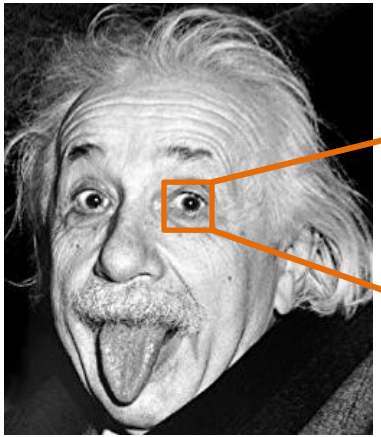
Filtering/Convolution/Gradient

IMAGE SPATIAL FILTERING ~ CORRELATION

$$I_f(i,j) = \sum_{k,l} I(i+k, j+l) z(k,l)$$



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

I

Image

$\otimes$

a	b	c
d	e	f
g	h	i

z

Filter

$=$

	y			

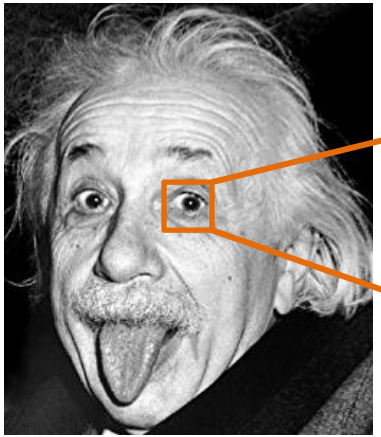
I_f

Filtered image

$$y = 2a + b + 4c + d + 2e + 2f + 3g + 3h + 5i$$

BOUNDARY

$$I_f(i,j) = \sum_{k,l} I(i+k, j+l) z(k,l)$$



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

I

Image

$\otimes$

a	b	c
d	e	f
g	h	i

z

Filter

$=$

y				

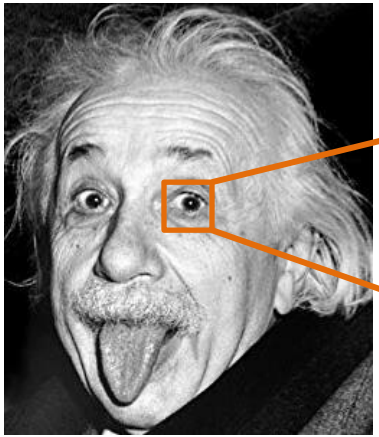
I_f

Filtered image

$y = ?$

BOUNDARY

$$I_f(i,j) = \sum_{k,l} I(i+k, j+l) z(k,l)$$



0	0	0	0	0	0	0
0	2	1	4	4	7	0
0	1	2	2	3	6	0
0	3	3	5	8	9	0
0	5	2	2	6	7	0
0	8	3	2	1	3	0
0	0	0	0	0	0	0

I

Image

Zero padding

$\otimes$

a	b	c
d	e	f
g	h	i

z

Filter

$=$

y				

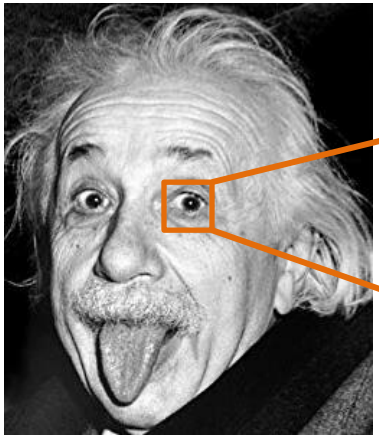
I_f

Filtered image

$$y = 2e + f + h + 2i$$

BOUNDARY

$$I_f(i,j) = \sum_{k,l} I(i+k, j+l) z(k,l)$$



0	0	0	0	0	0	0
0	2	1	4	4	7	0
0	1	2	2	3	6	0
0	3	3	5	8	9	0
0	5	2	2	6	7	0
0	8	3	2	1	3	0
0	0	0	0	0	0	0

I

Image

$\otimes$

a	b	c
d	e	f
g	h	i

z

Filter

$=$

				y

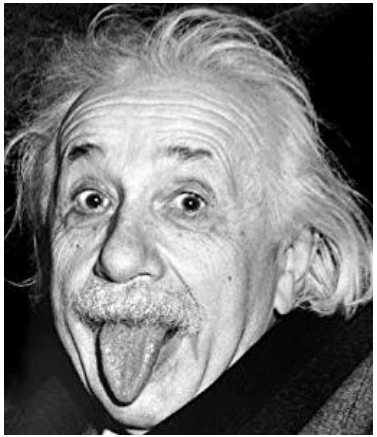
I_f

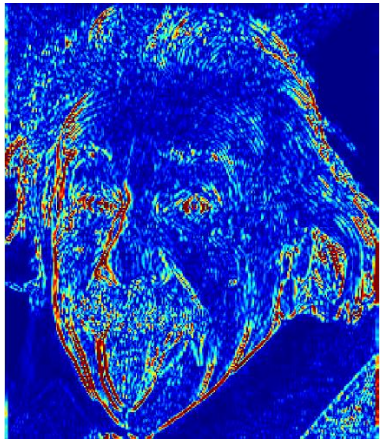
Filtered image

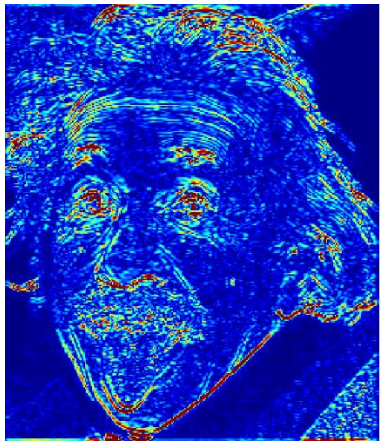
Zero padding

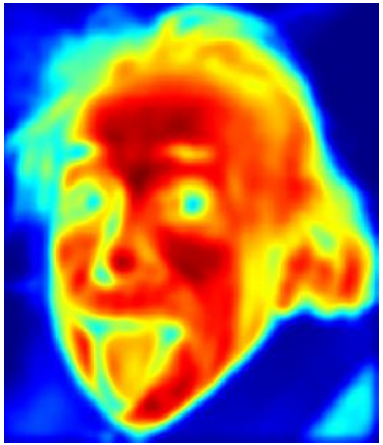
$$y = 8a + 9b + 6d + 7e + g + 3h$$

Filtering as a Feature Extraction

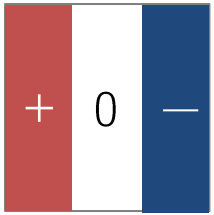


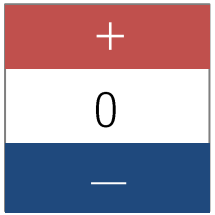


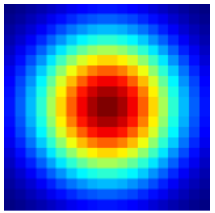




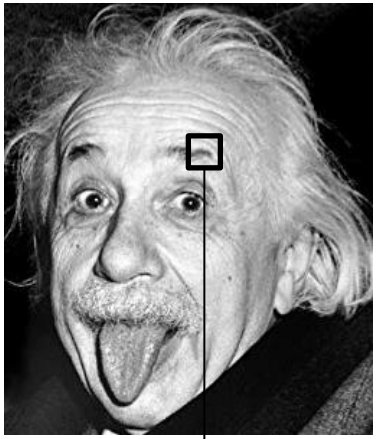
• • •

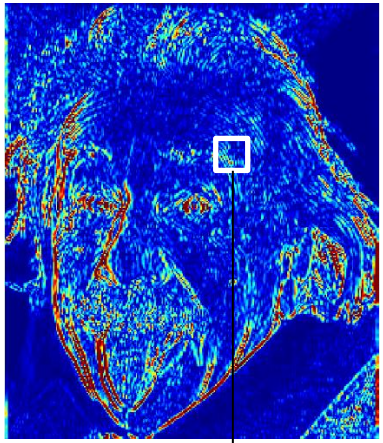


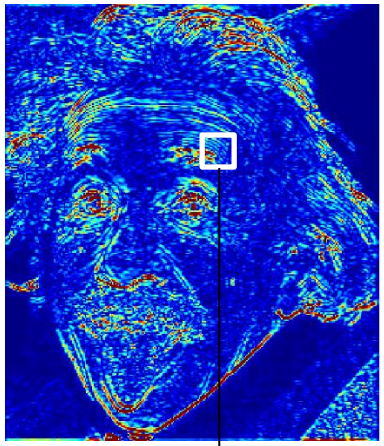


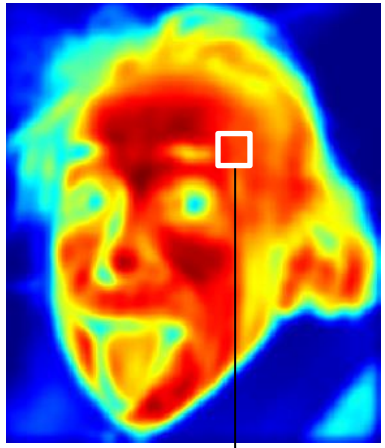


FILTERING AS A FEATURE EXTRACTION









$I(x_1) = (212, 200, 221)$

$f_1(x) = 0.3$

$f_2(x) = -0.1$

$f_3(x) = 0.9$

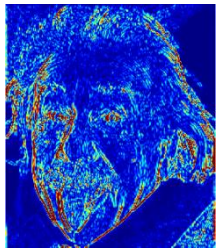
Correlation vs. Convolution

Image correlation:

$$I \otimes z = I_f$$

$$\sum_{k,l} I(i+k, j+l) z(k,l) = I_f(i,j)$$

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$ **F** $=$ 

Correlation vs. Convolution

Image correlation:

$$I \otimes z = I_f$$

$$\sum_{k,l} I(i+k, j+l) z(k,l) = I_f(i,j)$$

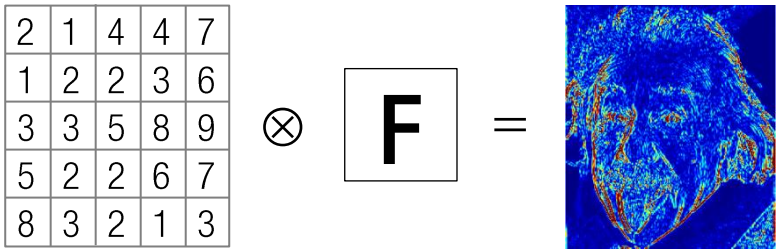
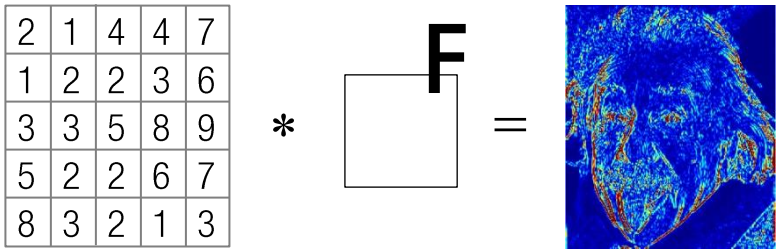


Image convolution:

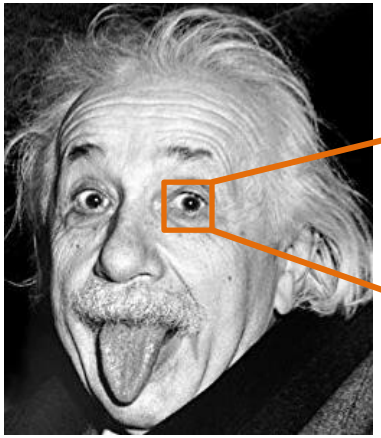
$$I * z = J$$

$$\sum_{k,l} I(i-k, j-l) z(k,l) = J(i,j)$$



Flip the filter in both dimension (bottom to top, right to left)

Correlation vs. Convolution



Correlation

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

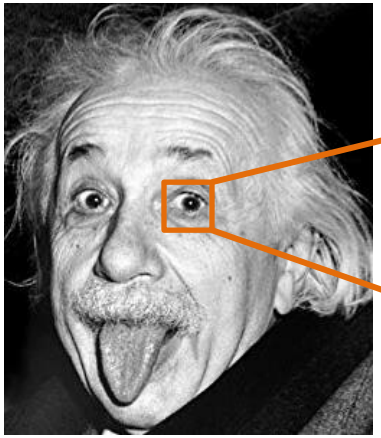
a	b	c
d	e	f
g	h	i

$=$

	y			

$$y = 2a + b + 4c + d + 2e + 2f + 3g + 3h + 5i$$

Correlation vs. Convolution



Correlation

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

a	b	c
d	e	f
g	h	i

$=$

	y			

$$y = 2a + b + 4c + d + 2e + 2f + 3g + 3h + 5i$$

Convolution

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

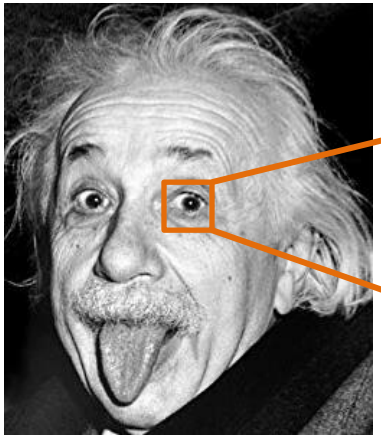
$*$

a	b	c
d	e	f
g	h	i

$=$

	z			

Correlation vs. Convolution



Correlation

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

a	b	c
d	e	f
g	h	i

$=$

	y			

$$y = 2a + b + 4c + d + 2e + 2f + 3g + 3h + 5i$$

Convolution

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

Flip the filter

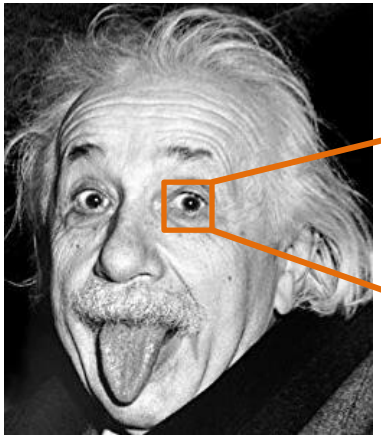
i	h	g
f	e	d
c	b	a

$=$

	z			

$$z = 2i + h + 4g + f + 2e + 2d + 3c + 3b + 5a$$

Correlation vs. Convolution



Correlation

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

a	b	c
d	e	f
g	h	i

$=$

		y		

$$y = 2a + 2b + 3c + 3d + 5e + 8f + 2g + 3h + 6i$$

Convolution

2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$\otimes$

Flip the filter

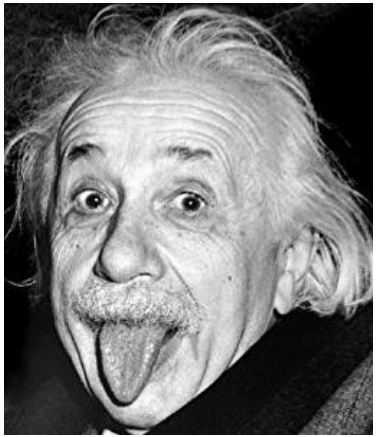
i	h	g
f	e	d
c	b	a

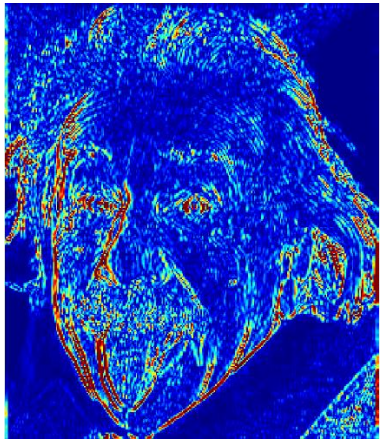
$=$

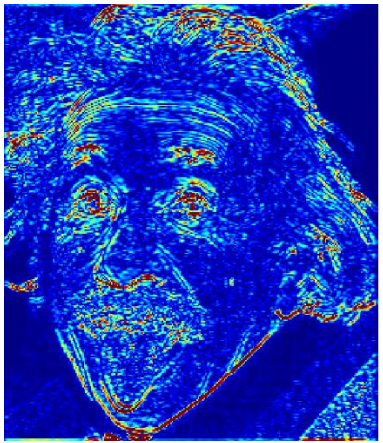
		z		

$$z = 2i + 2h + 3h + 3f + 5e + 8d + 2c + 3b + 6a$$

Recall: Edge Response







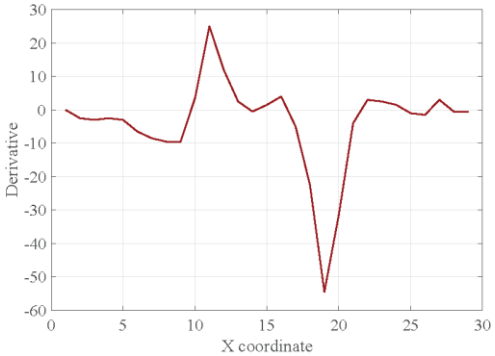
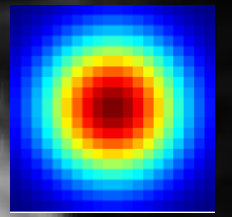


Image differentiation can be used for edge detection but it is VERY noisy.

How to detect edge reliably?

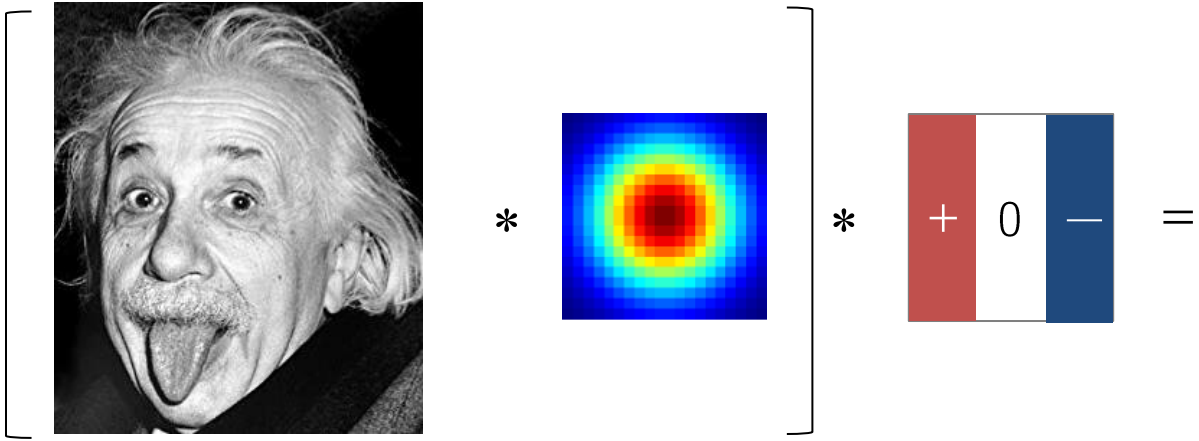
Recall: Gaussian Blurring~Denoising



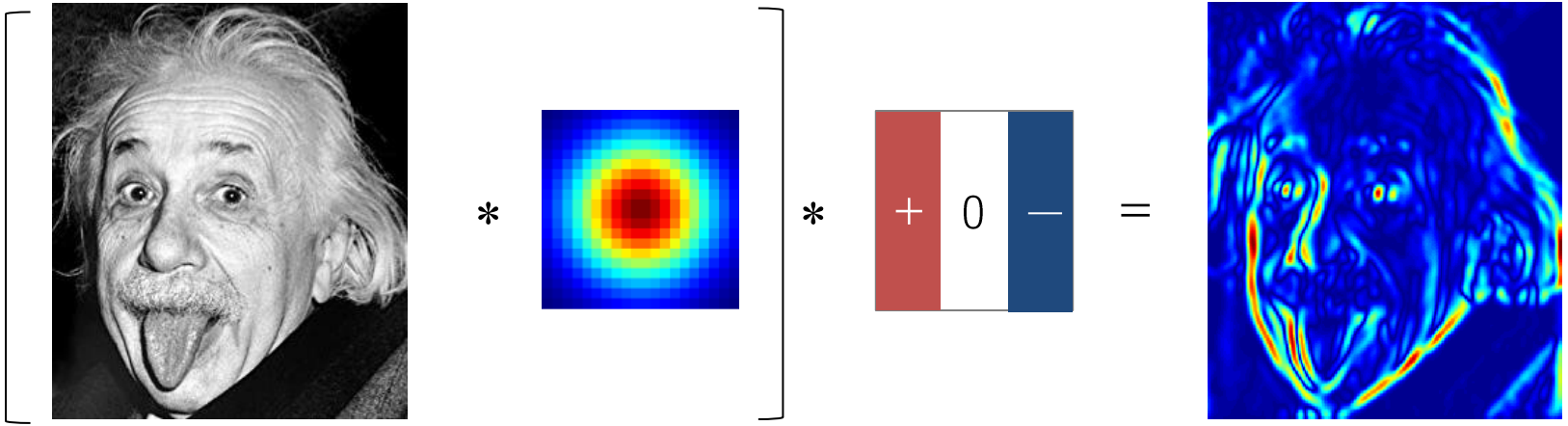
Noisy image

Denoised image

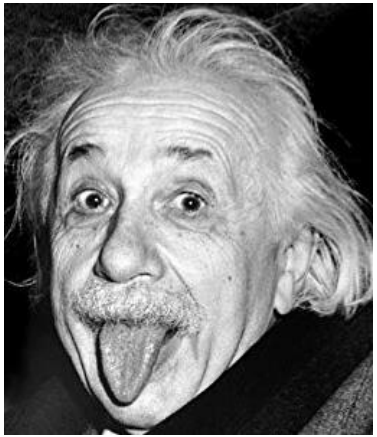
Strategy: Denoise and Differentiate



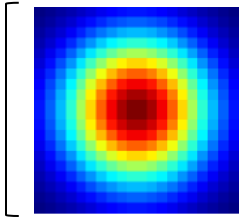
Strategy: Denoise and Differentiate



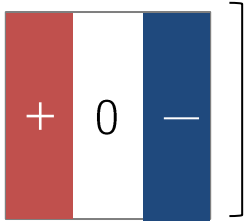
ASSOCIATIVITY



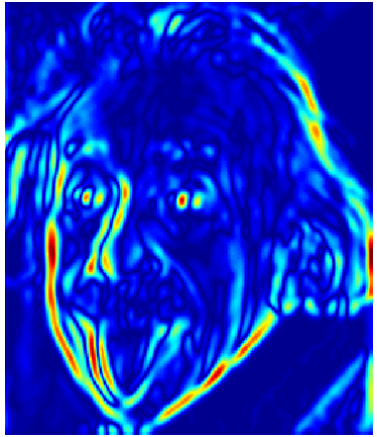
$*$



$*$

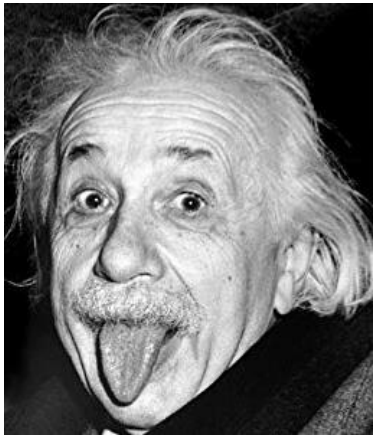


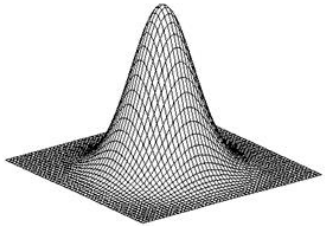
$=$



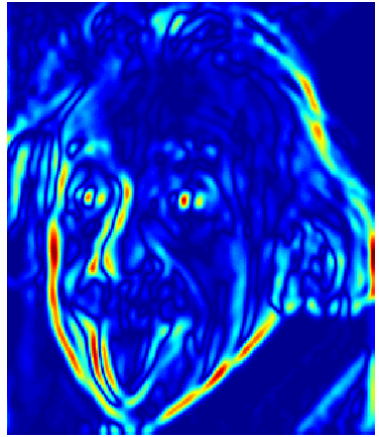
$$\frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}} \qquad \frac{\partial}{\partial u}$$

COMMUTATIVITY



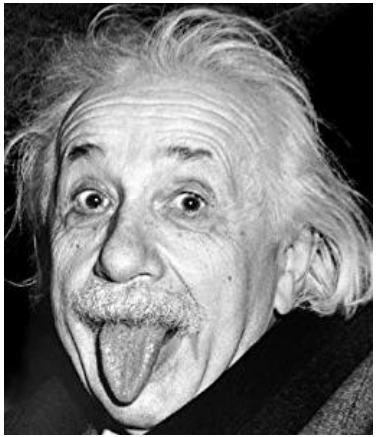


$$* \left[\begin{array}{|c|c|c|} \hline + & 0 & - \\ \hline \end{array} \right] * \left[\begin{array}{|c|} \hline \\ \hline \end{array} \right] =$$

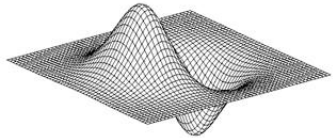


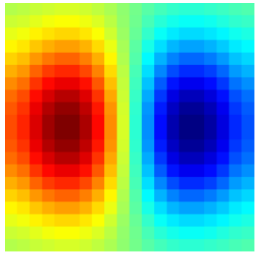
$$\frac{\partial}{\partial u} \qquad \frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}}$$

SOBEL FILTER



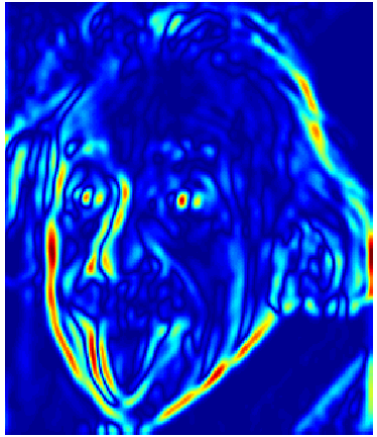
*





$$\frac{\partial}{\partial u}\frac{1}{2\pi\sigma^2}e^{-\frac{k^2+l^2}{2\sigma^2}}$$

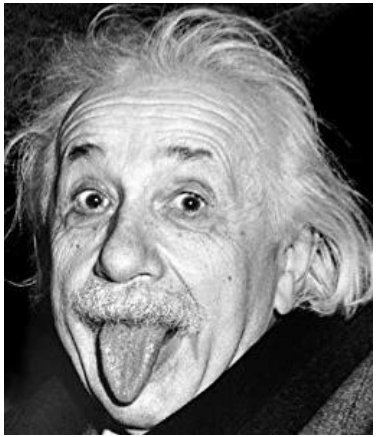
=



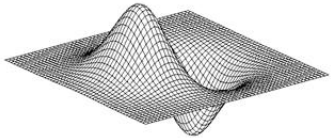
Sobel filter: derivative of Gaussian filter, e.g.,

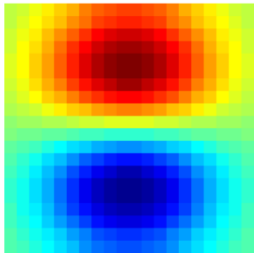
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

SOBEL FILTER



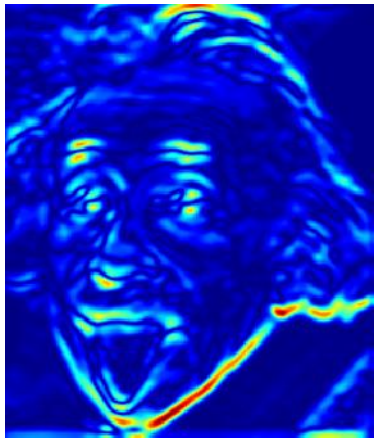
*





$$\frac{\partial}{\partial v}\frac{1}{2\pi\sigma^2}e^{-\frac{k^2+l^2}{2\sigma^2}}$$

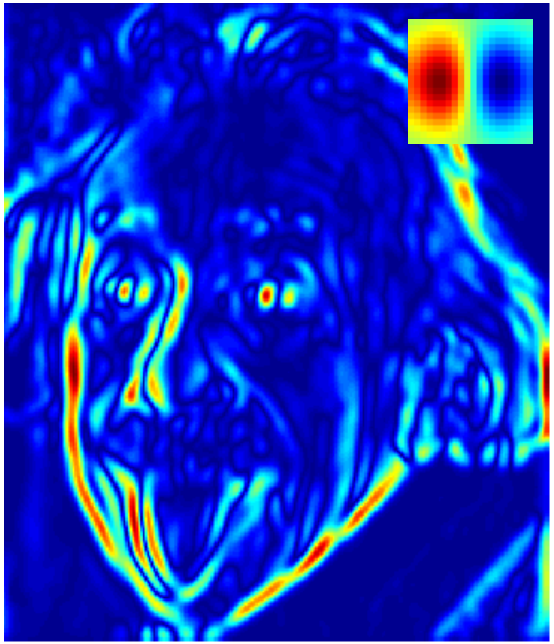
=

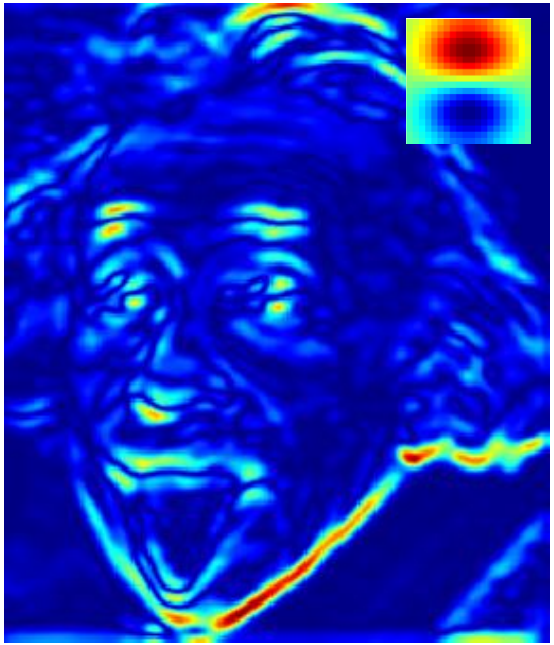


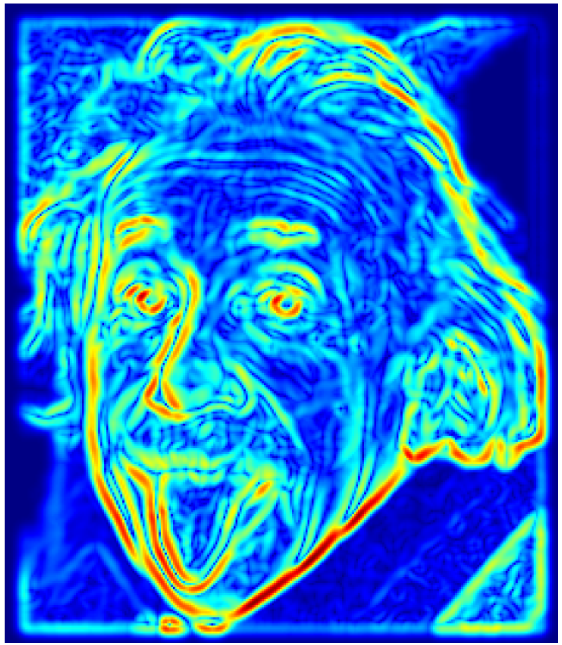
Sobel filter: derivative of Gaussian filter, e.g.,

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

Image Gradient Magnitude





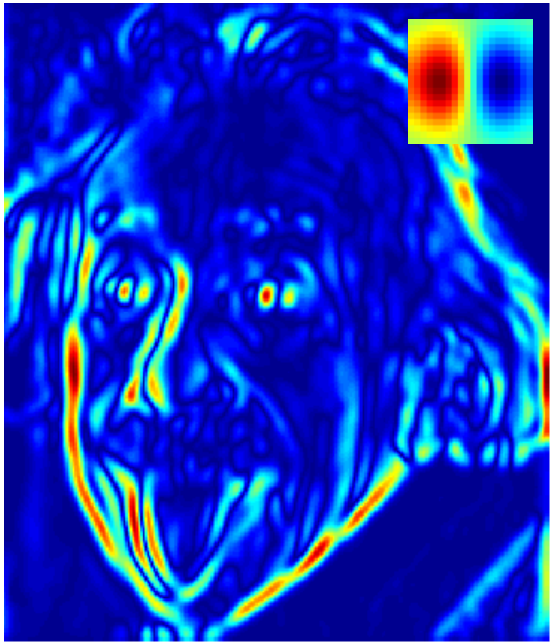


$$\frac{\partial I}{\partial u}$$

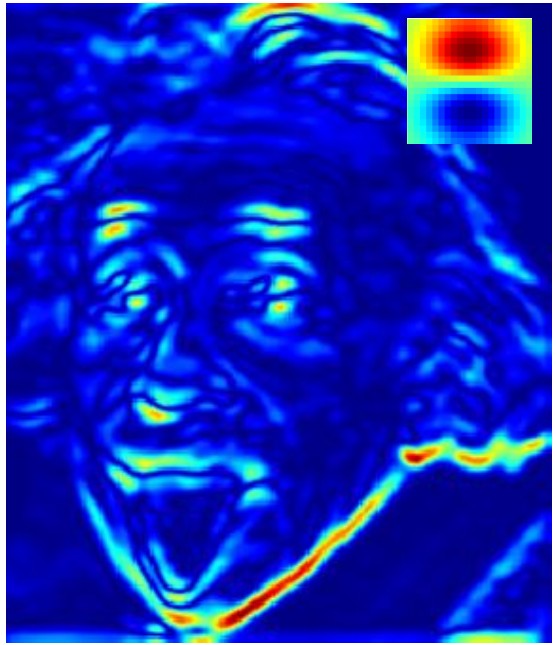
$$\frac{\partial I}{\partial v}$$

$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial u}\right)^2 + \left(\frac{\partial I}{\partial v}\right)^2}$$

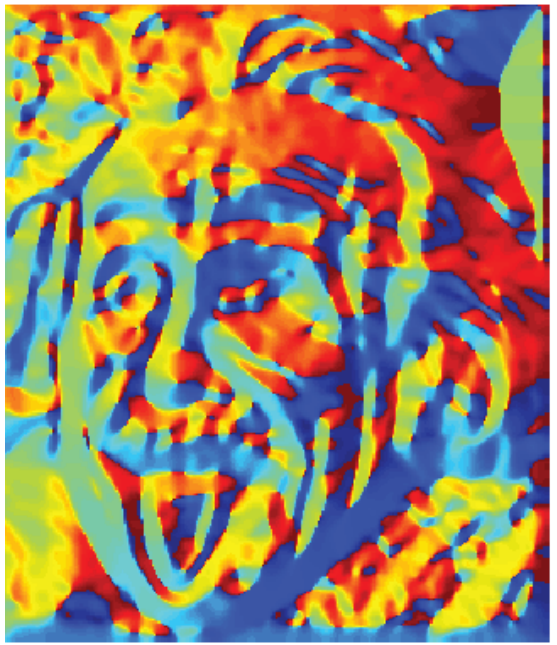
IMAGE GRADIENT DIRECTION



$$\frac{\partial I}{\partial u}$$

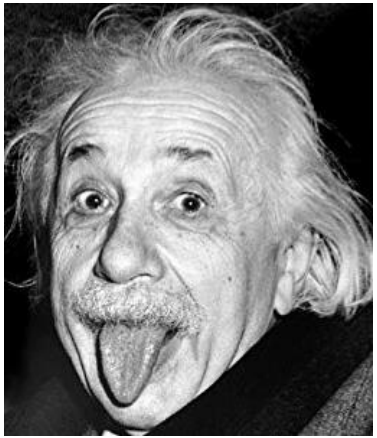


$$\frac{\partial I}{\partial v}$$

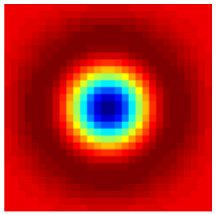


$$\measuredangle \nabla I = \tan^{-1}\left(\frac{\partial I}{\partial v} / \frac{\partial I}{\partial u}\right)$$

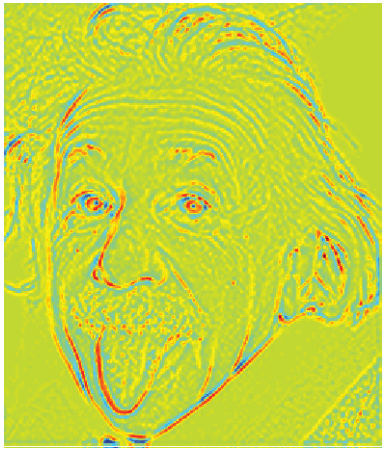
LAPLACIAN

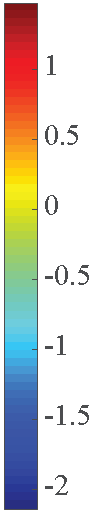


*

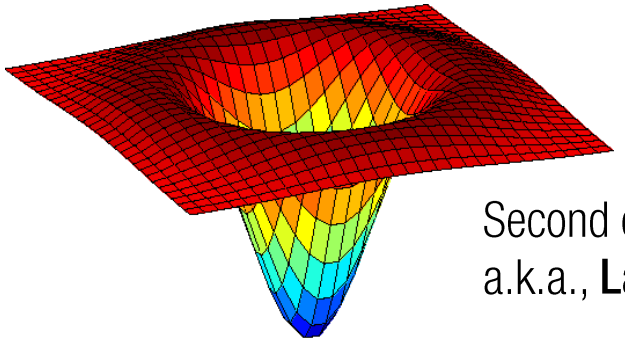


=



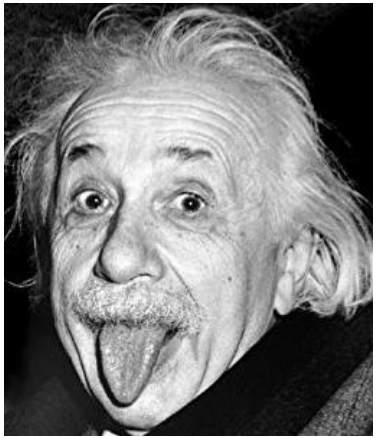


$$\nabla \cdot \nabla G = \nabla \left(\frac{\partial G}{\partial u}\mathbf{i} + \frac{\partial G}{\partial v}\mathbf{j} \right) = \frac{\partial^2 G}{\partial u^2} + \frac{\partial^2 G}{\partial v^2}$$



Second order derivative of Gaussian,
a.k.a., **Laplacian of Gaussian**

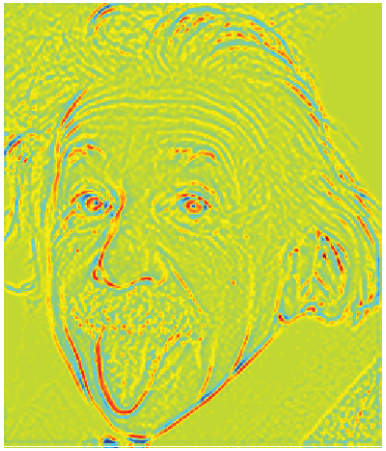
LAPLACIAN

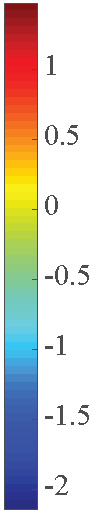


$*$

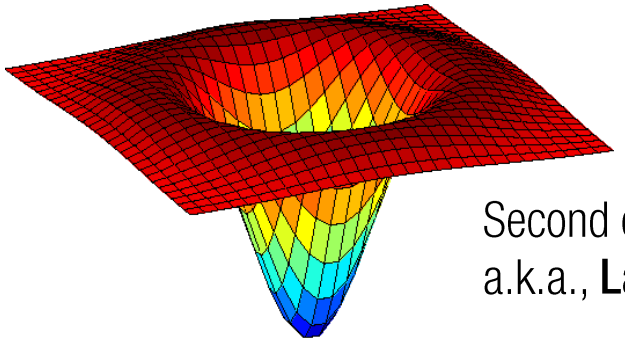
0	1	0
1	-4	1
0	1	0

$=$



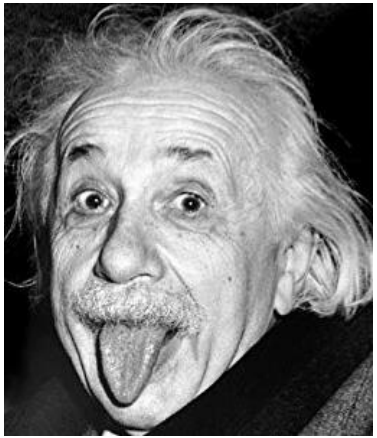


$$\nabla \cdot \nabla G = \nabla \left(\frac{\partial G}{\partial u}\mathbf{i} + \frac{\partial G}{\partial v}\mathbf{j} \right) = \frac{\partial^2 G}{\partial u^2} + \frac{\partial^2 G}{\partial v^2}$$



Second order derivative of Gaussian,
a.k.a., **Laplacian of Gaussian**

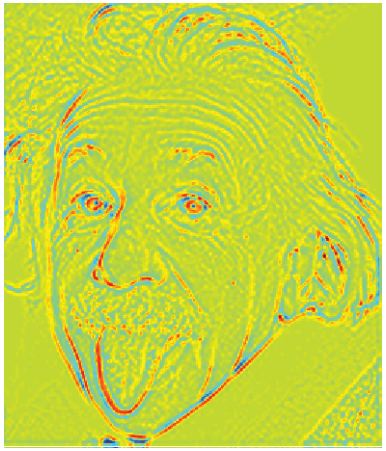
LAPLACIAN

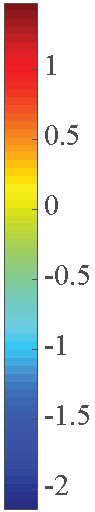


*

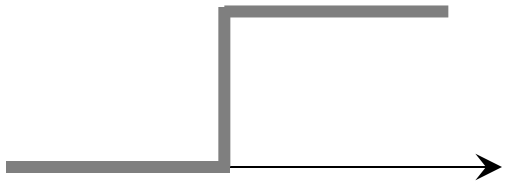
0	1	0
1	-4	1
0	1	0

=



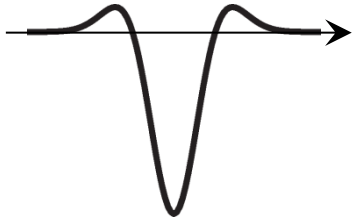


$$\nabla \cdot \nabla G = \nabla\left(\frac{\partial G}{\partial u}\mathbf{i} + \frac{\partial G}{\partial v}\mathbf{j}\right) = \frac{\partial^2 G}{\partial u^2} + \frac{\partial^2 G}{\partial v^2}$$

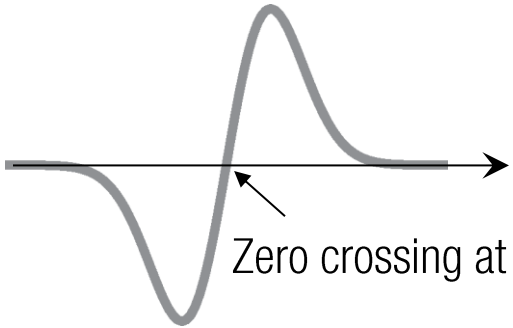


Edge

*

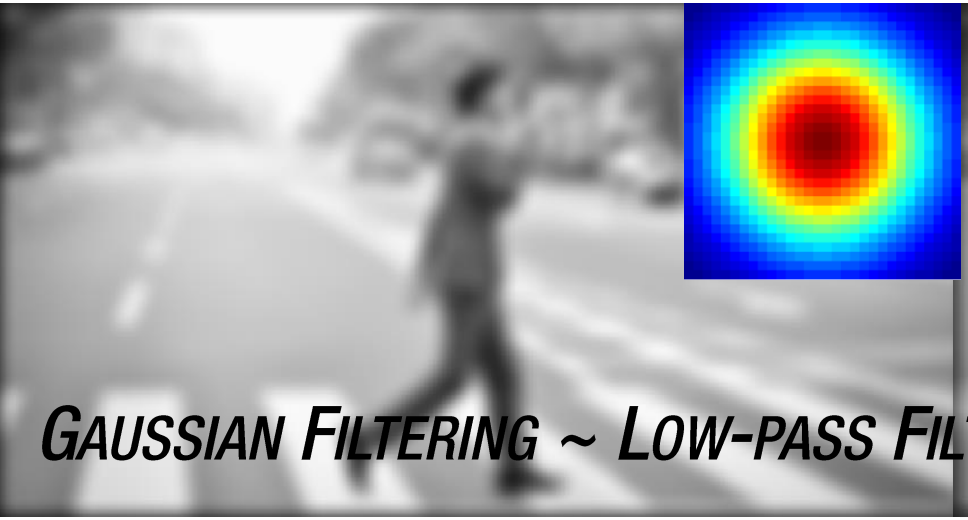


=

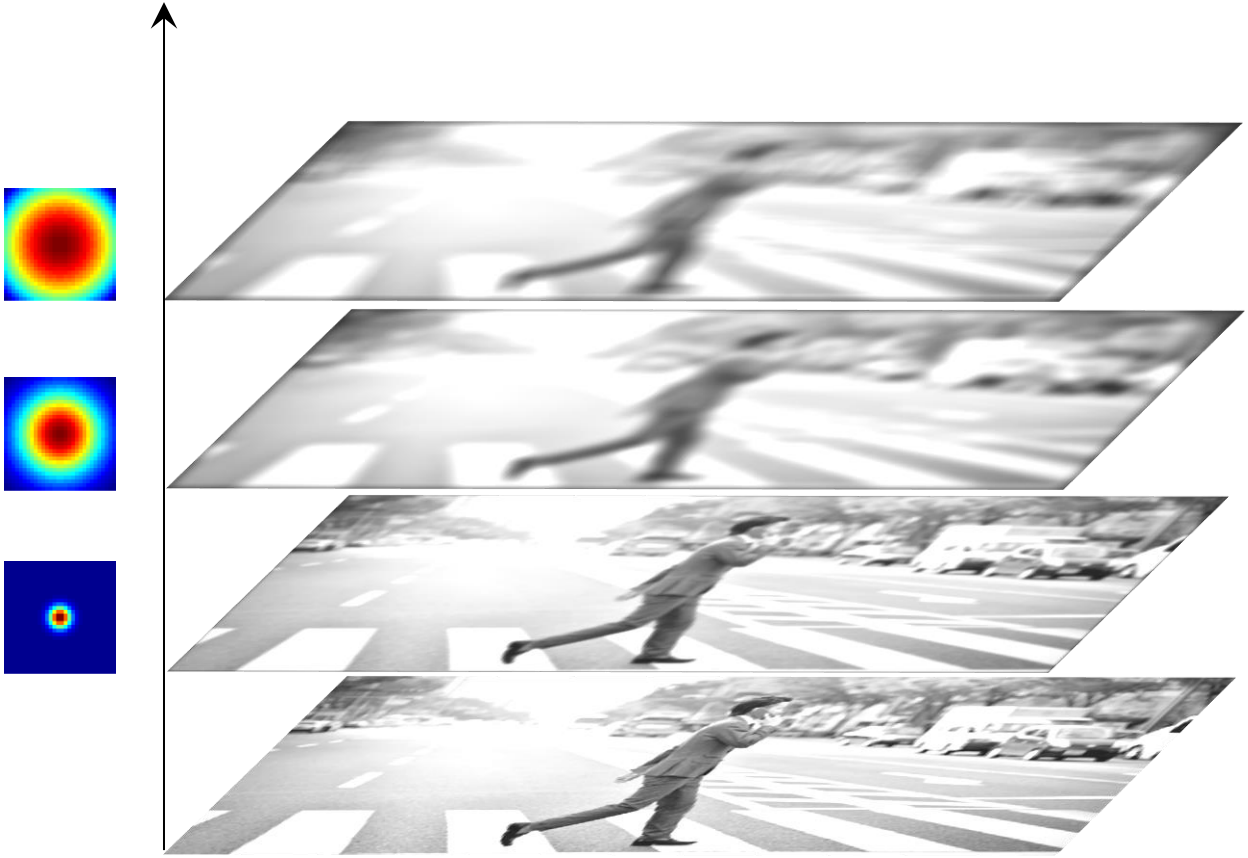


Zero crossing at edge

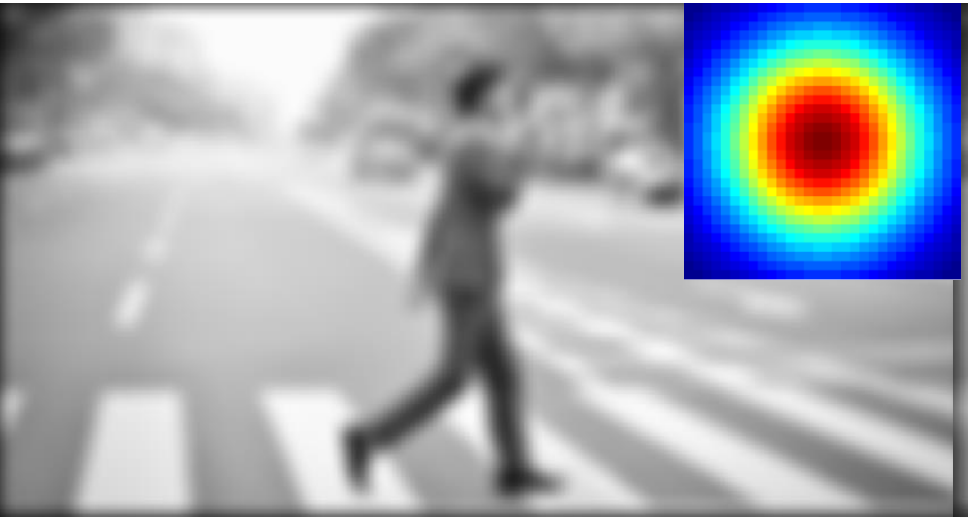
Pyramid

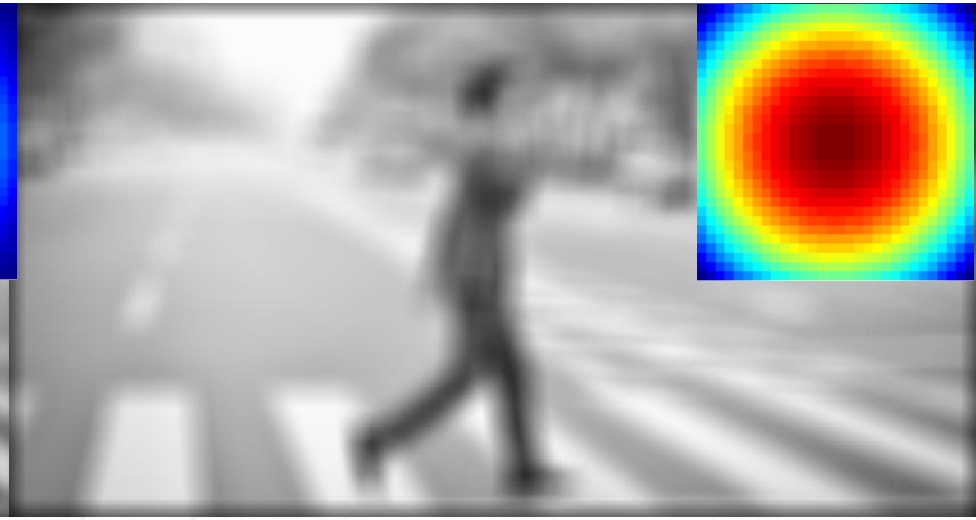


Gaussian Filtering ~ Low-pass Filtering ~ Image Blurring

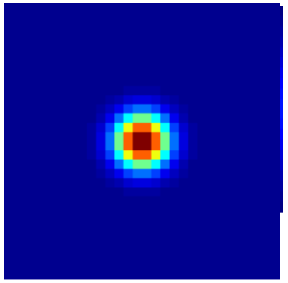


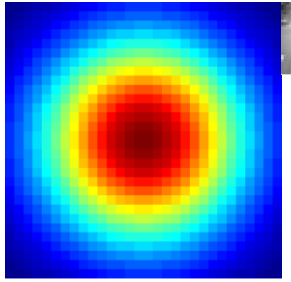
MULTI-DIMENSIONAL IMAGE REPRESENTATION

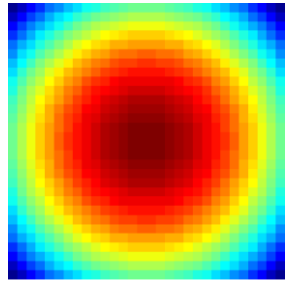








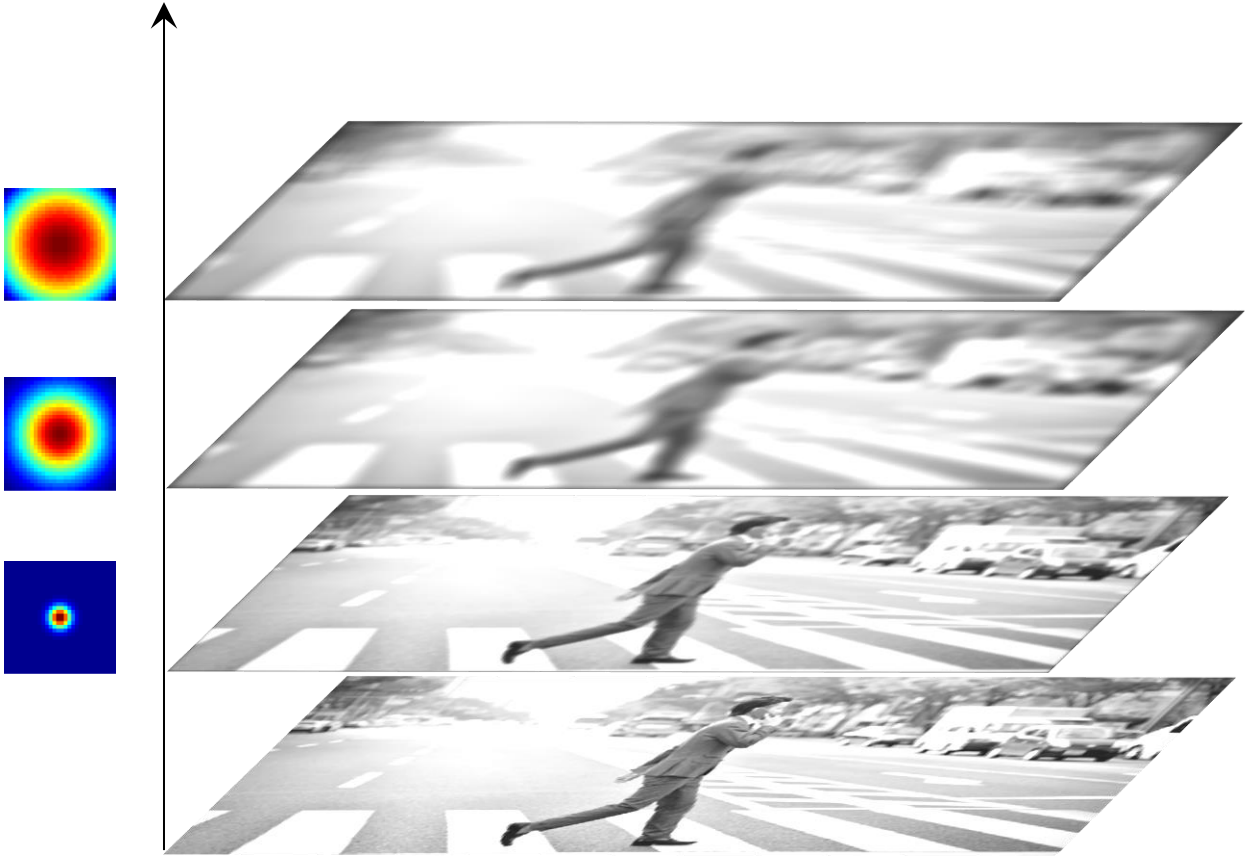




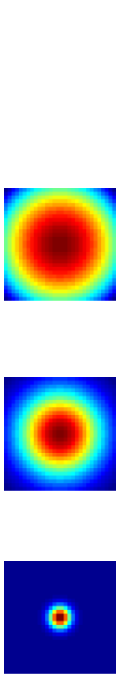
Gaussian Filtering and Then Subsampling

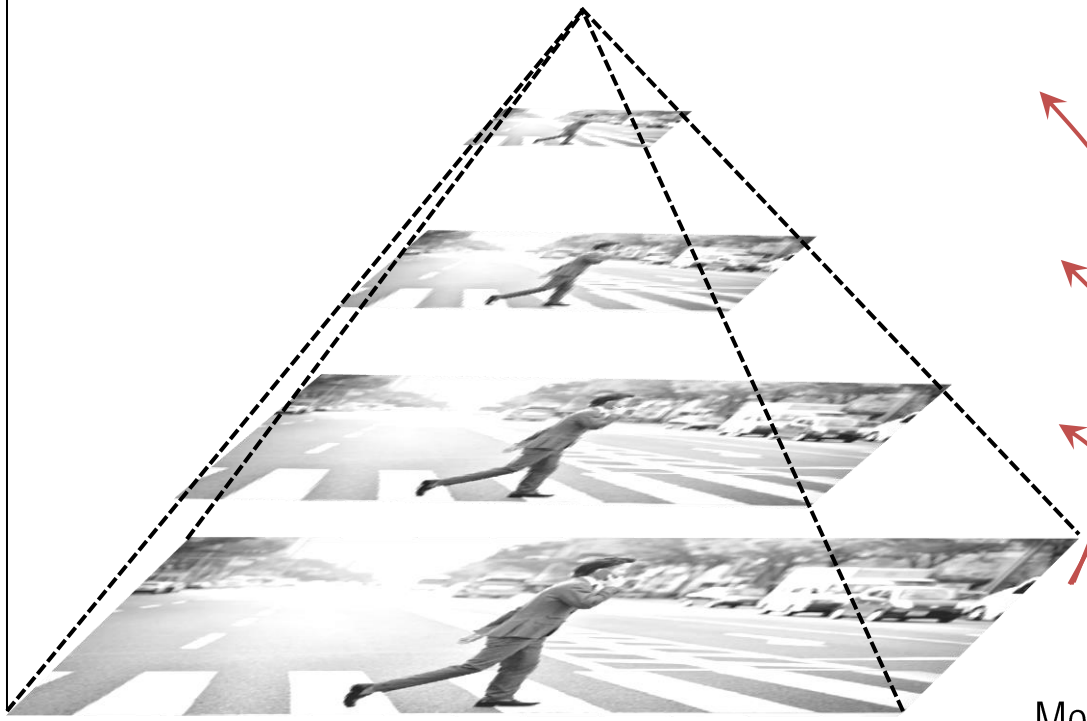
Image Reconstruction: Upsampling and Gaussian Blurring

CF) NAÏVE IMAGE SUBSAMPLING AND UPSAMPLING



MULTI-DIMENSIONAL IMAGE REPRESENTATION





1/16 subsampling

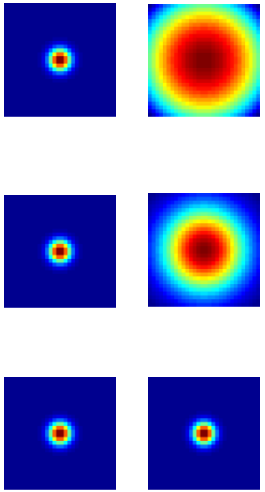
¼ subsampling

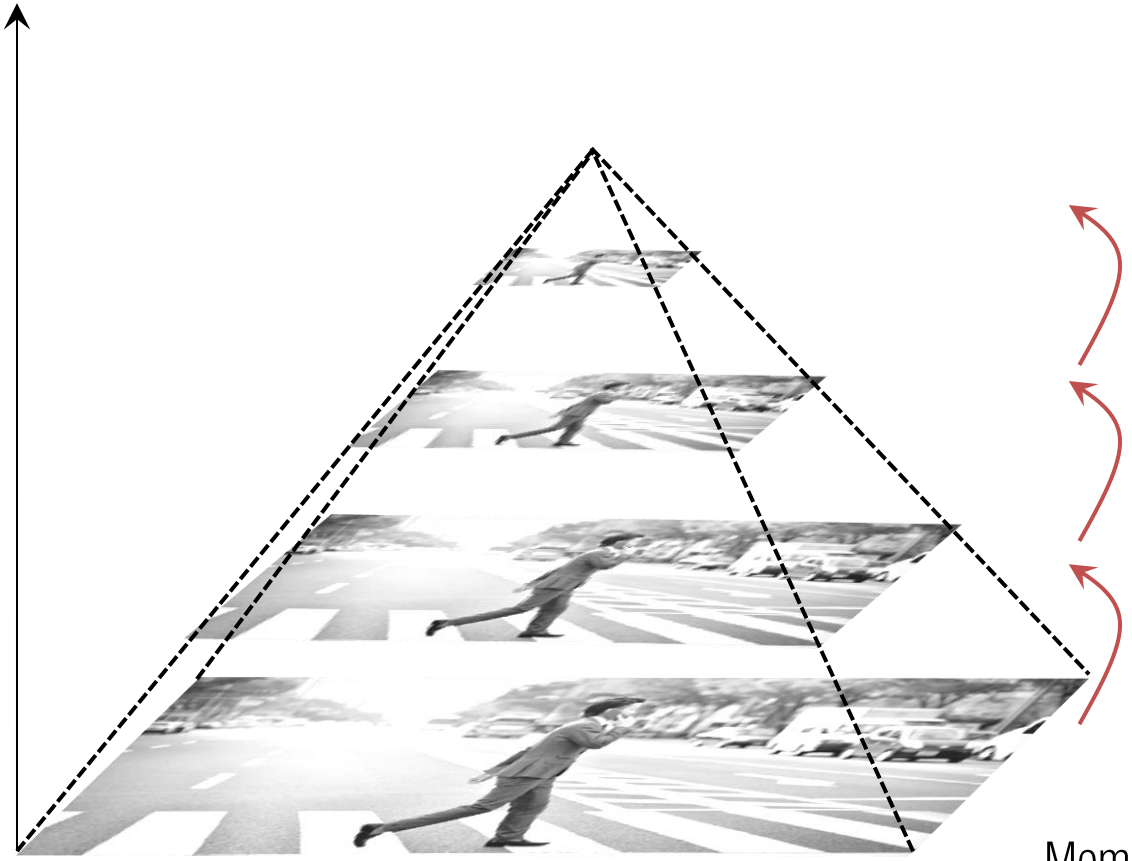
½ subsampling

Memory consumption

$$|I|(1+\frac{1}{4}+\frac{1}{16}+\cdots)=\frac{4}{3}|I|$$

GAUSSIAN IMAGE PYRAMID





½ subsampling

½ subsampling

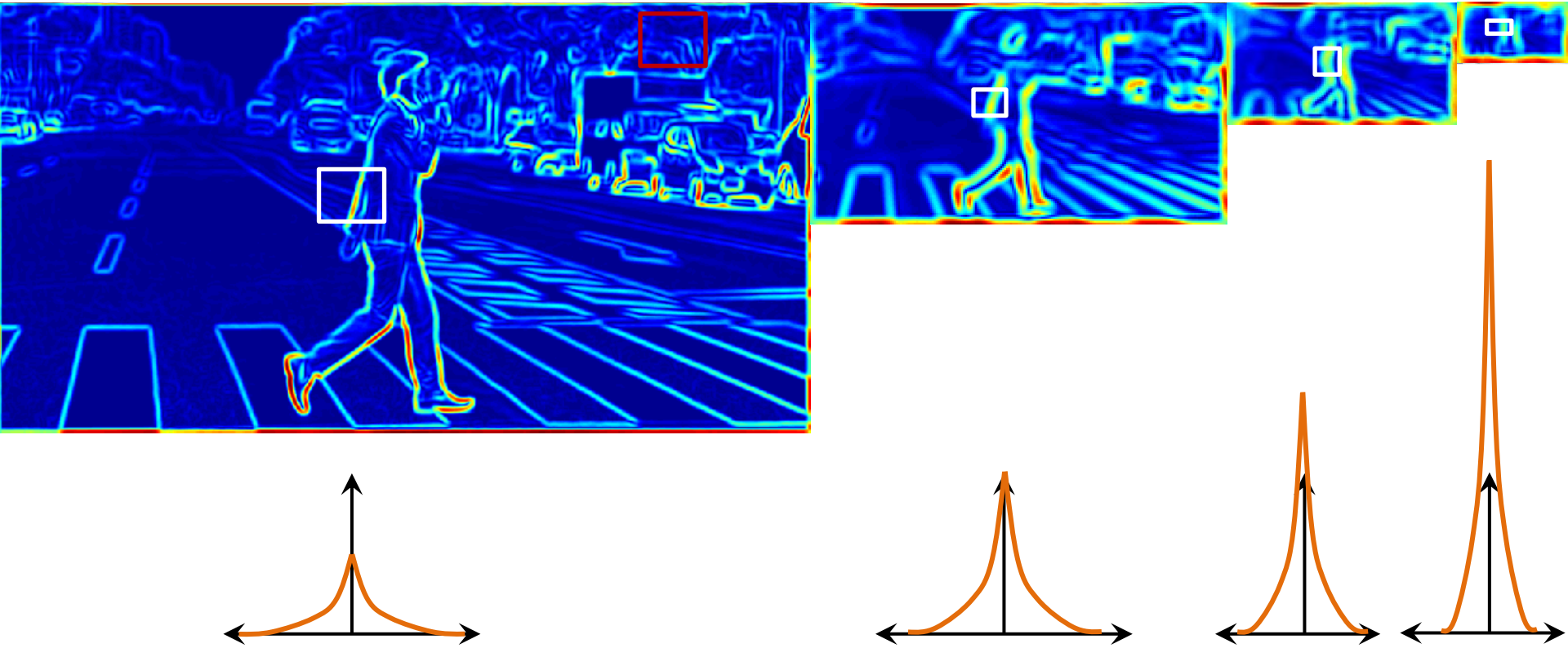
½ subsampling

Memory consumption

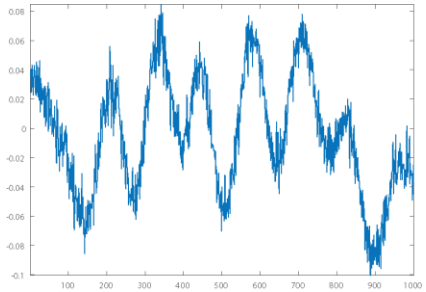
$$|I|(1+\frac{1}{4}+\frac{1}{16}+\cdots)=\frac{4}{3}|I|$$

GAUSSIAN IMAGE PYRAMID

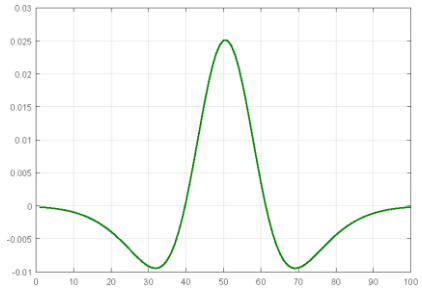
Redundant Representation of Gaussian Pyramid



Difference of Gaussian (DoG) ~ Band-pass Filter

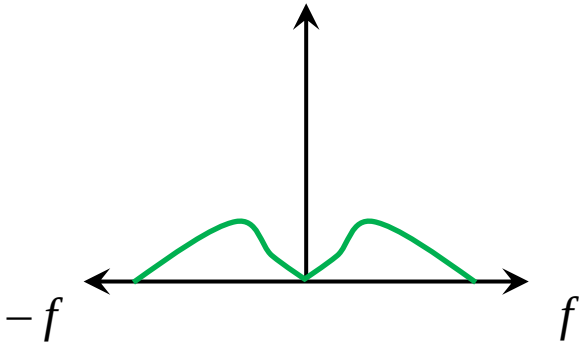


*



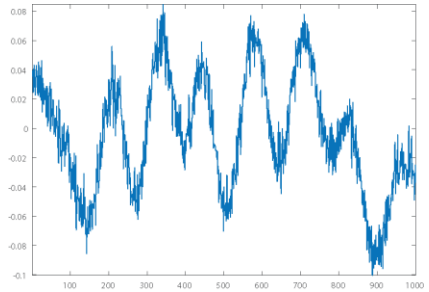
FT

Inverse FT

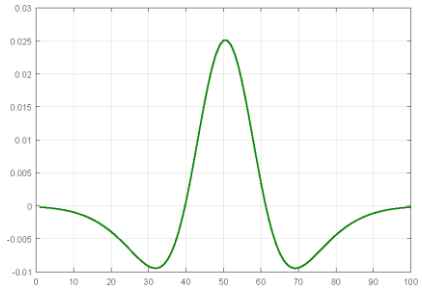


$$x(t) \quad * \quad g(t;\sigma_1) - g(t;\sigma_2) \qquad X(f)\left(G(f;\sigma_1) - G(f;\sigma_2)\right)$$

LAPLACIAN OF GAUSSIAN (LOG) ~ DOG

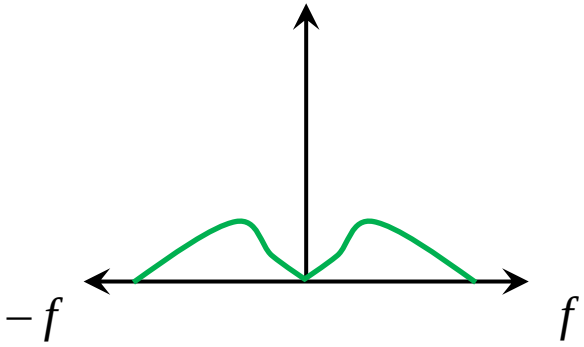


*



FT

Inverse FT



$x(t)$ * $g(t;\sigma_1) - g(t;\sigma_2)$

$\approx \nabla \cdot \nabla g$

Laplacian of Gaussian

$X(f)\left(G(f;\sigma_1) - G(f;\sigma_2)\right)$

Laplacian of Gaussian (LoG) ~ DoG

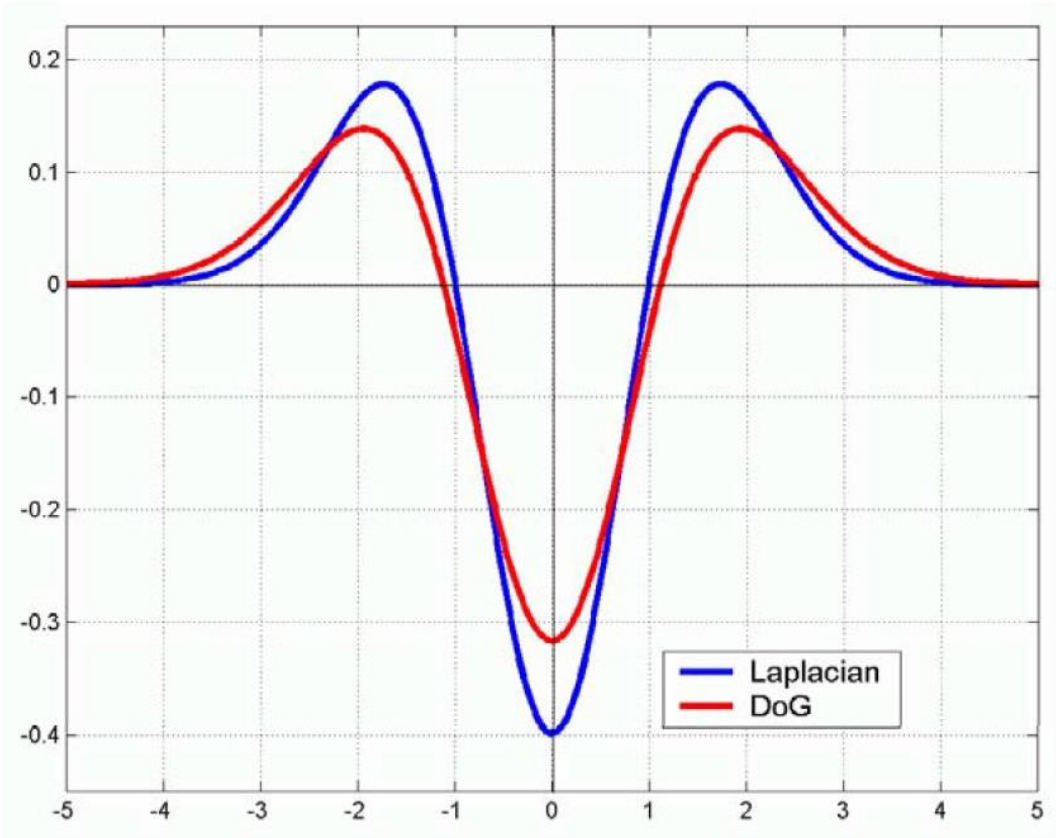
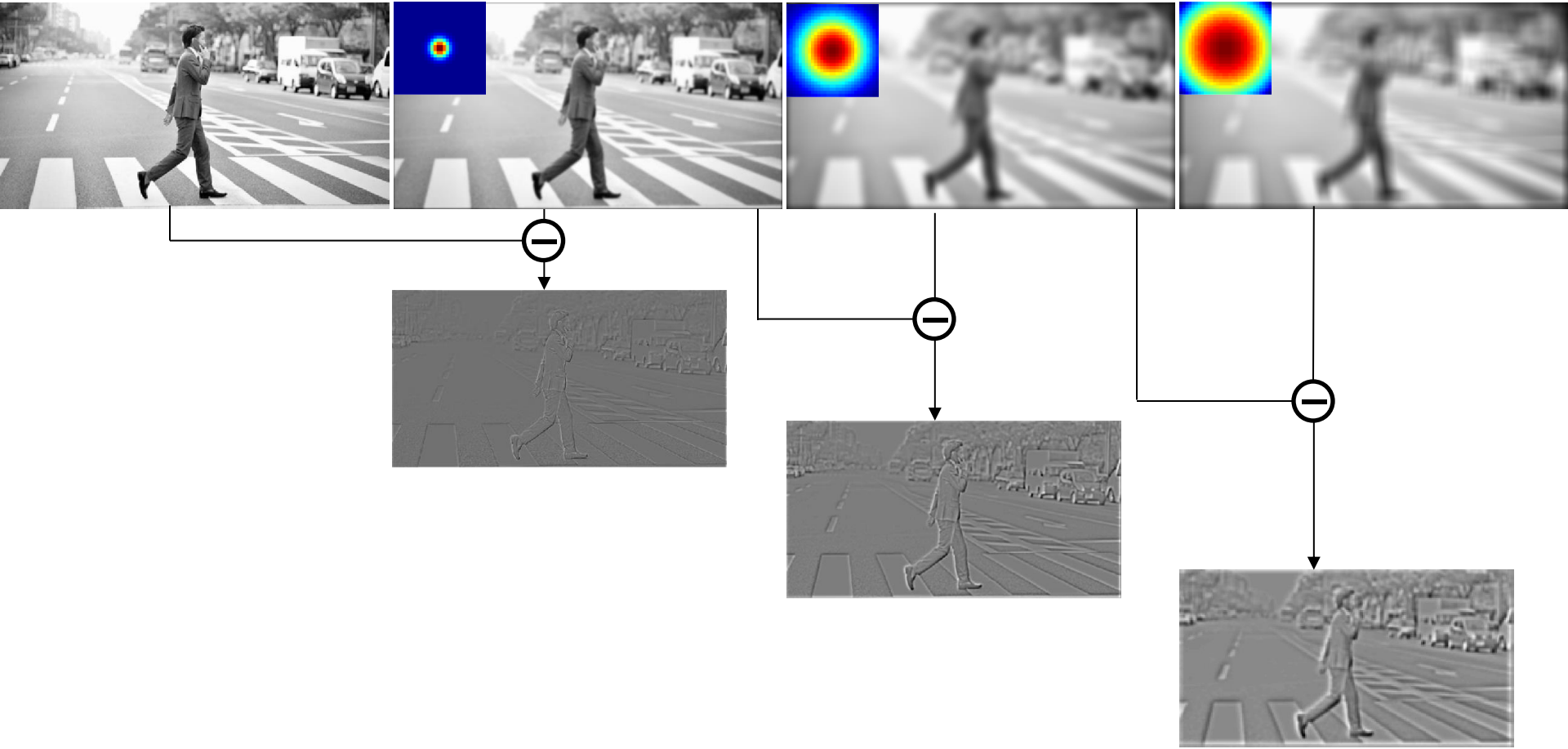
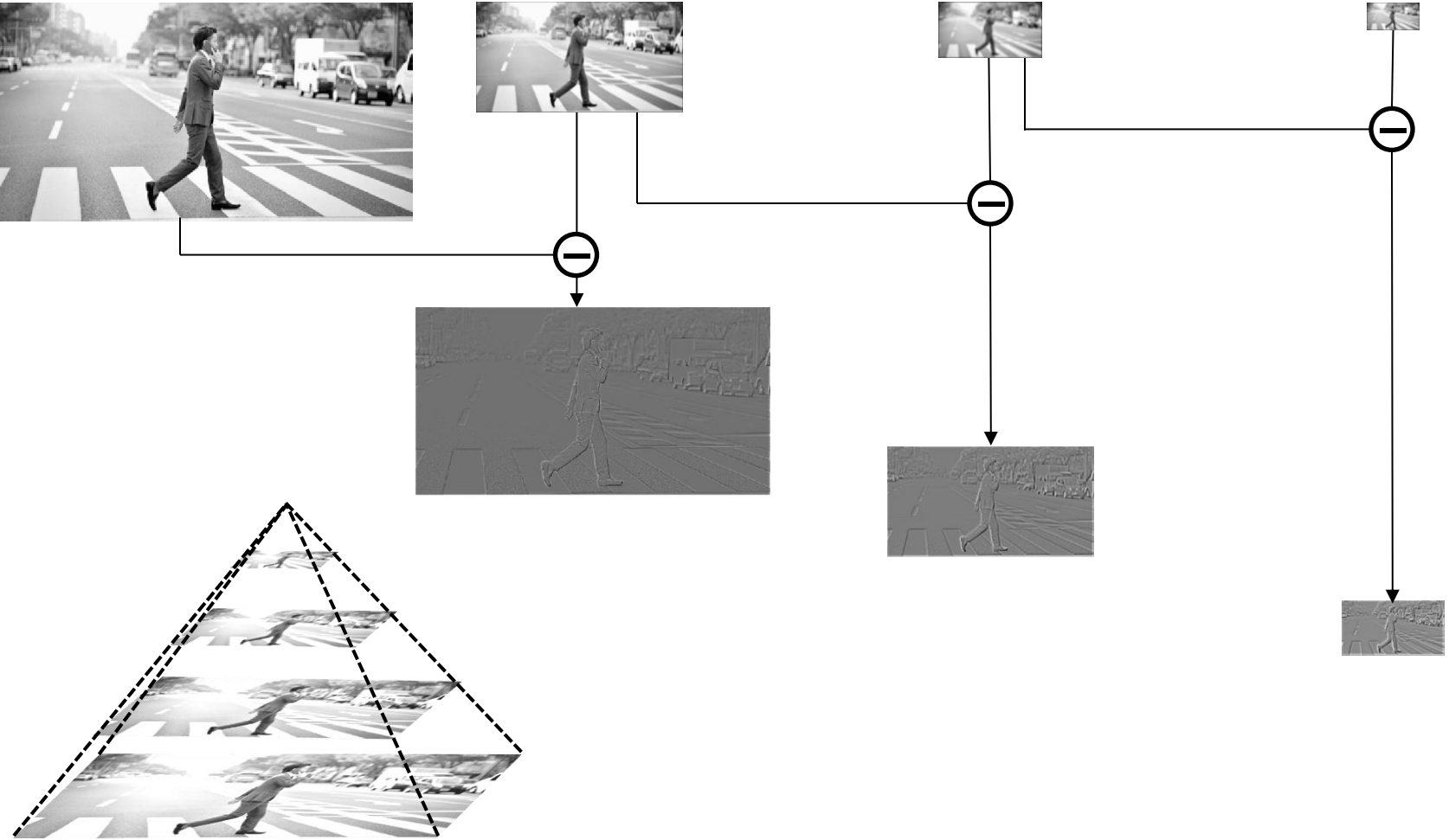
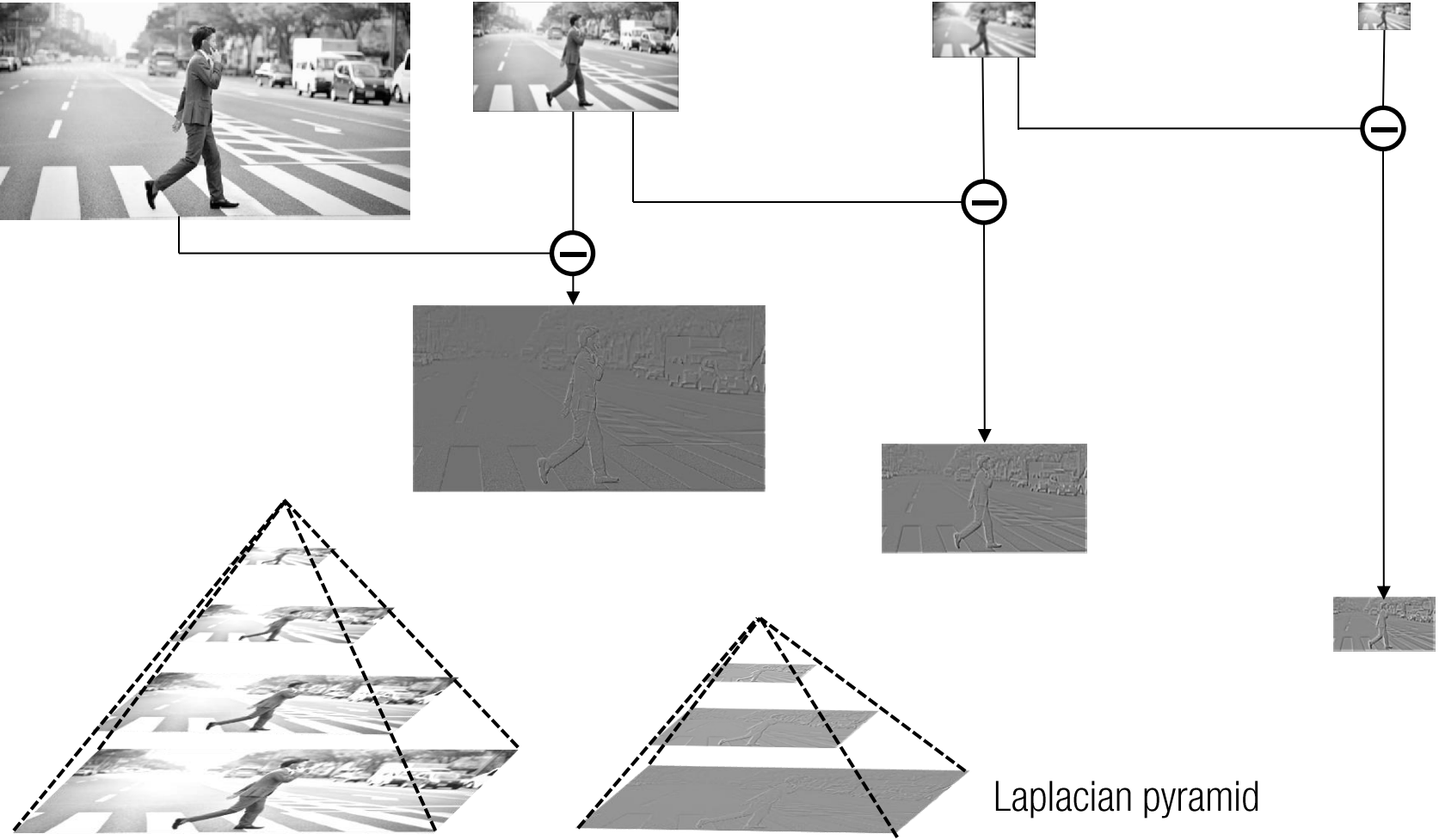


IMAGE LAPLACIAN

IMAGE LAPLACIAN







Laplacian pyramid

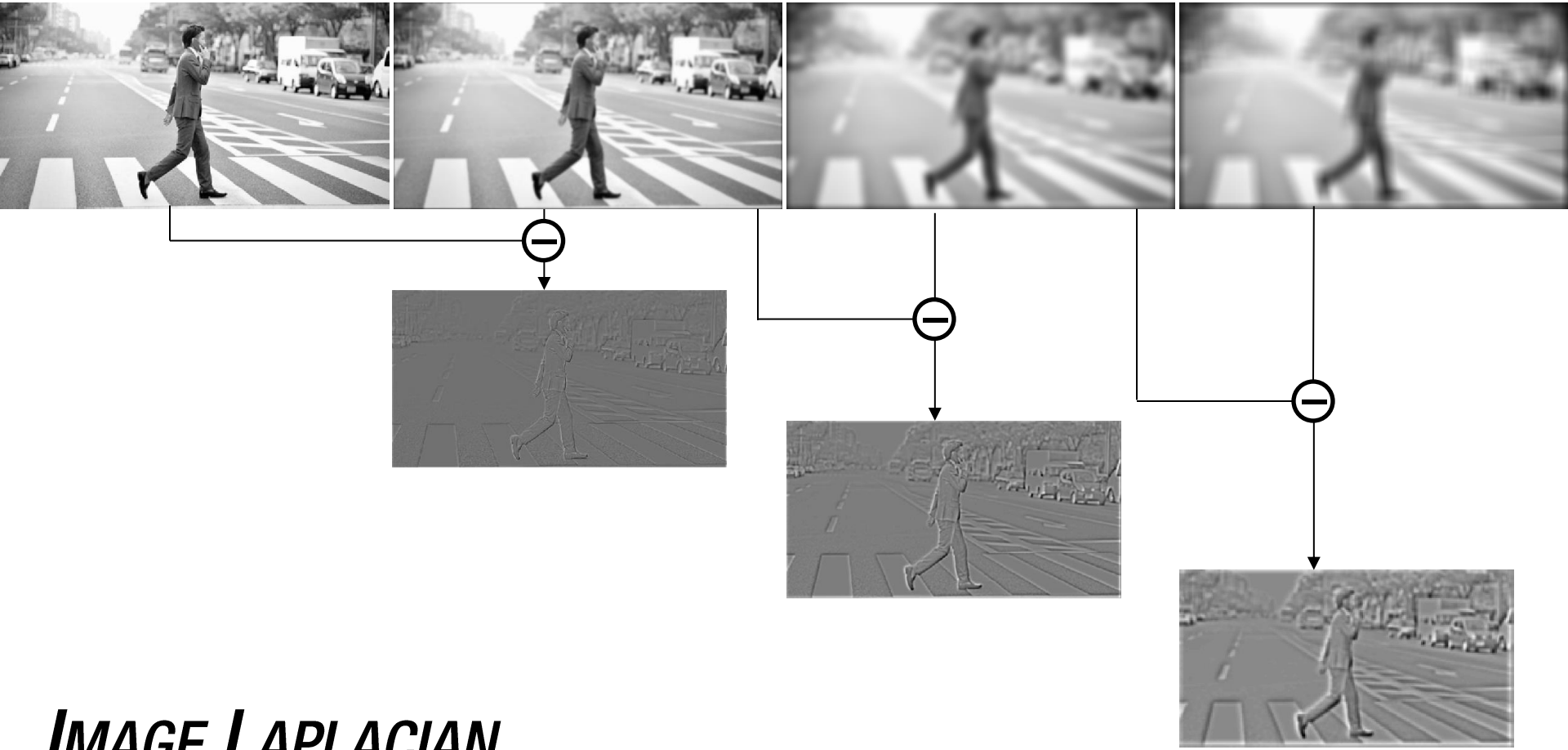


IMAGE LAPLACIAN

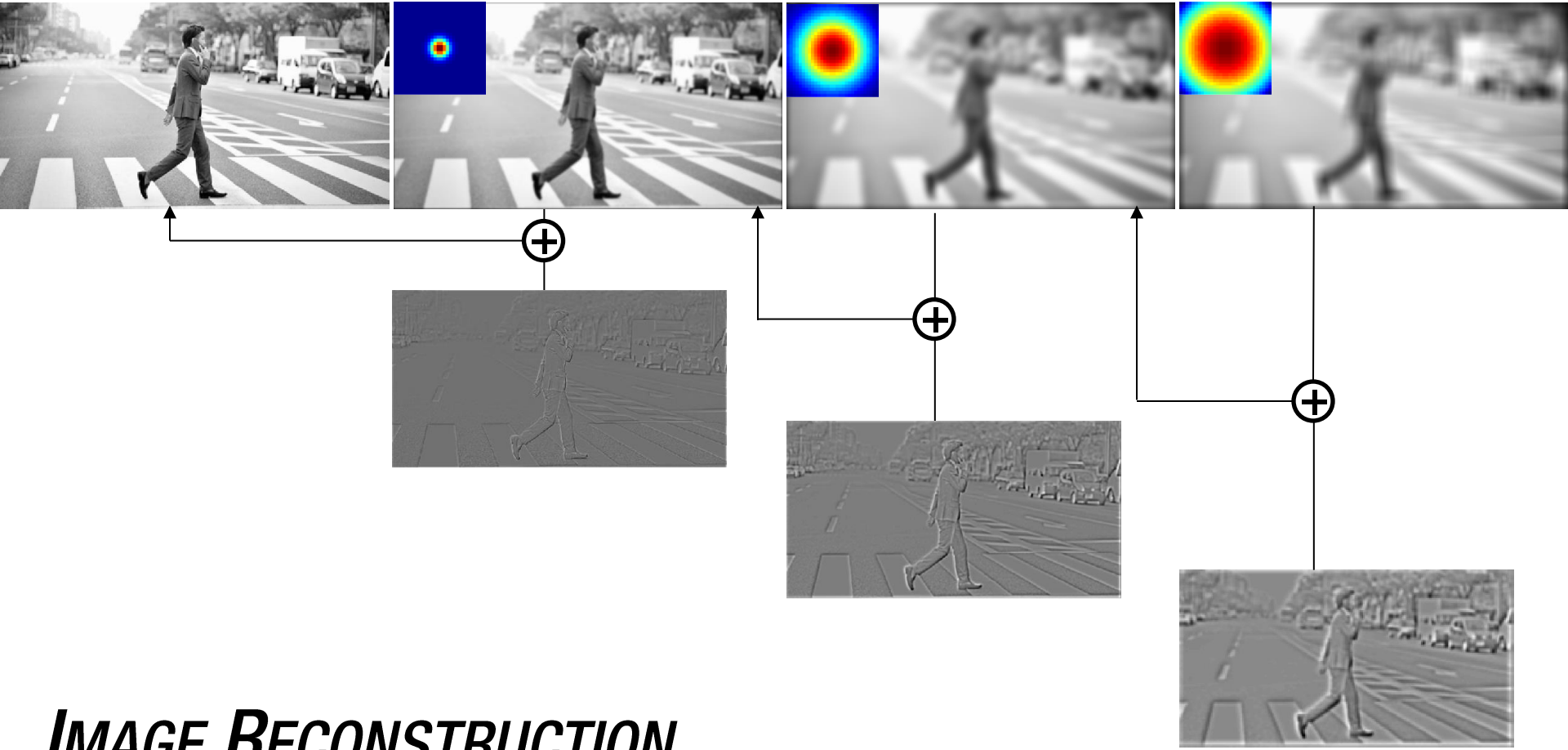
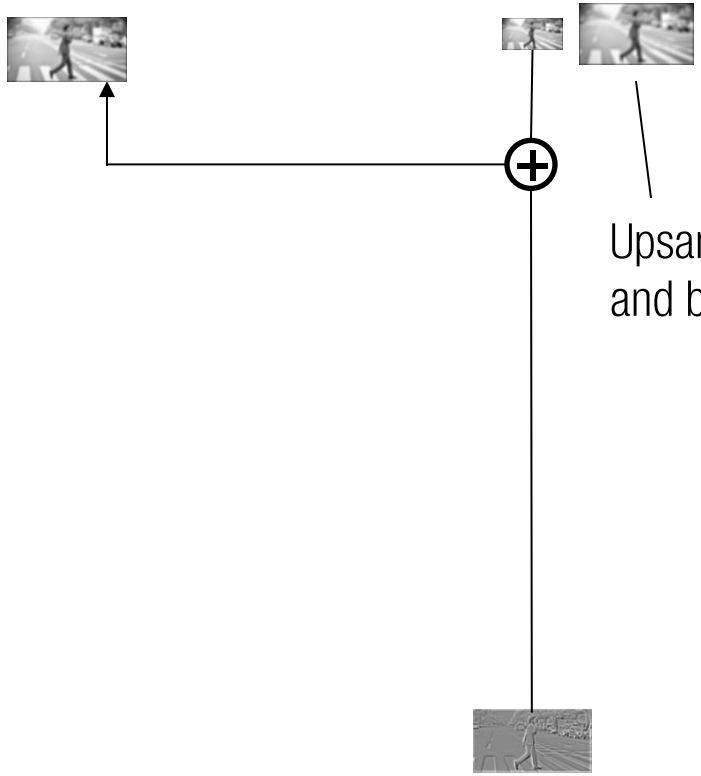
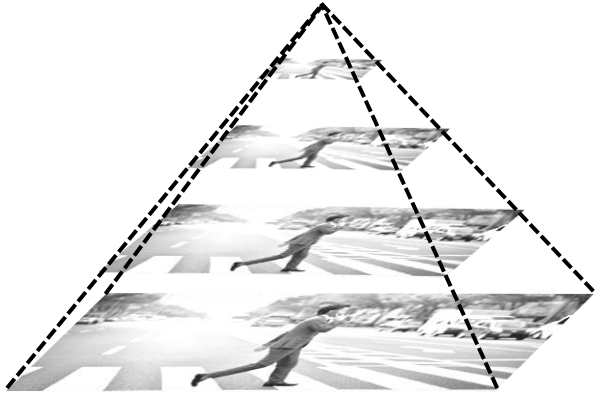
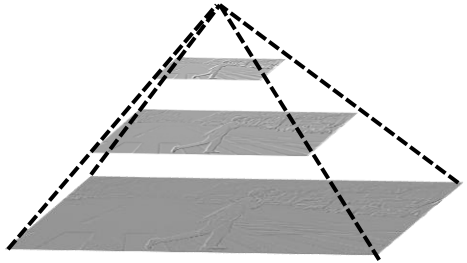


IMAGE RECONSTRUCTION

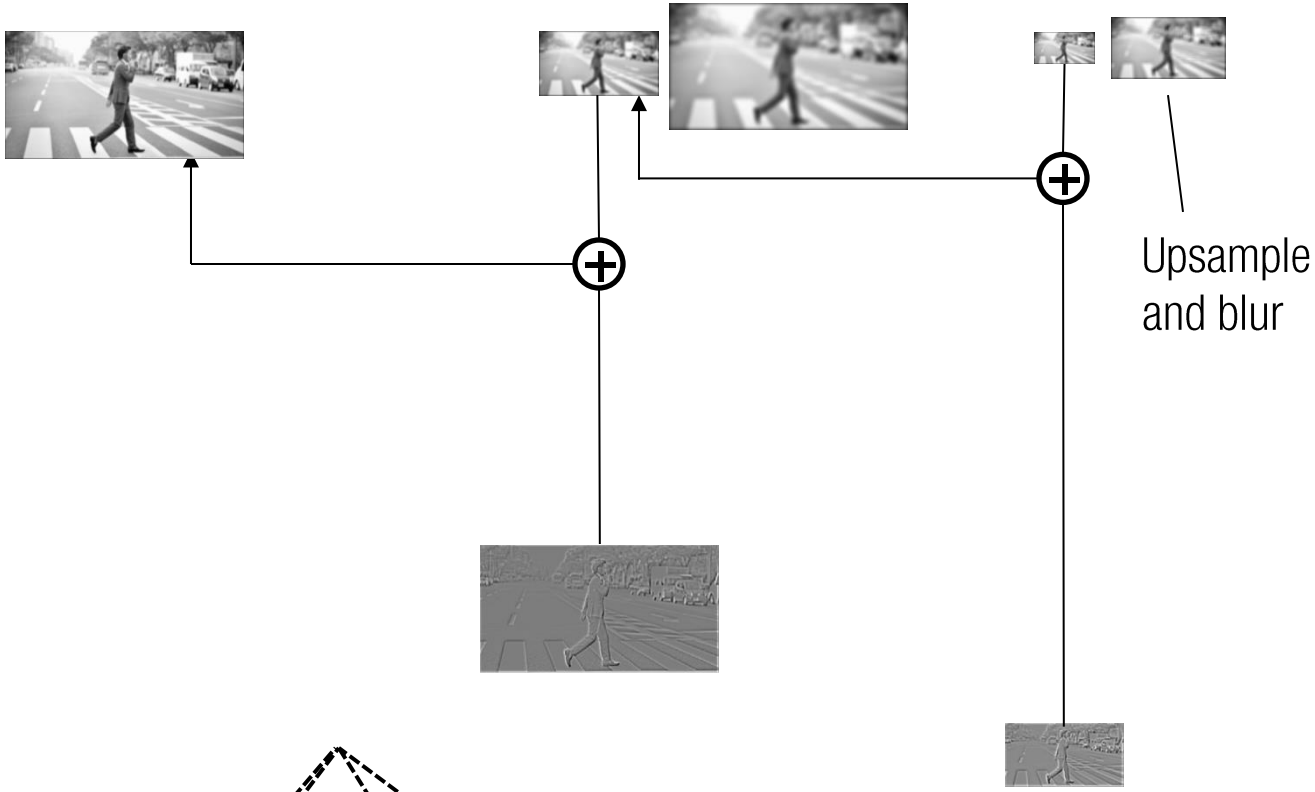


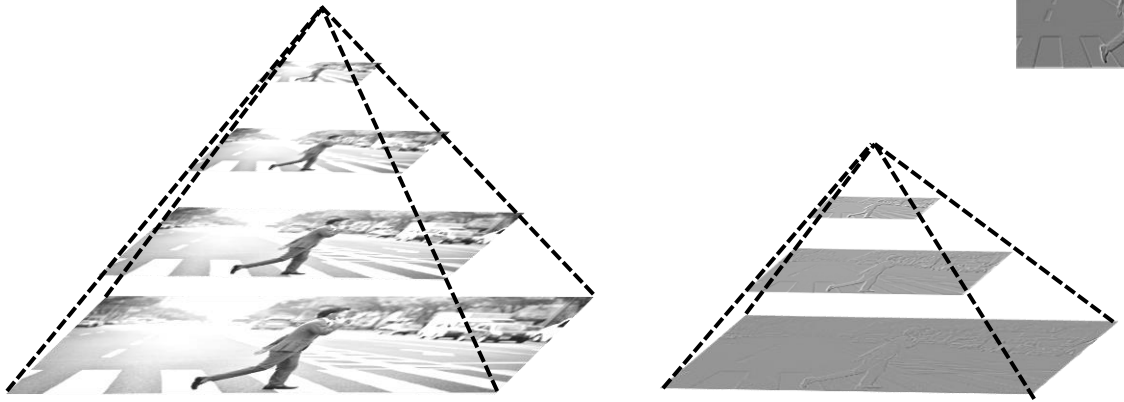
Upsample
and blur





Laplacian pyramid



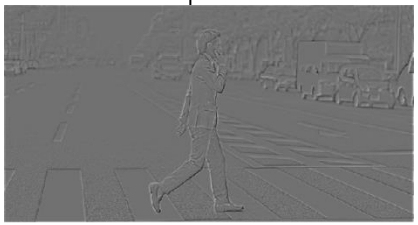


Laplacian pyramid









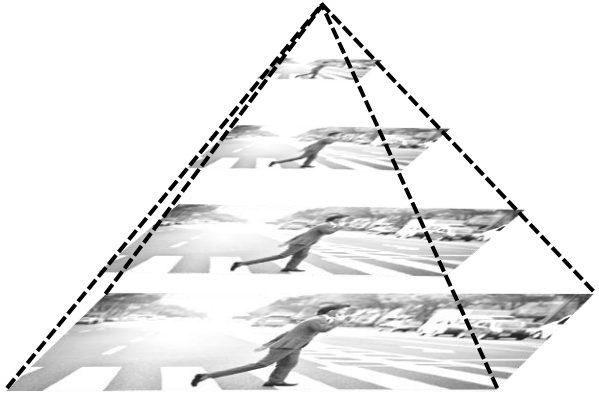
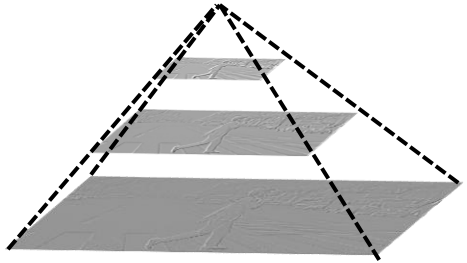
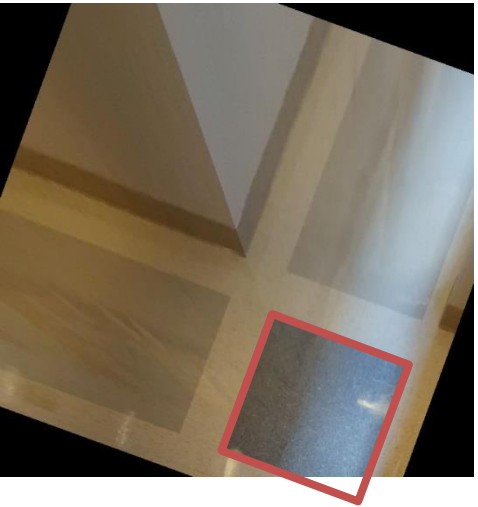



Upsample and blur
Laplacian pyramid

Image Transformation

HIERACHY OF TRANSFORMATIONS



Euclidean (3 dof)

- Length
- Angle
- Area

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix}$$

Similarity (4 dof)

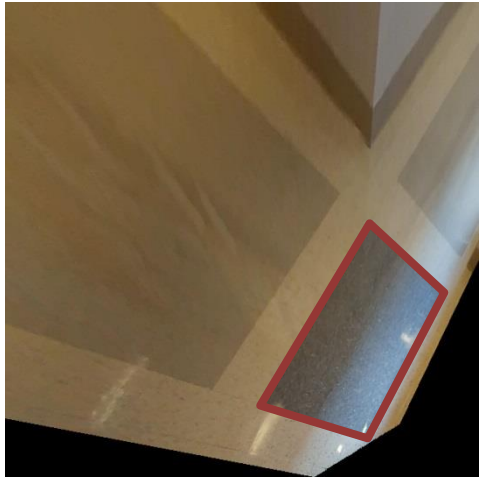
- Length ratio
- Angle

$$\begin{bmatrix} \alpha\cos\theta & -\alpha\sin\theta & t_x \\ \alpha\sin\theta & \alpha\cos\theta & t_y \\ & & 1 \end{bmatrix}$$

Affine (6 dof)

- Parallelism
- Ratio of area
- Ratio of length

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

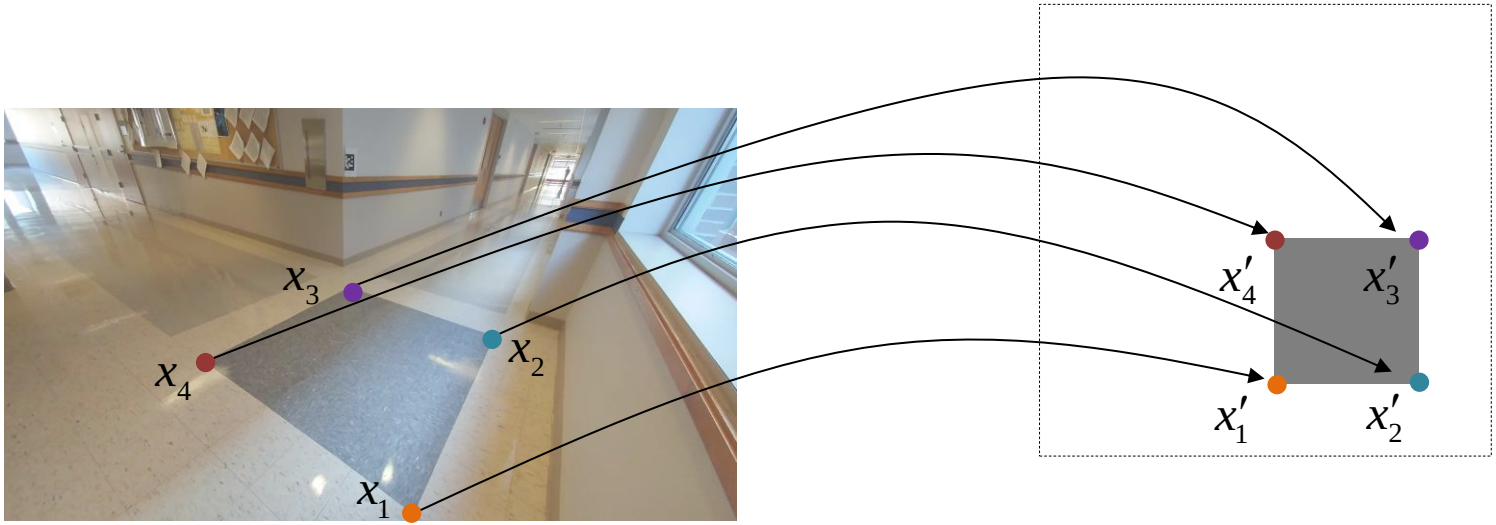


Projective (8 dof)

- Cross ratio
- Concurrency
- Colinearity

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

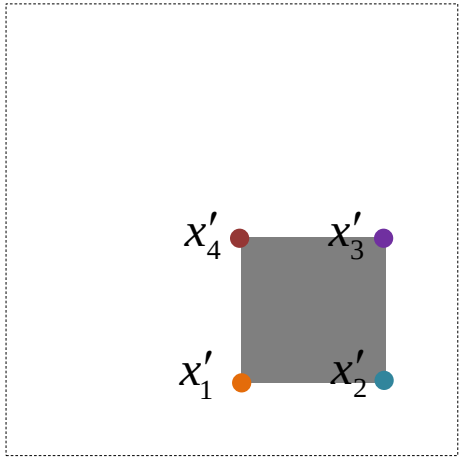
HOMOGRAPHY COMPUTATION



$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

The image can be rectified as if it is seen from top view.

HOMOGRAPHY COMPUTATION



$$u'_1 = \frac{h_{11}u_1 + h_{12}v_1 + h_{13}}{h_{31}u_1 + h_{32}v_1 + 1}$$

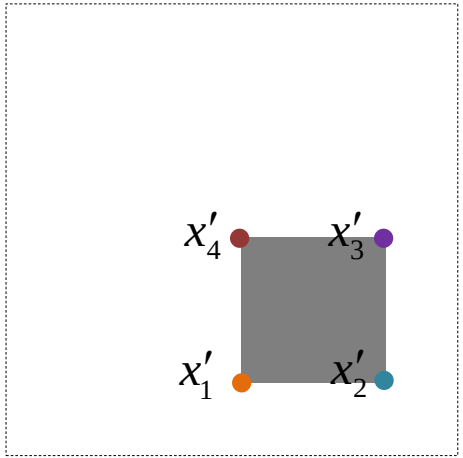
$$v'_1 = \frac{h_{21}u_1 + h_{22}v_1 + h_{23}}{h_{31}u_1 + h_{32}v_1 + 1}$$

$$\rightarrow$$

$$h_{11}u_1 + h_{12}v_1 + h_{13} - h_{31}u_1u'_1 - h_{32}v_1u'_1 - u'_1 = 0$$

$$h_{21}u_1 + h_{22}v_1 + h_{23} - h_{31}u_1v'_1 - h_{32}v_1v'_1 - v'_1 = 0$$

HOMOGRAPHY COMPUTATION

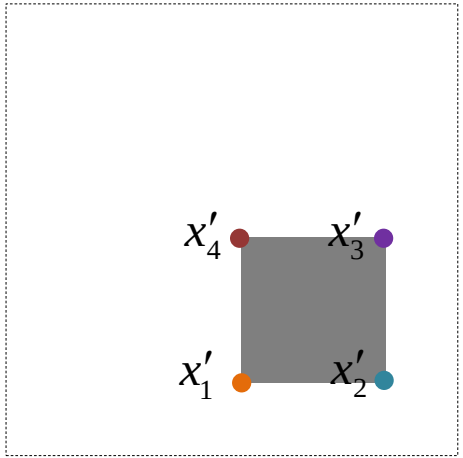


$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u' & -v_1u' \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v' & -v_1v' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} u'_1 \\ v'_1 \end{bmatrix}$$

# of equations: 2
# of unknows: 8

How many correspondences are needed?

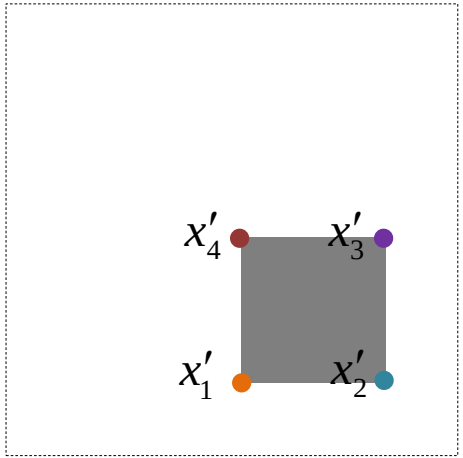
HOMOGRAPHY COMPUTATION



$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u'_1 & -v_1u'_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v'_1 & -v_1v'_1 \\ & & & & \vdots & & & \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -u_4u'_4 & -v_4u'_4 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -u_4v'_4 & -v_4v'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} u'_1 \\ v'_1 \\ \vdots \\ u'_4 \\ v'_4 \end{bmatrix}$$

# of equations: 2
# of unknows: 8

HOMOGRAPHY COMPUTATION

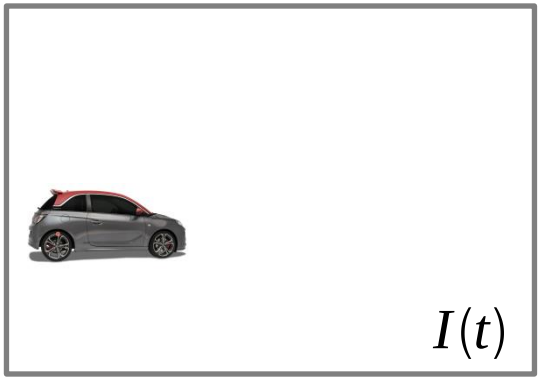


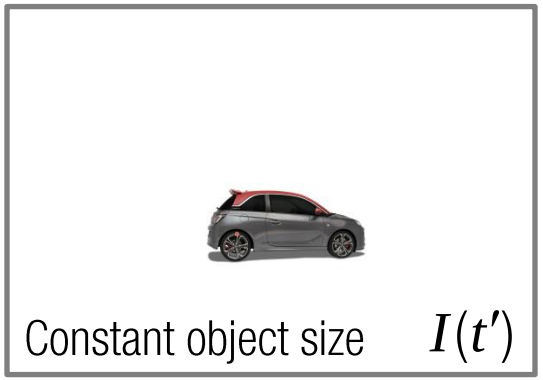
$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u'_1 & -v_1u'_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v'_1 & -v_1v'_1 \\ & & & & \vdots & & & \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -u_4u'_4 & -v_4u'_4 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -u_4v'_4 & -v_4v'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ \vdots \\ h_{32} \end{bmatrix} = \begin{bmatrix} u'_1 \\ v'_1 \\ \vdots \\ u'_4 \\ v'_4 \end{bmatrix}$$

$$Ax = b \longrightarrow x = \left(A^T A\right)^{-1} A^T b$$

Optical Flow

When (u,v) Flow Makes Sense?



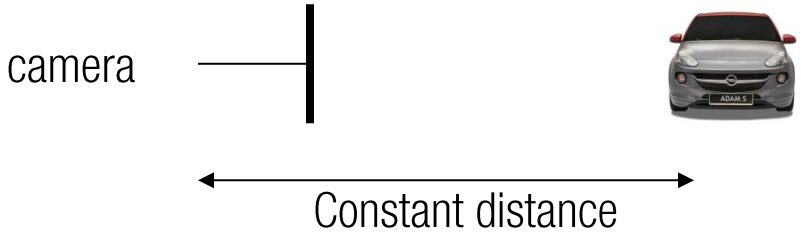


$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

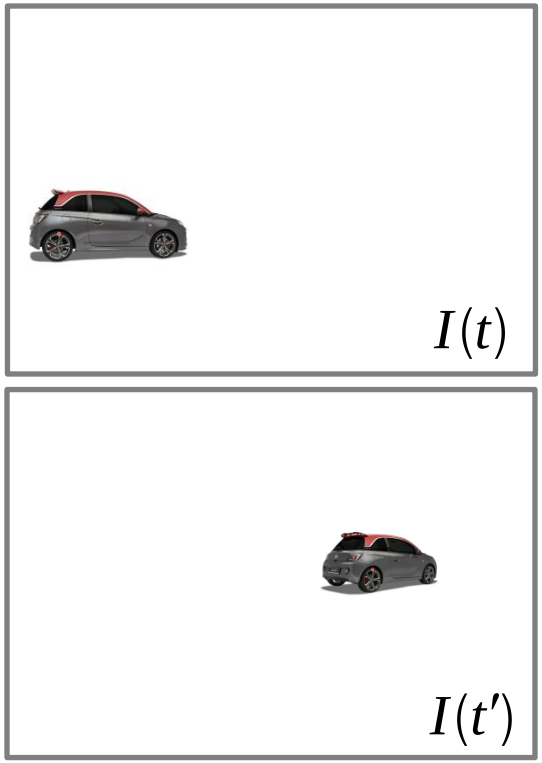
$$x' = x + u$$

$$y' = y + v$$

Side view



General Object Motion



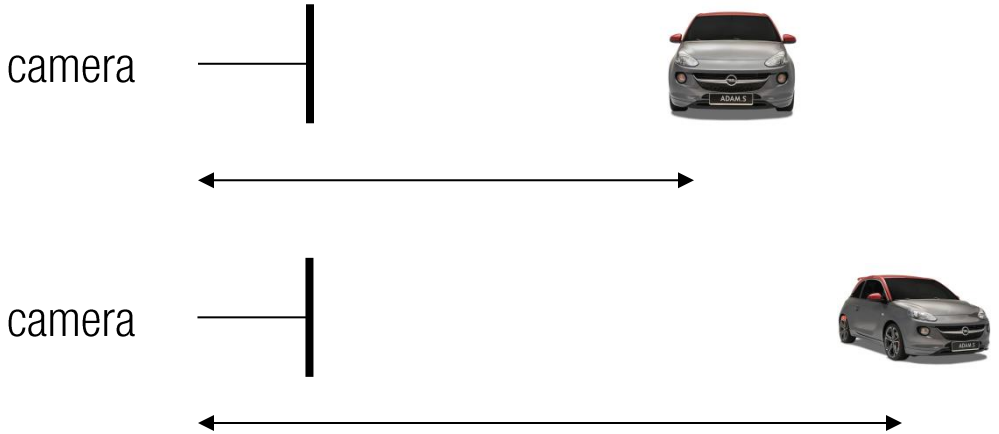
Different depth/orientation

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

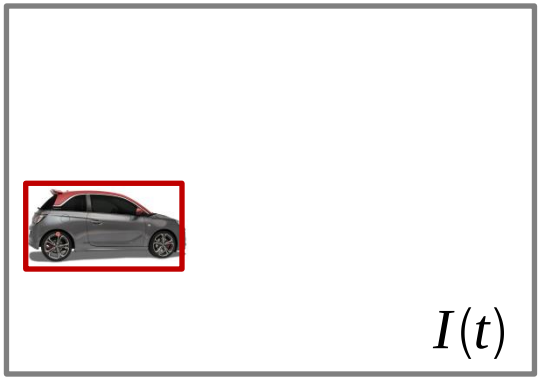
$$x' \neq x + u$$

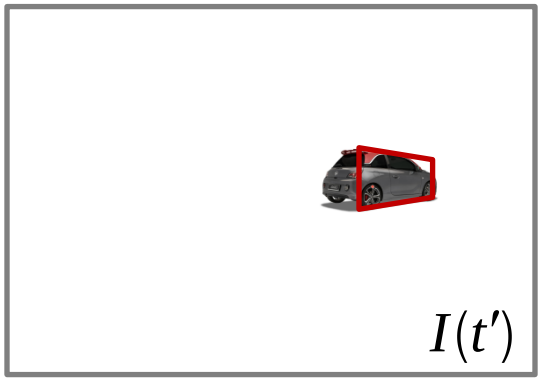
$$y' \neq y + v$$

Side view

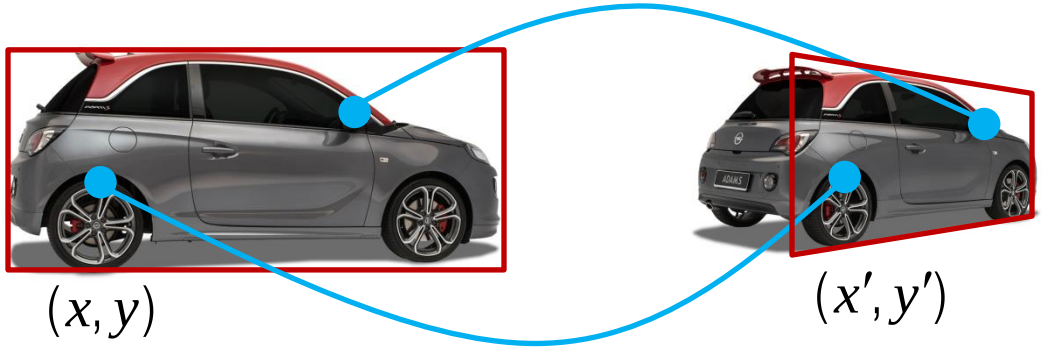


Parametric Transformation





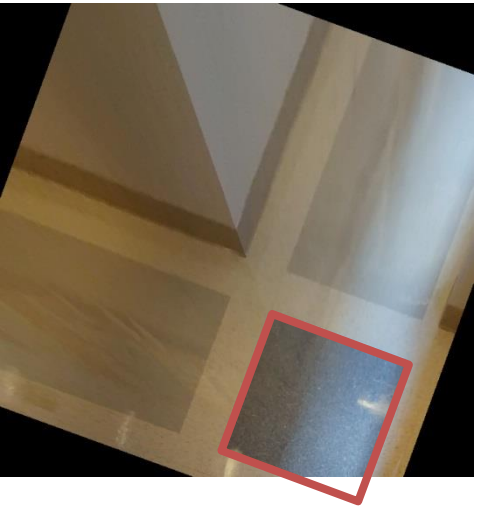
Different depth/orientation



$$\mathbf{x}' = W(\mathbf{x}; p)$$

Unknowns: p

RECALL: PARAMETRIC TRANSFORMATIONS



Euclidean (3 dof)
- Length
- Angle
- Area

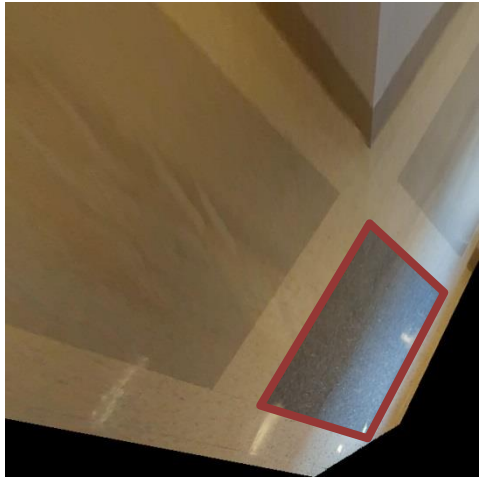
Ex) Aerial images

Similarity (4 dof)
- Length ratio
- Angle

Change of depth

Affine (6 dof)
- Parallelism
- Ratio of area
- Ratio of length

Far objects

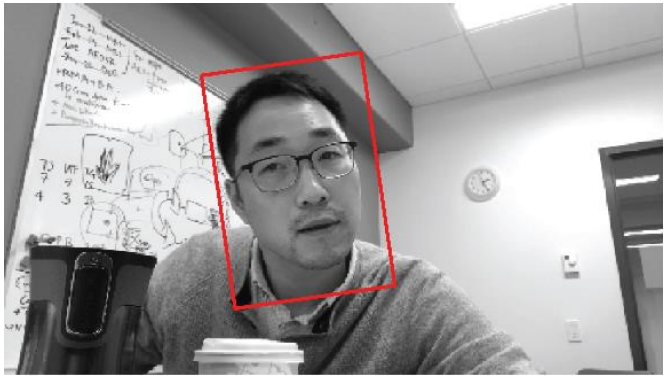


Projective (8 dof)
- Cross ratio
- Concurrency
- Colinearity

Planar objects

IMAGE ALIGNMENT

Template
$T(x)$



Target image
$I(x)$

Brightness constancy

$$I(W(x;p)) \approx T(x)$$

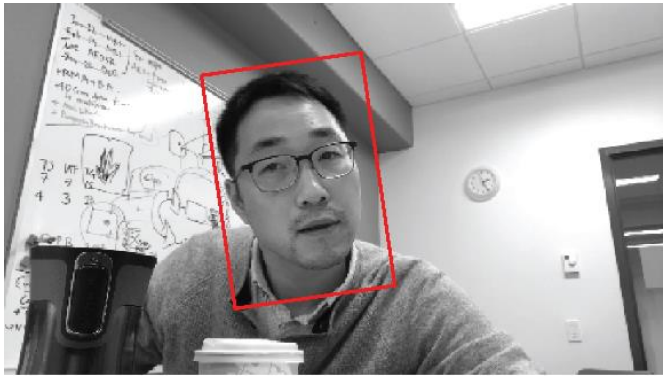
ex) affine transform

$$W(x;p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$= \begin{bmatrix} (p_1 + 1)u + p_2 v + p_3 \\ p_4 u + (p_5 + 1)v + p_6 \end{bmatrix}$$

IMAGE ALIGNMENT

Template
$T(x)$



Target image
$I(x)$

Brightness constancy

$$I(W(x;p)) \approx T(x)$$

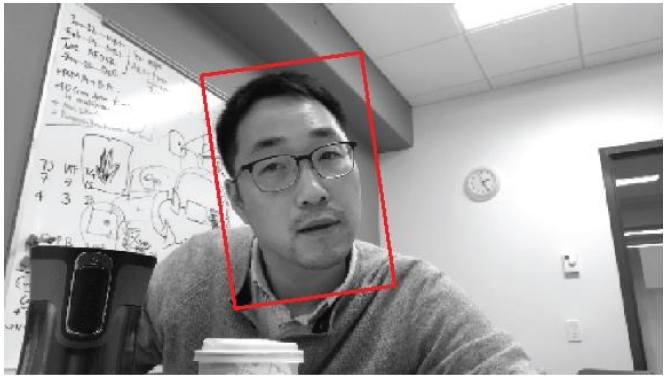
Objective: to find the optimal warping parameter p that minimizes warping error.

$$p* = \underset{p}{\text{minimize}} \sum_{x} \Big(I(W(x;p)) - \overline{T(x)} \Big)^2$$

Template

IMAGE ALIGNMENT

Template
$T(x)$



Target image
$I(x)$

Brightness constancy

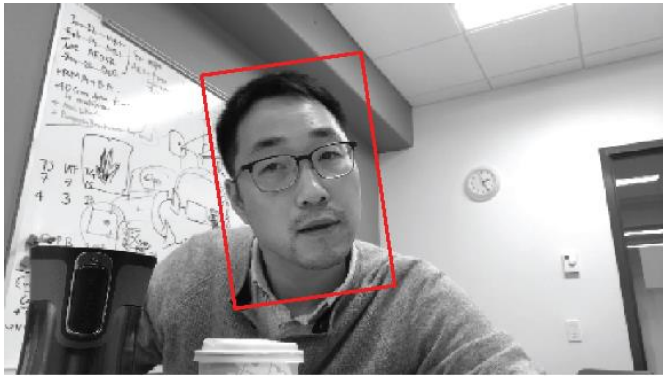
$$I(W(x;p)) \approx T(x)$$

Objective: to find the optimal warping parameter p that minimizes warping error.

$$p* = \underset{p}{\text{minimize}} \sum_{x} \Big(\underbrace{I(W(x;p))}_{\text{Warped image}} - \underbrace{T(x)}_{\text{Template}} \Big)^2$$

IMAGE ALIGNMENT

Template
$T(x)$



Target image
$I(x)$

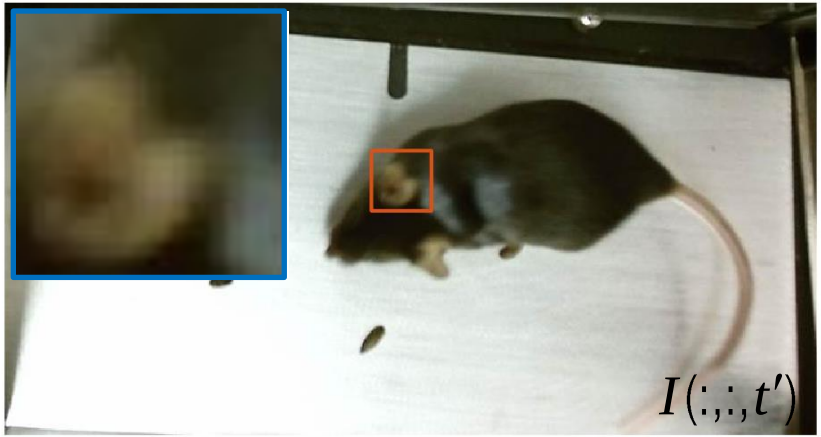
Brightness constancy

$$I(W(x;p)) \approx T(x)$$

Objective: to find the optimal warping parameter p that minimizes warping error.

$$p* = \underset{p}{\text{minimize}} \sum_{x} \underbrace{\Big(I(W(x;p)) - T(x) \Big)^2}_{\text{Error image}}$$

RECALL: LOCAL PATCH TRACKING



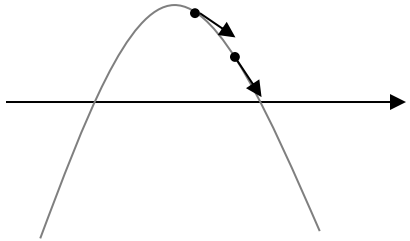
NCC: 0.923520; SSD: 3.854872

$$x = \left(A^T A\right)^{-1} A^T b \qquad x = \begin{bmatrix} -2.76 \\ 1.27 \end{bmatrix}$$

First order approximation

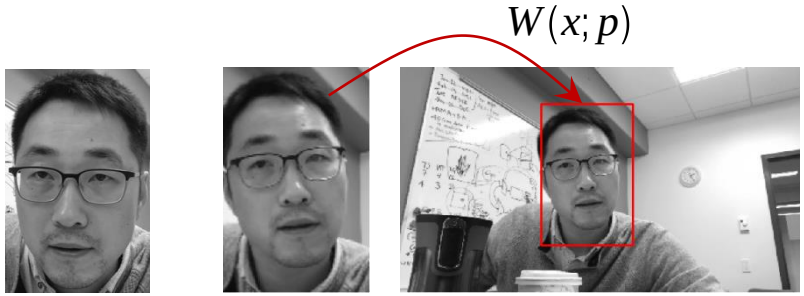
$$I(x + u\delta t, y + v\delta t, t + \delta t)$$

$$\approx I(x, y, t) + \frac{\partial I}{\partial x} u\delta t + \frac{\partial I}{\partial y} v\delta t + \frac{\partial I}{\partial t} \delta t$$



Guass-Newton's method

IMAGE ALIGNMENT OBJECTIVE



$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1. Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p} \Delta p$$

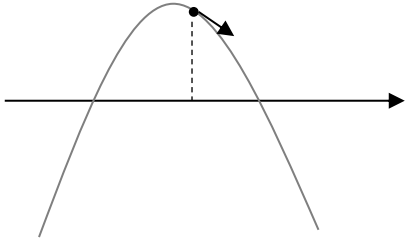
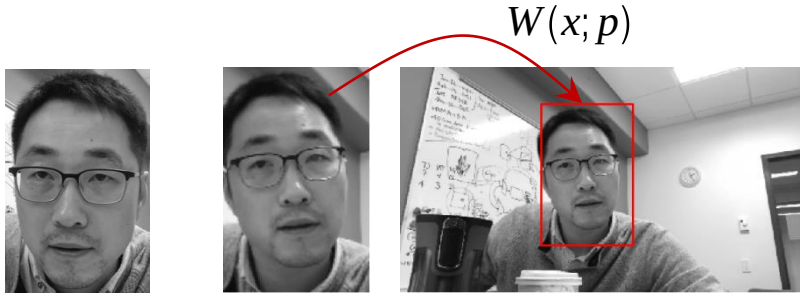


IMAGE ALIGNMENT OBJECTIVE

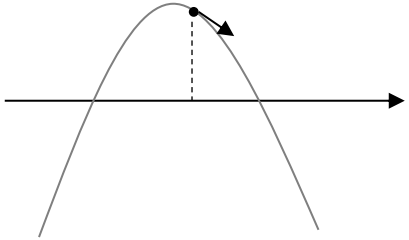


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p} \Delta p$$



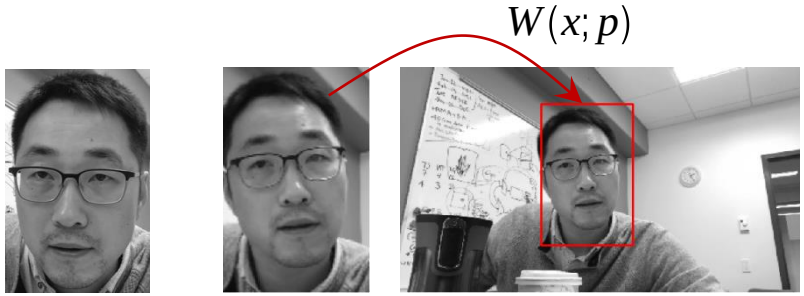
Ex) optical flow (translation)

$$W(x;p) = x + \Delta x$$

$$\rightarrow \frac{\partial I}{\partial p} \Delta p = \frac{\partial I}{\partial x} \Delta x = \nabla I \Delta x$$

$$= \quad \Delta u + \quad \Delta v$$

IMAGE ALIGNMENT OBJECTIVE

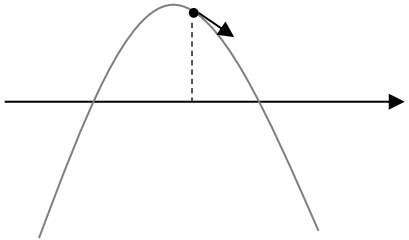


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p} \Delta p$$



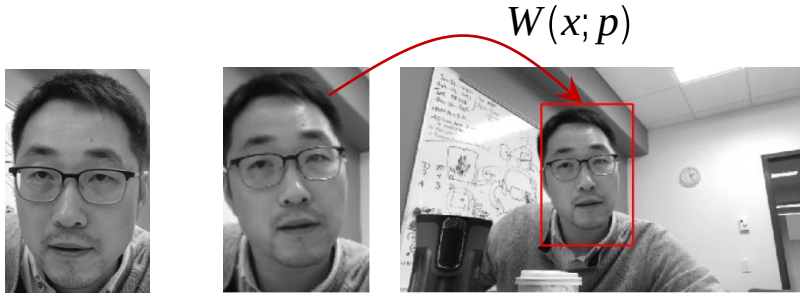
Ex) affine transform

$$W(x;p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$\rightarrow \frac{\partial I}{\partial p} \Delta p$$

What does the gradient image w.r.t. the affine parameters mean?

IMAGE ALIGNMENT OBJECTIVE

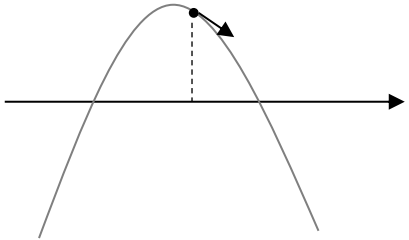


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p} \Delta p$$



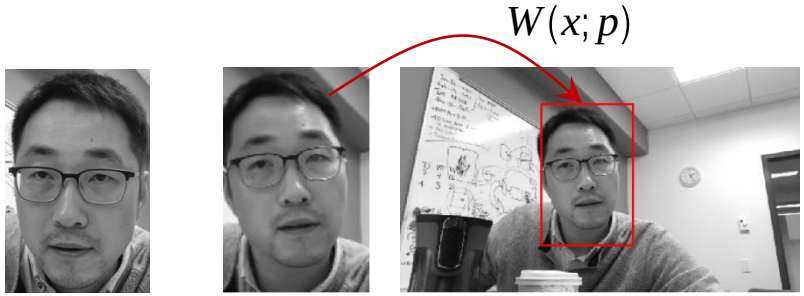
Ex) affine transform

$$W(x;p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$\rightarrow \quad \frac{\partial I}{\partial p} \Delta p = \frac{\partial I}{\partial x} \frac{\partial W}{\partial p} \Delta p$$

Chain rule

IMAGE ALIGNMENT OBJECTIVE

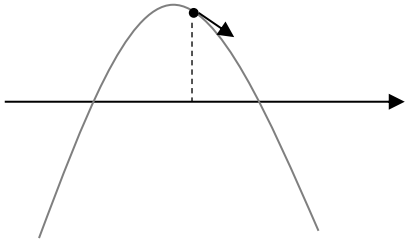


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p} \Delta p$$



Ex) affine transform

$$W(x;p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$\rightarrow \quad \frac{\partial I}{\partial p} \Delta p = \frac{\partial I}{\partial x} \frac{\partial W}{\partial p} \Delta p = \nabla I \frac{\partial W}{\partial p} \Delta p$$

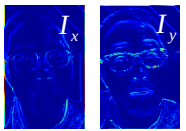
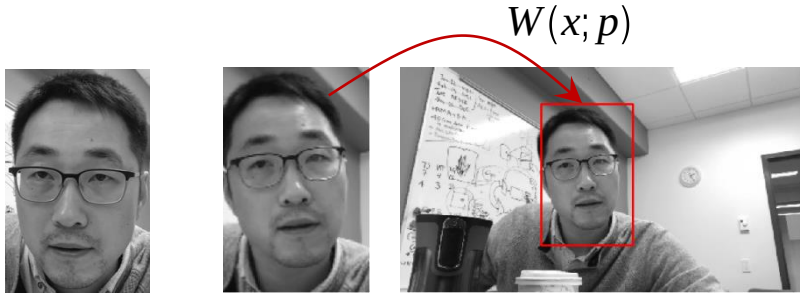


IMAGE ALIGNMENT OBJECTIVE

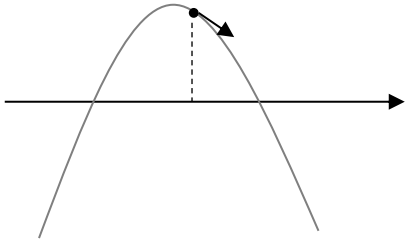


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \frac{\partial I}{\partial p}\Delta p$$



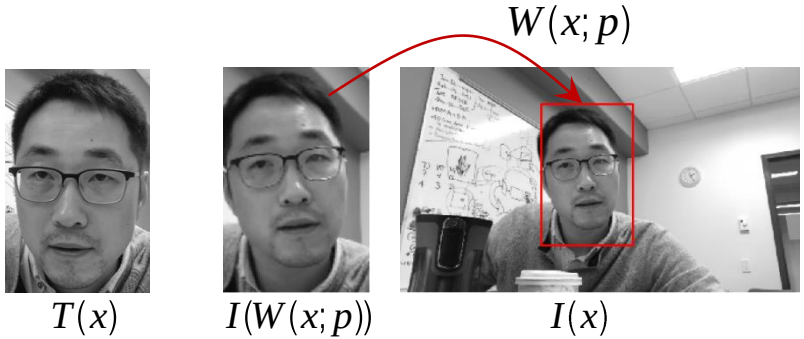
Ex) affine transform

$$W(x;p) = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} p_1 u + p_2 v + p_3 \\ p_4 u + p_5 v + p_6 \end{bmatrix}$$

$$\rightarrow \quad \frac{\partial I}{\partial p}\Delta p = \frac{\partial I}{\partial x}\frac{\partial W}{\partial p}\Delta p = \nabla I \frac{\partial W}{\partial p}\Delta p$$

Jacobian: $\frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial u}{\partial p_1} & \cdots & \frac{\partial u}{\partial p_6} \\ \frac{\partial v}{\partial p_1} & \cdots & \frac{\partial v}{\partial p_6} \end{bmatrix}$

IMAGE ALIGNMENT OBJECTIVE

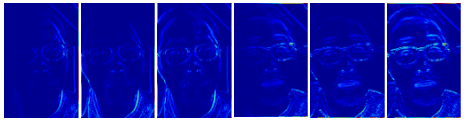


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

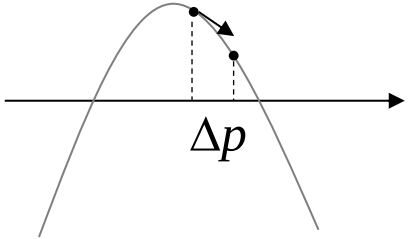
Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$

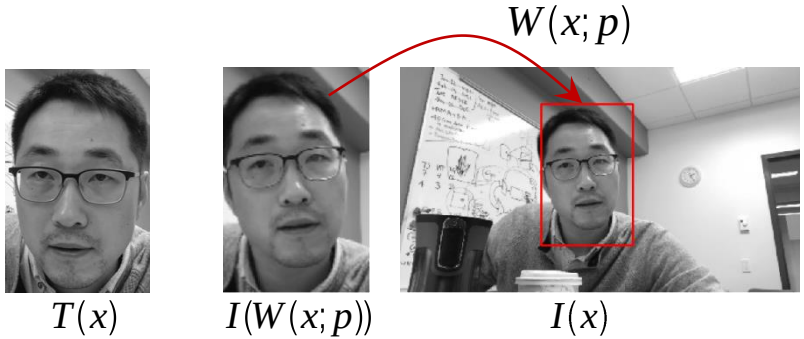


2. Find Δp that minimizes the obj. function at p



$$I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) = 0 \longrightarrow \nabla I \frac{\partial W}{\partial p} \Delta p = T(x) - I(W(x;p))$$

IMAGE ALIGNMENT OBJECTIVE

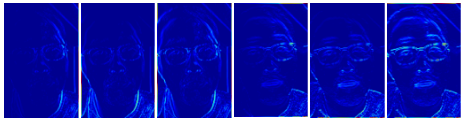


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$



2. Find Δp that minimizes the obj. function at p

$$I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) = 0 \longrightarrow \nabla I \frac{\partial W}{\partial p} \Delta p = T(x) - I(W(x;p))$$

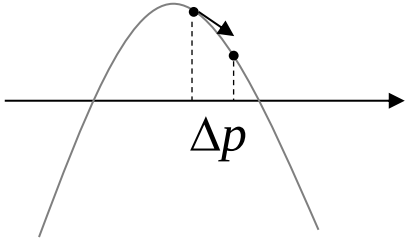
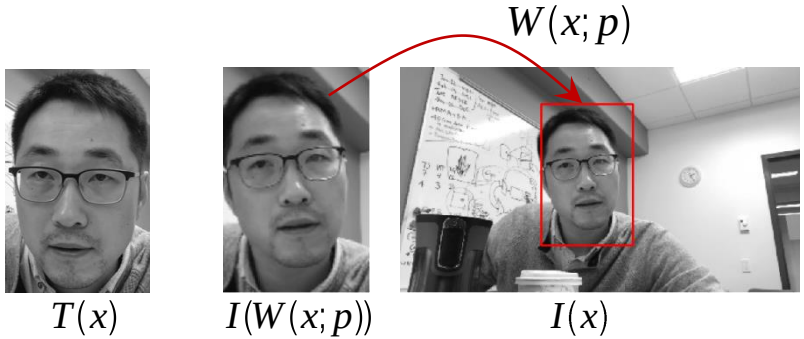


IMAGE ALIGNMENT OBJECTIVE

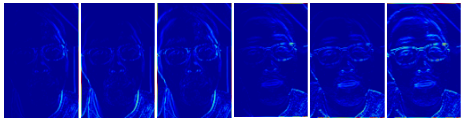


$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

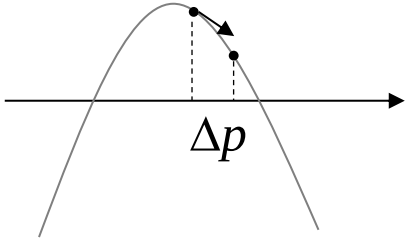
1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$



2. Find Δp that minimizes the obj. function at p

$$I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) = 0 \longrightarrow \nabla I \frac{\partial W}{\partial p} \Delta p = T(x) - I(W(x;p))$$



$$\begin{bmatrix} \nabla I \frac{\partial W}{\partial p}\Big|_{x_1} \\ \vdots \\ \nabla I \frac{\partial W}{\partial p}\Big|_{x_n} \end{bmatrix} \Delta p = - \begin{bmatrix} I(W(x_1;p)) - T(x_1) \\ \\ I(W(x_n;p)) - T(x_n) \end{bmatrix}$$

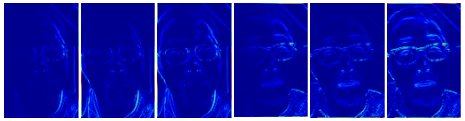
IMAGE ALIGNMENT OBJECTIVE

$$p* = \underset{p}{\text{minimize}} \sum_{x} \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

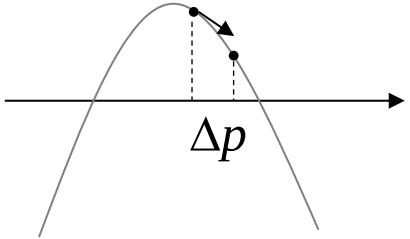
1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$



2. Find Δp that minimizes the obj. function at p

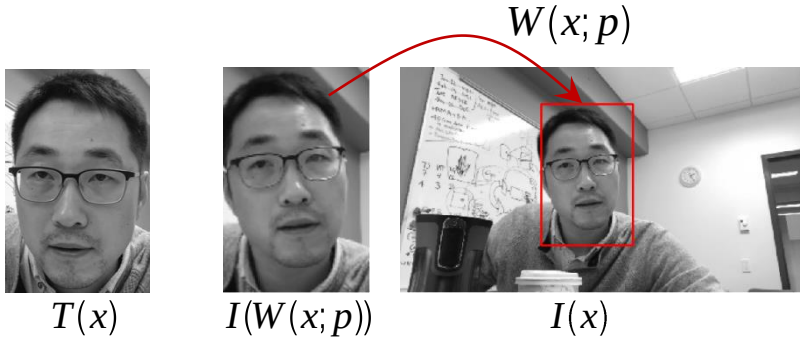
$$I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) = 0 \longrightarrow \nabla I \frac{\partial W}{\partial p} \Delta p = T(x) - I(W(x;p))$$



$$\begin{bmatrix} \nabla I \frac{\partial W}{\partial p}\Big|_{x_1} \\ \vdots \\ \nabla I \frac{\partial W}{\partial p}\Big|_{x_n} \end{bmatrix} \Delta p = \begin{bmatrix} I(W(x_1;p)) - T(x_1) \\ \vdots \\ I(W(x_n;p)) - T(x_n) \end{bmatrix}$$

$$A \quad x = b$$

IMAGE ALIGNMENT OBJECTIVE

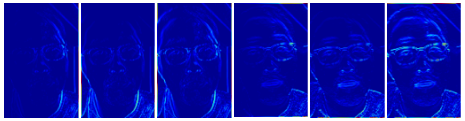


$$p* = \underset{p}{\text{minimize}} \sum_x \left(I(W(x;p) - T(x) \right)^2$$

Guass-Newton's method

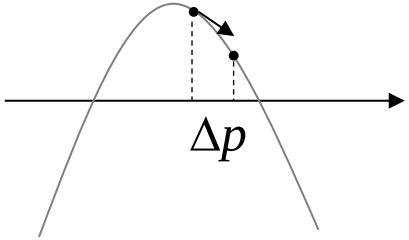
1.Linearize the obj. function at p

$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$



2. Find Δp that minimizes the obj. function at p

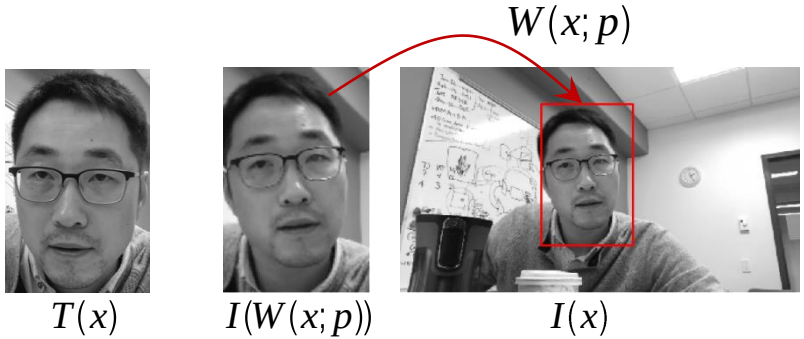
$$I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) = 0 \longrightarrow \nabla I \frac{\partial W}{\partial p} \Delta p = T(x) - I(W(x;p))$$



$$\Delta p = H^{-1} \sum_x \left(\nabla I \frac{\partial W}{\partial p} \right)^T (T(x) - I(W(x;p)))$$

where $H = \sum_x \left(\nabla I \frac{\partial W}{\partial p} \right)^T \left(\nabla I \frac{\partial W}{\partial p} \right)$

IMAGE ALIGNMENT OBJECTIVE



$$p* = \underset{p}{\text{minimize}} \sum_x \left(I(W(x;p) - T(x) \right)^2$$

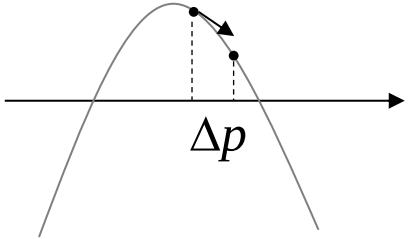
Guass-Newton's method

1.Linearize the obj. function at p

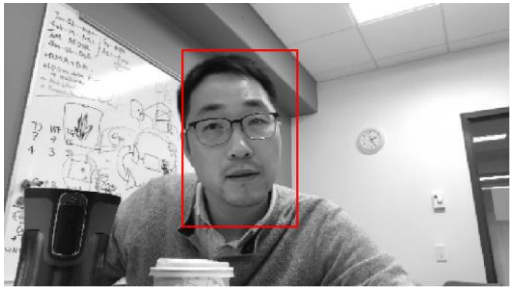
$$I(W(x;p+\Delta p)) \approx I(W(x;p)) + \nabla I \frac{\partial W}{\partial p} \Delta p$$

2. Find Δp that minimizes the obj. function at p

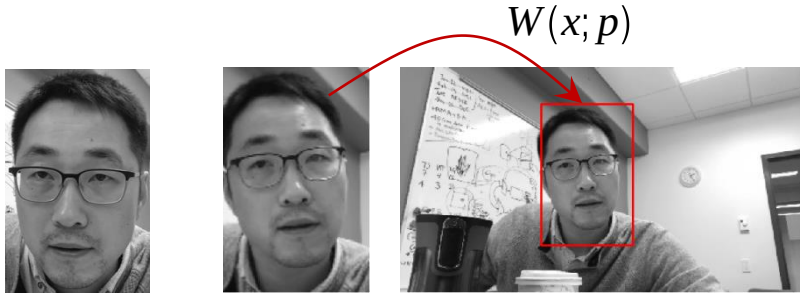
$$\Delta p = H^{-1} \sum_x \left(\nabla I \frac{\partial W}{\partial p} \right)^T (T(x) - I(W(x;p)))$$



3. Update $p \leftarrow p + \Delta p$



Optical Flow Derivation

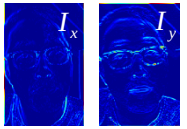


$$\Delta x^* = \underset{\Delta x}{\text{minimize}} \sum_x \left(I(x;\Delta x) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at x

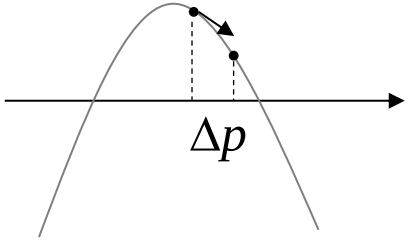
$$I(x+\Delta x) \approx I(x) + \nabla I \Delta x$$



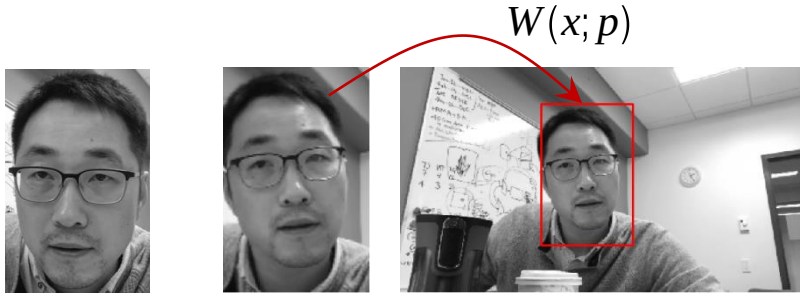
: steepest descent images

2. Find Δx that minimizes the obj. function at x

$$\nabla I \Delta x = T(x) - I(x) = -(I(x) - T(x)) \quad \triangleq \quad I_x \Delta u + I_y \Delta v = -I_t$$



Optical Flow Derivation

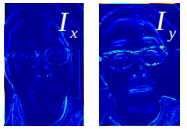


$$\Delta x^* = \underset{\Delta x}{\text{minimize}} \sum_x \left(I(x;\Delta x) - T(x) \right)^2$$

Guass-Newton's method

1.Linearize the obj. function at x

$$I(x+\Delta x) \approx I(x) + \nabla I \Delta x$$



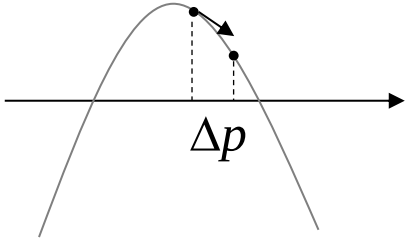
: steepest descent images

2. Find Δx that minimizes the obj. function at x

$$\nabla I \Delta x = T(x) - I(x) = -(I(x) - T(x)) \quad \triangleq \quad I_x \Delta u + I_y \Delta v = -I_t$$

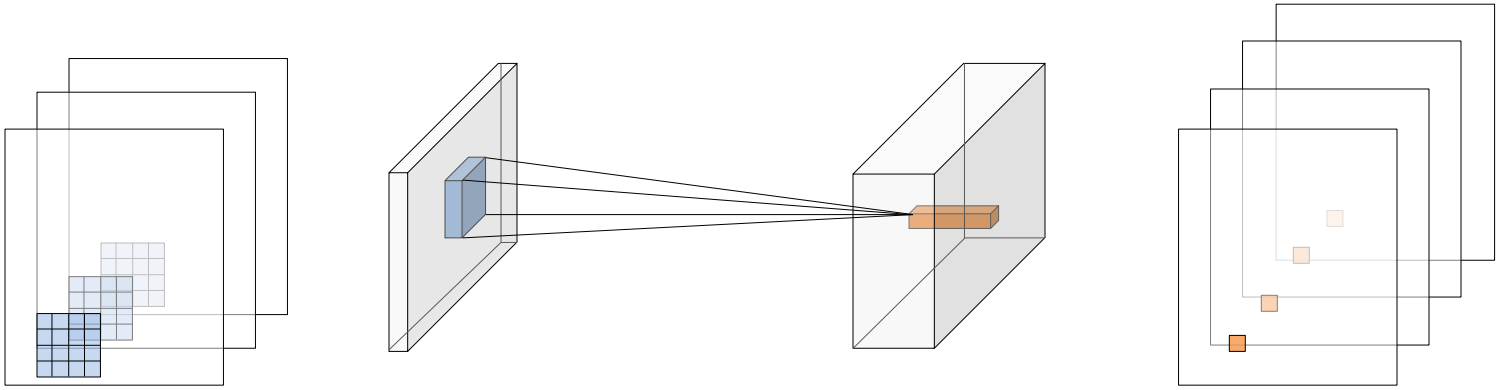
$$\Delta x = H^{-1} \sum_x \left(\nabla I \right)^T (T(x) - I(W(x;p)))$$

where $H = \sum_x \nabla I^T \nabla I$



CNN

Convolutional Layer



$H \times W \times C_i$

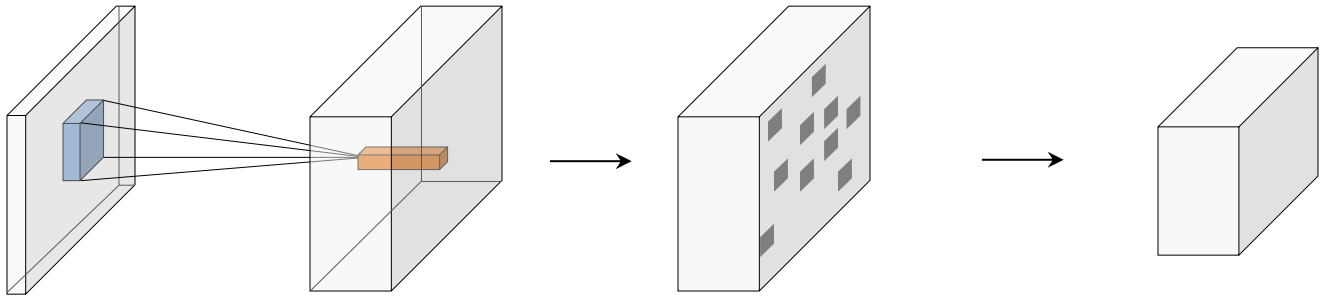
$H \times W \times C_o$

Multi-channel input

Multi-channel output

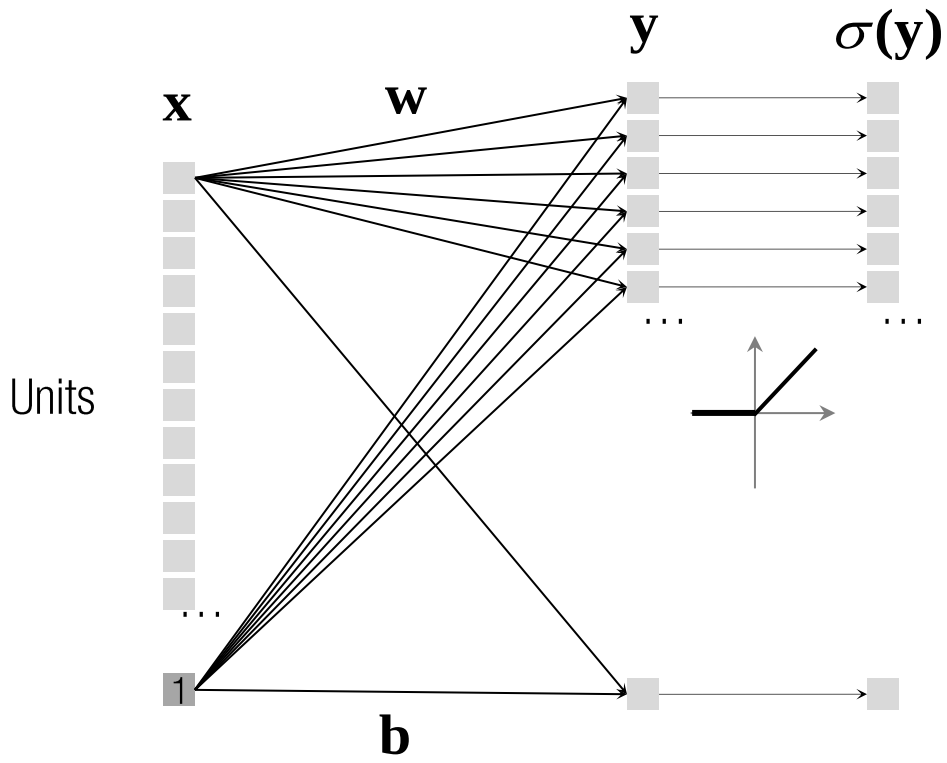
Feature map

Conv+ReLu+Pool

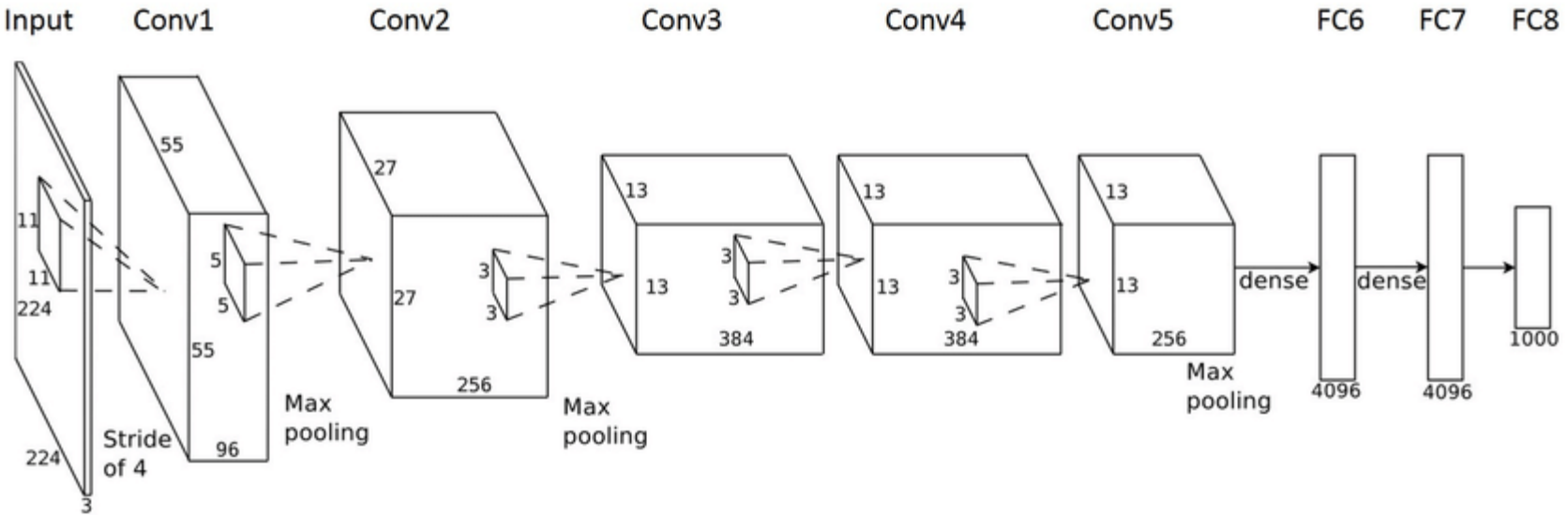


Operations:	Conv	ReLu	Max-pool
# of units:	$H \times W \times C_1$	$H \times W \times C_2$	$H \times W \times C_2$	$H_1 \times W_1 \times C_2$
# of weights:	$F_1 \times F_2 \times C_1 \times C_2$	0	0
# of biases:	$1 \times C_2$	0	0

FC+ReLu

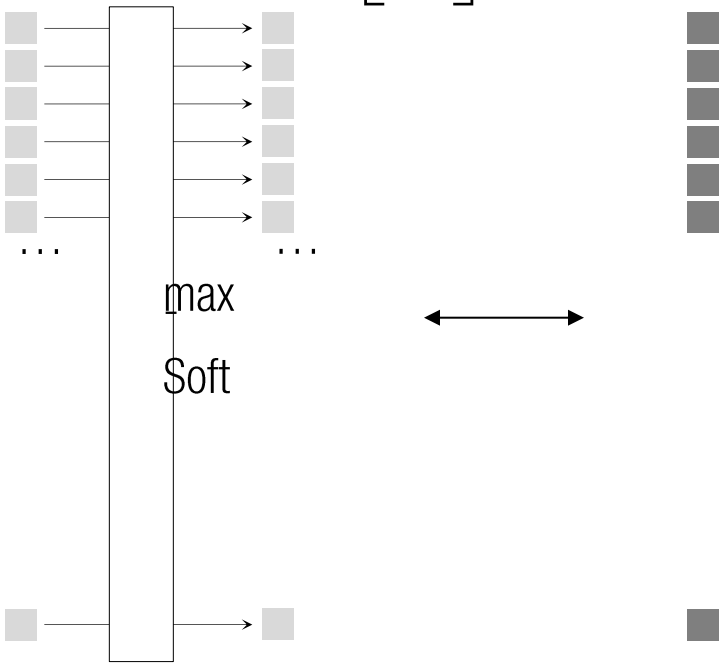


ALEX NET


Input
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
224
224
3
11
11
55
55
96
5
5
Stride of 4
27
27
256
3
3
Max pooling
13
13
384
3
3
Max pooling
13
13
384
3
3
13
13
256
Max pooling
dense
4096
dense
4096
1000

ERROR MEASURE (LOSS)

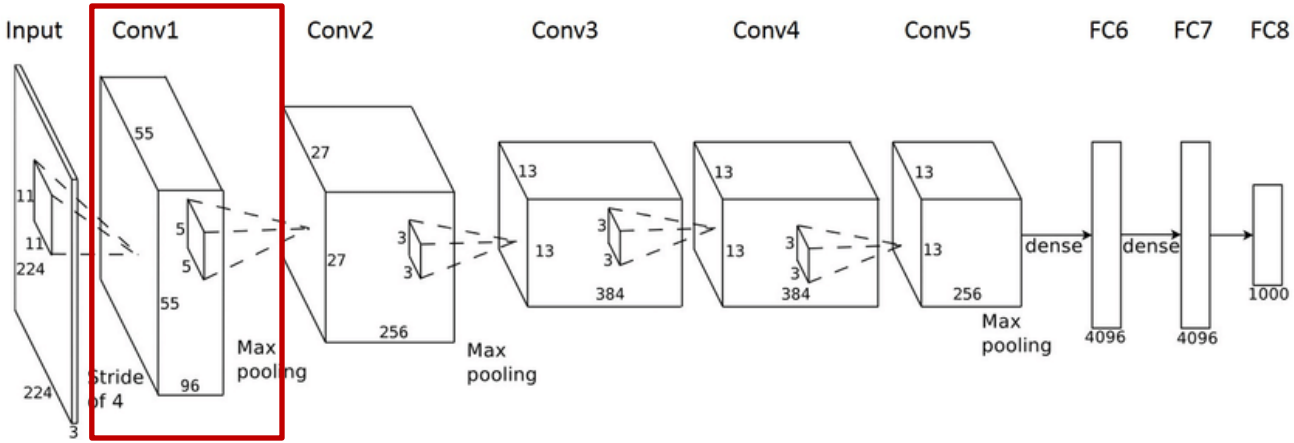
$\mathbf{x} \in \mathbb{R}^L$ $\quad \tilde{\mathbf{y}} \in [0,1]^L$ $\quad \tilde{\mathbf{y}} \in \{0,1\}^L$ $\quad L$: # of class labels



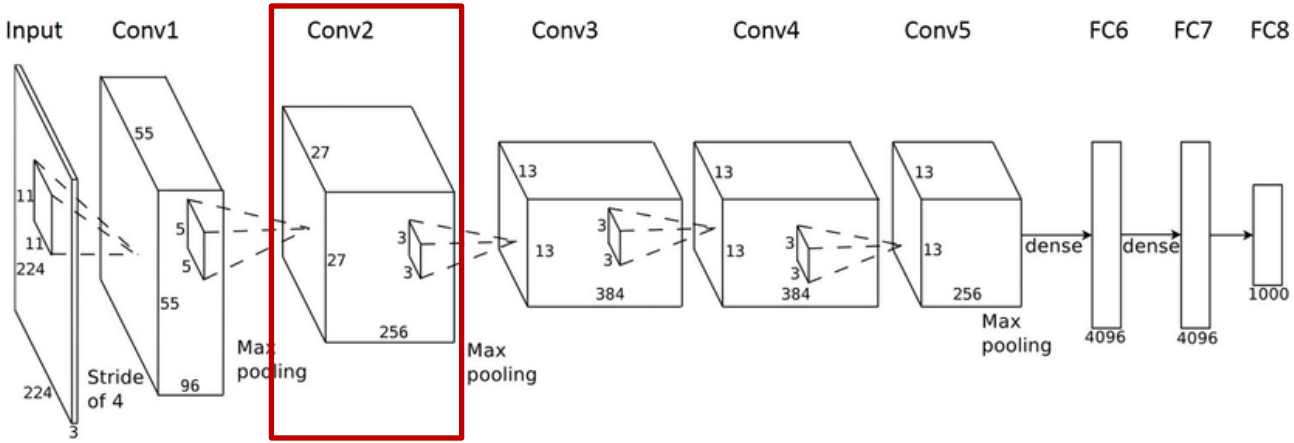
Prediction Ground truth

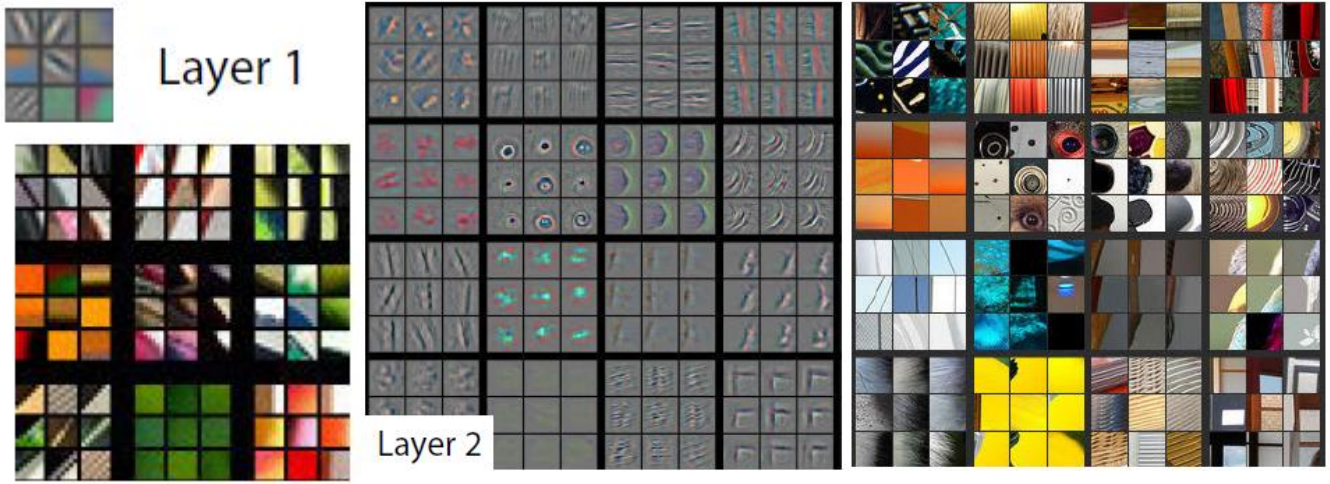
Cross-entropy loss (matching prob. distribution)

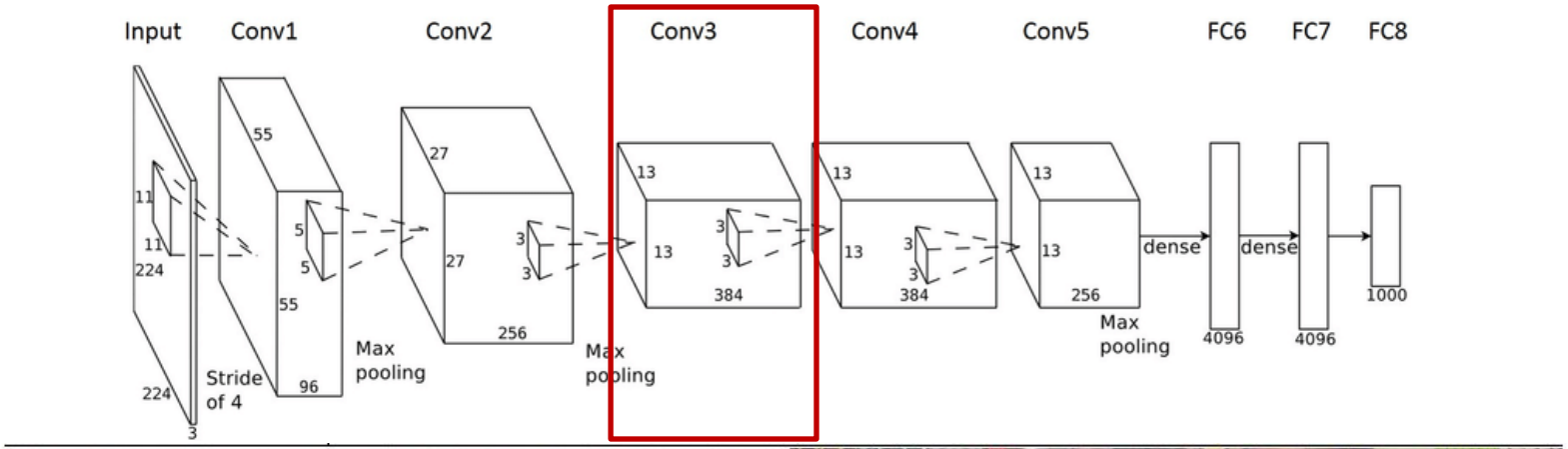
$$L = \sum_i \mathbf{y}_i \log \tilde{\mathbf{y}}_i$$

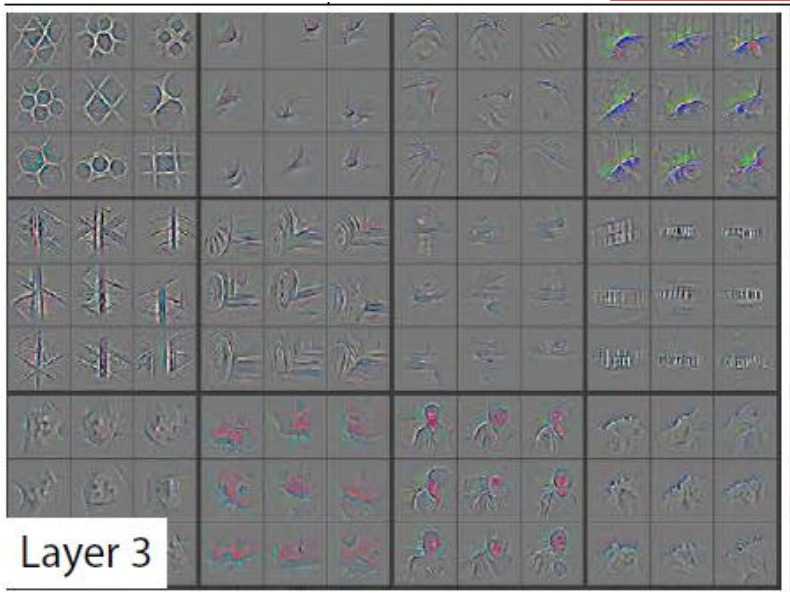

Input
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
224
224
3
11
11
55
55
96
5
5
Stride of 4
Max pooling
27
27
256
3
3
Max pooling
13
13
384
3
3
13
13
384
3
3
13
13
256
Max pooling
dense
4096
dense
4096
1000

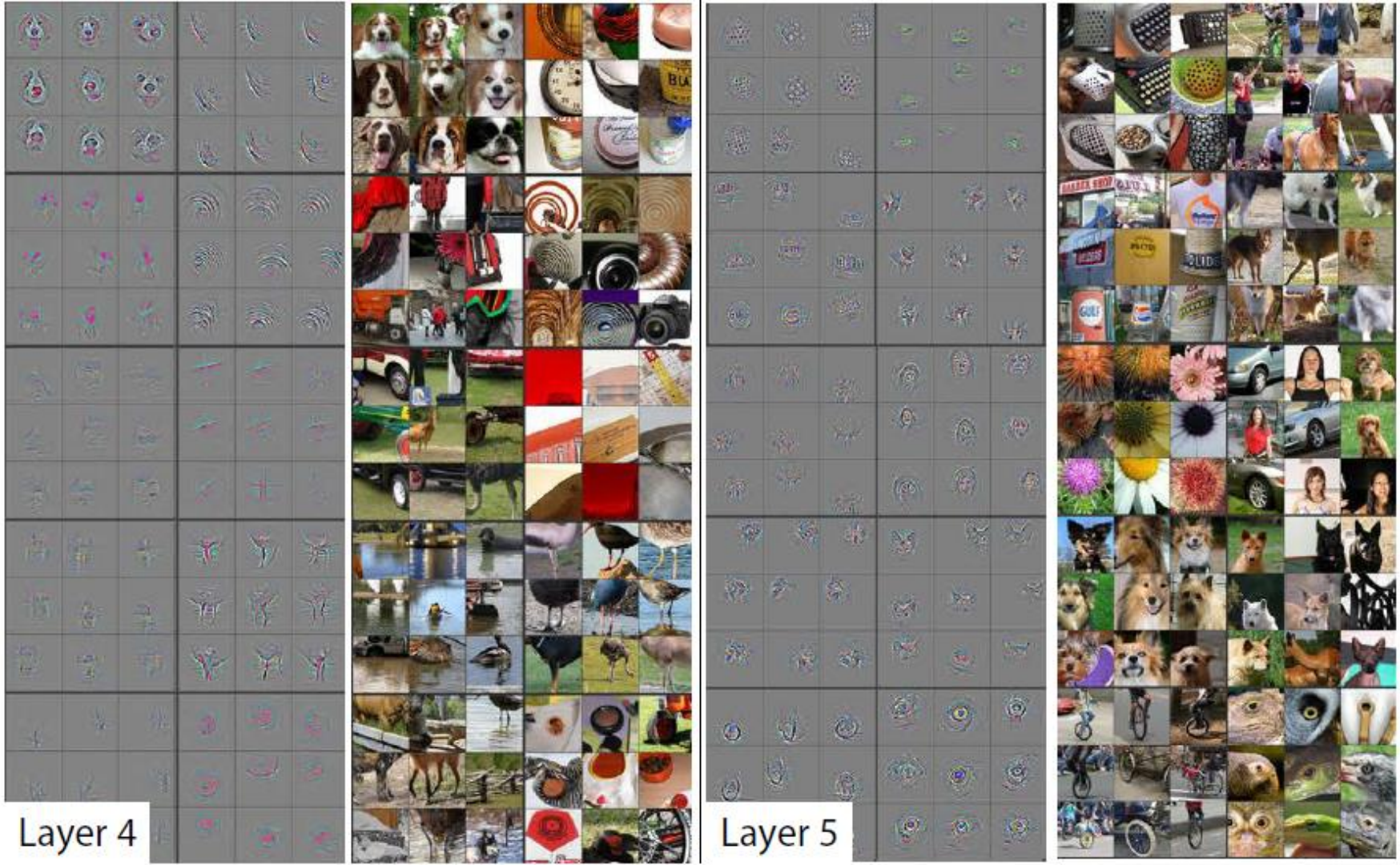
Layer 1


Input
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
224
224
3
11
11
55
55
96
5
5
Stride of 4
27
27
256
3
3
Max pooling
13
13
384
Max pooling
13
13
384
13
13
256
Max pooling
dense
4096
dense
4096
1000

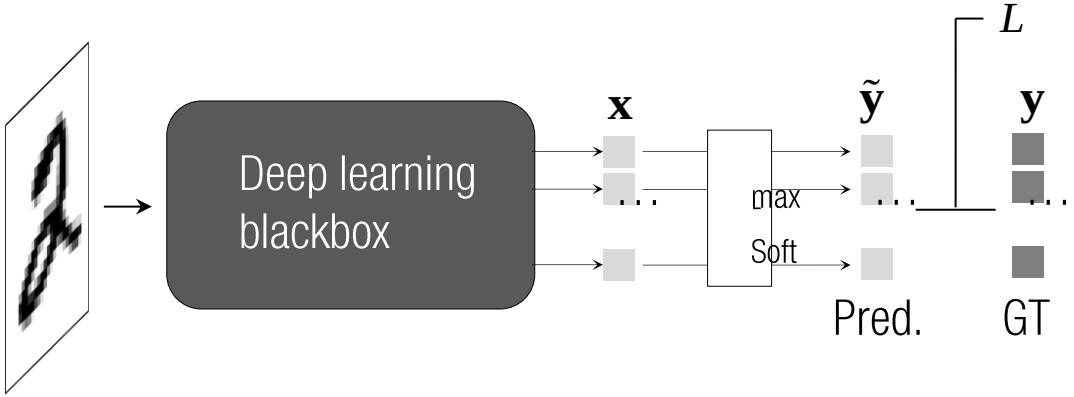

Layer 1
Layer 2


Input
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
11
11
224
224
3
55
55
5
5
96
Stride
of 4
27
27
5
256
Max
pooling
13
13
3
3
384
Max
pooling
13
13
3
3
384
13
13
3
3
256
Max
pooling
dense
4096
dense
4096
1000

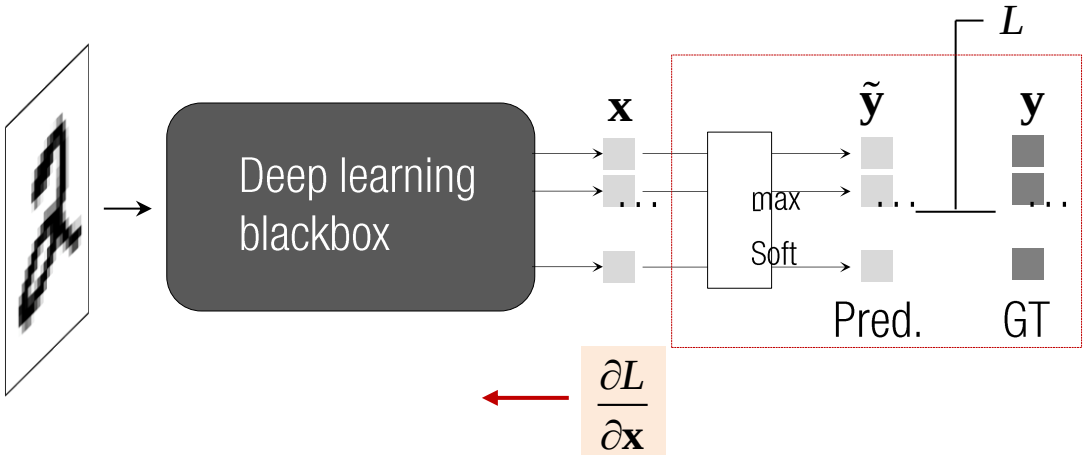

Layer 3


Layer 4
Layer 5

ENTROPY LOSS DERIVATIVE



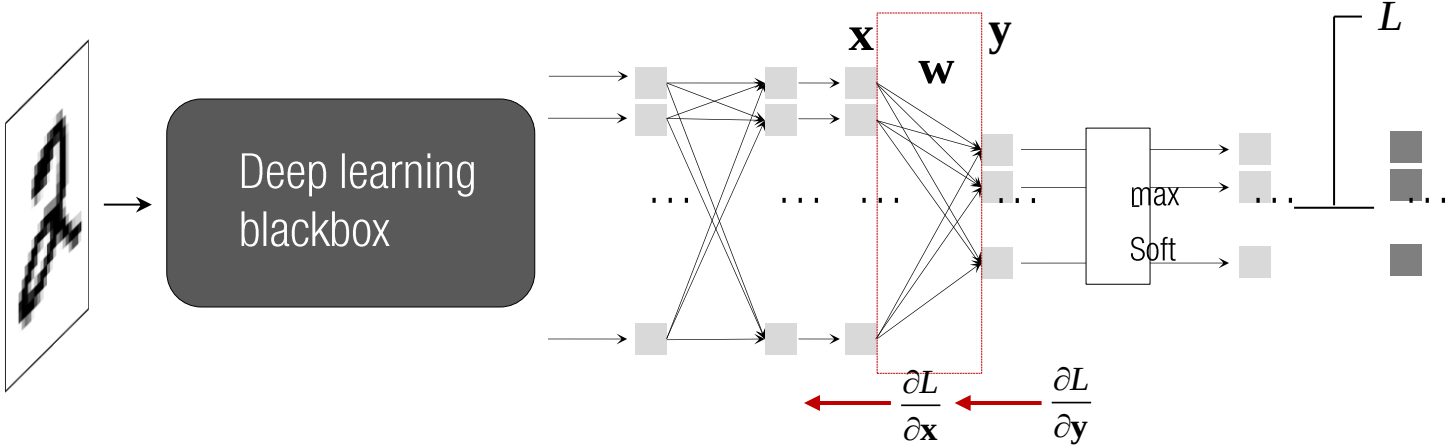
ENTROPY LOSS DERIVATIVE



function [L, dLdx] = Loss(x, y)

Input: $\mathbf{x}$	$\frac{\partial L}{\partial \mathbf{x}_i} = \tilde{\mathbf{y}}_i - \mathbf{y}_i$ 1 x n
Trainable var.: None	None
Output: $L = \sum_i \mathbf{y}_i \log \tilde{\mathbf{y}}_i$ where $\tilde{\mathbf{y}}_i = \frac{e^{\mathbf{x}_i}}{\sum_i e^{\mathbf{x}_i}}$	

FULLY CONNECTED LAYER



Input: $\mathbf{x} \in \mathbb{R}^n$

Trainable var.: $\mathbf{w} \in \mathbb{R}^{m \times n}$

Output: $\mathbf{y} = \mathbf{w}\mathbf{x}$

$\mathbf{y} \in \mathbb{R}^m$

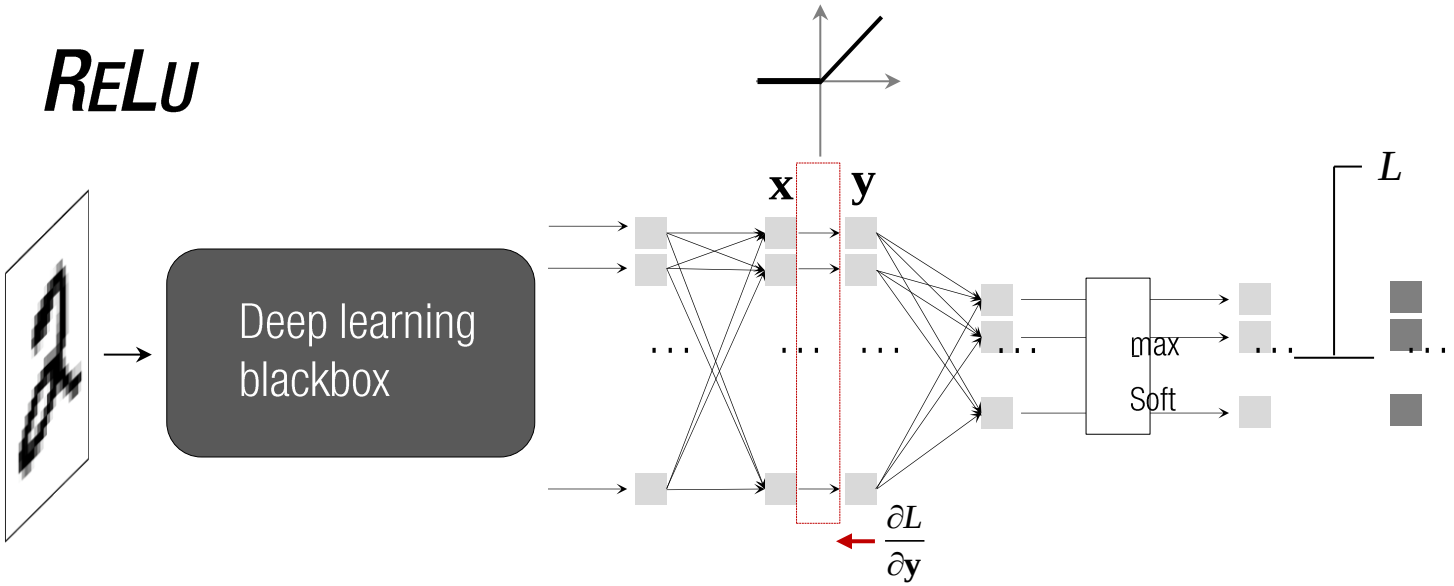
$$\frac{\partial L}{\partial \mathbf{x}_i} = \sum_j \frac{\partial L}{\partial \mathbf{y}_j} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i} = \sum_j \frac{\partial L}{\partial \mathbf{y}_j} \mathbf{w}_{ji} \qquad 1\text{xn}$$

$$\frac{\partial L}{\partial \mathbf{w}_{ij}} = \sum_i \frac{\partial L}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial \mathbf{w}_{ij}} = \frac{\partial L}{\partial \mathbf{y}_i} \mathbf{x}_j \qquad 1\text{x(nm)}$$

function [y] = FC(x, w)

function [dLdx, dLdw, dLdb] = FC_back(dLdy, x, w, y)

ReLu



Input: $\mathbf{x} \in \mathbb{R}^n$

$$\frac{\partial L}{\partial \mathbf{x}_i} = \sum_j \frac{\partial L}{\partial \mathbf{y}_j} \frac{\partial}{\partial \mathbf{x}_i} \max(0, \mathbf{x}_j) = \frac{\partial L}{\partial \mathbf{y}_i} \frac{\partial}{\partial \mathbf{x}_i} \max(0, \mathbf{x}_i) = \begin{cases} \frac{\partial L}{\partial \mathbf{y}_i} & \text{if } \mathbf{y}_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

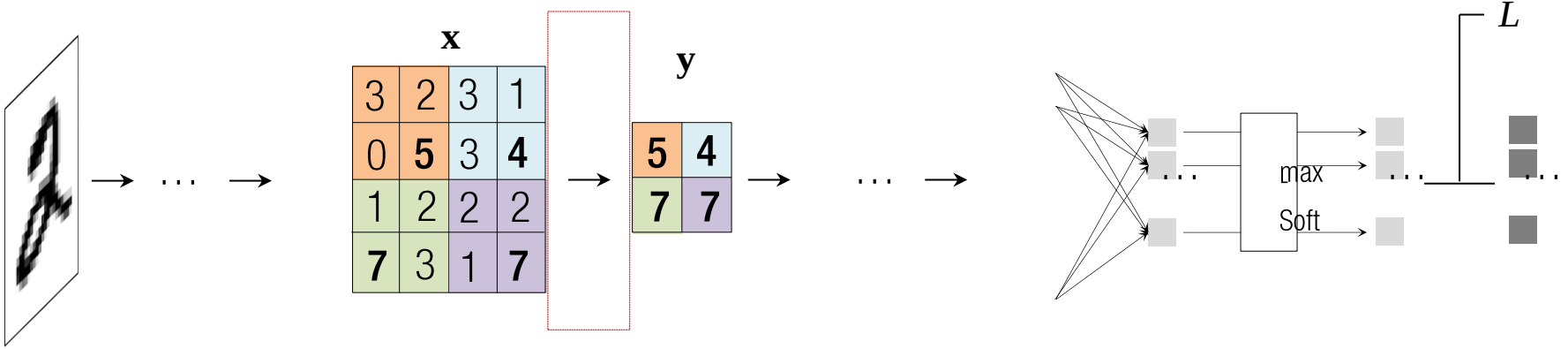
Trainable var.: None

Output: $\mathbf{y}_i = \max(0, \mathbf{x}_i)$

$\mathbf{y} \in \mathbb{R}^n$

function [y] = Relu(x)
function [dLdx] = Relu_back(dLdy, x, w, y)

MAX-POOL



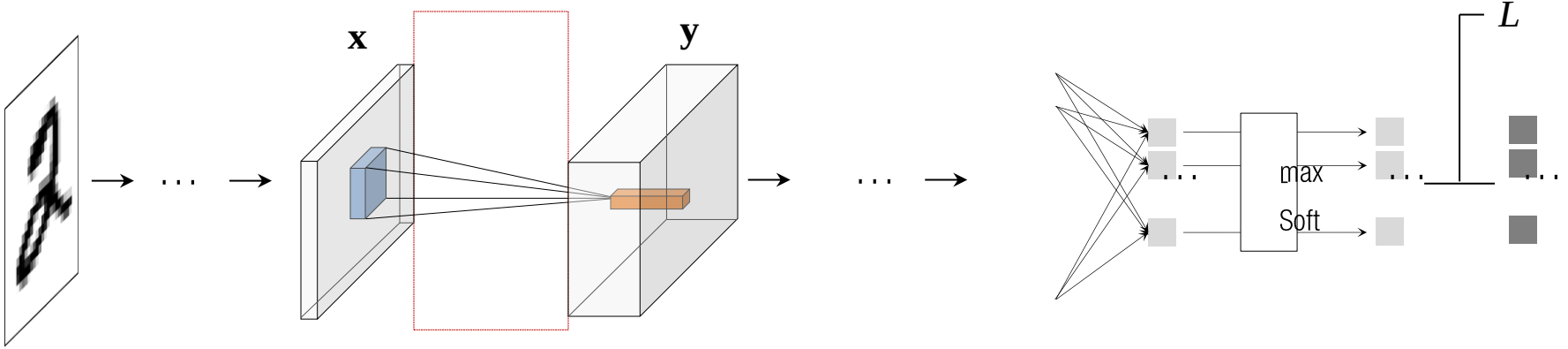
Input: $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$

Trainable var.: None

Output: $\mathbf{y} \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times C}$

```
function [y] = Maxpool(x, size, stride)
function [dLdx] = Maxpool_back(dLdy, x, size, stride, y)
```

CONVOLUTIONAL OPERATION



Input: $\mathbf{x} \in \mathbb{R}^{H \times W \times C_1}$

Trainable var.: $\mathbf{w} \in \mathbb{R}^{F \times F \times C_1 \times C_2}$

Output: $\mathbf{y} = \mathbf{x} * \mathbf{w}$

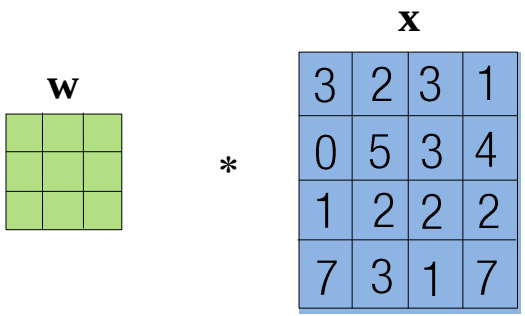
$\mathbf{y} \in \mathbb{R}^{H \times W \times C_2}$

$$\frac{\partial L}{\partial \mathbf{x}_{ijk}} = \sum_m \sum_n \sum_l L_{mnl} \mathbf{w}_{m-i,n-j,k,l}$$

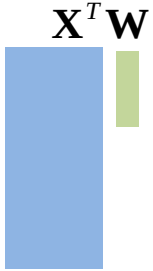
$$\frac{\partial L}{\partial \mathbf{w}_{ijcd}} = \sum_k \sum_l L_{kld} \mathbf{x}_{k+i,l+j,c}$$

function [y] = Conv(x, w, b)
function [dLdx dLdw dLdb] = Conv_back(dLdy, x, w, b, y)

Convolution via im2col (Derivative w.r.t. W)

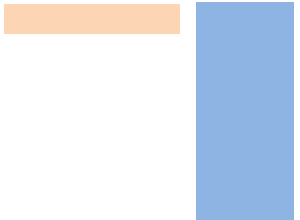


$$\frac{\partial L}{\partial \mathbf{w}_{ij}} = \sum_k \sum_l \frac{\partial L}{\partial \mathbf{y}_{kl}} \frac{\partial \mathbf{y}_{kl}}{\partial \mathbf{w}_{ij}}$$

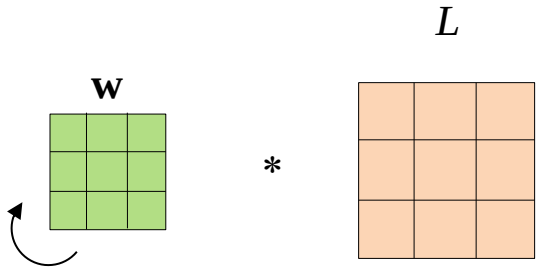


$$\frac{\partial \mathbf{y}}{\partial \mathbf{w}} = \mathbf{X}^T$$

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{w}}$$



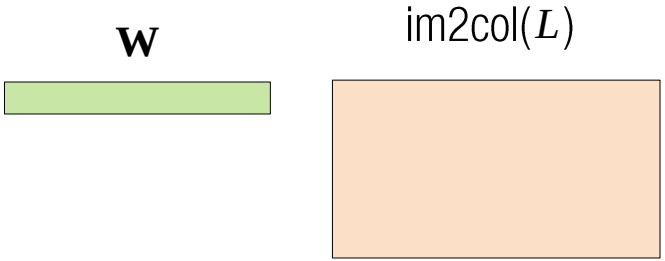
Convolution via im2col (Derivative w.r.t. X)



$$= \frac{\partial L}{\partial \mathbf{x}}$$

$$\frac{\partial L}{\partial \mathbf{x}_{ij}} = \sum_k \sum_l L_{ij} \mathbf{w}_{i-k,j-l}$$

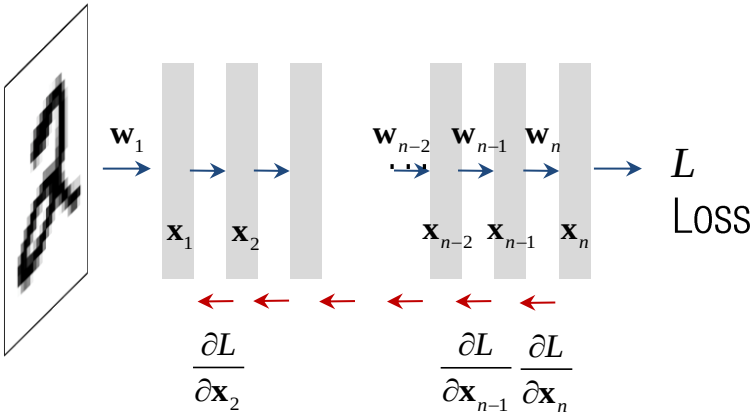
Reverse order



$$= \frac{\partial L}{\partial \mathbf{x}}$$

Row version of dLdx

SUMMARY



Forward prediction

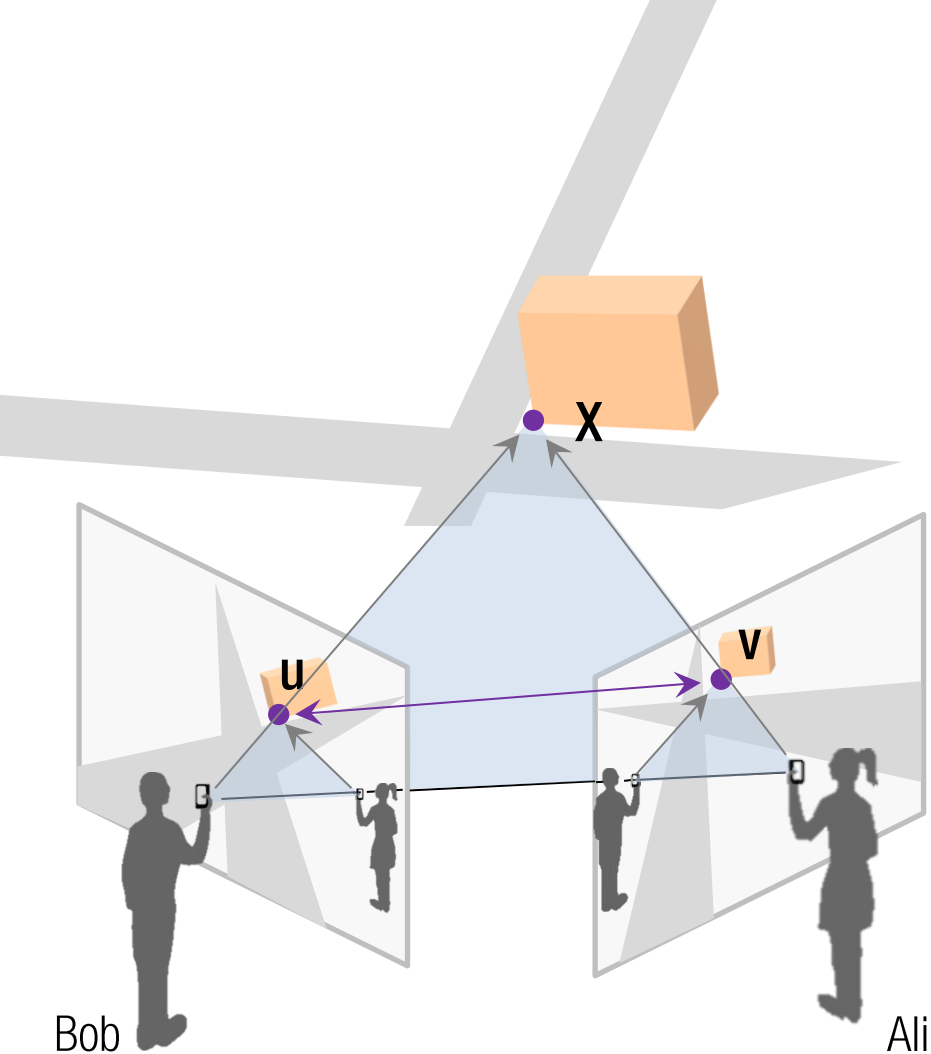
```
pred1 = conv(x, w1)
pred2 = relu(pred1)
pred3 = pool(pred2)

pred10 = flatten(pred9)
pred11 = fc(pred10, w10)

pred16 = loss_ce_sm(pred15, label)
```

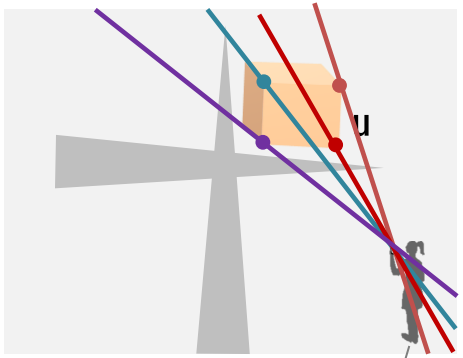
Back-propagation

```
dLdx = loss_ce_back(pred15, label)

dLdx, dLdw10 = fc_back(pred10, w10, pred11)
dLdx = flatten_back(dLdx, pred9, pred10)

dLdx = pool_back(dLdx, pred2, pred3)
dLdx = relu_back(dLdx, pred1, pred2)
dLdx, dLdw1 = conv_back(dLdx, x, w1,pred1)
```

```
function [dLdx dLdy] = foo_back(dL, x, y)
```

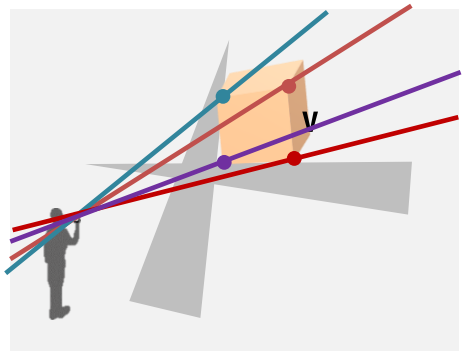
Epipolar Geometry





Bob's image

Alice from Bob's view

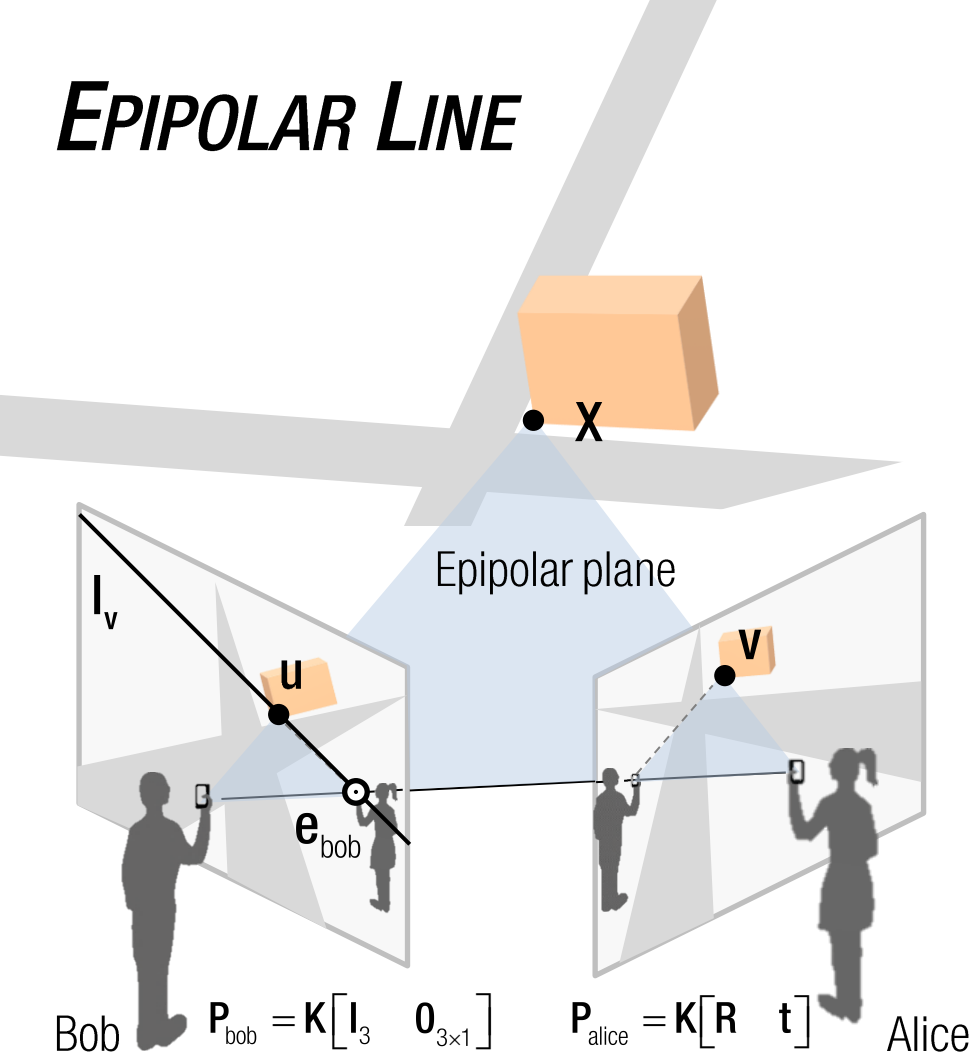


Alice's image

Epipolar constraint between two images:

1. A point, **u**, in Bob's image corresponds to an epipolar line $\mathbf{l}_u$ in Alice's image.
2. The epipolar line passes the corresponding point in Alice's image, **v**: $\mathbf{v}^T\mathbf{l}_u = 0$
3. Any point along the epipolar line can be a candidate of correspondences.
4. Epiploar lines meet at the epipole: $\mathbf{e}_{bob}^T\mathbf{l}_u = 0$ $\mathbf{e}_{alice}^T\mathbf{l}_v = 0$

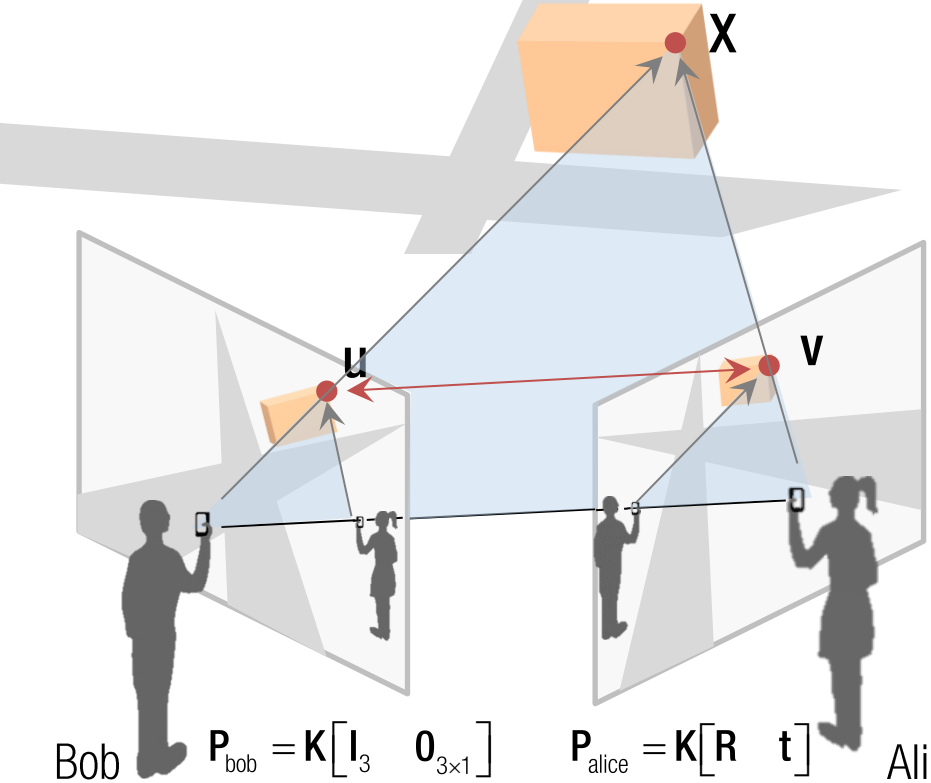
EPIPOLAR LINE

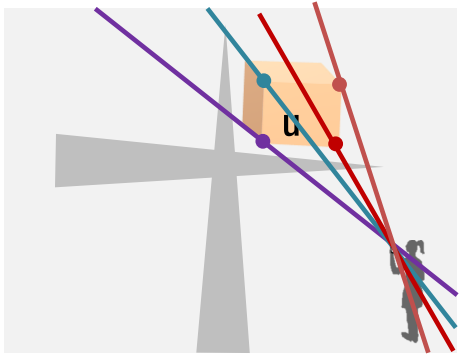


$$l_v = Fv$$

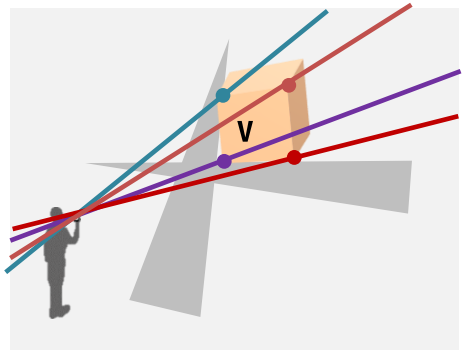
Fundamental matrix

Fundamental Matrix





Bob's image



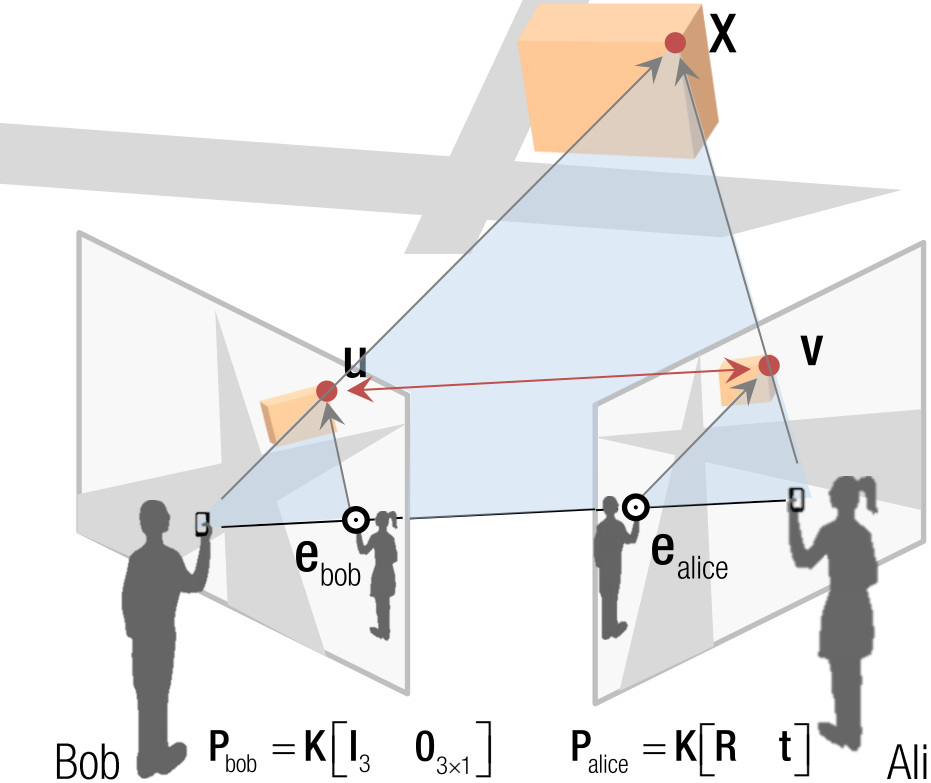
Alice's image

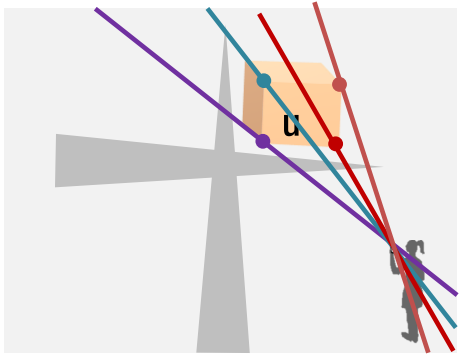
$$\mathbf{v}^T \mathbf{l}_u = \mathbf{v}^T \underbrace{\mathbf{K}^{-T} [\mathbf{t}]_\times \mathbf{R} \mathbf{K}^{-1}}_{\text{Common for all points}} \mathbf{u} = \mathbf{0}$$

$$= \mathbf{v}^T \mathbf{F} \mathbf{u} = 0$$

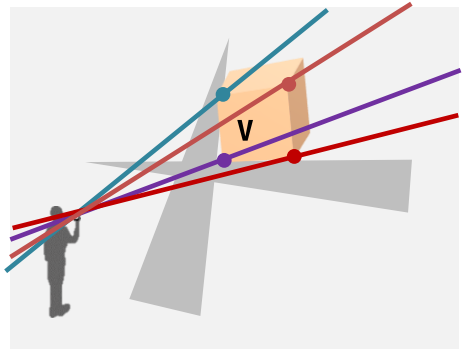
$$= \mathbf{v}^T (\mathbf{F}\mathbf{u}) = \mathbf{u}^T (\mathbf{F}^T \mathbf{v}) = 0$$

FUNDAMENTAL MATRIX





Bob's image

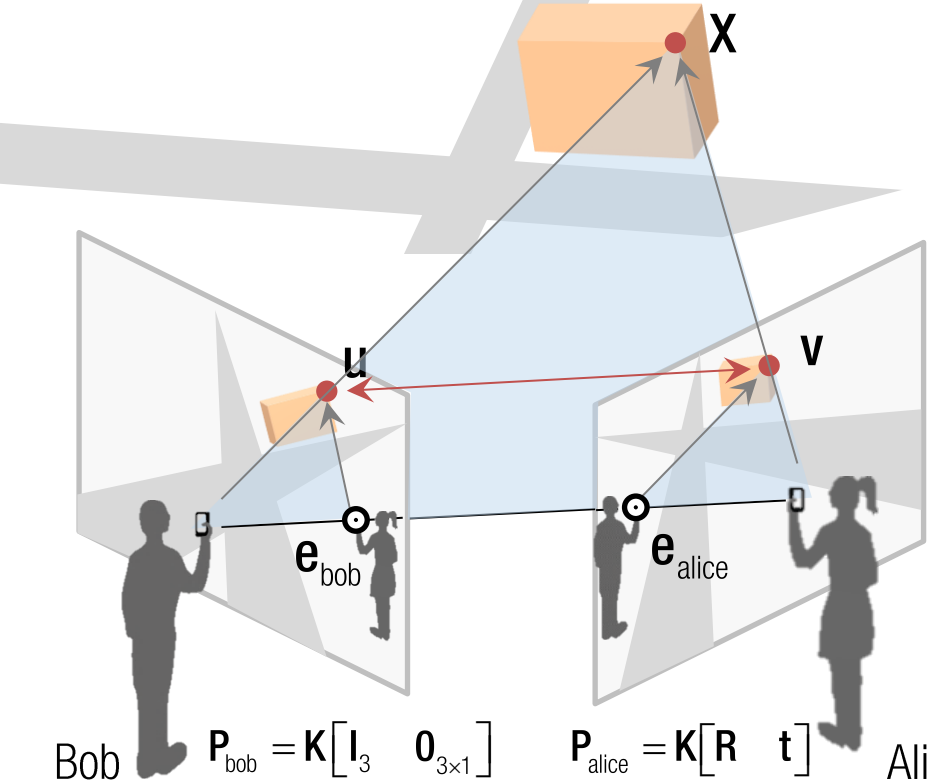


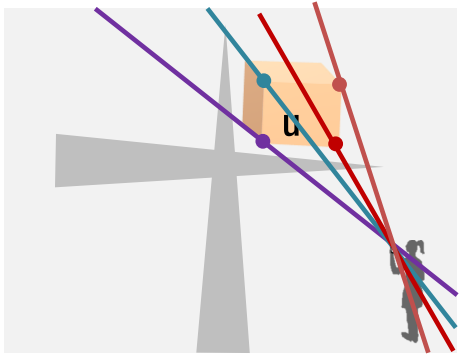
Alice's image

Properties of Fundamental Matrix

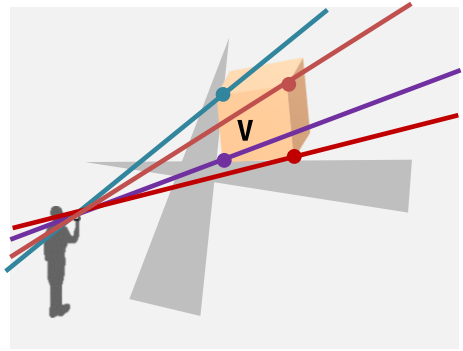
- Transpose: if **F** is for $\mathbf{P}_{bob}$, $\mathbf{P}_{alice}$, then $\mathbf{F}^T$ is for $\mathbf{P}_{alice}$, $\mathbf{P}_{bob}$.

FUNDAMENTAL MATRIX





Bob's image

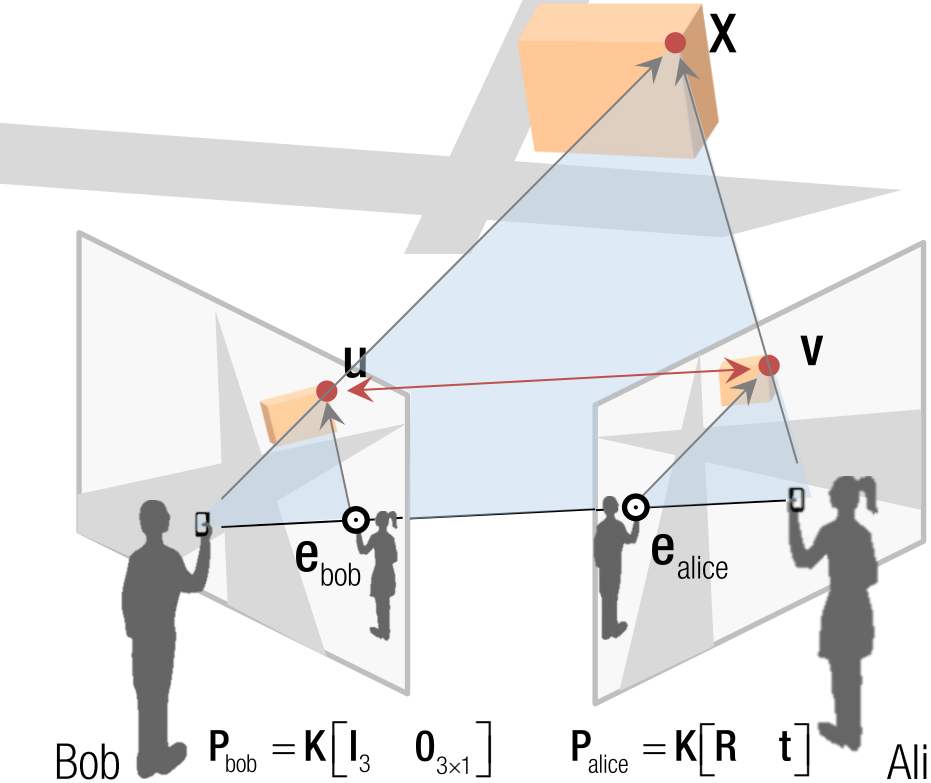


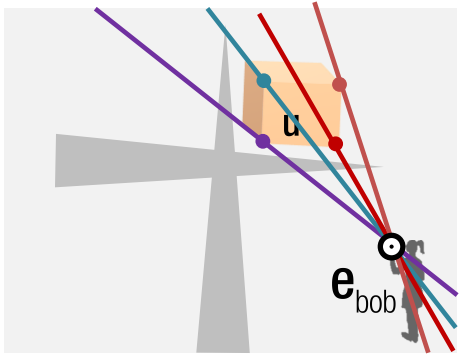
Alice's image

Properties of Fundamental Matrix

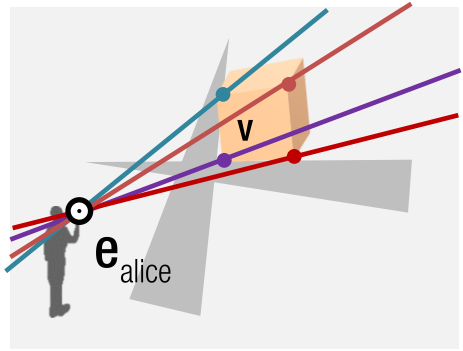
- Transpose: if $\mathbf{F}$ is for $\mathbf{P}_{bob}$, $\mathbf{P}_{alice}$, then $\mathbf{F}^T$ is for $\mathbf{P}_{alice}$, $\mathbf{P}_{bob}$.
- Epipolar line: $l_u = Fu \quad l_v = F^T v$

FUNDAMENTAL MATRIX





Bob's image



Alice's image

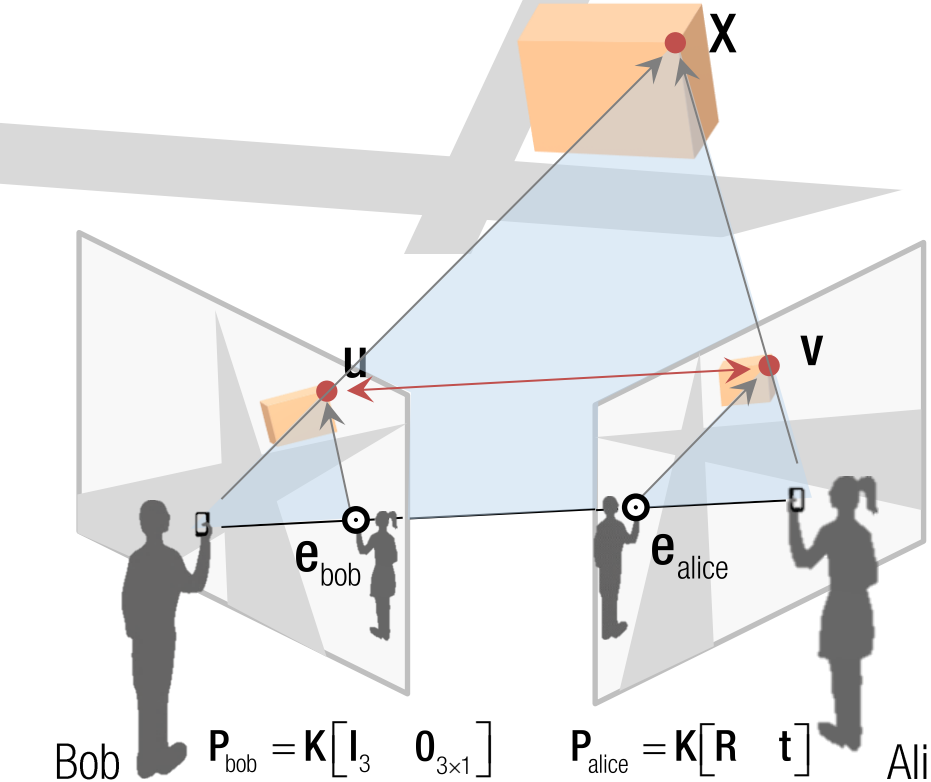
Properties of Fundamental Matrix

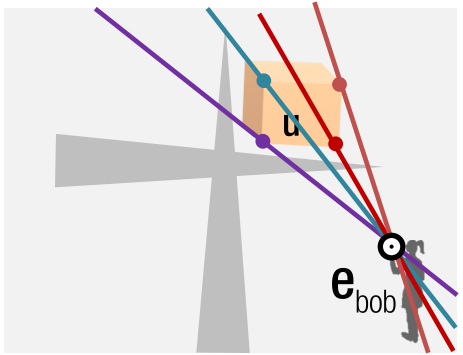
- Transpose: if $\mathbf{F}$ is for $\mathbf{P}_{bob}$, $\mathbf{P}_{alice}$, then $\mathbf{F}^T$ is for $\mathbf{P}_{alice}$, $\mathbf{P}_{bob}$.
- Epipolar line: $\mathbf{l}_u = \mathbf{F}\mathbf{u} \quad \mathbf{l}_v = \mathbf{F}^T\mathbf{v}$

  $\mathbf{F}\mathbf{e}_{bob} = \mathbf{0} \quad \mathbf{F}^T\mathbf{e}_{alice} = \mathbf{0}$
- Epipole: $\because \mathbf{v}_i^T\mathbf{F}\mathbf{e}_{bob} = 0, \quad \mathbf{u}_i^T\mathbf{F}^T\mathbf{e}_{alice} = 0, \quad \forall i$

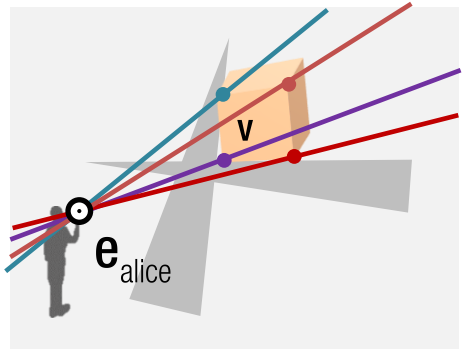
  $\longrightarrow \mathbf{e}_{bob} = \text{null}(\mathbf{F}), \quad \mathbf{e}_{alice} = \text{null}(\mathbf{F}^T)$

FUNDAMENTAL MATRIX





Bob's image



Alice's image

Properties of Fundamental Matrix

- Transpose: if $\mathbf{F}$ is for $\mathbf{P}_{bob}$, $\mathbf{P}_{alice}$, then $\mathbf{F}^T$ is for $\mathbf{P}_{alice}$, $\mathbf{P}_{bob}$.
- Epipolar line: $l_u = Fu \quad l_v = F^T v$
  $Fe_{bob} = 0 \quad F^T e_{alice} = 0$
- Epipole:
- rank($\mathbf{F}$)=2:
  DoF 9 (3x3 matrix)-1 (scale)-1 (rank)=7