NAVIGATION

Dynamic obstacle avoidance for quadrotors with event cameras

Davide Falanga*, Kevin Kleber, Davide Scaramuzza

Today's autonomous drones have reaction times of tens of milliseconds, which is not enough for navigating fast in complex dynamic environments. To safely avoid fast moving objects, drones need low-latency sensors and algorithms. We departed from state-of-the-art approaches by using event cameras, which are bioinspired sensors with reaction times of microseconds. Our approach exploits the temporal information contained in the event stream to distinguish between static and dynamic objects and leverages a fast strategy to generate the motor commands necessary to avoid the approaching obstacles. Standard vision algorithms cannot be applied to event cameras because the output of these sensors is not images but a stream of asynchronous events that encode per-pixel intensity changes. Our resulting algorithm has an overall latency of only 3.5 milliseconds, which is sufficient for reliable detection and avoidance of fast-moving obstacles. We demonstrate the effectiveness of our approach on an autonomous quadrotor using only onboard sensing and computation. Our drone was capable of avoiding multiple obstacles of different sizes and shapes, at relative speeds up to 10 meters/second, both indoors and outdoors.

INTRODUCTION

Micro-aerial vehicles (MAVs) may open up new markets potentially worth several billion dollars. These markets include aerial imaging [forecast value of $4 billion by 2025 (*1*)], last-mile delivery [$90 billion by 2030 (*2*)], and aerial mobility [worth about $8 billion in 2030 (*3*)]. Although the market potential for MAVs is promising, safety remains a key challenge for MAVs. Several drone crashes have been reported in the news, due to either objects tossed at quadrotors during public events (*4*, *5*) or collisions with birds (*6*, *7*). Thus, enabling MAVs to evade fast-moving objects (Fig. 1) is critical for the deployment of safe flying robots on a large scale.

The temporal latency between perception and action plays a key role in obstacle avoidance. The higher the latency, the lower the time the robot has to react and execute an avoidance maneuver (*8*). This is especially critical for MAVs, where a collision can not only damage the environment but also cause severe hardware failure. In addition, micro-quadrotors have reduced payload capabilities, which puts a hard bound on the sensing and computing resources they can carry.

The existing literature on obstacle avoidance for MAVs relies on standard cameras [in a monocular (*9*–*11*) or stereo configuration (*12*–*15*)] or on depth cameras (*16*–*18*). However, these works assume that the obstacles in the environment are either static or quasistatic (i.e., slow relative motion). Similarly, state-of-the-art consumer drones are not currently capable of reliably detecting and avoiding moving obstacles. For example, the Skydio drone, one of the most advanced autonomous drones on the market to date, is not capable of dealing with moving objects (*19*). Therefore, developing effective solutions to avoid dynamic obstacles is a key challenge in robotics research, as well as a highly prized goal by major industry players.

To avoid fast-moving obstacles, we need to perceive them quickly. For this purpose, standard cameras are not good enough because they have an average latency of tens of milliseconds (the exposure time of a standard camera varies between 1 and 100 ms). Therefore, their limitations arise from their physical nature and cannot be solved with sophisticated algorithms. One solution could be to use event cameras; these sensors have reaction times of microseconds. Event cameras (*20*) are bioinspired sensors that work differently from traditional cameras. Instead of capturing images at a fixed rate, an event camera measures per-pixel brightness changes asynchronously. This results in a stream of events at microsecond resolution.

In a recent study on the role of perception latency for high-speed sense and avoidance (*8*), it was shown analytically that, using event cameras, the latency between the time a visual signal is triggered by the sensor and processed to output control commands is remarkably lower than that of standard cameras (microseconds versus tens of microseconds). These results are promising for robotics. However, standard vision algorithms cannot be applied because the output of an event camera is not images but a stream of asynchronous events. Thus, alternative algorithms need to be devised to unlock the full potential of event cameras for the task at hand.

An event camera has smart pixels that trigger information independently of each other: Whenever a pixel detects a change of intensity in the scene (e.g., caused by relative motion), that pixel will trigger an information at the time the intensity change was detected. This information is called an event and encodes the time (at microsecond resolution) at which the event occurred, the pixel location, and the sign of the intensity changes. Let t_{k-1} be the last time when an event fired at a pixel location $\mathbf{x}$, and let $L_{k-1} = L(\mathbf{x}, t_{k-1})$ be the intensity level at such pixel at time t_{k-1}. A new event is fired at the same pixel location at time t_k as soon as the difference between the intensity L_{k-1} and L_k is larger than a user-defined threshold $C > 0$. In other words, an event is fired if $\|L(\mathbf{x}, t_k) - L(\mathbf{x}, t_{k-1})\| > C$ (positive event) or $\|L(\mathbf{x}, t_k) - L(\mathbf{x}, t_{k-1})\| < -C$ (negative event). We refer the reader to (*21*) for further details. To better highlight what happens across the entire sensor, we compare the output of an event camera with the one of a conventional camera in Fig. 2 and in a video (*22*).

Event cameras may be considered to be asynchronous, motion-activated sensors, because they provide measurements only if and where there is relative motion between the camera and the environment. Also, because their latency is on the order of microseconds,

Department of Informatics, University of Zurich, Zurich, Switzerland.
*Corresponding author. Email: falanga@ifi.uzh.ch

Fig. 1. Sequence of an avoidance maneuver.

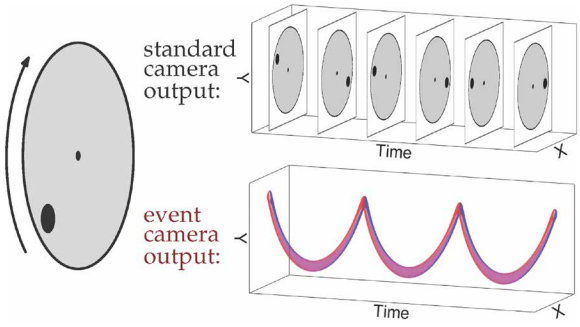

Fig. 2. Comparison of the output of a conventional camera versus an event camera for a rotating disk with a black dot. A conventional camera captures frames at a fixed rate; an event camera only outputs the sign of brightness changes continuously in the form of a spiral of events in space time (red, positive changes; blue, negative changes).

they are a natural choice for detection and avoidance of fast-moving obstacles by flying MAVs. If one removes the events induced by the ego-motion of the vehicle (*23*, *24*), one can directly obtain information about the moving part of the scene. This leads to multiple advantages over standard cameras for detection of dynamic obstacles: (i) The output is sparser and lighter than a frame and therefore cheaper to process; (ii) no segmentation between static and dynamic objects is necessary, because to do so, it is possible to exploit the temporal statistic of each event; (iii) their high temporal resolution (in the order of microseconds) allows low-latency sensing.

In recent years, event cameras have attracted the interest of the robotics community (*21*). Obstacle detection is among the applications with the highest potential, and previous works investigated the use of these sensors to detect collisions (*25*) and track objects (*23*, *26*). However, very few examples of closed-loop control based on event cameras are available in the literature. Among these, the majority of the works focuses on simple, low-dimensional tasks, such as 1–degree of freedom (DoF) heading regulation (*27*, *28*), stereo camera gaze control (*29*, *30*), 2-DoF pole balancing (*31*), 1-DoF robotic goal-keeping (*32*, *33*), or navigating ground robots among static obstacles (*34*–*36*).

We have reported closed-loop control of more complex robotic systems, such as quadrotors, using event cameras recently (*37*–*39*). In (*37*), we proposed an event-based visual-inertial odometry (VIO) algorithm for state estimation and closed-loop trajectory tracking of a quadrotor. Instead, (*38*) and (*39*) are the most related to this paper. In (*38*), we analyzed the feasibility of detecting spherical objects thrown at a stationary event camera on small embedded processors for quadrotors. In (*39*), we showed preliminary results in using shallow neural networks for segmenting moving objects from event streams and demonstrated an application to quadrotor obstacle avoidance. However, the resulting sensing latency was 60 ms rather than the 3.5 ms of this paper, thus strongly limiting the maximum relative speed at which moving objects could be evaded. In addition, differently from this paper, there, we did not consider the relative distance and velocity to compute the avoidance commands. This work implements and demonstrates low-latency (3.5 ms) dynamic obstacle dodging on an autonomous quadrotor with relative speeds up to 10 m/s.

Our moving-obstacle detection algorithm works by collecting events during a short-time sliding window and compensating for the motion of the robot within such a time window. Figure 3 shows the effects of the ego-motion compensation: On the left side, the three-dimensional (3D) volume of the events accumulated during an arbitrary time window of 10 ms, and on the right side, the same events, after ego-motion compensation, back-projected on the image plane. We analyzed the temporal statistics of the motion-compensated events to remove those generated by the static part of the environment.

Broadly speaking, our algorithm is based on the intuition that the static part of the scene fires events uniformly across the entire time window, and after the ego-motion compensation, the pixels belonging to static objects show a uniform distribution of time stamps; conversely, dynamic objects generate ego-motion–compensated events that are accumulated around specific sections of the time window and can, therefore, be distinguished. Our technique to detect moving obstacles using event cameras is based on the method proposed in (*23*), where the authors used an optimization-based ego-motion compensation scheme. We modified that method to only rely on an inertial measurement unit (IMU) for the ego-motion compensation, without using any optimization scheme, which makes the algorithm sufficiently fast to run in real time on a small onboard computer.

An analysis of the impact of neglecting the linear component of the robot's ego-motion and an intuitive explanation of how and why our algorithm works are provided in the Supplementary Materials. A detailed explanation of ego-motion compensation of the events—which allowed us to obtain a so-called event frame, containing only events coming from moving objects, at a very high rate—is explained in Materials and Methods. We leveraged a fast clustering algorithm to tell apart different objects in the event frame and used a Kalman filter to obtain information about their velocity. Figure 4 provides a visual explanation of the steps involved in our algorithm and described in more detail in Materials and Methods.

The position and velocity of each obstacle relative to the camera were then fed to a fast avoidance algorithm designed to leverage the low sensing latency. To do so, we used a reactive avoidance scheme

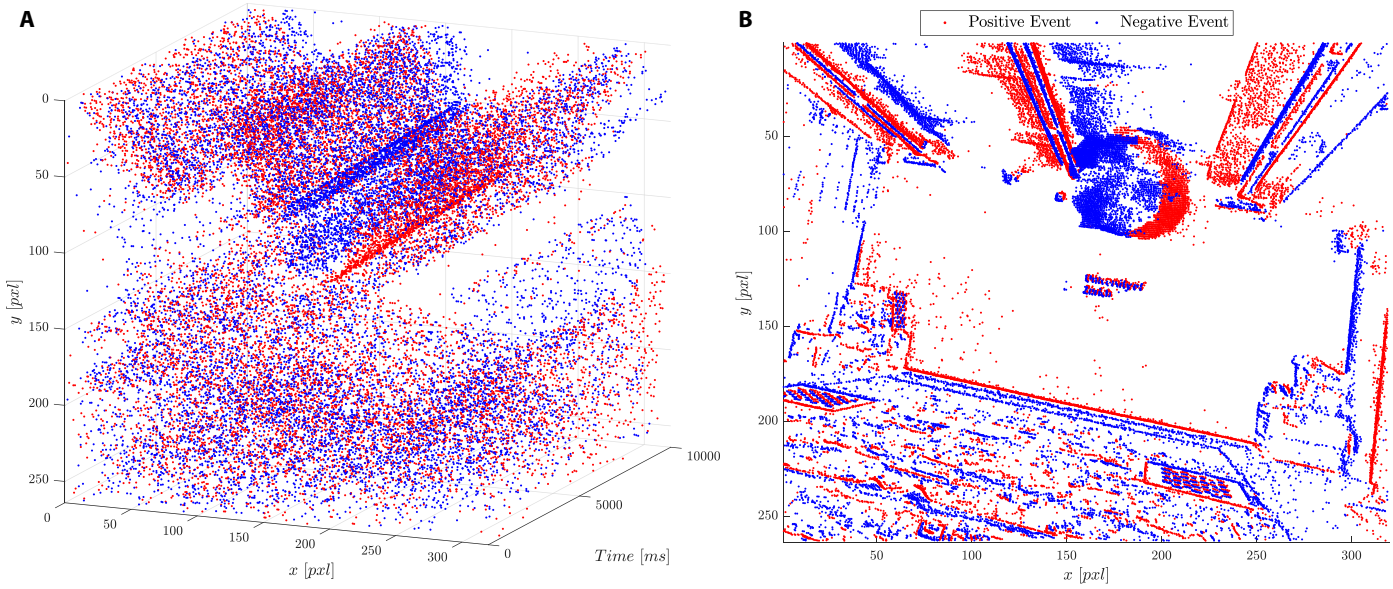

Fig. 3. Effects of the ego-motion compensation. Our algorithm collects all the events that fired during the last 10 ms, here represented in the 3D volume (left side), and used the IMU to compensate for the motion of the camera. The ego-motion compensated events are therefore projected into a common image frame (right side) where each pixel (pxl) contains potentially multiple events. By analyzing the temporal statistics of all the events projected into each pixel, our approach is able to distinguish between pixels belonging to the static part of the scene and to moving objects.

based on the use of artificial potential fields (*40*) relying on fast geometric primitives to represent the obstacles, which renders it computationally inexpensive. We proposed a formulation of the repulsive field, which better suits the task of avoiding fast-moving obstacles by taking into account the need for a prompt reaction of the robot when an obstacle is detected. Compared with previous approaches, our formulation of the repulsive potential increased remarkably faster as the distance between the robot and the obstacle decreased to render the avoidance maneuver more reactive and agile. In addition, we considered both the magnitude and the direction of the obstacle's velocity to decide in which direction to evade and introduced a decay factor in the magnitude of the potential to take into account that the obstacles we consider are dynamic; i.e., they do not occupy the same position in time.

Our approach prioritizes computation speed over accuracy; therefore, we trade off detection accuracy for latency. Nevertheless, we show that our algorithm only takes, on average, 3.5 ms (from the moment it receives the events to process to when it sends the first command to avoid the detected obstacles) to detect moving obstacles with a position error usually in the order of a few tens of centimeters. Trading detection accuracy for latency is not only a necessity for robotic platforms, but it has been frequently observed also among animals (*41*) for the execution of several tasks involving visual sensing.

We demonstrated the effectiveness of our approach in real experiments with a quadrotor platform. We validated our system with both a monocular setup (for obstacles of known size) and a stereo setup (for obstacles of unknown size), both indoors and outdoors. The entire avoidance framework is capable of running in real time on a small single-board computer onboard the vehicle, together with the entire software stack necessary to let the robot fly (i.e., state estimation, high-level control, and communication with the flight controller). Experimental results show that our framework allowed the quadrotor to avoid obstacles moving toward it at relative speeds up to 10 m/s from a distance of around 3 m. The integration of known techniques for event-based obstacle detection (*23*) and avoidance (*40*), adapted to make it possible for the entire framework to run in real time on a small computer as well as to deal with fast-moving obstacles, represents the main contribution of this paper.

RESULTS

Accuracy and success rate

We collected a dataset of more than 250 throws, obtaining around 1200 detections, and compared the output of our event-based detector with ground truth data from a motion capture system. For each detection, we computed the norm of the position error, and in Table 1, we summarize the results.

We grouped together measurements falling within bins of 0.5 m, with the first one starting from a distance of 0.2 m along the camera's optical axis, because the algorithm did not successfully detect the obstacles at closer ranges. In the case of the monocular detector, it was necessary to discard some of the data we collected due to the fact that, at short distances, the obstacle was often only partially visible, and therefore, our monocular algorithm failed to correctly estimate its distance because it would fit a known size to a partially visible object. This issue became less notable as the distance to the camera increased, and after 1 m, it did not substantially affect the detector's performance. On the other hand, as expected, the stereo configuration was more precise at low ranges: The further the distance between the cameras and the object, the higher the uncertainty in the triangulation and, therefore, the larger the error.

Independently of the configuration, however, the data in Table 1 show that our algorithm, although not tailored toward accuracy but rather optimized for low latency, provides measurements that are sufficiently accurate to allow a quadrotor to perceive its surroundings and detect moving obstacles effectively. The position error is, on

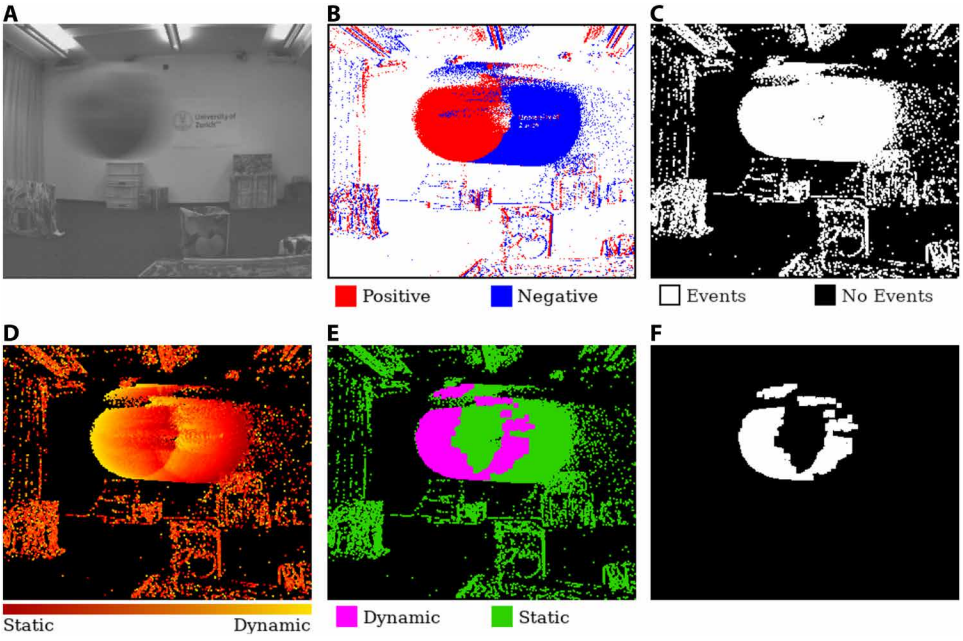

Fig. 4. Stages of the ego-motion compensation algorithm to isolate the events belonging to moving obstacles. (**A**) A frame captured by the Insightness SEES1 camera showing the motion blur due to the relative motion between the sensor and the moving obstacle. (**B**) All the events accumulated in the last window, with red and blue indicating the polarity (positive and negative, respectively). (**C**) The result of the ego-motion compensation, showing in white all the pixels where there has been at least one event in the time window. (**D**) The motion-compensated events, with color code representing the normalized mean time stamp: The events belonging to the dynamic part of the scene are represented in yellow. (**E**) A mean time stamp image after thresholding: Green and purple indicate the static and the moving part of the scene, respectively. (**F**) Events belonging to moving obstacles. This frame is used to segment out the different dynamic objects in the scene.

average, smaller than the vehicle size, whereas the SD is slightly higher but still reasonable up to 1 m. Among the factors that contribute to the error in estimating the obstacles' position, the low resolution of the camera certainly plays a key role.

Another important aspect to consider in our detection algorithm is its success rate. For the entire framework to be effective, not only it has to guarantee low latency, but also it has to guarantee robustness in terms of success rate. To assess the robustness of our algorithm, we performed an evaluation with objects of different sizes and shapes. Such objects were thrown through the field of view of the sensors, and using the intrinsic calibration of the camera and ground truth data from a motion capture system, we obtained information about when they were supposed to be visible. If the object was in the field of view of the cameras, but the system did not report any detection, we considered it as a failure of the detection algorithm. This allowed us to analyze the success rate of our algorithm to detect moving objects, which is reported in Table 2. We used objects of size up to 30 cm, and in the table, we grouped together objects belonging to three different categories: smaller than 10 cm, smaller than 20 cm, and up to 30 cm. For each category, we provided data about the success rate when the objects move at different distances from the camera and grouped the detections according to such a distance. As one can notice, our algorithm provided high success rates in different scenarios, with small and large objects. Given the limited fields of view of the cameras, large objects have a lower detection rate at short distances, mostly because of their different appearance in the two cameras used in the stereo setup, which makes it hard for the algorithm to match the individual detections. Similarly, small objects are harder to detect at large distances because of the limited angular resolution of the sensor used for the experiments. In addition, we did not notice any substantial impact of the objects' speed or incoming angle on the success rate.

Computational cost

By throwing objects within the field of view of the event camera, while simultaneously rotating it and measuring the time necessary to process all the events that fired within the last time window of 10 ms, we could quantify the computational cost of our detection algorithm. Table 3 shows the results of our evaluation, highlighting how each step of the algorithm contributes to the overall computation time. Our evaluation was performed on an NVIDIA Jetson TX2 board, with the algorithm running exclusively on the central processing unit (i.e., the graphics processing unit available on the same board was not used at all). The numbers reported in Table 3 refer to the time required to run the detection algorithm with one camera; however, running multiple instances of the same algorithm for multiple cameras (as for example in the stereo case) did not affect the performance in a substantial way, because the individual parts can be computed in parallel.

The most expensive part of the algorithm is given by the ego-motion compensation, which, on average, requires 1.31 ms (36.80% of the overall time), with an SD of 0.35 ms. The time necessary for this step depends on the number of events that need to be processed, and fig. S4 shows a linear dependence between the two. To understand how many events are typically generated in real-world scenarios during our experiments, we collected some statistics about the number of events that the algorithm needs to process. The collected data indicated that, on average, both indoors and outdoors, the number of events belonging to a time window of 10 ms spanned between 2000 and 6000.

Another step that depends on the relative motion between the camera and the scene is the clustering of the events belonging to the dynamic obstacles. This step is necessary to understand how many objects are in the scene and to associate each event with them. Clustering the events usually required 0.69 ms (19.39% of the overall time), with an SD of 0.20 ms. The actual processing time to cluster the events depends on the number of pixels where events belonging to dynamic obstacles fired, and fig. S5 shows how long our clustering algorithm takes as a function of the number of pixels to process.

Last, thresholding the mean time stamp image and applying some morphological operations to the thresholded image do not depend on the number of events to be processed (as shown by their very low SDs), because the entire picture has to be processed, and they required, on average, 0.98 ms (27.52%) and 0.58 ms (16.29% of the overall time), respectively.

Table 1. Accuracy of our event-based algorithm to detect moving obstacles. We analyzed both the monocular and the stereo setups and compared the detections with ground truth data provided by a motion capture system. For each configuration, we report (expressed in meters) the mean, the median, the SD, and the maximum absolute deviation (M.A.D.) of the norm of the position error for different ranges of distances.

	Monocular				Stereo			
Distance (m)	**Mean**	**Median**	**SD**	**M.A.D.**	**Mean**	**Median**	**SD**	**M.A.D.**
0.2–0.5	0.08	0.05	0.18	0.09	0.07	0.05	0.07	0.06
0.5–1.0	0.10	0.05	0.22	0.10	0.10	0.05	0.18	0.10
1.0–1.5	0.10	0.05	0.20	0.10	0.13	0.07	0.21	0.12

Table 2. Success rate of the event-based detector. Each column reports the success rate for objects moving at a certain distance range from the camera. Each row shows the success rate of detecting objects smaller than a certain size. The results are obtained on a dataset comprising 100 throws of objects belonging to each size.

	Distance (m)		
Size	**≤0.5**	**≤1**	**≤1.5**
≤0.1 m	92%	90%	88%
≤0.2 m	87%	92%	97%
≤0.3 m	81%	88%	93%

Table 3. The mean, μ, and SD, σ, of the computation time of the obstacle detection algorithm.

Step	**μ (ms)**	**σ (ms)**	**Percentage (%)**
Ego-motion compensation	1.31	0.35	36.80
Mean time stamp threshold	0.98	0.05	27.52
Morphological operations	0.58	0.04	16.29
Clustering	0.69	0.20	19.39
Total	3.56	0.45	100

It is important to notice that in our evaluation of the algorithm's computational time, we neglected the estimation of the 3D position of the obstacle. This step requires very simple calculations, which are independent on the number of events generated and, on average, require times in the order of few microseconds. Therefore, their impact on the overall computational time is negligible.

Different types of obstacles

The main reason for us to adopt a stereo configuration for our framework is the necessity to be able to detect moving obstacles independently on their shape and size correctly. This is not possible, using a single camera, as long as the size of the obstacle is not known in advance. Figure S5 shows that the algorithm we propose in this paper to detect moving obstacles using two event cameras is able to detect different kinds of obstacles. In that figure, obstacles with completely different geometries can be detected: a small ball, a box, a whiteboard marker, a frisbee, a quadrotor, and a bowling pin. The first column reports a frame grabbed from the SEES1 camera, where the object was often not visible because of motion blur (we manually highlighted the region where the objects are in the frame with a red circle). The remaining columns depict the previously described steps of our detection algorithm, with the same color code used in Fig. 4.

Detection of multiple, simultaneous obstacles

Our pipeline is able to deal with multiple obstacles moving in the scene simultaneously thanks to the clustering process and measurements' association step described in Materials and Methods. Figure S6 shows an example where the proposed algorithm correctly detects and clusters the events belonging to three different moving obstacles in the scene. In this case, three small-sized balls (manually highlighted by a red circle to facilitate the reader's understanding) were thrown in front of the camera, and the algorithm successfully associated each event with the respective object.

The main limitation of our approach is due to the fact that, when two or more obstacles are very close to each other in the frame containing the events belonging to dynamic objects, it is very hard, if not impossible, to disambiguate among them. This is due to the fact that no prior information about the obstacles is used (e.g., shape or, in the stereo case, size), as well as not exploiting any intensity information (i.e., the frames from the onboard camera) to tell apart objects that are impossible to segment out using only events. In our experimental evaluation, this turned out not to be a real issue for the system itself, because as soon as the overlapping obstacles moved away from each other, the system was able to detect them promptly and treat them as separate entities.

Obstacle avoidance framework validation

To validate our obstacle avoidance framework, we conducted a large set of experiments in real-world scenarios. The experiments were executed in two different scenarios: one indoors and the other one outdoors. The indoor experiments were conducted within a motion capture system, in the same setup we used in our previous work (*8*), and the aim was twofold: (i) collecting ground truth data to verify the effectiveness of the framework in situations where a collision with the obstacle would have happened (which was checked in post-processing thanks to the data from the motion capture) and (ii) validating the overall framework in an easier setup before moving to more complex scenarios. We used the same quadrotor platform we presented in (*8*) for the indoor experiments, equipped with a

monocular setup. Conversely, the outdoor experiments were conducted using a different vehicle, equipped with a stereo setup.

Indoor experiments

The main goal of the indoor experiments was to determine the effectiveness of our framework to avoid dynamic obstacles by determining whether a collision was actually prevented by analyzing the data coming from a motion capture system. The indoor experiments were realized using the same platform described in (*8*), in the monocular setup. We repeatedly threw a ball of known size toward the quadrotor, which used the event camera to detect and avoid it. Using the ground truth measurements coming from the OptiTrack motion capture system, we could intersect the trajectory of the ball with the position where the vehicle was hovering to determine whether, without the execution of the escape maneuver, the ball would have hit the vehicle or not. The outcome of this analysis is that our algorithm was capable of preventing actual collisions between a flying robot and dynamic obstacles, at relative speeds up to 10 m/s, as confirmed by the ground truth data about the trajectory of the object provided by the motion capture system.

Figure 5 shows one of the indoor experiments, reporting four snapshots recorded with a static camera. The ball took about 0.25 s to reach the vehicle from the moment it was thrown (Fig. 5A). At that time, as shown in Fig. 5D, the quadrotor already moved to the side to prevent the collision, showing that the algorithm successfully detected the ball and planned an evasive maneuver with very low latency.

Outdoor experiments

After evaluating the performance of our framework in an indoor setup, we performed outdoor experiments using the quadrotor platform equipped with two Insightness SEES1 cameras in a stereo setup. We executed two types of experiments, namely, in a static scenario, where the vehicle hovers at the desired position, and in a dynamic scenario, where the robot flies toward a target location. In both cases, we threw different kinds of objects toward the quadrotor, which only relied on the two event cameras to detect them and avoid them.

We tested the performance of our algorithm in static scenarios with different types of objects, with multiple obstacles moving toward it at the same time, as well as throwing them consecutively one after the other to benchmark the restiveness out of the overall approach. The vehicle successfully managed to detect them and avoid them most of the time, although in some cases, the detection was not successful and led to a collision between the robot and the obstacles. Later, we discuss the major failure causes of our algorithm; nevertheless, in outdoor experiments, the algorithm successfully detected and avoided the obstacles thrown toward it more than 90% of the time.

Figure 6 shows four snapshots captured from a sequence recorded in a dynamic scenario. The robot moved toward a target position, from left to right in the pictures, at a linear speed of 1.5 m/s. While reaching its destination, the robot detected the yellow ball thrown toward it (shown on the right side of Fig. 6A). The vehicle decided to execute an evasive maneuver upward while keeping its travel speed toward the desired position constant. This resulted in a maneuver that simultaneously allowed the vehicle to proceed along with its task and avoid a collision. Additional experiments in a dynamic situation are shown in movie S1.

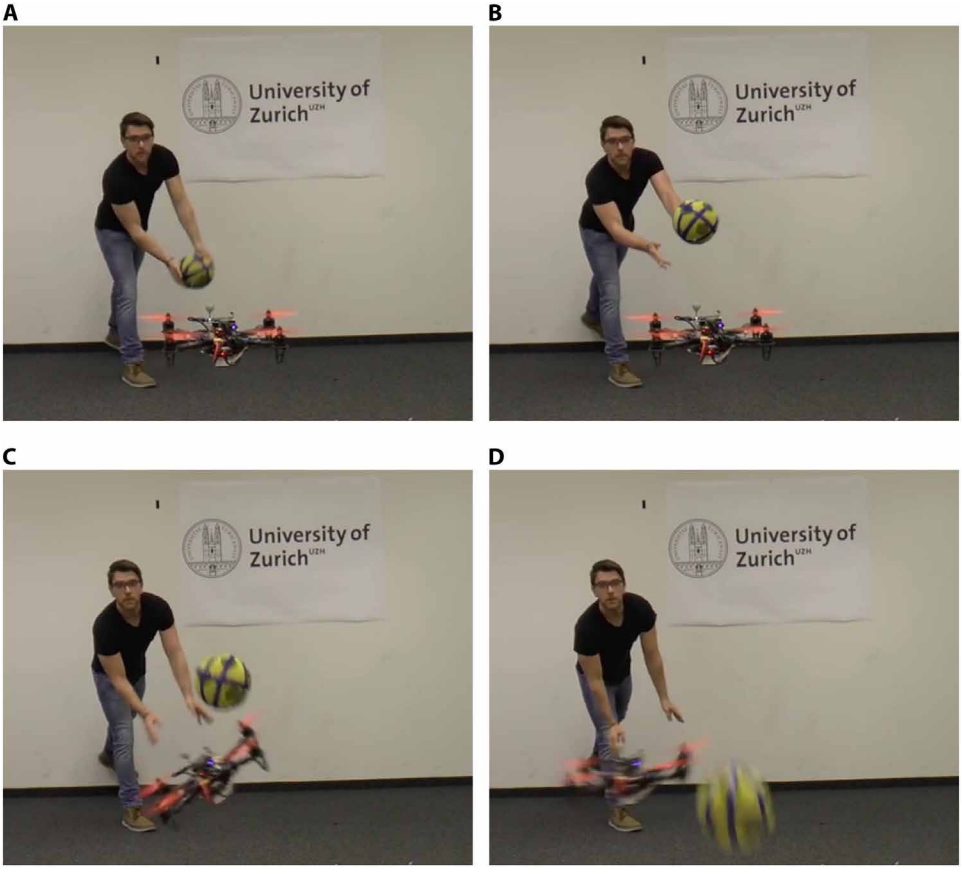

Fig. 5. A sequence from one of the indoor experiments. A ball is thrown toward the vehicle, equipped with a monocular event camera, which is used to detect and evade the obstacle. (**A**) $t = 0$ s. (**B**) $t = 0.075$ s. (**C**) $t = 0.15$ s. (**D**) $t = 0.225$ s. The ball is thrown at time $t = 0$ s and reaches the position where the quadrotor is hovering at about time $t = 0.225$ s. The robot successfully detects the incoming obstacle and moves to the side to avoid it.

Limitations of event sensors

As we have previously shown, event cameras allow fast, low-latency detection of moving obstacles. We discussed the advantages of these bioinspired neuromorphic sensors against standard cameras. However, as of today, they are mostly a research-oriented sensor and thus still require a substantial engineering effort to solve the main issues they present.

One of the problems with current event cameras is their weight. Most of the event cameras available nowadays are larger and heavier than state-of-the-art standard cameras for robotic applications, which are typically below 50 g. The Insightness SEES1 is, to the best of our knowledge, the smallest event camera that also provides frames (which is particularly convenient to easily calibrate the intrinsic and extrinsic parameters of the sensor) and can be easily mounted on a quadrotor (its size is 3.5 cm by 3.5 cm,

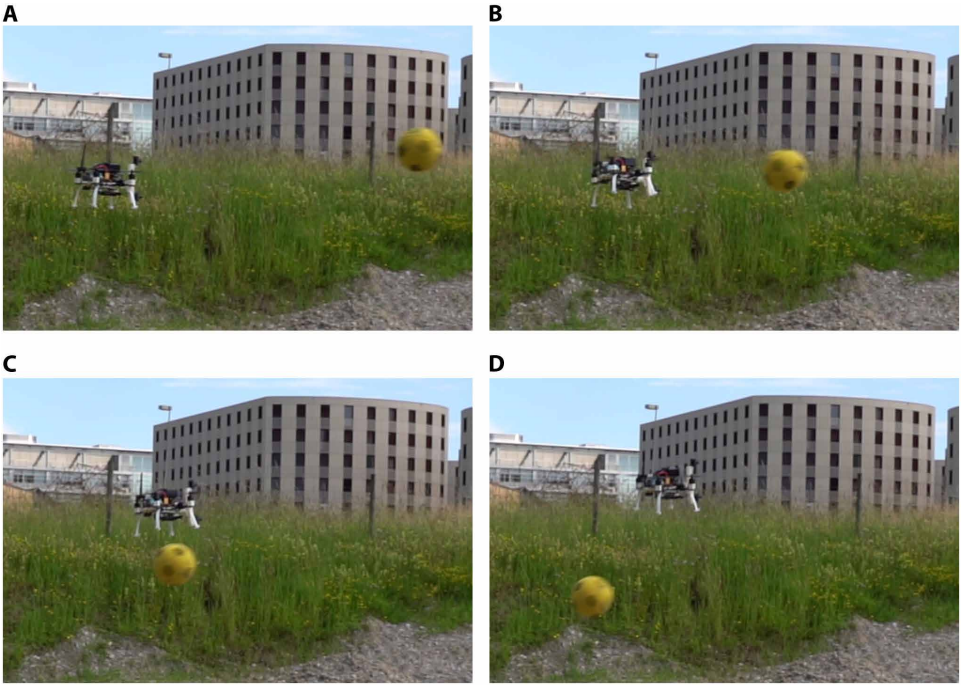

Fig. 6. A sequence from our outdoor experiments. (**A**) $t = 0$ s. (**B**) $t = 0.15$ s. (**C**) $t = 0.30$ s. (**D**) $t = 0.45$ s. The quadrotor is flying toward a reference goal position when an obstacle is thrown toward it. The obstacle is successfully detected using a stereo pair of event cameras and is avoided by moving upward.

and it weighs 15 g). However, its resolution [320 pixels by 240 pixels, QVGA (Quarter Video Graphics Array)] is particularly low compared with standard cameras. This imposes the necessity to find the right trade-off between the field of view and the angular resolution: The larger the former, the smaller the latter, which reduces the sensing range at which it is possible to detect objects reliably (*8*). A small field of view, however, has a negative impact on the detection of obstacles entering the sensing range of the vehicle from the side, as, for example, in our outdoor dynamic experiments: The larger the field of view, the earlier the vehicle can detect and avoid obstacles moving toward it from the sides.

Another problem characterizing these sensors is their noise characteristics. These sensors show higher noise than standard cameras, which often has a negative impact on the performance of event-based vision algorithms. In our approach, for example, to obtain reliable detections and to eliminate false positives caused by the sensor noise, we had to substantially increase the threshold used to separate events generated by the static part of the scene from those caused by moving objects. This resulted in an obstacle detection algorithm less reactive to small relative motion, especially at large distances. For this reason, we discarded all the detections reporting distances between the camera and the obstacle beyond 1.5 m.

The aforementioned reasons represent the main failure causes of our approach. In most of the cases, when our quadrotor was not able to avoid an object thrown toward it, this was due to the fact that it was detected too late, either because it entered the field of view of the camera at a distance that was too short (and therefore the vehicle could not complete the evasive maneuver in time) or because the motion of the obstacle did not generate sufficient events to allow our algorithm to detect it.

CONCLUSIONS

We presented a framework that enables a quadrotor to dodge fast-moving obstacles using only onboard sensing and computing. Different from conventional cameras, we used event cameras—neuromorphic sensors with reaction times of microseconds. Each pixel of an event camera reacts to changes in intensity, making this sensor a perfect fit for detecting and avoiding dynamic obstacles. Event cameras can overcome the physical limitations of standard cameras in terms of latency but require unconventional algorithms to process the asynchronous stream of events they generate. We investigated the exploitation of the temporal statistics of the event stream to tell apart the dynamic part of a scene, showing that it is possible to detect moving objects with a perception latency of 3.5 ms. We showed that our algorithm is capable of accurately and reliably detecting multiple simultaneous objects with different shapes and sizes. We combined our event-based detection algorithm with a fast strategy to generate commands that allow the vehicle to dodge incoming objects. We validated our approach with extensive experiments on a real quadrotor platform, both indoors and outdoors, demonstrating the effectiveness of the method at relative speeds up to 10 m/s.

MATERIALS AND METHODS

Obstacle detection

This section describes how our event-based algorithm to detect moving obstacles works. An additional explanation of the working principle of this algorithm is provided in movie S2.

Ego-motion compensation of the events

An event camera generates events when intensity changes occur in the image. This can happen because of either moving objects or the ego-motion of the sensor. Because we are only interested in avoiding moving objects, the first step is to remove all data generated by the quadrotor's ego-motion.

One way of removing ego-motion from an event stream is described by Mitrokhin *et al.* (*23*). This approach does, however, use an optimization routine to estimate the ego-motion, which is computationally demanding and, therefore, introduces latency in the perception system. In this work, we replaced the optimization step with a more simple and computationally efficient ego-motion compensation algorithm. To do this, we used the IMU's angular velocity average over the time window where the events were accumulated to estimate the ego-rotation and use this rotation to warp the events in the image. Our approach does not consider the translational motion of the camera but rather assumes that the events are generated mostly by rotational motion. To compensate for the translational motion, it would be necessary to estimate the depth of the points generating each event, which would increase the computational

complexity too much to be practical. As long as the distance to stationary objects is large enough, our system is not substantially affected by this assumption. In addition, an analysis of the impact of neglecting the linear component of the robot's ego-motion is provided in the Supplementary Materials. This choice allows our pipeline to be fast enough to guarantee real-time performance but comes at the cost of a potentially higher amount of noncompensated events. To cope with this, we tuned the parameters of our algorithm, whose working principle is described below, so that it is able to filter out most of the events generated by the static part of the scene.

The first step of our algorithm requires the collection of a batch of events and IMU data over a specified time δt. In our experiments, we used a time window of length δt = 10 ms, because we realized that this value represents a good compromise between sensitivity and real-time performance. A very short time window renders the entire algorithm too little sensitive, because the events collected do not contain enough information to perform reliable detection. On the other hand, increasing the time window too much leads to a very high number of events to process, making the entire algorithm slower, and does not provide much added value, because the additional events are generated by the past history of the obstacle motion. Next, we average the IMU's angular velocity over δt as $\overline{\omega} = \sum_{\delta t} \omega_t$. We then apply the Rodrigues rotation algorithm to build the rotation matrix from $\overline{\omega}\delta t$ (*42*). Each event e_i of the batch is then warped in the image plane by $\overline{\omega}(t_i - t_0)$, where t_0 is the time stamp of the first event of the batch and t_i is the time stamp of event e_i. This warping is described by a field $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ that warps the events' 2D displacement as $\phi(x, y, t - t_0) : (x, y, t) \rightarrow (x', t', t)$. These motion-compensated events are denoted by

$$C' = \Pi\{\phi(C)\} = \Pi\{\phi(x, y, t - t_0)\} = \{x', y', t_0\} \; \forall \{x, y, t\} \in C \quad (1)$$

The original event position (x, y) is part of a discretized image plane in $\mathbb{N}^2$, whereas (x', y') are part of $\mathbb{R}^2$. From the warped events, we construct the event-count image I, where the pixel value records the total number of events mapped to it by the event trajectory

$$\xi_{ij} = \{\{x', y', t\} : \{x', y, t_0\} \in C', i = x', j = y'\} \quad (2)$$

Here, $(i, j) \in \mathbb{N}^2$ denotes the integer pixel coordinates of the discretization bin for $(x', y') \in \mathbb{R}^2$. From this, we construct the event-count pixel I_{ij} as $I_{ij} = |\xi_{ij}|$, with $|A|$ being the cardinality of the set A. Next, we construct the time-image T, which is also in the discretized plane $\mathbb{N}^2$. Here, each pixel contains the average time stamp of the warped events as

$$T_{ij} = \frac{1}{I_{ij}} \Sigma t : t \in \xi_{ij} \quad (3)$$

To determine which pixels belong to a moving object or the background, they are each given a score $\rho(i, j) \in [-1, 1]$ for $\{i, j\} \in T$ as

$$\rho(i, j) = \frac{T(t : t \in \xi_{ij}) - \text{mean}(T)}{\delta t} \quad (4)$$

These scores produce the so-called normalized mean time stamp image ρ. Now, if $\rho(i, j) \geq \tau_{\text{threshold}}$, with $\tau_{\text{threshold}}$ being a specified threshold, the pixel belongs to a moving object, otherwise to the background.

Whereas the original approach (*23*) uses a fixed threshold to distinguish between ego-motion–generated events and those generated by a moving object, we instead use a linear function that depends on the angular velocity's magnitude, i.e., $\tau_{\text{threshold}}(\omega) = a \cdot \|\omega\| + b$. Here, a and b are design parameters, where b regulates the threshold, whereas the camera is static, and a increases it with an increase in the angular velocity's magnitude. This has the advantage that it is easier to detect moving objects, whereas the quadrotor is static, while still reducing the increased noise generated by faster rotational velocities. After thresholding, it can happen that some events belonging to the static part of the scene are not filtered out, generating some salt-and-pepper noise that we remove using morphological operations.

It is important to notice that, because our algorithm relies only on the IMU to perform ego-motion compensation, it is less computationally demanding than the approach in (*23*) but at the same time also more sensitive to false-positive detection generated by the ego-motion of the vehicle. To cope with this, we adopted fairly large values for the threshold parameters to avoid false positive. This came at the cost of a less sensitive detection algorithm (i.e., it discards more detections than it would with lower thresholds), and therefore, we had to find a good compromise between sensitivity and reliability.

Figure 4 shows our algorithm in action. All the events generated in the last time window (Fig. 4B) are motion-compensated using the IMU, and, for each pixel, we compute the normalized mean time stamp (Fig. 4D), which is then thresholded (Fig. 4E) to obtain a frame containing only events belonging to moving obstacles (Fig. 4F).

The same algorithm running across different consecutive time windows is shown in fig. S8. Each column corresponds to a different time, with the first row reporting the frame captured by the onboard camera, the second row showing the events collected in the window, and the third row presenting the same events after the ego-motion compensation and thresholding of the normalized mean time stamp.

Obstacle segmentation

After performing the ego-motion compensation of the events that fired in the last time window, we obtained a frame containing the location of the events belonging to the dynamic part of the scene (Fig. 4F). It is important to note that at this point, our algorithm already discarded all the static parts of the scene, with very little computational cost. To do so with a standard camera, one has to receive at least two frames to be able to distinguish between static and dynamic objects, and each frame needs to be entirely processed. The output of an event camera, instead, is much more sparse, allowing us only to process the pixels where at least one event fired.

Clustering

The thresholded image created by the ego-motion compensation described in "Ego-motion compensation of the events" can include multiple moving obstacles, as well as noise. Therefore, the next step is to separate the image points of the individual objects, as well as the noise.

The goal for the system is to be capable of handling an arbitrary number of obstacles, as well as being robust against noise. In addition, because of the low-latency requirement, the clustering has to be performed in the shortest time possible. With these requirements, we evaluated different algorithms to decide on the best fitting one for our system.

Our experimental evaluation highlighted that the density-based spatial clustering of applications with noise (DBSCAN) algorithm

(*43*) has all the required characteristics for low-latency detection of moving objects. It detects clusters without previous knowledge about their shape or their amount. In addition, it can handle noise by combining it into a separate category. It has an average time complexity of $O(n \log(n))$ and a maximum one of $O(n^2)$ but without the need for an iterative solution, which makes it comparatively fast. Another advantage is that its cost function can be arbitrarily chosen and therefore be optimized for our system. Besides the cost function, it only has two design parameters: the minimum number of data points within a cluster and the maximum cost ϵ for choosing whether a data point belongs to a given cluster. A detailed description of it is found in (*43*).

Optical flow

The density of image points and their distance in the image plane depend on the objects' velocity, distance to the sensor, and their overall size. Having only the mean time stamp information and image position resulting from the ego-motion compensation, as described in "Ego-motion compensation of the events," makes it impossible to effectively cluster the image points of objects with different velocities and distances from the dynamic vision sensor (DVS). Therefore, we require additional features. One available possibility is to calculate the image points' optical flow and therefore get an estimate of their image plane velocity. An added advantage is that two objects that generate image point clusters in close proximity to each other but move in different directions are easier to distinguish. Ideally, one would directly calculate the optical flow from the event data, but existing algorithms for this either only produce the velocities magnitude or direction or are extremely computationally expensive while having a low accuracy as evaluated in (*44*). Instead, we decided to use a conventional optical flow algorithm on the unthresholded normalized mean time stamp image produced by the ego-motion compensation. The high temporal resolution of the DVS and high update frequency of our system allow us to assume that the displacement between two frames is small and approximately constant in a region around an image point. Therefore, we used the Lucas-Kanade algorithm (*45*), which has the advantages that it is less sensitive to noise compared with pointwise methods, and by combining the information of several nearby points, it is better at handling the ambiguity of the optical flow equations. To increase the robustness of the optical flow, we applied an averaging filter both to the input images and to the resulting velocity field.

Combined clustering algorithm

To maximize the accuracy of the clustering, we use all the available data information: the image position $\mathbf{p}$, the normalized mean time stamp value ρ, and the through optical flow estimated velocity $\mathbf{v}$. With these quantities, we constructed the DBSCAN's cost function as

$$C_{i,j}(\mathbf{p}, \mathbf{v}, \rho) = w_p \|\mathbf{p}_i - \mathbf{p}_j\| + w_v \|\mathbf{v}_i - \mathbf{v}_j\| + w_\rho \mid \rho_i - \rho_j \mid \quad (5)$$

Here, $\mathbf{w} = [w_p, w_v, w_\rho]^T$ is a weight vector for the influence of the individual parts.

Even though the DBSCAN algorithm is quite efficient with a maximum data size scaling of $O(n^2)$, the computation time increases with the data size. Especially for fast-moving objects or ones that move close to the sensor, the density of the generated events and, therefore, the overall data size to be clustered increase. This leads to far greater computation time. To overcome this, we performed a preclustering step of the image points using an eight-way connected components clustering algorithm. For this, we assume that two image points that are located directly next to each other in the image plane always belong to the same object. We then calculate the mean velocity of the image points belonging to the cluster, as well as the mean normalized mean time stamp, and fit a rotated rectangle around the points. The DBSCAN's cost function is adapted to the new features. Instead of using the individual point's velocity and normalized mean time stamp, we use their corresponding mean values, whereas the difference in position is substituted by the minimal distance of the corresponding rectangles as

$$C_{i,j} = w_p \, \mathrm{dist}_{\min}(r_i, r_j) + w_v \|\mathbf{v}_{\mathrm{mean},i} - \mathbf{v}_{\mathrm{mean},j}\| + w_\rho \mid \rho_{\mathrm{mean},i} - \rho_{\mathrm{mean},j} \mid \quad (6)$$

If two corresponding rectangles should overlap, their distance is set to zero. Instead of using rectangles, ellipses could have been used, but finding the minimal distance between two ellipses requires the root calculation of a fourth-order polynomial, requiring an iterative solution, which takes drastically more time. Because the connected components algorithm has a time complexity of $O(n)$ and reduces the DBSCAN's data size by orders of magnitude, the overall clustering computation time was decreased, on average, by a factor of 1000.

3D position estimation

After receiving a set of cluster points, we first fit a rotated rectangle around them to reduce the data dimensionality. From this, we get the four corner points, as well as the center position in the image plane.

For the next step, the estimation of the obstacle's depth toward the image plane, we have to distinguish between the monocular and stereo case.

Monocular case. Because we are not able to calculate the depth of an image point from a single monocular image, we instead limit our system to objects of known size. With the added size information, we can then estimate the depth of an object in the camera's frame of reference as

$${}_C\hat{Z} = \frac{f \omega_{\mathrm{real}}}{\hat{\omega}} \quad (7)$$

where f is the focal length, ω_{real} is the width of the object, and $\hat{\omega}$ is the measured side length of the fitted rectangle.

Stereo case. For the stereo case, we use the disparity between two corresponding clusters of the stereo image pair for the depth estimation. This allows the algorithm to function with objects of unknown size. To determine cluster correspondences, we use a matching scheme minimizing the cost

$$C = w_p \mid x_{c,\mathrm{top}} - x_{c,\mathrm{bottom}} \mid + w_a \max\left(\frac{A_{\mathrm{top}}}{A_{\mathrm{bottom}}}, \frac{A_{\mathrm{bottom}}}{A_{\mathrm{top}}}\right) + w_n \max\left(\frac{n_{\mathrm{top}}}{n_{\mathrm{bottom}}}, \frac{n_{\mathrm{bottom}}}{n_{\mathrm{top}}}\right) - 2 \quad (8)$$

with $\mathbf{w} = (w_p, w_a, w_n)$ being weights, x_c being the cluster's center's position in the image plane, A being the fitted rectangle's area, and n being the number of cluster points. Next, we use the cluster's disparity to calculate the depth as described in (*46*). To increase the robustness, we use the cluster's centers to estimate the depth instead of directly projecting the corner points into 3D space. Having estimated

the obstacle's depth, we approximate its size using the rearranged formulation as in the monocular case as

$$\omega_{\text{est}} = \frac{{}_C\hat{Z}\,\hat{\omega}}{f} \tag{9}$$

Image to world projection

With the obtained obstacle's depth and size, we now project the cluster's corner and center points into 3D space using the perspective projection model in homogeneous coordinates

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}_C \mathbf{X}_i \tag{10}$$

with $\mathbf{K}$ being the intrinsic camera matrix, λ_i being the scale factor, ${}_C\mathbf{X}_i$ being the Cartesian coordinates of each point of the cluster in the camera frame, and u_i and v_i being the pixel coordinates of their projection in the image. The points ${}_C\mathbf{X}_i$ are then transformed into the world's frame of reference by applying

$$\begin{bmatrix} {}_W X_i \\ 1 \end{bmatrix} = \mathbf{T}_{\text{WB}} \mathbf{T}_{\text{BC}} \begin{bmatrix} {}_C X_i \\ 1 \end{bmatrix} \tag{11}$$

where $\mathbf{T}_{WB}$ and $\mathbf{T}_{BC}$ are transformation matrices representing the pose (rotation and translation) of the body frame with respect to the world frame and of the camera with respect to the body frame, respectively. Here, the center point's depth is both increased and decreased by the obstacle's estimated size as

$${}_C\hat{Z}_{C,\pm} = {}_C\hat{Z} \pm \omega_{\text{est}} \tag{12}$$

This gives us a total of six points ${}_W\mathbf{X}_{1:6}$ representing the obstacle.

Obstacle correspondence

To estimate an obstacle's velocity, we first have to determine whether a newly detected obstacle corresponds to a previous one and, if this is the case, to which. This is done by finding the best match between the new obstacle's center and the predicted position of the saved obstacles' centers. This is done by finding the closest position match within a sphere around the newly detected obstacle.

Obstacle velocity estimation

Once the 3D position of an obstacle has been estimated, our algorithm requires some further processing to provide valuable information to the planning stage for a twofold reason: (i) The event-based detections are sometimes noisy, especially at large distances; (ii) it is necessary to estimate the obstacle's velocity, which is used to determine the avoidance direction, as well as a scaling factor for the repulsive potential field ("obstacle avoidance"). To do so, we use a Kalman filter (*47*), with the obstacle's position estimate as input for the measurement update. This introduces some time lag (typically below 0.3 ms) because the Kalman filter behaves as a low-pass filter, but the increased accuracy is, in this case, preferable. For this, we assume a constant velocity model having as state the obstacle's position and velocity

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \dot{\mathbf{x}}_{k-1}\,\Delta t \tag{13}$$

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} \tag{14}$$

$$\Delta t = t_k - t_{k-1} \tag{15}$$

With this, we can formulate the linear motion model as

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{16}$$

$$\hat{\mathbf{x}}_k = \mathbf{A}_k\,\hat{\mathbf{x}}_{k-1} + \mathbf{v}_k \tag{17}$$

$$\mathbf{z}_k = \mathbf{H}\,\hat{\mathbf{x}}_k + \mathbf{w}_k \tag{18}$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{I}_{3x3} & \Delta t \cdot \mathbf{I}_{3x3} \\ 0_{3x3} & \mathbf{I}_{3x3} \end{bmatrix} \tag{19}$$

$$\mathbf{H} = [\mathbf{I}_{3x3}\ 0_{3x3}] \tag{20}$$

where $\mathbf{v} \sim \mathcal{N}(0, \mathbf{Q})$ is the normally distributed process noise and $\mathbf{w} \sim \mathcal{N}(0, \mathbf{R})$ is the normally distributed measurement noise. Using this, we now construct the Kalman filter as follows (*47*)

$$\hat{\mathbf{x}}_{p,k} = \mathbf{A}_k\,\hat{\mathbf{x}}_{m,k-1} \tag{21}$$

$$\mathbf{P}_{p,k} = \mathbf{A}_k\,\mathbf{P}_{m,k-1}\,\mathbf{A}_k^T + \mathbf{Q} \tag{22}$$

$$\mathbf{K}_k = \mathbf{P}_{p,k}\,\mathbf{H}^T\,(\mathbf{H}\,\mathbf{P}_{p,k}\,\mathbf{H}^T + \mathbf{R})^{-1} \tag{23}$$

$$\hat{\mathbf{x}}_{m,k} = \hat{\mathbf{x}}_{p,k} + \mathbf{K}_k\,(\mathbf{z}_k - \mathbf{H}\,\hat{\mathbf{x}}_{p,k}) \tag{24}$$

$$\mathbf{P}_{m,k} = (\mathbf{I} - \mathbf{K}_k\,\mathbf{H})\,\mathbf{P}_{p,k}\,(\mathbf{I} - \mathbf{K}_k\,\mathbf{H})^T + \mathbf{K}_k\,\mathbf{R}\,\mathbf{K}_k^T \tag{25}$$

This has the added advantage that we receive a filtered estimate of the obstacle's velocity without any further computations.

Obstacle avoidance

The primary objective of our avoidance framework is to guarantee low latency between sensing and actuation. The low latency on the perception side is guaranteed by the previously described event-based obstacle detection pipeline. For the overall system to be effective, however, it is necessary to reduce the latency of the decision-making system responsible for driving the robot away from the detected obstacles. On the basis of this consideration, it is intuitive to understand that any optimization-based avoidance technique is not suited for our purpose because numerical optimization would introduce latency due to the non-negligible computation times. Rapid methods to compute motion primitives for aerial vehicles exist in the literature (*48*). However, they present a number of drawbacks. First, it is necessary to sample both space and time to find a safe position for the robot and a suitable duration of the trajectory. In addition, continuity in the control inputs is not always guaranteed. Last, including this kind of methods within existing motion generation frameworks is not always trivial because of multiple reasons: It is necessary to continuously switch between the main navigation algorithm, driving the robot toward its goal, and the avoidance algorithm, steering it away from obstacles; it is not always trivial to obtain a behavior that allows the robot to keep executing its mission (e.g., reach its goal) while simultaneously avoiding moving obstacles.

The artificial potential field method is a natural and simple solution to all the aforementioned issues. Given a closed-form expression of the attractive and repulsive fields, it is particularly simple to compute

their gradients within negligible computation time to generate the resulting force responsible for letting the robot move. Considering an obstacle as the source of a repulsive field also allows us to not require any sampling in space and time because the resulting potential decides in which direction the robot should move at each moment in time. Last, the resulting potential can be used at different levels of abstraction to integrate the command derived from its gradient into existing motion generation algorithms, for example, as velocity or acceleration commands.

Using potential fields for pathfinding and obstacle avoidance has been extensively researched. This approach is, however, mostly used in static scenarios, whereas our system is thought for dynamic obstacles. The typical approach is to build a discretized map, where each element represents the potential combined from the attractive and repulsive parts. This map building approach is feasible in 2D, but its size and the required computational power to build and analyze it drastically increase when doing so in 3D, as it increases from $O(n^2)$ to $O(n^3)$. Instead of building a map, we represent the obstacles as a struct of features, resulting in a sparse and minimal data representation. The obstacles are represented as ellipsoids, with a potential that is decaying over time. We use the estimated obstacles' position and velocity to calculate their repulsive forces at each time step. In addition, given a reference target position, we compute the attractive force toward it. With the combined force, we then produce a velocity command that is sent to the controller. In addition, the system's behavior, when no obstacles are present, is similar to the one generated by a high-level trajectory planner driving the robot toward the desired goal location.

Obstacle representation

We chose to represent the obstacles as ellipsoids, because they are a good representation of the expected Gaussian error of both position and size. In addition, they allow us to generate a continuous repulsive force when an obstacle is detected. Using the six coordinate points ${}_W\widehat{X}_{1:6}$ in Eq. 12, we fit a minimal volume ellipsoid around them using the approach described in (*49*) and illustrated in fig. S9.

Repulsive potential field

Each obstacle produces a potential field $U_{r,i}$, from which we get the repulsive force $\mathbf{F}_{r,i}$ by calculating its gradient as $\mathbf{F}_{r,i} = -\nabla U_{r,i}$. One way of formulating the potential field was proposed by Khosla and Volpe (*50*), which, in turn, is a modification of the original artificial potential field definition by Khatib (*40*), as

$$U_{r,i}(\eta_i) = \begin{cases} k_{r,i}\left(\frac{\eta_0 - \eta_i}{\eta_0}\right)^{\gamma}, & \text{if } 0 \le \eta_i \le \eta_0 \\ 0, & \text{if } \eta_i > \eta_0 \end{cases} \quad (26)$$

with a resulting force

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} \frac{k_{r,i}\gamma}{\eta_0}\left(\frac{\eta_0 - \eta_i}{\eta_0}\right)^{\gamma-1} \nabla \eta_i, & \text{if } 0 \le \eta_i \le \eta_0 \\ 0, & \text{if } \eta_i > \eta_0 \end{cases} \quad (27)$$

where k_r,γ, and η_0 are design parameters and η_i is the distance to the obstacle *i*. This kind of field does, however, produce a gradient whose magnitude increases slowly as the distance to the obstacle decreases, as shown in fig. S10A. This has the effect that the repulsive force acting on the quadrotor only reaches substantial values when the obstacle is close, or a high repulsive gain k_r has to be chosen, which might lead to unstable, aggressive behavior.

Therefore, we propose a new formulation of the repulsive force as

$$\left\|\mathbf{F}_{r,i}\right\| = \begin{cases} k_{r,i}\left(1 - \frac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right), & \text{if } 0 \le \eta_i \le \eta_0 \\ 0, & \text{if } \eta_i > \eta_0 \end{cases} \quad (28)$$

as shown in fig. S10B. Here, η_i is the minimal distance to the ellipsoid's surface of obstacle *i*. Through this formulation, the force's magnitude is limited to a specified value k_r and increases much faster. This is desirable when evading fast-moving obstacles, as compared with static ones, for which the fields described in other works were developed, because the quadrotor's dynamics require it to start evading before an obstacle comes too close, as discussed in (*38*).

Conventionally, the gradient of the distance toward the obstacle $\nabla\eta_i$ is responsible for the direction of the repulsive force $\mathbf{F}_{r,i}$. It points in the direction of the steepest descent of the obstacle's distance, which is the opposite direction between the quadrotor's center and the closest point on the obstacle's ellipsoid's surface. This means that an obstacle pushes the quadrotor away from it. We do, however, want to apply a different avoidance strategy. Instead, we use the obstacle's predicted velocity $\dot{\mathbf{x}}_i$ and the distance's gradient $\nabla\eta_i$ and calculate the normalized cross-product as

$$\theta_i = \frac{\nabla\eta_i \times \dot{\mathbf{x}}_i}{\|\nabla\eta_i \times \dot{\mathbf{x}}_i\|} \quad (29)$$

Next, we project this vector into the plane orthogonal to the quadrotor's heading $\theta_{\text{quadrotor}}$ as

$$\theta_{i,n} = \theta_i - < \theta_i, \theta_{\text{quadrotor}} > \theta_{\text{quadrotor}} \quad (30)$$

With the new avoidance direction $\theta_{i,n}$, the repulsive force $\mathbf{F}_{r,i}$ becomes

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} k_{r,i}\left(1 - \frac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right)\theta_{i,n}, & \text{if } 0 \le \eta_i \le \eta_0 \\ 0, & \text{if } \eta_i > \eta_0 \end{cases} \quad (31)$$

This formulation of the potential field yields to a behavior such that, if the quadrotor is moving toward the goal location, it flies around any detected obstacle if the goal position is behind it, whereas if it is in hover conditions, it moves in a direction orthogonal to the obstacle's velocity. Last, we include the magnitude of the obstacle's estimated velocity $\|\dot{\mathbf{x}}_i\|$ into the repulsive force $\mathbf{F}_{r,i}$ as

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} \|\dot{\mathbf{x}}_i\| k_{r,i}\left(1 - \frac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right)\theta_{i,n}, & \text{if } 0 \le \eta_i \le \eta_0 \\ 0 & \text{if } \eta_i > \eta_0 \end{cases} \quad (32)$$

By doing so, faster obstacles produce a larger repulsive force, and the quadrotor will therefore perform a more aggressive avoidance maneuver. This is desirable because the faster an obstacle, the lower the avoidance time, which therefore implies the necessity for a quick evasive maneuver.

In addition, we ensure that the *z* component of the repulsive force is always positive, namely, $F_{r,i,z} = |F_{r,i,z}|$, as quadrotors with sufficiently large thrust-to-weight ratios are typically capable of producing larger accelerations upward than downward.

The repulsive constant $k_{r,i}$ is, in our case, dynamic and decays with time as

$$k_{r,i}(t) = k_{r,0}\, e^{-\lambda_{\text{decay}}(t - t_{\text{detection},i})} \tag{33}$$

where $k_{r,0}$ is the initial repulsive constant, λ_{decay} is a factor regulating the decay rate, t is the current time, and $t_{\text{detection},i}$ is the last time the specific obstacle was detected. Through this decay, obstacles are kept in case of a temporary occlusion or when they leave the camera's field of view. Their effect on the quadrotor, however, decreases as the time to their last detection increases. If $k_{r,i}$ falls below a given threshold $k_{r,\tau}$, the obstacle is removed.

Last, the parameter η_i represents the minimal distance between the quadrotor's center to the obstacle's ellipsoid's surface minus the quadrotor's radius. The computation of the minimal distance between a point and an ellipsoid's surface is described in (*51*). The total repulsive force is then the sum over all individual obstacles as

$$\mathbf{F}_{r,\text{total}} = \sum_i \mathbf{F}_{r,i} \tag{34}$$

Attractive potential field

The goal of the attractive potential field is to allow the vehicle to reach the desired target position and hover there until the user provides a new reference. In this work, we provide a simple formulation for the attractive potential that assumes that no static obstacles are present in the scene, i.e., the straight-line path between the robot and the obstacle is collision free. However, one can easily replace this component of our avoidance scheme with more sophisticated methods to generate commands that drive the vehicle toward its goal. These can be based, for example, on potential field-based techniques dealing with static obstacles and local minima, which is out of the scope of this work, or completely different methods able to generate velocity or acceleration commands [for example, see (*52*)].

For the attractive potential, we want the system to not only produce the same velocity toward a goal as a high-level planner would produce if no obstacle is present but also produce stable dynamics close to the goal. Therefore, we chose the hybrid approach of a conical and polynomial potential field (*53*) as

$$U_a = \begin{cases} \dfrac{k_a}{(\gamma_a + 1)\, e_0^{\gamma_a}} \|\mathbf{e}\|^{\gamma_a + 1}, & \text{if } \|\mathbf{e}\| < e_0 \\ k_a \|\mathbf{e}\|, & \text{if } \|\mathbf{e}\| \geq e_0 \end{cases} \tag{35}$$

This function is differentiable at e_0, i.e., the crossover distance between the two different potential fields, with $\mathbf{e}$ being the error between the goal's and quadrotor's positions, k_a being the attractive constant, and γ_a being a design parameter. By taking its gradient, we get the attractive force as

$$\mathbf{F}_a = -\nabla U_a = \begin{cases} k_a \dfrac{\mathbf{e}}{\|\mathbf{e}\|} \left(\dfrac{\|\mathbf{e}\|}{e_0} \right)^{\gamma_a}, & \text{if } \|\mathbf{e}\| < e_0 \\ k_a \dfrac{\mathbf{e}}{\|\mathbf{e}\|}, & \text{if } \|\mathbf{e}\| \geq e_0 \end{cases} \tag{36}$$

which is continuous in $\mathbf{e}$. The constant k_a regulates the output velocity $\dot{\mathbf{x}}$ (see "Output velocity"), and by setting it to $k_a = \|\mathbf{v}_{des}\|$, the quadrotor's velocity's magnitude is $\|\dot{\mathbf{x}}\| = \|\mathbf{v}_{\text{des}}\|$, whereas $\|\mathbf{e}\| \geq e_0$, and no obstacles are present.

If we would instead solely rely on the conical potential field, the quadrotor would start to oscillate around its goal position, because the resulting force's magnitude would be k_a, regardless of the error. The attractive force's magnitude is shown in fig. S11. If $\gamma_a = 0$, then $\|\mathbf{F}_a\|$ is identical to that of the conical part, producing a constant magnitude of the attractive force, whereas for $\gamma_a = 1$, the magnitude goes linearly to 0. With increasing γ_a, the magnitude drops faster with an increasingly large area around $\|\mathbf{e}\| = 0$, where it is close to 0.

Output velocity

The velocity is the output of our system and is given to the controller to derive the required total thrust and body rates. From the total repulsive force $\mathbf{F}_{r,\text{total}}$ and attractive force $\mathbf{F}_a$, we get the total virtual force acting on the quadrotor as $\mathbf{F}_{\text{total}} = \mathbf{F}_{r,\text{total}} + \mathbf{F}_a$. With this force, we now have three possible design choices to calculate the quadrotor's desired velocity $\dot{\mathbf{x}}$

$$\ddot{\mathbf{x}} = \frac{\mathbf{F}_{\text{total}}}{m} \tag{37}$$

$$\dot{\mathbf{x}} = \mathbf{F}_{\text{total}} \tag{38}$$

$$\dot{\mathbf{x}} = \mathbf{F}_{\text{total}} \tag{39}$$

where m denotes the quadrotor's mass.

Both Eqs. 37 and 38 produce a first-order dynamic, whereas Eq. 39 directly produces the velocity output. Introducing further dynamics into the system results in additional delays, which is undesirable because we want our system to be as responsive as possible. We therefore chose Eq. 39 because it produces the fastest response.

Experimental platform

Hardware

To validate our approach with real-world experiments, we designed a custom quadrotor platform, shown in fig. S3. The main frame is a 6″ Lumenier QAV-RXL, and, at the end of each arm, we mounted a Cobra CM2208-2000 brushless motor equipped with a 6″, three-bladed propeller. The vehicle is equipped with two onboard computers: (i) a Qualcomm Snapdragon Flight, used for monocular, vision-based state estimation using the provided Machine Vision SDK; and (ii) an NVIDIA Jetson TX2, accompanied by an AUVIDEA J90 carrier board, running all the rest of our software stack. In this regard, the output of our framework is a low-level control command comprising the desired collective thrust and angular rates the vehicle should achieve to fly. These commands are sent to a Lumenier F4 AIO Flight Controller, which then produces single-rotor commands that are fed to DYS Aria 35a motor controllers.

The quadcopter is equipped with two front-facing Insightness SEES1 cameras, in a vertical stereo setup, connected via USB to the Jetson TX2. The SEES1 sensor provides both frame and events and has a QVGA resolution (320 pixels by 240 pixels). To have a sufficiently high angular resolution, each camera has a lens providing a horizontal field of view of about 80°. Such a small field of view is particularly low for tasks such as obstacle avoidance, where a large field of view is preferable to increase the area that the robot can sense. The choice of adopting a vertical stereo setup rather than a more common horizontal setup was driven by the necessity of maximizing the overlap between the field of view of the two cameras while guaranteeing a sufficiently large baseline (in our case, 15 cm).

In addition to the previous sensing suite, we mounted a TeraRanger EVO 60-m distance sensor looking downward. The goal of this additional sensor is to constantly monitor the height of the vehicle to detect whether there is any drift in the state estimate provided by the VIO pipeline running on the Snapdragon Flight. Whenever we detect a discrepancy beyond a manually defined threshold, the quadrotor automatically executes an emergency landing maneuver.

Software

We developed the software stack running on our quadrotor in C++ using Robot Operating System (ROS) for communication among different modules. To reduce latency, we implemented the obstacle detection and avoidance algorithms within the same ROS module, so that no message exchange is necessary between the camera drivers and the code responsible for detecting moving obstacles, as well as between the latter and the planning stage. The output of this module is a velocity command, which is then fed to the position controller proposed in (*54*) and available as open source (http://rpg.ifi.uzh.ch/rpg_quadrotor_control.html). The low-level controller, responsible for tracking desired body rates and collective thrust, is the default one provided by the Lumenier F4 AIO Flight Controller, which then communicates with the electronic speed controllers (ESCs) to generate the single rotor thrusts.

In our outdoor experiments, the state of the vehicle is estimated using the VIO pipeline provided by the Qualcomm Machine Vision SDK (https://developer.qualcomm.com/software/machine-vision-sdk), which, however, only provides new estimates at camera rate (up to 30 Hz). This is not sufficient to control our vehicle with low latency and would represent a bottleneck in the entire pipeline. To obtain a higher-rate state estimate, we feed the output of the VIO into an extended Kalman filter (*55*), together with IMU measurements, to obtain information about the position, orientation, and velocity of the vehicle at 250 Hz.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/5/40/eaaz9712/DC1
Fig. S1. Monodimensional example to explain the working principle of event-based detection of moving obstacles.
Fig. S2. Time statistics of the events belonging to static and dynamic regions.
Fig. S3. The quadrotor platform we used in our outdoor experiments.
Fig. S4. Ego-motion compensation computation time as function of the number of events.
Fig. S5. Clustering computation time as function of the pixels count.
Fig. S6. Detection of objects having different sizes and shapes.
Fig. S7. Detection of multiple objects simultaneously.
Fig. S8. Sequence of detection.
Fig. S9. Obstacle ellipsoid.
Fig. S10. Repulsive potential.
Fig. S11. Attractive potential.
Movie S1. Outdoor dynamic experiments.
Movie S2. Explanation of the working principle of the event-based detection algorithm.
References (*56*, *57*)

REFERENCES AND NOTES

1. "Aerial imaging market size, share and industry analysis by camera orientation (oblique, vertical), platform (fixed-wing aircraft, helicopter, uav/drones), end-use industry (government, energy sector, defense, forestry and agriculture, real estate, civil engineering, insurance) and regional forecast, 2018–2025," *Fortune Business Insights* (2019); www.fortunebusinessinsights.com/industry-reports/aerial-imaging-market-100069.
2. Markets and Markets, "Autonomous last mile delivery market worth $91.5 billion by 2030," *Bloomberg* (2019); www.bloomberg.com/press-releases/2019-07-15/autonomous-last-mile-delivery-market-worth-91-5-billion-by-2030-exclusive-report-by-marketsandmarkets.
3. Reports and Data, "Urban air mobility market to reach USD 7.9 billion by 2030," *Globe NewsWire* (2019); www.globenewswire.com/news-release/2019/03/18/1756495/0/en/Urban-Air-Mobility-Market-To-Reach-USD-7-9-Billion-By-2030-Reports-And-Data.html.
4. G. McNeal, "Video shows Kings fans knocking drone out of sky, did it belong to LAPD?" *Forbes* (2014); www.forbes.com/sites/gregorymcneal/2014/06/14/video-shows-kings-fans-knocking-drone-out-of-sky-did-it-belong-to-lapd/#4377a6584284.
5. T. Powell, "Bizarre moment argentine football fan takes down drone with well-aimed toilet roll as it films crowd," *Evening Standard* (2017); www.standard.co.uk/news/world/bizarre-moment-argentine-football-fan-takes-down-drone-with-wellaimed-toilet-roll-as-it-films-crowd-a3591066.html.
6. M. O. Reporter, "When eagles attack! drone camera mistaken for rival," *Daily Mail* (2016); www.dailymail.co.uk/video/news/video-1154408/Golden-Eagle-attacks-drone-camera-mistaking-rival.html.
7. A. Domanico, "Hawk attacks drone in a battle of claw versus machine," *CNet* (2014); www.cnet.com/news/this-hawk-has-no-love-for-your-drone/.
8. D. Falanga, S. Kim, D. Scaramuzza, How fast is too fast? The role of perception latency in high-speed sense and avoid. *IEEE Robot. Autom. Lett.* **4**, 1884–1891 (2019).
9. O. Esrafilian, H. D. Taghirad, Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM, in *2016 4th International Conference on Robotics and Mechatronics (ICROM)* (RSI, 2016), 26 to 28 October 2016, Tehran, Iran, pp. 240–245.
10. H. Alvarez, L. M. Paz, and D. Cremers, Collision avoidance for quadrotors with a monocular camera, in *Experimental Robotics*, M. Hsieh, O. Khatib, V. Kumar, Eds. (Springer, Cham, 2016), pp. 195–209.
11. Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, S. Shen, Autonomous aerial navigation using monocular visual-inertial fusion. *J. Field Robot.* **35**, 23–51 (2018).
12. H. Oleynikova, D. Honegger, M. Pollefeys, Reactive avoidance using embedded stereo vision for mav flight, in *2015 IEEE International Conference on Robotics Automation (ICRA)* (IEEE, 2015), Seattle, WA, USA, 26 to 30 May 2015, pp. 50–56.
13. M. Burri, H. Oleynikova, M. W. Achtelik, R. Siegwart, Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015), Hamburg, Germany, 28 September to 2 October 2015, pp. 1872–1878.
14. K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, V. Kumar, Fast, autonomous flight in gps-denied and cluttered environments. *J. Field Robot.* **35**, 101–120 (2018).
15. A. J. Barry, P. R. Florence, R. Tedrake, High-speed autonomous obstacle avoidance with pushbroom stereo. *J. Field Robot.* **35**, 52–68 (2018).
16. S. Liu, M. Watterson, S. Tang, V. Kumar, High speed navigation for quadrotors with limited onboard sensing. in *2016 IEEE International Conference on Robotics Automation (ICRA)* (IEEE, 2016), Stockholm, Sweden, 16 to 21 May 2016, pp. 1484–1491.
17. B. T. Lopez, J. P. How, Aggressive 3-D collision avoidance for high-speed navigation, in *2017 IEEE International Conference on Robotics Automation (ICRA)* (IEEE, 2017), Singapore, Singapore, 29 May to 3 June 2017, pp. 5759–5765.
18. A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, N. Roy, Visual odometry and mapping for autonomous flight using an RGB-D camera, in *Robotics Research*, H. Christensen, O. Khatib, (Springer, Cham, 2017), pp. 235–252.
19. E. Ackerman, Skydio demonstrates incredible obstacle-dodging full autonomy with new R1 consumer drone, *IEEE Spectrum* (2018); http://spectrum.ieee.org/automaton/robotics/drones/skydio-r1-drone.
20. P. Lichtsteiner, C. Posch, T. Delbruck, A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits* **43**, 566–576 (2008).
21. G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, D. Scaramuzza, Event-based vision: A survey. arXiv:1904.08405 [cs.CV] (2019).
22. E. Mueggler, B. Huber, D. Scaramuzza, Event-based, 6-dof pose tracking for high-speed maneuvers using a dynamic vision sensor. Youtube; https://youtu.be/LauQ6LWTkxM?t=32.
23. A. Mitrokhin, C. Fermüller, C. Parameshwara, Y. Aloimonos, Event-based moving object detection and tracking, in *2018 IEEE/RSJ International Conference Intelligent Robots Systems (IROS)* (IEEE, 2018), Madrid, Spain, 1 to 5 October 2018, pp. 1–9.
24. T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, D. Scaramuzza, Event-based motion segmentation by motion compensation, in *IEEE International Conference on Computer Vision (ICCV)* (IEEE, 2019), pp. 7244–7253.
25. L. Salt, D. Howard, G. Indiveri, Y. Sandamirskaya, Differential evolution and bayesian optimisation for hyper-parameter selection in mixed-signal neuromorphic circuits applied to UAV obstacle avoidance (2017); http://arxiv.org/abs/1704.04853.
26. M. B. Milde, O. J. N. Bertrand, H. Ramachandran, M. Egelhaaf, E. Chicca, Spiking elementary motion detector in neuromorphic systems. *Neural Comput.* **30**, 2384–2417 (2018).
27. A. Censi, Efficient neuromorphic optomotor heading regulation, in *2015 American Control Conference (ACC)* (IEEE, 2015), Chicago, IL, USA, 1 to 3 July 2015, pp. 3854–3861.
28. E. Mueller, A. Censi, E. Frazzoli, Low-latency heading feedback control with neuromorphic vision sensors using efficient approximated incremental inference, in *2015 54th IEEE Conference on Decision Control (CDC)* (IEEE, 2015), Osaka, Japan, 15 to 18 December 2015, pp. 992–999.

29. A. Glover, C. Bartolozzi, Event-driven ball detection and gaze fixation in clutter, in *2016 IEEE/RSJ International Conference on Intelligent Robots System (IROS)* (IEEE, 2016), Daejeon, South Korea, 9 to 14 October 2016, pp. 2203–2208.
30. A. Glover, C. Bartolozzi, Robust visual tracking with a freely-moving event camera, in *2017 IEEE/RSJ International Conference Intelligent Robots and System (IROS)* (IEEE, 2017), Vancouver, BC, Canada, 24 to 28 September 2017, pp. 3769–3776.
31. J. Conradt, R. Berner, M. Cook, T. Delbruck, An embedded AER dynamic vision sensor for low-latency pole balancing, in *2009 IEEE 12th International Conference on Computer Vision Workshop, ICCV Workshop,* (IEEE, 2009), Kyoto, Japan, 27 September to 4 October 2009, pp. 780–785.
32. T. Delbruck, M. Lang, Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Front. Neurosci.* **7**, 223 (2013).
33. T. Delbruck, P. Lichtsteiner, Fast sensory motor control based on event-based hybrid neuromorphic-procedural system, in *2017 IEEE International Symposium on Circuits System (ISCAS)* (IEEE, 2007), New Orleans, LA, USA, 27 to 30 May 2007, pp. 845–848.
34. X. Clady, C. Clercq, S.-H. Ieng, F. Houseini, M. Randazzo, L. Natale, C. Bartolozzi, R. Benosman, Asynchronous visual event-based time-to-contact. *Front. Neurosci.* **8**, 9 (2014).
35. F. Galluppi, C. Denk, M. C. Meiner, T. C. Stewart, L. A. Plana, C. Eliasmith, S. Furber, J. Conradt, Event-based neural computing on an autonomous mobile platform, in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2014), Hong Kong, China, 31 May to 7 June 2014, pp. 2862–2867.
36. H. Blum, A. Dietmüller, M. Milde, J. Conradt, G. Indiveri, Y. Sandamirskaya, A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor, in *Robotics: Science and Systems (RSS)* (2017).
37. A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, D. Scaramuzza, Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios. *IEEE Robot. Autom. Lett.* **3**, 994–1001 (2018).
38. E. Mueggler, N. Baumli, F. Fontana, D. Scaramuzza, Towards evasive maneuvers with quadrotors using dynamic vision sensors, in *2015 European Conference Mobile Robots (ECMR)*, Lincoln, UK, 2 to 4 September 2015, pp. 1–8.
39. N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermuller, D. Scaramuzza, Y. Aloimonos, Evdodge: Embodied AI for high-speed dodging on a quadrotor using event cameras. arXiv:1906.02919 [cs.RO] (2019).
40. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in *Autonomous Robot Vehicles* (Springer, 1986), pp. 396–404.
41. L. Chittka, P. Skorupski, N. E. Raine, Speed–accuracy tradeoffs in animal decision making. *Trends Ecol. Evol.* **24**, 400–407 (2009).
42. J. E. Mebius, Derivation of the euler-rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations. arXiv:math/0701759 [math.GM] (2007).
43. M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the 2nd International Conference on Knowledge Discovery Data Mining*, vol. 96, Portland, OR, 1996, pp. 226–231.
44. B. Rueckauer, T. Delbruck, Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Front. Neurosci.* **10**, 176 (2016).
45. S. Baker, I. Matthews, Lucas-Kanade 20 years on: A unifying framework. *Int. J. Comput. Vis.* **56**, 221–255 (2004).
46. A. Fusiello, Elements of geometric computer vision, University of Edinburgh - School of Informatics, (16.09.2012); http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html#x1-130004.
47. R. Kalman, A new approach to linear filtering and prediction problems. *J. Basic Eng.* **82**, 35–45 (1960).
48. M. W. Mueller, M. Hehn, R. D'Andrea, A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Trans. Robot.* **31**, 1294–1310 (2015).
49. N. Moshtagh, Minimum volume enclosing ellipsoid. *Convex Optim.* **111**, 112 (2005).
50. P. Khosla, R. Volpe, Superquadric artificial potentials for obstacle avoidance and approach, in *Proceedings of the 1998 IEEE Interantional Conference on Robotics Automation (ICRA)* (IEEE, 1988), Philadelphia, PA, USA, 24 to 29 April 1988, pp. 1778–1784.
51. D. Eberly, Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid, Geometric Tools, LLC, (2011); www.geometrictools.com/Documentation/DistancePointEllipseEllipsoid.pdf.
52. T. Cieslewski, E. Kaufmann, D. Scaramuzza, Rapid exploration with multi-rotors: A frontier selection method for high speed flight, in *IEEE/RSJ International Conference Intelligent Robots and System (IROS)* (IEEE, 2017), Vancouver, BC, Canada, 24 to 28 September 2017, pp. 2135–2142.
53. B. Siciliano, O. Khatib, *Springer Handbook of Robotics* (Springer Publishing Company, Incorporated, ed. 2, 2016).
54. M. Faessler, A. Franchi, D. Scaramuzza, Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.* **3**, 620–626 (2018).
55. S. Lynen, M. Achtelik, S. Weiss, M. Chli, R. Siegwart, A robust and modular multi-sensor fusion approach applied to MAV navigation, in *IEEE/RSJ International Conference Intelligent Robots and System (IROS)*, (IEEE, 2013), Tokyo, Japan, 3 to 7 November 2013, pp. 3923–3929.
56. P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* (Springer, 2011), vol. 73.
57. T. Delbruck, V. Villanueva, L. Longinotti, Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision, in *2014 IEEE International Symposium Circuits and Systems (ISCAS)* (IEEE, 2014), Melbourne VIC, Australia, 1 to 5 June 2014, pp. 2636–2639.

Acknowledgments: We thank H. Rebecq and G. Gallego for helpful discussions. **Funding:** This work was supported by the SNSF-ERC Starting Grant and the Swiss National Science Foundation through the National Center of Competence in Research (NCCR) Robotics. **Author contributions:** The project ideas were conceived by D.F., K.K., and D.S. The experiments were designed and performed by D.F. and K.K. The paper was written by D.F., K.K., and D.S. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials.

Submitted 24 October 2019
Accepted 18 February 2020
Published 18 March 2020
10.1126/scirobotics.aaz9712

Citation: D. Falanga, K. Kleber, D. Scaramuzza, Dynamic obstacle avoidance for quadrotors with event cameras. *Sci. Robot.* **5**, eaaz9712 (2020).

Science Robotics

Dynamic obstacle avoidance for quadrotors with event cameras

Davide Falanga, Kevin Kleber, and Davide Scaramuzza

Sci. Robot. **5** (40), eaaz9712. DOI: 10.1126/scirobotics.aaz9712

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.