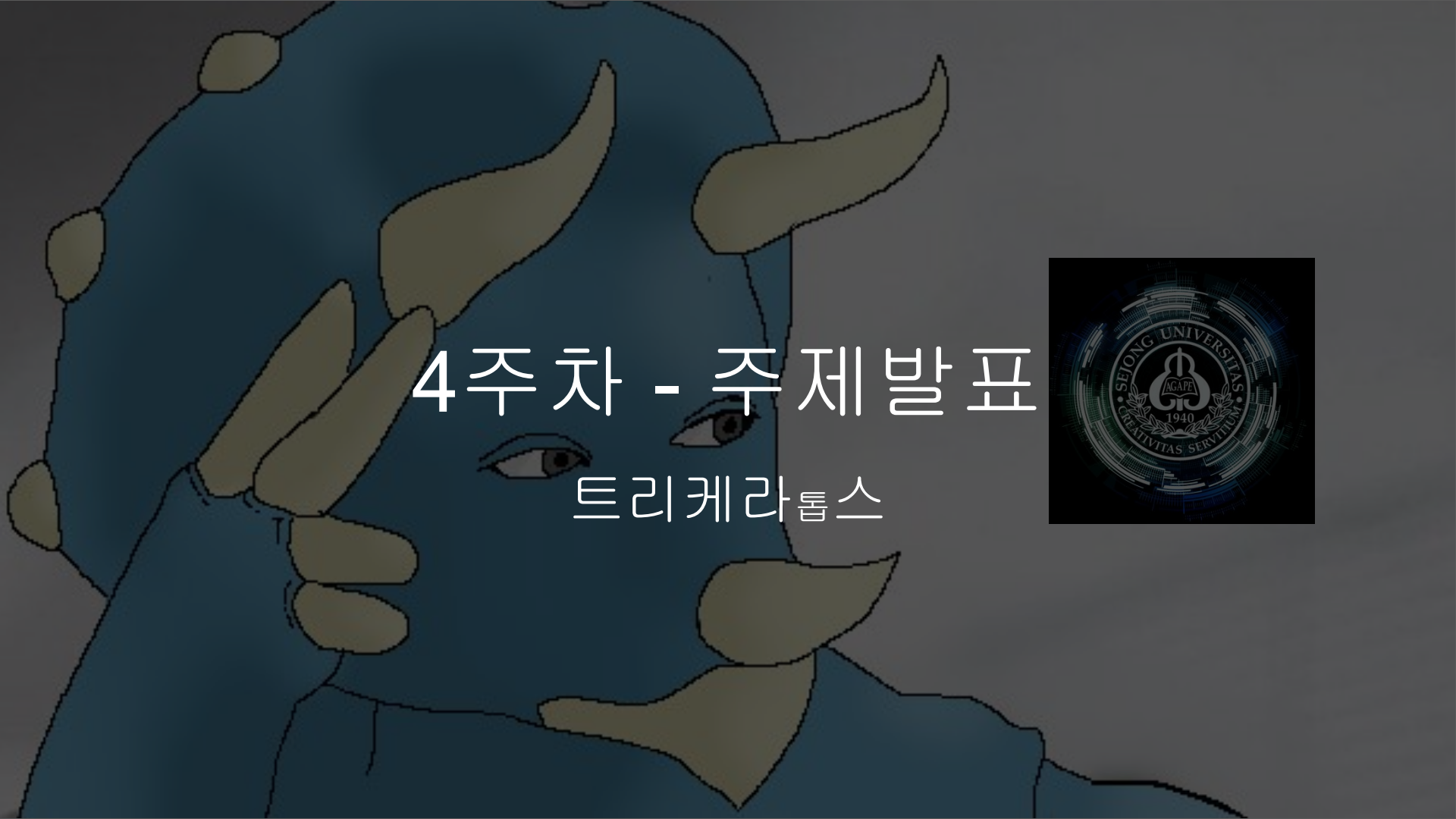


4주차 - 주제발표

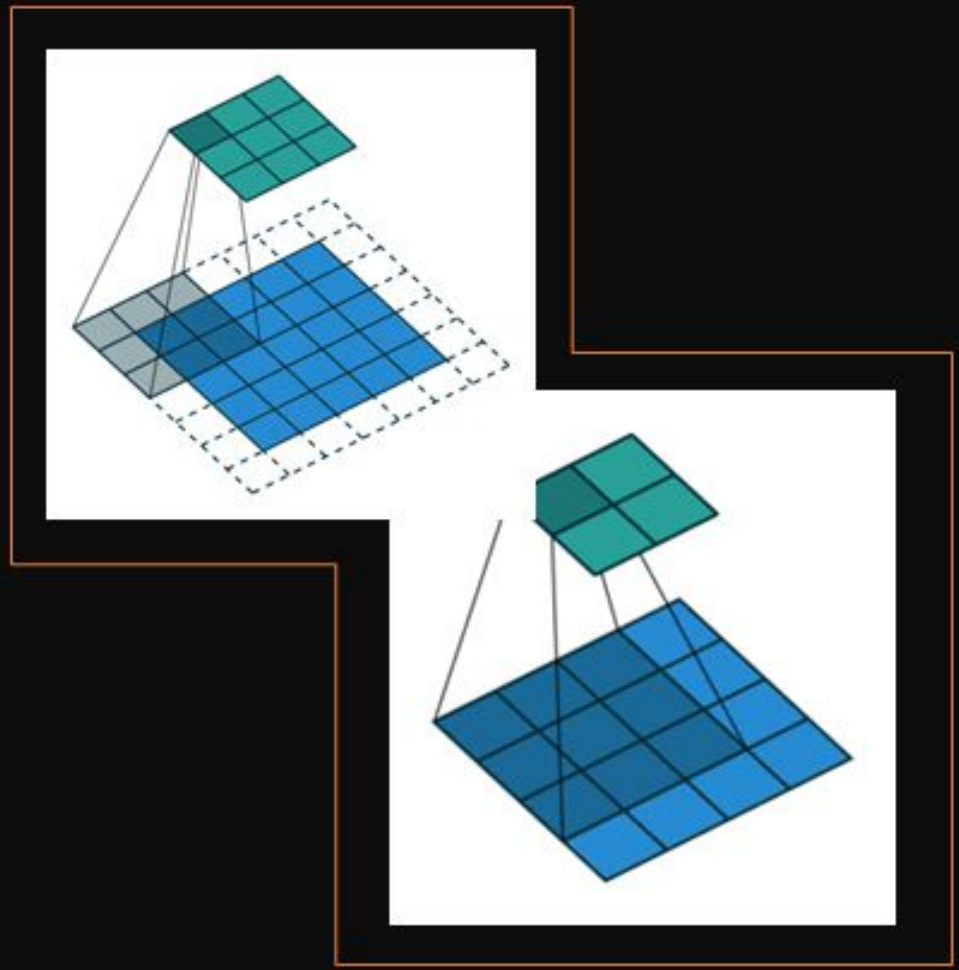
트리케라톱스

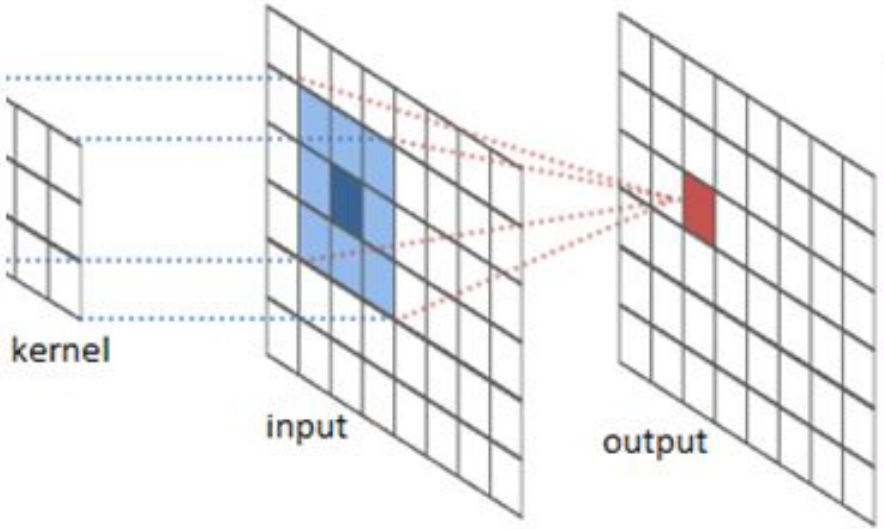
INDEX

- Conv개념 + Filter 간단하게 [현영]
- Sobel Filter + Gradient orientation and Gradient magnitude [해리]
- 최소자승법 (least square method) [윤지]
- RANSAC (최소자승법의 한계에서 스무스하게 이어지면 좋을 듯) [현동]
- 전체적으로 부가설명 및 코드 리뷰 [영채] + 해리..(약간)

INDEX

About Convolution and Filter





About Convolution and Filter

CNN(Convolutional Neural Networks)

CNN?

-데이터에서 직접 학습하고 패턴을 사용해 이미지를 분류

-자율주행자동차, 얼굴인식과 같은 객체인식이나 computer vision이 필요한 분야에 많이 사용.

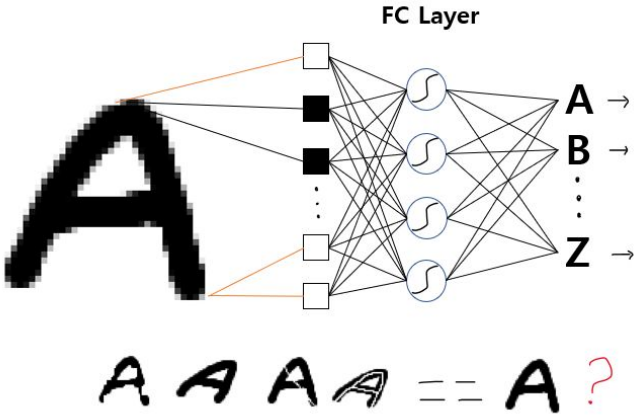
CNN이 유용한 이유?

CNN이전 -> **Fully Connected Layer**

-이미지의 형상 고려 x

-이미지가 회전하거나 움직이면 새로운 입력으로 데이터를 처리

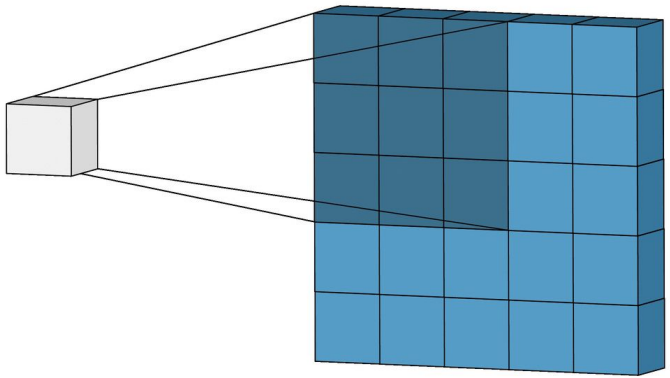
-이미지의 특성을 이해하지 못하고 단순 1D데이터로 보고 학습



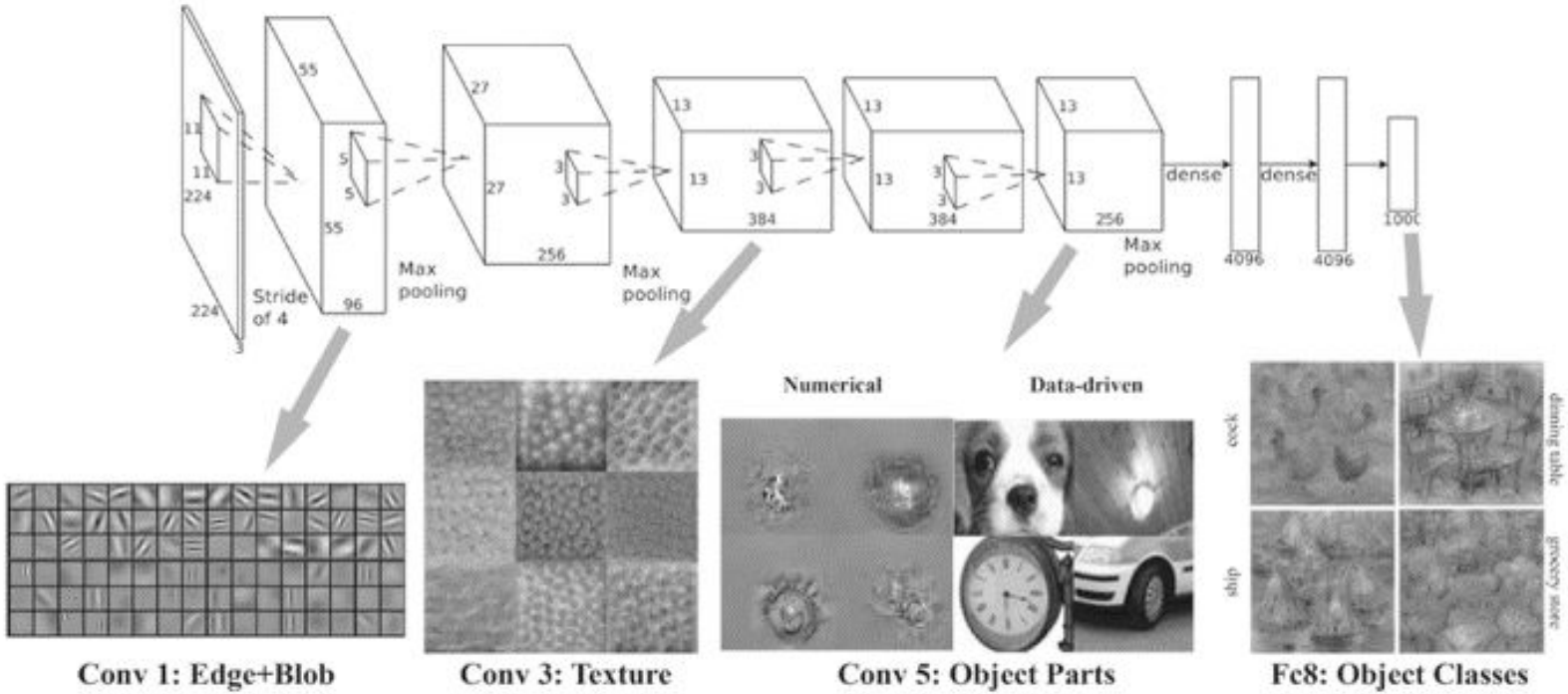
CNN(Convolutional Neural Networks)

FC의 단점을 보완하여

*“ 이미지의 공간정보를 유지한 채 학습 “*하게 하는 모델

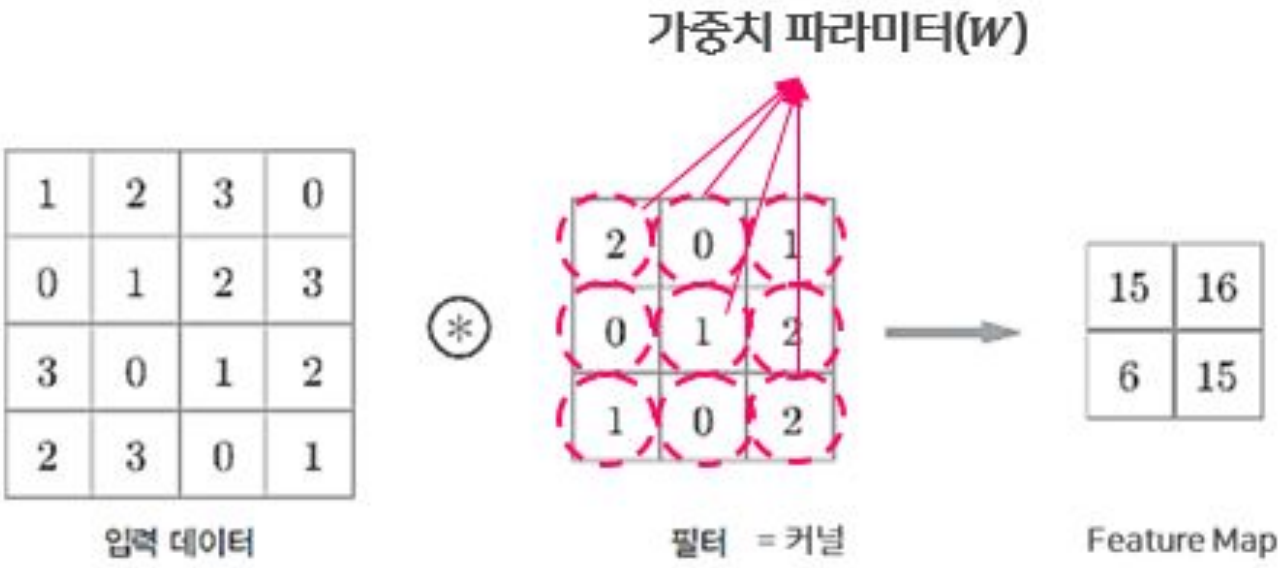


CNN(Convolutional Neural Networks)

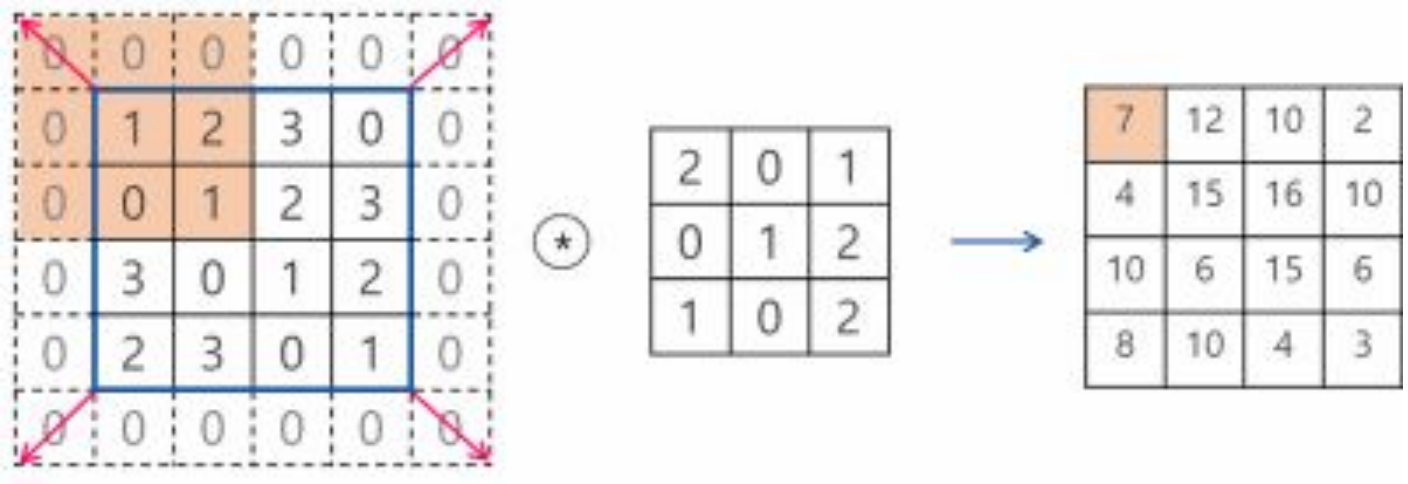


Filter

“수용영역(receptive field) = 필터(filter) = 커널(kernel)”



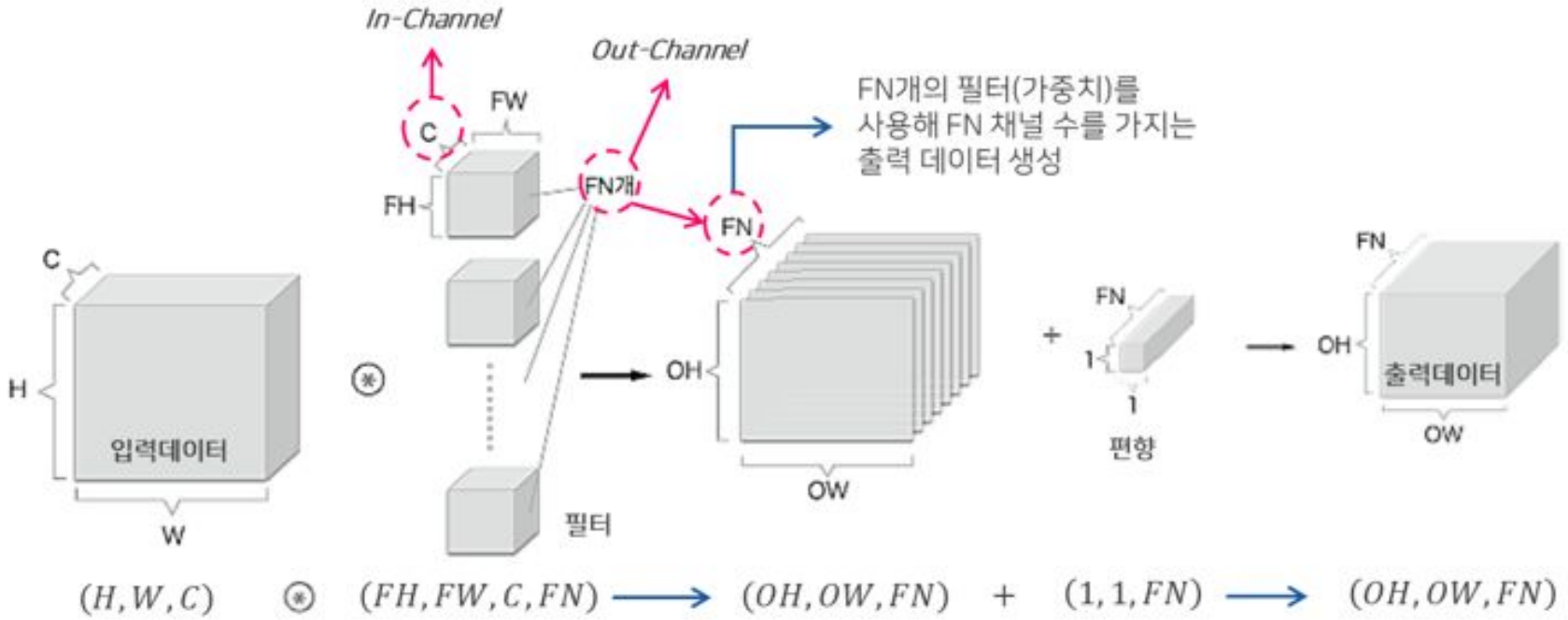
Padding & Stride



3차원 데이터의 합성곱

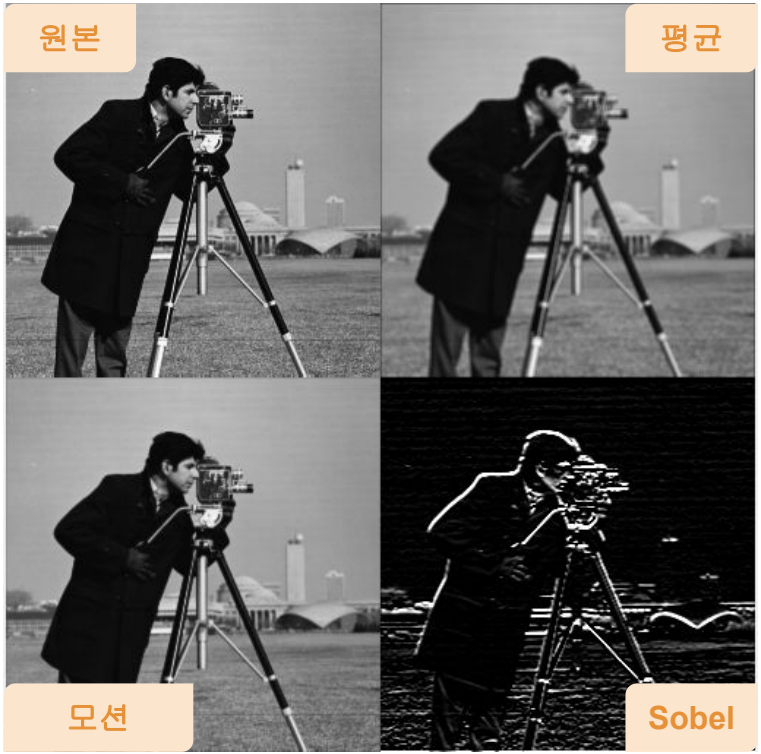
🚨주의!!🚨

입력 데이터 채널 수 = 필터의 채널 수



About Sobel Filter

Sobel Filter



특정한 목적을 위해
특수하게 설계된 필터들이 몇몇 있는데,
(average, gaussian, prewitt, motion, disk, 등등…)

Sobel 도 이런 특수한 필터 중 하나다!

Sobel의 목적은 edge 검출

Introduction to Sobel Filter

X – Direction Kernel

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

Y – Direction Kernel

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

Introduction to Sobel Filter

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

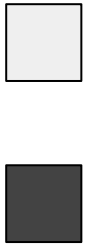
X – Direction Kernel

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

10 + 20 + 10
= 40

-10 -20 -10 +10 +20 +10
= 0

filter



왜 Sobel Filter는 저렇게 생겼을까?

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$\frac{f(x+1) - f(x)}{\frac{f(x+1) - f(x-1)}{2}}$$

$$\begin{bmatrix} f(x-1) & f(x) & f(x+1) \end{bmatrix} \times \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = f(x+1) - f(x-1)$$

edge가 급격히 변하는 것을 막기 위한 **smoothing** 처리

-1
0
1

X

1	1	1

=

Prewitt filter

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

-1
0
1

X

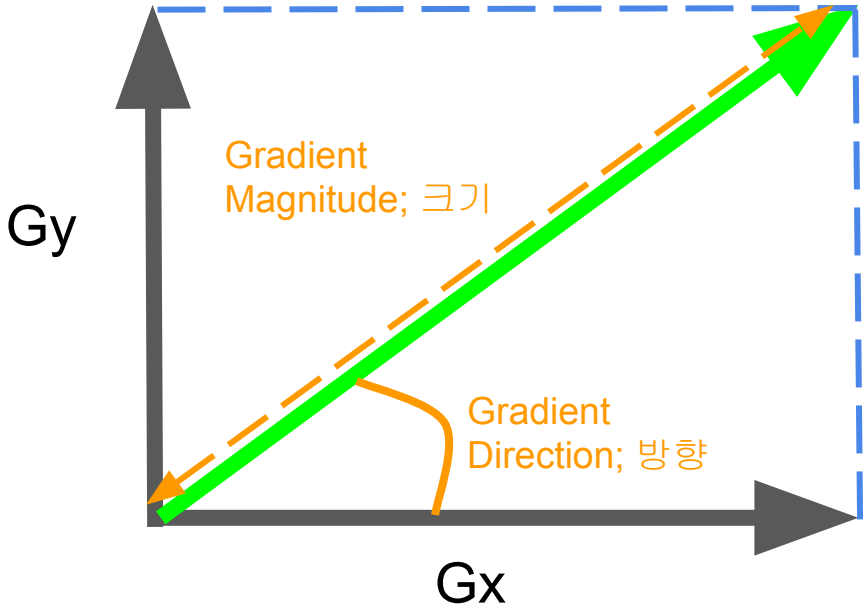
1	2	1

=

Sobel filter

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Image Gradient



At each pixel in the image, the gradient approximations given by G_x and G_y are combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2}$$

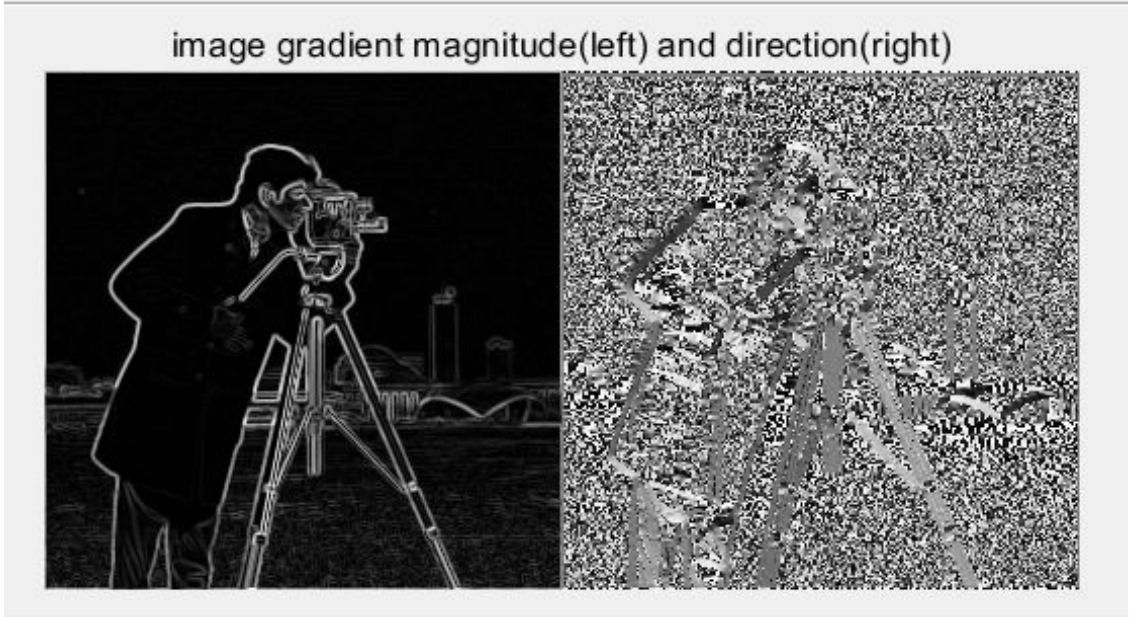
Typically, an approximate magnitude

$$|G| = |Gx| + |Gy|$$

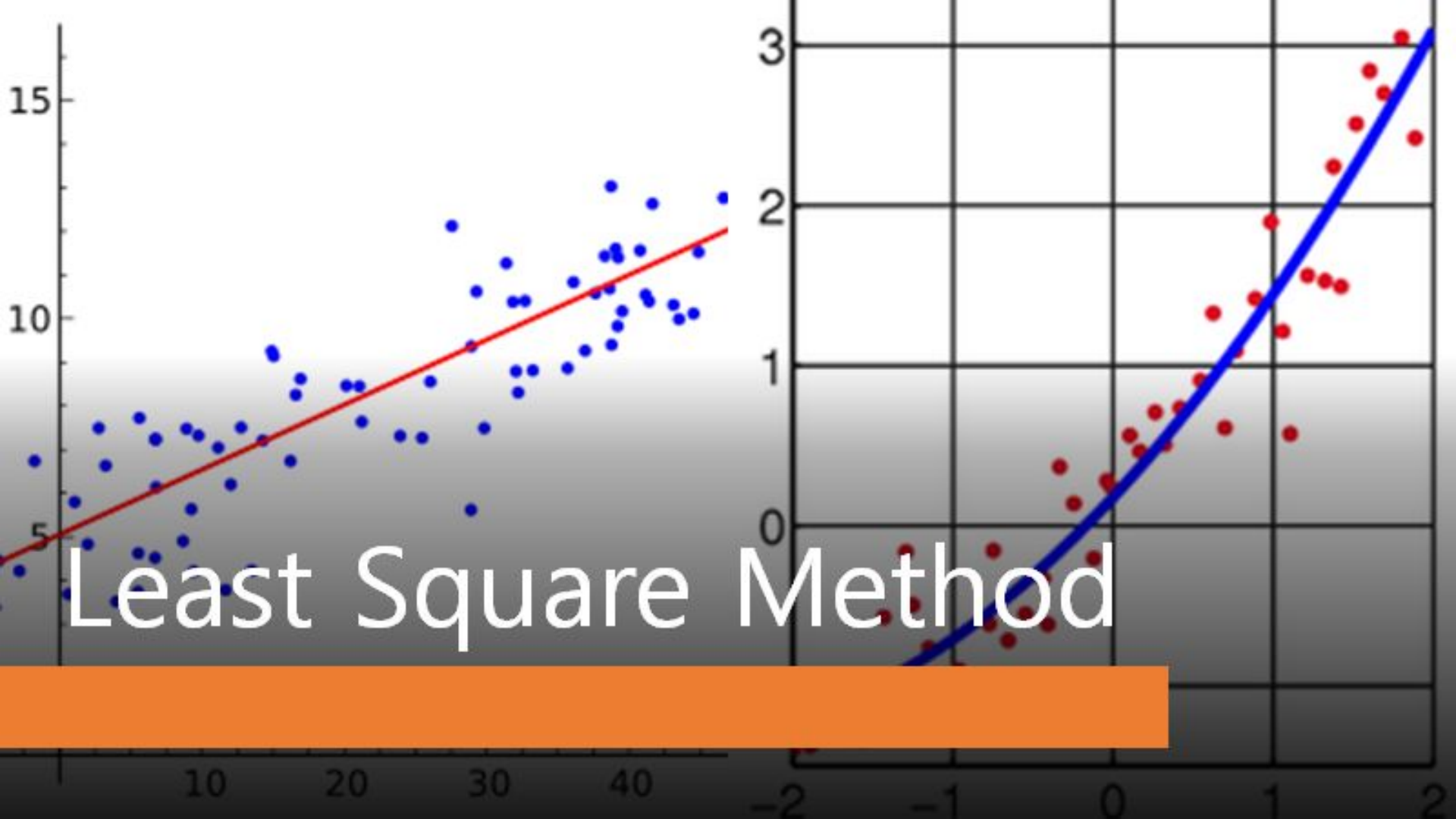
The gradient's direction is calculated using:

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Sobel filter 이용한 Image Gradient 구하기

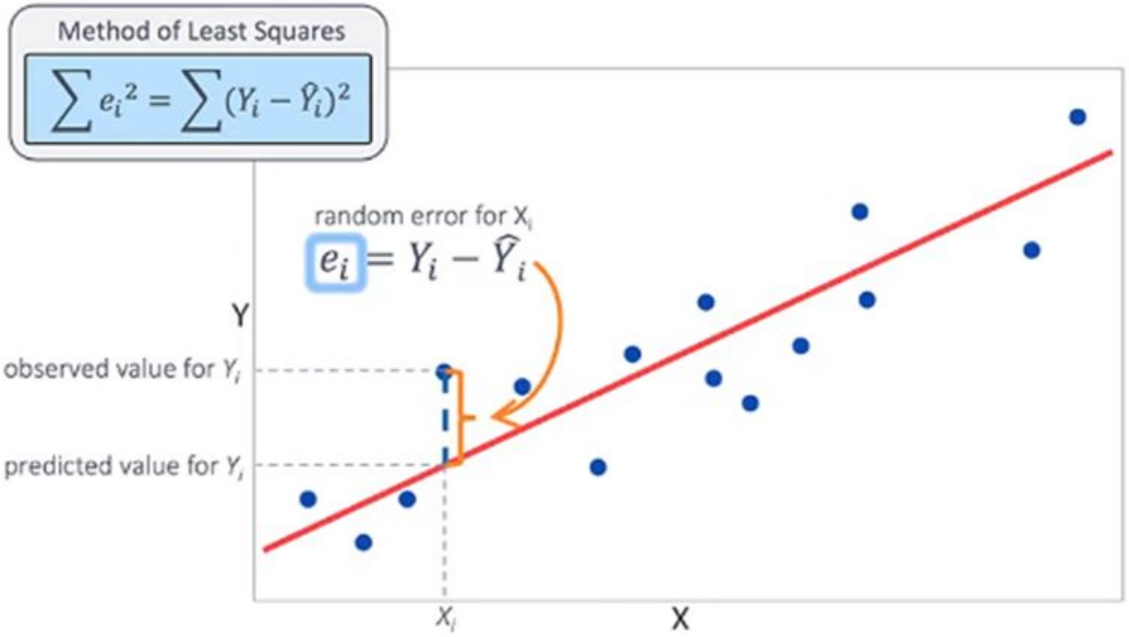



```
sobel_x = filter2([-1 0 1; -2 0 2; -1 0 1], blur_im);
sobel_y = filter2([1 2 1; 0 0 0; -1 -2 -1], blur_im);
Gmag = sqrt(sobel_x.^2 + sobel_y.^2);
Gdir = atan(sobel_y ./ (sobel_x + 1e-4));    % arctan(y/x)를 통해 각도 구하기
```



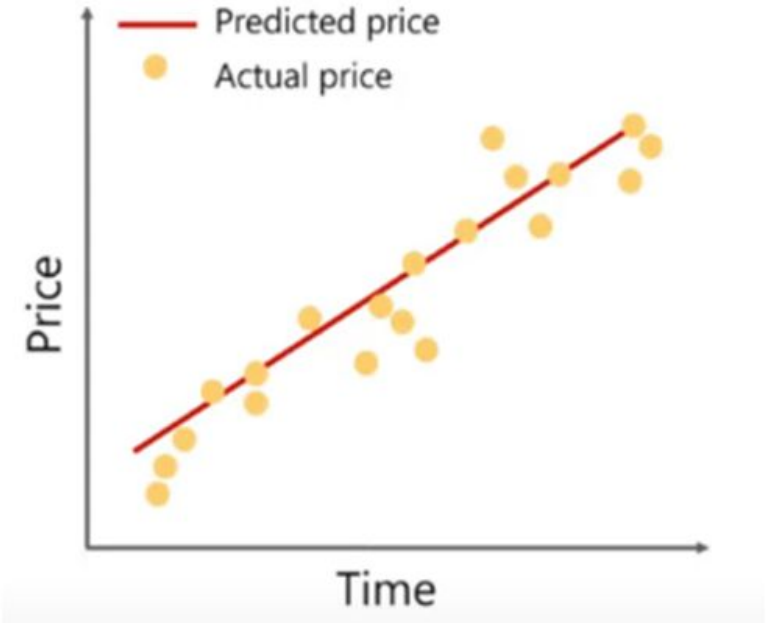
Least Square Method

Least Square Method 최소자승법(최소제곱법)



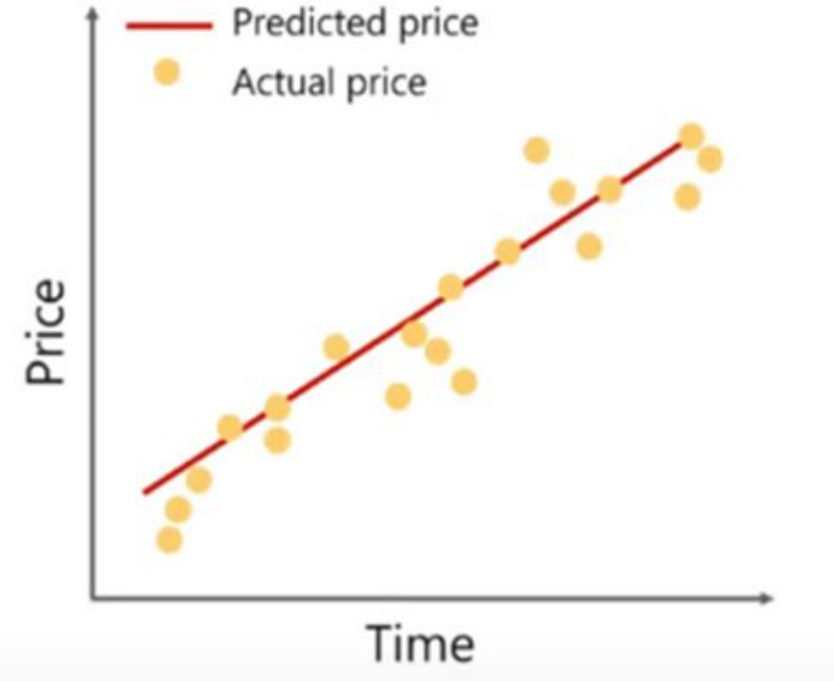
회귀에서 독립변수와 종속변수 사이의 관계를 가장 잘 나타내는 (=오차가 가장 적은) 선을 찾기 위해 사용하는 수학적인 방법

Least Square Method 최소자승법(최소제곱법)



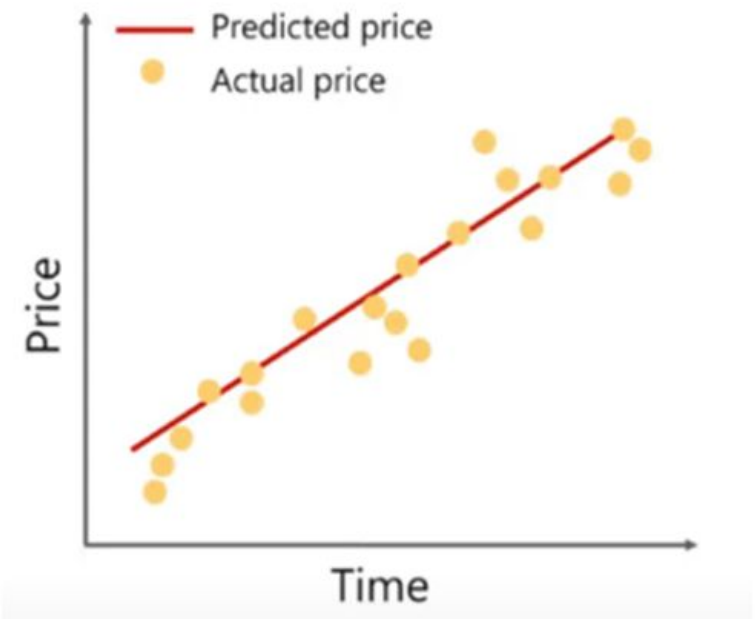
최적선 : 관측치들 사이의 관계 파악을 위해 산포도를 그렸을 때 그 산포도 안에서 점들의 관계를 가장 잘 나타내주는 선

Least Square Method 최소자승법(최소제곱법)



X(독립변수)	Y(종속변수)
2	4
3	5
5	7
7	10
9	15

Least Square Method 최소자승법(최소제곱법)



$$y = mx + c$$

$$m = \frac{n\Sigma xy - (\sum x)(\sum y)}{n\Sigma x^2 - (\Sigma x)^2} = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

$$C = y - mx = \frac{\sum y - m\Sigma x}{n}$$

Least Square Method 최소자승법(최소제곱법)

	X(독립변수)	Y(종속변수)	Xy	x^2	$x-\bar{x}$	$y-\bar{y}$
	2	4	8	4	-3.2	-4.2
	3	5	15	9	-2.2	-3.2
	5	7	35	25	-0.2	-1.2
	7	10	70	49	1.8	1.8
	9	15	135	81	3.8	6.8
Σ	26	41	263	168		
평균	5.2	8.2				

Least Square Method 최소자승법(최소제곱법)

```
n = len(x)

mean_x = np.mean(x)

mean_y = np.mean(y)


numer = 0
denom = 0

for i in range(n) :
        numer += (X[i] – mean_x) * (Y[i] – mean_y)
        denom += (X[i] – mean_x) **2
```

X(독립변수)	Y(종속변수)
2	4
3	5
5	7
7	10
9	15

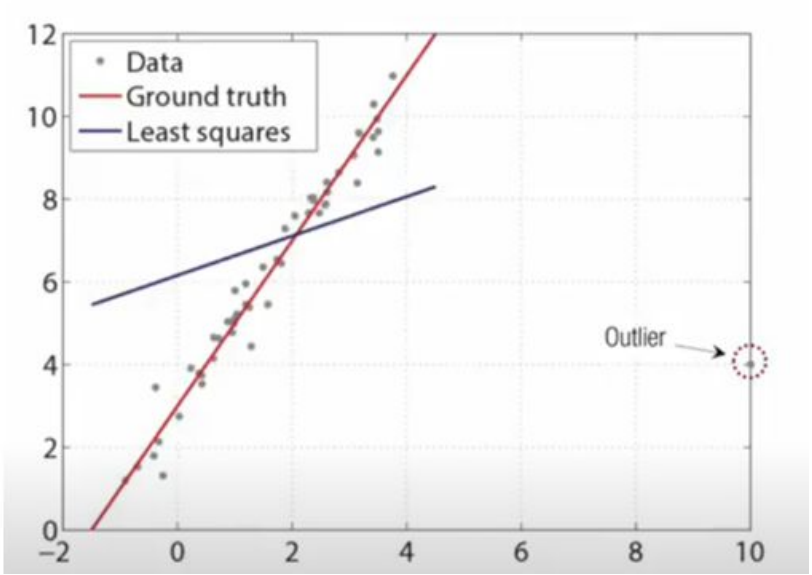
If x = 8, y = ?

y = 1.518x + 0.305

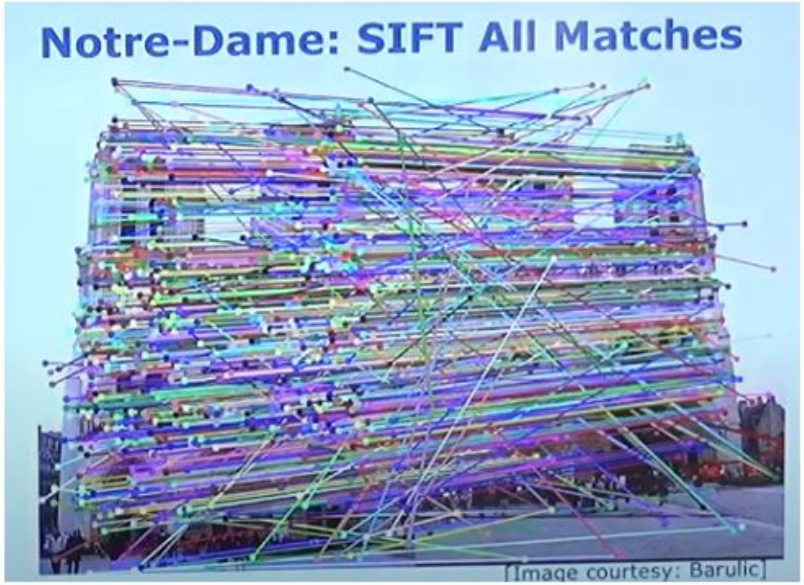
Least Square Method 최소자승법(최소제곱법)

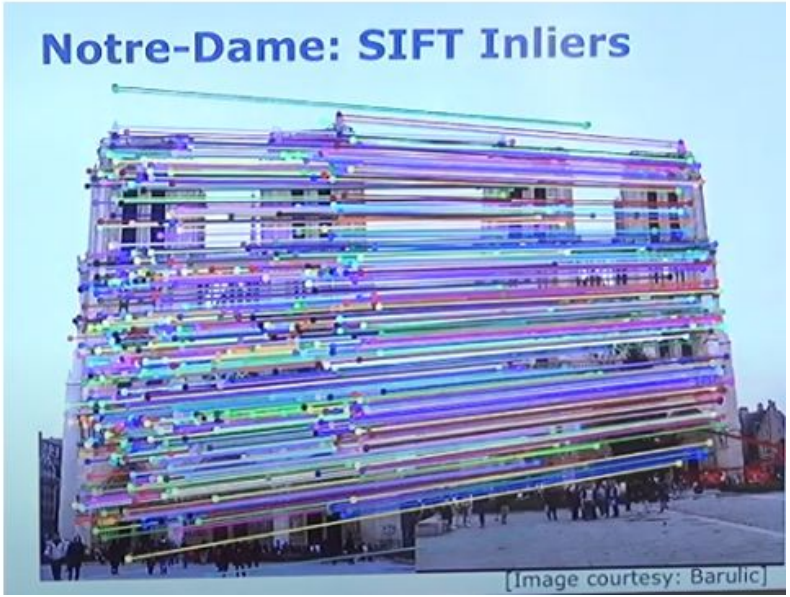
장점
- scikit learn 등의 패키지를 통해 회귀선 오차제곱의 최소값 쉽게 계산 가능
- 선형적이지 않은 데이터여도 꽤 정확히 예측 가능

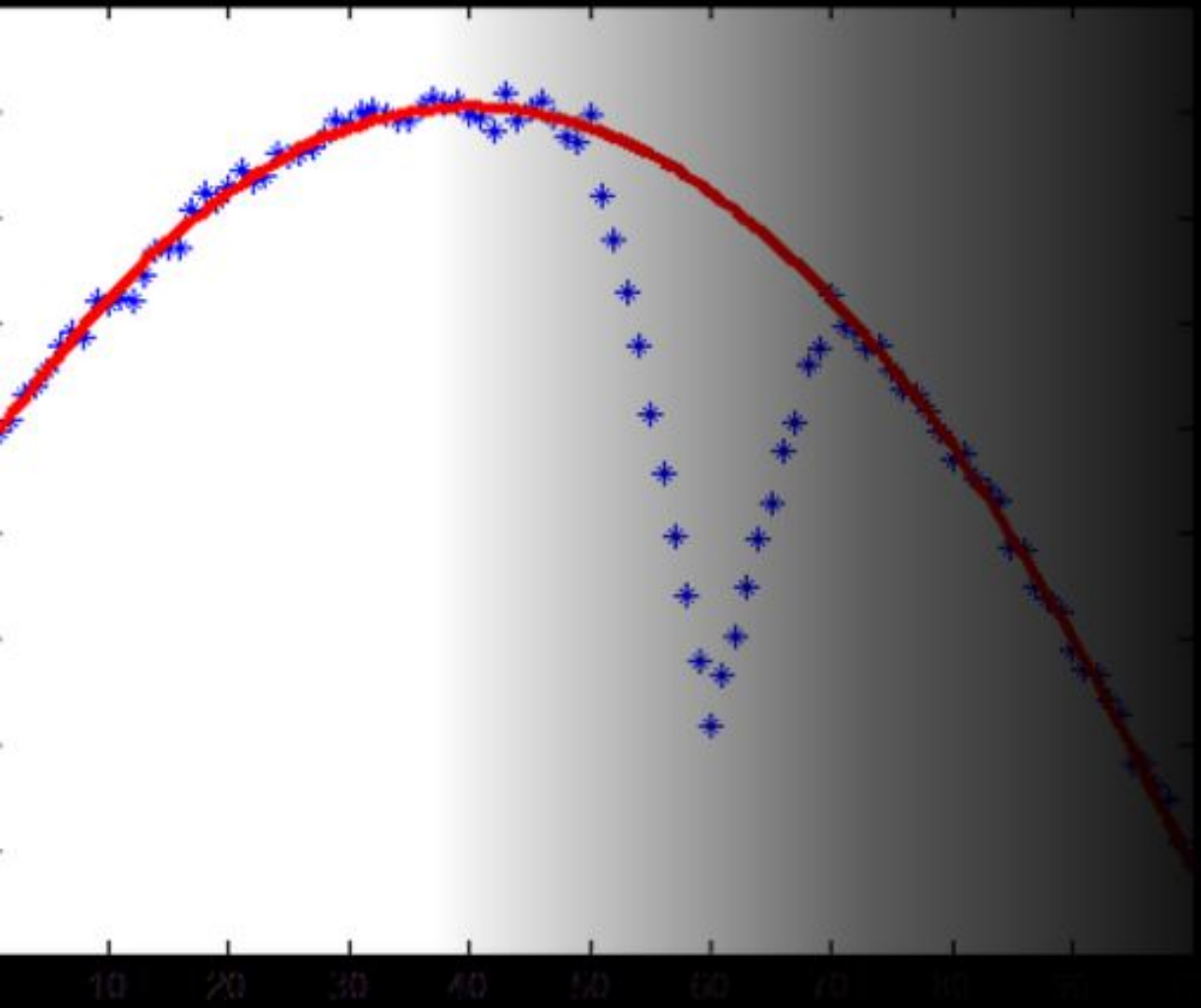
단점 : 이상치에 크게 영향을 받음



Least Square Method 최소자승법(최소제곱법)







About RANSAC

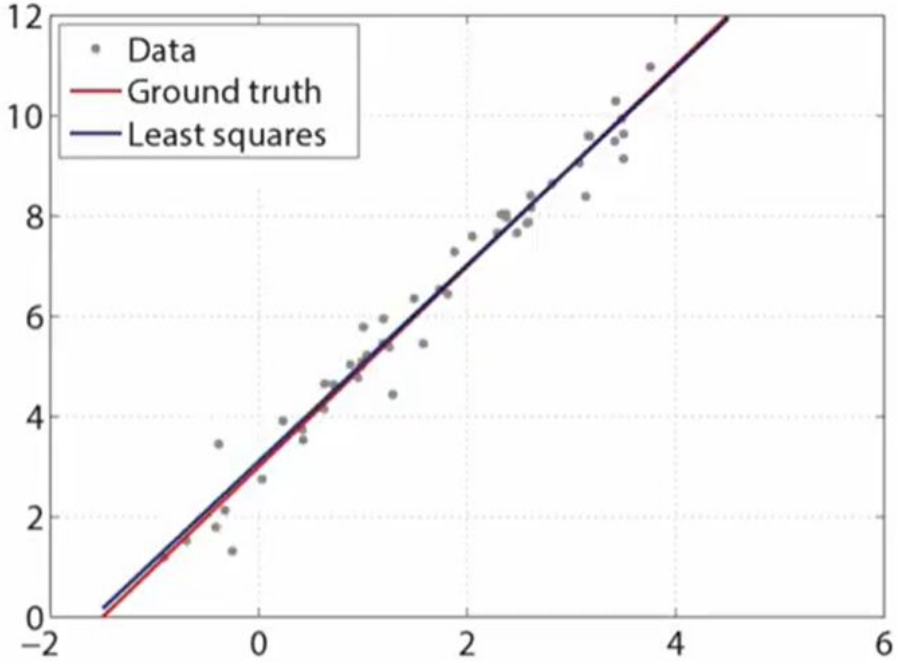
RANSAC(Random Sample Consensus)

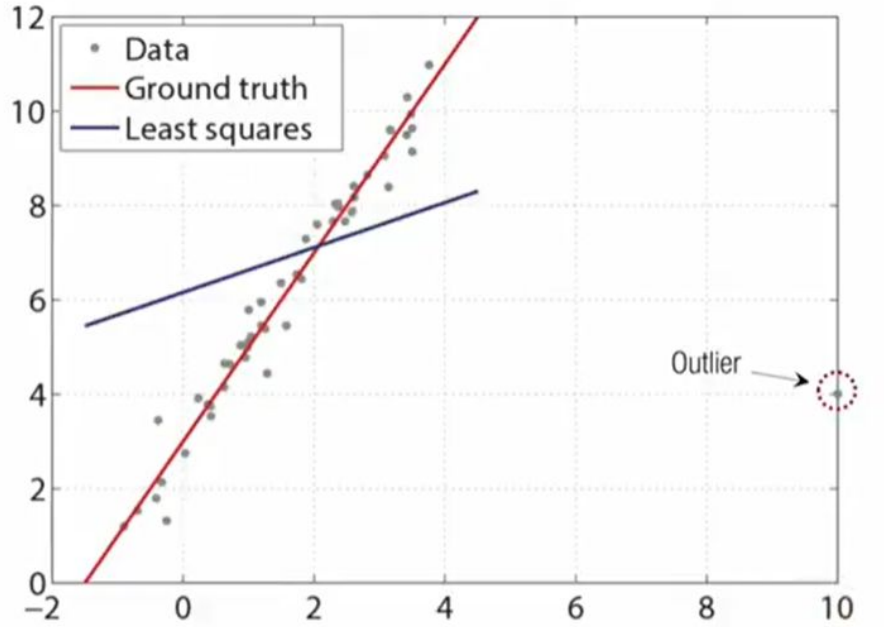
랜덤 샘플 +

컨센서스(consensus)는 공동체 구성원의 일반적인 동의를 말한다.

관측 데이터로부터 수학적 모델의 파라미터를 추정하는 방법

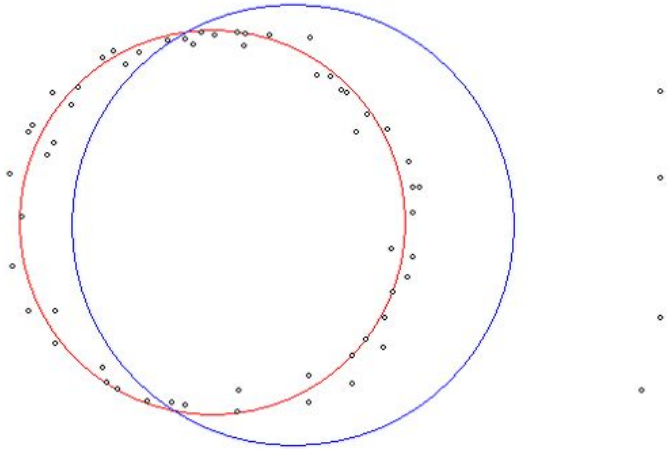
문제 정의





우리가 추정하고자 하는 **모델에 해당하지 않는 데이터**: 이상치, 특이치, gross error, noise… -> **Outlier**
모델에 적합한 데이터: Inlier

원 추정 예시



red: Ransac
blue: LSM

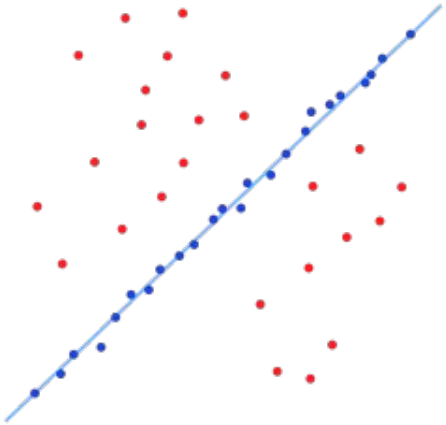
Outlier와 Inlier로 분류 후 가장 많은 Inlier에게 Consensus받은 모델을 채택하는 것이 RANSAC

핵심 전략

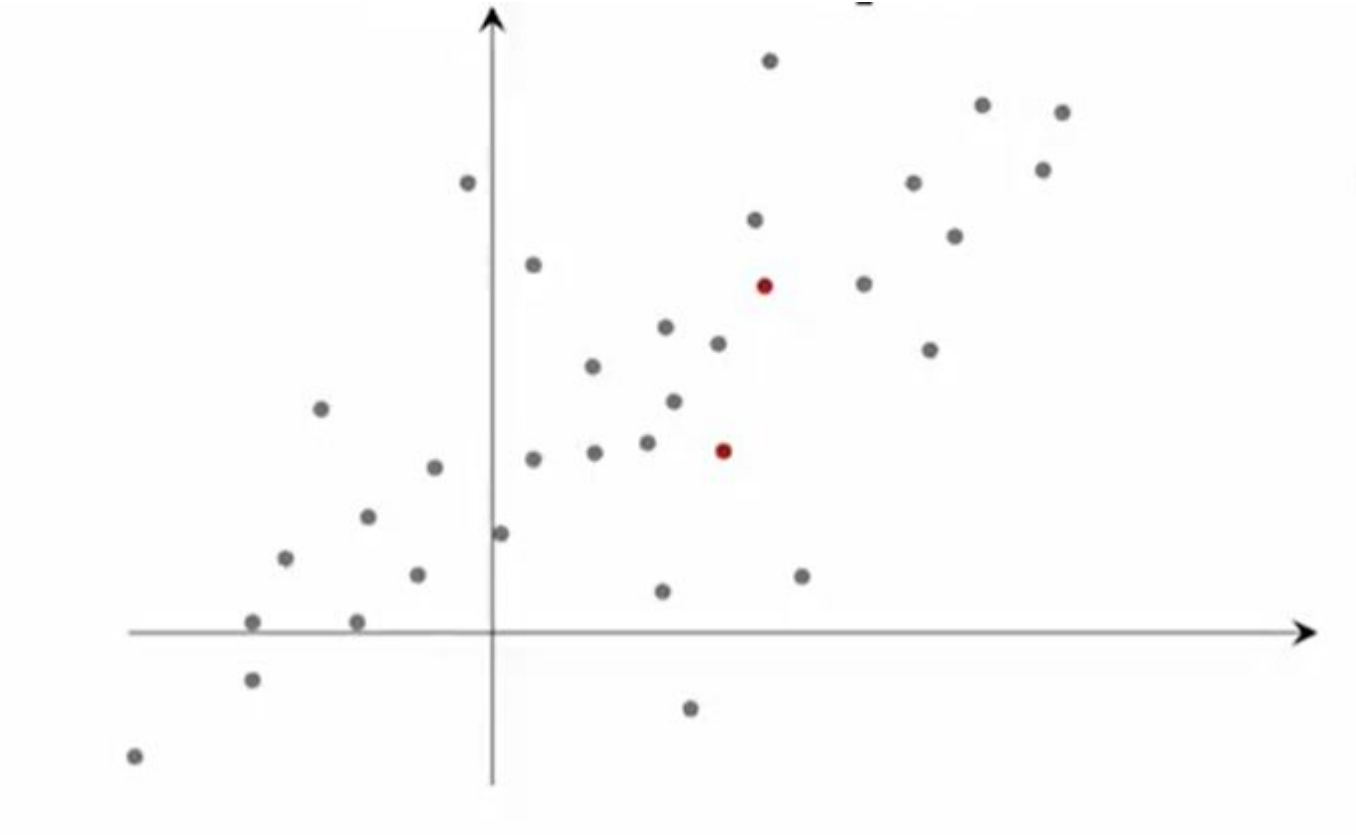
최소의 데이터로 모델을 추정: N

ex) 원을 추정 -> 3개, 직선-> 2개

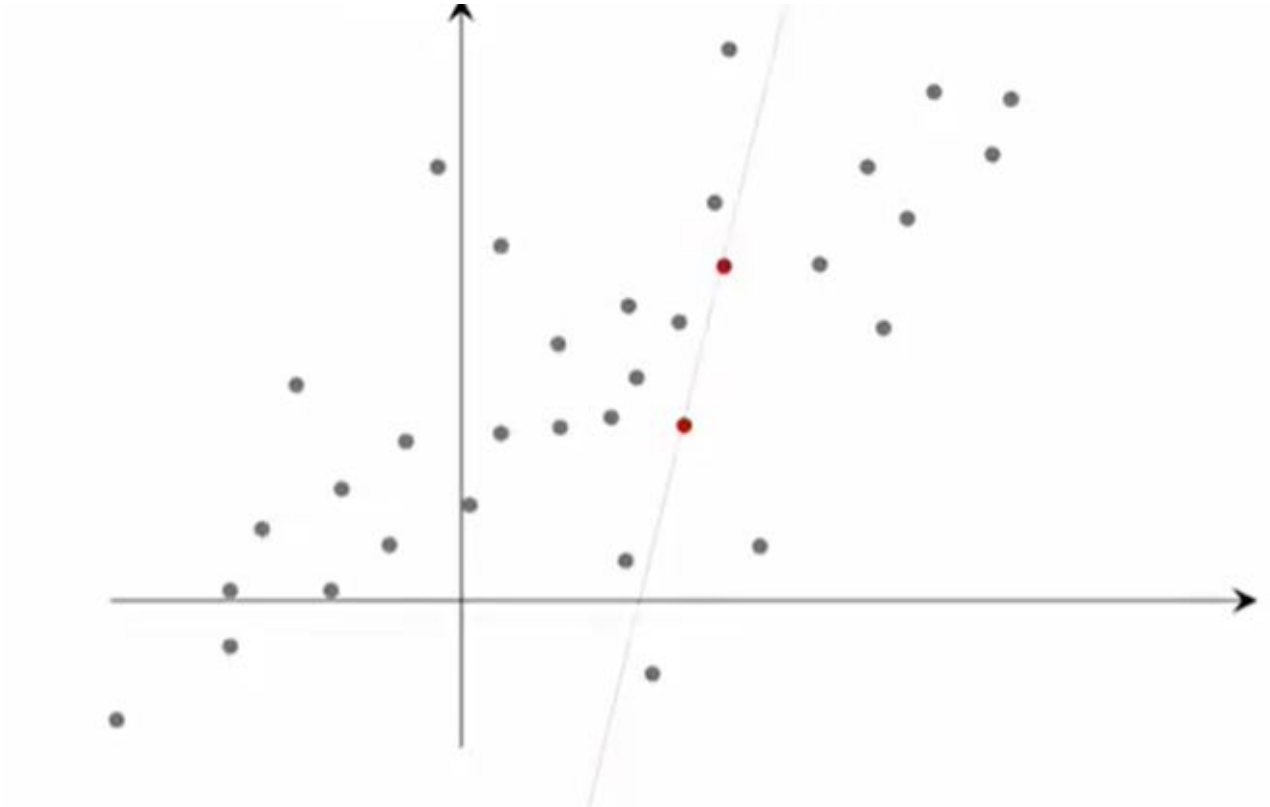
1. 관측 데이터로부터 N개의 데이터 Random Sampling
2. 모델 생성
3. Thresholding(임계값- 거리 -을 통한 분리)
4. Inlier counting



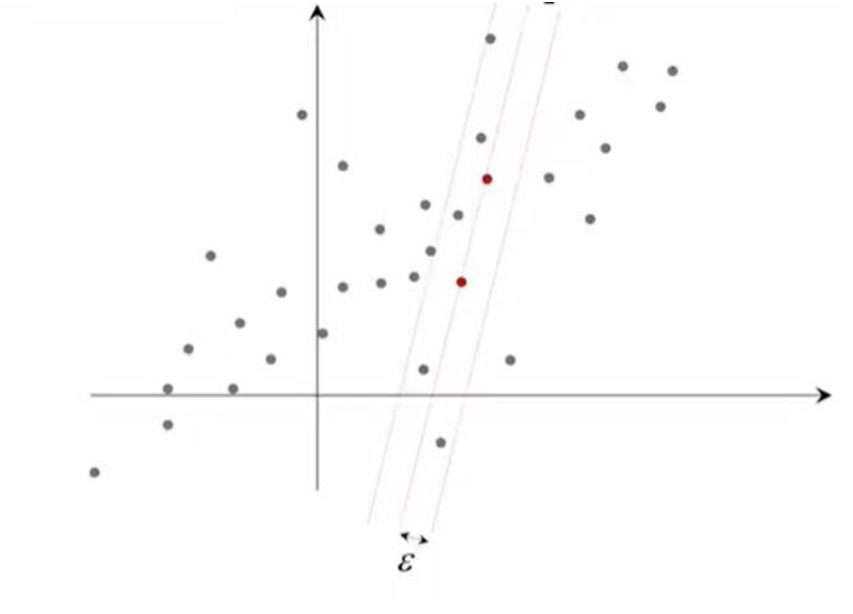
1.관측 데이터로부터 N개의 데이터 Random Sampling



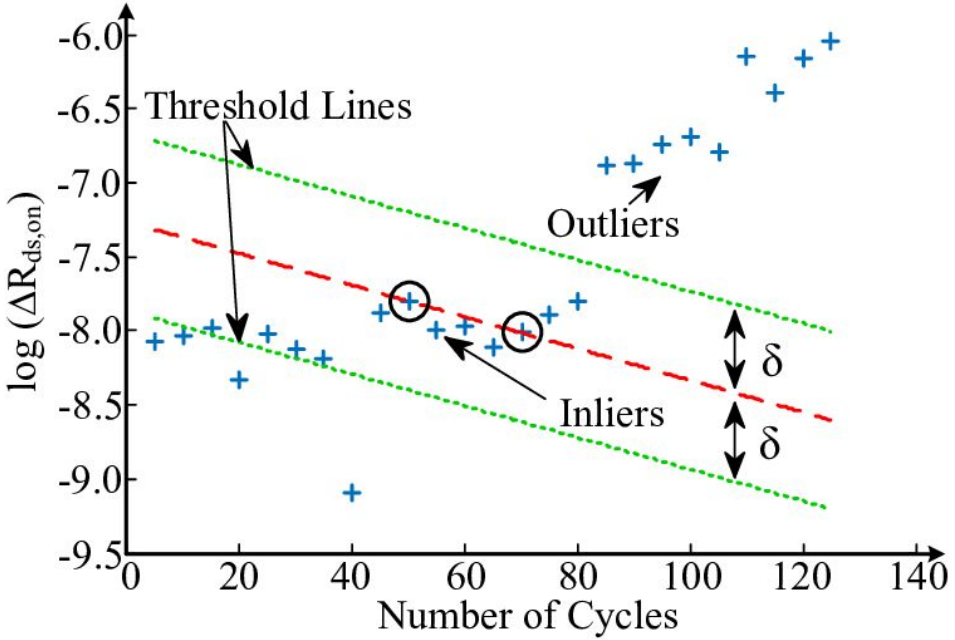
2. 모델 생성



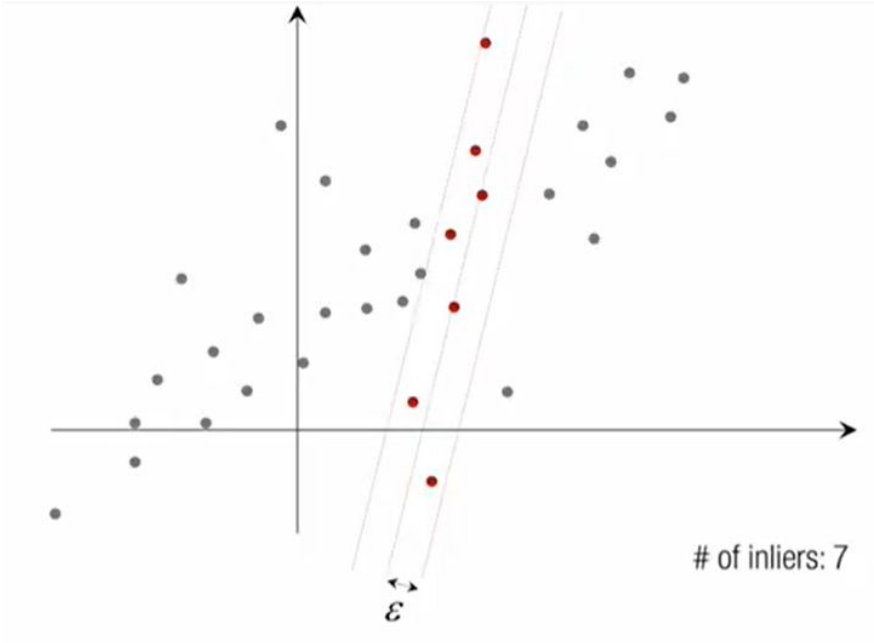
3. Thresholding

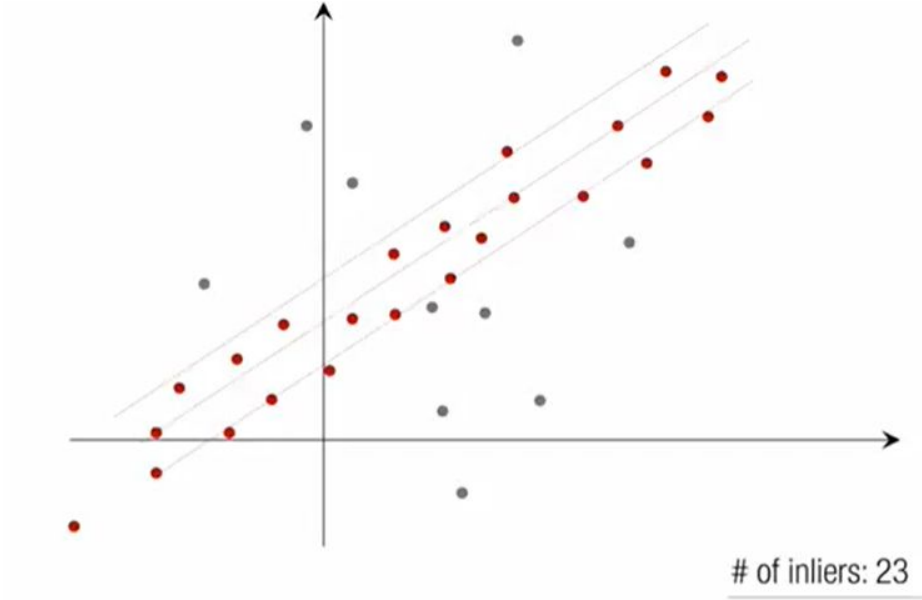


혹은 거리(Distance)로 표현



4. Inlier counting





몇 번을 반복해야 하는가?

Probability of choosing an inlier: $w = \frac{\#\text{ of inliers}}{\#\text{ of samples}}$

Probability of building a correct model: w^n where n is the number of samples to build a model.

Probability of not building a correct model during k iterations: $\left(1-w^n\right)^k$

$\left(1-w^n\right)^k = 1-p$ where p is desired RANSAC success rate. $k = \frac{\log(1-p)}{\log\left(1-w^n\right)}$

입력 데이터들 중에서 **inlier의 비율을 α**
inlier에서만 샘플이 뽑힐 **확률 p**

$$p=1-(1-\alpha^m)^N$$

$$N=\frac{\log(1-p)}{\log(1-\alpha^m)}=\frac{\log(1-0.999)}{\log(1-0.8^3)}=9.6283$$

https://darkpgmr.tistory.com/61

○ 원하는 실패확률 아래로 유지될 수 있도록 N을 충분히 높게 선택하라.

$$N=\log(1-p)/\log\left(1-(1-e)^s\right)$$

Sample size	Proportion of outliers						
S	**5%**	**10%**	**20%**	**25%**	**30%**	**40%**	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

adapted from Hartley & Zisserman

[그림 7.14] p=0.99일 때 N의 선택

징징이

장단점

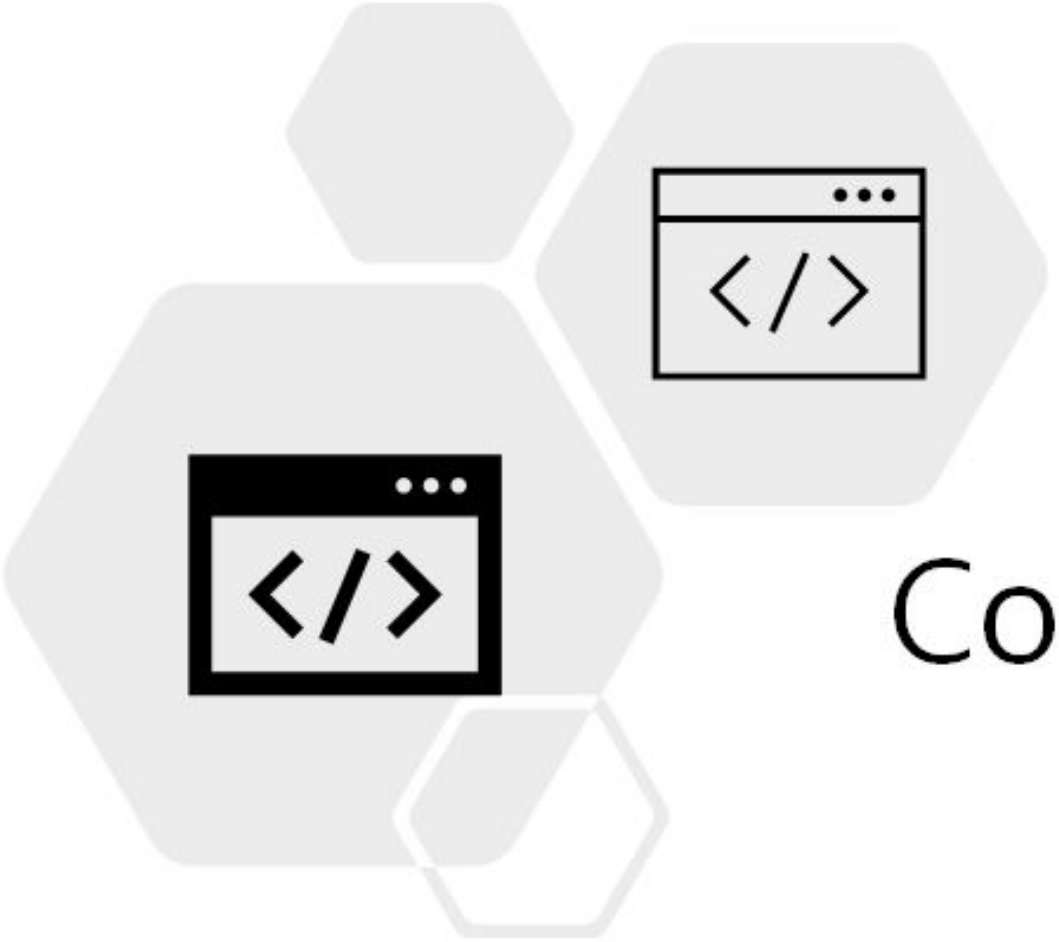
RANSAC의 장점은 **모델 매개변수의 강력한 추정을 수행**할 수 있다.
즉, 데이터 집합에 **상당한 수의 특이치가 있더라도 높은 정확도로 매개변수를 추정**할 수 있다.

RANSAC의 단점은 이러한 **파라미터를 계산하는 데 걸리는 시간에 상한이 없다**는 것이다.
계산된 반복 횟수가 제한될 때 얻은 모델은 **최적 상태가 아닐 수 있으며 데이터에 적합한 솔루션도 아닐 수 있다**.

더욱이 RANSAC은 오염도가 적당히 높은 세트에 대해서도 **최적의 세트를 항상 찾을 수 있는 것은 아니며**,
일반적으로 **Inlier가 50% 미만일 때 성능이 떨어진다**.

또 다른 단점은 문제별 **임계값을 설정**해야 한다는 것이다.

1. 결과 일관성 X(랜덤)
2. 반복 횟수 결정 X -> 시간 상한 X
3. 하나의 feature



by 해리

Code Review

실습 주제 소개

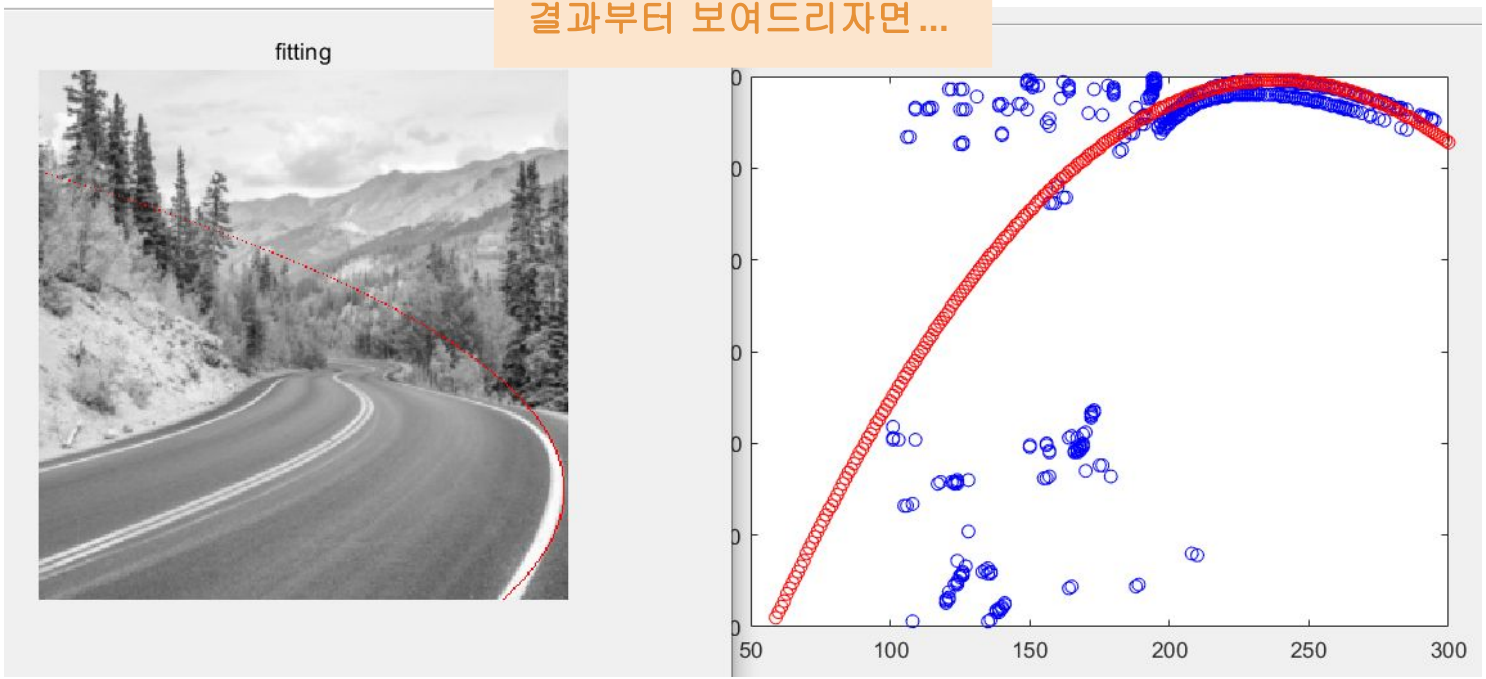
- 1. 공부해서 발표할 내용
- Least Square method and line fitting
- RANSAC Based Line Fitting Method
- RANSAC 과 Neural Network 이 어떤 차이가 있는지 고민해 보기

- 2. 코드를 작성해서 실험할 내용
- - Sobel filter 기반 gradient magnitude와 orientation 계산
- - Gradient orientation과 gradient magnitude의 범위를 설정하여 왼쪽 차선을 구성하는 edge만 선택하는 방법 구현
- (고민해서 해보세요. 힌트: 특정 차선을 구성하는 edge는 같은 방향의 orientation을 가지며, 큰 magnitude 를 갖는다)
- - RANSAC 방법론을 통해 인터넷에서 아무 고속도로 이미지나 긁어와서, 차량 기준 우측이나 좌측 차선을 따 보기 구현

요약: 도로 이미지 하나 긁어 와서 차선 따는 작업 해보기

Code Review - 해리

결과부터 보여드리자면...



Code Review - 해리

원본이미지

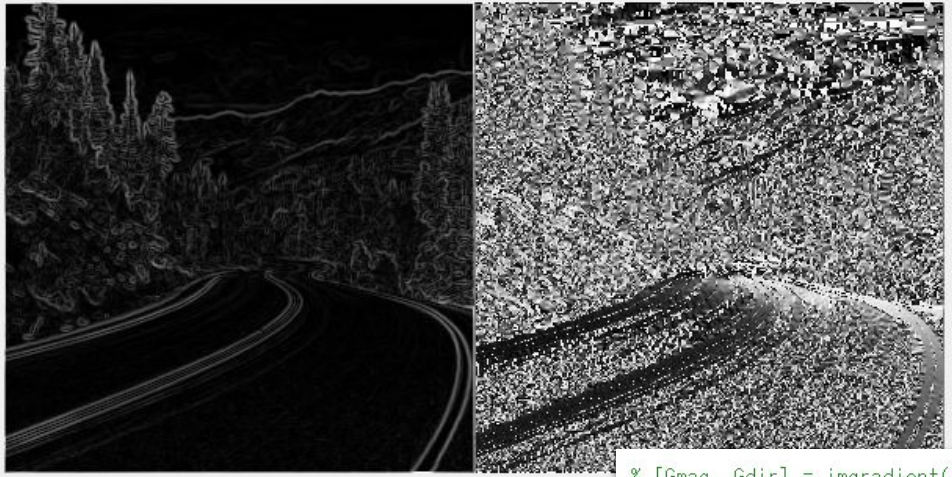
grayscale처리
& 흐리게 만든
이미지

1 이미지를 처리하기 좋게 변경

※ 주의 ※ MATLAB 코드임

```
clear all
clc

original_im = imread("Roads-300x300.jpg");  % 이미지 로드
im = rgb2gray(original_im);                 % 그레이스케일로 변환
blur_im = imgaussfilt(im);                  % 노이즈 제거
figure, imshow(blur_im);
```

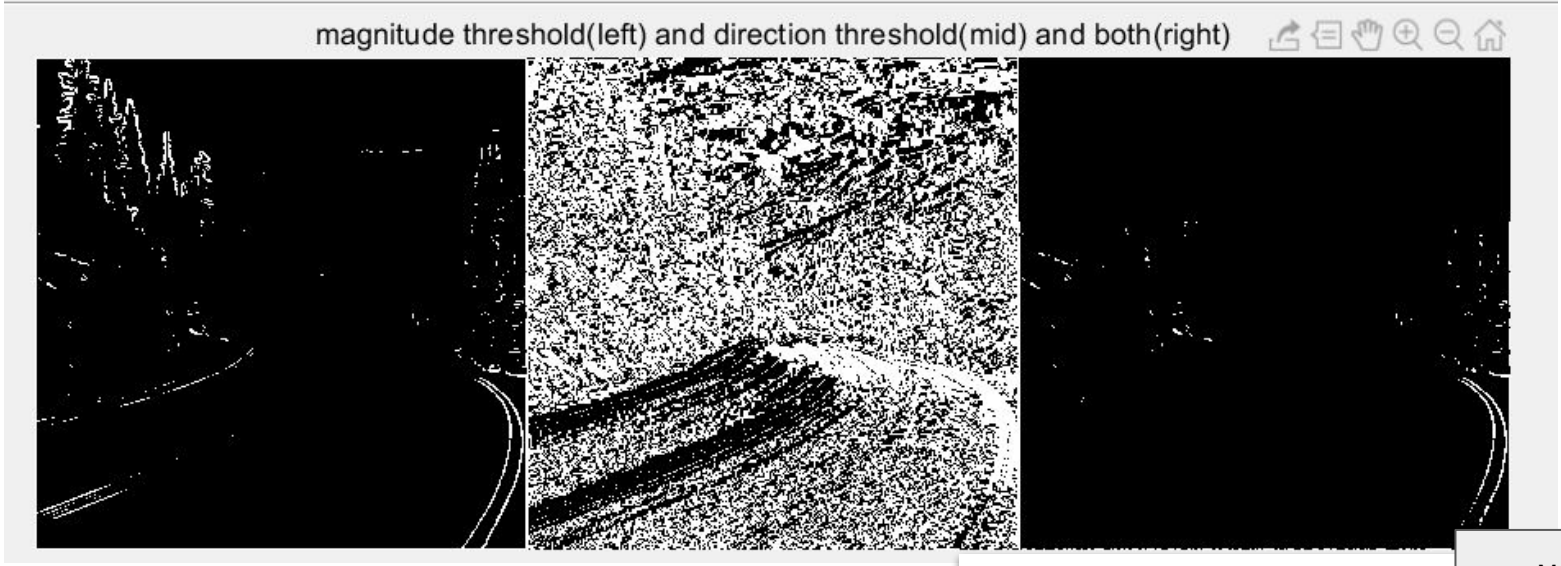
image gradient magnitude(left) and direction(right)



2 Sobel Filter 적용해서 Gradient Magnitude, Gradient Orientation(?) 구하기

※ 주의 ※
MATLAB코드임

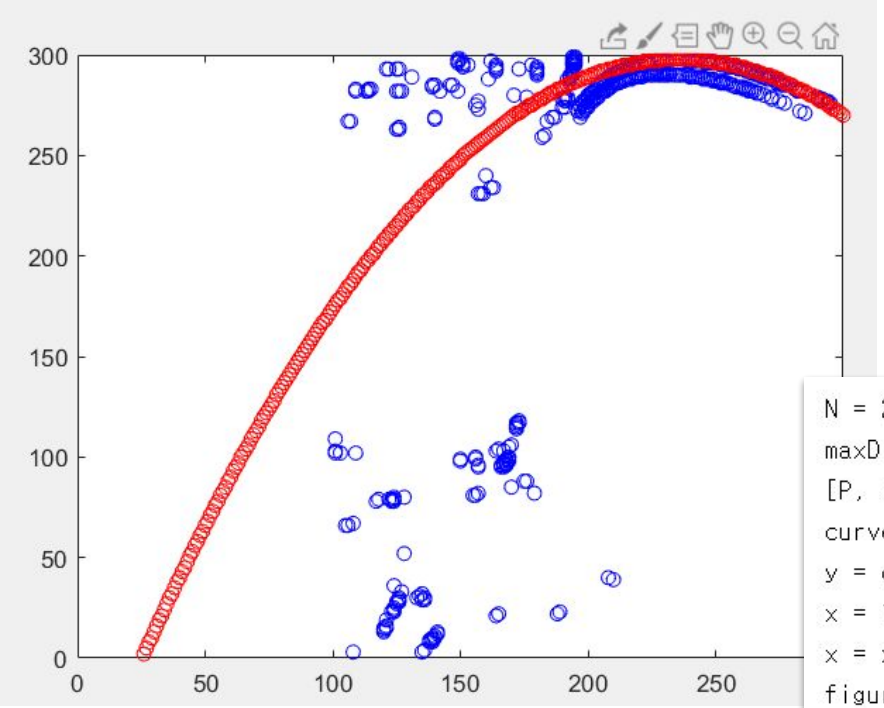
```
% [Gmag, Gdir] = imgradient(im, 'sobel');      % sobel filter를 이용해서 gradient의 방향과 크기 구하기
% 아마 직접 구한다면 이런 코드일 듯
sobel_x = filter2([-1 0 1; -2 0 2; -1 0 1], blur_im);
sobel_y = filter2([1 2 1; 0 0 0; -1 -2 -1], blur_im);
Gmag = sqrt(sobel_x.^2 + sobel_y.^2);
Gdir = atan(sobel_y ./ (sobel_x + 1e-4));    % arctan(y/x)를 통해 각도 구하기

% magnitude와 direction 성분을 이미지로 나타낼 수 있도록 0-1 사이로 scaling
Gmag_max = max(max(Gmag));
Gmag_min = min(min(Gmag));
Gdir_max = max(max(Gdir));
Gdir_min = min(min(Gdir));
mag_img = (Gmag - Gmag_min)./(Gmag_max - Gmag_min);
dir_img = (Gdir - Gdir_min)./(Gdir_max - Gdir_min);
```



3 Threshold(문턱값) 정해서 통과되는 픽셀만 골라내기

※ 주의 ※
MATLAB코드임

```
m_img = zeros(300, 300);
m_img(mag_img > 0.3) = 1.;
d_img = zeros(300, 300);
d_img(dir_img > 0.35) = 1.;
f_img = zeros(300, 300);
f_img(m_img & d_img) = 1.;
```

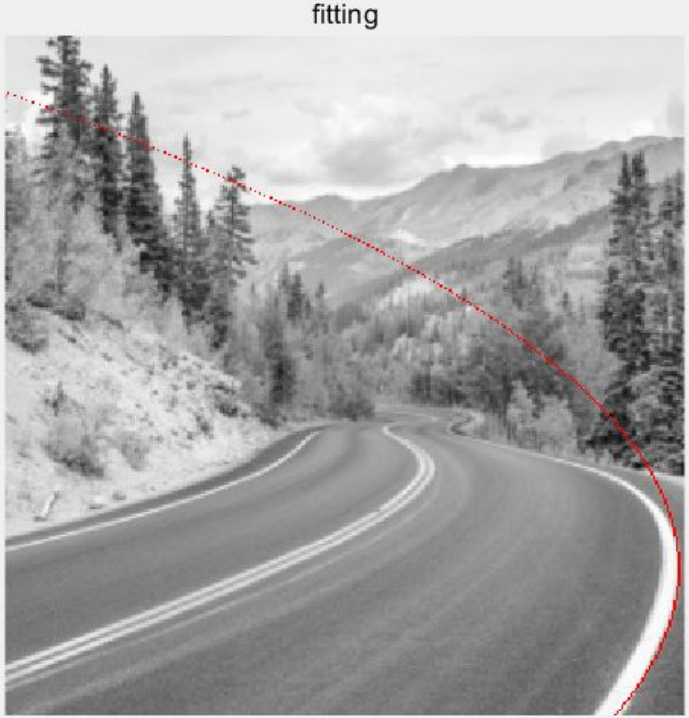


4 RANSAC 이용한 fitting

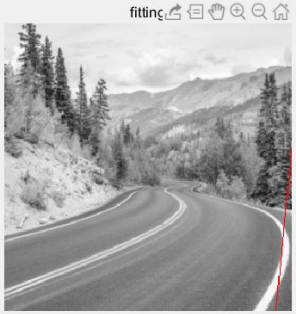
※ 주의 ※
MATLAB코드임

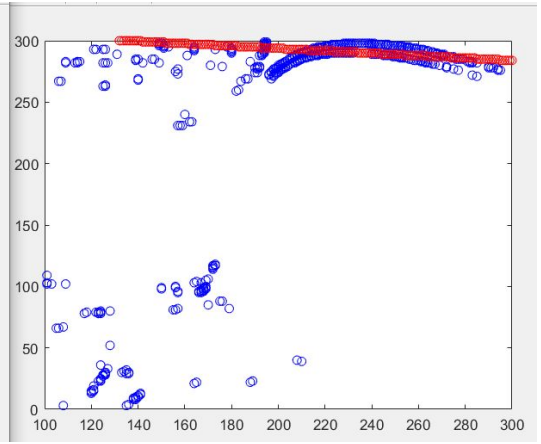
```
N = 2; % Nth polynomial
maxDistance = 1; % inlier 기준
[P, inlierIdx] = fitPolynomialRANSAC([row column],N, maxDistance);
curve = polyval(P, 1:300);
y = ceil(curve(curve<=300 & curve>=1)); % column index
x = 1:300; % row index
x = x(curve<=300 & curve>=1);
figure, plot(row, column,'bo', x, y,'ro');

% line / curve 그런 것 이미지로 표시할 준비(인덱스 뽑아오기)
mask = zeros(300, 300);
mask(sub2ind(size(mask), x, y)) = 1.;
```

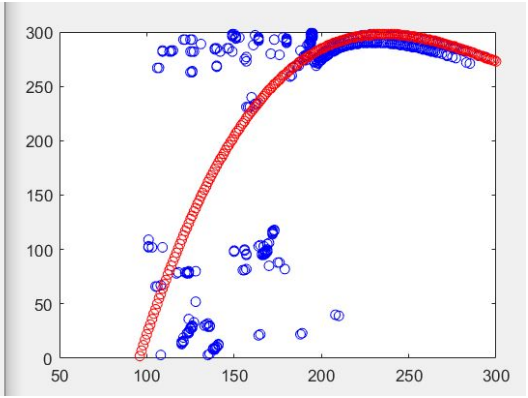
2차함수 fitting



1차함수(line) fitting





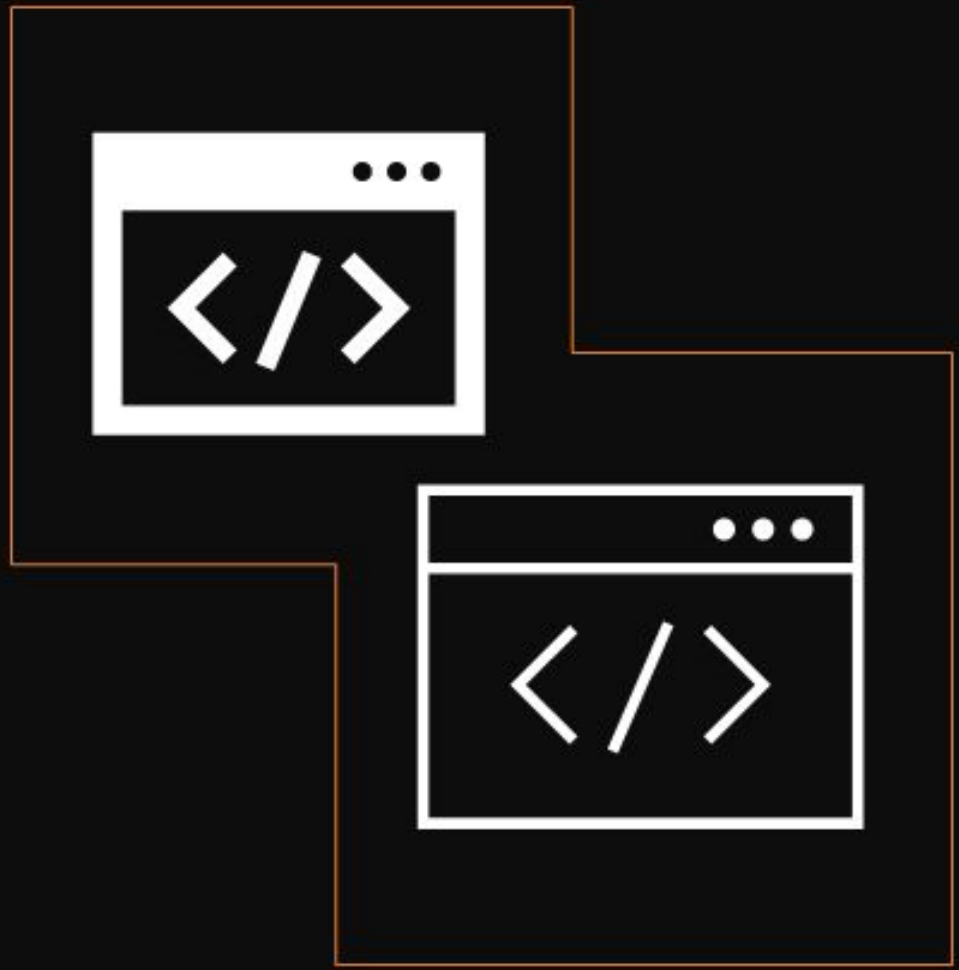


3차함수 fitting

https://www.notion.so/44422c055e0d4669ba721ab6569ffa8b 상세 코드는 여기에!

Code Review

by 영채



Code Review

Convert Gray	흑백 사진으로 변환 (처리속도 향상)
Gaussian Blur	Edge로 인식되는 노이즈를 줄이기 위해 사진을 뭉겜
Sobel Filter	Sobel x,y 를 합쳐 Edge를 검출함.
Data Preprocessing	학습을 위해 픽셀데이터를 좌표데이터로 변환함
RANSAC Regression	노이즈에 최대한 영향을 받지 않게 회귀시킴

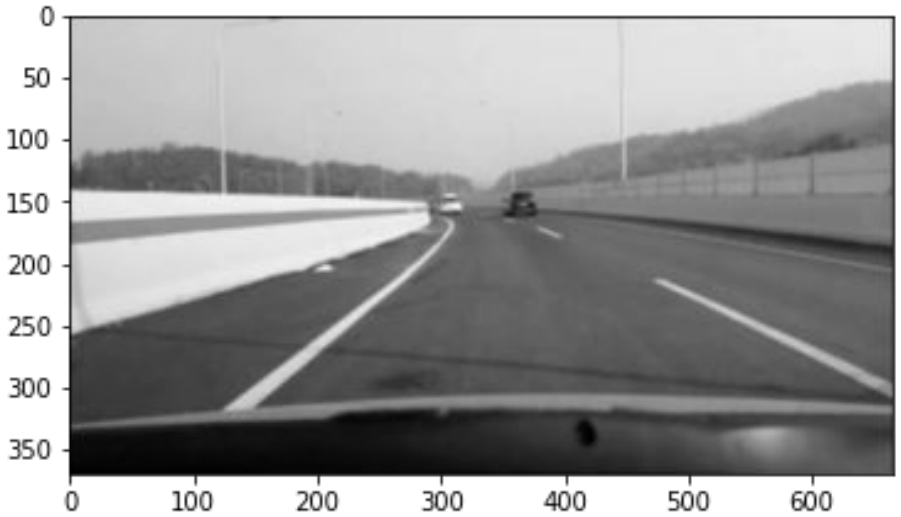
Code Review

- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression

⬆️ 원본 사진

Code Review

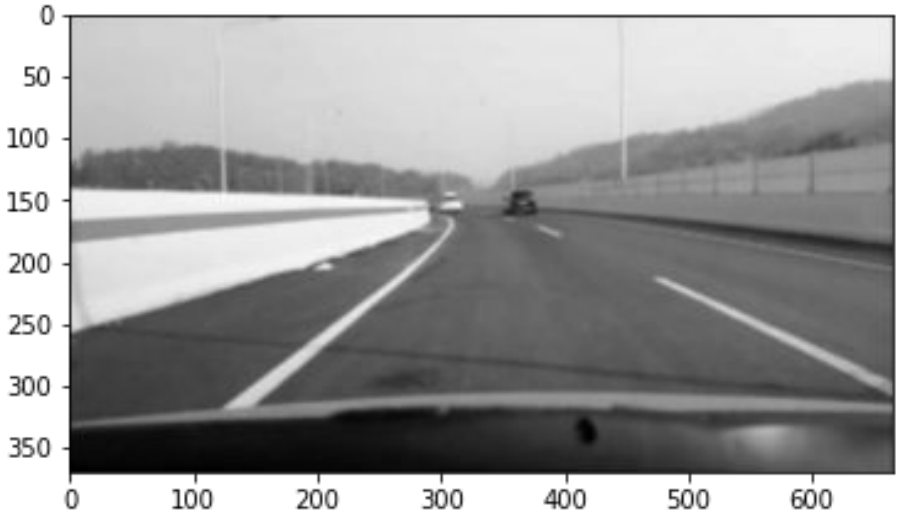
- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression



⬆️ 원본 사진

Code Review

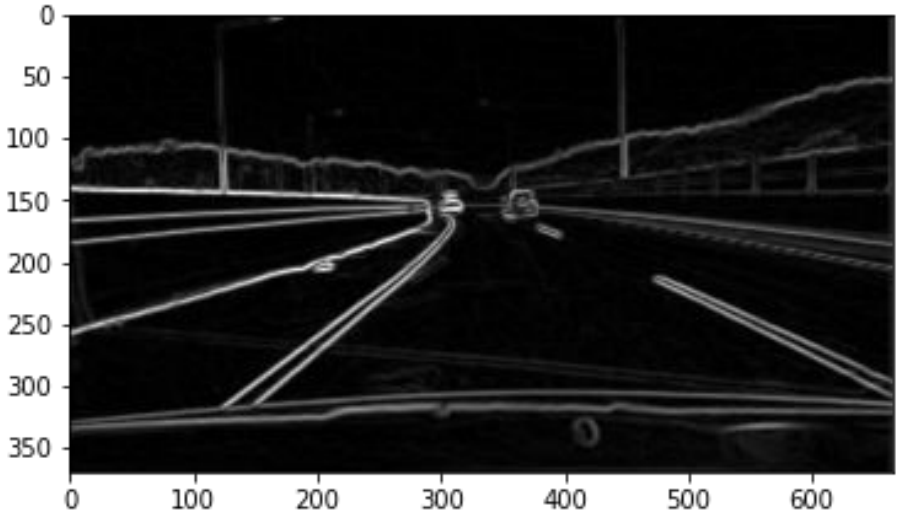
- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression



⬆️ 가우시안 블러 처리

Code Review

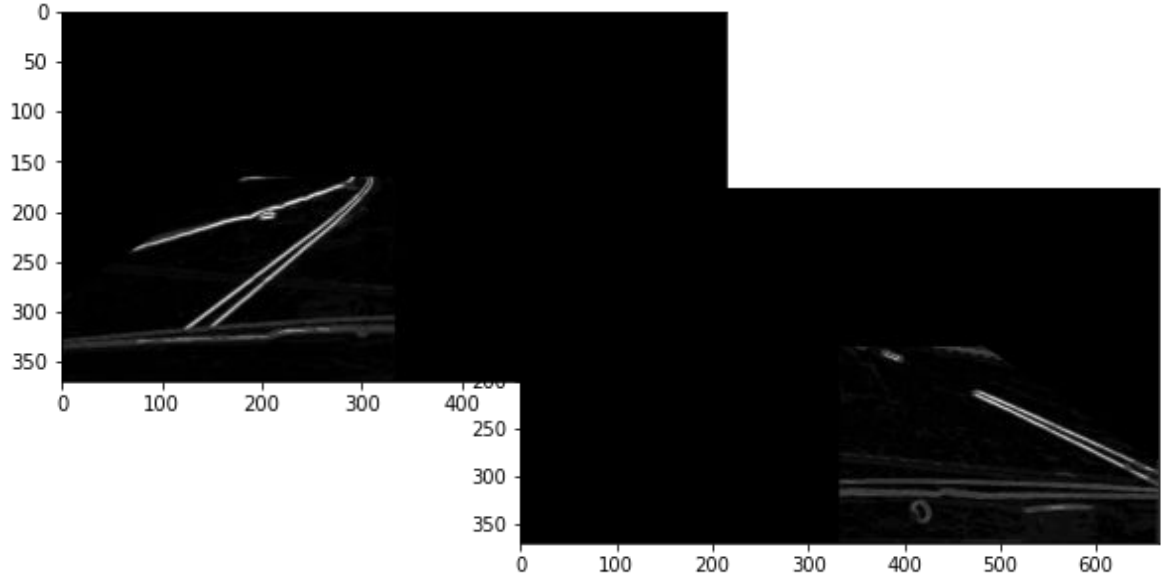
- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression



⬆️ Sobel 블러 처리

Code Review

- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression



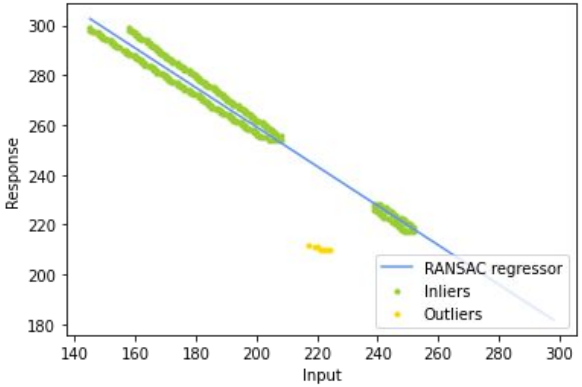
⬆️ 오른쪽 왼쪽 차선구역 나누기

Code Review

- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression

```
[[0.      0.      0.      ... 0.      0.      0.     ]
 [0.      0.      0.      ... 0.      0.      0.     ]
 [0.      0.      0.      ... 0.      0.      0.     ]
 ...
 [8.      8.      3.53125 ... 0.      0.      0.     ]
 [2.      2.      2.      ... 0.      0.      0.     ]
 [0.      0.      0.      ... 0.      0.      0.     ]]
```

⬆️ 픽셀데이터 [H, W]




```
[382 383 393 394 387 388 391 392 393 480 481 482 482 483 484 485 486 476
 477 478 479 487 488 479 480 481 489 490 491 492 480 481 482 483 491 492
 493 494 483 484 485 493 494 495 496 484 485 486 487 495 496 497 498 487
 488 489 490 498 499 500 501 489 490 491 492 500 501 502 503 491 492 493
 494 503 504 505 493 494 495 496 505 506 507 496 497 498 508 509 510 497
 498 499 500 509 510 511 512 500 501 502 512 513 514 502 503 504 514 515
 505 506 507 506 507 508 509 519 520 509 510 511 521 522 523 511 512 523
 524 525 514 515 526 527 528 515 516 517 528 529 517 518 519 531 532 519
 520 521 532 533 534 521 522 523 524 535 536 537 523 524 525 537 538 539
 526 527 539 540 541 528 529 541 542 530 531 532 544 545 531 532 533 545
 546 533 534 535 536 548 549 535 536 537 550 551 537 538 539 540 539 540
 541 542 542 543 544 544 545 567 568 569 570 571 572 557 572 573 559 575
 576 561 577 563 564 580 564 565 566 581 582 567 568 584 585 568 569 570
 585 586 587 571 572 588 589 572 573 574 590 575 595 580 597 598 581 582
 599 600 583 584 585 602 585 586 587 588 589 589 590 591 591 592 593 593
 594 596 604 605 616 618 624 625 626 647 648 648 649 650]
[174 174 175 175 176 176 178 178 178 213 213 213 214 214 214 215 215 216
 216 216 216 216 216 217 217 217 217 217 217 217 218 218 218 218 218 218
 218 218 219 219 219 219 219 219 219 220 220 220 220 220 220 220 220 221
 221 221 221 221 221 221 221 222 222 222 222 222 222 222 222 223 223 223
 223 223 223 223 224 224 224 224 224 224 224 225 225 225 225 225 225 226
 226 226 226 226 226 226 226 227 227 227 227 227 227 227 228 228 228 228 228
 229 229 229 230 230 230 230 230 230 231 231 231 231 231 231 232 232 232
 232 232 233 233 233 233 233 234 234 234 234 234 235 235 235 235 235 236
 236 236 236 236 236 237 237 237 237 237 237 237 238 238 238 238 238 238
```

⬆️ X, Y 좌표 데이터 [X], [Y]

Code Review

- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression

```
df = pd.DataFrame(roi/255).stack().rename_axis(['y', 'x']).reset_index(name='val')
df = df[df['val']>0.7].reset_index() # 임계점 설정

x, y = df.loc[:,'x'].values, df.loc[:,'y'].values
x = x.reshape((-1,1))
y = y.reshape((-1,1))
```

	y	x	val
0	0	0	0.0
1	0	1	0.0
2	0	2	0.0
3	0	3	0.0
4	0	4	0.0
...	...	...	...
397795	467	845	0.0
397796	467	846	0.0
397797	467	847	0.0
397798	467	848	0.0
397799	467	849	0.0

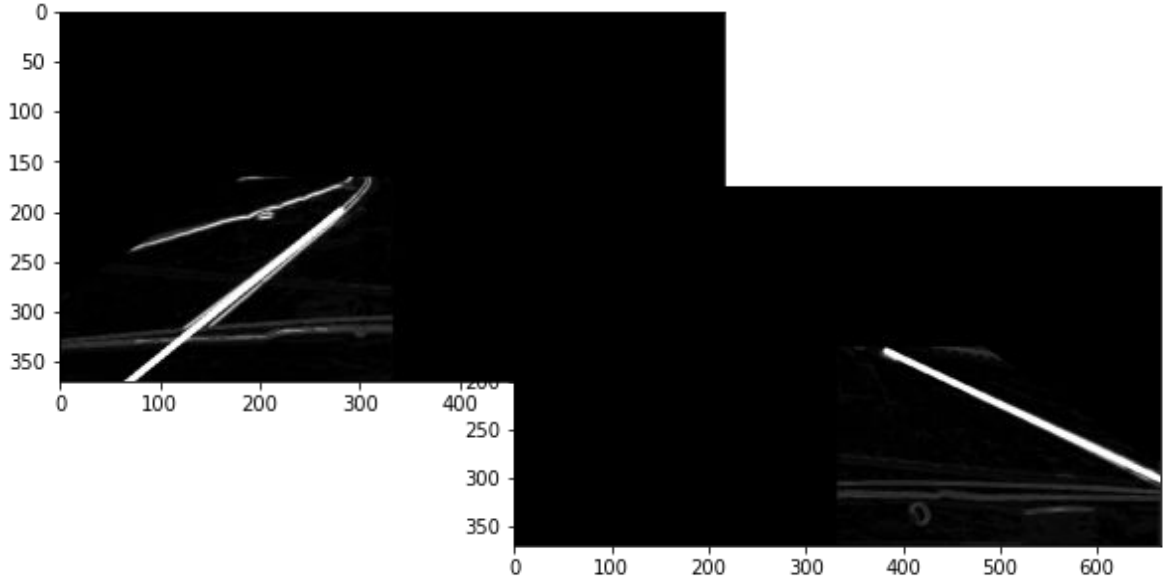
⬆️ 임계값 설정 전

	index	y	x	val
0	182333	214	433	0.717647
1	182334	214	434	0.717647
2	182335	214	435	0.705882
3	182371	214	471	0.745098
4	182372	214	472	0.803922
...	...	...	...	...
1673	370280	435	530	0.737255
1674	370281	435	531	0.737255
1675	370282	435	532	0.721569
1676	371128	436	528	0.717647
1677	371129	436	529	0.717647

⬆️ 임계값 설정 후

Code Review

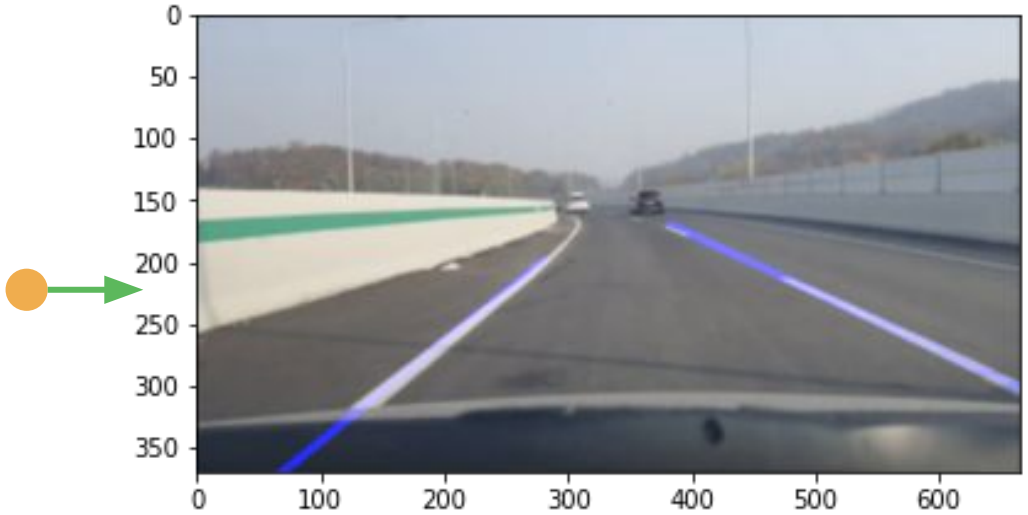
- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression



⬆️ RANSAC Regression & 선 그리기

Code Review

- Convert Gray
- Gaussian Blur
- Sobel Filter
- Data Preprocessing
- RANSAC Regression

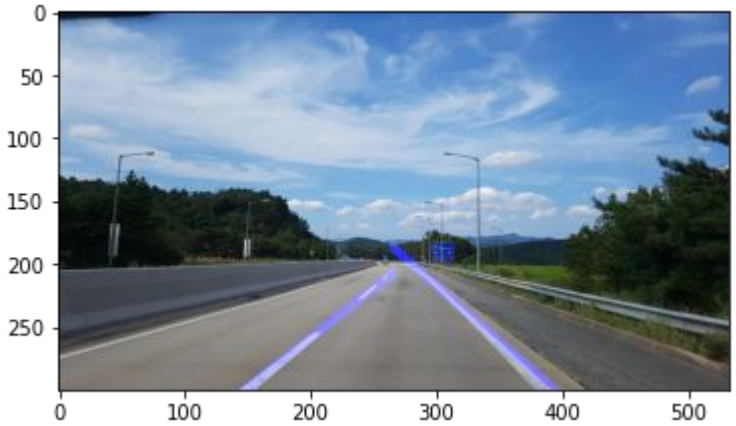


⬆️ 원본 이미지에 라인 추가

Result

Experiment

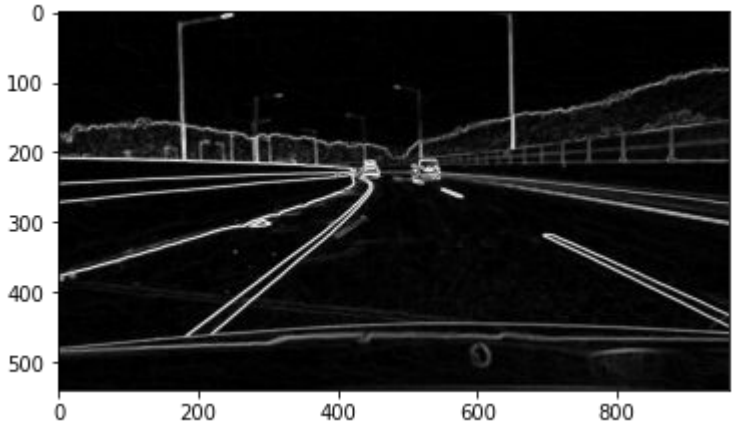
⬆️ 나쁜 예



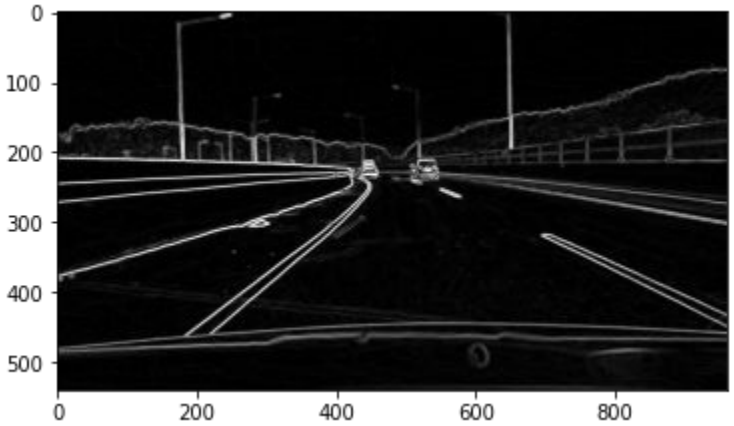
⬆️ 좋은 예

앞의 차가 차선을 가리는 문제.
차가 빼곡한 도로에서는 확실히 예측이 잘 되지않을 때가 있었다.

Experiment



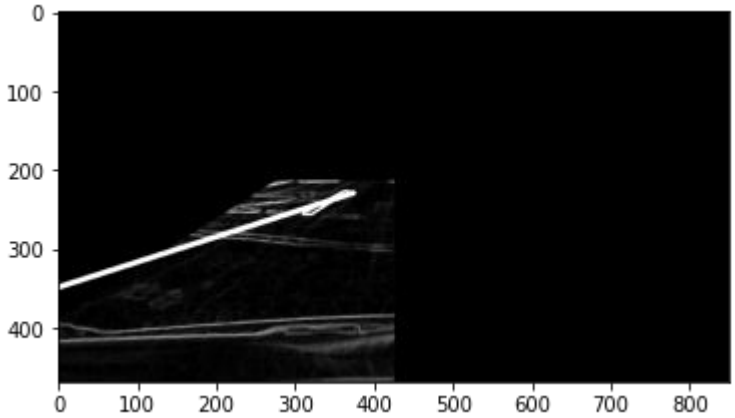
⬆️ abs(sobel_x) + abs(sobel_y)



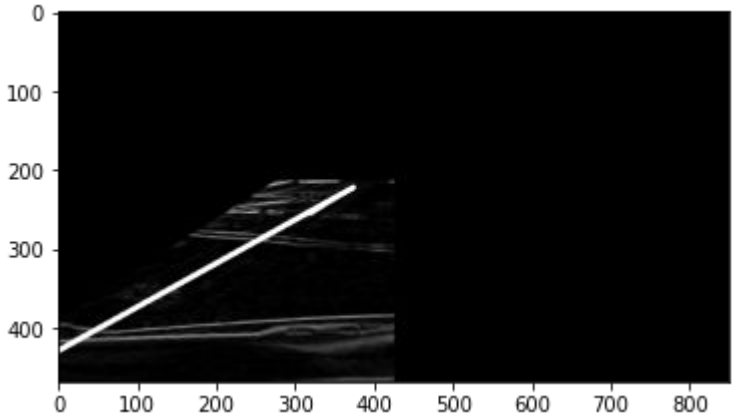
⬆️ sqrt(sobel_x^2+sobel_y^2)

계산 속도는 당연 전자가 더 낫다.
정확도는,,, 잘 모르겠다.

Experiment



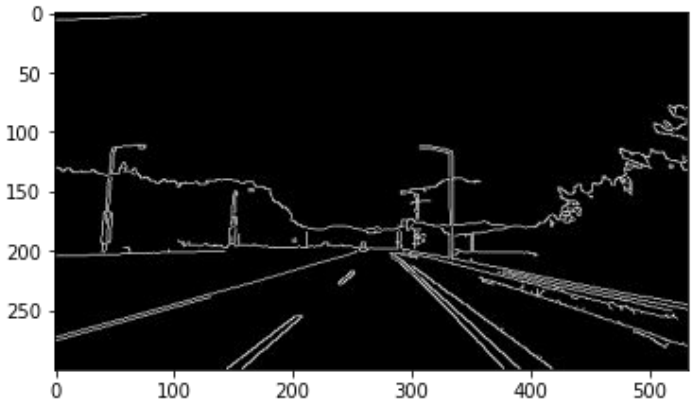
⬆️ abs(sobel_x) + abs(sobel_y)



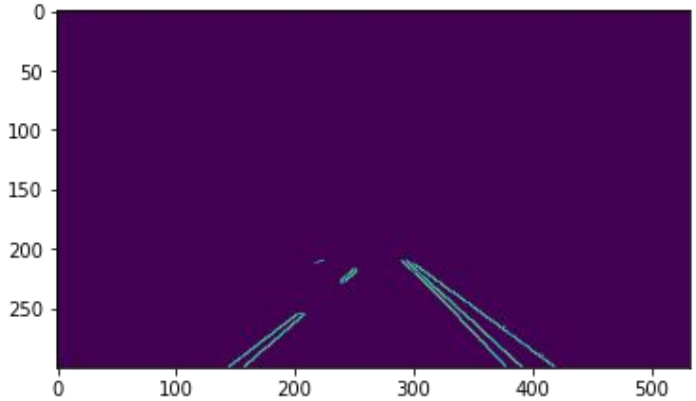
⬆️ abs(sobel_y)

방향성을 감지할 수 있는 필터 특성을 이용하여 실험해봤다.
항상은 아니지만 특정 몇몇 상황에서는 위와 같이 더 나은 성능을 보였다.
이는 임계값 조절도 크게 연관 있다고 생각한다.

Experiment



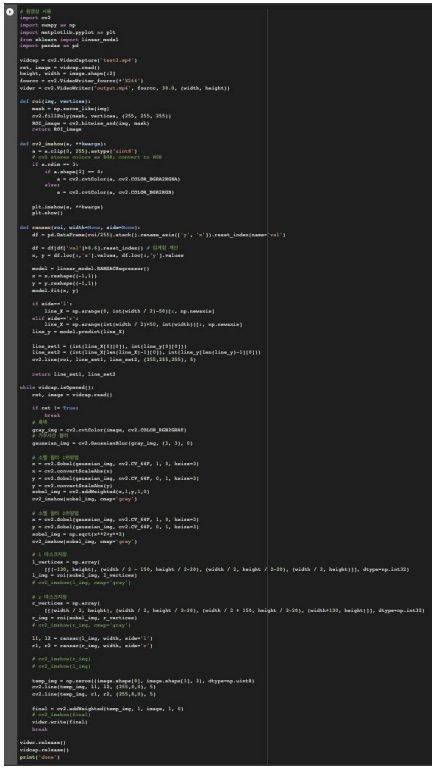
⬆️ Canny



⬆️ Cutting & drop_line

Canny 알고리즘을 사용하여 선을 따고, (라인이 매우 깔끔하게 나온다)
Hough transform을 이용하여 선형 데이터로 변환한 후,
차선 라인에 맞지않는 각도의 선을 제거하면 위와 같이 특정 각도의 선만 검출 할 수 있다.

Fin



허프변환이 뭔가요?

- https://wkdtjsgur100.github.io/Hough-Transform/

Sklearn에서 RANSAC사용 공식문서

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html

opencv 라인검출 알고리즘 공식문서

- https://opencv-python.readthedocs.io/en/latest/doc/13.imageGradient/imageGradient.html

참고 포스팅

- https://m.blog.naver.com/windowsub0406/220894645729

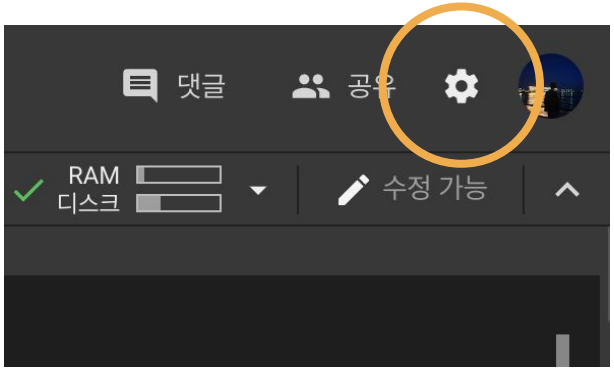
영채의 1시간짜리 코드리뷰 (#지루함)

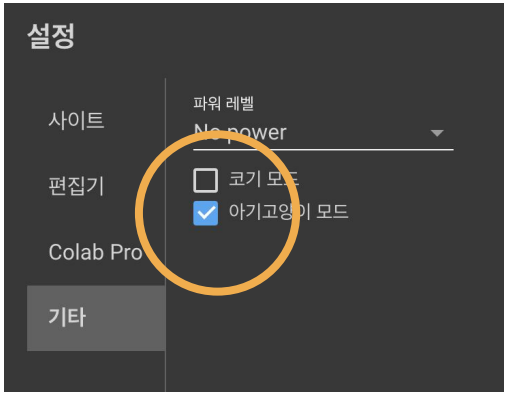
- https://www.youtube.com/watch?v=XeXlZev2FxY&feature=youtu.be

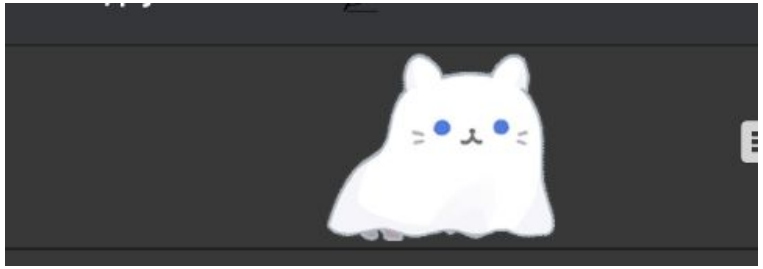
코드 완성본

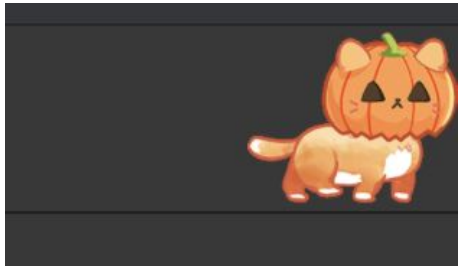
- https://colab.research.google.com/drive/1AfxHvdBThJNa3Bzn3S__nCaPf8h-IdJv?usp=sharing

Fin


댓글
공유
RAM
디스크
수정 가능


설정
사이트
편집기
Colab Pro
기타
파워 레벨
No power
코기 모드
아기고양이 모드





발표 끝

트리케라톱스

김윤지 김해리 나영채 이현동 진현영