

# Parallel Stereo on Fixed Size Arrays using Zero Crossings \*

Pani N. Chakrapani

Department of Computer Science  
University of Redlands  
Redlands, CA 92373-0999

Ashfaq A. Khokhar and Viktor K. Prasanna

Department of EE-Systems  
University of Southern California  
Los Angeles, CA 90089-2562

## Abstract

In this paper, we present a processor-time optimal partitioned implementation of a stereo matching algorithm using zero crossing points as matching primitives. We provide  $O(nm/P)$  time algorithm on  $\sqrt{P} \times \sqrt{P}$  processor mesh array, where  $n$  is the number of zero crossing points in the left image,  $m$  is the set of possible candidate points in the right image for a given zero crossing point, and  $1 \leq P \leq n$ . The sequential algorithm takes  $O(nm^2)$  time while the faster sequential algorithm shown in this paper runs in  $O(nm)$  time.

## 1 Introduction

Stereo matching is one of the well known methods for extraction of depth information. Two images, left and right image, captured at the same time but at different angles are matched. Several sequential techniques have been proposed for stereo matching [1, 2, 8]. One of the well known methods for stereo matching is using zero crossing patterns [6]. The algorithm combines three disambiguating constraints, namely, continuity of disparity, figural continuity, and smoothness of the probability of matching. In the past, several implementations of various stereo matching algorithms have been derived [4, 5, 9, 2, 7]. These implementations differ in terms of matching primitives used and target architectures employed in parallel solutions.

In this paper, we propose  $O(nm)$  time sequential algorithm which is a faster version of the algorithm proposed in [6], where  $n$  corresponds to the number of zero crossing points in the left image and  $m$  is the set of possible candidates in the right image that match a given zero crossing point. Also, a processor-time optimal partitioned implementation of the sequential algorithm of time complexity  $O(nm/P)$  is presented.

The organization of this paper is as follows. In Section 2, a model of the architecture used for our implementations is described. Section 3 provides parallel implementation of the stereo matching algorithm. Conclusions are presented in Section 4.

## 2 Fixed Size Mesh Array

A *fixed size mesh array* is a two dimensional array of  $P \times P$  processors, where  $P^2$  is less than or equal to the problem size. The processors are connected

through bidirectional local links and the array operates in SIMD mode. Each processor  $PE_{ij}$  is connected to  $PE_{i+1j}$ ,  $PE_{i-1j}$ ,  $PE_{ij-1}$ , and  $PE_{ij+1}$ , if they exist. A memory plane of  $P \times P$  memory modules (MMs) is provided such that each memory module is connected to exactly one processor in the array. Each  $PE_{ij}$  is attached to memory module  $MM_{ij}$  to store the relevant data.

Following assumptions are made regarding computations in this model:

- Each arithmetic/logic operation performed in a PE takes  $O(1)$  time.
- Each access by  $PE_{ij}$  to memory module  $MM_{ij}$ ,  $0 \leq i, j \leq P - 1$ , takes  $O(1)$  time.
- A unit data transfer between adjacent PEs takes  $O(1)$  time.

## 3 Stereo Matching on a Fixed Size Mesh Array

Various stereo matching algorithms differ with respect to the primitives used for matching [1, 2, 8]. Each technique has its own advantages and disadvantages. The stereo matching algorithm presented here uses two independent measures of similarity namely zero crossing pattern and intensity gradient. The matching process uses a relaxation method based on the continuity of local disparities, the similarity of the zero crossing patterns and the smoothness of the probability of matching. Following the terminology in [6], the matching process associates each zero crossing point in the left image  $Z_l(x_i, y_i)$  with a point in the right image  $Z_r(x_j, y_j)$ . In the parallel axis geometry, the search process for matching is applied only on the left side of the transferred coordinates of the candidate point [6]. The search for correspondence is constrained by the epipolar line and is determined by the maximum disparity value (denoted as  $d_{max}$  in Fig. 1). The  $d_{max}$  value can be computed to suit the application. In the following section, for the sake of completeness, the matching process is described. The details can be found in [6].

### 3.1 Sequential Stereo Matching Algorithm

The collection of locations of all the zero crossing points which do not form a horizontal zero crossing pattern in the left image forms the set of nodes  $n_i$  for the relaxation process. Each node  $n_i = (x_i, y_i)$  has

\*This research was supported in part by NSF under grant IRI-9145810 and in part by DARPA and AFOSR contracts F-49260-89-C-0126 and F-49620-90-C-0078.

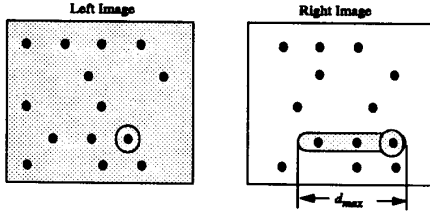


Figure 1: Finding a Candidate Match

a set of labels  $D_i$  which consist of a set of possible disparity values  $\{d_j\}$  and each label is associated with a probability  $P_i(d_j)$  that the node  $n_i$  has disparity value  $d_j$ . Initially the set of labels for each node is established by finding all the candidate points for each zero crossing point in the left image. Once a set of labels is associated with a node, to assign the initial probability to each label, two weighting functions are used. One of the weighting functions is based on the similarity of the zero crossing point and the other is based on the difference in the gradients of the grey level intensity. These weighting functions  $w_{j1}$  and  $w_{j2}$  are computed as follows. Let  $I_l(x, y)$  and  $I_r(x, y)$  be the image functions at point  $(x, y)$  of the left and right images respectively, and  $DP_{ij}$  be the directional difference between  $ZP_l(x_i, y_i)$  and  $ZP_r(x_j, y_j)$ . Then, for label  $d_j$  in  $D_i$ , we have,

$$w_{j1} = \frac{1}{1 + |DP_{ij}|}$$

$$w_{j2} = \frac{1}{1 + |G_l(x_i, y_i) - G_r(x_j, y_j)|}$$

where

$$G_l(x_i, y_i) = \frac{[I_l(x_{i+1}, y_i) - I_l(x_{i-1}, y_i)]}{2}$$

$$G_r(x_i, y_i) = \frac{[I_r(x_{j+1}, y_j) - I_r(x_{j-1}, y_j)]}{2}$$

and the total weight is assigned as follows:

$$W_j = a * w_{j1} + b * w_{j2}$$

where  $a$  and  $b$  are positive constants ( $a + b = 1$ ) which affect the influences of  $w_{j1}$  and  $w_{j2}$  on the total weight  $W_j$ .

Let us suppose that  $D_i$  has  $m$  labels and the  $j$ th label has weight  $W_j$ ; then the initial probability that node  $n_i$  has disparity value  $d_j$  is given by

$$P_i^0(d_j) = \frac{W_j}{W_* + \sum_{k=1}^m W_k}, \quad j = *, 1, \dots, m$$

```

repeat
  change ← 0;
  Compute_Weights(zero-crossings
    in left and right images);
  Update_Probabilities();
  Vote_for_the_match();
until (change = 0)

```

Figure 2: Sequential Algorithm for Stereo Matching Using Zero Crossings as Matching Primitives

where  $W_*$  is the weight of no match which is calculated by  $W_* = 1 - \max(W_j)$ . The initial probabilities which are assigned to all the labels of  $D_i$  are modified using an iterative updating scheme. Let  $P_i^k(d_j)$  be the  $k$ th iterated probability of the point  $Z_l(x_i, y_i)$  whose first and second connected zero crossing points be  $Z_l(x_f, y_f)$  and  $Z_l(x_s, y_s)$  respectively. Let  $P_f^k(d_j)$  and  $P_s^k(d_j)$  be the probabilities at locations  $(Z_l(x_f, y_f), d_j)$  and  $(Z_l(x_s, y_s), d_j)$  respectively. Then,

$$P_i^{k+1}(d_j) = P_i^k(d_j) + c * F(P_i^k(d_j)) * P_s^k - d * P_i^k(d_j) * I(P_{FS})$$

$$P_i^{k+1}(d_*) = P_i^k(d_*)$$

where

$$P_F^K = \max[P_f^k(d_j - 1), P_f^k(d_j), P_f^k(d_j + 1)]$$

$$P_S^K = \max[P_s^k(d_j - 1), P_s^k(d_j), P_s^k(d_j + 1)]$$

$$F(P_i^k(d_j)) = \begin{cases} [P_i^k(d_j)]^2 & 0 < P_i^k(d_j) \leq 0.5 \\ P_i^k(d_j) * (1 - P_i^k(d_j)) & 0.5 < P_i^k(d_j) < 1 \end{cases}$$

$$I(P_{FS}) = \begin{cases} 0, & P_F^K + P_S^K \neq 0 \\ 1, & P_F^K + P_S^K = 0 \end{cases}$$

The algorithm terminates after a constant number of iterations and each iteration takes  $O(nm^2)$  time. Each time unit corresponds to a simple arithmetic/logic operation.

### 3.2 A Fast Sequential Algorithm

The update probability procedure presented in Fig. 4 takes  $O(nm^2)$  time. A faster sequential algorithm for the update procedure is shown in Fig. 6. The speed-up is achieved by computing the  $P_F^K$  and  $P_S^K$  values of the neighboring matches independent of each other. Due to space limitations, we omit the details and concentrate on the parallel implementation. The details of this work can be found in [3].

```

Compute_Weights(N, M)
for i = 1 to N do
  Wk[i] ← 0
  for j = 1 to m such that j is a candidate
    match point of i do
      compute Gl, Gr
      compute Wj1, Wj2, Wj
      if Wj > Max[i][j] then
        Max[i][j] ← Wj
      Wk[i] ← Wk[i] + Wj
    Next j
  Compute initial Probability Pi0[j]
Next i
end

```

Figure 3: Algorithm for Computing Initial Weights for Candidate Matches

```

Update_Probabilities()
for i = 1 to n do
  for j = 1 to m such that j is a candidate
    match point of i do
      compute PFk and PSk
      compute I(PFS)
      compute F(Pik)
      compute Pik+1
    Next j
  Next i
end

```

Figure 4: Algorithm for Updating Probabilities of Candidate Matches

```

Vote_for_the_match()
for i = 1 to N do
  for j = 1 to m such that j is a candidate
    match point of i do
      if Pik < 0.05 then
        Discard the pair from the match list
      else if Pik > 0.7 then
        Accept the pair as matched and
        remove i from future iterations.
      Next j
    Next i
  end
end

```

Figure 5: Algorithm for Voting for Candidate Matches

```

Faster_Update_Probabilities()
for i = 1 to n do
  compute PFk and PSk;
  for i = 1 to n do
    for j = 1 to m such that j is a candidate
      match point of i do
        compute I(PFS)
        compute F(Pik)
        compute Pik+1
      Next j
    Next i
  end
end

```

Figure 6: Faster Algorithm for Updating Probabilities of Candidate Matches

```

repeat
  change ← 0;
  Parallel_Compute_Weights(zero-crossings
    in left and right images);
  Parallel_Update_Probabilities();
  Parallel_Vote_for_the_match();
until (change = 0)

```

Figure 7: Parallel Algorithm for Stereo Matching Using Zero Crossings as Matching Primitives

### 3.3 A Partitioned Parallel Implementation on a Fixed Size Mesh Array

In this section, we present a partitioned parallel implementation of the sequential algorithm shown in Fig. 6. The parallel algorithm is shown in Fig. 7. Each PE<sub>*i,j*</sub> in the array is assigned to find a match for  $n/P$  zero crossings points in the left image. The zero crossings, computed both in left and right images, are stored in snake-like row major order in the processor array. A PE is marked as a leader if it contains a right-most zero crossing point of a row in the right image. The memory module MM<sub>*i,j*</sub>, associated with PE<sub>*i,j*</sub> contains DP<sub>*i,j*</sub> intensity values for the corresponding zero crossing points in right and left images, and a list of  $m$  candidate matches for each of the  $n/P$  points assigned to it.

An initialization procedure, "P\_Compute\_Weights()", is executed in each PE to compute initial probability for each of the candidate points. This can be performed in  $O(nm/P)$  time. Next, during each iteration of the procedure, "Parallel\_Update\_Probabilities()", the initial probabilities of candidate matches are updated based on the neighboring  $O(m)$  points. The procedure "distribute()" is assigned to route the required data. This procedure can be executed in a pipelined fashion in each PE in  $O(nm/P)$  time. The PEs marked as "leaders" terminate the pipeline process initiated by the "distribute()" procedure. The claim that the pro-

```

P_Compute_Weights(N, M)
  in parallel for each i in the left
    image,  $0 \leq i \leq n - 1$  do
       $W_k[i] \leftarrow 0$ 
      compute  $G_i$ 
  parallel_end

  in parallel for each j in the right
    image,  $0 \leq j \leq n - 1$  do
      compute  $G_r$ 
      distribute( $G_r$ )
  parallel_end

  in parallel for each i in the left
    image,  $0 \leq i \leq n$  do
      for j = 1 to m such that j is a candidate
        match point of i do
          compute  $W_{j1}, W_{j2}, W_j$ 
          if  $W_j > \text{Max}[i][j]$  then
             $\text{Max}[i][j] \leftarrow W_j$ 
             $W_k[i] \leftarrow W_k[i] + W_j$ 
          Next j
      Compute initial Probability  $P_i^0[j]$ 
  parallel_end

```

Figure 8: Parallel Algorithm for Computing Initial Weights for Candidate Matches

```

Parallel_Update_Probabilities()
  in parallel for each i in the left
    image,  $0 \leq i \leq n - 1$  do
      compute  $\max(P_i^k(d_j - 1), P_i^k(d_j), P_i^k(d_j + 1))$ 
      distribute( $\max, F, S$ )
  parallel_end

  in parallel for each i in the left
    image,  $0 \leq i \leq n - 1$  do
      for j = 1 to m such that j is a candidate
        match point of i do
          compute  $I(P_{FS})$ 
          compute  $F(P_i^k)$ 
          compute  $P_i^{k+1}$ 
        Next j
  parallel_end

```

Figure 9: Parallel Algorithm for Updating Probabilities of Candidate Matches

cedure “distribute()” takes  $O(nm/P)$  is supported by the assumption that a parallel axis geometry is used and a zero crossing in the left image has its match, if any, within a distance of  $d_{max}$  (approximately  $m$  points) and is present on the same epipolar line. After each iteration of the update procedure, voting for each candidate point is performed in  $O(nm/P)$  time. The algorithm terminates after a constant number of iterations of update and voting procedures [6].

#### 4 Conclusions

We have presented a processor-time optimal parallel implementation for stereo matching problem based on zero crossing on fixed size mesh arrays. Several sequential approaches to the stereo matching problem have been investigated [1, 2, 3, 5, 7, 8]. These solutions vary mainly in terms of the primitives used for matching. An extensive work is needed to consolidate these approaches and provide a common framework for parallel stereo matching problems.

#### References

- [1] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *International Joint Conference on Artificial Intelligence*, pages 631–636, Vancouver, Canada, August 1981.
- [2] S. T. Barnard. Stochastic stereo matching over scale. In *DARPA Image Understanding Workshop*, pages 769–778, Cambridge, MA, April 6–8 1988.
- [3] A. Khokhar, *Ph.D. thesis*, in preparation, Dept. of EE-Systems, University of Southern California, Los Angeles.
- [4] A. Khokhar, W. Lin, and V. K. Prasanna. Stereo and image matching on fixed size mesh arrays. In *International Conference on Computer Architectures and Machine Perception*, Paris, France, December 1991.
- [5] A. Khokhar and V. K. Prasanna. Parallel algorithms for stereo and image matching. In *DARPA Image Understanding Workshop*, pages 1057–1062, San Diego, CA, January 1992.
- [6] Y. C. Kim and J. K. Aggarwal. Positioning 3-d objects using stereo images. *IEEE Transactions on Robotics and Automation*, RA-3(4):361–373, Aug 1987.
- [7] A. F. Laine and G. C. Roman. A parallel algorithm for incremental stereo matching on SIMD machines. *IEEE Transactions on Robotics and Automatic*, 7(1):123–134, February 1991.
- [8] G. Medioni and Nevatia. Segments-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.
- [9] C. Reinhart and R. Nevatia. Efficient parallel processing in high level vision. In *DARPA Image Understanding Workshop*, pages 829–839, September 1990.