특집논문-07-12-4-04

Multi-View Video Processing: IVR, Graphics Composition, and Viewer

Junsup Kwon[a)], Won Young Hwang[a)], Chang Yeol Choi[a)], Manbae Kim[a)‡],
Eun-Young Chang[b)], Namho Hur[b)], and Jinwoong Kim[b)]

ABSTRACT

Multi-view video has recently gained much attraction from academic and commercial fields because it can deliver the immersive viewing of natural scenes. This paper presents multi-view video processing being composed of intermediate view reconstruction (IVR), graphics composition, and multi-view video viewer. First we generate virtual views between multi-view cameras using depth and texture images of the input videos. Then we mix graphic objects to the generated view images. The multi-view video viewer is developed to examine the reconstructed images and composite images. As well, it can provide users with some special effects of multi-view video. We present experimental results that validate our proposed method and show that graphic objects could become the inalienable part of the multi-view video.

Keyword: Multi-view video, IVR, Graphics composition

Ⅰ. INTRODUCTION

Multi-view video has recently gained much attraction from academic and commercial fields [1, 2, 3]. The multi-view video not only provides the wide viewing of natural scenes, but delivers immersive and realistic contents to users, especially when combined with synthetic contents such as computer graphics. In practice, due to the finite number of cameras and the large inter-camera distance for a wide view, intermediate views need to be reconstructed. The intermediate view reconstruction (IVR) requires accurate camera calibration, depth data, and texture image. From the texture image and depth data of camera views, novel views can be reconstructed. One of possible techniques is image-based rendering (IBR) that has attracted much attention in the past decade [4]. IBR aims to capture a real scene using a number of cameras. Any view of the scene can be generated from the camera views.

Previous works for generating virtual images were aimed at generating a novel image at any arbitrary viewpoint using image data acquired from the cameras. An effective way to generate virtual images is the utilization of depth and texture images [5, 6]. With depth and image data of all camera views, it is possible to reconstruct any arbitrary views. On the contrary, the research for mixing graphics objects into multi-view video is relatively few. The composition of the multi-view images being composed of camera and virtual views with graphic objects is an important subject, because

a) Dept. of Computer and Communications Engineering Kangwon National University
b) Broadcasting System Research Group, Radio & Digital Broadcasting Division Electronics and Telecommunications Research Institute
‡ 교신저자 : Manbae Kim (manbae@kangwon.ac.kr)
※A part of this paper was published in IWAIT 2007. This work was supported by the IT R&D program of MIC/IITA. [2007-S004-01, Development of Glassless Single-User 3D Broadcasting Technologies] and by the MIC, Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (GIST, IITA-2006-C1090-0502-0022).

composite scenes can increase and deliver better immersive perception. Furthermore, we have developed a multi-view video viewing software to test the quality of output images as well as the effect of special display functions.

In this paper we present multi-view video processing being composed of IVR, graphics composition, and viewer. The structure of the multi-view video processing is given in Fig. 1. Our approach to multi-view video composition with graphic objects includes two main steps. First we generate a virtual texture image between base cameras using depth and texture representation. Second we combine the obtained texture image of a virtual view with a synthetic graphic object. Also during the texture image generation, we generate its depth map. The depth map is stored in graphics Z-buffer and after image composition it is flushed. Based on camera calibration parameters and depth data, we reconstruct virtual intermediate images. As well, in order to carry out the image composition, the locations of graphic cameras are fixed to be at the identical places of the camera and virtual cameras so that the real scene and graphic objects are correctly registered. The texture image and depth data are obtained for both virtual and graphics views and a final composite image is made using Z-keying. Then, the output images are examined by a multi-view video viewer with special display functions to enhance the immersive perception.

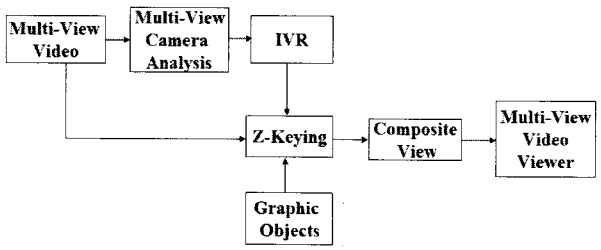


Fig. 1 The structure of the multi-view video processing

Following chapter describes the intermediate view reconstruction in details. The image composition with graphic objects is presented in Chapter Ⅲ. Chapter Ⅳ describes the multi-view video viewer. Finally, Chapter Ⅴ presents experimental results performed on multi-view video followed by the conclusion and future works of Chapter Ⅵ.

Ⅱ. IVR

We use three types of cameras in our method: base camera, virtual camera, and graphics camera. The base camera is a multi-view camera which captures video. With the virtual camera, new intermediate views are generated between two neighboring base cameras. The graphics camera is used to make graphic objects. The three cameras are set to accurately produce synthesized and composite views. For instance, camera parameters of the base cameras are implicitly or explicitly utilized. The position of a graphics camera coincides with that of a corresponding base or virtual camera

A distinctive feature is that we use an identical world coordinate system (WCS) for all types of cameras. The reason to use the identical WCS is the ability of full interactions between a real scene and graphic objects. Occlusion between them is automatically handled by a graphics engine because their Z-values are stored in the same Z-buffer.

Camera calibration parameters consist of two matrices: an intrinsic matrix K, which describes perspective projection and an extrinsic matrix [R | t], which describes camera orientation and position with respect to a world coordinate system [7]. A camera projects a 3-D world point to a 2-D point in an image. If the world and image points are represented by homogeneous vectors, then the mapping between their homogeneous coordinates can be expressed as

$$x = MX \tag{1}$$

where X represents a world point by the homogeneous vec-

tor (X, Y, Z, 1)T, and x represents an image point as a homogeneous vector.

The projection matrix M is expressed by

$$M = K [R \mid t] \tag{2}$$

where

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

and f is focal length, γ is aspect ratio, s is skew, and (x_0, y_0) is an image center.

A point x = (u, v) in an image can correspond to many 3-D points in the world. The ambiguity in determining the correct 3-D point can be resolved by using depth z of the pixel x, which is the distance along the principal axis from the camera center C to the 3-D point X which projects to x. Given depth z for every pixel x = (u, v) and the calibration matrix M, the corresponding 3-D point X can be given by the

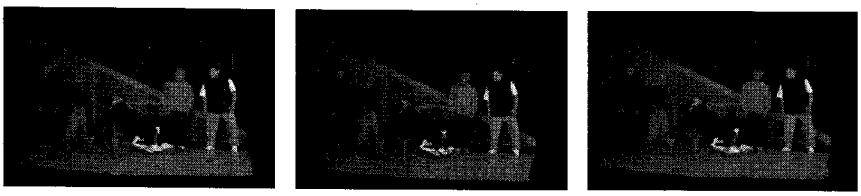
$$X = C + zM^{+}x \tag{4}$$

where X represents the 3-D point for the pixel x = (u, v), z is depth value corresponding to the pixel x, x is pixel homogenous coordinate (u, v, 1), M^{+} is the pseudo inverse of camera projection matrix M such that $MM^{+} = I$, and C is the camera center coordinate.

Now we present how we determine the virtual camera positions and orientations. To place any virtual camera in a proper position between selected base cameras, we have used information from these base cameras about how they are disposed in the space and in what direction they look. From camera extrinsic matrices of neighboring base cameras, we specify a camera extrinsic matrix [R | t] for a virtual camera to set its orientation and position. In case of parallel setup of the base cameras, a rotation matrix for any virtual camera should be identity matrix and we only need to specify a translation vector to set a virtual camera position between two chosen base cameras. For arc setup, we also need to specify a rotation matrix for any virtual camera to aim it in a proper direction, corresponding to viewing directions of the neighboring base cameras. First we compute a translation vector for every virtual camera. We simply compute delta (difference) between x coordinates of neighboring base cameras and do the same for z coordinates. Then we divide the delta into equal intervals according to a specified number of virtual views. And finally we set virtual camera centers at the joints of the intervals. The derivation of a rotation matrix is similarly carried out.

An algorithm to generate a texture image of a virtual camera is carried out as follows:

1) Specify two base cameras; base cameras 1 and 2 (see Fig. 2)
2) Project pixels from the base camera 1 back to 3-D space. For every pixel, its associated 3-D point is estimated by Eq. (4).



(a) (b) (c)

Fig. 2 Two virtual images are obtained (a) from the base camera 1 and (b) from the base camera 2. The final image is shown in (c).

3) Project all 3-D points reconstructed in 2) in virtual camera direction and obtain a new image.
4) Repeat 2) and 3) for the base camera 2 and update the new image.

The example in Fig. 2 illustrates the virtual views obtained from the proposed method. Fig. 2 (a) shows a virtual image obtained fromthe base camera 1. The black region contains pixels occluded from the base camera 1. The image in Fig. 2 (b) is a virtual view made from the base camera 2. Also, the black region occluded by the camera 2 is observed. The image in Fig. 2 (c) is the final image. Most of occluded pixels are solved. A red synthetic object is mixed according to a method described in the next chapter.

III. Graphics Composition

After obtaining texture and depth images of virtual views, we proceed to an image composition. The composition is identically applied to both camera and virtual view, Real-world Z-value is stored in the graphics Z-buffer so that when we insert a graphic object, occlusion between real scene and graphic objects is automatically resolved.

The mixing of the 3-D real-image-based scene and graphic objects is performed by Z-keying that compares pixelwise depth information [8]. To maintain the consistency between the real space and the graphics space, we need to use a unified set of camera parameters. Thus the parameters of the camera are used as the reference. We first use the camera parameters to set up a multi-view camera. In mixing graphics objects into the real scene, one of the important technical issues is the registration between graphic objects and the real scenes. To provide a natural-looking image, the exact position of the real camera must be known to place the objects correctly in the scene.

Now we present how we determine the graphics camera positions and orientations. To place any graphics camera in a proper position coinciding with selected base or virtual camera, we adopt their intrinsic and extrinsic camera matrices for the graphics camera and then we convert that matrices to the form, which is acceptable for the graphics engine.

To generate a graphic image, we first derive a relationship between base/virtual camera and graphics camera. Then, from the graphics camera corresponding to its associated base or virtual camera, a graphic image is made. For this we need to convert the intrinsic camera matrix K and the extrinsic camera matrix [R | t] of a base camera to the graphics projection matrix P and the graphics viewing matrix V accordingly. Remind that the base camera projection matrix M is M = K [R | t]. This matrix completely determines how 3-D world point described in the world coordinate system projects to 2-D point in the image plane.

We derive an analogous projection matrix for the graphics. First we convert [R | t] to the viewing matrix V by adding [0 0 0 1] to the bottom row.

$$V = \begin{pmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

The next step is to convert K into a 4 x 4 projection matrix P as follows [9].

$$P = \begin{pmatrix} \frac{2N}{R-L} & 0 & \frac{R+L}{R-L} & 0 \\ 0 & \frac{2N}{T-B} & \frac{T+B}{T-B} & 0 \\ 0 & 0 & \frac{-(F+N)}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{6}$$

where N is near plane, F is far plane, R is right plane, L

is left plane, B is bottom plane, and T is top plane of a view volume (e.g., frustrum).

For consistency, we set N equal to a focal length f. Other parameters are set automatically when we specify a vertical view angle and an aspect ratio as follows:

$$T=N\bullet\tan(\frac{\pi}{180}ViewAngle/2),B=-T,R=T\bullet AspectRatio,L=-R \quad (7)$$

P puts a view volume (frustum) into the canonical view volume (cube) with all dimensions from -1 to 1. Graphics camera projection matrix G is the product of a projection matrix P and a viewing matrix V, and expressed by

$$G = P\,V \quad (8)$$

Then, the projection of a 3-D world point X in homogeneous coordinate system to 3-D point Y in homogeneous camera coordinates is given by

$$Y = G\,X \quad (9)$$

The base camera projects a 3-D point in the world coordinate system to a 2-D point in the image coordinate system. Essentially, depth information is lost in this process. In homogeneous coordinate system, M is a 3 × 4 matrix, which maps a 3-D point to a 2-D point. Graphics camera projection is identical to base camera projection. In addition, the graphics camera projection also stores depth in normalized form, which is used in hidden surface elimination (e.g., z-buffer algorithm). Since a 3-D point is mapped to another 3-D point in the homogeneous coordinates, G is a 4 × 4 matrix. Taking into account the features of base and graphics cameras, the reason why we convert the camera matrix from the base camera to the graphics camera is that z-coordinate of 3-D point is stored in the z-buffer, since graphics camera projects a 3-D point to a 3-D point in contrast to base camera that maps a 3-D point to a 2-D point.

Z-keying method implies that on the input we have texture data for real and graphics images, and also we have depth data for them. Merging real and graphics texture images is done on comparing z-values of pixels from depth data of real and graphics images.

Ⅳ. Multi-View Video Viewer

The multi-view video viewer in Fig. 3 provides a tool to test the quality of the reconstructed images as well as the effect of graphics composition. Besides these, special effects could be tested for enhanced viewing of multi-view videos. $F_i(j)$ denotes the jth frame of the ith view. The main functions of the multi-view video viewer are introduced as follows.

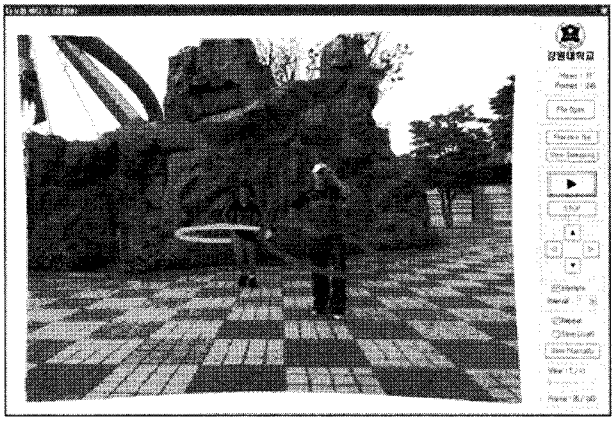


Fig. 3 Multi-view video viewer

A) *2D/3D display*: The viewer supports 2D and 3D stereoscopic video. In the 2D mode, a single view is rendered. The interlaced image of $F_i(j)$ and $F_k(j)$ is displayed in the 3D mode. The value of k can be varied based upon the depth control.

B) *Depth Control*: It is possible to control the degree of stereoscopic depth. For instance, a pair of stereoscopic im-

age can be made from $\{F_i(j), F_{i+1}(j)\}$, $\{F_i(j), F_{i+2}(j)\}$, ···, or $\{F_i(j), F_{i+n}(j)\}$. The larger the frame interval, n is, the larger depth is perceived.

C) *Special effects*: The following three effects are provided:

1) View switching: Users are able to switch flexibly from one view to another as the video continues along time.
2) Frozen-moment and rotate: In the frozen-moment and rotate, time is frozen and the camera view rotates about a given point. One example is that users can view frames $F_1(j)$, $F_2(j)$, ···, $F_n(j)$ back and forth at the jth frame of time instant. This effect is suitable for a convergent camera setup.
3) View sweeping: It involves sweeping through adjacent view direction while the time is still moving. It allows the user to view the event from different view direction. One example is that a user can view frames $F_1(j)$, $F_2(j+1)$, ···, $F_n(j+n-1)$ starting at the jth frame of the 1st view.

V. Experimental Results

In this chapter, we present experimental results that prove our proposed method and show that a graphic object becomes the inalienable part of a natural scene. For validation of our method, we have used a *breakdancing* multi-view sequence provided by Microsoft Research (MSR) as well as an *aerobic* multi-view sequence provided by ETRI. MSR provides camera calibration parameters of eight base cameras and 100 frames of video along with depth data [9]. Similarly, ETRI provides the necessary camera data. The number of cameras is eight and 300 frames are provided for each camera view.

The graphics objects are rendered in Open Graphics Library (OpenGL) environment [10]. In OpenGL, a texture image and its corresponding depth image are obtained from the front-color and depth buffer.

Fig. 4 shows two examples of graphics composition. Fig. 4 (a) shows mixing a graphic object into one of breakdancing images. Similar results of ETRI sequence are shown in Fig. 4 (b). *Snowman* and *hula-hoop* were used

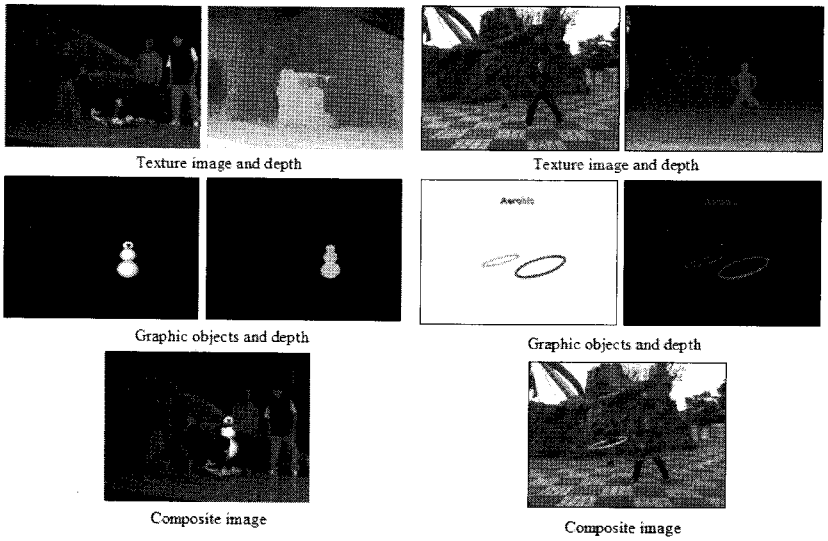


Fig. 4 Composite images of (a) MSR sequence and (b) ETRI sequence

for graphics objects, respectively.

Fig. 5 shows six composite views captured from a video that was made from camera 4 position. The graphic object snowman is designed in a manner that it moves around the dancer in order to show the performance of the proposed method and Z-keying. As we observe, our synthesized graphic object becomes the integral part of the scene, since any occlusion between graphic and real objects are successfully resolved by our algorithm.

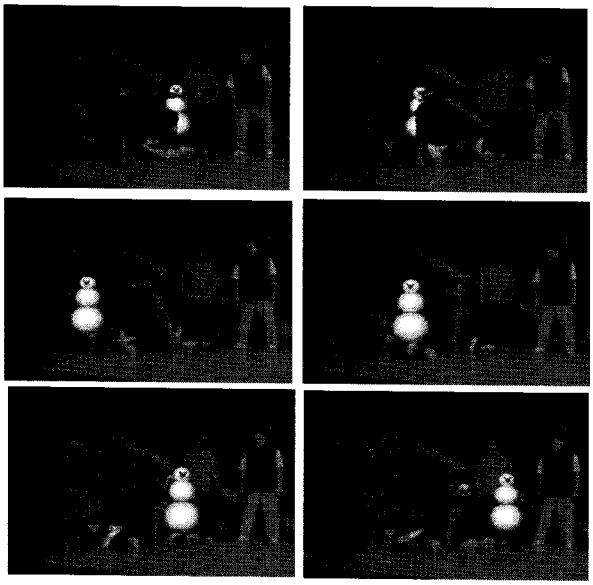


Fig. 5 The six successive images combined with a synthetic graphic object snowman

Fig. 6 shows the eight composite images chosen from the ETRI frames, $F_2(1)$, $F_2(20)$, $F_2(40)$, $F_2(60)$, $F_2(80)$, $F_2(100)$, $F_2(120)$, $F_2(140)$, $F_2(160)$, and $F_2(180)$.

Ⅵ. Conclusion and Future Works

In this paper we have presented multi-view video processing being composed of the synthesis of virtual intermediate views, the composition of graphic objects given multi-view sequences, and a multi-view video viewer. For virtual images, 3-D depth and texture information of base cameras is utilized. As well, for the accurate registration of real scene and 3-D graphics data, the theoretical relationship between them was derived. The multi-view video viewer is designed to test the quality of output images as well as special multi-view effects. Experimental results have shown that our method enables to generate high-quality synthesized images using information only from two neighboring base cameras so that graphic objects could become an inalienable part of immersive scenes. Furthermore, the occlusion between real and graphics objects is success-

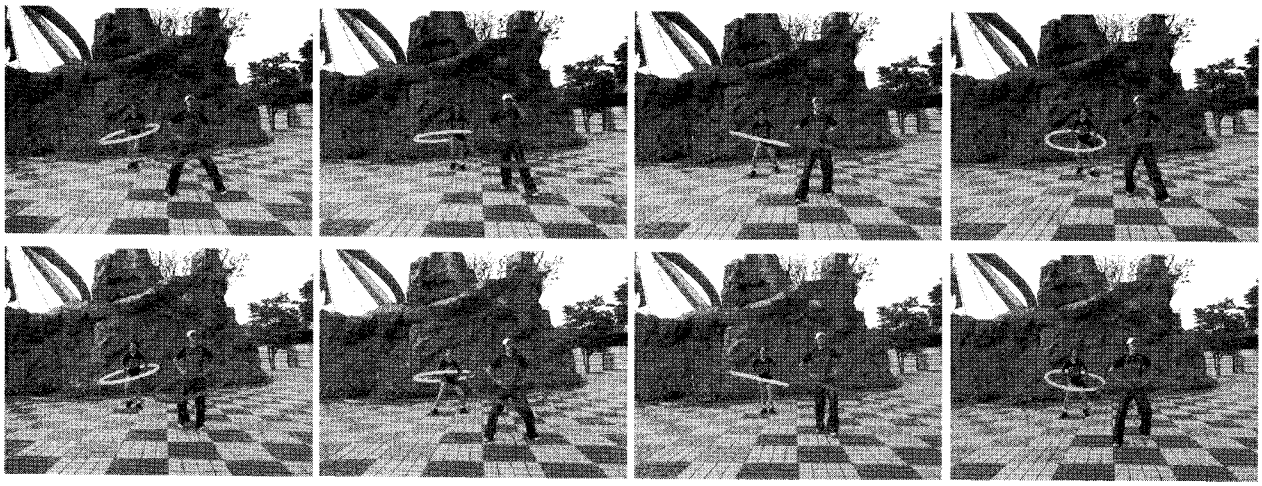


Fig. 6 The eight composite images combined with a graphic object *hula-hoop*

fully resolved. Our method is expected to deliver more immersive and realistic multi-view contents, especially when viewed in 3-D monitors.

References

[1] A. Vetro, W. Matusik, H. Pfister, J. Xin, "Coding approaches for end-to-end 3D TV systems," Picture Coding Symposium, Dec. 2004.

[2] Q. Zhang, W. Zhu and Y-Q Zhang, "Resource allocation for multimedia streaming over the Internet," IEEE Trans. on Multimedia, Vol. 3, No. 3, Sep. 2001.

[3] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-Quality Video View Interpolation Using a Layered Representation", SIGGRAPH04, Los Angeles, CA, USA, August 2004.

[4] C. Zhang and T. Chen, "A survey on image-based rendering - representation, sampling and compression", EURASIP Signal processing: image communication, vol. 19. no. 1, pp. 1-28, 2004.

[5] L. McMillan, "An image-based approach in three-dimensional computer graphics", Ph.D thesis, UNC, 1997.

[6] P. J. Narayanan, "Visible space models: 2½-D representations for large virtual environments, In ICVC99, 1999.

[7] Hartley, R.I. and Zisserman, A. Multiple View Geometry in Computer Vision, Cambridge University Press, 2nd Edition, 2004.

[8] W. Woo, N. Kim, and Y. Iwadate, "Photo-realistic interactive 3D virtual environment generation using multi-view video," In SPIE PW-EI Image and Video Communications and Processing (IVCP), Bellingham, WA, Jan. 2001, Vol. 4310, pp. 245-254.

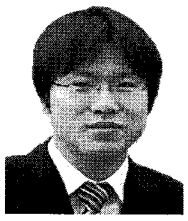
[9] http://research.microsoft.com/vision/InteractiveVisualMediaGroup/3DVideoDownload/, Microsoft Research.

[10] F. S. Hill, Jr. Computer Graphics using OpenGL, Prentice Hall, 2nd ed., 2001.

저 자 소 개

Junsup Kwon

- 2005년 : 강원대학교 컴퓨터정보통신공학과 학사
- 2007년 : 강원대학교 컴퓨터정보통신공학과 석사과정 (현재)
- 주관심분야 : 모바일 멀티미디어 스트리밍, 입체영상처리



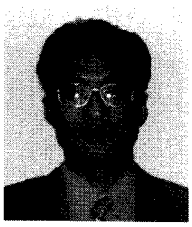
Won Young Hwang

- 2006년 : 강원대학교 컴퓨터정보통신공학과 학사
- 2007년 : 강원대학교 컴퓨터정보통신공학과 석사과정 (현재)
- 주관심분야: 영상통신, 입체영상처리

Chang Yeol Choi

- 1979년 : 경북대학교 전자공학과 학사
- 1981년 : 경북대학교 전자공학과 석사
- 1995년 : 서울대학교 컴퓨터공학과 공학박사
- 1984년~1996년 : ETRI 컴퓨터연구단 책임연구원 / 연구실장
- 1996년~현재 : 강원대학교 컴퓨터정보통신공학과 교수
- 주관심분야 : 컴퓨터시스템, 모바일컴퓨팅, 미디어서비스

저 자 소 개



Manbae Kim

- 1983년 : 한양대학교 전자공학과 학사
- 1986년 : University of Washington 전기공학과 공학석사
- 1992년 : University of Washington 전기공학과 공학박사
- 1992년~1998년 : 삼성종합기술원 수석연구원
- 1993년 : Georgetown University 의과대학 객원연구원
- 1996년 : University of Rochester 전기공학과 객원연구원
- 1998년~현재 : 강원대학교 컴퓨터정보통신공학과 교수
- 주관심분야 : 3D 비디오처리, 다시점영상처리, 3DTV 시스템

Eun-Young Chang

- 1999년 : 전북대학교 정보통신공학과 학사
- 2001년 : 광주과학기술원 정보통신공학과 공학석사
- 2001년~현재 : 한국전자통신연구원 전파방송연구단 방송시스템연구그룹 3DTV시스템연구팀 연구원
- 주관심분야 : 3D 비디오/CG 처리, 3DTV

Namho Hur

- 1992년 2월 : 포항공과대학교 전기전자공학과 학사
- 1994년 2월 : 포항공과대학교 대학원 전기전자공학과 석사
- 2000년 2월 : 포항공과대학교 대학원 전기전자공학과 박사
- 2000년 4월~현재 : 한국전자통신연구원 전파방송연구단 방송시스템연구그룹 3DTV시스템연구팀
- 주관심분야 : 3DTV, 3D DMB, Free-viewpoint TV

Jinwoong Kim

- 1981년 2월 : 서울대학교 공과대학 전자공학과 학사
- 1983년 3월~현재 : ETRI 근무 중 (전파방송연구단 책임연구원)
  TDX-1, TDX-10 개발 참여, HDTV 비디오 인코더 및 ASIC 칩세트 개발,
  MPEG-7, MPEG-21 기술 개발 및 표준화, 데이터방송, 맞춤형 방송 기술 개발,
  3DTV 기술 개발 과제 책임자
- 1993년 7월 : 미국 Texas A&M 대학교 전기공학과 박사
- 주관심분야 : 디지털 방송, 멀티미디어 응용 시스템, 3DTV