Kovacs *et al. Robomech J* (2020) 7:25
https://doi.org/10.1186/s40648-020-00172-3

ROBOMECH Journal

RESEARCH ARTICLE Open Access

# Trajectory estimation and position correction for hopping robot navigation using monocular camera

Gabor Kovacs*, Yasuharu Kunii, Takao Maeda and Hideki Hashimoto

**Abstract**

In this paper, a navigation and environment mapping method is presented for small exploration robots that use hopping motion. While previous research about hopping rovers mostly focuses on mobility and mechanical design, the motivation for the proposed method is to provide a fully autonomous navigation system using only a monocular camera. The method accurately estimates the hopping distance and reconstruct the 3D environment using Structure from Motion, proving that a monocular system is not only feasible, but accurate and robust at the same time. The relative scale problem of the reconstructed scene and trajectory is solved by the known gravity and parabolic motion constraints. After each hop, the error in landing position is corrected by a modified Iterative Closest Point algorithm with non-overlapping part elimination. The environmental point cloud is projected onto a 2D image, that is used to find the most suitable landing position for the next hop using protrusion based obstacle detection, and navigate the robot towards the goal direction. Both virtual environment simulations and real experiments confirm the feasibility and highlight the advantages of the presented method.

**Keywords:** Hopping robot, Structure from Motion, Navigation, Image processing, Space exploration

## Introduction

In recent years there is extensive research in outdoor, space and field robotics. While traditional wheeled robots continue to be an essential part of space exploration, different approaches to robot mobility are also emerging. Space exploration rovers MINERVA-II1 launched by the Japan Aerospace eXploration Agency (JAXA), successfully landed and recorded images of an asteroid surface the first time while moving with hopping motion [1]. The low gravity of small celestial bodies make it possible to utilize non-traditional movement types with low energy consumption. In traditional rovers, the wheel diameter limits the size of obstacles the robot can traverse, small hopping robots can simply hop over an obstacle, eliminating the need for detour as demonstrated in Fig. 1.

An additional benefit of using small robots is the possibility to launch multiple units with a single rocket, or piggy back loading them with other cargo, greatly reducing the cost of space exploration missions. Multiple deployed robots can also work together in distributed exploration [2]. Figure 2 shows the different possible designs and working mechanism of hopping rovers the proposed navigation system is intended to be used for. An actuator tensions the spring that stores the energy used for hopping, while either a single large wheel attached to one end of the robot is used to turn the robot to the desired direction, or two wheels are used for turning and moving small distances [3, 4].

Previous research about small hopping rovers mostly focused on the mobility and mechanical design, and only a few methods have been proposed for navigation or visual odometry for such rovers. This paper proposes a novel approach to the navigation system and image processing of small hopping rovers, with the main motivation of reducing complexity and therefore reducing

*Correspondence: gabor_16@hmsl.elect.chuo-u.ac.jp
Faculty of Science and Engineering, Chuo University, Tokyo, Japan

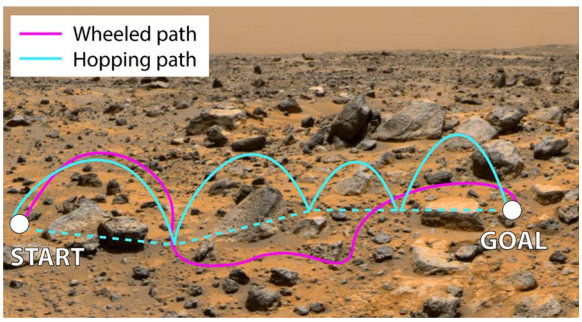



**Fig. 1** Advantages of hopping motion compared to wheeled robot paths (illustrated on a Mars image by NASA [1])

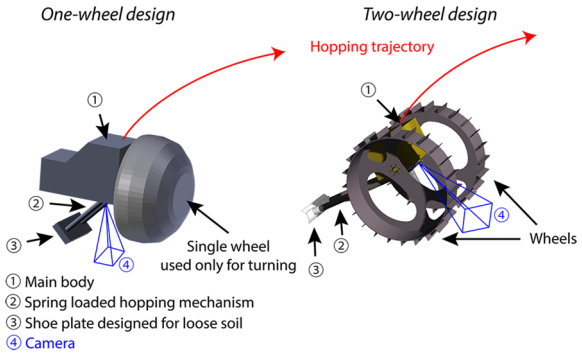



**Fig. 2** Different hopping rover designs [4]

the size and weight of the robot. Since hopping motion includes a higher risk of hitting obstacles, reducing the weight of rovers can be beneficial due to the lower impact forces, as well as reducing the necessary energy used for hopping.

Previously proposed navigation methods use stereo vision for at least the initialization step, and may have used multiple cameras facing different directions for trajectory reconstruction and scale estimation [5–7]. By reducing the size of the rover, the baseline between stereo cameras also becomes smaller especially compared to the hopping distance, reducing the accuracy of stereo reconstruction. Therefore, a fully monocular camera system is proposed. Taking advantage of the fact that the rover has to prepare for the next hop after landing, images can be batch processed in an offline manner without any real-time requirement. A Structure from Motion (SfM) pipeline is used for reconstruction, that provides more accurate and detailed results compared to tracking or optical flow based visual odometry, that uses images sequentially for reconstruction. The proposed method relies on accelerometers to calculate the gravity vector, and a sun direction sensor to determine the heading of the rover, both of which can be miniaturized with ease.

The main challenge of monocular systems is the scale ambiguity problem, i.e. not being able to reconstruct the absolute scale without additional information. To solve this problem, parabolic motion constraints are introduced to determine the real scale and orientation compared to the gravity vector. The reconstructed environment point cloud is evaluated based on protrusion from the ground plane to find obstacles, and image processing is used to determine the navigation path, taking the uncertainties of motion into account. To achieve global consistency and eliminate the position error caused by unpredictable motion after landing, the environmental point clouds of successive hops are matched with an Iterative Closest Point (ICP) based algorithm with non-overlapping part estimation, where only a section of the point clouds are matched. Conventional ICP performs poorly when there is only partial overlap between the point clouds, however iteratively eliminating non-overlapping parts can make the result converge to the global minimum. This position correction step not only eliminates the position error caused by landing, but also helps to create a seamless global map of the environment.

Figure 3 shows the overview of the proposed hopping rover navigation system. The proposed method is fully autonomous and is able to reconstruct the trajectory of hops and create a map of the environment, that is used to determine the navigation path, proving that a truly monocular system is not only feasible, but can accurately localize the rover and provide a detailed map of the environment as well. The following sections present each part shown in the overview, and finally results of the experiments conducted in simulation and real environments are presented and conclusions are drawn.

The next section explains the scene reconstruction technique, including sparse and dense reconstruction. In

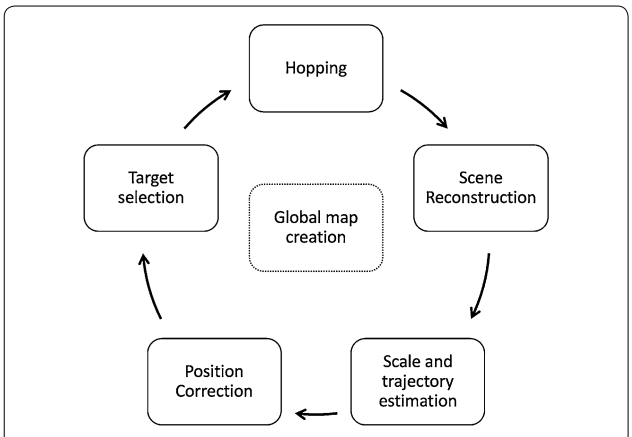



**Fig. 3** Overview of the navigation system

the following section, the scale and trajectory estimation process is presented, that is one of the main contributions of the proposed system and makes it possible to use a monocular camera. Next, the position correction step is discussed, that is performed after every hop to correct the position error introduced by the uncertainties of landing. In the following section, the target selection method is introduced along with the criteria to select the most suitable navigation path. The next section shows the global map creation after performing multiple hops. In the final section, experimental results are presented to prove the feasibility and effectiveness of the proposed system.

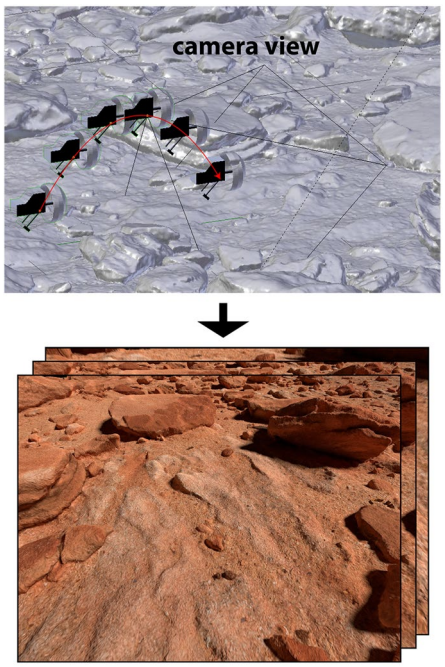
## Scene reconstruction

In extraterrestrial missions, exploration and mapping are both significant tasks alongside sampling and data acquisition. Although a Simultaneous Localization And Mapping (SLAM) [8] or Visual SLAM (VSLAM) approach could be used for localization and mapping, Structure from Motion (SfM) and Multi-View Stereo (MVS) reconstructions outperform them in terms of quality, but sacrificing computational speed [9].

However, in space exploration the robot has time to process data between hops and real-time operation is not required, making it possible to use the more accurate SfM and MVS. While the most common use for SfM is to reconstruct a sparse point cloud of a photographed subject or scene, it simultaneously recovers the poses of used cameras from the input images, that can be used to reconstruct the flight trajectory. Figure 4 shows how image frames are collected during hopping.

Since in the case of hopping rovers images are taken in order, the matching process can be sped up by using sequential matching, where correspondences are only searched in a number of consecutive frames, instead of exhaustive matching where every image frame is compared with every other. During reconstruction, images are added to the scene one by one. The newly added images' camera positions in the world coordinate system are determined by solving the Perspective-n-Point (PnP) problem. A triangulation process tries to estimate the 3D coordinates of the common points of the newly added image and images already added to the scene.

SfM has the benefit of using every image taken during a hop together for reconstruction, with the ability to disregard outliers that can introduce errors. Furthermore, Bundle Adjustment can be used, which is an optimization step minimizing the projection error of images [10]. Only some state of the art SLAM algorithms utilize BA, and usually only in a limited form. With MVS, every pixel of an image can be projected to a line and its spatial position can be determined by




col

**Fig. 4** Images taken during hopping in a simulated environment [19]

epipolar geometry, and a more detailed dense point cloud can be reconstructed, that can be used to identify and reduce the risk of hitting obstacles, unlike SLAM where only the extracted and matched feature points are mapped, creating only a sparse reconstruction. The dense point cloud is used to visualize the environment as well as to determine the position of the next hop for navigation in the target selection step.

Figure 5 illustrates the difference between the level of detail found in sparse and dense reconstruction using the same images. In sparse reconstruction only the 3D position of feature points used to solve the camera poses are calculated, just like in SLAM algorithms. In dense reconstruction, with the known camera poses stereo imaging is used to calculate the 3D position of every pixel in the image, creating a highly detailed cloud, that can be used to identify obstacles and select the navigation path.

An additional benefit of using an SfM approach is that it is more robust against accidental rotations and blur caused by fast motion. If it is not possible to calculate the camera poses from multiple images at some stage of the hop, other section of the motion can still be used to achieve accurate reconstruction for the whole hop, since all frames are processed together.

From our experience, as well as based on surveys on the performance of various methods, our system uses the COLMAP pipeline [11, 12].

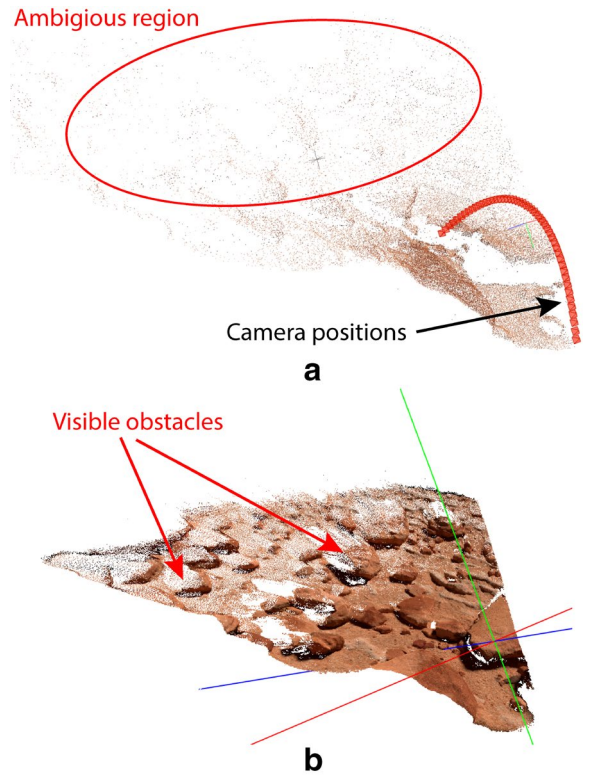



**Fig. 5** Difference in detail in sparse and dense reconstruction from the same images. **a** Sparse reconstructio. **b** Dense reconstruction with an SfM pipeline and MVS

## Trajectory and scale estimation

The main difficulty of monocular camera systems is finding the real scale factor to get scale-accurate distance measurements. While some conventional methods utilize stereo vision for initialization before hops to determine the scale factor, the proposed method relies only on monocular vision, and solves this challenge by a trajectory and scale estimation method with the introduction of parabolic motion constraints and the a priori knowledge of the gravity vector, that can be acquired by the accelerometers already equipped by the robot. Since no external forces work on the robot during flight it's center of gravity is moving along a parabola. The proposed algorithm finds the correct orientation and scale from the known gravity vector and time information. From this, the scale factor is determined with high accuracy while eliminating the need for additional cameras, thus saving weight and reducing the size of the rover, that is one of the key motivations of the presented navigation system. The flowchart of the scale and trajectory estimation process and it's place in the whole system can be seen in Fig. 6.

If the camera focal point is not in the center of mass but fixed to the robot, a simple geometric transformation can be used to adjust the reconstructed camera positions to the center of mass using the known distance between the two and the rotation of the camera.

$$\vec{P}' = \mathbf{T}\vec{P} \tag{1}$$

where $\mathbf{T}$ is the local-to-world transformation matrix created from the reconstructed camera position and rotation, $\vec{P}$ is the local vector pointing from the camera to the center of mass and $\vec{P}'$ is the world coordinate of the adjusted camera position.

A point cloud is created from these adjusted camera positions using the Point Cloud Library framework [13]. The camera positions are expected to be on a parabolic trajectory. By fitting a plane to the points, a plane that is parallel to the gravity vector can be obtained. Outlier camera positions with large position errors could distort this plane. Therefore, RANSAC (RANdom SAmple Consensus) algorithm is used for plane fitting [14], that has the advantage of disregarding outliers, improving the accuracy of scale estimation. The points are then projected to the plane and the plane is rotated to match the $XY$ plane. The camera position of the first image that is taken at the start of the hop is translated to the origin of the coordinate system, reducing the three-dimensional problem to two-dimensional. However, at this stage the the rotation of the parabola is still unknown. To find the rotation, a conic section is fitted to the points using least-squares method.

The equation of a general conic section is:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{2}$$

Since the first camera position was moved to the origin, by assuming that the conic section contains this point, $F$ can be eliminated.

Therefore the conic section is searched in form:

$$ax^2 + bxy + cy^2 + dx + ey = 0 \tag{3}$$

The equation contains five parameters, that is the minimum number of points that define a general conic section. To get the unknown parameters with least-squares method, the following function needs to be minimized:

$$f(x,y) = \sum_{i=1}^{n} (ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i)^2 = min \tag{4}$$

This can be solved by taking the derivatives with respect to each parameter, such as:

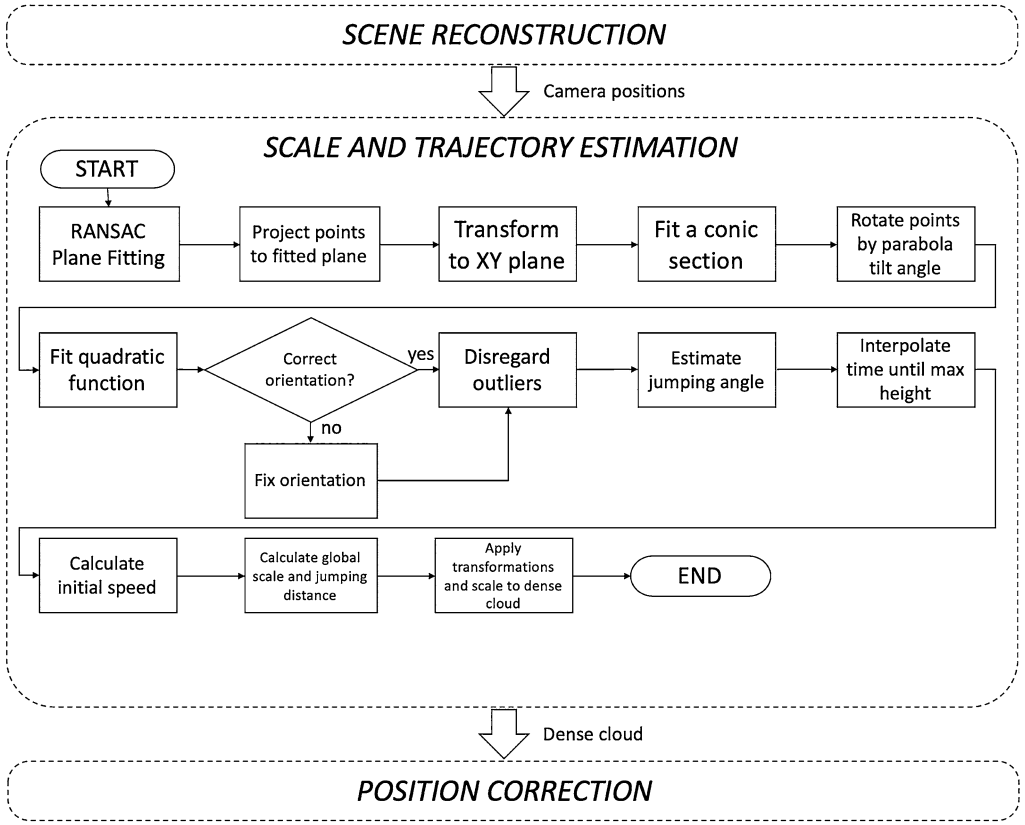



**Fig. 6** Flowchart of the scale estimation process

$$\begin{aligned}\frac{\partial}{\partial a}f(x,y) &= \frac{\partial}{\partial a}\sum_{i=1}^{n}(ax_i^2+bx_iy_i+cy_i^2+dx_i+ey_i)^2\\ &= \sum_{i=1}^{n}2ax_i^4+\sum_{i=1}^{n}2bx_i^3y_i+\sum_{i=1}^{n}2cx_i^2y_i^2\\ &\quad+\sum_{i=1}^{n}2dx_i^3+\sum_{i=1}^{n}2ex_i^2y_i=0\end{aligned} \tag{5}$$

and so on for each parameter. These can be arranged to form an equation system. The resulting equation system forms a symmetric matrix, where the eigenvalues hold the solution for the parameters assuming a non-trivial case where parameters are non-zero.

The rotation of the conic section can be calculated as:

$$\theta = \frac{1}{2}\arctan\left(\frac{b}{a-c}\right) \quad where \quad 0 \le \theta \le \frac{\pi}{4} \tag{6}$$

$\theta$ either shows the angle of the axis of symmetry or the angle of the diretrix. Therefore, after rotating the point around the origin with $\theta$, the points form a parabola where the axis of symmetry is parallel to either the $X$ or $Y$ axis. To find out the orientation, two parabolas are fitted to the points with least-squares using the following equations:

$$y = a + bx + cx^2 \qquad and \qquad x = a + by + cy^2 \tag{7}$$

In the correct orientation, the sum of least-square errors is significantly lower than the other. If the axis of symmetry is parallel to the $X$ axis, the points are rotated around the origin with 90°. To ensure that the flight trajectory starts in the $x \ge 0, y \ge 0$ quadrant, if $x$ values are negative the points are rotated around the $Z$ axis by 180°. If $c > 0$ in the $y = a + bx + cx^2$ parabola fitted to these points, the points are rotated around the $X$ axis by 180°. After fixing the orientation, more points that do not fit the parabola are disregarded by calculating the quadratic distance between the parabola and each point. If the parabola equation is $y = a + bx + cx^2$ , the square of the distance between the parabola and point $(x_i, y_i)$ is:

$$\begin{aligned}r^2 &= (x-x_i)^2+(y-y_i)^2\\ &= (x-x_i)^2+(a+bx+cx^2-y_i)^2\end{aligned} \tag{8}$$

Since $r \geq 0$, the solution can be found by minimizing $r^2$ instead of $r$:

$$\frac{\partial(r^2)}{\partial x} = 2c^2x^3 + 3bcx^2 + (b^2 + 2ac - 2cy_i + 1)x + (ab - by_i - x_i) = 0 \quad (9)$$

The resulting cubic function can be solved using the Cardano formula. Renaming the coefficients of equation 9, we get the form $ax^3 + bx^2 + cx + d = 0$, that can be reduced to a depressed cubic function $\chi^3 + p\chi + q = 0$ by substituting $\chi - \frac{b}{3a}$ for $x$, where $p = \frac{3ac-b^2}{3a^2}$ and $q = \frac{2b^3-9abc+27a^d}{27a^3}$. Since the curve and points are real, it is safe to assume that the cubic function above has one real solution and a complex conjugate pair that are irrelevant. The real solution also has to be positive, since $r$ is positive. Therefore, the following equation can be used to find the solution for $\chi$:

$$\chi = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} \quad (10)$$

To refine the parabola equation, every point whose distance is larger than the mean of distances is disregarded and the parabola is refitted for the last time using Eq. 7. An example of camera positions and the calculated parabola can be seen in Fig. 7.

#### Calculating the real scale

The vertical component of the hopping launch angle $\Theta_v$ is estimated from the derivative of Eq. 7.

$$\frac{\partial}{\partial x}a + bx + cx^2\Big|_{x=0} = b = \tan(\Theta_v) \quad (11)$$

From Eq. 7, the axis of symmetry can be calculated as $x_{sym} = -\frac{b}{2c}$

Because in projectile motion the horizontal component of velocity is constant, the following proportionality is true for every point: $\frac{x_i}{x_{sym}} = \frac{t_i}{t_{sym}}$, where $x_i$ is the x coordinate of point $i$, $x_{sym}$ is the axis of symmetry, $t_i$ is the timestamp of point $i$ (the first point in the origin has a timestamp of 0) and $t_{sym}$ is the timestamp at the maximum height of the trajectory. Time at the maximum height is interpolated by calculating $t_{sym}$ for every point and getting the median of the values. The vertical component of the hop is uniformly accelerated linear motion where the initial speed can be calculated as:

$$v_0 = \frac{t_{sym} \cdot g}{\sin(\Theta_v)} \quad (12)$$

where $g$ is the known gravitational constant.

From this, the scale factor is:

$$S = \frac{v_0 \cdot \cos(\Theta_v) \cdot t_{sym}}{x_{sym}} \quad (13)$$

where the numerator is the axis of symmetry calculated and denominator $x_{sym}$ is the axis of symmetry of the current point cloud.

The points are then scaled by the scale factor, and all of the previous transformation steps are applied to the dense environmental point cloud as well. Figure 8 shows the reconstructed environment and camera positions with adjusted scale and orientation.

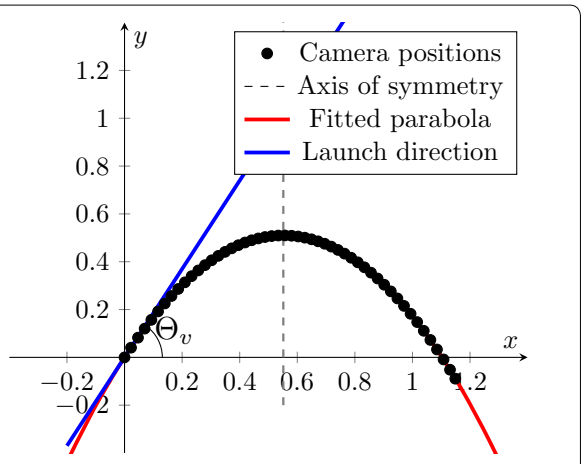



**Fig. 7** Camera positions in 2D with the estimated trajectory, vertical launch angle and axis of symmetry

## Position correction to eliminate error caused by bounces after landing

When landing the rover can hit the ground or an obstacle in a way that it bounces off to an unknown direction. Furthermore, depending on the gravity and type of surface, secondary bounces can also occur. However, in successive hops the launch position of a new hop is assumed to be the estimated landing position of the previous hop. The uncertainty in landing position deviation can be estimated, but since the robot is too close to the surface while resting on the ground, localization is challenging and unreliable using only the camera image.

To overcome the problem of this accumulating position error and make it possible to create an accurate global map and find the robot position in world coordinates, a position correction step is introduced using the created point environmental point clouds of successive hops. Because the camera has a wide field of view and the

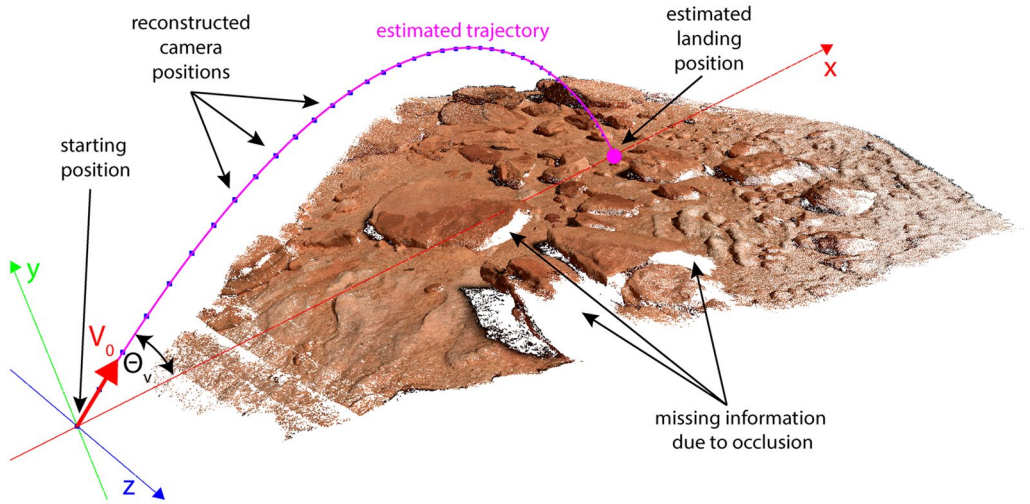



**Fig. 8** Reconstructed camera positions and point cloud of the environment

hops are performed in a way that there is a large overlap between visible areas, the created dense point cloud of successive hops can be aligned.

Alignment could be performed by minimizing the difference between the two point clouds using the Iterative Closest Point (ICP) algorithm [15]. However, ICP performs poorly when there is only partial overlap between the point clouds. This problem is solved by iteratively eliminating non-overlapping parts of the clouds. A smaller section of both point clouds are extracted by establishing boundaries in X and Z coordinates (where the Y axis is parallel to the gravity vector) slightly ahead of the second hop landing position, where overlap is expected even if the heading direction changed. Due to the possibly large non-overlapping parts relative to the section size, the ICP algorithm will most likely converge to a local minimum. After alignment, the point clouds are cropped with new boundaries set by the more limiting extremas of the two clouds, and ICP is performed again. This cropping step is iterated until the biggest difference in each respective boundary value of the two clouds gets below a set threshold, or in other words the number of cut points converge to zero. With the proposed method, correct alignment can be achieved, while using conventional ICP algorithm the point cloud often converges to a false alignment. The transformation matrix that aligns the point cloud of the last hop to the previous cloud is applied to the camera positions as well. One major benefit of the proposed correction step is that it fixes translational as well as rotational error that could cause a large position deviation after several hops.

Figure 9 shows the result of the position correction algorithm and the aligned clouds. Although there are

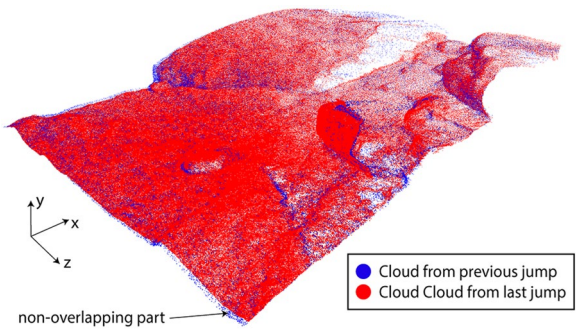



**Fig. 9** Result of the position correction

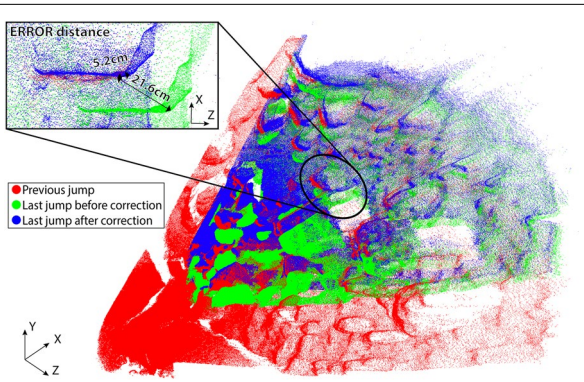



**Fig. 10** Dense point clouds before and after alignment

about 10 times more points in the same area in the second cloud seen in red due to the area is closer to camera in the second hop and therefore more detailed than the

first hop points seen in blue, the algorithm gets accurate alignment as long as the overlapping parts are eliminated.

Figure 10 shows the transformation matrix resulting from the algorithm being applied to the second hop's dense point cloud. The first hop points are shown in red, and the green and blue points are the second dense cloud before and after alignment respectively. The top left of the figure shows a detail of a rock from above. It can be clearly seen that the alignment improved the localization and successfully merged the two point clouds.

## Target selection with uncertainty calculation

Path planning in most traditional navigational systems are performed using a priori information about the environment and obstacles [16, 17]. However, the proposed system relies on the environmental information acquired by the previous hop, therefore information is limited and the rover only selects the next hop landing position, while knowing the desired direction of travel. After estimating the current position of the rover, it is necessary to select a suitable area for the next landing, taking into consideration the uncertainty of the landing position to guarantee the safety of the rover. The environment is evaluated in an autonomous process to find suitable areas for landing. The uncertainty and position error of the real landing position compared to the intended one increases with hopping distance. On the other hand, too short hopping distance produces less frames used for reconstruction as well as environment point cloud creation. Looking at the amount of energy used to traverse the robot horizontally, an 'ideal' launch angle would be 45°, but based on our experimental results the proposed system prefers 60°, that increases the number of images taken during hopping and decreases the possibility of large position error after landing due to larger impact angle from the ground plane.

Image processing is used for target selection. A square section of the dense point cloud around the current (estimated) landing position is projected to the $XZ$ plane to create a 2D image, where the current goal direction is upwards. During projection, each dense point is assigned to a pixel based on its coordinate values. If multiple points correspond to the same pixel, the arithmetic mean of values for each color channel is assigned. The square image size is chosen in a way that half of the image size is somewhat bigger than the maximum allowed hopping distance. The image resolution has to be selected accordingly to the camera resolution and field of view, that have a direct effect on the distance between neighboring dense points. If the resolution is too low, details are lost and the resulting image is blurred. If it is too high, unassigned

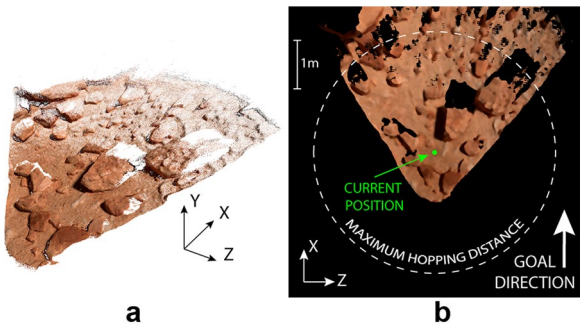



**Fig. 11** 2D image creation from the dense point cloud, **a** Original dense cloud, **b** 2D image (1 pixel = 1.2 [cm])

black pixels will remain especially in the regions more distant from the rover. Black pixels can be treated as noise and removed by applying a median filter to the image creating more uniform areas. However, some larger areas will still have no data, such as areas of the terrain not visible by the cameras due to being obscured by large obstacles. Occlusions can cause areas of the map to have missing information. These are treated as possibly dangerous areas. However, since the size of obstacles are much smaller than the hopping distance, occlusions does not affect the robustness of the navigation system.

Figure 11 shows an example of image creation. The image is 500 × 500 pixels that covers a 6 × 6 m area around the robots position, which means that 1 pixel is equal to 1.2 cm in real coordinates.

### Obstacle detection

Celestial surfaces usually have large rocks and boulders that the robot has to avoid hitting. The intended operational environment of the hopping rover is a relatively flat surface of softer soil covered in smaller and larger rocks, meaning the safe areas and obstacles can be classified based on protrusion. It is possible that the rover is located on a slight hill or slope so this ground plane is not necessarily parallel to the to the $XZ$ plane, that has a normal vector parallel to the gravity vector. However, locally large deviations in the shape of the ground are not expected and are considered obstacles. A plane is fitted to the dense points to get an estimated ground plane using RANSAC to exclude the points of obstacles that can distort the ground plane orientation. If the distance of a cloud point from this plane is larger than a threshold value, it is considered an obstacle. This threshold value can be set accordingly to the environment considering the size of the rover, the intended hopping distance and height as well as the size of the

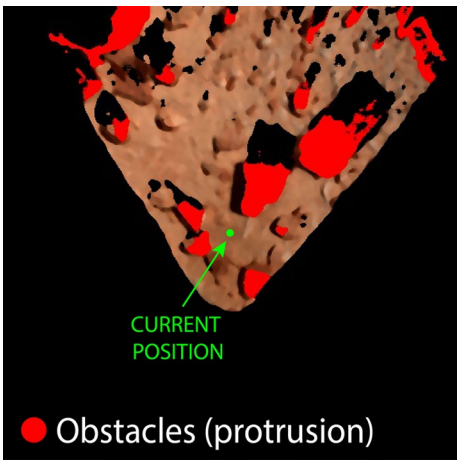



**Fig. 12** Areas identified as obstacles

previously encountered obstacles. A binary image mask is then created from these points with the same size as the 2D image map created beforehand, to classify areas into obstacle and non-obstacle. To remove noise and create more uniform areas, median filtering and closing (dilation followed by erosion) is applied. Figure 12 shows the created obstacle mask overlaid on the original image. The protrusion threshold is set to 8 [cm] in this environment.

#### Error ellipse for landing position

To increase the safety of the rover and find the most suitable areas for landing, uncertainties of motion have to be taken into consideration. The expected deviation from the planned landing position can be estimated by a number of factors, namely the angular error of the hop $\Delta\Theta$, launch speed error $\Delta v$ and uncertainty of the current position $\Delta s$ that can be gathered during experiments. Since the robot uses a loaded spring to hop, the launch speed is linear to the compression distance. Therefore, $\Delta v$ is independent of the hopping distance. $\Delta\Theta$ comes from the uneven surface and misalignment and also independent of the hopping distance. Figure 13 shows the error ellipse of a hop and the different uncertainties the system accounts for.

To simplify error calculations, the altitude difference between the launching and landing point is assumed to be zero. In this case the hopping distance can be calculated as:

$$f(v_0, \Theta_v) = d = \frac{v_0^2}{g} sin(2\Theta_v) \tag{14}$$

where $d$ is the hopping distance, $\Theta_v$ is the vertical launch angle, $v_0$ is the launch speed and $g$ is the gravity constants. Since the variables are independent of each other and the errors are assumed to have Gaussian distribution, the variance formula can be used to calculate error propagation. The major axis of the error ellipse can be calculated as:

$$\begin{aligned}\Delta d_{major} &= \sqrt{\left(\frac{\partial f}{\partial \Theta_v}\Delta\Theta\right)^2 + \left(\frac{\partial f}{\partial v_0}\Delta v\right)^2 + \Delta s^2} \\ &= \sqrt{\frac{4\Delta\Theta^2 v_0^4 cos^2(2\Theta_v)}{g^2} + \frac{4v_0^2\Delta v^2 sin^2(2\Theta_v)}{g^2} + \Delta s^2}\end{aligned} \tag{15}$$

And the minor axis of the error ellipse as:

$$\Delta d_{minor} = \sqrt{d^2 sin(\Delta\Theta)^2 + \Delta s^2} \tag{16}$$

The major and minor axes of the ellipse are calculated for each pixel that was not marked as an obstacle but has values assigned. A binary image mask is then created with this ellipse. If the ellipse overlaps with unknown or

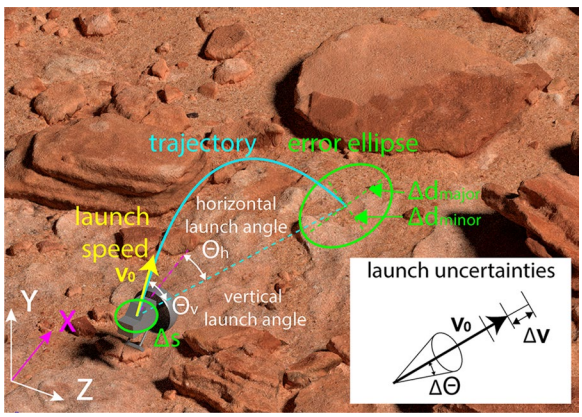



**Fig. 13** Error ellipse and types of uncertainties

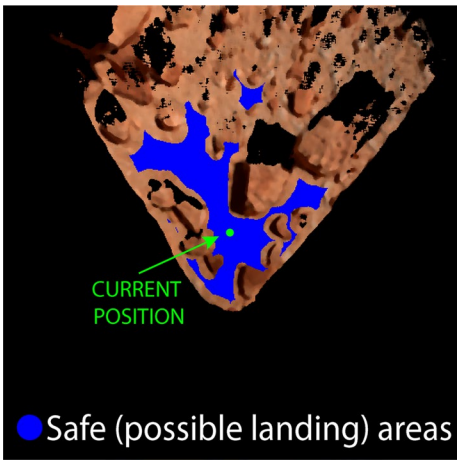



**Fig. 14** Areas safe to land on

obstacle areas, the ellipse center pixel is marked unsafe for landing.

The blue areas of Fig. 14 show the areas considered safe for landing.

### Landing point selection

After creating a mask with possible landing positions, the most suitable landing position needs to be selected. The landing position with the lowest possibility of hitting obstacles is the one most distant from dangerous areas. Therefore, a distance map is created from possible areas. Distance transform is performed by calculating the Euclidean distance of each pixel from the closest black pixel. Due to the physical limitations of the rover and to avoid high speed impacts during landing, the maximum hopping distance is limited. Furthermore, a minimum hopping distance limit is also set to make sure there are enough images for trajectory reconstruction and the camera covers a sufficiently large area for the environmental reconstruction as well. A maximum of 90° horizontal deviation is allowed from the desired travel direction, but smaller angular deviation is preferred. To quantify these requirements a simple weighting function is defined:

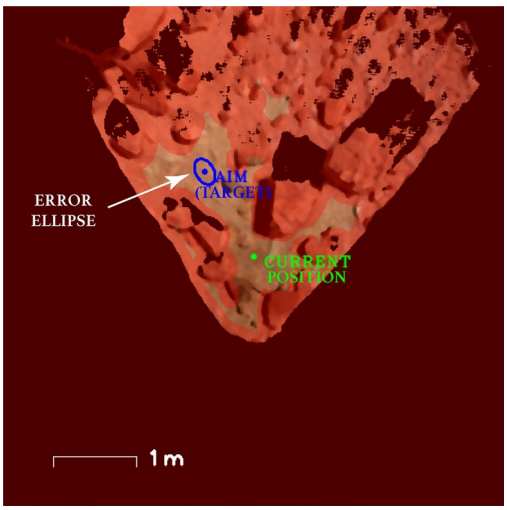



**Fig. 16** Selected target for next landing showing the current position, error ellipse and dangerous areas

$$I_{gradient}(u,v) = \begin{cases} cos(\Theta_h) = \left| \dfrac{v_c - v}{\sqrt{(u_c-u)^2 + (v_c-v)^2}} \right|, \\ \quad \text{if } d_{max} \geq \sqrt{(u_c-u)^2 + (v_c-v)^2} \geq d_{min} \text{ and } v \leq v_c \\ 0, \quad \text{otherwise} \end{cases} \tag{17}$$

where $u$ and $v$ are the pixel coordinates, $\Theta_h$ is the horizontal hopping angle defined by the angle between the line of the desired direction (up) and the line defined by $(u, v)$ and the image center $(u_c, v_c)$, $d_{min}$ and $d_{max}$ are the mimimum and maximum allowed hopping distances respectively. Figure 15 shows the distance map of the safe areas and the created gradient map.

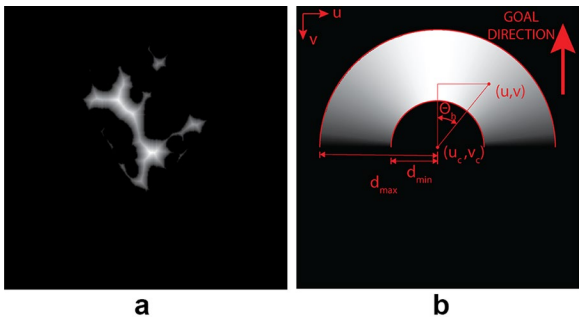



**Fig. 15** Landing position selection. **a** Distance map, **b** Gradient map

Gradient maps like this can be used as weighting functions to find the most suitable points based on application specific criteria [18]. The distance and gradient map can be blended together by calculating the pixel-wise weighted average, excluding unsafe and unknown regions. This method makes it possible to select a single best candidate, based on any number of independent criteria. The weights can be set to enforce a certain criteria more strictly, the safety aspect with the distance map or goal oriented travel with the gradient map. The next target for hopping is the highest pixel value of resulting selection image. The current position and next target can be seen in Fig. 16, where the dangerous areas are marked red.

The pixel coordinates are transformed into real world coordinates and the required launch speed and angle can be calculated from Eq. 14. To avoid hitting obstacles during hopping, possible collision is checked. Every point that is closer to the next translation plane than the largest protrusion of the rover's body measured from it's center of mass is projected to this plane. The distance of these points from the next trajectory are calculated using Eqs. 8 and 9. Points closer to the trajectory than the larges protrusion are considered to be on collision course. To exclude possible outliers, at least a number of points have

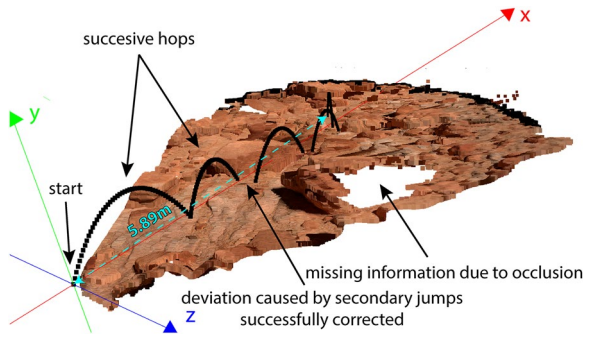



**Fig. 17** Reconstruction of a series of hops

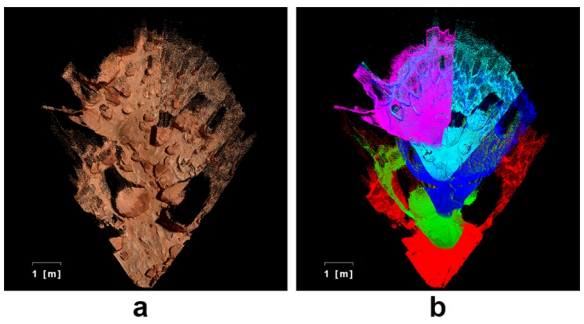



**Fig. 18** Global map created after 5 successive hops. **a** True color **b** False color

to be on collision course to disregard the current trajectory. If collision is detected, the current error ellipse in the image used for target selection is masked black and the next best landing position candidate is examined. The horizontal component of the hopping angle can be calculated from the current and target landing positions. Before loading the spring with the necessary force, the horizontal orientation of the robot is adjusted using the information from the equipped sun direction sensor. Sun direction sensors are a set of photodiodes mounted in a configuration that can be used to calculate the direction of a light source [2].

## Global map creation

The surface of a planets and asteroids hold valuable information and mapping these surfaces to find areas of interest as well as to gather data is one of the objectives of space missions. Therefore, to map the surface, dense point clouds are projected to the XZ plane and a 2D color image is created the same way as for target selection. Figure 17 shows the reconstructed trajectories and environment of a series of hops.

The global map resulting from 5 hops and a false color image showing the areas visible by each hop can be seen in Fig. 18. The false color image shows that there is enough overlap between point clouds even when the angular difference between hopping directions is substantial.

## Results and discussion

To prove the feasibility of the proposed system, multiple tests were conducted in simulated as well as real environments. First the motion model of hopping is discussed.

During hopping the only force affecting the rover is gravity. Therefore, the hopping trajectory is a parabola on a plane that is perpendicular to the ground plane (defined by the gravity vector as normal vector). By design, the center of mass is not perfectly in line with the spring force at launch, which introduces a constant rotation of the rover during the hop. Since the takeoff surface is not necessarily parallel to the ground plane, the rotation plane and translation plane are also not parallel. However, during experiments with hopping rover designs not discussed in this paper, we found that the rotation during hops is minor, and it is guaranteed that even a single camera observes the terrain during the whole hop. Furthermore, experiments concluded that reconstruction is successful even in extreme cases, where multiple full rotations occur before landing. With the presented SfM approach, the trajectory can be reconstructed as long as at least 3 different image frames cover a common ground area at any point in the trajectory, but accuracy and the detail of the reconstructed environment increases with the number of reconstructed camera positions.

Since the computational cost rapidly increases with the images used, the framerate has to be considered. We found that using roughly 50 images is a good compromise in quality/computational cost. The framerate has to be set accordingly to the environment, since gravity and the desired hopping distance correlates to the time spent hopping. Generally the rover is designed to hop 1–10 [m] spending a few seconds off the ground.

Especially after launch and near landing, the velocity of the rover is high and the camera is close to the ground, which increases motion blur. Therefore motion blur was implemented in the simulation environment experiments as well, although we found that even though it is noticeable it has minor effect on the reconstruction and map creation.

### Evaluation in simulation environment

Chains of successive hops were performed in simulated environments. The purpose of this evaluation is to show that the proposed method is able to reconstruct the flight

**Table 1 Estimated positions compared to ground truth after multiple hops**

| | hop 1 | hop 2 | hop 3 | hop 4 | hop 5 |
|---|---|---|---|---|---|
| GT x [m] | 0 | 1.350 | 2.450 | 3.740 | 4.920 |
| GT y [m] | 0 | – 0.050 | – 0.070 | – 0.090 | – 0.071 |
| GT z [m] | 0 | 0.300 | 0.260 | 0.120 | 0.191 |
| est. w/o ICP x [m] | 0 | 1.157 | 2.144 | 3.320 | 4.444 |
| est. w/o ICP y [m] | 0 | 0.028 | 0.037 | 0.052 | 0.067 |
| est. w/o ICP z [m] | 0 | – 0.133 | – 0.255 | – 0.184 | – 0.208 |
| error w/o ICP [m] | 0 | 0.480 | 0.609 | 0.538 | 0.636 |
| error w/o ICP [%] | 0 | 34.718 | 24.689 | 14.362 | 12.921 |
| ICP [m] | 0 | 0.235 | 0.141 | 0.277 | 0.078 |
| est. w/ ICP x [m] | 0 | 1.250 | 2.331 | 3.619 | 4.868 |
| est. w/ ICP y [m] | 0 | – 0.004 | – 0.023 | – 0.009 | 0.019 |
| est. z w/ICP [m] | 0 | 0.080 | 0.179 | 0.356 | 0.141 |
| error w/ ICP [m] | 0 | 0.017 | 0.025 | 0.067 | 0.095 |
| error w/ ICP [%] | 0 | 1.239 | 0.995 | 1.788 | 1.936 |

trajectory as well as the environment accurately, and also navigate by choosing targets for landing without hitting obstacles. The criteria for evaluation is the position error of the launch point of each hop compared to the ground truth, that is easily accessible in a simulation environment. Furthermore, the performance of the position correction step is also evaluated to show that it can successfully eliminate position errors caused by landing. A highly detailed photoscanned surface model of a rock sandstone desert environment [19] was imported into a popular 3D graphics and rendering software [20] to be used as ground truth. Using the built in physics engine with addons [21], the hops of the rover can be simulated. A wide angle camera is fixed to the rover and the frames such as ones seen in Fig. 4 are used for reconstruction. The camera was facing 45° downwards and pointing in the direction of travel and the robot had no rotational motion during hopping. However, the deviation from the landing position to the next hop starting position caused by secondary hops is simulated, to test the correction algorithm's performance. The average hopping distance was 0.98 [m], average number of frames taken during hopping was 49 with the average hopping time of 1.6 [s] in 1.62 $[\frac{m}{s^2}]$ simulated Moon gravity. The reconstruction results can be seen in Fig. 17.

Table 1 contains the ground truth and estimated starting position of each hop for every coordinate, the adjusted distance of the position correction step and the absolute and relative error, that is defined by the absolute error divided by the traveled distance from the origin both with and without the position correction step. Since the navigation relies on the data acquired by the previous hop, the rover can operate long distance independently

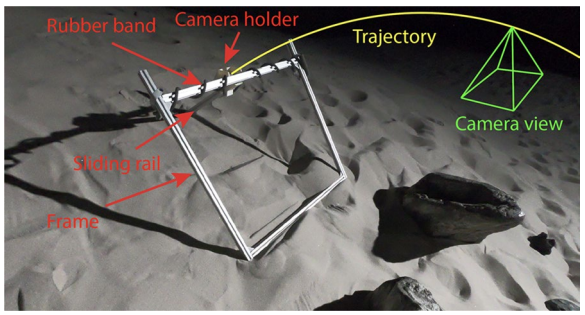



**Fig. 19** Launch equipment

and select targets even if the self-position is inaccurate. However, the accumulation of absolute error is much larger without the correction step. Furthermore, with the help of the introduced correction method, the dense point clouds were successfully merged to a seamless global map. From the data as well as the created dense point cloud of the environment, we can conclude that the system is able to reconstruct its surrounding environment as well as self-position with high accuracy, and also navigate independently without hitting obstacles.

#### Evaluation in real environment

To test the ability of the proposed a system in a real environment, several experiments were performed in a sandbox at a Japan Aerospace eXploration Agency (JAXA) test facility, where many currently operational or previously successful space exploration rovers are being tested. Here a celestial surface-like environment and lighting can be recreated. The environment consists of a large sandbox with natural and artificial rocks and boulders scattered randomly. A single large spotlight was placed approximately 20 m away to imitate sunlight and create hard shadows.

Since the robot is designed to operate in a gravity much lower than Earth, it cannot be used to hop long distances with enough image frames. Therefore, a special launch equipment was created to throw a wide angle camera instead, shown in Fig. 19 to test the proposed system independently of the hardware limitations. Due to the physical limitations of higher gravity, successive hops and long distance navigation were not performed. Since the ground is soft, the camera sank into the sand after every landing without secondary bounces. This made it possible to verify the hopping distance by a simple measuring tape with cm accuracy, providing ground truth for the hopping distance estimated by the proposed method. The purpose of this evaluation is to show that the proposed method is able

to accurately reconstruct the hopping trajectory in real environments, even in low light conditions.

The launch equipment is constructed from aluminum frames and a 3D printed camera holder tray slides along a rail. The launch energy is provided by a rubber band. As soon as the holder hits the end of the rail, the camera flies toward the goal facing the ground with some rotation. A 94.4° vertical field of view wide angle camera was used. Since it is facing towards the ground, the reconstructed point clouds cover a smaller area than in simulated environments. The launch angle was approximately 45° and the average hopping distance roughly 5 [m] with the hopping time of about 1 [s]. To compensate for the difference in gravity, the framerate of the camera was increased and faster shutter speed was used to decrease the effect of motion blur. This consequently means that the images are slightly underexposed, further pushing the limits of the already low light environment. One of the advantages of the proposed method is that it can calculate the trajectory and scale even when the launch was imperfect and the robot unexpectedly rotates during hopping.

Experimental results show that the average error between the calculated hopping distance and the real measured distance was less than 4% and never exceeded 10% or roughly 15 [cm]. This was verified by a measuring tape between distinct points in the real environment and comparing it to the distance calculated from the cloud coordinates. Even in extreme conditions, when the robot performed several full rotations during hopping, the trajectory and scale estimation succeeded and only details of the dense point cloud was lost due to the missing frames when the camera was not facing the ground. Figure 20 shows an example of the environmental reconstruction. Even in low light conditions and highly reflective and uniform looking sand, not only the obstacles but the ground plane is reconstructed, although missing patches and noise is more visible then in simulation environment.

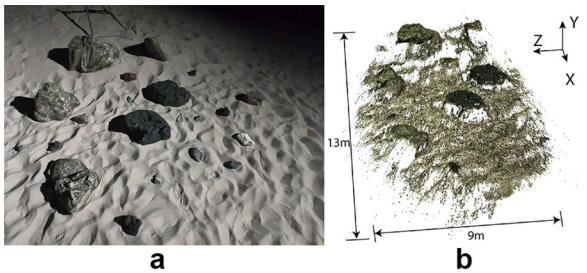



**Fig. 20** Real environment experiments. **a** Test environment, **b** Reconstructed dense point cloud

## Conclusions

In this paper a novel monocular navigation system was presented for hopping exploration robots using SfM. The proposed trajectory and scale estimation method is able to accurately determine the real scale and orientation of the environment for single hops. The introduced position correction method is able to eliminate errors caused by landing, making long distance navigation and mapping possible. Furthermore the proposed target selection process can navigate the robot towards the desired direction without hitting obstacles. From the experimental results it can be concluded that the presented method is feasible and makes it possible to use single monocular cameras, simplifying hardware and widening the possibility for low-cost space exploration. The proposed system is also robust and adaptable to different environments and lighting conditions as well as unaffected by the accidental rotations during hopping.

Limitations of the system are the required computational power, that makes it difficult to implement a similar system to applications where fast operation is required, and the required external light sensor to accurately determine the orientation before hopping.

Future work includes the implementation of the system on hopping rover prototypes to perform more experiments analyzing the placement and type of camera used and robustness of navigation under different conditions.

**Acknowledgements**
The authors would like to thank JAXA for making their test facility sandbox available for the real environment evaluation.

**Authors' contributions**
GK developed the concept and formulation, programmed and implemented the system, evaluated it in simulated and real environments and wrote the manuscript. YK, TM and HH supervised the findings of this work. All authors read and approved the final manuscript.

**Funding**
A part of this work has been supported as joint project research subject by the Institute of Science and Engineering in Chuo University.

**Availability of data and materials**
Not applicable.

**Competing interests**
The authors declare that they have no competing interests.

Received: 24 December 2019 Accepted: 16 May 2020
Published online: 09 June 2020

**References**

1. Yabuta H (2019) Arrival, touchdown and sequel to the voyage of hayabusa2. Nat Astron 3:287. https://doi.org/10.1038/s41550-019-0750-y
2. Matsumoto R, Sato T, Maeda T, Kunii Y, Kida H (2019) Position and attitude estimation for distributed exploration of small rovers using flash light from leader agent. In: 2019 9th international conference on

recent advances in space technologies (RAST), pp. 741–745. https://doi.org/10.1109/RAST.2019.8767809
3. Yoshikawa K, Otsuki M, Kubota T, Maeda T, Ushijima M, Watanabe S, Sakamoto K, Kunii Y, Umeda K (2017) A new mechanism of smart jumping robot for lunar or planetary satellites exploration. In: 2017 IEEE Aerospace Conference, pp 1–9. https://doi.org/10.1109/AERO.2017.7943807
4. Maeda T, Kunii Y, Yoshikawa K, Otsuki M, Yoshimitsu T, Kubota T (2018) Design of shoe plate for small hopping rover on loose soil. In: 2018 IEEE Aerospace Conference, pp. 1–7 https://doi.org/10.1109/AERO.2018.8396530
5. So EWY, Yoshimitsu T, Kubota T (2009) Hopping odometry: Motion estimation with selective vision. In: 2009 IEEE/RSJ international conference on intelligent robots and systems, pp 3808–3813. https://doi.org/10.1109/IROS.2009.5354065
6. So EWY, Yoshimitsu T, Kubota T (2011) Visual odometry for a hopping rover on an asteroid surface using multiple monocular cameras. Adv Robot 25(6–7):893–921. https://doi.org/10.1163/016918611X563355
7. So EWY, Yoshimitsu T, Kubota, T (2010) Divergent stereo visual odometry for a hopping rover on an asteroid surface. In: i-SAIRAS - International Symposium on Artificial Intelligence, Robotics and Automation in Space
8. Davison AJ, Reid ID, Molton ND, Stasse O (2007) Monoslam: real-time single camera slam. IEEE Trans Pattern Anal Mach Intell 29(6):1052–1067. https://doi.org/10.1109/TPAMI.2007.1049
9. Bianco S, Ciocca G, Marelli D (2018) Evaluating the performance of structure from motion pipelines. J Imag 4:98
10. Triggs B, McLauchlan P, Hartley R, Fitzgibbon A (2000) Bundle adjustment – a modern synthesis. In: VISION ALGORITHMS: THEORY AND PRACTICE, LNCS, pp. 298–375. Springer
11. Schönberger JL, Frahm J-M (2016) Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR)
12. Schönberger JL, Zheng E, Pollefeys M, Frahm J-M (2016) Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV)
13. Rusu RB, Cousins S (2011) 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China
14. Raguram R, Chum O, Pollefeys M, Matas J, Frahm JM (2013) Usac: a universal framework for random sample consensus. IEEE Trans Pattern Anal Mach Intell 35(8):2022–2038. https://doi.org/10.1109/TPAMI.2012.257
15. Besl PJ, McKay ND (1992) A method for registration of 3-d shapes. IEEE Trans Pattern Anal Mach Intell 14(2):239–256. https://doi.org/10.1109/34.121791
16. Injarapu ASHHV, Gawre SK (2017) A survey of autonomous mobile robot path planning approaches. In: 2017 International conference on recent innovations in signal processing and embedded systems (RISE), pp 624–628. https://doi.org/10.1109/RISE.2017.8378228
17. Kovacs B, Szayer G, Tajti F, Burdelis M, korondi P (2016) A novel potential field method for path planning of mobile robots by adapting animal motion attributes. Robotics and autonomous systems. https://doi.org/10.1016/j.robot.2016.04.007
18. Kovacs G, Kunii Y, Maeda T, Hashimoto H (2019) Saliency and spatial information-based landmark selection for mobile robot navigation in natural environments. Adv Robot 33(10):520–535. https://doi.org/10.1080/01691864.2019.1602564
19. Quixel Megascans Library (2019) https://quixel.com/assets/rkhtq. Accessed 20 July 2019
20. Blender (2019) https://www.blender.org/. Accessed 20 July 2019
21. Craddock N (2019) Projectile addon for Blender. https://github.com/nateccraddock/projectile. Accessed 20 July 2019

## Publisher's Note