

Bachelor's Thesis

# **A Study of Sensor Calibration for Mounting Inspection of ADAS Camera and LiDAR**

School of Mechanical and Control Engineering  
Handong Global University

Ye-Ham Kim

Ye-Eun Lee

# **A Study of Sensor Calibration for Mounting Inspection of ADAS Camera and LiDAR**

A Bachelor's Thesis

Submitted to the School of  
Mechanical and Control Engineering of  
Handong Global University

Ye-Ham Kim

Ye-Eun Lee

December 2019

This certifies that the bachelor's thesis is approved.

---

Thesis Advisor: Ph.D. Yeong-Keun Kim

---

The Dean of Faculty: Ph.D. Kwon-Yeong Lee

School of Mechanical and Control Engineering

Handong Global University

December 2019

## CONTENTS

<b>Extended Abstract .....</b>	<b>i</b>
<b>1. 서 론.....</b>	<b>3</b>
1.1 연구목표.....	3
1.2 연구배경.....	3
1.3 연구요약.....	5
<b>2. 연구 방법.....</b>	<b>6</b>
2.1 좌표계 정의.....	6
2.2 LiDAR 포인트 클라우드 상의 타겟 특징점 추정 .....	8
2.3 카메라 이미지 상의 타겟 특징점 추정 .....	11
2.4 상대위치 및 변환행렬 추정.....	12
<b>3. 연구 결과.....</b>	<b>16</b>
3.1 시뮬레이션 결과.....	16
3.2 실험 결과.....	17
<b>4. 토 의.....</b>	<b>20</b>
4.1 시뮬레이션 결과 분석.....	20
4.2 실험 결과 분석.....	21
<b>5. 결 론.....</b>	<b>22</b>
<b>6. References.....</b>	<b>23</b>

# Extended Abstract

Development of technologies for driving assistance and self-driving systems is essential as many automobile companies are required to equip ADAS. Camera and LiDAR sensor are key components of ADAS, and researches are underway on sensor fusion technology that uses information from various sensors together. Calibration is the process of obtaining location information from different sensors, i.e. the conversion matrix between sensors, which is essential for multiple sensors in a vehicle to recognize the driving environment equally. In this study, we will conduct research for more accurate calibration of RGB cameras and 3D-LiDAR (Light Detection and Ranging) sensors.

Generally, camera – LiDAR calibration technique uses planar targets such as planar boards with black circle, chessboard and polygonal planar boards for calibration targets. However, planar calibration is affected by the resolution of LiDAR when detecting edges of planar board, and it results in a lot of error in calculating extrinsic parameters. Recently, a research was published which uses 3D target and shows more accurate result in estimation of relative pose [1]. In this paper, the calibration algorithm will be implemented using a cube target. In addition, a synthesis test by simulation and real-world experiment will be performed to validate the proposed algorithm. Finally, based on the result, the applicability will be reviewed of the automobile production line for mounting inspection of camera – LiDAR.

To find relative pose and transformation matrix between LiDAR and camera, the process of calculating 7 vertexes from each sensor data is preceded. Because it is based on side detection method, the perpendicular characteristic of a 3D cube can be used. The algorithm will estimate the coordinate of vertexes on LiDAR point cloud data. Firstly, RANSAC algorithm is applied to ROI (Region of Interest) data of LiDAR data and detects three planes which have most inlier among random planes. Using normal vectors of three planes and the size of cube target, 3D coordinate of 7 vertexes can be estimated. In this research, camera should be pre-calibrated which means the intrinsic parameter of camera is already known. Zhang's method [13] was used for camera calibration, and intrinsic parameter matrix was obtained. After filtering by 'sobel' filter on camera image, 'hough lines' were detected which are 8 edges of the target. The coordinate of 7 vertexes on image plane can be calculating by finding intersections of each edges of target.

After finding 3D and 2D coordinate of target, these points are input to EPnP algorithm [12] which calculates coordinate transformation matrix between camera and LiDAR. EPnP algorithm shows more accurate result than other solutions of PnP problem in calculating transformation matrix. Thus, the goals of this research are to implement semi-automatic calibration algorithm and to achieve accurate camera – LiDAR calibration in experiment.

For simulation, LiDAR point cloud and camera image about a cube target were obtained in virtual environment using ‘Blender’ program. After applying the algorithm on 10-frame of LiDAR point cloud and one image, the average estimation of relative pose was analyzed when Gaussian noise ( $\sigma = 0.02[m]$ ) is added on point cloud. In experiment, Velodyne VLP-16 and Microsoft HD camera were used to verify the algorithm. Experimental environment was configured with a cube target of  $500 \times 500 \times 500$  [mm]. 6 relative poses were considered in experiment, and the average of pose estimation results about 30 scans of each case was obtained by applying the proposed algorithm.

Both simulation and experiment results were analyzed in section 4. Relative pose estimation by proposed algorithm showed more accurate results than other calibrations which use planar targets. The result of proposed algorithm is transformation matrix which synchronize the coordinates of each sensor, LiDAR and camera. This transformation matrix is essential information for implementing sensor fusion algorithm. Therefore, the transformation matrix obtained by proposed algorithm can be utilized to map LiDAR point cloud on camera image. In vehicles with various sensors, such as self-driving cars, driving environment can be recognized by different kind of sensors with coordinate transformation matrix. In addition, the proposed algorithm can be used in production line in car factories to inspect the mounting position of each sensor.

# 1. 서 론

## 1.1 연구목표

본 연구의 목적은 기존 캘리브레이션 연구와는 다르게 3차원 큐브를 타겟으로 사용하여 LiDAR와 카메라 사이의 캘리브레이션 알고리즘을 구현하는 것이다. 또한, 알고리즘 구동에 있어서 수동적인 요소를 최소화하고 시뮬레이션과 실험을 통해 상대 위치 추정 결과를 산출하여 분석함으로써 알고리즘을 검증할 것이다. 제안하는 캘리브레이션 기법은 양산 후 ADAS 센서들이 생산라인에서 정확한 위치에 부착되었는 지 파악하는 검사 장비에 사용될 수 있을 것이다. 캘리브레이션 알고리즘 검증을 위해 시뮬레이션과 실제 실험환경을 구상하였으며, 추후 자동차 생산라인에 적용 가능한지 검토하고자 한다.

## 1.2 연구배경

국내외 다수의 자동차 기업에서 ADAS 장착을 의무화하고 있는 만큼 주행보조 및 자율주행시스템에 대한 기술 개발은 필수적이다. 카메라와 LiDAR 센서는 ADAS의 핵심 부품으로, 다양한 센서의 정보를 함께 사용하는 센서융합기술에 관한 연구 개발이 진행되고 있다. 캘리브레이션은 서로 다른 센서의 위치 정보, 즉 센서 간 변환 행렬을 구하는 과정으로, 차량에서 여러 센서들이 주행환경을 동일하게 인식하기 위해 필수적인 작업이다. 본 연구에서는 RGB 카메라와 3D-LiDAR(Light Detection And Ranging)센서의 보다 정확한 캘리브레이션을 위한 연구를 수행할 것이다.

기존의 3D LiDAR - 카메라 캘리브레이션 연구는 타겟의 모양에 따라 크게 세 분야로 구분할 수 있다. 다각형의 평면 보드를 사용하거나[6, 7], 사각 체스보드를 사용하거나[8, 9], 타겟을 사용하지 않는 연구[10, 11]로 나눌 수 있다. 하지만, 평면 보드를 타겟으로 사용하는 기법의 문제는 라이다 포인트 클라우드로부터 타겟의 모서리를 검출하는 것이 쉽지 않다는 것이다. 특히, 저사양의 LiDAR일수록 vertical 채널 수가 적으므로 모서리 검출이 더욱 어렵다. 또한, 체 보드의 경우 검은색과 흰색이 반복되는 패턴으로 인해 LiDAR 포인트 클라우드에 노이즈가 많이 발생한다는 문제가 있다. 기존의 캘리브레이션 연구들을 몇 가지 소개하자면 다음과 같다.

Rodriguez *et al*의 연구[2]는 체스 보드에서 포인트 클라우드에 노이즈가 많이 발생하는 것을 방지하기 위해 검은 원이 그려진 평면 보드를 타겟으로 사용하였다. 알고리즘은

점은 원의 중심 좌표를 찾고 타겟의 면을 탐색하는 방식으로 구성되어 있다. 또한, 초기에 도출된 변환행렬을 Levenberg-Marquardt(LM) [3] 알고리즘을 통해 최적화한다.

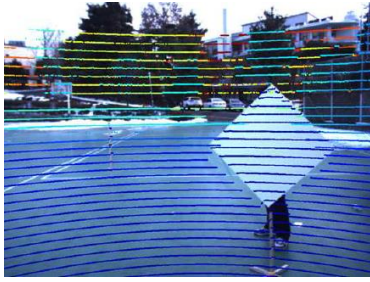
Alismail *et al*의 연구[4]는 중심점을 표시한 원형의 평면 보드를 타겟으로 사용하였다. 연구에서 제안된 알고리즘은 RANSAC을 적용하여 원의 중심과 평면의 normal vector를 찾고 변환행렬을 refine하기 위해 point-plane ICP [5]과 LM을 사용하였다.

Park *et al*의 연구[6]는 흰색의 삼각형 또는 다이아몬드 형태의 평면을 타겟으로 사용하였다. 이미지가 여러 장 필요한 기법이므로, 타겟을 동시에 여러장 두고 촬영하거나 하나의 타겟에 대해 여러 위치에서 촬영한 이미지들을 사용하였다. 이 연구에서는 LiDAR 스캔 라인에 따라 가상의 포인트를 잡고, 이를 토대로 평면 보드의 모서리를 검출한다. 모서리의 교점을 통해 다이아몬드 보드의 꼭짓점의 3차원 좌표를 추정한다. 이미지 상에서는 FAST 알고리즘을 사용하여 다이아몬드 보드의 꼭짓점을 찾는다. 결과적으로, SVD (Singular Value Decomposition) 와 LM 알고리즘을 통해 변환행렬 및 상위치 추정치를 도출한다.

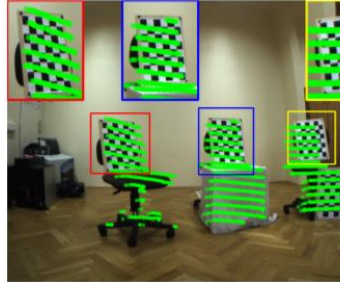
Vales *et al*의 연구[7]는 흰색 배경에 4개의 원이 뚫린 평면 보드를 타겟으로 사용한다. 알고리즘의 핵심은 포인트 클라우드와 이미지에서 뚫린 원을 자동적으로 찾는 것이다. 하지만 Velodyne社의 VLP-16과 같은 저해상도 모델로는 알고리즘 적용이 어렵다는 단점이 있다.

앞서 언급한 일반적인 카메라 - LiDAR의 캘리브레이션 기법은 ‘원이 그려진 평면’, ‘체커보드’, ‘다이아몬드 형태의 평면’ 등 다양한 형태의 2차원 타겟 보드를 캘리브레이션 물체로 선정한다. 하지만, planar calibration 수행시 LiDAR 해상도에 따라 평면의 모서리 검출이 정확하지 않으며, 이로 인해 변환행렬 도출에 큰 오차가 발생한다는 단점이 있다. 최근에 3차원 타겟을 사용하여 기존 planar calibration보다 정밀한 결과를 산출한 연구가 발표되었다[1]. 이에 따라 본 연구에서는 3차원 큐브를 타겟으로 사용하는 캘리브레이션 알고리즘을 구현하고 시뮬레이션 및 실험을 통해 검증할 것이다. 또한, 추후 자동차 생산 라인의 센서 장착 검사 단계에 알고리즘 적용이 가능한 지 검토하고자 한다.

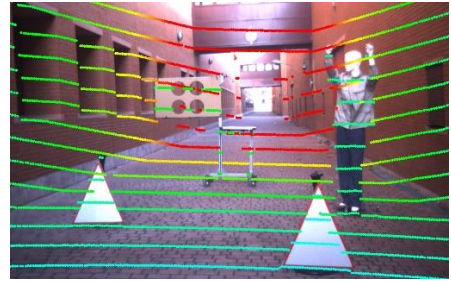




(a) 다이아몬드 평면[6]



(b) 체스 보드[8]



(c) 다각형 및 원이 뚫린 평면[7]

그림 1. 다양한 캘리브레이션 타겟

### 1.3 연구요약

본 연구의 알고리즘은 크게 세 부분으로 나뉜다 (그림 2). 먼저, LiDAR 포인트 클라우드 상에서 큐브의 꼭짓점을 추정하여 3차원 좌표를 얻는다. 이때, RANSAC 알고리즘을 이용하여 큐브의 세 평면을 검출한 후 이상치를 제거하는 단계를 거친다. 또한 LiDAR 좌표계 상에서 꼭짓점의 좌표는 세 평면의 교점과 법선 벡터, 박스의 크기를 통해 구할 수 있다. 두번째로, 카메라 이미지 상에서 큐브의 꼭짓점을 추정하여 이미지 평면 상의 좌표를 얻는다. 이미지 필터링 및 큐브의 모서리 검출 후 각 모서리의 교점을 통해 꼭짓점을 추정한다. 마지막으로 앞서 구한 큐브 꼭짓점 7개의 3차원, 2차원 좌표를 EPnP 알고리즘[12]에 대입 후 LiDAR와 카메라 간의 상대위치 변환 행렬을 얻을 수 있다.

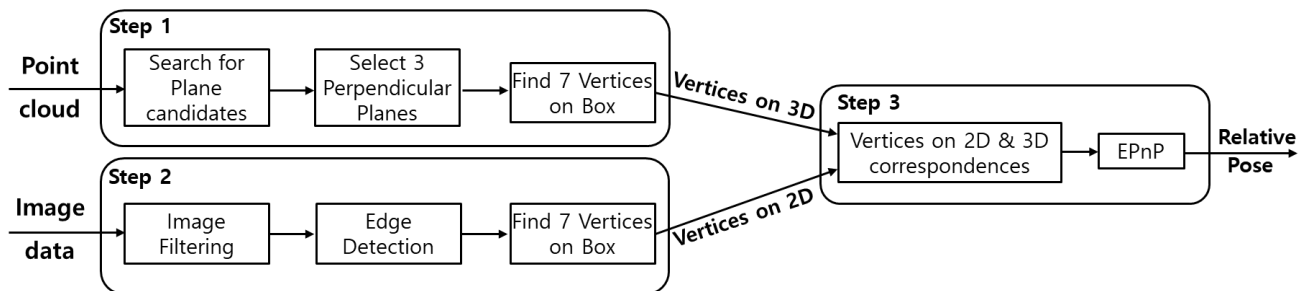


그림 2. 알고리즘 흐름도

## 2. 연구 방법

서론에서 언급한 바와 같이 본 연구의 주된 목표는 카메라와 LiDAR 센서 간의 상대 좌표와 변환행렬을 구하는 것이다. 이를 위해서, 각 센서를 통해 큐브 타겟에 대한 데이터를 얻은 후, 데이터로부터 큐브 타겟에 대한 특징점인 7개의 꼭짓점을 찾는 과정이 진행된다. 3차원 큐브를 캘리브레이션 타겟으로 사용하는 경우, 세 면이 수직인 특성을 이용할 수 있으며 면을 검출하는데 기반을 둔다. 평면을 캘리브레이션을 타겟으로 사용하는 일반적인 방법의 경우, 모서리 검출을 기반으로 하기 때문에 LiDAR 해상도에 많은 영향을 받는다. 따라서, 제안된 알고리즘은 이를 보완할 수 있다.

각 센서를 통해 얻은 데이터로부터 특징점을 추출한 후 EPnP 알고리즘을 사용하여 상대 위치와 변환 행렬을 추정할 것이다. 상대위치는 LiDAR 센서 좌표계의 기준에서 카메라 센서의 좌표계와 가지는 각도와 거리 차이를 나타낸다. 변환행렬은 LiDAR 데이터를 카메라 이미지 위에 매핑할 수 있도록 한다.

본 연구에서 구현한 알고리즘을 시뮬레이션과 실험 각각에 대하여 검증할 것이며, 센서 간 거리와 각도를 변화시키며 추정된 상대위치와 절대위치를 비교할 것이다.

### 2.1 좌표계 정의

각각의 센서는 서로 다른 좌표계를 가진다. 따라서, 각 센서의 좌표계를 정의하고 통일하는 과정이 우선적으로 이루어져야 한다. 카메라 센서는 3차원 공간의 점들을 2차원 이미지로 투영시킨다. 이 때, 카메라 내부행렬  $\mathbf{A}$ 가 사용되며,  $\mathbf{A}$ 는 카메라 센서의 규격으로 주어지는 focal length와 이미지의 center point에 의해 결정된다.  ${}^c\mathbf{P}$ 는 카메라 좌표계 상의 점이고,  $x, y$ 는 투영된 이미지 평면 상의 좌표이다.

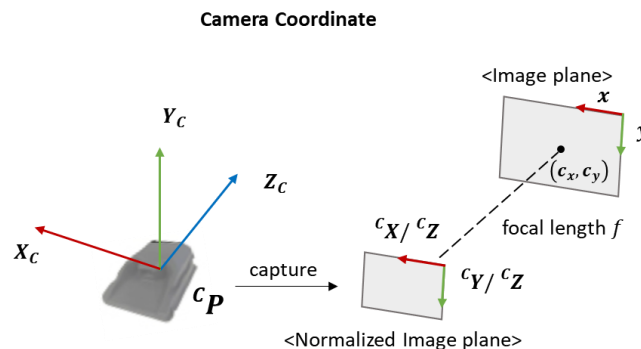


그림 3. 카메라 좌표계 정의

$${}^c\mathbf{P} = [{}^cX \quad {}^cY \quad {}^cZ]^T$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^cX / {}^cZ \\ {}^cY / {}^cZ \\ 1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

3차원 점들의 좌표계를 변환시키기 위해서 회전행렬과 병진행렬의 첨가행렬인 변환행렬을 사용한다. 각 좌표계를 그림으로 간략히 나타냈다 (그림 4). 이때 첨자 W는 월드, L은 LiDAR, C는 카메라를 뜻한다. 본 연구에서 명시된  ${}^c\mathbf{M}_L$  는 LiDAR의 포인트 클라우드 데이터를 카메라 좌표계로 변환하는 행렬이며, 이를 통해 LiDAR 데이터를 카메라 이미지 데이터 위에 매핑하여 나타낼 수 있다.

$${}^c\mathbf{M}_L = [{}^c\mathbf{R}_L \mid {}^c\mathbf{T}_L] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A} {}^c\mathbf{M}_L = \mathbf{A} [{}^c\mathbf{R}_L \mid {}^c\mathbf{T}_L] \begin{bmatrix} {}^LX \\ {}^LY \\ {}^LZ \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} {}^LX \\ {}^LY \\ {}^LZ \\ 1 \end{bmatrix}$$

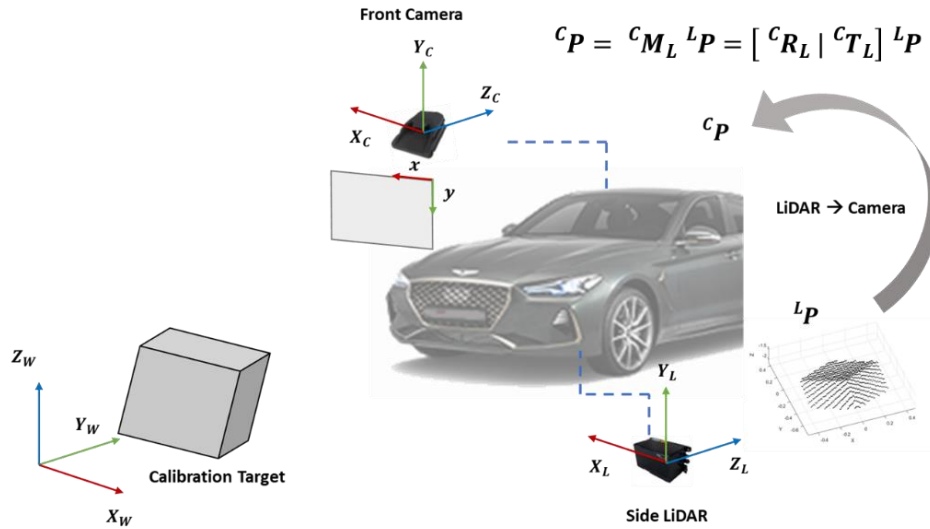


그림 4. 월드, LiDAR, 카메라 좌표계 정의

## 2.2 LiDAR 포인트 클라우드 상의 타겟 특징점 추정

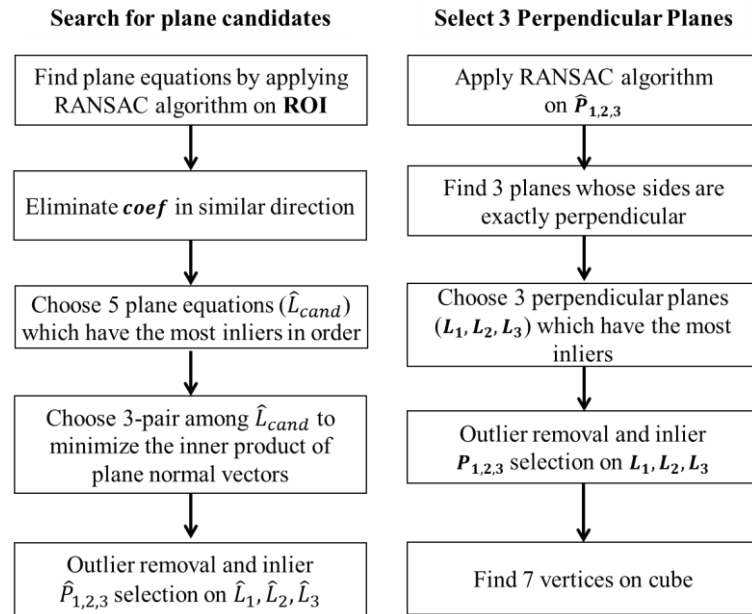


그림 5. LiDAR 포인트 클라우드 상의 타겟 특징점 추정 알고리즘

본 연구에서 제안하는 캘리브레이션 알고리즘의 핵심은 포인트 클라우드와 이미지 상에서 큐브의 일곱 꼭짓점 좌표를 추정하면 EPnP 알고리즘을 통해 쉽게 변환행렬을 도출할 수 있다는 것이다. 따라서 포인트 클라우드 상에서 꼭짓점을 정확히 추정해야 하며, 알고리즘은 위와 같은 과정으로 구성된다.

### 2.2.1 포인트 클라우드 클러스터링

포인트 클라우드 상에서 큐브의 세 꼭짓점을 찾기 위해 먼저 큐브의 세 평면을 추정해야 한다. ROI(Region of Interest)를 설정하여 타겟이 측정된 부분의 포인트 클라우드를 추출한다. 다음으로, ROI 내의 데이터에 Sequential RANSAC 알고리즘[4]을 적용하여 무작위로 추출된 세 점으로부터 평면 방정식을 구한다. 이 과정을 반복하면서 inlier가 가장 많은 평면 상위 5개를 선정한다. 이때, inlier의 기준은 LiDAR 포인트와 평면 간의 수직 거리가 0.02[m] 이내인 점으로 한다. 또한 동일한 평면을 반복해서 선정하는 것을 방지하기 위해 두 법선 벡터가 이루는 각이  $\pm 10^\circ$  이하인 경우, 동일한 방향의 평면으로 간주하여 한 평면은 제외했다.

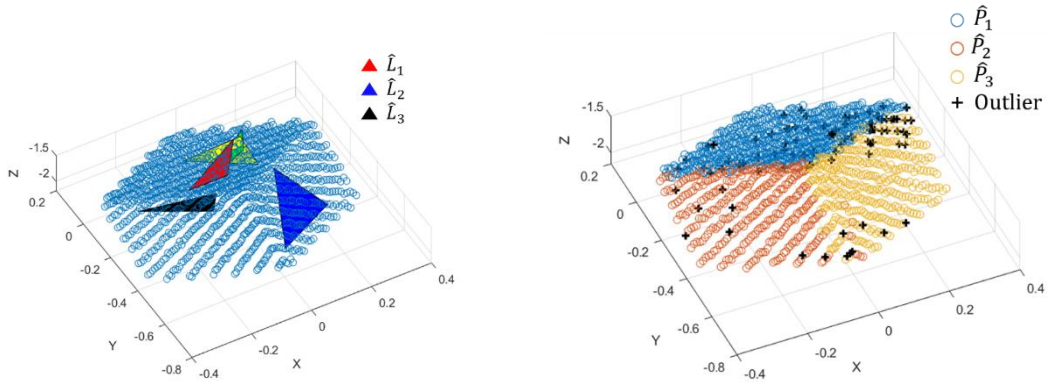


그림 6.  $\hat{L}_{1,2,3}$  및 이상치 제거

Inlier가 가장 많은 상위 5개의 평면 중 거의 수직한 세 평면쌍  $\hat{L}_{1,2,3}$  을 찾기 위해 각 평면의 법선벡터 사이 내적의 합(E)을 최소화하는 평면쌍을 찾는다(식 1).  $\hat{L}_{1,2,3}$  에 대해 각각의 이상치를 제거하고 각 평면에 대한  $\text{inlier}(\hat{P}_{1,2,3})$ 를 클러스터링한다. 이때,  $n_i$  는  $\hat{L}_i$  의 법선 벡터이다.

$$E = |n_1^T n_2| + |n_2^T n_3| + |n_3^T n_1| \quad (1)$$

### 2.2.2 수직한 세 평면 탐색

2.2.1에서 찾은  $\hat{L}_{1,2,3}$  는 RANSAC 알고리즘을 이용하여 포인트 클라우드로부터 각각 추정된 평면이기 때문에 완벽히 서로 수직일 수 없다. 따라서 RANSAC 알고리즘을 한번 더 적용하여 큐브에서 서로 완벽히 수직을 이루는 세 평면을 찾는다. 먼저,  $\hat{P}_1$  에서 무작위로 세 점을 추출하여 평면  $\tilde{L}_1$  을 구성한다.  $\tilde{L}_1$  에 수직하고  $\hat{P}_2$  에서 무작위로 추출된 두 점을 동시에 지나는 평면은  $\tilde{L}_2$  로 정해진다. 마지막으로,  $\tilde{L}_1$  와  $\tilde{L}_2$  에 수직하고  $\hat{P}_3$  에서 무작위로 추출된 한 점을 동시에 지나는 평면은  $\tilde{L}_3$  로 정해진다.

앞의 과정을 반복하면서 inlier 가 가장 많은 서로 수직한 세 평면쌍  $L_{1,2,3}$  을 찾는다. 최종적으로 큐브의 세 면으로 추정되는  $L_{1,2,3}$  에 대해 각각의 이상치를 제거하고 각 평면에 대한  $\text{inlier}(P_{1,2,3})$ 를 클러스터링한다.

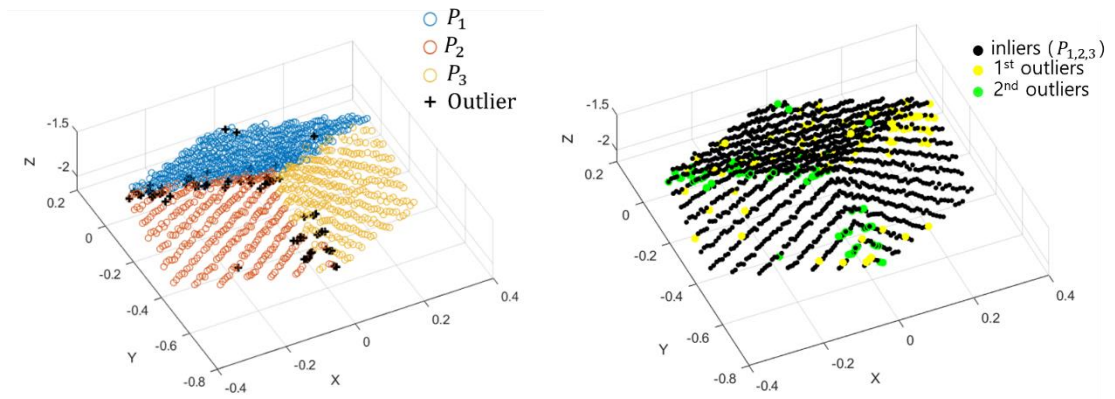


그림 7.  $L_{1,2,3}$  및 이상치 제거

### 2.2.3 타겟 특징점 추정

2.2.2에서 구한 서로 수직한 평면 방정식  $L_1$ ,  $L_2$ ,  $L_3$ 의 교점을 구하기 위해, 각 평면의 평면 방정식과  $L_1$  위 임의의 점을 이용한다.  $L_1$  위의 한 점을  $L_2$ 로 사영시킨 후,  $L_3$ 으로 다시 사영시킨 결과를 세 평면의 교점(그림 8의 7번 꼭짓점)으로 간주한다. 남은 6개의 꼭짓점 좌표를 추정하기 위해 각 평면의 법선 벡터와 이미 알고 있는 타겟의 크기를 활용한다.

예를 들어, 그림 8의 2번 꼭짓점을 찾는 경우, 7번 꼭짓점의 좌표에서  $L_3$ 의 법선 벡터와 타겟의 모서리 길이의 곱을 더하여 구할 수 있다. 이와 같은 방식으로 모든 꼭짓점의 좌표를 추정할 수 있다.

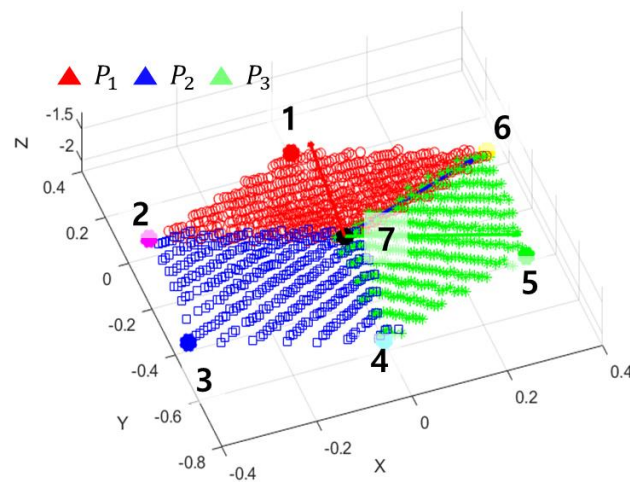


그림 8. 타겟 특징점 추정

## 2.3 카메라 이미지 상의 타겟 특징점 추정

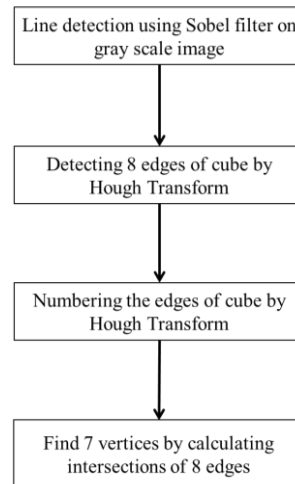


그림 9. 카메라 이미지 상의 타겟 특징점 추정 알고리즘

이미지 상에서 타겟의 특징점을 추출하기 위해, 이미지 처리 기법을 사용하여 이미지 상에서 타겟의 7개의 특징점을 추정하였다. Sobel Filter를 통해 타겟의 모서리를 검출 후, 그림 10(b)와 같이 Hough Transformation을 통해 이미지 평면에서 Hough Line을 검출하였다. 타겟의 모서리를 나타내는 8개의 Hough Line 직선 방정식으로부터 계산한 7개의 교점을 타겟의 꼭짓점으로 추정한다.

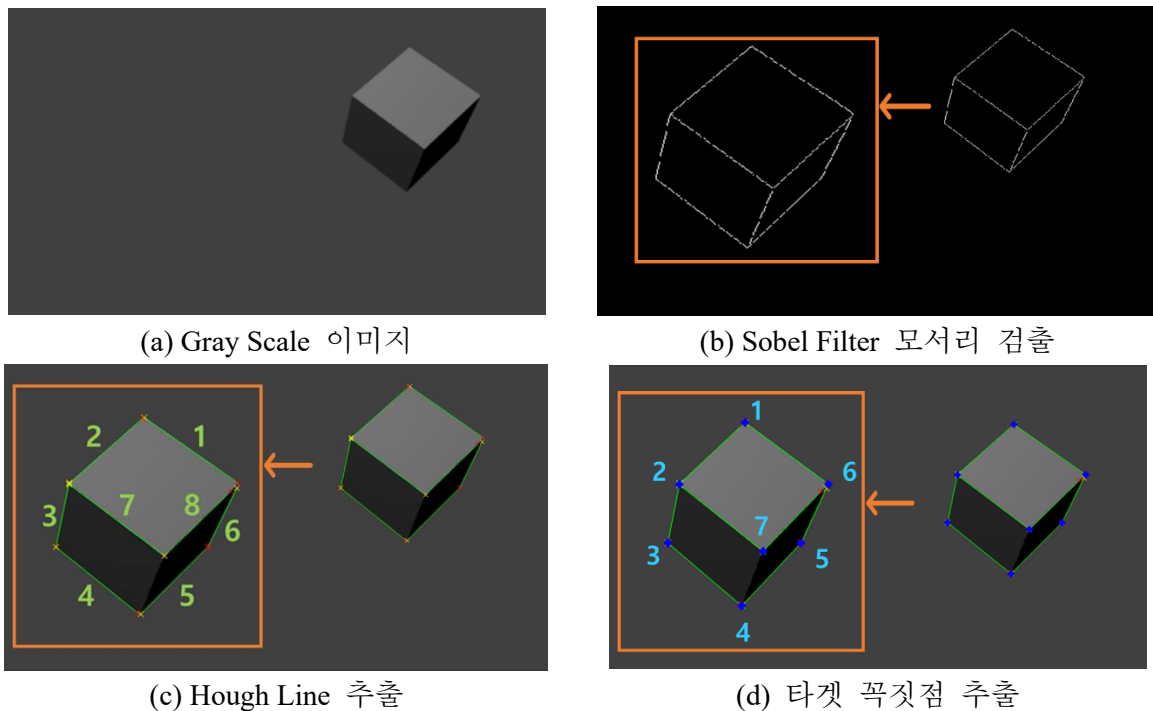


그림 10. 이미지 특징점 추출 과정

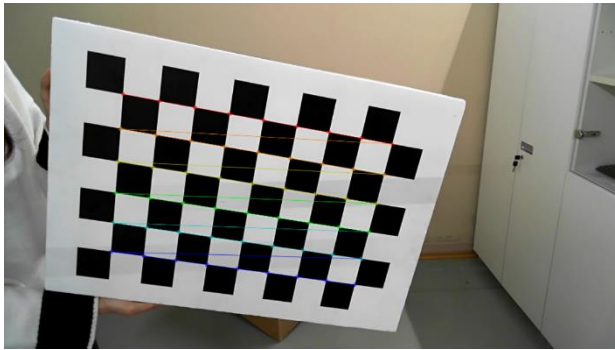


## 2.4 상대위치 및 변환행렬 추정

### 2.4.1 카메라 내부변수행렬 A 도출

EPnP 알고리즘을 사용하기 앞서, 카메라 내부변수 행렬을 구하는 것은 필수적이다. 내부변수행렬이란, 카메라의 규격과 연관되어 있는 카메라 고유의 행렬이며, 카메라 좌표계 상의 점들을 이미지 평면으로 투영시키기 위한 행렬이다.

일반적으로 내부변수행렬은 카메라의 focal length와 이미지 평면의 중점의 좌표를 통해 알아낼 수 있다. 시뮬레이션에서는 ‘Blender’ 프로그램에서 제공하는 센서 규격에 따라 내부변수행렬을 구성하였다. 하지만 실험에서는 내부변수 행렬을 구하기 위해, 센서의 규격에 명시된 고유의 값들을 직접 사용하지 않고, Zhang’s method[13]의 결과를 사용하였다. Zhang’s method란 여러 장의 체커보드를 사용하여 카메라의 내부변수를 추정하는 방법으로, 구성된 실험환경에 적합한 내부변수를 구할 수 있다는 장점이 있다. 그림 11은 ‘Darkcamcalibrator’ 프로그램을 통해 Zhang’s method를 이용하여 실험에서 사용한 카메라 (MS 1080p HD)의 내부변수 행렬을 추정한 결과이다.



(a) 카메라 캘리브레이션 수행

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1448 & 0 & 960 \\ 0 & 1450 & 540 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) 추정된 내부변수 행렬

그림 11. 카메라 내부변수 행렬 추정



### 2.4.2 EPnP 알고리즘

본 연구에서는 카메라 - LiDAR 사이의 상대위치를 추정하기 위해 EPnP 알고리즘[12]을 사용한다. EPnP 알고리즘의 핵심은 reference point들의 좌표를 control point의 weighted sum으로 나타내는 것이다(식 2). EPnP 알고리즘은 기존의 PnP 문제를 다루는 솔루션보다 훨씬 더 빠르고 정확한 결과를 도출했다.

본 연구에서 EPnP 알고리즘에 대한 입력은 LiDAR 좌표계와 이미지 평면 상에서의 꼭짓점 좌표 7개( $\mathbf{p}_i^l, \mathbf{u}_i$ )이며, 알고자 하는 출력은 카메라 좌표계 상에서 타겟의 일곱 꼭짓점의 3차원 좌표( $\mathbf{p}_i^c$ )이다.

EPnP 알고리즘에서 control point는 총 4개가 필요하며, 이는 알고리즘 내부에서 알고리즘의 안정성을 높이도록 자동으로 지정된다. 이때, reference point의 중심(centroid)을 control point 하나로 선정하고 나머지 control point는 입력 데이터의 주방향을 따라 기저 방향을 형성하도록 선정된다.

$$\mathbf{p}_i^l = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^l, \quad \text{with} \quad \sum_{j=1}^4 \alpha_{ij} = 1 \quad (2)$$

카메라의 내부변수 행렬이  $\mathbf{A}$ 일 때, 2.1의 좌표계 정의에 따라 다음 식이 성립한다.

$$\forall i, \quad w_i \begin{bmatrix} u_i \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{p}_i^c = \mathbf{A} \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c \quad (3)$$

$$\forall i, \quad w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \quad (4)$$

식 4의 3행을 1, 2 행에 대입 후 정리하면  $w_i$ 는 소거되고 reference point에 대한 선형 방정식 2개를 얻을 수 있다 (식 5). 이를 선형 시스템으로 나타낼 수 있다 (식 6).  $n$ 은 reference point의 개수로, 본 연구에서는 꼭짓점 7개를 사용하므로  $n = 7$ 이다.  $\mathbf{M}$ 은  $2n \times 12$  행렬,  $\mathbf{x}$ 는  $12 \times 1$  벡터이다.

$$\begin{aligned} \sum_{j=1}^4 \alpha_{ij} f_u x_j^c + \alpha_{ij} (u_c - u_i) z_j^c &= 0 \\ \sum_{j=1}^4 \alpha_{ij} f_v y_j^c + \alpha_{ij} (v_c - v_i) z_j^c &= 0 \end{aligned} \quad (5)$$

$$\mathbf{M}\mathbf{x} = 0, \quad \text{where } \mathbf{x} = [\mathbf{c}_1^{cT} \quad \mathbf{c}_2^{cT} \quad \mathbf{c}_3^{cT} \quad \mathbf{c}_4^{cT}]^T \quad (6)$$

식 6을 풀기 위해  $\mathbf{x}$ 를 식 7과 같이 표현한다. 이때,  $\mathbf{M}$ 을 특이값 분해(SVD) 후  $\mathbf{V}$  행렬의 열 벡터를 취하여  $\mathbf{v}_i$ 로 명시하였으며,  $\beta_i$ 는 각 원소에 대한 weight를 의미한다. 또한,  $N$ 은  $\mathbf{M}^T\mathbf{M}$  행렬의 null space의 크기이다.

$$\mathbf{x} = \sum_{i=1}^N \beta_i \mathbf{v}_i \quad (7)$$

$N$ 에 따른 경우를 나누어 weight를 찾아낸 후  $\mathbf{x}$ 를 알 수 있다.  $N=1$ 인 경우,  $\mathbf{x} = \beta\mathbf{v}$ 이고, LiDAR 좌표계와 카메라 좌표계에서 control point 간의 거리는 동일해야 하므로 식 9를 도출할 수 있다. 이때,  $\mathbf{v}^{[i]}$ 는  $\mathbf{v}$ 의  $i$ 번째 벡터를 의미한다. 따라서, 식 9의 우변을 알고 있으므로 weight인  $\beta$ 를 식 10과 같이 구할 수 있다.

$$N=1: \mathbf{x} = \beta\mathbf{v} \quad (8)$$

$$\|\beta\mathbf{v}^{[i]} - \beta\mathbf{v}^{[j]}\|^2 = \|\mathbf{c}_i^L - \mathbf{c}_j^L\|^2 \quad (9)$$

$$\beta = \frac{\sum_{\{i,j\} \in [1;4]} \|\mathbf{v}^{[i]} - \mathbf{v}^{[j]}\| \cdot \|\mathbf{c}_i^L - \mathbf{c}_j^L\|}{\sum_{\{i,j\} \in [1;4]} \|\mathbf{v}^{[i]} - \mathbf{v}^{[j]}\|^2} \quad (10)$$

다음으로,  $N=2$ 인 경우, 마찬가지로 control point간의 거리가 카메라와 LiDAR 좌표계에서 동일하다는 조건으로 인해 식 12를 얻을 수 있다. 식 13과 같이 선형 시스템을 구성한 후  $L$  행렬의 pseudoinverse를 통해 weight를 구할 수 있다.

$$N=2: \mathbf{x} = \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2 \quad (11)$$

$$\|(\beta_1\mathbf{v}_1^{[i]} + \beta_2\mathbf{v}_2^{[i]}) - (\beta_1\mathbf{v}_1^{[j]} + \beta_2\mathbf{v}_2^{[j]})\|^2 = \|\mathbf{c}_i^L - \mathbf{c}_j^L\|^2 \quad (12)$$

$$\begin{aligned} L: & 6 \times 3 \text{ matrix with the element of } \mathbf{v}_1 \text{ and } \mathbf{v}_2 \\ L\boldsymbol{\beta} &= \boldsymbol{\rho} \quad \boldsymbol{\beta}: [\beta_1^2 \quad \beta_1\beta_2 \quad \beta_2^2]^T \quad (13) \\ \boldsymbol{\rho}: & 6 \times 1 \text{ vector with squared distances } \|\mathbf{c}_i^L - \mathbf{c}_j^L\|^2 \end{aligned}$$

$N=3, 4$ 인 경우,  $N=2$ 와 동일한 과정을 통해 weight를 구할 수 있다. Weight  $\beta$ 를 구한 후 식 8과 11에 의해  $\mathbf{x}$ 를 도출할 수 있다. 결과적으로,  $\mathbf{x}$ 의 원소인  $\mathbf{c}_i^c$ 를 식 2에 대입하여 reference point의 카메라 좌표계 상 좌표인  $\mathbf{p}_i^c$ 를 얻을 수 있다. 또한,  $\mathbf{p}_i^c$ 와  $\mathbf{p}_i^L$ 를 토대로 LiDAR 좌표계와 카메라 좌표계 간 변환행렬을 얻을 수 있다.

본 연구에서는 타겟인 큐브의 꼭짓점 7개의 LiDAR 좌표계 상 3차원 좌표와, 이미지 평면 상 2차원 좌표를 EPNP 알고리즘의 입력으로 하여 변환행렬  ${}^L M_C$  을 도출하였다.  ${}^L M_C$  을 토대로 카메라 - LiDAR의 상대위치를 추정하였고 시뮬레이션 및 실험을 통해 상대위치 추정치와 ground truth를 비교함으로써 본 연구에서 제안하는 알고리즘을 검증하였다.

## 3. 연구 결과

### 3.1 시뮬레이션 결과

#### 3.1.1 시뮬레이션 환경

LiDAR와 카메라의 가상 데이터를 수집하기 위해 “Blender” 프로그램을 사용하여 그림 12와 같이 시뮬레이션 환경을 구상하였다. 각 센서와 타겟은 ADAS 탑재 차량의 경우를 가정하여 배치했다. Ground Truth는 표 2와 같이 설정했으며 타겟으로부터 각 센서 간의 거리는 2[m]이다. 각 센서의 규격은 표 1과 같다.

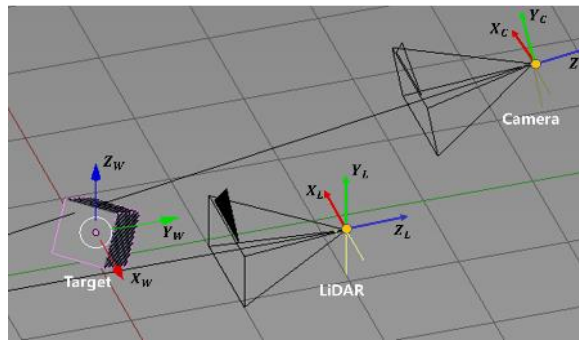




그림 12. Blender 시뮬레이션 구상

표 1. 시뮬레이션 환경 센서 규격

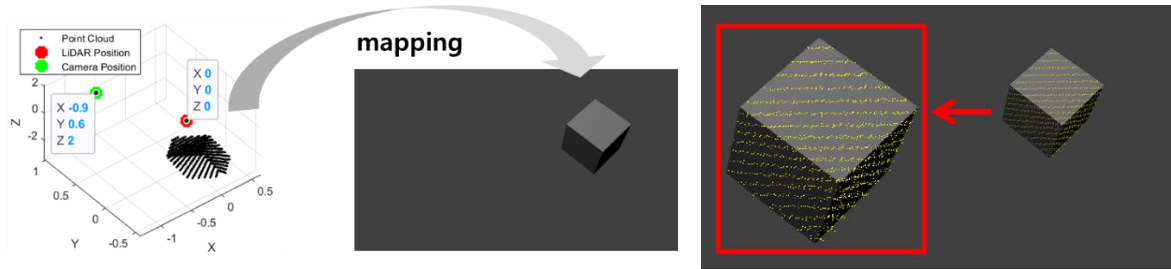
Sensor Specification				Virtual Sensor
LiDAR (Velodyne HDL 32E)	Scan resolution	Noise mean (Depth)	Noise $\sigma$ (Depth)	 Velodyne HDL 32E
	0.17 [deg]	0.00 [m]	0.02 [m]	
Camera (RGB)	Sensor size	Image size	Focal Length	
	32×32 [mm]	960×540 [px]	35 [mm]	

#### 3.1.2 시뮬레이션 결과

위의 시뮬레이션 환경 구상에서 얻은 가상의 LiDAR와 카메라 데이터셋 10 프레임에 본 연구의 알고리즘을 적용하여 표 2와 같이 상대위치 추정 결과를 도출하였다. 또한, EPnP 알고리즘을 통해 도출한 변환행렬(회전 행렬과 병진 행렬)을 이용하여 LiDAR 3D 포인트를 이미지 평면에 매핑하여 그림 13의 결과를 얻을 수 있었다.

표 2. 시뮬레이션 상대위치 추정 결과

Simulation Result	Rotation[deg]			Translation[m]		
	roll	pitch	yaw	dx	dy	dz
Ground Truth	-15.00	0.00	0.00	-0.90	0.60	2.00
Estimation	-15.61	-0.01	-0.76	-0.89	0.66	1.98
Error	0.61	0.01	0.76	0.01	0.06	0.02



(a) LiDAR, 카메라 측정 데이터

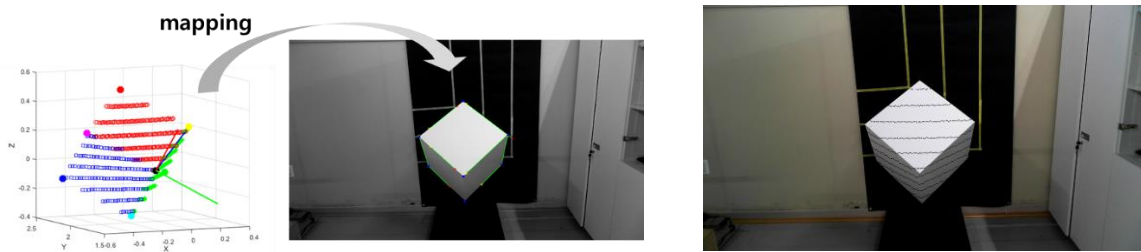
(b) 매핑된 데이터

그림 13. 시뮬레이션 변환행렬 적용 결과

## 3.2 실험 결과

### 3.2.1 실험 환경 구성

그림 15와 같이 실험 환경을 구성하였다. 타겟으로부터 각 센서 간의 거리는 2.1[m]이며, LiDAR의 초기 위치와 카메라 사이의 높이 차이는 73[cm]이다. 알고리즘 검증을 위해 각 센서의 절대위치 간의 차이인 Ground Truth와 알고리즘을 통해 추정한 상대위치를 비교하였다. 또한, LiDAR를 상, 하, 좌, 우 15[cm] 평행 이동, z축 방향 +10[deg] 회전 이동하며 동일한 타겟에 대한 캘리브레이션을 수행하였다. 실험을 위해 사용한 각 센서의 규격은 표 3과 같다.



(a) LiDAR, 카메라 데이터 측정

(b) 매핑된 데이터

그림 14. 실제 실험 변환행렬 적용 결과

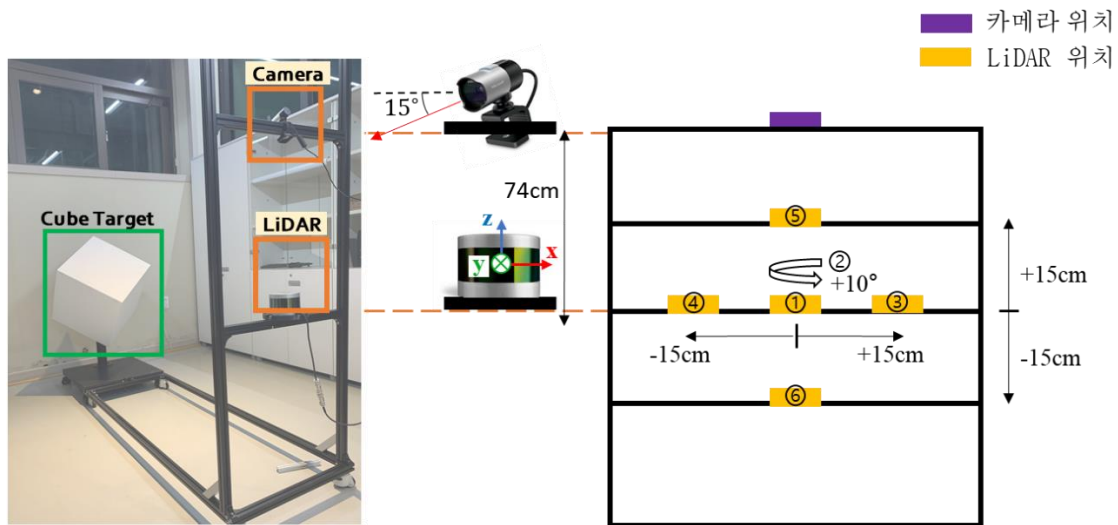




그림 15. 실험 환경 구성

표 3. 실험 환경 센서 규격

Sensor Specification				Sensor Image
<b>LiDAR</b> (Velodyne VLP-16)	VER./HORIZ. FOV	Range Accuracy	VER./HORIZ. Angular Resolution	 Velodyne VLP-16
	$\pm 15^\circ / 360^\circ$	$\pm 3[\text{cm}]$	$0.1^\circ / 0.4^\circ$	
<b>Camera</b> (MS 1080p HD)	Image size		Focal Length	 MS 1080p HD
	1920×1080[px]		60[mm]	

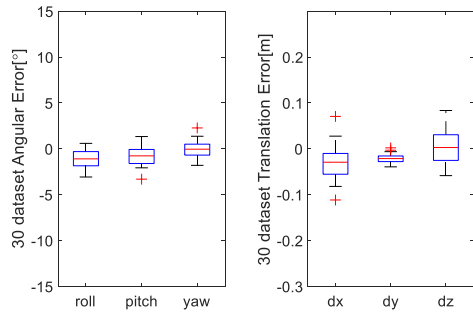
### 3.2.2 실험 결과

그림 15의 LiDAR 위치 번호와 동일한 순서대로 LiDAR 위치를 변화하면서 각 경우마다 30개의 데이터 셋에 대한 위치 추정 결과의 평균치를 표 4에 나타내었다. 실제 LiDAR의 위치 이동량과 알고리즘 상에서 추정된 이동량을 비교하기 위해, ① 위치에서의 추정량을 기준으로, 다른 위치의 추정량 간의 오차를 분석하였다.

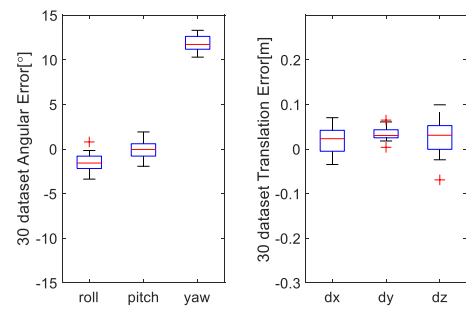
그림 16(a)는 초기 상대위치에서 Ground Truth와 각 센서의 초기 상대위치에 대한 오차를 나타낸 Box plot이다. 그림 16(b) ~ (f)는 동일한 타겟에 LiDAR의 위치를 변화시키며 얻은 상대위치 추정 값과 초기 위치(Case 1)에서 얻은 추정 값 사이의 오차를 나타낸다. 표 5는 그림 16(b) ~ (f)의 평균오차를 나타낸 표이며, 이를 통해 실제 이동량과의 알고리즘을 통해 추정된 이동량의 차이를 알 수 있다.

표 4. 실험 결과

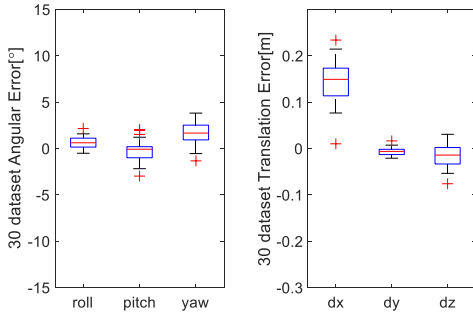
Case No.	Rotation [deg]			Translation [m]		
	roll	pitch	yaw	dx	dy	dz
Ground Truth	<b>75</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0.74</b>
1	<b>76.079</b>	<b>0.802</b>	<b>0.0315</b>	<b>0.0300</b>	<b>0.021</b>	<b>0.737</b>
2	74.539	0.675	11.829	0.050	0.056	0.754
3	76.732	0.530	1.628	0.170	0.015	0.720
4	77.012	0.777	1.755	-0.115	0.013	0.710
5	75.769	0.727	1.801	0.040	0.034	0.861
6	76.023	1.015	1.960	0.040	0.030	0.560



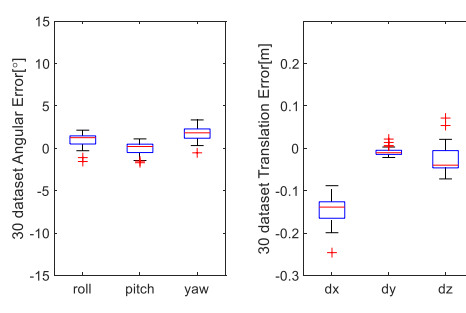
(a) Ground Truth 대비 Case 1의 오차



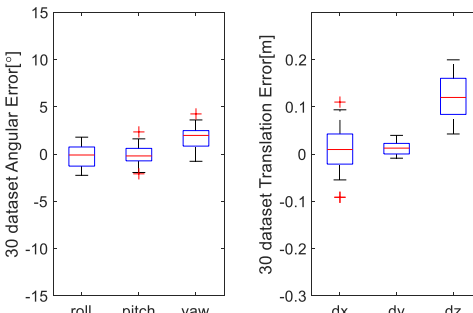
(b) Case 1 대비 Case 2의 오차



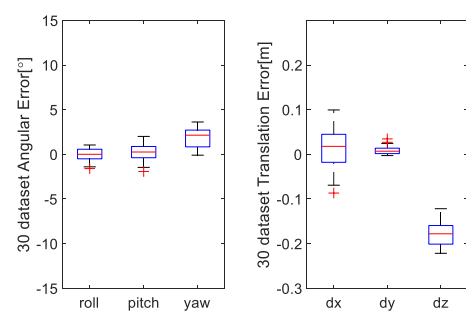
(c) Case 1 대비 Case 3의 오차



(d) Case 1 대비 Case 4의 오차



(e) Case 1 대비 Case 5의 오차



(f) Case 1 대비 Case 6의 오차

그림 16. 실험 결과 Box Plot

표 5. Case 1 대비 각 case의 상대위치 추정 오차

Case No.	Rotation Error [deg]			Translation Error [m]		
	roll	pitch	yaw	dx	dy	dz
(b)	-1.54	-0.127	1.7975	0.02	0.035	0.017
(c)	0.653	-0.272	1.5965	-0.01	-0.006	-0.017
(d)	0.933	-0.025	1.7235	0.005	-0.008	-0.027
(e)	-0.31	-0.075	1.7695	0.01	0.013	-0.026
(f)	-0.056	0.213	1.9285	0.01	0.009	-0.027

## 4. 토 의

### 4.1 시뮬레이션 결과 분석

시뮬레이션 환경에서 동일한 카메라 - LiDAR 위치에서 10개의 프레임을 수집 후 본 연구의 알고리즘을 통해 상대 위치 추정 결과의 평균치를 얻었다. 이때, 실험환경에서 LiDAR 포인트 깊이 값에 노이즈가 발생할 것을 감안하여 LiDAR 포인트 클라우드는 z축 방향으로 Gaussian noise( $\sigma = 0.02[m]$ )를 인가했다. 상대 위치 추정 시 회전이동량에서 최대  $0.76^\circ$ , 병진이동량에서 최대  $0.06m$ 의 오차가 발생하였다 (표 2).

시뮬레이션 결과, 평면 보드를 사용한 기존의 캘리브레이션 연구[6, 8] 보다 상대위치 추정에 우수한 정확도를 보였다. 큐브 모양의 타겟의 경우, 큐브의 세 면을 추정하면 타겟의 모서리 길이와 세 평면의 교점을 이용하여 다른 꼭짓점들을 쉽게 추정할 수 있다는 장점이 있다. EPnP 알고리즘의 특성 상, reference point가 많을수록 정확도가 향상되므로 일반적인 평면 보드를 타겟으로 하는 경우보다 특징점이 7개인 큐브를 사용하는 것이 상대위치 추정 정확도가 높다.

시뮬레이션을 통해 구현된 알고리즘이 포인트 클라우드와 이미지에 대해 올바르게 구동되는지 확인할 수 있었다. 하지만, Blender 프로그램에서는 32, 64채널의 LiDAR (Velodyne 社)와 하나의 RGB 카메라로 측정한 데이터만을 제공하므로 실제 실험에서 사용할 센서와 동일한 모델로 시뮬레이션할 수 없다는 한계가 있다.



## 4.2 실험 결과 분석

그림 15의 실험 환경에서 동일한 카메라 - LiDAR 위치에서 30개의 프레임을 수집 후 본 연구의 알고리즘을 통해 상대 위치 추정 결과의 평균치를 얻었다. 실험에서는 절대적인 Ground Truth를 정확하게 측정하기 어려우므로, 1번 위치에서의 상대위치 추정 결과를 기준으로 LiDAR의 위치를 변화시키며 추정 결과를 비교하였다 (표 4, 그림 16). 상대 위치 추정 실험 결과, 회전이동량에서는 최대  $1.93^{\circ}$ , 병진이동량에서는 최대  $0.035[m]$ 의 오차가 발생하였다. z축 방향 yaw의 추정치에 가장 큰 오차가 발생하였는데, 이는 LiDAR의 depth 노이즈로 인해 RANSAC을 통한 평면 추정에 오차가 생겼기 때문으로 추정된다. 또한 실험 시 라이다가 움직이지 않도록 완벽히 고정하기 위한 체결 장치가 없었는데 이로 인해 z축 방향에 오차가 발생했을 가능성도 있다.

실험에서는 시뮬레이션에서 사용된 센서보다 저사양의 16채널 LiDAR (Velodyne VLP-16)을 사용하였지만 데이터 상의 노이즈는 더 적은 범위에서 발생하였다. 시뮬레이션과 실험 환경이 동일하지 않으므로 두 결과를 직접적으로 비교하기는 어려울 것으로 보인다. 그러나, 시뮬레이션과 실험 모두 평면 보드를 사용한 기존의 캘리브레이션 연구[6, 8] 보다 병진이동량 추정에 우수한 정확도를 보였다.

본 연구에서 구성한 실험환경(그림 15)으로는 Ground Truth를 정확히 측정하기 어렵고 카메라와 LiDAR가 다양한 상대위치를 갖도록 설치하는 것이 쉽지 않았다. 따라서 카메라는 고정시키고 LiDAR만 움직였을 때의 상대위치 추정 결과를 초기 위치에서의 상대위치 추정 결과와 비교하였다. 추후, 정밀한 실험환경을 구성하여 카메라 - LiDAR의 상대적인 회전 및 병진이동량을 다양하게 부여하고 알고리즘을 검증하는 과정이 수행되어야 한다.

## 5. 결 론

본 연구에서는 저해상도 라이다와 RGB 카메라를 이용하여 캘리브레이션을 수행하는 알고리즘을 구현하였다. 일반적인 캘리브레이션 방식인 평면 보드를 타겟으로 사용하지 않고 3차원 큐브를 타겟으로 사용하여 캘리브레이션을 수행하였다.

각 센서의 데이터로부터 큐브 타겟의 특징점인 7개의 꼭짓점을 추출하였고, PnP의 솔루션으로는 각각의 좌표를 매칭하여 상대위치와 변환행렬을 얻는 EPnP 알고리즘을 사용하였다. 특히 LiDAR 포인트의 특징점 추출 시, 두 번의 RANSAC 알고리즘을 수행하며, 이상치를 제거하는 과정을 반복하였다. 이를 통해 특징점의 좌표를 더 정확하게 추정할 수 있었다. 알고리즘을 수행하여 얻은 결과는 센서 간 상대위치와 각 센서 데이터의 좌표계를 동기화하기 위한 변환행렬이며, 센서융합 알고리즘 구현 시 필수적인 정보이다.

본 연구의 알고리즘을 통해 추정한 변환행렬은 LiDAR 3D 데이터를 카메라 2D 이미지에 매핑하여 사용될 수 있다. 이를 활용하면 센서 부착 차량의 이동 환경에서, 변환행렬과의 연산을 수행하여 실시간으로 주행 환경을 인지하고 주변 환경을 파악할 수 있을 것이다. 또한, 생산라인과 같은 정지된 환경에서 알고리즘을 통해 추정한 상대위치를 사용하면, 센서의 Ground Truth 값과 실제 부착량 사이의 오차를 파악할 수 있을 것이다.

## 6. References

- [1] Zoltan Pustai and Levente Hajder. “Accurate Calibration of LiDAR-Camera Systems using Ordinary Boxes”, *IEEE International Conference on Computer Vision Workshops*, 2017
- [2] S. Rodriguez F, V. Fremont, and P. Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. 1, 2
- [3] K. Levenberg. “A method for the solution of certain problems in least squares”. *Quart. Appl. Math.*, 2:164–168, 1944. 2, 6
- [4] H. S. Alismail, L. D. Baker, and B. Browning. “Automatic calibration of a range sensor and camera system”. In 2012 Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT 2012), Pittsburgh, PA, October 2012. *IEEE Computer Society*. 2
- [5] Y. Chen and G. Medioni. “Object modelling by registration of multiple range images”. *ImageVisionComput.*, 10(3):145–155, Apr. 1992. 2
- [6] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim. “Calibration between color camera and 3d lidar instruments with a polygonal planar board”. *Sensors*, 14(3):5333–5353, 2014. 1, 2, 3, 5, 6
- [7] M. Velas, M. Spanel, Z. Materna, and A. Herout. Calibration of rgb camera with velodyne lidar. In WSCG 2014 Communication Papers Proceedings, volume 2014, pages 135–144. Union Agency, 2014. 1, 2
- [8] A. Geiger, F. Moosmann, O. Car, and B. Schuster. “Automatic camera and range sensor calibration using a single shot”. In *IEEE International Conference on Robotics and Automation, ICRA 2012*, 14-18 May, 2012, St. Paul, Minnesota, USA, pages 3936–3943, 2012. 1, 2, 5, 7
- [9] G. Pandey, J. McBride, S. Savarese, and R. Eustice. “Extrinsic calibration of a 3d laser scanner and an omnidirectional camera”. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 7, Lecce, Italy, September 2010. 1
- [10] R. Frohlich, Z. Kato, A. Trémeau, L. Tamas, S. Shabo, and Y. Waksman. “Region based fusion of 3d and 2d visual data for cultural heritage objects”. In *23rd International Conference on Pattern Recognition, ICPR 2016*, Cancun, Mexico, December 4-8, 2016, pages 2404–2409, 2016. 1
- [11] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice. “Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information”. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 2053–2059, Toronto, Canada, July 2012. 1

- [12] V. Lepetit, F. Moreno-Noguer, and P. Fua. “Epnnp: An accurate  $O(n)$  solution to the pnp problem”, *International Journal Computer Vision*, 81(2), 2009
- [13] Zhengyou Zhang, Senior Member, “A Flexible New Technique for Camera Calibration”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), 2000