ROS2_4일차_최범수

19기 지능형 로봇팀 예비단원 최범수

함수 설명

- **Parameter**

파라미터란 매개변수를 의미한다. 파라미터의 기능은 노드 내부에 있는 시스템을 외부 client로 통신해 변경할 수 있다는 것이다. 쉽게 표현하면 노드 내에 하나의 파라미터 네트워크 서버가 있고 안에 장치의 버전, 기능 등을 불러와서 바꿔주는 것이라고 할 수 있다. 이 파라미터는 turtlesim_node에서 펜의 색이나 배경색을 바꿀 때도 사용하였는데 이 역시도 관련 서비스를 외부에서 호출해 변경한 것이다.

ROS2에서 파라미터를 선언할 때는 보통 `this →declare_parameter<type>(”“, value);` 과 같은 형식으로 사용한다. 그 후 안에 value를 조정해 원하는 값으로 바꿀 수 있다.
값을 가져오는 방식은
`this →get_parameter(”parameter_name”).get_value<type>()` 이다. 두 가지 방법의 차이는 변수에 값을 저장하는 방법과 파라미터를 통해 값을 가져오는 방법이라는 차이가 있다. 쉽게 생각하면 출력과 입력의 개념이다.

```
이번 과제인 H_Upper를 통해 예를 들자면

node->declare_parameter("H_lower", 0);
int h_lower;
this->get_parameter("H_lower", h_lower);
```

이렇게 사용하였다.

`RCLCPP_INFO(this->get_logger(), "H_lower value: %d", h_lower);` 이 구문을 통해 터미널 창에서 어떤 값을 받고 있는지 확인할 수 있다.

ROS2에서는 런치파일을 통해 파라미터를 불러올 수도 있다.

```
from launch import LaunchDescription
from launch_ros.actions import Node
```

```
def generate_launch_description():
    return LaunchDescription([
        Node(
            package='my_package',
            executable='my_node',
            name='my_node',
            parameters=[
                {"H_lower": 10},  # H_lower 파라미터 설정
                {"H_upper": 100}  # H_upper 파라미터 추가 설정
            ]
        ),
    ])
```

my_node라는 노드에서 H_lower와 H_upper 파라미터 전달한다. 그 후 아까와 마찬가지로 declare_parameter와 get_parameter를 통해서 값을 가져올 수 있다.

- Grayscale

각 픽셀마다 0~255 사이의 값으로 흰색과 검정색 사이의 색을 표현가능하다. canny등의 기능에서 외곽선을 따야하거나 호프라인에서 직선을 표현해야 할 경우 사용한다.

- Binary image

흰색 255와 검정 0만으로 구성된 이미지이다.

- RGB 이미지

각 픽셀마다 0~255 사이의 값이 RED, GREEN, BLUE 3개가 들어있어 색상을 표현가능하다. (0, 0, 0)

- H,S,V

Hue, Saturation,Value로 구성되어있는 색상을 나타낼 수 있는 포맷이다. Hue는 0 ~ 180 사이로 구성되어 있고 나머지는 0~255로 최대를 조정했다. 3가지 값을 조절해서 특정 색상을 제외하고 전부 검정으로 표시해 원하는 색상만을 추출할 수 있다. 보통 주황색을 찾는다면 Hue의 minimum을 0으로 maximum을 10정도로 놓는다면 인식 가능하다.

```
        int h_lower = getParameter("H_lower");
        int h_upper = getParameter("H_upper");
        int s_lower = getParameter("S_lower");
        int s_upper = getParameter("S_upper");
```

```
        int v_lower = getParameter("V_lower");
        int v_upper = getParameter("V_upper");
```

이런식으로 parameter 값을 읽도록 하였고, 함수 내용은 다음과 같다.

```
int ImageNode::getParameter(const std::string &name)
{
    int value;
    if (node->get_parameter(name, value))
    {
        return value;
    }
    else
    {
        RCLCPP_WARN(node->get_logger(), "Parameter %s not fou
        return 0; // 기본값 반환
    }
}
```

value라는 변수값에 파라미터에 접근해 값을 인식한다.

```
cv::Mat hsv_image;
cv::cvtColor(cv_image, hsv_image, cv::COLOR_BGR2HSV);
```

imageCallback 함수 내에서 cv 관련 함수를 사용한다. COLOR_BGR@HSV에 대해서 설명하자면 openCV에서 RGB를 BGR8로 변환해 출력하므로 그에 맞춰서 출력하도록했다.

```
QImage qimage_binary(binary_image.data, binary_image.cols, bi
QPixmap pixmap_binary = QPixmap::fromImage(qimage_binary.rgbS
```

이미지 변환 과정이다.

- Resize

```
resize(image, image_new_setted, Size(640, 360));
```

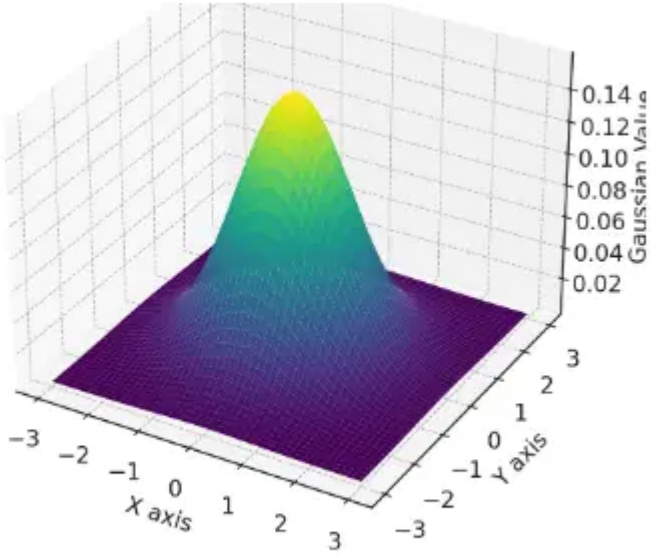
이 코드를 통해 size 조절이 가능하고 실제 코드는 다음과 같다.

```
node->declare_parameter("image_width", 640);
node->declare_parameter("image_height", 480);

QPixmap resized_original = pixmap_original.scaled(targetWidth
QPixmap resized_binary = pixmap_binary.scaled(targetWidth, ta
QPixmap resized_edges = pixmap_edges.scaled(targetWidth, targ
Q_EMIT newImage(resized_original, resized_binary, resized_edg
```

다음과 같이 작성해서 main_window.cpp에 전달해 UI에 출력함으로써 사이즈를 비율에 맞으면서 줄어들도록 했다.

- 가우시안 블러

이미지의 노이즈를 줄이고 부드러운 효과를 주기 위해서 사요왼다. 가우시안 함수를 이용해서 각 픽셀의 값을 주변 픽셀들을 이용해서 가중치를 곱한 평균값으로 변경해 적용하는 방법이다. 중심에서 멀어질수록 가중치가 작아진다는 특징이고 이런 방법으로 중앙 픽셀의 값에 주변 픽셀의 값으로 영향을 주어서 더 부드러운 이미지가 완성된다.



가우시안 함수

- erode, dilate

erode는 침식이라는 뜻으로 binary image 이용했을 때 흰색의 작은 노이즈들과 본 이미지가 뜨는데 이 과정에서 작은 흰 점들을 없애기 위해서 erode를 적용하면 상대적으로 큰 이미지가 남아서 더 선명한 이미지를 추출받을 수 있다. 또한 dilate는 팽창이라는 뜻으로 만약

이미지가 제대로 추출을 못해서 흰색 원 모양에 검정색 빈 공간이 생길경우 흰 점들을 팽창시켜서 더 채워진 이미지를 얻을 수 있다.

```
node->declare_parameter("Erode", 1);
node->declare_parameter("Dilate", 2);
```

erode,dilate의 초기값이다.

```
int erode_iter = getParameter("Erode");
int dilate_iter = getParameter("Dilate");

if (erode_iter > 0)
{
    cv::erode(mask, mask, cv::Mat(), cv::Point(-1, -1), erode_
}
if (dilate_iter > 0)
{
    cv::dilate(mask, mask, cv::Mat(), cv::Point(-1, -1), dila
}
```

다음과 같은 방식으로 erode, dilate를 저장받고 cv::Mat을 통해서 각 이미지에 erode, dilate를 적용하였다.

- ROI 관심영역

ROI 영역을 지정하기 위해서 4개의 점 x, y 좌표를 입력받는 것이 필요하다. Slider, Spinbox를 통해서 입력받았고 이를 통해 조건문으로 4 포인트가 있는 도형을 저장받는다. 그 후 point 라는 도형의 배열을 동적할당해서 만든다. 이 때 vector를 통해서 동적할당하였다. 그 때 fillPoly라는 도형을 채우는 함수가 필요하다. 동적할당 된 4포인트가 있는 다각형의 내부를 픽셀 데이터로 채우고 그 외부를 검정색으로 표현함으로써 관심영역을 지정할 수 있고 전에 쓴 함수에 적용하면 관심영역 내에서 binary_image, 또는 H,S,V가 제한된 영역 내에서 송출된다.

- Canny 이미지

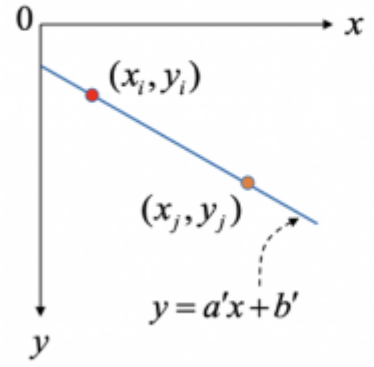
외곽선을 검출하는 함수이다. 외곽선을 검출하기 위해서 노이즈 제거를 먼저 한다, Gradient 값의 높은 값을 찾아서 최대가 아닌 픽셀을 0으로 만들어서 검정색과 흰색으로 구현한다. Lower 값과 Upper 값을 조절해서 최대값, 최소값을 조절해서 외곽선을 뚜렷하게 보일 수 있다.

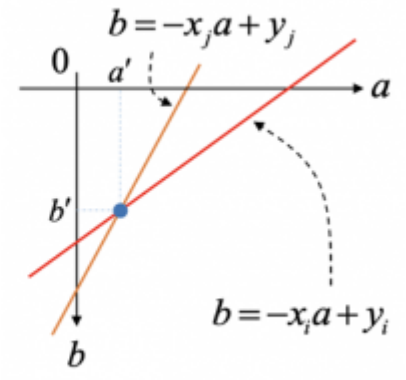
```
int threshold1 = getParameter("Threshold1");
int threshold2 = getParameter("Threshold2");

cv::Mat edges;
cv::Canny(binary_image, edges, threshold1, threshold2);
```

이를 통해서 적절한 slider, spinbox 조절값을 변수에 저장받아서 canny image를 구성한다. binary_img와 edge, 그리고 변수들의 값을 이용해 canny image를 구성하였다.

- 호프라인 함수

canny image와 유사하지만 호프라인 검출 방식은 2차원의 x, y 좌표 안에 직선의 방정식을 파라미터 공간으로 변화해 직선을 검출하는 방식이다. 쉽게 표현하자면 y = ax + b라는 식을 a,b라는 공간을 표현하기 위해서 b = -ax + y라는 식으로 바꾼다. 그 후 두 직선의 교점을 (a′, b′)로 표현하고 이제 이 두 점(a′, b′), (b′, a′)을 지나는 직선은 한가지 이므로 유일한 직선을 뽑아낼 수 있다. 이 방식으로 이미지 상에서의 직선을 검출해 낸다.





호프라인 함수는 다음과 같이 구성된다.

```
void HoughLinesP(InputArray image, OutputArray lines, double rho,
                 double theta, int threshold, double minLineLength = 0,
                 double maxLineGap = 0);
# => lines
```

- 레이블링 함수

레이블링이란 인접한 같은 값을 갖는 픽셀끼리 하나의 그룹으로 묶어주는 것이다. 쉽게 말해 비슷한 영역에 이름을 부여해 같은 영역으로 치환해 생각한다. 레이블링이 필요한 이유로는 영상 즉, 비전과 같은 작업에서 노이즈를 줄여서 카메라의 선명도를 높여주는 것이 중요한데

레이블링을 통해 좀 더 선명하고 깨끗한 이미지를 얻을 수 있다. 이 레이블링 함수를 사용하려면 4-neighbors와 8-neighbors가 필요하다. 8-neighbors는 대각선에 있는 점의 좌표까지도 인식해 연속성을 따진다. 총 9개의 칸에서 스캔 방향은 왼쪽에소 오른쪽, 위에서 아래가 일반적이다. 즉, 행렬식으로 봤을 때 1행 1열을 확인하고 1행 2열을 확인하고 없으면 2행 1열로 넘어가는 방식이다. 이런 식으로 확인을 맞춘 뒤 비슷한 객체끼리 이름을 매겨서 구별해주는 역할이다.

```
Mat img = imread("bacteria.tif");

Mat img_gray;
cvtColor(img, img_gray, COLOR_BGR2GRAY);

Mat img_threshold;
threshold(img_gray, img_threshold, 100, 255, THRESH_BINARY_
INV);

Mat img_labels, stats, centroids;
int numOfLables = connectedComponentsWithStats(img_threshol
d, img_labels, stats, centroids, 8, CV_32S);
```

이런식으로 openCV에서 제공된 기본 함수를 통해서 labeling을 작업할 수 있다.

코드 설명

image를 띄우는 image_node.hpp와 UI에 반영 및 UI의 슬라이더 값을 반영하는 main_window.hpp 그리고 파라미터 값을 설정하는 qnode.hpp로 구성되어 있다.

```
class ImageNode : public QThread
{
    Q_OBJECT

public:
    ImageNode();
    ~ImageNode();
```

```
    void setParameter(const std::string &name, int value); //
    int getParameter(const std::string &name);

protected:
    void run();

private:
    void imageCallback(const sensor_msgs::msg::Image::SharedP
    std::shared_ptr<rclcpp::Node> node;
    rclcpp::Subscription<sensor_msgs::msg::Image>::SharedPtr

signals:
    void newImage(const QPixmap &original, const QPixmap &bin
    void rosShutDown();
};
```

image_node에서는 getParameter 값을 통해서 적용되거나 설정된 파라미터의 값을 입력 받는다. 그 후 imageCallback이라는 Private 멤버 변수 내에서 image 띄우는 부분을 설정하고 subscribe기능이 구현되어 있다.

```
class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  MainWindow(QWidget *parent = nullptr);
  ~MainWindow();
  QNode *qnode;
  ImageNode *imageNode; // QNode 대신 ImageNode 사용

public slots:
  // 이미지 표시 슬롯
  void displayImages(const QPixmap &original, const QPixmap &
  void updateDisplaySize();

  // HSV 파라미터 업데이트 슬롯
  void updateParameterHLower(int value);
```

```
  void updateParameterHUpper(int value);
  void updateParameterSLower(int value);
  void updateParameterSUpper(int value);
  void updateParameterVLower(int value);
  void updateParameterVUpper(int value);

  // Canny, Erode, Dilate 파라미터 업데이트 슬롯
  void updateParameterThreshold1(int value);
  void updateParameterThreshold2(int value);
  void updateParameterErode(int value);
  void updateParameterDilate(int value);

  // 이미지 크기 파라미터 업데이트 슬롯
  void updateParameterImageWidth(int value);
  void updateParameterImageHeight(int value);

private:
  Ui::MainWindowDesign *ui;
  QLabel *labelImageOriginal; // 이미지를 표시할 QLabel
  int displayWidth = 640;
  int displayHeight = 480;
  void closeEvent(QCloseEvent *event);
};
```

main_window.hpp에서는 이미지 표시, HSV 파라미터 슬롯, canny, erode, dilate, image크기 등의 슬롯이 있고

```
    node->declare_parameter("H_lower", 0);
    node->declare_parameter("H_upper", 180);
    node->declare_parameter("S_lower", 0);
    node->declare_parameter("S_upper", 255);
    node->declare_parameter("V_lower", 0);
    node->declare_parameter("V_upper", 255);
    node->declare_parameter("Threshold1", 75);
    node->declare_parameter("Threshold2", 175);
    node->declare_parameter("Erode", 1);
    node->declare_parameter("Dilate", 2);
```

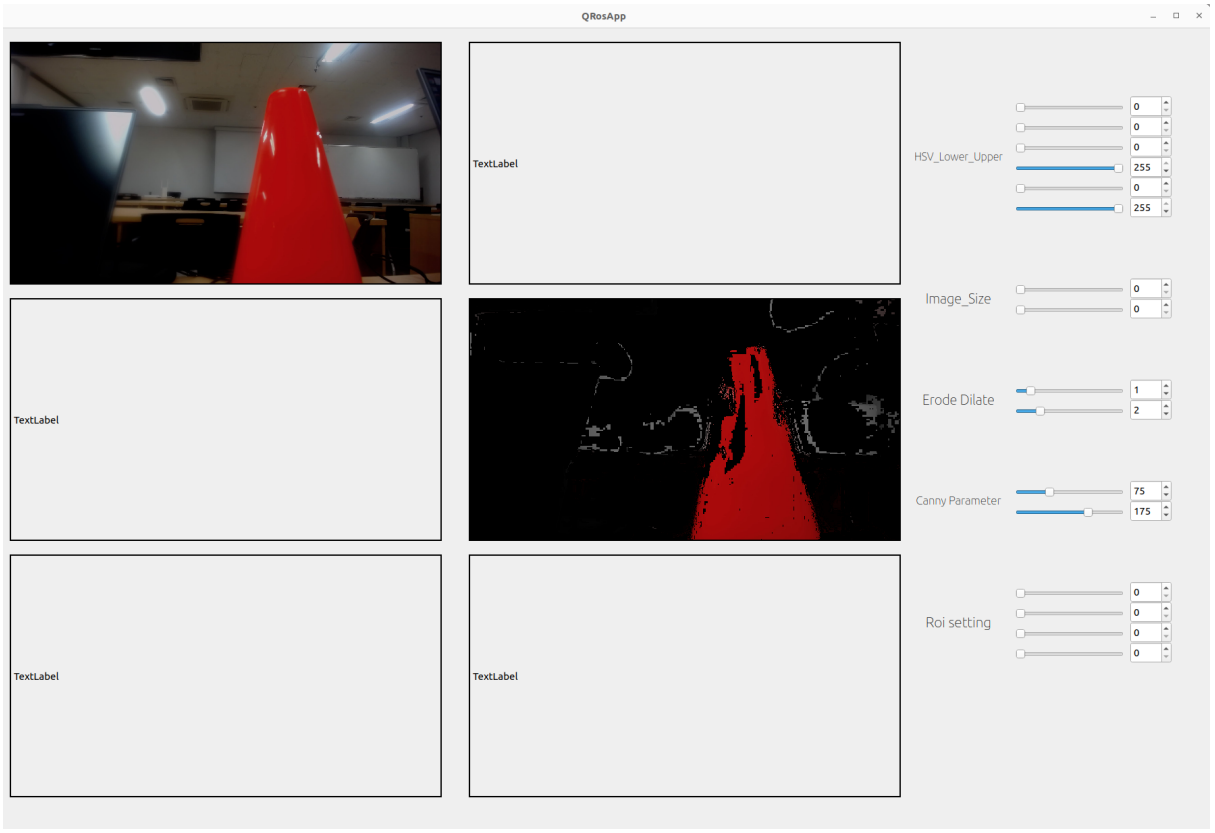
```
    node->declare_parameter("image_width", 640);
    node->declare_parameter("image_height", 480);
```

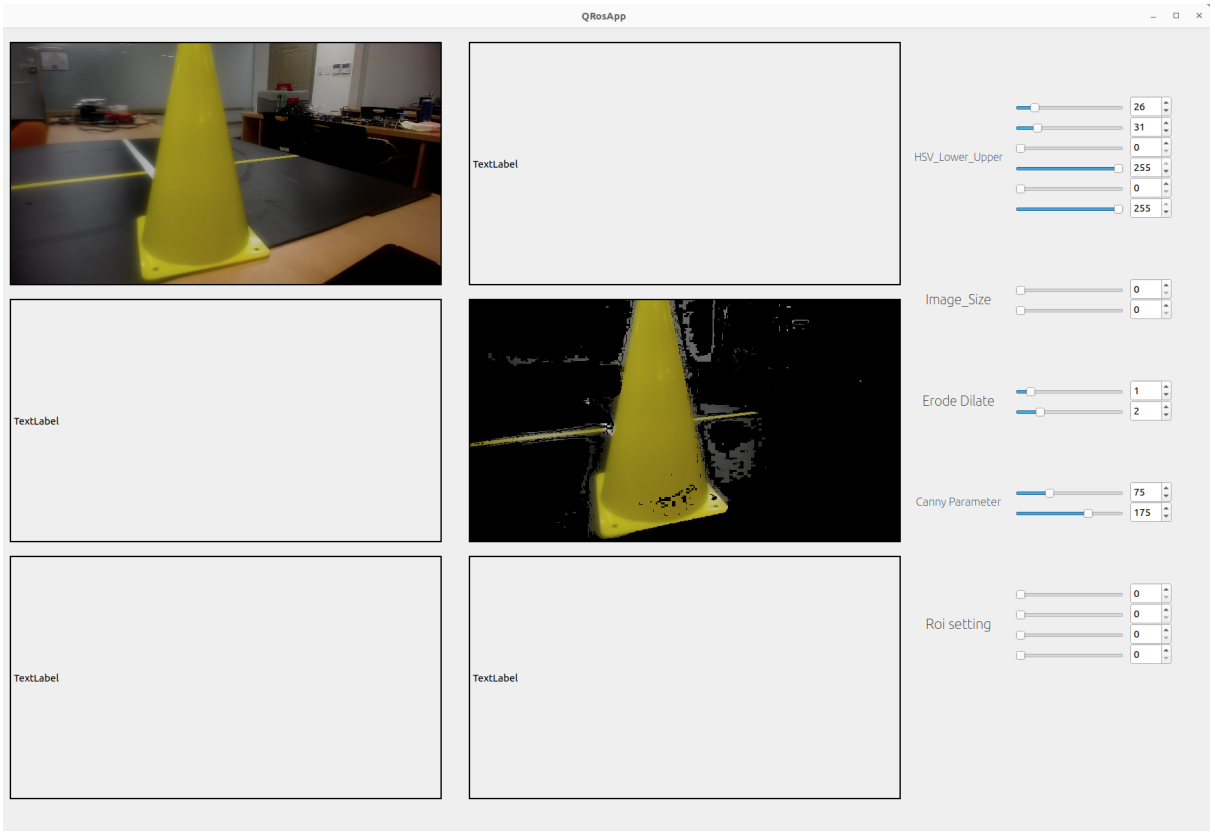
image_node.cpp에서 파라미터 값의 초기값을 설정하고 뒤에 getParameter를 통해 값을 읽어온다.

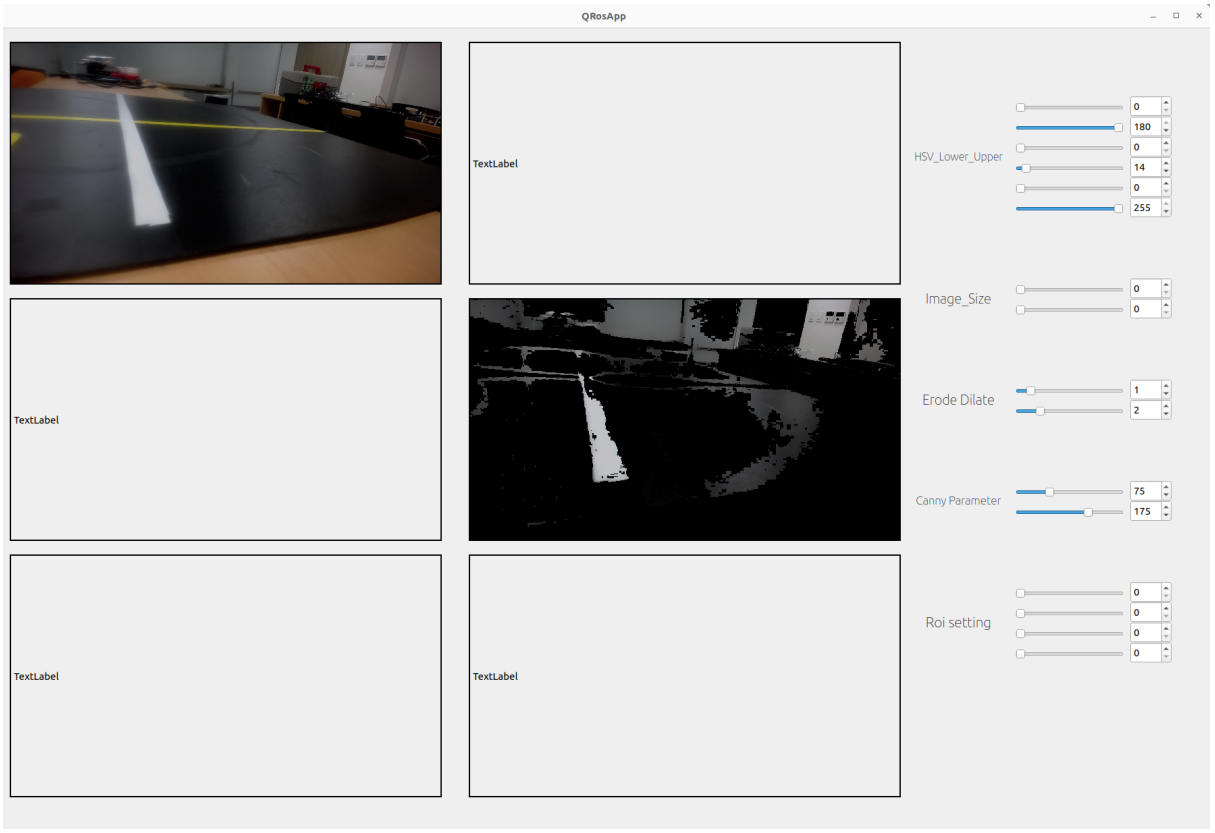
```
void MainWindow::updateDisplaySize()
{
  displayWidth = ui->spinWidth->value();
  displayHeight = ui->spinHeight->value();

  // 이미지를 표시하는 라벨의 크기 조정
  ui->labelImageOriginal->setFixedSize(displayWidth, displayH
  ui->labelBinaryWhite->setFixedSize(displayWidth, displayHei

  qnode->setParameter("image_width", displayWidth);
  qnode->setParameter("image_height", displayHeight);

  qnode->updateImageSize(displayWidth, displayHeight);
}
```

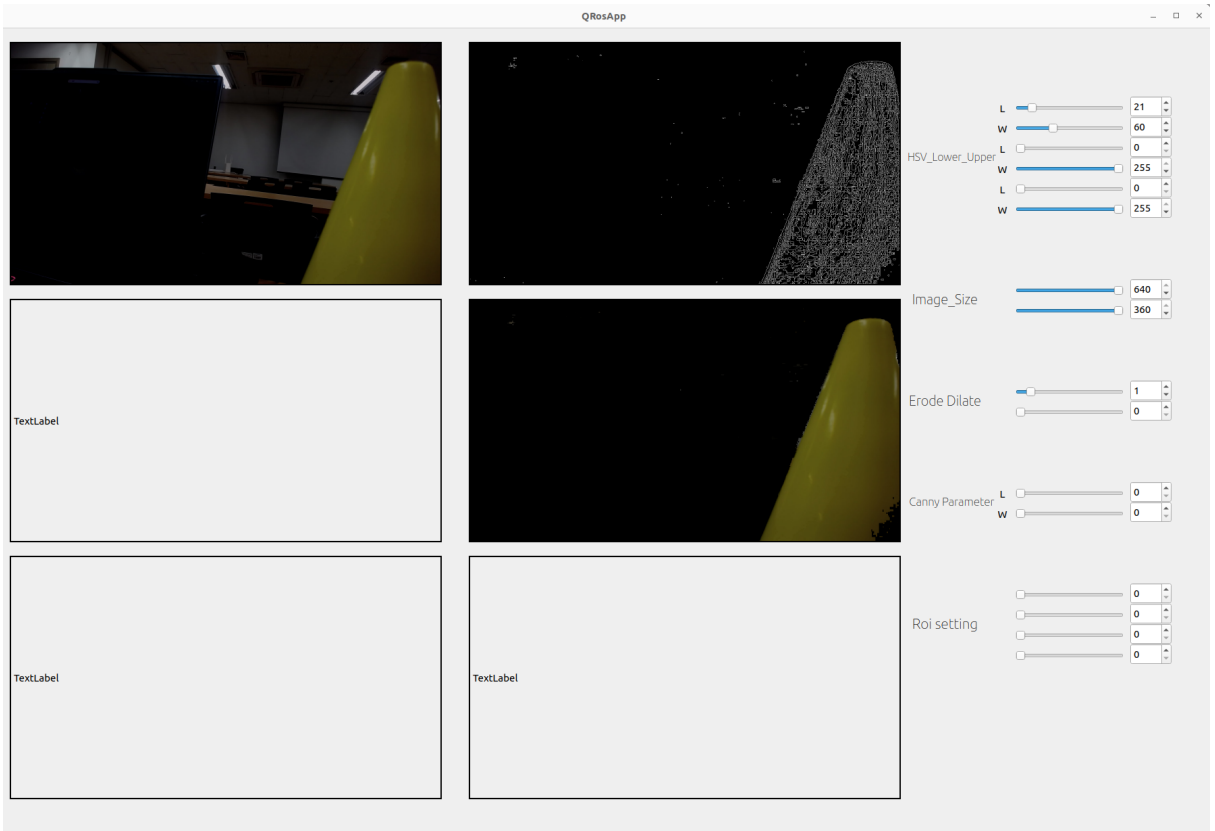
main_window.cpp에서는 Qnode와 imageNode가 connect 되어 있고 UI의 slider, spinbox와도 연결이 되어 있어 값을 조정할 수 있다. 물론 slider와 spinbox 역시 상호작용 가능하다.

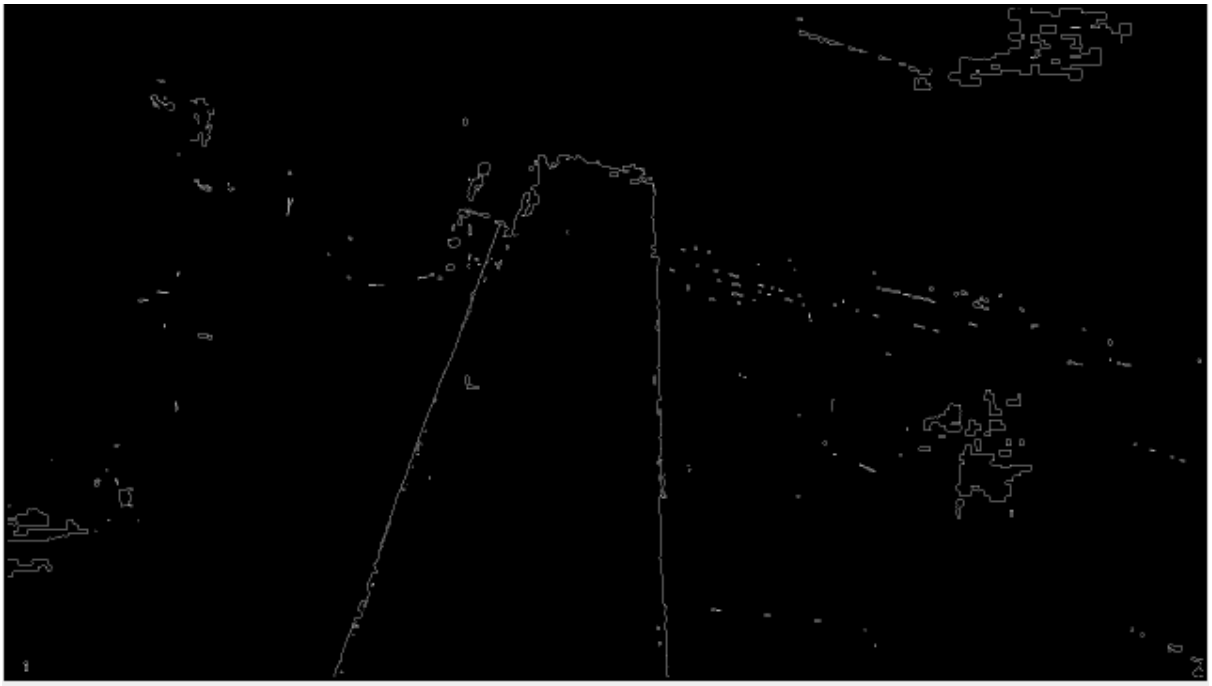
실행 결과

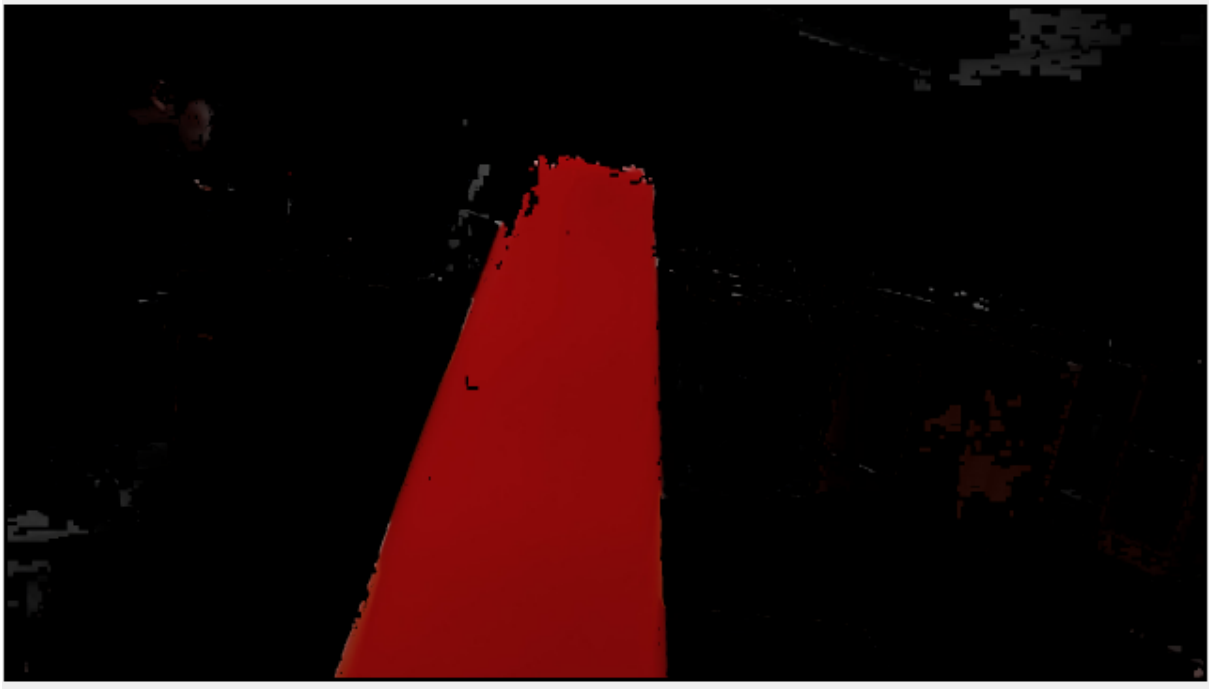

QRosApp
TextLabel
TextLabel
TextLabel
TextLabel
HSV_Lower_Upper
0
0
0
255
0
255
Image_Size
0
0
Erode Dilate
1
2
Canny Parameter
75
175
Roi setting
0
0
0
0

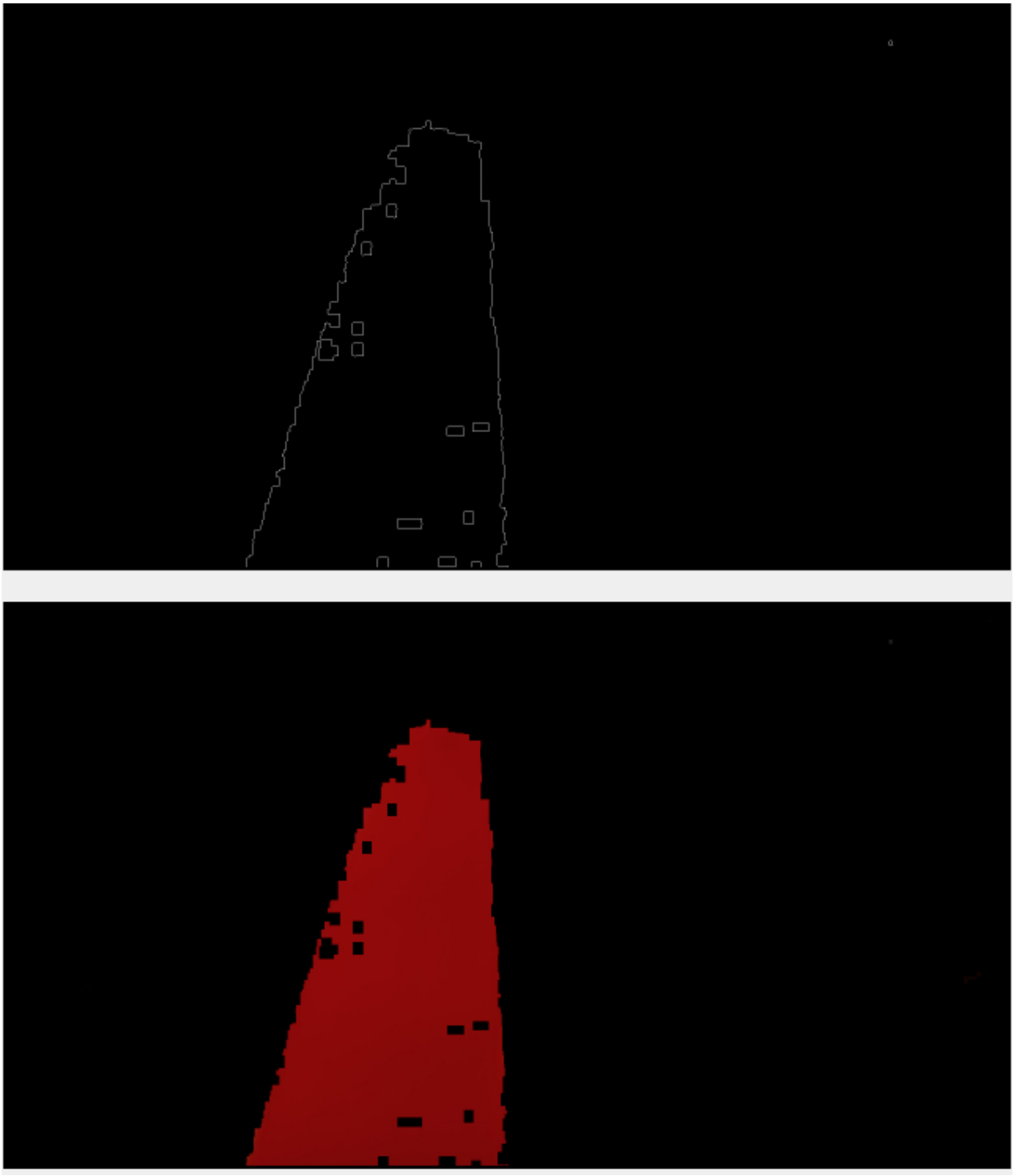

QRosApp
TextLabel
TextLabel
TextLabel
TextLabel
HSV_Lower_Upper
26
31
0
255
0
255
Image_Size
0
0
Erode Dilate
1
2
Canny Parameter
75
175
Roi setting
0
0
0
0

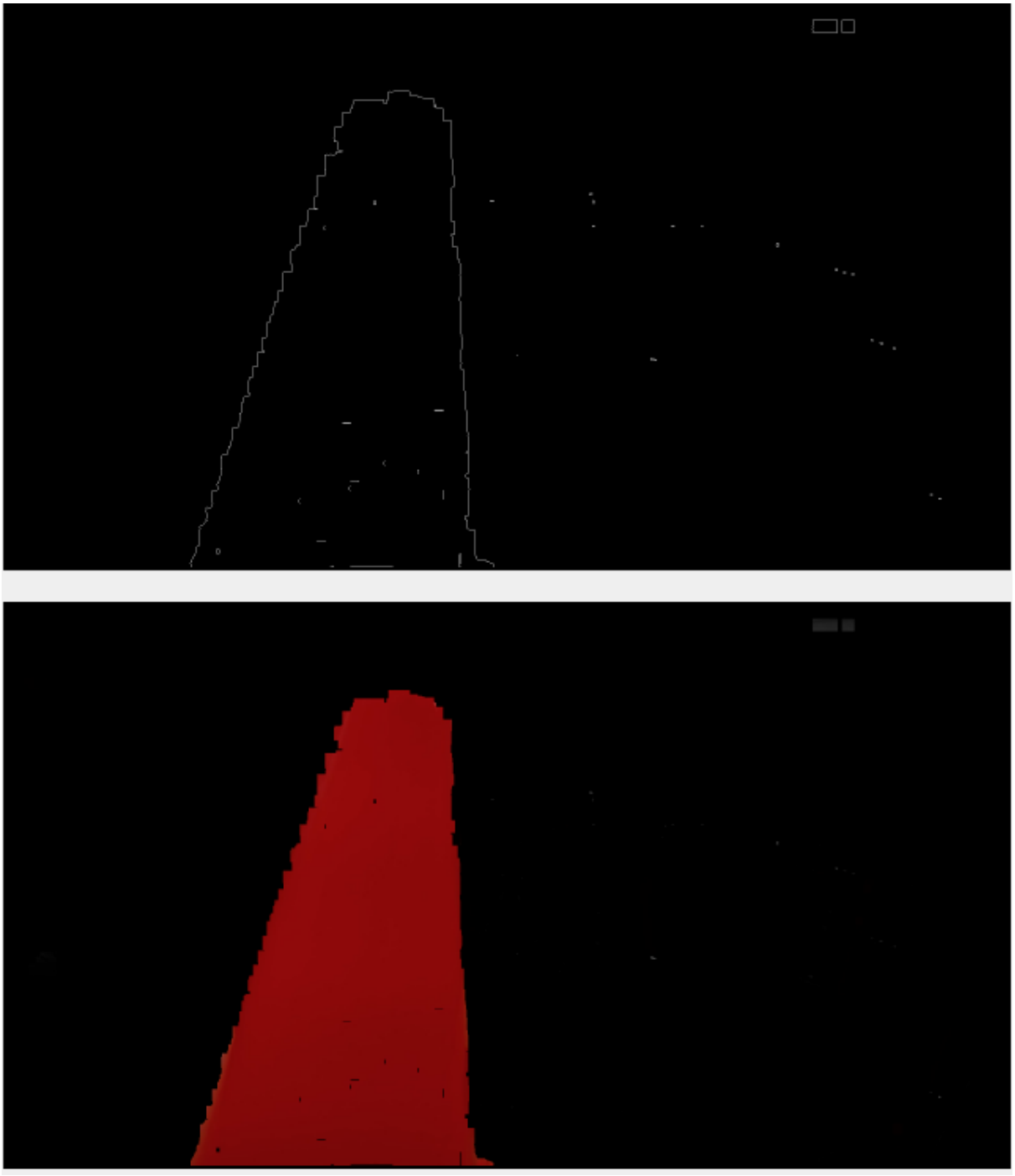

QRosApp
TextLabel
TextLabel
TextLabel
TextLabel
HSV_Lower_Upper
0
180
0
14
0
255
Image_Size
0
0
Erode Dilate
1
2
Canny Parameter
75
175
Roi setting
0
0
0
0


QRosApp
TextLabel
TextLabel
TextLabel
HSV_Lower_Upper
L
21
W
60
L
0
W
255
L
0
W
255
Image_Size
640
360
Erode Dilate
1
0
Canny Parameter
L
0
W
0
Roi setting
0
0
0
0









최하단의 3가지는 순서대로 기본, 위는 erode, 아래는 dilate 작업이 처리된 결과이다.