RASCor: REALTIME ASSOCIATIVE STEREO CORRESPONDENCE

Vikram Simhadri, Premanand Chandramani*, Yusuf Ozturk**
San Diego State University
Department of Electrical and Computer Engineering
simhadri@gmail.com, chandram@attila.sdsu.edu*, yozturk@mail.sdsu.edu**

ABSTRACT

This paper presents a hardware implementation solution to a real time stereo matching problem using system of associative relations (SOAR) computational model. SOAR makes use of pair-wise pixel interactions captured as direction of derivatives to determine the underlying structure of associations within an analysis token. SOAR also offers a similarity measure to assess similarity of two such structures of associations. This paper presents a suitable hardware implementation for real-time stereo correspondence using SOAR structural analysis tokens and similarity measure. The cost of hardware implementation using FPGA is presented along with performance improvements from frequency scaling, unit multiplicity and search depth for stereo matching. Real time stereo matching is achieved through hardware unit multiplicity and frequency scaling.

Index Terms— Stereo Matching, Real Time, Associative Relations.

1. INTRODUCTION

Real time stereo correspondence has been a center of attraction in robotics research and computer vision [1]. Several real time stereo matching systems have been developed in the literature. INRIA implementation [2], based on normalized cross correlation, was implemented on a FPGA consisting of 23 Xilinx logic cell arrays (LCA) and achieved a frame rate of 3.6 frames per second. In 1995, Jet Propulsion Laboratory developed a stereo system using a Datacube MV-200 image processing board and a 68040 CPU board [3] achieving a rate of 1.7 frames per second. The CMU stereo machine [4] implemented a stereo matching method exploiting multibaseline stereo to improve depth estimates. CMU machine consisted of custom hardware and an array of eight TI C40 DSPs. The system was capable of processing 30 frames per second and claimed to be the fastest available system at the time.

Woodfill and Henzen [5] implemented a stereo algorithm using census matching on the custom PARTS engine. The PARTS engine developed at the International Research Corporation was made up of 16 Xilinx 4025 FPGAs, arranged to fit on a standard PCI card. SRI's SVS runs at 30 frames per second on a 700 MHz Pentium III processor, making real time stereo matching possible on the common desktop environment. Porter and Bergmann investigate in [6] the flexibility offered by an FPGA implementation and the suitability of various stereo algorithms for FPGA implementation.

A reconfigurable matrix "VoC" is presented in [7] as a solution to stereo matching. VoC provides a highly parallel implementation of the SAD (Sum of Absolute Distances) metric. The design conceived in [7] implements the SAD computation either for blocks of 7 x 7 pixels or for blocks of 5 x 5 pixels, and has the capability of computing 9 simultaneous SAD for 5 x 5 pixel blocks. The structure of the VoC matrix is presented as a parallel processing unit to be integrated into a reconfigurable DSP which would be a part of a larger image processing system. The FPGA implementation produced a SAD rate of 1.4 billion comparisons per second at peak rate. In 2004, a commercial stereo vision system "DeepSea" was presented [8]. Deepsea is based on the census algorithm. A dedicated silicon implementation known as the "Deep Sea processor" attains a very high disparity rate. The DeepSea system can attain 2.6 billion pixel disparities per second.

Most hardware based solutions reported in the literature are area based methods. A fast model-based stereo matching algorithm is presented in [9]. The algorithm is based on System of Associative Relations (SOAR) [10] architecture. The algorithm proposed in [9], unlike most area-based algorithms, does not depend solely on gray level averages. This algorithm utilizes feature vectors (tokens) formed by direction of derivates [9]. RASCor (Realtime Associative Stereo Correspondence) presented here is based on SOAR algorithm and provides a hardware implementation of SOAR to achieve real time stereo correspondence using video sizes of 512x512 and frame rates of 30 frames/sec.

2. STEREO MATCHING USING SOAR

The System of Associative Relations (SOAR) makes use of the pair-wise pixel associations. The pixels defined by a token over an image area are used to determine associations among pixels. This analysis token is basically an irregular mask that will determine the underlying structure of associations. To explore the associations among pixel elements defined by the token, SOAR maps each pixel to a processing node and sets the internal status of this processing node to be equal to the intensity of that particular pixel.

Suppose pixel intensities in a block (token) are : { V_{ij}; i,j=1,2,….q }. SOAR encodes the larger/smaller information called the "interpixel connection strength" between pixels in this block as the signum of differences as

$$T(i,j;k,l) = \operatorname{sgn}\{V_{ij}, V_{kl}\} \quad \text{where } 1 \le i,j,k,l \le q \quad \text{except: } (i,j) \ne (k,l) \tag{1}$$

In addition to formulating a prescription to encode inter-pixel relationships in a block, an ensemble inter-pixel association over P number of blocks is also formulated in [3] as given in eq.(2).

$$T_P(i,j;k,l) = \sum_{p=1}^{P} \mathrm{sgn}(V_{ij}^{p} - V_{kl}^{p}) \qquad (2)$$

SOAR also defines a similarity measure among the token which has been stored in connections *T(i,j;k,l)* and a new token at another location by:

$$E = \sum_{i}\sum_{j}\sum_{k}\sum_{l} T(i,j;k,l) * \mathrm{sgn}(V_{ij} - V_{kl}) \qquad (3)$$

It is not difficult to deduce from eq. (3) that E will attain its maximum value if the two patterns are correlated with a correlation coefficient of "1." But it will reach its minimum value if the two patterns are correlated with a correlation coefficient of "–1." Finally, E will have a value near zero if the two patterns are not correlated. That is, there is no resemblance between ordering of the pixels within the tokens compared. It can also be seen that the similarity measure given in eq. (4) is a *correlation indicator function* between two specific patterns.

In [9] authors proposed a solution to the correspondence problem in stereo vision using SOAR. The degree of association of pixels within any arbitrary token of the image has been used as the parameter to obtain a match between the left and right images. This study utilizes inherent parallelism in SOAR stereo matching algorithm to offer hardware architecture to achieve real time stereo matching.

3. RASCor ARCHITECTURE

The real time associative stereo correspondence (RASCor) architecture is a custom designed hardware implementation. During design and implementation of RASCor emphasis is given to avoid use of complex logic blocks, and to maximize utilization of resources. The hardware design is split into functional blocks: performing inter-pixel strength computation, and pixel correlation. A data flow through model of the design has been illustrated in Figure 1.

In the data flow model, the stereo images are received by the hardware through the serial input and the disparity map is transmitted back using serial output. The system used for implementation is the Xilinx ML310 FPGA based embedded development platform. The heart of the board is a Virtex II Pro XC2VP30-FF896 FPGA containing dual PowerPC™ 405 processors. The stereo images delivered to the hardware are buffered using multiple memory blocks on MIL310 development platform.

The pixel matching block in Figure 1 is the heart of the system and computes the disparity of the pixels between the left image and right image. The disparity values computed are mapped to a memory blocks and transferred to the host system through serial output.

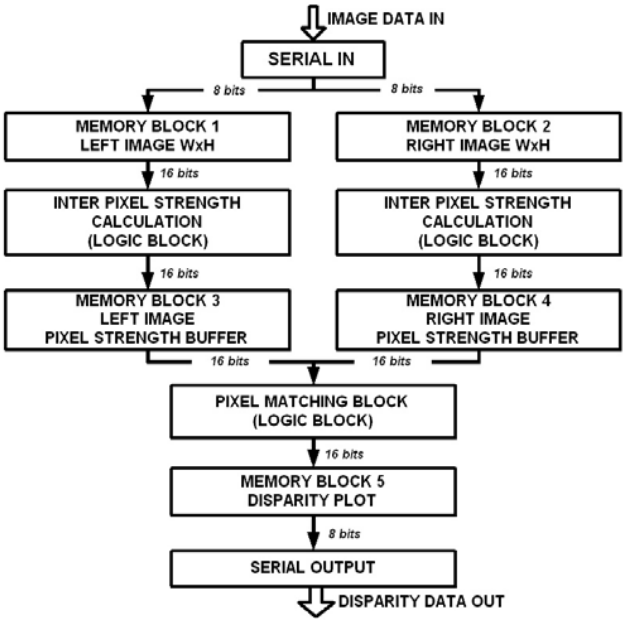


Figure 1. Data Flow Diagram

3.1. The Pixel Matching (PM) Block

The Hardware model of the pixel matching block is illustrated in Figure 2. Two 5x16 bit register arrays form the primary input register set. The output of this block is the 16-bit disparity value. All inputs and output are aligned to 16-bits wide address locations to enable easy accessibility. Two register arrays of 5x16 bits (REG C and REG D) are assigned to hold the IPS (inter-pixel strength) values corresponding to the pixels from the left and right images. Two other 16-bit registers (REG A and REG B) are assigned to hold the position vectors of the left and right pixel. An equivalence gate consisting of eighty 1-bit XNOR logic gates is created and registers REG C (80 bits) and REG D (80 bits) are connected as inputs to the equivalence gate. The resultant 80-bit output is split and stored into five individual 16-bit shift registers which are connected to a 5-bit counter each. The outputs of the five counters are summed up using an adder tree. The result obtained is the measure of similarity between the current left and right pixels. The output of the adder tree is then input to a 7-bit comparator. The second input to the comparator is a feedback from a 7-bit register holding the highest similarity measure after the previous operating cycle. The output of the comparator then acts as a select line to two 2-input multiplexers. One of them is a 7-bit multiplexer; with the '0' input being the feedback line from REG 1 and the '1' input line being the current output from the adder tree.

The second multiplexer is a 16-bit device with the '0' input being a feedback line from the REG 1 and the '1' input coming from a 16-bit subtractor computing REG B – REG A (using equation (16)).

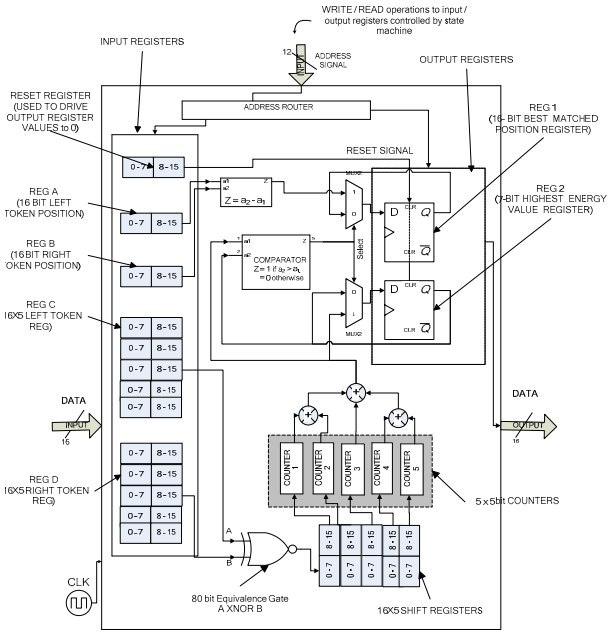


Figure 2. Pixel Matching Block

Based on results obtained from this logic both REG 1 and REG 2 will be updated with the current values (if the comparator output is a '1'). REG1 holds the position of the best pixel and REG2 holds the energy associated with the match. When the search depth is exhausted the values of REG1 and REG2 are output. REG1 indicates the disparity and REG2 indicates the dependability of this decision.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

Table 1 lists the various hardware resources used by the design using two units of IPS blocks, for parallel computation of left and right image IPS values, and one PM block. Adders are added to the PM block, to enable faster counting of '1's while computing the measure of similarity between two pixels.

Table 1 – Hardware resource utilization of the SOAR system

Hardware IP	IPS Block (2 instances)	PM Block
Sequential logic Registers / Latches	356	284
Comparators	160	2
Adders		10
XOR		80

The FPGA resources utilized by the design are listed in Table 2. An FPGA contains configurable (programmable) logic blocks and configurable interconnect between these blocks. A *Configurable Logic Block* (CLB) is a block of logic surrounded by routing resources. CLBs are the functional elements for constructing logic circuits. The Virtex-II Pro CLB is made up of four *slices*; each slice contains 2 Logic Cells (LC). An LC includes a 4-input *Look-Up Table* (LUT), carry logic, and a storage element. A LUT is a function generator with N inputs and one output. A LUT can implement any logic function of its N inputs where N is between 3 and 6; most popular are 4-input LUTs.

Table 2 – FPGA resources utilized by SOAR

FPGA Resources	IPS Block (2 instances)	PM Block	Total
Slices	4968	357	5325
Flip Flops	1720	331	2051
LUTs	9448	642	10090

It can be observed from Table 2, that a major portion of FPGA resources are consumed for implementing logic units for IPS computation. One of the prominent expenses in terms of area is the data width requirement. Because each pixel in an image translates into an 80 bit IPS value, handling of 80 bit data strings is required. The large flip-flop count in the IPS block is due to the shift RAM used for implementing the IPS hardware.

Figure 3 shows a plot of the number of pixels vs. the execution time of IPS and PM units. The slopes of the line and the fact they will never intersect indicates that the IPS values will always be ready for the PM block making the pipeline free of hazards. Therefore, the IPS execution time is completely hidden by the PM processing time. It can be said that the system computation time will be the same as the PM block execution time.

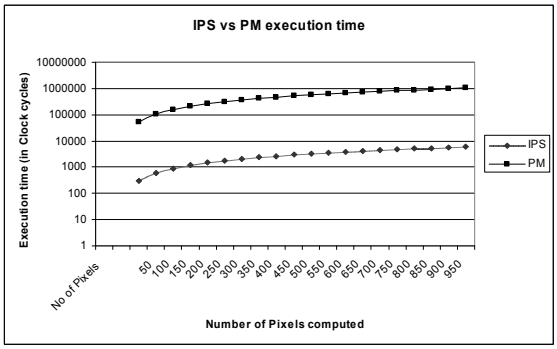


Figure 3. IPS execution time vs. PM execution time

Using two IPS units ,one for left image token and one for right image token, and 1 PM unit the RASCor achieved a processing time of 2903.09 ms for 512 x 512 image sizes using a 9 x 9 square token, a search depth of 64 pixels and a system clock of 100MHz. This amounts to 5.6 million pixels matched per second. For 256 x 256 with the same parameters the processing time per stereo image pair was 697.14 ms. Clearly with one IPS unit per image and one PM, one cannot achieve real time stereo rates. The solution lies in either increasing the clock rate of FPGA, reducing the search depth and/or hardware unit replication.

The throughput obtained from the hardware blocks depend on factors like the input image sizes, the search depth employed for pixel matching and most importantly the clock speed. Only through frequency scaling the processing rate can be reduced to 100 ms for a 256x256 image and 300MHz FPGA clock frequency. The computation time also varies with the search depth employed while computing the disparity of a pixel. The performance versus search depth is given in Table 3 for an image size of 256x256 and token size 9. The time reported is the time in millisecond required to match two images of 256x256 to each other. The near real time video rate is achievable using a 500 MHz clock and a search depth of 16.

Table 3 – Performance vs. Search depth

F in MHz	100	200	300	400	500
Search Depth (S) in pixels	**Time (T) in ms**				
64	622.71	311.35	207.57	155.68	124.54
32	312.87	156.43	104.29	78.22	62.57
16	157.95	78.97	52.65	39.49	31.59

An increase in the search depth required means an increase in the computation time and an increase in the hardware resources required to implement the system. Better throughput can be achieved by replicating some of the hardware components used within the system. One such option is to replicate the hardware components used to compute the equivalence energy, thereby reducing the search steps by increasing the number of equivalence units. In Table 4, the time to match two images is given for image sizes of 512x512 and 256x156 using a token size of 9 and FPGA clock frequency of 200 MHz. The number of search steps is set to 64. As the results presented in this table indicates real time performance is achievable for an image size of 512x512 and a search size of 64 by replicating equivalence blocks.

Table 4 – Performance vs. number of equivalence components

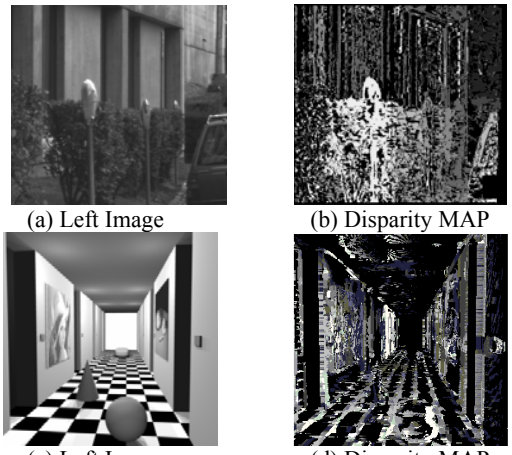
Image Width (M) in pixels	**512**	**256**
Number of equivalence energy components	**Time T in ms**	
1	1296.56	311.35
4	651.43	156.43
32	86.94	20.88
64	26.46	6.35

However the gain in performance is marred by the large increase in hardware resources required since each equivalence component requires an IPS block. IPS computation as reported in Table 2 accounts for most of the resource consumption. A less drastic approach would be to replicate the PM unit itself so as to be able to process two rows and columns of pixels in parallel. In Table 5 , performance improvements versus PM block replication is given for a clock rate of 200 MHz and search depth of 32 using a token size of 9.

Table 5 – Performance with PM unit replication

Image size	**512x512**	**256x256**
Number of PM Units	**Time T in ms**	
1	651.43	156.43
4	162.86	39.11
8	81.43	19.55

The disparity images for two well studied images are given in Figure 5 using SOAR stereo matching algorithm software implementation (a and b) and the hardware implementation (c and d). Please note that the hardware and software implementations are identical and thus should produce same results.



(a) Left Image (b) Disparity MAP

(c) Left Image (d) Disparity MAP

Figure 4: Disparity MAPs for Park and Synthetic images

5. REFERENCES

[1] M.Z. Brown, D. Burschka, G.D. Hager, "Advances in Computational Stereo", IEEE Transactions on Pattern Analysis and Machine Intelligence, August 2003, pp. 993-1008, Vol. 25, No. 8.

[2] O. Faugeras et.all, "Real Time Correlation-Based Stereo: Algorithm, Implementations and Applications," INRIA Technical Report, RR 2013, 1993. Number - RR 2013.

[3] L. Matthies, A. Kelly, T. Litwin, and G. Tharp, "Obstacle Detection for Unmanned Ground Vehicles: A Progress Report," Proceedings of IEEE Intelligent Vehicles `95 Conference, September, 1995, pp. 66 – 71, Month: September.

[4] S. Kimura, T. Kanade, H. Kano, A. Yoshida, E. Kawamura, and K. Oda, "CMU Video-Rate Stereo Machine," Mobile Mapping Symposium, May 24-26, 1995, Columbus, OH.

[5] K. Muhlmann, D. Maier, J. Hesser, and R. Manner, "Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation," Proc. IEEE Workshop Stereo and Multi-Baseline Vision, 2001, pp. 30-36.

[6] Reid B. Porter Neil W. Bergmann , "A generic implementation framework for fpga based Stereo Matching", TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications'., Proceedings of IEEE, 2-4 Dec 1997, pp. 461-464, Vol 2.

[7] Jacobi, R.P, Cardoso, R.B., Borges, G.A, "VoC: a reconfigurable matrix for stereo vision processing", 20th International Parallel and Distributed Processing Symposium, 2006. (IPDPS 2006), 25-29 April 2006, pp. 6.

[8] Woodfill J.I., Gordon, G, Buck, R, "Tyzx DeepSea High Speed Stereo Vision System", Conference on Computer Vision and Pattern Recognition Workshop, 2004, 02 June 2004, pp. 41 – 41.

[9] Yusuf Ozturk and Arvind Sridharan, "FAST MODEL BASED STEREO MATCHING USING SOAR", 2004 International Conference on Image Processing, 24-27 Oct. 2004, pp. 1337 – 1340, Vol.2.

[10] Y. Ozturk, H. Abut, "SOAR: System of Associative Relations", Thirty-First Asilomar Conference on Signals, Systems & Computers, 2-5 Nov. 1997, pp. 668 – 672, Vol.1