Busan science high school

2023 Ocean ICT Festival
2023 BOIF

D 01

QR 코드 영역
QR 삽입 후
테두리 삭제

Youtube 영상 QR

별자리를 파악하고 항해방향을 판단하는 머신러닝 알고리즘 구축 및 실체화

Polaris : 1207 강민재, 1210 김성민

탐구동기 : 옛날 사람들은 넓은 바다를 항해할 때 북극성을 찾아 방향을 파악했다고 한다. 따라서 우리는 지구과학에서 배운 방위에 대한 지식을 활용하여 별자리를 보고 방위를 파악하는 간단한 AI를 구현하고 오션 ICT를 통하여 공유함에 따라 다양한 사람들이 항해의 역사를 알아보고, 제작된 AI를 통해 4차 산업혁명을 체감할 수 있는 학습과 체험의 장을 형성함을 목적으로 하고 있다. 이러한 작품을 만들게 된 동기는 옛날 사람들이 바다에서 길을 찾는 방법을 응용하면 우리가 우주에서 더 효율적으로 길을 찾을 수 있지 않을까 하는 호기심에서 시작되었다.

작동 과정

밤하늘 사진을 활용하여 머신러닝 모델을 학습시킴 → 사진을 입력받아 방위를 판정 → 판정된 방위를 아두이노로 전송

8x8 LED matrix를 통해 방위를 출력

코드

별자리 이미지 생성 코드 및 머신 러닝 과정

별 모양 그리기 함수

```python
# 별 그리는 함수 정의
def draw_star(x, y, rotation, color, flip):
    star.penup()
    star.goto(x, y)
    star.pendown()
    star.setheading(rotation)
    if flip:
        star.left(180)  # flip이 True이면 터틀을 뒤집음
    star.begin_fill()
    star.fillcolor(color)
    for _ in range(5):
        star.forward(40)
        star.right(144)
    star.end_fill()
```

별의 x, y 좌표값을 입력받은 후 터틀과 펜 관련 코드를 사용하여 별모양을 그린다.

별 위치 설정

```python
# 별 간격 설정 (100으로 수정)
star_spacing = 100

# Ursa Minor의 별 위치 정의
stars1 = [(-400, 200), (-320, 180), (-240, 160), (-160, 180), (-80, 240), (-20, 200), (-120, 120)]
stars2 = [(400, 200), (320, 180), (240, 160), (160, 180), (80, 240), (20, 200), (120, 120)]
stars3 = [(x + 360, y - 140) for x, y in stars1]

# stars1과 stars2를 화면 끝으로 평행이동
stars1 = [(x - 200, y) for x, y in stars1]
stars2 = [(x + 200, y) for x, y in stars2]

# stars3와 stars4 리스트 생성 (십자가 모양으로 별 4개가 위치)
stars4 = [
    (0, star_spacing),              # 위쪽 별
    (0, 2 * (-star_spacing)),       # 아래쪽 별
    (-star_spacing, 0),             # 왼쪽 별
    (star_spacing, 0),              # 오른쪽 별
]
```

별의 간격을 100으로 맞춰주는 함수를 정의하고 star1과 star2의 좌표를 설정한다음 이 좌표를 이용하여 star3와 star4의 좌표까지 설정한다

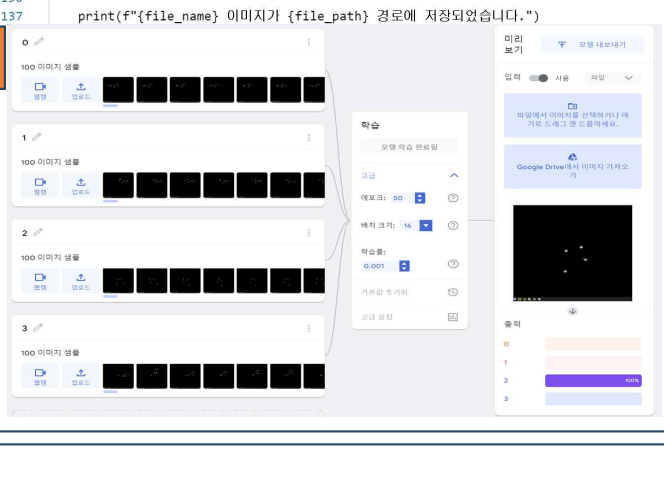
생성된 별 이미지 저장

```python
# 바탕화면 경로를 얻습니다.
desktop_path = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')

# star 폴더를 생성합니다.
star_folder_path = os.path.join(desktop_path, 'star')
if not os.path.exists(star_folder_path):
    os.mkdir(star_folder_path)

# star_test 폴더를 생성합니다.
star_test_folder_path = os.path.join(star_folder_path, 'star_test')
if not os.path.exists(star_test_folder_path):
    os.mkdir(star_test_folder_path)
# stars1, stars2를 그리고 저장하기
for i in range(1, 6):
    star.clear()
    for j, (x, y) in enumerate(stars1):
        rotation = (i * 3.6)  # 3.6 degrees rotation for each iteration
        flip = random.choice([True, False])
        if j == 0:
            color = "blue"  # 첫 번째 별(가운데 별)은 파란색으로 지정
        else:
            color = "white"
        draw_star(x, y, rotation, color, flip)

    file_name = f'stars1_{i}.png'
    file_path = os.path.join(star_test_folder_path, file_name)
    screenshot = ImageGrab.grab()
    screenshot.save(file_path)

    print(f"{file_name} 이미지가 {file_path} 경로에 저장되었습니다.")

for i in range(1, 6):
    star.clear()
    for j, (x, y) in enumerate(stars2):
        rotation = (i * 3.6)  # 3.6 degrees rotation for each iteration
        flip = random.choice([True, False])
        if j == 0:
            color = "blue"  # 첫 번째 별(가운데 별)은 파란색으로 지정
        else:
            color = "white"
        draw_star(x, y, rotation, color, flip)

    file_name = f'stars2_{i}.png'
    file_path = os.path.join(star_test_folder_path, file_name)
    screenshot = ImageGrab.grab()
    screenshot.save(file_path)

    print(f"{file_name} 이미지가 {file_path} 경로에 저장되었습니다.")
```

```python
# stars3, stars4를 그리고 저장하기
for i in range(1, 6):
    star.clear()
    for j, (x, y) in enumerate(stars3):
        rotation = (i * 3.6)  # 3.6 degrees rotation for each iteration
        new_x = x * math.cos(rotation * math.pi / 180) - y * math.sin(rotation * math.pi / 180)
        new_y = x * math.sin(rotation * math.pi / 180) + y * math.cos(rotation * math.pi / 180)
        flip = random.choice([True, False])
        if j == 0:
            color = "blue"  # 첫 번째 별(가운데 별)은 파란색으로 지정
        else:
            color = "white"
        draw_star(new_x, new_y, rotation, color, flip)

    file_name = f'stars3_{i}.png'
    file_path = os.path.join(star_test_folder_path, file_name)
    screenshot = ImageGrab.grab()
    screenshot.save(file_path)

    print(f"{file_name} 이미지가 {file_path} 경로에 저장되었습니다.")
for i in range(1, 6):
    star.clear()
    for j, (x, y) in enumerate(stars4):
        rotation = (i * 3.6)  # 3.6 degrees rotation for each iteration
        new_x = x * math.cos(rotation * math.pi / 180) - y * math.sin(rotation * math.pi / 180)
        new_y = x * math.sin(rotation * math.pi / 180) + y * math.cos(rotation * math.pi / 180)
        flip = random.choice([True, False])
        color = "white"  # 파란색 별이 없으므로 모두 흰색으로 지정
        draw_star(new_x, new_y, rotation, color, flip)

    file_name = f'stars4_{i}.png'
    file_path = os.path.join(star_test_folder_path, file_name)
    screenshot = ImageGrab.grab()
    screenshot.save(file_path)

    print(f"{file_name} 이미지가 {file_path} 경로에 저장되었습니다.")
```

별들을 그린 다음에 바탕화면에 star 폴더를 생성하여 저장한다

머신 러닝 과정



생성된 별자리 이미지를 모델에게 학습시킴
-> 모델은 별자리 이미지만 봐도 현재의 방향을 추론 가능

마무리 코드

```python
# 화살표 모양 숨기기
star.hideturtle()

# 윈도우 유지
wn.mainloop()
```

아두이노 코드

1. 화살표 모양 이미지 생성

```
#include
<FrequencyTimer2.h>

char result = '0';

#define E  { \
   {0, 1, 1, 1, 1, 1, 1, 0}, \
   {1, 0, 0, 0, 0, 0, 1, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 1, 1, 1, 0, 0, 1}, \
   {1, 1, 1, 1, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 1, 1}, \
   {0, 1, 1, 1, 1, 1, 1, 0} \
}

#define W { \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 1, 1, 1, 0, 1}, \
   {1, 0, 0, 0, 1, 1, 1, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 1, 1}, \
   {0, 1, 1, 1, 1, 1, 1, 0} \
}

#define N { \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
   {1, 0, 0, 0, 1, 0, 1, 1}, \
   {1, 0, 0, 1, 1, 0, 0, 1}, \
   {1, 0, 0, 1, 1, 0, 0, 1}, \
   {1, 0, 0, 1, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 1, 1}, \
   {0, 1, 1, 1, 1, 1, 1, 0} \
}

#define S { \
    {0, 1, 1, 1, 1, 1, 1, 0}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 0, 0, 0, 1}, \
   {1, 0, 0, 0, 1, 0, 0, 1}, \
   {1, 0, 0, 1, 1, 0, 0, 1}, \
   {1, 0, 0, 1, 1, 0, 0, 1}, \
   {1, 0, 0, 1, 0, 0, 1, 1}, \
   {0, 1, 1, 1, 1, 1, 1, 0} \
}
```

2. led 관련 코드

```
int tact = A5;     // 택트 스위치 a%에 연결
byte col = 0;      // COL을 0으로 초기화
byte leds[8][8];   // 현재 출력해야 할 LED 모양 업로드하는 배열

// 맨 처음 PINS[0]은 사용하지 않기때문에 -1로 설정. 1~16번까지의 핀을 PIN에 연결
int pins[17]= {-1, 5, 4, 3, 2, 14, 15, 16, 17, 13, 12, 11, 10, 9, 8, 7, 6};

// 행 0~7번까지 핀 연결해 주기
int cols[8] = {pins[13], pins[3], pins[4], pins[10], pins[6], pins[11], pins[15], pins[16]};

// 열 0~ 7번까지 핀 연결해 주기
int rows[8] = {pins[9], pins[14], pins[8], pins[12], pins[1], pins[7], pins[2], pins[5]};

const int numPatterns = 7;     // 총 사용할 패턴 수
byte patterns[numPatterns][8][8] = {E, W, S, N};  // 위에서 정의한 led 모양을 patterns에 입력해 주기

int pattern = 0;

void setup() {
 // 1~16번 까지의 핀을 출력으로 설정
 for (int i = 1; i <= 16; i++) {
  pinMode(pins[i], OUTPUT);
 }

 // 행 0~7번까지를 high로
 for (int i = 0; i < 8; i++) {
  digitalWrite(cols[i], HIGH );
 }
 // 열 0~7번 까지를 low로
 for (int i = 0; i < 8; i++) {
  digitalWrite(rows[i], HIGH);
 }

 clearLeds();  // led 초기화

 FrequencyTimer2::setOnOverflow(display);  // leds를 보여주기 위해서 setOnOverFlow를 사용

 pinMode(tact, INPUT); // SW 를 설정, 아두이노 풀업저항 사용

 setPattern(pattern);
}
```

3. 시리얼 코드

```
void loop() {
   int readTact =
digitalRead(tact);

   if(readTact ==
HIGH){    // 택트 스위치가 high일때
     if
(Serial.available() > 0) {
       result =
Serial.read();
     }
     switch (result) {
       case '0':
         pattern = 0;
         break;
       case '1':
         pattern = 1;
         break;
       case '2':
         pattern = 2;
         break;
       case '3':
         pattern = 3;
         break;
       default:
         pattern = 0;
     }
   }
   setPattern(pattern);
// 패턴 출력
}
```

4. 마무리 코드

```
void clearLeds() {  // led 다 초기화
 for (int i = 0; i < 8; i++)
   for (int j = 0; j < 8; j++)
     leds[i][j] = 0;
}

void setPattern(int pattern)
{      // LED 배열에 PATTREN 입력하기
 for (int i = 0; i < 8; i++)
   for (int j = 0; j < 8; j++)
     leds[i][j] =
patterns[pattern][i][j];
}

// Interrupt routine
void display() {
 digitalWrite(cols[col], HIGH);  // 이전 행들 다 꺼지게끔 해주기.
 col++;
 if (col == 8) {
   col = 0;
 }
 for (int row = 0; row <= 7; row++) {
   if (leds[col][7 - row] == 1)
{      // 위의 배열에서 1일때 해당되는 위치 불빛 켜주기
     digitalWrite(rows[row], HIGH);
   }
   else {                  // 위 배열에서 0일때 해당되는 위치 불빛 꺼주기
     digitalWrite(rows[row], LOW);
   }
 }
 digitalWrite(cols[col], LOW);
// 다음 패턴을 위해 led 다 꺼주기
}
```

코드 실행 결과

별자리를 인식하여
그에 맞는 방위 값을 출력

시리얼 코드를 통해 입력받은 방위대로 나침반모양을 출력

기대 효과

오션 ICT에 함께 참여하는 본교의 학생들에게 머신러닝과 같은 AI기술에 대한 이해도를 별자리라는 흥미로운 소재를 통해 증진시켜 줄 수 있으며, 별자리를 통한 방위를 판단하는 법을 알게되며 자연을 통해 방위를 파악하는 생존 기술을 간단히 익힐 수 있을 것으로 기대된다.

느낀점

강민재: 간단하지만 직접 AI를 제작하여 머신러닝을 해보니 코딩실력을 더욱 성장하게 되는 것 같고, R&E에서 AMD에 대하여 배울 기회가 생기게 되었는데 우리의 프로젝트에도 AMD로 업그레이드 시켜 더 좋은 성능과 짜임새있는 코드를 이끌어내보고 싶다.
김성민: 이번 활동을 통해 아두이노를 처음 알게 되었는데, led를 작동하는 것부터 복잡한 탐구까지 직접 코딩해보면서 아두이노에 대해 자세히 알아가는 과정이 되게 재미있었다. 또한 이 원리를 이용하여 우주에서도 별과 행성을 이용해 방향을 찾는 코드를 만들어 보고 싶다.