Event-based spacecraft landing using time-to-contact

Olaf Sikorski, Dario Izzo, Gabriele Meoni
Advanced Concepts Team, European Space Research and Technology Centre
Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands
olaf.sikorski@gmail.com
{dario.izzo, gabriele.meoni}@esa.int

Abstract

We study event-based sensors in the context of spacecraft guidance and control during a descent on Moon-like terrains. For this purpose, we develop a simulator reproducing the event-based camera outputs when exposed to synthetic images of a space environment. We find that it is possible to reconstruct, in this context, the divergence of optical flow vectors (and therefore the time to contact) and use it in a simple control feedback scheme during simulated descents. The results obtained are very encouraging, albeit insufficient to meet the stringent safety constraints and modelling accuracy imposed upon space missions. We thus conclude by discussing future work aimed at addressing these limitations.

1. Introduction

Conventional strategies for autonomous spacecraft landing rely on velocity and altitude estimation using, for example, laser or radar altimeters for guidance. Despite the robustness of those methods, there has been much interest in lightweight and low-power solutions based on alternative neuromorphic sensors [32, 33] coupled to emerging computer vision algorithms particularly suitable for future smaller crafts [18].

Visual guidance is commonly observed in nature, insects for example are extremely effective at safe landing in a variety of dynamically changing scenes [3, 31]. This fact has drawn much interest from researchers investigating both the structure of the sensor and the control strategy employed by living creatures leading to some of the findings being translated to bio-inspired systems, both at the software and hardware level. This study builds on two particular developments in the field of bio-inspired, neuromorphic descents and specifically event-based sensors for visual data [28] and time-to-contact (TTC) landing [17].

Event-based cameras (or event cameras) are a relatively novel class of vision sensors that introduced a whole paradigm shift in visual information acquisition [12, 28]. As opposed to conventional cameras, these devices completely depart from the concept of frames sampled at regular intervals and instead work by asynchronously reporting singular events registered within the observed scene. The output is a stream of events each consisting of an x, y address of the pixel on the imager matrix, an ON or OFF polarity of the events (indicating either the increase or decrease of illumination) and a timestamp indicating when the event was received and processed by the interface. Performances of event cameras are characterized by low latencies of single pixels (translating to high effective frame-rate) and a great dynamic range. While high frame rates are certainly achievable using conventional imager technology, they require a powerful backend to even collect the data, never mind being able to process it in real-time. This issue is further exacerbated with dynamically changing scenes: a frame-based camera continues to generate overwhelmingly useless data in a static scene while waiting for some rapid event, while an event-based sensor will only report the essential information from selected pixels when something happens. Finally, event-based computation has demonstrated to be suitable for applications on the edge having real-time constraints because of its low-power properties [12]. This represents an interesting feature for spacecraft, given their strict power and energy constraints [11, 26].

This work proposes the use of event-based sensors in autonomous visual-based landing techniques that rely on optic flow measurements such as that of the time to contact (TTC). The final aim is that of achieving a self-contained, low power, lightweight descent system. Measuring TTC alone has been proven sufficient to control the flight of the craft and the properties of this strategy have been described in depth in [17]. Landing using TTC was also performed using conventional cameras and corner detection algorithms for TTC estimation on board micro-air vehicles [16]. We show how those same necessary quantities can be estimated from the output of an event-based camera with sufficient precision to control the spacecraft descent thus paving the

way to a purely neuromorphic landing system [18] [32].

The paper is structured as follows: Section §2 presents the dynamic model of the craft and demonstrates the principle of TTC usage for automated landing guidance. Section §3 introduces the model of an event camera used in this study and show event data produced from frames generated by a "regular" camera. Section §4 introduces the algorithm used to estimate TTC from the event stream. Section §6 presents closed-loop simulations of landings over nine different moon-like sites and discusses the results. Finally, Section §7 discusses our work and proposes possible future investigations.

2. TTC based landing

We consider here the same vertical descent dynamics studied in [17], modelling a spacecraft descending vertically in the final phase of a planetary landing. The spacecraft camera and thrust are both assumed to be oriented vertically and thus no modelling of the spacecraft attitude is necessary. The exact set of equations thus are:

$$\begin{aligned} \dot{v}_z &= \tfrac{u_{th}}{m} - g \\ \dot{z} &= v_z \\ \dot{m} &= -\tfrac{u_{th}}{I_{sp} g_0} \end{aligned} \tag{1}$$

where I_{sp} is the specific impulse of the spacecraft propulsion system, $g_0 = 9.81$ [m/s^2] the value of the Earth acceleration at sea level and g denotes the gravitational acceleration, which assumed constant during the modelled descent phase. The spacecraft upward thrust is indicated with u_{th}. All quantities are in a frame of reference attached to the planet with $z = 0$ considered to be at the surface. Following [17] we introduce the time to contact (TTC) as the quantity:

$$\tau = -\frac{z}{v_z}$$

the minus sign accounts for the fact that under our assumptions, for a descending craft $v_z < 0$ and $z > 0$. This definition ensures the TTC is a positive quantity representing the time left before reaching a zero altitude when assuming a uniform motion from z with velocity v_z. We will study constantly decreasing TTC landing where, ideally, the controller is set to track a target τ^* which obeys a simple relation:

$$\dot{\tau}^* = -c^2$$

Where c is a free parameter used to control the landing duration. Previous studies [21] [17] have used the value of 0.5 for c^2 and the same will be done throughout this paper. A simple proportional controller with gain K_p is applied to regulate the spacecraft thrust as follows:

$$u = K_p(\tau^* - \tau)$$

Note that during the entire landing sequence engine thrust needs to also counter the gravitational pull as well as tracking τ^*. Therefore the final feedback law used for u_{th} is:

$$u_{th} = u + mg = K_p(\tau^* - \tau) + mg \tag{2}$$

The success of this control feedback is guaranteed whenever perfect sensing is assumed, but in a real application it requires a precise enough estimation of the parameter τ along the landing sequence. If successful, the resulting landing scheme can safely drive the spacecraft to land without the need for any further exteroceptive sensors.

3. Event-based camera model

3.1. Dynamic pixel

The sensor simulated in this study was modelled after the Dynamic Vision Sensor (DVS) [22], the earliest practical prototype of an event camera. Other more sophisticated sensors have been developed since [5, 27] showing an increased interest in the technology. The original DVS we here use has a 128x128 pixels matrix and outputs only events, as opposed to some camera designs that enable a hybrid combination of frame&events output data [27]. A more recent version of the sensor allows for more pixels, but are not studied here. The basic building block of the event camera is the "dynamic pixel" [10], an analogue electronic circuit around a photodiode that tracks the change in illumination and decides when to signal an event. Arranging many such pixels in a grid and connecting them with an interface handling the events constitutes the imaging matrix of the sensor. The circuitry of a single-pixel can be split into 3 stages: the photoreceptor circuit, the differencing circuit and the thresholding circuit. The photoreceptor circuit consists of a photodiode generating photocurrent from incident light and a transimpedance loop that converts it to a voltage through a logarithmic law:

$$V_p = \frac{kT_p}{q} ln(I/I_0) + const$$

where k is the Boltzmann constant and T_p is the temperature. The differencing circuit is sensitive to temporal contrast (TCON) changes, which it integrates to track the deviation of V_p from the Reset Level:

$$TCON = \frac{d(ln(I(t)))}{dt}$$

$$\Delta V_{diff} = -\alpha \int_t^{t+\Delta t} TCON(t')dt'$$

Where α is a proportionality constant determined from physical parameters. Finally, the thresholding circuit consists of the ON and OFF comparators that switch state when ΔV_{diff} deviates far enough from Reset Level. A switching

of either of the comparators triggers a series of events that start with the pixel indicating a new event to the AER interface and end with the differencing circuit resetting and "zeroing" the voltage corresponding to the pixel illumination at that time. The principle of operation and a detailed analysis of the circuits can be found in [10, 22].

3.2. Parametrizing the camera

The above process can be abstracted with 2 main parameters: pixel latency T and contrast threshold θ. Pixel latency is the shortest time that needs to pass between 2 consecutive events that a single pixel can report. If the scene changes too rapidly then some information will inevitably be lost. The contrast threshold is the fraction by which pixel illumination has to change in order to trigger an event. It is a dimensionless quantity that regulates how sensitive the camera is to changes.

For the DVS the value of T is $15\mu s$, though it might increase in low-light conditions depending on the hardware interfaces and sensors current biases from hundreds of microseconds to few milliseconds [22]. In any case its a time scale by far shorter than anything that would be expected in a planetary landing context, hence it is not a limiting factor and has little impact. The value of θ is adjustable in hardware and can be set between 0.1 and 0.6 [22] (typically, a lower θ leads to a very high number of events and higher θ suffers from a non-uniform response from pixels across the matrix). In this study, θ was set to be 0.1 for all experiments reported.

3.3. Frames-to-event conversion

Emulating the behaviour of a dynamic pixel and generating event data is done by rendering a sequence of frames in third-party software and comparing intensities of corresponding pixels between them. To this aim, a specialised software called PANGU (Planet and Asteroid Natural Scene Simulation Utility) [2] was used. PANGU offers a set of tools allowing users to generate models of extraterrestrial objects or surfaces and render images of them under a variety of conditions as they would be captured by imaging instruments on-board probes. It allows the user to adjust a range of physical parameters of the imager and request output in a "raw", high-precision format that, essentially, corresponds to the photocurrent induced by incident light, including a modelled noise. For that reason, it is very well suitable for the purpose of our simulations.

Each frame, a 128x128 array of pixel photocurrents, then has an element-by-element logarithm operation applied to it. This causes the value in the newly formed array to essentially correspond to V_p, the voltage exiting the photoreceptor stage. It is important to note that the values obtained this way, will represent a pseudo-voltage $\tilde{V}_p = c_0 + c_1 V_p$ where the offset c_0 and the factor c_1 will be constant between frames. Since the next stage looks only at proportional differences we may use $\tilde{V}_p$ directly. Such is the simplicity of abstracting the operation of the pixel using the contrast threshold θ.

At this point having a frame F at time t and a subsequent frame at $t + \Delta t$ we determine which pixels reported any events in between. The condition for spiking and the corresponding event polarity p is:

$$|F_{t+\Delta t}(x, y) - F_t(x, y)| > (1 + \theta) F_t(x, y) \tag{3}$$

$$p(x, y) = sign(F_{t+\Delta t}(x, y) - F_t(x, y))$$

The problem with implementing directly the approach above is, especially in our case where the overall dynamics happens in rather large timescales, that it requires to generate many samples of the frames as to mimic the asynchronous nature of an even camera (instead of producing planes of events every Δt) and to avoid missing events if the illumination change corresponds to a contrast change of 2θ or more. To overcome this issue we interpolate between frames, assuming that the illumination of a pixel varies linearly between them and thus recursively seeking crossings with thresholds until a threshold is found that does not satisfy the condition from (3). This way it is possible to generate all events, with polarity and time of occurrence information, while only sampling at a much lower rate and thus with a much lower computational cost.

The frames-to-event conversion algorithm we developed works exactly as explained above: after being initialised with the first frame it can be updated with a new frame and return all the events that would have been generated up to this point (assuming a linear interpolation between pixel values). Interestingly this method allows the frames themselves to be captured in intervals having a varying distance, though all simulations described here use a fixed Δt. Finally, the algorithm needs to be mindful not to violate the minimum latency of the pixel, however, as mentioned earlier, landing scenes evolve too slowly for that to be a real constraint. Nevertheless, this feature was included in the codebase developed.

The code is available as an open-source package and can be found at [1]. Some visualisation of the synthetic event data is presented in Figure 1 and Figure 3. On the former, it can be seen what elements within the observed scene trigger pixel response in the event camera, particularly how edges of darker regions result in trails of pixels reporting events of the same polarity (red and blue lines). While understandable for a human this form of presenting event data is misleading as it ignores the enhanced temporal resolution of event data. It is more realistic to view a 3D scatter of the registered data points as shown in Figure 3. Notice how

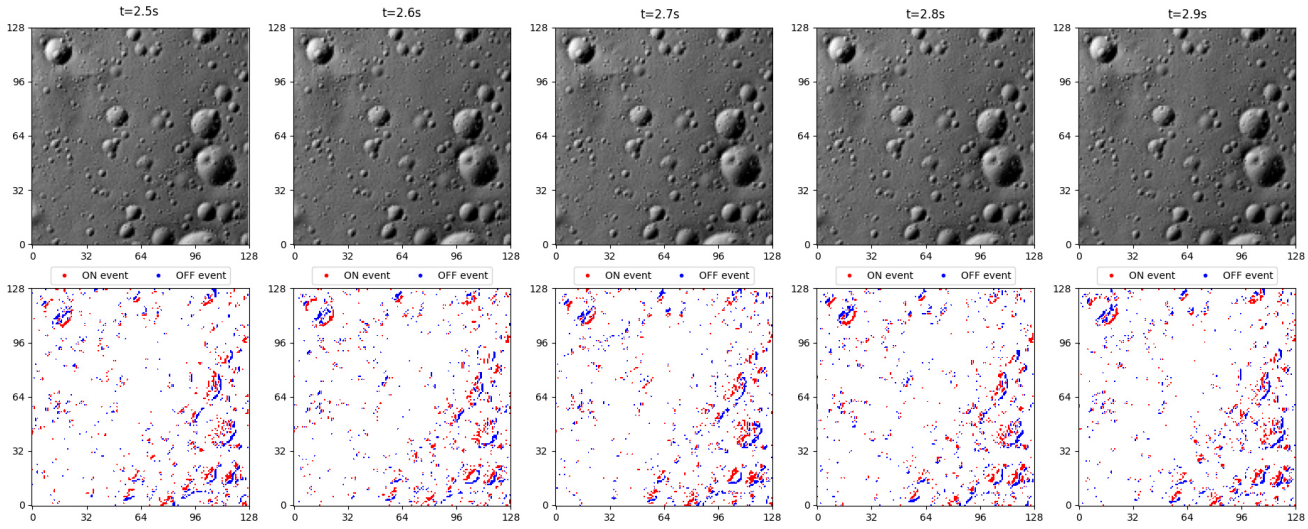

Figure 1. Comparison of conventional and event camera scene registrations over $0.5s$. Images rendered in PANGU and converted using [1] *Top:* lunar-like scene as would be captured by a camera mounted on a probe during lunar landing. *Bottom:* same scene registered by an event camera. colouring indicates which pixels fired at least a single positive or negative event since previous frame.

the most distinct features produce trails of events over time. Light-blue indicates the time period and data corresponding to Figure 1 and Figure 4. A rather complete, albeit different, camera to event conversion software called *v2e* [9] was also made recently openly available by the Sensors Group, at the Institute of Neuroinformatics of the University of Zurich and ETH. Compared to *v2e*, our codebase must be considered as an idealized model of an event-based sensor as it implements less of the specific mechanics of the dynamic pixel while offering a simplified, straightforward, low-level API. The approach taken is, in this sense, closer in philosophy to previous work such as *rpg_vid2e* [14]. Image interpolation approaches such as *slomo* [19] are not integrated and left to the user together with any other possible manipulations of the input, including luma conversion and sensor noise modelling which, in our case, is taken care of by PANGU and is thus to be considered as an idealized abstraction for an event camera.

4. Divergence estimation

Recent works [13, 25] proposed frameworks able to estimate optic flow quantities, including divergence, from event-based sensors and show its use during flying drones landings. The type of events streams one can expect from the applications considered in this recent literature are, though, quite different from the ones produced during a spacecraft landing scenario, which suggests that the approach to optic flow estimation has to be tailored to this particular application. In this section we present an algorithm to estimate the divergence, and thus the TTC, from event data generated during simulated landings over solar system bodies and in particular developed using moon surface surrogates.

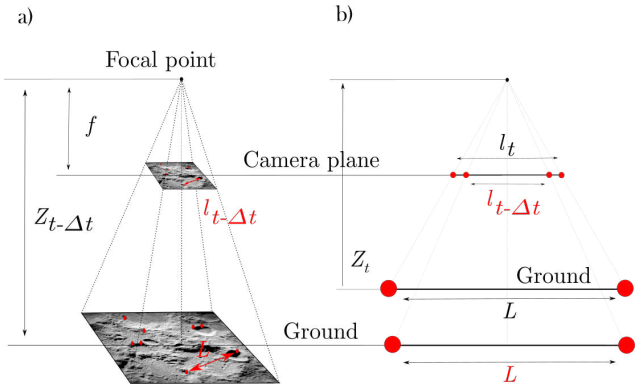

Figure 2. a) Abstraction of the pinhole camera and b) projection of lengths and distances between feature during descent.

4.1. Divergence estimation from sparse tracked features

As mentioned earlier, TTC is related to the rate of perceived expansion of the observed scene as the observer descends: the optic flow divergence. The relation between TTC and the optic flow divergence D is rather straight forward:

$$D = -\frac{1}{\tau} = \frac{v_z}{z} \tag{4}$$

Consider a simple model of a pinhole camera (see Figure 2) with focal length f that observes a scene with identified features (marked by red dots in the figure). The real dis-

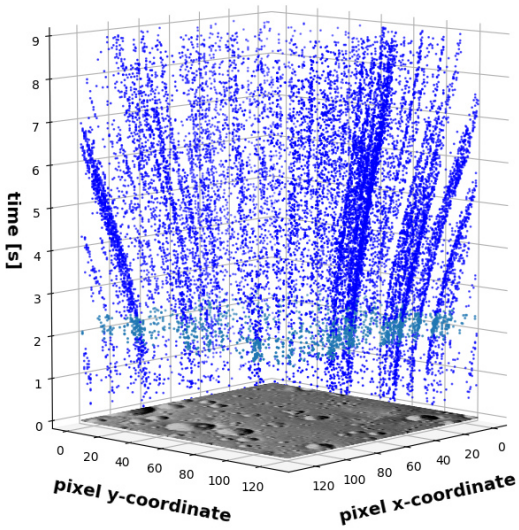

Figure 3. 3D scatter plot of "OFF" events registered by an event camera observing the surface during a simulated descent. The time interval between $t = 2.5s$ and $t = 3.0s$ (coloured in light-blue) was used for producing Figures 1 and 4

tance between a pair of features is denoted by L and the distance on the image plane is denoted by l. Assume that over a time Δt the camera moves from an altitude $Z_{t-\Delta t}$ to an altitude Z_t. In the Figure, the panel b) shows the geometric relationship between the lengths and distances between two randomly selected features. Using properties of similar triangles we may write:

$$\frac{L}{Z_{t-\Delta t}} = \frac{l_{t-\Delta t}}{f}, \frac{L}{Z_t} = \frac{l_t}{f}$$

Eliminating L from these relationships yields:

$$\frac{Z_t}{Z_{t-\Delta t}} = \frac{l_{t-\Delta t}}{l_t} \tag{5}$$

The divergence D can be approximated as:

$$D = \frac{V_t}{Z_t} = \frac{Z_t - Z_{t-\Delta t}}{\Delta t Z_t} = \frac{1}{\Delta t}\left[1 - \frac{Z_{t-\Delta t}}{Z_t}\right] \tag{6}$$

substituting Equation 5 into Equation 6 we may then derive the following relation:

$$D = \frac{1}{\Delta t}\left[1 - \frac{l_t}{l_{t-\Delta t}}\right] \tag{7}$$

which can be used to estimate the divergence of the optic flow and hence the TTC once features can be tracked across time. In practice, using only two features for estimation makes this method extremely prone to noise, so the divergence flow is, instead, estimated for all P pairs ij of tracked

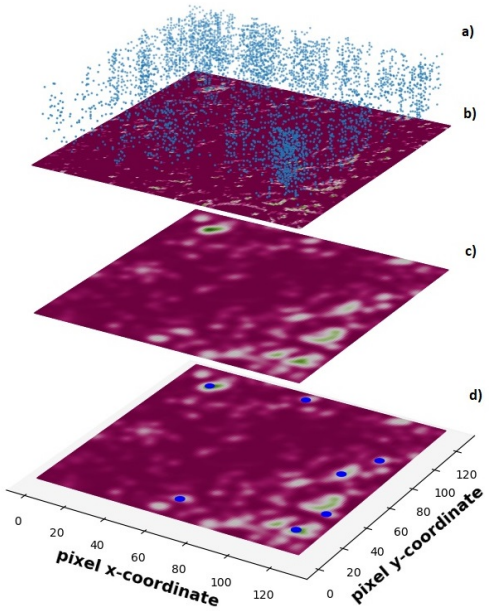

Figure 4. Sequence of transformations in the centroid initialising stage. a) Cloud of stored events. b) Projection of events on a 2D plane. c) Projection image after blurring d) Same image with the located peaks marked in blue.

and identified features and the final value is assumed to be the mean of those separate estimations.

$$D = \frac{1}{P}\sum_{ij=1}^{P}\frac{1}{\Delta t}\left[1 - \frac{l_t^{ij}}{l_{t-\Delta t}^{ij}}\right] \tag{8}$$

4.2. Feature extraction

In order to use Equation 8 with event-based data, we need to identify features in the event stream that correspond to static objects on the target planet surface. Some objects are particularly suitable for that, such as craters and tall rocks, because they cast dark shadows that contrast sharply with the light grey regolith of the moon. As their representations move in the image plane they leave a dense stream of events that produce a trail pattern such as that shown in Figure 3. As visible in the Figure, after adjusting the contrast threshold θ these trails are clearly visible with little accompanying noise.

At a given epoch t all the events within a time interval are considered for processing. An event is defined by its timestamp t_j and the coordinate in the camera plane where it happens. Given the nature of event data, the same coordinate (x_j, y_j) appears frequently in the batch at different timestamps t_j. The events are stored in a first-in-first-out (FIFO) queue according to their timestamps. The FIFO queue only

contains events that are recent enough:

$$t_{event} \geq t_{global} - \mathcal{T} \tag{9}$$

where $\mathcal{T}$ is a parameter called "temporal memory". It defines how much an event can be in the past to still be remembered and thus considered as relevant for the divergence estimation. A clustering algorithm is then tasked to identify, in the camera plane, the (x, y) locations of the centroids of the event batch. It is these centroids that constitute our tracked features for the application of Equation 8. The algorithm performs two separated core functions: it identifies new candidate centroids for tracking (if, at any time, it has too few to be able to estimate divergence) and it updates the locations of the existing centroids. The first function is called only at the beginning, upon initialization of the landing sequence, and whenever too many tracked objects drift outside the camera plane, reducing the number of tracked clusters and thus requiring to find more.

4.2.1 Identifying centroids

Figure 4 illustrates the process of initialising centroids, via our proposed approach called Density Peaks Ranking (DPR). First, the batch of stored events is projected on a 2D plane along the time axis. Each event is given a weight of 1 and if multiple events occupy the same x,y coordinate they are summed, hence producing a 2D array of natural numbers. Next, a Gaussian blurring mask is convolved with the array to smoothen it into an image of event density. Finally, all the existing density maxima are first detected using morphological methods [30] and then sorted by density value. Centroid candidates are thus selected from the sorted list up to a maximum number N_{max}. Note that modern versions of the mean-shift algorithm [7] can also find centroids with no priors and requiring a similar computational cost, but in the event data considered, its performances were found to be unsatisfactory as well as the optimal parameters choice problematic, leading to deteriorated performances.

4.2.2 Updating centroids

The task of updating the positions of the centroids as the event batch changes is carried out by seeding a mean-shift algorithm with the previously found centroids. Between iterations the algorithm checks for two conditions that define a reliable centroid: the existence of a minimum number of events n_{min} within the radius of interest R and of a minimal distance r_{min} from the edge of the image. Failing to meet one of these causes the centroid to be discarded and as the scene changes this will lead to the total number of centroids to fall below the minimal allowed number C_{min} and trigger a re-initialisation. C_{min} is chosen so that divergence can still be estimated one last time before re-initialisation to prevent discontinuities. Unfortunately, this cannot be guaranteed and may happen for example when all centroids travel outside the image plane simultaneously.

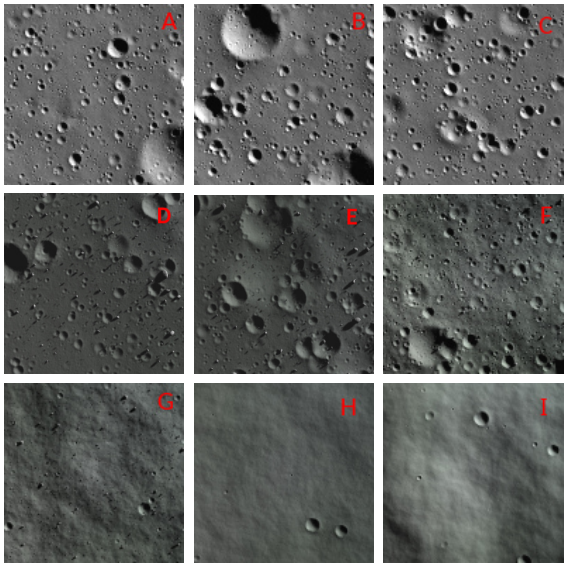

Figure 5. The nine virtual landing sites used. Views captured from $5000m$ above ground.

5. Experiments

We simulate Eq.(1) during a descent on nine different moon surface surrogates created using the PANGU software [2]. The spacecraft thrust is controlled in a closed loop via Eq.(2) thus targeting a constantly decreasing TTC descent. The propulsion system specific impulse is set to $I_{sp} = 600$s. We experiment with two values of the controller gain $K_p = 5 \times 10^6$ kg/m/s^3 and $K_p = 4 \times 10^6$ kg/m/s^3.

During the simulation, a new camera frame is generated from PANGU at a frequency of 10Hz and converted to events using the algorithm outlined in §3. This value is adequate in terms of the number of events detected between two consecutive frames, given the slow dynamics of landing. The events within the temporal memory are then used to estimate the time to contact used in Eq.(2) to generate the thrust request. The camera contrast threshold is kept constant during all descents, regardless of the spacecraft velocity or the illumination conditions, at a value $\theta = 0.1$. For the divergence estimation algorithm the parameters were set to: $N_{max} = 15$, $n_{min} = 15$, $R = 4px$, $C_{min} = 3$, $r_{min} = 3px$ and the temporal memory $\mathcal{T} = 500ms$. Divergence was estimated at each iteration of the simulation, hence in Eq. 8 $\Delta t = 0.1s$. The divergence signal is filtered using a moving average filter. We experimented with two different values for the number of bins used in the filter: $n_{bin} = 5$ and $n_{bin} = 10$. The values were tailored experimentally to the two values chosen for the controller

gain K_p. Adding more aggressive filtering prevents a single outlier estimate from triggering a spike that would slow the craft and disrupt the event stream. However, filtering adversely impacts the stability of the control system, hence more bins required decreased gain. We elaborate further on the stability issues in the results section.

Figure 5 shows the nine landing sites generated. The sites vary in the placement of features on the surface as well as in their quantity and type distribution, hence creating different textures and densities and event stream properties. A, B, C correspond to a terrain where only craters are present. In D and E we add sparse rocks and boulders to the craters. F and G represent rocky terrains with only a few craters. H is our most challenging site with mostly flat, dusty terrain and extremely scarce features. Finally, we generate I similar to the case H but with less extreme parameters letting somehow more features to enter the field of view. For the terrains A to C we use as initial conditions: $z_0 = 10km$, $v_0 = -200ms^{-1}$. Since the early phase of all descents revealed to be the least challenging, for all remaining terrains we focused on the last part and used, as initial conditions $z_0 = 4km$, $v_0 = -100ms^{-1}$. All simulations were terminated whenever one of the following conditions were met during the simulation:

- The altitude reached $z = 50m$: When this condition is met the landing is called a success since the remaining conditions have not triggered and thus the vertical velocity allows for a soft landing. We choose a non zero final altitude as in the very last meters of a descent, an additional landing control system is likely to be used as camera sensors are affected by the plumes generated by the spacecraft thrusters.
- The spacecraft velocity assumed negative values ($v < 0$) indicating an ascending path. Oscillations are a common problem in the controller and any attempt of the craft to ascend is considered evidence of the controller becoming unstable. When this condition is met the landing has to be considered as not fully successful, also depending on the value of the altitude for which the condition is triggered.
- The divergence exceeded a safe value: $D = -\frac{v}{z} \geq 0.5$). If the divergence exceeds this threshold, the descend velocity is considered to be too high for the altitude the spacecraft finds itself and the thrusters will not be able to slow down into a gentle soft landing. When this condition is met the landing has to be considered unsuccessful and the descent is tagged as a "crash".

6. Results

The results of a single example of closed-loop simulation are shown in figure 6, which demonstrates in detail how the

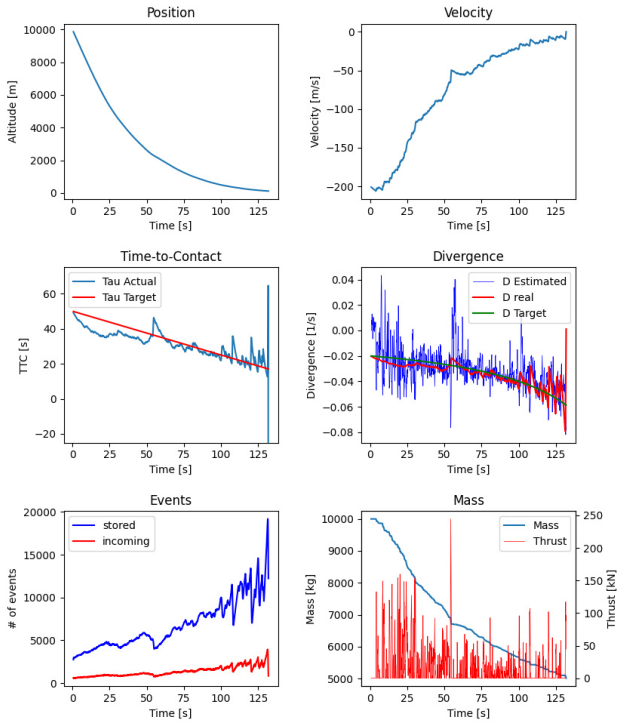

Figure 6. Example of a descent over the surface A.

controller handled the descent on model A from $10km$. As expected, both altitude and velocity decrease exponentially to converge at zero. The estimated divergence signal (in blue) is rather noisy but clearly follows the true divergence value. In the final seconds of the descent $\approx$ 20s the "actual" divergence oscillates rapidly, leading to a spike in the very end that triggers a thrust pushing v above zero and thus terminating the simulation. This phenomenon is observed in previous literature on divergence based drone landing and is a result of the estimation of divergence becoming increasingly sensitive to changes in velocity as the craft approaches the surface [8]. While the problem can be mitigated using advanced control techniques, we have not investigated possible solutions in this work. Note also that the resulting thrust requests, shown in Figure 6, do not represent a realistic scenario, adding to the issues on the control aspects of the problem that will be addressed in the future.

Nevertheless, it is clear from our simulations that the necessary information to guide a planetary landing can be inferred from the registered event data. Our simulations show that the divergence estimation algorithm proposed in this paper already manages to guide the spacecraft calling for further research to extract realistic and superior performances in terms of control profile, mass usage and other quantities of importance to actually engineer and use such a system reliably.

The guidance was most successful over models "D" to

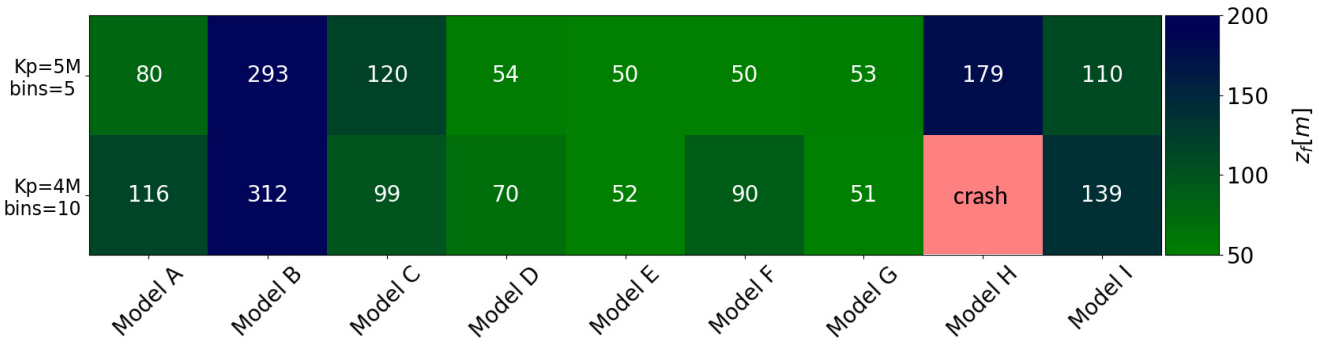

Figure 7. Lowest altitude reached by craft during landing without violating simulation constraints (any landing reaching around 50m is to be considered as successful). Simulations where run for two variations of controller gain and filter bin number for all nine models presented in Figure 5.

"G" for which in two cases the target altitude of $50m$ was reached. This is possible because the large number of boulders, when viewed from lower altitudes, had contrasting shadows which created many events for the divergence estimator to lock onto. The outcome was also moderately successful for models "A" and "C" where guidance became unstable only at the very end and likely for issues related to the simplicity of the proportional controller used as discussed.

It is of interest to compare landing results from the two feature scarce models "H" and "I". It would appear that landings on "H" were far less successful, with a crash in one case, but in truth, both experienced extremely jittery control as the estimator struggled to produce any good estimates for the divergence. The extra unevenness of the ground in model "I" created shadows that occasionally allowed proper functioning of the algorithm and hence a few thrusts that slowed the craft. The poor performance on those models is not surprising as the lack of distinct features broke the assumptions behind the estimator algorithm and are deemed to be extreme cases of scarce importance in a realistic scenario where added texture from the surface would be provided also by smaller scale features not rendered by PANGU. Finally, landings on model "B" ended much earlier than on other comparable models. This was the effect of a brief period of very large error in the estimation. Because it happened fairly low, it triggered large oscillations in the controller when it was already close to instability leading to earlier termination. This is as much a shortcoming of the divergence estimation algorithm as of the whole control system.

7. Discussion and Future investigations

This work aims to pave the way for engineering an onboard, neuromorphic based spacecraft landing system. To this aim, we developed a simulator for event-based sensors and a proof-of-concept algorithm to control the spacecraft landing trajectories. Future work is planned to decrease the probability of a failure and increase the overall simulation fidelity, with a particular focus on the pipeline to extract the divergence D and the control loop.

Recent studies have proposed a number of methods to process event-based data – for example in the context of autonomous driving – which might represent suitable approaches to be pursued also for space-based imagery [6,12]. Among the different solutions, Spiking Neural Networks (SNNs) have been attracting the interest of researchers for their capability in the processing of event-based data [4, 12, 20, 23, 34], showing energy-efficient, low-power and low-latency properties [4, 15, 23, 29]. These features pushed researchers to investigate their applicability for different computer-vision at the edge, including feature extraction [12], optical flow processing [15, 24], and feature tracking [29]. In our case, an end-to-end SNNs-based feature-extraction pipeline can be considered to improve the divergence D estimation and make the overall system truly neuromorphic. Alternatively, SNNs could substitute and empower a portion of the feature-extraction pipelines, such as the one devoted to extract and track centroids.

The use of more complete DVS frameworks, such as *v2e* [9], is also interesting and a comparison to to our current video to event conversion solution is necessary. This would also reveal to what extent noise, and the limited sensor bandwidth of DVS sensors might affect the error in landing. In this paper we focused on a vertical landing with a ventral camera, but more complex landing profiles and camera angles may result in new challenges. The additional events due to the tilts and rotations of the camera might make the reconstruction of the optic-flow harder and require more sophistication in the algorithm to extract the optic flow. Finally, effects of different illumination conditions on the landing accuracy will be considered. In that respect, the higher dynamic range of DVS sensors could make this technology promising for scenarios in which both illuminated and dark areas are included in the camera frame [9, 12].

References

[1] Event-based-vision: software repository under the European Space Agency. https://gitlab.com/EuropeanSpaceAgency/event-based-vision.

[2] Planet and Asteroid Natural Scene Generation Utility product website. https://pangu.software/. Accessed: 2020-09-20.

[3] Friedrich G. Barth, Joseph A.C. Humphrey, and Mandyam V. Srinivasan. *Frontiers in Sensing*. Springer, 2012.

[4] Zhenshan Bing, Claus Meschede, Guang Chen, Alois Knoll, and Kai Huang. Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle. *Neural Networks*, 121:21–36, 2020.

[5] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240 $\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49:2333–2341, 2014.

[6] Guang Chen, Hu Cao, Jorg Conradt, Huajin Tang, Florian Rohrbein, and Alois Knoll. Event-based neuromorphic vision for autonomous driving: a paradigm shift for bio-inspired visual sensing and perception. *IEEE Signal Processing Magazine*, 37(4):34–49, 2020.

[7] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[8] Guido de Croon. Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy. *Bioinspiration & biomimetics*, 11(1):016004, 2016.

[9] Tobi Delbruck, Yuhuang Hu, and Zhe He. V2E: From video frames to realistic DVS event camera streams. *arXiv preprint arXiv:2006.07722*, 2020.

[10] T. Delbruck and C. A. Mead. Adaptive photoreceptor with wide dynamic range. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, volume 4, pages 339–342 vol.4, May 1994.

[11] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Léonie Buckley, Mihalis Psarakis, Kay-Obbe Voss, et al. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, 35(12):44–56, 2020.

[12] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.

[13] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018.

[14] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.

[15] Germain Haessig, Andrew Cassidy, Rodrigo Alvarez, Ryad Benosman, and Garrick Orchard. Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system. *IEEE transactions on biomedical circuits and systems*, 12(4):860–870, 2018.

[16] Hann Woei Ho, Guido de Croon, and Q. Chu. Distance and velocity estimation using optical flow from a monocular camera. *International Journal of Micro Air Vehicles*, 9:198–208, 03 2017.

[17] Dario Izzo and Guido de Croon. Landing with time-to-contact and ventral optic flow estimates. *Journal of Guidance, Control, and Dynamics*, 35:1362–, 07 2012.

[18] Dario Izzo, Nicolás Weiss, and Tobias Seidl. Constant-optic-flow lunar landing: Optimality and guidance. *Journal of Guidance, Control, and Dynamics*, 34(5):1383–1395, 2011.

[19] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[20] Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In *European Conference on Computer Vision*, pages 366–382. Springer, 2020.

[21] Denis N Lee. The optic flow field: The foundation of vision. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 290(1038):169–179, 1980.

[22] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal con-

trast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, Feb 2008.

[23] Riccardo Massa, Alberto Marchisio, Maurizio Martina, and Muhammad Shafique. An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor. *arXiv preprint arXiv:2006.09985*, 2020.

[24] Federico Paredes-Vallés, Kirk YW Scheper, and Guido CHE de Croon. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE transactions on pattern analysis and machine intelligence*, 42(8):2051–2064, 2019.

[25] Bas J Pijnacker Hordijk, Kirk YW Scheper, and Guido CHE De Croon. Vertical landing for micro air vehicles using event-based optical flow. *Journal of Field Robotics*, 35(1):69–90, 2018.

[26] George Pitsis, Grigorios Tsagkatakis, Christos Kozanitis, Ioannis Kalomoiris, Aggelos Ioannou, Apostolos Dollas, Manolis GH Katevenis, and Panagiotis Tsakalides. Efficient convolutional neural network weight compression for space data classification on multi-fpga platforms. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3917–3921. IEEE, 2019.

[27] C. Posch, D. Matolin, and R. Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2011.

[28] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck. Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014.

[29] Alpha Renner, Matthew Evanusa, and Yulia Sandamirskaya. Event-based attention and tracking on neuromorphic hardware. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1709–1716. IEEE, 2019.

[30] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[31] M Srinivasan, S Zhang, M Lehrer, and T Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199(1):237–244, 1996.

[32] Florent Valette, Franck Ruffier, Stéphane Viollet, and Tobias Seidl. Biomimetic optic flow sensing applied to a lunar landing scenario. In *2010 IEEE International Conference on Robotics and Automation*, pages 2253–2260. IEEE, 2010.

[33] Anup Vanarse, Adam Osseiran, and Alexander Rassau. A review of current neuromorphic approaches for vision, auditory, and olfactory sensors. *Frontiers in neuroscience*, 10:115, 2016.

[34] Yannan Xing, Gaetano Di Caterina, and John Soraghan. A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition. *Frontiers in Neuroscience*, 14, 2020.