TRaitement des Images pour la Vision Artificielle

Vincent Lepetit

slides in part based on material from Mathieu Aubry, Andrew Zisserman, David Lowe, Jean Ponce

LIGM-Imagine Lab

Instructors

Vincent Lepetit
vincent.lepetit@enpc.fr

Pascal Monasse
pascal.monasse@enpc.fr

+ Teaching assistants for the TP/TD

Don't hesitate to contact us!

École des Ponts
ParisTech

Goals

- Introduce computer vision;
- Introduce important mathematical tools for computer vision;
- Getting use to work with images.

Relation to Other Courses

- At the ENPC
  - Traitement du signal et Analyse spectrale
  - Recherche opérationnelle (flot dans un graphe et coupe minimale)
  - *Machine Learning*

- Preparation for the MVA master (Mathématiques, Vision, Apprentissage).

Planed Schedule

Alternating between lectures and exercise sessions;

2 last sessions dedicated to the projects presentations.

Resources

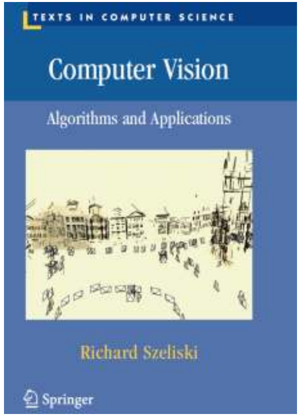
- Slides on Educnet.
- Book online

  http://szeliski.org/Book/

  Include most of the lectures content

  Support for many projects

- More references on Educnet



Evaluation

- TP/TD every 2 weeks
  - 60% of the final grade;
  - In Python;
  - To return for the next session on Educnet.
  - **Do not spend more than 4-5 hours on each TP!**

- Project, group of 3-4 students:
  - 40% of the final grade
  - Reading/summarizing/presenting an important topic in Computer Vision
  - Choice in a list on Educnet.
  - sessions at the end of the lectures and the end of the semester.

TP/TD report

- Send an archive with both your code and a report.
- Make full sentences, be clear.
- Your code must be clean and commented.
- Use explicit names for variables.

- More instructions on Educnet.

Collaboration Policy for the TD/TP

PERSONNAL, NO PLAGIA

- You can discuss the TP/TD but each report/code must be done INDIVIDUALLY.
- Write on your report the people you worked with.
- Plagiarism is easy to detect.

Today

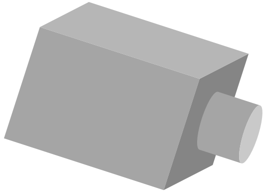
- What is Computer Vision?
  - applications and challenges
- Image formation;
- Image filtering.

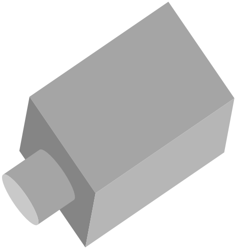
What Is Computer Vision?

processing of digital
images by a computer

“offline” or “real-time”

What Is Computer Vision?





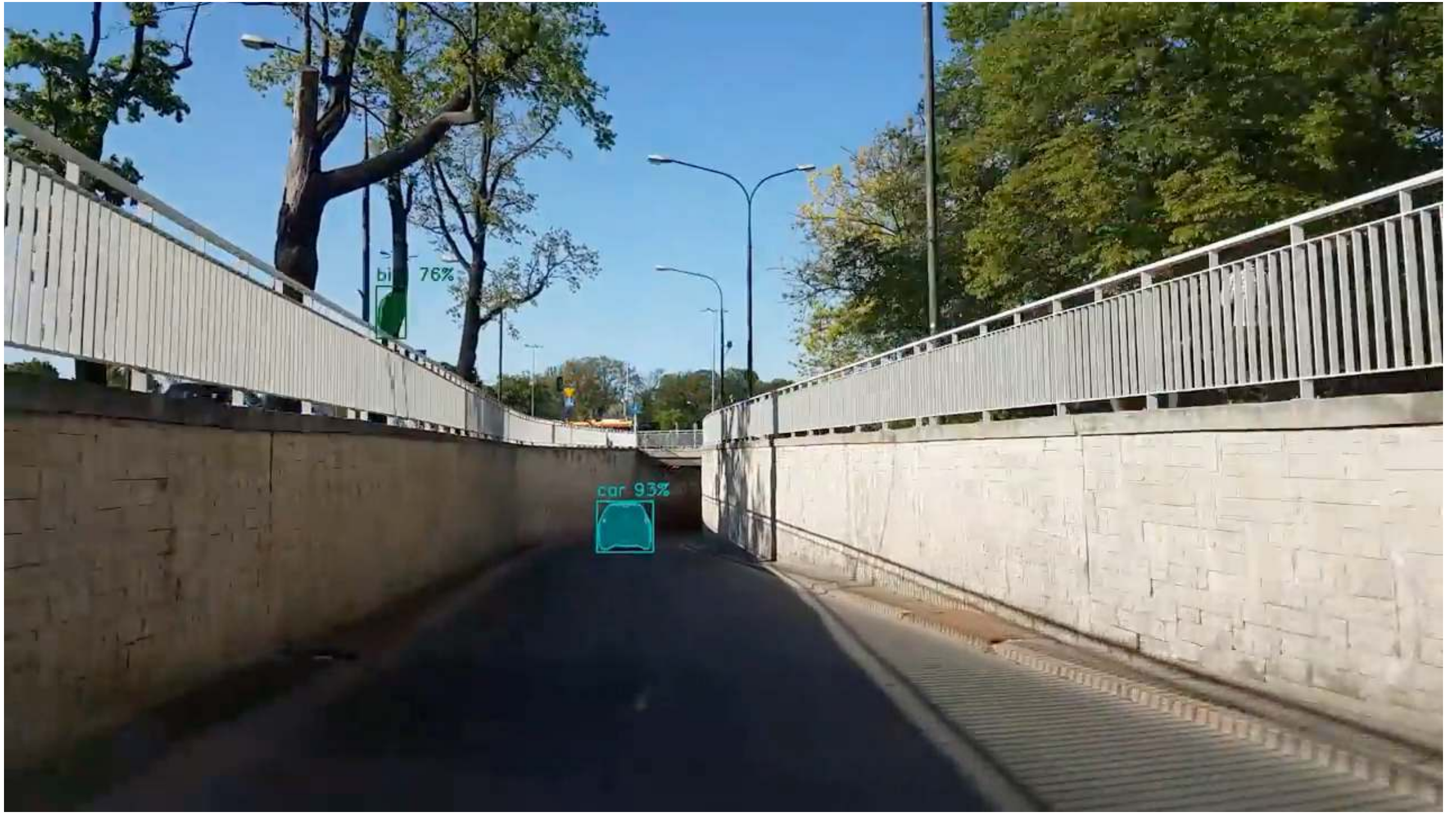
possibly several cameras
or images simultaneously

possibly using a
cluster of computers

What Is Computer Vision?

the camera can be in motion,
the scene can be dynamic

Applications



object detection, recognition, segmentation

Applications

action recognition / scene recognition

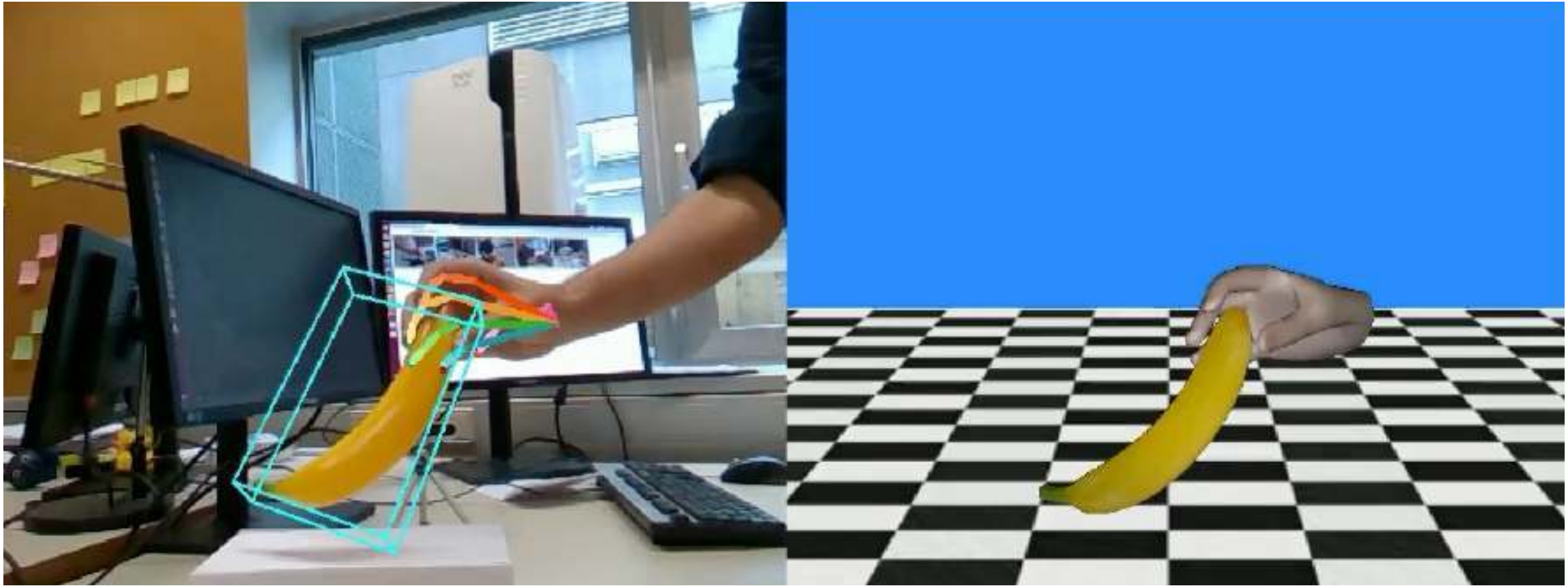
Applications

(video from Pix4D)

3D reconstruction from multiple images [3D geometry]

École des Ponts ParisTech

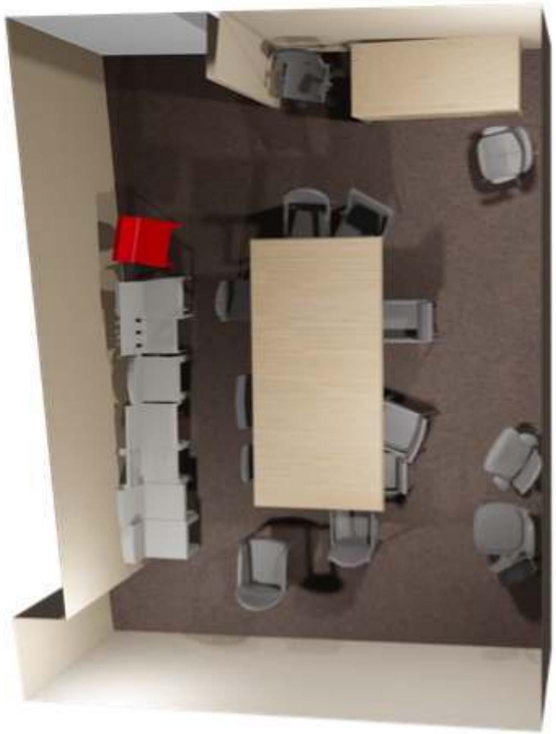
Applications



3D pose estimation from a single image [Machine Learning]

École des Ponts
ParisTech

3D Scene Understanding



from unstructured 3D geometry to 3D “semantic” [3D geometry + Machine Learning]

École des Ponts ParisTech

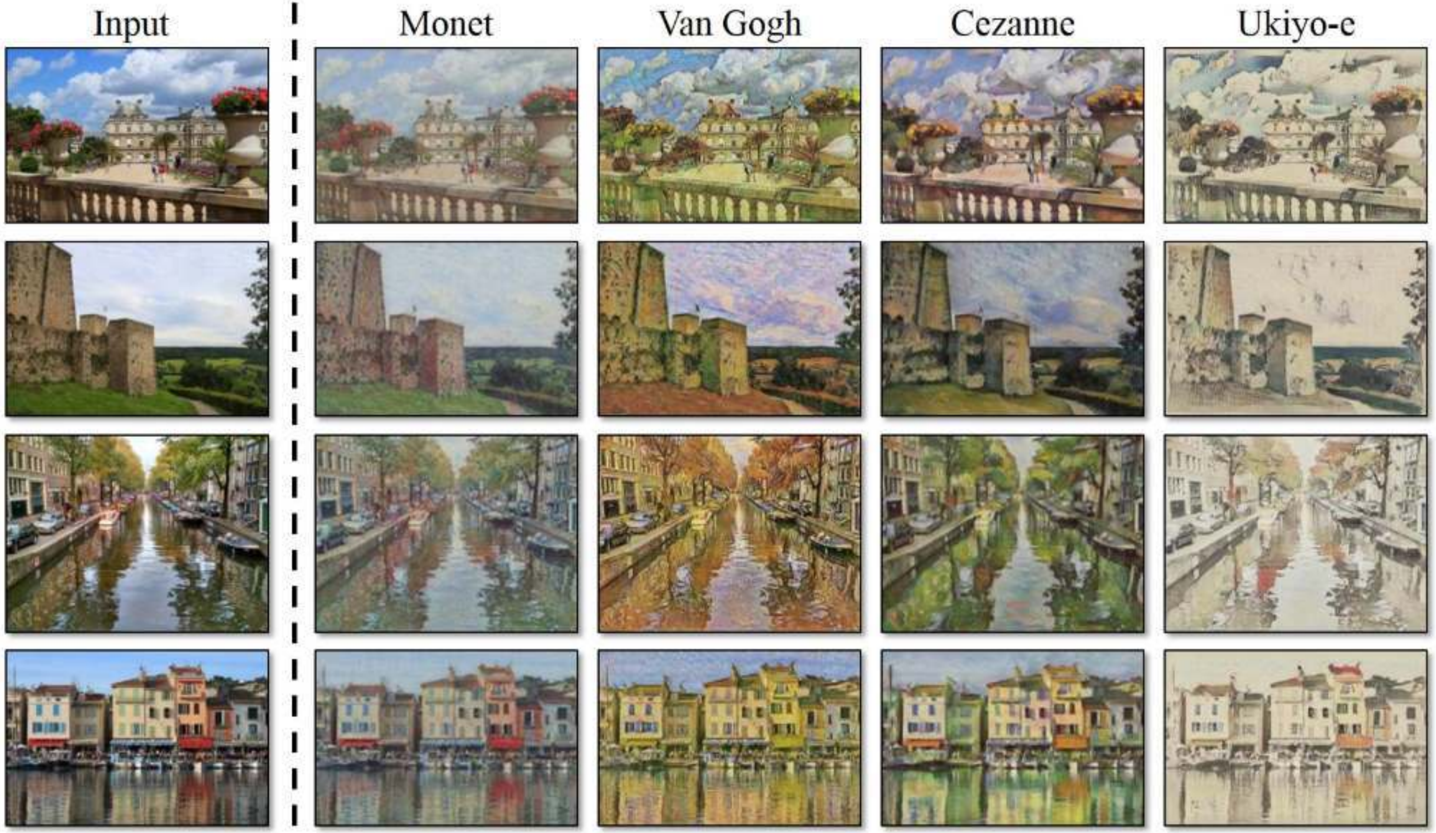
Image and Video Manipulation

Seamless copy-paste:

+ Deep Fakes

École des Ponts
ParisTech

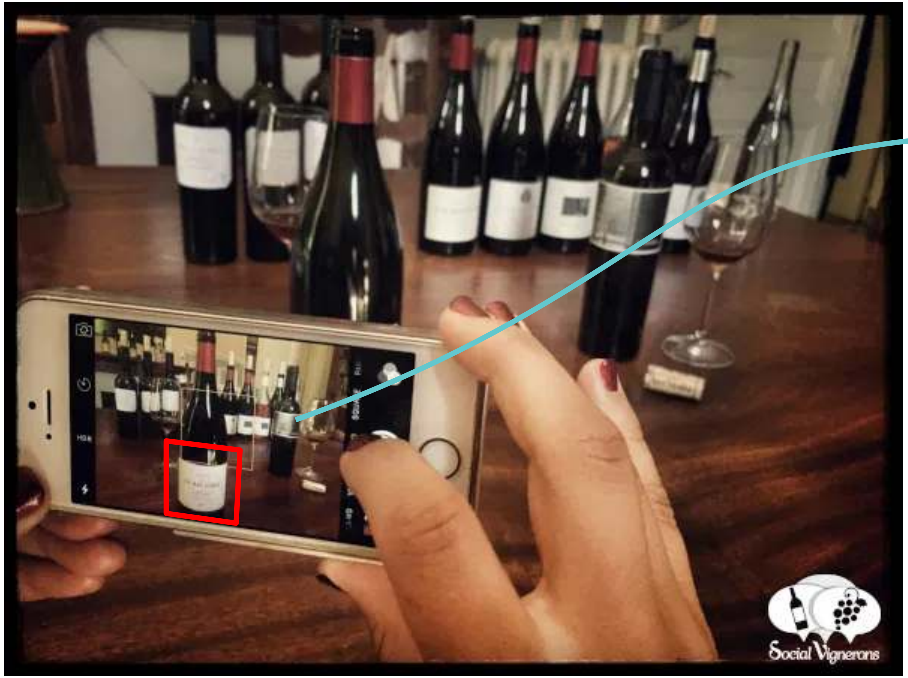
Style Transfer

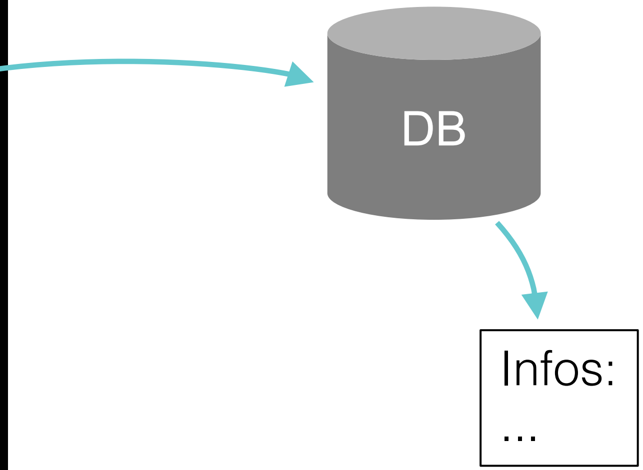


low-level techniques or Machine Learning

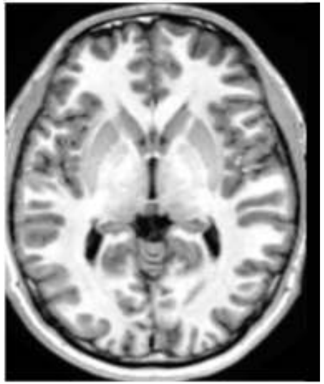
École des Ponts
ParisTech

Image Retrieval

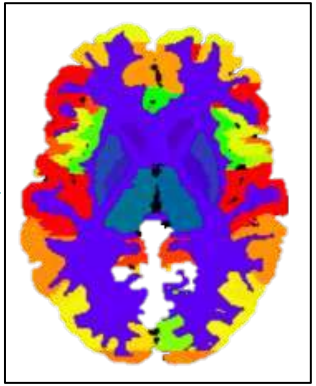




Other Imaging Sensors



MRI

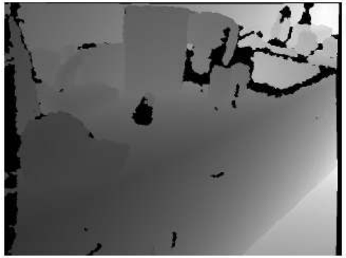


segmentation

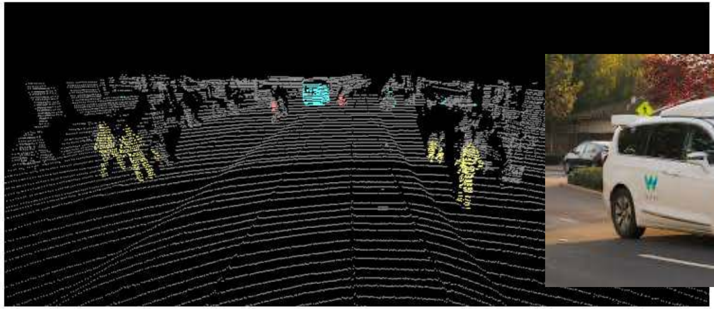
medical imagery

(active) depth sensors

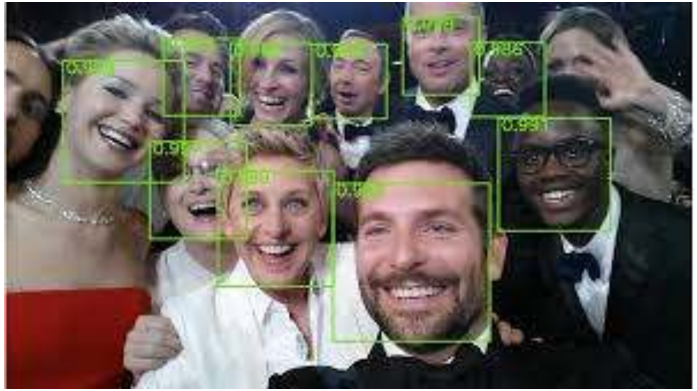
RGB-D cameras

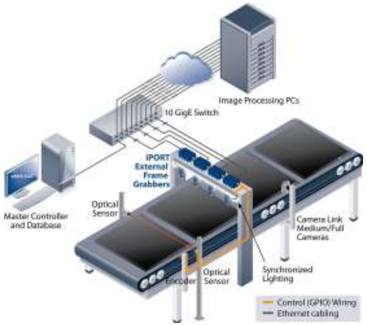


LiDARs



Computer Vision is Already Here

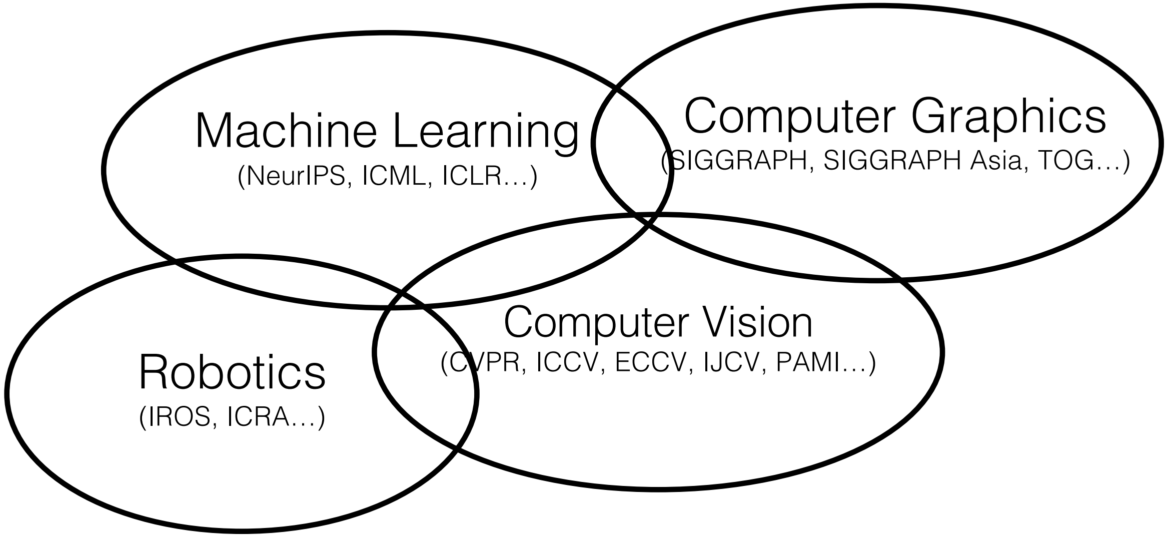




quality inspection

Augmented / Virtual Reality

Vision and Other Fields / Communities



École des Ponts

Today

- What is Computer Vision?
  - applications and **challenges**

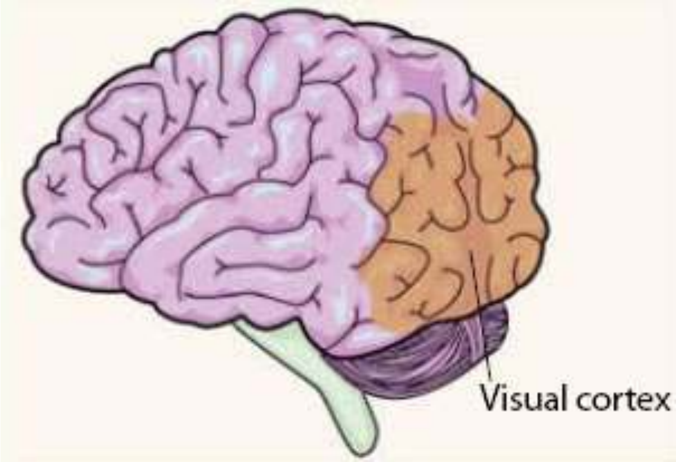
- Image formation;

- Image filtering.

Challenges

Vision is Hard!

Human Vision



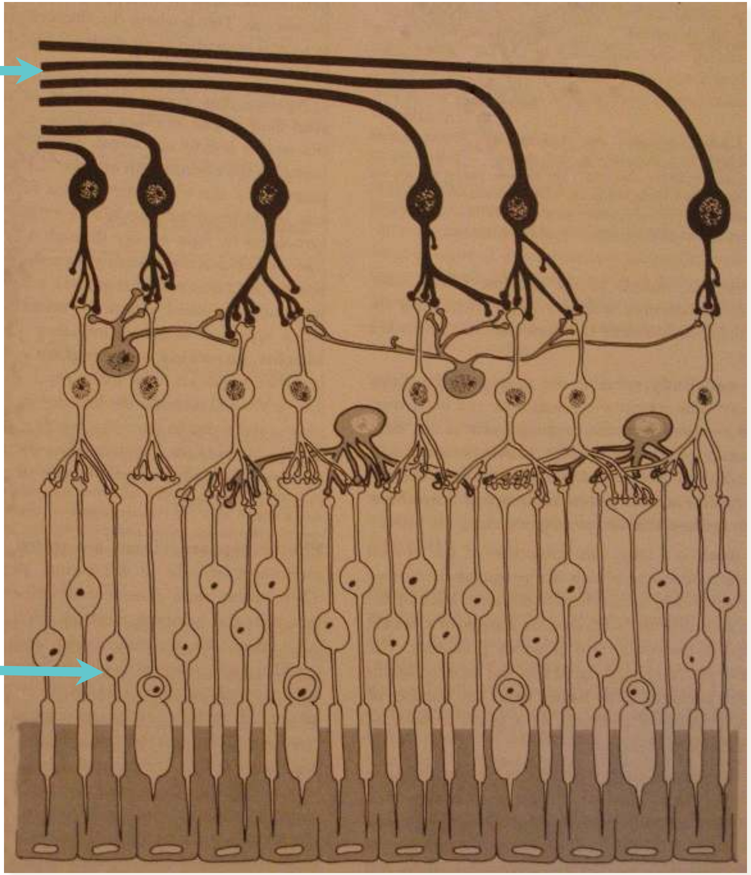
The visual cortex represents about 20-50% of our brain.

Human vision is unconscious (most of the time).

Intuitions about how human vision works are often wrong...

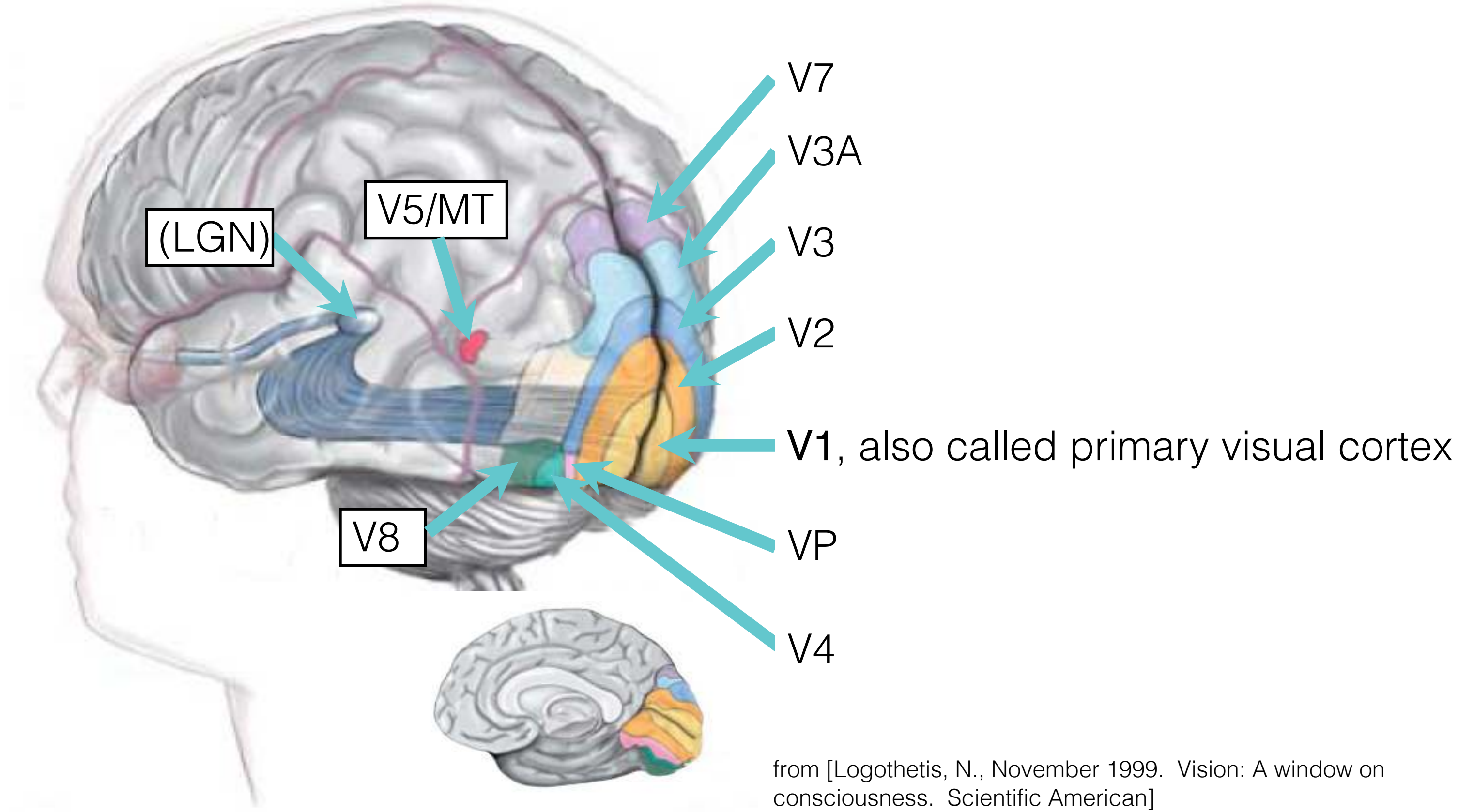
The Retina

Optic Nerve



Receptors

The Visual Cortex



from [Logothetis, N., November 1999. Vision: A window on consciousness. Scientific American]

Why Is Vision So Hard?

Too much data:

A color image of resolution 1000 x 1000 is made of
1000 x 1000 x 3 x 8 = 2.4 10^7 bits.

123	034	089	045	145	178	009	078	044	084	245	190	066	008	055	094	046	098
045	145	178	009	078	066	008	055	123	034	089	059	044	084	245	066	008	055
034	089	045	145	123	034	089	094	046	098	123	034	089	178	009	078	034	009
084	245	190	044	084	055	094	084	245	190	078	044	084	044	084	245	190	123
123	034	089	078	044	084	055	094	046	123	034	089	009	078	044	084	143	162
033	178	055	094	046	098	145	178	009	078	044	084	123	034	089	045	145	178
084	245	190	044	084	055	094	046	098	009	078	044	084	078	123	034	089	056
066	008	055	009	078	009	078	044	034	089	045	145	178	078	044	066	008	055
012	034	089	045	145	178	098	078	123	034	089	034	089	045	145	178	067	034
098	084	245	190	178	009	078	044	084	044	084	245	190	044	084	084	245	190
055	094	046	098	034	089	045	145	178	084	009	078	044	084	245	190	201	206
190	156	123	034	089	009	078	034	089	045	145	123	034	089	009	078	044	084
018	055	094	046	098	078	044	084	034	089	045	044	084	245	190	009	078	075
234	084	245	190	078	044	084	245	190	055	094	046	098	078	044	123	034	089
157	044	084	245	190	046	098	123	034	089	078	044	084	044	084	245	190	012
066	008	055	084	245	190	034	089	045	145	178	009	078	044	044	084	245	190
084	245	190	044	034	089	045	145	178	009	044	084	245	190	044	084	089	043
044	084	245	190	178	009	078	055	044	084	245	190	034	089	044	084	245	190

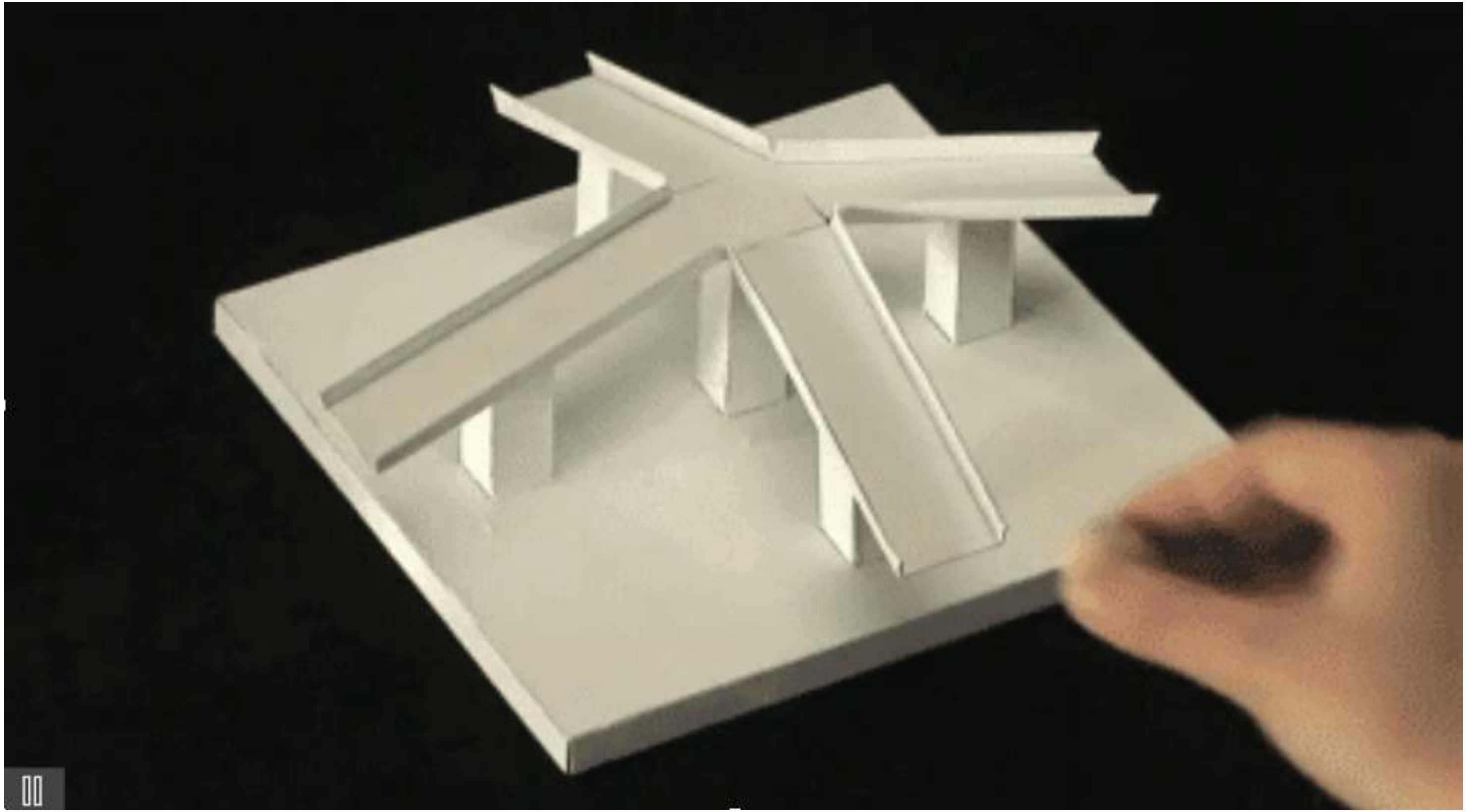
a small image

Why Is Vision So Hard?

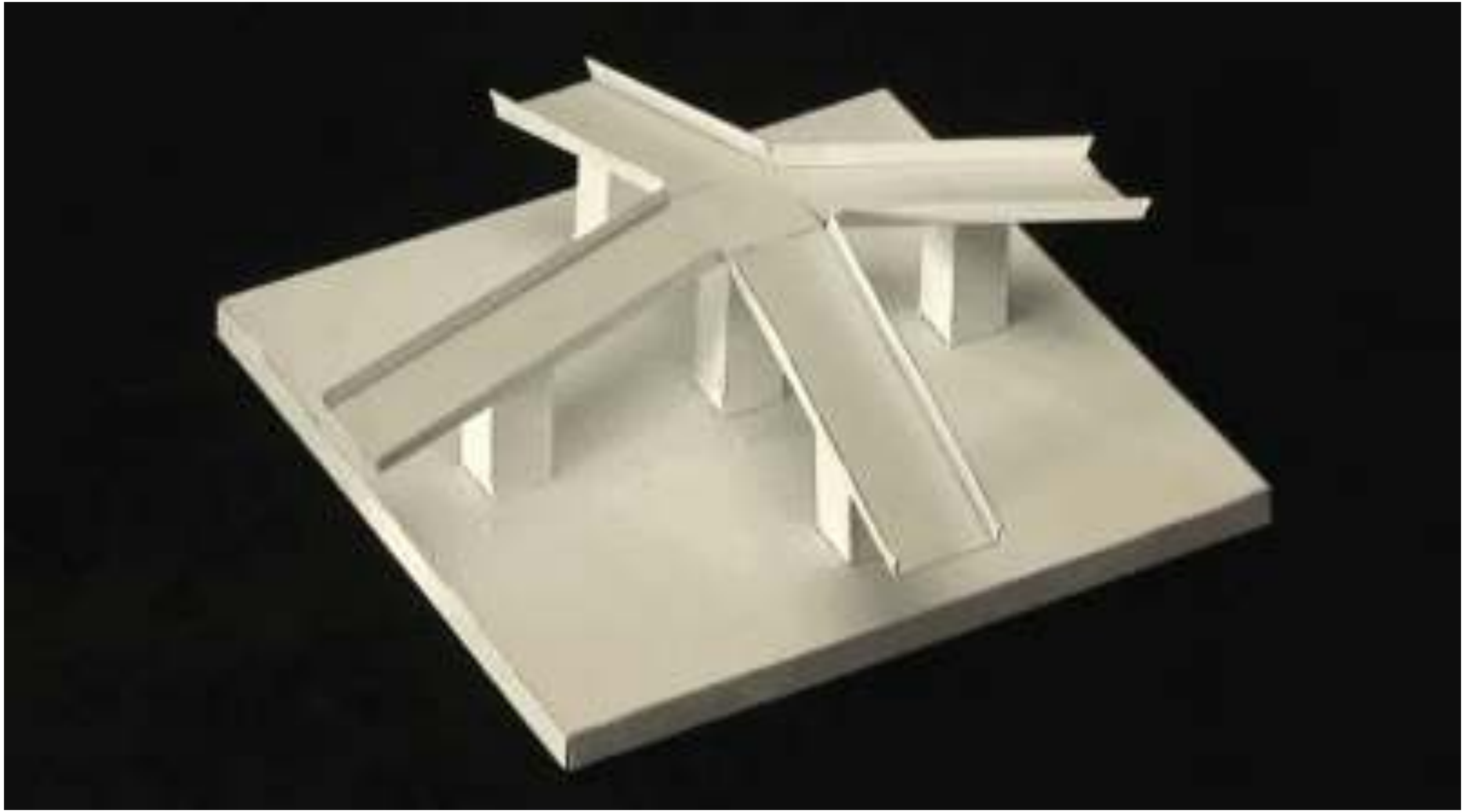
Not enough information

Why Is Vision so Hard?

Why Is Vision so Hard?



Why Is Vision so Hard?

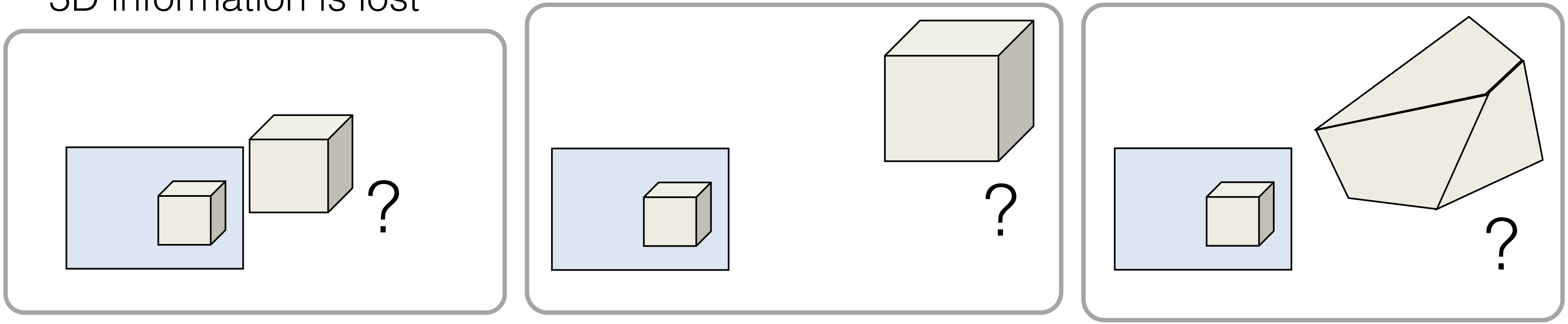


École des Ponts
ParisTech

Why Is Vision So Hard?

Not enough information

3D information is lost

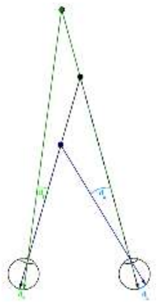


Given an image, there is an infinite family of 3D scenes that could create this image;
How do we should which one is correct?

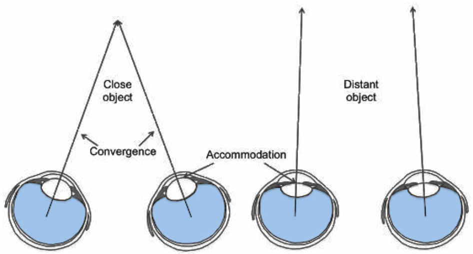
- Context (e.g. car washing video);
- Prior information (e.g. lines tend to be orthogonal or parallel);
- etc.

3D Perception in Humans

binocular cues



from binocular parallax



from convergence

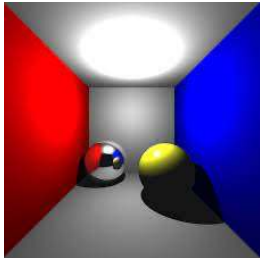
monocular cues

from geometric cues

from context

from motion

from focus and defocus



from shading and cast shadows

from atmospheric effects

Why Is Vision So Hard?

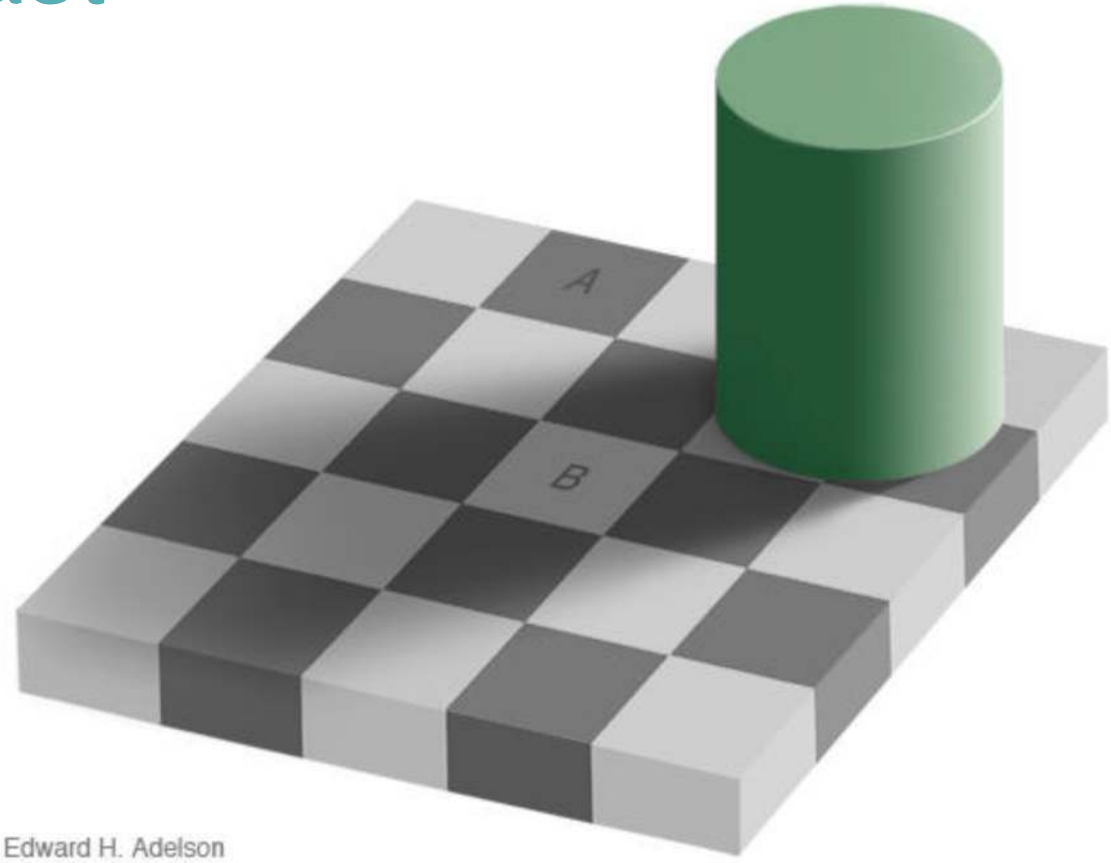
Not enough information

Why Is Vision So Hard?

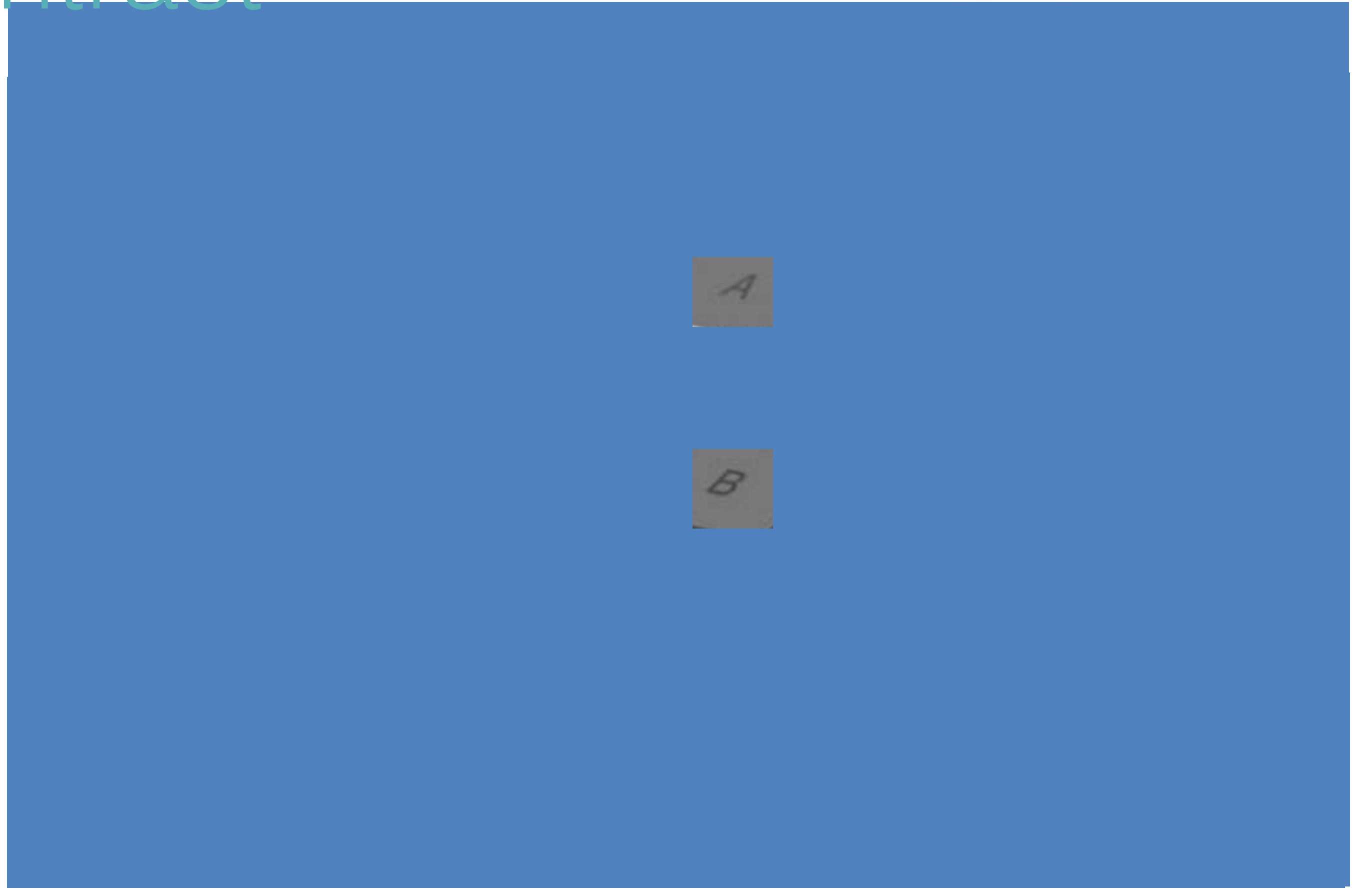
Not enough information

The crater/dome illusion;
We are expecting the ground to be below us, and the light coming from above.

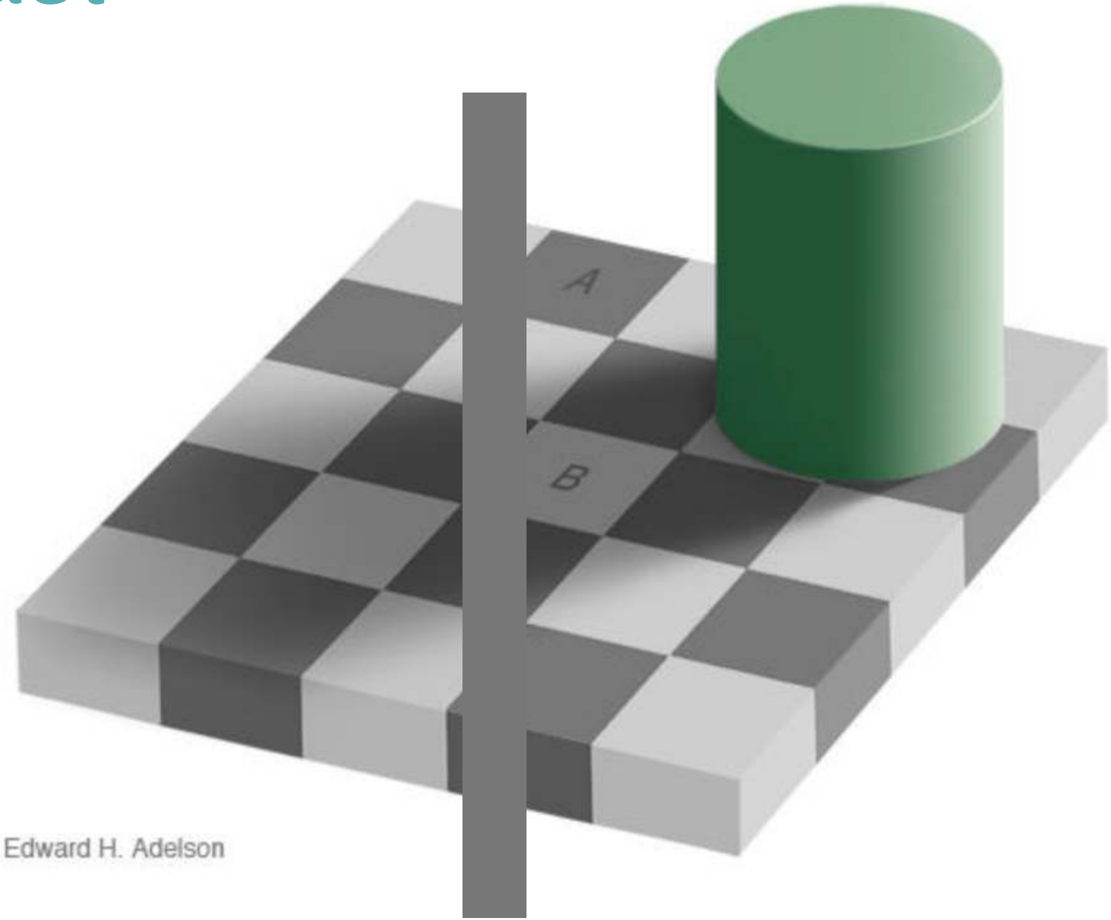
Contrast



Contrast



Contrast



Why Is Vision So Hard?

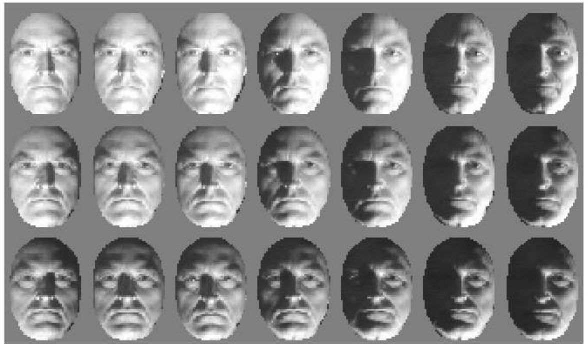
Not enough information

Partial occlusions.

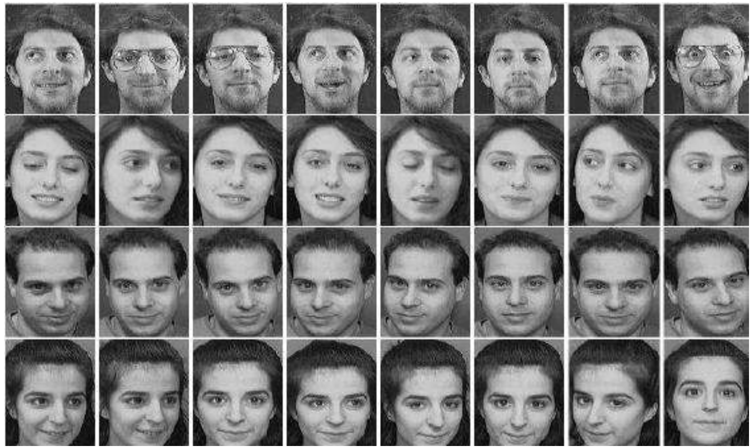
Why Is Vision So Hard?

"Varying" information

different orientations



different lights



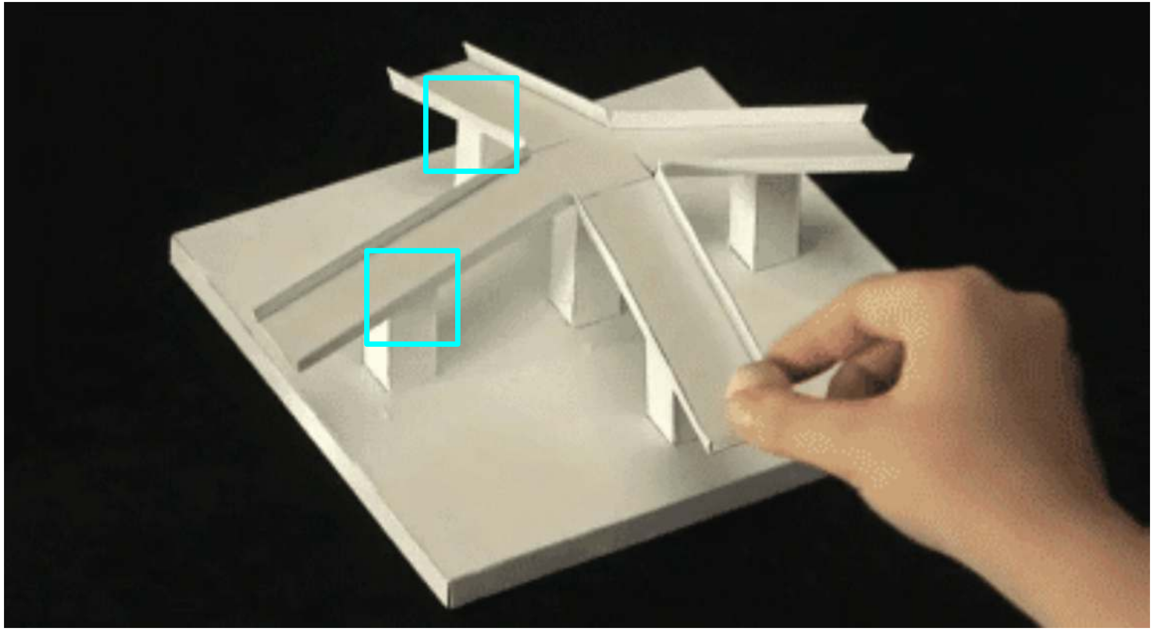
different deformations

Why Is Vision So Hard?

Intra-class variations

Complex Light Effects

Contour "Illusions"



École des Ponts
ParisTech

One of the abilities of our visual cortex:

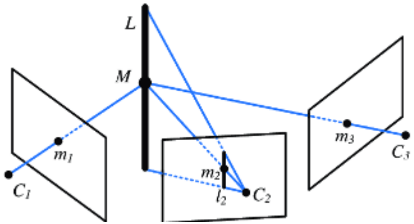
- Fast inference for “typical” scenes;
- Unusual scenes can still be understood with more time.

École des Ponts ParisTech

Computer Vision

- We know it is possible.
- We know it is difficult.
- We don't know how to do it. Well, we are starting to know...

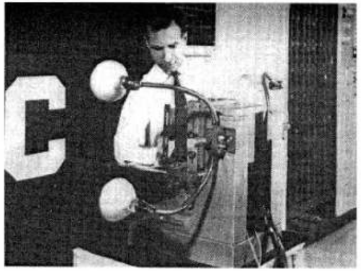
A Short History of Computer Vision



~2000: beginning of Machine Learning being used in Computer Vision

1966: MIT undergrads project

~1958: Perceptron



~1992: first Deep Learning systems in Computer Vision

1980s-:
- low-level Computer Vision (edge, segment, keypoint detection, etc.);
- [3D] object tracking;
- ...

~2010: new beginning of Deep Learning in Computer Vision

MIT undergrads project, 1966

”The primary goal of the project is to construct a system of programs which will divide a vidisector picture into regions such **as likely objects, likely background** areas and chaos. We shall call this part of its operation FIGURE-GROUND analysis. It will be impossible to do this without considerable **analysis of shape and surface properties**, so FIGURE-GROUND analysis is really inseparable in practice from the second goal which is REGION DESCRIPTION. The final goal is OBJECT IDENTIFICATION which will actually name objects by matching them with a vocabulary of known objects.”

This course

- Image and signal processing, Fourier analysis in Vision
- Introduction to 3D reconstruction
- Energy minimization, graph cuts, MRFs for segmentation and stereo
- Overview of many important topics/methods (projects)
- Little recognition (except in projects) to minimize overlap with ML

Today

- What is Computer Vision?
  - applications and challenges
- Image formation;
- Image filtering.

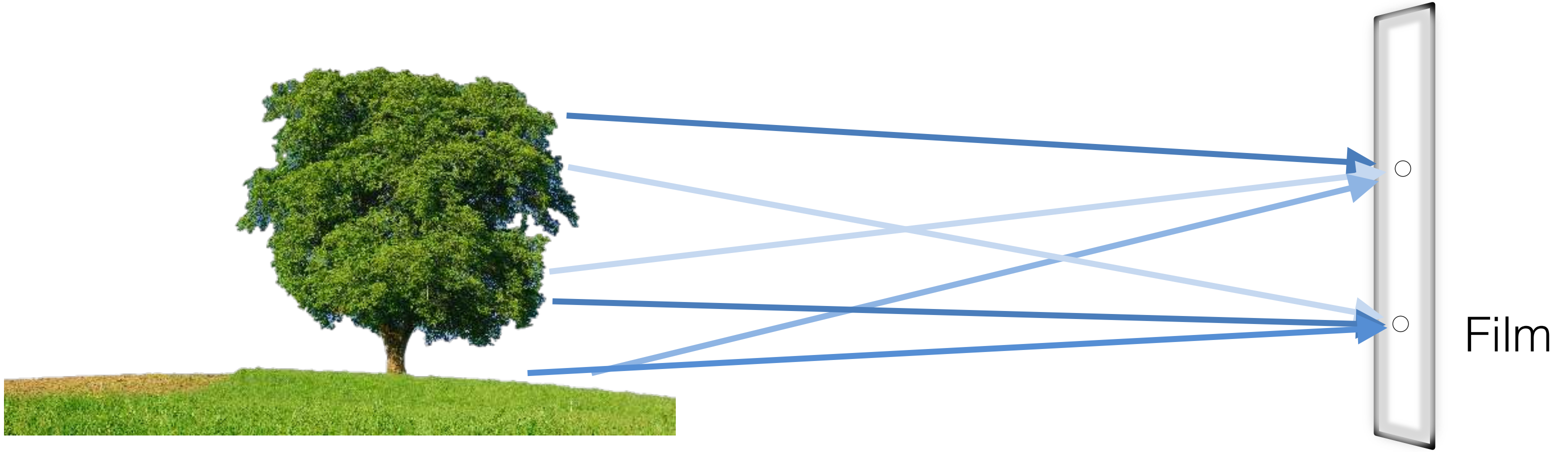
Image Formation

Image Formation

Let’s design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

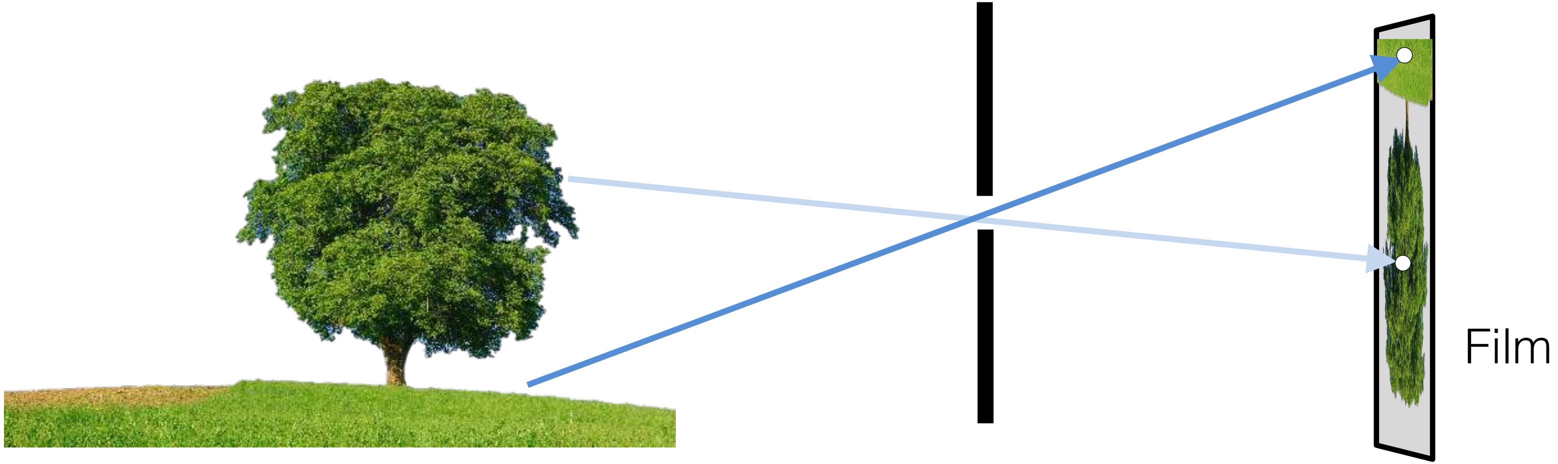
Image Formation



Let’s design a camera

– Idea 1: put a piece of film in front of an object
– Do we get a reasonable image?

Image Formation: Pinhole Camera

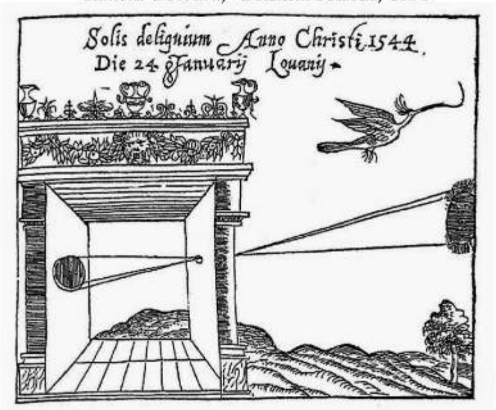


Idea 2: add a barrier to block off most of the rays

- This reduces blurring
- The opening is known as the **aperture**

Camera Obscura

Camera Obscura, Gemma Frisius, 1558

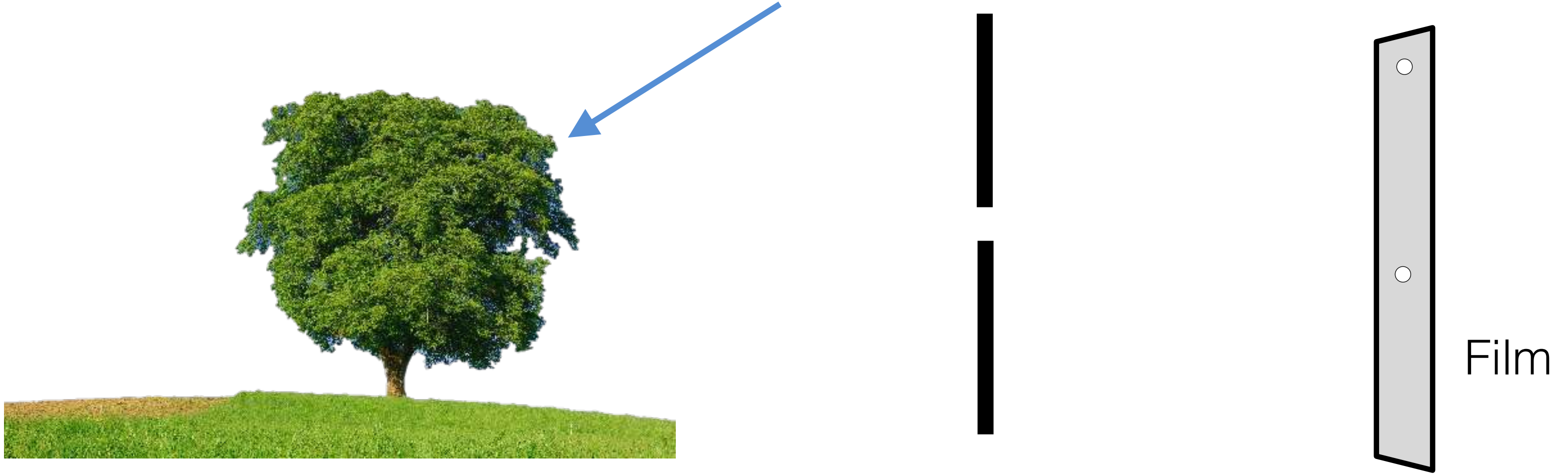


already mentioned in the 5th century in China

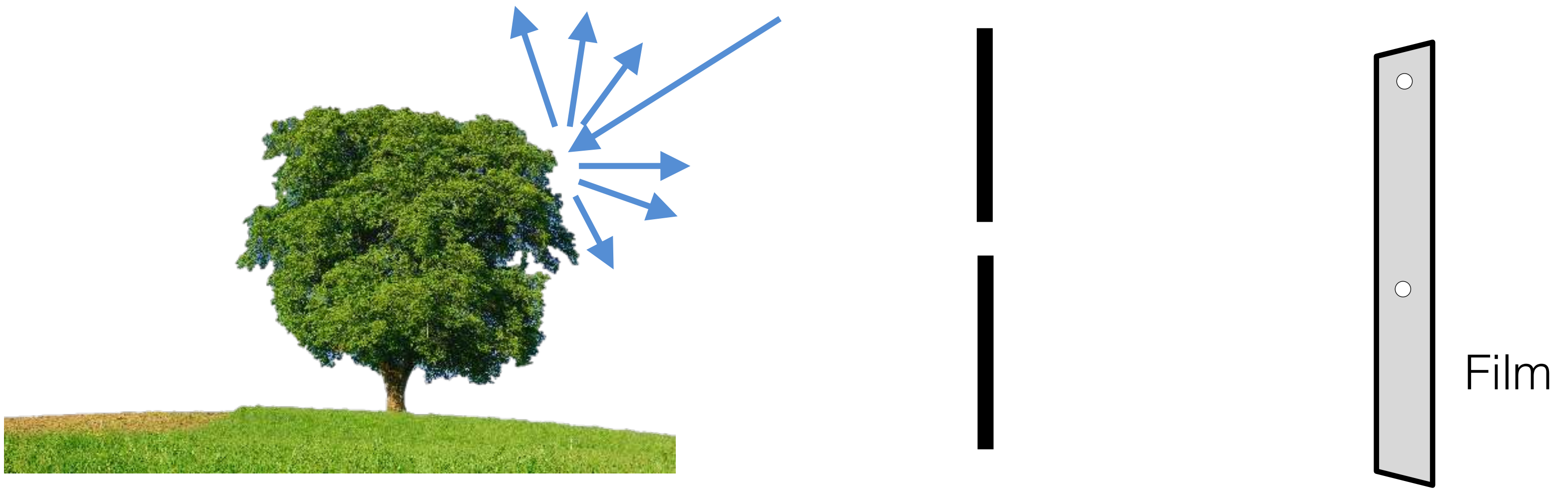
Accidental Cameras

What is happening?

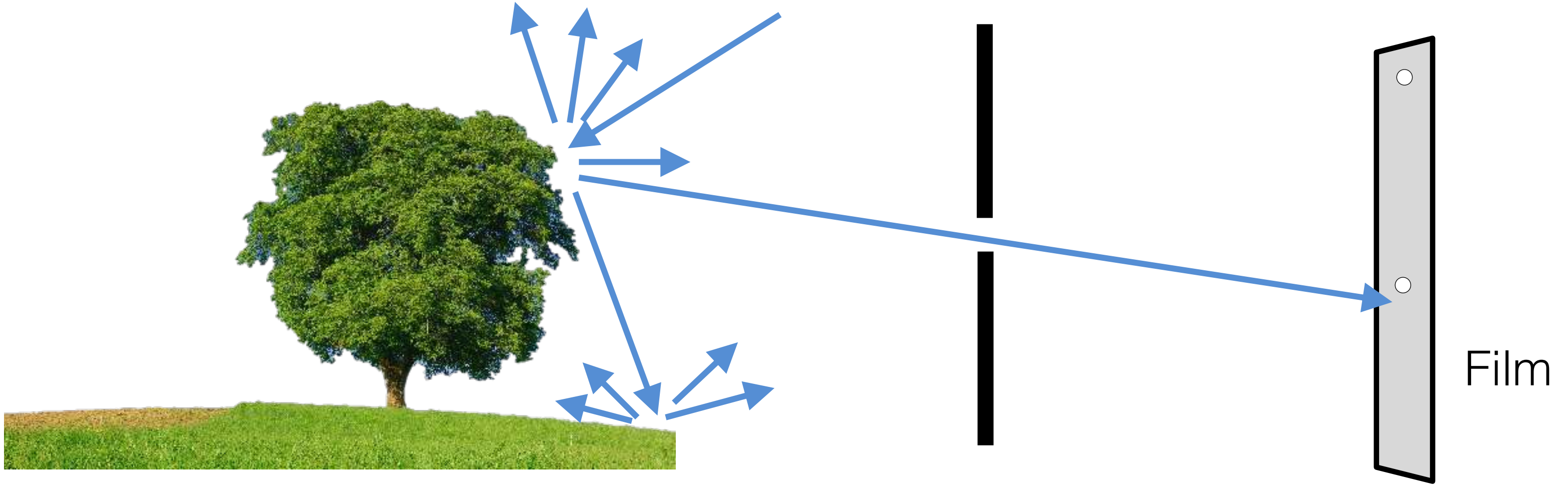
From Light to Sensor



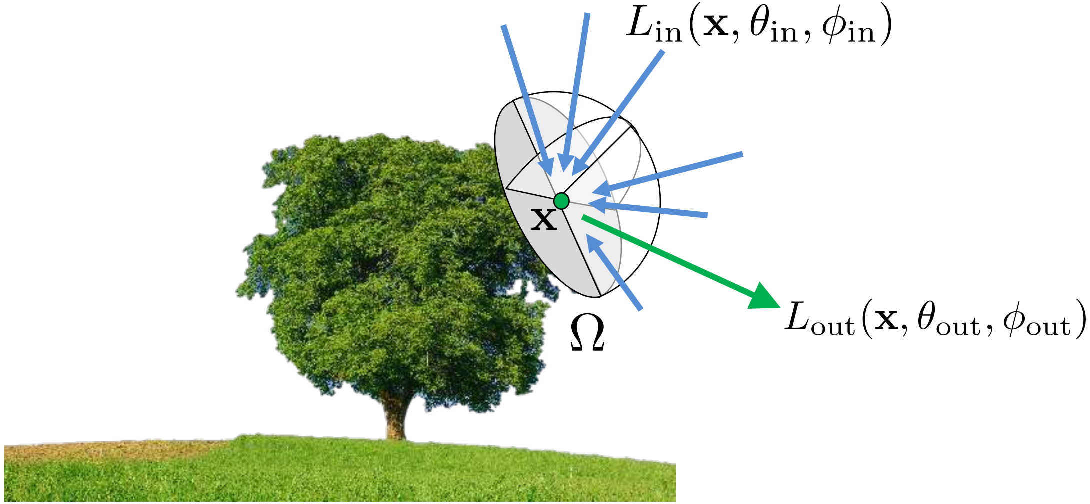
From Light to Sensor



From Light to Sensor



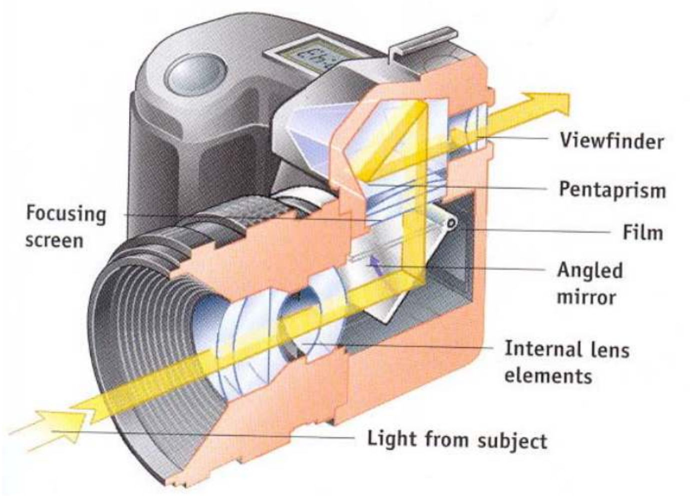
Modelling Interaction between Light and Surfaces



Bidirectional Reflectance Distribution Function (BRDF):
the ratio of the radiance in the outgoing direction to the incidence

$$L_{\text{out}}(\mathbf{x}, \theta_{\text{out}}, \phi_{\text{out}}) = \int_{\Omega} \underbrace{\rho(\theta_{\text{out}}, \phi_{\text{out}}, \theta, \phi)}_{\text{BRDF}} L_{\text{in}}(\mathbf{x}, \theta, \phi) \cos\theta \, d\theta \, d\phi$$

Cameras



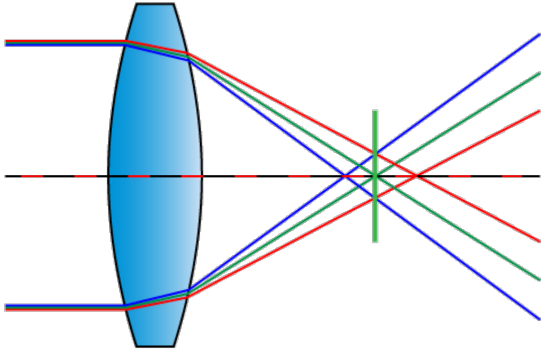
Lenses:

- allow to focus the light
- Introduce artefacts: chromatic aberration, distortion, vignetting, etc…

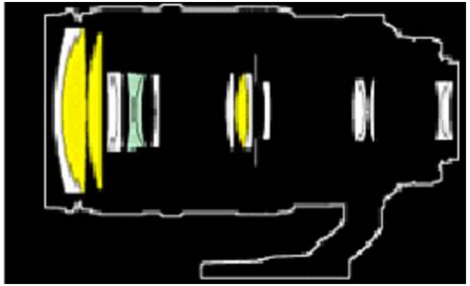
École des Ponts ParisTech

Chromatic Aberration

Different wave lengths are refracted at different angles:



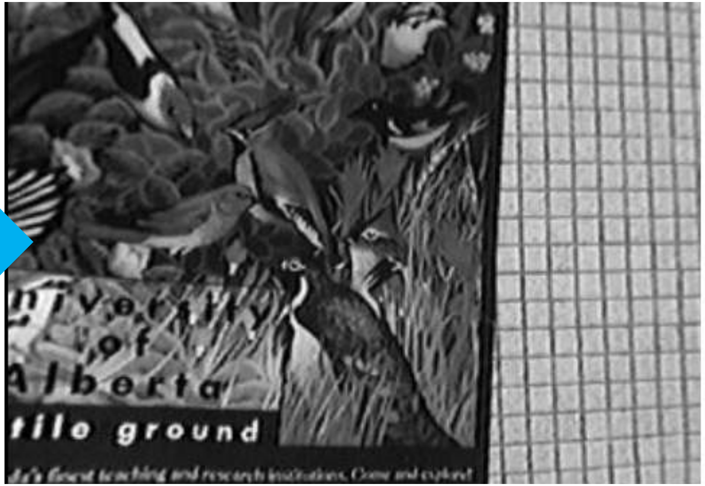
This is the main reason lenses are so complex:



Distortion

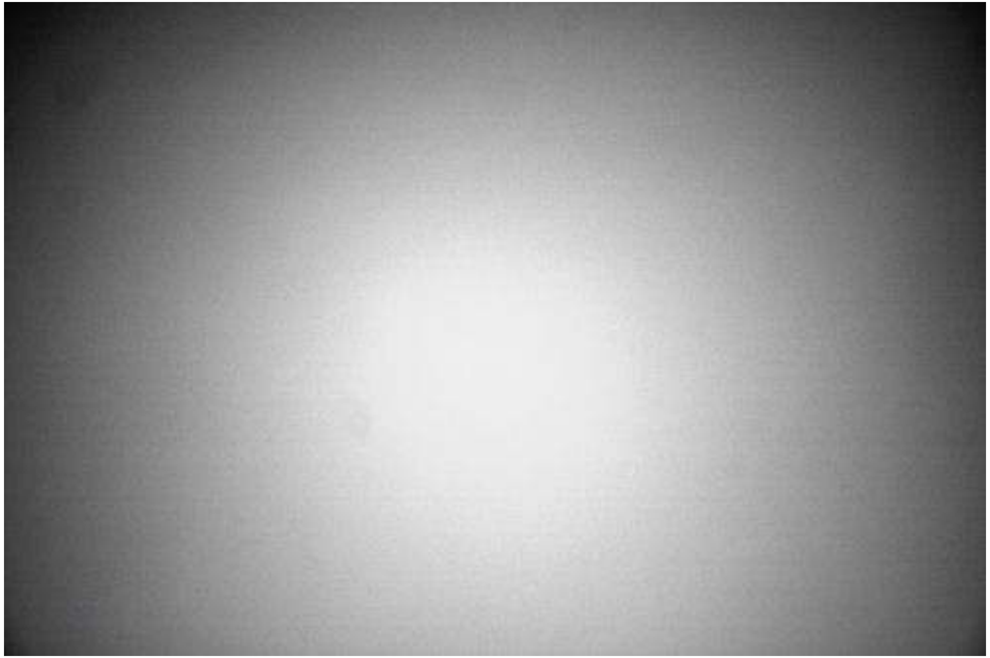
image distorted by the camera lenses

2D transformation



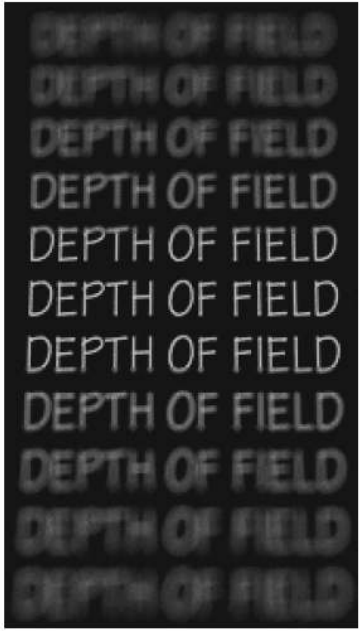
undistortion possible with a 2D transformation

Vignetting



Also due to the lenses: The image tends to be brighter at the center than on the borders

Depth of Field



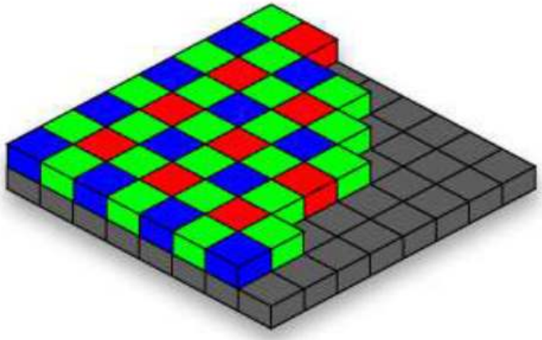
http://www.cambridgeincolour.com/tutorials/depth-of-field.htm

Motion Blur

- Exposure time:
  - The recording is not done instantly;
  - During the exposure time, the camera can move or the scene can change, leading to motion blur.

From Ideal to Real Camera…

- CCD sensor:
  - Discrete → quantization;
  - Bayer filter mosaic;
  - Limited dynamic range → saturation;
  - Sensor noise.



Bayer filter mosaic

From Ideal to Real Camera…

Rolling shutter:

on cheap sensors, the image is captured line by line.

Dynamic Range

(except from very expensive ones,) cameras have a much shorter dynamic range as the human eye:

Mark Fairchild's HDR Photographic Survey

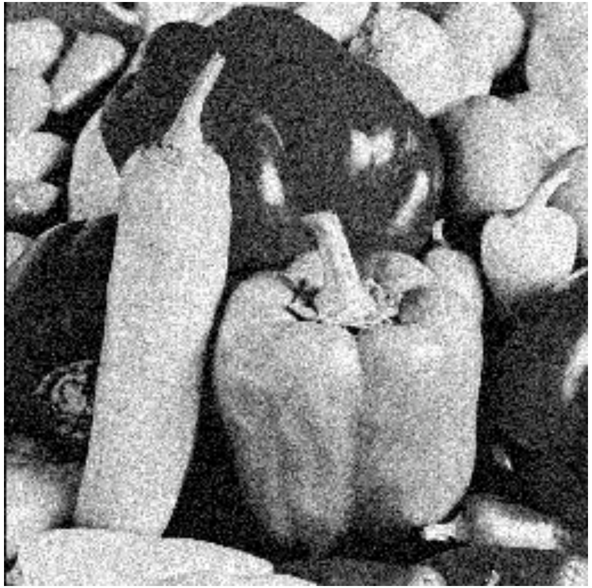
École des Ponts ParisTech

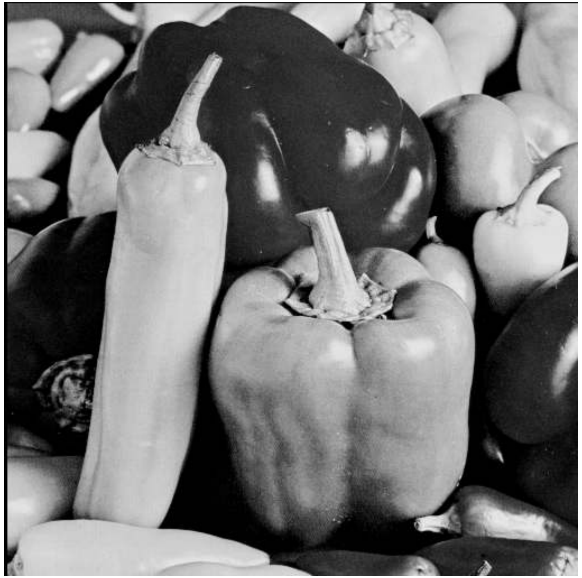
Today

- What is Computer Vision?
  - applications and challenges
- Image formation;
- Image filtering.

Image Filtering

example for denoising:





Edge Detection

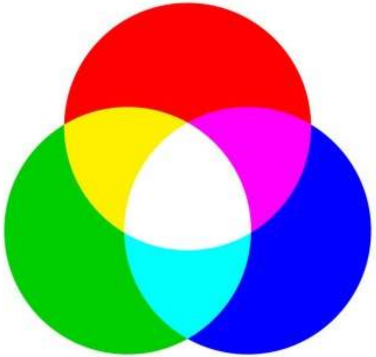
École des Ponts
ParisTech

Let’s Formalize

Captured images are **discrete:** $I : \{0 \dots w-1\} \times \{0 \dots h-1\} \to \{0 \dots 255\}^{(3)}$

Grayscale image: 8 bits per pixel. 0 → noir, 128 → gray, 255 → white.

Color image: 3x8=24 bits per pixel. In RGB space (Red-Green-Blue):
(255, 0, 0) → red; (0, 255, 0) → green; (0, 0, 255) → blue
(255, 255, 0) → yellow; (255, 0, 255); (0, 255, 255)
(255, 255, 255) → white
Other color representations exist (Lab, HSV…).



École des Ponts ParisTech

Let's Formalize

Captured images are **discrete:** $I : \{0 \ldots w-1\} \times \{0 \ldots h-1\} \rightarrow \{0 \ldots 255\}(^3)$

After some computations, it is possible to obtain images with floating point values for each pixel.

Image Noise

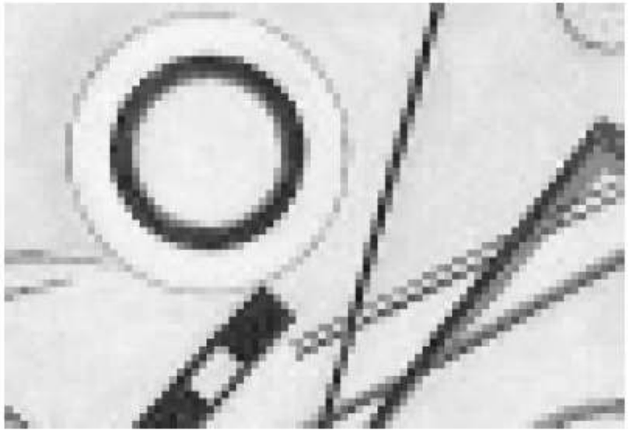
- Model of an image with noise: $O = I + N$

- Real photos tend to have noise:

École des Ponts
ParisTech

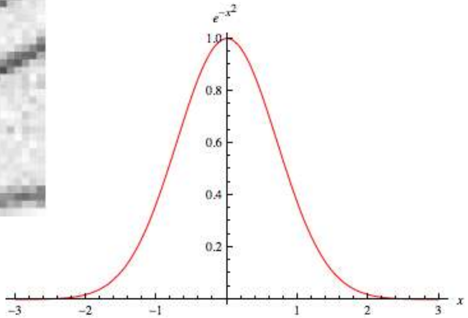
Noise Models

Model of an image with noise: $O = I + N$



Ideal image I

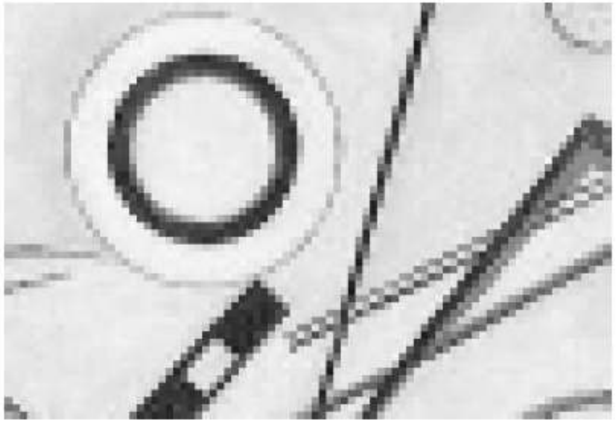
+ Gaussian noise N



N

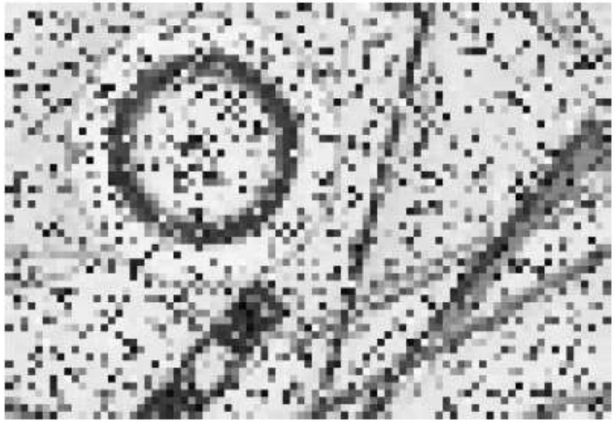
Salt-and-Pepper/Impulse Noise

Some pixels are randomly assigned black or white



Ideal image I

+ Gaussian noise N



+ Salt-and-Pepper noise

Convolutions (1D)

Continuous convolution in 1D:

$$f, g : \mathbb{R} \rightarrow \mathbb{R}$$

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(u)g(x-u)du = \int_{-\infty}^{+\infty} f(x-u)g(u)du = (g * f)(x)$$

Discrete convolution in 1D:

$$f, g : \mathbb{Z} \rightarrow \mathbb{R}$$

$$(f * g)(n) = \sum_{m=-\infty}^{+\infty} f(m)g(n-m) = \sum_{m=-\infty}^{+\infty} f(n-m)g(m) = (g * f)(n)$$

..if the integrals/sums exist. In our case, f and g are compactly supported and bounded, so the integrals/sums exist.

Discrete Convolution in 2D

$$(f * g)(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(i-k, j-l) g(k,l)$$

$$= \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(i+k, j+l) g(-k,-l)$$

Used *e.g.* for image filtering, and in Convolutional Neural Networks...

Can be done efficiently with Fourier Transform.
Can now also be implemented very efficiently on GPUs.

Discrete Convolution in 2D

$$(f * g)(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(i+k, j+l) g(-k, -l)$$

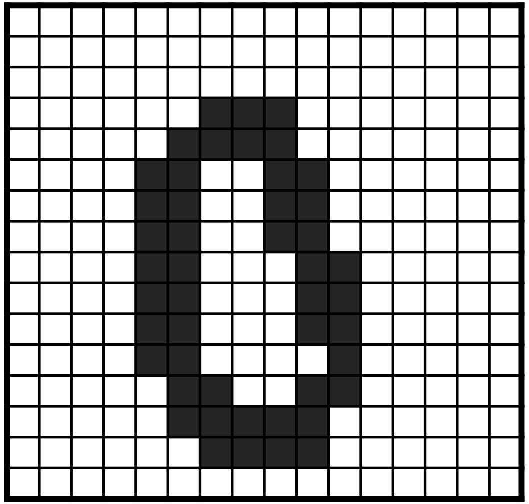
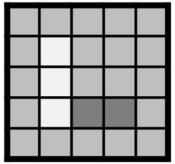
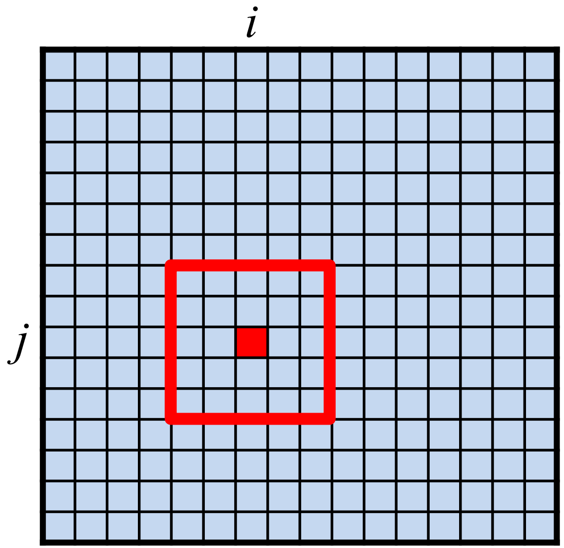


Image f



Linear filter g



Result $f * g$?

Discrete Convolution in 2D

$$(f * g)(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(i+k, j+l) g(-k,-l)$$

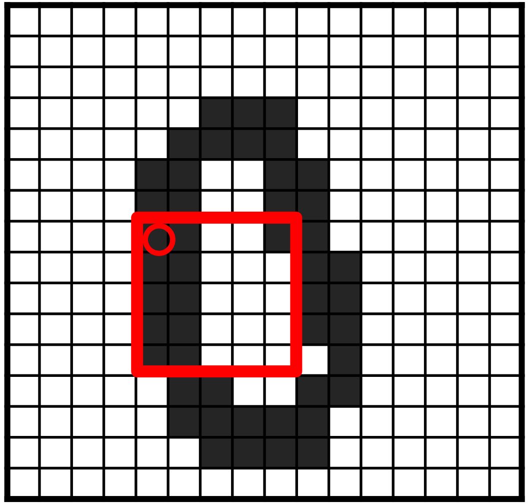
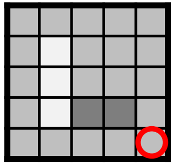
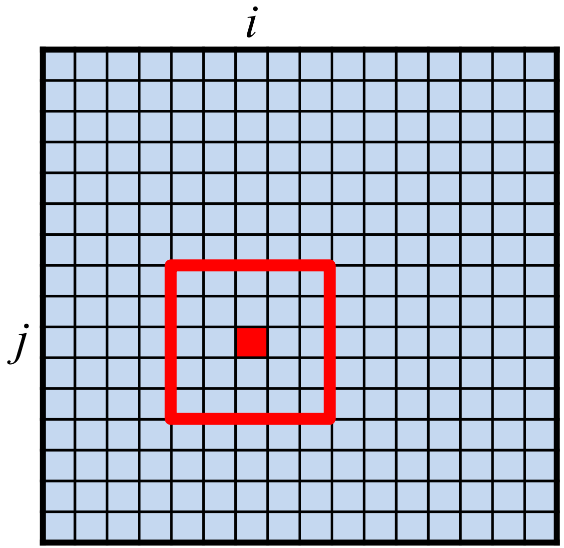


Image f



Linear filter g



Result $f * g$?

Discrete Convolution in 2D

$$(f * g)(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(i+k, j+l) g(-k,-l)$$

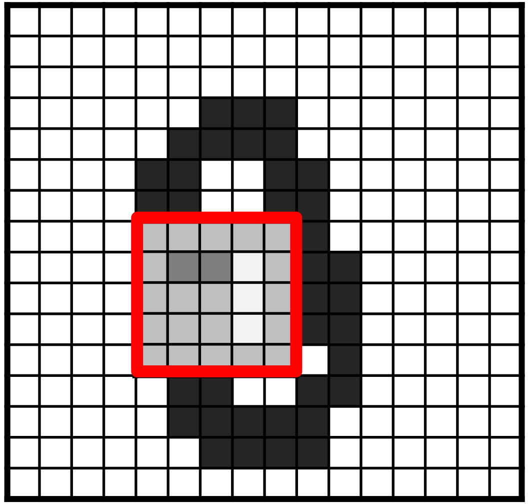
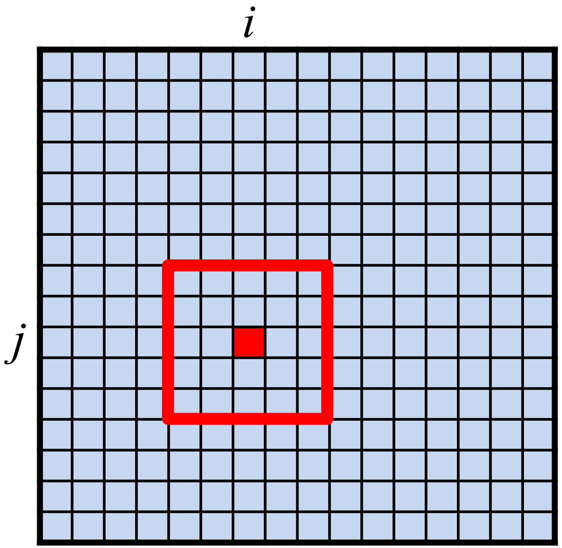


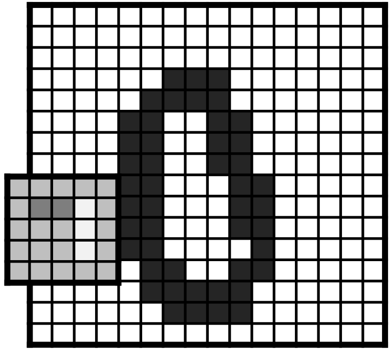
Image f

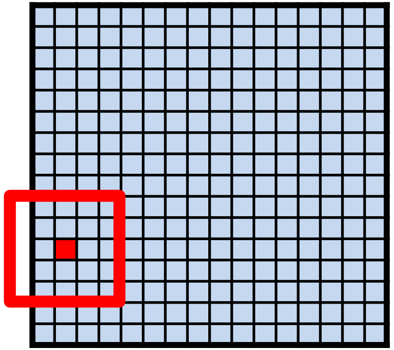


Result $f * g$?

École des Ponts ParisTech

What To Do On the Borders?

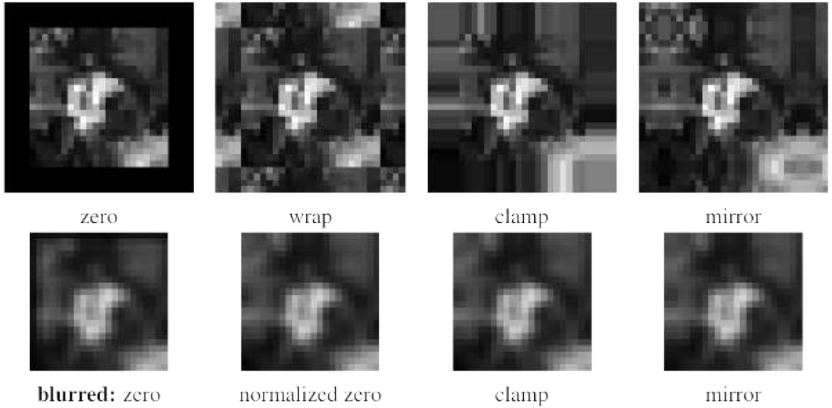




→ ignore the borders, but then the result has a different size than the original image;

→ pad with 0 (or other constant), wrap (loop around), clamp (replicate pixels on the borders), mirror (reflect pixels across edges). Results in some artefacts.

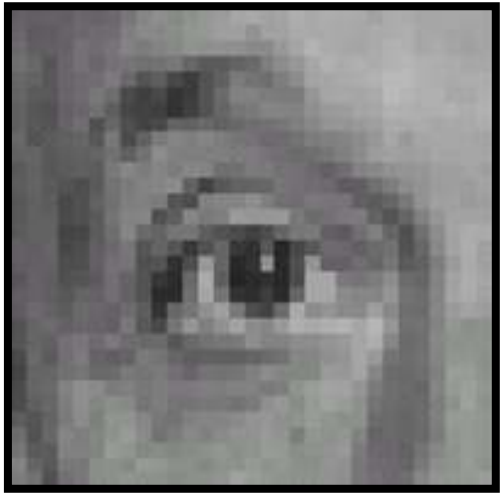
Blurring examples :



Szelisky 2010 © Springer

École des Ponts ParisTech

Practice with Linear Filters



Image

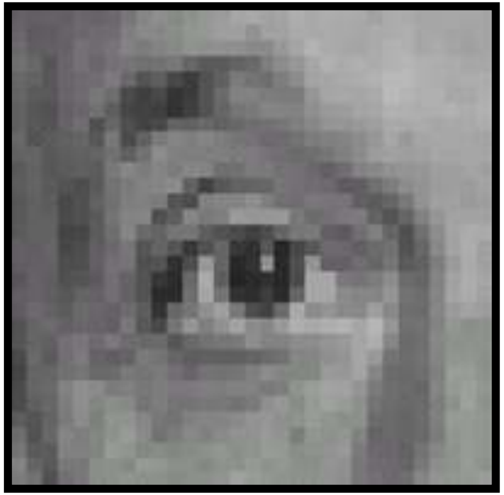
0	0	0
0	1	0
0	0	0

Filter

Result?

Source: D. Lowe

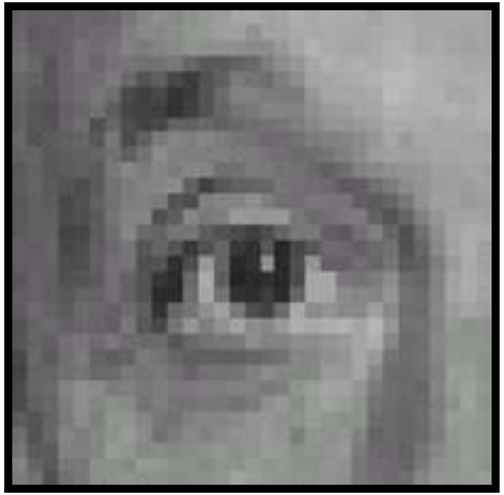
Practice with Linear Filters



Image

0	0	0
0	1	0
0	0	0

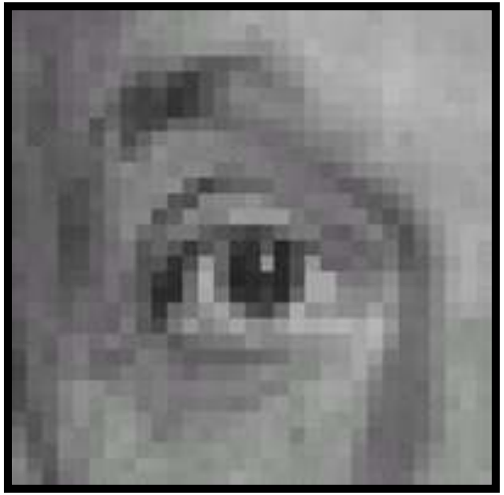
Filter



Result (no change)

Source: D. Lowe

Practice with Linear Filters



Image

0	0	0
1	0	0
0	0	0

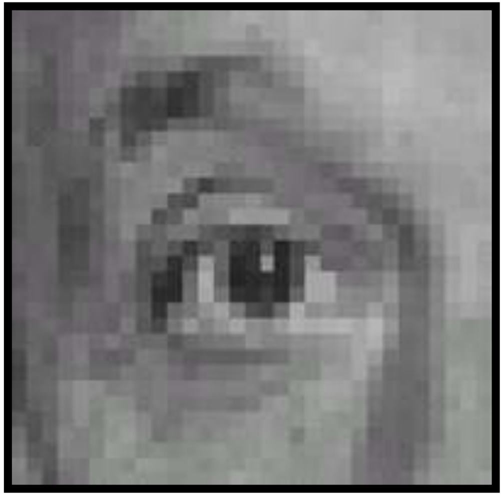
Filter

Result?

École des Ponts
ParisTech

Source: D. Lowe

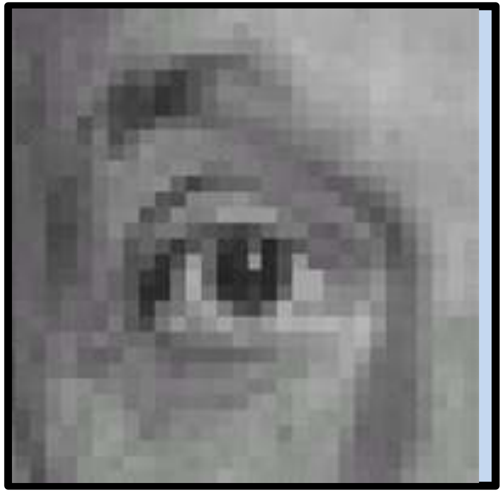
Practice with Linear Filters



Image

0	0	0
1	0	0
0	0	0

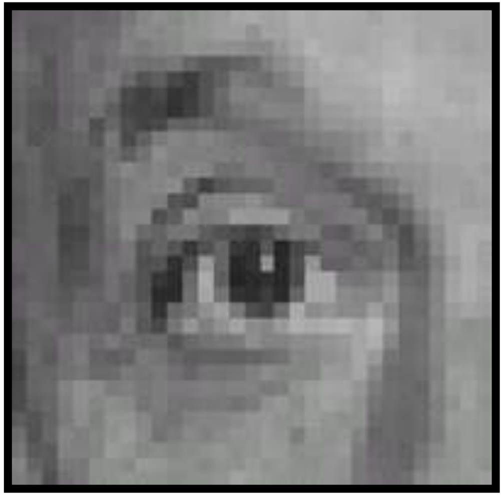
Filter



Result: shifted *left* by one pixel

Source: D. Lowe

Practice with Linear Filters



Image

$$\frac{1}{9}$$

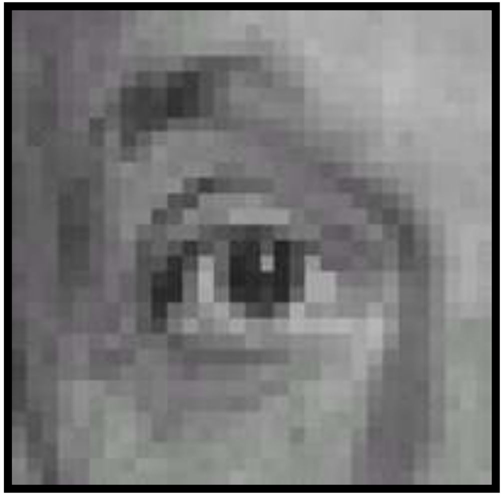
1	1	1
1	1	1
1	1	1

Filter

Result?

Source: D. Lowe

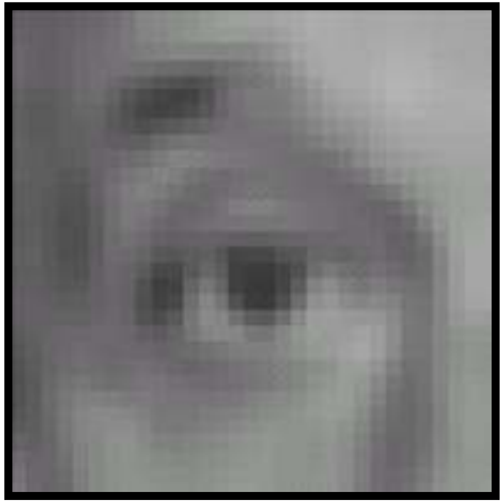
Practice with Linear Filters



Image

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

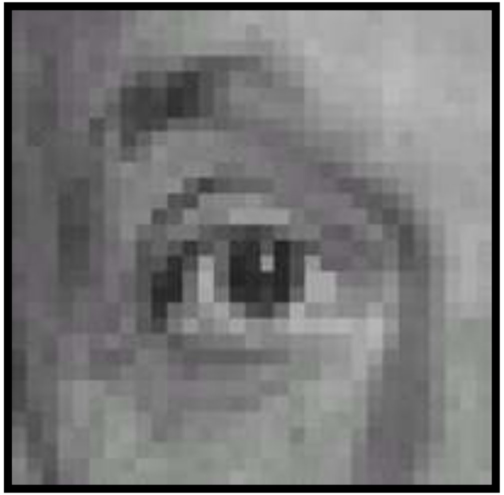
Filter



Result: blurred

Source: D. Lowe

Practice with Linear Filters



Image

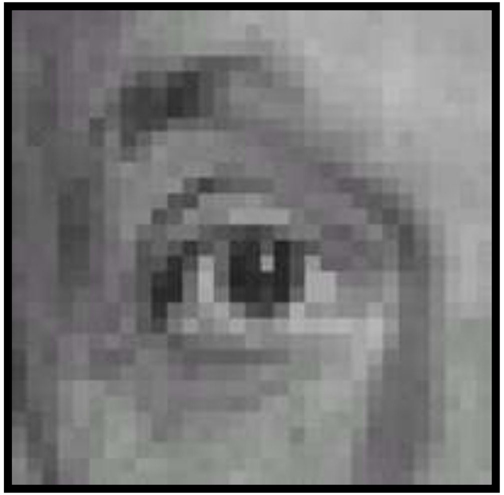
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filter

Result?

Source: D. Lowe

Practice with Linear Filters



Image

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

Filter

Result: Sharpened image

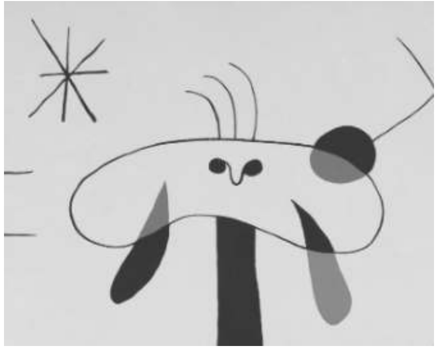
École des Ponts
ParisTech

Source: D. Lowe

Sharpening

! gray here corresponds to 0, black to a negative value, white to a postive value !

What does blurring take away?



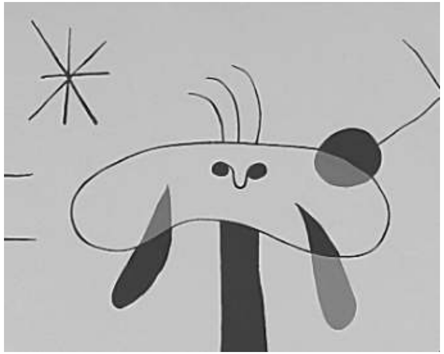
-

=

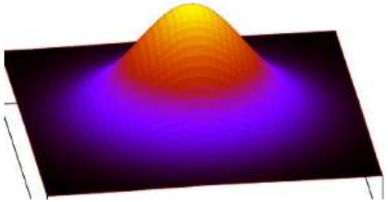
Adding it to the original image:

+

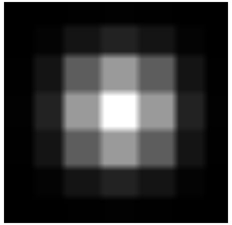
=

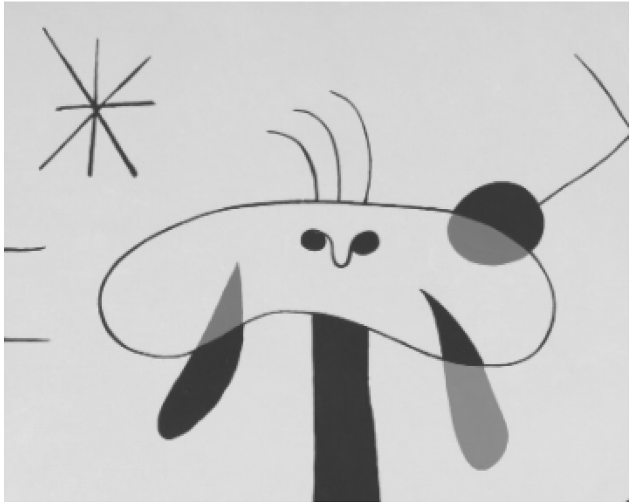


Gaussian Convolution



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





Gaussian filter with σ = 1.0 (normalized: black = 0, white = max value)

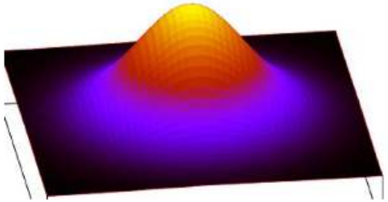
Common approximation:

1/16

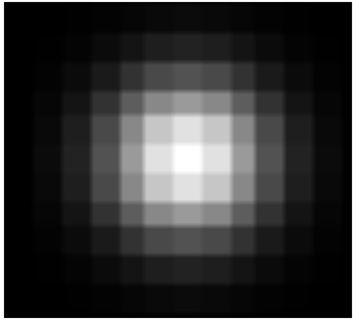
1	2	1
2	4	2
1	2	1

École des Ponts ParisTech

Gaussian Convolution

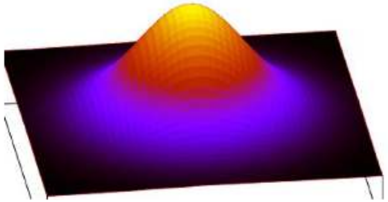


$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

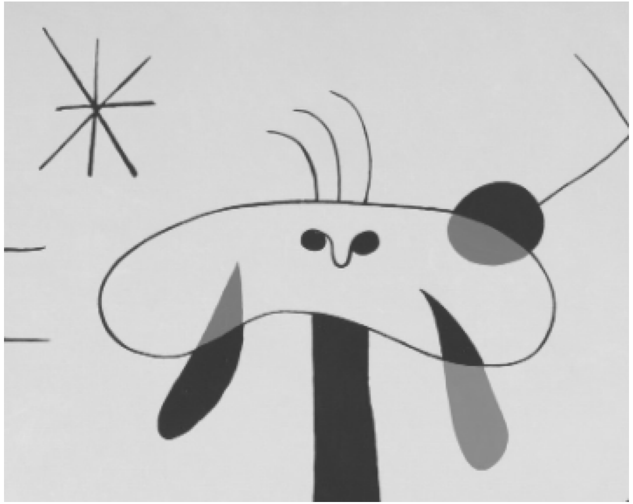


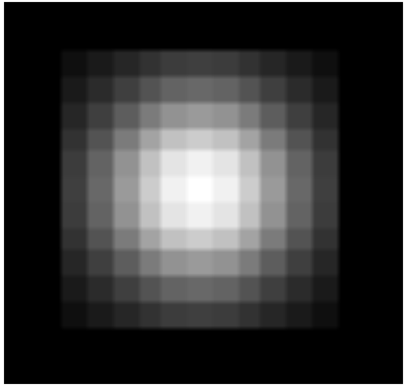
Gaussian filter with σ = 2
(normalized: black = 0,
white = max value)

Gaussian Convolution



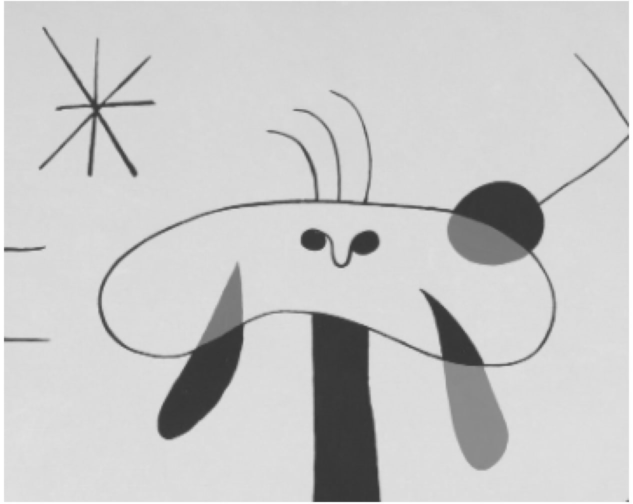
$$g_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





Gaussian filter with σ = 3 (normalized: black = 0, white = max value)

Practice with Linear Filters



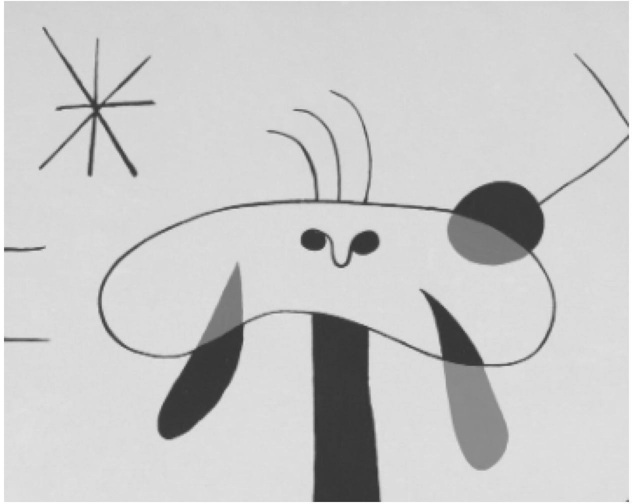
Image

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

Filter

Result

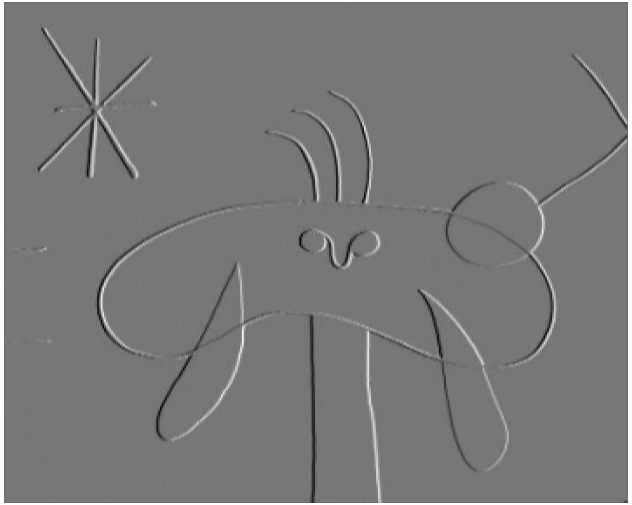
Practice with Linear Filters



Image

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

Filter

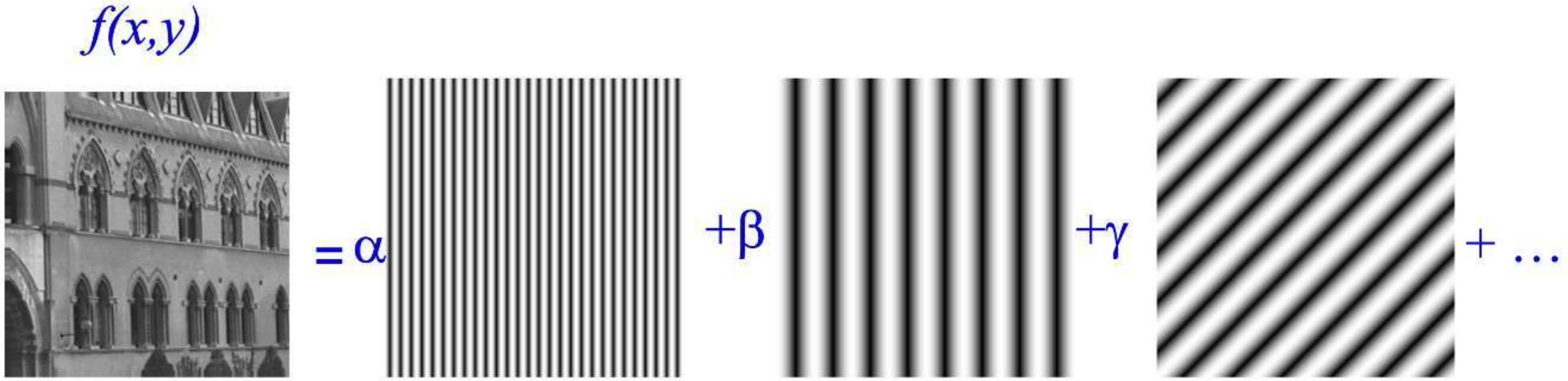


Result
(gray corresponds to 0)

Fourier Transform and Convolutions

Fourier Transform: High Level Description (1)

Decomposes image f into a weighted sum of 2D orthogonal basis functions:



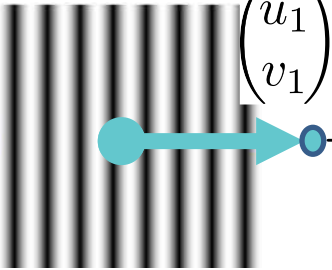
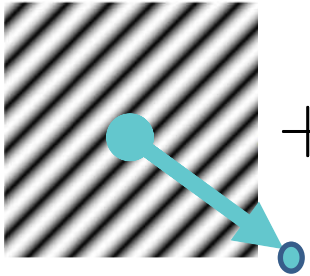
Fourier Transform: High Level Description (2)

$$f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}\,dudv$$

f a 2D image, F its Fourier transform;

$F(u, v)$ is complex in general;

f

$= |F(u_1, v_1)|$  $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}$ $+ |F(u_2, v_2)|$  $\begin{pmatrix} u_2 \\ v_2 \end{pmatrix}$ $+ \ldots$

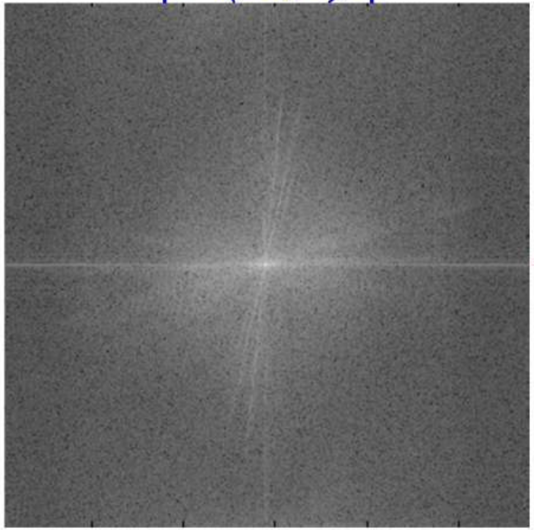
frequence: $\sqrt{u_1^2 + v_1^2}$

phase: $\mathrm{atan2}(F_{\mathrm{im}}(u_1, v_1), F_{\mathrm{re}}(u_1, v_1))$

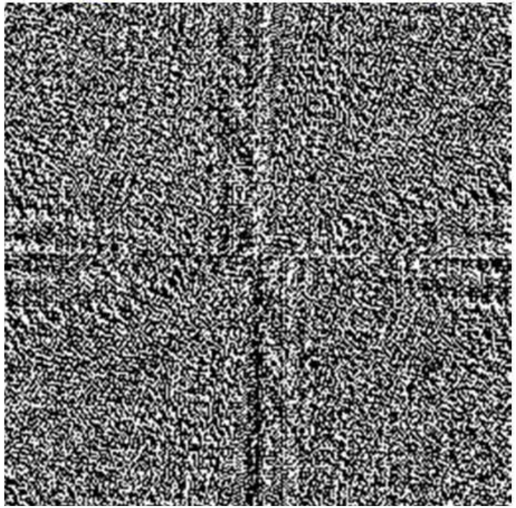
Fourier Transform: Example

f

$|F(u,v)|$

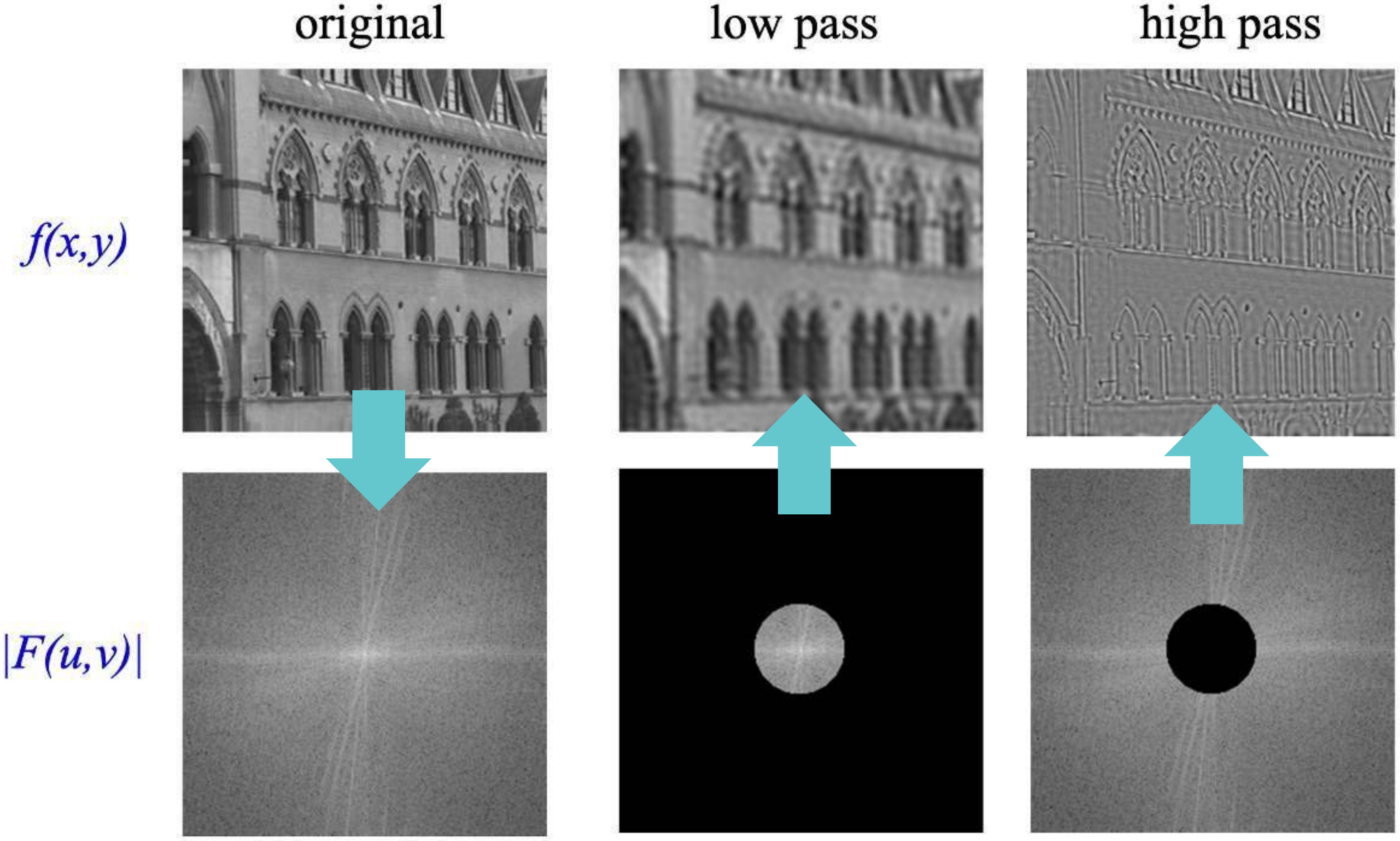


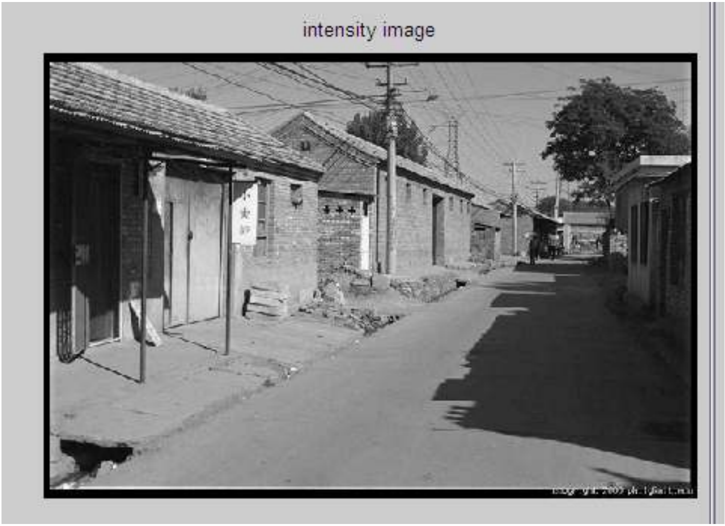
phase $F(u,v)$



- $|F(u,v)|$ generally decreases with higher spatial frequencies;
- phase appears less informative.

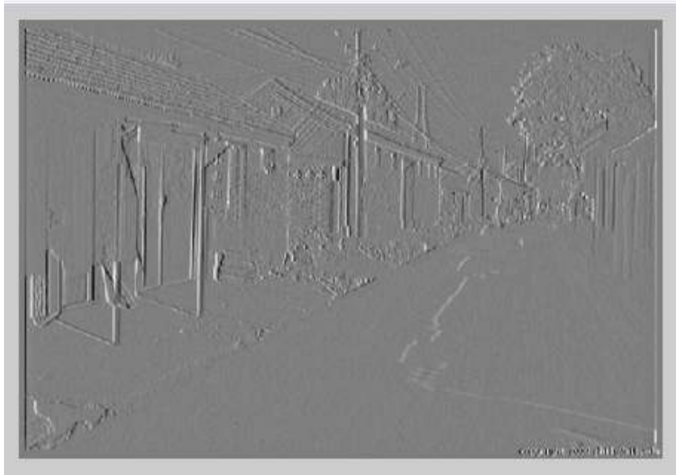
High and Low Frequencies





$*$

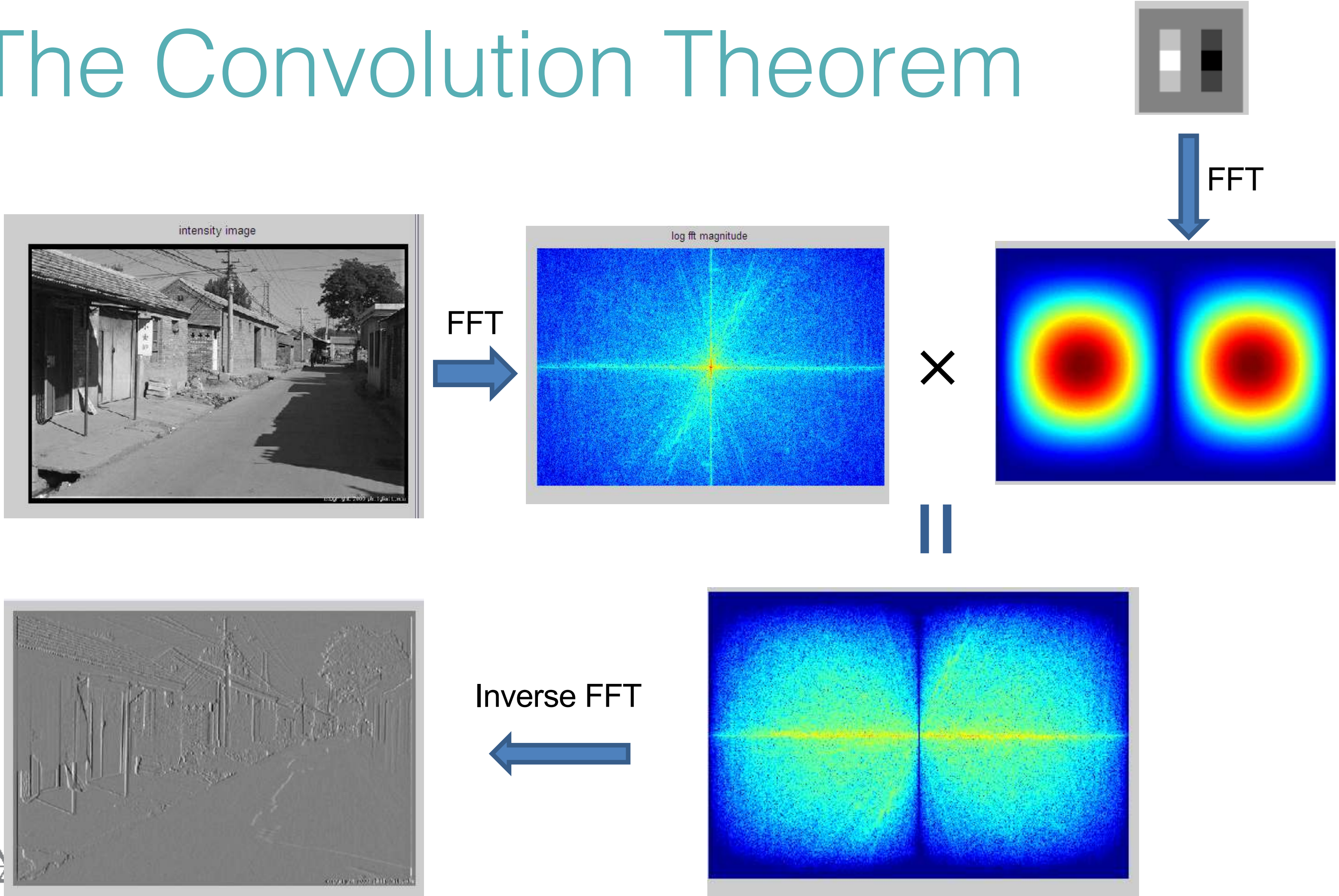
$=$



The product of convolution can be replaced by a regular product in the Fourier space.

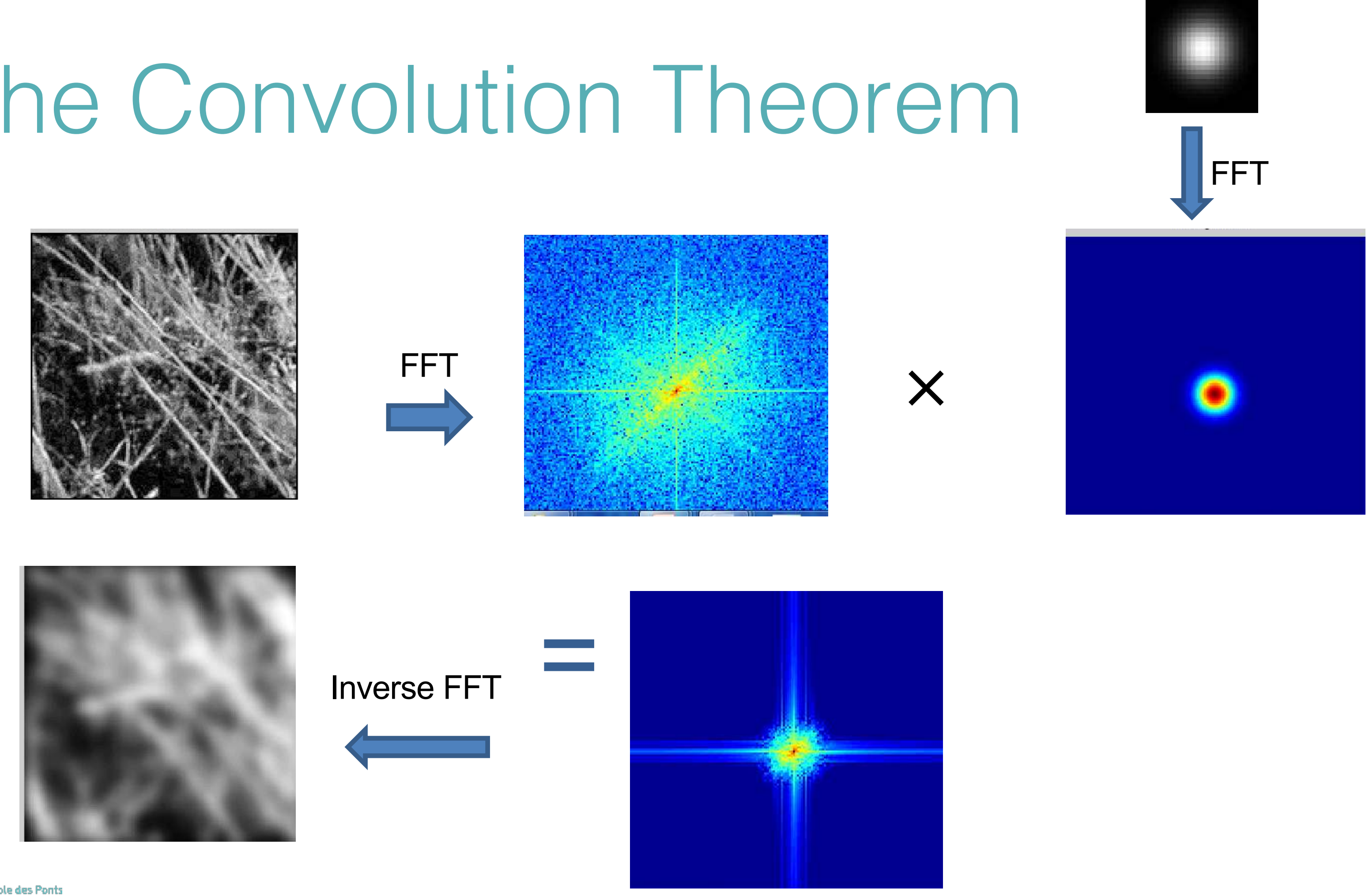
Slide: Hoiem

The Convolution Theorem



Slide: Hoiem

The Convolution Theorem



Slide: Hoiem

Is Convolution Invertible?

Slide from J. Hayes

Is Convolution Invertible?

- If convolution is just multiplication in the Fourier domain, isn't deconvolution just division?
- Sometimes, it clearly is invertible (e.g. a convolution with an identity filter)
- In one case, it clearly isn't invertible (e.g. convolution with an all zero filter)
- What about for common filters like a Gaussian?

Slide from J. Hayes

Deblurring and Denoising

Deblurring (Wiener filter)

O: Observed (blurred) image, I: Ideal image, K: blurring filter/kernel:

$$O = I * K$$

then deblurring is possible if the Fourier Transform $\widehat{K}$ of K is never 0 (this is the case when K is a Gaussian kernel (why?):

$$\hat{I} = \hat{O}/\hat{K}$$

because the coefficients of the Fourier Transform of the inverse of a function are the inverse of the coefficients of the Fourier Transform of the function.

École des Ponts ParisTech

Deblurring (Wiener filter)

In practice, there is noise in the observed image:

$$O = I * K + N$$

Then:

$$\hat{I} = \hat{O}/\hat{K} - \hat{N}/\hat{K}$$

But noise N is unknown. We could try to ignore it, as it is smaller than the image values. However, the Fourier Transform $\widehat{K}$ has also small values (why?), and ignoring the term $\widehat{N}/\widehat{K}$ would result in large errors.

Denoising

Blurring an image removes the noise: why?

Denoising

$$O = I + N$$

with N: noise iid, mean 0 and standard deviation σ^2

With K a constant window kernel:

$$K(i,j) = \frac{1}{(2S-1)^2}\mathbb{1}_{\max(i,j)<S}$$

Blurred image: $K * O = K * I + K * N$

Denoising

$$K * O = K * I + K * N$$

with N: noise iid, mean 0 and standard deviation σ^2

and: $K(i,j) = \frac{1}{(2S-1)^2} \mathbb{1}_{\max(i,j)<S}$

$$\mathbb{E}[K * N] = 0$$

$$\begin{aligned}
\mathbb{E}[K * N(i,j)^2] &= \mathbb{E}\left[\left(\sum_{k,l\in\mathbb{Z}} N(i-k,j-l)K(k,l)\right)^2\right] \\
&= \frac{1}{(2S-1)^4}\mathbb{E}\left[\left(\sum_{k,l=-S+1}^{S-1} N(i-k,j-l)\right)^2\right] \\
&= \frac{1}{(2S-1)^2}\sigma^2
\end{aligned}$$

École des Ponts ParisTech

Denoising

- If S is large enough: $K * O \sim K * I$
- No more noise, but blurred image

➢ How to preserve edges/texture?

Denoising with Convolution: Limits

Blurring salt-and-pepper noise:

3x3

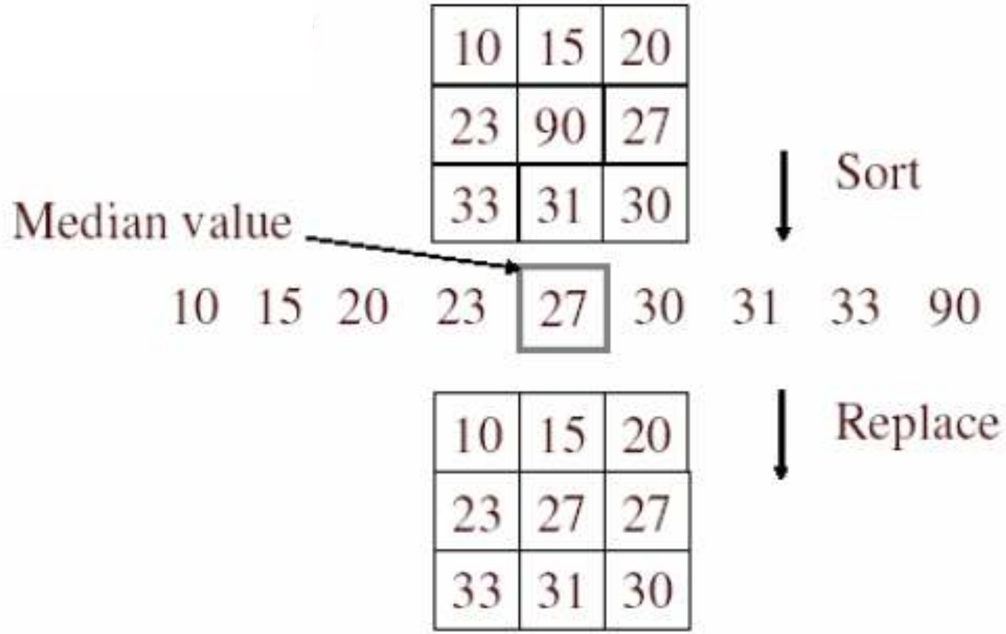
5x5

7x7

Source: L. Lazebnik

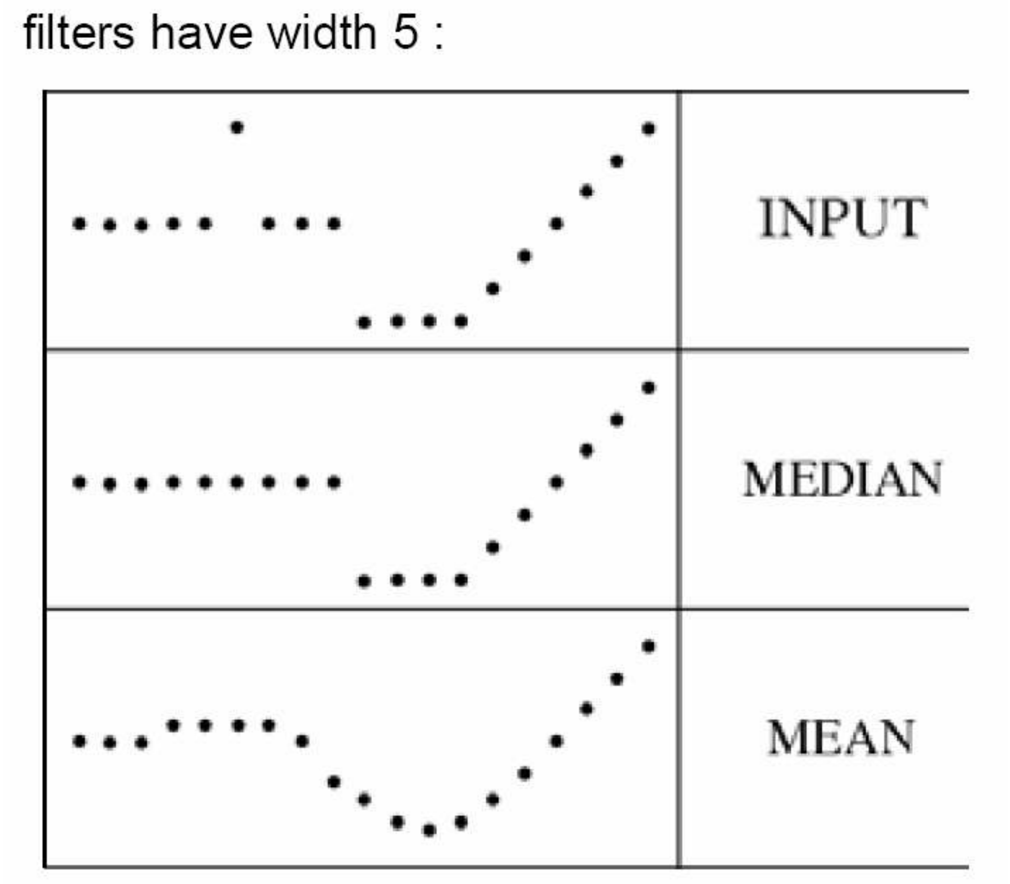
Alternative: Median filtering

A **median filter** operates over a window by selecting the median intensity in the window:



Source: K. Grauman

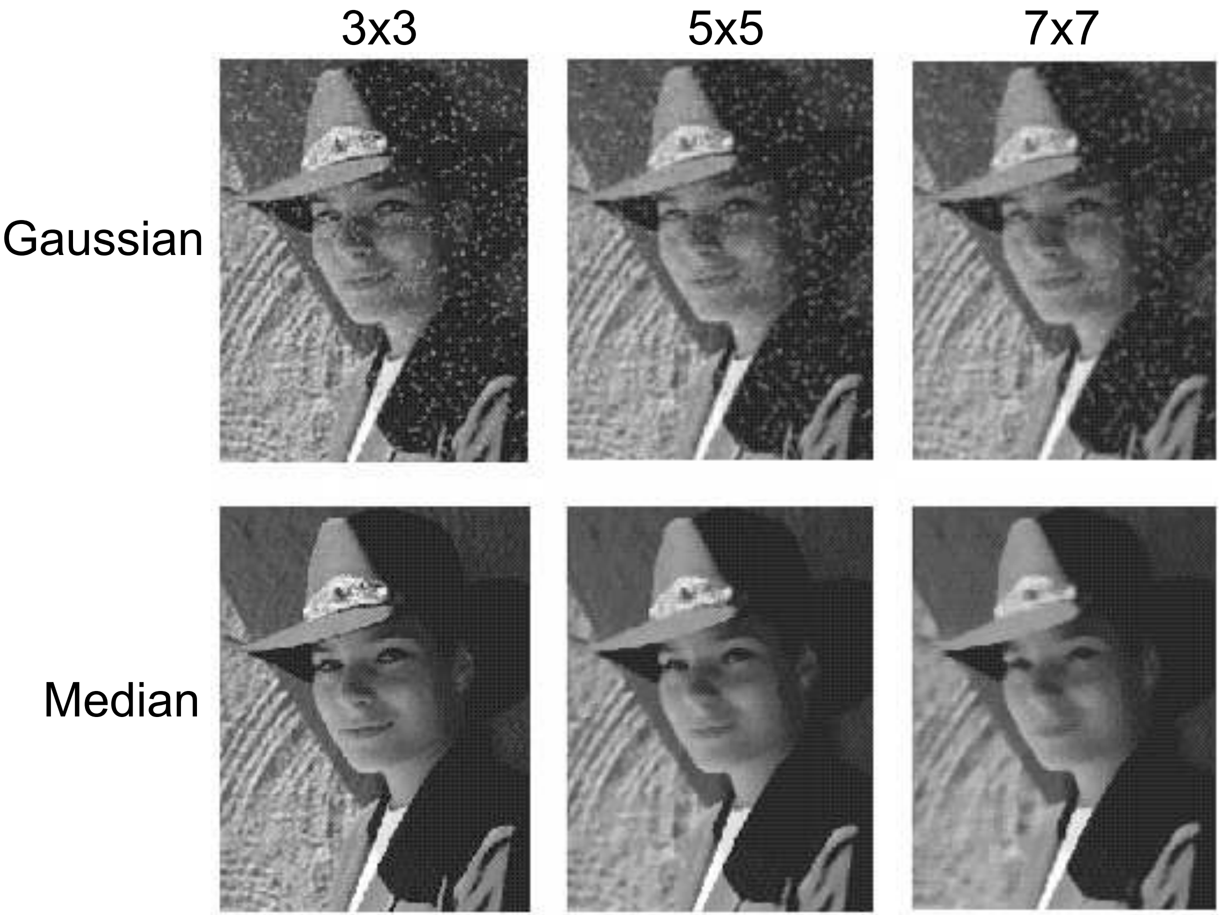
Median Filter



École des Ponts ParisTech

Source: K. Grauman

Gaussian vs. Median filtering



Source: L. Lazebnik

Median Filtering: Limitations

- Remove fine details;
- Slow with large windows.

École des Ponts
ParisTech

Bilateral Filter

With slides from Sylvain Paris and Frédo Durand

Gaussian Smoothing

Smooth the image, but remove edges:

Gaussian Smoothing

Smooth the image, but remove edges:

Bilateral Filter

Smooth image while preserving the main edges

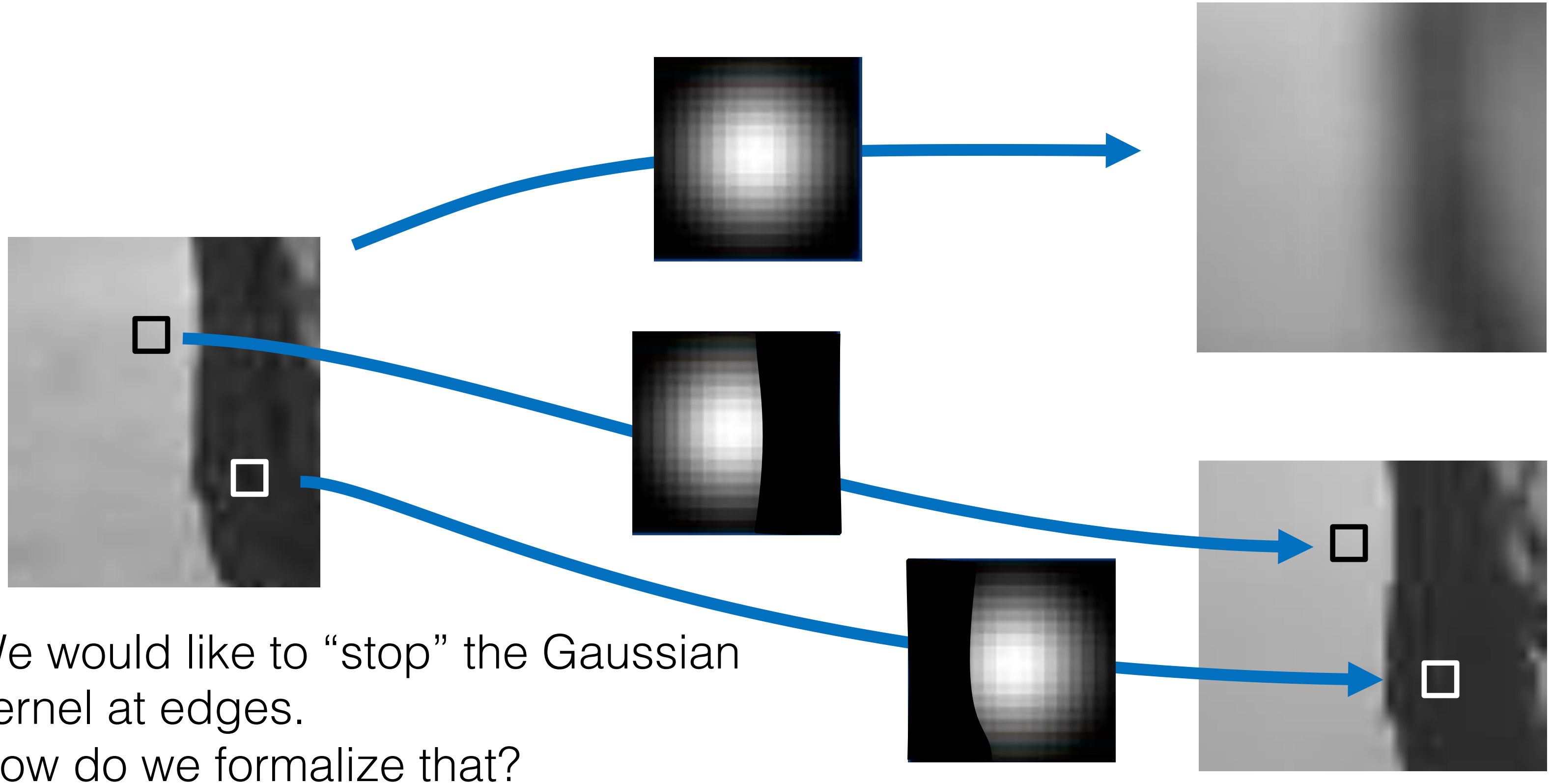
Tomasi 98

Bilateral Filter

Smooth image while preserving the main edges

Tomasi 98

Gaussian Smoothing → Bilateral Filter



We would like to "stop" the Gaussian kernel at edges.
How do we formalize that?

Equations

- Gaussian smoothing:

$$G[I]_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times I_{\mathbf{q}}$$

- Bilateral Filter:

$$BF[I]_{\mathbf{p}} = \qquad \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \qquad\qquad \times I_{\mathbf{q}}$$

École des Ponts
ParisTech

Equations

- Gaussian smoothing:

$$G[I]_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times I_{\mathbf{q}}$$

- Bilateral Filter:

$$BF[I]_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times G_{\sigma_R}(I_{\mathbf{p}} - I_{\mathbf{q}}) \times I_{\mathbf{q}}$$

Equations

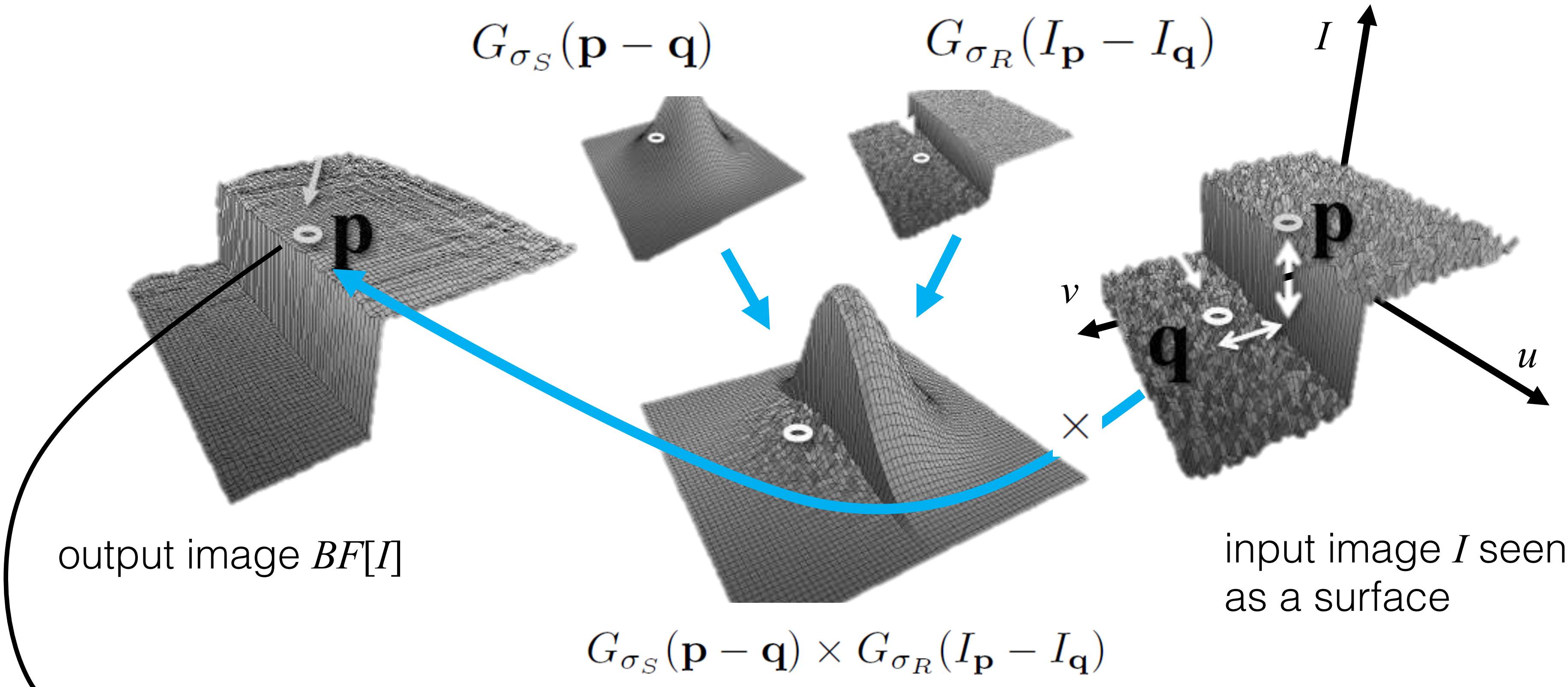
- Gaussian smoothing:

$$G[I]_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times I_{\mathbf{q}}$$

- Bilateral Filter:

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times G_{\sigma_R}(I_{\mathbf{p}} - I_{\mathbf{q}}) \times I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times G_{\sigma_R}(I_{\mathbf{p}} - I_{\mathbf{q}})$$



$$BF[I]_\mathbf{p} = \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} G_{\sigma_S}(\mathbf{p}-\mathbf{q}) \times G_{\sigma_R}(I_\mathbf{p}-I_\mathbf{q}) \times I_\mathbf{q}$$

Bilateral Filter for Denoising

École des Ponts
ParisTech

Computation

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}} G_{\sigma_S}(\mathbf{p} - \mathbf{q}) \times G_{\sigma_R}(I_{\mathbf{p}} - I_{\mathbf{q}}) \times I_{\mathbf{q}}$$

Depends from the image value at each pixel:
cannot easily be pre-computed, no Fourier Transform
→ slow to compute *a priori*, but see 2 next slides

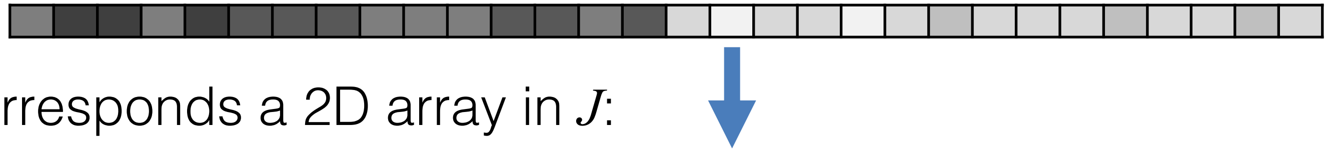
Fast Computation

Idea: Can be seen as filtering with an additional dimension

$$I(\mathbf{p}) \to J(\mathbf{p}, l) \text{ with } J(\mathbf{p}, l) = \begin{cases} l & \text{if } I(\mathbf{p}) = l \\ 0 & \text{otherwise} \end{cases}$$

if I is seen as a 2D array, J is a 3D array.

To a row of image I:



corresponds a 2D array in J:

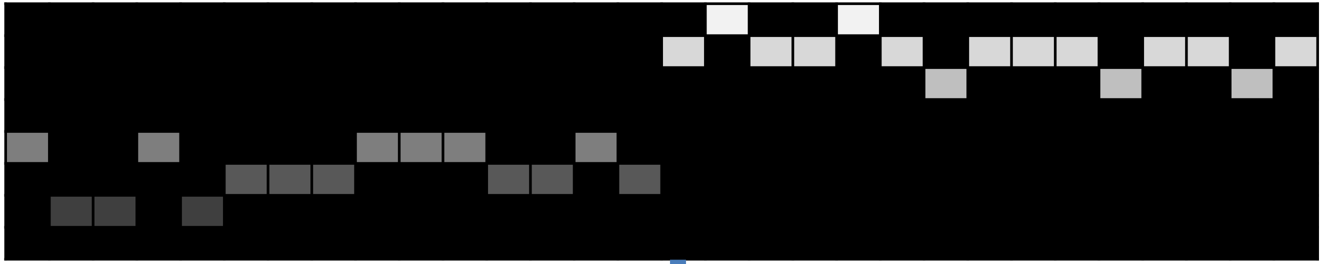
$$I(\mathbf{p}) \rightarrow J(\mathbf{p}, l) \text{ with } J(\mathbf{p}, l) = \begin{cases} l & \text{if } I(\mathbf{p}) = l \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}
BF[I](\mathbf{p}_u, \mathbf{p}_v) &= \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} G_{\sigma_S}(\mathbf{p}-\mathbf{q}) \times G_{\sigma_R}(I_\mathbf{p} - I_\mathbf{q}) \times I_\mathbf{q} \\
&= \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} k_S \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma_S^2}\right) k_R \exp\left(-\frac{(I_\mathbf{p}-I_\mathbf{q})^2}{2\sigma_R^2}\right) \times I_\mathbf{q} \\
&= \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} k_S k_R \exp\left(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2\sigma_S^2} - \frac{(I_\mathbf{p}-I_\mathbf{q})^2}{2\sigma_R^2}\right) \times I_\mathbf{q} \\
&= \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} k_S k_R \exp\left(-\frac{(\mathbf{p}_u-\mathbf{q}_u)^2}{2\sigma_S^2} - \frac{(\mathbf{p}_v-\mathbf{q}_v)^2}{2\sigma_S^2} - \frac{(I_\mathbf{p}-I_\mathbf{q})^2}{2\sigma_R^2}\right) \times I_\mathbf{q} \\
&= \frac{1}{W_\mathbf{p}} \sum_\mathbf{q} k_S k_R \exp\left(-\frac{1}{2}\mathbf{d}^T \begin{bmatrix} \sigma_S^2 & 0 & 0 \\ 0 & \sigma_R^2 & 0 \\ 0 & 0 & \sigma_R^2 \end{bmatrix}^{-1} \mathbf{d}\right) \times I_\mathbf{q} \text{ with } \mathbf{d} = \begin{bmatrix} (\mathbf{p}_u - \mathbf{q}_u) \\ (\mathbf{p}_v - \mathbf{q}_v) \\ (I_\mathbf{p} - I_\mathbf{q}) \end{bmatrix} \\
&= \left(\frac{1}{W_\mathbf{p}}(G_{\sigma_S,\sigma_S,\sigma_R} * J)\right)(\mathbf{p}_u, \mathbf{p}_v, I_\mathbf{p})
\end{aligned}$$

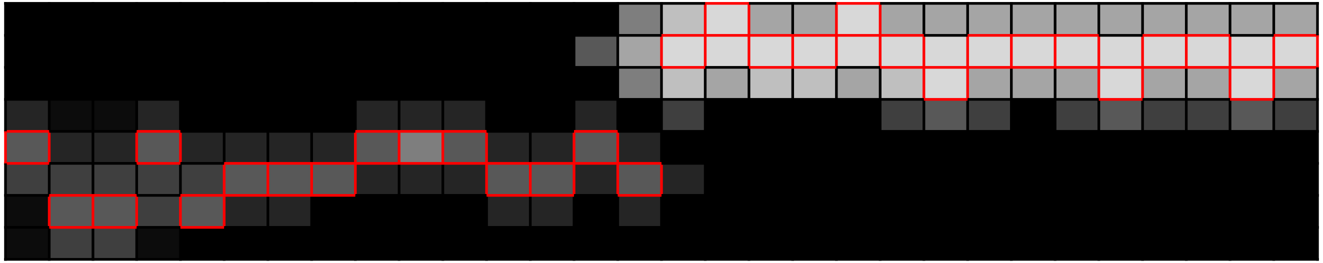
École des Ponts ParisTech

Gaussian filtering can be computed very efficiently (see Lecture #2).

Array J:



Gaussian convolution with kernel (σ_S, σ_R)



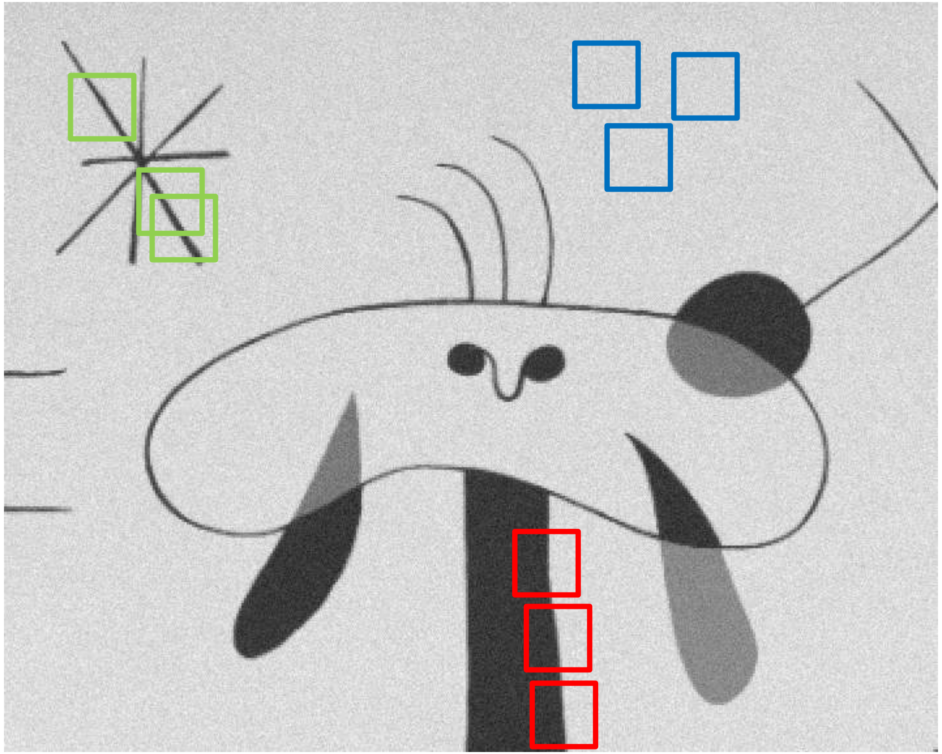
Keep only the pixels in red to build the bilateral filter output:

École des Ponts ParisTech

Non-Local Means for Denoising

Non Local Means

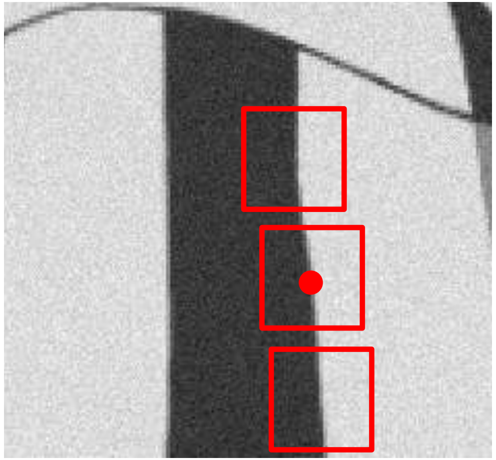
Observation: Images are self-repetitive



École des Ponts
ParisTech

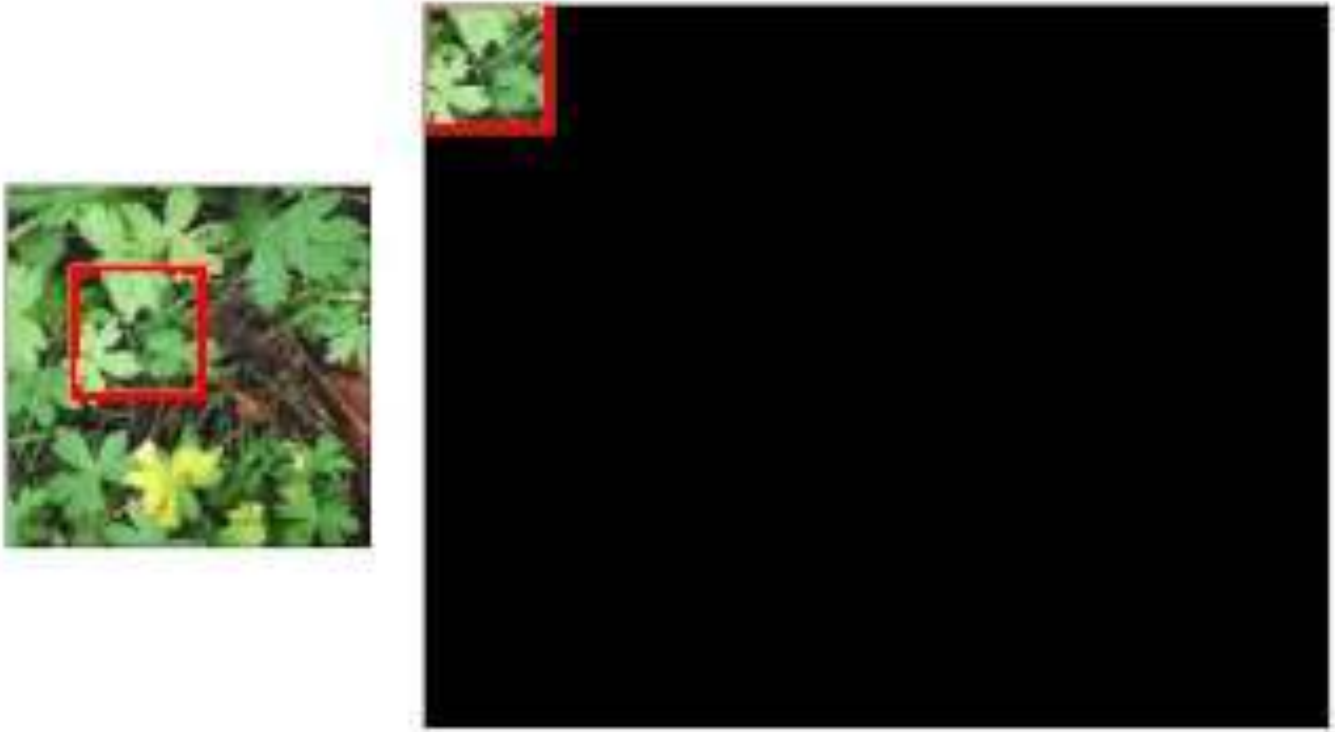
Non Local Means

For each pixel: use the mean of intensity values of the neighbors that have similar appearance (can be seen as an extension of the BF)



Basis for state of the art denoising

Patch-Based Texture Synthesis & Image Completion



Wikipedia