**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | Depth map 을 활용한 후방카메라의 한계 개선 |
| 팀 명 | 천 리 안 |
| 문서 제목 | 중간보고서 |

|  |  |
| --- | --- |
| **Version** | 1.5 |
| **Date** | 2015-04-9 |

|  |  |
| --- | --- |
| **팀원** | 최승혁 |
| 신태섭 |
| 박성우 |
| 신동호 |
| 최성현 |
| 박민욱 |
| **지도교수** | 한재일 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “Depth map을 활용한 후방카메라의 한계 개선”를 수행하는 팀 “천 리 안”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “천 리 안”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 중간보고서-Depth map을 활용한 후방카메라의 한계 개선 |
| **원안작성자** | 최승혁, 신태섭, 박성우, 신동호, 최성현, 박민욱 |
| **수정작업자** | 최승혁, 신태섭, 박성우, 신동호, 최성현, 박민욱 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2015-03-20 | 최성현 | 1.0 | 최초 작성 |  |
| 2015-04-03 | 신태섭 | 1.1 | 내용 작성 | 수행 내용 및 중간결과 작성 |
| 2015-04-05 | 신태섭  박민욱 | 1.2 | 내용 작성 | 향후 추진 계획 작성 |
| 2015-04-07 | 최승혁  박성우 | 1.3 | 내용 수정 | 수행 내용 및 중간결과,  향후 추진 계획수정 |
| 2015-04-08 | 신동호  최성현 | 1.4 | 내용 수정 | 전체 내용 수정 확인 및 오타 작업 |
| 2015-04-08 | 팀원 전원 | 1.5 | 최종 확인 |  |
|  |  |  |  |  |

**목 차**

[1 프로젝트 목표 4](#_Toc411285431)

[2 수행 내용 및 중간결과 5](#_Toc411285432)

[2.1 계획서 상의 연구내용 5](#_Toc411285433)

[2.2 수행내용 5](#_Toc411285434)

[3 수정된 연구내용 및 추진 방향 6](#_Toc411285435)

[3.1 수정사항 6](#_Toc411285436)

[4 향후 추진계획 7](#_Toc411285437)

[4.1 향후 계획의 세부 내용 7](#_Toc411285438)

[5 고충 및 건의사항 8](#_Toc411285439)

# 프로젝트 목표

1. Depth map 정보를 이용하여 자동차 운전자에게 자동차 후방과 장애물 사이의 직관적인 깊이 정보를 제공할 수 있는 영상처리 알고리즘을 구현한다.
2. 후방 영상카메라와 Depth map을 측정하는 Kinect의 Depth map 카메라의 영상을 Calivration하여 두 영상 사이의 이질감을 최소화한다.
3. 모형 자동차를 통하여 Depth map을 활용한 후방카메라의 기능 테스트를 가능하게 한다.
4. 기존의 후방카메라와 초음파 센서의 한계점인 거리감의 부재와 사각지대의 존재를 Depth map을 이용하여 개선한다.
5. Depth map 정보를 활용하여, 모형차량과 장애물 사이의 충돌 가능성이 높아진 경우 사용자에게 위험을 알리고, 모형차량에 제동을 하는 제어 알고리즘을 구현한다.

# 2. 수행 내용 및 중간결과

# 2.1 계획서 상의 연구내용

# 2.1.1 Depth map

해상도 512 \* 424 정도의 비교적 고화질의 depth 영상을 출력하는 것을 목표로 Kinect부터 depth 영상을 제공받는다. Kinect를 pc와 연결하고, Kinect를 구동하기 위한 SDK를 설치한다. Kinect studio를 실행시키면, 기존의 카메라 영상 (color image)와 depth image를 함께 얻을 수 있다. 이 두 영상을 가지고 정확히 일치하도록 합성해야 한다.

****

# 2.2.2 Calibration

위에 제시된 그림이 Kinect 를 pc에 연결하고, Kinect studio를 실행하여 얻은 depth 영상과 color 영상을 캡쳐한 모습이다. 이제 이 두 개의 영상을 합성하는 작업을 해야 한다. 그런데 kinect의 카메라 중에서 color 영상을 촬영하는 카메라와 depth 영상을 촬영하는 카메라의 위치가 약간 떨어져 있기 때문에, 같은 곳을 촬영하더라도 두 영상의 위치가 완벽히 일치하지 않게 된다. 따라서 두 영상을 정확히 일치하도록 Calibration 알고리즘을 적용하여 깔끔하게 합성하는 작업이 필요하다. 그리고 depth 영상에서 나온 물체의 테두리가 깔끔하게 촬영되지 않는(데이터가 튀는) 현상이 종종 발생하게 된다. 이를 해결하기 위해서도 보정하기 위한 알고리즘을 정확하여 좀 더 정밀하고 정확한 영상을 획득할 수 있도록 해야 한다.

# 2.2.3 영상처리

합성한 영상에서 물체와 카메라 사이의 거리에 따라 하이라이트 표시를 해야 한다. 운전자에게 물체와 자동차 사이의 거리 정보를 확실하게 제공할 수 있고, 시각정보를 기존의 카메라보다 눈에 띄게 나타낼 수 있기 때문이다. Visual Studio 2012를 이용하여, 합성된 영상에서 일정 거리 이하의 물체에 도형이나 색깔로 표시를 하여 영상 처리를 한다. 그리고 필요하다면, 하이라이트 표시 이외에 경고음을 발생시켜서 운전자에게 확실히 각인시켜 줄 수 있는 방법도 생각해 볼 것이다.

# 2.2.4 자동차 제어

Kinect로부터 획득한 Depth정보를 이용하여 모형차량 제어를 해야 한다. 따라서, Kinect와 연결되어 있는 PC와 모형차량 간의 통신이 가능해야 하는데 그러기 위해선 먼저

모형차량의 전체적인 제어를 할 수 있는 Raspberry Pi를 설정해야 한다. Raspberry Pi설정을 단계별로 나타내면 다음과 같다.

1. SD카드에 OS 설치

Raspberry Pi에서 사용하는 운영체제는 일반적으로 Raspbian을 사용한다. Raspbian은 Debian linux가 Raspberry Pi용으로 포팅 된 운영체제이다.

1. Wireless 네트워크 환경 설정

운영체제가 설치되면 네트워크 설정을 해야 한다. Wireless환경을 설정하기 위해서 Wifi동글을 이용하거나 블루투스 모듈을 사용해야 한다. 데이터 전송속도를 고려하여 Wifi동글을 이용하여 설정한다.

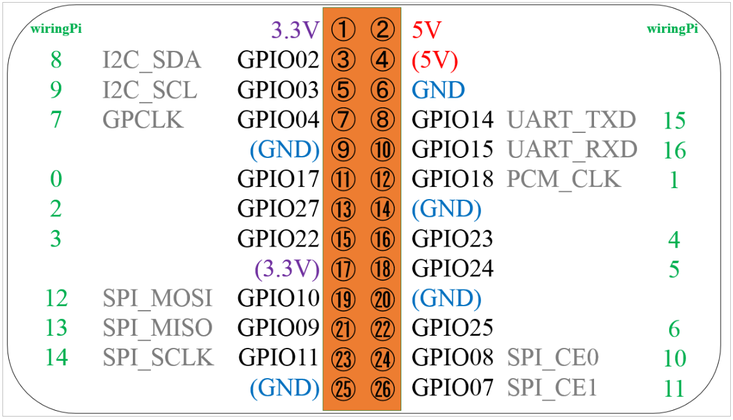
1. Client/Server 모델을 적용할 언어 선정

통신을 위하여 네트워크모델을 정해야 한다. PC를 Server로, Raspberry Pi를 Client로 하여 설정하고, 이 모델을 적용할 언어를 설정한다. 개발언어는 C언어, Java등을 사용 할 수 있다.

또한, Raspberry Pi와 모형차량의 각 모듈의 실질적인 제어를 하는 Micro controller와 UART를 통신을 위해 GPIO핀을 사용하기 위한 설정을 해야 한다. 이를 단계별로 나타내면 다음과 같다.

1. Wiring Pi 모듈 설치

Wiring Pi 모듈은 Raspberry Pi의 GPIO핀에 접근할 수 있게 하는 C언어로 작성된 라이브러리들을 사용할 수 있게 한다. Wiring Pi는 아두이노에서 사용하는 Wriring system과 유사하기 때문에, 쉽게 사용이 가능하다.



위의 GPIO그림과 같이 GPIO14, 15번 핀에 Wiring Pi 라이브러리를 이용하여Controller와 값을 주고 받을 수 있다.

1. Wiring Pi 라이브러리를 이용하여 차량 제어 프로그래밍

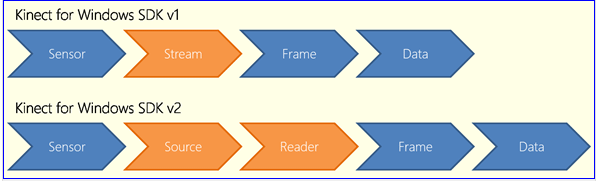
앞서 선택한 개발 언어를 이용하여 알고리즘을 구현한다. 이 프로그램에는 Wiring Pi 라이브러리와 서버/클라이언트 라이브러리를 사용한다. 이를 통해 PC에서 Depth map 정보를 이용하여 해당하는 제어를 Raspberry Pi로 전송할 수 있고, 해당하는 제어에 맞도록 차량을 구동할 수 있게 된다.

## 2.2 수행내용

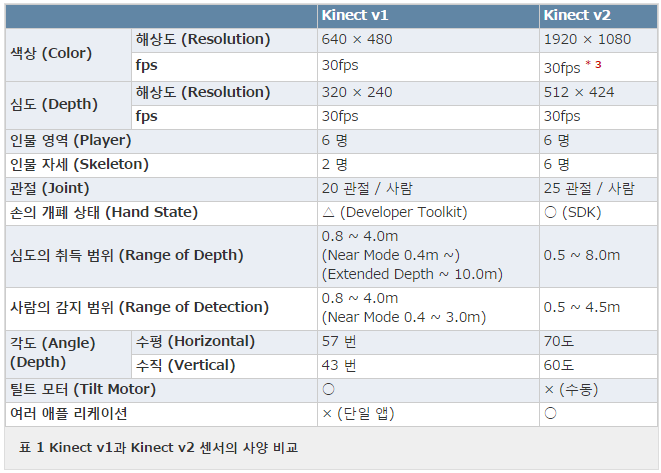
# 2.2.1 Depth map

**1. 구현 진행 상황**

기존 Kinect v1 for windows를 사용하려고 하였으나 야외에서의 Depthmap 획득이 어렵다는 지적받아 그런 문제점을 해결하기 위해 Kinect v2 for windows로 장치를 교체하였다. 그리고 Kinect v2 for windows를 사용해 보니 Kinect v1 for windows보다 더 선명하고 정밀한 depth map을 획득 할 수 있었다. Kinect v1 for windows 와 Kinect v2 for windows의 윈도우 환경에서 장치를 구동할 수 있게 해주는 SDK의 버전이 다르기 때문에 기존 SDK 1.8에서 2.0 버전으로 교체하였다**.**

****

Kinect SDK v1 Sensor 에서 Stream 에서 Frame 을, Frame에서 Data를 검색하는 흐름이다. Kinect SDK v2 Sensor 에서 얻은 Source 에서 Reader 를 열고, Reader 에서 Frame 을, Frame에서 Data를 검색하는 흐름이다. Kinect SDK v2 Source 는 Kinect에서 Color 와 Depth 데이터의 흐름마다 존재 한다. 이것은 Kinect SDK v1 의 Stream 과 동일 하다.



1. **예정 일정과 진도비교**

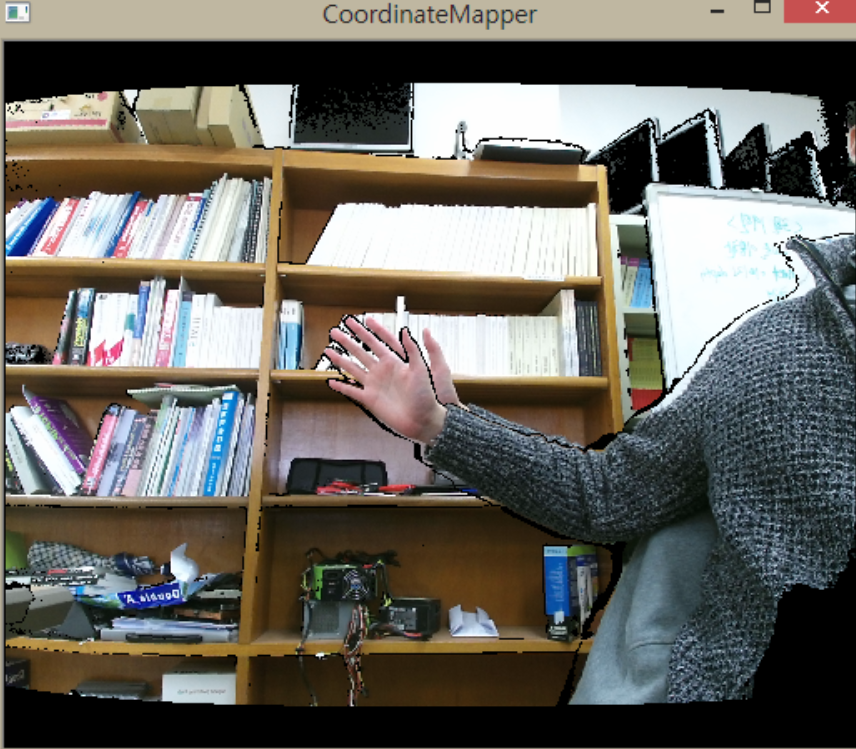
SDK가 교체 됨에 따라 관련 함수도 많이 바뀌었다. 그래서 여러 open소스와 Documentation을 참고하여 depth map 과 image 영상을 출력하는 코드를 작성 하였다. 예정 일정과 비교하여 큰 무리 없이 일정에맞게 구현을 마칠 수 있었다.

# 2.2.2 Calibration

**1. 구현 진행 상황.**

우리 조에서 구현할 영상을 얻기 위해서는 image 영상과 depth map을 합성하여야 한다. 그런데 Kinect에서 image 영상을 출력해주는 센서와 depth map을 출력해 주는 센서가 각각 따로 존재 하는데, 이 센서 사이에 약간의 거리가 있다. 그러므로 단순히 image와 depth를 합성하면 두 영상 사이에 좌표 값의 차이가 발생하게 된다. 따라서 그 차이를 보정하기 위한 작업이 필요했다. image와 depth의 좌표 값을 일치시키기 위해서 image 쪽의 좌표를 depth 쪽으로 맞추는 방법을 선택하였다. 왜냐하면 image 영상의 크기가 depth 영상의 크기보다 더 크기 때문에, 큰 부분에서 작은 부분으로 변환하여 맞추는 것이 반대의 경우보다 영상이 깔끔하게 출력되기 때문이다. Kinect SDK 2 참조하여 좌표 값을 일치시키는 것과 관련된 함수를 사용하여 calibration 작업을 진행하였다.





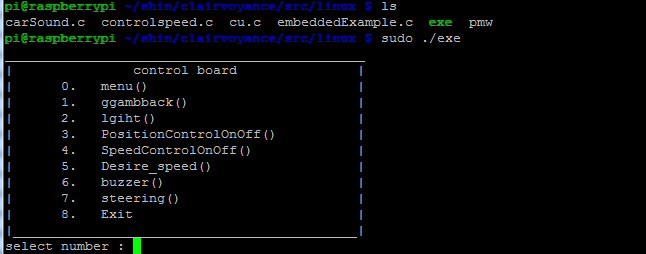
**2. 예정 일정과 진도비교**

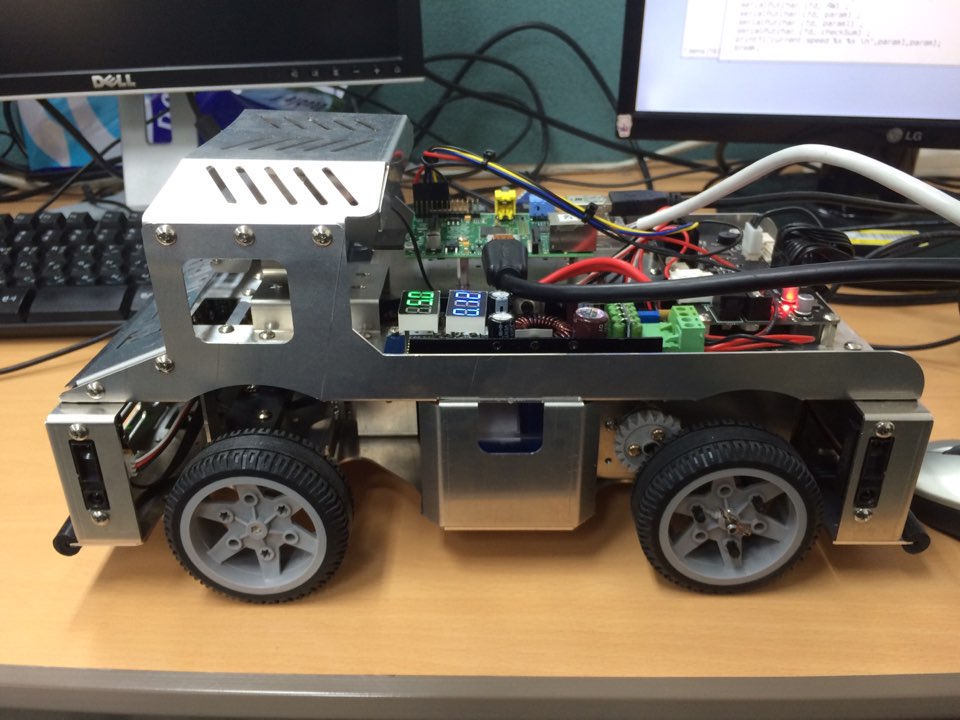
처음에 계획서에서도 image와 depth를 합성하여 calibration하는 작업까지 구현하기로 계획했었다. 실제 구현된 내용도 calibration까지 마쳤다. 한 가지 아쉬운 점이 있다면, 완전하게 image와 depth의 좌표 값이 정합되지 않고 약간의 오차가 발생하였다는 점이다. 이 부분은 추후에 더 개선해보도록 할 예정이다.

# 2.2.3 자동차 제어

**1. 구현 진행 상황.**

기존 모형 자동차에 설치된 Raspberry Pi에 Rasbian OS 와 Wiring Pi 를 설치하여 프로그래밍을 할 수 있는 환경을 구축하였다. 구축된 환경에서 모형 자동차를 제작한 ㈜미니로봇 에서 제공하는 무인자동차 명령 코드를 이용하여 차량의 전진 후진, 스티어링, 부저, 경고등, 등의 제어를 Keyboard로 controll 하는 환경을 구축하였다. 아래 그림과 같이 각 제어 기능을 Keyboard로 제어할수 있다.



****

**2. 예정 일정과 진도비교**

현재 모형 자동차의 Raspberry Pi에서 구현된 차량 제어 프로그램에 통신을 할 수 있는 모듈을 추가하여 Client 프로그램으로 구현하고, Kinect 영상처리 작업을 수행하는 PC에는 Server 프로그램을 구현하여 두 시스템 간의 소켓 통신을 할 수 있는 환경을 구축하는 것이 목표이다. 이후, PC를 통해 모형 자동차를 제어하고, Kinect로부터 얻은 정보를 바탕으로 자동차와 후방 장애물과의 거리가 일정 거리 미만이 되면 자동차의 움직임을 제어하는 환경을 구축하는 것을 목표로 한다.

# 3. 수정된 연구내용 및 추진 방향

## 수정사항

1. 기존 Kinect v1 for windows를 사용하려고 하였으나 야외에서의 Depthmap 획득이 어렵다는 지적 받아 그런 문제점을 해결하기 위해 Kinect v2 for windows로 장치를 교체하였다.
2. Kinect v1 for windows 와 Kinect v2 for windows의 윈도우 환경에서 장치를 구동할 수 있게 해주는 SDK의 버전이 다르기 때문에 기존 SDK 1.8에서 2.0 버전으로 교체하였다.

# 향후 추진계획

## 향후 계획의 세부 내용

**4.1.1 Kinect와 모형자동차의 네트워크 연결**

Raspberry Pi에서 구현된 차량 제어 프로그램에 통신을 할 수 있는 모듈을 추가하여 Client 프로그램으로 구현하고, Kinect 영상처리 작업을 수행하는 PC에는 Server 프로그램을 구현하여 두 시스템 간의 소켓 통신을 할 수 있는 환경을 구축한다. 이후, PC를 통해 모형 자동차를 제어하고, Kinect로부터 얻은 정보를 바탕으로 자동차와 후방 장애물과의 거리가 일정 거리 미만이 되면 자동차의 움직임을 제어한다.

**4.1.2 마일스톤**

|  |  |  |  |
| --- | --- | --- | --- |
| **마일스톤** | **개요** | **시작일** | **종료일** |
| 2 차 중간 보고 | - 모형차와 컴퓨터 통신 완료  - 모형차와 Kinect 연동 완료  **산출물:**  1. 프로젝트 2차 중간 보고서  2. 프로젝트 진도 점검표  3. 2차분 구현 소스 코드 및 하드웨어 | 2015-04-09 | 2015-05-13 |
| 구현완료 | 시스템 구현 완료  **산출물:**  1. 구현 완성 프로젝트 결과물 | 2015-04-09 | 2015-05-21 |
| 테스트 | 시스템 통합 테스트  **산출물:**  1. 오류 수정 목록  2. 프로젝트 수정 결과물 | 2015-05-21 | 2015-05-26 |
| 최종 보고서 | 최종 보고 | 2015-05-13 | 2015-0528 |

# 고충 및 건의사항