REAL-TIME LANDING BASED ON OPTIMALITY PRINCIPLES AND VISION

de Croon, G.C.H.E[(1)] **and Izzo, Dario**[(2)]
[(1)(2)]*Advanced Concepts Team, European Spage Agency*

Abstract: *The mainstream approach to landing spacecraft on planetary bodies makes use of a guidance profile computed off-line and uploaded to the spacecraft, a navigation system relying on sensors such as laser altimeters, and a control system able to close the loop and land the spacecraft safely by keeping it as close as possible to the precomputed descent profile. In this paper, in contrast, we propose and develop an alternative architecture where a state-feedback based on optimality principles is computed on-board and sensor fusion between vision and accelerometers are used to provide a state estimate fed into the state-feedback. Our architecture is suitable for "nano-landers" not carrying expensive mass and power-hungry sensors as well as for emergency landing procedures able to bring the spacecraft to touch down without any supervision.*

Keywords: *Onboard real-time optimal control, vision-based landing, spacecraft landing*

1. Introduction

From a control theory point of view, spacecraft are fairly complex systems having many degrees of freedom (including multiple rigid and flexible modes), thus the space of all possible controls is rather large and finding an opimal control can be a rather time-consuming endeavour which is typically not made by the on-board CPUs. While computers are continuously improving their performance and new parallel computing paradigms and architectures are radically changing the definition of what is and what is not feasible in terms of on-board computations and thus in terms of autonomy, the need for fast algorithms able to allow spacecraft to take autonomous (possibly optimal) decisions has not faded away.

With this respect a valuable lesson can be learnt from nature. Observables such as the ventral optic flow and the time-to-contact (known to form the basis of the information processed by some insects [1, 2, 3, 4] but also humans [5, 6]), have been proposed and studied in the context of spacecraft landing scenarios and mass optimality [7, 8, 9]. Tracking an exponentially decreasing time-to-contact was shown to result in a rather low mass consumption, while being computationally efficient and requiring only estimates of the ventral optic flow and of the time-to-contact (obtained from on-board cameras) [9]. While these studies led to interesting algorithms from the point of view of computational efficiency, the propellant mass penalty associated (estimated to be around 15% in [9]) can be of concern for applications where it is not affordable to mis-use such a precious resource.

Some inspiration to improve on those results could come from recent ideas put forward in neuroscience on optimal feedback control as a theory of motor coordination [10, 11]. Humans (and so should robots) plan their actions optimally using a simplified model of their body and of the environment, key to allow fast neural computations of optimal actions. The difference between reality and the models used to plan optimally is then accounted for (and almost cancelled) by the continuous updates coming from sensing the world and re-evaluating the optimal course of action.

In this paper we take such a scheme and apply it to spacecraft landing where mass optimality is seeked as a main objective. In Section 2. we introduce the simplified internal model the spacecraft uses to plan its descent and estimate the numerical effort needed by the spacecarft to compute its actions based on optimality principles. In Section 3. we introduce the spacecraft navigation system used to estimate the state. We make use of a rather basic navigation platform made of a downward pointing camera and three accelerometers. In the following sections we perform a few landing experiments showing the method's robustness and its reduced propellant mass consumption (only 3% away from the optimal value). The choice of the particular basic navigation platform is motivated by our interest in spacecraft such as "nano-landers" where the overall mass budget forbids the use of heavy sensors, but our results and the proposed control architecture is more general and we infact use an Apollo-like scenario and spacecraft throughout the paper to simulate the proposed scheme.

2. On-board computation of the optimal state-feedback

Consider the optimal control problem of a landing spacecraft written in the form of Eq.(1).

$$\begin{aligned} \text{find:} \quad & t_f, \mathbf{u} \in \mathscr{U} \\ \text{to maximize:} \quad & m(t_f) \\ \text{subject to:} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ & \overline{\mathbf{x}}(t_f) = \overline{\mathbf{x}}_f \\ & \mathbf{x}(0) = \mathbf{x} \end{aligned} \tag{1}$$

where $\mathscr{U}$ is the space of admissible controls and $\overline{\mathbf{x}}$ is a "slice" of the state $\mathbf{x}$. Let us indicate with $\mathbf{u}^*(t)$ the solution to this problem. Such an optimal control solution depends on the spacecraft state $\mathbf{x}$: we may thus write $\mathbf{u}^*(t,\mathbf{x})$ and define the state feedback $\mathbf{u}(\mathbf{x}) = \mathbf{u}^*(0,\mathbf{x})$. Under the assumption to be able to find $\mathbf{u}^*(t,\mathbf{x})$, this trick allows us to have a numerical procedure to compute an optimal state-feedback. Such a procedure is not very appealing from the computational point of view as the complexity of the spacecraft dynamic $\mathbf{f}(\mathbf{x},\mathbf{u})$ makes the cost of computing $\mathbf{u}(\mathbf{x})$, following the above scheme, quite significant. This conclusion is however radically different if we substitute the dynamic $\mathbf{f}$ with a much-simplified version $\bar{\mathbf{f}}$. The computation of $\mathbf{u}^*(t,\mathbf{x})$ can be done numerically using direct, indirect or pseudospectral methods. In this paper we use the direct method described in Izzo et al.[8] which considers the trajectory as divided into n segments and places an impulsive $\Delta\mathbf{V}$ change in the middle of each segment. The thrust may then be derived by assuming it constant along a segment and equal to $\mathbf{u}^* = \Delta\mathbf{V}\delta t$, where $\delta t = t_f/n$ is the segment duration. In order to obtain the solution $\mathbf{u}^*(t,\mathbf{x})$ with the highest possible frequency, when solving the optimal control problem we use the simplified dynamic $\bar{\mathbf{f}}$ detailed in Eq.(2).

$$\begin{aligned} \dot{x} &= v_x, \quad \dot{v}_x = u_x/m \\ \dot{y} &= v_y, \quad \dot{v}_y = u_y/m \\ \dot{z} &= v_z, \quad \dot{v}_z = u_z/m - g \\ \dot{m} &= -\sqrt{(u_x^2 + u_y^2 + u_z^2)}/I_{sp}g_0 \end{aligned} \tag{2}$$

The spacecraft, in reality, has a far more complex dynamic, but it is convenient to plan its actions using a simpler model as to be able to plan actions using a state-feedback as explained above. In

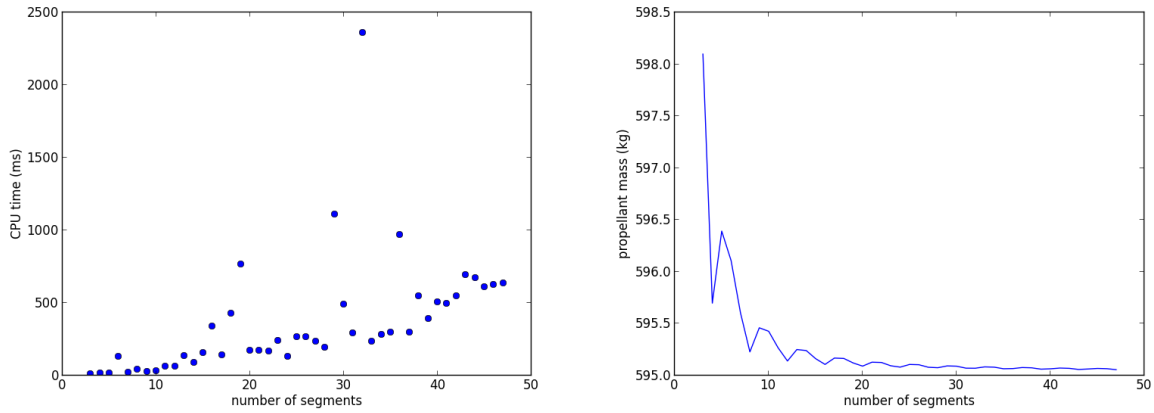

Figure 1. Performance of the optimal control solver with respect to the number of segments used. CPU time is shown in millisecond (left), while the optimal value found for the propellant mass is shown in Kg (right)

this paper we thus define the state-feedback $\mathbf{u}(\mathbf{x})$ as $\mathbf{u}^*(0,\mathbf{x})$, where $\mathbf{u}^*(t,\mathbf{x})$ is the solution to the optimal control problem in Eq.(1) computed using the impulsive direct method developed in [8] and the simplified dynamics $\bar{\mathbf{f}}$ defined by Eq.(2). The computed state-feedback will thus only be an approximation to the real optimal state-feedback: it employs a simplified model of the spacecraft and its environment (Eq.(2)) and it employs a simplified model of the thrusting action (impulsive) over a finite number of segments n. We now discuss how much this last approximation impacts on the resulting planned descent. We introduce an Apollo-like scenario [12]. The scenario involves the following high-gate conditions: $T_{max} = 45760$ [N], $I_{sp} = 311$ [s], $m_0 = 9472.06$ [kg], $v_{x_0} = 150$ [m/s], $v_{z_0} = -44$ [m/s], $z_0 = 2300$ [m], and $g = 1.623$ [m/s^2]. At low-gate we set: $v_{x_{t_f}} = 0$ [m/s], $v_{z_0} \in [-2.5, 0]$ [m/s], $z_0 <= 10$ [m].

Using our scheme, we solve the optimal control problem with varying number of segments n and we report the results in terms of propellant mass and CPU-time employed in Figure 1[1]. Looking to such a figure, it is immediately apparent how, already using 5 to 10 segments, one obtains a satisfactory approximation to the optimal descent (in terms of used propellant), while having a rather small CPU-load. The result in Figure 1 has been obtained using SNOPT [13] as a solver for the non linear programming problem resulting from the impulsive direct method. Other solvers were also tested, notably the open source interior point optimization method IPOPT [14] was also found to be able to achieve similar performances but its convergence starts to be troublesome for $n > 10$ when propellant mass optimization is seeked.

Starting from a state $\hat{\mathbf{x}}$ estimated by the navigation system (described later) we may then compute the optimal thruster action $\mathbf{u}(\hat{\mathbf{x}})$ using the techniques described above. With $n = 10$ nodes in the impulsive approximation, we are able to compute this at frequencies as high as 10Hz-20Hz, compatible with an on-board implementation of the whole system.

[1] Our experiments were done in a single Intel(R) Xeon(R) CPU - X5355 @ 2.66GHz.

3. Vision-based state estimation

In this section, it is first explained what visual measurements are used for state estimation and how they are combined with the accelerations. Subsequently, the computer vision algorithm that performs the visual measurements is discussed. Finally, the data fusion involved in the state estimates is explained.

3.1. Combining visual measurables and accelerations

The combination of an Inertial Measurement Unit (IMU) with vision is referred to as *vision-aided inertial navigation* and it is known to considerably improve the accuracy of the state estimates. Previous studies in this area [15, 16, 17, 18] assume an accurate initial estimate, typically relying on additional sensors such as laser altimeters. Instead, in the current approach also the initial state is obtained purely on the basis of vision and proprioception. In particular, the state estimation relies on the ventral optic flow and the time-to-contact. These bio-inspired visual observables can be measured with extremely light-weight and energy efficient neuromorphic sensors [19] or with an uncalibrated linear camera.

Formally, the ventral flow is defined as $(\omega_x, \omega_y) = (\frac{v_x}{z}, \frac{v_y}{z})$. It provides information on the lateral velocities relative to the height. The time-to-contact is defined as $\tau = -\frac{z}{v_z}$, and captures the vertical velocity relative to the height.

The ventral flow and time-to-contact can be used directly for control [20, 7, 21], and one can even obtain an acceptable mass efficiency by having the time-to-contact decrease exponentially [9, 22]. However, additional access to accelerometer readings allows one to retrieve the actual height and velocities of the spacecraft. The necessary equations are derived as follows. It starts with the equation for the time-to-contact:

$$v_z \tau = -z \tag{3}$$

Taking the time derivative, we get:

$$a_z \tau + v_z \dot{\tau} = -v_z \tag{4}$$

$$(1 + \dot{\tau}) v_z = -a_z \tau \tag{5}$$

$$v_z = -\frac{a_z \tau}{1 + \dot{\tau}} \tag{6}$$

A similar approach can be applied to the ventral flow. Below, we derive the equation for ω_x:

$$\omega_x z = v_x \tag{7}$$

Taking the time derivative gives:

$$\dot{\omega}_x z + \omega_x v_z = a_x \tag{8}$$

$$v_z = \frac{a_x - \dot{\omega}_x z}{\omega_x} \tag{9}$$

Substituting z with the equivalent $-v_z\tau$ gives:

$$v_z = \frac{a_x + v_z\dot{\omega}_x\tau}{\omega_x} \rightarrow v_z = \frac{a_x}{\omega_x - \dot{\omega}_x\tau}. \tag{10}$$

Given the three accelerations (a_x, a_y, a_z) and the visual observables $(\tau, \omega_x, \omega_y)$, there are three different estimates of v_z. The v_z measurements can be very different from each other, depending on the context. In particular, Eqq. 6 and 10 both become ill-conditioned when the accelerations approach zero ($\dot{\tau}$ then approaches -1 and $\dot{\omega}_x\tau$ approaches ω_x). Another factor of influence on the accuracy of the v_z estimates is the accuracy of the τ-estimate (figuring in both equations), which is worse at higher τ. The way in which the three v_z estimates are fused into one estimate is discussed further below.

Given a single estimate of v_z the other relevant state variables can be estimated as follows:

$$z = -v_z\tau \tag{11}$$

$$v_x = \omega_x z \tag{12}$$

$$v_y = \omega_y z \tag{13}$$

3.2. Vision algorithm

The vision algorithm to estimate τ and ω is introduced and explained in detail in [22]. The algorithm's main components are briefly discussed in this subsection. The vision processing is illustrated in Figure 2. The processing consists of two interconnected parts, illustrated with the dashed boxes ('A' and 'B'). The first part tracks visual features $\mathscr{F}$ over time, estimating the corresponding optic flow vectors. The second part processes the optic flow vectors in order to estimate the parameters of the optic flow field $\mathbf{p_U}, \mathbf{p_V}$, and consequently the time-to-contact τ and ventral flow ω_x. The estimates are fed back to improve the efficiency and performance of the first part.

3.2.1. Feature Tracking

The tracking of visual features over time is performed as follows. New features are detected in the image with the well-known algorithm of Shi and Tomasi [23]. The features are tracked to the next image with the Lucas-Kanade algorithm [24, 25]. Two methods are used to obtain accurate estimates: (1) unreliable optic flow vectors are discarded, and (2) features are followed over time with a Kalman Filter. For the first step the optic flow algorithm is applied 'backward'. If the backward flow brings the points in the second image close to their original positions in the first image, the flow vector is considered as reliable. If the backward flow results in an image coordinate that is too far from the original position, it is regarded as unreliable and removed. In the experiments, the maximal accepted distance is 2 pixels. The second step is to track features over multiple frames, while improving the estimate of their 'state' with a Kalman filter. A feature's state is a vector: $\mathbf{s} = (X, Y, U, V)^\intercal$, with (X, Y) the feature's image coordinate, and (U, V) the feature's optic flow.

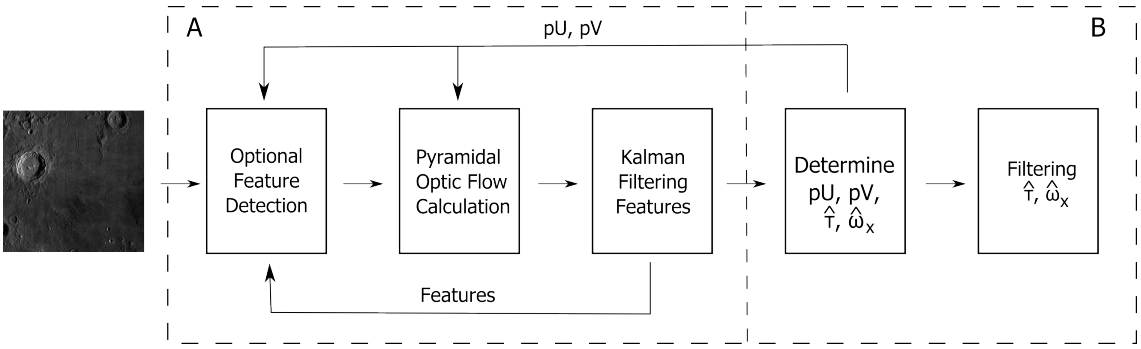

Figure 2. Overview of the vision processing. The vision processing can be subdivided in two parts. The first part tracks visual features over time, leading to a set of reliable optic flow vectors. The second part processes these vectors in order to estimate parameters of a planar optic flow field pU, pV (see the text for further details). These parameters allow the calculation of τ and ω_x over time. The parameters pU and pV are fed back to improve the efficiency and performance of the first part.

3.2.2. Estimation of τ and ω_x

The vision algorithm used in the experiments estimates the TTC by determining the divergence of the optic flow field. The algorithm assumes (1) a downward looking camera, (2) that the part of the landing surface in sight is predominantly planar, and (3) that camera rotations are either not present (as with a gimballed camera) or are accounted for by means of proprioception (viz. derotation with the help of gyrometers).

The algorithm is based on the findings in [26]. As in that study, an image plane model for the camera will be assumed. The following notation will be used. Image coordinates of an imaged point p are denoted as $P = (X,Y)$, optic flow as (U,V), and the spatial derivatives of optic flow as U_X, U_Y, V_X, and V_Y. Translational velocity is expressed as $\mathbf{v} = (v_x, v_y, v_z)$. Furthermore, point p_C is the point on the landing surface that is located in the image center $P_C = (0,0)$. The distance to p_C is h. The camera only provides information on the normalized velocity: $\vartheta = (\vartheta_x, \vartheta_y, \vartheta_z) = \mathbf{v}/h$. Finally, the inclination of the surface around p_C is represented by z_x, z_y.

Assuming no camera rotation, the above setting leads to the following formulas for the optic flow (from [26]):

$$U = U(C) + U_X X + U_Y Y \tag{14}$$

$$V = V(C) + V_X X + V_Y Y \tag{15}$$

, with:

$$U(C) = -\vartheta_x, \quad V(C) = -\vartheta_y \tag{16}$$

$$U_X = -\vartheta_z + \vartheta_x z_x, \quad V_Y = -\vartheta_z + \vartheta_y z_y \tag{17}$$

$$U_Y = \vartheta_x z_y, \quad V_X = \vartheta_y z_x \tag{18}$$

The divergence at p_C can then be written as:

$$\mathrm{div}(p_C) = U_X + V_Y = -2\vartheta_z + \vartheta_x z_x + \vartheta_y z_y. \tag{19}$$

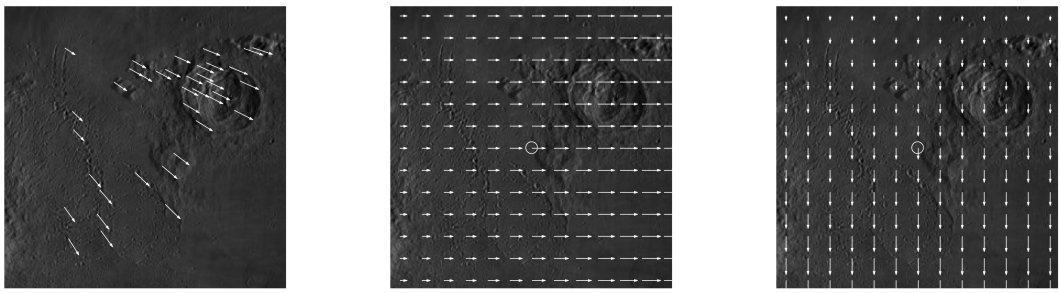

Figure 3. **Left:** Measured optic flow vectors. **Center:** Estimated horizontal optic flow field. The circle indicates the horizontal ventral flow ω_x. **Right:** Estimated vertical optic flow field. The circle indicates the vertical ventral flow ω_y.

The divergence is reciprocally related to the TTC (denoted as τ). For example, in the case of $z_x = z_y = 0$:

$$\tau = \frac{1}{-\vartheta_z} = \frac{2}{\mathrm{div}(p_C)} = \frac{2}{U_X + V_Y}, \tag{20}$$

with τ the time at which the camera center will touch the surface if the velocity v_z stays the same. With nonzero $\vartheta_x z_x$ or $\vartheta_y z_y$, the divergence can still be regarded reciprocal to a time-to-contact, if the surface around P_C is assumed to extend to the point in which it intersects the direction of motion.

As stated before, the algorithm introduced here to estimate the TTC assumes the landing surface in sight to be predominantly planar. It uses the features $\mathscr{F}_i = (X_i, Y_i, U_i, V_i)$ to approximate the horizontal and vertical optic flow with two planar equations. Figure 3 illustrates this process. The left image contains a set of observed optic flow vectors at different locations ($\tau = 2$s). The spacecraft is moving to the top left, while descending toward the surface. The center image in Figure 3 shows the planar approximation of the horizontal optic flow field, while the right image shows the approximation for the vertical optic flow field.

More formally, the algorithm estimates the parameters $\mathbf{p_U}$, $\mathbf{p_V}$ of the equations:

$$U = U(C) + U_X X + U_Y Y = (1, X, Y)\mathbf{p_U}^\mathsf{T}, \tag{21}$$

$$V = V(C) + V_X X + V_Y Y = (1, X, Y)\mathbf{p_V}^\mathsf{T}. \tag{22}$$

Having the parameter vectors $\mathbf{p_U}$, $\mathbf{p_V}$ permits the calculation of the divergence and therefore the estimation of the TTC:

$$\hat{\tau} = \frac{2}{\mathrm{FPS}(U_X + V_Y)} \tag{23}$$

, with FPS the number of frames per second. The horizontal and vertical ventral optic flow are determined with the help of $U(C)$ and $V(C)$, i.e., the optic flow in pixels at the center of the image. In Figure 3 a circle indicates the ventral optic flow vectors. The estimate of the horizontal ventral flow $\widehat{\omega_x}$ can then be determined as:

$$\widehat{\omega_x} = \mathrm{FPS}\,\frac{2U(C)}{w}\tan(\mathrm{FOV}/2) \tag{24}$$

, with w the width of the image in pixels, and FOV the horizontal field of view of the camera in degrees.

As mentioned above, estimation of the parameters in Eqq. 21 and 22 is done with the help of the set of features $\mathscr{F} = \{(U_1, V_1, X_1, Y_1), (U_2, V_2, X_2, Y_2), \ldots (U_N, V_N, X_N, Y_N)\}$. The algorithm finds least-squares solutions to the systems:

$$\mathbf{U} = \mathbf{A}\,\mathbf{p}_\mathbf{U}, \quad \mathbf{V} = \mathbf{A}\,\mathbf{p}_\mathbf{V}, \tag{25}$$

where $\mathbf{U}$ is an $N \times 1$ vector consisting of all U_i, $\mathbf{V}$ a $N \times 1$ vector consisting of all V_i, $\mathbf{A}$ is an $N \times 3$ matrix with rows $(X_i, Y_i, 1)$, and $\mathbf{p}_\mathbf{U}, \mathbf{p}_\mathbf{V}$ are 3×1 plane parameter vectors. In order to be robust to both noise and deviations from the planar assumption, a RANSAC procedure [27] is used for both fits. The RANSAC settings are to use 5 flow vectors for a fit and to perform 20 iterations. The parameters with lowest error on all features $\mathscr{F}$ are selected. If there are too few points or if all the points are colinear, the estimates $\widehat{\omega_x}$ and $\hat{\tau}$ are considered to be erroneous. Both estimates $\hat{\tau}$ and $\widehat{\omega_x}$ are filtered over time.

Finally, $\hat{\dot{\tau}}$ and $\hat{\dot{\omega}}$ are computed by making a linear least-squares error fit through FPS previous estimates of the TTC. The linear fit introduces additional delay, but is necessary to obtain less noisy estimates of $\dot{\tau}$ and $\dot{\omega}_x$.

3.3. Data fusion

As mentioned before, no initial estimate of the spacecraft state is employed. Instead, the spacecraft uses a period of 2 seconds to (i) initialize the visual measurements ($1s$), and (ii) initialize the state estimate ($2s$). The second phase of the initialization requires acceleration of the spacecraft. Therefore, the control during the initialization period can involve a free fall (in case of a planetary landing) or a specific thrust maneuver (in case of an asteroid landing).

After initializing the filters on τ and ω, the state is estimated on the basis of equations 6 and 10. The current implementation of the initial estimate assumes the three different estimates $\hat{v_z}_{1:3}$ to be statistically independent and distributed according a normal distribution: $p(\hat{v_z}_i|v_z) \sim \mathscr{N}(\mu, \sigma)$. The parameters of the normal distribution are assumed to be $\mu = v_z$ and σ a function of the relevant accelerometer reading a:

$$\sigma(a) = \begin{cases} 100 - 22.5a^2 & \text{if } |a| \leq 2 \\ 16 - 3a & \text{if } 2 < |a| \leq 5 \\ 1 & \text{if } |a| > 5, \end{cases} \tag{26}$$

a formula tuned on the basis of preliminary experiments. Then the maximal log likelihood estimate is used as v_z:

$$\hat{v_z} = \frac{\frac{1}{\sigma_1^2} v_{z1} + \frac{1}{\sigma_2^2} v_{z2} + \frac{1}{\sigma_3^2} v_{z3}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} + \frac{1}{\sigma_3^2}} \tag{27}$$

At the end of initialization, the median of the state estimates during the initialization is used as initial estimate for a Kalman filter.

The Kalman filter involves a state vector of the form: $\mathbf{s} = \langle x, v_x, a_x, y, v_y, a_y, z, v_z, a_z \rangle$. The measurement vector is $\mathbf{o} = \langle z, v_x, v_y, v_z, a_x, a_y, a_z \rangle$. The measurement variances of the velocities are set to $R_v = 1/(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} + \frac{1}{\sigma_3^2})$, while the measurement variance of the height is set to $R_z = 100R_v$. The variance on the accelerometer measurements is assumed to be $R_a = 0.1$. The process variance for the height is set to $Q_z = 2.5$, while the variance for the speeds is $Q_v = 0.5$ and for the accelerations is $Q_a = 0.01$. A new $\hat{v_z}$-measurement is considered an outlier if it is more than $50m/s$ away from the current filtered estimate. In that case, the Kalman innovation and update steps are not performed, but the uncertainty is propagated to the next time step.

4. Experimental setup

In order to test the concepts introduced in the previous sections, experiments are performed in simulation. The simulator involves a simple dynamics model of the equations of motion for the spacecraft (Eq. 2).

The generation of camera images is handled by creating views of a large base-image representing a flat ground surface. In order to employ visually realistic texture, the publicly accessible large image stitch from the Lunar Reconnaissance Orbiter Camera (LROC) is used for the base-image[2]. From the image stitch the center area of 15000×15000 pixels is selected, since it has limited perspective effects. The image of the center area has been resized to 5000×5000 pixels for use in the experiments. The settings for the virtual camera are rather conservative, with a low number of frames per second (FPS $= 10$), relatively small image size (256×256 pixels), and a field of view (FOV $= 50^\circ$) that leads to a relatively small ratio of pixels / degree.

Although the images are generated artificially and rotations are assumed to be taken care of by means of gyro derotation, the vision problem is rather challenging. For example, the initial conditions in the Apollo scenario imply a $\tau \approx 52s$, so the vision algorithm has to estimate the time-to-contact in the order of τFPS $= 520$ frames. The divergence for this τ is 0.0038, which means that two horizontal optic flow vectors that are 128 pixels apart have an optic flow difference of only 0.245 pixels. Such accurate readings can be difficult to extract from the images, as they are sometimes only sparsely textured. In addition, the control is further complicated by the necessary filtering of vision signals, which introduces a delay that can approximate one second [22].

The settings for the optimal control algorithm are as follows. The number of nodes is 10, and the algorithm is executed at 10 Hz. The quantity optimized is the final mass of the spacecraft, under the constraint that at $z = 10$ [m], $v_x = 0$ [m/s] and $v_z \in [-2.5, 0]$ [m/s].

5. Results

The spacecraft successfully lands in the lunar scenario. Here the results for a landing are shown with a final mass of 8853.51 kg. The mass expenditure is 618.55 kg, which is only 23.15 kg more than the optimal mass expenditure, 595.4 kg. In Section 2., it was shown that utilizing only 10 nodes for control optimization leads to a relatively small mass loss of 0.4 kg. Therefore the difference

[2] `http://lroc.sese.asu.edu/`

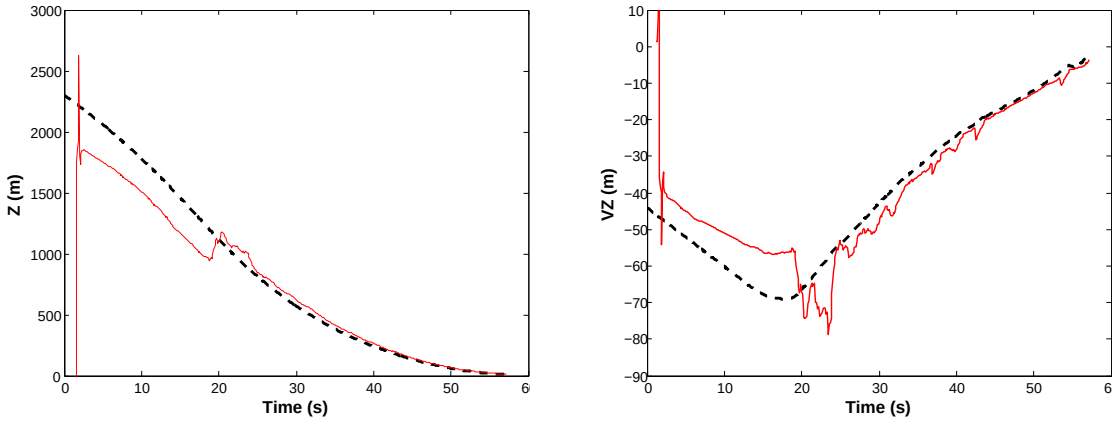

Figure 4. **Left:** $\hat{z}$ (red solid line) and z (black dashed line) over time. **Right:** $\widehat{v_z}$ (red solid line) and v_z (black dashed line) over time.

between the optimal mass expenditure and the one obtained in the experiments, mostly lies in the noise on the inertially aided, vision-based state estimates.

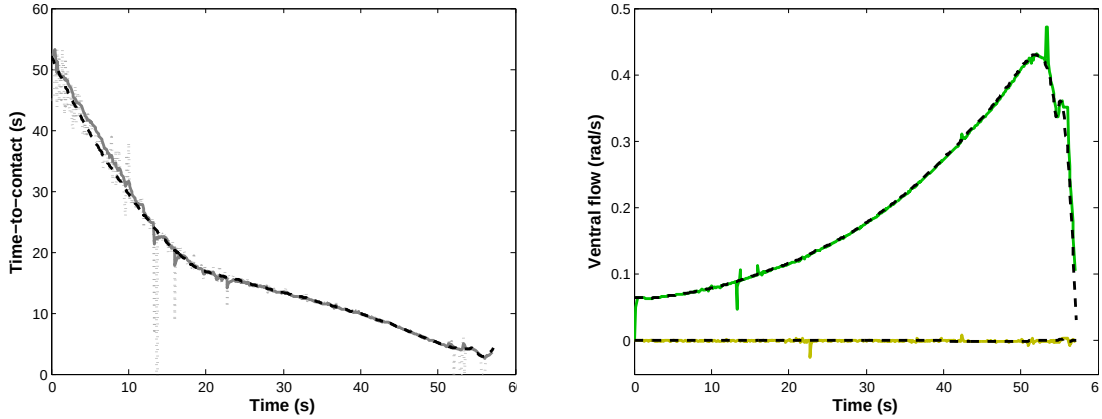

Figure 5. **Left:** Ground-truth time-to-contact (bold black dashed line), instantaneous $\hat{\tau}$ (grey dotted line) and filtered $\hat{\tau}$ over time. **Right:** Ventral flow estimates $\widehat{\omega_x}$ (green line) and $\widehat{\omega_y}$ (orange line) over time, with their corresponding ground-truths (black dashed lines).

Figure 4 shows the height (left) and vertical velocity (right) over time. Estimates are shown with

solid red lines, the ground-truth values with dashed black lines. The initial estimates using equation 27 are rather noisy, but they provide a sufficiently accurate state estimate for initialization of the Kalman filter. Initial estimates are typically different from the ground-truth in the order of 10-20%. The main observation from Figure 4 is that the state estimates (red lines) first are rather far from the ground truth values (black dashed lines), in the order of 20%. The spacecraft underestimates both the height and vertical velocity (although their relation is rather accurate). On the basis of these estimates the optimal control decides not thrust. As soon as the spacecraft starts thrusting, at $\sim 22s$, the estimates drastically improve, differing only a few meters per second from the ground-truth.

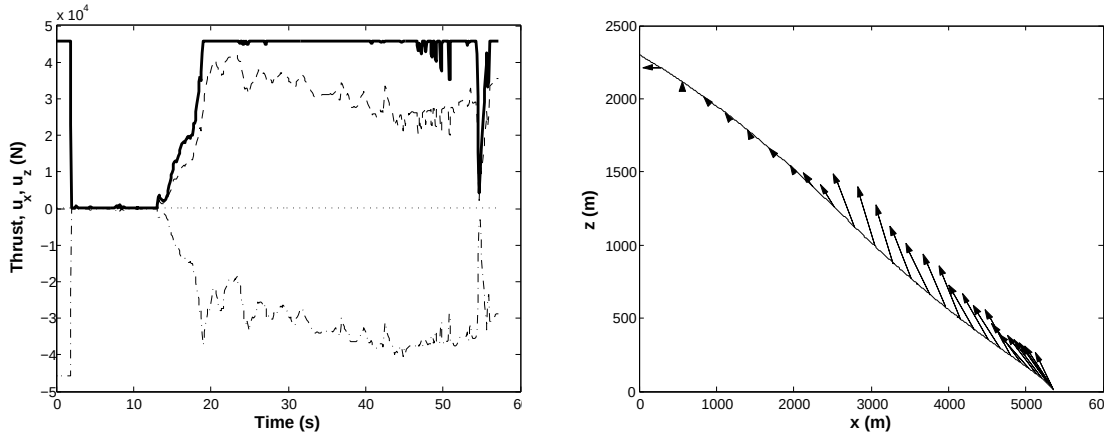

Figure 6. **Left:** the total thrust (bold solid line), u_z (dashed line), and u_x (dotted-dashed line) over time. **Right:** (x,z)-trajectory of the spacecraft. The arrows indicate the thrust directions during the trajectory (only for thrusts ≤ 1000 [N]).

Figure 5 shows the visual measurements (solid lines) and the respective ground-truths (dashed lines) over time. The left plot shows the instantaneous τ-measurements (dotted line) and the filtered estimates (dashed line) over time. The right plot shows the estimated ventral flow $\widehat{\omega_x}$ (green line) and $\widehat{\omega_y}$ (orange line). All visual observables are close to their ground-truth values.

Finally, Figure 6 shows the thrust magnitudes over time (left) and the (x,z)-trajectory with thrust directions (right). The main observation is the obvious (but noisy) bang-bang strategy to obtain an optimal mass solution: the optimal control starts the landing with a free fall, and then thrusts fully.

The advantages of having onboard real-time optimal control include the application of the approach to any (feasible) initial condition and the adaptation of the *guidance profile* to possible large disturbances. To illustrate such advantages, the simulated spacecraft has been applied to a lunar landing with entirely different initial conditions: $z_0 = 800m$, $v_{z0} = -30m/s$, $v_{x0} = 80m/s$. The results of this experiment can be seen in Figure 7 and 8.

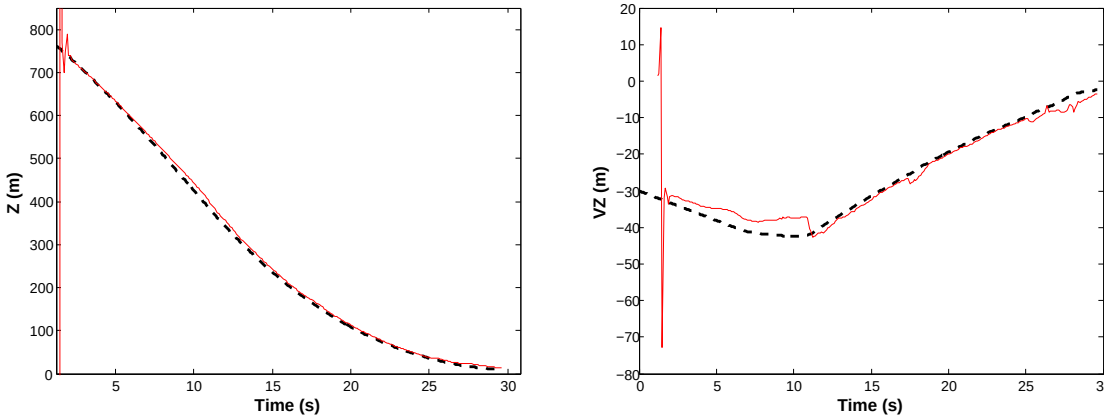

Figure 7. **Left:** $\hat{z}$ (red solid line) and z (black dashed line) over time. **Right:** $\widehat{v_z}$ (red solid line) and v_z (black dashed line) over time.

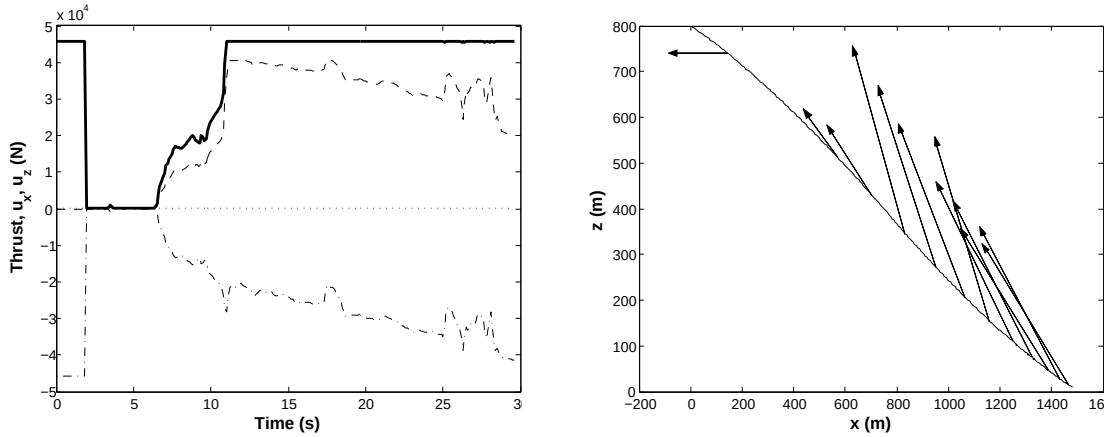

Figure 8. **Left:** the total thrust (bold solid line), u_z (dashed line), and u_x (dotted-dashed line) over time. **Right:** (x,z)-trajectory of the spacecraft. The arrows indicate the thrust directions during the trajectory (only for thrusts ≤ 1000 [N]).

6. Conclusions

The new proposed guidance and navigation architecture is able to successfully land in full autonomy in our simulations. The estimated propellant penalty associated to its use (4% in the simulated Apollo scenario) and its robustness, inherited from the property of the state-feedback used which is based on optimality principles, makes it an interesting and viable alternative to modern guidance and navigation methods. After a coarse initial estimate at the start of the trajectory, a Kalman filter further refines the estimated state over time. In the simulation experiments, the state-feedback successfully uses these state estimates to reach the end conditions with acceptable speeds and close-to-optimal mass expenditure.

Future work will involve further validation of the inertially aided vision-based state estimates. For example, it is interesting to investigate how the estimate accuracy depends on accelerations of the spacecraft. This may obviously have implications for the method's application to different gravity conditions. Moreover, future work will concern further investigation in the use of the state feedback proposed and on its performance in the presence (noisy) state estimates. Besides mass-optimal trajectories, time-optimal tajectories and pinpoint landings are also of interest. These different objectives can all be studied in the same framework.

7. References

[1] Preiss, R. "Motion parallax and figural properties of depth control flight speed in an insect." Biological Cybernetics, Vol. 57, No. 1–2, pp. 1–9, 1987.

[2] Srinivasan, M., Zhang, S., Lehrer, M., and Collett, T. "Honeybee Navigation en Route to the Goal: Visual Flight Control and Odometry." The Journal of Experimental Biology, Vol. 199, pp. 237–244, 1996.

[3] Baird, E., Srinivasan, M., Zhang, S., and Cowling, A. "Visual control of flight speed in honeybees." The Journal of Experimental Biology, Vol. 208, No. 20, pp. 3895–3905, 2005. doi:10.1242/jeb.01818.

[4] Baird, E., Srinivasan, M., Zhang, S., Lamont, R., and Cowling, A. "Visual control of flight speed and height in the honeybee." Lecture notes in computer science, pp. 40–51, 2006.

[5] Lee, D. "A theory of visual control of braking based on information about time-to-collision." Perception, Vol. 5, No. 4, pp. 437–459, 1976.

[6] Lee, D., Davies, M., Green, P., and Weel, F. v. "Visual control of velocity of approach by pigeons when landing." The Journal of Experimental Biology, Vol. 180, No. 1, pp. 85–104, 1993.

[7] Valette, F., Ruffier, F., Viollet, S., and Seidl, T. "Biomimetic optic flow sensing applied to a lunar landing scenario." "Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)," pp. 2253 – 2260. 2010.

[8] Izzo, D., Weiss, N., and Seidl, T. "Constant-Optic-Flow Lunar Landing: Optimality and Guidance." Journal of Guidance, Control, and Dynamics, Vol. 34, No. 5, pp. 1383–1395, 2011. doi:10.2514/1.52553.

[9] Izzo, D. and de Croon, G. "Landing with time-to-contact and ventral optic flow estimates." Journal of Guidance, Control, and Dynamics, Vol. 35, No. 4, pp. 1362–1367, 2012.

[10] Todorov, E. and Jordan, M. "Optimal feedback control as a theory of motor coordination." Nature neuroscience, Vol. 5, No. 11, pp. 1226–1235, 2002.

[11] Todorov, E. "Optimality principles in sensorimotor control." Nature neuroscience, Vol. 7, No. 9, pp. 907–915, 2004.

[12] Cheatham, D. and Bennett, F. "Apollo Lunar Module Landing Strategy." Tech. rep., Makerere University, 1966.

[13] Gill, P., Murray, W., and Saunders, M. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization." SIAM Journal on Optimization, Vol. 12, No. 4, pp. 979–1006, 2002.

[14] Wächter, A. and Biegler, L. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming." Mathematical Programming, Vol. 106, No. 1, pp. 25–57, 2006.

[15] Johnson, A., Willson, R., Cheng, Y., Goguen, J., Leger, C., Sanmartin, M., and Matthies, L. "Design through operation of an image-based velocity estimation system for Mars landing." International Journal of Computer Vision, Vol. 74, No. 3, pp. 319–341, 2007.

[16] Shuang, L., Pingyuan, C., and Hutao, C. "Vision-aided inertial navigation for pinpoint planetary landing." Aerospace Science and Technology, Vol. 11, pp. 499–506.

[17] Mourikis, A., Trawny, N., Roumeliotis, S., Johnson, A., Ansar, A., and Matthies, L. "Vision-Aided Inertial Navigation for Spacecraft Entry, Descent, and Landing." IEEE Transactions on Robotics, Vol. 25, No. 2, pp. 264–280, 2009.

[18] Sibley, G., Matthies, L., and Sukathme, G. "Sliding window filter with application to planetary landing." Journal of Field Robotics. Special Issue: Visual Mapping and Navigation Outdoors., Vol. 27, No. 5, pp. 587–608, 2010.

[19] Expert, F., Viollet, S., and Ruffier, F. "Outdoor field performances of insect-based visual motion sensors." Journal of Field Robotics, Vol. 28, No. 4, pp. 529–541, 2011.

[20] Ruffier, F. and Franceschini, N. "Aerial robot piloted in steep relief by optic flow sensors." "IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)," pp. 1266–1273. 2008.

[21] Hérissé, B., Hamel, T., Mahony, R., and Russotto, F.-X. "Landing a VTOL Unmanned Aerial Vehicle on a moving platform using optical flow." IEEE Transactions on Robotics, Vol. 28, No. 1, pp. 77–89, 2012.

[22] de Croon, G., Alazard, D., and Izzo, D. "Guidance, navigation, and control of optic-flow based spacecraft landing." submitted.

[23] Shi, J. and Tomasi, C. "Good features to track." "9th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)," pp. 593 – 600. 1994.

[24] Lucas, B. and Kanade, T. "An iterative image registration technique with an application to stereo vision." "Proceedings of Imaging understanding workshop," pp. 121–130. 1981.

[25] Bouguet, J.-Y. "Pyramidal Implementation of the Lucas Kanade Feature Tracker. Description of the algorithm." Intel Corporation Microprocessor Research Labs, OpenCV Documents, Vol. 3, pp. 1–9, 2000.

[26] Longuet-Higgins, H. and Prazdny, K. "The interpretation of a moving retinal image." Proceedings Royal Society London B, Biological Sciences, Vol. 208, No. 1173, pp. 385–397, 1980.

[27] Fischler, M. and Bolles, R. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." Communications of the ACM, Vol. 24, No. 6, pp. 381–395, 1981.