creative commons

COMMONS DEED

저작자표시 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000263743

2013年 2月

碩士學位論文

위치 추정을 위한 영상 특징점 정보 검출 방법

朝 鮮 大 學 校 大 學 院

制 御 計 測 工 學 科

羅 正 賢

위치 추정을 위한 영상 특징점 정보 검출 방법

A Detection Method of Visual Landmark Information for Mobile Robot Localization

2013년 2月 25日

朝 鮮 大 學 校 大 學 院

制 御 計 測 工 學 科

羅 正 賢

위치 추정을 위한 영상 특징점 정보 검출 방법

指導教授 高 樂 溶

이 論文을 碩士學位申請 論文으로 提出함.

2012年 10月

朝鮮大學校 大學院

制御計測工學科

羅 正 賢

羅正賢의 碩士學位論文을 認准함

委員長　朝鮮大學校　教授　　李鎭伊　印

委　員　朝鮮大學校　教授　　高樂溶　印

委　員　朝鮮大學校　教授　　趙昌鉉　印

2012年　11月

朝鮮大學校　大學院

목 차

제1장 서론 ····· 1
제1절 연구 배경 및 목적 ····· 1
제2절 연구 범위와 방향 ····· 3
제3절 논문의 구성 ····· 4

제2장 센서 정보의 중요성 및 기존 특징점 검출 방법 ····· 5
제1절 위치 추정에 대한 센서 정보 획득의 중요성 ····· 5
1. 이동 로봇의 모션 모델 ····· 5
가. 이동 로봇의 위치에 대한 불확실성 ····· 5
나. 불확실성 추가 소스 코드 ····· 6
다. 불확실성이 포함된 이동 로봇의 위치 예측 소스 코드 ····· 7
2. 시뮬레이션을 통한 이동 로봇의 위치 ····· 8
가. 노이즈 추가에 따른 이동 로봇의 위치 ····· 8
나. 정확한 센서 정보 획득 후 이동 로봇 위치 추정 ····· 8
다. 이동 로봇 위치 추정 비교 ····· 9
제2절 기존 영상을 이용한 특징점 검출 방법 ····· 11
1. SURF를 이용한 특징점 검출 ····· 11
2. 영상위치센서 ····· 12
3. 컬러표식기반 ····· 13
제3절 기존 센서를 이용한 특징점 검출 방법 ····· 15
1. GPS를 이용한 거리 측정 ····· 15
2. LRF를 이용한 특징점 검출 방법 ····· 16

제3장 제안된 특징점 검출 방법 ····· 17

제1절 키넥트(Kinect) 활용 ····· 17
1. 키넥트 소개 및 개발 환경 ····· 17
가. 키넥트 소개 ····· 17
나. 키넥트 구조 및 원리 ····· 18
다. 키넥트 SDK 사용 및 개발 환경 ····· 19
2. 키넥트의 기본 영상 및 거리정보 ····· 20
가. 키넥트 영상 정보 ····· 20
나. 키넥트 거리 정보 ····· 22
제 2절 영상 처리를 이용한 특징점 검출 방법 ····· 23
1. 이미지의 채널 변환 ····· 23
가. RGB와 Ycbcr ····· 23
나. 채널 변환 ····· 24
다. 영상 이진화 ····· 25
라. 채널 변환 및 이진화 소스코드 ····· 25
마. 특징점 레이블링(Labeling)방법 ····· 27
바. 컨벡스헐(Convex Hull) ····· 28
사. 컨벡스헐를 이용한 꼭지점 검출 ····· 28
아. 꼭지점 검출 소스 코드 ····· 29
제 3절 영상의 좌표 변환 ····· 31
1. 기존의 좌표 변환 방법 ····· 31
가. 핀홀 카메라(Pinhole Camera) 모델 ····· 31
나. 핀홀 카메라 투영 ····· 32
2. 제안한 좌표 변환 방법 ····· 33
가. 영상 좌표계와 월드 좌표계 ····· 33
나. 제안된 특징점의 정보 추출 방법 ····· 33
3. 특징점 구별 및 좌표 보정 ····· 35
가. 특징점의 크기 구별 방법 ····· 35
나. RGB영상과 Depth영상 좌표 보정 ····· 37

다. 키넥트 영상 보정 함수 ····· 38
라. 키넥트 영상 보정 방법 ····· 39

제4장 실험 및 결과 ····· 41
제 1절 키넥트를 이용한 특징점 검출 ····· 41
1. 특징점 검출 ····· 41
가. RGB채널 영상에서 특징점 검출 ····· 41
나. Ycbcr채널 영상에서 특징점 검출 ····· 41
2. 레이블링 검출 확인 ····· 42
가. Ycbcr 채널을 이용한 레이블링 검출 결과 ····· 42
나. 주변 환경의 변화에 따른 레이블링 검출 결과 ····· 43
다. 관심 영역 확인 ····· 44
3. 컨벡스헐을 이용한 특징점 좌표 추출 ····· 45
가. 컨벡스 헐 결과 확인 ····· 45
나. 특징점의 꼭지점 좌표 추출 ····· 46
제 2절 키넥트의 RGB영상과 Depth영상 보정 ····· 47
1. RGB영상과 Depth 영상의 오차 ····· 47
가. RGB영상과 Depth영상의 차이 확인 ····· 47
나. RGB영상과 Depth영상의 오차 ····· 48
다. RGB영상과 Depth영상 Calibration 결과 ····· 49
제 3절 특징점의 위치 좌표 추출 ····· 50
1. 고정된 위치에서 특징점 검출 및 실거리 확인 ····· 50
가. 실험환경 ····· 50
나. 영상의 좌표계 ····· 51
다. 검출된 특징점의 꼭지점 좌표 ····· 52
라. 거리와 각도의 변화에 따른 특징점 검출 및 정보 ····· 53
마. 거리 변화에 따른 측정된 꼭지점의 거리 그래프 ····· 56
2. 마우스 포인트를 이용한 특징점의 실거리 확인 ····· 57

가. 마우스 포인트를 이용한 꼭지점의 정보 ····· 57
나. 주변환경의 특징점 정보 ····· 58
3. 이동 로봇의 주행중 특징점 검출 ····· 60
가. 실험 환경 ····· 60
나. 거리에 따른 특징점 검출 정보 ····· 63

제5장 결론 및 향후 계획 ····· 68
참고문헌 ····· 69
후 기 ····· 72

표 목차

표 3.1 키넥트를 사용하기 위한 프로그램 ····· 19
표 3.2 키넥트 사양 ····· 20
표 3.3 Skeleton ID ····· 21
표 4.1 특징점 정보 ····· 61
표 4.2 이동 로봇 및 특징점 초기 위치 ····· 63

그림 목차

그림 2.1 가우신안 노이즈 생성 함수 ····· 7
그림 2.2 이동 로봇의 모션 모델 소스 코드 ····· 7
그림 2.3 이동 로봇의 위치 시뮬레이션 ····· 8
그림 2.4 위치 추정 시뮬레이션 ····· 9
그림 2.5 시뮬레이션을 이용한 이동 로봇 x좌표의 위치 ····· 10
그림 2.6 시뮬레이션을 이용한 이동 로봇 y좌표의 위치 ····· 10
그림 2.7 적분 영상 ····· 12
그림 2.8 검출된 특징점 영상 ····· 12
그림 2.9 인공표식에 의한 위치인식 예 ····· 13
그림 2.10 표식의 구조 ····· 14
그림 2.11 표식 모델 ····· 14
그림 2.12 양방향 거리 측정 ····· 15
그림 2.13 환경 모서리 점의 종류 ····· 16
그림 2.14 모서리 특징점 검출 단계 ····· 16
그림 3.1 키넥트 외형 ····· 17
그림 3.2 주요 영상센서 ····· 18
그림 3.3 키넥트 구조 ····· 18
그림 3.4 키넥트 원리 ····· 19
그림 3.5 키넥트 영상 ····· 20
그림 3.6 검출 가능한 사용자 관절 ····· 21
그림 3.7 키넥트 적외선 레이저 조사 ····· 22
그림 3.8 RGB 가산 혼합 원리 ····· 23
그림 3.9 채널 변경된 영상 ····· 24
그림 3.10 분리된 Ycbcr 채널 영상 ····· 24

그림 3.11 이진화 처리된 특징점 ····· 25
그림 3.12 특징점 검출 소스코드 ····· 26
그림 3.13 성분에 따른 번호 ····· 27
그림 3.14 레이블링 처리된 영상 ····· 27
그림 3.15 컨벡스헐의 입력과 출력 ····· 28
그림 3.16 컨벡스헐 알고리즘을 통한 꼭지점 검출 ····· 29
그림 3.17 레이블링과 컨벡스 헐 처리 소스코드 ····· 30
그림 3.18 핀홀 카메라의 영상 맺힘 ····· 31
그림 3.19 핀홀 카메라 모델 ····· 32
그림 3.20 영상 좌표계 ····· 33
그림 3.21 키넥트와 특징점의 각도 ····· 34
그림 3.22 특징점의 실제 거리를 구하기 위한 기하학적 모델 ····· 34
그림 3.23 다양한 방향에서의 특징점 ····· 36
그림 3.24 다양한 방향에서의 특징점의 크기 ····· 36
그림 3.25 화면상에 비춰진 특징점 및 실제 특징점의 모습 ····· 37
그림 3.26 키넥트 내부 센서 ····· 37
그림 3.27 RGB영상과 Depth영상 매칭 ····· 38
그림 3.28 RGB영상과 Depth영상 픽셀 좌표 보정 함수 1 ····· 38
그림 3.29 RGB영상과 Depth영상 픽셀 좌표 보정 함수 2 ····· 39
그림 3.30 RGB영상과 Depth영상의 매칭 소스 코드 ····· 39
그림 4.1 RGB 채널을 사용한 특징점 검출 ····· 41
그림 4.2 Ycbcr채널을 이용한 특징점 검출 ····· 42
그림 4.3 검출된 특징점의 레이블링 처리 ····· 42
그림 4.4 레이블링 처리된 특징점의 이진화 영상 ····· 43
그림 4.5 주변 노이즈에 따른 특징점의 레이블링 처리 결과 ····· 43
그림 4.6 외부 노이즈에 따른 검출된 특징점의 이진화 영상 ····· 44
그림 4.7 특징점 검출 환경의 노이즈 ····· 44
그림 4.8 관심영역 영상 ····· 45

그림 4.9 컨벡스헐 처리된 특징점 영상 ····· 46
그림 4.10 특징점의 꼭지점 ID 확인 ····· 46
그림 4.11 특징점의 꼭지점 좌표 확인 및 Depth영상 ····· 47
그림 4.12 RGB영상과 Depth영상의 오차 ····· 48
그림 4.13 RGB영상과 Depth 영상 매칭 ····· 48
그림 4.14 특징점의 꼭지점 좌표 오차 ····· 49
그림 4.15 RGB영상과 Depth 영상 보정 후 매칭 ····· 49
그림 4.16 특징점의 꼭지점 좌표 ····· 50
그림 4.17 사용된 특징점의 외형 ····· 51
그림 4.18 특징점 검출 환경 ····· 51
그림 4.19 화면 영상 좌표계 ····· 52
그림 4.20 검출된 특징점의 영상 ····· 52
그림 4.21 검출된 특징점의 정보 ····· 53
그림 4.22 거리에 따른 검출된 특징점 영상(1.5m) ····· 54
그림 4.23 거리에 따른 특징점 정보(1.5m) ····· 54
그림 4.24 거리에 따른 검출된 특징점 영상(3m) ····· 55
그림 4.25 거리에 따른 특징점 정보(3m) ····· 55
그림 4.26 다른 방향에서의 특징점 검출 정보1 ····· 56
그림 4.27 측정된 꼭지점 거리 그래프 ····· 57
그림 4.28 마우스 포인터를 이용한 특징점 정보 ····· 58
그림 4.29 마우스 포인터를 이용한 주변환경의 특징점 정보 ····· 59
그림 4.30 NRLAB-02 ····· 60
그림 4.31 이동 로봇에 설치된 키넥트 ····· 60
그림 4.32 특징점의 모양 ····· 61
그림 4.33 주행 환경 ····· 62
그림 4.34 주행 방향 및 특징점의 위치 ····· 62
그림 4.35 주행중 특징점 검출 ····· 63
그림 4.36 주행 영상 및 특징점 검출(원점) ····· 64

그림 4.37 측정된 특징점 정보(원점) ····· 64
그림 4.38 주행 영상 및 특징점 검출(3m) ····· 64
그림 4.39 측정된 특징점 정보(3m) ····· 65
그림 4.40 주행 영상 및 특징점 검출(6m) ····· 65
그림 4.41 측정된 특징점 정보(6m) ····· 65
그림 4.42 주행중 검출된 특징점의 거리(1~4) ····· 66
그림 4.41 주행중 검출된 특징점의 방향(1~4) ····· 67

ABSTRACT

A Detection Method of Visual Landmark Information for Mobile Robot Localization

Na Jung Hyun
Advisor : Prof. Ko, Nak Yong, Ph. D.
Dept. of Control and Instrumentation Eng.,
Graduate School of Chosun University

This paper investigates a method to find coordinate of a feature captured by the image sensing device Kinect provided by microsoft. It detects landmarks from the image and calculates their coordinates. From the color image and depth information by the Kinect, it extracts features from the background. It uses channel transformation, binary transformation, labeling, convex hull algorithm for the extraction. Once feature is extracted, it's coordinate is calculated. For these processes, it uses OpenCV library. To remove the discrepancy between the color image and depth image, coordinate adjustment is needed. The coordinate is represented with respect to the Kinect coordinate frame. The method is tested and verified indoors. The proposed method can be used for localization, mapping, and simultaneous localization and mapping(SLAM) which are indispensable for autonomous navigation of mobile robots.

제1장 서론

제1절 연구 배경 및 목적

로봇산업이 발달함에 따라 로봇은 제조, 생산성향상 등 노동력을 대체하는 도구로서의 역할에서 인간과 친화적인 지능형 서비스 로봇이 발전하고 있다. 이로 인해 응용분야가 확장되어 전 산업 분야로 확장되고 있다[1]. 그중 이동 로봇의 자율주행은 다양한 분야에서 응용되며 개발되고 있으며 많은 서비스를 제공하고 있다[1]. 그러나 이동 로봇이 미지의 환경에서 자율 주행을 하기 위해서는 환경에 따른 지도 정보와 이동 로봇의 위치 그리고 목표점의 위치 및 그 지점까지의 경로 등 다양한 정보와 알고리즘이 요구된다[2][3][4]. 또한 이동 로봇이 원하는 목표점에 도달하기 위해서는 특히 로봇의 위치가 정확하게 추정되어야 가능하다. 이동 로봇의 위치를 추정하기 위해서는 절대적이거나 상대적인 측정방식으로 접근한다. 이러한 측정된 정보를 이용하여 로봇의 위치를 추정해내야 한다[5]. 하지만 이러한 모든 정보는 센서와 로봇이 이동하는 노면 그리고 이동 로봇의 상태 등 모든 요소에 대한 불확실성이 포함되어 있어 정확한 이동 로봇의 위치를 추정하기가 불가능하다. 이러한 불확실성이 포함되어 있는 정보를 이용하여 정확한 이동 로봇의 위치를 추정하기 위해서는 파티클 필터(Particle filter)나 칼만 필터(Kalman filter)등 다양한 알고리즘을 필요하다. 또한 초음파 센서 및 레이저 센서 등을 이용하여 로봇을 중심으로 특징점의 거리 및 방향을 정확히 측정하기 위해서는 고가의 장비가 필요하다[6]. 또한 각각의 센서의 노이즈에 대한 불확실성이 포함되어 정확한 정보를 얻기 위해서는 다양한 필터링(Filtering)이 필요로 하며 구현 하는데 어려운 경우가 많이 있다. 이와 같이 이동 로봇의 위치에 대한 정보를 얻기 위한 센서가 필요하며 다양한 알고리즘과 센서를 이용하여 이동 로봇의 위치를 추정하기 위한 방법들이 많이 연구 및 개발이 되고 있다[7][8][9]. 그 중 영상을 이용한 기존 방식으로 두 개의 카메라를 사용하여 스테레오(Stereo) 방식를 이용한 영상의 거리 정보를 추출 하는 방법이 연구되고 있다[10][11][12]. 그러나 이러한 방법은 많은 연산 속도가 매우 늦고 스테레오 방식을 이용하여 영상의 특징점의 정보를 추출하는 구조가 매우 복잡하며 정확한 특징점의 정보를 획득하는데 많은 어려움이 존재한다.

본 논문에서는 이동 로봇의 위치를 추정하기 위한 센서 정보 획득 방법으로 초음파 센서나 레이저 센서를 대체하는 영상 정보를 이용한 특징점(LandMark) 검출 관한 방법 설명한다. 또한 다수의 카메라를 이용한 스테레오 방식이 아닌 한 개의 센서를 이용하여 영상과 거리 정보를 동시에 획득 가능한 키넥트 센서를 이용하였다. 키넥트 센서의 영상에서 검출된 특징점의 거리와 방향 그리고 크기를 계산 하여 이동 로봇의 위치 추정에 필요한 센서 정보로 활용하는 방법이다. 즉 임의의 환경에서 특징점이 주어져 있을 때 영상을 이용하여 특징점을 검출하고 이동 로봇을 기준으로 거리와 방향을 구하는 방법이다. 이는 다양한 영상처리 방법으로 특징점을 구별해내며 특징점의 크기와 모양을 구별하여 서로 다른 특징점의 정보를 획득할 수 있다. 이러한 특징점의 정보는 이동 로봇의 위치를 추정하기 위한 알고리즘에 적용 되며 센서융합기술의 초기 단계로 이동 로봇의 위치 추정에 대한 신뢰성을 높이는 목적을 갖고 있다. 그리고 본 논문에서 제안한 방법을 이용하여 레이저 센서 및 초음파 센서 등 고가의 장비를 사용하지 않고 영상을 이용하여 보다 신뢰성이 뛰어난 센서 정보를 측정하는데 목적을 두고 있다.

제2절 연구 범위와 방향

본 논문에서는 영상을 이용하여 특징점을 검출하고 검출된 특징점과 이동 로봇 간의 거리와 방향를 계산한다. 또한 다양한 정보를 얻기 위한 특징점의 구별하는 방법으로 특징점의 크기와 모양을 확인 가능하다. 이는 영상에서 여러 개의 특징점이 검출됐을 때 서로 다른 특징점의 정보를 구해낼 수 있으며 신뢰성이 매우 높은 정보를 획득할 수 있다. 방법은 크게 4가지 방법으로 구성된다.

첫째, 특정 색으로 구성된 다각형의 특징점을 검출하는 방법으로 RGB영상을 Ycbcr영상으로 변환한다. 그리고 Ycbcr 영상을 각각의 세 개의 채널로 분리하여 특징점의 색을 검출하는 각각 채널의 임계값을 찾아 이진화 처리를 한다.

둘째, 이진화된 특징점의 외각의 화면좌표를 찾는 방법으로 레이블링(Labeling) 처리를 한다. 이는 이진화되어 검출된 영역의 크기가 작은 영역은 잡음으로 처리한다. 또한 특징점의 외각의 좌표를 구해 관심영역(ROI)으로 지정하기 위함이다.

셋째, 관심영역으로 지정된 영상 안에서 컨벡스헐(Convex Hull)처리를 한다. 이는 실제 다각형으로 구성된 특징점의 꼭짓점을 검출하는 방법으로 화면상의 꼭짓점 좌표를 추출 해낼 수 있다. 또한 검출된 꼭짓점의 개수를 파악하여 특징점의 구분을 나눌 수 있게 된다. 그리고 RGB영상에서 검출된 특징점의 꼭짓점 좌표와 거리 정보를 나타내는 Depth 영상의 좌표를 매칭시키는 보정(Calibration)을 하여 꼭짓점의 좌표의 정확한 거리정보를 추출해낸다.

넷째, 추출된 꼭짓점은 화면성의 x,y 좌표로 구성된다. 그리고 카메라의 시야각(Field of View)과 삼각함수를 이용하여 가로(Horizontal)와 세로(Vertical) 의 한 픽셀 거리를 구한다. 이를 이용하여 화면 좌표계(Screen coordination system)에서 월드 좌표계(World coordination system)로 나타내고 이동 로봇과 특징점의 사이의 거리와 방향을 계산한다. 또한 일정한 크기의 특징점은 영상에서 비춰지는 거리와 각도에 따라 왜곡되는 현상을 보정함으로써 어떠한 각도에서도 실제 특징점의 크기를 구해낼 수 있다.

제3절 논문의 구성

2장에서는 기존에 연구되고 있는 영상을 이용한 특징점 검출 방법과 특정 센서를 이용한 방법을 설명한다. 3장에서는 특징검을 검출하기 위한 영상 센서 및 라이브러리를 소개하며 특징점의 검출 방법 및 꼭짓점 좌표 추출 그리고 화면 좌표계를 이용하여 실제 특징점의 위치 및 방향 그리고 크기를 측정하는 방법에 대해서 설명한다. 또한 4장에서는 본 논문에서 제안한 방법을 이용하여 가상의 환경에서 이동 로봇의 주행중 특징점의 검출 및 정보 추출에 대한 실험 결과를 서술한다. 마지막으로 5장에서는 본 논문의 결론을 맺는다.

제2장 센서 정보의 중요성 및 기존 특징점 검출 방법

제 1절 위치 추정에 대한 센서 정보 획득의 중요성

1. 이동 로봇의 모션 모델

가. 이동 로봇의 위치에 대한 불확실성

이동 로봇은 외부적인 요소와 내부적인 요소에 대해 에러 성분이 추가되어야 한다. 이는 타이어의 공기압, 휠의 크기, 지면의 재질 등 다양한 노이즈가 포함되기 때문에 실제 속도 명령이나 엔코더 데이터 등 이러한 정보를 이용하여 위치 추정은 불확실성이 크다. 그러므로 시뮬레이션을 구동할 때에는 이동 로봇의 속도 명령에 불확실성한 에러 성분을 추가되어 이동 로봇의 위치를 예측하여야 한다. 식 2.1은 이동 로봇의 불확실성이 포함된 직진 속도(Translational velocity)나타내는 $\hat{v}$와 회전 속도(Rotational velocity)나타내는 $\hat{w}$ 그리고 회전 방향 $\hat{\gamma}$를 구해내며 a_1~a_6은 이동 로봇의 불확실성을 나타내는 상수이다.

$sample()$ 함수는 속도 명령과 이동 로봇의 방향에 대한 불확실성 노이즈를 생성하는 함수 이다[13].

$$\begin{aligned} \hat{v} &= v + sample(\alpha_1|v| + \alpha_2|w|) \\ \hat{w} &= w + sample(\alpha_3|v| + \alpha_4|w|) \\ \hat{\gamma} &= sample(\alpha_5|v| + \alpha_6|w|) \end{aligned} \qquad \text{(식 2.1)}$$

식 2.1에서 불확실성이 추가된 속도 정보를 이용하여 이동 로봇 위치를 예측할 수 있다.

아래의 알고리즘은 이동 로봇의 모션 모델이다. 알고리즘에서 t는 시간을 나타내며 u_t는 t의 시점에서 이동 로봇의 속도 명령이며 x_{t-1}는 이전 시간 $t-1$일때 이동 로봇의 위치(x,y,θ)를 나타낸다. 또한 식 2.1과 같이 $\hat{v}$와 $\hat{w}$는 불확실성이 포

함된 이동 로봇의 직진속도와 회전 속도를 나타내며 $x^{'}, y^{'}, \theta^{'}$ 는 예측된 이동 로봇의 위치와 방향이다.

$$
\begin{array}{ll}
1: & Algorithm\, sample_motion_\mathrm{model}_velocity(u_t, x_{t-1}) \\
2: & \hat{v} = v + sample(\alpha_1|v| + \alpha_2|w|) \\
3: & \hat{w} = w + sample(\alpha_3|v| + \alpha_4|w|) \\
4: & \hat{\gamma} = sample(\alpha_5|v| + \alpha_6|w|) \\
5: & x^{'} = x - \frac{\hat{v}}{\hat{w}} - \sin\theta + \frac{\hat{v}}{\hat{w}} sin(\theta + \hat{w}\triangle t) \\
6: & y^{'} = y - \frac{\hat{v}}{\hat{w}} - \cos\theta + \frac{\hat{v}}{\hat{w}} cos(\theta + \hat{w}\triangle t) \\
7: & \theta^{'} = \theta + \hat{w}\triangle t + \hat{\gamma}\triangle t \\
8: & return \quad x_t = (x^{'}, y^{'}, \theta^{'})^T
\end{array}
$$

나. 불확실성 추가 소스 코드

가우시안 노이즈(Gaussian Noise)를 추가하는 알고리즘은 다음과 같이 표현되며 여기에서 b는 가우시안 노이즈를 발생시키는 표준편차이다.

$$
\begin{array}{ll}
1: & Alorithm\, sample_nomal_distribution(b): \\
2: & return \frac{1}{2}\sum_{i=1}^{12} rand(-b, b)
\end{array}
$$

그림 2.1은 가우시안 노이즈 발생시키는 알고리즘을 바탕으로 한 C++의 소스 코드이다. 여기서 range_max와 range_min은 표준편차의 크기를 나타내며 u는 가우시안 분포에서 발생된 노이즈 값이다[14].

```
float CRobotMotion::Sample(float b)
{
        srand((unsigned int)time(NULL));
        float range_max = b;
        float range_min = -b;
        float u = 0 ;
        for (int i = 0 ; i < 12 ; i++)
        {
                u += (float)rand() / (RAND_MAX + 1) * (range_max - range_min) + range_min;
        }
        u = u/2.0f;
        return u;
}
```

그림 2.1 가우시안 노이즈 생성 함수

다. 불확실성이 포함된 이동 로봇의 위치 예측 소스 코드

그림 2.2는 이동 로봇의 모션 모델 알고리즘을 바탕으로 한 C++의 소스 코드이다. 여기에서 불확실성이 포함된 이동 로봇의 위치를 예측할 수 있다. 또한 Sample 함수에 입력되는 a1~a6 값을 이용하여 불확실성의 크기를 조절할 수 있으며 값이 적을수록 이동 로봇의 위치에 대한 불확실성이 감소된다. 또한 실제 로봇 적용할 때에는 이동 로봇의 특성에 따라 a 값이 결정되므로 실험을 통해 값을 찾아내야 한다. m_DrTv와 m_DrRv는 이동 로봇 직진 속도와 회전 속도를 나타내는 변수이며 m_DrTheta는 이동 로봇의 방향을 나타낸다. 그리고 m_DrRobotPos.x와 m_DrRobotPos.y 그리고 m_DrRobotPos.r은 예측된 이동 로봇의 위치와 방향이다[15].

```
void CRobotMotion::DrawRobot_DR(void)
{

        m_DrTv = m_Tv + Sample((a1 * abs(m_Tv)) + (a2 *abs(m_Rv)));
        m_DrRv = m_Rv + Sample((a3 * abs(m_Tv)) + (a4 *abs(m_Rv)));
        m_DrTheta = Sample((a5 * abs(m_Tv)) + (a6 *abs(m_Rv)));


        m_DrRobotPos.x  = m_DrRobotPos.x  - ((m_DrTv / m_DrRv) * sin(m_DrRobotPos.r )) +
                ((m_DrTv / m_DrRv) * sin(m_DrRobotPos.r   + (m_DrRv * 0.1f)));
        m_DrRobotPos.y  = m_DrRobotPos.y  + ((m_DrTv / m_DrRv) * cos(m_DrRobotPos.r  )) -
                 ((m_DrTv / m_DrRv) * cos(m_DrRobotPos.r   + (m_DrRv * 0.1f)));
        m_DrRobotPos.r  = m_DrRobotPos.r  + (m_DrRv * 0.1f) + (m_DrTheta * 0.1f);

}
```

그림 2.2 이동 로봇의 모션 모델 소스 코드

2. 시뮬레이션을 통한 이동 로봇의 위치

가. 노이즈 추가에 따른 이동 로봇의 위치

그림 2.3은 시뮬레이션을 통한 노이즈가 추가된 이동 로봇의 주행 과 노이즈가 추가되지 않은 이동 로봇의 주행 결과이다. 그림에서 보이는 것과 같이 실제 일정한 속도 명령이 주어졌을 때 노이즈 추가 여부에 따라 이동 로봇의 위치는 다르게 측정되게 된다.

그림 2.3 이동 로봇의 위치 시뮬레이션

나. 정확한 센서 정보 획득 후 이동 로봇 위치 추정

그림 2.4는 특징점의 정확한 센서 정보를 획득한 뒤 이동 로봇의 위치를 추정한 모습이다. 그림에서 보이는 것과 같이 특징점의 거리 및 각도를 정확하게 측정

이 되면 노이즈가 포함 된 실제 이동 로봇의 위치를 추정할 수 있게 된다.

그림 2.4 이동 로봇의 위치 추정 시뮬레이션

다. 이동 로봇의 위치 추정 비교

그림 2.5와 2.6은 45초 동안 시뮬레이션을 이용한 이동 로봇의 위치를 추정한 x, y좌표에 대한 결과이다. 그림에서 보이는 것과 같이 속도 정보만을 이용한 이동 로봇의 데드레코닝(Dead reckoning) 위치와 실제 이동 로봇의 위치는 차이가 있다. 하지만 정확한 센서 정보를 획득하여 위치를 추정한 이동 로봇의 위치는 실제 위치를 잘 추정할 수 있다. 이와 같이 이동 로봇의 위치를 추정하는데 정확한 센서 정보를 획득 하는 것이 중요 하게 작용 된다. 따라서 정확한 센서 정보를 획득할수록 이동 로봇의 위치를 추정하는 것에 대한 신뢰성도 높아지게 된다.

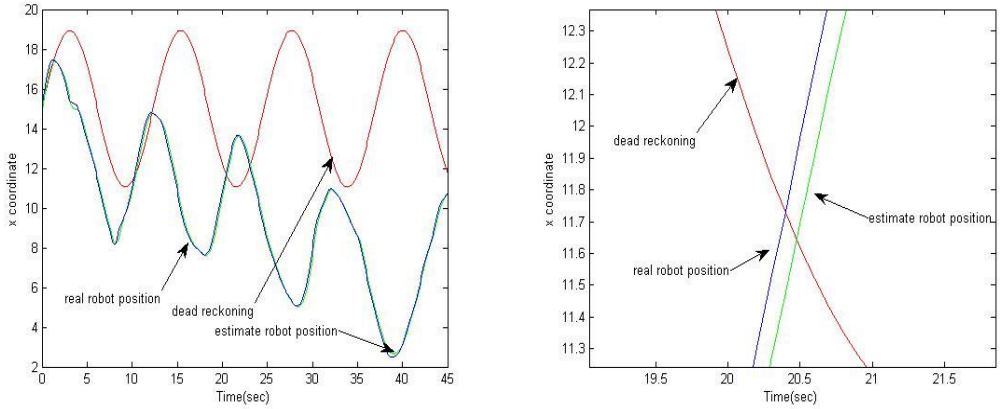


그림 2.5 시뮬레이션을 이용한 이동 로봇 x좌표의 위치

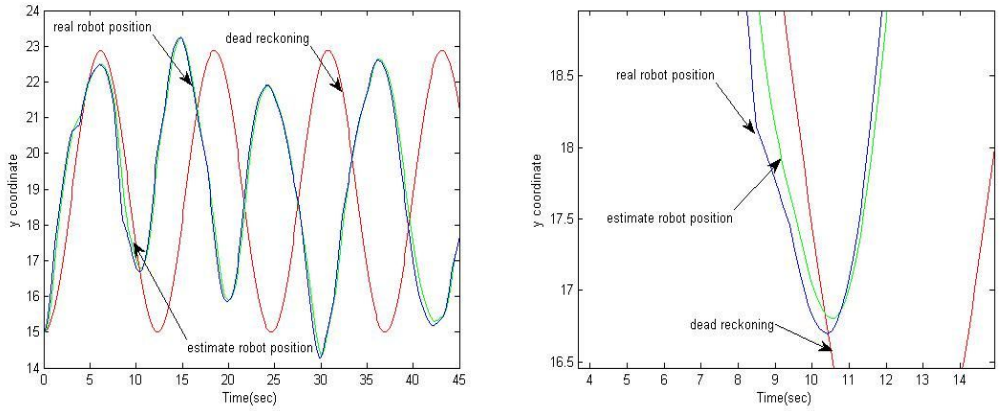


그림 2.6 시뮬레이션을 이용한 이동 로봇 y좌표의 위치

제2절 영상을 이용한 특징점 검출 방법

1. SURF를 이용한 특징점 검출

SURF(Speeded Up Robust Features)는 특징점을 검출하기 위해 사용되는 알고리즘으로 적분 영상과 근사화된 헤시안 행렬식의 컨볼루션을 통해 이루어진다. 적분 영상은 원점으로부터 각 픽셀의 위치까지 모든 영역의 픽셀 값을 더한 영상이며 식 2.2와 같이 정의된다. 여기서 x,y 위치에서 $I(x,y)$는 적분 영상 값이며 $X(i,j)$의 i,j는 원본 영상의 픽셀 값이다. 그리고 헤시안 행렬식은 식 2.3과 같이 정의되며 여기서 $L_{xx}(x,y,\sigma)$는 x,y 위치의 입력 영상과 σ를 갖는 가우시안 2차 미분 값인 $\partial^2 g(\sigma)/\partial x^2$ 과 컨벌루션 값을 나타낸다. 또한 $L_{xy}(x,y,\sigma)$, $L_{yy}(x,y,\sigma)$ 값은 xy 방향과 y 방향으로 가우시안 필터와 컨볼루션 값을 나타낸다.

$$I(x,y)=\sum_{i=0}^{i\le x}\sum_{j=0}^{i\le y}X(i,j) \qquad \text{(식 2.2)}$$

$$H(x,y,\sigma)=\begin{bmatrix}L_{xx}(x,y,\sigma)L_{xy}(x,y,\sigma)\\ L_{xy}(x,y,\sigma)L_{yy}(x,y,\sigma)\end{bmatrix} \qquad \text{(식 2.3)}$$

원본 영상과 적분 영상은 그림 2.7과 같이 나타낼 수 있으며 원본 영상의 사각형의 합을 구하기 위해서 12개의 화소 값을 더해야하지만 적분 영상의 경우 $A-B-C+D$인 4개의 화소 값만 계산할 수 있다[16][17].

	25	156	134	234	111	135
	245	32	45	85	200	58
	27	**46**	**16**	**124**	54	82
	45	**87**	**77**	**99**	62	45
	78	**187**	**177**	**199**	162	145
	45	34	12	8	35	9

0	0	0	0	0	0	0
0	25	181	315	549	660	795
D 0	270	458	637	**B 956**	1267	1460
0	297	531	726	1169	1534	1809
0	342	663	935	1477	1904	2224
C 0	420	928	1377	**A 2118**	2707	3172
0	465	1007	1468	2217	2841	3315

그림 2.7 적분 영상 (좌: 원본영상 우: 적분영상)

SURF를 이용한 특징점을 검출된 영상은 그림 2.8과 같다.

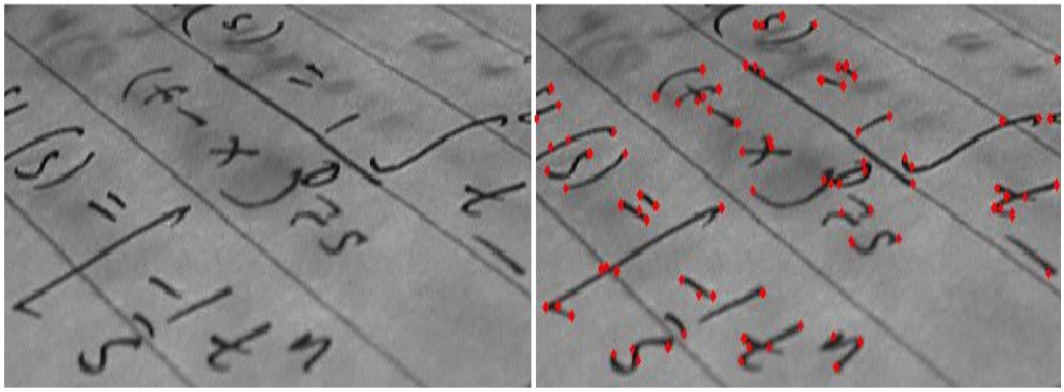
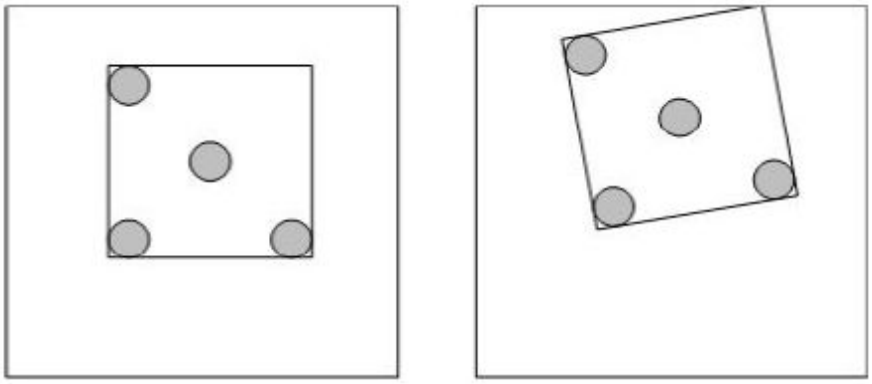


그림 2.8 검출된 특징점 영상 (좌: 원본영상 , 우: 검출 영상)

2. 영상위치센서

영상위치센서는 인공표식을 천장에 고정시키고 로봇에 부착된 카메라가 인공표식의 영상을 획득한다. 인공표식은 기준으로 한 로봇의 위치를 인식하는 센서이다. 인공표식은 여러 개의 마크로 조합하여 구성하며 각각의 아이디를 식별하기 위한 마크들이 포함된다. 이동하는 로봇의 위치에 따라 카메라좌표계에서 마크의 위치가 다르므로 로봇의 현재 위치를 구할 수 있다. 또한 영상위치센서는 인공표식의 위치를 미리 아는 경우 로봇의 전역위치 및 방향을 구할 수 있다.

그림 2.9는 상용화된 영상위치센서에 사용되는 인공표식의 예이다. 적외선 마크와 적외선 카메라를 사용하는 경우 조명의 영향을 받지 않아 위치를 인식할 수 있다[18].



(a) $x_c = 0, y_c = 0, \theta_c = 0$ (b) $x_c = 50, y_c = 0, \theta_c = 10$

그림 2.9 인공표식에 의한 위치인식 예

3. 컬러표식기반

실내 환경에서 시각적 표식을 이용하여 자기 위치추정을 수행할 경우 로봇의 위치를 결정하기 위해서는 표식 발견, 표식 확인, 자기 위치 추정의 세 단계의 과정이 필요하다. 여기서 자기 위치 추정은 다양한 주위환경의 특징적 정보를 이용하여 기존에 작성된 기하학적 혹은 위상학적 지도 정보와의 비교를 통해 지도상에서의 자신의 위치를 인식하는 것이다. 표식을 이용하여 자기 위치 추정을 수행하는 경우 일반적으로 2차원 위치 정보만으로도 충분하므로 표식의 수직 위치가 로봇 시야의 수준에 맞추어져 있다고 가정 하에 로봇의 지도상 위치와 방향만을 추정한다.

조명변화에 의한 밝기 값 변화를 극복하기 위해서 색도 컬러 공간(Chromaticity color space)을 이용하며 색도 변환을 통해 효과적으로 밝기 값 성분을 제거 하여 어두운 조명에서 동작할 수 있다. RGB 입력 영상으로부터 색도 컬러 공간으로 변환은 식 2.4로 이루어진다.

$$[r \;\; g] = \left[\frac{R}{R+G+B} \;\; \frac{G}{R+G+B} \right] \qquad \text{(식 2.4)}$$

표식의 형태와 구조는 그림 2.10과 2.11로 구성되며 컬러 모델을 이용하여 만들어진 컬러 패턴은 그림 2.10과 같이 수직으로 이루는 뒷 평판에 대하여 45도를 이루도록 위치되어 있으며 뒷 평판이 서로 수직으로 교차하는 부분에 검은 선으로 코너의 위치를 표시한다[19].

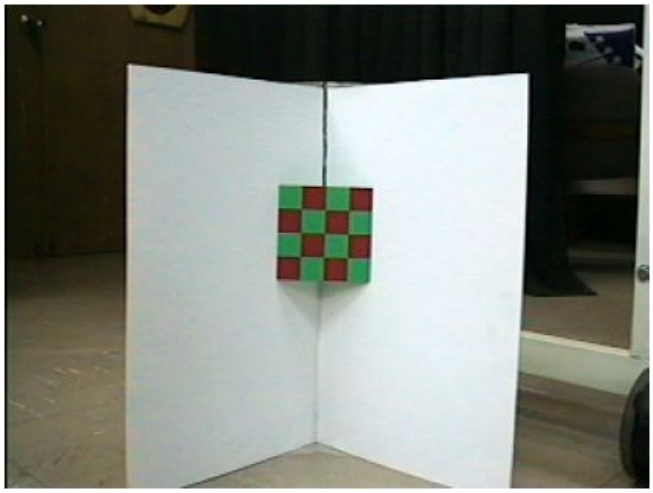


그림 2.10 표식의 구조

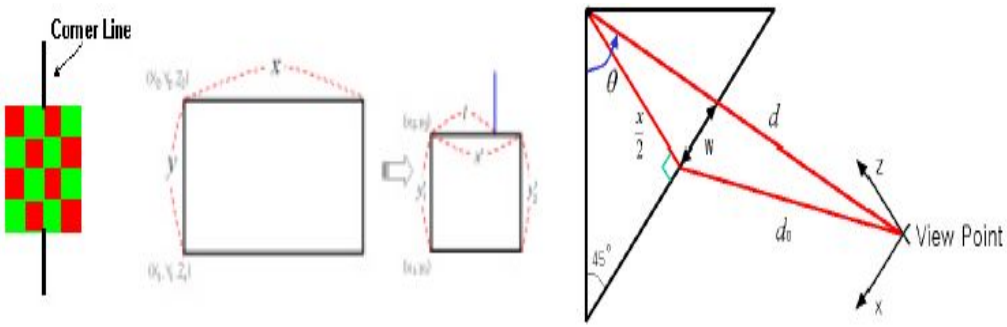


그림 2.11 표식 모델

제3절 센서를 이용한 특징점 검출 방법

1. GPS 이용한 거리 측정

CCS(Chirp spread spectrum)는 무선 주파수 신호를 전달하는 방법 중 하나로 지그비(Zigbee)에 비해 신호의 전달 특성이 좋고 근거리에 정확한 위치를 추적 하는데 이용된다. CSS는 식 2.5과 같이 표현 된다. 여기서 w_s 는 초기 주파수, w_{BW} 는 대역폭을 나타내고 T_c 와 u 는 계단 함수를 나타낸다[20].

$$S_c(t) = \exp\left[j(\frac{w_z + w_{BW}}{2T_c}t)t + \theta_t\right] \times \left[u(t) - (t - T_c)\right] \qquad \text{(식 2.5)}$$

그림 2.12과 같이 측정하고자 하는 두 CCS 노드가 각각 신호를 보낸 후 응답 신호를 보낼 때 소요되는 시간을 측정하는 TWR(Tow Way Ranging)을 서로 대칭이 되도록 수행한다. TWR에 의해 얻어진 시간을 이용하여 최종적으로 두 노드간의 신호 전달 시간은 식 2.6과 같이 계산된다. 여기서 T_d 는 두 노드 간 전달 시간이며 T_1, T_2 는 각각 노드에서 측정한 신호의 시간이다. 그리고 T_{pA}, T_{pB} 는 A, B 에서 응답 확인 신호를 보낼 때까지의 소요 시간이다[20].

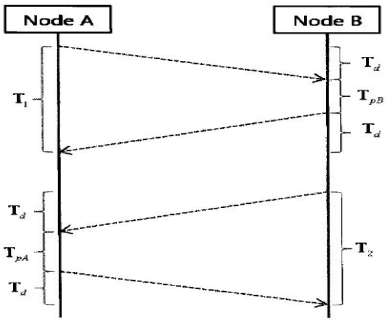


그림 2.12 양방향 거리 측정

$$T_d = \frac{1}{4}(T_1 - T_{pB} + T_2 - T_{pA}) \qquad \text{(식 2.6)}$$

2. LRF를 이용한 특징점 검출 방법

주변 환경의 특징점을 관찰하기 위한 센서의 하나로서 LRF(Laser range finder)를 이용한다. 주변 환경의 특징점으로 모서리 점은 외란에 대해 강인하게 활용 될 수 있으며 복잡한 환경을 비교적 간략화 할 수 있다는 장점이 있다.

실내 환경의 모서리 점에는 크게 두 가지 종류로 나누어진다. 로봇이 모서리 점을 관찰할 때 모서리 점은 LRF로부터 거리정보 값의 지역적 최대점 혹은 최소점에 위치하게 된다. 이러한 최대 최소점은 주어진 조건에 의해 모서리 점을 검출하게 된다. 그림 2.13은 주어진 조건에 의해 모서리 점을 정의된 것을 도식적으로 나타낸 그림이며 그림 2.14와 같은 단계로 구성된다[21].

그림 2.13 환경 모서리 점의 종류

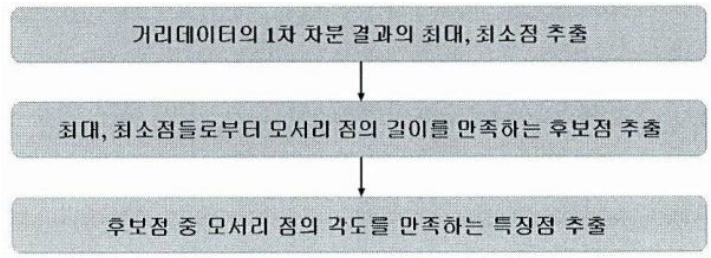


그림 2.14 모서리 특징점 검출 단계

3장 제안한 특징점 검출 방법

제1절 키넥트(Kinect) 활용

1. 키넥트 소개 및 개발 환경

가. 키넥트 소개

2010년 11월 19일 마이크로소프트(Microsoft)에서 출시된 키넥트는 별도의 컨트롤러(Controller)를 이용하지 않고 사용자의 신체를 이용하여 게임과 엔터테인먼트를 경험할수 있는 주변기기이다. 또한 카메라 모듈이 장착되어 모션 캡처로 사용자의 신체를 인식할 뿐만 아니라 마이크 모듈로 음석을 인식할 수도 있다.

또한 개발자들이 키넥트를 이용하여 연구 및 개발할 수 있도록 키넥트 SDK(Microsoft Kinect SDK for Windows)를 제공하고 있다. 그 외형은 그림 3.1과 같다[22].

그림 3.1 키넥트 외형

나. 키넥트 구조 및 원리

키넥트는 그림 3.2와 같이 2개의 깊이 센서(Depth Sensor)와 1개의 색상 카메라(RGB Camera)로 구성된 3개의 렌즈와 4개의 마이크로폰(Microphone array)으로 구성 되어 있다. 그리고 키넥트의 주 기능인 사람을 검출 하거나 거리정보를 획득하기 위한 센서들이다. 또한 깊이 센서는 단색 CMOS센서와 적외선 레이저 프로젝터(IR laser projector)로 구성되어 있다. 그림 3.3과 그림 3.4는 키넥트의 구조와 원리를 나타내는 모습이다[23][24].

그림 3.2 주요 영상센서 (좌 : RGB Carmera, 중: IR Camera 우: IR laser projector)

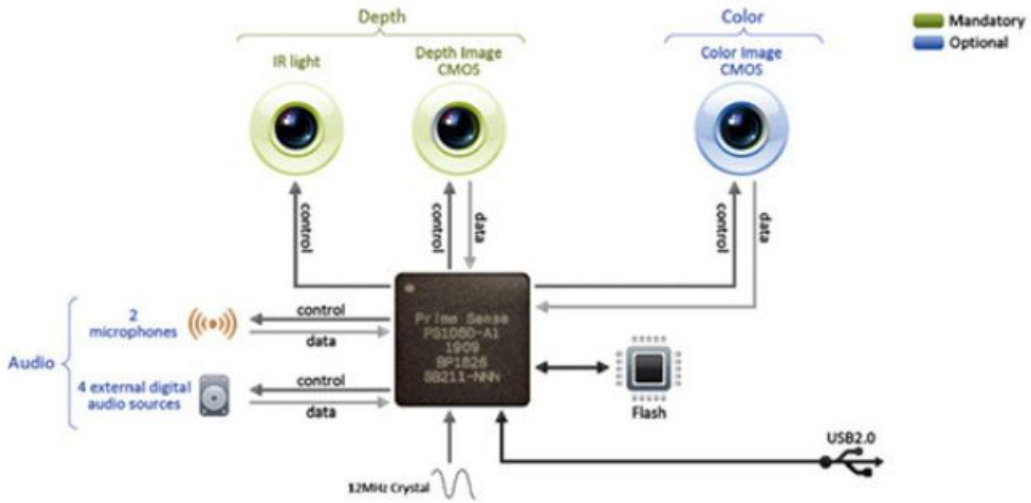


그림 3.3 키넥트 구조

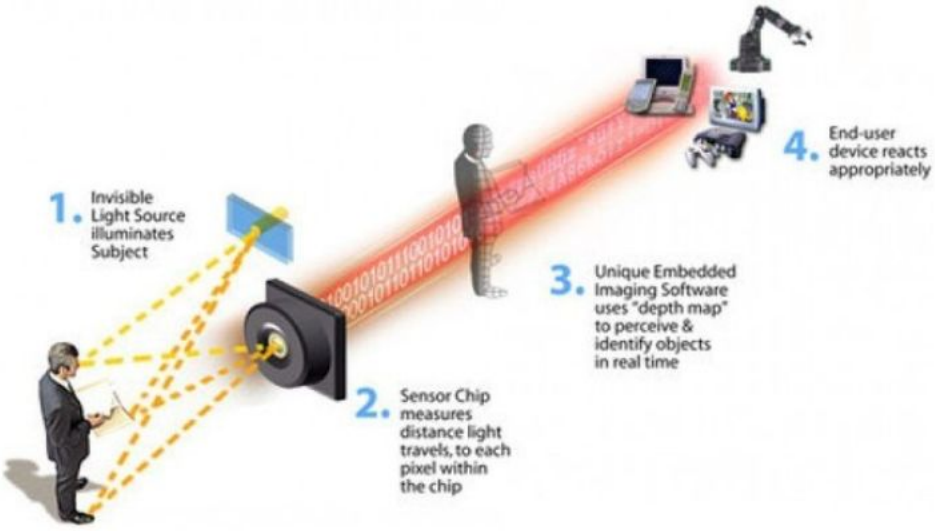


그림 3.4 키넥트 원리

다. 키넥트 SDK 사용 및 개발 환경

키넥트를 사용하여 개발 및 연구를 하기 위해서 해당 디바이스 드라이버와 키넥트 SDK를 설치해야 한다. 그리고 키넥트 SDK 설치과정 중 키넥트 센서에 대한 디바이스 드라이버는 자동으로 설치된다. 표 3.1과 표 3.2는 키넥트를 사용하기 위한 프로그램 및 키넥트 사양(Specifications)이다[25].

표 3.1 키넥트를 사용하기 위한 프로그램

Related SDKs	Relate Tools	Related Downloads	Previous Versions
Microsoft DirectX SDK	Microsoft Visual Studio Express 2012 Express	Kinect for Windows Runtime v1.6	Kinect for Windows SDK 1.5
Microsoft Speech Platform SDK v11	XNA Game Studio 4.0	.NET Framework 4.5	Kinect for Windows SDK 1.0
		.NET Framework 4.0	Kinect for Windows SDK beta
		XNA Framework Redistributable 4.0	Kinect for Windows Runtime v1.5

표 3.2 키넥트 사양

Kinect	Array Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	± 27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC)
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

2. 키넥트의 기본 영상 및 거리정보

가. 키넥트 영상 정보

키넥트는 그림 3.5와 같이 일반적인 RGB 카메라로 촬영되는 영상(Color view)과 촬영된 영상의 깊이 정보를 나타내는 영상(Depth view) 그리고 검출 된 사용자의 골격을 나타내는 영상(Skeleton view)을 제공해준다. 그림과 같이 컬러 영상에 사용자가 촬영되면 Depth 영상에서 사용자에 해당 영역은 특정한 색으로 나타나진다. 또한 사용자의 20개에 해당하는 관절에 대한 좌표(Coordinate)를 획득할 수 있다. 그림 3.6은 키넥트에서 지원해주는 사용자의 관절을 나타낸 것이다.

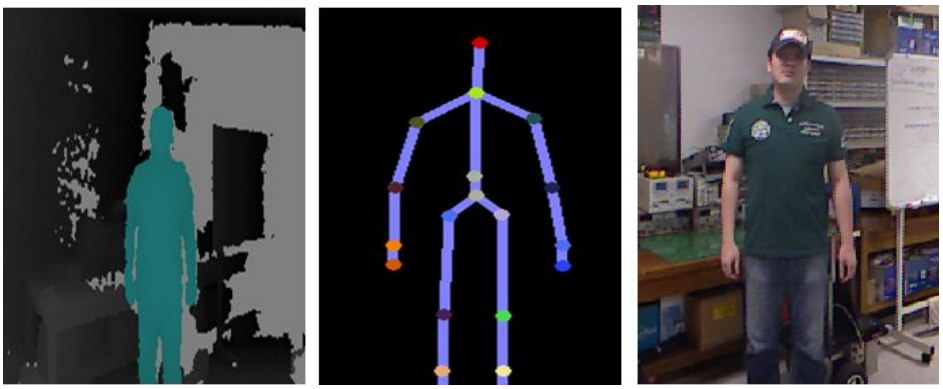


그림 3.5 키넥트 영상

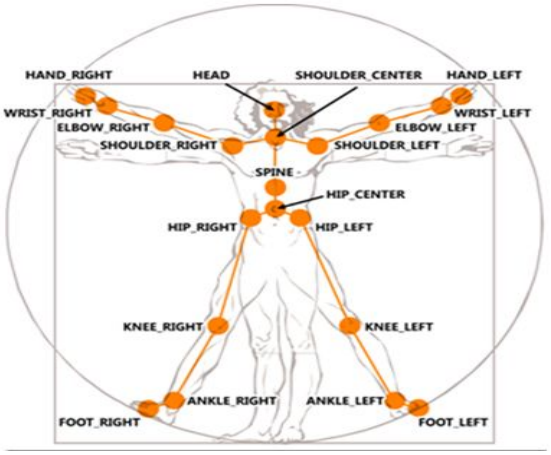


그림 3.6 검출 가능한 사용자 관절

위의 20개 의 관절에 대한 정보는 키넥트 SDK의 구조체 형태로 구성되어 있으며 해당 관절에 대한 ID는 표 3.3과 같다.

표 3.3 Skeleton ID

Index	Constant	Description
0	NUI_SKELETON_POSITION_HIP_CENTER	Center, between hips.
1	NUI_SKELETON_POSITION_SPINE	Spine.
2	NUI_SKELETON_POSITION_SHOULDER_CENTER	Center, between shoulders.
3	NUI_SKELETON_POSITION_HEAD	Head.
4	NUI_SKELETON_POSITION_SHOULDER_LEFT	Left shoulder.
5	NUI_SKELETON_POSITION_ELBOW_LEFT	Left elbow.
6	NUI_SKELETON_POSITION_WRIST_LEFT	Left wrist.
7	NUI_SKELETON_POSITION_HAND_LEFT	Left hand.
8	NUI_SKELETON_POSITION_SHOULDER_RIGHT	Right shoulder.
9	NUI_SKELETON_POSITION_ELBOW_RIGHT	Right elbow.
10	NUI_SKELETON_POSITION_WRIST_RIGHT	Right wrist.
11	NUI_SKELETON_POSITION_HAND_RIGHT	Right hand.
12	NUI_SKELETON_POSITION_HIP_LEFT	Left hip.
13	NUI_SKELETON_POSITION_KNEE_LEFT	Left Knee.
14	NUI_SKELETON_POSITION_ANKLE_LEFT	Left ankle.
15	NUI_SKELETON_POSITION_FOOT_LEFT	Left foot.

16	NUI_SKELETON_POSITION_HIP_RIGHT	Right hip.
17	NUI_SKELETON_POSITION_KNEE_RIGHT	Right knee.
18	NUI_SKELETON_POSITION_ANKLE_RIGHT	Right ankle.
19	NUI_SKELETON_POSITION_FOOT_RIGHT	Right foot.
	NUI_SKELETON_POSITION_COUNT	Used as an index to terminate a loop. Not used as a position index.

나. 키넥트 거리정보

적외선 센서에서 송출된 많은 점들은 물체에 반사되어 다른 CMOS 카메라가 인식하고 거리를 인식해서 계산한다. 그리고 거리에 대한 이미지를 생성하게 되며 이는 320x240의 화소를 갖는다.

키넥트의 거리 정보는 적외선 센서에서 적외선을 방출하면 그림 3.7와 같이 적외선 카메라를 이용하여 관찰하면 무수히 많은 점을 관찰할 수 있다[27].

그림 3.7 키넥트 적외선 레이저 조사

제 2절 영상 처리를 이용한 특징 점 검출 방법

1. 이미지의 채널 변환

가. RGB 와 Ycbcr

RGB(Red, Green, Blue)는 빛의 삼원색을 이용하여 색을 표현하는 방식이다. RGB 색 공간은 색을 혼합하면 명도가 올라가는 가산 혼합 방식으로 색을 표현한다. 그림 3.8은 RGB의 가산 혼합의 원리를 나타낸다.

Ycbcr은 영상 시스템에서 사용되는 색공간의 일종으로 Y는 휘도 성분을 나타내며 Cb와 Cr은 색차 성분이다. 또한 Ycbcr은 절대 색공간이 아니며 RGB정보를 인코딩 하는 방식으로 RGB 정보에 의존한다. 또한 Ycbcr은 감마 보정된 RGB 원본 영상에서 식 3.1 과 같이 Kb 와 Kr 의 두 가지의 상수를 이용하여 얻을 수 있다 여기서 R, G, B 는 0과 1사이의 값을 갖고 0은 최소값 1은 최대값을 나타낸다[28].

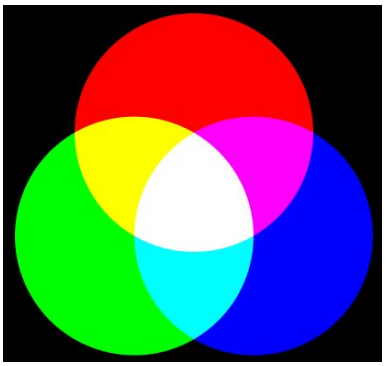


그림 3.8 RGB 가산 혼합 원리

$$
\begin{aligned}
Y' &= Kr \cdot R' + (1 - Kr - Kb) \cdot G' + Kb \cdot B' \\
Cb &= 0.5 \cdot (B' - Y')/(1 - Kb) \\
Cr &= 0.5 \cdot (R' - Y')/(1 - Kr)
\end{aligned}
\qquad \text{(식 3.1)}
$$

나. 채널 변환

키넥트에서 촬영된 일반적인 컬러 영상은 640(Horizontal)x480(Vertical)의 화소를 갖는 영상으로 RGB의 3개의 채널을 보유하고 있다. 이를 특징점의 색을 정확히 검출하기 위해 RGB의 채널을 Ycbcr 채널로 변경해준다. 이 방법으로 OpenCV 라이브러리를 이용하였다. 사용된 함수로는 cvCvtColor()를 사용하였으며 RGB영상을 입력인자로 하며 출력 코드 CV_BGR2YCrCb를 이용하여 Ycbcr영상을 출력하였다. 그림 3.9는 키넥트에 검출된 특징점의 RGB 채널의 영상을 Ycbcr 채널의 영상으로 변환된 모습이다.

그림 3.9 채널 변경된 영상 (좌: RGB 채널, 우: Ycbcr 채널)

그림 3.10은 Ycbcr의 채널 변환한 영상을 Y, Cb, Cr 의 각각의 채널로 분리 한 모습이다. 이는 cvCvtPixToPlane() 함수를 이용하여 각각의 채널에 해당하는 이미지를 분리시킬 수 있다.

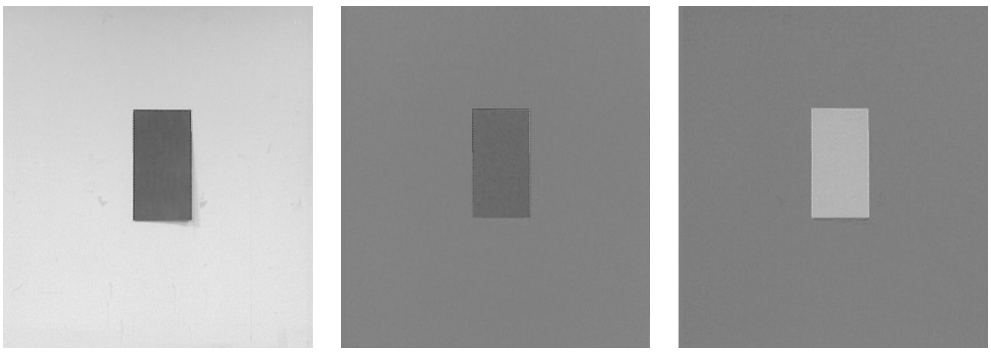


그림 3.10 분리된 Ycbcr 채널 영상 (좌: Y, 중: Cb, 우: Cr)

다. 영상 이진화(Image Binarization)

이진화란 영상의 픽셀의 값을 0과 1의 값으로 갖는 방법이다. 이는 특징점을 검출하여 다양한 정보를 추출하기 위한 필수적인 전처리 단계에 해당한다.

Ycrcb채널로 변환된 특징점의 영상은 Y, Cr, Cb의 휘도와 색차의 임계값을 조절하여 Ycrcb 채널의 영상에서 특징점의 색만 추출해야 한다. 그리고 채널 변환이 되기 전의 RGB 영상을 별도로 복사한 영상에 검출된 특징점의 픽셀 좌표의 값을 R,G,B 채널 각각에 255의 값으로 입력한다. 또한 특징점을 제외한 배경은 검은색으로, 각각의 채널의 0으로 입력한다. 그리고 3채널의 영상으로 구성된 이진화된 영상을 1채널 이진화 영상으로 변환 하는 과정이 필요로 한다. 이는 마찬가지로 Opencv라이브러리를 이용하여 1채널의 영상으로 변환하였다. 그림 3.11은 검출된 특징점을 이진화 처리한 영상이다.

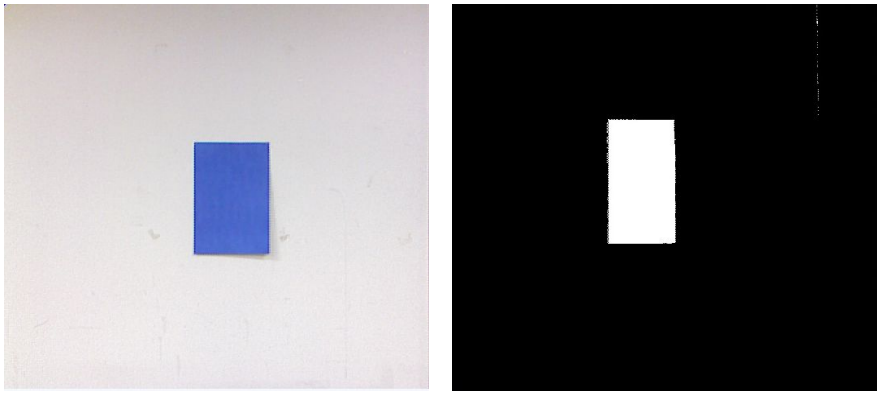


그림 3.11 이진화 처리된 특징점 (좌: 원본 영상, 우: 이진화된 영상)

라. 채널 변환 및 이진화 소스코드

그림 3.12은 영상의 채널 변환 및 이진화 처리하여 특징점을 검출하는 소스 코드이다. 소스 코드에서 보이는 것과 같이 cvCvtColor 함수를 이용하여 RGB채널에서 YCbCr채널로 변경하고 이를 cvCvtPixToPlan 함수를 이용하여 각각 채널로 이미지를 분리시킨다. 또한 분리된 각각의 채널의 이미지의 픽셀 값과 사용자가 검출하고자하는 특징점의 픽셀 값을 비교하여 이진화 처리를 한다.

아래의 소스코드에서 m_DownImage는 원본 영상이며 m_YcbcrImage는 채널변환된 영상이다. 또한 m_YcbcrImage->height와 m_YcbcrImage->width는 영상의 크기를 나타내며 Color_y, Color_cb, Color_cr은 채널변환된 영상의 각각 채널의 픽셀 값들을 나타낸다. 그리고 m_CbMin과 m_CbMax 그리고 m_CrMin과 m_CbMax는 검출하고자 하는 특징점의 색의 찾는 픽셀의 임계값이다.

```
cvCvtColor(m_DownImage,m_YcbcrImage,CV_BGR2YCrCb);
cvCvtPixToPlane( m_YcbcrImage, m_yPlanImage, m_crPlanImage, m_cbPlanImage, 0);

for( int y = 0 ; y < m_YcbcrImage->height ; y++ )
{
        for( int x = 0 ; x < m_YcbcrImage->width ; x++)
        {
                Color_y = m_yPlanImage->imageData[y * m_yPlanImage->widthStep + x];
                Color_cb = m_cbPlanImage->imageData[y * m_cbPlanImage->widthStep + x];
                Color_cr = m_crPlanImage->imageData[y * m_crPlanImage->widthStep + x];

                if((m_CbMin <= Color_cb) && (Color_cb <= m_CbMax))
                {
                        if((m_CrMin <= Color_cr) && (Color_cr <=m_CrMax))
                        {
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3)] = (unsigned char)255;
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+1)] = (unsigned char)255;
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+2)] = (unsigned char)255;
                        }
                        else
                        {
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3)] = (unsigned char)0;
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+1)] = (unsigned char)0;
                                m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+2)] = (unsigned char)0;
                        }
                }
                else
                {
                        m_DownImage->imageData[y*m_DownImage->widthStep + (x*3)] = (unsigned char)0;
                        m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+1)] = (unsigned char)0;
                        m_DownImage->imageData[y*m_DownImage->widthStep + (x*3+2)] = (unsigned char)0;
                }


        }
}
```

그림 3.12 특징점 검출 소스코드

특징점을 검출하는 알고리즘은 다음과 같다.

$$
\begin{aligned}
&YcbcrImage = cvCvtColor(RGB\ Image)\\
&Ycbcr\ Channel\ Image = cvCvtPixToPlane(Ycbcr\ Image)\\
&\text{for } Verical\ size\ of\ Image\\
&\quad \text{for } Horizontal\ size\ of\ Image\\
&\qquad Cb = Cb\ Channel\ Image\ pixel(x,y)\\
&\qquad Cr = Cr\ Channel\ Image\ pixel(x,y)\\
&\quad \text{if } Cb, Cr \in Landmark\ color\\
&\qquad RGB\ Image = 255\\
&\quad else\\
&\qquad RGB\ image = 0\\
&\quad endfor\\
&endfor
\end{aligned}
$$

마. 특징점 레이블링(Labeling)방법

레이블링이란 인접한 화소에 모두 같은 번호(Label)를 붙이고 연결되지 않는 다른 성분에는 다른 번호를 붙이는 일이다. 전처리된 이진화 영상에서 특징점의 외각 영역을 구분하는 단계이다. 이것은 검출된 특징점 영역인 흰색과 그 외의 배경 즉 특징점이 아닌 검은색의 영역은 서로 다른 성분으로 번호가 그림 3.13과 같은 형식으로 구분된다. 이는 특징점을 검출하기 위해 필요하지 않는 영역과 잡음 제거를 위한 이진화 처리와 같이 필수적인 전처리 단계에 속한다. 그림 3.14는 이러한 작업으로 검출된 특징점은 좌측 상단을 원점으로 우측 하단으로 사용자가 지정해 놓은 임의의 색으로 구성된 다각형의 외각을 사각형으로 구분 지을 수 있다[29][30].

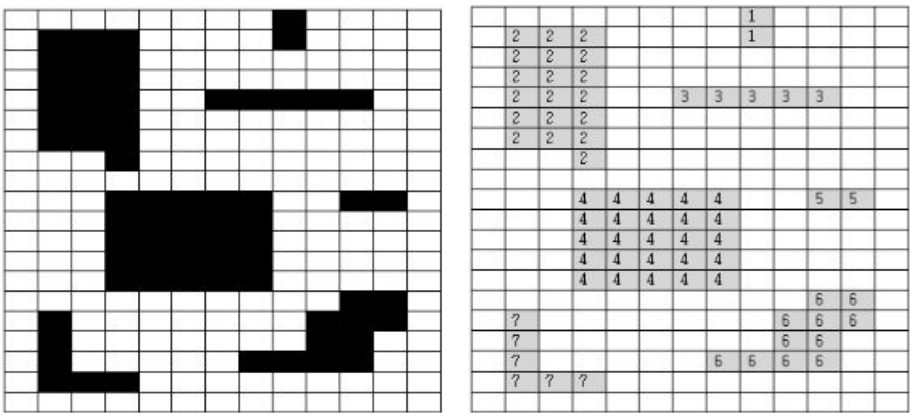


그림 3.13 성분에 따른 번호

그림 3.14 레이블링 처리된 영상 (좌: 원본 영상, 우: 레이블링 처리된 영상)

바. 컨벡스 헐(Convex Hull)

컨벡스 헐은 객체의 모양 또는 외각선을 분석하는 효과적인 방법 중 하나로 그 방식에 따라 Graham scan, Jarvis march, Divide and conquer로 불린다. 이는 불연속적으로 분포하는 점들을 포함하는 다각형을 결정하는 알고리즘이다. 이는 객체의 컨벡스 헐을 구하고 이로부터 컨벡스 결함을 구하는 것이다. 또한 복잡한 경우에도 효과적으로 사용할 수 있다. 영상은 많은 점들을 갖고 있으며 이러한 점들은 동일한 성질 그룹을 형성하며 컨벡스 헐은 그룹의 경계선을 추출하기 위해 연구 되었다. 그리고 그 성질이 같은 점들의 외각 정보를 그림 3.15와 같이 계산할 수 있다[31].

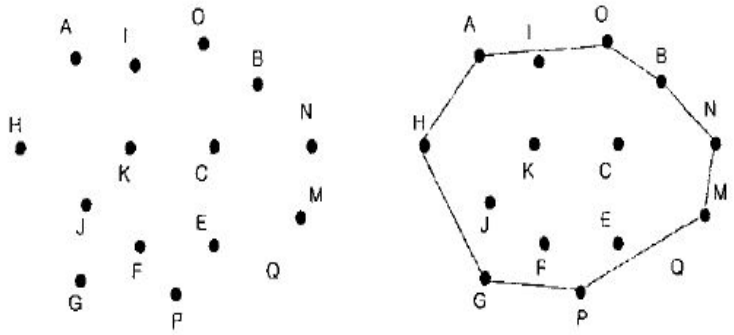


그림 3.15 컨벡스 헐의 입력과 출력 (좌: 입력, 우: 출력)

사. 컨벡스헐를 이용한 꼭지점 검출

컨벡스 헐 알고리즘은 가장 아랫점을 기준으로 각도 별로 정렬한 뒤 세 점으로 이루는 각도의 방향에 따라 포함 또는 제거하는 과정으로 시작점에 도작할 때 까지 반복된다. 이는 검출된 특징점의 꼭지점을 구하는데 사용되고 화면상의 좌표를 제공해준다. 즉 이진화 된 영상에 레이블링 처리를 한 후 컨벡스 헐 알고리즘을 이용하면 얻을 수 있는 점들은 특징점의 꼭짓점만 해당되게 된다. 그림 3.16의 흰색 사각형의 꼭지점은 Opnecv 라이브러리의 함수인 cvConvexhull2()의 함수를 이용하여 검출된 모습이다.

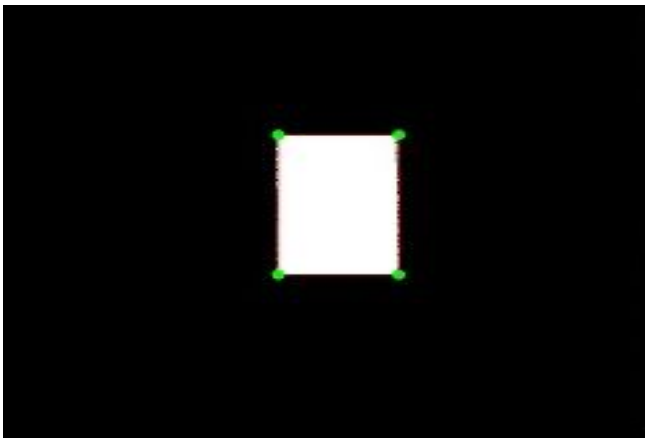


그림 3.16 컨벡스 헐 알고리즘을 통한 꼭지점검출

아. 꼭지점 검출 소스 코드

그림 3.17는 특징점의 꼭지점 검출하는 소스 코드이다. 여기서 Blob 클래스를 이용하여 이진화된 이미지를 입력한다. 입력된 이미지에서 흰색 영역을 레이블링 하게 된다. BlobSmallSizeConstraint()와 BlobSmallSizeConstraint()는 레이블링 처리를 할 최소 흰색 영역의 픽셀 사이즈와 최대 픽셀 사이즈를 정해주게 된다. 이는 특징점의 흰색 영역이 아닌 잡음으로 된 영역을 제거 하는 함수이다.

영상에서 처리된 레이블링의 개수만큼 cvFindContours()를 이용하여 코너 점을 추출하게 되며 이를 저장하고 이를 꺾이는 점이 큰 코너점 즉 검출된 특징점의 꼭지점에 해당하는 픽셀만 추출해낸다.

```
m_Blob.SetParam(m_GrayImage,50);
m_Blob.DoLabeling();
int nMaxWidth   = m_GrayImage->width  * 9 / 10;
int nMaxHeight  = m_GrayImage->height * 9 / 10;

m_Blob.BlobSmallSizeConstraint( 10, 10 );
m_Blob.BlobBigSizeConstraint( nMaxWidth, nMaxHeight );


for(int i=0 ; i < m_Blob.m_nBlobs ; i++)
{
        CvPoint pt1 = cvPoint(m_Blob.m_recBlobs[i].x,m_Blob.m_recBlobs[i].y);
        CvPoint pt2 = cvPoint(pt1.x + m_Blob.m_recBlobs[i].width,pt1.y + m_Blob.m_recBlobs[i].height);
        CvScalar color = cvScalar(0,0,255);
        cvDrawRect(m_DownImage,pt1,pt2,color);

        cvCircle(m_DownImage,pt1,3,cvScalar(0,255,0));
        cvCircle(m_DownImage,pt2,3,cvScalar(0,255,0));
        cvCircle(m_DownImage,cvPoint(pt1.x,pt2.y),3,cvScalar(0,255,0));
        cvCircle(m_DownImage,cvPoint(pt2.x,pt1.y),3,cvScalar(0,255,0));

        cvSmooth(m_GrayImage,m_GrayImage,CV_MEDIAN);
        cvSmooth(m_GrayImage,m_GrayImage,CV_GAUSSIAN);


        cvSetImageROI(m_GrayImage,cvRect(pt1.x,pt1.y,m_Blob.m_recBlobs[i].width,m_Blob.m_recBlobs[i].height));
        m_RoiImage = cvCreateImage(cvGetSize(m_GrayImage),IPL_DEPTH_8U,1);
        cvCopy(m_GrayImage,m_RoiImage);
        cvResetImageROI(m_GrayImage);
        cvFindContours( m_RoiImage, storage, &contour, sizeof(CvContour),
                CV_RETR_LIST,CV_CHAIN_APPROX_NONE, cvPoint(0,0));

        if (!(contour == NULL))
        {
                hull = cvConvexHull2(contour,storage,CV_CLOCKWISE,1);
                CvSeq* Approx_contours = cvApproxPoly( hull, sizeof(CvContour), storage,CV_POLY_APPROX_DP, 9,10);
                CvPoint* st  = (CvPoint*)cvGetSeqElem( Approx_contours, 0 );
                for (int i = 0; i < Approx_contours->total ; ++i)
                {
                        CvPoint* st1  = (CvPoint*)cvGetSeqElem( Approx_contours, 0 );
                        cvLine(m_DownImage, *st, *st1, CV_RGB(255, 255, 255));
                        cvCircle(m_DownImage,cvPoint(st[i].x+pt1.x ,st[i].y+pt1.y),2,cvScalar(0,0,255),2);
                }
                SetVertextPoint((POINT *)st,pt1,Approx_contours->total);
        }
        cvReleaseImage(&m_RoiImage);
}
```

그림 3.17 레이블링과 컨벡스헐 처리 소스코드

꼭지점 검출 알고리즘은 다음과 같다.

1 : for $Number\,of\,Labeling$
2 : $x, y = Labeling\ image$
3 : $ROI\ Image = SetROI(x, y)$
4 : $contour = cvFindContours(ROI\ Image)$
5 : if $Contour = TRUE$
6 : $Hull = cvConvexHull2(contour)$
7 : $Approx\ _contours = cvApproxPoly(Hull)$
8 : for $contours\ total$
9 : $x, y = cvGetSeqElem(Approx\ _Contours)$
10 : $endfor$
11 : $endfor$

제 3절 영상의 좌표 변환

1. 기존의 좌표 변환 방법

가. 핀홀 카메라(Pinhole Camera) 모델

핀홀 카메라는 렌즈대신 핀홀을 사용한 것으로 바늘구멍 사진기라고도 한다. 핀홀 카메라모델은 물체의 반사된 빛이 카메라의 렌즈를 거쳐 화면에 보여지는 구조를 관찰할 수 있다. 핀홀 카메라의 원리는 그림 3.18과 같이 3차원인 실제 세계에서 빛이 물체의 어느 한 점이 영상에서 반전된 모습으로 상이 맺히는 원리이다.

핀홀 카메라에서 측정 되는 점들의 좌표를 왜곡이 된 위치는 식 3.2 로 구성된다. 여기서 측정되는 좌표는 (x_p, y_p) 이고 왜곡이 된 자표는 (x_d, y_d) 이다. 또한 보정 정보를 이용한 식 3.3로 구성 된다 여기서 k_1, k_2, p_1, p_2, k_3는 왜곡 계수이다[32][33].

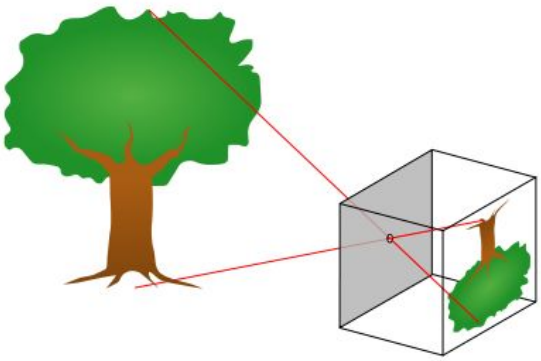


그림 3.18 핀홀 카메라의 영상 맺힘

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} f_x X^W / Z^W + c_x \\ f_y Y^W / Z^W + c_y \end{bmatrix} \qquad \text{(식 3.2)}$$

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2\left(r^2 + 2x^{2_d}\right) \\ 2p_2 x_d y_d + p_2\left(r^2 + 2y^{2_d}\right) \end{bmatrix} \qquad \text{(식 3.3)}$$

나. 핀홀 카메라 투영

카메라가 보정된 후 월드 좌표계의 한 점은 그림 3.19과 같이 영상의 한 점으로 투영(projection)할 수 있다.

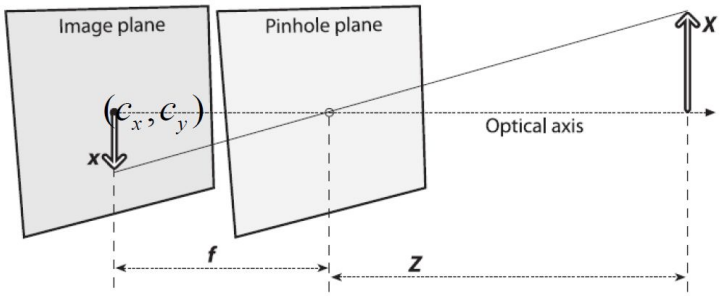


그림 3.19 핀홀 카메라 모델

3차원 실세계에 존재하는 점 $Q=(X,Y,Z)$가 영상 평면에 맺히는 점의 좌표 (x_s,y_s)는 다음과 같은 식 3.4로 표현된다. 여기에서 f 는 초점거리(focal length)이고 (c_x,c_y)는 광축과 영상 평면이 만나는 점이다. 또한 실세계의 한 점(X,Y,Z)이 투영 스크린 상의 한 점(x,y)로 변환되는 관계를 투영 변환이라고 한다. 이는 식 3.5로 정의 된다[34].

$$x_s = f_x \frac{X}{Z} + c_x,\ y_s = f_y \frac{X}{Z} + c_y \qquad \text{(식 3.4)}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sH \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad H = MW, \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(식 3.5)}$$

$$W = [R \ \ t], \quad R = [r_1, r_2, r_3]$$

2. 제한한 좌표 변환 방법

가. 영상 좌표계와 월드 좌표계

키넥트에서 촬영된 영상의 크기는 가로(horizontal) 640, 세로(vertical) 480 픽셀로 구성되어 있다. 또한 특징점이 검출 된 RGB영상에서 꼭지점의 좌표계인 그림 3.20과 같은 영상 좌표계와 실제 특징점이 위치하는 월드 좌표계로 오른손 좌표계로 표현된다.

특징점의 스크린 좌표계를 이용하여 실제 특징점이 존재하는 월드 좌표를 구할 수 있다. 또한 이러한 정보를 가지고 특징점의 크기를 구분해 낼 수 있다.

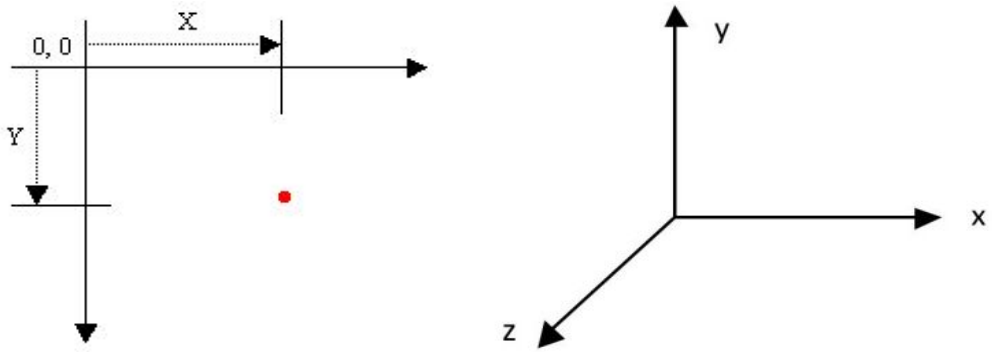


그림 3.20 좌표계 (좌: 화면좌표계, 우: 월드좌표계)

나. 제안된 특징점의 정보 추출 방법

키넥트의 영상에서 검출된 특징점의 꼭지점 좌표와 거리를 이용하여 그림 3.21과 같이 θ_x, θ_y를 구한다.

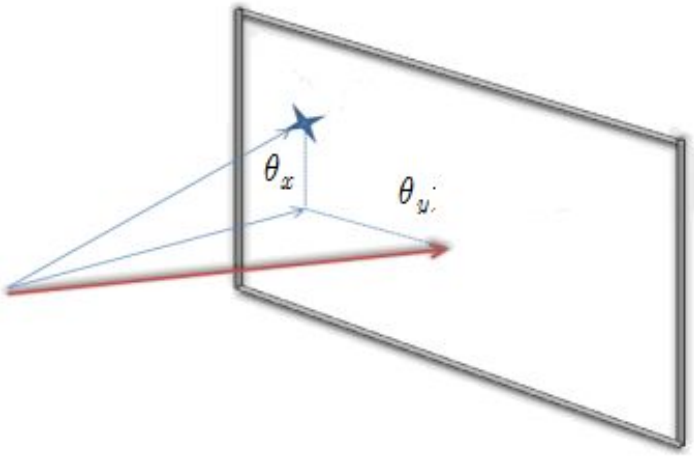


그림 3.21 키넥트와 특징점의 각도

키넥트에서 검출된 특징점 각도를 구하기 위해서는 그림 3.22와 같이 다양한 정보가 필요하다.

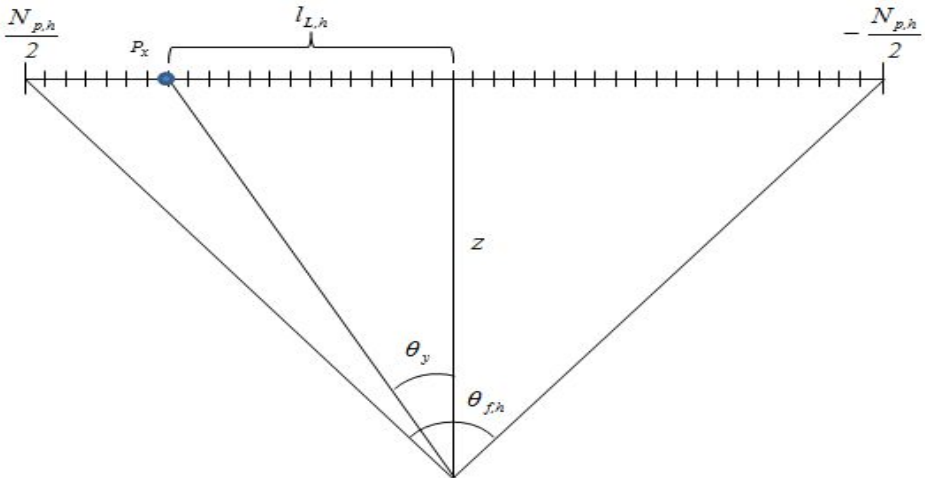


그림 3.22 특징점의 실제 거리를 구하기 위한 기하학적 모델

그림에서 3.22에서 검출된 특징점의 한점과 키넥트 영상의 원점의 x거리인 $l_{L,h}$를 구하기 위해서는 식 3.6과 같이 정의될 수 있다. 여기서 $l_{L,h}$는 키넥트 영상의 중점에서 검출된 특징점의 x축 방향의 실제 거리를 나타내며 $\theta_{f,h}$는 키넥트의 거리 정보를 나타내는 Depth 영상의 시야각(Field of View)이다. 또한 $N_{p,h}$는 가로

(Horizontal)의 픽셀 개수를 나타내며 P_x는 검출된 특징 점의 꼭지점 x좌표를 나타낸다. 또한 식 3.7은 y 방향의 실제 거리를 나타내는 식이다.

$$l_{L,h} = \frac{\tan\left(\frac{\theta_{f,h}}{2}\right)}{\left(\frac{N_{p,h}}{2}\right)} \bullet \left(\frac{N_{p,h}}{2} - P_x\right) \qquad \text{(식 3.6)}$$

$$l_{L,v} = \frac{\tan\left(\frac{\theta_{f,v}}{2}\right)}{\left(\frac{N_{p,v}}{2}\right)} \bullet \left(\frac{N_{p,v}}{2} - P_y\right) \qquad \text{(식 3.7)}$$

이와 같이 $l_{L,h}$를 구한 후 키넥트를 기준으로 특징점 각각의 꼭지점의 x방향의 각도 θ_x, θ_y 를 구하는 방법으로 식 3.8과 같이 정의할 수 있다

$$\theta_x = \tan^{-1}\left(\frac{l_{L,h}}{z}\right),\ \theta_y = \tan^{-1}\left(\frac{l_{L,v}}{z}\right) \qquad \text{(식 3.8)}$$

3. 특징점 구별 및 좌표 보정

가. 특징점의 크기 구별 방법

미리 주어진 특징점은 이동 로봇이 구동하면서 다양한 방향에서 특징점이 검출이 되게 된다. 또한 특징점의 크기는 다양한 방향에서 영상에 비춰지는 모습이 다르기 때문에 영상 좌표만을 이용한 특징점의 크기 구별은 불가능하다. 그러므로 영상에 비춰지는 특징점의 좌표와 거리를 이용하여 검출된 특징점의 각각의 크기를 구별해야 한다. 또한 크기가 다른 같은 모양의 특징점을 구별하기 위해서는 특징점의 본래의 크기를 계산해야 한다.

키넥트의 영상 그림 3.23과 같이 바라보는 시점에 따라 영상에 비춰지는 특징점의 크기는 그림 3.24와 같이 다르게 비춰진다.

그림 3.23 다양한 방향에서의 특징점

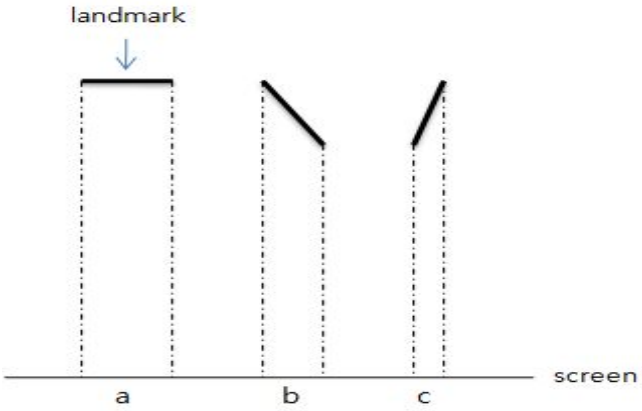
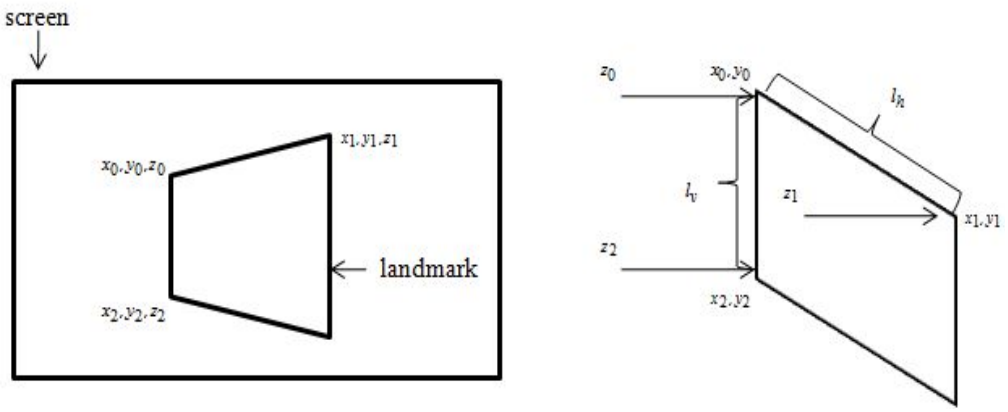


그림 3.24 다양한 방향에서의 특징점의 크기

그림 3.25의 a는 검출된 특징점의 영상에서 비춰진 모습과 b는 실제 특징점의 모습을 나타낸다. 이와 같이 특징점이 정면이 아닌 다른 방향에서 검출됐을 때 특징점의 크기는 식 3.9와 식 3.10과 같이 유클리디안 거리(Euclidean distance)를 이용하여 실제 크기를 구할 수 있다. 여기서 x_i, y_i는 영상에 비춰지는 특징점의 꼭지점 좌표 이며 z_i 는 키넥트에서 획득 한 특징점의 각각의 꼭지점 거리 정보이다.



(a) (b)

그림 2.25 화면상에 비춰진 특징점 및 실제 특징점의 모습

$$l_h = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} \quad \text{(식 3.9)}$$

$$l_v = \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2 + (z_2 - z_0)^2} \quad \text{(식 3.10)}$$

나. RGB영상과 Depth영상 좌표 보정

그림 3.26과 같이 키넥트의 내부는 RGB 영상 카메라와 깊이 정보를 나타내는 적외선 센서의 위치가 다르다. 또한 RGB의 영상과 Depth영상은 서로 다른 시야각과 초점거리로 인해 그림 3.27과 같이 보여지는 영상이 1:1 매칭이 되지 않는다.

그림 3.26 키넥트 내부 센서

그림3.27를 보면 RGB영상과 Depth 영상이 1:1 매칭이 되지 않는 것을 확인할 수 있다. 여기서 좌측 영상은 실제 RGB영상에서 촬영된 원본 영상이고 중간 영상이 거리를 나타내는 Depth영상이다. 그리고 우측 영상은 RGB영상과 Depth영상을 매칭 시킨 모습이다. 우측 영상에서 검은색 영역은 키넥트 센서의 감지거리 밖의 모습 즉 키넥트에 검출되지 못한 영역이고 컬러로 된 영역은 검출된 영역이다.

그림 3.27 RGB영상과 Depth영상 매칭 (좌: RGB영상, 중: Depth영상, 우: 매칭 영상)

다. 키넥트 영상 보정 함수

특징점을 검출하는 RGB영상과 검출된 특징점의 거리 정보를 나타내는 Depth 영상간의 픽셀 좌표의 오차가 발생한다. 이동 로봇의 중심으로 검출된 특징점의 거리와 방향을 구하기 위해 RGB영상에서 검출된 특징점의 픽셀 좌표가 Depth 영상의 픽셀 좌표와 1:1 매칭이 되어야 한다.

그림 3.28 과 그림 3.29는 키넥트 SDK 라이브러리에서 RGB영상과 Depth 영상의 픽셀 좌표를 보정하는 함수이다[35].

```
HRESULT ( STDMETHODCALLTYPE *NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution )(
    INuiSensor * This,
    /* [in] */ NUI_IMAGE_RESOLUTION eColorResolution,
    /* [in] */ NUI_IMAGE_RESOLUTION eDepthResolution,
    /* [in] */ const NUI_IMAGE_VIEW_AREA *pcViewArea,
    /* [in] */ LONG lDepthX,
    /* [in] */ LONG lDepthY,
    /* [in] */ USHORT usDepthValue,
    /* [out] */ LONG *plColorX,
    /* [out] */ LONG *plColorY);
```

그림 3.28 RGB영상과 Depth 영상 픽셀 좌표 보정 함수 1

```
HRESULT ( STDMETHODCALLTYPE *NuiImageGetColorPixelCoordinateFrameFromDepthPixelFrameAtResolution )(
    INuiSensor * This,
    /* [in] */ NUI_IMAGE_RESOLUTION eColorResolution,
    /* [in] */ NUI_IMAGE_RESOLUTION eDepthResolution,
    /* [in] */ DWORD cDepthValues,
    /* [size_is][in] */ USHORT *pDepthValues,
    /* [in] */ DWORD cColorCoordinates,
    /* [size_is][out][in] */ LONG *pColorCoordinates);
```

그림 3.29 RGB영상과 Depth 영상 픽셀 좌표 보정 함수 2

라. 키넥트 영상 보정 방법

키넥트의 RGB영상과 Depth영상의 보정 방법 및 확인으로 키넥트의 감지 거리에 검출된 RGB영상을 Depth 영상에 매칭 하여야 한다. 즉 Depth 영상에 검출된 픽셀 좌표를 위 함수를 이용하여 보정된 RGB의 픽셀 좌표를 얻는다.

보정된 RGB의 픽셀 좌표의 색상 정보를 검출된 Depth 영상의 픽셀에 입력 하게 되면 RGB영상과 Depth 영상의 보정된 영상 및 결과를 확인할 수 있다. 그림 3.30은 RGB영상과 Depth영상을 매칭 시키는 소스 코드이다.

NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution()함수를 사용하여 두 영상을 보정 및 매칭 시킬 수 있다.

```
long cx;
long cy;
int ax;
int ay;
int icnt = 0;
for( ay = 0 ; ay < 480 ; ay++ )
{
        for( ax = 0 ; ax < 640 ; ax++)
        {
                m_pNuiSensor->NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(
                NUI_IMAGE_RESOLUTION_640x480,NUI_IMAGE_RESOLUTION_640x480,NULL,ax,ay,0,&cx,&cy);

                if (cx > 640)
                {
                        cx = 639;
                }
                if (cy > 480)
                {
                        cy = 479;
                }
                if (m_realDepth[icnt] == 0 )
                {
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+0] = 0;
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+1] = 0;
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+2] = 0;
                }
                else
                {
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+0]=
                                m_pImage->imageData[cy*m_pImage->widthStep+cx*m_pImage->nChannels+0];
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+1]=
                                m_pImage->imageData[cy*m_pImage->widthStep+cx*m_pImage->nChannels+1];
                        m_DepthImage->imageData[ay*m_DepthImage->widthStep+ax*m_DepthImage->nChannels+2]=
                                m_pImage->imageData[cy*m_pImage->widthStep+cx*m_pImage->nChannels+2];
                }
                icnt++;
        }
}
```

그림 3.30 RGB영상과 Depth 영상의 매칭 소스 코드

두 영상의 보정 및 매칭 알고리즘으로 다음과 같다.

```
x,y = Depth image coordinate
x',y' = Color image coordinate
for Verical size of Image
    for Horizontal size of Image
    x',y' = NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(x,
    if Depth of Pixel(x,y) = FLASE
        Depth Image(x,y) = 0
    else
        Depth Image(x,y) = Color Image(x',y')
    endfor
endfor
```

제4장 실험 및 결과

제 1절 키넥트를 이용한 특징점 검출

1. 특징점 검출

가. RGB 채널 영상에서 특징점 검출

그림 4.1은 RGB 채널을 그대로 사용하여 특징점의 색을 검출한 모습이다. 그림에서 보는 것과 같이 주변 환경의 노이즈로 인한 이진화 처리가 의미가 없다. 또한 이진화 처리를 하여 검출했을 경우 불확실성이 크게 나타나진다.

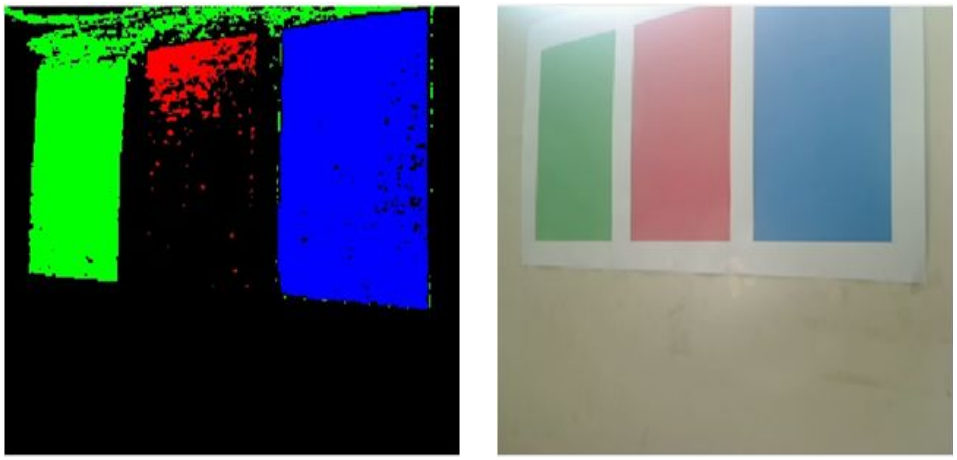


그림 4.1 RGB 채널을 사용한 특징점 검출

나. Ycbcr 채널 영상에서 특징점 검출

그림 4.2은 Ycbcr 채널로 변환하여 그 결과를 확인하였다. 그림 4.1과 비교하여 같은 환경에서 RGB채널을 사용했을 경우보다 검출이 잘되고 검출된 특징점의 이진화 처리 또한 좋은 결과를 얻었다.

그림 4.2 Ycbcr 채널을 이용한 특징점 검출

2. 레이블링 검출 확인

가.Ycbcr 채널을 이용한 레이블링 검출 결과

그림 4.3은 Ycbcr 채널로 변환 하여 그 결과를 확인하였다. 그 그림에서 보는 것과 같이 사각형의 블록으로 특징점의 외각을 나타내었다. 외부 조명의 노이즈가 조금 포함된 상태이지만 비교적 잘 검출된 것을 확인할 수 있다.

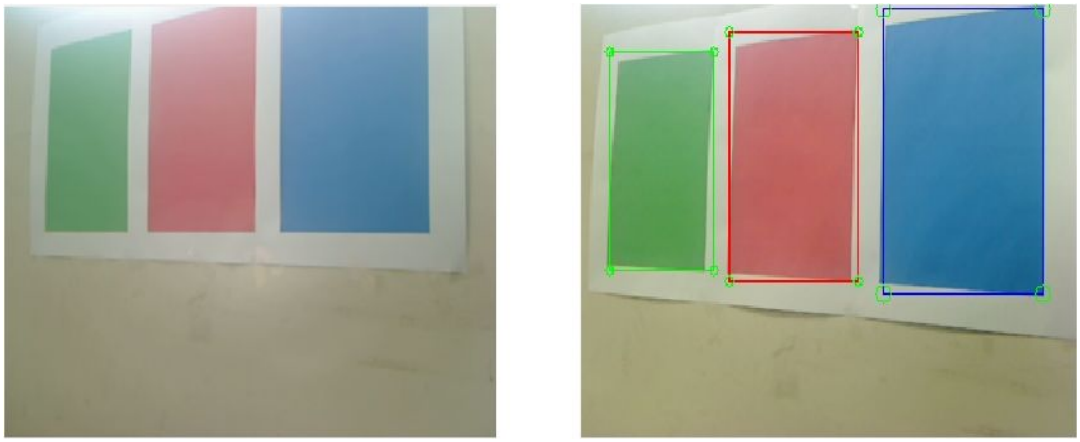


그림 4.3 검출된 특징점의 레이블링 처리

그림 4.4는 검출된 특징 점의 레이블링하여 결과를 확인하고 이때의 이진화 영

상의 형태를 본 영상이다. 외부 빛의 노이즈를 조금 받은 상태이지만 이진화 영상 또한 비교적 정확한 검출을 된 것을 확인하였다.

그림 4.4 레이블링 처리된 특징점의 이진화 영상

나. 주변 환경의 변화에 따른 레이블링 검출 결과

그림 4.5는 주변 환경의 변화에 따른 특징점 검출 및 레이블링의 형태를 나타낸다. 그림에서 보는 것과 같이 주변 환경의 노이즈에 따라 잘못된 특징점의 레이블링 처리된 것을 확인하였다.

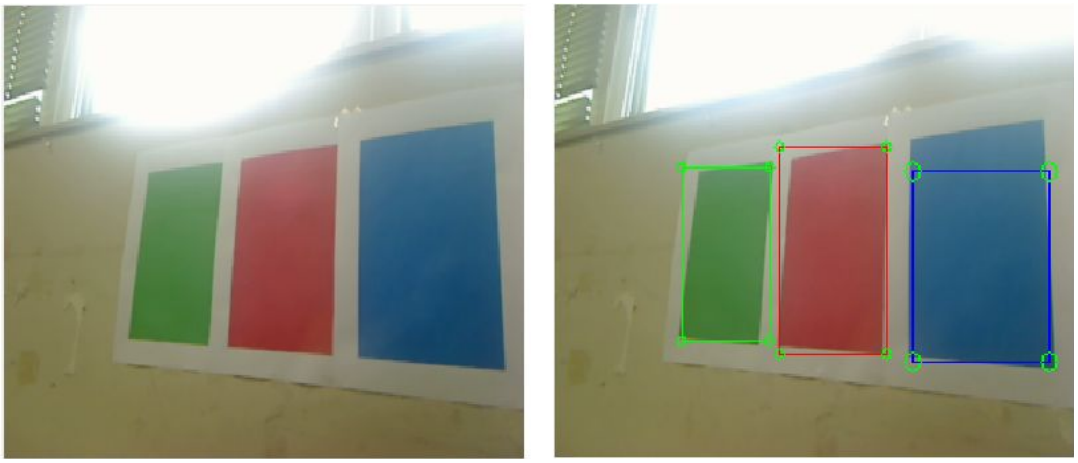


그림 4.5 주변 노이즈에 따른 특징점의 레이블링 처리 결과

그림 4.6은 외부 환경의 노이즈가 심할 경우 검출된 특징점의 이진화 영상이다. 그림에서 보이는 것처럼 외부 조명에 노이즈를 받아 검출하고자 하는 색의 변질이 일어나기 때문에 정확한 특징점을 검출하지 못하였다.

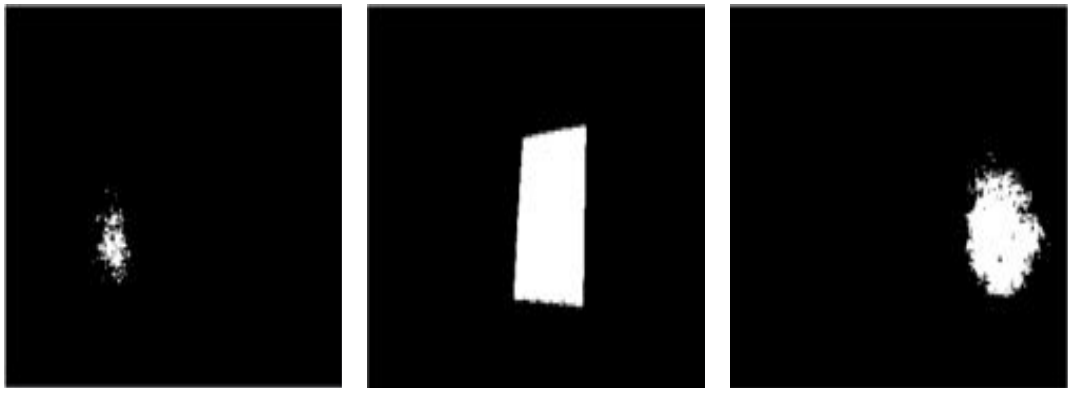


그림 4.6 외부 노이즈에 따른 검출된 특징점의 이진화 영상

다. 관심 영역 확인

이진화된 특징점의 영상에서 검출된 특징점의 흰색 영역과 잡음으로 발생한 흰색 영역이 발생하게 된다. 그림 4.7에 보이는 것처럼 특징점 영역과 외부 노이즈로 발생한 영역이 검출되었다. 이를 제거하기 위해 레이블링 처리를 한 검출된 특징점의 꼭지점을 찾는 전처리 단계로 검출된 특징점의 영역만 추출해내는 관심영역을 지정하였다.

그림 4.7 특징점 검출 환경의 노이즈

그림 4.8은 검출된 특징점의 영역을 관심영역으로 지정하고 그 영역만 분리시킨 영상이다. 그 결과 영상의 배경에 검출되었던 노이즈 영역을 제거할 수 있었을 뿐만 아니라 이미지 처리에 소요되는 연산 속도도 줄일 수 있게 되었다.

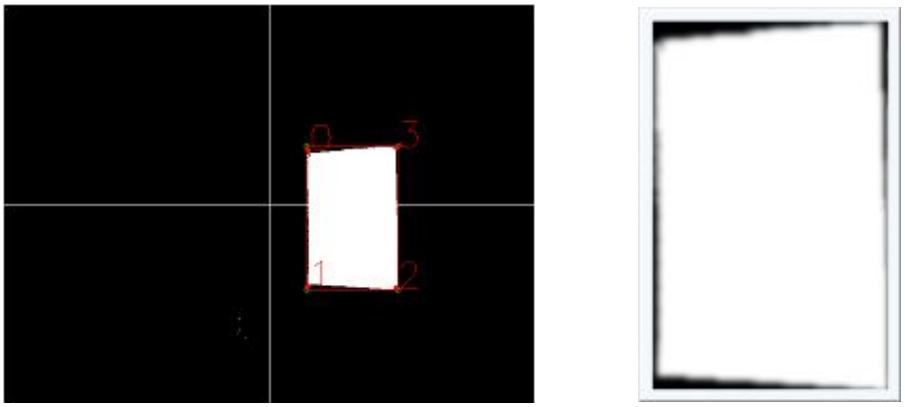


그림 4.8 관심영역 영상 (좌: 검출된 특징점, 우: 분리된 특징점 영상)

3. Convex Hull을 이용한 특징점 좌표 추출

가. 컨벡스 헐 결과 확인

그림 4.9에 보이는 검출된 특징점의 외각을 잇는 레이블링과 그것을 관심영역으로 지정된 영상에서 컨벡스 헐 처리를 한 영상이다. 그림에서 보이는 녹색원으로 표시가 된 부분은 레이블링 처리가 된 특징점의 외각의 부분이며 붉은색 원은 실제 특징점의 꼭지점을 검출 한 부분이다.

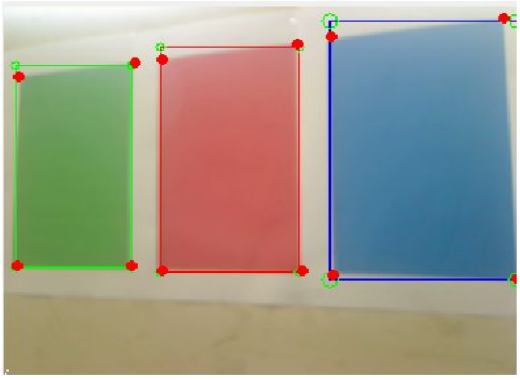


그림 4.9 컨벡스 헐 처리된 특징점 영상

나. 특징점의 꼭지점 좌표 추출

그림 4.10에 보이는 것처럼 좌측 상단은 1번으로 반 시계 방향으로 ID가 생성된다. 이는 각각의 해당 좌표를 확인하기 위해서 Text를 삽입하였다. 이는 검출된 꼭지점의 좌표를 나타내는 프로그램이다. 이는 키넥트 센서의 감지 거리에 포함되지 않았기 때문에 그림 중간에 보이는 검은색 영상이 비춰지게 된다. 또한 특징점을 검출 하지 못하였고 좌표 또한 추출해내지 못하였다. 하지만 그림 4.11과 같이 키넥트 센서의 감지 거리에 포함이 될 때에는 특징점의 좌표를 잘 추출해내는 것을 확인하였다.

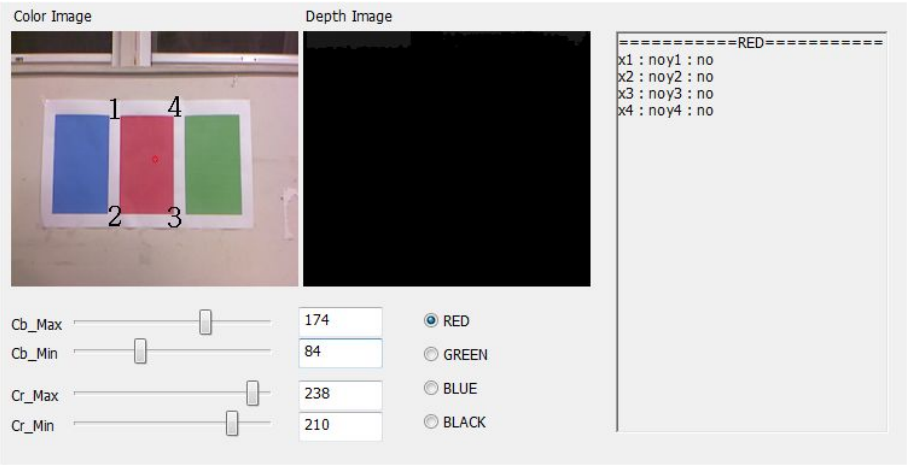


그림 4.10 특징점의 꼭지점 ID 확인

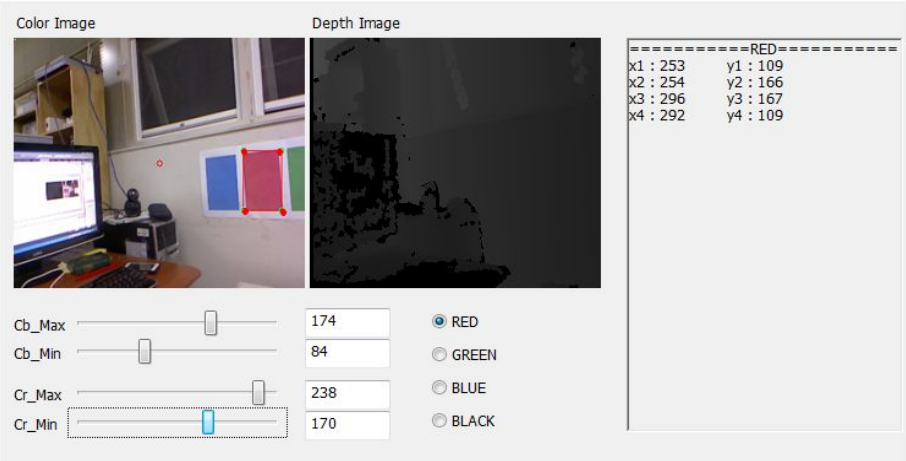


그림 4.11 특징점의 꼭지점 좌표 확인 및 Depth영상

제2절 키넥트 RGB 영상과 Depth 영상 보정

1. RGB 영상과 Depth 영상의 Calibration

가. RGB영상과 Depth 영상의 차이 확인

그림 4.12는 키넥트에서 보여지는 RGB 영상과 Depth 영상간의 오차를 확인하는 과정이다. 그림에서 보는 것과 같이 RGB영상과 Depth 영상간의 시야각의 차이로 인하여 1:1 매칭이 되지 않은 것을 확인할 수 있으며 이때의 RGB의 $x=0, y=0$ 의 픽셀의 깊이는 Depth의 영상에서의 $x \neq 0, y \neq 0$의 된다. 즉 오차 보정을 하지 않을 시에는 정확한 거리 정보를 획득하는데 무리가 있다. 또한 보정 작업을 하지 않고 진행시에는 화면 좌표계에서 특징점의 각도를 구하는 과정에서 오차가 심한 결과를 나타내는 상황이 발생 할 것이다.

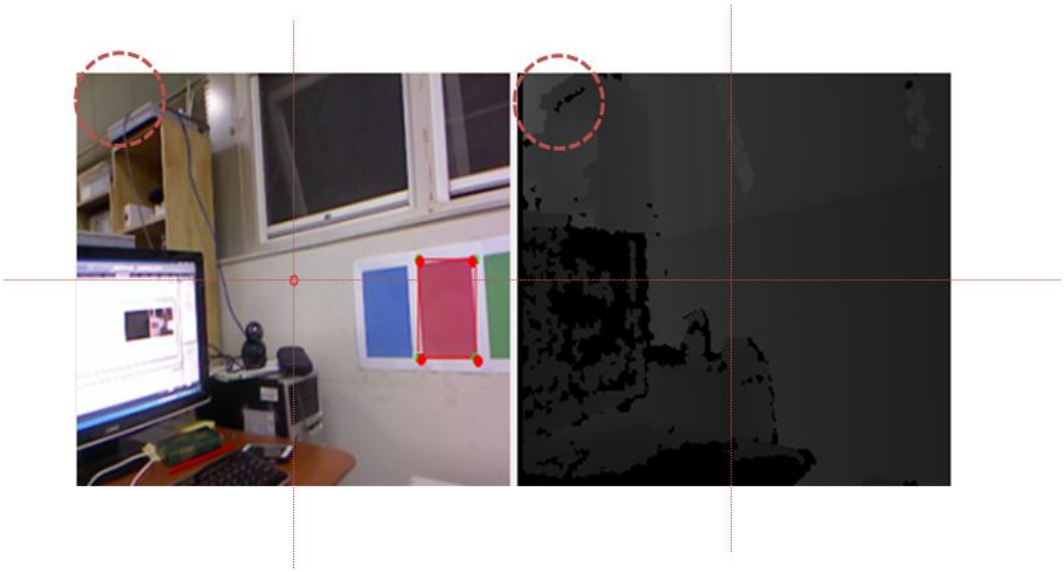


그림 4.12 RGB영상과 Depth영상의 오차

나. RGB영상과 Depth 영상의 오차

그림 4.13은 RGB 영상과 Depth 영상 간의 보정을 하지 않을 때의 매칭 한 결과이다. 그림의 우측 영상은 RGB영상과 Depth 영상을 매칭 시킨 영상이다. 붉은색 원으로 키넥트에 감지된 영역은 RGB영상의 색상으로 하고 검출되지 않은 다른 영역은 검은색으로 처리를 하였다. 그림에서 붉은색 영역처럼 RGB 영상과 Depth 영상이 1:1 매칭 되지 않은 것을 확인할 수 있다.

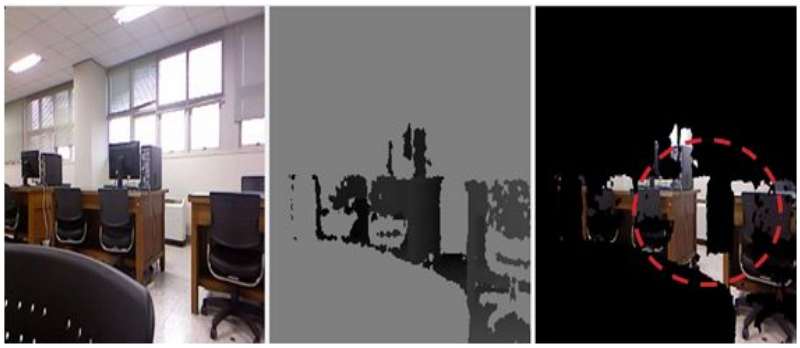


그림 4.13 RGB영상과 Depth 영상 매칭

그림 4.14는 RGB영상과 Depth 영상의 좌표 보정을 하지 않은 상태에서의 특징점을 검출한 모습이다. 그림 우측 영상에서 검출된 특징점의 꼭지점은 정확한 위치를 하고 있는 좌표를 Depth 영상의 좌표에 표시하게 되면 그림의 좌측 영상처럼 특징점의 꼭지점과 매칭 되지 않은 것을 확인할 수 있다.

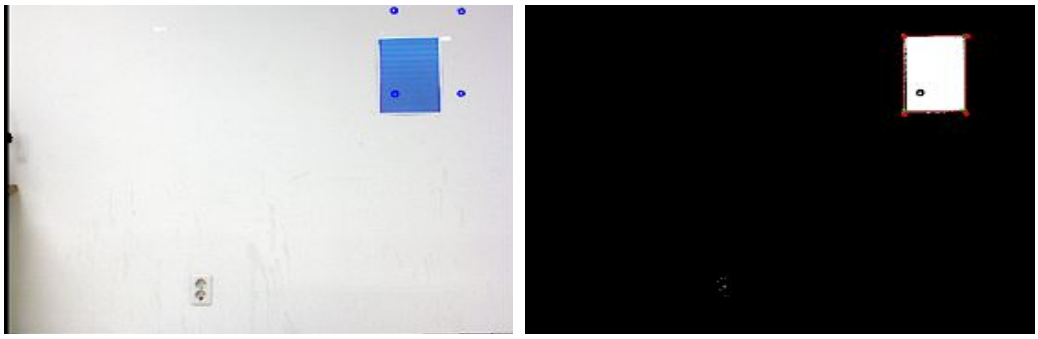


그림 4.14 특징점의 꼭지점 좌표 오차 (좌: Depth 영상, 우: 이진화된 RGB 영상)

다. RGB영상과 Depth 영상 Calibration 결과

그림 4.15는 오차 보정을 한 후의 매칭 한 결과이다. 이것은 키넥트 SDK 라이브러리에서 제공해주는 함수를 사용하여 오차보정을 하였다. 그 결과 그림에서 보이는 것과 같이 보정된 것을 확인하였다. 하지만 완벽한 1:1 매칭은 하지 못하였다. 그러나 보정하기 전과 비교하였을 때 확연히 좋은 결과를 나타낼 것으로 예상된다.

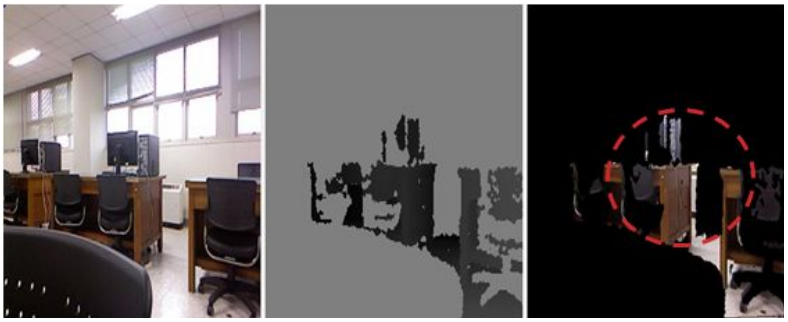


그림 4.15 RGB영상과 Depth 영상 보정 후 매칭

그림 4.16은 RGB영상과 Depth 영상의 좌표 오차 보정을 한 후 이진화된 RGB영상에서 특징점의 검출된 꼭지점 좌표를 Depth 영상에 표시한 영상이다. 오차 보정되지 않은 그림 4.14 와 다르게 그림 4.16은 검출된 꼭지점의 좌표가 Depth영상에서의 좌표와 일치하고 있는 것을 확인할 수 있다.

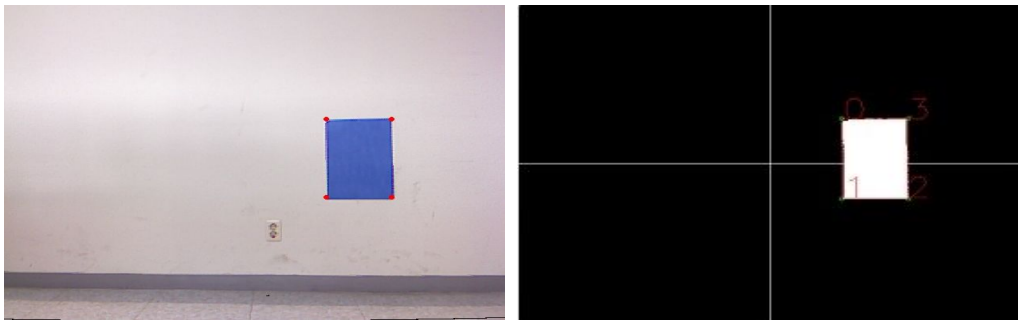


그림 4.16 특징점의 꼭지점 좌표 (좌: Depth 영상, 우: 이진화된 RGB 영상)

제3절 특징점의 위치 좌표 추출

1. 고정된 위치에서 특징점의 검출 및 실 거리 확인

가. 실험 환경

본 실험은 조선대학교 전자 정보 공과대학 6층 로봇 시스템 설계 실험실에서 이루어 졌다. 그림 4.17은 사용된 특징점으로 푸른색 사각형으로 가로 30cm 세로 45cm의 크기를 갖고 있다. 또한 그림 4.18과 같이 이동체에 키넥트를 설치하여 다양한 거리 및 각도에서 영상을 획득할 수 있게 하였다.

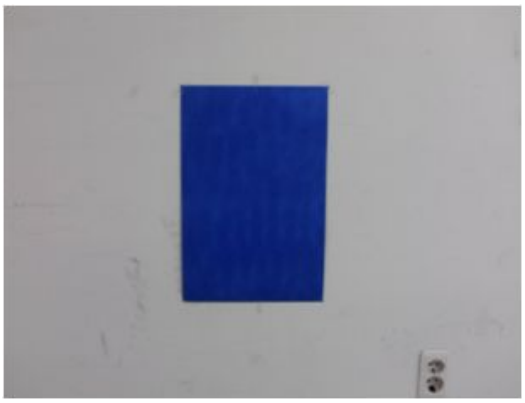


그림 4.17 사용된 특징점의 외형

그림 4.18 특징점 검출 환경

나. 영상의 좌표계

그림 4.19는 영상의 좌표계를 나타내는 것이다. 이는 스크린의 중심 점을 $x=0$, $y=0$ 으로 하여 +, -의 부호가 변하고 x축을 기준으로 하여 특징점의 위치에 대한 각도는 θ_x로 하고 y축의 중심으로 하는 각도를 θ_y로 정의하였다.

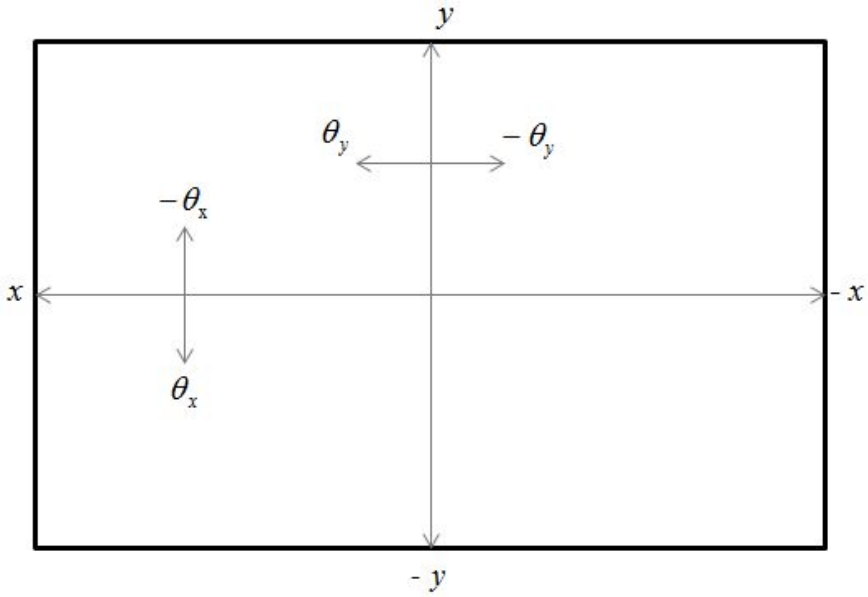


그림 4.19 화면 영상 좌표계

다. 검출된 특징점의 꼭지점 좌표

그림 4.20과 4.21는 특징점 및 꼭지점의 검출된 영상에서 좌표 정보와 키넥트를 기준으로 하는 특징점의 거리 그리고 각도 정보를 추출해 나타내는 모습이다. 그림 4.20에 좌측 영상에 좌측 상단 특징점의 꼭지점을 0번으로 반시계 방향으로 ID 값을 결정하였다.

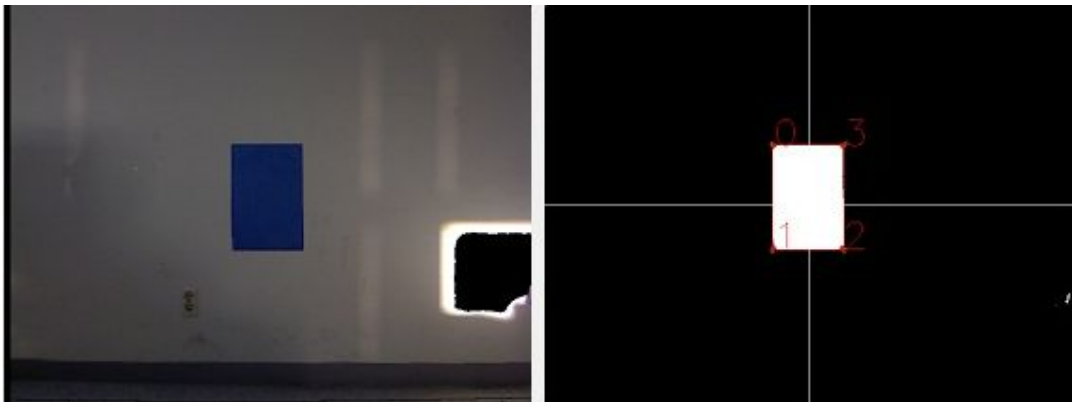


그림 4.20 검출된 특징점의 영상

그림 4.21에서 좌측에 있는 특징점 정보는 꼭지점의 개수와 좌표 그리고 키넥트를 기준으로 꼭지점 각각의 거리를 나타낸 정보이다. 실제 거리 2m 거리에서 측정한 결과 키넥트에 검출된 꼭지점의 거리는 약 4~5cm 정도 오차가 발생하였다.

실제 가로(Horizontal) 30cm, 세로(Vertical) 45cm의 크기를 갖고 있는 특징점을 키넥트를 기준으로 각도(Degree)와 영상의 원점에서 꼭지점 각각의 실제 크기(Meter)를 측정하였다. 그 결과 영상에 비춰지는 특징점 크기와 실제 유클리디안 거리를 이용하여 측정한 특징점의 거리는 mm 단위의 오차가 발생하였다.

그림 4.21 검출된 특징점의 정보

라. 거리와 각도의 변화에 따른 특징점 검출 및 정보

그림 4.22와 4.23은 거리 변화에서 측정된 꼭지점의 거리 정보와 각도 그리고 특징점의 크기를 나타낸 모습이다. 측정 거리는 약 1.5m에서 진행되었고 그 결과 측정된 꼭지점의 거리는 약 2~6cm 정도의 오차를 갖고 있으며 특징점의 크기는 약 0.2cm 정도 오차 정도가 측정되었다.

그림 4.22 거리에 따른 검출된 특징점 영상(1.5m)

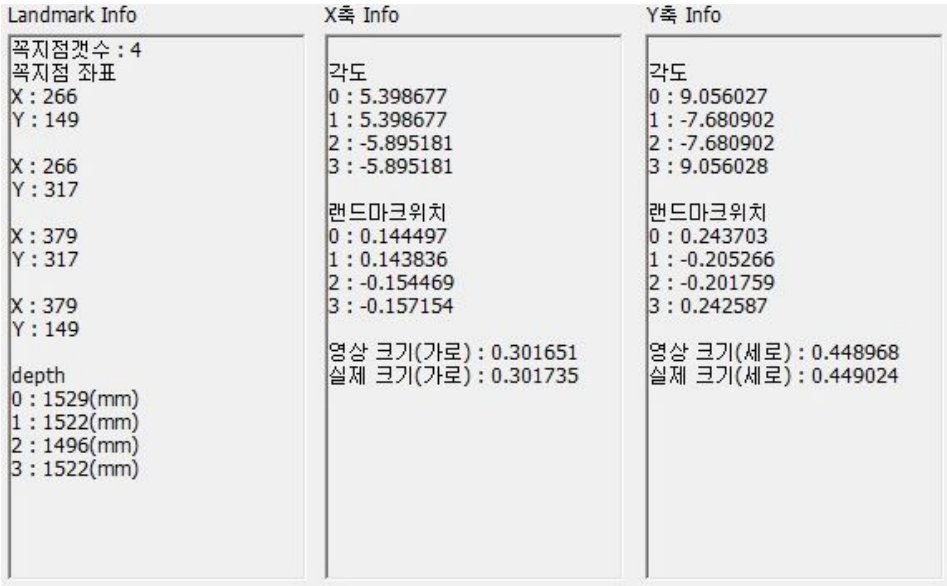


그림 4.23 거리에 따른 특징점 정보(1.5m)

그림 4.24와 4.25는 측정 거리 3m에서 진행 되었고 그 결과 측정된 꼭지점의 거리는 약 3~5cm 정도의 오차를 갖고 있으며 특징점의 크기도 약 0.1~0.7cm 정도 오차 정도가 측정되었다.

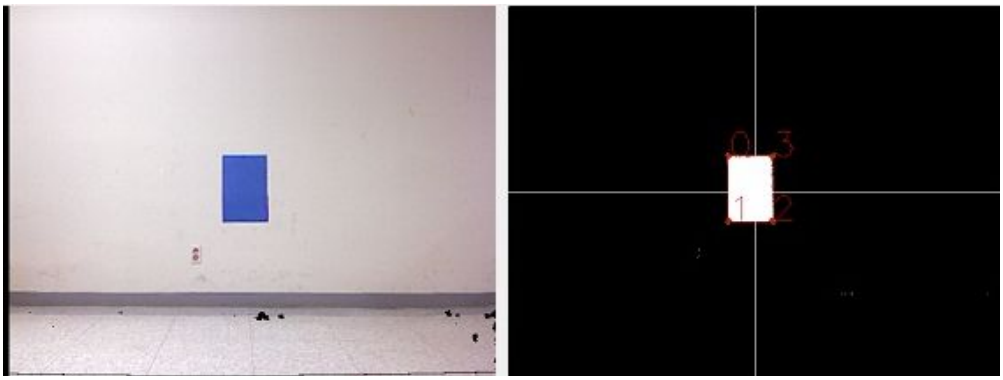
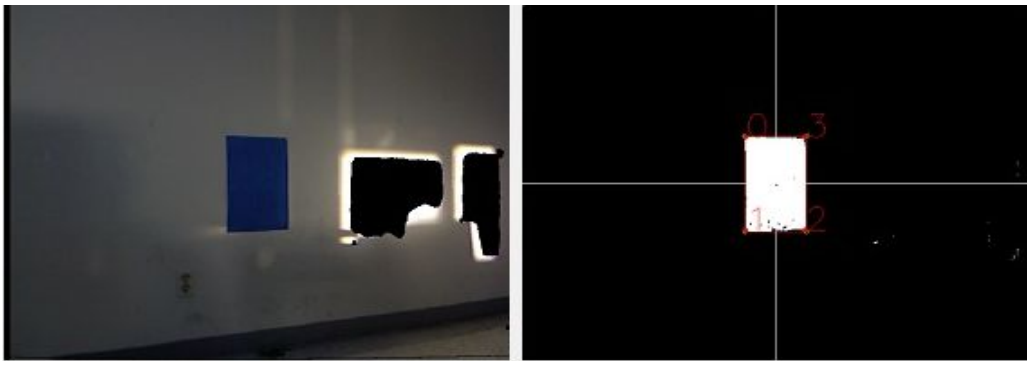


그림 4.24 거리에 따른 검출된 특징점 영상(3m)

그림 4.25 거리에 따른 특징점 정보(3m)

그림 4.26과 4.27은 각도에 따른 측정 결과를 나타낸 모습이다. 측정 거리는 특징점의 중점으로부터 약 2m에서 측정하였다. 그 결과 특징점의 꼭지점 거리의 차이가 있게 된다. 또한 영상에 비춰지는 특징점의 가로 크기는 3cm 오차가 발생하였다. 하지만 유클리디안 거리를 이용한 특징점의 실제 크기는 0.4cm의 오차가 발생 하였다.



Landmark Info

꼭지점갯수 : 4
꼭지점 좌표
X : 282
Y : 178

X : 282
Y : 304

X : 357
Y : 304

X : 357
Y : 178

depth
0 : 1998(mm)
1 : 1986(mm)
2 : 2107(mm)
3 : 2120(mm)

X축 Info

각도
0 : 3.804749
1 : 3.804750
2 : -3.704907
3 : -3.704907

랜드마크위치
0 : 0.132873
1 : 0.132075
2 : -0.136435
3 : -0.137277

영상 크기(가로) : 0.270150
실제 크기(가로) : 0.296716

Y축 Info

각도
0 : 6.197653
1 : -6.395947
2 : -6.395946
3 : 6.197654

랜드마크위치
0 : 0.216969
1 : -0.222623
2 : -0.236187
3 : 0.230218

영상 크기(세로) : 0.439593
실제 크기(세로) : 0.439757

그림 4.26 다른 방향에서의 특징점 검출 정보1

마. 거리 변화에 따른 측정된 꼭지점의 거리 그래프

그림 4.27은 키넥트에 검출된 꼭지점을. 총 100개의 데이터를 받아 측정된 오차 범위를 확인 한 결과 특징점과의 거리가 멀어질수록 측정된 꼭지점의 오차는 증가 되었다. 측정된 실제 거리는 2m~3.5미터로 약 50cm 간격으로 측정하였다.

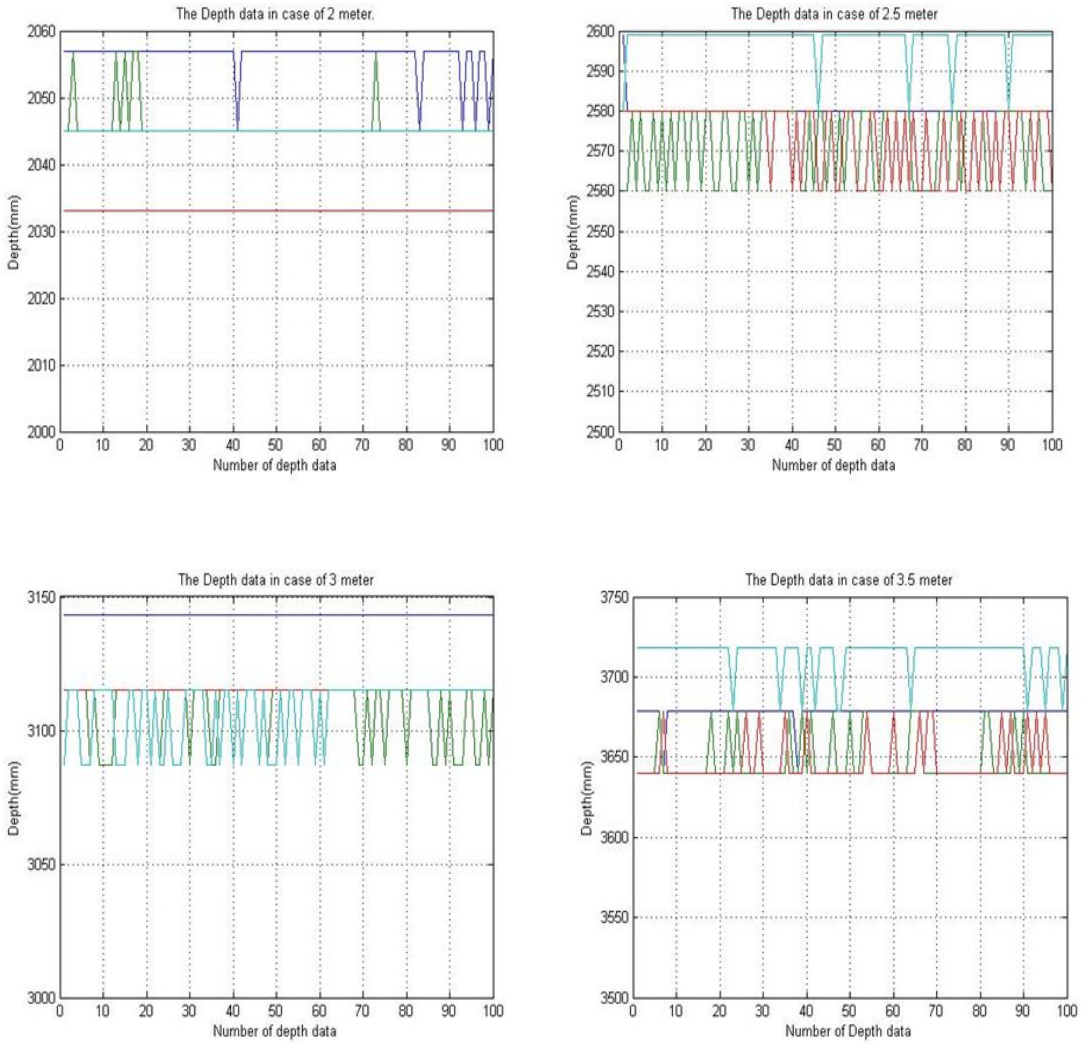


그림 4.27 측정된 꼭지점의 거리 그래프

2. 마우스 포인트를 이용한 특징점의 실거리 확인

가. 마우스 포인트를 이용한 꼭지점의 정보

영상에서 마우스 포인터가 가르키는 곳의 좌표와 거리 그리고 각도를 표현 하였다. 그림 4.28은 키넥트에서 검출된 특징점의 꼭지점에 마우스 포인터를 가르켜

측정된 결과를 비교한 것이다. 그림에서 보이는것 처럼 특징점의 꼭지점의 좌표에 마우스 포인터를 가르켰을때 정보와 키넥트에서 검출된 특징점의 꼭지점 정보가 비교적 일치하는 것을 확인할 수 있다.

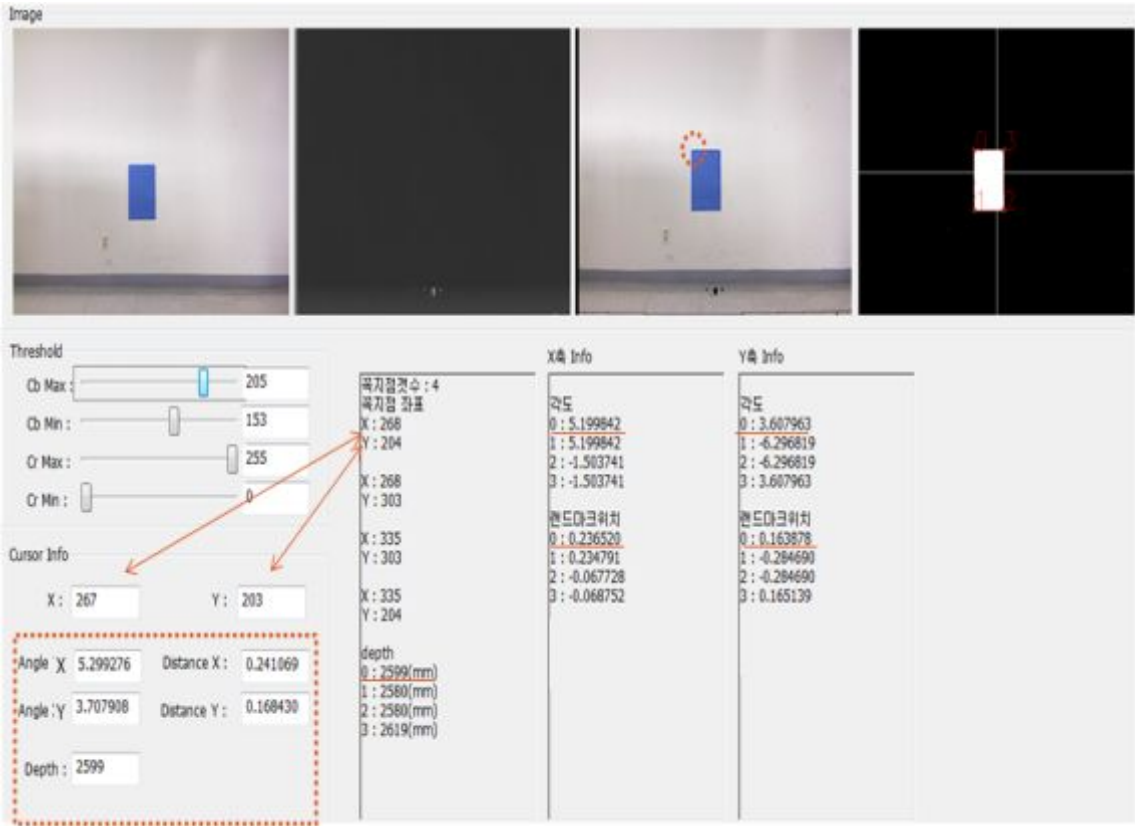


그림 4.28 마우스 포인터를 이용한 특징점 정보

나. 주변환경의 특징점 정보

지금까지는 기존의 키넥트에서 검출되었던 사각형의 특징점 정보를 보았다. 그림 4.29는 특정색으로 구성된 특징점을 제외한 마우스 포인터로 주변 환경을 가르켜 그 지점에 대한 정보를 나타내었다.

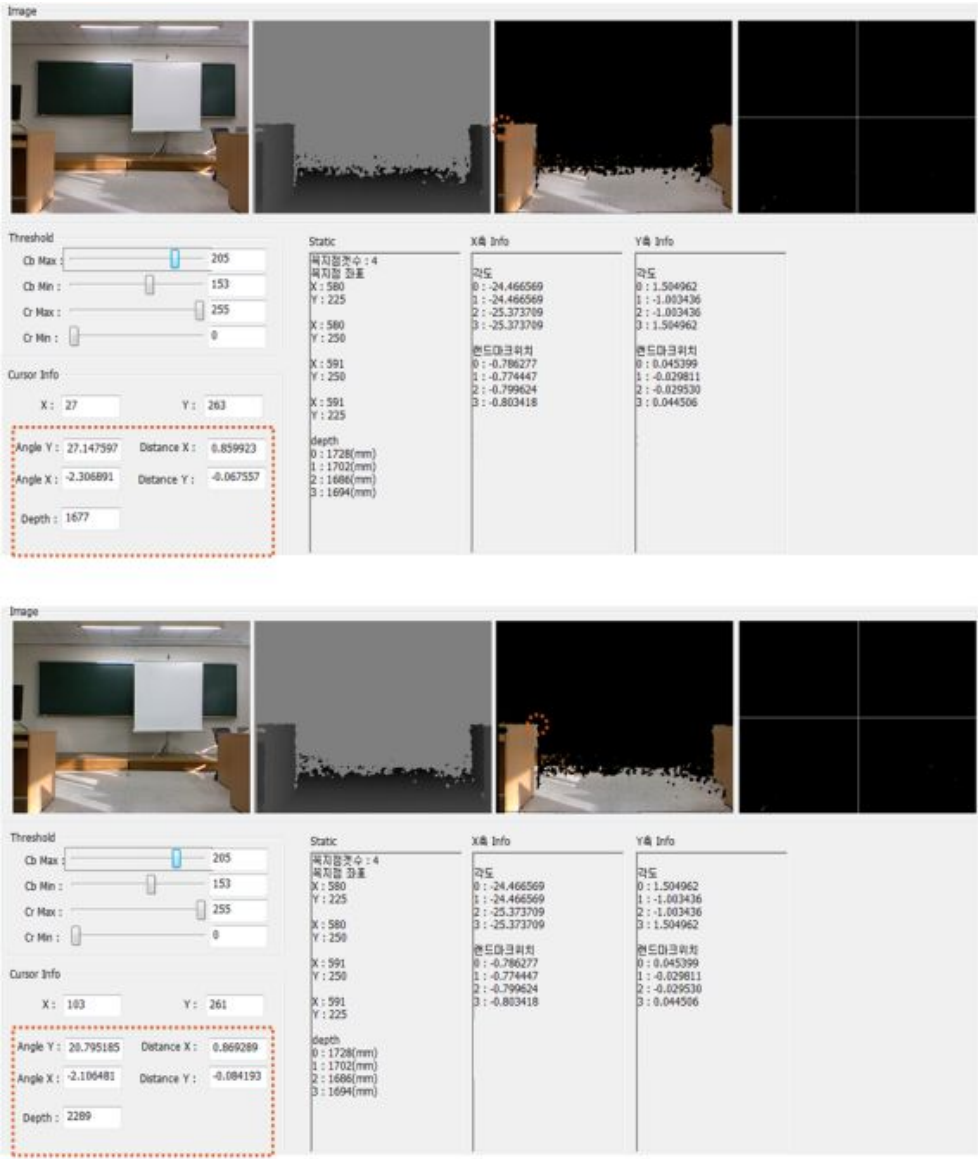


그림 4.29 마우스 포인터를 이용한 주변 환경의 특징점 정보

이와 같이 마우스 포인터를 이용하여 실제 거리와 방향이 측정된 정보와 비교할 수 있다. 또한 검출된 영상을 이용하여 주어진 특징점이 아닌 주변 사물에 대한 거리 및 방향 정보를 획득할 수 있다.

3. 이동 로봇의 주행중 특징점 검출

가. 실험 환경

본 실험은 고정된 위치에서 실험 장소와 동일한 장소에서 이루어졌다. 그림 4.30은 사용된 이동 로봇으로 레드원테크놀러지(주)에서 개발된 NRLAB-02 모델을 사용하였으며 그림 4.31과 같이 키넥트를 이동 로봇에 설치하여 실험하였다.

그림 4.30 NRLAB-02

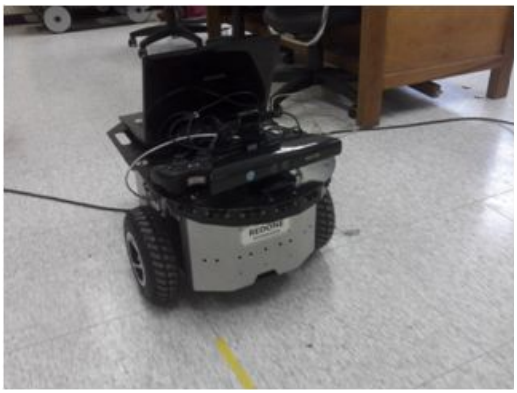


그림 4.31 이동 로봇에 설치된 키넥트

또한 그림 4.32는 주행 중 검출해야 할 특징점의 모양이다. 특징점의 정보는 표 5.1 과 같다.

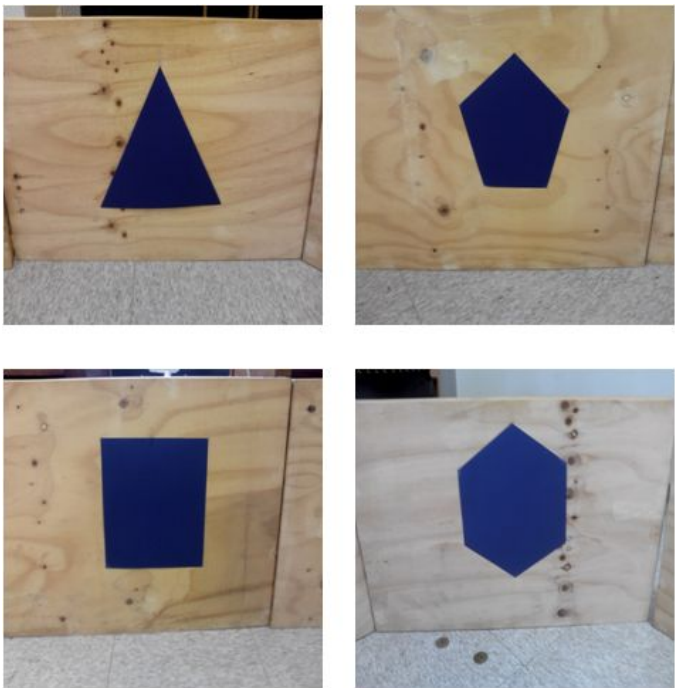


그림 4.32 특징점의 모양

표 4.1 특징점 정보

특징점의 모양	특징점의 개수	특징점의 한 변의 길이
정삼각형	2개	30cm, 40cm
정사각형	2개	30cm, 40cm
정오각형	2개	18cm, 25cm
정육각형	1개	19cm

그림 4.33과 같이 임의의 주행 환경을 설치하여 7개의 다른 모양과 크기에 대한 특징점을 부착하여 이동 로봇이 주행 중 특징점을 검출하여 정보를 획득할 수 있게 구성 하였다.

그림 4.33 주행 환경

그림 4.34는 이동 로봇의 진행 방향 및 특징점의 부착된 위치이며 주행 거리는 12m 이다.

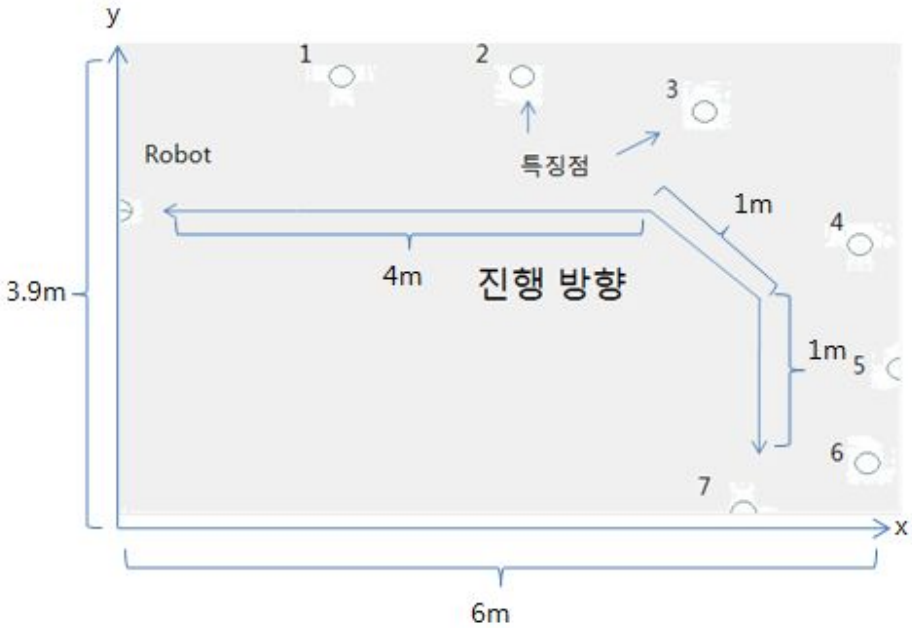


그림 4.34 주행 방향 및 특징점의 위치

그림 4.35는 주행중 특징점이 검출되는 모습이다. 그림에서 보이는 것과 같이 각각 다른 모양의 특징점을 검출하는 모습을 확인할 수 있다. 또한 각각의 다른 모양의 특징점에 대한 정보를 획득할 수 있다.

표 4.2는 이동 로봇의 초기 위치와 부착된 특징점의 위치를 나타낸다.

표 4.2 이동 로봇 및 특징점 초기 위치

	x	y
1	1.7m	3.90m
2	3.10m	3.90m
3	4.50m	3.60m
4	5.70m	2.40m
5	6.00m	1.30m
6	5.75m	0.45m
Robot	0.0m	2.70m

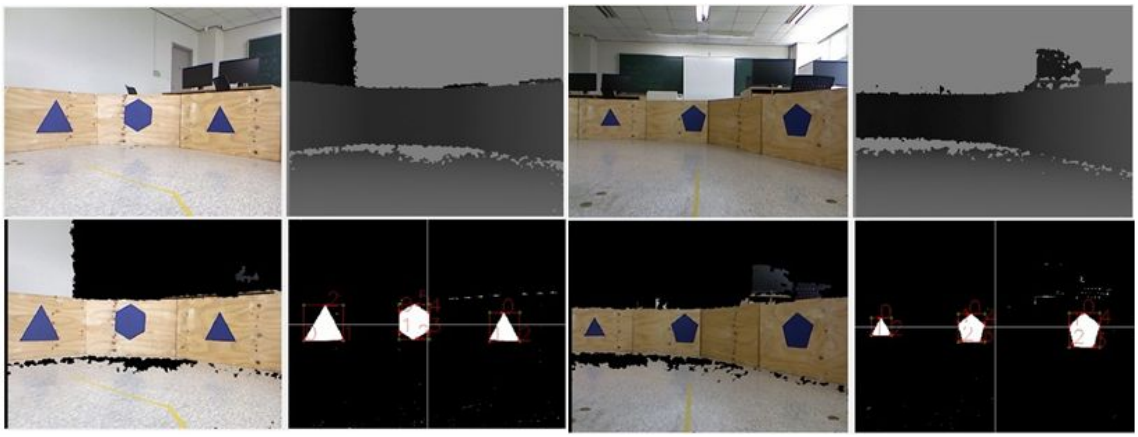


그림 4.35 주행중 특징검 검출

나. 거리에 따른 특징점 검출 정보

설치된 주행 환경에 이동 로봇 구동하여 특징점의 검출 및 정보 획득하였다. 그림 4.36은 원점에서의 특징점이 검출된 영상이며 그림 4.37는 측정된 특징점의 거리 및 방향 정보와 실제 거리 및 방향을 비교한 그래프이다. 그림에서 보이는 것과 같이 원점에서 하나의 특징점이 검출되었다.

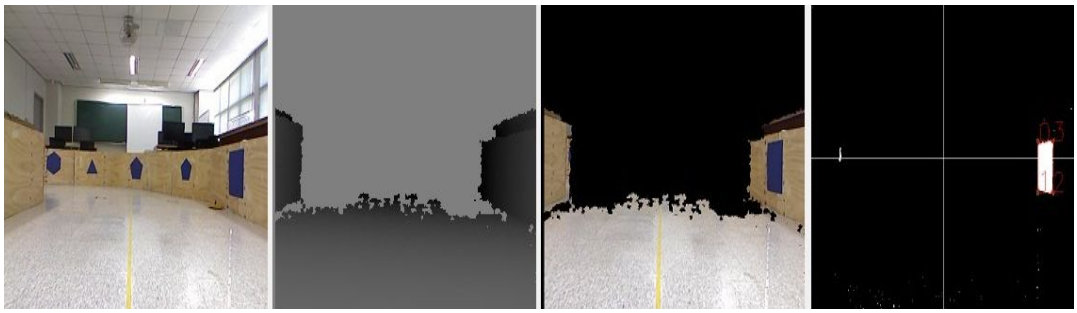


그림 4.36 주행 영상 및 특징점 검출 (원점)

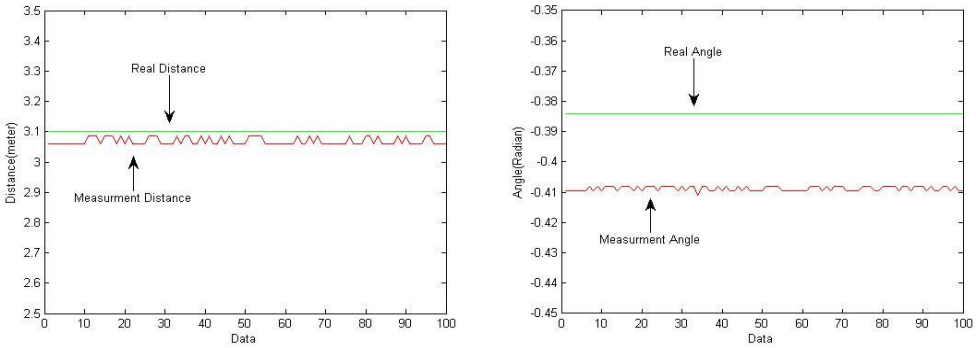


그림 4.37 측정된 특징점 정보 (원점) (좌: 거리(Meter), 우: 각도(Radian))

그림 4.38은 3미터 지점에서의 특징점이 검출된 영상이며 그림 4.39는 측정된 특징점의 거리 및 방향 정보와 실제 거리 및 방향을 비교한 그래프이다.

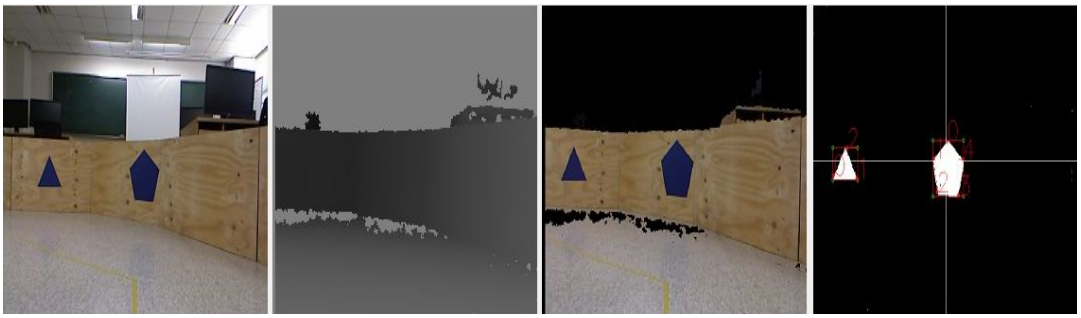


그림 4.38 주행 영상 및 특징점 검출 (3m)

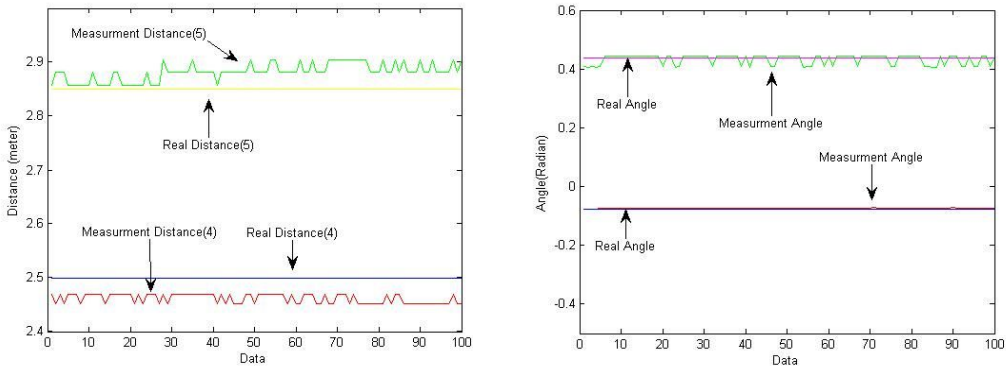


그림 4.39 측정된 특징점 정보 (3m) (좌: 거리(Meter), 우: 각도(Radian))

그림 4.40은 6미터 지점에서의 특징점이 검출된 영상이며 그림 4.41은 측정된 특징점의 거리 및 방향 정보와 실제 거리 및 방향을 비교한 그래프이다.

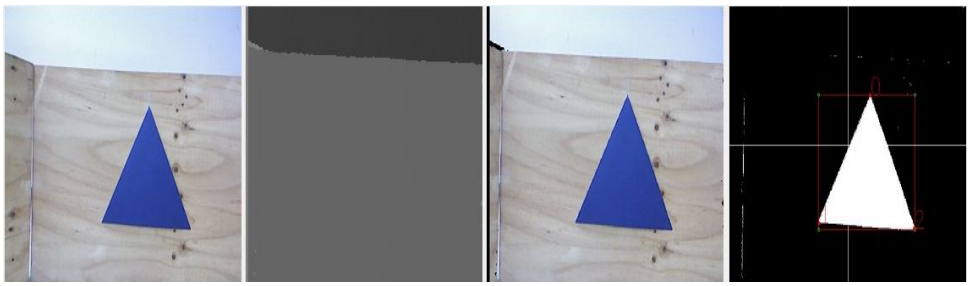


그림 4.40 주행 영상 및 특징점 검출 (6m)

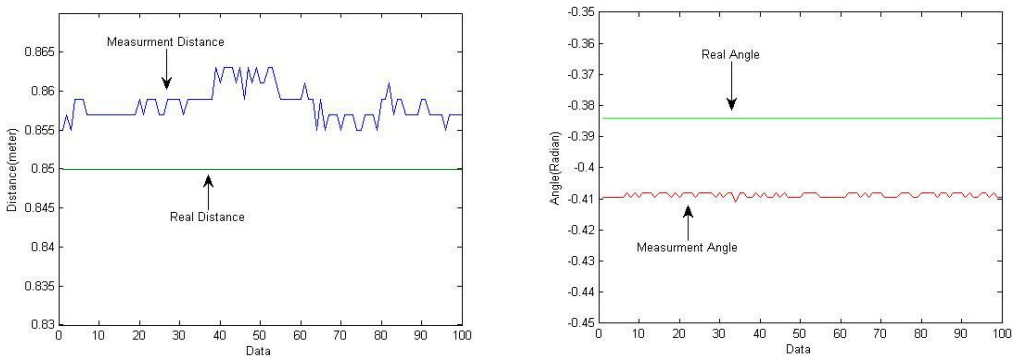


그림 4.41 측정된 특징점 정보 (6m) (좌: 거리(Meter), 우: 각도(Radian))

위 그림와 같이 이동 로봇의 주행중 특징점 검출 및 정보 획득에 대하여 실험한 결과 특징점의 모양과 크기를 구분하여 각각의 거리와 방향 정보를 획득 하는데 성공 하였다. 그림 4.42와 4.43은 5회 주행을 연속적으로 하여 특징점 1번부터 4번까지의 검출된 거리와 방향을 그래프로 표현한 모습이다. 그림에 표시된 부분과 같이 주변의 조명과 외부 노이즈에 따라 특징점이 검출되지 않고 정확한 정보를 획득할 수 없는 상황이 발생하는 경우가 있다. 따라서 노이즈에 대한 별도의 알고리즘이 적용되어야 한다.

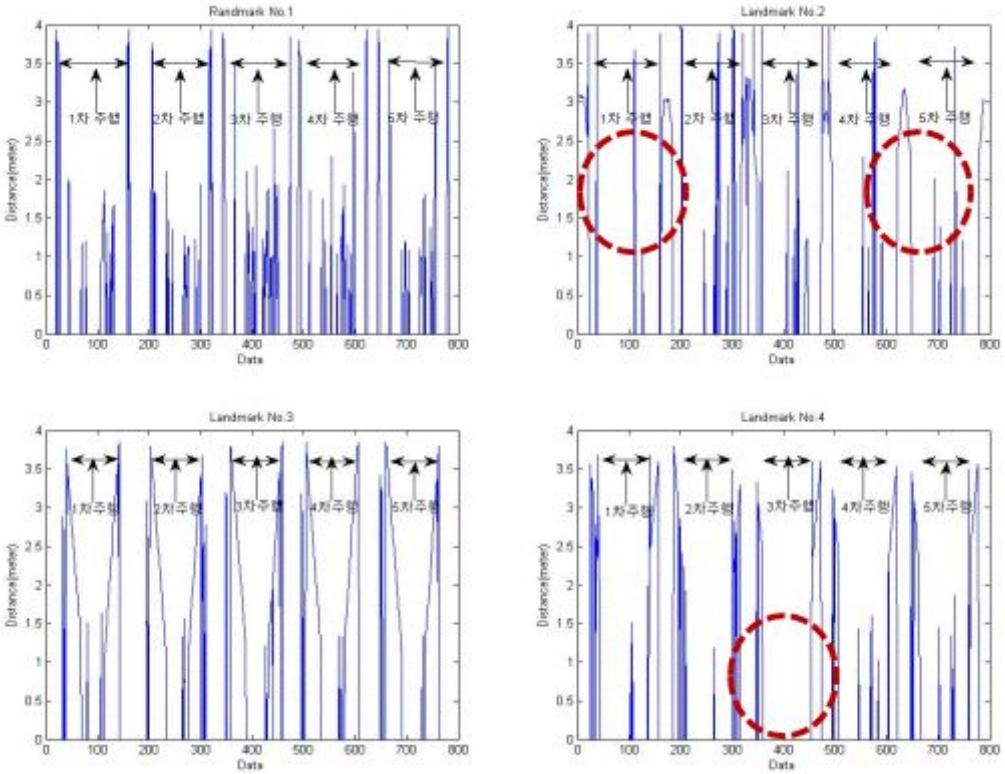


그림 4.42 주행중 검출된 특징점의 거리(1~4번)

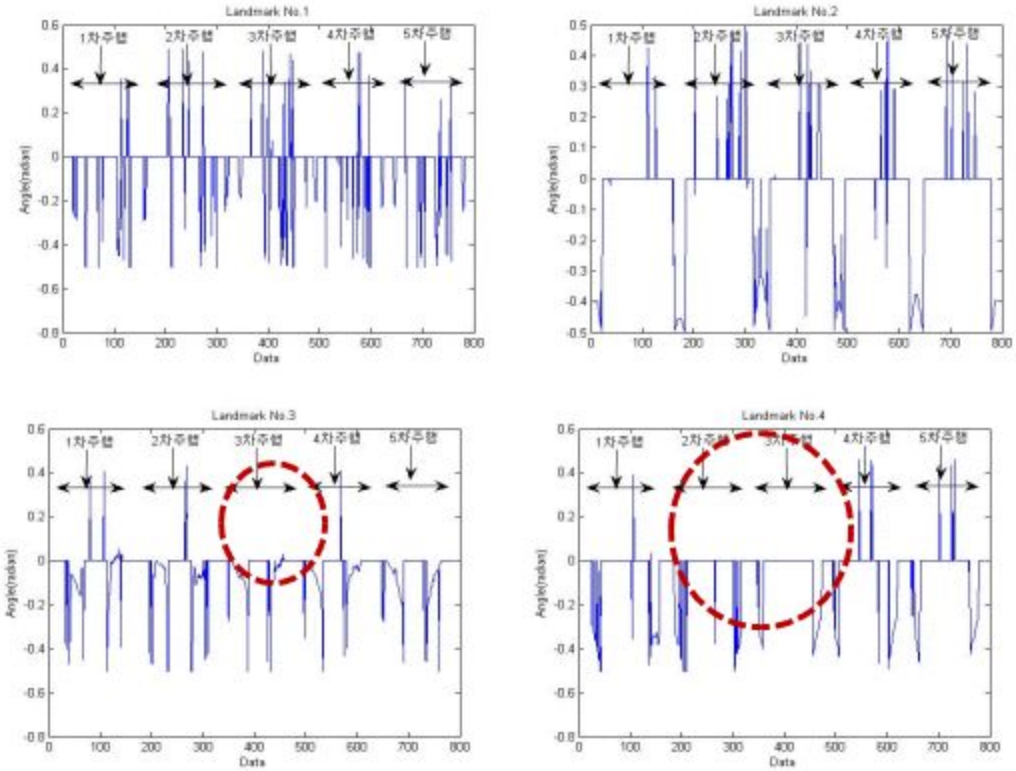


그림 4.43 주행중 검출된 특징점의 방향(1~4번)

제5장 결론 및 향후 계획

본 연구에서는 이동 로봇의 위치를 추정하기 위한 특징점 검출방법에 대해서 설명 하였다. 그 방법으로 영상과 거리 정보를 획득 하는 키넥트 센서를 사용 하였다. 또한 획득한 영상을 OpneCV 라이브러리를 이용하여 영상속에 비춰지는 특징점을 Ycbcr채널로 변경하여 특징점의 색을 검출였다. 또한 특징점의 모양을 구분짓기 위하여 레이블링과 컨벡스헐 기법을 사용하여 검출된 특징점의 꼭지점 좌표를 추출해 냈다. 그리고 획득한 특징점의 화면 좌표와 키넥트 Depth 정보를 이용하여 각각의 특징점의 거리와 방향을 획득하는데 성공하였다. 이렇게 특징점의 색과 모양을 검출하는데 성공을 하였으며 실제 특징점의 정보와 비교 하였을 때 뛰어난 정확성을 확인하였다. 그리고 본 논문에서는 푸른색 특징점을 사용하였으나 특징점의 임계값을 조절하게 되면 여러 가지 색도 검출이 가능하며 많은 특징점을 구성할 수 있게 된다. 그러나 영상에 비춰지는 특징점의 꼭지점을 검출할 때 영상의 가장자리에 검출 되면 정확한 특징점의 정보를 획득 하지 못할 뿐만 아니라 특징점의 왜곡되는 현상이 발생하게 된다. 또한 환경에 따른 조명으로 인해 검출 하고자 하는 특징점의 색 구별 방법으로 영상의 픽셀 임계값을 항상 조절해줘야 하는 단점과 노이즈로 인한 정확한 특징점의 모양과 크기를 계산하지 못하는 상황이 발생 하였다. 따라서 실험을 할 때 빛의 영향을 상대적으로 적게 받는 재질의 특징점을 선택하였다. 또한 외부의 조명을 차단하기 위해 블라인드를 사용하였다. 그리고 이동 로봇이 동작하면서 진동으로 영상에서 검출된 특징점의 꼭지점 좌표가 흔들려 오차가 발생 하였다. 따라서 영상을 이용한 특징점 검출 방법은 노이즈에 영향을 많이 받으므로 많은 연구가 필요하며 별도의 노이즈 제거 필터링을 할 수 있는 알고리즘이 구현되어야 정확한 특징점의 정보를 얻을 수 있다.

향후 계획으로 보다 정확한 특징점 정보를 획득하기 위한 알고리즘 연구 및 실험을 통해 구현할 것이다. 또한 실제 영상을 이용한 검출된 특징점 정보를 확장칼만필터 알고리즘에 적용하여 이동 로봇의 위치를 추정하는 연구를 진행할 것이며 영상뿐만 아닌 다른 센서를 사용하여 보다 높은 신뢰성을 갖는 센서 정보로 이동 로봇의 위치를 추정할 것이다.

참고문헌

[1] http://en.wikipedia.org/wiki/Robot

[2] 이동욱, "실내환경에서 비전과 초음파센서를 이용한 이동로봇의 위치추정", 석사학위논문, 2011

[3] R. Siegwart, and I. R. Nourbakhsh, “Autonomous Mobile Robots,” MIT Press, Cambridge, 2004.

[4]. S. Thrun, W. Burgard, and D. Fox, “Probabilistic Robotics,” The MIT Press,Cambridge, 2005.

[5] 엄기환,강성호,김주웅, "이동 로봇의 위치추정을 위한 변형된 칼만필터 방식", 한국해양정보통신학회논문지, 제12권, 제4호 pp.781-790

[6] 김지용,이지홍,변재민,김성훈, "경사도로에서 다중 센서를 이용한 이동로봇의 위치추정 성능 개선", 전자공학회지, 제47권, 제1호, pp.67-75

[7] 이연석, "칼만필터의 잔류오차에 최소적응알고리즘을 적용한 이동로봇의 위치추정오차 검출기법", 한국해양정보통신학회논문지, 제10권, 제7호, pp.1332-1337, 2006

[8] 최문규, "RF/초음파센서와 Global features를 이용한 고속 이동로봇의 위치추정기법", 석사학위논문, 2009

[9] 진태석,이장명,"이동물체의 기하학적 위치정보를 이용한 자율이동로봇의 위치추정", 한국퍼지 및 지능시스템학회지, 제14권 제4호 pp.438-444, 2004

[10] 박현석,"능동 스테레오카메라를 이용한 실시간 이동물체 추적 및 거리측정 시스템", 석사학위논문, 2006

[11] 이동석, 이동욱, 김수동, 김태준, 유지상, "스테레오 카메라를 이용한 이동객체의 실시간 추적과 거리 측정 시스템", 한국방송공학회지, 제14권, 제3호, pp.366-377, 2009

[12] 서강래, 강근택, 이원창, "거리 측정 센서와 단일 비전 시스템을 이용한 자율 주행", 한국지능시스템학회, 제19권, 제2호, 2009

[13] 김태균, 고낙용, 노성우 "파티클 필터 방법을 이용한 특징점과 로봇 위치의 동시 추정",한국지능시스템학회지 논문지, 제22권, 제3호, pp.353-360, 2012

[14] 김태균, "유비쿼터스 센서 환경에서의 이동로봇 위치추정", 석사학위논문,

2009
[15] Sebastian Thrun, Wolfram Burgard, Dieter Fox,"Probabilistic ROBOTICS", MIT press, 2005
[16] 문현수, 주영훈, "SURF와 Particle filter를 이용한 이동 로봇의 위치 추정", 한국지능시스템학회 논문지 제20권, 제4호,pp.586-591, 2010
[17] 강민구,"SURF 특징점 추출 알고리즘을 이용한 얼굴인식 연구", 석사학위논문, 2010
[18] 윤국진, 장기정, 김성호, 권인소,"이동 로봇을 위한 컬러 표식 기반 자기 위치 추정 기법", 제어로봇시스템학회논문지, 제7권, 제9호, pp.749-757, 2001
[19] 허동혁,"이동로봇을 위한 인공표식기반 위치추정시스템", 석사학위논문, 2011
[20] 강한구, 윤재오, 이지홍, "선택적 Anchors 기반 Indoor GPS 및 EKF를 이용한 이동 로봇 위치 추정", 로봇학회논문지, 제6권, 제1호, 2011
[21] 장철웅, "다중센서를 이용한 이동로봇의 강인한 자기 위치 추정에 관한 연구", 석사학위논문, 2008
[22] http://en.wikipedia.org/wiki/Kinect
[23] http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/2
[24] http://minsug.tistory.com/24
[25]http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx
[26] http://msdn.microsoft.com/en-us/library/hh973074.aspx
[27] http://www.slideshare.net/opendori/nui-kinect
[28] http://en.wikipedia.org/wiki/RGB_color_model
[29] 박은진, "방향성 레이블링을 이용한 효과적인 코너 및 직선 검출", 박사학위논문, 2011
[30] http://en.wikipedia.org/wiki/Connected-component_labeling
[31] 신현덕, 전영철,"다방향 선형 스캐닝과 컨벡스 헐을 이용한 아무르불가사리의 특징 추출", 한국컴퓨터정보학회지, 제16권, 제3호, pp.99-107, 2011
[32] 박희진, "핀홀 카메라의 광학적 메카니즘에 관한 연구", 현대사진영상학회논문집, 제1권, pp.145-157, 1998
[33] http://en.wikipedia.org/wiki/Pinhole_camera

[34] 박세환, 송혁, 장은영, 허남호, 김지웅, 김진수, 이상훈, 유지상,"다시점 동영상에서 임의시점영상 생성을 위한 가려진 영역 보상기법". 한국통신학회논문지, 제33권, 제12호, pp.1029-1038
[35] http://www.microsoft.com/en-us/kinectforwindows/

감사의 글

2011년 대학원에 입학하고 벌써 2년이 지나 졸업을 앞두고 있습니다. 2년 동안 많은 일과 경험을 통하여 배운 것이 참 많았습니다. 본 논문은 그동안 해보지 못했던 영상을 다루는 내용이라 많이 힘들고 어려웠습니다. 이렇게 부족한 저를 지도 해주시고 같이 고민해주신 저의 지도교수님이신 고낙용 교수님께 정말 감사드립니다. 하지만 교수님께서 가르쳐주신 만큼 열심히 하지 않은 것 같아서 정말로 죄송한 마음이 가득 합니다. 그리고 지도교수님 만큼 많은 조언을 해주신 최한수 교수님, 이진이 교수님, 장순석 교수님, 반성법 교수님, 조창현 교수님, 곽근창 교수님께 정말로 감사드립니다.

2년동안 함께 생활한 태균이형과 성우형은 많은 것을 알려주고 지적해 주셨지만 좋지 않은 모습만 보여드린 것 같아서 죄송한 마음만 가득합니다. 그리고 비록 잘하진 못했지만 끝까지 챙겨주시고 믿어주셔서 항상 고마운 마음을 갖고 있습니다. 그리고 졸업하진 동진이형, 광진이형 나이차이가 많이(?) 나서 어려움만 가득했는데 항상 친동생 처럼 대해주시고 많은 노하우를 알려주셔서 감사합니다. 그리고 좋은 취업 자리 있으면 소개 좀 시켜주세요. 그리고 영남이형 취업했지만 자주 와서 맛있는것도 사주고 좋은 경험도 시켜주시고 정말로 감사드립니다. 다음에 서울에 가면 또 맛있는거 많이 사주세요. 그리고 하나밖에 없는 후배 석기 서로 교대로 서울로 공부하러 가서 같이 지낸 시간은 매우 짧았지만 그래도 나를 잘 대해줘서 고맙다. 나보다 더 잘 하겠지만 항상 더 열심히 하고 좋은 성과가 있길 바란다.

그리고 저의 하나밖에 없는 대학원 동기 김민구, 넌 항상 나를 귀찮게 했지만 그래도 동기가 있었다는게 얼마나 다행인지 모른다. 그리고 같은 취미를 갖고 있어서 서로 말도 잘 통했던거 같고 너 때문에 많이 힘든 시간을 즐겁게 보낼 수 있었다. 이제 너도 졸업이니 좋은 직장 얻어서 출세하자.

그리고 지금까지 믿어주시고 가장 큰 힘이 되셨던 저희 부모님 그리고 형에게 정말로 감사한 마음과 죄송스러운 마음만 갖고 있습니다. 비록 못난 자식이지만 학업에 충실할 수 있도록 새벽 마다 일어나셔서 출근하시고 밤 늦게 퇴근하셔 항상 마음이 좋지 않습니다. 하루라도 빨리 좋은 곳에 취직하여 효도 하겠습니다. 정말로 감사드립니다. 그리고 어머니 지금 사고로 병원에 입원 하셨는데 빨리 완치 되셨으면 좋겠습니다. 그리고 다음부터 다신 그런 사고가 일어나지 않도록 항상 기도할께요. 마지막으로 형, 하고 있는 공부 잘 준비해서 원하는 성과를 이뤘으면 하는 마음으로 항상 응원 하고 있을게 파이팅~ 비록 앞으로 해야 할 일과 더 배워야 할 일도 많은 상태이지만 조금 더 발전하고 멋진 아들이 될 것을 약속드리며 이 논문을 바칩니다. 감사합니다 그리고 사랑합니다.

2012.12

저작물 이용 허락서

학 과	제어계측공학과	학 번	20117092	과 정	석사
성 명	한글: 나 정 현 한문 : 羅 正 賢 영문 : Na Jung Hyun				
주 소	광주 광역시 북구 두암3동 주공아파트 104동1406호				
연락처	E-MAIL : chaplin2004@naver.com				
논문제목	한글 : 위치 추정을 위한 영상 특징점 정보 검출 방법 영어: A Detection Method of Visual Landmark Information Mobile Robot Localization				

본인이 저작한 위의 저작물에 대하여 다음과 같은 조건아래 조선대학교가 저작물을 이용할 수 있도록 허락하고 동의합니다.

- 다 음 -

1. 저작물의 DB구축 및 인터넷을 포함한 정보통신망에의 공개를 위한 저작물의 복제, 기억장치에의 저장, 전송 등을 허락함
2. 위의 목적을 위하여 필요한 범위 내에서의 편집·형식상의 변경을 허락함. 다만, 저작물의 내용변경은 금지함.
3. 배포·전송된 저작물의 영리적 목적을 위한 복제, 저장, 전송 등은 금지함.
4. 저작물에 대한 이용기간은 5년으로 하고, 기간종료 3개월 이내에 별도의 의사 표시가 없을 경우에는 저작물의 이용기간을 계속 연장함.
5. 해당 저작물의 저작권을 타인에게 양도하거나 또는 출판을 허락을 하였을 경우에는 1개월 이내에 대학에 이를 통보함.
6. 조선대학교는 저작물의 이용허락 이후 해당 저작물로 인하여 발생하는 타인에 의한 권리 침해에 대하여 일체의 법적 책임을 지지 않음
7. 소속대학의 협정기관에 저작물의 제공 및 인터넷 등 정보통신망을 이용한 저작물의 전송·출력을 허락함.

동의여부 : 동의(○) 반대( )

2013년 2월

저작자: 나 정 현 (서명 또는 인)

조선대학교 총장 귀하