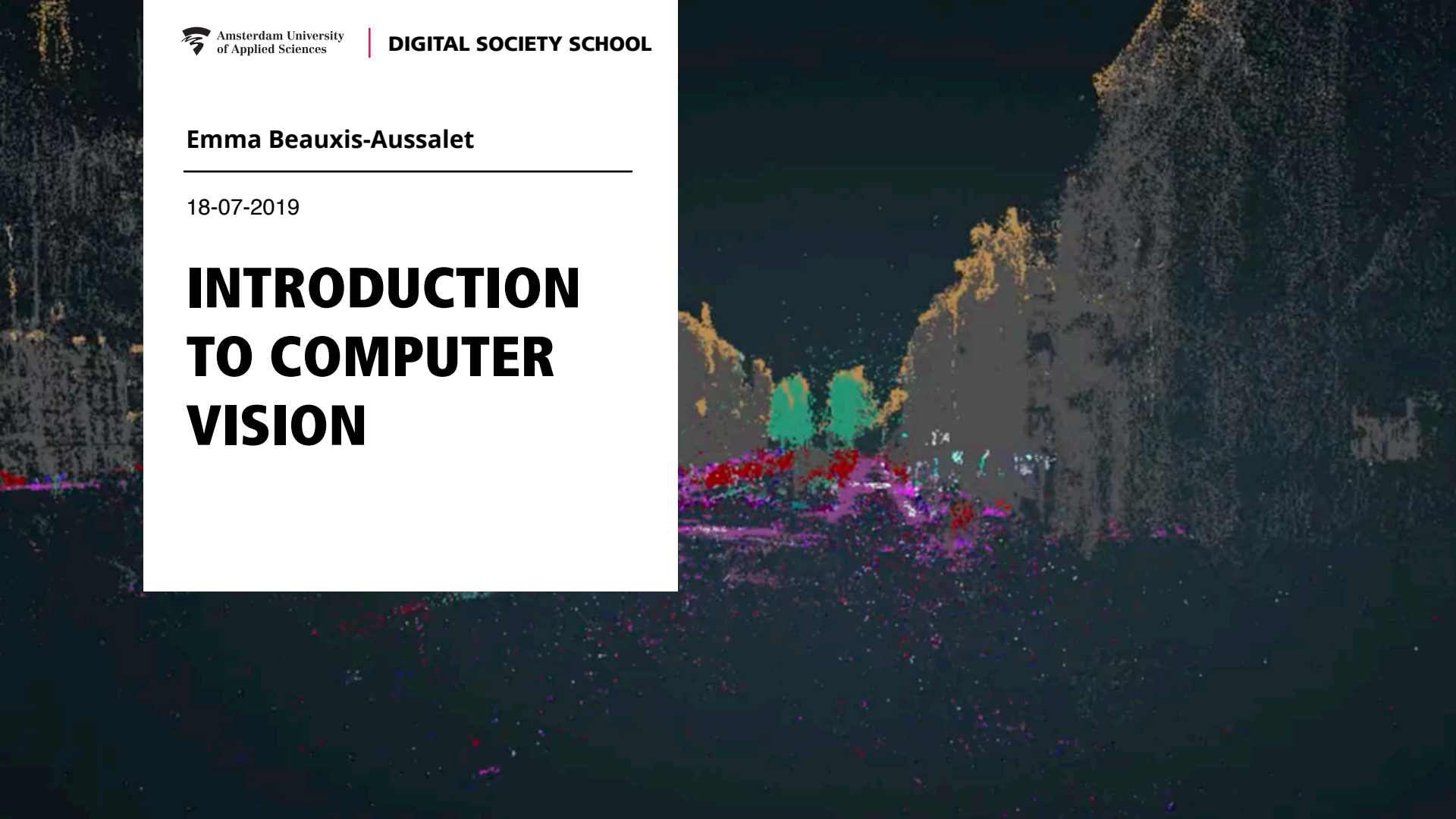
Amsterdam University of Applied Sciences | DIGITAL SOCIETY SCHOOL

Emma Beauxis-Aussalet

18-07-2019

INTRODUCTION TO COMPUTER VISION



PROGRAMME week1
15-19 July 2019

15 Monday
09:00 - 10:00 Welcome to Digital Society School + Introduction to Summer School
10:00 - 12:00 Teamwork
12:00 - 13:00 *Lunch break*
13:00 - 17:00 Design Thinking + SDG workshop

16 Tuesday
09:30 - 11:30 Datasets
11:30 - 12:30 *Lunch time*
12:30 - 15:30 Data gathering through the city
15:30 - 17:00 Data check-in

17 Wednesday
09:30 - 12:30 Introduction to AI
12:30 - 13:30 *Lunch break*
13:30 - 17:00 Data gathering through the city

18 Thursday
09:30 - 12:00 Classification error and biases
12:00 - 13:00 *Lunch break*
13:00 - 17:00 Computer Vision Theory + Python Assistance

19 Friday
09:30 - 12:30 Network visualisation with Gephi
12:30 - 13:30 *Lunch break*
13:30 - 16:00 Retrospective and peer pitch
16:00 Beers at IJ Brewery

PROGRAMME week2
22-25 July 2019

22 Monday
09:30 - 11:00 Raw Graphs Data visualisation
11:00 - 12:00 Illustrator basics
12:00 - 13:00 *Lunch break*
13:00 - 14:00 Visualising images
14:30 - 16:00 Dealing with Clarifi and Google APIs

23 Tuesday
09:30 - 11:30 R for beginners
11:30 - 12:30 R for beginners (experimentation)
12:30 - 13:30 *Lunch break*
13:30 - 17:00 Take more sample to improve classifier - Data Gathering

24 Wednesday
09:30 - 12:30 Free space for working + Preparing Presentation
12:30 - 13:30 *Lunch break*
13:30 - 17:00 Free space for working + Preparing Presentation

25 Thursday
09:30 - 12:00 Free space for working + Preparing Presentation
12:00 - 13:00 *Lunch break*
13:00 - 15:00 Presentations + Feedbacks
15:00 - 17:00 Beers!

INTRODUCTION TO CV

WHAT IS COMPUTER VISION?

HAVE YOU EVER USED IT?

Laptop: Biometrics auto-login (face recognition, 3D), OCR

Smartphones: QR codes, computational photography (Android Lens Blur, iPhone Portrait Mode), panorama construction (Google Photo Spheres), face detection, expression detection (smile), Snapchat filters (face tracking), Google Tango (3D reconstruction), Night Sight (Pixel)

Web: Image search, Google photos (face recognition, object recognition, scene recognition, geolocalization from vision), Facebook (image captioning), Google maps aerial imaging (image stitching), YouTube (content categorization)

VR/AR: Outside-in tracking (HTC VIVE), inside out tracking (simultaneous localization and mapping, HoloLens), object occlusion (dense depth estimation)

Motion: Kinect, full body tracking of skeleton, gesture recognition, virtual try-on

Medical imaging: CAT / MRI reconstruction, assisted diagnosis, automatic pathology, connectomics, endoscopic surgery

Industry: Vision-based robotics (marker-based), machine-assisted router (jig), automated post, ANPR (number plates), surveillance, drones, shopping

Transportation: Assisted driving (everything), face tracking/iris dilation for drunkeness, drowsiness, automated distribution (all modes)

Media: Visual effects for film, TV (reconstruction), virtual sports replay (reconstruction), semantics-based auto edits (reconstruction, recognition)

Source: https://cs.brown.edu/courses/csci1430/lectures/2019Spring_01_Introduction.pdf

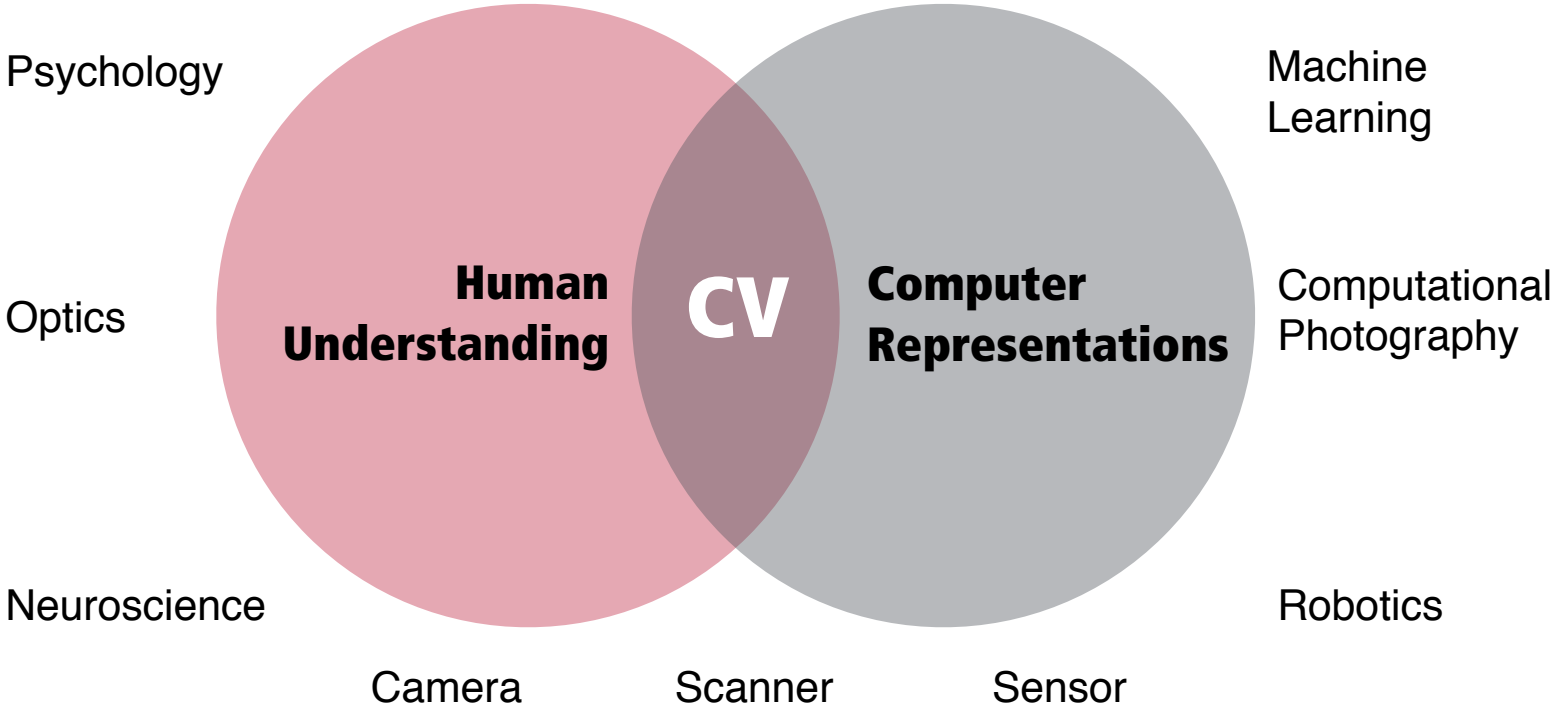
Most probably…

DEFINITION

“Computer vision is concerned with
the automatic extraction, analysis & understanding of useful information
from a single image or a sequence of images.
It involves the development of **a theoretical and algorithmic basis**
to achieve automatic visual understanding.”

Attributed to The British Machine Vision Association and Society for Pattern Recognition

MULTIDISCIPLINARY



SUB-DOMAINS

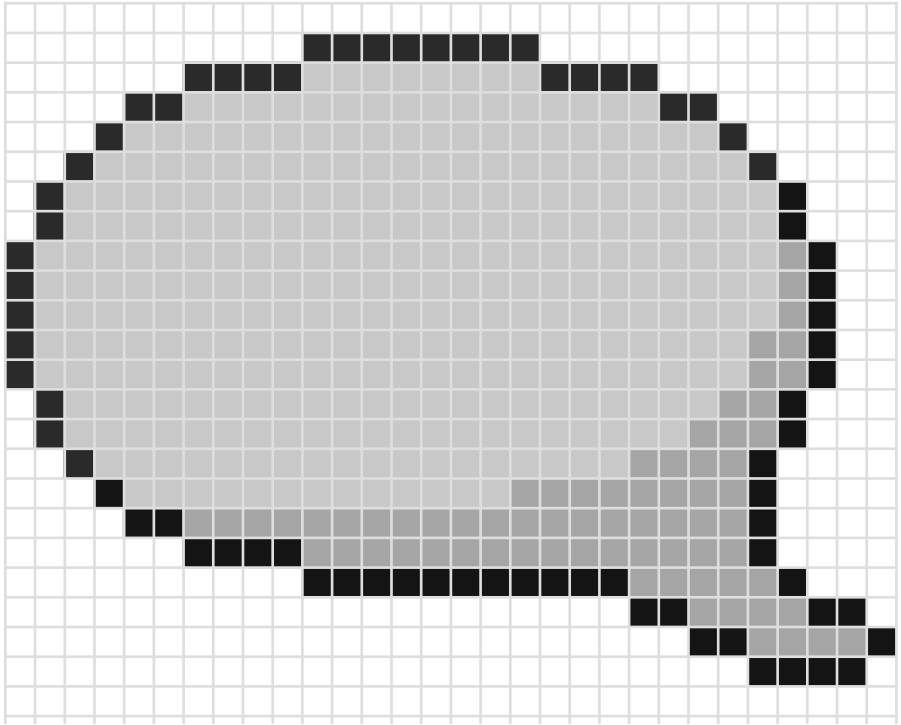
- **Image (pre-)processing** deals with the low-level features of images.
- **Feature detection** provides refined representation of images.
- **Segmentation** detects the parts of images.
- **3D reconstruction** creates 3D models of objects from 2D images.
- **Object recognition** labels what appears in images.
- **Motion analysis** deals with moving objects in videos.

INTRODUCTION TO CV

IMAGE (PRE-)PROCESSING

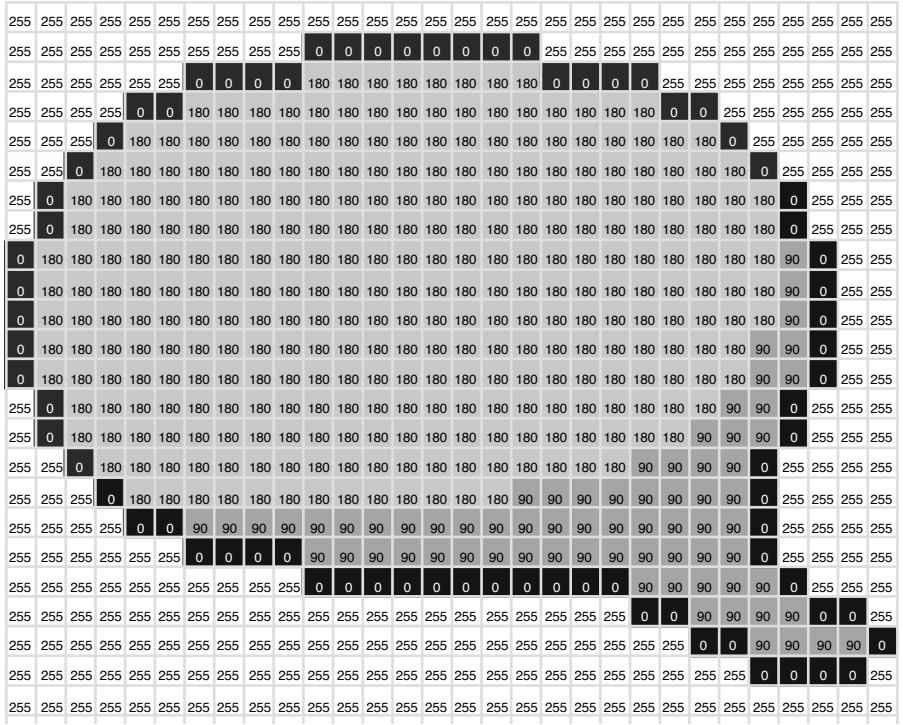
PIXEL

(picture element)



PIXEL

(picture element)



PIXEL

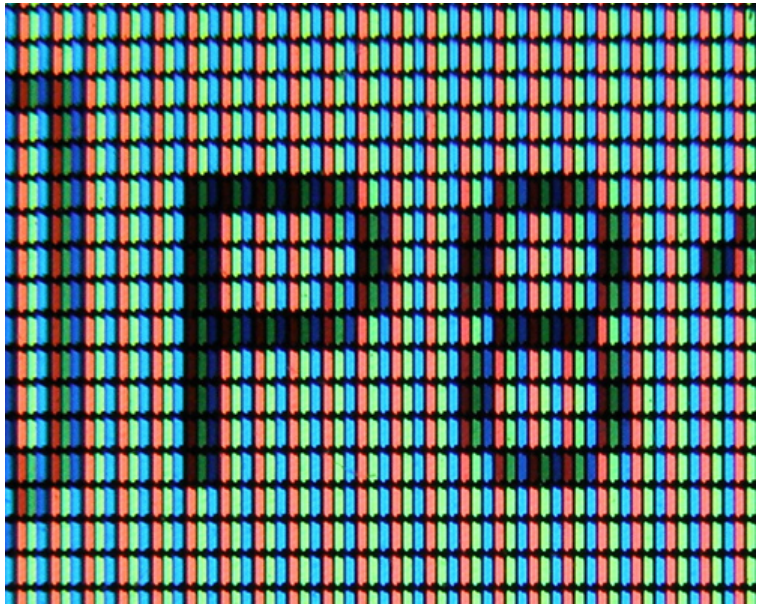
(picture element)

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	0	0	0	0	180	180	180	180	180	180	180	180	0	0	0	0	255	255	255	255	255	255	255	255
255	255	255	255	0	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	0	0	255	255	255	255	255	255
255	255	255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	0	255	255	255	255	255
255	255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	0	255	255	255	255
255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	0	255	255	255
255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	0	255	255	255
0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	0	255	255
0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	0	255	255
0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	0	255	255
0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	0	255	255
0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	0	255	255
255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	0	255	255	255
255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	90	0	255	255	255
255	255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	90	90	0	255	255	255	255
255	255	255	0	180	180	180	180	180	180	180	180	180	180	180	180	180	90	90	90	90	90	90	90	90	0	255	255	255	255
255	255	255	255	0	0	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	0	255	255	255	255
255	255	255	255	255	255	0	0	0	0	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	0	255	255	255	255
255	255	255	255	255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	0	0	90	90	90	90	90	0	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	90	90	90	90	0	0	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	90	90	90	90	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	0	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

PIXEL

(picture element)

3 colors…



PIXEL

(picture element)

3 colors…

…3 channels

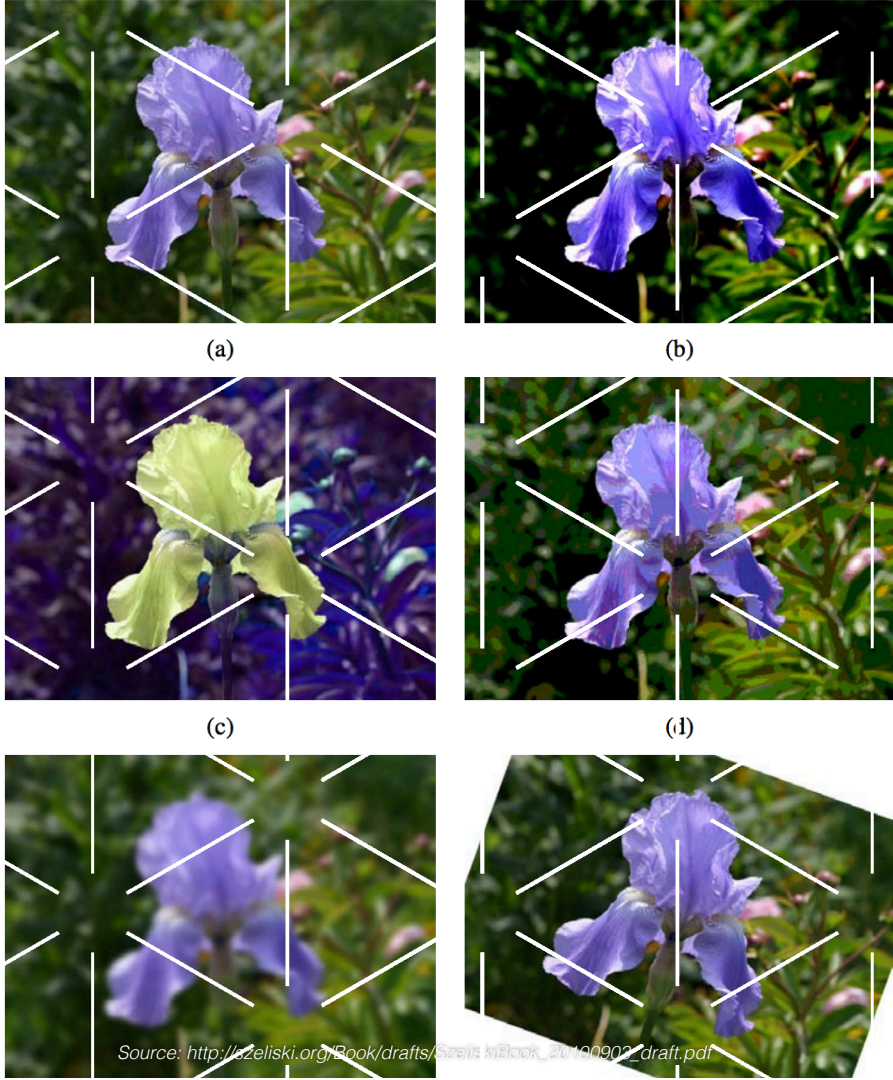
Fig. 1.4 Original RGB colour image `Fountain` (*upper left*), showing a square in Guanajuato, and its decomposition into the three contributing channels: *Red* (*upper right*), *Green* (*lower left*), and *Blue* (*lower right*). For example, *red* is shown with high intensity in the *red channel*, but in low intensity in the *green* and *blue channel*

Source: Reinhard Klette, Concise Computer Vision, Springer

(PRE-)PROCESSING

Image pre-processing is the direct manipulation of pixel values.
A variety of operations are possible:

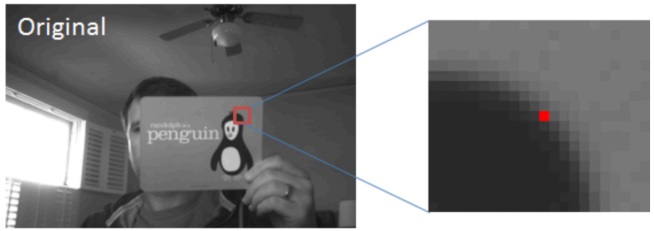
- **BRIGHTNESS, CONTRAST**
- **HISTOGRAM EQUALISATION**
- **COLOR NORMALIZATION**
- **FILTERING**

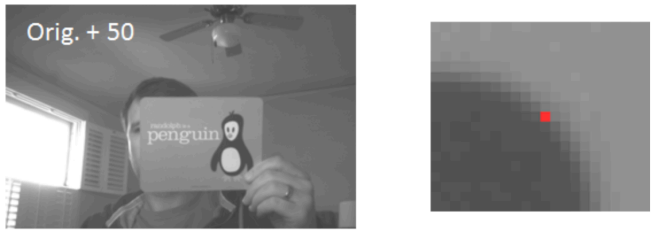


Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100902_draft.pdf

BRIGHTNESS, CONTRAST

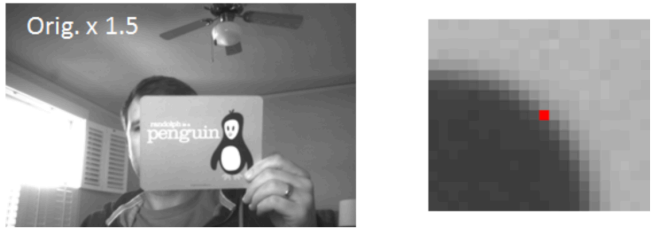
Add X to
all pixel values…





…to increase
brightness.

Multiply X with
all pixel values…



…to increase
contrast.

Source: http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/

BRIGHTNESS, CONTRAST

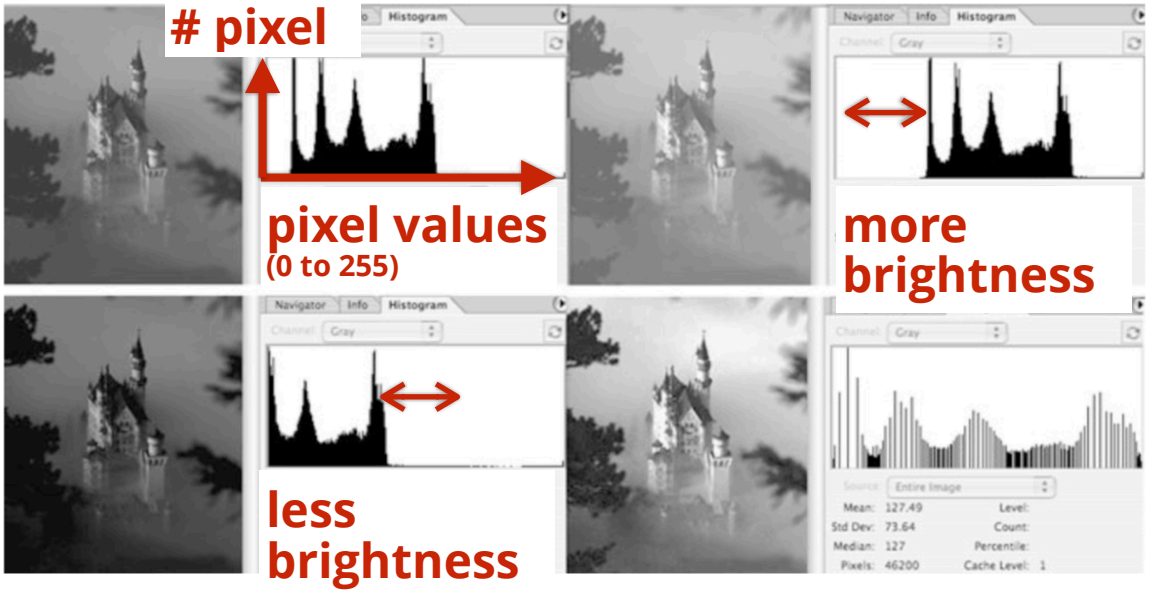
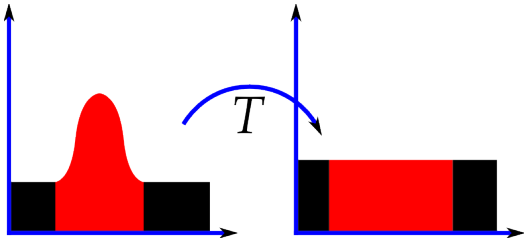


Fig. 1.5 Histograms for the 200 × 231 image `Neuschwanstein`. *Upper left*: Original image. *Upper right*: Brighter version. *Lower left*: Darker version. *Lower right*: After histogram equalization (will be defined later)

Source: Reinhard Klette, Concise Computer Vision, Springer

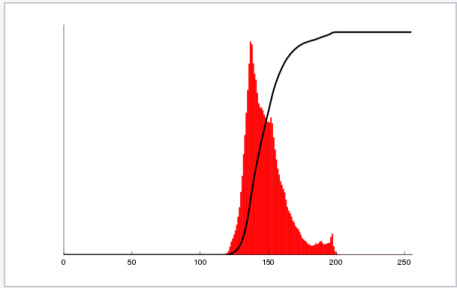
HISTOGRAM EQUALIZATION



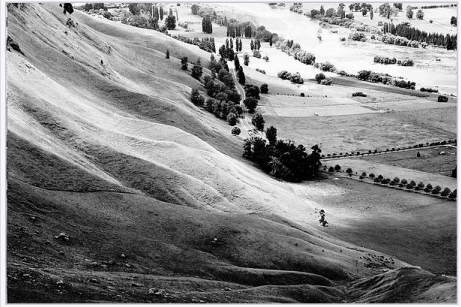
Source: Reinhard Klette, Concise Computer Vision, Springer

HISTOGRAM EQUALIZATION

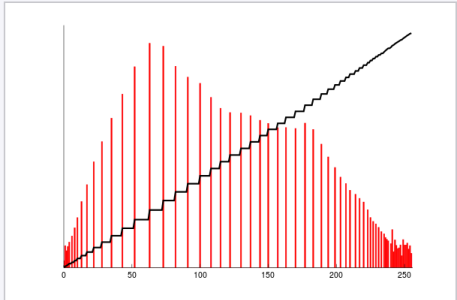
Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)



After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Source: https://en.wikipedia.org/wiki/Histogram_equalization

COLOR NORMALISATION

(a) RGB (b) R (c) G (d) B

(e) rgb (f) r (g) g (h) b

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}$$

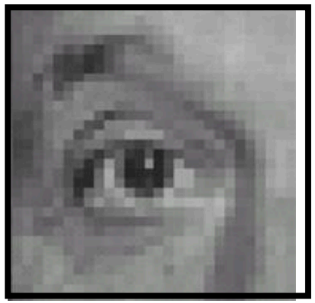
Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

FILTERING

Linear filters combine each pixel with its neighbours.

Original

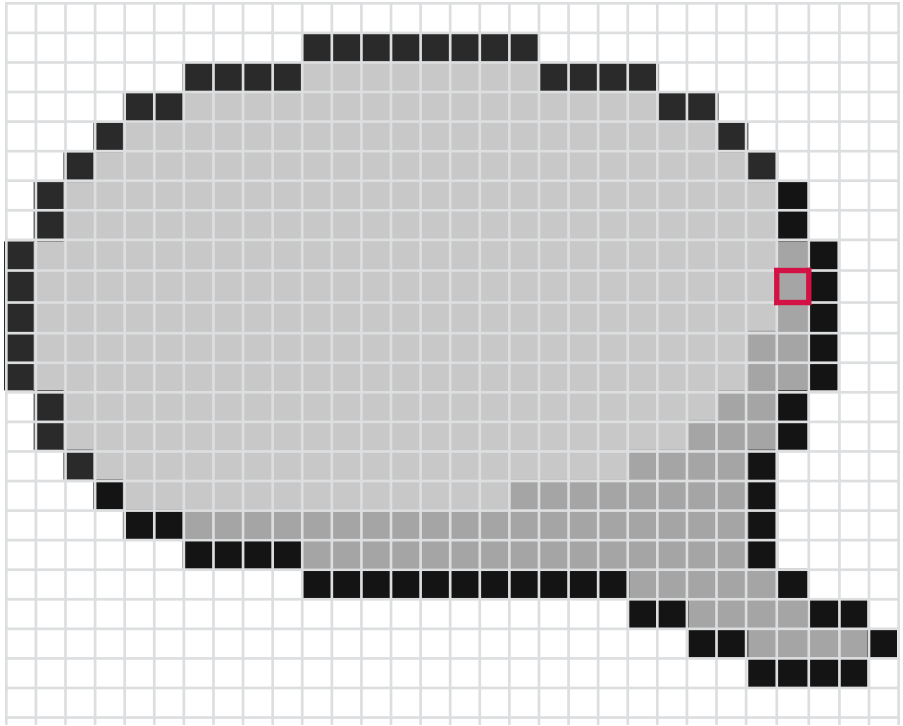
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

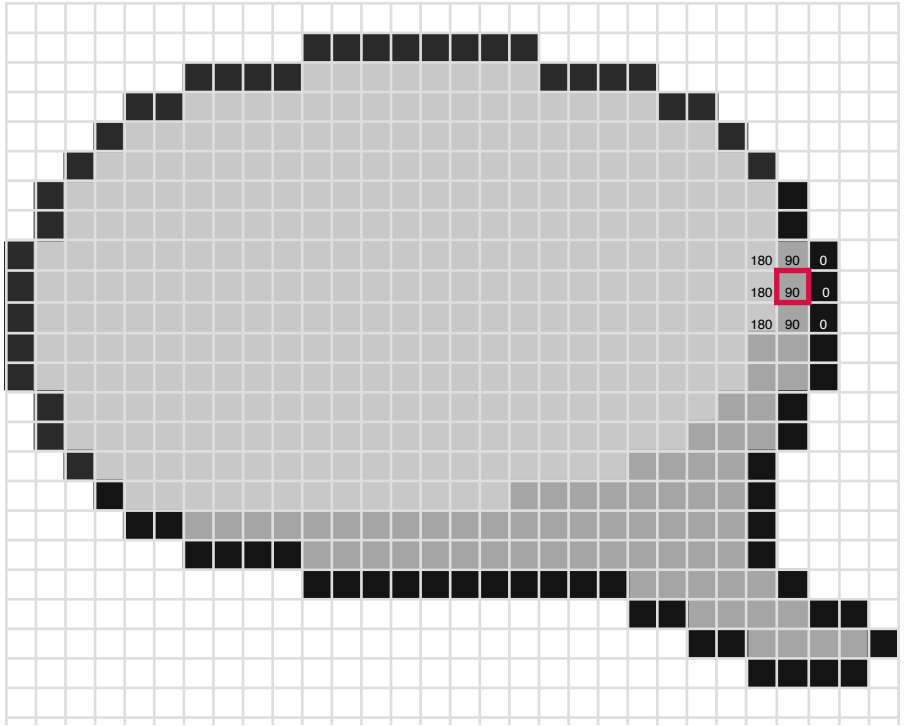
Source: https://cs.brown.edu/courses/csci1430/lectures/2019Spring_01_Introduction.pdf

FILTERING



0	0	0
0	0	1
0	0	0

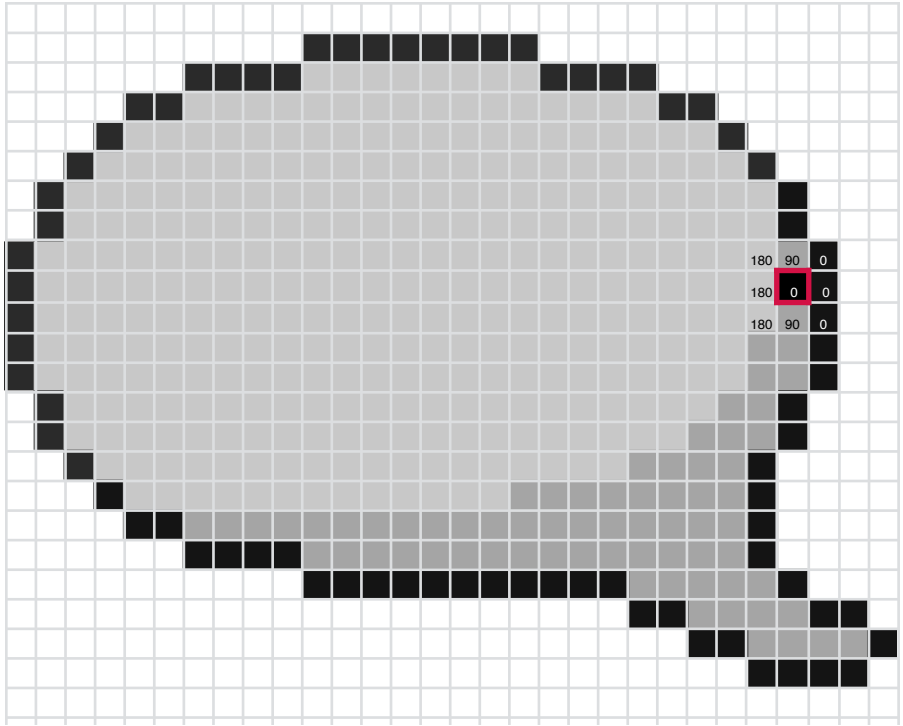
FILTERING



0	0	0
0	0	1
0	0	0

0 x 180 + 0 x 90 + 0 x 0
+ 0 x 180 + 0 x 90 + 1 x 0
+ 0 x 180 + 0 x 90 + 0 x 0
= 0

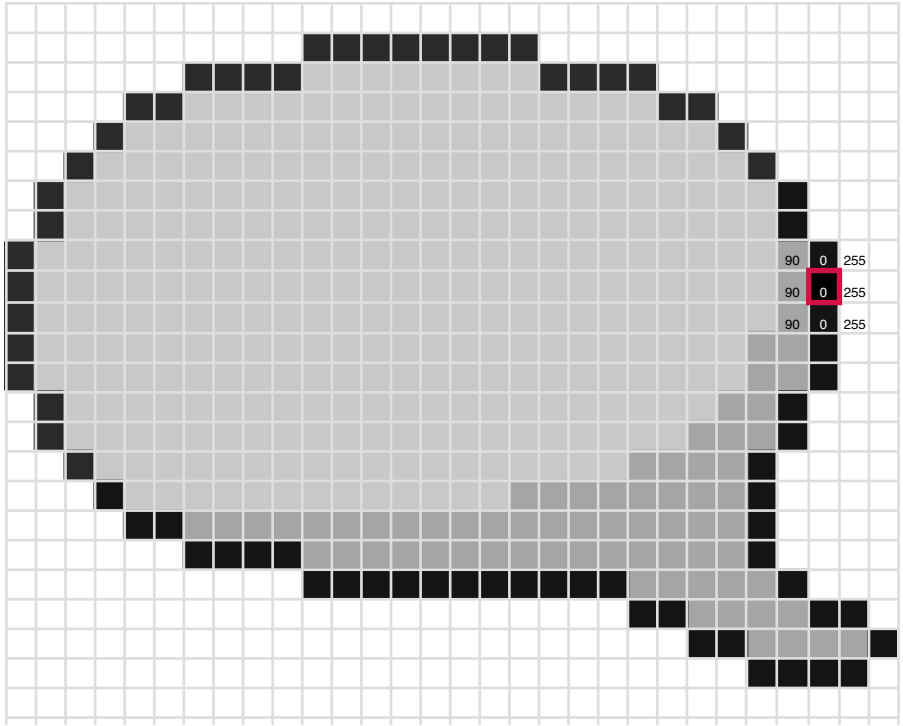
FILTERING



0	0	0
0	0	1
0	0	0

0 x 180 + 0 x 90 + 0 x 0
+ 0 x 180 + 0 x 90 + 1 x 0
+ 0 x 180 + 0 x 90 + 0 x 0
= 0

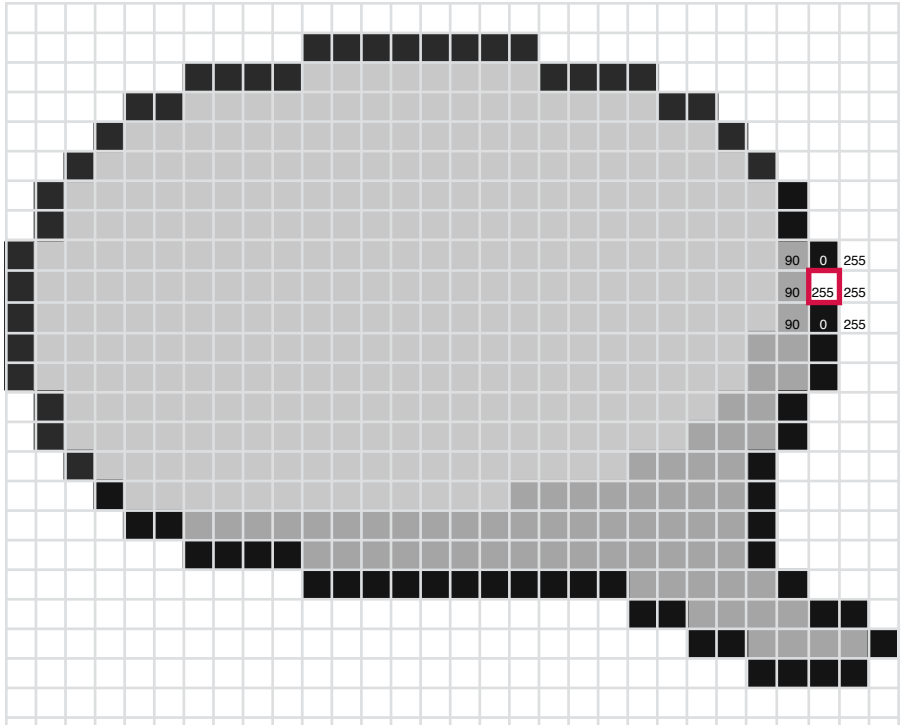
FILTERING



0	0	0
0	0	1
0	0	0

0 x 90 + 0 x 0 + 0 x 255
+ 0 x 90 + 0 x 0 + 1 x 255
+ 0 x 90 + 0 x 0 + 0 x 255
= 255

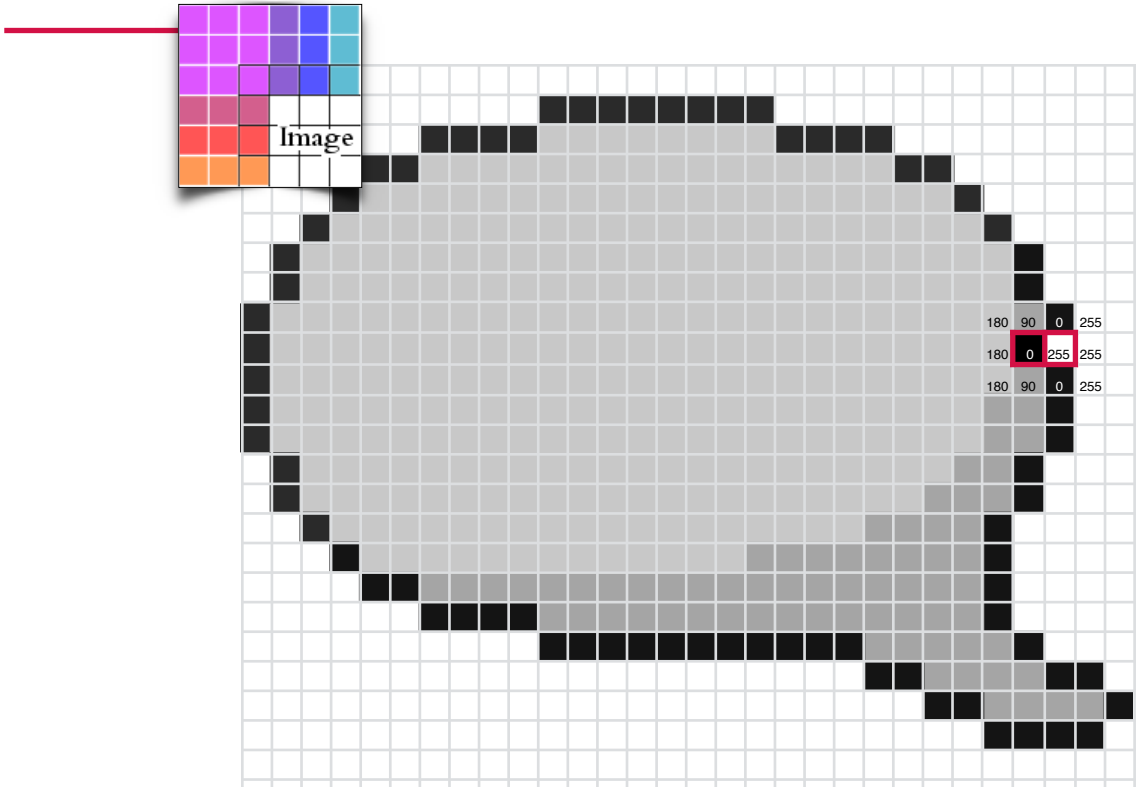
FILTERING



0	0	0
0	0	1
0	0	0

0 x 90 + 0 x 0 + 0 x 255
+ 0 x 90 + 0 x 0 + 1 x 255
+ 0 x 90 + 0 x 0 + 0 x 255
= 255

FILTERING



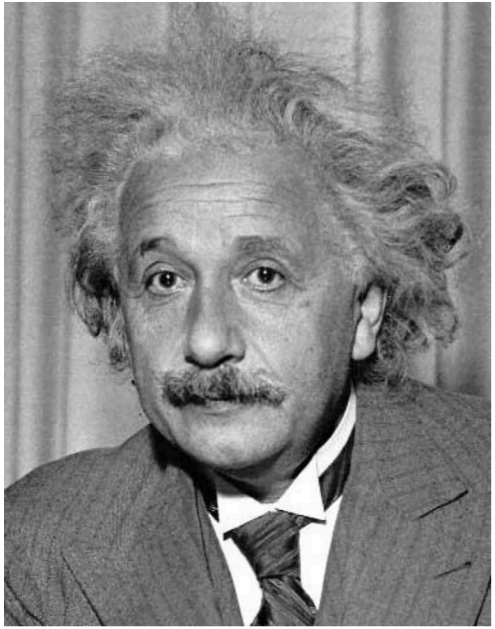
0	0	0
0	0	1
0	0	0

FILTERING

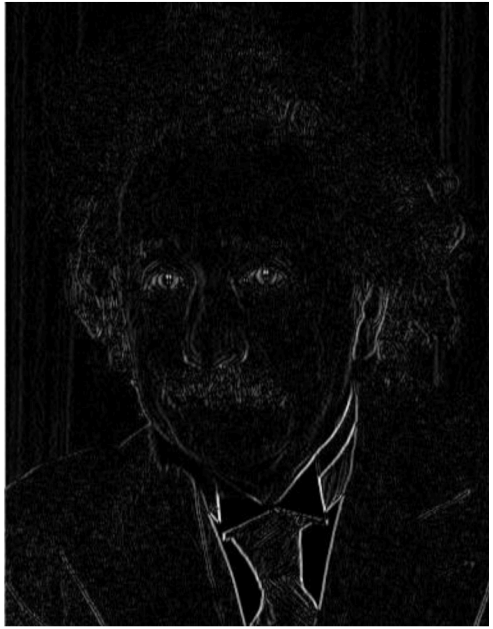
What filter would leave the image unchanged?

0	0	0
0	1	0
0	0	0

FILTERING



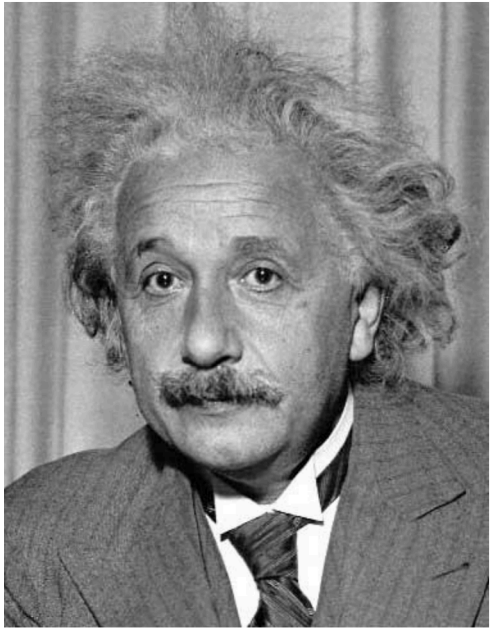
1	0	-1
2	0	-2
1	0	-1



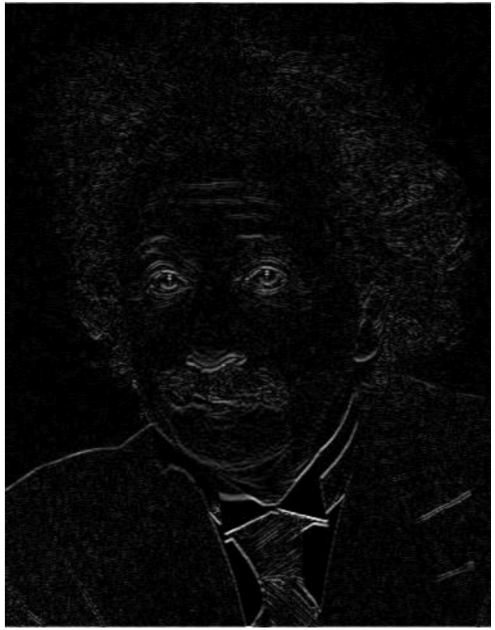
Vertical Edge

Source: https://cs.brown.edu/courses/csci1430/lectures/2019Spring_01_Introduction.pdf

FILTERING



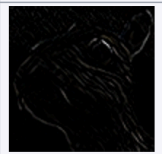
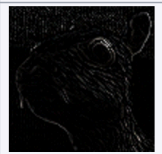
1	2	1
0	0	0
-1	-2	-1

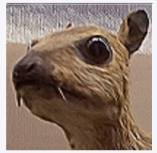
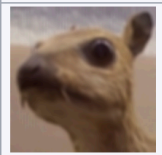
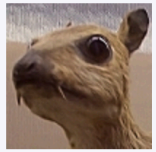


Horizontal Edge

Source: https://cs.brown.edu/courses/csci1430/lectures/2019Spring_01_Introduction.pdf

FILTERING

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Operation	Kernel ω	Image result g(x,y)
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

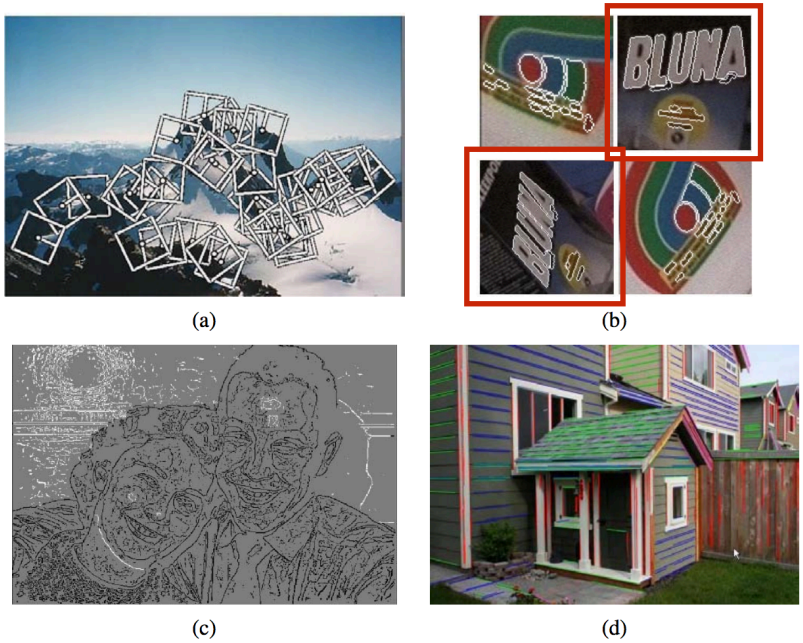
Source: https://en.wikipedia.org/wiki/Kernel_%28image_processing%29

INTRODUCTION TO CV

FEATURE DETECTION

INTEREST POINTS

Interest points include edges, corners, blobs, patches, ridges, textures.



Primarily to match images

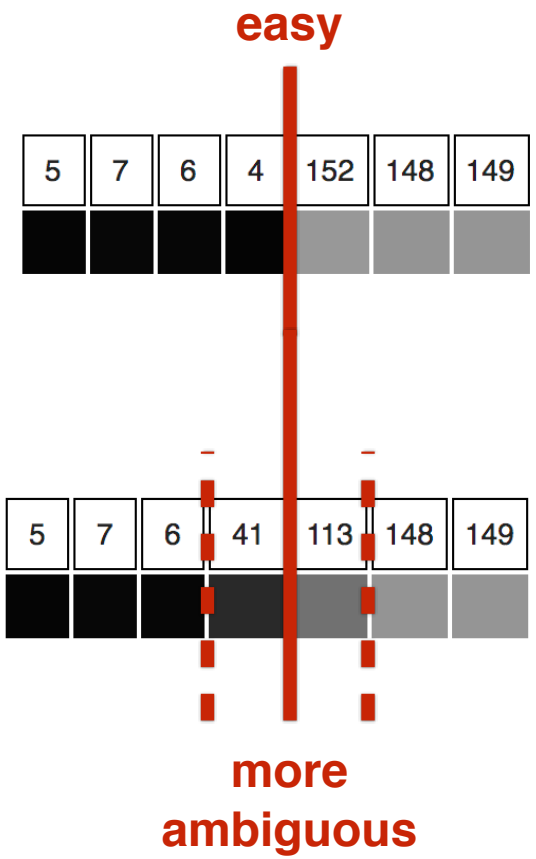
Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

EDGE

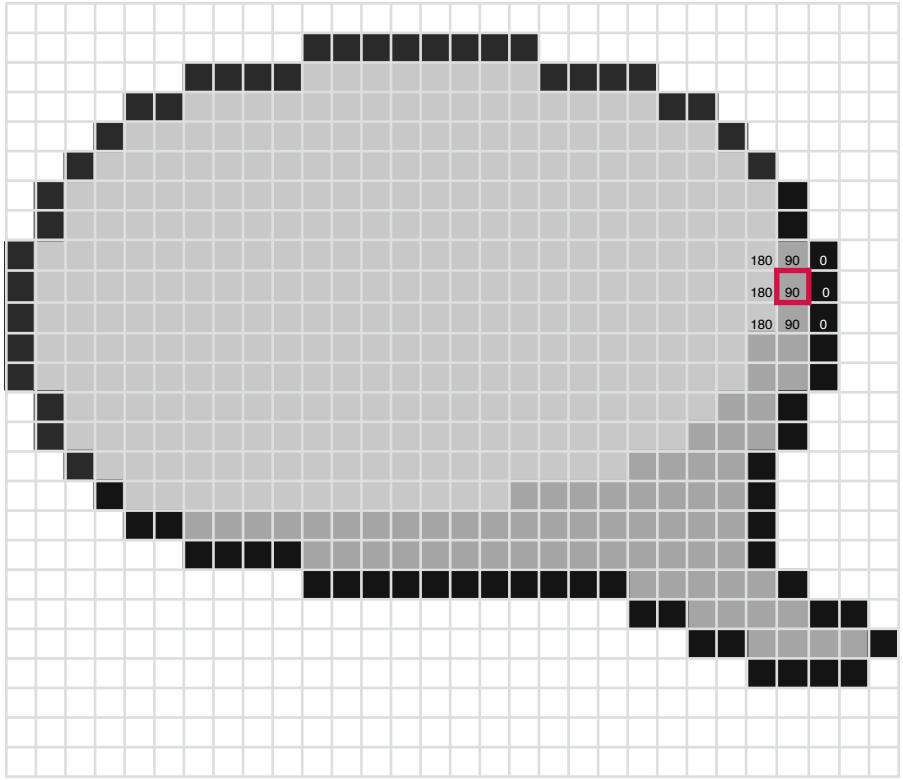
Edge detection identify points where brightness changes sharply. (formally called discontinuities)

Source: https://en.wikipedia.org/wiki/Edge_detection

EDGE

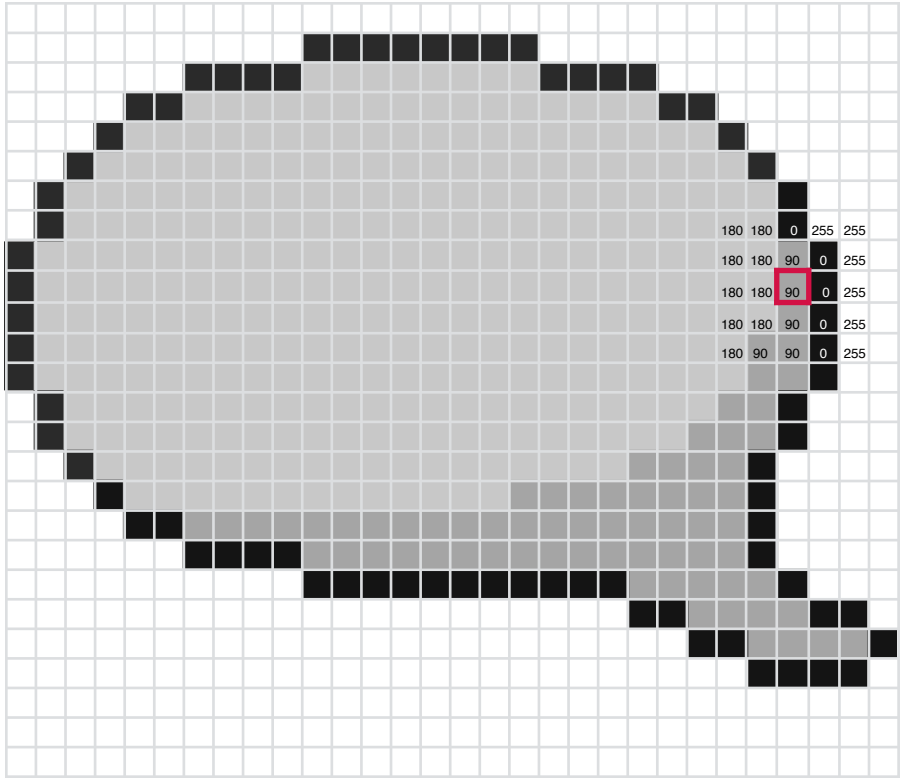


DoG (DIFFERENCE OF GAUSSIAN)



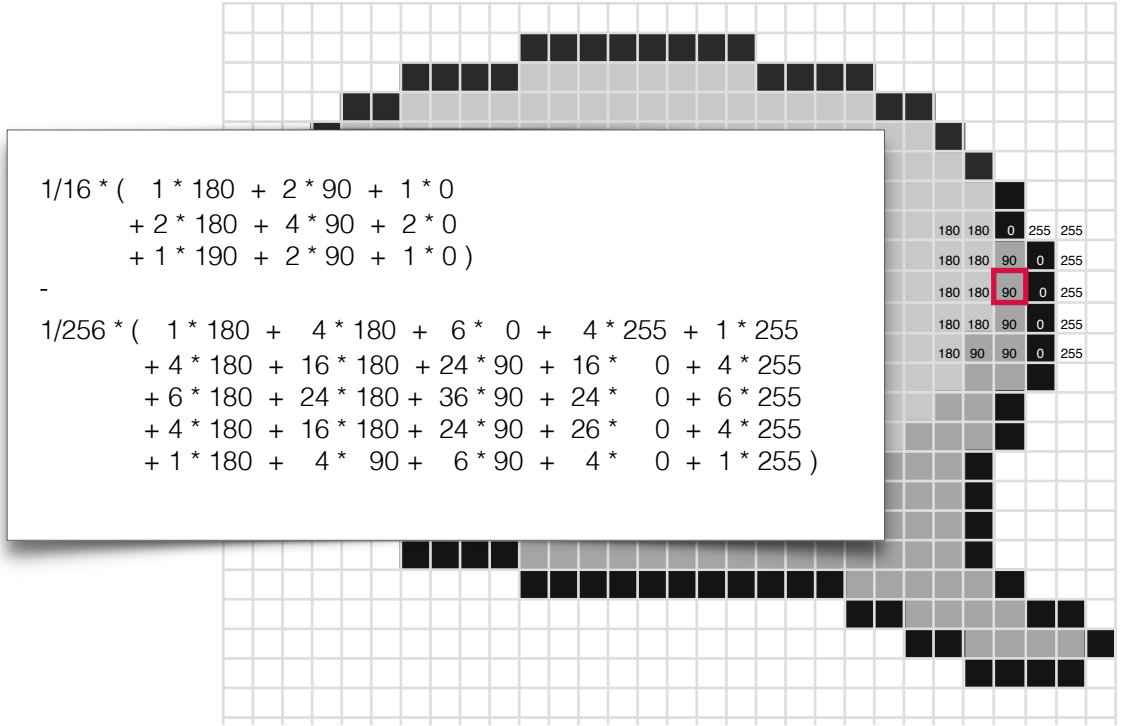
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

DoG (DIFFERENCE OF GAUSSIAN)



$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

DoG (DIFFERENCE OF GAUSSIAN)



$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

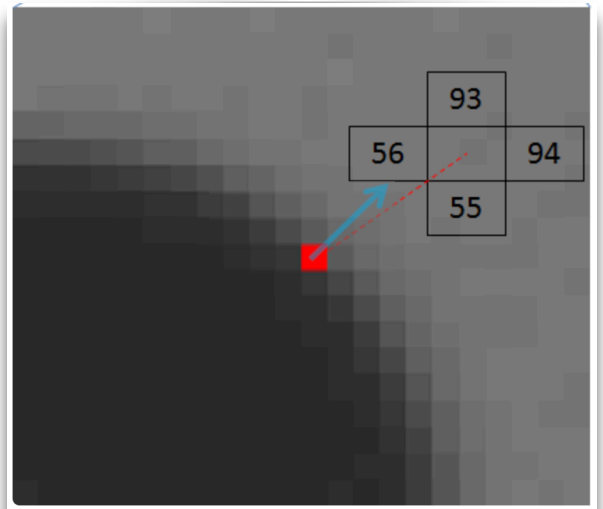
$$\begin{aligned} &1/16 * (\ 1 * 180 + 2 * 90 + 1 * 0 \\ &\quad + 2 * 180 + 4 * 90 + 2 * 0 \\ &\quad + 1 * 190 + 2 * 90 + 1 * 0\ ) \\ &- \\ &1/256 * (\ 1 * 180 + 4 * 180 + 6 * 0 + 4 * 255 + 1 * 255 \\ &\quad + 4 * 180 + 16 * 180 + 24 * 90 + 16 * 0 + 4 * 255 \\ &\quad + 6 * 180 + 24 * 180 + 36 * 90 + 24 * 0 + 6 * 255 \\ &\quad + 4 * 180 + 16 * 180 + 24 * 90 + 26 * 0 + 4 * 255 \\ &\quad + 1 * 180 + 4 * 90 + 6 * 90 + 4 * 0 + 1 * 255\ ) \end{aligned}$$

DoG (DIFFERENCE OF GAUSSIAN)

Source: https://en.wikipedia.org/wiki/Difference_of_Gaussians

HOG (HISTOGRAM OF ORIENTED GRADIENTS)

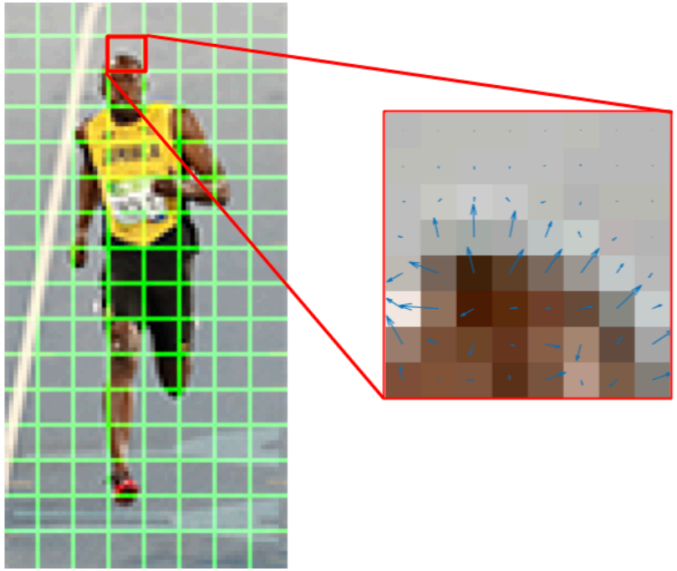
Compute the gradient vector of each pixel…



Source: http://mccormickml.com/2013/05/07/gradient-vectors/

HOG (HISTOGRAM OF ORIENTED GRADIENTS)

Compute the gradient vector of each pixel…
and the gradient histograms of patches.



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

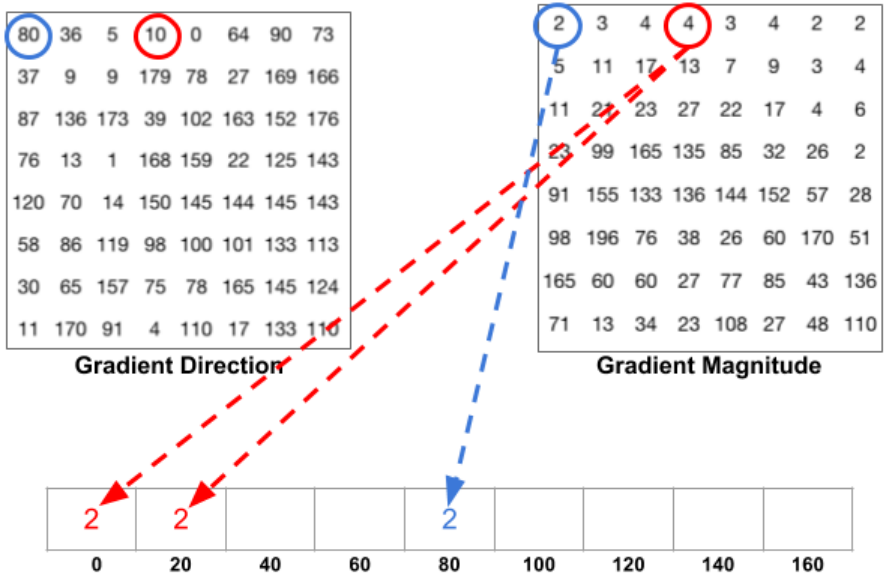
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Source: https://www.learnopencv.com/histogram-of-oriented-gradients/

HOG (HISTOGRAM OF ORIENTED GRADIENTS)

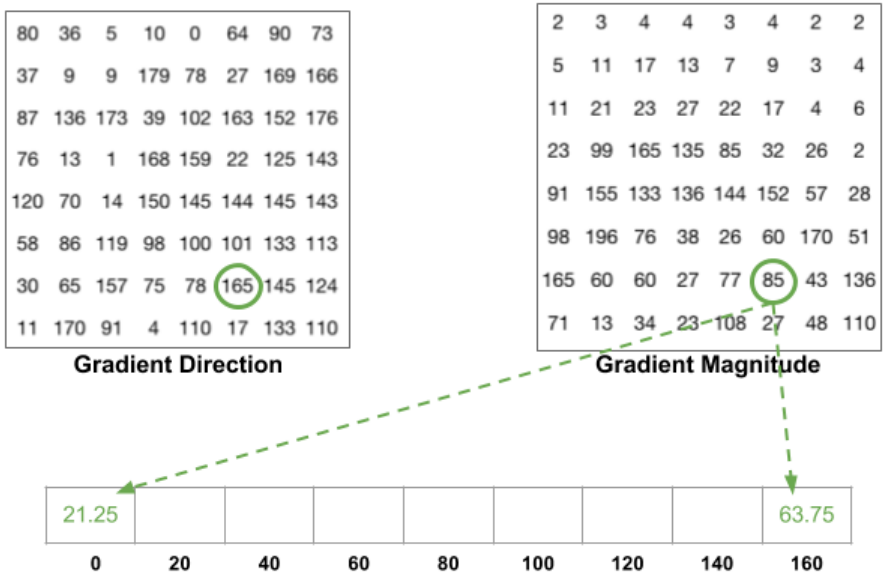
Compute the gradient vector of each pixel…
and the gradient histograms of patches.



Source: https://www.learnopencv.com/histogram-of-oriented-gradients/

HOG (HISTOGRAM OF ORIENTED GRADIENTS)

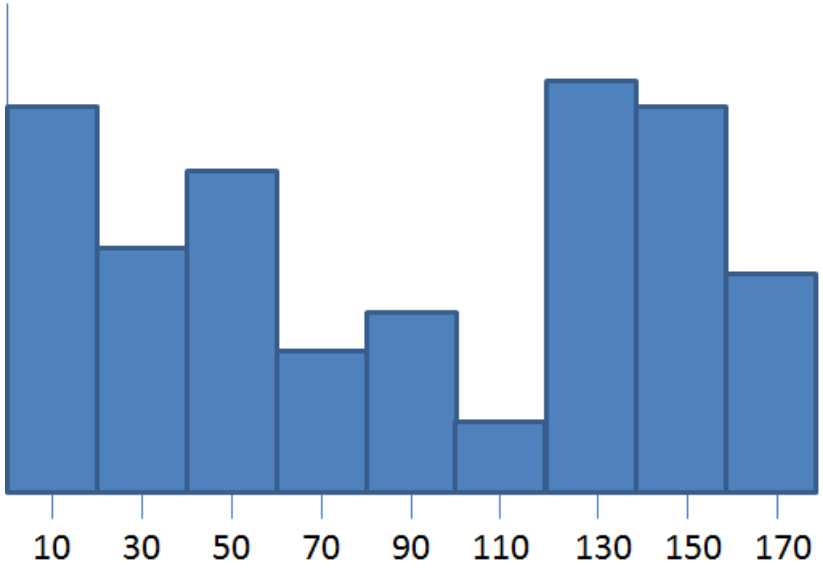
Compute the gradient vector of each pixel…
and the gradient histograms of patches.



Source: https://www.learnopencv.com/histogram-of-oriented-gradients/

HOG (HISTOGRAM OF ORIENTED GRADIENTS)

Compute the gradient vector of each pixel…
and the gradient histograms of patches.



HOG (HISTOGRAM OF ORIENTED GRADIENTS)

Compute the gradient vector of each pixel…
and the gradient histograms of patches.

Fig. 10.8 The length of vectors in nine different directions in each cell represents the accumulated magnitude of gradient vectors into one of those nine directions

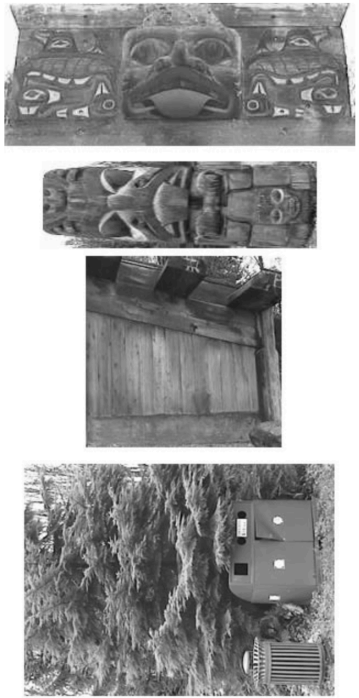
Source: Reinhard Klette, Concise Computer Vision, Springer

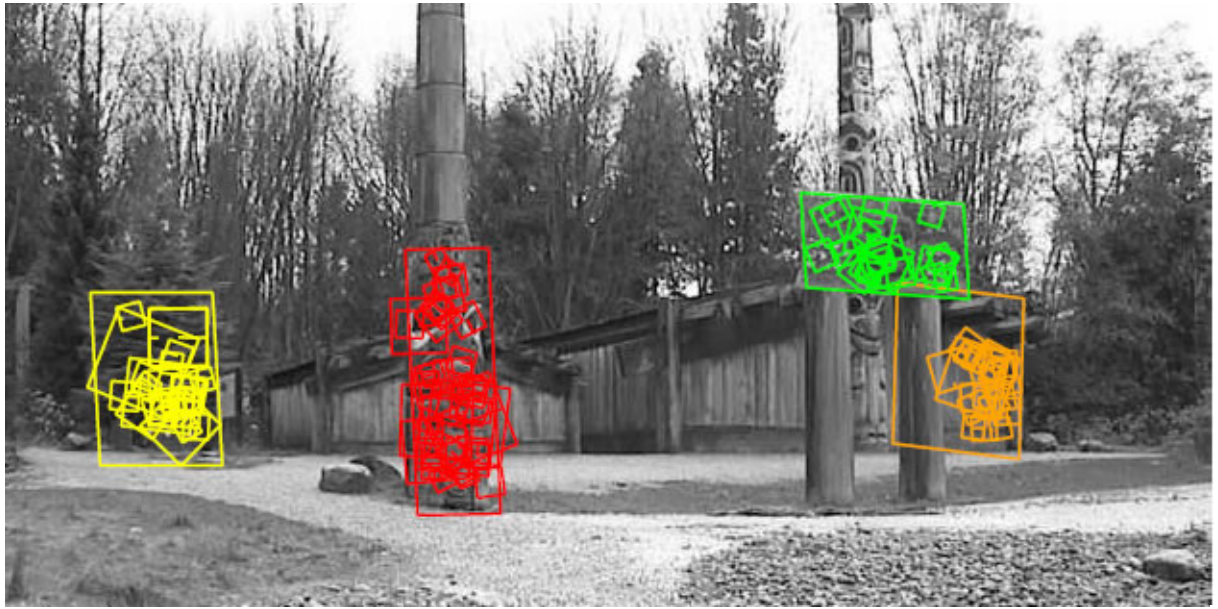
SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

Scale-Invariant means that SIFT description of interest points do not change with:

- Scale
- Rotation
- Illumination
- Viewpoint (affine distortions)

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)





Source: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

The algorithms is quite elaborate, and combines different techniques.

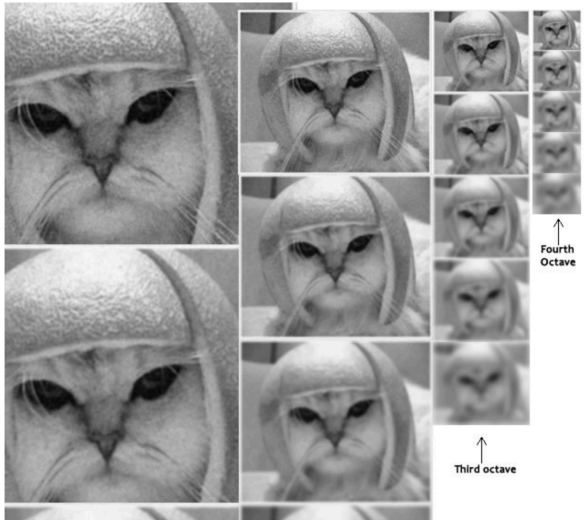
1. **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
3. **Finding keypoints** With the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2
4. **Get rid of bad key points** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.
5. **Assigning an orientation to the keypoints** An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.
6. **Generate SIFT features** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board). That was an overview of the entire algorithm. Over the next few days, I'll go through each step in detail. Finally, I'll show you how to **implement SIFT in OpenCV**!

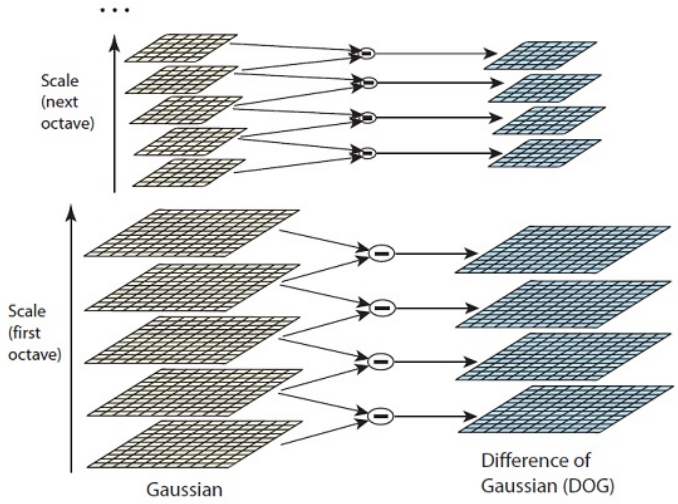
Source: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

First it uses DoG with different scales and blurs…

1. **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created

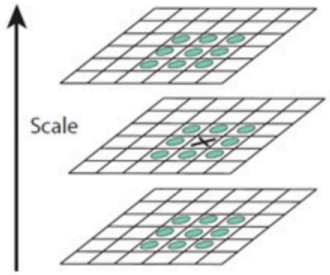




SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

Then it finds maxima & minima in DoG results across scale…

1. **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
3. **Finding keypoints** With the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2



X marks the current pixel. The green circles mark the neighbours.

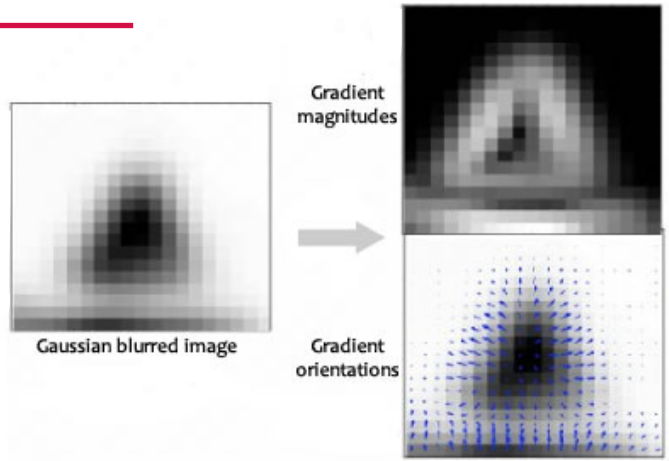
SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

Then it finds maxima & minima in DoG results across scale…

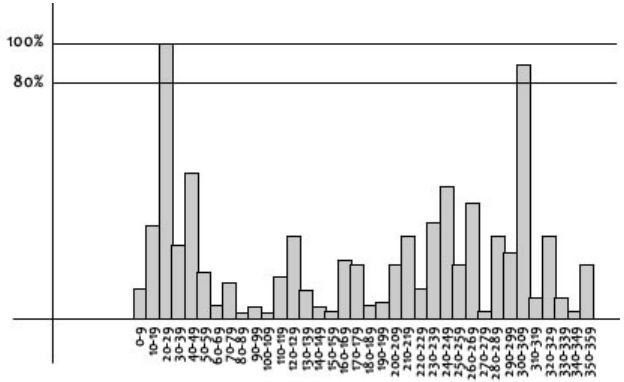
1. **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
3. **Finding keypoints** With the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2
4. **Get rid of bad key points** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.

…and elicits the most interesting keypoints.

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)



Find the bin with max #pixels,
select all bins with #pixels > 80% max #pixels.



5. **Assigning an orientation to the keypoints** An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

Then it computes HOG…

…and identify interest points for each peak orientation.

Peak orientations are used to compare rotated images.

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

Finally, HOGs are encoded around the keypoints.

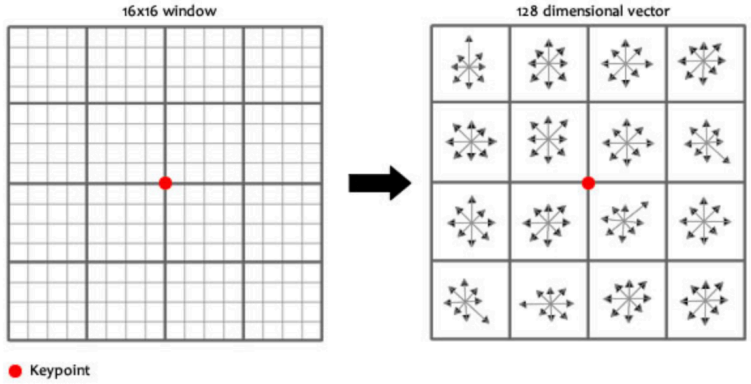
1. **Constructing a scale space** This is the init
image to ensure scale invariance. This is don

+ Subtract key orientation
(for orientation invariance)

+ Normalize gradient magnitude
(for illumination invariance)

it rotation invariant.

To do this, a 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows.



6. **Generate SIFT features** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board). That was an overview of the entire algorithm. Over the next few days, I'll go through each step in detail. Finally, I'll show you how to **implement SIFT in OpenCV**!

Source: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

SIFT (SCALE-INVARIANT FEATURE TRANSFORM)

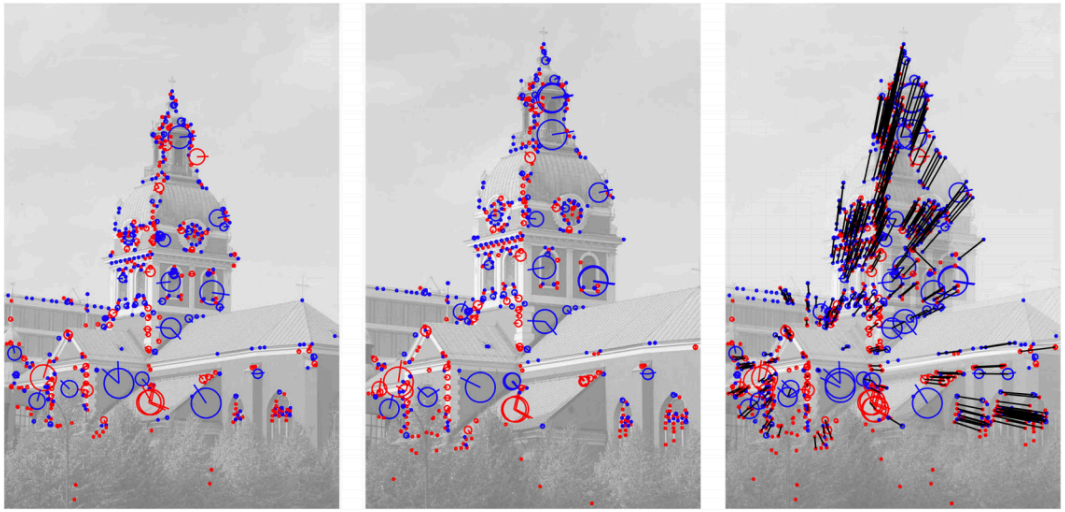


Fig. 1 Illustration of image matching using Laplacian interest points with locally adapted SIFT descriptors computed around each interest point. (*left*) and (*middle*) Two images of a building in downtown Stockholm taken from different 3-D positions with the interest points shown as *circles* overlaid on a bright copy of the original image with the size of each *circle* proportional to the locally adapted scale estimate and with the orientation estimate used for aligning the orientation of the SIFT descriptor drawn as an *angular line* from the center of the interest point. The colour of the *circle* indicates the polarity of the interest point, with *red* corresponding to *bright blobs* and *blue* corresponding to *dark blobs*. (*right*) Matching relations between the interest points drawn as *black lines* on *top* of a superposition of the original grey-level images

Source: Tony Lindeberg. Image Matching Using Generalized Scale-Space Interest Points. https://people.kth.se/~tony/papers/Lin15-JMIV.pdf

INTRODUCTION TO CV

SEGMENTATION

WHAT IS SEGMENTATION?

Segmentation is finding consistent regions in an image.

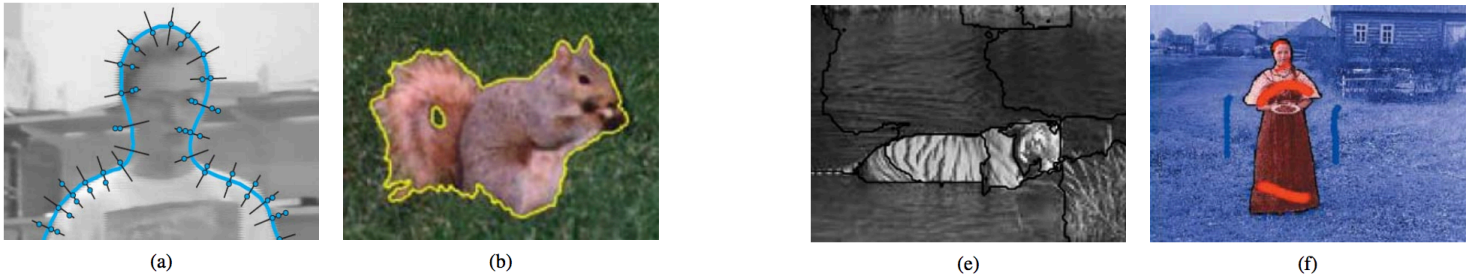
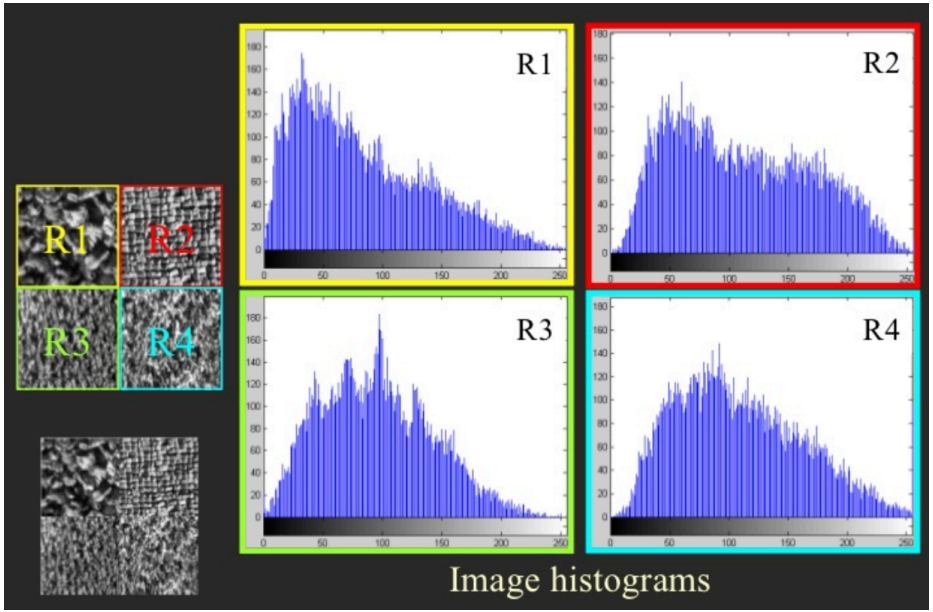


Figure 5.1 Some popular image segmentation techniques: (a) active contours (Isard and Blake 1998) © 1998 Springer; (b) level sets (Cremers, Rousson, and Deriche 2007) © 2007 Springer; (c) graph-based merging (Felzenszwalb and Huttenlocher 2004b) © 2004 Springer; (d) mean shift (Comaniciu and Meer 2002) © 2002 IEEE; (e) texture and intervening contour-based normalized cuts (Malik, Belongie, Leung *et al.* 2001) © 2001 Springer; (f) binary MRF solved using graph cuts (Boykov and Funka-Lea 2006) © 2006 Springer.

Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

SEGMENTATION TECHNIQUES

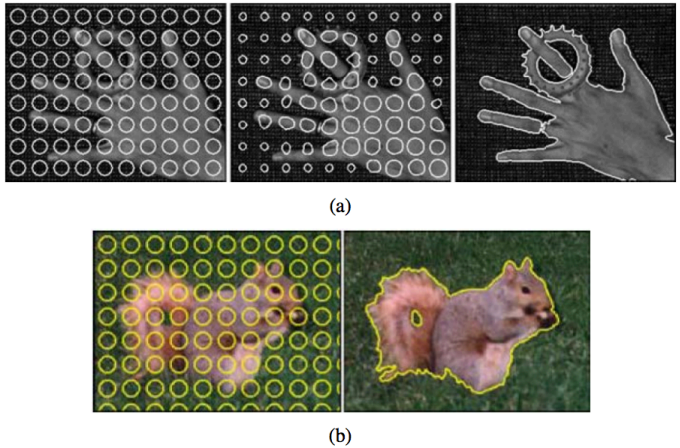
Segmentation techniques use statistics on pixel distribution within regions.



Source: https://www.slideshare.net/lalitxp/image-texture-analysis?next_slideshow=1

WITHOUT CLASSIFICATION

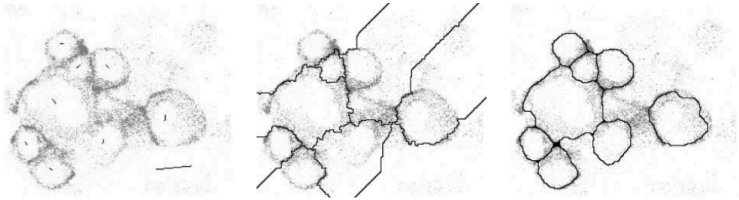
Many techniques are based on comparing adjacent regions.



(a)

(b)

Figure 5.11 Level set segmentation (Cremers, Rousson, and Deriche 2007) © 2007 Springer: (a) grayscale image segmentation and (b) color image segmentation. Uni-variate and multi-variate Gaussians are used to model the foreground and background pixel distributions. The initial circles evolve towards an accurate segmentation of foreground and background, adapting their topology as they evolve.



(a) (b) (c)

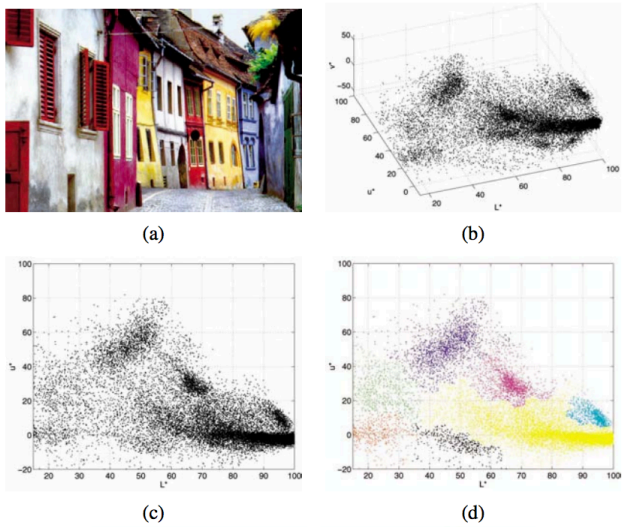
Figure 5.13 Locally constrained watershed segmentation (Beare 2006) © 2006 IEEE: (a) original confocal microscopy image with marked seeds (line segments); (b) standard watershed segmentation; (c) locally constrained watershed segmentation.

Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

UNSUPERVISED CLASSIFICATION

Clustering & neural networks* can be used to group similar patches.

* e.g., self-organizing maps (SOM) or constraint satisfaction neural networks (CSNN)



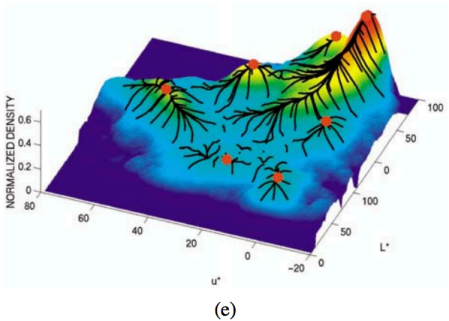


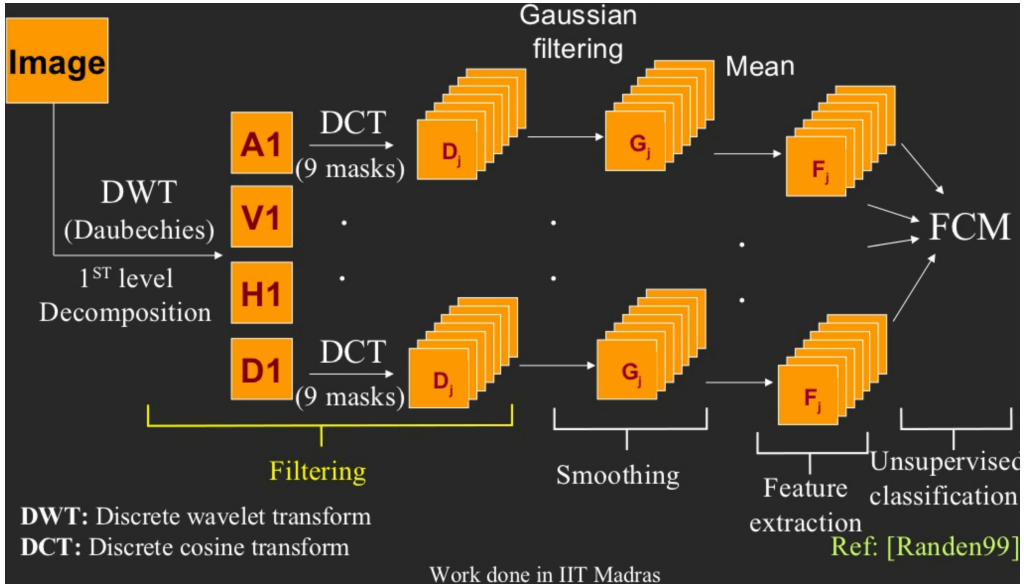
Figure 5.16 Mean-shift image segmentation (Comaniciu and Meer 2002) © 2002 IEEE: (a) input color image; (b) pixels plotted in L*u*v* space; (c) L*u* space distribution; (d) clustered results after 159 mean-shift procedures; (e) corresponding trajectories with peaks marked as red dots.

Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

UNSUPERVISED CLASSIFICATION

Clustering & neural networks* can be used to group similar patches.

* e.g., self-organizing maps (SOM) or constraint satisfaction neural networks (CSNN)

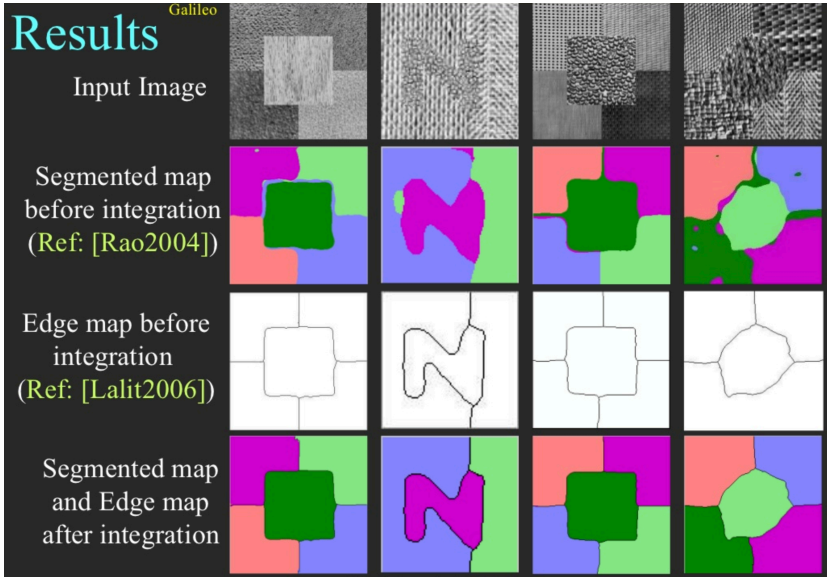


Source: https://www.slideshare.net/lalitxp/image-texture-analysis?next_slideshow=1

UNSUPERVISED CLASSIFICATION

Clustering & neural networks* can be used to group similar patches.

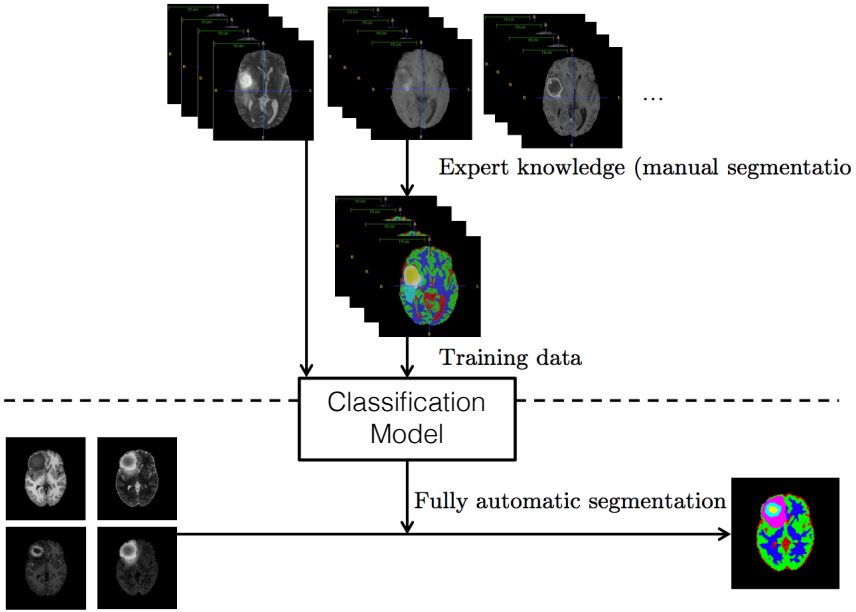
* e.g., self-organizing maps (SOM) or constraint satisfaction neural networks (CSNN)



Source: https://www.slideshare.net/lalitxp/image-texture-analysis?next_slideshow=1

SUPERVISED CLASSIFICATION

Many classification techniques can be applied to labelled data.



Source: http://people.ee.ethz.ch/~cattin/MIA-ETH/pdf/MIA-08-SupervisedSegmentation.pdf

PROBABILISTIC SEGMENTATION

Also called soft segmentation, it assigns pixels a probability of belonging to a segment.

Good for hair
and fuzzy patches

Algorithms can be based on
K-mean clustering
or Gaussian Mixture Models…

Source: https://www.slideshare.net/lalitxp/image-texture-analysis?next_slideshow=1

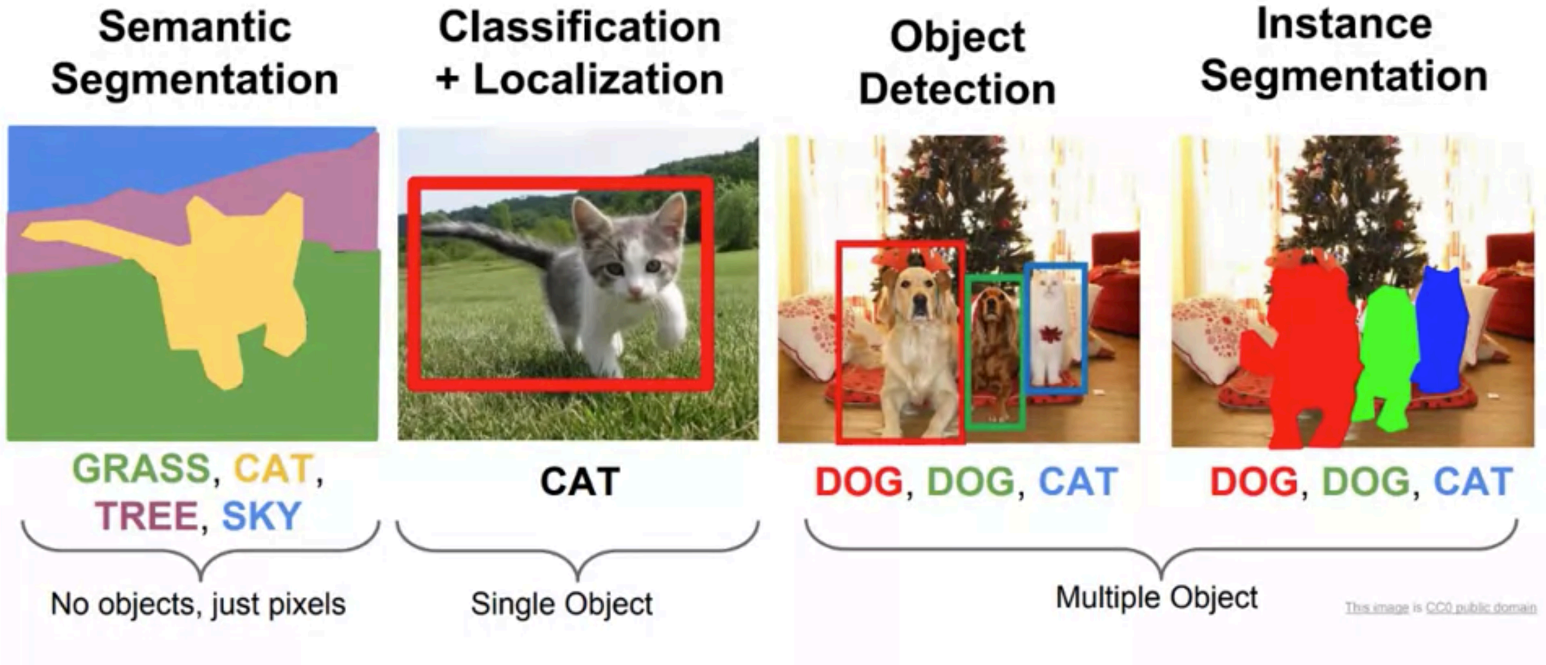
RESOURCES

Title of Resource:
Resource Type: Website
Description: A definition and discussion of the principles of grouping.
Title of Resource:
Resource Type: Video (4:26)
Description: A Khan Academy lecture on bottom-up versus top-down processing.
Title of Resource: Thresholding
Resource Type: Website
Description: Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images.
Title of Resource: Otsu Thresholding
Resource Type: Website
Description: Converting a greyscale image to monochrome is a common image processing task. Otsu's method, named after its inventor Nobuyuki Otsu, is one of many binarization algorithms. This page describes how the algorithm works and provides a Java implementation, which can be easily ported to other languages.
Title of Resource:
Resource Type: PDF
Description: This paper reviews the main approaches of partitioning an image into regions by using gray values in order to reach a correct interpretation of the image.
Title of Resource: Cluster Analysis
Resource Type: Website
Description: The definition of cluster analysis.
Title of Resource:
Resource Type: Video (1:21)
Description: The video shows my K-Means Clustering algorithm running on an image, iterating from K=1 to K=80 clusters, with the last 3 frames being the original image.
Title of Resource:
Resource Type: Video (2:58)
Description: This video accompanies the article "Semantic Soft Segmentation" by Yağız Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys and Wojciech Matusik. The article and additional resources are available on the project webpage: https://yaksoy.github.io/sss/.
Title of Resource:
Resource Type: Website
Description: How graph cuts are applied in the field of computer vision.
Title of Resource:
Resource Type: Video (2:11)
Description: This video is part of the Udacity course "Introduction to Computer Vision"

INTRODUCTION TO CV

OBJECT RECOGNITION

CLASSIFICATION TASKS

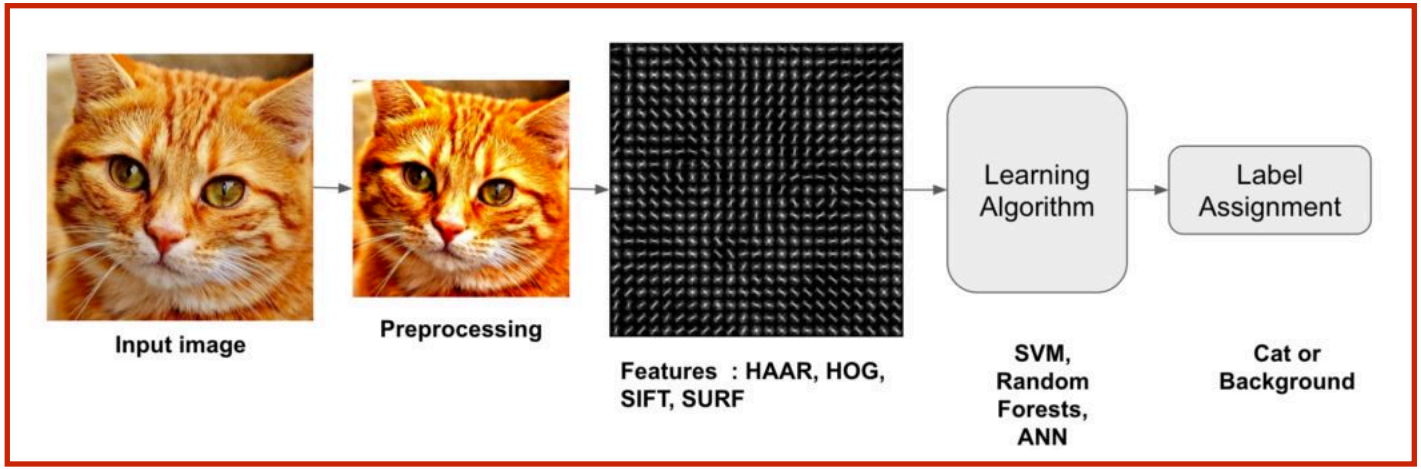


TYPICAL PIPELINE

From segmentation or object detection
(classifying segments or pixels as background or not)

Source: https://www.learnopencv.com/image-recognition-and-object-detection-part1/

TYPICAL PIPELINE



This pipeline is also used for segmentation or object detection

…but these can be unsupervised classification

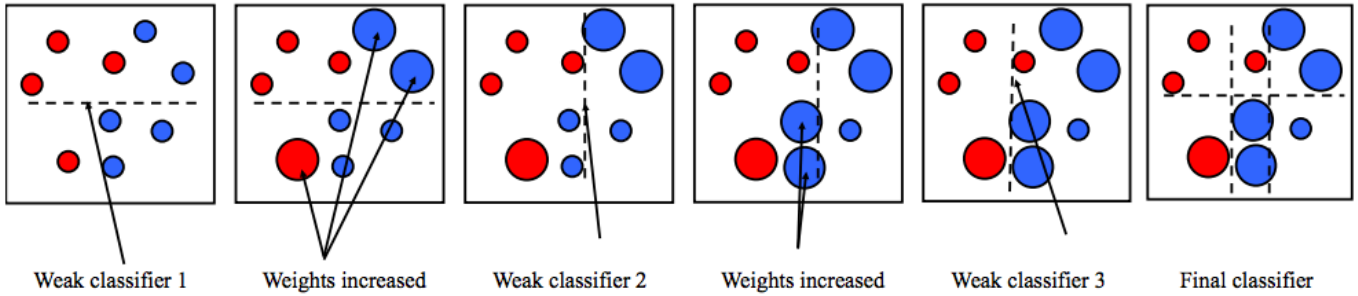
Source: https://www.learnopencv.com/image-recognition-and-object-detection-part1/

DIFFICULTIES

- **Occlusions** can hide key parts of objects.
- **Viewpoints** may not show all key parts of objects.
- **Illumination** can greatly change object appearances (e.g., shadows).
- **Shape-shifting** can be expected of many objects.
  (e.g., flexible bodies)
- **Intra-class variations** can be considerable.
  (e.g., chairs can be very different from one another)

TECHNIQUES

- **Neural Networks & Deep Learning** are the most common technique.
- **Support Vector Machines (SVM)** learn boundaries between classes in the chosen feature space.
- **Boosting** combines weak (extremely simple) classifiers.



Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

TECHNIQUES

- **Neural Networks & Deep Learning** are the most common technique.
- **Support Vector Machines (SVM)** learn boundaries between classes in the chosen feature space.
- **Boosting** combines weak (extremely simple) classifiers.
- Bag-of-Word learns typical parts of objects and their spatial distribution.
- Active Appearance Models specify typical key points, their spatial distribution, and their variability.

Source: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

RESOURCES

Title of Resource:
Resource Type: Video
Description: Find meaning in visual data on Watson Studio! Analyze images for scenes, objects, faces, and other custom content. Take advantage of pretrained models, or create your own custom classifier. Develop smart applications that analyze the visual content of images or video frames to understand what is happening in a scene.Then, deploy your model for use in applications.
Title of Resource:
Resource Type: Website Article
Description: Compares and contrasts Computer Vision and Visual Recognition to clearly explain their differences.
Title of Resource: Hough Transform
Resource Type: Website
Description: Defines a Hough Transform.
Title of Resource:
Resource Type: Video
Description: This video explains how the Hough Transform works to detect lines in images. First, apply an edge detection algorithm to the input image, and then compute the Hough Transform to find the combination of Rho and Theta values in which there are more occurrences of lines.
Title of Resource:
Resource Type: Journal Article (PDF)
Description: This paper introduces a new scene-centric database called Places with over 7 million labeled pictures of scenes. It then proposes new methods to compare the density and diversity of image datasets and shows that Places is as dense as other scene datasets and has more diversity.
Title of Resource:
Resource Type: Website
Description: Defines k-nearest neighbors algorithm.

Amsterdam University of Applied Sciences | DIGITAL SOCIETY SCHOOL

Emma Beauxis-Aussalet

18-07-2019

QUESTIONS / DISCUSSIONS

YOU TRANSFORM SOCIETY BY DESIGN

Amsterdam University of Applied Sciences

DIGITAL SOCIETY SCHOOL WHERE CHANGE TAKES SHAPE