# FPGA-based fast response image analysis for autonomous or semi-autonomous indoor flight

Robert Ladig, Kazuhiro Shimonomura
Department of Robotics, Ritsumeikan University
Kusatsu, Shiga, 525-8577 Japan
gr0150ff@ed.ritsumei.ac.jp, skazu@fc.ritsumei.ac.jp

## Abstract

*Small aerial vehicles, like quadrotor, have a high potential to be helpful tools in first response scenarios like earthquakes, landslides and fires. But even simple tasks like holding position and altitude can be challenging to accomplish by a human operator and even more challenging autonomously. When outdoors, using GPS and pressure sensors is feasible, but indoors or in GPS denied environments it is not. Until now, for indoor flight scenarios either a lot of energy consuming sensors and hardware or a perfectly defined surrounding is required. In this approach, the viability of an onboard FPGA based indoor flight navigation system with a pan-tilt camera mount and a single VGA camera is tested. It can be used to either support an operator performing a hold position and altitude task, or act completely autonomously to achieve this task.*

## 1. Introduction

Due to the decrease of cost and size of accelerometers and other sensors needed for a stable autonomous flight [10], in recent years the interest and research in aerial robotics has continuously grown and surely will continue to grow even faster. Because of the rather simple dynamic model of a certain type of aerial vehicles, so called quadrotor, we have not only seen wide commercial success and public recognition (as with the smart phone controlled AR.Drone [6]) but also a rapid miniaturization in the last few years (for example the 19g light Crazyflie quadrotor [2]). The possibilities that open up, especially in search and rescue (SAR), with the use of autonomous air vehicles are numerous. Due to their small design and a mobility that is independent of terrain, they seem to be the perfect tools for locating areas of attention in a catastrophic event like earthquakes, fires or landslides. Yet, even though the technology is openly available and affordable, the systems used in SAR are mostly ground based, *e.g.* described by Casper and Murphy in the case of the World Trade Center rescue efforts [3].

There are several reasons why quadrotor are not yet widely used in SAR operations. One reason is the need for an experienced operator, when using a quadrotor, to avoid crashes. Even simple tasks like holding a certain position and height are a challenge to inexperienced operators. This is especially true for indoor usage where, due to the turbulences produced by a medium sized quadrotor, disturbance correction requires high steering accuracy by the operator. Especially in high stress situations, an autonomous or semi-autonomous solution to support the operator in the task at hand is highly desired.

There are solutions that use multiple calibrated cameras and a base-station connected via remote signals to achieve a hold position task (for example Michal *et al.* [5]). And while this approach gives very precise results, it is unfortunately not viable in a SAR situation that requires a fast, or if possible zero, setup time. This makes an on board solution desirable. But if we desire an onboard vision analysis solution, we have to keep track of space, weight and energy consumption. These are all resources that are vital to the successful operation of a quadrotor and usually very sparse.

One possibility to solve this problem is to use a highly specialized integrated circuit that takes care of the navigational control of the quadrotor. Furthermore, to reduce the stress on the onboard battery, a low amount of energy efficient sensors would be required. One approach to keep the number of sensors low is the use of only one camera. While this monocular vision approach has been explored in the past, (*e.g.* by Yang, Scherer and Zell [11] or Tournier *et al.* [8]), using a low power consumption system on a chip device like a FPGA for onboard navigation and image analysis computation, instead of a general purpose microprocessor system or a base station, has yet to be fully explored.

The goal of this study therefore was to develop a zero-setup, vision analysis system via field programmable gate array (FPGA), using a single camera mounted on a quadrotor.

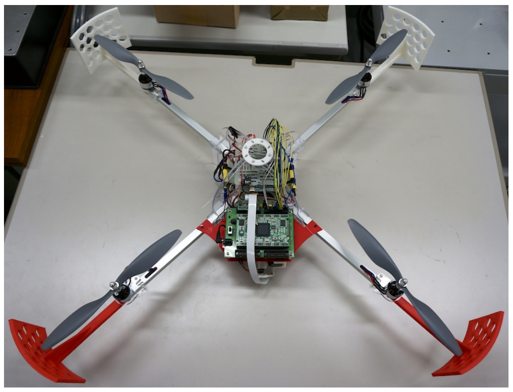


Figure 1. Picture of the test platform

## 2. Setup

The quadrotor used in this project is based on the open source project Arducopter [1]. The flight controller that is used, the Ardu Pilot Mega (APM), has an Atmega1280 as main processor and an Atmega328 co-processor to handle the RC interface processing. It is connected with an inertia measurement unit (IMU) shield. The sensors that are available on this shield consist of a 3-Axis accelerometer, z-gyro, xy-gyro, pressure sensor and a digital compass. We are using the standard Arducopter dome to protect the electronics and a custom designed 3d-printed ABS case to protect the quadrotor blades. The size of the quadrotor is 70x70cm and, with all added electronics and 3300 milliampere battery that grants around 15min flight time, measures in at 1,8kg (see Figure 1).

The firmware of the APM was modified to enable the transmission of roll and pitch position of the quadrotor from the APM via pulse-width modulation (PWM) to the FPGA.

The FPGA used in this research is a XC6SLX100 of the Xilinx Spartan 6 family. It is configured to have sixteen inputs and twenty-two outputs. We distinguish between four different input groups and four different outputs groups.

The first input group consists solely of the clock, which is running at 50MHz and is actually hardwired on the FPGA board. The second group contains the data signals we receive from the camera CMOS and other signals necessary for the picture acquisition. The third group is made up of pitch and roll information processed by the APM and send via PWM to the FPGA. The last group consists of the PWM signals received by the remote of the operator. They involve potential operator steering and correction signals.

The first of the four output groups that have been configured are the control signals necessary for the operation of the camera. The second group consists of the roll and pitch signals for the mini servos of the 2-axis camera mount. The third group involves the modified control signals as PWM. The last and final group is necessary to have a standard

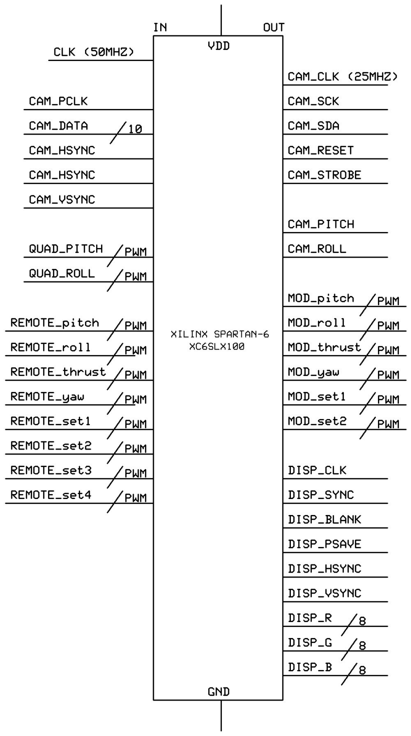


Figure 2. Schematic of the FPGA in- and outputs

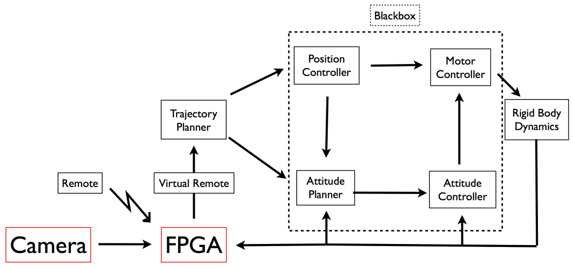


Figure 3. Signal diagram

60Hz VGA monitor output, but is not required for operation. Nevertheless they proved very valuable for debugging of the FPGA. A schematic of the FPGA in- and outputs can be seen in Figure 2 and a schematic how it is implemented in the overall system can be seen at Figure 3.

## 3. Approach

The initial task that was chosen to evaluate the use of an FPGA for image analysis and quadrotor control is the tracking of a round target of known size and color. The navigational task is to stay directly over this target in a certain flight height. The target chosen in this approach is a circular piece of red styrofoam with a 25cm diameter.

There is a fundamental difference between designing a system on a CPU or FPGA. One common limitation when

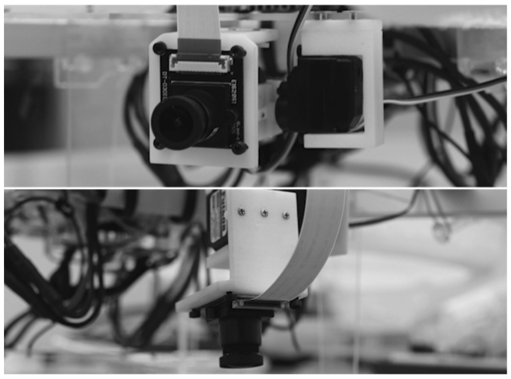


Figure 4. The camera mount in frontal (upper picture) and top-down (lower picture) position

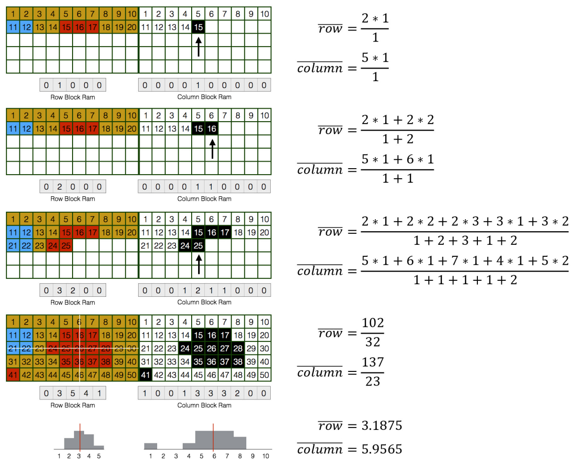
working with a CPU is the clocking speed. Since the processing tasks of a CPU are usually linear structured, the faster the CPU the more instructions we are able to process in a certain amount of time. Since the common task structure of a FPGA program is highly parallel, FPGAs can outperform CPUs with much higher clocking speed if specialized for a certain task (*e.g.* shown by Underwood [9]). This means the limiting factor of the FPGA is not the clocking speed, but the number of gates available on the FPGA. Therefore it is important to optimize the number of gates used as often as possible, especially when implementing a rather complex task as vision analysis.

So instead of performing an distortion correction by calculating the visual Jacobean(as explained by Mahony *et al.* [4]) via FPGA, it was therefore decided to solve the problem of perspective distortion of the projected camera image on a ground target, produced by roll and pitch movement of the quadrotor, by using a simple two-axis camera mount (Figure 4). The camera is driven by two mini servos controlled by the FPGA which is correcting the camera position by

$$\begin{bmatrix} cam_{pitch}(t) \\ cam_{roll}(t) \end{bmatrix} = \begin{bmatrix} k_{p\theta}(\theta_{error}(t)) + k_{d\theta}(\dot{\theta_{error}}(t)) \\ k_{p\phi}(\phi_{error}(t)) + k_{d\phi}(\dot{\phi_{error}}(t)) \end{bmatrix}, \tag{1}$$

where $k$ represents the specific gains and $\theta$, $\phi$ the pan-tilt angle. This is keeping the camera in a perpendicular position to the ground at all time, preventing perspective distortion. The gains in this formula have been experimentally obtained and can differ when a another frame design, motors or blades are used.

The image data is received as a 10bit YUV steam and converted in to a 24bit RGB stream. To reduce the limited amount of block memory used the initial resolution of the camera, 640x480 pixel, is down scaled to 320x240 pixel. To improve debugging and prototyping, the timing for the



$\overline{row} = \frac{2*1}{1}$
$\overline{column} = \frac{5*1}{1}$
$\overline{row} = \frac{2*1+2*2}{1+2}$
$\overline{column} = \frac{5*1+6*1}{1+1}$
$\overline{row} = \frac{2*1+2*2+2*3+3*1+3*2}{1+2+3+1+2}$
$\overline{column} = \frac{5*1+6*1+7*1+4*1+5*2}{1+1+1+1+2}$
$\overline{row} = \frac{102}{32}$
$\overline{column} = \frac{137}{23}$
$\overline{row} = 3.1875$
$\overline{column} = 5.9565$
Row Block Ram
Column Block Ram

Figure 5. A visualization of the the creation of a weighted average mean via FPGA. With this, the target position can be determined.

image analysis has been decided to be the same timing as the output signal, 25MHZ.

With every clock impulse, the RGB information of one pixel is written as 24 bit in block ram. After the end of each line of a frame, a computation to achieve target estimation via histogram analysis is initiated. The assumption is made, that the illumination of the scene is appropriate enough to have a clear, untainted sight on the tracking target. Since the color of the desired object is known beforehand, the euclidean distance of the pixel color is compared with the desired color:

$$dist_{RGB} = \left\| \begin{bmatrix} des_r \\ des_g \\ des_b \end{bmatrix} - \begin{bmatrix} pixel_r \\ pixel_g \\ pixel_b \end{bmatrix} \right\|. \tag{2}$$

If the pixel is in the desired color range, the pixel is marked as hit and furthermore the number of hits in the current row and the current column is raised by one. At the same time, the weighted arithmetic mean of the average target pixel position $\overline{u}$ and $\overline{v}$ is computed by:

$$\overline{u} = \frac{\sum_{i=1}^{x_{hits}} w_i x_i}{\sum_{i=1}^{x_{hits}} w_i}$$
$$\overline{v} = \frac{\sum_{i=1}^{y_{hits}} w_i y_i}{\sum_{i=1}^{y_{hits}} w_i}. \tag{3}$$

To enhance accuracy while still maintaining a one-pass approach, the weight $w_i$ is dependent on the number of hits already encountered in the corresponding row or column. This lowers the influence that noise and other outliers produce on the overall result. To further illustrate the whole process as it is processed by the FPGA, a simple 10x5 pixel frame sample is given in Figure 5.

After the processing of each pixel in the current frame, the weighted average mean is immediately computed. Since

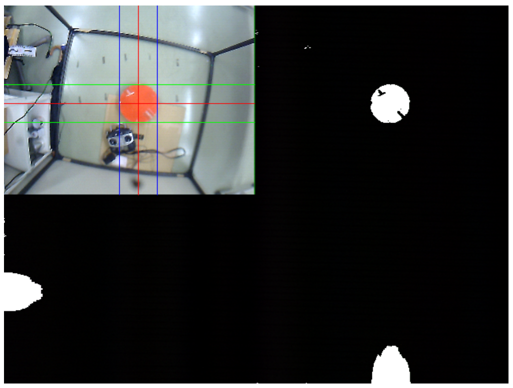


Figure 6. Debug output of the FPGA. The correct acquisition of the target position and size and the histograms are shown.

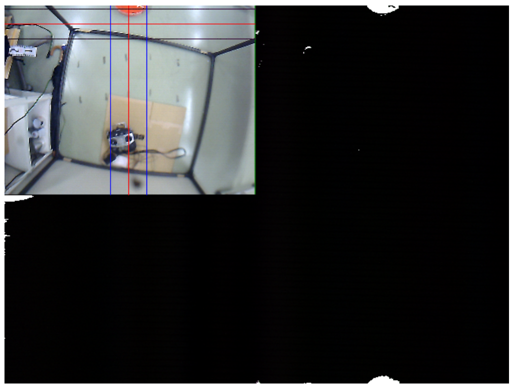


Figure 7. Debug output of the FPGA in case occlusion of the target by the image frame border. The correct size and last relative direction can still be computed.

in this process the number of pixels found in each row and column is also written to block ram, it is also possible to use this data to create a histogram as visualization (Figure 6). After the weighted arithmetic mean of the vertical and horizontal histogram is computed, it is assumed that we found the middle of the tracking target. Since a tracking target of circular shape is used, we can simply use horizontal and vertical Histogram $H_h$ and $H_v$

$$d = \frac{H_h(\overline{u}) + H_v(\overline{v})}{2}, \tag{4}$$

to also extract the estimated target diameter. To further improve accuracy, a mean of several lines around the horizontal and vertical histogram mean can be used.

After processing the estimated center and estimated size of the target, there are several checks to ensure a save tracking when the image analysis algorithm is used in-flight.

As a first step, the size of the acquired target is checked. If the acquired targets diameter is not reaching a size of at least 5 pixel in horizontal and vertical direction, it is assumed that the right target could not be found.

If the circular tracking target touches the edge of the picture frame, the assumption that the projected image area is circular does not hold true anymore. This leads to wrong results of the weighted mean average and target size estimation. To counter this problem it is assumed that the last known size of the target equals the current target size. This assumption can be made since normally in a stable and controlled indoor flight, sudden changes in z-direction are not desired. We also assume that we are able to get back to a position where the target is in the picture frame again, before significant changes in the z direction can happen. We can thus just ignore the faulty data in one dimension (the one were loss of information is expected) and only use knowledge about the last target size and the horizontal or respectively the vertical histogram data in addition to the knowledge which frame border is breached. This proves to be sufficient information to still compute a result that is good enough to move the quadrotor, and with this the projected camera image, back to a position that is able to catch the full target yet again (Figure 7).

To calculate the altitude of the quadrotor, we assume the camera lens as a thin lens and using the relationship between the beforehand known focal length $f$, the measured image distance $s'$, magnification $m$ and object distance $s$:

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'} \tag{5}$$

$$m = s'/s \tag{6}$$

$$s = f(1 + \frac{1}{m}). \tag{7}$$

Since object diameter $D$, object diameter in pixel $d$, pixel size on the image sensor $p$ and focal length $f$ are given we can calculate $m$ also by:

$$m = \frac{d \times p}{D}, \tag{8}$$

and get

$$\overline{z} = f(1 + \frac{D}{d \times p}), \tag{9}$$

thus giving us the distance of the camera towards the tracking target.

A PD-controller was implemented for controlling the x and y position of the quadrotor and a PID-controller for the control of height. The desired position in the picture frame is notated with $u_{des}$, $v_{des}$ and is located in its middle while the desired hight $z_{des}$ is controllable by the operator. We get the FPGA computed control signals $F$:

$$\begin{aligned} u_e &= u_{des} - \overline{u} \\ v_e &= v_{des} - \overline{v} \\ z_e &= z_{des} - \overline{z} \end{aligned} \tag{10}$$

$$\begin{bmatrix} F_{pitch}(t) \\ F_{roll}(t) \\ F_{thrust}(t) \end{bmatrix} =$$

$$\begin{bmatrix} k_{px}(u_e)(t) + k_{dx}\frac{d}{dt}(u_e)(t) \\ k_{py}(v_e)(t) + k_{dy}\frac{d}{dt}(v_e)(t) \\ k_{pz}(z_e)(t) + k_{iz}\int_0^t (z_e)(\tau)d\tau + k_{dz}\frac{d}{dt}(z_e)(t) \end{bmatrix}. \quad (11)$$

The gains $k$ here have also been experimentally obtained and, as the gains of the camera mount, can greatly differ depending on frame, blades, motors and load. The differential part of the controllers is gained by comparing the estimated positions of the current frame and the last sampled frame. The sampling rate for this process is the same timing as the image analysis part of the FPGA. In this case it is 25MHz. The $k_p$, $k_d$ and $k_i$ parameters were estimated experimentally and separately for pitch,roll and thrust signal.

The FPGA computed control signals $F$ are mixed together as PWM signals with the operator signals $O$ as:

$$\begin{bmatrix} mod\_pitch(t) \\ mod\_roll(t) \\ mod\_thrust(t) \end{bmatrix} = \begin{bmatrix} F_{pitch}(t) \\ F_{roll}(t) \\ F_{thr.}(t) \end{bmatrix} + \begin{bmatrix} O_{pitch}(t) \\ O_{roll}(t) \\ O_{thr.}(t) \end{bmatrix}. \quad (12)$$

The control signals $F$ are limited to a threshold lower than half of the maximum steering signal the operator is able to produce. This ensures a save flight, even if the image data should be temporarily misinterpreted. It gives the operator the possibility to override the FPGA created signals at any given time, given that a continuous misinterpretation of the image data will lead to a crash and possible damage of the hardware.

These control signals are then transmitted to the flight controllers trajectory planer, and converted to the corresponding motor speeds. Since we do not require knowledge of the motor speeds, most of the flight controller is treated as a black box. For the flight controller, there is no difference between a normal operator input and a FPGA processed input, so the FPGA is effectively acting as a virtual remote. The advantage of this approach is that the system is not only limited on this hardware (frame and flight controller), but can, with minimal adaptations, potentially be ported to other systems that are able to do a hovering flight and require pitch, roll and thrust signals.

## 4. Results

Our prove of concept prototype is able to generate and modulate operator signals according to the picture perceived by the onboard camera. This enables the quadrotor to do stable hover flight over a predetermined tracking target of known size as seen on Figure 8. We achieve a stable hover flight by using only a FPGA as control signal generator and a pan-tilt camera with an accuracy of around 20 cm. This has been measured by analyzing videos of the flight of

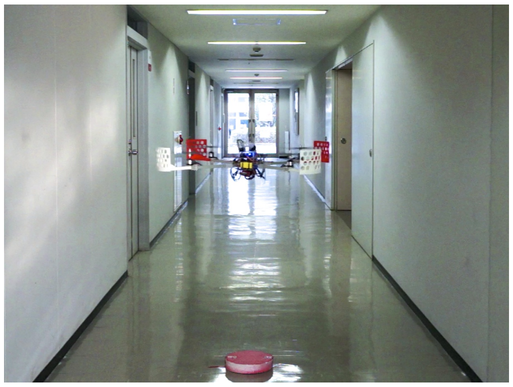


Figure 8. Successful test of the hold position and altitude task in a corridor

the quadrotor over multiple flights manually. The perceived tracking image is analyzed fast enough to counter chaotic air distortions that occur when a rather large quadrotor like this flies in a confined area like a corridor. In our case, the response time is dependent on the transport protocol used. Since the output of the calculated data as a PWM signal is initiated with 50Hz, the worst case response time (new output data computed on clock after the PWM output was initiated) is 20ms. A table of the specifications of the hardware used, response time we can achieve with this setup and detailed statistics of how much resources of the FPGA were used for this project can be seen in Table 1.

Since additionally the original operator signals are mixed with the FPGA signals, it is also possible to fly in an assisted semiautomatic flight modus. This enables also inexperienced operators to safely control the quadrotor even in confined areas. The operator is able to fly the quadrotor in a certain area of operation, surrounding the tracking target. If the operator chooses to stop the control, the quadrotor returns to its stable hovering position over the tracking target. In this scenario though, the possible area of operation is limited by the ground area visible by the camera, in other words the maximum possible flight height of the surrounding area.

A log of the control signals generated by the FPGA while tracking a target indoors can be seen in Figure 9.

## 5. Future Development

While the preliminary system works for the task it was optimized for, there is room for improvement. The top-down view of the camera proved very limiting for the area of operation and can also lead to a loss of the tracking target, if the flight height should be too low and thus being outside of the camera view angle at any given point in time. In the future we hope to tackle this problem with a tracking of more sophisticated image feature points. In an indoor flight scenario, the use of vanishing point estimation ,simi-

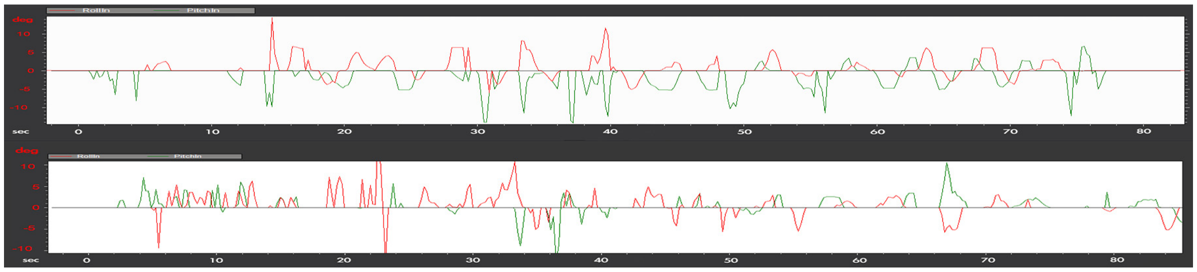


Figure 9. Roll (red) and pitch (green) control signals generated by the FPGA (upper graph) and by an operator (lower graph) while hovering over a tracking target. The FPGA generated control signals produce less spikes and more simultaneous pitch and roll corrections then a human operator.

| | |
|---|---|
| Resolution | 320x240px |
| Camera frame rate | 30fps |
| FPGA family | XILINX Spartan-6 |
| Model | XC6SLX100 |
| Number of occupied Slices | 24% |
| Number of RAMB16BWERs | 40% |
| Number of bonded IOBs | 21% |
| FPGA clock | 50MHz |
| Output frequency | 50Hz |
| Highest response time | 20ms |

Table 1. Major specifications of the currently implemented vision system

lar to Shi and Samarabandu [7], will be one of the elements that will be explored in a future research.

There are much higher timings for the image analysis process possible than the timings achieved in this approach. One of the limiting factor is the time is takes to capture a single image frame. If required, the use of a camera with higher frame rate is possible and it would impact the already established implementation only minimal while greatly improving the overall result.

To improve the reliability of this approach in regard to different illumination, it would be of benefit to use not the RGB space to classify the target, but to use a color space more indifferent to illumination (*e.g.* YUV).

One problem evaluating the approach has been that the ground truth of the quadrotors position in 3d space is not exactly measured. When exactly measured, it would be easier to show the feasibility of the approach in graphs.

Since the hardware has been prepared and has proven to be feasible, it is reasonable to assume that a future research will be build on the foundation that has been laid by this work.

## 6. Conclusion

With the successful implementation of the hold altitude and position tracking-task, it was shown that the utilization of FPGA in flying robotics applications is feasible. A hold altitude and position task was accomplished, generating control signals by only using a single camera mounted on a pan-tilt 2-axes mount and a FPGA for image analysis connected to a flight controller.

## References

[1] Arducopter-Project-Website. http://code.google.com/p/arducopter/. 2

[2] BitcrazeAB. http://www.bitcraze.se/crazyflie/. 1

[3] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(3):367–385, 2003. 1

[4] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *Robotics &amp; Automation Magazine, IEEE*, 19(3):20–32, 2012. 3

[5] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *Robotics &amp; Automation Magazine, IEEE*, 17(3):56–65, 2010. 1

[6] ParrotSA. http://ardrone2.parrot.com. 1

[7] W. Shi and J. Samarabandu. Corridor line detection for vision based indoor robot navigation. In *Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on*, pages 1988–1991. IEEE, 2006. 6

[8] G. P. Tournier, M. Valenti, J. P. How, and E. Feron. Estimation and control of a quadrotor vehicle using monocular vision and moire patterns. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 21–24, 2006. 1

[9] K. Underwood. Fpgas vs. cpus: trends in peak floating-point performance. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pages 171–180. ACM, 2004. 3

[10] P. L. Walter. The history of the accelerometer. *Sound and vibration*, 31(3):16–23, 1997. 1

[11] S. Yang, S. A. Scherer, and A. Zell. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1-4):499–515, 2013. 1