Journal of Korea Game Society 2020 Jun; 20(3): 35-44
http://dx.doi.org/10.7583/JKGS.2020.20.3.35

게임 프로그래밍

명령어 배치 인식을 활용한 AR 코딩퍼즐 모바일앱 개발

서범주, 조성현
홍익대학교 게임학부 게임소프트웨어전공
{bseo,scho}@hongik.ac.kr

Development of AR-based Coding Puzzle Mobile Application Using Command Placement Recognition

Beomjoo Seo, Sung Hyun Cho
School of Games, Hongik University

요 약

본 연구에서는 현재 운영 중인 코딩교육 플랫폼인 코딩퍼즐 시스템에서 학습자들이 직접 손으로 조작할 수 있는 탠저블 블록형태로 제작된 코딩퍼즐 입력용 명령어들의 배치를 증강현실 환경에서 제한 시간 안에 안정적으로 다수의 블록을 인식할 수 있는 인식시스템의 설계 및 배치 인식 성능 측정 결과를 제시한다. 그 결과, 5초 이내로 30개 이상의 탠저블 블록 형태의 명령어들의 배치를 안정적으로 인식할 수 있었다. 본 인식시스템을 기존 코딩퍼즐 모바일 앱에 성공적으로 이식하였으며, 블루투스에 연동되는 모바일 앱을 통해 IoT 로봇을 구동할 수 있다.

ABSTRACT

In this study, we propose a reliable command placement recognition algorithm using tangible commands blocks developed for our coding puzzle platform, and present its performance measurement results on an Augmented Reality testbed environment. As a result, it can recognize up to 30 tangible blocks simultaneously and their placements within 5 seconds reliably. It is successfully ported to an existing coding puzzle mobile app and can operate an IoT attached robot via bluetooth connected mobile app.

Keywords : Augmented Reality(증강현실), Mobile(모바일), Marker(마커), Placement Recognition(배치인식), IoT Robot(IoT 로봇), Movement Control(움직임 제어)

Received: May. 11. 2020 Revised: Jun. 04. 2020
Accepted: Jun. 07. 2020
Corresponding Author: Sung Hyun Cho (Hongik University)
E-mail: scho@hongik.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

코딩퍼즐 시스템은 본 연구진과 메이커스랩이 2014년부터 공동으로 개발해온 코딩 교육용 퍼즐 게임 플랫폼으로서 2018년 이래 무료 서비스 형태로 웹서비스를 제공하고 있다[1,2].

코딩퍼즐 시스템은 코딩교육용 퍼즐정보를 관리하는 Amazon Web Services(AWS)기반 클라우드 서버시스템, 웹페이지 상에서 구동되는 WebGL 기반 클라이언트 애플리케이션, 그리고 웹서비스와 독립적으로 구동 가능한 퍼즐정보들을 탑재한 안드로이드형 모바일 애플리케이션 등이 있다.

하지만 현재 구축되어 있는 시스템은 웹클라이언트, 모바일 앱 등의 형태로 구동되는 소프트웨어들로서 능동적인 학습, 팀 활동 같은 오프라인 활동 등을 요구하는 일반 코딩교육 환경에 활용되기에는 그 활용영역이 제한된다. 자칫 코딩퍼즐 교육 시스템에서 제공하는 게임 플레이가 코딩교육의 본질을 훼손하는 것으로 오해를 받을 수도 있다.

이에 능동적인 학습과 오프라인 활동을 강화하기 위해 본 연구에서는 손으로 조작 가능한 탠저블(tangible) 블록 형태의 코딩 명령어 활용 방법론을 제안한다. 여기에서는 탠저블 블록 형태로 만들어진 명령어들을 다수의 학습자들이 협력을 통해 가상의 로봇 아바타나 실제 움직일 수 있는 IoT 로봇[2]들을 구동할 수 있도록 가상공간이 아니고 실제 공간에서 명령어들을 배치하여 명령을 내리는 체험 학습형 코딩교육 교구를 제작하고자 한다.

본 연구를 통해 체험학습이 강화된 교구 활용에 필요한 접촉식 플라스틱(탠저블 블록) 모형 개발 및 증강현실(Augmented Reality, AR) 환경 하에서 복수개의 탠저블 블록을 인식하고, 그 배치 순서를 안정적으로 인식하는 인식시스템을 개발하고자 한다. 또한 성능 평가를 통해 개발한 인식시스템의 활용가능성을 모색하고, 기존에 개발한 모바일 앱과 통합함으로써 실제 운용 가능성을 확인하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 탠저블 명령어들을 인식할 수 있는 다양한 컴퓨터 비전 기반 인식 방법론과 관련된 연구들을 알아본다. 3장에서는 본 연구에서 목표로 하는 탠저블 블록에 대한 요구사항 및 제한사항, 인식시스템 개발 전에 수행한 사전 연구 등을 기술한다. 4장에서는 본 연구를 통해 개발한 명령어 배치 인식 알고리즘 및 기존 가정 등을 기술한다. 5장에서는 개발한 알고리즘에 대한 다양한 인식 관련 성능을 측정하고 측정 결과를 토의한다. 6장에서는 개발된 알고리즘을 기존 모바일 앱에 포팅하고 그 포팅결과를 제시한다. 마지막으로 7장에서 본 연구의 연구결과를 요약하고 향후 과제를 기술한다.

2. 관련 연구

본 장에서는 증강현실 공간에서 탠저블 블록들을 인식할 수 있는 다양한 기법들을 알아본다. 탠저블 블록 인식은 이미지 혹은 픽토그램 형식의 탠저블 명령어들의 인식과 복수개의 명령어를 실시간 인식 여부를 중심으로 기존 컴퓨터 비전에서 다루는 기법들을 알아본다.

컴퓨터 비전에서 다루는 기술들은 이미지 식별과 이미지 트래킹으로 구분된다. 여기에서는 이미지 식별 기술을 중심으로 관련 연구 동향을 알아보고자 한다. 컴퓨터 비전에서 다루는 이미지 인식 기법은 크게 마커리스(Marker-less) 기반과 마커 기반(Marker-based) 두 가지 형태로 분류할 수 있다[3].

마커를 사용하면 객체의 인식과 추적을 위한 정보를 비교적 쉽게 획득할 수 있다. 마커는 내부에 이진 패턴이나 모양을 가지고 있고, 3차원 좌표를 알고 있는 코너(corner)를 가지고 있기 때문에 간단한 영상처리로 쉽게 객체를 인식할 수 있다[3]. 이러한 장점으로 초기 증강현실 시스템들은 ARToolkit과 같은 SDK를 사용하여 마커를 인식하였다[3]. 그 이후에 ARToolKit를 기반으로 마커

인식 속도가 빠른 ARTag[4]가 개발되었으며, 저가 모바일 기기에서도 마커를 인식할 수 있는 ARToolKitPlus[5]가 개발되었다. ARTag나 ARToolKitPlus보다 성능이 우수하고 마커의 가려짐에도 마커를 인식할 수 있는 ArUco[6]가 소개되었다.

코르도바대에서 개발한 ArUco 마커는 기준 마커(fiducial marker)중 오픈소스로 널리 알려진 라이브러리로서 원하는 개수만큼의 마커들을 생성할 수 있다[6]. 검은 테두리가 있는 정사각형 공간에 동일크기의 흑백 형태의 블록들을 배치하고 블록들의 배치정보를 빠르게 인식하여 원하는 마커들을 구분할 수 있게 한다. 마커 간 거리를 최대화하여 오류를 최소화하고, 마커의 포즈를 추정할 수 있지만 카메라 시야에 마커 형태가 모두 보여야 한다.

마커를 사용한 시스템은 마커의 가려짐이나 인위적인 마커의 시각적 불편함으로 몰입감을 저해하는 단점이 있지만 마커의 단점이 크게 문제가 되지 않는 오락이나 교육 분야에서는 여전히 마커를 사용하고 있다[3]. 본 연구에서도 ArUco 기반 마커 인식 기술을 사용하고자 한다.

최근에는 2D나 3D 마커리스 객체와 62.5%가 가려진 마커를 성공적으로 인식하는 Vuforia SDK도 소개되었다[7,8]. 또한 눈에 띄는 마커의 가림 문제를 해결하기 위하여 눈에 보이지 않는 자외선 마커가 제안되기도 하였다[9].

마커리스 인식 방법은 특징점을 추출하고 이 특징점을 기반으로 좌표계를 추출해 내는 방식이다. 텍스처(Texture), 에지(Edge), 템플릿 (Template) 기반으로 세분하기도 한다[3]. 객체의 3차원 모델이 텍스처를 가지고 있는 경우에는 텍스처에서 객체의 특징을 검출하고[10], 텍스처 정보가 충분히 있지 않은 경우에는 일반적으로 에지를 특징으로 이용하여 객체를 검출한다[11]. 그 외에 객체의 투영된 모델과 영상의 픽셀의 밝기 차이를 이용하여 객체를 검출하는 템플릿 기반 방법도 있다[12]. 마커리스 인식 기법은 마커기반 인식 방법에 비교하여 객체 인식에 많은 계산이 필요할 뿐만 아니라, 우리의 연구 환경인 모바일 기기에서 많은 객체를 동시에 인식하는데 한계가 있기 때문에 본 연구에서는 마커기반의 방법을 사용한다.

3. 명령어 인식

본 장에서는 본 연구에서 사용한 ArUco 기반 마커 인식 기술을 소개한다.

(a) 3D-printed blocks (b) Command Panels in Mobile App

[Fig. 1] Tangible Block Prototypes and Their Designated Codingpuzzle Mobile App

3.1 인식 수준 요구사항 분석

탠저블 블록은 학습자가 직접 손으로 조작할 수 있는 3D 프린터로 출력된 정사각형 플라스틱 모형이다([Fig. 1](a) 참조). 모형 상판에는 AR 앱에서 인식 가능한 스티커 형태로 디자인된 명령어를 부착 및 탈착할 수 있다.

초기 블록에 부착하는 스티커 도안은 [Fig. 1](a)에서 보는 바와 같이 픽토그램 형태의 이미지를 인식할 수 있는 마커리스 기반 명령어로 고안되었으나 사전 실험을 통해 인식 안정성과 실시간 인식 측면에서 원하는 수준의 성능이 나오지 않았기 때문에 [Fig. 2]에서 제시하는 바와 같은 마커기반 명령어들로 디자인을 변경하였다.

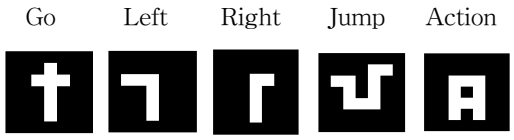
이전 연구[1]에서 개발한 코딩퍼즐 모바일 앱에서는 함수당 입력 가능한 최대 명령어개수가 그림 [Fig. 1](b)에서 보듯이 Main 함수는 16개, G1 함수는 8개, G2 함수는 8개이다. 각 함수별로 탠저블 블록들을 인식시키려면 최대 동시 인식 가능한 탠저블 블록들의 개수는 16개이다. 또한 증강현실 환경에서 탠저블 블록들의 최대 인식 시간을 5초로 한정하고자 한다.

이러한 명령어 인식 요구사항들을 요약하면 마커기반 인식시스템은 5초(실시간 요구조건) 이내에 최대 16개(중복된 명령어들 포함)의 명령어들을 동시에 인식(동시인식 요구조건)해야 한다.

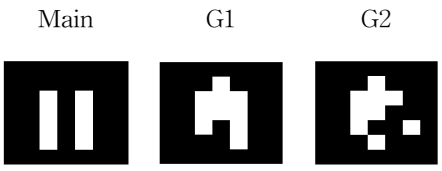
3.2 마커리스 인식

본 연구에 앞서 AR 공간에서 마커리스 기반 인식시스템의 가능성을 연구하였다. 당시 마커리스 기반 인식시스템으로 널리 쓰이는 Vuforia SDK를 유니티 환경에 적용해 보았다[8]. 당시 해당 SDK에서 제공하는 이미지 인식 기능은 촬영이미지에 대해 한 개의 이미지 인식 기능만 제공하여 동시 인식 요구조건을 만족하지 못했다. 현재 해당 SDK는 촬영한 이미지에서 최대 인식 가능한 이미지 개수가 4개로 제한되고 있다.

이에 본 연구진은 컴퓨터 비전 기술을 집대성한 OpenCV 라이브러리를 직접 활용하여 복수개의 이미지를 동시에 인식하는 방법을 모색하였다. 모바일 앱 형태로 인식시스템을 구동하는 것을 목표로 했기 때문에 유니티 상에서 구동 가능한 OpenCV 에셋인 OpenCV for Unity(OpenCV 버전 3.0)를 기반으로 하여 SIFT 특징점 추출법[13]을 적용하여 이미지를 학습시키고 실제 공간에서 복수개의 명령어를 인식시키는 실험을 진행하였다. 실험에서 동시에 인식 가능한 마커리스 이미지의 개수가 카메라의 해상도와 주변 환경에 심한 영향을 받았으며 3~6개 정도의 명령어들을 1초 이내 인식하는 것이 가능하였다. 이미 인식된 명령어들을 제거하고 동일 촬영이미지에 대해 재인식시킴으로써 1~2개의 명령어들을 추가로 인식할 수 있었으나 안정적으로 인식되지는 않았다. 따라서 특징점 추출 인식 속도에 강점이 있는 SURF[14]나 ORB[15] 특징점 추출법을 적용하였다. 이들 방법은 동일 촬영이미지에 대해 SIFT추출법보다 빠른 인식 속도를 보여주었으나 인식 안정성이 더욱 떨어졌으며 동시 인식 가능한 명령어의 개수가 매우 가변적으로 변동되었다. 따라서 마커리스 기반 명령어 인식시스템은 본 연구에서 채택할 수 없었다.



(a) Executable Commands: Go, Turn Left/Right, Jump, Action



(b) Function Commands: Main, G1, G2

[Fig. 2] ArUco Markers used as Tangible Blocks

3.3 마커기반 명령어 블록

[Fig. 2]에서는 최종 선정된 마커기반 명령어들로 일반 명령어들은 5종, 함수 형태로 호출이 가능한 3종의 함수 명령어들로 구성된다[1]. 일반 명령어들은 코딩퍼즐에서 퍼즐을 푸는 로봇 아바타를 움직일 수 있는 전진, 우로 90도 회전, 좌로 90로 회전, 점프, 문맥에 따라 다양한 기능을 수행하는 액션으로 구성되어 있다. 함수 명령어는 Main, G1, G2가 있다.

명령어들은 OpenCV에서 샘플 애플리케이션으로 제시된 ArUco 마커들로서 테두리를 제외하고 5x5 정사각형 그리드공간에 배치되었다. 촬영이미지에서는 등록된 8종의 마커들을 인식하고 무작위 순으로 탐지한 마커들의 사각형 꼭지점 4개와 마커식별자들은 OpenCV API를 통해 얻었다.

4. 블록 배치 인식

ArUco 스티커 마커가 부착된 탠저블 블록들을 AR 공간에 배치하고 촬영된 이미지 프레임별로 등록된 8종의 마커들을 인식시키면 식별된 명령어 마커들의 3차원 공간 좌표 정보가 무작위순서로 반환된다. 개별 마커들의 공간 좌표 정보들로부터 명령어 입력 순서에 맞게 배치를 인식하기 위해 다음과 같은 배치 인식 알고리즘을 고안하였다.

4.1 가정

탠저블 블록들은 증강현실용 기기에서 사용하는 카메라에서 한 번에 모두 인식될 수 있도록 배치되어야 한다. 그러나 카메라가 모든 함수들의 블록 배치를 한 번에 인식하기 어렵기 때문에 함수별로 블록들을 인식한다고 가정하였다. 블록들은 물리 공간상에 연속으로 배치되어야 하며 일반적으로 널리 통용되는 왼쪽에서 오른쪽 순으로 배치되며, 한 줄로 모든 명령어들을 나열할 수 없기 때문에 바둑판 형태로 명령어가 나열된다고 가정하였다. 요약하면, 가상의 그리드 공간에 블록들이 왼쪽 상단에서 오른쪽 하단 순으로 배치된다.

4.2 블록 배치 인식 알고리즘

본장에서는 3차원 공간상에 배치된 마커들을 2차원 공간으로 매핑하고 가상의 그리드 공간에 배치된 순서를 계산하는 알고리즘을 기술한다([Algorithm 1] 참조).

먼저, 첫 단계에서는 매 프레임별로 이미지를 획득한 후 OpenCV에서 제공하는 마커 탐지 알고리즘을 적용하여 탐지된 모든 마커들을 무순위로 마커 리스트에 저장한다. 마커 정보에는 마커를 탐지 시 인식한 마커 식별자와 3차원 공간상의 위치 정보를 포함하고 있다. 이 정보로부터 2차원 위치 정보를 추출하고 인식 카메라의 카메라 방향에 맞추어 2차원 위치 정보를 재가공(단계 3.2)한 후에 위치정보 리스트(P)에 저장한다.

단계 3.2에서 모바일 기기의 경우, landscape, portrait, landscape(flipped), portrait(flipped)와 같이 4개의 상이한 orientation이 존재하기 때문에 orientation별로 상이한 좌표 값이 나타난다. 동일한 알고리즘이 적용되도록 landscape을 기준으로 DirectX에서 차용한 좌상귀를 스크린 원점(0,0)으로 하는 2차원 좌표계로 변환하였다.

1. 마커 리스트(M) ←프레임별 탐지된 마커들
2. 마커 2D 위치(P) ←∅
3. ∀m ∈M
   3.1 p(m) ← m의 2차원 위치
   3.2 카메라 방향에 맞추어 p(m) 보정
   3.3 P ∪= p(m)
4. 최종 결과값(F) ← ∅
5. y^{avg} ← 모든 마커들의 평균 높이를 구함
6. while (|M| ≠0)
   6.1 $y_{\min} \leftarrow \min_{m \in M} y(m)$
   6.2 $flt \leftarrow \{m | y(m) < y_{\min} + y^{avg}, m \in M\}$
   6.3 P -= p(m) where m∈ flt, M -= flt
   6.4 x축으로 flt 정렬하고 순서를 F에 저장
7. return F

[Algorithm 1] Block Placement Recognition Algorithm

이후 아직 배치가 완료되지 않은 마커들 중에서 y값이 가장 작은 마커를 찾아 평균 높이보다 작은 y값을 가진 모든 마커들을 찾아(단계 6.2) x값 오름차순으로 정렬한 결과를 결과 리스트에 저장한다(단계 6.4). 남아 있는 마커들이 없을 때까지(단계 6) 이 과정을 반복함으로써 모든 마커의 배치를 완료한다.

본 배치 인식 알고리즘은 3차원 위치 정보를 2차원 위치 정보로 단순화하였기 때문에 2차원 공간상에서 블록의 크기가 대동소이한 것을 가정하고 있다. 따라서 블록 인식 시 인식카메라와 블록이 배치된 평면 공간이 수직 방향일 때 가장 효율적

으로 인식한다. 반면 카메라를 블록 배치 평면 공간에 비스듬하게 촬영하면 상단과 하단의 블록 크기 오차가 커지게 되어 배치 인식 오류가 커진다.

5. 배치 인식 성능 평가

본 장에서는 사용한 마커 인식 및 배치 인식 측정 환경 및 성능 측정 요소, 측정 결과 및 결과 분석 내용을 다룬다. 5.1에서는 인식 성능 측정에 사용한 실험 환경을 설명하며 5.2, 5.3, 5.4에서는 카메라와 탠저블 블록간의 최적의 인식 측정 거리, 인식 시간, 인식 정확도 등에 대한 성능 평가 결과를 기술한다.

5.1 실험 환경

본 실험에서 사용한 탠저블 블록은 3D 프린터로 출력된 정사각형 플라스틱 모형에 스티커 형태로 부착 가능한 ArUco 마커들로 구성된 모두 8종 명령어들이 사용되었다. 또한 명령어별로 최소 5개에서 8개까지 준비하였다. 실험에 사용한 PC 환경의 상세 하드웨어 내용은 [Table 1]에 제시되어 있다.

인식 소프트웨어는 유니티 환경에서 OpenCV for Unity 라이브러리에서 개발된 인식테스트 프로그램으로 PC에서 실행 가능하도록 빌드하였다. [Fig. 3]에서 제시하는 바와 같이 측정 전에 테스트하려는 탠저블 블록들을 인식 카메라의 수직 방향으로 배치하고 인식 중 현재 인식된 명령어들을 확인한 후 그 결과를 화면에 배치 순서대로 인식한 마커를 아이콘 형태로 표현한 결과를 보여주었다.

5.2 최적 측정 거리

탠저블 블록에 배치되는 배치 평면과 인식카메라의 거리는 배치판에 있는 모든 블록들이 카메라에 모두 보이는 거리어야 한다. 본 측정 실험에서는 배치 평면과 카메라간의 거리에 따라 변화하는 인식률을 알아보고자 한다.

측정을 위해 탠저블 블록들을 일렬로 가로방향으로 최대 10개까지 배치하도록 한 후 모든 탠저블 블록들이 화면 공간에 보이는 최소거리인 15cm를 기준으로 5cm 씩 배치 평면과 카메라 거리를 변화시켰을 때 인식정도를 평가하였다.

[Table 1] Testbed Hardware Specification

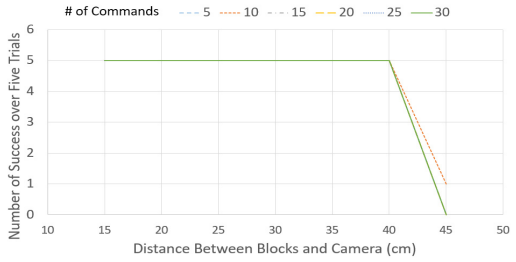
Operating System	Windows 10
CPU	Intel Core i7-7700
Memory	16GB
Camera	MicrosoftLifeCam HD-5000
Image Resolution	1280x720
Field Of View	66°

(a) Before (b) Recognizing (c) After

[Fig. 3] Screenshots of Marker Identification and Placement Recognition System

동일 배치에 대해 5회 반복 인식 평가를 시행하고 그 성공 횟수를 [Fig. 4]에서 제시하고 있다. 블록들의 개수를 5개에서 30개까지(최대 3줄까지) 변화시켰으며 인식 거리를 5cm 단위로 변화시켰을 때 인식 성공 회수를 보여주고 있다. 결과에서 보는 바와 같이 블록의 개수에 상관없이 배치된 모든 블록들을 순서대로 모두 정확하게 인식하였으나 40cm를 넘어가는 순간부터 블록 인식에 실패하였다. 이는 테스트에 사용한 카메라의 이미지 촬영 해상도 때문에 인식 가능한 최대 거리는 본 측정의 경우 40cm이하이다. 따라서 본 측정을 통해 탠저블 블록과 카메라간의 거리차이가 최대 40cm이며 40cm는 일반 모바일 앱 운용환경에서 코딩퍼즐 과제를 수행하기에 무리가 없는 수준의 거리라고 평가할 수 있다. 이후에 기술될 실험들은 이 실험

결과를 바탕으로 인식거리를 15cm부터 40cm로 제한하여 측정을 진행하였다.

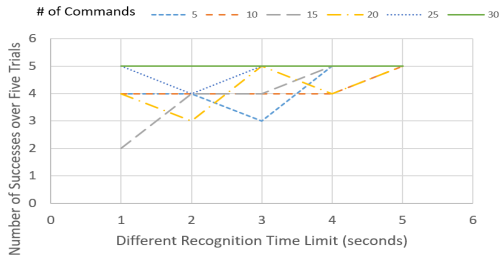


[Fig. 4] Number of Marker Recognition Successes over Different Distances between Command Blocks and Camera

5.3 인식시간

본 실험에서는 인식 시작 후 주어진 시간 안에 마커 배치를 5회 시도 중에 몇 회를 성공적으로 인식하였는가를 평가하였다.

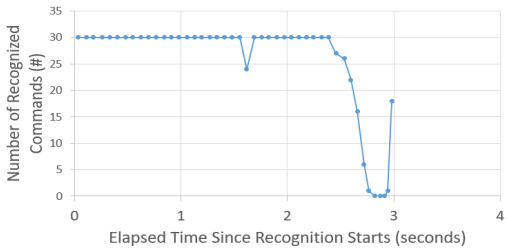
배치별 인식 거리 측정에서 사용한 동일한 명령어 배치에 대하여 1초, 2초, 3초, 4초, 5초를 인식 시간을 제한하고 이에 대한 배치 인식 정확도를 측정하였다.



[Fig. 5] Number of Recognition Successes over Five Trials for Different Recognition Time Limits

[Fig. 5]는 5개~30개의 탠저블 블록들에 대해 동일 실험을 5회 반복하였을 때 상이한 제한 시간 동안 성공적으로 배치 순서를 인식한 회수를 기록한 결과이다. 그림에서 보는 바와 같이 인식 시간을 늘릴수록 인식의 성공 횟수가 늘어나는 것을 확인할 수 있다. 본 연구의 제한 시간인 5초 이내에는 주어진 모든 명령어 배치에 대해 정확하게 인식하는 것을 확인할 수 있었다.

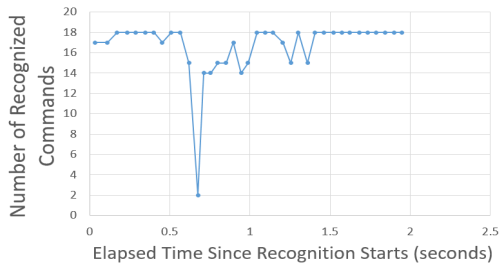
일반적으로 인식에 성공한 결과들은 [Fig. 6]에서 제시하는 바와 같은 패턴을 따르고 있다. [Fig. 7]은 3초 제한을 주었을 때 30개의 탠저블 블록을 인식시키는 경우 모든 30개의 명령어들을 0.05초 이내에 배치 순서대로 정확하게 인식하였다. 이후 2.5초 부분에서 카메라의 초점 조정으로 인하여 인식개수가 급격하게 변화함을 보여주고 있다.



[Fig. 6] A Typical Case of Recognition Success

[Fig. 7]은 2초간 20개의 블록을 인식시켰을 때 18개만 인식하고 2개의 인식에 실패하는 경우를 보여준다. 2개의 블록이 다른 블록들과 일부가 겹쳐 정확하게 인식되지 않는 현상을 보여주고 있다. 인식에 실패하는 주원인은 탠저블 블록과 블록사이에 마커 겹침 현상에 의하며 불확정적인 카메라 초점 조정 시간 때문에 인식 제한시간이 짧을 때에는 인식이 실패할 수 있다.

마커 겹침 현상을 해소하기 위해서는 인식되지 않는 명령어들과 인식되는 명령어들을 증강현실 공간상에서 구분하여 보여주는 시각적 사용자 피드백 시스템이 필요하다. 또한 너무 짧은 인식 제한시간은 카메라 초점 조정 문제로 인식에 실패할 수 있으므로 적정 수준(본 측정에서는 5초정도)의 인식 시간에 제한을 두는 것이 바람직할 것이다.



[Fig. 7] Recognition Failures of 20 blocks

5.4 배치 인식 정확도

앞장에서 기술한 측정결과를 바탕으로 최종 배치 인식 정확도를 다양한 배치에 대해 적용하고 이를 확인하기 위해 공인 시험 인증기관의 도움(인증번호 KSEL-PT-R-2019-011)을 받아 탠저블 블록의 개수를 최소 5개에서 최대 30개까지 임의 순서로 배치하였으며 인식 제한 시간을 5초로 설정하고, 인식거리는 15cm에서 40cm로 한정하여 100번 무작위로 명령어들을 배치하였으며 인식을 수행하였을 때 모든 배치를 성공적으로 인식하였다. 본 연구에서 개발한 배치 인식 알고리즘은 정확하게 모든 배치를 인식하였음을 확인하였다.

6. 배치 인식시스템 활용

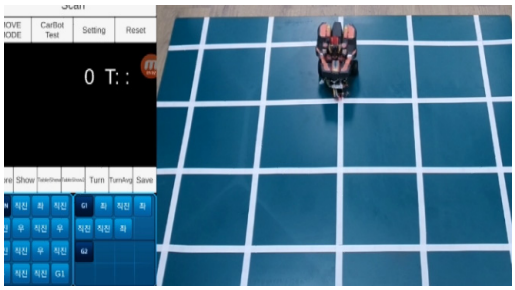
본 연구에서 개발한 마커기반 명령어 인식 시스템을 안드로이드형 모바일 환경으로 포팅하였다. 유니티로 개발된 인식시스템은 모바일 환경에 바로 적용 가능하였다.

적용된 모바일 앱은 기존 선행연구를 통해 개발된 모바일 앱으로 총 25개의 교육과제가 포함되어 있다[1]. 인식 시스템은 아이콘으로 표현된 명령어를 사용자가 직접 명령어 버튼 클릭을 통해 입력하는 입력시스템뿐만 아니라, AR 상황에서 명령어들을 입력할 수 있도록 명령어 입력창에 제시된 세 개의 함수 패널에서 임의의 위치를 클릭하면 바로 증강현실 카메라 장면으로 자동 전환될 수 있도록 했다. AR 카메라 장면에서는 화면에 보이는 명령어들을 실시간 인식하고 인식여부를 시각적 피드백시스템을 통해 보여줌으로써 인식한 명령어들을 모두 확인하도록 하였다. 모든 명령어가 인식됨을 시각적으로 확인한 사용자는 인식 종료를 요청하거나 5초 인식 시간이 경과되면 자동적으로 인식이 종료되며 퍼즐 과제 풀기 장면으로 전환된다. 전환 후에는 인식된 명령어들은 함수 패널에 명령어 입력 창에 아이콘 형태로 자동 입력된다. 모든 명령어 입력을 마친 후에 사용자는 명령어 실행 버튼을 누르면 주어진 퍼즐과제를 입력된 명령어에 따라 순차적으로 시행되어 퍼즐 풀기 여부를 확인할 수 있다. 잘못된 명령어 때문에 과제 완료에 실패하면 AR을 통한 명령어 인식을 반복하여 과제 수행을 완료하게 된다.

모바일 환경에서 성능 저하 현상을 최소화하기 위해 초당 인식 횟수를 줄이는 방식으로 모바일 앱의 성능 저하 현상을 문제를 해결하였다. 갤럭시 노트 3에서는 실제 인식 회수를 초당 10회에서 3회 수준으로 실시간성을 저하시켰음에도 주어진 한계 시간동안 25개 과제 수행시 모든 명령어 배치를 정확하게 인식하였다. [Fig. 8]에서는 인식시스템이 포팅된 모바일 앱의 실행 화면의 예를 제시하고 있다.

또한 본 연구에서 개발한 인식시스템을 [Fig. 9]에서 보는 바와 같이 기존 연구[2]에서 개발한 저비용 로봇을 구동하는 명령어 입력시스템으로 포팅하였다. 과제를 풀기 위해 실제 저비용으로 구현된 코딩퍼즐 명령어에 특화된 움직임을 제어하는 로직을 갖춘 실제 로봇을 구동하였다. AR 공간에서 탠저블 블록 배치들을 인식시켰으며 시작버튼을 누르면 로봇이 배치된 그리드 평면 공간상에서 입력된 명령어 순서에 따라 로봇이 구동되었다. 모바일 앱은 IoT 로봇에게 블루투스를 통해 인식된 명령어들을 전달한다. 점프 기능이나 액션 기능은 현재 로봇에 적용되지 않았으며 전진, 좌로 90도 회전, 우로 90도 회전, 함수 호출 기능 등만 동작하고 있다.

[Fig. 8] Sample Screenshots of Coding Puzzle Mobile App



[Fig. 9] AR-based IoT Robot Operation

7. 결 론

본 연구에서는 기존 연구에서 수행한 코딩퍼즐 시스템 및 명령어를 통한 움직임 제어가 가능한 저비용 로봇 시스템과 연동이 가능한 증강현실에서 ArUco 마커기반 탠저블 블록 배치 인식 알고리즘을 제시하였으며, 이 알고리즘의 인식률 측정을 통해 해당 알고리즘의 실제 환경에서 실행 가능함을 확인하였다. 또한 알고리즘이 탑재된 인식 시스템을 코딩퍼즐 과제 모바일앱과 IoT 구동 모바일앱으로 성공적으로 포팅하였다.

실험을 통해 8종의 명령어들을 최대 30개까지 동시에 인식 가능하다는 것을 확인하였으며, 인식하는데 걸리는 적정 한계 시간, 카메라와 탠저블 블록간의 적정 인식 거리 등을 측정하였다.

본 연구에서는 탠저블 블록을 인식시키기 위해 카메라를 수직방향으로 탠저블 블록을 인식할 때 가장 효과적으로 동작하지만, 향후에는 사용자가 모바일 앱을 사용하기 편한 형태로 구동하도록 카메라와 블록사이에 비스듬하게 배치하여도 정확하게 인식하도록 개선해야 할 것이다.

본 연구를 수행하는 중에 마커기반 명령어 인식은 명령어 디자인 도안을 제한시키는 걸림돌로 작용하기 때문에 마커기반이라 하더라도 매력적인 디자인 요소를 넣을 수 있도록 픽토그램 형태의 탠저블 블록을 인식할 수 있도록 개선되어야 할 것이다.

본 연구 결과는 기존에 소프트웨어 기술 중심으로 개발되어온 코딩퍼즐 플랫폼에 다양한 실제 공간과 인터렉션 모델을 제시하였으며, 제시된 플랫폼은 보다 다양한 환경에서 코딩교육을 수행할 수 있는 도구를 제공하고 있다. 본 연구에서 기반으로 하는 코딩퍼즐 플랫폼은 블록기반 시각적 프로그래밍 학습 교구로서 Lightbot[16]과 동일한 블록기반 명령어를 사용하고 있으나 이를 AR 환경에 적용한 상용화 사례는 없으며, 본 연구를 통해 개발된 AR 기반 시각적 프로그래밍 기반 교구가 학습자의 지속적인 학습효과를 유도하는데 기여할 것으로 기대한다.

ACKNOWLEDGMENTS

This work was supported by 2017 Hongik University Research Fund.

REFERENCES

[1] Beomjoo Seo and Sung Hyun Cho, “Design and Implementation of Students’ Coding Assessment System for a Coding Puzzle Game“, Journal of Korea Game Society, Journal of Korea Game Society, Vol.18, No.1, pp.7 - 18, 2018.

[2] Kyeonbok Park, Sung Hyun Cho, and Beomjoo Seo, “An Observation-based Movement Control for Educational Coding Robots”, Journal of Korea Game Society, Vol. 16, No.6, pp.131 - 142, 2016.

[3] Hanhoon Park and Jong-il Park, "Trend on Vision-Based Object Recognition and Tracking for Augmented Reality", Communications of the Korea Institute of Information Scientists and Engineers, Vol.34, No.12, pp.8-17, 2016.
[4] Hongfei Wu, Fengjing Shao, and Pencheng Sun, "Research of quickly identifying markers on Augmented Reality",
[5] Daniel Wagner and Dieter Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices", pp. Computer Vision Winter Workshop, 2007. (저가 mobile device에서 상용하기 위한 ARToolKitPlus)
[6] S. Garrido-Jurado, R. Munoz-Salinas, F.J.Madrid-Cevas, "Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion", Pattern Recognition, Vol. 47, No. 6, pp.2280-2292, 2014.
[7] Grishma Alshi, Mansi Dandiwala, Mikhail Cazi, and Renuka Pawar, "Interactive Augmented Reality-based System for Traditional Educational Media using Marker-derived Contextual Overlays", Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology, pp.930-935, 2018.
[8] Vuforia SDK, available at http://www.vuforia.com.
[9] Changmin Lim, Chanran Kim, Jong-Il Park, and Hanhoon Park, "Mobile Augmented Reality based on Invisible Marker", 2016 IEEE International Symp. on mixed and Augmented Reality Adjunct Proceedings, pp.78-81.
[10] Vincent Lepetit, Francesse Moreno-Noguer, and Pascal Fua, "EPnP: An accurate O(n) solution to the PnP problem," IJCV, Vol. 81, pp. 155-166, 2009.
[11] Byung-Kuk Seo, HanhoonPark, Jong-Il Park, Stefan Hinterstoisser, and Slobodan Ilic, "Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds", IEEE Transactions on Vision and Computer Graphics, Vol.20, pp.99-110, 2014.
[12] Alberto Crivellaro and Vincent Lepetit, "Robust 3D tracking with descriptor fields", 2014 IEEE Conference on Computer Vision and Pattern Recognition, 3514-3421, 2014.
[13] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, Vol. 60, pp. 91-110, 2004.
[14] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, "Speeded-Up Robust Features (SURF)", Computer Vision and Image Understanding, Vol.110, pp.346-359, 2008.
[15] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "ORB:An efficient alternative to SIFT or SURF", International Conference on Computer Vision, pp.2564-2571, 2011.
[16] https://lightbot.com

서 범 주 (Beomjoo Seo)

약 력 : 1996-2001 LG전자 DTV연구소 주임연구원
2009-2012 싱가폴국립대 Senior Research Fellow
2013-현재 홍익대학교 게임학부 조교수

관심분야 : 교육용 게임, 가상현실, 분산 멀티미디어 DBMS

조 성 현(Sung Hyun Cho)

약 력 : 1974-1978 서울대학교 계산통계학과 이학사
1978-1980 서울대학교 계산통계학과 이학석사
1989-1995 UCLA 컴퓨터과학과 이학박사
1996년-현재 홍익대학교 게임학부 교수

관심분야 : 게임 프로그래밍, 게임 인공지능