A virtual test environment for validating spacecraft optical navigation

S. M. Parkes
I. Martin
M. N. Dunstan
N. Rowell
nickrowell@computing.dundee.ac.uk
University of Dundee
Space Technology Centre
Dundee, UK

O. Dubois-Matra (Telespazio VEGA UK Ltd)
T. Voirin
European Space Agency
Noordwijk
The Netherlands

ABSTRACT

The use of machine vision to guide robotic spacecraft is being considered for a wide range of missions, such as planetary approach and landing, asteroid and small body sampling operations and in-orbit rendezvous and docking. Numerical simulation plays an essential role in the development and testing of such systems, which in the context of vision-guidance means that realistic sequences of navigation images are required, together with knowledge of the ground-truth camera motion. Computer generated imagery (CGI) offers a variety of benefits over real images, such as availability, cost, flexibility and knowledge of the ground truth camera motion to high precision. However, standard CGI methods developed for terrestrial applications lack the realism, fidelity and performance required for engineering simulations.

In this paper, we present the results of our ongoing work to develop a suitable CGI-based test environment for spacecraft vision guidance systems. We focus on the various issues involved with image simulation, including the selection of standard CGI techniques and the adaptations required for use in space applications. We also describe our approach to integration with high-fidelity end-to-end mission simulators, and summarise a variety of European Space Agency research and development projects that used our test environment.

Paper No. 3931. Manuscript received 3 December 2012, revised version received 9 May 2013, accepted 26 June 2013.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

NOMENCLATURE

P	projection matrix (camera to clipping frame transformation)
V	view matrix (world to camera transformation)
x_{clip}	position vector in clipping frame
x_{world}	position vector in world frame
π_{clip}	plane equation in world frame
T	denotes Matrix Transpose
z_{far}	distance to far clipping plane along camera boresight
z_{near}	distance to near clipping plane along camera boresight
AMICA	Asteroid Multi-band Imaging Camera
API	Application Programming Interface
BRDF	Bidirectional Reflectance Distribution Function
CGI	Computer Generated Imagery
DKE	Dynamics, Kinematics and Environment
ECSS	European Cooperation on Space Standardisation
EDL	Entry, Descent and Landing
ESA	European Space Agency
GLSL	OpenGL Shading Language
GNC	Guidance, Navigation and Control
GPU	Graphics Processing Unit
MER	Mars Exploration Rovers
NASA	National Aeronautics and Space Administration
NEO	Near Earth Object
PANGU	Planet and Asteroid Natural Scene Generation Utility
R&D	Research and Development
SLAM	Simultaneous Localisation and Mapping
SPICE	Spacecraft, Planet, Instrument, C-matrix, Events
SURF	Speeded Up Robust Features
TCP/IP	Transmission Control Protocol/Internet Protocol
VMC	Visual Monitoring Camera

1.0 INTRODUCTION

With the exception of the Moon, the scientific exploration of remote parts of the Solar System has been conducted entirely by robotic spacecraft. This requires advanced guidance systems, coupled with carefully selected navigation sensor arrays and, where appropriate, control autonomy to support missions where direct teleoperation from Earth is not feasible due to the round-trip time delay on radio signals or bandwidth and broadcast time constraints.

Within the framework of the European Space Agency's (ESA) Aurora Programme[1] and the National Aeronautics and Space Administration's (NASA) Solar System Exploration Roadmap[2], the next generation of Solar System exploration missions will see much of the guidance, navigation and control (GNC) responsibility delegated to the spacecraft onboard systems. The potential for precision guidance, in the absence of teleoperation constraints, brings a much wider range of missions into scope, such as pin-point landing on planets and asteroids, and in-orbit rendezvous and docking. Machine vision is currently under investigation as the crucial enabling technology for precision guidance of spacecraft. The use of cameras as navigation sensors has been successful

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

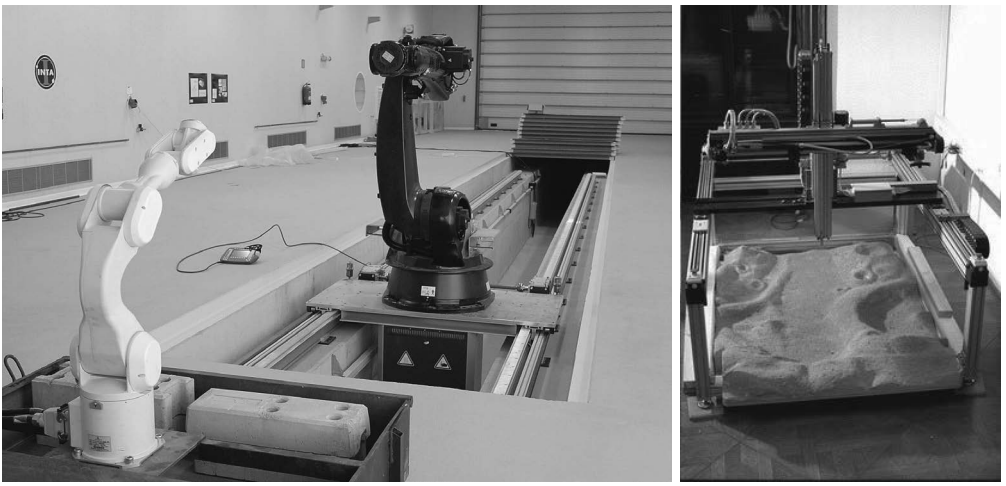
in a range of mission scenarios to date, including: velocity estimation during descent and landing (EDL) onto Mars[3]; orbit determination[4] and proximity sampling operations[5] targeting asteroids; and navigation of the Mars Exploration Rovers (MER)[6].

Several technology demonstration and proof-of-concept missions are either currently flying, or planned for the near future. The Swedish Prisma mission, launched in June 2010, is an in-orbit test bed for formation flying and rendezvous systems using a variety of onboard sensors including optical cameras[7]. It consists of two spacecraft that perform a series of experiments focussing on GNC systems to demonstrate various key technologies of multi-satellite missions. This includes an autonomous rendezvous from 100km separation using far and close range cameras as primary navigation sensors[8]. ESA's Lunar Lander mission is intended to demonstrate key technologies for a future Mars Sample Return mission within the Aurora Programme. Although funding for the mission has recently been withdrawn, technology development projects and related work are continuing. The main objective of this mission scenario is an autonomous, safe and precise landing at the Lunar South Pole. The use of vision based navigation is seen as vital to achieving this. A combined camera and dedicated image processing board has been developed to support this mission scenario[9], with intensive image processing tightly integrated to the navigation filter.

Looking further ahead, several groups are investigating the application of terrestrial vision based navigation techniques to the space domain, where computing power is typically the main limitation. Simultaneous Localisation and Mapping (SLAM) (see Ref. 10 and references therein) potentially offers very accurate target relative navigation using vision, and may have applications in rendezvous and docking with an uncooperative target[11], and pin point landing on asteroids[12,13].

1.1 Vision based GNC test environments

With developments in vision-based GNC technology comes the need for suitable test environments. Existing sets of navigation sensor data in conjunction with ground truth sensor motion can be



(a) rendezvous and docking simulator (b) lunar EDL simulator

Figure 1. Image (a) shows a satellite rendezvous and docking simulation facility, including the robotic arms mounted on rails that maneuver the chaser and target spacecraft (Instituto Nacional de Técnica Aeroespacial (INTA) and Grupo Mecánica de Vuelo (GMV) Flight Mechanics Group). Image (b) shows CamRobII apparatus, which incorporates a scale model of a Lunar-type surface with a moving camera rig (Joanneum Research/Gerhard Paar).

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

used for this purpose. Navigation images downlinked from real spacecraft provide one source, for example the MER descent images[14]; however these are severely limited in number and scope, and are only really representative of the original mission scenario and hardware. The nature of spacecraft navigation means that field testing is extremely costly and hazardous to perform, even in terrestrial analogue environments such as the Mojave and Moroccan deserts[14,15]. In a similar vein, large scale dynamic simulation facilities exist for performing tests using mock-ups of real scenes, in particular in-orbit rendezvous and docking simulations using full-scale satellite replicas on moveable gantries and real sensors – see Fig. 1(a) and Refs 17,18. On the smaller scale, laboratory mock-ups of planetary surfaces with moving camera rigs have been used in the past for generating simulated images for EDL and rover navigation, for example CamRobII[19,20] shown in Fig. 1(b). Equipment such as this is typically expensive, slow and unrepeatable. As scale models of planetary surfaces are difficult to make, there is also the risk of inadvertently optimising the navigation system for a specific terrain model. While field trials such as these are of course vital for exercising hardware and validating complete systems prior to deployment, they are simply not feasible for testing early GNC algorithm conceptual designs.

1.2 Computer simulation

For early testing and validation of GNC algorithm concepts, computer simulation of navigation images and other sensor data offers a cheap and flexible alternative to field trials. Although limited in scope, computer simulation has a variety of benefits over field test data. For example, the ground truth sensor motion is known to arbitrary precision, whereas even the best field trials or scale models will be subject to errors and engineering tolerances. Synthetic sensor data is also noise-free, enabling users to employ their own tightly controlled noise models and tune guidance algorithms for optimal performance in their intended scenario, and the ability to regenerate 3D terrain models with ease allows for more rigorous Monte Carlo test regimes covering a wider range of scenarios. However, a virtual test environment based on computer simulation must satisfy two principal requirements in order to be of practical use to GNC developers: it must be both realistic, and fast.

Both of these requirements can be met by leveraging the hardware acceleration power of modern 3D graphical processing units (GPUs), which provide a highly parallelised data processing pipeline for real time image generation and other calculations. We note that real time image generators (‘rendering engines’) for simulators, games and scientific visualisation are nothing new – OGRE ? and the Quake engine are two popular options in the public domain – and techniques developed



(a) (b) (c)

Figure 2. Selection of example PANGU images: asteroid and Earth including atmosphere (left); Martian surface image with aerial dust (centre); the Automated Transfer Vehicle with background stars (right).

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

for these applications provide a good starting point. However, high-integrity rendering of celestial scenes poses several major problems to traditional rendering engines, which are designed generally for terrestrial scenes, and often sacrifice certain subtle aspects of the realism in order to improve aesthetic appeal and reduce rendering time.

Over the last ten years, the University of Dundee Space Technology Centre has developed a virtual test environment for spacecraft vision based guidance. The development of the Planet and Asteroid Natural Scene Generation Utility (PANGU) has been the focus of several PhDs and ESA-funded contracts, and is being used in various ESA projects and feasibility studies for vision-based GNC design. In PANGU, many popular 3D modelling and CGI techniques have been adopted, refined and optimised for use in the space domain. Certain restricted PANGU features have been published at conferences over the years during incremental development, see for example Refs 22-28 and references therein. However, in this paper we take a bottom-up approach and explain how we simulate engineering quality images using GPUs to achieve at least real time performance, with particular emphasis on the pitfalls in applying standard CGI techniques to space and how these can be worked around. The main requirements driving the design of PANGU have been both the need for a high degree of realism and integrity, and the need for faster than real time performance to support intensive Monte Carlo simulation campaigns.

In Section 2, we provide a brief review of standard 3D graphics pipeline design, to give some context to PANGU developments. In Section 3 we explain how image simulation is performed and the problems that are encountered when dealing with large-scale celestial scenes. In Sections 4 and 5.2 we explain how our virtual test simulator can interface with high fidelity functional engineering simulators for the validation of GNC systems, and review several ESA projects and feasibility studies where our virtual test simulator was used in this manner. In Section 6 we draw conclusions and outline areas for future research.

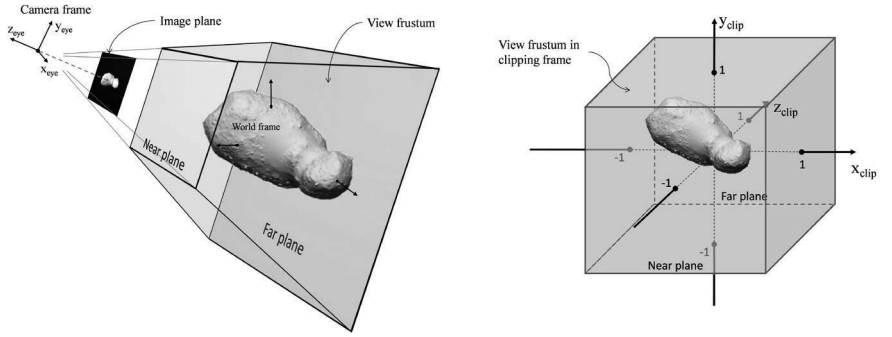
2.0 REVIEW OF 3D GRAPHICS ENGINE DESIGN

Modern 3D computer graphics engines are formidable pieces of software that perform a wide range of tasks beyond simply drawing polygons onto the screen. For large and complex environments a great deal of effort is required simply to manage the hierarchy of objects that inhabit the 3D world, their properties and their interactions. This scene graph management is ultimately responsible for feeding the rendering system with 3D objects to draw, and efficient scene graph management is vital in achieving real time performance. Almost all modern graphics engines perform the rendering tasks in dedicated GPUs, most often making use of cross-platform APIs such as OpenGL and DirectX to interface with the GPU via a set of standard routines. This alleviates the laborious task of triangle and polygon rasterisation, which is suitable for parallelisation and historically was a major processing bottleneck for software rendering. The rendering process is appropriately described as a pipeline, with geometric entities fed in one end and raster images produced from the other. The pipeline operations vary from application to application, but most 3D engines share a common set of essential processes. We focus on these throughout this section, and refer the interested reader to Ref. 29 for a broader introduction to 3D graphics engines, and to the Game Programming Gems series (e.g. Ref. 30) for more detailed issues of implementation and optimisation.

2.1 Scene graph management

Several examples of the kinds of object that inhabit a simple computer graphics 3D environment are light sources, geometric primitives such as vertices, normal vectors and vertex interconnections

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) world and eye coordinates (b) clipping coordinates

Figure 3. The view frustum (a truncated pyramid in the world and camera frames) defines the visible region of space, with the camera usually considered to point along the negative axis. Objects are transformed via perspective projection to the clipping frame, where the view frustum forms a cube. Note that the clipping frame is left-handed by convention. Objects lying outside this region are 'clipped' from view. In this figure, there is a single visible object – the asteroid 25143 Itokawa.

that define polygonal surfaces, textures (2D images that can be 'pasted' onto 3D objects), and diffuse objects and atmospheric entities such as smoke, flame and gas. Each object can be endowed with certain properties such as position, orientation, colour, surface reflectance and optical depth. Physical objects such as asteroids are defined by sets of geometric primitives, themselves defined in some appropriate model-fixed co-ordinate frame. Composite objects such as multi-component spacecraft models are built from heirarchical sets of geometric primitives, often arranged into a tree structure with the object itself at the root node, a different component at each child node and geometric primitives at the leaves. The complete set of objects and entities in the 3D environment is itself arranged into a logical structure referred to as the scene graph, often implemented as a tree structure.

Scene graph management provides the interface between the user and the rendering system, and has the task of managing the properties of all objects and the rendering state based on commands from the user and interactions between the objects themselves. This may involve commands such as specifying the position of an object, or turning a light source on or off, or activating certain rendering options. In the context of game programming, interactions between objects may include collisions, explosions, dialogue and commands derived from artificial intelligence. However, most of these are irrelevant to the present context, where we are concerned mainly with image simulation. In this respect, the primary role of scene graph management is to identify the subset of objects that are presently visible, and to send them to the rendering pipeline in a recursive manner by performing a traversal of the scene graph. At this stage, the visibility of each object can only be tested in a rudimentary manner by checking if it lies completely outside of the visible region of space, which in three dimensions is represented by a truncated pyramid with the camera at the apex. This is known as the view frustum, and is a fundamental concept in computer graphics. The notion of the view frustum, and the world and camera co-ordinate frames that define the relative positions and orientations of objects and the camera, is illustrated in Fig. 3. Each object has a tightly fitting bounding volume (e.g. a sphere), which can be checked quickly to see whether it lies completely outside the view frustum by measuring the distance to each of the six frustum planes in turn.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

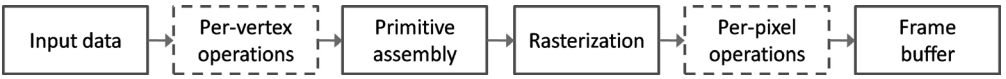


Figure 4. Flowchart depicting the high level functions of a rendering pipeline. The behaviour of much of the pipeline is fixed; the dashed boxes indicate stages that can be reprogrammed by the user by writing shader functions.

It is common for several 'rendering passes' to be performed for each image, where each pass renders only certain aspects of the scene, such as shadows, reflected images, atmospheric effects or solid objects. In a simplified picture, each pass involves transmitting geometric primitives, textures and lighting information to the rendering pipeline, and causes certain regions of the image to be drawn.

2.2 Rendering pipeline

The order of operations in a typical rendering pipeline is depicted in Fig. 4. Graphics processing units are built to perform these operations in hardware, and as such the functionality of many stages is fixed. The dashed boxes indicate stages that can be reprogrammed by the user in modern GPUs, allowing more flexibility and control over the rendering process. This is done by writing shader functions (or simply shaders) in a suitable graphics programming language such as GLSL[31]. The use of shaders is key to unlocking the full power of GPUs, and they are used extensively in our work to introduce a wide range of effects, such as exotic material reflectance functions (Section 3.6) and atmospheric processes (Section 3.5).

The initial stages are concerned with transforming geometric primitives between a number of different co-ordinate frames. This includes transforming the 3D positions of vertices from some external 'world' co-ordinate frame to eye coordinates where the camera is at the origin, then projecting these into clipping coordinates where the view frustum planes transform to a cube. This is a useful representation for culling vertices that lie outside of the image, and for performing operations like depth and intersection tests. The components of the position vector of a vertex transform between the world frame and the clipping frame according to the vector transformation

$$\underline{x}_{clip} = PV\,\underline{x}_{world} \qquad \ldots (1)$$

where V is the 'view matrix' that rotates and translates vertices to eye coordinates, and P is the projection matrix that transforms vertices to clipping coordinates. P will turn out to be an important quantity in later discussions of numerical accuracy. Both V and P are transmitted to the rendering pipeline and stored in hardware during rendering. Vertex transformations, and other operations that act on individual vertices such as lighting calculations, are collectively referred to as per-vertex operations, and shader functions that override this stage of the pipeline are called vertex shaders.

The intermediate stages of primitive assembly and rasterisation are concerned with connecting transformed vertices to produce polygons in the image plane ('fragments'), and then chopping these up into pixels. At this point, the colours of polygons at the position of each pixel are computed by first interpolating the colours and surface normals of neighbouring vertices. The directions toward the camera and light source relative to the surface normal are used in conjunction with a Bidirectional Reflectance Distribution Function (BRDF) to compute the final colour and brightness of the pixel. The precise form of the BRDF subtly affects the appearance of rendered polygons, and judicious selection of BRDF is the key to simulating different types of material in computer graphics.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

The distance from the camera centre to each pixel is calculated at this stage, and can be used to add other effects such as fog and atmospheric extinction, and additional effects such as shadows may be added at this stage too using various techniques. These operations are collectively referred to as per-pixel operations, and shader functions that override this stage of the pipeline are called pixel shaders or fragment shaders.

3.0 SPACE IMAGE SIMULATION

In this section, we discuss important aspects of the design of our celestial image simulator. These include various numerical issues that arise when handling large 3D environments, issues arising from the necessity for real-time performance, and various techniques in computer graphics used to provide the degree of realism required for functional engineering simulators.

3.1 Low level design considerations

3.1.1 Numerical precision at large distances

One of the main factors affecting the low level design of our image simulator is the sheer scale of the environments that need to be rendered. For Earth orbit applications, the camera is at a very large distance from the origin of the Earth Centred Earth Fixed frame. Therefore, the camera view matrix V requires a large dynamic range in order to avoid problems of quantisation when transforming vertices to eye coordinates. Standard GPU-accelerated rendering pipelines, such as that described above, store the view matrix in hardware often at single- or even half-precision floating point accuracy. This is sufficient for standard 3D environments that render scenes on local terrestrial scales, but is simply not accurate enough for many space applications. We therefore perform all vertex transformations in software at double precision, before sending them to the rendering pipeline. This incurs some performance cost, as allowing the GPU to perform the transforms is considerably faster.

3.1.2 Numerical precision at large near/far plane ratios

Another problem arises when the ratio of the farthest to the nearest object from the camera is very large. This ratio is used in the projection matrix P that transforms vertices from eye coordinates to clipping coordinates, and when it becomes too large the transformation can suffer severe numerical errors. For example, clipping coordinates are used to compare the depth of overlapping polygons and determine which is closest to the camera and should be drawn in the image. Section 3.3 discusses ways to alleviate this particular problem. It is also common to use the boundaries of the clipping volume to determine which objects lie outside the view frustum prior to the rendering stage. This is known as view culling and speeds up the rendering process by skipping objects that do not appear in the final image. The boundaries of the view volume form a cube in the clipping co-ordinate frame, and each planar face $\underline{\pi}$ is transformed to the world frame using the plane transformation equation

$$\underline{\pi}_{\text{world}} = (PV)^T \underline{\pi}_{\text{clip}} \quad \ldots (2)$$

The distance of each object to these planes is then checked, and any object lying completely outside the view volume is culled. Loss of precision in the projection matrix can therefore also cause objects to be incorrectly culled. Careful selection of the near and far plane distances so

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

that the view volume tightly fits the visible objects can help alleviate this problem, but in certain situations large far/near ratios are simply unavoidable, such as rendezvous simulations where a nearby satellite and distant moon may appear in the image at the same time. This problem is again solved by calculating the frustum planes in software at double precision.

3.1.3 Model geometry

The way we generate 3D models of spacecraft and planetary surfaces is somewhat detailed and is outwith the scope of this paper. However, there are certain aspects of the design that differ to conventional computer graphics techniques, and relate directly to achieving rapid rendering and high integrity images in our test environment.

It is common practice in computer graphics engines to use various effects to provide the illusion of 3D structure in a model, where in fact there is none. This is used to reduce the amount of memory required to store a model and speed up the rendering process. These techniques include the use of texture mapping to paste images ('textures') onto 2D surfaces to give the appearance of structure. An example might be the use of gravel textures to simulate a grainy surface. Taking this one stage further, it is also common to render objects individually into smaller images, and paste these onto planar polygons in the scene, so as to be indistinguishable from the original 3D object. This 'imposter' image is then used for several frames until the position or orientation of the object has changed appreciably, and the error between the true appearance of the object and the imposter image is significant. This splits the rendering load over several frames, at the expense of introducing systematic errors on the appearance of the object. Another technique called bump or normal mapping involves disturbing the surface normals to alter the way that light reflects from a flat surface, simulating microfacets or sloped regions. The problem with these techniques is that they only reproduce the superficial effects of genuine 3D, such as the shading of a surface in response to the slope. Vision guidance often relies on more subtle 3D effects such as parallax and foreshortening to derive relative motion and shape estimates, so the use of these techniques could severely compromise the integrity of our test enviroment.

Our models therefore rely on a large number of vertices and polygons to accurately reproduce all 3D effects. The drawback is that the additional geometry takes longer to process. We alleviate this by using exclusively triangular strips to connect vertices into surface mesh segments. Triangular strips provide a highly efficient means of transferring geometric primitives to the graphics pipeline and storing models in memory, because once the first two vertices are specified, every following vertex defines a complete triangle by implicitly connecting it to the preceeding two vertices.

3.2 Imaging model

Hardware accelerated rendering pipelines typically provide two types of image projection: that of orthographic projection, where a 2D image is obtained by simply collapsing the 3D scene along one direction, and perspective projection based on the perfect pinhole camera model (see e.g. Ref 32 and Fig. 3). Real cameras are most closely represented by the perspective projection model, though there are a number of non-ideal effects that are not reproduced, such as the radial distortion and depth of focus inherent to lens-based projection systems. Real cameras may also suffer from scattered light within the optical system, and from blurring due to imperfect optics and uncollimated light, typically characterised by a point spread function that describes the distribution of light at the focal plane received from a point source. At the detector end, multiple sources of noise can conspire to produce less than perfect images. These include thermal noise (photoelectrons liberated by thermal fluctuations in the detector), cosmic rays, shot noise (statistical noise in photon numbers, which is a Poisson process) and readout noise (noise introduced by readout electronics). The sensitivity of individual pixels to incoming light also typically varies across the detector.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

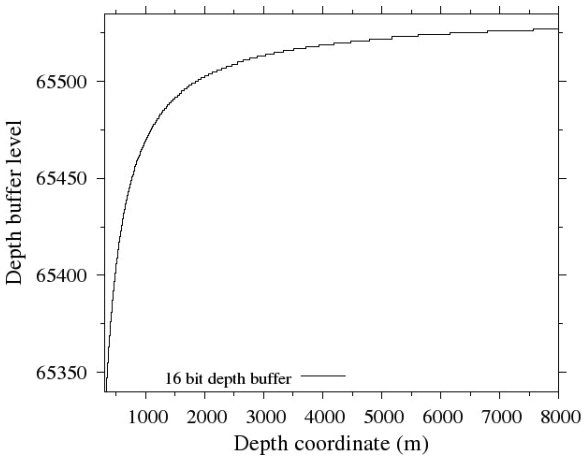


Figure 5. Quantisation of depth coordinate in finite-size depth buffer. At large distances, consecutive levels in the depth buffer can represent large changes in physical depth – in this example, all distances between ~6,200m and ~6,800m map to the same depth buffer level, so all sensitivity within this range is lost. This can lead to artefacts at large distances, where the rendering system is unable to correctly determine the closest polygon.

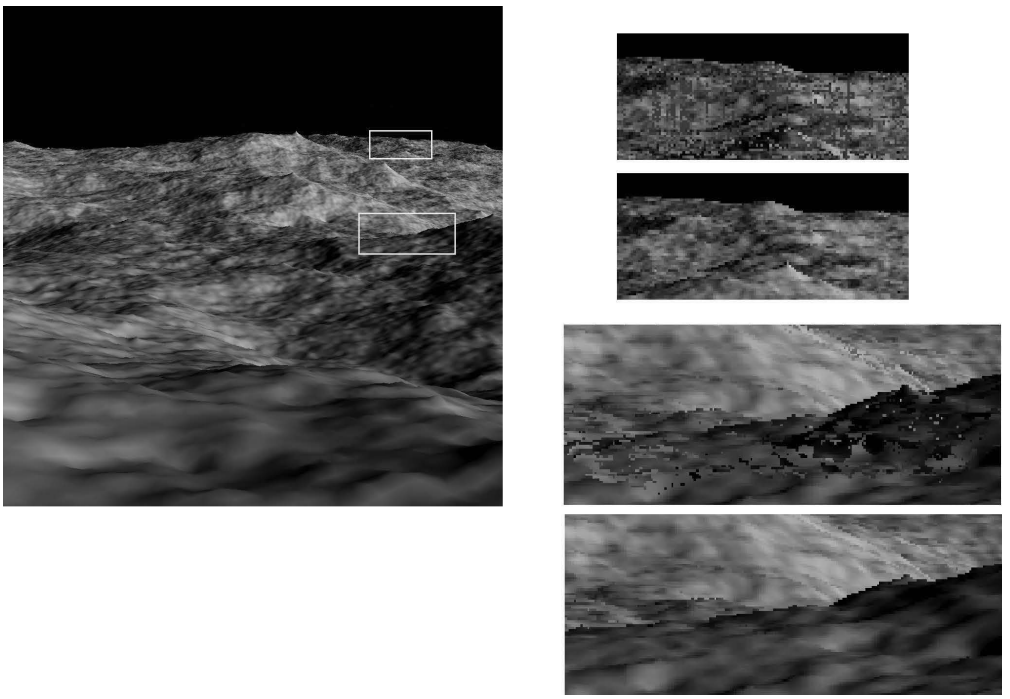


Figure 6. Image artefacts due to limited depth resolution. The image on the left contains both very close and very distant terrain. The two highlighted areas are shown enlarged on the right; the upper image of each pair has been rendered using a single depth pass, leading to grainy artefacts due to the poor depth resolution. The lower image of each pair has been rendered using four depth passes, allowing much greater resolution and eliminating the artefacts.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

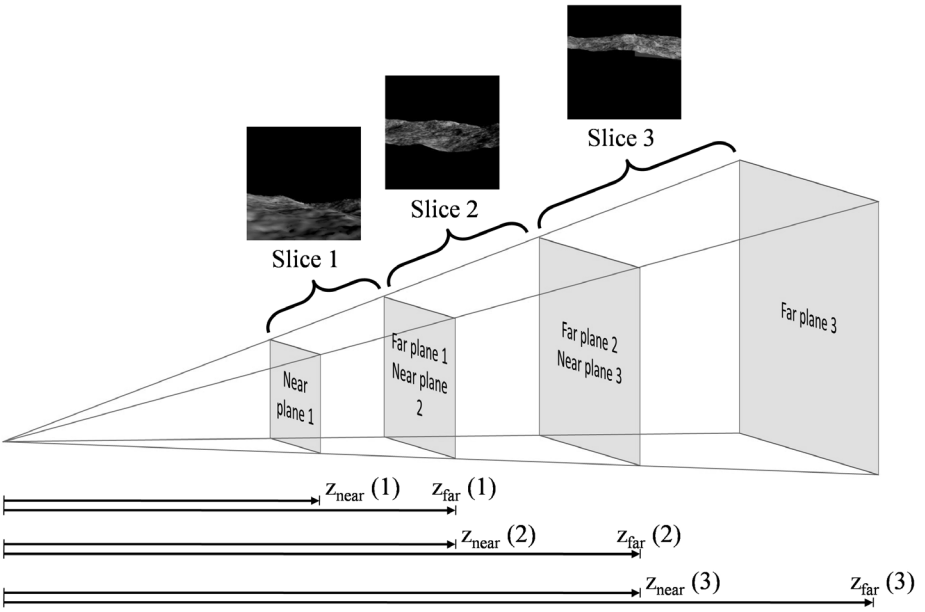


Figure 7. Positioning of near and far planes when rendering an image in multiple depth passes. Depth slices are rendered in reverse order of distance (slice 3 – 1 in this example). The images show the corresponding depth slices when Fig. 6 is rendered using 3 slices. The positions of each plane are calculated so that the ratio z_{far}/z_{near} is constant. A small overlap is added so that no gap is visible between slices in the final image.

It is not feasible to model all of these effects, because many depend closely on the optoelectronic properties of specific camera and detector hardware, which is generally not known during the early GNC testing and development stages. It is also outwith the scope of early testing, which focuses more on GNC algorithms proof-of-concept than performance in real, noisy hardware, which is the focus of later field trials. From the point of view of supporting early testing of GNC systems, the major optical parameters that we would wish to adjust are the camera field of view and detector size in pixels. These adjust the image resolution and degree of perspective distortion, which have a direct impact on the performance of vision guidance. Being able to test the sensitivity to these parameters at an early stage would also help to drive the selection of appropriate hardware. Our image simulator therefore allows these parameters to be specified separately for each simulated camera. Note also that there is no concept of exposure time: the polygon shading method that we use produces pixels with a level that represents more the instantaneous intensity of received light (though the images are not photometrically calibrated), rather than the quantity of photoelectrons in the detector.

3.3 Depth resolution

When multiple objects or sections of terrain overlap each other along the line of sight, the rendering system must decide which polygons are visible and which are occluded by foreground objects. The standard solution known as depth culling is to render the scene simultaneously to a depth buffer on the GPU, where the depth of each pixel is recorded. As each successive polygon passes through the rendering pipeline, its depth (distance from the near plane) is tested against the value

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

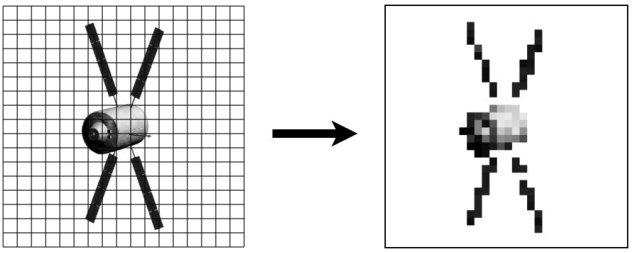


Figure 8. For distant views of spacecraft, the target may subtend only a few pixels (left). When many polygons fall in the same pixel, the default rendering pipeline simply uses the last one to pass through the pipeline to determine the colour of the pixel. The final image exhibits significant aliasing (right) and loss of colour resolution.

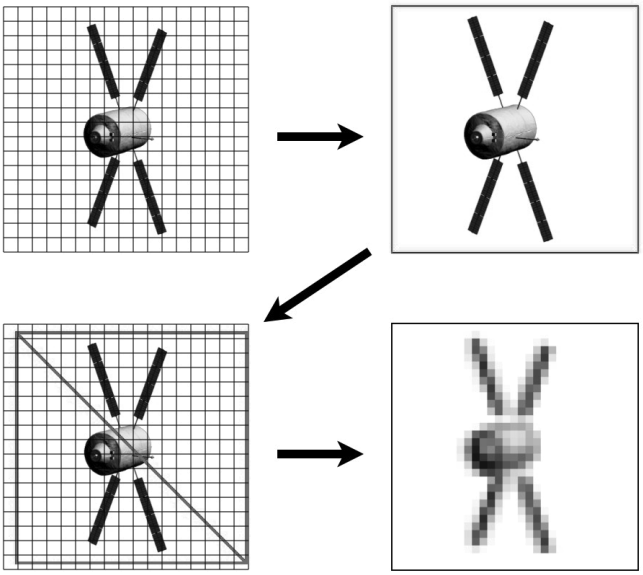


Figure 9. When rendering long range views of spacecraft, an intermediate image of the spacecraft is first rendered at high resolution (top right). This 'imposter' image is then drawn onto a large polygon quad (red triangles, bottom left) and this is painted into the final image (bottom right). By using imposter images, long range views of spacecraft can be rendered free from aliasing.

currently stored in the corresponding depth buffer pixel. If it is smaller (closer to the camera), then it replaces the value in the depth buffer and is also rendered to the main frame buffer. If it is larger then it is discarded. Once rendering is complete, the frame buffer will contain pixels corresponding to the closest object along each line of sight.

Due to the limited precision of the depth buffer pixels (which normally have 16 or 24 bits), stored depth values are quantised (see Fig. 5). This means that for sets of overlapping polygons of similar distance, the rendering system may be unable to determine which is the closest, and the pixel eventually drawn to the frame buffer will simply be the last to pass through the pipeline. This is sometimes referred to as *z*-fighting, and leads to grainy artefacts in the image that are worse at larger distances from the camera due to the way 3D coordinates are transformed into the projective

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

frame of the camera prior to depth culling. Figure 6 demonstrates this effect in a simulated Lunar scene with large depth variation.

We can alleviate this problem by dividing the view volume into several distinct ranges of depth and rendering the scene in several passes, where in each pass only one of the depth 'slices' is rendered – see Fig. 7. The depth slices are chosen to maximise the depth precision within each slice and keep artefacts to a minimum. This is done by setting the near and far plane distances for each depth slice so that the ratio z_{far}/z_{near} is below a constant threshold. This ratio roughly determines the degree of depth quantisation, and by avoiding large values we can keep artefacts to a minimum within each slice.

The slices are rendered from farthest to nearest, so that objects in slices closer to the camera obscure those in more distant slices. The lower panel of each pair on the right of Fig. 6 shows the same image rendered using several depth passes, which eliminates the artefacts present in the single-pass image.

3.4 Long range views of spacecraft

One of the main difficulties in rendering mesh-based objects is matching the resolution of the mesh to that of the pixels in the final image. No polygon in the mesh should be less than one pixel in size otherwise aliasing may occur. In the context of rasterising systems like OpenGL (as used by PANGU), only one polygon of the mesh can determine the colour of an individual pixel. If more than one polygon falls inside the same pixel, then one of the polygons will be effectively selected at random to determine the pixel colour (see Fig. 8). The result is that distant spacecraft models will flicker as they move.

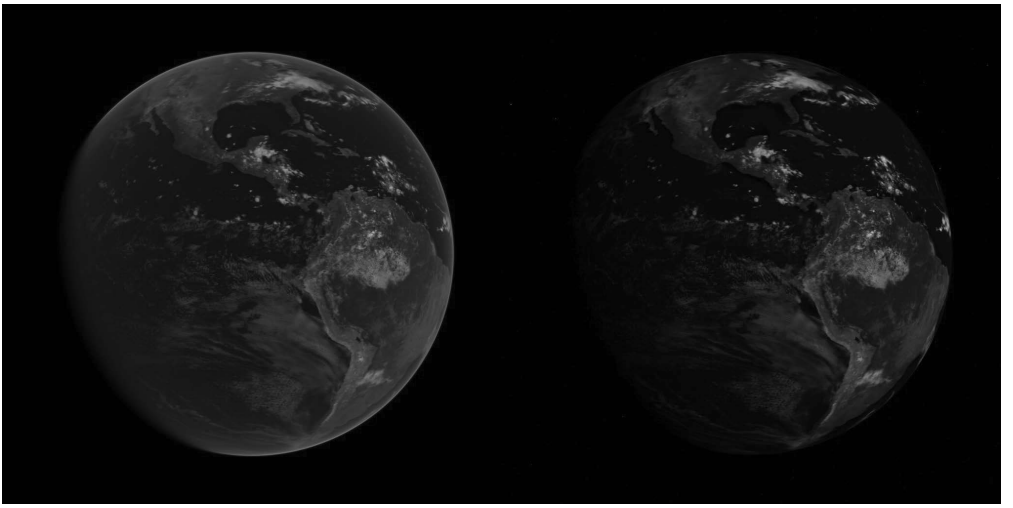
To get around this, we use the technique of 'imposter' images described briefly in Section 3.1.3. An image of the spacecraft is taken from the same camera position as the original, but swivelled and zoomed in to produce a much higher resolution image where the spacecraft fills the frame. This imposter image is drawn onto a rectangular polygon quad, and this quad is then rendered into the original image at an appropriate size – see Fig. 9. The larger polygons of the imposter quad cover multiple pixels, so the problems of aliasing and loss of colour resolution are avoided. Also, when drawing textures into a frame, OpenGL interpolates colours between pixels, providing an additional stage of filtering and resulting in very smooth images.

As mentioned previously, this technique is normally used to spread the work of rendering a model over multiple frames: the same imposter is used in several images, until the position and orientation of the original model has changed significantly and the imposter is redrawn. In our work, we cannot tolerate the systematic errors in the appearance of an object that are introduced by this optimisation, and we render a new imposter with every frame. In implementing this scheme, care has been taken to ensure that imposters are correctly handled when multiple targets overlap, and when targets include mirrored and/or shadowed surfaces.

3.5 Planetary atmosphere modelling

For several of our target scenarios, modelling of atmospheric attenuation effects is crucial for simulating highly realistic images. This includes in-orbit rendezvous and other types of satellite mission, where the Earth (or other large planet) appears in the background of navigation imagery and the surface is viewed through the full depth of the atmosphere. It also includes rover and lander missions, where images are captured inside the atmosphere at varying levels and the appearance of both the sky and distant objects are significantly affected by attenuation. Such environments include, for example, the surfaces of Mars and Titan[33]. The importance of including the effect

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) atmosphere (b) no atmosphere

Figure 10. The presence of the atmosphere has a large effect on the appearance of Earth from orbit, and must be considered in order to simulate realistic images.

of the Earth's atmosphere in simulating orbital imagery is evident from Fig. 10, which shows the same Earth model rendered with and without the atmosphere.

An established computer graphics technique for simulating the loss of contrast of distant objects due to light extinction is that of distance fog, where the colours of objects are blended with a uniform fog colour according to their distance from the camera. However, in real atmospheres consisting of gas and dust particles, the attenuation shows a dependence on wavelength, viewing direction and distance, as well as the scattering properties of individual particles. As such, distance fogging is not suitable for reproducing planetary atmospheric effects with sufficient realism.

We have therefore implemented the atmosphere models of Refs 34 and 35. These consider the absorption of light from the incoming ray by particles along a given viewing direction, as well as the scattering of light both into and out of the incoming ray. The full model includes Rayleigh and Mie scattering components from gas and dust particles respectively. We have adopted the Henyey-Greenstein approximation[36] for the angular dependence of the Mie scattering process, which has a single parameter that varies the scattering probability between isotropic and predominantly in the forward/backward directions. The exponential scale height for each type of atmospheric constituent can be configured separately, and we also permit a uniform density model to simplify the simulation when this assumption is appropriate, such as rover and ground operations. The optical depth along the zenith direction can be specified for each colour channel separately, and is key to reproducing the wavelength dependence of attenuation.

We consider three modes of attenuation: that of light arriving at ground level after passing through the atmosphere, that of light arriving at the camera after reflection from the ground (the so-called 'aerial perspective'), and that of light arriving at the camera directly after passing through the atmosphere (the 'blue sky' effect on Earth). The relative importance of each mode depends on the scenario; for example, in rover simulations the sky rendering is important because much of the scene will be composed of clear sky. Figure 11 shows a simulated image of Mars taken at ground level, demonstrating both the loss of contrast of distant objects and the sky colouring arising from the simulated atmosphere. To improve the rendering speed, the scattering integrals

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



Figure 11. Simulated image of Mars with the atmosphere model rendered at ground level, showing the aerial perspective effect where distant objects lose contrast.

are partially pre-computed according to Ref. 37, and the remaining numerical integration required for each pixel is performed by a dedicated GPU shader function at runtime.

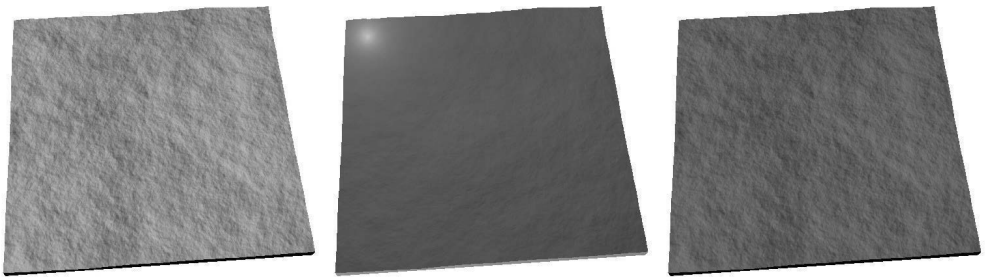
3.6 Realistic reflectance functions

In order to produce realistic space imagery, it is necessary to simulate the appearance of quite a wide range of different materials, both natural and manmade. In computer graphics, this is achieved by adjusting the way that light reflects from a surface. The reflectance properties of many surfaces can be split into two components: the specular reflectance from the surface layers (mostly in the forward direction) and the diffuse reflectance from deeper layers (mostly isotropic).

The surfaces of many Solar system bodies are covered with a layer of fine regolith, which shows a flat diffuse reflectance with a significant backscattering component called the 'opposition surge'[38], an effect that is particularly strong on the Moon. Much of the surface of Mars is covered with sand, which from a reflectance point of view consists of many microscopic reflectors that cannot be individually included as physical objects in the surface mesh, and the bulk reflectance must be modelled in a phenomenological way. Spacecraft components are made from a wide range of artificial materials including highly specular solar panels and metallic multi-layer insulation.

The spatial reflectance properties of a material are typically modelled using a BRDF, which defines the reflectance in terms of the directions of incident and reflected light. Hardware rendering APIs often include a basic set of BRDFs as part of the fixed-function rendering pipeline. In the case of

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

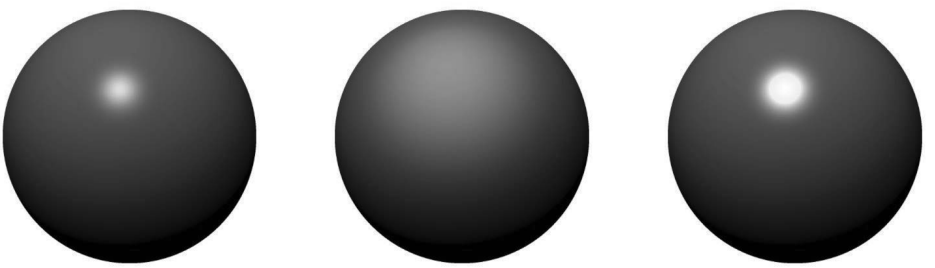


(a) Lambert (b) Hapke (c) Oren-Nayar

Figure 12. Surface rendered with different diffuse BRDFs: standard isotropic Lambert model (a); Hapke model with strong backscattering component at top left (b); Oren-Nayar model (c).

PANGU, which uses OpenGL, this basic set contains the isotropic Lambert BRDF for perfectly diffuse reflectors, and the Blinn-Phong BRDF[39] for highly specular (mirror-like) reflectors. In terms of the materials discussed above, these BRDFs are rather limited: the Lambert law is a poor approximation to the reflectance of regolith and sand, and the Blinn-Phong model (in which the specular highlight is just a cosine distribution about the angle of reflection) is too restrictive. It is possible to override the fixed-function rendering pipeline and introduce alternative BRDFs. This is done by writing fragment shaders (see Section 2.2) that implement the reflectance equations at the appropriate stage in the rendering pipeline.

We have implemented the Hapke quasi-empirical diffuse BRDF[40] that models the reflectance of airless regolith. For rough surfaces such as sand and cloth, we have implemented the Oren-Nayar BRDF[41]. This is a generalised form of the Lambert BRDF for surfaces composed of many microscopic facets that are individually Lambertian. The bulk reflectance is non-Lambertian, the main difference being that light is no longer reflected isotropically but preferentially back along the illumination direction. This model is parameterised in terms of a single 'roughness' parameter, which quantifies the width of the (Gaussian) slope distribution of the microfacets. In Fig. 12 these functions are used to render an artificial surface model, to demonstrate the difference between the effects achievable with the fixed function and programmable shaders.



(a) Blinn-Phong (b) Cook-Torrance rough (c) Cook-Torrance smooth

Figure 13. Gold sphere rendered with a combination of the Lambert diffuse BRDF and the following specular BRDFs: (12) standard Blinn-Phong; (12) Cook-Torrance function with high roughness; (12) Cook-Torrance function with low roughness.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

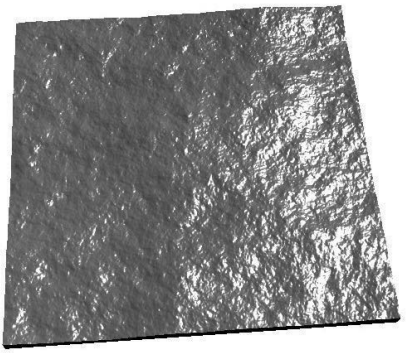


Figure 14. Example of combining diffuse and specular reflectance functions: the Oren-Nayar diffuse BRDF with Cook-Torrance specular highlights can simulate shiny foil material.

For more accurate specular reflectance, we have implemented the Cook-Torrance BRDF[42], which is applicable to rough surfaces composed of microfacets, where each facet is individually a perfect specular reflector. This is an appropriate model for a wide range of metallic and plastic materials. This model considers reflectance by individual facets, the slope distribution of the ensemble of facets and the shadowing and masking that occurs between neighbouring facets. The model is parameterised in terms of the root-mean-square of the facet slope distribution (the roughness), and we allow surfaces to be a linear combination of up to two different roughnesses. Figure 13 compares the Blinn-Phong and Cook-Torrance reflectance functions.

By selecting different combinations of diffuse and specular reflectance functions, a much richer range of effects can be achieved. For example, Fig. 14 uses a combination of the Oren-Nayar and Cook-Torrance functions to simulate shiny foil material.

4.0 TEST ENVIRONMENT SUPPORT

In software testing of a complete vision-based guidance, navigation and control platform, image simulation is only part of the picture. A major requirement is a model of the physical and dynamical environment through which the spacecraft moves. This allows control actions and external forces to be resolved into accelerations, and the equations of motion of the spacecraft to be solved. As the spacecraft moves, simulated sensor readings and images provide feedback to the guidance system, closing the loop for further control decisions based on the evolving GNC estimate of the spacecraft trajectory. Performance analysis is done by comparing, for example, the estimated trajectory with the ground truth, although in vision systems other metrics may be used, such as the estimated terrain elevation or triangulated location of tracked feature points.

4.1 Server mode for closed loop testing

In order to support the testing and validation of vision based GNC systems, we have built a server mode into our image generation software that allows images to be requested over a TCP/IP interface. In closed loop simulations, the actual spacecraft trajectory is not known in advance and images must be generated on-the-fly as the spacecraft moves. The server mode enables all of the models and rendering state to be loaded in advance, so that when an image is requested the only latency arises from the time taken to render the image. Clients communicate with the image server over a socket connection, allowing the image generation software to run on a separate machine to the

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

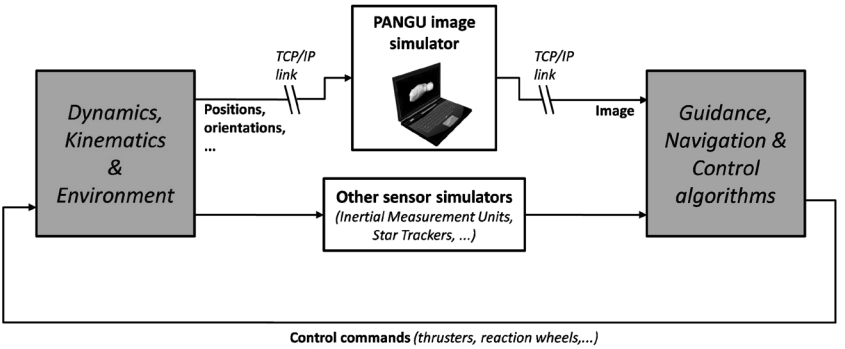


Figure15. High level architecture of a typical end-to-end mission simulator that uses PANGU as a camera module. See the text for a description of each block. In this example, the simulator communicates with PANGU over a TCP/IP link such as an Ethernet connection. This allows PANGU to run in a separate computer, enabling faster image generation.

rest of the functional end-to-end simulator. Many parameters can be set dynamically over the network as the simulation evolves, such as the positions and orientations of all models and the camera, the positions of joints in articulated models (e.g. rover wheels or extending solar panels) and the positions of light sources. Clients can then request an image of the current view, which is rendered and transmitted back across the network.

Figure 15 depicts the high level architecture of a typical end-to-end mission simulator incorporating our image simulator (PANGU) as a camera module. On the left, the Dynamics, Kinematics and Environment (DKE) block represents the 'real world': the true states of all bodies and the various environmental phenomena such as gravity and radiation pressure. In the middle, sensor simulators are used to model instrument readings given the true state of the spacecraft. On the right, the GNC algorithms operate on the simulated sensor data and determine appropriate control commands for the thrusters and other actuators. These commands are fed back to the DKE block, which resolves them into forces and solves the equations of motion for the spacecraft in a closed loop. The simulator communicates with PANGU over a TCP/IP link such as an Ethernet connection, allowing PANGU to run on a separate computer for faster image generation times.

4.2 Vision guidance performance metrics

Vision guidance systems may use images to measure the shape of the environment through which they are navigating, as well as their trajectory. For example, the GNC may use optical feature tracking[43] combined with SLAM and machine vision techniques[10,32] to estimate the 3D locations of tracked points, and build a map of the terrain represented as a cloud of points. This technique may be useful for both asteroid pinpoint landing, and rendezvous and docking with uncooperative target satellites[11,12]. Other systems designed for landing on large flat terrain regions can use tracked points to maintain an estimate of the ground plane orientation[44]. Another popular technique is shape-from-shading ?[45], where the slope of the terrain at each pixel is inferred from the intensity of reflected light, and the shape of the entire surface can be inferred by applying constraints such as the elevation being continuous from pixel to pixel. Combinations of these techniques come under the broad heading of multiview reconstruction, and are now used widely in space exploration, perhaps most prominently in mapping small bodies[46] and in autonomous rover navigation[47].

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

On-the-fly image generation allows us to provide, along with the images, ground truth data for evaluating the performance of such GNC systems. Clients can request the 3D coordinates of points in the image, for comparison to estimates produced by the GNC. These are obtained by projecting a ray along the line of sight towards the point, and finding the 3D location where this intersects the surface. Clients can also request their instantaneous elevation, for scenarios such as landing on large planetary surfaces, and range and slope maps for each pixel in the image for comparison to results from shape-from-shading or ground plane and slope estimation.

4.3 Integration with engineering simulators

In terms of the European Cooperation on Space Standardisation (ECSS), our virtual image simulator is designed to be integrated with a range of mission simulation tools including functional engineering simulators and functional validation testbenches[48]. These are major steps in the development of a complete mission simulation, and are used to verify critical elements of system design, trade off different GNC algorithms, and support the analysis of system performance.

In order to ease integration of our image generator with a range of different modelling environments, we provide a library of standard functions for communicating with the image server. This is implemented in both C and Java, allowing integration with a wide range of third-party simulation tools such as Simulink® and dSPACE®.

To give a concrete example, Fig. 16 depicts the use of our image simulator in a Simulink model, where a selection of client library functions has been wrapped in a Java S-function. This is a discrete functional block in a larger system model, and in this case would represent a navigation camera subsystem.

During the simulation, the position and attitude of the camera and a range of objects are passed into the S-function, which calls client library functions to send these on to the image server. The S-function then requests an image from the server and passes this on via the output ports for use elsewhere within the simulation. The specific set of input/outputs can be changed by updating the S-function to call different client library functions.

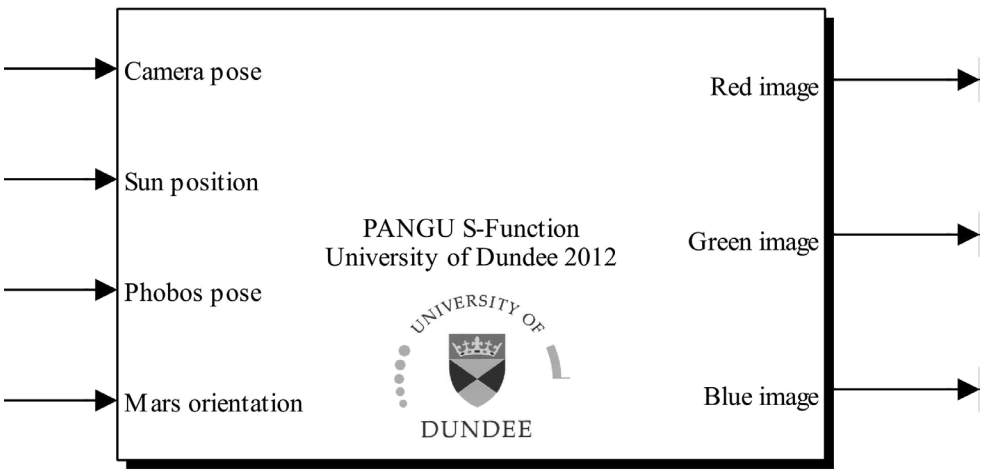


Figure 16. PANGU image simulation can be introduced as a modular block in a variety of system modelling tools, for example as a Simulink® S-function in this image. This allows easy integration with such mission analysis tools as functional engineering simulators and functional validation testbenches.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) Mars Express VMC image
Images courtesy of ESA

(b) PANGU reconstruction

Figure 17. Image 17 was captured by Mars Express VMC on the 2010/01/30 at 23:59:32, at an altitude of 9578·53 km and on orbit number 7794. Image 17 is a PANGU reconstruction generated using the field of view and image size of the VMC, with the positions and orientations of all relevant solar system bodies obtained from the Mars Express SPICE data. The shadow of Phobos appears as a black dot in the lower left region of the illuminated portion of Mars.

5.0 RESULTS

In this Section we present a variety of results that demonstrate the performances of PANGU in a range of scenarios. Appropriate quantitative performance metrics that test the geometry of PANGU images are hard to derive: in this analysis, we will focus on visual comparisons between real images and PANGU reconstructions derived from high accuracy flight dynamics data. Image generation times will be provided to demonstrate the feasibility of using PANGU as a camera simulator in real-time or faster-than-real-time mission simulators. In addition, we report on a couple of European Space Agency projects that incorporated our virtual image simulator in their mission simulation tools.

5.1 Comparison to real images

Many space exploration missions take images of their target bodies. If appropriate shape models of the targets and sufficient flight dynamics data are available, these can be used to recreate the images in PANGU. Fortunately, most missions record this data and make it available to the public through the Spacecraft, Planets, Instruments, C-matrix and Events (SPICE) system. This system aims to provide a common format for the archiving and distribution of space science data, including flight dynamics records. Note that direct pixel-by-pixel comparison between real images and PANGU reconstructions is not appropriate, because although PANGU simulates perfect images that match the idealised projection model of a given optical sensor, it does not model the exposure time and various sources of noise and optical distortions that are present in real images.

5.1.1 Mars express visual monitoring camera

The Visual Monitoring Camera (VMC) onboard Mars Express is used to take wide-angle colour images of Mars, which are then released to the public[49]. Corresponding SPICE data for Mars Express is available[50], which allows the relative positions and orientations of Mars, the VMC and the sun to be obtained at the instant each image was captured. Figure 17 presents an original

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) Hayabusa AMICA image
Images courtesy of JAXA

(b) PANGU reconstruction

Figure 18. Image 18(a) was captured by the Hayabusa AMICA instrument on 2005/10/16 at 14:30:13 in the V band. Image 18(b) is a PANGU reconstruction generated using the field of view and image size of the AMICA instrument, with the positions and orientations of Itokawa, Hayabusa and the sun obtained from the Hayabusa SPICE data.

VMC image captured on 2010/01/30 at 23:59:32 UTC, together with a PANGU reconstruction. The reconstruction was generated using the properties of the VMC camera to set the field of view and image size, and the Mars Express SPICE data to position the relevant bodies correctly. In this image, Phobos passed between the sun and Mars; the shadow is just visible in the lower left region of the illuminated portion of Mars in both images. In the PANGU image, the Mars 3D model is produced by wrapping a Mars surface image, derived from the Mars Global Surveyor image catalog MOC2-143, onto a spherical mesh with 51681 vertices. The PANGU atmosphere model (Section 3.5) with a dust component is used to reproduce the scattering of light in the Martian sky.

5.1.2 Hayabusa AMICA instrument

The Hayabusa spacecraft arrived at asteroid 25143 Itokawa in September 2005, surveying the surface for several months before touching down in November 2005 to collect a sample of material for return to Earth. Images taken with the Asteroid Multiband Imaging Camera (AMICA) were used, in conjunction with other data, to measure the shape and albedo of the asteroid. All of the raw science data, shape model and SPICE data are available through the Hayabusa Project Science Data Archive[51], enabling reconstruction of AMICA images with PANGU. Figure 18 presents an original AMICA V band image taken on 2005/10/16 at 14:30:13 UTC, alongside a PANGU reconstruction. The reconstruction uses the same image size and field of view as the AMICA instrument, and uses the Hayabusa SPICE data to position the asteroid, sun and camera. The shape of the asteroid was obtained from the highest resolution model available, with 1·57 million vertices. In this image, a 70%/30% combination of the Hapke and Lambertian BRDFs are used (see Section 3.6) to render the asteroid. This was found to better reproduce the appearance of the asteroid than either BRDF individually. In addition, an imposter image (section 3.4) is used to render the asteroid. This is helpful to reduce aliasing effects that occur along the sharp outline of the asteroid against the dark sky.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

5.1.3 Frame rates

The reconstructed VMC image takes 99ms to generate on an Intel® dual core CPU running at 2·53GHz, and with an NVIDIA Quadro NVS 160M graphics card. This includes network overheads associated with running PANGU in server mode, and is representative of the frame rate achievable when using PANGU as a camera module within an end-to-end mission simulator. A VMC-based vision guidance system that cycles at 1Hz could therefore potentially be tested in simulation up to ten times faster than real time on this hardware. On the same hardware, the reconstructed AMICA image takes 580ms to generate. The larger generation time is mainly due to the large number of vertices, larger image size, and the use of the imposter image which requires an extra stage of rendering. For example, when switching to the next lowest resolution model with $3{\cdot}6 \times 10^5$ vertices, the image generation time reduces to 229ms. When the imposter image is disabled, the time reduces to 321ms.

In general, different applications will enable and disable different functions of PANGU in an attempt to balance the opposing demands of realism and rendering time, in order to meet the requirements of the particular simulation.

5.2 Examples of use in vision guidance projects

PANGU has been used in a variety of ESA vision based GNC development projects, covering a range of different mission scenarios that require simulated images tailored to their specific requirements. In this Section, we provide short summaries of a selection of these projects, explaining how the virtual image simulator was used to support closed loop simulation, testing and validation of the GNC systems.

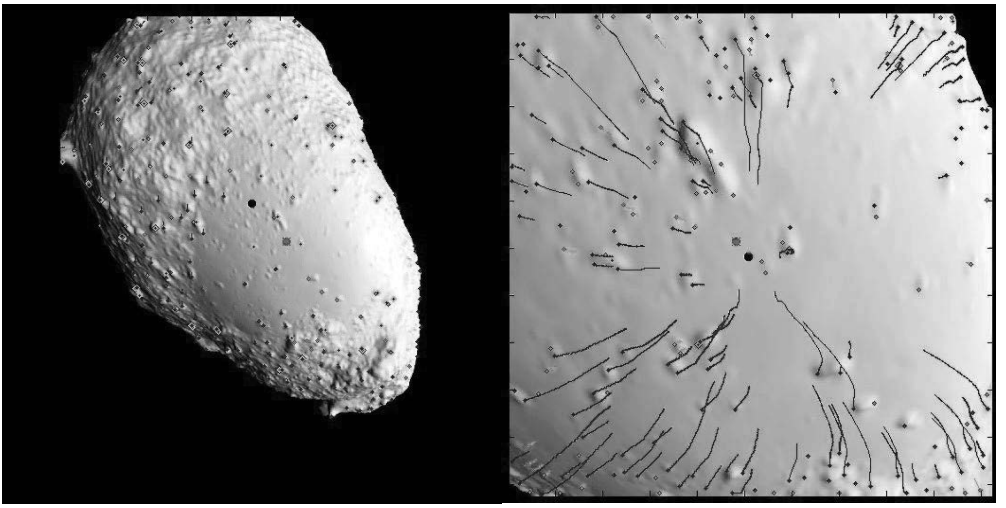
5.2.1 NEOGNC: Autonomous GNC technologies for NEO proximity, landing and sampling operations

The Marco Polo mission (now superceded by Marco Polo-R), part of ESA's Cosmic Vision programme, was a mission concept to land on a small, primitive near-Earth asteroid (NEO), and return a sample of material to Earth for laboratory analysis. Landing on such an object presents considerable difficulties to traditional GNC techniques: the gravity is so weak that Keplerian orbits suffer large perturbations due to the radiation pressure and gravity from the Sun, the rotation period may be short (~ hours) and the landing ellipse is very small (~ metres). In this context, the project NEOGNC[1] was carried out by a consortium led by GMV under ESA contract, with the aim of developing an autonomous vision-based GNC system for asteroid proximity operations, including descent and pinpoint landing[52].

Several reference mission scenarios were identified including asteroid Itokawa, and PANGU was incorporated into the project's high fidelity functional engineering simulator via an S-Function (see Section 4.3) in order to provide imagery within the closed loop simulation environment. A Monte Carlo simulation campaign verified that, in nominal conditions, the proposed GNC algorithms met the landing accuracy requirements of 3·5m at 3σ, and a touchdown horizontal velocity of several centimetres per second[53]. Two processed images from the simulation campaign are shown in Fig. 19. Figure 18 is taken at the start of the terminal descent and landing phase (the 'low gate'), where the spacecraft is 500 metres above the ground, and figure 18 is taken from an altitude of 100 metres during the descent. In these images, the blue and green lines indicate the motion of tracked surface features, and are one of the measurements used to feed the optical navigation filter. The small blue, red and green points mark the present location of each feature. The black

1 ESA Contract No. 23048/09/NL/AF

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) Start of terminal descent; 500m altitude (b) 100m altitude

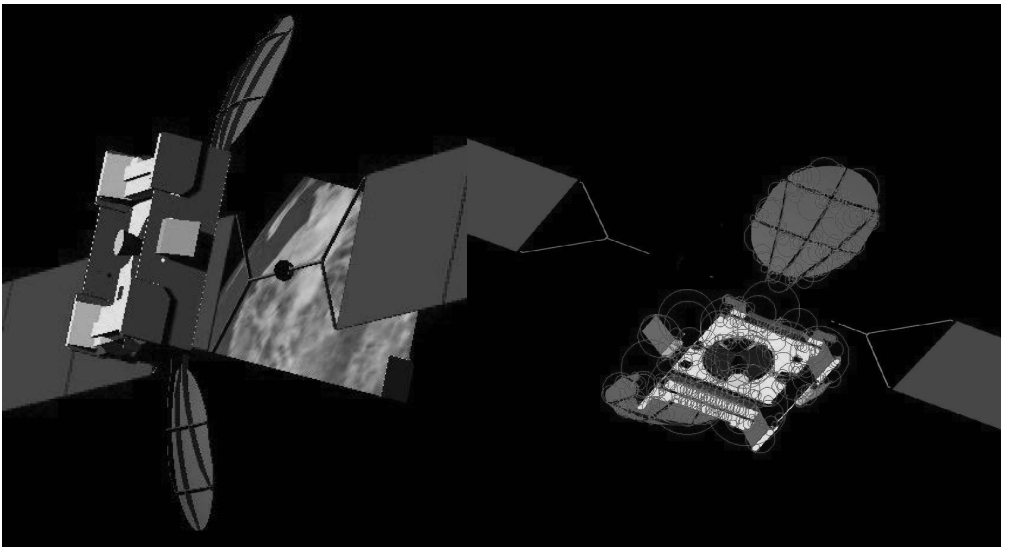
Figure 19. Processed images from the NEOGNC Functional Engineering Simulator of the descent and landing phase onto the asteroid Itokawa. PANGU was used to generate the raw images within the closed loop simulation environment. The images have been processed using optical feature tracking and landmark recognition algorithms; see the main text for a description of the features marked in each image.

mark indicates the centre of brightness (another measurement estimated by the image processing algorithm) and the red mark indicates the target landing site. Light blue diamonds mark the sites of 'known landmarks': these are surface features of known position in the asteroid body-fixed reference frame, mapped out during a previous stage of asteroid observation and characterisation. They are detected in the images during descent and landing by applying automatic landmark recognition algorithms. Known landmarks are important navigation points as observations of them in the images allows the absolute position and orientation of the spacecraft to be determined in the asteroid body-fixed frame.

5.2.2 HARVD: High integrity, autonomous rendezvous and docking

A key enabling technology for future sample return and in-orbit servicing missions is the ability to perform autonomous rendezvous with a co-operative or non-cooperative target. The High integrity, Autonomous, multi-range RendezVous and Docking project (HARVD)[54] developed a vision-based GNC system using an optical camera that could perform autonomous rendezvous in a wide range of scenarios, including capture of a passive target for sample return applications (where a canister is placed into orbit for another spacecraft to collect), and docking between a servicing satellite and uncooperative target (for example to boost the orbit of a fuel-depleted satellite and extend its operational lifetime). PANGU was incorporated into the HARVD functional engineering simulator to provide synthetic images of the target satellite, in order to validate the performance of the GNC system and in particular to develop a 3D pose estimation algorithm that could estimate the attitude of the target from the images alone. The benefit of synthetic imagery in this context is that the true pose of the target is known precisely. Figure 20 shows a couple of simulated images from the project. The image on the left demonstrates some image generation effects that were necessary for HARVD, including reflective OSR (optical solar reflector) tiles and self-shadowing by the target spacecraft[27]. The mirror effect is achieved in the following manner: first, the reflected image is generated by rendering the main scene from a virtual point

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press



(a) Self-shadowing and reflections (b) Pose estimation

Figure 20. Two simulated images from the HARVD project. Image 20(a) demonstrates some features required for this project – mirrored surfaces to represent OSR tiles and shadows to reproduce self-shadowing effects between components of the target spacecraft. The processed image 20(b) is from the pose estimation trials. The red circles indicate SURF interest points, which are used to link image points to known locations on the target body, enabling attitude determination.

obtained by reflecting the main camera across the mirror plane. This image is then drawn onto the mirror in the main image during an additional rendering pass. On the right, we show a processed image from the pose determination tests. The red circles indicate SURF interest points[55] which are salient features used to match points in the image to known locations on the target body. This allows the relative attitude and position of the target to be determined from the images alone, using techniques of pose estimation.

6.0 CONCLUSIONS

In this paper, we have presented an image generation environment designed for testing vision based GNC systems for space applications. We have emphasised in particular the benefits, limitations and pitfalls of various standard CGI techniques, and outlined how these can be used and adapted to increase realism in simulated space environments. We have also discussed how such an image generation tool can be integrated within the standard framework of mission simulation tools, and included a few examples of its use in European Space Agency R&D projects.

Development of our image generation techniques is an ongoing process, as graphics technology improves and mission scenarios evolve. In particular, simulations for high-velocity planetary landing scenarios require frame rates in excess of 20Hz. Our image simulator is not yet capable of meeting this rate with sufficient realism, and this is a major goal of future development. Some other potential areas for work include incorporating realistic models of sensor noise and optical distortions, generalising the material reflectance functions further, building a library of reflectance parameters for standard satellite construction materials, and implementing high dynamic range imaging for very high contrast scenes.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

ACKNOWLEDGMENTS

The authors would like to thank the ESA technical officers Stein Strandmoe, Salvatore Mancuso, Christian Philippe, Phillip Harwood and Olivier Dubois-Matra, who oversaw various PANGU development contracts funded by ESA. These include 'LunarSim' (no. 12821/98/NL/MV), 'PANGU' (no. 11747/95/NL/JG), 'Whole Planet Simulation with PANGU' (no. 17338/03/NL/LvH/bj) and 'Virtual Spacecraft Image Generator Tool' (no. RFQ/3-12351/08/NL/MP). PANGU is available under license from ESA for use on ESA-funded projects, and is available from STAR-Dundee Ltd for other applications.

REFERENCES

1. BONNET, R.-M. and SWINGS, J.-P. The Aurora Programme, February 2004, volume BR-214, ESA Publications Division, .
2. National Aeronautics and Space Administration (NASA) Solar System Exploration, September 2006.
3. CHENG, Y., GOGUEN, J., JOHNSON, A., LEGER, C., MATTHIES, L., MARTIN, M.S. and WILLSON, R. May 200419(3), 13–21.
4. OWEN JR., W.M., WANG, T.C., HARCH, A., BELL, M. and PETERSON, C. 30 July 2001 In AAS/AIAA Astrodynamics Specialist Conference Quebec, Canada.
5. KUBOTA, T., HASHIMOTO, T., SAWAI, S., KAWAGUCHI, J., NINOMIYA, K., UO, M. and BABA, K., *Acta-Astronautica*, January 2003, **52**, (2), pp 125–131.
6. CARSTEN, J., RANKIN, A., FERGUSON, D. and STENTZ, A. March 3–10 2007 In Aerospace Conference Pro- ceedings 2007, IEEE Big Sky, MT, pp. 1–11.
7. PERSSON, S., JACOBSSON, B. and GILL, E. October 17–21 2005 In Proceedings of the 56th International Astronautical Congress Fukuoka, Japan.
8. PERSSON, S., D'AMICO, S. and HARR, J. Sep. 27– Oct. 1 2010 In Proceedings of the 61st International Astronautical Congress Prague, Czech Republic.
9. FLANDIN, G., POLLE, B., LHERITIER, J. and VIDAL, P. In NASA/ESA Conference on Adaptive Hardware and Systems Anaheim California, 15–18 June 2010, USA, pp 9–16.
10. DURRANT-WHYTE, H. and BAILEY, T. June 2006, **13**, (2), pp 99–110.
11. SCHNITZER, F., SONNENBURG, A., JANSCHEK, K. and WILLICH, G. In Proceedings of the 8th International ESA Conference on Guidance and Navigation Control Systems Carlsbad, 5 - 10 June 2011, Czech Republic:
12. COCAUD, C. and KUBOTA, T. August 29 – Sep. 1 2010 In Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Sapporo, Japan.
13. JOHNSON, A.E., CHENG, Y. and MATTHIES, L.H. In Aerospace Conference Proceedings 2000, 18–25 March 2000, IEEE, **7**, Big Sky, MT. pp 661–671.
14. WILLSON, R., JOHNSON, A. and GOGUEN, J. In Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS) volume SP-603, Munich, Germany.
15. JOHNSON, A., WILLSON, R., GOGUEN, J., ALEXANDER, J. and MELLER, D. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA), 18–22 April 2005, pp 4463–4469.
16. GUIZZO, G.P., DRAI, R., DESPR`E, N. and JURADO, M.M. In Proceedings of the 8th Inter- national ESA Conference on Guidance, Navigation and Control Systems, Karlovy Vary, 5-10 June 2011, Czech Republic:
17. CABALLERO, F., MARTIN-ROMERO, J. R., MOLLINEDO, L. and SUATONI, M. In Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques ESA/ESTEC, 29 May – 1 June 2012, The Netherlands.
18. BOGE, T., BENNINGHOFF, H. and TZSCHICHHOLZ, T. In Proceedings of the 8th International ESA Conference on Guidance, Navigation and Control Systems, 5–10 June 2011, Karlovy Vary, Czech Republic.
19. PAAR, G. and POELZLEITNER,W. of Proceedings of the SPIE Philadelphia, Pennsylvania, USA, October 23–24 1995, **2591**, pp 157–169.
20. KOLESNIK, M. and PAAR, G. *Algorithmic solution and simulation results for vision-based au- tonomous mode of a planetary rover,* **1296**, 1997, Lecture Notes in Computer Science, pp 677–685, Springer Berlin/Heidelberg.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

21. OGRE – Open Source 3D Graphics Engine http://www.ogre3d.org/ Accessed: 15/06/2012.
22. PARKES, S. M., MARTIN, I. and MILNE, I. May 17–21 1999 In Proceedings of the Data Systems in Aerospace Conference (DASIA) Lisbon, Portugal.
23. PARKES, S. M., MARTIN, I. and STRANDMOE, S. May 28– Jun 1 2001 In Proceedings of the Data Systems in Aerospace Conference (DASIA) Nice, France:
24. PARKES, S. M., DUNSTAN, M., MATTHEWS, D., MARTIN, I. and SILVA, V. June 2–6 2003 In Proceedings of the Data Systems in Aerospace Conference (DASIA) Prague, Czech Republic.
25. MARTIN, I., PARKES, S.M., and DUNSTAN, M. 28 June–1 July 2004 In Proceedings of the Data Systems in Aerospace Conference (DASIA) Nice, France.
26. PARKES, S. M., DUNSTAN, M. and MARTIN, I. 30 May–2 June 2005 In Proceedings of the Data Systems in Aerospace Conference (DASIA) Edinburgh, Scotland, UK.
27. MCCRUM, M., DUNSTAN, M. and PARKES, S.M. 2–5 June 2008 In Proceedings of the 7th International ESA Conference on Guidance, Navigation and Control Systems Tralee, Ireland.
28. MCCRUM, M., PARKES, S. M., MARTIN, I. and DUNSTAN, M. In Proceedings of the 10th International Symposium on Artificial Intelligence, 29 Aug–1 Sept 2010, Robotics and Automation in Space, Sapporo, Japan.
29. EBERLY, D.H. 3D Game Engine Design: A Practical Approach to Real-time Computer Graphics, Morgan Kaufmann Publishers Inc., 2000, San Francisco, California, USA.
30. DELOURA, M.A. (Ed) Game Programming Gemsvolume 1, of Game Programming Gems series Charles River Media, 2000.
31. ROST, R.J. *OpenGL(R) Shading Language (2nd ed)*, 2005, Addison-Wesley Professional.
32. Hartley, R.I. and Zisserman, A. *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, 2004, 2nd ed.
33. NIEMANN, H.B., ATREYA, S.K., BAUER, S.J., CARIGNAN, G.R., DEMICK, J.E., FROST, R.L., GAUTIER, D., HABERMAN, J.A., HARPOLD, D.N., HUNTEN, D.M., ISRAEL, G., LUNINE, J. I., KASPRZAK, W. T., OWEN, T. C., PAULKOVICH, M., RAULIN, F., RAAEN, E. and WAY, S.H. *Nature*, December 2005, **438**, pp 779–784.
34. NISHITA, T., SIRAI, T., TADAMURA, K. and NAKAMAE, E. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques SIGGRAPH '93, 1993, New York, NY, USA: ACM. pp 175–182.
35. HOFFMAN, N. and PREETHAM, A.J. In Proceedings of the Games Developer Conference, 2002.
36. HENYEY, L.G. and GREENSTEIN, J.L. *The Astrophysical J*, January 1941, **93**, pp 70–83.
37. O'NEIL, S. *Accurate atmospheric scattering In Matt Pharr*, (Ed), 2005, GPU Gems 2, Addison-Wesley Professional.
38. HAPKE, B. *J Quantitative Spectroscopy and Radiative Transfer*, March 1999, **61**, (5), pp 565–581.
39. BLINN, J. ACM SIGGRAPH Computer Graphics, 1977, pp 192–198.
40. HAPKE, B. *J Geophys Res,* **86(B4)**, 1981, pp 3039–3054.
41. OREN, M. and NAYAR, S.K. In Proceedings of the 21st annual conference on Computer graphicsand interactive techniques SIGGRAPH '94 New York, USA, 1994, NY, USA: ACM. pp 239–246.
42. COOK, R.L. and TORRANCE, K.E. In Proceedings of the 8th annual conference on Computergraphics and interactive techniques SIGGRAPH '81 New York, 1981, NY, USA: ACM. pp 307–316.
43. ROWELL, N., PARKES, S.M., DUNSTAN, M.N., MARTIN, I. and DUBOIS-MATRA, O. In Proceedings of the 8th International ESA Conference on Guidance, 5–10 June 2011, Navigation and Control Systems Karlovy Vary, Czech Republic.
44. JEAN-MARIUS, T. and STRANDMOE, S.E. In Proceedings of the 49th International Astronautical Congress, Melbourne, Australia.
45. WU, T.-P., SUN, J., TANG, C.-K. and SHUM, H.-Y. ACM Trans. Graph, December 2008, **27**, (5), pp 119:1–119:9.
46. GASKELL, R., BARNOUIN-JHA, O., SCHEERES, D., MUKAI, T., HIRATA, N., ABE, S., SAITO, J., ISHIGURO, M.,KUBOTA, T., HASHIMOTO, T., KAWAGUCHI, J., YOSHIKAWA, M., SHIRAKAWA, K. and KOMINATO, T. In Proceedings of the Astrodynamics Specialist Conference Keystone, 21–24 August 2006, Colorado, USA.
47. CARSTEN, J., RANKIN, A., FERGUSON, D. and STENTZ, A.T. In IEEE Aerospace Conference, 2007.
48. ECSS Secretariat Space engineering: System modelling and simulation Technical report ESA-ESTEC Requirements and Standards Division, 2010.
49. GIL-FERN´ANDEZ, J., CADENAS-GORGOJO, R., PRIETO-LLANOS, T., ANSALONE, L. and DRAI, R. In Proceedings of the 8th International ESA Conference on Guidance and Navigation Control Systems Carlsbad, 5–10 June, 2011, Czech Republic.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press

50. GIL-FERN´ANDEZ, J., CADENAS-GORGOJO, R., PRIETO-LLANOS, T., ROWELL, N., DUNSTAN, M., PARKES, S., HOMEISTER, M., SALEHI, S. and AGNOLON, D. 1–5 October 2012, In Proceedings of the 63rd International Astronautical Congress, Naples, Italy.
51. DESPR´E, N., KERAMBRUN, S., POLLE, B., REBOUL-SALZE, F., CROPP, A., SALEHI, S., BASTANTE, J.C., PIZZIGHELLA, M., GOGIBUS, E. and GANET, M. 3–6 May 2010, In Proceedings of the 4th International Conference on Astrodynamics Tools and Techniques ESAC, Madrid, Spain,.
52. BAY, H., ESS, A., TUYTELAARS, T. and VANGOOL, L. Computer Vision and Image Understanding (CVIU), June 2008, **110 , (3)**, pp 346–359.

https://doi.org/10.1017/S000192400000871X Published online by Cambridge University Press