2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
September 14-18, 2014. Chicago, IL, USA,

Event-based, 6-DOF Pose Tracking for High-Speed Maneuvers

Elias Mueggler, Basil Huber and Davide Scaramuzza

***Abstract*— In the last few years, we have witnessed impressive demonstrations of aggressive flights and acrobatics using quadrotors. However, those robots are actually blind. They do not see by themselves, but through the "eyes" of an external motion capture system. Flight maneuvers using onboard sensors are still slow compared to those attainable with motion capture systems. At the current state, the agility of a robot is limited by the latency of its perception pipeline. To obtain more agile robots, we need to use faster sensors. In this paper, we present the first onboard perception system for 6-DOF localization during high-speed maneuvers using a Dynamic Vision Sensor (DVS). Unlike a standard CMOS camera, a DVS does not wastefully send full image frames at a fixed frame rate. Conversely, similar to the human eye, it only transmits pixel-level brightness changes at the time they occur with microsecond resolution, thus, offering the possibility to create a perception pipeline whose latency is negligible compared to the dynamics of the robot. We exploit these characteristics to estimate the pose of a quadrotor with respect to a known pattern during high-speed maneuvers, such as flips, with rotational speeds up to $1{,}200\,^\circ$/s. Additionally, we provide a versatile method to capture ground-truth data using a DVS.**

Supplementary Material

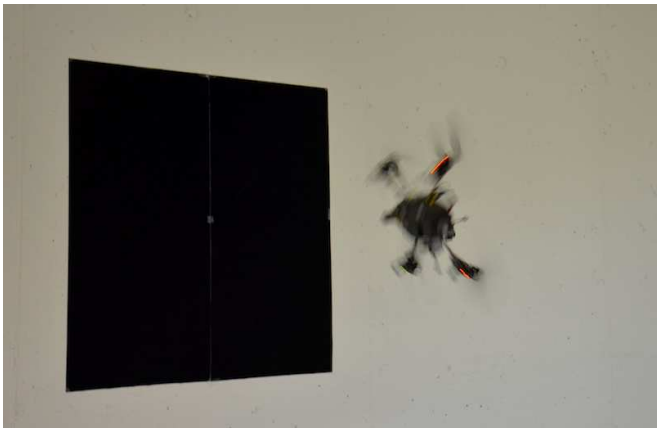
A video attachment to this work is available at:
`http://rpg.ifi.uzh.ch`.

I. Introduction

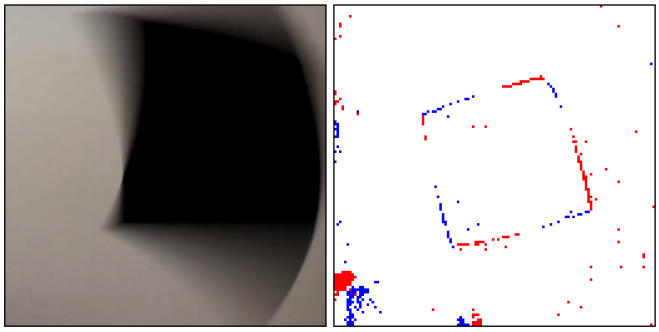
A. Motivation

In the last few years, impressive demonstrations of aggressive flight and acrobatics with quadrotors have been presented [1], [2]. Those systems are based on external motion-capture systems such as Vicon[1] or OptiTrack.[2] However, these setups are expensive, need active cameras, and are limited to small, confined workspaces. Thus, using onboard sensors is preferable for real-world applications. Many different sensor modalities have been proposed, such as laser scanners [3], [4], stereo cameras [5], and monocular cameras [6]. However, such systems achieve flight maneuvers that are still slow—especially in rotational speed—compared to those attainable with motion capture systems. Such high-speed performance is not achievable with commonly-used onboard sensors, such as CMOS cameras or laser range rangefinders.

The achievable agility of a robotic platform depends on the accuracy and latency of perception. The latency depends on the frequency of the sensor data, plus the time it takes to process the data. At the current state of the art, the latency of a CMOS-camera–based robot-perception pipeline is at the minimum in the order of 50–250 ms and the sampling rate in the order of 15–40 Hz. This puts a hard bound on the agility of the platform. We aim to overcome these limitations by exploiting a Dynamic Vision Sensor (DVS) [7]. Contrarily to standard frame-based CMOS cameras, which send entire images at fixed frame rates, a DVS only sends the local pixel-level changes caused by movement in a scene at the time they occur. The DVS output is a sequence of *asynchronous* events. Each pixel produces an event whenever it perceives a change of intensity. While the sensor's spatial resolution of 128×128 pixels is still low, the temporal resolution is in the order of *microseconds*. Thus, we can achieve low-latency



(a) Our quadrotor performing a flip.



(b) Standard CMOS camera (c) Integrated DVS events (2 ms)

Fig. 1: A quadrotor equipped with a standard CMOS camera and a DVS performing a flip. While the image of a standard CMOS camera suffers from high motion blur, a rendering of the DVS output shows that it can detect fast motion accurately. Blue and red indicate the polarity of the events (i.e., negative or positive changes of intensity).

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. 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 Robotics, and Google.

[1] `http://www.vicon.com/`

[2] `http://www.naturalpoint.com/optitrack/`

pose estimation even during very fast maneuvers, such as flips of a quadrotor (Figure 1a). In addition, since a DVS only streams relative brightness *changes* in the sensor's field of view, the computational load can be reduced drastically. However, to take full advantage of the DVS capabilities, we must rethink the way we interpret visual data.

The method presented in this paper estimates the 6 Degrees-Of-Freedom (DOF) pose of a DVS with respect to a known passive pattern. A naïve solution would be to accumulate the events occurred over a certain time interval and adapt known pose-estimation algorithms for standard CMOS cameras to these "integrated" images (an example "integrated" image is shown in Figure 1c, where we used an integration time of 2 ms). However, this is not desirable, because it would result in the same latency of a regular camera. Ideally, to have the lowest latency for the perception pipeline, one would want each single event to be reflected in small instantaneous changes of commands to the actuators. Therefore, we want to design methods that make use of the information contained in each single event. Since a DVS only detects changes of intensity, only scenes rich in gradient information are relevant. For simplicity, we chose a black square on a white background. However, our approach can be generalized to any planar shape or gradient map that is known a priori. Our algorithm starts by integrating events until the pattern is detected. Then, it tracks the line segments, which define the borders of the pattern, by updating both the lines and the pose at microsecond time resolution, as soon as a new event arrives.

B. 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. The key to achieve such high-speed performance lies in an event-based adaptation of the Hough-transform line-detection algorithm [9] to track the pencil.

Asynchronous, event-based optical flow was presented in [10], [11]. The authors adapted the Lucas-Kanade tracking algorithm to cope with the event-based nature of the DVS.

An Event-based Iterative Closest Point Algorithm (ICP) was used in [12] for closed-loop control of a micro gripper. The mean update rate was 4 kHz. However, the algorithm integrates events over a predefined time interval and only works in 2D.

In our previous work [13], 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.

Localization using a DVS on a ground robot was first presented in [14] and later extended to Simultaneous Localization And Mapping (SLAM) in [15]. 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 our previous work [16], we presented a visual-odometry pipeline using a DVS in combination with a standard CMOS camera. We used a probabilistic framework that updates the pose likelihood relative to the previous CMOS frame by processing each event individually as soon as it arrives. As in [15], the experiments were performed at relatively low speeds (up to 30 °/s), while the system was limited to planar motion. Although higher speeds would in principle be possible, this was not feasible with those settings due to the occurrence of motion blur in the CMOS camera at higher speeds. In contrast, in this paper we focus on full 6-DOF pose estimation using only DVS input and demonstrate successful pose tracking at rotational speeds up to 1,200 °/s, such as during quadrotor flips.

C. Contributions and Outline

The main contribution of this paper is an event-based, low-latency method for 6-DOF localization that works for high-speed maneuvers, which we demonstrate during quadrotor flips. Additionally, we provide a versatile method to generate realistic datasets of simulated trajectories on artificial scenes with ground truth. Since we use the DVS in the loop, we can generate ground truth with real sensor noise.

The remainder of the paper is organized as follows. In Section II, we describe the DVS and a calibration procedure. Our algorithm is described in Section IV and evaluated in simulation and with real experiments in Section VI.

II. Dynamic Vision Sensor

Standard CMOS cameras send full frames at fixed frame rates. On the other hand, retinal cameras such as a DVS have independent pixels that generate spike events at local relative brightness changes in continuous time. These events are timestamped and transmitted asynchronously at the time they occur using a sophisticated digital circuitry. Each event is a tuple $\langle x, y, t, p \rangle$, where 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 Figure 2.

III. Calibration

Since the optics of a DVS is the same as that of a regular camera, we use the standard pinhole camera model [18] to determine the intrinsic parameters (i.e., focal length, projection center, and distortion coefficients). For standard cameras, off-the-shelf calibration toolboxes based on regular patterns are the best choice [19]. However, it is not straightforward to use passive patterns with a DVS. Since relative motion is necessary to generate events, one would need to move the pattern in front of the DVS and integrate a sufficient number of

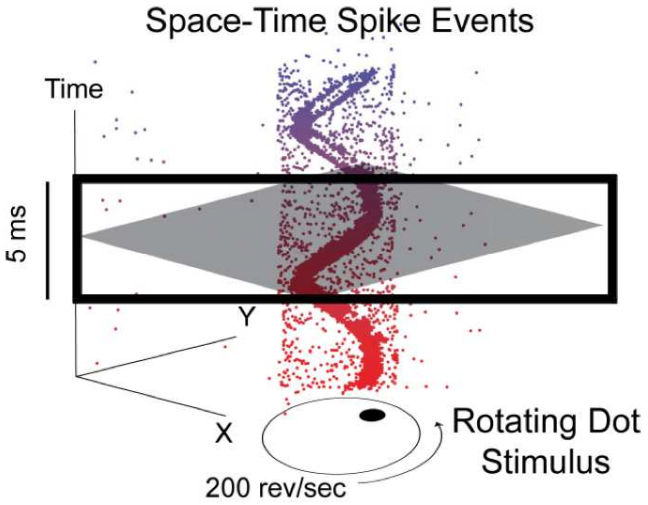
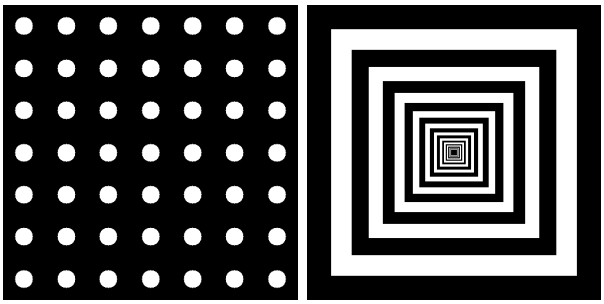


Fig. 2: Visualization of the output of a DVS looking at a rotating dot. Colored dots mark individual events. The polarity of the events is not shown. Events that are not part of the spiral are caused by sensor noise. Figure adapted from [17].



(a) Intrinsic calibration (b) Focus calibration

Fig. 3: Calibration patterns for the DVS.

events in order to "see" it.[3] Therefore, we calibrate the DVS using a computer screen with blinking patterns.[4] We use two different patterns: blinking dots (as depicted in Figures 3a and 8a) and concentric black-and-white squares (Figure 3b). We use the former for intrinsic-parameter calibration (we utilize a standard calibration tool, such as [20]) and the latter for focus adjustment (we proceed by manually tuning the focus of the camera until the squares appear sharp). To be independent of the distance to the screen, we chose the squares to be spaced and scaled logarithmically.

IV. Event-based Pose Estimation

Since a DVS only detects changes of intensity, only scenes rich in gradient information are relevant. For simplicity, we chose a black square on a white background (Figure 1). However, our approach can be generalized to any planar shape or gradient map that is known a priori. Our algorithm

[3]Remember that a DVS only generate asynchronous events; therefore, one would have to integrate the DVS events over a certain time interval in order to render an image that could be used with standard calibration tools.

[4]LED screens use pulse-width modulation of the background light for dimming. This high-frequency blinking generates events; thus, a static image on the screen appears blinking for the DVS.

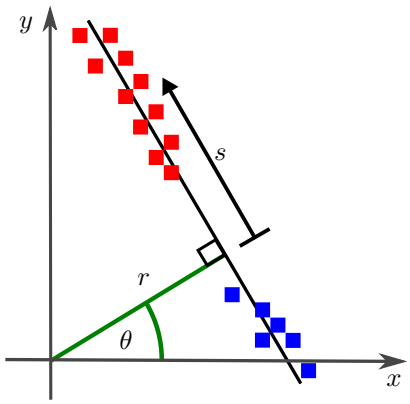


Fig. 4: Events (squares) belonging to a candidate line (black) detected in the Hough space (represented by r and θ). Events belonging to this line are ordered by their distance s and then clustered into line segments (e.g., red and blue). If a line segment is too short (less than 20 pixels), this is rejected by the algorithm (e.g., the blue cluster).

starts by integrating events until the pattern is detected. Then, it tracks the line segments, which define the borders of the pattern, by updating both the lines and the pose at microsecond time resolution, as soon as a new event arrives.

A. Initialization

Lines are detected using the Hough transform [9]. We chose the polar representation of lines and discretize the Hough space with equidistant bins of $7.5°$ and 2.5 pixels. Each event is added to the Hough space as it arrives. If a bin reaches a threshold of 25, it is considered a line candidate. If at least four distinct candidates are found, events are then assigned to each candidate based on their distance to the line. Events that are too far from the candidate line are removed. Then, all the events corresponding to a candidate line are ordered on the corresponding line (Figure 4). If the gap between two consecutive events on the same line is too large (8 pixels), they are considered to belong to two different line segments. Only segments with a minimum length of 20 pixels are considered for the next step.

We perform an exhaustive search to find 4-sided shapes in the set of detected line segments. We start with one segment and append additional segments if they can be added in clockwise order and the angle is between $45°$ and $135°$. If the fourth segment connects to the first one, the square is found. Then, we determine the four corners of the square by intersection of the estimated lines. Finally, we calculate the initial pose P from the homography relating the planar pattern and its image [21].

B. Line tracking

Lines are tracked in an event-based manner, which means, each event that arrives is used to directly update the pose estimate. When a new event arrives, we check whether it is close to one of the lines. If so, we use it to update that line and, subsequently, the pose estimate. Otherwise, we treat it

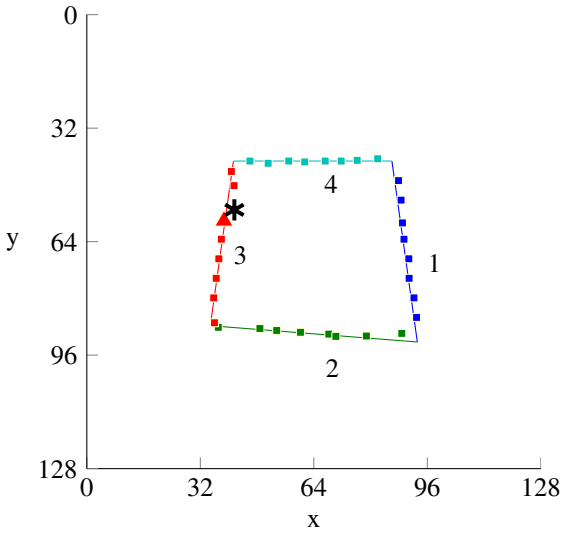
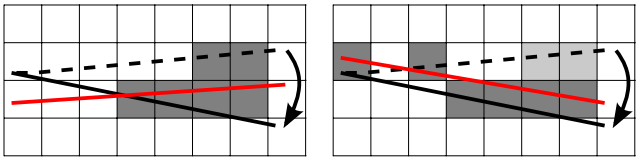


Fig. 5: Visualization of the tracking algorithm in the DVS image plane. A line is represented by 8 events (squares). When a new event (star) arrives, we check whether it is close to any line. If so, we replace the closest event with the new one. Otherwise, we treat it as an outlier and reject it. In this illustration, the event marked with the red triangle is replaced by the new event (represented by the star).



(a) Replacing oldest pixels (b) Replacing closest pixels

Fig. 6: Pixel-level schematics showing the problem of replacing the oldest event of a line during rotational motion. The true line (black dashed) is rotated (black solid). The line (red) is estimated by the events marked in gray. (a) Notice how replacing the oldest event shifts all events of a line towards one end, thus, corrupting the line estimate. (b) Instead, replacing the closest pixels does not suffer from this issue.

as an outlier (i.e., the event was either generated by another object or by sensor noise) and reject it.

We represent each line with N past events. A new event replaces the *closest* one, as illustrated in Figure 5. Note that always replacing the oldest event would eventually corrupt the line estimate as illustrated in Figure 6. The choice of N is a tradeoff between latency and accuracy. While using many points would result in smoother trajectories, higher latency would be introduced. We found that $N = 8$ is good tradeoff for our setup.

C. Pose estimation

We update the pose by minimizing the sum of squared distances between the reprojection of each line and the events belonging to it, that is

$$P^* = \arg\min_P \sum_{l=1}^{4} \sum_{i=1}^{N} \| d\left(\pi(L_l, P), e_{l,i}\right) \|^2, \qquad (1)$$

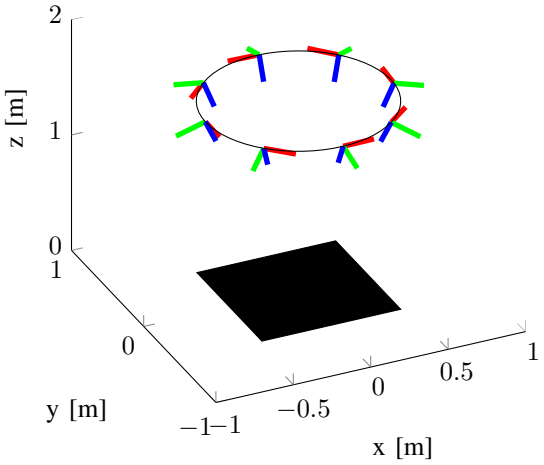
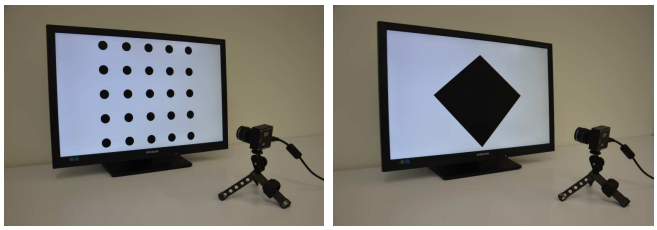


Fig. 7: Visualization of the pattern (black square) and the circular trajectory of the virtual camera.



(a) Calibration (b) Rendered scene

Fig. 8: Setup to simulate artificial scenes with ground truth. (a) A regular pattern is used to estimate the perspective transform between the DVS and the screen (b) The scene is rendered from a virtual camera that follows a given trajectory.

where L_l denotes a line belonging to the pattern, $\pi(\cdot,\cdot)$ projects a line onto the image plane, $e_{l,i}$ denotes an event i belonging to line l, and $d(\cdot,\cdot)$ returns the distance between the point and the line. The lines are updated with the new pose estimate P^* by projecting the pattern onto the image plane.

V. DVS Simulation

To assess the quality of our pose estimation algorithm, we need datasets with ground truth. To do this, we generated virtual camera views on a computer screen by simulating trajectories of a camera moving in front of a pattern, as depicted in Figure 7. Instead of simulating a DVS output (which is not trivial given the sophisticated digital circuitry of a DVS), we placed a real DVS in front the screen and recorded the generated artificial views (Figure 8). Having the DVS in the loop has the advantage that the sensor noise levels are real.

We denote the world frame of the artificial scene with a subscript W, the virtual camera frame with V, the computer screen frame with S, and the DVS frame with D. A world point X_W is mapped onto the virtual camera through a

perspective transformation:

$$X_V = K_V \left(R_{VW} X_W + T_{VW} \right). \tag{2}$$

Since the output of the virtual camera is independent of the screen size, a scale factor α is introduced,

$$X_S = K_S X_V = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} X_V. \tag{3}$$

Because the screen and the DVS are not aligned, screen points are also mapped through a perspective transformation:

$$X_D = K_D \left(R_{DS} X_S + T_{DS} \right). \tag{4}$$

Substituting (2) and (3) into (4) gives

$$X_D = K_D \left(R_{DS} K_S K_V \left(R_{VW} X_W + T_{VW} \right) + T_{DS} \right), \tag{5}$$

where the virtual camera trajectory with respect to the world frame $P_{VW}(t) = [R_{VW}(t)|T_{VW}(t)]$ is a continuous function of time t.

We estimated the pose of the DVS with respect to the screen automatically before each recording, using a blinking pattern as described in Section III.

The scene was rendered in real-time using OpenGL.[5] For each event from the DVS, we evaluated the virtual camera pose $[R_{VW}(t)|T_{VW}(t)]$ at the specific event time. Thus, we know the ground truth DVS pose for each event.

VI. Experimental Evaluation

We evaluated our algorithm both with simulated data and real data from a quadrotor performing flips. In the evaluation, we used the angle of the angle-axis representation as an error metric for orientation.

A. Simulated Data

We used the simulation setup described in Section V. We simulated a planar scene containing a single black square on the $x-y$ plane centered in the origin of the world frame on a white background, as depicted in Figure 7. We generated a circular trajectory at constant altitude z and commanded the angular velocity of the virtual camera such that its optical axis always intersected the origin of the world frame, that is:

$$P_{VW}(t) = \left[\begin{array}{ccc|c} c(\alpha) & s(\alpha) & 0 & 0 \\ -s(\alpha)c(\gamma) & c(\alpha)c(\gamma) & s(\gamma) & 0 \\ s(\alpha)s(\gamma) & -c(\alpha)s(\gamma) & c(\gamma) & z \end{array}\right],$$

where $s(\cdot) = \sin(\cdot)$, $c(\cdot) = \cos(\cdot)$, $\alpha(t) = 2\pi t/T$, $\gamma = 200°$, $z = 1.7\,\text{m}$, and $T = 2\,\text{s}$ is the time it takes to complete a full circle. The square's side length is $0.9\,\text{m}$.

Figure 9 shows the error of our pose estimation algorithm. The mean position error is $1.47\,\text{cm}$ with a standard deviation of $0.72\,\text{cm}$. The mean orientation error is $2.28°$ with a standard deviation of $1.08°$.

[5] http://www.opengl.org/

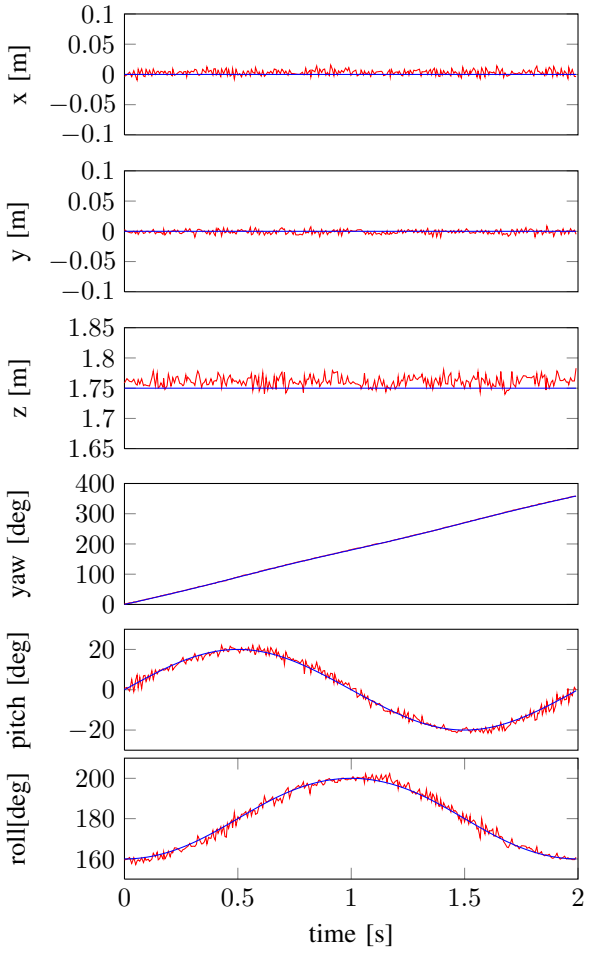


Fig. 9: Estimated trajectory (red) compared to ground truth (blue) on a simulated dataset. The trajectory is the one depicted in Figure 7, which was generated as described in Section V.

B. Real Data

1) Experimental Setup: We used a DVS with a $2.8\,\text{mm}$ S-mount lens. We calibrated it as described in Section III and found its focal length to be 69 pixels. We mounted the DVS on a Parrot AR.Drone 2.0 equipped with an Odroid U2 onboard computer (Figure 10). The event stream was recorded onboard and streamed to a laptop over WiFi to visualize data in real-time. In addition to the DVS output, we also recorded the video of the front-looking standard CMOS camera.

As a pattern, we used a black square ($0.9 \times 0.9\,\text{m}$) attached to a white wall, the origin of the world frame coinciding with the center of the pattern, x being oriented perpendicularly to the wall and z parallel to the gravity vector.

Ground truth was captured using an OptiTrack motion capture system. Markers were placed all around the body of the quadrotor to ensure tracking during flips.

2) Evaluation: We controlled the quadrotor to perform multiple flips around the principal axis of the camera (roughly aligned with the x-axis of the world frame). The peak angular speed (i.e., roll rate) during such high-speed maneuvers was measured to be $1{,}200\,°/\text{s}$ (cf. Figure 15).

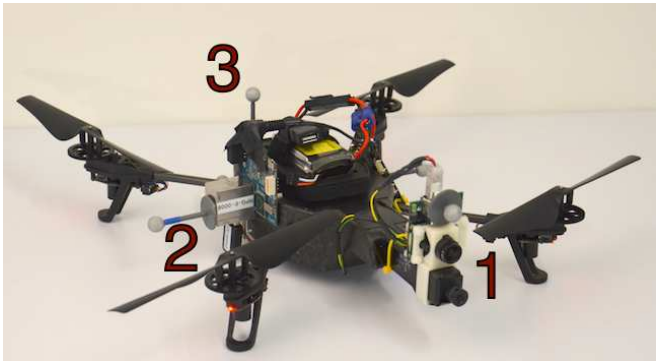


Fig. 10: Experimental setup on an AR.Drone. 1) The DVS (top) and a standard CMOS camera (bottom), 2) Odroid U2 computer for recording and streaming the DVS data over WiFi, and 3) markers to collect ground truth with a motion capture system.

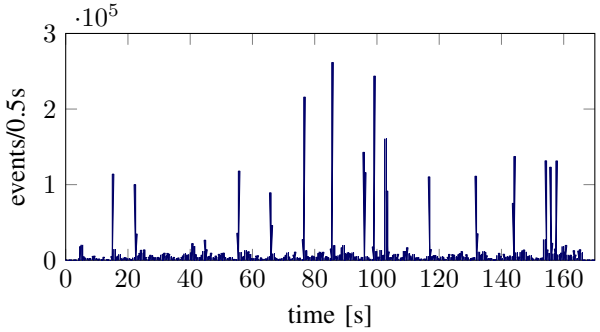


Fig. 11: Number of events as a function of time (we counted events in a time interval of 0.5 s) during an experimental session containing 15 flips. This plot clearly shows that during flips the density of events is much larger than during near-hover flights. During the first and last 5 s, the quadrotor is resting on the floor; thus, virtually no events are generated.

While this results in severe motion blur effects for the standard CMOS camera (cf. Figure 13), for the DVS we can still see very sharp lines if we integrate the events for an appropriate period of time (cf. Figure 12). However, our algorithm does not rely on such integrated images, but updates the 6-DOF pose of the robot by processing each event individually as soon as it arrives. The estimated trajectory for three consecutive flips with ground truth is shown in Figure 14. The mean position error is 10.8 cm with a standard deviation of 7.8 cm. The mean orientation error is 5.1° with a standard deviation of 2.4°.

During our experimental flight session, we recorded data for a total of 25 flips. Our algorithm could track the DVS trajectory for 24 of them (96 %). In only one case, tracking was lost during the flip. Figure 11 shows the number of events as a function of the time during the first 15 flips. As observed, the density of events generated during flips is much larger than during near-hover flights.

3) Comparison with Theoretical Limit: Since our pose estimate is very noisy, we are interested to determine the accuracy of the pose estimate that one could achieve with an "ideal" CMOS camera (not a DVS) characterized by infinite frame rate and no motion blur, but having the same

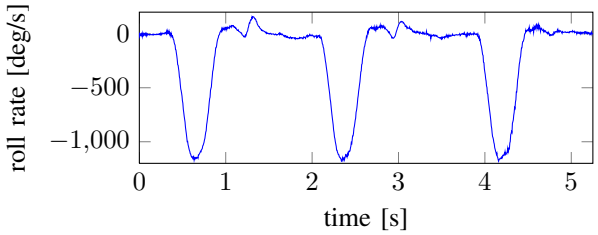
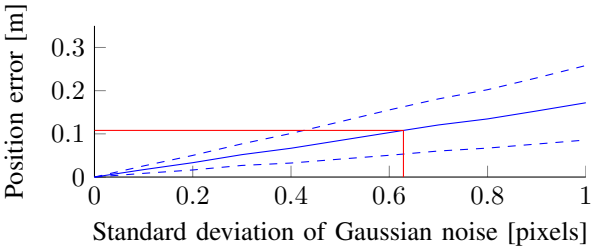
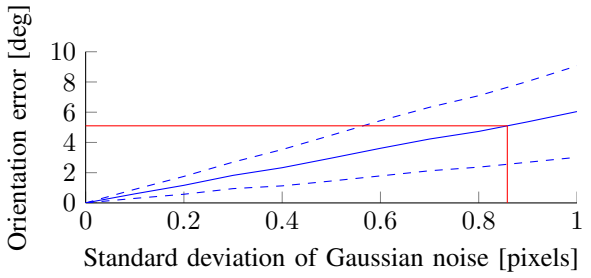


Fig. 15: Roll rate of the trajectory shown in Figure 14. The maximum roll rate is 1,200 °/s during a flip.



(a) Position error

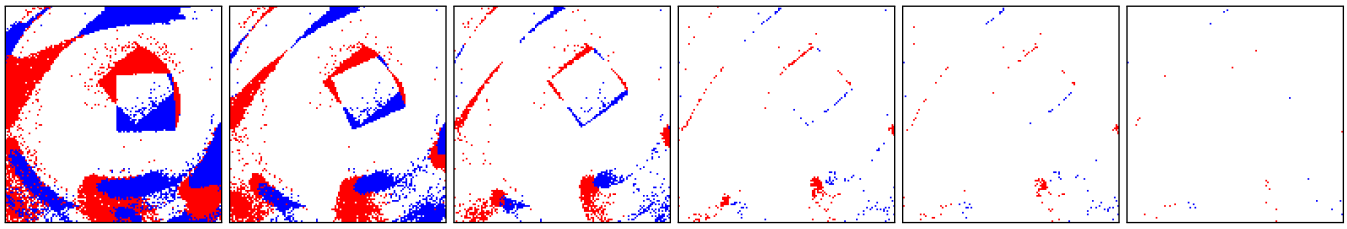


(b) Orientation error

Fig. 16: Plots of mean pose error (solid) ± one standard deviation (dashed) in world coordinates for an ideal sensor with Gaussian noise in the image plane. The configuration is close to the experimental setup presented in Section VI-B. The mean position and orientation error of the quadrotor flip experiment is marked in red and corresponds to 0.63 and 0.86 pixels, respectively.

resolution as the DVS (i.e., 128 × 128 pixels). This problem is equivalent to characterize the pose estimation error of a CMOS camera in static settings in a configuration close to the real experimental setup (i.e., same intrinsic parameters, same pattern size, and same relative position between camera and pattern). Clearly, the answer depends on the accuracy (pixel or sub-pixel) of the edge detector. We addressed this by means of Monte-Carlo simulation, by adding Gaussian noise with different variances to all image points and by optimizing the pose by minimizing the reprojection error. We ran this simulation 1,000 times for each variance value.

The resulting error in position and orientation is shown in Figure 16. The position and orientation accuracies of the DVS-based pose estimator described in this paper are indicated with horizontal red lines, corresponding to a mean position error of 10.8 cm and a mean orientation error of 5.1° respectively. As observed, these accuracies corresponds to a standard deviation of the error, which is in both cases smaller than 0.9 pixels. Since this can be considered as



(a) $\Delta T = 33\,\text{ms}$ (b) $\Delta T = 15\,\text{ms}$ (c) $\Delta T = 5\,\text{ms}$ (d) $\Delta T = 1\,\text{ms}$ (e) $\Delta T = 0.5\,\text{ms}$ (f) $\Delta T = 0.1\,\text{ms}$

Fig. 12: Integrated events of the DVS over different time intervals. Blue and red indicate the polarity of the events.

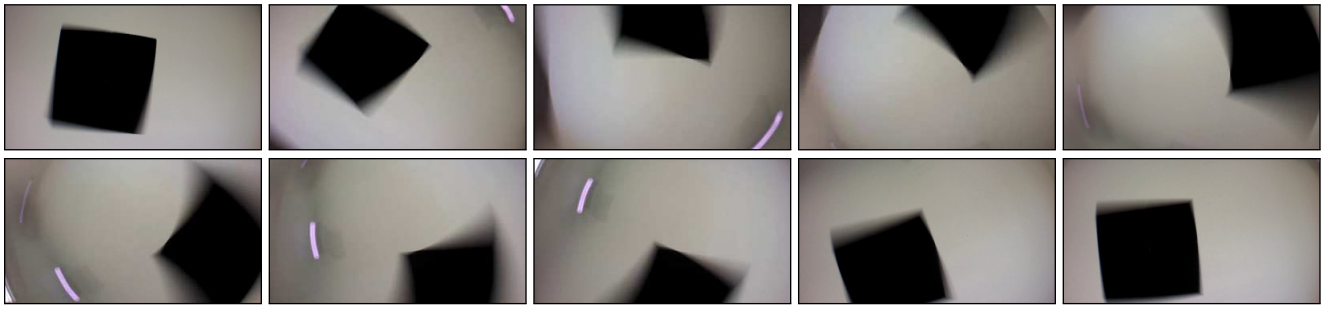


Fig. 13: Standard CMOS camera frames at 30 Hz during a flip (from left to right). Motion blur is clearly visible in all frames except the first and last one. The violet traces correspond to the LED lights of the OptiTrack cameras.

reasonably precise, we claim that the error of our DVS-based pose estimation is mainly caused by the poor resolution of the DVS (i.e., 128×128 pixels) and the results would significantly improve with a higher-resolution DVS.

VII. Conclusion

In the last few years, we have witnessed impressive demonstrations of aggressive quadrotor flights and acrobatics using motion capture systems. Flight maneuvers using onboard sensors are still slow. At the current state, the agility of a robot is limited by the latency of its sensing pipeline. To obtain more agile robots, we need to use faster sensors. A Dynamic Vision Sensor (DVS) only transmits pixel-level brightness changes at the time they occur with microsecond resolution, thus, offering the possibility to create a perception pipeline whose latency is negligible compared to the dynamics of the robot. This technology is the most promising candidate for enabling highly aggressive autonomous maneuvers with flying robots. The current DVS prototypes suffer from a relatively poor resolution, which is currently being worked upon. In this paper, we presented the first onboard perception system for 6-DOF localization during high-speed maneuvers using a DVS. We demonstrated robust motion tracking during quadrotor flips with angular speeds up to 1,200 °/s. Future work will involve a generalization of the approach to arbitrary environments and the use of the DVS in closed-loop control.

Acknowledgement

We gratefully acknowledge the contribution of Flavio Fontana and Matthias Faessler for helping with the quadrotor experiments. We would also like to thank Tobi Delbruck and Vicente Villeneuva for helping us making the DVS lightweight enough for our experiments.

References

[1] 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.

[2] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK, May 2010.

[3] S. Shen, N. Michael, and V. Kumar, "Autonomous Multi-floor Indoor Navigation with a Computationally Constrained MAV," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[4] S. Grzonka, G. Grisetti, and W. Burgard, "A Fully Autonomous Indoor Quadrotor," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[5] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

[6] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *J. of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[7] 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.

[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)*, Taipei, Taiwan, May 2009.

[9] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[10] 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.

[11] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-Based Visual Flow," *IEEE Trans. Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, 2014.

[12] 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.

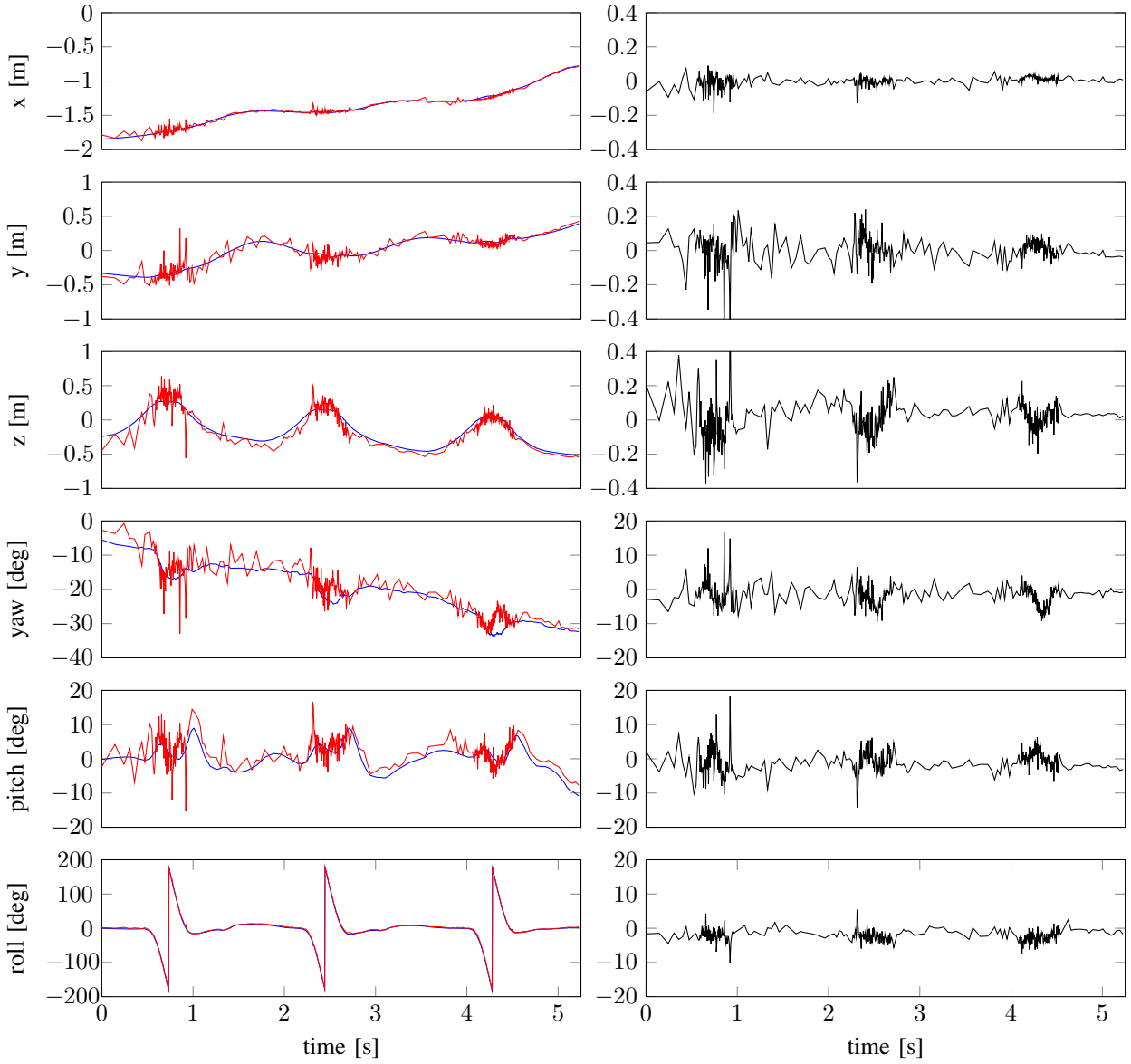


Fig. 14: Estimated trajectory (red) with ground truth (blue) and errors (black) for three consecutive flips with a quadrotor. Notice how the error gets smaller towards the end of the trajectory. This happens because the quadrotor moves closer to the pattern, which in turn appears larger in the DVS. Also notice how the density of pose estimates gets higher during flips. This occurs because the pose is updated whenever a new event arrives and the number of events increases during faster relative motion (cf. Figure 11).

[13] 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)*, Tokyo, Japan, Nov. 2013.

[14] D. Weikersdorfer and J. Conradt, "Event-based Particle Filtering for Robot Self-Localization," in *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, Guangzhou, China, Dec. 2012.

[15] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous Localization and Mapping for event-based Vision Systems," in *Intl. Conf. on Computer Vision Systems (ICVS)*, St. Petersburg, Russia, July 2013.

[16] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014.

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

[18] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY: Springer-Verlag New York, Inc., 2010.

[19] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[20] Y. Bouguet. Camera Calibration Toolbox for Matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[21] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.