FPGA-based Real-Time Disparity Computation and Object Location

Pedro Miguel Santos
DEEC, Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
Email: pedro.miguel.santos@fe.up.pt

João Canas Ferreira
INESC Porto, Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
Email: jcf@fe.up.pt

***Abstract*—This paper describes an FPGA-based system capable of computing the distance of objects in a scene to two stereo cameras, and use that information to isolate objects in the foreground. For this purpose, four disparity maps are generated in real time, according to different similarity metrics and sweep directions, and then merged into a single foreground-versus-background bitmap. Our main contribution is a custom-built hardware architecture for the disparity map calculation, and an optional post-processing stage that coarsens the output to improve resilience against spurious results. The system was described in Verilog, and a prototype implemented on a Xilinx Virtex-II Pro FPGA proved capable of processing 640×480 black-and-white images at a maximum frame rate of 40 fps, using 3×3 matching windows and detecting disparities of up to 135 pixels.**

I. Introduction

The extraction of disparity maps from a stereo camera in real-time is a great contender as a practical method for determining the distance of all physical points in a scene to the camera. It is also a powerful tool to segment a scene into objects, by aggregating areas of points that are at similar distance from the cameras. Abundant use can be found in the automotive industry, robotics and video surveillance.

The FingerMouse [1] system operates under this premise. The system is composed of a device with two cameras that the user wears on his chest. Under the assumption that the user's hand is the closest object to the cameras, it computes four disparity maps in real-time, extracts the closest object in the scene, and determines the coordinates of its center. Its goal is to have your own hand replace conventional pointing devices.

To handle the computationally demanding disparity maps, the authors have designed a dedicated ASIC, described in [1]. Inspired by that work, we have made our own implementation of the concept on FPGA. Our main contributions are an alternative hardware architecture to compute the disparity maps, and an innovative post-processing stage, which can be of interest for many applications.

Our system was dimensioned to process pairs of 640×480 images (8-bit pixels) at a rate of 40 frames per second (fps) and detect a maximum disparity of 135 pixels. The implementation was verified to work in real-time at 25 fps, the maximum image rate of the available image sensors. Comparing to the FingerMouse ASIC implementation [1], our system handles larger images (640×480 vs. 320×480), a larger disparity range (135 vs. 47) and a higher image data rate (24.5 Mpixels/s vs. 5 MPixels/s).

The remainder of paper is organized as follows. Section II presents background information and describes previous work. The overall architecture of the system is described in Section III. The internal structure and operation of the main parts of the system are detailed in Section IV. Section V summarizes the overall outputs, and Section VI presents the conclusions.

II. Background and Related Work

In a stereoscopic setup, two cameras are set side by side, capturing a scene from two slightly different angles. Assuming that both cameras are horizontally aligned, images of the same object on the two cameras will exhibit an horizontal displacement relative to each other. The displacement is called *disparity*, and is proportional to the object's closeness to the cameras. Computing the disparity of all pixels yields a *dense disparity map*.

An extensive overview on methods for producing dense disparity maps is given in [2]. A subclass of such methods, of reduced mathematical complexity but computationally intensive, and therefore well-suited for parallel hardware implementation, is the area-match algorithms.

Area-match algorithms try to find a correspondence for all pixels of one image in the other image of the same stereo pair. First, one of the images is chosen as the reference image. For every pixel in the reference image, the algorithm extracts the matrix of pixels around it and searches the other image - the candidate image - for the matrix of pixels that most resembles it. The candidate matrix that is most similar to the reference one is assumed to indicate the position of that area on the candidate image. The difference of the horizontal coordinates of the matrices yields the central pixel's disparity.

If both stereo images have the same horizontal baseline, the area of search in the candidate image is straightforward. Assuming the right image to be the reference one, no object on the left image (the candidate one) can appear at a position closer to the left margin than it does in the right image. Thus, the search has only to be done leftwards, starting from the X-coordinate of the pixel under analysis in the reference image.

Several metrics can be used to evaluate the similarity of two matrices of pixels [3], [4]. We will highlight two of them, which measure different aspects of the neighborhood. The first is the Sum of the Absolute Differences (SAD) of the pixels of two matrices, which allows to compare the brightness of the matrices. The candidate matrix that yields the smallest SAD will be the most similar to the reference matrix.

The second metric used in our system is a non-parametric summary of the local spatial structure called *Census* [5]. The Census metric is calculated by assigning each pixel of a matrix a bit, according to if it is brighter or darker than the central pixel. The number of identical bits in the resulting sequences of bits associated with the reference and the candidate matrices provides a measure of their similarity.

After the above procedure has been done to *all* pixels of the reference image, one is in possession of a dense disparity map. This means that for every pixel, a value is assigned indicating the distance in pixels to the pixel, in the candidate image, that most likely corresponds to the same physical point. Upon this information, the segmentation of a scene into focal planes is done by selecting only pixels with disparity within a certain range. In this particular work, only two planes are considered (foreground and background), but more could be defined.

The processing of occluded areas presents a difficulty for this type of algorithms. In this case, a background pixel appears in only one of the images; the search for the corresponding pixel in the candidate image will come up with a false match, since no such pixel is actually present in it. For a two-camera setup, one way to avoid this type of error is to perform two disparity computations, each one using a different image as reference (right pixels are searched for on the left image, and vice-versa). A consistency check is then made to assess if a given pixel belongs to the foreground, by verifying if it was classified as such in both disparity computations (foreground pixels are never occluded).

FPGAs have since long been an ideal platform for experimenting hardware architectures oriented for the computationally intensive, highly parallelized nature of most image processing algorithms. A seminal implementation of depth map computation used the PARTS system [6], which was composed of 16 Xilinx 4025 FPGAs and capable of computing 24 stereo disparities on 320 by 240 pixel images at 42 frames per second (using the Census metric).

A more recent example is the hand-sized trinocular system based on FPGA is described in [7], which uses a SAD-based metric to process 320×240 images at 120 fps and can handle disparities of up to 64 pixels. The system of [8] employs a Spartan-3 FPGA, and performs the Census transform on 320×240 images at a rate of 30 fps (eventually up to 150, with adequate cameras) using 7×7 matrices and achieving a maximum disparity of 20 pixels. Further examples and comparisons of hardware systems can be found in [9], [10].

III. System Overview

The overall sequence of steps that compose the implemented system will now be presented. For reference, a diagram of the system's architecture is shown in Fig. 1.

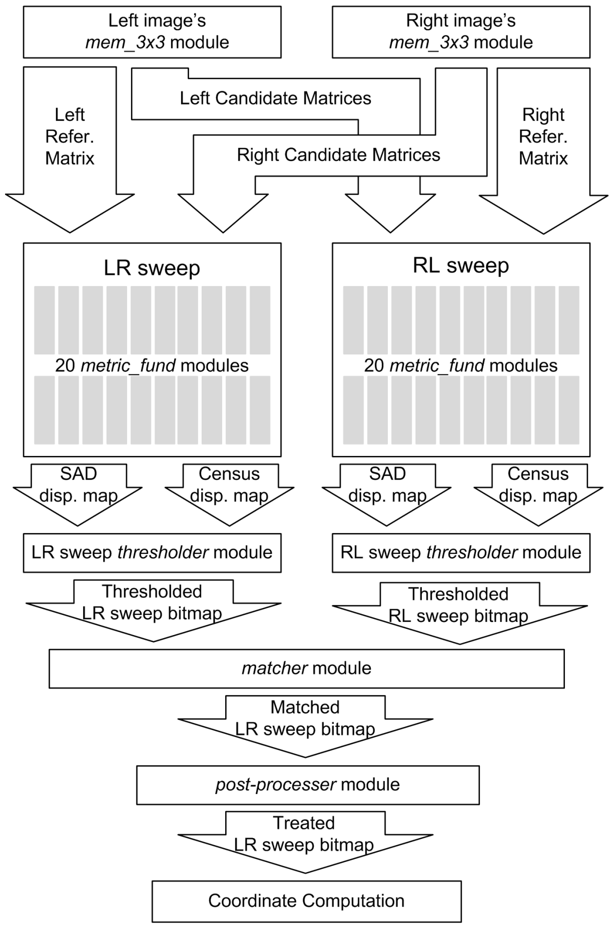

Fig. 1. Overview of System Organization and Data Flow.

1. Image acquisition — The disparity map computation requires that at least 3 lines of each image are buffered. Receiving pixel data from the cameras, storing and delivering it in the correct order to the subsequent modules are the tasks of the hardware modules *mem_3x3* (see Fig. 1).

2. Disparity map computation — This step implements the area-match algorithm presented in the previous section, and requires the joint co-operation of the hardware modules *mem_3x3*, *metric_fund* and *global_ctrl* (not shown in Fig 1). The chosen similarity measures between matrices were SAD and Census, due to their quality/resource consumption trade-off and complementary nature. Four disparity maps are output at this stage: two maps (one for each metric) using the left image as reference and the right as candidate - LR sweep -, and two other using the right image as reference - RL sweep.

3. Binarization — The scene is segmented in two focal planes by thresholding the disparity values at the *thresholder* modules. Since only two focal planes are of interest (background and foreground), a simple binarization operation is used to assign pixels to focal planes.

The results of the two metrics (SAD and Census) are combined for each sweep: a pixel is assigned to the foreground if it has a high enough disparity by any of the metrics. The final output are two bitmaps (one per sweep), indicating to which plane a specific pixel belongs.

4. Matching — The hardware module *matcher* receives the two combined disparity maps, and performs a consistency check on one of the maps to detect areas that were erroneously considered as foreground due to occlusion. The resulting map is the final output of the system.

5. Post-processing — This is an optional stage, implemented by module *post_processer*, which coarsens the output of *matcher*. Possible uses are described in Section IV-E.

6. Coordinate computation — The center of gravity of the foreground pixels is computed in this step: the number of foreground pixels of each row and line is determined and multiplied by the index of the corresponding row or line. Dividing the sum of all row (column) values by the number of foreground pixels yields the horizontal (vertical) coordinate.

IV. Hardware Organization

This section provides more details on the hardware implementation of the main processing steps.

A. Image Acquisition

In our system, 640×480-pixels images are provided by two CMOS cameras with a common horizontal baseline. Each camera delivers to its own *mem_3x3* module an 8-bit luminance value at 12.5 MHz. The two 9×8-bit output ports produce data at 100 MHz: one feeds the reference matrix bus of one of the sweeps, and the other the candidate matrix bus of the other sweep (cf. Fig. 1).

The design of module *mem_3x3* was driven by two major requirements of the system. Firstly, later operation of the system requires pixel matrices to be at least 3×3, which means a minimum of 3 lines of the image must be stored at all times. Secondly, real-time operation implies that one line has to be processed in the same time it takes to receive a new line.

To meet these requirements, module *mem_3x3* uses four internal memories, mapped to FPGA block RAMs, that operate in a round-robin fashion. While three of them provide the data for the matrix operations, the fourth one is being filled with data from the camera. When a complete new line has been received, the oldest line can be discarded and the corresponding memory is used for the next line.

B. Disparity Map Computation

Three different types of modules are required for this stage: *global_ctrl*, *mem_3x3*, and *metric_fund*. Module *global_ctrl* is in charge of managing the entire disparity map computation process: it provides the *mem_3x3* modules with the addresses of the reference and candidate windows, and controls the activation of the *metric_fund* modules.

The core elements of the computation are the 20×2 *metric_fund* modules. Their input data comes from the reference matrix port of one of the *mem_3x3* modules and the candidate matrix port of the other *mem_3x3* module, as shown in Fig. 1. Each receives and stores a reference matrix at a given time, and for the following n cycles they will receive candidate matrices to compare with the reference one. After those n cycles, they output the index of the candidate matrix that was the most similar, according to SAD and Census measures.

As discussed in the previous subsection, the system receives pixels from the cameras at 12.5 MHz and operates at 100 MHz, which gives 8 clock cycles to process each reference pixel. If only one module *metric_fund* is used, at most 8 candidate matrices can be compared with the reference one, i.e., the maximum disparity would be only of 8. Adding more *metric_fund* modules allows for more reference matrices to be handled simultaneously, thus extending the maximum disparity range achievable. The number of modules to use should result as a trade-off between desired disparity range and available resources. Also, for optimal implementation, it should be a divisor of 640. We chose 20 modules for our prototype, allowing each reference matrix to be assigned 160 cycles for processing, and therefore attain a maximum disparity of 160.

However, taking advantage of all 160 cycles would have an important requirement: the stream of candidate matrices presented to each module, over all the 160 cycles, would be different. This would demand 20 independent and simultaneous accesses to memory. To circumvent this problem, the modules *metric_fund* are loaded with reference matrices sequentially, over 20 cycles. Although the maximum disparity attainable decreases from 160 to 140, this strategy allows us to feed exactly the same stream of candidate images to all *metric_fund* modules. Due to implementation details, the effective maximum disparity attainable is 135, which is still quite reasonable as it allows the detection of objects up to 40 cm from the cameras.

Finally, the reason for having two sets of 20 *metric_fund* modules is that disparity calculations have to be made both for the LR sweep (left image as reference) and for the RL sweep (right image as reference). However, the direction on which reference images are swept is from the left to the right in both cases, even though in the case of the LR sweep the other way around would be more intuitive. This option allows us to have the results of both computations available close in time and space, which makes implementation of the Matching stage easier.

We believe the hardware architecture dedicated to this step to be the most important contribution of our work, with respect to previous implementations. A major feature is its good scalability. In its current size, it can work with 640×480 images at a frame rate of 40 fps with a maximum disparity of 138 using windows of 3×3 pixels. However, there is plenty of room for trade-offs. The architecture could, just as well, work at 160 fps over 320×240 images with a maximum disparity of 55. For this last resolution, at 40 fps, the maximum disparity is 295 pixels. On the downside, increasing the matrix size would be very demanding in terms of FPGA resource consumption.

C. Binarization

The previous computation outputs the disparity maps in bursts of 20 pixels, on the last 20 cycles of every sequence of 160 cycles. Recall, as explained in Section III, that there are two sets of results: one for the RL sweep and other for the LR sweep. For each, two disparity maps are available: one using SAD, and the other using Census. The disparity maps are transformed into simple bitmaps by assigning a 1 or a 0 to each pixel (1 indicates a foreground pixel), according to whether their disparity is above or below an user-defined threshold. The resulting bitmaps from the SAD and Census are merged by ORing their outputs, thus producing at the end of this stage two bitmaps, one for each sweep.

D. Matching

The module *matcher* performs a consistency check between the bitmaps of the LR and RL sweeps that result from the last stage: only if both sweeps agree a specific pixel belongs to the foreground, it is considered as such.

The matching consists of ANDing the bit of each LR sweep pixel with the bit of the RL sweep pixel indexed by its disparity. Thresholded results from the LR sweep are stored in a 20-bit long memory with their corresponding disparities; thresholded results from the RL sweep are stored in a 155-bit long memory, so that the full range of disparity is available. This stage outputs one bitmap based on the LR sweep.

E. Post-processing

The post-processing stage is optional and, in the current system, it is used to produce a coarser representation of the segmented image, where each bit corresponds to a 20×20 pixel square of the original bitmap. The final value depends on the number of foreground pixels present in the source square.

The coarser-grained version is less sensitive to small variations of the image data and makes the center of gravity calculation simpler. In addition, the smaller image can be useful for searching in a data base of image templates. This can be useful if the shape of the object in the foreground must be identified from a set of very different alternatives.

V. Physical Implementation

In order to verify the correct functioning of the hardware architecture, we have built a laboratory prototype using two CMOS cameras, based on OV7120 image sensors, horizontally aligned and 7.5 cm apart. Both are connected to a Xilinx FPGA Virtex-II Pro XC2VP30, whose embedded CPUs are not used. For visualization of the results, the disparity maps and final bitmap can be sent to a VGA monitor, and the coordinates are sent by an RS232 connection to a PC. The architecture was specified in Verilog and synthesized using ISE 9.1 from Xilinx. Table I summarizes the resource usage for the implementation with 20 *metric_fund* modules per sweep.

TABLE I
Resource usage for implementation on XC2VP30

Element	Total available	Used	Occupancy
Slices	13,696	9,565	69%
LUTs	27,392	16,847	39%
Flip-flops	27,392	10,936	61%
BRAMs	136	18	13%

The processing core uses a 100 MHz clock. The embedded digital clock manager is used to generate the camera pixel clock (12.5 MHz) and the VGA pixel clock (25 MHz).

VI. Conclusion

We have designed, and implemented on a FPGA, a system capable of determining, in real-time, the position of the object that is in the foreground of the scene captured by two stereoscopic cameras. It is capable of processing two 640×480 black-and-white images at a maximum frame rate of 40 fps, using 3×3 windows, and covering a disparity range of 135.

It does so by computing four dense disparity maps, using the images of both stereo cameras as reference and two similarity metrics, SAD and Census. The distance information obtained in this way is then used to isolate the closest object in the scene and compute its position. A threshold operation individualizes the foreground object, and a consistency check is made to handle false results caused by occluded areas.

Our main contribution, with respect to previous works, is the new hardware architecture for disparity map computation. It is highly flexible, and augmenting disparity range and matrix size is simple and limited only by host platform resources. A novel post-processing stage is also included, which can be useful for several applications.

Acknowledgment

Work partially supported by research contract PTDC/EEAELC/69394/2006 from the Foundation for Science and Technology (FCT), Portugal. The first author was supported by a travel grant from the Masters Program in Electrical and Computer Engineering.

References

[1] P. de la Hamette and G. Tröster, “Architecture and applications of the FingerMouse: a smart stereo camera for wearable computing HCI,” *Personal and Ubiquitous Computing*, vol. 12, no. 2, pp. 97–110, 2008.

[2] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Intl. J. Comput. Vis.*, vol. 47, no. 1, pp. 7–42, Apr. 2001.

[3] J. Banks, M. Bennamoun, and P. Corke, “Non-parametric techniques for fast and robust stereo matching,” in *Proc. IEEE Region 10 Ann. Conf. Speech Image Tech. Comput. Telecomm.*, 1997.

[4] N. W. Bergmann and R. B. Porter, “A generic implementation framework for FPGA based stereo matching,” in *Proc. IEEE Region 10 Ann. Conf. Speech Image Tech. Comput. Telecomm.*, 1997.

[5] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” in *Computer Vision ECCV ’94*, J.-O. Eklundh, Ed. Springer Berlin / Heidelberg, 1994, pp. 151–158.

[6] J. Woodfill and B. Von Herzen, “Real-time stereo vision on the parts reconfigurable computer,” in *Proc. 5th Ann. IEEE Symp. FPGAs Custom Comput. Machines*, 16-18 1997, pp. 201–210.

[7] L. An, Y. Jia, M. Li, and X. Zhang, “A miniature stereo vision machine (MSVM-III) for dense disparity mapping,” *Proc. 17th Intl. Conf. Pattern Recogn.*, 2004.

[8] C. Murphy, D. Lindquist, A. Rynning, T. Cecil, S. Leavitt, and M. Chang, “Low-cost stereo vision on an FPGA,” *15th Ann. IEEE Symp. Field-Programmable Custom Comput. Mach.*, pp. 333–334, 2007.

[9] R. Carrillo, J. Díaz, A. Prieto, and E. Ros, “Real-Time System for High-Image Resolution Disparity Estimation,” *IEEE Trans. Image Process.*, vol. 16, 2007.

[10] K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger, “SAD-based stereo matching using FPGAs,” in *Embedded Computer Vision*, B. Kisacanin, A. Nhattacharya, and S. Chai, Eds. Springer Verlag, 2008, ch. 6.