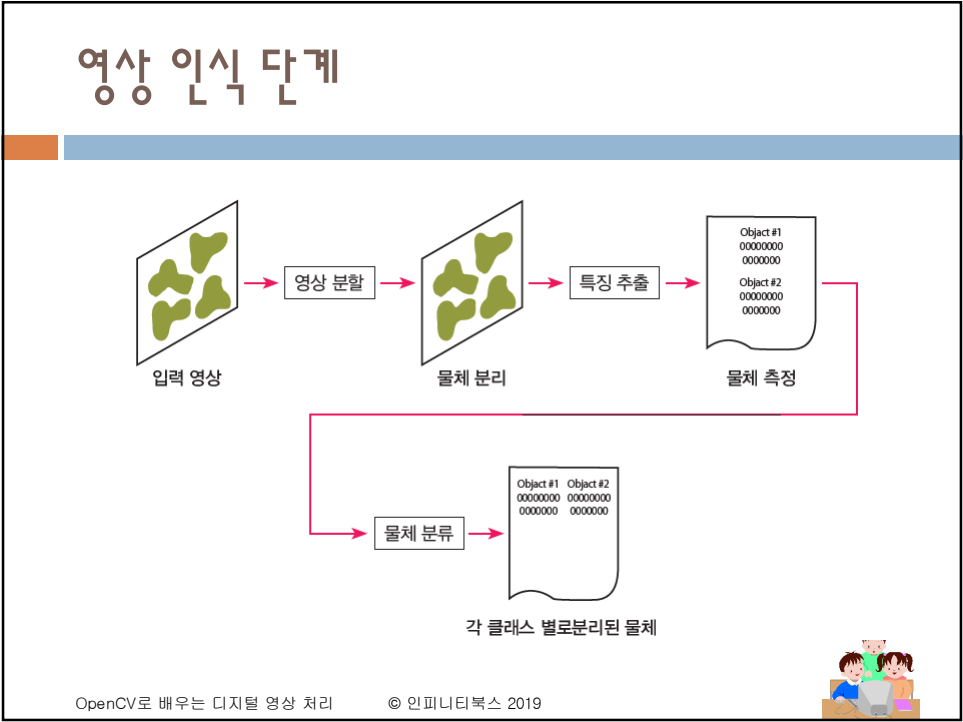


1



2



3

영상 특징

- 에지(edge)
- 직선
- 원
- 코너(corner detection)
- 블로브 검출(blob detection)
- 리지 검출(ridge detection)
- Haar 특징
- HOG(Histogram of Oriented Gradients)
- SIFT(Scale-Invariant Feature Transform)
- SURF(Speeded Up Robust Feature)

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

4

조인
합성

특징
검출



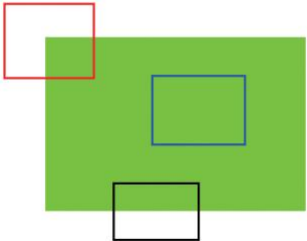
OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019




5

조인
합성

특징
검출

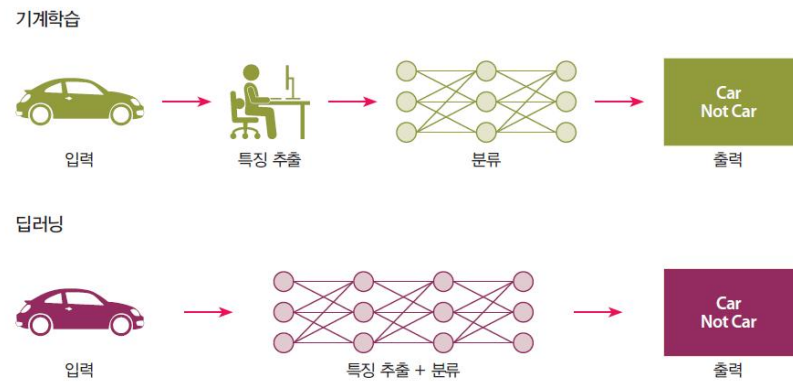


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



6

딥러닝의 특징 추출

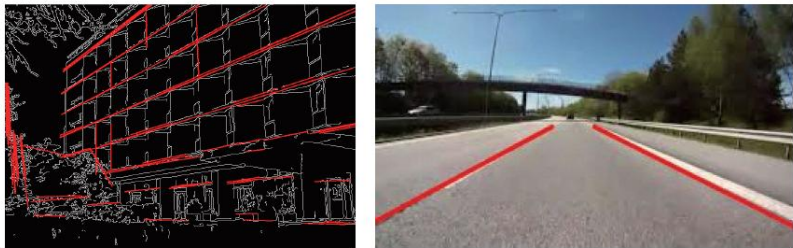


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



7

허프(Hough) 변환



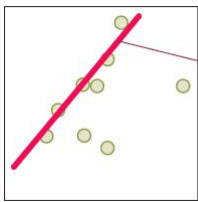
출처: OpenCV 튜토리얼

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



8

에지 검출기를 이용하는 방법의 문제점



에지가 연결되지 않아서 완전한 직선을 얻기는 힘들다.



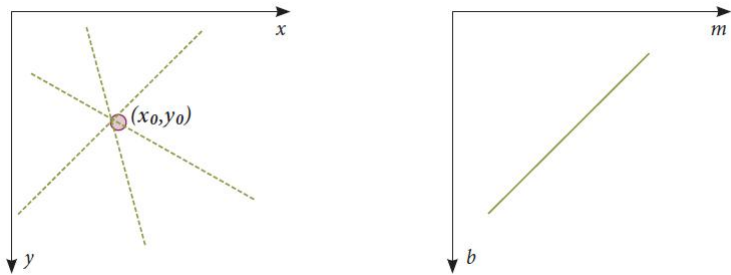
허프 변환

- 허프 변환은 영상 분석이나 컴퓨터 비전에서 아주 많이 사용되는 특징 추출 기법이다.
- 허프 변환은 1962년에 P. V. C. Hough에 의해 소개되었고 Duda와 Hart가 이 아이디어를 이용하여 직선 검출




허프 변환

$$y = mx + b$$



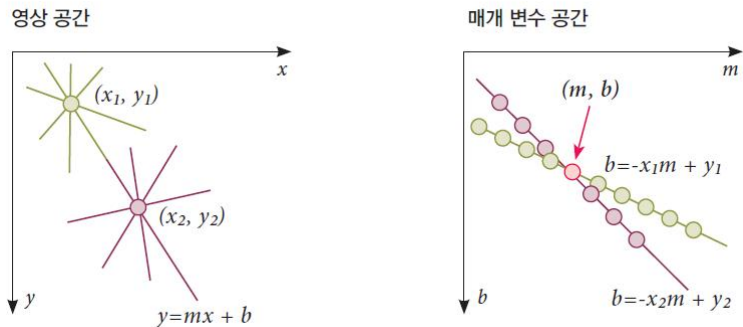
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019




11

허프 변환

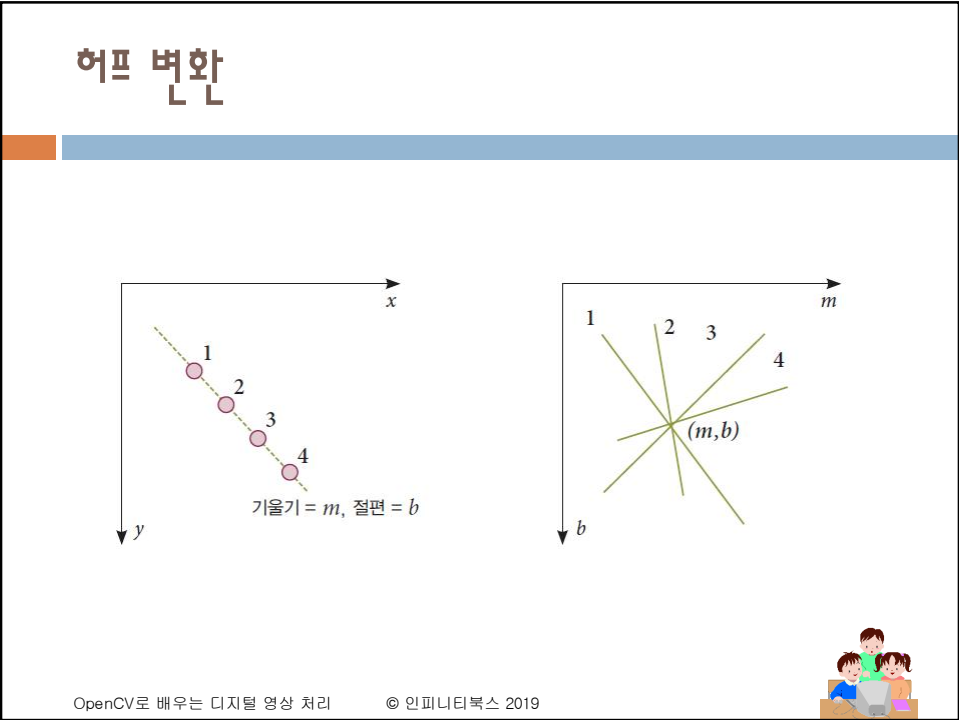


OpenCV로 배우는 디지털 영상 처리

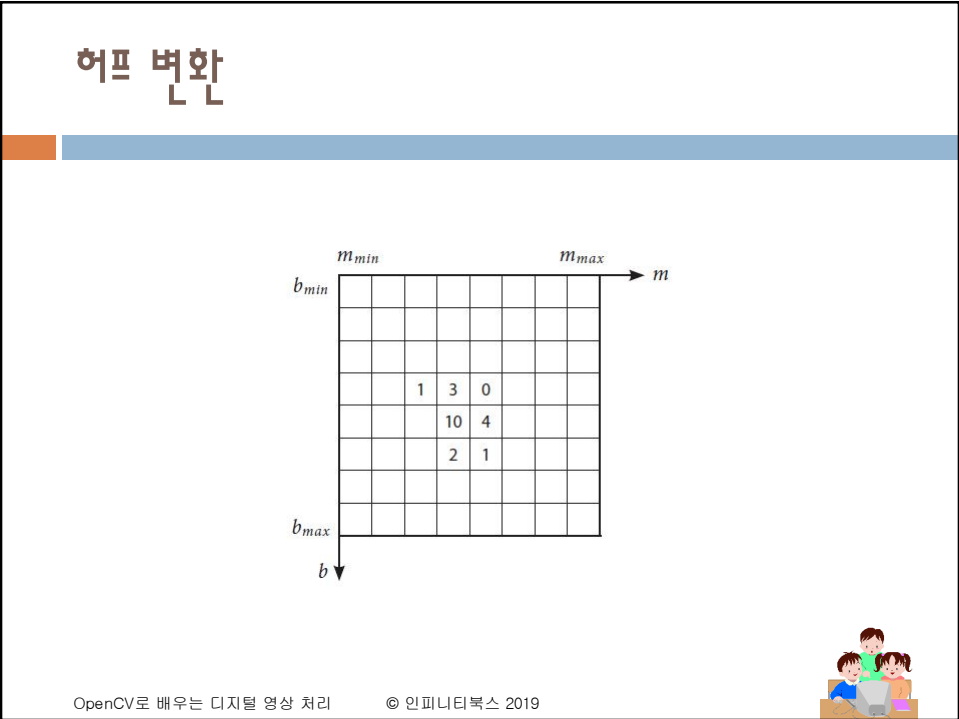
© 인피니티북스 2019



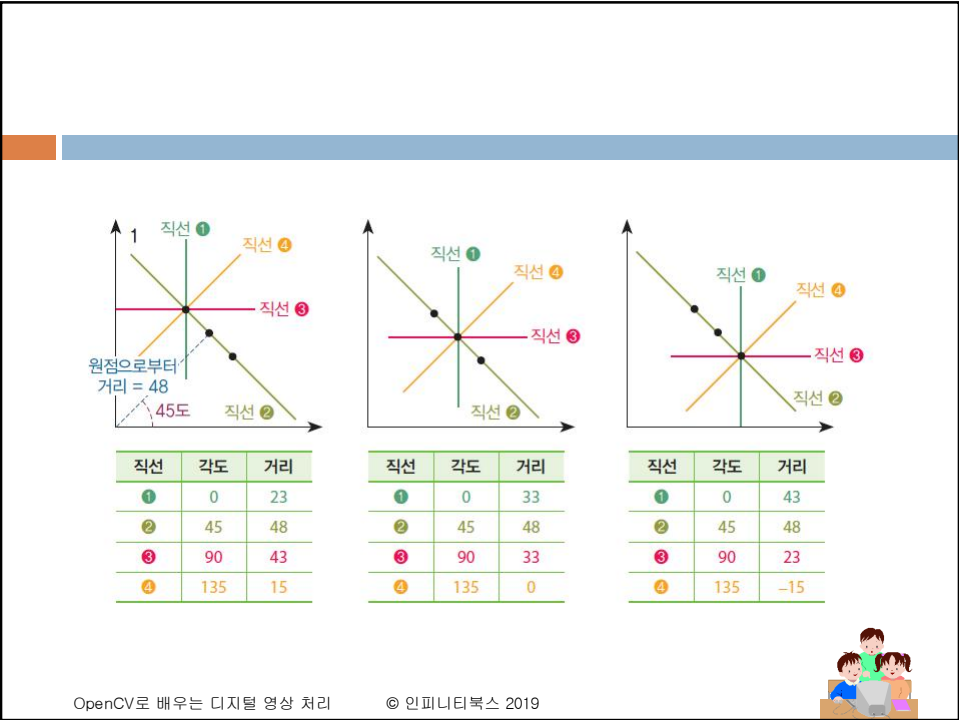
12



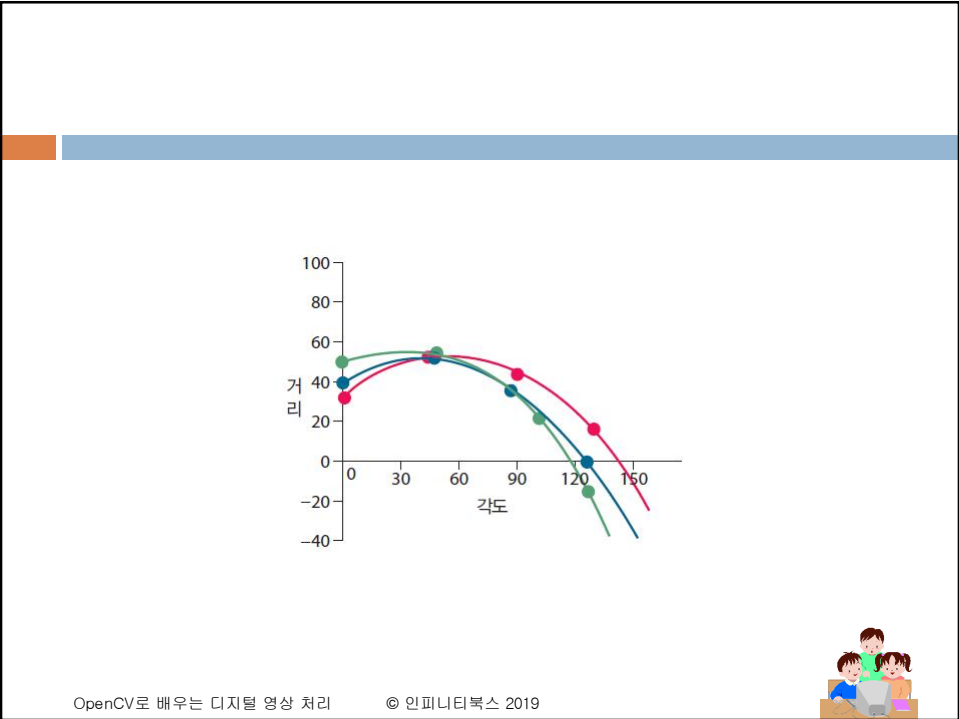
13



14

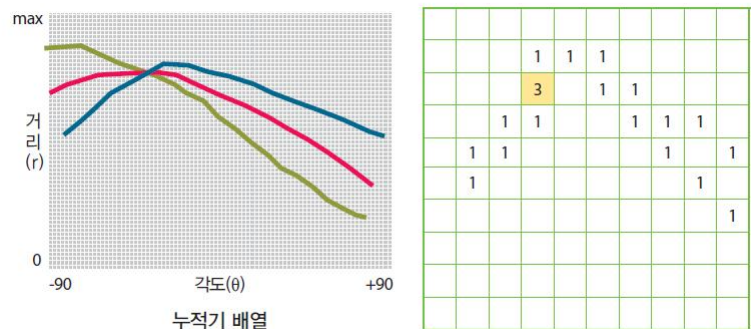


17



18

누적기 아이디어



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



19

OpenCV에서의 허프 변환

```
void HoughLines(InputArray image, OutputArray lines, double rho,
double theta, int threshold)
```

매개 변수	설명
image	에지 검출기의 출력이다. 그레이스케일 영상이어야 한다.
lines	검출된 직선의 매개변수를 저장하는 (θ, r) 형태의 벡터이다.
rho	매개변수 r 의 해상도를 화소로 나타낸 것이다. 우리는 1을 사용한다.
theta	매개변수 θ 의 해상도를 나타낸 것이다. 우리는 1도를 사용한다(CV_PI/180).
threshold	직선을 검출하는데 필요한 교차점의 최소 개수이다.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



20

OpenCV에서의 허프 변환

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

매개 변수	설명
image	에지 검출기의 출력이다. 그레이스케일 영상이어야 한다.
lines	검출된 직선의 매개변수를 저장하는 (x_0, y_0, x_1, y_1) 형태의 벡터이다.
rho	매개변수 r 의 해상도를 화소로 나타낸 것이다. 우리는 1을 사용한다.
theta	매개변수 θ 의 해상도를 나타낸 것이다. 우리는 1도를 사용한다($CV_PI/180$).
threshold	직선을 검출하는데 필요한 교차점의 최소 개수이다.
minLineLength	직선의 최소 길이
maxLineGap	하나의 직선으로 간주되는 점들 사이의 최대 거리



```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main()
{
    Mat src = imread("d:/building.jpg", 0);
    if (src.empty()) { cout << "can not open " << endl; return -1; }

    Mat dst, cdst;
    Canny(src, dst, 100, 200);
    imshow("edge", dst);
    cvtColor(dst, cdst, CV_GRAY2BGR);
```



```

vector<Vec4i> lines;
HoughLinesP(dst, lines, 1, CV_PI / 180, 50, 100, 20);
for (size_t i = 0; i < lines.size(); i++) {
    Vec4i l = lines[i];
    line(cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 3,
CV_AA);
}

imshow("source", src);
imshow("detected lines", cdst);
waitKey();
return 0;
}

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



23

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



24

실행 결과







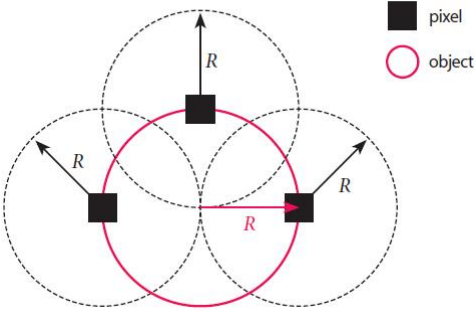
© 2019




25

원형 허프 변환

$$x = a + R \cos (\theta)$$
$$y = b + R \sin (\theta)$$

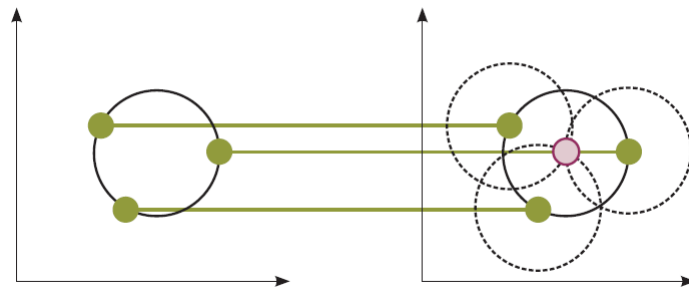


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



26

원형 허프 변환



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



27

알고리즘

1. 모든 $A[a, b, r]$ 을 0으로 초기화한다.
2. 가우시안 블러링을 수행한다.
3. 캐니 에지 연산자를 이용하여 에지를 추출한다.
4. 모든 에지에 대하여 누적기 배열에 투표를 한다.
5. 누적기 값 중에서 최대값을 계산한다.

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



28

알고리즘

```

for each pixel(x,y)
  for each radius r = 10 to r = 60 // 가능한 반지름 값
    for each theta t = 0 to 360 // 가능한 세타 값(0부터 360)
      a = x - r * cos(t * PI / 180); // 원의 중심 좌표 계산
      b = y - r * sin(t * PI / 180); // 원의 중심 좌표 계산
      A[a, b, r] +=1; // 투표, 누적기 배열을 증가시킨다.
    end
  end
end

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



29

OpenCV 함수

```

void HoughCircles(InputArray image, OutputArray circles, int method,
double dp, double minDist, double param1=100, double param2=100,
int minRadius=0, int maxRadius=0 )

```

매개 변수	설명
image	입력 영상(그레이스케일)
circles	3개의 요소를 저장하는 벡터
method	검지 방법을 정의하는 상수
dp	영상 해상도와 검지기 해상도의 비율의 역수. 만약 dp=1이면 검지기와 영상의 해상도는 같다.
minDist	검지된 원 중심 간의 최소 거리
param1	캐니 에지 연산자의 상위 임계값
param2	원 검지의 임계값. 이 값이 적으면 더 많은 원이 검지된다.
minRadius	최소 원 반지름
maxRadius	최대 원 반지름

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



30

```

int main()
{
    Mat src, src_gray;

    src = imread("d:/plates.jpg", 1);
    imshow("src", src);
    // 그레이스케일로 변환한다.
    cvtColor(src, src_gray, CV_BGR2GRAY);

    // 가우시안 블러링 적용
    GaussianBlur(src_gray, src_gray, Size(9, 9), 2, 2);

    vector<Vec3f> circles;

    // 원을 검출하는 허프 변환
    HoughCircles(src_gray, circles, CV_HOUGH_GRADIENT, 1, src_gray.rows /
8, 200, 50, 0, 0);

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



31

```

// 원을 영상 위에 그린다.
for (size_t i = 0; i < circles.size(); i++) {
    Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
    int radius = cvRound(circles[i][2]);
    circle(src, center, 3, Scalar(0, 255, 0), -1, 8, 0); // 원의 중심을
그린다.
    circle(src, center, radius, Scalar(0, 0, 255), 3, 8, 0); // 원을 그린다.
}

imshow("Hough Circle Transform", src);
waitKey(0);
return 0;
}

```

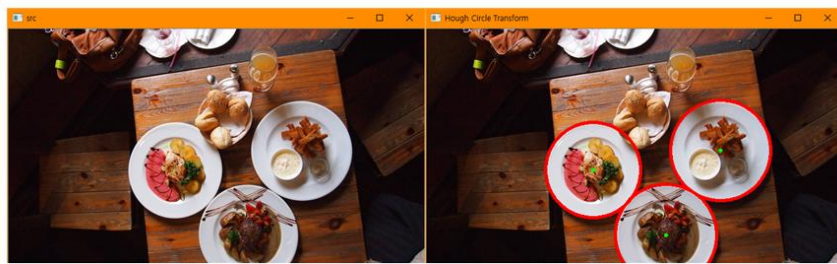
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



32

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



33

코너 검출

- 해리스 코너 검출은 각종 코너 검출 방법 중에서도 고전적인 방법이다.

$$E(u, v) = \sum_{x, y} \omega(x, y) [I(x + u, y + v) - I(x, y)]^2$$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x, y} \omega(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



34

코너 감지

$$R = \det(M) - k[\text{trace}(M)]^2$$

$$\det(M) = \lambda_1 \lambda_2$$
$$\text{trace}(M) = \lambda_1 + \lambda_2$$

Edges $\lambda_2 \gg \lambda_1$

Corners $\lambda_1 \sim \lambda_2$

Flat

Edges $\lambda_1 \gg \lambda_2$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019

35

OpenCV 함수

```
void cornerHarris(InputArray src, OutputArray dst, int blockSize, int ksize, double k)
```

매개 변수	설명
src	입력 영상. 그레이스케일 및 float32 유형이어야 한다.
dst	출력 영상. CV_32FC1 타입이다.
blockSize	코너 감지에 사용되는 커널의 크기
ksize	소벨 연산자를 계산하는데 사용되는 커널 크기
k	$R = \det(M) - k[\text{trace}(M)]^2$ 수식에 사용되는 k값

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019

36

```

int main()
{
    Mat src, gray;
    int thresh = 150;
    int blockSize = 2;
    int apertureSize = 3;
    double k = 0.04;

    src = imread("d:/chessboard.jpg", 1);
    cvtColor(src, gray, CV_BGR2GRAY);
    imshow("src", src);

    Mat dst, dst_norm, dst_norm_scaled;
    dst = Mat::zeros(src.size(), CV_32FC1);

    cornerHarris(gray, dst, blockSize, apertureSize, k);

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



37

```

normalize(dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat());
convertScaleAbs(dst_norm, dst_norm_scaled);

for (int j = 0; j < dst_norm.rows; j++) {
    for (int i = 0; i < dst_norm.cols; i++) {
        if ((int)dst_norm.at<float>(j, i) > thresh) {
            circle(src, Point(i, j), 5, Scalar(0, 0, 255), 2, 8,
0);
        }
    }
}

imshow("corners_window", src);
waitKey(0);
return(0);
}

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



38

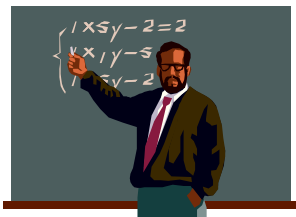
실행 결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



Q & A



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

