지능형 SoC 로봇워

출전자격 TEST - SoC Taekwon

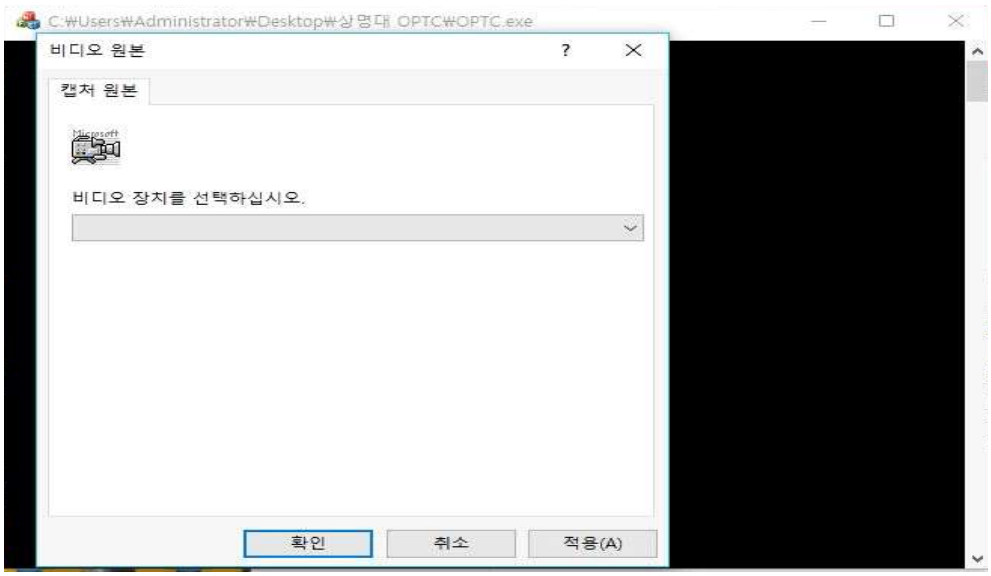
SW 매뉴얼

상명대학교 OPTC

사용 설명서

1. OPTC.exe 실행

2. 비디오 장치에서 Logitech HD webcam C270을 선택 후 확인 버튼 클릭



3. 원하는 대상을 카메라에 촬영

소스 설명서

1. RGB값을 HIS값으로 변환

```
void rgb2hsi(unsigned char *originalImage, unsigned short *hImage, unsigned char *sImage, unsigned char *iImage, int height, int width)
{
    int i, j;
        double minc, angle;

        int red, green, blue;
        float fR, fG, fB;

        for (i = 0; i < HEIGHT; i++) {
                for (j = 0; j < WIDTH; j++) {
                        red = originalImage[(i * WIDTH + j) * 3 + 2];
                        green = originalImage[(i * WIDTH + j) * 3 + 1];
                        blue = originalImage[(i * WIDTH + j) * 3 + 0];


                        //정규화된 RGB로 변환
                        if (red == 0 && green == 0 && blue == 0)
                        {
                                fR = 0.0f;
                                fG = 0.0f;
                                fB = 0.0f;
                        }
                        else
                        {
                                fR = (float)red / (red + green + blue);
                                fG = (float)green / (red + green + blue);
                                fB = (float)blue / (red + green + blue);
                        }

                        minc = MIN(fR, fG);
                        minc = MIN(minc, fB);

                        iImage[i * WIDTH + j] = (unsigned char)((red + green + blue) / 3);

                        if ((fR == fG) && (fG == fB)){          // Gray Scale
                                hImage[i * WIDTH + j] = 0;
                                sImage[i * WIDTH + j] = 0;
                                continue;
                        }
                        else {
                                //sImg[i * width + j] = (BYTE)(255.0 - 3.0 * minc *255 / (red + green + blue));
                                sImage[i * WIDTH + j] = (BYTE)(1.0f - 3.0f*minc *  255.0f);//fR+fG+fB = 1.00이가

                                //angle = (((red - green) + (red - blue)) >> 1) / (double)(sqrt((double)((red - green) * (red - green) + (red - blue) * (green - blue))));
                                angle = (fR - 0.5f*fG - 0.5f*fB) / (float)sqrt((fR - fG)*(fR - fG) + (fR - fB)*(fG - fB));
                                hImage[i * WIDTH + j] = (WORD)(acos(angle) * 57.29577951);
                        }

                        if (fB > fG) hImage[i * WIDTH + j] = (WORD)(360 - hImage[i * WIDTH + j]);

                        //sImg[i * width + j] += 255.0f;
                }
        }
        return;

}
```

2. 영상 이진화

```
// h,s 값 기준으로 이진화
for (i = 0; i < HEIGHT; i++) {
    for (j = 0; j < WIDTH; j++) {
        //r
        if ((hImage[i*WIDTH + j] < 5 || hImage[i*WIDTH + j]>350)
            && sImage[i*WIDTH + j] > 70) {
            binary_Image[i*WIDTH + j] = 255;
            check_color[i*WIDTH + j] = 1;
            //histo[1]++;
        }
        //y
        else if ((hImage[i*WIDTH + j] < 63 && hImage[i*WIDTH + j]>35)
            && sImage[i*WIDTH + j] > 70) {
            binary_Image[i*WIDTH + j] = 255;
            check_color[i*WIDTH + j] = 2;
            //histo[2]++;
        }
        //g
        else if ((hImage[i*WIDTH + j] < 95 && hImage[i*WIDTH + j]>63)
            && sImage[i*WIDTH + j] > 50) {
            binary_Image[i*WIDTH + j] = 255;
            check_color[i*WIDTH + j] = 3;
            //histo[3]++;
        }

        //b
        else if ((hImage[i*WIDTH + j] < 210 && hImage[i*WIDTH + j]>190)
            && sImage[i*WIDTH + j] > 117 && sImage[i*WIDTH + j] < 198) {
            binary_Image[i*WIDTH + j] = 255;
            check_color[i*WIDTH + j] = 4;
            //histo[4]++;
        }

        else {
            binary_Image[i*WIDTH + j] = 0;
            check_color[i*WIDTH + j] = 0;
        }
    }
}
```

3. 라벨링 (Grass fire)

```
// 라벨링(Grassfire) 후 가장 큰 영역 구분
m_BlobColoring(binary_Image, HEIGHT, WIDTH);

int gX = 0, gY = 0, cnt = 0;

//라벨링 후 색 검사
for (i = 0; i < HEIGHT; i++) { //아웃풋에 넣기
    int temp1 = i * WIDTH;

    for (j = 0; j < WIDTH; j++)
    {
        if (binary_Image[temp1 + j] == 255) {
            histo[check_color[temp1 + j]]++;
        }
    }
}
```

4. 최빈 값 이용해서 검은색 처리

```
//가장 많이 검출된 색 검출
for (i = 1; i < 5; i++)
    if (histo[manyColor] < histo[i]) {
        manyColor = i;
    }
if (manyColor == 1)
    printf("빨간색₩n");
else if(manyColor==2)
    printf("노란색₩n");
else if(manyColor==3)
    printf("초록색₩n");
else if(manyColor==4)
    printf("파란색₩n");
//printf("%d", manyColor);
//검출된 부분의 중심 찾고 적은 색상 제거
for (i = 0; i < HEIGHT; i++) { //아웃풋에 넣기
    int temp1 = i * WIDTH;

    for (j = 0; j < WIDTH; j++)
    {
        if (binary_Image[temp1 + j] == 255 && check_color[temp1+j] == manyColor) {
            gX += j;
            gY += i;
            cnt++;
        }
        else binary_Image[temp1 + j] = 0;
    }
}
```

5. 예상 경로 추적

```
    // 예상경로 추적
    nextX=gX+(gX-beforeX);
    nextY=gY+(gY-beforeY);
    if (nextX > WIDTH-16) nextX = WIDTH - 15;
    else if(nextX < 16) nextX = 16;

    if (nextY > HEIGHT-16) nextY = HEIGHT - 15;
    else if (nextY < 16) nextY = 16;
    for (i = -15; i < 15; i++) {
        binary_Image[nextX+i+nextY*WIDTH] = 254;
    }

    for (i = -15; i < 15; i++) {
        binary_Image[nextX +(nextY+i)*WIDTH] = 254;
    }


    beforeX = gX;
    beforeY = gY;
}

for (i = 0; i < HEIGHT; i++) { //아웃풋에 넣기
    int temp1 = i * WIDTH;

    for (j = 0; j < WIDTH; j++)
    {
        int temp2 = temp1 + j;

        if (binary_Image[temp2] == 255 ) {
            output_Image[temp2 * 3 + R] = input_Image[temp2 * 3 + R];
            output_Image[temp2 * 3 + G] = input_Image[temp2 * 3 + G];
            output_Image[temp2 * 3 + B] = input_Image[temp2 * 3 + B];
            //output_Image[temp2 * 3 + R] = 255;
            //output_Image[temp2 * 3 + G] = 255;
            //output_Image[temp2 * 3 + B] = 255;

        }
        //예상경로 중심
        else if (binary_Image[temp2] == 254) {
            output_Image[temp2 * 3 + R] = 255;
            output_Image[temp2 * 3 + G] = 255;
            output_Image[temp2 * 3 + B] = 255;
        }
        else {
            output_Image[temp2 * 3 + R] = 0;
            output_Image[temp2 * 3 + G] = 0;
            output_Image[temp2 * 3 + B] = 0;
        }

    }
}
```