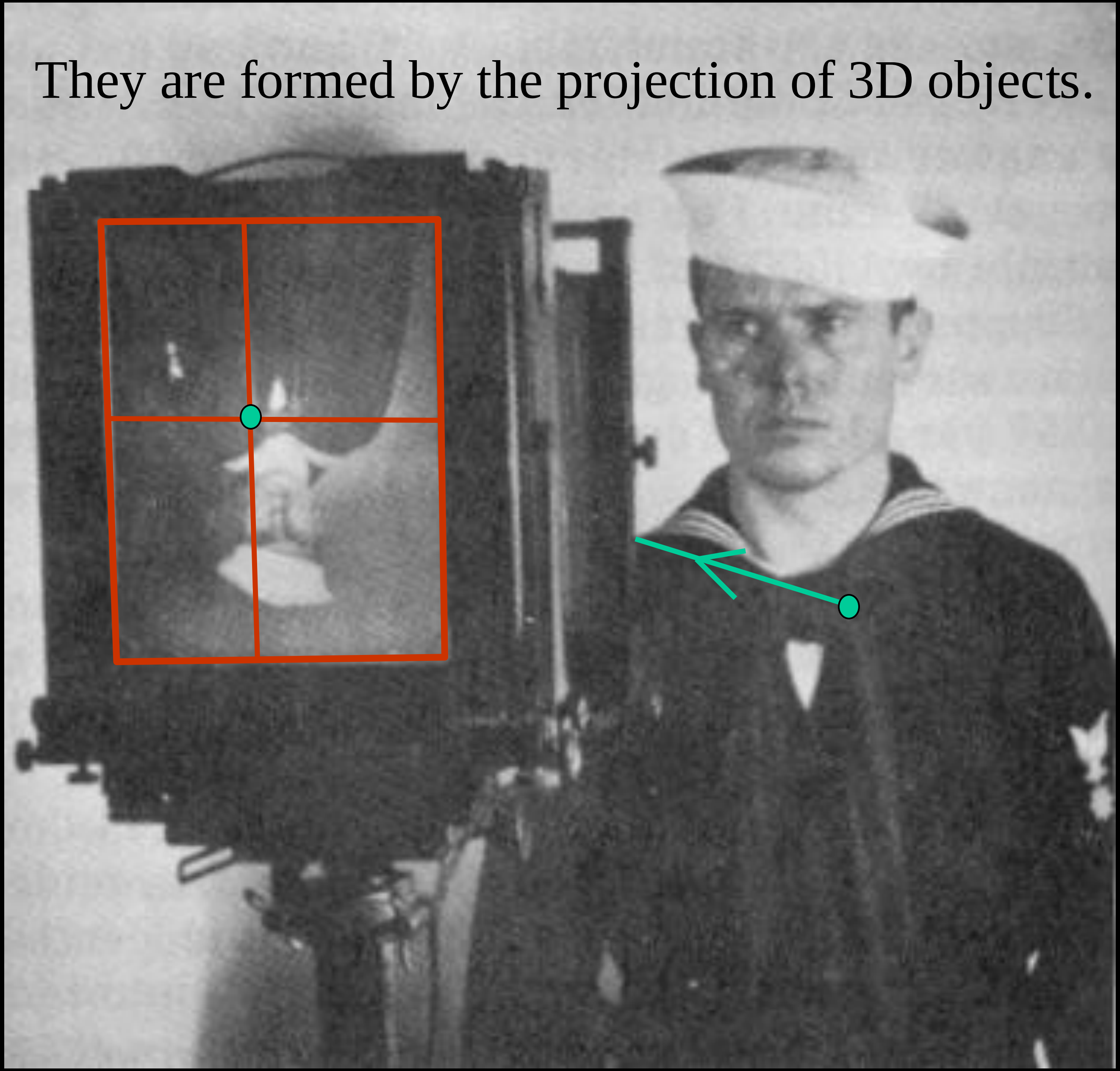
Elements of Camera geometry and Image processing

Jean Ponce
jean.ponce@ens.fr

Camera geometry and calibration

- Pinhole perspective projection
- Orthographic and weak-perspective models
- Non-standard models
- A detour through sensing country
- Intrinsic and extrinsic parameters
- Camera calibration


They are formed by the projection of 3D objects.

Images are two-dimensional patterns of brightness/color values

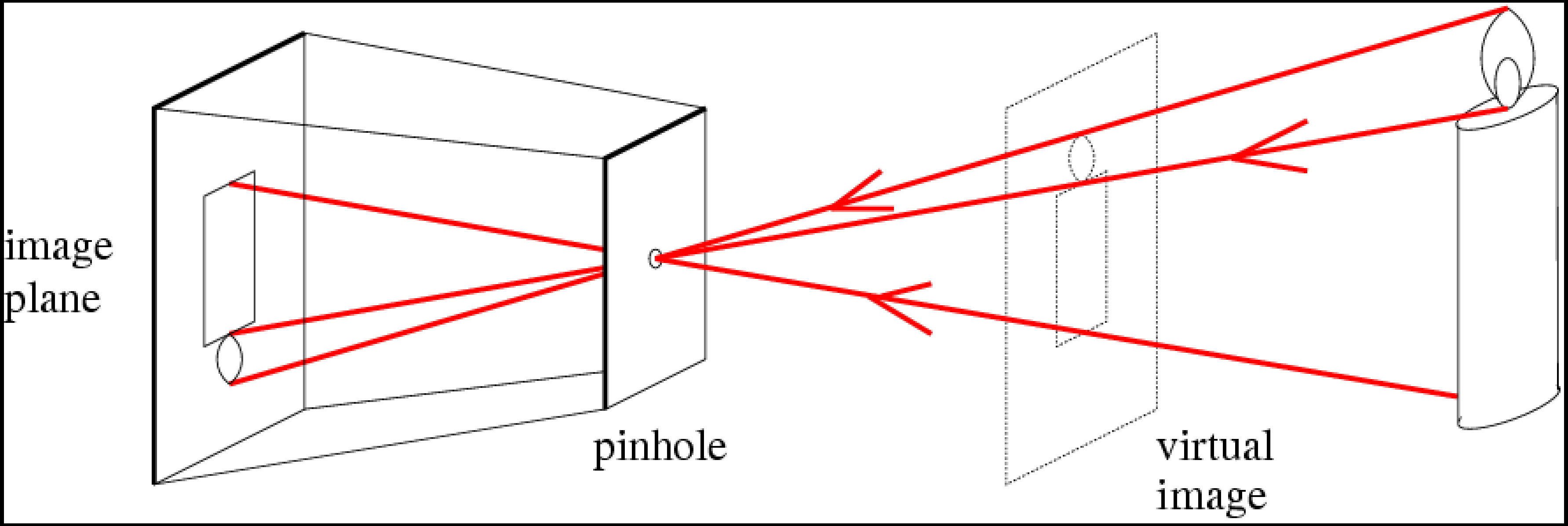
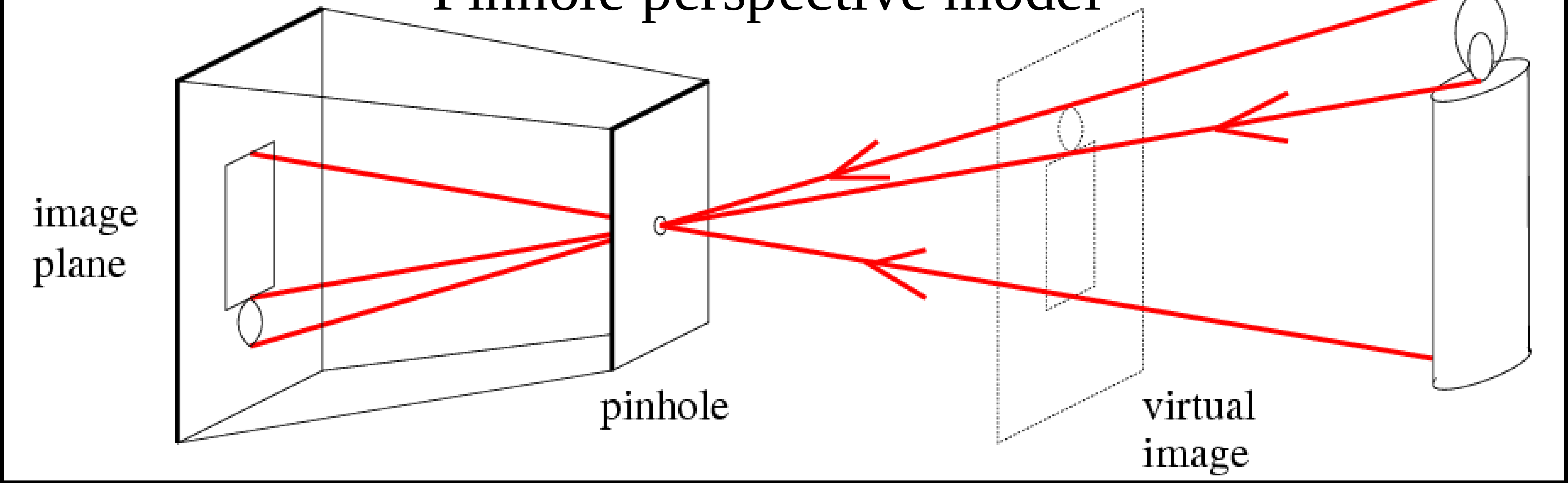
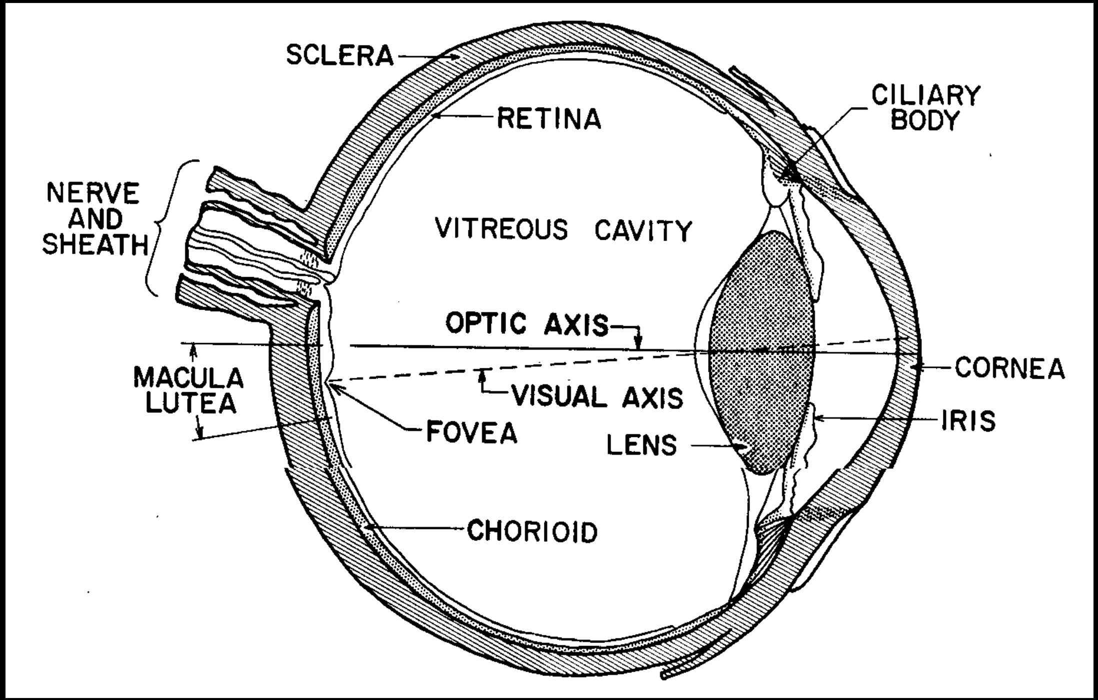
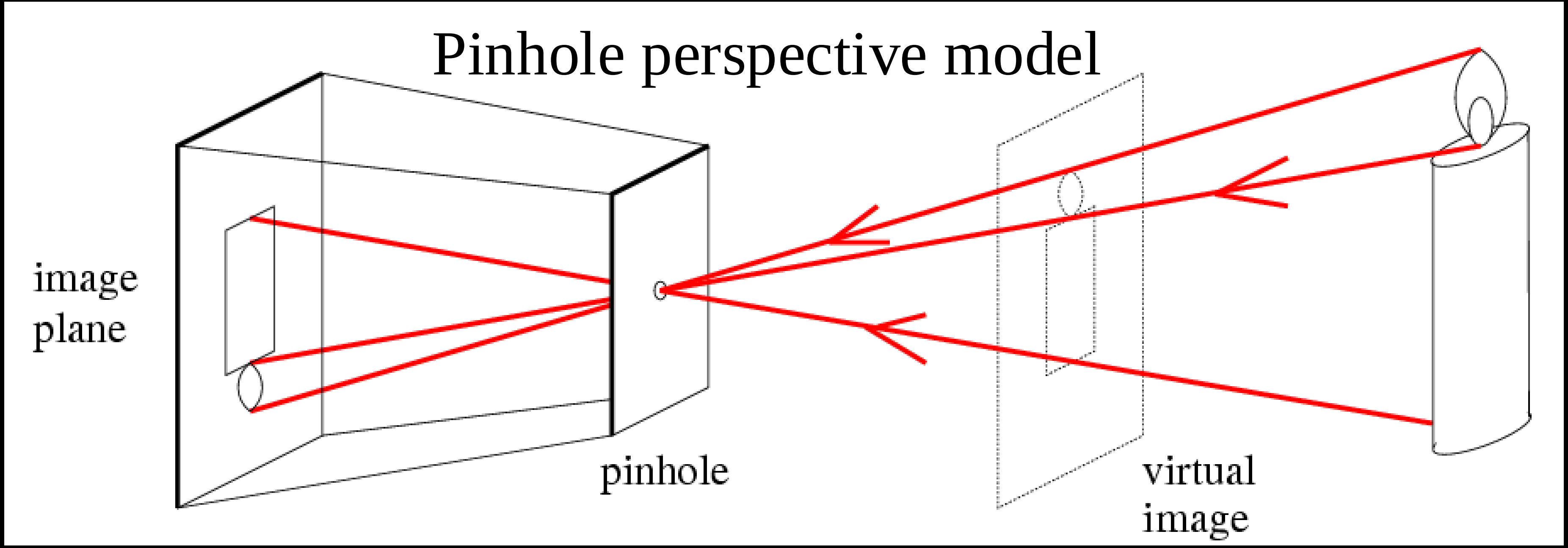


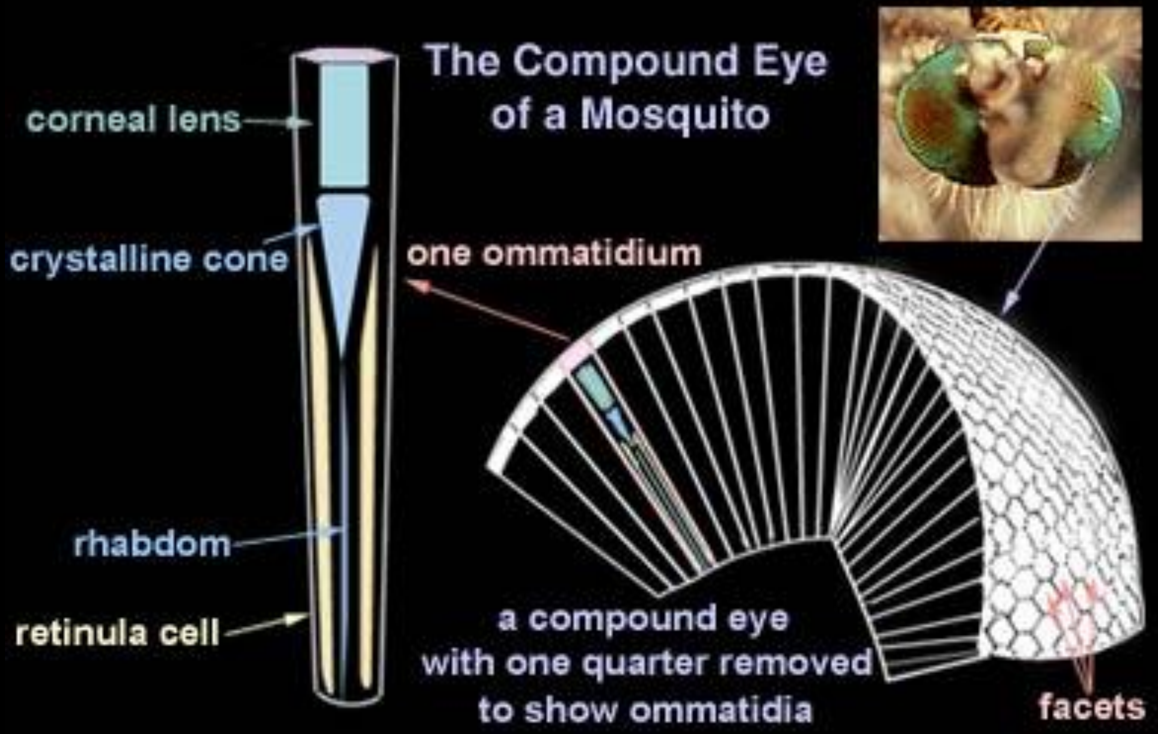
image plane
pinhole
virtual image

Pinhole perspective model



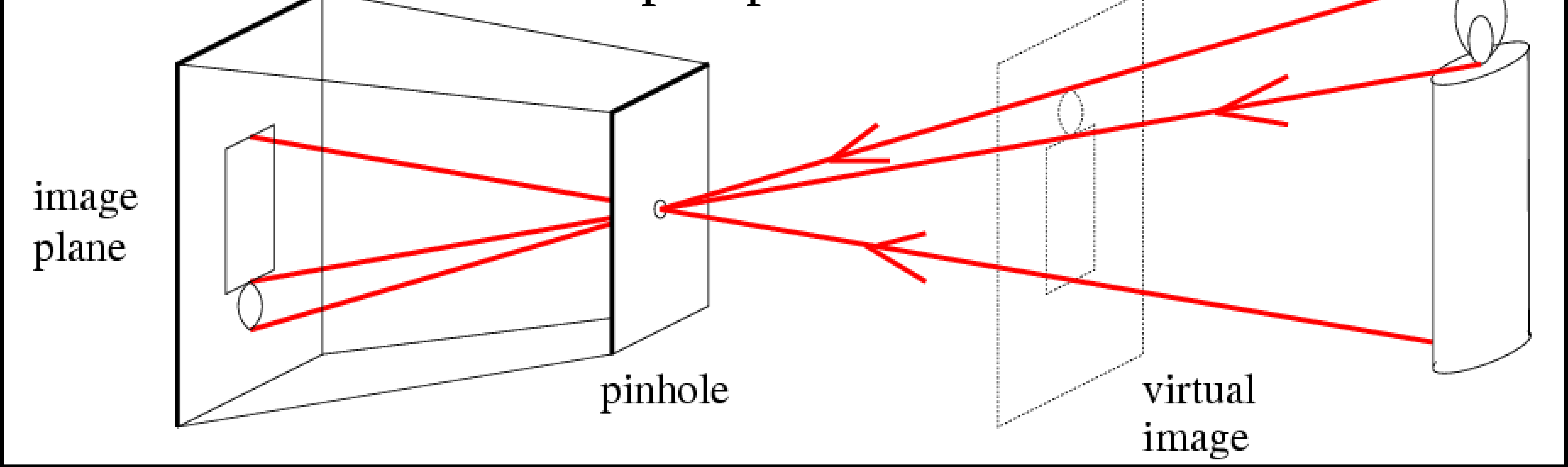


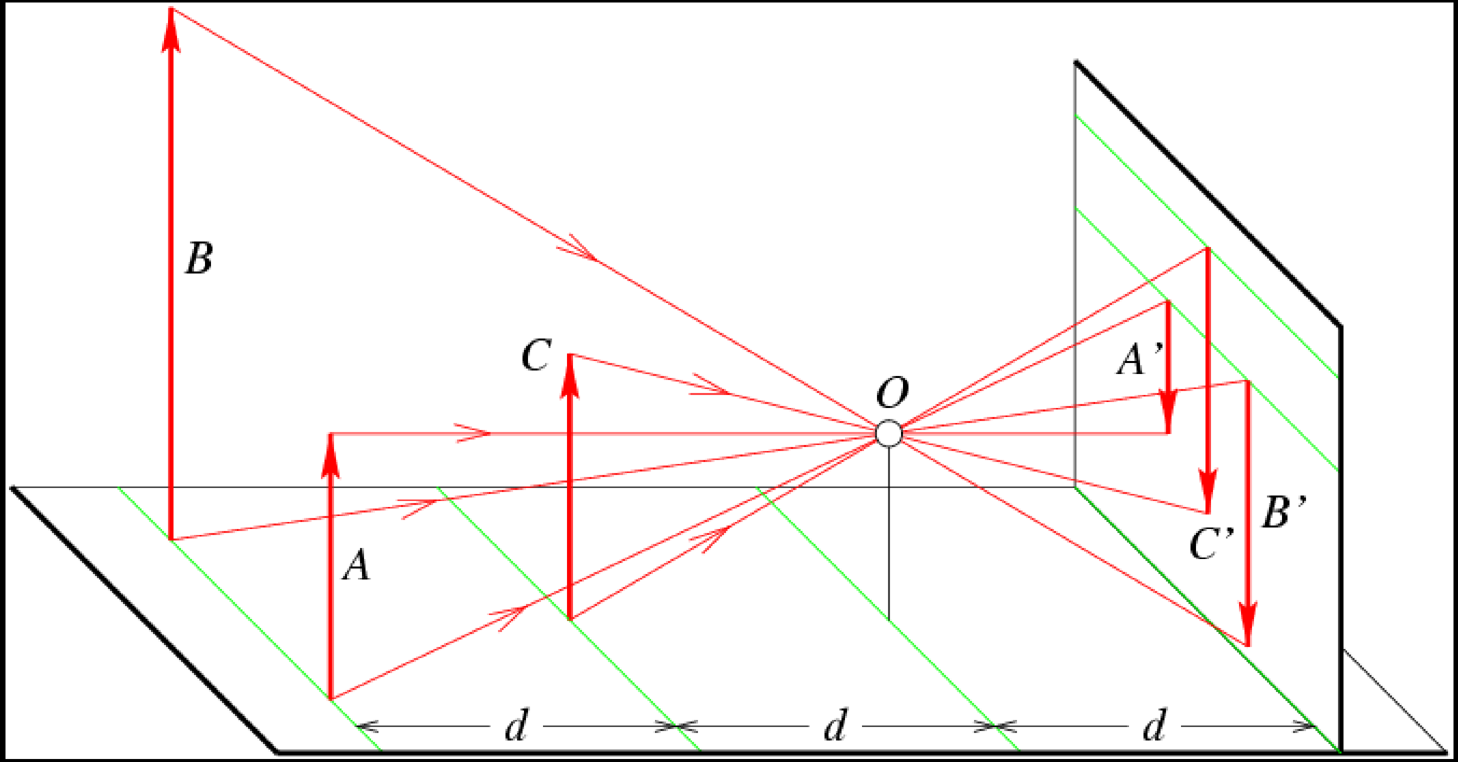


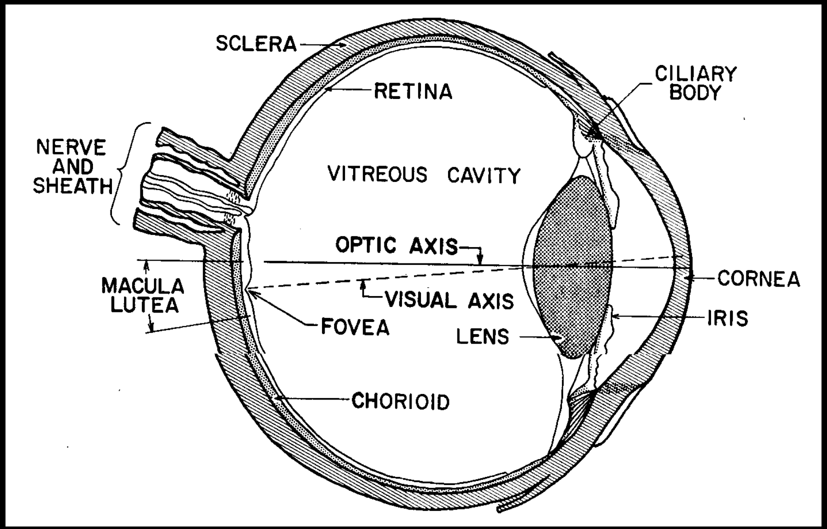

The Compound Eye of a Mosquito
corneal lens
crystalline cone
one ommatidium
rhabdom
retinula cell
a compound eye with one quarter removed to show ommatidia
facets

Land & Nilsson
"Animal Eyes"
Oxford, 2012

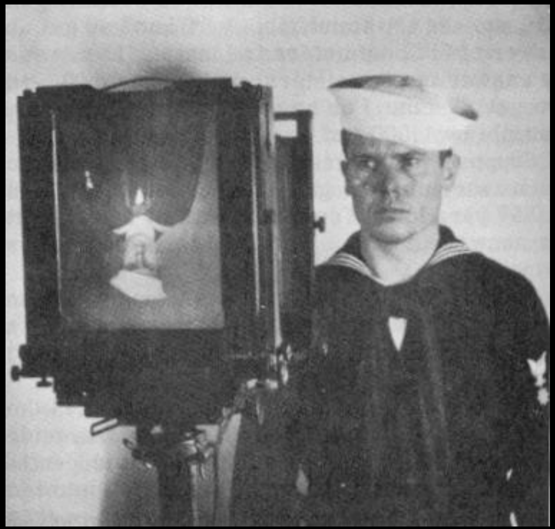
Pinhole perspective model



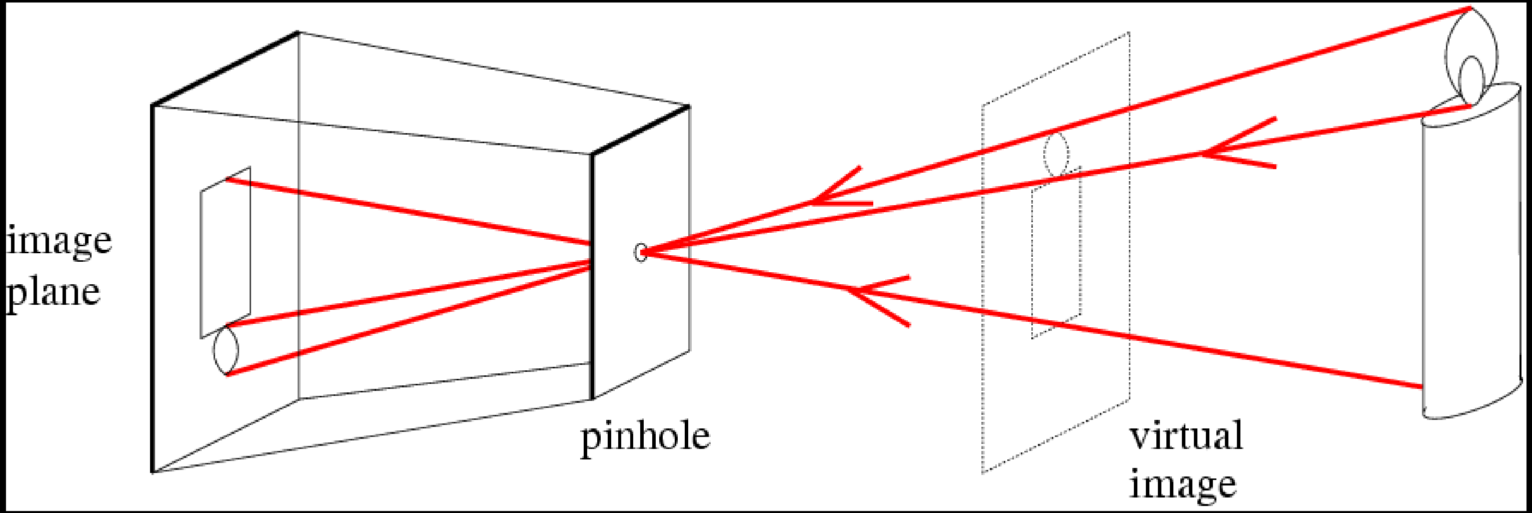




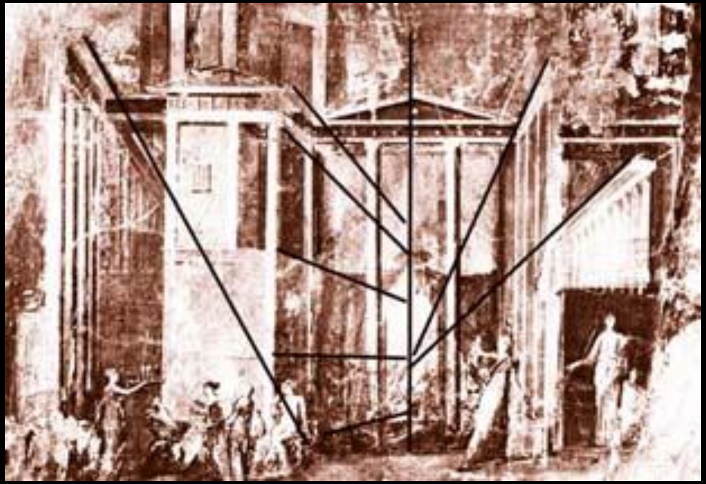
Animal eye: a looonnng time ago.

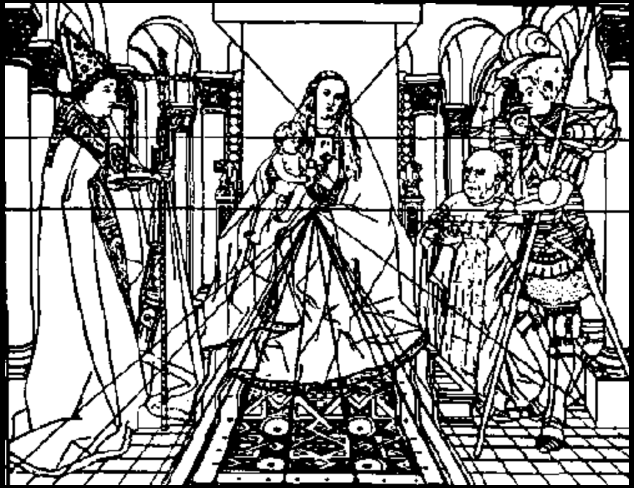


Photographic camera: Niepce, 1816.



Pinhole perspective projection: Brunelleschi, XVth Century. Camera obscura: XVIth Century.


Pompei painting, 2000 years ago

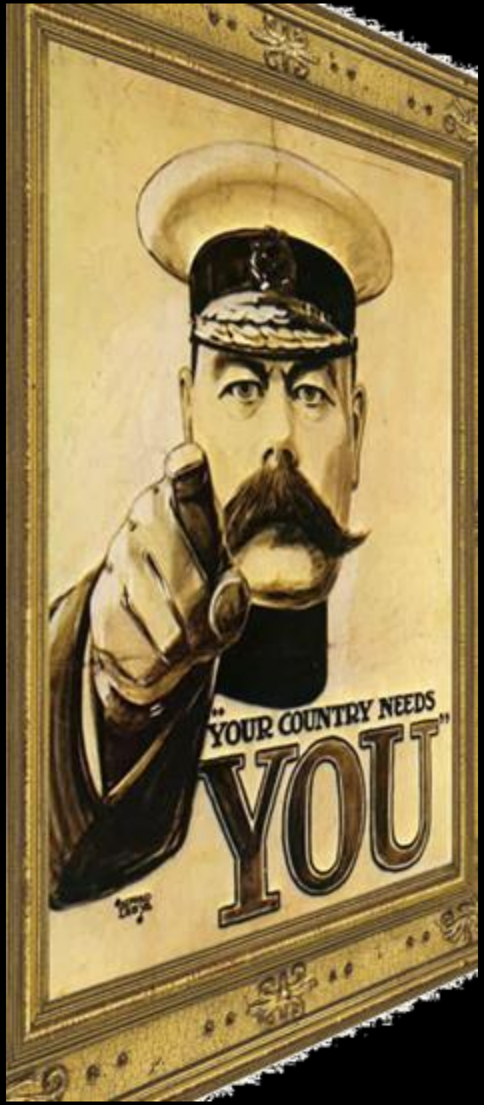

Van Eyk, XIV^th Century

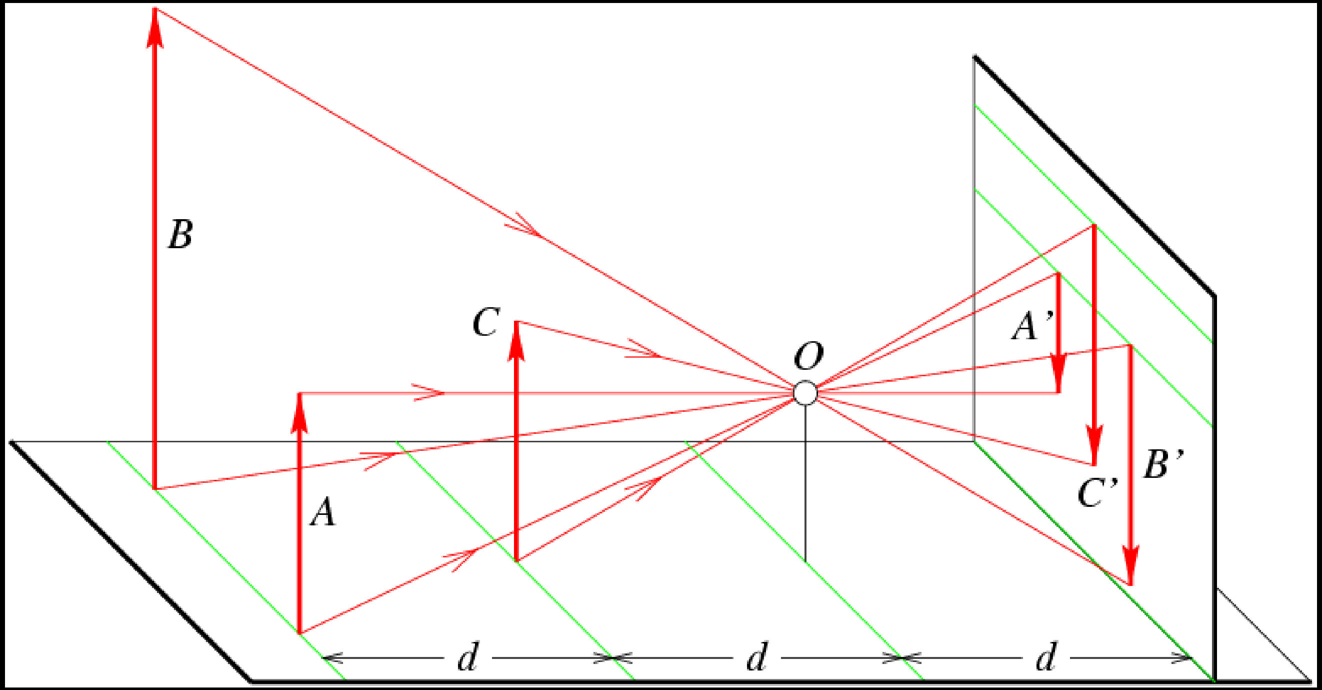
Brunelleschi, 1415

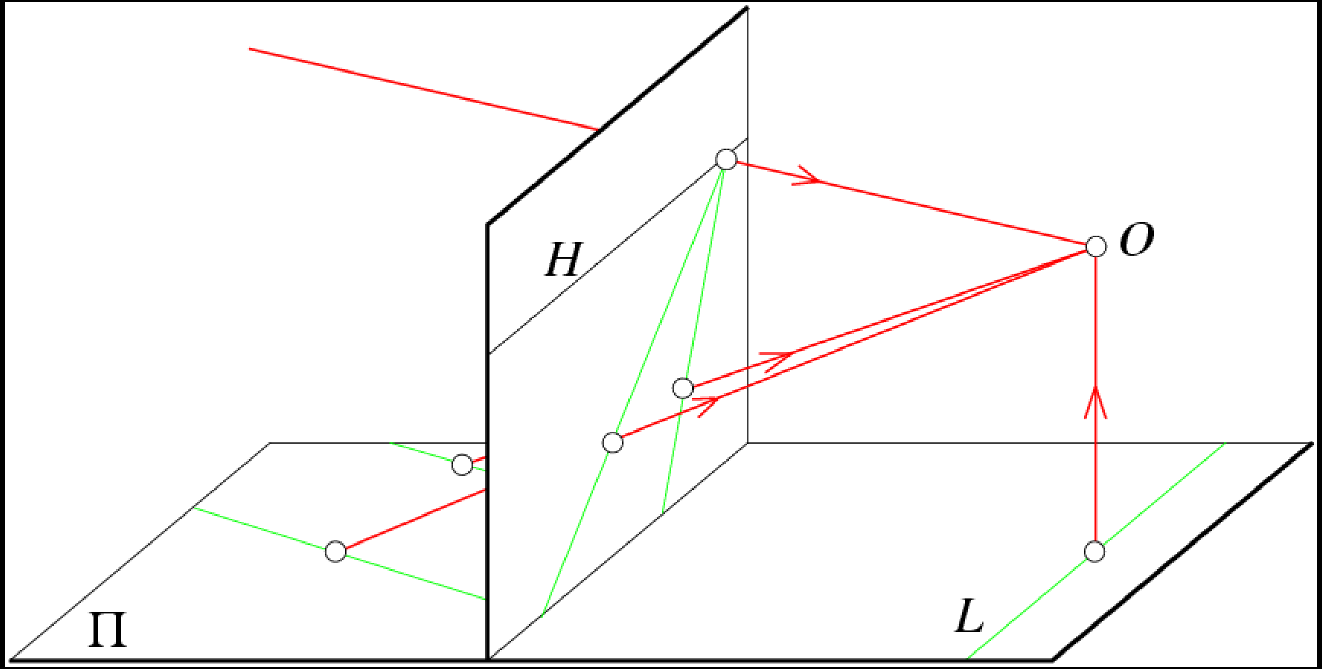
Massaccio’s Trinity, 1425

How do we see images?

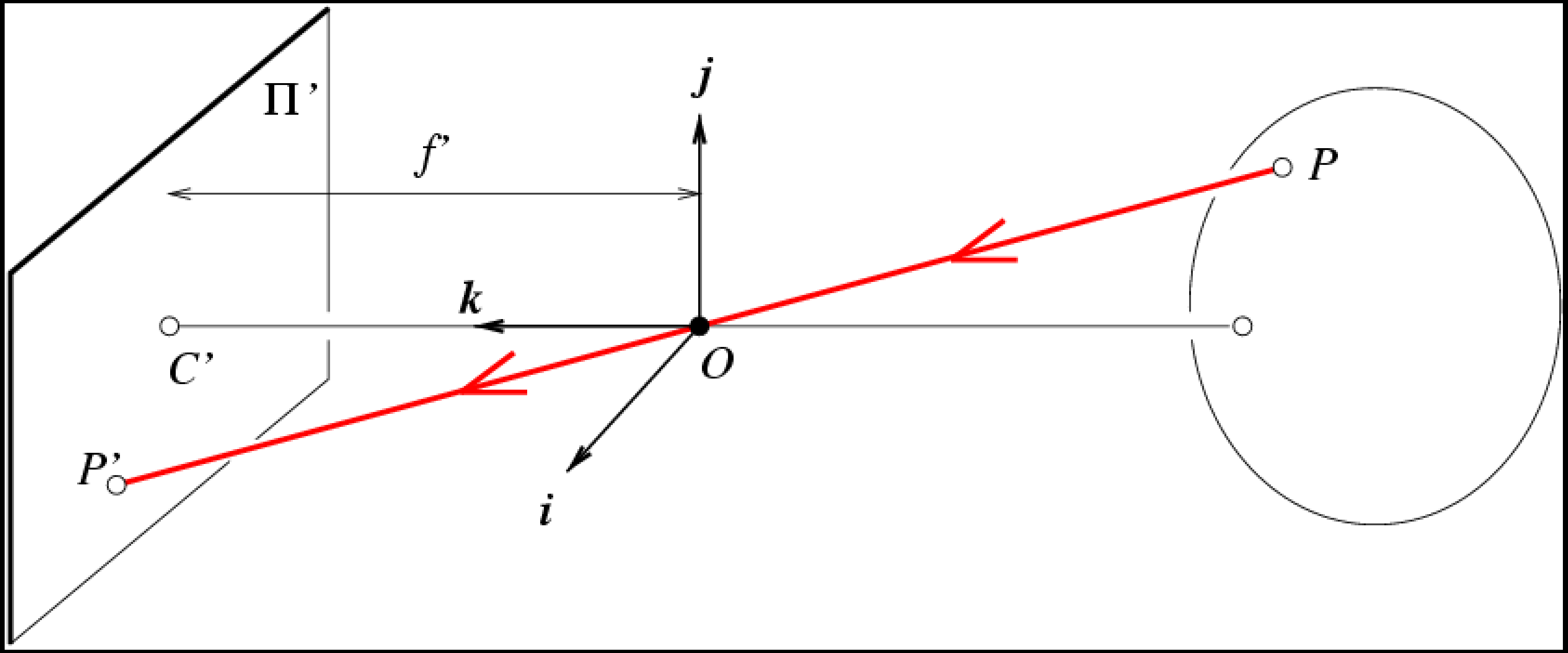
"YOUR COUNTRY NEEDS
YOU"




B
C
O
A
A'
C'
B'
d
d
d


H
O
Π
L

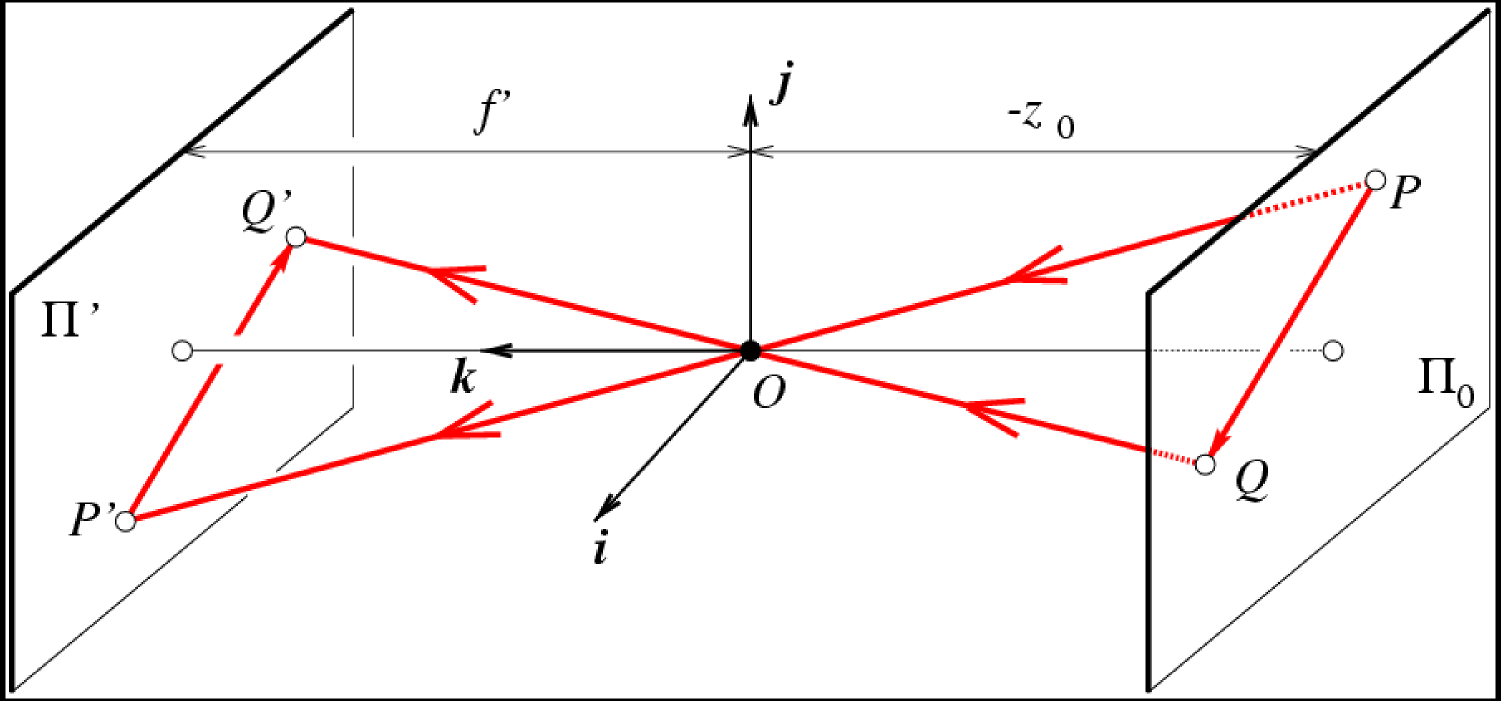
Pinhole Perspective Equation



$$\begin{cases} x' = f'\dfrac{x}{z} \\ y' = f'\dfrac{y}{z} \end{cases}$$

NOTE: *z* is always negative..

Affine projection models: Weak perspective projection

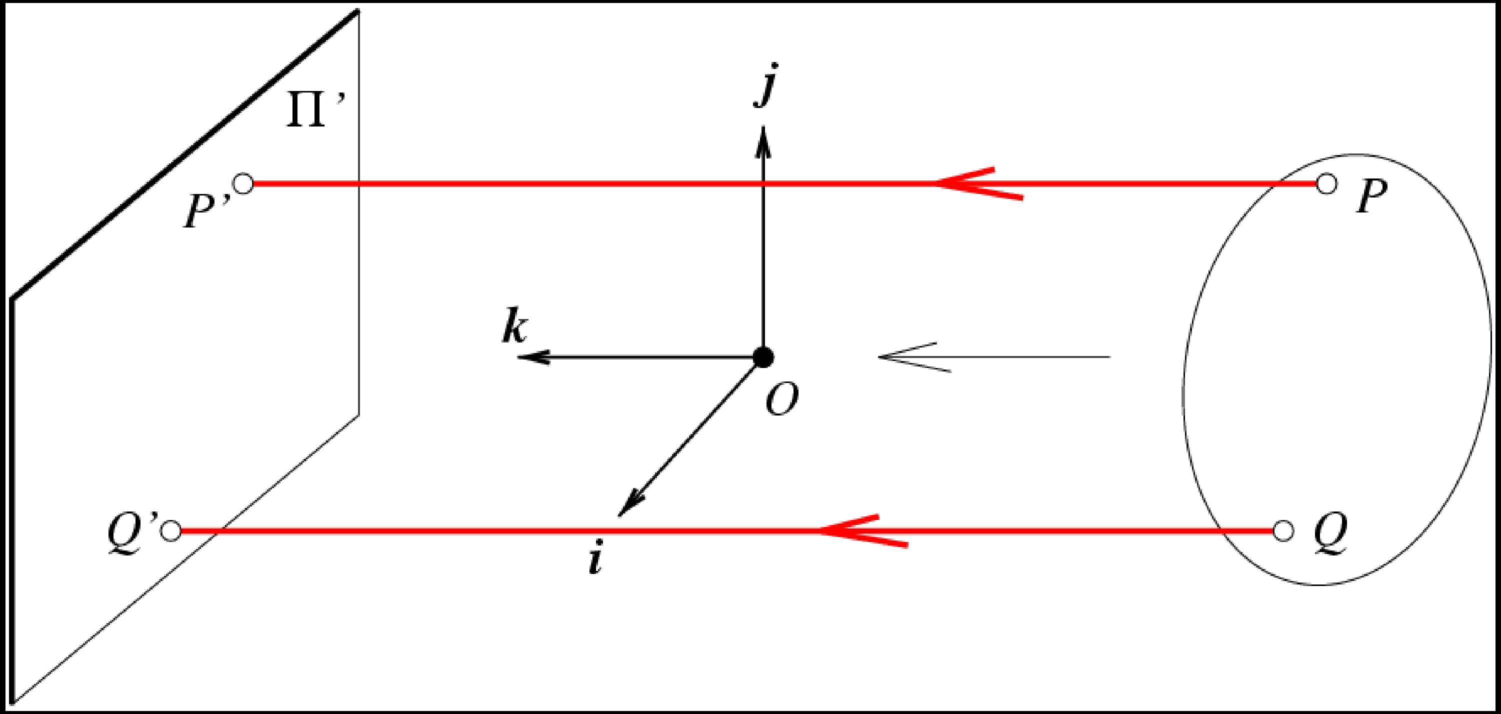


$$\boxed{\begin{cases} x' = mx \\ y' = my \end{cases}} \quad \text{where} \quad m = \frac{f'}{z_0}$$

is the (negative) magnification.

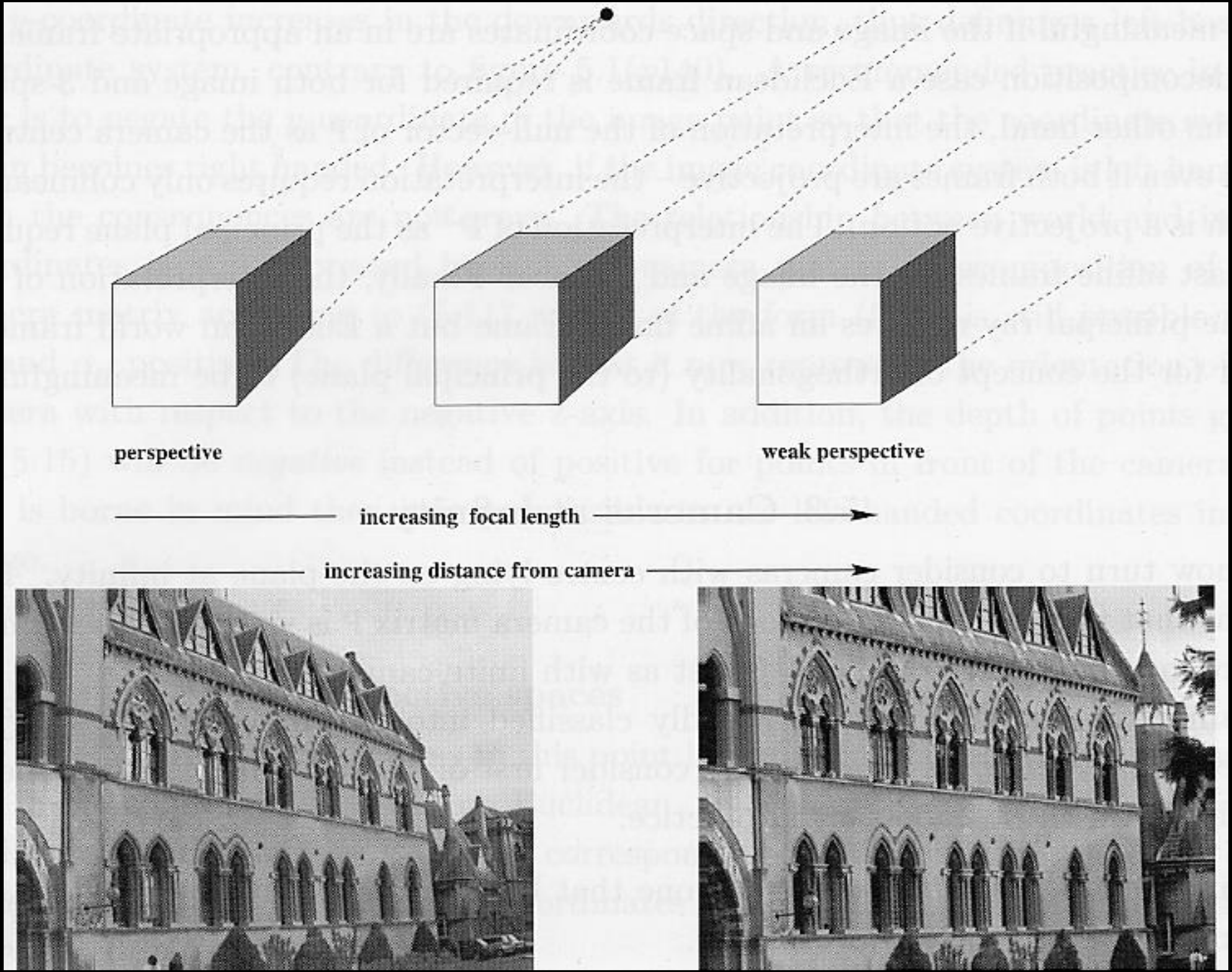
When the scene relief is small compared its distance from the Camera, *m* can be taken constant: weak perspective projection.

Affine projection models: Orthographic projection

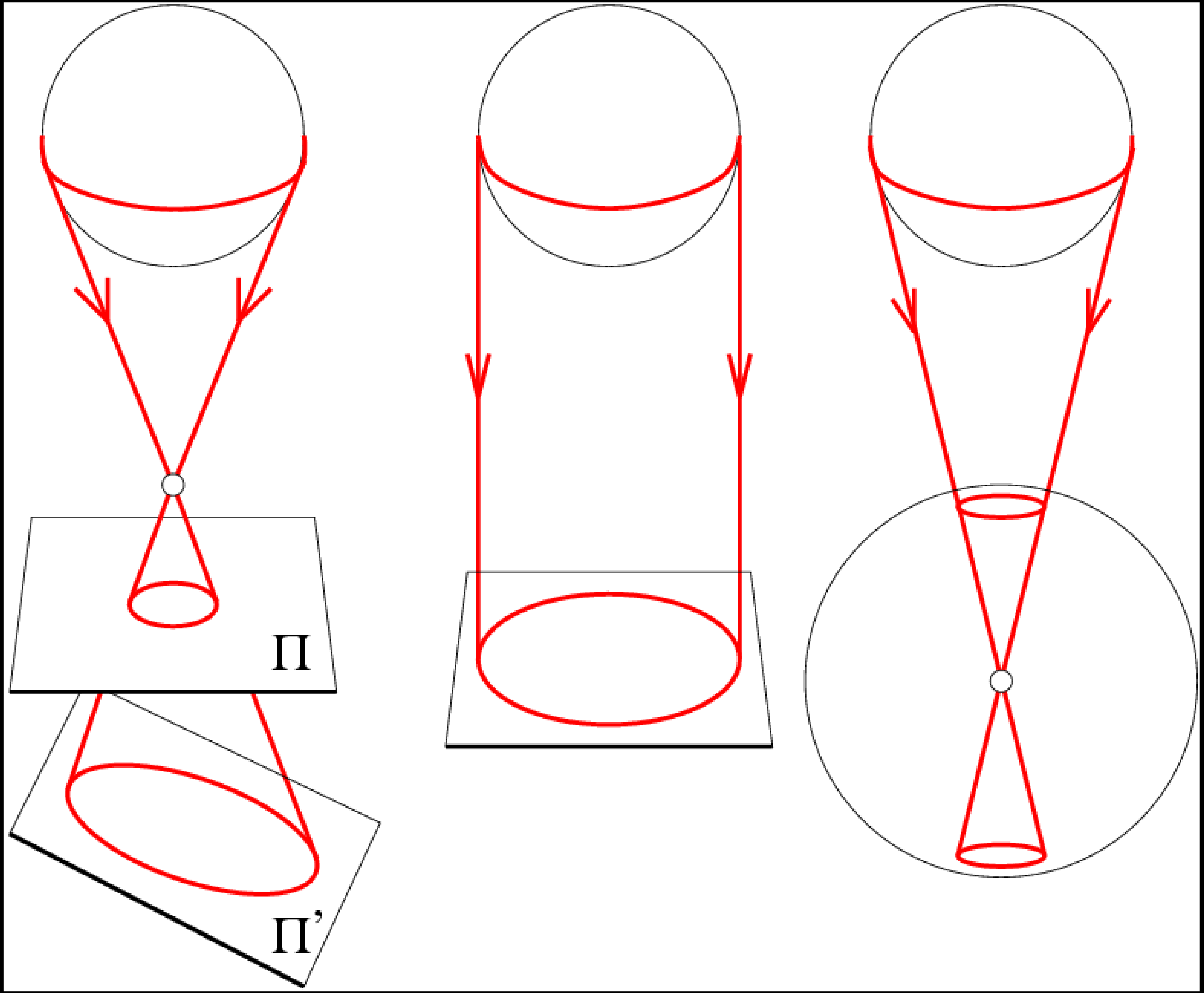


$$\begin{cases} x' = x \\ y' = y \end{cases}$$

When the camera is at a (roughly constant) distance from the scene, take $m = -1$



From Zisserman & Hartley

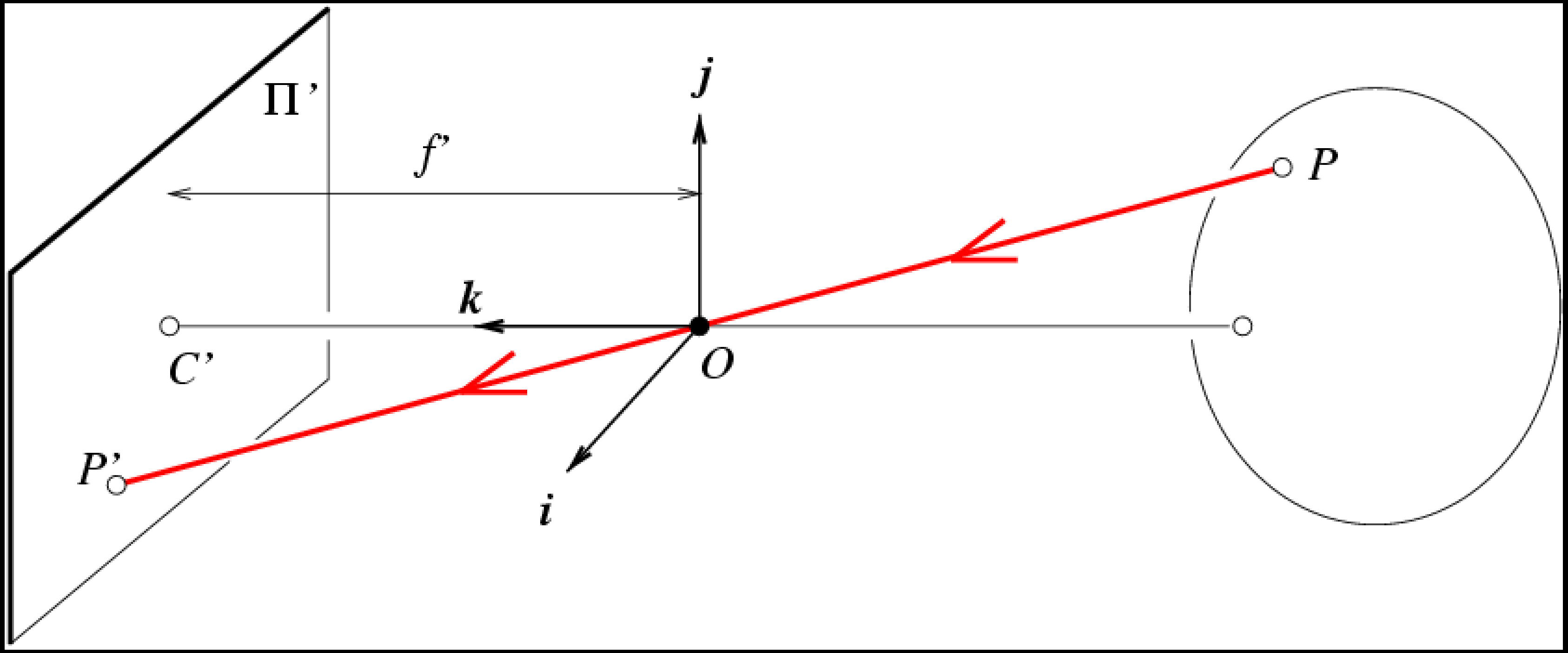


Planar pinhole perspective

Orthographic projection

Spherical pinhole perspective

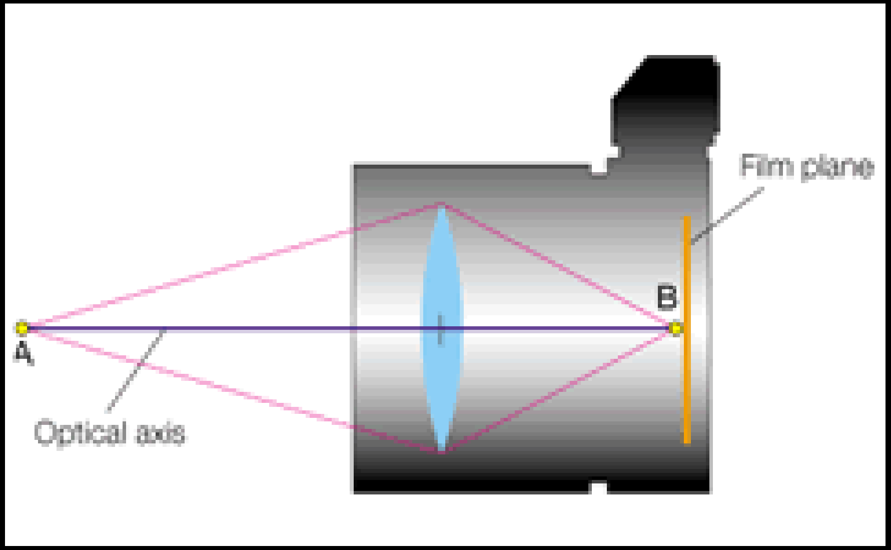
Pinhole Perspective Equation



$$\begin{cases} x' = f' \dfrac{x}{z} \\ y' = f' \dfrac{y}{z} \end{cases}$$

NOTE: *z* is always negative..

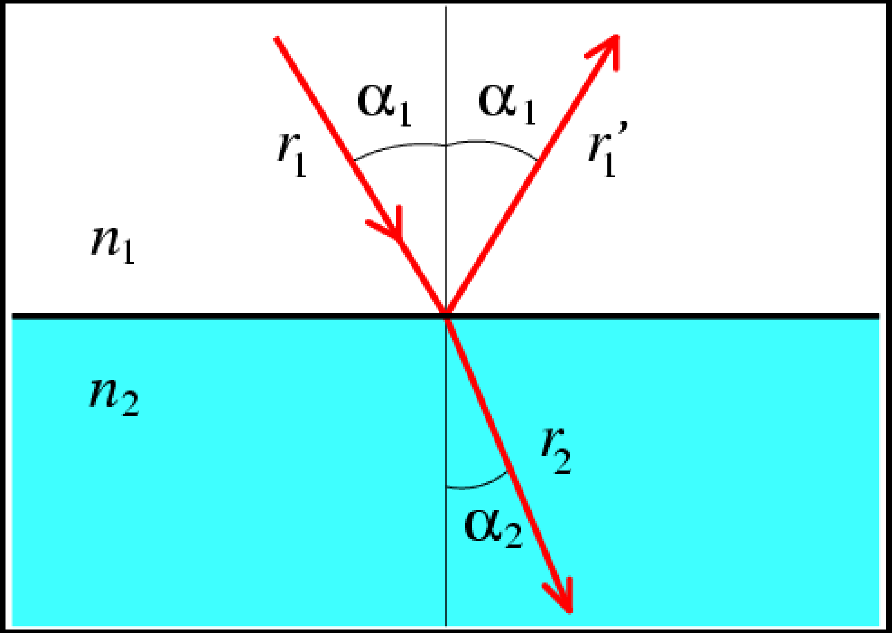
Lenses



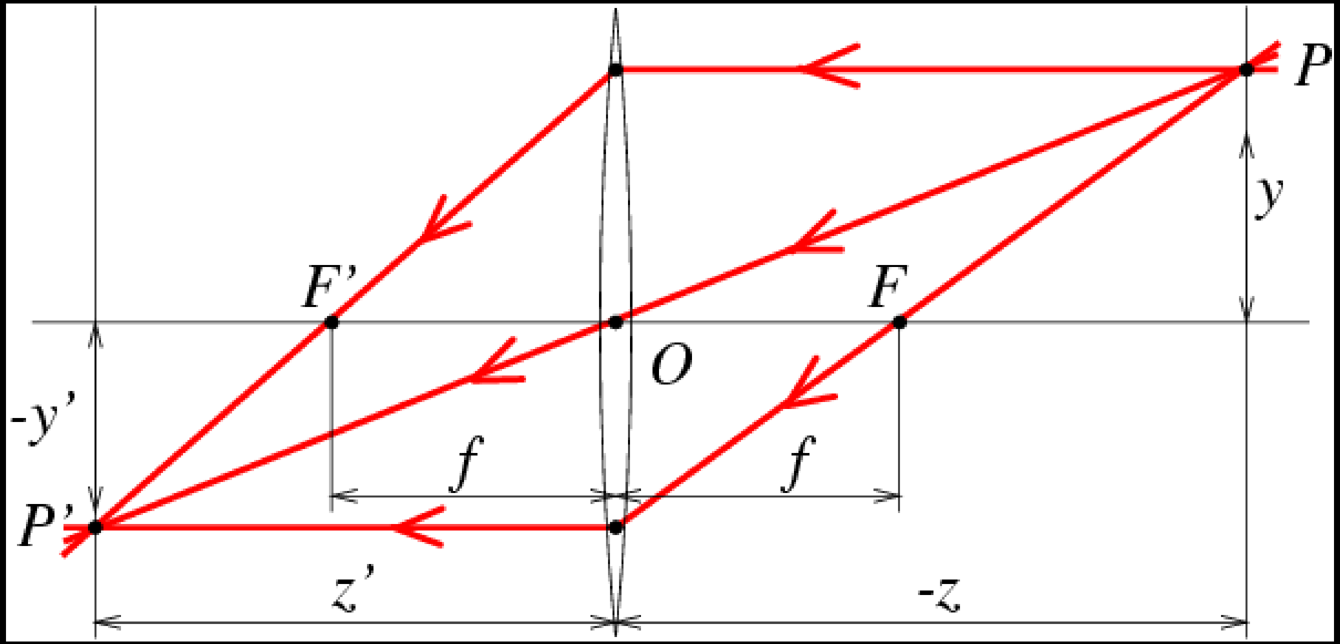
Snell's law

$$n_1 \sin \alpha_1 = n_2 \sin \alpha_2$$

(Descartes' law for Frenchies)

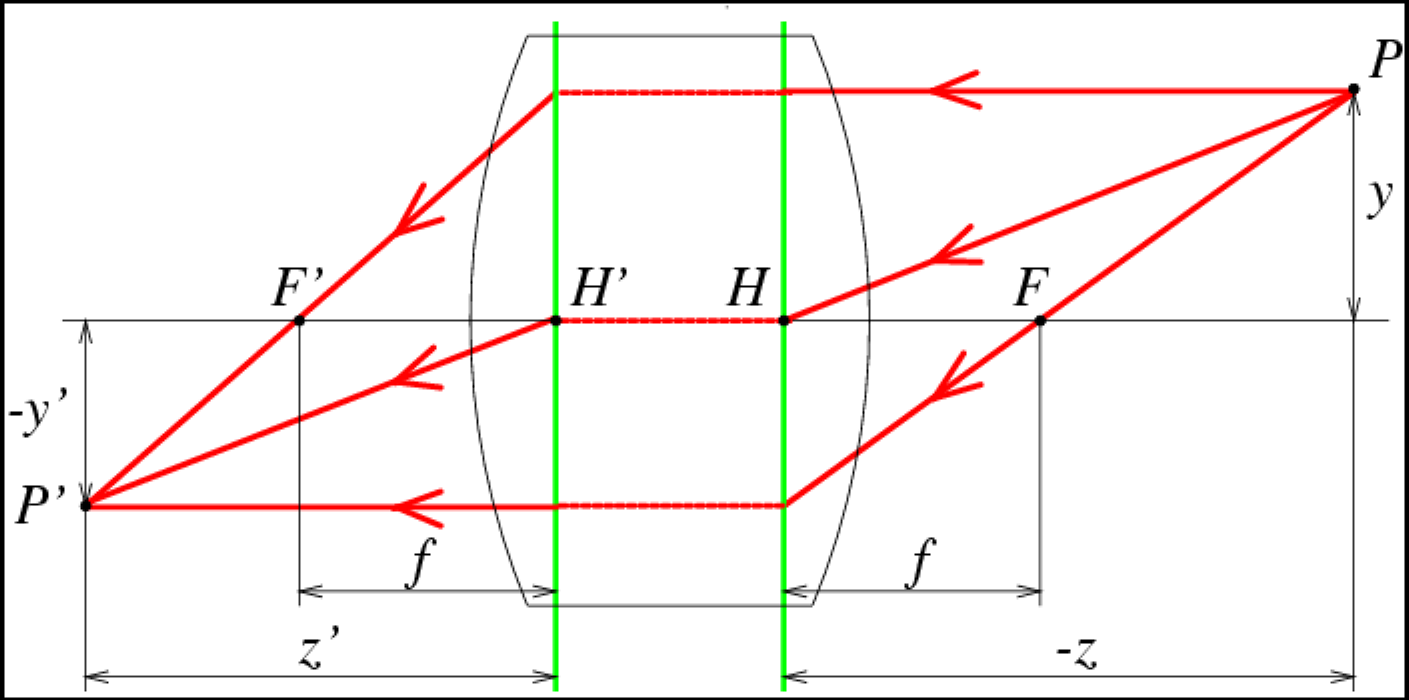


Thin Lenses (including paraxial approximation)

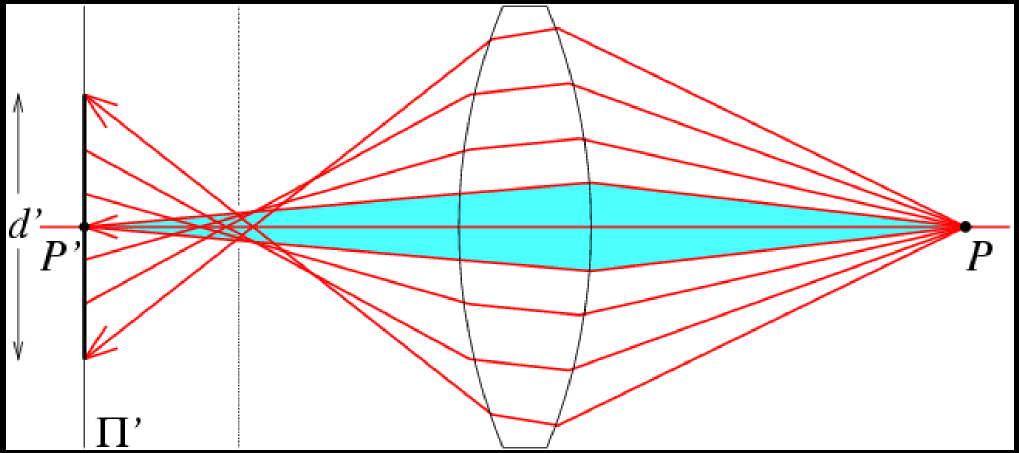


$$\boxed{\begin{cases} x' = z'\dfrac{x}{z} \\ y' = z'\dfrac{y}{z} \end{cases}}$$ where $$\boxed{\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}}$$ and $$\boxed{f = \frac{R}{2(n-1)}}$$

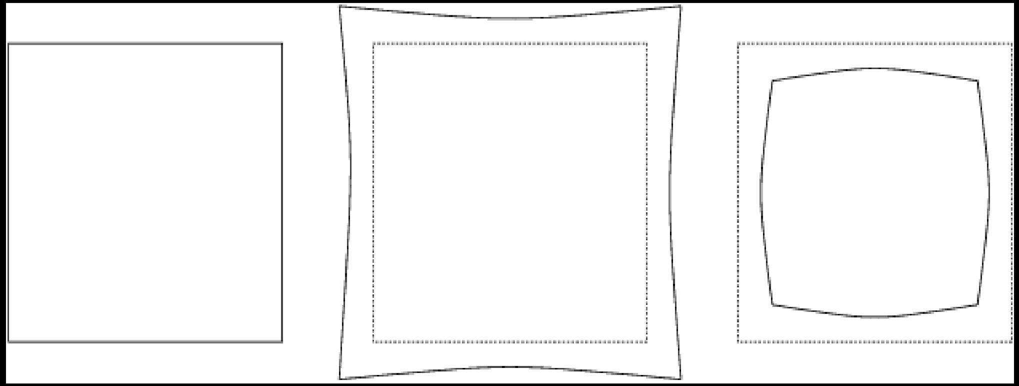
Thick Lenses



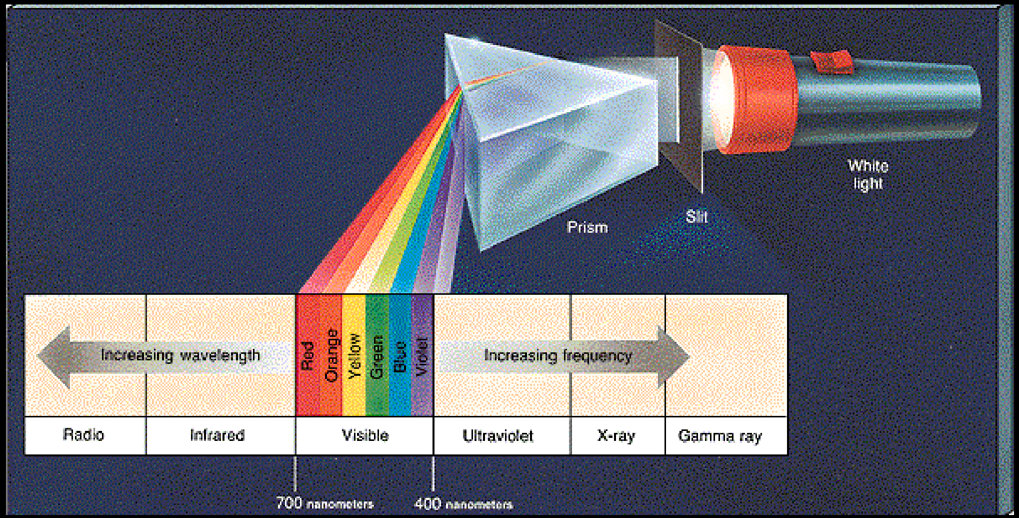
Spherical Aberration



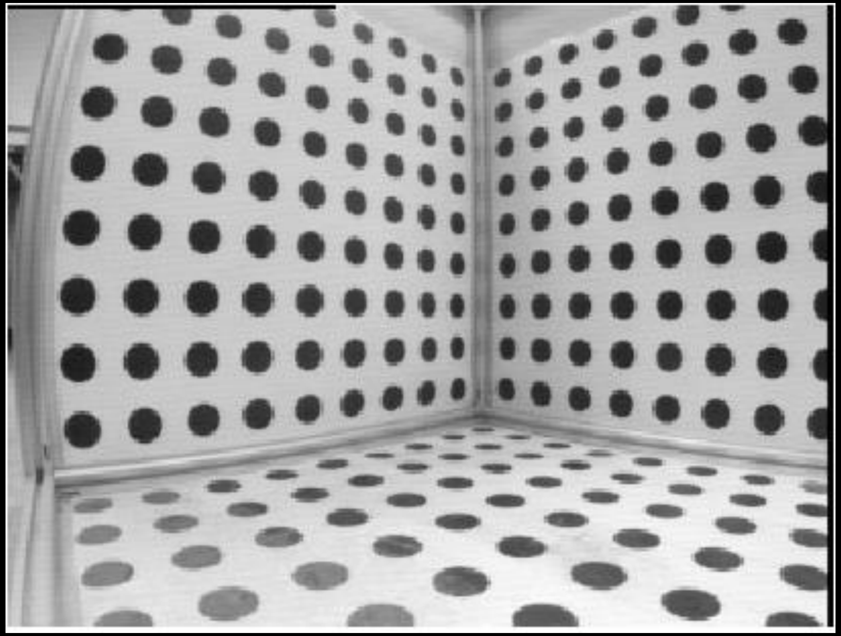
Distortion

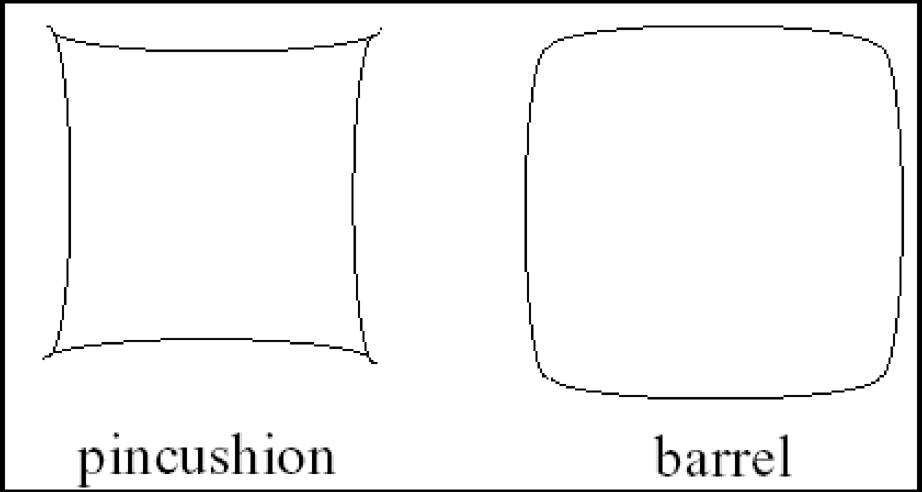


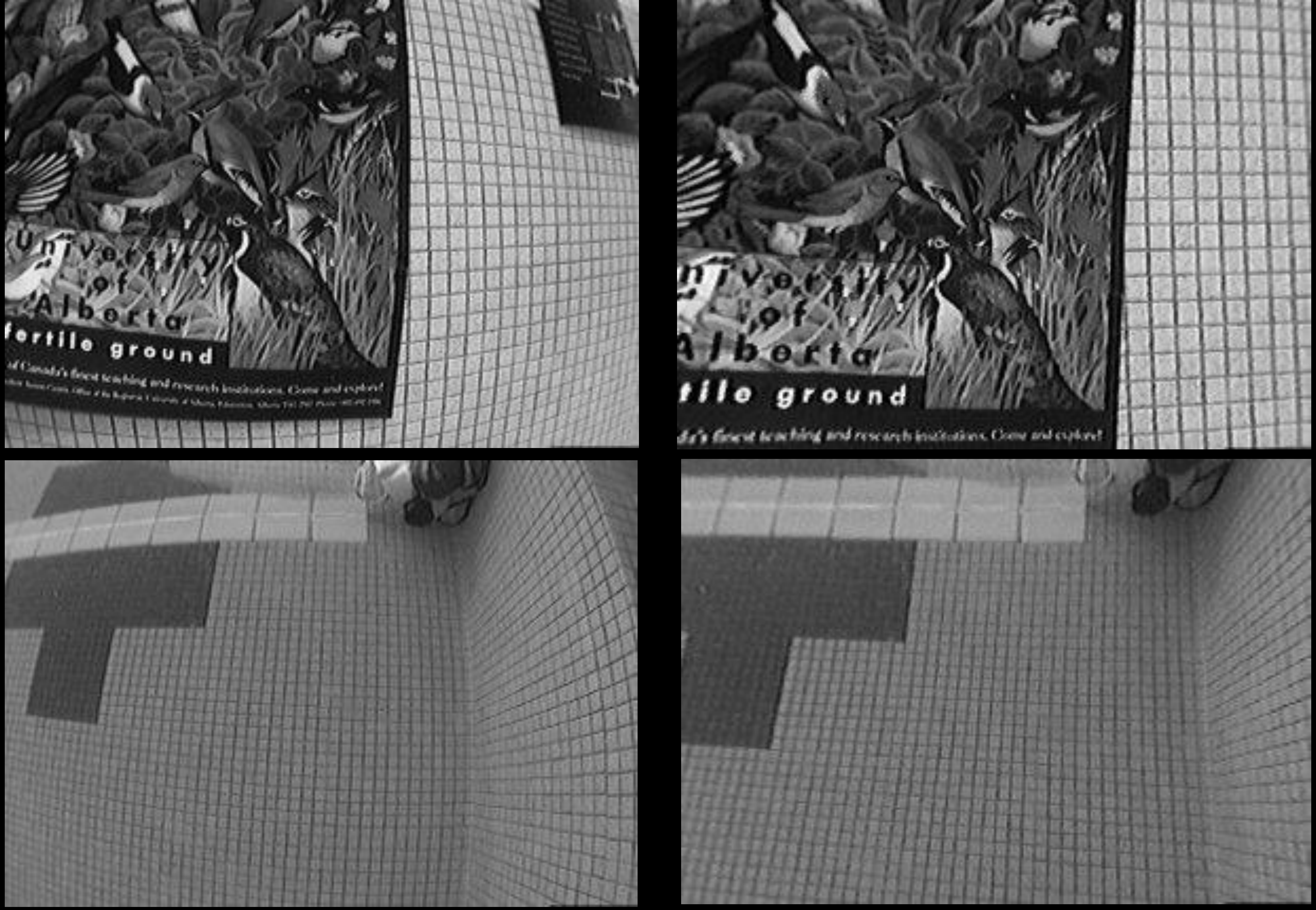
Chromatic Aberration



Geometric Distortion

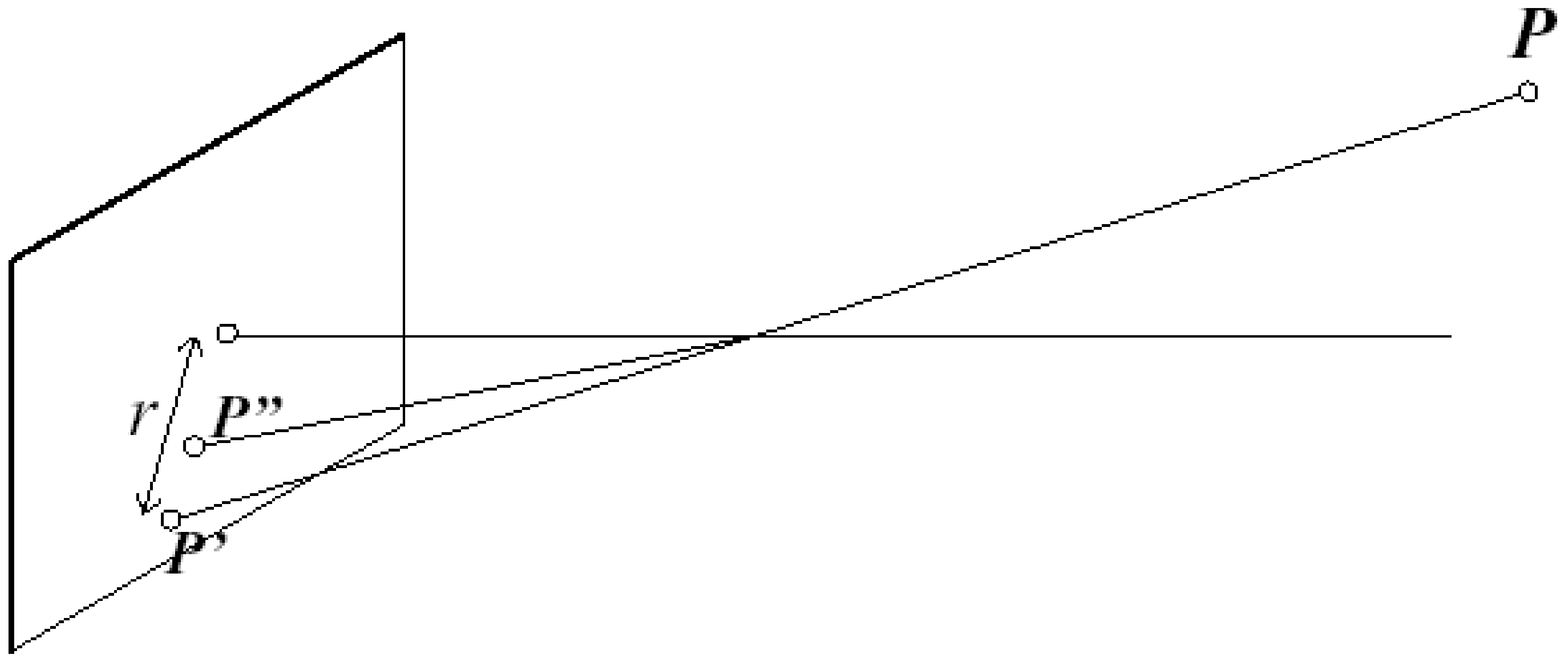





University of Alberta
fertile ground
of Canada's finest teaching and research institutions. Come and explore!

Rectification

Radial Distortion Model



Ideal:

$$x' = f\frac{x}{z}$$

$$y' = f\frac{y}{z}$$

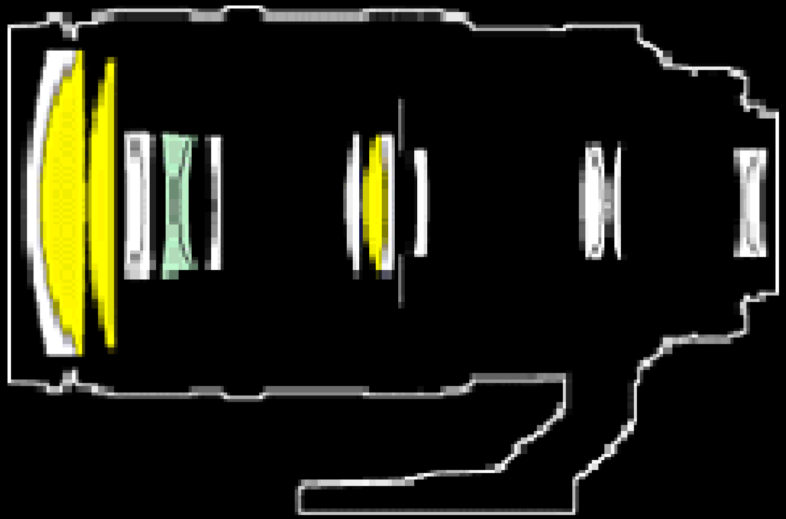
Distorted:

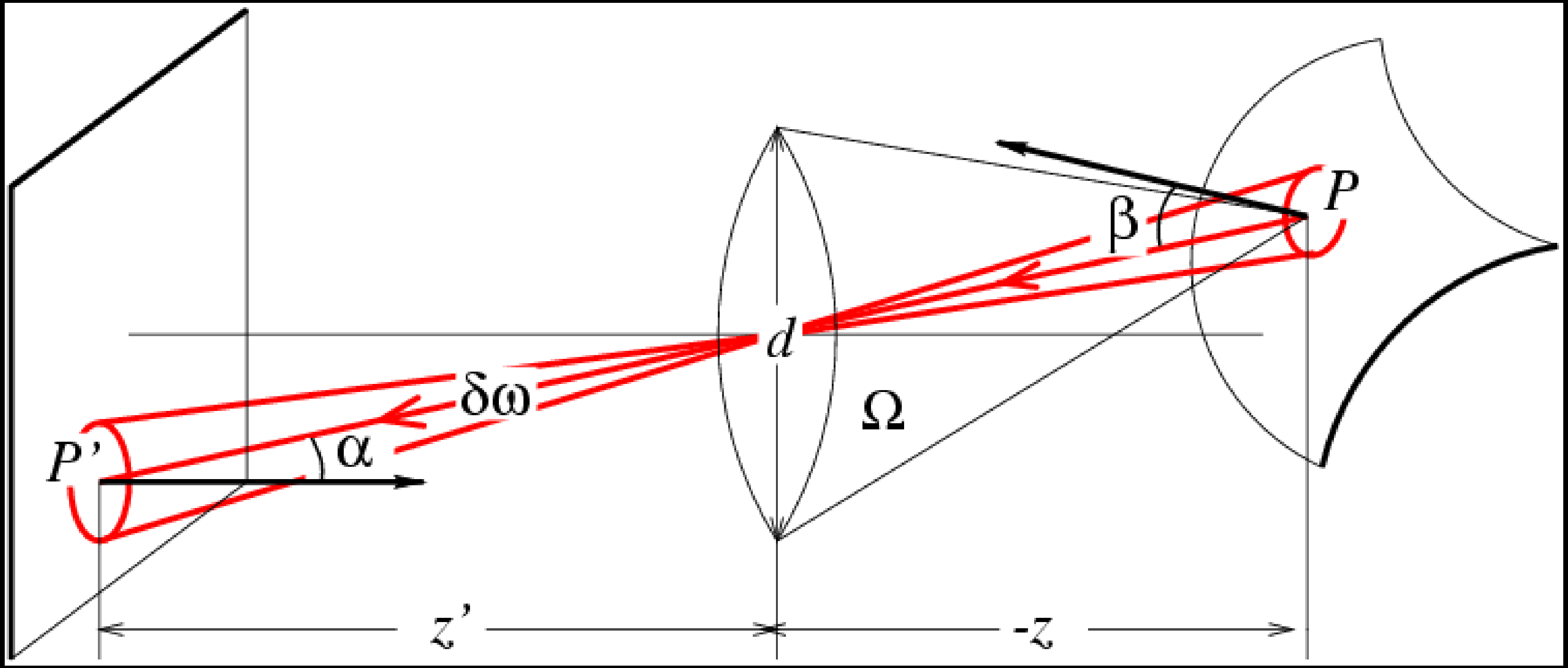
$$x'' = \frac{1}{\lambda}x'$$

$$y'' = \frac{1}{\lambda}y'$$

$$\lambda = 1 + k_1 r^2 + k_2 r^4 + \cdots$$

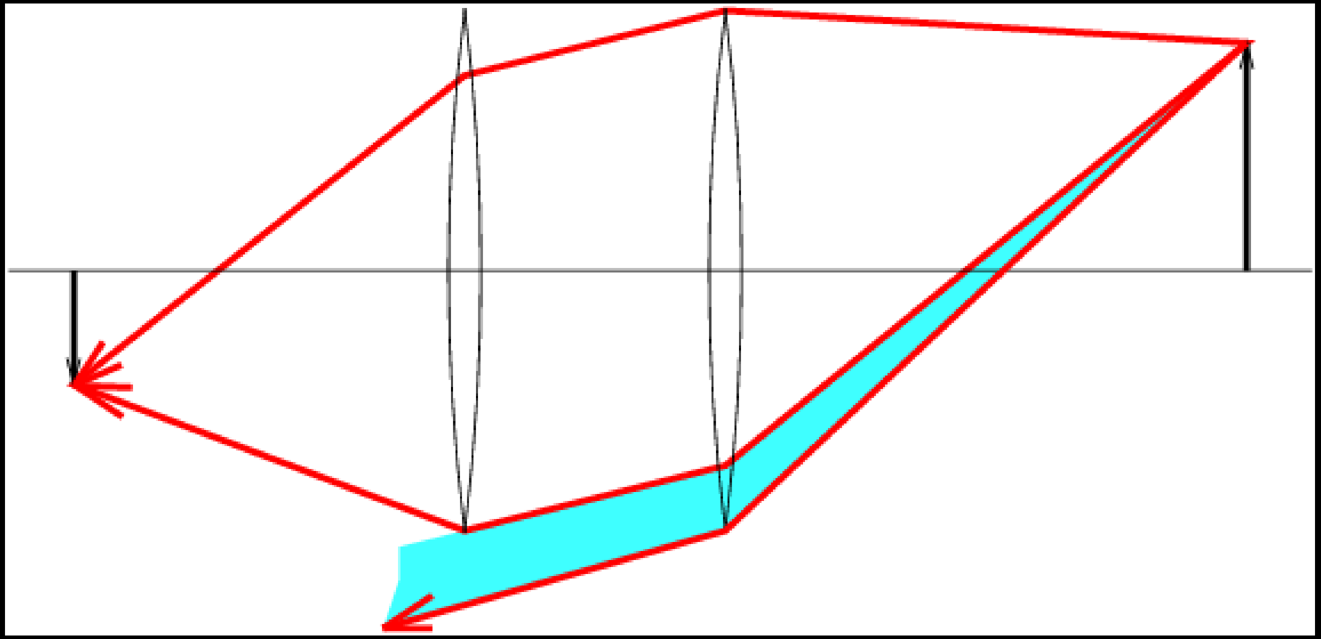
A compound lens



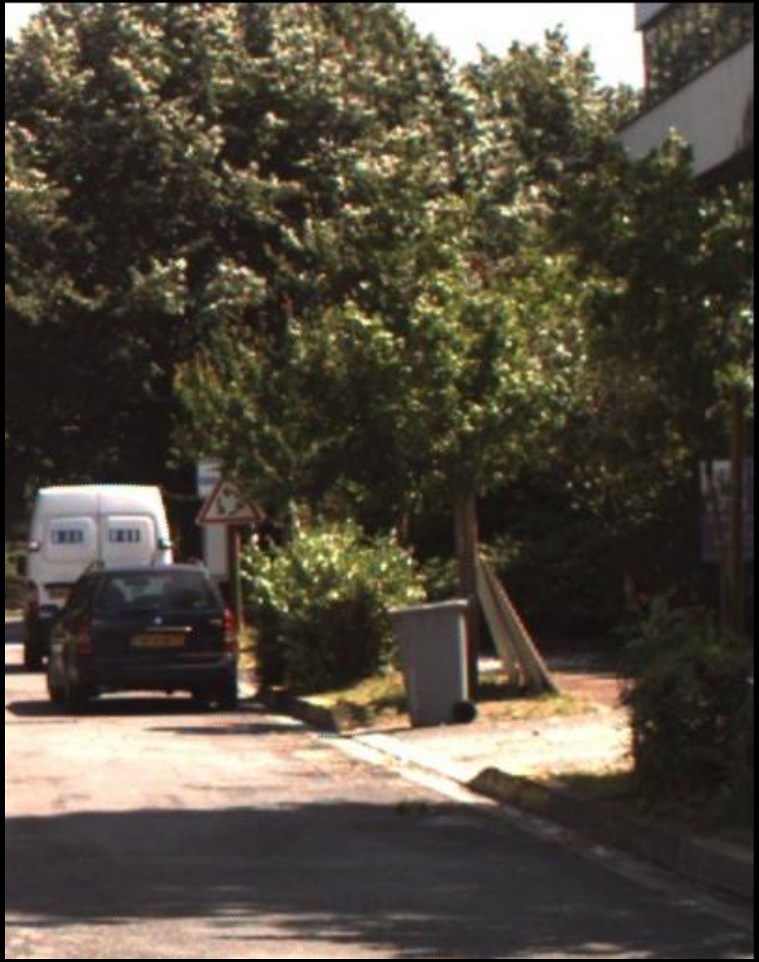


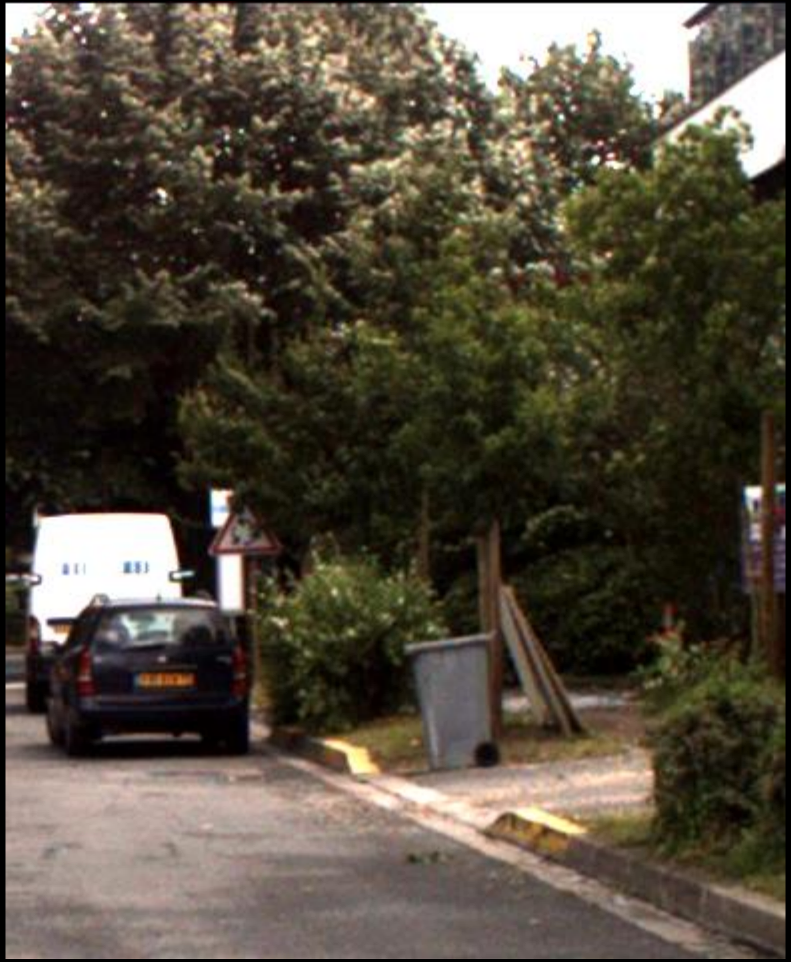
$$\boxed{E=(\Pi/4)\left[\,(d/z')^2\cos^4\alpha\,\right]L}$$

Vignetting



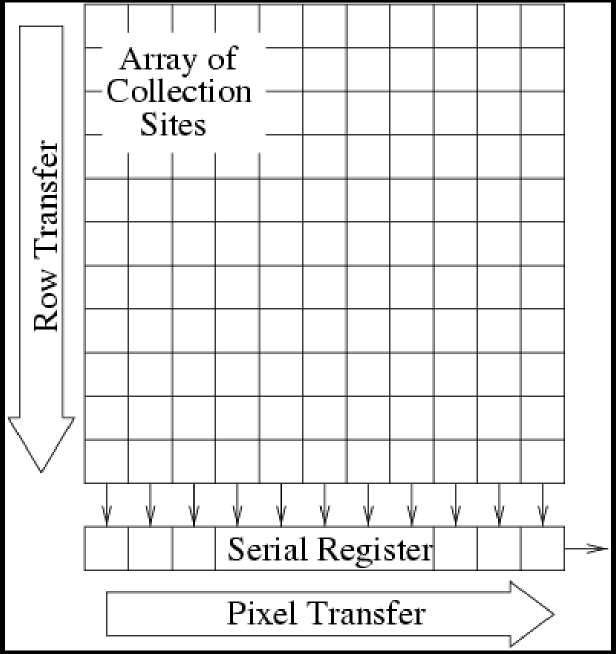
Challenge: Illumination – What is wrong with these pictures?





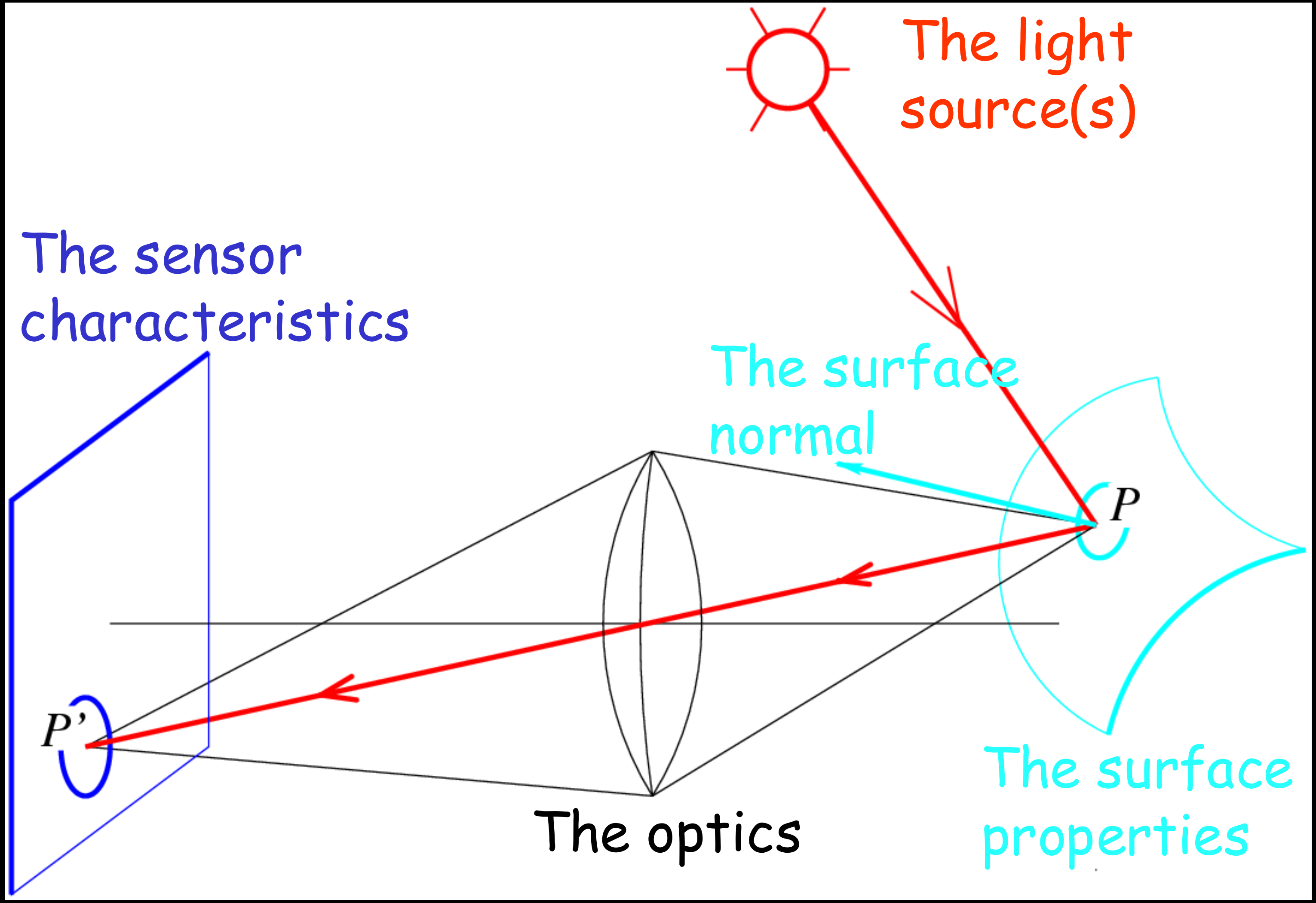
Photography
(Niepce, "La Table Servie," 1822)

Milestones:
- Daguerréotypes (1839)
- Photographic Film (Eastman, 1889)
- Cinema (Lumière brothers, 1895)
- Color Photography (Lumière brothers, again, 1908)
- Television (Baird, Farnsworth, Zworykin, 1920s)



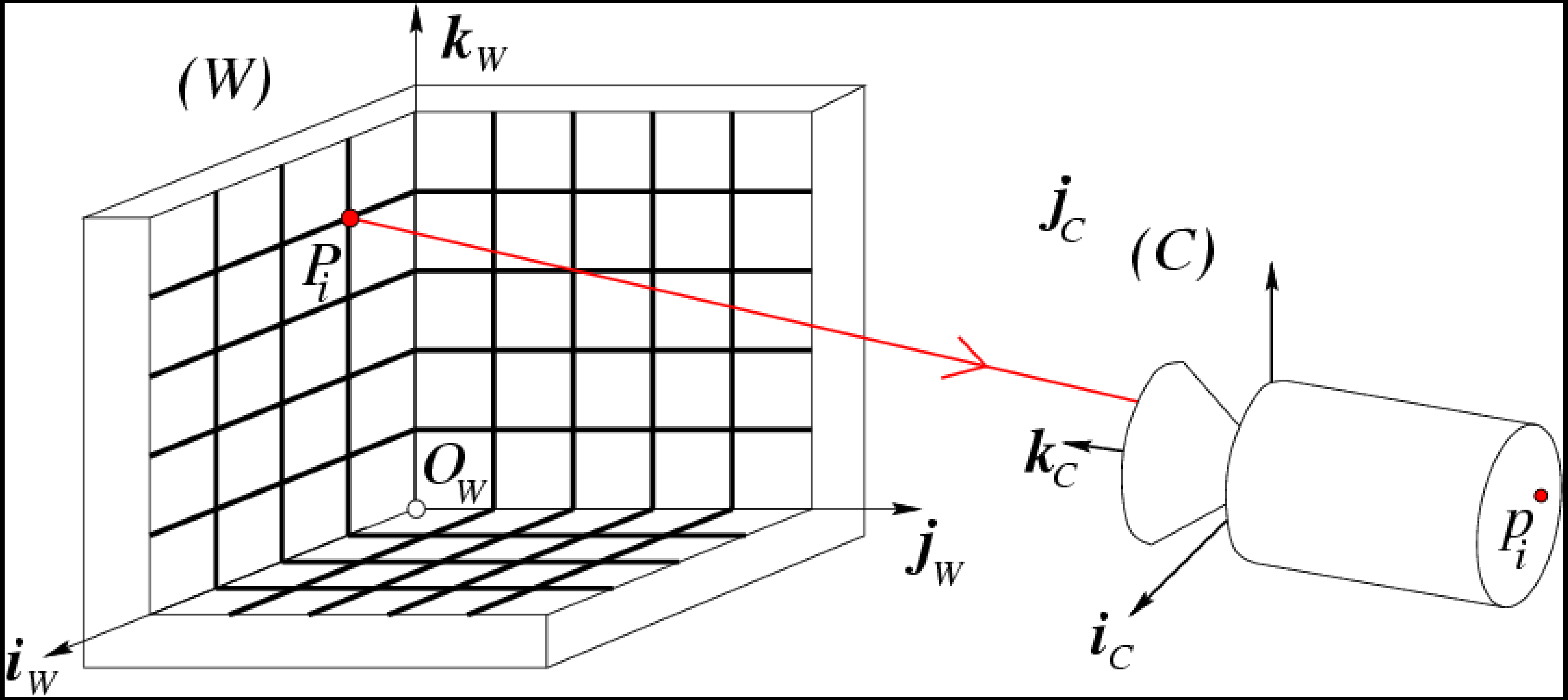
CCD Devices (1970), etc.

Image Formation: Radiometry



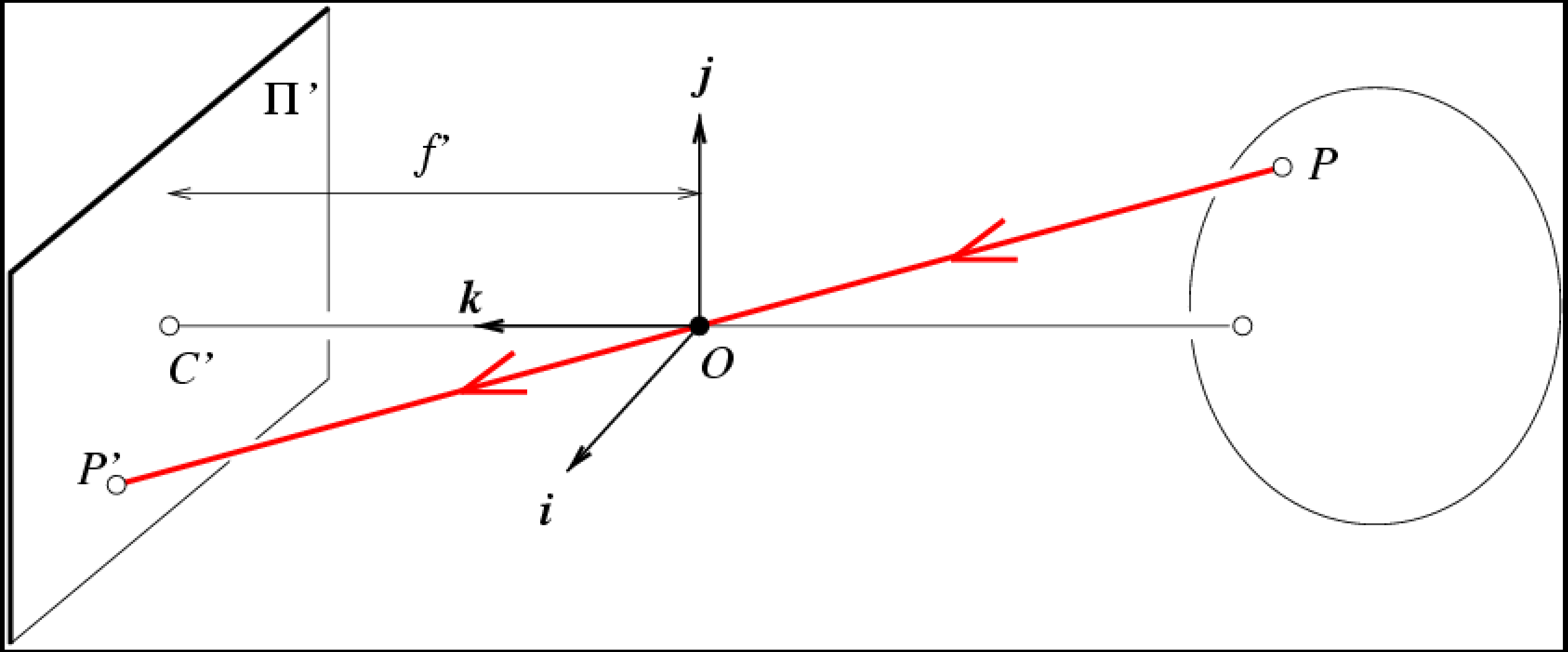
What determines the brightness of an image pixel?

Quantitative Measurements and Calibration



Euclidean Geometry

Pinhole Perspective Equation



$$\begin{cases} x' = f'\dfrac{x}{z} \\ y' = f'\dfrac{y}{z} \end{cases}$$

$$u = f\frac{x}{z}$$

$$v = f\frac{x}{z}$$

$$p = \frac{1}{z}\mathrm{P}$$

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Homogeneous coordinates

Conversion

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogeneous coordinates

Invariant to scaling

$$k\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

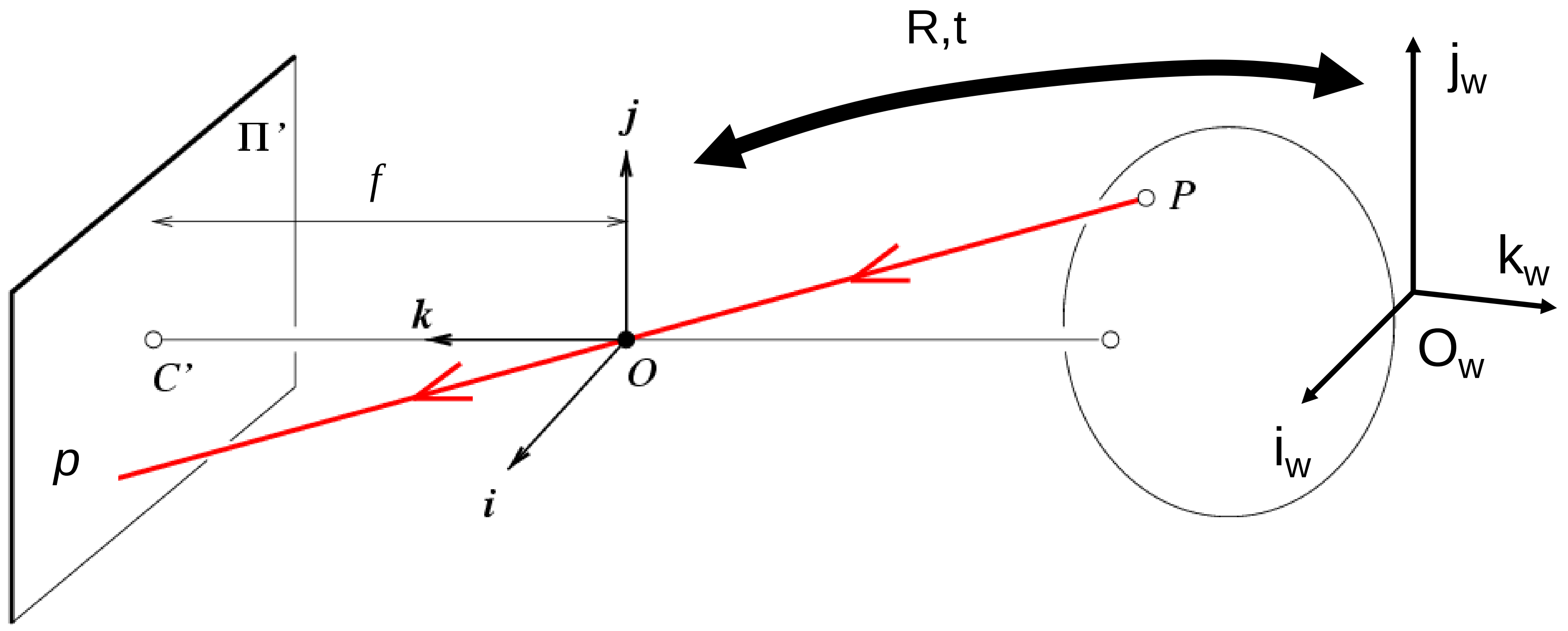
Homogeneous Coordinates

Cartesian Coordinates

A point in Cartesian coordinates is a ray in homogeneous ones

Slide Credit: Savarese

Projection matrix



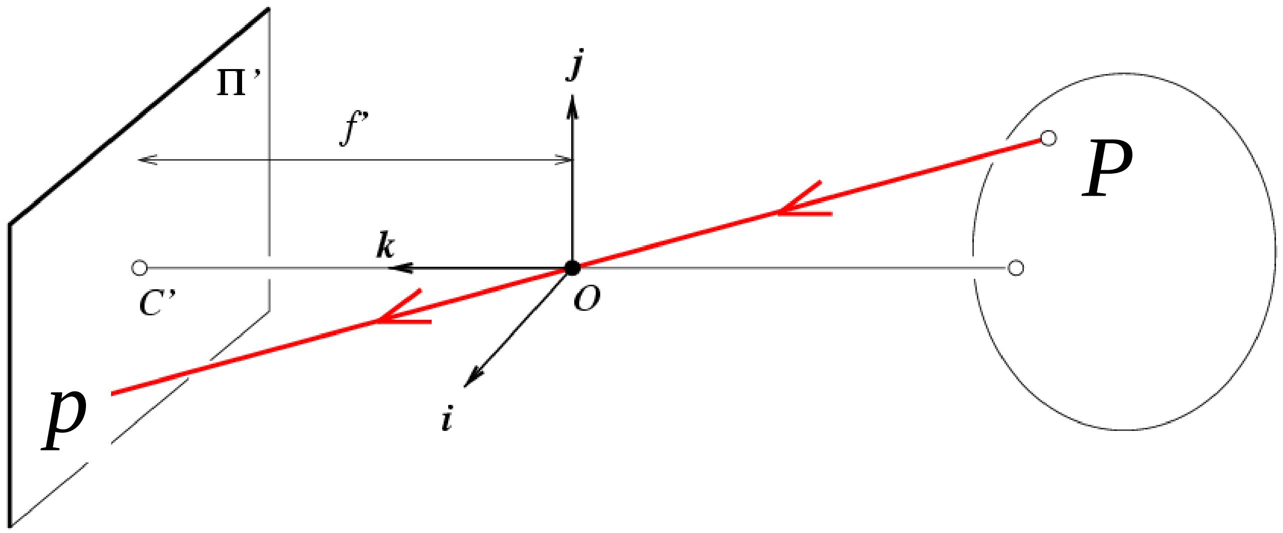
p: Image Coordinates: (u,v,1)
M: 3x4 projection matrix
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
P: World Coordinates: (x,y,z,1)

$$p \approx MP = K\,[R\ t]P$$

$\Updownarrow$

$$p = \lambda\, MP \text{ for some } \lambda \neq 0$$

Projection matrix



Intrinsic Assumptions

- Unit aspect ratio
- Image center at (0,0)
- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$p \approx K\,[I\;0]P \Rightarrow w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \underbrace{\begin{matrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{matrix}}_{K} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(Note: here $w = z$)

Slide Credit: Savarese

Remove assumption: known image center

Intrinsic Assumptions

- Unit aspect ratio
- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$p \approx K\,[I\ 0]P \quad \Rightarrow \quad w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Remove assumption: square pixels

Intrinsic Assumptions

- No skew

Extrinsic Assumptions

- No rotation
- Camera at (0,0,0)

$$p \approx K\ [I\ 0]P \quad \Rightarrow \quad w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Remove assumption: rectangular pixels

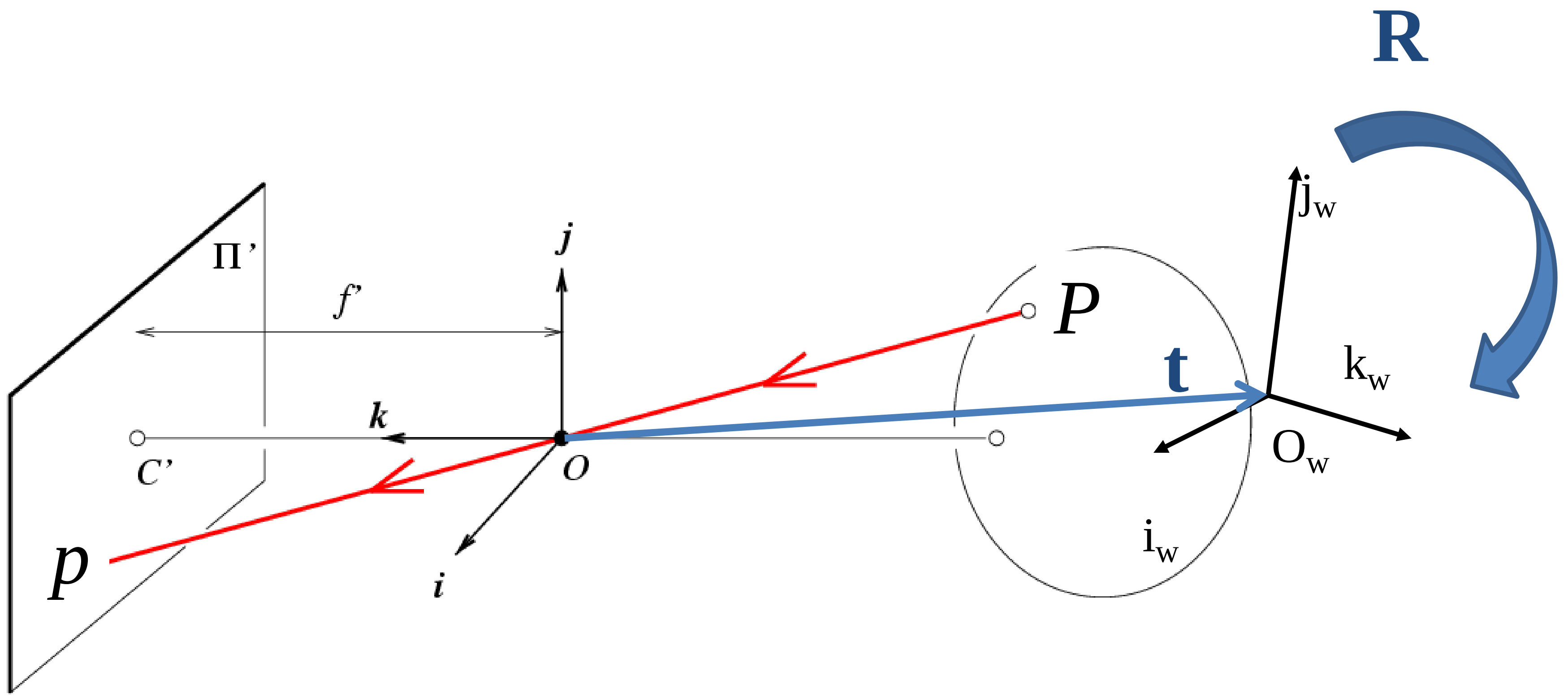
Intrinsic Assumptions

Extrinsic Assumptions
- No rotation
- Camera at (0,0,0)

$$p \approx K\,[I\;0]P \;\Rightarrow\; w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Note: different books use different notation for parameters

Oriented and Translated Camera



Allow camera translation

Intrinsic Assumptions

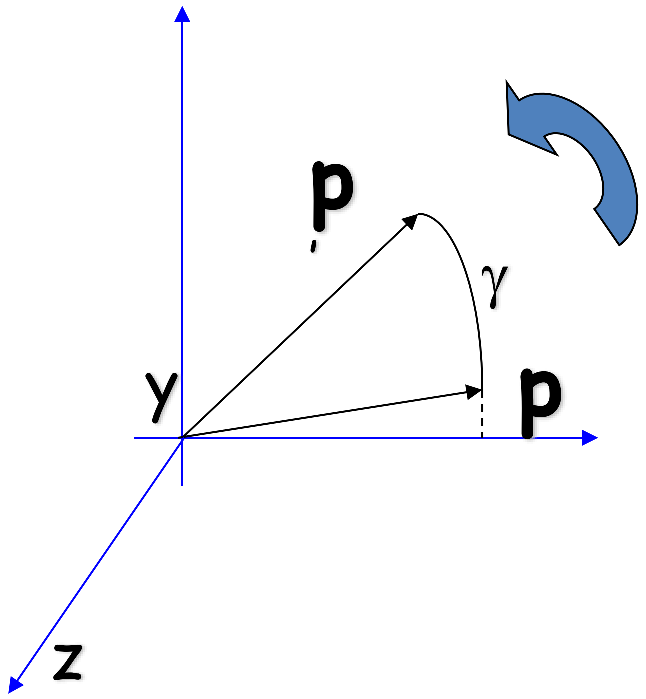
Extrinsic Assumptions
- No rotation

$$p \approx K\,[I\ t]P \Rightarrow w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slide Credit: Saverese

3D Rotation of Points

Rotation around the coordinate axes, counter-clockwise:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Allow camera rotation

$$p \approx K\,[R\ t]P$$

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Degrees of freedom

$$p \approx K\,[R\ t]P$$

$$\Downarrow$$

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overset{5}{\begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{6}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$p \approx MP$ or $p = \frac{1}{z} MP$

$M = K\,[R\ t]$

$p = K\hat{p}$ and $\hat{p} = \frac{1}{z}\widehat{M}P$

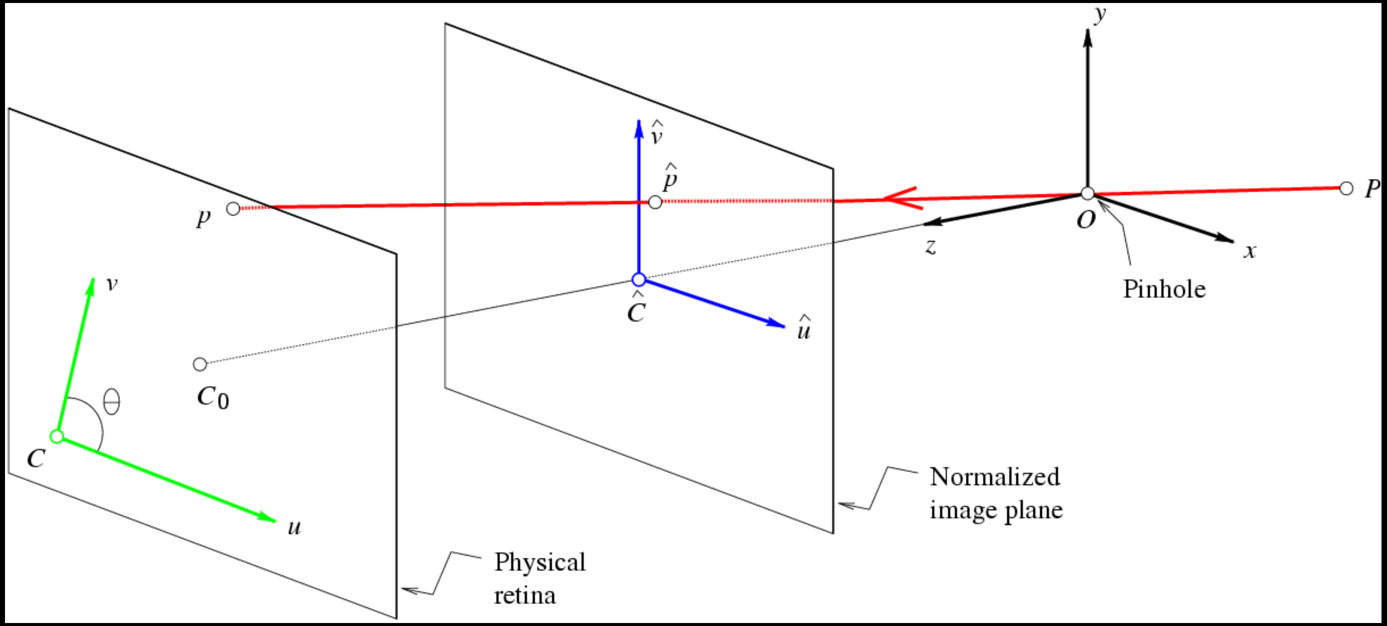
$\widehat{M} = [R\ t]$

normalized coordinates

Note: z is *not* independent of $\mathcal{M}$ and $\boldsymbol{P}$:

$$\mathcal{M} = \begin{pmatrix} \boldsymbol{m}_1^T \\ \boldsymbol{m}_2^T \\ \boldsymbol{m}_3^T \end{pmatrix} \Longrightarrow z = \boldsymbol{m}_3 \cdot \boldsymbol{P}, \quad \text{or} \quad \begin{cases} u = \dfrac{\boldsymbol{m}_1 \cdot \boldsymbol{P}}{\boldsymbol{m}_3 \cdot \boldsymbol{P}}, \\ v = \dfrac{\boldsymbol{m}_2 \cdot \boldsymbol{P}}{\boldsymbol{m}_3 \cdot \boldsymbol{P}}. \end{cases}$$

Explicit form of the projection matrix



$$\mathcal{M} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix}$$

Explicit Form of the Projection Matrix

$$\mathcal{M} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix}$$

Note: If $\mathcal{M} = (\,\mathcal{A} \quad \boldsymbol{b}\,)$ then $|\boldsymbol{a}_3| = 1$.

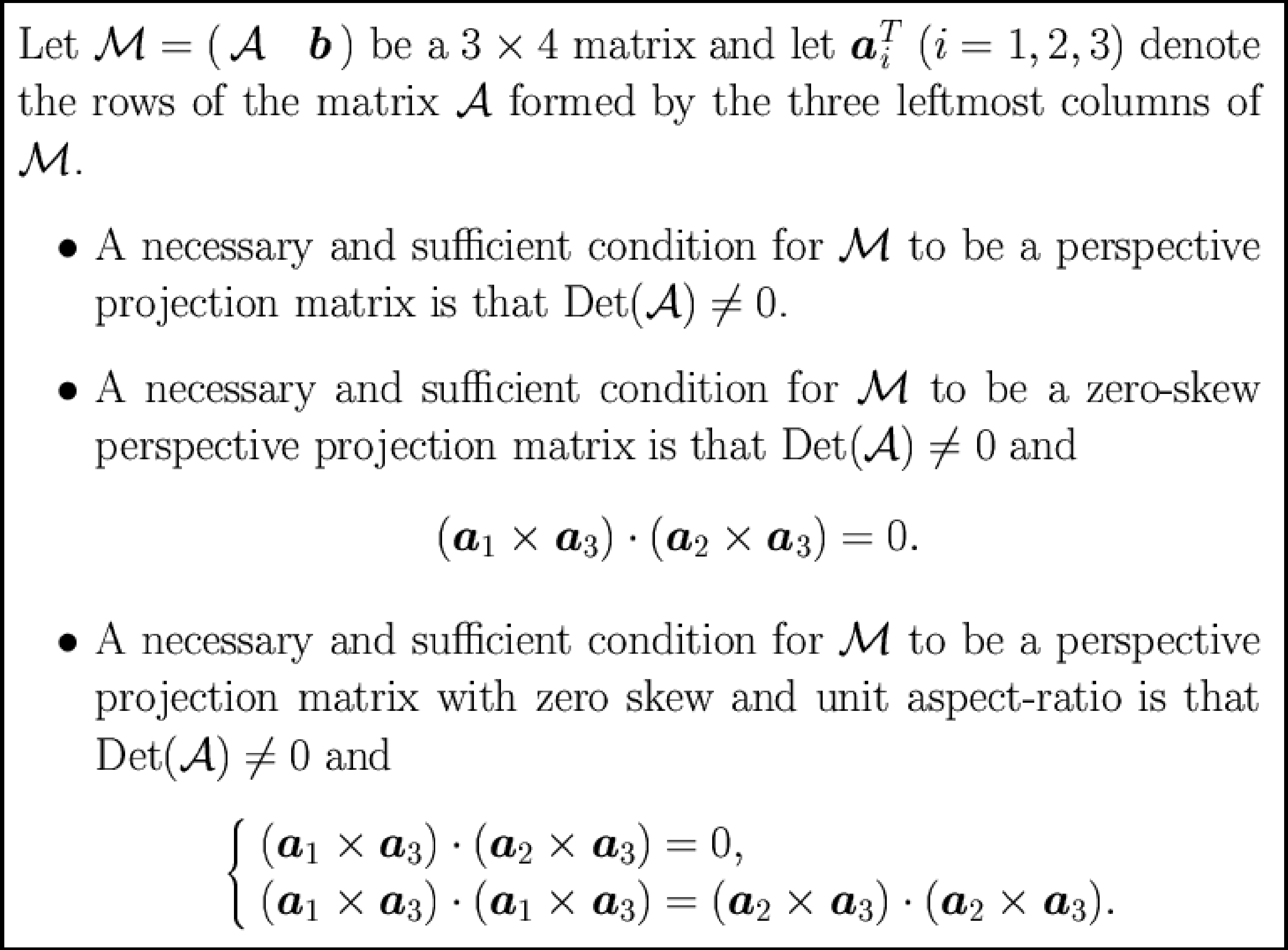
Replacing $\mathcal{M}$ by $\lambda\mathcal{M}$ in

$$\begin{cases} u = \dfrac{\boldsymbol{m}_1 \cdot \boldsymbol{P}}{\boldsymbol{m}_3 \cdot \boldsymbol{P}} \\ v = \dfrac{\boldsymbol{m}_2 \cdot \boldsymbol{P}}{\boldsymbol{m}_3 \cdot \boldsymbol{P}} \end{cases}$$

does not change u and v.

M is only defined up to scale in this setting!!

Theorem (Faugeras, 1993)



Let $\mathcal{M} = (\mathcal{A} \quad \boldsymbol{b})$ be a 3×4 matrix and let $\boldsymbol{a}_i^T$ $(i = 1, 2, 3)$ denote the rows of the matrix $\mathcal{A}$ formed by the three leftmost columns of $\mathcal{M}$.

- A necessary and sufficient condition for $\mathcal{M}$ to be a perspective projection matrix is that $\text{Det}(\mathcal{A}) \neq 0$.
- A necessary and sufficient condition for $\mathcal{M}$ to be a zero-skew perspective projection matrix is that $\text{Det}(\mathcal{A}) \neq 0$ and

$$(\boldsymbol{a}_1 \times \boldsymbol{a}_3) \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3) = 0.$$

- A necessary and sufficient condition for $\mathcal{M}$ to be a perspective projection matrix with zero skew and unit aspect-ratio is that $\text{Det}(\mathcal{A}) \neq 0$ and

$$\begin{cases} (\boldsymbol{a}_1 \times \boldsymbol{a}_3) \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3) = 0, \\ (\boldsymbol{a}_1 \times \boldsymbol{a}_3) \cdot (\boldsymbol{a}_1 \times \boldsymbol{a}_3) = (\boldsymbol{a}_2 \times \boldsymbol{a}_3) \cdot (\boldsymbol{a}_2 \times \boldsymbol{a}_3). \end{cases}$$

Linear Camera Calibration

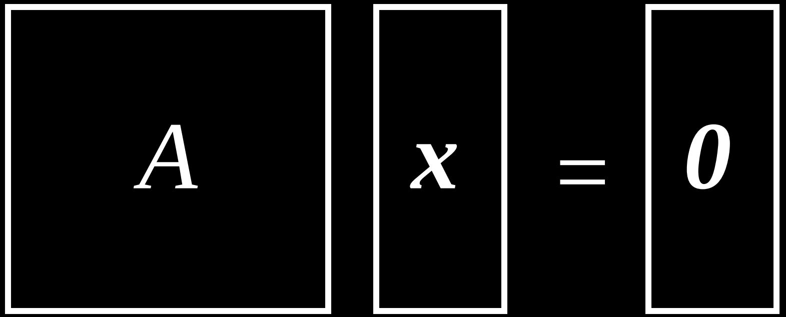
Given n points $P_1, \ldots, P_n$ with *known* positions and their images $p_1, \ldots, p_n$

Remember: $a \cdot b = a^T b$

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \dfrac{\boldsymbol{m}_1 \cdot \boldsymbol{P}_i}{\boldsymbol{m}_3 \cdot \boldsymbol{P}_i} \\[2ex] \dfrac{\boldsymbol{m}_2 \cdot \boldsymbol{P}_i}{\boldsymbol{m}_3 \cdot \boldsymbol{P}_i} \end{pmatrix} \iff \begin{pmatrix} \boldsymbol{m}_1^{\mathrm{T}} - u_i \boldsymbol{m}_3^{\mathrm{T}} \\ \boldsymbol{m}_2^{\mathrm{T}} - v_i \boldsymbol{m}_3^{\mathrm{T}} \end{pmatrix} \boldsymbol{P}_i = 0$$

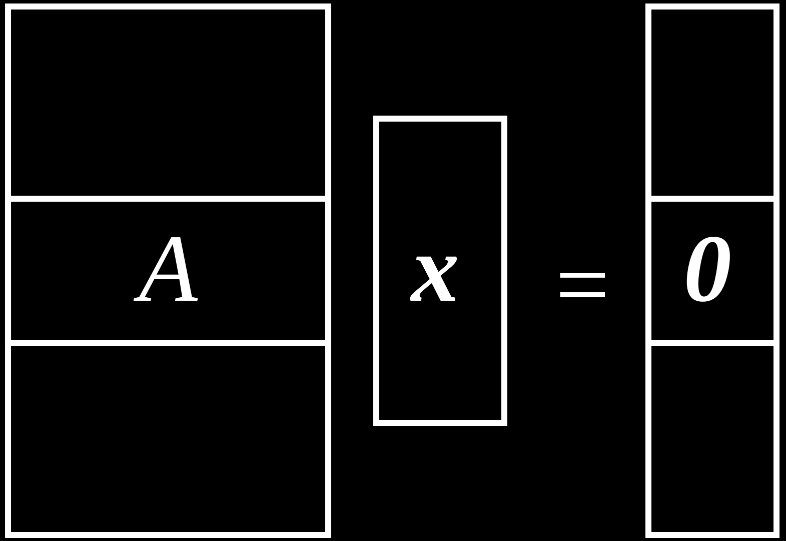
$$\mathcal{P}\boldsymbol{m} = 0 \text{ with } \mathcal{P} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{P}_1^T & \boldsymbol{0}^T & -u_1 \boldsymbol{P}_1^T \\ \boldsymbol{0}^T & \boldsymbol{P}_1^T & -v_1 \boldsymbol{P}_1^T \\ \ldots & \ldots & \ldots \\ \boldsymbol{P}_n^T & \boldsymbol{0}^T & -u_n \boldsymbol{P}_n^T \\ \boldsymbol{0}^T & \boldsymbol{P}_n^T & -v_n \boldsymbol{P}_n^T \end{pmatrix} \text{ and } \boldsymbol{m} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \boldsymbol{m}_3 \end{pmatrix} = 0$$

Homogeneous Linear Systems



Square system:

- unique solution: 0
- unless Det(A)=0



Rectangular system ??

- 0 is always a solution

Minimize $||Ax||^2$ under the constraint $||x||^2 = 1$

How do you solve overconstrained homogeneous linear equations ?? Homogeneous linear least squares

$$E = |\mathcal{U}\boldsymbol{x}|^2 = \boldsymbol{x}^T(\mathcal{U}^T\mathcal{U})\boldsymbol{x}$$

- Orthonormal basis of eigenvectors: $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_q$.
- Associated eigenvalues: $0 \leq \lambda_1 \leq \ldots \leq \lambda_q$.
- Any vector can be written as

$$\boldsymbol{x} = \mu_1\boldsymbol{e}_1 + \ldots + \mu_q\boldsymbol{e}_q$$

for some μ_i $(i = 1, \ldots, q)$ such that $\mu_1^2 + \ldots + \mu_q^2 = 1$.

$$\begin{aligned} E(\boldsymbol{x})-E(\boldsymbol{e}_1) &= \boldsymbol{x}^T(U^TU)\boldsymbol{x}-\boldsymbol{e}_1{}^T(U^TU)\boldsymbol{e}_1 \\ &= \lambda_1\mu_1{}^2+ \ldots +\lambda_q\mu_q{}^2-\lambda_1 \\ &\geq \lambda_1(\mu_1{}^2+ \ldots +\mu_q{}^2-1)=0 \end{aligned}$$

The solution is $\boldsymbol{e}_1$.

Linear Camera Calibration

Given n points $P_1, \ldots, P_n$ with *known* positions and their images $p_1, \ldots, p_n$

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \dfrac{\boldsymbol{m}_1 \cdot \boldsymbol{P}_i}{\boldsymbol{m}_3 \cdot \boldsymbol{P}_i} \\ \dfrac{\boldsymbol{m}_2 \cdot \boldsymbol{P}_i}{\boldsymbol{m}_3 \cdot \boldsymbol{P}_i} \end{pmatrix} \iff \begin{pmatrix} \boldsymbol{m}_1 - u_i \boldsymbol{m}_3 \\ \boldsymbol{m}_2 - v_i \boldsymbol{m}_3 \end{pmatrix} \boldsymbol{P}_i = 0$$

$$\mathcal{P}\boldsymbol{m} = 0 \text{ with } \mathcal{P} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{P}_1^T & \boldsymbol{0}^T & -u_1 \boldsymbol{P}_1^T \\ \boldsymbol{0}^T & \boldsymbol{P}_1^T & -v_1 \boldsymbol{P}_1^T \\ \ldots & \ldots & \ldots \\ \boldsymbol{P}_n^T & \boldsymbol{0}^T & -u_n \boldsymbol{P}_n^T \\ \boldsymbol{0}^T & \boldsymbol{P}_n^T & -v_n \boldsymbol{P}_n^T \end{pmatrix} \text{ and } \boldsymbol{m} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \boldsymbol{m}_3 \end{pmatrix} = 0$$

Minimize $||Pm||^2$ under the constraint $||m||^2 = 1$

Once *M* is known, you still got to recover the intrinsic and extrinsic parameters !!!

This is a decomposition problem, not an estimation problem.

$$\rho\, \mathcal{M} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix}$$

- Intrinsic parameters
- Extrinsic parameters

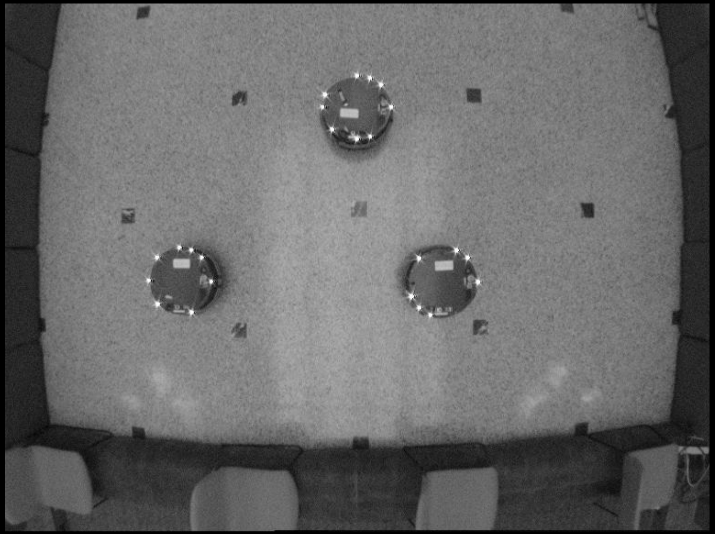
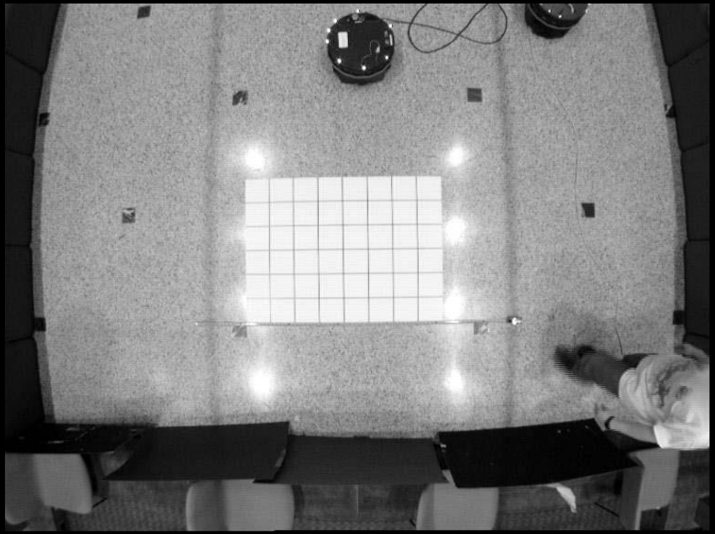
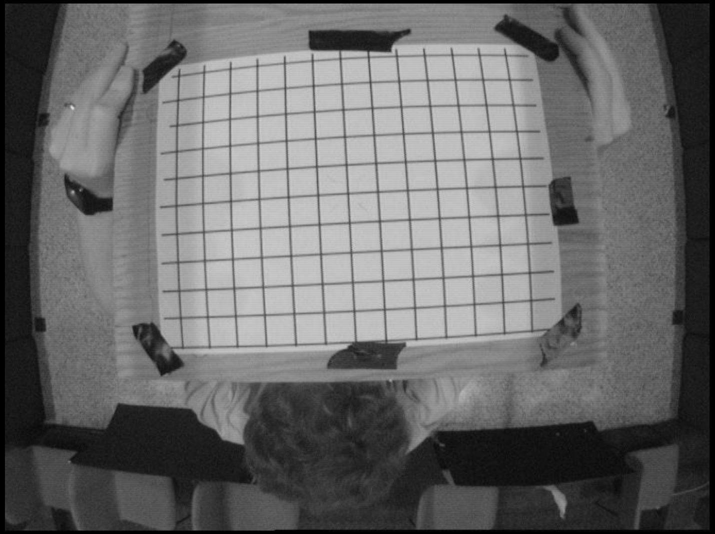
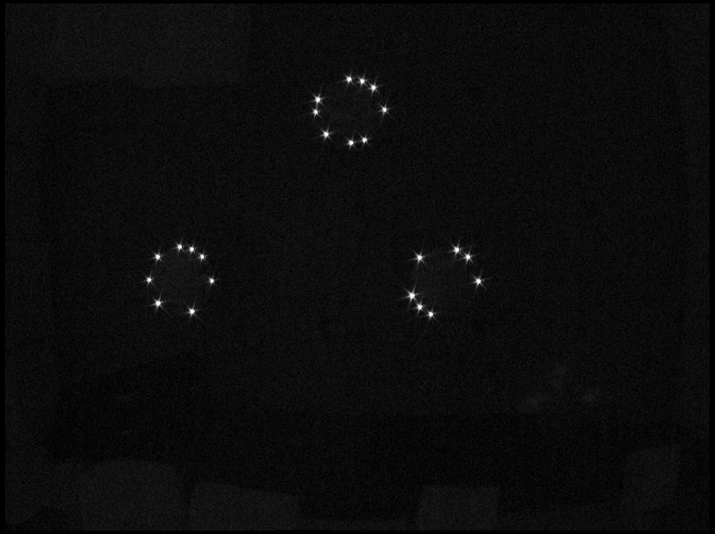
Weak-Perspective Projection Model

$$\boldsymbol{p} = \frac{1}{z_r}\mathcal{M}P$$ (***p*** and *P* are in homogeneous coordinates)

$p = M P$ (*P* is in homogeneous coordinates)

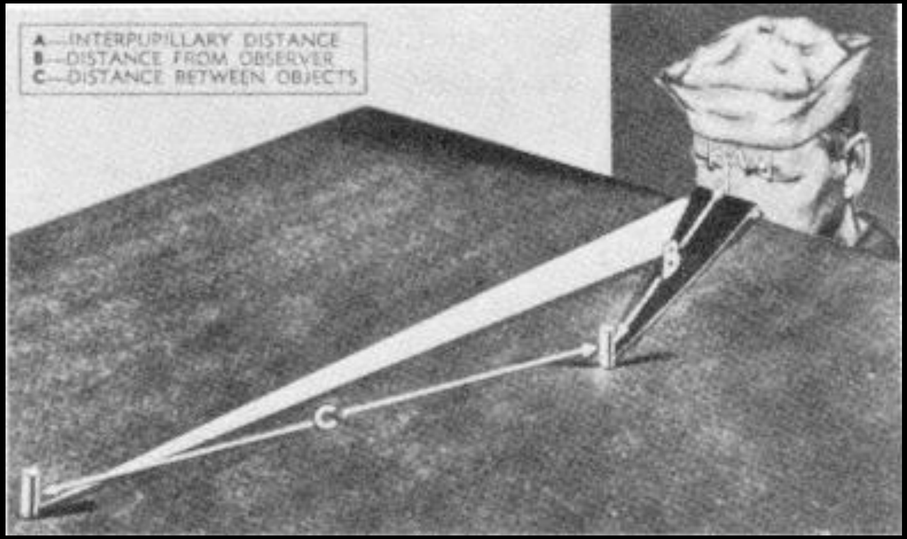
$p = A P + b$ (neither ***p*** nor *P* is in hom. coordinates)

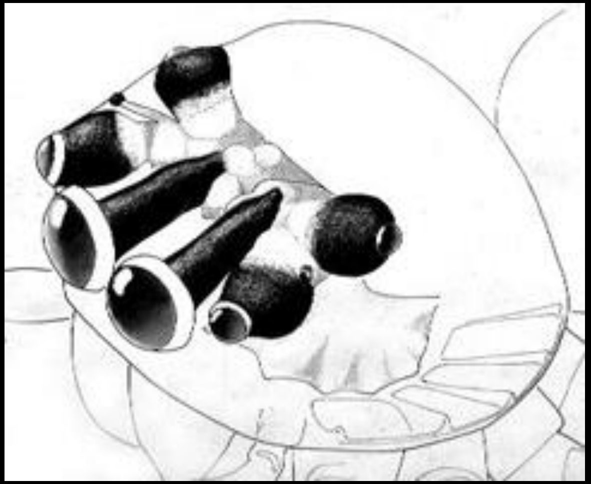
Applications: Mobile Robot Localization (Devy et al., 1997)

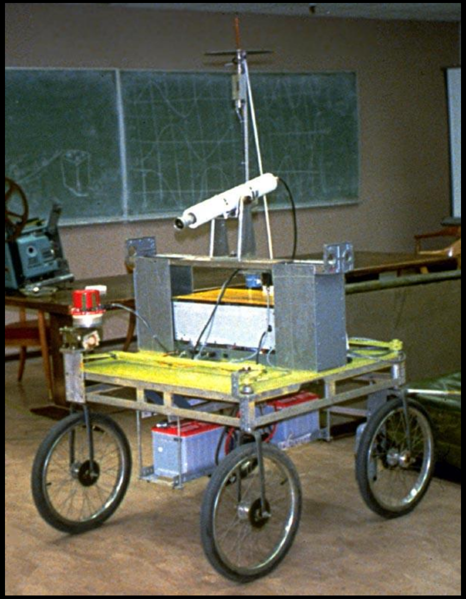





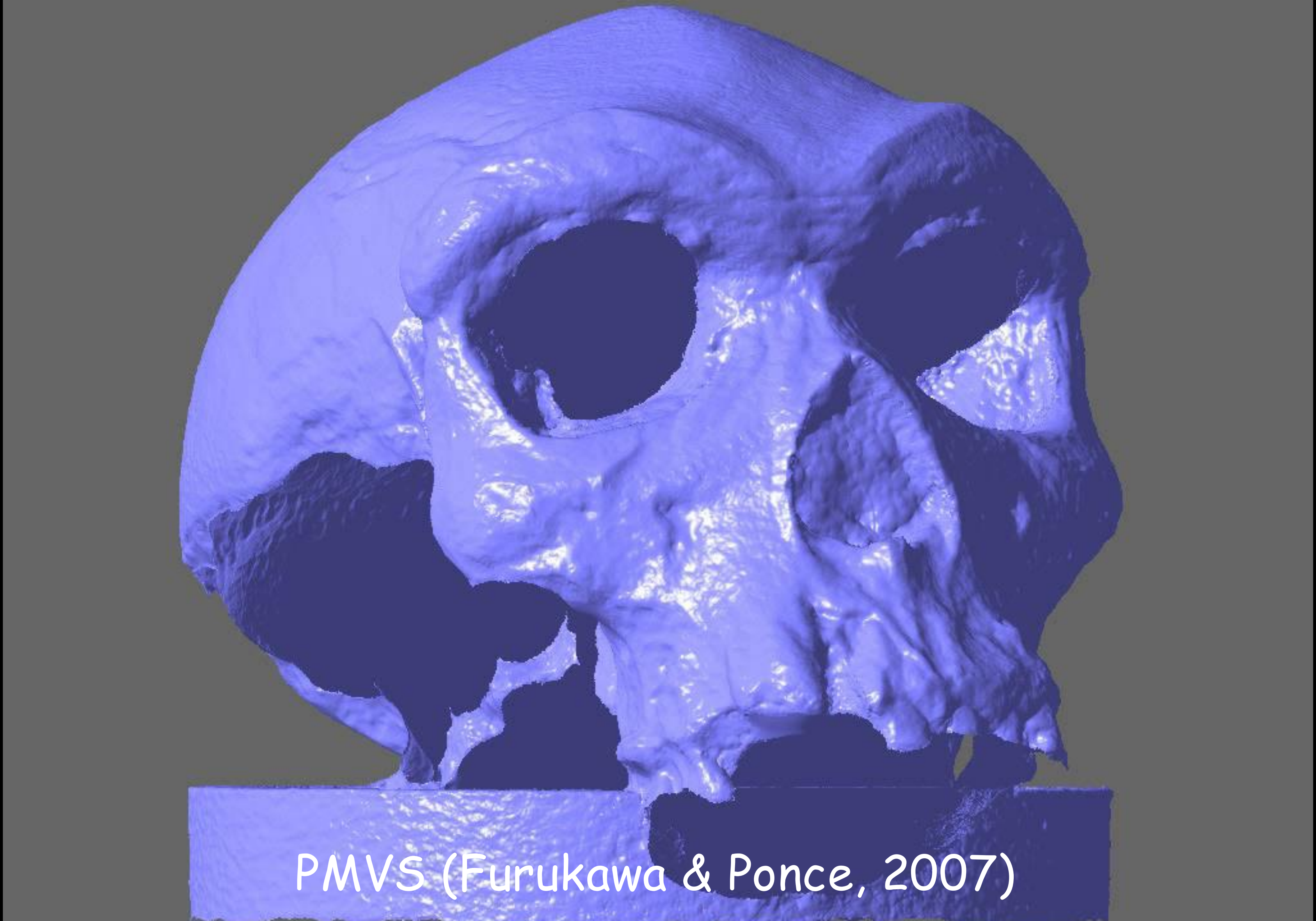
(Rothganger, Sudsang, Ponce, 2002)

How do we perceive depth?



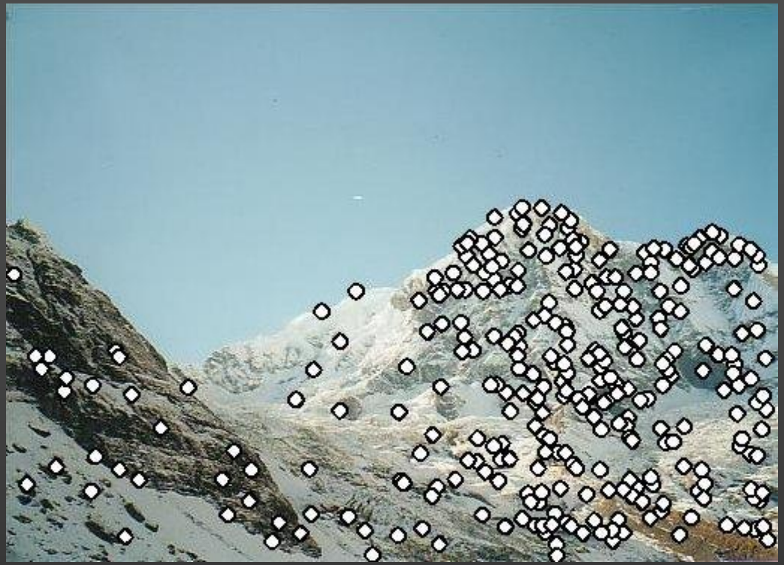
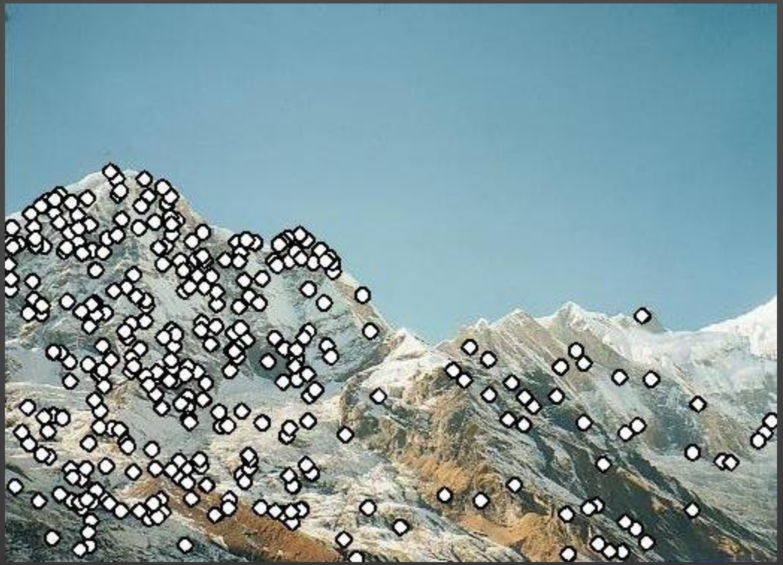





PMVS (Furukawa & Ponce, 2007)

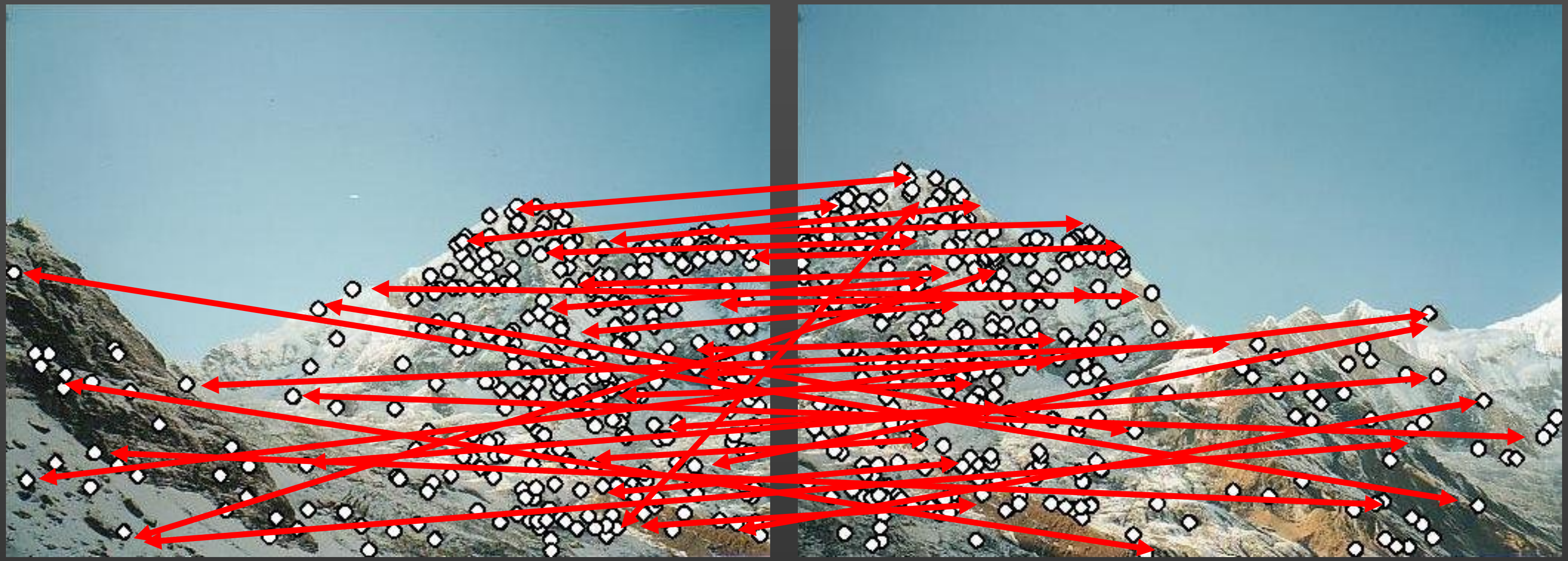
Feature-based alignment outline

Feature-based alignment outline

Extract features

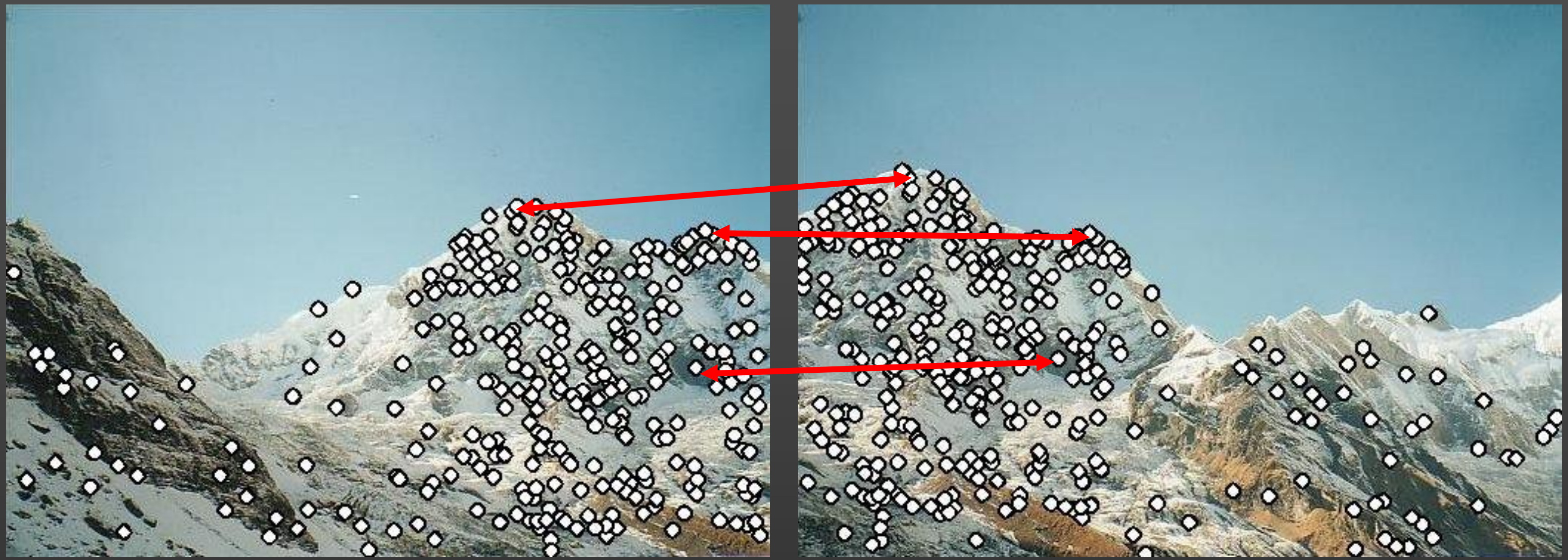
Feature-based alignment outline



Extract features

Compute *putative matches*

Feature-based alignment outline



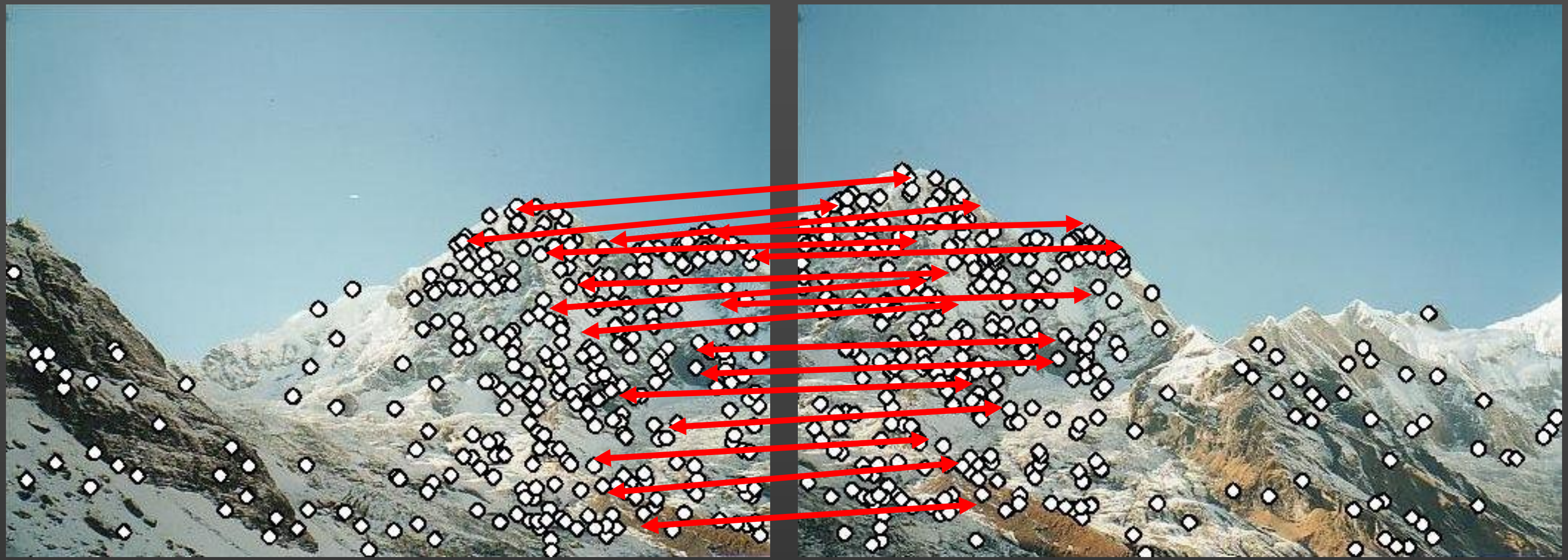
Extract features

Compute *putative matches*

Loop:

- *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

Feature-based alignment outline



Extract features

Compute *putative matches*

Loop:

- *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
- *Verify* transformation (search for other matches consistent with *T*)

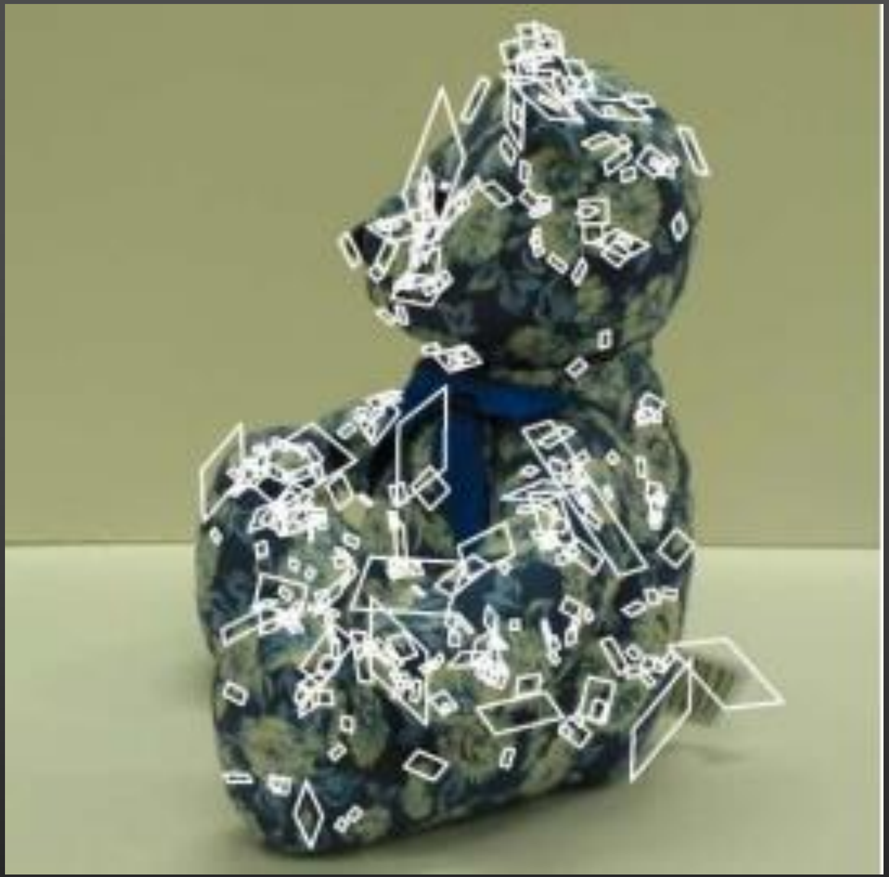
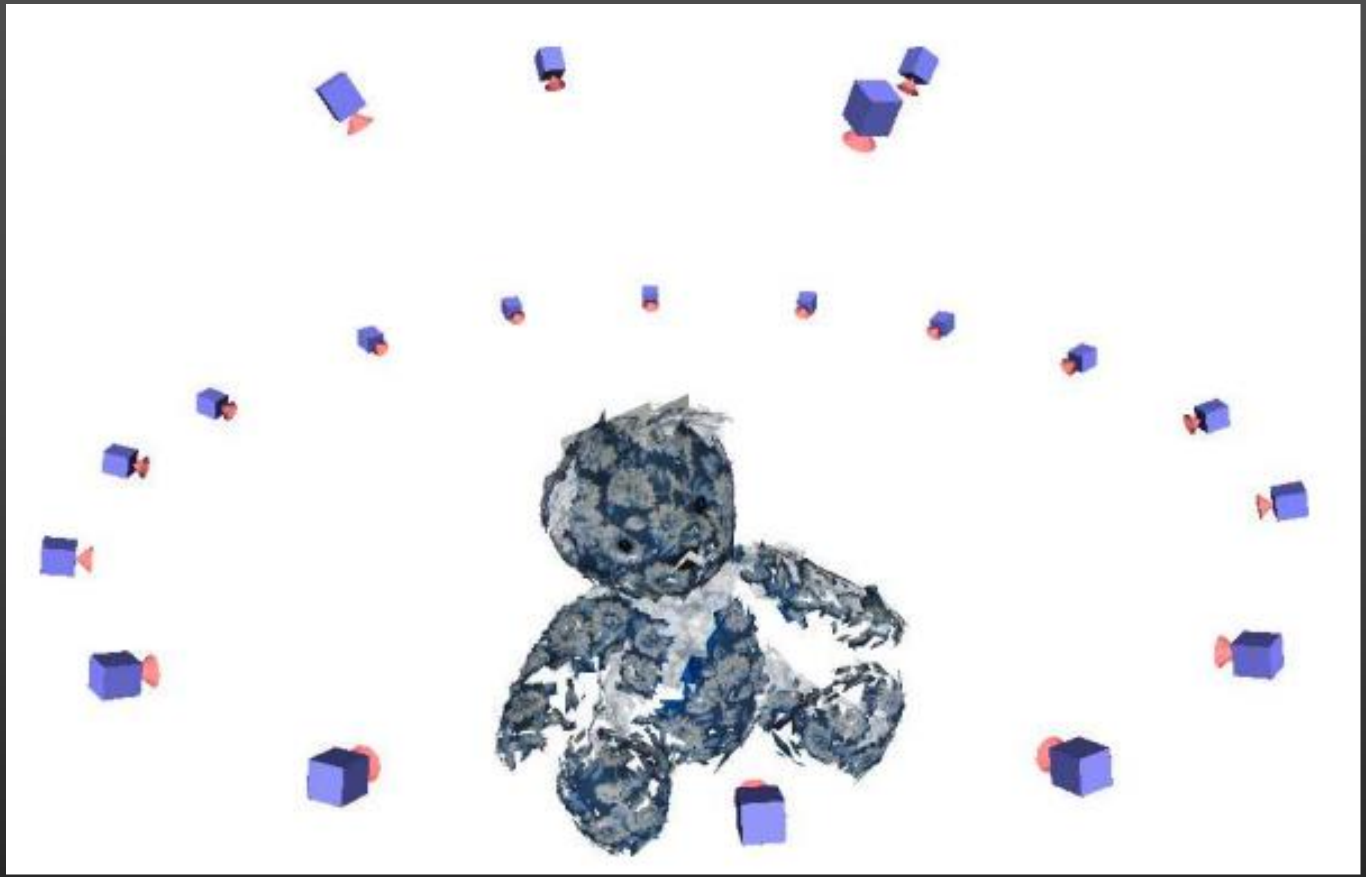
Feature-based alignment outline

Extract features

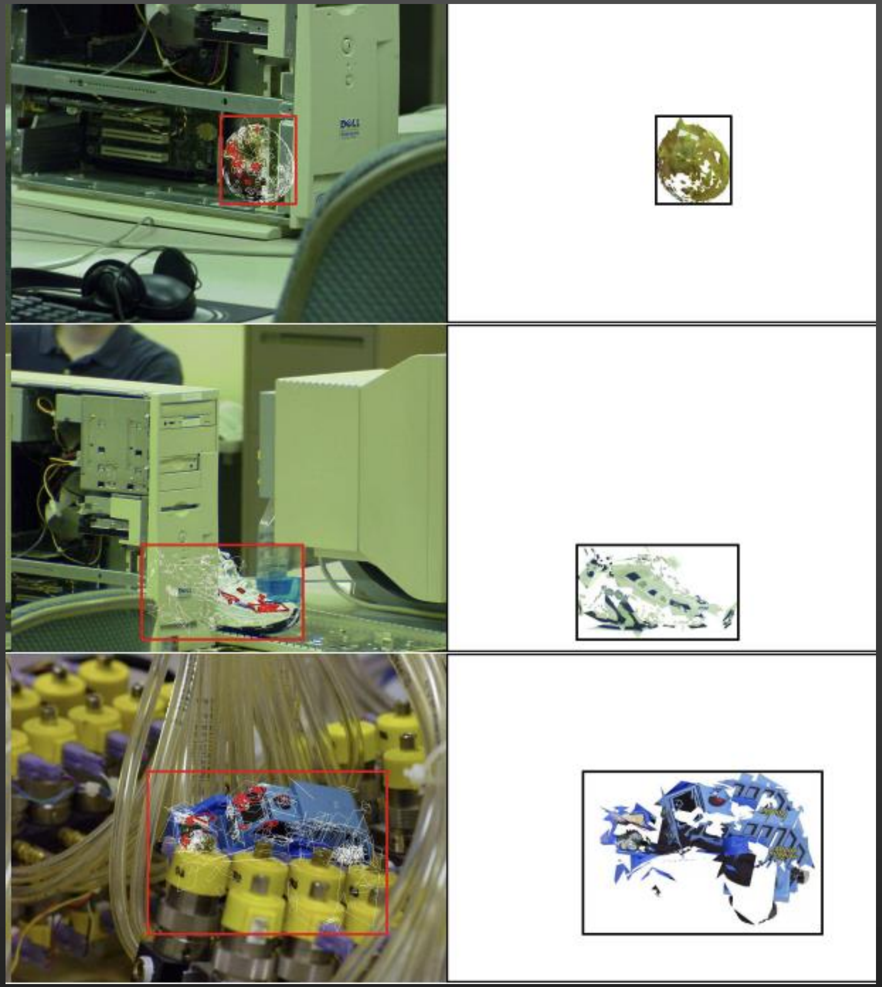
Compute *putative matches*

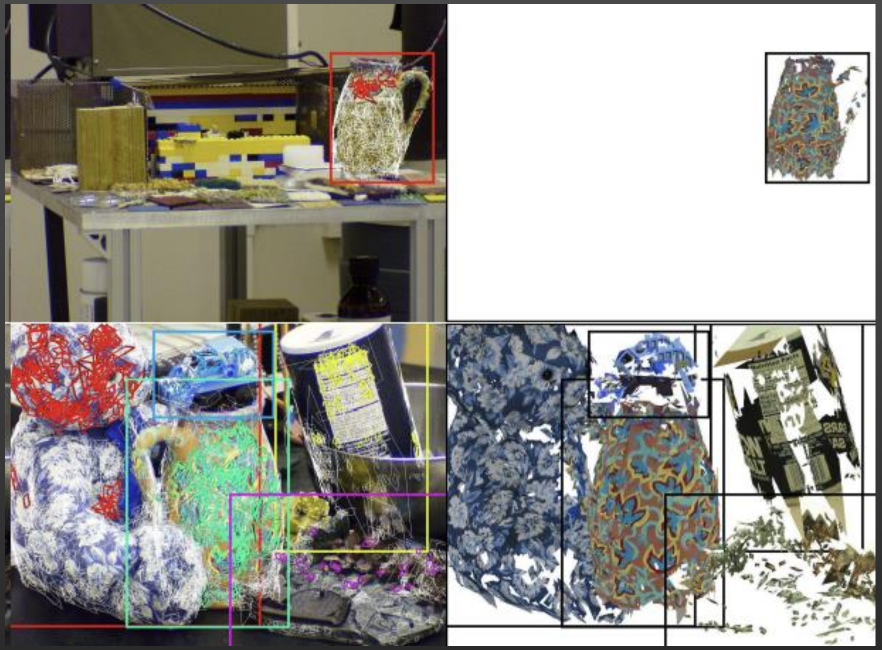
Loop:

- *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
- *Verify* transformation (search for other matches consistent with *T*)

3D object modeling from multiple images
(Rothganger et al., 2003)

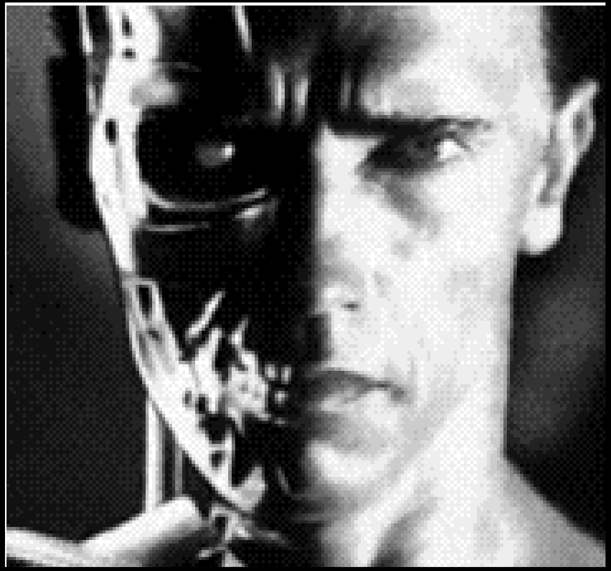


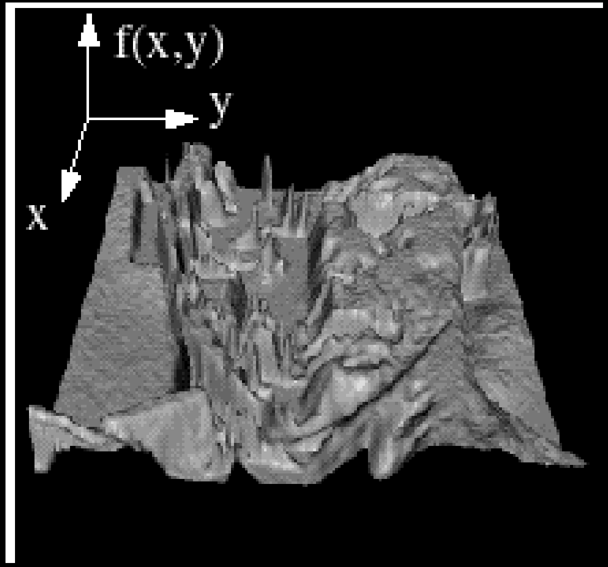


Recognition examples with major clutter and partial occlusion (Rothganger et al., 2003)

Image processing

- Filters and convolution
- Derivatives and edge detection
- The Canny edge detector
- Denoising, sparsity and dictionary learning
- Super-resolution

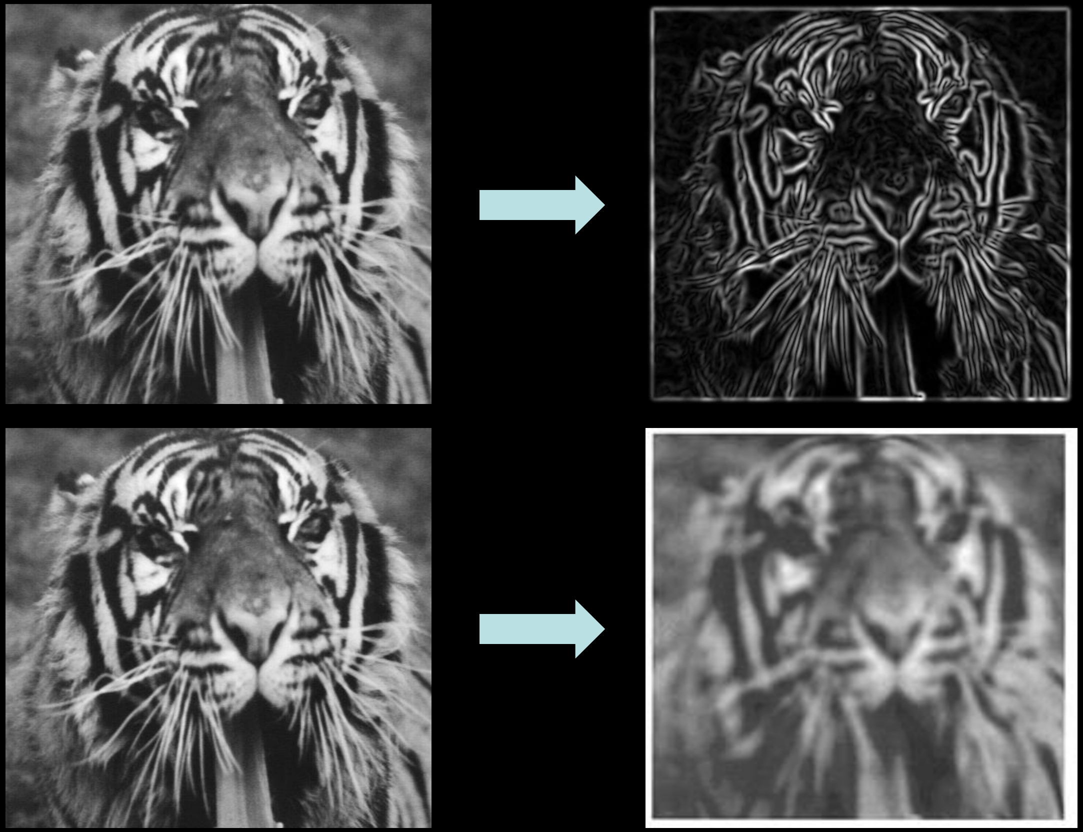




An image can be interpreted either as:

- a continuous function $f(x, y)$
- a discrete array $F_{u,v}$

Basic Filters



Convolution

Linear filters = Weighted averages

- Represent the weights by a rectangular array F.

- Applying the filter to an image G is equivalent to performing a convolution:

$$R_{ij} = (F*G)_{ij} = \sum_{u,v} F_{i-u,\, j-v} \; G_{u,\, v}$$

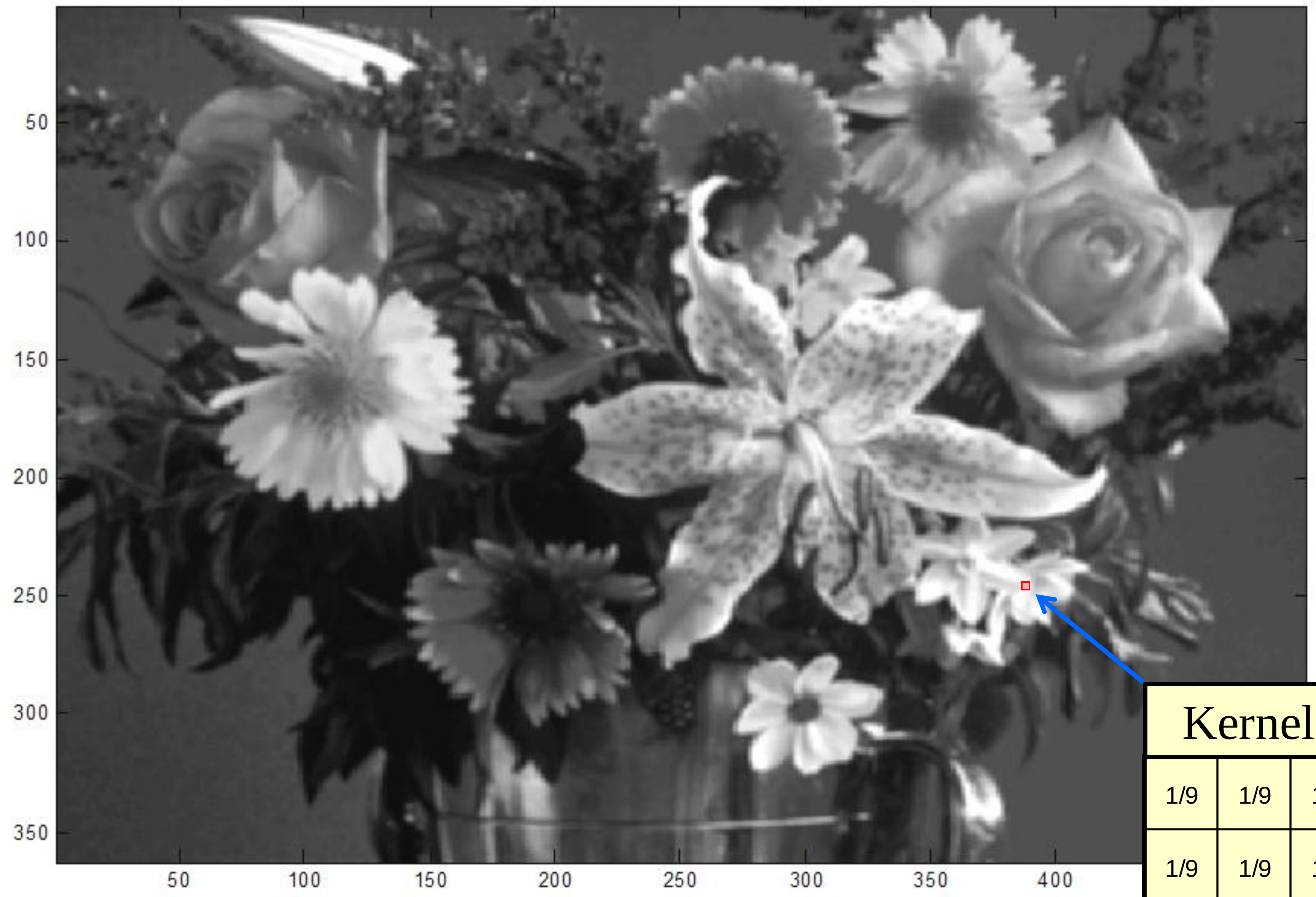
- In the continuous case:

$$(f * g)\,(x,y) = \int_{u,v} f(x-u,y-v)\; g(u,v)\; du\; dv$$

- Note: $f*g=g*f$.

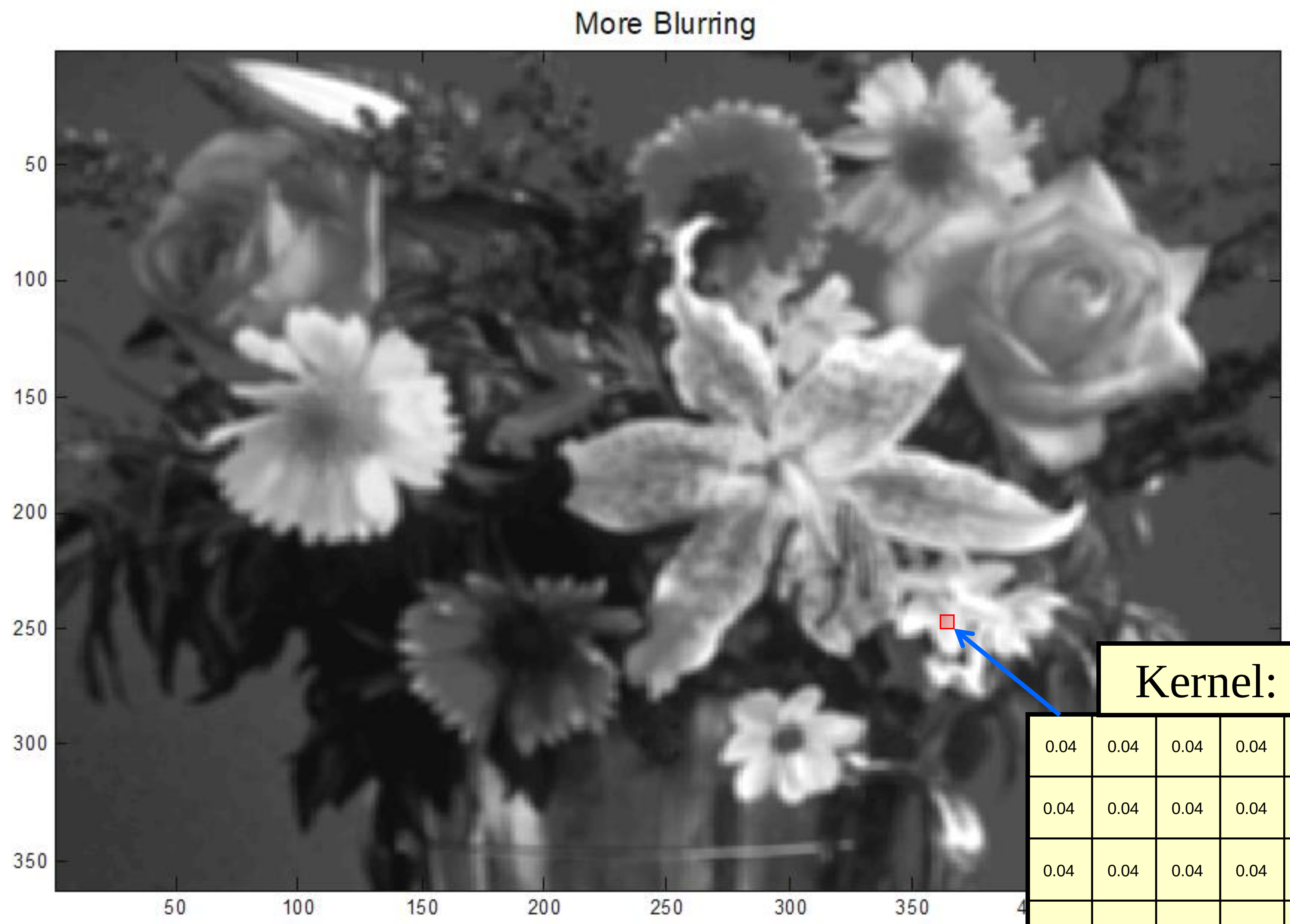
Original Image

Slight Blurring



Kernel:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Kernel:

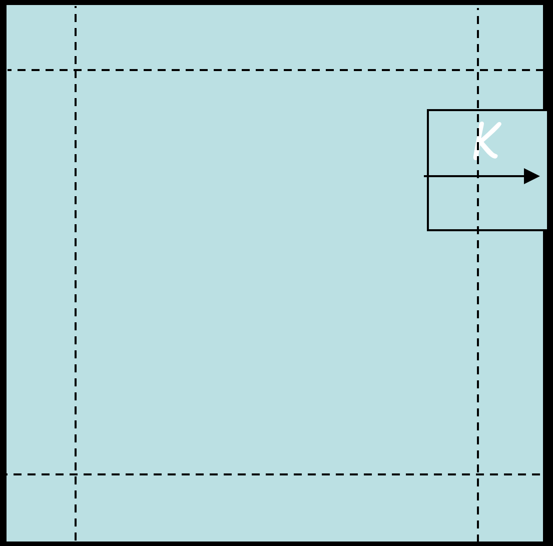
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

Basic Properties

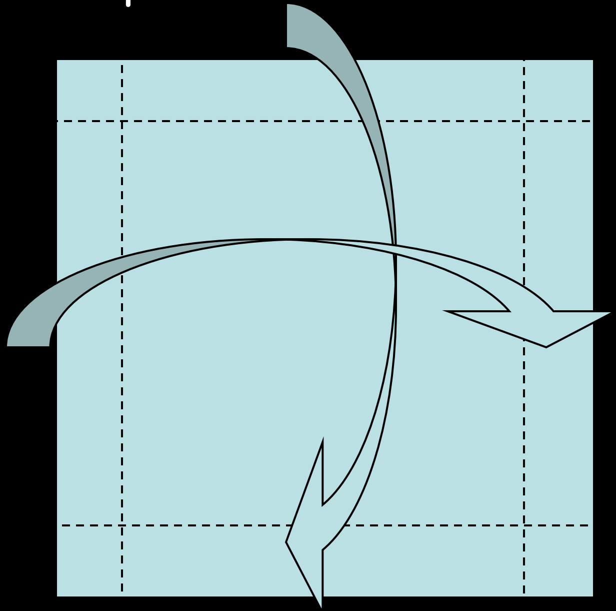
- Commutativity: $f * g = g * f$
- Associativity: $(f * g) * h = f * (g * h)$
- Linearity: $(af + bg) * h = a f * h + b g * h$
- Shift invariance: $f_t * h = (f * h)_t$
- Only operator both linear and shift invariant
- Differentiation: $$\frac{\partial}{\partial x}\left(f * g\right) = \frac{\partial f}{\partial x} * g$$

Practicalities (discrete convolution)

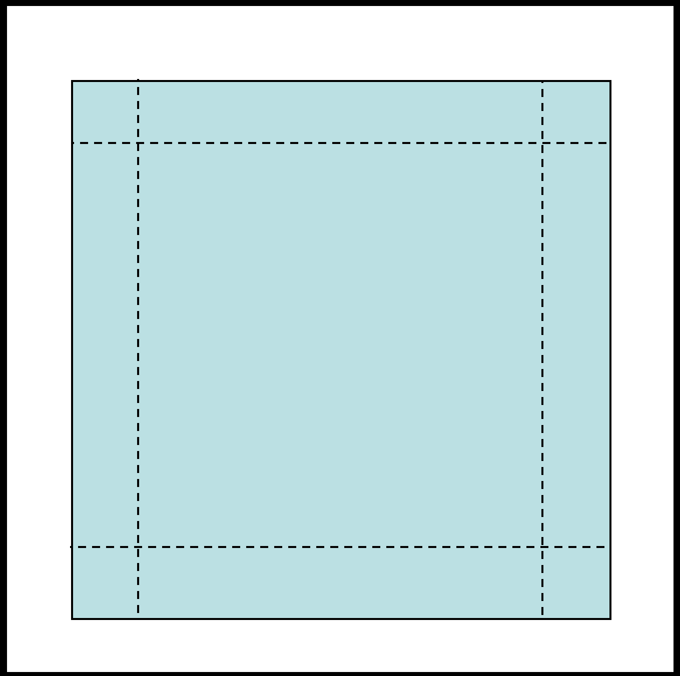
- Python: convolve function
- Border issues:
  - When applying convolution with a KxK kernel, the result is undefined for pixels closer than K pixels from the border of the image
- Options:



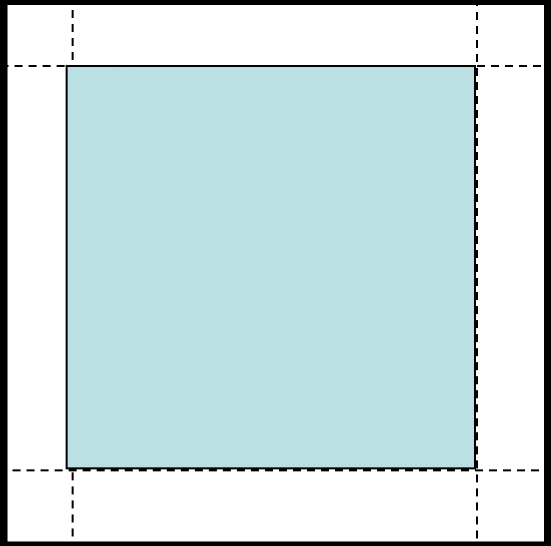
Wrap around



Expand/Pad



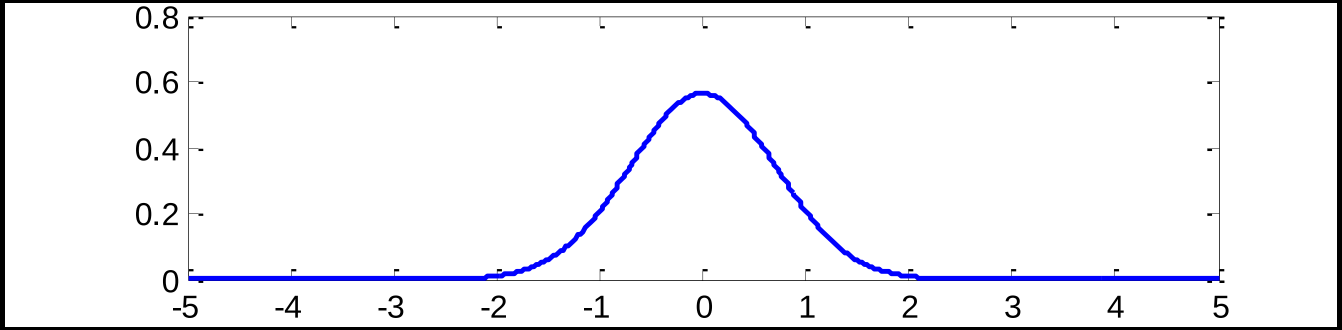
Crop



Gaussian filters

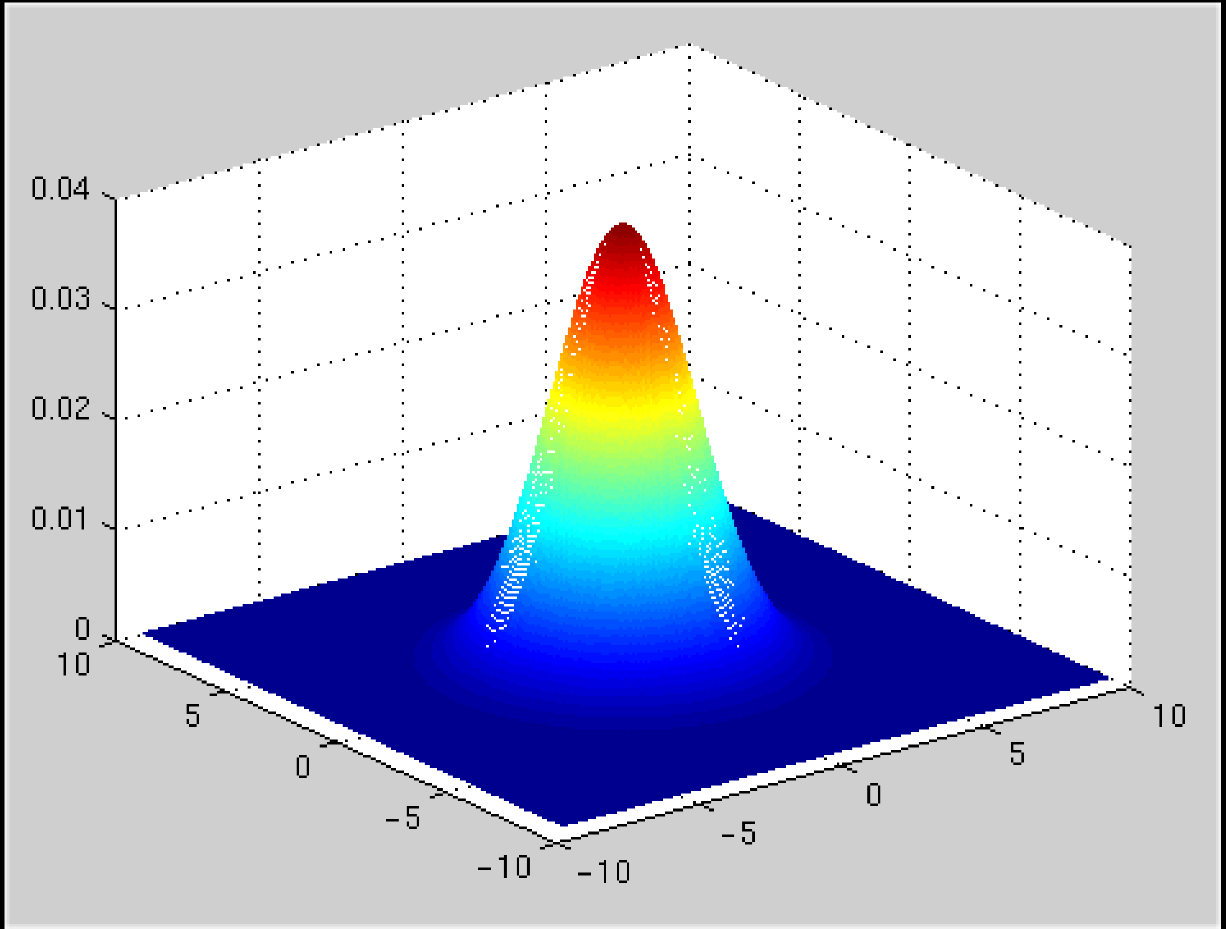
1-D:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



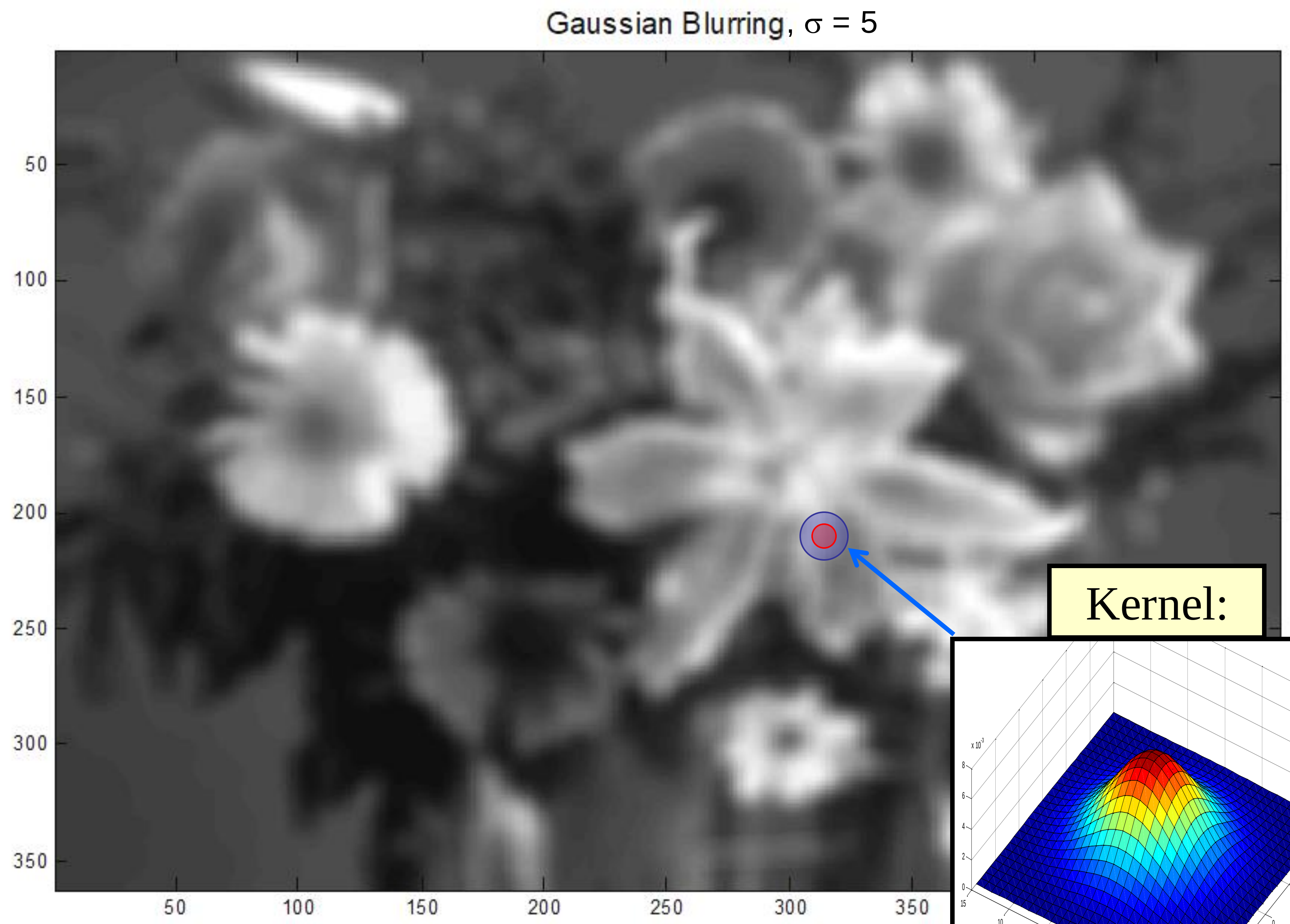
2-D:

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

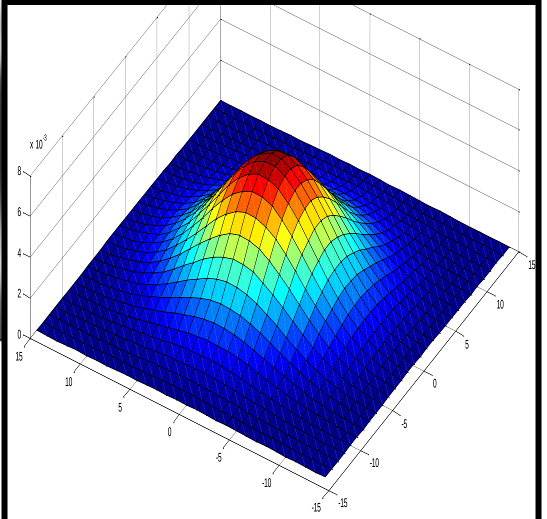


Slight abuse of notation:
We ignore the normalization constant such that

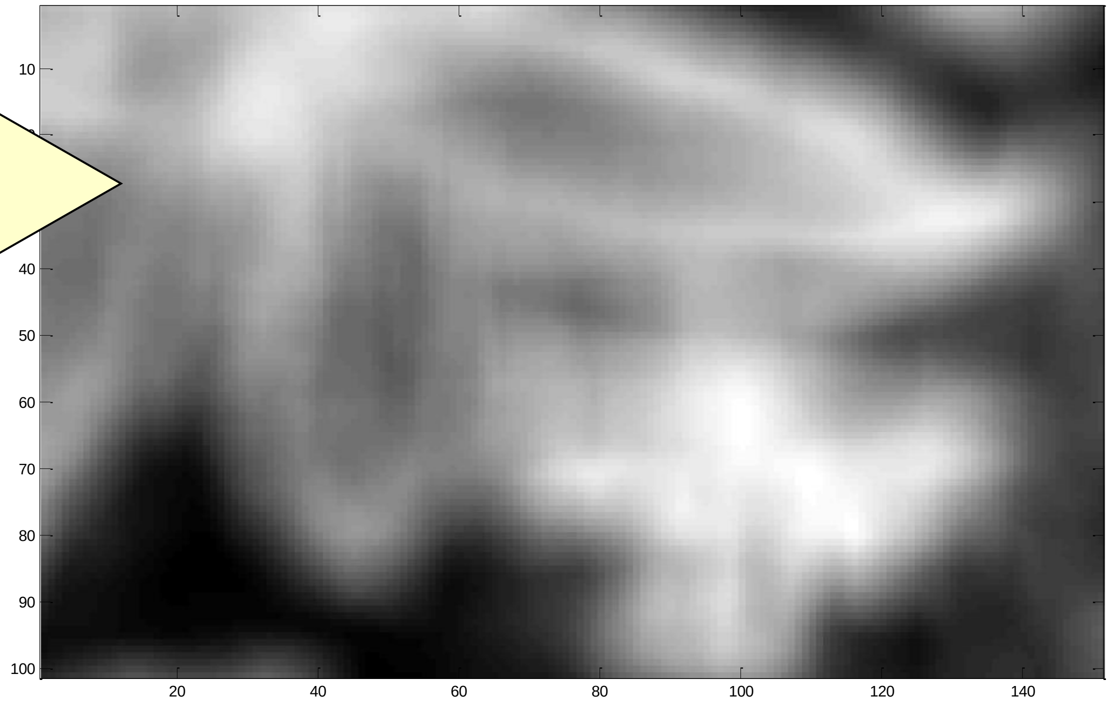
$$\int g(x)dx = 1$$


Gaussian Blurring, σ = 5
50
100
150
200
250
300
350
50
100
150
200
250
300
350

Kernel:



Simple Averaging



Gaussian Smoothing

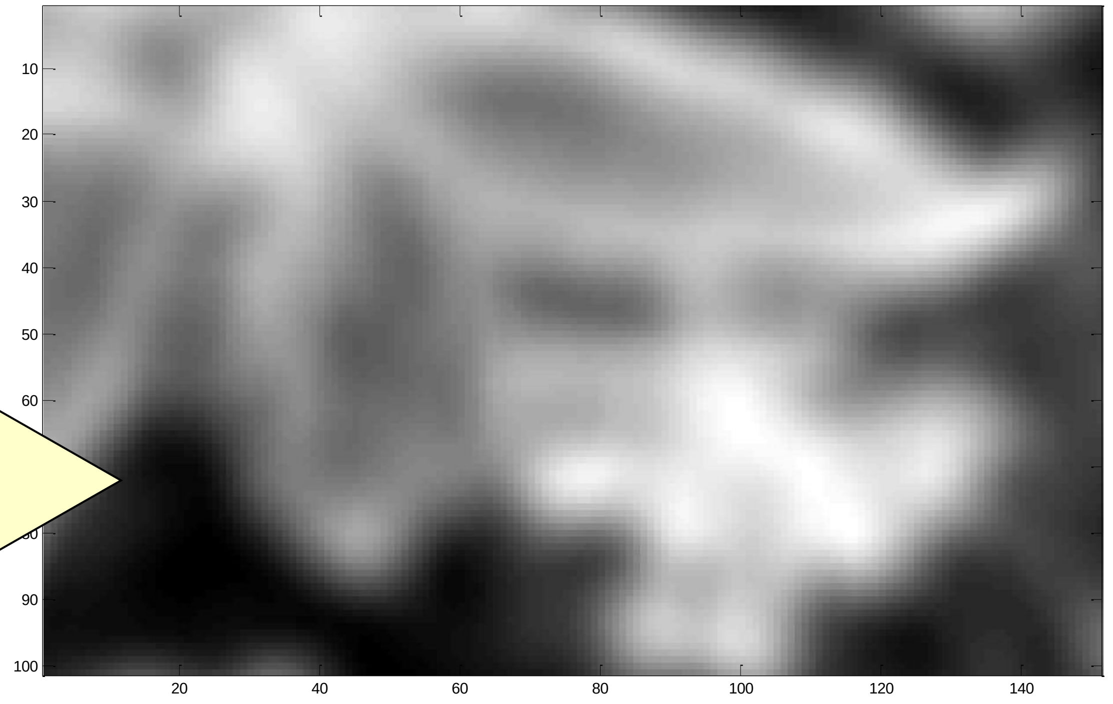
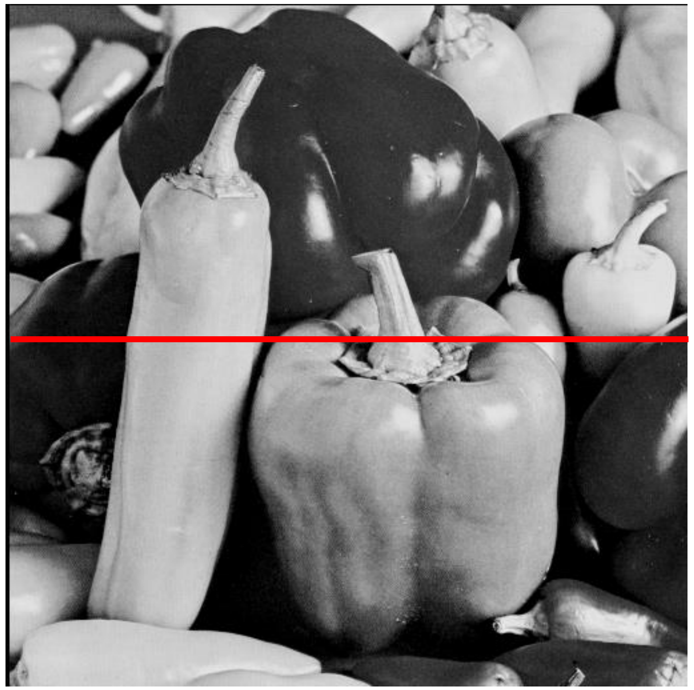
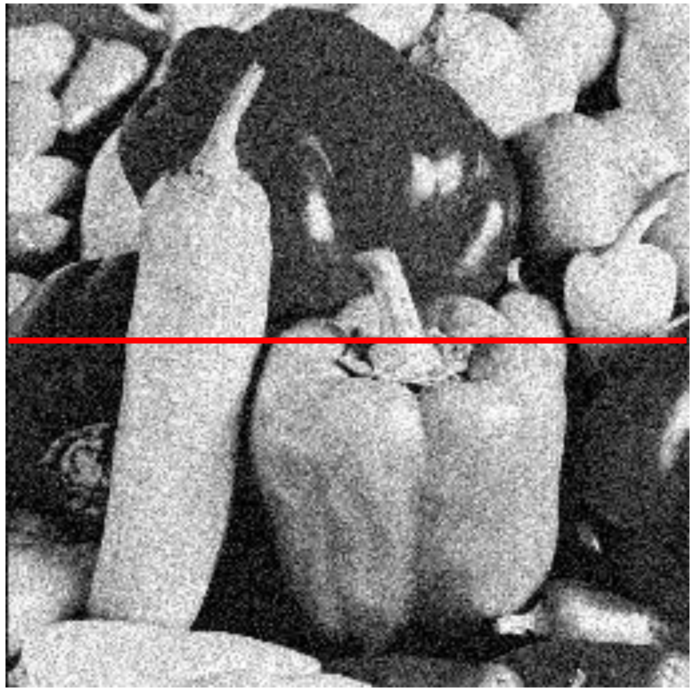
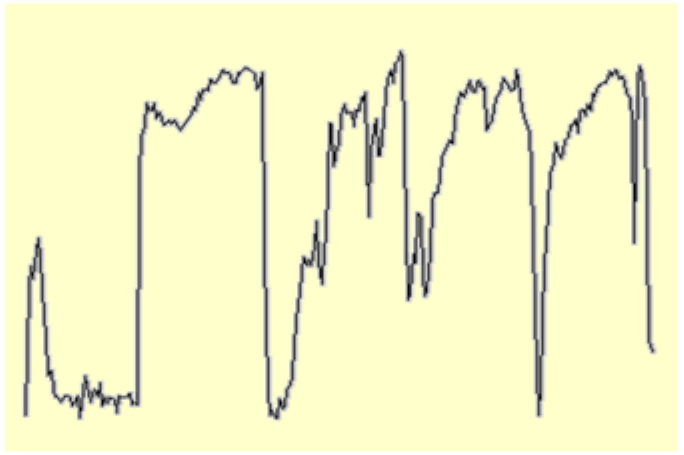
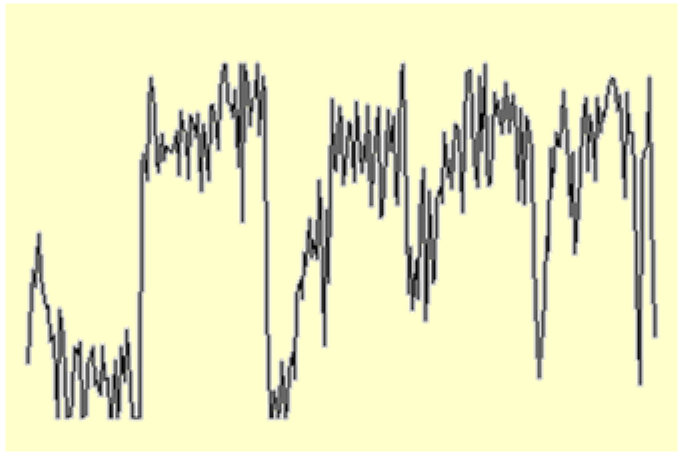


Image Noise





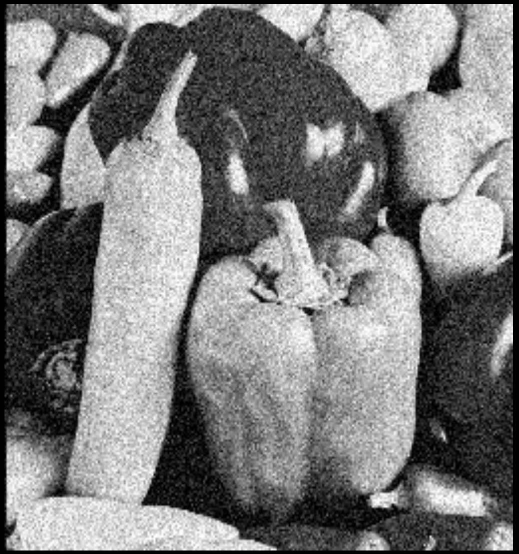
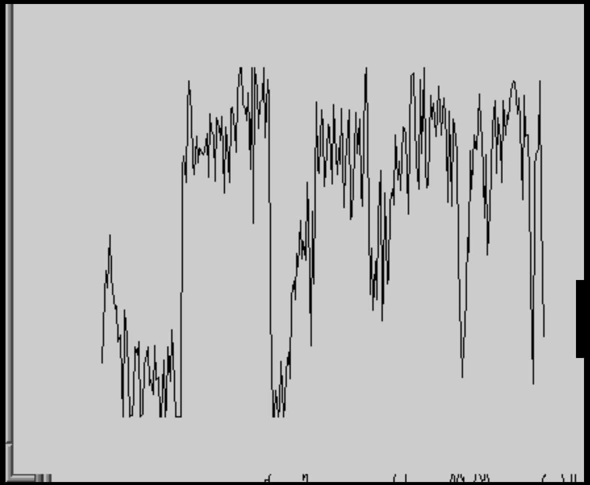




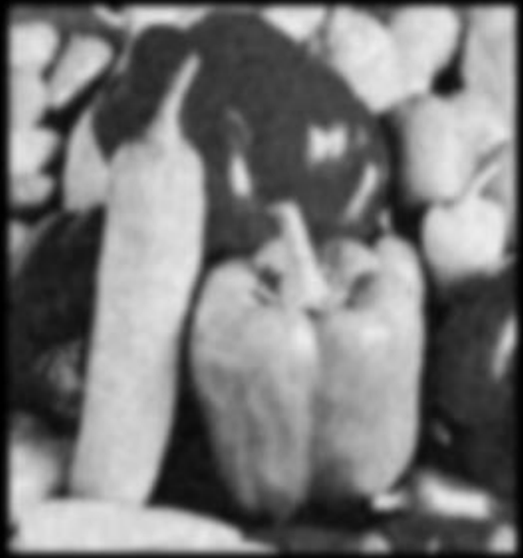
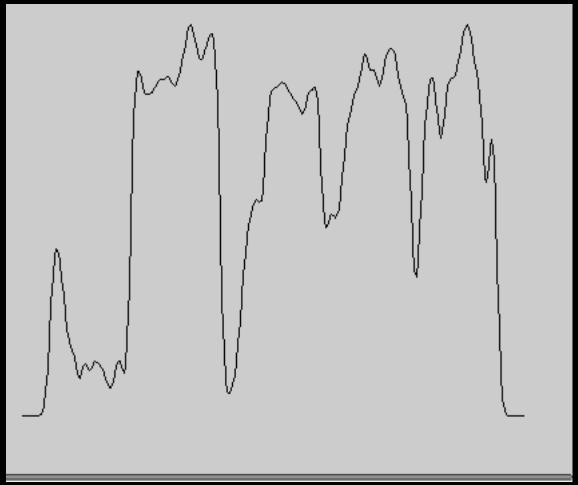
$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

IID Gaussian white noise
$\eta(x,y) \sim N(0,\sigma)$

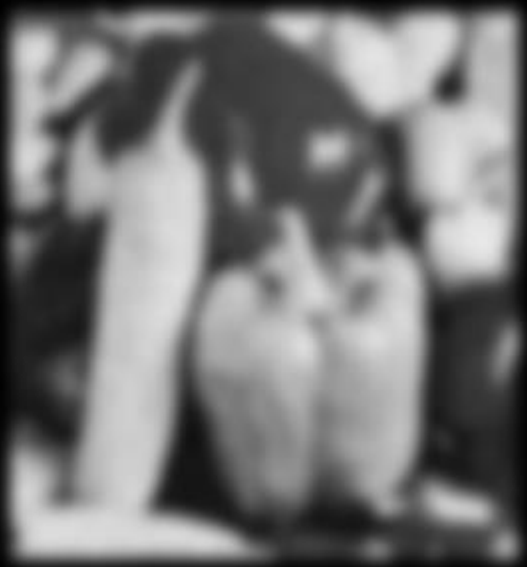
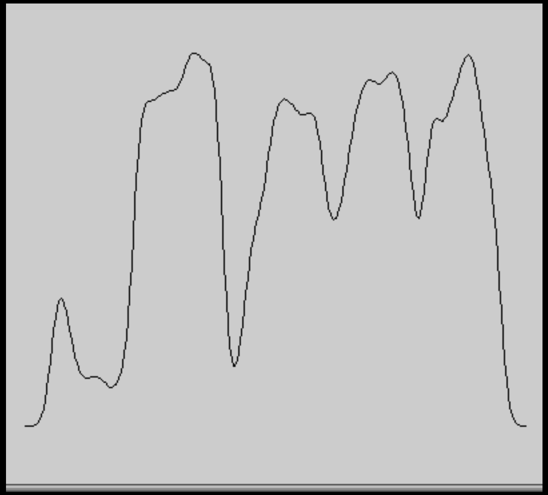
Gaussian Smoothing to Remove Noise

No smoothing

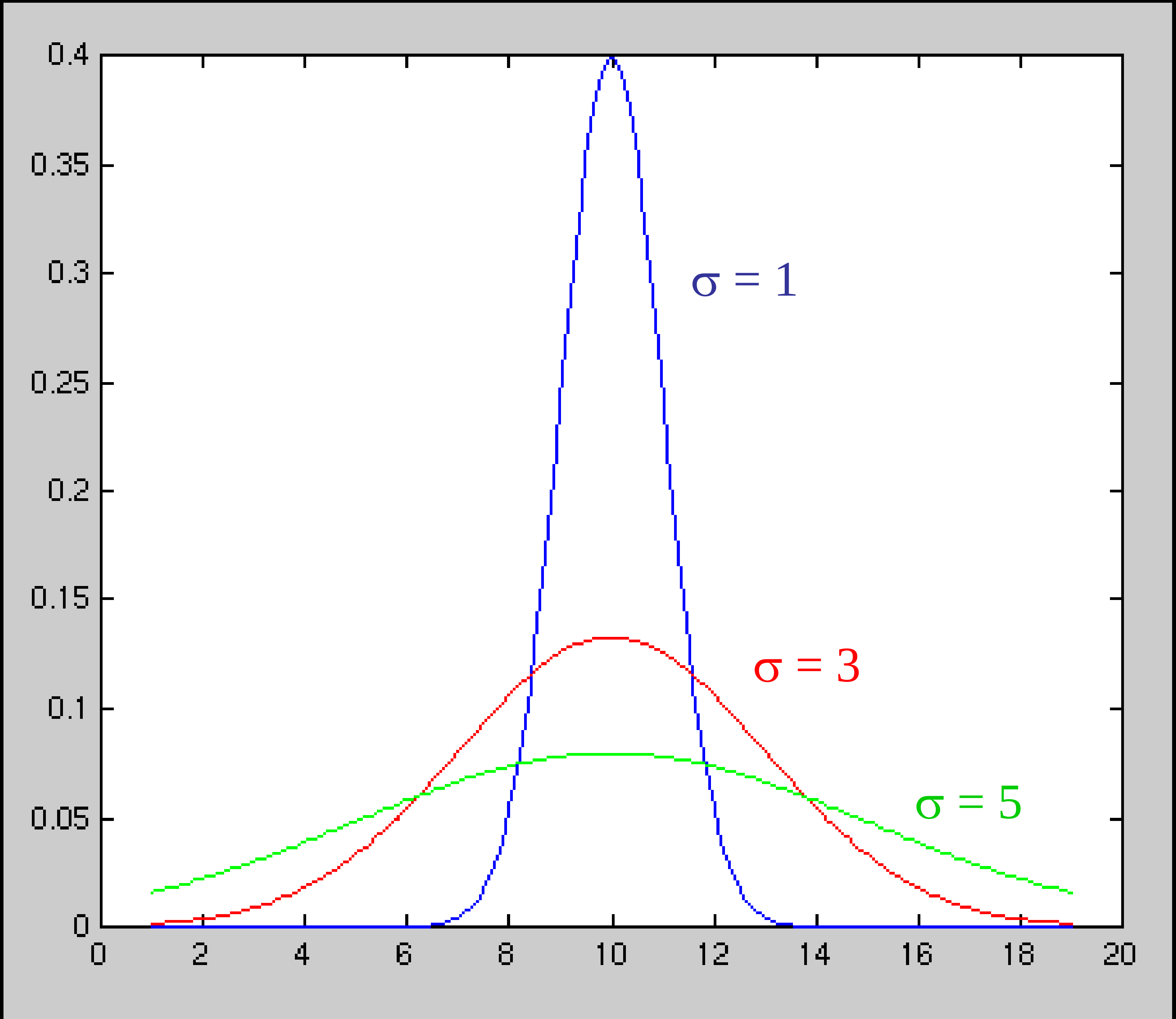
$\sigma = 2$

$\sigma = 4$

Bottom line: The standard deviation of white noise is divided by k*sigma

Shape of Gaussian filter as function of σ



Basic Properties

- Gaussian removes “high-frequency” components from the image → “low pass” filter
- Larger σ remove more details
- Combination of 2 Gaussian filters is a Gaussian filter:

$$G_{\sigma_1} * G_{\sigma_2} = G_\sigma \quad \sigma^2 = \sigma_1{}^2 + \sigma_2{}^2$$

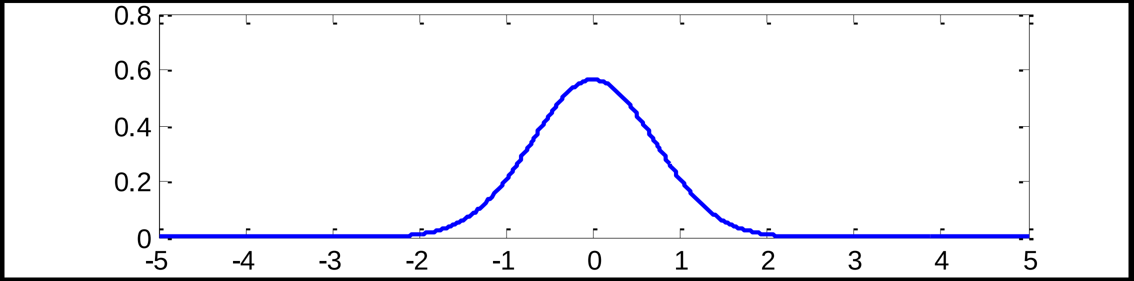
- Separable filter:

$$G_\sigma * f = g_{\sigma\rightarrow} * g_{\sigma\uparrow} * f$$

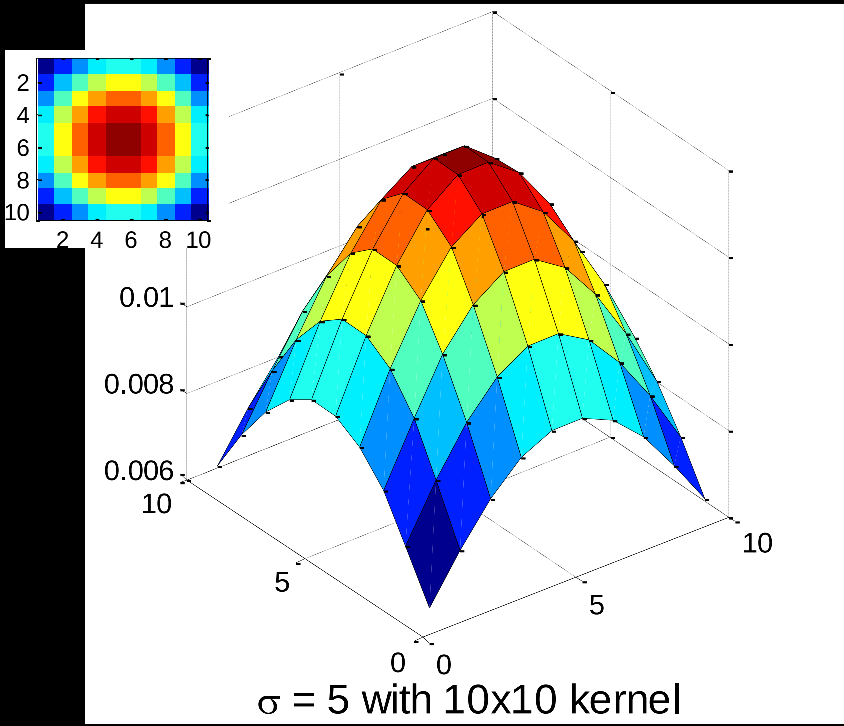
- Critical implication: Filtering with a *NxN* Gaussian kernel can be implemented as two convolutions of size *N* → reduction quadratic to linear → *must* be implemented that way

Note about Finite Kernel Support

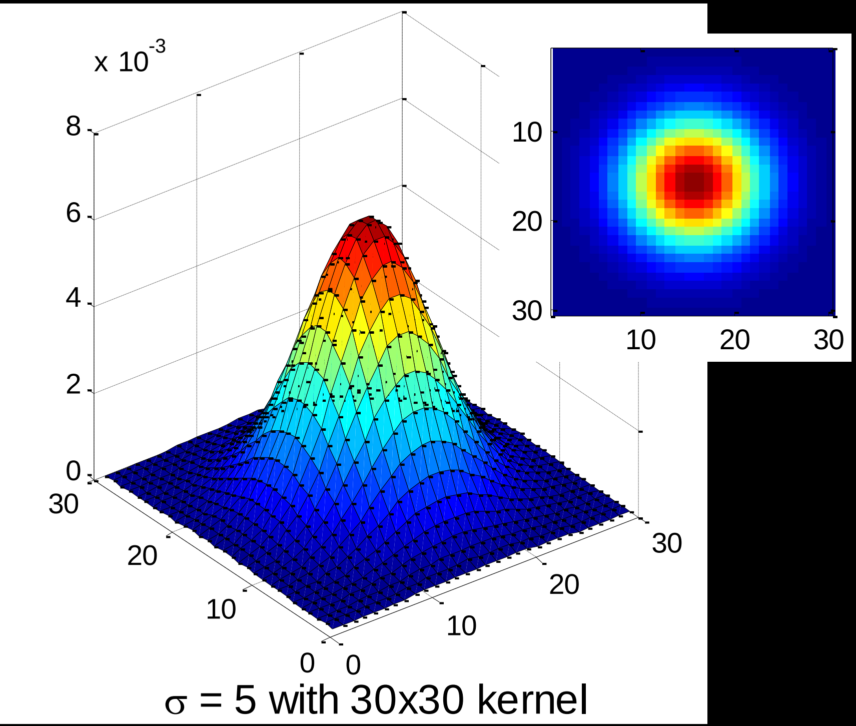
- Gaussian function has infinite support



- In actual filtering, we have a finite kernel size



σ = 5 with 10x10 kernel



σ = 5 with 30x30 kernel

Image Derivatives

- We want to compute, at each pixel (x,y) the derivatives:
- In the discrete case we could take the difference between the left and right pixels:

$$\frac{\partial I}{\partial x} \approx I(i+1, j) - I(i-1, j)$$

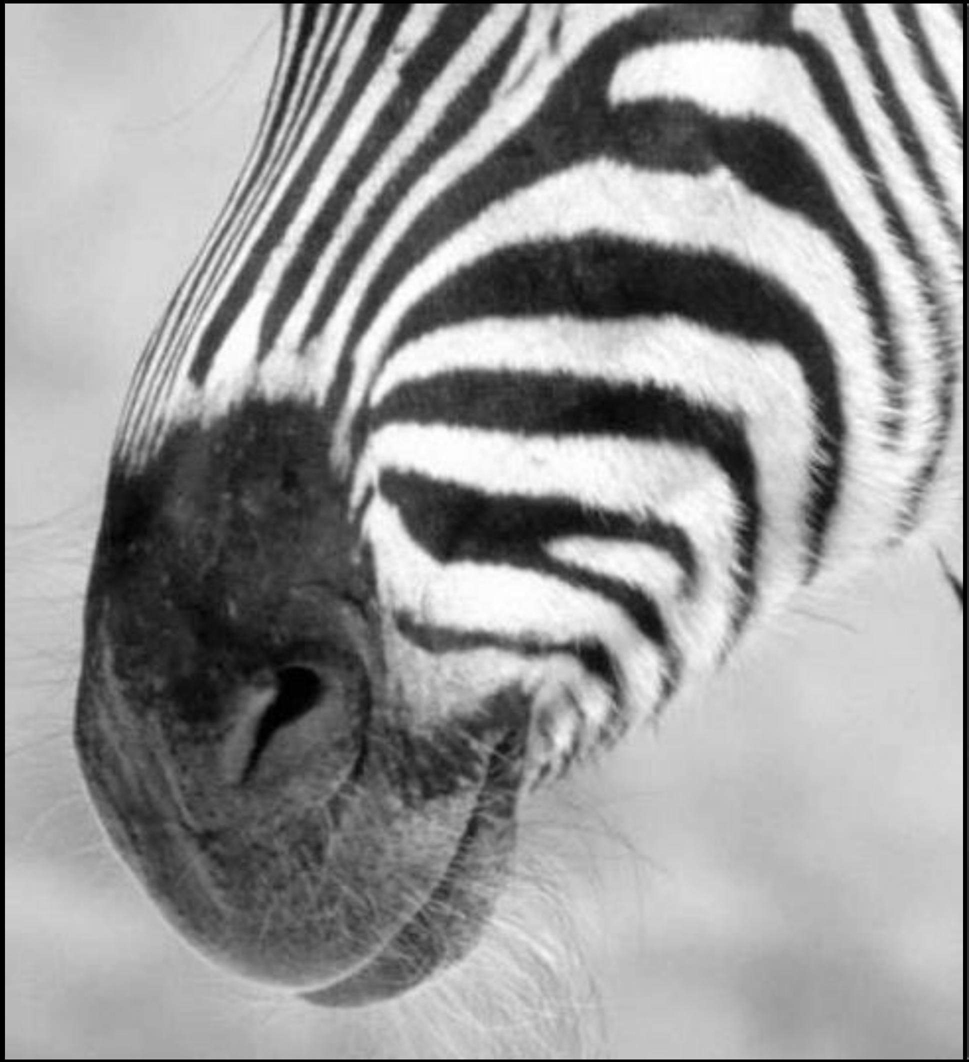
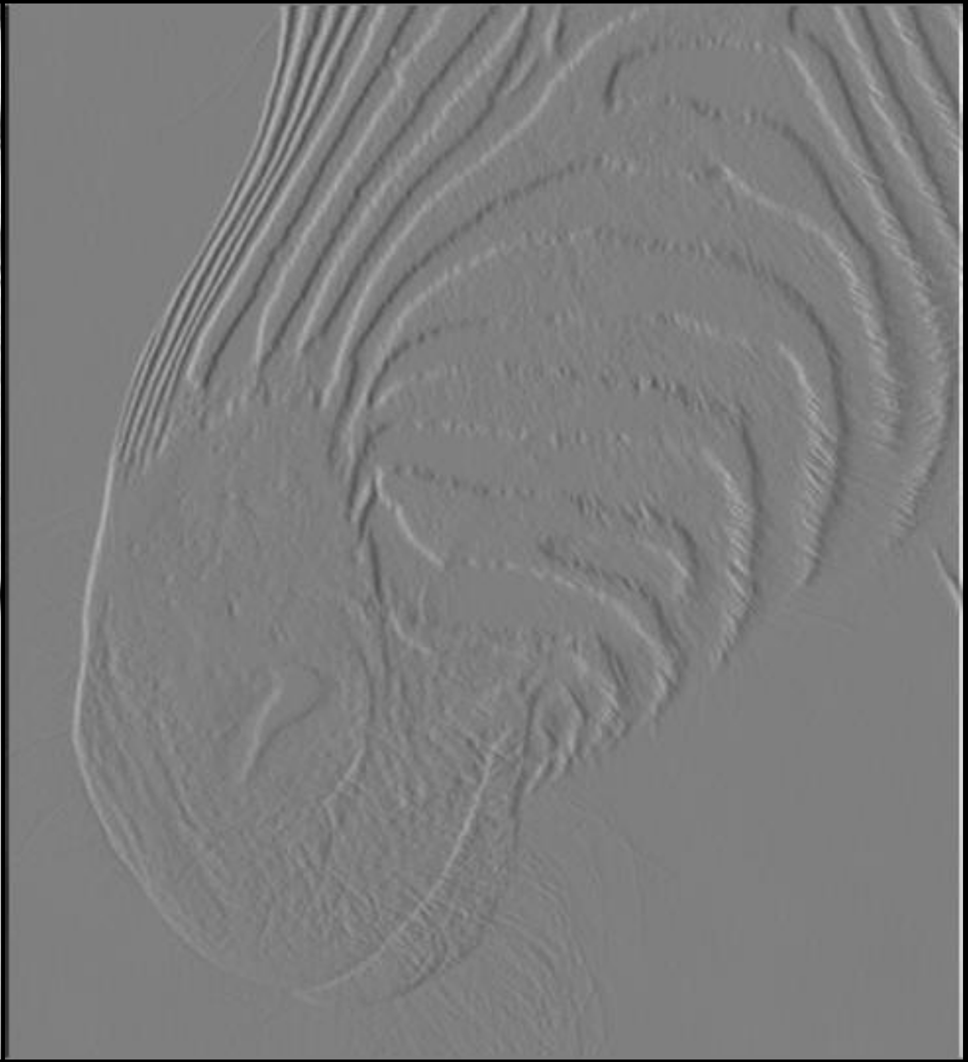
- Convolution of the image by

$$\partial_x = \boxed{1}\ \boxed{0}\ \boxed{-1}$$

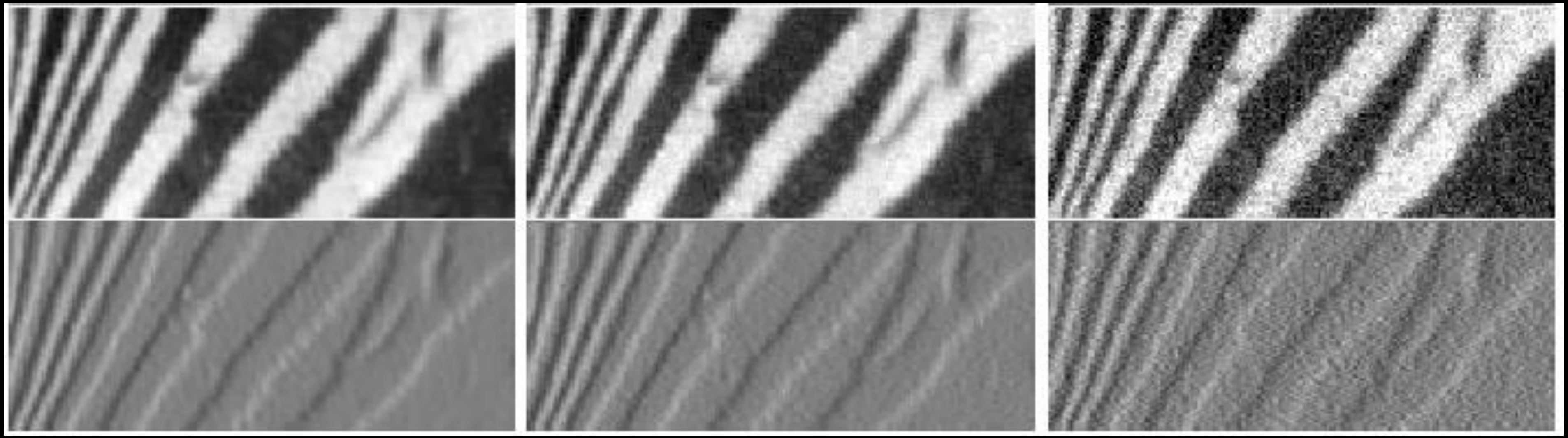
- Problem: Increases noise

$$\underbrace{I(i+1, j) - I(i-1, j)}_{\text{Difference between Actual image values}} = \underbrace{\hat{I}(i+1, j) - \hat{I}(i-1, j)}_{\text{True difference (derivative)}} + \underbrace{n_+ + n_-}_{\text{Sum of the noises}}$$

Finite differences

Finite differences responding to noise



Increasing zero-mean Gaussian noise

Smooth Derivatives

- Solution: First smooth the image by a Gaussian G_σ and then take derivatives:

$$\frac{\partial f}{\partial x} \approx \frac{\partial\left(G_\sigma * f\right)}{\partial x}$$

- Applying the differentiation property of the convolution:

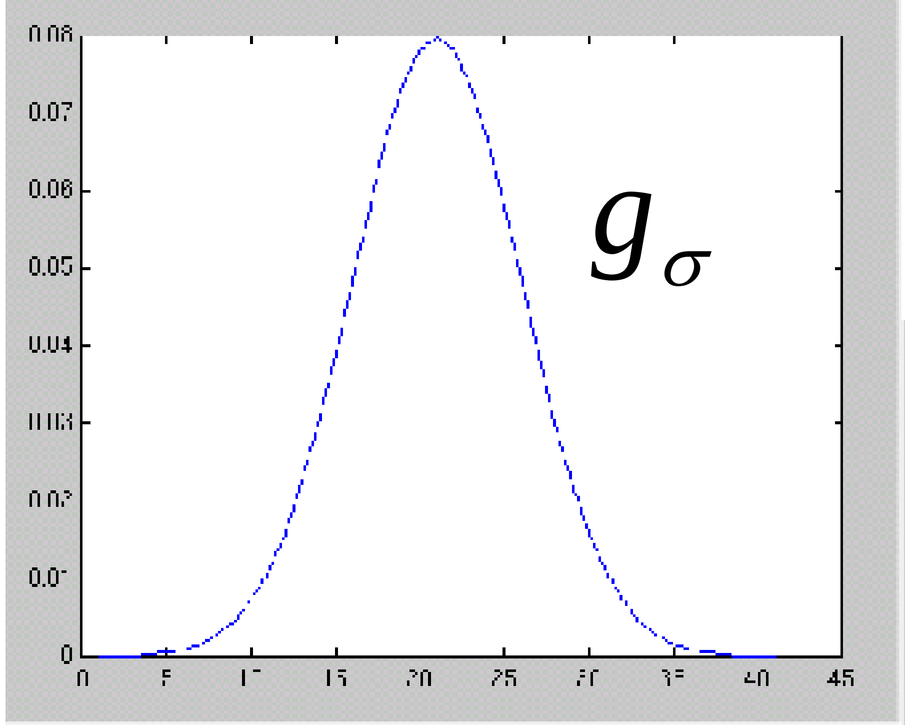
$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

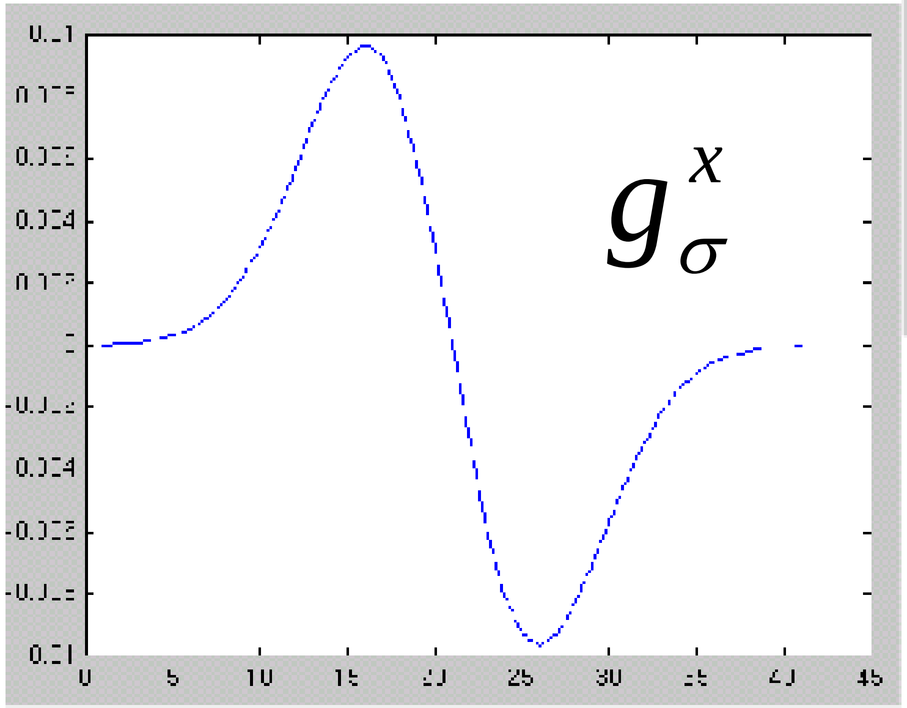
- Therefore, taking the derivative in x of the image can be done by convolution with the derivative of a Gaussian:

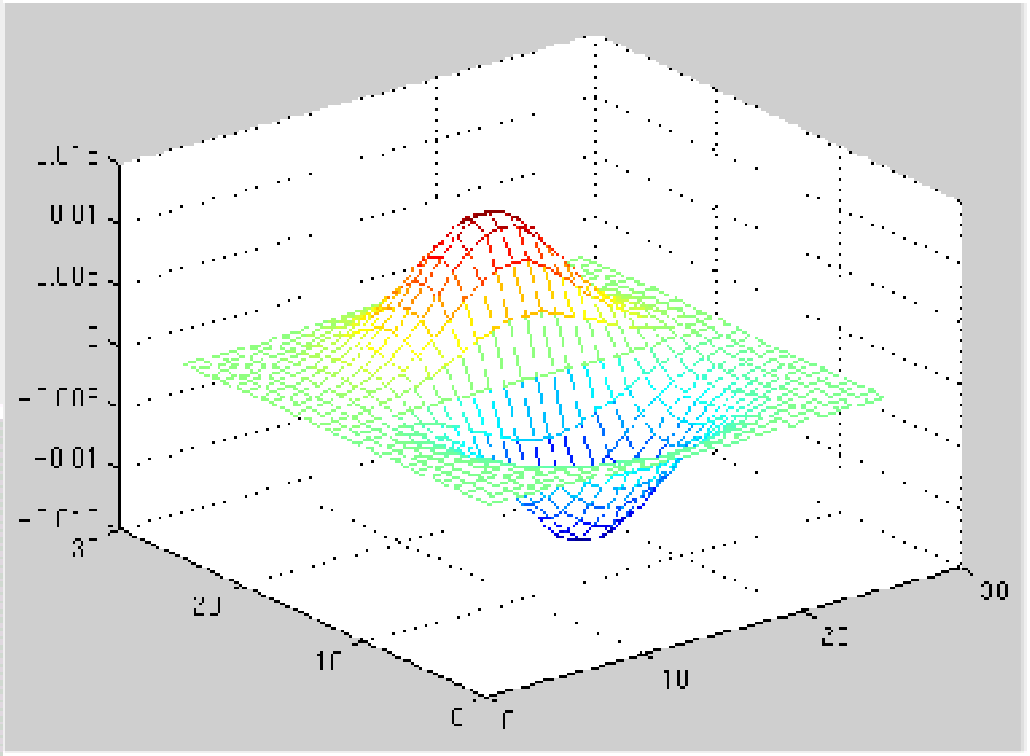
$$G_\sigma^x = \frac{\partial G_\sigma}{\partial x} = xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Crucial property: The Gaussian derivative is also separable:

$$G_\sigma^x * f = g_\sigma^x * g_{\sigma\uparrow} * f$$

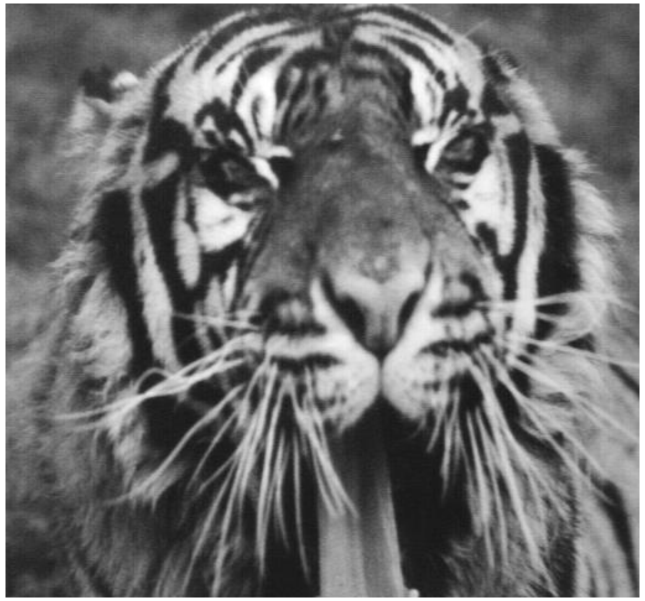

g_σ

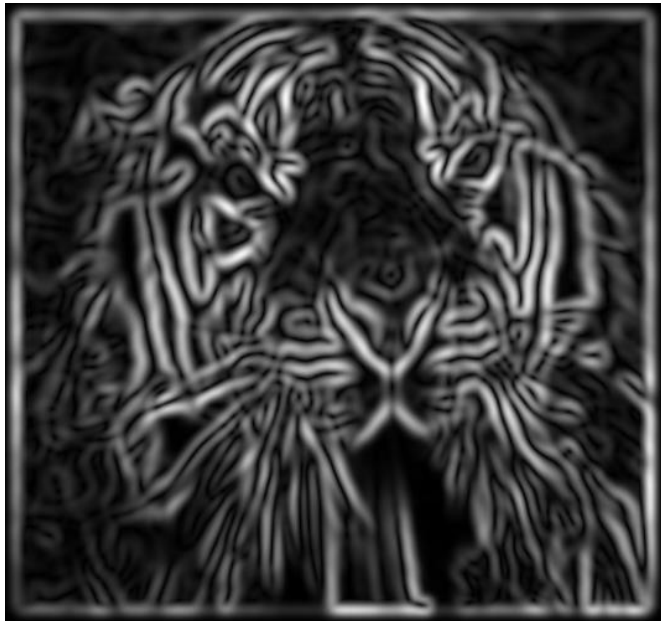

g_σ^x



Applying the first derivative of Gaussian

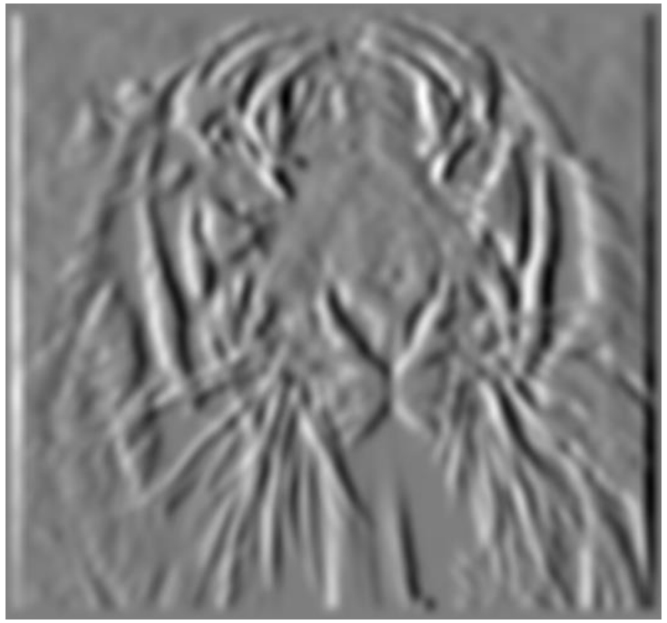
I

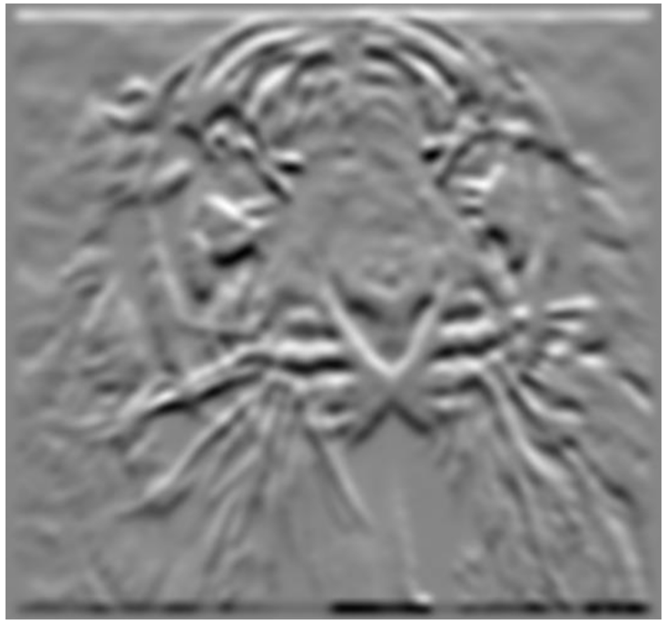




$$|\nabla I| = \sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$$

$$\frac{\partial I}{\partial x}$$

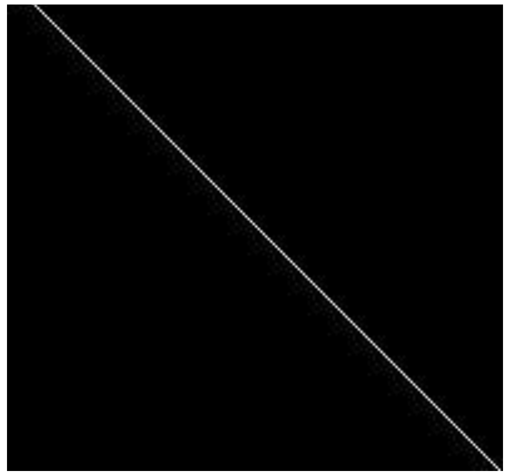




$$\frac{\partial I}{\partial y}$$

There is ALWAYS a tradeoff between smoothing and good edge localization!

Image with Edge


Edge Location

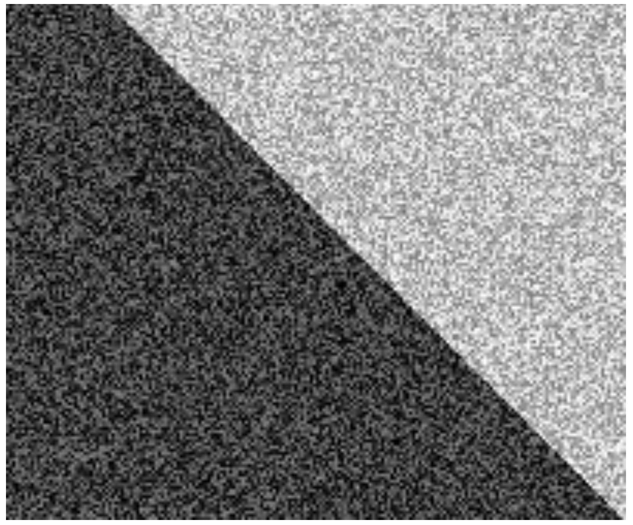
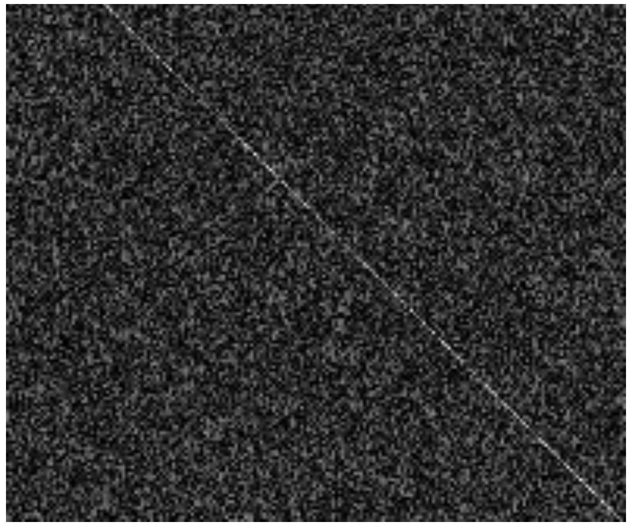
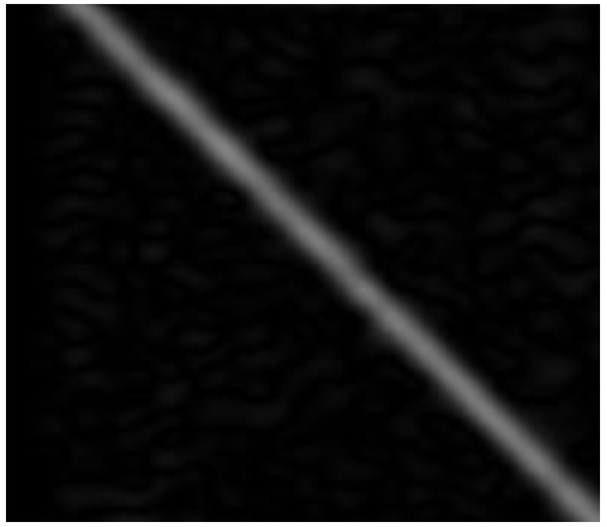
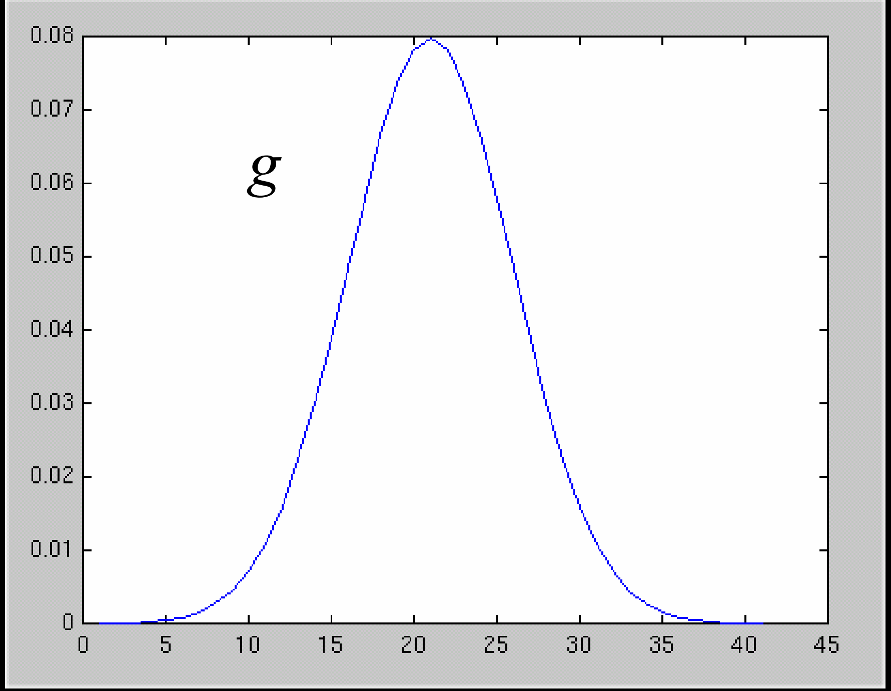


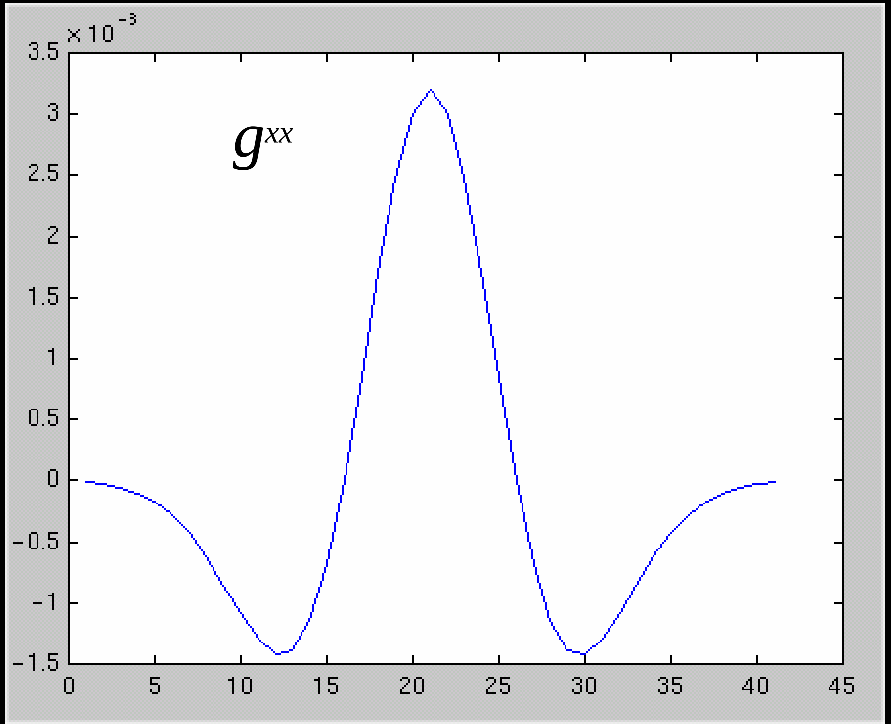
Image + Noise

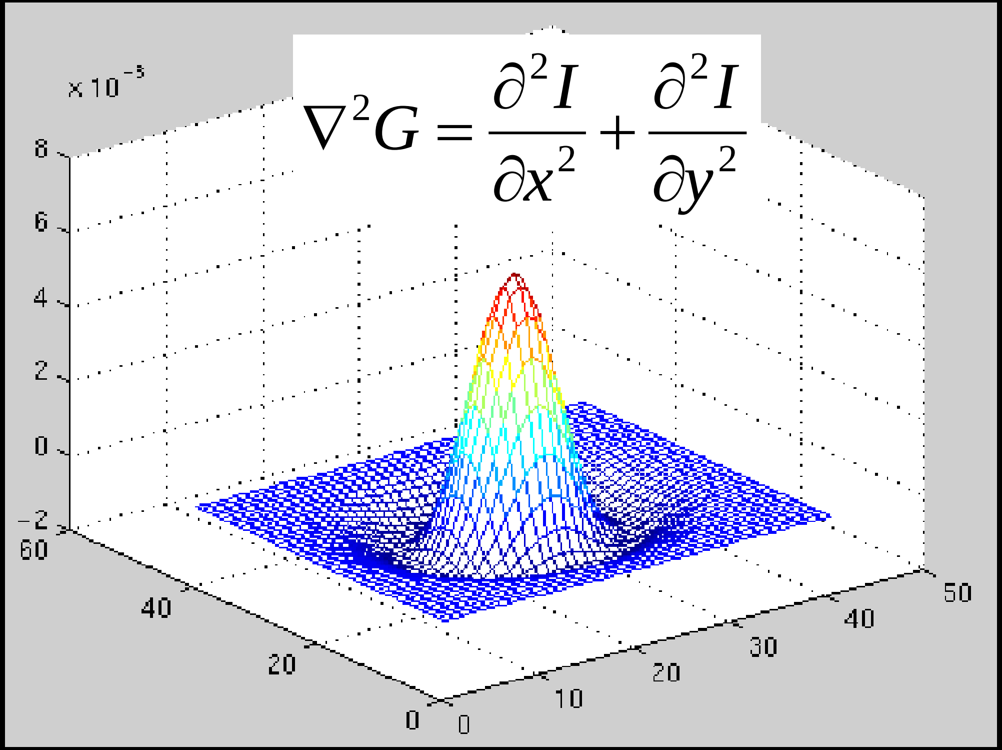

Derivatives detect edge *and* noise


Smoothed derivative removes noise, but blurs edge

Second derivatives: Laplacian



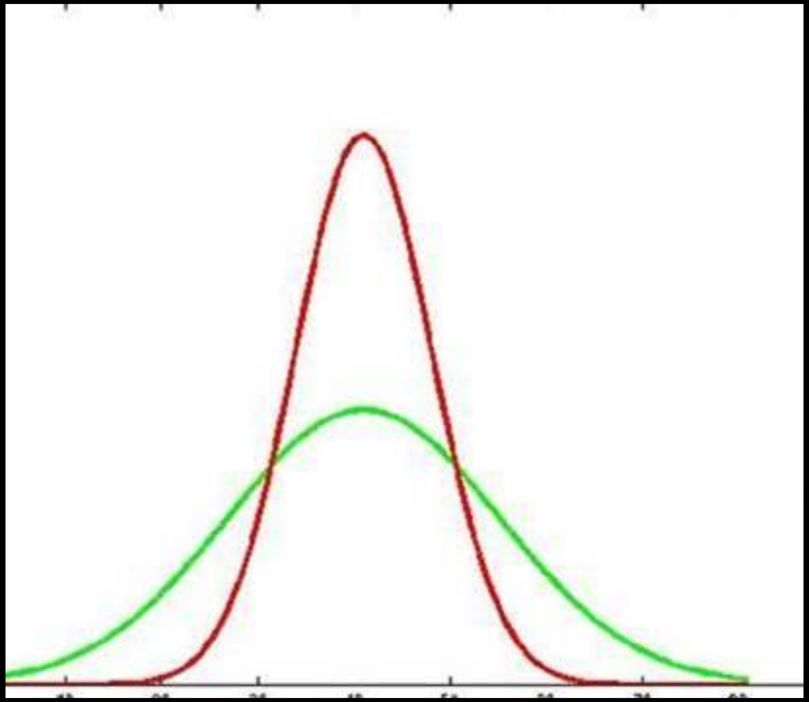


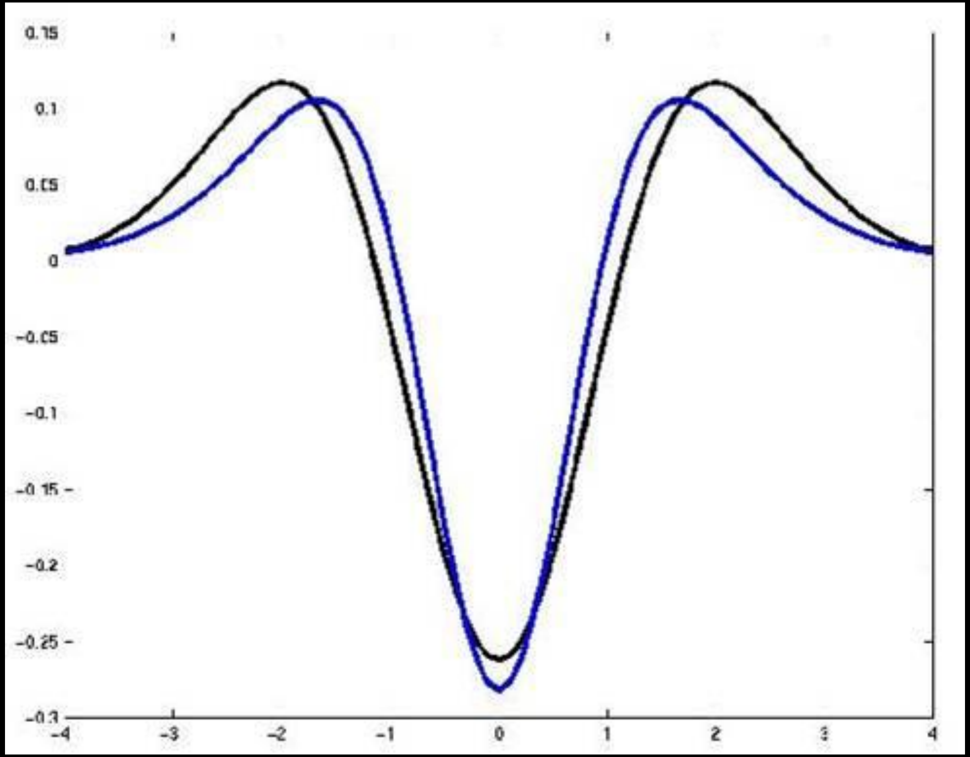


$$\nabla^2 G = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

DOG Approximation to LOG

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

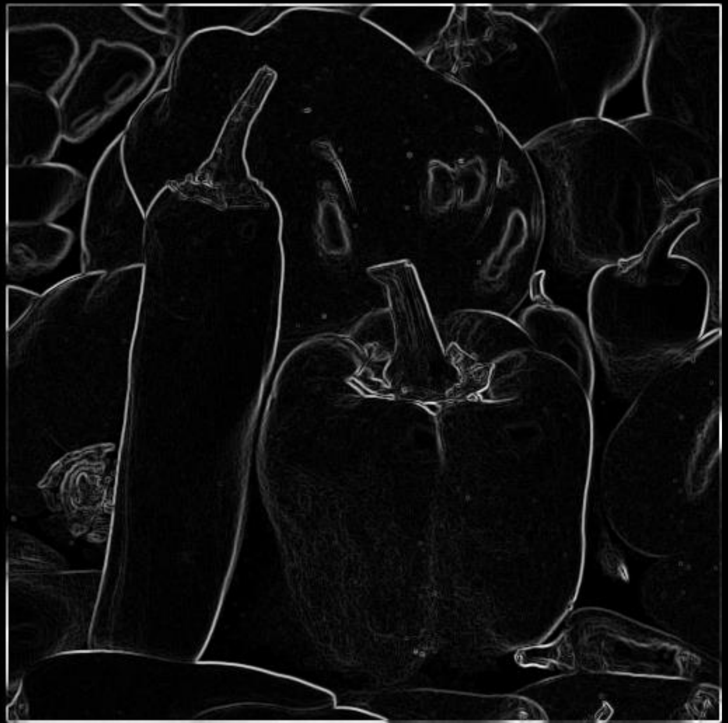
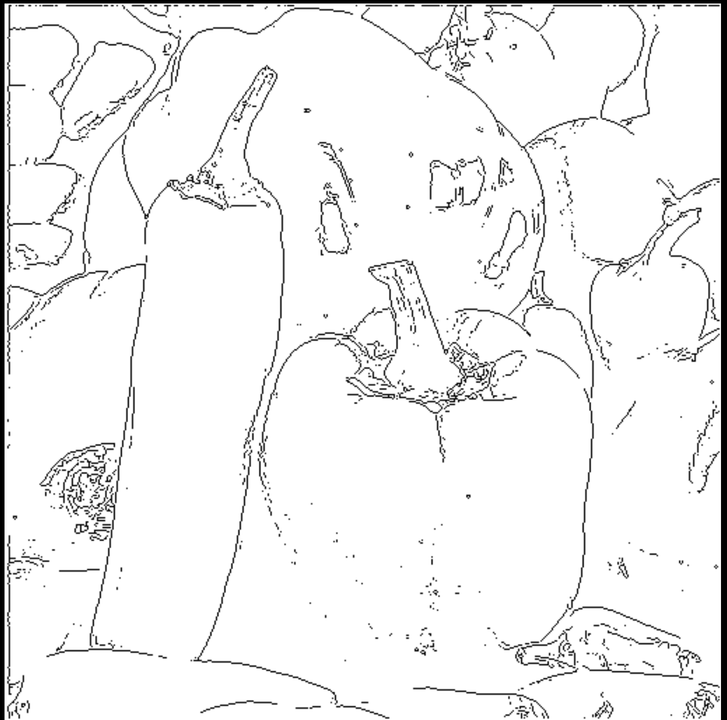




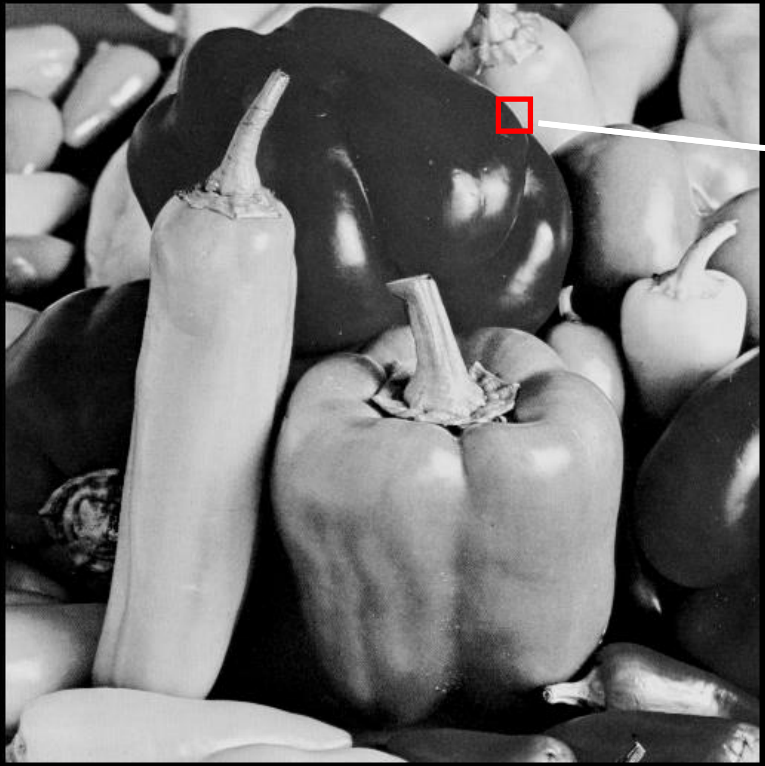
Edge Detection

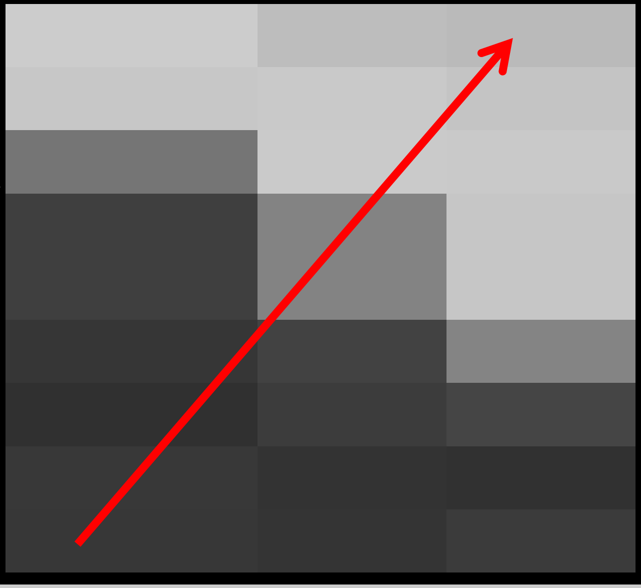
Edge Detection

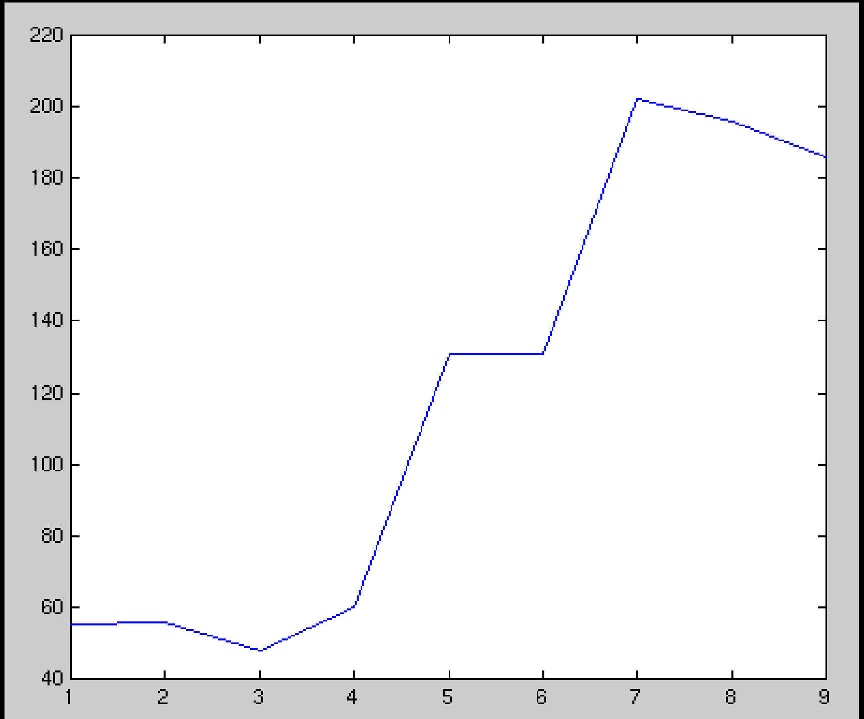
- Gradient operators
- Canny edge detectors
- Laplacian detectors

What is an edge?

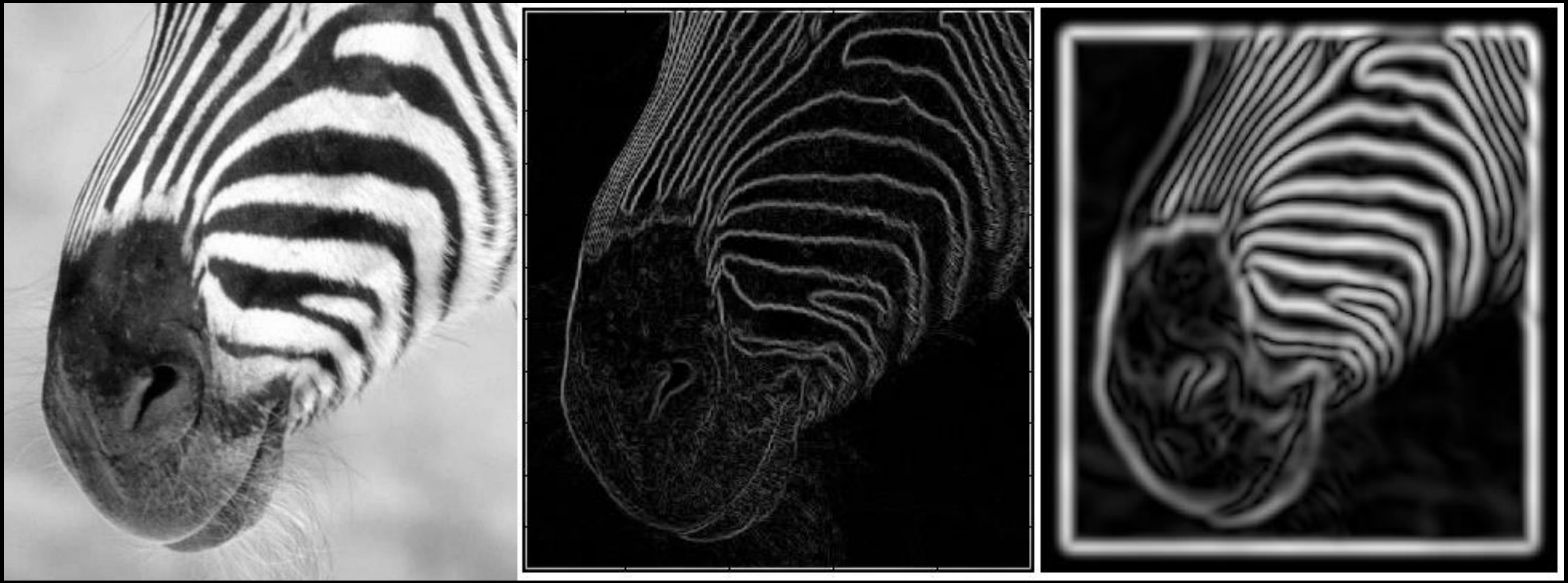






Edge = discontinuity of intensity in some direction.
Could be detected by looking for places where the derivatives of the image have large values.

Gradient-based edge detection



There are three major issues:

1) The gradient magnitudes at different scales are different; which one should we choose?
2) The gradient magnitude is large along thick trails; how do we identify the significant points?
3) How do we link the relevant points up into curves?

The Laplacian of Gaussian (Marr-Hildreth 80)

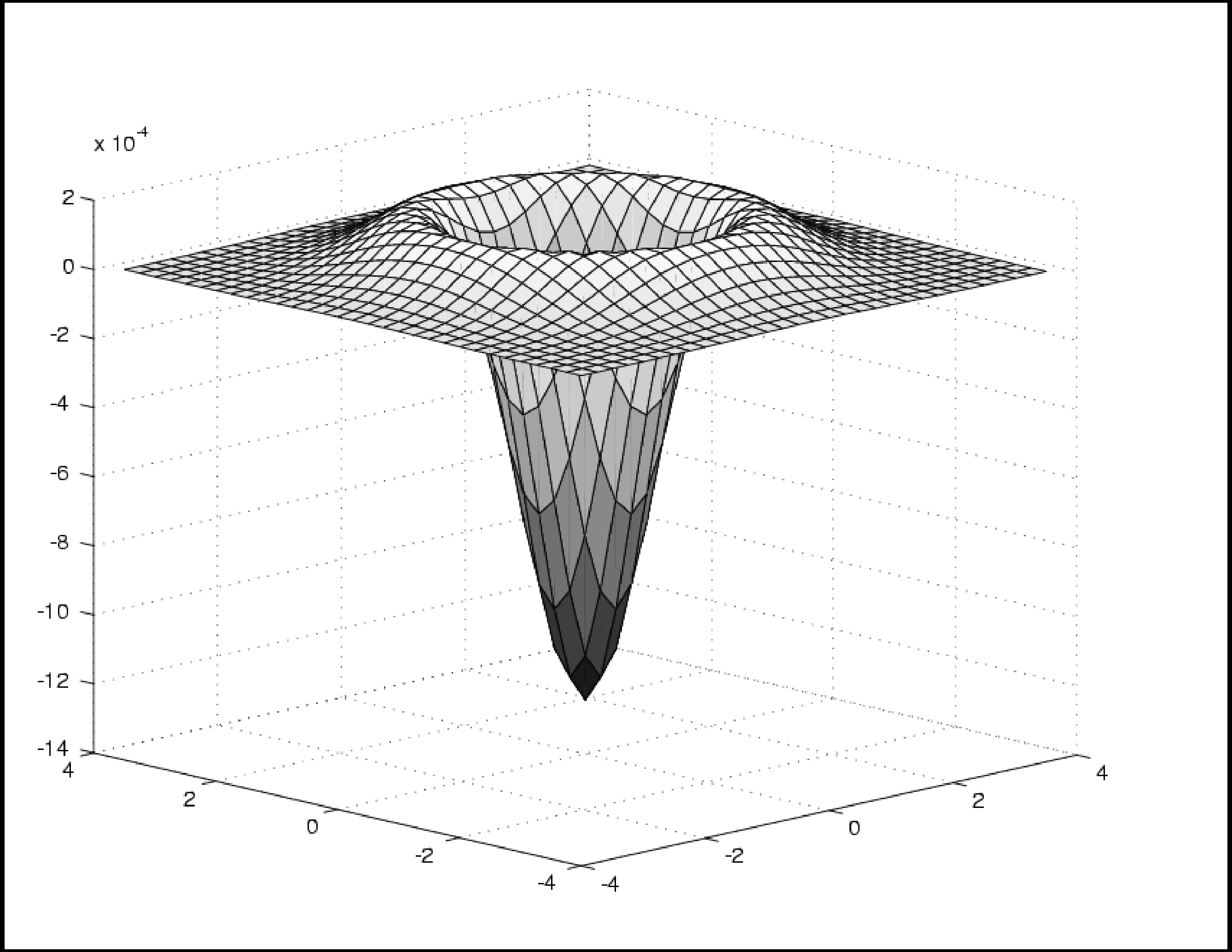
- Another way to detect an extremal first derivative is to look for a zero second derivative.

- Appropriate 2D analogy is rotation invariant:
  - the Laplacian

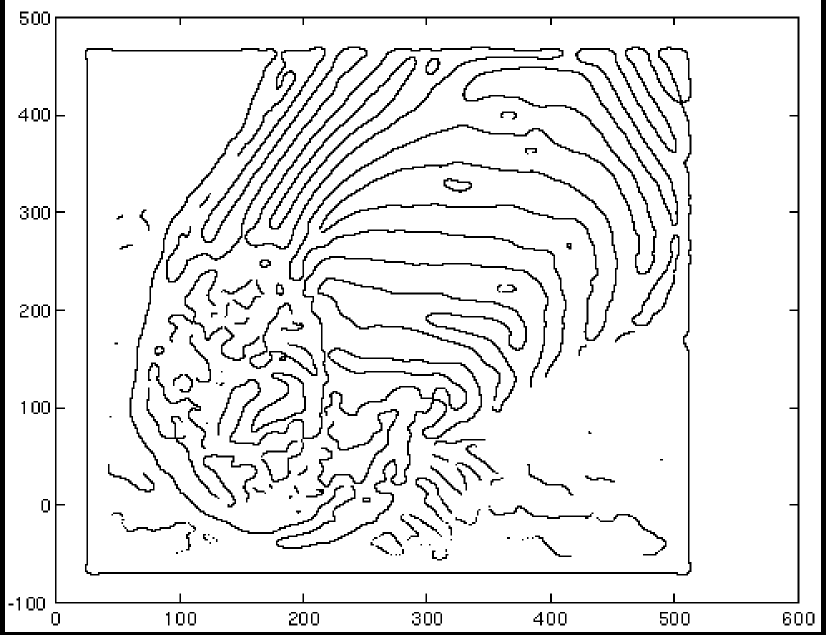
$$r^2 f=\partial^2 f/\partial x^2+\partial^2 f/\partial y^2$$

- Bad idea to apply a Laplacian without smoothing:
  - Smooth with Gaussian, apply Laplacian.
  - This is the same as filtering with a Laplacian of Gaussian filter.

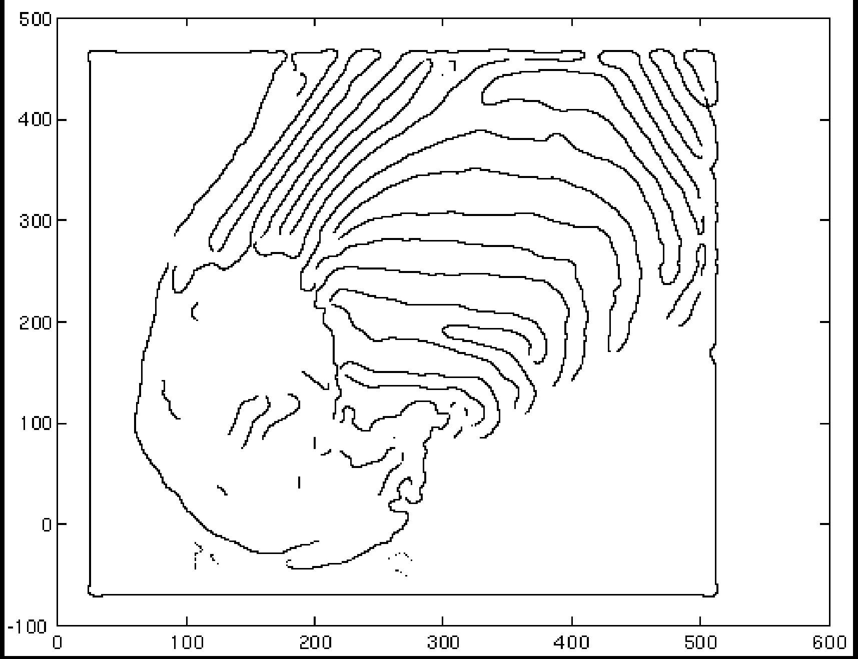
- Now mark the zero points where there is a sufficiently large derivative, and enough contrast.



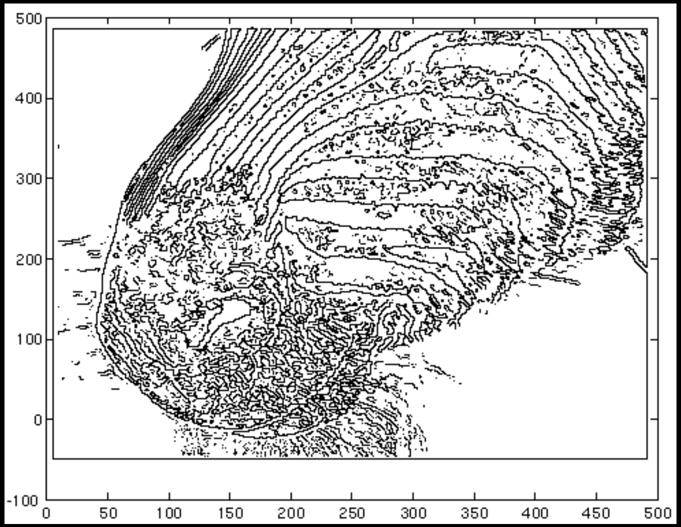
The Laplacian of a Gaussian



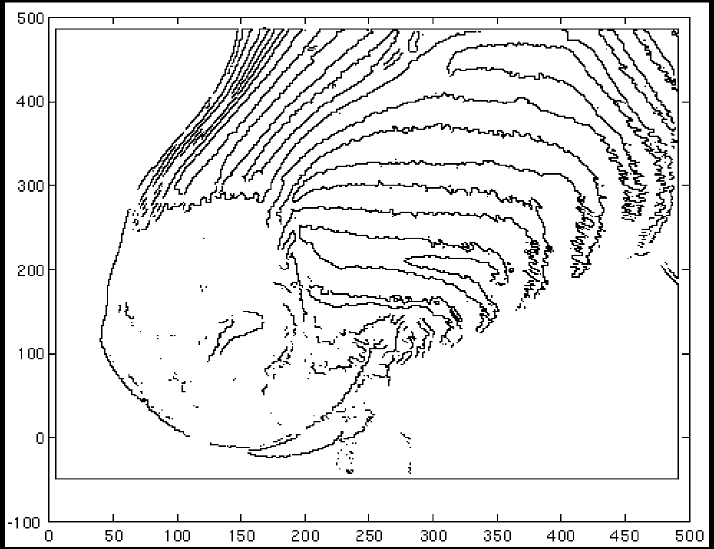
sigma=4

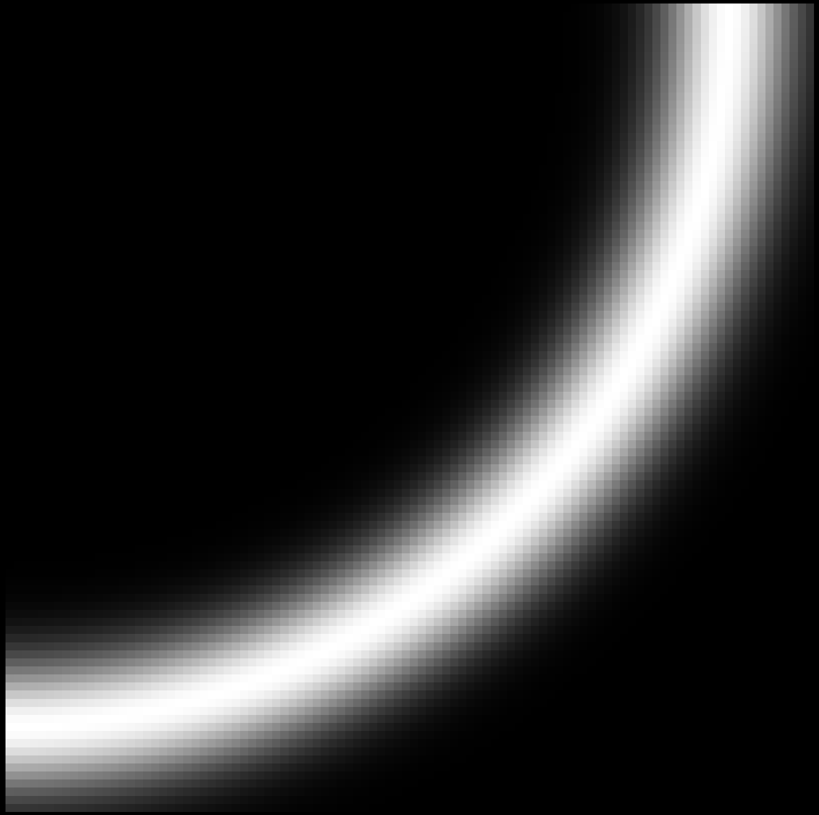
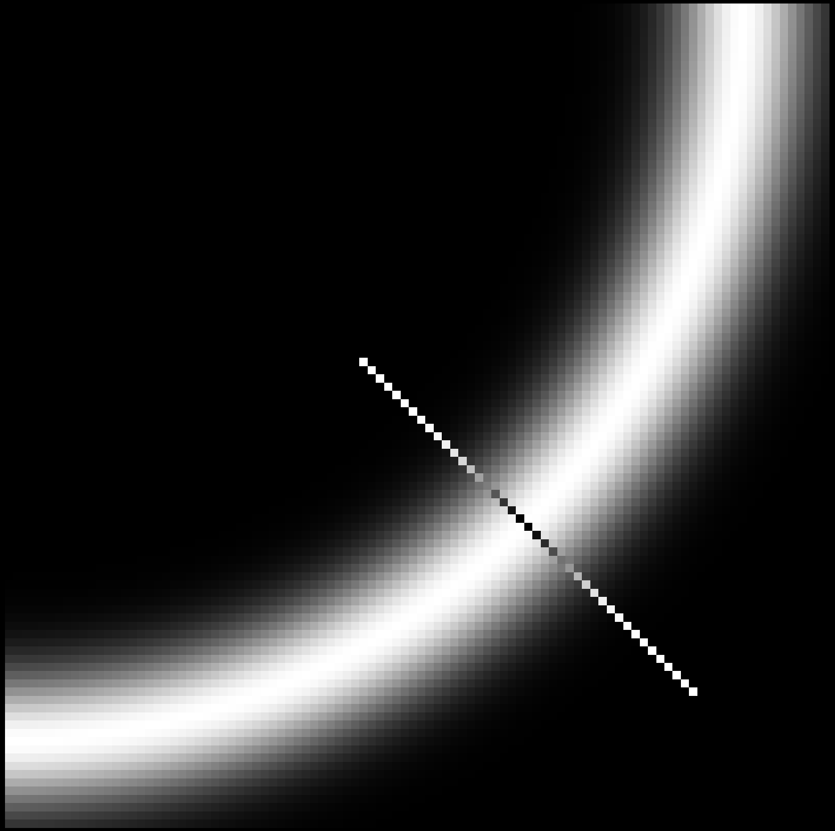


contrast=1 LOG zero crossings contrast=4



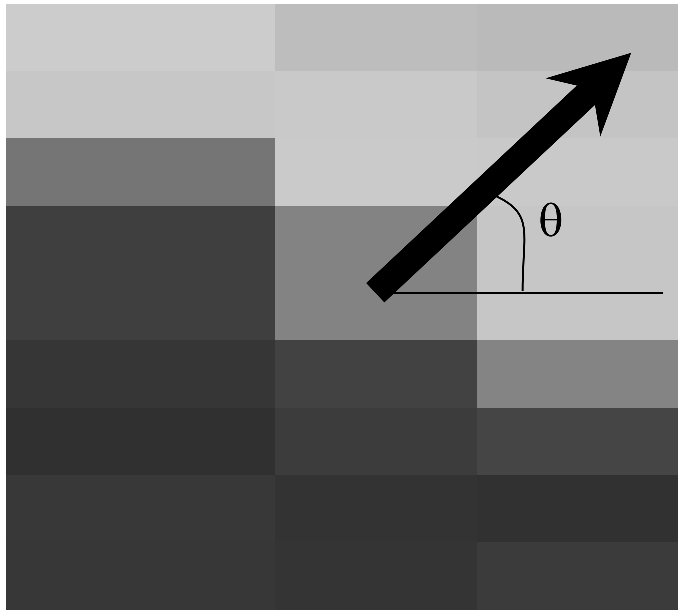
sigma=2

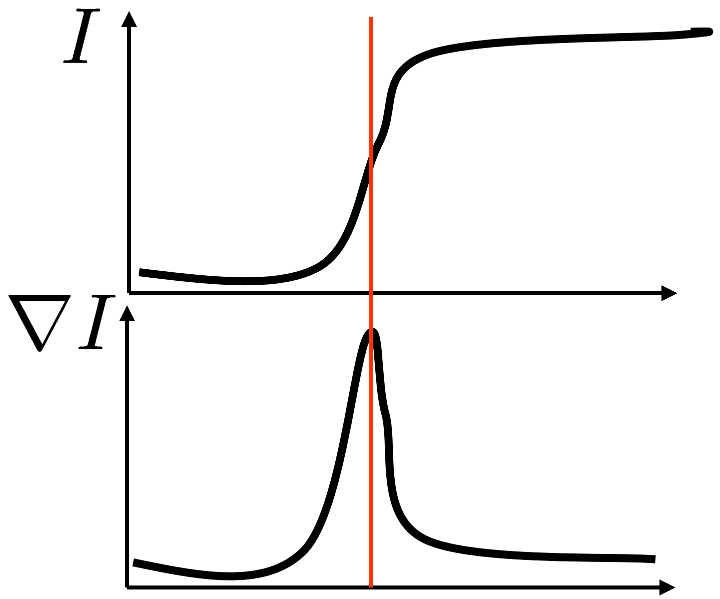


Gradient magnitude along an idealized curved edge.

Curved edges are locally straight: The gradient is orthogonal to the edge direction.

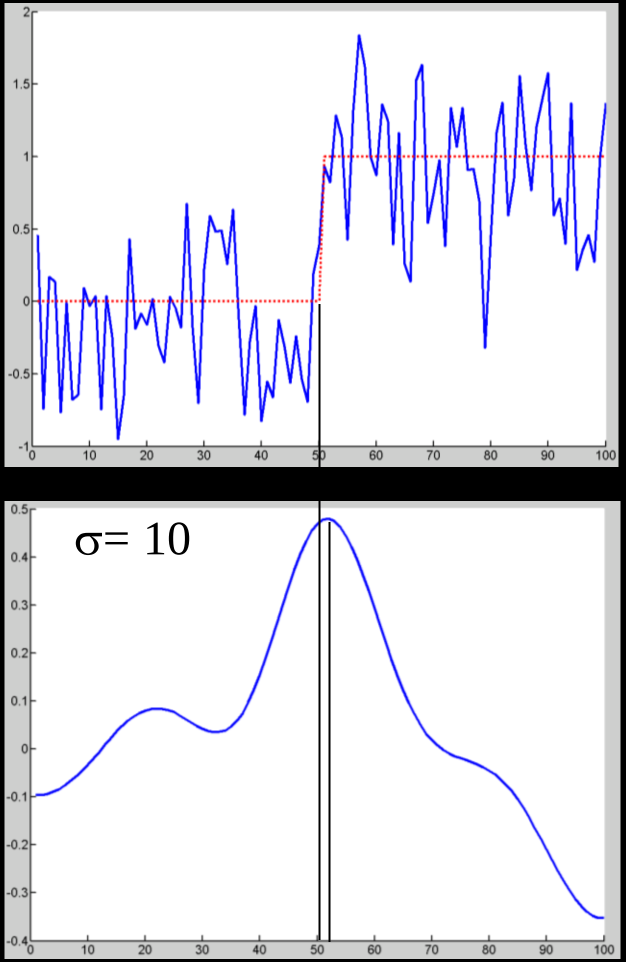


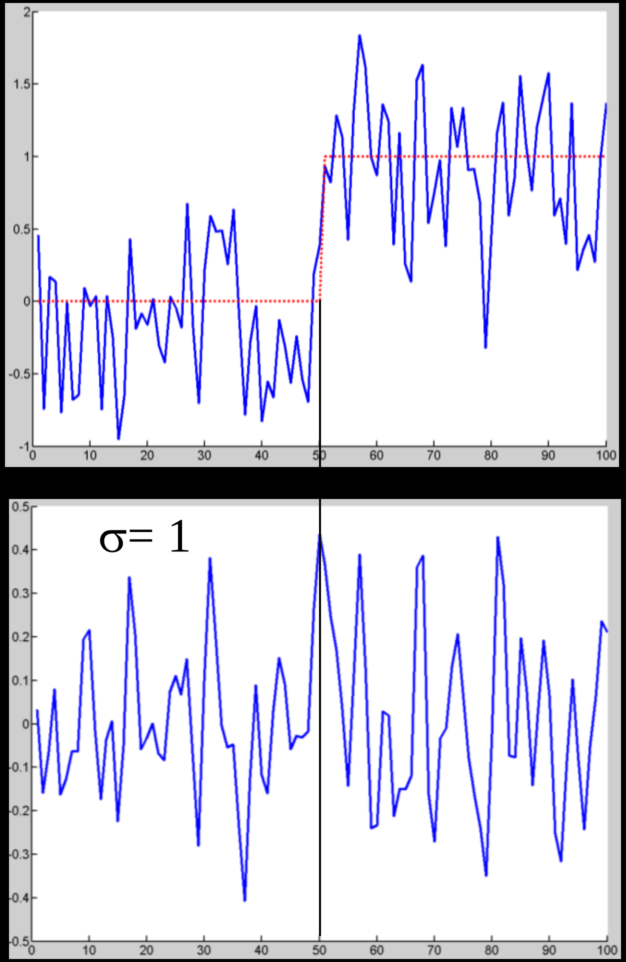


Edge pixels are at local maxima of gradient magnitude
Gradient computed by convolution with Gaussian derivatives
Gradient direction is always perpendicular to edge direction

$$\frac{\partial I}{\partial x} = G_\sigma^x * I \qquad \frac{\partial I}{\partial y} = G_\sigma^y * I$$

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \quad \theta = atan2(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x})$$





Large σ → Good detection (high SNR)
Poor localization

Small σ → Poor detection (low SNR)
Good localization

Canny's Result

- Given a filter *f*, define the two objective functions:

  $\Lambda(f)$ large if *f* produces good localization

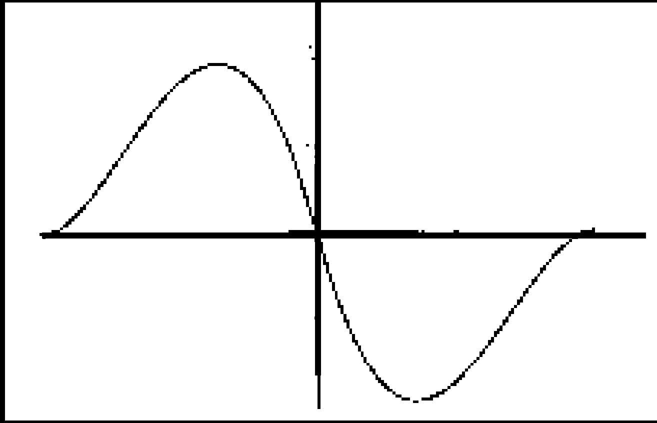
  $\Sigma(f)$ large if *f* produces good detection (high SNR)

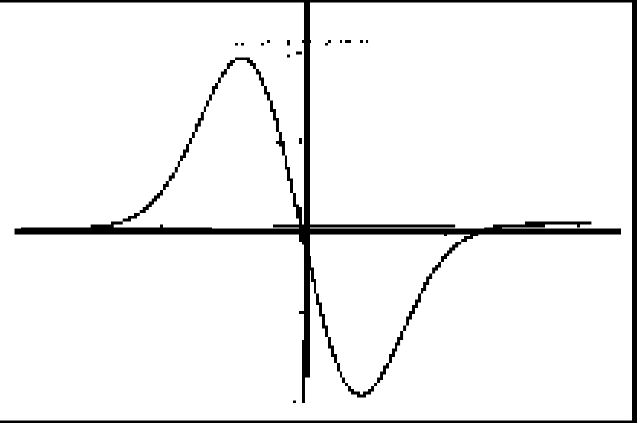
- Problem: Find a family of filters *f* that maximizes the compromise criterion

$$\Lambda(f)\Sigma(f)$$

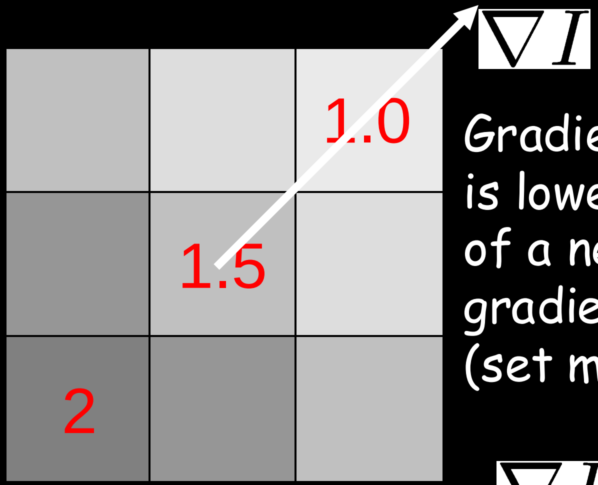
under the constraint that a single peak is generated by a step edge

- Solution: Unique solution, a close approximation is the Gaussian derivative filter!

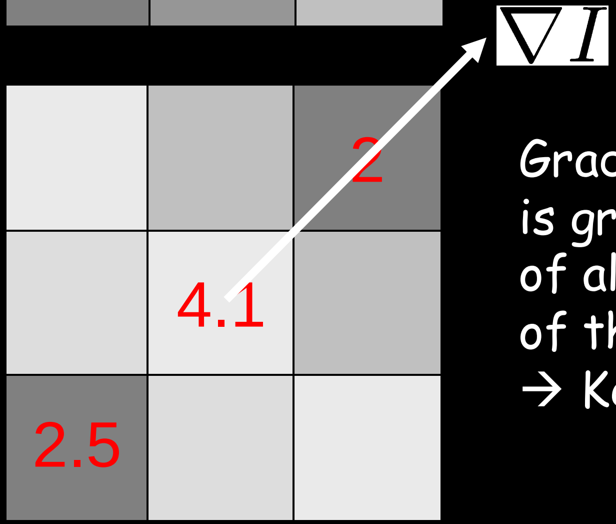

Canny


Derivative of Gaussian

Non-Local Maxima Suppression

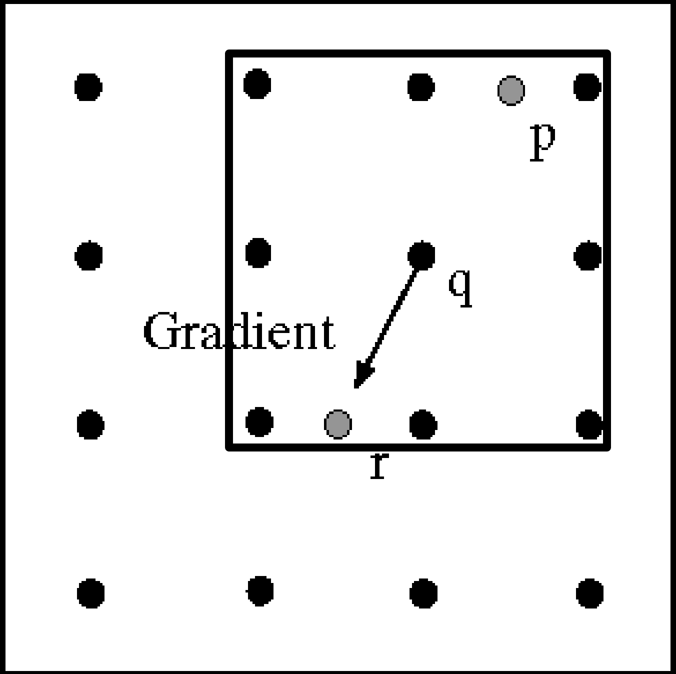


Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor in the direction of the gradient → Discard center pixel (set magnitude to 0)

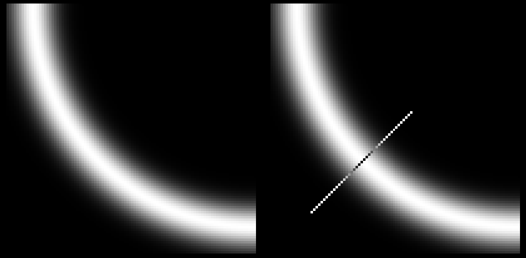


Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors in the direction of the gradient
→ Keep center pixel unchanged

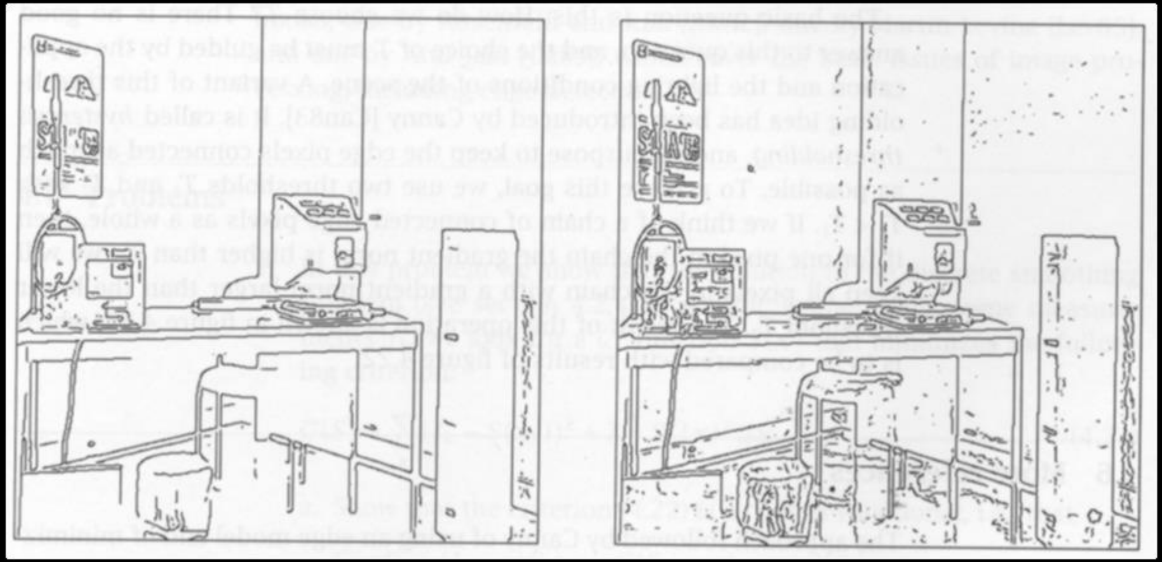
Non-maximum suppression



At q we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Input image

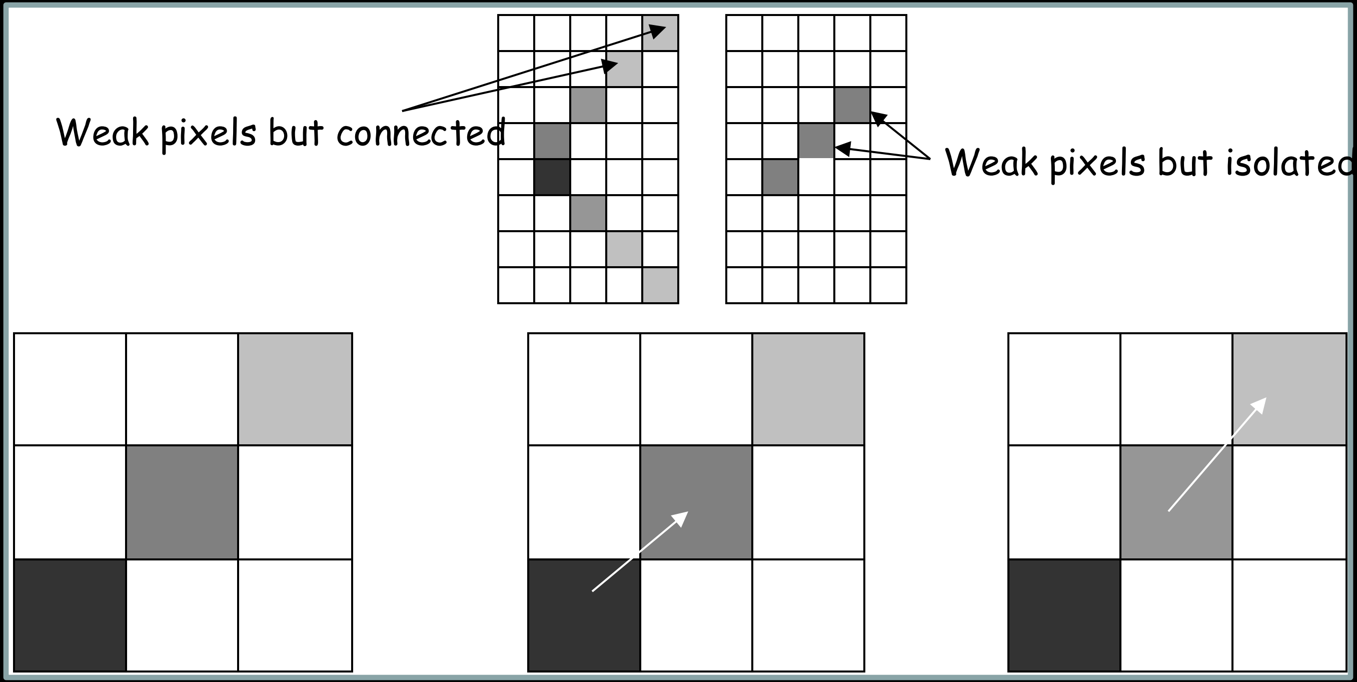


T = 15

T = 5

Two thresholds applied to gradient magnitude

Hysteresis Thresholding

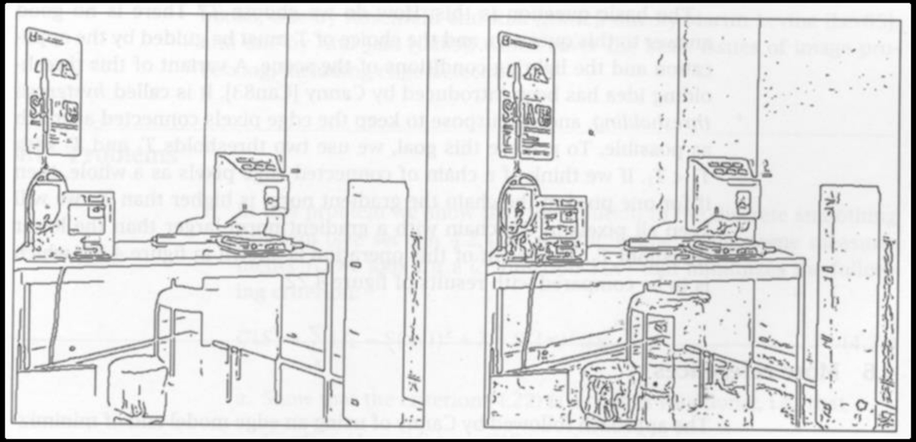


Very strong edge response.
Let's start here

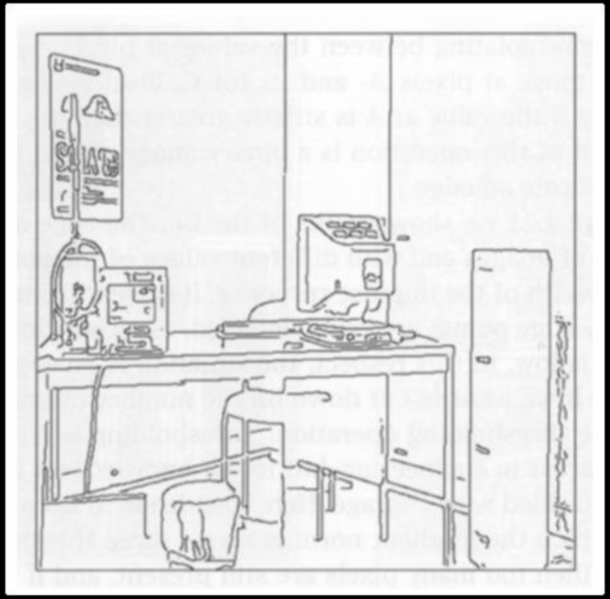
Weaker response but it is connected to a confirmed edge point. Let's keep it.

Continue…

T=15

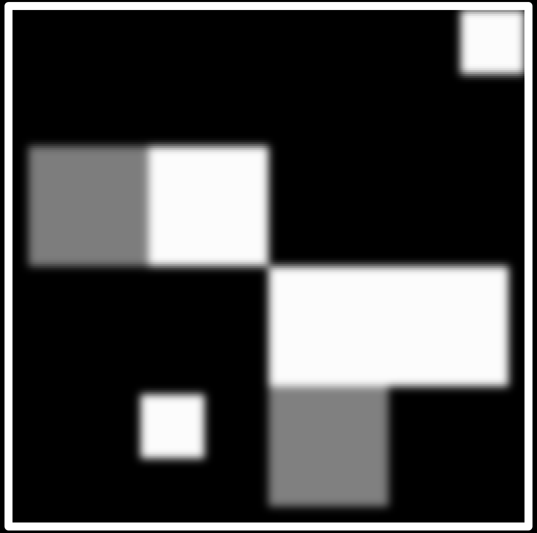


T=5

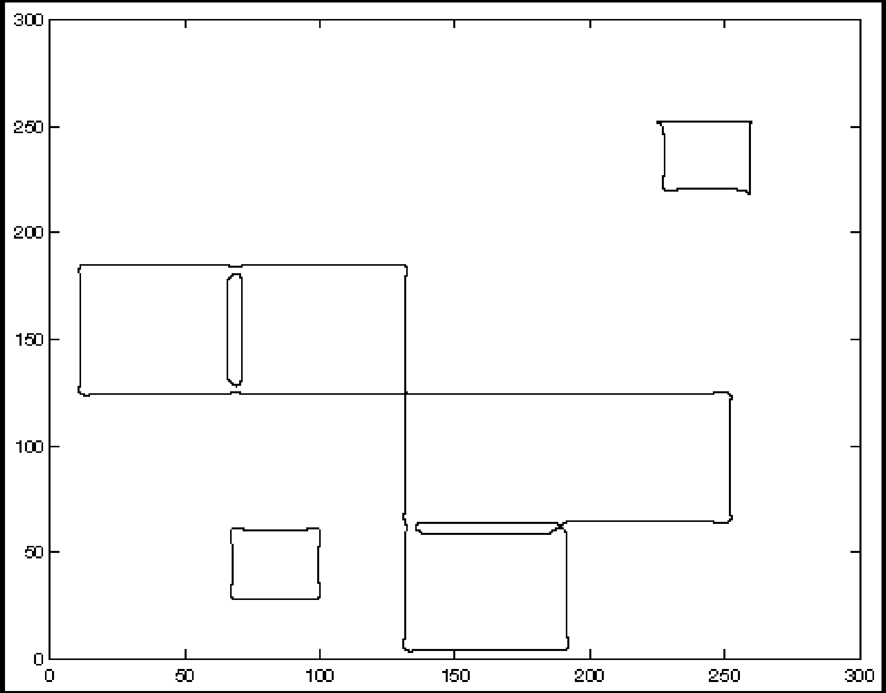


Hysteresis
T_h=15 T_l = 5

Hysteresis thresholding



We have unfortunate behaviour at corners



Why machine learning for image restoration?

Reasonable physical models of image corruption

- For example: $y=A(x)+\varepsilon$

- For example: $A(x) = k * x$

➢ One can use prior knowldege

- For example: sparsity, self similarities

➢ Realistic simulated training examples

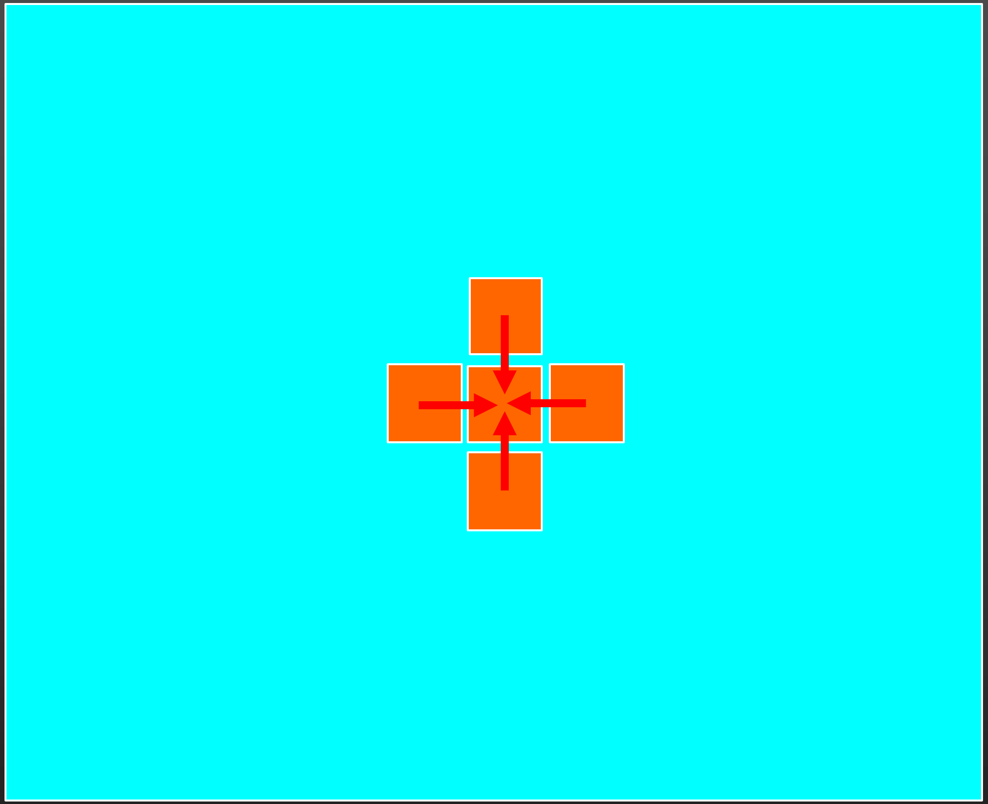
➢ Interpretable, “functional” architectures

Why machine learning for image restoration?

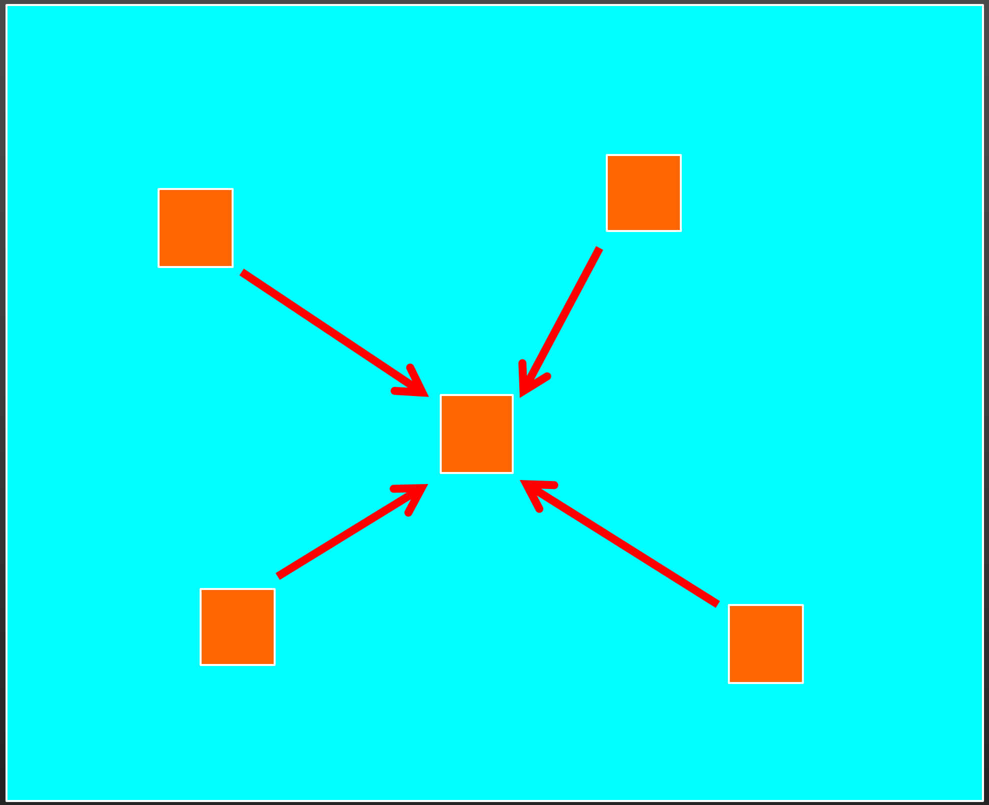
- Reasonable physical models of image corruption
  - For example: $y=A(x)+\varepsilon$
  - For example: $A(x) = k * x$
- One can use prior knowldege
  - For example: sparsity, self similarities
- Realistic simulated training examples
- Interpretable, “functional” architectures

But where does the real ground truth come from, whether for model-based or data-driven methods?

Let us start simple: How to denoise an image

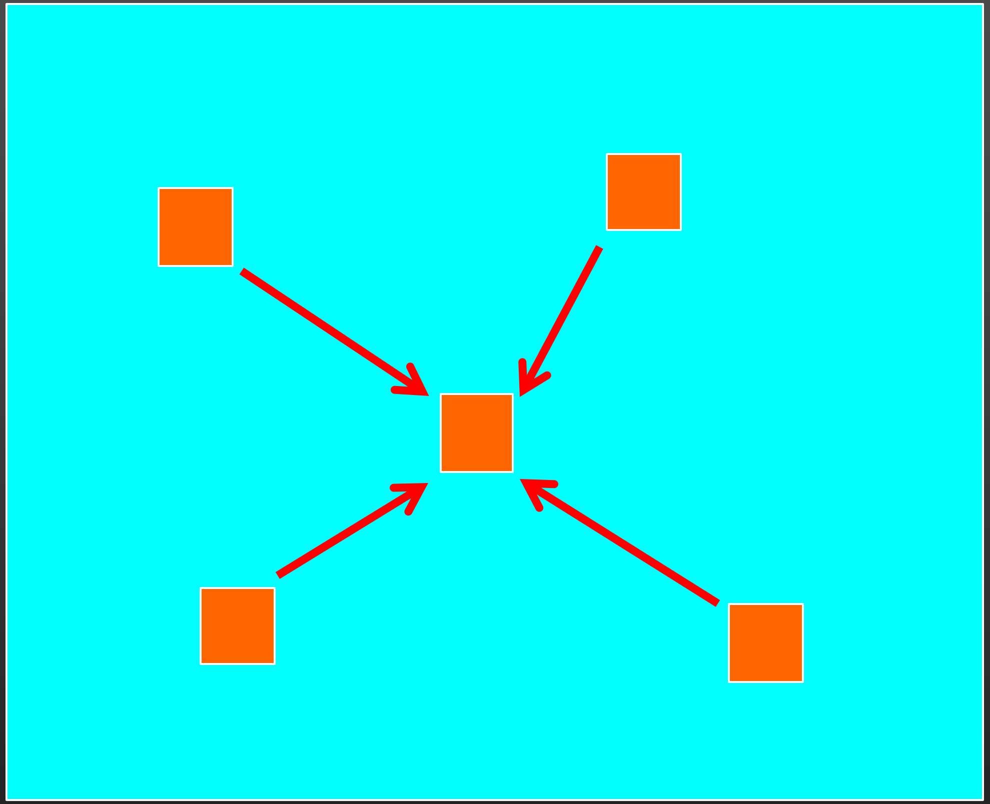


Let us start simple: How to denoise an image

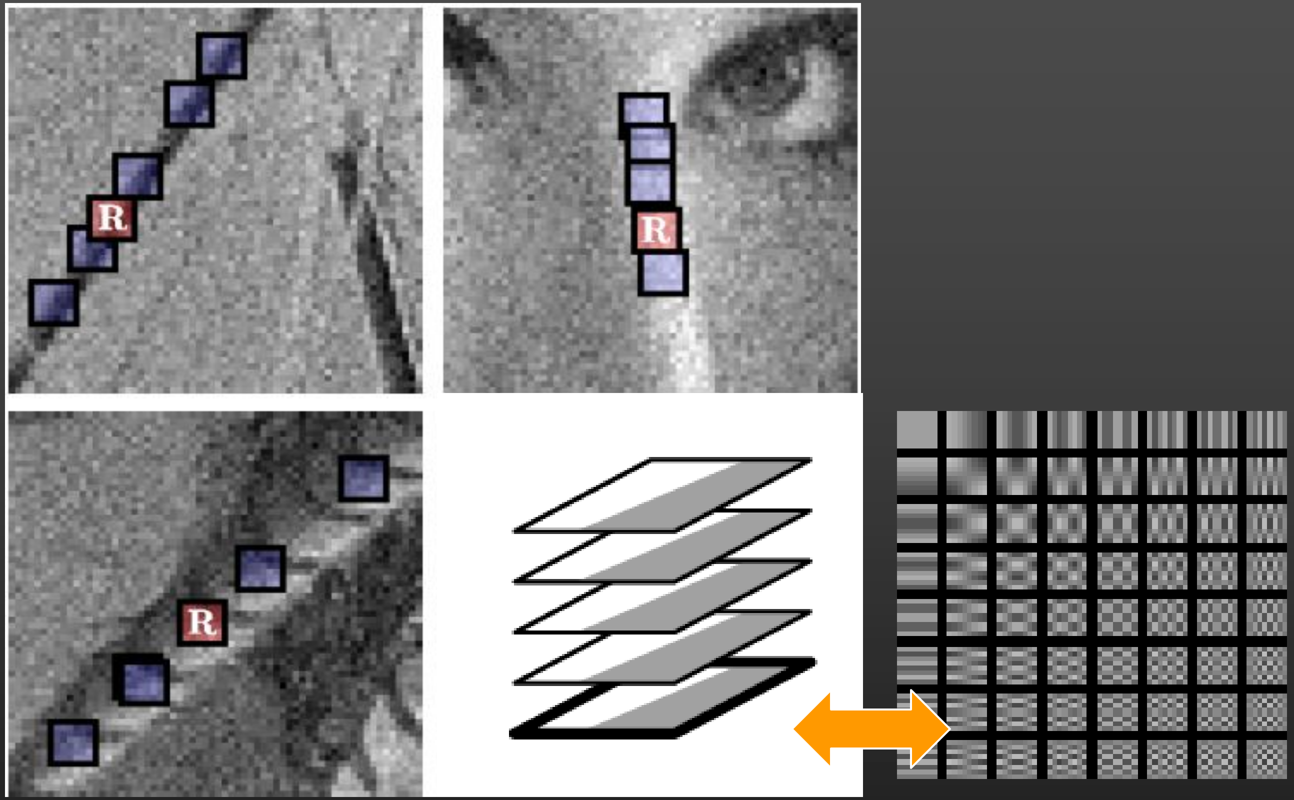


Non-local means filtering (Buades et al.'05)

Let us start simple: How to denoise an image



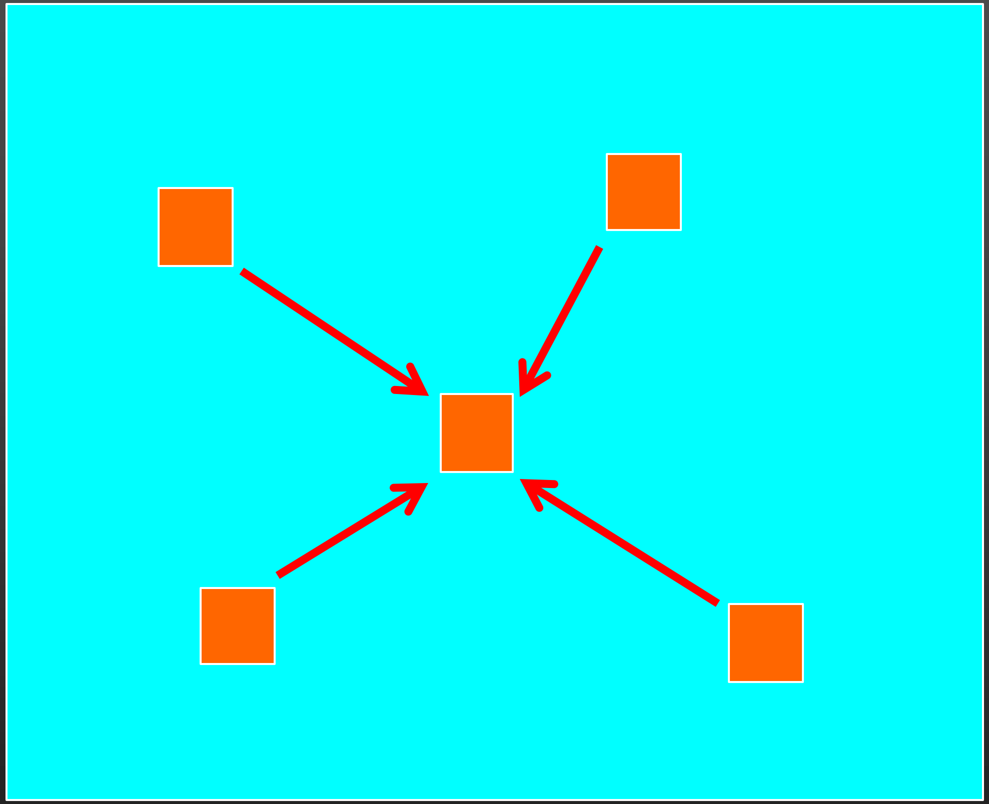
Non-local means filtering (Buades et al.'05)



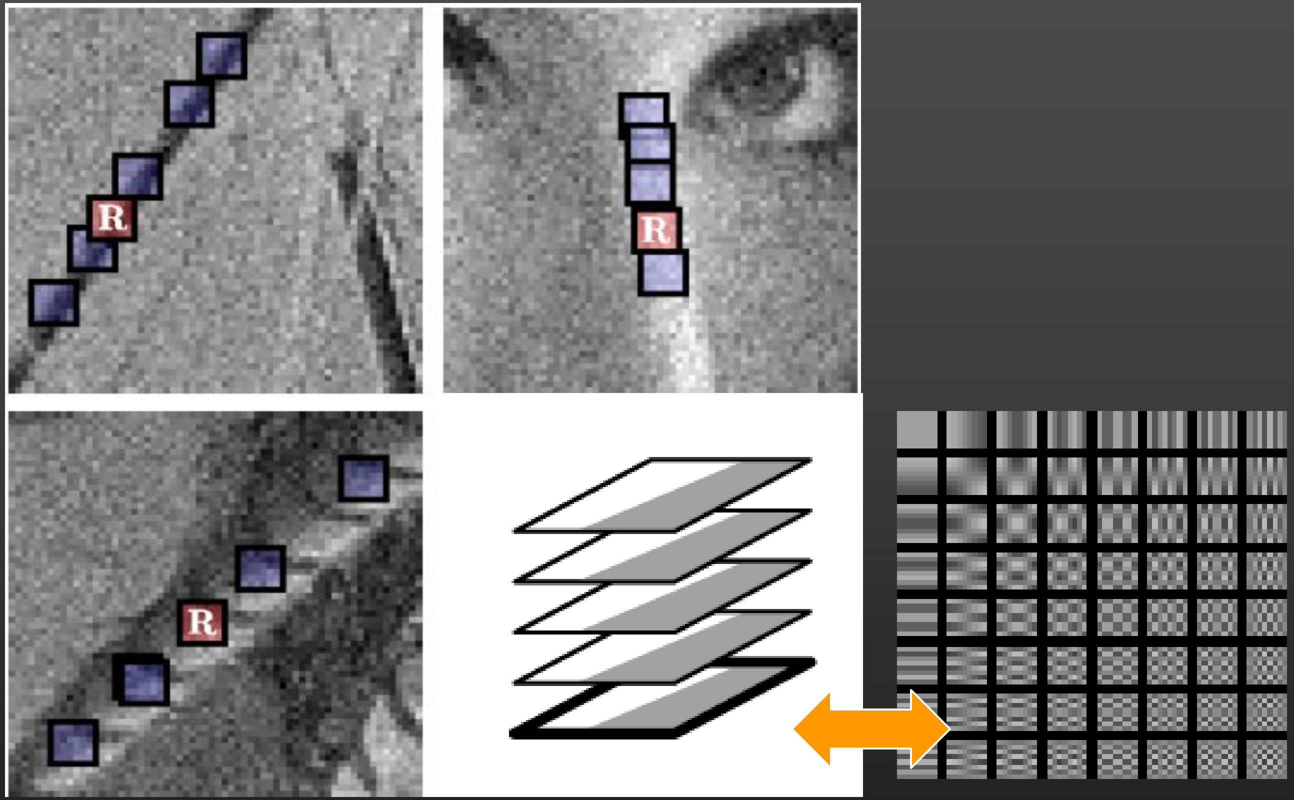
BM3D = Sparse representation on (3D) DCT dictionary + NLM (Dabov et al.'07)

Observation: natural image patches can be sparsely represented as linear combinations of a few elements of appropriate dictionaries, e.g., discrete cosine transform basis functions (e.g., Olshausen and Field, 1997; Chen et al., 1999; Mallat, 1999).

Let us start simple: How to denoise an image



Non-local means filtering (Buades et al.'05)



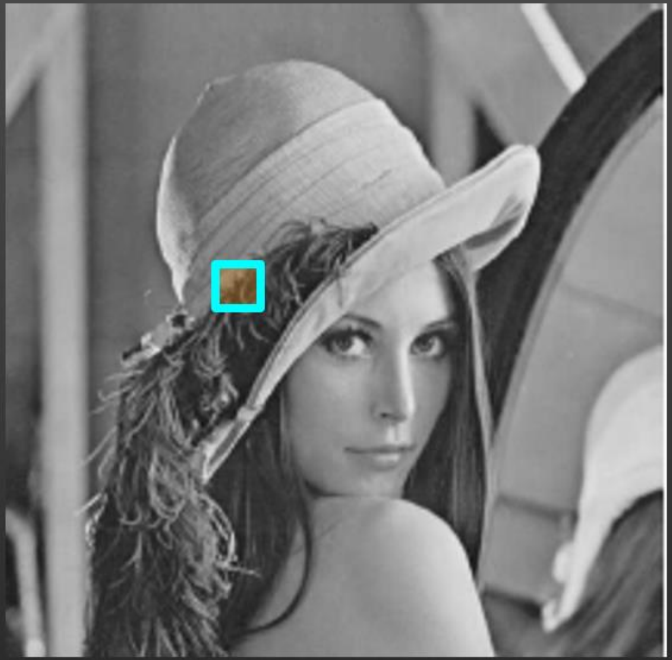
BM3D = Sparse representation on (3D) DCT dictionary + NLM (Dabov et al.'07)

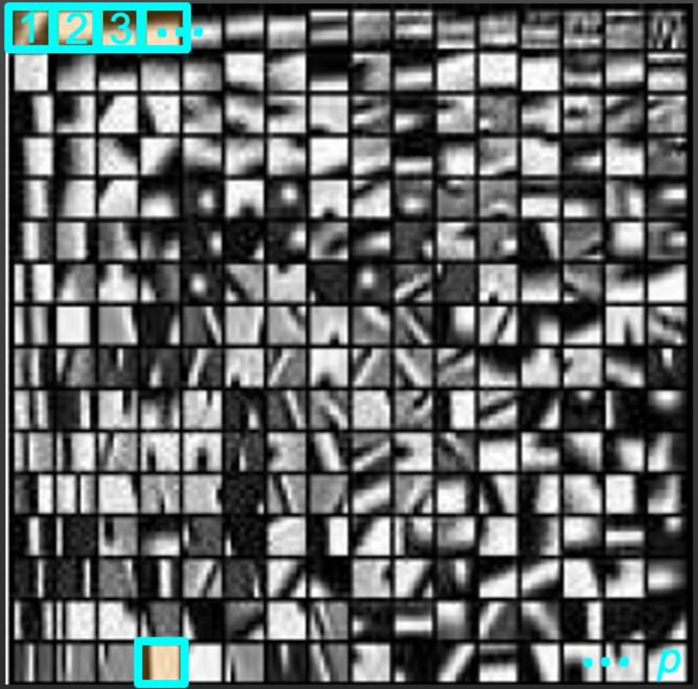
Take $x \approx \sum_j \alpha^j d^j = D\alpha$ but limit the number of nonzero coefficients $||\alpha||_0 \leq k$

Linear signal models

Signal: $x \in \mathbb{R}^m$

Dictionary: $D=[d_1,\ldots,d_p] \in \mathbb{R}^{m \times p}$





$$x \approx \alpha_1 d_1 + \alpha_2 d_2 + \ldots + \alpha_p d_p = D\alpha, \text{ with } \alpha \in \mathbb{R}^p$$

Note: ➢ In general $p \geq m$. Here $p=256$, $m=100$.

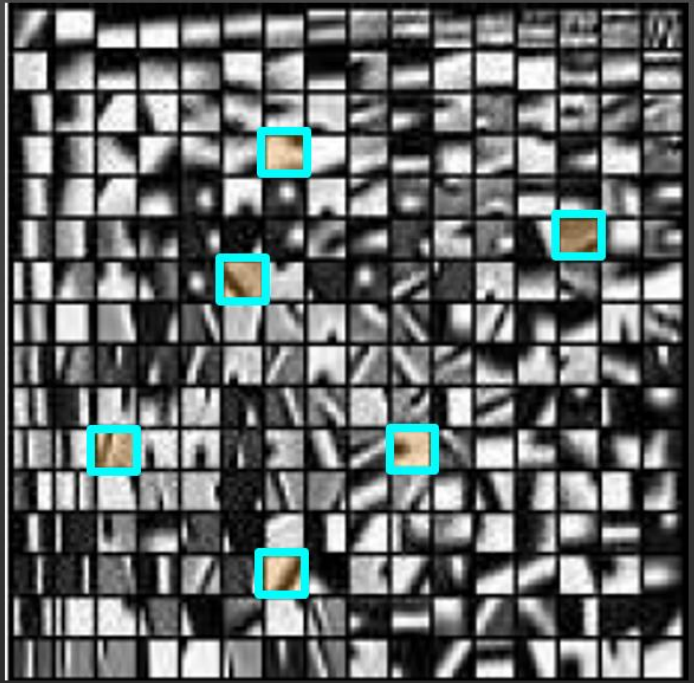
➢ The dictionary has no spatial structure

Sparse linear models

Signal: $x \in \mathbb{R}^m$

Dictionary:

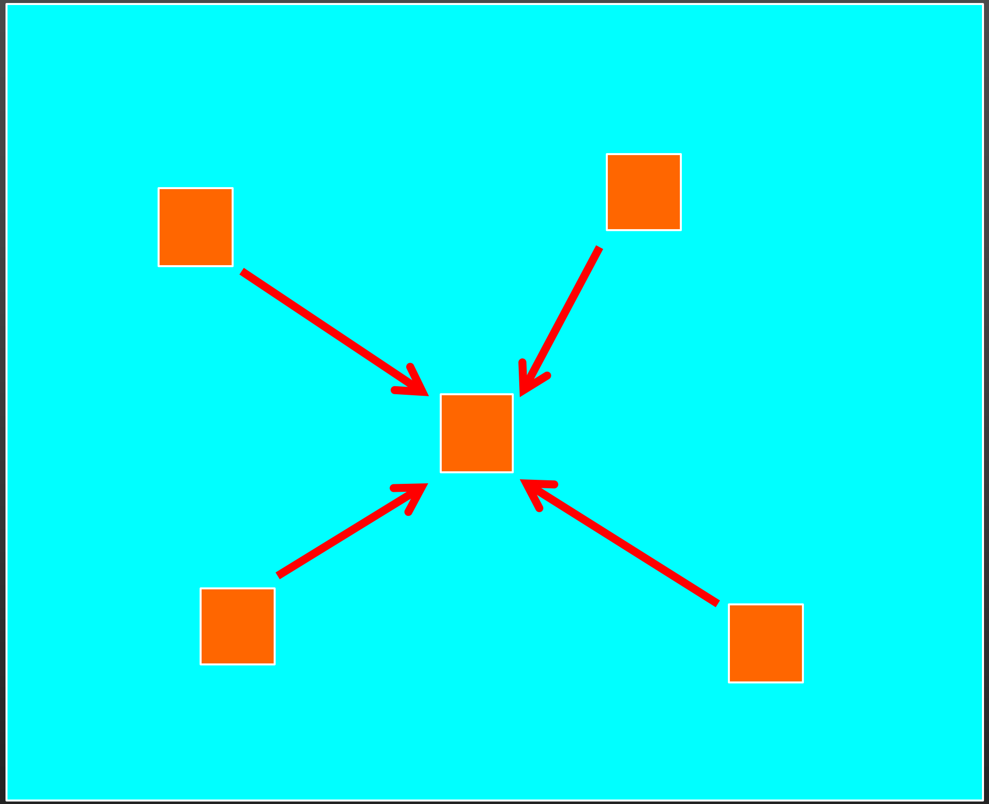
$D = [d_1, \ldots, d_p] \in \mathbb{R}^{m \times p}$



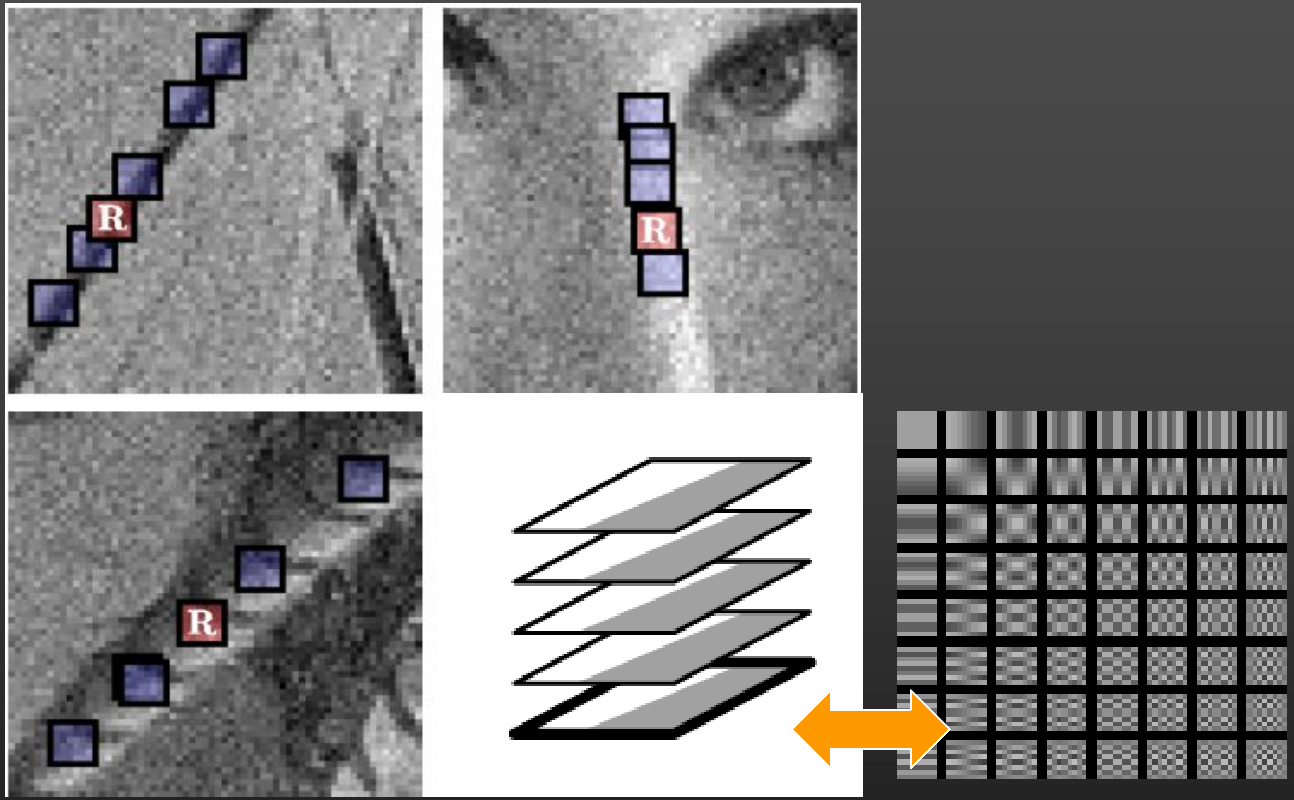
$$x \approx \alpha_1 d_1 + \alpha_2 d_2 + \ldots + \alpha_p d_p = D\alpha, \text{ with } |\alpha|_0 \ll p$$

(Olshausen and Field, 1997; Chen et al., 1999; Mallat, 1999; Elad and Aharon, 2006)
(Kavukcuoglu et al., 2009; Wright et al., 2009; Yang et al., 09; Boureau et al., 2010)

Let us start simple: How to denoise an image



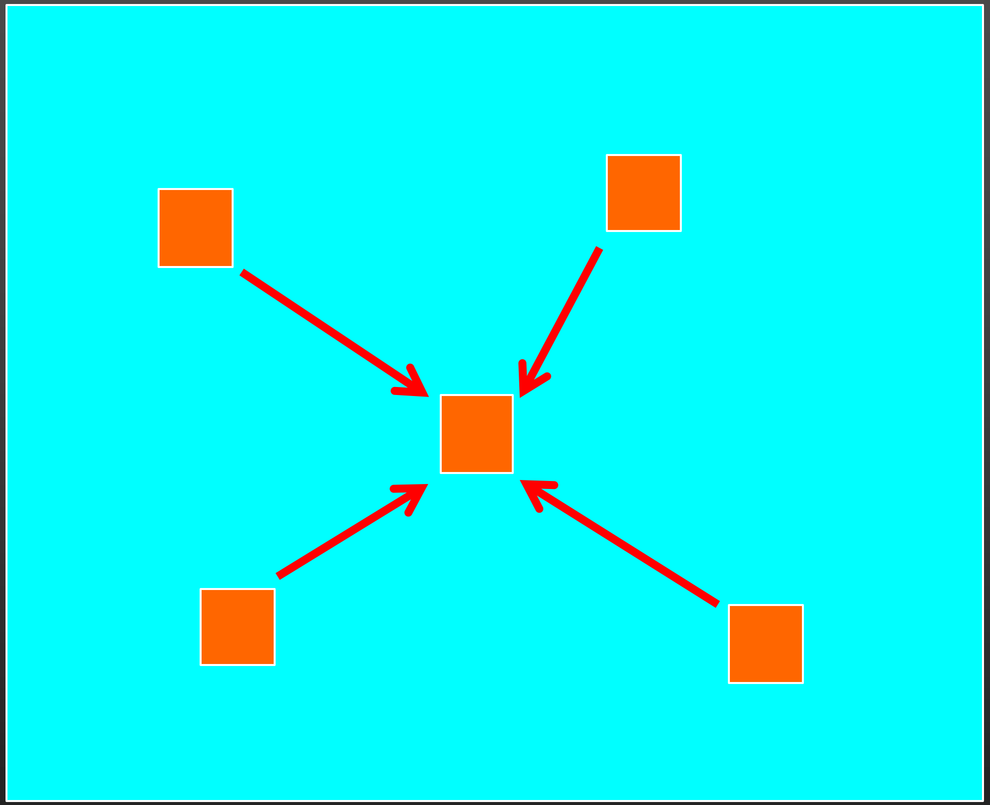
Non-local means filtering (Buades et al.'05)



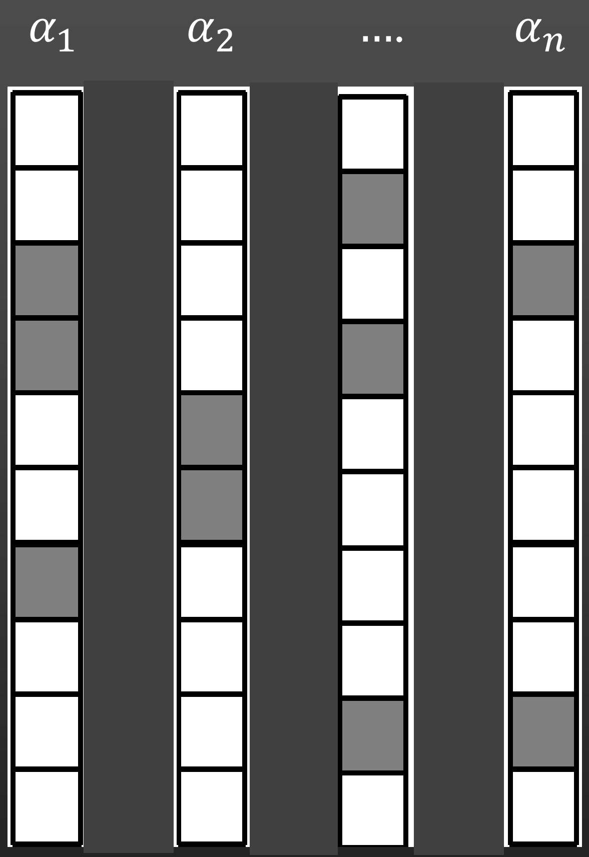
BM3D = Sparse representation on (3D) DCT dictionary + NLM (Dabov et al.'07)

Take $x \approx \sum_j \alpha^j d^j = D\alpha$ but limit the number of nonzero coefficients $||\alpha||_0 \leq k$

Let us start simple: How to denoise an image



Non-local means filtering (Buades et al.'05)



LSC: Dictionary learning with sparsity
(Elad & Aharon'06; Mairal et al.'08)

$$\min_{D,\alpha_1,\ldots,\alpha_n} \sum ||x_i - D\alpha_i||_F^2 + \lambda||\alpha_i||_1$$

Sparse coding and dictionary learning: A hierarchy of optimization problems

$\min_{\alpha} 1/2 \,|\, x - D\alpha \,|_2^2$

$\min_{\alpha} 1/2 \,|\, x - D\alpha \,|_2^2 + \lambda \,|\alpha|_0$

$\min_{\alpha} 1/2 \,|\, x - D\alpha \,|_2^2 + \lambda \,\psi(\alpha)$

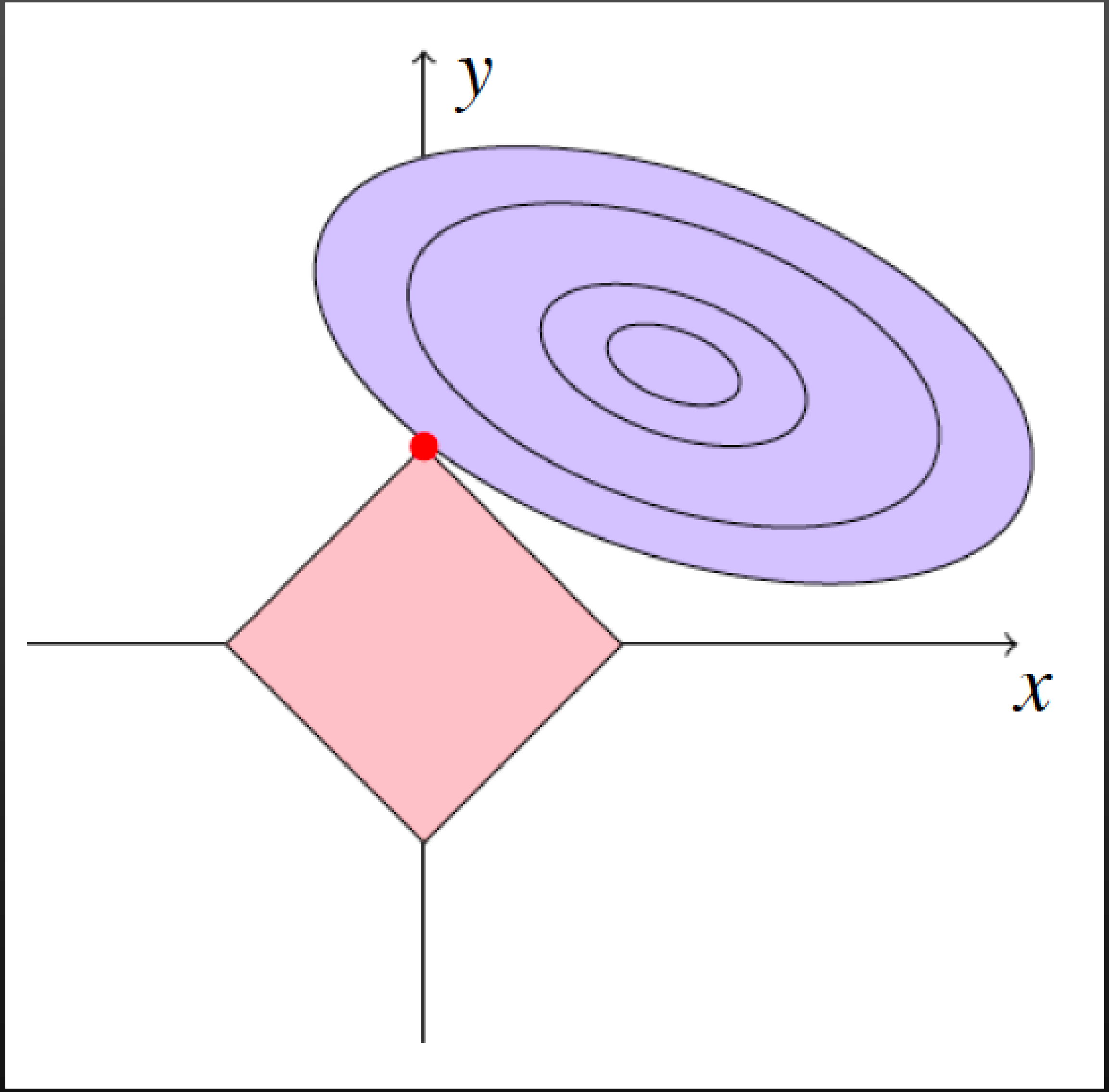
$\min_{D \in C, \alpha_1, \ldots, \alpha_n} \sum_{1 \le i \le n} [\, 1/2 \,|\, x_i - D\alpha_i \,|_2^2 + \lambda \,\psi(\alpha_i) \,]$

$\min_{D \in C, \alpha_1, \ldots, \alpha_n} \sum_{1 \le i \le n} [\, f\,(x_i, D, \alpha_i) + \lambda \,\psi(\alpha_i) \,]$

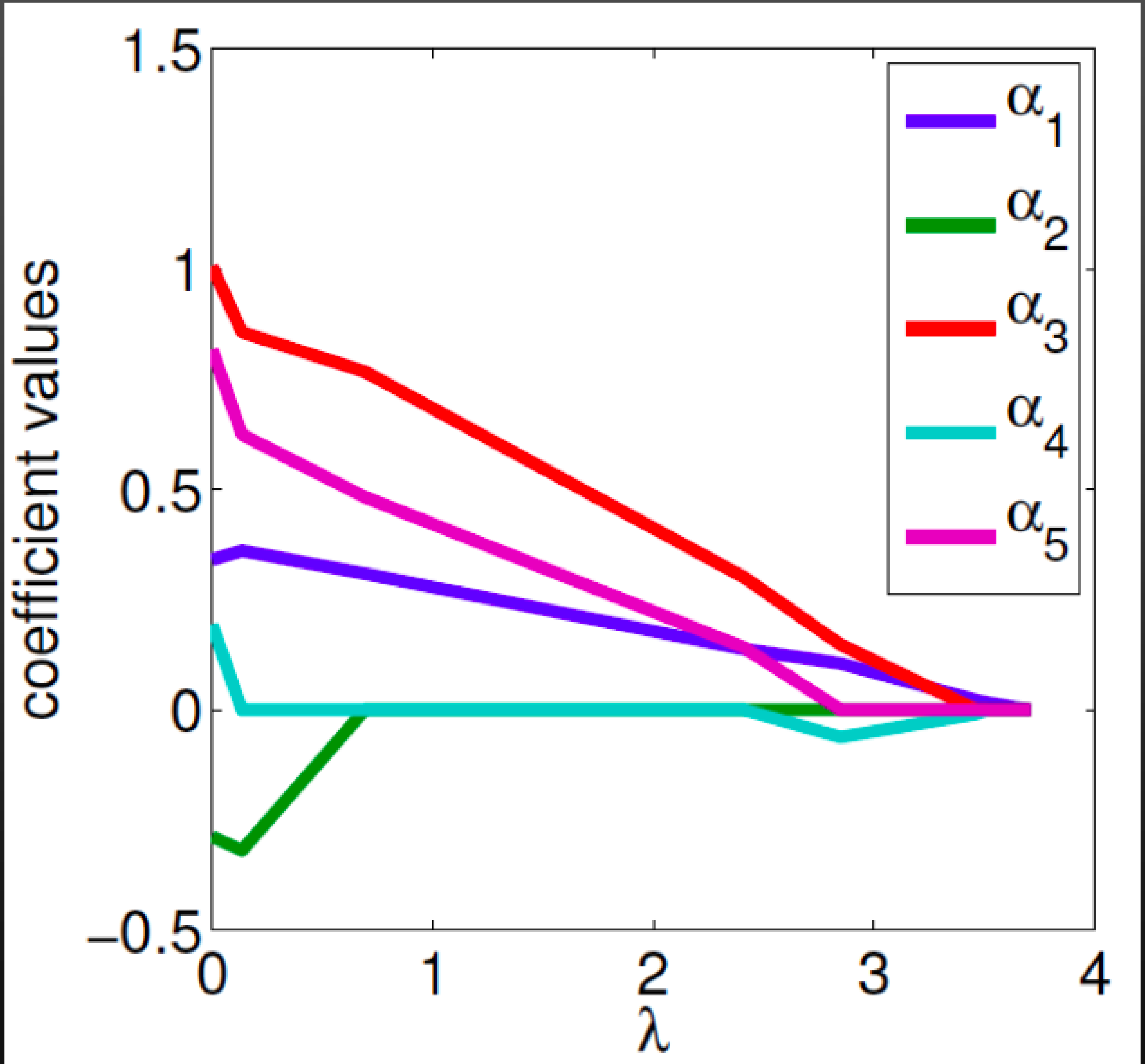
$\min_{D \in C, \alpha_1, \ldots, \alpha_n} \sum_{1 \le i \le n} [\, f\,(x_i, D, \alpha_i) + \lambda \sum_{1 \le k \le q} \psi(d_k) \,]$

Least squares
Sparse coding
Dictionary learning
Learning for a task
Learning structures

The l_1 norm and sparsity



LARS (Efron et al., 2004)



Dictionary learning

- Given some loss function, e.g.,

$$L(x, D) = 1/2 \, | x - D\alpha |_2^2 + \lambda \, |\alpha|_1$$

- One usually minimizes, given some data x_i, i = 1, ..., n, the empirical risk:

$$\min_D f_n(D) = \sum_{1 \leq i \leq n} L(x_i, D)$$

- But, one would really like to minimize the expected one, that is:

$$\min_D f(D) = \mathbb{E}_x [L(x, D)]$$

(Bottou& Bousquet'08 $\rightarrow$ Stochastic gradient descent)

Online sparse matrix factorization

(Mairal, Bach, Ponce, Sapiro, ICML'09, JMLR'10)

Problem:

$$\min_D f(D) = E_x [L(x, D)]$$

$$L(x, D) = 1/2 \, | x - D\alpha |_2^2 + \lambda \, |\alpha|_1$$

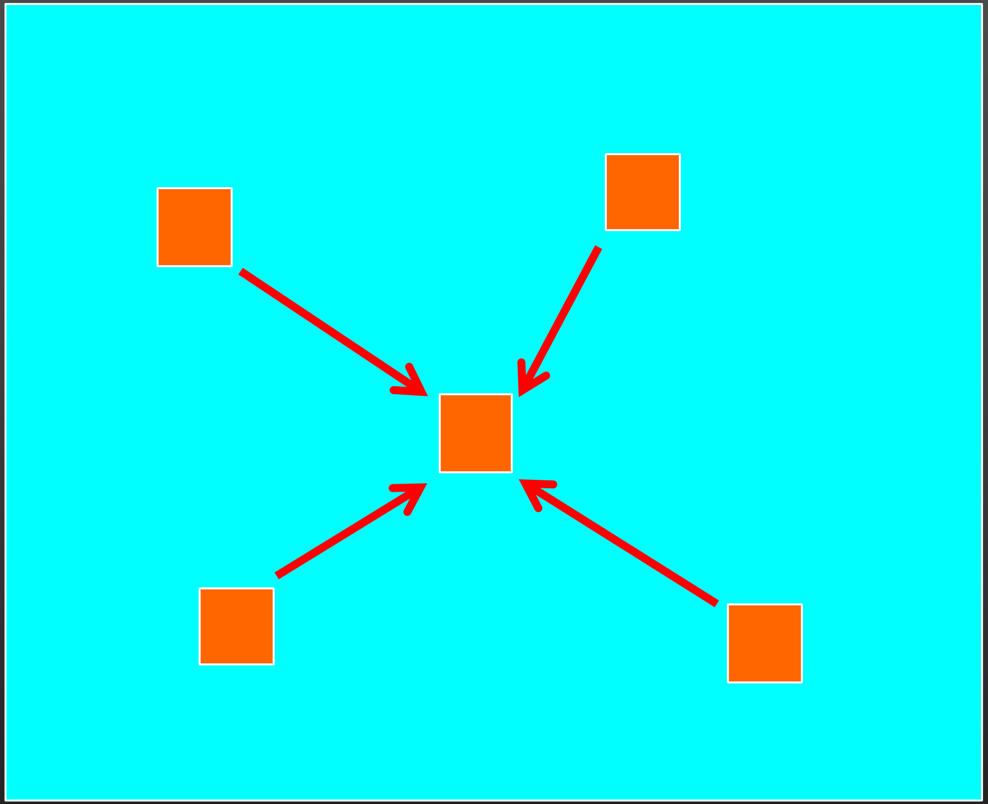
Algorithm:

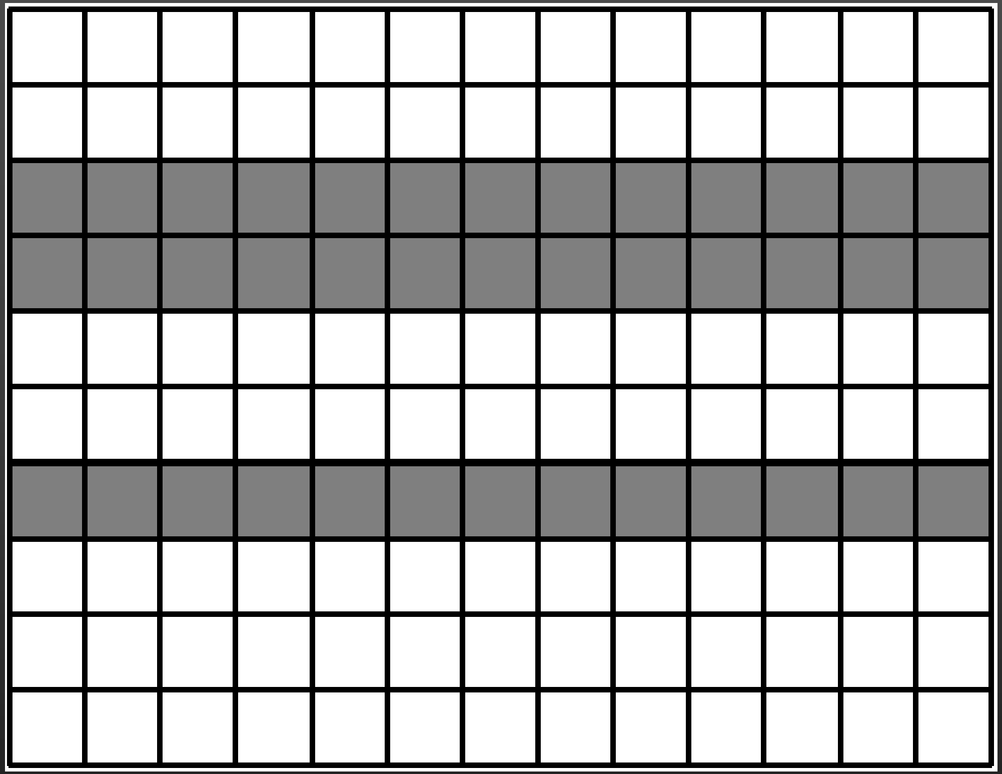
Iteratively draw one random training sample x_t and minimize the quadratic surrogate function:

$$g_t(D) = 1/t \sum_{1 \le i \le t} [\, 1/2 \, | x_i - D\alpha_i |_2^2 + \lambda \, |\alpha_i|_1 \,]$$

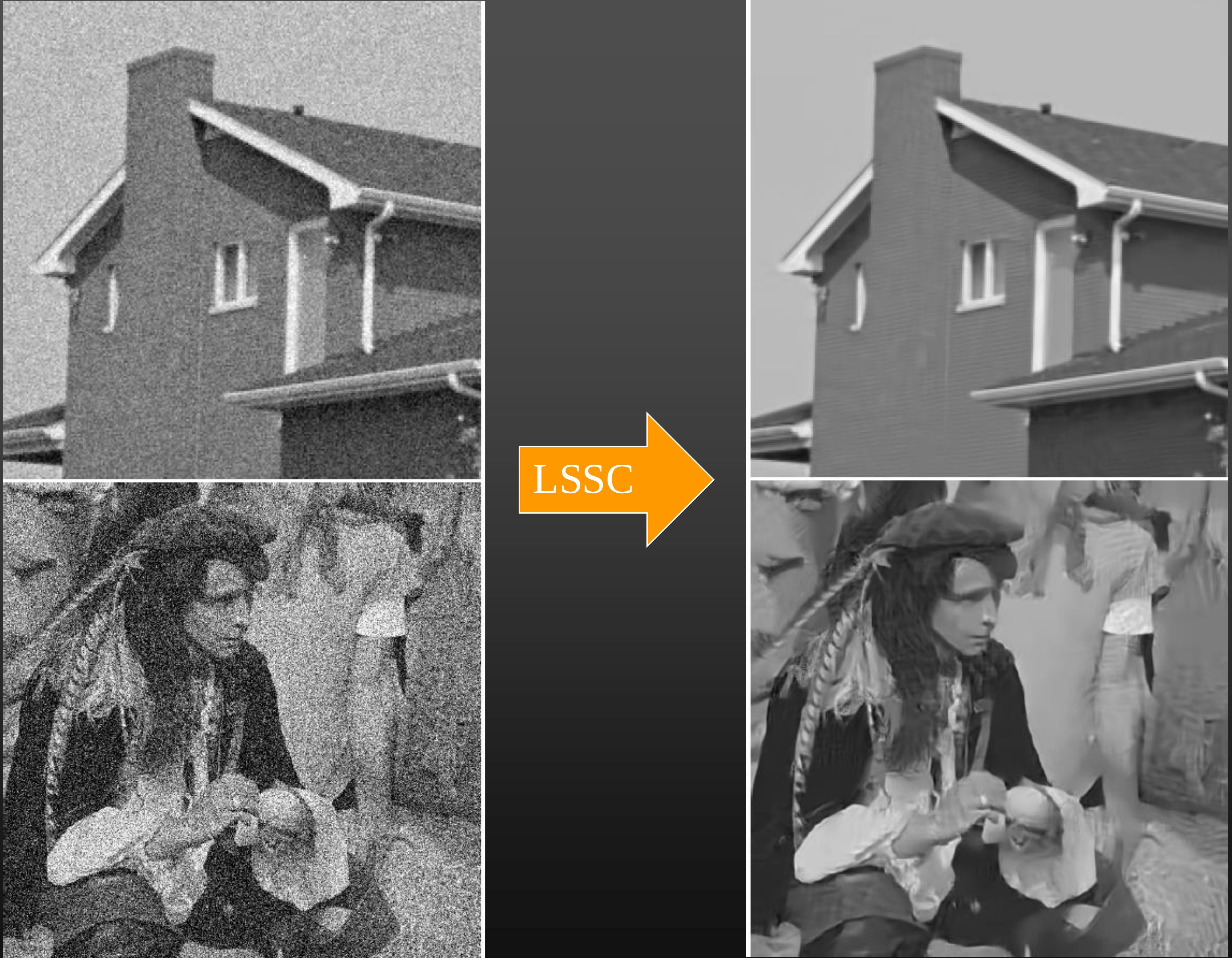
(Lars/Lasso for sparse coding, block-coordinate descent with warm restarts for dictionary updates, mini-batch extensions, etc.)

Let us start simple: How to denoise an image


Non-local means filtering (Buades et al.'05)


LSSC: Dictionary learning with structured sparsity (Mairal et al.'09)

$$\min_{D,A} \sum_i ||X_i - DA_i||_F^{\,2} + \lambda ||A_i||_{1,2} \quad \text{where} \quad ||A||_{1,2} = \sum_r ||\alpha^r||_2$$


LSSC

Real noise is complicated

- Noise = shot noise (physics) plus read noise (electronics)
- Random variable following a Gaussian distribution with zero mean and signal-dependent standard deviation function (Foi et al., 2008)

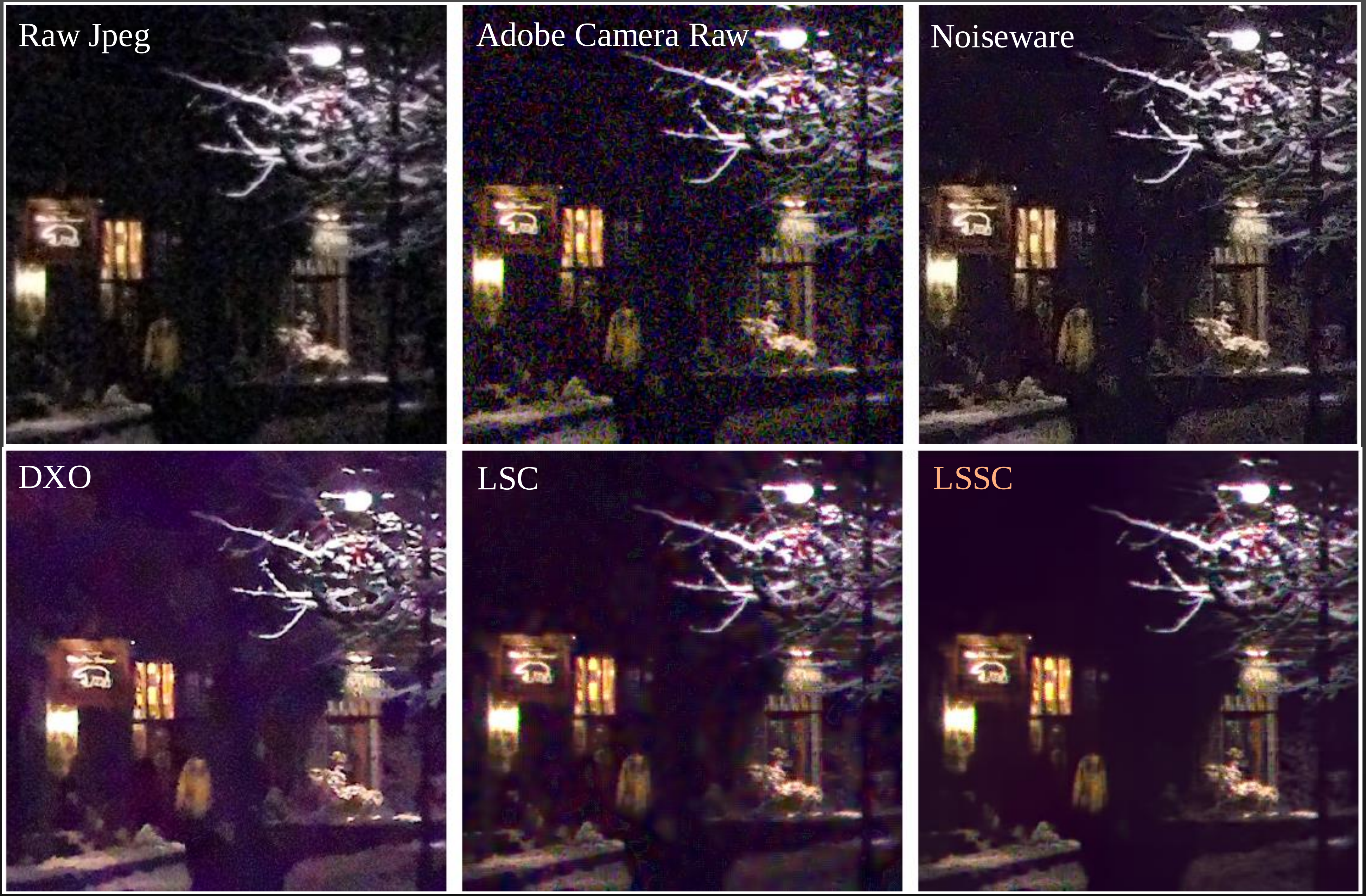
$$s(u) = \sqrt{\alpha y(u) + \beta}$$

whose parameters α and β can be determined for a given camera

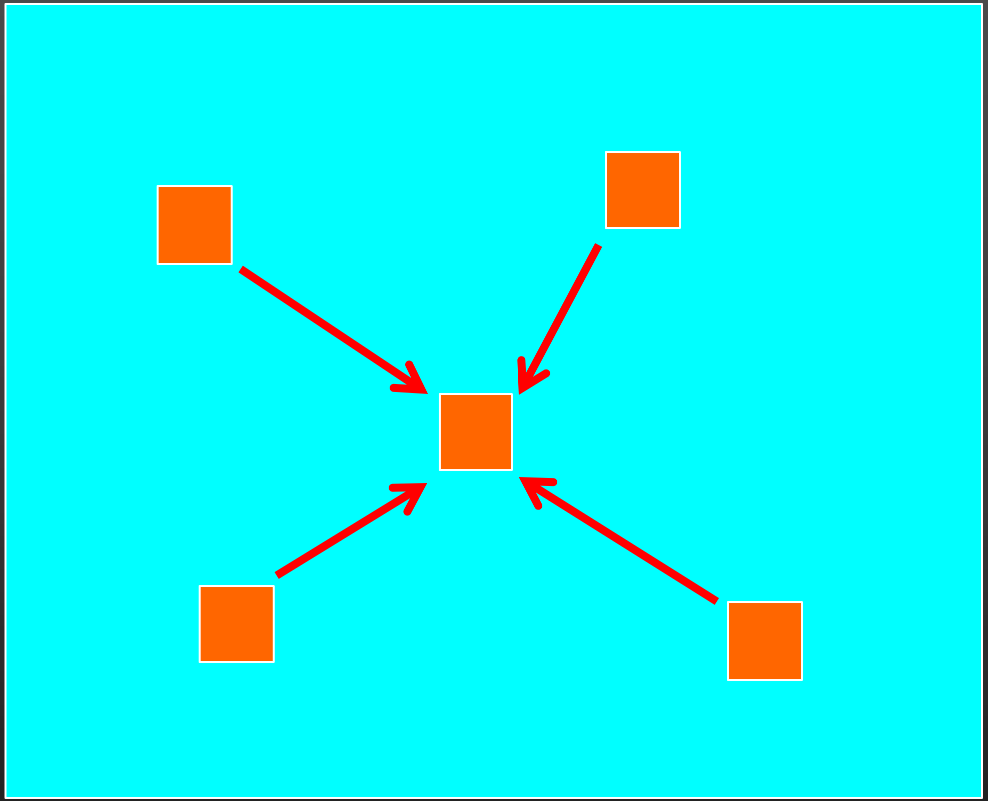
- This is only true for raw images. More on that later

A. Foi, M. Trimeche, V. Katkovnik, K. Egiazarian, “Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data”, IEEE TIP 17(10):1737–1754 (2008).

Real noise (Canon Powershot G9, 1600 ISO)



Let us start simple: How to denoise an image



Non-local means filtering (Buades et al.'05)

Self-attention (Vaswani et al., 2017)

$$X_i = S_i V_i, \text{ where } S_i = \text{softmax}(\frac{1}{\tau} K_i Q_i^T)$$

where

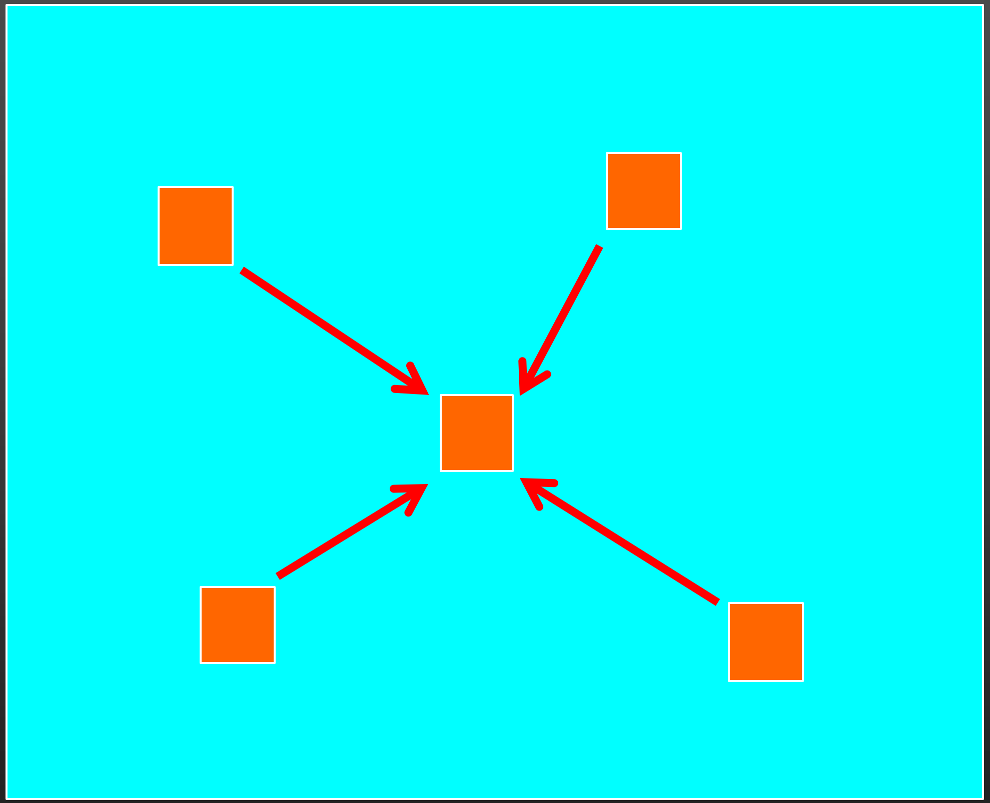
$$K_i = X_{i-1} A_i, \; Q_i = X_{i-1} B_i, \text{ and } V_i = X_{i-1} C_i$$

$$\Downarrow$$

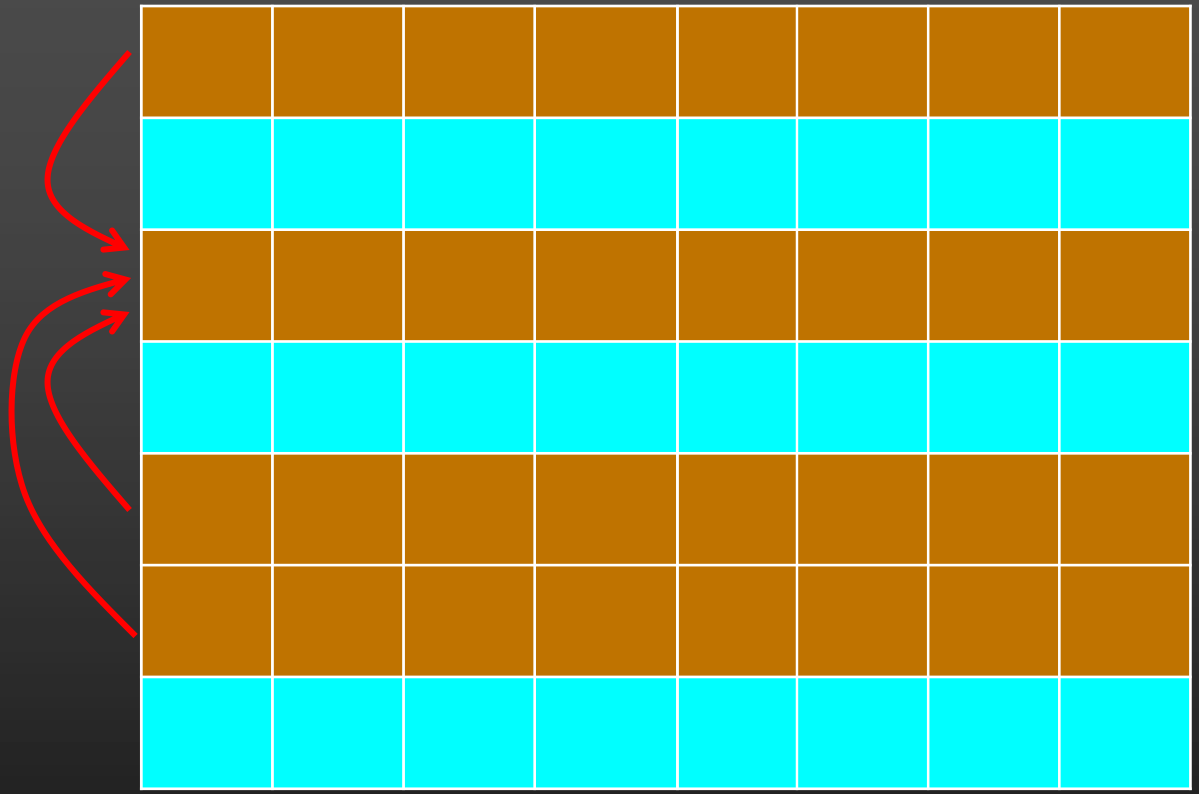
$$\begin{aligned} X' &= X_k = S_k X_{k-1} C_k = S_k (S_{k-1} X_{k-2} C_{k-1}) C_k \\ &= (S_k \ldots S_1) X (C_1 \ldots C_k) = T_k X D_k, \end{aligned}$$

Note: T_k is a stochastic matrix, thus the rows of $T_k X$ are barycentric combinations of all the rows of X, weighted in a complex way by their affinities $X_{i-1} A_i B_i^T X_{i-1}^T$. (See also Andrej Karpathy's talk.)

Let us start simple: How to denoise an image



Non-local means filtering (Buades et al.'05)



Note: T_k is a stochastic matrix, thus the rows of $T_k X$ are barycentric combinations of all the rows of X, weighted in a complex way by their affinities $X_{i-1} A_i B_i^T X_{i-1}^T$. (See also Andrej Karpathy's talk.)

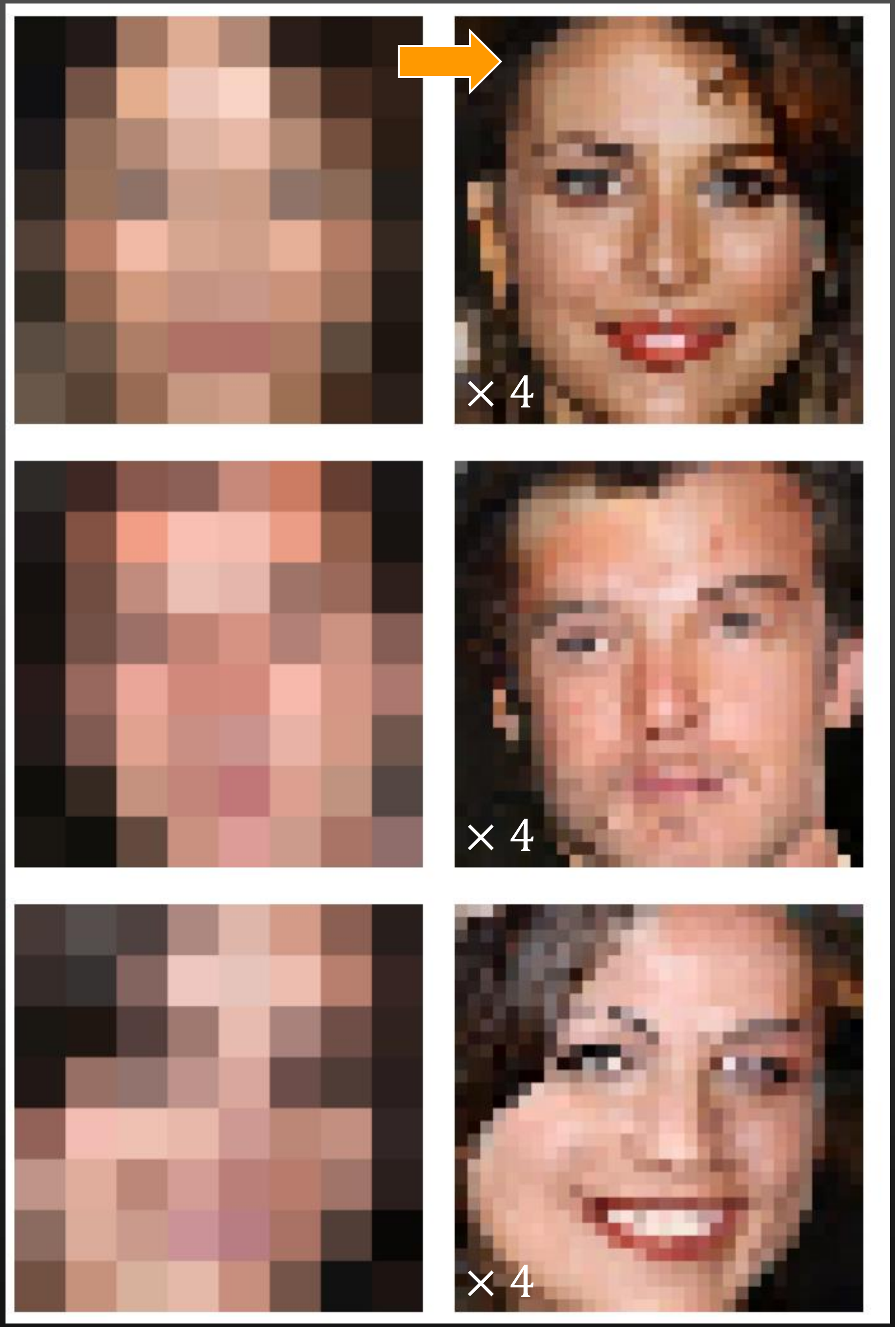


Image interpolation
aka
Depixellisation
aka
Example-based super-resolution
aka
Single-image super-resolution

(Dahl et al., 2017)

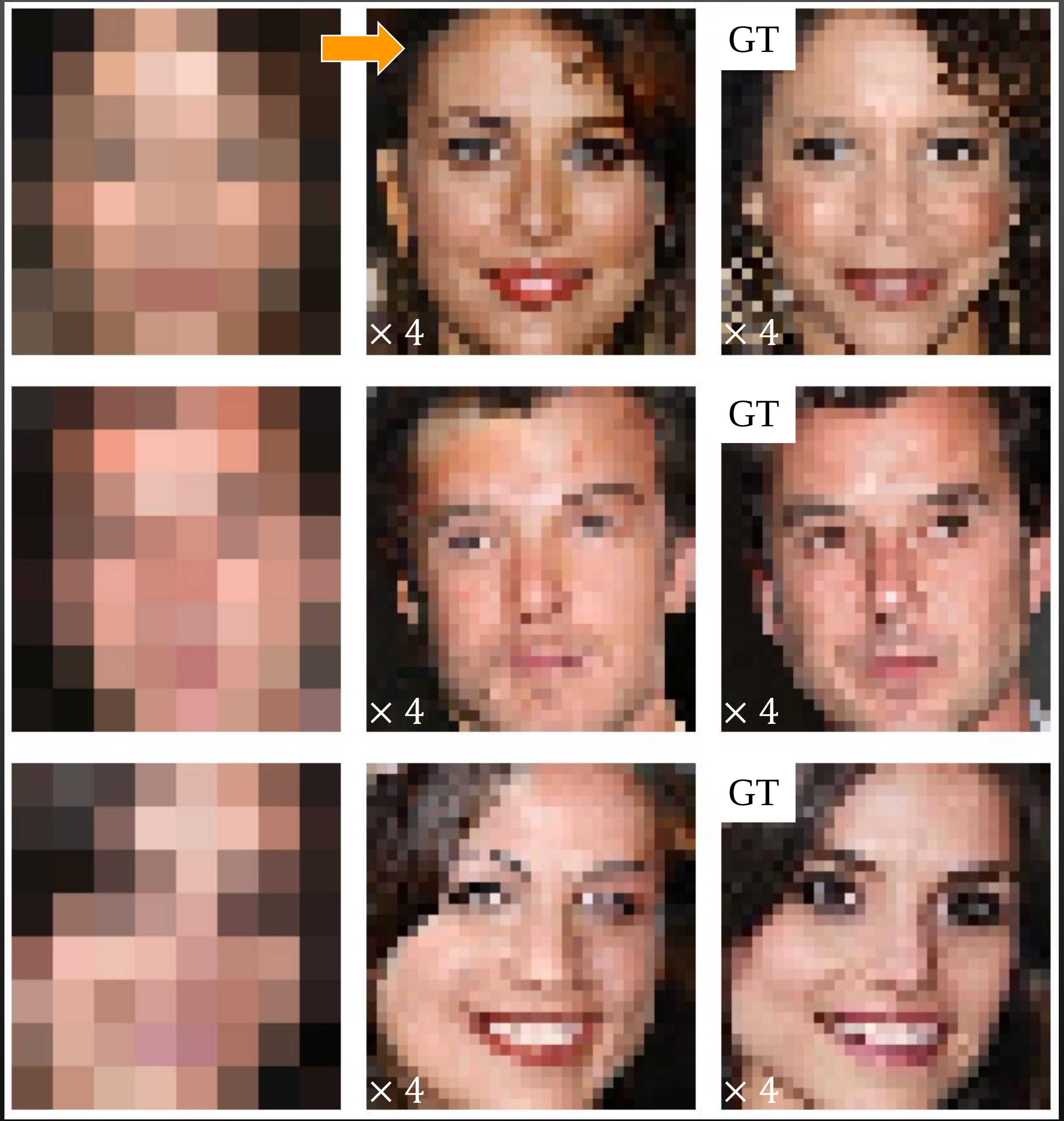


Image interpolation

aka

Depixellisation

aka

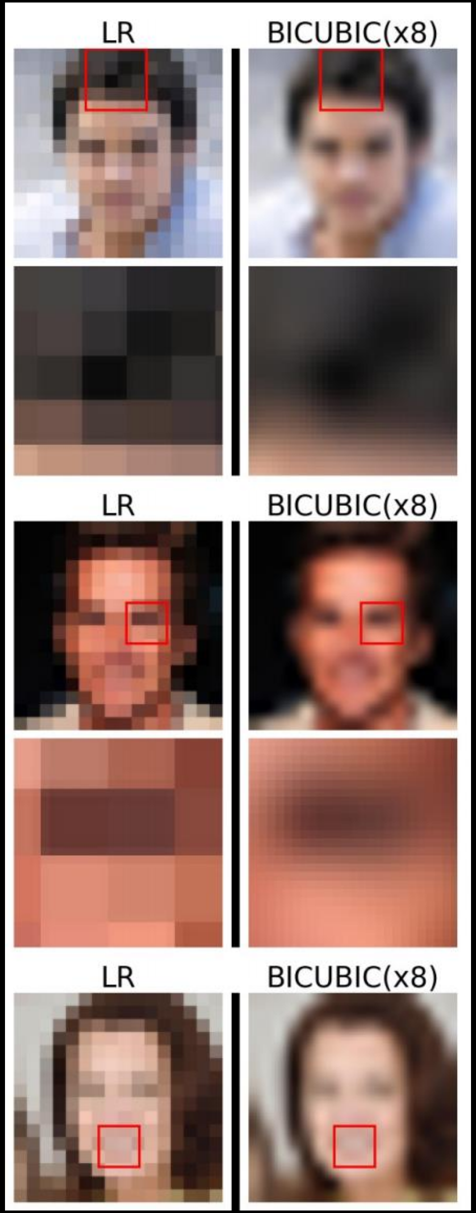
Example-based super-resolution

aka

Single-image super-resolution

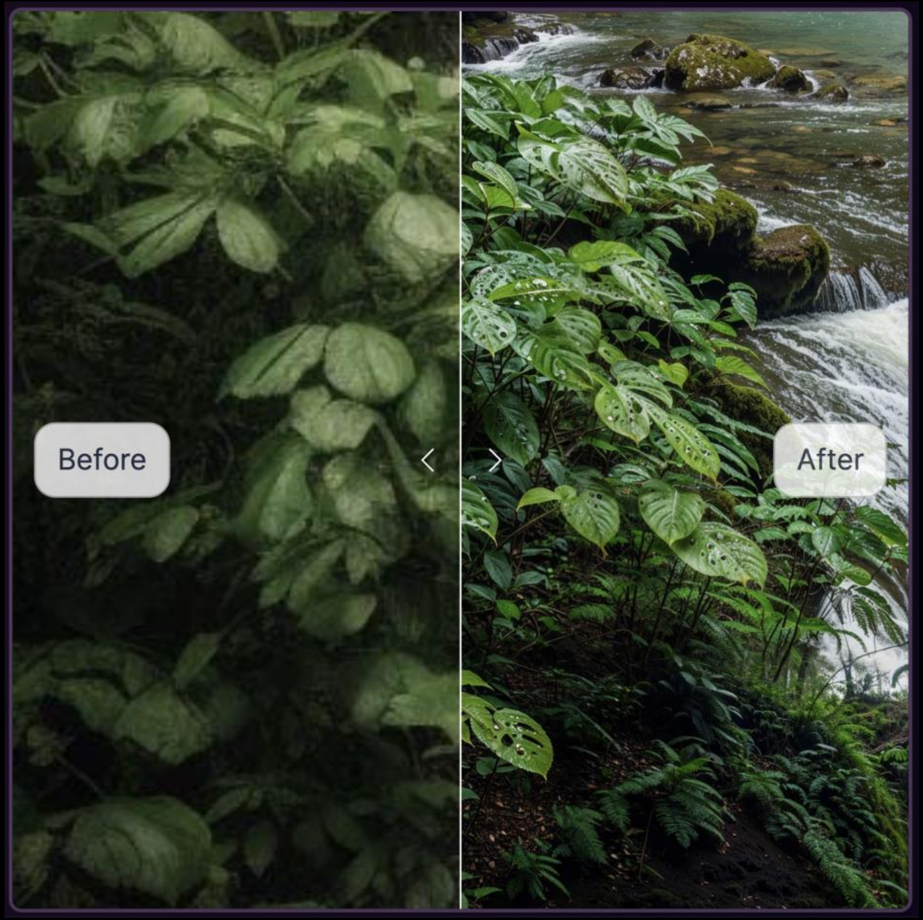
(Dahl et al., 2017)

Problem: Not enough information in a single image: details must be hallucinated



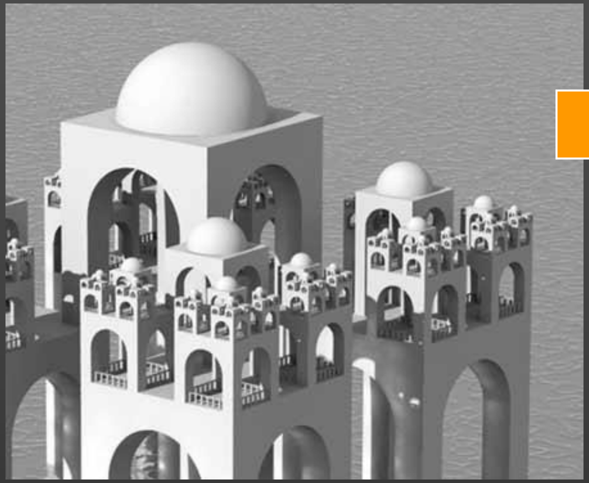
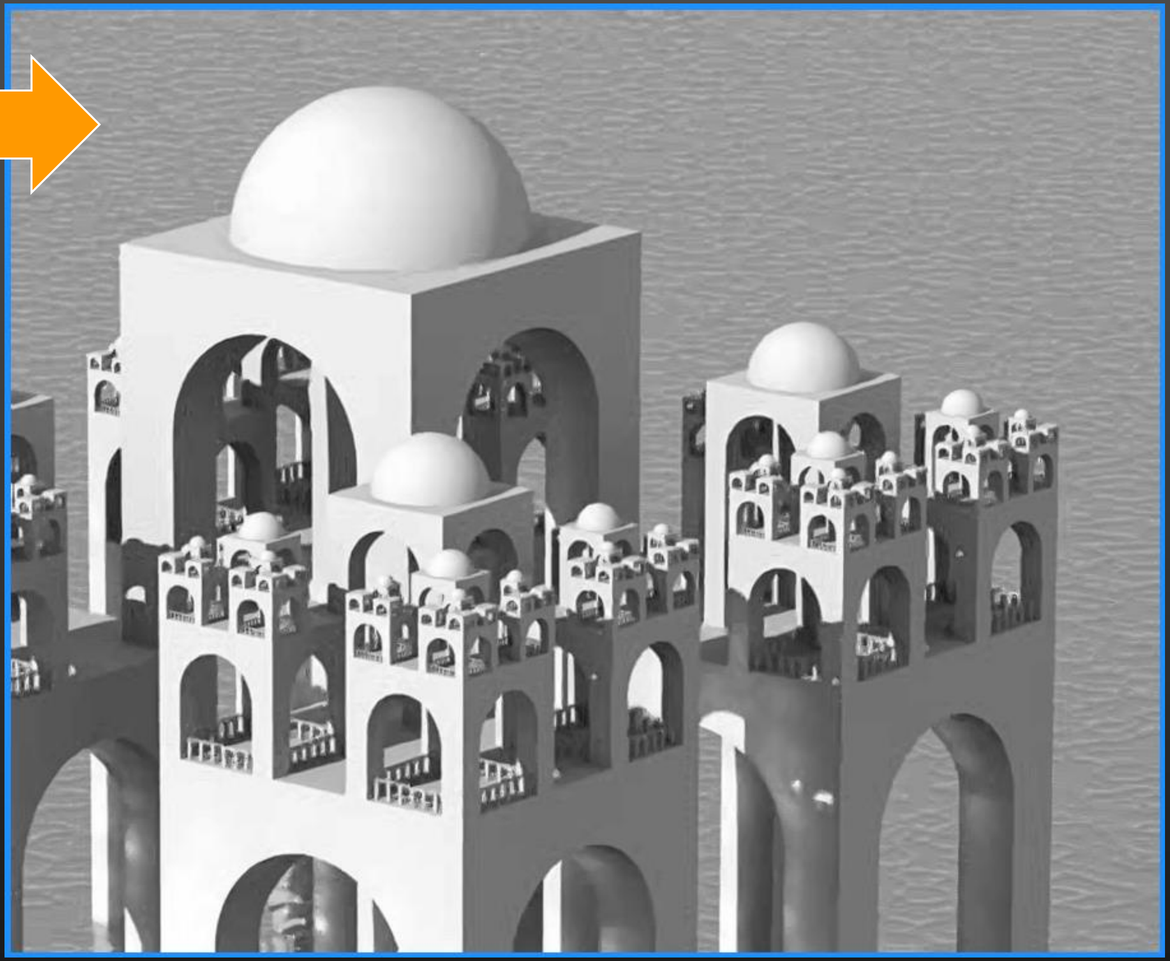
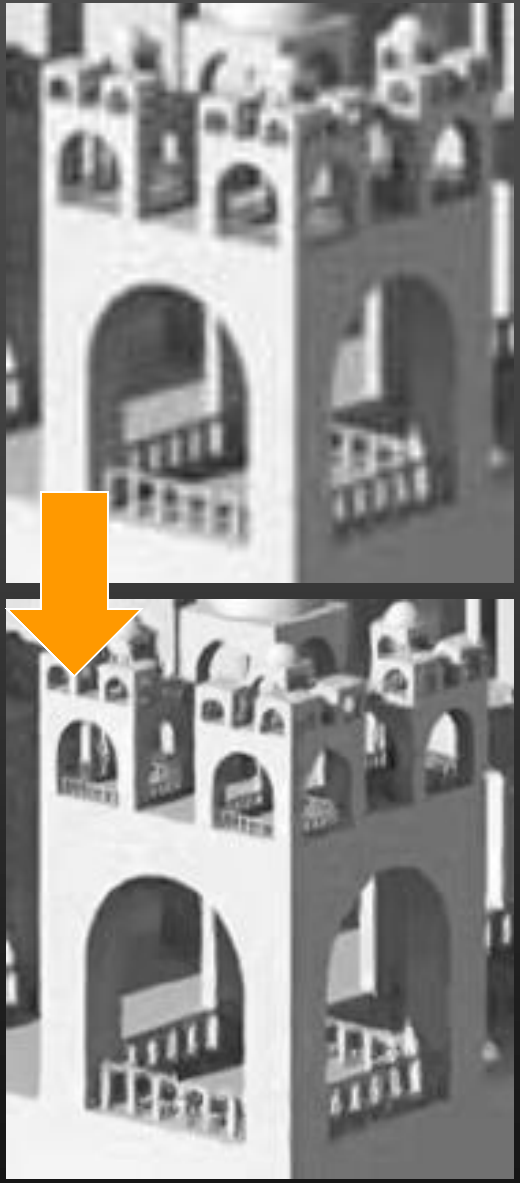
(FSRNET Chen et al., 2017)
(FSRGAN Zhu et al., 2020)
(PULSE Menon et al., 2020)

Generative vision

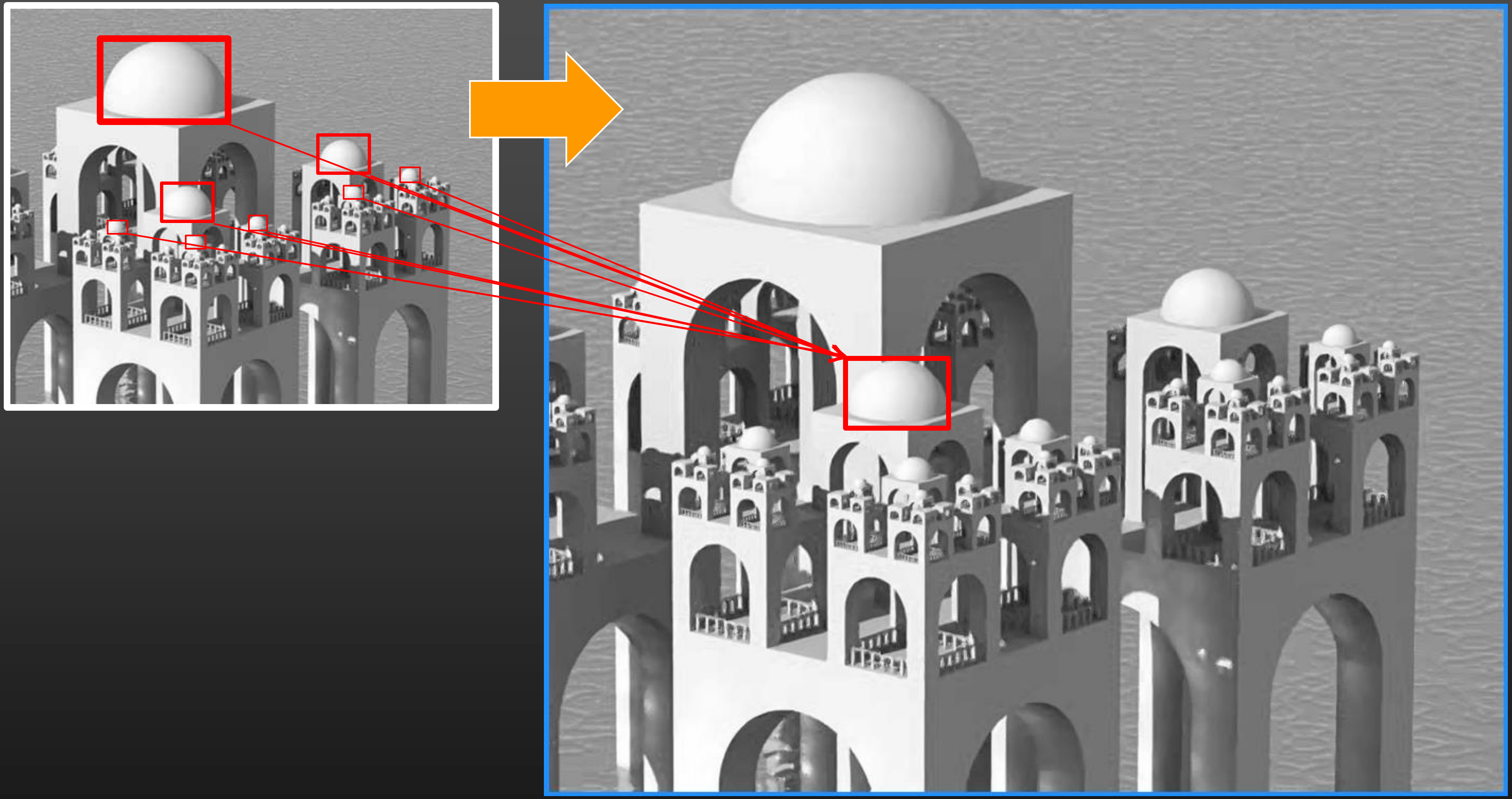


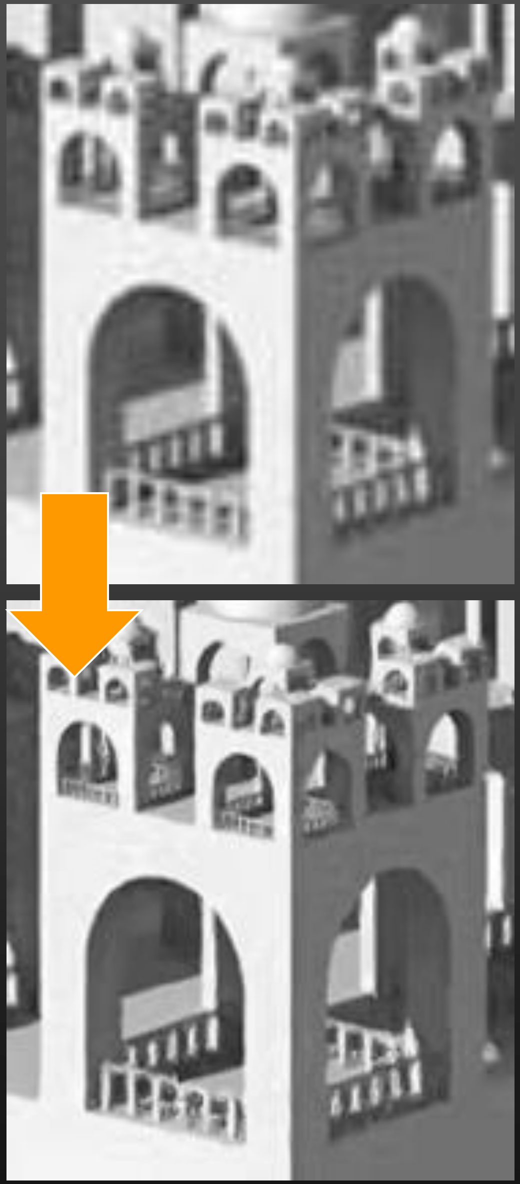
https://magnific.ai/

Super-Resolution from a Single Image (Glasner, Bagon, Irani, ICCV'09)

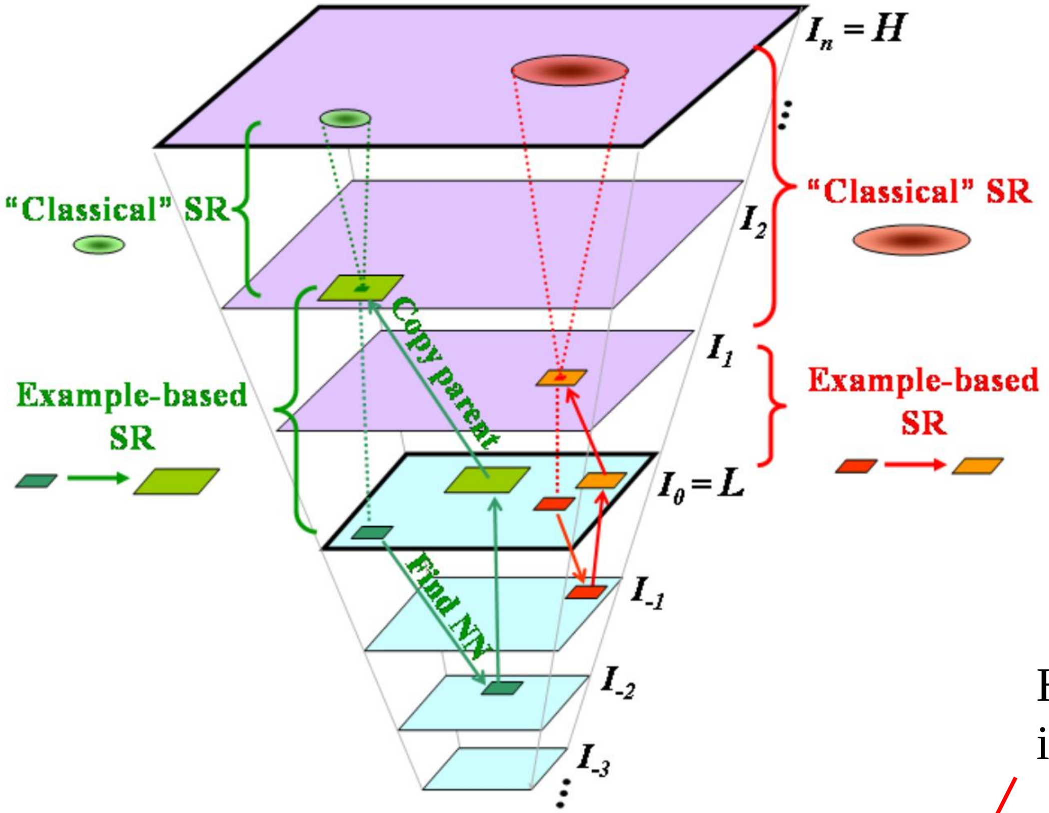
Super-Resolution from a Single Image (Glasner, Bagon, Irani, ICCV'09)





Key idea: exploit internal self-similarities

Super-Resolution from a Single Image (Glasner, Bagon, Irani, ICCV'09)



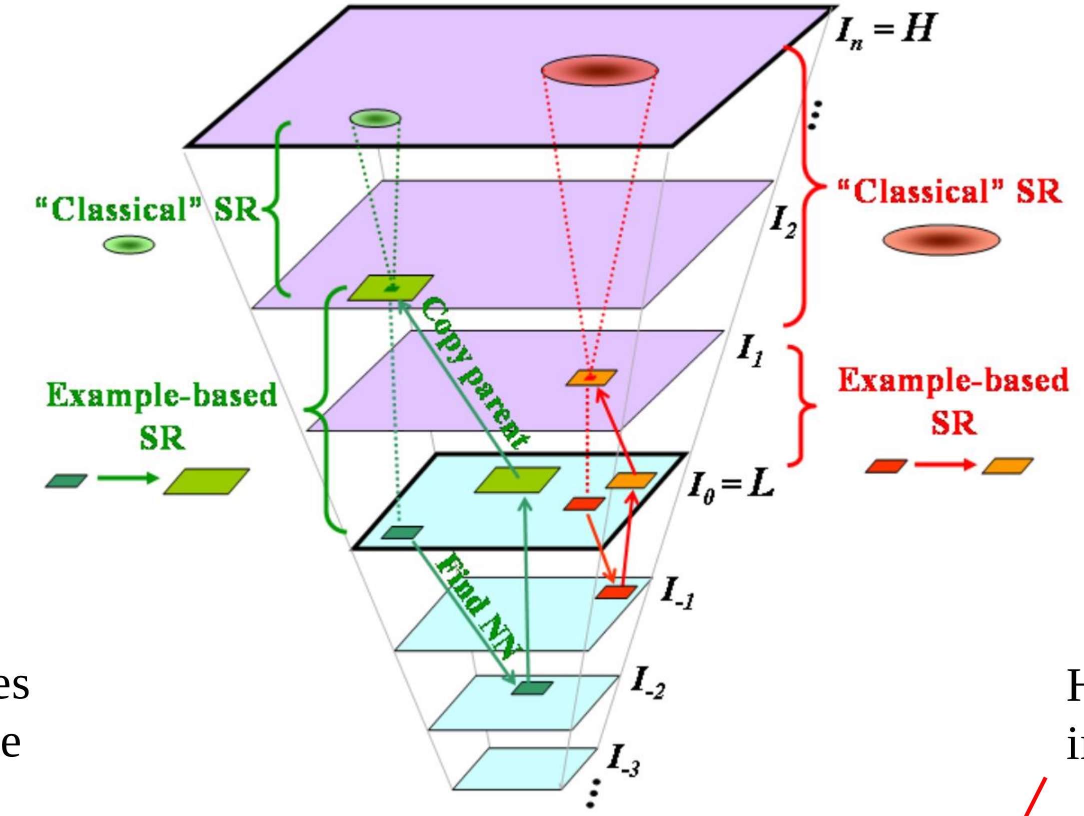
Lo-res image

Hi-res image

Sampling blur

$$L_j(p) = (H * B_j)(q) = \Sigma_{q_i \in Support(B_j)} H(q_i)\, B_j(q_i - q)$$

Super-Resolution from a Single Image (Glasner, Bagon, Irani, ICCV'09)

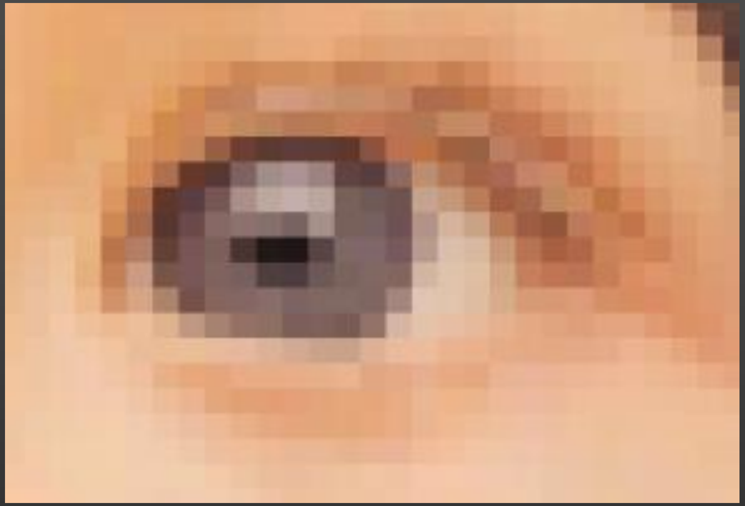


Lo-res image

Hi-res image

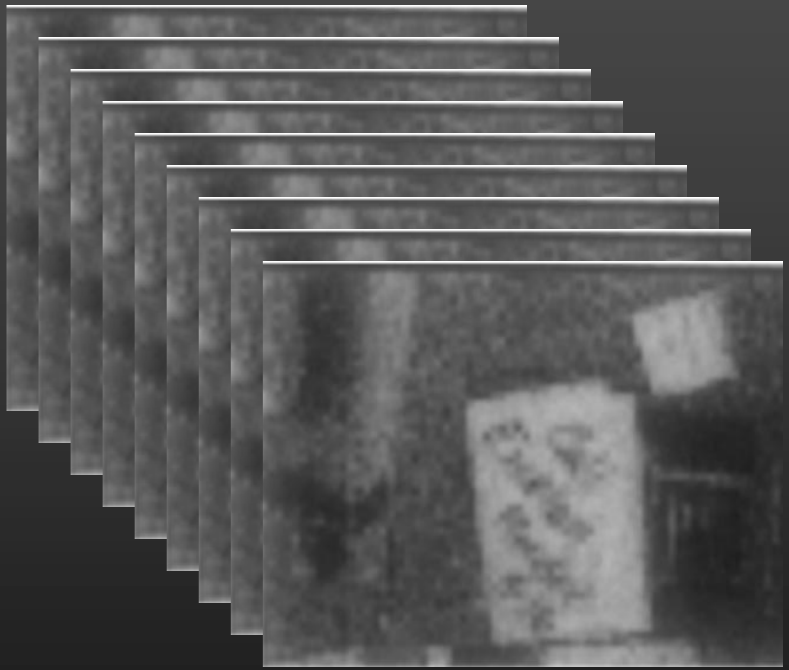
Sampling blur

$$L_j(p) = (H * B_j)(q) = \Sigma_{q_i \in Support(B_j)} H(q_i)\, B_j(q_i - q)$$

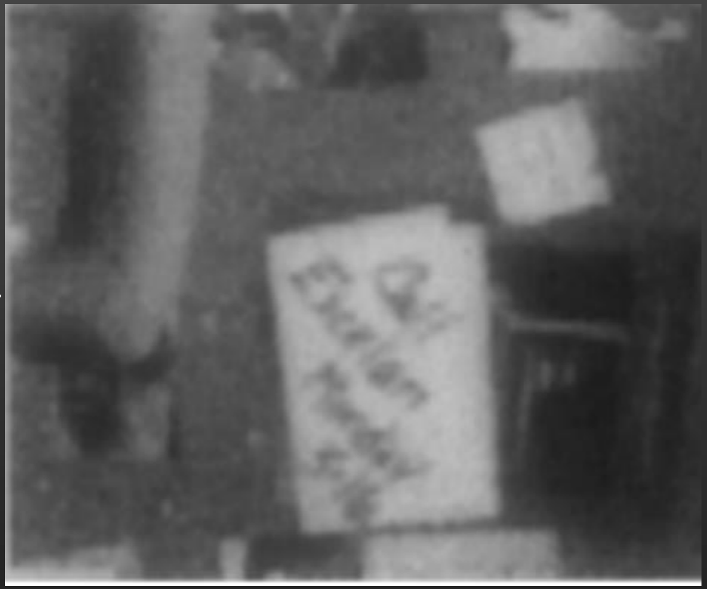


True (= multi-frame) super-resolution: no need to hallucinate details

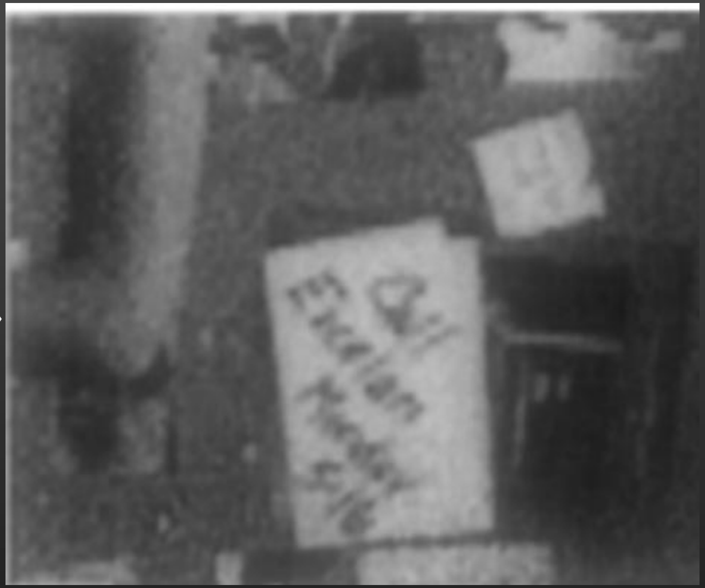
Input burst



Initial estimate

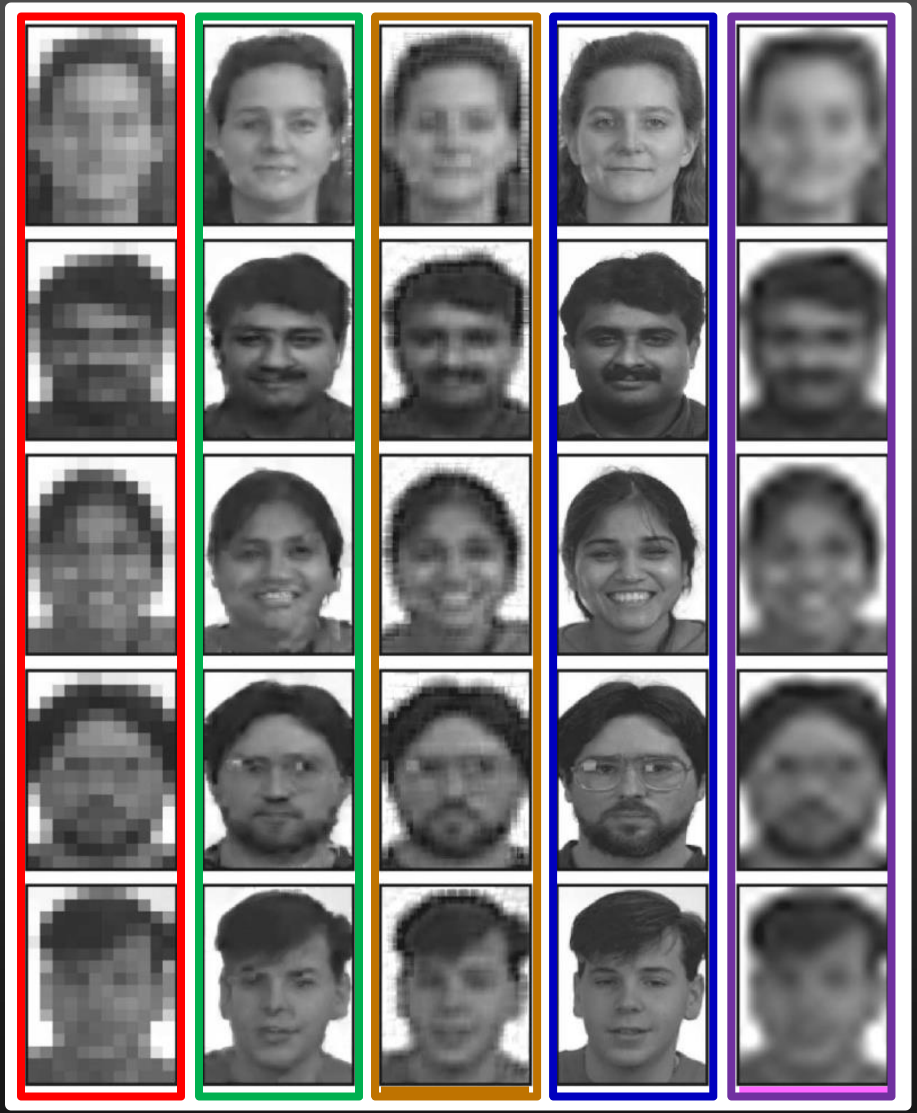


Refined estimate



(Irani & Peleg, 1991; see also Tsai & Huang, 1984)

Super-resolution with “hallucination/recogstruction”


(Baker and Kanade, 2002)

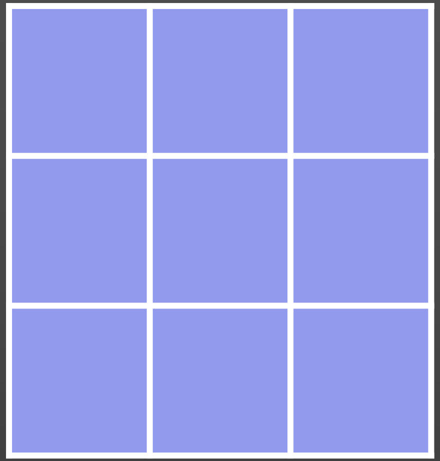
- LR input image (1 of 4)
- Recogstruction
- Ground-truth HR image

- (Hardie et al., 1997)
- Bicubic interpolation

$\times 4$, alignment known exactly

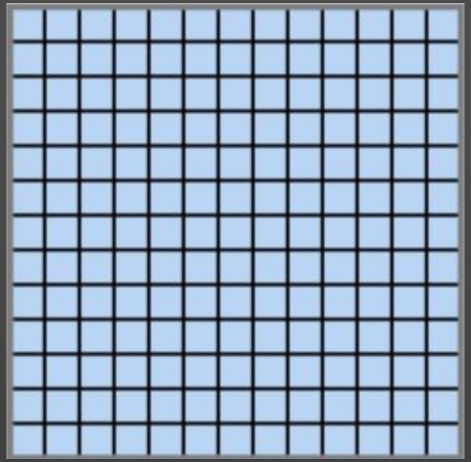
Key idea: learn a prior on the spatial distribution of the image gradient for frontal images of faces

1 LR RGB image

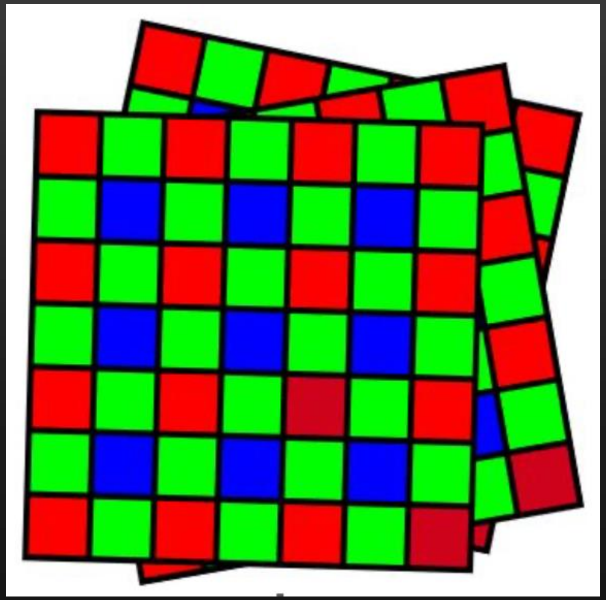


Single-image interpolation

1 HR RGB image

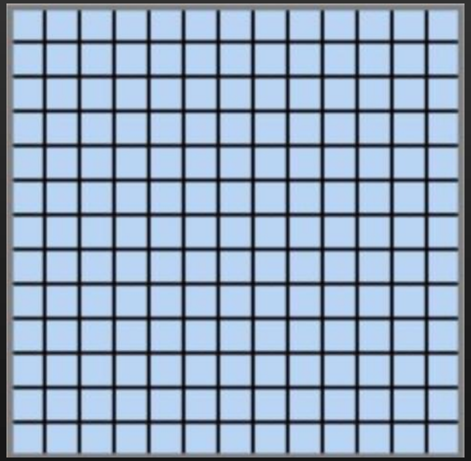


20 LR raw images = burst



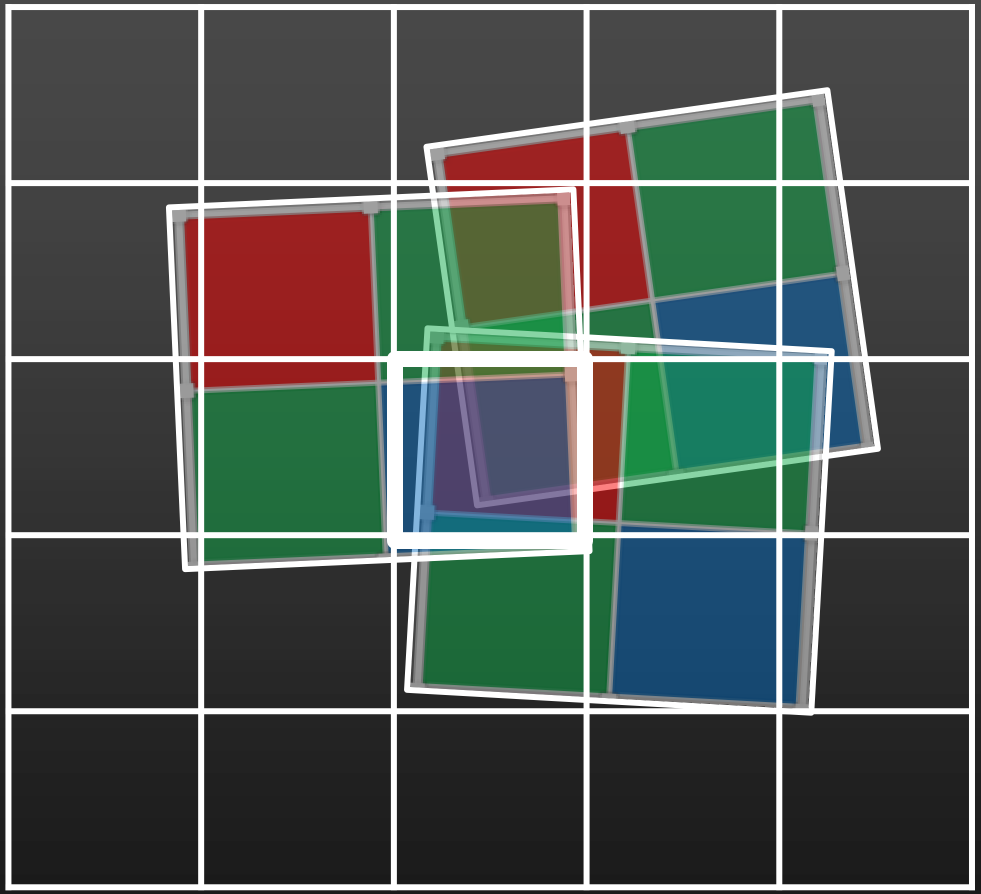
Super-resolution

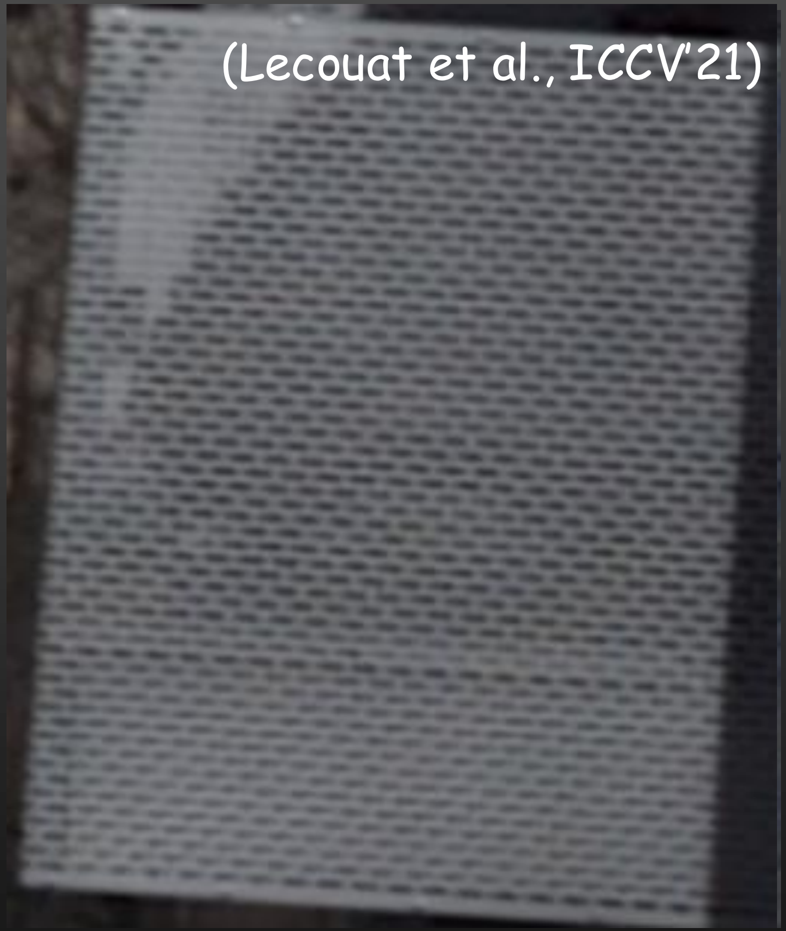
1 HR RGB image



Handheld Multi-Frame Super-Resolution

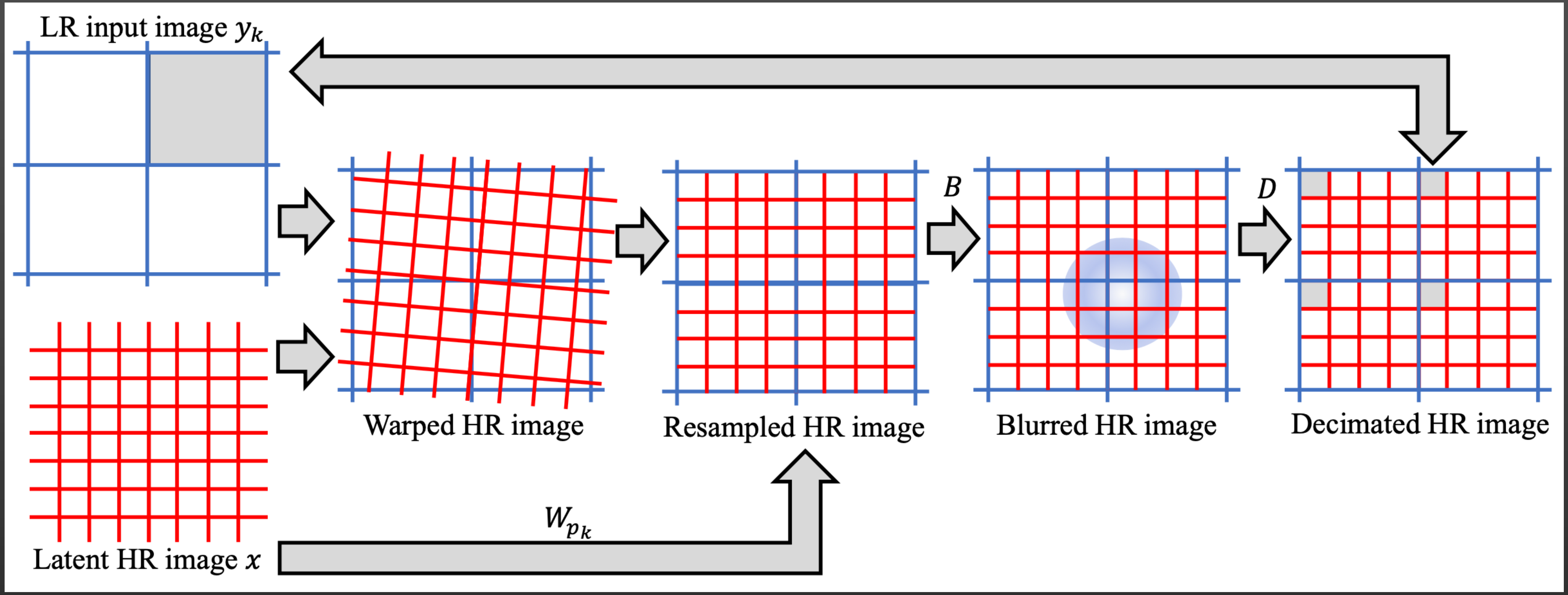
(Wronski, Garcia-Dorado, Ernst, Kelly, Kainin, Liang, Levoy, Milanfar, SIGGRAPH'19)





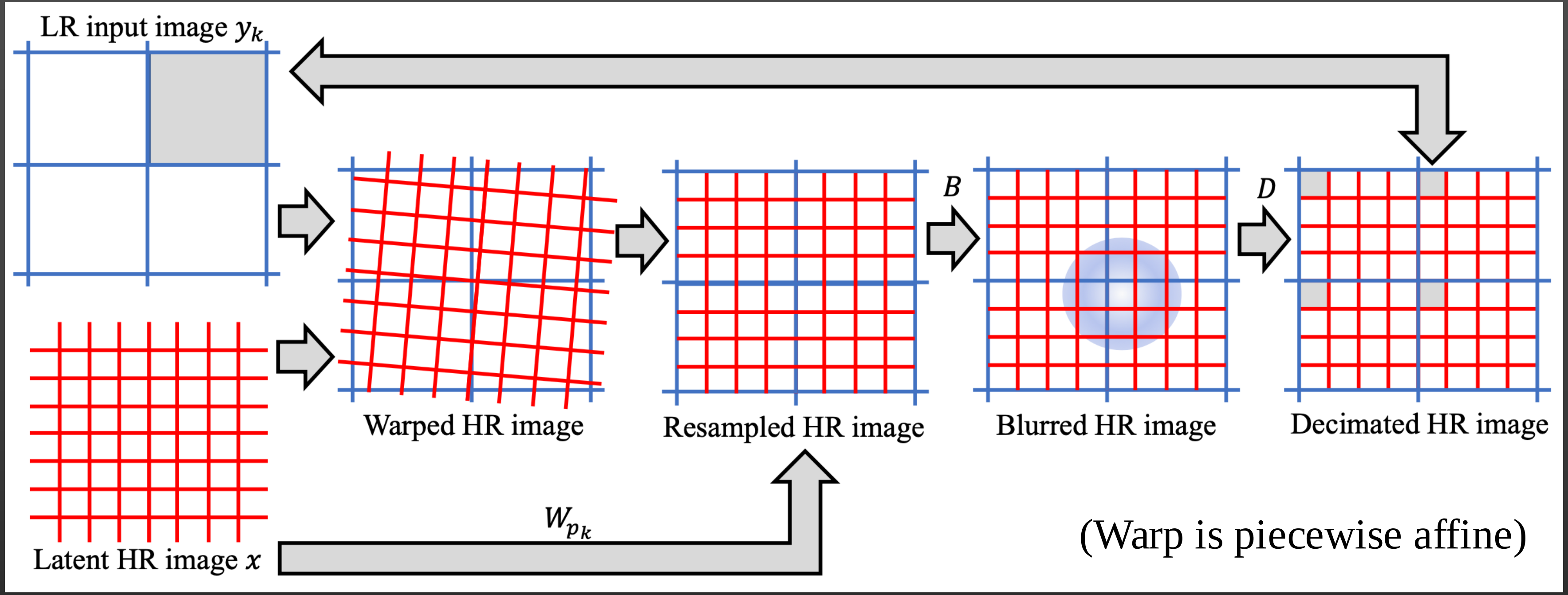
Key idea: exploit natural hand tremor and avoid single-image demosaicing altogether

Super-resolution as an inverse problem



- Forward model: $y_k = U_{p_k}\, x + \varepsilon_k$ for $k = 1, \dots, K$ with $U_{p_k} = DBW_{p_k}$
- Solve $\min_{x,p} \frac{1}{2}\, || y - U_p\, x||^2 \; + \lambda\varphi\,(x)$ where $y = \begin{bmatrix} y_1 \\ \vdots \\ y_K \end{bmatrix}$ and $U_\text{p} = \begin{bmatrix} U_{p_1} \\ \vdots \\ U_{p_K} \end{bmatrix}$

Lucas-Kanade Reloaded: End-to-End Super-Resolution from Raw Image Bursts (Lecouat, Ponce, Mairal, ICCV'21)



- $y_k = U_{p_k}\, x + \varepsilon_k$ for $k = 1, \dots, K$ with $U_{p_k} = DBW_{p_k}$
- Define $x_\theta(y) = \operatorname{argmin}_{x,p} \frac{1}{2} \| y - U_p\, x \|^2 + \lambda \varphi_\theta(x)$
- Minimize wrt θ the objective $\frac{1}{n} \sum_{1 \le i \le n} \| x_i - x_\theta(y_i) \|_1$

Note:
- Almost impossible to get real training data
- "Semi-synthetic" training data constructed using "ISP inversion" (Brooks at al., 2019) with a realistic noise model

Optimization: unrolled iterative algorithm

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\|\mathbf{y} - U_{\mathbf{p}}\,\mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

$$\min_{\mathbf{x},\mathbf{p},\mathbf{z}} E_\mu(\mathbf{x},\mathbf{z},\mathbf{p}) = \frac{1}{2}\|\mathbf{y}-U_{\mathbf{p}}\,\mathbf{z}\|^2 + \frac{\mu}{2}\|\mathbf{z}-\mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

Quadratic penalty (aka HQS) method (three iterations)

- ➢ $\mathbf{z}^t \leftarrow \mathbf{z}^{t-1} - \eta_t \left[U_{\mathbf{p}^{t-1}}^\top (U_{\mathbf{p}^{t-1}}\mathbf{z}^{t-1} - \mathbf{y}) + \mu(\mathbf{z}^{t-1} - \mathbf{x}^{t-1})\right]$ — One step of gradient descent (or a few)
- ➢ $\min_{\mathbf{p}_k} \frac{1}{2}\|\mathbf{y}_k - DBW_{\mathbf{p}_k}\mathbf{z}^t\|^2$ — Gauss-Newton (aka Lucas-Kanade)
- ➢ $\mathbf{x}^t \leftarrow \arg\min_{\mathbf{x}} \frac{\mu_{t-1}}{2}\|\mathbf{z}^t - \mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$ — Proximal update
- ➢ Increment μ

K. Gregor, Y. LeCun, “Learning fast approximations of sparse coding”, ICML’10

Optimization: unrolled iterative algorithm

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\|\mathbf{y} - U_{\mathbf{p}}\,\mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

$$\min_{\mathbf{x},\mathbf{p},\mathbf{z}} E_\mu(\mathbf{x},\mathbf{z},\mathbf{p}) = \frac{1}{2}\|\mathbf{y} - U_{\mathbf{p}}\,\mathbf{z}\|^2 + \frac{\mu}{2}\|\mathbf{z}-\mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

Quadratic penalty (aka HQS) method (three iterations)

- $\mathbf{z}^t \leftarrow \mathbf{z}^{t-1} - \eta_t\left[U_{\mathbf{p}^{t-1}}^\top (U_{\mathbf{p}^{t-1}}\mathbf{z}^{t-1} - \mathbf{y}) + \mu(\mathbf{z}^{t-1} - \mathbf{x}^{t-1})\right]$ — One step of gradient descent (or a few)
- $\mathbf{p}_k^t \leftarrow \mathbf{p}_k^{t-1} - \left(\mathbf{J}_k^{t\top}\mathbf{J}_k^t\right)^{-1}\mathbf{J}_k^{t\top}\mathbf{r}_k^t$ (3 times) — Gauss-Newton (aka Lucas-Kanade)
- $\mathbf{x}^t \leftarrow \arg\min_{\mathbf{x}} \frac{\mu_{t-1}}{2}\|\mathbf{z}^t - \mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$ — Proximal update
- Increment μ

K. Gregor, Y. LeCun, "Learning fast approximations of sparse coding", ICML'10

Optimization: unrolled iterative algorithm

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\|\mathbf{y} - U_{\mathbf{p}}\,\mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

$$\min_{\mathbf{x},\mathbf{p},\mathbf{z}} E_\mu(\mathbf{x},\mathbf{z},\mathbf{p}) = \frac{1}{2}\|\mathbf{y} - U_{\mathbf{p}}\,\mathbf{z}\|^2 + \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}\|^2 + \lambda\phi_\theta(\mathbf{x})$$

Quadratic penalty (aka HQS) method (three iterations)

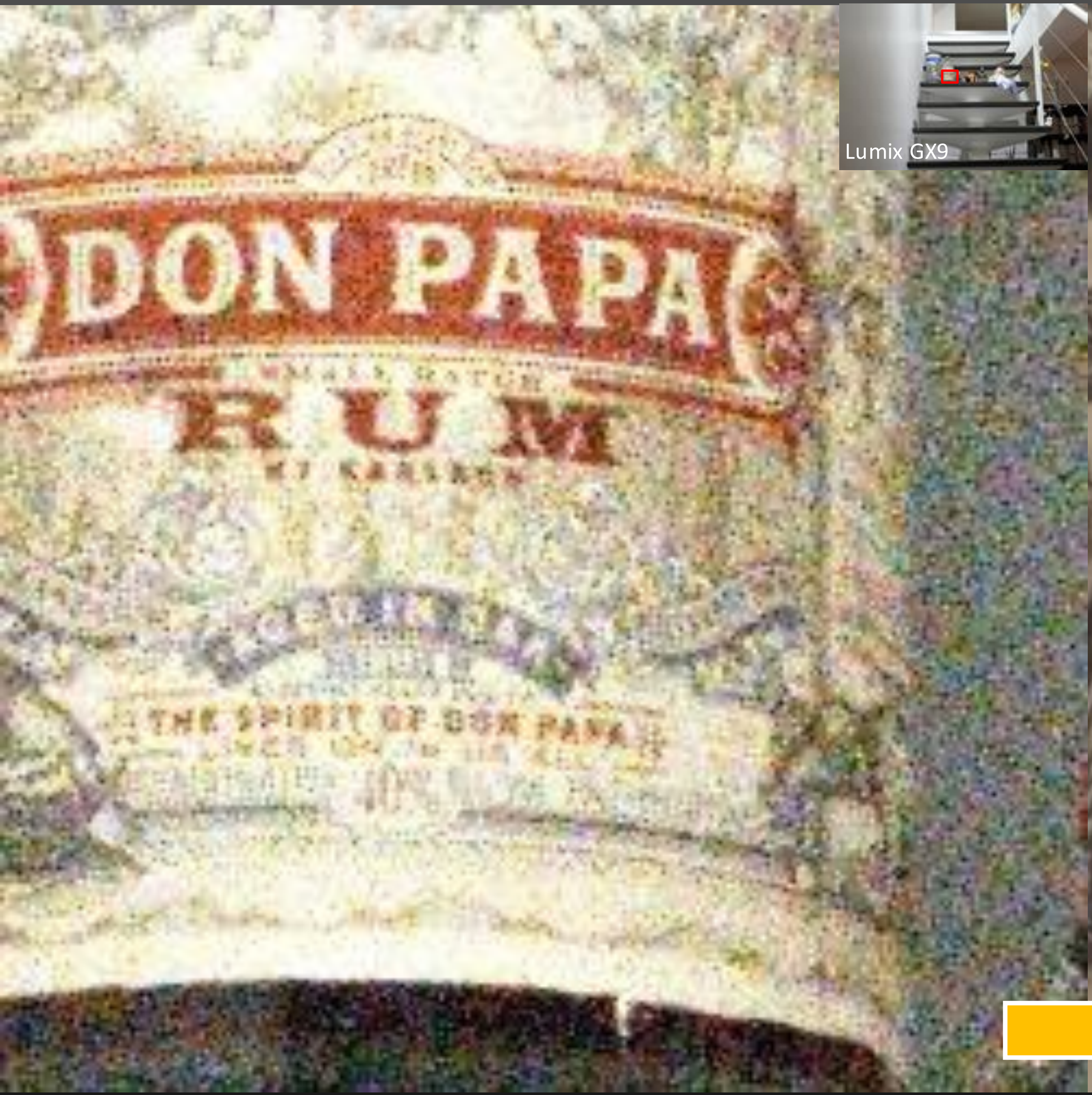
- $\mathbf{z}^t \leftarrow \mathbf{z}^{t-1} - \eta_t\left[U^\top_{\mathbf{p}^{t-1}}(U_{\mathbf{p}^{t-1}}\mathbf{z}^{t-1} - \mathbf{y}) + \mu(\mathbf{z}^{t-1} - \mathbf{x}^{t-1})\right]$ — One step of gradient descent (or a few)
- $\mathbf{p}^t_k \leftarrow \mathbf{p}^{t-1}_k - \left(\mathbf{J}^{t\top}_k\mathbf{J}^t_k\right)^{-1}\mathbf{J}^{t\top}_k\mathbf{r}^t_k$ (3 times) — Gauss-Newton (aka Lucas-Kanade)
- $\mathbf{x}^t \leftarrow f_\theta(\mathbf{z}_t)$ — Plug-and-play approach (small residual U-net)
- Increment μ

K. Gregor, Y. LeCun, "Learning fast approximations of sparse coding", ICML'10

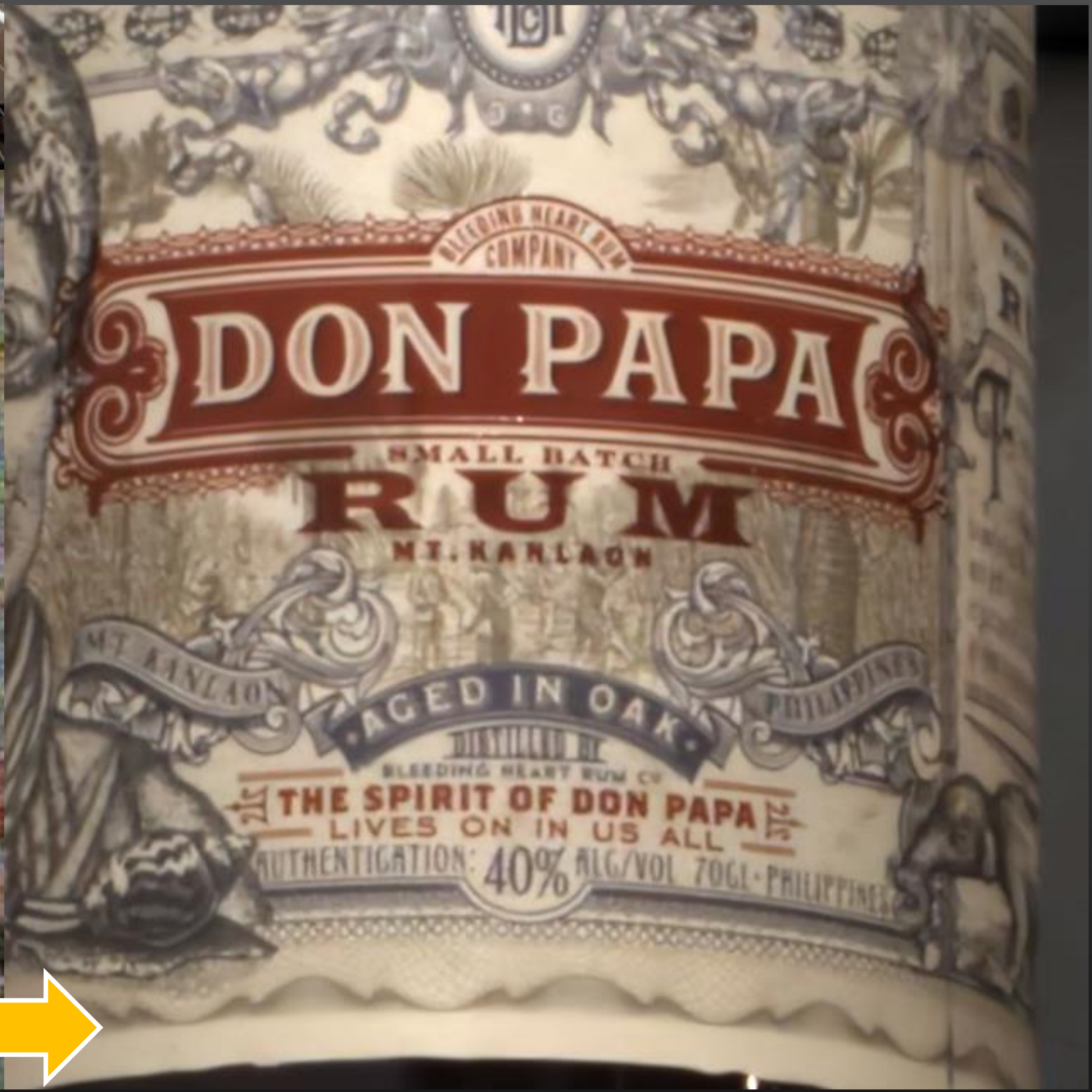
Example

Raw image burst (Lumix GX9)

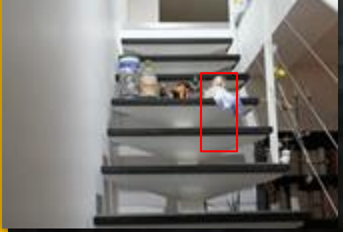
High-quality picture

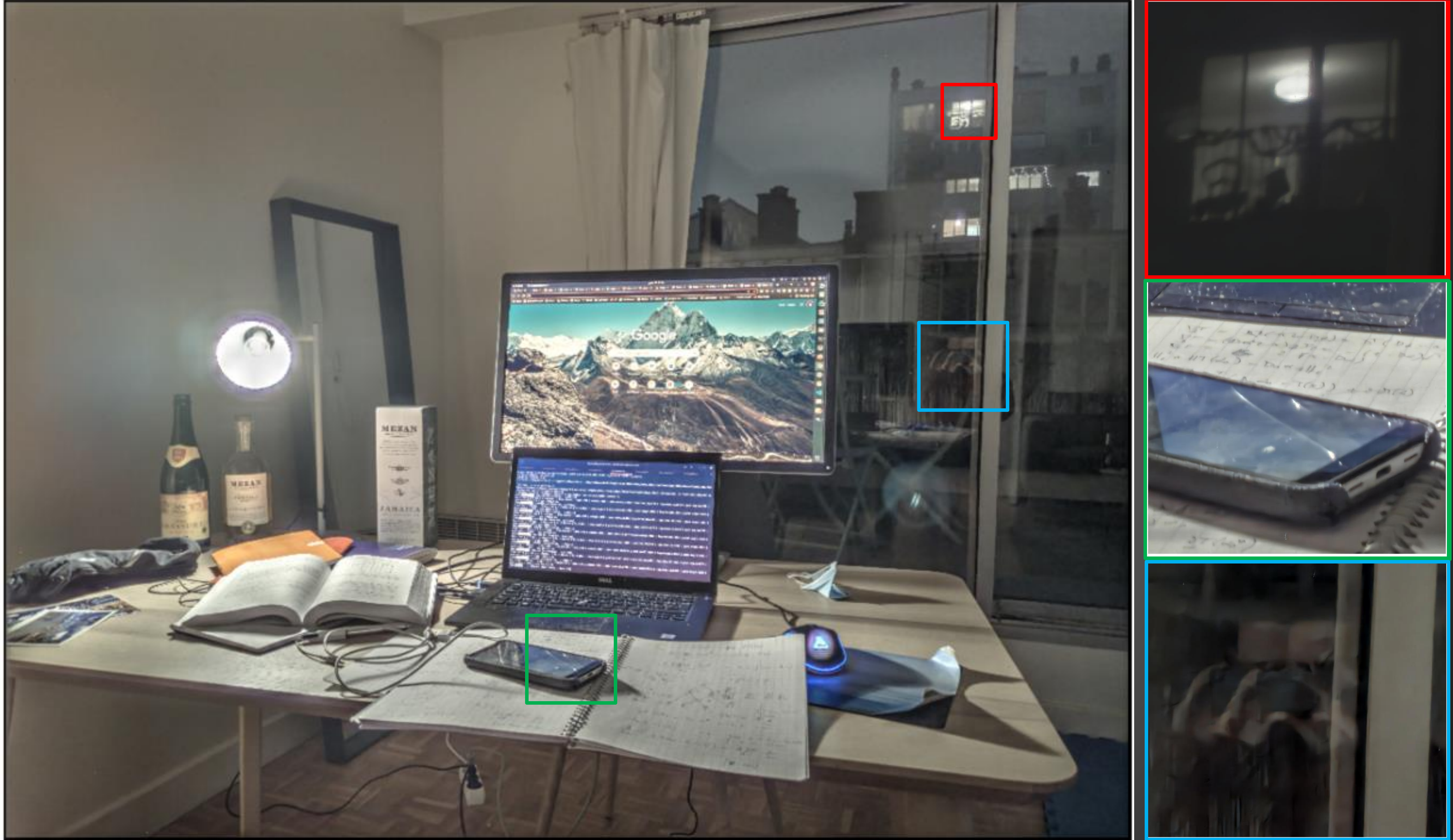


(Small crop of) Burst of raw pictures



(Lecouat et al., ICCV'21)




MEZAN
MEZAN
JAMAICA
Google
DELL

Application: Thermal imaging - denoising + x4 super-resolution, 20 frames
80x62 waveshare IR camera, less than 150Euro