University of Zurich
Department of Informatics

ai lab
Robotics and Perception Group

Jonas Strubel

Quadrotor tracking and pose estimation using a dynamic vision sensor

Master Project

Robotics and Perception Lab
University of Zurich

Supervision

Prof. Dr. Davide Scaramuzza
Dr. Andrea Censi
Christian Brändli

February 2013

Contents

Abstract **ii**

1 Introduction **1**

2 Approach **3**
- 2.1 Considerations and Challenges 3
- 2.2 Preparation 3
  - 2.2.1 C++ interface to the DVS 3
  - 2.2.2 LED markers 4
- 2.3 The tracking algorithm 4
  - 2.3.1 Frequency filter 4
  - 2.3.2 Frequency period estimation 6
  - 2.3.3 Period accumulation 7
- 2.4 LED position estimation 8
  - 2.4.1 Particle filter 8
  - 2.4.2 Extracting LED positions 8
  - 2.4.3 Pose estimation 9
  - 2.4.4 Discussion 9

3 Results **10**
- 3.1 Experiments 10
  - 3.1.1 Setup 11
- 3.2 Evaluation 13
  - 3.2.1 Tracking speed 13
  - 3.2.2 Pose estimation 14
- 3.3 Discussion 16

4 Conclusion **21**

Abstract

Compress the introduction in a few key sentences. No more than half a page.

Chapter 1

Introduction

A crucial property of micro aerial vehicle (MAV) navigation and control is the estimation of their position towards the environment. There exist several methods to achieve this, a popular one being the use of cameras. The cameras can be either placed on the MAV itself or statically in the environment for external tracking. In order to estimate the pose specific cues in the camera image must be located. This works well as long as there are significant features available. Since conventional camera sensors generally provide not more than 30 frames per second, fast motion blurs the camera image and thus hinders feature detection. Therefore, during an aggressive maneuver, tracking is usually lost and needs to be reacquired after the camera data are clear again. To re-establish a lost track already know cues need to be found, inducing a latency for tracking. A possible solution to this drawback is the use of cameras with higher frame rates. This comes at an expense though, not only in terms of data processing but also in price. Therefore we wanted to find a solution which is not only computationally cheaper, while still providing high speed, but also comes at a reasonable price. The Institute of Neuroinformatics at the University of Zurich and ETH has developed a dynamic vision sensor(DVS) which should provide the solution. Inspired by the human retina, each pixel fires asynchronously and only if there is change in illumination on a specific pixel. Thus, the data from the DVS are events, where an event stores the following values:

- Position {x, y}
- Polarity {on | off}
- Time stamp {t}

Therefore events depict the time and place of a single impulse rather than pixels in a frame. The change of illumination is thereby binary, i.e. only the direction of change is advertised. The time stamp is generated by DVS internals and has microsecond resolution. This provides high sampling rates, while maintaining a relatively low data output. As the processing is done asynchronously the sampling rate is not fixed but rather dependent on the amount of overall activity on the pixels. There are several reasons for that, for example the limit set by the serial time-stamping mechanism, as well as the data output via USB 2.0. Nevertheless, the camera provides up to 1 million events per second. Employing special circuitry comes at an expense though as the camera has a comparably low resolution of 128x128 pixels. For a deeper insight into the camera technology please refer to [4]. In order to facilitate tracking, given the high temporal resolution of the DVS, we decided to use active markers in the form of differently pulsed LEDs with the DVS. The feasibility and high robustness of this approach had already been demonstrated by [2]. The LEDs were attached to a quadrotor helicopter, which would then be tracked by the DVS. For the marker tracking we developed an algorithm able to track several LEDs pulsed with different frequencies. For estimating the pose we used an already existing library. In the following part the challenges an preparations are discussed. After that a detailed description of the algorithm will be given followed by the experiments and evaluation. Finally a conclusion summarizes our findings as well as discusses possible future improvements and projects.

Chapter 2

Approach

2.1 Considerations and Challenges

First of all, working with special hardware like the DVS requires a different approach to handling and processing the data. Using normal cameras, naturally conventional vision algorithms are based on the analysis of frames. In order to make use of the asynchronous behaviour an image-based approach is not feasible for maintaining the high temporal resolution, i.e. the algorithm had to make use of the event-based behaviour of the DVS. Another property to keep in mind with the DVS is the fact that data is only provided if there is a change in illumination, e.g. motion. This means that a static input will not generate significant output. A limitation to keep in mind is the low spatial resolution. This makes it infeasible to detect smaller features which high resolution cameras are able to. Thus the low resolution also introduces a higher uncertainty towards visual cues in comparison to conventional cameras. As explained in the introduction, the sampling frequency i.e. the temporal resolution of the DVS varies. Therefore it was necessary to measure what frequencies could be handled by the DVS especially in respect to our drone flight scenario.

2.2 Preparation

In this paragraph the necessary preparations before the implementation of the tracking algorithm are discussed. These comprise of writing a driver interface in C++ to the DVS as well as building a set of active markers.

2.2.1 C++ interface to the DVS

The framework provided with the DVS called jAER[1] is completely written in Java as well as the tracking algorithm developed by [2]. Since we wanted to implement our approach in C++ due to performance and portability considerations (e.g. to embedded systems) it was necessary to write a driver interface to the DVS first. Thesycon[2], the provider of the USB device driver, already offers a framework to communicate with their driver. This was employed to pull the

[1]http://sourceforge.net/apps/trac/jaer
[2]http://www.thesycon.de/eng/home.shtml

event data from the DVS. Similar to jAER, the events are wrapped up in packets of several events. The packet size depends on the junk of data which arrives at once from the USB . This not only makes sense as the data arrive in chunks of up to several hundred events from the USB buffer; packet-based processing is furthermore advantageous for certain operations, e.g. if they are computationally costly and single events do not provide enough relevant information to be processed individually.

2.2.2 LED markers

In order to build a set of blinking LEDs with different frequencies we used the bronze board[3] from INI which is based on an AVR32. A USB interface provides the means for simple programming and communication with the microcontroller. Although a less powerful microprocessor would have been sufficient, a high number of independent Pulse-Width-Modulation (PWM) drivers was an important property in order to drive several LEDs with different frequencies. The bronze board has 7 PWM drivers, from which 5 are free to use. An already existing C code example was taken and adapted to our needs. Additionally, a python script provided the means of remotely adjusting the PWM via USB. The choice of LEDs was important, since they should be well visible. The ideal LED for our project would provide high illuminance and a wide emission angle. Since the DVS is very sensitive to the infra-red spectrum, we decided to go for infra-red LEDs[type hint]. We decided to attach four LEDs to our quadrotor which is the minimum number to estimate the pose in 3D space.

2.3 The tracking algorithm

The following paragraphs describe the separate steps of the tracking algorithm we developed. Figure 2.1 provides an overview over the stages described in the following sections. The approach designed by [2] works on a more general level assuming the blinking pattern to be previously unknown. As we wanted to design our algorithm for a specific task, namely tracking a helicopter, these constraints gave us the opportunity to simplify the algorithm. In our setup we would be tracking specific LEDs attached to a helicopter, of which we already knew the blinking pattern, as well as the position towards each other. While in [2] the algorithm starts by finding high-activity clusters, denoting the position of a blinking LED, we decided to take a different approach. Fixing different frequencies to our LEDs allowed us to first filter frequency space to get rid of background noise and extract the relevant information. The signals we used for this resemble a square wave with 50% duty rate. The following section describes the filtering process, as well as the choice of frequencies.

2.3.1 Frequency filter

There are two main challenges to consider in frequency analysis:

- The correct measurement of frequencies
- The influence of background noise in frequency measurement

[3]http://www.ini.uzh.ch/ tobi/wiki/doku.php?id=dig:uc

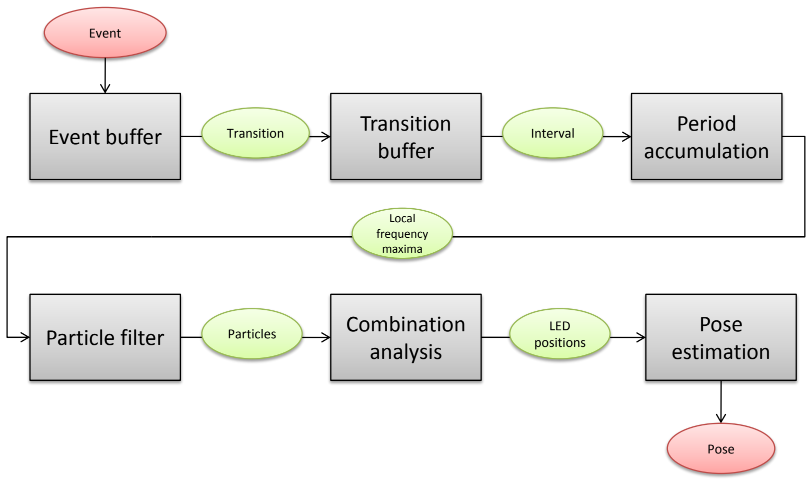

Figure 2.1: Sequential diagram of the processing stages. The boxes show the different processing stages while the circles denote the different data containers.

Having static LEDs in the camera's "'image"' the straight-forward approach of measuring the inter-spike-intervals[4] (ISI) would be evident as changes of LED state(from on to off or vice versa) should generate consecutive events of different polarity and thus yield to the signal period. This approach should also work for moving LEDs as long as the frequencies are high and the LED velocity is low enough, so that consecutive state changes overlap to generate events on the same pixels. Since this has shown to be true for the maximum possible velocity of the quadrotor towards the camera, we went for this approach. In order to address the influence of background noise we needed to find out what ISIs were usually generated by it. Therefore, we recorded data over several scenarios with the DVS by moving it over our set of pulsed LEDs fixed to the ground. This included different distances and speeds, as well as varying backgrounds. Thereby, background containing many edges would generate a considerably higher amount of events than a uniform backgrounds (e.g. a white wall). Evaluating the different data on a frequency histogram we found that most of the movement induced background noise generates frequencies up to around 500 Hz (2.2). Thus, to make our LEDs well distinguishable they needed to be pulsed with frequencies above this limit. Additionally to the lower boundary an upper limit also needed to be considered as the sampling accuracy of the DVS drops with higher frequencies [2]. Furthermore, the spacing between the frequencies was also important in order to prevent inaccurate measurement to

[4]A term often used in neuroscience to describe the time interval between two consecutive spikes on the same neuron, here used for a pixel.

deliver false detections and thus keep our markers well distinguishable. Last but not least, the frequencies should not be multiples of each other to ensure robustness towards occasional data loss (i.e. if a state change of the LED is missed the period will not be confused with the one of another LED). Experiments showed frequencies of 740, 1030, 1320 and 1610 Hz to work well.

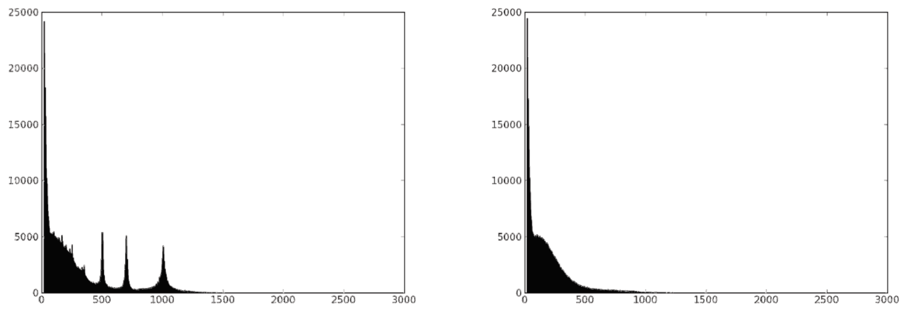

Figure 2.2: Frequency histograms for a DVS recording with high motion. Left: Includes three LEDs pulsed at 500, 700 and 1000 Hz. Right: Without LEDs.

2.3.2 Frequency period estimation

Following the approach of using ISIs for period estimation, where an event denotes a rising or falling edge of the signal, these needed to be measured first for each pixel. As we found, two consecutive events on a pixel excited by a blinking LED did not always have different polarity. Instead, several successive events of the same polarity are generated. This can be related to the high sensitivity of the DVS as well as the non-uniform illumination coming from an LED over time. Therefore, it was necessary to observe consecutive events and note a state transition, whenever they were of different polarity. This provided us with two kinds of transitions, namely from positive to negative polarity or vice versa. Two consecutive transitions of the same type would then yield the signal period. In order to keep track of the latest events on a per pixel level, we use a 2D array resembling the pixel space of the DVS, where only the latest event for each pixel is stored. Before inserting a new event into the array, its type is compared to the preceding event. If their polarities differ a transition is generated. The values contained in a transition are similar to events:

- Position {x,y}
- Type {on-to-off | off-to-on}
- Time stamp {t}

Like the previous step each transition is mapped onto a 2D array with pixel space resolution. To keep track of each transition type two 2D-arrays are used to store the latest transitions. Following our previous assumption, before insertion of a new transition the time interval to its predecessor is now measured. This provides a period and thus a certain frequency estimate. An interval is stored with:

  - Position {x,y}
  - Time stamp {t}

- Time interval $\{\delta t\}$

2.3.3 Period accumulation

In order to find the relevant frequencies each measurement is weighted with a Gaussian according to its distance to each of the frequencies we want to find. This provides us with different likelihoods (according to the magnitude of weight) of having found a relevant frequency. The weights are stored in a weight map, again a 2D array of pixel-space resolution. Figure 2.3 shows the raw DVS output and the aggregated weight map over all frequencies. In order to increase the robustness and to avoid noise induced weights to influence our frequency filter it was important to gather several weights per pixel to support the hypothesis of having found a pulsed signal. Since we are interested in extracting four distinct signals a weight map for each needs to be maintained. Each pixel in the map gathers the sum of weights measured during a certain time. An interval of 10 ms during weights are gathered provided feasible results. The most likely position of a certain LED is then estimated by taking the pixels with the most support into account, i.e. the ones with the highest weight. Therefore a set of local maxima in the weight map needs to be maintained. We also introduced a minimum distance between maxima to allow only one maximum per LED to be found. A distance of 15 pixels was chosen, as this is larger than the usual area covered by a single LED. The number of local maxima to maintain was set to three, as experiments showed additional maxima to have a comparably much lower significance. For the Gaussian weighting we allowed a standard deviation of 30 Hz.

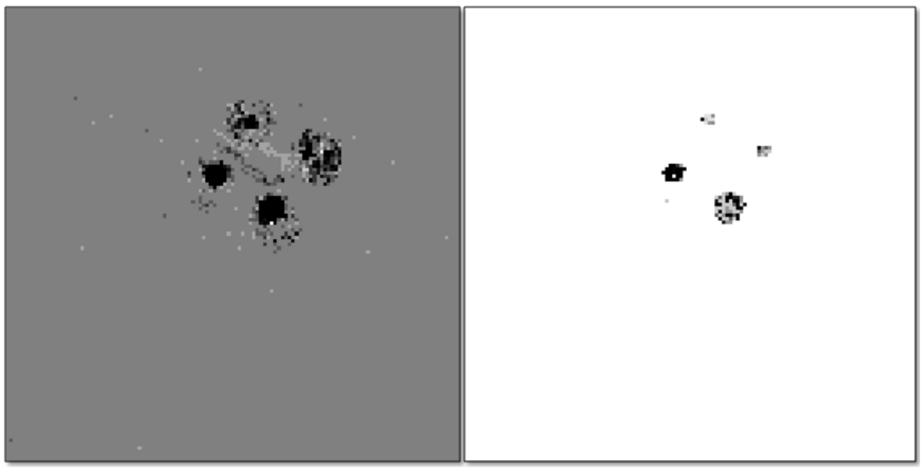

Figure 2.3: Left: The raw DVS output. Right: An aggregation of all weight maps. The darkness of the pixels indicates the magnitude of weight.

2.4 LED position estimation

So far the local maxima provide us different hypothesis of the location of each LED. This gives us only a snapshot in time of the current most likely locations though. In order to increase robustness by observing the positions over time as well as taking motion of the quadcopter into account we applied a particle filter. This technique is widely used in robotics for state estimation. The following paragraph describes our application this approach.

2.4.1 Particle filter

So far each evaluation of local maxima provides three hypotheses denoting the possible marker positions. Each hypothesis is now used to update a particle filter. For each signal we want to find, there is a particle filter maintained with a fixed set of particles. Such a particle stores the following values:

- Position {x,y}
- Time stamp {t}

- Uncertainty $\{\sigma\}Weight\{w\}$

The position in pixel-space thereby has sub-pixel resolution. The uncertainty of the particle denotes the possible radius in pixels in which the LED can move in time representing our motion model. Thus the uncertainty of a particle grows in time, according to the maximum possible speed an LED can move in image space. If the uncertainty reaches a certain threshold, i.e. the uncertainty is too high, the particle expires and is deleted. The weight denotes the amount of support a particle has and the time stamp marks the time of its creation.

Initially, the particle filter is empty. For every new hypothesis a candidate particle is created. Thereby, the position and weight are inherited from the local weight maxima while the uncertainty is set to a default value of 2 pixels. The latest incoming event marks the time of particle creation. Each candidate can now become a new particle in the filter or is merged with an existing one. Candidates are thus compared to existing particles to see if their position falls into the uncertainty radius of an existing particle. In this case, and under the assumption of our motion model, this would mean that the candidate indicates a valid consecutive position in time of an LED. If no existing particle can be updated, the candidate is inserted into the filter while replacing the oldest particle if the filter is full. During merging the uncertainties decide upon the contribution of the two particles to the new one created. The new position is linearly interpolated. Similarly, the new weight is a summation of particle values weighted by their uncertainty. Finally, the new uncertainty is calculated by a multiplication of Gaussians.

2.4.2 Extracting LED positions

In this final step we need to decide on the definitive positions of the four LEDs. So far we have a particle filter for each LED with a number of particles denoting the most likely positions for the LEDs. A straight-forward approach would choose the particle with the highest weight from each filter, i.e. the particle with the highest support. Unfortunately though, we found that the frequencies we used were still close enough to be ambiguous for frequency estimation. Thus, apart from random noise, close-by frequencies tended to have particles of different frequency on the same LED position. This required then, to find the best valid combination of particles where they all had distinct positions and thus should resemble our marker positions. The significance of a valid combination was thereby found by multiplying the particle weights. In order to solve this combinational problem a recursive function is used to traverse all the different combinations. Since this combinatorial problem implies many recursions, becoming computationally costly, we introduced some heuristics to reduce the number of recursions. First, each particle filter was sorted in descending order according to particle weights. This should favour traversing the most significant combinations early. In addition, since we were only interested in a limited number of most likely combinations, after finding enough hypotheses a branch would only be expanded if its score was bigger than the least significant score of the hypotheses found so far. For choosing the right combination, we took the straight forward approach of selecting the one with the highest score.

2.4.3 Pose estimation

Given the marker positions, we were now able to estimate the relative pose of the helicopter to the camera. We used an existing algorithm from OpenCV[5] for that purpose. Given the feature points in the objects reference frame, the correspondent points in image space, as well as the intrinsic camera parameters and its distortion values, the algorithm computes a rotation and translation of the object towards the camera reference frame.

2.4.4 Discussion

As mentioned in section 2.4.2, the inaccuracy of frequency estimation caused several frequencies detected on the ISIs generated by the same LED. It can therefore also happen that positions jump for a short time. Although this happens only on very few occasions further measures need to be taken to improve robustness. Another problem in the approach is the fact that the local maxima, which come from a single pixel, are taken to update a particle. Since an LED usually produces a blob comprising several pixels on the camera, maxima tend to jump between different pixels of this blob, which gives the particles and thus also the final LED positions an oscillating behavior.

[5]http://opencv.org/

Chapter 3

Results

3.1 Experiments

As already mentioned in the beginning we were interested in two qualities of our system. Most importantly, we wanted to find out what advantages the speed gain with the DVS would provide for quadcopter tracking. An already proven advantage is the opportunity to track active markers. This is beneficial for searching trackable features as they are easier to identify and thus faster to find. In order to test these performance properties of our algorithm a suitable experimental setup needed to be designed. To measure how fast our system would perform under conditions where convetional cameras tend to fail, we planned to test it under real conditions by flying aggressive maneuvers with a quadcopter. Furthermore, since the DVS has a much lower resolution than today's standard CMOS cameras, the accuracy at which pose can be estimated with the DVS needed to be evaluated as well. [6] has developed a framework for controlling the ARDrone including the ability to let the drone execute a flip, i.e. a 360roll. We decided to use this flip as our aggressive maneuver. We would then measure the time interval during which tracking was lost.

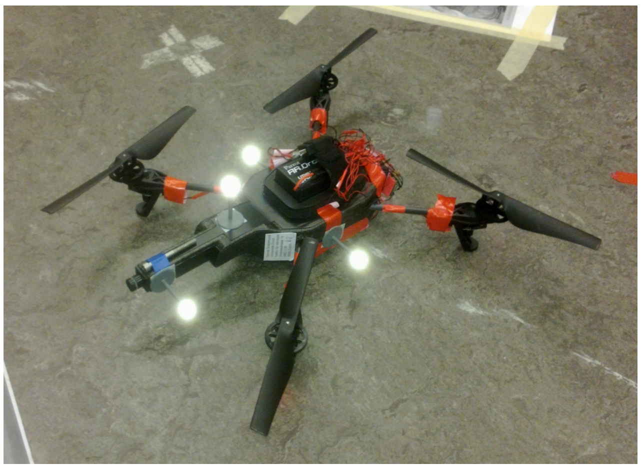

Figure 3.1: The ARDrone 2.0 equipped with reflecting markers and LEDs.

3.1.1 Setup

For our setup we used the commercially available ARDrone 2.0 (Figure 3.1) on which we attached our active markers as well as the microcontroller. Each LED was fixed facing downwards (if one assumes the drone to be in steady flight mode), one under each of the four rotors (Figure 3.2). All four LEDs where lying on a plane forming a square with 20cm side length. The USB connector available on the drone provided power to the microcontroller and LEDs. For tracking the quadcopter, the DVS was installed on the floor facing upwards (Figure 3.3).

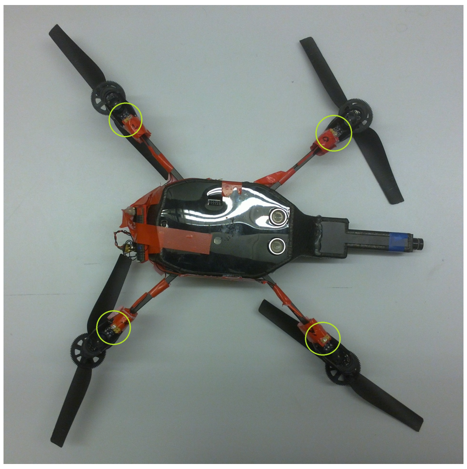

Figure 3.2: The underbelly of the quadcopter. The circles indicate the LED positions.

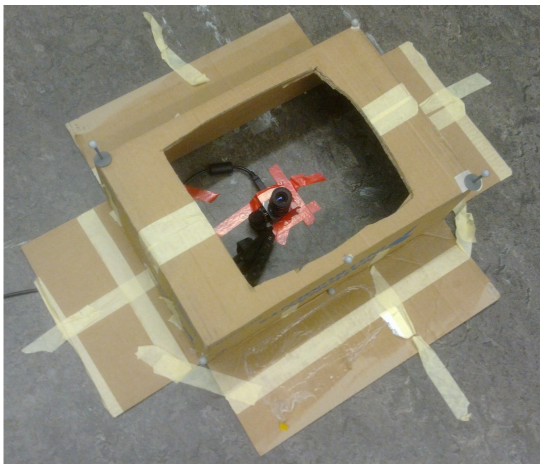

Figure 3.3: DVS setup for data recording.

Tracking speed comparison

In order to be able to compare the performance of the DVS to a conventional camera, we used the onboard front-looking camera on the quadcopter. The image data was streamed to a computer via network interface, were the parallel tracking and mapping algorithm (PTAM)[3] was employed for pose estimation.

Pose estimation accuracy

To measure the pose estimation accuracy we needed a reference. For this task we used the OptiTrack from NaturalPoint[1] which is a marker-based optical motion tracking system. It offers millimeter accuracy and was thus a very good approximation to the real positon. Four markers have been applied to the drone in order to make it trackable with the OptiTrack.

Data recording

Using this setup we did several recordings, where the position data from the OptiTrack, the image data and pose from the drone's onboard camera, as well as the raw event data from the DVS were recorded. We used the robot operation system[2] (ROS) for this purpose. As the data came from different systems we needed a way to synchronize the measurements afterwards. We decided to use a motion induced cue for synchronization by moving the drone up and down by hand, generating a sinusoid curve in the position data. This was later used for manual time synchronization of the recordings. Each recording then started with inducing the cue followed by the drone being remotely piloted to fly over the DVS camera where we would trigger a sideway flip.

Difficulties

During the recordings we met a number of unforeseen difficulties. Having attached the LEDs and microcontroller to the drone we found that it had become unstable during flight and hard to control remotely. Although the drone's motors are powerful enough to pull it upwards it had obviously not been designed with additional payload in mind. Another problem of the additional payload was that the drone was not able to output enough power to keep a stable height after the flip. Instead it usually came crashing down onto the floor. This meant that a stable reacquisition of the localization with the drone's internal camera had a higher latency than expected. The optical tracking system introduced another challenge into the experiments, since it uses reflecting markers for tracking which illuminated by high-power infrared spotlights. In the OptiTracks standard setup, the spotlights are pulsed. The DVS is very sensitive to the infrared spectrum and, as we found, the strobing generated a buffer overflow on the DVS as it could not the high amount of events arriving at the same time. Therefore it was paramount to deactivate the strobing for all the cameras prior to recording. Also, the infrared illumination from the OptiTrack seemed to impede the sensitivity towards our LEDs, although this barely influenced our experiment setup.

[1]http://www.naturalpoint.com/optitrack/

[2]http://www.ros.org

3.2 Evaluation

Due to the above difficulties out of 18 executed flips only 6 had feasible data. In the following two paragraphs the measurements of tracking downtime, as well as the pose estimation accuracy are discussed. Before evaluation the data from the DVS had yet to be passed through our algorithm.

3.2.1 Tracking speed

To measure how fast the different systems are able to reacquire tracking we first synchronized the recordings by hand. As during the recordings the DVS lost track of the LEDs on several occasions, due to them being out of view, the interval during which a flip happened, as well as the time measurements needed to be taken manually. The time for 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 where the helicopter was in a level orientation to the floor. To measure the onset and offset of losing tracking on the DVS, 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 same was done for the PTAM data. Unfortunately, as previously mentioned, the helicopter usually crashed into the floor after doing a flip. Therefore there often was no stable position data for several seconds. Nevertheless, few single samples with reasonable positioning were recorded before crashing which were thus taken as a reference. Figure 3.4 shows a statistical comparison of the time intervals in which the different approaches lost tracking compared to the duration of flips. The red bar indicates the median while the blue box marks the first and third quartile. The whiskers extend to a range of 1.5 times the range between the first and third quartiles. All data points outside this range are considered outliers and are marked with a red plus. Table 3.2.1 shows the mean standard deviation of the different approaches. Our algorithm lost track during the average time of 0.35 seconds. PTAM respectively 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. As figure 3.5 further illustrates, the downtime for the DVS stays inside the interval of the flip duration. The results emphasize that the DVS is faster in recovering lost tracks than the PTAM approach. As verified with our recordings, the downtimes of the DVS correspond to losing sight of the LED markers because of their emission angle. In contrary to PTAM tracking on the DVS is lost due to the marker configuration rather than fast motion. We reckon that with a suitable configuration of either more markers or dynamic vision sensors, tracking could be maintained during the whole flip. PTAM shows to lose track for a longer duration than the flip takes. In contrary to the DVS the the camera of the drone loses sight of its tracked features due to blurring in the camera image and thus takes a longer time to recover.

	$\overline{x}$ $[s]$	σ $[s]$
DVS	0.35	0.10
PTAM	0.80	0.33
Flip	0.56	0.15

Table 3.1: Mean and standard deviation of tracking downtime intervals and the flip duration.

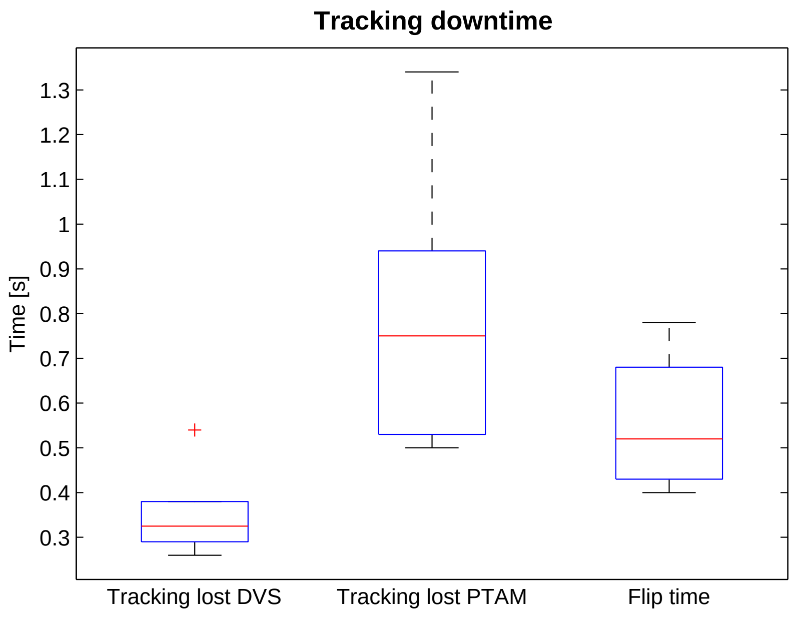

Figure 3.4: Statistical plot of measure time interval. The boxplots show the time interval in which tracking is lost for the our algorithm and PTAM compared to the duration of a flip.

3.2.2 Pose estimation

In order to measure the accuracy of pose estimation from the DVS the OptiTrack data was used as a reference. Apart from synchronizing the measurements in time we also needed to align the different reference frames from the DVS and OptiTrack system first. This was achieved using equation 3.1 to find the correct transformation $\{R,t\}$.

$$\sigma = \min_{R,t} \sum_i (S_i - (RT_i + t))^2 \text{ with } S, T \in \mathbb{R}^3. \tag{3.1}$$

Before doing so the data sets were first brought to the same number of samples with a common time stamps. As the DVS has a lower sampling rate than the OptiTrack the OptiTrack data was resampled by linearly interpolating the translation and rotation. The transformation gained from the least-squares al-

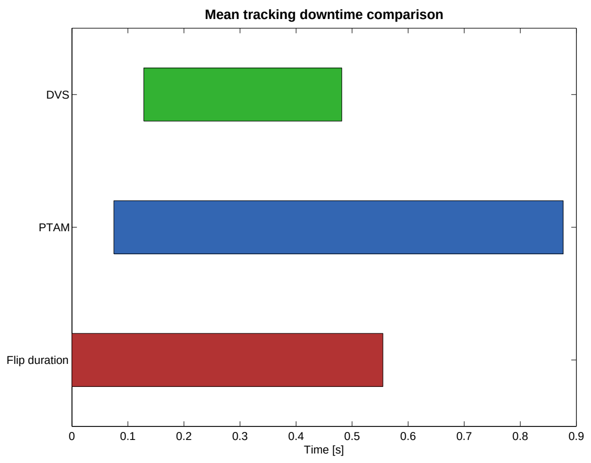

Figure 3.5: Comparison of the mean tracking downtime intervals. The mean time intervals of both algorithms are compared against the mean flip time on a time line.

	$\overline{x}$ $[cm]$	σ $[cm]$
DVS	8.9	12.6
PTAM	19	12.4

Table 3.2: Mean and standard deviation of translation error of the DVS and PTAM.

gorithm was then used to align the OptiTrack coordinate system to the one from the DVS. Last, the absolute errors were taken from the translation data. The rotation data, expressed in roll pitch and yaw, was again fitted from the OptiTrack to the DVS frame according to the least-square error. The same analysis was done on the PTAM data in order to compare the two approaches in their accuracy. Table 3.2.2 shows the mean and standard error for the translation in both approaches. The DVS' average error is roughly two times lower than PTAM. Figure 3.6 shows the statistical distribution of the pose errors (for details on the boxplot, please refer to 3.2.1). 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 the DVS approach to have a higher accuracy with less spread if neglecting the outliers. Figures 3.7,3.8 and 3.9 display the error distribution for roll, pitch and yaw respectively. The DVS performs worse in roll and pitch compared to yaw. This was to expect, due to the oscillating LED position as described in section 2.4.4, position estimations in depth have less accuracy. As roll and pitch play a minor roll in qaudrotor pose estimation these can be neglected for finding the drone's orientaion. Table 3.2.2 demonstrates that the DVS performs slightly worse than PTAM with a mean error of 6 degrees and a deviation of 15 degrees.

	$\overline{x}$ [°]	σ [°]
DVS	19	27
PTAM	7	22

Table 3.3: Mean and standard deviation of roll error of the DVS and PTAM.

	$\overline{x}$ [°]	σ [°]
DVS	17	18
PTAM	5	11

Table 3.4: Mean and standard deviation of pitch error of the DVS and PTAM.

3.3 Discussion

Overall the pose estimation for the DVS provides good results compared to PTAM. While the translation estimation with the DVS has a clear advantage, the rotation error is less accurate. The oscillating position information of LEDs seems to have a higher influence on the rotation estimation compared to translation. Improving the algorithm to provide more stable marker tracking is thus an important starting point to improve the overall accuracy.

	$\overline{x}$ [°]	σ [°]
DVS	6	15
PTAM	3	10

Table 3.5: Mean and standard deviation of yaw error of the DVS and PTAM.

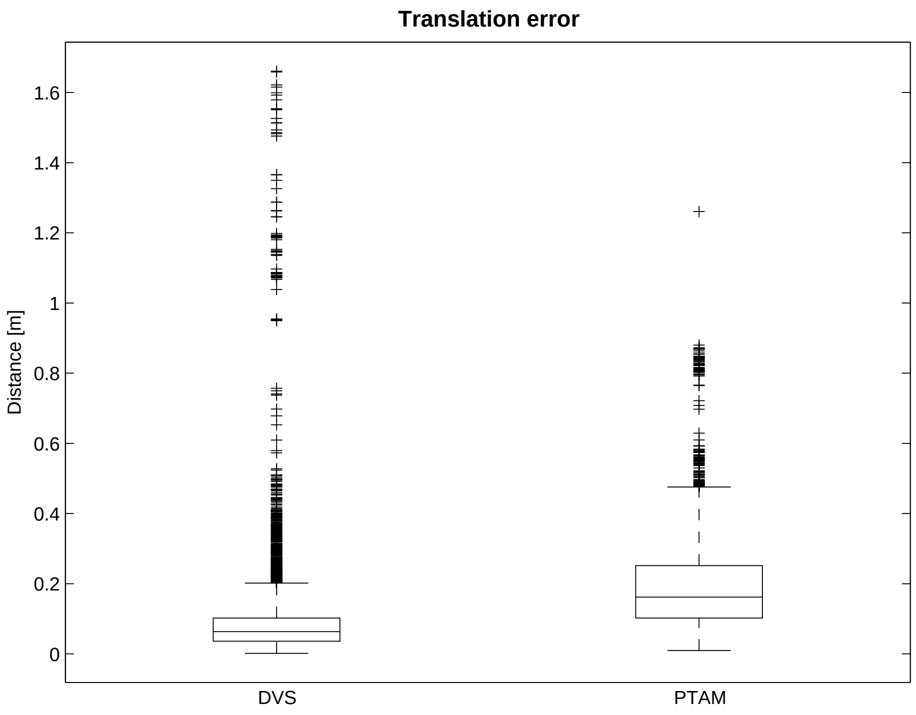

Figure 3.6: Statistical plot of the pose estimation error of the DVS in reference to the OptiTrack measurements.

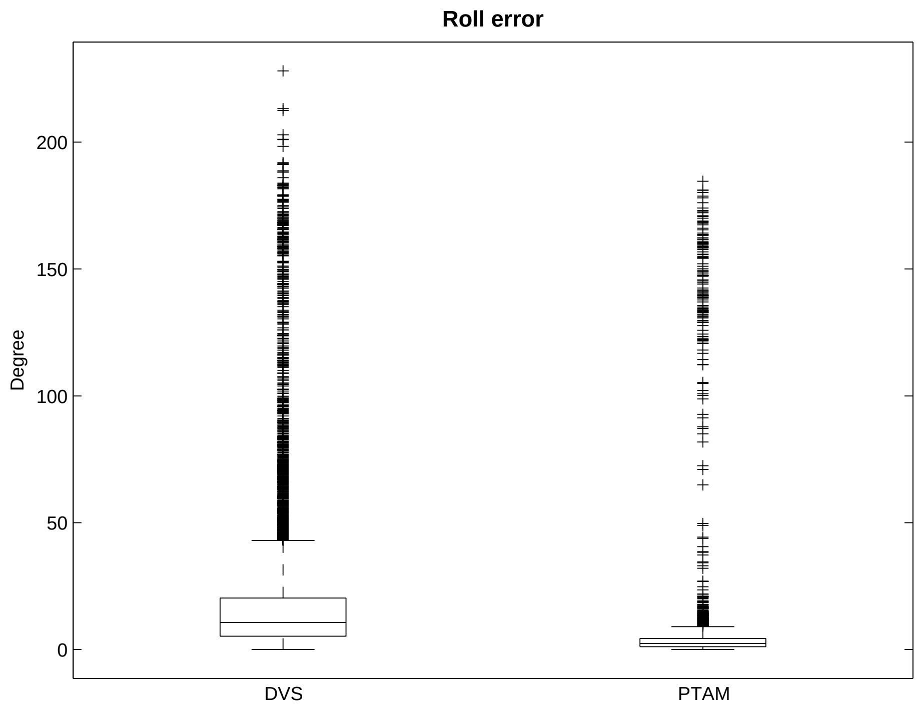

Figure 3.7: Statistical plot of the rotation error of the DVS in reference to the OptiTrack measurements.

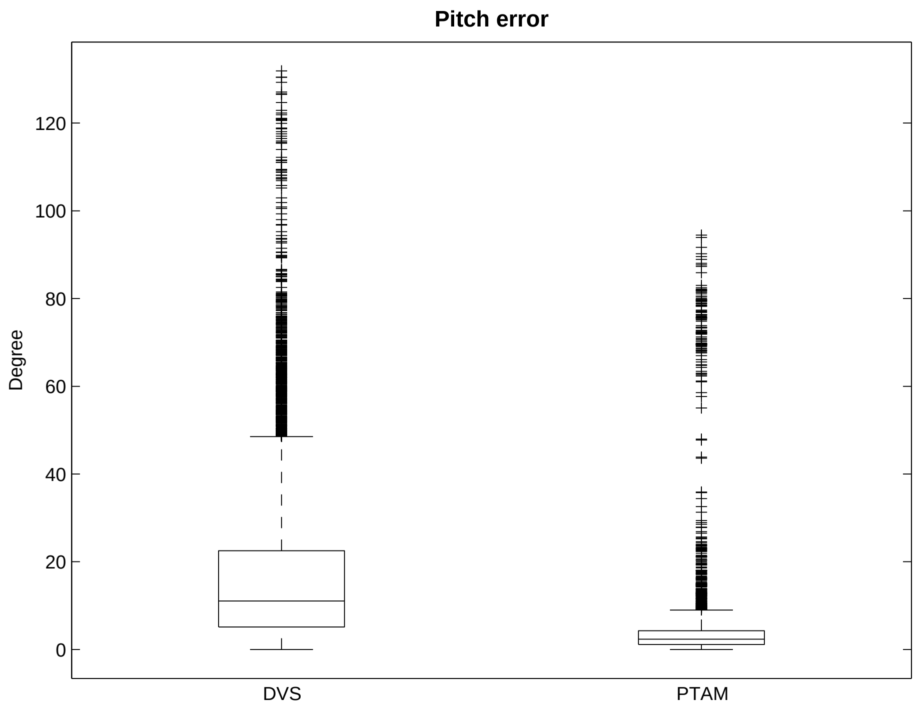

Figure 3.8: Statistical plot of the pose estimation error of PTAM in reference to the OptiTrack measurements.

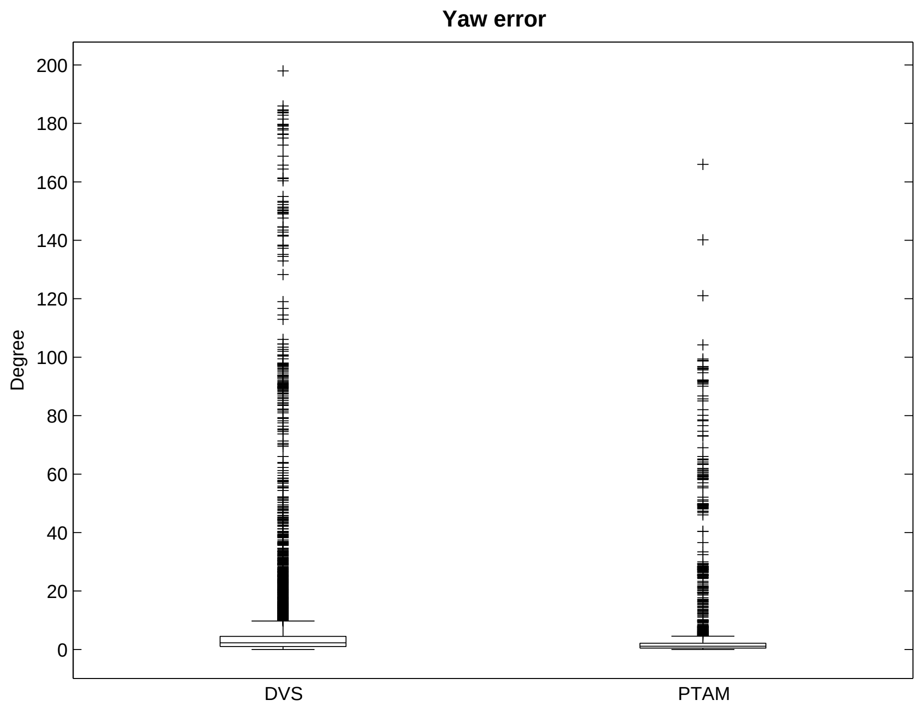

Figure 3.9: Statistical plot of the rotation error of PTAM in reference to the OptiTrack measurements.

Chapter 4

Conclusion

Using the DVS for tracking has some clear advantages as the high sampling speed enables the tracking of LEDs pulsed with frequencies above a 1000 Hz. Compared to normal high speed cameras, the data output and thus the processing is reduced, as only change is advertised by the camera. This makes the DVS also interesting for embedded processing. The measurements revealed that the DVS is able to reacquire stable tracking after as the quadrotor flips with negligible delay as soon as the LEDs are visible again. In comparison to the PTAM used by the quadrotor, it is more than twice as fast. The DVS thus has a clear advantage in comparison to conventional cameras as it does not suffer from blurring. Nevertheless, we used active markers in our experiment which are easier to identify than image features. Due to the fact that the DVS not only needs motion in order to produce input but also has a lower resolution than conventional imaging sensors it is not clear yet, how well features from the environment can be used as a tracking reference. The pose estimation has proven to be more accurate than PTAM and should thus perform well in helicopter navigation. Nevertheless, the low resolution of the DVS limits the range in which a helicopter can be tracked robustly. As there are possible improvements to our algorithm in terms of stability and robustness, we reckon that is should be possible to even improve the pose estimation accuracy on the DVS. A possible approach could include considering not only local maxima from single pixels, but averaging over several pixels activated by an LED. Alternatively another particle filter could be used to smoothen the position readings over time. This approach would also help with the occasional appearance of misdetections. On the hardware side we found a couple of improvements as well. For further experiments reliable drone control is important. While reducing or better balancing the payload could have an impact, we were also not sure if the highly used rotor blades or other parts of the quadcopter were responsible for the unstable flight. As recalibration of the internals seems unfeasible one might also consider using a different drone in the future. To improve visibility of the markers, LEDs with higher angle and power output would be beneficial. This requires some additional hardware though which needs to be considered again in terms of payload.

Apart from a number of possible improvements, the approach has shown to be feasible and has demonstrated the advantages of using and event-driven approach for vision based robotics. As this utilization of asynchronous vision is a rather novel approach in robotics it would be interesting to use this camera in other navigational tasks in future projects, for example for visual SLAM on autonomous ground vehicles. While small image features as used in SIFT [5] might be difficult to use due to resolution line feature extraction, as describe in [1], could be a feasible approach. Additionally, as edges are the natural source for DVS events it would have an advantage in terms of processing compared to normal cameras.

Bibliography

[1] R. Deriche and O. Faugeras. Tracking line segments. *Image and Vision Computing*, 8(4):261 – 270, November 1990.

[2] M. Hofstetter. Temporal pattern-based active marker identification and tracking using a dynamic vision sensor. Master thesis, Institue of Neuroinformatics, ETH Zurich, January 2012.

[3] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. *IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007.

[4] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 x 128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566 – 576, February 2008.

[5] D.G. Lowe. Object recognition from local scale-invariant features. *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2(2):1150 – 1157, 1999.

[6] C. Saner. Autonomous map building with a low-cost quadcopter. Semester thesis, Artificial Intelligence Lab, University of Zurich, December 2012.

ai lab
Robotics and Perception Group

Title of work:

Quadrotor tracking and pose estimation using a dynamic vision sensor

Thesis type and date:

Master Project, February 2013

Supervision:

Prof. Dr. Davide Scaramuzza
Dr. Andrea Censi
Christian Brändli

Student:

Name:	Jonas Strubel
E-mail:	jonas.strubel@uzh.ch
Legi-Nr.:	07-916-901

Statement regarding plagiarism:

By signing this statement, I affirm that I have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.ethz.ch/students/semester/plagiarism_s_en.pdf`

Zurich, 9. 2. 2013: ____________________