論文2002-39SD-4-12

3단계 파이프라인 구조를 갖는 Multi-View 영상 디코더

(A 3-stage Pipelined Architecture for Multi-View Images Decoder)

裵 昶 熙 *, 梁 榮 日 **

(Chang-Ho Bae and Yeong-Yil Yang)

요 약

본 논문에서는 *multi-view* 영상 디코딩 알고리듬을 구현하는 디코더의 구조를 제안하였다. 현재까지 *multi-view* 영상 처리를 위한 하드웨어 구조에 관한 연구는 이루어지지 않았다. 제안한 *multi-view* 영상 디코더는 3 단계 파이프라인 방식으로 동작하며, 매 클럭마다 디코드된 영상의 화소 값을 추출한다. *Multi-view* 영상 디코더는 3 부분으로 구성된다. 노드의 값을 반복적으로 전송하는 Node Selector, 4개의 노드 값으로부터 각 화소의 값을 추출하는 *Depth Extractor*와 주어진 시점과 화소의 깊이 값으로부터 영상 평면에 투영되는 위치를 생성하는 *Affine Transformer*로 구성되어 있다. 제안된 구조는 MAX+PLUSII 설계 툴로 설계되었고 시뮬레이션을 수행하였으며, 동작 주파수는 30MHz이다. 제안된 구조를 갖는 디코더로 영상을 실시간으로 복원할 수 있다.

Abstract

In this paper, we proposed the architecture of the decoder which implements the multi-view images decoding algorithm. The study of the hardware structure of the *multi-view image* processing has not been accomplished. The proposed *multi-view images* decoder operates in a three stage pipelined manner and extracts the *depth* of the pixels of the decoded image every clock. The *multi-view images* decoder consists of three modules, *Node selector* which transfers the value of the nodes repeatedly and Depth Extractor which extracts the depth of each pixel from the four values of the *nodes* and *Affine Transformer* which generates the projecting position on the image plane from the values of the pixels and the specified *viewpoint*. The proposed architecture is designed and simulated by the Max+Plus II design tool and the operating frequency is 30MHz. The image can be constructed in a real time by the decoder with the proposed architecture.

Key words : *Multi-view images, pipelined architecture, image decoder*

* 正會員, (株)노키아 tmc, Manufacturing Engineering Team
(Nokia tmc Ltd., Manufacturing Engineering Team)

** 正會員, 慶尙大學校 電氣電子工學部 컴퓨터情報通信研究所
(GyeongSang National University, Division of Electric and Electronics Engineering)

※ 본 연구는 정보통신부에서 지원하는 대학기초연구지원사업으로 수행되었고, 반도체설계교육센터(IDEC)의 CAD Tool 지원을 받아 이루어졌슴.

接受日字:2000年7月25日, 수정완료일:2002年4月10日

I. Introduction

Information has various expression patterns such as letters, figures, voices, sounds, still pictures, moving pictures (movie and TV) etc.. The technical development in the fields of semiconductor, integrated circuit design, image compression and digital telecommunications makes it possible to deliver the information compounding many different media to

one medium. The successive development of the technology related to the multimedia makes it possible to test broadcast HDTV that affects to the industry and life through ISDN. By this development trend, the 3-dimensional TV will be raised as a next generation medium bringing out the revolution of the information system to transfer lots of data more than HDTV. The 3-dimensional image coding is one of the most important technologies in the 3D TV technologies. The 3-dimensional image processing requires the high image compression technology to transfer or store the image data because of its nature of a great number of data. In addition, it requires fast image processing algorithm and the parallel processing architecture for the applications.

The problem of the stereo image correspondence has been studied to obtain the depth information of the object.[1~3] And also it has been worked that a *multi-view image* system with the look-around capability is an extended stereoscopic system.[4,5] The concept of disparity compensation has extended to the *multi-view images* to store or transfer *multi-view images*.[4~7] The methods proposed in the reference 5 and 6 compressed the *multi-view images* by utilizing only the correlation between two consecutive views, ignoring the high correlation among all of the multi-view images. *Fujii et. al*[7] suggests the compressing method using the geometric constraint on *multi-view images*. These methods expressed *depth information and texture* by using the disparity compensation which is based on the affine transform from the *multi-view images*. In the reference 8, the *depth information* is determined by the block-based disparity compensation and then *texture* is extracted from the *multi-view images* by using a disparity compensation based on the affine transform and finally quality of the decoded image is improved. *Grammalidis et. al*[9] proposed the video coding technique which can be applied for *multi-view images* coding.

The study of the hardware architecture of the *multi-view images* processing has not been studied. In this paper, we propose the architecture of the *multi-view images* decoder to generate images at the specific *viewpoint* from the *depth information* and *texture* of *multi-view images* expressed by the methods in reference 7 and 8. The *multi-view images* decoder extracts the *depth* value of each pixel from the values of the *nodes* and calculates the projection point on the image plane for the given specific *viewpoint*. And then, the decoded image is generated by mapping the texture to each projected pixel. The proposed *multi-view images* decoder operates on the 3-stage pipelined manner and consists of *Node Selector*, *Depth Extractor*, and *Affine Transformer*. The proposed *multi-view images* decoder generates the *depth* value of the pixels of the decoded image every clock.

In section 2, the *multi-view images* decoding algorithm is explained. In section 3, the proposed hardware architecture of the *multi-view images* decoder with the 3-stage pipelined architecture is described. Conclusion is described in section 4.

II. Multi-view image decoding algorithm

In this section, we describe the *multi-view images* decoding algorithm generating the *multi-view images* at specific viewpoint from depth information and triangular patches with texture. When photographing the objects at the same time by using several cameras or by moving one camera, we can get slightly different images for the same object. These obtained several images are called the *multi-view image* set or the *multi-view images*.

Fig. 1 shows the *multi-view images* obtained by shifting the focus *of* the camera along the X axis satisfying the epipolar constraint. The locations *of* the focus are $(-2D, 0, z0)$, $(-D, 0, z0)$, $(0, 0, z0)$, $(D, 0, z0)$, and $(2D, 0, z0)$. The camera is oriented toward the z direction and the generated image is aligned to the X-Y plane. Fig. 2 shows the *multi-view image* encoding and decoding process.

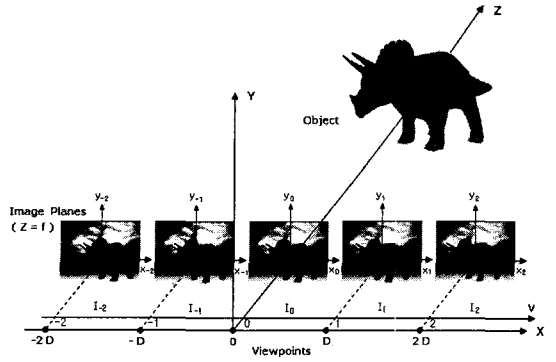


그림 1. 다시점 영상 셋
Fig. 1. The set of the multi-view images.

The *multi-view images* encoder normalize the object space and reconstruct the object in the normalized space.[7,8] In the normalized space, objects are expressed with the *depth information* and *texture* by the *multi-view images* encoder. The *multi-view images* decoder generates the images at specific viewpoints with the *depth information* and *texture*.

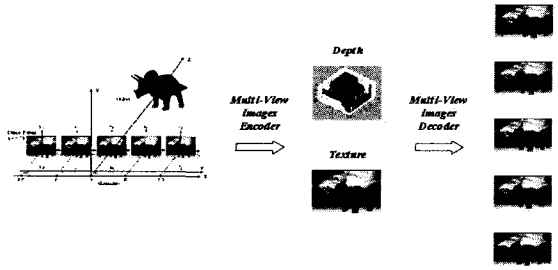


그림 2. 다시점 영상의 인코딩 / 디코딩 과정
Fig. 2. The process for encoding/decoding the *multi-view images*; *Depth* and *texture* are extracted from the *multi-view images* by encoder and the decoder generates the images at arbitrary *viewpoint* from the *depth* and *texture information*.

The *multi-view images* are compressed to the 3-dimensional *depth information* in normalized object space and *texture* mapped on the triangular patch. Fig. 3 shows the process of generating the *multi-view images* from the *depth information* and *texture* of the triangular patch extracted in the encoding process. The inputs to the *multi-view images* decoder consist of the depth information of the nodes, *texture* and *viewpoint*. The *depth information* of the *nodes* is made up with the first *depth information* of 8×8 pixels in normalized object space as shown in Fig. 3. It appears every 8 pixel on the *X-Y* planes and the value of the *node* is on the *Z* axis. *Texture* is the intensity value corresponded to triangular patch plane constructed with the values of the three adjacent *nodes*. The number of pixels is 256×256 and each pixel has the 256 gray levels with the 8 bits. *Viewpoint* is used to generate the images from the *depth information* of the *node* and *texture* at the specific camera position.

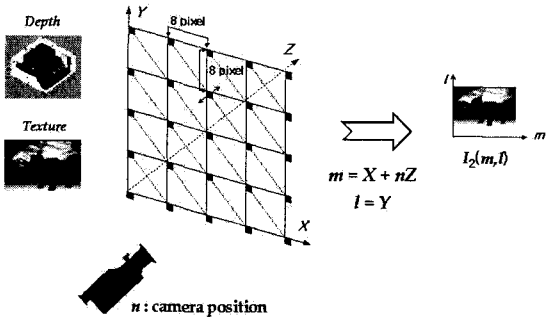


그림 3. 다시점 영상 디코더는 주어진 시점에서 영상 생성. $I_2(m, n)$은 시점 $n=2$에서 생성된 영상
Fig. 3. The *multi-view images* decoder generates the image for the specific *viewpoint*. $I_2(m, n)$ is the generated image when the viewpoint n is 2.

In Fig. 3, we can suppose that 3 dimensional triceratops is in the normalized object space from the *depth information* and *texture*. When the dinosaur in the object space is photographed from the specific camera position n, the relation between the *objects* in the normalized object space and the projected image $I_n(m, l)$is given Eq. (1) and Eq. (2).[7,8] The value m which is the X axis value of the image $I_n(m, l)$ is expressed as in Eq. (1). Because of the epipolar constraint, the value l which is the Y axis value of the image $I_n(m, l)$ is expressed as in Eq. (2).

$$m = X + nZ \quad (1)$$

$$l = Y \quad (2)$$

Algorithm 1 shows the pseudo-code of the decoding algorithm generating the images at the specific *viewpoint* from the *depth information* of the *nodes* and *texture*.

Input : *depth information of the nodes, texture, viewpoint*

Output : *the generated image at the specific viewpoint.*

Step1 : The *depth information* of the *nodes* composed of 33×33 is inputted in fours repeatedly and each pixel value of 256×256 images is extracted sequentially.

Step2 : By using each pixel value obtained from step1 and the specific *viewpoint*, the position projected on the image plane, that is, Eq. (1) is calculated.

Step3 : With mapping *texture* of triangular patches at each pixel projected, images are generated at the specific *viewpoint*.

알고리듬 1. 다시점 영상 디코딩 알고리듬의 슈도 코드
Algorithm 1: The pesudo-code of the algorithm for decoding the multi-view images.

Ⅲ. *Multi-view images* decoder with the 3-stage pipelined architecture

In this section, we describe the hardware architecture of the *multi-view images* decoder with the 3-stage pipelined architecture suggested in this paper. Fig. 4 is the block diagram of the *multi-view images* decoder. The *multi-view images* decoder of the proposed pipelined architecture consists of three parts, *Node selector* which transfers the values of the *nodes* repeatedly and *Depth Extractor* which extracts the *depth* value of each pixel from the values of the nodes and *Affine Transformer* which computes the projecting position on the image plane from the depth value of the pixels and specific *viewpoint*.

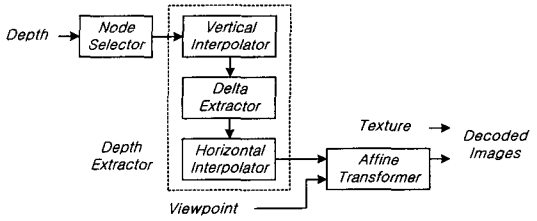


그림 4. 제안된 3단계 파이프라인 구조를 갖는 다시점 영상 디코더의 블록 다이어그램
Fig. 4. The block diagram of the proposed *multi-view images* decoder with the 3-stage pipelined architecture.

1. The architecture of *Node Selector*

The input of the *multi-view images* decoder consists of the depth information of the *nodes*,

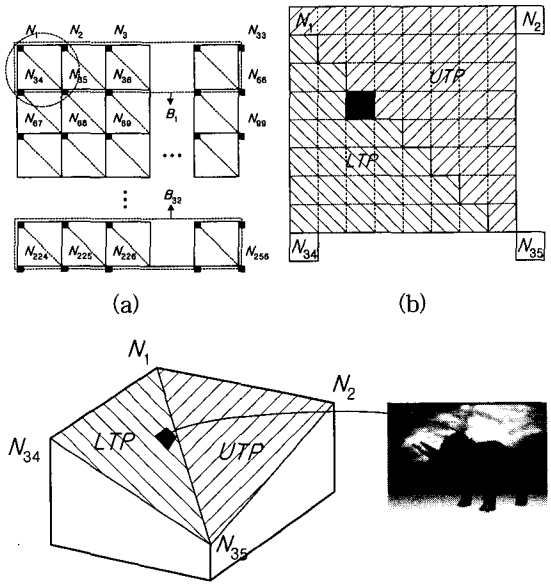


그림 5. 다시점 영상의 디코딩 과정
Fig. 5. The process for decoding the *multi-view images* (a) The pixel shaded with the dark (■) called *nodes* represents the pixel with the *depth* value (b) The enlarged drawing of the four adjacent *nodes* in (a); The 8×8 pixels are divided into two groups, *UTP* (*Upper Triangular Patch*) and *LTP*(*Lower Triangular Patch*). (c) The pixel with the *depth* value is mapped into the image by affine transform.

texture and *viewpoint*. The size of the texture is 256 ×256 and each pixel has the 256 gray levels with 8 bits. The *node* is defined every 8 pixels and the number of the *node* is 33 in row and 33 in column as shown Fig. 5(a). Only the pixels called *node* have the value and the value of the *node* is represented with 6 bits. Viewpoint is the location of the focus in the camera. The range of the *viewpoint* is $-8 \le n \le 7$ and the value is represented with 4 bits including the sign bit.

Fig. 5(b) shows the pixels bounded by the *nodes* N_1, N_2, N_{34}, and N_{35}. The rectangle with the 8×8 pixels is divided into two triangular patches called *UTP*(*Upper Triangular Patch*) and *LTP*(*Lower Triangular Patch*). Fig. 5(c) shows the 3-dimensional plot of Fig. 5(b). The pixel shaded with ■ is the pixels on *LTP*, so it has the position whose value is determined by the values N_1, N_{34} and N_{35}. The *depth* value of the pixels on *UTP* is determined by the *nodes* N_1, N_2 and N_{35}. As shown in Fig. 5(c), the node shaded with ■ is projected on the image plane by Eq. (1) and Eq. (2) and we will explain details in the description of *Affine Transformer* module. Fig. 6 is the circuit of *Node* Selector. The signal *Depth* in the circuit is the *depth* value of the *nodes*. The depth values of the nodes are inputted to *Node Selector* from the *depth* value of the *nodes* on the first row (N_1, N_2, ···, N_{33}) to the *depth* value of the *nodes* on the last row (N_{224}, N_{225}, ···, N_{256}) sequentially. To calculate the *depth* value of pixels on the 8 lines within the rectangle B_1 in Fig. 5(a) which is bounded by the *nodes* of the first row and the *nodes* of the second row, the value of the nodes on the first row and the second row are transferred in fours repeatedly. That is to say, N_1 is transferred to A, N_2 is to *B*, N_{34} is to *C* and N_{35} is to *D*. Because the rectangle B_1 consists of 8 lines, the four values of the *nodes* are transfered to *A*, *B*, *C* and *D* repeat 8 times. *Node Selector* repeats the upper process from the rectangle B_2 to the rectangle B_{32} sequentially until all the pixel *depth* values are obtained.

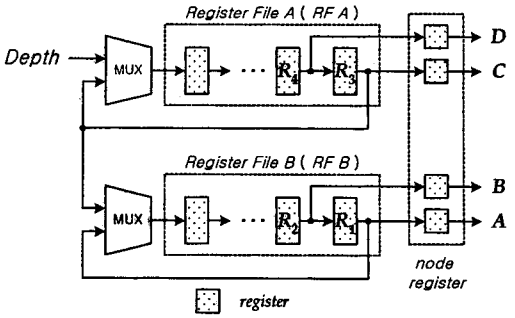


그림 6. Node Selector의 회로도
Fig. 6. The circuit diagram of Node Selector.

Node selector consists of 2 *MUXs*, 2 register files, *RF A* and *RF B* each of which can store the 33 *depth* values of the *node*, and four *Node* Registers which latch the value during the 8 clock periods. We will explain the behavior of *Node Selector* while the pixels are processed in the rectangle B_1 in Fig. 5(a). *RF A* and *RF B* in Fig. 6 have the *depth* values of the *nodes* on the second row and the first row, respectively. The registers R_1, R_2, R_3, and R_4 in Fig. 6 have the *depth* value of the *nodes* N_1, N_2, N_{34}, and N_{35} in Fig. 5(a), respectively. The values of Register File is shifted every clock, and then the next *node* values (N_2, N_3, N_{35}, N_{36}) is latched after the 8 clocks and transferred to the value of the *nodes A*, *B*, *C*, and *D*, respectively. To calculate the *depth* value of the pixels in the rectangle B_1 in sequence, *Node selector* repeats the upper process 8 times. When the pixels on the eight line in the rectangle B_1 is computed, the *depth* values of the nodes on the third row (N_{67}, N_{68}, ···, N_{99}) is transferred to the *RF A* and the values of the *RF A* is transferred to the *RF B*. Therefore *RF A* and *RF B* have the depth values of the nodes (N_{67}, N_{68}, ···, N_{99}) and (N_{33}, N_{34}, ···, N_{66}), respectively.

2. The architecture of Depth Extractor

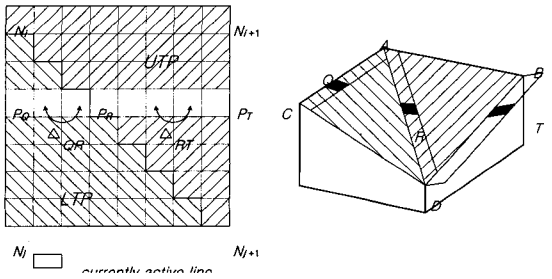
Depth Extractor calculates the value of the eight pixels on the activated line sequentially with the four values coming from *Node Selector*. Fig. 7 shows the process for the calculation of the eight pixels on the

fourth line which is the activated line from the values A, B, C and D which are the values of the pixels, N_i, N_{i+1}, N_j and N_{j+1}, respectively.

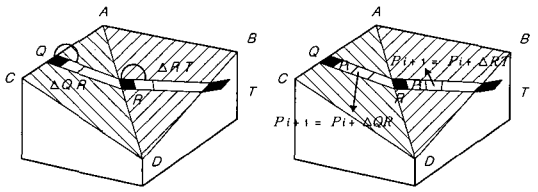
The activated line is the row which is under calculation. In Fig. 7(a), the fourth line is the activated line and the *depth* value of the pixels on the activated line should be calculated. The *depth* values of the pixels are calculated through the three steps. The first step is shown in Fig. 7(b). The value Q of the pixel PQ which is the first pixel on the activated line is calculated by linearly interpolating the value A (the *depth* value of the pixel Ni) and C (the *depth* value of the pixel N_j). The value R (the *depth* value of the pixel PR) and T (the *depth* value of the pixel PT) are also calculated by linearly interpolating the value A and the value D, the value B and the value D, respectively. In the second step which is shown in Fig. 7(c), the incremental amounts between the adjacent pixels on the activated line ΔQR and ΔRT are calculated from the values Q, R and T. In the final step, the *depth* value of the each pixels on the activated line are calculated with the values Q and R calculated in step1 and ΔQR and ΔRT calculated in step2. The final step is shown in Fig. 7(d). The pixel on the *LTP*(*Lower Triangular Patch*) are calculated sequentially by Eq. (3) from the values Q and ΔQR. The pixels on the *UTP*(*Upper Triangular Patch*) are calculated sequentially by Eq. (4) from the values R and ΔRT.

$$P_{i+1} = P_i + \Delta QR \tag{3}$$

$$P_{i+1} = P_i + \Delta RT \tag{4}$$



(a) (b)



(c) (d)

그림 7. activated line 상의 화소들의 깊이 값을 계산하는 과정

Fig. 7. The steps for calculating the depth value of the pixels on the activated line. (a) Fourth line is the activated line and the *depth* value of the pixels on the activated line should be calculated. (b) The values Q, R and T are computed in step1. (c) The incremental amounts between the adjacent pixels, ΔQR and ΔRT are extracted in step2. (d) In step3, the *depth* values of the pixels on the activated line are calculated.

Depth extractor operates on the 3-stage pipelined manner consisting of three blocks called three *Vertical* Interpolators, Delta Extractor and Horizontal Interpolator. Fig. 8 shows the block diagram of *Depth* Extractor which performs the computations of the step1 described above. The functional block *VI1* calculates the value Q of the pixel PQ from the A and C. The functional block *VI2* and VI3 compute the value R and T, respectively. *Delta Extractor* and *Horizontal Interpolator* perform the computation of the step2 and step3 described above, respectively.

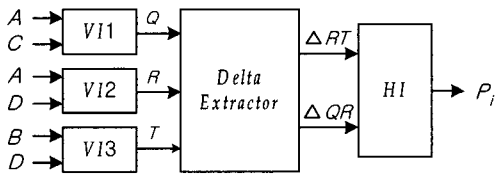


그림 8. 3개의 *Vertical Interpolator*, *Delta Extractor*와 *Horizontal Interpolator*로 구성된 Depth Extractor의 블록 다이어그램

Fig. 8. The block diagram of *Depth Extractor* with the 3-stage pipelined architecture consisting of three *Vertical Interpolators*, *Delta Extractor*, and *Horizontal Interpolator*.

Each functional block performs the computation within the eight clock periods. Fig. 9 shows the

pipelined operations of *Depth Extractor*. The three *Vertical Interpolators* calculate the values Q, R and T of the pixels P_Q, P_R and P_T on the *activated line* in the block $Block_{k+2}$ during the eight clock periods starting from the time T. Also, Depth Extractor computes ΔQR and ΔRT for the block $Block_{k+1}$ and Horizontal Interpolator outputs the depth value of the pixels on the activated line in the $Block_k$. During eight clock periods starting from the time $T+8$, *Vertical Interpolators* compute the values Q, R and T of the pixels on the activated line in the block $Block_{k+3}$ and *Depth* Extractor computes the ΔQR and ΔRT on the activated line in the block $Block_{k+2}$ and *Horizontal Interpolator* outputs the *depth* value of the pixels on the activated line in the $Block_{k+1}$ every clock. If we investigate the computation performed on the pixels on the activated line in the block $Block_{k+2}$, represented within the dotted line in Fig. 9, the values Q, R and T are calculated during the eight clock periods starting from the time T and ΔQR and ΔRT are calculated during the eight clock periods starting from the time $T+8$ and finally the *depth* value of each pixel is outputted every clock starting from the time $T+16$.

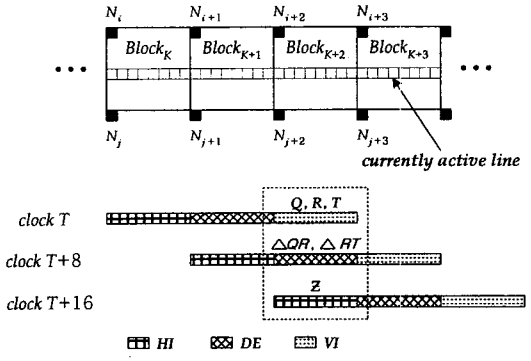


그림 9. Depth Extractor의 파이프라인 동작의 타이밍 다이어그램 ▭으로 표현된 영역은 Vertical Interpolator가 계산되는 블록이고 ▩와 ▭로 표현된 영역은 Depth Extractor와 *Horizontal Interpolator*가 계산되는 블록이다

Fig. 9. This figure shows the timing diagram of the pipelined operation of Depth Extractor. The shaded area with ▭ represents the block where Vertical Interpolator is computing. The shaded area with ▩ and ▭ represents the blocks where *Depth* Extractor and *Horizontal Interpolator* are computing, respectively.

3. The architecture of *Affine Transformer*

The corresponding point on the generated image at the *viewpoint* can be determined by the Affine transformation using in Eq. (1) with the *depth* value of the pixel and the *viewpoint*. In Fig. 10(a), the *depth* of the pixel P_i is Z and the texture of the pixel is represented as $T(P_i)$. The corresponding point on the generated images are shifted to the right linearly according to the increase of the

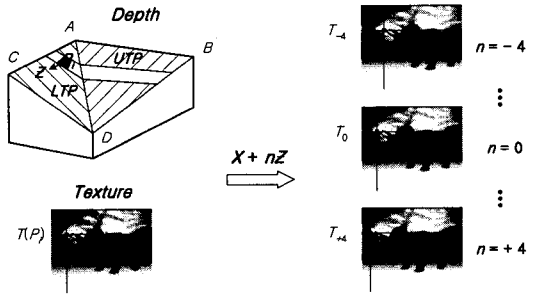


그림 10. 화소 P_i의 깊이 값과 시점 n으로부터 아핀 변환에 의해 결정되는 생성된 영상에서의 대응점.

Fig. 10. The corresponding point on the generated image is determined by the Affine transformation from the depth value of the pixel P_i and the viewpoint n. The *texture* on the pixel P_i is mapped to the corresponding point on the generated images.

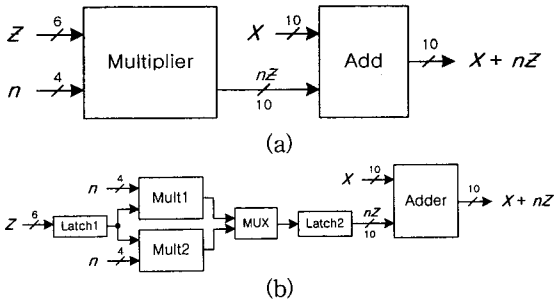


그림 11. (a) Affine Transformer의 블록 다이어그램 (b) Affine Transformer의 자세한 회로도

Fig. 11. (a) The block diagram of Affine Transformer. (b) More detailed circuit diagram of Affine Transformer

viewpoint n as shown in Fig. 10(b). The block diagram and more detailed circuit diagram are shown in Fig. 11, respectively, To compute Eq. (1) in the Affine *Transformer*, the multiplier and adder are used.

The proposed architecture is designed and simulated by the Max+PlusII design tool and the operating frequency is 30MHz. The number of gates used is approximately 27,000 which is the reasonable size in VLSI implementation.

Ⅳ. Conclusions

The multi-view images decoder with the 3-stage pipelined architecture is proposed. We designed the proposed architecture and simulated with Max+Plus II design tool. The *multi-view image* decoder consists of three modules called i) *Node Selector* which generates the node value repeatedly, ii) *Depth Extractor* which have the pipelined architecture of *Vertical Interpolator*, *Delta Extractor* and *Horizontal Interpolator*, and iii) *Affine Transformer* finding the corresponding point on the generated image for the computed value of the pixels. The corresponding points on the decoded image are outputted every clock and the image can be generated in a real time by the decoder with the proposed architecture.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, pp. 68~71, 1993.

[2] I. J. Cox, S. L. Hingorani and S. B. Rao, "A Maximum likelihood stereo algorithm," Computer Vision and Image Understanding, Vol. 63, No. 3, pp. 542~567, May 1996.

[3] U. R. Dhond and J. K. Aggarrwal, "Structure from Stereo - A Review," *IEEE Trans. Systems, Man, Cybernetics*, Vol. 19, No. 6, pp. 1489~1510, 1989.

[4] M. E. Lukas, "Predictive Cording of Multi-Viewpoint Image Sets," *IEEE ICASSP'86*, pp. 521~524, 1986.

[5] M. G. Perkins, "Data Compression of Stereo-pairs," *IEEE Trans. Commun.40*, pp. 684 ~696, 1992.

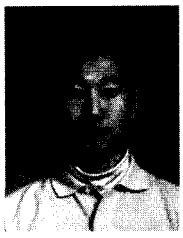
[6] H. Aydinoglu and M. H. Hayes III, "Compression of Multi-View Images," *IEEE ICIP'94*, pp. 385~389, 1994.

[7] T. Fujii and H. Harashima, "Data Compression and Interpolation of Multi-View Image Set," IEICE Trans. Inf. & Syst., E77-D, 9, pp. 987~995, 1994.

[8] D. H. Kim and Y. Y. Yang, "An Advanced Image Coding Algorithm for the Representation of the Set of Multi-View Images," *IEEK*, Vol. 35-S, No.7, 1998.

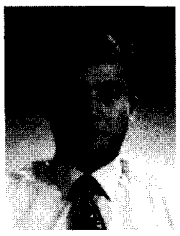
[9] N. Grammalidis, M. G. Strintzis, "Disparity and Occlusion Estimation in Multiocular Systems and Their Coding for the Communication of MultiView Image Sequence," *IEEE Trans. Circuits and Systems*, pp. 328~344, 1998.

저 자 소 개



裵 昶 熄(正會員)

1997년 2월 경상대학교 전자재료공학과 졸업(학사). 2000년 2월 경상대학교 대학원 전자재료공학과 졸업(석사). 2000년 1월~현재 NOKIA tmc. <주관심분야 : 3차원 영상처리와 ASIC 설계>



梁 榮 日(正會員)

1983년 2월 경북대학교 전자공학과 졸업(학사). 1985년 한국과학기술원 전기 및 전자공학과 졸업(석사). 1989년 한국과학기술원 전기 및 전자공학과 졸업(박사). 1990년~현재 경상대학교 전기전자공학부 교수. 1994년 1월~1995년 1월 UC, Irvine 교환교수. <주관심분야 : VLSI & CAD, 영상신호처리 등>