University of Zurich
Department of Informatics

ai lab
Robotics and Perception Group

Jonas Strubel

Quadcopter pose estimation using the dynamic vision sensor

Semester Thesis

Robotics and Perception Lab
University of Zurich

Supervision

Davide Scaramuzza
Christian Brändli
Andrea Censi

February 2013

Contents

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 a certain reference in their environment. There exist several methods to achieve this, the most popular one being the use of cameras. The cameras can be either placed on the MAV itself or statically in the environment to track MAVs in their field of view. In order to estimate the pose, specific features in the camera images must be found and tracked. This works well as long as there are significant features to be found in the image. Since conventional camera sensors generally provide not more than 30 frames per second, fast motion will blur the camera image and thus hinder feature detection. Therefore, during an aggressive maneuver tracking is usually lost and the tracking needs to be reacquired after getting clear images again. To re-establish a lost track a certain amount of data needs to be gathered which induces a significant delay in the reacquisition of tracking. To overcome this problem one could of course employ cameras with higher frame rates. This would come at an expense though, not only in terms of data processing but also in much higher price. Therefore we wanted to find a solution which is not only computationally cheaper while still providing high speed but also comes at a lower price as todayâs high speed cameras. The Institute of Neuroinformatics at the University of Zurich and ETH has developed a dynamic vision sensor (DVS). Inspired by the human retina, each pixel fires asynchronously and only if there is change in illumination on this specific pixel. Thus, the data from the DVS are not frames but events depicting the time and place of an impulse. The change of illumination is thereby binary, i.e. only the direction of change is advertised (This binary representation in events will be referred to as polarity in the following). That 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 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. Using such special hardware 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 [DVS paper]. 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 [Matthias]. The LEDs were attached to a quadrotor helicopter, while the quadrotor itself was tracked by the DVS. For the marker tracking we developed an algorithm able to track several LEDs pulsed with different frequencies while using an already existing library for pose estimation. 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 results. Finally a conclusion summarizes our findings as well as discusses possible improvements. Also, an outlook on further interesting projects will be given.

1.1 Related Work

Chapter 2

Approach

2.1 Considerations and Challenges

First of all, working with special hardware like the DVS, which functions differently than conventional cameras, requires a different approach to handling and processing the data from ground up. Using normal cameras, naturally conventional vision algorithms are based on the analysis of frames. In order to make use of the asynchronous behavior an image-based approach thus is not feasible, i.e. the algorithm had to make use of the event-based behavior 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 any significant output apart from noise. Another limitation is the low spatial resolution. This makes it unfeasible to detect small features in the field of view. Furthermore, the low resolution will have a higher uncertainty of the position of a feature or marker respectively in image space compared to conventional cameras. As previously mentioned, the sampling frequency, i.e. also the accuracy will vary. Therefore it was not clear beforehand what frequencies could be handled by the DVS under our conditions with the quadrotor.

2.2 Preparation

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

2.2.1 C++ interface to the DVS

The framework provided with the DVS (jAER) is completely written in Java as well as the tracking algorithm develop by [Mathias]. Since we wanted to implement the algorithm in C++ due to performance and portability (e.g. to embedded systems) considerations it was necessary to write a driver interface for the DVS in C++ first. Thesycon [footnote], the provider of the USB device driver, already provides basic frameworks for different programming languages to interface with their driver. A basic interface was written to pull the event data from the DVS. Similar to jAER, the events were 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 also advantageous for certain operations, e.g. if they are computationally costly and furthermore if single events donât 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 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 number of independent Pulse-Width-Modulation (PWM) drivers was an important property in order to pulse several LEDs with different frequencies. The AVR has 7 PWM drivers, from which 5 are free to use. An already existing example C code [link] was taken and adapted to our needs. A python script was also written to remotely adjust the PWMs via USB from a PC. The choice of LEDs was also important, since they should be well visible. The ideal LED for our project would be high power and with a wide emission angle. Since the DVS is very sensitive to the infrared spectrum, we decided to go for infrared LEDs[type hint]. Finally the accuracy of the induced frequencies was verified with an oscilloscope. In order to find a position and orientation of an object in space at least four reference points are necessary [ref]. Therefore we decided to attach four LEDs to our quadcopter.

2.3 The tracking algorithm

The following paragraphs describe the separate steps of the tracking algorithm we developed. The approach previously designed by [Mathias] is 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 our approach. In our setup 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 the approach by [Mathias] 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 resulting signal resembles 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 stable measurement of frequencies
- The influence of background noise in frequency measurement

Having static LEDs in the cameraâs âimageâ the straight-forward approach of measuring the inter-spike-intervals (ISI) [A neuroscientific terms for the time interval between two firings of the same neuron, here a pixel] 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 DVS data over several scenarios by moving the DVS camera 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 will generate a considerably higher amount of events than uniform backgrounds (e.g. white walls). Evaluating the different data on a frequency histogram we found that most of the movement induced background noise generates frequencies up to around 500Hz [plot]. 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 (Mathias Master thesis). Furthermore, the spacing between the frequencies was also important in order to prevent inaccurate measurement to 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 the one of another LED). Experiments showed frequencies of 740, 1030, 1320 and 1610 Hz to work well.

2.3.2 Frequency period estimation

Following the assumption of using the ISIs on single pixels for period estimation, where an event denotes a rising or falling edge of the signal, these needed (i.e. the change of polarity of consecutive events) to be taken into account for each pixel first. 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 provides 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 including the current timestamp. Similar to an event, pixel location, the transition polarity and timestamp of the transition are stored. Similar to the previous step each transition is again 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 transition for each pixel. Following our previous assumption, before insertion of a new transition the time interval to its predecessor is now measured. This provides a period estimate and thus a certain frequency.

2.3.3 Frequency 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, i.e. a 2D-array of pixel-space resolution as before. 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 finding 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. As an interval during which to gather weights 10ms showed to provide robust results. The most likely position of a certain LED was 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 of the weight map needs to be maintained. Such a local maximum should denote the position of an LED. Therefore we introduced a minimum distance between maxima. 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.

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. For more insight, please refer to [particle filter paper]. The following paragraph describes our application of such a filter.

2.4.1 Particle filter

So far each evaluation of local maxima provides us with 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 with a fixed set of particles. Such a particle stores the following values: the position in pixel-space with sub-pixel resolution, an uncertainty value, the weight of the particle and a time stamp of its creation. The uncertainty of the particle denotes the possible radius in pixels in which the LED could have moved since the last update of the particle 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. In the following the process of maintaining the particles is explained: Initially, the particle filter is empty. For every new hypothesis a candidate particle is created. Thereby, the position and weight are inherited from the hypothesis 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 can be merged with an existing one. Therefore candidates are compared to the 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, replacing the oldest particle if the filter is full. During merging two particles, their uncertainties decide upon their contribution for the new particle created. The new position is thus linearly interpolated according to their uncertainty. Similarly, the new weight is a summation of particle values weighted by their uncertainty. 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. The simplest approach would choose from each filter the particle with the highest weight, i.e. the particle with the highest support. Unfortunately though, we found that the frequencies we used were close enough to be ambiguous for frequency estimation. Thus, apart from random noise, close-by frequencies tended have particles of different signal in similar locations. This required then, to find the best valid combination of particles. This means, that each possible combination needs to be checked for a minimum distance between the particles to each other. The significance of a valid combination was found by multiplying the particle weights. In order to solve this combinational problem a recursive function was used to traverse all the different combinations. Since such combinatorial problem implies many recursions and thus becomes 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 favor finding 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 LED position, we were now able to estimate the relative pose of the helicopter to the camera. We used an existing algorithm from the openCV [link] library for that purpose. Given the feature points in the objects reference frame, the correspond 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

It could happen, that position would shortly jump, when the weights of a wrong placement were winning. Another problem in the particle filter approach is the fact that the local maxima, i.e. the position of a single pixel, is taken to update a particle. Since an LED usually produces a blob comprising several pixels on the camera, maxima tend to change position, which gives the particles an oscillating behavior.

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 perform badly, we planned to test it under real conditions by flying aggressive maneuvers quadcopter while tracking it with the DVS. Furthermore, since the DVS has a much lower resolution than todayâs standard CMOS cameras, the accuracy at which pose can be estimated with the DVS needed to be evaluated as well. [Saner] has developed a framework for controlling the ARDrone including the ability to let the drone do a flip, i.e. a 360 degree roll. We decided to use this flip for our aggressive maneuver. As we were only using one camera, that would mean, that the LEDs would lose track somewhere during the flip. Thus we wanted to see for how long the tracking would be lost.

3.1.1 Setup

For our setup we used the commercially available ARDrone 2.0 on which we attached our set of four LEDs as well as the microcontroller. Each LED was fixed looking downwards (if one assumes the drone to be in steady flight mode), one under each of the four rotors. 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 placed on to floor facing upwards.

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) 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 [link] 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 droneâs onboard camera, as well as the raw event data from the DVS were recorded. 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 droneâs motors are powerful enough to pull it upwards fast, which is also necessary in order to do a flip, 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 would usually crashing down on the floor. This meant that a stable reacquisition of the localization with the droneâs internal camera was mostly not possible for a longer time than expected. The optical tracking system introduced another challenge into the experiments, since it uses reflection markers for tracking 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 very important 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 was not a problem for our experiment setup.

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 will be discussed. Before evaluation the data from the DVS was passed through our algorithm logging the position data output for each recording.

3.2.1 Tracking speed

To measure how fast the different systems are able to reacquire tracking we first synchronized the recordings by hand. Since 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 time 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 the reference. Figure 3.1 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. Our algorithm lost track during a mean of 0.35 second with a standard deviation of 0.1 seconds. PTAM respectively lost track for a mean of 0.8 seconds with a standard deviation of 0.33 seconds. The average flip took 0.55 seconds with a standard error of 0.15 seconds. One can clearly see that the time where tracking is lost is much shorter with our approach in respect to PTAM. As figure 3.2 further illustrates, the downtime for the DVS is shorter than the flip duration. As verified with our recordings the downtime of the DVS corresponds to losing sight of the LED markers because of their emission angle. We reckon that with a suitable configuration of either more markers or cameras, tracking could be maintained during the whole flip. The PTAM data on the other hand 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.

3.2.2 Pose estimation

As already mentioned, 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 by minimizing function [formula] using the non-linear least squares algorithm from MATLAB. [function explanation] 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

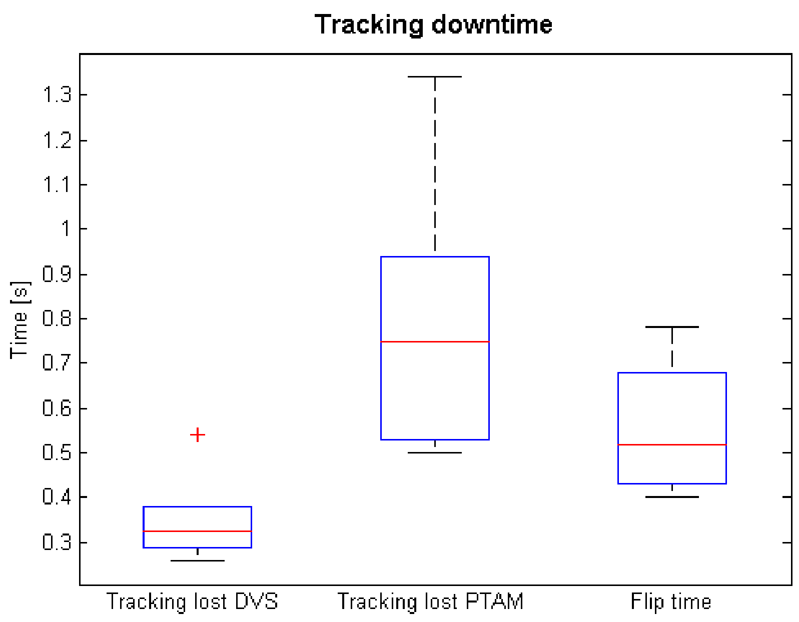

Figure 3.1: Statistical plot of measure time interval. The boxplots show the time interval in which tracking is lost for the our algorithm and PTAM compared to the duration of a flip.

interpolating the translation and rotation. The transformation gained from the least-squares algorithm 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, again was fitted using the non-linear least squares algorithm from the OptiTrack to the DVS frame. The mean translation error for the pose estimation over all our recordings was found to be 8.9 cm with a standard error of 12.6 cm. Figure **??** shows the statistical distribution of the pose errors (for details on the boxplot, please refer to 3.2.1).

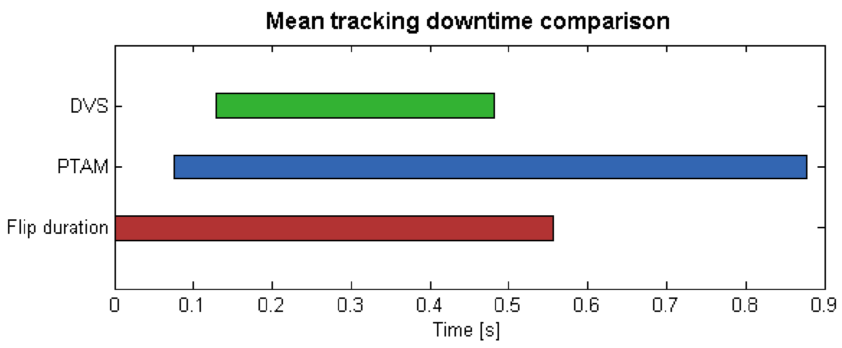

Figure 3.2: Comparison of the mean tracking downtime intervals. The mean time intervals of both algorithms are compared against the mean flip time on a time line.

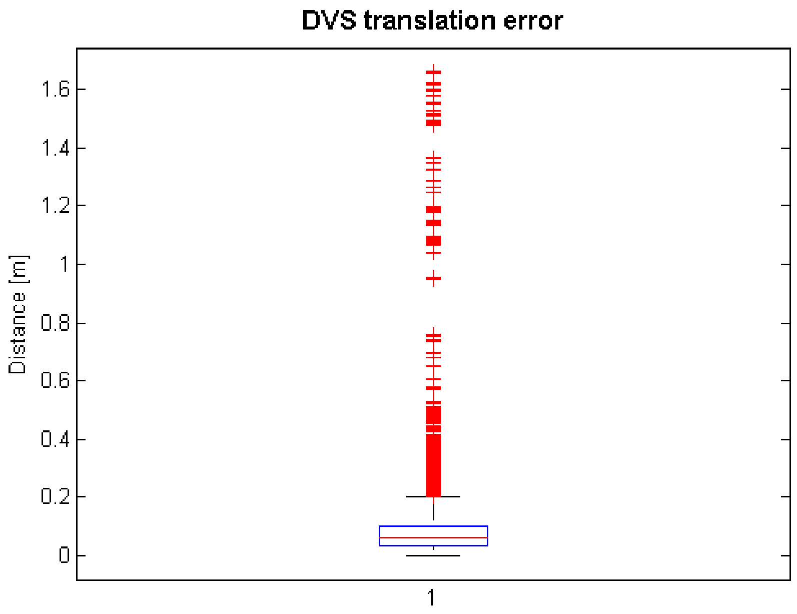

Figure 3.3: Statistical plot of the pose estimation error of the DVS in reference to the OptiTrack measurements.

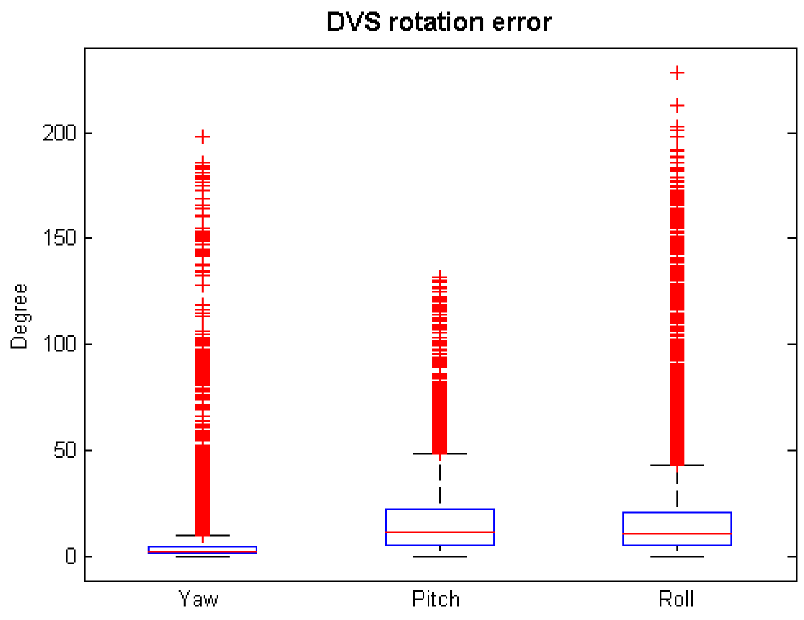

Figure 3.4: Statistical plot of the rotation error of the DVS in reference to the OptiTrack measurements.

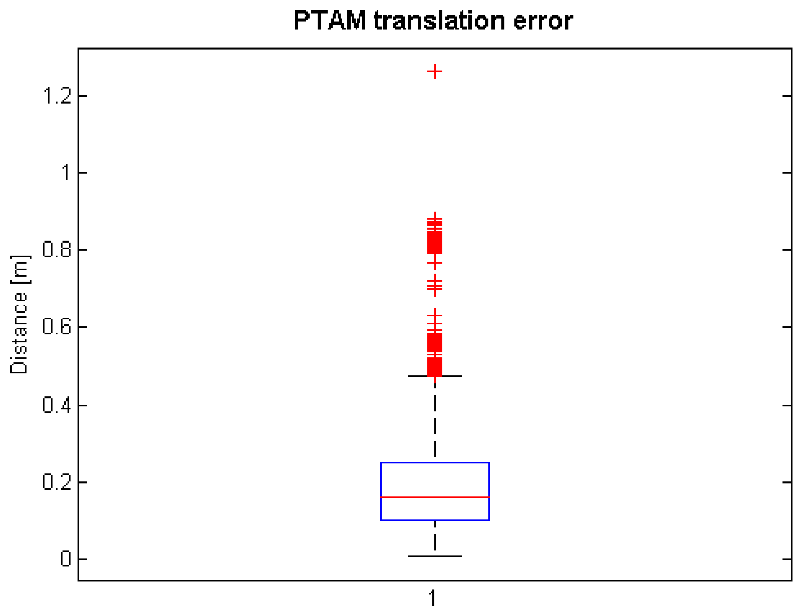

Figure 3.5: Statistical plot of the pose estimation error of PTAM in reference to the OptiTrack measurements.

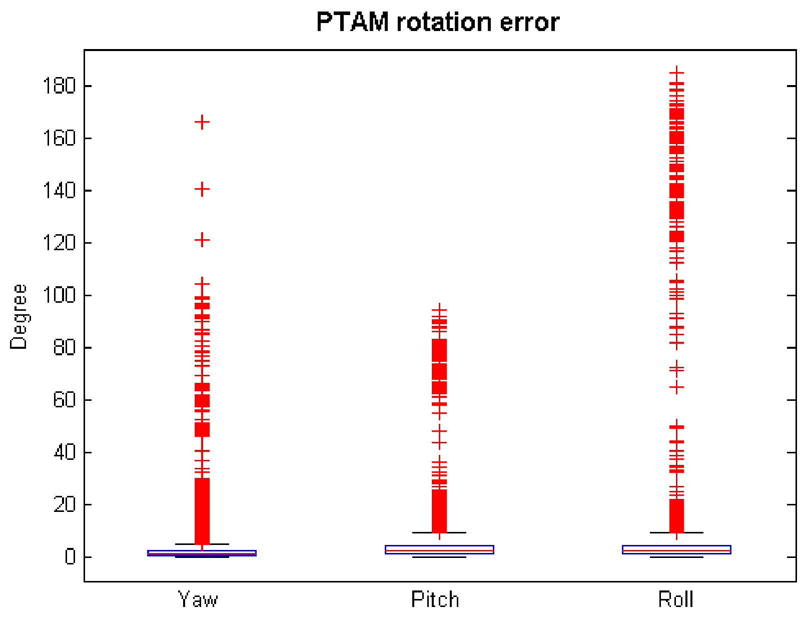

Figure 3.6: Statistical plot of the rotation error of PTAM in reference to the OptiTrack measurements.

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 1000Hz. Furthermore. 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 mean pose estimation error has shown to be low enough to be used in helicopter navigation (comparison???) despite the low resolution of the DVS. 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 invalid LED combinations. 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 in the payload problem. 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.

Appendix A

Something

In the appendix you can provide some more data, a tutorial on how to run your code, a detailed proof etc.

ai lab
Robotics and Perception Group

Title of work:

Quadcopter pose estimation using the dynamic vision sensor

Thesis type and date:

Semester Thesis, February 2013

Supervision:

Davide Scaramuzza
Christian Brändli
Andrea Censi

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, 7. 2. 2013: ______________________________