REAL-TIME ARBITRARY VIEW RENDERING ON GPU FROM STEREO VIDEO AND TIME-OF-FLIGHT CAMERA

Tuğrul K. Ateş, A. Aydın Alatan

Middle East Technical University
Department of Electrical and Electronics Engineering
tugrul.ates@uzay.tubitak.gov.tr, alatan@eee.metu.edu.tr

ABSTRACT

Generating in-between images from multiple views of a scene is a crucial task for both computer vision and computer graphics fields. Photorealistic rendering, 3DTV and robot navigation are some of many applications which benefit from arbitrary view synthesis, if it is achieved in real-time. GPUs excel in achieving high computation power by processing arrays of data in parallel, which make them ideal for real-time computer vision applications. This paper proposes an arbitrary view rendering algorithm by using two high resolution color cameras along with a single low resolution time-of-flight depth camera and utilizing GPUs to achieve real-time processing rates. The presented ideas are examined in an experimental framework and based on the experimental results, it could be concluded that it is possible to realize content production and display stages of a free-viewpoint system in real-time by using only low-cost commodity computing devices.

Index Terms— FTV, TOF, time-of-flight, GPU, graphics processing unit, image based rendering, video plus depth, bilateral filtering

1. INTRODUCTION

Free viewpoint television (FTV) [1] is any television setup which allows fine control of view angle and position through a remote control. Freeview paradigm adds an enhancement of user experience by adding interactivity and choice over the presentation of video content. FTV can be combined with either traditional 2D displays or 3D technologies (stereoscopy or auto-stereoscopy), if 3D content is supplied to the device in appropriate formats.

Free viewpoint, inherently, requires rendering of video content for requested arbitrary views. However, arbitrary view rendering can supplement display technologies even without explicit viewpoint controls. Stereoscopic viewing technologies can benefit from arbitrary view rendering, if head-tracking is employed to estimate the viewing angle and position of the viewer; thus, interactive experience of 3D, which is similar to autostereoscopic displays, is achieved. On the other hand, autostereoscopic devices require different views of a scene to achieve acceptable results. A feasible solution is dynamic generation of in-between views from multiview content captured with a less number of cameras [2].

In *computer vision* terms, McMillan and Bishop [3] regard arbitrary view rendering as the estimation of the values of the *plenoptic function* [4] for a certain viewing angle. The plenoptic function maps viewing parameters, namely viewing angle, scene location and time, to the amount of light intensity that can be seen. Their *plenoptic modeling* approach achieves generation of arbitrary views from projections of interpolation of plenoptic function observations. Full extraction of 3D information from multiview images is offered in works of Yaguchi [5] and Ito [6] where volumetric methods are employed to form model based representations of scenes from correspondences between images. Arbitrary views are generated by rendering obtained models in 3D.

In *computer graphics* field, Debevec et al. [7] tries to combine traditional model based approaches with image based algorithms for realistic graphics rendering and presents *projective texture mapping* algorithm for view dependent mapping of scene textures. Buehler et al. [8] builds a common framework for view dependent texture mapping algorithms through *unstructured lumigraph rendering* algorithm.

Research in free viewpoint television has led to *multiview video plus depth*, a content format presented by Smolic et al. [9]. Arbitrary view rendering is accomplished by warping color plus depth data from source views to target views.

Given two or more images from separate views of the same scene, *stereo correspondence* can lead to an estimate to the world coordinates of pixel locations on the images. However, state-of-the-art techniques still try to overcome problems occurring at boundaries and textureless regions [10]. A hardware solution to the drawbacks of stereo matching algorithms is *time-of-flight sensors,* which are relatively new and superior to some previous hardware solutions in various ways [11].

Time-of-flight cameras provide depth images at a rate equal to or greater than real-time speeds. However, their pixel resolution is quite low compared to contemporary color cameras. One approach to increase resolution of depth images obtained from range sensors is upscaling. Modern graphics processing units provide hardware support for *bilinear filtering* which may be satisfying, while *bilateral filtering* [12] is shown to perform better [13].

This work focuses on the scenario for 3D imaging where scenes are captured through a *stereo camera* pair and a *time-of-flight camera*, transmission is applied in compressed *multiview video plus depth* format and arbitrary view rendering is employed for 2D free viewpoint controlled display. High speed realization of all mentioned steps is enabled with *graphics processing units* and their programming mindset.

This paper is organized as follows. The proposed approach is examined in Section 2. Section 3 presents the experimental results and their interpretations. Finally, Section 4 gives conclusive remarks and future directions.

2. PROPOSED METHOD

2.1. Problem Formulation

Pinhole camera model for camera projection is an ideal approximation where camera aperture is assumed to be dimensionless, distortions caused by the actual physical lenses that perform real world projections are omitted and projection plane is

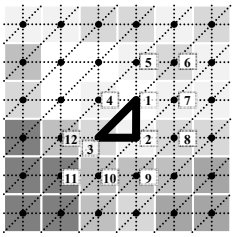

Figure 1 Triangulation of depth map into a surface mesh.

parallel to aperture plane. The pinhole camera model is defined by (1) and (2) in a homogeneous right-handed coordinate system.

$$p = \boldsymbol{I}\boldsymbol{E}P \tag{1}$$

$$\begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \alpha_x & 0 & -\mu_u & 0 \\ 0 & \alpha_y & -\mu_v & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} & \boldsymbol{R} & & \boldsymbol{T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

In (1) and (2), P denotes the 3D coordinates of scene points and p denotes the 2D coordinates, u and v, of image pixels. $\boldsymbol{E}$ represents the 4×4 matrix for extrinsic camera transformation which is composed of 3×3 camera orientation matrix $\boldsymbol{R}$ and 3×1 camera translation vector $\boldsymbol{T}$. $\boldsymbol{I}$ represents the 4×4 matrix for intrinsic camera transformation, in which $\boldsymbol{\alpha}$ combines focal length (f) and pixel scale in a single vector and $\boldsymbol{\mu}$ represents the principal point on image coordinates. This matrix performs *perspective transformation* from camera space to image space.

A viewpoint V, can be represented by $\boldsymbol{P}_V$, a 4×4 projection matrix of a 3D homogeneous projective transformation as given in (3). Points in front of the camera will have negative z-coordinates in scene space due to right handedness of the coordinate system. This z-coordinate z', is made positive and passed as a factor, resulting in a perspective division as given in (4). If depth values of pixels are known, $\boldsymbol{P}_V^{-1}$, provides a transformation from image space to scene space as given in (5).

$$\boldsymbol{P}_V = \boldsymbol{I}_V \boldsymbol{E}_V \tag{3}$$

$$\begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \\ z' \end{bmatrix} = \boldsymbol{P}_V \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V^{-1} \begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V^{-1} \begin{bmatrix} uz' \\ vz' \\ 1 \\ z' \end{bmatrix} \tag{5}$$

Beginning with calibrated stereo viewpoints, L and R, their corresponding color images $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$, calibrated range sensor viewpoint ToF and its respective depth map $\boldsymbol{D}_{ToF}$, initial aim is to estimate the artificial depth maps, $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ in order to have a complete stereo video plus depth representation of the scene. In the second stage, two views, L and R, along with their color and depth maps are transmitted and intermediate views are generated upon request. Given an intermediate viewpoint V, depth estimation and view rendering stages are summarized in (6) and (7).

$$\boldsymbol{D}_L, \boldsymbol{D}_R = Depth\ Estimation(L, \boldsymbol{C}_L, R, \boldsymbol{C}_R, ToF, \boldsymbol{D}_{ToF}) \tag{6}$$

$$\boldsymbol{C}_V = View\ Rendering(V, L, \boldsymbol{C}_L, \boldsymbol{D}_L, R, \boldsymbol{C}_R, \boldsymbol{D}_R) \tag{7}$$

2.2. Depth Estimation

Depth map captured with a time-of-flight camera is represented with $\boldsymbol{D}_{ToF}$ and represents the surface of scene content visible from viewpoint ToF. Since depth maps are inherently 3D descriptions, they can be projected into other views. Projection of any point on this surface to viewpoint V is carried out with (8) and (9). Depth values of target viewpoint are given by (10). However, surface projection is not strictly a one-to-one mapping. Some pixels at the target depth map cannot be deduced and some others can have multiple projections due to occlusions. Multiple projection values can be singled out with depth testing and missing depth values can be interpolated from neighboring projections. Both of these solutions are provided by graphics processing units.

$$\begin{bmatrix} u_V \\ v_V \\ 1/z_V \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V \boldsymbol{P}_{ToF}^{-1} \begin{bmatrix} u_{ToF} \\ v_{ToF} \\ 1/z_{ToF} \\ 1 \end{bmatrix} \tag{8}$$

$$z_{ToF} = \boldsymbol{D}_{ToF}(u_{ToF}, v_{ToF}) \tag{9}$$

$$\boldsymbol{D}_V^{warp}(u_V, v_V) = z_V \tag{10}$$

Depth map projection can be implemented on the GPU by modeling the source depth map as a 3D surface in the form of a triangle mesh. Pixels of the depth map provided by the time-of-flight camera are converted to 3D vertices and triangles are formed in-between as shown in Figure 1 and (11). These triangles are first transformed to scene space and then projected onto target depth map. This operation is called *warping* and covers the entire target image leaving no gaps.

After mesh warping, discontinuities are covered by larger triangles which usually span from background to foreground resulting in large depth patches. Areas of the target depth map that are not visible to ToF are interpolated between background and foreground. These *rubber sheet* artifacts remain at discontinuities after warping and they can be removed with mesh segmentation techniques [14]. Problematic triangles are detected inside a geometry shader and they are suppressed to background to limit their favorability over foreground patches from other views. Depth selection process for triangles is given in (12) and (13). Depth of the furthest vertex among neighboring triangles is used as the suppression depth. Comparison area a' is an empirical value to detect occlusion patches. If depth testing is enabled, using depth and color information from these patches becomes only a last resort; thus, the occlusions are handled.

$$T = \left\{ \begin{bmatrix} u_1 \\ v_1 \\ 1/z_1 \\ 1 \end{bmatrix}, \begin{bmatrix} u_2 \\ v_2 \\ 1/z_2 \\ 1 \end{bmatrix}, \begin{bmatrix} u_3 \\ v_3 \\ 1/z_3 \\ 1 \end{bmatrix} \right\} \tag{11}$$

$$T' = \begin{cases} T & ;\ \text{area}(T) < a' \\ \left\{ \begin{bmatrix} u_1 \\ v_1 \\ 1/z_{max} \\ 1 \end{bmatrix}, \begin{bmatrix} u_2 \\ v_2 \\ 1/z_{max} \\ 1 \end{bmatrix}, \begin{bmatrix} u_3 \\ v_3 \\ 1/z_{max} \\ 1 \end{bmatrix} \right\} & ;\ \text{area}(T) \geq a' \end{cases} \tag{12}$$

$$z_{max} = \max(z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}, z_{11}, z_{12}) \tag{13}$$

Bilateral filtering [12] is a low-pass image filtering algorithm which aims preserving edges while smoothing the image. An adaptive pixel filter kernel, which takes high values for similar neighbors, is applied to all pixels and therefore smoothing is

Figure 2 Several (a) left, (b) time-of-flight and (c) right frame groups as input.

Figure 3 Generated intermediate views for input obtained from data acquisition setup.

prevented at edges. Similarity is defined both in color and spatial domain. The idea of bilateral filtering can be extended to upscaling depth maps with respect to related color maps to align depth boundaries with color boundaries [13]. Depth values obtained from adaptive filter is found by (14) using pixel similarity measure given in (15). This similarity incorporates decays of Euclidean RGB color distance s and pixel location distances. Final depth value is a weighted average over the square neighborhood $\mathbf{N}$ of (u, v).

$$\boldsymbol{D_V}(u,v) = \frac{\sum_{(i,j)\in\mathbf{N}(u,v)} \boldsymbol{D}_V^{warp}(i,j)\, w(\boldsymbol{C}_V, u, v, i, j)}{\sum_{(i,j)\in\mathbf{N}(u,v)} w(\boldsymbol{C}_V, u, v, i, j)} \tag{14}$$

$$w(\boldsymbol{C}, u, v, i, j) = e^{-\frac{s(\boldsymbol{C}(u,v),\boldsymbol{C}(i,j))}{\gamma_d}} e^{-\frac{\sqrt{(i-u)^2+(j-v)^2}}{\gamma_s}} \tag{15}$$

Estimations to $\boldsymbol{D}_L(u,v)$ and $\boldsymbol{D}_R(u,v)$ along with $\boldsymbol{C}_L(u,v)$ and $\boldsymbol{C}_R(u,v)$ make up the stereo video plus depth data needed for arbitrary view rendering at the display side of the broadcast system.

2.3. Arbitrary View Rendering

Given any viewpoint V and stereo video plus depth data, arbitrary view rendering for V is governed by the input-output relation given in (7). $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ store sufficient information to relate pixels in $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ to 3D scene space coordinates. Much like the depth warping process to transform time-of-flight output to other viewpoints; $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ can be back-projected to scene space and then projected onto target viewpoint V. Pixels on the image space of V are related to the pixels of the image space of L as given in (16). A vertex shader transforms incoming vertices according to $\boldsymbol{P}_V\boldsymbol{P}_L^{-1}$ and a geometry shader performs triangle transformation according to (12). A similar relation between V and R can easily be deduced.

$$\begin{bmatrix} u_V \\ v_V \\ 1/z_V \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V\boldsymbol{P}_L^{-1} \begin{bmatrix} u_L \\ v_L \\ 1/\boldsymbol{D}_L(u_L, v_L) \\ 1 \end{bmatrix} \tag{16}$$

Rendering of generated fragments through a fragment kernel with depth testing result in two alternative arbitrary view images, $\boldsymbol{C}_V^L$ and $\boldsymbol{C}_V^R$. Fusion of two color maps is performed through a selective blending scheme given in (17).

$$\boldsymbol{C}_V^{com}(u,v) = \begin{cases} \boldsymbol{C}_V^L(u,v) & ; \quad \Delta\boldsymbol{D} \le -\boldsymbol{D}_t \\ \dfrac{\alpha_L \boldsymbol{C}_V^L(u,v) + \alpha_R \boldsymbol{C}_V^R(u,v)}{\alpha_L + \alpha_R} & ; \quad |\Delta\boldsymbol{D}| < \boldsymbol{D}_t \\ \boldsymbol{C}_V^L(u,v) & ; \quad \Delta\boldsymbol{D} \ge \boldsymbol{D}_t \end{cases} \tag{17}$$

$$\Delta\boldsymbol{D} = \boldsymbol{D}_V^L(u,v) - \boldsymbol{D}_V^R(u,v) \tag{18}$$

$$\alpha_L = 1 - \frac{\mathrm{dist}(L,V)}{\mathrm{dist}(L,V) + \mathrm{dist}(R,V)} \tag{19}$$

$$\alpha_R = 1 - \frac{\mathrm{dist}(R,V)}{\mathrm{dist}(L,V) + \mathrm{dist}(R,V)} \tag{20}$$

$$\mathrm{dist}(V_A, V_B) = \|\boldsymbol{T}_A - \boldsymbol{T}_B\| \tag{21}$$

Color images are blended primarily according to their depth counterparts. If a pixel from a view is nearer than the respective pixel from the other view at least by $\boldsymbol{D}_t$, it is directly copied to the target color map. If corresponding depth values are similar, a weighted color is produced with weights calculated from the Euclidean distances between viewpoints inside the scene space, with $\boldsymbol{T}_V$ being the translation vector of viewpoint V defined by the pinhole camera model. This interpolation ensures smooth transition as V moves from one viewpoint to another [9].

Single pixel artifacts remain on $\boldsymbol{C}_V^{com}$ due to imperfect alignment of color and depth images during previous stages of the algorithm. These artifacts are corrected with 2D selective median filtering which preserves pixels which are similar to their neighbors and replaces others with the local median. Selective median operation is given in (22) where m is median color inside $\mathbf{N}(u,v)$ and s' is the selection threshold.

$$\boldsymbol{C}_V^{med}(u,v) = \begin{cases} \boldsymbol{C}_V^{com}(u,v) & ; \quad s\left(\boldsymbol{C}_V^{warp}(u,v), m\right) \le s' \\ m & ; \quad s\left(\boldsymbol{C}_V^{warp}(u,v), m\right) > s' \end{cases} \tag{22}$$

Mesh warping, selective color map fusion and selective median filtering add up to an unintentional sharpness over the generated arbitrary view. A final touch of selective low pass filtering at edges provides a more natural rendered image. This low pass filter is explained by (23). In this equation, sob is the Sobel filtering response of the image, e' is the edge threshold and box is a linear smoothing filter.

$$\boldsymbol{C}_V(u,v) = \begin{cases} \boldsymbol{C}_V^{med}(u,v) & ; \quad \mathrm{sob}\left(\boldsymbol{C}_V^{med}\right)(u,v) \le e' \\ \mathrm{box}\left(\boldsymbol{C}_V^{med}\right)(u,v) & ; \quad \mathrm{sob}\left(\boldsymbol{C}_V^{med}\right)(u,v) > e' \end{cases} \tag{23}$$

3. EXPERIMENTS

Proposed algorithm is tested with color and time-of-flight cameras and a real-time application is developed. Data acquisition from the cameras and display of arbitrary view generation results take stage together in a single architecture with NVidia GeForce 8600M GT GPU. Two LightWise LW-3-S-1394 FireWire cameras are used for obtaining color images. SwissRanger SR-3000 time-of-flight camera is used for range measurements. Systematic error in raw depth measurements from time-of-flight camera are corrected with software using SDK of SR-3000. This correction includes both *denoising* and *median filtering*. Examples to captured images from this setup are shown in Figure 2 and several intermediate views generated with the algorithm are given in Figure 3. The system runs at 30fps with 512x480 stereo color input and 172x144 time-of-flight depth input.

Quality of the rendered color maps can be estimated with the *peak signal-to-noise ratio* (PSNR) with respect to a reference color map. The algorithm presented in this study is tested with *Breakdancers* and *Ballet Studio* datasets from Zitnick et al. [15]. The arbitrary view rendering algorithm is tested in different configurations. Camera 4 is selected as the reference point and its color map is estimated through neighboring color maps.

Low resolution output of SR-3000 and its systematic errors are simulated by downscaling the depth map of Camera 4 and introduction of white Gaussian depth noise with a deviation value that attacks the robustness of the algorithm. Obtained quality results are given in Table 1 with respect to the standard deviation of Gaussian noise added to the initial depth map. PSNR values obtained from high resolution depth maps without time-of-flight camera simulation is also given.

Quality comparison tests for final rendered images show that the algorithm performance is very susceptible to the stereo configuration used. Depth warping onto stereo viewpoints yields the best results when these camera position are nearer to the depth sensor. Furthermore, the algorithm is robust against depth distortions up to a certain noise level where performance decays beyond. Erosion occurs for small object regions at both depth estimation and view rendering stages. This weakness is most exposed within numerical results of the ballet studio dataset.

4. CONCLUSIONS

This paper proposes an arbitrary view rendering system that uses stereo color cameras and a time-of-flight camera at content acquisition stage and renders intermediate rendered views at display stage. Video plus depth broadcast format is used in between these stages and all computation is burdened to graphics processing units to achieve real-time operation rates.

After per pixel estimation of depth maps, global optimization methods can be utilized. Error minimization oriented approaches can be time consuming, but real-time rates can be achieved with compute unified programming [16]. Access to high performance parallelization on GPUs for general purpose tasks is helpful when computing capabilities of CPUs are left alone for other tasks.

Arbitrary view rendering systems commonly employ post-processing steps that correct artifacts occurring after color warping. Smolic et al. [9] provide a set of correction techniques which are not trivial to parallelize with stream processing paradigms. Compute unified architectures, again, can be useful for extra view correction measures after rendering passes are completed.

Table 1 PSNR values for different stereo setups and noise levels.

Dataset	Stereo Pair	High Res.	$\sigma_E = \frac{0.01}{255}$	$\sigma_E = \frac{0.1}{255}$	$\sigma_E = \frac{1}{255}$	$\sigma_E = \frac{10}{255}$
Breakdancers	3 and 5	30.36	29.10	29.10	29.13	26.47
Breakdancers	2 and 6	29.58	28.17	28.17	28.11	24.08
Breakdancers	1 and 7	27.29	26.14	26.14	26.07	22.42
Ballet studio	3 and 5	27.19	25.37	25.36	25.32	22.19
Ballet studio	2 and 6	25.70	24.24	24.24	24.10	21.19
Ballet studio	1 and 7	21.51	20.68	20.68	20.69	18.80

5. REFERENCES

[1] M. Tanimoto, "Free viewpoint television—ftv," in *PCS, Session 5*, 2004.

[2] D. Jung and R. Koch, "Efficient depth-compensated interpolation for full parallax displays," in *3DPVT*, 2010.

[3] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system," in *SIGGRAPH*, 1995, pp. 29-46.

[4] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, Michael Landy and J. Anthony Movshon, Eds. Cambridge, Massachusetts: The MIT Press, 1991, ch. 1, pp. 3-20.

[5] S. Yaguchi and H. Saito, "Arbitrary viewpoint video synthesis from multiple uncalibrated cameras," *IEEE SMC, Part B: Cybernetics*, vol. 34, no. 1, pp. 430-439, February 2004.

[6] Y. Ito and H. Saito, "Free-viewpoint image synthesis from multiple-view images taken with uncalibrated moving cameras," in *IEEE ICIP*, 2005, pp. 29-32.

[7] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *SIGGRAPH*, 1996, pp. 11-20.

[8] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *SIGGRAPH*, 2001, pp. 425-432.

[9] A. Smolic et al., "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems," in *IEEE ICIP*, 2008, pp. 2448-2451.

[10] J. Zhu, L. Wang, R. Yang, and J. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps," in *IEEE CVPR*, 2008.

[11] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-flight cameras in computer graphics," *Computer Graphics Forum*, vol. 29, no. 1, pp. 141-159, February 2010.

[12] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, 1998, pp. 839-846.

[13] Q. Yang, R. Yang, J. Davis, and D. Nistér, "Spatial-depth super resolution for range images," in *IEEE CVPR*, 2007.

[14] R. Pajarola, M. Sainz, and Y. Meng, "Dmesh: fast depth-image meshing and warping," *IJIG*, vol. 4, no. 4, pp. 653-681, October 2004.

[15] C. L. Zitnick, Sing B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *SIGGRAPH*, 2004, pp. 600-608.

[16] J. Congote, J. Barandiaran, I. Barandiaran, and O. Ruiz, "Realtime dense stereo matching with dynamic programming in CUDA," in *CEIG*, 2009, pp. 231-234.