OpenCV day2 hw1 결과보고서

2025407012/로봇학부/송연우

목차

1. OpenCV

해당 과제의 구현 코드를 Qtcreator를 활용하여 작성했습니다.

2. mainwindow.h

사용할 함수로 비디오를 캡쳐한 이미지를 지속적으로 스트리밍할 함수인 streaming()과 이미지 이진화를 실시할 때 사용하는 make_binary(), 바운더리 박스를 생성하는 기능을 담당하는 make_box()를 선언했습니다. 슬롯 함수로는 라디오버튼 4개의 클릭 시그널에 대한 슬로 함수와, 수평 슬라이드바 6개의 위치 변화 시그널에 대한 슬롯 함수가 있습니다.

```
62        Mat bus_img, white, blue, neon, orange, white_b, blue_b, neon_b, orange_b, img;
```

또한 이미지를 저장할 Mat 객체인 bus_img, white, blue, neon, orange, white_b, blue_b, neon_b, orange_b, img, src, label, stats, centroids, src_color를 선언했습니다.

White, blue, neon, orange는 각각 물체의 이진화 이미지를 저장하는 객체이고, _b가 붙은 경우는 바운더리 박스만큼 추출한 이미지를 저장하는 객체입니다.

```
9         int hsv[24]={0};          int x[4];
```

또한 슬롯 바의 값을 저장하는 hsv배열과 바운더리 박스 내 이미지를 추출할 때 시작점의 x, y좌표와 가로, 세로 길이를 저장할 배열 x, y, width, height를 선언했습니다.

3. mainwindow.cpp

각 함수들에 대해서 자세히 설명드린 후, 전체 알고리즘에 대해 다루겠습니다.

(1) MainWinDow(생성자)

생성자 함수에서는 먼저 .h파일에서 생성한 Videocapture cap 객체를 open하고, open되지 않았으면 오류 메시지를 띄운 후 종료합니다.

```
15          if(!cap.open(1)) {
```

카메라가 제대로 열렸다면 Mat객체인 bus_img에 캡쳐한 프레임을 저장합니다.

```
21          cap >> bus_img;
```

다음으로, 각종 변수들을 초기화합니다. 각각의 라디오버튼을 누를 때마다 값이 변하는 변수 whatclk는 초기화 시 0으로 설정하고, 타이머에 값을 할당합니다.

```
31          timer = new QTimer(this);
```

그리고 타이머의 초기화 시간은 100ms정도로 설정합니다.

```
35          timer->setInterval(100);
```

타이머와 streaming()함수를 묶어 지속적으로 화면에 이미지가 갱신되어 동영상이 재생되도록 합니다.

```
38          connect(timer, &QTimer::timeout, this, &MainWindow::streaming);
```

코드 상에서는 connect로 묶었지만 타이밍 차이로 인해 존재하지 않는 이미지 파일을 화면에 출력하지 않도록 사용하지 않는 함수도 있습니다.

```
connect(timer_range, &QTimer::timeout, this, &MainWindow::make_binary);
```

connect코드 작성 후 타이머를 작동시킵니다.

```
41          timer->start();
```

(2) streaming

원래는 make_binary함수와 따로 기능했지만 타이밍 문제를 없애기 위해 해당 함수의 코드를 옮겨와 inRange함수까지 실행합니다.

따라서 없는 이미지를 이진화시키지 않도록 bus_img가 비었다면

```
49          if(bus_img.empty())
```

함수를 종료하도록 합니다.

```
51              std::cout << "video end" << std::endl;      return;
```

이미지가 존재하면 해당 이미지의 색차원을 rgb에서 hsv로 바꾸어 img에 저장합니다.

```
cvtColor(bus_img, img, COLOR_BGR2HSV);
```

다음으로 inRange함수에 슬롯 바 값이 저장된 hsv배열을 인수로 넣은 Scalar함수를 넣고 이미지 변환을 시작합니다.

```
{inRange(img, Scalar(hsv[13], hsv[17], hsv[21]), Scalar(hsv[1], hsv[5], hsv[9]), blue);qDebug() << whatclk;}
```

나머지 neon(실제로는 노란색 콘), orange, white도 동일하게 변환해 저장합니다.

이후에 있는 코드는 바운더리 박스를 만들기 위한 코드인데, 해당 코드로 박스를 생성하는 것에 실패하여 따로 출력하거나 하지는 않는 코드입니다.

바운더리 박스 생성 코드 밑에 있는 QImage를 생성하는 코드부터가 label에 이미지를 출력하는 부분입니다.

```
QImage qimg(neon.data, neon.cols, neon.rows, neon.step, QImage::Format_Grayscale8);
```

먼저 QImage객체를 만듭니다.

```
ui->label_2->setPixmap(QPixmap::fromImage(qimg).scaled(ui->label_2->size(), Qt::KeepAspectRatio, Qt::SmoothTransformation));
```

생성한 객체를 label_2 객체의 setPixmap함수를 사용해 이름이 label_2인 Qlabel에 출력합니다.

나머지 색상의 물체도 동일하게 출력합니다.

```
QImage qimg_2(orange.data, orange.cols, orange.rows, orange.step, QImage::Format_Grayscale8);
```

```
ui->label_3->setPixmap(QPixmap::fromImage(qimg_2).scaled(ui->label_3->size(), Qt::KeepAspectRatio, Qt::SmoothTransformation));
```

밑의 코드는 디버깅을 위한 부분입니다.

```
qDebug() << "스트리밍 실행 중";
```

(3) 슬롯 함수

라디오버튼: 각각의 라디오 버튼은 white, blue, neon, orange 물체를 선택하는 용도

이며, 클릭되었을 때 whatclk변수 값을 1, 2, 3, 4로 변화시킵니다.

```
whatclk =1;
whatclk =2;
whatclk =3;
whatclk =4;
```

수평 슬롯 바: 각각의 슬롯 바는 H, S, V의 high, low값을 담당하며 whatclk값에 따라 hsv배열의 다른 인덱스에 접근해 position값을 저장합니다.

```
void MainWindow::on_horizontalSlider_sliderMoved(int position) //hue high
```

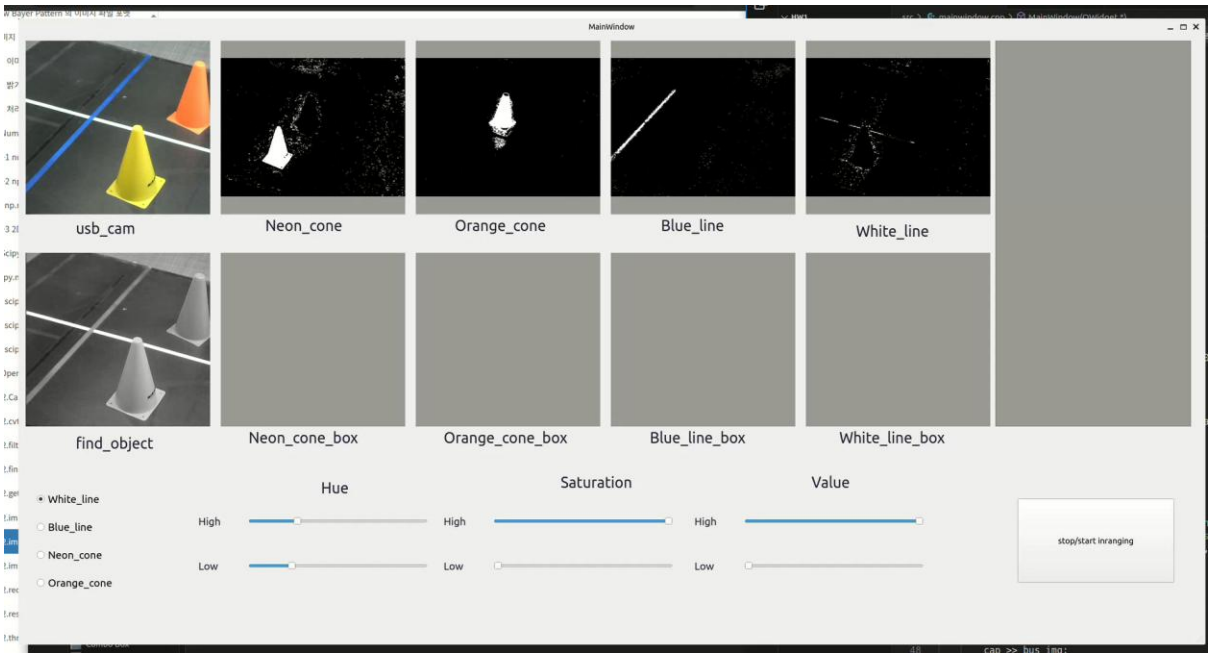
만약에 whatclk값이 초깃값인 0으로 버튼이 아무것도 선택되지 않으면 아무것도 하지 않고 함수를 종료합니다.

```
if(whatclk==0)return;
hsv[whatclk-1]=position;
```

나머지 함수는 사용하지 않습니다.

프로그램의 알고리즘은 main문에서 시작되지만 단순히 connect로 이어진 타이머로 주기적으로 작동합니다. 100ms마다 streaming()함수가 실행되어 영상을 재생하며 슬롯 함수로 이진화 설정을 조정합니다.

5. 실행 결과: 다음은 프로그램을 실행시켰을 때의 ui 화면입니다.



하단의 슬라이드 바를 조절하면 위의 화면이 변하고, 다른 라디오 버튼을 선택하면 물체당 설정을 유지할 수 있습니다. 우측 하단의 stop/strat inranging버튼은 작동하지는 않습니다. 화면에서는 usb 화면과 이진화 이미지 화면 사이즈가 맞지 않는데 이진화 이미지의 경우 label 사이즈에 맞추어 크기를 설정했지만 usb 화면은 그대로 잘려 나오기 때문입니다. 바운더리 박스를 생성하는 기능은 구현하지 못했습니다.