REAL-TIME ON-BOARD OBSTACLE AVOIDANCE FOR UAVS BASED ON EMBEDDED STEREO VISION

B. Ruf[a,b], S. Monka[a], M. Kollmann[a], M. Grinberg[a]

[a]Fraunhofer IOSB, Video Exploitation Systems, Karlsruhe, Germany -
{boitumelo.ruf, sebastian.monka, matthias.kollmann, michael.grinberg}@iosb.fraunhofer.de
[b]Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology,
Karlsruhe, Germany - {boitumelo.ruf}@kit.edu

Commission I, ICWG I/II

KEY WORDS: embedded stereo vision, obstacle detection, obstacle avoidance, disparity estimation, semi-global matching, real-time stereo processing

ABSTRACT:

In order to improve usability and safety, modern unmanned aerial vehicles (UAVs) are equipped with sensors to monitor the environment, such as laser-scanners and cameras. One important aspect in this monitoring process is to detect obstacles in the flight path in order to avoid collisions. Since a large number of consumer UAVs suffer from tight weight and power constraints, our work focuses on obstacle avoidance based on a lightweight stereo camera setup. We use disparity maps, which are computed from the camera images, to locate obstacles and to automatically steer the UAV around them. For disparity map computation we optimize the well-known semi-global matching (SGM) approach for the deployment on an embedded FPGA. The disparity maps are then converted into simpler representations, the so called U-/V-Maps, which are used for obstacle detection. Obstacle avoidance is based on a reactive approach which finds the shortest path around the obstacles as soon as they have a critical distance to the UAV. One of the fundamental goals of our work was the reduction of development costs by closing the gap between application development and hardware optimization. Hence, we aimed at using high-level synthesis (HLS) for porting our algorithms, which are written in C/C++, to the embedded FPGA. We evaluated our implementation of the disparity estimation on the KITTI Stereo 2015 benchmark. The integrity of the overall real-time reactive obstacle avoidance algorithm has been evaluated by using Hardware-in-the-Loop testing in conjunction with two flight simulators.

1. INTRODUCTION

In recent years, the use of unmanned aerial vehicles (UAVs) in different markets has increased significantly. Currently, most important markets are aerial video/photography, precision farming and surveillance/monitoring. Ongoing technical development, e.g. in size, endurance and usability, enables the use of UAVs in even more areas. First prototypes for delivering goods or even for transporting people are already available.

In order to improve usability and safety, UAVs are equipped with sensors to monitor the environment, such as laser-scanners and cameras. One important aspect in this monitoring process is to detect obstacles in the flight path in order to avoid collisions. With the increasing performance of state-of-the-art algorithms in combination with modern hardware, camera systems are typically more practical than laser-scanners in performing this task, especially in terms of costs, weight and power-consumption, inherent to most commercial off-the-shelf (COTS) UAVs.

The TULIPP[1] project (Kalb et al., 2016), funded by the European Commission under the H2020 Framework Programme for Research and Innovation, aims to remedy these restrictions by setting up an ecosystem for a low-power image processing platform, consisting of a exemplary hardware instantiation, an energy-aware tool chain (Sadek et al., 2018) and a high performance real-time operating system (Paolillo et al., 2015). The Xilinx Zynq Ultrascale+ was selected as a main processing unit for the hardware instantiation since its combination of an embedded quad-core CPU and a FPGA provides high performance at low power.

In order to validate the developed components, the project defines three use cases in three different domains. One of these use cases shall enhance the development of modern UAVs by providing a system for a real-time obstacle avoidance based on a lightweight and low-cost stereo camera setup. Hereby, the cameras are orientated in the direction of flight. We use disparity maps, which are computed from the camera images, to locate obstacles and to automatically steer the UAV around them. As in the other use cases of the TULIPP project, the focus of our work lies, among other things, on a user-friendly development and optimization of image processing algorithms for embedded hardware. The aim is to close the gap between algorithmic development and hardware optimization and to facilitate the reduction of time-to-market, development and rework costs.

The main contribution presented in this paper is a system for on-board obstacle avoidance that

- employs the semi-global matching (SGM) algorithm (Hirschmueller, 2008) that is synthesized from C/C++ code and is deployed on an embedded FPGA in order to compute disparity information from a stereo camera setup,
- performs a reactive obstacle avoidance based on the pre-computed depth information by calculating the shortest path

[1] http://tulipp.eu/

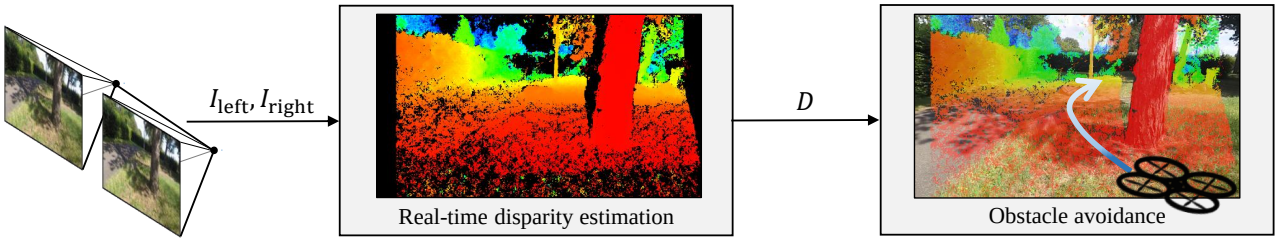


Figure 1. Overview of our system for real-time on-board obstacle avoidance based on embedded stereo. We use a stereo camera system to acquire two input images I_{left} and I_{right}. These are passed to the first processing stage, where a disparity map D is computed in real-time on an embedded FPGA. This disparity map is used by the subsequent step to compute and execute an evasion manoeuvre in order to avoid obstacles in the flight path.

around the obstacle and executing the evasion manoeuvre on the UAV, and

- runs in real-time on an embedded Xilinx Zynq Ultrascale+, which provides a good performance-to-watt ratio and is therefore a suitable choice when it comes to low-power real-time processing.

The paper is structured as follows: in Section 2 we give a brief overview of previous work done in the field of autonomous obstacle avoidance for UAVs and the advancements achieved in embedded stereo vision. This is followed by a detailed description of our approach in Section 3, which involves an algorithm of real-time embedded disparity estimation from stereo imagery and a system for obstacle avoidance based on the computed disparity information. In Section 4 we present the experimental results that were achieved. Section 5 concludes the paper by a short summary and an outlook.

2. RELATED WORK

In past years a substantial amount of work has been done in the field of navigating mobile robots securely through a known or unknown environment. Ground operating robots, or slow moving and heavy UAVs, typically use light detection and ranging (LiDAR) technology to conceive the surrounding environment (Bachrach et al., 2012). Scherer et al. (2008) used a heavy helicopter able to carry maximum payload of 29 kg and flying with speeds of up to 10 m/s. The LiDAR system has a range up to 150 m, which is suitable given the speeds and the mass of the system. However, for fast reactions in dynamic or cluttered scenes, heavy LiDAR systems are not practical. They consume significantly more power than a camera-based system, thus requiring a corresponding power supply, making them heavier and sluggish.

Due to higher speeds and more agile movements, UAVs need fast, light and energy-efficient solutions for robust obstacle avoidance. Systems relying on a single camera typically use optical flow information, computed between two consecutive frames to perceive the structure of the environment. The main advantage of such structure-from-motion approaches is that they are lightweight and energy-efficient and can therefore be used for small UAVs. Merrell and Lee (2005) proposed a probabilistic method of computing the optical flow for more robust obstacle detection. Ross et al. (2012) used a data driven optical flow approach learning the flight behaviour from a human expert. Nonetheless, since the information gained by a single camera system is limited, such systems are typically infeasible for robot navigation.

The use of active RGB-D systems, such as structured-light cameras or time-of-flight sensors, provides both colour images and per-pixel depth estimates. The richness of this data and the recent development of low-cost sensors make them very popular in mobile robotics research (Bachrach et al., 2012). However, such sensors suffer from a limited range or they require homogeneous light conditions, making them impractical for the use in outdoor environments.

In recent years the performance of image-based depth estimation algorithms has increased significantly. These achievements in conjunction with modern hardware, make passive RGB-D vision sensors, such as stereo camera systems feasible for indoor and outdoor environments. Depending on the baseline of the cameras, i.e. the distance at which they are mounted on the rig, and their image resolution, such sensor system can be used for close range as well as far range depth estimation (Gallup et al., 2008). Numerous studies have shown the strength of stereo vision w.r.t. on-board obstacle avoidance for UAVs (Barry et al., 2015; Oleynikova et al., 2015; Schmid et al., 2013).

When it comes to real-time stereo vision, the SGM approach (Hirschmueller, 2008) has emerged as one of the most widely used algorithm, especially for embedded systems. This is not only because of its accuracy and performance, but also due to the fact that it is well suited for parallelization on many architectures. Spangenberg et al. (2014) as well as Gehrig and Rabe (2010) optimized the SGM algorithm to run on a conventional CPU, reaching 12 and 14 fps respectively at a VGA image resolution. The utilization of general purpose computation on a GPU (GPGPU) can lead to faster processing speeds at higher image resolutions, as the work (Banz et al., 2011) and (Haller and Nedevschi, 2010) show.

Nonetheless, since GPGPU is typically not very energy-efficient, most adaptations of the SGM algorithm for embedded systems were done for the FPGA architecture. To mention a few, Barry et al. (2015) developed an FPGA based stereo vision sensor running with 120 fps at an image resolution of 320×240 pixels with the SGM algorithm. Hofmann et al. (2016) deployed the SGM algorithm on an Xilinx Zynq 7000 FPGA, achieving 32 fps on an image of 1280×720 pixels. Even frame rates of up to 197 fps can be achieved at a similar frame size depending on the performance of the FPGA, as (Li et al., 2017) shows.

However, in order to achieve such performance, a thorough optimization of the algorithm is required, which typically involves the use of hardware specific APIs such as VHDL or Verilog. In contrast, we aim at using high-level synthesis (HLS) for porting our algorithm, which is written in C/C++, to the FPGA. This is done

specifically because one of the fundamental goals of the TULIPP project is the reduction of development costs by closing the gap between application development and hardware optimization. In doing so, we accept some potential performance loss.

3. METHODOLOGY

The real-time on-board obstacle avoidance system is based on disparity images of a calibrated stereo camera rig, which is mounted onto a UAV and is orientated in the direction of flight. These disparity images, which encode the structure of the environment in front of the UAV, are then used to detect objects in the flight path and to avoid a collision in further steps. The processing is done on-board in real-time and can be partitioned into two parts as depicted in Figure 1, namely: real-time disparity estimation and obstacle avoidance. While our algorithm for disparity estimation is optimized for the execution on the FPGA, the detection of obstacles and the computation for the evasion manoeuvre is run on a general-purpose CPU. We use the Micro Air Vehicle Link (MAVLink) protocol to control the UAV and execute the evasion manoeuvre. In the following, both parts of our system will be discussed in more detail.

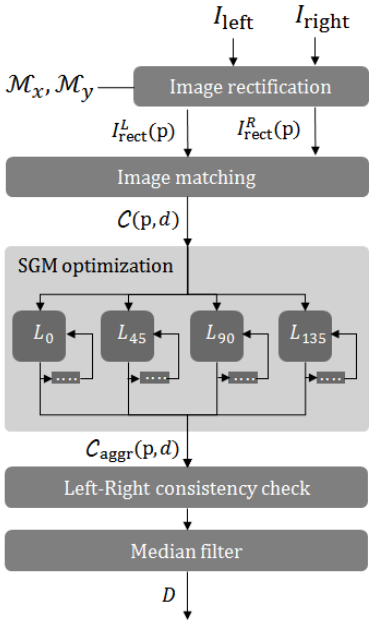


Figure 2. Schematic overview of our processing pipeline for real-time disparity estimation of a FPGA. Within the SGM optimization the costs are aggregated along 4 concentric paths, i.e. L_0 - horizontal from left to right, L_{45} - diagonal to bottom right, L_{90} - vertical from top to bottom, L_{135} - diagonal to bottom left. The aggregation costs are buffered in order to accumulate them while iterating of the image data.

3.1 Real-time disparity estimation

A stereo camera setup typically consists of two cameras which are mounted on a fixed rig with a baseline b and orientated in a way that their optical axes are aligned in parallel. Due to the slightly different viewing perspectives, a scene point M is projected onto different positions m_1 and m_2 in the images of both cameras. When the camera parameters are known, finding corresponding pixels in the images of the cameras allows to compute the depth of the corresponding points and to reconstruct their 3d positions. For correct scene reconstruction this has to be done for all pixels of the camera images.

In general, a correspondence search for a pixel in the image of one camera in the image of the second camera has to be done along the so called epipolar curve, which corresponds to the viewing ray of the first camera in the image of the second camera (epipolar constraint). If it is possible to compensate for the camera distortions, the curve becomes a straight line. Another simplification can be achieved through rectification of the images, i.e. transformation of both images onto a common image plane. As a result, the epipolar lines coincide and corresponding image points of both rectified images turn out to lie in the same image row. The displacement of the image points m_1 and m_2 of the scene point M, which is called disparity, is then defined as $d = |\mathrm{m}_{x1} - \mathrm{m}_{x2}|$. Together with the known calibration parameters of the stereo rig, the disparity allows to reconstruct depth of the corresponding scene point.

Finding pixel-wise matches between two input images and thus the corresponding disparities leads to a disparity image. Hereby, the correctness of the result greatly depends on the matching function and the optimization strategies that are used. The optimization strategies are typically categorized into local and global approaches. While local methods rely on a fixed neighbourhood around a central pixel, global methods optimize an energy function on the full image domain. Due to the finite window size, local methods are very efficient but often lack in accuracy (Scharstein and Szeliski, 2002).

In the past decade the proposed SGM approach (Hirschmueller, 2008) demonstrated a good trade-off between runtime and accuracy in numerous studies. It uses dynamic programming to optimize a global energy function by aggregation the cost of each pixel p given the disparity d_{p} along multiple one-dimensional paths that cover the complete image. Hereby, each path computation is an independent sub-problem, making SGM suitable for highly parallel execution on dedicated hardware. While it is initially proposed to use 16 concentric paths, most studies done on conventional hardware only use eight paths in order to avoid sub-pixel image access. However, due to the limited amount on memory on embedded FPGAs and their strengths in processing data streams, it is common practice to only use four paths when porting the algorithm onto such hardware (cf. Section 3.1.3). Studies show that a reduction from eight to four paths lead to a loss in accuracy of 1.7% while greatly increasing the performance of the algorithm (Banz et al., 2010).

We have adopted and optimized this approach for real-time image-based disparity estimation on embedded hardware. Our full algorithmic chain is comprised of the following steps:

- Image rectification
- Image matching
- SGM optimization
- Left-Right consistency check
- Median filter

These steps are pipelined in order to efficiently process the pixel stream produced by the cameras (cf. Figure 2). We use FIFO buffers to pass the data from one step to the next without using too much memory. Everything is implemented in C/C++ and then synthesized into VHDL code with HLS to run on the FPGA.

3.1.1 Image rectification Each pair of input images, for which the disparity estimation is to be done, needs to be rectified. However, the calibration parameters can be precomputed so that each image pair of the sequence is transformed in the same manner. We use a standard calibration routine (Zhang, 2000) to compute two rectification maps for each image $\mathcal{M}_\mathrm{x}, \mathcal{M}_\mathrm{y} \in \mathbb{N}_0^{W \times H}$, which hold for each pixel $\mathrm{p} = (x, y)$ in the rectified image I_rect the coordinates of the corresponding pixel in the unrectified image I: $I_\mathrm{rect}(\mathrm{p}) = I(\mathcal{M}_\mathrm{x}(\mathrm{p}), \mathcal{M}_\mathrm{y}(\mathrm{p}))$. We have implemented a line buffer to store $n = \mathrm{abs}\max(\mathcal{M}_\mathrm{y})$ lines of the input pixel stream in order to buffer enough pixel data to compute the rectified image.

3.1.2 Image matching In the next stage of the pipeline the rectified images $I^\mathrm{L}_\mathrm{rect}, I^\mathrm{R}_\mathrm{rect}$ of the two cameras are used to perform a pixel-wise image matching by computing a similarity score Φ between each pixel in both images given the expected disparity range with a maximum disparity d_max. This will generate a three dimensional cost volume $\mathcal{C} \in \mathbb{N}_0^{W \times H \times d_\mathrm{max}}$ in which each cell contains the pixel-wise matching cost for a given disparity d:

$$\begin{aligned} &\forall d \in \mathbb{N}_0, 0 \leq d < d_\mathrm{max} : \\ &\qquad \mathcal{C}(\mathrm{p}, d) = \Phi(I^\mathrm{L}_\mathrm{rect}(\mathrm{p}), I^\mathrm{R}_\mathrm{rect}((\mathrm{p}_x - d, \mathrm{p}_y)). \end{aligned} \tag{1}$$

Due to the rectification done in the previous step, the search space for similar pixels in both images is reduced to the scan lines of the images. A relative translation of the second camera to the right, results in a displacement of the corresponding pixel to the left of the reference pixel. Hence, the occurrence of a pixel correspondence in the right image is restricted to the left side of the original position of the reference pixel in the left image. Since the pixel data of both images are processed as streams, $m = d_\mathrm{max}$ pixels of the right image have to be buffered in order to allow a search for similar pixels in already processed data.

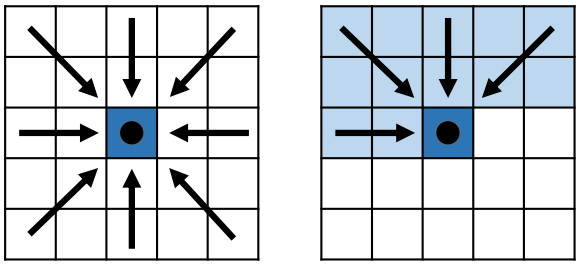
There exist a numerous number of appropriate cost functions to measure the similarity between two pixels, which work on different information a pixel can provide. We have employed a simple pixel-wise sum of absolute differences (SAD), as well as the non-parametric Hamming distance of the Census Transform (CT) (Zabih and Woodfill, 1994), due to its popularity in real-world scenarios. Again, due to the local support regions, these cost function require the use of line buffers in order to use pixel data in the local neighbourhood.

3.1.3 SGM optimization The previously computed cost volume $\mathcal{C}$ holds the plausibilities of disparity hypotheses for each pixel p in the left image. This could already be used to extract a disparity image by finding the winner-takes-it-all (WTA) solution for each p, i.e. the disparity with the minimum costs. However, due to ambiguities in the matching process this would not generate a suitable result. Therefore an optimizations strategy is typically applied to regularize the cost volume $\mathcal{C}$ and remove most of the ambiguities. As already stated, we use the SGM optimization scheme, due to its suitability in parallelization and hence, its suitability for use on embedded hardware.

In its initial formulation the SGM approach optimizes an energy function by aggregating the cost for each pixel p in $\mathcal{C}$ given disparity d_p along concentric paths that center in p. As stated by Equation (2), the costs of each neighbourhood pixel $\mathrm{q} \in \mathcal{N}_\mathrm{p}$ along the aggregation paths is penalized with a small penalty P_1 if its disparity d_q differs by one pixel from the disparity d_p. If the disparity change between d_p and d_q is larger than ± 1, a greater penalty P_2 is added to the cost of pixel q. The cost aggregation can efficiently be computed by traversing along each aggregation path, beginning at the image border, successively accumulating the costs and storing the cost for each pixel p in the aggregated cost volume $\mathcal{C}_\mathrm{aggr}$:

$$\begin{aligned} \mathcal{C}_\mathrm{aggr}(\mathrm{p}, d_\mathrm{p}) &= \mathcal{C}(\mathrm{p}, d_\mathrm{p}) \\ &\quad + \sum_{\mathrm{q} \in \mathcal{N}_\mathrm{p}} \begin{cases} \mathcal{C}(\mathrm{q}, d_\mathrm{q}) + P_1, & \text{if } |d_\mathrm{p} - d_\mathrm{q}| = 1 \\ \mathcal{C}(\mathrm{q}, d_\mathrm{q}) + P_2, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

As the data of the cost volume $\mathcal{C}$ is streamed through this processing stage, the amount of aggregation paths is reduced from eight to four paths as depicted in Figure 3.



(a) (b)

Figure 3. Eight vs. four path aggregation. While the use of eight paths (a) require full image access for each pixel being aggregated, the use of only four paths (b) can efficiently be computed while streaming the pixel data and accumulating previously computed costs (light blue).

The SGM aggregation can efficiently be computed by simply buffering and accumulating the already computed costs. While the horizontal path only requires to buffer d_max values from the previous pixel, the vertical and diagonal paths require to buffer $3 \times d_\mathrm{max}$ for each pixel of the previous row. The use of eight paths would require a second pass from back to front. In order to avoid an overflow in the cost aggregation, the minimal path costs are subtracted after each processed pixel, as stated by Hirschmueller (2008).

From the aggregated cost volume $\mathcal{C}_\mathrm{aggr}$ the resulting disparity image D can easily be extracted by finding the pixel-wise WTA solution $\hat{d}_\mathrm{p}$, i.e. $D(\mathrm{p}) = \arg\min_d \mathcal{C}_\mathrm{aggr}(\mathrm{p}, d)$

3.1.4 Left-Right consistency check Typical for stereo imagery occluded areas are not the same for both cameras due to different viewing perspectives onto the scene. Hence, it is not possible to compute disparities for areas which are occluded at least in one of the two cameras.

In order to remove outliers, a common approach is to perform a left-right consistency check. As the name implies a disparity map D_left corresponding to the left image is compared to a disparity map D_right corresponding to the right image. For all disparities $d_\mathrm{left} \in D_\mathrm{left}$ the corresponding disparity in D_right is evaluated. If both disparities do not coincide, i.e. $|D_\mathrm{left}(\mathrm{p}_x, \mathrm{p}_y) - D_\mathrm{right}(\mathrm{p}_x + d_\mathrm{left}, \mathrm{p}_y)| \leq 1$, the corresponding pixel in D_left is invalidated. This process typically requires the computation of an additional disparity map. However,

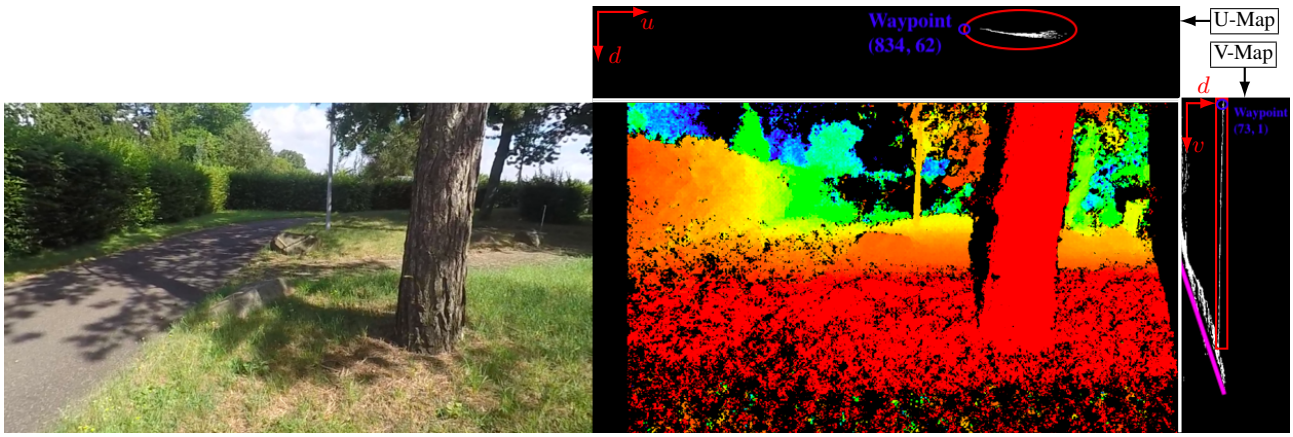


Figure 4. Illustration of the U-/V-map. The image on the left shows the reference image from the left camera of the stereo system. On the right the coloured disparity image is depicted, together with the U-map on top and the V-map on the right. The depths in the disparity image are colour coded, going from red (near) to blue (far). The white contour of the U-map represents the tree and its width (red). The V-map encodes ground plane (highlighted in pink) and the height of the tree (red). Furthermore the U-/V-map depict the computed waypoint.

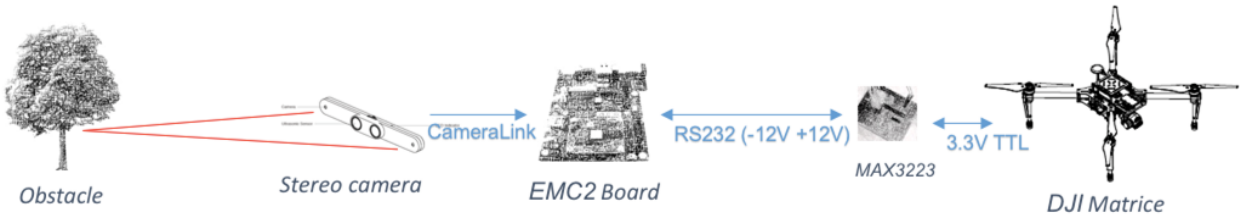


Figure 5. Schematic overview depicting interoperability of our approach with the DJI Matrice 100 UAV system.

D_{right} can be efficiently approximated by reusing the previously computed $\mathcal{C}_{\text{aggr}}$ to compute the WTA disparities: $D_{\text{right}}(\text{p}) = \arg\min_d \mathcal{C}_{\text{aggr}}((\text{p}_x + d, \text{p}_y), d)$.

3.1.5 Median filter A final median filtering is employed in order to remover further outliers. Similar to the cost functions, the implementation of a $k \times k$ median filter on FPGAs requires k line buffers.

3.2 Obstacle Avoidance

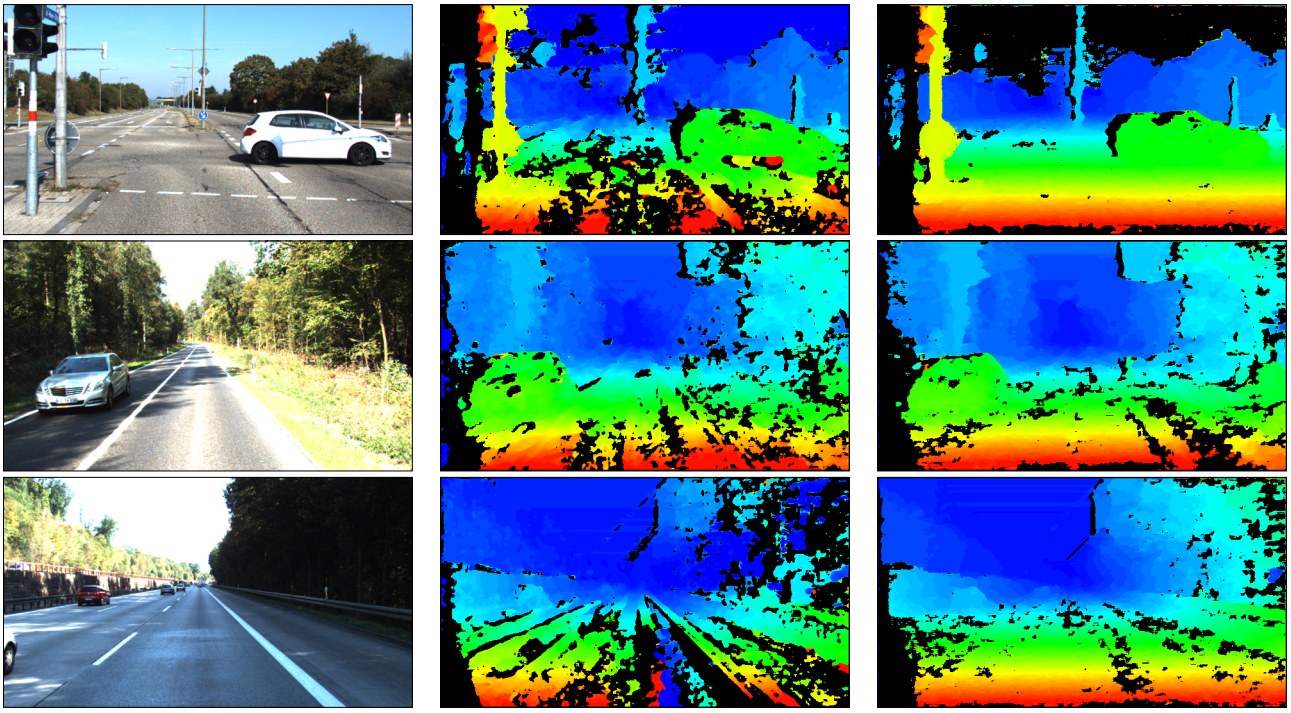
The second part of our system aims at the actual obstacle avoidance based on the previously computed data. The disparity map is used to estimate the depth of the depicted objects and detect obstacles in the flight path. Given that an obstacle is detected, our algorithm computes the shortest route around the object and manoeuvres the UAV around it. In the field of autonomous navigation there has been a lot of work for avoiding obstacles and navigating securely through an known or unknown environment. Nevertheless, the TULIPP use-case aims at fast reaction times by keeping the weight and energy consumption as low as possible.

Our approach is therefore based on a reactive obstacle avoidance. The disparity information of the stereo images is used to build U-/V-Maps of the environment (Labayrade et al., 2002; Li and Ruichek, 2014; Oleynikova et al., 2015). Hereby, the disparity image is transformed into a simpler representation in order to reduce complexity, to improve the robustness regarding inaccuracies in the measurements, and to simplify the obstacle detection. For every column of the disparity image a histogram of disparity occurrences is computed resulting in a map of size $W \times d_{\max}$ (cf. Figure 4). This so called U-Map encodes the depth and width of each object in the disparity map and can be interpreted as a bird’s eye view on the scene. Analogous to the U-Map, a V-Map of size $d_{\max} \times H$ is computed by creating a histogram of disparity occurrences for each row of the disparity image. The V-Map reveals the ground plane, highlighted by the pink line in Figure 4, as well as the height of the objects at a given disparity.

By applying a threshold filtering to the U-/V-Maps we transform them into binary maps, hereby suppressing uncertainties and revealing prominent objects. We further filter the results by applying dilatation and a Gaussian filtering to the maps. To be independent from the cluttered horizon only a region of interest is considered. In the next step we extract the largest contours together with their centroids from the U-/V-Maps analogous to the approach in (Suzuki and Abe, 1985). Then we perform an abstraction step by drawing ellipses in the binarized U-Map and rectangles in the binarized V-Map around the extracted contours as shown in Figure 4. This corresponds to simplified cylindrical object models of the obstacles in 3d space. If the obstacles have a critical distance to the UAV, or algorithm tries finds the shortest path to the left, right, up or down. The algorithm is executed periodically in order to exclude false positive detection. A wrong decision due to a false detection in one frame can be corrected in the next frame.

4. EXPERIMENTS

We implemented and deployed our system for image-based obstacle avoidance on an embedded Xilinx Zynq Ultrascale+ with an ARM Cortex-A53 quad-core CPU and a FinFET+ FPGA. It is operated on the EMC^2-DP carrier board from Sundance. As input we use two industrial cameras from Sentech which are connected by CameraLink to the carrier board. Our system uses the



(a) Refence image (b) Disparity image produced by using SAD (c) Disparity image produced by using CT

Figure 6. Qualitative results achieved by our algorithm for disparity estimation on the KITTI Stereo 2015 Benchmark (Menze and Geiger, 2015). The disparity is colour coded, going from red (large disparity, near) to blue (small disparity, far). The black areas correspond to undefined disparity areas.

MAVLink protocol to control the UAV and execute the evasion manoeuvre. We use the serial port to connect the carrier board to the flight controller of the UAV (cf. Figure 5).

As already stated, the disparity estimation algorithm is run on the FPGA, while the obstacle avoidance is processed on quad-core CPU. Table 1 shows runtime measurements of different parts of our system. A thorough description on the experiments done and the results achieved is given in the following two sections.

Table 1. Runtime measurements of the proposed system achieved on a Xilinx Zynq Ultrascale+.

Operation	Time (s)	Average time (s)
Disparity map	0.0314 - 0.0361	0.0345
Obstacle avoidance	0.0027 - 0.0110	0.0042
Σ	0.0341 - 0.0471	0.0387

4.1 Real-time disparity estimation

We have used Xilinx Vivado HLS to synthesize our algorithm from C/C++ into VHDL code and deployed on the FPGA running at 200 MHz. The FPGA of the Ultrascale+ provides 154 k logic cells. Table 2 holds figures on how many resources our algorithm utilizes on the Ultrascale+. We have configured our algorithm to use a 5×5 support region for the SAD and CT similarity measure, as well as for the final median filtering. The disparity sampling range is set to $d = [0, 60)$. The SGM penalties were set to $P_1 = 200$ and $P_2 = 800$ for the sum of absolute differences cost function, and to $P_1 = 8$ and $P_2 = 32$ when using the Hamming distance of the Census Transform. Our full pipeline achieves an average processing speed of 29 Hz at a frame size of 640×360 pixels. The latency of our synthesized code is calculated to be 28.5 ms. Experiments have shown that a reconfiguration of the FPGA to run at only 100 MHz leads, to a processing speed of approx. 20 Hz and a latency of 53.2 ms.

Table 2. Resource utilization with respect to the Zynq Ultrascale+ of the HLS code for disparity estimation.

Resource	Used	Available	Utilization
DSP48E	22	360	6 %
BRAM_18K	132	432	30 %
FF	12561	141120	8 %
LUT	27063	70560	38 %

To quantitatively evaluate the results achieved, we have adapted our processing pipeline to read pre-recorded and rectified imagery from memory. We have used the KITTI Stereo 2015 benchmark (Menze and Geiger, 2015) for evaluation, cropping a region of interest (RoI) of size 640×360 in order to not alter the algorithm. Figure 6 shows exemplary results achieved by our algorithm. While the middle column (Figure 6(b)) contains the results achieved when using the SAD similarity measure, the results in the right column (Figure 6(c)) reveal that a use of the CT allows a more robust matching between the input images and thus yielding results that are more accurate and hold less pixels that are invalidated due to inconsistencies in the left-right check. These observations are justified by the contents of Table 3, holding the quantitative measurements w.r.t. to the benchmark. The performance of our algorithm on subsequence of the benchmark is given at: `https://youtu.be/gzIFqUmqM7g`.

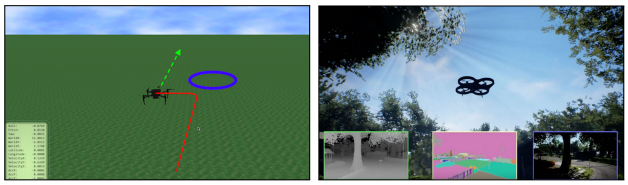
4.2 Obstacle Avoidance

We integrated the obstacle avoidance system into two different UAV systems, namely a Pixhawk based system and the the DJI

Table 3. Quantitative results achieved by our algorithm for disparity estimation w.r.t. to the KITTI Stereo 2015 benchmark, depending on the similarity measure used. A pixel is considered to be estimated correctly if the disparity < 3 px w.r.t. to the ground truth. The density specifies percentage of pixels in the resulting disparity image for which the disparity was computed.

Cost function	Density	Correct pixels
SAD	64.1 %	76.0 %
CT	74.2 %	95.4 %

Matrice 100 as shown in Figure 5. To validate the integrity of our algorithm we used Hardware-in-the-Loop testing in conjunction with a UAV flight simulators from DJI and Microsoft AirSim (Shah et al., 2018), as shown in Figure 7. Both flight controller were connected through their USB interface to the host PC running the flight simulator. With the Hardware-in-the-Loop connection the application for the UAV is the same as in real environment, but instead of controlling the motors the flight of the UAV is simulated.



(a) Screenshot of the DJI flight simulator software (b) Screenshot of the Microsoft AirSim simulator software

Figure 7. The two flight simulators used for Hardware-in-the-Loop testing. While the main view in (b) shows a third-person view point on the scene, the three smaller displays at the bottom reveal the disparity map, an object segmentation and camera image from the UAV point of view.

For the DJI based system we used the proprietary flight simulator, which is very minimalistic but provides all functionality we need. The tests were executed in autonomous flight mode. Hereby the UAV takes off to a height of one meter and accelerates in forward direction. If there is an object in the flight path, the UAV finds the shortest path to the edge of the object and accelerates to the left, right, up or down. If the object has moved out of the field of view the UAV starts accelerating in forward direction again. This setup allowed us to test the reaction of the our algorithm w.r.t. to incoming obstacles. We provide a visual demonstration of the simulation at: `https://youtu.be/pMDMTUCVwKc`

For further improvement of the algorithm for obstacle avoidance it is necessary to use appropriate simulation software. Therefore we changed to the Pixhawk flight controller which is supported by the Microsoft AirSim simulator. The simulator is based on the Unreal Engine which provides a photo-realistic environment. It allows to generate depth maps of the environment which allowed us to test the performance of our obstacle avoidance algorithm directly. However, these depth maps are synthetically generated and thus do not resemble realistic results achieved by our disparity estimation in a real world scenario. The AirSim simulator allows to thoroughly test and analyse the behaviour of our obstacle avoidance in different scenarios.

5. CONCLUSION & FUTURE WORK

In this paper we present a system for real-time on-board obstacle avoidance for UAVs based on an embedded stereo vision approach. It uses imagery from a stereo camera system for a real-time disparity map estimation of the environment in front of the UAV, which is then used for the adaptation of the flight path.

The algorithm for image-based disparity estimation, which adopts the semi-global matching approach, is optimized for the deployment on an embedded FPGA. Hereby the strategy for optimizing the SGM energy function is adapted in order to account for the strengths of FPGAs to process data streams. This is done, among other, by using four aggregation paths rather than the commonly used eight. It runs on the FPGA, which is operating at 200 MHz and achieves a processing speed of 29 Hz and a latency of 28.5 ms at a frame size of 640×360. This is significantly less powerful than state-of-the-art systems, however, since this work is part for the TULIPP project, which is aiming, amongst other things, at reducing the gap between application development and hardware optimization, we focus on a user friendly development by implementing the algorithm in C/C++ and porting it with the use of high-level synthesis to the FPGA.

The second part of our system transforms the previously computed disparity map into simpler representations, the so called U-/V-Maps, which are used for obstacle detection. Having found an obstacle in the flight path, the algorithm computes the shortest path and manoeuvres the UAV around it. Due to the low computational complexity of our reactive obstacle avoidance algorithm, it can be deployed on the embedded CPU without any major performance loss.

We deployed our system on an embedded Xilinx Zynq Ultrascale+ with an ARM Cortex-A53 quad-core CPU and a FinFET+ FPGA.

When computing disparity maps we compared two commonly used similarity measures for image matching, namely the sum of absolute differences and the Hamming distance of the Census Transform and found out that CT outperforms the SAD. The accuracy of the algorithm for disparity estimation was evaluated on the KITTI Stereo 2015 benchmark achieving 95.4% of correctly estimated pixels at a density of 74.2% w.r.t. to the groundtruth when using the CT as the similarity measure. The integrity of the reactive obstacle avoidance algorithm is evaluated by using Hardware-in-the-Loop testing in conjunction with two flight simulators.

Our current work is focused on carrying out experiments in the real-world environment with the EMC^2-DP carrier board, together with the Ultrascale+ and the camera system mounted to the UAV. In parallel, we plan to evaluate our stereo algorithm w.r.t. energy consumption on different embedded hardware architectures, namely FPGA, CPU and GPU.

ACKNOWLEDGEMENTS

The TULIPP project is funded by European Commission under the H2020 Framework Programme for Research and Innovation under grant agreement No 688403.

REFERENCES

Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A. S., Krainin, M., Maturana, D., Fox, D. and Roy, N., 2012. Estimation, planning, and mapping for autonomous flight using an RGB-d camera in GPS-denied environments. *The International Journal of Robotics Research* 31(11), pp. 1320–1343.

Banz, C., Blume, H. and Pirsch, P., 2011. Real-time semi-global matching disparity estimation on the GPU. In: *Proc. IEEE International Conference on Computer Vision Workshops*, pp. 514–521.

Banz, C., Hesselbarth, S., Flatt, H., Blume, H. and Pirsch, P., 2010. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In: *Proc. International Conference on Embedded Computer Systems*, pp. 93–101.

Barry, A. J., Oleynikova, H., Honegger, D., Pollefeys, M. and Tedrake, R., 2015. Fpga vs pushbroom stereo vision for UAVs. In: *IROS Workshop on Vision-based Control and Navigation of Small Lightweight UAVs*.

Gallup, D., Frahm, J.-M., Mordohai, P. and Pollefeys, M., 2008. Variable baseline/resolution stereo. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Gehrig, S. K. and Rabe, C., 2010. Real-time semi-global matching on the CPU. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 85–92.

Haller, I. and Nedevschi, S., 2010. GPU optimization of the sgm stereo algorithm. In: *Proc. IEEE International Conference on Intelligent Computer Communication and Processing*, pp. 197–202.

Hirschmueller, H., 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30(2), pp. 328–341.

Hofmann, J., Korinth, J. and Koch, A., 2016. A scalable high-performance hardware architecture for real-time stereo vision by semi-global matching. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 27–35.

Kalb, T., Kalms, L., Ghringer, D., Pons, C., Marty, F., Muddukrishna, A., Jahre, M., Kjeldsberg, P. G., Ruf, B., Schuchert, T., Tchouchenkov, I., Ehrenstrahle, C., Christensen, F., Paolillo, A., Lemer, C., Bernard, G., Duhem, F. and Millet, P., 2016. TULIPP: Towards ubiquitous low-power image processing platforms. In: *Proc. International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pp. 306–311.

Labayrade, R., Aubert, D. and Tarel, J. P., 2002. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In: *Proc. IEEE Intelligent Vehicle Symposium*, Vol. 2, pp. 646–651 vol.2.

Li, Y. and Ruichek, Y., 2014. Occupancy grid mapping in urban environments from a moving on-board stereo-vision system. *Sensors* 14(6), pp. 10454–10478.

Li, Y., Li, Z., Yang, C., Zhong, W. and Chen, S., 2017. High throughput hardware architecture for accurate semi-global matching. *Integration*.

Menze, M. and Geiger, A., 2015. Object scene flow for autonomous vehicles. In: *Proc. Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070.

Merrell, P. C. and Lee, D.-J., 2005. Structure from motion using optical flow probability distributions. In: *Proc. SPIE 5803, Intelligent Computing: Theory and Applications III*, SPIE, pp. 5803:1–10.

Oleynikova, H., Honegger, D. and Pollefeys, M., 2015. Reactive avoidance using embedded stereo vision for mav flight. In: *Proc. IEEE International Conference on Robotics and Automation*, pp. 50–56.

Paolillo, A., Desenfans, O., Svoboda, V., Goossens, J. and Rodriguez, B., 2015. A new configurable and parallel embedded real-time micro-kernel for multi-core platforms. In: *Proc. Operating Systems Platforms for Embedded Real-Time applications*, pp. 25–27.

Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A. and Hebert, M., 2012. Learning monocular reactive UAV control in cluttered natural environments. *arXiv preprint arXiv:1211.1690*.

Sadek, A., Muddukrishna, A., Kalms, L., Djupdal, A., Podlubne, A., Paolillo, A., Goehringer, D. and Jahre, M., 2018. Supporting utilities for heterogeneous embedded image processing platforms (sthem): An overview. In: *Proc. Applied Reconfigurable Computing. Architectures, Tools, and Applications*, Springer International Publishing, Cham, pp. 737–749.

Scharstein, D. and Szeliski, R., 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* 47(1-3), pp. 7–42.

Scherer, S., Singh, S., Chamberlain, L. and Elgersma, M., 2008. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research* 27(5), pp. 549–574.

Schmid, K., Tomic, T., Ruess, F., Hirschmueller, H. and Suppa, M., 2013. Stereo vision based indoor/outdoor navigation for flying robots. In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 3955–3962.

Shah, S., Dey, D., Lovett, C. and Kapoor, A., 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: *Proc. Field and service robotics*, pp. 621–635.

Spangenberg, R., Langner, T., Adfeldt, S. and Rojas, R., 2014. Large scale semi-global matching on the CPU. In: *Proc. IEEE Intelligent Vehicles Symposium*, pp. 195–201.

Suzuki, S. and Abe, K., 1985. New fusion operations for digitized binary images and their applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7(6), pp. 638–651.

Zabih, R. and Woodfill, J., 1994. Non-parametric local transforms for computing visual correspondence. In: *Proc. European Conference on Computer Vision*, pp. 151–158.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, pp. 1330–1334.