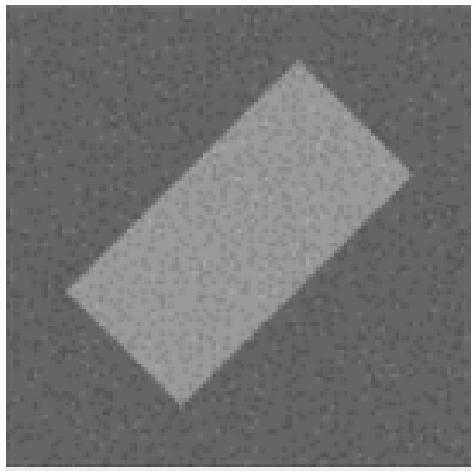
OpenCV를 활용한 이미지 처리3

무엇을 해볼까요?

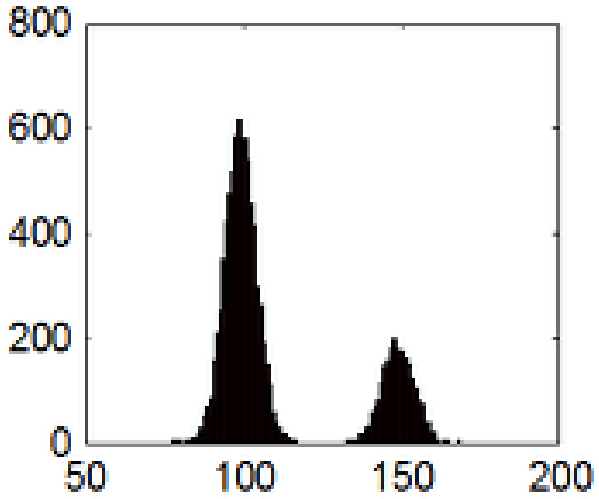
- pixel과 image
- 특정 색상을 찾아라
- 영상 2진화 및 ROI 영역
- 이미지 Convolution과 이미지 블러링(Blur)
- 이미지에서 특징 추출(corner, line)
- Template Matching으로 물체인식
- 숨은 그림 찾기
- 이미지 변환(이동, 회전, Affine, Perspective Transform)

무엇을 해볼까요?

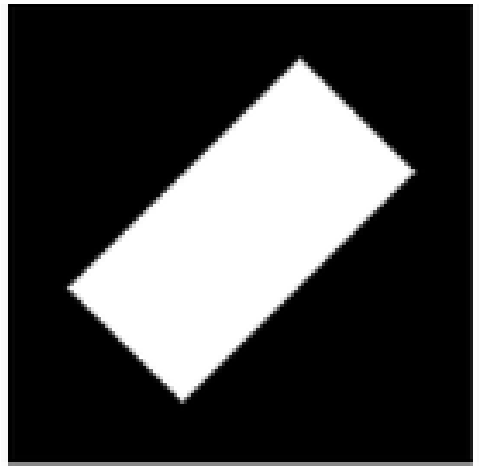
- 영상 2진화
  - 어떤 주어진 임계값(threshold)보다 밝은 픽셀들은 모두 흰색으로, 그렇지 않은 픽셀들은 모두 검은색으로 바꾸는 것을 지칭한다.



(a)



(b)



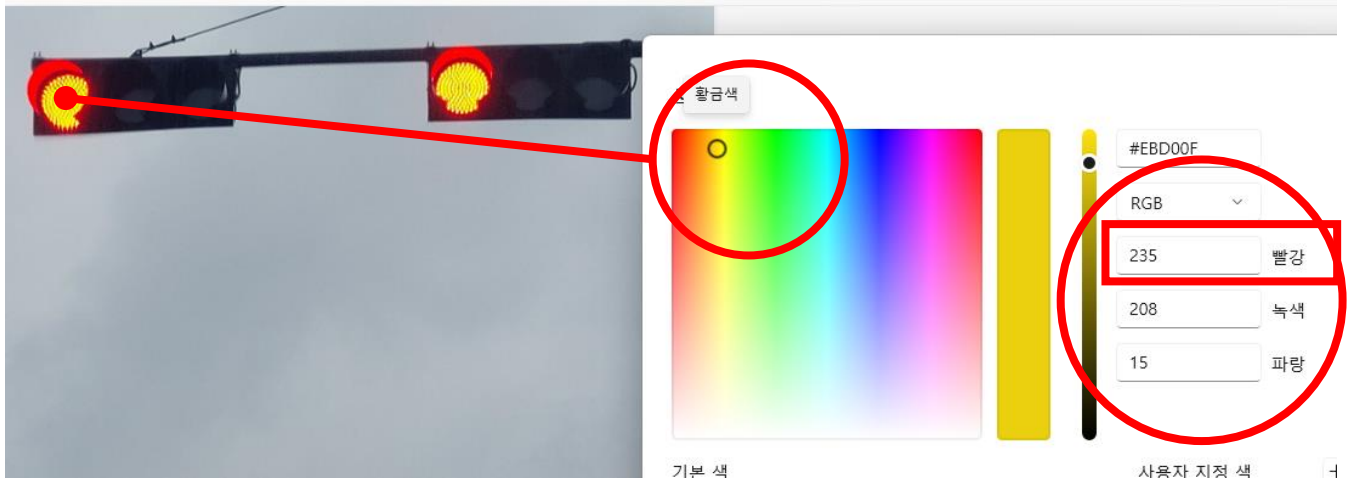
(c)

무엇을 해볼까요?

- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

무엇을 해볼까요?

- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?



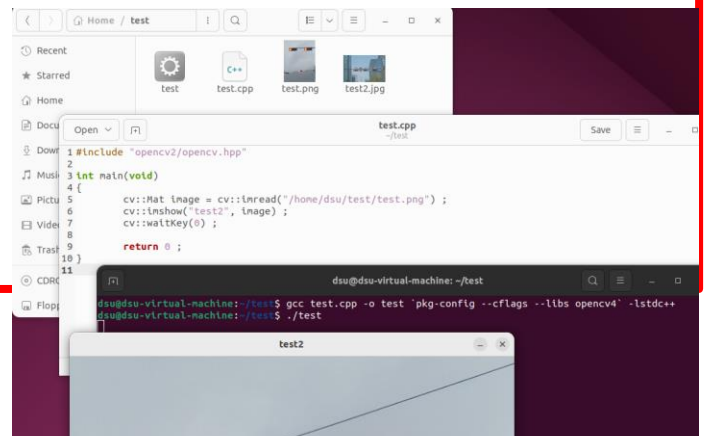
무엇을 해볼까요?

- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

다시한번 이미지를 읽어 **cv::imshow를 통해 화면에 표시**하자.

```
#include "opencv2/opencv.hpp"

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/test.png") ;
        cv::imshow("test2", image) ;
        cv::waitKey(0) ;

        return 0 ;
}
```

경로 주의



무엇을 해볼까요?

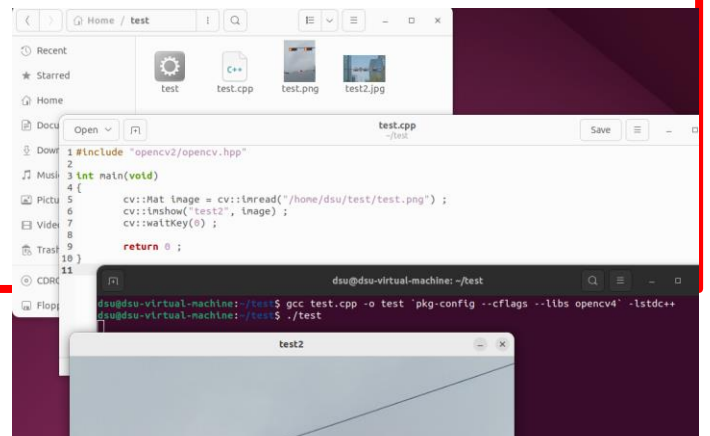
- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

다시한번 이미지를 읽어 **cv::imshow를 통해 화면에 표시**하자.

```
#include "opencv2/opencv.hpp"

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/test.png") ;
        cv::imshow("test2", image) ;
        cv::waitKey(0) ;

        return 0 ;
}
```

CPP파일 컴파일

```
$ gcc test.cpp -o test `pkg-config --cflags --libs opencv4` -lstdc++
```



무엇을 해볼까요?

- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

실행

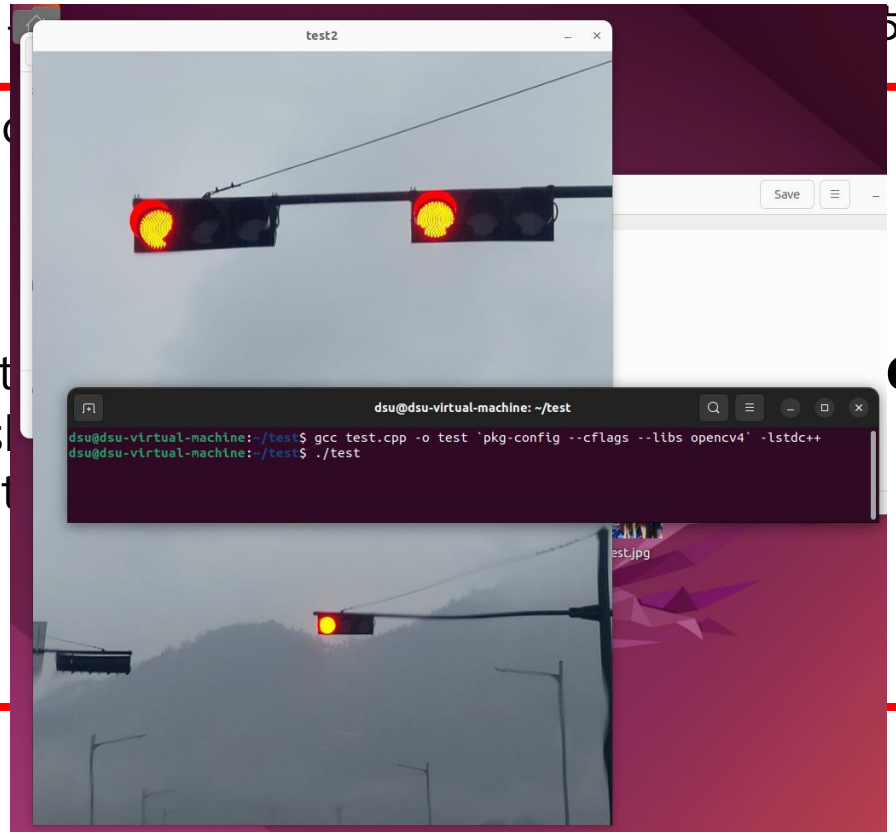
```
$ ./test
```

다시한번 이미지

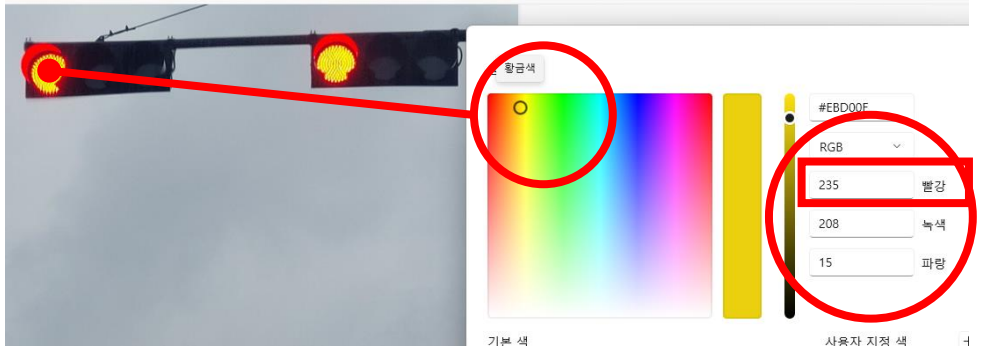
하자.

```
#include "openc

int main(void)
{
        cv::Mat
        cv::ims
        cv::wait

        return
}
```

est/test.png") ;



무엇을 해볼까요?



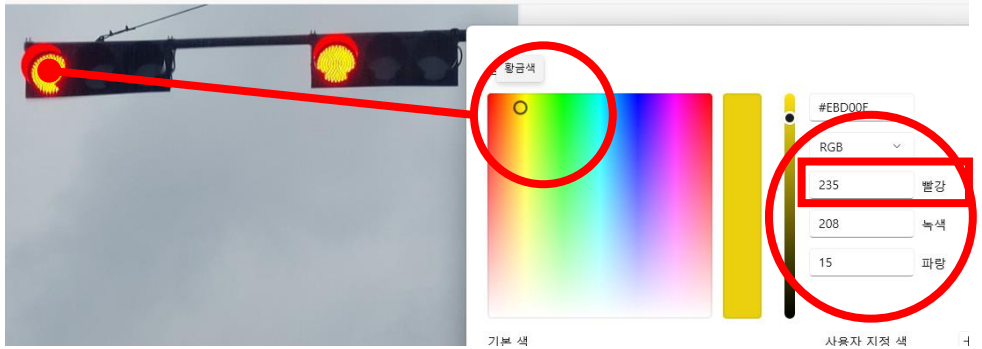
- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

Red값이 큰 값을 갖는 pixel만 남기고 나머지는 모두 RGB(0,0,0) 으로 만들자

```
if( red_value > 220  )
{
        image.data[red_index] = 255 ;   //red
        image.data[green_index] = 255 ;         //green
        image.data[blue_index] = 255 ;  //blue
}
else
{
        image.data[red_index] = 0 ;     //red
        image.data[green_index] = 0 ;   //green
        image.data[blue_index] = 0 ;    //blue
}
```

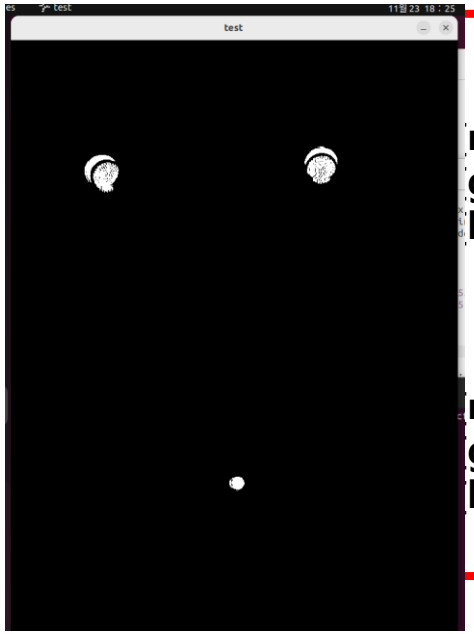
예제) 09_opencv/07_red_bin.cpp

무엇을 해볼까요?



- 영상 2진화
  - 아래의 이미지에서 신호등이 정지 신호인지 어떻게 알 수 있지?

Red값이 큰 값을 갖는 pixel만 남기고 나머지는 모두 RGB(0,0,0) 으로 만들자




```
[red_index] = 255 ;   //red
[green_index] = 255 ;          //green
[blue_index] = 255 ;  //blue


[red_index] = 0 ;     //red
[green_index] = 0 ;   //green
[blue_index] = 0 ;    //blue
```

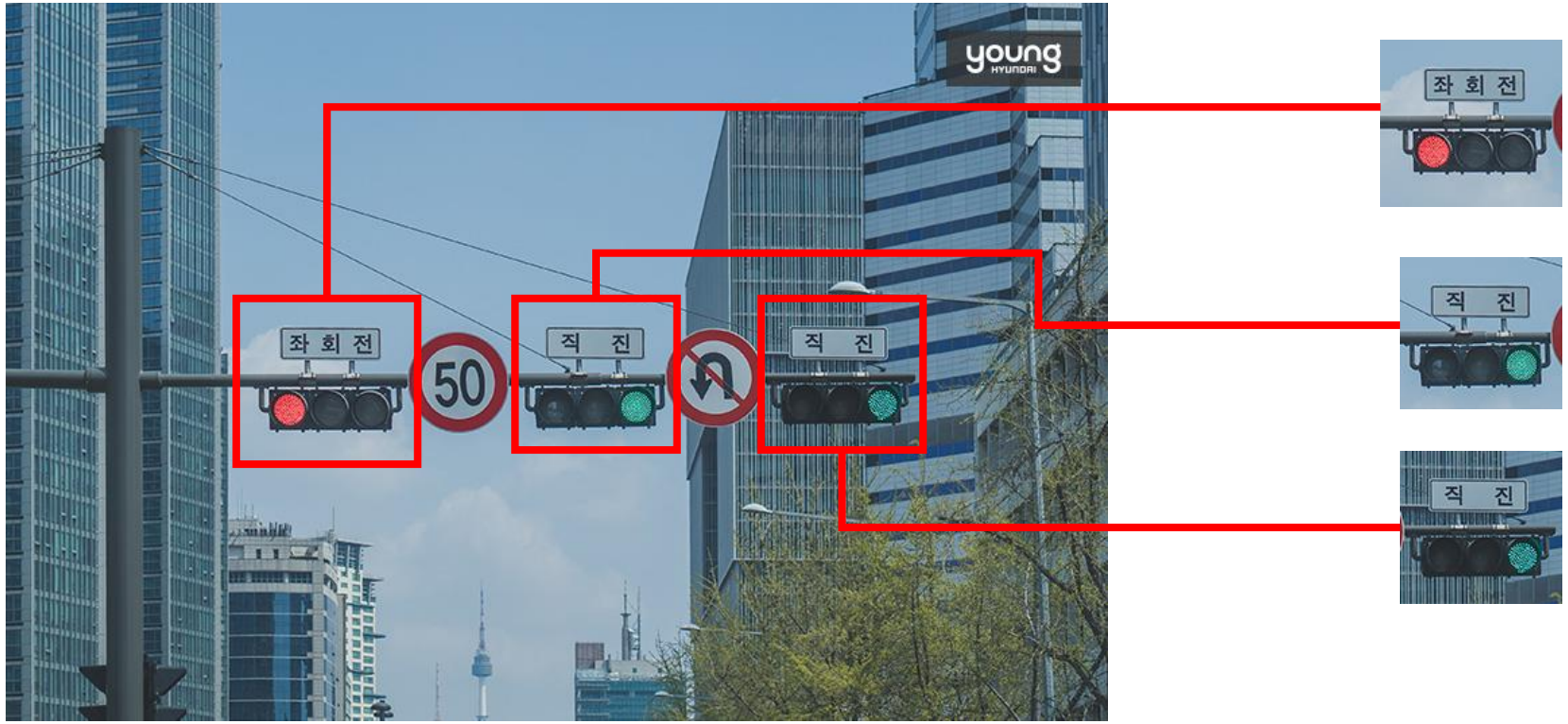
예제) 09_opencv/07_red_bin.cpp

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - ROI 란? 말 그대로 **영상 내에서 관심이 있는 영역**을 뜻합니다.

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - ROI 란? 말 그대로 **영상 내에서 관심이 있는 영역**을 뜻합니다.

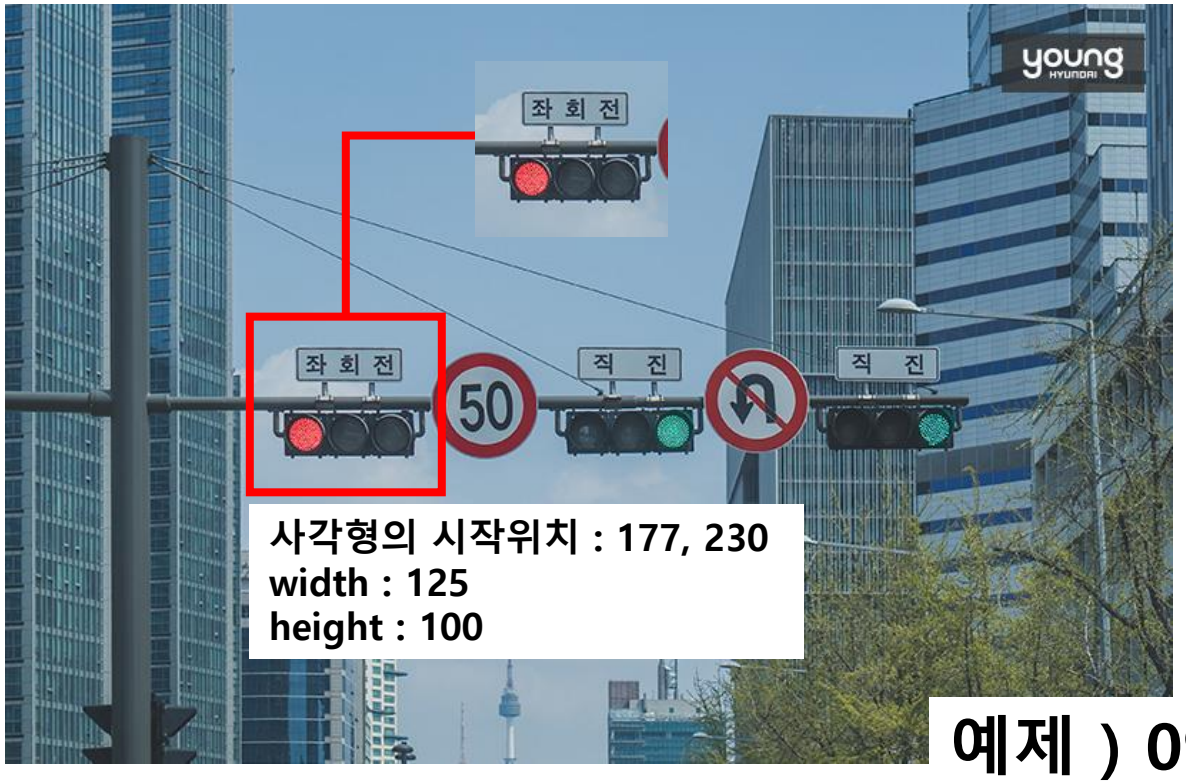


무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - 이미지에서 신호등 이미지만 잘라보자.

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - 이미지에서 신호등 이미지만 잘라보자.

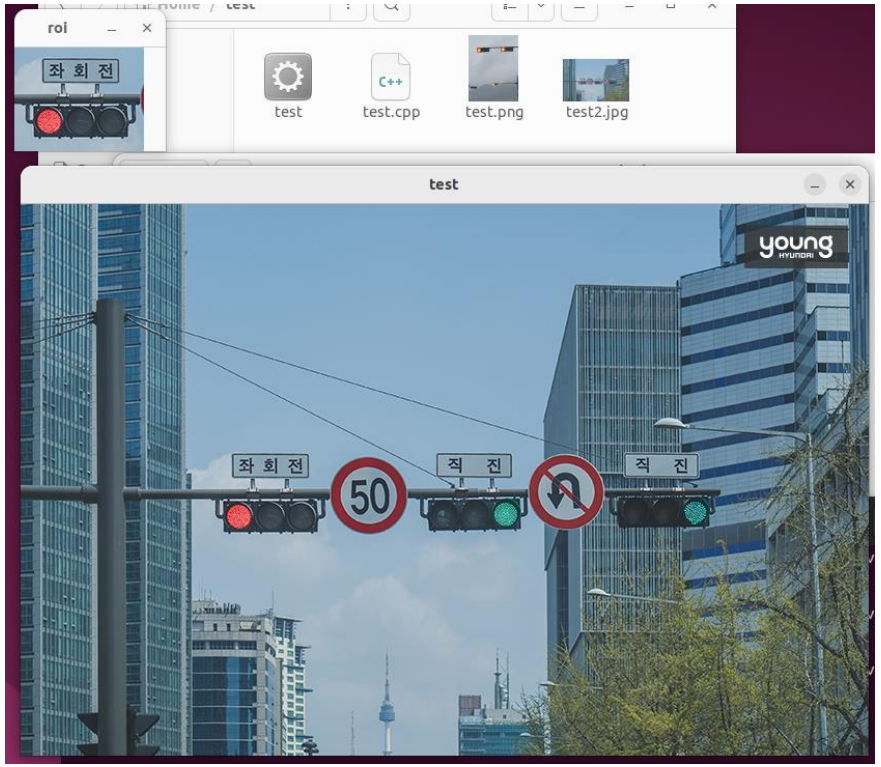



```
cv::Rect roi ;
roi.x = 177 ;
roi.y = 230 ;
roi.width = 125 ;
roi.height = 100 ;

cv::Mat roi_image ;
image(roi).copyTo(roi_image) ;
```

예제) 09_opencv/08_roi_image.cpp

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - 이미지에서 신호등 이미지만 잘라보자.

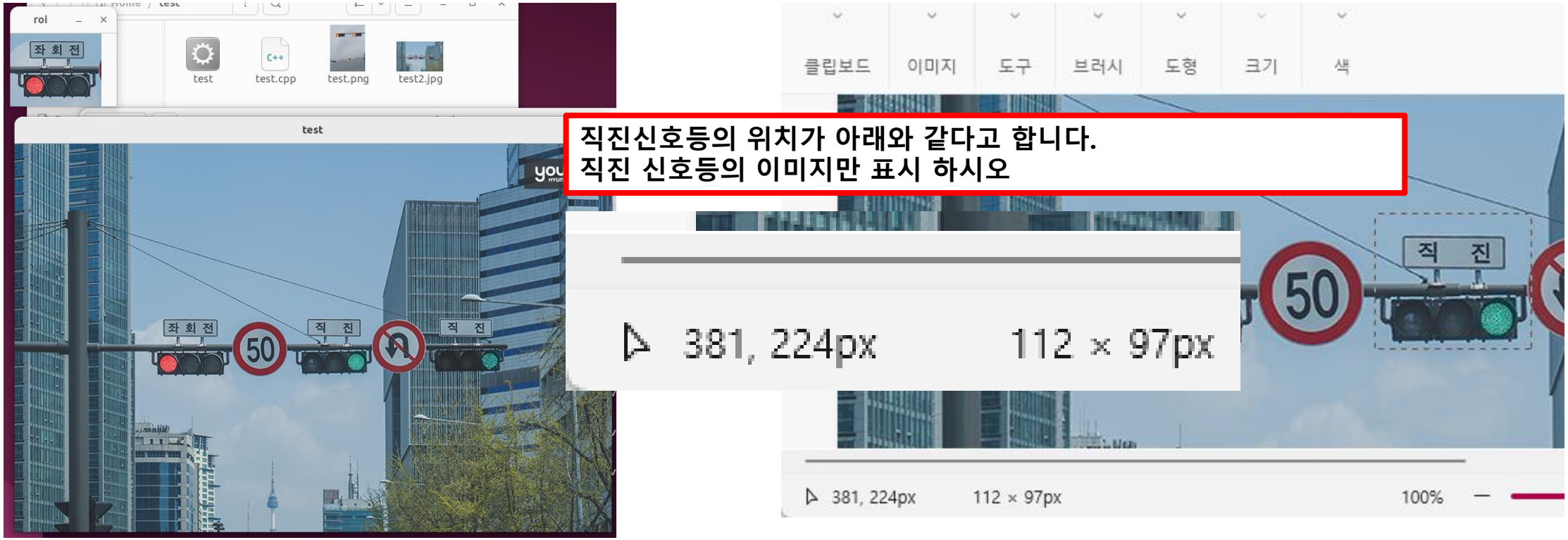



```
cv::Rect roi ;
roi.x = 177 ;
roi.y = 230 ;
roi.width = 125 ;
roi.height = 100 ;

cv::Mat roi_image ;
image(roi).copyTo(roi_image) ;
```

예제) 09_opencv/08_roi_image.cpp

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역)
  - QUIZ) 이미지에서 직진 신호등 이미지만 잘라보자.



무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역) 이미지에서 영상처리
  - ROI 이미지에서 신호등 색상을 분석하자.

관심 영역 안에서만 처리하기 때문에
전체 영역을 검색하는 것이 아니라 속도가 빠르고 정확도가 높다.

무엇을 해볼까요?

- 이미지 ROI(Region of interest 관심 영역) 이미지에서 영상처리
  - ROI 이미지에서 신호등 색상을 분석하자.


```
cv::Rect roi ;
roi.x = 177 ;
roi.y = 230 ;
roi.width = 125 ;
roi.height = 100 ;

cv::Mat roi_image = image(roi) ;

if( red_value > 220 )
{
        roi_image.data[red_index] = 255 ;       //red
        roi_image.data[green_index] = 255 ;     //gree
        roi_image.data[blue_index] = 255 ;      //blue
}
else
{
        roi_image.data[red_index] = 0 ;         //red
        roi_image.data[green_index] = 0 ;       //green
        roi_image.data[blue_index] = 0 ;        //blue
}
```

예제) 09_opencv/09_roi_image_bin.cpp

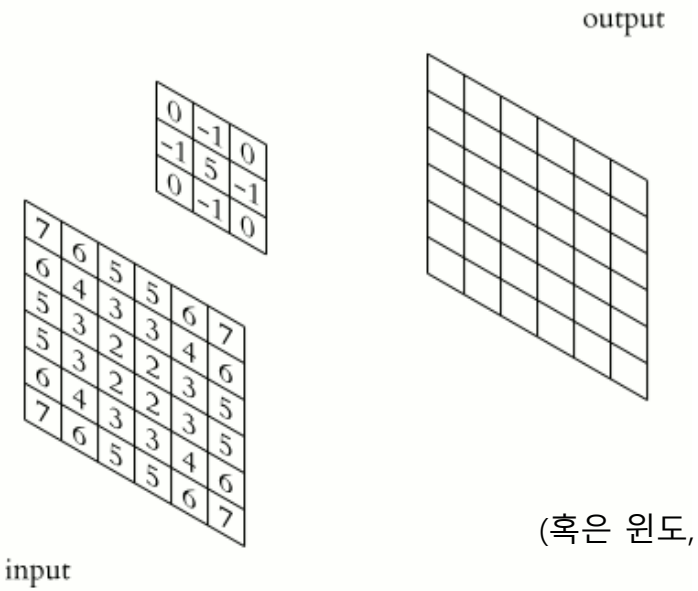
무엇을 해볼까요?

- ROI는 그럼 어떻게 알수있지?? : 물체 검출 알고리즘을 이용
  - YOLO

https://pjreddie.com/darknet/yolo/

무엇을 해볼까요?

- 이미지 Convolution
  - Filter(필터)와 Convolution(컨볼루션)
  - 컨볼루션 연산은 공간 영역 필터링을 위한 핵심 연산 방법
  - 커널을 이용한 콘볼루션 계산을 통해 새로운 정보로 만드는 방법

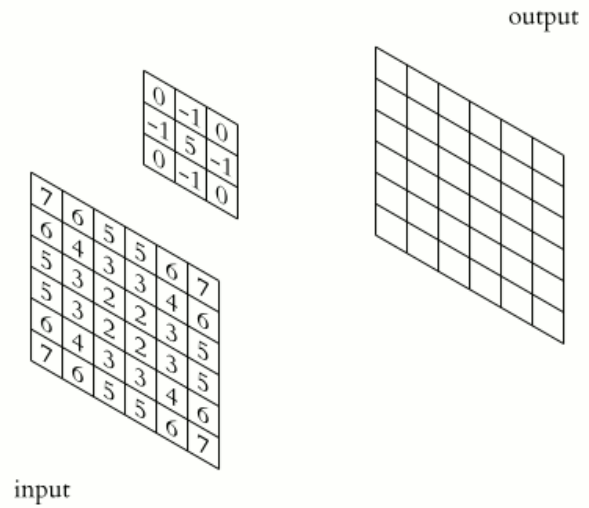


$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

커널(kernel)
(혹은 윈도, 필터, 마스크라고도 부름)

무엇을 해볼까요?

- 이미지 Convolution
  - Filter(필터)와 Convolution(컨볼루션)
  - 평균 필터(Blur)를 적용해 보자



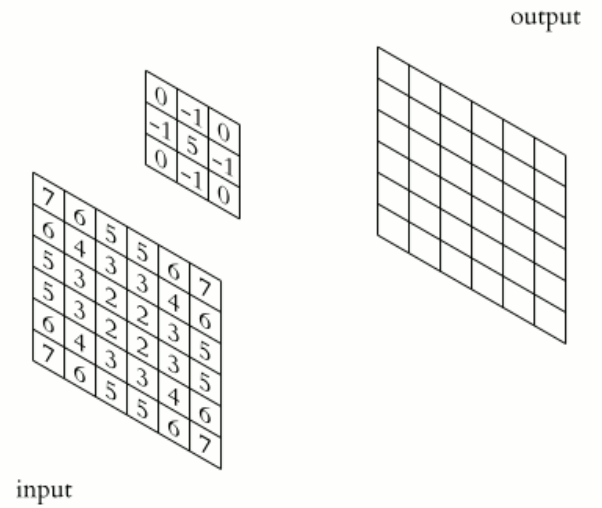
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{25}\times\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

평균 커널(kernel)
(혹은 윈도, 필터, 마스크라고도 부름)

무엇을 해볼까요?

- 이미지 Convolution
  - Filter(필터)와 Convolution(컨볼루션)
  - Edge 강조 필터(Sobel)



$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Edge 강조 커널(kernel)
(혹은 윈도, 필터, 마스크라고도 부름)

무엇을 해볼까요?

- 이미지 Convolution
  - Filter(필터)와 Convolution(컨볼루션)
  - Edge 강조 필터(Sobel)

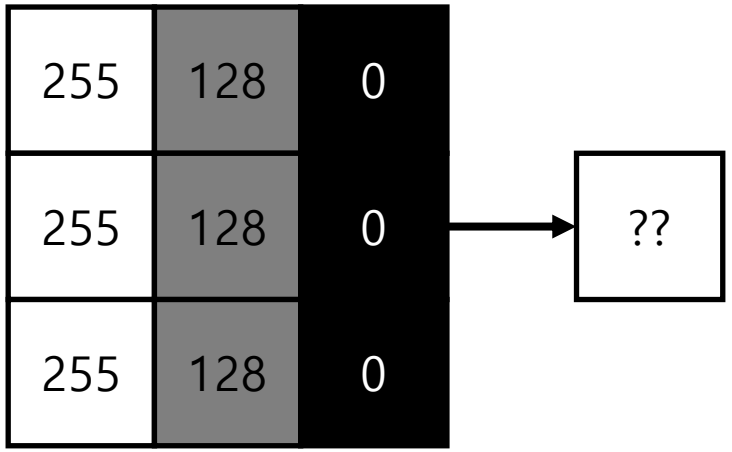
-1	0	+1
-2	0	+2
-1	0	+1

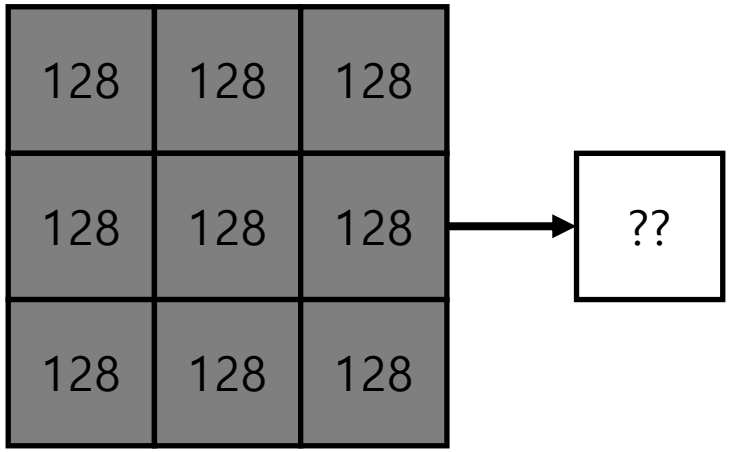
x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

Edge 강조 커널(kernel)
(혹은 윈도, 필터, 마스크라고도 부름)



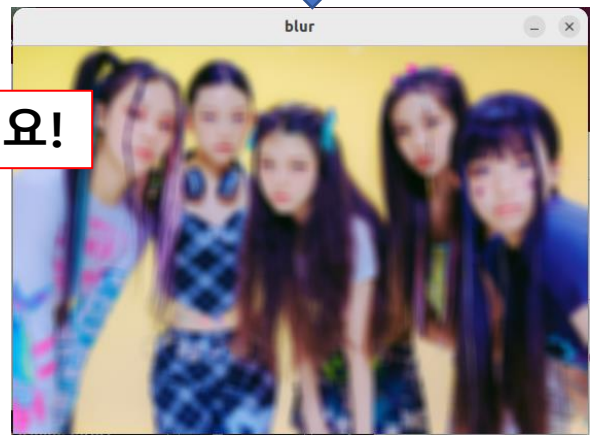


무엇을 해볼까요?

- OpenCV를 이용한 이미지 블러링
  - cv::blur

```
#include "opencv2/opencv.hpp"
#include <stdio.h>

int main(void)
{
        printf("Hello World₩n") ;
        cv::Mat image = cv::imread("/home/dsu/test/test.jpg") ;

        cv::Mat blur_image ;
        cv::blur(image, blur_image, cv::Size(5,5)) ;

        cv::imshow("test", image) ;
        cv::imshow("blur", blur_image) ;

        cv::waitKey(0) ;

        return 0 ;
}
```

커널사이즈 변경



커널사이즈를 변경하며 실험 해보세요!

무엇을 해볼까요?

- OpenCV를 이용한 흑백 이미지 만들기
  - cv::cvtColor

```
#include "opencv2/opencv.hpp"
#include <stdio.h>

int main(void)
{
		printf("Hello World₩n") ;
		cv::Mat image = cv::imread("/home/dsu/test/test.jpg") ;

		cv::Mat gray_image ;
		cv::cvtColor(image, gray_image, cv::COLOR_BGR2GRAY) ;

		cv::imshow("test", image) ;
		cv::imshow("gray", gray_image) ;

		cv::waitKey(0) ;

		return 0 ;
}
```

무엇을 해볼까요?

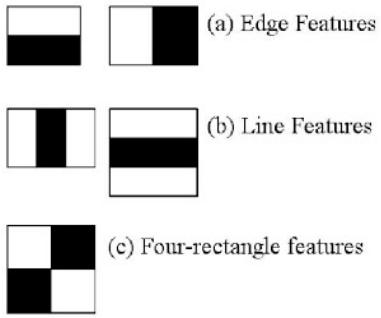
- OpenCV를 이용한 Edge 검출
  - cv::Sobel

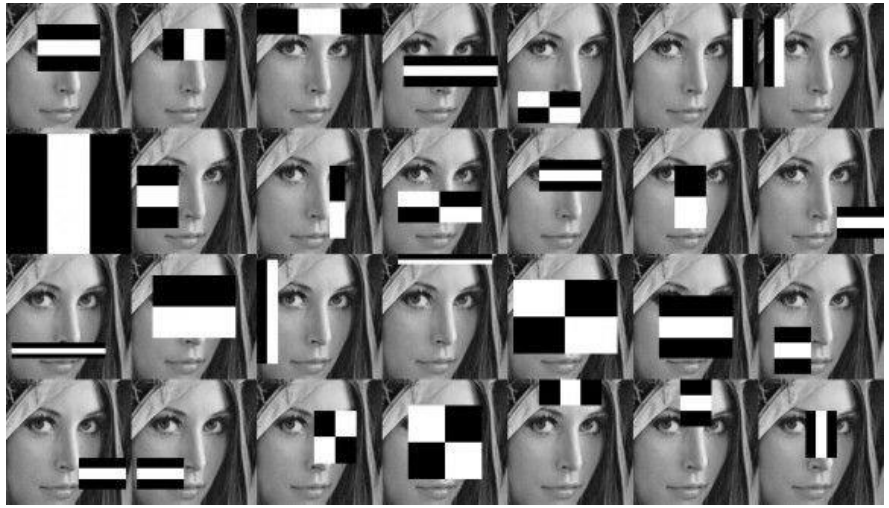
```
int main(void)
{
....
        cv::Mat sobel_x_image ;
        cv::Sobel(gray_image, sobel_x_image, CV_8UC1, 1, 0) ;

        cv::Mat sobel_y_image ;
        cv::Sobel(gray_image, sobel_y_image, CV_8UC1, 0, 1) ;

        cv::imshow("sobel_x_image", sobel_x_image) ;
        cv::imshow("sobel_y_image", sobel_y_image) ;

....
}
```

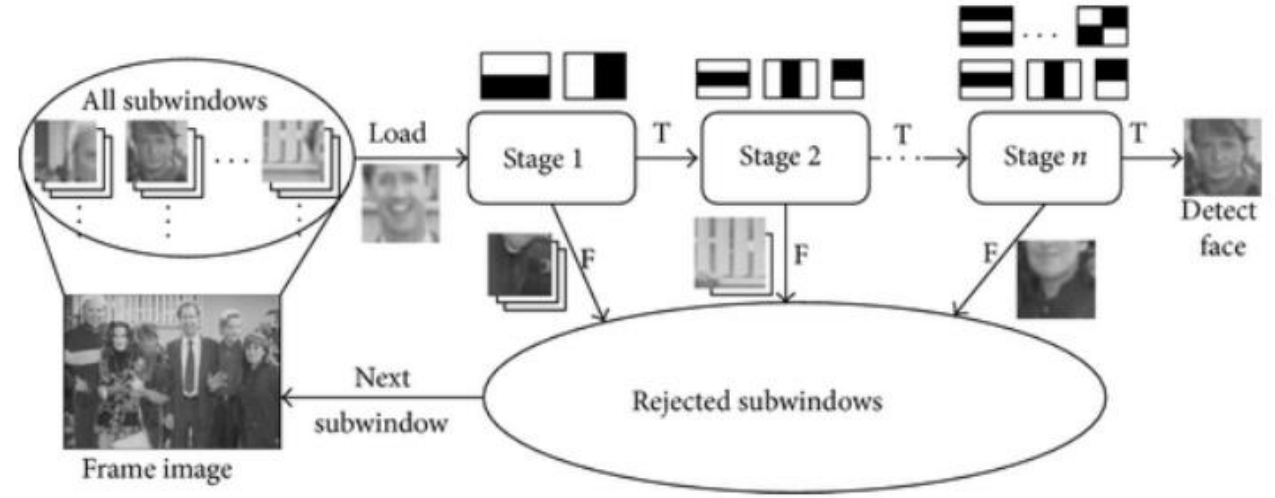
코드의 일부분만 있습니다. 원문은 QR코드 링크에서 확인!

무엇을 해볼까요?

- OpenCV를 이용한 Face Detection

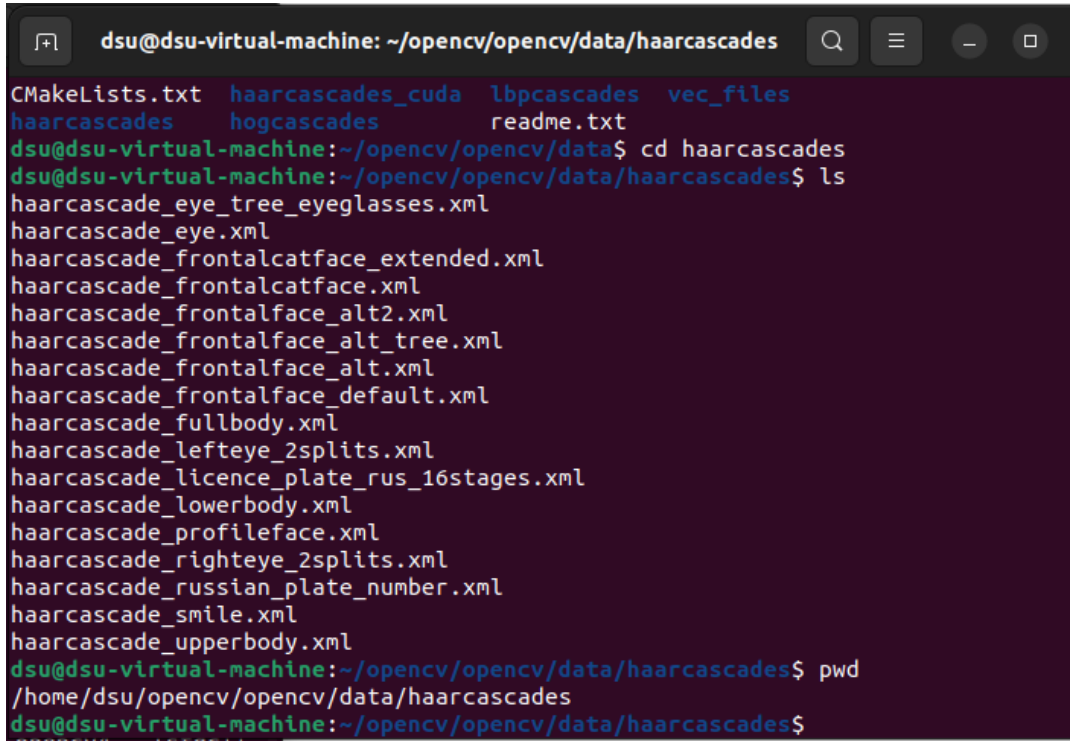






무엇을 해볼까요?

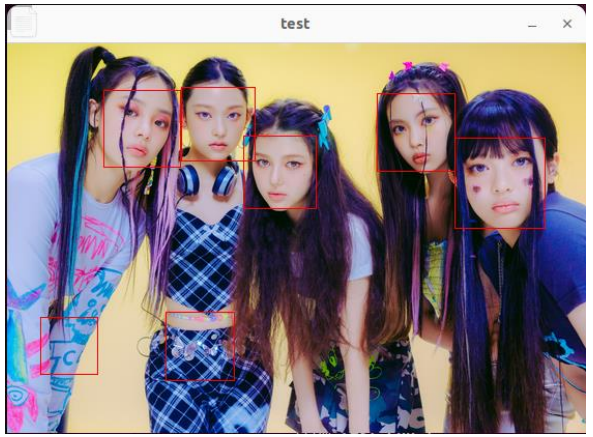
- OpenCV를 이용한 Face Detection
  - 학습 결과물 : /home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml



무엇을 해볼까요?

- OpenCV를 이용한 Face Detection
  - 학습 결과물 : /home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml

```
#include "opencv2/opencv.hpp"
#include <stdio.h>

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/test.jpg") ;

        cv::CascadeClassifier cascade;
        cascade.load( "/home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml" ) ;

        std::vector<cv::Rect> faces;
        cascade.detectMultiScale( image, faces );

        for( int i=0 ; i<faces.size() ; i++ )
        {
            cv::rectangle(image, faces[i], cv::Scalar(0,0,255)) ;
        }

        cv::imshow("test", image) ;

        cv::waitKey(0) ;

        return 0 ;
}
```

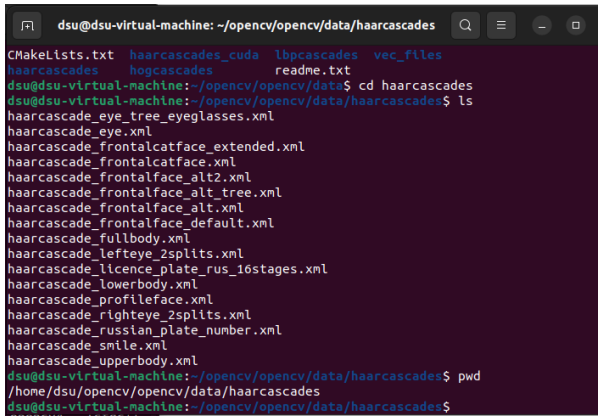


무엇을 해볼까요?

- OpenCV를 이용한 Face Detection
  - 학습 결과물 : /home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml

```
#include "opencv2/opencv.hpp"
#include <stdio.h>

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/test.jpg") ;

        cv::CascadeClassifier cascade;
        cascade.load( "/home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml" ) ;

        std::vector<cv::Rect> faces;
        cascade.detectMultiScale( image, faces );

        for( int i=0 ; i<faces.size() ; i++ )
        {
                cv::rectangle(image, faces[i], cv::Scalar(0,0,255)) ;
        }

        cv::imshow("test", image) ;

        cv::waitKey(0) ;

        return 0 ;
}
```

학습 결과물을 변경하며 실험 해보세요!



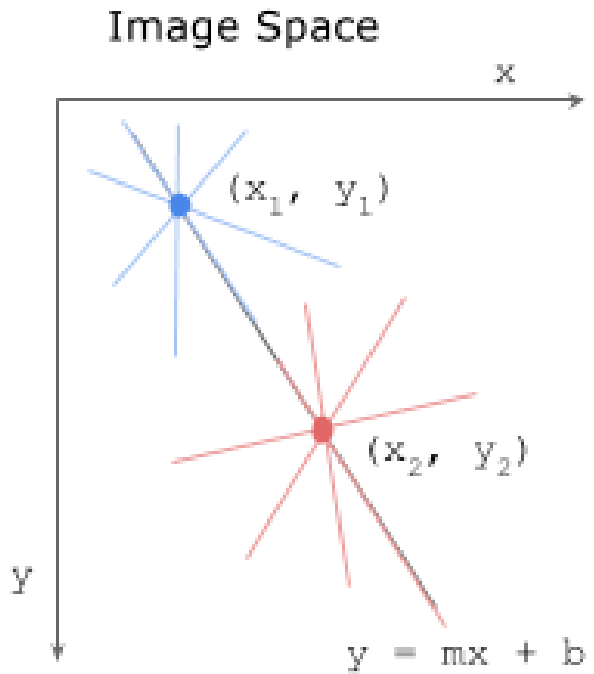
무엇을 해볼까요?

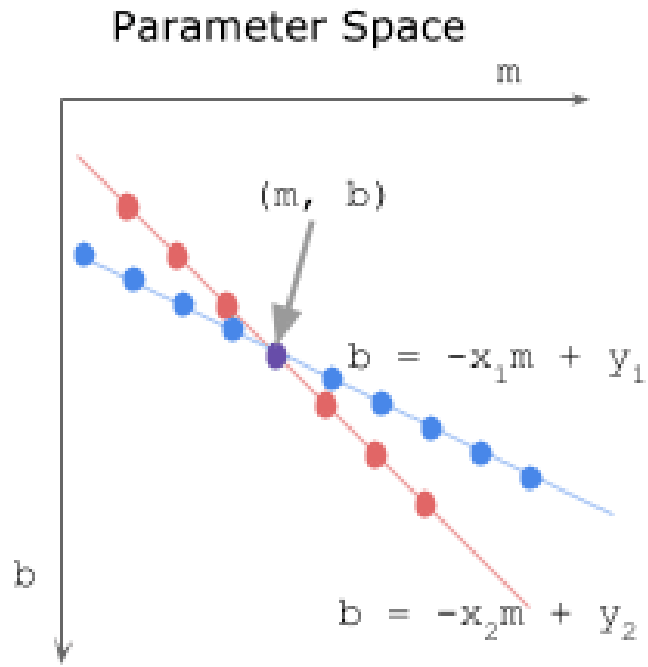
- OpenCV를 이용한 Face Detection + Blur (ROI의 원리)
  - 학습 결과물 : /home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml

```
#include "opencv2/opencv.hpp"
#include <stdio.h>

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/test.jpg") ;

        cv::CascadeClassifier cascade;
        cascade.load( "/home/dsu/opencv/opencv/data/haarcascades/haarcascade_frontalface_default.xml" ) ;

        std::vector<cv::Rect> faces;
        cascade.detectMultiScale( image, faces );

        for( int i=0 ; i<faces.size() ; i++ )
        {
            cv::Mat roi_face_image = image(faces[i]) ;
            cv::blur(roi_face_image, roi_face_image, cv::Size(10,10)) ;
        }

        cv::imshow("test", image) ;
        cv::waitKey(0) ;

        return 0 ;
}
```

무엇을 해볼까요?

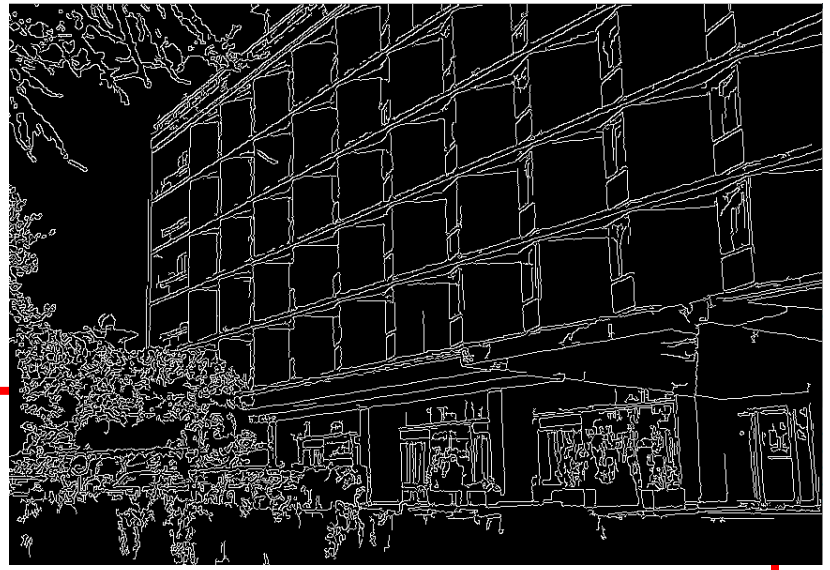
- OpenCV를 이용한 Line 검출 (Hough Transform)

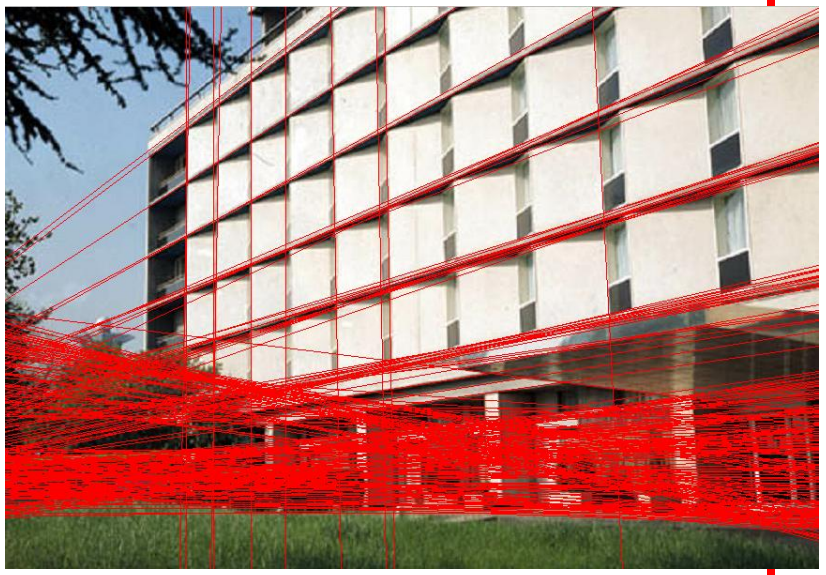




무엇을 해볼까요?

- OpenCV를 이용한 Line 검출
  - cv::HoughLines



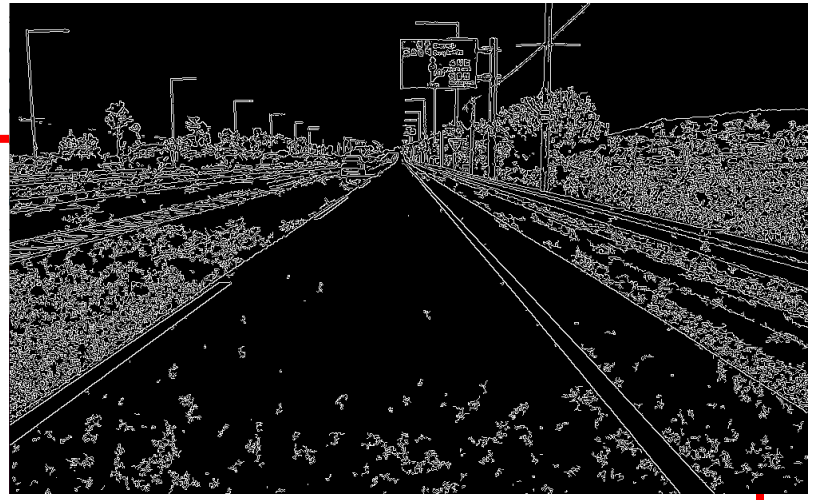


```
#include <math.h>

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/building.jpg");

        cv::Mat gray ;
        cv::cvtColor(image, gray, cv::COLOR_BGR2GRAY);

        // Edge detection
        cv::Mat edge ;
        cv::Canny(image, edge, 50, 200, 3);

        // Standard Hough Line Transform
        std::vector<cv::Vec2f> lines;
        cv::HoughLines(edge, lines, 1, CV_PI/180, 180) ;


  ....
        cv::imshow("edge", edge) ;
        cv::imshow("line", image) ;
        cv::waitKey(0) ;

        return 0 ;
}
```

무엇을 해볼까요?

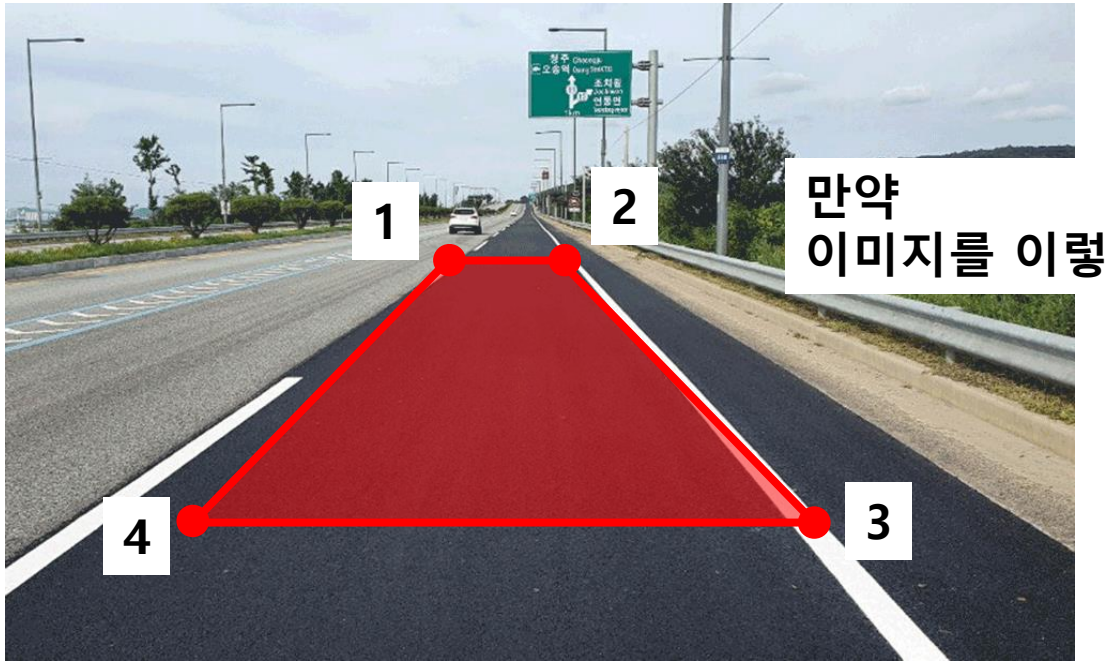
- OpenCV를 이용한 차선(Line) 검출
  - cv::HoughLines

```
#include <math.h>

int main(void)
{
        cv::Mat image = cv::imread("/home/dsu/test/line.png");

        cv::Mat gray ;
        cv::cvtColor(image, gray, cv::COLOR_BGR2GRAY);

        // Edge detection
        cv::Mat edge ;
        cv::Canny(image, edge, 50, 200, 3);

        // Standard Hough Line Transform
        std::vector<cv::Vec2f> lines;
        cv::HoughLines(edge, lines, 1, CV_PI/180, 300) ;


  ....
        cv::imshow("edge", edge) ;
        cv::imshow("line", image) ;
        cv::waitKey(0) ;

        return 0 ;
}
```

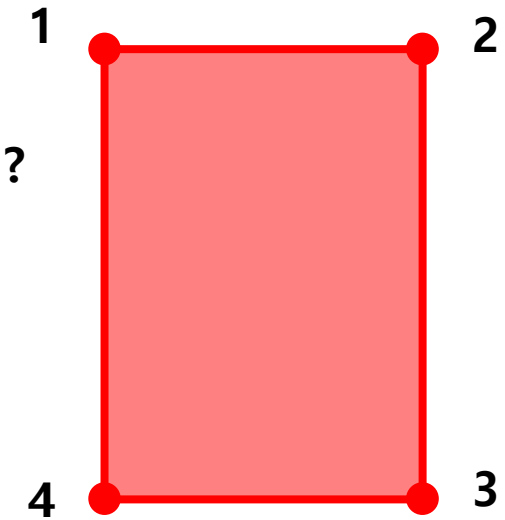


무엇을 해볼까요?

- IPM 이미지 변환을 통한 정확한 차선 인식

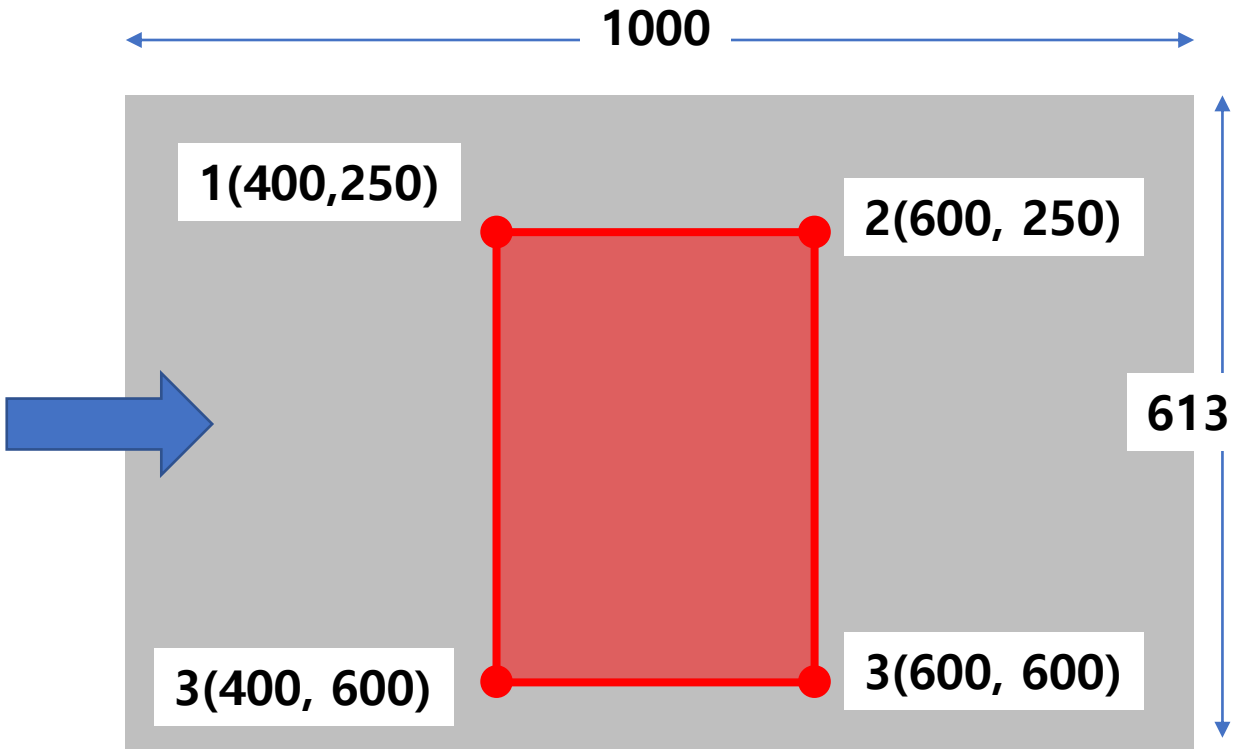


만약
이미지를 이렇게 변형 가능하다면?



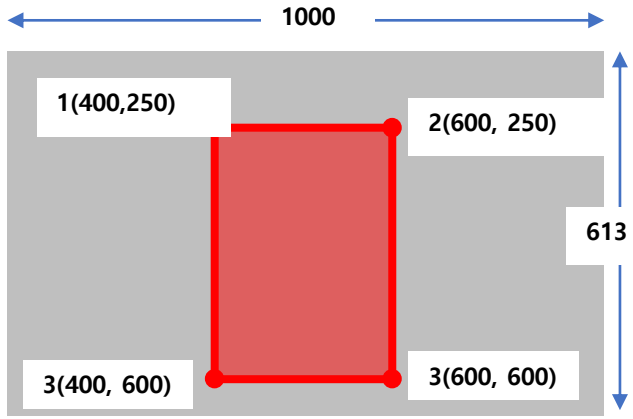
무엇을 해볼까요?

- IPM 이미지 변환을 통한 정확한 차선 인식



무엇을 해볼까요?

- IPM 이미지 변환을 통한 정확한 차선 인식

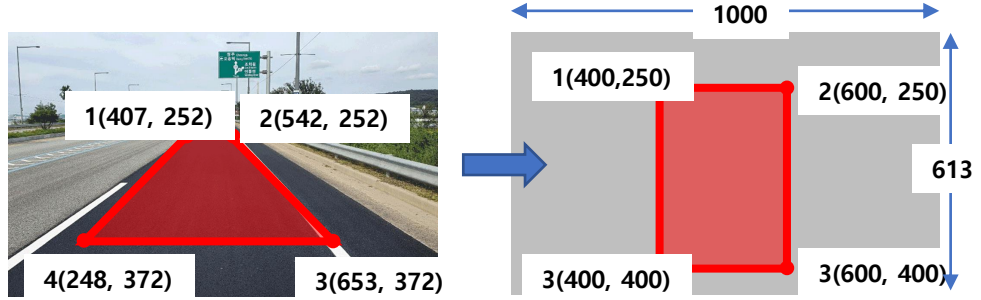



```
cv::Point2f src_points[4] ;
src_points[0] = cv::Point2f(407, 252) ;
src_points[1] = cv::Point2f(542, 252) ;
src_points[2] = cv::Point2f(653, 372) ;
src_points[3] = cv::Point2f(248, 372) ;
```

getPerspectiveTransform

```
cv::Point2f dst_points[4] ;
dst_points[0] = cv::Point2f(400, 250) ;
dst_points[1] = cv::Point2f(600, 250) ;
dst_points[2] = cv::Point2f(600, 600) ;
dst_points[3] = cv::Point2f(400, 600) ;
```

무엇을 해볼까요?



- IPM 이미지 변환을 통한 정확한 차선 인식
  - getPerspectiveTransform

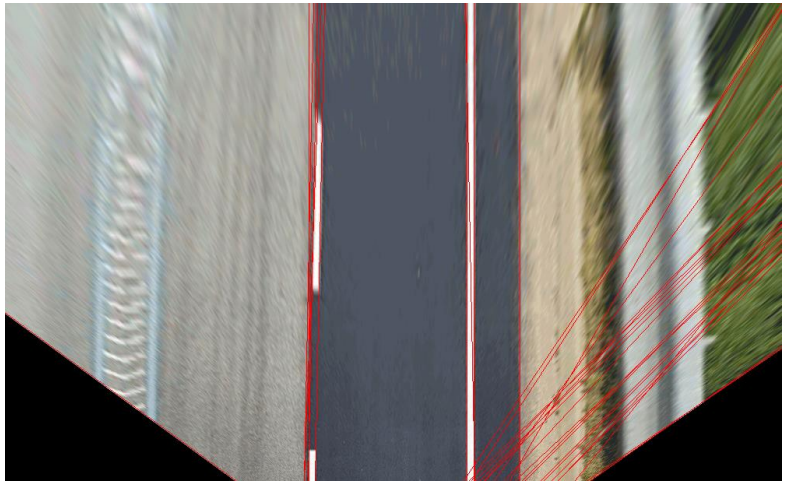
....

```
cv::Point2f src_points[4] ;
src_points[0] = cv::Point2f(407, 252) ;
src_points[1] = cv::Point2f(542, 252) ;
src_points[2] = cv::Point2f(653, 372) ;
src_points[3] = cv::Point2f(248, 372) ;

cv::Point2f dst_points[4] ;
dst_points[0] = cv::Point2f(400, 250) ;
dst_points[1] = cv::Point2f(600, 250) ;
dst_points[2] = cv::Point2f(600, 600) ;
dst_points[3] = cv::Point2f(400, 600) ;

cv::Mat perspective = cv::getPerspectiveTransform(src_points, dst_points) ;

cv::Mat dst ;
cv::warpPerspective(image, dst, perspective, image.size()) ;
```

....

무엇을 해볼까요?

- IPM 이미지 변환을 통한 정확한 차선 인식
  - getPerspectiveTransform + Hough Line Transform



무엇을 해볼까요?

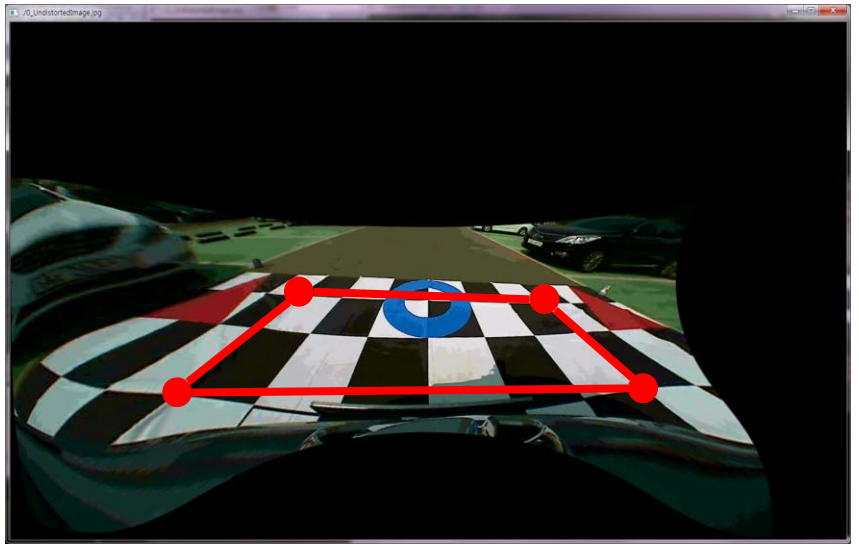
- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰

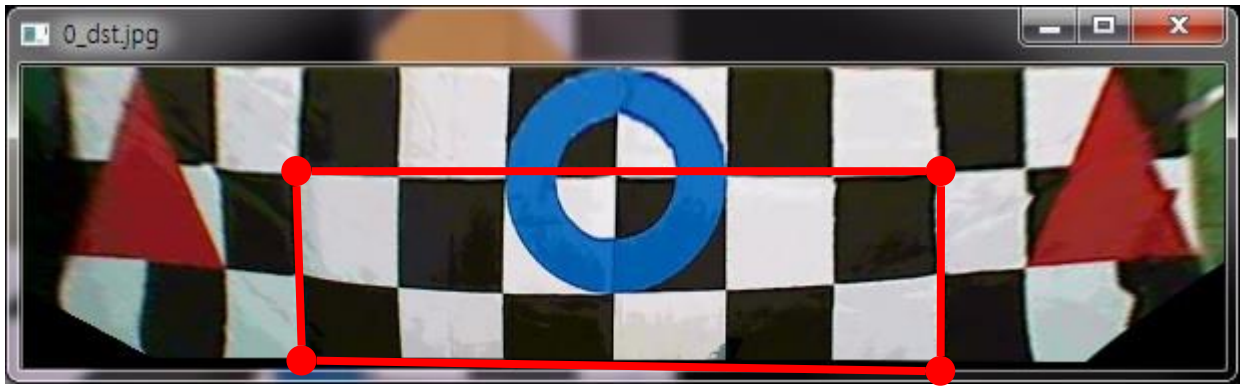
무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰

무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰





무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰

무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰

무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰

무엇을 해볼까요?

- IPM 변환을 이용한 응용 : 4대의 카메라를 이용한 어라운드뷰