A Real Time 3D Sensor for Smart Cameras

Stefano Mattoccia
Dep. of Computer Science and Engineering
University of Bologna
Viale Risorgimento, 2 40136 Bologna, Italy
stefano.mattoccia@unibo.it

Paolo Macri'
Dep. of Computer Science and Engineering
University of Bologna
Viale Risorgimento, 2 40136 Bologna, Italy
paolo.macri@studio.unibo.it

ABSTRACT

Thank to the widespread diffusion of RGBD sensing devices based on active technologies, in recent years, many research and industrial applications have taken advantage of reliable cues computed from dense depth data. However, although these sensors can be very effective in many circumstances, they are not always well suited for outdoor environments and can also interfere with each other when sensing the same area. On the other hand, traditional systems based on passive stereo vision technology, due to their computational/energy requirements, reliability, size, cost etc, have been, so far, mostly confined to research projects. Nevertheless, recent advances in computation architectures and algorithms enable to overcome most of these issues and, in this paper, we describe the architecture and the processing pipeline of an effective RGBD sensor based on stereo vision suited for real time applications. This sensor allows us to infer, in indoor and outdoor environments, dense and accurate depth maps computed according to state-of-art algorithms and with minimal energy requirements that fit with typical constraints of smart camera systems.

Keywords

3D, stereo vision, FPGA, depth sensing, smart camera

1. INTRODUCTION

Accurate and (sometimes) inexpensive depth sensors have greatly increased the interest for 3D vision in recent years and this fact has lead to the development of very interesting applications. Most of these sensors provide dense depth map and images of the sensed environment and for this reason they are often referred to as RGBD (RGB + Depth) sensors.

Most of these sensors rely on active technologies that, by perturbing the sensed environment, allow to infer depth. The Kinect is certainly the most popular sensor belonging to this class. It relies on a infrared (IR) structured light pattern projected into the sensed scene and a patented algorithm that, by analyzing how this pattern appears in the scene, allows to infer depth. This device also includes a rolling shutter color image sensor at VGA resolution (640×480) tuned on the visible spectrum.

Although not so widespread diffused as the Kinect, another interesting active 3D sensing technology is *time of flight* (TOF). This technology projects into the sensed scene a coded invisible pattern and infers depth measuring the time required by the pattern for bouncing from the emitter to the receiver. In most cases TOF sensors provide also an image (often monochrome) of the sensed scene and, for this reason, these devices belong to the class of RGBD sensors.

Finally, another active technology based on elapsed time required by a signal for bouncing from the transmitter to the receiver is LIDAR. In this case the signal consists of a laser pulse. Devices based on this technology, are however typically cumbersome, expensive and with moving mechanical parts. Moreover, they do not natively provide an image of the sensed scene and, although frequently deployed in different conditions, they are more suited for static scenes. On the other hand, compared to sensors based on structured light or TOF, LIDARs are perfectly suited to outdoor applications where other active technologies, in particular under sunlight, would be very noisy (e.g., TOF) or completely useless(e.g., Kinect).

A different well known passive RGBD technology, purely based on imaging devices, is stereo vision. This technology infers depth by triangulating corresponding (or homologous) points projected from the sensed scene to, at least, two imaging sensors with known relative position in space (i.e., *extrinsic* parameters) and known internal camera parameters (i.e., *intrinsic* parameters). Compared to other 3D sensing technologies, stereo vision is passive, although pattern projection can be used to improve its effectiveness in determining corresponding points. This enables simultaneous sensing of the same area with multiple stereo vision sensors. Moreover, this RGBD technology is well suited for indoor as well as for outdoor environments. Despite these positive aspects, stereo vision is computationally demanding and for this reason has been considered for a long time not suited for smart cameras. In fact, the typical computing platforms, such as high end CPUs and GPUs, that allow for inferring dense depth map in real time have high energy requirements (and size).

However, as we'll show in this paper, modern FPGAs enable to design very compact, lightweight and effective depth sensors based on stereo vision technology that fit with the constrained energy requirements of typical smart camera systems. Compared to most existing stereo cameras with

ICDSC '14 November 04 - 07 2014, Venezia Mestre, Italy

ACM 978-1-4503-2925-5/14/11 ...$15.00.
http://dx.doi.org/10.1145/2659021.2659058.

FPGA processing, our proposal allows us to obtain accurate and dense depth maps according to a processing pipeline self contained into a low cost FPGA.

2. RELATED WORK

Among the technologies aimed at inferring depth, stereo vision is probably the most dated. Early attempts to tackle this problem were proposed in the 60s and since then many approaches aimed at solving the *correspondence problem* have been proposed. An exhaustive review and evaluation of significant approaches proposed in this field was proposed in [14]. A more recent review of this area, focused on computing architectures suited for real-time stereo vision systems, was proposed in [15].

Compared to most other 3D sensing technologies, in stereo the key component, excluding the imaging system, is the algorithm that determines homologous points. It, starting from two rectified stereo pairs [14], aims at detecting homologous point by searching for each point of the reference image (e.g., left) the corresponding point in the target image (e.g., right) within a prefixed disparity range.

The correspondence problem remains very challenging due to many pitfalls (e.g., occlusions and other artifacts that may appear in the scene or in the images). The outcome of this algorithm is an image, referred to as *disparity map*, that encodes information concerned with the distance between each scene point and the camera.

Different algorithms for stereo correspondence enable different degrees of accuracy in depth perception and often results also in very different computational requirements and memory footprint. According to the taxonomy proposed in [14] most stereo correspondence algorithms can be divided in two broad classes: *local* and *global*.

Local algorithms have a simple computational structure and determine, independently for each point in the reference image, the supposed corresponding point in the target image by evaluating the best score, computed according to certain cost functions aggregated within patches (in most cases) centered on the examined points. A review and an evaluation of cost functions frequently used in practical stereo vision systems can be found in [9]. Although the computational structure outlined for local algorithms intrinsically exposes a high degree of parallelism, on the other hand, this means that these algorithms explicitly do not enforce constraints on the depth map. From the computational point of view this class of algorithms has, in most cases, a very limited memory footprint. This fact is crucial when dealing with architectures with limited resources such as that considered in our project. A local algorithms based on a simple, yet robust, cost aggregation strategy is *fixed window* [10, 19] that we mapped into our computing architecture. Hardware implementations based on more sophisticated cost aggregation strategies inspired by [17] were proposed in [4], in [3] using the mini-census transform and a simplified and computationally efficient cost aggregation strategy, and in [16] using image segmentation as main cue for matching cost aggregation.

On the contrary, global algorithms explicitly enforce on the disparity map a 2D smoothness constraint. These algorithms tackle the correspondence problem as a labeling assignments that minimize an energy function with two terms that evaluate how well the disparity assignment fits with data (i.e., the input images) and the smoothness constraint. Typically, in these algorithms, the disparity assignment that minimize the energy function is obtained by means of iterative optimization approaches [14] such as *belief propagation* or *graph-cuts*. Unfortunately, these techniques have very high memory footprints and very high memory bandwidth requirements.

Since our target computing platform has constrained resources, especially in terms of available memory, global algorithms seem not well suited for our purposes. However, in the latter class falls also algorithms based on simplified energy minimization methodologies that enforce a smoothness constraint on 1D domains. These algorithms are typically based on *dynamic programming* or *scanline optimization* [14] techniques. In particular, an algorithm based on multiple independent scanline optimization that, thanks to its efficiency and effectiveness, has become very popular in recent years is the SGM approach [8]. It determines corresponding points by combining the cost provided by multiple instances (8 or 16 in [8]) of independent scanline optimizations computed along different converging paths. When considering architectures with constrained resources, such as those based on FPGAs, this method has some major drawbacks. In fact, it requires to scan the stereo pair from top to bottom and in the opposite direction (this requires storing of the whole input images). Moreover, it has a high memory footprint and high memory bandwidth requirements. Despite this fact, it can be implemented in FPGAs as reported in [5] and [1]. The implementation proposed in [5] is very similar to the original formulation and allows for processing stereo pairs at 320×400 while the implementation proposed in [1] is a simplified approach aimed at reducing memory bandwidth at the expense of a small reduction in terms of overall accuracy. Both implementations rely on high end FPGAs.

In the next sections we provide a description of the architecture of our 3D camera and of its processing pipeline entirely mapped on a low cost FPGA that, in the configuration reported in this paper, consists of a Xilinx Spartan 6 Model 75. Our processing pipeline can be easily configured *on the field* with two algorithms, fixed window and our memory efficient version of the SGM approach [8].

3. OVERVIEW OF THE FPGA-BASED COMPUTING ARCHITECTURE

As previously outlined, a major goal in our work was aimed at designing a compact, lightweight and energy efficient RGBD sensor enabling accurate depth sensing based on stereo vision. After a thorough evaluation of different strategies, we found that a good trade-off between hardware complexity and depth map accuracy could be obtained by deploying a memoryless computing architecture based on low cost FPGA. This choice allowed us to design a very simple processing board (essentially made of an FPGA and a communication controller) that, for the current prototype, is compliant to USB 2.0. This means that the overall processing pipeline is self-contained into the FPGA, as shown in Figure 1, with notable advantages in terms of design complexity, costs, size, power consumption and portability to other computing platforms with similar architecture.

On the other hand, our design strategy also enforces major constraints to the vision algorithms that can be mapped on it. In particular, the main constraint is concerned with memory, available only inside the FPGA. To give an idea,

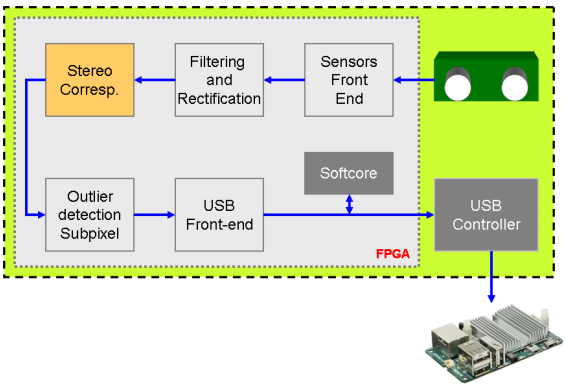

Figure 1: Main functional blocks of the proposed stereo camera design. The 3D camera is connected and powered by the host via USB.

the overall memory available on low cost FPGAs, such as those target of our design, has a size comparable to hundreds of KB. This value, is often smaller than the ≈ 300 KB required to encode a single monochrome image at VGA resolution (i.e., 640 × 480). For this reason, we applied to each implemented module the *on-the-fly* processing methodology in order to reduce as much as possible buffer requirements [12].

Another major constraint arising with the proposed computing architecture is concerned with *frame buffering*. In our design, as previously highlighted, there is not enough memory to store a single image and thus we can't put the outcome of the processing pipeline into a frame buffer. For this reason, in our design we stream data to the USB controller adopting, again, an on-the-fly approach that relies only on a small buffer, of few kilobytes, synthesized into the FPGA logic. It is worth observing that, since in most cases we are interested in streaming more than a single image (e.g., the rectified reference image, the disparity map and the map encoding subpixel values), we designed an effective module that allows us to combine more images into a single video stream. This module enables to reliably transmit multiple images using the same endpoint of the USB controller and hence allows us to avoid a physical external frame buffer.

Figure 1 outlines the main functional blocks of our 3D camera implemented on the reconfigurable FPGA logic (in light gray). In the figure we can observe a *front-end* that converts the input video streams provided by the two imaging sensors and the processing pipeline made of: *filtering and rectification* module, *stereo correspondence* module and *outliers detection and subpixel interpolation* module. Although not explicitly reported in the figure, a small FIFO aimed at handling small latency during USB transfers is synthesized into the FPGA. The figure also highlights a *softcore*, synthesized on the reconfigurable logic, that handles all the communications (excluding those involving the video streams processed by the vision pipeline) with the host computer. The proposed hardware design also allows to obtain power supply from the same USB connector used for data. The overall power consumption processing stereo pairs at more than 30 fps and at 640 × 480 resolution is about 2 Watt with a Xilinx Spartan 6 Model 75. Unfortunately, the USB 2.0 bandwidth does not allow streaming at 30+ fps of three images (encoding the rectified reference image, the disparity map and the subpixel measurements) at 640 × 480. In these circumstances, with the USB 2.0 interface, the three images are internally processed at more than 30 fps but they are actually acquired at about 22 fps by the host computer due to USB bandwidth constraints. In the figure, outside the FPGA, we can see the a single notable device included in our design; a Cypress FX2 USB 2.0 controller aimed at managing all the communication between the 3D camera and the host.

4. PROCESSING PIPELINE

In this section we describe the main processing modules of the stereo camera implemented into the FPGA logic. All the modules are connected by means of small FIFOs synthesized into the reconfigurable device and, excluding the input of sensor front end and the output of the USB front end, they receive the same pixel clock. The maximum frequency allowed by the sensors used for our experiments is 27 MHz.

4.1 Sensor front end

This module converts the 18 bit, 16 data bits plus a *start* bit and a *stop* bit, serial LVDS stereo video stream sent by the imaging sensors into two distinct 8 bit parallel video streams containing the synchronized images sensed by the camera. Once completed this phase, the pixel frequency of the converted video stream is reduced by a factor 18 and sent to the processing pipeline. The *deserialization* described is implemented into the FPGA logic and does not affect the quality of the images. The resulting pixel clock of the parallel encoded video stream is upper bounded by the maximum frequency of the sensors that in our specific case is 27 MHz (for original serial LVDS video stream the maximum frequency is 486 MHz).

4.2 Filtering and rectification

This module receives the two *deserialized* video stream and performs a sequence of operations required by the stereo correspondence module. The filtering and rectification module may include a pre-processing operation (e.g. smoothing) aimed at reducing the image noise. However, its main task consists in warping the input video streams according to the parameters determined by the calibration phase. This phase is crucial for the successive processing modules in order to obtain images with *epipolar* lines aligned to the same image scanlines. This fact, given a certain point in the reference image, enables to restrict the the position of the corresponding point on the same scanline of the target image. In our architecture image warping is carried out by means of a bilinear interpolation of distorted input pixels selected and weighted according to parameters computed by processing a data structure containing only sparse target coordinates. This strategy allows us to obtain good accuracy with a minimal overhead in terms of logic resources and memory requirements. Of course this strategy requires buffering, on the internal FPGA memory, of a certain number of scanlines. The amount of scanlines is related to the maximum image distortion computed during off-line calibration. Once completed image warping of the two video streams, the filtering and rectification module can optionally compute additional image filtering operations such as Laplacian of Gaussian, edge detection/enhancement, etc. In our current implementation, we apply to the warped images a variant of the census

transform [18] that, by considering the relative ordering of pixel intensity, is robust [9] with respect to strong radiometric distortions.

4.3 Stereo correspondence algorithms

The stereo correspondence algorithm aimed at solving the correspondence problem is the core component of the stereo camera and we implemented two distinct algorithms: the fixed window approach and a memory efficient version of the SGM algorithm.

The fixed window approach can be implemented on the reconfigurable logic without any modification with respect to its software counterpart by means of appropriate strategies aimed at reducing redundant calculations as well as at limiting buffer requirements. By following these guidelines our implementation perfectly fits with the limited resources available into our design.

Concerning the SGM algorithm [8], as previously highlighted this algorithm has a very high memory footprint and also very high requirements in terms of memory bandwidth. Since in our design we decided to avoid at all external memories, in its original formulation this method wouldn't be implementable on such a computing architecture. However, by including for cost computation a restricted number of scanlines enables to significantly reduce, at the expense of a small degradation in terms of accuracy, the overall memory requirements for storing intermediate cost computation. This fact, coupled with other methodologies aimed at further reducing memory requirements, allowed us to match the constrained resources available in our design.

4.4 Outliers detection and subpixel interpolation

Once computed the disparity map, we validate each value according to two constraints. The first one is concerned with the amount of texture detected in proximity of the examined point in the reference image. Since the census transform is not reliable in uniformly textured areas we mark as unreliable those disparities that have an insufficient amount of texture in their corresponding position of the reference image. For this purpose, we evaluate the image texture according to the amount of image gradients detected in proximity of the examined point. Although other approaches, such as *ternary census*, would certainly improve the effectiveness of the original census transform they also would lead to significantly increased hardware requirements.

We also perform a further validation of the computed disparity value by enforcing the *uniqueness* constraint. Finally, the outliers detection and subpixel interpolation module, performs subpixel disparity interpolation by fitting a parabola according to the matching costs computed in proximity of the best score found by the stereo correspondence module. In the setup reported in this paper, subpixel interpolation is performed at $\frac{1}{8}$ of pixel but other configurations can be implemented for specific application requirements.

4.5 USB front end

The USB front end is aimed at providing a minimal buffering for the processed images before they are sent to the USB communication controller. This module, consists of a small FIFO, synthesized in the FPGA, that enables to deal with small latency in the USB communication channel. This module also combines multiple video streams in order to use a single endpoint for all the images/maps.

Of course, the USB front end is also responsible for handling any communication between the stereo camera and the host. This task is accomplished with the supervision of the softcore, synthesized into the FPGA logic, depicted in Figure 1.

5. EXPERIMENTAL RESULTS

In order to prove the effectiveness of our 3D camera in challenging scenarios, in this section we provide experimental results concerned with three practical applications that rely on the dense and accurate depth measurements provided by our 3D camera. Specifically, we report experimental results in three very different applications (obstacle detection, 3D people tracking and 3D SLAM) with three different configurations of the camera.

5.1 Obstacle detection for robot navigation

This application aims at detecting obstacles in front of a robot in order to enable autonomous navigation, path planning and so on. In this case the setup consists of the proposed 3D camera connected to an embedded board Odroid U3 [6] based on the ARM SoC, model Exynos 4412, a quad core Cortex A9 CPU clocked at 1.7 GHz, with the Linux operating system. By means of a robust RANSAC based approach applied to the *v-disparity histogram* [11] computed according to the dense disparity maps provided by the downward looking 3D camera, the algorithm implemented on the ARM board detects the ground plane at each new frame and highlights obstacle in sensed area.

Figure 2 shows four frames concerned with an urban scenario. In this specific case, a person simulates the robot by walking on a narrow sidewalk. The processing pipeline in this experiment is configured with our implementation of the SGM algorithm, monochrome sensor with image resolution set to 320×240 in order to reduce the computational requirements for the Odroid platform and a stereo baseline of about 6 cm. Observing the figure we can notice that, in the four frames reported, the 3D camera enables to obtain very accurate and robust depth maps (encoded with warmer colors for points closer to the camera). In the figure, dark blue points represent point marked as unreliable by the outliers detection module of the processing pipeline mapped into the the FPGA. The figure shows that the 3D camera enables to infer smooth, accurate and dense depth maps of the sidewalk and of the surrounding areas (cars on the left side and small walls and shrubs on the right side). The person with the shopping bags approaching the camera is correctly sensed by the camera in all the three reported frames. Dark blue points are concerned with potentially unreliable regions corresponding to untextured areas in the reference images that are correctly highlighted by the outliers detection module. The accurate dense depth maps inferred by the camera allow to obtain a robust ground plane detection as shown, in white, in the rightmost picture of Figure 2. In this case we detect potential obstacles (i.e., a minimum amount of pixel not belonging to the ground plane) in the sensed squared area highlighted in white in the rightmost images. Green means no obstacles while red means obstacle detected (in this latter case we provide the distance from the closest sensed obstacle).

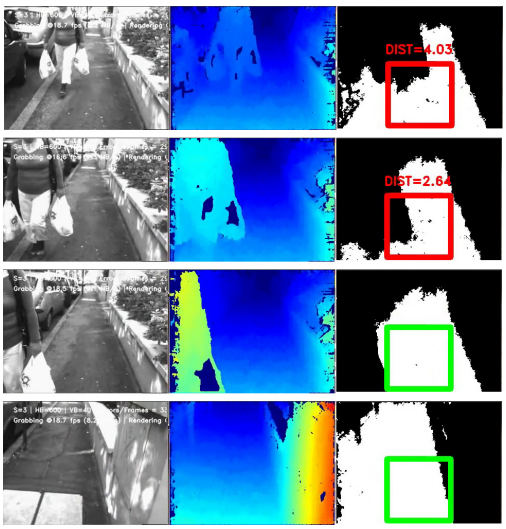

Figure 2: Four frames concerned with a robust and real time obstacle detection algorithm for autonomous robot navigation based on the dense depth data provided by the camera. For each frame: (Left) The reference image (Middle) The disparity map computed by the stereo camera (warmer colors encode points closer to the 3D camera, in dark blue those points marked as unreliable by the camera) (Right) In white the detected ground plane. The obstacles are detected within the square window: green means no obstacles while red means obstacle detected.

The overall algorithm, thank to the 3D camera front-end with FPGA processing, runs at more than 15 Hz on the Odroid U3 platform in the reported configuration. Moreover, the overall weight of reported system (3D camera with lenses plus the Odroid) is less than 150 g and the overall power consumption less than 10 Watt (the camera accounts only for about 2 Watt). For these reasons, this guiding system is well suited also for small, self-powered robots as tested during an extensive evaluation in indoor and outdoor environments.

5.2 3D people tracking

Another interesting application of the proposed sensing device is concerned with a 3D people tracking system with downward looking camera. The algorithm proposed to tackle this problem relies on a *bird-eye view* projection ([7, 13]), shown at the bottom of Figure 3, of the point cloud inferred by the stereo camera and on the *meanshift* algorithm for tracking.

Figure 3 shows, on top, five consecutive frames of the reference image concerned with a sequence used for testing the 3D tracking system in outdoor. In the middle of the figure are reported the corresponding disparity maps computed by the 3D camera encoded with grayscale values: brighter values encode closer points, black encodes points filtered by the occlusion detection module. Finally, the bottom of the figure shows the outcome of 3D people tracking on a virtual ground plane. The 3D camera in this experiment is configured with our implementation of the SGM algorithm, monochrome sensors configured at 640 × 480 image resolution and a baseline of 6 cm. Focusing our attention on the disparity maps, we can notice that the 3D camera provides very accurate and dense depth measurements. The shape of the two people sensed by the sensor is correctly inferred and the camera provides, even at higher resolution wrt the setup described in the previous section, accurate, smooth and dense maps for the background as well. The outliers detection module implemented in the processing pipeline correctly and consistently filters out occlusions (mostly corresponding to points on the right side of the two people) and the uniform bright region, at the bottom left of the reference image. There are only few small outliers corresponding to some points in the upper side of the background. Thank to the accurate and reliable depth maps inferred by the 3D camera and the tracking algorithm, as shown at the bottom of the figure, turns out to be very effective, even when people get very close, and fast (in the described setup it runs at about 20 Hz on a Pentium notebook).

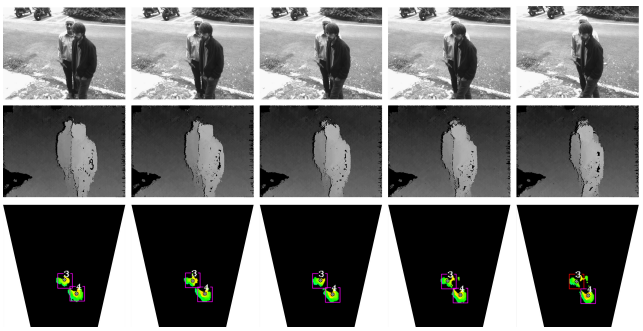

Figure 3: Reliable and real time 3D people tracking. The figure shows five consecutive frames concerned with a 3D people tracking application. (Top) The reference image (Middle) The disparity maps computed by the 3D sensor (encoded with grayscale levels, brighter means closer and black means points marked as unreliable by the camera) (Bottom) Outcome of tracking in the sensed area.

5.3 3D SLAM

Finally, in this section we report preliminary experimental results regarding a 3D *simultaneous localization and mapping* (SLAM) system aimed at obtaining a registered point cloud of an environment framed, from unconstrained positions, by the 3D camera. This system relies on 2D features extracted from the reference image and on dense depth maps inferred by the 3D camera. The 2D features, extracted in this experiment by means of the SURF [2] approach, are matched between adjacent views in order to detect corresponding points. Reliable matched points are then projected into the 3D space, according to the depth map inferred by the 3D camera, in order to estimate the pose of the camera by means of the SVD approach. The computed pose is used as initial guess for a variant (i.e., *point to plane*) of the the ICP algorithm. This refined pose enables to add the current 3D points to the the global point cloud obtained by previous registrations. Although our current SLAM pipeline implemented in software is not yet ready for real time, by processing according to the approach outlined the point cloud inferred by the 3D camera, it yields promising results as shown in Figure 4 concerned with 3D scanning of an office room.

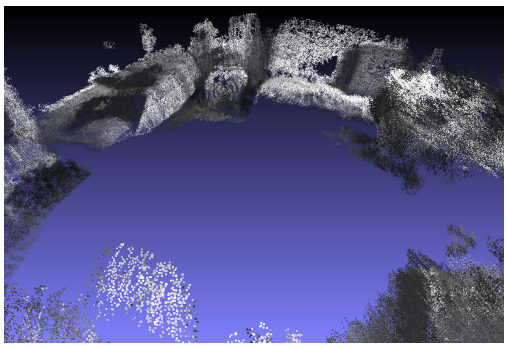

Figure 4: Preliminary 3D registration of an indoor environment using SLAM techniques. The registered point cloud shown in this figure was obtained by processing the images and the depth maps, computed according to our hardware implementation of the fixed window approach, provided by the 3D sensor during unconstrained movements.

The results reported in the figure outline that our camera is also suited for 3D fine reconstructions based on image and depth cues. In fact, observing the figure we can notice that, despite the limitations of the current SLAM pipeline implemented, the 3D structure of the scene is correctly inferred. Nevertheless, outliers and drifts in the registration can be observed in the figure. In these experiments, the camera is configured with the fixed-window algorithm, monochrome imaging sensors at 640×480 and a baseline of about 6 cm.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have described the motivations behind the design of a computing architecture based on a low cost FPGA and the processing pipeline of a 3D camera enabling real time, accurate and dense depth measurements according to state-of-the-art stereo vision algorithms. The resulting compact and lightweight device enables to process color or monochrome stereo pairs sensed by global shutter imaging sensors. The optimized hardware design and algorithms implementation allowed us to obtain a small and compact 3D camera with a reduced power consumption of about 2 Watt at 640×480. These facts make this device well suited for smart camera systems as reported in the experimental results concerned with three practical applications (in three different configurations of the device) where the 3D camera is effectively being used. Future work is aimed at implementing into the same FPGA board used for depth computation algorithms for feature detection and description, in particular those based on *binary descriptors*.

7. REFERENCES

[1] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In *ICSAMOS*, pages 93–101, 2010.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[3] N. Y.-C. Chang, T.-H. Tsai, P.-H. Hsu, Y.-C. Chen, and T.-S. Chang. Algorithm and architecture of disparity estimation with mini-census adaptive support weight. *IEEE Trans. Circuits Syst. Video Techn.*, 20(6):792–805, 2010.

[4] J. Ding, J. Liu, W. Zhou, H. Yu, Y. Wang, and X. Gong. Real-time stereo vision system using adaptive weight cost aggregation approach. *EURASIP J. Image and Video Processing*, 2011, 2011.

[5] S. K. Gehrig, F. Eberli, and T. Meyer. A real-time low-power stereo vision engine using semi-global matching. In *ICVS*, pages 134–143, 2009.

[6] HardKernel. Odroid-u3. `http://hardkernel.com/main/main.php`.

[7] M. Harville. Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. *Image Vision Comput.*, 22(2):127–142, 2004.

[8] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, 2008.

[9] H. Hirschmüller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1582–1599, 2009.

[10] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim, and J. W. Jeon. Fpga design and implementation of a real-time stereo vision system. *IEEE Trans. Cir. and Sys. for Video Technol.*, 20(1):15–26, Jan. 2010.

[11] R. Labayrade and D. Aubert. Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation. In *Intelligent Vehicle Symposium, IEEE*, volume 2, pages 646 – 651, 2002.

[12] S. Mattoccia. Stereo vision algorithms for fpgas. In *The 9th IEEE Embedded Vision Workshop, CVPR 2013 Workshop*, 2013.

[13] C. Pane, M. Gasparini, A. Prati, G. Gualdi, and R. Cucchiara. A people counting system for business analytics. In *AVSS*, pages 135–140, 2013.

[14] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. Journal Computer Vision*, 47(1/2/3):7–42, 2002.

[15] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald. Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing*, 2013.

[16] C. Ttofis and T. Theocharides. Towards accurate hardware stereo correspondence: A real-time fpga implementation of a segmentation-based adaptive support weight algorithm. In *DATE*, pages 703–708, 2012.

[17] K. Yoon and I. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. PAMI*, 28(4):650–656, 2006.

[18] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV 1994*, pages 151–158. Springer-Verlag, 1994.

[19] P. Zicari, S. Perri, P. Corsonello, and G. Cocorullo. Low-cost fpga stereo vision system for real time disparity maps calculation. *Microprocess. Microsyst.*, 36(4):281–288, June 2012.