Nonlinear model predictive control applied to vision-based spacecraft landing

Dario Izzo and Guido de Croon

Abstract Real-time optimal control has eluded practical implementation for most systems so far. The reason being mainly related to the scarce computational resources available and the high CPU requirements of commonly proposed real-time optimal control architectures. In this paper we show how, by a careful use of the Nonlinear Model Predictive Control approach one can obtain a real-time control system able to drive a mass optimal spacecraft landing in the presence of highly noisy navigation inputs such as those coming from a light weight solution including only one IMU and a camera. The introduced approach is applicable to a broader class of systems, as is shown by applying the method to find time-optimal maneuvers for a quad rotor model.

1 Introduction

In a typical control architecture an optimal guidance profile is determined and fed as a reference signal to the navigation and control loop that will thus track such a profile as closely as possible during operation. This scheme is widely used for example in mass-optimal interplanetary trajectories, in spacecraft landing [6, 22] or in time-optimal trajectories for agile quad rotors maneuvers [20, 11] to only quote a few. The advantage of such a scheme is in the possibility to determine the optimal guidance profile upfront thus coping with the computational effort that is typically involved in the optimal guidance definition. While simple vehicle / world models may be solved in closed form, more elaborate problems require solving a complex

Dario Izzo
Advanced Concepts Team, European Space Agency, Noordwijk, the Netherlands e-mail: dario.izzo@esa.int

Guido de Croon
Advanced Concepts Team, European Space Agency, Noordwijk, the Netherlands e-mail: guido.de.croon@esa.int

and highly non-convex optimization problem [6]. One important factor leading to a large computational effort is that optimal guidance profiles often have discontinuous changes of control variables (bang-bang controls). This means that the optimization needs to employ a high time resolution (small time steps), which slows down the whole process even further.

The result of the optimization process is a guidance profile valid for the considered initial state. For spacecraft landing, considerable effort in terms of mass, sensor redundancy, etc. is invested in order to meet the initial state as closely as possible and consequently follow it closely. Too far an offset from the initial or an intermediary state may lead to an unacceptable cost of propellant mass. While this may be acceptable in a single, well-planned landing, this is not the case in general as retargeting and replanning is often important and large offsets from the nominal trajectory may arise. This is especially true for vehicles other than complex and costly spacecraft, for instance, a quad rotor is an agile platform that during its operation has to perform many different maneuvers, having many different optimal guidance profiles. Storing a large set of optimal guidance profiles would be an unwieldy solution, especially given the limited resources onboard such small platforms.

In some cases, it is possible to determine a guidance profile (optimal in some sense) in real-time, updating it at every sampling instant. In particular, in Model Predictive Control (MPC), a linear model is solved at every time step (often analytically). For highly nonlinear systems one can linearize around the current state, but this imposes a short time horizon and can result in extremely sub-optimal solutions. An alternative solution is to employ Nonlinear Model Predictive Control (NMPC) [10, 1, 21], in which the nonlinear model is used in the optimization. Although there is significant progress in the field of NMPC, the computational effort required to optimize nonlinear models still entails that NMPC is mostly used for systems with relatively slow dynamics such as chemical plants [10] or long-lasting low-thrust spacecraft maneuvers [1]. In addition, many important issues in the field of NMPC are still open, including its robustness in the presence of (noisy) state estimation.

The main contribution of this paper lies in devising a real-time optimal control (NMPC) architecture and demonstrating its performance in systems of interest in aerospace engineering, characterized by a rather fast dynamics and noisy sensory inputs. Our approach makes the application of NMPC to such systems possible by using (1) a Hermite-Simpson collocation method to transform the optimal control problem (OCP) into a Nonlinear Programming problem (NLP) [4], (2) an interior point optimization technique (from the open source package IPOPT [29]) with gradients computed via automated differentiation, and (3) a careful selection of the simplified model used internally to plan optimally.

The approach is studied first in the case of perfect state measurements and then, more interestingly, in the case of state estimation based on noisy measurements. In particular, we consider an inertially aided vision-based spacecraft landing scenario. The spacecraft includes only a downward pointing camera and three accelerometers. We show how our approach is able to cope with such an arguably limited sensing capability by continuous and fast replanning of the optimal actions.

The remainder of the article is organized as follows. First, in Section 2, the relation between model complexity, computational effort, and optimality is investigated. In particular, three (increasingly complex) spacecraft models and a quad rotor model are studied for different time resolutions. Second, the robustness of the approach to navigation errors is investigated. To this end, the novel scheme is applied to the aforementioned vision-based spacecraft landing scenario. The experimental setup of the simulations is discussed in Section 3, while the results are investigated in Section 4. Finally, conclusions are drawn in Section 5.

2 Nonlinear model predictive control implementation

Let us consider the problem of real-time optimal control of complex systems. We define this as the problem of finding a "computationally viable" form of the optimal state feedback $\mathbf{u} = \mathbf{u}^*(\mathbf{x})$. Taking the generic form of a finite horizon optimal control problem, see for example [28], seeking to minimize the Mayer cost functional and considering a rich enough functional space (Sobolev spaces $W^{n,m}$ are typically needed), optimal control theory guarantees us the existence and uniqueness of such a state feedback. One is then left with the problem of finding a "computationally viable" algorithm to compute $\mathbf{u} = \mathbf{u}^*(\mathbf{x})$ (analytical solutions are very rare and available only for simple problems). In this paper we use ideas from nonlinear model predictive control [10]: we build a procedure to solve the optimal control problem for a simpler dynamical representation of the system from a generic initial state x_0 thus finding $\mathbf{u}^*_{x_0}(t)$. We then set at each time instant $\mathbf{u}^*(\mathbf{x}) = \mathbf{u}^*_x(0)$. In our scheme we then need to reconstruct the control to be fed into the real plant as the model plant mismatches, here intentionally introduced, makes it impossible to use the computed $\mathbf{u}^*(\mathbf{x})$ directly. In this paper, though, we do not discuss this last step which is the subject of a future work, and we focus on the rest of our real-time nonlinear model predictive control approach and its coupling with our vision based navigation. In other words, in this paper, the model plant and the real plant will be assumed to be coincident.

The main problem with non-linear model predictive control is the low frequency at which one is able to compute $\mathbf{u}^*(\mathbf{x})$. Solving the optimal control problem, in fact, entails the solution of a nonlinearly constrained high dimensional nonlinear programming (NLP) problem (assuming that a direct approach is employed). These types of problems require quite intensive computational resources and, in any case, a very good initial guess to ensure their convergence. We here overcome this problem by using simplified nonlinear systems tailored at computational efficiency as internal models (as mentioned above this will introduce a plant mismatch that has to be dealt with by another block in the overall control architecture). The idea is rather simple and applicable more in general to robotics systems: the system thinks of itself and of the environment in simplified terms when planning its optimal actions, the continuous update of such a plan ensures that the gap between the theoretical

optimal action and the one planned is filled. The scheme also allows for frequent changes of higher level goals which simply results in a replanned course of action.

The very feasibility of our approach relies on the computational efficiency of the optimal control solver which needs to be able to compute the optimal action at a high frequency as to allow replanning and accounting for the differences between model and reality. We here study four simple cases of interest to the engineering community as they can represent internal models of widely studied systems such as quadrotors or spacecraft. The first three cases model a spacecraft landing at different levels of complexity, a problem that will also be the main case study presented in the rest of the paper. In particular, we consider a "point with variable mass model" (model I):

$$\begin{aligned} &\dot{x} = v_x, \quad \dot{v}_x = u\sin(u_\theta)\cos(u_\phi)/m \\ &\dot{y} = v_y, \quad \dot{v}_y = u\sin(u_\theta)\sin(u_\phi)/m \\ &\dot{z} = v_z, \quad \dot{v}_z = u\cos(u_\theta)/m - g \\ &\dot{m} = -u/I_{sp}g_0 \end{aligned} \tag{1}$$

a "point with variable mass and pitch angle controlled via a reaction wheel" model (model II):

$$\begin{aligned} &\dot{x} = v_x, \quad \dot{v}_x = u\sin(\theta)/m \\ &\dot{z} = v_z, \quad \dot{v}_z = u\cos(\theta)/m - g \\ &\dot{\theta} = u_\theta \\ &\dot{m} = -u/I_{sp}g_0 \end{aligned} \tag{2}$$

and a "point with variable mass and pitch dynamics controlled via thrusters" (model III):

$$\begin{aligned} &\dot{x} = v_x, \quad \dot{v}_x = (u+u_L+u_R)\sin(\theta)/m \\ &\dot{z} = v_z, \quad \dot{v}_z = (u+u_L+u_R)\cos(\theta)/m - g \\ &\dot{\theta} = \omega, \quad \dot{\omega} = R(u_R - u_L) \\ &\dot{m} = -(u+u_L+u_R)/I_{sp}g_0 \end{aligned} \tag{3}$$

In contrast, the fourth case models the dynamics of a UAV (a quadrotor) as modelled in [11] (model IV):

$$\begin{aligned} &\dot{x} = v_x, \quad \dot{v}_x = u\sin(\theta)/m \\ &\dot{z} = v_z, \quad \dot{v}_z = u\cos(\theta)/m - g \\ &\dot{\theta} = u_\theta \end{aligned} \tag{4}$$

In all the above cases the controls (indicated always with the letter u) are considered as bounded and the optimal control structure is thus bang-bang. Final mass maximization is seeked for the first three models, while we consider, in the fourth case, a minimum time problem. The optimal control problem is transcribed into a nonlinear programmig (NLP) problem using the well-known Hermite-Simpson transcription [4]. To solve the NLPs arising in the various cases we make use of an interior point optimization method (IPOPT [29]) providing to it all the gradients computed with the aid of automated differentiation. For the first three models we test an Apollo-like scenario [7]. The scenario involves the following high-gate conditions at the Moon (when applicable): $I_{sp} = 311$ [s], $m_0 = 9472.06$ [kg], $v_{x_0} = 150$ [m/s], $v_{z_0} = -44$ [m/s], $z_0 = 2300$ [m], $\theta_0 = -60$ [deg] and $g = 1.623$ [m/s^2]. At

low-gate we set: $v_{x_{t_f}} = 0$ [m/s], $v_{z_f} \in [-2.5, 0]$ [m/s], $z_f <= 10$ [m] and $\theta_f =$ free [deg,]. For model I, we set $u \leq 45760$ [N], for model II we set $u \leq 45760, \geq 0$ [N] and $u_\theta \leq 0.0698, \geq -0.0698$. For model III we set $u \leq 44000, \geq 0$ [N] and $u_L, u_R \leq 80$ [N] and $R = 3$ [m]. For model IV, we use the agile quadrotor used in [11] and simulate an horizontal displacement maneuver with $x_0, z_0, v_{x_0}, v_{z_0}, \theta_0 = 0$ and $x_f = 15$ [m], $\theta_f = 2\pi$. The quadrotor has $m_0 = 1$ [kg] and we set $u \leq 20, \geq 1$ [N] and $u_\theta \leq 10, \geq -10$ [rad/s]. The tests[1] are done recording the final achieved objective function and the employed CPU time with varying number of nodes employed by the Hermite-Simpson transcription.

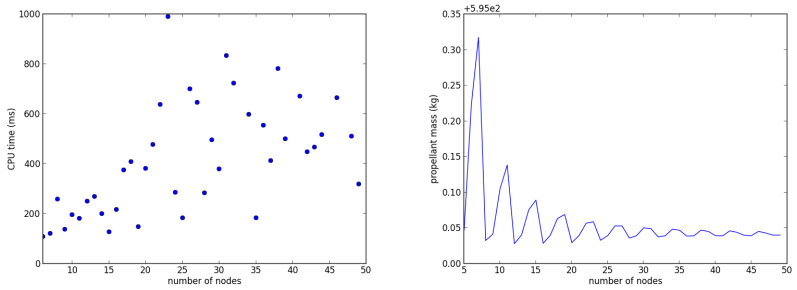

Fig. 1 CPU performance and precision in the case of model I: point with variable mass

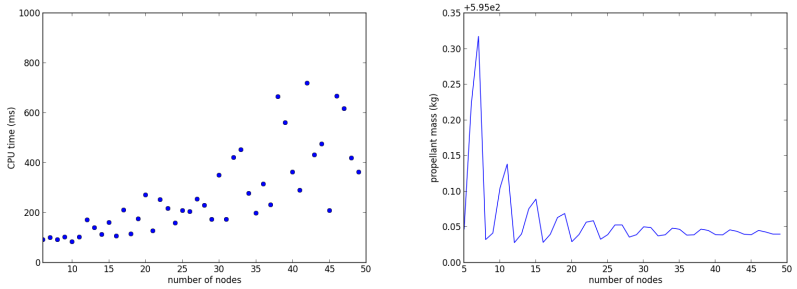

Fig. 2 CPU performance and precision in the case of model II: point with variable mass and pitch angle controlled via a reaction wheel

The results are reported in Figures 1-4. From the results it is shown how all our models can be solved with a good accuracy already using only $n = 5$ nodes. Increasing the number of nodes does improve the accuracy of the computed solution, but only marginally. Using our scheme, in all tested cases one can compute the optimal control solution with a frequency of 10 Hz and an accuracy which is within one

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

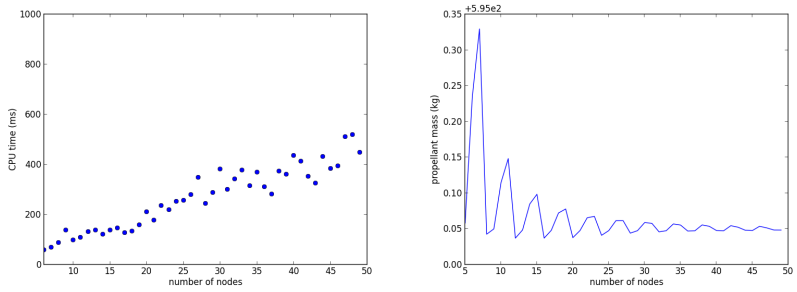

Fig. 3 CPU performance and precision in the case of model III: point with variable mass and pitch angle controlled via thrusters

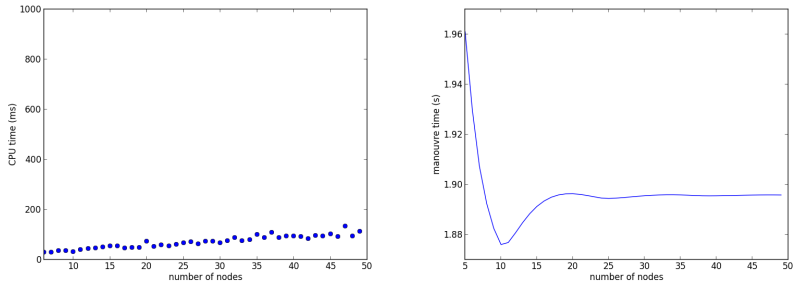

Fig. 4 CPU performance and precision in the case of model IV: a quadrotor

percent of the real optimal solution. Surprisingly, in the case of spacecraft landing, model III seems to be the most computationally efficient.

We also note how the convergence pattern with respect to the number of nodes is non monotonous in all cases. This is due to the discontinuous nature of the optimal control (bang-bang) and its interaction with an equally spaced time grid (when the control switching point overlaps to a point on the time grid a benefit on the objective function is derived). Also the CPU time has a rather scattered behaviour with model I being the most irregular and the quadrotor being the smoother. This behaviour depends on the complex interaction between the IPOPT stopping criterias, its search strategy and the different NLP problems created using the Hermite-Simpson transcription with different nodes number.

In the rest of the paper we will be using model I and show how the nonlinear model predictive control approach is able to integrate with our vision based navigation system cancelling the arguably high state etimation errors and still landing using an amount of propellant very close to the optimal value.

3 Case study: inertially aided vision-based spacecraft landing

In the previous section simplified models with a coarse time resolution were shown to lead to solutions that are close to optimal. This implies that approximate optimal solutions can be calculated in real-time on board a vehicle such as a spacecraft or quad rotor. However, it remains unclear whether the more typical guidance-tracking control of such a vehicle can be successfully supplanted by the commands directly coming out of the optimization process. For example, it is unclear whether navigation errors lead to a graceful decay or to complete failure.

In order to study the impact of navigation errors, in this section, our real-time optimal control scheme is tested on a simulated spacecraft landing scenario. The focus is on a minimal landing system that can be used as an emergency landing system of a normal spacecraft or as the main landing system of a nano-lander. The state estimates are based on a combination of an Inertial Measurement Unit (IMU) with vision, typically referred to as *vision-aided inertial navigation*. Previous studies in this area [14, 24, 18, 25] 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 biologically relevant visual cues of ventral optic flow [19, 26, 2, 3] and time-to-contact [15, 16]. These bio-inspired visual observables can be measured with extremely light-weight and energy efficient neuromorphic sensors [9] or with an uncalibrated linear camera. These cues have previously been proposed and studied in the context of spacecraft landing scenarios and mass optimality [27, 13, 12]. 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) [12, 8]. 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 [12]) can be of concern for applications where it is not affordable to mis-use such a precious resource. Below it is shown that the novel real-time optimal control scheme is both robust and significantly more mass-efficient (with a propellant mass consumption only 3.4% away from the optimal value for the same case studied in [12]).

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

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. In this section we

demonstrate how combining the ventral flow and time-to-contact with accelerometer readings allows 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:

$$v_z\tau = -z \tag{5}$$

Taking the time derivative, we get:

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

$$(1+\dot{\tau})v_z = -a_z\tau \tag{7}$$

$$v_z = -\frac{a_z\tau}{1+\dot{\tau}} \tag{8}$$

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

$$\omega_x z = v_x \tag{9}$$

Taking the time derivative gives:

$$\dot{\omega}_x z + \omega_x v_z = a_x \tag{10}$$

$$v_z = \frac{a_x - \dot{\omega}_x z}{\omega_x} \tag{11}$$

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

$$v_z = \frac{a_x + v_z\dot{\omega}_x\tau}{\omega_x} \rightarrow v_z = \frac{a_x}{\omega_x - \dot{\omega}_x\tau}. \tag{12}$$

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. 8 and 12 both become ill-conditioned when the accelerations approach zero ($\dot{\tau}$ then approaches -1 and $\dot{\omega}_x\tau$ approaches ω_x). This can easily be seen, by looking at the time derivates:

$$\dot{\tau} = \frac{a_z z}{v_z^2} - 1 \tag{13}$$

$$\dot{\omega}_x = \frac{a_x}{z} - \frac{v_x v_z}{z^2} = \frac{a_x}{z} + \frac{\omega_x}{\tau} \tag{14}$$

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 in Subsection 3.3.

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

$$z = -v_z\tau \tag{15}$$

$$v_x = \omega_x z \tag{16}$$

$$v_y = \omega_y z \tag{17}$$

3.2 Vision algorithm

The vision algorithm to estimate τ and ω is introduced and explained in detail in [8]. The algorithm's main components are only briefly discussed in this subsection. It 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 vision processing is illustrated in Figure 5. 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. 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 [17, 5]. The second part processes the optic flow vectors in order to estimate the parameters of a planar optic flow field $\mathbf{p_U}, \mathbf{p_V}$. The time-to-contact τ is inversely related to the divergence of the optic flow field, while the ventral flow ω_x is the optic flow in the center of the optic flow field. Figure 6 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 6 shows the planar approximation of the horizontal optic flow field, while the right image shows the approximation for the vertical optic flow field. The estimates $\hat{\tau}$ and $\hat{\omega}$ are filtered over time (leading to $\hat{\tau}_f$ and $\hat{\omega}_f$) and fed back to improve the efficiency and performance of the first part.

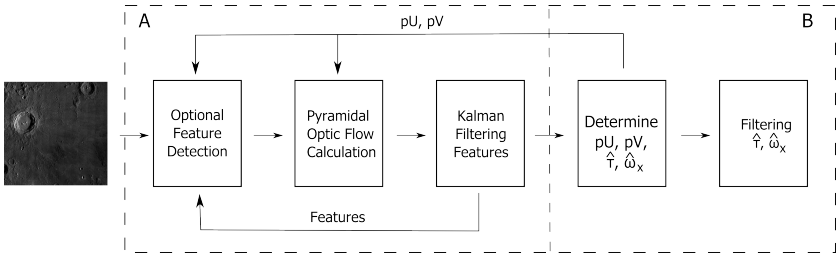

Fig. 5 Overview of the vision processing. The vision processing can be subdivided in two parts. The first part tracks visual features over time, leading to a set of reliable optic flow vectors. The second part processes these vectors in order to estimate parameters of a planar optic flow field pU, pV (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.

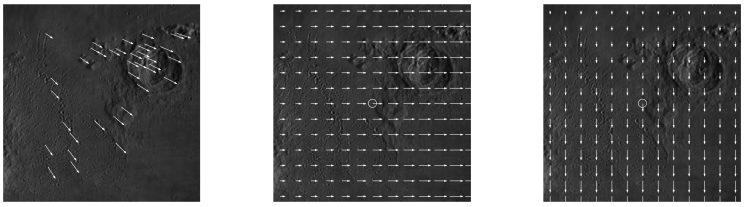

Fig. 6 Left: Measured optic flow vectors. **Center:** Estimated horizontal optic flow field. The circle indicates the horizontal ventral flow ω_x. **Right:** Estimated vertical optic flow field. The circle indicates the vertical ventral flow ω_y.

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

3.3 Data fusion and state estimation

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 ($1s$). 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 sufficient acceleration due to gravity) or a specific thrust maneuver (under low-gravity conditions).

After initializing the filters on τ and ω, the state is estimated on the basis of equations 8 and 12. 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: $p(\hat{v_z}_i|v_z) \sim \mathcal{N}(\mu,\sigma)$. The parameters of the normal distribution are assumed to be $\mu = v_z$ and σ a function of the relevant accelerometer reading a:

$$\sigma(a) = \begin{cases} 100 - 22.5a^2 & \text{if } |a| \leq 2 \\ 16 - 3a & \text{if } 2 < |a| \leq 5 \\ 1 & \text{if } |a| > 5, \end{cases} \tag{18}$$

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

$$\hat{v_z} = \frac{\frac{1}{\sigma_1^2}\hat{v_{z1}} + \frac{1}{\sigma_2^2}\hat{v_{z2}} + \frac{1}{\sigma_3^2}\hat{v_{z3}}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} + \frac{1}{\sigma_3^2}} \tag{19}$$

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.

The entire data fusion and state estimation process is illustrated in Figure 7. The state estimate of the Kalman filter is used for the NMPC.

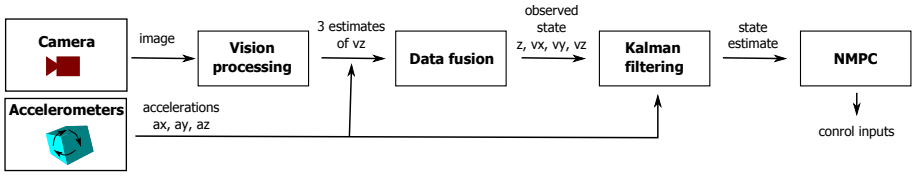

Fig. 7 Overview of how the sensor measurements (left) are processed to obtain a state estimate. This estimate is used by the nonlinear model predictive control to determine the current control inputs.

3.4 Experimental setup

Experiments are performed in simulation with model I (Eq. 1). The motivation for choosing this model is that model I captures 3D-motion as allowed by the simulator. For the experiments, the lunar scenario discussed in Section 2 is employed: $I_{sp} = 311$ [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].

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.

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

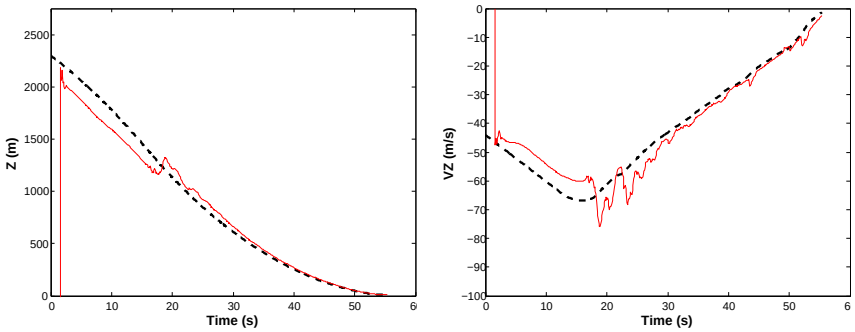

Fig. 8 Left: $\hat{z}$ (red solid line) and z (black dashed line) over time. **Right:** $\widehat{v_z}$ (red solid line) and v_z (black dashed line) over time.

Although the images are generated artificially and rotations are assumed to be absent or 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 [8].

The settings for the optimal control algorithm are as follows. Model I is used, with 10 nodes, 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].

4 Results

The spacecraft successfully lands in the lunar scenario. Here the results for a landing are shown with a final mass of 8856.92 kg. The mass expenditure is 615.14 kg, which is only 20.04 kg (3.4%) more than the optimal mass expenditure, 595.10 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 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. In addition, the end condition on v_x is not exactly met: at $z = 9.90m$ the spacecraft velocities are $v_x = -0.64$ m/s and $v_z = -1.39$ m/s. These velocities can easily be cancelled in the remaining meters.

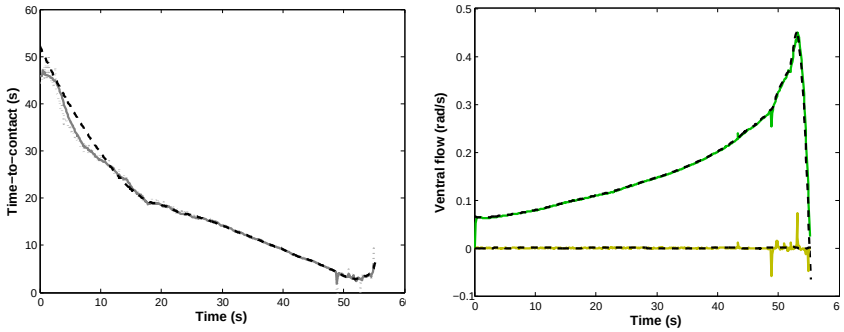

Fig. 9 Left: Ground-truth time-to-contact (bold black dashed line), instantaneous $\hat{\tau}$ (grey dotted line) and filtered $\hat{\tau}$ over time. **Right:** Ventral flow estimates $\widehat{\omega_x}$ (green line) and $\widehat{\omega_y}$ (orange line) over time, with their corresponding ground-truths (black dashed lines).

Figure 8 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 19 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 8 is that the state estimates (red lines) after initialization are in the order of 10% off from the ground truth values (black dashed lines). The spacecraft initially underestimates both the height and vertical velocity (although their relation is rather accurate). On the basis of these estimates the optimal control decides not to thrust. As soon as the spacecraft starts thrusting, at $\sim 15s$, the estimates start to further improve, differing $\sim 5\%$ from the ground-truths.

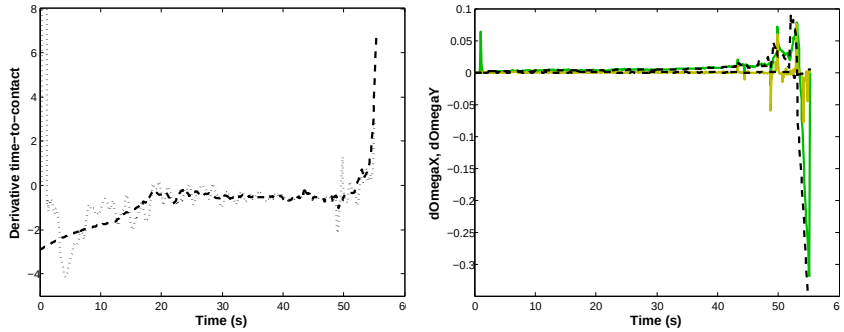

Fig. 10 Left: Ground-truth time derivative of the time-to-contact (bold black dashed line), and $\hat{\dot{\tau}}$ over time. **Right:** Estimated time derivatives of the ventral flow $\widehat{\dot{\omega}_x}$ (green line) and $\widehat{\dot{\omega}_y}$ (dark yellow line) over time, with their corresponding ground-truths (black dashed lines).

Figure 9 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), the filtered estimates (solid line), and the ground-truth τ values (dashed line) over time. The right plot shows the estimated ventral flow $\widehat{\omega_x}$ (green line) and $\widehat{\omega_y}$ (dark yellow line). All visual observables are close to their ground-truth values, although the estimates get noisier towards the end of the landing. A partial explanation for this is that close to the surface the images contain much less visual texture due to the digital image zoom.

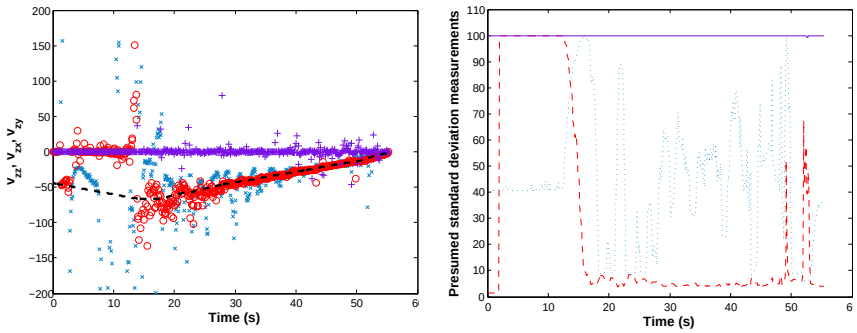

Fig. 11 Left: The three different v_z-estimates over time: based on the vertical dynamics (v_{zz}, 'x'), and based on the ventral flow ω_x (v_{zx}, 'o') and ω_y (v_{zy}, '+'). **Right:** The standard deviations of the different v_z-estimates over time: $\sigma(a_z)$ (dotted blue line), $\sigma(a_x)$ (dashed red line), and $\sigma(a_y)$ (solid purple line).

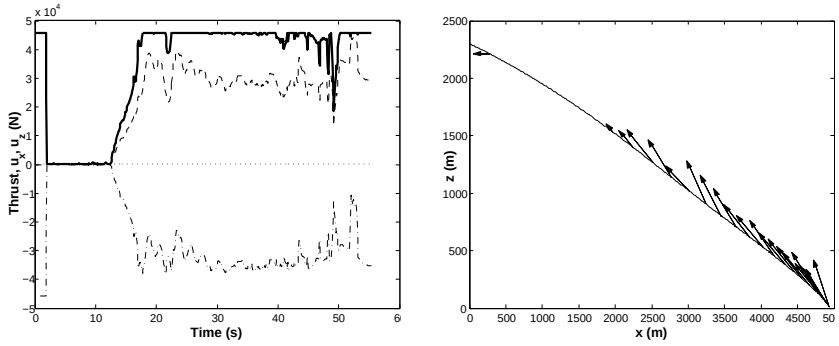

Fig. 12 Left: the total thrust (bold solid line), u_z (dashed line), and u_x (dotted-dashed line) over time. **Right:** (x,z)-trajectory of the spacecraft. The arrows indicate the thrust directions during the trajectory (only for thrusts ≥ 1000 [N]).

Figure 10 shows the estimated time derivatives of the visual observables. The left plot shows the estimated $\hat{\tau}$ over time (grey dotted line) and its ground-truth

(bold black dashed line). The right plot shows the estimated time derivatives of the ventral flow $\widehat{\omega_x}$ (green line) and $\widehat{\omega_y}$ (dark yellow line) over time, with their corresponding ground-truths (black dashed lines). At the start and at the end of the landing the estimated time derivatives are somewhat noisy. Depending on the context, this noise can lead to large estimation errors. Nonetheless, the estimates generally approximate the ground-truths reasonably well, allowing for sufficiently accurate speed estimates.

The left part of Figure 11 shows the individual estimates of v_z over time. The estimates based on the vertical axis are indicated with 'x', the ones based on the ventral flow with 'o' (ω_x) and '+' (ω_y). The right part of the figure shows the corresponding standard deviations: $\sigma(a_z)$ (dotted blue line), $\sigma(a_x)$ (dashed red line), and $\sigma(a_y)$ (solid purple line). The figures show that the initial thrust maneuver (in the opposite direction of the ventral flow) significantly helps the state estimation: around $t = 2-3$s, the estimates based on ω_x ($\widehat{v_{zx}}$) are rather accurate. In the free fall that follows, only the $\widehat{v_{zz}}$ are sometimes close to the ground truth v_z. The Kalman filter copes with the lack of reliable vision-based information during the free-fall by assigning a high standard deviation to $\widehat{v_{zx}}$ and $\widehat{v_{zy}}$, and a reasonably high standard deviation to $\widehat{v_{zz}}$. When the spacecraft starts thrusting, the estimates immediately improve and the standard deviations of the measurements are set much lower for use in the Kalman filter.

Finally, Figure 12 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 13 and 14.

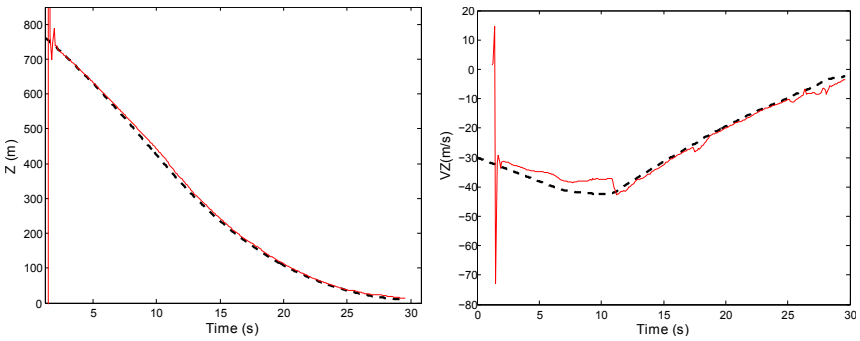

Fig. 13 Left: $\hat{z}$ (red solid line) and z (black dashed line) over time. **Right:** $\widehat{v_z}$ (red solid line) and v_z (black dashed line) over time.

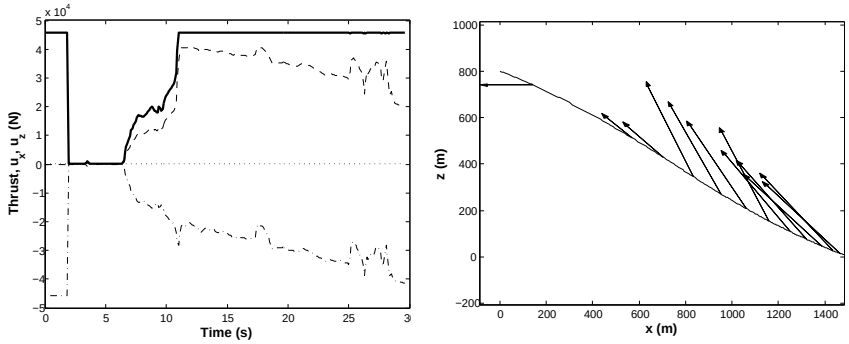

Fig. 14 Left: the total thrust (bold solid line), u_z (dashed line), and u_x (dotted-dashed line) over time. **Right:** (x,z)-trajectory of the spacecraft. The arrows indicate the thrust directions during the trajectory (only for thrusts ≥ 1000 [N]).

5 Conclusions

We have introduced an approach to Nonlinear Model Predictive Control (NMPC) in order to obtain real-time optimal control of complex systems. We succesfully tested our approach in the case of a vision-based spacecraft moon landing in presence of noisy navigation inputs. Our approach makes use of Hermite-Simpson method and interior point optimization with gradients computed via automated differentiation to achieve the requested computational efficiency. We show that the approach is able to achieve real-time performances. In our simulated landings, a vision based navigation, suitable for an emergency procedure or for a nano-lander, is used and provides noisy estimates of the state. The nonlinear model predictive control copes with the noisy state estimation and guides the whole landing using a few percents more than the optimal propellant mass.

References

1. Arrieta-Camacho, J., Biegler, L.: Real time optimal guidance of low-thrust spacecraft: an application of nonlinear model predictive control (2005)
2. Baird, E., Srinivasan, M., Zhang, S., Cowling, A.: Visual control of flight speed in honeybees. The Journal of Experimental Biology **208**(20), 3895–3905 (2005). DOI 10.1242/jeb.01818
3. Baird, E., Srinivasan, M., Zhang, S., Lamont, R., Cowling, A.: Visual control of flight speed and height in the honeybee. Lecture notes in computer science pp. 40–51 (2006)
4. Betts, J.: Practical methods for optimal control and estimation using nonlinear programming, vol. 19. Society for Industrial & Applied (2010)

5. 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)
6. Bryson, A., Ho, Y.: Applied optimal control. Wiley New York (1975)
7. Cheatham, D., Bennett, F.: Apollo lunar module landing strategy. Tech. rep., Makerere University (1966)
8. de Croon, G., Alazard, D., Izzo, D.: Guidance, navigation, and control of optic-flow based spacecraft landing (submitted)
9. Expert, F., Viollet, S., Ruffier, F.: Outdoor field performances of insect-based visual motion sensors. Journal of Field Robotics **28**(4), 529–541 (2011)
10. Findeisen, R., Allgwer, F.: An introduction to nonlinear model predictive control. In: Control, 21st Benelux Meeting on Systems and Control, Veidhoven, pp. 1–23 (2002)
11. Hehn, M., Ritz, R., DAndrea, R.: Performance benchmarking of quadrotor systems using time-optimal control. Autonomous Robots **33**(1–2) (2012)
12. Izzo, D., de Croon, G.: Landing with time-to-contact and ventral optic flow estimates. Journal of Guidance, Control, and Dynamics **35**(4), 1362–1367 (2012)
13. Izzo, D., Weiss, N., Seidl, T.: Constant-optic-flow lunar landing: Optimality and guidance. Journal of Guidance, Control, and Dynamics **34**(5), 1383–1395 (2011). DOI 10.2514/1.52553
14. Johnson, A., Willson, R., Cheng, Y., Goguen, J., Leger, C., Sanmartin, M., Matthies, L.: Design through operation of an image-based velocity estimation system for Mars landing. International Journal of Computer Vision **74**(3), 319–341 (2007)
15. Lee, D.: A theory of visual control of braking based on information about time-to-collision. Perception **5**(4), 437–459 (1976)
16. Lee, D., Davies, M., Green, P., Weel, F.v.: Visual control of velocity of approach by pigeons when landing. The Journal of Experimental Biology **180**(1), 85–104 (1993)
17. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of Imaging understanding workshop, pp. 121–130 (1981)
18. Mourikis, A., Trawny, N., Roumeliotis, S., Johnson, A., Ansar, A., Matthies, L.: Vision-aided inertial navigation for spacecraft entry, descent, and landing. IEEE Transactions on Robotics **25**(2), 264–280 (2009)
19. Preiss, R.: Motion parallax and figural properties of depth control flight speed in an insect. Biological Cybernetics **57**(1–2), 1–9 (1987)
20. Ritz, R., Hehn, M., Lupashin, S., D'Andrea, R.: Quadrocopter performance benchmarking using optimal control (2011)
21. Ross, I., Fahroo, F.: Issues in the real-time computation of optimal control. Mathematical and computer modelling **43**(9), 1172–1188 (2006)
22. Shen, H., Seywald, H., Powel, R.: Desensitizing the minimum-fuel powered descent for mars pinpoint landing. Journal of Guidance, Control and Dynamics **33**(1) (2010)
23. Shi, J., Tomasi, C.: Good features to track. In: 9th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 593 – 600 (1994)
24. Shuang, L., Pingyuan, C., Hutao, C.: Vision-aided inertial navigation for pinpoint planetary landing. Aerospace Science and Technology **11**, 499–506
25. Sibley, G., Matthies, L., Sukathme, G.: Sliding window filter with application to planetary landing. Journal of Field Robotics. Special Issue: Visual Mapping and Navigation Outdoors. **27**(5), 587–608 (2010)
26. Srinivasan, M., Zhang, S., Lehrer, M., Collett, T.: Honeybee navigation en route to the goal: Visual flight control and odometry. The Journal of Experimental Biology **199**, 237–244 (1996)
27. Valette, F., Ruffier, F., Viollet, S., Seidl, T.: Biomimetic optic flow sensing applied to a lunar landing scenario. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2253 – 2260 (2010)
28. Vinter, R.: Optimal control. Birkhäuser Boston (2010)
29. Wächter, A., Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)