Embedded Real-time Multi-Baseline Stereo

Dominik Honegger, Torsten Sattler and Marc Pollefeys

Abstract— Dense depth map estimation from stereo cameras has many applications in robotic vision, e.g., obstacle detection, especially when performed in real-time. The range in which depth values can be accurately estimated is usually limited for two-camera stereo setups due to the fixed baseline between the cameras. In addition, two-camera setups suffer from wrong depth estimates caused by local minima in the matching cost functions. Both problems can be alleviated by adding more cameras as this creates multiple baselines of different lengths and since multi-image matching leads to unique minima. However, using more cameras usually comes at an increase in run-time. In this paper, we present a novel embedded system for multi-baseline stereo. By exploiting the parallelization capabilities within FPGAs, we are able to estimate a depth map from multiple cameras in real-time. We show that our approach requires only little more power and weight compared to a two-camera stereo system. At the same time, we show that our system produces significantly better depth maps and is able to handle occlusion of some cameras, resulting in the redundancy typically desired for autonomous vehicles. Our system is small in size and lightweight and can be employed even on a MAV platform with very strict power, weight, and size requirements.

I. INTRODUCTION

Depth perception is a key ability for all autonomous vehicles such as Micro Aerial Vehicles (MAVs) or self-driving cars and is used for obstacle detection [1]–[3], visual navigation [4], and 3D mapping [5].

Cameras are important sensors in robotics due to their ability to work under a wide range of illumination and weather condition while being inexpensive, lightweight, and consume only little power.

As such, depth perception via stereo vision methods suits best to autonomous vehicles such as MAVs whose payload and power resources are limited. The most widely used stereo setup consists of two cameras. Its drawback is that the fixed baseline between the cameras limits the range in which the scene depth can be estimated accurately. Multi-camera systems offer multiple baselines and thus a better depth accuracy over a wider range of depth values [6], [7]. However, they are less common in practice as processing more images usually results in a higher computational load and thus also a higher energy consumption. In this paper, we therefore consider the problem of implementing real-time multi-camera stereo via embedded hardware.

Stereo vision algorithms estimate a depth or disparity for each pixel in an image by matching a patch around this pixel against one or more other images. Given the position of the best matching pixel in another image and the baseline between the two cameras, the depth for the pixel can be estimated via triangulation. Since larger patches lead to problems with non-planar regions and are computationally less efficient, small patches are typically used for stereo matching. The drawback of using smaller patches is that they are less discriminative. Consequently, matching algorithms are sensitive to disturbances such as image noise, occlusions or repetitive patterns. These result in local minima for the matching costs, which in turn cause wrong disparity value estimates. Depending on the underlying algorithm, these wrong values can be detected as invalid matches and filtered out of the final disparity map. The resulting holes in the depth maps can then be filled by propagating depth estimates, e.g., by applying semi-global matching (SGM) [8].

It has been shown that adding more cameras to the stereo system resolves the ambiguities that result in wrong disparity estimates and leads to unique global minima in the matching cost function [6]. Consequently, multi-camera stereo systems produce depth maps of a higher quality. Such systems offer baselines of different lengths that can be used to accurately triangulate objects both close to the camera and farther away. In contrast, two-camera setups usually have a rather limited range in which depth values can be estimated with high precision. In addition, it is possible to resolve small occlusions as some of the cameras may look around the obstacle and may thus capture parts of the scene occluded in the other images.

In this paper we present a novel embedded real-time stereo system that uses four cameras for depth perception. Our approach is able to harness the advantages of a multi-camera setup while maintaining the run-time and power efficiency of classical stereo systems. At the same time, our system is light enough to be used for any kind of mobile robot. Compared to a two-camera stereo approach [9], it produces significantly more complete and accurate depth maps, even without post-processing, without a significant overhead in run-time or energy consumption. In detail, we make the following contributions: i) we present a system for multi-baseline stereo that runs in real-time on constrained platforms. To the best of our knowledge, ours is the first such system. ii) we detail our efficient embedded implementation on a FPGA, enabling other researchers to re-implement our system. iii) we experimentally demonstrate the advantages of our approach in terms of depth map quality and the handling of occlusions compared to standard two-camera stereo. We show results of our approach in indoor and outdoor scenarios.
II. RELATED WORK

While the first publications on depth estimation from stereo have been released decades ago, it is still a very active area of research in both robotics [1], [9]–[11] and computer vision [12]–[14]. While the classical stereo setup involves only two cameras, there are multiple advantages to using more cameras: As part of searching for matching pixels along an epipolar line, the cost functions of two-camera setups usually have multiple minima. Okutomi and Kanade show that this is not the case if multiple cameras are used as such setups exhibit an unique and well-defined minimum of the cost volume at the correct depth [6]. Consequently, using more cameras leads to more accurate depth maps containing, a fact that is exploited in by state-of-the-art multi-view stereo methods [12], [13], [15]. For classical two-camera stereo approaches, the error in depth increases quadratically with the distance to the camera for two-camera setups. Gallup et al. show that for systems with multiple cameras, and thus multiple baselines, it is possible to obtain a constant error over the full depth range by selecting the cameras and the image resolution used for matching depending on the depth [7]. These two advantages motivated the development of our multi-baseline stereo system. Typically computer vision approaches to multi-view stereo focus on the accuracy of the resulting depths maps [12], [13], [15]. As such, they often do not meet the strict real-time constraint of robotic applications.

Yang and Pollefeys were among the first to show that consumer grade GPUs can be used for real-time multi-resolution stereo [16], an approach that has been extended to mobile devices [14]. However, GPUs usually require significant amounts of power, rendering GPU-based methods unsuitable for small robots, e.g. MAVs, that have strict limits on both power consumption and weight. Besides using an external server for dense mapping [5], multiple approaches for power-efficient stereo have been developed. Instead of computing a full depth map in every frame, Barry and Tedrake propose to only consider a single depth [1], for which stereo match is extremely efficient. As the stereo system moves through the scene and objects move into this depth, a complete map of the scene can be recovered via this pushbroom stereo approach. Pillai et al. propose an iterative and tunable stereo approach that can be computed at high frame-rates on a single CPU thread [11]. Their method initially computes a sparse set of 3D points, which are used to approximate the scene via a (coarse) triangle mesh. This coarse prior is then iteratively refined, where the level of refinement controls both the density of the estimated depth maps and the computational costs. An alternative to using programmable GPUs or CPUs is to use specialized hardware such as FPGAs for disparity computations [9], [17]–[21], freeing up the CPU for other tasks such as path planning or obstacle avoidance. Jin et al. presented a real-time capable FPGA implementation of stereo matching [17] and Banz et al. used an FPGA implementation of SGM [8] to further improve the quality of the estimated depth maps [18]. At the time, both approaches required FPGA boards that were too large for being applicable for mobile robots while [20] presented a hybrid system that still required a CPU. Honegger et al. were the first to show that stereo matching with SGM can be implemented in real-time on a FPGA, resulting in significantly improved disparity maps compared to a two camera system. At the same time, our method runs on the same light-weight FPGA board as [9], [21] while requiring only insignificantly more energy. While Webb already provided an FPGA implementation for multi-baseline stereo about two decades earlier [19], his method was limited to processing low-resolution images. To the best of our knowledge, ours is the first system that is directly applicable to mobile robotics.

III. MULTI-BASELINE STEREO

In this section we describe the algorithm that uses multiple cameras to estimate disparity values. First of all, lens distortion of the individual cameras is corrected and the images are aligned with respect to epipolar geometry. Using epipolar geometry reduces the complexity of the disparity estimation as the search range appears one-dimensional aligned with the image scanlines. Secondly, a pyramid approach is used to combine the different baselines of the cameras. Lastly stereo matching is performed based on a matching cost table depending on all cameras.

A. Lens Distortion Correction & Rectification

As is common practice, lens distortion effects and rotation offsets among the different cameras are removed. We employ the lens model introduced by [22]. The individual pixels are corrected with respect to radial and tangential distortion. A homography is then applied to align the different cameras. Distorted coordinates \(x_d\), normalized with respect to focal length \(f_c\) and principal point \(c\), are calculated according to

\[
x_d = (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) x + x_t.
\]

(1)

Here, \(x\) denotes the original normalized coordinate, \(r\) the distance to the principal point

\[
r^2 = x_1^2 + x_2^2
\]

(2)

and

\[
x_t = \frac{2\xi_1 x_1 x_2 + \xi_2 (r^2 + 2x_1^2)}{\xi_1 (r^2 + 2x_2^2) + 2\xi_2 x_1 x_2}
\]

(3)

represents the shift caused by the tangential distortion. The radial distortion parameters \([\kappa_1, \kappa_2, \kappa_3]\), tangential distortion parameters \([\xi_1, \xi_2]\), camera intrinsic parameters \([f_c x, f_c y, c_x, c_y]\) as well as the homography for rectification are estimated based on [23] in an offline calibration routine.
B. Multi-Scale Setup

A disparity value \( d \) is related to the depth \( z \) by

\[
d = \frac{bf_c}{z},
\]

where \( b \) is the baseline of the two cameras and \( f_c \) is the focal length. This equation indicates that the baseline \( b \) is a magnification factor in measuring \( d \). With a fixed focal length \( f_c \), the disparity value that measures the same depth with a shorter baseline is a fraction of the original baseline disparity value. Therefore to combine matchings of different baselines it is required to calculate subpixel resolution for shorter baselines than the reference baseline. On the contrary larger baselines lead to a larger search range to cover the same depth range as with the reference baseline. Therefore a multi-baseline stereo system needs to handle images at different scale in order to combine the disparity estimations into a single depth map.

C. Stereo Matching

As shown in [6], summing the matching cost functions obtained from matching with multiple images at different baselines leads to an unique and well defined minimum at the correct depth for each pixel. We thus first match the leftmost image against all other images. Computing the matching costs for each pixel for all disparity candidates results in a cost volume per camera pair. These costs volumes are aligned by scaling the individual cost volumes to the size of the volume of a reference camera pair (and thus a reference baseline). Summing up the volumes yields a global cost volume that defines a global cost function for each pixel. For each pixel, the disparity corresponding to the minimum in its global cost function is then selected as the disparity output.

IV. IMPLEMENTATION

In the following section we introduce the physical test system and its constituent parts, and then describe an efficient implementation of the developed multi-baseline stereo algorithm within the different hardware components. An overview of the setup is shown in Figure 1.

A. System Setup

In total, four CMOS sensors with global shutter are connected to a single FPGA module. The individual pixel data streams are synchronized with respect to a few pixels offset and sent to the lens distortion correction and rectification module. The subsequent stereo matcher estimates disparity values. Corrected image data as well as disparity values are sent to a USB controller chip and finally transferred to a host computer.

We configured the four-cameras in a single line with increasing baselines between the left camera and the corresponding right of 35 mm, 70 mm and 140 mm. Figure 3 shows the four camera system with indicated baselines.

B. Multi-Scale Setup

To facilitate the disparity computation as shown in [6], we used a multiple of two and a multiple of four of the initial baseline to place the third and fourth camera (cf. Figure 3). However you could use arbitrary baselines with a more complex scale alignment procedure.

B. Lens Distortion Correction & Rectification

The four data streams of the synchronized cameras are stored in independent internal buffers first. In parallel, the module calculates the distorted and unrectified pixel coordinate with respect to the calibration parameters as shown in Equation 1. The address coordinate calculation is time-shared among all four cameras and runs at four times the pixel clock speed to save multiplication resources. A fixed point representation is chosen with the accuracy set to 18-bit to fit in the available multiplication units of the used FPGA. The calculated address points to the corresponding buffer position where the pixel data is stored to generate a corrected data stream. Bilinear interpolation is performed as the address points to fractions of buffer positions. The pixel displacement forces a buffer size of ±20 lines, which generates a latency of 1 ms. Four corrected data streams are finally sent to the disparity matching module.

C. Multi-Scale Setup

Figure 2 shows the cost computation path in a simplified example for a disparity search range of four pixels. The bit width of the input pixel streams is 8 bits as indicated on the arrows. The output bit width in the simplified example to represent the four possible values is two.

To align the cost functions of the cameras with different baselines with respect to the left camera, a multi-scale approach is used. We set the 70 mm baseline as the reference and added a camera with shorter baseline 35 mm and longer baseline 140 mm to include both possible extensions. Single level subpixels are generated for the 35 mm baseline camera at twice the clock speed to keep at the data rate of the regular baseline. To align the cost function of the 140 mm baseline camera additional storage is used to double the search range.
Fig. 2: Cost aggregation data path, simplified example for a disparity search range of four pixels. The census mask of the left camera is matched with the census masks of the three right cameras. Right2 corresponds to the reference baseline, Right1 has half the baseline and Right3 the double baseline with respect to the left camera. Subpixel resolution is calculated for the costs of Right1 and a larger search range is stored for the costs of Right3 to align the scale level introduced by the different baselines. Cost values of all three matches are summed up and the disparity value with minimum cost is chosen as valid output.

D. Stereo Matching

Census cost masks [24] are generated individual for all cameras and matched against the leftmost camera using an XOR operation. A 7x7 pixel window size is used to generate the census masks. The hamming distance, that is the number of bits different in two census masks, is summed up among all cameras and candidates to generate a global cost function. Since the census mask is less sensitive to illumination changes than other algorithms, all image sensors can use independent internal automatic gain and exposure functions. The candidate with the lowest aggregated cost is selected as the valid disparity output. In addition to the diagramm shown in Figure 2, a left-right consistency check is performed to detect occluded regions and a 3x3 pixel median filter removes spikes.

E. Hardware Components

We built cameras with M8 sized lenses and MT9V024 CMOS image sensors from ONSemiconductors. The sensors feature 752x480 pixels resolution with global shutter and provide image data at 60 frames per second. An SO-DIMM memory sized FPGA board from Enclustra is used including a Xilinx Artix 7 XC7A100T FPGA. Processed images and disparity maps are sent to a host computer using a FX3 USB3 controller from Cypress. Figure 3 shows the system setup with four camera modules connected to the FPGA.

Fig. 3: System setup. The four cameras are aligned parallel on a metal bar. The baseline is increasing with respect to the left camera. All cameras are connected to a single FPGA board.

and outdoor scenarios. This paper is accompanied by a video showing the stereo matching results.

A. Specifications

Table I shows the specifications of the prototype system including size, power consumption and frame rate. The system produces disparity maps based on census matching costs of four cameras with different baselines. The FPGA implementation is pipelined at pixel clock speed of the image sensors and the frame-rate of 60 Hz is limited by the image sensors not by the computational power. The rolling buffer of ±20 lines in the lens distortion correction and rectification module causes a delay of 1ms. The power consumption of the entire system including FPGA, USB3 controller and four cameras is 4.25 Watts, the system can therefore be bus-powered from the USB interface. In comparison, the system with only two cameras connected consumes 4.0 Watt, i.e. only 0.25 Watt less. The form factor is 72 mm by 38 mm and the overall weight 70 grams. A single camera weights 8 grams.

The USB3 controller is configured as a composite USB device including a USB video class interface, a USB communication device class, and a USB mass storage device. The three interfaces are used to stream video data, enable communication to upload calibration parameters and store the parameters in embedded FLASH memory. Most operating systems include support for all three interfaces. Therefore no additional driver is needed.

B. Stereo Matching

We tested the matching performance of the disparity estimation pipeline in an indoor and outdoor setup. Both experiments were performed with one, two and three right

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
<th>Unit</th>
<th>Characteristic</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>72x38</td>
<td>mm</td>
<td>Resolution</td>
<td>752x480</td>
<td>pixel</td>
</tr>
<tr>
<td>Weight</td>
<td>70</td>
<td>g</td>
<td>Frame rate</td>
<td>60</td>
<td>Hz</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>4.25</td>
<td>Watts</td>
<td>Latency</td>
<td>1</td>
<td>ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pixel Clock</td>
<td>25</td>
<td>MHz</td>
</tr>
</tbody>
</table>

TABLE I: Overview of the technical specifications
cameras operating to visualize the effect of the multi-baseline approach.

1) Indoor fixed Depth: We placed a well textured object at fixed distance of 1.75 meter (corresponds to a disparity value of 26 pixels in our setup) in front of the cameras as shown in 4a. The depth maps are visible in 4b for all three right cameras in operation, in 4c for only two right cameras enabled and in 4d with only one active right camera. With all three right cameras enabled the disparity map contains the most valid matches. In Figure 5 we analyzed the number of correct matches for a single frame on the textured object highlighted in 4a and show them as cumulative histogram of disparity errors. Using all three right cameras, there are more than 95% correct disparity values within a 5 pixel error band. With two active right cameras the rate drops to 80-90%. Only 20-35% of the pixels are matched correctly with one active right camera. This clearly shows using more cameras and combining the matching costs results in more accurate and complete disparity maps.

2) Outdoor: We compared the stereo matching results in an outdoor scene using different amounts of right cameras. In Figure 6a, a split view is presented containing stripes of all four cameras. Figure 6c shows the split view with camera Right2 disabled. In Figure 6e two cameras, Right2 and Right3, are disabled. The corresponding disparity maps are shown in Figure 6b for three right cameras operating and in Figure 6d for only two right cameras working. Finally, Figure 6f shows the disparity map generated using only one right camera. As in the indoor experiments, using more active cameras results in more valid disparity matches. All cameras use independent auto exposure functions and have slightly different settings. However, due to the underlying illumination insensitive census cost function no effects are recorded. The multi-baseline setup is more reliable compared to a regular two camera system as depth perception is possible even with one right camera completely occluded.

Finally, we compared the disparity map generated with the multi-baseline setup with a disparity map estimated by a two-camera system using an SGM implementation as shown in [9]. Figure 7a shows the multi-baseline disparity map and Figure 7b shows the disparity map generated by the SGM algorithm. The SGM approach smoothes out small details in producing a dense map. In contrast, the multi-baseline approach is able to recover fine structures and creates a dense map with a less complex algorithm compared to SGM.

VI. Conclusion

In this work, we have presented a low latency multi-baseline stereo system. The disparity maps generated by all four cameras outperform the maps generated by only three or two cameras. Adding two additional cameras therefore improves the disparity map accuracy while only adding little additional weight of 16 grams and power consumption of 0.25 Watt compared to a two camera stereo system [9].

The current multi-baseline system outputs local cost matched results only. Consequently, a global optimization on top as SGM could further improve the quality of the disparity map. Furthermore, the different baselines can be exploited to increase the depth resolution compared to a two camera stereo system.

REFERENCES

In Fig. 6: Outdoor scene: A split view of four cameras is presented in (a), (c), and (e). The image is split into four strips showing cameras Left, Right1, Right2, and Right3. In (c) Right2 is disabled and in (e) Right2 and Right3 are disabled. The disparity map resulting from three, two or only one active right camera are shown in (b), (d) and (f).

Fig. 7: Outdoor scene: Disparity map of the multi-baseline system in (a), two camera disparity map with SGM optimizations in (b). The SGM approach clearly smoothes out fine structures, while the multi-baseline approach preserves details.