清華大學
Tsinghua University

Advanced Computer Vision

THU×SENSETIME – 80231202

商汤
sensetime

Chapter 1 - Section 2

Feature Detection

Dr. Li Hongyang

Thursday, March 3, 2022

Highlights

Learn the image filtering and feature descriptors

Learn the characteristics of feature descriptors

Learn how to add feature descriptors to CV tasks

Understand the features that basic descriptors bring to CNN

Outline

Part 1	Image filtering、Feature detectors and descriptors
Part 2	Traditional CV Application

Image filtering

- Image -- A 2D discrete signal

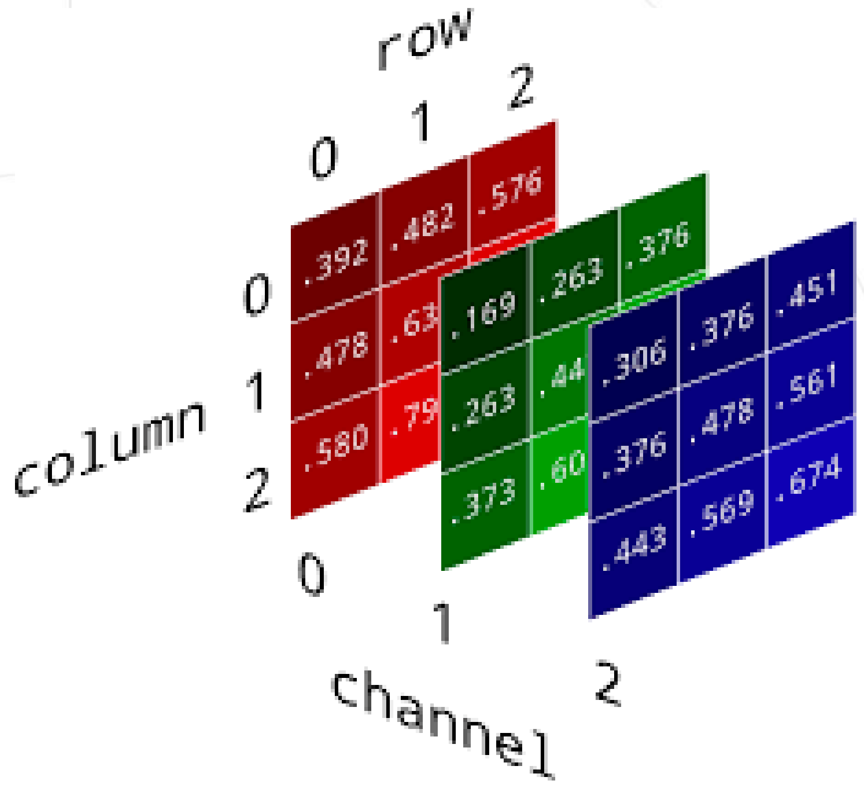
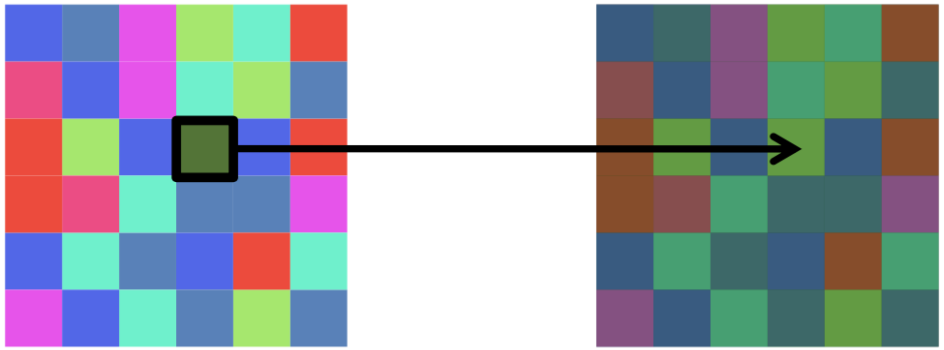


Image filtering

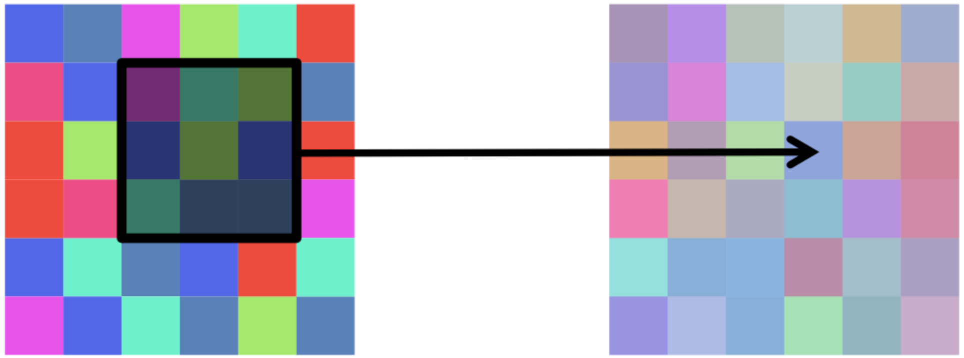
- Image filtering

Point Operation



point processing

Neighborhood Operation



“filtering”

Image filtering

- Image filtering - Point processing

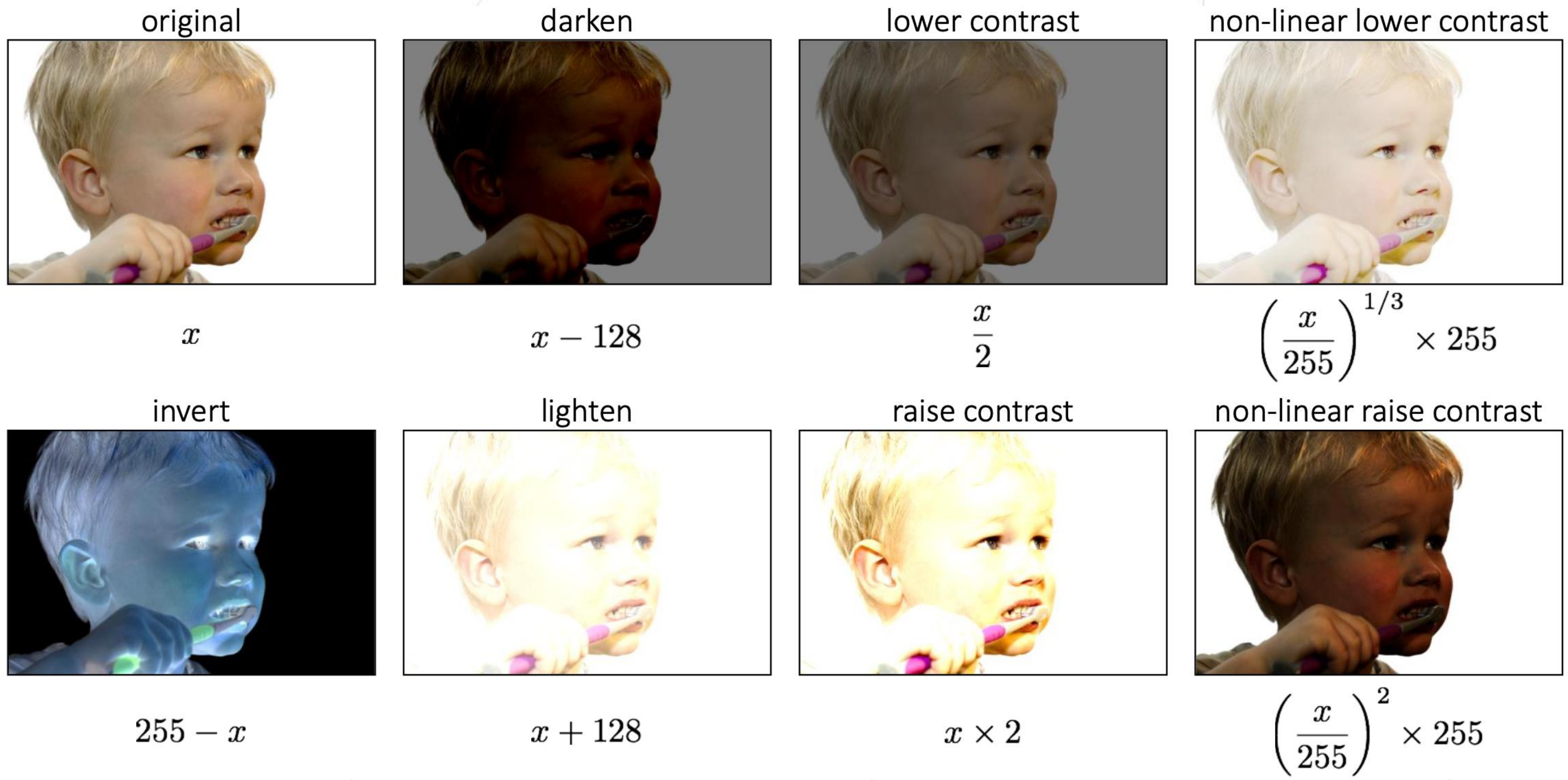


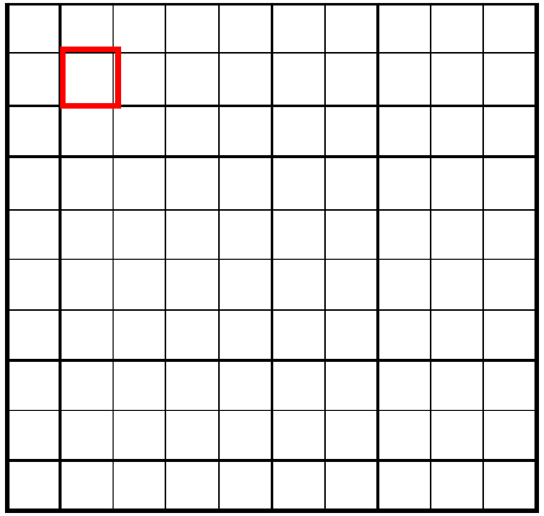
Image filtering

- Image filtering - box filtering

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$



$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

box filtering

Image filtering

- Image filtering - box filtering example

- Image filtering - Gaussian filtering

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

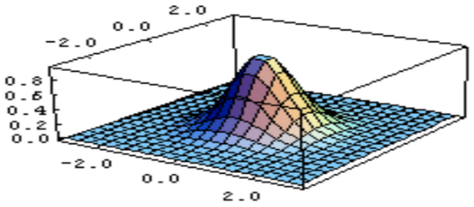


Image filtering

- Image filtering - box filtering vs Gaussian filtering

original

box filtering

Gaussian filtering

- Image filtering - other filters

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

output

unchanged

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

output

shift to left by one

input

filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

output

sharpening

sharpening

Image filtering

- Image filtering - Detecting edges

definition of a derivative using forward difference

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

For discrete signals: Remove limit and set h = 2

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

second-order finite difference

$$f''(x) = \lim_{h\to 0} \frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

1D derivative filter

1	0	-1

Laplace filter

1	-2	1

Image filtering

- Image filtering - Sobel filter

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

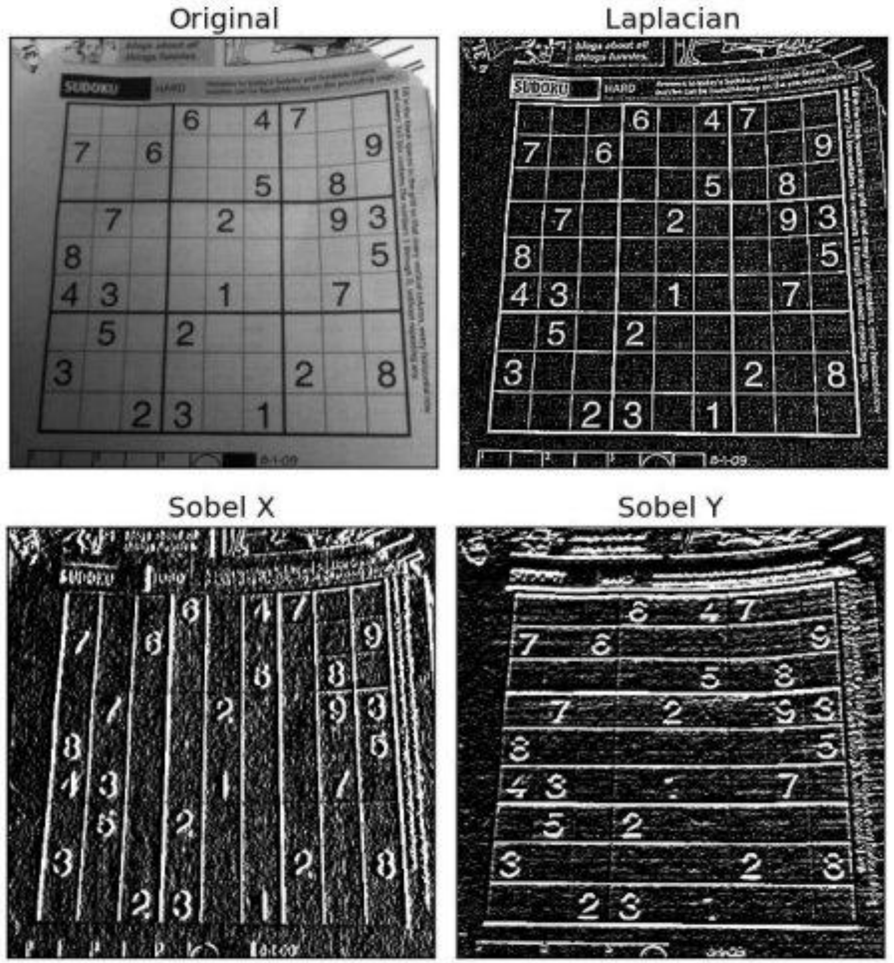


Image filtering

- Image filtering -Several derivative filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

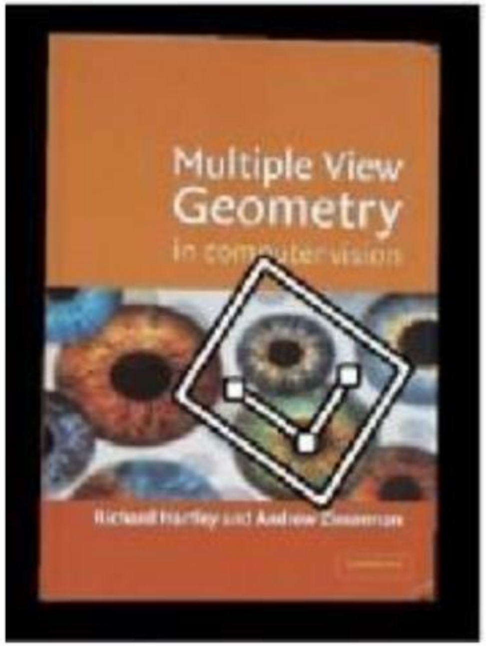
0	1
-1	0

1	0
0	-1

Feature detectors and descriptors

- Photometric transformations

- **Geometric transformations**
  - objects will appear at different scales, translation and rotation



Feature detectors and descriptors

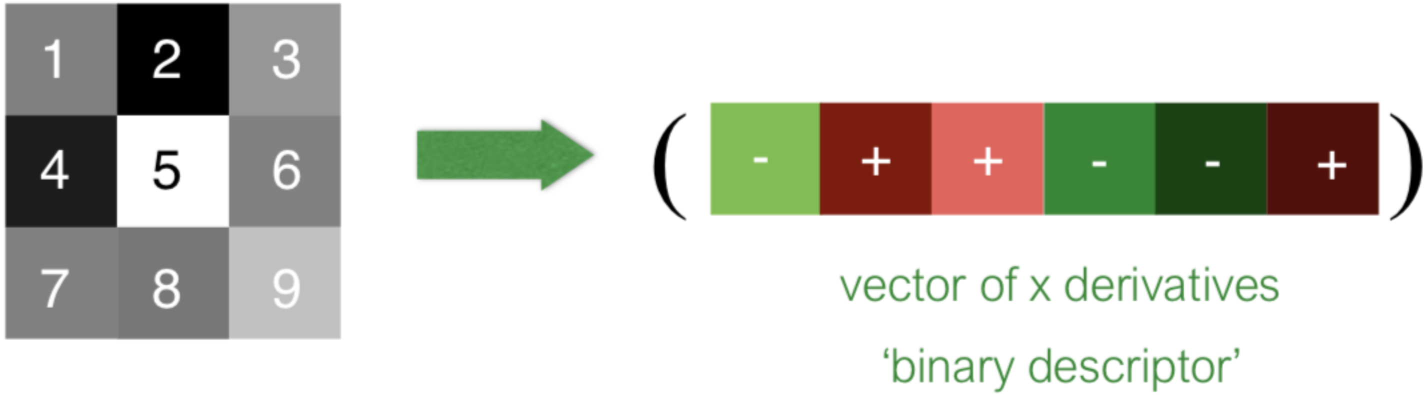
What is the best descriptor for an image feature?

Feature detectors and descriptors

- **Image patch**
  - Just use the pixel values of the patch

  - Pros: Perfectly fine if geometry and appearance is unchanged
  - Cons: sensitive to absolute intensity values

- **Image gradients**
  - Use pixel differences

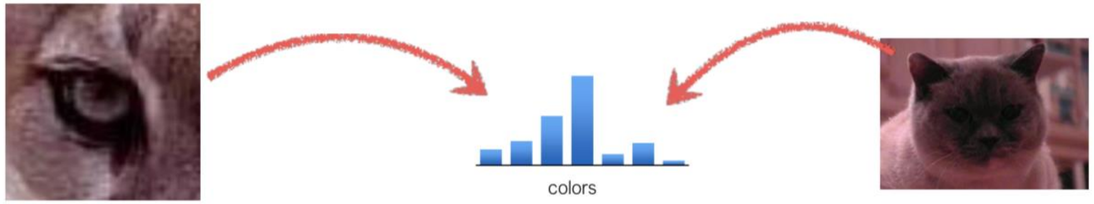


  - Pros: Feature is invariant to absolute intensity values
  - Cons: sensitive to deformations

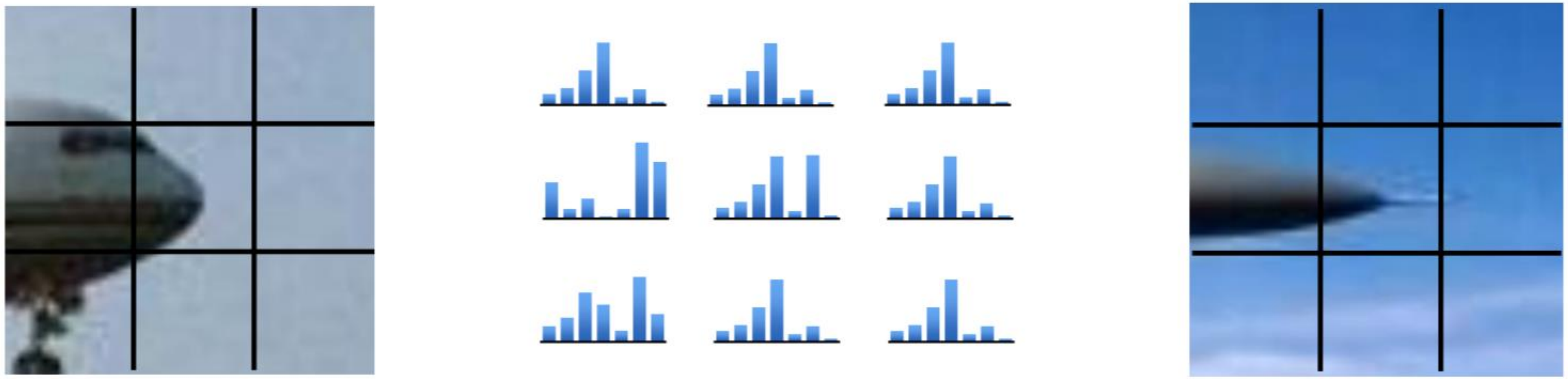
Feature detectors and descriptors

- **Color histogram**
  - Count the colors in the image using a histogram

  - Pros: Invariant to changes in scale and rotation
  - Cons: Insensitive to spatial layout

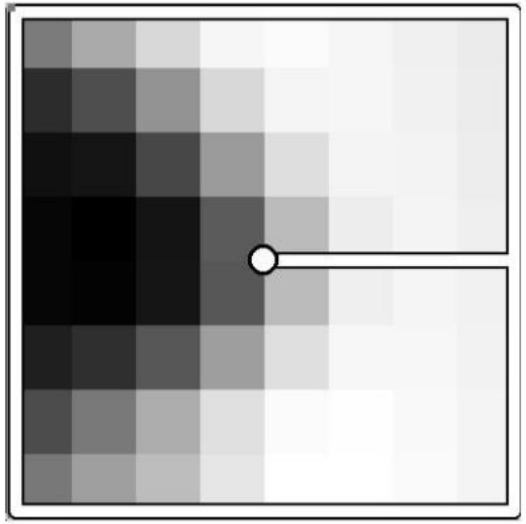


- **Spatial histograms**
  - Compute histograms over spatial ‘patch’



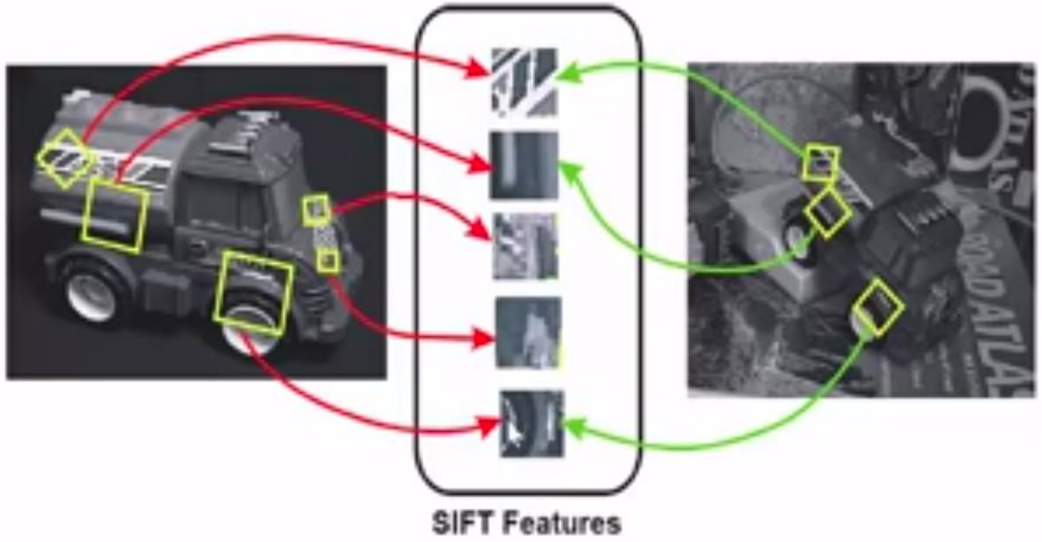
  - Pros: Retains rough spatial layout
  - Cons: sensitive to rotation

- **Orientation normalization**
  - Use the dominant image gradient direction to normalize the orientation of the patch

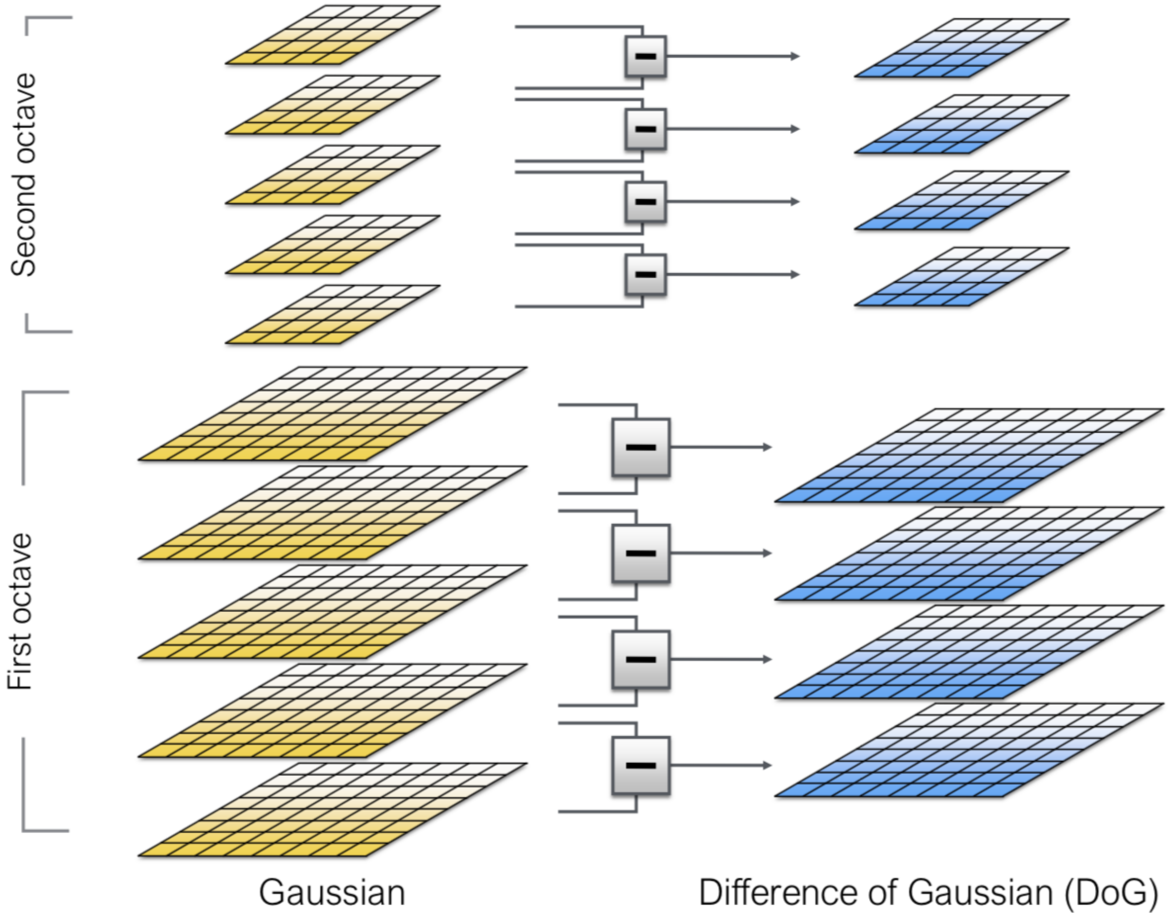


- **Scale Invariant Feature Transform （SIFT)**

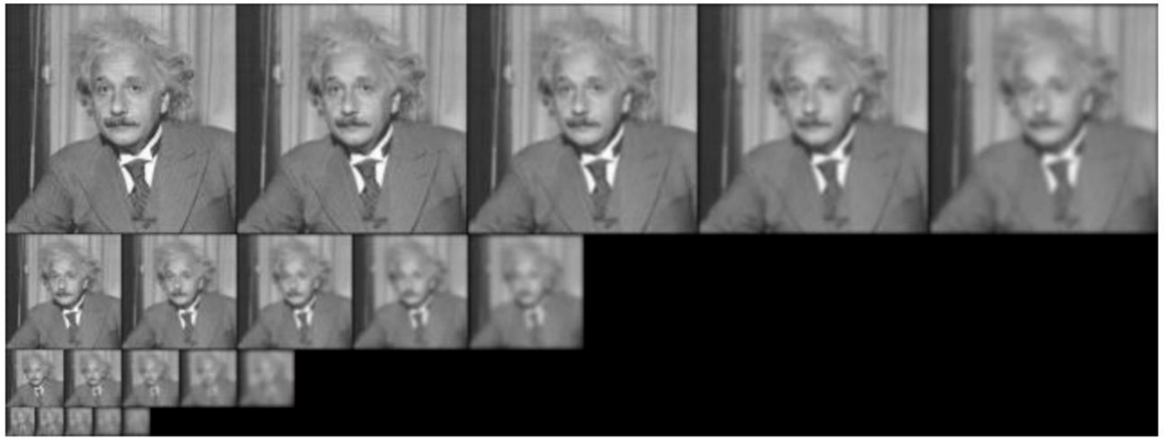
  1. Multi-scale extrema detection
  2. Keypoint localization
  3. Orientation assignment
  4. Keypoint descriptor



1. Multi-scale extrema detection



1. Multi-scale extrema detection

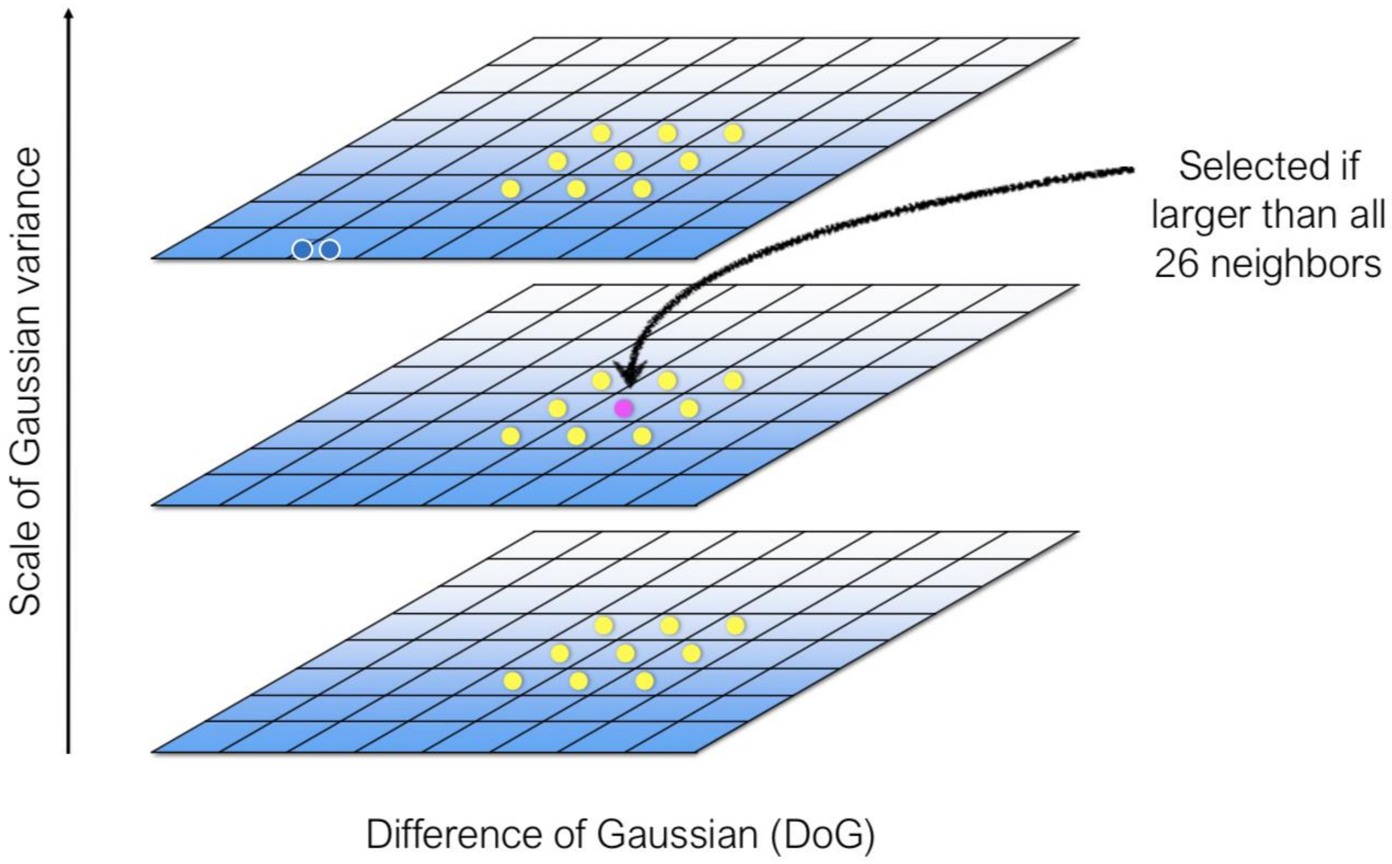


Gaussian

Laplacian

Feature detectors and descriptors

- **Scale-space extrema**



2. Keypoint localization

- 2nd order Taylor series approximation of DoG scale-space

$$f(\mathbf{x}) = f + \frac{\partial f}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 f}{\partial \mathbf{x}^2}\mathbf{x}$$

$$\mathbf{x} = \{x, y, \sigma\}$$

- Take the derivative and solve for extrema

$$\mathbf{x}_m = -\frac{\partial^2 f}{\partial \mathbf{x}^2}^{-1} \frac{\partial f}{\partial \mathbf{x}}$$

- Additional tests to retain only strong features

3. Orientation assignment

- For a keypoint, L is the Gaussian-smoothed image with the closest scale

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- Detection process returns

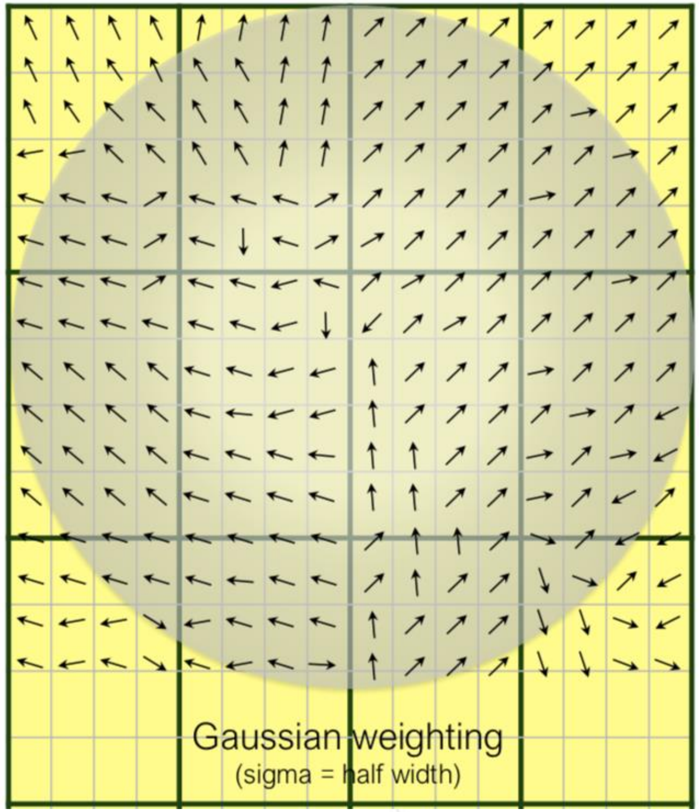
$$\{x, y, \sigma, \theta\}$$

location scale orientation

4. Keypoint descriptor

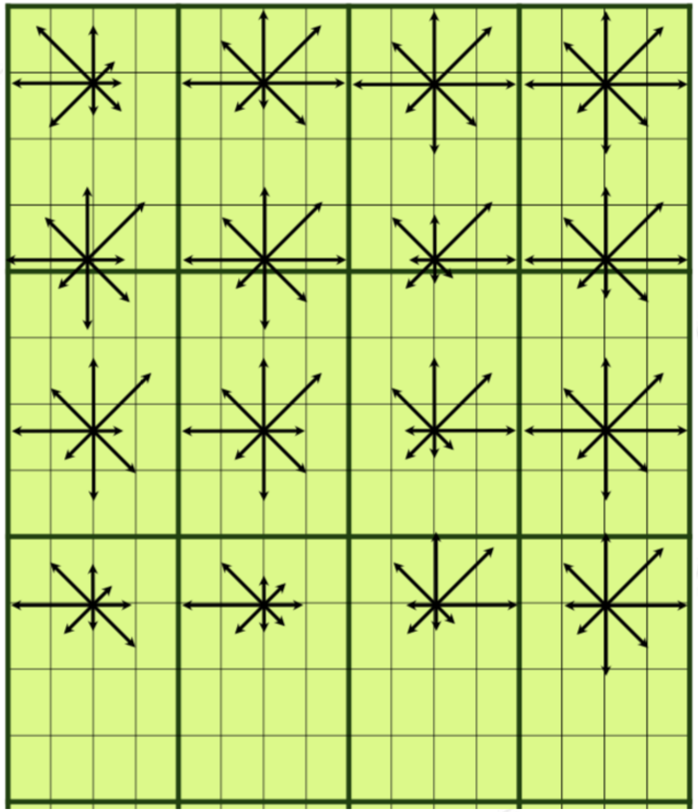
Image Gradients

(4 x 4 pixel per cell, 4 x 4 cells)

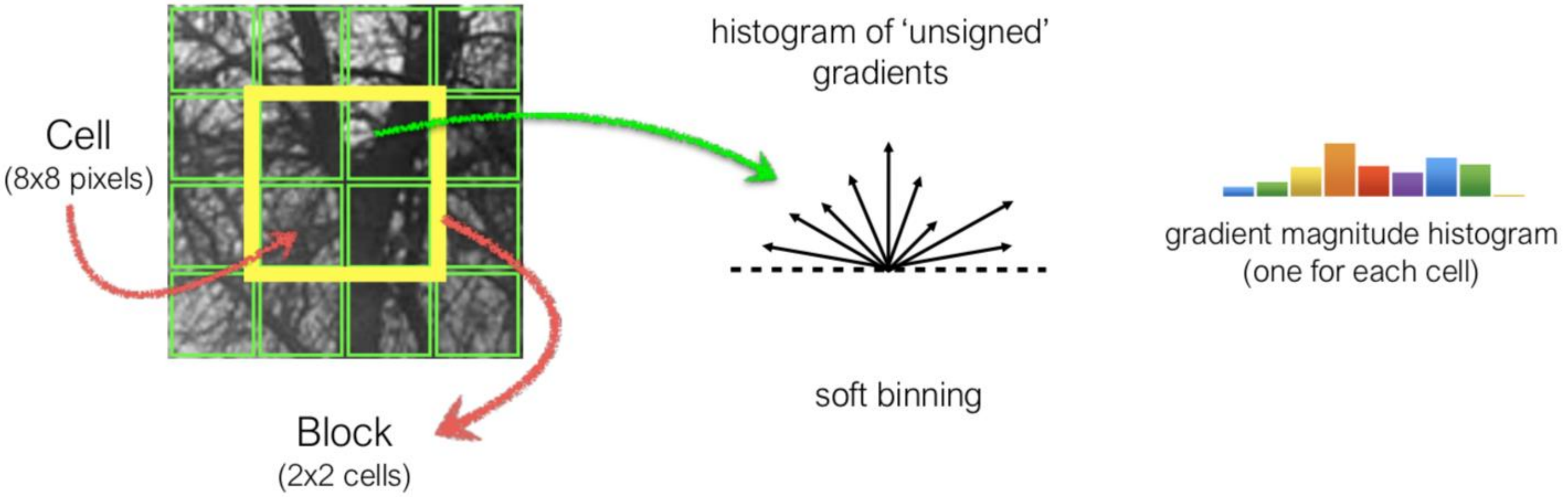


SIFT descriptor

(16 cells x 8 directions = 128 dims)



- **Histograms of Oriented Gradients （HOG）**



- Concatenate and L-2 normalization

Dalal, Triggs. Histograms of Oriented Gradients for Human Detection. CVPR, 2005

Feature detectors and descriptors

- **HOG for Pedestrian detection**

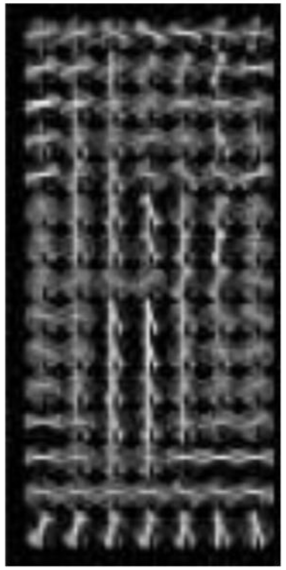
1 cell step size

128 pixels
16 cells
15 blocks

64 pixels
8 cells
7 blocks

15 x 7 x 4 x 9 =
3780

visualization

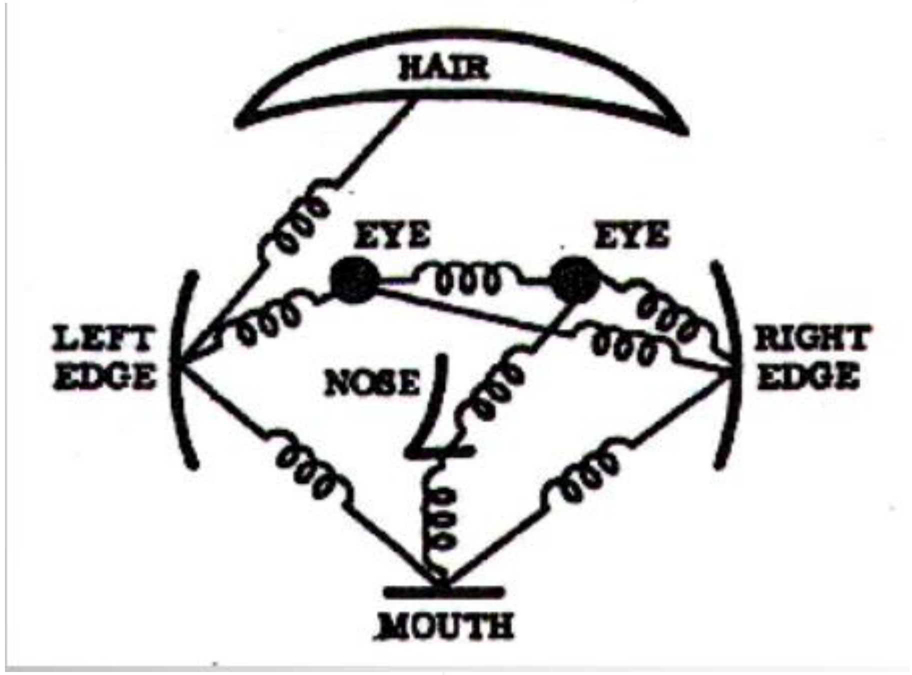


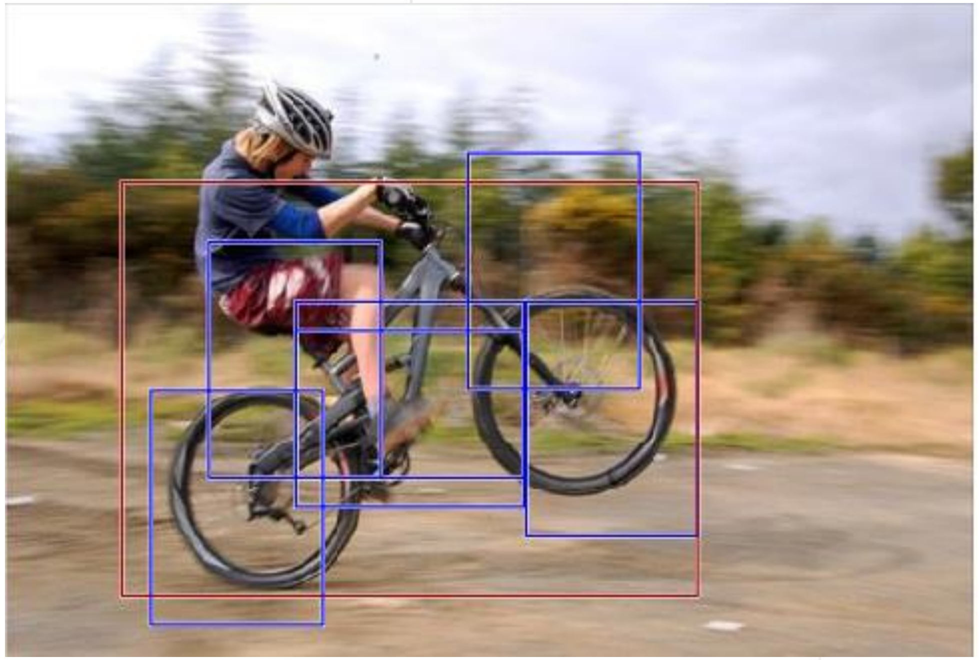
http://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/

Outline

Part 1	Image filtering、Feature detectors and descriptors
Part 2	Traditional CV Application

Traditional CV Application——DPM

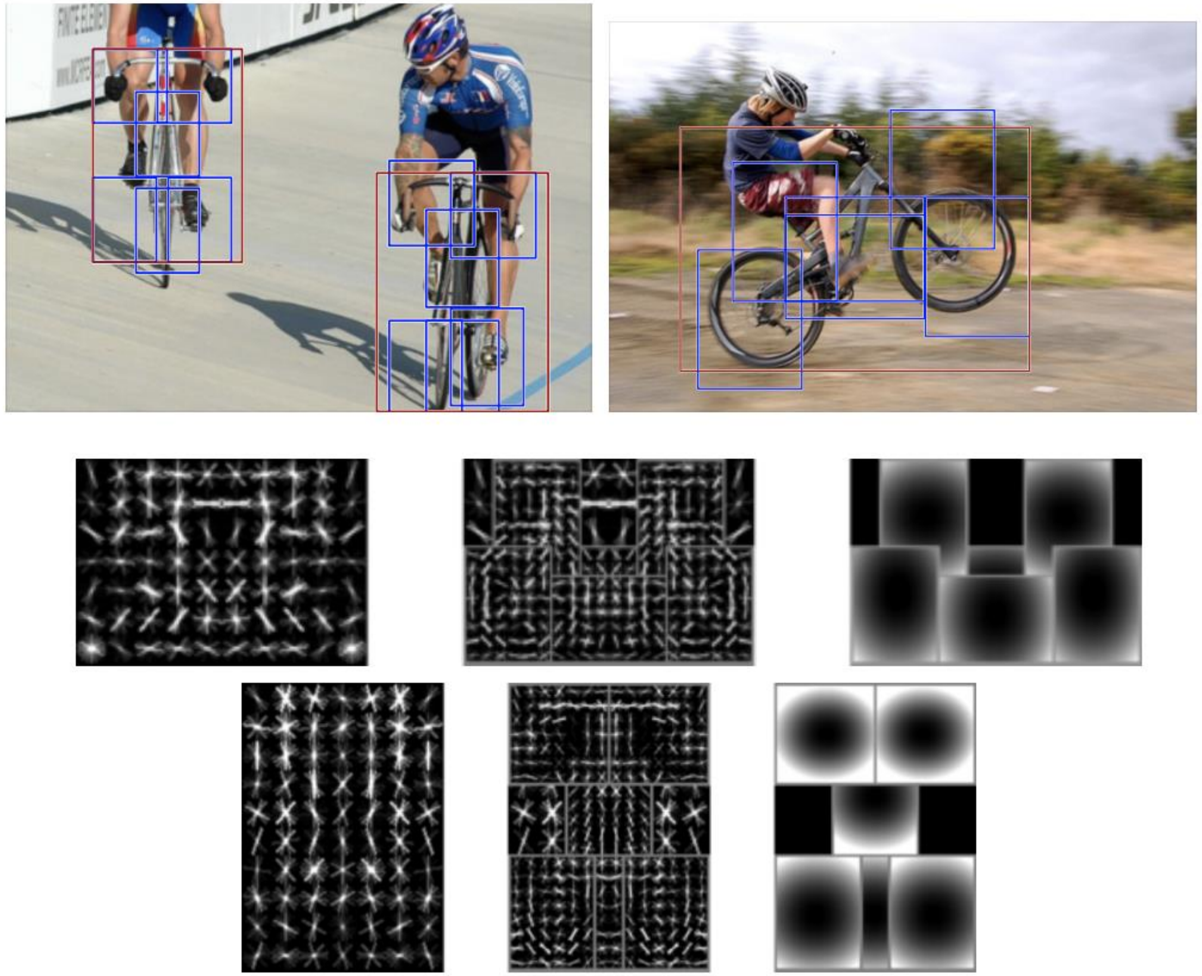


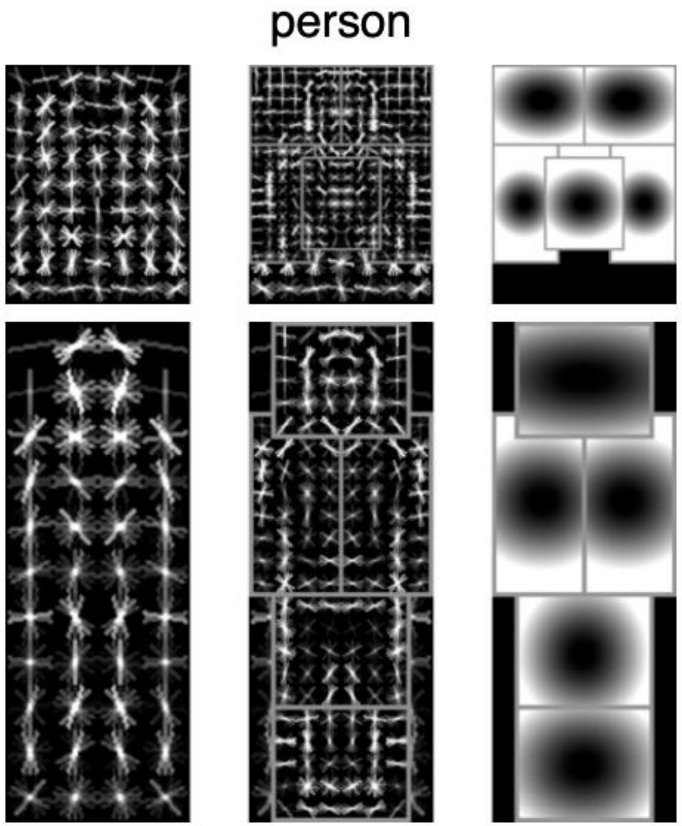


Spring-based Models

https://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf

Traditional CV Application——DPM



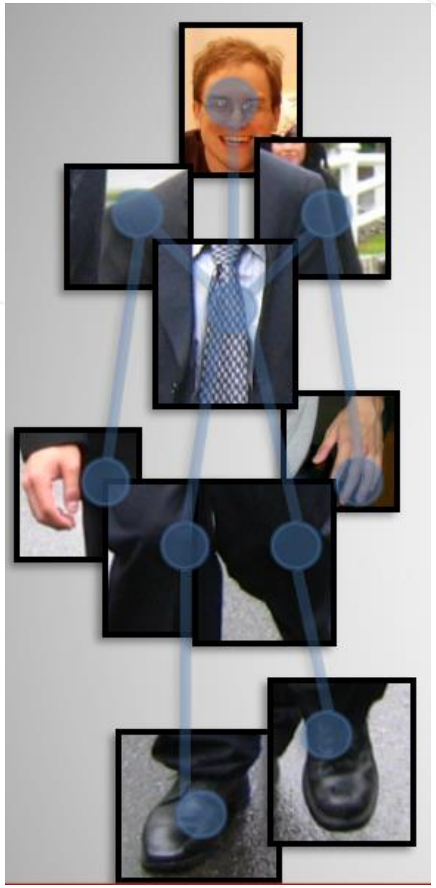


https://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf

Traditional CV Application——DPM

- Model is represented by a graph $G = (V, E)$
  - $V = \{v_1,\ldots,v_n\}$ are the parts
  - $(v_i,v_j) \in E$ indicates a connection between parts
- $m_i(l_i)$ is a cost for placing part i at location l_i
- $d_{ij}(l_i,l_j)$ is a deformation cost
- Optimal configuration for the object is $L = (l_1,\ldots,l_n)$ minimizing

$$E(L) = \sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i,l_j)$$



https://www.cs.ucf.edu/~bagci/teaching/computervision16/Lec21.pdf

Traditional CV Application——BM3D

Image Denoise

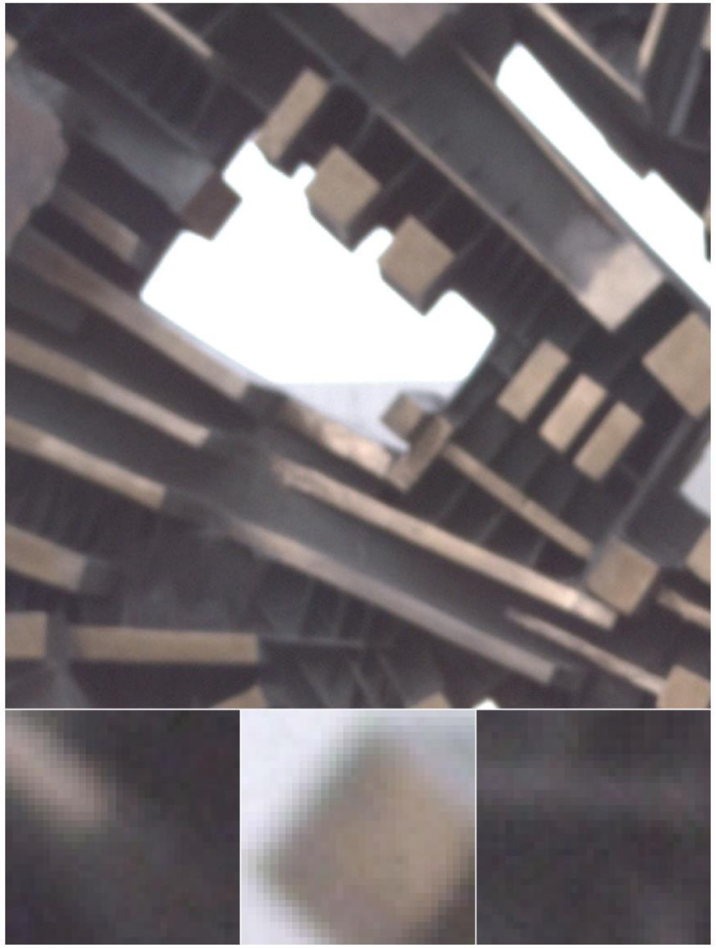
Gauss White Noise

$$z(x) = y(x) + \eta(x) \qquad \eta(\cdot) \sim \mathcal{N}(0, \sigma^2)$$

Gauss-Poisson Noise

$$y \sim \mathcal{N}(\mu = x, \sigma^2 = \lambda_{read} + \lambda_{shot} x).$$

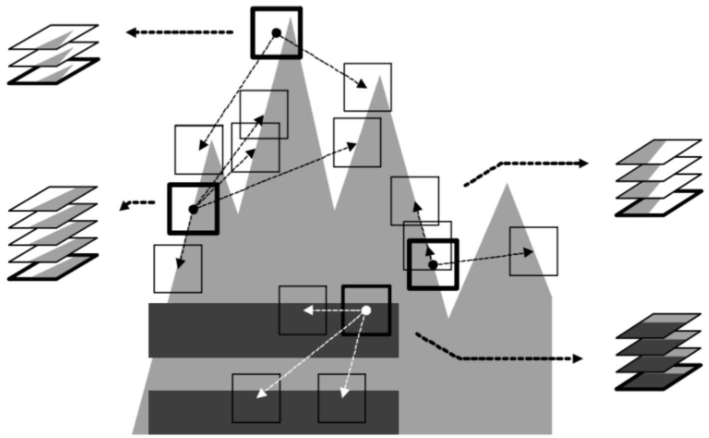
(a) Noisy Input, PSNR = 18.76

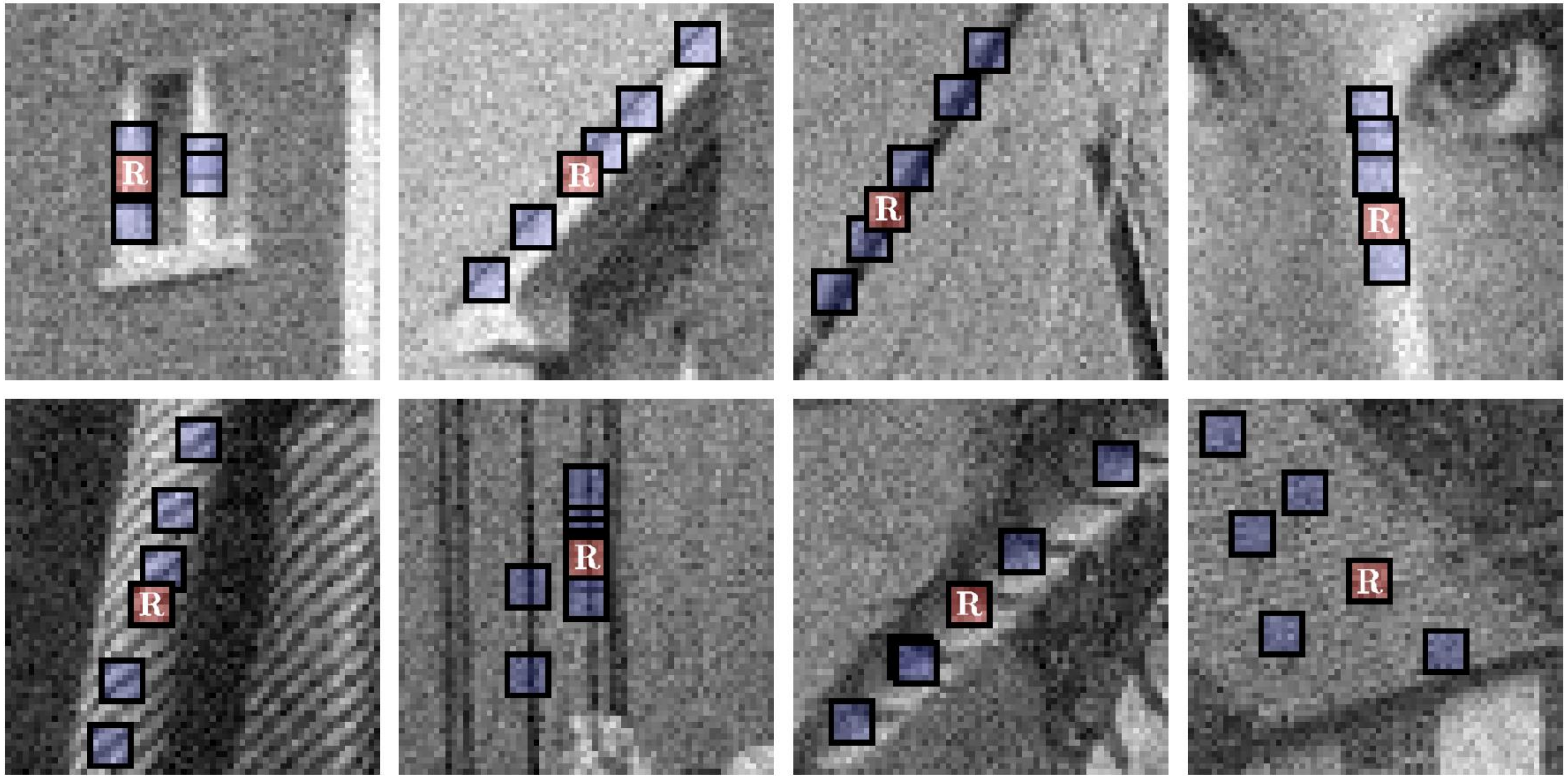


(b) Ground Truth

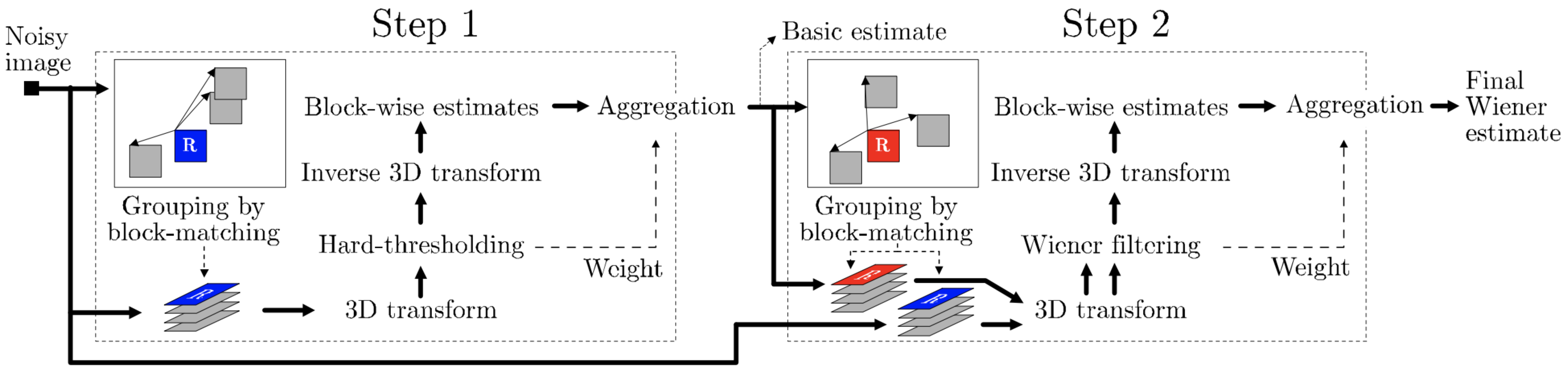
Traditional CV Application——BM3D

Reference block
and matched blocks



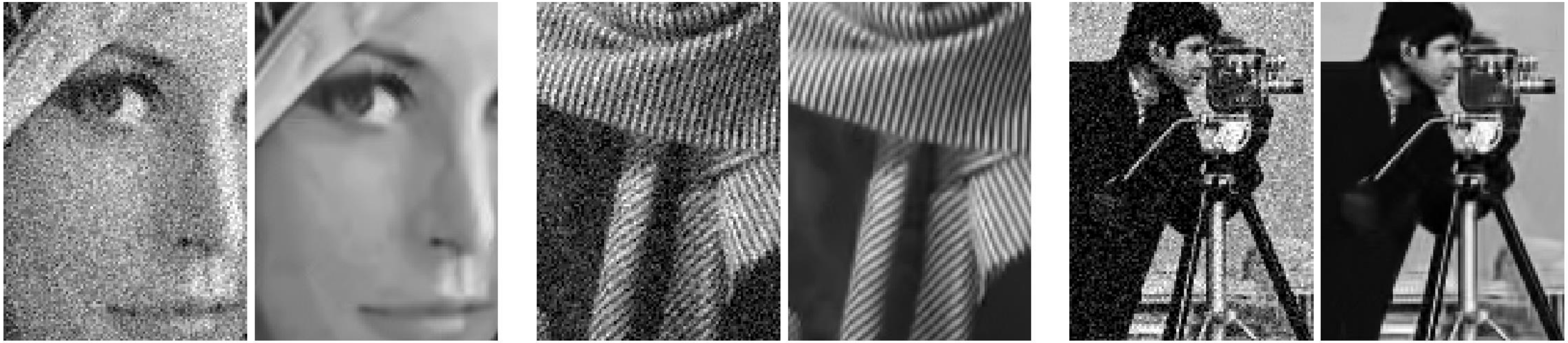


Traditional CV Application——BM3D

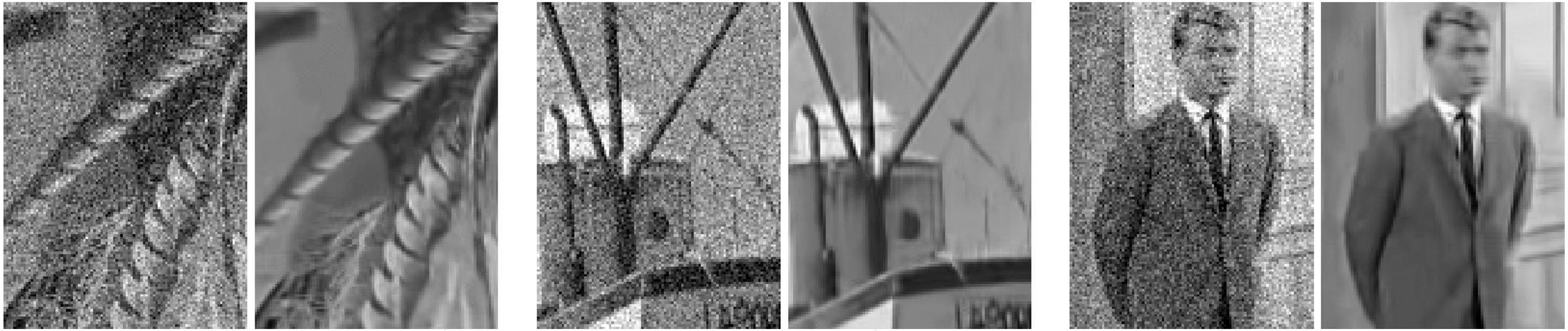


https://www.cs.tut.fi/~foi/GCF-BM3D/BM3D_TIP_2007.pdf
https://zh.wikipedia.org/wiki/%E4%B8%89%E7%BB%B4%E5%9D%97%E5%8C%B9%E9%85%8D%E7%AE%97%E6%B3%95

Traditional CV Application——BM3D



(d) *Man* (PSNR 29.62 dB) (e) *Boats* (PSNR 29.91 dB) (f) *Couple* (PSNR 29.72 dB)



Traditional CV Application——BM3D

PSNR 29.33 dB

PSNR 29.32 dB

PSNR 29.48 dB

PSNR 29.68 dB

PSNR 29.91 dB

PSNR 28.29 dB

PSNR 28.91 dB

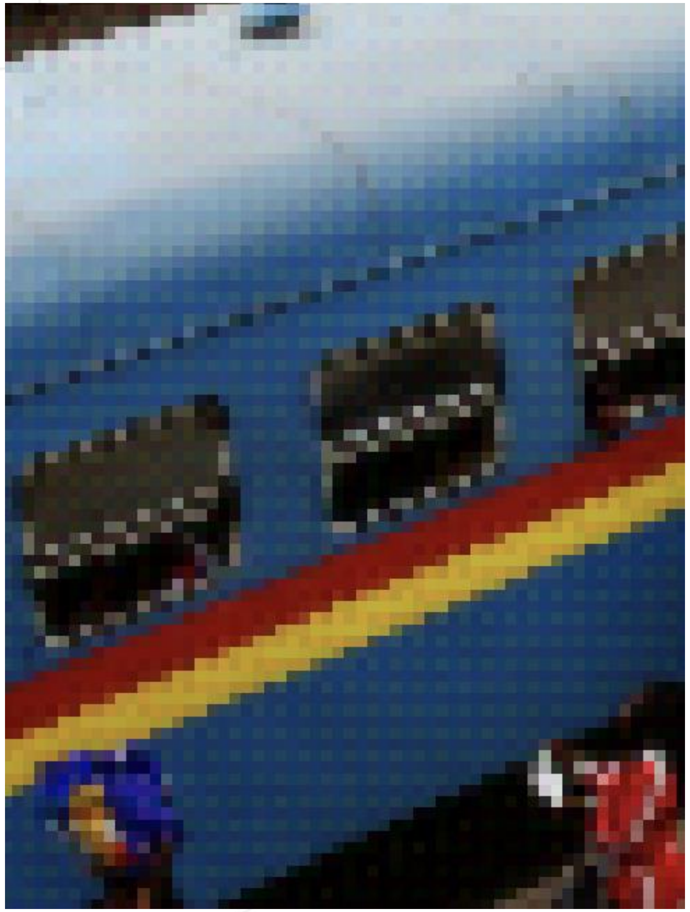
PSNR 29.11 dB

PSNR 29.08 dB

PSNR 29.45 dB

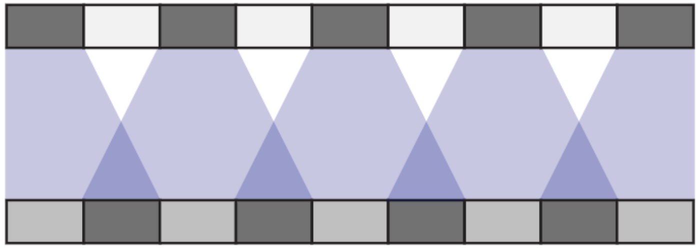
Traditional CV + Deep Learning

Checkerboard artifact



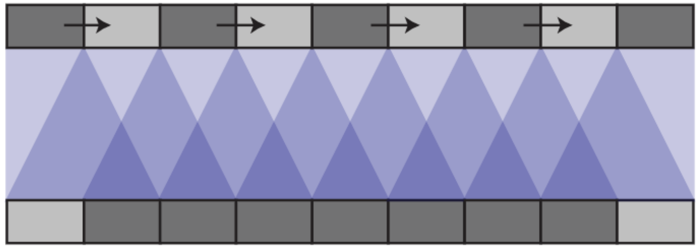
Traditional CV + Deep Learning

Checkerboard artifact——upsample layer



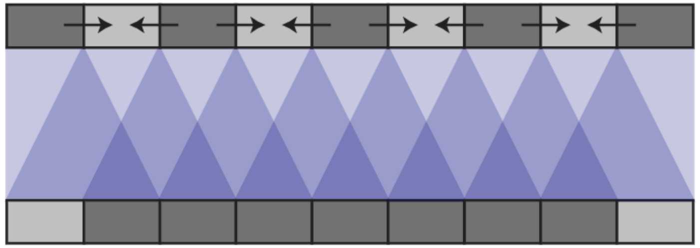
$$\begin{bmatrix} a & c & \\ & b & \\ & a & c \\ & & b \end{bmatrix}$$

Deconvolution



$$\begin{bmatrix} a+b & c & \\ a & b+c & \\ & a+b & c \\ & a & b+c \end{bmatrix}$$

NN-Resize Convolution

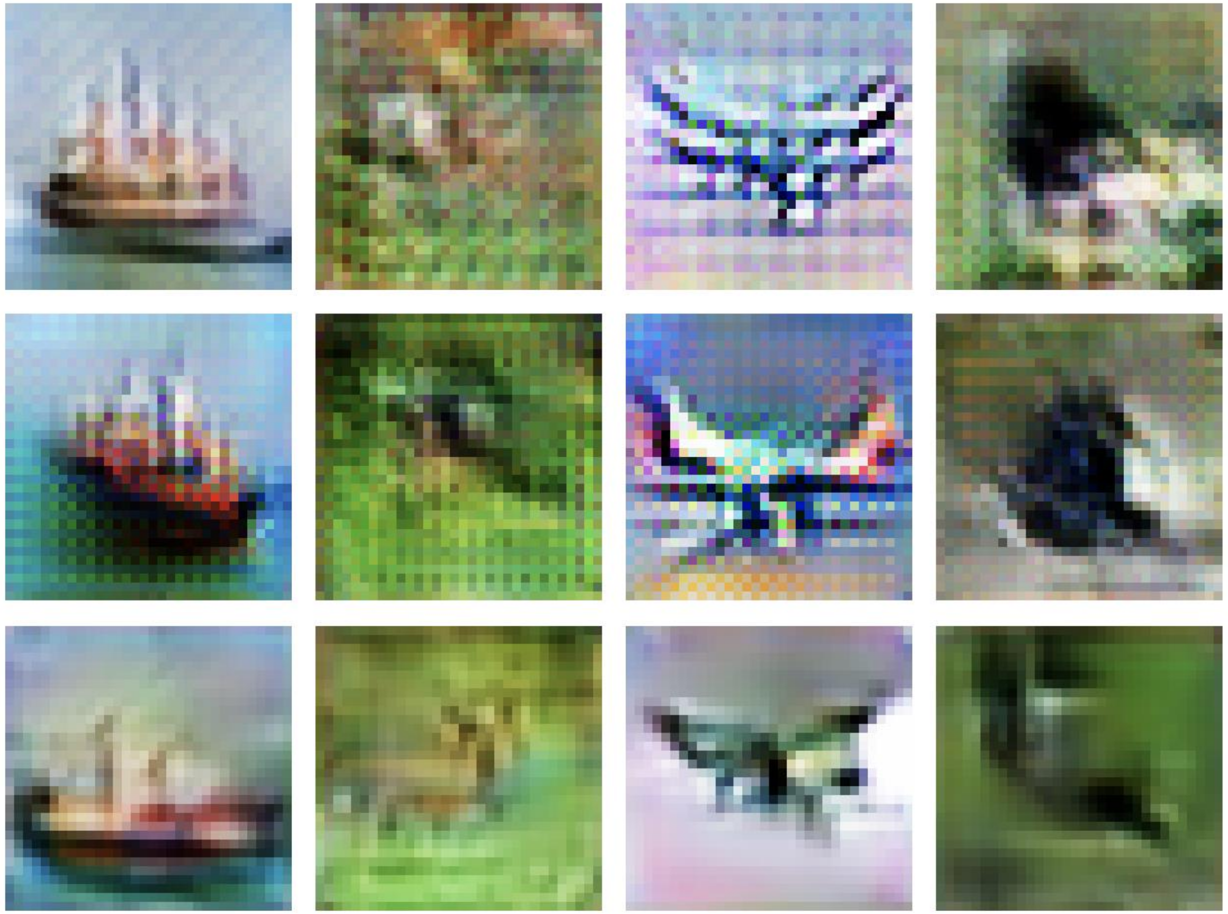


$$\begin{bmatrix} a+\frac{1}{2}b & \frac{1}{2}b+c & & \\ \frac{1}{2}a & \frac{1}{2}a+b+\frac{1}{2}c & \frac{1}{2}c & \\ & a+\frac{1}{2}b & \frac{1}{2}b+c & \\ & \frac{1}{2}a & \frac{1}{2}a+b+\frac{1}{2}c & \frac{1}{2}c \end{bmatrix}$$

Bilinear-Resize Convolution

https://distill.pub/2016/deconv-checkerboard/

Checkerboard artifact——upsample layer



Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

All layers use resize-convolution.
No artifacts.

Checkerboard artifact——downsample layer

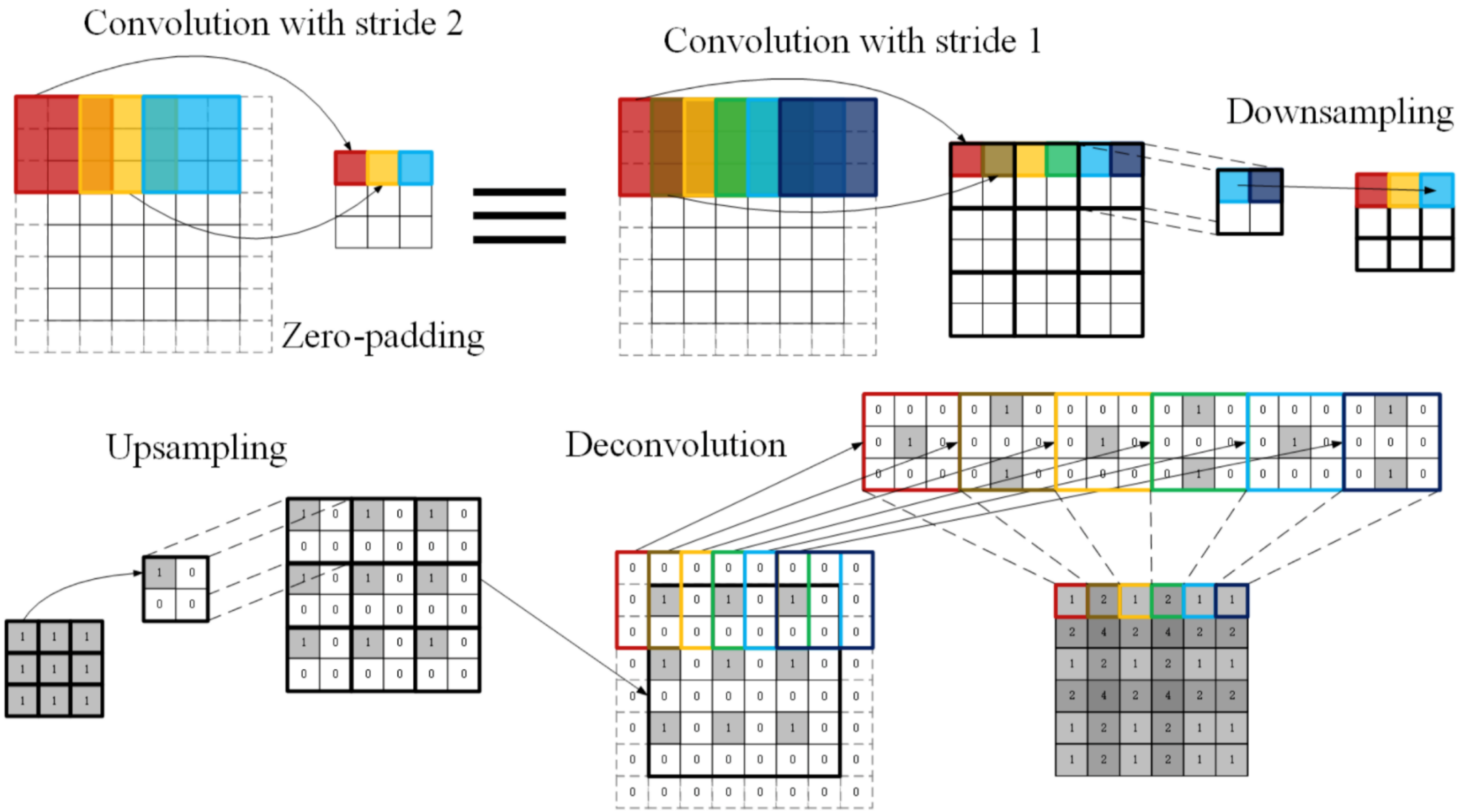
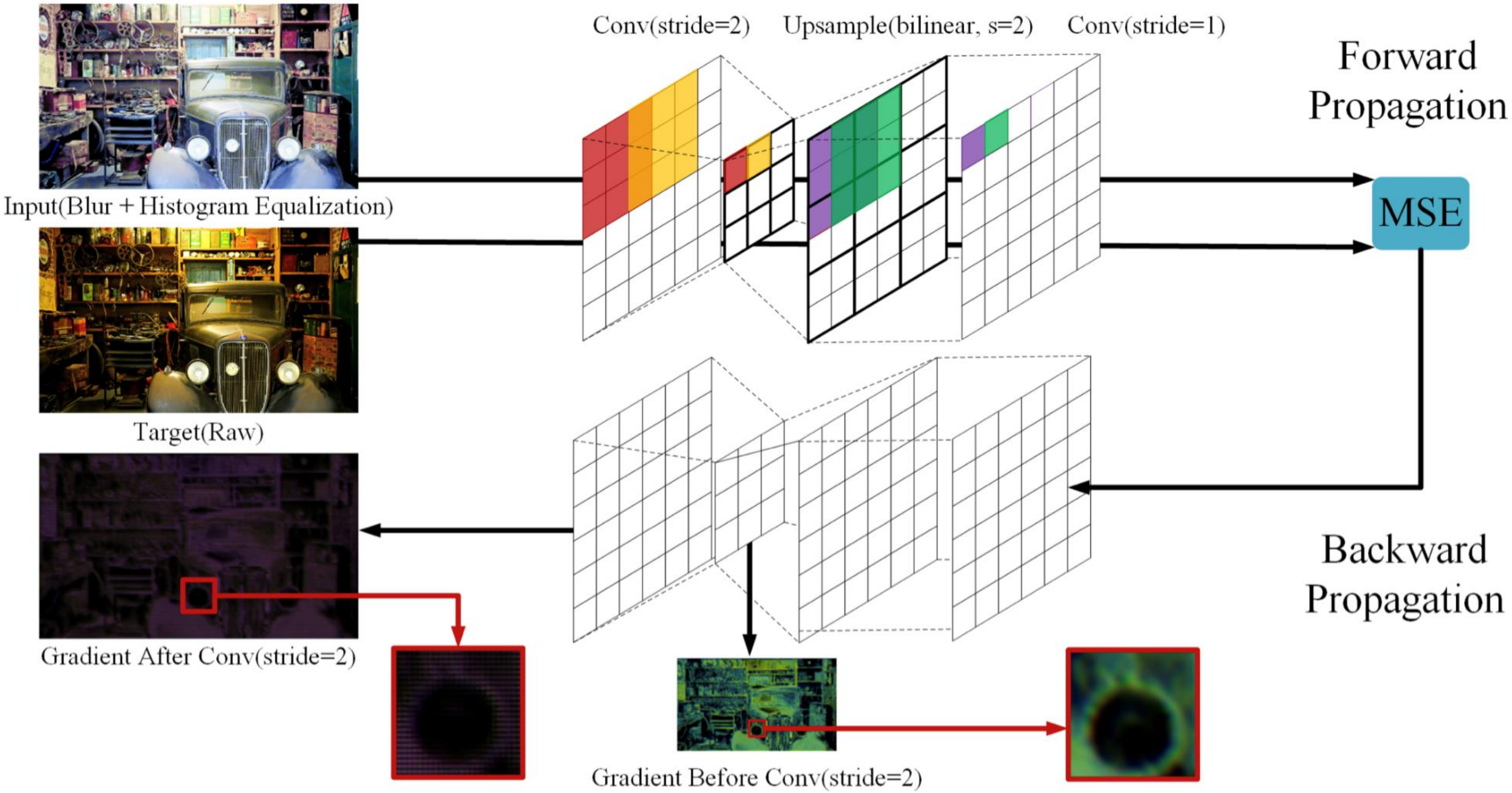


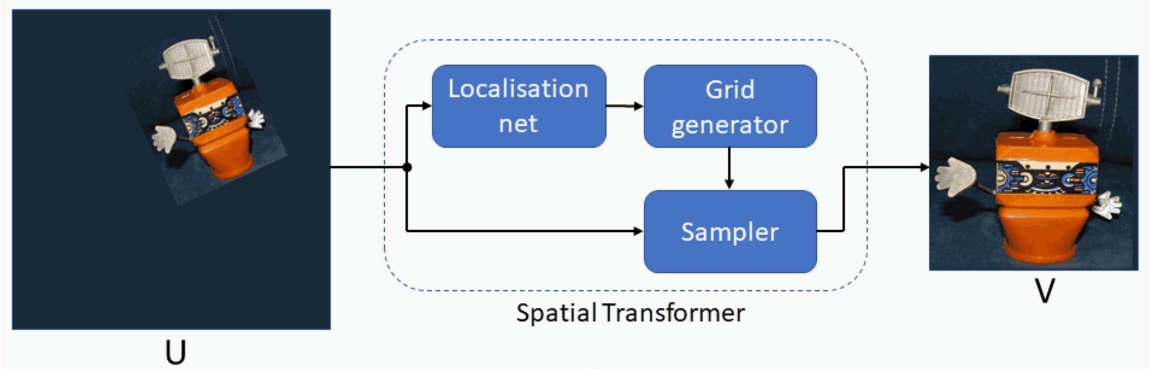
Figure 2. Forward propagation and backward propagation of convolution with stride 2.

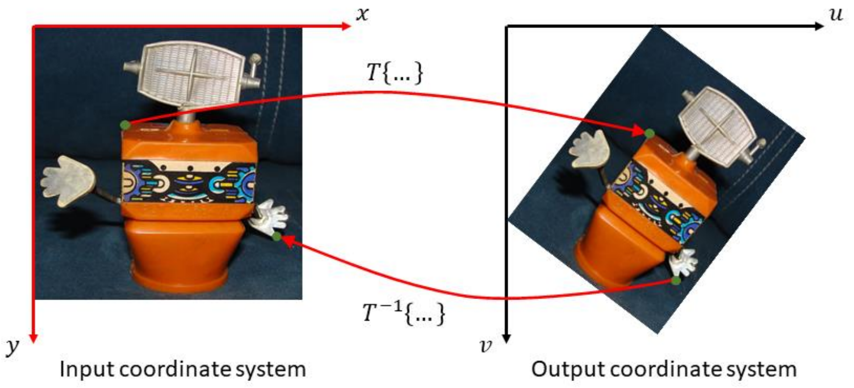
Traditional CV + Deep Learning

Checkerboard artifact——downsample layer



Spatial Transformer Networks





Spatial Transformer modules are a popular way to increase spatial invariance of a model against spatial transformations such as translation, scaling, rotation, cropping, as well as non-rigid deformations. They can be inserted into existing convolutional architectures: either immediately following the input or in deeper layers.

The forward transformation $T\{\ldots\}$ maps a location in input space to a location in output space:

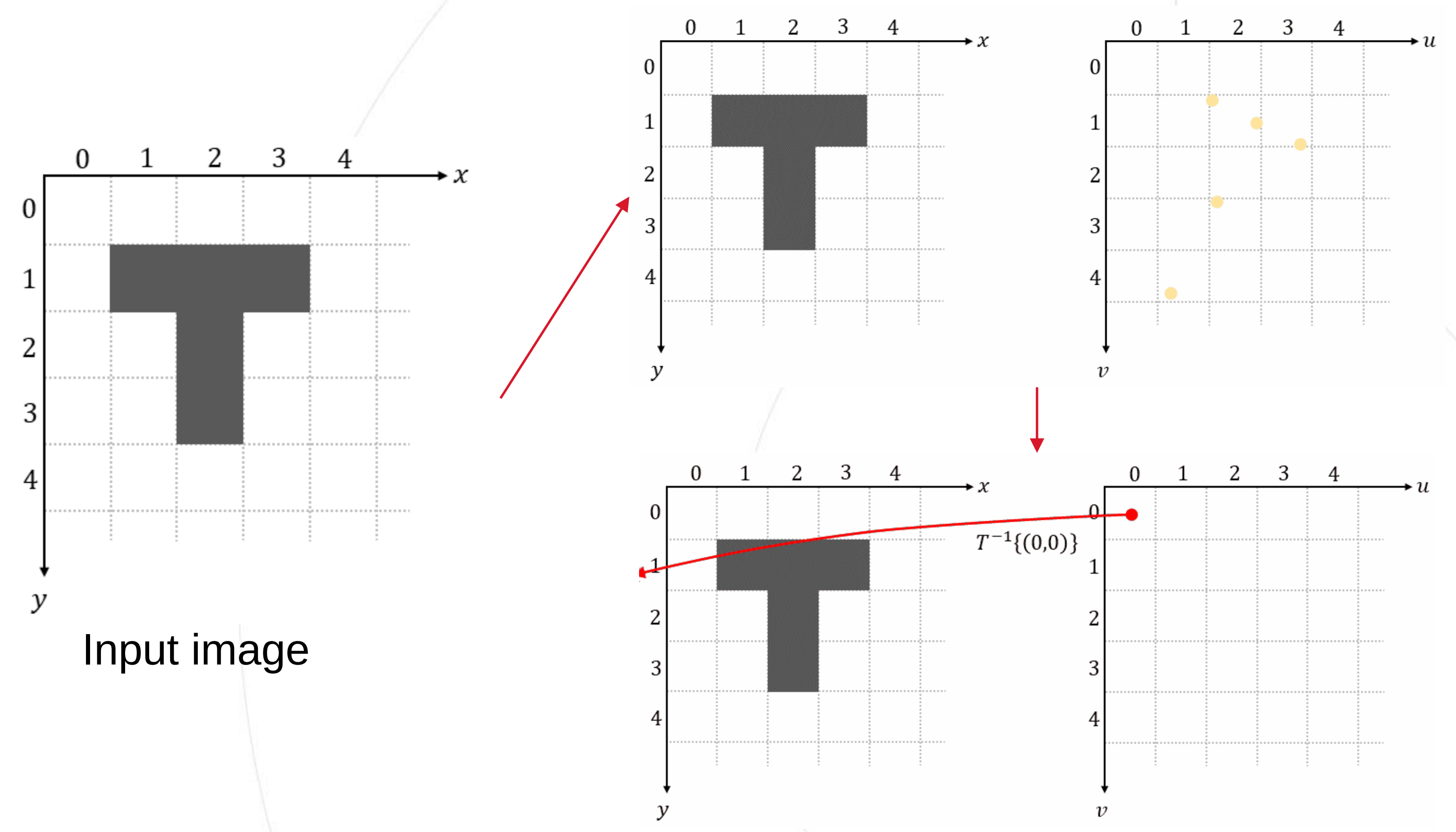
$$(v, u) = T\{(y, x)\}$$

The inverse transformation T-1{…} maps a location in output space back to a location in input space:

$$(y, x) = T^{-1}\{(v, u)\}$$

https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

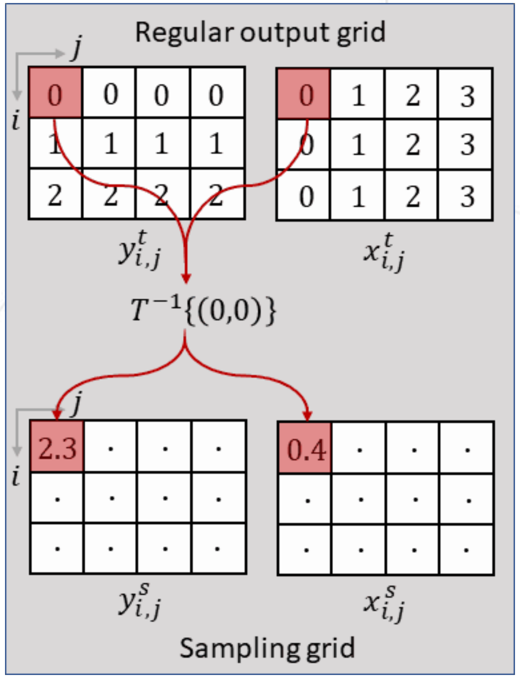
Spatial Transformer Networks



https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

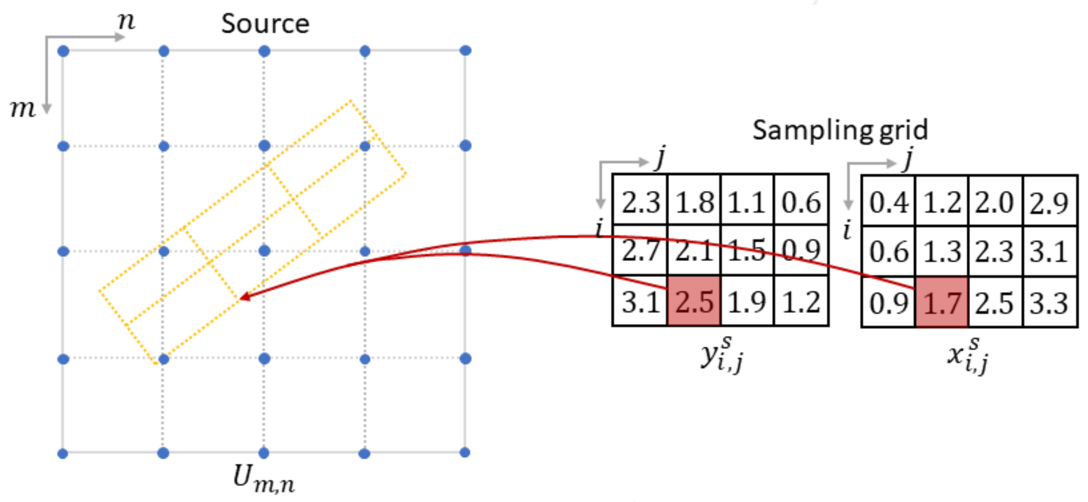
Spatial Transformer Networks

The **grid generator** iterates over the regular grid of the output/target image and uses the inverse transformation T-1{…} to calculate the corresponding (usually non-integer) sample positions in the input/source image:



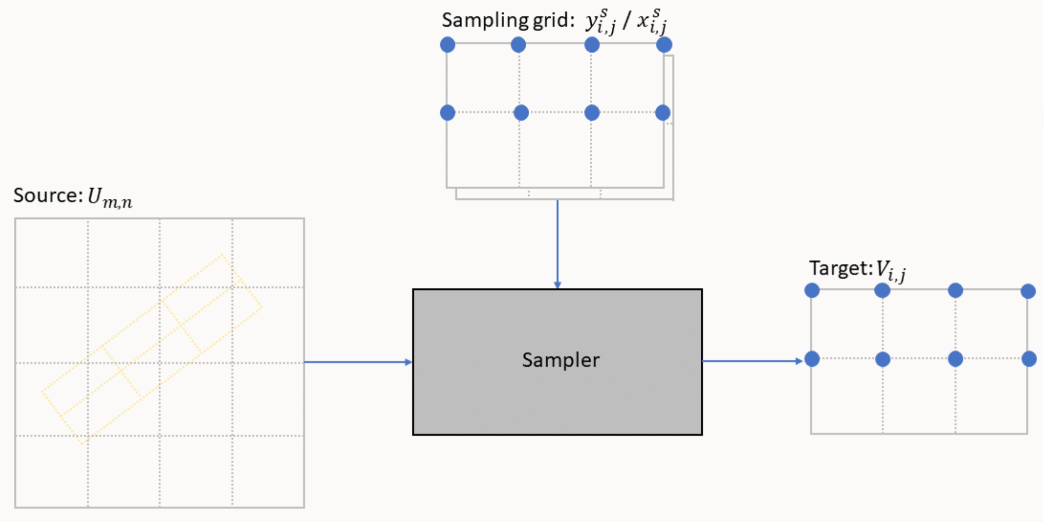
https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

Spatial Transformer Networks



The output of the grid generator is the so called **sampling grid**, which is a set of points where the input map will be sampled to produce the spatially transformed output:

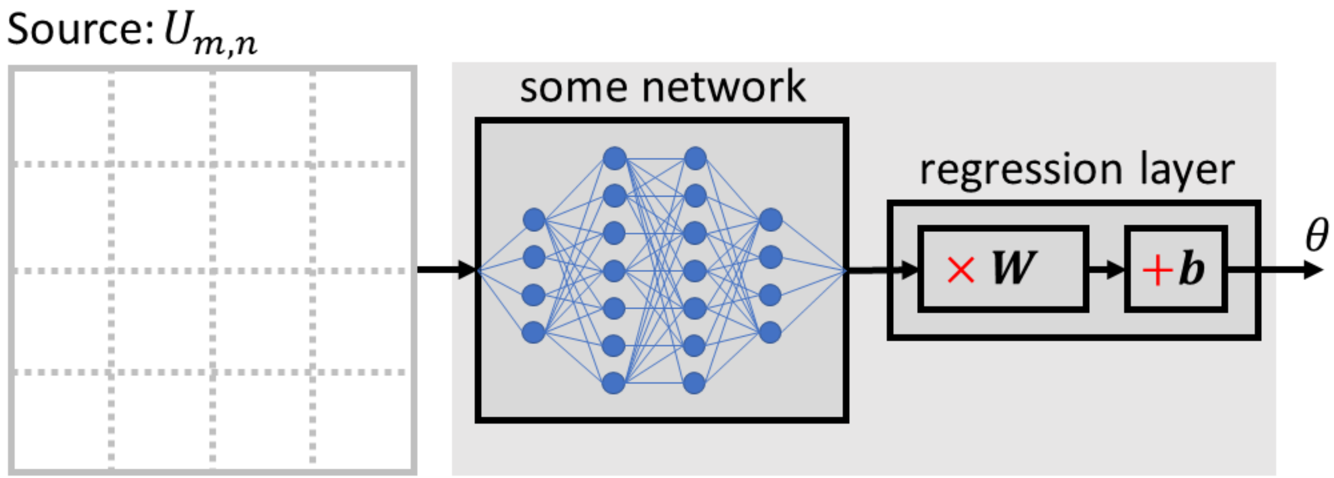
$$(y^s_{i,j}, x^s_{i,j})$$



The **sampler** iterates over the entries of the **sampling grid** and extracts the corresponding pixel values from the input map using bilinear interpolation:

https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

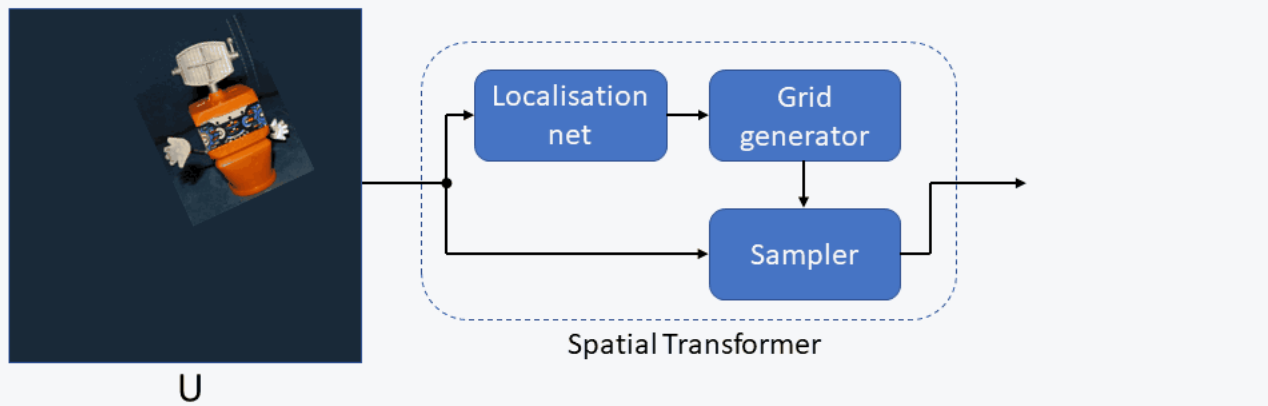
Spatial Transformer Networks



The task of the **localisation network** is to find parameters θ of the inverse transformation T-1{…}, which puts the input feature map to a canonical pose, thus simplify recognition in the following layers.
The **localisation network** can take any form, such as a fully-connected network or a convolutional network, but should include a final regression layer to produce the transformation parameters θ

https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

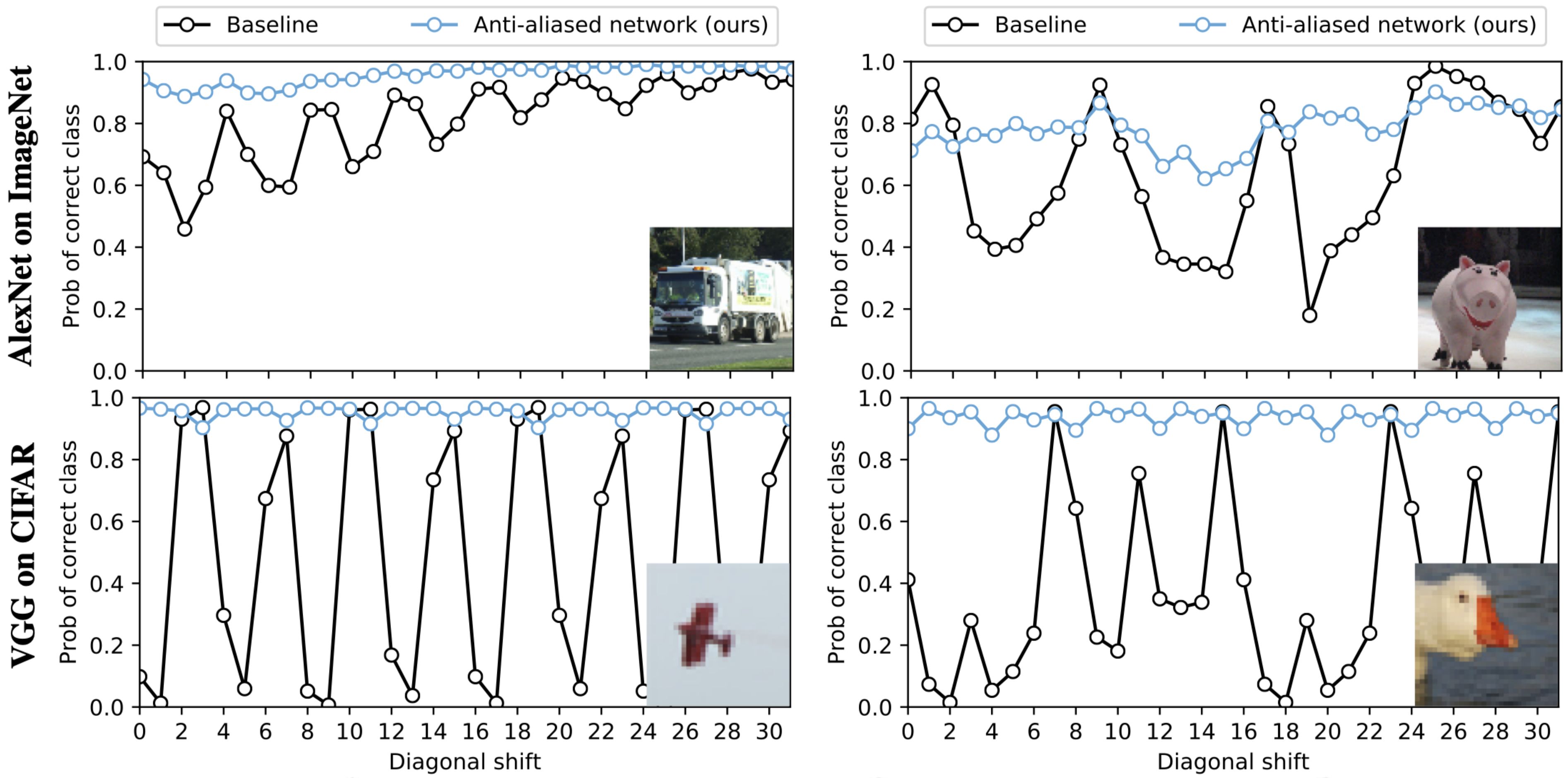
Spatial Transformer Networks



The input feature map U is first passed to the **localisation network**, which regresses the appropriate transformation parameters θ. The **grid generator** then uses the transformation parameters θ to produce the **sampling grid**, which is a set of points where the input feature map shall be sampled. Finally, the **sampler** takes both the input feature map and the **sampling grid** and using e.g. bilinear interpolation outputs the transformed feature map.

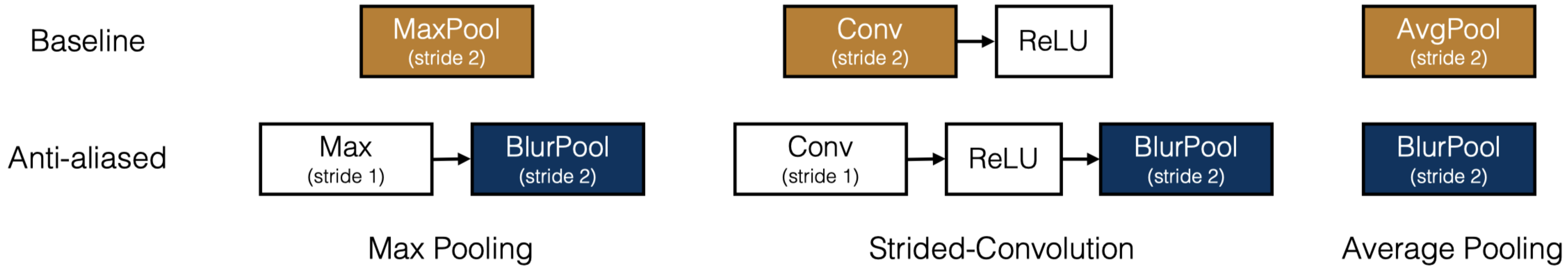
https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be

From Gauss filter to blurpool——instable examples to shifts



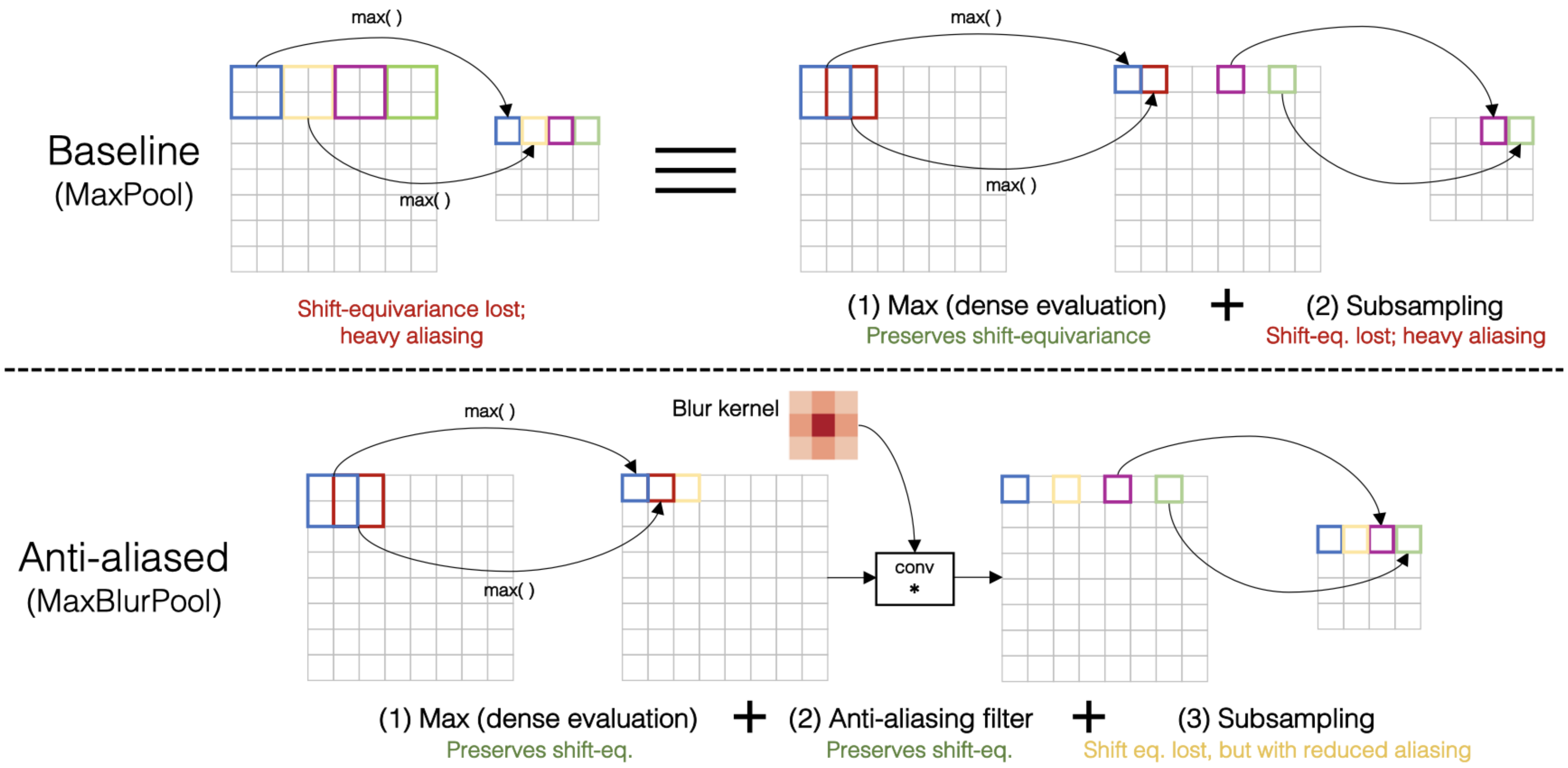
Traditional CV + Deep Learning

From gauss filter to blurpool——anti-aliasing strided layer



https://arxiv.org/abs/1904.11486?utm_source=aidigest&utm_medium&utm_campaign=63

From gauss filter to blurpool——anti-aliasing strided layer(MaxBlurPool)

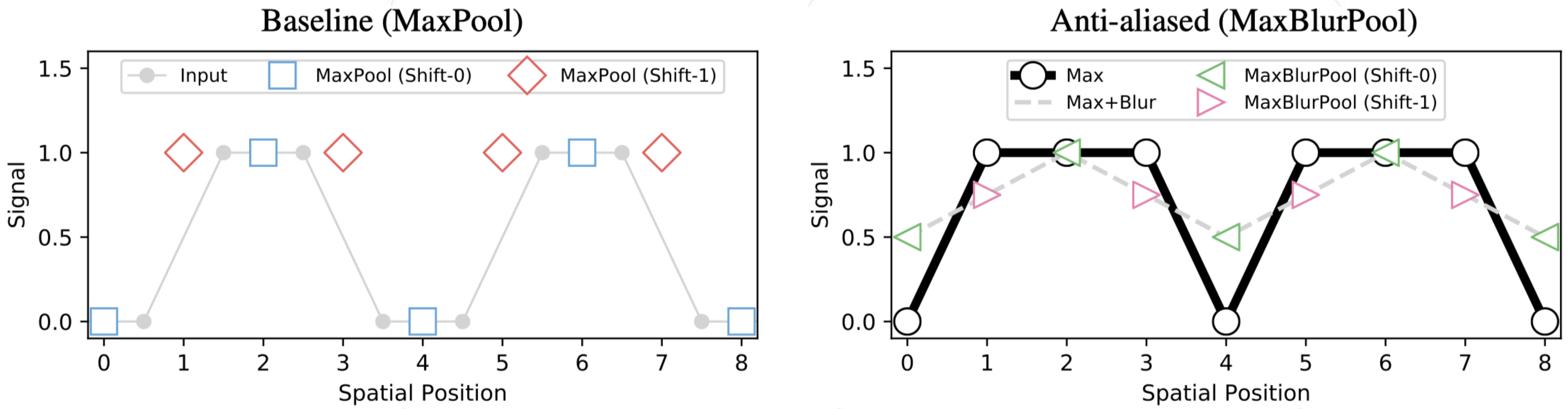


Traditional CV + Deep Learning

From gauss filter to blurpool——Shift-equivariance and invariance

$$\text{Shift}_{\Delta h,\Delta w}(\widetilde{\mathcal{F}}(X)) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h,\Delta w}(X)) \quad \forall\,(\Delta h, \Delta w)$$

$$\widetilde{\mathcal{F}}(X) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h,\Delta w}(X)) \quad \forall\,(\Delta h, \Delta w)$$

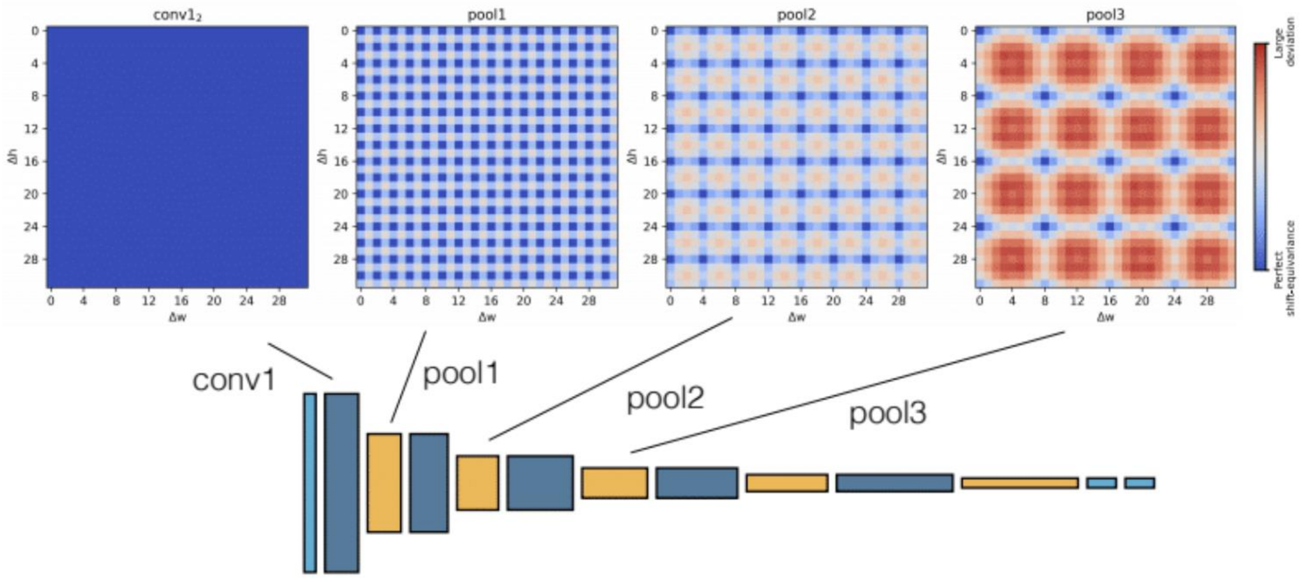


From gauss filter to blurpool——Shift-equivariance in CNN

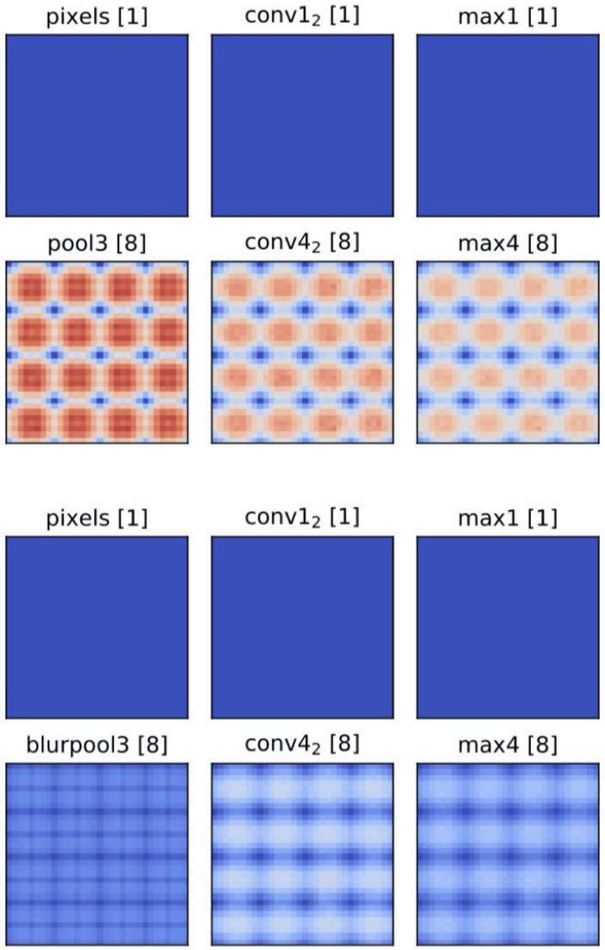
SHIFT-EQUIVARIANCE PER LAYER

VGG network on CIFAR classification
Circular convolution/shift (no made-up pixels)
Test shift-equivariance of each internal layer

$$\mathrm{dist}(F(\mathrm{Shift}_{\Delta h,\Delta w}(X)),\mathrm{Shift}_{\Delta h,\Delta w}(F(X)))$$

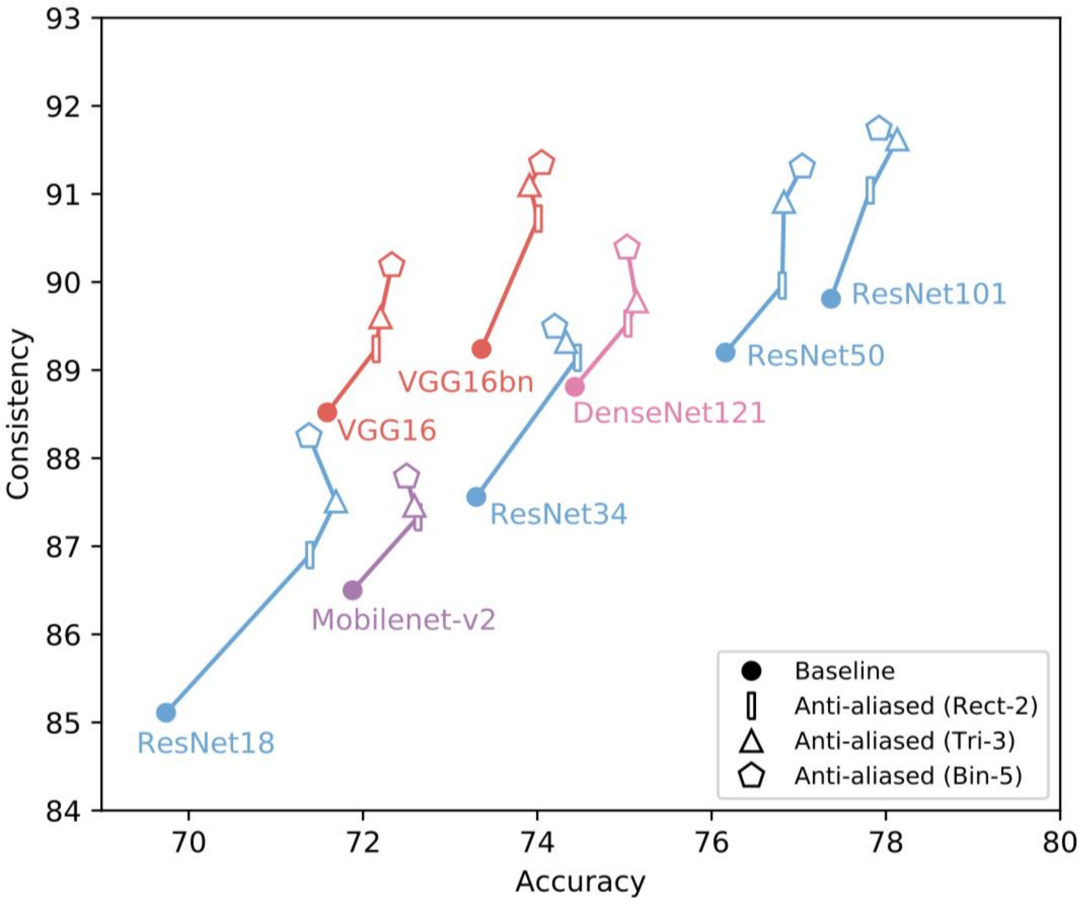


Shift-equivariance progressively lost in each downsampling due to **aliasing**



Traditional CV + Deep Learning

From gauss filter to blurpool——BlurPool result



Improved Stability + Robustness

Stability on ImageNet-P

- Data from [Hendrycks & Dietterich ICLR '19]
- Antialiasing theoretically motivated by shifts, but **increased stability to other perturbations** observed

	Flip Rate (FR) (lower is better)									
	Noise		Blur		Weather		Geometric			
	Gauss	Shot	Motion	Zoom	Snow	Bright	Translate	Rotate	Tilt	Scale
Baseline	14.04	17.38	6.00	4.29	7.54	3.03	4.86	6.79	4.01	11.32
Antialiased	12.39	15.22	5.44	3.72	6.76	3.15	3.78	5.67	3.44	9.45
% Reduction	11.81	12.42	9.27	13.28	10.28	-4.10	22.27	16.59	14.11	16.50

Robustness on ImageNet-C

- Performance degrades slower when images are corrupted, indicating **increased robustness**

	Corruption Error (CE) (lower is better)						
	Noise			Blur			
	Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom
Baseline	68.70	71.10	74.04	61.40	73.39	61.43	63.93
Antialiased	64.31	66.39	69.88	60.31	71.37	61.60	61.25
% Reduced	6.39	6.62	5.62	1.78	2.75	-0.28	4.19

	Weather				Digital			
	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	Jpeg
Baseline	67.76	62.08	54.61	32.04	61.25	55.24	55.24	46.32
Antialiased	66.82	59.82	51.84	31.51	58.12	55.29	50.81	42.84
% Reduced	1.39	3.64	5.07	1.65	5.11	-0.09	8.02	7.51

→ **Improved accuracy, stability, robustness**