CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. LLuch (Editors)

Realtime Dense Stereo Matching with Dynamic Programming in CUDA

John Congote[1,2], Javier Barandiaran[2] Iñigo Barandiaran[2], Oscar Ruiz[1]

[1]EAFIT University, CAD/CAM/CAE Laboratory, Medellin - Colombia
[2]VICOMTech Research Center, San Sebastian - Spain

Abstract

Real-time depth extraction from stereo images is an important process in computer vision. This paper proposes a new implementation of the dynamic programming algorithm to calculate dense depth maps using the CUDA architecture achieving real-time performance with consumer graphics cards. We compare the running time of the algorithm against CPU implementation and demonstrate the scalability property of the algorithm by testing it on different graphics cards.

Categories and Subject Descriptors (according to ACM CCS): ARTIFICIAL INTELLIGENCE [Computing Methodologies]: I.2.10—Vision and Scene Understanding3D/stereo scene analysis

1. Introduction

Dense depth map calculation is a common problem in computer vision that tries to recover three-dimensional information from two-dimensional images. This problem has been widely studied and diverse approaches to solve it have been proposed [WZ08] [KSK06]. Furthermore, the problem had been tracked since 2002 by Scharstein and Szeliski [SS02] who had also defined a taxonomy for stereo vision algorithms dividing them into local, global and scan-line, or semi-global methods. Depth map calculation is very useful in the area of robotics, 3D scene reconstruction, and in the emerging field of 3D television.

Real-time (**RT**) and near real-time algorithms to generate depth maps have improved substantially in recent years due to the following two factors. First, research in the field has developed new algorithms which reduce the complexity of calculations and the increase in computational power required by the **CPU**. Nowadays, these advances have been more dramatic with Graphic Processing Unit (**GPU**) [OJL*07]. New technologies such as Compute Unified Device Architecture (**CUDA**) opens a new promising field of work for these kind of problems. Recently, much of the research has been carried out on the implementation of different depth estimation algorithms in **GPU**.

Local methods are more straightforward to implement using **GPU**. These methods were measured and compared by Gong [GYWG07] and Tombari [TMDSA08]. Also global methods were implemented in **GPU**, as shown by Gibson [GM08] and Yang [YWY*06]. The highest quality results are achieved by using methods based on Belief Propagation (**BP**) [KSK06]. However, these methods are computationally intensive. Methodologies that shows a good balance between velocity and quality are algorithms that use a combination of a local method, dynamic programming and a postprocessing step, as showed by Wang [WLG*06]. Our proposal is based mainly on this work.

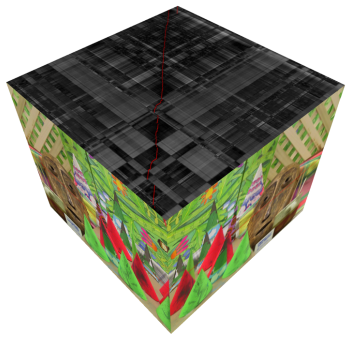


Figure 1: *Calculated Matrix M_{190} of the cones model, and the solution obtained with the **DP** algorithm*

© The Eurographics Association 2009.

Dynamic Programming (**DP**) has been a difficult problem to be solved by using parallel architectures, as shown by Galil [GP91]. A useful practical implementation of the **DP** algorithm for depth map calculation has been attempted in the past by Gong [GY05] without achieving significant improvements over comparative **CPU** implementations.Recently, some implementations of similar (**DP**) algorithms have been implemented in **GPU**, Manavski [MV08], obtaining good results.

The contribution of this paper is the implementation of the **DP** algorithm in **GPU** using the Nvidia **CUDA** architecture. For benchmarking we use a comparable **CPU** implementation as a baseline and measure the scalability of the **GPU** implementation across multiple **GPU** configurations. The results prove that the algorithm can be successfully implemented on inexpensive consumer graphics cards, obtaining real time performance,

The paper is structured as follows: The basics of the Nvidia **CUDA** architecture shall be described in section (2). The original algorithm for the calculation of the dense depth map is explained in section (3). The parallelisation methodology applied in this paper is detailed in section (4). The results of the quality of the algorithm and a comparison of running times for each configuration are shown in section (5). Finally, some conclusions are presented in section (6).

2. CUDA

Graphics processor is a new processing unit that is present in any personal computer nowadays. These processors are optimised for parallel processing and have represented a major step in 2D and 3D graphics generation and visualization. The evolution of this kind of hardware in recent years has opened new possibilities for complex computations that were restricted in the past because of the excessive time required for execution of some algorithms making them impractical. General Purpose computation on **GPU**s (**GPGPU**) is a new programming paradigm which generates new kinds of algorithms that can be processed in the graphics units via new frameworks such as **CUDA** from Nvidia. This framework provides a new subset of C written primitives for parallel computation.

CUDA framework allows the execution of parallel threads which are grouped in blocks. Each thread has access not only to a global memory but also to some other faster memory spaces such as texture, shared or cache memory, which can be used to improve the running time of the algorithm. These features give the possibility of an effective implementation of the **DP** algorithm in the **GPU**. This work was a challenging undertaking as explained by Gong [GY05].

3. Dynamic Programming

Dense depth map calculation is a process (See figure 2), in which, given two images I_l and I_r each pixel of I_l is matched to a pixel in I_r. In our case the match can be expressed as a displacement for each pixel $p(x,y) \in I_l$ to a pixel $q(x',y') \in I_r$. Images need to be previously rectified so that the epipolar lines of the images are coincident with the scan-lines. In a rectified image each pixel from the I_l could have their match in the same line in the I_r and viceversa. Therefore, a pixel match between I_r and I_l only differs in a horizontal displacement, $d = x - x'$. In stereo techniques this displacement is known as *disparity*. The output of any dense stereo algorithm is an structure known as *disparity map* where for each pixel in I_l we have a disparity value d.

The problem of dense depth map estimation is *NP-hard*. Approximations used to solve it is by generating a cost function for the estimation. This cost function is defined per pixel and the differences between the neighbouring pixels could also be used. The cost per pixel is normally calculated by an aggregation cost function such as **SAD** defined in equation (1) as the sum of the absolute differences of the matched pixels colour in **RGB**. The problem of selecting optimal matches between pixels is solved by **DP**. The **DP** approach returns an optimal match for each scan line of the images. but the differences between scan lines is a typical problem with this method.

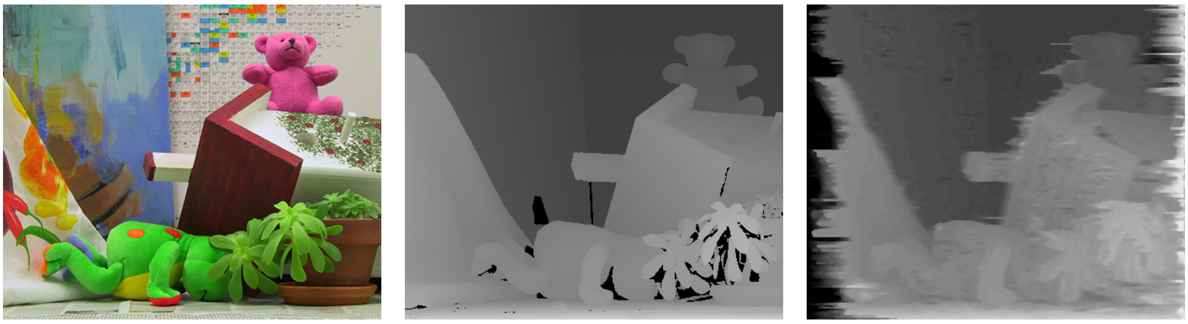
$$\begin{aligned} SAD(x,d) &= \mathbf{ABS}(p_y(x)_r - q_y(x+d)_r) \\ &+ \mathbf{ABS}(p_y(x)_g - q_y(x+d)_g) + \mathbf{ABS}(p_y(x)_b - q_y(x+d)_b) \end{aligned} \tag{1}$$

DP algorithm for disparity map computation works as follows: For each line y in both images I_l and I_r a cost matrix M_h is created with dimensions $W \times D_{max}$ where W and H is the width and height of the images. D_{max} is the *maximum disparity* dependant on the separation between cameras and minimum and maximum distances allowed. In a first step, this matrix M_h is filled with the cost corresponding to assign the disparity d for the pixels $p(x,y)$ and $q(x+d,y)$ when $x+d < W$. The cost matrix M_h is calculated with the equation (2) where the values of λ are assigned emipirically and represent the penalty of the change in the disparity value between neighboring pixels. In our test, we used a value of 5. The aggregation cost **SAD** is calculated between two pixels with the equation (1) where p_y and q_y are the values of Red-Green-Blue (**RGB**) colour of the pixels in the processed scan line y. A graphical representation of the cost matrix can be seen in the figure 1.

$$\begin{aligned} M_h(x,d) = SAD(x,d) + \mathbf{MIN}(&\lambda + M_h(x-1,d-1), \\ &M_h(x-1,d), \lambda + M_h(x,d+1)) \end{aligned} \tag{2}$$

The disparity for each pixel of the scanline, is calculated by a back tracking process, starting in the position $M(W, D_{max})$ and following the minimum cost path, assign-

© The Eurographics Association 2009.



(a) Right image (b) Ground Truth (c) Calculated Disparity

Figure 2: *Depthmap calculation for teddy image of Middlebury stereo vision data set. the calculated disparity was done using our algorithm and a median filter as a posprocessing step.*

ing the disparity value d corresponding to the last position of the path in each dimension of W.

4. Parallel DP

Our proposed parallelisation method of **DP** algorithm was based a parallelisation pattern of the matrix M_h. The pattern defines the parallel steps which can be executed simultaneously by a block of a several threads. The number of threads in each block changes dynamically in the execution of the algorithm. This number is managed by the architecture, along with the global, shared memory and synchronization functions.

The presented algorithm 1 uses a block of threads to calculate the cost matrix M_h. The size of the block of threads is $(1 \times BX)$, being BX a number between 1 and $\frac{D_{max}}{2}$. In each iteration the threads select an available cell to calculate the cost function. A cell is defined as available if it has had all of their required values already computed. The position of the cell is calculated in the variables c and d (see algorithm 1). For each cell (c,d), that is different for each thread, the minimum value of the neighbours $(c-1,d+1)$, $(c-1,d)$, $(c,d-1)$ is calculated in t. Then the current cell (c,d) is updated with the value of the aggregation cost function between the pixels $p(c,h)$ and $q(c+d,h)$ of the images I_l and I_r and the minimum value of the neighbours t.

The back tracking step is processed after filling the matrix M_h. The Calculation of the path for each scanline is carried out in parallel.

5. Results

We compared the results obtained with the proposed algorithm implemented in several **GPUs** against the same implementation on several **CPU** (see 3) The **GPU** version of the algorithm improves the running time of the algorithm with modern graphics cards. Results demostrates that the evolution of the running time of the algorithm between various types of **CPU** did not generate a substantial difference, but the evolution of the **GPU** implementation shows a significant improvement in the running time of the algorithm.

The images used in the benchmark are the middlebury images of tsukuba, cones, venus and teddy [SS03]. The time presented is the average running time of the same algorithm in 10 test cycles eliminating the extreme data, and the **DP** algorithm was ran 10 times every test cycle. Running the algorithm several times with the same data was done with the objective of discarding measurement error due to race conditions in the computer and the precision problems with the **PC** clock. The quality of the disparity estimation results of are the same of the dynamic programming implementation in CPU which were studied in [SS02].

6. Conclusion

We have presented a parallel implementation for a **DP** algorithm that takes benefits from the **GPGPU** computing model. Our implementation shows a high scalability running on **CUDA** maximising the performance of modern **GPU**s. These allows us to implement real-time stereo methods with increasing resolution and precision.

7. Acknowledgements

This work has been partially supported by the Spanish Administration agency CDTI, under project CENIT-VISION 2007-1007. CAD/CAM/CAE Laboratory - EAFIT University and the Colombian Council for Science and Technology – COLCIENCIAS –.

© The Eurographics Association 2009.

Input: I_r, I_l, W, H, D_{max}
Output: M_h
$h \leftarrow$ **blockIdx.x**
$j \leftarrow$ **threadIdx.y**
for $i \leftarrow 0$ **to** $\frac{W \cdot D_{max}}{\textbf{blockDim.y}} + D_{max} - \frac{\textbf{blockDim.y}}{2}$ **do**
    $c \leftarrow j + \left(\textbf{blockDim.y} \cdot \frac{i-2j}{D_{max}}\right)$
    $d \leftarrow (i-2j) \mod D_{max}$
    $t \leftarrow +\infty$
    if $(c-1 \geq 0) \wedge (c-1 < W)$ **then**
        if $(d+1 \geq 0) \wedge (d+1 < D_{max})$ **then**
            $t \leftarrow \lambda + \min(t, S[d+1])$
        end
        if $(d \geq 0) \wedge (d < D_{max})$ **then**
            $t \leftarrow \min(t, S[d])$
        end
    end
    if $(c \geq 0) \wedge (c < W)$ **then**
        if $(d-1 \geq 0) \wedge (d-1 < D_{max})$ **then**
            $t \leftarrow \lambda + \min(t, S[d-1])$
        end
    end
    if $t = +\infty$ **then**
        $t \leftarrow 0$
    end
    if $(c \geq 0) \wedge (c < W) \wedge (d \geq 0) \wedge (d < D_{max})$ **then**
        $S[d] \leftarrow$
        `AggregationFunc`$(c, h, d, I_l, I_r, W, H) + t$
        $M_h[c,d] = S[d]$
    end
    `syncthreads()`
end
return M_h

Algorithm 1: **CUDA** implementation of the **DP** algorithm, the number of running threads in the algorithm must be lower than the half of the D_{max} value.

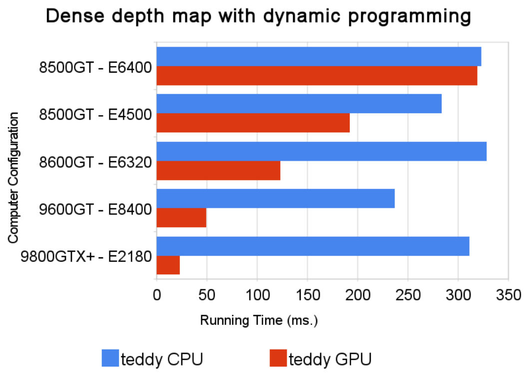


Figure 3: *Running time of the* ***DP*** *algorithm in different* ***CPU*** *and* ***GPU*** *configurations. The difference between the two 8500GT graphic card is because the different clock rates of the cards.*

References

[GM08] GIBSON J., MARQUES O.: Stereo depth with a unified architecture gpu. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on* (June 2008), 1–6.

[GP91] GALIL Z., PARK C. K.: *Parallel dynamic programming*. Tech. rep., Department of Computer Science Columbia University, 1991.

[GY05] GONG M., YANG Y.-H.: Near real-time reliable stereo matching using programmable graphics hardware. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 924–931.

[GYWG07] GONG M., YANG R., WANG L., GONG M.: A performance study on different cost aggregation approaches used in real-time stereo matching. *Int. J. Comput. Vision 75*, 2 (2007), 283–296.

[KSK06] KLAUS A., SORMANN M., KARNER K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on 3* (0-0 2006), 15–18.

[MV08] MANAVSKI S., VALLE G.: Cuda compatible gpu cards as efficient hardware accelerators for smith-waterman sequence alignment. *BMC Bioinformatics 9*, Suppl 2 (2008).

[OJL*07] OWENS, JOHN D., LUEBKE, DAVID, GOVINDARAJU, NAGA, HARRIS, MARK, KRUGER, JENS, LEFOHN, AARON E., PURCELL, TIMOTHY J.: A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum 26*, 1 (March 2007), 80–113.

[SS02] SCHARSTEIN D., SZELISKI R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision 47* (2002), 7–42.

[SS03] SCHARSTEIN D., SZELISKI R.: High-accuracy stereo depth maps using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1* (June 2003), 195–202.

[TMDSA08] TOMBARI F., MATTOCCIA S., DI STEFANO L., ADDIMANDA E.: Classification and evaluation of cost aggregation methods for stereo correspondence. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (June 2008), 1–8.

[WLG*06] WANG L., LIAO M., GONG M., YANG R., NISTER D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 798–805.

[WZ08] WANG Z.-F., ZHENG Z.-G.: A region based stereo matching algorithm using cooperative optimization. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (June 2008), 1–8.

[YWY*06] YANG Q., WANG L., YANG R., WANG S., LIAO M., NISTÉR D.: Real-time global stereo matching using hierarchical belief propagation. In *BMVC* (2006), British Machine Vision Association, pp. 989–998.