2019**지능형모형차 경진대회 보고서**

|  |  |
| --- | --- |
| **학 교** | 건국대학교 |
| **팀 명** | 숑지나가요 |
| **유 형** | 본 경기 |
| **팀 장** | 김승수 (전기전자공학부) |
| **팀 원** | 김은주 (전기전자공학부)  민경민 (전기전자공학부)  유진호 (전자공학과)  김서정 (전기전자공학부) |

1. 개요

1.1 설계 배경

오늘날 전 세계는 여가시간을 창출해 내고 앞으로의 인류를 위해 두 가지 키워드를 화두로 삼고 있다. 첫째로 5G, big data, IoT, deep learning, AI등의 4차 산업, 둘째로는 수소, 전기, 풍력 등의 친환경이다. 이들 중에서도 많은 관심을 두고 있는 자율주행이라는 단어는 점점 우리에게 익숙한 단어가 되었고 이제는 생활 속에서 함께할 일만을 남기고 있다.

이런 세계적인 흐름을 따라 애플의 경우 핸드폰, 노트북 등의 기존의 주력 산업에서 자율주행 자동차 스타트업 기업을 인수했고 LG전자 역시 블랙베리와 함께 자율주행차 기술을 갖추는데 노력함으로써 그 흐름에 뛰어들었다. 그렇지만 우리의 생활 속에 완전히 녹아 들어오기에는 아직 많은 시간이 필요할 것으로 예상된다. 가장 최근의 사건으로 서울시에서 시연한 5G 자율주행 버스의 중앙선 침범이 있었으며 해외에서는 자율주행차 시범운전을 하던 중 사망사고가 발생한 경우가 있었기 때문이다. 사실 서울시에서 자율주행 버스를 시연하기 전까지는 대부분 사람, 가로등, 건물 등과 같은 장애물을 피하고 이 때 필요하다면 긴급 제동을 해야 하기에 이 두 가지에 큰 관심을 두었을 것이다.

그러나 이제는 가장 기본인 선을 인식하는 것에서 부족함이 나타나는 것을 확인했다. 따라서 우리 팀은 이 Line Scan이라는 기본을 단단하게 다진 후에 장애물을 회피하고 긴급제동을 하는 것을 쌓아 올리는 것을 목표로 삼고 지능형 모형차 대회에 참가하게 되었다.

.

1.2 설계 목표

지능형 자동차는 자동차 스스로가 현재, 이전 모든 상황을 종합하여 앞으로의 주행상황을 예측한다. 따라서 차선 확인(Line Scan), 장애물 인식, 회피, 긴급 제동(AEB)의 크게 네 가지를 목표로 각각의 상황마다 속도와 방향을 조절해야 한다. 따라서 이를 가지고 세부 목표를 세워보면 다음과 같다.

* Line Scan camera를 이용하여 차선을 인식함으로써 자동차가 스스로 자신의 위치(차선)를 파악할 수 있도록 한다.
* 적외선 센서를 이용해 장애물, AEB, 언덕의 경우를 각각 알고리즘을 통해 판단한 후 각각의 경우에 맞춰 자동차가 속도와 방향을 제어한다.

이 두가지 큰 목표를 통해 자동차가 스스로 상황을 판단하고 결정을 내리는 자율주행 시스템에 대해 이해하고 설계를 해보고 직접 확인까지 하는 것이 목표이다.

2. 설계 내용

2.1 하드웨어 구성

2.1.1 센서부

|  |  |
| --- | --- |
| 전자기기, 회로이(가) 표시된 사진  자동 생성된 설명  Figure 1. 센서부(상) | 전자기기, 실외이(가) 표시된 사진  자동 생성된 설명  Figure 2. 센서부(하) |

2.1.1.1 Line Scan camera

|  |  |
| --- | --- |
| 실내, 전자기기, 벽, 사람이(가) 표시된 사진  자동 생성된 설명  Figure 3. Line Scan camera (Middle) | Figure 4. Line Scan camera (Left, Right) |

Left, Right에 쓰인 카메라는 왼쪽 오른쪽 차선을 판단하는 데에 쓰이며 가운데 카메라는 좀더 멀리 바라보며 차선을 한 번 더 인식한다. Line camera는 128개의 배열로 값을 받아오며 밝은 픽셀이 어두운 픽셀보다 값이 크다. 따라서 이를 이용해 라인을 탐지할 수 있다.

2.1.1.2 PSD(적외선 센서)

|  |
| --- |
| 실내, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 5. PSD(전면) |

이 센서는 5V를 공급해주고 입사각에 따라 전류가 변하며 출력전압을 내보낸다. 이 때 20cm~150cm를 감지할 수 있어 좀 더 일찍 차량이 장애물, AEB, 언덕 등을 판단할 것 같아 선택했다. 이 PSD의 출력 값을 MCU로 받아 일정 값 이상, 이하일 때를 나눠 AEB와 장애물을 판단한다.

2.1.1.3 엔코더

|  |  |
| --- | --- |
| 사람, 오토바이, 개체, 남자이(가) 표시된 사진  자동 생성된 설명  Figure 6. 엔코더(전면) | 실내, 오토바이, 검은색, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 7. 엔코더(상) |

Autonics 사의 E30S Series 로터리 엔코더를 이용하여 차량의 속도를 측정하는데 이용하였다.

2.1.2 전원부

|  |  |
| --- | --- |
| 전자기기, 회로, 테이블, 실내이(가) 표시된 사진  자동 생성된 설명  Figure 8. 전원부(전면) | 전자기기, 회로이(가) 표시된 사진  자동 생성된 설명  Figure 9. 전원부(후면) |

모터드라이버를 사용하게 되면서 보드에서 나오는 VCC를 직접적으로 사용하는 것이 불가 해져 배터리에서 나오는 7.2V를 레귤레이터를 이용해 각각 3.3V. 5V로 변환시킨 후 이를 센서부로 전달한다.

2.1.3 구동부

|  |
| --- |
| **Figure 10. 구동부(전면)** |

DC모터를 사용하여 차량이 앞, 뒤로 나아갈 수 있도록 함.

2.1.4 조향부

|  |  |
| --- | --- |
| 개체이(가) 표시된 사진  자동 생성된 설명  Figure 11. 조향부(상) | 머리장식, 헬멧, 의류이(가) 표시된 사진  자동 생성된 설명  Figure 12. 조향부(하) |
| 오토바이, 검은색이(가) 표시된 사진  자동 생성된 설명  Figure 13. 조향부(측면) | |

장애물이나 AEB등의 경우 각각의 상황에 맞게 바퀴의 방향이 바로 바뀔 수 있도록 하는 Servo Motor를 찾고 설치하였다.

2.1.5 전체외관

|  |  |
| --- | --- |
| 실내, 천장, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 14. 전면(1) | Figure 15. 후면 |
| Figure 16. 측면(1) | 실내, 벽이(가) 표시된 사진  자동 생성된 설명  Figure 17. 측면(2) |
| 재봉틀, 기기이(가) 표시된 사진  자동 생성된 설명  Figure 18. 상부 | Figure 19. 전면(2) |

3D프린터를 이용한 틀에 보드와 모터드라이버, 전원부, 센서부를 쌓아 올리는 구조로 설계하였고 안정적인 동작을 위해 MCU 보드와 모터드라이버를 결합하였다. 또한 Line Scan camera와 PSD의 경우 기둥위에 3D프린팅을 붙여 위쪽의 판과 기둥에 각각 위치하도록 하였다.

2.2 소프트웨어 구성

2.2.1 소프트웨어 구성1

2.2.1.1 Core1

라인스캔카메라는 데이터를 읽는 주기를 조절해 Raw Data의 크기를 조절할 수 있다. 때문에 주변 환경의 빛의 밝기에 따라 데이터 읽는 주기를 조절하도록 하였으나 주기를 길게 설정할 경우 Scheduler의 다른 함수들의 주기도 바뀌게 된다. 따라서 새로운 Scheduler를 만들어 Core1에서 라인스캔카메라 함수만 독립적으로 실행하도록 하였다.

2.2.1.2 Line Detect

Differential Filter와 Gaussian Filter를 씌운 데이터 값은 흰색과 검은색의 경계에서 양수와 음수 두 방향으로 큰 값이 나타난다. 이때 가장 높은 값을 가지는 Peak를 라인으로 인식하였다. 이 방식은 세 카메라 모두에 적용되어 왼쪽 카메라에 인식된 라인을 왼쪽라인, 오른쪽 카메라에 잡힌 라인을 오른쪽라인으로 판단하게 하였다. 가운데 카메라의 경우 시야각이 넓어 절반으로 나눠 각 라인을 인식하였다.

2.2.1.3 Servo steering

Servo Motor 제어에 두가지 정보를 이용하였다. 첫번째는 차선의 곡률이다. 이 값은 가운데 카메라는 먼 곳을 확인하고 양쪽 카메라는 가까운 곳을 확인해서 얻은 라인 값의 차이를 이용해 얻어냈다. 두번째는 차선 안에서의 모형차 위치다. 이 값은 양쪽 라인의 중앙을 차선의 중앙으로 판단해 차선의 중앙과 모형차 위치의 차이를 이용해 얻어냈다. 또한, Simulink 상의 Lane Controller Stanley 블록을 사용하여 제어를 구현하였다. 추후, 더 안정적으로 제어가 가능한 방법을 채택할 것이다.

2.2.1.4 Crosswalk, Dotted Line Detect

일반 주행 중 횡단보도가 나타나면 라인이 여러 개 존재하는 것처럼 Peak가 여러 개 나타나게 된다. 이때 일정 값 이상의 Peak의 수를 세어 일정 개수 이상이면 횡단보도로 판단하였다.

라인을 인식할 때 일정 값 이상의 Peak가 없으면 라인이 없다고 판별하는데 라인 인식과 라인 미인식이 일정 횟수이상 반복되면 점선이라고 판단하였다. 이를 통해 점선이 오른쪽에 있으면 왼쪽레인, 왼쪽에 있으면 오른쪽레인에 있다고 판단하였다.

2.2.1.5 Obstacle Detect

아래쪽 PSD와 나머지 두개의 기둥에 달린 PSD의 값을 이용해 언덕, 장애물, AEB를 구분하였다. 우선 아래쪽 PSD의 값이 일정거리내에 들어올 때 나머지 두개의 기둥에 달린 PSD의 값이 일정거리보다 가까우면 장애물 혹은 AEB라고 인식한다. 이때 모형차가 1차선을 주행하고 있는지, 2차선을 주행하고 있는지에 따라 장애물과 AEB를 구분하였다. 또한 아래쪽 PSD의 값이 일정거리내에 들어와도 나머지 두개의 기둥에 달린 PSD의 값이 일정거리보다 멀면 언덕으로 인식해 무시하였다.

2.2.1.6 Obstacle Avoidance

장애물을 피하기 위해 두가지 정보를 이용했다. 하나는 반대쪽 차선의 실선이고 나머지 하나는 점선이다.

우선 장애물을 인식하면 반대 차선으로 가도록 Servo 모터를 회전시켰다. 이 회전 각도를 반대쪽 차선의 실선이 보일때까지 진행하였고 실선이 보인 이후에는 Servo를 반대방향으로 회전하였다. 그리고 점선이 일정범위내에서 검출되면 기본주행모드로 전환하였다.

2.2.1.7 PID, AEB

Simulink를 이용해 종방향과 횡방향 PID 제어를 구현하였고 모형차의 속도를 원하는 속도로 유지하도록 제어하였다. 이를 통해 언덕에서도 일정한 속도가 유지되었고 AEB 장애물을 감지할 경우 약간의 역회전이 걸려 빠르게 정지하도록 하였다.

3. 주요 장치 이론 및 적용 방법

3.1 Line Scanning

3.1.1 Filter

라인스캔카메라로 받은 Raw Data는 카메라 특성상 가운데 부분을 더 밝게 인식한다. 이를 무시하고 라인을 검출하기위해 Differential Filter를 사용하였다.

|  |
| --- |
| Figure 20. <초록색 : Raw Data, 빨간색 : 미분필터 적용 후> |

하지만 이 경우 노이즈가 심해 라인을 정확히 검출하기 어려웠고 때문에 Gaussian Filter를 추가적으로 사용해 노이즈를 줄였다.

|  |
| --- |
| Figure 21. <초록색 : Raw Data, 빨간색 : 미분필터, 가우시안필터 적용 후> |

3.2 Obstacle Detect

3.2.1 ADC 값을 이용한 거리계산

|  |  |
| --- | --- |
| 텍스트, 지도이(가) 표시된 사진  자동 생성된 설명  Figure 21. datasheet상의 그래프 | Figure 22. 직접 구한 거리 함수 그래프 |

**‘digitizer’이라는** 프로그램을 이용하여 데이터 시트의 그래프의 값을 직접 얻어 식을 찾으려고 했으나 IR 센서를 직접 연결하게 되면 노이즈가 생겨 캐패시터를 연결해 노이즈를 줄였다. 이렇게 설계한 회로를 이용해 얻어진 ADC값을 10번을 측정하여 이들의 평균값을 내고 최댓값을 찾고 그 때의 거리 값보다 더 큰 값들에 해당하는 ADC값을 가지고 역함수를 구해 거리함수를 구했다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 거리(cm) | 5 | 10 | 15 | 20 | 25 | 30 | 35 |
| Average(V) | 1.18071 | 1.186755 | 2.30159 | 2.094625 | 1.931625 | 1.739925 | 1.52259 |
| 거리(cm) | 40 | 45 | 50 | 55 | 60 | 65 | 70 |
| average | 1.326495 | 1.185592 | 1.076557 | 0.955678 | 0.86569 | 0.805128 | 0.745422 |
| 거리(cm) | 75 | 80 | 85 | 90 | 95 | 100 | 105 |
| average | 0.709402 | 0.666667 | 0.602565 | 0.601344 | 0.558609 | 0.537852 | 0.482906 |

3.3 Encoder

3.3.1 Median Filter

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 300.068 | 382.237 | 382.237 | 347.974 | 344.104 | 343.667 | 343.618 | 343.612 | 343.612 | 343.612 | 343.612 |
| 299.512 | 338.631 | 343.049 | 343.049 | 343.548 | 300.061 | 338.693 | 343.056 | 300.006 | 338.687 | 338.687 |

<Motor0Vol을 0.3으로 유지했을 때 시간에 따른 Encoder 속도 값>

Encoder의 속도 값은 많은 노이즈가 포함되어 있다. 주로 일정 값을 중심으로 일정 범위내의 값이 검출되는데 낮은 값과 높은 값을 노이즈로 판단하기 위해 Median Filter를 사용했다.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 300.608 | 343.612 | 343.612 | 343.612 | 343.612 | 343.618 | 343.667 | 344.104 | 347.974 | 382.237 | 382.237 |

<Median Filter를 씌운 후 Encoder 속도 값>

3.4 Smart Cruise Control

3.3.1 Speed Control System



Figure 23. Simulink Discrete PID Controller Block Equation

임의의 PWM값을 DC 모터에 주입하여 인코더에서 검출되는 속도를 확인한 결과, 값이 일정하게 도출되지 않는 것을 확인하였다. 특히, 주행코스 중 하나인 높은 언덕코스는 어지간히 강한 PWM 값이 아니면 통과하지 못하였다. 지나치게 빠른 속도로 주행 시 라인 검출 및 주행 자체에 문제가 생겼고, 이를 해결하기 위하여 PID 제어로 정밀한 속도 제어를 하였다.

3.3.2 Lane Keeping Control System

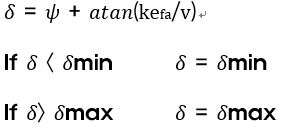
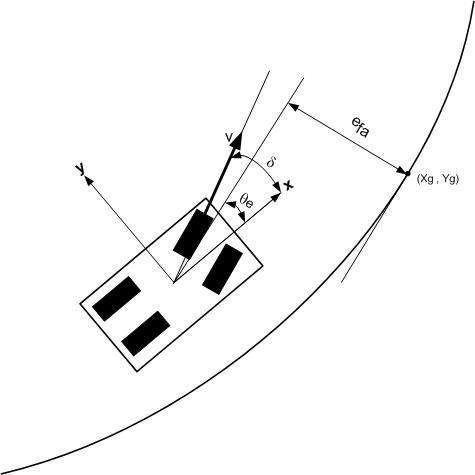


Figure 23. Lateral Controller Stanley

카메라로 도출한 라인 값들의 위치 차를 이용하여 차량의 현재 가로축 위치를 도출해냈다. 이를 이용하여 차량이 도로의 중심을 유지하며 나아갈 수 있도록 Simulink의 Lateral Controller Stanley 블록을 사용하였다. 이 블록은 Stanley Method를 사용하여 주행 경로를 이탈하지 않도록 도와준다. 현재 자동차의 위치(x,y), 경로의 위치(Xg, Yg), 현재 자동차의 주행 방향(Θe)와 향후 경로 상의 주행 방향, 그리고 차량의 속도(v), 그리고 servo 모터의 최대 각을 계산하여 차량의 방향을 servo 모터의 값으로 도출해준다.(k는 속도 등으로 servo 모터에 가해지는 힘의 비례 상수)

주행 방향은 차량 전방의 Lane Offset(라인 간의 위치 차이)을 PID 제어하여 구하였다. 주행을 해보았을 때, k의 값이 클수록 servo 모터의 각이 급격하게 변하고, 주행에 자체에 문제가 생기었다. 이에, k 값을 낮추며 적절한 k 값을 구하여 안정되게 주행이 가능하도록 구현하였다.

4. 결론 및 토의

서두에서 밝힌 두가지 목표에 대한 결론은 아래와 같다.

차선 인식의 경우 일반적인 1차선과, 2차선 그리고 횡단보도와 곡선으로 크게 나눠볼 수 있었다. 2차선의 경우 가운데 차선이 점선인 것, 횡단보도는 굵은 선이 여러 개인 것, 곡선은 곡률이라는 각자의 특징들을 가지고 판단을 하는 조건으로 세웠다. 점선의 경우에는 라인 인식과 미인식의 반복 횟수, 횡단보도의 경우 라인이 여러 개가 존재하는 것처럼 보이는 것, 가운데 Line Scan camera와 양쪽의 Line Scan camera간의 거리 차를 이용해 곡률을 각각 계산하였다. 비록 조건을 구하는데 있어서 카메라의 값을 어떤 부분부터 참값이라고 해야 하는지에 대한 고민이 있었으나 이는 여러 번의 실험을 통해 점점 정확도를 높여갔다.

먼저 장애물 회피, AEB, 언덕의 경우 적외선 카메라를 3개를 달아 각자에 해당하는 알고리즘을 찾기 위해 특징을 찾아보았고 이론적인 값이 아닌 잠시 다른 값이 나올 수 있기 때문에 함수를 두 번 실행시켜 확인하는 방법을 사용했다. 따라서 장애물을 피해야 하는 경우 여러 번의 실험을 거친 servo motor의 조향각대로 움직이고 AEB의 경우 알고리즘에 충족된다면 서서히 속도를 줄여 멈추며 언덕의 경우 속도를 높이는 방향으로 차량의 속도와 방향을 조절하였다.

이번 대회를 준비하며 초기에는 라인, 장애물 인식은 ccd카메라를 이용하고 Line Scan camera와 PSD센서를 보조수단으로 사용하는 방식으로 설계해 두었으나 MCU가 TC275로 바뀌면서 LCD가 사라져 ccd카메라로 받아오는 값을 확인할 수 없게 되어 사용하지 못하였다. 또한 ccd 카메라로 받아오는 방대한 정보를 처리하는 속도에 대해 의문점이 생겨 이 부분은 대회 이후에 실험을 더하여 추후 대회에 이용해야 할 것 같다. 더불어 라이다 센서를 장애물 회피 알고리즘에 적용시키려 했으나 데이터를 처리하는 과정에 문제가 있어 이 역시 추후에 더 공부가 필요할 것 같다.

추가적으로 수업시간에 cmd를 이용한 단편적인 결과를 확인하는 코드를 작성해봤다면 이제는 MATLAB이나 serial plot등을 이용해 결과를 그래프로 바꿔서 확인해 보거나 하드웨어를 직접 작동시켜 봄으로써 전체적인 상황을 판단하는 방법을 배웠다. 또한 항상 원하는 결과를 얻기 위해 코드를 수정해 왔다면 하드웨어를 다루다 보니 원하지 않는 다른 값들도 얻어보면서 이론과 실제가 차이가 나는 부분을 피할 수 없기에 어떻게 줄여갈 것인지에 대한 고민을 하고 이를 해결하는 방향으로 코드를 수정하는 법을 배웠다. 이는 실제로 트랙을 제작하여 자동차를 굴려보면서 직접 세운 알고리즘의 문제점을 발견하며 고쳐 나가면서 최대한 완성해가려고 노력했다. 모두가 함께 차량을 만들어 가는 것이기 때문에 분명 각자 맡은 부분이 있을 것이고 이에 대해 부족한 점이 있다면 모두가 함께 고민해 해결해 나가는 과정을 통해 부족한 점을 채우고 역량을 이론에서 실전으로 키우는 큰 계기가 된 것 같다.

[첨부양식 1]

**전자 부품 목록**

|  |  |  |  |
| --- | --- | --- | --- |
| **제조사** | **부품명** | **수량** | **사용목적** |
| TAOS | TSL1401-DB | 2 | Line Scan camera., Lane을 탐지한다. |
| TAOS | TSL1401-CL | 1 | Line Scan camera., Lane을 탐지한다. |
| Sharp | GP2Y0A02YK | 3 | IR 거리센서. 장애물, 언덕, AEB를 판단. |
| Tower-Pro | MG995 | 1 | 차량 조향 조절 |
| Autonics | E30S Series | 1 | 엔코더, 차량의 속도 계산 |
| REEDY | AAK694 Wolf Pack 7.2V 3000mAh Ni-MH | 1 | 전원 공급 |
| Mabuchi motor | DC motor(IFX007T) | 1 | 차량 구동조절 |
| Infineon | Motor driver  (IFX007T) | 1 | Servo motor와 DC motor의 방향과 속도 제어를 위해 사용 |
| Infineon | TC275 | 1 | Main board로써 data를 읽고 분석 등 담당. |
| Microchip Technology | LM2575- | 2 | 일정전압 유지 |
| Tower-Pro | MG995 | 1 | 서보혼 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[첨부양식 2]

**차체 변경 목록**

**\*변경 내용 및 변경 사유**

|  |  |
| --- | --- |
| **변경내용(사진 포함)** | **변경사유** |
| **바닥, 실내, 앉아있는, 자동차이(가) 표시된 사진  자동 생성된 설명**  **사람, 실내, 전자기기이(가) 표시된 사진  자동 생성된 설명** | 1. **변경사유**   **3D 프린팅을 설치하기 위해 구멍 뚫음**   1. **변경 근거 조항** 2. 제4조 (차량 키트) 3-②   부품 장착을 목적으로 플랫폼에 구멍을 뚫는 것은 허용되지만 그 이외에 목적으로 플랫폼을 절단하거나 변형하는 것을 금지한다.   1. 제4조 (차량 키트) 3- ④   케이스, 받침대 제작은 가능하나 검차 시에는 제거할 수 있어야 한다. |
| **가위이(가) 표시된 사진  자동 생성된 설명** | 1. **변경사유**   **서보모터 장착**   1. **변경사유 근거 조항** 2. 제4조 (차량 키트) 3- ③   엔코더 장착 혹은 조향부의 서보모터 장착을 목적으로 차체 일부를 절단하는 것은 허용하지만 그 이외의 목적으로 플랫폼을 절단하거나 변형하는 것을 금지한다.   1. 제4조 (차량 키트) 3-②   부품 장착을 목적으로 플랫폼에 구멍을 뚫는 것은 허용되지만 그 이외에 목적으로 플랫폼을 절단하거나 변형하는 것을 금지한다. |
| **오토바이이(가) 표시된 사진  자동 생성된 설명** | 1. **변경사유**   **엔코더 장착**   1. **변경 근거 조항** 2. 제4조 (차량 키트) 3- ③   엔코더 장착 혹은 조향부의 서보모터 장착을 목적으로 차체 일부를 절단하는 것은 허용하지만 그 이외의 목적으로 플랫폼을 절단하거나 변형하는 것을 금지한다. |
| **실내, 자동차이(가) 표시된 사진  자동 생성된 설명** | 1. **변경사유**   **나사가 걸려 플랫폼 절단**   1. **변경 근거 조항** 2. 제4조 (차량 키트) 3-②   부품 장착을 목적으로 플랫폼에 구멍을 뚫는 것은 허용되지만 그 이외에 목적으로 플랫폼을 절단하거나 변형하는 것을 금지한다. |