3D Reconstruction of Small Solar System Bodies using Rendered and Compressed Images

Gabriel Jörg Schwarzkopf

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 24.2.2020

Supervisor

Prof. Jaan Praks

Advisor

Dr. Andris Slavinskis

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Gabriel Jörg Schwarzkopf

Title 3D Reconstruction of Small Solar System Bodies using Rendered and Compressed Images

Degree programme Space Science and Technology

Major Space Robotics and Automation **Code of major** ELEC3047

Supervisor Prof. Jaan Praks

Advisor Dr. Andris Slavinskis

Date 24.2.2020 **Number of pages** 82+3 **Language** English

Abstract

Synthetic image generation and reconstruction of Small Solar System Bodies and the influence of compression is becoming an important study topic because of the advent of small spacecraft in deep space missions. Most of these missions are fly-by scenarios, for example in the Comet Interceptor mission. Due to limited data budgets of small satellite missions, maximising scientific return requires investigating effects of lossy compression. A preliminary simulation pipeline had been developed that uses physics-based rendering in combination with procedural terrain generation to overcome limitations of currently used methods for image rendering like the Hapke model. The rendered Small Solar System Body images are combined with a star background and photometrically calibrated to represent realistic imagery. Subsequently, a Structure-from-Motion pipeline reconstructs three-dimensional models from the rendered images. In this work, the preliminary simulation pipeline was developed further into the Space Imaging Simulator for Proximity Operations software package and a compression package was added. The compression package was used to investigate effects of lossy compression on reconstructed models and the possible amount of data reduction of lossy compression to lossless compression. Several scenarios with varying fly-by distances ranging from 50 km to 400 km and body sizes of 1 km and 10 km were simulated and compressed with lossless and several quality levels of lossy compression using PNG and JPEG 2000 respectively. It was found that low compression ratios introduce artefacts resembling random noise while high compression ratios remove surface features. The random noise artefacts introduced by low compression ratios frequently increased the number of vertices and faces of the reconstructed three-dimensional model.

Keywords deep space exploration, simulation, Small Solar System Bodies, compression, image rendering, computer vision

Acknowledgements

I would first like to thank my thesis advisor Dr. Andris Slavinskis and my supervisor Prof. Jaan Praks of the School of Electrical Engineering at Aalto University. Their office door was always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank the person who started this research project: Dr. Mihkel Pajusalu. Without laying the foundation of the project and his continuous input, the project would not have been successful.

I would also like to acknowledge Prof. Mikael Granvik of the Department of Computer Science, Electrical and Space Engineering at the Luleå University of Technology as the second reader of this thesis, and I am gratefully indebted for his valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents and to my girlfriend for providing me with unfailing support and continuous encouragement throughout my studies and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Otaniemi, 24.2.2020

Gabriel Jörg Schwarzkopf

Contents

Symbols and Abbreviations

Symbols

Astronomical Symbols

Υ	Aries
$\odot$	Sun
♀	Venus

Greek Symbols

α	Albedo
δ	Declination
λ	Wavelength
$\Delta\lambda$	Wavelength bandwidth
ν	True anomaly
ω	Argument of periapsis
Ω	Right ascension of ascending node
α_r	Right ascension
σ	Standard deviation of a Gaussian distribution

Latin Symbols

a	Orbit semi-major axis
A	Aperture Area
A_{pixel}	Area of a pixel
f_c	Calibration factor for photometric calibration
v_d	Vector pointing along the optical axis of a camera
$\hat{v}_d$	Normalised vector pointing along the optical axis of a camera
c_u	Coordinate of image principle point along image u-axis
k_u	Scaling factor along image u-axis
c_v	Coordinate of image principle point along image v-axis
k_v	Scaling factor along image v-axis
d_s	Data size
D	Aperture diameter
d	Distance of the spacecraft from the Sun
e	Orbit eccentricity
e_i	i^{th} edge vector of the Field of View
e_{left}	Vector pointing to the left edge of the Field of View
e_{lower}	Vector pointing to the lower edge of the Field of View
e_{right}	Vector pointing to the right edge of the Field of View
e_{upper}	Vector pointing to the upper edge of the Field of View
$\mathbf{E}$	Essential matrix
F	Photon flux density
F_d	Scaled photon flux density
F_0	Photon flux density at magnitude $m = 0$
F_{ref}	Reference photon flux
F_{stars}	Total photon flux of stars in the Field of View

f	Focal length
$\mathbf{F}$	Fundamental matrix
G	Value of a Gaussian distribution
$\mathbf{H}$	Homography matrix
i	Orbit inclination
I_{ref}	Reference intensity of an image
$\mathbf{K}$	Camera matrix
l_{pixel}	Length of a pixel
l'_i	Epipolar line of point i in an image
m	Apparent magnitude
m_d	Multiplier for systems that are not diffraction limited
M	Mean anomaly
p	Cartesian coordinate vector
v	Value of a pixel
v_0	Original value of a pixel
P	Set of points
$\mathbf{R}$	Rotation matrix
T	Runtime of a program
r_u	Number of pixels of an image sensor along the u-axis
r_v	Number of pixels of an image sensor along the v-axis
s_h	Height of a camera sensor
s_w	Width of a camera sensor
s_k	Width or height of a camera sensor
t	Translation vector
u_{pix}	Pixel u-coordinate
S_{stars}	Sum of pixel values of a single channel of a star map
v_{pix}	Pixel v-coordinate
$\hat{v}_r$	Normalised vector pointing right in the image plane
$\hat{v}_u$	Normalised vector pointing up in the image plane
x_i	Cartesian coordinates of point i in an image

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
67P	67P/Churyumov–Gerasimenko
81P	81P/Wild
AC-RANSAC	A Contrario RANSAC
AKAZE	Accelerated-KAZE
AOCS	Attitude and Orbit Control System
ASP	NASA Ames Stereo Pipeline
BRIEF	Binary Robust Independent Elementary Features
BSDF	Bidirectional Scattering Distribution Function
CCD	Charge-Coupled Device

CI	Comet Interceptor
CPU	Central Processing Unit
CV	Computer Vision
DART	Double Asteroid Redirection Test
DCT	Discrete Cosine Transform
DSAN	Deep Space Autonomous Navigation
DWT	Discrete Wavelet Transform
ESA	European Space Agency
FAST	Features from Accelerated Segment Test
FoV	Field of View
GPU	Graphics Processing Unit
HDR	High Dynamic-Range
IAU	International Astronomical Union
JP2	JPEG 2000
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LZMA	Lempel–Ziv–Markov Algorithm
MANTIS	Main-belt Asteroid and NEO Tour with Imaging and Spectroscopy
MAT	Multi-Asteroid Touring
MP3	MPEG-2 Audio Layer III
MSAC	M-estimator Sample Consensus
MVS	Multi-View Stereo
NASA	National Aeronautics and Space Administration
OpenMVG	Open Multiple View Geometry
OpenMVS	Open Multi-View Stereo reconstruction
ORB	Oriented FAST and rotated BRIEF
PANGU	Planet and Asteroid Natural Scene Generation Utility
PNG	Portable Network Graphics
PROSAC	Progressive Sample Consensus
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RGB	Red, Green and Blue
RGBA	Red, Green, Blue and Alpha
SfM	Structure-from-Motion
SIFT	Scale-Invariant Feature Transform
SISPO	Space Imaging Simulator for Proximity Operations
skimage	Scikit-Image
SSSB	Small Solar System Body
SURF	Speeded-Up Robust Features
TIFF	Tagged Image File Format
UBVRI system	Ultraviolet, Blue, Visual, Red, Infrared system
UBV system	Ultraviolet, Blue, Visual system
UCAC4	USNO CCD Astrograph Catalog 4
USNO	United States Naval Observatory
VM	Virtual Machine

1 Introduction

Studying Small Solar System Bodies (SSSBs) provides a unique opportunity to understand the evolution of the Solar System, since asteroids, comets and other SSSBs are remnants of the early formation phase of the Solar System [1, 97]. SSSBs are studied to understand the origin of water and carbon-based molecules on Earth and other planets. While asteroids can be studied to understand the chemical composition of the inner Solar System, long- and short-period comets originate from the Kuiper belt and Öpik-Oort cloud and can help to understand the chemical composition of the outer and early Solar System.

Currently, most information about SSSBs is obtained from remote observation with ground based telescopes or space telescopes [11]. Remote observations provide statistics of thousands of objects using the same instrument. However, mostly bluk information about orbits, albedos, colours, spectra and sizes of SSSBs can be obtained due to their small size. In contrast, space missions provide detailed maps of albedos, colours, spectra and surface features, Three-Dimensional (3D) models, precise mass and density estimates, physical properties and chemical composition, both surface and sub-surface composition. Several missions to SSSBs are confirmed. The *Comet Interceptor (CI)* mission by European Space Agency (ESA) will target an unknown dynamically-new comet [81]. The *HERA* mission by ESA and the *Double Asteroid Redirection Test (DART)* mission by National Aeronautics and Space Administration (NASA) will jointly target the double asteroid (65803) Didymos (1996 GT) [22, 89].

More than 3600 comets and 930 000 asteroids are known [63, 86]. The largest group of SSSBs, namely asteroids, are categorised into 24 classes based on their variety in size, spectral type, surface activity and other characteristics [21]. Approximately 20 SSSBs have been visited by spacecraft [52]. These missions were monolithic, single spacecraft missions which targeted one or a few SSSBs each. Often these missions were designed as orbiters around one or two SSSBs. For example the *Hayabusa 2* mission targeted asteroid (162173) Ryugu (1999 JU3) or the *OSIRIS-REx* mission which orbited (101955) Bennu [54, 99]. With costs around $1 billion for each mission, it seems infeasible to visit many targets and study the large existing variety [2, 23].

Performing fly-bys at several objects increases the number of targets at the cost of limited observation time. For example, the *Lucy* mission will perform fly-bys at several of Jupiter's Trojans [84]. The *CASTAway* or *Main-belt Asteroid and NEO Tour with Imaging and Spectroscopy (MANTIS)* mission concepts aim at touring main belt asteroids [11, 75]. Rapidly developing technology provides more opportunities to study SSSBs since interplanetary missions become feasible at reasonable costs using small spacecraft [50, 71, 81]. For example, the *Multi-Asteroid Touring (MAT)* mission proposes to perform fly-bys at hundreds of main belt asteroids using a fleet of small spacecraft [80]. The *HERA* mission will deploy its daughter-spacecraft *APEX* which will autonomously navigate at the Didymos system [51]. In the *CI* mission, two daughter-spacecraft will independently fly-by a target and perform measurements [81].

Small and low-cost spacecraft can not only increase the number of spacecraft that study SSSBs but also allow to fly closer to their nuclei, which is scientifically

more interesting but associated with a higher risk of losing a spacecraft. Flying closer to an SSSB and using smaller spacecraft inflicts additional constraints onto the data budget of such spacecraft. Miniature radios provide lower data rates due to size and power limitations. A higher probability of losing the spacecraft requires transmitting the most important data first and complementing it later if possible. Compression is used to reduce the amount of data that needs to be transmitted. The two main categories of compression are lossless and lossy compression. Lossless compression does not lose information during the compression process. In contrast, lossy compression accepts a certain amount of irreversible loss of information to achieve higher compression ratios. Lossy image compression results in artefacts. However, artefacts can be made nearly imperceptible depending on the compression method. Therefore, it is necessary to investigate the effect of compression on image data and science return, especially in lossy compression. In addition, an algorithm is required that prioritises which data is transmitted first and which data transmission can be postponed to maximise scientific return.

Designing such deep space missions, maximising their scientific return and targeting unknown objects require a versatile simulation environment to develop, test and validate systems and algorithms. The SurRender software offers such capabilities. However, it is proprietary software developed by Airbus which cannot be accessed by external parties [12]. Another software suite that is available for ESA projects or commercially is Planet and Asteroid Natural Scene Generation Utility (PANGU). However, PANGU uses OpenGL 3.0 which is limited in its accuracy [57]. Other software suites such as the NASA Ames Stereo Pipeline (ASP) only provide stereogrammetry capabilities for geodesy applications and thus can only be used for missions to bodies with existing image archives [8].

Although data sets from the *OSIRIS* instrument aboard *Rosetta* [66] or from the *OSIRIS-REx*, *Dawn* and *NEAR* missions [69] are publicly available, the total number of images and their variation remains small. Since there is only a limited number of real images it is necessary to synthetically create images of SSSBs for developing systems and algorithms. Synthetic image creation of SSSBs is currently based on the parametric empirical models for directional reflectance properties of airless regolith surfaces developed in a paper series by Hapke [34–40]. However, the Hapke model is challenged because it is an empirical model with shortcomings, for example with shadows especially at slopes [79]. Shadows are crucial to realistically represent SSSB surface features. The increase in computing power in recent years can be leveraged to employ ray-tracing techniques to overcome such problems [53, 79].

The Space Imaging Simulator for Proximity Operations (SISPO) is being developed to cover the three topics mentioned above. First, physics-based image rendering using ray-tracing is implemented. Second, compression and decompression with different algorithms is performed to investigate the quality loss on the rendered images. Third, a Structure-from-Motion (SfM) processing pipeline is established to reconstruct a 3D model of the SSSB. A preliminary code base was developed for a proof of concept for rendering, reconstruction and studying the possibility of transmitting reconstructed 3D models instead of raw images [68].

The aim of this project is to develop a first version of SISPO that covers a basic

functionality for rendering, compression and reconstruction by adding a compression package and developing the original code base further into a consistent software package. Based on an initial 3D model, SISPO shall be able to render images of realistic fly-by scenarios, compress and decompress these images to introduce compression artefacts and reconstruct a 3D model. This pipeline shall be used to investigate the effects of compression and the possible amount of data reduction. To see the effects of compression, a lossy compression technique with different compression ratios shall be compared to lossless compression. This will provide a first insight into artefacts created by lossy compression and by how much data can be reduced.

This thesis is structured into three main sections and a conclusion. **Section 2 Scientific Background** provides background information about SSSBs. Furthermore, important concepts of image rendering and processing are explained. Moreover, an introduction to a the SfM technique is provided. The section concludes with background information about relevant image compression and processing techniques. **Section 3 Space Imaging Simulator for Proximity Operations** explains the implementation, design choices and input parameters of the SISPO software package. **Section 4 Results** presents rendered images, reconstructed 3D models and effects of compression on them. Moreover, an analysis of the results is presented along with a discussion. **Section 5 Conclusion** summarises the most important findings of the thesis and provides an outlook on possible future developments of SISPO.

2 Scientific Background

2.1 Small Solar System Bodies

The International Astronomical Union (IAU) defines an SSSB as any object in the Solar System, that is not a planet, dwarf planet or satellite [43]. Within this work, the term SSSB mostly refers to asteroids and comets.

In a classical definition, asteroids and comets were two distinct SSSB classes based on observational, physical and dynamical properties. However, the discovery of active asteroids and dormant comets began to blur these definitions. Many observed objects cannot be strictly classified into the classical categories anymore. Therefore these objects are more frequently described as part of an asteroid-comet continuum with asteroids and comets at the ends of the continuum [42].

Despite the shift towards a continuum definition, the classical definition is used in this work. Non-active objects are referred to as asteroids and active objects are referred to as comets where it is necessary to distinguish these types of objects. Consequently, a description following the classical definition of asteroids and comets is provided.

2.1.1 Asteroids

Asteroids are rocky bodies that mostly reside in an orbit between Mars and Jupiter, i.e. the asteroid main belt. More than 930 000 asteroids are known of which more than 540 000 are numbered. The ephemeris of more than 930 000 and absolute magnitude of more than 925 000 is known. However, physical parameters like diameter ($> 136\,000$), albedo ($> 135\,000$), rotation period ($> 19\,000$), spectral type (> 2000) and the standard gravitational parameter (> 10) are known for only a small fraction of asteroids [47].

Asteroids are often referred to as rubble piles, i.e. objects which gravitationally aggregated boulders resulting in a low bulk density because of many void spaces in their internal structure [74]. An example of rubble pile asteroids are Bennu and Ryugu due to their low density [15, 100].

2.1.2 Comets

Comets are small icy bodies that mostly reside farther away from the Sun than asteroids. Inside the heliosphere, the source of comets is the Kuiper belt while comets from outside the heliosphere are thought to originate in the Öpik-Oort cloud. Comets are occasionally perturbed by the gravity of other objects which moves their orbits closer to the Sun. While being close to the Sun, volatiles begin to evaporate which creates the well-known coma around the nucleus. In most cases the coma is several orders of magnitudes larger than the nucleus. The size of nuclei is on the order of a few kilometres while comas can extend hundreds of thousands of kilometres. Additionally, a gas and a dust tail are formed. The gas tail originates from coma charged particles that are carried away from the nucleus by the magnetic field carried by the solar wind. The dust tail is formed by small dust particles in the coma

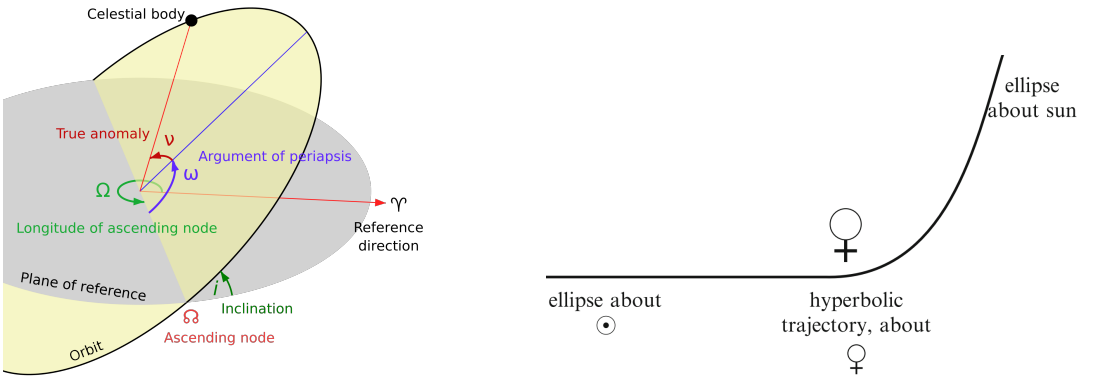
which are carried away from the nucleus by the solar radiation pressure. Larger dust particles remain along a comets trajectory as a trail forming part of the meteoroid environment. If a comet is heated unevenly, trapped gas escapes at weak spots in the surface forming jets of gas and dust [1, 17, 82].

Physical parameters of comets are not well-known, since comet nuclei are small. It is difficult to image a comet nucleus when it moves closer to the Sun since the nucleus is surrounded by the coma. Approximately 3500 comets are known of which more than 400 are numbered [47]. The ephemeris (> 1700) and magnitude (> 1300) are known for many comets. Only a few comets have known physical properties like the diameter (> 100) or the rotation period (> 20) [47].

It is not clear whether comets are generally similar to rubble piles or rather consolidated, monolithic bodies. It is assumed that most bodies in the range of 200 m to 10 km are rubble piles [98]. However, density measurements and surface features of 67P/Churyumov–Gerasimenko (67P) are not conclusive so far [101].

2.1.3 Orbital Mechanics

All objects in space move on trajectories, following the Kepler equations. Trajectories are commonly defined using six orbital elements. Semi-major axis a, eccentricity e, inclination i, right ascension of ascending node Ω, argument of periapsis ω and the true anomaly ν or mean anomaly M at a given epoch. This parameterisation is also called modified Keplerian elements [41]. Figure 1a shows the geometric relations of the angular elements.



(a) (b)

Figure 1: Important astrodynamic definitions and relations. (a) Modified Keplerian elements [19]. (b) A closed orbit around the Sun (☉) becomes a hyperbolic trajectory around another body, in this example Venus (♀) [41].

Fly-by scenarios in the Solar System are often closed orbits around the Sun, i.e. $e_{\odot} < 1$ while they are hyperbolic trajectories in the reference frame of the target, i.e. $e_{♀} > 1$. An example is given in Figure 1b. A high relative velocity and the small gravitational force exerted by a SSSB result in a trajectory closer to a straight line than the bend trajectory presented in Figure 1b.

Most asteroids move on orbits with low eccentricity while most comet orbits have high eccentricities [14]. The eccentricity distribution is caused by the origin of asteroids and comets. While asteroids mostly originate from the main belt which is a nearly circular region, comets enter the inner Solar System from far out, i.e. comets' aphelion is far from the Sun resulting in high eccentricities.

2.2 Image Rendering

Rendering is the process of creating Two-Dimensional (2D) images from 3D objects. A virtual 3D world with objects is created which is used to calculate 2D images. Light sources and cameras produce artificial illumination and define which part of the world is being captured. A rendering engine calculates the pixel values to generate the final image based on energy conservation by using the rendering equation [48].

2.2.1 Path Tracing

Path tracing is a special form of ray tracing. Ray tracing is a rendering technique where the path of light rays is traced to generate pixels while simulating effects of encounters with objects. Path tracing does not branch into an exponentially growing number of rays when being reflected or refracted but only a single path is followed. Path tracing cuts computation time dramatically compared to ray tracing in scenes with a lot of reflection, refraction and shadow rays per pixel [48]. Path tracing can simulate different effects, such as reflection, refraction, scattering and dispersion. There are four types of rays: camera rays, reflection rays, transmission rays and shadow rays. Reflection and transmission can be further categorised as either diffuse, glossy or singular. Path tracing is a popular rendering technique where a high degree of realism is necessary since it is a realistic simulation of light transport. However, the high degree of realism created by path tracing requires substantial computational power [93].

2.2.2 3D Models and Shaders

A 3D model is a set of vertices in 3D space linked by surfaces. The most common type of 3D models are polygonal meshes. Polygonal meshes are shell models that consist of polygons. A vertex is a corner of a polygon, i.e. every triangle has three vertices and every tetrahedron has four vertices. A face refers to the surfaces that make up a 3D model. Depending on the rendering environment, 3D models can have varying levels of detail and their surface can have different properties such as reflection, refraction and transmission [29].

A shader is used to calculate effects of interactions of rays and objects during the rendering process. Shaders provide a flexible method to influence the rendering outcome during the rendering process. Shaders can be used to change the positions of vertices, colours, lighting and surface properties by using equations. Complex surface structures and texture can be generated procedurally with shaders. Shaders are commonly executed on Graphics Processing Units (GPUs) since GPU hardware is well suited for this task [70, 76, 83].

Shaders that influence the surface of an object use a Bidirectional Scattering Distribution Function (BSDF). BSDFs are a mathematical functions that describe light scattering behaviour of surfaces of 3D models. Several shader classes such as diffuse, glossy, refraction and transparent shaders create the respective effect based on BSDFs [29, 70].

2.2.3 Field of View

The Field of View (FoV) defines the visible extent of a camera view in 3D scene. Vectors that point to the edges of the FoV can be constructed from geometric considerations resulting in

$$e_i = v_d \pm \frac{v_j \times s_k}{2 \times f}, \tag{1}$$

where e_i is a vector for the i^{th} edge of the FoV, v_d is the direction vector of the camera, v_j is the vector pointing right or up in the FoV plane, s_k is the camera sensor width or height and f is the focal length. The left and right edge vectors, e_{left} and e_{right}, are calculated with the vector v_r pointing right in the FoV plane and the sensor width s_w. The upper and lower edge vectors, e_{upper} and e_{lower} are calculated with the vector v_r pointing up in the FoV plane and the sensor height s_h.

In addition to the extent of the FoV within the image plane, rendering requires the FoV to be clipped to a minimum and maximum distance. Clipping defines which objects appear in a rendered scene. Clipping is necessary to limit the required computational power for rendering.

2.2.4 Photometric calibration

Photometric calibration is the process of correcting raw images from a sensor to a common level of brightness. Differences can originate from varying exposure times, gains or different camera sensors [5]. Photometric calibration uses a photometric system, such as the Ultraviolet, Blue, Visual system (UBV system), which are reference systems with which star magnitude measurements can be compared for a given band of the system [6].

The apparent magnitude of a star is converted into photon flux density F relative to magnitude 0 by using,

$$F = 10^{-0.4 \times m}, \tag{2}$$

where m is the apparent star magnitude. Given the apparent magnitude of a specific band, the photon flux density F of a reference object can be calculated using

$$F = F_0 \times \frac{\Delta\lambda}{\lambda} \times 10^{-0.4 \times m}, \tag{3}$$

where F_0 is the photon flux density at magnitude 0, $\Delta\lambda$ is the wavelength bandwidth, λ is the centre wavelength and m is the object's magnitude. The required constants

Table 1: Constants for calculating photon fluxes using the UBV system [6].

Band	**Centre wavelength** λ [nm]	F_0 $[1 \times 10^{-26}\,\mathrm{W\,m^{-2}\,Hz^{-1}}]$	$\frac{\Delta\lambda}{\lambda}[\ ]$
U	0.36	1810	0.15
B	0.44	4260	0.22
V	0.55	3640	0.16

to calculate the photon flux of an object from its magnitude in a given band are presented in Table 1.

Consequently, the reference photon flux for one pixel of the Charge-Coupled Device (CCD) F_{ref} is calculated using

$$F_{ref} = F \times \frac{A \times A_{pixel}}{f^2 \times \pi}, \tag{4}$$

where F is the photon flux density, a is the aperture area, A_{pixel} is the area of a pixel and f is the focal length of the instrument.

In the last step, the calibration factor f_c is calculated. It is defined as

$$f_c = \frac{F_{ref} \times \alpha}{I_{ref}}, \tag{5}$$

where F_{ref} is the reference flux, α is the albedo and I_{ref} is the reference intensity. Multiplying the image with f_c produces the calibrated image.

The effect of photometric calibration is delineated in Figure 2. Photometric calibration corrects the brightness difference of a set of images towards a common brightness level [5].

Figure 2: Image series before (top) and after (bottom) photometric calibration which eliminated the brightness differences [5].

2.3 Computer Vision

Computer Vision (CV) is the science of extracting information from digital photos and videos by mimicking the human vision system using a computer. CV encompasses a wide field of activities, from image formation, processing, detecting and

matching features, image segmentation and 3D reconstruction [88]. CV is computer based photogrammetry which aims to obtain information about physical objects and the environment from photographic images [49]. A common approach for 3D reconstruction is stereo-photogrammetry, also referred to as computer stereo vision. Stereo-photogrammetry applies the binocular vision principle of the human vision system to obtain structural information from images [9]. Since most, if not all, deep space missions have a visual imager instrument on-board, CV provides a useful framework to obtain the 3D structure of an observation target. A similar approach is SfM where the camera motion creates the stereo perspective.

2.3.1 Pinhole Camera Model

All CV problems require a camera model. The most commonly used model in CV is the pinhole camera. Figure 3 presents an overview of the important elements of the pinhole model [87].

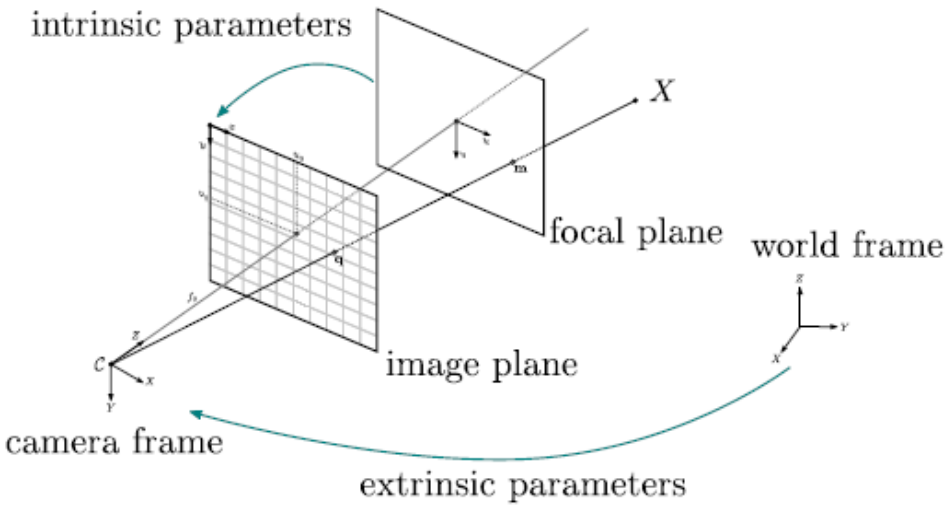


Figure 3: Overview of the components of the pinhole camera model. Intrinsic camera parameters model optical distortions, extrinsic parameters model camera position and orientation. The image plane is where the light sensor is. Locations in the image plane are described with u,v-coordinates. The focal plane is where the focus of the optical system is. The thin line from the origin of the camera frame through the centres of the image and focal plane is the optical axis [3].

The pinhole camera model can be described using the 3×4 camera matrix $\mathbf{K}$ defined as

$$\mathbf{K} = \begin{bmatrix} f \times k_u & 0 & c_u \\ 0 & f \times k_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & t \end{bmatrix}, \tag{6}$$

where f is the distance between the focal plane and the image plane, k_u and k_v are scaling factors, c_u and c_v are the coordinates of the principle point on the image plane,

$\mathbf{R}$ is a 3×3 rotation matrix and t being a 3×1 translation vector. The first matrix of $\mathbf{K}$ reflects the camera intrinsic parameters while the second matrix describes the extrinsic parameters. More sophisticated pinhole camera models include distortions. Distortions can include one or more factors for radial and tangential distortions. A special case with three radial and two tangential distortion factors is called Brown T2 model [3, 87].

The conversion between object coordinates and image coordinates is obtained from geometric considerations of the pinhole camera model shown in Figure 3 and is defined as

$$u_{pix} = \frac{f}{s_w} \times \frac{\hat{v}_r \cdot p}{\hat{v}_d \cdot p + 1} \times (r_u - 1), \tag{7}$$

where u_{pix} is the u-coordinate of a pixel in the image frame, f is the focal length, s_w is the sensor width, $\hat{v}_r$ is the unit vector pointing right in the image plane, p is the Cartesian coordinate vector of a star, v_d is the vector of the optical axis and r_u is the number of pixels in u-direction. Similarly, the conversion for v-coordinate of a pixel v_{pix} is obtained by replacing the sensor width s_w with the sensor height s_h, $\hat{v}_r$ by the unit vector $\hat{v}_u$ pointing up in the image plane and the number of pixels in u-direction r_u by the number of pixels in the v-direction r_v. The vectors $\hat{v}_r$ and $\hat{v}_u$ are calculated by subtracting v_d from the field of view edge vectors e_{right} and e_{upper} as defined in Equation 1.

2.3.2 Structure-from-Motion

SfM can be considered the inverse process to rendering, i.e. creating 3D models from 2D images. SfM uses multiple views of the same object from different camera positions to reconstruct an object's geometry. SfM recovers the 3D structure of an object and the camera poses of each image [88]. Depth information is obtained through the motion parallax created by the moving camera. Generic steps of an SfM processing pipeline are presented in Figure 4.

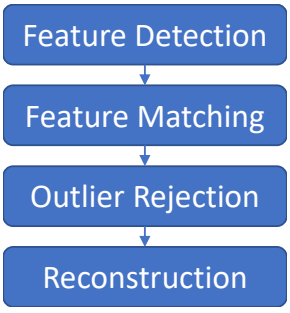


Figure 4: Generic steps of a SfM processing pipeline.

Figure 5 shows a generic observation geometry in an SfM problem. Various images containing feature points of the same object point are taken from varying positions. Lines connecting the features points and object point depict the relation between 2D feature points and a 3D object point.

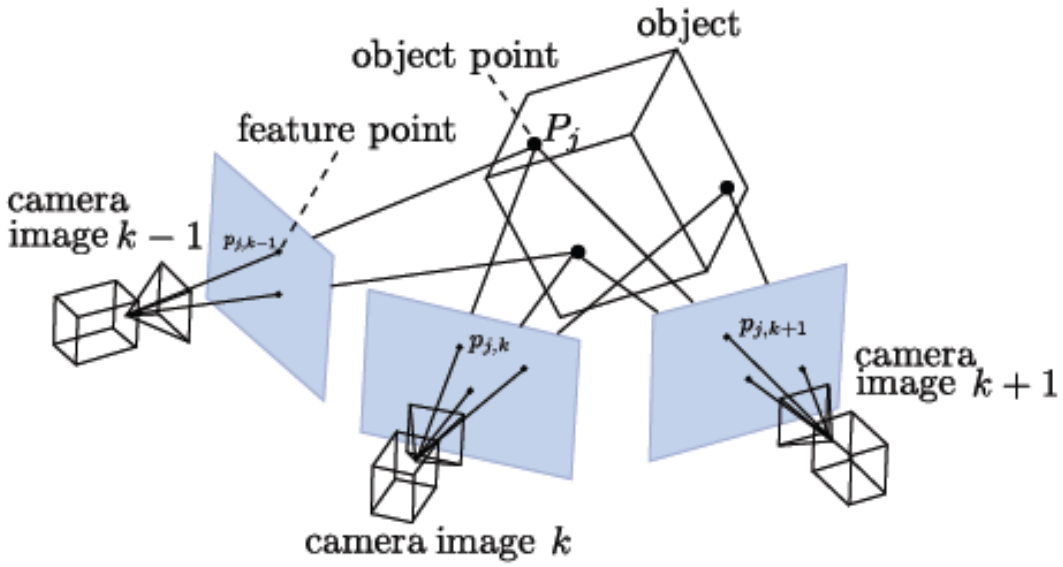


Figure 5: Generic observation geometry in a SfM problem. The 3D structure can be reconstructed from several point observations and intrinsic camera parameters [4].

Reconstructing 3D points from an image series requires finding correspondence between images. Correspondence is found by detecting and matching features in multiple images. A feature point is a local, meaningful and detectable part of an image. Feature points are also referred to as key-points or interest-points. Features can be image regions of sudden change, shape features or texture contours. Commonly detected features are corners, edges, junctions, blobs and lines [90]. Feature descriptors assign a distinct identity features after feature description for later matching. A range of feature description algorithms exist. Most feature descriptors are combined with a distinct feature detector. However, detectors and descriptors can be interchanged. Feature detectors are tasked with detecting feature-points in an image. Feature descriptors and detectors are frequently required to be scaling, rotation and affine invariant. The most prevalent feature detector-descriptor pairs are Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Oriented FAST and rotated BRIEF (ORB), KAZE and Accelerated-KAZE (AKAZE) [90]. After detection and description of all images, features are matched, i.e. an algorithm identifies the same feature in multiple images. Either an L1 or L2-norm for scalar descriptors or the hamming distance for binary descriptors are used. Feature matching can use image pairs or a series of images. Various geometries can be described using different mappings. The homography matrix $\mathbf{H}$ describes a coordinate mapping of two different views and is defined as

$$x_i' = \mathbf{H} x_i, \tag{8}$$

where x_i' are the coordinates of point i in the second image, x_i are the coordinates of point i in the first image and $\mathbf{H}$ is the homography matrix. However, $\mathbf{H}$ describes only

a purely rotating or moving camera capturing a planar scene [77]. The fundamental matrix **F** describes the relation between two images of the same scene for a moving camera. It relates points of uncalibrated images and is defined as

$$x_i'^T \mathbf{F} x_i = 0, \tag{9}$$

where x_i' are the coordinates of point i in the second image, x_i are the coordinates of point i in the first image and **F** is the fundamental matrix. The epipolar line l_i' constitutes all possible locations of point x_i'. The epipolar line is defined as

$$l_i' = \mathbf{F} x_i, \tag{10}$$

where **F** is the fundamental matrix and x_i are the coordinates of a point in the first image.

Taking into account the camera intrinsic parameters, the fundamental matrix **F** becomes the essential matrix **E** which is defined as

$$\mathbf{E} = (\mathbf{K}')^T \mathbf{F} \mathbf{K}, \tag{11}$$

with $\mathbf{K}'$ being the camera matrix of the second view, **F** being the fundamental matrix as defined in Equation 9 and **K** being the camera matrix of the first view. The essential matrix is intended to be used in conjunction with calibrated images where the camera intrinsic parameters are available.

If sufficient points are correctly mapped using one of the transformations from Equations 8, 9 and 11, the points are geometrically verified. However, geometric verification does not necessarily remove all outliers. Therefore, outlier rejection is performed to remove incorrect matches. Several algorithms such as Random Sample Consensus (RANSAC) [25], A Contrario RANSAC (AC-RANSAC) [58], M-estimator Sample Consensus (MSAC) [33] and Progressive Sample Consensus (PROSAC) [16] are used for outlier removal. These outlier rejection algorithms attempt to robustly estimate the correct model and remove outliers. The result of outlier rejection is the view graph which relates different views to each other with images as nodes and pairs as edges [77].

The reconstruction process produces a 3D point cloud based on the view graph. A point cloud is a set of independent points with coordinates in 3D space. Two principle methods for SfM-based reconstruction exist, incremental and global. For unorderd image sets the incremental approach is used more frequently [77].

Incremental reconstruction starts with an initial view pair. Selecting the initial pair is a critical step because the reconstruction algorithm might not converge after using a bad initial pair. Typically, starting with a scene with many overlapping camera views constitutes a robust initialisation and results in higher accuracy because of redundant information from many images [77]. Subsequently, additional images are registered to the model based on corresponding features which can be used to triangulate additional points in already registered images. The triangulation step can include estimating camera poses and camera intrinsic parameters for uncalibrated images. Triangulation is an essential step of making SfM robust by adding redundant

information about existing points to a model and adding additional points which increases the coverage of a model [77].

Image registration and triangulation are separate processes although their results are linked. Errors of one process can increase the error of the other process. If pose estimation during image registration is erroneous, the error propagates to the location estimate of a triangulated point. Bundle adjustment is often employed to increase robustness of this step. Bundle adjustment is the combined refinement of camera pose and point position estimate. A frequently used error minimisation algorithm is the Levenberg-Marquardt algorithm [59, 77].

Global SfM pipelines try to eliminate the problem of accumulating errors of incremental SfM by using a common reference frame for all views. While incremental reconstruction accumulates errors with every added view, global reconstruction attempts to distribute these residuals equally for all reconstructed points [60]. Global SfM pipelines calculate the essential matrix for all image pairs in the initial step. Rotation estimation is often separated from translation and structure estimation. First a consistent set of rotations is computed in a global frame based on relative rotations of input pairs. Second, camera translations and the object structure are estimated.

Stability of either SfM approach can be improved by providing priors for the camera intrinsic and extrinsic parameter estimation which are then used as initial guess for the optimisation process [45].

Incremental and global SfM methods create a sparse point cloud and estimate the intrinsic and extrinsic camera parameters. The sparse point cloud can be densified using Multi-View Stereo (MVS) techniques [67]. Image pairs with established camera positions are used to calculate depth maps. The depth maps are fused and filtered to increase the number of points in the point cloud. Point cloud densification mimics human depth perception and leverages the information to improve the accuracy of point clouds.

Subsequently, a point cloud is converted into a 3D model by estimating a mesh surface from the point cloud. Outliers are detected and removed from the scene during mesh creation. Different approaches exist which yield different results. Some techniques only use strongly supported faces, i.e. faces that are supported by many points of a point cloud. Other methods include weakly supported surfaces due to e.g. obstructions [46]. The output of mesh creation is a rough surface model of a 3D object.

A frequently used mesh creation algorithm is Delaunay tetrahedralisation. Delaunay tetrahedralisation is a triangulation method in which a set of points P is triangulated to not contain any point of P within the circumscribed sphere of any tetrahedron (3-simplex). Delaunay tetrahedralisation is a frequently used algorithm because it provides a unique solution to the triangulation problem [95].

The accuracy of a rough surface model can be increased with mesh refinement. In cases where a sparse point cloud was used to create the rough mesh, refinement can improve the model accuracy substantially. One such algorithm for mesh refinement is variational multiview stereovision. Initial mesh creation captures the main features of a structure hence variational multiview stereovision can employ local optimisation

algorithms since local optimisation is unlikely to be trapped in local minima [24, 95].

The final step in 3D reconstruction is texturing the surface model. Ideally, camera poses and surface models are exact. However, frequently texturing needs to cope with inaccurate camera poses and surface models. Likewise, effects such as differing illumination and exposure, unreconstructed occluding objects or varying image scales need to be addressed. Texturing is often separated in image selection and optimising the resulting image set for consistency [96]. Image selection algorithms are categorised into two types. One type blends multiple images per face to achieve consistent texture across patches. The other type uses a single image per face [96]. Subsequently, colours differences originating from exposure and illumination variations need to be adjusted. Seams between patches are corrected by photometric adjustments. Colour adjustment can be split into local or global adjustments. Local adjustment smooths the transition between texture patches by introducing gradients in texture luminances, e.g. example with heat diffusion equations [94]. Global adjustment methods calculate the luminance correction terms for a global optimum [55].

2.4 Image Compression and Processing

Digital image processing uses computers to process digital images through algorithms. Processing techniques relevant to this thesis are image compression (cf. Section 2.4.1), Gaussian filtering (cf. Section 2.4.2) and image down-sampling (cf. Section 2.4.3).

2.4.1 Image Compression

Data compression is tasked with encoding information using less bits than the original representation. Other terms for data compression are source coding or bit-rate reduction [56]. Two principle methods of compression exist, lossless and lossy compression. While it is possible to recover all information after lossless compression, lossy compression accepts a certain irreversible loss of information to achieve higher compression ratios.

Lossless compression leverages statistical redundancy in data to decrease the number of bits necessary to encode the same information in a reversible process. Many lossless compression algorithms exist such as Huffman coding, arithmetic coding or Lempel-Ziv algorithms [10].

Lossy compression is an irreversible process meaning some of the information is lost and cannot be recovered. The idea of lossy compression is that individual image properties are perceived with varying degrees of precision and are therefore not equally important. Lossy compression is a trade-off between compression ratio and data distortion. Audio data can be encoded with a lossy scheme by decreasing the accuracy of acoustic components that are beyond the capabilities of most humans. Another example is the human eye, which is more sensitive to changes in brightness (luminance) than to changes in colour (chrominance). Therefore, compression can lose colour information without having a strong influence on the perceived image quality. Consequently, image compression algorithms tend to use luminance-chrominance representations. The components in such a representation are almost uncorrelated

and it is easier to reduce colour information without changing luminance information. Frequently used formats for lossy compression are MPEG-2 Audio Layer III (MP3) or Joint Photographic Experts Group (JPEG). Most lossy compression algorithms are based on transform coding, especially Discrete Cosine Transform (DCT) [10].

Image compression is a sub-discipline of data compression. A frequently used format for lossless compression is Portable Network Graphics (PNG) which relies on a Lempel-Ziv algorithm and Huffman-coding. Two types of image compression techniques commonly used for lossy compression are the DCT and Discrete Wavelet Transform (DWT). DCT is used for example in the JPEG format and DWT is used in the JPEG 2000 (JP2) format [10].

The Lempel-Ziv algorithm used in PNG replaces repeating sections of data with a single copy and references to that copy. Subsequent Huffman coding is a form of entropy coding which estimates the number of occurrences of symbols and encodes symbols with higher probability using the least amount of bits.

DCT is a type of Fourier transform, hence it expresses finite data as a sum of cosines. In contrast to the Fourier transform itself, DCT employs only real components. DCT compresses data in form of discrete data blocks. Various block sizes can be used, e.g. an 8×8 pixel block which is used in JPEG compression [10]. The frequency response of the DCT improves with longer harmonic functions. Hence, DCT does not have favourable compression characteristics if an image has many small details or contours [10].

DWT uses a wavelet transform, hence it expresses data using a set of wavelet base functions. DWT has better properties for finite, non-periodic or non-stationary signals than DCT. Consequently, DWT-based compression is particularly well suited to compress images with high frequency components, such as a star background [10]. The properties of wavelets naturally create a hierarchical structure of the output data after applying a DWT. The hierarchical structure of an image transformed with DWT is depicted in Figure 6. After transforming an image with a DWT, the lowest lowpass subband of an image contains a rough approximation of a given image. Every additional higher frequency subband adds detail to the image. Only a small fraction of an image contains high frequency components, such as contours and sharp details, therefore these subbands can be strongly compressed using lossless algorithms since these subbands mostly contain zeros [10].

A comparison of an image compressed with JPEG and with JP2 with roughly similar compression ratio is presented in Figure 7. Artefacts are created around the contours in the JPEG image in contrast to the JP2 image, despite the higher compression ratio of the JP2 image.

2.4.2 Gaussian Filtering

Gaussian filtering convolves an image with a 2D Gaussian distribution function. The effect of Gaussian filtering is blurring which removes noise and details. The 2D Gaussian function $G(x, y)$ with a mean of zero is defined as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{12}$$

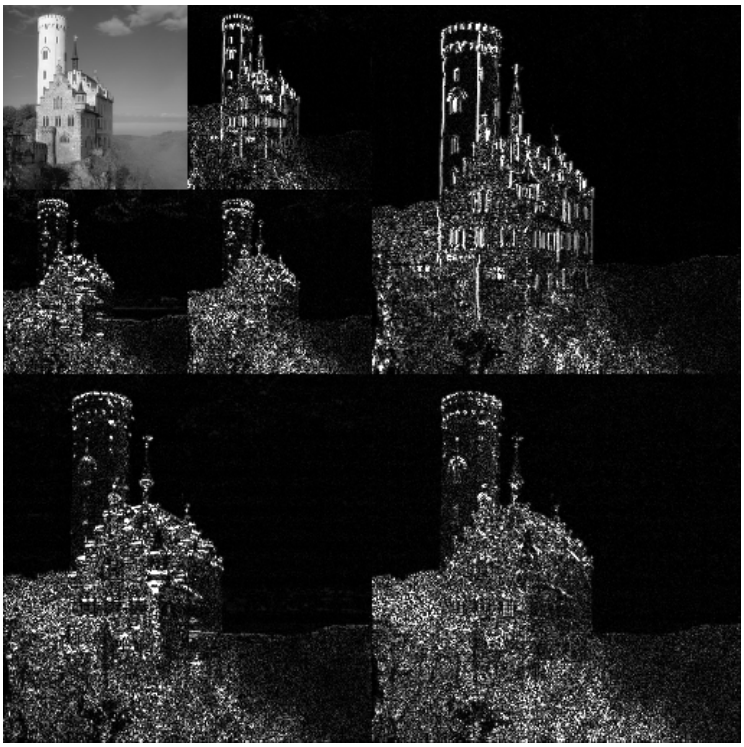
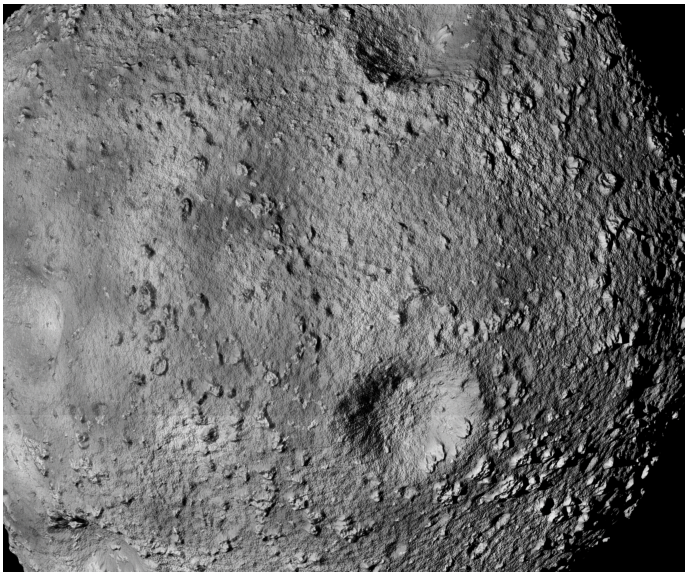


Figure 6: Hierarchical structure of an image transformed with 2-level DWT [18].

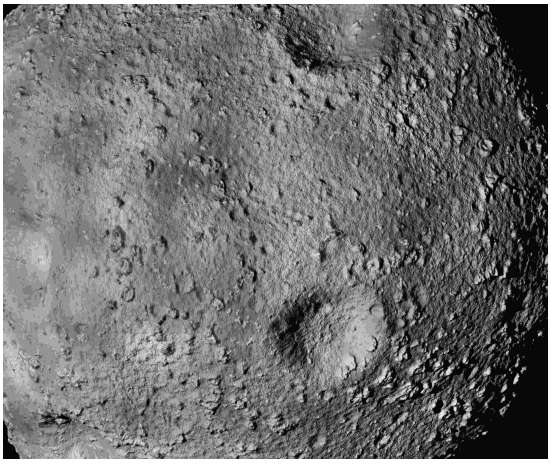
where σ is the standard deviation of the Gaussian distribution, and x and y are coordinates. The Gaussian distribution theoretically extends to infinity. Practical implementations cut the Gaussian distribution at a certain distance from the centre. The kernel size determines the range over which the filter is applied. Since 99 % of the distribution falls within three standard deviations, the kernel size is related to the standard deviation of the Gaussian distribution. Furthermore, the Gaussian distribution is discretised in a digital system. The 2D Gaussian filter is rotationally symmetric and larger standard deviation creates a stronger blurring effect due to a wider peak.

2.4.3 Down-sampling with Local Means

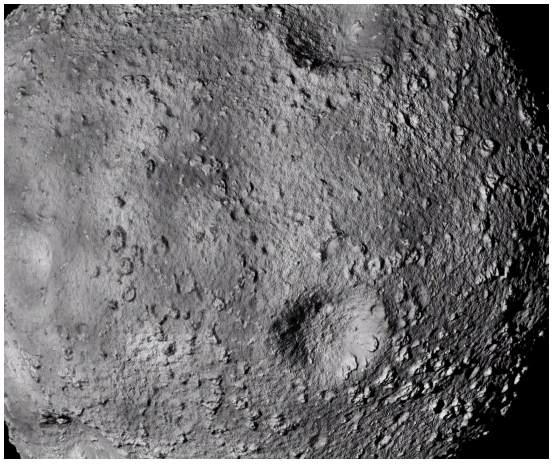
Down-sampling is the process of reducing the number of pixels in an image. Down-sampling with local means refers to replacing a set of pixels with a single pixel having the average value of the pixel set. If an image is reduced by a factor of two, the values of four pixels are averaged to obtain a single output pixel. In the example in Figure 8, a 4×4 matrix is down-sampled by a factor of two resulting in a 2×2 matrix.



(a) Original image without compression.



(b)



(c)

Figure 7: Comparison of images compressed with JPEG and JP2. Despite a higher compression ratio, the JP2 image produces a better result for a SSSB image. (a) Raw image without compression. (b) Image compressed with DCT using the JPEG format with a compression ratio of 61.2:1. (c) Image compressed with DWT using the JP2 format with a compression ratio of 64.3:1.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \rightarrow \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix}$$

Figure 8: A 4×4 matrix down-sampled with local means by a factor of two resulting in a 2×2 matrix

3 Space Imaging Simulator for Proximity Operations

SISPO is a software package that integrates trajectory simulation with physics-based rendering, image compression and 3D reconstruction. These functionalities are split into three sub-packages. The first sub-package uses Keplerian orbit data for a SSSB and a simplified definition of the encounter geometry for the spacecraft to propagate realistic trajectories and render an image series of the encounter. The second sub-package provides various algorithms for image compression and decompression. The third sub-package uses images to reconstruct a textured 3D model using the SfM technique. The three sub-packages combined provide a processing pipeline from an initial 3D model to a reconstructed 3D model via rendered and compressed images. SISPO is a Python software package which is hosted on a public GitHub repository under a BSD-2-Clause license and maintained by the author [78]. SISPO is based on source code developed by Mihkel Pajusalu [68]. The original source code consisted of a rendering and reconstruction pipeline. In this thesis, the original processing pipeline was developed further into a software suite and a compression module was added. In the following, a description of the functionality and input parameters provides insights into the development process and design choices made during development.

3.1 Simulation Package

The simulation sub-package handles orbit propagation for the SSSB and spacecraft as well as rendering photo-realistic images. Propagation and rendering are two separate steps. First the SSSB and spacecraft are propagated along their trajectories using the space dynamics library Orekit [72]. Subsequently, trajectory and attitude data is used to render four images per frame which are composed and photometrically calibrated to form the final rendering output. Images during rendering and calibration use the High Dynamic-Range (HDR) image format OpenEXR to minimise information loss in intermediate steps [26]. Reduced quality images of the rendered images are stored as PNG files for quick preview. Raw data of intermediate steps is saved for possible in-depth analysis. The raw data consist of the four rendered images, their PNG previews, metadata, the Blender scene and the star data. The simulation sub-package was developed to contain all general information about the environment in the Environment class in order to have one consistent instance of time, constants and parameters.

3.1.1 Propagation

Propagation is based on the Python bindings of the Orekit library. The Python bindings run a Virtual Machine (VM) to execute the underlying Java code. Orekit requires physical data which is distributed with the simulation sub-package. The VM and physical data are initialised in the simulation module. Other modules are imported subsequently to only execute one instance of the VM. Having only one instance of the Java VM executing reduces resource consumption and provides a

single consistent set of physical constants. During propagation, Orekit determines state information of the SSSB and the spacecraft for each sample along the trajectory. The state information includes the date, position and the rotation angles of the SSSB. Propagation is based on Orekit's KeplerianOrbit and KeplerianPropagator classes. The input parameters required to define the SSSB trajectory are presented in Table 2.

Table 2: Input parameters that define the SSSB orbit in SISPO. The parameters represent the modified Keplerian elements presented in Section 2.1.3.

Parameter [Unit]	**Type**
a [au]	float
e [-]	float
i [rad]	float
omega [rad]	float
Omega [rad]	float
M [rad]	float
Date of the orbital parameters [-]	dict[1]

The spacecraft trajectory is not defined by Keplerian elements but calculated relative to the SSSB at closest approach. However, the trajectory around the SSSB is a closed orbit around the Sun (cf. Section 2.1.3). The parameters presented in Table 3 are used to calculate the state vector of the spacecraft at the encounter which defines the trajectory of the spacecraft. The sssb_state is calculated based on the SSSB input data and the encounter data.

Table 3: Parameters that define the encounter state of the spacecraft in SISPO. The first five parameters are required as input.

Parameter [Unit]	**Type**	**Description**
encounter_distance [m]	float	Minimum distance between SSSB and spacecraft.
with_terminator [-]	bool	Determines whether the terminator is visible at the closest approach.
with_sunnyside [-]	bool	Determines whether the spacecraft passes the SSSB on the Sun facing side or the side facing away from the Sun.
relative_velocity [$\mathrm{m\,s^{-1}}$]	float	Relative velocity of the spacecraft to the SSSB at the encounter.
encounter_date [-]	dict[1]	Date of the closest approach of the spacecraft and the SSSB.
sssb_state [m & $\mathrm{m\,s^{-1}}$]	tuple	SSSB state vector containing 3 position and 3 velocity components at the encounter. The spacecraft encounter state is calculated relative to this state vector. The SSSB state is not required as input since it is calculated based on the SSSB trajectory and encounter date.

Propagation is defined by the duration of the fly-by, number of frames, timesampler mode and a slowmotion factor as presented in Table 4. The timesampler mode determines whether the steps are distributed linear in time (mode 1, default) or whether an exponential model (mode 2) is used which increases the number of frames around the encounter. How many additional samples are taken can be controlled with the slow motion factor. Mode 2 is especially helpful when simulating a long fly-by since far from the SSSB nucleus only minor changes are visible in rendered images.

Table 4: Input parameters that define the propagation step in SISPO.

Parameter [Unit]	**Type**	**Description**
duration [s]	float	Total length of the simulation. The encounter date is reached after half of the duration.
frames [-]	int	Number of frames (samples) taken during the encounter.
timesampler_mode [-]	int	Mode 1 = linear, mode 2 = exponential sampling.
slowmotion_factor [-]	float	Determines how many more state samples are taken around the encounter. Only applies if timesampler_mode is exponential.

3.1.2 SSSB Rendering

The 3D creation suite Blender was selected for rendering within SISPO [27]. Blender provides the path tracing rendering engine Cycles which renders photo-realistic, physics-based images [28].

Rendering requires 3D models of the SSSB, the light reference and the Sun as a light source. The SSSB and light reference objects are kept at the origin while the Sun object and cameras change location. Three scenes with one camera each are used. The scenes are called *SssbOnly* with the *ScCam* camera, *SssbConstDist* with the *SssbConstDistCam* and *LightRef* with the *LightRefCam* camera. The three scenes with their respective cameras are necessary for the composition step described in Section 3.1.4. Each camera is configured with the same physical instrument characteristics provided as input as presented in Table 5. All cameras are perspective Red, Green and Blue (RGB) cameras with a FoV clipped to the interval $[1 \times 10^{-5}, 1 \times 10^{32}]$. The star background is not rendered with Blender and therefore described separately in Section 3.1.3.

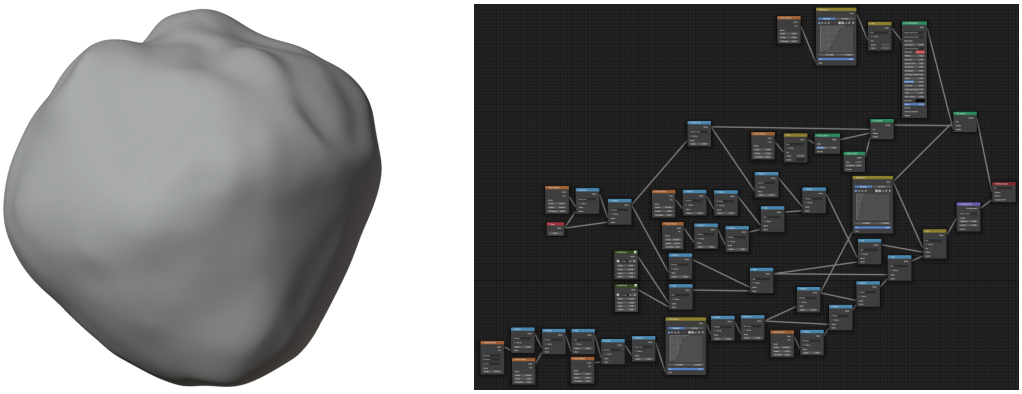
The precise shape and surface details of an SSSB are not provided as texture files and exact shape models within SISPO. The raw Blender model is a smooth body without any textures as shown in Figure 9a. Blender shader nodes are used to create the surface shape and texture during rendering. Blender nodes are a method of visually programming complex algorithms. Blender nodes are combined in a network or tree. An overview of the node network used in SISPO's rendering process can be seen in Figure 9b. The network consists of two main trees, one for shading and one

[1] A Python dictionary with an int for year, month, day, hour, minute and float for second.

Table 5: Input parameters that define an instrument in SISPO.

Parameter [Unit]	**type**	**Description**
res [-]	(int, int)	Number of pixels of the CCD in x- and y-axis. Two values need to be given as either list or tuple.
pixel_l [µm]	float	Length of a pixel of the CCD .
focal_l [mm]	float	Focal length of the optical bench.
aperture_d [cm]	float	Diameter of the aperture of the optical bench.
wavelength [nm]	float	Centre wavelength of the instrument.
quantum_eff [-]	float	Optical efficiency of the optical system, including optical bench, quantum efficiency of the CCD.
color_depth [bit]	int	Bit depth of the CCD.

for displacement. The two trees are interlinked, i.e. displacement influences surface colour and vice-versa. Procedural terrain generation allows to use the same model on a large range of distances since surface details are generated automatically [44].



(a) (b)

Figure 9: (a) Raw, smooth 3D model before applying texturing and displacement shaders. (b) Shader node network overview depicting the shaders used for creating the final SSSB surface and terrain. The overview shall provide an idea of the complexity of the network, a larger version of this image with higher resolution is presented in Appendix A.

The shader implementation creates a material that combines surface shading and displacement. The used nodes are grouped into five categories. Math nodes, texture nodes, colouring nodes, vector nodes and shader nodes. A detailed description can be found in the Blender manual [44]. Only the most relevant information for the implementation in SISPO is provided.

Math nodes include mathematical operators such as add, subtract, multiply, greater than or maximum of a value. Vector nodes allow transformations in 3D space such as translation, scaling or displacement. SISPO uses the Displacement node to move the surface along the surface normals. Colouring nodes are used to alter colours. SISPO uses the RGB Curves node which allows manipulation of colours

per channel using a curve that maps the input to output values. Mix nodes are used to combine colours. Texture nodes provide procedural creation of texture by calculating colours based on mathematical functions and model coordinates. The Noise Texture node creates textures based on Perlin noise. The Voronoi Texture node creates textures using Voronoi patterns, or Worely noise. Three shader nodes are used in our model, the Principled BSDF, Diffuse BSDF and the Mix Shader node. The Principled BSDF node combines many shading features into a single node. The BSDF at the output includes a mix of sheen tint, surface roughness and index of refraction, among others. The Diffuse BSDF node provides diffuse reflection using a Lambertian and Oren-Nayar model. The Mix Shader node combines its input shaders with a given probability for using either of the input shaders as its output.

The resulting shader based Blender material can be applied to any 3D model inside Blender. SISPO can be used to create various SSSBs by changing parameters or modifying the 3D model itself. Four rendered example images created with the described shader node network are presented in Figure 10.

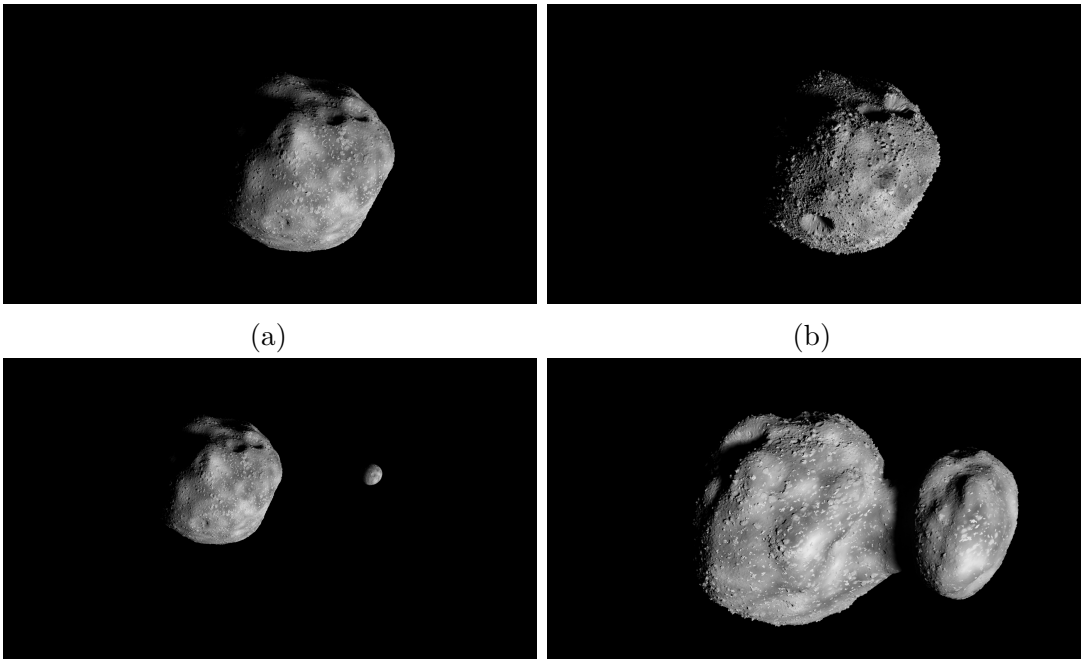


Figure 10: Four rendered images using the Blender material used in SISPO. (a) Render output with default shader parameters using the 3D model presented in Figure 9a. (b) Rendered image with changed shader parameters using the same model as in (a). (c) Render of a binary system with the same main body as in (a) and default shader parameters. (d) Bilobate body using default shader parameters.

Blender is interfaced using its Python bindings which need to be compiled manually. Some predefined settings are used for each scene and some parameters can be provided as input. The paths to the ".blend" files of models for the Sun, light reference and SSSB need to be provided with the object name within the respective file.

Since images are composed and calibrated in a later step, described in Section 3.1.4, the settings shown in Table 6 are selected to not alter images with Blender while saving the raw rendered images.

Table 6: Blender settings that define the colour space and are fixed to not alter images after rendering.

Name in Blender	Value	Description
Color mode	RGBA	Use Red, Green, Blue and Alpha (RGBA) channels.
Exposure	0	Exposure in stops, applied before display transform.
Sequencer colorspace	Raw	Colour space sequencer uses linear colour space.
View transform	Raw	No colour space conversion.
Look	None	No artistic effect is applied before colour space conversion.
Film transparent	True	World background is transparent, i.e. alpha values can be used for image composition and occultation.

The rendering performance of Blender can be configured depending on the available hardware and required image quality with the parameters presented in Table 7.

Table 7: Input parameters that affect rendering performance and image quality in SISPO.

Parameter	Default Value	Description
device	AUTO	Device used for rendering, either "AUTO", Central Processing Unit (CPU) or GPU. "AUTO" attempts to use GPU, if no GPU is available uses CPU.
tile_size	512 (GPU)/ 128 (CPU)	Size of a rendering tile. When rendering with a GPU a bigger tile size tends to have better performance. When rendering with a CPU, a smaller tile size tends to have better performance[2].
samples	48	Number of samples rendered per pixel.

Due to problems during rendering, Blender scenes use kilometres instead of metres, i.e. one Blender unit corresponds to 1 km. This is the only deviation from SI base units within SISPO.

3.1.3 Star Rendering

The background stars are rendered based on the star catalogue USNO CCD Astrograph Catalog 4 (UCAC4). The UCAC4 catalogue includes stars up to magnitude

[2] A more detailed description is found at https://docs.blender.org/manual/en/latest/render/cycles/render_settings/performance.html?highlight=tile%20size

16. The FoV is determined based on the spacecraft camera in the *SssbOnly* scene which represents the FoV of the spacecraft instrument. The edges of the FoV are calculated using Equation 1.

SISPO retrieves the list of stars in the FoV. The interface for the UCAC4 is a command-line tool u4test.exe [32]. The command-line interface requires the FoV defined as right ascension α_r, declination δ, width and height.

The conversion of the vectors into their respective right ascension and declination is a transform between Cartesian coordinates to spherical coordinates. The relation is defined as

$$\delta = \arcsin(z), \tag{13}$$

$$\alpha_r = \arccos\left(\frac{x}{\cos\delta}\right) + \pi, \tag{14}$$

where δ is the declination, α_r the right ascension, x is the x-coordinate and y is the y-coordinate. Using this information the star catalogue returns a list of visible stars including their right ascension, declination and magnitude.

The star background is created by generating an image with four times the size of the final image. The coordinates are converted from right ascension and declination of each star into pixel coordinates of the image frame using Equation 7. The pixel coordinates are multiplied with two to fit the oversized image.

The pixel value of each colour channel is set to the flux calculated from the magnitude as defined in Equation 2. The resulting image is Gaussian filtered and down-sampled to the final image size using local means (cf. Section 2.4.3). Star background images are stored in the OpenEXR format with all alpha channel values equal to one. An example image is presented in Figure 11.

3.1.4 Image Composition

Rendered images vary in brightness thus photometric calibration is necessary. Star maps are not rendered with Blender, thus the raw images need to be combined. Optionally, the combined images can be transformed to have a specified colour depth of a CCD (cf. instrument input parameters in Table 5). Consequently, the composition process has two steps and an optional third step.

First, the SSSB image and the star background are calibrated. Calibration compares a rendered image of a calibration disk with theoretically calculated values for the same conditions. In the second step, these images are combined to create an SSSB image with star background. Occultation of stars by the nucleus is included in the image combination.

Composition begins with a set of four images with the prefixes *SssbOnly*, *SssbConstDist*, *LightRef* and *Stars*. An example set is shown in Figure 12. *SssbOnly* represents the FoV of the spacecraft instrument. *SssbConstDist* is a follower camera at a constant distance of 1000 km from the SSSB. *LightRef* contains a calibration disk. *Stars* is an image of the star background. The *SssbOnly* and *Stars* are rendered for the same FoV.

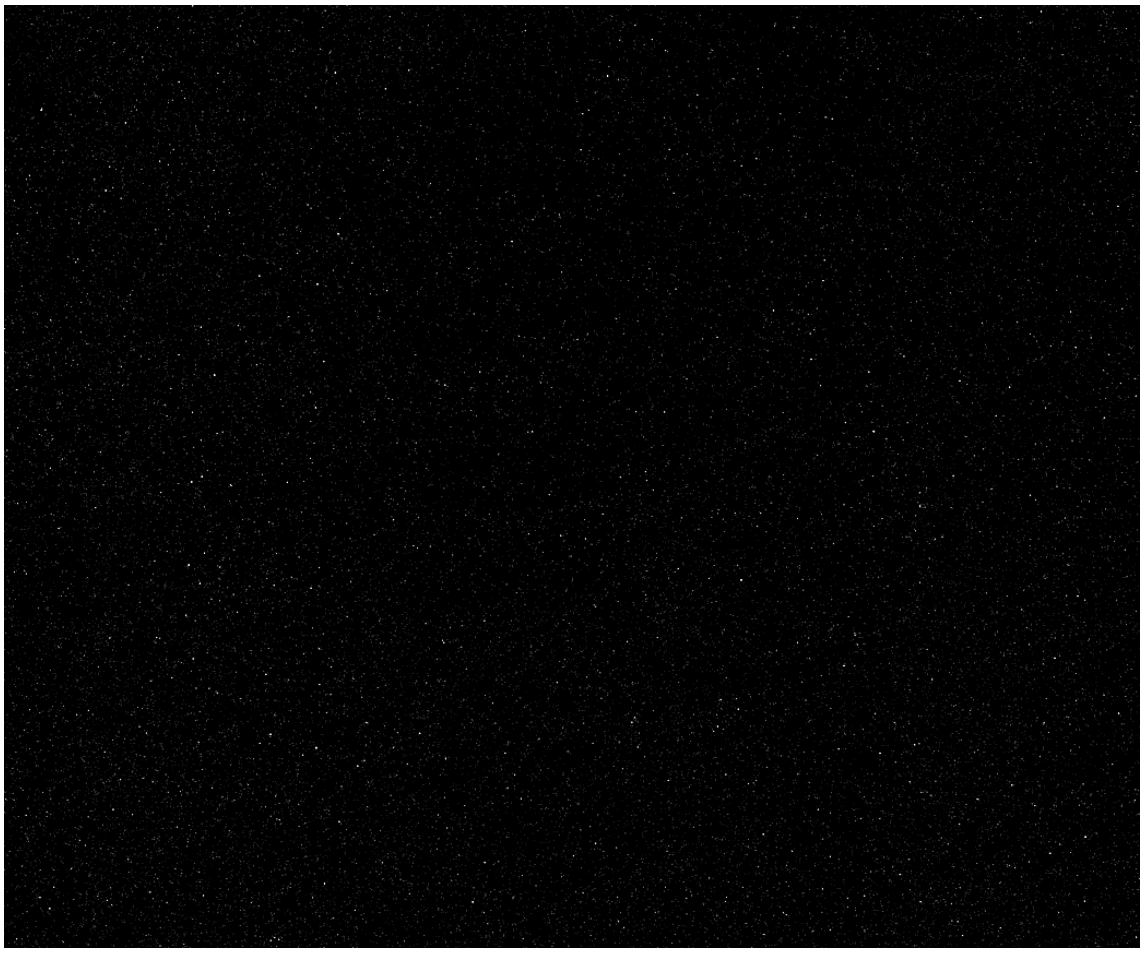


Figure 11: Rendered star map containing several thousand stars.

Photometric calibration is based on the V-band of the UBV system as described in Section 2.2.4. The UBV system was created before CCDs were frequently used in imaging. The UBV system does not consider the higher sensitivity in the red and infrared spectrum of CCDs. However, the UBV system was implemented because of its simplicity.

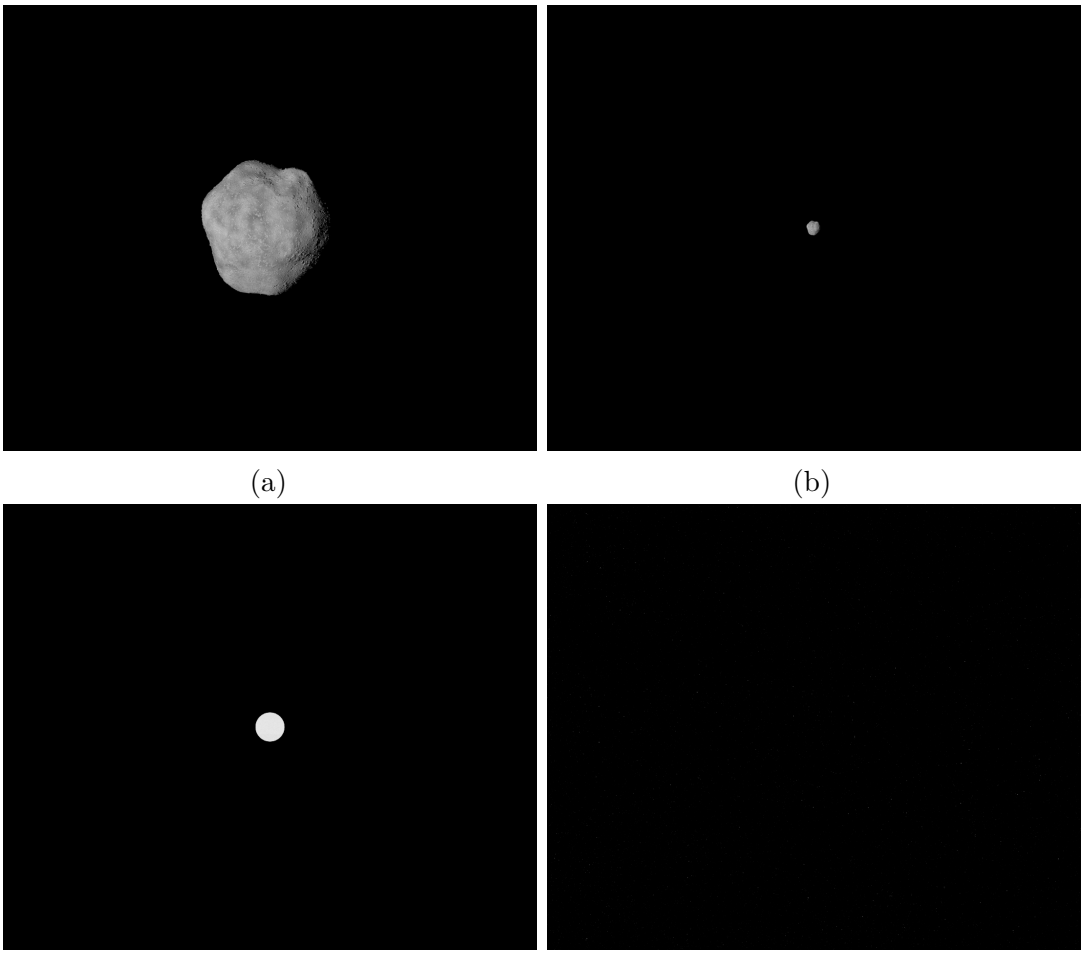
The calibration reference is the solar photon flux in the V-band. The Sun has a magnitude $m_{\odot} = -26.74$ at a distance of 1 au. Using Equation 3 and the constants for the V-band from Table 1, the solar photon flux density at 1 au becomes $F_{\odot} = 4.367\,152\,06 \times 10^{20}\,\mathrm{s^{-1}\,m^{-2}}$.

The spacecraft is typically not at a 1 au distance to the Sun, therefore the flux density is scaled using an inverse square law defined as

$$F_d = F \times \left(\frac{1\,\mathrm{au}}{d}\right)^2, \tag{15}$$

where F_d is the scaled photon flux density, F is the flux density at 1 au and d is the spacecraft's distance from the Sun in au.

Subsequently, the reference photon flux for one pixel is calculated using Equation 4 and the scaled photon flux density F_d.



(a) (b)

(c) (d)

Figure 12: Example set of images used for composition to the final rendering output. (a) *SssbOnly* image which represents the instrument FoV. (b) *SssbConstDist* image of a camera that follows the nucleus at a constant distance. (c) *LightRef* scene which contains the light reference used for calibration. (d) *Stars* image which represents the star background in the instrument FoV.

The calibration factor for each pixel of the SSSB image is obtained using Equation 5. The reference intensity I_{ref} of the image is calculated as the mean intensity of 70×70 pixels of the centre of the light reference image.

If the rendered image is far away from the SSSB nucleus, the apparent size of the nucleus is too small to be used for calibration. If the SSSB nucleus is only in a few pixels of the image the intensities are not rendered correctly. In such a case, a point source SSSB image is generated and used for calibration in combination with the *SssbConstDist* image. The point source image is created by Gaussian filtering a single white pixel at the centre of an oversized image. The image is oversized by a factor of five and subsequently down-sampled to the same size as the *SssbOnly* image using local means as described in Section 2.4.3.

For the star background, every pixel value v is calibrated using

$$v = v_0 \times F_0 \times A \times \frac{F_{stars}}{S_{stars}}, \tag{16}$$

with v_0 being the original pixel value, F_0 being the flux at a magnitude m= 0, A being the aperture area, F_{stars} being the total flux of visible stars and S_{stars} being the summed pixel value of one channel of a star map.

The star background and calibrated SSSB image are merged considering alpha channels. The alpha channel is used to implement occultation of background stars by the nucleus. The film_transparent option of Blender is activated to make the scene background transparent, i.e. only pixels containing the SSSB have an alpha value larger than zero.

The composed image is multiplied by the efficiency of the instrument. The efficiency parameter combines all efficiencies to convert incoming photon flux into pixel value, not only the CCD quantum efficiency. A diffraction pattern and noise based on a Poisson distribution are added. The diffraction pattern can be approximated with a Gaussian filter. The standard deviation for the Gaussian approximation of a diffraction pattern is $\sigma \approx 0.45\times\ \lambda\ \times\frac{f}{D}$ when the Gaussian profile should contain the same energy as the diffraction pattern [102]. Therefore, the standard deviation σ for Gaussian filtering in SISPO is defined as

$$\sigma = 0.45 \times \lambda \times \frac{f}{D} \times \frac{m_d}{l_{pix}}, \tag{17}$$

where λ is the observed wavelength, f is the focal length, D is the aperture diameter, l_{pixel} is the length of one side of a pixel and m_d is a multiplier for systems that are not diffraction limited. A perfectly diffraction limited system would have $m_d = 1$). SISPO uses $m_d = 2$ to simulate systems that are not diffraction limited.

The noisy image is scaled to an interval $[0, 1]$ by dividing through the maximum value. If a point source SSSB is used, the maximum value is clipped to five times the maximum of the SSSB point source reference if the maximum value of the merged image is above this threshold. Figure 13 shows the composed image which is based on the image set presented in Figure 12.

The additional third step allows to scale the image values to the colour depth of a CCD (cf. Table 5). The maximum permissible colour depth is 16 bit due to an increase in code complexity for higher bit depths. Colour depth clipping can be turned off by setting the with_clipping parameter to false. By default, colour depth clipping is used since SISPO aims to realistically simulate the imaging capabilities of a spacecraft.

An additional feature is to add an information box that includes the spacecraft distance to the SSSB and the date of the frame in the lower right corner of the images. The infobox can be added by setting the with_infobox parameter to true.

Throughout the composition process, the astropy package [73, 91] handles unit conversion between quantities. Down-sampling and Gaussian filtering are handled by OpenCV during the composition process [20].

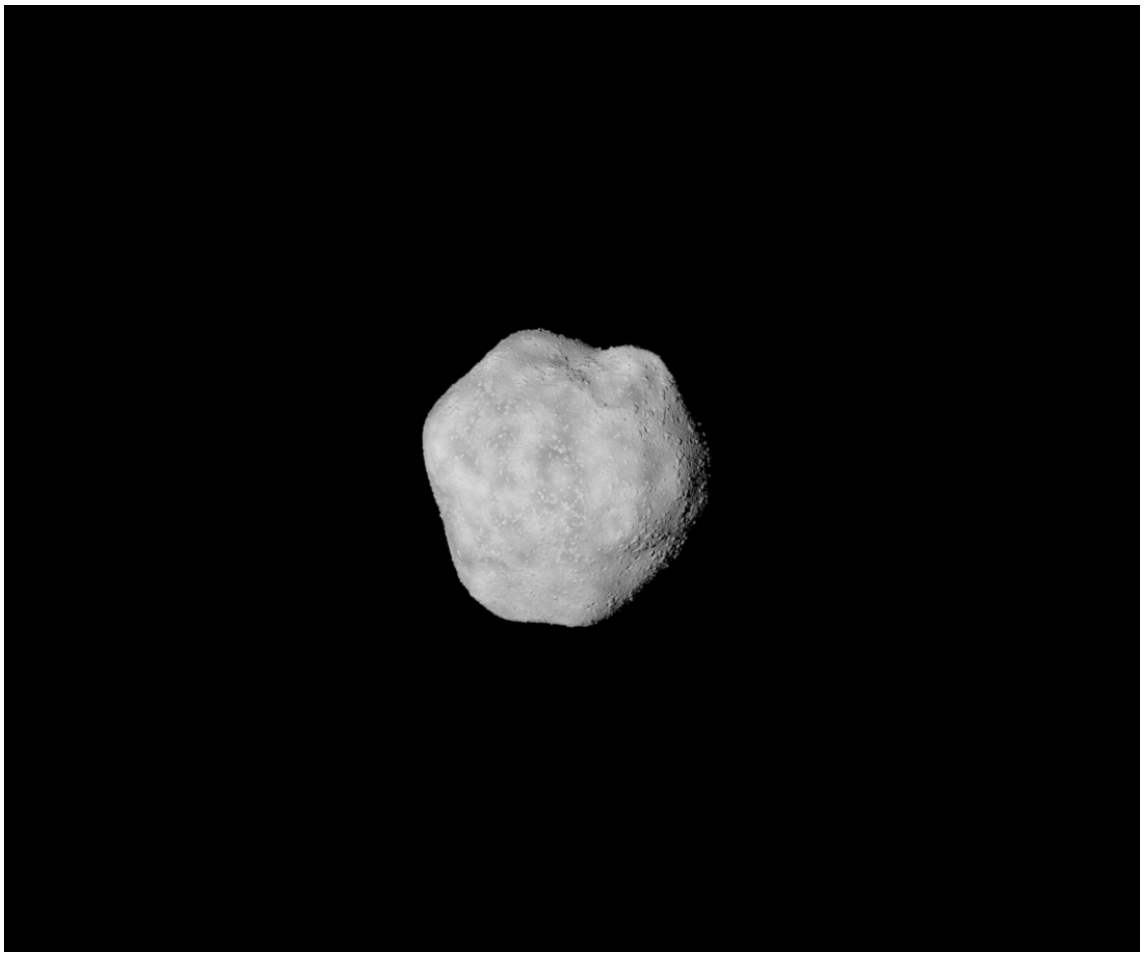


Figure 13: Composed image of the four images shown in Figure 12. No stars are visible in the background because the nucleus is much brighter than the brightest background star.

3.2 Compression Package

The compression package provides various compression and decompression algorithms. The algorithms can be tested in various mission scenarios and image series to investigate the impact of compression and decompression on the scientific return of images. Table 8 presents the available compression and decompression algorithms.

Independent of the algorithm, files are handled equally. Images are loaded into NumPy arrays in SISPO to have a common raw format. Before encoding, the information is reduced to 8 bit since the reconstruction module only works correctly with images of 8 bit colour depth correctly. Images are encoded with the selected algorithm and converted into a binary stream. A copy of this data is saved as a file in the raw folder. The raw binary data is converted back to a NumPy array and decoded. The decompressed data is stored as a PNG file to not lose any further information.

The formats PNG, JPEG, JP2 and TIFF are implemented using OpenCV. There

Table 8: Lossless and lossy compression algorithms and image formats provided in SISPO. The implementation uses either native Python libraries* [30], OpenCV† [20] or native OpenEXR‡ [26]. The Tagged Image File Format (TIFF) format can use either a lossless or lossy compression algorithm.

Lossless Formats	**Lossy Formats**
PNG†	JPEG†
OpenEXR‡	JP2†
bzip2*	TIFF†
gzip*	
Lempel–Ziv–Markov Algorithm (LZMA)*	
zlib*	
TIFF†	

are various settings for each format[3].

The most relevant input parameter to all compression algorithms is the "level" argument in "settings" which describes either the compression level or the quality level. The algorithms bz2, gzip, LZMA, zlib and PNG use a compression level from one to nine where a higher number means more compression. The formats JPEG and JP2 use the quality level which is defined from zero to ten where a higher number means less compression. Other input parameters to the compression package are presented in Table 9.

Table 9: Input parameters that define the compression behaviour in SISPO.

Name	**Default Value**	**Description**
algo	PNG	Compression algorithm, for a list of available algorithms see Table 8
settings	{"level": 9}	Python dictionary including all settings for compression algorithms. For a description of the possible settings see the code or documentation of the providing library.
img_ext	PNG	Image extension to search for when loading images for compression.

Compression and decompression use a simple implementation of multi-threading to execute several image compression and decompression processes concurrently to reduce execution time. Images are only loaded during the compression and decompression process and are removed from memory directly afterwards to reduce memory consumption.

[3] A detailed description can be found at https://docs.opencv.org/4.1.2/d4/da8/group__imgcodecs.html#ga292d81be8d76901bff7988d18d2b42ac

3.3 Reconstruction Package

The reconstruction package generates a 3D model of an object based on a set of images. Reconstruction is based on the SfM technique. Two libraries are combined within SISPO to create the full reconstruction pipeline. Open Multiple View Geometry (OpenMVG) is used to create a sparse point cloud [61]. Open Multi-View Stereo reconstruction (OpenMVS) is used to refine the sparse point cloud and create a textured 3D model [64].

The steps of the complete pipeline are:

1. Loading images with associated information such as priors (ImageListing in OpenMVG)
2. Compute visual features (ComputeFeatures in OpenMVG)
3. Match computed features between different images (MatchFeatures in OpenMVG)
4. Generate a sparse point cloud from matched features (IncrementalSfM in OpenMVG)
5. Export sparse point cloud to the OpenMVS format (openMVG2openMVS in OpenMVG)
6. Increase the number of points in the point cloud (DensifyPointCloud in OpenMVS)
7. Triangulate a rough mesh from the dense point cloud (ReconstructMesh in OpenMVS)
8. Refine the rough mesh (RefineMesh in OpenMVS)
9. Apply texture to refined mesh to create final 3D model (TextureMesh in OpenMVS)

OpenMVG and OpenMVS are controlled with numerous parameters. All parameters have default values in SISPO itself which occasionally differ from the original default settings of the software packages. The default settings were adapted to improve results in the SSSB fly-by scenarios.

During the image listing step, the spacecraft locations from the simulations are added as motion priors to improve stability of the SfM algorithms. Priors are used based on the assumption that a rough spacecraft trajectory is known.

Reconstruction is based on two incremental and one global SfM approach provided by OpenMVG. The three algorithms are executed and the number of reconstructed points are compared. The result containing the most points is exported to OpenMVS for further processing. Residuals are not considered hence the selection method might not choose the highest quality point cloud due to a possibly high number of outliers. If the reconstructed 3D model is considered unacceptable, a better model might be created from a sparse point cloud of another SfM algorithm.

The IncrementalSfM2 algorithm is non-deterministic, i.e. results can differ between executions using the same data sets. Consequently, IncrementalSfM2 should be executed several times and the best result should be selected [68].

SISPO saves intermediate results of the reconstruction process, starting from the sparse point cloud and ending in the textured 3D model. Figure 14 depicts the evolution of the sparse point cloud to a textured 3D model and a rendered image used in the reconstruction process for comparison.

The highest priority of the reconstruction pipeline is to reconstruct a model. SISPO attempts to create the most detailed model possible by densifying the point cloud and refining the mesh. If point cloud densification or mesh refinement is unsuccessful, SISPO continues with either the sparse point cloud or the rough mesh.

3.4 User Interface

SISPO is a Python package and therefore the user interface is a Python console. Settings for the three sub-packages along with general settings of SISPO are stored in a JavaScript Object Notation (JSON) file. After importing, SISPO can be executed by one of two equivalent functions sispo.run() or sispo.main(). An example definition.json file is provided with the repository in the data/input folder. The example provides the most important settings. However, more settings exist which can be found in the source code. Especially the SfM software packages use many default settings which can be customised. An explanation of all parameters is found in the documentation of the respective software package.

The high-level behaviour of SISPO can be controlled with a set of execution flags presented in Table 10.

3.5 Performance

The performance of SISPO and its components was analysed during the development to find code section that increase execution time. While reducing the number of dependencies, it was ensured that performance does not degrade.

Two computers were available during performance analysis. A laptop with 8 GB Random Access Memory (RAM), an Intel® Core™ i7-6700HQ with 4 cores at 2.6 GHz and Windows 10. The second computer is a workstation with 16 GB of RAM, an Intel® Core™ i7-8700 processor with 6 cores at 3.2 GHz and Ubuntu 18.04.3 LTS.

3.5.1 Overall Performance

SISPO was executed on the workstation computer to assess overall performance.

Two test cases were analysed with the parameters and main results given in Table 11. The performance assessment of two SISPO trials shows that rendering is the most time consuming process. More than 90 % of the total execution time is dedicated to rendering. Only 5 % to 6 % are used for reconstruction and less than 1 % for compression. Consequently, compression and reconstruction do not contribute much to the total execution time. The difference of the total execution time of the

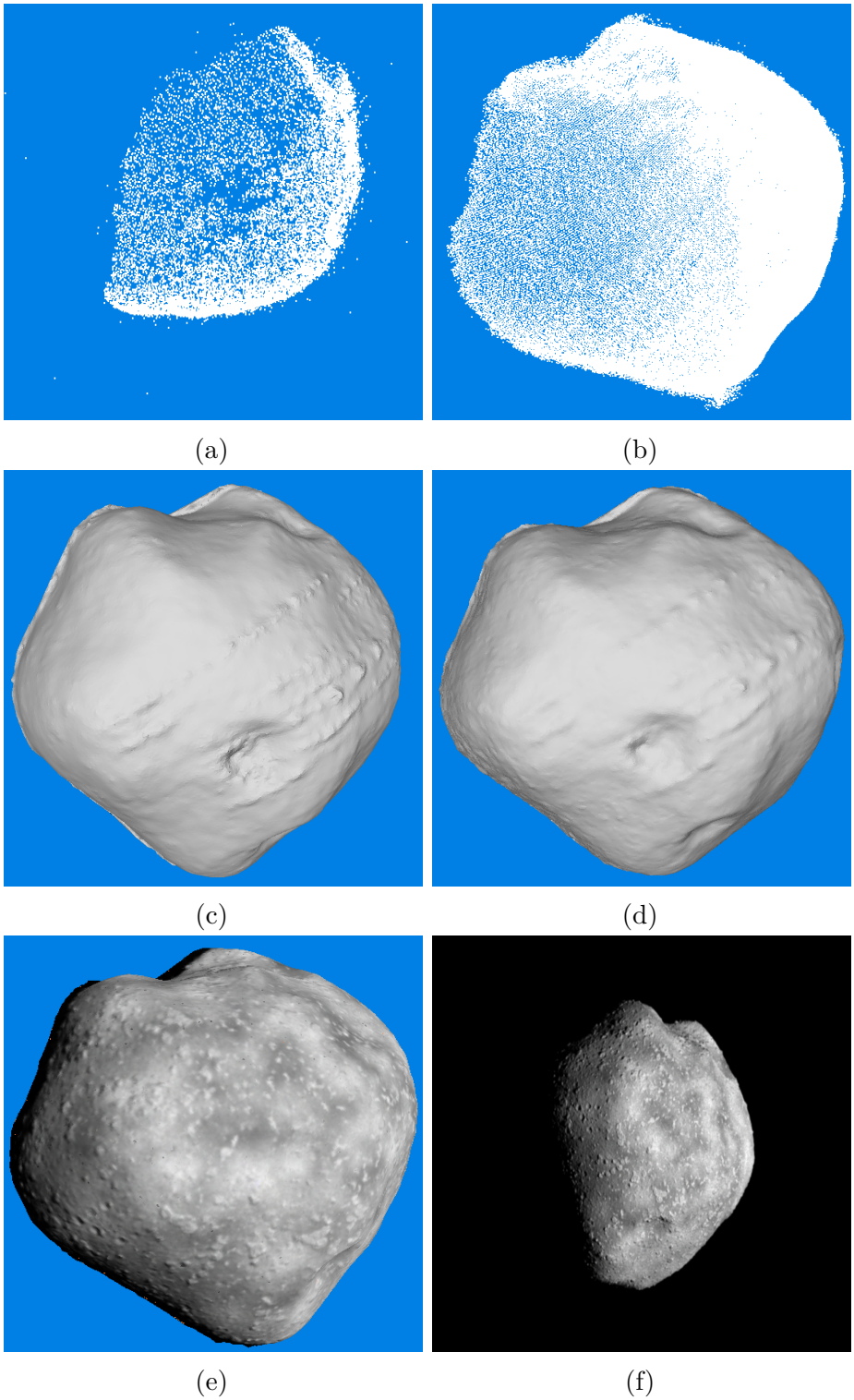


Figure 14: Example images of intermediate results of the reconstruction pipeline. (a) Sparse point cloud created by OpenMVS. (b) Point cloud densified with OpenMVS. (c) Mesh created from the dense point cloud in (b) using OpenMVS. (d) Refined mesh based on the mesh created in (c) using OpenMVS. (e) Mesh textured using OpenMVS. (f) Reference image for comparison with the textured mesh.

Table 10: Input flags to control high-level behaviour of SISPO.

Name	Variable Name	Default Value	Description
–help		—	Prints list of arguments with hints
-i	i	definition.json	Path to a definition file that defines the settings
-v	v	False	Flag to use verbose output, i.e. logging information will also be displayed to console
–profile	profile	False	Flag to use Python's cProfile to profile SISPO execution
–no-sim	with_sim	True	Flag to skip simulation step
–no-render	with_render	True	Flag to skip rendering step
–no-compression	with_compression	True	Flag to skip compression
–no-reconstruction	with_reconstruction	True	Flag to skip reconstruction
–sim-only	sim_only	False	Flag to do simulation step
–sim-render-only	sim_render_only	False	Flag to do simulation and rendering steps
–render-only	render_only	False	Flag to do rendering step
–compress-only	compress_only	False	Flag to do compression step
–reconstruct-only	reconstruct_only	False	Flag to do reconstruction step
–compress-reconstruct-only	compress_reconstruct_only	False	Flag to do compression and reconstruction steps

two trials is explained by the different apparent size of the SSSB. Based on the two examples, the execution time for reconstruction is affected stronger by a decreasing fly-by distance than rendering.

Most execution time of a simulation is related to rendering thus improving the performance of SISPO further requires improving the performance of the rendering process. Either rendering parameters can be changed or the shader implementation can be improved. Parts of the code were optimised before profiling such as multi-threading for compression or replacing the Scikit-Image (skimage) library with OpenCV.

Table 11: Summary of two profiles obtained while executing complete trials of SISPO, i.e. rendering, compression and reconstruction. More than 90 % of total time is spent rendering images.

Distance	**SSSB size**	**Total time**	**Rendering**		**Compression**		**Reconstruction**	
[km]	[km]	[s]	[s]	[%]	[s]	[%]	[s]	[%]
200	10	346939.8	325834.6	93.9	55.6	0.2	21048.2	6
400	10	212209.4	201443.2	94.9	24.7	0.1	10740.1	5.1

3.5.2 Image Processing Benchmark

The original source code used the skimage and OpenCV libraries concurrently. In order to reduce the number of dependencies, relevant functions of the two libraries were compared. Parameters for the OpenCV functions were selected to mimic the respective skimage function. The benchmark compares performance of Gaussian filtering and image down-sampling using local means as described in Section 2.4.2 and Section 2.4.3 respectively. Five images are selected for the benchmark. The image set is presented in Appendix B. Two star maps were selected due to the large variance in the number of visible stars. The selected images contain 1804 and 51338 stars.

The ratio of execution time is used to compare performance of the two libraries. The ratio is defined as

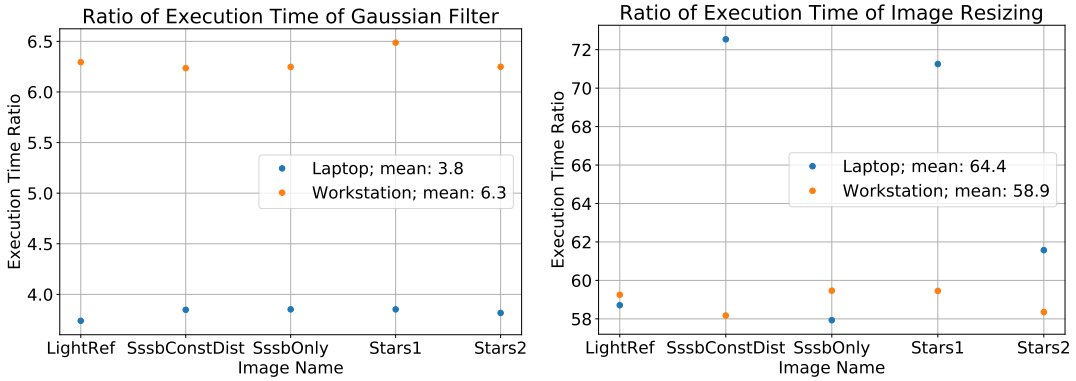
$$Ratio = \frac{T_{skimage}}{T_{opencv}}, \tag{18}$$

where $T_{skimage}$ is the execution time of skimage and T_{opencv} is the execution time of OpenCV. Each command is executed and timed for 1000 trials. The lowest value is selected as result, since higher values are more likely influenced by other processes running on the respective machine than the relevant code snippet itself [31].

Both available computers were used for the image processing performance benchmark. Figure 15 shows execution time ratios and averages for the image set. A ratio larger than one represents a longer execution time of skimage. On average, OpenCV outperforms skimage for both tests on both computers. The maximum absolute difference between pixel values of images is 1.486×10^{-6} and 7.153×10^{-7} for the Gaussian filtered and the resized images respectively. Such differences are not relevant for image colour depths of up to 16 bit.

OpenCV has a performance advantage over the skimage library, hence only OpenCV is used in SISPO. OpenCV might also be used to replace the OpenEXR dependency in the future, if the OpenEXR implementation of OpenCV includes alpha channel support[4].

[4] A GitHub issue was created at https://github.com/YgabrielsY/sispo/issues/128 that links to the relevant OpenCV GitHub issue.



(a) (b)

Figure 15: Comparison of execution time ratios of (a) Gaussian filtering and (b) resizing five images using OpenCV and skimage on two computers. Values > 1 correspond to OpenCV executing faster. The mean values are presented in the legend.

4 Results

Several simulation trials were executed with SISPO to assess the capabilities of SISPO and quality of the output of the three stages, i.e. rendering, compression and reconstruction. Furthermore, the effects of compression using the JP2 format on rendered images and the quality of 3D model reconstruction were investigated. Table 12 shows a summary of the simulated scenarios.

Table 12: Simulation parameters used for investigating capabilities of SISPO. Image compression in each scenario used the formats PNG and JP2 quality levels 1000, 100, 10 and 1 compression to investigate compression effects.

SSSB Size [km]	**Encounter Distance** [km]	**Number of Images** [-]
1	50	120
1	100	120
1	200	120
1	400	120
10	50	120
10	100	120
10	200	120
10	400	120

The hardware presented in Section 3.5 was used to create the results. During the simulation trials, reconstruction was found to be more successful on the Windows computer. Consequently, images were rendered on the Linux computer and reconstructions were carried out on the Windows computer. It was not possible to determine the cause of the higher reconstruction success of the Windows computer.

4.1 Rendering

Some settings were kept constant for all simulations. The instrument settings are presented in Table 13 and the settings for the SSSB in Table 14.

Table 13: Instrument settings used in all simulation scenarios presented in Table 12. The parameters represent the preliminary instrument design presented in [68].

Parameter Name	**Value**
res	2464×2054
pix_l	3.45 µm
focal_l	230 mm
aperture_d	4 cm
wavelength	550 nm
quantum_eff	0.25
color_depth	8 bit

Table 14: SSSB orbit settings used in all simulation scenarios presented in Table 12. The orbital elements and rotation rate of Didymos were used comparable to [68].

Parameter Name	Value
a	1.644 641 475 071 416 au
e	$3.838\,774\,437\,558\,215 \times 10^{-1}$
i	3.408 231 185 574 551 rad
omega	$3.192\,958\,853\,076\,784 \times 10^{2}$ rad
Omega	$7.320\,940\,216\,397\,703 \times 10^{1}$ rad
M	$1.967\,164\,895\,190\,036 \times 10^{2}$ rad
date	2017-08-19T00:00:00.000
rotation_rate	8133.48 s^{-1}
albedo	0.15
max_dim	512

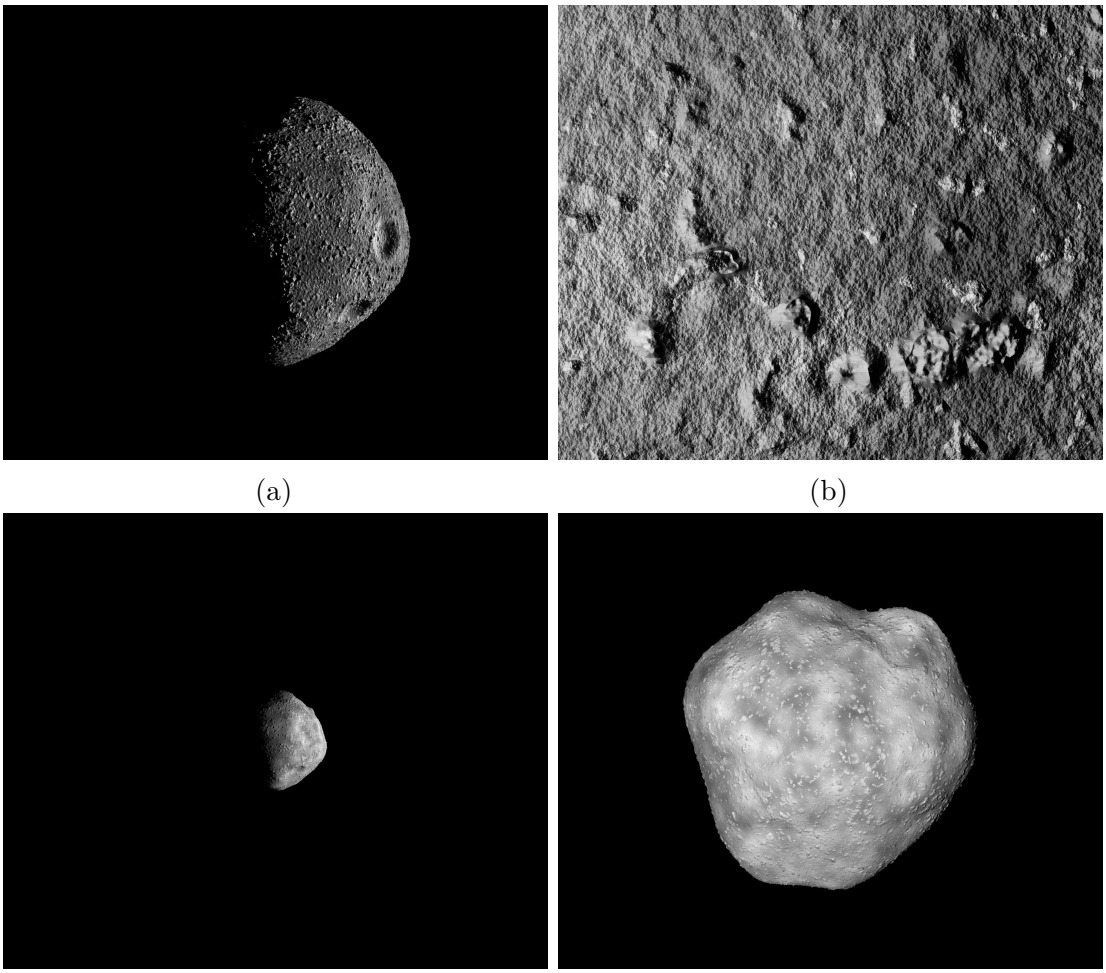
Table 15: Propagation and rendering settings used in all simulation scenarios presented in Table 12. The number of frames was selected on the assumption of a 1 Hz imaging frequency for a fly-by duration of two minutes.

Parameter Name	Value
duration	120 s
encounter_date	2017-08-15T12:00:00.000
frames	120
relative_velocity	10 km s^{-1}
with_terminator	0
with_sunnyside	1
timesampler_mode	1
exposure	0
samples	48
device	GPU
tile_size	512
with_clipping	1

4.1.1 Image Comparison

The overall image quality is compared visually to real images. A set of images at different SSSB distances with varying apparent SSSB sizes is depicted in Figure 16. A set of five images of asteroid Bennu taken during the *OSIRIS-REx* mission is shown in Figure 17a. An image of comet 67P taken by the *Rosetta* spacecraft is presented in Figure 17b. A collection of views of the comet 81P/Wild (81P), also known as Wild 2, is shown in Figure 17c. 81P was imaged during the *Stardust* mission [13].

All objects in Figures 16 and 17 show pits. The overall appearance of the rendered images resembles the smoother pits of Bennu better than the sharper pits of 67P or 81P. The rocks and boulders on Bennu's surface appear similar to the rocks and boulders in the rendered images, especially Figures 16a and 16b. The most pronounced difference between rendered images and the images of 67P or 81P are



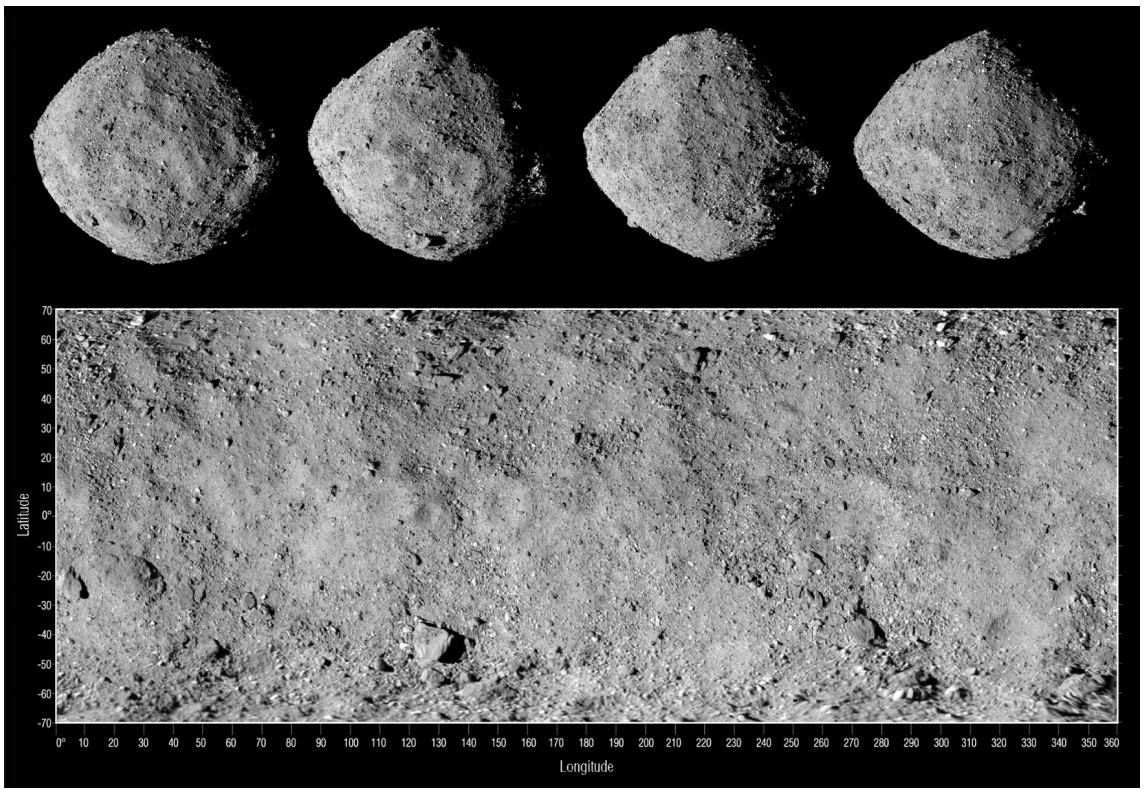
(a) (b)

(c) (d)

Figure 16: Rendered images of differently sized SSSBs from varying distances. (a) Nucleus of 10 km from 566 km distance. (b) Nucleus of 10 km from 106 km distance. (c) Nucleus of 1 km from 149 km distance. (d) Nucleus of 1 km from 50 km distance.

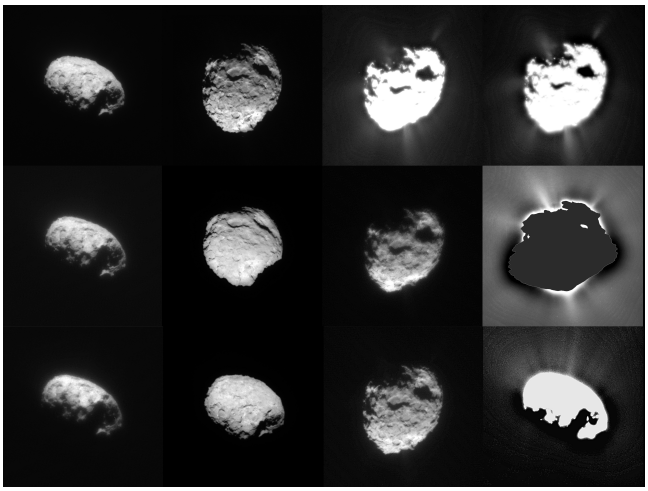
jets. SISPO does not yet contain a gas and dust model that would produce a coma or jets. Therefore, coma and jets are missing in the rendered images. While the surface of both comets feature some boulders, their surfaces are more defined by ridge-like structures. Ridge-like features are missing in the rendered images. Consequently, the rendering images of SISPO are more similar to asteroids than comets.

Procedural terrain generation within SISPO creates results for a large range of encounter distances. Images with different surface distances and SSSB sizes are presented in Figure 16. No visual degradation of surface features and details is visible in Figure 16. Moving closer to the surface reveals more details, such as tiny bumps between larger structures which are not visible from larger distances. The visible quality of surface features is defined by the shader implementation.



(a)

(b)



(c)

Figure 17: (a) Four images of asteroid Bennu and a global surface mosaic. The images were taken by the PolyCam aboard the *OSIRIS-REx* mission [62]. (b) Representative image of comet 67P. The image was captured by the OSIRIS imager aboard the *Rosetta* mission [65]. (c) Image collection of comet 81P taken during the *Stardust* mission by its navigation camera [85].

4.1.2 Image Composition

The composition process uses raw images rendered with Blender and produces photometrically calibrated images. An example set of four images consisting of two images before and after calibration is shown in Figure 18. Two effects can be seen. First, the overall brightness differs in the original images. The brightness difference is corrected by calibration in the processed images. Secondly, images become brighter by the composition process. Image brightening originates from scaling images to the interval $[0, 1]$.

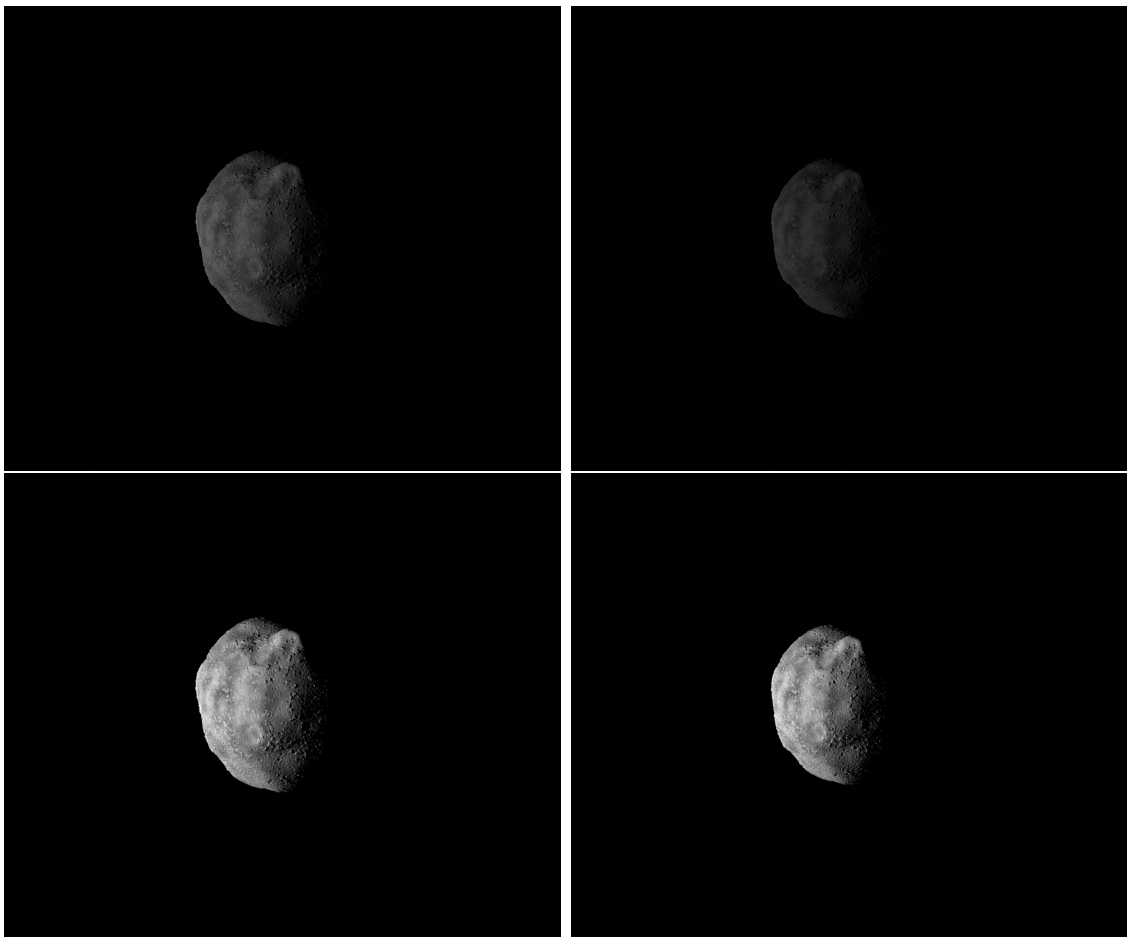
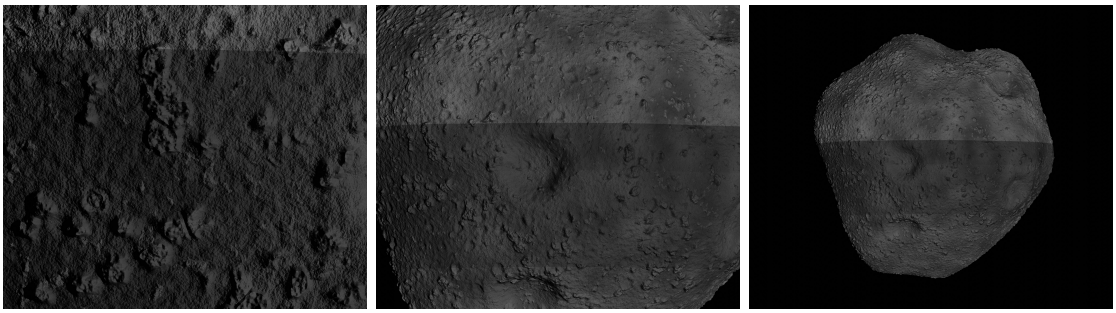


Figure 18: Two consecutive images before (top) and after (bottom) composition and calibration. The nucleus is much brighter than background stars thus no stars are visible in these images after calibration. The four images are reduced to 8 bit colour depth.

4.1.3 Rendering Problems

Rendered images of a fly-by scenario with a 10 km SSSB contain artefacts. Figure 19 shows rendered images of fly-bys with varying fly-by distances. All three images are raw rendered images, before composition. The images show a stripe, a darker patch

across the SSSB with sharp brightness transitions. The stripe is at the same location across the nucleus in all three images. The stripe artefact does not appear in all images with a 10 km SSSB and not for other SSSB sizes. The most likely explanation are errors while scaling the nucleus from the original 1 km to the 10 km model issues with the shader implementation.



(a) (b) (c)

Figure 19: Surface of a 10 km SSSB for varying fly-by distances. Rendering artefacts, the stripes, are visible in all images. (a) Fly-by distance 50 km. (b) Fly-by distance 200 km. (c) Fly-by distance 400 km.

A second problem occurred in a 50 km fly-by simulation with a 1 km SSSB. A single image was found to be darker than any other image in the data set. Figure 20 shows three consecutive images. The middle image is overall darker except for a small patch of white pixels. Three pixels are much brighter than any other pixel in the image. The bright pixels are called fireflies [92]. Figure 20 contains composed images for better visibility of the artefact. The fireflies are not introduced by the composition process since they exist in the raw rendered image. No second image with the same issue was found in any other data set thus the problem was not investigated further.

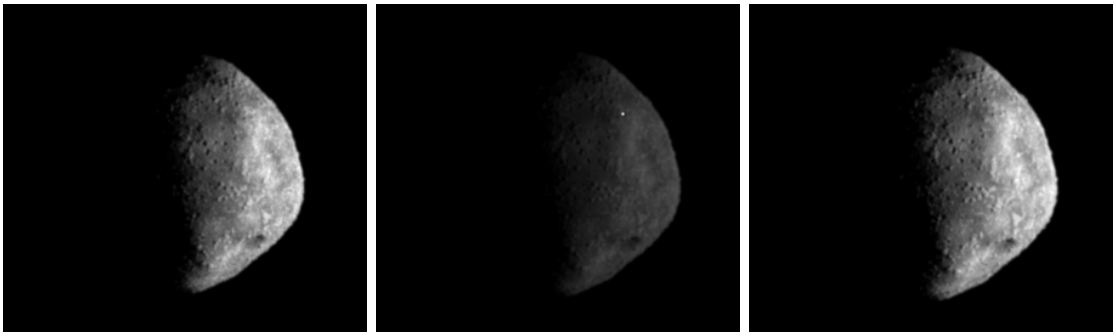


Figure 20: Overall darkened SSSB image due to three pixels being much brighter. These pixels are referred to as fireflies [92]. All three images are composed images for better visibility of the artefact. The images are cropped to display the SSSB nucleus and artefact better.

4.2 Compression

The SISPO software package was used to study the effects of compression in different scenarios. The scenarios are presented in Table 12. Two compression algorithms were used to study compression effects. The PNG format was used because of its wide support among different software packages and JP2 is used as an improved version over commonly used JPEG. The PNG and JP2 formats were selected as general examples to characterise compression effects on reconstruction. Neither PNG nor JP2 are intended to represent image processing on-board a spacecraft. Scenarios with varying SSSB nucleus sizes and fly-by distance were simulated.

The comparison of the different compression algorithms is based on several parameters. The used parameters are the size of the compressed image series, the number of points in the dense reconstructed point cloud, the number of vertices and the number of faces of the refined reconstructed model. These outputs relate to the level of detail of the rendered images since SfM algorithms rely on surface details for reconstruction.

JP2 quality levels presented in the results are the quality levels as defined in the JP2 implementation used by OpenCV, i.e. ranging from 0 to 1000.

4.2.1 Image Quality Comparison

A specific image was selected to compare image quality after different levels of compression. Since reconstruction is mostly influenced by features, a scene with a distance of 50 km to the a 1 km nucleus was selected for comparison. Overall images contain the entire SSSB in the FoV. Close-up images show the area highlighted by the red frame in Figure 21. Overall images show compression effects on an entire scene while close-up images reveal compression effects on surface details. Difference images and histograms are used to investigate compression effects in more detail.

Difference images are created by converting the RGB images to greyscale and calculating the L2-norm after subtracting each pixel from the respective pixel value of the PNG greyscale image. The result is a greyscale difference image showing the L2-norm differences. The zero values in histograms were removed to only show alterations by compression. Therefore, total number of altered pixels and the percentage relative to the original image are presented in the histogram.

Figures 22 to 26 show the compressed overall image, difference histograms and the difference images after compression with JP2 with quality levels 1, 5, 10, 100 and 1000. Quality level 5 is used since most changes due to compression occur at low quality levels.

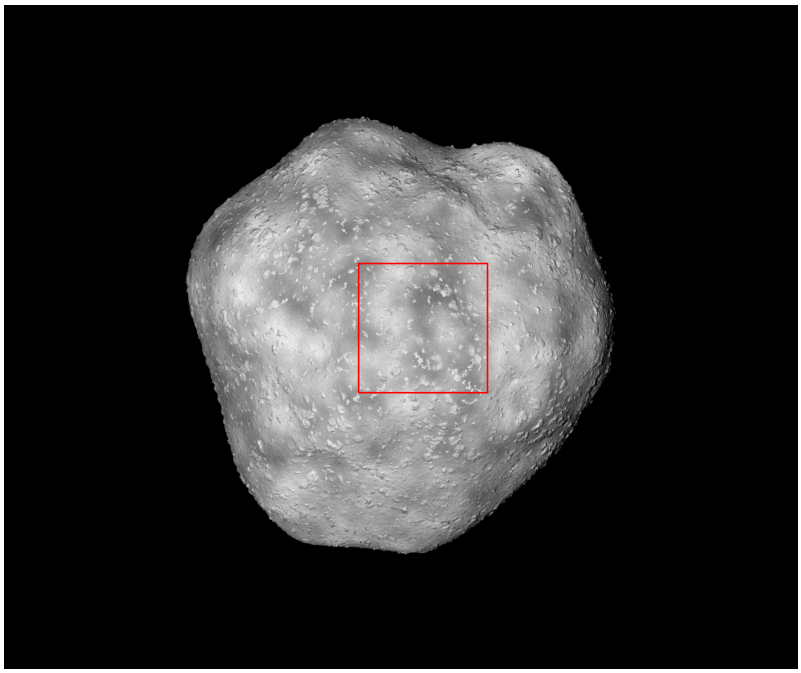


Figure 21: Scene used for quality comparison. The area highlighted in red is studied up closer. The specific area was selected since it includes a wide range of colours and various sized surface features.

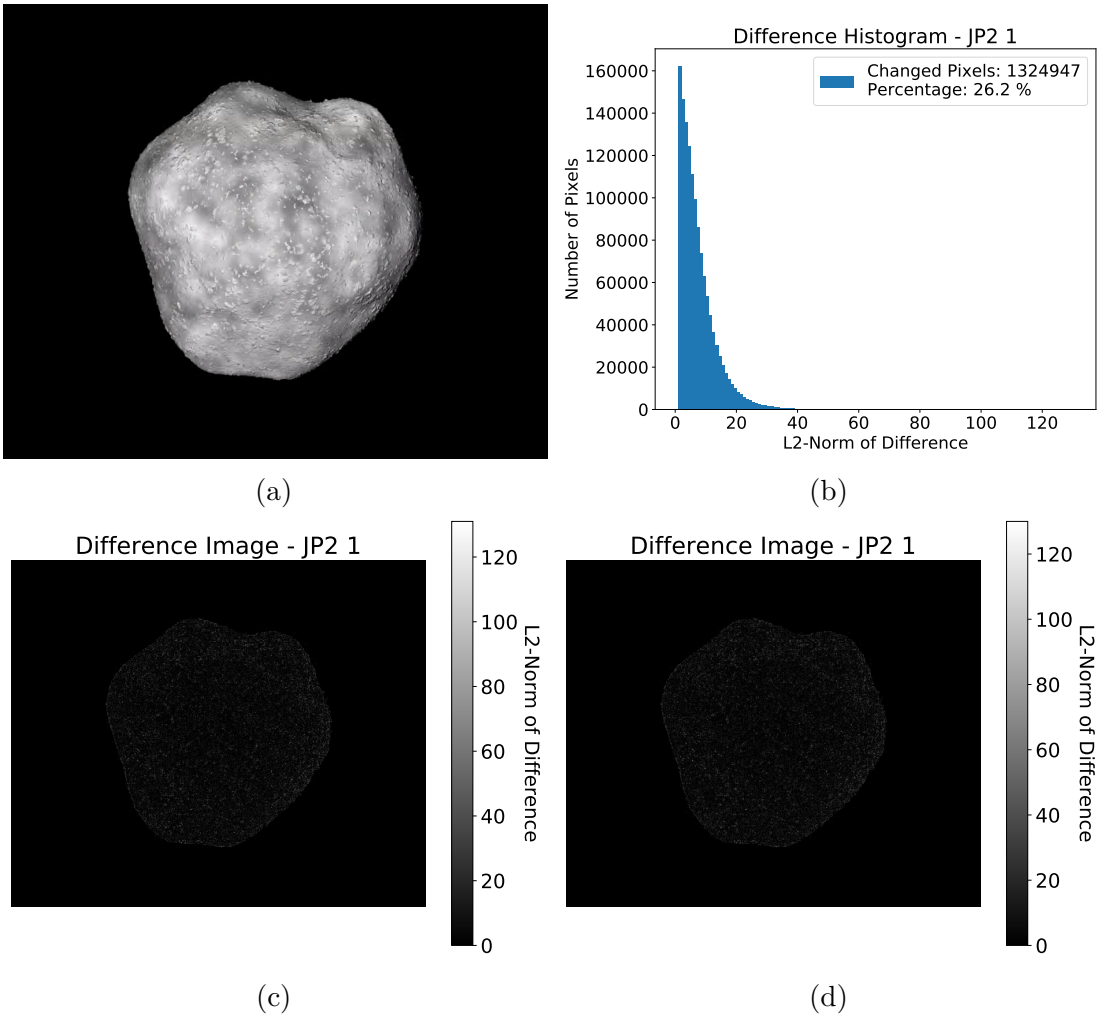
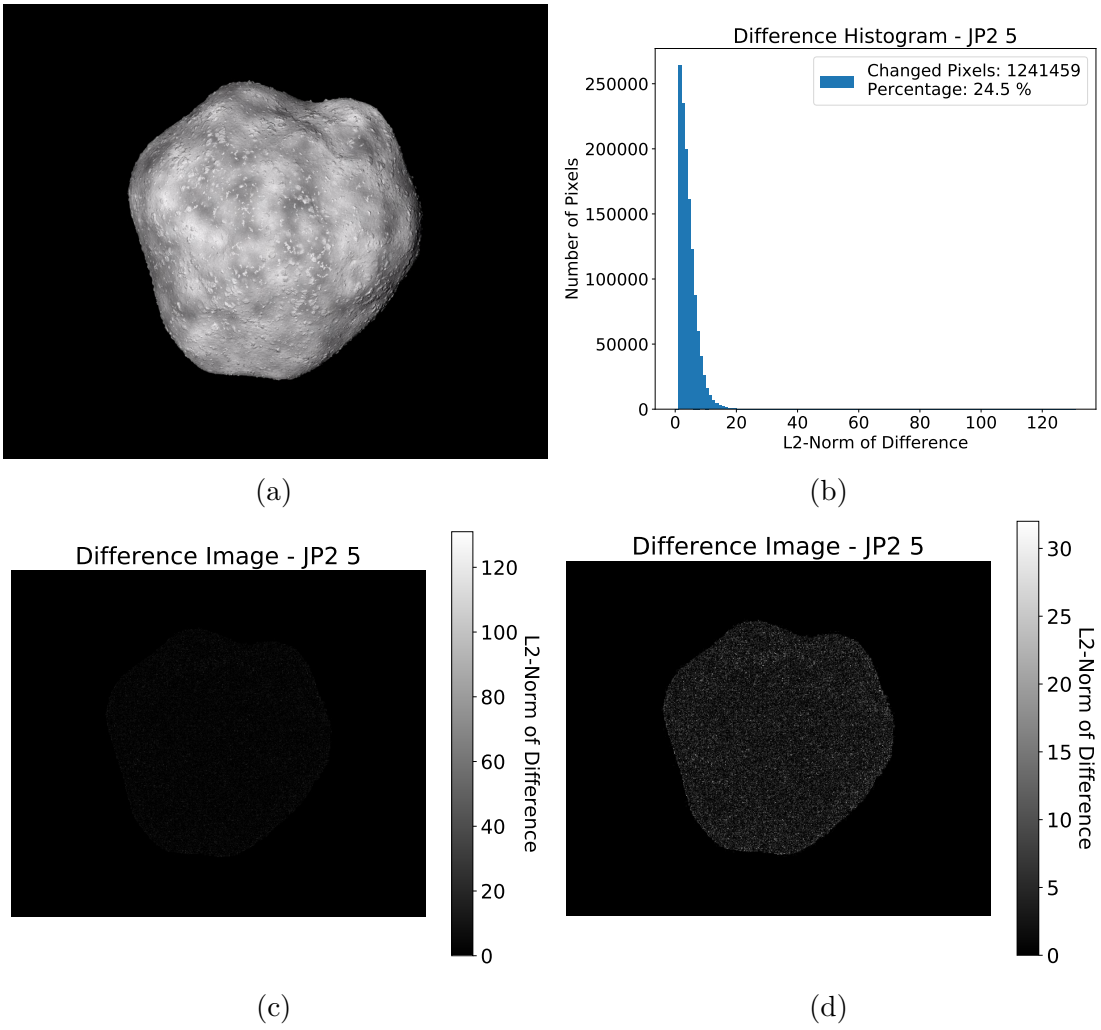


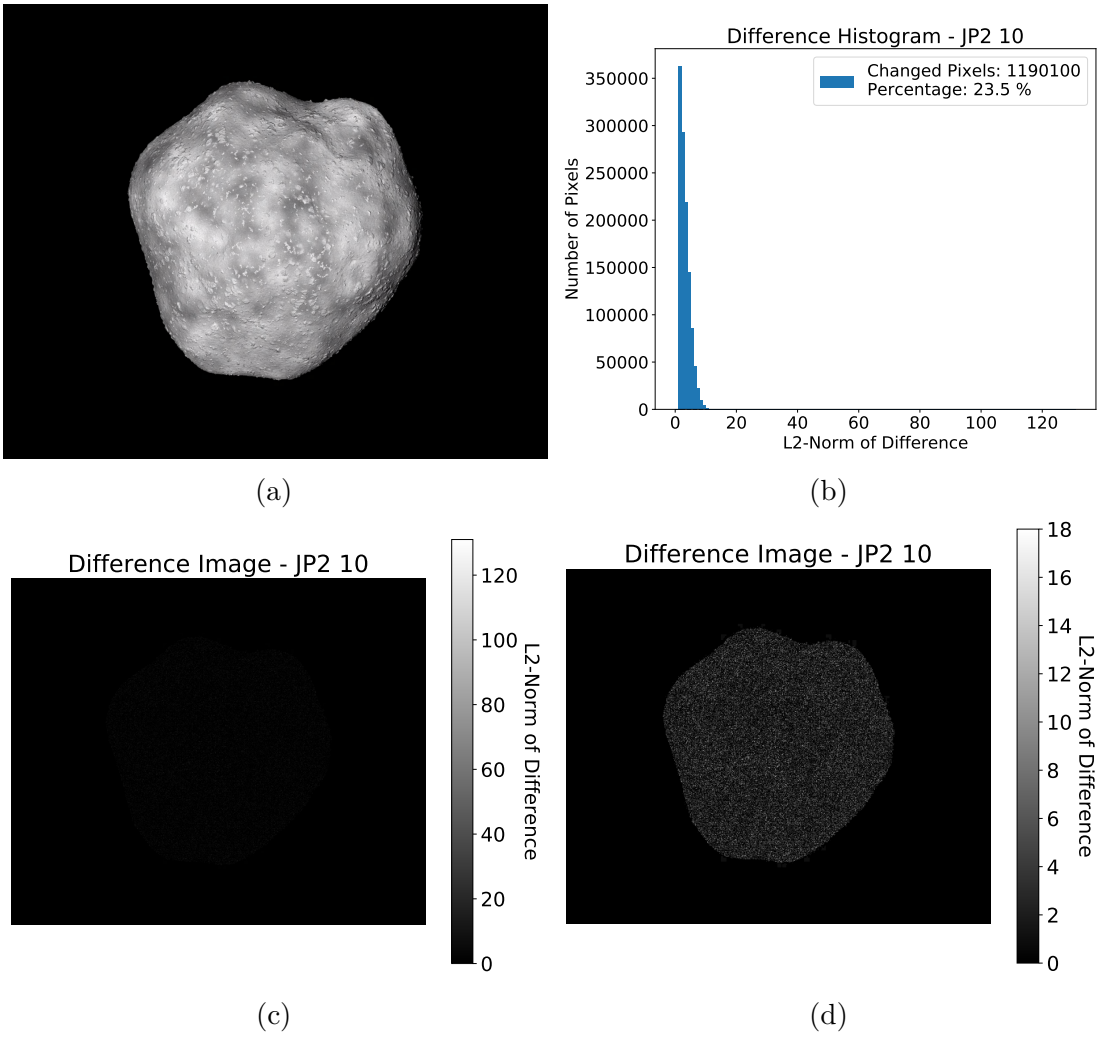
Figure 22: Overall rendered image after compression with JP2 quality level 1. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 131 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

Figure 23: Overall rendered image after compression with JP2 quality level 5. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 131 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

Figure 24: Overall rendered image after compression with JP2 quality level 10. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 131 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.

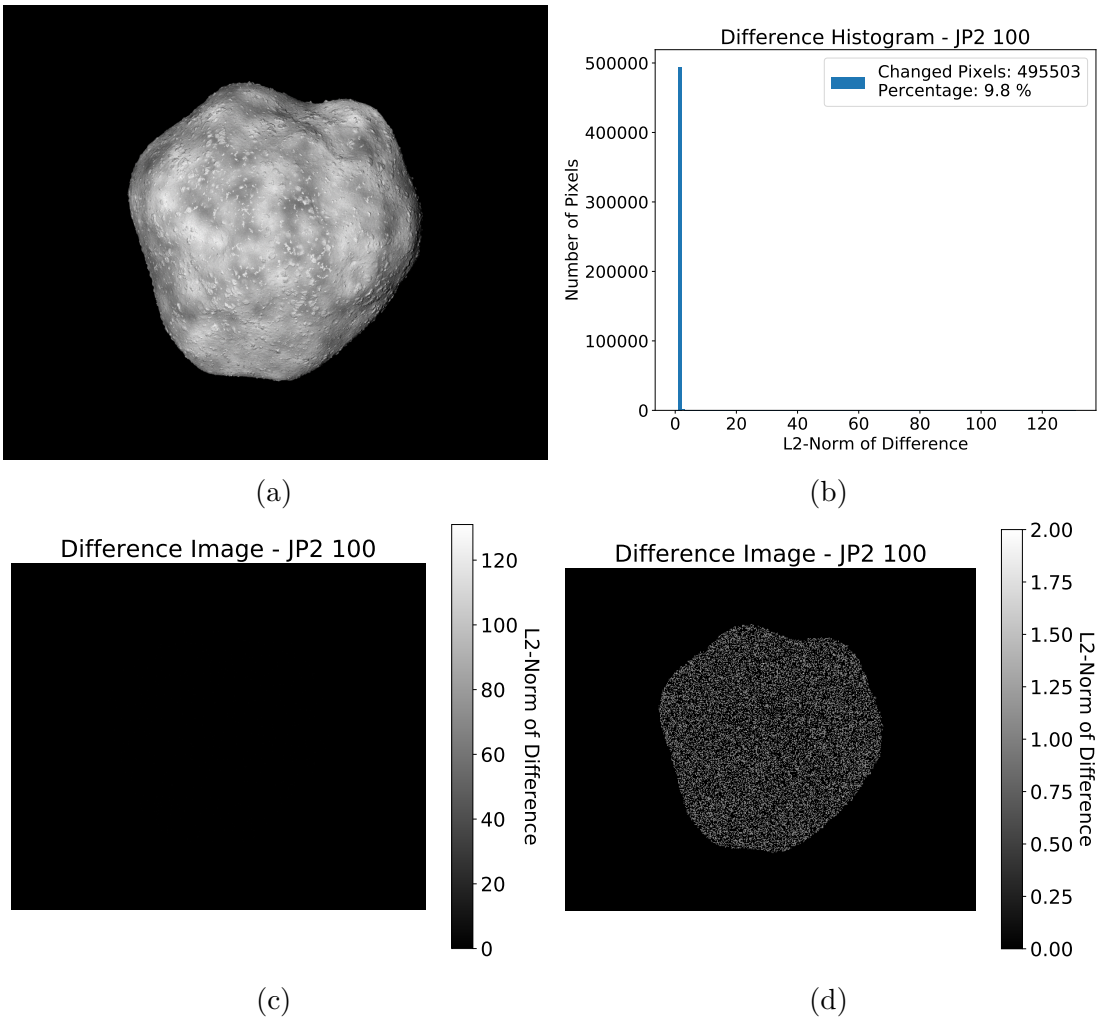
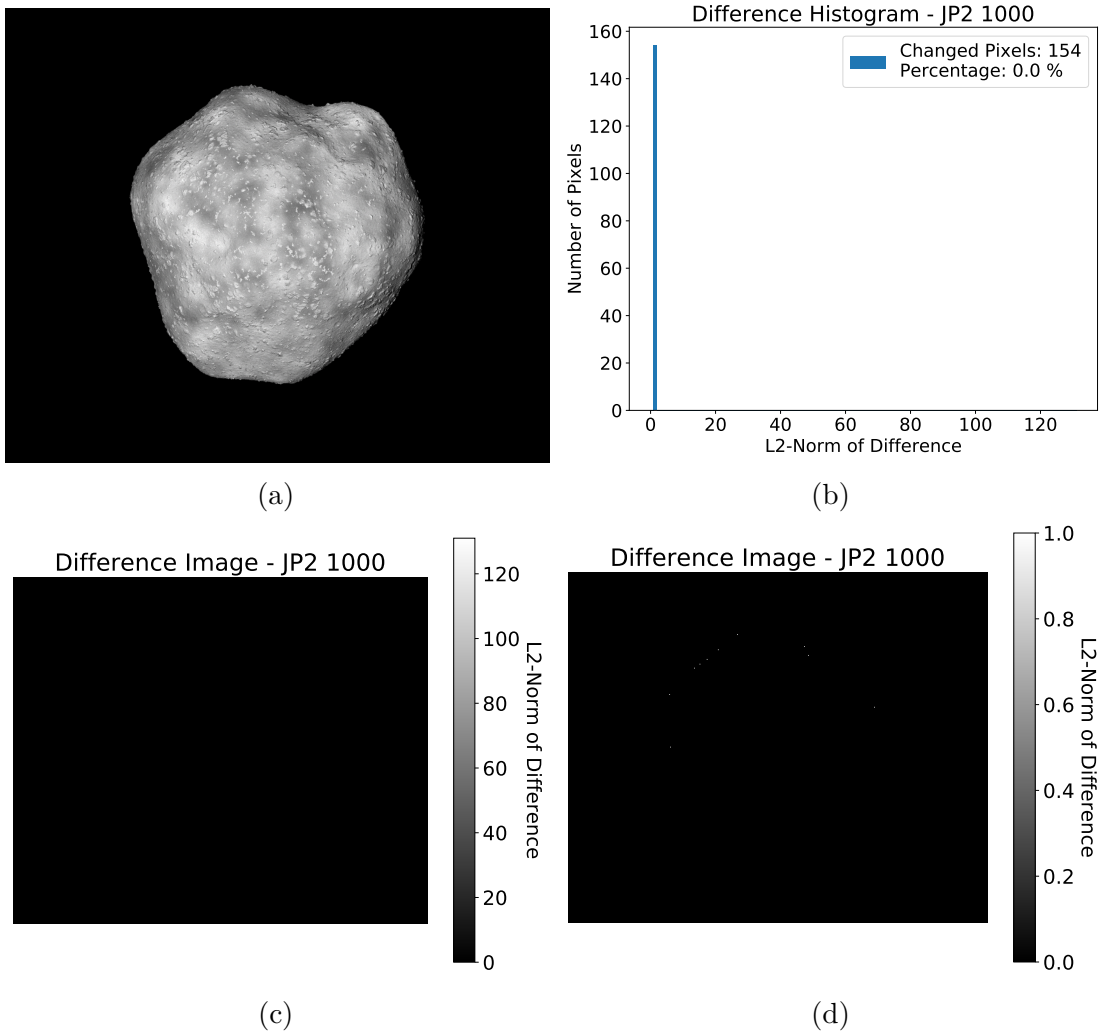


Figure 25: Overall rendered image after compression with JP2 quality level 100. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 131 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

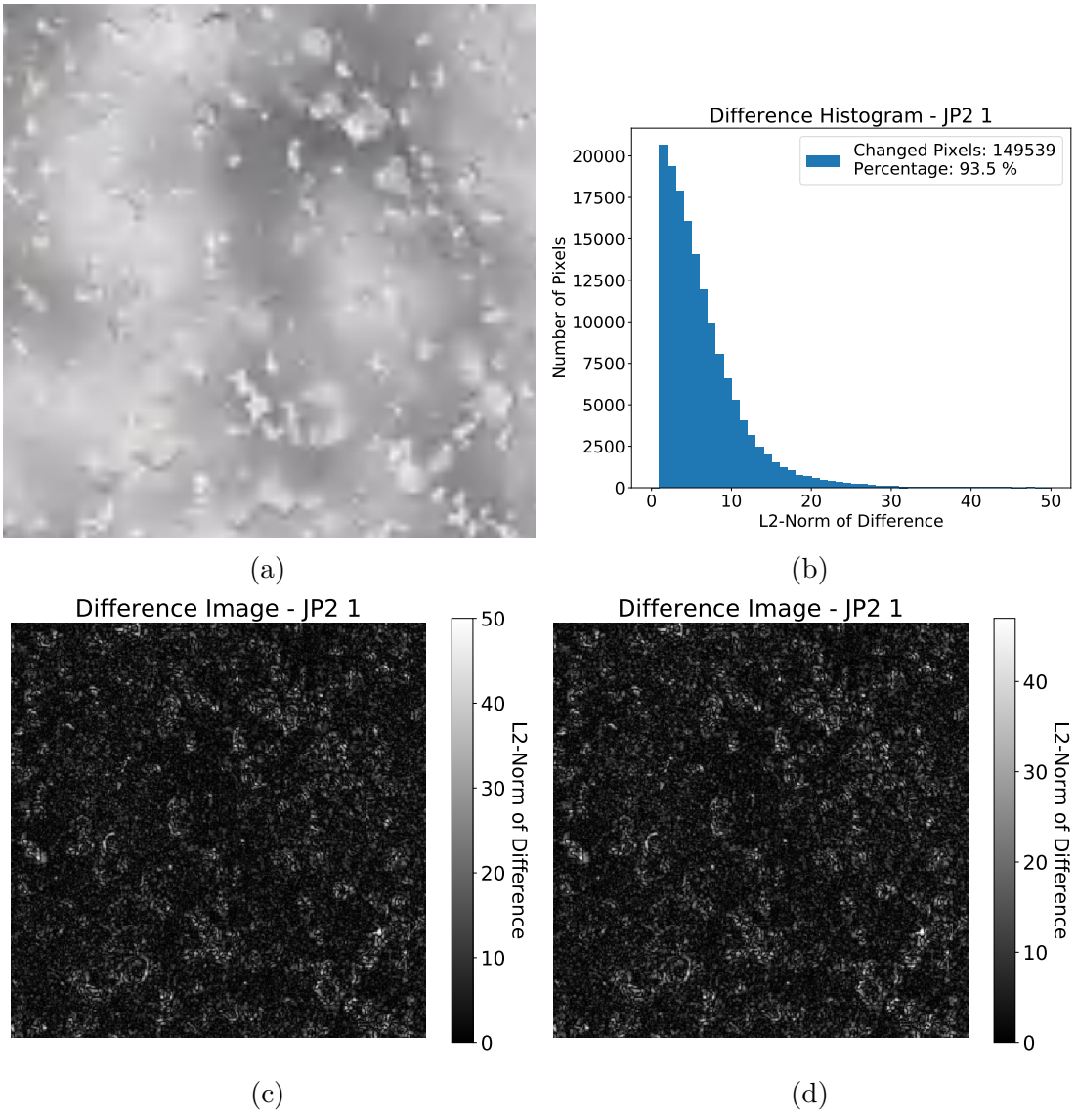
Figure 26: Overall rendered image after compression with JP2 quality level 1000. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 131 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.

Overall images show that lossy compression introduces artefacts for all quality levels. Visual inspection of the rendered images does not reveal many changes between different quality levels. However, histograms reveal that the number of altered pixels and the amount of alteration increases with decreasing quality level. The difference image for quality level 1000 in Figure 26d shows the minute changes from compression. The difference images in Figures 22d, 23d, 24d and 25d outline the shape of the SSSB hence compression artefacts are spread across the entire SSSB. Comparing the difference images in Figures 22c, 23c, 24c, 25c and 26c which use the same scale for all quality levels show that the alteration in quality levels 100 and 1000 are much lower compared to quality levels 1, 5 and 10.

Figures 27 to 31 show compressed close-up images, difference histograms and difference images after compression with JP2 with quality levels 1, 5, 10, 100 and 1000.

Comparing the close-up images of various quality levels in Figures 27a, 28a, 29a, 30a and 31a reveals a degradation of image quality with decreasing compression quality levels. Histograms confirm that the number of altered pixels and the amount of alteration increases with decreasing quality level. The difference image for quality level 1000 in Figure 31d and the respective histogram show that there were no altered pixels for quality level 1000, i.e. compression was lossless. The difference image in Figure 30d resembles random noise. In contrast, difference images in Figures 27d, 28d and 29d show non-random artefacts correlated with surface features when comparing to the unaltered image in Figure 31a. However, when comparing the difference images in Figures 27c, 28c, 29c, 30c and 31c which use the same scale for all quality levels, we see the alteration in quality levels 100 and 1000 are much lower compared to quality levels 1, 5 and 10.

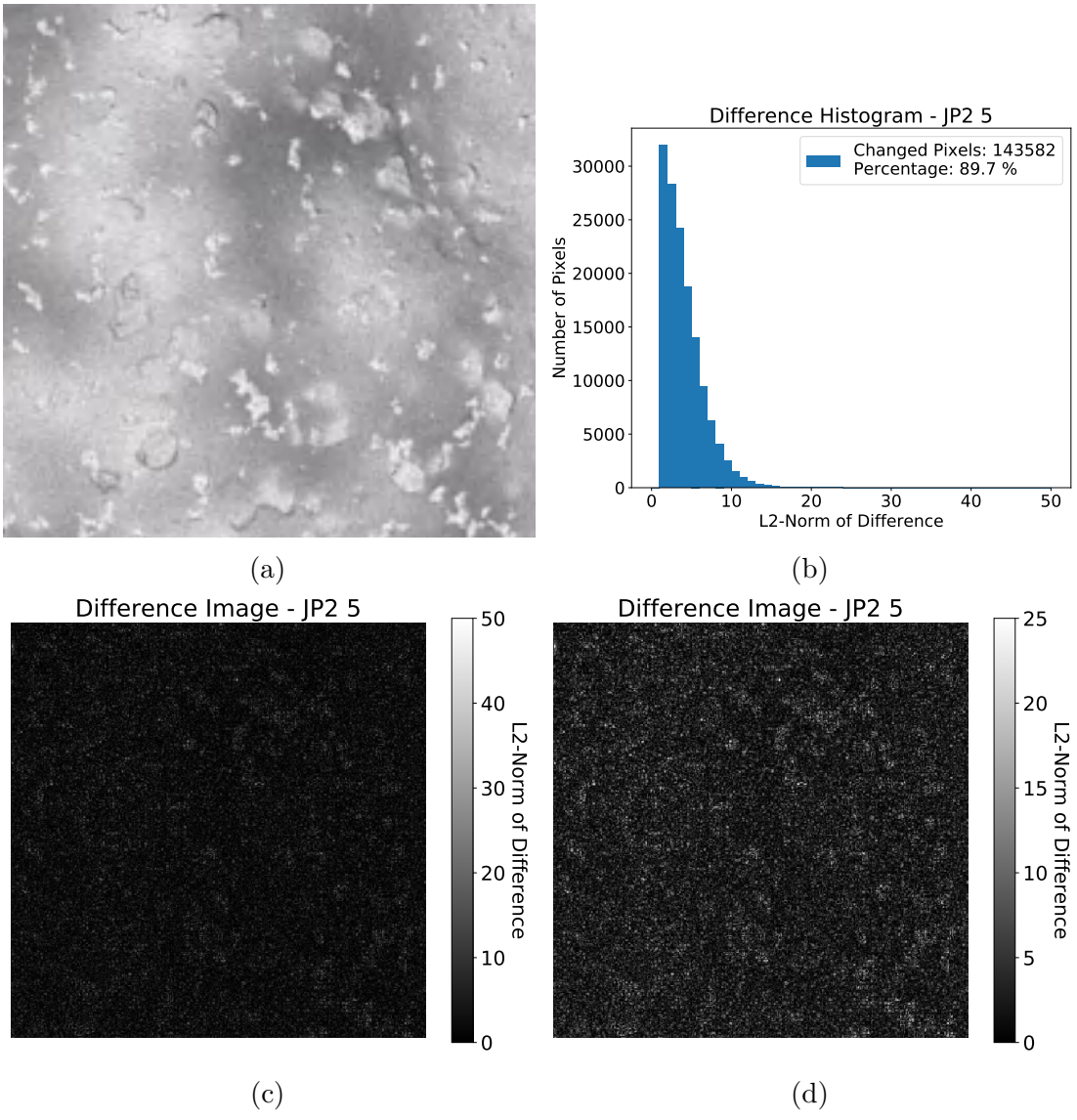
Comparing results of the overall images with results of the close-up images reveals that overall images are less altered by compression relative to their size. A substantial portion of overall images is covered by a black background thus compression does change less pixels than in close-up images. Overall difference images resemble random noise. In contrast, close-up images reveal a relation between surface features and compression artefacts for low quality levels.



(a) (b)

(c) (d)

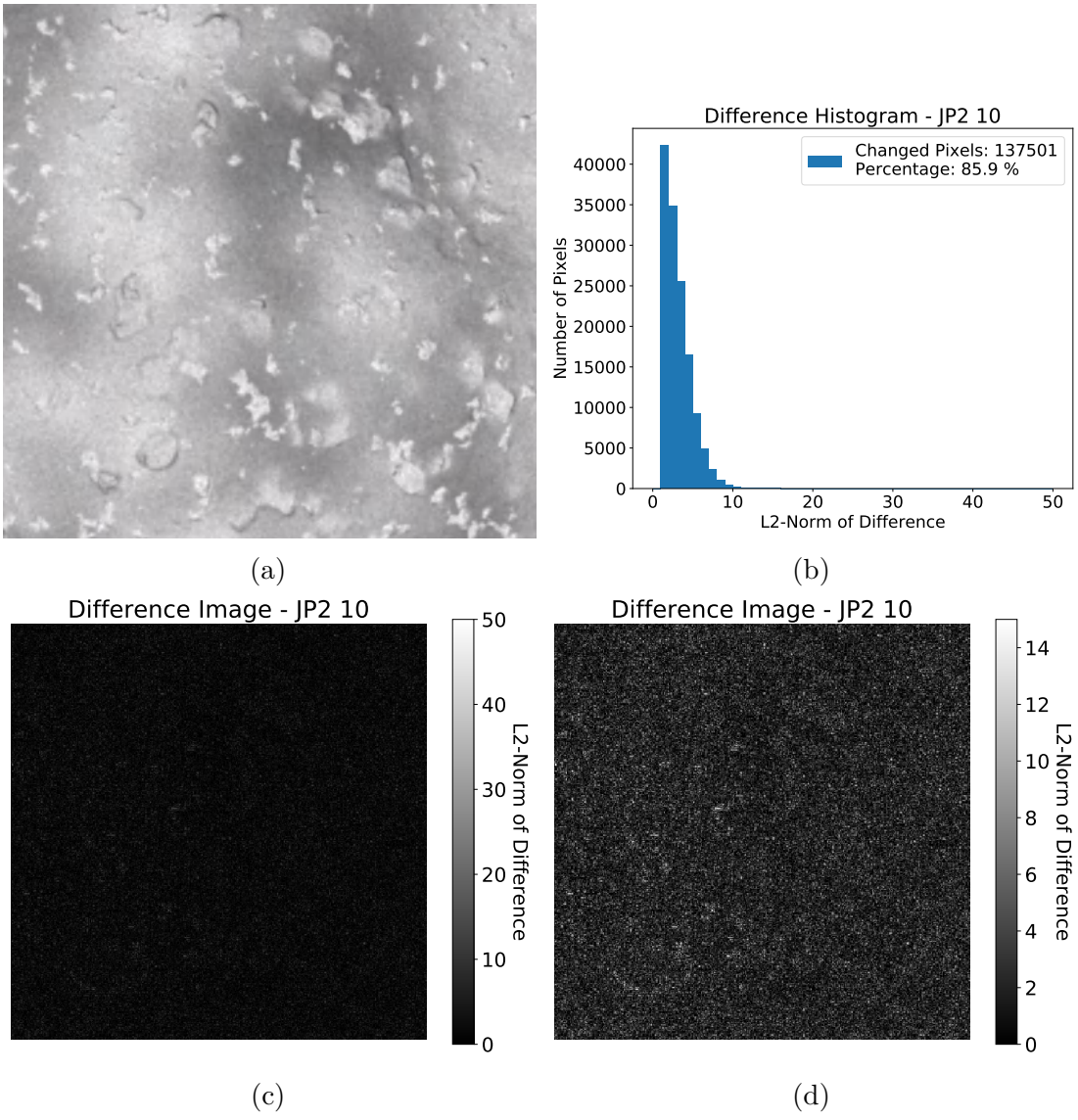
Figure 27: Close-up rendered image after compression with JP2 quality level 1. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 51 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

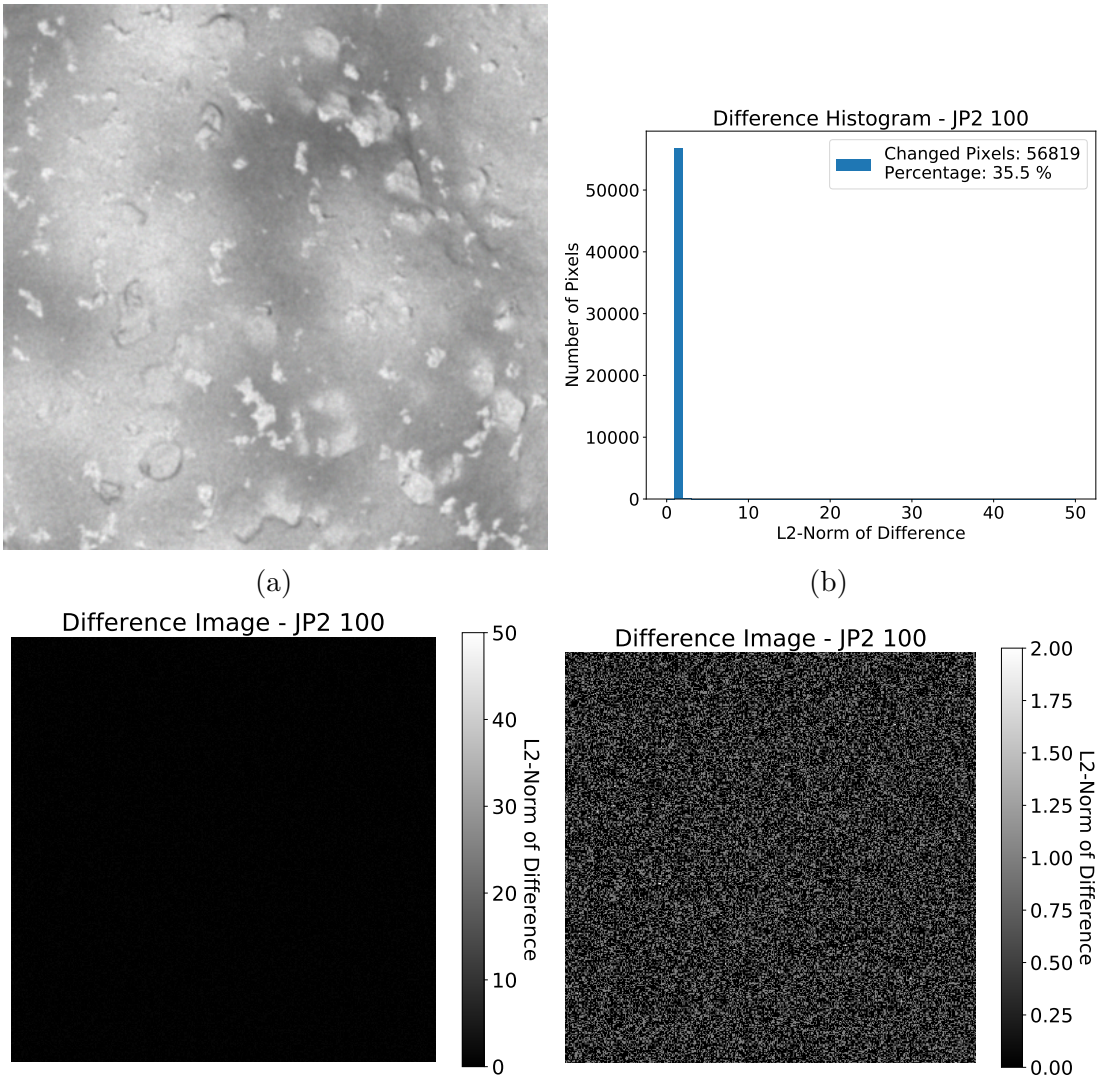
Figure 28: Close-up rendered image after compression with JP2 quality level 5. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 51 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

Figure 29: Close-up rendered image after compression with JP2 quality level 10. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 51 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.



(a) (b)

(c) (d)

Figure 30: Close-up rendered image after compression with JP2 quality level 100. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 51 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.

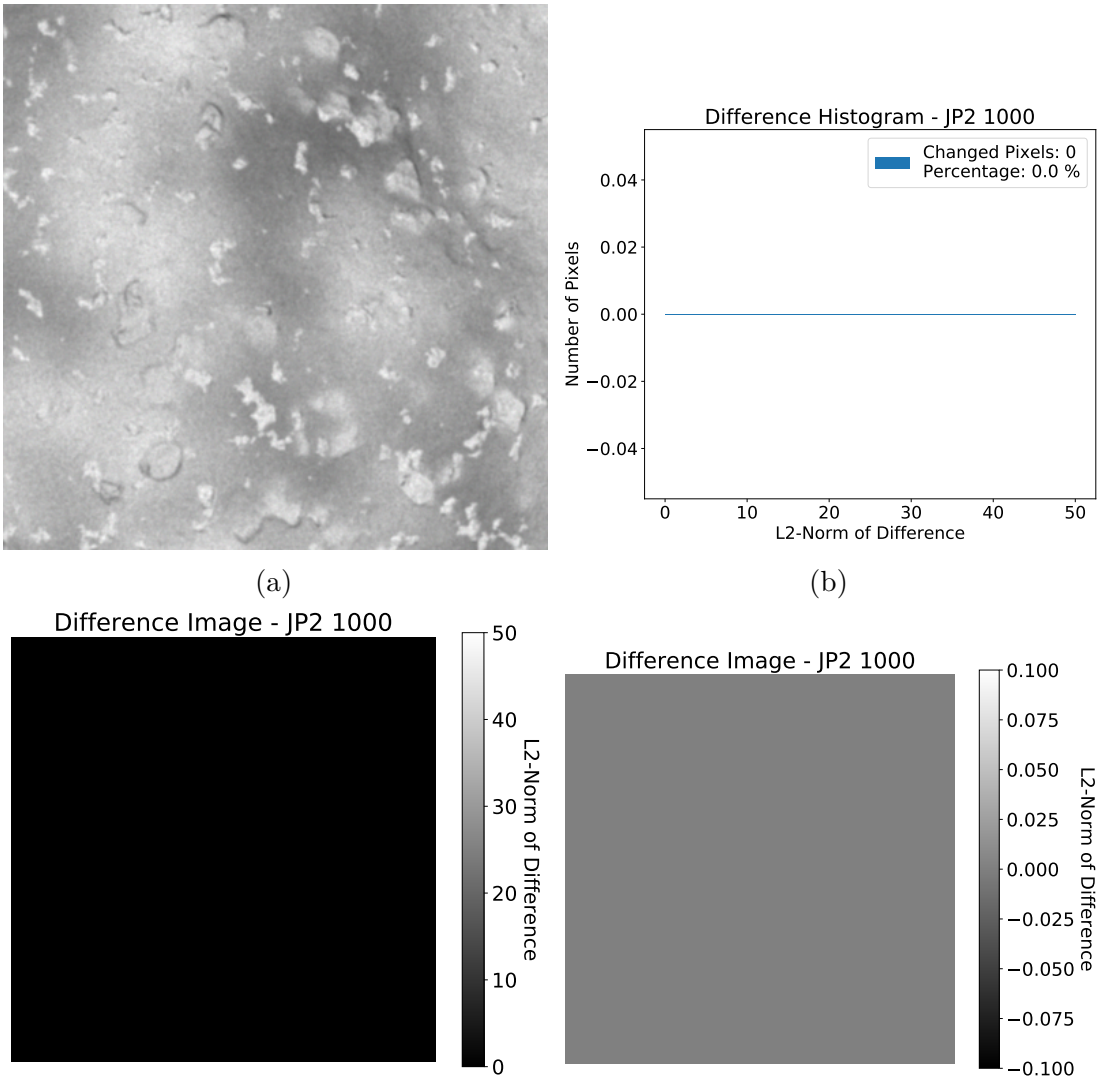


Figure 31: Close-up rendered image after compression with JP2 quality level 1000. The L2-norm is applied to the difference between the greyscale images of the lossless and respective lossy image. (a) Image after lossy compression. (b) Histogram of L2-norms of differences. (c) L2-norm difference image with a colour scale from 0 to 51 for comparison between various compression levels. (d) L2-norm difference image with a colour scale from 0 to the maximum L2-norm value for better visibility of compression effects.

4.3 Reconstruction

SISPO reconstruction used the parameters given in Table 16. refine_options is set to NONE to increase the probability of the SfM algorithms to converge. Camera calibration as described in Section 2.3 is not required since intrinsic camera parameters can be determined with sufficient accuracy before launch in space missions hence the parameters do not need to be optimised.

Table 16: Reconstruction Settings

Parameter Name	**Value**
export_type	obj
focal	66667
cam_model	1
geo_model	$10\,\mathrm{km\,s^{-1}}$
num_overlaps	4
use_prior	1
use_upright	0
force_compute	0
descriptor	SIFT
d_preset	ULTRA
method	FASTCASCADEHASHINGL2
refine_options	NONE
reduce_memory	1

4.3.1 Reconstructed Model Comparison

Reconstruction was successful in all cases presented in Table 12 except for a 400 km fly-by of a 1 km nucleus and the lowest compression quality level. The number of reconstructed points decreases with decreasing SSSB size and an increasing closest distance, i.e. with decreasing visible size in the images.

The number of points in the densified point cloud ranges from approximately 6×10^6 for a 50 km fly-by of a 10 km SSSB to approximately 2×10^3 for a 400 km fly-by of a 1 km SSSB. These two fly-by scenarios represent the two boundary cases of the simulation results, these are compared in more detail. The quality of other reconstructed 3D models is in between the presented results and are therefore not analysed further.

A comparison of the point clouds and resulting meshes of the two scenarios is shown in Figure 32. The large variation in points between Figure 32a and 32b after reconstruction and densification strongly influences 3D model quality. Both point clouds contain visible outliers which are removed in the 3D models.

Texturing only alters the appearance of a 3D model but not its quality thus meshes after refinement are compared. Comparing the 3D models in Figures 32c and 32d shows the influence of the number of points on the 3D models. Single vertices of the model in Figure 32d are visible. In contrast, the more detailed model in Figure 32c represents detailed surface features such as boulders.

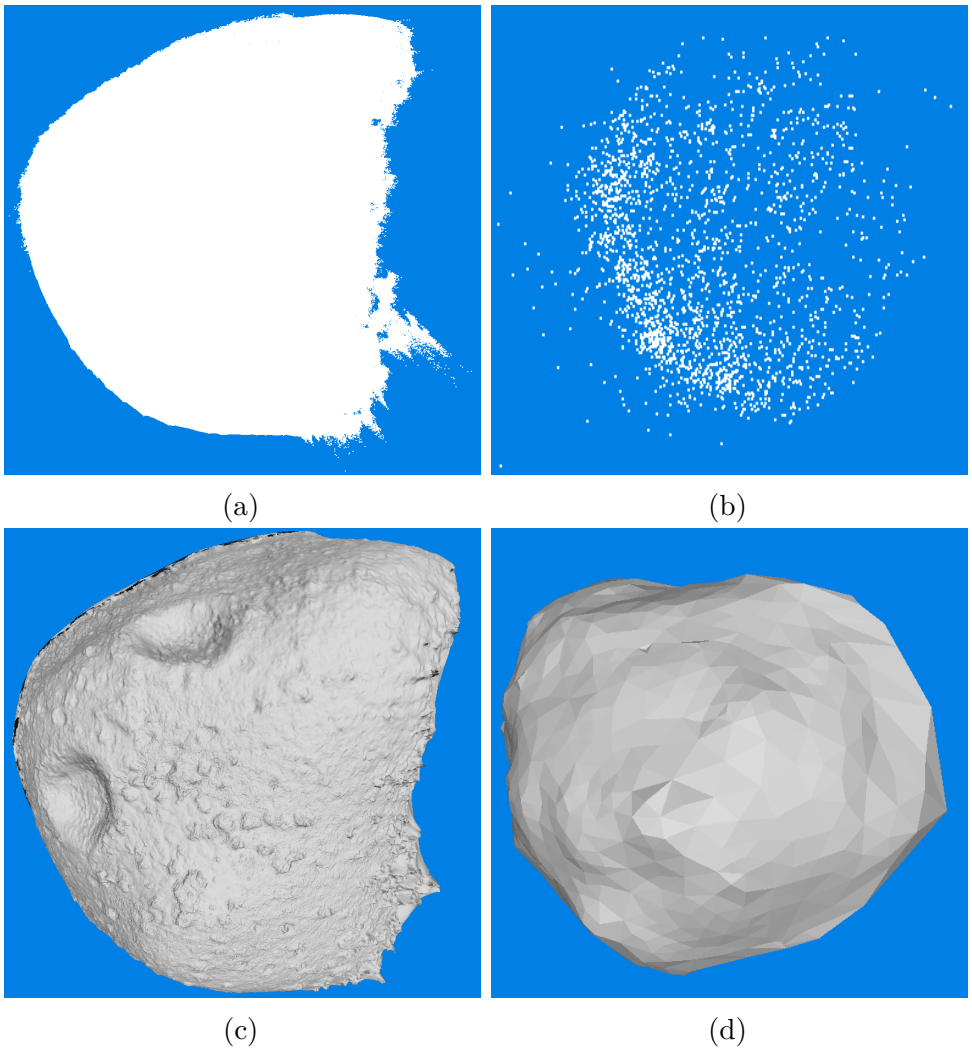


Figure 32: Images showing point clouds and resulting 3D models of two fly-by scenarios representing boundary cases with successful reconstructions. (a) Point cloud with $\approx 6 \times 10^6$ points representing a 10 km SSSB after a 50 km fly-by. (b) Point cloud with $\approx 2 \times 10^3$ points representing a 1 km SSSB after a 400 km fly-by. Point cloud densification failed for in this fly-by scenario, therefore the sparse point cloud is depicted. (c) Mesh based on the point cloud in (a) with $\approx 1 \times 10^6$ vertices. (d) Mesh based on the point cloud in (b) with ≈ 700 vertices. Mesh refinement failed in this scenario, therefore the sparse mesh is depicted.

4.3.2 Compression Effects on Reconstructed 3D Models

The quality of reconstructed 3D models is compared numerically. The number of points after densification, the number of vertices and the number of faces of the refined meshed model are compared for different levels of compression for the same simulation scenario. The number of points, vertices and faces relate to the level of detail of a 3D model. Moreover, the number of points in relation to the number of vertices can be used to analyse the amount of outliers in the point cloud since outlier

points cannot be included into a meaningful 3D model. Values are normalised to the results of PNG to compare lossless compression against lossy compression.

The theoretical size of 120 images is compared to the highest compression ratio of lossless compression using PNG. A series of 120 RGB images with 2454×2054 pixels and a colour depth of 8 bit has a data size of $d_s = 120 \times 2454 \times 2054 \times 3 \times 8\,\text{bit} = 1814.6\,\text{MB}$. The PNG data sets have sizes ranging from 5.2 MB to 564.6 MB, i.e. 0.3 % to 31.1 % of the raw data size. Varying apparent sizes of the nucleus and the resulting varying black portion of an image explain the varying and high possible compression ratios of PNG.

Figure 33 and Figure 34 compare effects of lossless compression using PNG and varying quality levels of lossy compression using JP2.

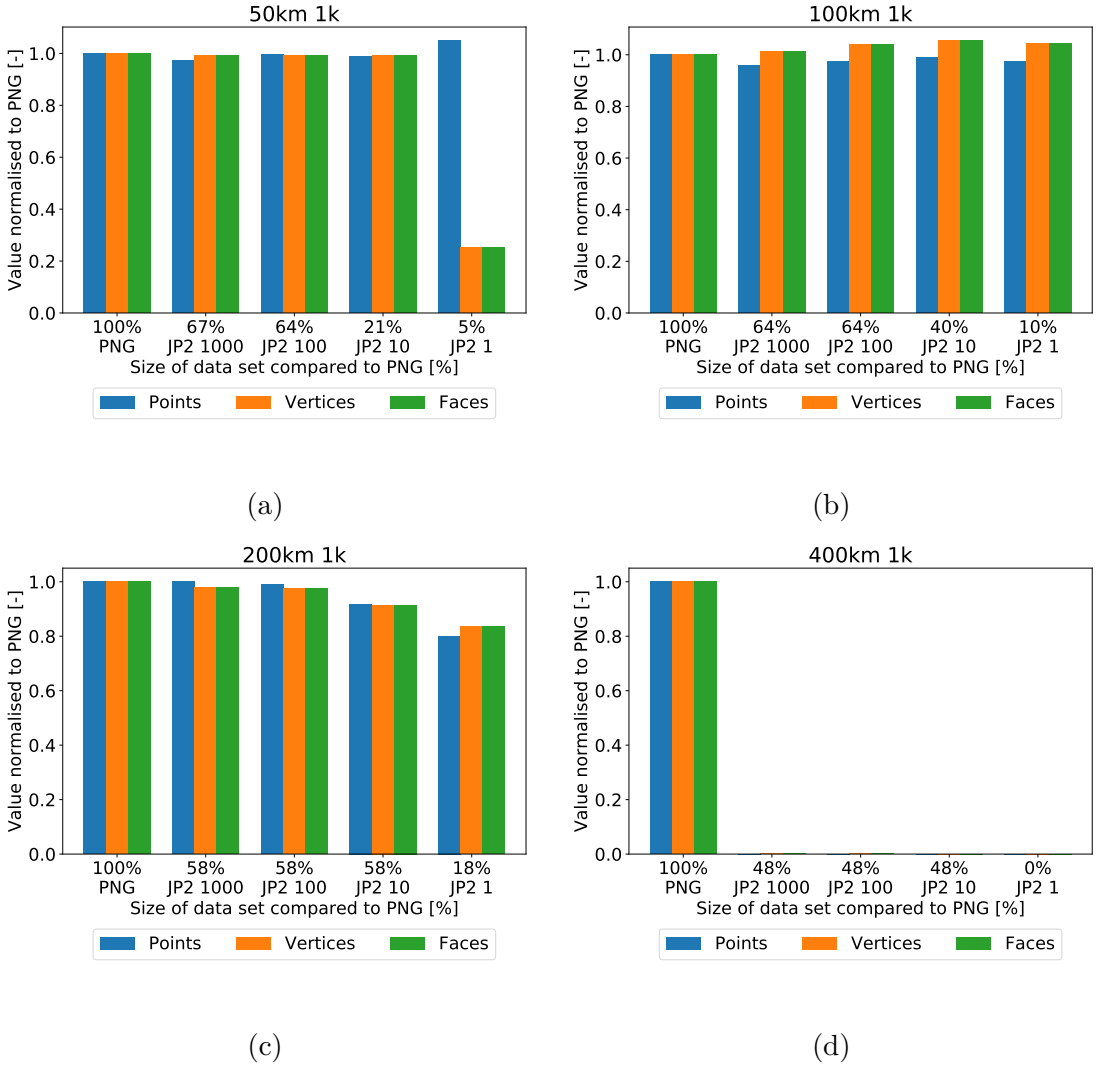


Figure 33: Comparison of reconstruction output after image compression with varying quality levels for a 1 km SSSB with varying fly-by distances. (a) Fly-by distance 50 km. (b) Fly-by distance 100 km. (c) Fly-by distance 200 km. (d) Fly-by distance 400 km.

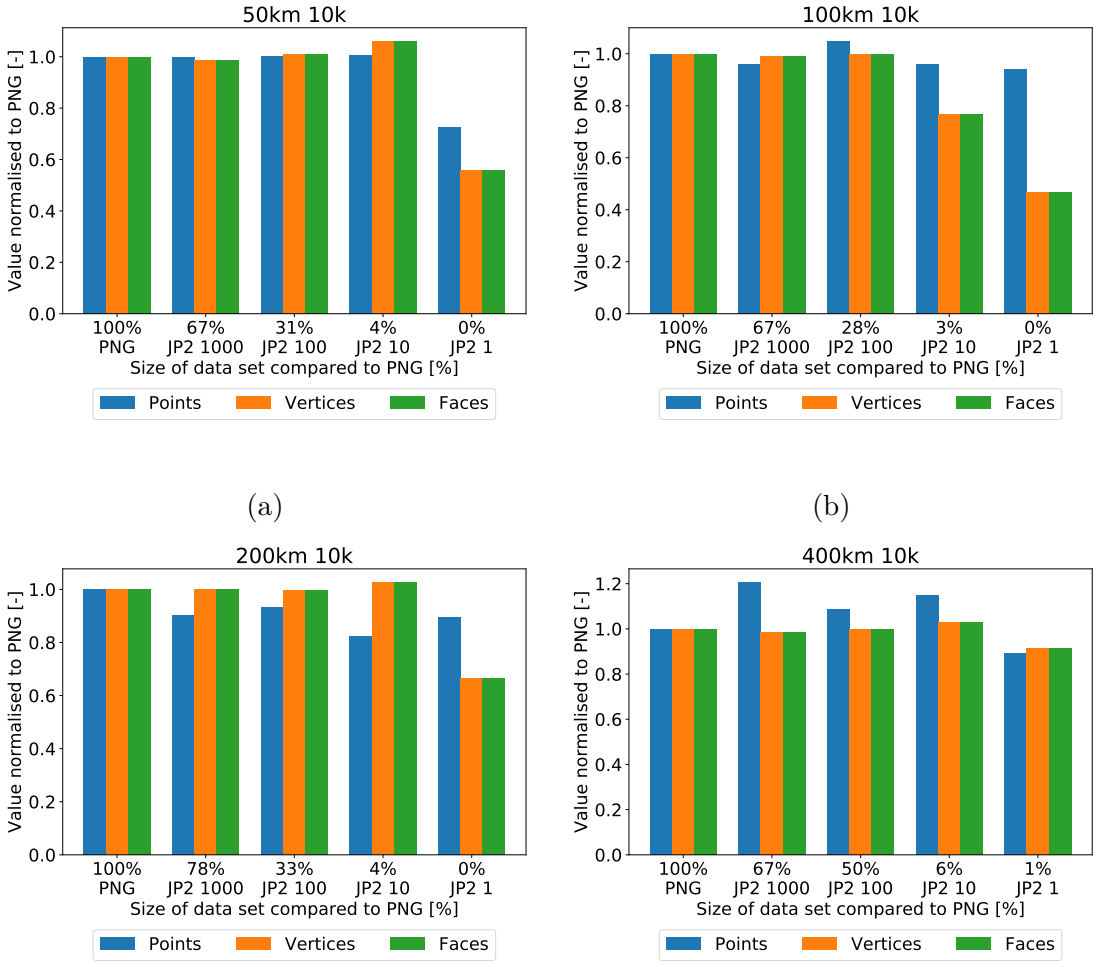


Figure 34: Comparison of reconstruction output after image compression with varying quality levels for a 10 km SSSB with varying fly-by distances. (a) Fly-by distance 50 km. (b) Fly-by distance 100 km. (c) Fly-by distance 200 km. (d) Fly-by distance 400 km.

Several observations can be made by analysing the graphs in Figure 33. The number of reconstructed vertices and faces increases with increasing degrees of compression for a 100 km fly-by. Artefacts introduced by compression create additional features for the SfM algorithms. The number of points of the lowest quality level of a 50 km fly-by contains more points than the PNG scenario. However, the number of vertices and faces is much lower hence the densified point cloud contains a higher number of outliers. If the number of vertices decreased more than the number of points, compression increased the number of outliers. The 3D model of a 400 km fly-by is strongly affected by compression. The level of detail of the reconstructed models deteriorated. The apparent size of the SSSB is small resulting in a small number of usable features which are removed by compression. For a 200 km fly-by,

the level of detail of the reconstructed 3D model decreases gradually with decreasing data size. Comparing data size in Figure 33 reveals that the amount of data can be reduced to approximately half of the size of PNG data sets without losing much detail except for a 400 km fly-by.

The JP2 quality level 10 data sets contain the highest number of faces after reconstruction for all cases except the 100 km fly-by. As described in Section 4.1.3, rendered images with a 10 km SSSB contain stripes. The stripes increase the density of points in the sparse point cloud as seen in Figure 35. Realistic images would not have stripe artefacts and therefore the number of points in the point cloud reconstructed from real images would be lower. The size of the four data sets in Figure 34 decreases with decreasing quality levels, i.e. most images contain the SSSB to a large extent. The data can be reduced without losing much detail in the 3D models to about a quarter of the PNG data size except for the 400 km fly-by.

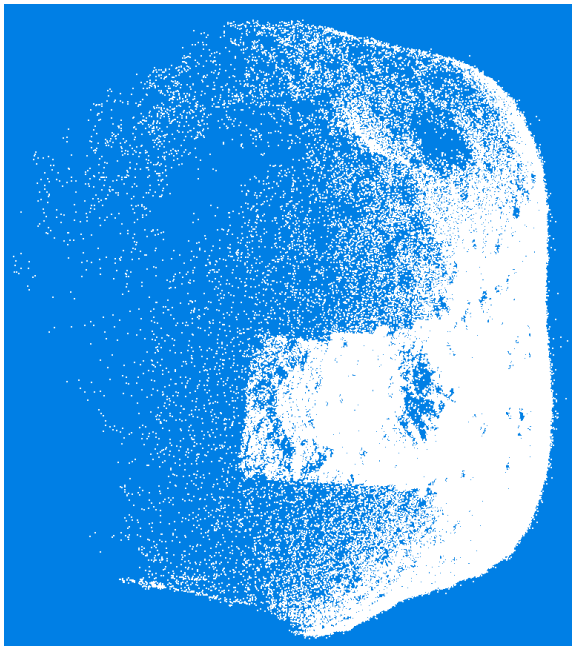


Figure 35: Sparse point cloud of a 10 km SSSB after a 50 km fly-by. The point cloud contains a high point density where the stripe artefact exists in rendered images.

Comparing the graphs in Figure 33 to the graphs in Figure 34 reveals that the data size reduces more gradual for a 10 km SSSB. The gradual decrease is explained by a larger black portion in 1 km SSSB images because PNG can compress the black area well. For the 10 km data set, PNG cannot compress data much because the images are covered by the SSSB to a big extent.

More samples exist with a stronger decreased number of vertices than reconstructed points in the 10 km SSSB scenarios compared to 1 km SSSB scenarios. Compression introduces more outliers for a 10 km SSSB than for a 1 km SSSB.

4.3.3 Reconstruction Algorithms

SISPO selects the best result of the three reconstruction algorithms based on the number of reconstructed points of each algorithm. Since SISPO uses three SfM algorithms, it is investigated which algorithm is more successful under which parameter set. Tables 17 and 18 show which algorithm reconstructed the most points in which scenario. The tables are colour-coded to improve readability.

Table 17: SfM algorithm with most reconstructed points for each scenario with a 1 km SSSB. Seq1 refers to algorithm IncrementalSfM and Seq2 refers to algorithm IncrementalSfM2.

Compression/ Distance [km]	PNG	JP2 1000	JP2 100	JP2 10	JP2 1
50	Seq1	Seq1	Seq1	Seq1	Seq1
100	Seq2	Seq1	Seq1	Seq1	Seq2
200	Seq2	Seq2	Seq2	Seq2	Seq2
400	Seq2	Seq2	Seq2	Seq2	—

Table 18: SfM algorithm with most reconstructed points for each scenario with a 10 km SSSB. Seq1 refers to algorithm IncrementalSfM, Seq2 refers to algorithm IncrementalSfM2 and Glob refers to algorithm GlobalSfM.

Compression/ Distance [km]	PNG	JP2 1000	JP2 100	JP2 10	JP2 1
50	Seq2	Seq2	Seq2	Seq2	Seq2
100	Seq2	Seq2	Seq2	Seq2	Seq2
200	Seq2	Seq2	Seq2	Seq2	Seq2
400	Seq1	Glob	Seq2	Seq1	Seq1

The number of points in the reconstructed point clouds of incremental SfM algorithms exceeded the point count of the global SfM algorithm in all scenarios except a 400 km fly-by compressed with JP2 quality level 1000.

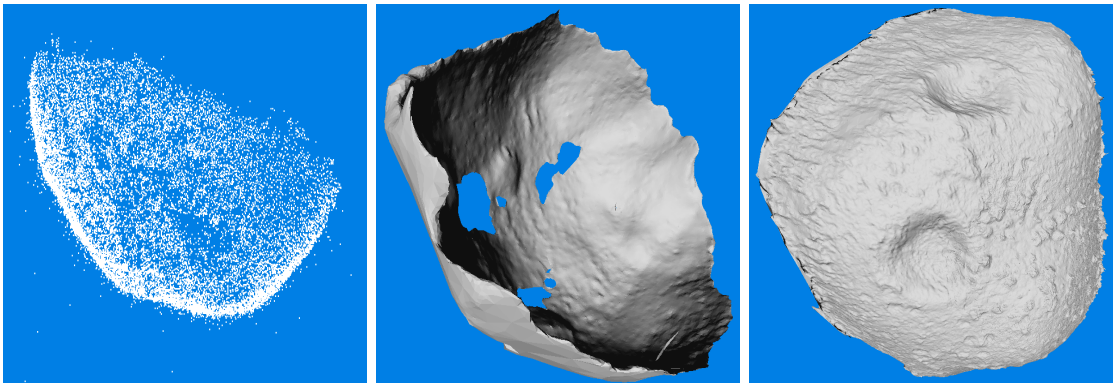
The results in Table 17 suggest that IncrementalSfM produces better results for small fly-by distance and IncrementalSfM2 produces better results when being further away from the SSSB. Table 18 contradicts this conclusion since IncrementalSfM2 was the best algorithm in nearly all cases. Other algorithms than IncrementalSfM2 were only used for the largest distance. Consequently, the success of the algorithms does not only depend on the amount of details in the images.

A possible explanation for the results is, that IncrementalSfM is most successful in well-defined scenarios, i.e. scenarios where many surface details are visible, but the nucleus never covers the entire image. As discussed in Section 4.3.4, if the nucleus

covers the entire FoV it causes problems to the reconstruction process, making the scenario less well-defined. For less well-defined scenarios, IncrementalSfM2 reconstructs the most points. The scenario where GlobalSfM reconstructed the most points is considered an outlier.

4.3.4 Reconstruction Problems

One problem of the reconstruction pipeline were inverted models. Some 3D models also did not have a closed surface. A third issue is flattening of models. Erroneous models containing all three problems are presented in Figure 36. The problem with



(a) (b) (c)

Figure 36: Erroneous reconstructed point cloud and models. (a) Sparse point cloud containing holes and being inverted, i.e. resembling the inside of an egg shell. (b) Refined mesh of created from the point cloud in (a). (c) Mesh which does not resemble a hemisphere but rather a flat wall which is slightly bent on the left side of the image.

holes and inverted models was overcome by using priors and fixed intrinsic camera parameters. Flattening of a model occurs when the minimum distance to the SSSB is small. In the image series used for reconstructing Figure 36c, the SSSB does not entirely fit into the FoV in 65 out of 120 images. The FoV of 39 out of 120 images is completely covered by the nucleus, i.e. the SSSB extends beyond the corners of the FoV. The model becomes flattened since such a high percentage of images does not provide information about the overall shape of the object to the reconstruction pipeline. Flattening can be overcome by adding views that provide more shape information about an object.

5 Conclusion

In this work, the original code base which included rendering and reconstruction capabilities was developed further into the first version of the SISPO software package. Furthermore, SISPO was extended to include an image compression and decompression package.

SISPO was used to simulate several fly-by scenarios with SSSB sizes of 1 km and 10 km and fly-by distances of 50 km, 100 km, 200 km and 400 km.

Comparison with real images from asteroid Bennu and comets 67P and 81P show that the current shader implementation creates imagery mostly resembling an asteroid rather than a comet. Differences between rendered images and real comet images include missing gas and dust jets as well as ridge-like surfaces features.

Rendered images are compressed with either PNG or JP2 representing lossless and lossy compression techniques respectively. Lossy compression reduces the amount of data more than lossless compression at the cost of introducing artefacts. Lossy compression with low compression ratios introduces artefacts which resemble random noise. Artefacts introduced by high compression ratios show correlation to surface features. Visual inspection confirms that low compression ratios do not degrade specific parts of an image in contrast to high compression ratios which blur contours of surface features.

Compressed images are used to reconstruct a 3D model leveraging SfM algorithms provided by OpenMVG and OpenMVS. Two incremental and one global SfM algorithm were used during simulations. In total, the IncrementalSfM2 algorithm reconstructed the most points in most scenarios. Well-defined scenes were better reconstructed with the IncrementalSfM algorithm while less well-defined scenes were better reconstructed with IncrementalSfM2. The incremental SfM approach is more successful in SISPO since the GlobalSfM algorithm reconstructs more points than IncrementalSfM and IncrementalSfM2 in only one scenario. A 3D model could successfully be reconstructed in all tested cases except for a 400 km fly-by of a 1 km SSSB with highest lossy compression ratio. The number of reconstructed points ranged from $\approx 2 \times 10^3$ points for a 400 km fly-by of a 1 km SSSB to $\approx 6 \times 10^6$ points for a 50 km fly-by of a 10 km SSSB. It was found that lossy compression introduces artefacts that can increase the number of reconstructed points from the SfM algorithms if the artefacts are distributed randomly. High lossy compression ratios alter surface features which might lead to an increased point count in the point cloud. However, such points are often outliers since the points are removed when creating a 3D mesh from the point cloud.

For fly-by scenarios with 1 km SSSBs, lossy compression allows data reduction to about half of the data size of lossless compression.

Problems were found during rendering and reconstruction. One rendering problem are stripes, dark patches across the entire SSSB, which were found on many images of a 10 km SSSB. Rendering fireflies are another problem that was encountered. Fireflies were only found in one image of a 50 km fly-by of a 10 km nucleus. Both problems could not be resolved in this work.

Reconstruction problems included inverted 3D models and surfaces not being

closed. Both problems were overcome by removing optimisation of intrinsic camera parameters and using priors. Removing the optimisation is justified with the assumption of known calibration values for imagers in space missions. Motion priors, i.e. initial guesses for extrinsic camera parameters, are used since it is assumed that a rough trajectory of spacecraft is known from other measurements.

SISPO can aid in designing missions to SSSBs. The *HERA* and *CI* missions target a SSSB with a mother spacecraft carrying a number of small satellites. Therefore, investigating the effect of compression is a first step in maximising the useful scientific image data that can be transmitted from these small spacecraft. Mission concepts like *MANTIS*, *CASTAway* or *MAT* could use SISPO to improve their design and maximising their science return. Additionally, SISPO can be used for developing optical navigation algorithms. Either indirectly by creating a large number of images or directly by utilising the camera pose estimates provided by the SfM pipeline.

While providing a foundation for SISPO, several issues could not be addressed in the course of this work. First, a realistic model of spacecraft attitude motion and control is missing. The camera of the simulation environment is perfectly oriented to the centre of the SSSB in SISPO. Realistic rotation should cover at least two effects, motion blur due to instantaneous rotation velocities of spacecraft and off-centre pointing due to control inaccuracies. Furthermore, it is necessary to include imaging distortions such as astigmatism, bokeh, coma, field curvature, glare.

Moreover, it is necessary to include a gas and dust environment around the SSSB to extend rendering capabilities from asteroids to comets. Including multiple SSSBs into the simulation environment would allow more complex simulation scenarios including for example a binary system. Furthermore, a more recent and accurate star catalogue, like the GAIA catalogue, could be implemented to improve star map rendering.

Currently, SISPO assumes that an instrument always uses RGB channels. Since many imagers used in deep space use monochrome CCDs, an option to select either RGB or monochrome rendering should be implemented. Furthermore, the realism of rendered images can be increased by using an improved photometric system, such as the Ultraviolet, Blue, Visual, Red, Infrared system (UBVRI system) [7], that takes into account the sensitivity of CCDs in the red and infrared spectrum.

A simulation of data transmission should be included. For example, a realistic simulation for packet loss using common radio transmission methods and protocols. Additionally, compression techniques which are commonly used in deep space missions should be added to increase the realism in SISPO. Effects of non-optimal lighting conditions, i.e. overexposure and underexposure, on compression should be investigated. A substantial part of images far from a SSSB is black which could be cropped away to reduce the amount of data. Consequently, image cropping should be added to increase the realism of image compression and data transmission. The ultimate goal, e.g. for the *CI* mission, would be to develop a prioritisation algorithm for the images which can prioritise data transmission on packet level.

Furthermore, the shader that implements the procedural terrain generation should be developed further. The shader should represent a comet's surface better, e.g. by including ridge-like surfaces features. Since most of the execution time of SISPO is

used for rendering, the shader should be developed to be less computationally heavy. Additionally, the *SssbConstDist* is currently rendered in each time step. However, the *SssbConstDist* image is only required for calibration for long distances from the nucleus. Rendering performance can be improved by checking the apparent size of the SSSB in the *SssbOnly* image directly after rendering to decide whether it is necessary to render the *SssbConstDist* image. The interface for it should be included into SISPO and restricted to values that create reasonable images.

An attempt to include a Hapke model via the synthspace package[5] was unsuccessful in this work. It would be interesting to compare the results of SISPO and the Hapke model to show strengths and weaknesses of both models. Moreover, a Hapke model is substantially faster while being less accurate. One possible use case would be creating real-time imagery which is currently not possible with SISPO.

[5] https://github.com/oknuutti/synthspace

References

[1] Michael F A'Hearn. "Comets: looking ahead". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375.2097 (2017), p. 20160261.

[2] University of Arizona. *OSIRIS-REx Frequently Asked Questions*. [accessed: 2020-02-23]. URL: https://cdn.uanews.arizona.edu/s3fs-public/download-media/OSIRIS-REx%5C%20FAQ.pdf.

[3] OpenMVG authors. *cameras*. [accessed: 2020-01-30]. URL: https://openmvg.readthedocs.io/en/latest/openMVG/cameras/cameras/.

[4] OpenMVG authors. *sfm*. [accessed: 2020-01-30]. URL: https://openmvg.readthedocs.io/en/latest/openMVG/sfm/sfm/.

[5] Paul Bergmann, Rui Wang, and Daniel Cremers. "Online Photometric Calibration of Auto Exposure Video for Realtime Visual Odometry and SLAM". In: *IEEE Robotics and Automation Letters* (2018). ISSN: 23773766. DOI: 10.1109/LRA.2017.2777002.

[6] M. S. Bessell. "UBVRI photometry. II - The Cousins VRI system, its temperature and absolute flux calibration, and relevance for two-dimensional photometry". In: *Publications of the Astronomical Society of the Pacific* 91 (Oct. 1979), p. 589. ISSN: 0004-6280. DOI: 10.1086/130542.

[7] Michael S. Bessell. "Photometric Systems". In: *International Astronomical Union Colloquium* 136 (1993), pp. 22–39. ISSN: 0252-9211. DOI: 10.1017/s025292110000734x.

[8] Ross A. Beyer, Oleg Alexandrov, and Scott McMichael. "The Ames Stereo Pipeline: NASA's Open Source Software for Deriving and Processing Terrain Data". In: *Earth and Space Science* 5.9 (Sept. 2018), pp. 537–548. ISSN: 23335084. DOI: 10.1029/2018EA000409. URL: http://doi.wiley.com/10.1029/2018EA000409.

[9] Phuong Ngoc Binh Do and Quoc Chi Nguyen. "A Review of Stereo - Photogrammetry Method for 3-D Reconstruction in Computer Vision". In: *Proceedings - 2019 19th International Symposium on Communications and Information Technologies, ISCIT 2019*. IEEE. 2019, pp. 138–143. ISBN: 9781728150093. DOI: 10.1109/ISCIT.2019.8905144.

[10] Irina Bocharova. *Compression for multimedia*. Vol. 9780521114. 2009, pp. 1–269. ISBN: 9780511804069. DOI: 10.1017/CBO9780511804069.

[11] N.E. E. Bowles et al. "CASTAway: An asteroid main belt tour and survey". In: *Advances in Space Research* 62.8 (Oct. 2018), pp. 1998–2025. ISSN: 02731177. DOI: 10.1016/j.asr.2017.10.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0273117717307597.

[12] Roland Brochard et al. "Scientific image rendering for space scenes with the SurRender software". In: (Oct. 2018). URL: http://arxiv.org/abs/1810.01423.

[13] D. E. Brownlee et al. "Stardust: Comet and interstellar dust sample return mission". In: *Journal of Geophysical Research E: Planets* 108.10 (Oct. 2003). ISSN: 01480227. DOI: 10.1017/s0252921100501584.

[14] Alan Chamberlin. *Comet and Asteroid Orbital Element Distribution*. [accessed: 2020-02-23]. URL: https://ssd.jpl.nasa.gov/?dist%5C_ae%5C_sb.

[15] Steven R. Chesley et al. "Orbit and bulk density of the OSIRIS-REx target Asteroid (101955) Bennu". In: *Icarus* (2014). ISSN: 10902643. DOI: 10.1016/j.icarus.2014.02.020.

[16] Ondřej Chum and Jiří Matas. "Matching with PROSAC - Progressive sample consensus". In: *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. 2005, pp. 220–226. ISBN: 0769523722. DOI: 10.1109/CVPR.2005.221.

[17] *Comets*. [accessed: 2020-02-22]. URL: https://ssd.jpl.nasa.gov/?comets.

[18] Wikimedia Commons. *2-level wavelet transform-lichtenstein*. [accessed: 2020-02-10]. 2019. URL: https://commons.wikimedia.org/w/index.php?title=File:Jpeg2000_2-level_wavelet_transform-lichtenstein.png&oldid=357227594.

[19] Wikimedia Commons. *Orbit*. [accessed: 2020-02-10]. 2019. URL: https://commons.wikimedia.org/w/index.php?title=File:Orbit1.svg&oldid=349069009.

[20] Gaudenz Danuser. "Computer Vision in Cell Biology". In: *Cell* 147.5 (Nov. 2011), pp. 973–978. ISSN: 00928674. DOI: 10.1016/j.cell.2011.11.001. URL: http://opencv.willowgarage.com%20http://www.ncbi.nlm.nih.gov/pubmed/22118455%20https://linkinghub.elsevier.com/retrieve/pii/S0092867411012906.

[21] Francesca E. DeMeo et al. "An extension of the Bus asteroid taxonomy into the near-infrared". In: *Icarus* 202.1 (July 2009), pp. 160–180. ISSN: 00191035. DOI: 10.1016/j.icarus.2009.02.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0019103509000554.

[22] ESA. *Hera*. [accessed: 2020-02-01]. URL: https://www.esa.int/Safety_Security/Hera.

[23] ESA. *Rosetta's frequently asked questions*. [accessed: 2020-02-23]. URL: http://www.esa.int/Science_Exploration/Space_Science/Rosetta/Frequently_asked_questions.

[24] Olivier Faugeras and Renaud Keriven. "Variational principles, surface evolution, PDEs, level set methods, and the stereo problem". In: *IEEE Transactions on Image Processing* 7.3 (Mar. 1998), pp. 336–344. ISSN: 10577149. DOI: 10.1109/83.661183. URL: http://ieeexplore.ieee.org/document/661183/.

[25] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* (1981). ISSN: 15577317. DOI: 10.1145/358669.358692.

[26] Academy Software Foundation. *OpenEXR*. [accessed: 2020-01-31]. URL: https://www.openexr.com/index.html.

[27] Blender Foundation. *Blender v2.5- a 3D modeling and rendering package*. [accessed: 2020-01-31]. Stichting Blender Foundation, Amsterdam, 2010. URL: http://www.blender.org.

[28] Blender Foundation. *Cycles*. [accessed: 2020-02-20]. URL: https://www.cycles-renderer.org/.

[29] Blender Foundation. *Cycles - Introduction*. [accessed: 2020-02-23]. URL: https://docs.blender.org/manual/en/2.80/render/cycles/introduction.html.

[30] Python Foundation. *Data Compression and Archiving*. [accessed: 2020-02-23]. URL: https://docs.python.org/3.7/library/archiving.html.

[31] Python Software Foundation. *timeit — Measure execution time of small code snippets*. [accessed: 2020-01-31]. 2017. URL: https://docs.python.org/3.5/library/timeit.html.

[32] Bill Gray. *star_cats*. [accessed: 2020-02-11]. URL: https://github.com/Bill-Gray/star_cats.

[33] Hanzi Wang, Daniel Mirota, and G.D. Hager. "A Generalized Kernel Consensus-Based Robust Estimator". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (Jan. 2010), pp. 178–184. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.148. URL: http://ieeexplore.ieee.org/document/5184846/.

[34] Bruce Hapke. "Bidirectional reflectance spectroscopy: 1. Theory". In: *Journal of Geophysical Research: Solid Earth* (1981). ISSN: 0148-0227. DOI: 10.1029/jb086ib04p03039.

[35] Bruce Hapke. "Bidirectional reflectance spectroscopy. 3. Correction for macroscopic roughness". In: *Icarus* (1984). ISSN: 10902643. DOI: 10.1016/0019-1035(84)90054-X.

[36] Bruce Hapke. "Bidirectional reflectance spectroscopy. 4. The extinction coefficient and the opposition effect". In: *Icarus* (1986). ISSN: 10902643. DOI: 10.1016/0019-1035(86)90108-9.

[37] Bruce Hapke. "Bidirectional reflectance spectroscopy. 5. The coherent backscatter opposition effect and anisotropic scattering". In: *Icarus* (2002). ISSN: 00191035. DOI: 10.1006/icar.2002.6853.

[38] Bruce Hapke. "Bidirectional reflectance spectroscopy. 6. Effects of porosity". In: *Icarus* (2008). ISSN: 00191035. DOI: 10.1016/j.icarus.2008.01.003.

[39] Bruce Hapke. "Bidirectional reflectance spectroscopy 7". In: *Icarus* (2012). ISSN: 00191035. DOI: 10.1016/j.icarus.2012.10.022.

[40] Bruce Hapke and Eddie Wells. "Bidirectional reflectance spectroscopy: 2. Experiments and observations". In: *Journal of Geophysical Research: Solid Earth* 86.B4 (1981), pp. 3055–3060. ISSN: 0148-0227. DOI: 10.1029/jb086ib04p03055.

[41] Gerald R. Hintz. "Fundamentals of Astrodynamics". In: *Orbital Mechanics and Astrodynamics*. Cham: Springer International Publishing, 2015, pp. 1–21. DOI: 10.1007/978-3-319-09444-1{_}1. URL: http://link.springer.com/10.1007/978-3-319-09444-1_1.

[42] Henry H. Hsieh. *Asteroid-comet continuum objects in the solar system*. 2017. DOI: 10.1098/rsta.2016.0259.

[43] International Astronomical Union. *IAU Resolution 5a: Definition of a "Planet" in the Solar System*. 2006. URL: https://www.iau.org/static/resolutions/Resolution_GA26-%205-6.pdf.

[44] *Introduction — Shader Nodes*. [accessed: 2020-02-14]. URL: https://docs.blender.org/manual/en/2.80/render/shader_nodes/introduction.html.

[45] Arnold Irschara et al. "Efficient structure from motion with weak position and orientation priors". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2011. ISBN: 9781457705298. DOI: 10.1109/CVPRW.2011.5981775.

[46] Michal Jancosek and Tomas Pajdla. "Exploiting Visibility Information in Surface Reconstruction to Preserve Weakly Supported Surfaces". In: *International Scholarly Research Notices* 2014 (2014), pp. 1–20. ISSN: 2356-7872. DOI: 10.1155/2014/798595.

[47] *JPL Small-Body Database Search Engine*. [accessed: 2020-02-22]. URL: https://ssd.jpl.nasa.gov/sbdb_query.cgi.

[48] James T. Kajiya. "The rendering equation". In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86*. New York, New York, USA: ACM Press, 1986, pp. 143–150. ISBN: 0897911962. DOI: 10.1145/15922.15902. URL: http://portal.acm.org/citation.cfm?doid=15922.15902.

[49] Michel Kasser and Yves Egels. *Digital Photogrammetry*. 2002. DOI: 10.4324/9780203305959.

[50] Tomas Kohout and Jan-erik Wahlund. "Asteroid Prospection Explorer (APEX) CubeSat for Hera mission Didymos Interior". In: *EPSC* 2019.2132 (2019), EPSC–DPS2019.

[51] Tomas Kohout et al. "Feasibility of asteroid exploration using CubeSats-ASPECT case study". In: *Advances in Space Research* 62.8 (Oct. 2018), pp. 2239–2244. ISSN: 18791948. DOI: 10.1016/j.asr.2017.07.036. URL: https://linkinghub.elsevier.com/retrieve/pii/S027311771730546X.

[52] Jet Propulsion Laboratory. *Asteroid and Comet Spacecraft Missions*. [accessed: 2020-02-01]. URL: https://ssd.jpl.nasa.gov/?targets.

[53] Eric Lafortune. "Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering". In: *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven* 20 (1995), pp. 74–79. DOI: 10.1.1.38.3626. URL: http://www.students.science.uu.nl/~3220516/advancedgraphics/papers/variance_reduction_techniques.pdf.

[54] D. S. Lauretta et al. "OSIRIS-REx: Sample Return from Asteroid (101955) Bennu". In: *Space Science Reviews* 212.1-2 (Oct. 2017), pp. 925–984. ISSN: 0038-6308. DOI: 10.1007/s11214-017-0405-1. URL: http://link.springer.com/10.1007/s11214-017-0405-1.

[55] Victor Lempitsky and Denis Ivanov. "Seamless mosaicing of image-based texture maps". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2007. ISBN: 1424411807. DOI: 10.1109/CVPR.2007.383078.

[56] Omar Adil Mahdi, Mazin Abed Mohammed, and Ahmed Jasim Mohamed. "Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique". In: *International Journal of Computer Science Issues* 9.6 (2012), pp. 53–59. ISSN: 1694-0784.

[57] Iain M. Martin, Martin N. Dunstan, and Manuel Sanchez Gestido. *Planetary Surface Image Generation for Testing Future Space Missions with PANGU*. Saratoga Springs, 2019. URL: https://pangu.software/wp-content/pangu_uploads/pdfs/SpaceImagingWorkshop_2019_paper_pangu_final.pdf.

[58] Lionel Moisan, Pierre Moulon, and Pascal Monasse. "Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers". In: *Image Processing On Line* 2 (May 2012), pp. 56–73. ISSN: 2105-1232. DOI: 10.5201/ipol.2012.mmm-oh. URL: http://www.ipol.im/pub/art/2012/mmm-oh/?utm_source=doi.

[59] Pierre Moulon, Pascal Monasse, and Renaud Marlet. "Adaptive Structure from Motion with a Contrario Model Estimation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2013, pp. 257–270. ISBN: 9783642374463. DOI: 10.1007/978-3-642-37447-0{_}20. URL: http://link.springer.com/10.1007/978-3-642-37447-0_20.

[60] Pierre Moulon, Pascal Monasse, and Renaud Marlet. "Global fusion of relative motions for robust, accurate and scalable structure from motion". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013. ISBN: 9781479928392. DOI: 10.1109/ICCV.2013.403.

[61] Pierre Moulon et al. *OpenMVG. An Open Multiple View Geometry library.* [accessed: 2020-01-30]. 2016. URL: https://github.com/openMVG/openMVG.

[62] NASA. *Four Sides of Asteroid Bennu*. [accessed: 2020-02-12]. URL: https://solarsystem.nasa.gov/resources/2318/four-sides-of-asteroid-bennu/?category=planets_jupiter.

[63] NASA/JPL/SSD. *How Many Solar System Bodies*. [accessed: 2020-02-01]. 2014. URL: http://ssd.jpl.nasa.gov.

[64] OpenMVS. *OpenMVS: open Multi-View Stereo reconstruction library*. [accessed: 2020-01-30]. 2020. URL: https://github.com/OpenMVS/openMVS.

[65] ESA/Rosetta/MPS for OSIRIS Team MPS/UPD/LAM/IAA/SSO/INTA/UPM/DASP/IDA. *OSIRIS Image Archive*. [accessed: 2020-02-12]. URL: https://rosetta-osiris.eu/image/NAC_2015-08-01T20.55.10.160?gallery=nucleus&ipp=24&page=54&sort=startTime&seq=asc&target=67p&content=N,NC,NR,NF&camera=NAC,WAC&filterNAC=Orange&filterWAC=Vis610,Red_WAC&maxImage=2596&targetDistanceFrom=50&targetDistanceTo=1000&maxPage=109&startImage=1273&lastImage=1296&imgNr=1279.

[66] ESA/Rosetta/MPS for OSIRIS Team MPS/UPD/LAM/IAA/SSO/INTA/UPM/DASP/IDA. *The Comet - OSIRIS Image Archive*. [accessed: 2020-02-12]. URL: https://rosetta-osiris.eu/.

[67] Alain Pagani et al. "Dense 3D Point Cloud Generation from Multiple High-resolution Spherical Images". In: *International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*. 2011. DOI: 10.2312/VAST/VAST11/017-024.

[68] Mihkel Pajusalu and Andris Slavinskis. "Characterization of Asteroids Using Nanospacecraft Flybys and Simultaneous Localization and Mapping". In: *2019 IEEE Aerospace Conference*. Vol. 2019-March. IEEE, Mar. 2019, pp. 1–9. ISBN: 978-1-5386-6854-2. DOI: 10.1109/AERO.2019.8741921. URL: https://ieeexplore.ieee.org/document/8741921/.

[69] E Palmer et al. "The Small Bodies Imager Browser—finding asteroid and comet images without pain". In: *Asteroids, Comets, Meteors 2014*. 2014.

[70] Matt Pharr and Greg Humphreys. "Chapter Nine - Materials". In: *Physically Based Rendering: From Theory to Implementation* (2010), pp. 476–499. DOI: 10.1016/B978-0-12-375079-2.50009-3.

[71] Armen Poghosyan and Alessandro Golkar. *CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions*. 2017. DOI: 10.1016/j.paerosci.2016.11.002.

[72] Véronique Pommier-Maurussane and Luc Maisonobe. "Orekit : an Open-source Library for Operational Flight Dynamics Applications". In: *4th International Conference on Astrodynamics Tools and Techniques*. 1. 2010.

[73] A. M. Price-Whelan et al. "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package". In: *The Astronomical Journal* 156.3 (Aug. 2018), p. 123. ISSN: 1538-3881. DOI: 10.3847/1538-3881/aabc4f. URL: https://iopscience.iop.org/article/10.3847/1538-3881/aabc4f.

[74] Dc Richardson and Zm Leinhardt. "Gravitational aggregates: Evidence and evolution". In: *Asteroids III* (2002).

[75] Andrew S. Rivkin et al. "The Main-belt Asteroid and NEO Tour with Imaging and Spectroscopy (MANTIS)". In: *IEEE Aerospace Conference Proceedings*. Vol. 2016-June. IEEE. 2016, pp. 1–14. ISBN: 9781467376761. DOI: 10.1109/AERO.2016.7500757.

[76] Francisco Sans and Rhadamés Carmona. "A Comparison between GPU-based Volume Ray Casting Implementations: Fragment Shader, Compute Shader, OpenCL, and CUDA". In: *CLEI Electronic Journal* (2017). ISSN: 0717-5000. DOI: 10.19153/cleiej.20.2.7.

[77] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.445.

[78] Gabriel Jörg Schwarzkopf and Mihkel Pajusalu. "Space Imaging Simulator for Proximity Operations". In: (Feb. 2020). DOI: 10.5281/ZENODO.3661054. URL: https://doi.org/10.5281/zenodo.3661054.

[79] Y. Shkuratov et al. "A critical assessment of the Hapke photometric model". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 113.18 (Dec. 2012), pp. 2431–2456. ISSN: 00224073. DOI: 10.1016/j.jqsrt.2012.04.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022407312001926.

[80] Andris Slavinskis et al. "Nanospacecraft fleet for multi-asteroid touring with electric solar wind sails". In: *2018 IEEE Aerospace Conference*. Vol. 2018-March. IEEE, Mar. 2018, pp. 1–20. ISBN: 978-1-5386-2014-4. DOI: 10.1109/AERO.2018.8396670. URL: https://ieeexplore.ieee.org/document/8396670/.

[81] Colin Snodgrass and Geraint H. Jones. "The European Space Agency's Comet Interceptor lies in wait". In: *Nature Communications* 10.1 (Dec. 2019), p. 5418. ISSN: 2041-1723. DOI: 10.1038/s41467-019-13470-1. URL: http://www.nature.com/articles/s41467-019-13470-1.

[82] R. H. Soja et al. "IMEM2: a meteoroid environment model for the inner solar system". In: *Astronomy & Astrophysics* 628 (Aug. 2019), A109. ISSN: 0004-6361. DOI: 10.1051/0004-6361/201834892. URL: https://www.aanda.org/10.1051/0004-6361/201834892.

[83] Peter Späth. *Advanced Audio Visualization Using ThMAD*. Apress, 2018. DOI: 10.1007/978-1-4842-3504-1.

[84] Dale Stanbridge et al. "Lucy: Navigating a Jupiter Trojan tour". In: *Advances in the Astronautical Sciences* 162 (2018), pp. 3781–3798. ISSN: 00653438.

[85] *Stardust - Comet Wild 2 Images*. [accessed: 2020-02-13]. URL: https://stardust.jpl.nasa.gov/photo/cometwild2.html#row3.

[86] Otto Struve. "The International Astronomical Union". In: *Science* 117.3039 (Mar. 1953), pp. 315–318. ISSN: 0036-8075. DOI: 10.1126/science.117.3039.315. URL: https://www.sciencemag.org/lookup/doi/10.1126/science.117.3039.315.

[87] Peter Sturm. "Pinhole Camera Model". In: *Computer Vision*. Boston, MA: Springer US, 2014, pp. 610–613. DOI: 10.1007/978-0-387-31439-6{_}472. URL: http://link.springer.com/10.1007/978-0-387-31439-6_472.

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

[89] Tricia Talbert. *Double Asteroid Redirection Test (DART) Mission | NASA*. [accessed: 2020-02-01]. June 2019. URL: https://www.nasa.gov/planetarydefense/dart.

[90] Shaharyar Ahmed Khan Tareen and Zahra Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". In: *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, iCoMET 2018 - Proceedings*. 2018. ISBN: 9781538613702. DOI: 10.1109/ICOMET.2018.8346440.

[91] The Astropy Collaboration et al. "Astropy: A community Python package for astronomy". In: *Astronomy & Astrophysics* 558 (July 2013), A33. ISSN: 0004-6361. DOI: 10.1051/0004-6361/201322068. URL: http://www.aanda.org/10.1051/0004-6361/201322068.

[92] Enrico Valenza. *Blender Cycles: Materials and Textures Cookbook*. Packt Publishing Ltd, 2015.

[93] Elena Vasiou et al. "A detailed study of ray tracing performance: render time and energy cost". In: *Visual Computer* 34.6-8 (June 2018), pp. 875–885. ISSN: 01782789. DOI: 10.1007/s00371-018-1532-8.

[94] Luiz Velho and Jonas Sossai. "Projective texture atlas construction for 3D photography". In: *Visual Computer*. Vol. 23. 9-11. 2007, pp. 621–629. DOI: 10.1007/s00371-007-0150-7.

[95] H-H Vu et al. "High Accuracy and Visibility-Consistent Dense Multiview Stereo". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (May 2012), pp. 889–901. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.172. URL: https://academic.microsoft.com/paper/2165306775%20http://ieeexplore.ieee.org/document/5989831/.

[96] Michael Waechter, Nils Moehrle, and Michael Goesele. "Let there be color! Large-scale texturing of 3D reconstructions". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8693 LNCS. PART 5. 2014, pp. 836–850. DOI: 10.1007/978-3-319-10602-1{_}54.

[97] Kevin J. Walsh. "Rubble Pile Asteroids". In: *Annual Review of Astronomy and Astrophysics* 56.1 (Sept. 2018), pp. 593–624. ISSN: 0066-4146. DOI: 10.1146/annurev-astro-081817-052013. URL: https://www.annualreviews.org/doi/10.1146/annurev-astro-081817-052013.

[98] Kevin J. Walsh. "Rubble Pile Asteroids". In: *Annual Review of Astronomy and Astrophysics* 56.1 (Sept. 2018), pp. 593–624. ISSN: 0066-4146. DOI: 10.1146/annurev-astro-081817-052013. URL: https://www.annualreviews.org/doi/10.1146/annurev-astro-081817-052013.

[99] Sei ichiro Watanabe et al. *Hayabusa2 Mission Overview*. 2017. DOI: 10.1007/s11214-017-0377-1.

[100] S. Watanabe et al. "Hayabusa2 arrives at the carbonaceous asteroid 162173 Ryugu-A spinning top-shaped rubble pile". In: *Science* 364.6437 (Apr. 2019), pp. 268–272. ISSN: 10959203. DOI: 10.1126/science.aav8032.

[101] Paul Weissman et al. *Origin and Evolution of Cometary Nuclei*. Feb. 2020. DOI: 10.1007/s11214-019-0625-7.

[102] Bo Zhang, Josiane Zerubia, and Jean Christophe Olivo-Marin. "Gaussian approximations of fluorescence microscope point-spread function models". In: *Applied Optics*. Vol. 46. 10. OSA - The Optical Society, Apr. 2007, pp. 1819–1829. DOI: 10.1364/AO.46.001819.

A Shader Node Network

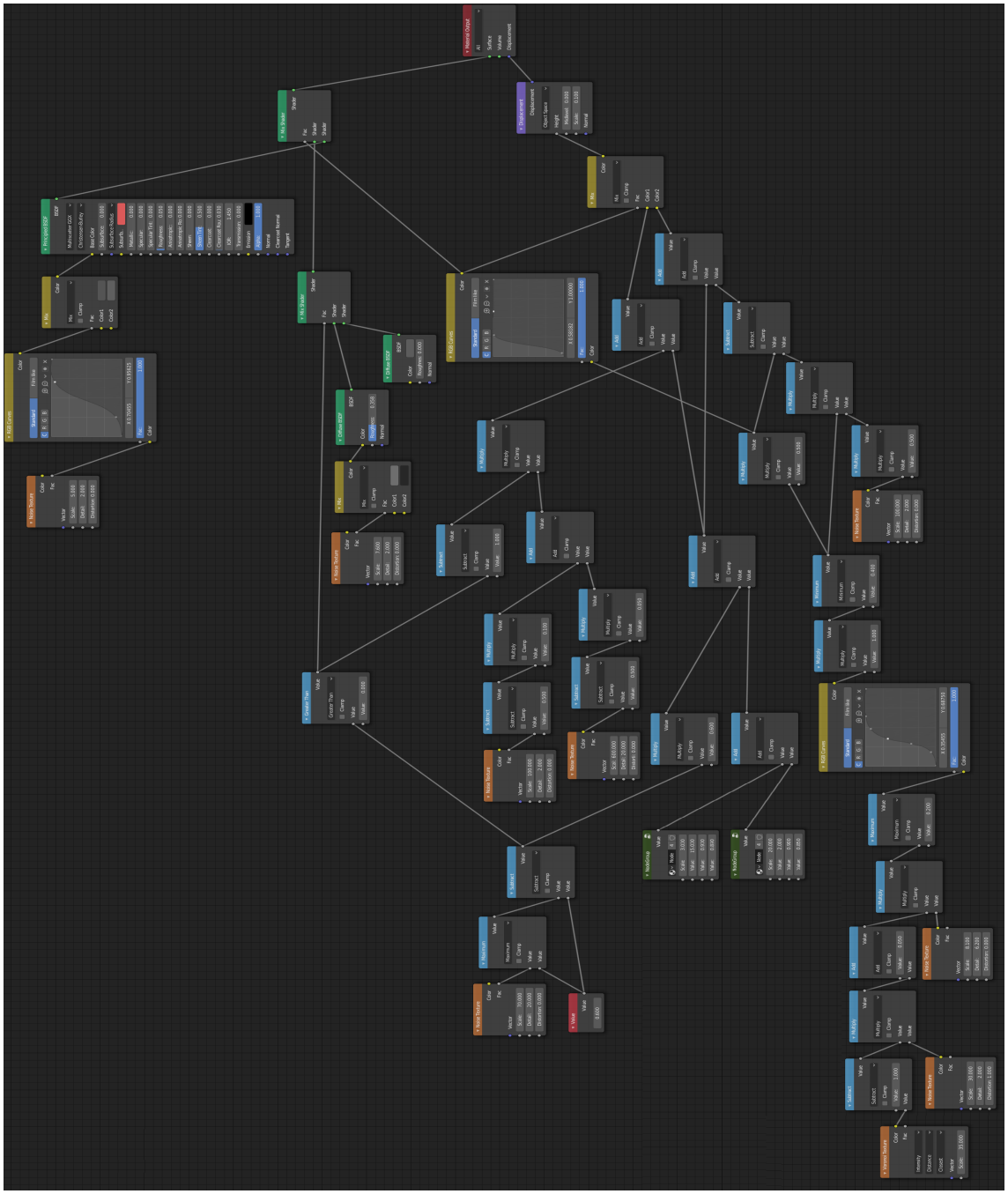


Figure A1: High resolution image of the shader node network presented in Figure 9b used in SISPO. Two main trees are visible in the network that are combined in the output, one for shading and the other for displacement. The two main trees are interlinked, i.e. displacement and shading are partially influenced by the same input.

B Image Set for Image Processing Benchmark

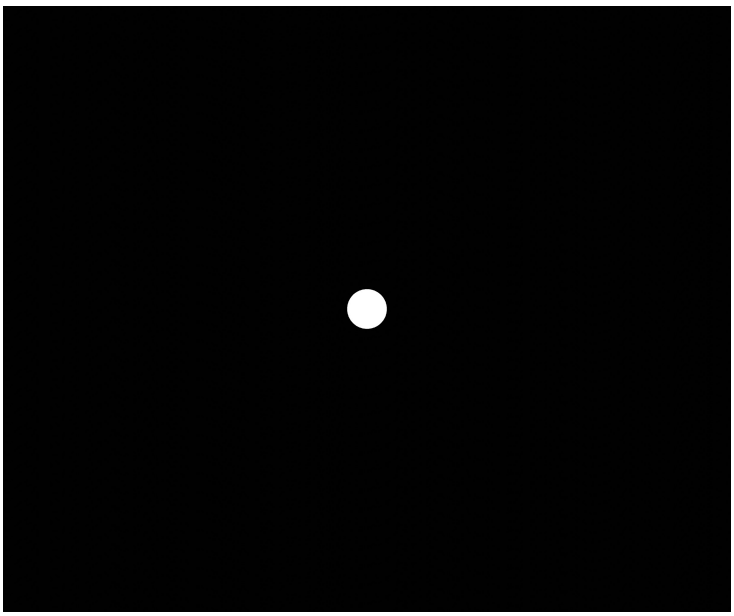


Figure B1: LightRef image used in benchmark.

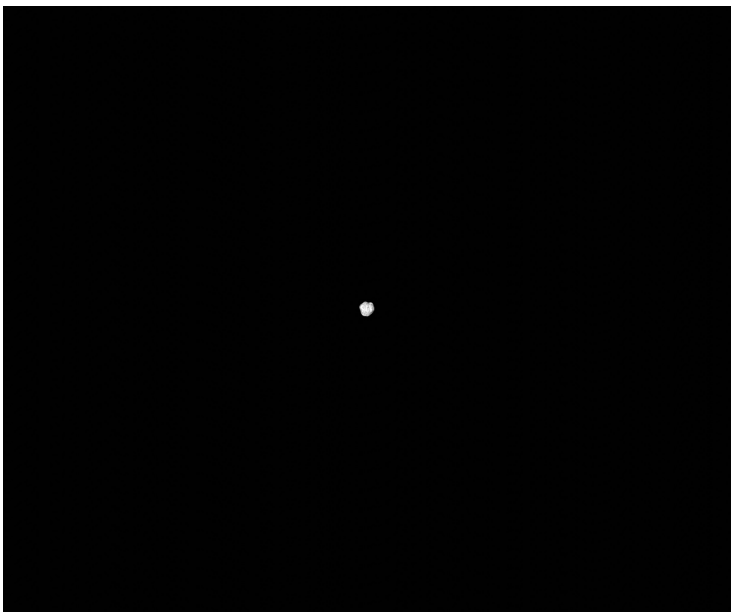


Figure B2: SssbConstDist image used in benchmark.

Figures B1 through B5 are the images used in the benchmark. Their names are given in the respective caption.

Figures B6 and B7 show the difference of the SssbOnly image from the benchmark on the two different computers. Only a small fraction of pixels differ. As discussed

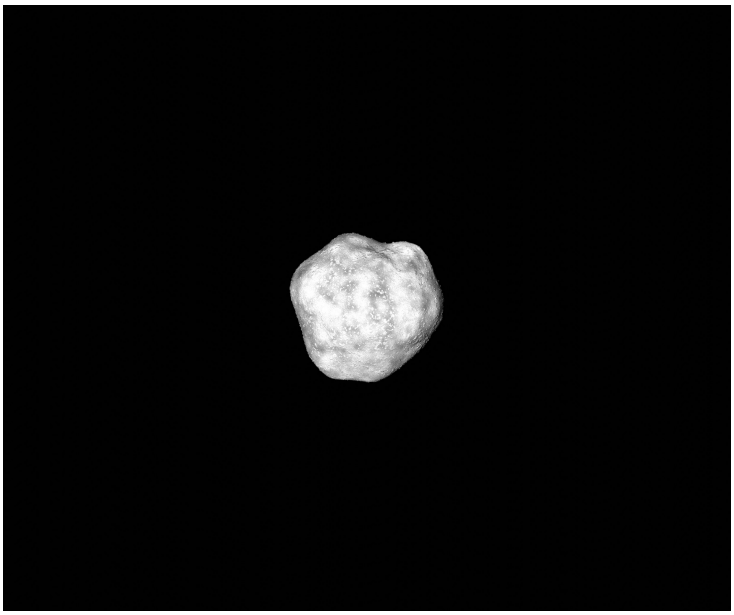


Figure B3: SssbOnly image used in benchmark.

Figure B4: Stars1 image used in benchmark. The image is based on 1804 stars.

previously, the maximum absolute difference is on the order of 1×10^{-6}, corresponding to the brightest point in each image.

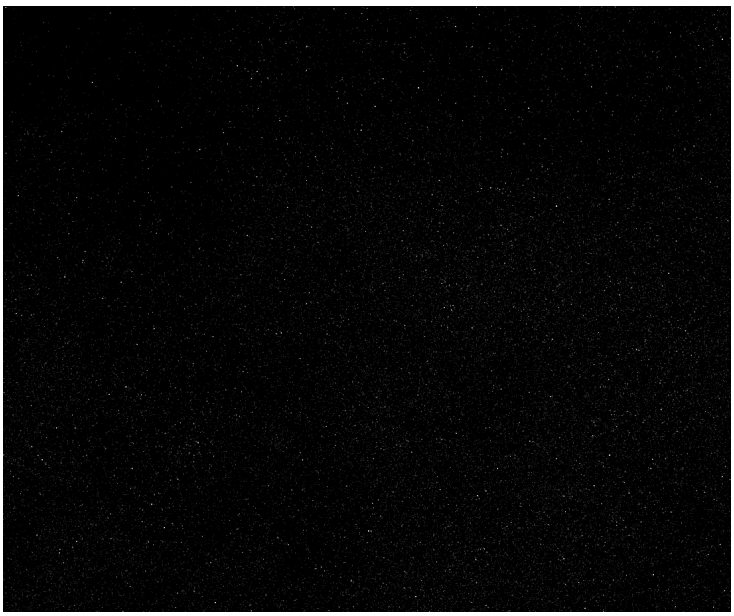


Figure B5: Stars2 image used in benchmark. The image is based on 51338 stars.

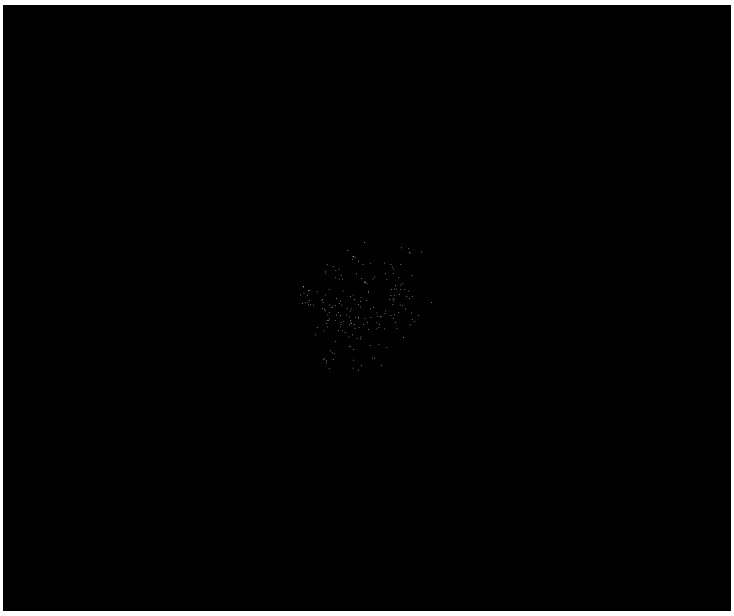


Figure B6: Difference between skimage and OpenCV Gaussian filtered SssbOnly image of the laptop.

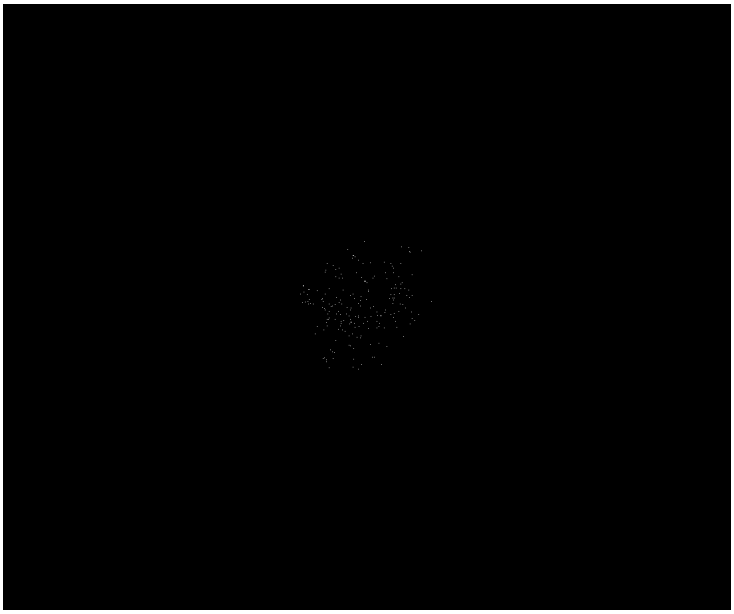


Figure B7: Difference between skimage and OpenCV Gaussian filtered SssbOnly image of the workstation computer.