캡스톤 설계 최종 보고서 요약본

가. 연구의 필요성

RC 카의 경우 단순 조종을 통한 레이싱 경주 외엔 별다른 콘텐츠가 없다. 최근 이런 장난감들로 취미를 즐기는 성인들이 늘어남에 따라 색다른 콘텐츠가 개발 된다면 상업적인 이득을 취하거나 시장에서의 RC 카의 입지가 더욱 확고해질 것이다. 이를 위해 우리가 채택한 콘텐츠는 고전 게임 중 하나인 “Pac-Man”을 AR 과 접목하여 RC 카를 통해 구현한다면 충분히 상업성을 가질 수 있을 것이라 예상했다. 게임을 플레이 시 사람이 불편함을 느끼지 않는 일반적인 Frame Rate 는 25 정도로 알려져 있다. 허나 게임에 필요한 기능들을 구현할 때 사용될 컴퓨터 비전 알고리즘들은 정확도는 높으나 실행 시간이 오래 걸려 우리의 목표를 달성하기에 적합하지 못하다. 그러므로 Target Detecting 이나 UI 처리에 사용될 알고리즘으로 기존의 알고리즘에서 개선된 성능을 가지는 새로운 알고리즘을 개발할 필요가 있다. 결과적으로 필요한 기능에 적합한 방식의 알고리즘 구현을 목표로 한다.

나. 연구 목표 및 방법

가) 연구 목표

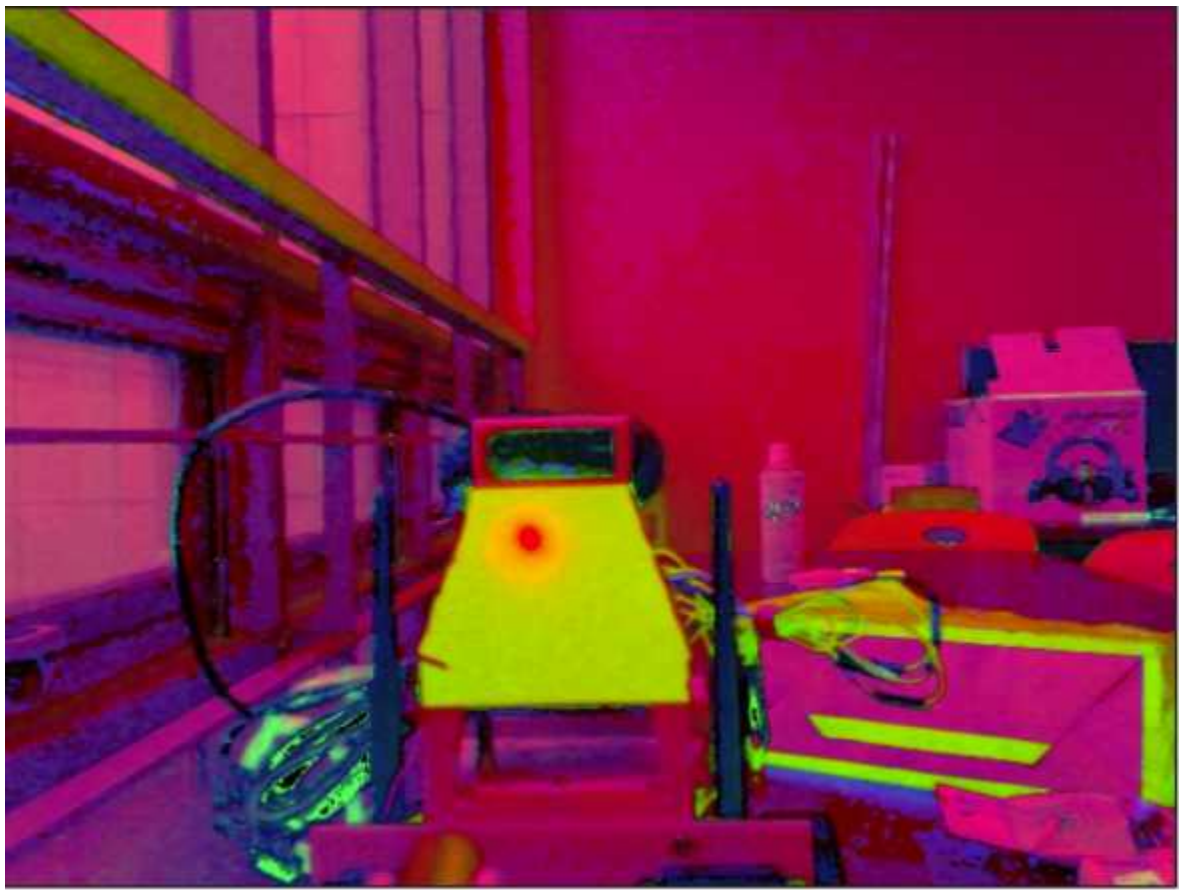
구분	내용
최종 목표	RC car Game development with Image Processing using Jetson Nano
세부 목표 및 내용	1. Edge Detection에 일반적으로 사용되는 Canny Edge Detection의 경우 정확도는 높으나 MCU(라즈베리파이4)에서 실행 시 일반적으로 걸리는 시간이 한 프레임 당 300ms정도 이다. 이는 목표로 한 Frame Rate인 초당 25프레임이 되기엔 너무나 느린 속도이다. 정밀한 정확도를 요하지 않으므로 정확도는 비교적 떨어지나 좀 더 빠른 알고리즘을 구현해야했고 hsv값을 기준으로 타깃을 구별해주는 간단한 알고리즘을 이용하여 한 프레임 당 20ms정도 걸리는 알고리즘을 구현. 2. hsv값을 이용하여 타깃을 검출 시 비슷한 색의 물체를 전부 인식하여 에러가 굉장히 많이 발생한다. 조건부 UI 출력을 위해 일정 수준 이상의 정확도를 필요로 한다. 이에 따른 Noise Canceling 방법에 첫째로 Erosion을 통해 세세한 오류들을 제거해주었고 둘째로 비교적 크기가 큰 에러의 경우 Contour 점의 개수를 통하여 구별해주는 방식을 채택하여 대부분의 환경에서 정확한 검출에 성공. 3. 후방 충돌 회피를 구현 시 Arduino와 Jetson 간의 통신 delay를 고려한 여유 제동거리를 확보해야한다. 이를 위한 제한 값을 trial and error 방식으로 찾아내어 구현. 4. 역할에 따른 동작 구현을 위한 각 차량의 상태를 서버 통신을 통해 공유하는 방식으로 TCP Socket Server & Multi-Client 통신 방식을 채택했고 이를 실패하여 1:1로 각 차량을 각각의 서버와 연결하는 방식으로 시뮬레이션에 맞게 임시로 구현. 5. AR기술을 통한 가상 Object 생성과 트랙의 전체적인 상황을 보여줄 미니맵 구현을 위해 ROS를 사용하기로 했으나 미완성.

다. 연구 결과

가) 세부 목표별 연구 결과

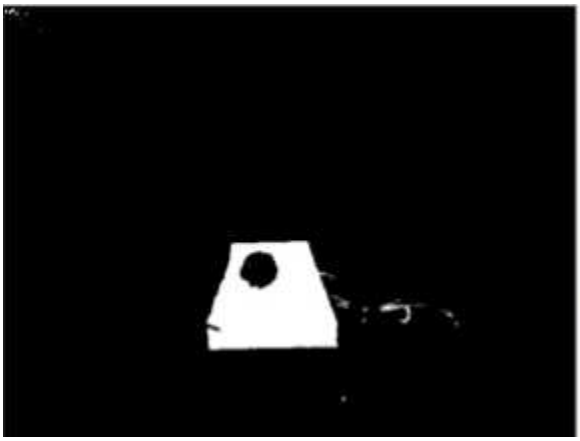
- Target Detection

• Transform RGB to HSV



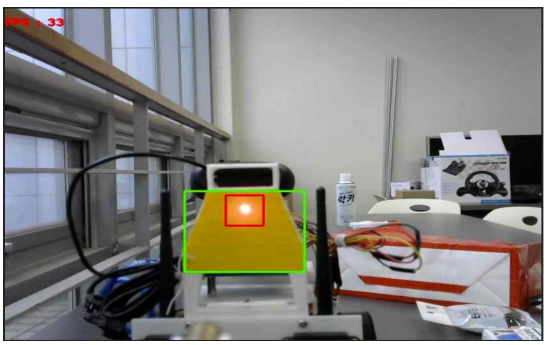
색을 특정하기에 RGB 값을 이용 시 복잡하다고 판단했다. 색의 종류를 H값으로 채도를 S로 밝기를 V로 표현해주는 HSV 값을 사용한다면 조금 더 분류에 용이할 것이라 판단하여 이미지를 우선 HSV이미지로 변환해주었다.

• Thresholding and Detecting




왼쪽 이미지의 경우타깃에 사용될 색에 해당하는 특정 HSV 값을 기준으로 하여 Thresholding 해준 후 타깃 옆의 짜잘한 에러가 검출됨을 확인할 수 있다. 하지만 오른쪽 이미지에서 점의 개수를 통해 오류들은 타깃 후보에서 제외해줌으로써 정확히 타깃만을 검출해낸 모습이다. 기존의 경우 해당되는 contour들을 전부 출력해주어 타깃 검출에 있어 어려움을 겪었으나 contour에 포함된 점의 개수로 이를 구별해줌으로써 빠르고 정확하게 타깃을 검출하는데 성공했다.

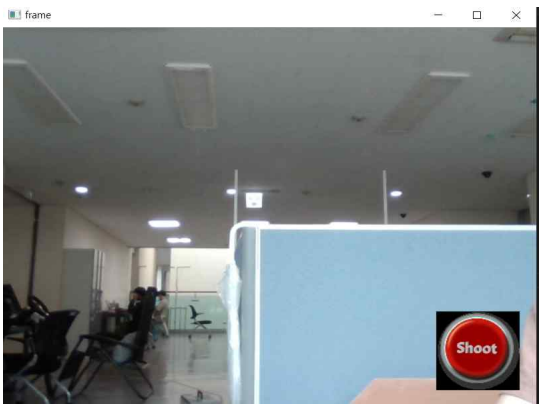
• over 25 Frame Rate

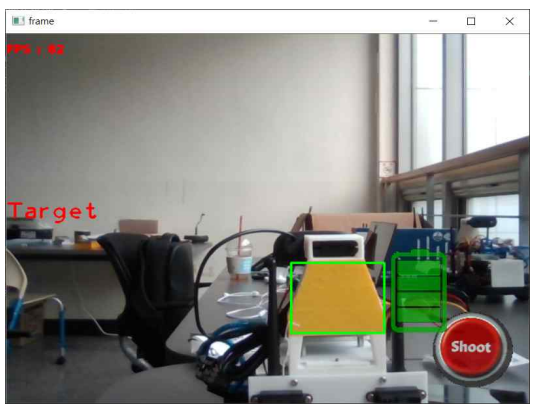



목표했던 25보다 높은 33을 유지하는 것을 볼 수 있다. 기존의 Edge Detection 알고리즘이나 SIFT를 이용 시 약 3~5정도의 프레임을 유지했다. 이는 게임 플레이에 있어 상당히 불편함을 준다. 그러나 이미지 처리만을 했을 시엔 7~80을 보여주는 반면 서버 통신까지 합쳤을 경우 WIFI 상태에 따라 프레임이 30~50으로 크게 변동하는 모습을 보였다.

- UI

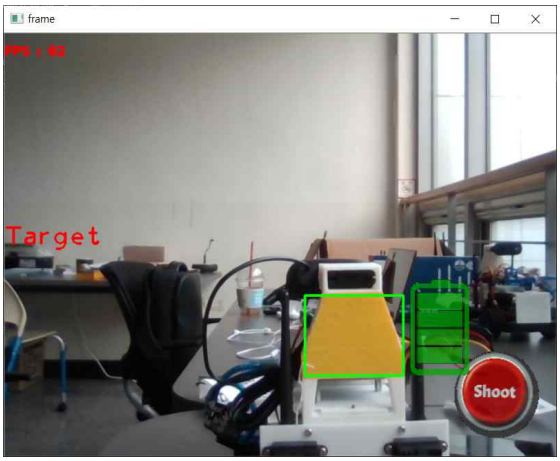
• Mask





UI를 입힐 때 사각형의 형태로 덧씌우기 때문에 필요 없는 배경 부분이 까맣게 나와 미관상 좋지 않다. 이를 해결하기 위해 원하는 부분에만 가중치를 주는 Mask를 생성하여 필요한 부분만을 출력하는 모습이다.

• different design by role

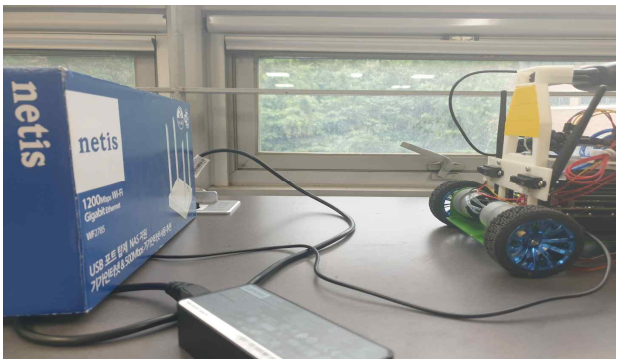


차량의 역할에 따라 UI에 차이를 주었다. 왼쪽의 경우 술래의 UI이고 오른쪽의 경우 도망자의 UI이다.

위 사진의 경우 술래에게 피격 당했을 시의 화면이다. HP 게이지가 천천히 줄어들고 화면이 전체적으로 빨개진다.

- Collision Avoidance

• Serial Communication




```
Start Comm
/dev/ttyACM0 ... Connected
164       188
199       251
164       189
241       192
169       182
167       178
176       186
165       189
164       189
comm exit
```

PSD센서가 Arduino를 통해 거리에 따른 값을 읽어낸다. 이를 시리얼 통신을 이용하여 Jetson으로 정보를 전달해준다. 약 30cm의 거리에서 센서 값이 180 정도의 값을 출력하는 것을 확인했다.

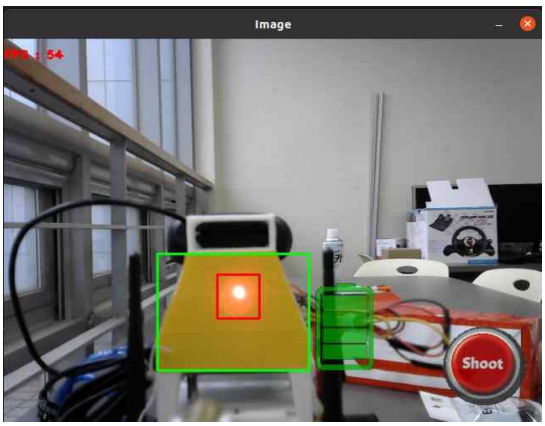
• pwm

```
axis = j.ax_map[number]
fvalue = value / 32767.0
if L > 250 or R > 250 :
    self.motor._pca.channels[0].duty_cycle = int(0xFFFF * (0.8 * throttle_gain))
    self.motor._pca.channels[1].duty_cycle = 0xFFFF
    self.motor._pca.channels[2].duty_cycle = 0
    self.motor._pca.channels[3].duty_cycle = 0
    self.motor._pca.channels[4].duty_cycle = int(0xFFFF * (0.8 * throttle_gain))
    self.motor._pca.channels[7].duty_cycle = int(0xFFFF * (0.8 * throttle_gain))
    self.motor._pca.channels[6].duty_cycle = 0xFFFF
    self.motor._pca.channels[5].duty_cycle = 0
```

제동거리를 생각하여 여유롭게 기준 값을 설정해주기로 했다. 실험적으로 구한 250을 기준으로 pwm에 변화를 주어 적절한 속도로 전방으로 이동시켜 후방 충돌을 피해준다. 허나 psd센서의 경우 주변 환경의 변화(빛의 세기 등)에 따라 성능의 차이를 보였다.

- TCP Socket Server Communication

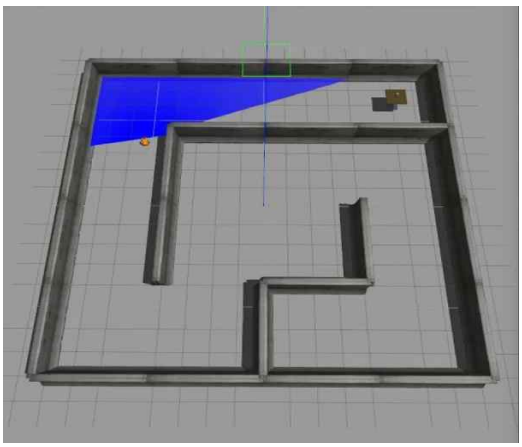
```
jetson@nano-4gb-jp451:~/jetracer/capstone$ python3 main.py
libpng warning: iCCP: known incorrect sRGB profile
car generated...
control
Start Comm
/dev/ttyACM0 ... Connected
server start
wait.
gamepad connected...
wait.
Connected by  192.168.43.246 : 63610
```

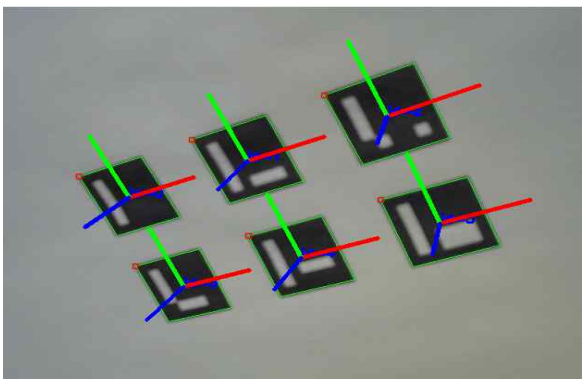


Jetson에서 이미지 데이터를 노트북 서버로 보내 영상을 띄우는 모습이다. 위 영상의 경우 Chaser Mode 차량의 영상이다. 도망자 차량의 target의 위치 정보를 바탕으로 버튼 및 HP 등 정보를 UI로 나타내준다. 원래는 레이저가 맞았다는 것을 레이저를 쏜 차량은 이미지 정보를 통해 인지할 수 있으나 정작 맞은 차량은 스스로 인지할 수 없다.

결과적으로 맞았을 때의 UI변화나 기능 구현을 위해 이러한 정보를 이를 서버 통신을 이용해 데이터를 공유해줄 필요가 있었다. 각 차량을 하나의 서버로 연결해주는 TCP socket server & Multi-client 방식으로 구 현할 계획이었으나 현재 서버로 사용하고 있는 가상환경은 포트 포워딩을 사용하지 않으면 외부에서 접근할 수 없다는 사실을 인지하지 못해 실패하여 1:1 통신만을 구현해 임시로 표현해주었다.

• AR with ROS





ROS의 world data와 트랙을 동기화하여 차량들의 실시간 위치와 도망자가 먹어야할 Coin 등의 위치를 동기화하여 실시간으로 보여주는 미니맵을 구현하고 AR Marker를 이용해 도망자가 먹어야할 Coin을 가상으로 생성하여 카메라에 띄워주는 것을 계획하였으나 기간 내에 완성하지 못했다.

나) 연구 결과 자체 평가

목표로 한 게임 콘텐츠에 있어 핵심 기능인 Target Detecting의 경우 기존의 Edge Detection 방식에 HSV 값을 기준으로 Thresholding하는 방법과 많은 Edge들 중 점의 개수를 통해 타깃을 특정해주는 방식을 추가하여 알고리즘을 간소화하였다. 이로 인해 프레임 등 초기 설정한 목표치를 충분히 달성했다.

Mask를 이용하여 덧씌울 UI 이미지에서 배경을 제거하여 UI의 완성도를 높였다. 그냥 씌울 경우 cv에서는 사각형의 형태로 이미지를 덧씌우게 되는데 이러한 이유로 필요 없는 부분은 검은색으로 표현되어 보기에 좋지 않다. 이를 해결하기 위해 가중치를 주는 방식을 채택하였는데 원하는 형태로 결과가 나왔다.

ROS를 이용한 가상 맵 자체는 구현을 하였으나 게임 재미에 큰 영향을 주는 요소라 생각되는 도망자가 먹을 Coin을 가상환경에 생성해주는 기능은 관련 지식이 부족해 기간 내에 완성하지 못했다. 이는 작품 전체의 완성도를 떨어트리게 됐다.

라. 고찰

- Steering의 불균형

>> Steering을 조절하는 서보모터의 throttle을 0으로 했을 경우에도 똑바로 가지 않고 옆으로 조금씩 휘어 정확한 방향 조정에 어려움이 있었다. Motor 자체의 문제인 것으로 판단하여 Offset값을 주어 어느 정도 보완하였으나 완벽하게 조정되진 않았다.

- 차량의 불규칙한 움직임

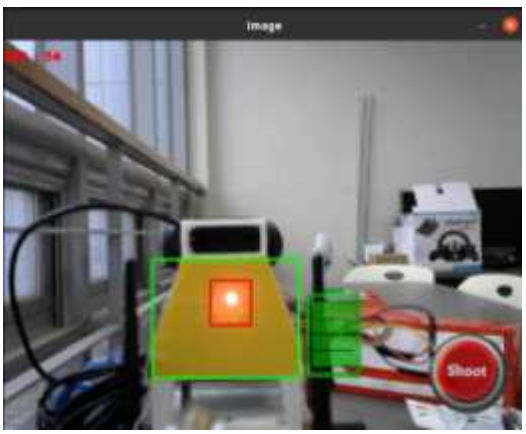
>> 차량이 방향전환을 할 때 바퀴가 헛돌거나 한쪽만 도는 경우를 볼 수 있었다. 적절하지 못한 Motor를 사용했거나 바퀴의 마찰력이 부족하여 이런 현상이 나타나는 것이라 판단하여 바퀴에 고무줄을 감아 시뮬레이션 해보았으나 여전히 Steering을 완전히 꺾어주는 경우 차량이 움직이지 않는 현상이 발생했다. 결과적으로 앞바퀴가 Steering을 최대로 주었을 경우 설계미스로 인해 전방의 범퍼와 접촉하는 것을 확인했고 모터의 pwm을 올려 일정 수준 해결하였으나 이로 인해 세밀한 속도조절에 어려움을 겪었다.

- 영상의 불안정한 실시간성 보장

>> 현재 차량의 경우 Jetson에서 카메라를 통해 받은 이미지를 TCP socket통신을 통해 게임 화면을 송출할 노트북으로 전송 후 출력하고 있다. 이 과정에서 약 0.1초 가량 영상이 밀리는 현상이 발생했다. TCP 통신 방식 중 고속 통신이 가능한 동기 방식을 채택하였음에도 불구하고 이러한 현상이 발생하는 것에 대해 이는 WIFI의 성능 문제라고 판단했다. 이를 해결하기 위해서는 조금 더 좋은 wifi를 사용하는 것 외의 방법을 찾지 못했다.

- 화면만으론 조종이 힘듦

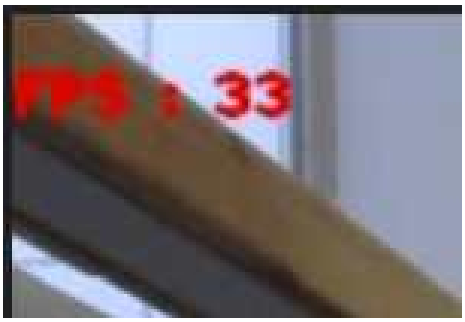
>> 아래 화면을 보면 후방과 측면이 보이지 않아 트랙의 주변 상황 등을 알 수 없다.



차량을 조종하는데 있어 필요한 정보가 불충분함에 더불어 위에 언급된 0.1초 정도 영상이 밀리는 현상이 겹쳐져 화면만으로 운전 시 커브가 많거나 복잡한 트랙에서 충돌이 잦았다. 추가적으로 카메라를 설치하거나 광각이 넓은 카메라를 사용한다면 해결할 수 있을 것으로 보인다.

- Detection 속도 향상

>> 아래 사진과 같이 초기 설정했던 목표치인 25보다는 높은 프레임이 나왔다.



하지만 이는 통신상태가 불안정할 경우 이를 보장하기가 힘들다고 판단했고 이를 개선하기 위한 방법을 고안해보았다. 첫째로는 통신 속도를 높이는 것이고, 둘째는 image처리 속도를 높이는 것인데 통신 속도의 경우 현재 상태에서 더 좋은 방법을 찾지 못해 후자를 선택해 조사한 결과 컴퓨터 비전에서 자주 사용되는 관심 영역만을 탐색하는 spatial image domain 방식을 사용한다면 처리 속도를 향상시킬 수 있을 것으로 예상된다.

마. 예산 및 결산

1. 아두이노 우노 외 4종 : 33,245원
2. 브레드보드 801 외 2종 : 19,990원
3. 진동센서 : 4,400원
4. 아두이노 우노 외 5종 : 49,770원
5. 점프선 외 3종 : 8,750

바. 참고문헌

[1] 임윤지, 김태훈, 김성호, 송우진, 김경태, 김소현. (2015). BMVT-M을 이용한 IR 및 SAR 융합기반 지상표적 탐지. 제어로봇시스템학회 논문지, 21(11), 1017-1026.

[2] https://link.springer.com/chapter/10.1007/978-1-4615-0913-4_11

[3] https://www.kiwi-electronics.nl/jetracer-ai-racing-robot-jetson-nano?lang=en

[4] https://bluexmas.tistory.com/970