Journal of Korea Game Society 2017 Aug; 17(4): 17-24
http://dx.doi.org/10.7583/JKGS.2017.17.4.17

VR, AR

현실 객체 인식 기반 모바일 증강현실 게임

이동춘, 이헌주
한국전자통신연구원
{bluepine, hjoo}@etri.re.kr

Real Object Recognition Based Mobile Augmented Reality Game

Dong-Chun Lee, Hun-Joo Lee
ETRI(Electronics and Telecommunications Research Institute)

요 약

본 논문은 마커를 사용하지 않고 현실 객체 대상 모바일 증강현실 게임의 일반적인 제작 과정에 대해서 기술하고 있다. 본 논문에서는 모바일 환경에서의 성능 최적화를 위해서 slam 기술을 사용하여 만들어진 포인트 클라우드 데이터를 별도의 편집툴을 사용하여 편집하였다. 또한 게임 실행단계에서 특징점 추출 및 디스크립터 매칭으로 인해 많은 부하가 발생하는데, 이를 줄이기 위해서 이전 입력 영상에서 매칭된 특징점에 대한 위치 추적을 위해 Opticalflow 추적을 사용하였다.

ABSTRACT

This paper describes the general process of making augmented reality game for real objects without markers. In this paper, point cloud data created by using slam technology is edited using a separate editing tool to optimize performance in mobile environment. Also, in the game execution stage, a lot of load is generated due to the extraction of feature points and the matching of descriptors. In order to reduce this, optical flow is used to track the matched feature points in the previous input image.

Keywords : SLAM, 모바일 증강현실, Opticalflow, descriptor matching

Received: Jul. 10. 2017 Revised: Aug. 14. 2017
Accepted: Aug. 16. 2017
Corresponding Author: Dong-Chun Lee(ETRI)
E-mail: bluepine@etri.re.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

증강현실은 실사의 영상 정보로 부터 카메라의 포즈(위치 및 자세) 정보를 계산하고, 계산된 카메라 포즈 데이터를 사용하여 가상의 영상을 만든 후 이를 실사 영상과 합쳐서 사용자에게 보여주는 기술이다. 사용자가 보고 있는 현실객체에 대한 부가 정보를 사용자의 시점에 맞춰 보여 줄 수 있기 때문에 2000년대 이전부터 많은 사람들이 관심을 가지고 연구한 분야 중의 하나이다.

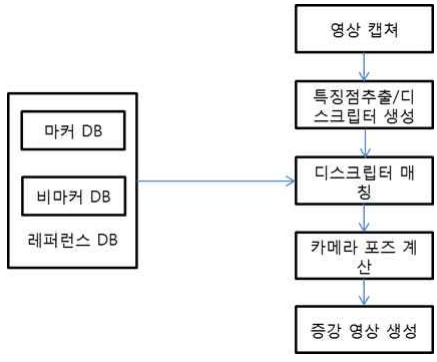
이러한 증강현실 기술은 초기에는 실내 환경에서 마커 위에 증강될 객체를 보여주었으나, 2008년 slam 기술을 이용한 PTAM이 소개 되면서[1] 실외 환경으로 그리고 비마커 기반으로 그 영역을 점차 넓혀갔다. 현재 증강현실을 위한 오픈 소스로는 PTAM, ATAM, ORB Slam 등 다양하게 있으며[1,2,3], 증강현실 콘텐츠 제작을 위해 제공되는 SDK로는 뷰포리아, Kudan, Wikitude, AR Toolkit 등이 있다[4,5,6]. 오픈 소스의 경우 대부분 SLAM(Simultaneous localization and mapping)[7,8] 기반의 증강현실 기술로써 PC 환경에서 구동이 되고 있고, 실세계 영상에 대한 3차원 포인트 클라우드 데이터를 생성하는 영상 복원에 초점이 맞추어진 반면, 증강현실용 SDK는 대부분 마커 기반에서 동작하거나 제한된 환경에서 비마커 기술이 동작한다. 증강현실 SDK를 사용하여 모바일 콘텐츠를 제작할 경우, 각각의 제품에서 제공하는 Unity Plug-In 프로그램을 사용하여 Unity 작업 환경에서 게임 객체를 원하는 위치에 배치 할 수 있으나, 오픈 소스를 사용하여 콘텐츠를 제작할 경우에는 Unity를 활용할 수 없어 게임 객체를 원하는 위치에 정밀하게 배치 할 수 없고, 게임 좌표계 및 Scale 정보가 카메라 초기화에 따라 매번 달라지는 단점이 있다.

본 논문에서는 마커를 사용하지 않고 현실 객체 인식 기반 모바일 증강현실 게임에 대해서 소개한다.

2. 본 론

비마커 기반의 증강현실 콘텐츠 제작을 위해서는 현실 공간과 가상 공간의 좌표계를 일치시켜야 한다. 이를 위해서 현실 공간에서 취득된 일련의 영상 정보로 부터 3차원 포인트 클라우드 데이터를 만들어 내는 맵 빌드 작업이 선행되어야 한다. 맵 빌드 단계가 완료되고 난 후 포인트 클라우드 데이터를 참조하여 게임 객체를 배치 및 이벤트를 등록하는 게임 제작 단계를 거치게 된다. 증강현실 콘텐츠 제작의 마지막 단계는 모바일 단말에서 실세계 객체 대상으로 증강현실 게임을 플레이하는 맵 매치 단계이다. 맵 매치 단계는 맵 빌드 단계에서 만들어진 포인트 클라우드 데이터를 읽어들인 후, 실시간에 카메라로 부터 입력되는 영상과 비교(매칭)하여 현재 영상에 대한 카메라 포즈를 계산한다. 이후 계산된 카메라 포즈를 사용하여 가상의 영상을 만들고 이를 실사 영상과 정합하여 증강현실 영상을 만들어 내게 된다. 일반적으로 증강현실 콘텐츠 제작은 맵 빌드, 게임 객체 배치, 게임 실행(맵 매치) 단계를 거치게 되나 본 논문에서는 맵 매칭의 속도 향상을 위해서 맵 빌드에서 만들어진 맵 데이터를 편집하는 단계를 추가 하였다.

2.1 증강현실 구성



[Fig. 1] Augmented Reality Flowchart

카메라 영상을 사용한 증강현실은 카메라로부터 입력된 영상에서 추출된 특징점 데이터를 레퍼런스

DB 에 저장된 특징점 데이터와 비교한 후, 매칭되는 특징점을 사용하여 카메라의 현재 포즈 정보를 계산하고, 계산된 카메라 포즈를 사용하여 증강 영상을 생성함으로써 이루어진다. 이때 계산된 카메라 포즈는 레퍼런스 DB에 저장된 카메라 좌표계를 기준으로 한 매트릭스 값이다. 이처럼 증강현실 영상 생성을 위해서는 레퍼런스 DB 데이터가 필요한데, 마커 기반의 증강현실의 경우 마커DB를 레퍼런스 DB로 사용하고, 비마커 기반의 증강현실의 경우 미바커 DB를 레퍼런스 DB로 사용한다. 레퍼런스 DB 데이터에는 이미지 데이터, 이미지 데이터에서 추출한 특징점 데이터, 이미지 상에 있는 특징점의 실제 3차원 위치 값 등의 정보가 저장되어 있다. 2차원 마커 기반 증강현실의 경우 마커 이미지 상의 특징점에 대한 3차원 위치 값을 쉽게 계산할 수 있어 특징점에 대한 3차원 위치값 계산을 위해 별도의 추가 작업이 필요하지 않으나, 비마커 기반 증강현실의 경우 이미지 상의 특징점에 대한 3차원 위치 값 계산을 위한 단계가 별도로 필요하다. 일반적으로 마커기반 증강현실의 경우 이미지 좌표 I(u,v)에 있는 특징점에 대한 3차원 위치 값은 P(u*s, v*s, 0)로 변환될 수 있다(s는 스케일 인자). 이에 반해 비 마커 기반의 증강현실의 경우 현실 공간 영상에서 추출한 2차원 특징점에 대한 3차원 위치값 계산을 위한 별도의 추가 작업(맵 빌드)이 필요하다.

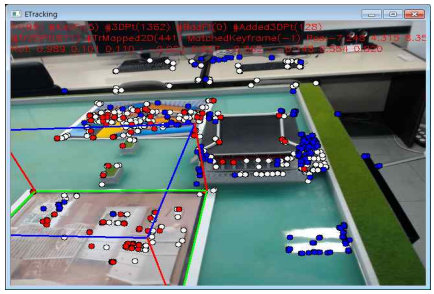
[Fig. 1]에서 보는 바와 같이 입력 영상에 대한 카메라 포즈 계산은 레퍼런스 DB에 저장된 영상과 카메라로 부터 들어오는 영상과의 비교를 통해 이루어진다. 즉 레퍼런스 DB에 저장된 영상에서 추출된 특징점으로부터 만들어진 디스크립터와 카메라 입력 영상에서 추출된 특징점으로부터 만들어진 디스크립터를 비교하여 매칭되는 특징점만을 필터링 한 후 PnP(Perspective-n-Point) 연산을 거쳐 카메라의 포즈를 계산 한다.

2.2 맵 빌딩

맵 빌드 과정은 증강현실을 위한 레퍼런스 DB 데이터를 만드는 단계로써, 현실세계의 영상 데이터로부터 특징점을 추출하고 추출된 특징점을 일정 시간 추적하여 궁극적으로 특징점에 대한 3차원 위치값을 계산하게 된다. PTAM, ATAM, ORB Slam 등 다양한 오픈 소스를 활용해서 특징점에 대한 3차원 위치값을 계산할 수 있으나, 본 논문에서는 ATAM 소스 코드를 기반으로 하여 특징점에 대한 3차원 위치값을 계산하였다. ORB Slam은 인접한 두장의 영상에서 매칭되는 특징점을 찾기 위해서 ORB 디스크립터를 사용하는 반면, ATAM의 경우 인접한 두장의 영상에서 매칭되는 특징점을 찾기 위해서 Optical flow 추적[9,10] 방법을 주로 사용하기에 ORB Slam에 비해 상대적으로 빠른 연산이 가능하다. ATAM은 ISMAR 학회에서 카메라 트래킹 경연을 목적으로 2015년 배포한 오픈 소스로써 OpenCV 함수를 사용하여 만들어졌기 때문에 OpenCV에 익숙한 사람은 이해하기가 편하다는 장점을 가진다. 일반적으로 맵 빌드 작업은 카메라 초기화 단계와 맵 확장 단계로 구성되어진다. 카메라 초기화 단계는 두장의 카메라 영상으로 부터 첫번째 영상에 대한 두번째 영상의 상대적인 카메라 포즈를 계산하는 단계로, 이를 위해서는 두 영상에서 공통으로 매칭되는 특징점에 대한 2차원 좌표 값이 필요하다. 공통으로 매칭되는 특징점들을 가지고 Essential 매트릭스를 계산하고, 계산된 Essential 매트릭스 값으로부터 첫번째 영상에 대한 두번째 영상의 상대적인 카메라 포즈 매트릭스를 계산한다. 또한 카메라 초기화 단계에서는 두 영상에서 공통으로 매칭되는 특징점에 대해서 삼각측량법을 사용하여 3차원 좌표 값을 계산한다.

두 장의 영상을 사용하여 카메라를 초기화하는 방법은 어떤 영상을 선택하느냐에 따라 게임 공간에 대한 Scale 정보가 변할 뿐 아니라, 첫번째 영상을 선택할 때의 카메라 포즈를 기준 좌표계로 사용하기 때문에 첫번째 영상을 어떤 영상으로 선택하느냐에 따라 기준 좌표계 역시 달라진다는 단점이 있다. 이 때문에 고정적인 카메라 좌표계를

사용하기를 원할 경우 마커를 사용하여 카메라를 초기화 하기도 한다. 특정 마커를 게임 공간에 배치한 후 마커를 인식하여 입력된 카메라 영상에 대한 초기 카메라 포즈를 계산하는 방법으로 가상 공간의 기준 좌표계가 마커의 기준 좌표계를 따르기 때문에 좌표계가 항상 고정되어 있고, 가상 공간의 Scale 값도 마커에 의해 고정되는 장점이 있다. 그렇지만 마커를 사용한 카메라 초기화 방법은 마커의 크기가 상대적으로 적기 때문에 넓은 실외 공간에 사용하기에는 힘들다는 단점이 있다.



[Fig. 2] Map Building Using Fiducial Marker

맵 확장 단계는 카메라 초기화 단계에서 계산되어진 카메라 포즈 값을 입력 영상의 변화에 따라 갱신하여 준다. 또한 맵 확장 단계에서는 3차원 좌표 값이 계산되지 않은 새로운 특징점의 위치를 일정 시간 추적한 후, 새로운 특징점에 대한 3차원 위치값을 계산한다. 일반적으로 새로운 특징점을 추가하거나 추적된 새로운 특징점에 대한 3차원 위치 값을 계산할 때, 모든 입력 영상에 대해서 수행하게 되면 연산량이 많아지기 때문에 특정한 중요 영상(키프레임 영상)에 대해서만 수행한다. [Fig. 2]는 기준 마커를 사용한 실내환경(경희루 디오라마)에서의 맵 빌드 과정을 보여주는 영상으로 기준 마커위에 와이어프레임으로 큐브를 세웠다. 또한 바닥면이 단일색이어서 특징점 추출이 되지 않아 바닥면에 레고 블럭 그림 이미지를 추가하여 특징점 추출을 용이하게 했다. [Fig. 3]은 맵 빌드 과정에서 자동 선택한 키프레임 영상의 예시이다.

a. 첫번째 키프레임 b. 두번째 키프레임

c. 네번째 키프레임 d. 일곱번째 키프레임

[Fig. 3] Selected Keyframes In Map Building

맵 빌드단계에서는 이렇게 만들어진 3차원의 맵 데이터를 저장하게 된다. 이 때 저장되는 데이터는 기본적으로 키프레임 영상 정보, 각각의 키프레임 영상에서 추출된 특징점의 2차원 이미지상에서의 위치값, 영상에서 추출된 특징점의 3차원 공간상에서의 위치값 데이터를 포함한다.

2.3 맵 데이터 편집

맵 데이터 편집 단계는 맵 빌드 단계에서 만들어진 3차원 점 데이터들을 편집하여 맵 매칭에서의 발생할 수 있는 오류나 연산 부하를 줄여주는 단계이다.

a. 3D 맵 데이터

b. 3D 맵 데이터+키프레임영상 정합

[Fig. 4] Raw Point Cloud Data

[Fig. 4]는 경회루 디오라마 대상 맵 빌딩 결과로 얻어진 포인트 클라우드 데이터를 가시화한 화면이다. 맵 빌드 단계에서 기준 마커로 사용할 마커를 경회루의 오른쪽에 배치하였고, 경회루 왼쪽 바닥면에는 많은 수의 특징점 추출을 위해서 레고 블럭 이미지를 배치하였다. 이 두개의 이미지가 없는 상태에서 맵 빌딩을 시도하였으나, 경회루 디오라마의 바닥면이 단일색이고 빛에 대한 반사가 많아 안정적인 맵 빌드가 불가능하였다. 경회루 디오라마의 맵 빌드를 위해 배치한 두개의 이미지는 증강현실 게임 플레이 시에는 불필요하기 때문에 맵 데이터 편집 단계를 통해 두개의 이미지에서 추출된 3차원의 위치값을 제거하였다.

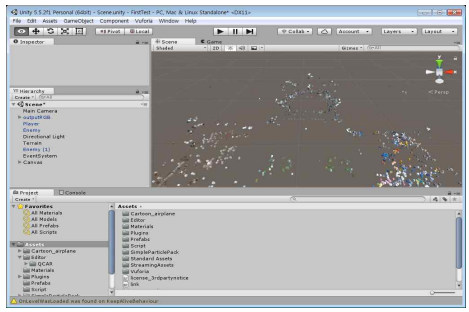
본 논문에서는 맵 데이터 편집을 위해서 특정 키프레임 영상의 삭제기능과 여러장의 키프레임 영상에서 공통으로 참조하고 있는 3차원 위치값의 삭제 기능을 추가하였다. 키프레임 영상의 삭제 기능은 연속된 여러장의 키프레임 영상 중 인접한 다른 키프레임 영상과 많이 겹치거나, 맵 빌드 단계에서 카메라 초기화를 위해 사용한 기준 마커 영상에 대한 키프레임 영상을 삭제하는데 사용 가능하다.

3차원 위치값 삭제 기능은 실제 증강현실 게임을 플레이 할 때 노이즈로 작용하는 3차원 위치값을 제거 할 때 사용된다. 경회루 디오라마 대상의 맵 빌딩에서는 왼쪽에 배치된 레고 블럭 이미지에서 추출된 3차원 위치값 데이터와 기준 마커에서 생성되는 3차원 위치값 데이터를 삭제하여 [Fig. 5]와 같이 최종 734개의 포인트 클라우드 데이터로 구성된 맵 데이터를 생성하였다.

[Fig 5] Edited Point Cloud Data

2.4 게임 객체 배치

본 논문에서는 게임 객체 배치를 위해서 Unity를 사용하였다. 증강현실 게임에서는 게임 객체 배치시 현실객체의 위치를 기준으로 하여 객체를 배치하여야 하는데, 이를 위해서 앞서 맵 빌딩 단계에서 만들어진 맵 데이터에서 3차원 위치 값만 별도로 빼내어 FBX 파일로 Export 했다. Export된 FBX 파일을 유니티에서 읽어 들인 후 이 데이터를 기준으로 해서 가상의 게임 객체를 배치하였다. [Fig. 6]은 FBX로 익스포터된 맵 데이터를 유니티 상에서 불러온 화면이다.



[Fig. 6] Raw Point Cloud Data On Unity

2.5 게임 실행(맵 매칭)

맵 매칭 단계는 맵 빌드 단계에서 만들어진 맵 데이터를 사용해서 실제 카메라 입력영상에 대한 카메라 포즈를 실시간 계산해 내고, 카메라 포즈에 맞추어 가상의 객체를 생성하여 실제 게임을 실행하는 단계이다. 맵 매칭 단계에서는 실시간으로 입력되는 카메라 영상에서 특징점을 추출하고, 추출된 특징점을 맵 빌드 단계에서 만들어진 특징점과 비교하여 매칭되는 점만을 필터링한다. 이렇게 매칭되는 특징점을 가지고 PnP 연산을 거쳐 실시간으로 카메라 포즈를 계산하게 된다. 모바일 단말에서 매 입력 영상에 대해서 특징점을 추출하고, 추출된 특징점과 레퍼런스 DB 데이터와 비교하는 작업은 많은 연산량을 필요로 하기 때문에 실시간 카메라 포즈 계산을 어렵게 만든다.

본 논문에서는 레퍼런스 DB 데이터와 입력 영상과의 비교 시간을 줄이기 위해서 이전 입력 영상에서 매칭된 결과 값을 활용하였다. 이전 입력 영상에서 레퍼런스 DB 데이터와 매칭된 특징점의 현재 입력 영상에서 위치 추적을 위해 OpticalFlow를 사용하였다. OpticalFlow 추적은 입력 영상으로부터의 특징점 추출 및 디스크립터 생성 과정이 필요 없고, 또한 레퍼런스 DB 데이터와 특징점 디스크립터 매칭 과정이 필요 없어, 이전 입력 영상에서 매칭된 특징점의 현재 입력 영상에서의 위치 변화 추적을 빠르게 할 수 있다. 또한 본 논문에서는 모바일 기기상에서의 맵 매칭 속도 향상을 위해서 매칭되는 특징점이 일정수 이상일 경우 특징점에 대한 디스크립터 매칭 과정없이 OpticlaFlow 추적만 수행하여 현재 입력 영상에 대한 카메라 포즈를 계산하였다.

3. 결 론

본 논문에서는 현실 객체 인식 기반 모바일 증강현실 게임기술을 사용하여 실내와 실외 환경의 객체를 대상으로 실험하였다. 실내 환경에서는 경회루 건물의 모형을 제작하여 실험하였고, 실외 환경에서는 건물 외벽을 대상으로 적용하였다. 맵 빌드와 맵 편집, 게임 제작 작업은 PC환경에서 수행하였고, 맵 매칭 단계는 모바일 단말(갤럭시 S7)에서 수행하였다. 맵 매칭시 입력 영상은 640x480 해상도의 영상을 사용하였고, 입력 영상으로부터 특징점 추출은 최대 400개로 설정하여 실험 하였다.

3.1 실내 환경 대상 실험(디오라마)

실내 환경에서 실험의 경우 경회루 건물이 회색조로 특징점 추출이 많이 이루어지지 않아 마커 이미지와 특징점 추출용 이미지를 디오라마 바닥에 부착하여 맵 빌딩을 하였다. 맵 빌딩된 데이터는 총 18장의 키프레임 영상과 2,819개의 포인트 클라우드 데이터로 구성되어졌다. 이중 바닥에 놓여진 이미지로부터 추출된 특징점과 디오라마 영역밖에서 추출된 특징점을 제거하여 734개의 포인트 클라우드 데이터와 17장의 키프레임 영상을 최종 맵 데이터로 활용하였다. 갤럭시 S7 모바일 단말상에서의 카메라 트래킹 속도는 30fps 이상을 보였다. [Fig. 7]은 맵 빌딩시 배치하였던 2장의 이미지를 없애고 모바일 단말상에서 실시간 카메라 트래킹 결과를 보여주는 화면으로 카메라 트래킹의 성능은 좌측 영상의 경우 31.6fps를 나타내었고, 우측 영상의 경우 35.8fps를 나타내었다.

[Fig. 7] Experimental Result for Indoor Object

3.2 실외 환경 대상 실험

실외 환경에서의 실험은 건물을 대상으로 하였다. 넓은 지역을 대상으로 하기 때문에 맵 빌딩시 기준마커를 사용하지 않고 사용자가 임의로 두 장의 영상을 선택하여 카메라를 초기화 하였다. 맵 데이터는 총 15장의 키프레임과 3,223개의 포인트 클라우드 데이터로 구성되어졌다. 맵 데이터의 편집없이 사용하였으며, 지면에 가상 몬스터를 배치하였고, 건물 유리창에 정합의 정확도 확인을 위해 2개의 WE:AR 로그 문자를 배치하였다. [Fig. 8]은 실외 건물 대상의 실험 결과를 보여주는 영상으로 카메라 트래킹의 연산 속도는 23fps 나타내었다.

[Fig. 8] Experimental Result for Outdoor building

3.3 실험 결과

본 논문에서는 비마커 기반의 현실 객체 인식 기술을 사용하여 실내 및 실외 객체를 대상으로 증강현실 콘텐츠를 실험하였다. 콘텐츠들은 트래킹이 잘되는 환경에서 23fps 이상의 성능을 보였으나, 트래킹이 안되는 환경에서는 10~13fps정도의 성능을 보였다. 트래킹이 잘되는 환경에서는 추적중인 특징점에 대해서 Optical flow 추적을 통해서 카메라 포즈를 계산하지만, 추적이 잘 안되는 환경에서는 입력 영상에 대해서 특징점을 추출하고 추출된 특징점에 대해서 디스크립터를 만든 후 이 디스크립터를 레퍼런스 DB 데이터와 매칭하여 카메라 포즈를 계산하기에 부하가 많이 발생하였다. 본 논문에서는 매칭되는 특징점이 30개 이상 유지되면 Optical flow추적만을 수행하여 카메라 포즈를 계산하였으며, 30개 이하로 떨어지면 Optical flow 추적과 디스크립터 매칭을 병행하여 카메라 포즈를 계산하였다. 실내외 증강현실 콘텐츠 실험에서 초기 카메라 포즈를 계산할 때는 10~13fps의 성능을 보였으나, 이 후 안정적으로 특징점을 추적한 후에는 실내 기준으로 31fps 이상의 성능을 보여 실시간 증강현실 콘텐츠 플레이가 가능하였다.

ACKNOWLEDGMENTS

This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2017.

REFERENCES

[1] G. Klein and D. Murray, “Parallel Tracking and Mapping on a Camera Phone”, ISMAR 2009, pp.83-86, 2009.

[2] ATAM download, https://github.com/CVfAR/ATAM

[3] Raúl Mur-Artal, et al, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System” IEEE Transactions on Robotics, Vol. 31, No. 5, pp.1147-1163, 2015.

[4] Kudan SDK site, https://www.kudan.eu/

[5] Wikitude SDK site, https://www.wikitude.com/

[6] ARToolkit Open Source site, https://www.artoolkit.org/

[7] S Thrun, JJ Leonard, “Simultaneous localization and mapping”, Springer handbook of robotics, 2008.

[8] E. Eade, P. Fong, and M. Munich, “Monocular graph SLAM with complexity reduction,” Intelligent Robots and Systems, pp. 3017 - 3024, 2010.

[9] Lucas, B., Kanade, T., “An iterative image registration technique with an application to stereo vision” Proceedings of Imaging Understanding Workshop, pp. 121 - 130, 1981

[10] OpenCV Optical Flow, http://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html

이 동 춘(Lee dong-chun)

약 력 2001- 한국전자통신연구원 선임연구원
1999-2001 경북대학교 컴퓨터 공학과 석사 졸업
1992-1999 경북대학교 컴퓨터 공학과 학사 졸업
관심분야 :
컴퓨터 그래픽스, 컴퓨터 비전, 증강현실

이 헌 주(Lee hun-joo)

약 력 : 1998- 한국전자통신연구원 책임연구원
1993-1998 중앙대학교 컴퓨터공학과 박사 졸업
1991-1993 중앙대학교 컴퓨터공학과 석사 졸업
관심분야 :
인공지능, 기계 학습, 컴퓨터 그래픽스, 증강현실