Real-time Velocity Estimation Based on Optical Flow and Disparity Matching

Dominik Honegger, Pierre Greisen, Lorenz Meier, Petri Tanskanen and Marc Pollefeys
ETH Zürich, Switzerland

Abstract— **A high update rate of metric velocity values is crucial for a robust operation of navigation control loops of mobile robots such as micro aerial vehicles (MAVs). An efficient way for obtaining metric velocity of robots without external reference are image-based optical flow measurements, scaled with the distance between camera and the observed scene. However, since optical flow and stereo vision are computationally intensive tasks, metric optical flow calculations on embedded systems are typically only possible at limited frame rate.**
In this work, we therefore present an FPGA-based platform with the capability of calculating real-time metric optical flow at 127 frames per second and 376x240 resolution. Radial undistortion, image rectification, disparity estimation and optical flow calculation tasks are performed on a single FPGA without the need for external memory. The platform is perfectly suited for mobile robots or MAVs due to its low weight and low power consumption.

I. INTRODUCTION

Mobile robots often require a constant update of their state parameters such as position and velocity to operate autonomously. In particular, autonomous micro aerial vehicles (MAV) require high-rate and low-latency updates of their current velocity to guarantee a successful flight operation [4]. Such navigational information can be obtained by inertial navigation systems (INS) in conjuncture with GPS or by the use of computer vision systems. Vision systems are very versatile because they can extract a lot of information from the environment and only need a few image sensors.

However, image sensors generate a high amount of data which needs to be processed in real-time and at low-latency. Hence, high computational power is necessary which conflicts with the often stringent limitations on energy consumption and also on maximal weight and size. Computer vision algorithms typically run in real-time only on high-end CPUs or GPUs consuming a prohibitive amount of energy. Also, the usage of CPUs increases processing latency which is unwanted in feedback loops.

Field Programmable Gate Arrays (FPGAs) are a potential alternative to overcome the computational and latency bottlenecks of real-time computer vision algorithms. FPGAs have high computational performance but only consume a fraction of the power compared to processor-based systems. Low-level vision tasks can indeed be efficiently executed on FPGAs [1]. Multiple camera streams can be handled in parallel in real-time and at low latency due to pipelined stream processing.

In this work, we show an FPGA system that performs radial distortion correction, stereo rectification, depth estimation, and optical flow at 127 stereo frames per second at a resolution of 376x240 pixels. The obtained flow values are scaled with the estimated depth values for each pixel, which are used to determine the metric velocity of the camera with respect to the observed scene. The system is implemented with a low-end FPGA and provides accurate results.

A lot of computer vision algorithms have been successfully ported to dedicated hardware: FPGA systems that perform disparity estimation in real-time on high resolution [6], ASIC implementations of stereo vision [2], and optical flow estimation on FPGA [5]. [3] presents a combination of stereo vision and optical flow on a GPU-FPGA system, targeted for automotive driver assistance. All these systems are however not targeted toward mobile robot operation. Very recently, a low-power optical flow sensor for robot applications has been presented [7]. We extend this work by increasing computational performance and, more importantly, by adding depth information for metric velocities.

In summary, the contributions of this work are as follows. We first provide a summary of simple yet efficient and well-performing algorithms that combined provide robust metric optical flow values which can be used to calculate metric velocities of a mobile robot. We show an efficient FPGA architecture of all of these algorithms and provide key results of the implementation. The resulting FPGA system is low-power, low-latency, low-cost, and small in size an thus perfectly suited for mobile robot applications. To validate our system, we show our measurement setup and compare our results to the results of a VICON system.

II. BACKGROUND

In this section, we summerize the relation between pixel-based optical flow and metric velocity. Section III provides more details on the employed algorithms for optical flow and disparity estimation.

A. Basic Equations of the Optical Flow Field

The optical flow field is the projection of the 3-D velocity field on the image plane. Let $\mathbf{P} = [\mathrm{X}, \mathrm{Y}, \mathrm{Z}]^\top$ be a 3-D point in the camera reference frame. The optical axis is the Z-axis, f denotes the focal length and the projection center is in the origin. The pixel coordinates of $\mathbf{P}$ on the image plane are given by

$$\mathbf{p} = \mathrm{f}\frac{\mathbf{P}}{\mathrm{Z}}. \qquad (1)$$

Since the distance of the image plane to the origin is equal to the focal length f, the third coordinate of $\mathbf{p}$ is constant

$\mathbf{p} = [x, y, f]^\top$. The relative motion between $\mathbf{P}$ and the camera is given by

$$\mathbf{V} = -\mathbf{T} - \omega \times \mathbf{P}, \tag{2}$$

where ω is the angular velocity and $\mathbf{T}$ the translational component of the motion. The derivative with respect to time of both sides of (1) leads to the relation between the velocity of $\mathbf{P}$ in the camera reference frame and the velocity or the flow of $\mathbf{p}$ in the image plane

$$\frac{\mathbf{flow}}{\mathbf{\Delta}\mathrm{time}} \mathrel{\widehat{=}} \mathbf{v} = f\frac{Z\mathbf{V} - V_z\mathbf{P}}{Z^2}. \tag{3}$$

Written in components and using (2) the optical flow field is defined as

$$v_x = \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy - \omega_y x^2}{f} \tag{4}$$

$$v_y = \frac{T_z y - T_y f}{Z} + \omega_x f - \omega_z x + \frac{\omega_x y^2 - \omega_y xy}{f}. \tag{5}$$

The components of the optical flow field are the sum of pure translational and pure rotational parts. The rotational parts are independent from Z and therefore the angular velocity does not carry scene depth information.
The translational components in (4) and (5) are scaled with the focal length and the current distance Z to the scene. In situations where we are only interested in the translational velocity, e.g., when the rotational veclocity is known or constant, the translational velocity can be transformed into *metric* scale

$$\mathbf{v}_{m,trans} = \mathbf{v}\frac{Z}{f} = \mathbf{v}\frac{b}{d} \tag{6}$$

where b is the interaxial distance between the two cameras and d the estimated disparity values. The combination of optical flow values and depth estimation leads to a translational velocity measurement per pixel in metric scale.

III. SYSTEM SETUP

The proposed system performs optical flow calculation and depth estimation. Cameras are directly connected to an FPGA, to process the image stream in real-time. An overview of the setup is shown in Figure 1.
The computer vision module consists of three main blocks. In the undistortion and rectification block, the images are corrected to account for nonlinear lense distortion and vertical position offsets of the left and right cameras. Inside the stereo block, the disparity estimation is performed and finally the flow block performs the optical flow calculation on one of the rectified images.

A. Undistortion and Rectification Block

The undistortion and rectification is done using backward mapping. The coordinates of the pixel in the target image are distorted and then unrectified to get the information where in the source image the corresponding pixel value is located. This value is taken and interpolated into the new image. A buffer is used to store the incoming source pixel values.
The undistortion as well as the rectification are warp operations, which allows for combining them into just one

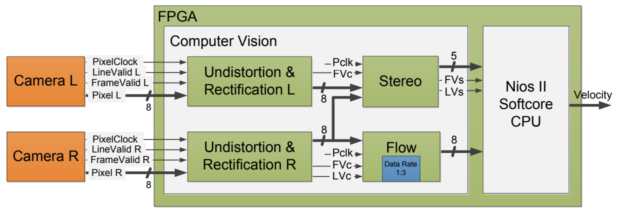


Fig. 1. System overview, cameras are directly connected to the FPGA. The video streams are undistorted and rectified in the Computer Vision Module. Then disparity estimation and optical flow calculation is performed. The flow values scaled with the corresponding distance are finally sent out.

operation.
The backward mapping outputs fractional coordinates. Bilinear interpolation is used to get the new value at the desired coordinate out of the four nearest neighbours. The final output is the undistorted and rectified pixel data, where the epipolar lines are aligned with the horizontal lines.

B. Stereo Block

After the undistortion and rectification, stereo matching can be performed between the two corresponding images. A bit-oriented cost computation is used based on a census transform [8]. For each pixel in one image the best match along the corresponding epipolar line in the other image is selected within a search range of 32 pixels, which corresponds to approximately 10% of the image width.
The different viewpoint of the two images can cause occlusions, i.e., a point in one image is not visible in the other. To prevent wrong matching results in such situations, a left-right consistency check is performed and the inconsistent points are rejected. A median filter of size three by three removes spikes on the disparity output.

C. Flow Block

Optical flow is calculated between two successive frames. Since the storage of one complete frame requires almost the whole available embedded memory, the flow is not calculated dense but on a downscaled grid. 36 rows and 23 columns are stored, resulting in 828 intersection points where the flow is calculated. Due to the reduced amount of points, the complete flow calculation can be performed within the same frame.
The rows and columns of the corresponding images are stored separately and a correlation between the stored rows and columns from the adjacent frames starts as soon as enough data has arrived. A search range of ± 7 pixels is used for the 32 pixel wide correlation window. The resulting horizontal and vertical optical flow is stored in a memory. A sub pixel estimation, using parabola fitting, creates additional accuracy in adding fractions to the ± 7 pixel correlation window result.
The ± 7 pixels correspond to ± 3.5 meters per second for an object at two meters at 127 frames per second (fps), which is enough for our application.

D. Components

The used image sensor is the MT9V032 CMOS sensor from Aptina. The active imaging pixel array is 752Hx480V with a pixel output clock of 25 MHz. This allows for up to 60 fps running at full resolution. Using windowing or pixel binning the frame rate can be further increased, by downscaling the image.
A well-performing configuration mode is obtained by binning rows and columns twice, since it reduces the resolution to 376Hx240V, but improves sensitivity and increases the frame rate to 127 fps.
A suitable FPGA development board, which is small and light weight enough to allow tests on a mobile robot is the DBM3C80 from DevBoards[1], it is equipped with a Cyclone III EP3C80 FPGA from Altera. There are enough I/O pins to connect multiple cameras as well as an ethernet phy.

E. FPGA Infrastructure

Besides computer vision and communication, the FPGA also performs maintenance tasks. The camera chips are configured over an I2C interface to achieve the desired resolution and frame rate. An ethernet link is used to transmit velocity values as well as processed images to a PC for debug and visualization purposes. A Nios II soft-core CPU from Altera is instanced within the FPGA to perform these tasks.

IV. FPGA ARCHITECTURE

In the following, we describe the FPGA architecture of the computer vision algorithms.

A. Undistortion and Rectification

For each frame, a coordinate counter steps uniformly up to the output resolution. To produce the target coordinate, the coordinates are distorted and unrectified according to the backward mapping algorithm. The resulting coordinate is used to get the corresponding pixel value out of the source pixel buffer. Since the calculated coordinates do not point exactly at one pixel position, the four nearest neighbouring pixels are combined using bilinear interpolation to the final corrected pixel value.
The design is fully pipelined and runs at 25 MHz pixel clock. The divisions inside the rectification are done using pipelined divison cores from Altera to achieve the required throughput of 25 MHz. The precision of the custom fixed-point arithmetic is designed to fit inside the available 18x18-bit multipliers of the FPGA.
To enable parallel readout of four neighbouring pixels in each cycle, incoming pixels are stored column- and row-interleaved in four separate buffers with a separate read/write port each.

[1]www.devboards.de

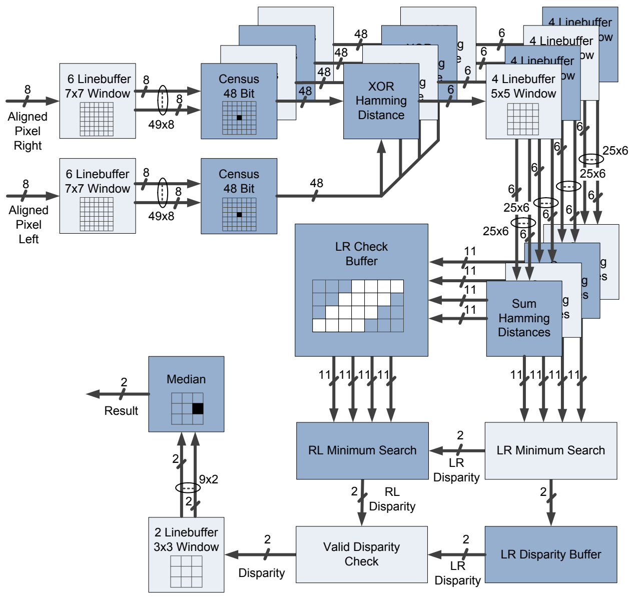


Fig. 2. Stereo module data path for a simplified disparity search range of four pixels. Left and right image pixels are census transformed using a 7x7 window. The correlation is done in parallel, the last four right census vectors are delayed for that reason. The disparity value is found at the minimal Hamming distance. For robustness a 5x5 window of Hamming distances is used where the minimum is searched out of the 11 bit total sum. A RL-consistency check is performed to minimize error due to occlusions. Finally, a 3x3 median filter removes spikes.

B. Disparity Estimation

The census transform creates a bit vector containing information about a present pixel p and its neighbouring pixels. The intensity values of the neighbouring pixels are compared with the intensity of pixel p. The corresponding bit in the census vector is set to zero or one depending on whether the intensity is less or greater.
The dissimilarity between two census vectors is measured with the Hamming distance, which is the number of bits different in the two vectors. Perfect matching would get a Hamming distance of zero, whereas completely different vectors lead to a Hamming distance equal to the total number of bits of one vector. The robustness can be increased using a window of Hamming distances instead of using just one to find the best match.
Figure 2 shows the data path inside the stereo module. Because of the window-based processing of the census transform a pixel and its neighbours need to be accessed simultaneously. A set of registers, called window buffer, and a set of line buffers are implemented to provide all of the needed intensity values within one census transform.
The rectified and undistorted pixel values of the stereo camera pair are streamed into the top left position of a window buffer. This square dimensioned window buffer is made out of shift registers, each of them capable to store one pixel value. Each pixel clock the values are shifted one position from left to right. The right-most column of the window buffer is connected to line buffers. The line buffers are capable to store one complete line minus the size of one

line of the window buffer. For the census transform a window buffer with dimension of seven by seven pixels and six line buffers are instanced for each camera. The shift registers in the window buffer allow a simultaneous access to all the pixels.

A total of 48 intensity compare operations are performed in parallel for each camera to get the census vector of the middle pixel within the seven by seven window. A simple XOR operation between the census vectors from both cameras results in a bit vector containing a '1' for a difference in the two census vectors at that position or a '0' for identity. Summing up this vector using an adder tree leads to the Hamming distance of the two compared pixel positions. Since the range of the Hamming distance between two pixels can be from 0 to 48, six bits are needed to represent it.

The disparity search is done finding the minimum Hamming distance within the search range of 32 pixels. To produce more robust results not the single Hamming distances are compared, but windows of size five by five containing Hamming distances.

All the correlation candidates are calculated in parallel. The last 32 census vectors from the right image are delayed and processed in parallel with the current census vector from the left image. The resulting 32 Hamming distances are compared within the five by five window. The window containing the minimum Hamming distances in total is selected as the best match. The coordinate difference from the current pixel in the left image and the one in the right with minimal sum of Hamming distances, represents the LR-disparity result.

The search for disparity is performed form left to right. To avoid wrong results caused by occlusions, every found disparity needs to be cross-checked from right to the left. The summations of the Hamming distances within the five by five windows are therefore stored in 32 independent LR-buffers. Since the disparity range is 0 to 31, the buffers need to store $\pm$ 32 summations each.

The found LR-disparity defines the output order of the LR-buffer to maintain the corresponding RL-candidates. The location of the minimum RL-candidate represents the RL-disparity. The difference between LR- and RL-disparity has to be less than a predefined threshold value of two, to result in a valid disparity output.

Using a total amount of 19 compare operations organized in nine pipeline steps, the median value out of the nine possible pixel values is found and sent out.

The design runs at a pixel clock of 25 MHz.

C. Flow

Figure 3 shows the data path of the optical flow module. The incoming pixel data is first filtered with a five by five gaussian filter mask. Horizontal and vertical gradients are built afterwards and stored in separate memory blocks.

Since pixel data is streamed row by row, only one data point for each vertical line arrives within one row.

Correlation is done for horizontal data first. Every time a complete horizontal line is stored in memory, the correlation stage performs the flow calculation at the 36 points within this line. 32 multipliers are implemented in parallel with a successive adder tree to allow a pipelined cross-correlation. The search range is in total 15 pixels wide.

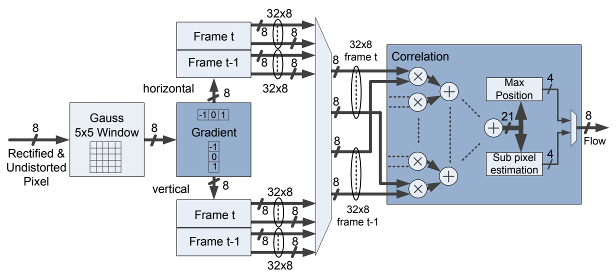


Fig. 3. Data path of the flow calculation module. Data is first Gaussian filtered, horizontal and vertical gradients are built afterwards. Both gradients are stored separately in memory. Horizontal flow is calculated first, vertical afterwards. The correlation is performed over a search range of 15 pixels in total. Sub pixel estimation based on a parabola fit gives additional accuracy.

Incoming data from frame n stays the same, whereas data from frame n-1 is shifted 15 pixels in total within one correlation. 64 pixel values from frame n and 96 values from frame n-1 are taken out of the memory to perform the correlation at four points in series. This is repeated nine times until the correlation is done at all 36 points in a row. The correlation position resulting in the maximum value is taken as the flow result.

The value of the maximum correlation and the precedent and successive neighbour are used within the sub-pixel estimation. Parabola fitting is performed to reach additional four bits accuracy. Vertical data is processed as soon as the first vertical line is complete in memory. The data is filled into the correlation stage in the same way as in the horizontal calculation.

Using dual port memory, the correlation stage can be clocked independently from the other processes. The clock rate is adjusted to finish all correlation calculations until the end of the frame. The correlation is running at 75 MHz, whereas the other parts of the design run at a pixel clock of 25 MHz.

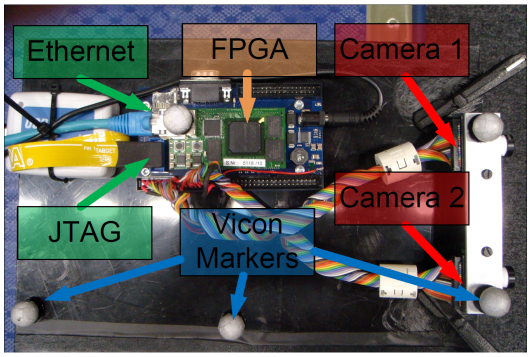
V. RESULTS

The resulting system runs at 127 fps, calculates a dense disparity map (376x240 pixels) and optical flow on 36*23 points. The overall latency is 30 lines that is 450 μs (micro seconds) at 25 MHz pixel clock. Figure 4 (a) shows a picture of the system with labeled main parts.

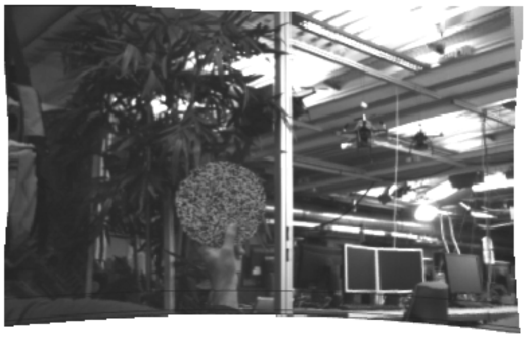
Velocity values, horizontal and vertical flow data and the disparity images are sent to a host PC using the ethernet link. The right undistorted and rectified image as well as the corresponding disparity map output and the flow field are shown in Figure 4 (b-d).

A. Implementation

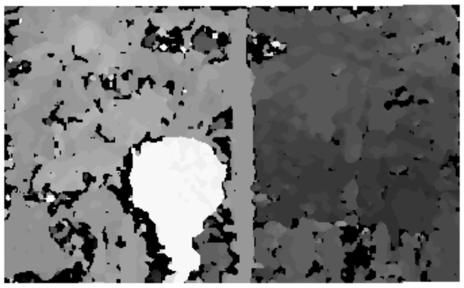
The resource utilization on the Cyclone III FGPA is provided in Table I. The design containing one disparity estimation-block and one optical flow calculation-block fits



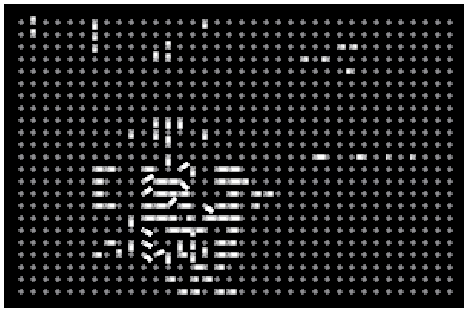
(a)



(b)



(c)



(d)

Fig. 4. Overview of the prototype system with labeled main parts in (a), right undistorted and rectified image (b), the output of the stereo module is shown in (c), flow field (d)

TABLE I
USED RESOURCES OF STEREO SETUP AND FLOW CALCULATION, IMPLEMENTED ON A CYCLONE III FPGA.

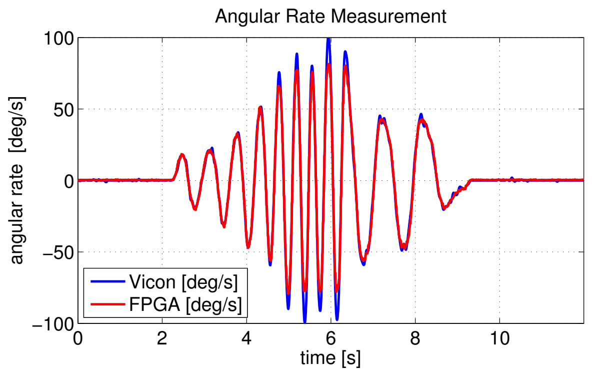
Module	Logic Elements	Embedded Memory (Kbits)	M9K Blocks	18-bit x 18-bit Multipliers
Flow	10273	272	34	16
Rectification& Undistortion L	3619	128	16	18
Rectification& Undistortion R	3656	128	16	18
Stereo	17577	342	83	0
Nios II	13122	153	45	2
Others	1408	88	14	0
Total	49655(49%)	1111(40%)	208(68%)	54(22%)

easily into the device.
The stereo module takes most of the used memory blocks to support the 32 pixel disparity range. Rectification and undistortion is performed with a line buffer of 40 lines, lenses with more distortion or a camera setup with poorly aligned cameras would require larger buffers.
However, 40 lines are sufficient in most practical situations. The system runs at the incoming pixel rate (25 MHz), except for the optical flow correlation stage, running at 75 MHz.
The FPGA consumes 2.8 Watt while performing optical flow calculation and disparity estimation. One camera consumes 0.3 Watt.

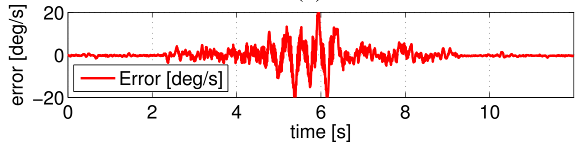
B. *Metric Velocity Measurement*

A Vicon camera tracking system is used to verify the measurements performed by the FPGA system. The Vicon system is capable of measuring velocities in a metric scale at up to 250 fps with high precision. This is fast enough to verify the FPGA measurements at 127 fps.
In order to validate the obtained velocity estimates two types of measurements are performed. First, pure rotational velocity is evaluated by avoiding translational movements. Second, pure translational velocities are evaluated by moving the system along one axis.
Note that, for the use in navigation-control loops, rotational movement is assumed to be zero or estimated and compensated with an auxiliary sensor such as a gyroscope.

1) Rotation: The system is placed on a rotatable plane. Angular velocity can be measured without distance scaling, which allows for a verification of the optical flow results without scaling with depth values. Without any translational movement and fixed axes except one, the angular velocity of the remaining axis is the measured flow scaled with the focal length as shown in (4) and (5).
The maximum measurable angular rate with the used 508 pixels focal length lenses, 127 fps update rate and ± 7 pixels maximum flow is ± 100 degrees per second. An average over all optical flow points is taken as the resulting output. The camera head is turned in front of a textured surface. Results of the velocity estimates compared to the angular

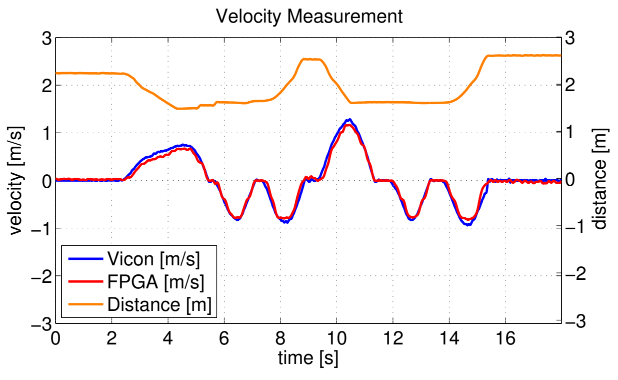


(a)

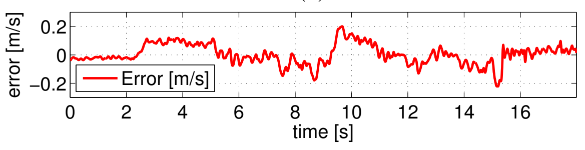


(b)

Fig. 5. Angular Rate Measurement, the camera is rotated in front of a textured wall. Rate measurements from Vicon and FPGA in (a), error in (b).



(a)



(b)

Fig. 6. Translation measurement with different textures in the scene located at various distances to the camera head. Velocity measurements from Vicon and FPGA (left scale axis) and current distance (right scale axis) in (a), error in (b).

velocity obtained from the VICON system are shown in Figure 5. Up to a velocity of 80 degrees per second the system is quite accurate. Since each wrong correlation result at measurements with maximum flow lowers the average over all points, the measured velocity is underestimated at peaks.

2) Translation: The camera head including the FPGA is horizontally placed on a cart and moved along one axis. All other axes are fixed, to only allow one translational movement. Measurements are done with different textures in the scene and located at various distances to the camera head. Flow and depth data are combined as shown in (6) to achieve a metric per pixel velocity measurement. Points rejected from the LR consistency check, caused by occlusions or the lack of texture, are considered invalid and are thus neglected. An average over all pixel velocities is taken as the resulting velocity of the system.
Figure 6 shows the measured velocity of a translational movement along textured planes at various distances. Due to the independent scaling of the flow points with the corresponding distance to the scene, any changes in distance are compensated.

VI. CONCLUSION

This paper has shown, how computational intensive tasks such as optical flow calculation and low-level stereo vision can be implemented in a single FPGA to produce accurate metric velocity for MAV applications.
The power consumption of 2.8 Watt is low compared to any other CPU based system, where the power consumption in idle mode is in best case in that range. FPGAs are thus promising candidates for realizing real-time computer vision algorithms. Since the system is also light weight, it is suitable for any mobile robot or micro aerial application.

REFERENCES

[1] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon. FPGA design and implementation of a real-time stereo vision system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(1):15 –26, jan. 2010.
[2] M. Kuhn, S. Moser, O. Isler, F.K. Gurkaynak, A. Burg, N. Felber, H. Kaeslin, and W. Fichtner. Efficient ASIC implementation of a real-time depth mapping stereo vision system. In *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, volume 3, pages 1478 – 1481 Vol. 3, dec. 2003.
[3] V. Mahalingam, K. Bhattacharya, N. Ranganathan, H. Chakravarthula, R.R. Murphy, and K.S. Pratt. A VLSI architecture and algorithm for lucas kanade-based optical flow computation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(1):29 –38, jan. 2010.
[4] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2992 –2997, may 2011.
[5] Clemens Rabe, Thomas Mueller, Andreas Wedel, and Uwe Franke. Dense, Robust, and Accurate Motion Field Estimation from Stereo Image Sequences in Real-Time. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Proceedings of the 11th European Conference on Computer Vision*, volume 6314 of *Lecture Notes in Computer Science*, pages 582–595. Springer, September 2010.
[6] Pierre Greisen Simon Heinzle, Markus Gross and Andreas P. Burg. An FPGA-based processing pipeline for high-definition stereo video. In *EURASIP Journal on Image and Video Processing*. Springer, September 2011.
[7] D. Watman and H. Murayama. Design of a miniature, multi-directional optical flow sensor for micro aerial vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2986 –2991, may 2011.
[8] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. pages 151–158. Springer-Verlag, 1994.