Computer Vision I - *Introduction & Filtering*

Carsten Rother

28/10/2015

Admin Stuff

- Language: German/English; Slides: English
  (all the terminology and books are in English)
- Lecturer: Carsten Rother (Eric Brachmann, Anita Sellent)
- Exercises: Dmitri Schlesinger, Eric Brachmann
- Staff Email: dmytro.shlezinger@tu-dresden.de
- Announcements: on our webpage

- Course Books:
  - Image Processing/Geometry:
    Computer Vision: Algorithms and Applications
    by Rick Szeliski; Springer 2011. An earlier version of the book is online:
    http://szeliski.org/Book/
  - Geometry:
    Multiple View Geometry; Hartley and Zisserman;
    Cambridge Press 2004. Second edition. Parts of book are online:
    http://www.robots.ox.ac.uk/~vgg/hzbook/
  - Also pointers to conference and journal articles

Course Overview (total 14 lectures)

Lecture 1 (23.10): Introduction & Filtering
Ex1: Intro to OpenCV

Lecture 2 (30.10): Filtering and Feature detection
Ex2: Intro to exercise: Filtering

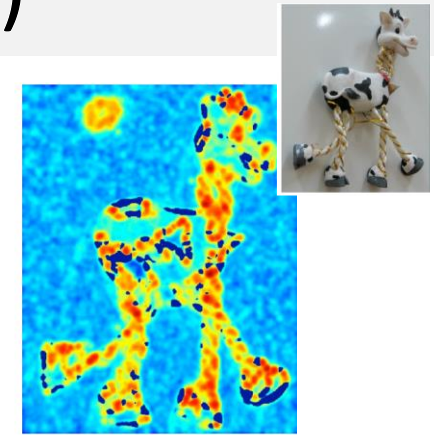
Lecture 3 (6.11): Image Matching and Projective Geometry
Ex3: homework

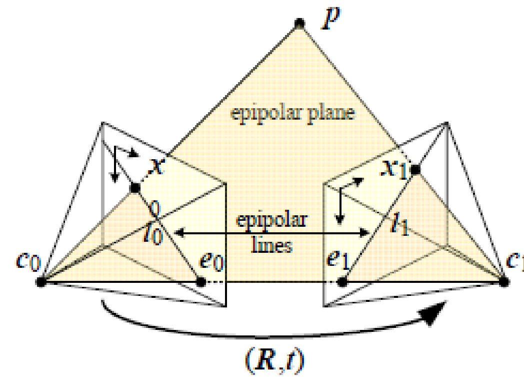
Lecture 4 (13.11): Geometry of One and Two Images
Ex4: homework

Lecture 5 (20.11): Robust Geometry Estimation
Ex5: Intro to exercise: Panoramic Stitching (Geometry)

Lecture 6 (27.11): Multi-View 3D Reconstruction
Ex6: homework

Lecture 7 (4.12): Object Pose estimation
Ex7: homework





Course Overview (total 14 lectures)

Lecture 8 (11.12): Tracking – Part 1
Ex8: Intro to exercise: Object Pose Estimation

Lecture 9 (18.12): Tracking – Part 2
Ex9: homework

Lecture 10 (8.1): Dense Matching – Optical Flow
Ex10: homework

Lecture 11 (15.1): Dense Matching – Stereo Part 1
Ex11: homework

Lecture 12 (22.1): Dense Matching – Stereo Part 2
Ex12: homework

Lecture 13 (29.1): Dense Matching – Scene Flow
Ex13: Prepare Poster Session

Lecture 14 (5.2): Poster Session
Ex14: homework

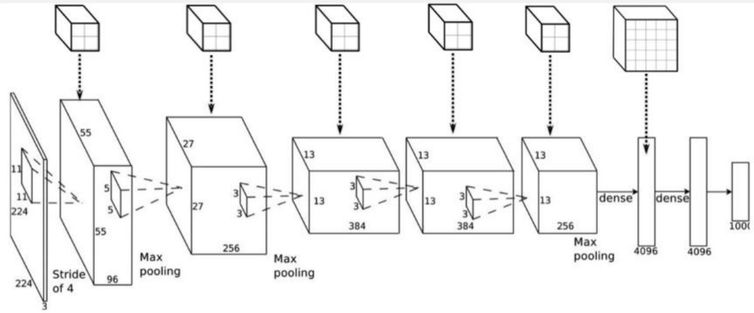
Exams and Exercises

- Exam: in Person

- Exercises/homework:
  - There are three blocks
  - In each block you have to do a certain amount of exercises
  - The exercises should to be handed in until end of semester (ideally after each block)

- Exam:
  - Exercises are not mandatory, i.e. you can sit the exam without having done the exercises
  - In the exam I may ask questions about the exact exercise you have done. If you have not done any exercise then this may result in a worse mark

CVLD Lectures



- WS 15/16
  - Computer Vision 1 (2+2)
  - Machine Learning 1 (2+2)
- SS 16
  - Computer Vision 2 (2+2)
  - Machine Learning 2 (2+2)
  - Image Processing (1+1)

- For doing a Master/PhD in the CVLD one should do the computer vision or machine learning track
- Computer graphics (Prof. Gumhold) (Introduction, I, II)
  3D Scanning with structured light; Illumination models; Geometry

Before we start … some Advertisement

Optimization & Learning

with a focus on graphical models, deep models, and large-scale optimization.

Image Matching

with a focus on 3-9DoF scene flow, as well as jointly recovering multiple physical aspe

Scene Understan

with a focus on

standing,

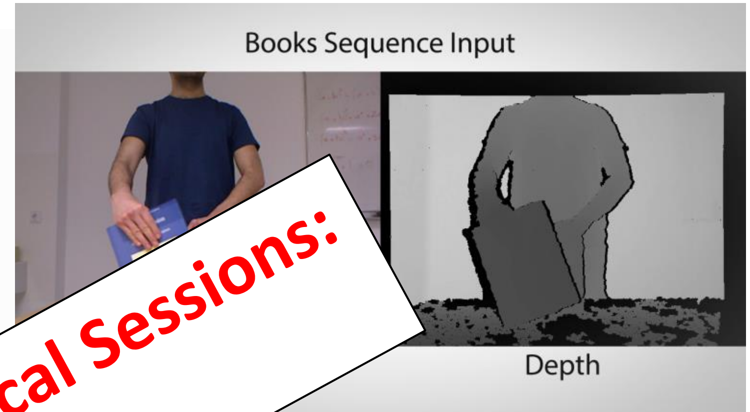
Bio Imagi

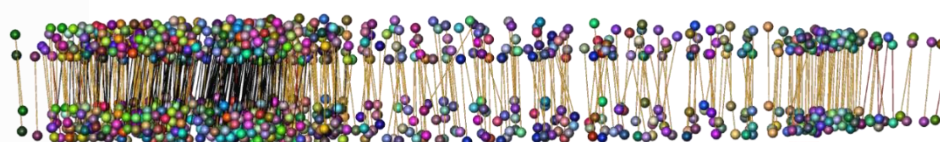
with a focus on segmentation, tracking, and matching – jointly with teams from MPI-CBG.

Image Analysis

with a focus on segmentation, interactive techniques, as well as camera design.

Thesis and Practical Sessions: Looking for People !





Future in Computer Vision

<u>A project work in the CVLD is a good stepping stone if you:</u>

- want to do a PhD in computer vision, graphics, machine learning
- Becoming a researcher or software developer in a research lab (Microsoft Research, Daimler, Google, Adobe, TechniColor, etc)
- If you are interested in doing a start-up
- Other “computer vision related” industry

Introduction to Computer Vision

What is computer Vision?

(Potential) Definition:
Developing computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in.

Introduction to Computer Vision

What is computer Vision?

(Potential) Definition:
Developing computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in.

What does it mean to “understand”?

(Potential) Definition:

Developing computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in.

<u>Physics-based vision:</u>

Geometry

Segmentation

Camera parameters

Emitted light (sun)

Surface properties: Reflectance, material

<u>Semantic-based vision:</u>

Objects: class, pose

Scene: outdoor,…

Attributes/Properties:

- old-fashioned train
- A-on-top-of-B

Image-formation model

Very many sources of variability

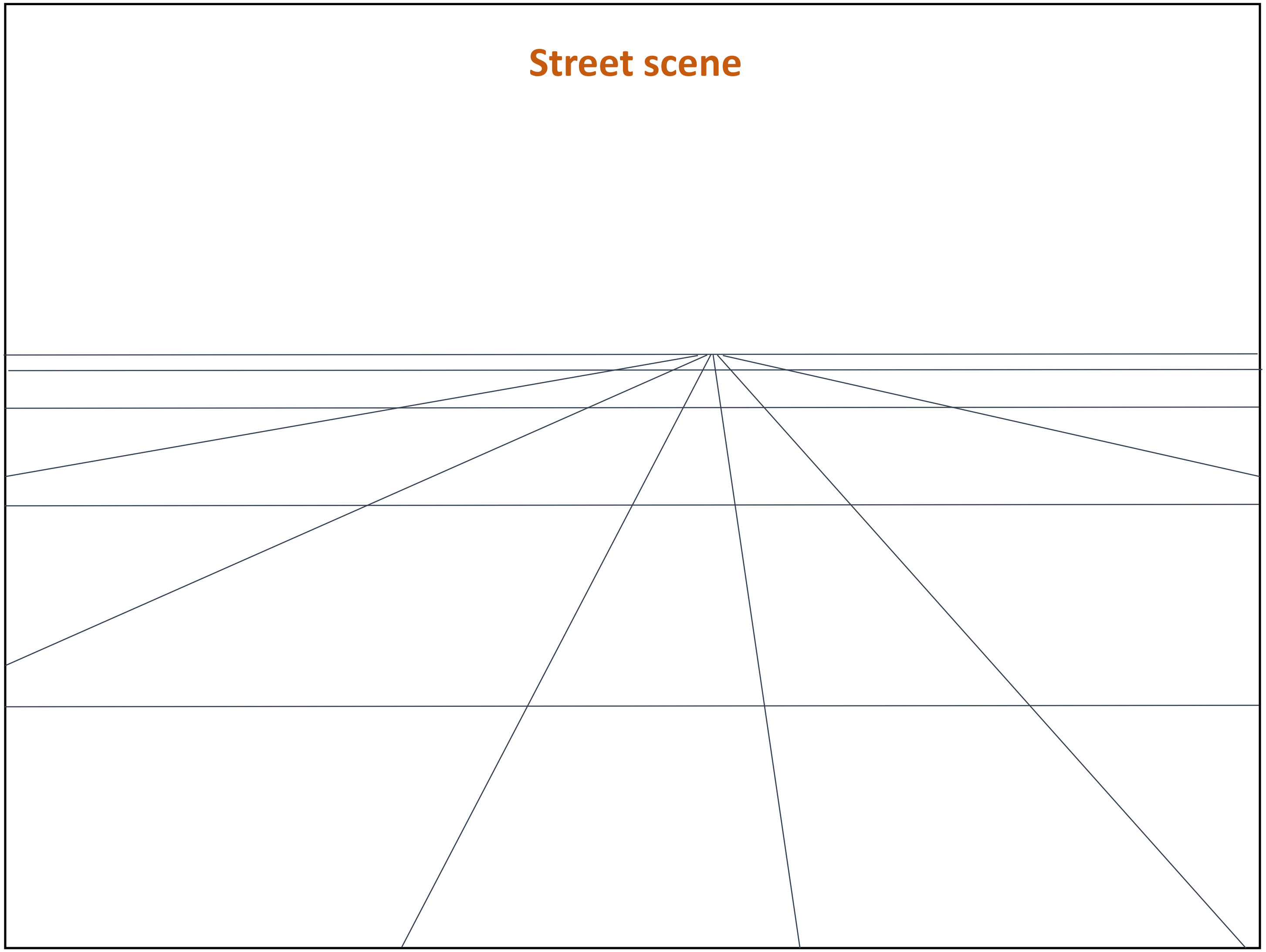
Image

[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type

Scene geometry



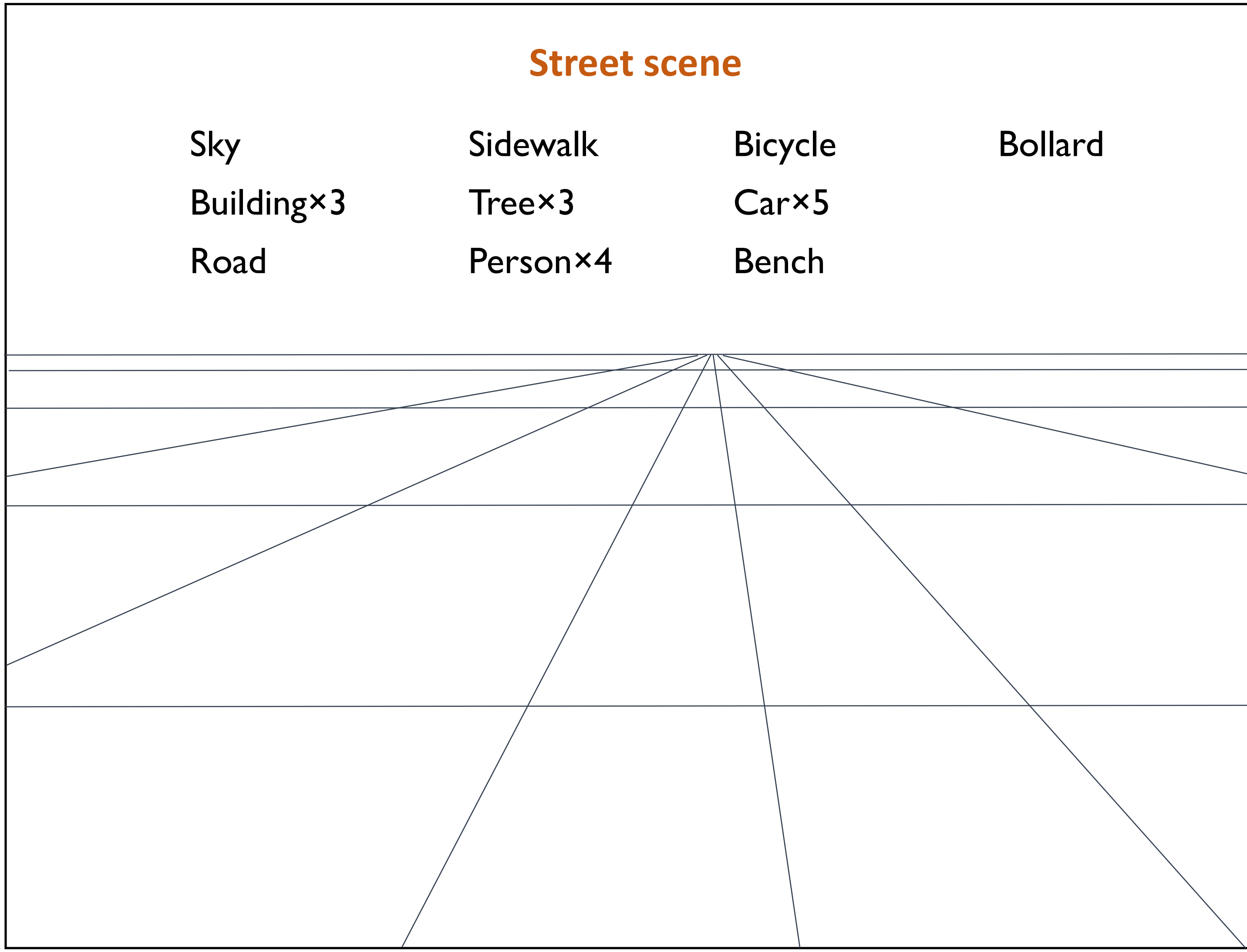
[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type

Scene geometry

Object classes



[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type

Scene geometry

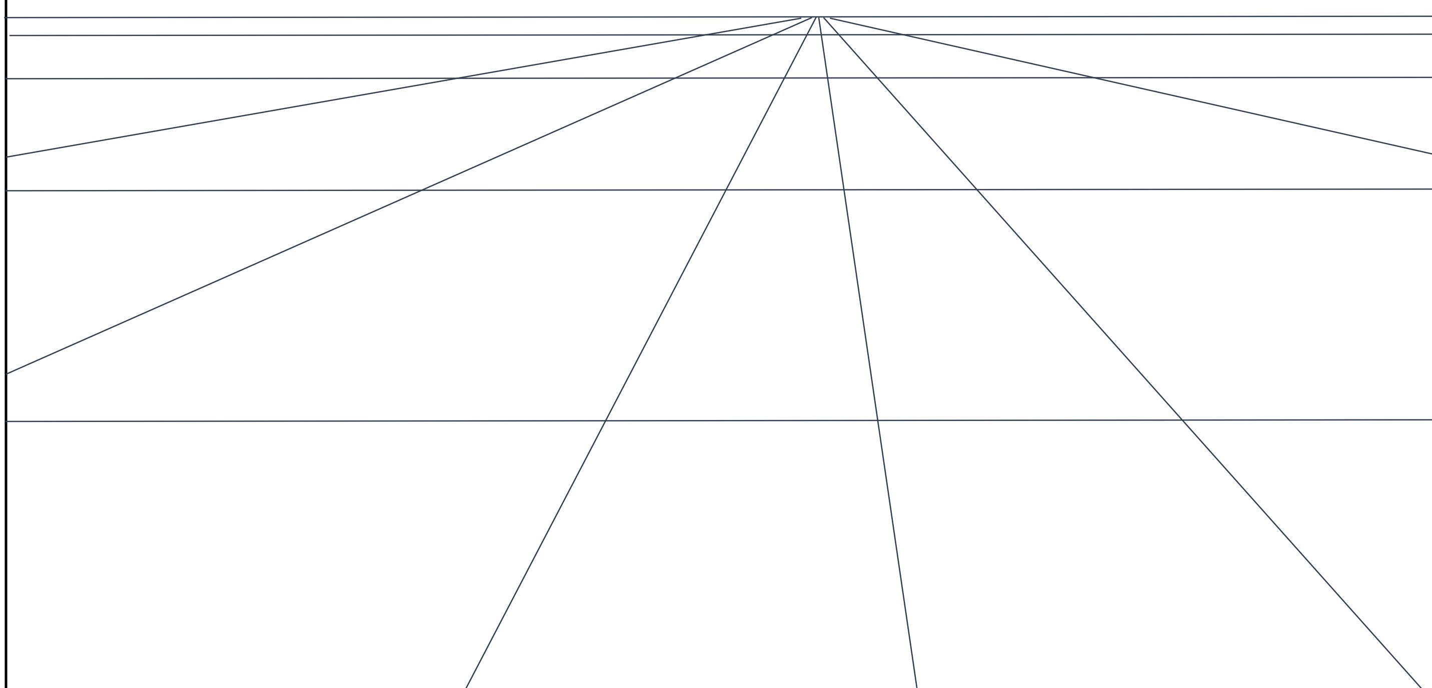
Object classes

Object position

Object orientation

Street scene

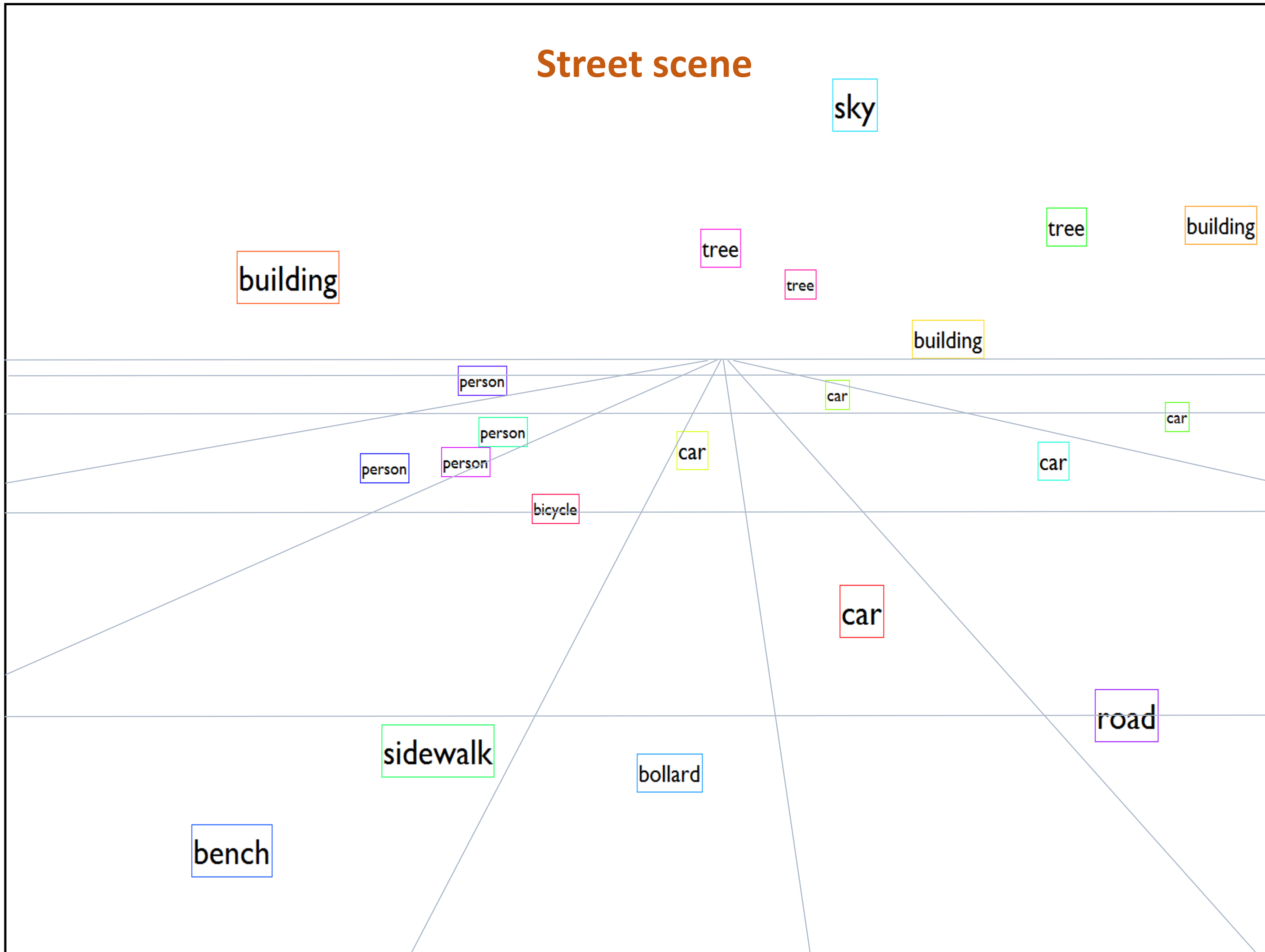
Sky	Sidewalk	Bicycle	Bollard
Building×3	Tree×3	Car×5	
Road	Person×4	Bench	



[Slide Credits: John Winn, ICML 2008]

Image-formation model

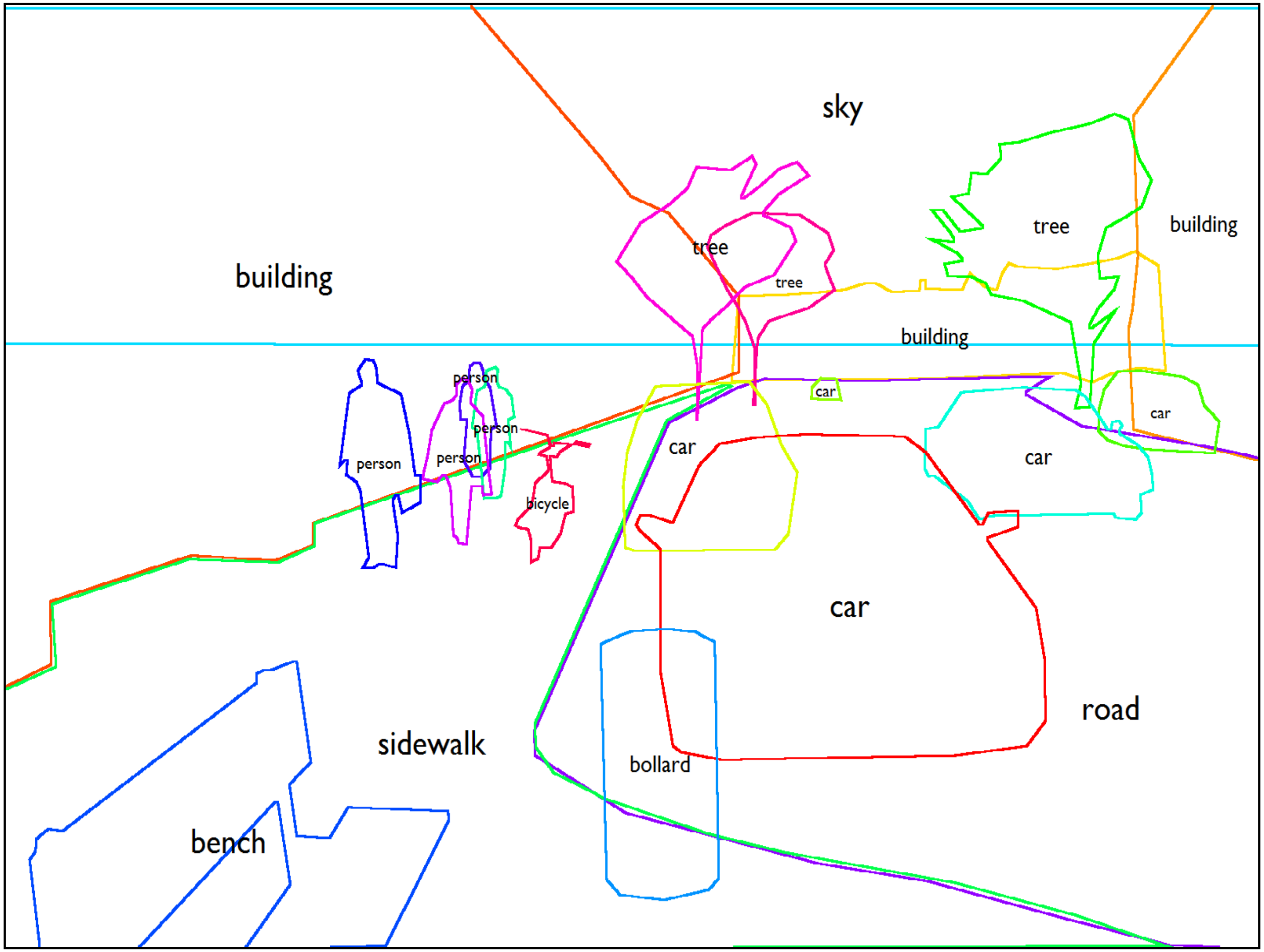
Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape



[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions



[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type

Scene geometry

Object classes

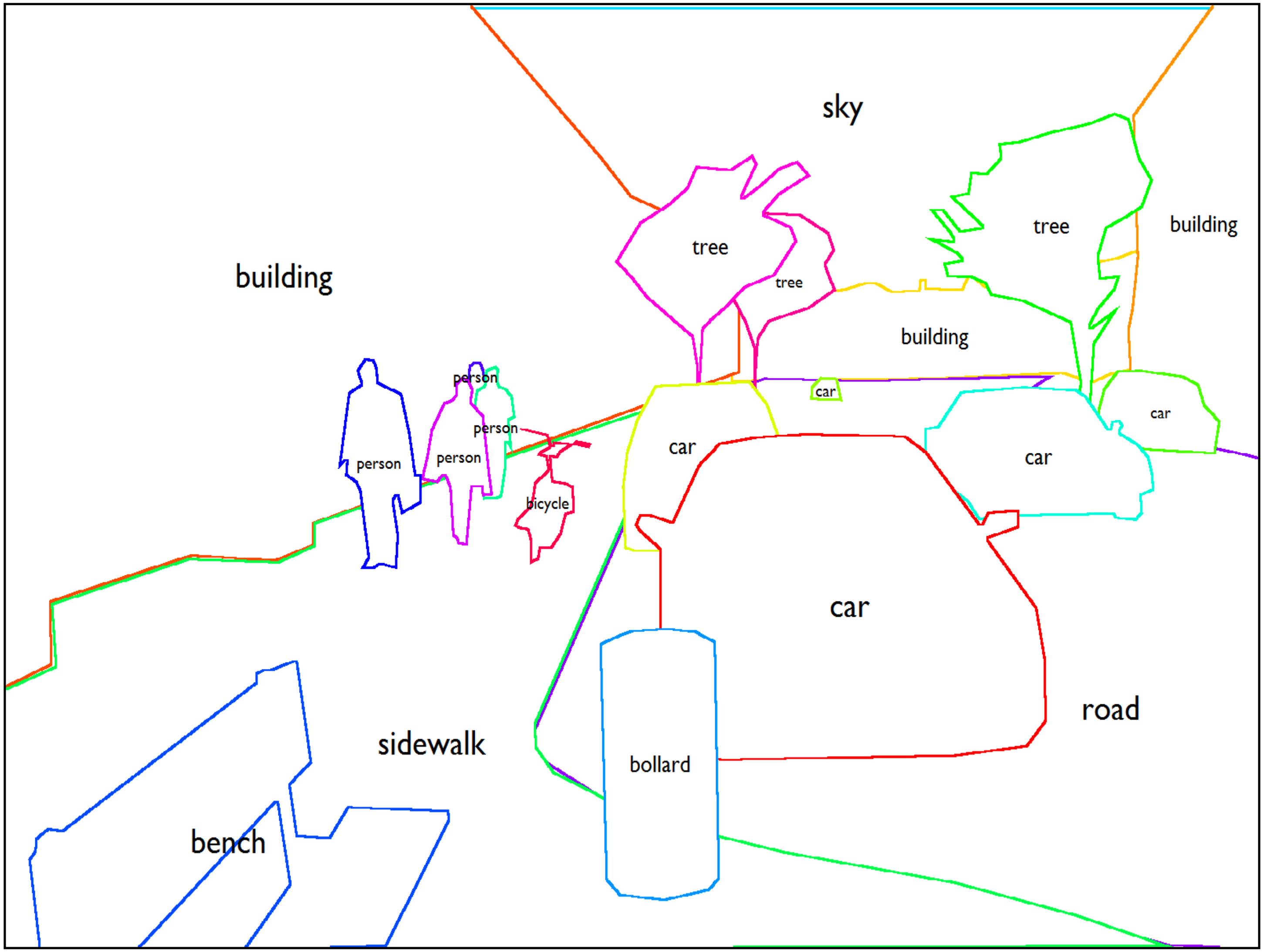
Object position

Object orientation

Object shape

Depth/occlusions

Object appearance



[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type

Scene geometry

Object classes

Object position

Object orientation

Object shape

Depth/occlusions

Object appearance

Illumination

Shadows

[Slide Credits: John Winn, ICML 2008]

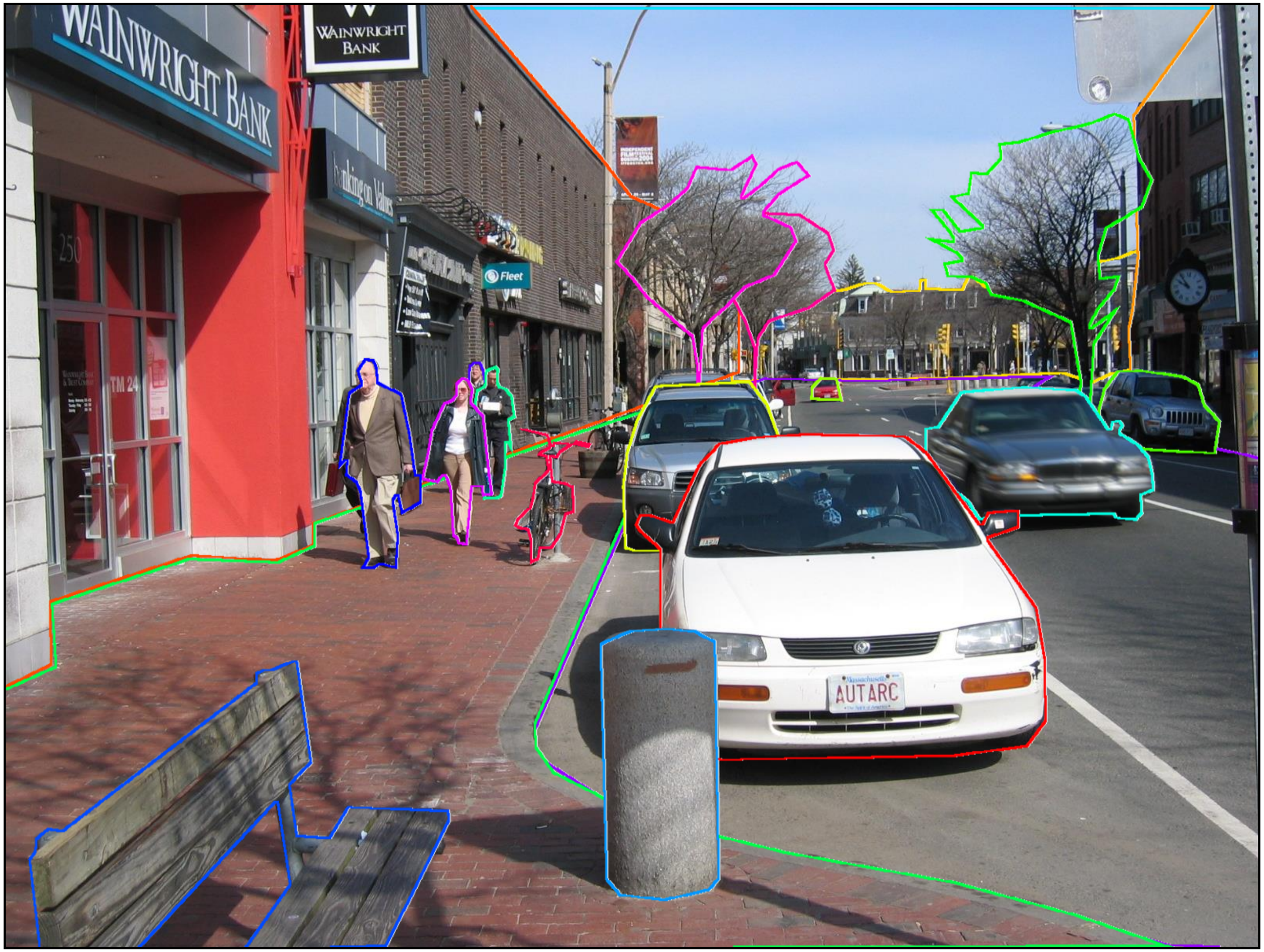
Image-formation model

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows

[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows
Motion blur
Camera effects



[Slide Credits: John Winn, ICML 2008]

Image-formation model

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows
Motion blur
Camera effects

[Slide Credits: John Winn, ICML 2008]

The “Scene Parsing” challenge --- a “grand challenge” of computer vision

Single image

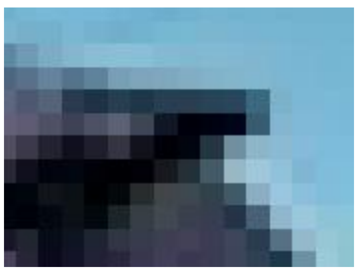
(Probabilistic) Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

Many applications do not have to extract the full probabilistic script but only a subset, e.g. “does the image contain a car?”

... many examples to come later

Why is “scene parsing” hard?

Computer Graphics



2D pixel representation

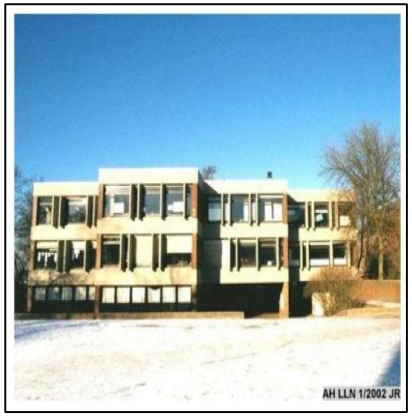
3D Rich Representation,

Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

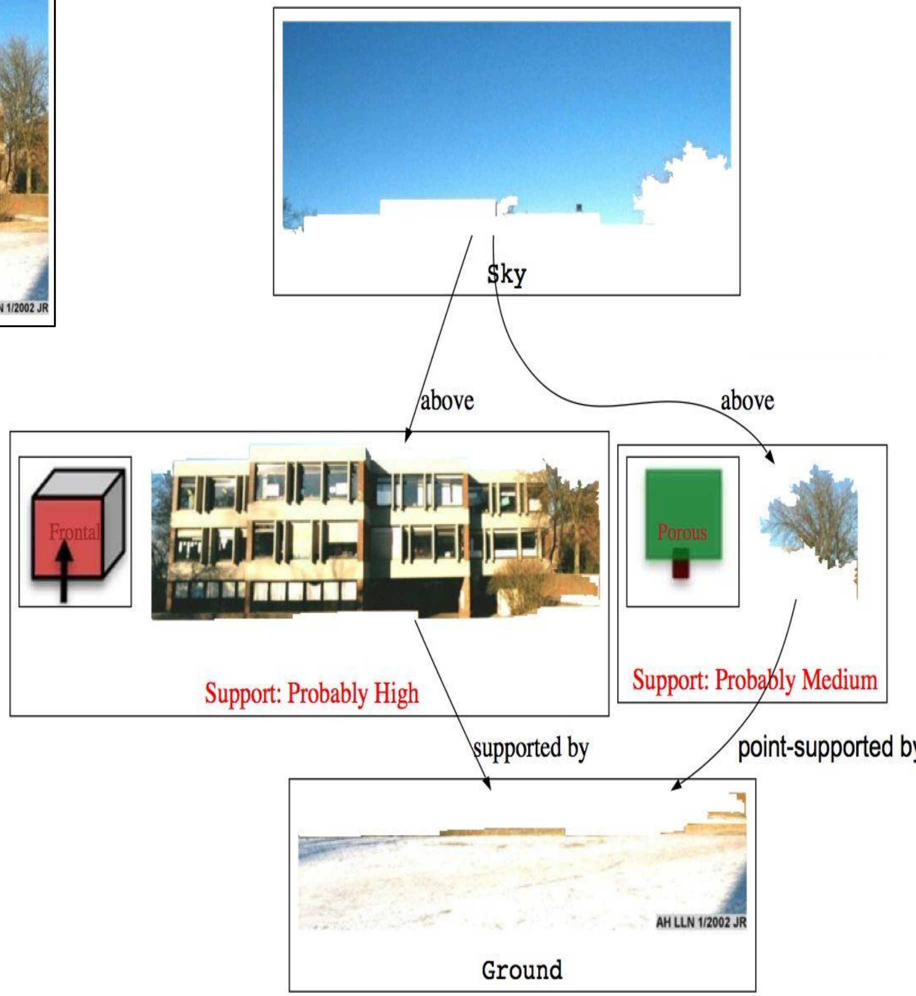
Computer Vision

Computer Vision can be seen as “inverse graphics”

Example of a recent work



Input



Scene graph

[Gupta, Efros, Herbert, ECCV ‘10]

Why is “scene parsing” hard?

<u>Subgoal for July</u>

Analysis of scenes consisting of non-overlapping objects from the

balls

bricks with faces of the same or different colors or textures

cylinders.

Each face will be of uniform and distinct color and/or texture.

<u>Extensions for August</u>

The first priority will be to handle objects of the same sort but with complex surfaces and backgrounds, e.g. cigarette pack with writing and bands of different color, or a cylindrical battery.

[Xiao et al. NIPS 2012]

[Sussman, Lamport, Guzman 1966]

[Slide credits Andrew Blake]

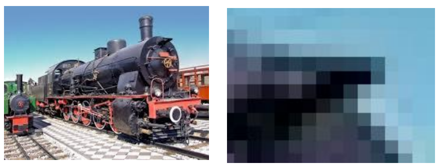
Introduction to Computer Vision

What is computer Vision?

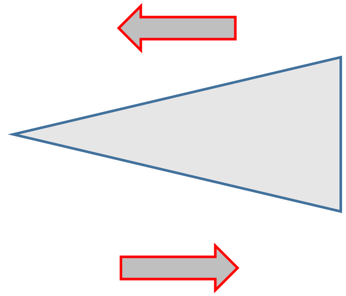
(Potential) Definition:
Developing computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in.

How can we interpret visual data?

Computer Graphics



2D pixel representation



3D Rich Representation,

Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

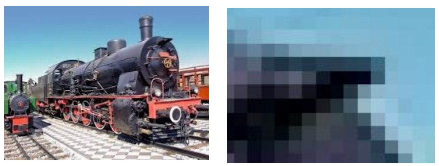
Computer Vision

- What general (prior) knowledge of the world (not necessarily visual) can be exploit?
- What properties / cues from the image can be used?

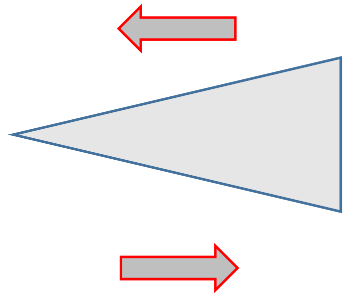
Both aspects are quite well understood (a lot is based on physics) … but how to use them is efficiently is open challenged (see later)

How can we interpret visual data?

Computer Graphics



2D pixel representation



3D Rich Representation,

Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

Computer Vision

- What general (prior) knowledge of the world (not necessarily visual) can be exploit?
- What properties / cues from the image can be used?

Both aspects are quite well understood (a lot is based on physics) … but how to use them is efficiently is open challenged (see later)

Prior knowledge (examples)

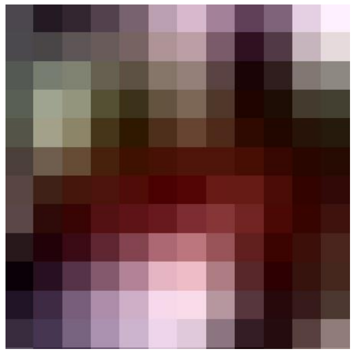
- "Hard" prior knowledge
  - Trains do not fly in the air
  - Objects are connected in 3D

- "Soft" prior knowledge:
  - The camera is more likely 1.70m above ground and not 0.1m.
  - Self-similarity: "all black pixels belong to the same object"

Prior knowledge – harder to describe

- Describe Image Texture

Real Image

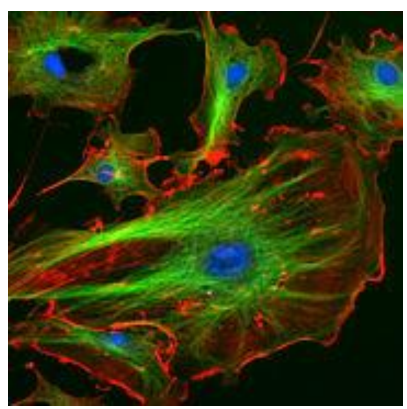


zoom

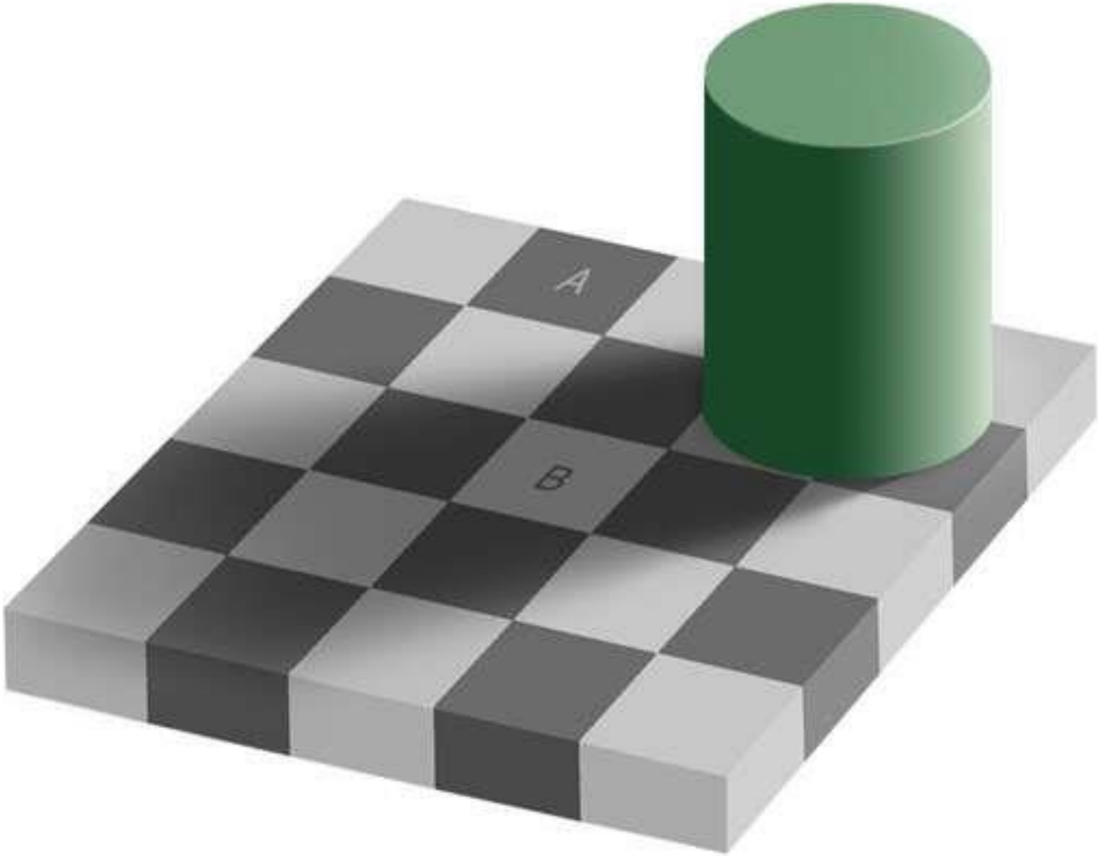
Not a real Image

zoom

- Microscopic Images. What is the true shape of these objects



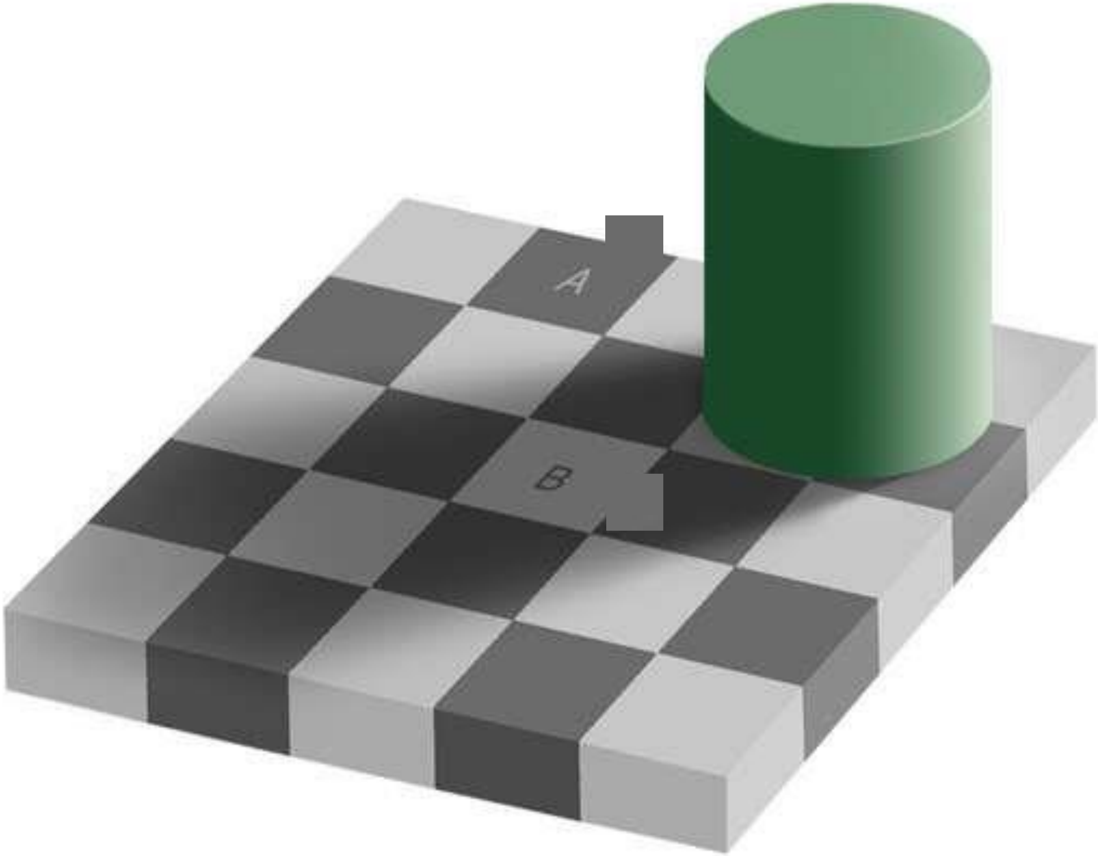
The importance of Prior knowledge



Which patch is brighter: A or B?

[Edward Adelson]

The importance of Prior knowledge

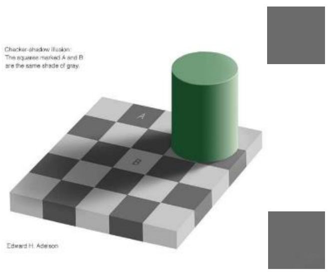


Which patch is brighter: A or B?

[Edward Adelson]

The importance of Prior knowledge

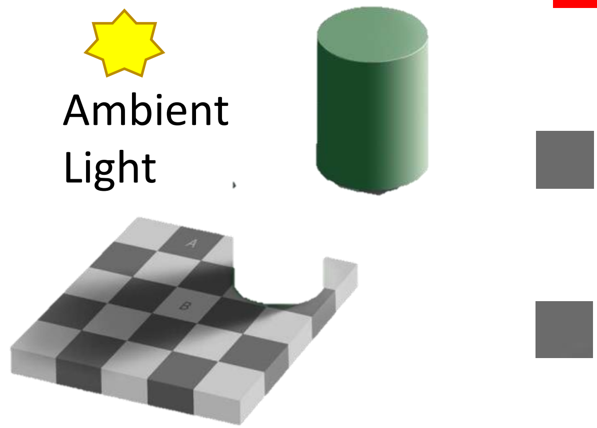
2D



2D Image - local

What the
computer sees

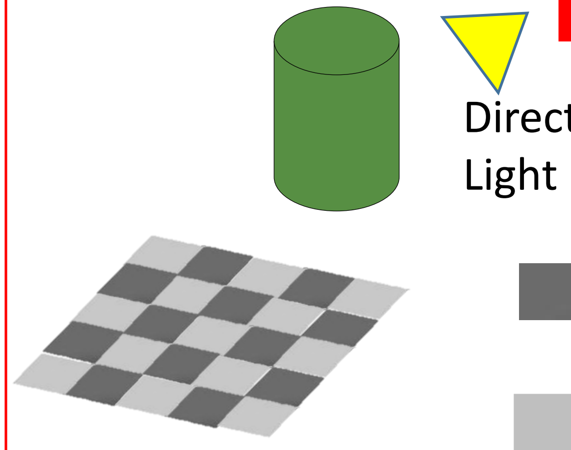
3D



True colors
in 3D world

An unlikely
3D representation
(hard to see for a human)

3D

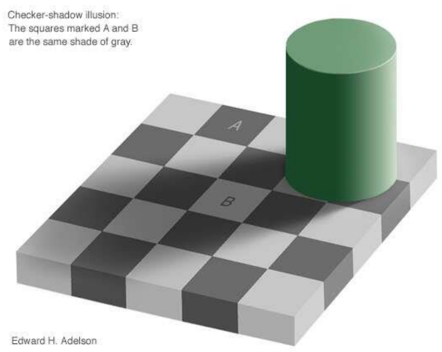


True colours
In 3D world

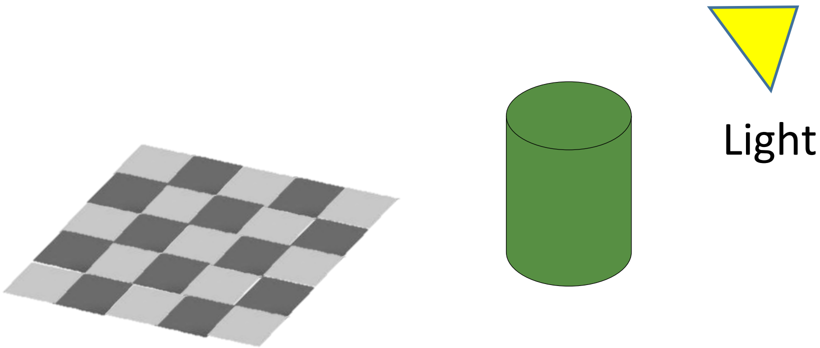
The most likely
3D representation

This is what humans see
implicitly. Ideally the computer
sees the sane.

The importance of Prior knowledge



2D Image



3D representation

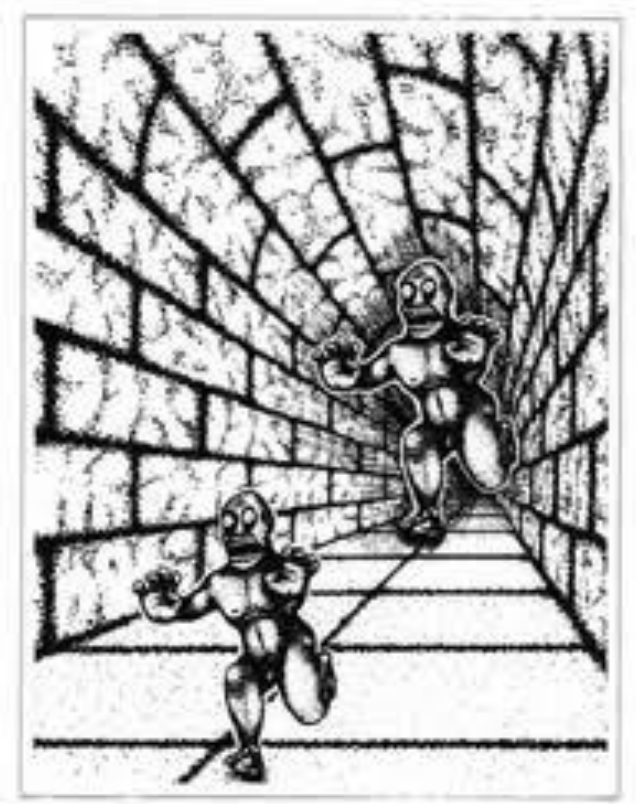
Humans see an image **not** as a set of 2D pixels. They understand an image as a projection of the 3D world we live in.

Humans have the prior knowledge about the world encoded, such as:

- Light cast shadows
- Objects do not fly in the air
- A car is likely to move but a table is unlikely to move

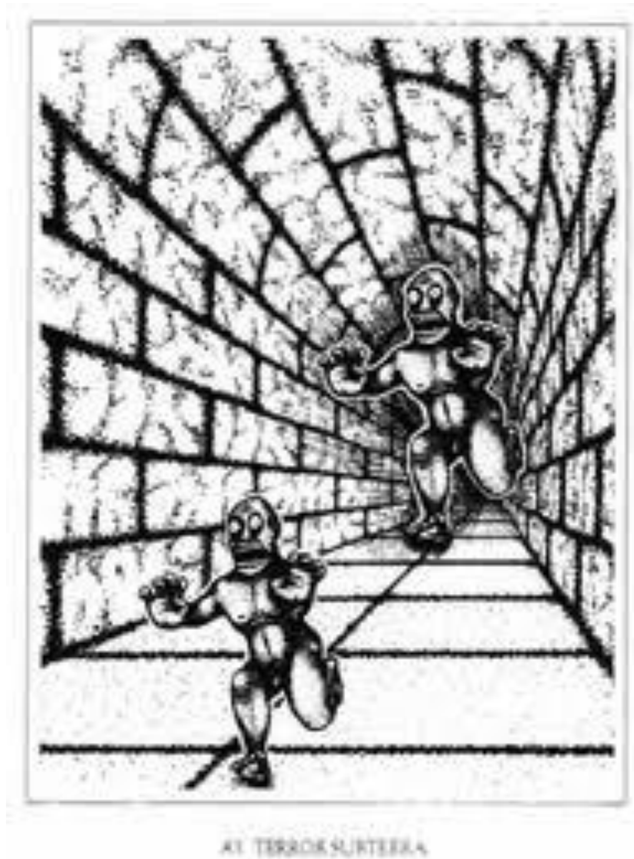
We have to teach the computer this prior knowledge to understand 2D images as picture of the 3D world

The importance of Prior knowledge



Which monster is bigger?

The importance of Prior knowledge



Which monster is bigger?

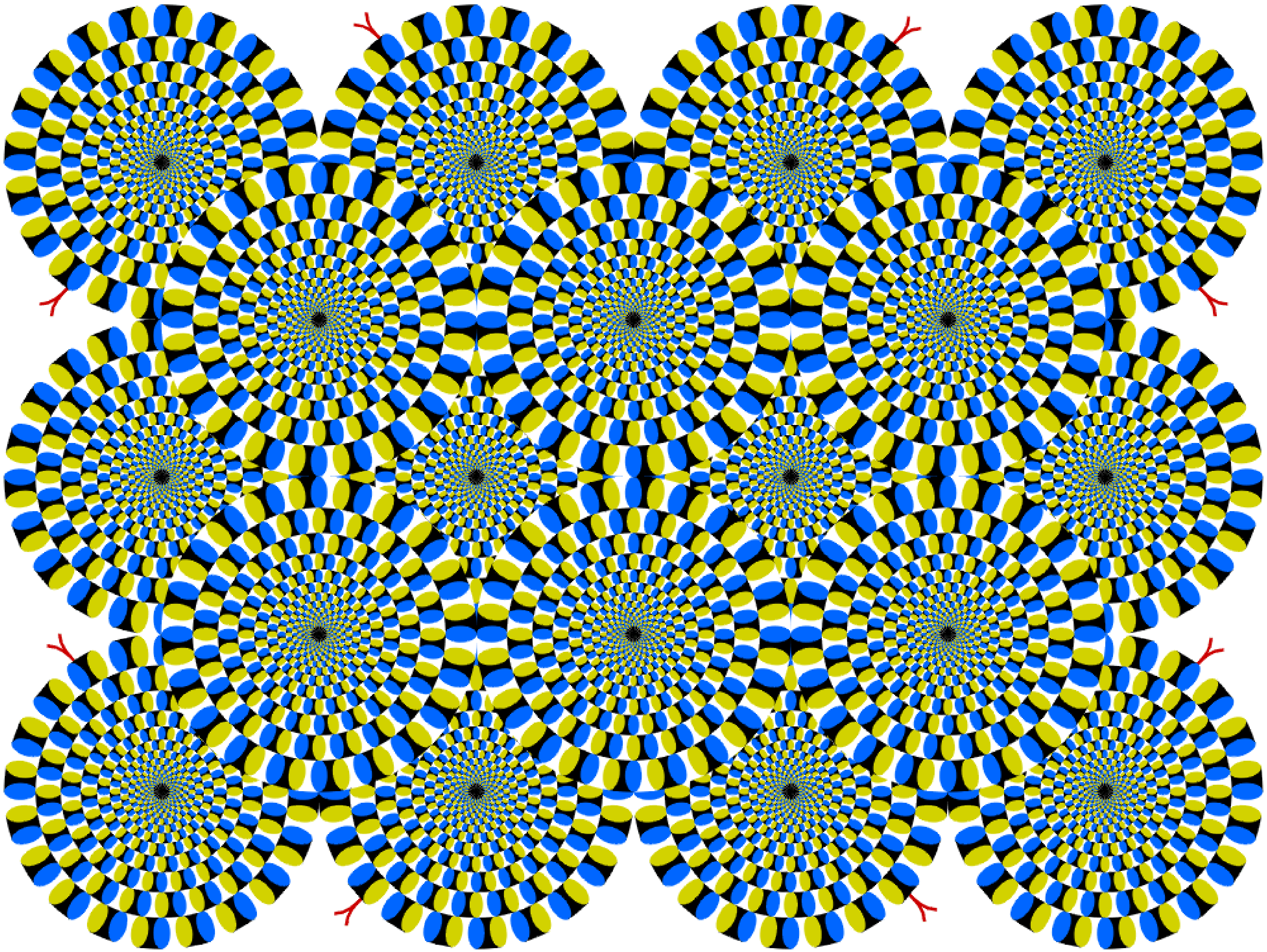
In the 2D Image

1meter

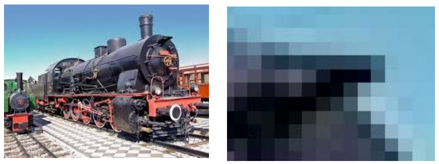
2meter

In the 3D world (true)

Human Vision can be fooled

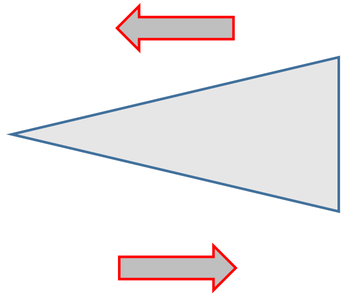


How can we interpret visual data?



2D pixel representation

Computer Graphics



Computer Vision

3D Rich Representation,

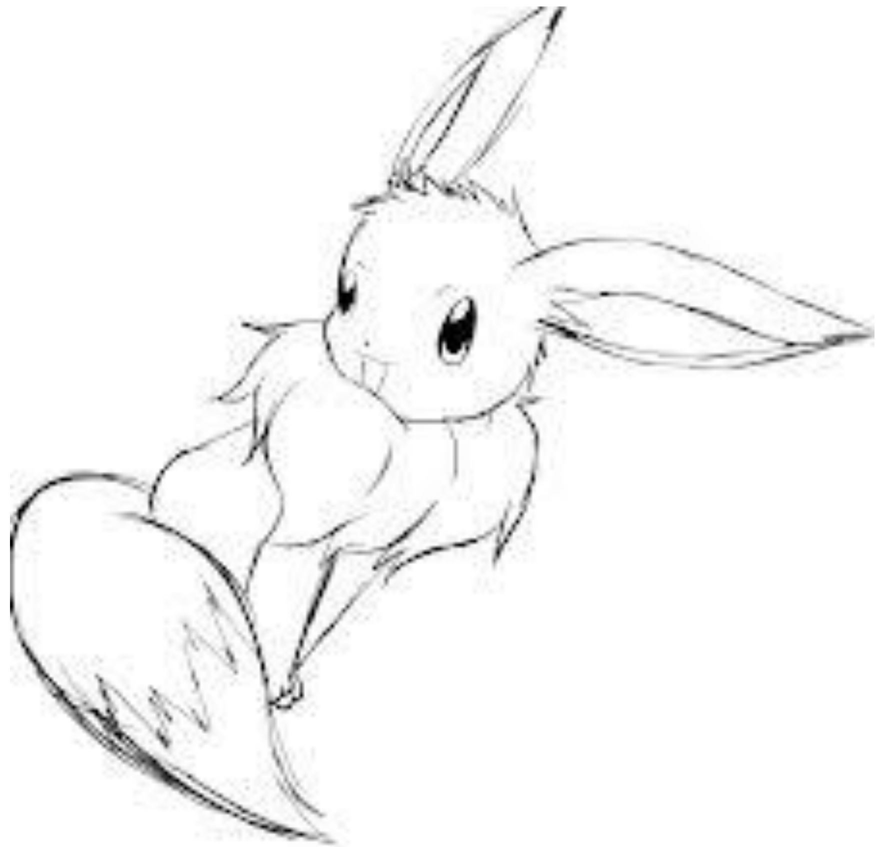
Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

- What general (prior) knowledge of the world (not necessarily visual) can be exploit?
- What properties / cues from the image can be used?

Both aspects are quite well understood (a lot is based on physics) … but how to use them is efficiently is open challenged (see later)

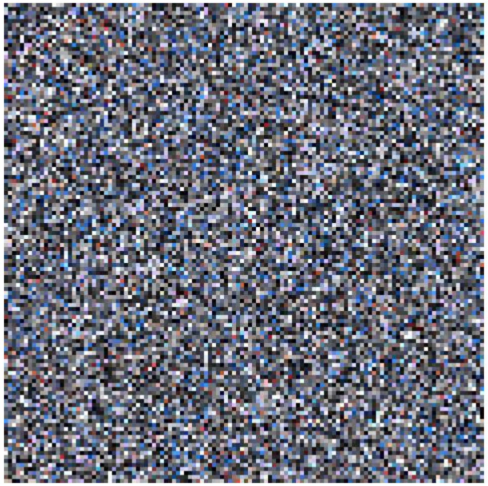
Cue: Appearance (Colour, Texture) for object recognition

Cue: Outlines (shape) for object recognition



Guess the Object

- Colour



- Texture

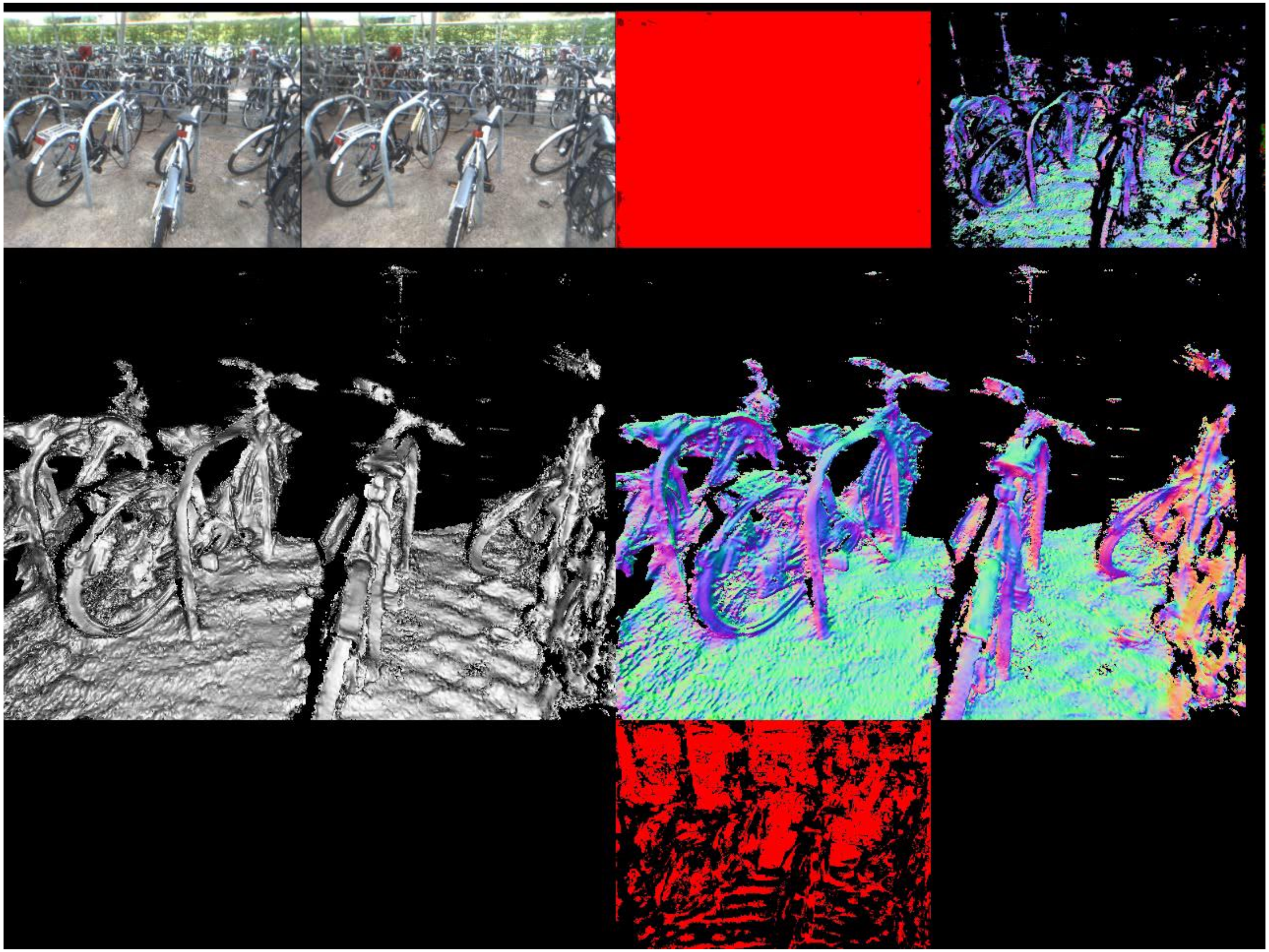
- Shape

[from JohnWinn ICML 2008]

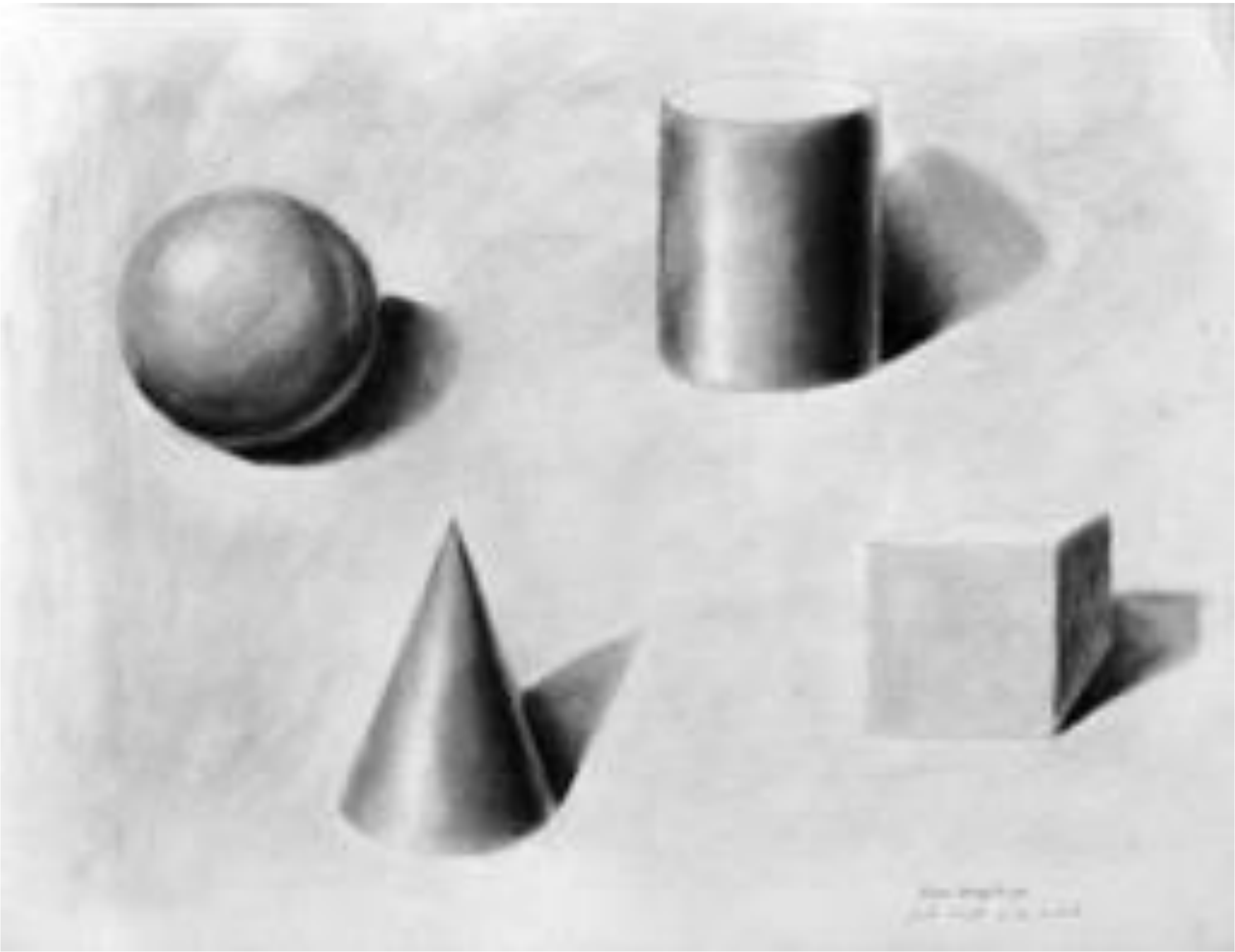
Cue: Context for object recognition

Cue: Context for object recognition

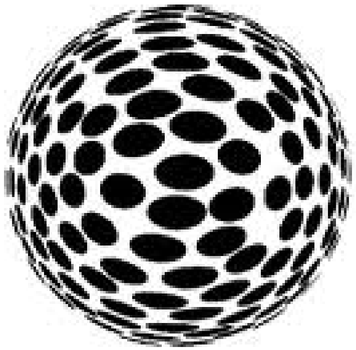
Cue: Multiple Frames for geometry estimation

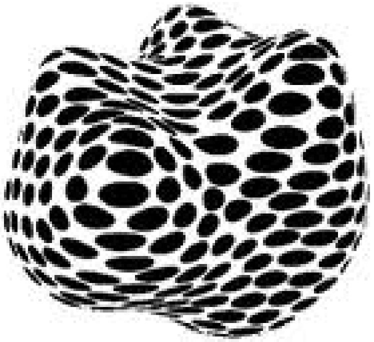


Cue: Shading & shadows for geometry and Light estimation



Texture gradient for geometry estimation





The "Scene Parsing" challenge --- a "grand challenge" of computer vision

Single image

(Probabilistic) Script = {Camera, Light, Geometry, Material, Objects, Scene, Attributes, Others}

Many applications do not have to extract the full probabilistic script but only a subset, e.g. "does the image contain a car?"

… many examples to come later

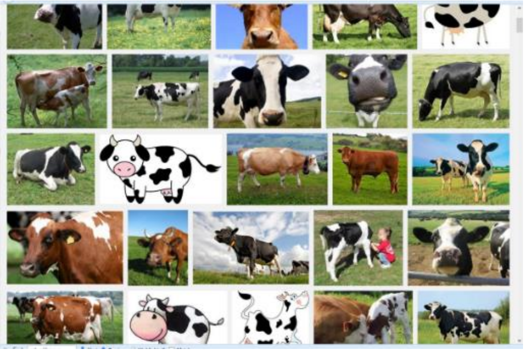
Many application scenarios are in reach

<u>To simplify/tackle the problem:</u>

1) **Richer Input:**
   Modern sensing technology; Video;
   Cameras everywhere

2) **Rich Models**: Deep Learning

3) **Lots of Data** to learn from:
   search engines; crowdsourcing;
   graphics engines



4) For **many practical** applications:
   We do not have to infer the full probabilistic script

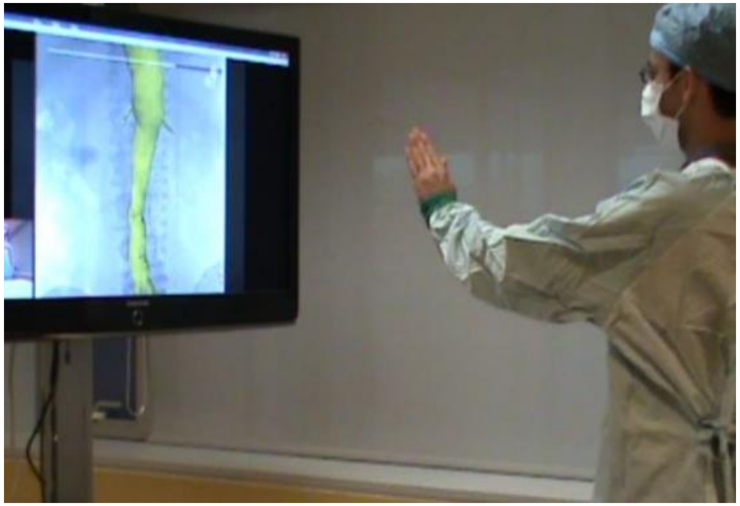
Kinect has simplified computer vision

[Izadi et al. ´11]

Animate the world

[Chen et al. UIST ‘12]

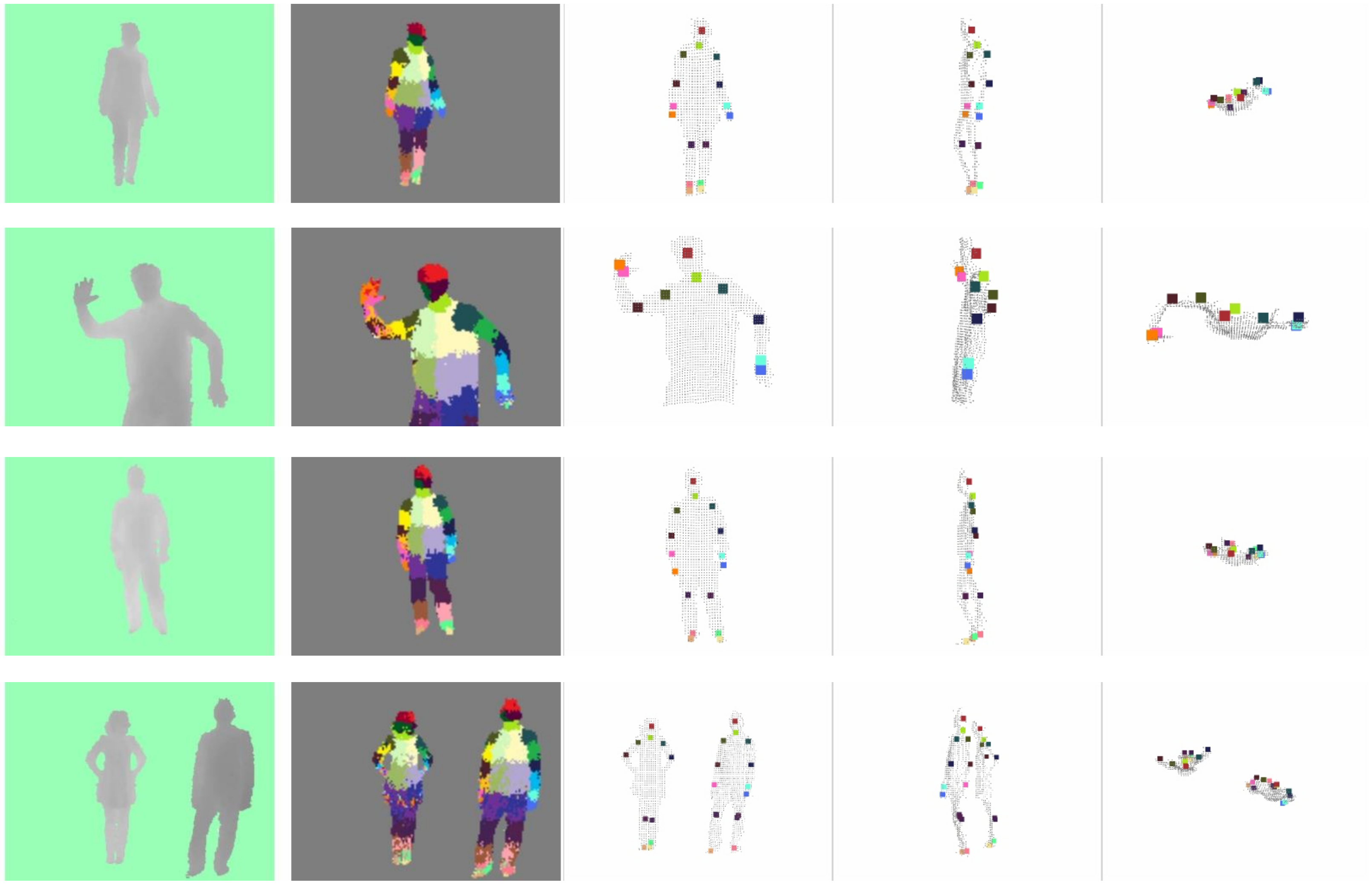
Kinect Body tracking and Gesture Recognition



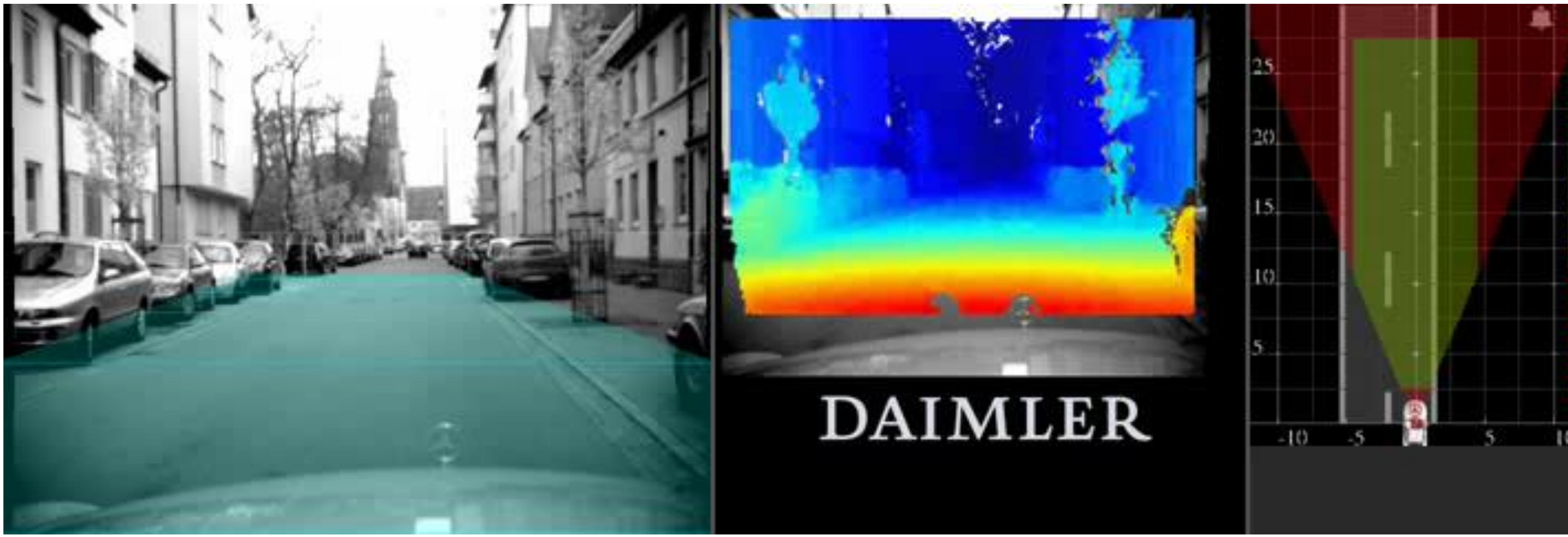
Start-Up 2012: Try Fashion online

Very large impact in many field: Gaming, Robotics, HCI, Medicine, …

Kinect Body Pose estimation and tracking

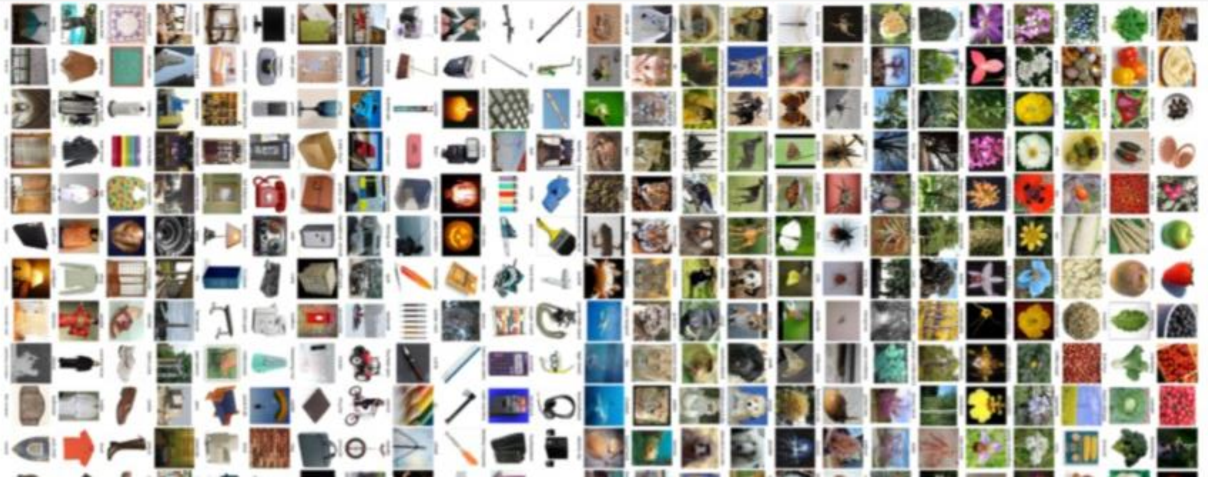


Real-time pedestrian detection

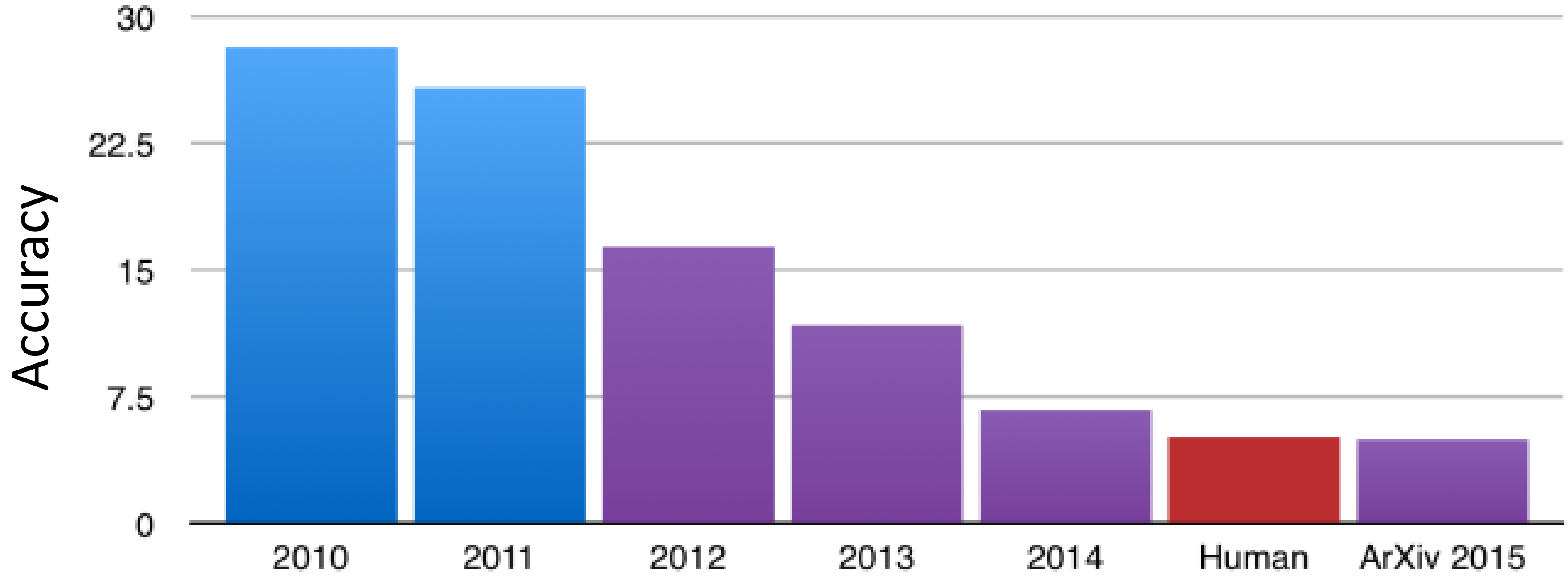


Daimler Research Lab

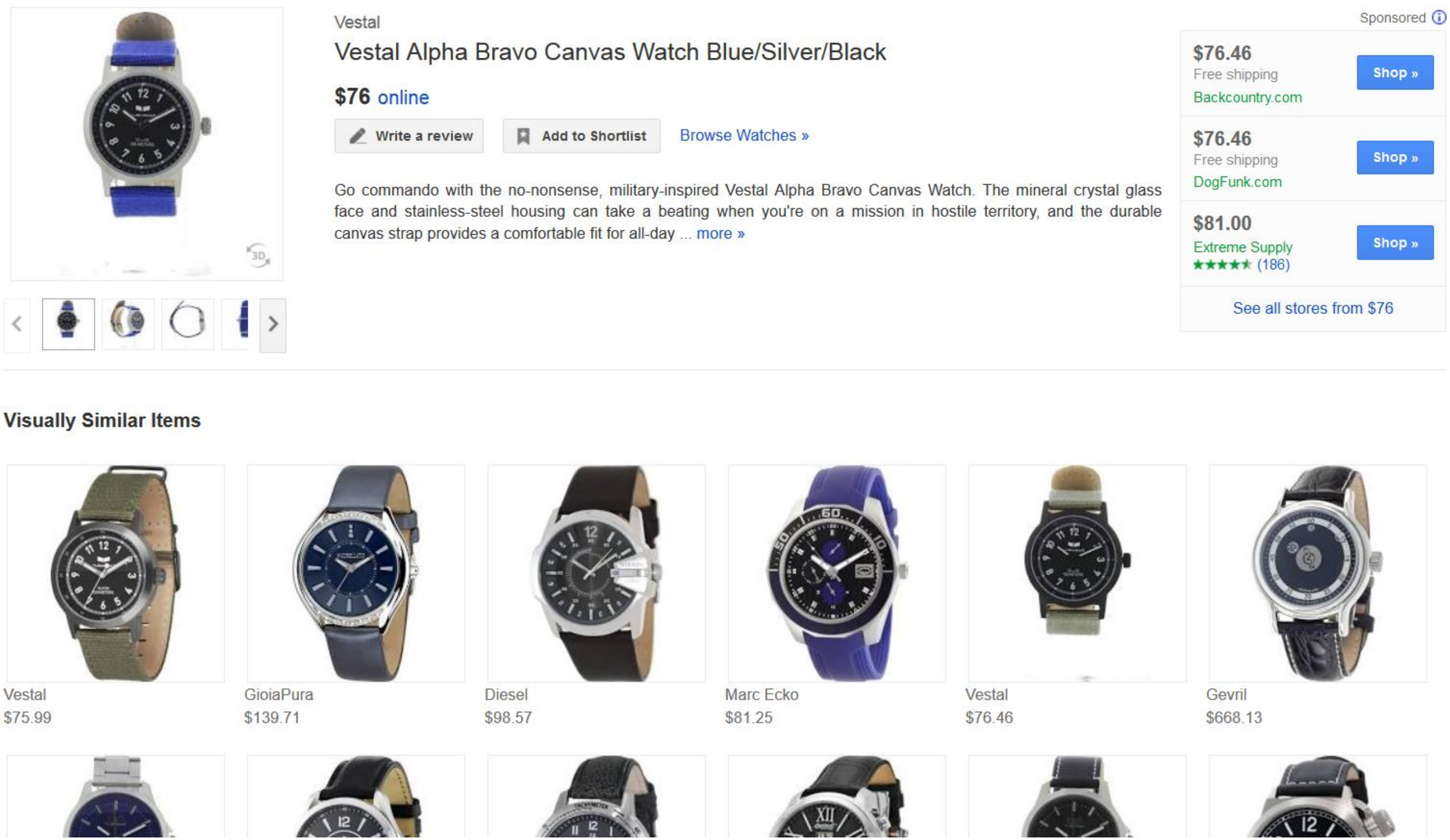
Object recognition – ImageNet



What's in the image (1000 classes)



Start-up Company: Like.com



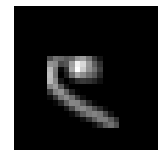
Interactive Image manipulation

[Agrawal et al '04]

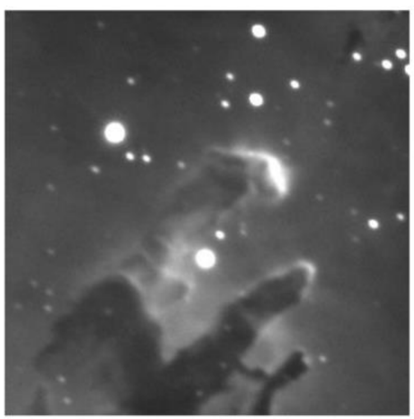
Image de-convolution

Input

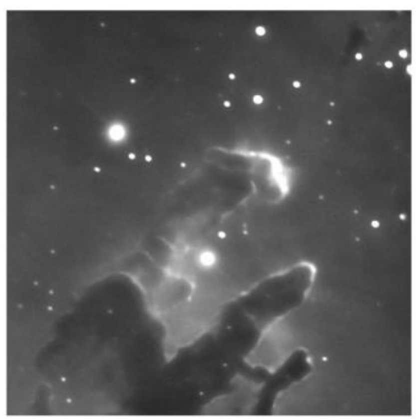
Output



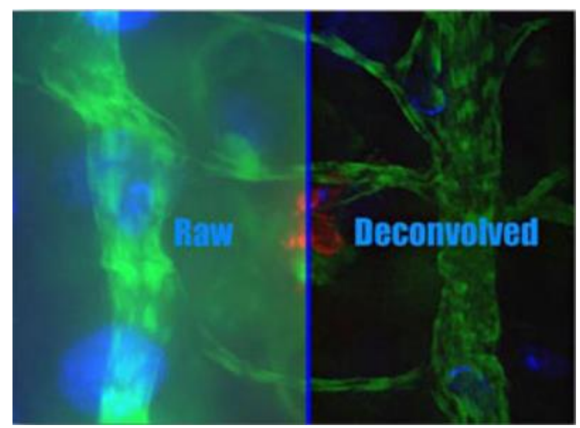
kernel



input



output



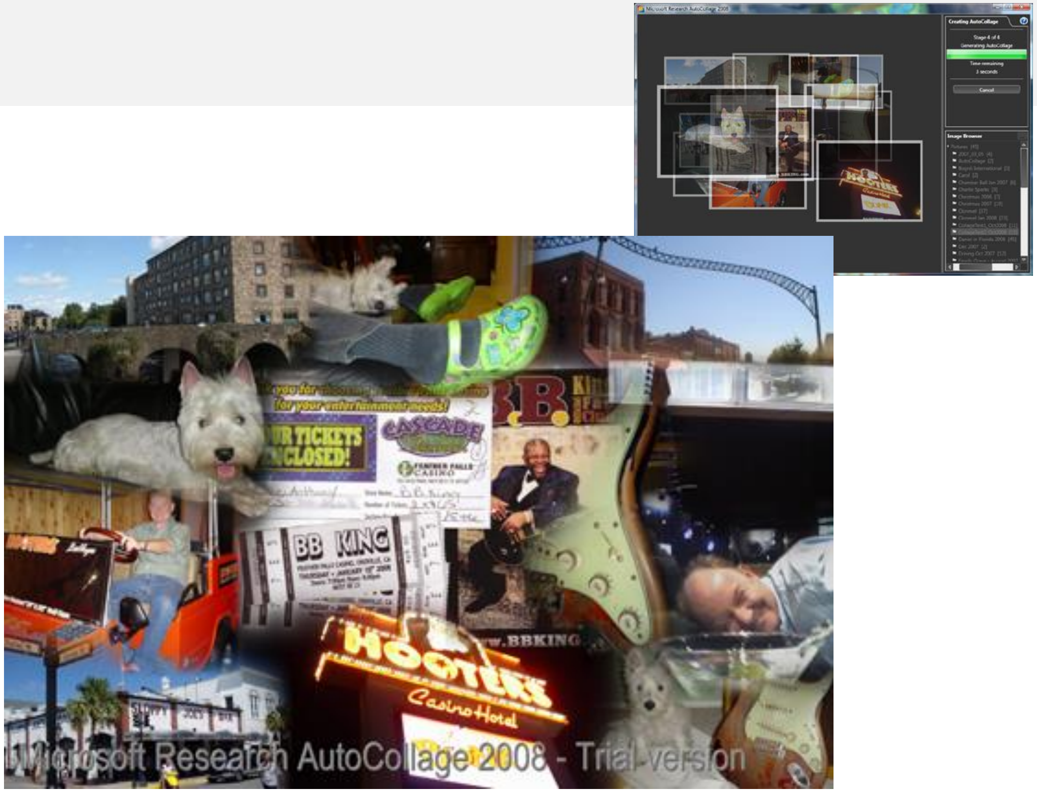
[Schmidt, Rother, Nowozin, Jancsary, Roth 2013] Best Student Paper award

Video Editing

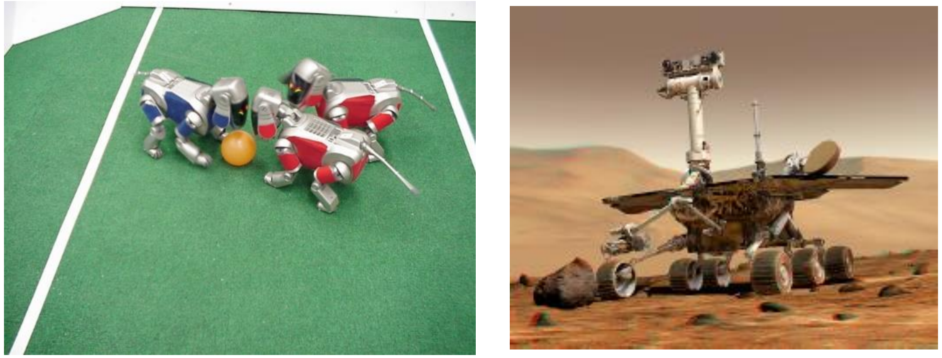
[Rav-Acha et al. ‘08]

Industry

Pirates of the Caribbean,
Industrial Light and Magic



AutoCollage 2008 - Microsoft Research
[Rother et al. Siggraph 2006]



Robotics

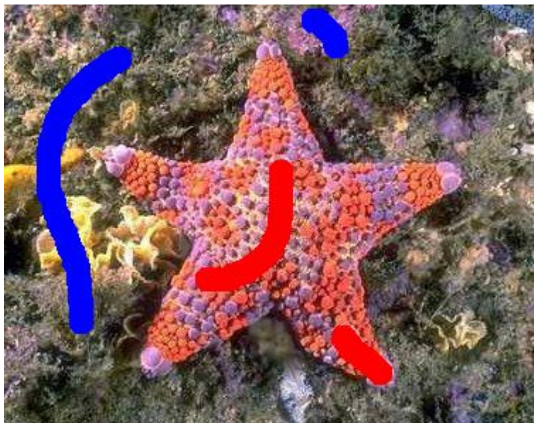
Introduction to Computer Vision

What is computer Vision?

(Potential) Definition:
Developing computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in.

Model versus Algorithm

Example: Interactive Segmentation



$\boldsymbol{z} = (R, G, B)^n$

Goal

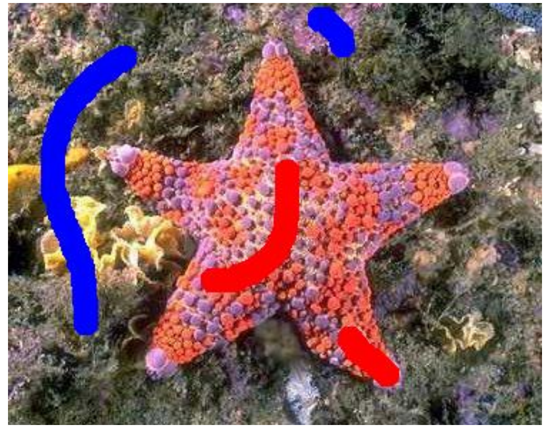
$\mathbf{x} = \{0,1\}^n$

Given **z**; derive binary **x**:

Model: Energy function $\boldsymbol{E}(\boldsymbol{x})$

Algorithm to minimization: $\boldsymbol{x}^* = argmin_x\, E(\boldsymbol{x})$

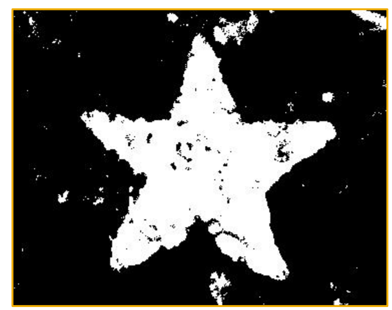
Model for a starfish



Goal: formulate $\boldsymbol{E}(\boldsymbol{x})$ such that

$\boldsymbol{E}(\boldsymbol{x}) = 0.01$

$\boldsymbol{E}(\boldsymbol{x}) = 0.05$



$\boldsymbol{E}(\boldsymbol{x}) = 0.05$

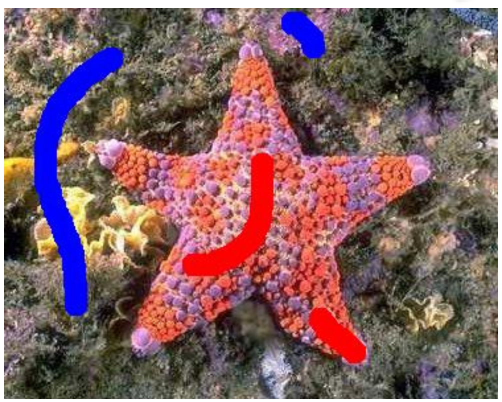
$\boldsymbol{E}(\boldsymbol{x}) = 0.1$

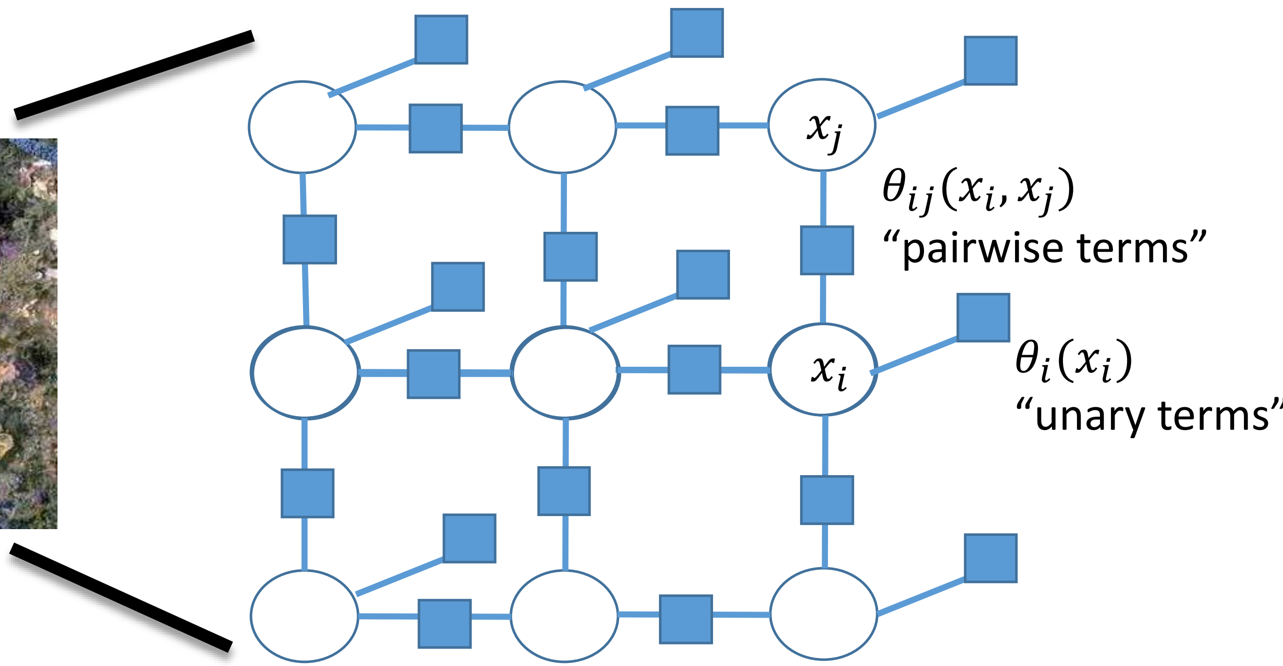
Optimal solution $\boldsymbol{x}^* = argmin_x\, E(\boldsymbol{x})$

How does the energy looks like?

Energy function (sum of local terms):

$$\boldsymbol{E}(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$





Undirected graphical model

This is the focus of machine Learning 1 and 2

Why is computer vision interesting (to you)?

- It is a challenging problem that is far from being solved
- It combines insights and tools from many fields and disciplines:
    - Mathematics and statistics
    - Cognition and perception
    - Engineering (signal processing)
    - Computer science

Why is computer vision interesting (to you)?

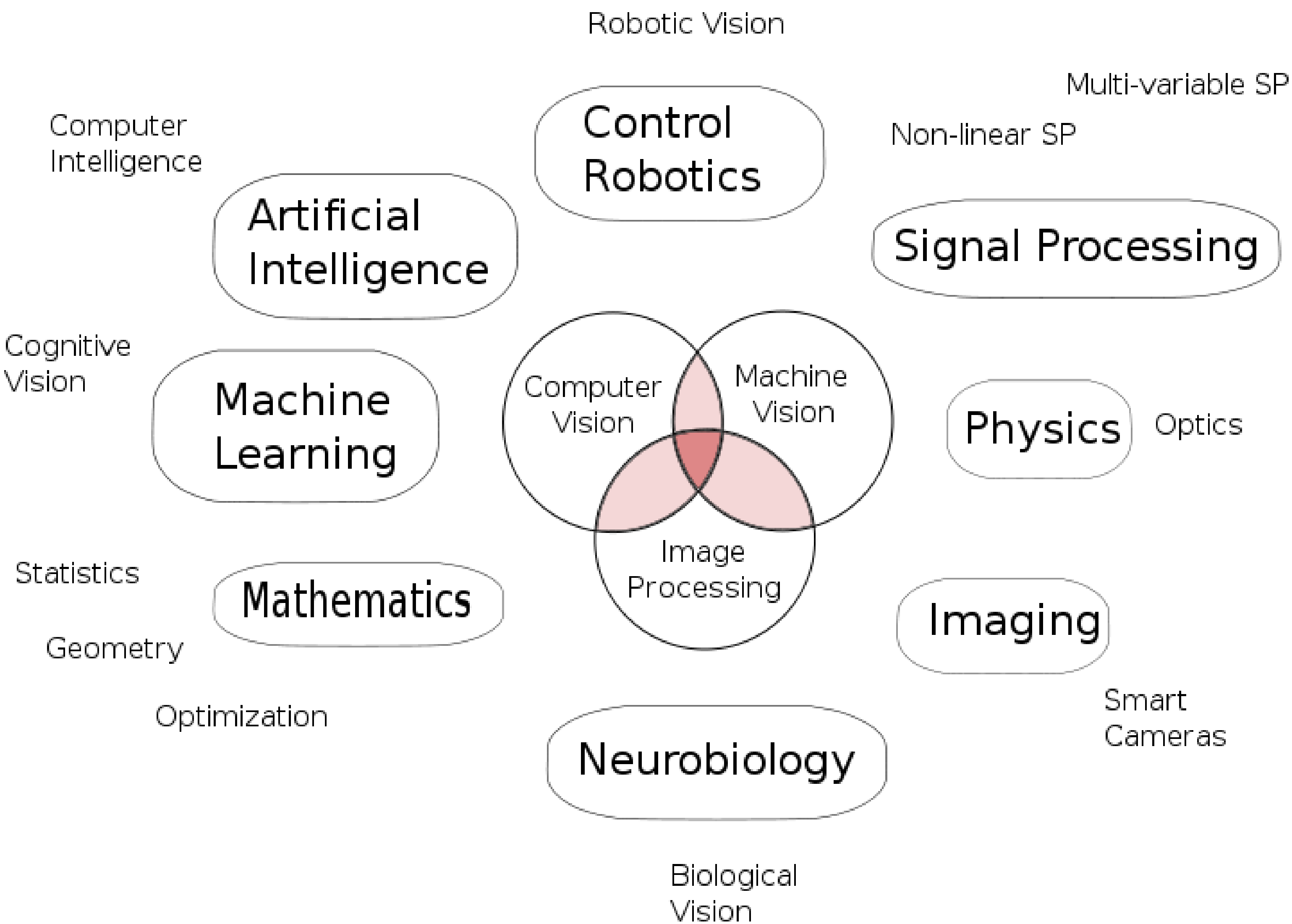
- Allows you to apply theoretical skills

  ... that you may otherwise only use rarely

- Quite rewarding:
  - Often visually intuitive and encouraging results

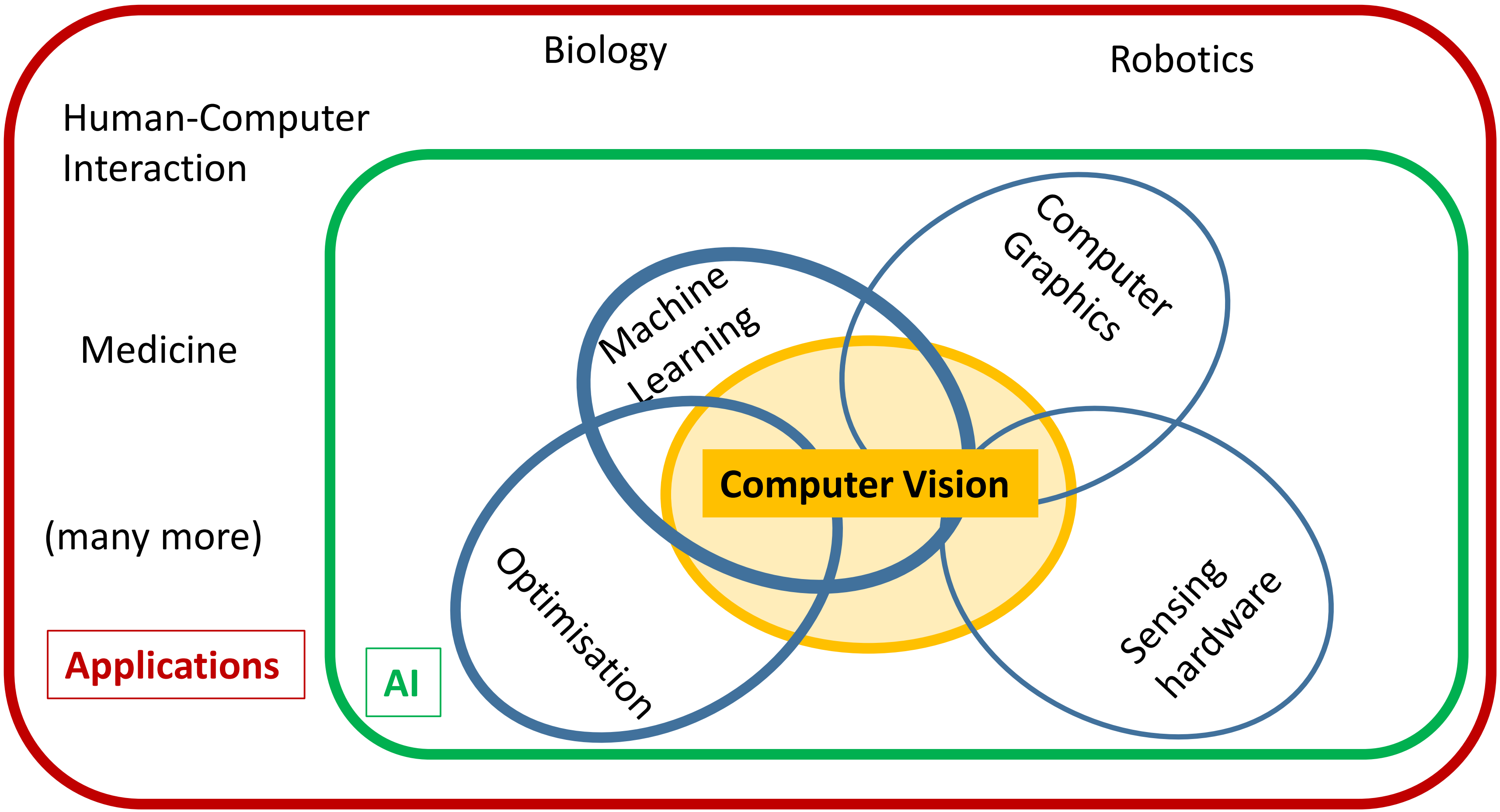
- It is a growing field:
  - Cameras are becoming more and more popular
  - There are a lot of companies (big, small, start-up) working in vision
  - Conferences are growing rapidly
  - Deep Learning has revolutionised the field

Relationship to other fields



[Wikipedia]

Relationship to other fields – my personal view



Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

What is an Image

- We can think of the image as a function:

$$I(x, y), \qquad I\colon\ \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

  - For every 2D point (pixel) it tells us the amount of light it receives
  - The size and range of the sensor is limited:

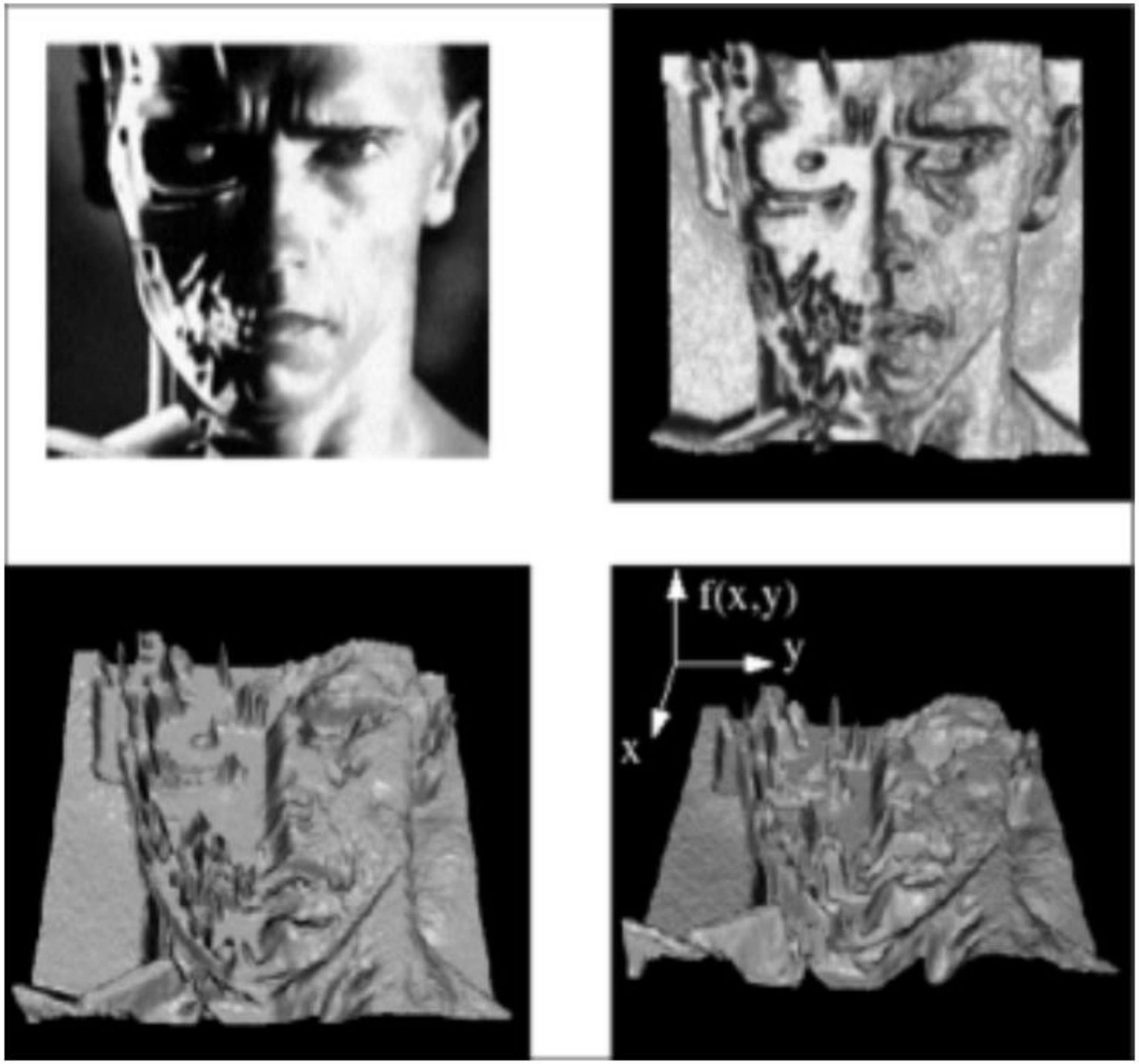
$$I(x, y), \qquad I\colon [a, b] \times [c, d] \rightarrow [0, m]$$

  - Colour image is then a vector-valued function:

$$I(x, y) = \begin{pmatrix} I_R(x, y) \\ I_G(x, y) \\ I_B(x, y) \end{pmatrix}, \qquad I\colon [a, b] \times [c, d] \rightarrow [0, m]^3$$

  - Comment, in most lectures we deal with grey-valued images and extension to colour is “obvious”

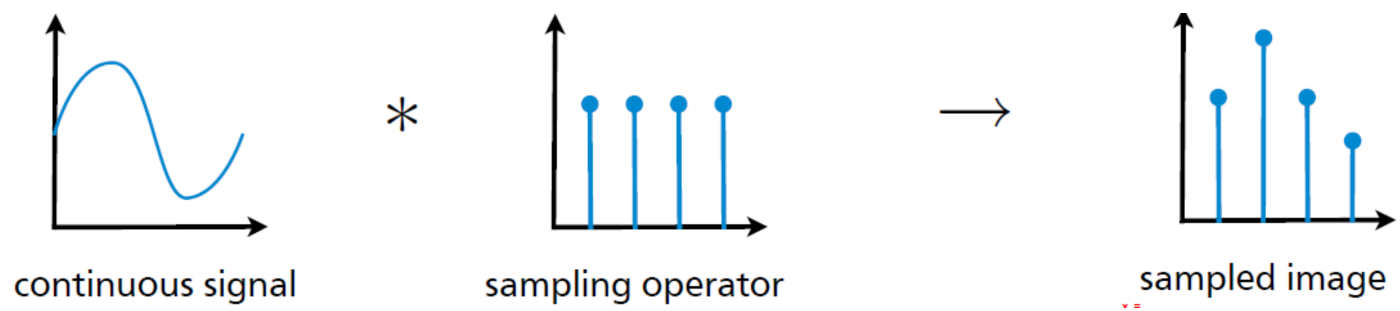
Images as functions



[from Steve Seitz]

Digital Images

- We usually do not work with spatially continuous functions, since our cameras do not sense in this way.
- Instead we use (spatially) discrete images
- Sample the 2D domain on a regular grid (1D version)



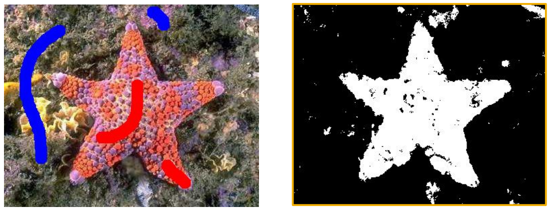
- Intensity/color values usually also discrete. Quantize the values per channel (e.g. 8 bit per channel)

y = \ x =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

Comment on Continuous Domain / Range

- There is a branch of computer vision research ("variational methods"), which operates on continuous domain for input images and output results

- Continuous domain methods are typically used for physics-based vision: segmentation, optical flow, etc.



- In this lecture and other lectures we mainly operate in discrete domain and discrete or continuous range for output results

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

Point operators

- Point operators work on every pixel independently:

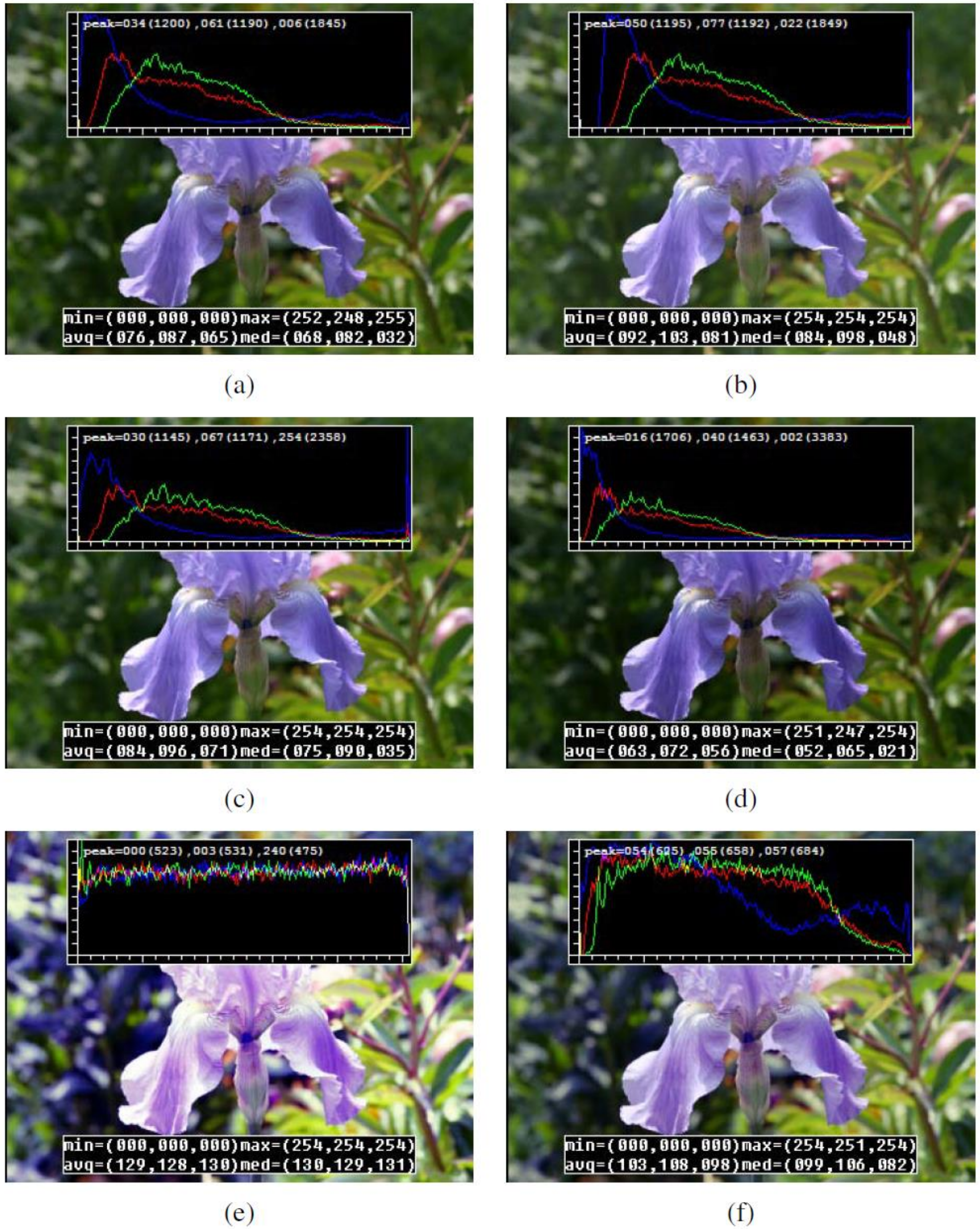
$$J(x, y) = h\big(I(x, y)\big)$$

- Examples for h:
  - Control contrast and brightness; $h(z) = az^b + c$

original

Contrast enhanced

Example


peak=034(1200),061(1190),006(1845)
min=(000,000,000)max=(252,248,255)
avg=(076,087,065)med=(068,082,032)
(a)
peak=050(1195),077(1192),022(1849)
min=(000,000,000)max=(254,254,254)
avg=(092,103,081)med=(084,098,048)
(b)
peak=030(1145),067(1171),254(2358)
min=(000,000,000)max=(254,254,254)
avg=(084,096,071)med=(075,090,035)
(c)
peak=016(1706),040(1463),002(3383)
min=(000,000,000)max=(251,247,254)
avg=(063,072,056)med=(052,065,021)
(d)
peak=000(523),003(531),240(475)
min=(000,000,000)max=(254,254,254)
avg=(129,128,130)med=(130,129,131)
(e)
min=(000,000,000)max=(254,251,254)
avg=(103,108,098)med=(099,106,082)
(f)

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

Linear Filters / Operators

- Properties:
  - Homogeneity: $T[aX] = aT[X]$
  - Additivity: $T[X + Y] = T[X] + T[Y]$
  - Superposition: $T[aX + bY] = aT[X] + bT[Y]$

- Example:
  - Convolution
  - Matrix-Vector operations

Convolution

- Replace each pixel by a linear combination of its neighbours and itself
- 2D convolution (discrete)

$$g = f * h$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

image
$f(x, y)$

*

Centred at 0,0

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

filter (kernel)
$h(x, y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

filtered image
$g(x, y)$

smaller output?

$$g(x,y) = \sum_{k,l} f(x-k, y-l)\,h(k,l)$$

"the image f is implicitly mirrored"

Convolution - Properties

- Linear $h * (f_0 + f_1) = h * f_0 + h * f_1$
- Associative $(f \; * g) * h = f * (g * h)$
- Commutative $f * h = \; h * f$
- Can be written in Matrix form: $g = H\, f$
- Correlation (not mirrored filter):

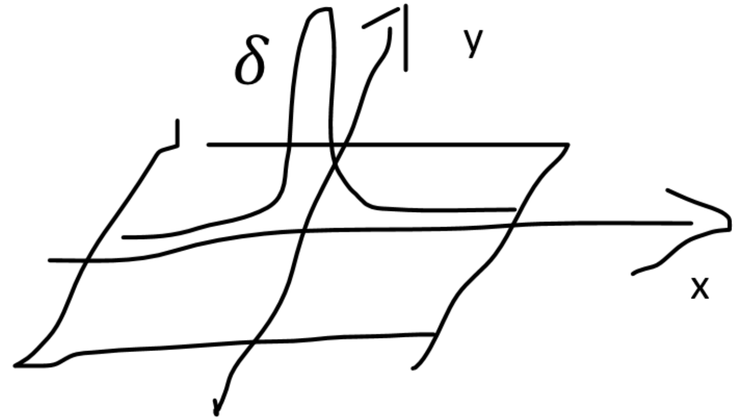
$$g(x,y) = \sum_{k,l} f(x+k, y+l) h(k,l)$$

$$\boxed{72}\boxed{88}\boxed{62}\boxed{52}\boxed{37} * \boxed{1/4}\boxed{1/2}\boxed{1/4} \Leftrightarrow$$

$$\frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix}\begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Examples

- Impulse function: $f = f * \delta$

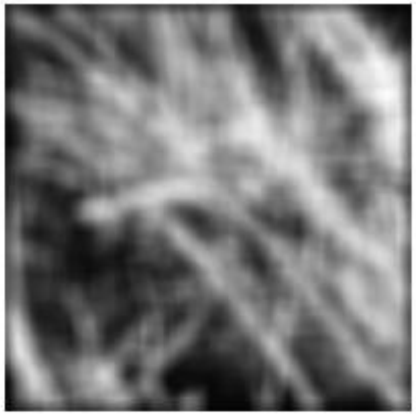


- Box Filter:

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

Original Image

Box-filtered image



Application: Noise removal

- Noise is what we are not interested in:
  sensor noise (Gaussian, shot noise), quantisation artefacts, etc

- Typical assumption is that the noise is not correlated between pixels

- Basic Idea: neighbouring pixel contain information about intensity

$$\begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 20 & 2 \\ \hline 3 & 2 & 3 \\ \hline\end{array} \longrightarrow \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 3 & 2 \\ \hline 3 & 2 & 3 \\ \hline\end{array}$$

The box filter does noise removal

- Box filter takes the mean in a neighbourhood

Image

Noise

Pixel-independent
Gaussian noise added

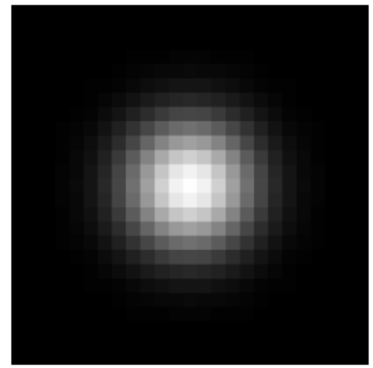
$$\frac{1}{9} \cdot \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

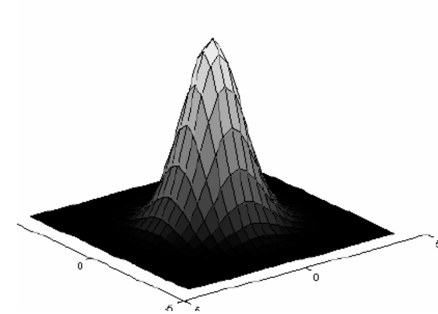
Filtered Image

Gaussian (Smoothing) Filters

- Nearby pixels are weighted more than distant pixels
- Isotropic Gaussian (rotational symmetric)

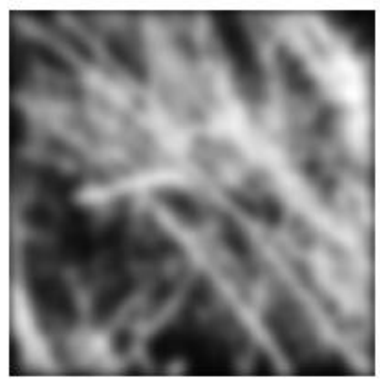
$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$



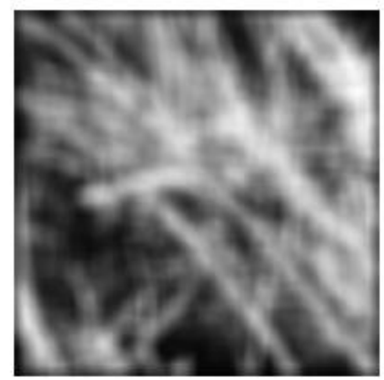


Original Image

Gaussian-filtered image



Box-filtered image

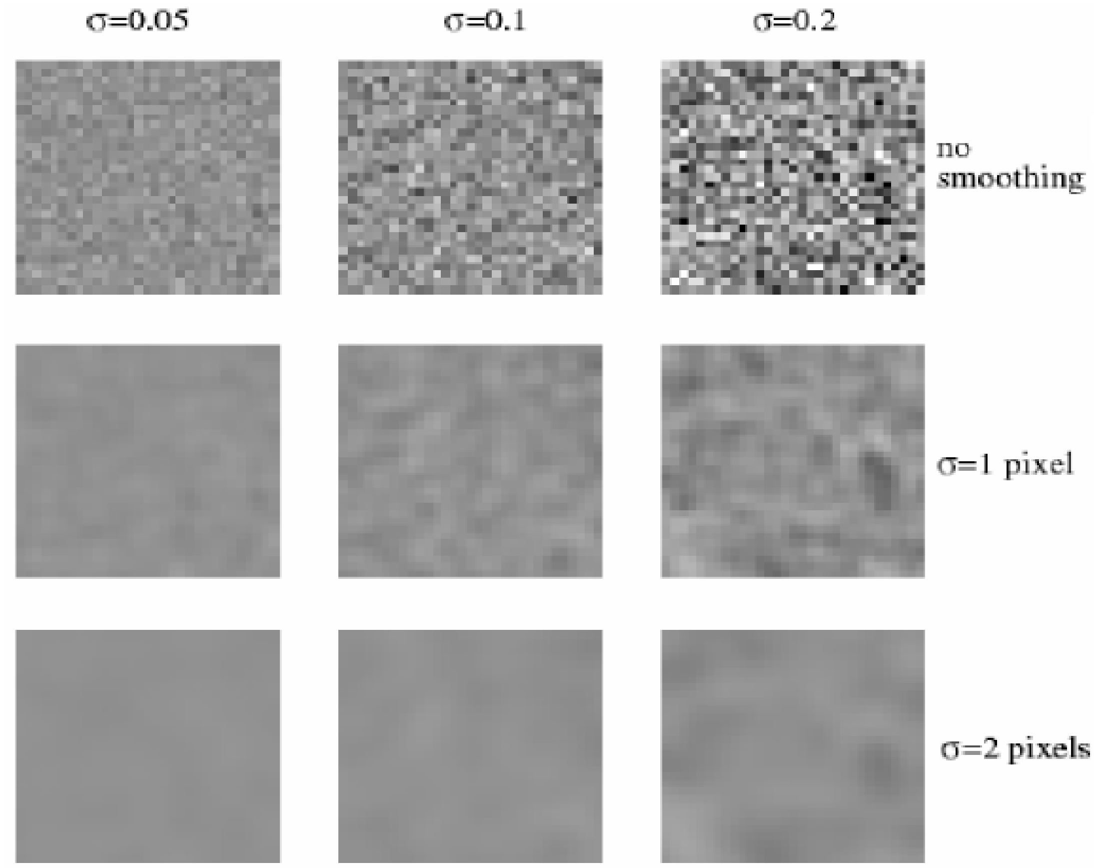


Gaussian Filter

Input: constant grey-value image

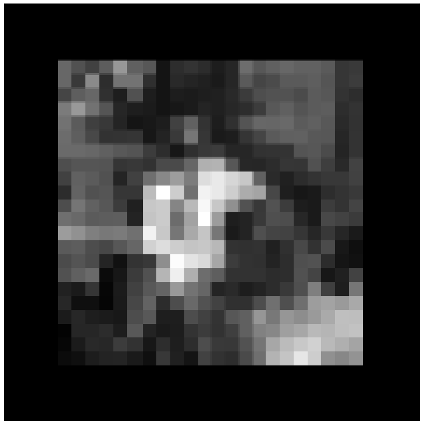
noise increases

smoothing increases

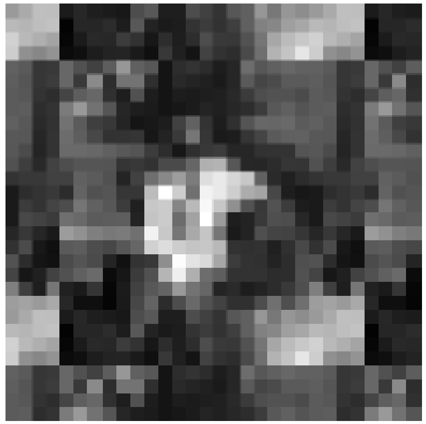


More noise needs larger sigma

Handling the Boundary (Padding)

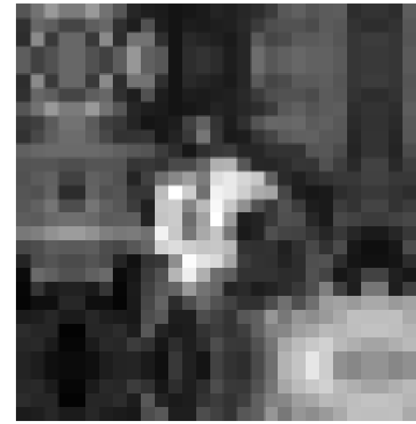


zero

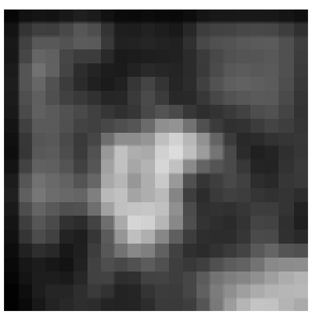


wrap

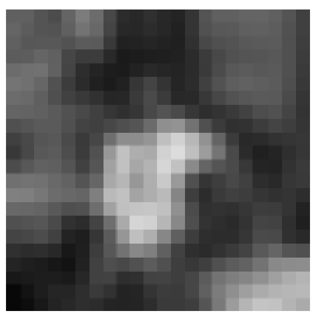
clamp



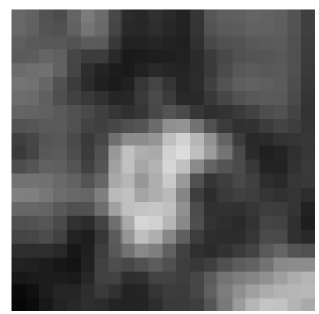
mirror



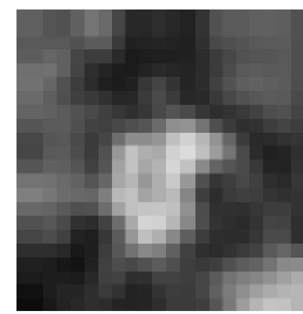
blurred zero



normalized zero



blurred clamp



blurred mirror

How to compute convolution efficiently?

- Separable filters (next)
- Fourier transformation
- Integral Image trick (see exercise)

<u>Important for later (integral Image trick):</u>

- Naive implementation would be $O(Nw)$
  where w is the number of elements in box filter
- The Box filter (and a few others) can be computed in $O(N)$

$$\frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

image

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

filter (kernel)

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

filtered image

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

Separable filters

For some filters we have: $f * h = f * (h_x * h_y)$

Where h_x, h_y are 1D filters.

Example Box filter:

$$\underbrace{\frac{1}{9} \cdot \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}}_{h_x * h_y} = \underbrace{\frac{1}{3} \cdot \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline \end{array}}_{h_x} * \underbrace{\frac{1}{3} \cdot \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}}_{h_y}$$

Now we can do two 1D convolutions:

$$f * h = f * \left(h_x * h_y\right) = (f * h_x) * h_y$$

Naïve implementation for 3x3 filter: 9N operations versus 3N+3N operations

Can any filter be made separable?

Note:

$$\overbrace{\frac{1}{9}\cdot\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&1&1\\\hline 1&1&1\\\hline\end{array}}^{h_x * h_y} = \frac{1}{3}\cdot\overbrace{\begin{array}{|c|c|c|}\hline 1&1&1\\\hline\end{array}}^{h_x} * \frac{1}{3}\cdot\overbrace{\begin{array}{|c|}\hline 1\\\hline 1\\\hline 1\\\hline\end{array}}^{h_y} = \frac{1}{3}\cdot\overbrace{\begin{array}{|c|}\hline 1\\\hline 1\\\hline 1\\\hline\end{array}}^{h_x}\cdot\frac{1}{3}\cdot\overbrace{\begin{array}{|c|c|c|}\hline 1&1&1\\\hline\end{array}}^{h_y}$$

<u>Apply SVD to the kernel matrix:</u>

$$\boldsymbol{A} = \left[\begin{array}{c|c|c} u_0 & \cdots & u_{p-1} \end{array}\right]\begin{bmatrix}\sigma_0 & & \\ & \ddots & \\ & & \sigma_{p-1}\end{bmatrix}\left[\begin{array}{c} v_0^T \\ \hline \cdots \\ \hline v_{p-1}^T \end{array}\right]$$

$$= \sum_{j=0}^{p-1} \sigma_j u_j v_j^T,$$

If all σ_i are 0 (apart from σ_0) then it is separable.

Example of separable filters

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

(a) box, $K = 5$

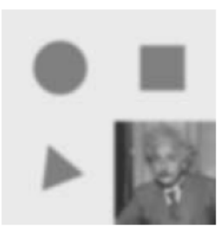
$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

(b) bilinear

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

$$\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$



(c) “Gaussian”

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array}$$

(d) Sobel

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(e) corner