from different angles of up to 30°.

Descriptors also define an error measure (e.g. the euclidean distance between the vectors) to define how similar two points are.

2.5.2 Finding Correspondences

In order to find corresponding point sets between two images, their point descriptors are compared with the above-mentioned error measure. If the difference is below a certain limit, we have a matching candidate. However a lot of reasons exist why a candidate could not actually be a valid match, e.g. repetitive structures on the images or noise. There are different methods how to filter false matches, e.g. by checking the epipolar geometry or the homography between the images. Random Sampling

Consensus (RANSAC) [22] is an often used approach for this task. It repeatedly calculates the geometric relation from a minimal amount of points and compares its appliance to all other points. If enough points agree with a solution, a consensus is calculated from these consistent points.

2.5.3 Calculate Camera Parameters

The process of calculating camera parameters from corresponding points is called *Bundle Adjustment*. All salient points of an image are cast into 3D space as *bundles* of rays of sight. If all point correspondence and camera parameters are correct, the rays of sight for all corresponding points intersect. This method is often used in sparse 3D reconstruction. We will describe its appliance to planetary panorama stitching in the next subsection.

For Bundle Adjustment, one basically tries to find a solution for

$$x_j^i = P^i X_j \tag{2.15}$$

where X_j is the j-th 3D point projected into camera i by projection matrix P^i onto point x_j^i [25]. This will be not possible in general due to noisy data. Therefore the re-projection error

$$d(x_j^i, P^i X_j)^2 \tag{2.16}$$

is minimized for all i, j, where $d(x, y)$ is the euclidean distance between x and y. x_j^i is known, the process simultaneously recovers P^i and X_j.

The details of initialisation and minimisation are explained in detail in literature (e.g. [25, 56]). Solving a problem in computer science often involves reducing or converting it into an already solved problem. (Nonlinear) minimisation is a very broad and intensively researched topic with very good implementations available (e.g. Ceres-Solver [3]). Noah Snavely built an open source tool called Bundler on top of Ceres-Solver to enable easy and fast 3D reconstruction of arbitrary image collections [57]. However, according to its documentation, it is not able to process sparsely overlapping image sets.

2.5.4 Miss-Distance Tie Points

The standard approach of Bundle Adjustment heavily relies on the 3D points for back-projection (see Equation 2.16). In some cases the baseline between cameras is too small for a reliable intersection. Therefore also the 3D positions of points are uncertain.

If – due to limited transmission bandwidth – e.g. only the right images of a site on Mars are downloaded to Earth, there is no possibility for 3D reconstruction. Still the baseline is too big to ignore it completely and assume the images to be taken from one spot. The resulting panorama would show a lot of parallax artefacts as the camera does not rotate around its centre of projection but around a common centre of the stereo rig. Bob Deen from Jet Propulsion Laboratory (JPL) at NASA developed a modification to the bundle adjustment minimisation term [17]. He does not rely directly on 3D intersection points. Instead he projects both corresponding image points as rays of sight into 3D space and calculates the minimal distance between those rays. This distance is used as minimisation criteria. Even if the rays have a glancing intersection, the minimal distance is an unambiguous property.

Deen mentions an additional advantage of his tie points. The error term is a metric distance and therefore more meaningful than the reprojection error measured in pixels. They respect the geometry of the image while not relying on diffuse intersections.

2.6 Cut Line Selection

Selecting a cut line between images was introduced as part of Section 2.3.4. But given scientific integrity of planetary panoramas as a requirement, blending of image seams and unconstrained warping of overlapping image areas is excluded. Therefore the selection of the cut line between images plays a very important role for the quality of the final panorama. Before presenting different approaches how to define or calculate cut lines, we will describe the general procedure and properties.

A cut line defines the common border of two overlapping images as seen in Figure 2.16. In order to completely define the region, the line has to run between the intersection points of the image borders. In case of more complex overlaps, this is still possible by ordering the overlapping points around the image centres (see Figure 2.16b) [54]. The possible course of the line covers the whole overlapping area, including the image borders.

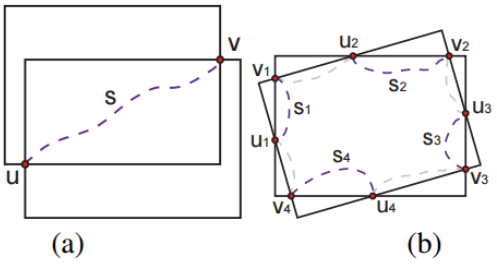


Figure 2.16: Example cut line between two images. The cut line s connects intersection points u and v. More complex overlaps can be handled by ordering the intersection points around the centre and connecting them pairwise. Note that two different configurations are possible (denoted in light-grey). Image courtesy by Summa et al. [54]

2.6.1 Image borders

The simplest form of calculating the cut line is to take the image borders as a cut line. This has the same effect as just using one of the two images in the overlapping area. There are systems ([16]) using this approach because of its predictable cut lines. If the images are well registered both geometric and radiometric, the seam is not visible. Any remaining alignment errors, parallax errors or radiometric errors are clearly visible as such. Therefore the probability of geologists misinterpreting stitching artefacts as natural structures is very low.

In addition, calculation is very inexpensive and fast. The only available degree of freedom is precedence of the images. By manipulating this ordering, parallax artefacts can be reduced [16]. Figure 2.17 shows the labelling result of two overlapping images using image borders as cut lines.

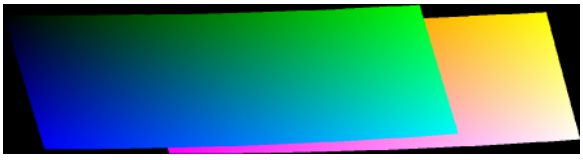


Figure 2.17: Colour coded image mapping for image borders as cut lines. The red channel is the image index, while green and blue encode the location of the pixel in the source image

2.6.2 Straight lines

Besides using image borders as cut lines, there are other ways to use straight lines. The easiest is to directly connect the intersection points. Alternatively, the distance of each pixel to the respective image centre [11] could be a potential decision criteria also yielding straight lines. The reasoning behind these strategies is that data in the centre of an image is potentially more accurate than data near the borders. Reasons are missing or inaccurate lens distortion information, vignetting or chromatic aberration. The advantages of this method are the same as for image borders as cut lines. Straight lines are very fast to compute and do not lead to misinterpret-able images. Figure 2.18 shows the usage of the centre-distance as a decision criteria. The resulting cut line is straight, bounded by the image borders.

2.6.3 Graph cut minimiser

We already mentioned above, that it is common to transform a problem in a way that it can be solved by other, already existing solutions. Instead of searching for a line between the intersection points of the image borders, we try to assign each pixel in the overlapping area a label. The label defines which of the source images is used for the final panorama. Boykov et al. [8] presented a graph cut minimiser to efficiently calculate an optimal labelling according to energy costs.

The energy function E of a labelling L is

$$E(L) = \sum_{p} E_d(p) + \sum_{p,q \in \mathcal{N}} E_s(p,q),$$

$E_d(p)$ being a unary cost term for pixel p (in this context usually 0 for valid pixels and ∞ for invalid pixels [4]). $E_s(p,q)$ is a pairwise term, defining the cost of cutting between neighbouring ($\mathcal{N}$) pixels p and q. A graph cut minimiser finds a solution L, so that $E(L)$ is minimal.

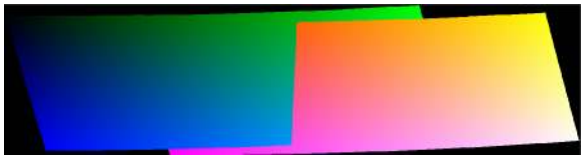


Figure 2.18: Assigning each pixel to the image whose centre is nearest using the same colour coding as in Figure 2.17

The binary energy function can for example be based on a gradient image [4], allowing a cut line (resp. a label change) on sharp edges in the images. Any naturally present visual seam does not taint the visual appearance of the panorama. Therefore switching images on natural edges is desired, in the sense that geometric seams can be hidden there.

The concept of labelling each pixel instead of calculating one line changes the possible output. Instead of simple lines, islands can occur, meaning the pixels of one image do not have to be connected anymore. While the solution may be energetically minimal, islands of "foreign" pixels are not favoured. They can not actually improve the visual perception, but only increase the risk of producing artefacts.

An advantage of the graph cut approach is that it can be applied to more than two images. It generalises to find a global solution of all overlapping areas simultaneously. As it finds a global solution, it takes into account every possible constellation of overlaps automatically without having to construct special cases (for e.g. complete occlusion of one image in another).

The underlying problem of assigning an optimal labelling is NP-complete [8], therefore all algorithms try to find approximations. Initially even the approximating solutions had long run times and high memory consumption. More recent developments such as [18] reduce memory consumption a lot and allow parallelisation. However, they only work on grid-graphs meaning that the label is ordered (such as a disparity/depth value). An arbitrary label as the image index is unsupported by them.

2.6.4 Panorama Weaving

The most recent work on calculating cut lines is panorama weaving, developed by Brian Summa et al. [54]. Summa et al. present a framework which reaches low energy cuts comparable to graph cut approaches. However there is the possibility for user intervention and it is computationally efficient.

They calculate pairwise optimal cut lines which can be calculated in parallel. The major contribution is an efficient way to merge those pairwise cuts into a global optimal solution. Figure 2.19 shows an example output of the panorama weaving workflow including an index image. It colour-codes the source image for visualisation of the cut lines.

Figure 2.19: Example output of the panorama weaving pipeline, showing three different near-optimal cuts. The user can decide which configuration looks best. Image courtesy by Summa et al. [54]

Pairwise seams

The first part of panorama weaving is the pairwise image boundaries. All global overlap relations are ignored in this step. An example of a pairwise panorama weaving seam is shown in a previous section in Figure 2.16. In order to calculate this line, a graph is constructed from the overlapping image area. Every pixel corresponds to one node, neighbouring pixels being connected by edges. The line is constructed using the Dijkstra algorithm [19], therefore each edge needs a weight. This weight contains information about the image and therefore makes the Dijkstra algorithm sensible to its content. As Dijkstra cuts *along* edges and not *across* edges, we need to transform the image graph into its dual form as seen in Figure 2.20.

The calculation of the edge weights can be done in different ways, similar as described in Section 2.6.3. Summa et al. propose

$$E_s(p,q) = \left\| I_{L(p)}(p) - I_{L(q)}(p) \right\| + \left\| I_{L(p)}(q) - I_{L(q)}(q) \right\|$$

to cut at similar pixel values

$$E_s(p,q) = \left\| \nabla I_{L(p)}(p) - \nabla I_{L(q)}(p) \right\| + \left\| \nabla I_{L(p)}(q) - \nabla I_{L(q)}(q) \right\|$$

to cut at similar gradients. $L(p)$ is the labelling of pixel p and $I_l(p)$ is the image intensity of the image with label l at pixel p. By using $L(p)$, neighbouring pixels labelled with the same image have automatically zero cut costs, as there is no cut line between them.

The Dijkstra algorithm finds the path through the given graph which has the minimal sum of edge weights. It performs a breath-first search beginning at the start node. Every reached node is added to an open-list, which is sorted by the cut cost up

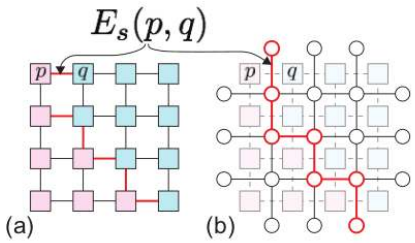


Figure 2.20: Dual image graph representation. The edge weight stays the same and contains information about the cost of cutting the image. Image courtesy by Summa et al. [54]

to the respective node. When all edges from a node are processed, it is added to the closed-list. The algorithm then continues the search with the item in the open-list which has the least cost. The loop terminates when the end node is found and every node in the open-list has a higher cost.

Summa et al. modified this standard implementation of the Dijkstra algorithm. They do not stop when finding the end node, but build a minimal path tree for reaching each pixel in the overlapping area. This tree is built from both start and end node. Although this means significant memory overhead, it enables interactive addition of support points, through which the cut line has to pass. The re-calculation of the optimal path through this point is just an iterative parent-lookup to both start and end node. See Figure 2.21 for a visualisation of this procedure.

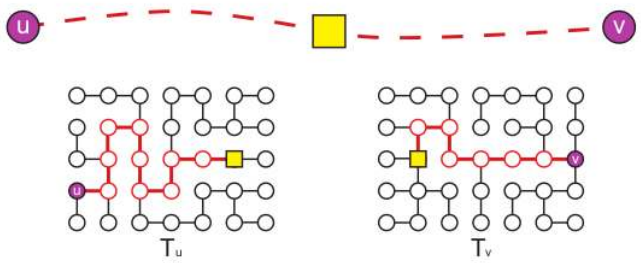


Figure 2.21: The yellow square has been inserted by the user to force the cut line through it. Calculating the new cut is done by iteratively looking up the parent in both directions Image courtesy by Summa et al. [54]

Global seams

Merging multiple pairwise seams into one global seam network is a non-trivial task. In nearly every setting there are areas in the panorama in which more than two images contribute. In those cases, the initial seams are unlikely to cross at exactly the same spot. Summa et al. provide an algorithm to calculate optimal "branching points".

They introduce the notation of an adjacency mesh, containing the overlap structure of the images. Starting from the full neighbouring graph, non-overlapping maximal cliques are searched (see Figure 2.22). Removing the inner edges from all such cliques yields a planar graph. Each remaining edge corresponds to a pairwise overlap, each face indicates a multi-overlap. The number of edges surrounding a face is the number of simultaneously overlapping images.

After calculating the adjacency mesh, each face can be handled independently and in parallel. The goal is to find an optimal branching point for each face as explained in Figure 2.23. There are however some configurations which still cannot be handled by this approach. For example, as shown in Figure 2.24, the cut lines can still cross, depending on the image content. In such cases, user interaction is *required* and not optional. We will not summarize the actual detection and handling of invalid cut lines, as there is no possibility of user interaction planned for this thesis. Although it should be possible to add this functionality later, such features are out of scope for now.

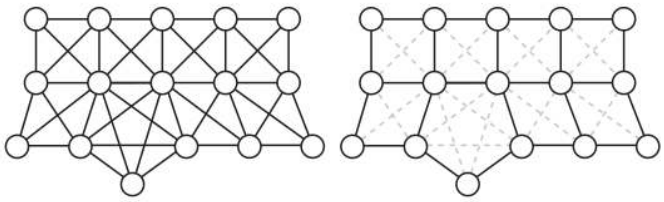


Figure 2.22: Starting from the global overlap graph on the left (nodes depict images, overlapping images are connected by edges), the inner edges of maximal cliques are removed. Although being NP-complete, clique finding is feasible for most real-world panorama settings. The resulting graph is a dual adjacency mesh, edges are orthogonal to cut lines, faces correspond to multi overlaps. Image courtesy by Summa et al. [54]

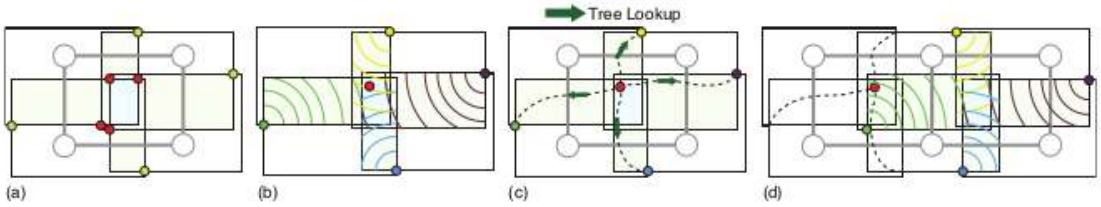


Figure 2.23: This figure shows the calculation of cut lines for multi overlaps. Yellow dots in figure (a) are start nodes, red dots are end nodes which should be combined to one branching point. The branching point is found by searching the node with the minimal combined cut cost from all four start nodes as visualised in figure (b). This point intrinsically has to lie within all involved images. The actual cut lines are then calculated by a simple tree lookup back to the start nodes (c). Real world applications showed that for calculating the branching point for neighbouring multi overlaps, the original end node can be used as start node. The resulting branching point is not influenced by small variations of the start node. Therefore the Dijkstra graph for this overlap does not have to be recalculated, and all branching points can be calculated in parallel. Image courtesy by Summa et al. [54]

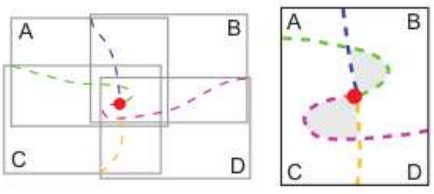


Figure 2.24: As the edge costs for the Dijkstra algorithm are based on pairwise overlaps, cut lines in multi overlaps can intersect which yields invalid configurations. The grey areas in the right diagram can be assigned to both A and B, or C and D respectively. Simply choosing one label is not favourable as the resulting cut line is not based on the image content Image courtesy by Summa et al. [54]

2.7 Summary of Literature Review

Before starting with our contributions, we briefly summarize existing approaches in this section. The main reason for creating another panorama stitching framework is the need of scientific integrity of the final panorama. Section 2.1 reviewed the basic definitions by Bob Deen [16].

Cameras as the principal component needed for panorama stitching are reviewed in Section 2.2. We describe the basic mathematical properties of central projective cameras as well as the relation of two cameras viewing the same scene. These equations are needed especially for pointing refinement but permit a better understanding of the whole topic.

Sections 2.3 to 2.6 introduce to the standard panorama stitching approach. Some modules such as the selection of an appropriate cut line in an overlapping area are reviewed in more detail because of their important role in this thesis. The concept of dense sampling panoramas reviewed in Section 2.4 can *not* be used for panorama stitching in planetary environments. It assumes completely different circumstances of image acquisition.

A conclusion of this literature review is that there is still ongoing research in the topic of panorama stitching. Most recently developed approaches are however often unsuitable for use in photogrammetry. The visual perception is the main objective in the final result, disregarding the correctness of the image content.

Nevertheless some building blocks obey the rules of scientific integrity, which renders them usable for this thesis. Especially the calculation of cut lines in the work of Summa et al. [54] is a very interesting approach for our new stitcher.

Chapter 3

Planetary Panorama Stitcher

This chapter contains the essential design of this master thesis. Section 3.1 contains the requirements on which we based our design decisions. The high-level requirements for the planetary panorama stitcher were already described in Section 2.1.

We present the actual modules in Section 3.2, along with alternative designs which are not included in the final implementation.

3.1 Requirements

In this section, we will describe the requirements on which we based design decisions. The functional criteria were defined by JR. The already defined principles of scientific integrity (Section 2.1) of panoramas will be recapitulated briefly.

3.1.1 Functional Requirements

Run-Time Complexity

As the images returned by the Mars rovers are rather small (1.4 megapixels each in case of Mastcam) and cover a small field of view, a lot of images are necessary to create a 360° panorama with considerable tilt angle. One of the biggest Mars panoramas, the "Billion-Pixel View" [38], combines 900 images to create a 1.3 gigapixel panorama. The planetary panorama stitcher has to be able to handle such large data sets within acceptable memory and time. Implications of this constraint are:

- Usage of linear ($\mathcal{O}(n)$) or at most quasi-linear ($\mathcal{O}(n \cdot log(n))$ algorithms to solve pixel-level problems. Any algorithm with higher complexity can render the workflow infeasible for large data sets. More complex algorithms can be used at image level to find e.g. overlapping image pairs.

- Try to minimise random access of memory. Once a memory region (e.g. an image or an overlap area) is loaded, it should be processed completely before loading new data. This helps memory management algorithms to recognise currently unused memory for efficiently handling memory pressure. Failure to to hold this constraint leads to unacceptable swapping and therefore run-time behaviour.

- Ability to resume partly processed panoramas. Every module should be able to store its intermediate results and load them if run with the same parameters. This saves run-time in case more images of a panorama arrive later.

Modularity and Flexibility

Apart from the scalability, the main objective is the modularity of the implementation. Splitting a problem into several sub-problems is a widely used approach in computer science. For new functionality only one module and not the whole workflow has to be changed. In addition it has the advantage of reusing components for other similar problems.

One implication of modular design is a better testability. Obviously every module can be and has to be tested on its own, narrowing bugs down to a much smaller code base. Secondly, as it is required that every module can store and load its results, testing later stages of the pipeline is much faster. All previous stages can just load the unchanged result instead of a potentially expensive recalculation.

Implementation Framework

The practical part of this thesis is implemented within the existing image processing framework of JR. It contains a lot of algorithms and workflows for image processing from basic image input and output to a fully automated 3D reconstruction pipeline. We made some design decisions to take advantage of available implementations of algorithms.

3.1.2 Scientific Requirements

In order to produce reusable results, a planetary panorama stitcher has to comply with some scientific constraints. The extended rules are described in Section 2.1. They boil down to following points [16]:

- No use of unconstrained warping
- No blending
- Traceability of each pixel into a source image
- Maintain any radiometric correction value

The workflow supports a mode to fulfil all the scientific requirements. However there exists other modes to generate panoramas for non-scientific uses, which do not obey those rules. By providing other modes we allow more possible applications of our framework. It can be extended to produce fast low resolution preview panoramas or focus on visual perception like other panorama stitchers. Some industrial processes may e.g. require blending of the images. Based on the known cut line, a later extension of this functionality should be easy.

3.2 Pipeline Design

In this section we will present the actual pipeline of the planetary panorama stitcher. Figure 3.1 shows the intermediate steps and modules.

The first, optional, step is **pointing refinement** (Section 3.2.1). In this module, the existing pointing information is refined using feature based registration in overlapping areas. As an initial pointing is known, the overlaps for calculating feature points can be approximated.

The next stage depends on the current use case. Either the input images are brought into a common point of view by **warping** them (Section 3.2.2). An estimated (or known) surface can be used for parallax mitigation. For this thesis we only implemented a simple plane as a surface model, but as mentioned any extension is possible. The input can consist of stereo pairs, then a pair wise **stereo reconstruction** (Section 3.2.3) is performed. The 3D reconstruction pipeline is already implemented in the given software framework and is not subject of this thesis.

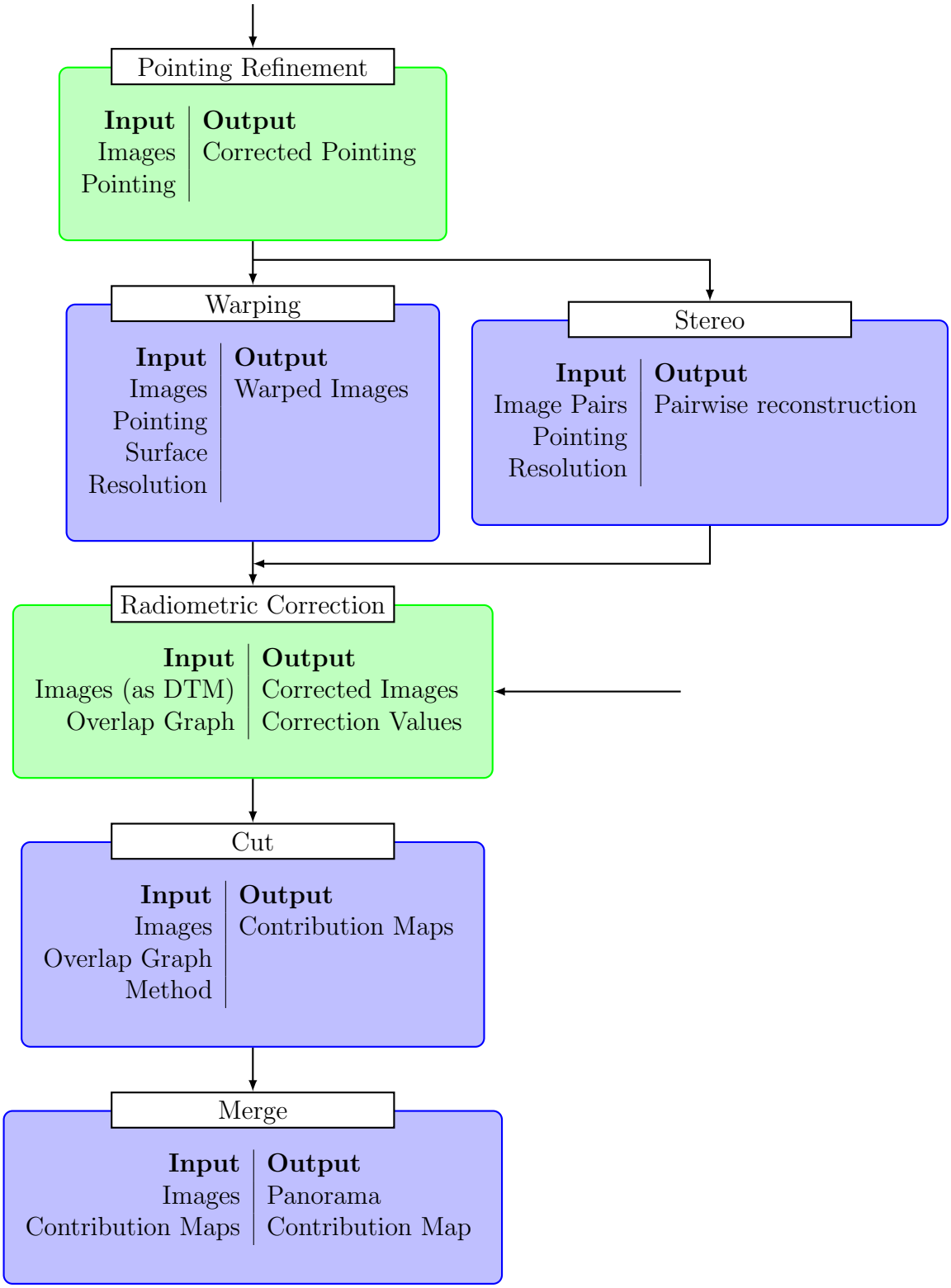


Figure 3.1: Pipeline overview. Green modules are optional. Depending on the available data, valid entry points are the pointing refinement module, or the radiometric correction module.

The **radiometric correction** (Section 3.2.4) is again an optional module. It is also a valid entry point of the pipeline, if the 3D reconstruction already was done beforehand. However, it is important, that the data arriving in this module shares the same coordinate system. All pixels in overlapping areas are superposable after warping or reconstruction. Depending on the scene and the illumination conditions, the user can decide to skip or perform this step.

When the input is corrected geometrically and radiometrically, the **cut** module (Section 3.2.5) is executed. There are several different algorithms implemented, which will be presented in above-mentioned section. The selection of the algorithms depends on the use case, heavily influencing quality and run time.

The last module produces the actual panorama by **merging** (Section 3.2.6) the single tiles together. This process is split from the cut module to permit later extensions like blending to be added independently of the cut line.

3.2.1 Pointing Refinement

The first step in this pipeline is the generation or refinement of pointing information. The scope of this thesis was only the refinement of existing camera parameters, with the extendability to complete generation of those information. As described in Section 2.5, there is an adaption of standard bundle adjustment for planetary environments by Bob Deen [17]. However, we found this method unsuitable for bundle adjustment when used without any further constraints.

The first implementation worked solely on the miss distance error and adjusted both location and orientation of the cameras. The minimiser found the trivial optimum where all rays exactly intersect in one point by moving all cameras into the same centre. In this case the miss distance is zero, but the solution has no relation to the geometric 3D structure.

In order to improve the performance of this module, we let the position fixed while still changing the orientation of cameras. This restriction still allows constrained adjustment yielding visually better results and circumvents finding the trivial minimum. It uses the advantage of miss-distance tie points of robust error measures even with glancing intersections. However it prevents proper improvements of the camera pointing information by holding the positions fixed.

Bob Deen proposes the use of a more constrained minimisation framework to use with his tie points in planetary environments [17]. Instead of handling position and orientation of each image position independently (which they are not), one can model reachable positions the rovers cameras. The six degrees of freedom are reduced to two per camera (pan and tilt angle) together with a turn radius as one global parameter, which is theoretically known beforehand. To support different rover positions, one could also model the centre and tilt of the circle of possible camera positions. Already with two cameras, there are less overall variable parameters than with completely independent cameras. Implementing this functionality was however out of scope of this thesis, but was prepared as far as possible.

To compensate the defective pointing refinement, we implemented another extension. Instead of calculating the minimal ray distance, we intersect the viewing rays with a surface model (which will be explained in Section 3.2.2. The new error metric is the distance between these intersection points on the surface model. In cases where the model approximates the actual martian surface well, this approach produces satisfying results. However, the scientific integrity is not honoured any more, because we actually adapt the data to the model while it should be vice-versa.

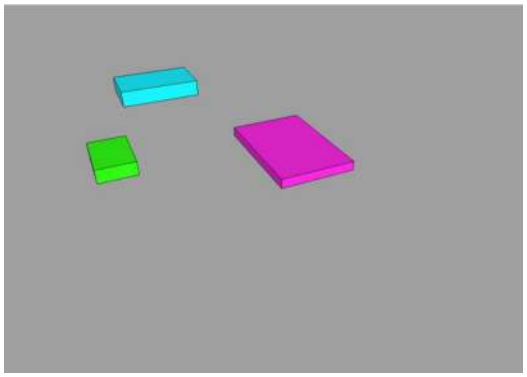
3.2.2 Surface Warping

After refining the orientation of the cameras, they do still not share the same centre of projection. If the actual 3D surface of the scene is known, a virtual image in a common point of view can be rendered. Figure 3.2 shows this process with an approximated surface on which the images get projected.

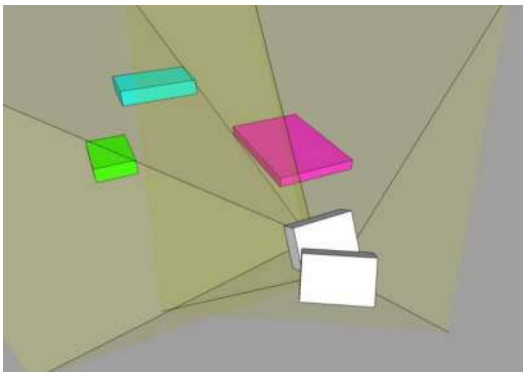
We do not have any possibility to generate a ground truth on Mars or any planet, so an approximation is needed for this step. As Mars rovers are not (yet) able to drive on too rough terrain, a plane often suits as a model. The only exceptions are crater rims and cliffs. But as the effect of parallax issues decrease fast with increasing distance, this can be neglected most of the time. Sometimes a tilted plane better reflects the surroundings [16].

Therefore we only implemented a general plane in space as a model. The plane can be given in normal form

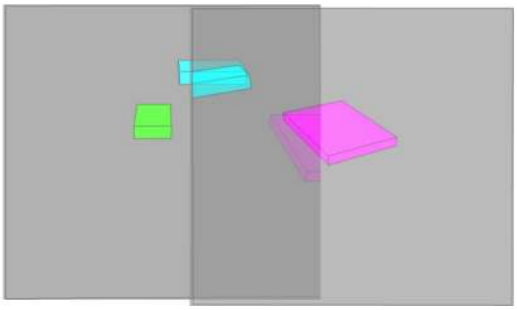
$$a \cdot x + b \cdot y + c \cdot z - d = 0$$



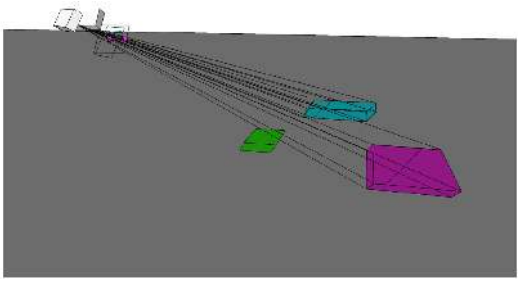
(a) Scene with several objects



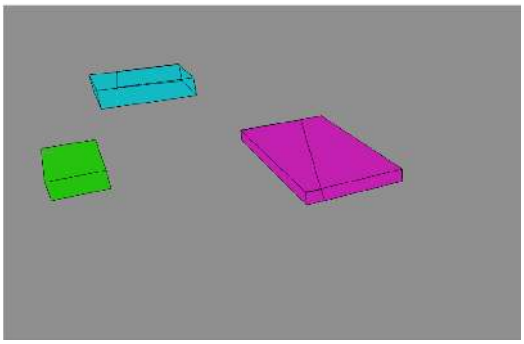
(b) Cameras not sharing a common centre



(c) Naively overlaying the resulting images



(d) Projecting the images on a surface



(e) Rendering the surface from a common point of view

Figure 3.2: A scene (a) is imaged by two cameras (b). As they do not share the same centre, stitching the images naively yields massive artefacts (c). The 3D objects all "lie" on the same surface and do not derive much from this surface. Therefore, projecting the images onto this surface (d) yields images which can be stitched much better (e). Image courtesy by Deen [16]

where a, b and c is a unit vector normal to the plane and d the minimal distance to the origin. The equation is fulfilled if point (x, y, z) lies in the plane.

To reproject an image, we first calculate its location in the panorama. This is done by projecting the four corner pixels onto the surface plane. Projecting this points pack into the panorama coordinate system yields a bounding box for the image. For the actual reprojection we use the reverse direction. Every pixel in the panorama gets projected onto the surface and back into the image. It is easier to interpolate a pixel value there instead of accumulating pixel values in panorama space. This is especially true if the resolutions differ a lot e.g. if the panorama is created as a preview.

For this system to work we need formulas to project 3D points into images, pixels as rays into 3D space and a line-plane intersection. The logical choice would be to represent all points as homogeneous coordinates in projective geometry. But unfortunately the given software environment does not support this data type. Adding this support in all already implemented camera models, transformations and projections is out of scope for this thesis.

The formula for projecting a 3D point $\vec{X} = (X, Y, Z)$ onto the image plane of a camera is

$$\begin{pmatrix} x_h \\ y_h \\ w_h \end{pmatrix} = \mathbf{KR} \cdot \left(\vec{X} - \vec{t} \right) \hat{=} \begin{pmatrix} \frac{x_h}{w_h} \\ \frac{y_h}{w_h} \end{pmatrix},$$

where $\mathbf{R}$ and $\vec{t}$ are the rotation resp. translation of the camera centre from the global coordinate system origin and $\mathbf{K}$ is the internal camera calibration. x_h, y_h and w_h are homogeneous pixel coordinates.

The inverse operation, projecting pixel to a 3D point is not possible, as the distance value is lost during projection. Taking an arbitrary distance and connecting the resulting point with the camera centre yields a ray of sight, which can be intersected with the surface.

For intersection of a ray with a plane, the ray is expressed in Plücker coordinates [26, 59] first. Such a line $L = \left[\vec{d}, \vec{m}\right]$ is defined by its direction $\vec{d}$ and moment $\vec{m}$.

$$L = \left[\vec{d}, \vec{p} \times \vec{d}\right]$$

converts a line with direction $\vec{d}$ through point $\vec{p}$ into Plücker coordinates. In fact, every point p which lies on L fulfils $\vec{m} = \vec{p} \times \vec{d}$ [59]. The canonical representation uses a unit vector as direction $\vec{d}$, s.t. $|\vec{d}| = 1$.

Now we want to calculate the intersection point $P = [\vec{p}, 1]$ of a line $L = \left[\vec{d}, \vec{m}\right]$ with plane $W = [\vec{w}, \epsilon]$. The derivation is done in [1].

Point P has to lie on plane W, therefore

$$\begin{aligned} P \cdot W &= 0 \\ \vec{p} \cdot \vec{w} + \epsilon &= 0 \\ \epsilon &= -\vec{p} \cdot \vec{w} \end{aligned} \tag{3.1}$$

.

and P also is on line L, which means

$$\vec{m} = \vec{p} \times \vec{d} \tag{3.2}$$

Using the vector triple product [28] we can derive

$$\begin{aligned} \vec{w} \times \vec{m} &= \vec{w} \times \left(\vec{p} \times \vec{d}\right) \\ \vec{w} \times \vec{m} &\overset{[28]}{=} \vec{p} \cdot \left(\vec{w} \cdot \vec{d}\right) - \vec{d} \cdot (\vec{w} \cdot \vec{p}) \\ \vec{w} \times \vec{m} &= \vec{p} \cdot \left(\vec{w} \cdot \vec{d}\right) - \vec{d} \cdot (-\epsilon) \\ \vec{p} &= \frac{\vec{w} \times \vec{m} - \epsilon \vec{d}}{\vec{w} \cdot \vec{d}} \end{aligned} \tag{3.3}$$

With the cross product matrix operator [34]

$$\vec{w}\times = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix},$$

equation (3.3) can be rewritten as

$$\begin{aligned} \begin{pmatrix} \vec{p'} \\ \delta \end{pmatrix} &= \begin{bmatrix} -\epsilon \mathbf{1} & \vec{w}\times \\ \vec{w}^T & 0 \end{bmatrix} \begin{pmatrix} \vec{d} \\ \vec{m} \end{pmatrix} \\ P &= [W\times]\, L, \end{aligned} \tag{3.4}$$

$W\times$ being the line-meet operator. p can be derived from p' as p'/δ if $\delta \neq 0$. If $\delta = 0$, L is parallel to W, therefore there is no intersection.

$W\times$ can be precalculated for the given plane, reducing the actual computation effort to a matrix multiplication.

After calculating the intersection point p, we check if the intersection is *in front* of the camera, resp. if

$$\overrightarrow{\vec{p} - \vec{C}} \cdot \vec{d} > 0$$

where C is the camera centre. In *any* real camera, every visible pixel is in front of the camera. If the intersection is not, then the ray hits the plane above its virtual horizon. As mathematically the ray expands in both direction, it intersects the plane behind the camera. This can happen if the rover looks straight ahead or up to a mountain or if the surface model is wrong. Supposing the first cases, a valid solution is to intersect such points with the plane at infinity Π_∞. Structures that far away do not cause parallax, so the influence of the surface model is negligible. For the last case, there is no "valid" solution, as the usage of a wrong model will cause a wrong result anyway.

The results of this warping module are stored as Digital Terrain Models (DTMs) (see Section 3.2.3 for a definition). While there is no actual depth value known, this provides compatibility with the stitching of 3D data explained in the next section.

3.2.3 3D Reconstruction

The stereo reconstruction is an alternative path in the panorama stitching work flow after refining camera orientations. Instead of estimating and approximating a surface model, we can reconstruct the actual surface if stereo images are available. However the surface is not needed for warping in this case, as we can reconstruct all stereo pairs with respect to one common centre.

The work flow for stereo reconstruction is already implemented in the given software environment and is not part of this thesis. For completeness, we will summarise the approach.

Stereo Reconstruction

The library does not perform a multi-view reconstruction directly, but is focused on stereo image pairs. After calculating dense disparities with Hierarchical feature vector matching (HFVM) [43], a DTM is calculated. It consists among other data of a depth image and an orthographic image.

Each pixel of the **depth image** contains the distance from a surface model at the specific location. The model can be any geometric surface such as planes, spheres or cylinders. Even more complex surfaces are possible like tunnel profiles along a path or aerofoil surfaces. A virtual grid is laid on the surface. x and y coordinates in the depth image correspond to the position on this grid, the depth value is the normal distance to the surface.

To retain colour information, each DTM contains an **orthographic image**. Using the same surface model as the depth image, it stores the respective colour value(s) of each pixel. Unlike the usual definition in literature, the orthographic view is not restricted to a projection onto a plane. Also spheres (e.g. to represent a surface on a planet) or more complex surfaces like tunnel profiles are supported.

For converting the disparities into depth values, there are currently two implemented approaches:

- Direct forward intersection uses the camera projection to calculate a ray of sight for each pixel. The distance is derived from the disparity value. The resulting point is projected onto the DTM, where all values are accumulated and interpolated.

- The locus method [6] inverts the direction of calculations similar to our warping approach. After estimating the boundaries of the DTM, each *target* pixel is projected onto a ray. On this ray, we try to find a distance which best matches the disparity in the source image. Instead of accumulating in the target, we now can interpolate in the dense disparity map.

Compositing Stereo Pairs

With the above approach only, the framework is limited to the reconstruction of two images. In order to create bigger 3D models, it is necessary to combine multiple DTMs together. Given that all information is available as images, this combination can be reduced to panorama stitching. The only condition is that all stereo pairs are reconstructed using the same model and coordinate system.

3.2.4 Radiometric Adaptions

As stated in the beginning of this chapter, the radiometric correction is an optional module. For very small panoramas without differences in illumination or externally preprocessed images there is often no correction necessary. Like the stereo reconstruction in Section 3.2.3, this module was already part of the given software framework. For sake of completeness, we will explain its functionality here.

The algorithm tries to globally minimise the brightness difference of neighbouring images. The error measure is solely based on the average brightness of an image. While this works well if all images show similar textures, it fails if they are very different. If one image contains parts of the sky and parts of a bright rock, its average brightness is not descriptive enough for a good correction. As pictured in Figures 3.3, the bright rock on the right causes the radiometric correction to darken the respective image. The middle image get brightened correctly, rendering the left seam nearly invisible.

Despite the problems with certain scenarios of the current solution, we decided against implementing a new radiometric correction module. Due to the modular design, a new implementation can substitute the current module.

3.2.5 Cut Line Selection

As already stated in Section 2.6, the cut line selection has a major influence on the final result. We implement various approaches for this calculation to serve as many use cases as possible. Most of those algorithms are described in Chapter 2. In this section we will present each implemented method.

Each algorithm has a binary region for each panorama tile as output. This regions store whether or not a pixel of is used in the final panorama. As blending is excluded in this thesis, those regions must not overlap before merging.

Images are always processed in order of input. In the rare cases in which the processing order matters, changing the input files is the only available option.

Following algorithms will be explained with help of two overlapping images illustrated in Figure 3.4. Green denotes areas belonging to image 1, blue means image 2.

(a) Three images stitched without radiometric correction. A change of illumination or exposure is clearly visible as radiometric seam between the left and middle area.

(b) After radiometric correction is applied on Figure 3.3a, the seam on the left vanishes. The textures are very similar and in reality equally bright so the correction factor is correct. The flat rock in the right part is very bright, inducing wrong darkening.

Figure 3.3: Impact of current radiometric correction algorithm. The image-wide brightness average is clearly not a suitable error measure. Image courtesy by NASA/JPL/JR

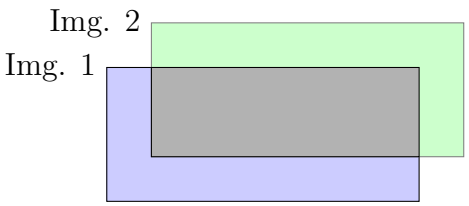


Figure 3.4: Two overlapping images. The grey area must be processed to belong either to image 1 or image 2.

Average

The first implemented approach on cut line calculation is merely used for showing the quality of image registration in overlapping areas. This algorithm does not trim anything from any image tile, therefore leaving the work for the merge module. As described below in Section 3.2.6, any pixel with multiple contributions will be assigned with the average colour value.

As seen in figure 3.5, misalignment errors can be seen clearly as semi-transparent doubling artefacts.

Ordering

The easiest cut line algorithm (apart from the average approach) is to use either of the images as a whole. This approach is used by Bob Deen [17] for processing MSL images at JPL/NASA. It is the only algorithm in this thesis which depends on the input ordering. For sake of simplicity, we always use image 1 (Figure 3.6). The only way to use the other image is to re-order the input by swapping image 2 and image 1.

The big advantage of this approach is the fast calculation. In terms of binary regions, one image is simply subtracted from the other. A use case which results from the computation speed is a fast in-situ panorama generation (on Earth). Photographers can produce low-resolution panorama previews to check if images are missing or if there are severe problems with e.g. exposure. A panorama consisting of 100 images, with 1.5 Megapixel each, only takes some seconds to compute. More detailed performance comparison will follow in Chapter 4.3.

Figure 3.5: The Average approach does actually not find a cut line, but merely overlays the images. The results are unusable for further science, but are used to show the quality of image registration. Image courtesy by AMASE/JR

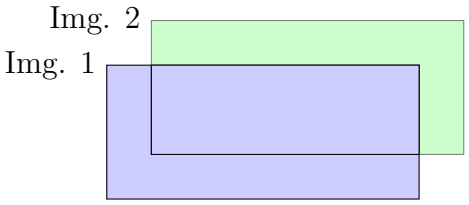


Figure 3.6: Ordering cut line. The overlap area is always assigned to image 1.

Nearest Centre

Another idea for defining cut lines is to assign each pixel in the overlapping area to the image whose respective centre is nearest. Figure 3.7 shows the effect of such an algorithm. The intention behind this approach is, that if there are inaccuracies during image acquisition, pixels near border are more affected. This effect is mostly due to missing or inaccurate lens distortion parameters and vignetting. In some cameras, as Mastcam left on the MSL rover, the border pixels show artefacts (Figure 3.8).

As seen in Figure 3.7, this technique can still have the image borders as cut line. However the affected area is covered by another overlapping image in most real-life scenarios.

Dijkstra

All above presented algorithms define their cut lines solely based on image geometry. In order to improve the results, the image content has to be taken into account.

The pairwise cut line is calculated with the Dijkstra algorithm as described in Section 2.6.4. Figure 3.9 shows an example result. There are different measures which can be used as foundation for edge weights. We implement edges, colour and depth measures while the latter two can be linearly combined.

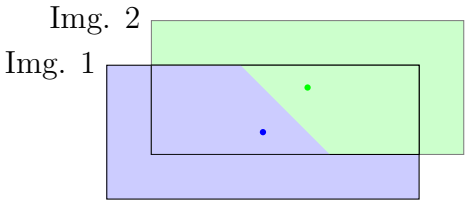


Figure 3.7: Nearest centre cut line. Each pixel is assigned to the image whose centre (green resp. blue dot) is nearest.

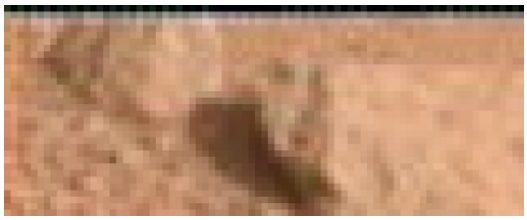


Figure 3.8: Magnified detail of the top left corner of a Mastcam image. Especially the top border shows a repetitive pattern which is clearly an artefact. Image courtesy by NASA/JPL

The method *cut on edges* is used in literature (e.g. [4]) and has been used previously at JR. Cutting an image on natural edges seems to be a good choice. Remaining radiometric errors do not stand out, as there is a disruption anyway. However for this method to work, the images have to be aligned very well. The effect only works if there is an edge in both images at the same position in the overlapping area. If parallax errors or misalignment are too large, the algorithm cuts on one of the corresponding edges or between them. This does not hide, but highlight the error.

Another widely used method is to *cut on equal colour*. As described in Section 2.6.4,

$$E_{colour}(p,q) = \|I_1(p) - I_2(p)\| + \|I_1(q) - I_2(q)\|$$

can be used as error measure in this case. $E_{colour}(p,q)$ is the cut cost for cutting the image between neighbouring pixels p and q. $I_1(x), I_2(x)$ is the image intensity or colour value at location x.

The Dijkstra algorithm follows paths where the colour difference between two overlapping neighbouring pixels is as low as possible. This leads to unobtrusive cut lines in the texture image. The already available implementation of the Dijkstra algorithm fits perfectly to this use case. It uses 8bit values to represent edge cut costs. As shown in Equation 17, we use the euclidean distance between colours. Like the rest of this thesis, the calculation of the edge costs can be adapted easily.

Both presented approaches work solely on the orthographic image. Reconstruction errors in the depth data are completely ignored in favour of seamless colour images. In order to take account of this additional information, we introduce a *cut on equal depth*. Instead of calculating the edge costs from colour differences, they are derived from depth. This corresponds to using $D_1(x)$ resp. $D_2(x)$ namely the depth values at location x instead of colour values $I_1(x)$ in Equation 17. In contrast to colour depth data is a floating point value. Before converting depth differences into costs, they have to be fit to a fixed integer scale. The order of magnitude can vary from planetary scale

with thousands of kilometres to microscopic scale for MAHLI, depending on camera and reference coordinate system. As a heuristic, we limit the maximum absolute depth difference so that 90% of the depth values lie within the interval. This removes sensibility to outliers during reconstruction and represents the depth deviation well.

Both *cut on equal colour* and *cut on equal depth* can be linearly combined in order to adapt to the use case. Depending on the importance of the structure versus the texture, the user can decide which one to give precedence. The final cost is calculated as

$$E_{combined}(p,q) = x \cdot E_{depth}(p,q) + (1-x) \cdot E_{colour}(p,q),$$

where x is a user defined value between 0 and 1.

If the result is triangulated and viewed in a 3D viewer for further analysis of geological structures, preventing jumps in depth is more important than the texture. On the other hand, the structure could only be a rough approximation for visualization while the texture contains important data. Then structural errors can be tolerated in favour of good textures.

There is however no possibility to cut depth images and texture images independently of each other. While this would yield seamless results in both domains, scientific integrity would be violated severely.

Combining pairwise results Summa et al. propose a way to merge multiple pairwise cut lines into one global solution. Figure 2.22 in Section 2.6.4, shows the initial neighbour graph and its reduction. We cannot use this method for our workflow for several reasons.

First, maximal clique finding is NP-complete. While this is no problem for small scale panoramas, our framework has to be capable of handling thousands of images. Spending too much time merely to find overlap candidates is not acceptable.

Another problem is a limitation mentioned in [54]. The current algorithm is not capable of resolving overlap situations as depicted in Figure 3.10a. In addition, an image completely covered by another image is not supported.

To overcome those limitations, we propose the following approach. Instead of processing faces in the overlap graph (see Figure 2.22), we process each image with its neighbours independently. Parallelisation is not yet implemented, but prepared as a future extension.

See Algorithm 1 for a pseudo code description of the approach. The first part of the while body (Line 2f) defines the two images to work on. The current image is I_1, its next-to-process neighbour is I_2. I_2 is always the neighbour with the highest overlap. This decreases the influence of image ordering. The biggest overlap has the most influence on the result while also having the highest degree of freedom for the cut line.

The Dijkstra algorithm returns regions R_1 and R_2, which have to be removed from I_1 respectively I_2. Neighbours of I_1 which only overlap in region R_1 are automatically removed by updating the neighbour graph in Line 9. We calculate $I_{1_{only}}$ and $I_{2_{only}}$ in Lines 10 and 11. Those variables contain the regions of both images which are unaffected by the cut. Neighbours of I_1 which overlap in region R_2, but not in $I_{1_{only}}$ (red area in Figure 3.10a) are removed as invalid. The loop starting in Line 17 handles opposite cases as seen in Figure 3.10b.

Algorithm 1 Pseudo code of our proposed cut line combination

```
1:  while current image has neighbours do
2:      I_1 ← current image
3:      I_2 ← find neighbour of I_1 with maximum overlap
4:      calculate cut line between I_1 and I_2
5:      R_1 ← region to remove from I_1
6:      R_2 ← region to remove from I_2
7:      remove R_1 from I_1
8:      remove R_2 from I_2
9:      update neighbour graph
10:     I_1_only ← I_1 \ R_2
11:     I_2_only ← I_2 \ R_1
12:     for all I_n ← neighbours of I_1 do
13:         if I_n does not overlap with I_1_only then
14:             remove R_2 from I_n
15:         end if
16:     end for
17:     for all I_n ← neighbours of I_2 do
18:         if I_n does not overlap with I_2_only then
19:             remove R_1 from I_n
20:         end if
21:     end for
22: end while
```

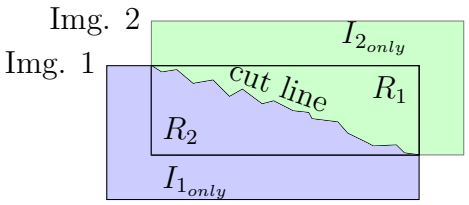
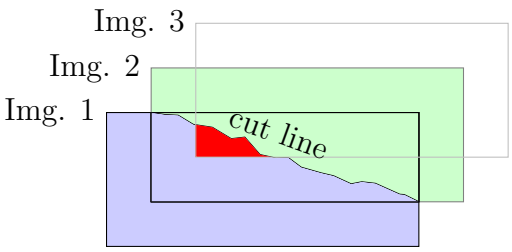
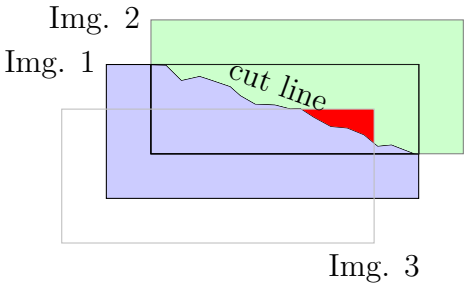


Figure 3.9: The cut line of two highly overlapping images. R_1 denotes the area which gets removed from image 1 and is limited by he overlap area. $I_{1_{only}}$ is the area of image 1 which is unaffected by the cut line. R_2 and $I_{2_{only}}$ are defined analogously.



(a) A third image overlapping with image 1 in the area which was clipped from image 2 results in invalid overlap entries



(b) A third image overlapping with image 1 in the area which was clipped from image 2 results in invalid overlap entries

Figure 3.10: This figure shows special cases which can occur in highly overlapping panoramas. These cases need to be taken care of in the algorithm.

Other cut ideas

We had several other ideas for defining and calculating cut lines. Due to their poor results, they were dismissed and removed from the final solution.

Most notable idea was an inverse of the *nearest centre* approach. Instead of using the minimal distance to the image centre, we used the maximal distance to any image border. While this is interchangeable behaviour for 2D images, it makes a big difference in DTMs. During reconstruction there are often occlusions which lead to undefined areas in depth and texture images. Our *furthest no-data* approach calculates the distance to the nearest undefined pixel, which is stored as a special no-data value. During cut line calculation, the pixel with the largest distance is used.

Data near undefined pixels often has above-average reconstruction errors. The *Furthest no-data* algorithm should minimise the impact of those errors. In reality this idea does not work very well. Occlusions in one image pair often occur in neighbouring image pairs in near regions. Slight differences which pixel is nearer the occlusion propagate through the image, leading to alternating stripes of contribution.

The calculation of the furthest no-data pixel is more computationally intensive than the *nearest centre* approach while yielding worse results. Therefore this algorithm was cancelled from the thesis.

3.2.6 Merging

The last step in our panorama stitching workflow is to merge all single tiles together into one image. If the tile boundaries do not overlap, this is a straight-forward process. Every image is just copied into its place in the final panorama. Overlapping tiles only occur in our pipeline when the "Average" approach is used in the cut-line module. The current implementation just averages all contributions to one pixel. In all other cases, there is only one value to average per pixel, producing the original value.

The final panoramic image is created in this module. Therefore it is responsible to add any available metadata to the file, like the used projection, resolution, units, coordinate system, etc. In addition to image data and metadata, every tile also includes the contribution map produced in previous modules. During the merge process, the two channels of this map (X and Y coordinate) are augmented with a third index channel. The contribution map is stored alongside the depth and orthographic images.

Tracing back a pixel is therefore reduced to a pixel lookup in this map instead of redoing all projection steps. The link between image index and file name is stored in a separate file.

Despite the small task, we implemented it as separate module for sake of extensibility and flexibility. Depending on future use cases this module can be adapted to do more sophisticated merging. One possibility is to introduce feathering [11] between image borders.

Chapter 4

Evaluation

We evaluate the implemented framework using three different data sets by comparison with other established panorama stitching tools. There exist approaches to automatically benchmark the image quality of the panorama [50]. The implementation of this framework, including verification of its functionality, was however out of scope of this thesis. We logged data like run time and memory consumption during stitching. The stitching quality was determined by manually screening for artefacts.

Section 4.1 contains a description which data sets were used and why. We present the results of our stitching framework in Section 4.2. The comparison to other panorama stitching tools is contained in Section 4.3.

The test environment was a Windows 7 workstation with 8GB main memory and an Intel i5-2500 CPU with 4 cores running at 3.3GHz. Apart from the document storing the results and the ProcessExplorer [53] no applications were run. We used ProcessExplorers "Peak Private Bytes" as a measure for memory consumption. This parameter reflects the maximum amount of memory which the process actively allocated during run time. The run time was measured manually for third party programs or automatically by the execution framework. It reflects the pure processing run time without configuration before and (in case of third party programs) during the stitching process.

4.1 Data sets

To get an as good as possible impression of the frameworks abilities, we chose three completely different data sets. Table 4.1 lists each set along with reasons for using it for evaluation.

Name	# Images	MP	Taken by	Reason for inclusion
Sol318	99	158	MSL Mastcam	Large planetary data set ("standard case")
Tunnel	4	$2 \cdot 23$	Handheld stereo rig	3D Data in depth images
Simulator	26	27	AUPE	Large parallax in data, compatibility with future Mars missions

Table 4.1: Metrics of the used data sets. MP denotes the overall megapixels of the input images.

Throughout this section all figures were created using the *Average* approach (see Section 3.2.5). As a result of the averaging, the images give the impression of being semitransparent in overlapping areas. While being unusable as final product, it shows the quality of the raw data set.

4.1.1 Sol318

We used a typical martian landscape as seen in Figure 4.1 as a standard data set for our framework. It was taken by MSL with Mastcam right and consists of 99 images, 9 rows with 11 images each. Testing on a large data set is necessary in order to ensure long-term usability.

Improving data transmission enables higher image resolutions or more images. The framework should be able to cope with more data without an exorbitant cut of performance. The reason why we did not include the aforementioned huge data sets of up to 1000 images is simple. Those sets are not taken in one sweep but over a time period of several days or weeks [38]. Due to hardware restrictions we used a more manageable but still relatively large data set for testing and evaluation.

4.1.2 Tunnel

A typical case of Earth-based industrial image processing at JR is tunnel reconstruction. Figure 4.2 shows the texture image, the depth image and a 3D rendering of the working face of a tunnel construction site. It consists of the 3D reconstruction of 4 stereo image pairs. This data set evaluates the ability to handle depth data. Our stitcher has to find a cut line according to the texture, the depth or a combination.

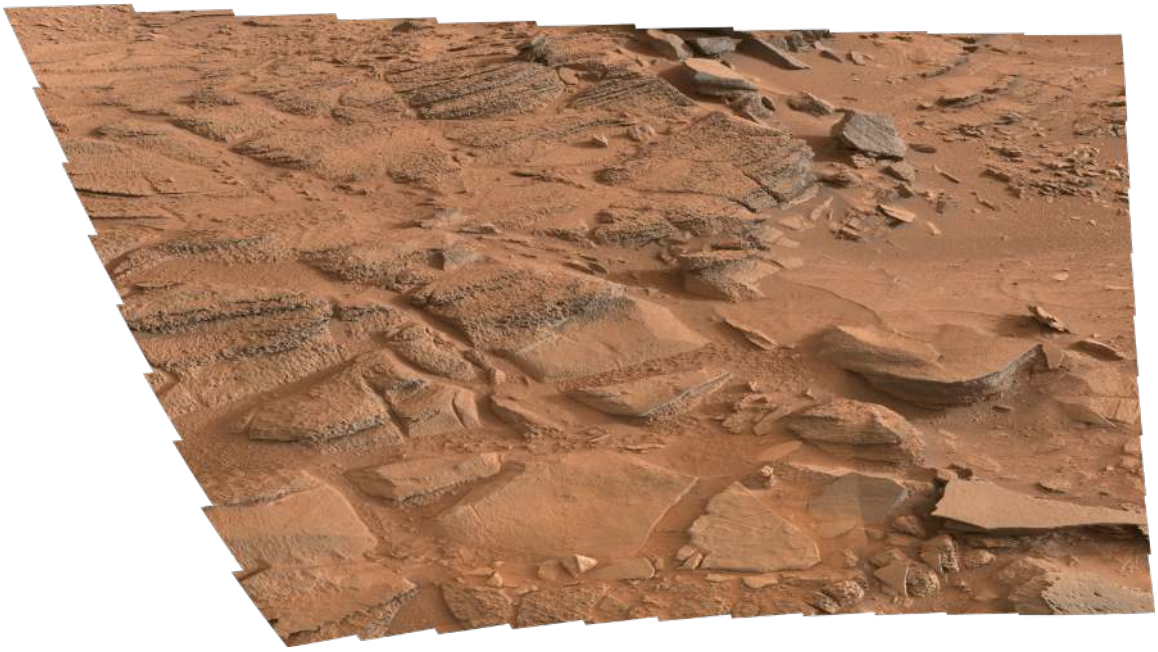
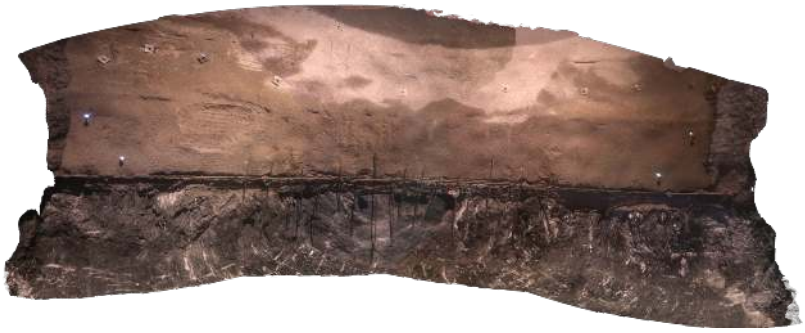


Figure 4.1: Martian landscape photographed by MSL

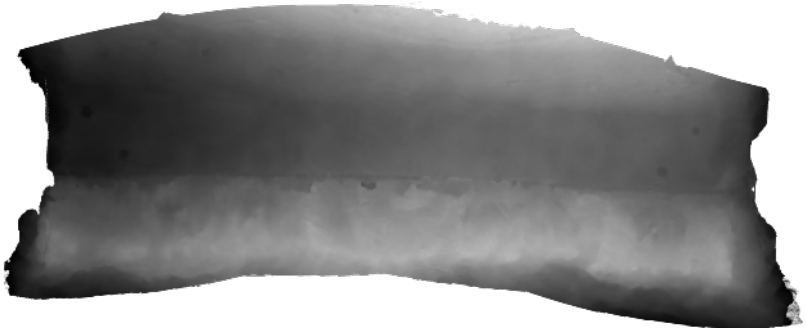
An additional difficulty of this data set is the fuzzy overlap of the pair wise reconstructions. Local matching variations lead to overlap situations way more complex than shown in Figure 2.16b in Section 2.6.

4.1.3 Simulator

In order to evaluate compatibility with future Mars missions, we also included a data set from the Arctic Mars Analog Svalbard Expedition (AMASE) [5] expedition 2013. These expeditions has the goal to test technologies such as cameras, sensors or whole vehicles. Svalbard has a lot similarities with Mars, it has a rocky landscape, low temperatures and no vegetation. The data set shown in Figure 4.4a was taken by Aberystwyth University Pancam Emulator (AUPE) [48], the simulator of ExoMars' panoramic camera (PanCam) (see Figure 4.3). The biggest difference of AUPE to MSL's Mastcam is a higher base line between the cameras. On the one hand it allows 3D reconstruction from higher distances. When imaging near objects on the other hand, even more parallax occurs as seen in Figure 4.4b.



(a) 2D projection of tunnel surface onto a tunnel profile raster (profile arc length vs. tunnel axis direction). Rock excavation surface at bottom, shotcrete protection with anchors and steel arches at top.



(b) The depth is coded directly into the grey value, brighter corresponds to bigger distance. Basis of the depth measurement is a cylindrical model.

(c) A 3D rendering of the tunnel. The working face is visible in the inside. Immediately before it is the raw tunnel surface as excavated, without shotcrete applied, and therefore has a slightly higher diameter and a rougher surface.

Figure 4.2: Tunnel data set Image courtesy by DIBIT/JR

Figure 4.3: AUPE image acquisition on Svalbard Image courtesy by Kjell Ove Storvik

(a) Challenging data set from Burovtoppen, Svalbard. The large rocks introduce a lot of parallax to handle. AUPE's wide base line increases this effect.

(b) Detailed view on parallax

Figure 4.4: Data from the panoramic camera simulator Image courtesy by JR

4.2 Results

We will present the results of our new planetary stitcher run with different settings, utilising all implemented functions. To reduce redundancy in the test results we did not run every possible combination of input parameters. This section contains a selected subset of the tests showing the effects of implemented features in different modules.

It is in the nature of panoramas that they are much bigger than standard images, rendering it impossible to identify stitching artefacts if printed in whole. Therefore we only show regions of interest when comparing approaches.

4.2.1 Pointing Refinement

We can confirm [16] that more accurate pointing information can increase the quality of the final output a lot. Figure 4.5 shows a detail of Sol318, stitched with the *Average* approach to visualise the accuracy of the images. Figure 4.5a uses raw pointing information delivered by the rover. A shift in horizontal direction is visible as blur in the overlap area. After pointing refinement the overlap area in Figure 4.5b is not distinguishable anymore.

In the remaining evaluation sections, pointing refinement was turned off to better bring out the effects of other modules.

4.2.2 Radiometric Correction

Despite the fact that we did not implement the module for radiometric correction, but merely adapted it to the new interface, we include it in the evaluation. As already shown in Section 3.2.4, the current approach has some difficulties with some scenarios. Figure 4.6 shows such an example.

However, the main goals of this thesis was to create a modular and flexible framework with emphasis on the seam-based approach, and ability to process huge data sets With the goal of flexibility achieved, it is easy to adapt or substitute this module later with a more sophisticated approach.

(a) Image detail stitched with raw pointing information. A shift in horizontal direction is clearly visible as blurred area with duplicated structures.

(b) The same detail after pointing refinement. The sharpness of the overlap area indicate good alignment of the images.

Figure 4.5: Effects of pointing refinement. Image courtesy by NASA/JPL/JR

(a) Three images stitched without radiometric correction. A change of illumination or exposure is clearly visible as radiometric seam between the left and middle area.

(b) After radiometric correction is applied above, the seam on the left vanishes. The textures are very similar and in reality equally bright so the correction factor is correct. The flat rock in the right part is very bright, inducing wrong darkening and creating a seam on the right side.

Figure 4.6: Impact of current radiometric correction algorithm. If the brightness is similar, it works. For big differences, this approach worsens the result. Image courtesy by NASA/JPL/JR

4.2.3 Ordering and Nearest Centre Cut Lines

The cut methods Ordering and Nearest Centre mainly focus on speed. Figure 4.7 shows a detail of the Sol318 data set. These methods are unable to fix or even mitigate parallax errors and inaccurate pointing as seen in Figure 4.8. They are however the easiest approach to fulfil all requirements of scientific integrity.

Furthermore these modules can be used for fast panorama preview generation. During image acquisition with handheld cameras or mounted cameras like the AUPE simulator it can be necessary to run fast sanity checks on the images. A low resolution panorama with roughly 1 megapixel for the Sol318 data set completed in about 10 seconds. Most of this time is spent for loading the (full resolution) input and some basic analysis before the low resolution comes into play. Such previews are unusable for scientific analysis, but effectively show missing frames or faulty exposure times.

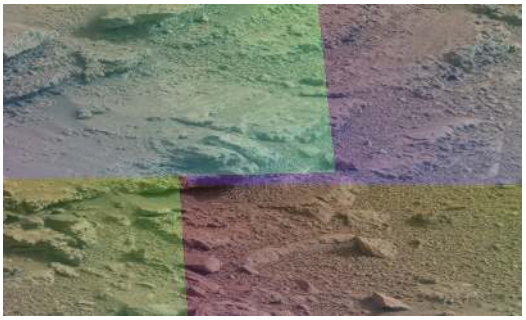
4.2.4 Dijkstra

We ran most tests with the Dijkstra cut line module enabled. It is the most advanced module and by time of writing already used in production. Figure 4.9 demonstrates its effects in a drastic way. The existing structures in the Simulator data set allow the nearly optimal hiding of parallax errors. Admittedly, the geometry is not correct along the cut lines. But as we have the contribution map, scientific integrity is preserved due to the ability to go uniquely trace back each pixel into the respective input images. In addition, the images are visually acceptable which makes their analysis easier.

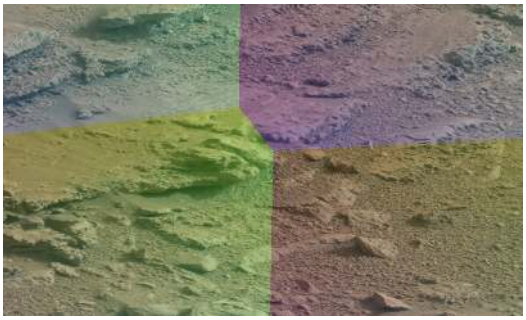
Responding not only to colour, but also on depth differences is another important ability of the Dijkstra module. Figure 4.10 shows the effects of changing the strategy.

4.3 Comparison

In order to justify the development of a whole new panorama stitching framework, we now compare our stitcher with other systems. On the one hand, the former solution at JR, *JR Warping*, is part of the comparison in this thesis. On the other hand publicly available panorama stitching solutions Image Composite Editor (ICE) and Hugin are included.



(a) Cut lines produced by Ordering. While having reproducible and traceable straight cut lines, the contribution map is unnecessarily fragmented.



(b) The Nearest Centre approach yields much shorter cut lines while being equally fast in computation.

Figure 4.7: A detail view of Sol318 with the contribution map overlaid. The contribution map is colour coded and contains the image index in channel red and the source coordinates of each pixel in channels green and blue. Overlaying it temporarily with the final panorama makes it very easy to distinguish between artificial and natural artefacts.

Figure 4.8: Doubling artefacts due to inaccurate pointing

(a) Detail of simulator data set with Average module

(b) Detail of simulator data set with Dijkstra module

(c) Detail of simulator data set overlaid with contribution map

Figure 4.9: Mitigation of Artefacts using Dijkstra.
Image courtesy by NASA/JPL/JR

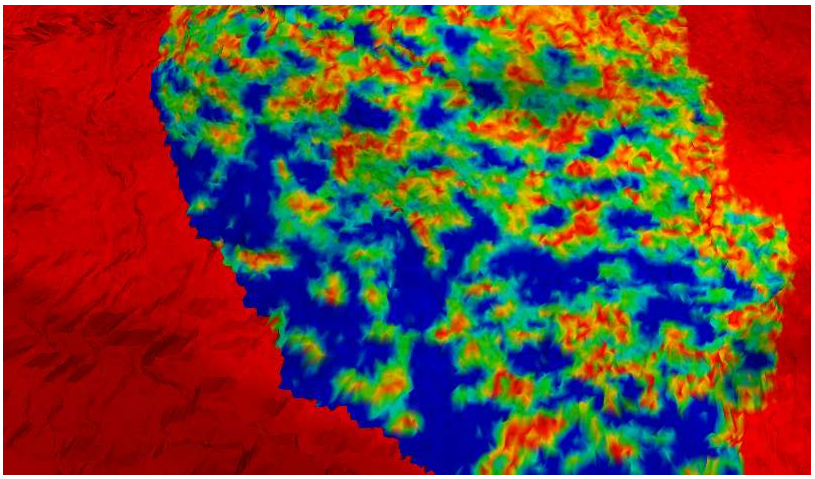


Figure 4.10: This figure shows the difference of two 3D models of the tunnel. One model was created by cutting based on colour differences, the other based on depth. The distance of a vertex in one model to the nearest vertex in the other model is colour coded. Red regions are identical, blue indicates discrepancies. Cutting by colour yields the hard cut from red to blue on the left side. This indicates a jump in height in the model. The cut on the right hand side was calculated from depth. Instead of jumping, the difference to the original surface is changing gradually which is a sign of good depth agreement.

4.3.1 JR Warping

As the name suggests, this solution of JR tries to find a solution to the stitching solution by warping. With HFVM [43], it densely matches overlapping areas and applies local distortions and blending to improve the visual perception. When configured correctly, it produces appealing results, as seen in Figure 4.11. However, the devil is in the details. Figure 4.12a shows a small detail with some obvious distortions. There are a lot of artefacts like this distributed over the whole panorama. Due to the lack of traceability, there is no way to distinguish natural from artificial artefacts.

In addition to the panorama quality, there are some performance issues when using the JR Warping approach. Table 4.2 shows the memory consumption and run time for each tested configuration. JR Warping has both by far the highest memory requirements and takes longest to compute the panorama.

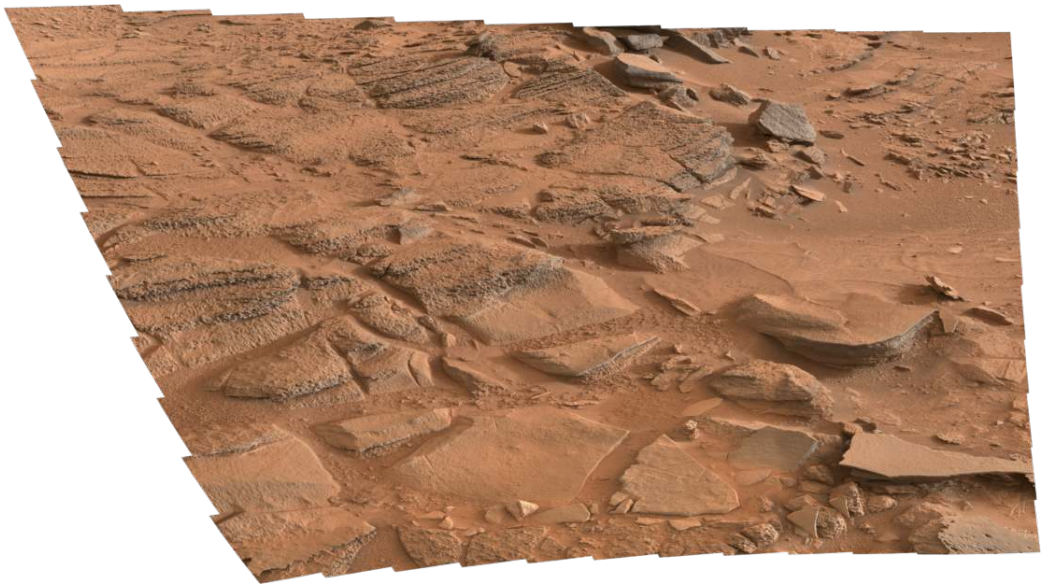


Figure 4.11: JR Warping approach. The resulting image is a seamless panorama with no major artefacts. Radiometric seams are not visible due to blending. Image courtesy by NASA/JPL/JR

(a) Detail view of Figure 4.11. Such artefacts are visible all over the image. Image courtesy by NASA/JPL/JR

(b) Detail view of Figure 4.11 but stitched by the Dijkstra module with the contribution map overlaid: the whole region was taken just from a single input image. This shows that the artefact in Figure 4.12a is indeed artificial. Image courtesy by NASA/JPL/JR

Figure 4.12: Example of JR Warping artefacts. The scientific integrity of our new framework allows to decide whether or not an artefact could be artificial. As there is no cut line near the region of interest, the panorama content correlates directly to an original image which does not show the artefact.

4.3.2 ICE - Image Composite Editor

ICE is an image stitching tool developed by Microsoft Research [36], freely available for Windows users. It focuses on a broad user base and is therefore easy to use for everyone. Most settings are detected automatically with a limited parameter set exposed to the user. The default settings support a wide range of scenarios but when it comes to special needs in planetary environments the configurability is limited.

It is the fastest and most memory-efficient program tested, as seen in Tables 4.2-4.4. Its results are surprisingly good, given the low run times. Only the high amount of parallax in the Simulator data set introduces noticeable doubling effects (see Figure 4.13).

The major disadvantage of ICE is the lack of scientific integrity as well as the support for 3D data. Due to the limited applications it is unlikely that these features will be added by Microsoft. The program's sources are not available which makes external contributions impossible.

4.3.3 Hugin

Hugin [14] is a mature open source program for image stitching. Its configurability is very high, rendering its usage more difficult for first-time users. On the other hand it allows experienced users to adjust parameters for better results.

The high portion of manual interaction makes it difficult to compare to automated solutions like our planetary stitcher or ICE. However while running the test we found out that the actual problem of Hugin is not the steep learning curve but its inability

Figure 4.13: A detail of the stitched Simulator data set. While the overall perception of the panorama is good, the rock on the top left shows severe problems. This artefact is clearly visible as stitching error, but taints the confidence of the remaining panorama.

to handle regions with no data available. Figure 4.14 shows its output when forcing Hugin to process the whole panorama. But even if we let Hugin allow to select the maximum output size, the results are not satisfactory as seen in Figure 4.15.

Hugin produced results for Sol318 and Simulator, but failed completely to generate a panorama for the Tunnel data set. Main cause is probably the difficult overlap configuration of those images.

In contrast to ICE, Hugin is open source, which allows contributions by external persons/companies. However, it is unfeasible to adapt the whole design of Hugin to the requirements of planetary environments. In addition the run time of Hugin is high compared to others (see e.g. Table 4.2).

4.3.4 Performance and Interpretation

The previous sections already contained references to the following performance comparison. We ran the data sets with different settings and programs and measured their peak memory consumption and run time. Tables 4.2-4.4 show the results. To reduce redundancy, we start from a base test case and activate single features one at a time. This reveals the individual influence of each feature.

The results show that for most settings, our framework has the lowest run time. Only Microsoft Research's ICE is faster compared to our Dijkstra approach.

Memory consumption is high compared to third party programs, but *much* lower than the former JR internal solution (see test cases 6 and 7 in Table 4.2). The main cause for the high memory utilisation is the maintenance of the contribution map which is a unique feature to comply with an important requirement of the planetary application context. The difference between test case 1 and 2 (both in Table 4.2) shows an increase of more than 50%. Even without contribution map, the memory footprint is higher than ICE, although it has the same order of magnitude.

As seen in Table 4.1, the overall input size roughly triples between Sol318 and Tunnel. This includes texture and depth images. The comparison of tests 7 (Table 4.2) and 15 (Table 4.3) shows an excessive increasing memory consumption of factor 8. Our framework (e.g. test 6 in Table 4.2 versus test 14 in Table 4.3) reflects the tripling nearly linearly.

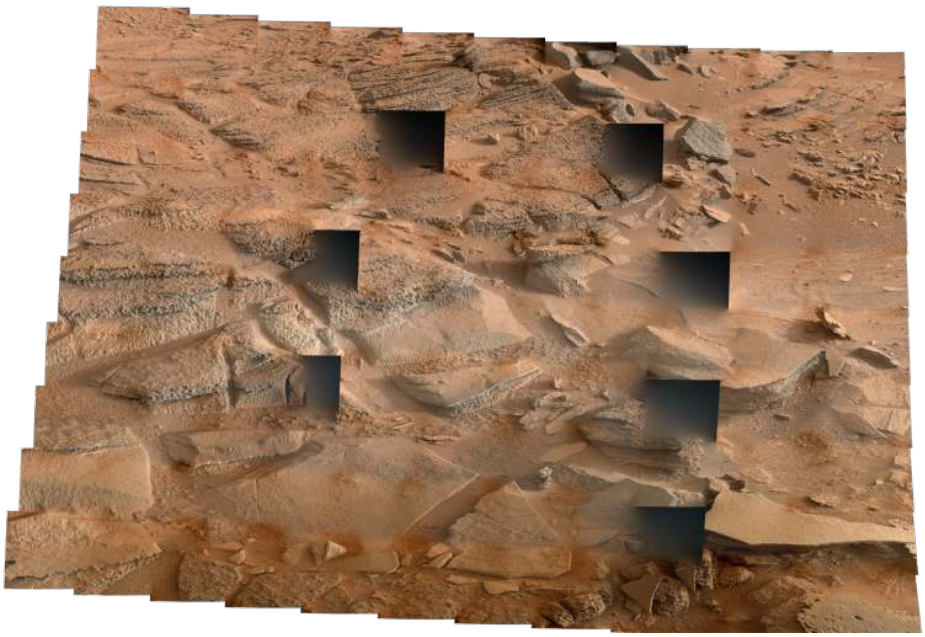


Figure 4.14: Hugin restricts its output to a fully defined region per default. If forced to include all data, it fails completely to produce a usable result. Interesting is that the areas affected by errors are not necessarily near the border.

Figure 4.15: Even without undefined areas, the output of Hugin has many radiometric variations,

Test 4 in Table 4.2 shows the scalability towards lower resolutions. After initially loading the input images into memory, the run time and memory consumption is solely defined by the output size. This is a useful property for checking the completeness of a data set after acquisition or download.

Hugin automatically calculates camera orientations and radiometric correction factors before panorama creation. For better comparison with frameworks with given camera orientations, test 8 in Table 4.2 shows the run time *after* this initial step. Test 9 includes the pointing refinement phase.

No.	Resolution	Pointing Refinement	Radiometric Corrections	Contribution Map	Stitching Method	Memory Consumption in MB	Run time (hh:mm:ss)
1	100%	No	No	No	**Average**	3 057	00:01:42
2	100%	No	No	Yes	**Average**	4 848	00:02:04
3	100%	No	No	Yes	**Ordering**	4 794	00:02:02
4	1%	No	No	Yes	**Ordering**	876	00:00:11
5	100%	Yes	No	Yes	**Average**	4 837	00:08:07
6	100%	No	No	Yes	**Dijkstra**	4 805	00:12:07
7	100%	No	No	N/A	JR Warping	26 168	01:49:09
8	100%	No	No	N/A	Hugin	1 526	00:17:45
9	100%	Yes	Yes	N/A	Hugin	1 526	00:38:55
10	100%	Yes	Yes	N/A	ICE	2 295	00:04:15

Table 4.2: Test settings and performance results for data set Sol318.

No.	Resolution	Pointing Refinement	Radiometric Corrections	Contribution Map	Stitching Method	Memory Consumption in MB	Run time (hh:mm:ss)
11	100%	No	No	Yes	**Average**	1 143	00:00:08
12	100%	No	No	Yes	**Dijkstra Colour**	1 496	00:00:34
13	100%	No	No	Yes	**Dijkstra Both**	1 497	00:00:33
14	100%	No	No	Yes	**Dijkstra Depth**	1 496	00:00:38
15	100%	No	No	N/A	JR Warping	2 910	00:14:23
16	100%	Yes	Yes	N/A	ICE	523	00:00:16

Table 4.3: Test settings and performance results for data set Tunnel. Hugin failed to produce an output for this data set.

No.	Resolution	Pointing Refinement	Radiometric Corrections	Contribution Map	Stitching Method	Memory Consumption in MB	Run time (hh:mm:ss)
17	100%	No	No	Yes	**Average**	2 358	00:00:53
18	100%	No	No	Yes	**Dijkstra**	2 400	00:02:54
19	100%	Yes	Yes	N/A	Hugin	277	00:04:22
20	100%	Yes	Yes	N/A	ICE	606	00:01:13

Table 4.4: Test settings and performance results for data set simulator.

Chapter 5

Conclusion

5.1 Summary

In this thesis we designed and implemented a flexible and extendible framework for planetary panorama stitching on top of the JR internal image processing library. We divided the process into independent modules, each with well defined tasks and interfaces. Some of our new modules has already been integrated into other workflows at JR – even for industrial production purposes – at the time of writing. This confirms the creation of reusable components.

While some modules only received a basic implementation to show functionality of the overall pipeline, the cut line module contains a lot of different methods. We evaluated different ideas in Section 2.6 and selected an approach similar to Panorama Weaving by Summa et al. [54] for the more advanced configuration. The rest of the framework was partially based on separate stages of panorama stitching defined by Szeliski [55] presented in Section 2.3.

Our framework presented in Chapter 3 is able to maintain scientific integrity of panoramas if needed. A contribution map makes it possible to determine the source image and location within this image of each pixel in the final panorama. This feature enables geologists to look up other related available data such as different spectral ranges for further investigations.

It is due to this traceability that our framework is usable for other application besides planetary environments. Industrial systems need a continuous documentation of all involved processes for quality management. Evaluation shows that our panorama stitcher can be used without modifications for logging the progress of tunnel construc-

tion sites (see Chapter 4). We showed that our framework holds up well with state of the art solutions. Especially compared to the former panorama stitching solution at JR, we accomplish a huge performance boost.

Some options which improve the visual perception of a panorama, but taint its integrity, were implemented and can be extended. This allows utilisation to more commercial applications, where traceability and documentation are not superficial.

5.2 Further Work

As mentioned several times in this thesis, the main objective was the overall design of the pipeline. Modules like Pointing Refinement or Radiometric Correction received only basic functionality and are considered subject to change in near future. Also in other modules there is room for improvement. This section will briefly point out already existing ideas and known limitations of the current state.

5.2.1 Pointing Refinement

According to literature [16], modelling the feasible space of the cameras during pointing refinement improves the results a lot. Instead of handling every camera independently, only the given angular settings vary slightly from image to image. The rover parameters such as height and turn radius of the stereo rig are fixed or at least equal for each image in a set.

Another improvement could be the simultaneous optimisation of the used surface models. Instead of predefining a fixed plane, its parameters could be adapted and refined to fit measures.

5.2.2 Radiometric Correction

The current implementation uses the image-global average brightness to determine one scaling factor for each image. This factor is chosen in a way that overlapping images have an as equal average brightness as possible.

Instead of optimising on the image-global average we could use the colour values of homologous points in overlapping regions. Homologous points are like tie points, showing the same point of the 3D scene in both images. As such, they should have the

same colour value. By varying a scale and offset parameter for each image, we have to minimise the overall colour difference. As in Pointing Refinement, proper outlier detection is necessary for this method to work.

One could also simultaneously adjust vignetting for all images of a camera. Implementing both methods should boost the quality of this module.

5.2.3 Cut Lines

Apart from completely other methods to find cut lines, there is some advancement possible in the current implementation. We sacrificed parallelisability and flexibility compared to the ideas of Summa et al. [54] for covering all possible overlap configurations. It could be topic of further research to combine the advantages of both approaches.

In addition we could define more cut criteria next to colour, edges and depth. For example we could base cut lines not only on the agreement of directly neighbouring pixels, but on a whole patch of pixels. This could yield cut lines which allow blending, as the whole neighbourhood agrees with a cut.

It is also possible to implement export and import of cut lines to visualise and edit them in external programs.

5.2.4 Merging

The current implementation of the Merge module does nothing more than copying the image tiles into the final panorama. While this completely covers the need for scientifically consistent panoramas, it does not provide enough functionality for other use cases.

Given the cut lines, it should be very easy to implement feathering, i.e. a linear blending region between the images. Adaptive feathering along the cut line based on the agreement of the images could produce even better panoramas with limited computational effort.

Bibliography

[1] J. A. Equation for a line through a plane in homogeneous coordinates. http://math.stackexchange.com/questions/400268/equation-for-a-line-through-a-plane-in-homogeneous-coordinates, 9 2014. Last seen: Tuesday 13th December, 2016.

[2] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[3] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org. Last seen: Tuesday 13th December, 2016.

[4] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23(3):294–302, Aug. 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015718. URL http://doi.acm.org/10.1145/1015706.1015718.

[5] AMASE. Arctic mars analog svalbard expedition. http://www.esa.int/Our_Activities/Preparing_for_the_Future/Space_for_Earth/Arctic/Mars_in_the_Arctic/, 2013. Last seen: Tuesday 13th December, 2016.

[6] A. Bauer and G. Paar. Stereo reconstruction from dense disparity maps using the locus method. In *Optical 3D Measurement Techniques II: Applications in Inspection, Quality Control, and Robotics*, pages 460–466. International Society for Optics and Photonics, 1994.

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

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23 (11):1222–1239, 2001.

[9] M. Brown and D. Lowe. Recognising panoramas. In *Proceedings of the 9th International Conference on Computer Vision*, volume 2, pages 1218–1225, Nice, October 2003.

[10] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[11] D. Capel. Image mosaicing. In *Image Mosaicing and Super-resolution*, pages 47–79. Springer, 2004.

[12] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

[13] R. T. Collins. A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 358–363. IEEE, 1996.

[14] P. d'Angelo et al. Hugin - Panorama photo stitcher. `http://hugin.sourceforge.net/`, 2016. Last seen: Tuesday 13th December, 2016.

[15] J. Davis. Mosaics of scenes with moving objects. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 354–360. IEEE, 1998.

[16] B. Deen. In-situ mosaic production at JPL/MIPL. Technical report, NASA's Jet Propulsion Laboratory, Pasadena, California, 2012.

[17] R. G. Deen, S. S. Algermissen, N. A. Ruoff, A. C. Chen, O. Pariser, K. S. Capraro, and H. E. Gengl. Pointing correction for mars surface mosaics. In *Second Planetary Data Workshop*, page Abstract #7055, Houston, 2015. Lunar and Planetary Institute. URL `http://www.lpi.usra.edu/meetings/planetdata2015/pdf/7055.pdf`.

[18] A. Delong and Y. Boykov. A scalable graph-cut algorithm for nd grids. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[19] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[20] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[21] ESA - European Space Agency. The exomars programme 2016-2020. `http://exploration.esa.int/mars/46048-programme-overview/`, 2016. Last seen: Tuesday 13th December, 2016.

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

[23] K. D. Gremban, C. E. Thorpe, and T. Kanade. Geometric camera calibration using systems of linear equations. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 562–567. IEEE, 1988.

[24] M. J. Hannah. Computer matching of areas in stereo images. Technical report, DTIC Document, 1974.

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

[26] W. V. D. Hodge, W. Hodge, and D. Pedoe. *Methods of algebraic geometry*, volume 2. Cambridge University Press, 1994.

[27] ISRO - Indian Space Research Organisation. Mars orbiter mission. `http://www.isro.gov.in/pslv-c25-mars-orbiter-mission`, 2016. Last seen: Tuesday 13th December, 2016.

[28] K. Itô and Nihon-Sūgakkai. *Encyclopedic Dictionary of Mathematics*. Bd. 1. MIT Press, 1993. ISBN 9780262590204. URL `https://books.google.at/books?id=azS2ktxrz3EC`.

[29] W. Jiang, M. Okutomi, and S. Sugimoto. Panoramic 3d reconstruction using rotational stereo camera with simple epipolar constraints. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 371–378. IEEE, 2006.

[30] S. B. Kang, R. Szeliski, and M. Uyttendaele. Seamless stitching using multi-perspective plane sweep. *Microsoft Research, Redmond, WA, Technical Report MSR-TR-2004-48*, 2004.

[31] R. Kingslake. *A History of the Photographic Lens*. Academic Press, 1989. ISBN 9780124086401. URL https://books.google.at/books?id=0JrJrEJ-r9QC.

[32] A. Levin, A. Zomet, S. Peleg, and Y. Weiss. Seamless image stitching in the gradient domain. In *European Conference on Computer Vision*, pages 377–389. Springer, 2004.

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

[34] S. Liu and G. Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *Int. J. Inf. Syst. Sci*, 4(1):160–177, 2008.

[35] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[36] Micosoft Research. Image Composition Editor. http://research.microsoft.com/en-us/um/redmond/projects/ice/, 2016. Last seen: Tuesday 13th December, 2016.

[37] D. L. Milgram. Computer methods for creating photomosaics. *IEEE Transactions on Computers*, 24(11):1113–1119, 1975.

[38] NASA. Interactive: Billion-pixel view of mars from curiosity rover. http://mars.nasa.gov/bp1/, 2013. Last seen: Tuesday 13th December, 2016.

[39] NASA. Mars science laboratory. http://mars.nasa.gov/msl/, 2016. Last seen: Tuesday 13th December, 2016.

[40] NASA. Msl science corner: Mast camera (mastcam). http://msl-scicorner.jpl.nasa.gov/Instruments/Mastcam/, 2016. Last seen: Tuesday 13th December, 2016.

[41] NASA. Mars exploration - missions. http://mars.nasa.gov/programmissions/missions/, 2016. Last seen: Tuesday 13th December, 2016.

[42] A. V. Oppenheim and R. W. Schafer. *Discrete-time signal processing*. Pearson Higher Education, 2010.

[43] G. Paar and W. Polzleitner. Robust disparity estimation in terrain modeling for spacecraft navigation. In *Pattern Recognition, 1992. Vol. I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, pages 738–741. IEEE, 1992.

[44] P. Peer and F. Solina. Panoramic depth imaging: Single standard camera approach. *International Journal of Computer Vision*, 47(1-3):149–160, 2002.

[45] S. Peleg. Elimination of seams from photomosaics. *Computer Graphics and Image Processing*, 16(1):90–94, 1981.

[46] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 338–343. IEEE, 1997.

[47] S. Peleg, M. Ben-Ezra, and Y. Pritch. Omnistereo: Panoramic stereo imaging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):279–290, 2001.

[48] S. Pugh, D. Barnes, L. Tyler, et al. Aupe–a pancam emulator for the exomars 2018 mission. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2012.

[49] L. H. Quam. Hierarchical warp stereo. *Readings in computer vision*, pages 80–86, 1984.

[50] H. Qureshi, M. Khan, R. Hafiz, Y. Cho, and J. Cha. Quantitative quality assessment of stitched panoramic images. *Image Processing, IET*, 6(9):1348–1358, 2012.

[51] C. Richardt, Y. Pritch, H. Zimmer, and A. Sorkine-Hornung. Megastereo: Constructing high-resolution stereo panoramas. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1256–1263, June 2013. doi: 10.1109/CVPR.2013.166.

[52] Roscosmos. Chronicle of soviet-russian space program. `http://en.federalspace.ru/174/`, 2016. Last seen: Tuesday 13th December, 2016.

[53] M. Russinovich. Process explorer. `https://technet.microsoft.com/en-us/sysinternals/processexplorer`, 2016. Last seen: Tuesday 13th December, 2016.

[54] B. Summa, J. Tierny, and V. Pascucci. Panorama weaving: fast and flexible seam processing. *ACM Transactions on Graphics (TOG)*, 31(4):83, 2012.

[55] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[56] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[57] P. Tourism. Exploring photo collections in 3d. *Microsoft Research, University of Washington Animation Research Labs, National Science Foundation*, 2006. URL `http://www.cs.cornell.edu/~snavely/bundler/`.

[58] M. Uyttendaele, A. Eden, and R. Skeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–509. IEEE, 2001.

[59] J. Yan-bin. Plücker coordinates for lines in the space. Technical report, Iowa State University, 2015.

[60] Z. Zhang and G. Xu. Epipolar geometry in stereo, motion and object recognition, 1996.

[61] Q. Zhi and J. Cooperstock. Toward dynamic image mosaic generation with robustness to parallax. *Image Processing, IEEE Transactions on*, 21(1):366–378, Jan 2012. ISSN 1057-7149. doi: 10.1109/TIP.2011.2162743.