창업연계공학설계입문

2조 AD프로젝트

수행 보고서

20191548 고강현
20191551 구형모
20160729 김태영

[목차]

Ⅰ. 개요

1. AD 프로젝트 계획

이번 학기 창업연계공학설계입문 수업을 수강하며 11주차의 과정에서 교수님께서 Xycar를 활용한 AD 프로젝트를 기말 과제로 부여하셨다. 교수님께서는 부여하시면서 도전할만한 주제를 가지고 프로젝트를 하셨는데 우리 조는 도전할 만하고 실용적으로 사용될 수 있는 주제를 생각하여 조원들은 서로 각자 사전 조사를 하고 브레인스토밍을 하였다. 서로 의논을 하며 결정한 AD 프로젝트의 주제는 주차에 관한 이슈를 다루기로 했다.

우리나라는 다른 나라들에 비해서 운전면허의 취득이나 운전에 대해서 관대한 편인데, 그 점은 주차에 대해서도 적용이 되는 문제이기 때문이다. 예를 들어, 일명 '김여사 주차'라고 불리는 우리나라에서 발생하는 불법 주정차, 주차 과정 중 추돌 사고 등의 주차에 있어 좋지 않은 문화는 우리 조에게 있어 자율 주행 주차에 관한 소프트웨어를 이번 AD 프로젝트를 통해 개발하게 된 계기가 되었다. 이러한 조원들의 활발한 논의와 브레인스토밍은 이 계기를 찾게 되어 자율 주행 주차 시스템(ADPS : Auto-Drive Parking System)을 제작하기로 결정했다.

그 계획을 토대로 우리가 제작하기로 한 자율 주행 주차 시스템(이하 ADPS) 는 기본적으로 시장에 나와 있는 다른 자동차들의 자율 주차 시스템과 같이 자동차가 운전자를 도와 조향, 핸들링을 직접 하여 주차하는 시스템이다. 우리 조는 ADPS를 Fully-Automated System으로 전자동이고 주야간, 운전면허를 취득하지 얼마 되지 않은 운전자들과 노약자들도 걱정 없이 사용할 수 있도록 설계하고자 했다. 또한, 사용자인 많은 운전자들이 주차에 불편함을 느끼는 상황인 일명 T자 주차로 알려져 있는 수직 주차, 평행 주차의 상황에 있어서도 케이스를 분리하여 주차를 수행할 수 있도록 개발을 계획했다.

우리 조는 다른 조에 비해서 1명에서 2명이 적은 상황이었기 때문에 계획 설계에 좀 더 힘을 쏟는 것이 중요한 것뿐만이 아니라 프로젝트 설계 및 구현에 있어 분업이 효과적으로 이루어져야하기 때문에 역할 분담을 정하는 것에 있어서도 어떤 부분을 담당할지 많은 고민을 했다. 역할 분담은 센서 / 카메라 담당, 조향 담당, 알고

리즘 로직 설계 담당으로 이렇게 3개의 부분으로 나누었다.
주제를 계획하면서 역할 분담을 계획했고 또한 일정에 있어서도 효율적으로 관리를 하고자 AD 프로젝트 계획 발표를 하기 전 주인 11주차에 있어서는 주제 선정을 위해 사전 조사 및 브레인스토밍, 12주차에 있어서는 Xycar의 센서 동작 점검, 13, 14주차에 있어서는 주차에 대한 알고리즘 구현, 15, 16주차에 있어서는 모든 코드를 구현하고 시연 동영상을 촬영한 다음 프로젝트 발표를 준비하기로 계획했다.

2. 하드웨어적 한계로 인한 프로젝트 수정

우리 조는 창업연계공학설계입문 강의 12주차에 이루어진 AD프로젝트 계획 발표 이후 교수님께서 지적해주신 후방 카메라의 부재로 인한 우리가 구현하고자 하는 ADPS를 완벽히 구현할 수 있는지에 대해 논의를 하게 되었다.
논의를 하며 메인보드에 있는 전방의 카메라를 임의로 떼다 후방에 부착하는 방법을 통해 ADPS의 기능 중 하나인 후방주차에 대응할 수 있다는 결론에 접근할 수 있었다.
하지만 물론, 기본적으로 전방의 카메라가 작동해야 주차할 라인을 검정할 수 있기 때문에 해당 방법을 구현할 프로젝트에 적용할 수 없다는 것이 문제가 되었다.
또한, 후방 카메라의 부재뿐만 아니라 하드웨어적인 문제 이전에 평행 / 수직 / 일반 주차에 대한 로직을 작성하였다. 카메라의 부재 때문에 센서를 이용하여 소스를 구현했다. 방식은 차가 직진하다가 주차장이라는 것을 인식하는 센서로 값을 받는다. 값이 들어오면 거기서부터 주차 공식을 따라서 앞으로 일정 시간 직진한 후 후진하다 후방 센서의 값이 45가 되었을 때 정차 하는 방식을 택했다. 하지만 이러한 방식은 자율 주차라고 하기에는 문제가 있었다. 그리고 실제로 차가 그런 방식으로 주차한다고 생각해 보았을 때 사람이 나오거나 옆에 차가 있거나 하는 변수가 발생하였을 때 대처가 불가능하다.
그래서 우리 조는 기존에 계획했던 프로젝트의 내용을 수정하여 주차 도우미를 구현하기로 하였다.

II. 설계 과정

주차 도우미를 하기에 앞서 기존에 구현하다 실패한 자율 주행은 가지고 가야 한다고 생각했다. 그래서 그냥 주차 도우미가 아닌 자율 주행이 되는 차량의 주차 도우미를 계획하였다. 첫째로 자율 주행을 하는 차를 주차할 영역을 알려주는 것을 만들 것이고 무료 주차장이 아닌 유료 주차라고 가정하고 들어오는 차량을 인식하고 그러한 차량이 나갈 때 가격을 알려주기로 하였다. 이러한 점들을 구현 하기위해서는 자동차의 움직임, 주차장 입출 차량 인식, 주차 가능 여부를 판단해야 한다.

1. 자동차 움직임 구현

일단 주차 도우미로 진행을 하기로 했기에 자동차가 주차장에 들어가서 주차를 하는 과정이 들어가야한다. 그러한 과정을 우리 조는 직접 차량을 움직여 주차를 하는 것을 선택하였다. 차량의 움직임을 간단하게 컨트롤러로 구현을 할 수 있지만 자율 주차를 실패한 시점에서 자율적으로 하지 못한다면 수동적인 부분들은 다 구현해 보는 게 좋겠다고 생각하였다. 그래서 cv2의 key값을 받아와 “w’,’a’’,s’’,d’로 차량을 제어하기로 계획하였다.

2. 주차장 요금 부과 시점 인식

주차장의 요금을 부과 하기 위해서는 들어오는 차량의 시간과 나가는 차량의 시간을 인식할 수 있어야 한다. 그러한 방법을 두가지를 생각 하였다. 차량 객체를 구현하여 생성자에 차량의 고유 번호를 겹치지 않게 랜덤으로 주고 그 번호를 이용하여 Qr코드로 주차장의 입구를 인식하여 차량의 입출력을 체크하는 방법을 생각하였다. 하지만 Qr코드를 인식해야 하는 부분에서 import를 해야한다는 것을 알았고 랜선을 꽂아 깔아보려 했으나 pip가 없어 실패하였다. 그래서 Qr코드가 아닌 색상으로 입구를 검출 해야겠다고 계획하였다.

3. 주차 가능 여부

차량이 주차장에 들어가면 주차되어있는 자리도 있고 추ㅊ주차장에 들어가면 주차되어있는 자리도 있고 주차가 되어 있지 않은 자리도 있다. 당연한 말이지만 주차는 차량이 없는 자리에 주차해야 한다. 처음에는 위에서 언급한 방법처럼 처음 계획은 Qr코드를 이용하여 이 자리에 차량이 있는지 없는지를 체크하려 하였다. 하지만 불가능 했기에 위의 주차장 인식처럼 색상을 이용하여 구분하기로 하였다. 차량이 없을 때의 없을 때를 색상으로 구분하여 차량이 없을 때 그 자리에서 자율 주행 자동차를 자율 주차 모드로 변경하여 주차를 시킬 계획이다.

Ⅲ. 구현 과정

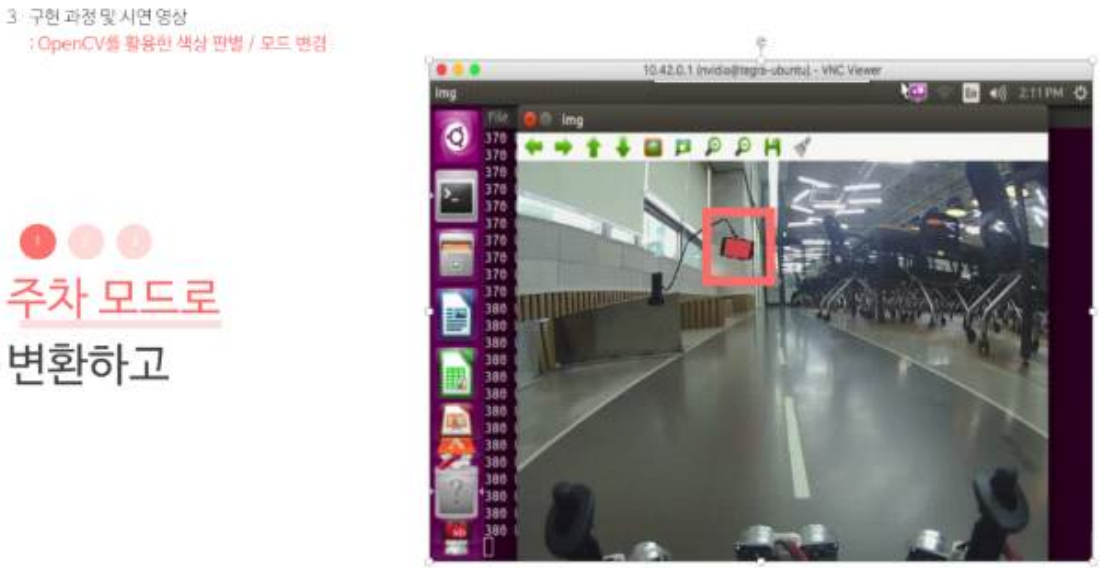
1. WASD 모드 구현

차량의 전후진과 조향을 수동으로 조작하기 위해 키보드 키입력을 감지한다. 그후, 정수 형으로 들어오는 키값을 매핑하여 조향을 하게 된다.

```
key = cv2.waitKey(1)
if key & 0xff == 27:
    break
if 81 <= key <= 84:
    if key == 82:
        speed = 120
    if key == 81:
        if angle - 10 >= 40:
            angle -= 10
    if key == 83:
        if angle + 10 <= 150:
            angle += 10
    if key == 84:
        #if prev_key != 84:
            # wheel_reset()
        speed = 65
else:
    speed = 90
drive(angle, speed)
```

차례로 cv2.waitkey(1)을 통해 반환된 keycode 81부터 84 는 하드웨어 키보드 방향키 상, 좌, 우, 하 에 해당하며 위 로직을통해 차량이 전후진 또는 좌,우로 조향을 하게 된다.

2. OpenCV를 활용한 색상 판별 / 모드 수정



위 사진과 같이 색상 판별은 주행 모드에서 주차 모드로 변경 할때 쓰이거나(빨간색), 주차를 마치고 요금을 정산 할 때(파란색) 필요하다. 이를 구현해내기 위해 OpenCV 를 활용하였으며, 다음과 같은 방식을 사용하였다.

2.1 HSV 이진화 (inRange)

HSV로 변환한 이미지의 H(Hue) 값은 다음 사진과 같은 값을 가진다. Hue 성분은 다음처럼 특정 색의 컬러가 일정한 범위를 갖기 때문에 분리해내기가 쉽다.

이와 같은 특성을 이용하여 OpenCV 내부함수인 inRange함수를 이용해서 특정 색상값을 가진 픽셀들만 추출해내는 과정을 거쳐서 인식률을 높이는 과정을 취한다.
예를 들자면, 빨간색을 검출 하고자 한다면 다음과 같은 과정을 거치게 된다.

```
lower_red = np.array([0, 50, 50])
upper_red = np.array([10, 255, 255])
mask = cv2.inRange(hsv, lower_red, upper_red)
```

2.2 Morphological Transformation

Morphological Transformation 이란 이미지의 형태 전환을 일컫는다. 이를 위해 2가지의 입력 값이 필요한데 보통 Frame과 Kernel 이다. Frame은 함수 내에 feed된 원본 이미지 이며, Kernel 은 이미지 변환을 결정하는 구조화된 요소 라고 할 수 있다. 이 과정에서는 두가지의 기법이 쓰이게 되는데 Erosion과 Dilation 이다.

원본

Erosion(침식) **Dilation**(팽창)

Erosion은 이미지를 침식시키는 과정으로 Foreground 가 되는 이미지의 경계부분을 침식시켜서 Background 이미지로 전환되어 가늘게 된다.
Dilation 은 이미지를 팽창시키는 과정으로 Erosion과 반대로 작동한다.

2.3 Contour

contour란 같은 값을 가진 값들을 하나의 선으로 연결하는 것을 뜻한다. 예를 들어 위도나 등고선 같은 수치를 가진 것 들을 선으로 연결 할 수 있다.

찾아낸 contourArea 중 Max 값을 찾은후, 일정 Radius 이상의 범위를 가진 색집합을 찾아내는 방식으로 인식하였다.

위와 같은 방법은 단순히 픽셀의 갯수를 세는것보다 인식률도 높고 화면내 어느 위치에서나 특정 크기 이상만 되면 인식하도록 구현할 수 있다는 장점이 있다.

3. 주차 차선 표시하는 주차 모드

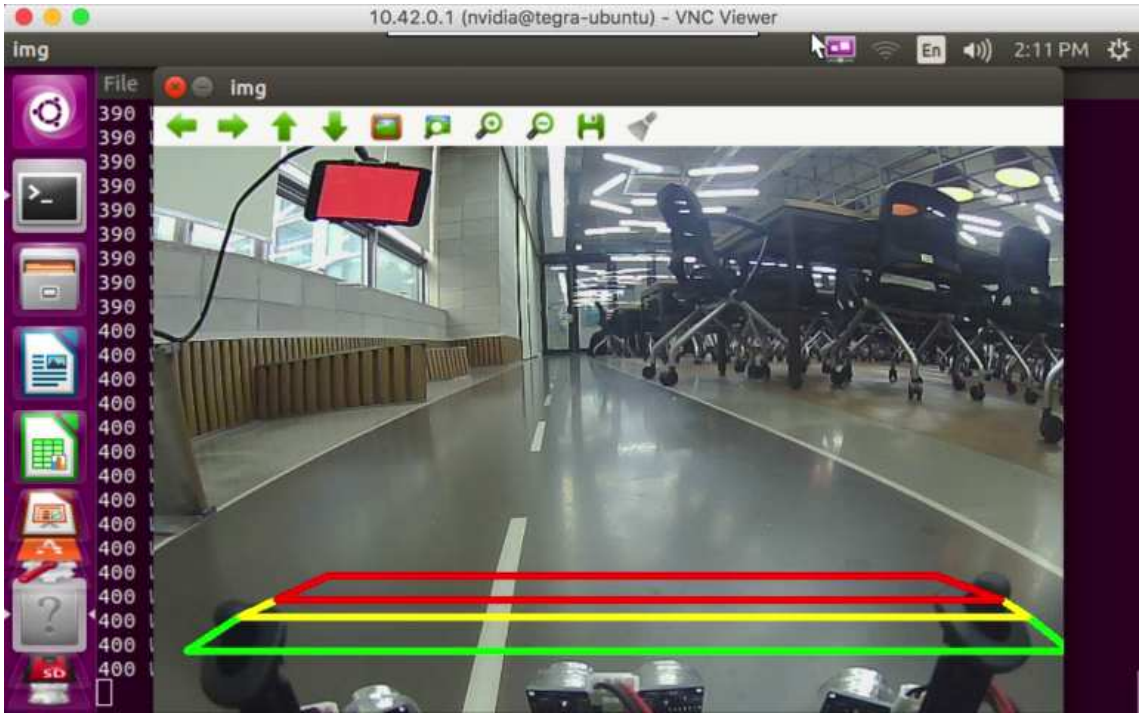
위의 색상 검출 소스로 주차장에 자리가 있는 지를 판단하여 boolean 형으로 받게 된다. 만약 빨간색을 검출하는 boolean 변수가 True가 되면 그 자리에는 주차된 차량이 없다는 뜻이므로 주차모드로 진입한다.

주차모드로 진입할때에는 park_lines()에 매개변수로 frame을 보내주어 frame에 주차선을 그려 return 한다. Return된 값은 기존에 imshow를 하던 frame에 저장하여 덮어씌운다.

```
22  def park_lines(frame):
23      # green
24      frame = cv2.line(frame, (23, 355), (645, 355), (0, 255, 0), 3)
25      frame = cv2.line(frame, (60, 331), (615, 331), (0, 255, 0), 3)
26      frame = cv2.line(frame, (60, 331), (23, 355), (0, 255, 0), 3)
27      frame = cv2.line(frame, (645, 355), (615, 331), (0, 255, 0), 3)
28      # yellow
29      frame = cv2.line(frame, (60, 331), (615, 331), (0, 255, 255), 3)
30      frame = cv2.line(frame, (86, 319), (596, 319), (0, 255, 255), 3)
31      frame = cv2.line(frame, (60, 331), (86, 319), (0, 255, 255), 3)
32      frame = cv2.line(frame, (596, 319), (615, 331), (0, 255, 255), 3)
33      # red
34      frame = cv2.line(frame, (86, 319), (596, 319), (0, 0, 255), 3)
35      frame = cv2.line(frame, (122, 302), (550, 302), (0, 0, 255), 3)
36      frame = cv2.line(frame, (86, 319), (122, 302), (0, 0, 255), 3)
37      frame = cv2.line(frame, (550, 302), (596, 319), (0, 0, 255), 3)
38
39      return frame
```

다음 사진과 같이 주차할 장소를 인식하면 자동으로 주차선이 그려진다. 원래의 계

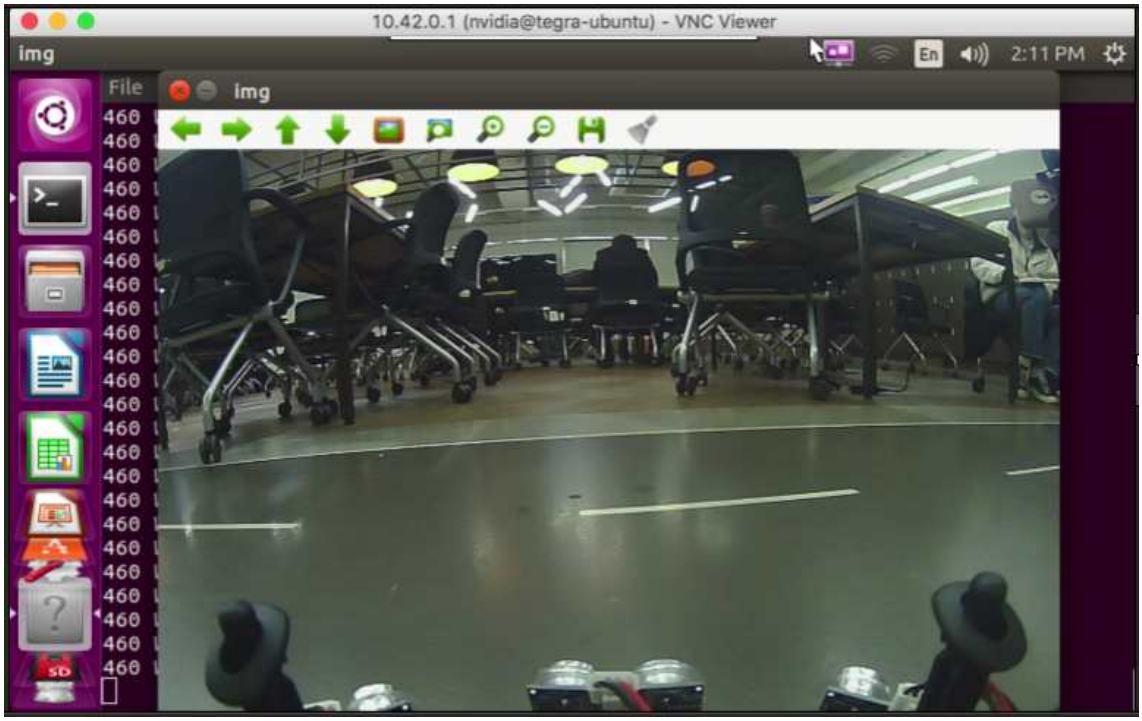
획은 후방 카메라에 그려져야 하지만 후방 카메라가 없으므로 앞에 그려지게 된다.



주차모드가 인식되어 주차선이 뜨면 자율 주차가 되는 것을 가정하여 수동으로 주차한다. 주차가 완료되면 기존의 계획대로면 주차장에 주차를 확인하는 센서로 인식하려 했다. 하지만 그러한 센서가 없기 때문에 직접 p키를 눌러 주차 완료를 알린다.

```
90          if key == 127:
91              p = False
```

P키를 누르면 다음사진과 같이 주차모드를 알리는 주차 도움 선이 사라진다.



4. 주차 요금 정산

현실 세계에서 운전자라면 주차 요금이 어떻게 정산될 지 알 것이다. 차량이 주차장으로 들어온 시간으로부터 나간 시간까지 주차장에서 시간 단위를 기준으로 하여 해당하는 시간에 요금을 곱하여 나온 출력값이 주차 요금이다.
운전자들은 주차 요금이 얼마나 나올 지 다들 생각을 하지만 일일이 신경써야 한다는 점에서 많은 불편함을 느낄 것이라는 발상에서 차량에서 주차 요금을 정산하여 사용자에게 알려주는 기능이 있다면 좋을 것 같다는 생각으로 주차 요금 정산 기능을 추가하였다.

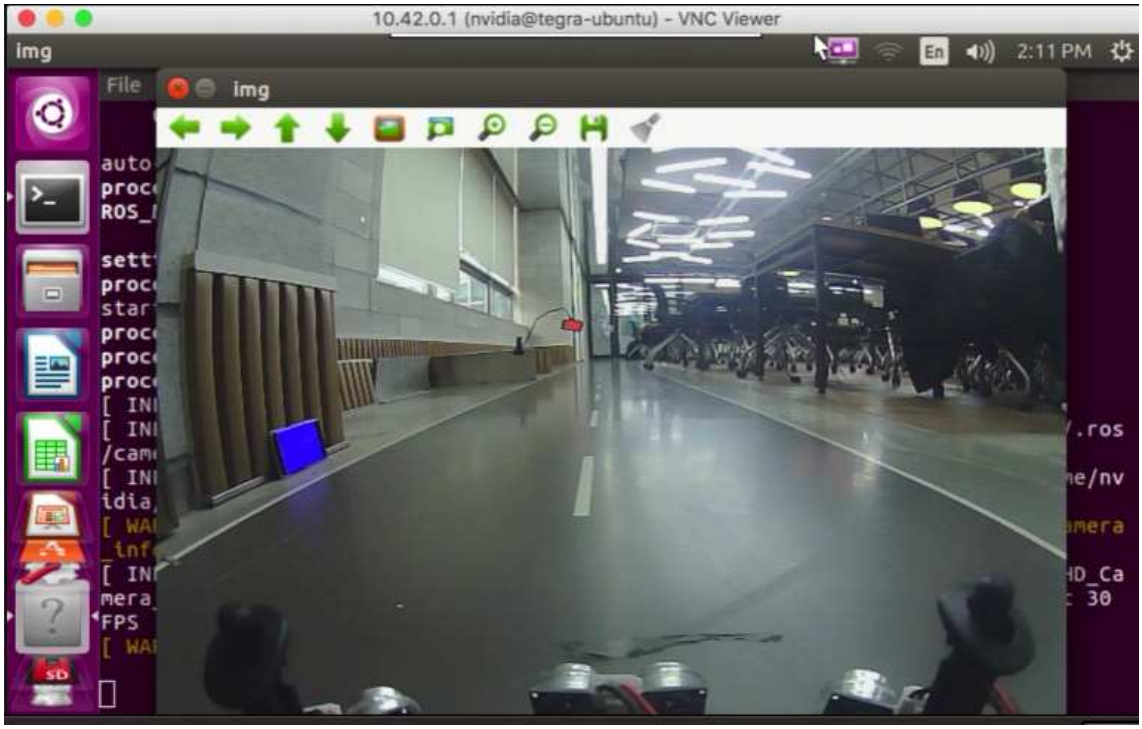
물론, 주차장에서 진입하면서 OpenCV를 통해 해당 주차장의 요금이 얼마인지, 차량의 종류에 따라 가격이 다르다는 변수를 찾아내 차량에서 주차 요금을 연산할 수 있었다면 그것으로 주차 요금이 아닌 다른 분야에도 응용할 수 있다고 생각이 들지만 구현을 하지 못한 것이 아쉬움으로 남는다.

```
51  angle = 90
52  speed = 90
53  park = False
54  before = False
55  starttime = 0
56  nowtime = 0
57
58  flag = False
```

먼저, 무한루프를 돌리기 전에 starttime과 nowtime 변수를 time.time()으로 기록해야 하기 때문에 정수 0으로 초기화한다.

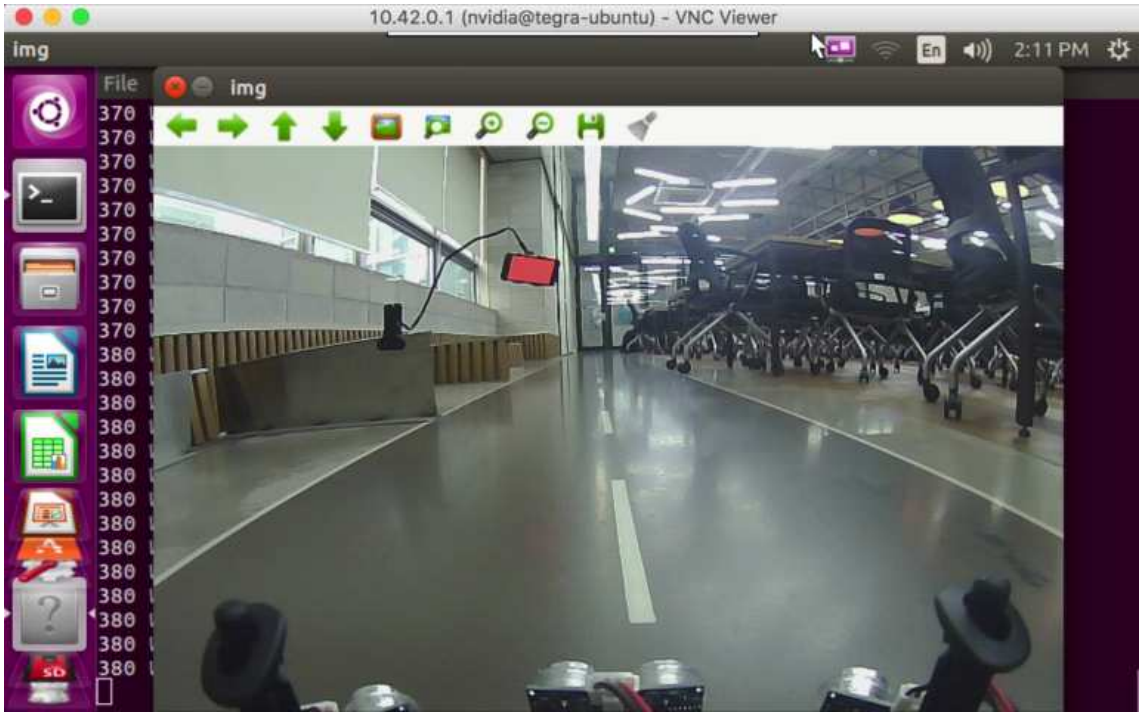
```
while True:
    frame = capture
    if park:
        frame = park_lines(frame)
    cv2.imshow('img', frame)
    park = detectRed(frame)
    blue = detectBlue(frame)
    if not flag and nowtime != 0:
        print("{} Won".format(int(nowtime - starttime) * 10))
        flag = True
    if blue != before:
        if starttime == 0:
            starttime = time.time()
        else:
            nowtime = time.time()
        before = blue
    key = cv2.waitKey(1)
    if key & 0xff == 27:
        break
    if 81 <= key and key <= 84:
        drive(angle, speed)
        if key == 82:
            speed = 120
        if key == 81:
            if angle - 10 >= 40:
                angle -= 10
        if key == 83:
            if angle + 10 <= 150:
                angle += 10
        if key == 84:
            speed = 65
    if key == 127:
        p = False
```

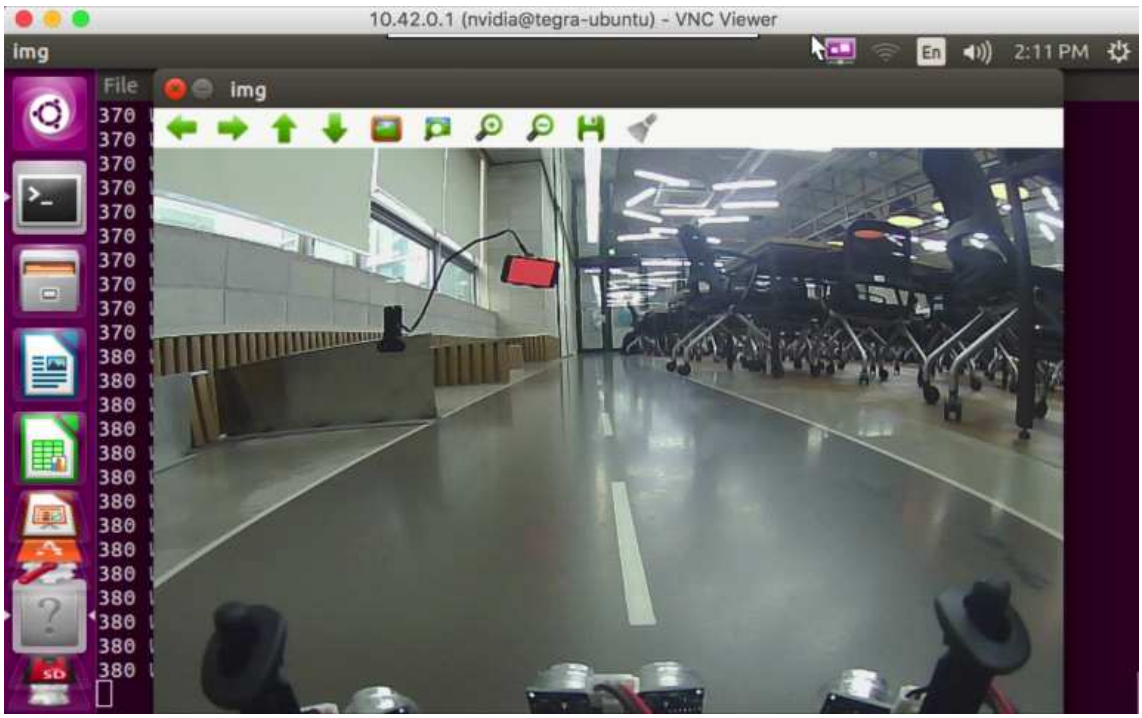
먼저, 우리가 준비한 트랙에서 파란 화면을 Xycar의 카메라 센서로 찾아 낼 수 있는 위치에 두고 파란 화면을 검정할 때에 starttime 변수가 루프 이전에 0으로 초기화가 되어 있기 때문에 시간을 time 모듈을 통해 starttime 변수에 기록한다. 현실 세계에서 주차장 앞에 있는 센서가 차량 번호판을 감지하여 주차장에 있는 모니터에 들어온 시간이 기록되는 것과 같은 원리이다.



차량이 파란 화면을 찾아내 주차장에 들어온 시간이 기록되고 트랙을 지나 빨간 화면을 접하게 된다. 빨간 화면을 접하게 되면 위에서 설명한 주차 모드가 변경되어 차량은 주차를 하게 되고 이후 나갈 일이 되면 차량은 다시 맨 처음에 접했던 파란 화면 쪽으로 나가게 될 것이다.

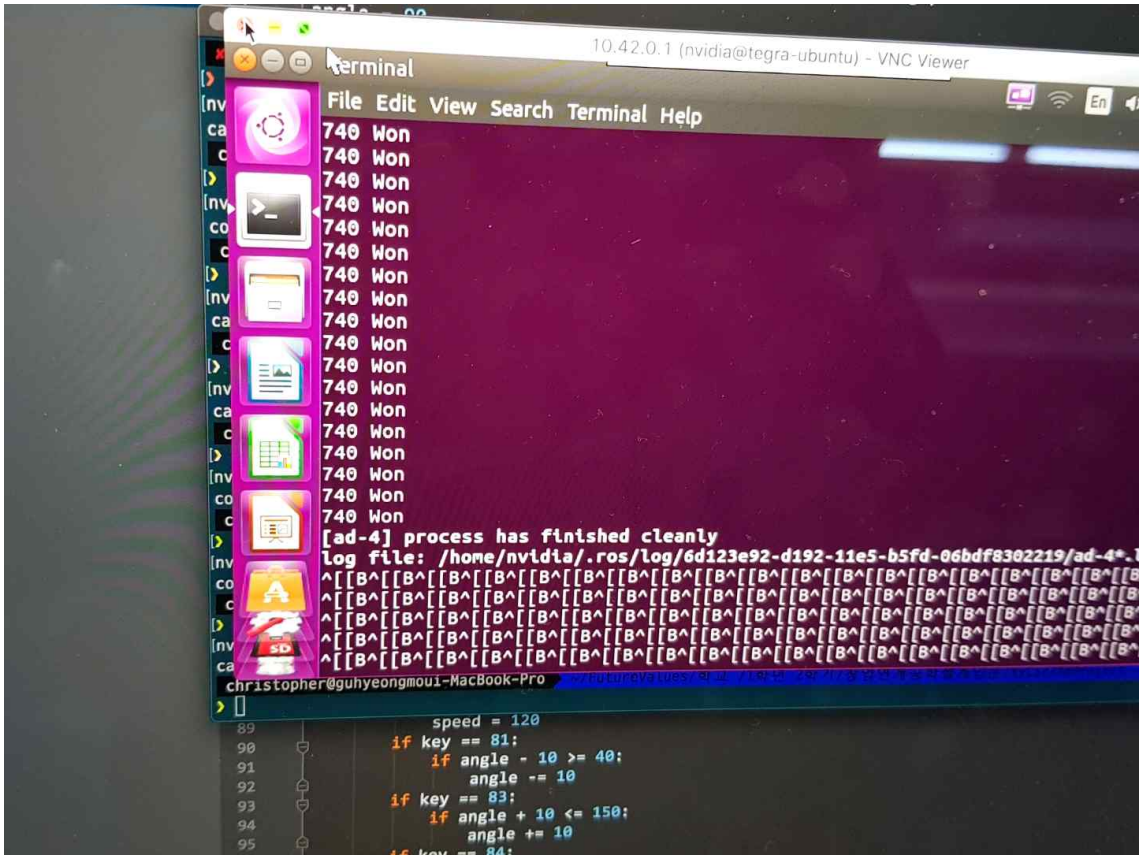
이는 주차장에서 출차하여 나간 시간이 기록되는 것과 같은 의미이다.





차량이 트랙에서는 맨 처음 위치로 돌아가고 현실 세계에서는 주차장에서 빠져나갈 때 맨 처음에 접했던 파란 화면을 다시 마주치게 된다. 위의 코드에서 나와 있는 로직대로 starttime이 맨 처음 들어올 때처럼 0으로 기록이 되어 있지 않기 때문에

이번에 파란 화면을 다시 마주치게 된 시간은 nowtime 변수에 기록이 된다.



이렇게 starttime과 nowtime 변수에 파란 화면을 마주친 시간이 기록이 되는데, nowtime과 starttime 변수에 저장된 값을 뺀 초에 대한 값에 임의의 특정 요금을 곱한다. 우리 조는 현실 세계에 주차장 앞에 있는 차량의 종류, 주차 시간 당 요금이 나와 있는 것을 영상 처리를 통해 주차 요금을 계산하는 것을 구현하지 못했기 때문에 위의 코드 로직대로 따르면 임의의 요금인 10을 곱하여 주차 요금을 출력하도록 했다.

Ⅳ. 프로젝트의 느낀 점 및 개선해야 할 점

1. 느낀 점

이번 창업연계공학설계입문 한 학기 수업을 들으면서 물론 많은 어려움도 겪으며 시행착오를 겪었지만 주행 과제와 AD 프로젝트를 진행하면서 팀 프로젝트 능력, 개인의 전공 능력 성장, 많이 접하질 못할 수도 있는 분야에 대해서 이번 수업을 통해 공부할 수 있어서 쉽게 얻지 못할 경험을 얻을 수 있었다고 생각한다.

우리 조는 이번 기말 AD 프로젝트를 수행했던 것뿐만이 아니라 한 학기 수업을 통해 실습을 진행하면서 다른 조에 비해 인원이 적었기 때문에 실습이나 프로젝트에 있어서 개인으로서 각자 투자해야 하는 노력이 많을 수밖에 없었기 때문에 이번 학기, 창업연계공학설계입문 수업을 잘 마칠 수 있을까 걱정이 되었다. 하지만 학기가 점점 진행하면서 다른 조에 비해 인원이 적었다는 위기를 기회로 삼아 소수의 인원으로 더 뭉칠 수 있다는 긍정적인 가능성을 점점 더 발견할 수 있었다고 생각한다. 또한, 토론을 준비할 때, 프로젝트의 기획이나 주행 평가에 있어서 계획한 대로 일이 잘 풀리지 않을 때 서로 머리를 맞대고 고민을 하는 등 이러한 긍정적인 팀에서의 움직임을 통해 소프트웨어 개발자가 갖추어야 할 덕목인 팀 플레이 능력 또한, 향상시킬 수 있었던 것 같다.

그 다음으로 개인의 전공 능력의 성장을 이번 학기를 통해 인상 깊게 느꼈던 부분으로 꼽을 수 있다. 프로젝트를 하면서 계획한 대로 구현이 되지 않을 때에 팀의 입장에서 서로 맞대고 고민을 하는 것도 있었지만 개인의 입장에서 코드에 적용할 수 있는 오픈소스, 다양한 소프트웨어적인 개념을 각자 개별로 공부하여 성공적으로 주행 평가 및 프로젝트를 마무리할 수 있었던 것 같고 한 학기를 뒤돌아보면 전공 능력 또한 성장할 수 있었다고 생각한다.

마지막으로는 이번 수업을 통해 어떻게 보면 많이 접하지 못할 수도 있는 분야를 이번 수업을 통해 접할 수 있었다는 점이다. 학부에서 임베디드 시스템에 대한 과목이 고학년에 존재하지만, 개인의 입장에서 관심이 없다면 해당 분야에 대해서 좀 더 깊이 접근하지 못할 가능성이 크다고 생각한다.

학기 초반에 교수님께서 수업을 통해 설명하셨던 물리적인 현실 세계와 컴퓨터를 잇는 physical computing은 물리 세계에서는 우리가 생각한 것과는 달리 상당히 많은 변수로 인해 문제를 쉽게 일반화하는데 상당히 많은 문제가 발생한다고 하셨다. 그러한 현실 세계의 복잡한 제약을 극복하고 문제를 우리가 구상한대로 풀어내기 위해 어떤 부분을 덜어내고 어떤 다른 부분은 추가하여 마침내 구상한대로 문제를 풀어낼 때 어찌 보면 많이 접하지 못할 분야를 이번 수업을 통해 접할 수 있었던 것 같고 나중에 있어 해당과 같은 문제를 마주치게 될 때 극복할 수 있는 힘을 얻을 수 있었던 한 학기의 수업이었던 것 같다.

2. 개선해야 할 점

다양한 Xycar CPU의 느린 처리 속도와 반응속도 그리고 인터넷 연결이 불가능하여 파이썬의 가장 큰 장점인 pip를 이용한 라이브러리 설치 불가, 후방 카메라의 부제로 인해 기존 계획 이였던 완전 자율주행 자동차에서 수동 주차 보조 시스템으로 변경 하였다. 시중에 판매되는 자율 주행 차량은 Lidar 같은 고급 센서와 앞뒤로 장착되어 있는 카메라 덕분에 완전 자율 주차가 가능하지만 이를 xycar 에 적용시키 위해서는 전방카메라 만을 의존하여 주차를 할 수 있는 알고리즘을 연구하게 된다면 가능하다고 생각된다.

기존의 주차 요금은 파란색을 두번 인식하여 인식된 시간을 빼서 요금을 측정했다. 하지만 이 방법은 다른 파란색을 인식하여 요금을 잘못 측정할 가능성이 매우 크다. 이러한 부분을 수정할 필요성이 있다. 그리고 기존에는 주차요금을 측정만 할뿐 금액을 내고 나가지 않았다. 이러한 부분을 금액을 내지 않을 때는 차단바로 막고 임의의 화폐를 내는 버튼을 눌러야 차단바가 올라가 나갈 수 있게 할 수 있었을 것 같다.

주차를 구현해 보면서 자율 주차는 불가능 하다고 느꼈다. 하지만 자율 주행은 수업시간에서 해보았다. 방향키를 이용하여 주행하고 주차하는 것 대신에 정해진 트랙에서 주행을 하다가 빈 주차표시를 인식하면 주차모드로 바뀌며 주행 또한 방향키로 할 수 있게 했으면 더 좋았을 것 같다. 그리고 주차를 완료 했을때 기존에는 p를

눌러 정차를 알렸다. 하지만 자동차의 모드만 주차모드로 바뀔 뿐 주차장의 표시는 그대로였다. 이것을 1학기때 배웠던 앱 인벤터를 이용하여 주차가 완료 되었을 때 색을 바꿔 줘줬다면 더 좋았을 것 같다.