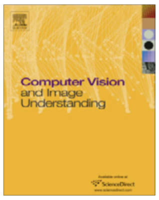
Computer Vision and Image Understanding 114 (2010) 1126–1138

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



Leveraging cost matrix structure for hardware implementation of stereo disparity computation using dynamic programming

W. James MacLean [a,*], Siraj Sabihuddin [a], Jamin Islam [b]

[a] University of Toronto, Department of Electrical & Computer Engineering, Toronto, Canada M5S 3G4
[b] Ryerson University, Department of Electrical & Computer Engineering, Toronto, Canada M5B 2K3

ARTICLE INFO

Article history:
Received 19 December 2008
Accepted 17 March 2010
Available online 29 April 2010

Keywords:
Stereo disparity
Dynamic programming
Field programmable gate arrays
Pipeline processing
Real-time systems
Hardware

ABSTRACT

Dynamic programming is a powerful method for solving energy minimisation problems in computer vision, for example stereo disparity computations. While it may be desirable to implement this algorithm in hardware to achieve frame-rate processing, a naïve implementation may fail to meet timing requirements. In this paper, the structure of the cost matrix is examined to provide improved methods of hardware implementation. It is noted that by computing cost matrix entries along anti-diagonals instead of rows, the cost matrix entries can be computed in a pipelined architecture. Further, if only a subset of the cost matrix needs to be considered, for example by placing limits on the disparity range (include neglecting negative disparities by assuming rectified images), the resources required to compute the cost matrix in parallel can be reduced. Boundary conditions required to allow computing a subset of the cost matrix are detailed. Finally, a hardware solution of Cox's maximum-likelihood, dynamic programming stereo disparity algorithm is implemented to demonstrate the performance achieved. The design provides high frame rate (>123 fps) estimates for a large disparity range (*e.g.* 128 pixels), for image sizes of 640 × 480 pixels, and can be simply extended to work well over 200 fps.

1. Introduction

Dynamic programming is an optimisation algorithm that exploits *optimal substructure* to greatly reduce the time required to compute an optimal solution [1]. It has numerous applications, such as biosequence matching (including DNA sequencing), the Viterbi algorithm for estimation of hidden Markov models (HMMs), and stereo disparity estimation in computer vision. Many problems that can be solved with dynamic programming involve matching data from two streams, such as the left- and right-image pixels from scanlines in stereo images, or problems involving matching string subsequences from two target strings. It should be noted that other problems share this structure, notable examples are the matrix-chain multiplication problem, the Needleman–Wunsch algorithm for biosequence matching, and the Viterbi algorithm for estimating hidden Markov models. This paper explores a dynamic programming approach applied to the stereo matching problem, and gives a sample implementation using FPGAs.

Fast, efficient implementations for dynamic programming problems are of interest, especially in cases where the data rate is high, as it is in image processing. A number of researchers have made attempts to accelerate dynamic programming algorithms using hardware and/or special processors [2–6]. Much of this work has involved protein and DNA sequence matching [2–5] using an "edit-distance" cost function for string comparison [7] which shares the same anti-diagonal parallelism that we exploit in our approach. In [2] the anti-diagonal structure is used to pipeline comparison of multiple target strings against a reference string, but each cost matrix is computed in a row-wise manner. Martins et al. [4] point out the challenges of adapting processing to the changing sizes of the anti-diagonals, and propose a block-wise anti-diagonal approach on a parallel processor architecture to ensure good processor utilisation. Anvik et al. [5] identify the anti-diagonal cost matrix structure as a *design pattern* and introduce a framework generator that minimises the development effort required to adapt this design pattern to specific applications (they demonstrate three such applications) running on a small array of four processors. They use a block-wise approach similar to that of [4]. Two of these approaches involve implementation on FPGAs [3,6], with both being tested in simulation only. The approach of [3] uses multiple FPGAs,[1] but does not re-map the cost matrix as we do, resulting in half of the processing elements being idle on any given cycle. Hoang and Ayala-Rincón et al. [3,6] both use systolic arrays in their processing architecture, although it is

* Corresponding author.
E-mail address: james.maclean@utoronto.ca (W.J. MacLean).
URL: http://www.wjamesmaclean.net/ (W.J. MacLean).

[1] At the time their paper was written, FPGA devices had much smaller capacity.

1077-3142/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2010.03.011

pointed out that adapting such arrays for computing only a diagonal-band of the cost matrix (as we do by limiting the disparity range D) is challenging.

Stereo matching has been extensively studied by the computer vision community. Detailed reviews of existing algorithms have, thus, also been published by a number of researchers. Gong et al. [8] provide an overview of general approaches that utilise sparse, dense, volumetric or level-set algorithms. Furthermore, Seitz et al. [9] provide a review of multi-view matching methods. Some image registration techniques also utilise sparse, feature based matching and are discussed by Zitova and Flusser [10].

The dynamic programming solution presented in this paper is the "Dynamic Programming Maximum Likelihood" (DPML) stereo disparity algorithm by Cox et al. [11]. It is a global and dense disparity estimation algorithm—Scharstein and Szeliski [12] and Brown et al. [13] provide extensive discussions and comparisons of similar dynamic programming approaches. These comparisons demonstrate that dynamic programming provides accurate estimates that compete well even with the best of existing matching methods. More accurate approaches do exist, however, they tend to operate at significantly lower speeds (see [14]). Closely related to the studies by Scharstein and Brown, Lu et al. [15] provide a further survey on cost aggregation for matching problems.

In current literature, the most common approaches towards FPGA hardware stereo matching are based on correlation or area-based methods. Of these, the most common methods make use of SAD aggregated cost functions applied to pixel intensities. As shown by Scharstein and Szeliski in [12] SAD correlation approaches do not provide very accurate disparity estimates. Typically these implementations make use of line buffering to align pixels in a stereo pair for parallel windowing computations. Works by Miyajima and Maruyama [16], Hariyama et al. [17], Perri et al. [18], Mitéran et al. [19] [7] and Han et al. [20] all utilise such buffering. Both Perri et al. [18] and Mitéran et al. [19] observe that neighbouring windows of SAD computations use many of the same values. These values are stored and re-used as required. Hariyama et al. [17] define two levels of parallelism to perform coarse to fine refinement of disparities within localised regions, and thus improve the accuracy. Both Hariyama et al. [17] and Simhadri et al. [21] utilise adder trees to perform cost and comparison computations. It is worth noting that adder trees are primarily useful in situations such as windowing—they do not typically apply to dynamic programming type solutions. Lee et al. [22] present an FPGA-based SAD stereo algorithm running at 31 fps for 640×480 images with a disparity of 64 pixels, although no quantitative accuracy results are given. Most of these implementations produce results at speeds of approximately 30 fps, although some go as high as 150 fps on smaller images. Typically, papers that present these solutions do not present an analysis of accuracy of their algorithms.

Additional hardware methods use phase based correlation. Díaz et al. [23], Darabiha et al. [24] and Masrani and MacLean [25] provide examples of these approaches implemented in hardware. Phase based methods have the advantage of producing significantly better results than SAD algorithms, obtaining results that compete well with dynamic programming solutions. The problem with these methods lies in the square root computations required—these computations are difficult to implement in hardware and come with an associated high resource cost, especially when implemented in parallel. Both Darabiha et al. [24] and Masrani et al. [25] provide an implementation that uses multi-scale Locally Weighted Phase Correlation (LWPC). Like SAD based implementations these phase based approaches achieve approximate 30 fps performance. Díaz et al. [23] demonstrate an algorithm that achieves over 200 fps performance but does so by reducing the search range to only four neighbouring pixels.

A final approach that produces very high frame rates (over 200 fps), on par with the approach presented in this paper, makes use of the census algorithm (see Woodfill et al. [26]). The census algorithm computes costs within a pixel neighbourhood using the census transform—the transform essentially utilises hamming distance based comparisons. The high frame rates and very simple computations associated with this approach come with poor accuracy. Murphy et al. [27] present a "low-cost" implementation of the census algorithm on a Xilinx Spartan-3 FPGA, with performance of 40 fps and a disparity range of 20 pixels, with no accuracy results presented.

To date, no hardware implementations of dynamic programming algorithms appear to have been explored by the vision community. Furthermore most existing solutions produce relatively low frame rates of approximately 30 fps at resolutions that are typically lower than 640×480 pixels. They achieve higher frame rates by sacrificing on accuracy, resolution or disparity search range. This paper explores a hardware based dynamic programming solution that achieves high frame rate performance with no compromise on accuracy and limited compromise on disparity search range or resolution. The solution provides good scaling characteristics relative to SAD and phase based correlation approaches.

The rest of this paper is structured as follows. In Section 2 we describe the structure inherent in a dynamic-programming cost matrix and how this structure can be exploited to design efficient hardware. In Section 3 we describe a hardware implementation of a dynamic programming algorithm for stereo disparity estimation, showing how the cost matrix structure leads to a fast yet efficient hardware design. Finally, in Section 4, we give performance results for the stereo hardware implementation. Further technical details of the implementation can be found in [28–30].

2. Cost matrix structure

When solving a dynamic programming problem using a bottom-up approach, a typical approach is to build a cost matrix, denoted by C. In this matrix element C_{ij} represents the optimal cost of associating two data elements $I_L(i)$ and $I_R(j)$. This paper will concentrate on computing stereo disparity using these data elements. Note that $I_L(i)$ represents the ith pixel from the left scanline and $I_R(j)$ the jth pixel from the right scanline. Both $I_L(i)$ and $I_R(j)$ lie within the same scanline (vertical position) in the left and right images.

In computing the cost matrix, an element C_{ij} depends on those neighbouring elements with lower indices. In computing an optimal cost for element C_{ij}, one assumes that optimal costs have already been computed for the elements $C_{i-1,j}$, $C_{i,j-1}$ and $C_{i-1,j-1}$, and computes the optimal cost for the current element in terms of these. These cost dependencies are shown in Fig. 1. In a traditional software implementation it is typical to do this row-wise with a nested loop structure, which computes elements left-to-right along each row, assuming the results from the previous row are complete.

This approach assumes that all the data required to compute an entire row of C is already present. In the case of stereo disparity computations, this means that the cost value for a particular combination of right/left pixel positions (i and j) in a scanline cannot be computed unless all previous dependant input pixels have been received and their respective costs computed. Even if, for a particular problem, all the data are available at the outset, the computations must be done sequentially as each element on the row depends on the ones before it. While this is not a problem for software designed to run on a serial processor, it hampers efforts to parallelise cost computations in hardware. It does allow for partial storage of C, as once a row of C has been computed, only the current row is