

Image Processing & Vision: Final Exam Report

중앙대학교 예술공학대학

20190807 민정우

Computer Vision & Image Processing

Computer Vision은 카메라 등의 센서로 얻은 시각 데이터를 컴퓨터를 통해 처리하는 기술이다. Computer Vision에서 처리하는 디지털 이미지는 색의 정보를 숫자로 표현한 픽셀의 집합으로, Computer Vision은 이런 숫자의 집합에서 실제로 사람이 이미지에서 얻는 의미를 찾아내는 것을 목표로 한다.

Image Processing은 알고리즘을 이용해 디지털 이미지를 수학적으로 처리하는 기법이다. Image Processing은 이미지를 향상시키거나 이미지에서 의미 있는 정보를 추출하며, Computer Vision에서 더 효과적으로 결과를 얻기 위해 사용된다.

Digital Images

디지털 이미지는 픽셀의 2차원 배열이며, 공간 좌표 x, y 에 대해 구간 $([1, \text{Width}], [1, \text{Height}])$ 에서 정의된 2차원 함수 $I(x, y)$ 로 정의할 수 있다.

디지털 이미지는 좌표 상의 픽셀 $I(x, y)$ 의 정보에 따라 다음과 같이 분류된다.

- Binary Image : 이미지의 픽셀이 0과 1로만 정의되며, 검은색과 흰색 정보만을 담는다.
- Grayscale Image : 이미지의 픽셀이 $[0, 255]$ 의 범위를 가지며, 흑백의 명암 정보를 가지고 있다.
- Color Image : 이미지의 픽셀이 $[0, 255]$ 범위의 배열 3개로 구성되며, 각각의 배열은 R, G, B 3개의 색상 정보를 가지고 있다.

Quantization & Sampling

아날로그 이미지는 연속적이고 무한한 집합이기 때문에 이를 그대로 사용할 수 없다. 따라서 아날로그 이미지에 Sampling과 Quantization을 거쳐 이산적이고 유한한 집합의 디지털 이미지로 변환한다.

- Sampling : 연속 신호에서 샘플을 추출하는 과정이다. 일반적으로 원본 신호의 최대 주파수의 2배 이상에서 정기적으로 Sampling하면 원본 신호와 유사하게 추출할 수 있다. Sample의 주기가 이보다 낮으면 정보가 손실돼 각기 다른 신호를 구별하지 못하는 Aliasing 현상이 발생하는데, 이 현상을 해결하기 위해 sample을 더 추출하는 Oversampling이나 Aliasing으로 발생한 정보를 제거

하는 Smoothing을 통해 Anti-Aliasing 작업을 수행한다.

- Quantization : 연속적인 집합의 입력 값을 가산 가능한 집합의 출력 값으로 매핑하는 과정이다. 아날로그 이미지의 임의 구간 (p, q) 의 연속된 색상값 $i(p, q)$ 을 n 개의 비트를 사용해 구간 $[0, M]$ 으로 매핑하는 식은 다음과 같다.

$$i(p, q) \rightarrow \left\lfloor \frac{i(p, q)}{M} (2^n - 1) + 0.5 \right\rfloor$$

Color Filter Arrays

Color Filter Arrays(CFA)는 이미지 센서의 픽셀 센서 위에 배치되는 color filter의 배열로 RGB 영역의 강도에 대한 정보를 수집한다. 시신경의 원추 세포는 S, M, L의 세 가지 타입으로 서로 다른 파장대를 감지하는데, 녹색 계열을 감지하는 세포가 가장 많기 때문에 CFA의 모자이크는 녹색 필터를 특히 많이 배치한다. CFA로 추출한 색상 이미지는 누락된 부분을 보간하는 Demosaicing 작업을 수행해 실제 원본과 유사한 색상을 갖게 된다.

Image Transformation

이미지는 행과 열의 2차원 함수로 표현되며, 2차원 행렬로 변환할 수 있다. 즉, 이미지의 변환은 행렬의 곱으로 표현할 수 있다. 그 중 Scaling과 Rotation은 행렬의 곱으로 계산하나 Translate는 행렬의 덧셈으로 계산하기 때문에 Homogeneous coordinates를 이용해 모든 변환을 행렬의 곱으로 표현한다.

$$\text{- Translation : } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{- Scaling : } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{- Rotation : } \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine Transformation

Affine Transformation은 기하학적 변환으로 직선과 평행성을 보존하나 유clidean 거리와 각도를 유지하지 않는다. Affine Transformation은 Translation, Scaling, Rotation, Reflection, Shear 변환을 포함한다.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Projective Transformation(Homography)

Projective Transformation은 3D 공간에서 2D 공간으로 투영하거나 서로 다른 두 평면 간의 매핑 관계를 모델링하는 변환이다.

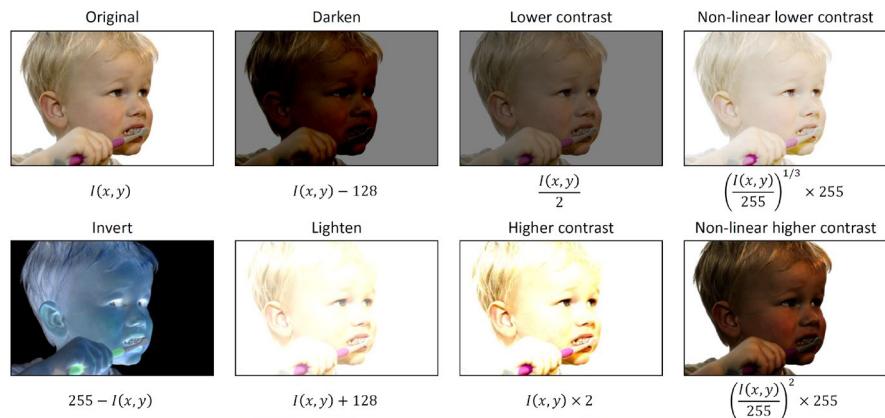
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image Filtering

Image Filtering은 원본 이미지를 수식을 이용해 이미지 픽셀 행렬을 변환해 이미지를 변형하는 기법이다.

Point operation

Point operation은 각각의 픽셀에 대해서 일괄적으로 연산하는 방법이다. 주위 픽셀에 영향을 받지 않는다.



Neighborhood operation

Neighborhood operation은 임의의 픽셀에 대해 그 주변의 픽셀을 이용해 연산하는 방법이다. 크게 Linear Filter와 Non-Linear Filter로 나뉜다.

Linear Filters

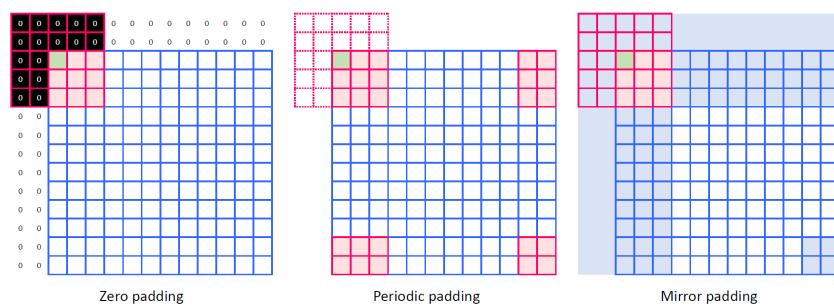
Linear Filters는 filter(또는 kernel)을 통해 이미지에 convolution 연산을 수행해 선형적으로 변환 한다. 이미지 $I(x, y)$ 와 필터 $F(x, y)$ 에 대해 이미지 내의 픽셀 (i, j) 에 다음 연산을 적용한다. (k 는 필터 $F(x, y)$ 의 중심으로부터 필터의 변까지의 거리)

$$I'(x, y) = \sum_{i=-k}^k \sum_{j=-k}^k F(i, j)I(x + i, y + j)$$



이미지를 filter로 filtering하면 가장자리 영역에서 값이 정의되지 않아 계산이 불가능한 픽셀이 생기는 Boundary effects가 발생한다. Boundary effects가 발생하는 영역에 대해 일반적으로 처리되는 과정은 다음과 같다.

- Ignore : 계산되지 않는 영역을 버린다. 이 경우 변환되는 이미지는 원본 이미지보다 크기가 작아진다.
- Zero padding : 정의되지 않은 값을 0으로 치환한다.
- Periodic padding : 동일한 이미지를 가장자리에 연속시키고 값을 계산한다.
- Mirror padding : 계산하는 가장자리의 값을 복사해서 계산한다.



Linear Filters를 통해 Shifting, Sharpening, Smoothing 변환을 수행할 수 있다.

- Shifting

Original image * Filter = Result
(Shift left by 1 pixel)

- Sharpening

Original image * $\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right] - \frac{1}{9} \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right] = \text{Result}$
(Sharpening)

- Smoothing

Original image * $\frac{1}{9} \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right] = \text{Result}$
(Blur with a box filter)

한편, Smoothing 연산에 사용되는 Box Filter는 픽셀을 주위 픽셀을 동일하게 반영하기 때문에 가운데 픽셀이 이웃 픽셀과 차이가 커져 부자연스러운 결과를 도출한다. Box Filter를 통한 Smoothing 변환의 부자연스러움을 개선하기 위해 다음 필터를 사용한다.

- Circular Pillbox

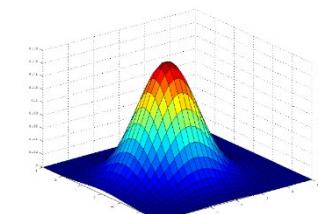
Circular Pillbox는 이미지를 원으로 Filtering하며 2차원 pillbox filter 함수는 다음과 같다.

$$f(x, y) = \frac{1}{\pi r^2} \begin{cases} 1, & \text{if } x^2 + y^2 \leq r^2 \\ 0, & \text{otherwise} \end{cases}$$

- Gaussian Filter

Gaussian Filter는 2차원 Gaussian 함수를 통해 거리의 가중치를 반영한다. 연속적인 2차원 Gaussian 함수는 다음과 같다.

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



Filter의 흐림 효과는 σ (표준 편차)가 클수록 강해진다. 그러나 위 함수는 연속적인 값을 가지므로 함수를 Filter에 그대로 적용할 경우 Filter의 요소의 합이 1이 되지 않아 이미지가 밝거나 어두워진다. 따라서 정규화를 통해 Filter의 요소의 합이 1이 되도록 변환한 후 사용한다.

Separability

크기가 $n \times n$ 인 이미지를 크기가 $m \times m$ 인 filter로 filtering하는 데 필요한 연산의 횟수는 $n^2 \times m^2$ 이다.

한편, Smoothing 변환에 사용되는 2차원 Gaussian 함수는 각 축에 대한 1차원 함수로 분해 가능하다.

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right)$$

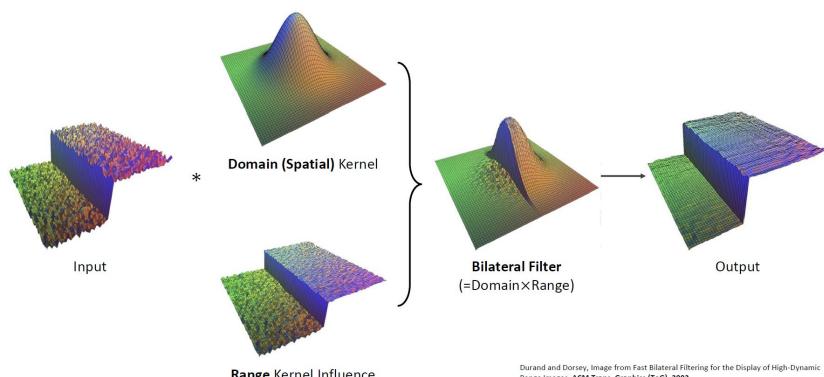
이 경우, filter는 각 축에 대해 크기가 m 인 1차원 filter 2개로 분해되며, 이 때 필요한 연산의 횟수는 $2 \times n^2 \times m$ 이다. 따라서 2차원 Gaussian Filter는 각 축에 대한 1차원 Gaussian Filter로 분해해 연산하는 것이 더 효율적이다.

Non-linear Filters

Non-linear Filters에 속하는 filter는 다음과 같다.

- Median Filter : 연산하는 픽셀을 filter에 속한 값 중 중간값으로 변환한다. 평균에서 크게 벗어나는 최댓값과 최솟값을 제거하기 때문에 이미지의 노이즈를 효과적으로 제거할 수 있다.
- Bilateral Filter : Gaussian Filter를 응용하여 중심으로부터의 거리와 주변 픽셀과의 유사도를 함께 계산한다. 이미지 함수 $I(x, y)$ 의 중심 픽셀 (i, j) 와 비교 픽셀 (k, l) 에 대해 Bilateral filter 함수는 다음과 같다.

$$B_{\sigma_d, \sigma_r}(i, j, k, l) = e^{-\frac{(k^2+l^2)}{2\sigma_d^2}} e^{-\frac{(I(i+k, j+l) - I(i, j))^2}{2\sigma_r^2}}$$



Durand and Dorsey, Image from Fast Bilateral Filtering for the Display of High-Dynamic Range Images, ACM Trans. Graphics (ToG), 2002

일반적인 Box Filter나 Gaussian Filter로 Smoothing 변환을 처리할 경우 이미지의 가장자리가 뭉개지는데, Bilateral Filter를 적용하면 이미지의 가장자리를 살리면서 Smoothing 변환을 수행할 수 있다.

- ReLU : ReLU 함수는 0보다 작은 값을 0으로 변환하는 함수로 다음과 같다.

$$f(x) = \max(0, x)$$

ReLU 함수는 Convolutional Neural Networks(CNN)에 주로 사용되며 CNN의 각 convolution filter를 거친 후 발생하는 음수값을 제거하기 위해 사용한다.

Template Matching

Template Matching은 전체 이미지에서 부분 이미지인 template과 유사한 영역을 탐색하는 기법이다. Template Matching은 Filtering과 유사하게 전체 이미지를 순회하면서 임의의 구간과 Template의 correlation 연산을 수행한다.

Template Matching은 연산하기 쉽고 noise에 비교적 강하며 translation에 강하다는 장점이 있으나, rotation과 scale 및 밝기 변화에 민감하다. 그 중 scale 변화에 민감한 문제를 해결하기 위해 Gaussian Filter로 다양한 사이즈의 template의 Image Pyramid를 제작해 연산을 수행한다.

Local Feature Detection

기존의 Template Matching은 template을 통해 오브젝트를 감지하므로 오브젝트의 변화에 대응하기 어렵다는 단점이 있다. Local Feature Detection은 오브젝트를 탐색할 때 모든 변화에 대응할 수 있도록 local 영역에서 오브젝트의 특징을 찾는다.

좋은 Feature는 다음 특징을 갖는다.

- Locality : 해당 영역에서의 occlusion과 clutter에 강해야 한다.
- Accurate : 해당 영역을 정확하게 표현해야 한다.
- Robustness : 노이즈, 블러, 압축 등에 크게 영향을 받지 않아야 한다.
- Distinctiveness : 다른 Feature와 구분이 되어야 한다.
- Efficiency : 단시간에 효율적으로 검출할 수 있어야 한다.

일반적으로 감지하는 local feature에는 edges와 corners가 있다.

Edge Detection

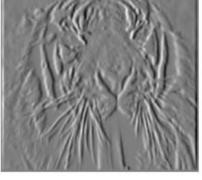
Edge는 이미지의 두 영역 사이에서 발생하는 경계를 의미한다. 따라서 Edge를 감지하기 위해서 이미지의 밝기의 순간 변화량을 추적한다.

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h}$$

디지털 이미지의 픽셀 변화량의 최소단위는 1이며, 순간 변화량은 다음 3가지 방법으로 근사화 할 수 있다.

$\frac{df}{dx} = f(x+1) - f(x)$	$\frac{df}{dx} = f(x) - f(x-1)$	$\frac{df}{dx} = \frac{f(x+1) - f(x-1)}{2}$
Forward Difference	Backward Difference	Central Difference

1차 미분을 통해 각 축에 대한 픽셀의 변화량을 계산한 뒤 이를 통해 Image Gradient를 구한다. 이렇게 구한 Gradient를 통해 변화의 크기와 변화의 방향을 구할 수 있다.

$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$	$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$	$\ \nabla f\ = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$
Image Gradient	Gradient direction	Gradient magnitude
 Original image	 Gradient direction	 Gradient magnitude
 The gradient in the x-direction	 The gradient in the y-direction	

Derivative of Gaussian

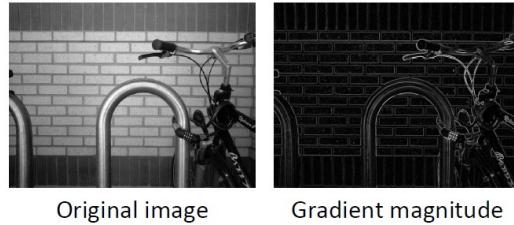
원본 이미지에서 직접 Gradient를 계산하면 모든 변화에 대해 민감하게 반응하기 때문에 노이즈가 발생한다. Derivative of Gaussian(DoG) filter는 Gaussian 함수를 통해 이미지를 Smoothing해 노이즈를 필터링하고 변화량을 추적해 Edge를 검출한다.

Sobel Edge Detector

Sobel Edge Detector는 Smoothing을 적용하면서 Central difference를 통해 gradient를 계산한다.

$$G_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * ([-1 \ 0 \ 1] * I) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I$$

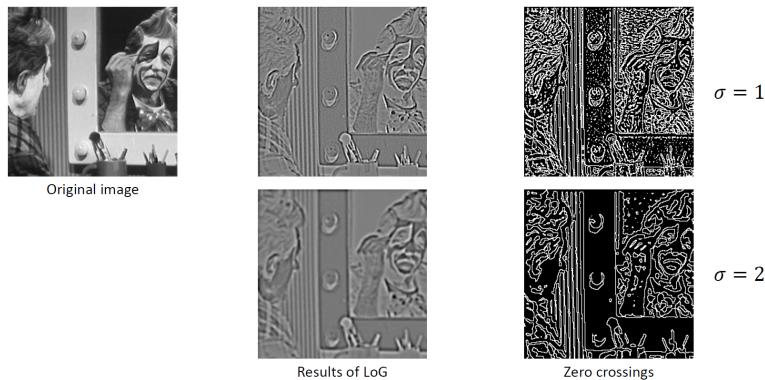
$$G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * ([1 \ 2 \ 1] * I) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$



Laplacian of Gaussian

Laplacian of Gaussian(LoG)는 노이즈를 제거하기 위해 Gaussian 함수를 사용해 이미지를 smoothing하고 Laplacian 연산자를 통해 수평 및 수직 방향으로의 2차 미분을 구해 이미지의 Edge를 추출한다.

$$\nabla^2 G(x, y; \sigma) = \left[\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right] G(x, y; \sigma)$$

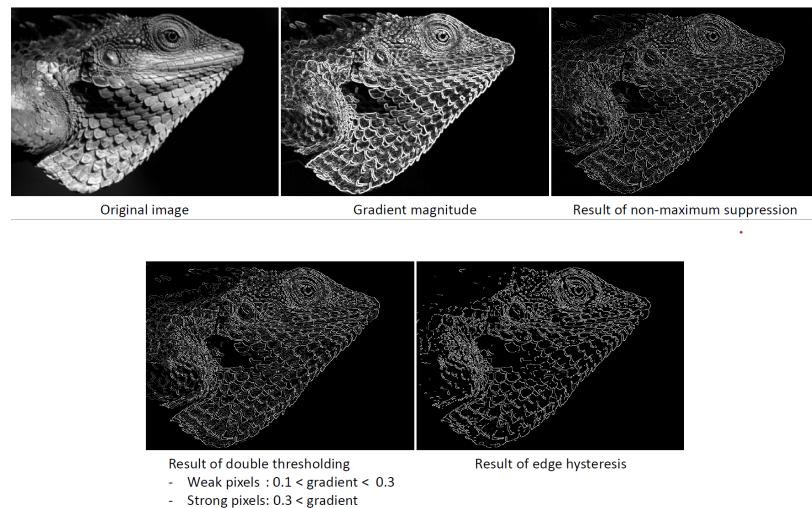


Canny Edge Detector

Canny Edge Detector는 다음 과정을 거쳐 Edge를 검출한다.

- 이미지에 DoG filter를 적용한다.
- 검출된 Gradient의 magnitude와 direction을 계산한다.
- Non-Maximum Suppression(NMS)를 적용한다. NMS는 하나의 gradient direction에 위치한 gradient의 magnitude가 최대인 픽셀만을 검출해 edge가 여러 개의 픽셀로 중첩되는 현상을 제거하는 과정이다.

- NMS 과정을 거쳐 검출한 Edge에 대해 두 개의 임계값을 사용해 끊어진 Edge를 연결한다.



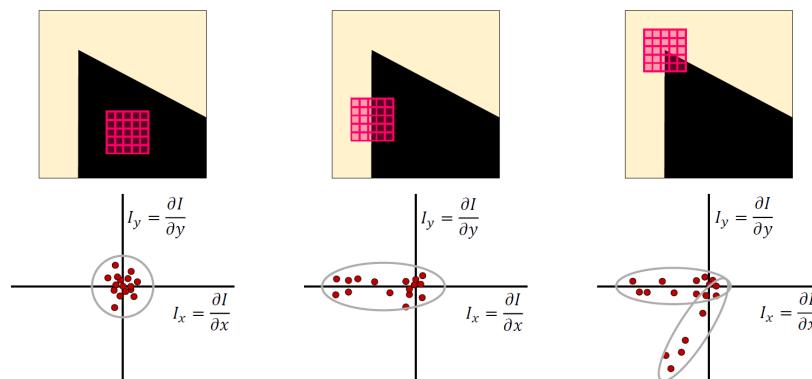
Corner Detection

Corner는 로컬 영역을 모든 방향으로 움직였을 때 값의 변화가 생기는 지점을 의미한다. 따라서 Corner를 검출하기 위해서는 작은 영역을 모든 방향으로 움직여서 변화가 가장 큰 지점을 찾아야 한다.

Harris Corner Detection

Harris Corner Detection은 다음 과정을 거쳐 corner를 검출한다.

- 작은 영역의 Image Gradient를 계산한다.



- covariance matrix를 계산한다. 로컬 영역 P 의 임의의 픽셀 p 와 이미지의 각 축에 대한 gradient 변화량 I_x, I_y 에 대해 covariance matrix는 다음과 같다.

$$M = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

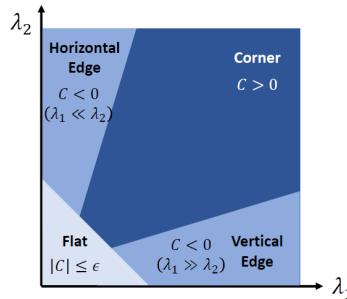
- covariance matrix의 고유값과 고유벡터를 계산한다. 두 개의 고유값은 변화하는 크기를 의미하고 각 고유값에 대응하는 고유벡터는 변화하는 방향을 의미한다.

- 임계값과 고유값을 통해 corner를 검출한다. 두 개의 고유값이 충분히 크다면 Corner, 하나의 고유값이 크고 다른 하나의 고유값이 작다면 Edge, 두 고유값 모두 0에 가까운 값을 가지면 Flat region으로 간주한다.

그러나 고유값과 고유벡터를 구하는 것은 계산적으로 복잡하기 때문에, 다음 공식을 통해 Corner를 검출한다. 이 때 α 는 $0.04 \leq \alpha \leq 0.06$ 의 범위를 가진다.

$$C = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 = \det(M) - \alpha \cdot \text{tr}(M)^2$$

C 의 절대값이 임계값 ϵ 보다 작으면 Flat region, C 가 0보다 작으면 Edge, 0보다 크면 Corner로 간주한다.



Local Features

Edge 및 Corner는 픽셀 단위의 Features이다. 그러나 이미지는 단일 픽셀보다 Local 영역에서 더 의미 있는 정보를 가지고 있기 때문에, Local 단위의 Features를 추출하는 것이 더 유용하다.

Local 영역의 유사도를 검출하는 방법으로 다음이 있다.

- Image Intensity : 영역의 각 픽셀의 밝기를 직접적으로 비교한다. 영역이 불변하는 경우 가장 정확하게 검출할 수 있는 방법이나 translate를 제외한 모든 변화에 대해 민감하다.
- Image Gradients & Edge : 영역의 Gradients나 Edges를 비교한다. 절대적인 밝기 변화에 상관 없이 검출할 수 있으나 오브젝트의 형상이 변화하는 경우에 민감하다.
- Color Histogram : 영역의 Color Histogram을 비교한다. scale과 rotation의 변화에 강하나 영역

의 모양과 질감을 무시하기 때문에 오차가 발생할 수 있다.

- Spatial Histograms : 영역의 구간별 Color Histogram을 계산해 비교한다. Color Histogram과 비교해 영역의 모양과 질감을 어느 정도 반영하나, rotation 변화에 민감하다.

기존의 이미지 유사도를 검출하는 방법은 여러 변화에 민감하다. 따라서 모든 변화에 강인한 Features를 검출하는 과정이 필요하며, 이런 features는 다음과 같은 특징을 가져야 한다.

- Locality : 해당 영역에서의 occlusion과 clutter에 강해야 한다.

- Distinctiveness : 각각의 Features가 독립적이어야 한다.

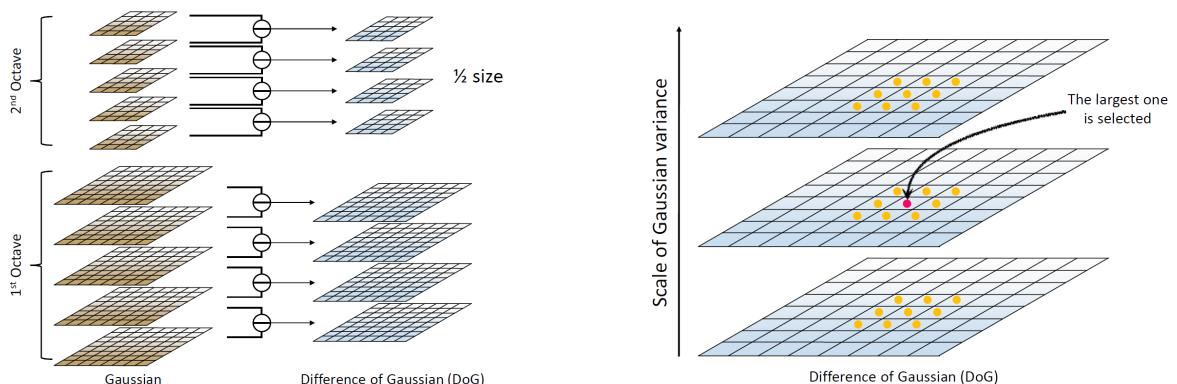
- Quantity : Features를 숫자로 정형화할 수 있어야 한다.

- Efficiency : 단시간에 효율적으로 검출할 수 있어야 한다.

SIFT

Scale-Invariant Feature Transform(SIFT)은 이미지의 이동, 회전, 크기 및 밝기의 변화에 불변하는 Features를 추출하는 알고리즘이다. SIFT 알고리즘은 다음 단계를 거쳐 동작한다.

- Multi-scale extrema detection : 크기 변화에 불변하는 특징점을 찾기 위해 Scale Space를 제작 한다. 이미지의 크기를 단계적으로 줄인 각 옥타브마다 여러 표준 편차를 가진 Gaussian Filter를 적용해 여러 개의 블러 이미지를 제작한다. 이후 동일 옥타브의 인접한 이미지들에 Difference of Gaussian(DoG) 연산을 적용해 DoG 이미지들을 출력한다. 이후 다른 크기의 이미지들을 크기를 일치시킨 후 3개의 3x3 크기의 윈도우에서 픽셀들을 비교해 극대값 또는 극소값을 Keypoint로 선정한다.

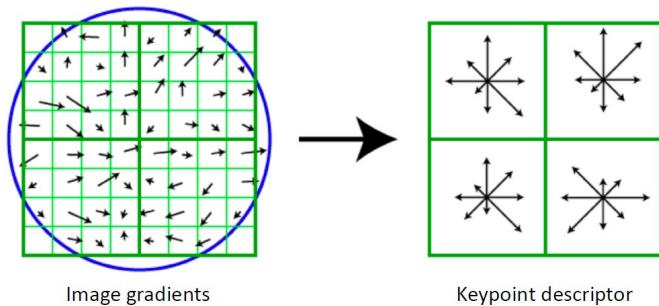


- Keypoint localization : 임계값보다 크기가 작거나 covariance matrix의 고유값을 통해 edge나 corner에서 검출되지 않은 것으로 확인된 불안정한 keypoints를 제거한다.

- Orientation assignment : Keypoints가 회전 변화에 불변하도록 local gradient의 방향을 계산한 후 방향 히스토그램을 제작한다. 이후 방향 히스토그램에서 가장 높은 값을 갖는 방향을 keypoint

의 방향으로 할당한다.

- Keypoint descriptor : 하나의 keypoint에 대해 16×16 칸의 정보를 16 개의 4×4 의 window로 표현하고, 이 window 내에서 gradient의 방향과 크기를 8개 방향으로 묶어 window의 gradient 크기와 방향을 구해 keypoint의 descriptor를 생성한다. 이렇게 생성된 keypoint descriptor는 밝기 변화에 영향을 받지 않도록 단위 길이로 정규화를 거친다.



Model Fitting

이미지에서 Features를 추출했다면 이 Features를 이용해 두 이미지에서 공통된 부분을 찾을 수 있다. 그러나 두 이미지는 서로 다르기 때문에 정확한 매칭을 찾을 수 없으며, 최대한 근접한 매칭을 찾아야 한다.

RANSAC

Random Sample Consensus(RANSAC) 알고리즘은 무작위로 데이터를 뽑아 가장 많은 데이터가 만족하는 모델을 선택하는 알고리즘이다. RANSAC 알고리즘은 다음과 같이 동작한다.

- 무작위로 샘플 데이터를 선택한다. 선택한 샘플 데이터들을 inlier로 가정하고 모델을 생성한다.
- 해당 모델과 임계값의 범위 내에서 일치하는 데이터의 수를 센다.
- 위 과정을 충분히 반복해 inlier를 최대로 가지는 모델을 선택한다.

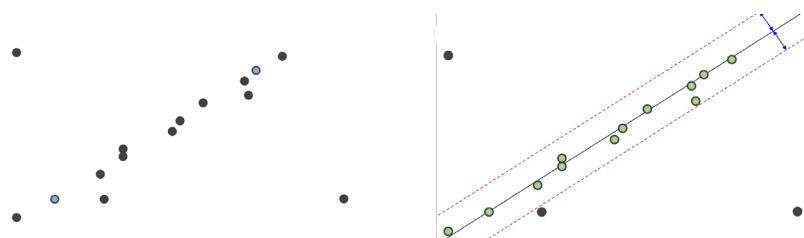
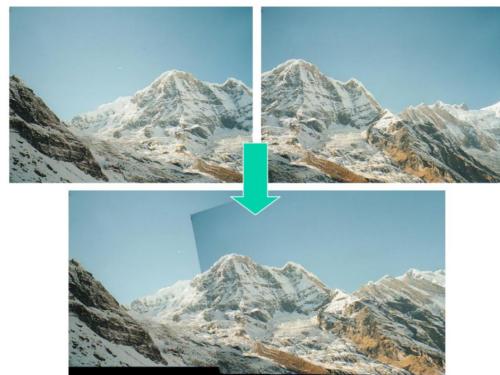


Image Stitching

SIFT 알고리즘으로 이미지의 Features를 추출해 매칭하고 매칭된 Features와 RANSAC 알고리즘으로 변환 행렬을 구하면 두 이미지를 이어붙일 수 있다.



Machine Learning in Computer Vision

Machine Learning(ML)은 데이터를 이용해 컴퓨터가 스스로 학습하여 알고리즘 모델을 구축하는 방법이다. ML은 Training과 Testing 과정을 통해 모델을 추측한다.

- Training : 훈련에 사용할 훈련 데이터 세트를 제공하고 훈련 세트의 예측 오류를 최소화해 예측 함수를 추정한다.
- Testing : Training으로 추정한 함수를 테스트 데이터를 통해 평가한다.

ML을 사용해 해결하는 문제는 데이터의 구성과 데이터의 성질에 따라 다음과 같이 분류할 수 있다.

	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality Reduction

- Classification : 입력 개체와 원하는 출력 값의 쌍 레이블에서 함수를 추론해 훈련된 예측 함수 이미지에서 원하는 출력을 얻는다.
- Clustering : 입력 데이터에서 사람의 개입 없이 패턴이나 데이터 그룹을 발견한다.
- Regression : 입력 변수에서 연속 출력 변수로의 쌍 레이블에서 변환 함수를 예측한다.
- Dimensionality Reduction : 데이터를 고차원 공간에서 저차원 공간으로 변환한다.

이 중, Supervised Learning은 입력값과 출력값을 같이 제공하는 학습 방법으로, 다음 과정을 거

친다.

- 대규모의 데이터셋을 수집한다.
- Testing에 사용할 일부 데이터를 제외한 나머지 데이터를 통해 Training한다.
- 일부 데이터를 통해 예측 함수를 Testing해 평가한다.

Image Classification

Image Classification은 입력 이미지와 분류 카테고리의 쌍 레이블을 통해 학습해 컴퓨터 스스로 이미지를 분류할 수 있는 것을 목표로 한다. 이 때 이미지를 학습하는 데 문제가 발생하는 부분은 다음과 같다.

- 밝기의 변화
- 크기의 변화
- 클래스 내의 변화
- 세분화된 카테고리 분류
- 카메라의 방향 변화
- 오브젝트가 가려짐
- 오브젝트의 상태 변화
- 주위 배경과 혼합

현재 사용되는 Machine Learning 모델은 이런 문제를 해결하고 높은 수준의 정확도를 보여주고 있다.

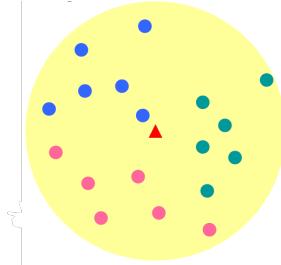
Classifiers는 다음과 같이 분류할 수 있다.

- Parametric classifiers : 학습 데이터를 통해 모델을 제작해 데이터를 분류한다.
- Non-parametric classifiers : 학습 데이터 자체를 모델로 데이터를 분류한다.

Nearest Neighbor

Nearest Neighbor는 주어진 집합에서 주어진 데이터와 가장 유사한 데이터를 찾는 방법이다. Nearest Neighbor는 다음 과정으로 진행한다.

- Training Stage : 주어진 모든 데이터와 레이블을 저장한다.
- Testing Stage : 가장 유사한 학습 이미지의 레이블을 예측한다.



Nearest Neighbor에서 거리를 계산하는 일반적인 방법은 다음 두 가지가 있다. 여기서 I_{test} 는 테스트 이미지, I_{train} 는 학습 이미지, W, H 는 이미지의 수평, 수직 크기다.

- L_1 distance(l_1 norm; Manhattan distance) : 두 점의 차이의 절대값을 계산한다.

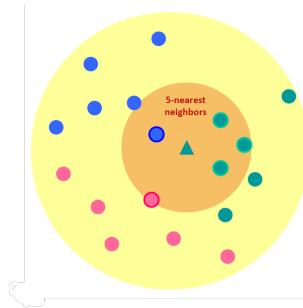
$$d_1(I_{test}, I_{train}) = \sum_{i=1}^W \sum_{j=1}^H |I_{test}(i, j) - I_{train}(i, j)|$$

- L_2 distance(l_2 norm; Euclidean distance) : 두 점의 직선 거리를 계산한다.

$$d_2(I_{test}, I_{train}) = \sqrt{\sum_{i=1}^W \sum_{j=1}^H (I_{test}(i, j) - I_{train}(i, j))^2}$$

K-Nearest Neighbors

인접한 한 개의 이웃만을 통해 분류를 시도하면 노이즈가 발생할 가능성이 높다. K-Nearest Neighbors(K-NN)은 기존의 Nearest Neighbors에서 나아가 K개의 인접한 이웃을 찾는 방법이다.



이 때, 가장 적합한 K를 찾기 위해 데이터 세트를 train, validation, test 데이터로 분리해서 학습 한다. Train 데이터에서 학습하고 validation 데이터에서 적합한 K를 찾고, test 데이터로 테스트한다.

Train	Validation	Test
-------	------------	------

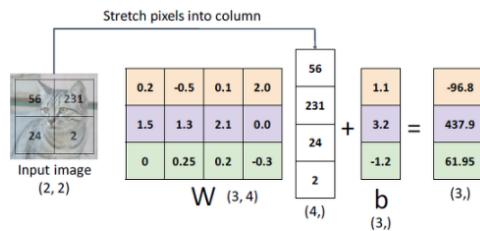
K-NN은 학습에 사용된 이미지를 전부 저장해야 하기 때문에 메모리의 부담이 크고, 테스트 비용이 크다. 이미지의 특징이 아닌 픽셀값 자체를 사용하기 때문에 K-NN에서 검출한 유사성이 실제 사람이 지각적 유사성과의 차이가 발생할 수 있다.

Linear Classifier with Images

Linear Classifier는 선형 함수를 통해 이미지를 분류하는 것으로, 다음 식으로 나타낼 수 있다.

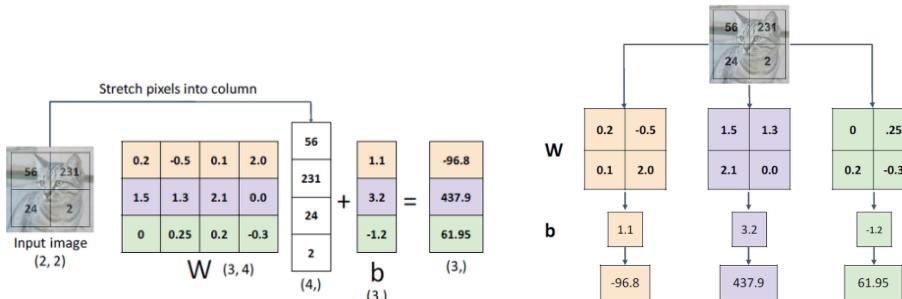
$$f(x, W) = Wx + b$$

여기서 x 는 입력 이미지 행렬을 변환한 1차원 벡터, W 는 weight를 나타내는 2차원 행렬, b 는 bias vector를 나타내는 1차원 벡터로, W 와 b 는 임의의 값을 가지며 학습을 거치면서 그 값이 변화한다.

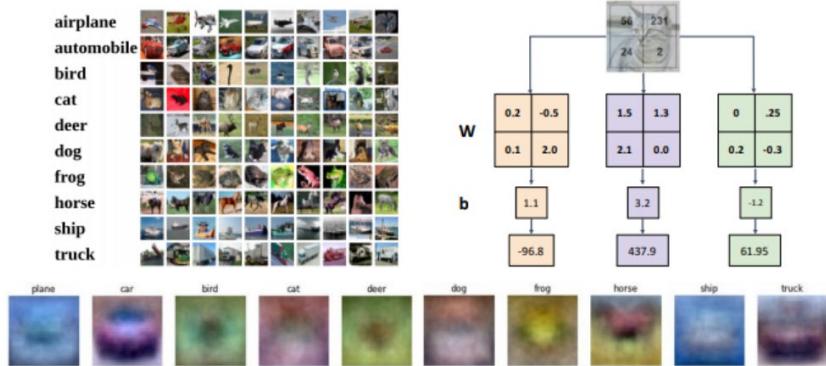


Linear Classifier는 다음 세 관점에서 해석할 수 있다.

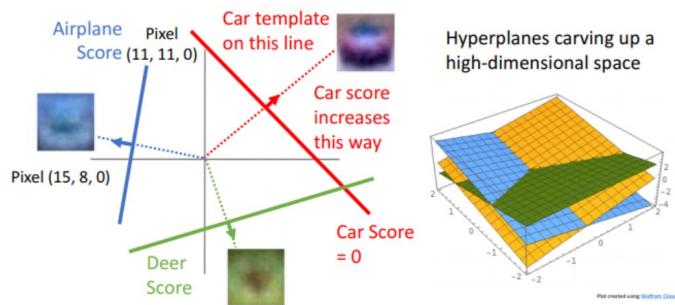
- Algebraic Viewpoint : Algebraic 관점에서 Linear Classifier는 각 클래스마다 계산이 독립적으로 수행되기 때문에 선형 함수를 행렬 W 의 각 행을 하나의 filter로 간주하고 convolution 연산을 수행한 것으로 해석할 수 있다.



- Visual Viewpoint : Visual 관점에서 Linear Classifier에 사용되는 행렬 W 의 각 행은 클래스 중 하나의 템플릿을 의미한다. 따라서 Linear Classifier는 내적을 통해 이미지와 각 템플릿을 비교해 가장 잘 매칭되는 템플릿을 선정하는 과정으로 해석할 수 있다.



- Geometric Viewpoint : Geometric 관점에서 이미지는 고차원의 유클리드 공간에서 확장되며, 이미지의 각 벡터는 고차원 공간 상의 하나의 점에 해당한다. Linear Classifier는 공간 상의 선형 함수가 돼 공간 상에서 점을 분류한다.



Feature Transforms

Visual Viewpoint에서 Linear Classifier로 추정한 템플릿은 그 클래스 내의 변화를 감지하지 못한다. Geometric Viewpoint에서 이미지의 분포는 비선형적이기 때문에 Linear Classifier로 구분하기 어렵다. 따라서 Linear Classifier가 잘 적용되도록 하는 Features를 찾는다.

- Color Histogram : 이미지의 색상 히스토그램을 사용한다. 색상 정보만 고려하기 때문에 공간 정보가 사라지는 단점이 있다.

- Histogram of Oriented Gradients(HOG) : Oriented Gradients의 히스토그램을 사용한다. Edge의 방향과 세기를 계산하고 이미지를 8×8 의 영역으로 나눈 뒤 각 영역의 edge 방향 가중치 히스토그램을 제작한다.

- Bag-of-Words : 원본 이미지에서 local image patches 또는 features를 추출한 뒤 이를 대표할 수 있는 code로 구성된 codebook을 제작한다. 이후 이미지를 분석할 땐 이미지의 features를 추출한 뒤 이 features에 대응되는 codeword를 찾아 이미지의 특징을 표현한다.

이미지의 features를 추출할 때 다양한 방법으로 추출한 features를 이어붙여 고차원의 features를 제작할 수 있다.

기준에는 Feature를 추출하는 과정과 이미지를 분류하는 과정을 분리해 단계별로 진행했으나, 분리된 과정에서는 Classification에 적합한 features를 적용하지 못하는 경우가 있어 Deep Learning 기반의 Classification 과정에서는 두 과정을 구분하지 않고 통합해서 진행한다.

Decision Tree

Decision Tree는 특정 기준에 따라 데이터를 구분하는 모델이다. 각 기능 테스트는 트리 형태로 구성되며 leaf 단계에서는 클래스 레이블에 대한 확률 분포를 저장한다.

Decision Tree에서는 Entropy와 Information Gain이 사용되는데, Entropy는 정보량의 기대값을 의미한다. Entropy의 계산은 다음과 같다. 이 때, C 는 데이터 샘플 S 에 표현된 클래스 집합, $p(c)$ 는 S 에서 클래스 c 의 경험적 분포를 의미한다.

$$H(S) = - \sum_{c \in C} p(c) \log(p(c))$$

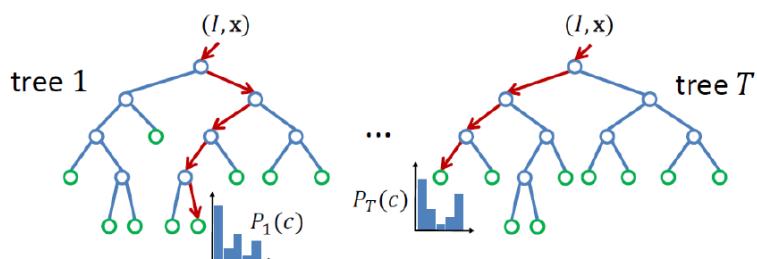
Entropy는 사건의 발생 확률의 차이가 클수록 값이 커진다.

Information Gain은 데이터 샘플 S 에서 정보 a 가 주어졌을 때의 Entropy 차이를 의미한다.

$$IG(S, a) = H(S) - H(S|a)$$

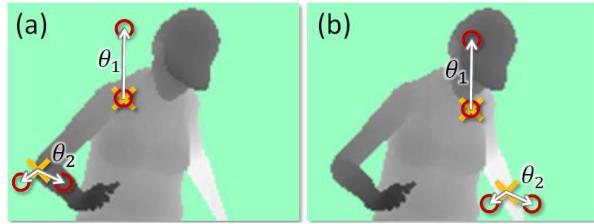
Random Forest

Random Forest는 다수의 Decision Tree를 조합해 다양한 계산을 수행해 평균 확률을 관측해 결과를 예측하는 방법이다.



Human Pose Recognition

Human Pose Recognition은 Random Forest를 사용해 사람의 행동을 감지하는 방법이다. Depth map에서 depth image features를 추출해 features의 이동을 Decision Tree로 표현한다. 이후 Information Gain을 계산해 의미 있는 움직임을 추정한다. 이 과정을 각각의 부분에 적용한 뒤 Random Forest로 구성해 평균화한다.



Combining Classifiers

독립적인 Classifiers를 훈련시키고 그 예측을 평균화해 Classifiers를 강화할 수 있다. 이 중에는 약한 classifiers를 반복적으로 학습해 강한 classifier를 구축하는 Boosting이 있다.

Object Detection

Object Detection은 이미지에서 오브젝트의 위치를 특정하는 Localization과 오브젝트의 종류를 파악하는 Classification을 함께 수행하는 것이다. 오브젝트는 일반적으로 bounding box를 통해 감지한다.

Sliding Window

Sliding Window는 크기가 고정된 window를 통해 이미지를 순회해 오브젝트를 감지하는 방법이다. 찾고자 하는 오브젝트의 크기가 정해지지 않기 때문에 다양한 크기와 종횡비의 window를 사용해야 하기 때문에 구현에 어려움이 있다.

Viola-Jones

Viola-Jones face detection는 sliding window을 기반으로 인간의 얼굴을 분석하는 감지기다. Viola-Jones face detection는 다음 단계를 거쳐 진행된다.

- Haar Feature Selection : 모든 인간의 얼굴이 공유하는 속성을 일치시킨 Haar features를 사용해 두 영역의 합계를 통해 feature를 추출한다.

- Creating an Integral Image : 이미지를 Integral Image로 변환한다. Integral Image의 임의의 픽셀의 값은 해당 픽셀과 해당 픽셀의 왼쪽 위에 위치한 모든 픽셀값의 합이다. 이를 통해 픽셀값의 합을 빠르게 구할 수 있다.

- Adaboost Training : 분류 성능이 높은 약한 classifier의 선형 조합을 통해 강한 classifier를 구축한다.

- Cascading Classifiers : 단계적으로 불필요한 features를 제거한다. 이 과정을 통해 최종적으로 정확한 features를 검출할 수 있다.

Recent Object Detection

Object Detection은 단일 RGB 이미지를 받아 감지된 오브젝트의 집합을 반환한다. 오브젝트를 감지하기 위해 오브젝트의 카테고리와 bounding box가 필요하다. Object Detection은 이미지에서 feature를 추출한 후 다음 두 과정을 수행한다.

- Classification : 감지한 오브젝트를 입력된 클래스에 따라 분류한다.
- Bbox Regression : bounding box로 오브젝트의 위치를 특정한다.

단일 오브젝트를 감지하는 경우 이미지 전체를 그대로 분석하지만, 다수의 오브젝트를 감지하는 경우 이미지의 구간을 나눠 오브젝트를 감지한다.

Region Proposals

Region Proposals 알고리즘은 오브젝트를 포함하는 최소 영역을 특정하는 방법이다. 이 과정을 통해 감지해야 하는 이미지의 영역의 개수를 효과적으로 줄일 수 있다.

Region Proposals은 다음 방법을 교차 수행해 최적의 영역을 탐지한다.

- Multiscale Saliency : 이미지 내에서 고유한 모양이 있는 영역을 추출한다.
- Color Contrast : 색상의 대비가 큰 영역을 추출한다.
- Edge Density : Edge의 밀도가 강한 지점을 추출한다.

Motion Estimation

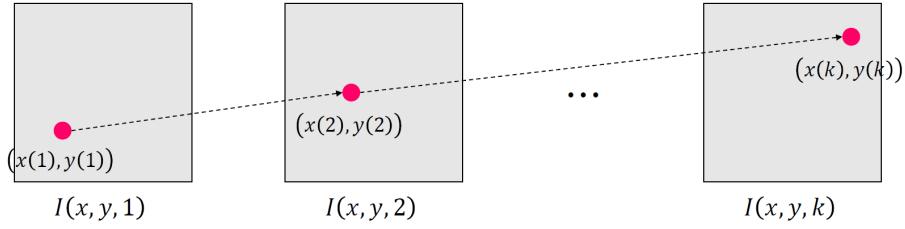
Motion Estimation은 오브젝트가 시간에 따라 어떻게 이동하는지를 추정하는 것이다. 시간이 연속적인 이미지에서 시간 변화에 따른 점의 대응을 찾아 오브젝트의 이동을 추정한다.

Optical Flow

Optical Flow는 이미지의 밝기 변화를 추적해 오브젝트의 움직임을 추정한다. Optical Flow는 연속한 프레임의 이미지에서 동일한 오브젝트의 픽셀이 다음 성질을 만족한다고 가정한다.

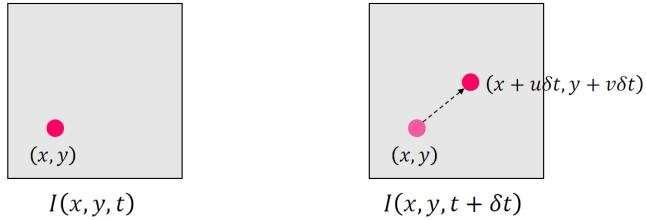
- Brightness constancy : 움직이는 물체의 밝기 변화가 없다.

$$I(x(t), y(t), t) = C \quad \text{where } t = 1, \dots, k$$



- Small motion : 픽셀의 이동 거리가 작다.

$$I(x + u\delta t, y + v\delta t, \delta t) = I(x, y, t)$$



따라서 Optical Flow는 동일한 밝기 값의 주위 픽셀을 찾아 오브젝트의 이동을 추정한다.

Brightness constancy와 Small motion을 모두 적용해 계산한 Brightness Constancy Equation은 다음과 같다.

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \rightarrow I_x u + I_y v + I_t = 0$$

여기서 $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ 는 Image gradients, $\frac{dx}{dt}, \frac{dy}{dt}$ 는 Flow velocities, $\frac{\partial I}{\partial t}$ 는 Temporal gradient를 의미하는데, Image gradients인 I_x, I_y 는 Sobel Filter나 Canny Edge Detection 등으로 구할 수 있고, Temporal gradient인 I_t 는 두 이미지 간의 차이를 통해 계산할 수 있다. 그러나 u 와 v 는 이 방정식만으로 계산할 수 없는데, 두 미지수를 구하기 위해 다음 두 알고리즘을 사용한다.

Lucas-Kanade method

Lucas-Kanade method는 window 안의 n 개의 픽셀이 전부 동일한 움직임을 가진다고 가정하고 window 내의 모든 픽셀에 대해 Brightness Constancy Equation을 구해 Optical Flow를 추정한다.

Brightness Constancy Equation을 행렬곱으로 나타내면 다음과 같다.

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}, \quad x = \begin{bmatrix} u \\ v \end{bmatrix}, \quad b = - \begin{bmatrix} I_{t_1} \\ I_{t_2} \\ \vdots \\ I_{t_n} \end{bmatrix}$$

위 식에서 미지수 두 개를 구하는데 식이 n 개이므로 least square를 통해 근사치를 계산한다.

$$\hat{x} = (A^T A)^{-1} A^T b$$

Lucas-Kanade method는 이미지의 작은 영역만을 보고 오브젝트의 이동 방향을 추정하기 때문에 불확실한 추정이 나타나는 aperture problem이 발생한다.

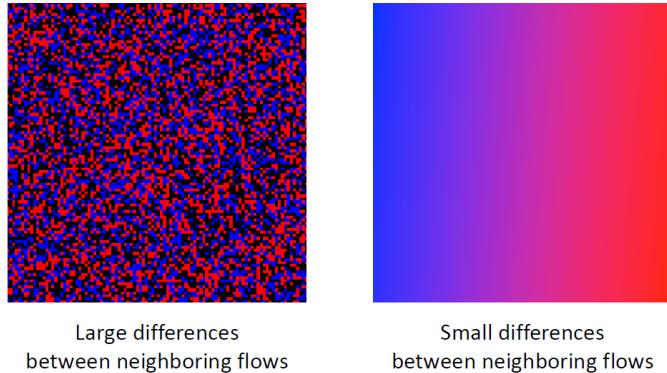
Horn-Shunck method

Horn-Shunck method는 Lucas-Kanade method와 같이 Brightness Constancy와 Small motion를 고려하면서 추가로 smooth flow field를 함께 고려해 Optical Flow를 추정하는 방법이다.

- Smooth Flow Field : 인접한 영역이 균일하게 움직임을 가정해 이미지 전체의 흐름을 계산한다.

Smooth Flow Field는 인접한 4방향을 고려하여 다음과 같이 계산한다.

$$E_s(i, j) = \frac{1}{4}[(u_{i,j} - u_{i+1,j})^2 + (u_{i,j} - u_{i,j+1})^2 + (v_{i,j} - v_{i+1,j})^2 + (v_{i,j} - v_{i,j+1})^2]$$



Horn-Schunck Optical Flow는 Smooth Flow Field와 Brightness Constancy를 함께 고려하여 다음과 같이 계산한다.

$$\min_{u, v} \sum_{i,j} E_s(i, j) + \lambda E_b(i, j)$$

Horn-Schunck method는 Image gradients I_x, I_y 와 Temporal gradient인 I_t 를 미리 계산한다. 이후 optical flow field의 벡터 u, v 를 0으로 초기화하고 다음 수식에 따라 점진적으로 갱신한다.

$$\hat{u}_{k,l} = \bar{u}_{k,l} - \frac{I_x \bar{u}_{k,l} + I_y \bar{v}_{k,l} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x, \quad \hat{v}_{k,l} = \bar{v}_{k,l} - \frac{I_x \bar{u}_{k,l} + I_y \bar{v}_{k,l} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

Linear Classifiers for Neural Networks

Loss Function

Loss Function은 Linear Classifier에서 산출된 weight가 일으키는 손실을 나타내는 지표로, weight가 얼마나 좋은지 정량화하는 함수다. Loss Function의 값이 낮을수록 weight가 좋은 모델이다. N 개의 Input 데이터 x 와 output 데이터 y 의 쌍 $\{(x_i, y_i)\}_{i=1}^N$ 에 대해 손실 함수 \mathcal{L} 는 각 데이터의 손

실값의 평균으로 다음과 같다.

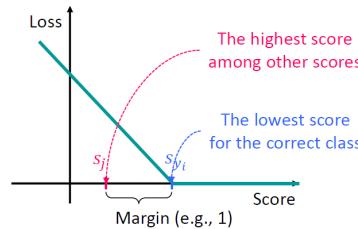
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(f(x_i; W), y_i)$$

Loss Function를 계산하는 방법에는 다음이 있다.

- Multiclass SVM Loss : i 번째 데이터 쌍 (x_i, y_i) 에 대해 Multiclass SVM Loss 함수는 다음과 같다.

$$\mathcal{L}_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

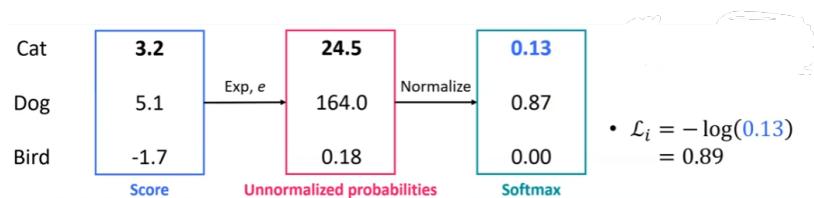
이 때, s_{y_i} 는 정답 클래스 점수의 최소값이고, s_j 는 정답 외의 클래스 점수의 최댓값으로, s_{y_i} 가 $s_j + 1$ 이상일 경우 함수의 값은 0이 된다.



- Softmax Function : 입력 데이터의 점수 함수 $s = f(x_i; W)$ 에 대해 Softmax Function은 다음과 같다.

$$\mathcal{L}_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Softmax Function을 위해 각 클래스의 점수를 지수화한 후 정규화한다. 이후 정답 점수에 Softmax Function을 적용한다.



Overfitting

Overfitting은 학습 데이터에 너무 과하게 학습돼있는 상태를 말한다. Overfitting된 모델의 경우 새로운 데이터를 입력받았을 때 제대로 동작하지 않는다.

Regularization

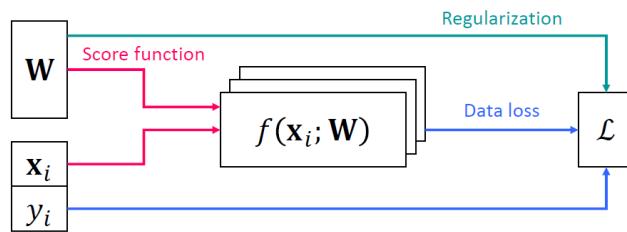
Regularization은 모델이 Overfitting되는 경우를 방지하는 기법이다. Regularization의 종류는 다음과 같다.

- l_1 regularization : $R(W) = \sum_{k,l} |W_{k,l}|$

- l_2 regularization : $R(W) = \sum_{k,l} W_{k,l}^2$

Loss Function은 Regularization 함수를 포함해 모델이 overfitting되지 않는 방향으로 학습되도록 돋는다.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(f(x_i; W), y_i) + \lambda R(W)$$



Optimization

Score function은 이미지의 각 픽셀을 클래스의 점수로 매핑하며, Loss function은 이 점수를 훈련 데이터의 레이블과 일치시키는 모델 W 의 품질을 측정하는 함수다. Optimization은 Loss Function을 최소화하는 모델 W 를 찾는 과정이다.

$$W^* = \underset{W}{\operatorname{argmin}} \mathcal{L}(W)$$

Optimization의 방법에는 다음이 있다.

- Random Search : 모델 W 를 무작위로 추출한다.

- Following the Gradient : loss function의 gradient를 구해 gradient가 가장 큰 지점을 따라 내려간다. Gradient가 큰 지점을 따라 내려가는 데 Gradient Descent Method를 사용한다.

$$W_{n+1} = W_n - \alpha \nabla_w \mathcal{L}$$

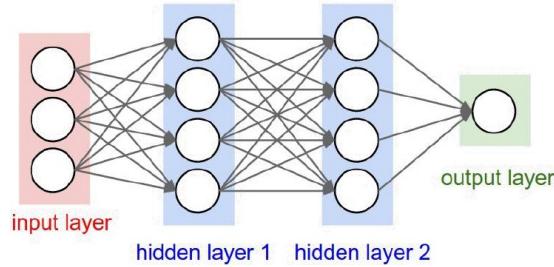
이 때, α 는 learning rate의 크기로, α 가 너무 크거나 작으면 목표하는 지점을 찾아가기 어렵다.

Neural Networks

Neural Networks는 Linear Classifier를 수행하는 여러 층의 layer를 쌓아 데이터를 훈련해 결과를 출력하는 방법이다.

Neural Networks의 layer는 다음과 같이 구분할 수 있다.

- Input layer : 이미지를 입력받아 Hidden layer에 전달한다.
- Hidden layer : Hidden layer의 각 neuron은 이전 layer의 모든 neuron과 연결돼있으며, 각 노드가 독립적으로 Linear Classifier를 수행한다.
- Output layer : Hidden layer를 거쳐 classification에서 클래스의 score를 나타낸다.



Activation Functions

단순히 Linear Classifier를 중첩하면 이는 또 다른 Linear Classifier가 되며 이 경우 아무리 층을 쌓아도 하나의 Linear Classifier 연산과 동일한 결과를 가진다. Neural Networks는 Non-linear한 연산을 위해 입력 신호를 weight와 곱하고 Activation Functions를 통해 변환한다.

Activation Functions의 종류는 다음과 같으며, 일반적으로 ReLU 함수를 사용한다.

- Sigmoid : $\sigma(x) = \frac{1}{1+e^{-x}}$
- tanh : $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- ReLU : $\max(0, x)$
- Leaky ReLU : $\max(0.2x, x)$
- ELU : $f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$

Backpropagation

Backpropagation은 다층의 layer에서 학습 후의 Weights를 갱신하기 위해 사용하는 방법이다. Forward pass를 통해 output을 계산하고, loss function을 통해 계산한 output과 실제 output을 비교한 뒤 Backward pass를 통해 gradient를 계산해 weight를 갱신한다.

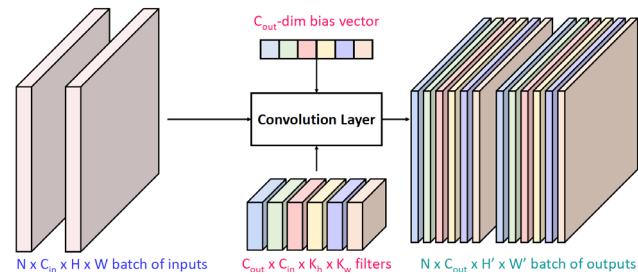
Convolutional Neural Networks

Neural Networks는 이미지의 공간적인 구조를 반영하지 못하는 문제가 있다. Convolutional Neural Networks(CNNs)는 기존의 Neural Networks에 convolution 연산을 수행하는 layer를 추가 한다. CNNs는 convolution 연산을 수행함으로써 이미지의 classification 외에 object detection 및 segmentation 기능을 제공한다.

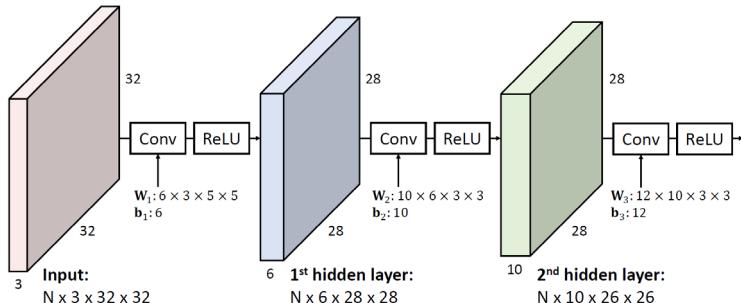
Convolutional Neural Networks는 기존의 Neural Networks가 가지는 Fully-Connected Layers와 Activation Function 외에 다음 기능을 사용한다.

- Convolution Layer

이미지를 1차원 벡터로 나열해 계산하는 Neural Networks와 달리 CNNs는 이미지 배열을 그대로 사용해 계산한다. Convolution Layer는 batch 단위로 작업하며, 이미지에 다양한 filter를 적용해 여러 convolution filter 결과를 얻는다. 이 때, convolution 연산에 적용되는 filter는 weight를 의미 한다.



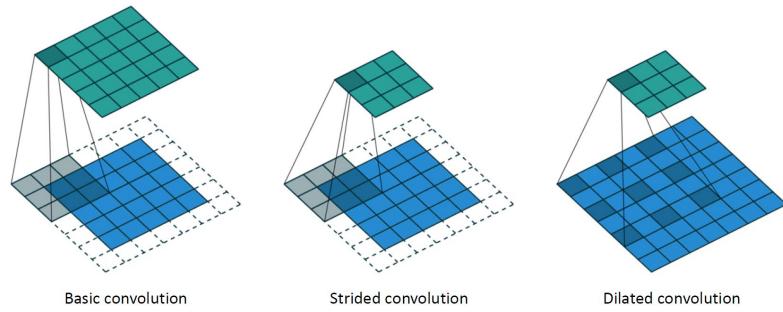
CNNs는 이런 Convolution Layer를 여러 층 쌓아 non-linear한 연산을 수행한다.



Convolution 연산을 수행하면 이미지의 정보가 압축된다. 따라서 Convolution layer를 거치면 이미지가 down-sampling되는 효과를 갖는다.

이미지를 down-sampling하기 위한 convolution 연산은 다음 종류가 있다.

- Basic convolution : 일반적인 convolution 연산을 수행하면서 가장자리를 제외한다.
- Strided convolution : 탐색 구간의 탐색 횟수를 줄인다.
- Dilated convolution : filter의 간격을 넓혀 광범위를 탐색한다.



- Pooling Layer

Pooling Layer는 down-sampling을 통해 변수의 크기와 계산량을 줄이기 위해 사용된다.

Pooling Layer에는 다음 방법이 사용된다.

- MAX Pooling : 해당 구간의 최댓값을 채택한다.
- Average Pooling : 해당 구간의 평균값을 채택한다.

- Batch Normalization

Batch Normalization은 각 batch 별로 평균과 분산을 통해 정규화하는 과정이다. 이 과정을 통해 학습에 필요한 시간을 효과적으로 줄일 수 있다.

