Towards Evasive Maneuvers with Quadrotors using Dynamic Vision Sensors

Elias Mueggler, Nathan Baumli, Flavio Fontana and Davide Scaramuzza
Robotics and Perception Group, University of Zurich, Switzerland
mueggler@ifi.uzh.ch, http://rpg.ifi.uzh.ch

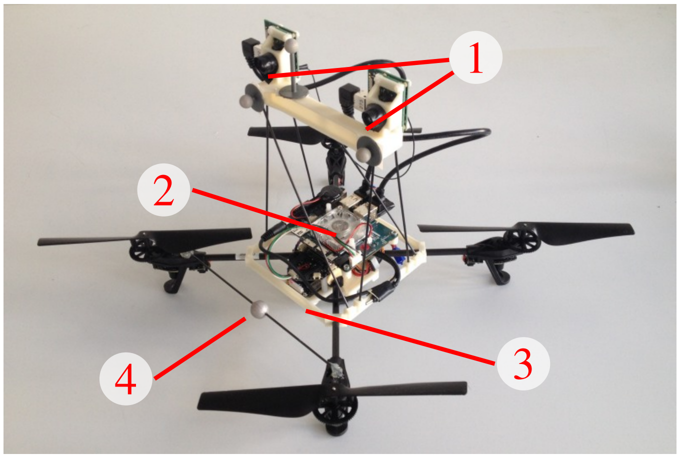
***Abstract*—We present a method to predict collisions with objects thrown at a quadrotor using a pair of dynamic vision sensors (DVS). Due to the *micro*-second temporal resolution of these sensors and the sparsity of their output, the object's trajectory can be estimated with minimal latency. Unlike standard cameras that send frames at a fixed frame rate, a DVS only transmits pixel-level brightness *changes* ("events") at the time they occur. Our method tracks spherical objects on the image plane using probabilistic trackers that are updated with each incoming event. The object's trajectory is estimated using an Extended Kalman Filter with a mixed state space that allows incorporation of both the object's dynamics and the measurement noise in the image plane. Using error-propagation techniques, we predict a collision if the 3σ-ellipsoid along the predicted trajectory intersects with a safety sphere around the quadrotor. We experimentally demonstrate that our method allows initiating evasive maneuvers early enough to avoid collisions.**

I. Introduction

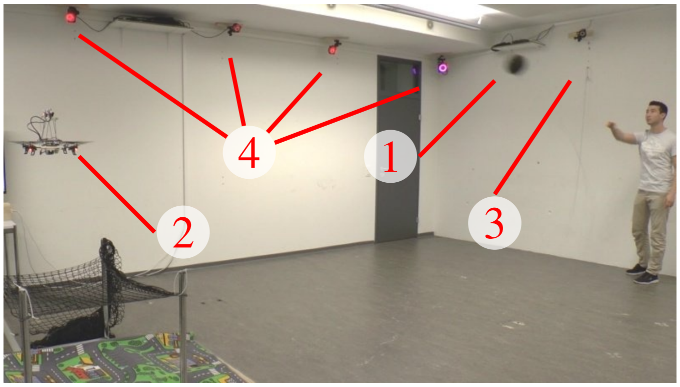
Collision avoidance of fast-moving objects requires high-frequency and low-latency sensors, algorithms, and control strategies. As an example, consider an object that is thrown at a robot at 30 m/s (i.e., 108 km/h) from a distance of 5 m, only 0.17 s are left to (i) detect the object, (ii) predict its trajectory, (iii) foresee if a collision will occur, and if so, (iv) initiate and (v) execute an evasive maneuver. Due to the inertia of the robot and its actuators, most of this time is required for the evasive action and only a small fraction (in the order of 10 ms) can be used for sensing and computation. During this time, a high-frequency camera running at 100 Hz would capture only one or two images, giving very limited data for the subtasks (ii) and (iii).

To achieve higher measurement frequencies while keeping the computational load small, new vision sensors are required. In this paper, we propose the use of Dynamic Vision Sensors (DVS) [2]. Contrarily to standard frame-based cameras that send entire images at fixed frame rates, a DVS only sends the local pixel-level brightness *changes* at the time they occur. These changes, which we call "events", are transmitted asynchronously and with low latency. While the sensor's spatial resolution of 128 × 128 pixels is still low, the temporal resolution is in the order of *micro*-seconds.

In the last few years, impressive demonstrations of aggressive flight and acrobatics with quadrotors have been presented [3], [4]. Among these demonstrations were also interactions with other, fast-moving objects, e.g. juggling balls [5], pole acrobatics [6], or flying through thrown circular hoops [7]. However, all these demonstrations are based on external motion-capture systems that track both the quadrotor and all other moving objects with very high precision and frequency (typically about 200 Hz). To bring these capabilities outside of laboratory environments, we cannot rely on external systems. Therefore, all sensing and computation must be performed *onboard* the vehicle. Due to their low weight and power consumption, vision-based approaches have been successfully demonstrated for autonomous, infrastructure-free flight with quadrotors [1]. However, due to motion blur at high speeds and



(a) Quadrotor platform: (1) stereo DVS rig, (2) smartphone computer, (3) down-looking camera for vision-based stabilization, and (4) markers for ground truth with a motion-capture system. The details of our quadrotor platform are provided in [1].



(b) Experimental setup: (1) thrown ball, (2) quadrotor, (3) leash to avoid actual collisions, and (4) motion-capture system for ground-truth measurements.

Fig. 1. A ball is thrown towards an autonomous, vision-based quadrotor. The ball is detected and tracked using a pair of Dynamic Vision Sensors in a stereo configuration. Our algorithm predicts whether a collision will occur and can be used to initiate evasive maneuvers. The motion-capture system was only used to record ground-truth data of the ball and the quadrotor.

computational complexity for high frame rates, current vision-based quadrotor systems fly at relatively low speeds and only navigate in static environments.

In this paper, we present a method to predict collisions with objects thrown at a quadrotor using a pair of DVS in a stereo configuration (see Fig. 1). To avoid a collision, a series of steps must be executed: (i) the thrown object must be detected, (ii) it must be tracked precisely to (iii) propagate its trajectory in time. Then, (iv) a decision on the action must be made to (v) initiate and execute an evasive maneuver. Possible applications are quick avoidance of other, uncooperative aerial vehicles and the escape of bird attacks[1].

The remainder of the paper is organized as follows. In Section II, we review related work. The DVS is described in Section III, followed by an evaluation of the latencies of both standard frame-based cameras and the DVS in Section IV. Our algorithm is described in Section V and experimentally evaluated in Section VI.

II. Related Work

An impressive demonstration of the low-latency capabilities of a DVS for control applications was presented in [8]. Using two DVS, the authors implemented a pencil-balancing system on a highly-reactive platform free to move on a plane. A robotic goalkeeper with a reaction time of 3 ms was presented in [9].

An Event-based Iterative Closest Point Algorithm (ICP) was used in [10] for closed-loop control of a micro gripper. The mean update rate was 4 kHz. The algorithm integrates events over a predefined time interval and only works in 2D.

Asynchronous, event-based optical flow was presented in [11]. The authors adapted the Lucas-Kanade tracking algorithm to cope with the event-based nature of the DVS. The event-based optical flow was later used [12] for event-based computation of the time-to-contact [13]. This approach, however, assumes that the trajectories of the robot and the obstacles are aligned, i.e., when the robot continues to move, a collision is unavoidable. In this paper, we explicitly estimate the trajectory of the thrown object, since it might no intersect with the robot and no evasive action is required.

Several approaches of event-based stereo matching can be found in the literature. In [14], the high temporal resolution of the DVS was exploited for stereo matching. In [15], event histograms were used for stereo correspondence. The output of the algorithm was used for gesture recognition. In [16], six synchronized DVS were used for 3D reconstruction using N-ocular stereo vision.

In our previous work [17], a DVS fixed to the ground was used to recover the pose of a quadrotor during flight by tracking LEDs mounted on the platform, which were blinking at very high frequencies. The DVS' time resolution allowed distinguishing different frequencies, thus avoiding the need for data association. While this system successfully showed low-latency pose-tracking capabilities using a DVS, it required active markers (i.e., the blinking LEDs). Furthermore, the DVS was not mounted onboard the quadrotor. We use a similar concept for intrinsic and extrinsic camera calibration.

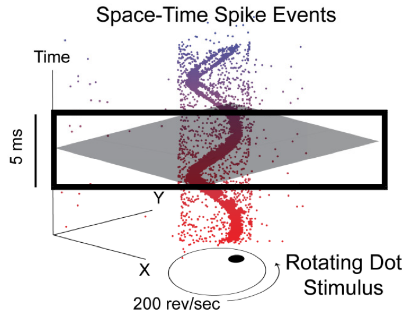


Fig. 2. Visualization of the output of a DVS looking at a rotating dot. Colored dots mark individual events. The polarity of the events is not shown. Events that are not part of the spiral are caused by sensor noise. Figure adapted from [21].

Localization using a DVS on a ground robot was first presented in [18] and later extended to Simultaneous Localization And Mapping (SLAM) in [19]. However, the system was limited to planar motion and a 2D map. In their experiments, the authors used an upward-looking DVS mounted on a ground robot moving at low speed.

In previous work, we showed how a DVS can be used onboard a flying robot for localization during high-speed maneuvers [20], where rotational speeds of up to 1,200 °/s were measured during quadrotor flips.

III. Dynamic Vision Sensors

A. Working Principle

Standard CMOS cameras send full frames at fixed frame rates. On the other hand, event-based (retinal) cameras such as the DVS [2] have independent pixels that generate events at local relative brightness changes in continuous time. These events are timestamped and transmitted asynchronously at the time they occur using sophisticated digital circuitry. Each event e is a tuple $\langle \mathbf{p}, t, p\rangle$, where $\mathbf{p} = (x, y)$ are the pixel coordinates of the event, t is the timestamp of the event, and $p \in \{-1, +1\}$ is the polarity of the event, which is the sign of the brightness change. This representation is sometimes also referred to as Address-Events Representation (AER). The DVS has a resolution of 128×128 pixels and is connected via USB. A visualization of the output of the DVS is shown in Fig. 2.

Due to its low latency and high temporal resolution, both in the range of *micro*-seconds, the DVS is a very promising sensor for high-speed mobile robot applications. Since the data stream from the DVS is sparse (only *changes* are reported), the bandwidth and computational load are low. An additional advantage for robotic applications is the DVS' high dynamic range of 120 dB (compared to 60 dB of expensive computer-vision cameras), which allows both indoor and outdoor operation without changing parameters. Since all pixels are independent, these contrasts can also take place within the same scene.

B. Calibration

We used a board with blinking LEDs for intrinsic and extrinsic calibration of the DVS stereo setup (see Fig. 3).

[1] See, e.g., http://youtu.be/DzfiLmbhvqg or http://youtu.be/smv7cBzg-Ok.

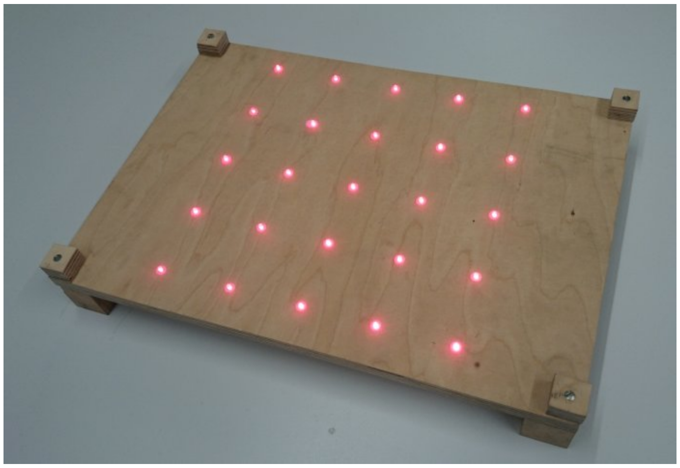


Fig. 3. Board with blinking LEDs for intrinsic and extrinsic calibration of the DVS stereo setup. The LEDs are blinking at a frequencies of 1 kHz, such that they can easily be detected by a DVS.

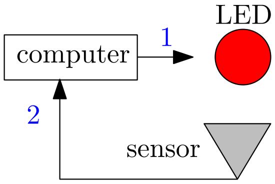


Fig. 4. Picture of sensor-latency measurement setup: the LED (1) is triggered by a computer and observed by a sensor (2). In our case, the sensor is either a DVS, a BlueFOX, or an ASUS Xtion. We measure the round-trip delay from sending the signal until the change was detected by the sensor. The experiments were performed for each sensor individually.

While we used computer screens in previous work [20] for calibration, we found that an LED checkerboard allows for larger viewing angles and, thus, better calibration results. Since a DVS only responds to changes in the scene, blinking LED allow us to artificially trigger events without moving the sensor. Due to its very high temporal resolution, a DVS can easily cope with the LED blinking frequency of 1 kHz. As shown in [17], this frequency is well above those generated by moving the sensor or moving objects in the scene. Since the optics are the same for the DVS as for frame-based cameras, we can rely on standard algorithms for intrinsic and extrinsic calibration. We released our ROS-compatible[2] DVS driver and calibration suite as open-source software[3].

IV. Sensor Latencies

To motivate the use of DVS for low-latency and high-speed robotic applications, we compare its latency to frame-based cameras. To do so, we measure the round-trip delay between toggling an LED and the detection of this change by the different sensors (see Fig. 4). This measurement includes sending the command to toggle the LED, the time to capture an image (in the frame-based case), data transfer to the computer, and simple computations to detect the change.

A. Experimental Setup

We compare the DVS with two frame-based cameras: the ASUS Xtion Pro Live and MatrixVision mvBlueFOX-MLC200w. The ASUS Xtion has a rolling shutter, a resolution of 640×480 pixels, and provides RGB-D images at 30 Hz. Exposure and gain are controlled by the sensor automatically. The BlueFOX camera has a global-shutter, a resolution of 752×480 pixels, and provides grayscale images up to 90 Hz. The exposure time was set to 4 ms and the gain to 0 dB. The bias-generation setting for the DVS were set to “fast”[4]. We interfaced all sensors over USB and evaluated the delays both on a laptop computer (Lenovo W530) and an embedded computer (Hardkernel Odroid U3).

Back-of-the-envelope calculations for the two frame-based cameras show the minimally achievable latencies for this setup: USB 2.0 is specified for 480 Mbit/s. An RGB image from the ASUS Xtion has a raw size of $3 \times 640 \times 480 \times 8$ bits $= 7.3$ Mbit, yielding a transfer duration of 15.2 ms. A grayscale image from the BlueFOX has a raw size of $752 \times 480 \times 8$ bits $= 2.1$ Mbit, yielding a transfer duration of 6.0 ms. This is only a lower bound on the transfer duration: the exposure time must also be added. An event of a DVS is encoded in 32 bit, yielding a transfer duration of 0.067 µs. Therefore, the event rate is bounded by 15 million events per second.

An LED is triggered using a PX4FMU-Autopilot board[5], which is the same board that interfaces the motor controllers on many quadrotor platforms. It is optimized for reliable and low-latency communication.

To detect the LED on the frame-based cameras, we defined a small region of interest in which the LED is visible. We then computed the mean intensity in that region and reported a detection when the mean changed by more than a threshold.

To detect the LED using the DVS, we measured the number of events per time unit. When nothing changes, only “background-activity” events are transmitted, which are caused by sensor imperfections. However, as soon as the LED is toggled, many events are generated.

For each combination of computer, toggle direction, and sensor, we collected more than 1,000 measurements. To avoid aliasing effects, we waited for a random amount of time after each detection before toggling the LED again. No triggering signal was missed in all the experiments.

B. Results

The sensor latency (i.e., capturing the image, transferring it to the host computer, and performing a simple computation to detect a change) is assumed to be Gaussian. It is modeled with a random variable $X_1 = \mathcal{N}(\mu, \sigma)$, where μ is the mean delay and σ is the standard deviation. Due to the fixed frequency of standard cameras (typically $f = 30$ Hz), a uniformly-distributed delay between 0 and $\Delta t = 1/f$ is added to the sensor latency: $X_2 = \mathcal{U}(0, \Delta t)$, where Δt is the time between two frames. Therefore, the expected delay Y is thus a sum of two independent random variables, $Y = X_1 + X_2$. We identify

[2] Robot Operating System, http://www.ros.org

[3] http://www.github.com/uzh-rpg/rpg_dvs_ros

[4] http://sourceforge.net/p/jaer/code/HEAD/tree/jAER/trunk/biasgenSettings/DVS128/DVS128Fast.xml

[5] http://www.pixhawk.org/modules/px4fmu

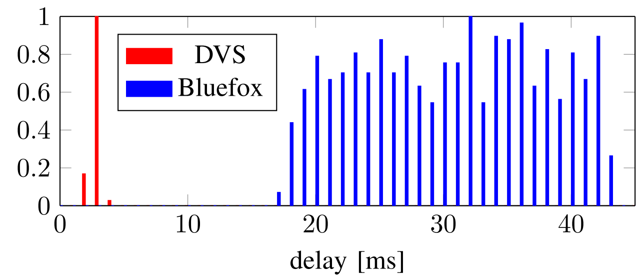


Fig. 5. Round-trip delays for a DVS (red) and a BlueFOX camera (blue) using a laptop computer. The delays for the BlueFOX camera are wide spread due to the synchronous nature of the sensor. In our experiments, it ran at 40 Hz, which corresponds to 25 ms. The histograms were normalized with their maximum value.

TABLE I. ROUND-TRIP DELAYS USING A LAPTOP COMPUTER

Sensor	μ_{on}	σ_{on}	μ_{off}	σ_{off}	Δt	Unit
ASUS Xtion	28.0	2.7	50.3	2.8	33	ms
BlueFOX	14.5	0.5	18.0	0.5	25	ms
DVS	2.8	0.3	2.8	0.3	0	ms

TABLE II. ROUND-TRIP DELAYS USING AN EMBEDDED COMPUTER

Sensor	μ_{on}	σ_{on}	μ_{off}	σ_{off}	Δt	Unit
ASUS Xtion	45.0	4.0	67.3	2.8	33	ms
BlueFOX	19.9	0.4	20.5	1.2	25	ms
DVS	4.5	1.2	4.2	0.8	0	ms

the mean μ and standard deviation σ of X_1 in terms of those of Y and X_2, and replacing the mean and variance of Y by their empirical values (sample mean and sample variance), as detailed in the Appendix. For the DVS, only a Gaussian was fitted because it works asynchronously.

A histogram comparing the latency distributions of the DVS and the BlueFOX camera is shown in Fig. 5. While the delays of the DVS are an order of magnitude lower, it can also be seen that the delays of the BlueFOX camera are wide spread due to the synchronous nature of the sensor. In our experiments, it was running at a framerate of 40 Hz, which corresponds to 25 ms.

In Table I, we summarize the round-trip delays for the three vision sensors when connected to the laptop computer. We report the identified parameters of the histogram distribution, where the indices *on* and *off* indicate measurements when turning on and off the LED, respectively. Table II provide the same results measured on the embedded computer. Our results are in line with the results reported in [9] of (2.2 ± 2.0) ms.

V. Algorithm

In this section, we describe an algorithm to track spherical objects thrown at flying quadrotor using two DVS in a stereo configuration (see Fig. 1(a)). We first describe an event-based circle tracker that is similar to the trackers introduced in [22]. Then, we match trackers from the left and right sensor that fulfill a set of geometric constraints. We exploit the high temporal resolution of the DVS to measure the disparity with sub-pixel accuracy using the correlation of spatio-temporal neighborhoods of matching trackers. These measurements are used to initialize and update an Extended Kalman Filter (EKF). A mixed state space allows incorporation of both the object's dynamics and the measurement noise on the image plane. We then propagate the current state of the system with its uncertainty in time and check for a collision of the 3σ-ellipsoid around the predicted trajectory with a safety sphere around the quadrotor. In the following, we detail these steps.

A. Event-based Circle Tracker

Our trackers are similar to the ones described in [22], but specialized to track spherical objects. We describe circular trackers by their mean position $\boldsymbol{\mu}_p$, radius $\boldsymbol{\mu}_r$, and uncertainty in radius σ_r as illustrated in Fig. 6. For each incoming event, we evaluate its score $p_i(\mathbf{p})$ that belongs to tracker i,

$$p_i(\mathbf{p}) = \frac{1}{\sqrt{2\pi}\sigma_{r,i}} \exp\left(-\frac{1}{2}\left(\frac{d_i - \mu_{r,i}}{\sigma_{r,i}}\right)^2\right), \quad (1)$$

where $d_i = \|\mathbf{p} - \boldsymbol{\mu}_{p,i}\|$ is the distance between the event's position $\mathbf{p}$ and the tracker's position $\boldsymbol{\mu}_{p,i}$. The tracker $i = i_{max}$ with the highest score is then updated using an Infinite Impulse Response (IIR) filter,

$$\begin{aligned}
\boldsymbol{\mu}_p(t) &= \boldsymbol{\mu}_p(t_{\text{prev}}) + \alpha_p \left(\mathbf{p} - \boldsymbol{\mu}_p(t_{\text{prev}})\right), \\
\mu_r(t) &= \mu_r(t_{\text{prev}}) + \alpha_r \left(d - \mu_r(t_{\text{prev}})\right), \\
\sigma_r^2(t) &= \sigma_r^2(t_{\text{prev}}) + \alpha_\sigma \left(\left(d - \mu_r(t)\right)^2 - \sigma_r^2(t_{\text{prev}})\right),
\end{aligned} \quad (2)$$

which corresponds to an exponentially-weighted moving average filter with *smoothing factors* $\{\alpha_{\text{p}}, \alpha_r, \alpha_\sigma\} \in (0, 1)$. Small values indicate more smoothing but also induce more latency. In our experiments, we empirically set $\alpha_{\text{p}} = 0.01$, $\alpha_r = 0.002$, and $\alpha_\sigma = 0.005$.

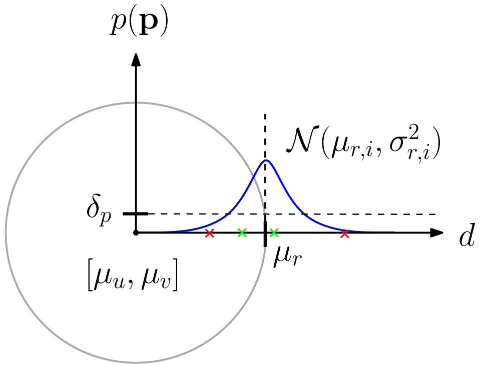


Fig. 6. Illustration of the Gaussian radius distribution $\mathcal{N}(\mu_r(t), \sigma_r^2(t))$ of a tracker. An event with a score p_i below δ_p, e.g., red crosses, is not considered, while an event above δ_p, e.g., green crosses, is considered to be generated by the circular shape.

The activity $\mathcal{A}_i$ of a tracker indicates when the tracker was last updated by events,

$$\mathcal{A}_i(t) = \begin{cases} \mathcal{A}_i(t - \Delta t)\exp\left(-\frac{\Delta t}{\tau}\right) + p_i(\mathbf{p}), & i = i_{max}, \\ \mathcal{A}_i(t - \Delta t)\exp\left(-\frac{\Delta t}{\tau}\right), & \text{otherwise.} \end{cases} \quad (3)$$

In addition, each tracker has k directional activities a_j with $j = 1, \ldots, k$, which is defined as (3) but only for a $1/k$-th section of the circle. From these directional activities, we compute a factor γ,

$$\gamma = \frac{1}{\mathcal{A}_i} \sum_{j=1}^{k} \frac{a_j}{k}, \tag{4}$$

which is high if the events are distributed uniformly around the circle and low otherwise. Therefore, γ indicates how well the tracker follows an actual circle. In the following, we only consider trackers that have $\mathcal{A} > 15$ and $\gamma > 0.7$, i.e., trackers that have received sufficient support from the event stream and follow a circle. To avoid that the trackers follow arbitrary shapes, we restrict the radius, its variance, and the variance-to-mean ratio to reasonable intervals. For the initialization of the trackers and more details of the algorithm, we refer the reader to [22].

B. Stereo Matching

We match active trackers from the left and right sensor plane that fulfill all of these four constraints: (i) the disparity d must be positive, (ii) the mean vertical positions μ_x of the trackers must be close, (iii) the radius r of the trackers must be similar, and (iv) each tracker can only have one matching tracker.

Due to the low spatial resolution of the DVS of only 128×128 pixels and the short baseline of 12 cm, the disparity at 5 m is only 3.1 pixels. Thus, noise in the tracker positions has a significant impact on the accuracy of the depth measurement. We therefore refine the disparity estimate with a sub-pixel estimation algorithm that we initialize with the stereo-matching estimate.

C. Sub-Pixel Disparity Estimation

To increase the precision of the stereo matching, we compute the correlation of the spatio-temporal neighborhood of the two matching trackers. More precisely, we correlate the Surface of Active Events (SAE) [23] using linear interpolation. The SAE $\Sigma_p(\mathbf{p})$ stores the last timestamp of an event with polarity p that was reported at pixel location $\mathbf{p}$,

$$\Sigma_p(\mathbf{p}) \leftarrow t. \tag{5}$$

We search for the maximum correlation of the left and right SAE,

$$d = \arg\max_{d} \sum_{p} \sum_{u} \sum_{v} \tilde{\Sigma}^l_p(u, v) \cdot \tilde{\Sigma}^r_p(u + d, v), \tag{6}$$

where l and r refer to the left and right SAE, respectively, and $\tilde{\Sigma}$ is the shifted SAE defined as

$$\tilde{\Sigma}(\mathbf{p}) = \max(\Sigma(\mathbf{p}) - t_{\text{curr}} + \Delta T, 0), \tag{7}$$

where t_{curr} is the current time and $\Delta T = 50\,\text{ms}$. All timestamps on the shifted SAE are between 0 and ΔT. We first evaluate (6) in 1-pixel steps and then refine it with 0.1-pixel steps. This results in a measurement of the ball's center of mass in the image plane, i.e., its location u, v and its disparity d with sub-pixel accuracy.

D. Extended Kalman Filter

Under the assumption of negligible air drag and a perfectly-stable hovering quadrotor (i.e., pitch, roll, and all angular rates are zero), the ballistic trajectory of a ball in the sensor's 3D coordinate frame is given by

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} v_{X0} \\ v_{Y0} \\ v_{Z0} \end{bmatrix} t + \frac{1}{2} \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} t^2, \tag{8}$$

with initial position $[X_0, Y_0, Z_0]^\top$, initial velocity $[v_{X0}, v_{Y0}, v_{Z0}]^\top$, and gravitational acceleration g.

For the Extended Kalman Filter (EKF), we use a mixed state $\mathbf{x}(t)$ consisting of the ball's position in image coordinates and its velocity in world coordinates,

$$\begin{aligned} \mathbf{x}(t) &= [x_1, x_2, x_3, x_4, x_5, x_6]^\top \\ &= [d, \dot{Z}, x_{\text{com}}, \dot{X}, y_{\text{com}}, \dot{Y}]^\top. \end{aligned} \tag{9}$$

This allows us to incorporate the measurement noise in pixel units with the dynamics of the ball (8). The coordinates in image space and world coordinates are linked by the pinhole camera model, i.e.,

$$X = b \cdot \frac{u}{d}, \;\; Y = b \cdot \frac{v}{d}, \;\; Z = b \cdot \frac{f}{d}, \tag{10}$$

where f is the focal length of the camera and b the baseline of the stereo setup.

Using (10) and (8), the nonlinear continuous-time system can be derived as

$$\dot{\mathbf{x}}(t) = \mathbf{q}\left(\mathbf{x}(t)\right) = \begin{bmatrix} \frac{-1}{fb} \cdot x_1^2 \cdot x_2 \\ 0 \\ \left(\frac{1}{b} \cdot x_4 - \frac{1}{fb} \cdot x_2 \cdot x_3\right) \cdot x_1 \\ 0 \\ \left(\frac{1}{b} x_6 - \frac{1}{fb} \cdot x_2 \cdot x_5\right) \cdot x_1 \\ g \end{bmatrix} \tag{11}$$

with observation vector

$$\mathbf{z}(t) = \mathbf{h}\left(\mathbf{x}(t)\right) = [x_1, x_3, x_5]^\top. \tag{12}$$

Using a first-order approximation of the derivate and a timestep of T_k, the discrete-time nonlinear system becomes

$$\mathbf{x}(t_{k+1}) = \mathbf{q}\left(\mathbf{x}(t_k)\right) = \begin{bmatrix} \left(\frac{-1}{fb} \cdot x_1^2 \cdot x_2\right) \cdot T_k + x_1 \\ x_2 \\ \left(\frac{x_4}{b} - \frac{1}{fb} x_2 \cdot x_3\right) \cdot x_1 \cdot T_k + x_3 \\ x_4 \\ \left(\frac{1}{b} x_6 - \frac{1}{fb} \cdot x_2 \cdot x_5\right) \cdot x_1 \cdot T_k + x_5 \\ g \cdot T_k + x_6 \end{bmatrix}.$$

The linearized system matrix $\mathbf{A}(t_k)$ is given by the Jacobian of the nonlinear system around the current state x_i, which has a closed-form solution,

$$\mathbf{A}(t_k) = \frac{\partial \mathbf{q}\left(\mathbf{x}(t_k)\right)}{\partial \mathbf{x}(t_k)}.$$

We initialize the filter state with a linear least-square regression on the first four measurements. While the filter could be initialized with only two measurements, we found that four yields a good tradeoff between latency and precision.

E. Trajectory Propagation

To propagate the trajectory, we transform the ball's position in image coordinates and its covariance to world coordinates. For the covariance transformation, we use a first-order approximation. We then use the dynamical model (8) to propagate the system in world coordinates. We define the world-coordinate origin to coincide with the quadrotor center. We compute a critical time t_{crit} at which the distance between the trajectory $\mathbf{r}(t)$ and the quadrotor is minimal and propagate the states up to that time. We find t_{crit} by

$$\frac{\partial}{\partial t}|\mathbf{r}(t_{\text{crit}})| \overset{!}{=} 0. \tag{13}$$

The more intuitive geometrical solution of this problem is illustrated in Figure 7, which yields

$$\mathbf{r}(t_{\text{crit}}) \cdot \dot{\mathbf{r}}(t_{\text{crit}}) \overset{!}{=} 0, \tag{14}$$

which results in a cubic equation in t_{crit} that can be solved by, e.g., Cardano's method.

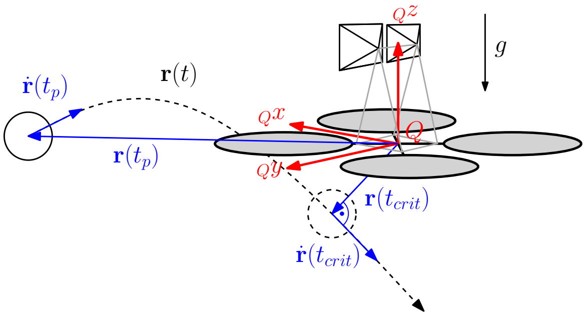


Fig. 7. Illustration of the geometrical solution to obtain the critical point $\mathbf{r}(t_{\text{crit}})$ on the ballistic trajectory $\mathbf{r}(t)$ defined by the vectors $\mathbf{r}(t_p)$ and $\dot{\mathbf{r}}(t_p)$. The critical point occurs at the time t_{crit}, when the position vector and its derivative are orthogonal.

F. Maneuver Decision

We define a spherical *safety zone* with radius r_{safe} around the quadrotor's origin. Since the trajectory prediction includes the expected value of the critical position and its uncertainty, a collision is predicted if the uncertainty ellipsoid around the critical position intersects with the safety zone.

If the distance-to-origin $\|\mathbf{r}(t_{\text{crit}})\|$ of the predicted critical position is smaller than r_{safe}, we expect a collision in any case. If it is larger, we need to consider the uncertainty of the prediction. The vector $\mathbf{r}_{\text{dist}}(t_{\text{crit}})$ (15) represents the distance from the expected critical position on the predicted trajectory to the closest point on the surface of the spherical safety zone,

$$\mathbf{r}_{\text{dist}}(t_{\text{crit}}) = \left(\frac{r_{\text{safe}}}{\|\mathbf{r}(t_{\text{crit}})\|} - 1\right) \cdot \mathbf{r}(t_{\text{crit}}). \tag{15}$$

The covariance matrix $\mathbf{\Sigma}_r(t_{\text{crit}})$ can be interpreted as an ellipsoid that gives us information about directional uncertainty of the predicted critical position. This information is extracted with the Principal Component Analysis (PCA) of the covariance matrix

$$\mathbf{\Sigma}_r(t_{\text{crit}}) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{16}$$

with

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] \quad \text{and} \quad \mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3),$$

where the normalized eigenvectors $\mathbf{u}_i$ are the principal axes of the uncertainty ellipsoid and the square roots of the corresponding eigenvalues λ_i represent the standard deviations along the axes.

Therefore, we rotate $\mathbf{r}_{\text{dist}}$ to the PCA space U using $\mathbf{U}$ and scale it with $3\sqrt{\mathbf{\Lambda}}$, which corresponds to a probability of $99.73\,\%$. We call this the *confidence vector*

$$_U\mathbf{r}_{\text{conf}}(t_{\text{crit}}) = \left(3\sqrt{\mathbf{\Lambda}}\right)^{-1} \cdot \mathbf{U} \cdot \mathbf{r}_{\text{dist}}(t_{\text{crit}}). \tag{17}$$

The norm of the confidence vector then gives us an indication of the predicted critical position not entering the safety zone with a probability of $99.73\,\%$. More precisely, we can rule out a collision with said probability, if the norm of the confidence vector is larger than 1.

Additionally, we set a threshold σ_λ for the largest eigenvalue that determines whether we trust the prediction. I.e., we predict a collision if

$$\max(\mathbf{\Lambda}) \leq \sigma_\lambda \text{ and } \|\mathbf{r}(t_{\text{crit}})\| \leq r_{\text{safe}} \tag{18}$$

or

$$\max(\mathbf{\Lambda}) \leq \sigma_\lambda \text{ and } \|_U\mathbf{r}_{\text{conf}}(t_{\text{crit}})\| \leq 1. \tag{19}$$

VI. Experiments

We first describe the experimental setup. Then, we evaluate the tracking performance and the effect of the EKF. We compare the measurements with a ground truth captured with a motion-capture system. Finally, we analyze the time margin between a collision is predicted and the time of collision.

A. Experimental Setup

We mounted two DVS in a stereo setup on a quadrotor (see Fig. 1(a)). Our quadrotor platform is described in detail in [1]. All sensing and computation for flying was performed onboard. We recorded the event streams from both DVS while hovering in vision-based flight that we later processed offboard. Then, we threw a ball towards the quadrotor from a distance of 6 m at speeds of about 10 m/s. The ball was secured with a leash that prevents a collision shortly before it would occur. We recorded ground truth for both the quadrotor and the ball using an OptiTrack motion-capture system.

B. Circle Tracking

Fig. 8 shows a snapshot of the tracking. The three trackers on the left event stream in Fig. 8(a) are filtered by the stereo-matching constraints. Since the trackers only roughly track the ball's position, a sub-pixel disparity refinement is used to improve the measurement.

C. EKF Performance

Figure 9 shows the stereo measurements and the output of the EKF. The first four measurements between t_1 and t_2 are used for initialization of the filter. The last measurements at time t_3 are corrupted since the ball is no longer fully visible by the DVS. However, since these measurements do not pass a 3σ validation gate, they are not used to update the state and only the dynamical model is propagated.

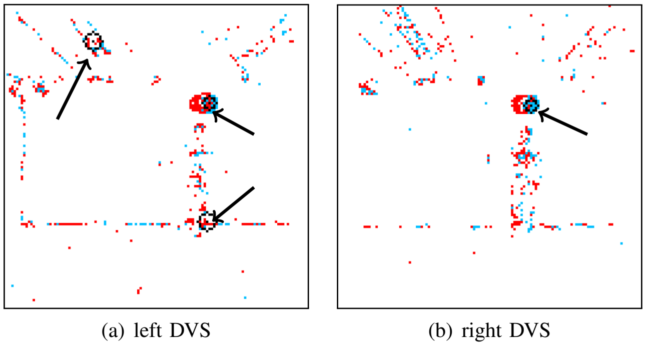


Fig. 8. Tracking of the ball during a throw. Red and blue points indicate events with positive and negative polarity, respectively. The active circle trackers are marked in black and highlighted with an arrow.

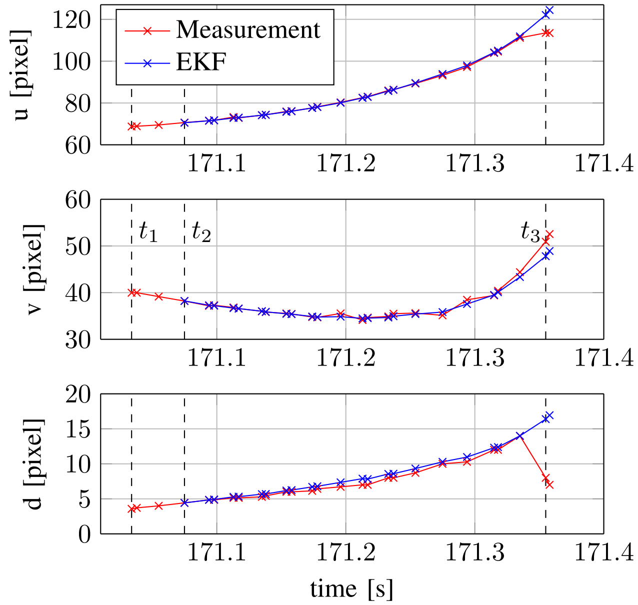


Fig. 9. Comparison of the measurement and the EKF estimate of the center of mass in image coordinates $[u, y, d]^\top$. The first measurement is obtained at t_1 and the EKF is initialized at t_2. The EKF recognizes corrupt measurement at t_3.

D. Comparison with Ground Truth

In Fig. 10, the output of the EKF is compared to ground truth obtained from a motion-capture system. It also shows the triggering signal for the evasive maneuver. The EKF is initialized at time t_2 and at time t_4, a triggering signal is sent. The elapsed time is $t_4 - t_2 = 15\,\mathrm{ms}$. Since the ball was on a leash in the experiment, it stops before hitting the quadrotor. When we extrapolate its trajectory, a collision with the quadrotor would occur at time t_5. In this case, this yields a time of $t_5 - t_4 = 330\,\mathrm{ms}$ for the quadrotor to escape before a collision would occur. Since the plots in Fig. 10 are time-stamped by the same clock, they include the latency of the algorithm.

E. Time Margin for Evasive Maneuver

The time margin is the time between a collision is predicted and the time of collision (assuming no maneuver) We evaluate the time margin for the evasive maneuver for a series of 19 experiments. In 15 experiments, the ball was thrown at the quadrotor, which was correctly predicted 12 times (80 %), while it was not detected in 3 experiments. In the other 4 experiments, the ball missed the quadrotor. Our algorithm reported 3 times (75 %) that no collision will occur and did not detect the ball in one experiment.

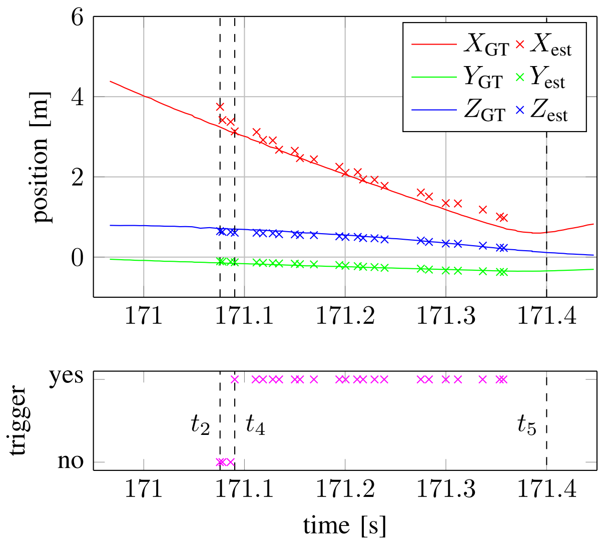


Fig. 10. Ground truth and center of mass estimate in quadrotor frame world coordinates. The EKF is initialized at t_2 and the collision prediction is made at t_4. The expected collision would take place at t_5. The evasive maneuver triggering signals are shown in magenta.

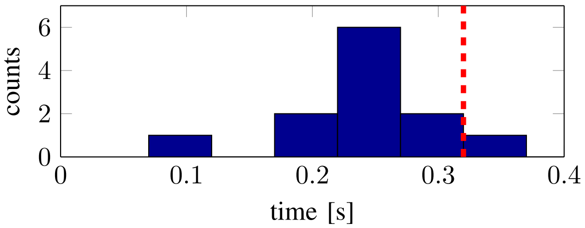


Fig. 11. Histogram of the escape time. The red dashed line indicates the $0.32\,\mathrm{s}$ escape time that is necessary for our quadrotor to escape the imminent collision with a free fall evasive maneuver (see Section VII).

We summarized the time margin for the 12 successfully predicted collisions in Fig. 11. We estimated the time of collision by extrapolating the ball trajectory until it would hit the quadrotor. The algorithm ran in real-time on the recorded data, thus we incorporate also the processing time in our analysis. In most experiments, the time margin was 250 ms.

VII. Conclusion

In this paper, we demonstrated a method to predict collisions of objects thrown at a quadrotor. We used two DVS in a stereo configuration. The ball was detected and tracked in both event streams. Using a set of geometric constraints, a rough stereo matching is obtained that is further refined by sub-pixel disparity estimation. These measurements are fused in an EKF that uses a mixed state to incorporate both the measurement noise in the image plane and the dynamical model of the ball's trajectory. We predict a collision if the 3σ-ellipsoid along the predicted trajectory intersects with a safety sphere around the quadrotor.

While the prediction of a collision is available within a reasonable time margin of $250\,\mathrm{ms}$ in most cases, our quadrotor platform currently does not allow to execute an aggressive evasive maneuver. In fact, due to the additional payload of two DVS and its sensor mount (which makes the quadrotor $40\,\%$ heavier), the quadrotor operates at its actuator limits (i.e., it can barely hover). The only possible evasive maneuver would be a free fall. However, to escape a safety sphere of $s = 0.5\,\mathrm{m}$ using free fall, a time margin of at least $\sqrt{2s/g} = 0.32\,\mathrm{s}$ is required, assuming no delays on the motor controllers and neglecting rotor dynamics. We are currently building a more powerful quadrotor platform to perform different evasive maneuvers in future work.

Acknowledgments

The authors wish to thank Matthias Faessler and Guillermo Gallego for their contributions to the sensor-latency measurements. This research was supported by the Swiss National Science Foundation through project number 200021-143607 (Swarm of Flying Cameras), the National Centre of Competence in Research (NCCR) Robotics, and Google.

Appendix

A. Probabilistic Sensor-Latency Model

The total sensor latency Y is the sum of two independent random variables,

$$\begin{aligned} X_1 &= \mathcal{N}(\mu, \sigma), && (20)\\ X_2 &= \mathcal{U}(0, \Delta t), && (21)\\ Y &= X_1 + X_2, && (22) \end{aligned}$$

where X_1 models the Gaussian sensor latency and X_2 accounts for the uniform delay due to the fixed sensor frequency (cf. Sec. IV-B). While Δt is known, we want to identify μ and σ. Since the two variables are independent, we can sum their expected values and variances,

$$\begin{aligned} \mathrm{E}[Y] = \quad & \mathrm{E}[X_1] + \mathrm{E}[X_2] & = \mu + \frac{\Delta t}{2}, \quad (23)\\ \mathrm{Var}[Y] = \quad & \mathrm{Var}[X_1] + \mathrm{Var}[X_2] & = \sigma^2 + \frac{\Delta t^2}{12}. \quad (24) \end{aligned}$$

Given enough samples, we can compute $\mathrm{E}[Y]$ and $\mathrm{Var}[Y]$ from the measurements and compute the mean μ and variance σ^2 of the Gaussian as

$$\begin{aligned} \mu &= \mathrm{E}[Y] - \frac{\Delta t}{2}, && (25)\\ \sigma^2 &= \mathrm{Var}[Y] - \frac{\Delta t^2}{12}. && (26) \end{aligned}$$

References

[1] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV,” *J. of Field Robotics*, 2015.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor,” *IEEE J. of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[3] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *Intl. J. of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[4] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D’Andrea, “A platform for aerial robotics research and demonstration: The Flying Machine Arena,” *J. of Mechatronics*, vol. 24, no. 1, pp. 41–54, Feb. 2014.

[5] M. Muller, S. Lupashin, and R. D’Andrea, “Quadrocopter ball juggling,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[6] D. Brescianini, M. Hehn, and R. D’Andrea, “Quadrocopter pole acrobatics,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Nov. 2013.

[7] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[8] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck, “A pencil balancing robot using a pair of AER dynamic vision sensors,” in *Intl. Conf. on Circuits and Systems (ISCAS)*, 2009.

[9] T. Delbruck and M. Lang, “Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor,” *Frontiers in Neuroscience*, vol. 7, no. 223, 2013.

[10] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, “Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics,” *IEEE Trans. Robotics*, vol. 28, pp. 1081–1089, 2012.

[11] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, “Event-based visual flow,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 407–417, 2014.

[12] X. Clady, C. Clercq, S.-H. Ieng, F. Houseini, M. Randazzo, L. Natale, C. Bartolozzi, and R. Benosman, “Asynchronous visual event-based time-to-contact,” *Frontiers in Neuroscience*, vol. 8, no. 9, 2014.

[13] D. N. Lee, “A theory of visual control of braking based on information about time-to-collision.” *Perception*, no. 5, pp. 437–59, 1976.

[14] R. Benosman, S.-H. Ieng, P. Rogister, and C. Posch, “Asynchronous event-based Hebbian epipolar geometry,” *IEEE Trans. Neural Netw.*, vol. 22, no. 11, pp. 1723–1734, 2011.

[15] J. Lee, T. Delbruck, P. Park, M. Pfeiffer, C. Shin, H. Ryu, and B. C. Kang, “Gesture-based remote control using stereo pair of dynamic vision sensors,” in *Intl. Conf. on Circuits and Systems (ISCAS)*, 2012.

[16] J. Carneiro, S.-H. Ieng, C. Posch, and R. Benosman, “Event-based 3D reconstruction from neuromorphic retinas,” *Neural Networks*, vol. 45, pp. 27–38, 2013.

[17] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza, “Low-latency localization by active LED markers tracking using a dynamic vision sensor,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[18] D. Weikersdorfer and J. Conradt, “Event-based particle filtering for robot self-localization,” in *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, 2012.

[19] D. Weikersdorfer, R. Hoffmann, and J. Conradt, “Simultaneous localization and mapping for event-based vision systems,” in *Intl. Conf. on Computer Vision Systems (ICVS)*, 2013.

[20] E. Mueggler, B. Huber, and D. Scaramuzza, “Event-based, 6-DOF pose tracking for high-speed maneuvers,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[21] S.-C. Liu and T. Delbruck, “Neuromorphic sensory systems,” *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[22] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, “Asynchronous event-based multikernel algorithm for high-speed visual features tracking,” *IEEE Trans. Neural Netw. Learn. Syst.*, 2014.

[23] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, “Asynchronous frameless event-based optical flow,” *Neural Networks*, vol. 27, pp. 32–37, 2012.