ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)
On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

2D-To-3D Video Conversion System

E.Anitha, P.S.Ramesh,

Department of CSE, Arunai Engineering College, Tiruvannamalai, Tamilnadu, India.

Department of CSE, Arunai Engineering College, Tiruvannamalai, Tamilnadu, India.

ABSTRACT-This paper is demonstrating about a 2D-TO-3D video conversion system capable of real-time 1920*1080p conversion. The proposed system generates 3D depth information by fusing cues from edge feature-based global scene depth gradient and texture-based local depth refinement, generated 3d images have comfortable and vivid quality, and algorithm has very low computational complexity. Software is based on a system with a multi-core CPU and a GPU. To optimize performance, we use several techniques including unified streaming dataflow, multi-thread schedule synchronization, and GPU acceleration for depth image-based rendering(DIBR).With proposed method, real-time 1920*1080p 2D-to-3D video conversion running at 30fps is then achieved.

KEY WORDS: Depth map generation, 2D-to-3D conversion, real-time implementation, 3D video

I. INTRODUCTION

3D video is getting immense public attention recently because of vivid stereo visual experience over conventional 2D video. There are several methods to produce 3D content, such as active depth sensing, stereo camera recording, and 3D graphics rendering. Active depth sensing uses active sensors such as

Structured lithe, time-of-flight sensor[1]to estimate actual depth. Stereo cameras record disparity between

Views and produce depth with stereo matching. However, these methods need specific devices and are only suitable for new Content production. Most of the existing videos do not include any pre-recorded depth information. To Convert these 2D video to 3D ones, time-consuming manual editing of the depth information is required and becomes a huge Barrie. The lack of 3D content has become the major problem for 3D display industry. An efficient automatic 2D-to-3D conversion system automatically generates depth information from single view video and converts it to 3D video by using the produced depth maps [2] as shown in Fig.1.

2D-to-3D video conversion is typically based on the characteristics of human depth perception. The human brain integrated various Heuristic depth cues to generate depth perception; including binocular cue from two eyes and various monocular cues from single eye.2D-TO-3D conversion recovers depth information from various depth cues in single view video. Various techniques have been proposed in[8]-[5].However, generating depth maps form singe view video is an ill-posed problem. Physical depth is hard to recover even with high complexity algorithms.

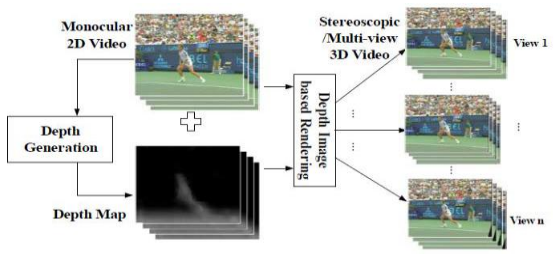

Monocular 2D Video
Depth Generation
Depth Map
Depth Image based Rendering
Stereoscopic /Multi-view 3D Video
View 1
View n

Fig.1.Typical 2D-to-3D video conversion flow.

For 3D consumer electronics devices, real-time on- the-fly conversion is required. Besides, the implementation cost must be reasonable. In our previous proposed system, we use different cues for depth generation[6],[10].Our latest work proposes an ultra low cost 2D-to-3D conversion system[7].We use human visual perception to generate visually comfortable depth maps rather than physically correct depth maps. The algorithm fuses global and local depth cues from video analysis and generates depth information with little side effects. In this work, the system is implemented on a laptop computer with a multi-core CPU and a GPU. Optimization techniques such as unified streaming dataflow, multi-thread schedule synchronization, and GPU acceleration are applied to this system. The proposed system is capable of real-time 1920*1080p conversion and suitable for 3D consumer electronic devices.

The rest of this paper is organized as follows. Section II describes the algorithm and system optimization techniques used in the proposed system. Section III summarizes the experimental results .Concluding remarks are finally made in Section IV.

II. PROPOSED SYSTEM

The algorithm and optimization
Techniques of the proposed system are described in this section. Algorithm for the 2D-to-3D conversion is based on human visual perception. Depth maps are generated by fusing global depth gradient and local depth refinement. Multi-view images are rendered by depth image-based rendering (DIBR) from the depth imaps and original 2D images. The output images are presented on a 3D display. For real-time consideration, we apply several optimization techniques on the multi-core CPU and the GPU. First, to eliminate the requirement of frame-level format conversion, we propose a format-friendly data access scheme with unified streaming dataflow. Next, for multi-threading optimization on the muti-core CPU, we use schedule synchronization to maximize data reuse. Finally, we further accelerate DIBR on the GPU if it is available in the system. Shared memory buffering and parallel dynamic programming are used to take care of bandwidth and visibility problem .With proposed techniques, bandwidth is reduced and parallelism is maximized for real-time performance.

A. Algorithm

We generated depth maps by fusing two low complexity cues based on human visual perception rather than physical depth information. First of all, in human’s living environment, objects in the lower visual field are mostly supposed to be closer to the observer. Near-to-far global scene depth is the most important cue in the real world .Secondly; lighting and colour gradient yield some depth perception and are used as the second cue. Some great painters as Paul Cezanne use “warm” pigments (read, orange and yellow) to indicate far objects. The above two depth cues are major cused for human depth perception and are fused together to generate perceptual depth fast and effectively in our system. The system block diagram is shown in Fig.3 in the following section.
Firstly, edge feature-based global depth gradient generates an initial scene depth map with the texture cue. In the following subsections, we explain the each part in detail.

1) Global Depth Map Generation

As human visual perception tends to interpret that the lower visual field is closer, we apply near-to-far global scene depth gradient as the major cue. To decide the gradient, we use the fact that the depth gradient of the ground is often larger than that of the sky. Besides the ground area is more complex than the sky. We use the horizontal complexity of the frame to distinguish between the ground and the sky. The horizontal complexity is obtained from the cumulative horizontal edge histogram. Near to far global depth ranging from 0to255 for the 8bit depth map is assigned according to the cumulative histogram .When horizontal complexity is higher, more depth gradient is assigned. This method yields a sharper depth change between the smooth sky and the objects, and between the defocus background and the in-focus foreground. This method has better protrusion effect than linear or fixed depth gradient.

2) Local Depth Refinement

The concept of local depth refinement is based on two characteristics .Firstly the edge of the input image has high potential to be the edge in the depth map. Secondly, people feel red colour is nearer, and blue colour is farther in visual perception. Besides, objects wit higher luminance feel like nearer than those with lower luminance .Therefore, colour can be used as a depth cue to enhance the depth perception on both edge and colour domains .Based on the concept, we use a nove combination of Y,Cr and Cb colour channels to generate the fine-grained depth map as discussed below.

Although not all the conditions satisfy the psychological hypothesis, the depth with high correlation to human perception also generates visually comfortable result. The preserved lighting gradient on the object surface also provides human depth perception in this case. In practice, Y and Crare mapping to linear increasing gains from1-Yth to 1+Yth, and Cb is mapping to a linear decreasing gain from 1+Cbth to 1-Cbth for depth fusion. The following equation is used to refine the depth map.

Depthfused(X)=G(x)*fy(Y(x))*fcr(Cr(x))*fcb(Cb(x))

Where x stamds for position,G(x),fy(Y(x)),fcr(Cr(x)),fcb(Cb(x)) are the function of global depth gradient luma Y channel gain, chroma Cr channel gain ,and chroma Cb channel gain ,respectively.

3) Depth image-based rendering (DIBR)

For 3D visualization, the input image is converted to multi-view images with the generated depth map. The disparity among the rendered images is observed by human eyes and then produces 3D effect. We derive the disparity from the depth, shown in Fig.2.Depth image-based rendering (DIBR) algorithm is used for the generation of multi-view images. We use pixel-based DIBR in [14],[15]

.

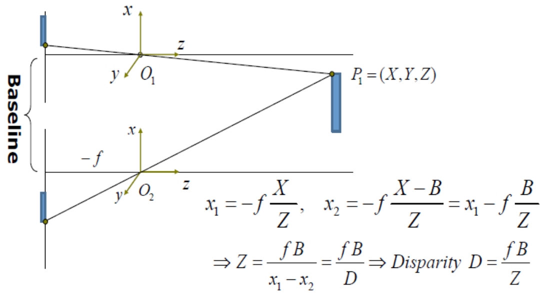

x
z
y
O_1
$P_1=(X,Y,Z)$
Baseline
x
$-f$
y
O_2
z
$x_1=-f\frac{X}{Z},\quad x_2=-f\frac{X-B}{Z}=x_1-f\frac{B}{Z}$
$\Rightarrow Z=\frac{fB}{x_1-x_2}=\frac{fB}{D}\Rightarrow Disparity\ D=\frac{fB}{Z}$

Fig2.Depth and disparity relations for DIBR algorithm

B. Performance Optimization

For real-time demonstration, the 2D-to-3D conversion system is integrated with video decoders, a 3D video player, and other related components in the operating system. Due to the high resolution of video input and output, memory bandwidth requirement is quite high and becomes a performance bound.To reduce excess memory access, several techniques are proposed below. The detail of each optimization technique is discussed in the following subsections.

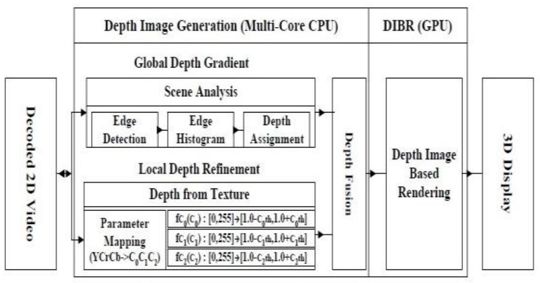

Depth Image Generation (Multi-Core CPU)
DIBR (GPU)
Global Depth Gradient
Scene Analysis
Edge Detection
Edge Histogram
Depth Assignment
Local Depth Refinement
Depth from Texture
Parameter Mapping (YCrCb->$C_0C_1C_2$)
$fc_0(C_0)$: [0,255]→[1.0-C_0th,1.0+C_0th]
$fc_1(C_1)$: [0,255]→[1.0-C_1th,1.0+C_1th]
$fc_2(C_2)$: [0,255]→[1.0-C_2th,1.0+C_2th]
Decoded 2D Video
Depth Fusion
Depth Image Based Rendering
3D Display

Fig.3.System architecture for proposed 2D-to-3D video conversion system

1).System Architecture

The system consists of two major parts:
Depth image generation and DIBR, as shown in fig.3.Bandwidth and computation optimization are the major concern.

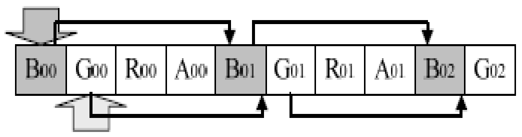
For depth image generation, the execution path is complex. The whole image frame is read several times by different computation modules and the flow is subject to change .As a result, it is more desired to execute on a multi-core CPU. Different computation can be executed concurrently with Multiple Instruction stream, Multiple Data stream (MIMD) architecture of a multi-core CPU.

For DIBR, the dataflow is rather fixed. Besides, the processing loading is high for pixel-based DIBR algorithm. It is more desired to put DIBR on a highly parallelized GPU to reduce the loading of the CPU. In addition, the bandwidth of transferring the rendered frame can be saved if DIBR directly renders the output frame on A GPU texture.

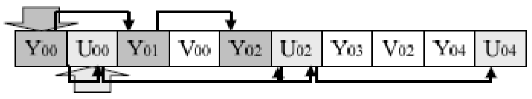
For these reasons, we put depth image generation on the CPU and DIBR on the GPU to optimized bandwidth and computation.

2) Unified Streaming Dataflow for Multi-Format Processing

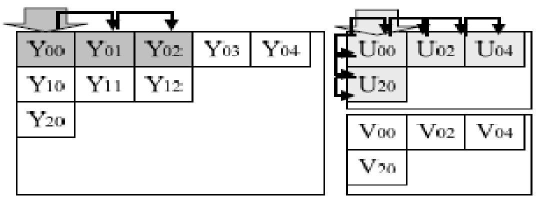
Video decoders in the system may output video in various formats, color space, and chroma sub-sampling modes as shown in Fig.4.Formats for input images, as RGB32, YUYV, and Y V12,may be in various kinds of colour space, packed or planar, horizontally or vertically sub-sampled. The final chosen format is the negotiation result of decoders, renderers, and other components in the system .If the format is not compatible, frame-level colour conversion by default is done beforehand. The conversion consumes excess bandwidth, and so affects performance. To avoid this, we firstly do the computation on the input colorspace. Filter parameters are projected to input colorspace instead to avoid frame-level conversion. Next, we propose unified streaming dataflow for the system pipeline. The depth image generation and DIBR are implemented with this dataflow to support various pixel packing order.


B_{00} G_{00} R_{00} A_{00} B_{01} G_{01} R_{01} A_{01} B_{02} G_{02}

(a)RGB32 case:stride=4.


Y_{00} U_{00} Y_{01} V_{00} Y_{02} U_{02} Y_{03} V_{02} Y_{04} U_{04}

(b)YUYV case:luma stride=2,chroma stride=4/2.


Y_{00} Y_{01} Y_{02} Y_{03} Y_{04}
Y_{10} Y_{11} Y_{12}
Y_{20}
U_{00} U_{02} U_{04}
U_{20}
V_{00} V_{02} V_{04}
V_{20}

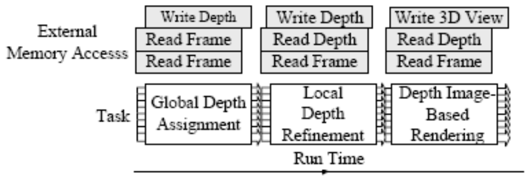
(c)YV12(planar)case:luma stride=1,chroma stride=1/2,skip=pitch/2.

Fig.4.Stream descriptors for various pixel packing orders.

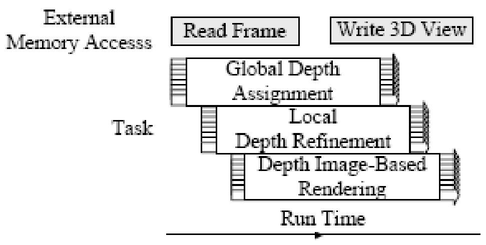
The unified streaming dataflow uses stream descriptors to describe the pixel packing order for various formats. The descriptors are based on shape descriptors in[16]with some modification .A stream
Descriptor consists of the pointer indicating the first element of the colour component and an description for pixel iteration order. The description consists of three parameters: stride, span and skip. Stride describes the spacing between elements.
Span describes how many elements are to iterate before applying a skip offset. Span is always equal to frame width(w)if no scaling is required. Skip in practice in the pitch, which is the actual offset between rows. In addition, stride and skip may be represented in fractional numbers in terms of K/R where K and R are the numerator and the denominator, respectively. For stride in fractional number Ks/Rs ,the value is repeatedly processed Rs times before applying an offset Ks.For skip in fractional number Kp/Rp, the whole row is repeatedly processed Rp times before skipping a pitch Kp. For example,RGB32 in fig.4(a) can be processed with stride=4.YUYV in Fig.4(b) can be processed with fractional chroma stride 4/2,which means processing 2 repeating pixels before applying a4-pixel offset. For YV12 case in Fig.4(c),we may use stride=1/2 and skip=pitch/2 on chroma components. As shown above, various formats can be processed with the descriptors. For practical implementation, pointers of the three color components are overloaded with stream descriptor-based pointers. Depth image generation and DIBR access the input and output color frames with proposed unified dataflow to avoid redundant frame-level color conversion.

3) Multi-Thread Schedule Synchronization for Data Locality Optimization

Since the system accesses the input and output frames multiple times, the performance will be degraded without proper scheduling. As shown in Fig.5(a),the major system pipeline consists of global depth assignment reads the original input frame twice and read/write temporary buffer for cumulative edge histogram ,and writes a generated global depth map to memory.Next,local depth refinement reads original input frame once,reads the global depth map,and writes the refined depth map to memory.Finally,DIBR reads the refined depth map and the original input frame to produce multi-view input frames.



(a)Before optimization



(b)After optimization.

Fig.5.Memory access optimization by multi-thread schedule synchronization.

Since the input frame is likely to exceed maximum CPU cache size for 1080p video, the cache cannot effectively save the bandwidth and the data is not reused. The same situation happens on the generated depth map. Therefore, the system suffers from repetitively load/store action. To eliminate the large buffer required for the system, we change the processing size from the whole frame to small regions.Synchronization mechanism is added to do this.A frame-level task for each stage is divided evenly into much smaller jobs as line fragments.Each of the fragment ranges a number of pixels in a horizontal row of the input frame.Synchronization points are placed at the start of the jobs.The jobs are put in job pools.At each synchronization point,each thread checks maximum displacement of synchronization points among various kinds of task. If the displacement is larger than a threshold value, the job is postponed and the corresponding worker thread steals job from other tasks. If no job is present, the thread sleeps temporarily. To prevent additional read required for histogram accumulation, the previous frame edge count is taken instead for normalization. This method has a practically unnoticeable effect on the visual quality. The refined schedule is shown as fig.5 (b) .The transaction size for each action is reduced to a line fragment. As a result, the CPU cache is effective in buffering data. Much external memory access is reduced. With the proposed multi-threading scheduling synchronization scheme, data locality is improved and 60% bandwidth is reduced.

4) GPU Acceleration for DIBR

If a GPU exists in the system, DIBR is accelerated on it to reduce the loading of the CPU. Previous work [11]-[12] proposes system that delivers real-time performance by using GPU or hardware. The DIBR is based on texture shift or supported by texture unit on GPU.The method is not very suitable for our algorithm. Every pixel has its own depth in our algorithm. We desire preserving the per-pixel depth and the object gradient detail for viewing experience. For this reason, we propose per-pixel parallel DIBR algorithm on GPU in the following.

We use the following methods for accelerating DIBR on GPU.Firstly, input and output frames need to be moved between main memory and graphics memory if standalone graphics memory is used. Data movement in between is overlapped with computation for better performance by using stream. If the video player supports 3D texture output, the output frames can be directly rendered and even more bandwidth is saved. Secondly, we use massive parallelism on GPU to accelerated DIBR. The scheme is shown in Fig.6.The output frames to be rendered are divided into multiple blocks for parallel rendering. Each block contains one horizontal line. Multiple threads in the same block render the line concurrently. Each thread renders 1 pixel at one time. If line width is larger than thread number, the operation repeats until all pixels in the line are rendered. In practice, the GPU usually contains multiple stream processors and so multiple blocks are rendered concurrently. Besides, each processor may contain a few blocks for latency hiding.

For this scheme in Fig.6, there are two major problems that affect performance: the low effectiveness of off-chip memory transaction and visibility problem. These two problems are the serious limiting factor for acceleration of DIBR on GPU and also mentioned in [20].Methods for solving the problems are discussed in the following.

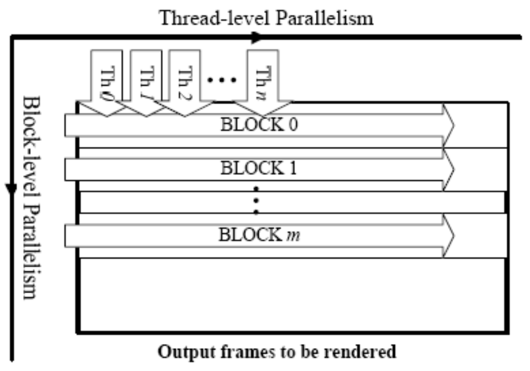


Fig.6.Thread and block level parallelism for DIBR on GPU.

Off-chip memory access has low bandwidth capacity and long latency, and so it affects performance without proper design. Since DIBR is per-pixel processing, direct implementation causes
Lots of 1-byte transaction. Many short transactions are not efficient for accessing off-chip memory. To avoid this, we use on-chip shared memory as an I/O buffer. Input color images and depth images are first loaded into shared memory. Output multi-view images are also buffered in the shared memory before flushing out to off-chip memory. All view are rendered at the same time for the reduction of common input data. Since the shared memory on chip is quite limited, we only save part of the line that is required for current computation. Size of the memory depends on the total number of threads in a block. More blocks can be loaded in a single stream processor (SP) with limited use of shared memory. Latency hiding is better in this way.

Another major problem is visibility problem for rendering pixels. For view rendering, objects in the same line of sight overlap each other. Only the nearest object should be rendered. This problem also exists in computer graphics. Since the DIBR is pixel-based rendering, reverse painter's algorithm [13] solves this problem efficiently. To find the nearest pixel without checking all the possible pixels, we use the most left pixel in the origin view that corresponds to the given line of sight for the left view, and the most right one for the right view. The problem then becomes a min/max problem. Here we propose a parallel dynamic programming technique to solve this problem efficiently. For dynamic programming, the overlapping structure of this problem is derived below.

$$F^k(x) \equiv M\{p(x-2^k+1)\ldots p(x)\}$$
$$= M\{M\{p(x-2^k+1)\ldots p(x-2^{k-1})\}, M\{p(x-2^{k-1}+1)\ldots p(x)\}\}$$
$$\Rightarrow F^k(x) = M\{F^{k-1}(x-2^{k-1}), F^{k-1}(x)\}$$

Where k is the level is the position(x) is the pixel value at x,F^k(x)is the desired min/max result at level k ranging from x-2^k+1 to x, and M is min/max function according to the viewers position.

Since the computation of F^k(x) for level k only depends on previous level result, we can perform calculation of the same level in parallel. The computing scheme is shown in Fig.7. The data is first loaded in shared memory. Fine-grained parallelism is used. Each step advances results in the shared memory with one level from previous level results. In each step, each thread performs one min/max operation and the result is written back to shared memory. The total number of required step is the binary logarithm value of the disparity range. As a result, we may find out the required min/max value efficiently by repeating a few steps of the above process. Since the disparity range can be derived for the given system in advance, loop unrolling is also used to eliminate iteration overhead. With this method, the visibility problem is solved efficiently.

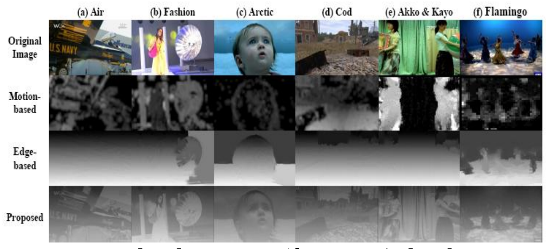


Fig.8.original 2d images (first row),depth maps result of motion-based(second row),edge-based(third row) and proposed algorithm(bottom).

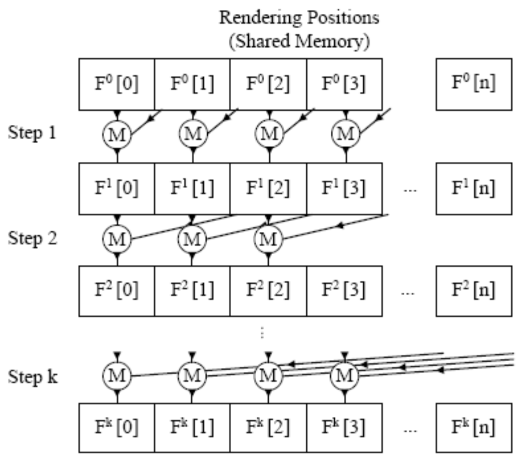

Rendering Positions
(Shared Memory)
Step 1
Step 2
Step k
M
$F^0[0]$ $F^0[1]$ $F^0[2]$ $F^0[3]$ $F^0[n]$
$F^1[0]$ $F^1[1]$ $F^1[2]$ $F^1[3]$ ... $F^1[n]$
$F^2[0]$ $F^2[1]$ $F^2[2]$ $F^2[3]$ ... $F^2[n]$
$F^k[0]$ $F^k[1]$ $F^k[2]$ $F^k[3]$ ... $F^k[n]$

Fig.7.The computing scheme of rendering positions. Parallel dynamic programming techniques are used on GPU to solve visibility problem.

III.EXPERIMENT RESULTS

To evaluate the algorithm, we compare the proposed algorithm with algorithms of two previous works. The analyses on visual quality and performance are shown in the following.

A. Visual Quality Analysis

The visual quality of the proposed algorithm was evaluated by comparing the result from three algorithms as the conventional motion-based algorithm. Motion-based algorithm was implemented based on [9].Four video sequences, Air,Fashion,Arctic,Cod from[16],and two video sequences, Akko& Kayo, Flamingo from MPEG Multi-view video coding were used to perform the subjective view evaluation. The results were evaluated using a slightly modified version of single-stimulus presentation method in ITU-R BT.500-10[15].The synthesized results were displayed on the 120Hz 3D display with active shutter glasses for evaluation. The subjective evaluation was performed by 20 individuals. The participants watched the stereoscopic videos in a random order and were asked to rate visual quality of each video. The overall quality of depth quality was assessed using a five-segment scale and mapped to 100 point scale.Fig9.shows the values of the two factors acquired by experiments for the six evaluation sequences.Fig.8.shows some examples of depth map.

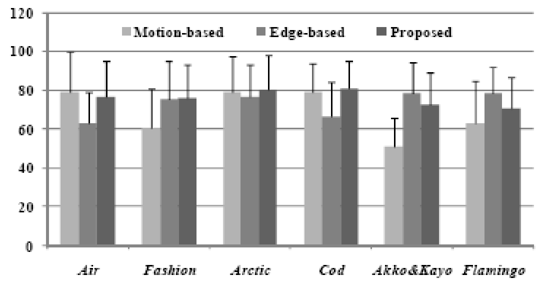


Fig.9.Subjective evaluation results.

The conventional motion-based algorithm as[9] relies on the quality of the motion vector. In the sequences with regular motions such as Air sequence, motion parallax is captured correctly and the depth has best protrusion effect among all. The regular motion implies that the object has a simple movement in the same direction. If the objects have complex self motions or varying lighting source such as Flamingo, or uncompensated ego motion as Fashion, the motion-based algorithm generates non-continuous or ill-predicted depth and makes viewers feel uncomfortable. In addition, the depth and makes viewers feel uncomfortable. In addition, the depth is not extracted correctly if the object is stationary or no relative motion exists.

The edge-based algorithm and the proposed algorithm have less side effects and yield good quality. Compared with the conventional motion-based algorithm that generates depth from multiple frames, the latter two methods use only a single image to generate depth. However, the quality of edge-based algorithm will drop if the assumption of the global depth does not hold or large foreground objects exist, such as Air. In comparison, proposed algorithm has texture cues and still generated satisfactory depth with little perceptible side effects.

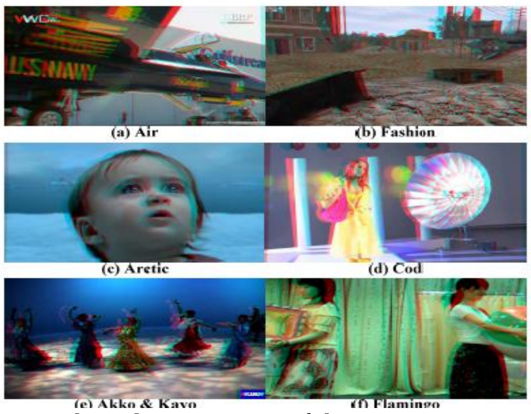


Fig.10.The red-cyan images of the six test sequences

From observation, we also discover an interesting phenomenon. Human visual perception still generated correct result even when the depth map of object is inverted. The phenomenon can also be found in the hollow-face illusions. When the light gradient on the surface is preserved, human visual system may overwrite the depth perception with daily life experience. Hence, texture gradient should play on important role on the depth perception. This could also explain the subjective quality test result of the proposed algorithm. The side effects are hard to discover even the depth is inverted. Finally, Fig.10 shows some examples of red-cyan stereoscopic images generated from the proposed algorithm.

B .Performance Analysis and Implementation

The system is implemented on a notebook computer, and integrated in a 3D video player software for evaluation.CPU of the notebook is an 1.60GHz quad-core CPU with 6M cache featuring simultaneous multithreading. The notebook has a 1.375GHz GPU with 7 stream processors inside. Each stream processor consists of 16 cores.

To compare performance of the three algorithm finally, all the algorithm is run on CPU with single thread. The performance is shown in TABLE1.We use our implementation of the motion estimation for the run-time of the motion-based algorithm. The time can be less if the motion vectors come from the decoder. As we can see, the proposed algorithm has relatively low computation time in computation time in comparison to the other two algorithms.

TABLE I
ALGORITHM PERFORMANCE COMPARISON

Algorithm	Average Performance
Motion-based Algorithm	~7231 ms / frame
Edge-based Algorithm	17261 ms / frame
Proposed Algorithm (Un-optimized)	901 ms / frame

With proposed optimization techniques,960*540p@30fps is achieved on the multi core CPU alone. With GPU acceleration
For DIBR,1920*1080@30fps video conversion is achieved. Because the DIBR is run on GPU,CPU usage is reduced to30%-50%.The specification of DIBR on GPU is shown in TABLE II. The usage of the shared memory is reported by compiler. Besides, the shared memory usage

TABLE II
SPECIFICATION OF DIBR ON GPU

Properties	Specification
Register used per thread	16 registers
Shared memory used per block	1440 bytes
Threads per block	64 threads
Rendering performance	1920x1080p@30fps
Speedup	26.4x

depends on number of threads per block. Since the shared memory resource is limited, we choose the number of threads for best balance. DIBR can be run on GPU efficiently with the proposed technique. With the proposed method, the performance of 2D-to-3D conversion is achieved for 1920*1080p video at real –time 30fps.

IV.CONCLUSION

An efficient algorithm and optimization of 2D-to-3D conversion are presented. The proposed algorithm uses simple assumption as depth cues with little side effects instead of combining computation-extensive depth cues. We demonstrate the system on a multi-core CPU and a GPU. Several techniques are proposed to optimize bandwidth bottleneck. Real –time performance is achieved in converting 1920*1080 @30fps.The proposed system is suitable for consumer 3D devices. In future, we may further integrate the whole system in the naked-eye multi-view 3D system for 3D applications.

REFERENCES

[1].S.B.Gokturk,H.Yalcin and C.Bamji,"A time of flight depth sensor,system description,issues and solutions","IEEE workshop on Real-Time 3D sensors and their use,2004.

[2].Sung-Yeol Kim,Sang-Beom Lee,and Yo-Sung Ho,"Three dimensional natural video system based on layered representation of depth maps,"in IEEE Transactions on Consumer Electronics,2006

[3].Wa James Tam, Carlos Vazquez, and Filippo Speranza"Threedimensional TV: A novel method for generating surrogate depth maps using color information" in SPIE Electronics Imaging, 2009

[4].Yong Ju Jung, Aron Baik, Jiwon Kim and Dusik Park,"A novel 2D-to-3D conversion technique based on relative height depth cue,"SPIE Electronics Imaging, Stereoscopic Dispays and Application XX, 2009.

[5]Chul-Ho choi, Byong-heon Kwon,and Myung-Ryul Choi,"A real-time field sequential stereoscopic image converter,"IEEE Transactions on Consumer Electronics,2004.

[6]Chao-Chung Cheng,Chung-Te Li,and Liang-Gee Chen,"A block –based 2D –to-3D conversion system with bilateral filter."International Conference on Consumer Electronics (ICCE), 2009.

[7]Chao-Chung Cheng,Chung-Te Li and Liang-Gee Chen,"An ultra low-cost 2D-to-3D conversion system,"SID Symposium Digest of Technica l papers,2010.

[8]Chao-Chung Cheng,Chung-TeLi,Yi-Min Tsai,and Liang-Gee Chen,"Quality-Scalable Depth-Aware Video Processing System,"SID Symposium Digest of Technical Papers,2009

[9]I.A.Ideses, L.P.Yaroslavsky, B.Fishbain,,
R.Vistuch,"3D from compressed 2D video, "Proceedings of SPIE, 2007

[10]Chao-Chung Cheng, Chung-TeLi, and Liang-Gee Chen,"A 2D-to-3D conversion system using edge information,"IEEE Transaction of Consumer Electronics, Aug.2010.

[11]Man HeeLee,and In KyuPark,"Accelerating Depth Image-Based Rendering Using GPU"lecture notes in computer science,2006.

[12]Yamada, K.., and Suzuki,Y.,"Real-time 2D-to-3D conversion at full HD 1080p Resolution"International Symposium on Consumer Electronics(ISCE),2009.

[13]Foley, James,vanDam,Andries,Feiner,Steven K.,Hughes,and John T.,"ComputerGraphics:Principles and Practice"Addison-Wesley,1990.

[14]AndreRedert,Robert-Paul Berretty,ChrisVarekamp,OscarWillemsen,JosSwillens,HansDriessen," Philips 3D solution: from content creation to visualization,"Proceedings of the Third International Symposium on 3D Data Processing,Visualization and Transmission(3DPVT),2006.

[15].ITU-R Recommendation BT.500-10(2000),
"Methodology for the subjective assessment of the quality of television pictures."

[16]W-Y.Chen,Y-L.Chang,L.-G.Chen,"Real-time Depth image based rendering hardware accelerator for advanced three dimensional television system,"IEEE Int.Conf.on Multimedia and Expo.2006.