Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor

Andrea Censi, Jonas Strubel, Christian Brandli, Tobi Delbruck, Davide Scaramuzza

***Abstract*—At the current state of the art, the agility of an autonomous flying robot is limited by the speed of its sensing pipeline, as the relatively high latency and low sampling frequency limit the aggressiveness of the control strategies that can be implemented. To obtain more agile robots, we need faster sensors. A Dynamic Vision Sensor (DVS) encodes changes in the perceived brightness using an address-event representation. The latency of such sensors can be measured in the microseconds, thus offering the theoretical possibility of creating a sensing pipeline whose latency is negligible compared to the dynamics of the platform. However, to use these sensors we must rethink the way we interpret visual data. We present an approach to low-latency pose tracking using a DVS and Active Led Markers (ALMs), which are LEDs blinking at high frequency (>1 KHz). The DVS time resolution is able to distinguish different frequencies, thus avoiding the need for data association. We compare the DVS approach to traditional tracking using a CMOS camera, and we show that the DVS performance is not affected by fast motion, unlike the CMOS camera, which suffers from motion blur.**

I. Introduction

Autonomous micro helicopters will soon play a major role in tasks like search and rescue, environment monitoring, security surveillance, inspection, etc. A key problem in aerial-vehicle navigation is the stabilization and control in six degrees of freedom. Today's systems handle well the attitude control. However, without a position control, they are prone to drift over time. In GPS-denied environments, this can be solved using onboard sensors, such as cameras [1] or laser rangefinders [2]; however, the achievable vehicle maneuvers are still too slow compared to those attainable with off-board motion-tracking systems (e.g., Vicon) [3].

The agility of an autonomous flying robot is limited by the speed of the sensing pipeline. More precisely, "speed" can be quantified in *observations frequency* and *latency* (Fig. I.1). In current state-of-the art autonomous navigation applications [1] cameras give observations with frequency of 15–30 Hz and the total latency, from acquiring the images to processing them, is in the order of 50–250 ms. To obtain more agile systems, we need to use faster sensors and low-latency processing.

In this paper, we consider the use of a Dynamic Vision Sensor (DVS) for pose tracking. The main difference between a DVS and a normal CMOS camera is that the DVS output is a stream of *events* that encode *changes* in the brightness. Each event encodes the location of the change, whether there was a positive or negative change in brightness, and has a 1 μs timestamp. These events are not unlike spikes in a biological visual system; however, while retinal ganglion cells show latencies of around 200 ms, the DVS chip has a latency of 15 μs.

Theoretically, using a DVS we could obtain sensing pipelines with a negligible latency compared to dynamics of the platform. We are a few years to the goal, however. On the hardware side, the DVS camera, though currently available commercially, has a few limitations, such as the limited resolution (128×128 pixels), which will be increased in the next generation of prototypes currently in development. On the software side, to take full advantage of this data we need to rethink completely the way we design robotic sensing pipelines. It is possible to integrate the events of a DVS camera to simulate a regular CMOS frame, on which to do standard image processing, however, that is not desirable, because that would give the same latency of a regular camera. Ideally, to have the lowest latency for the sensing pipeline, one would want each single event to be be reflected in a small but instantaneous change in the commands given to the actuators. Therefore, we consider approaches that possibly use the information contained in each single event.

In this paper, we consider the application of pose tracking based on Active LED Markers (ALMs), which are infrared LEDs blinking at high frequency (> 1 kHz). The DVS is fast enough to be able to distinguish different blinking frequencies, so that we can also uniquely assign an observable identity to each marker. We envision that this system could be used for inter-robot localization for high-speed acrobatic maneuvers, or that, in applications such as rescue robotics, these markers could be left in the environment to facilitate cooperative mapping.

One approach to using the DVS data is to cluster the events in order to find spatio-temporal features, like

A. Censi is with the Computing & Mathematical Sciences department, Division of Engineering and Applied Sciences, California Institute of Technology, Pasadena, CA. E-mail: andrea@cds.caltech.edu. J. Strubel and D. Scaramuzza are with the Department of Informatics, University of Zurich. E-mail: jonas.strubel@uzh.ch, davide.scaramuzza@ieee.org. C. Brandli and T. Delbruck are with the Institute of Neuroinformatics, University of Zurich and ETH Zurich. E-mail: {chbraen,tobi}@ini.uzh.ch.

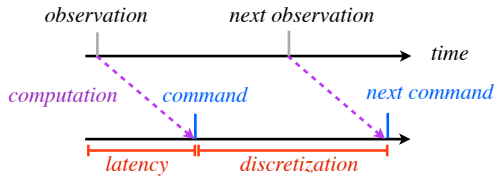
A. Censi was partially supported by the US National Science Foundation (NRI program, grant #12018687) and DARPA (MSEE program, grant #FA8650-11-1-7156). C. Brandli, T. Delbruck, and D. Scaramuzza were partly supported by the Swiss National Science Foundation through project number 200021-143607 ("Swarm of Flying Cameras") and the National Centre of Competence in Research Robotics. Full disclosure: T. Delbruck has a financial participation in inilabs GmbH, the start-up that commercially distributes the DVS camera.

points or lines, that are then tracked through time [4, 5, 13]. This approach works well when the camera is static, because the output is spatiotemporally sparse.

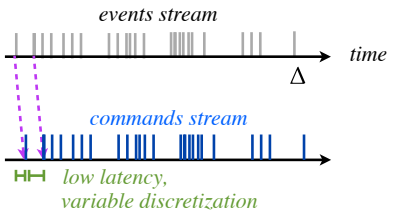
The algorithm presented in this paper uses a different approach. We found out that mounting a DVS camera on a flying robot creates a new set of challenges. Because of the apparent motion of the environment, the events are not spatiotemporally sparse anymore. Moreover, while in controlled conditions the DVS camera parameters can be tuned to obtain the best performance, a robot must be able to work in a wider range of environmental conditions and be robust to interferences. To achieve this robustness we have developed an approach that sacrifices some latency to be more robust to noise and unmodeled phenomena. We accumulate the events perceived in thin slices of times corresponding to the blinking frequency (1 ms slice for 1 kHz data). This allows to do detection of the ALMs position in image space. On top of this, we use a particle filter for tracking the position in image space of each detection, and a disambiguation stage to obtain coherent hypotheses on the joint position of the markers. Finally, we reconstruct the pose using a standard approach to rigid reconstruction.

We evaluate our method in tracking the pose of a drone during an aggressive maneuver (a flip). We compare our methods with a traditional approach, using a CMOS camera and a feature-based visual odometry method. We verify that our method, with a latency of 1 ms, is able to reacquire tracking instantaneously regardless of the fast motion, while the CMOS data is corrupted by motion blur. We evaluate the reconstruction accuracy using an OptiTrack system and find values that are compatible with the low spatial resolution (128×128) of the DVS, which proves to be the current limitation of this approach.

Software, datasets, and videos illustrating the method are available at the website http://purl.org/censi/2013/dvs.



(a) Traditional architecture



(b) Low latency, event-based architecture

Figure I.1. To improve the agility of autonomous flying robots, we need to improve the total latency of the sensing pipeline. Using a device like a Dynamic Vision Sensor (DVS) we can theoretically obtain a sensing pipeline which has microsecond latency.

II. The Dynamic Vision Sensor (DVS)

A regular CMOS camera records a visual scene by taking a stroboscopic series of still frames. Computer vision and robot perception methods work on the analysis of each frame separately. This established approach has some fundamental drawbacks: each pixel gets sampled and processed over and over again at each frame, independently of its relevance to the decision to be taken, or whether its value changed. Much processing power is used for considering redundant information, which translate into high latencies and low frame rates. In contrast to this, we observe that in biological systems there is redundancy suppression already on the "sensor": as recordings of nerve cells coming from the eye show, the retina mostly responds to *changes* in the perceived brightness.

The field of *neuromorphic engineering* tries to implement the computations done by nervous systems as VLSI circuits. The computation is *analogic*: currents, voltages and charges are used for computing, rather than binary representations. The resulting circuits are also *asynchronous*: like nervous cells, they operate independently of an external clock for state transitions. There has been a large amount of research in neuromorphic sensory systems of different scale and complexity [**liu10neuromorphic**]. The first system to be commercially available is the so-called "silicon retina" or *dynamic vision sensor* (DVS) [7].

Each pixel in the DVS operates independently and asynchronously from the others. The photons that reach a photodiod produce a photocurrent, which is converted into a voltage. This voltage gets continuously compared to the last sampled voltage. As soon as the difference between these two voltages is larger than a certain threshold, the pixel requests to send an *event* off chip. After the address and the timestamp of this event has been registered from a CPLD at the chip periphery, the pixel is reset, and the most recent voltage gets sampled to be compared against successive voltages. Depending on the sign of the difference, either a P event (positive) or an N event (negative) is generated. P events indicate that the brightness increased, N events indicated that it decreased. All of this computation is done without digitizing the signal.

The parameters that modulate the pixel behavior are dynamically programmed using a set of bias currents. Some of the parameters that can be dynamically changed are the temporal contrast frequency cutoff, the P & N threshold, and the event frequency cutoff. Each event carriers the following information: the *address* of the pixel that observed the change (equivalent to its x, y coordinates), the *polarity* (P or N), and a timestamp, which has a 1 µs resolution.

The sensor therefore only produces output if the observed scene changes. This is not only the case if the intensity of a light source gets modulated (as with a blinking LED), but also if a static scene moves relative to the sensor. If the sensor itself is moved it perceives all the edges and contours in its visual field.

Sensing the world with a set of autonomous, change-sensitive pixels has various advantages compared to traditional image sensors:

- Since the photocurrent gets converted to a voltage on pixel level, the brightness measurement does not require a uniform exposure time for all pixels. This leads to the high dynamic range of 120dB which is capped by the exposure time in conventional image sensors.
- By only sensing changes, the sensor performs on-chip data compression. This does not only make the amount of output data dependent on the activity in a scene but it also focusses the processing effort on the region of interest where it is changing. This focusing leads to controller update intervals of down to 125 µs [5].
- Redundant information does not occupy the output bus and tiherefore relevant information can be signaled very fast. The sensor chip itself has a latency of 15 µs which allows to come up with sensor-actuator latencies (including sensor and actuator USB communication and laptop processing) of 3 ms [4].
- The high resolution of the event timestamps can be used to investigate the dynamics in a scene such as motion or blinking frequencies. While conventional vision sensors only allow to investigate the scene at given times, the DVS produces a continuous stream of information.

The main drawbacks of the current generation of DVS are its low resolution of 128×128, and the inability to sample the absolute brightness levels like a normal camera.

The low resolution is due to the prototype fabrication process used (350 nm) and the fact that each pixel is associated to a complex circuit carrying on the analogous computation. The static scene content cannot be accessed because the according technology was not ready for the first series of sensors. But since the field of event-based vision sensors is steadily growing [**delbruck10activity**], these limitations will soon be overcome.

Some of the latest developments include higher temporal contrast sensitivity of 1.5% [**serrano13128**] and higher spatial resolution of 240×180 or 304×240, accompanied with a readout channel for pictures and movies that simulates a CMOS camera [**posch11qvga**, **berner13240**].

Ongoing efforts include increasing the availability of the embedded version of the sensor (eDVS, used in [5]), which weighs only a few grams and measures $30 \times 50 \times 50$ mm. Furthermore, future versions of event based vision sensors are expected to include a USB 3.0 interface for higher data transmission rates.

III. Hardware setup and event data

This section describes the hardware setup and gives an intuition of the event data that a DVS produces.

1) DVS and its interface: The DVS attaches to a computer using a common USB port. The camera is distributed with a portable software framework written in Java called jAER [10]. This project, developed in C++ for maximum efficiency, needed a special driver to be developed, based on the Thesycon USB device driver [11]. The events are transmitted from the camera in packets of up to several hundred events; however, each event is independently tagged at the source with its proper timestamp.

2) Active LED Markers (ALMs): The LED controller is based on the bronze board [12] from inilabs, which is based on an AVR32. We used infrared LEDs since the DVS is most sensitive in the infrared spectrum.

In our setup we could easily change the PWM frequency to the LEDs. An upper bound on the detectable frequency depends on the power of the LEDs and the distance to the camera; a DVS camera is not magic: there must be a large enough change in the number of photons reaching the photoreceptor to trigger an event. In our setup we found 2 KHz to be an adequate number. The frequencies for each LEDs were then decided in the interval 1–2 KHz making sure we did not choose frequencies with common harmonics. A reasonable lower bound on the blinking frequency was found experimentally to be 1 KHz to minimize the confusion between with background motion (see Section III-4 below).

3) Statistical properties of event sequences: Fig. III.1 gives an intuition of how the data looks like. In this scenario both the ALMs and the DVS are fixed. Fig. III.1*a* shows the histogram of the number of events coming from a particular pixels. The three peaks are the three ALMs ($f = 500, 700, 1000$ Hz). The number of events is different for each peak because the frequencies differ. The halo in Fig. III.1*a* cannot be explained by the refractive properties of the optics, and is probably due to non-ideal local interactions among neighbors in the sensing array.

Fig. III.1*b* shows the sequence of events obtained from one particular pixel, corresponding to an ALM with a frequency of 1 KHz. There is a different number of P and N events, which tells us that we cannot really interpret the DVS data as simply the derivative of the image.

For this particular data, the P events arrive noisily, while the N events arrive more regularly. Note that the what we observe here is the combination of the LED dynamics with the dynamics of the photoreceptor and the nonlinear detector. Experimentally we found that the interval between successive P/N transitions is repeatable: we have observed that the jitter is well approximated by a Gaussian with standard deviation equal to 6 μs (Fig. III.1*c*).

4) Effect of motion: Things get more complicated when the camera is moving, because the apparent motion of the environment creates changes in luminance that are unrelated to the ALMs. However, we have found that we can discriminate between the two types of events based on their temporal statistics.

Fig. III.2 shows the histogram of frequencies of the P/N transitions in three scenarios: a) a fixed camera looking at fixed ALMs; b) a moving camera looking at fixed ALMs; c) a moving camera with no LEDs. From this data we can see that the motion of the camera generates a large number of events, but at low frequencies, and notwithstanding the motion, we can still clearly see the peaks originated from the markers. Therefore, if we choose the marker frequencies high enough, we can filter for the camera motion just by ignoring the events

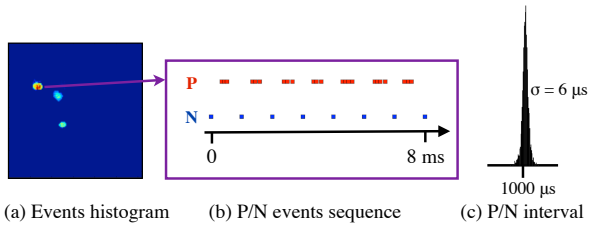


Figure III.1. Example event sequence from a DVS looking at an Active LED Marker (ALM). Subfigure (a) shows the histogram of events seen from a fixed camera looking at three ALMs. The difference in numbers is due to the different frequencies of the ALMs. Subfigure (b) shows a slice of the events seen at a particular pixel near the center of one of the ALMs which has a blinking frequency of 1 KHz. The data is a series of events with positive (P) and negative (N) events. (c) The sequence of NP transitions are highly regular; in this data we observed that the distribution of the intervals is well approximated by a Gaussian with mean 1000 μs and standard deviation $\sigma = 6\,\mu s$.

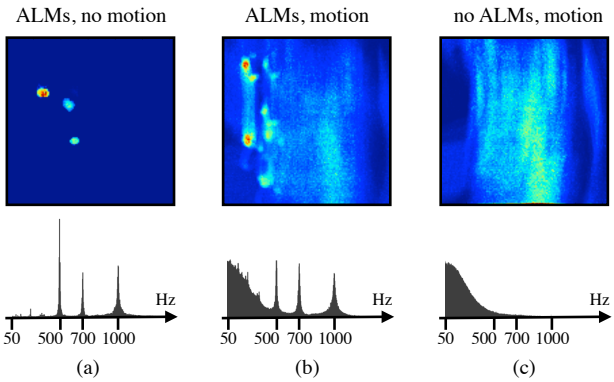


Figure III.2. Comparison of transition events statistics in three cases: a) visible markers, without motion; b) with camera motion; c) with camera motion but no marker visible. Note in (c) that the motion of the camera creates apparent motion of the environment and a large number of events; however they are of a low frequency. In this case, the events from the background are negligible after 700 Hz, though this depends on the statistics of the environment and the speed of the motion. We can choose the frequencies of the markers high enough such that they are not confused as background motion.

corresponding to frequencies under a certain threshold.

IV. DVS-BASED ACTIVE LED MARKER TRACKING

This section describes our method for tracking the position of a set of ALMs from the output of a DVS. The input to the algorithm is a sequence of events representing the change of luminance in a single pixel. The output is an estimate of the pose of the quadrotor. We describe the algorithm as a sequence of stages that process asynchronous events; in principle, several of them could be implemented in hardware.

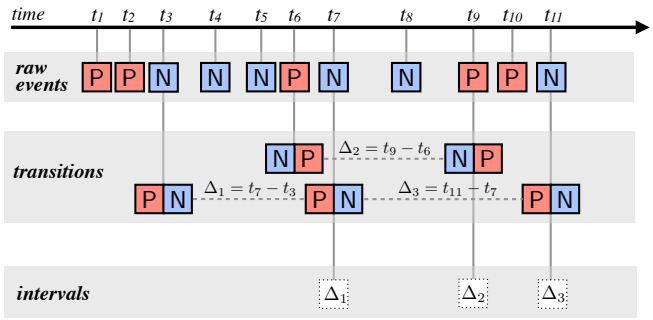


Figure IV.1. A single pixel produces an irregular series of raw events, with polarity either positive (P) or negative (N), each with its own timestamp. The first stage of processing consists in looking for positive-to-negative (NP) or negative-to-positive (PN) transitions. The second stage consists at looking at two successive transitions of the same kind. For example, two successive NP transitions at time t_3 and t_7 generate an hyper-event with interval $\Delta = t_7 - t_3$. Assuming that these events are generated by a blinking ALM, the Δ is a good robust estimator of the blinking period.

A. Raw events

The input to the algorithm is the sequence of events in the address-event representation. Each event can be represented by a tuple

$$\langle t_k,\ p_k,\ \langle x_k, y_k \rangle \rangle,$$

where:

- the scalar t_k is the timestamp of the event generated; these are not necessarily equispaced in time.
- the value $p_k \in \{\text{P}, \text{N}\}$ is the *polarity*, which is either positive (P) or negative (N), according to whether the luminance increased or decreased;
- the coordinates $\langle x_k, y_k \rangle \in \{0, \ldots, 127\} \times \{0, \ldots, 127\}$ identify the pixel that triggered the event.

B. Transitions

The first stage transforms the sequence of $\{\text{P}, \text{N}\}$ events into a sequences of *transition events* $\{\text{NP}, \text{PN}\}$. This stage is independent for each pixel. Consider only the events that are produced by a given pixel at coordinates $\langle x, y \rangle$. At all times, we remember the last event timestamp t_{k-1} and the polarity $p_{k-1} \in \{\text{P}, \text{N}\}$. Every time the polarity of the current event p_k is different than p_{k-1}, we create a *transition event*. If the polarity is the same, no transition event is generated. This is described by the rules in Table I.

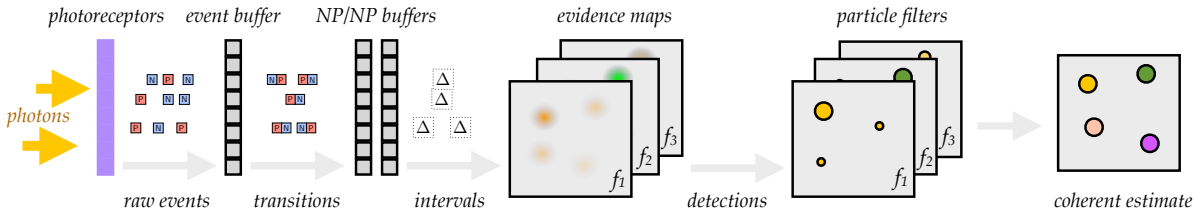


Figure III.3. Our method proceeds in stages. We buffer the raw events, which have either positive (P) or negative (N) polarity, as to find the transitions, either positive-to-negative (NP) or negative-to-positive (PN). Then, we look at the intervals Δ between two transitions of the same type. These will be converted into votes in an evidence map tuned to each frequency. From the evidence map we extract local maxima, which are the instantaneous detections on where is each ALM. The rest of the method is standard: for each frequency we use a particle filter to be robust to missed detections; then we choose the combination of particles that gives a coherent global estimate for all ALMs.

A transition event is a tuple

$$\langle t_k,\, q_k,\, \langle x_k, y_k \rangle \rangle,$$

where:

- the scalar t_k is the timestamp of the *second* event that triggered the transition.
- the value $q_k \in \{\text{P}, \text{N}\}$ is the transition polarity, which is either positive-to-negative (NP) or negative-to-positive (PN).
- $\langle x_k, y_k \rangle$ are the coordinates.

Table I
FROM RAW EVENTS TO TRANSITIONS

last event	*current event*	*transition event*
$\langle t_{k-1},\, \text{P},\, \langle x, y \rangle \rangle$	$\langle t_k,\, \text{P},\, \langle x, y \rangle \rangle$	none
	$\langle t_k,\, \text{N},\, \langle x, y \rangle \rangle$	$\langle t_k,\, \text{NP},\, \langle x, y \rangle \rangle$
$\langle t_{k-1},\, \text{N},\, \langle x, y \rangle \rangle$	$\langle t_k,\, \text{P},\, \langle x, y \rangle \rangle$	$\langle t_k,\, \text{PN},\, \langle x, y \rangle \rangle$
	$\langle t_k,\, \text{N},\, \langle x, y \rangle \rangle$	none

C. Hyper-transitions

The next stage of processing looks at the interval between successive transitions of the same type. For each pixel, we remember the last transition of either type (NP or PN) in a separate storage; then, for each transition, we generate a "hyper-transition", which is a tuple of the kind

$$\langle t_k,\, \Delta_k,\, \langle x_k, y_k \rangle \rangle,$$

where Δ_k is the interval between transitions of the same kind, and $\langle x_k, y_k \rangle$ are the coordinates. Note that we dropped the polarity of the transitions, as they are not needed in the following stages.

D. Evidence maps

We suppose to have been given a set of n frequencies $\{f_i\}$, $i \in \{1, n\}$ corresponding to the n ALMs to track. For each frequency separately we construct an "evidence map" $I_i(\langle x, y \rangle, t)$ over the visual field corresponding to the probability that the ALM is at that pixel. Each hyper-transition contributes to all evidence maps, but with a different weight, so that we can integrate all information and do not commit to saying that a given event belongs to a frequency. For non-noisy data, an alternative approach that uses clustering of events works just as well [13].

A hyper-transition with interval Δ_k contributes to the evidence map of frequency f_i with a weight that is proportional to $p(\Delta_k \mid f_i)$; that is, the likelihood that a marker ALM with that frequency produces a hyper-transition of that interval. The distribution $p(\Delta_k \mid f_i)$ is found experimentally to be well approximated by a Gaussian, as seen in the data in Fig. III.2*b*:

$$p(\Delta_k \mid f_i) = \mathcal{N}\left(\frac{1}{\Delta_k} - f_i, \sigma^2\right). \tag{IV.1}$$

In our experimental setting, the standard deviation is approximately $\sigma = 30$ Hz. The evidence maps collect events within a time slice corresponding to an interval of $1/f_i$. Therefore, the value of the evidence map $I_i(\langle x, y \rangle, t)$ for a pixel x, y and at time t is given by the sum of the contributions of all events at the given pixel and in the interval $[t - 1/f_i, t]$:

$$I_i(\langle x, y \rangle, t) = \sum_{t_k \in \left[t - \frac{1}{f_i}, t\right] \wedge \langle x_k, y_k \rangle = \langle x, y \rangle} \mathcal{N}\left(\frac{1}{\Delta_k} - f_i, \sigma^2\right).$$

To increase robustness at the expense of latency, it is also possible to use multiples of $1/f_i$ as the time slice interval.

At the end of the time slice, the evidence map $I_i(\langle x, y \rangle, t)$ can be interpreted as the likelihood that the i-th ALM with frequency f_i is at position $\langle x, y \rangle$. In our experimental setting, this map is multimodal, with a strong peak at the true position of the marker, and lower peaks at the positions at the other markers, because each event contributes weakly, according to (IV.1), also to the evidence maps of the other frequencies.

We extract m local maxima, at least δ pixels from each other (in our experiments $m = 3$, $\delta = 15\,\text{px}$). The value of the evidence map at the local maxima is used as a weight w to be carried forward to the next stage. The detections generated in this way have a time t, coordinates $\langle x, y \rangle$ and the weight w_j^i:

$$\{\langle t,\, \langle x_j^i, y_j^i \rangle,\, w_j^i \rangle\},\ j \in \{1, \ldots, m\}.$$

E. Filtering and reconstruction

Once we have these detections, the method proceeds in a conventional way, as in any tracking problem, to achieve robustness to missed detections and false alarms.

First we use a particle filter to evolve particles for each frequency. Each particle has coordinates $\langle x, y \rangle$, a weight w (carried over from the last step), as well as an isotropic spatial uncertainty r, which starts at 1 px. The uncertainty grows using a motion model, which should be chosen according to how on how fast things are predicted to move on the visual field. We have computed that for the range of motions of a quadrotor, the maximum apparent motion is approximately 1 pixel/ms.

We have a particle filter for each frequency. The particles in each filter represent the posterior over the pose of one ALM. To look for a globally consistent solution, we choose the combination of particles from all filters with the highest combined weight such that no two markers can be too close to each other (in our experiments, $d = 15$ pixels). Assuming we have the position of the ALMs in image space, and we know the relative position of the markers in the world, we can reconstruct the pose of the object using established techniques for rigid reconstruction.

V. Experiments

The experiments consider the advantages of a DVS-based tracking solution with respect to a tracking solution based on a traditional CMOS camera. We compare the DVS-based ALM tracking with vision-based tracking using the PTAM algorithm, using the output of an OptiTrack system as the ground truth. The data show that the DVS-based tracking is able to deal with faster motions due to the minimal latency, but the precision of the reconstructed pose is limited by the low resolution of the sensor.

A. Hardware

1) Robot platform: We used the commercially available ARDrone 2.0 We attached four custom-built ALMs to the bottom of the platform (Fig. V.1a). Each LED was fixed facing downwards, one under each of the four rotors, so that the four were lying on a plane forming a square of 20cm side length. The USB connector available on the drone provided power to the microcontroller and ALMs. The drone has also a front-facing 720×980 CMOS camera that is used in these experiments, while the ground-facing camera is not used.

(a) ALMs configuration (b) Infrared markers

Figure V.1. The ARDrone 2.0 equipped with four ALMs (shown in *a*) tracked by the DVS, and reflective markers used by the OptiTrack (shown in *b*).

2) DVS: The DVS128 camera was used for the tests. This model is currently commercially available from INI labs. It has a resolution of 128×128 pixels. The lens attached gave the sensor a FOV of approximately 65°, giving a resolution of 0.5 pixels/°. For tracking the quadcopter, the DVS was installed on the floor facing upwards. Note that the relative motion between DVS and quadcopter would be the same if the ALMs were on the floor and the DVS on board.

3) OptiTrack: To measure the pose estimation accuracy we used a OptiTrack tracking system from NaturalPoint [14], which is a marker-based optical motion tracking system using active infrared light and reflective marker balls. Four markers have been applied to the drone (Fig. V.1b). Our lab setup comprised 10 cameras in a 6×8 m area; the cost of this system is approximately 20,000 CHF ($21,000). The sampling frequency used was 250 Hz. The accuracy is stated as ~ 1 mm by the manufacturer, but this seems an optimistic estimate based on our experience with the system.

4) Motion: The prototypical aggressive maneuver that we use is a "flip" of the quadcopter, i.e. a 360° roll. During the flip the frontal camera images are severely blurred (Fig. V.2).

5) Interference OptiTrack / DVS: We encountered an unexpected incompatibility between OptiTrack and DVS. The OptiTrack uses high-power infrared spotlights. In the OptiTrack's standard configuration, the spotlights are pulsed at a high frequency. This is of course invisible to normal sensors and to the human eye, but it was a spectacular interference for the DVS. Like most cameras, the DVS is most sensitive in the infrared spectrum and is much faster than the OptiTrack strobing frequency. This generated a buffer overflow on the DVS as the electronics could not handle the large number of events to be processed contemporaneously. Eventually we understood how to deactivate the strobing for all the cameras prior to recording. Still there was a slight residual interference by the infrared illumination from the OptiTrack, but it should have relatively little impact to the results of our experiments.

B. Methods

We compare three ways to track the pose of the quadcopter: 1) The output of our DVS-based ALM tracking method; 2) The OptiTrack output; 3) The output of a traditional feature-based tracker using the data from the conventional CMOS camera mounted front-facing on the drone. The image data was streamed to a computer

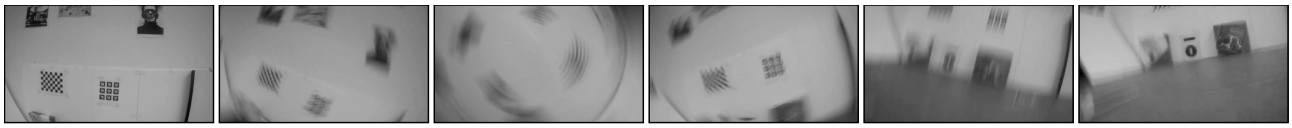


Figure V.2. Motion blur induced on CMOS image from flip motion.

via network interface, were the parallel tracking and mapping algorithm (PTAM) [15] was employed for pose estimation.

1) Data recording, synchronization, and alignment: Using this setup we did several recordings, in which we recorded the OptiTrack tracking data, using its native format, the image data using a ROS interface, as well as the raw event data from the DVS in the native format.

To synchronize the data from different sources we used a motion induced cue. We moved manually the drone up and down, generating an approximated sinusoid curve in the position data, which allowed easy manual matching of the sequences.

After adjusting for the delay, the data sets were brought to the same number of samples with a common time stamps. As our algorithm's output has a lower sampling rate than the OptiTrack (1 KHz vs 250 Hz), the OptiTrack data was resampled by linear interpolation.

As a final step, the time series were put in the same frame of reference. Given two sequences of points $\boldsymbol{x}_k, \boldsymbol{y}_k \in \mathbb{R}^3$, the rototranslation $\langle \boldsymbol{R}, \boldsymbol{t} \rangle \in \mathsf{SE}(3)$ that matches them can be found by solving the optimization problem

$$\min_{\langle \boldsymbol{R},\boldsymbol{t} \rangle \in \mathsf{SE}(3)} \sum_k \|\boldsymbol{x}_k - (\boldsymbol{R}\boldsymbol{y}_k + \boldsymbol{t})\|^2, \tag{V.1}$$

which is a classic Procrustes problem [16].

C. Results

We recorded data from 18 flips, of which only 6 were successful. During the recordings we met a number of unforeseen difficulties due to our modifications to the drone. Having attached the LEDs and microcontroller to the drone we found that it had become unstable during flight and hard to control due to the additional weight, so while it could hover normally, it did not have enough thrust to stabilize itself after a flip.

1) Tracking downtimes: During a flip, both the DVS and PTAM lose tracking: PTAM loses tracking while the image is blurred; the DVS loses track when the ALMs are not visible from the ground. The comparison of these "blackout times" gives a direct measurement of the latency of the two systems.

The length of a flip was measured by considering the roll data from the OptiTrack, taking the interval between the last measurement before the flip and the first measurement after the flip when the helicopter was in a level orientation to the floor.

Table II
TRACKING DOWNTIME INTERVALS AND THE FLIP DURATION.

DVS	0.35 ± 0.10 s
PTAM	0.80 ± 0.33 s
flip duration	0.56 ± 0.15 s

To measure the onset and offset of the blackout for the DVS, we considered the last sample before losing track (i.e. where the interval position samples were considerably higher than the mean sampling rate) and the first sample of reacquiring track (regaining a steady sample rate). The equivalent operation was performed on the PTAM data.

Table II shows the mean standard deviation of the different approaches. Our algorithm lost track during the average time of 0.35 seconds. PTAM lost track for a mean of 0.8 seconds, which is more than twice the time of the DVS and takes longer than the average duration of a flip. One can clearly see that the time where tracking is lost is much shorter with our approach in respect to PTAM. The results emphasize that the DVS is faster in recovering lost tracks than the PTAM approach due to not suffering from motion blur. As verified with our recordings, the downtimes of the DVS correspond to losing sight of the LED markers because of their emission angle. With a suitable configuration of either more markers or dynamic vision sensors, tracking could be maintained during the whole flip.

2) Accuracy of estimated pose: The statistics of the estimation error for DVS and PTAM, considering the OptiTrack as the ground truth, are summarized in Table III and shown in graphical form in Fig. V.3.

As for the translation, the DVS estimation error is roughly two times lower than PTAM (Fig. V.3*a*). Although the spread of outliers is higher in our approach compared to PTAM, the translation errors of the latter technique show a broader distribution around their median. Overall this proves that the DVS approach has higher accuracy with less spread, if we neglect the extreme tails of the distribution.

Fig. V.3*b*–*d* show the error distribution for roll, pitch and yaw respectively. The DVS performs worse in roll and pitch compared to yaw. This was to be expected, because of the position of the ALMs. As roll and pitch play a minor roll in quadrotor pose estimation these can be neglected for finding the drone's orientation. The DVS performs slightly worse than PTAM with a mean error of 6° and a deviation of 15° (Table III). This is explained by the much lower resolution of the DVS (128×128 pixels)

compared to the CMOS camera used by PTAM (720×980 pixels).

Table III
ESTIMATION ERROR OF DVS AND PTAM COMPARED TO OPTITRACK

	(a) Translation	(b) Roll	(c) Pitch	(d) Yaw
DVS	8.9 $\pm$ 12.6 cm	19 $\pm$ 27°	17 $\pm$ 18°	6 $\pm$ 15°
PTAM	19.0 $\pm$ 12.4 cm	7 $\pm$ 22°	5 $\pm$ 11°	3 $\pm$ 10°

VI. CONCLUSIONS

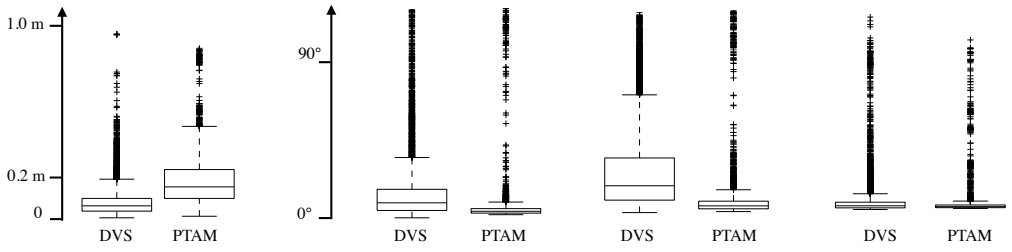
Fast robots need fast sensors. A *dynamic vision sensor* (DVS) returns changes in the visual field with a latency of a few microseconds. This technology is the most promising candidate for enabling highly aggressive autonomous maneuvers for flying robots. The current prototypes suffer a few limitations, such as a relatively low resolution, which are being worked upon. In the mean time, the sensing pipeline must be completely redesigned to take advantage of the low latency.

This paper has presented the first pose tracking application using DVS data. We have shown that the DVS can detect Active LEDs Markers (ALMs) and disambiguate their identity if different blinking frequencies are used. The algorithm that the we developed uses a Bayesian framework, in which we accumulate evidence of every single event into "evidence maps" that are tuned to a particular frequency. The temporal interval can be tuned and it is a tradeoff between latency and precision. In our experimental conditions it was possible to have a latency of only 1 ms. After detection, we used a particle filter and a multi-hypothesis tracker.

We have evaluated the use of this technology for tracking the motion of a quadrotor during an aggressive maneuver. Experiments show that the DVS is able to reacquire stable tracking with negligible delay as soon as the LEDs are visible again, without suffering from motion blur, which limits the traditional CMOS-based conventional feature tracking solution. However, the precision in reconstructing the pose is limited because of the low sensor resolution. Future work involving the hardware include improving the ALMs by increasing their power and their angular emittance field, as we have found these to be the main limitations.

In conclusion, DVS-based ALM tracking promises to be a feasible technology that can be used for fast tracking in robotics.

Acknowledgements: We gratefully acknowledge the contribution of Christian Saner for helping with the ARDrone software development, the assistance of Yves Albers-Schönberg as our pilot during test flights and in recording ROS logs, and the assistance of Matia Pizzoli in the CMOS-based tracking experiments.



(a) Translation error (b) Roll error (c) Pitch error (d) Yaw error

Figure V.3. Distributions of the errors of DVS/PTAM in reference to the OptiTrack measurements. The data is synthesized in Table III.

REFERENCES

[1] S. Weiss, D. Scaramuzza, and R. Siegwart. "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments". In: *J. Field Robot.* 28.6 (Nov. 2011), pp. 854–874. ISSN: 1556-4959. DOI: 10.1002/rob.20412. URL: http://dx.doi.org/10.1002/rob.20412.

[2] S. Shen, N. Michael, and V. Kumar. "Autonomous multi-floor indoor navigation with a computationally constrained MAV". In: *ICRA*. 2011, pp. 20–25.

[3] S. Lupashin and R. D'Andrea. "Adaptive fast open-loop maneuvers for quadrocopters". In: *Auton. Robots* 33.1-2 (2012), pp. 89–102.

[4] T. Delbruck and P. Lichtsteiner. "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system". In: *IEEE International Symposium on Circuits and Systems*. 2007, pp. 845–848. DOI: 10.1109/ISCAS.2007.378038.

[5] J Conradt, M. Cook, R Berner, P Lichtsteiner, R. J. Douglas, and T Delbruck. "A pencil balancing robot using a pair of AER dynamic vision sensors". In: (2009). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5117867.

[6] M. Boerlin, T. Delbruck, and K. Eng. "Getting to know your neighbors: unsupervised learning of topography from real-world, event-based input". In: *Neural computation* 21.1 (2009). DOI: 10.1162/neco.2009.06-07-554.

[7] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128 $\times$ 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (2008). DOI: 10.1109/JSSC.2007.914337.

[8] R Etienne-Cummings. "Intelligent robot vision sensors in VLSI". In: *Autonomous Robots* 7.3 (1999). DOI: 10.1023/A:1008968319725.

[9] M Oster, Y Wang, R Douglas, and S. C. Liu. "Quantification of a spike-based winner-take-all VLSI network". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55.10 (2008). DOI: 10.1109/TCSI.2008.923430.

[10] *The JAER library*. URL: http://sourceforge.net/apps/trac/jaer.

[11] *Thesycon driver*. URL: http://www.thesycon.de/eng/home.shtml.

[12] *Bronze Board*. URL: http://www.ini.uzh.ch/tobi/wiki/doku.php?id=dig:uc.

[13] M. Hofstetter. "Temporal Pattern-Based Active Marker Identification and Tracking Using a Dynamic Vision Sensor". Master thesis. Institue of Neuroinformatics, ETH Zurich, 2012.

[14] URL: http://www.naturalpoint.com/optitrack/.

[15] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *IEEE and ACM International Symposium on Mixed and Augmented Reality* (Nov. 2007).

[16] J. C. Gower and G. B. Dijksterhuis. *Procrustes problems*. Vol. 30. Oxford Statistical Science Series. Oxford, UK: Oxford University Press, 2004. ISBN: 978-0-19-851058-1.