COMPUTER VISION 2018 - 2019
>IMAGE FEATURES

UTRECHT UNIVERSITY

RONALD POPPE

IMAGE CLASSIFICATION

Aim is to find regions in an image that correspond to objects of interest

We will first look at how to represent (regions in) an image

- Lectures 5-6

We then address the training and testing of machine learning classifiers

- Lectures 7-8

Finally, we turn to convolutional neural nets for detection

- Lectures 9-11

OUTLINE

Image descriptors

Applications

Issues

Low-level image descriptors

Histograms of oriented gradients (HOG)

Scale-invariant feature transforms (SIFT)

IMAGE DESCRIPTORS

IMAGE DESCRIPTORS

Describe the characteristics of an image:

- Derived from the pixels

Describe an image (or part of it) in a compact way

- Should ideally be invariant to nuisance factors (viewpoint, scale, illumination, etc.)

Similar images should have similar image descriptors

- As usual: what is similar?

IMAGE DESCRIPTORS^2

Different object, different image

IMAGE DESCRIPTORS[3]

Same object, different image

IMAGE DESCRIPTORS[4]

Same object, same image?

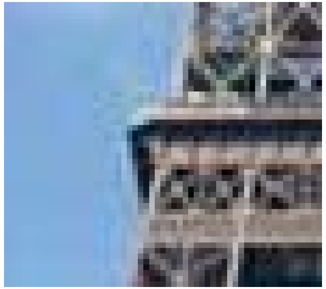


IMAGE DESCRIPTORS[5]

Simplest image descriptor: the pixels in an image

- Width * height * channels number of dimensions
- Not really compact
- Not really invariant to nuisance factors

IMAGE DESCRIPTORS[6]

Image descriptors are often termed image features

- A feature vector is a vector representation with each number/dimension derived from the image
- E.g. $\boldsymbol{x} = (x_1 \ldots x_n)$ with $n = rows * columns * color\ channels$ and each dimension is a pixel color value

Today, we will look at various image descriptors:

- What are they used for?
- Which nuisance factors should they be invariant to?
- How to calculate them?

APPLICATIONS

APPLICATIONS

Image stitching

Object detection

Duplicate detection

Video stabilization

IMAGE STITCHING

When images partly overlap, they can be stitched together

- Useful for making panoramas
- Used in Google Street View, any smartphone and camera

Overlap is never perfect. Differences in:

- Rotation
- Scale
- Lighting
- Etc.

IMAGE STITCHING[2]

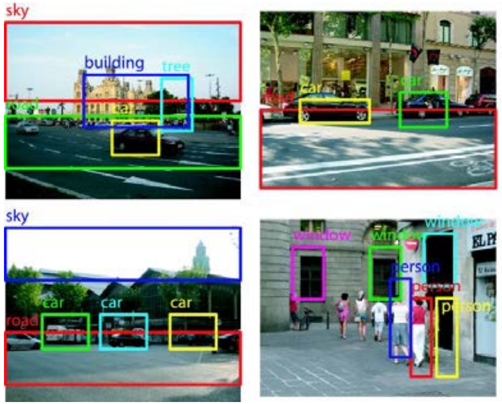
Example: stitch two images together

OBJECT DETECTION

Object detection is the process of finding objects in an image

Two types of object detection

- Generic: recognize classes of objects (all cars)
- Specific: recognize a specific instance (BMW 118)



OBJECT DETECTION[2]

Different levels of granularity:

- Just saying if the image depicts the object: recognition
- Finding the location (bounding box): detection
- Determining which pixels are part of it: segmentation

DUPLICATE DETECTION

Duplicate detection considers the problem of finding images that are identical apart from:

- Scale (resolution)
- Framing (crop region)
- Encoding (jpeg)
- Coloring (grayscale/rgb, variations, copies)
- Rotation (only in 2D)

This is a verification task:

- Are these two images near-duplicates?

DUPLICATE DETECTION[2]

Typical examples:

- Find pictures of the Mona Lisa
- Find the same picture but in a larger resolution
- Find out who uses your (copyrighted) pictures

FACE VERIFICATION

Face verification is the process of determining whether a face belongs to a specific user

- This is (obviously) a verification task

Can be done based on

- 2D images
- 2D + depth images (Kinect)
- Full 3D model (range scanner)
- Near infra-red

FACE VERIFICATION[2]

Applications are mainly in security

- Login with your face instead of password
- To filter out people with a neighborhood restriction

FACE VERIFICATION[3]

But applications can also be outside security:

- Searching for pictures of (famous) people
- Understanding/recognizing movies/series
- Automatic subtitle generation

ISSUES

ISSUES

The same object or scene can appear differently in images

- Viewpoint, illumination and image quality affect the image

We don’t want all factors to influence the image descriptors

- We call such factors nuisance factors
- It’s often favorable to have an image descriptor that is invariant to many nuisance factors

VIEWPOINT

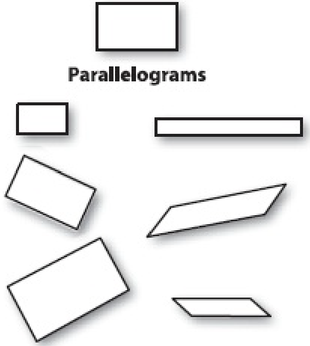
Images can be taken from the same viewpoint, but with a different rotation

- This is called in-plane rotation

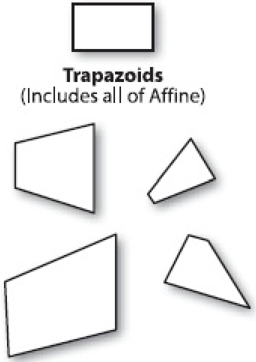
AFFINE PROJECTION

Or from the same distance but with a different angle

- This is termed an affine transformation



PERSPECTIVE PROJECTION



Or from another distance, with objects further away being smaller

- This is termed a perspective transformation

DIRECT LIGHTING

The direction of the light causes variation in

- Shadows
- Specular highlights



INDIRECT LIGHTING

Indirect lighting refers to (the amount of) ambient light

Less light lowers the contrast

- Color values are in smaller space

IMAGE QUALITY

The image quality is determined by:

- Image compression
- Resolution
- Color depth

IMAGE COMPRESSION

Images typically stored with compression

- JPEG, GIF, PNG, etc.

Compression can take many forms

- Usually detail is lost
- “noise” or “patterns” can be introduced

Original Image

GIF without dithering

GIF with dithering

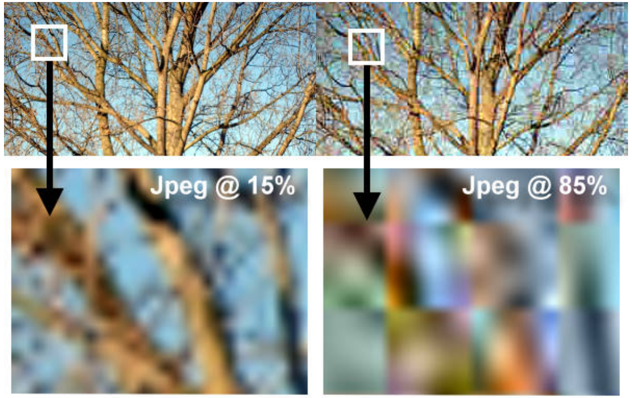


IMAGE RESOLUTION

When reducing the resolution of an image

- Details get lost
- Contrast regions become less pronounced

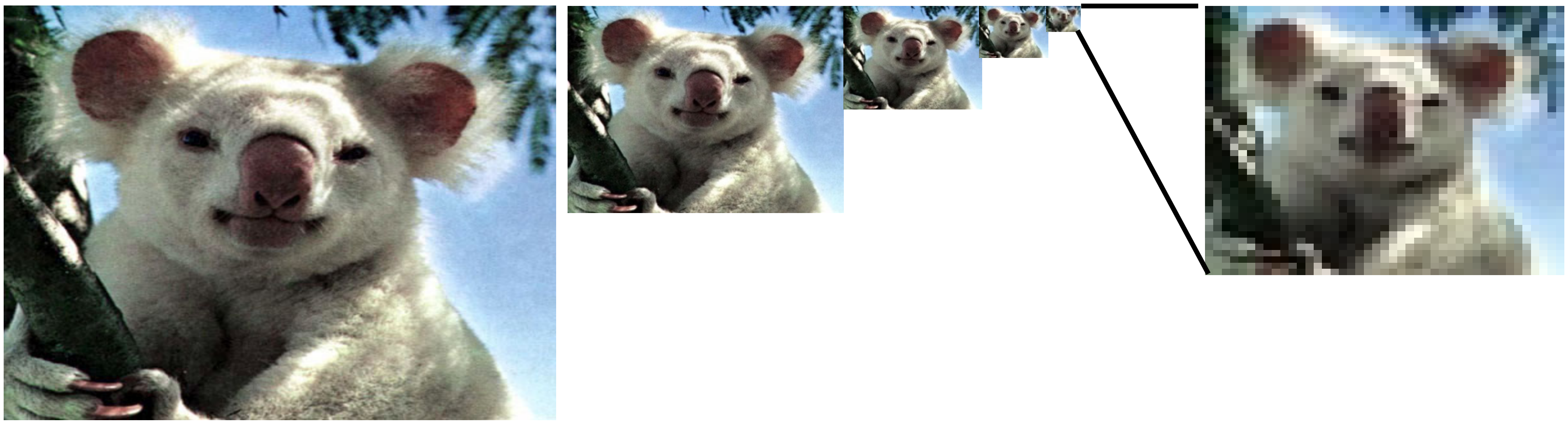


IMAGE RESOLUTION[2]

What is depicted here?

OBJECT ARTICULATION

Objects are not always rigid but often consist of parts that can move

- We call these objects articulated
- Detecting them is more difficult as they have different “shapes”
- Typical for humans in action

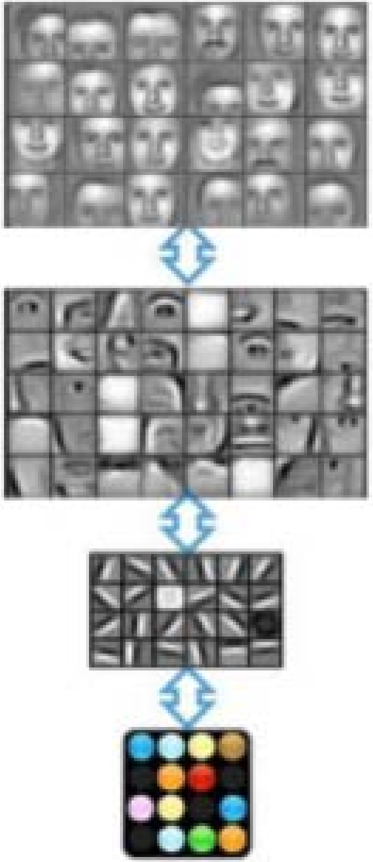
RECAP

When describing and comparing images, we would like our representations to be invariant to these issues

- How to describe images so we can achieve this?

We distinguish between low-level and high-level image descriptors

- Low-level descriptors are close to the pixels
- High-level descriptors are more semantic, on a higher abstraction level



QUESTIONS SO FAR?

LOW-LEVEL IMAGE DESCRIPTORS

LOW-LEVEL IMAGE DESCRIPTORS

Low-level image descriptors operate on pixels of an image

- Also called local descriptors

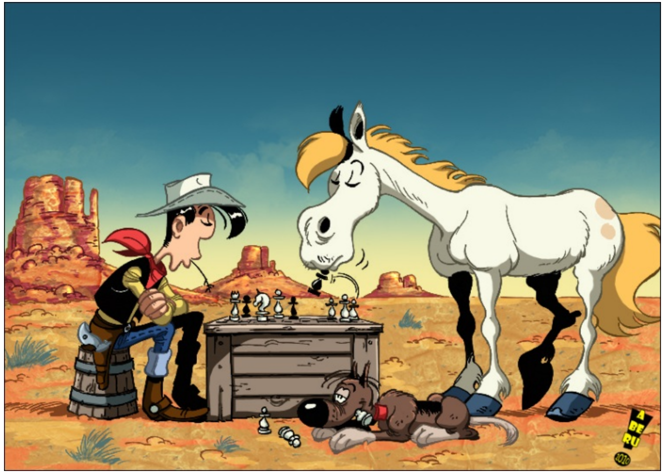
The most common ones are based on:

- Color (intensity)
- Edges (contrast)
- Motion (only for video, discussed in next lecture)

LOW-LEVEL IMAGE DESCRIPTORS^2

Makes sense for object detection:

- Objects typically stand out from their surroundings by different colors
- These also cause high contrast values



COLOR DESCRIPTORS

We looked at those before:

- Mean color
- Color histogram (equidistant bins, from clusters)
- Gaussian mixture model

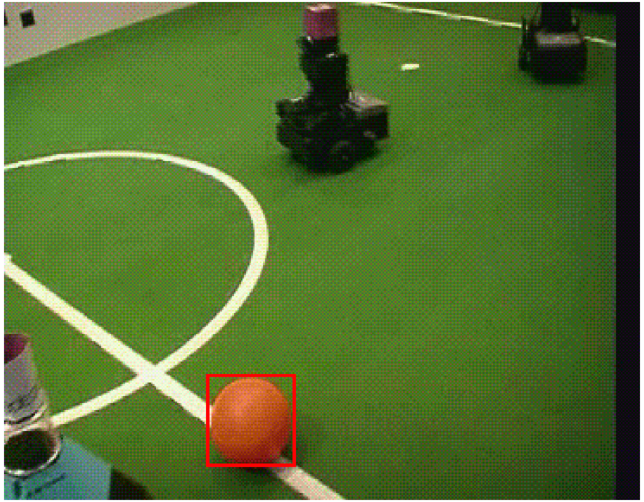
Several “flavors”:

- Different color spaces (RGB, HSV, etc.)
- Per channel, or combined

COLOR DESCRIPTORS[2]

Color descriptors for object recognition

- Can be a good cue
- Can be a bad cue



EDGES

Edges arise when neighboring pixels have contrasting intensities

- Each pixel can be an edge pixel or not

EDGES²

Calculated by taking the derivative of a pixel in both the horizontal and vertical direction

- Edges have a direction (orientation) and magnitude (strength)

Invariant to:

- Specific color

PIXEL DERIVATIVES

We can take the derivative of a pixel by applying filters:

- G_x is derivative in horizontal direction
- G_y is derivative in vertical direction

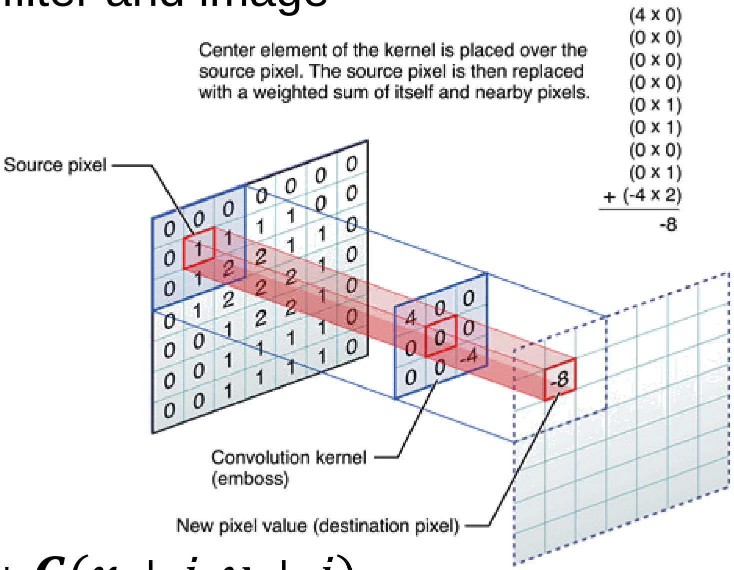
Prewitt: $$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Sobel: $$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

PIXEL DERIVATIVES^2

We apply filters using convolution:

- Center pixel is replaced by weighted sum of filter and image
- Calculated using dot product



For patch *A* and 3x3 kernel *G*:

- $A'(x, y) = \sum_{i=-1\ldots1, j=-1\ldots1} A(x+i, y+j) * G(x+i, y+j)$

PIXEL DERIVATIVES[3]

Example:

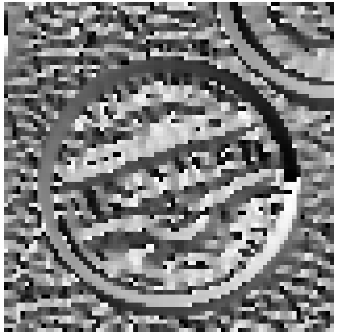
$$
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|c|c|}
\hline
1 & 1 & 0 & 0 & 0 \\ \hline
1 & 1 & 1 & 0 & 0 \\ \hline
1 & 1 & 1 & 1 & 0 \\ \hline
1 & 1 & 1 & 1 & 1 \\ \hline
1 & 1 & 1 & 1 & 0 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|c|}
\hline
 & & & & \\ \hline
 & \boxed{-1} & \boxed{-2} & -2 & \\ \hline
 & 0 & -1 & -2 & \\ \hline
 & \boxed{0} & 0 & -2 & \\ \hline
 & & & & \\ \hline
\end{array}
$$

PIXEL DERIVATIVES[4]

Once we have the derivates in x- and y-direction, we calculate the gradient magnitude as follows: $\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$

And the gradient orientation: $\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$

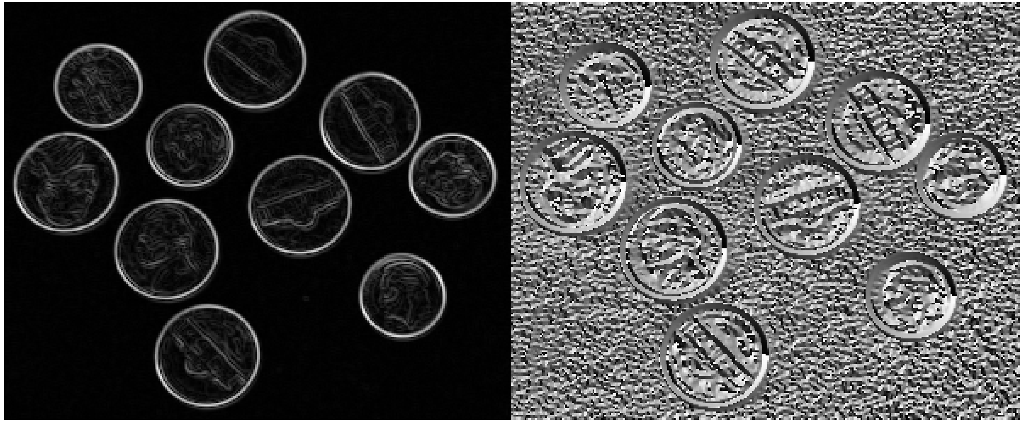
PIXEL DERIVATIVES[5]



Gradient magnitude and direction are informative:

- Magnitude is indicator of contrast
- Direction determines the direction of the edge

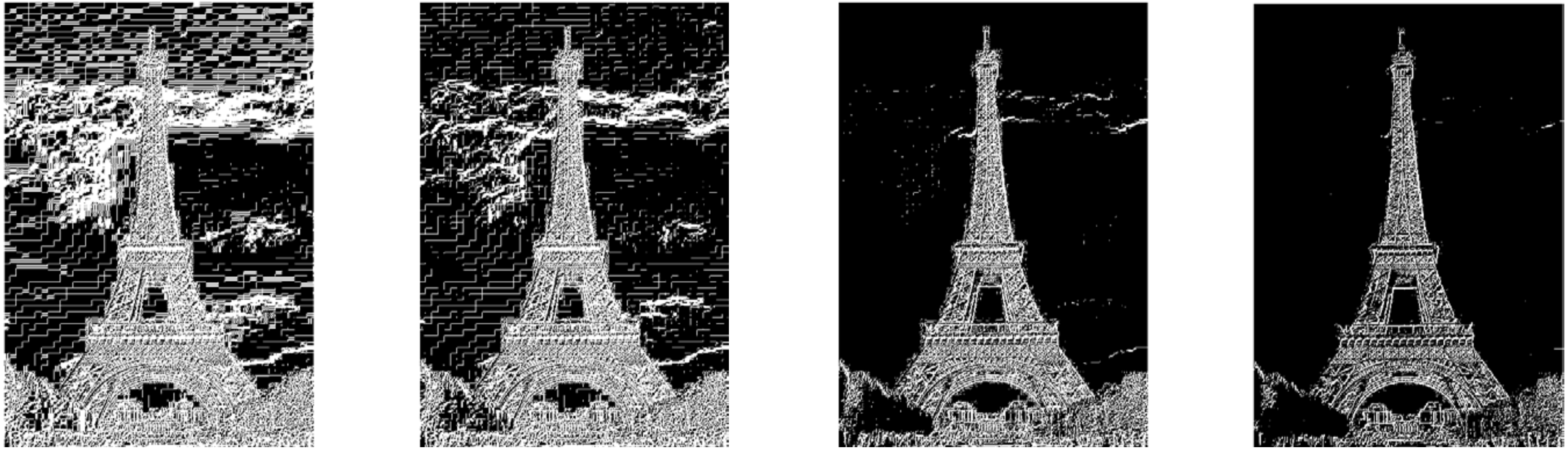
Direction can be noisy when magnitude is low



PIXEL DERIVATIVES[6]

To get a binary edge image, we can put a threshold on the gradient magnitude

- Noisy pixels typically have strong edges
- Determining threshold is subjective



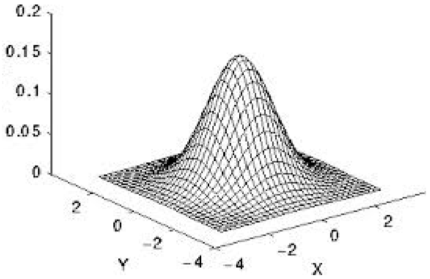
PIXEL DERIVATIVES[7]

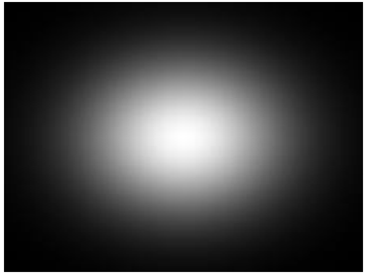
Different ways of suppressing noisy pixels:

- First apply a Gaussian filter (convolution)

Each pixel becomes weighted average of surrounding pixels

- Image appears more blurry
- Variance can be tuned (sigma)
- For discrete values: different window sizes (5 x 5)





1/256 x

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

PIXEL DERIVATIVES[8]

When applying a Gaussian filter:

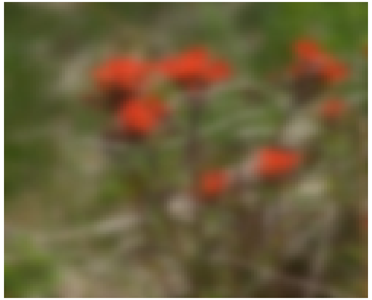
- Noisy pixels are averaged
- Details are also lost (is this a problem?)

Typically, edge detection is preceeded by Gaussian filtering with a small window size (3 x 3, 5 x 5)

Original Image

Blurred Image ($\sigma = 10$)

Blurred Image ($\sigma = 20$)



Blurred Image ($\sigma = 40$)

CANNY EDGE DETECTION

Another way to suppress noise due to noisy pixels:

- Look at neighboring pixels

Canny edge detection uses the following steps:

- Gaussian filtering (small window)
- Obtain gradient magnitude/direction per pixel (Sobel, Prewitt)
- Non-maximum suppression
- Tracing edges

CANNY EDGE DETECTION[2]

We want edges at points with (locally) maximum magnitude

- We want to ignore edges with lower values
- Non-maximum suppression is an edge thinning technique (remember erosion)

Consider the two neighbors orthogonal to the gradient direction

- Suppress the current pixel if it hasn’t the maximum magnitude

Four options:

- Left – right
- Top – bottom
- Left-top – right-bottom
- Right-top – left-bottom


CANNY EDGE DETECTION[3]

Eventually, we are only interested in edges that are “longer”

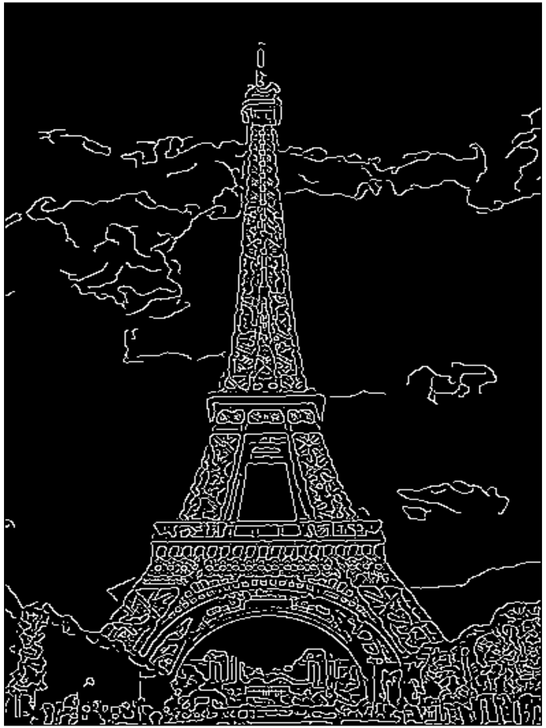
- Short edges considered to correspond to noise

Canny edge detection uses two-step approach:

- First filter out edges with low gradient magnitude (threshold)
- Next, trace each edge in the direction orthogonal to the gradient (allow some flexibility in direction)
- Only pixels belonging to longer traces are kept

CANNY EDGE DETECTION[4]

Result of Canny edge detection based on Prewitt operator



RECAP

Color and edges say something about an image

- Objects stand out from their surroundings based on color or contrast

Neither color descriptors nor edges invariant to:

- Rotation
- Scale
- Viewpoint
- Location in the image

We need better image descriptors!

QUESTIONS?

HISTOGRAM OF ORIENTED GRADIENTS

HOG

Histograms of oriented gradients (HOG) densely encode edges within a grid [Dalal & Triggs, CVPR 2005]

The representation is invariant to scale

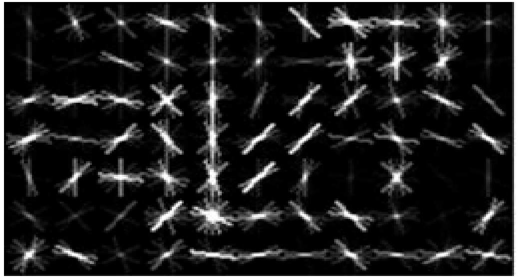
- Based on bounding box

HOGs are somewhat invariant to

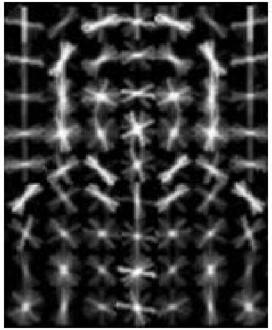
- Rotation (small angles)
- Local variations
- Illumination and color (based on edges)

HOG²

Can you guess which object was on the picture of which this HOG descriptor was calculated:



And this one?



HOG[3]

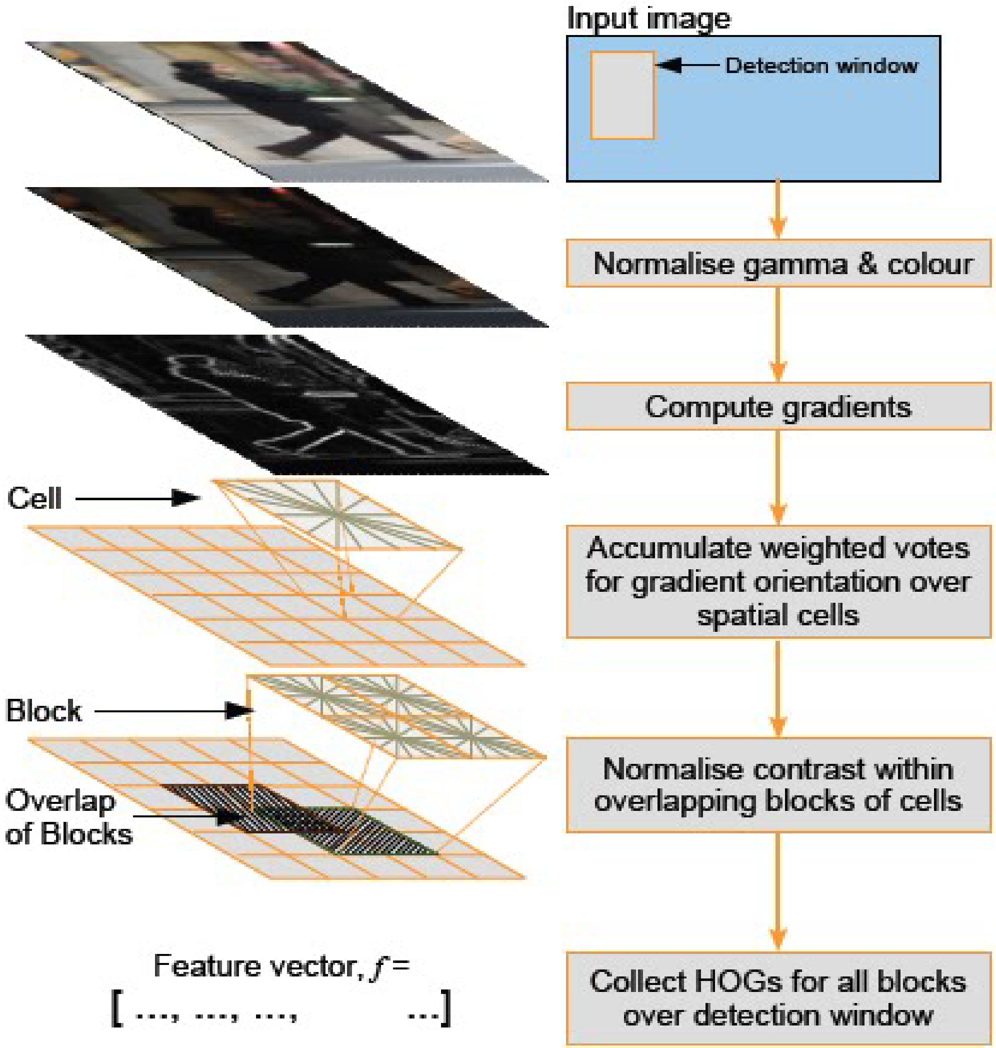
HOGs are calculated in several steps, based on a bounding box:

- Apply Gaussian filter + normalize color
- Calculate edge magnitude and orientation

Edge detection!

- Summarize edges for all pixels per cell in the grid
- Summarize blocks of cells
- Normalize descriptor to unit length

HOG[4]


Input image
Detection window
Normalise gamma & colour
Compute gradients
Accumulate weighted votes for gradient orientation over spatial cells
Normalise contrast within overlapping blocks of cells
Collect HOGs for all blocks over detection window
Cell
Block
Overlap of Blocks
Feature vector, f =
[..., ..., ..., ...]

HOG[5]

Grids contain *n x m* cells, usually of equal size

- So number of pixels per cell is also equal

Blocks contain several cells, typically 2 x 2

- Blocks are overlapping
- Each block is normalized to account for local intensity differences

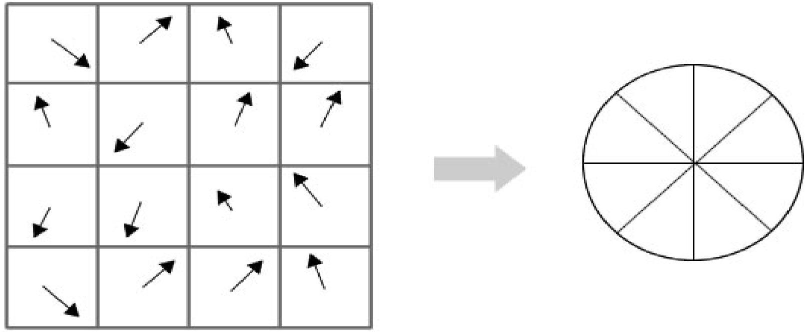
HOG[6]

Per cell, we construct a histogram of orientations

- Bins can be 0-20, 20-40 degrees, etc., or 0-45, 45-90, etc.

Each pixel's orientation contributes to orientation histogram of the cell

- Bins determined by gradient orientation
- The amount of "weight" determined by gradient magnitude

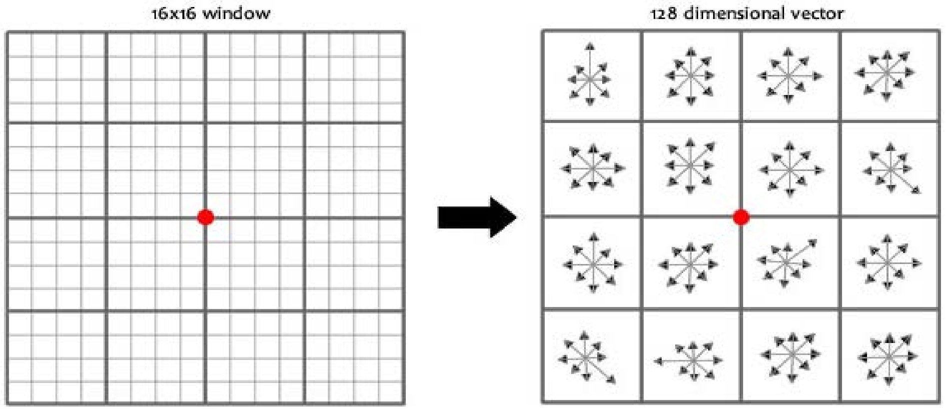


HOG[7]

Next step is to calculate blocks:

- Histograms in a block of neighboring cells are concatenated
- Length of the concatenated histogram is normalized to unit length (sum = 1)

Example: block of 4 x 4 cells, each 4 x 4 pixels, and 8 orientations



HOG[8]

Advantages of HOG:

- Can be calculated quickly (edge derivatives calculated once)
- Quite robust to local variations (especially within the cell)
- Quite robust to illumination changes (due to block normalization)

HOGs were introduced for pedestrian detection:

- Given a patch (region in the image), determine if it is a person
- Patch described as a HOG descriptor
- Each HOG descriptor was then classified as corresponding to a person or not

SCALE-INVARIANT FEATURE TRANSFORM

SIFT

Scale-invariant feature transform (SIFT) is an algorithm to describe image features [Lowe, 1999]

- It is commonly used in matching, stitching and tracking
- Try out the demo: http://www.cs.ubc.ca/~lowe/keypoints/

SIFT features are invariant to:

- Scale
- Rotation
- Partially to viewpoint changes
- Partially to illumination changes

SIFT[2]

Similarities with HOG:

- Basis are gradient differences
- Final descriptor shares similarities with orientation histogram

Differences with HOG:

- Not calculated at each pixel but only at specific points (sparse vs. dense)
- Encodes scale and rotation
- Can deal with partial occlusion

SIFT[3]

The SIFT algorithm has several steps:

- Detect scale-space extrema
- Detect keypoints
- Determine orientation
- Determine local descriptor

Once the local descriptors have be determined, they can be used for matching

SIFT[4]

First step is to find locations and scales that can be repeatedly assigned under different views of the same object

- Remember the nuisance factors!

When the distance to an object changes, so does:

- The size in the image
- The amount of detail that is visible

SIFT addresses these issues using:

- Size: pyramid images
- Detail: Gaussian filters

SIFT[5]

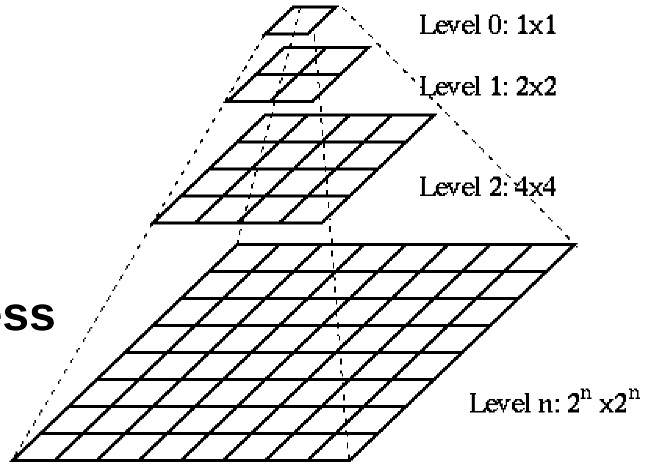
SIFT takes an image and analyses it at different scales

- Each scale is half the previous one
- All images together form a pyramid

At each level, Gaussian filters are applied

- Different levels of variance

Cope with objects of different sizes and sharpness



SIFT[6]

Example with 3 scales and 6 levels of Gaussian filtering

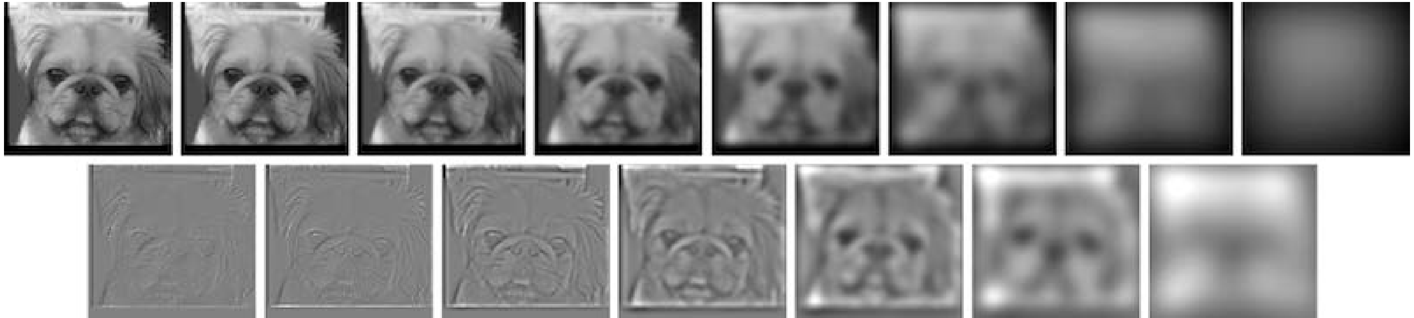
SIFT[7]

Images with the same scale but different Gaussian filtering are compared pairwise:

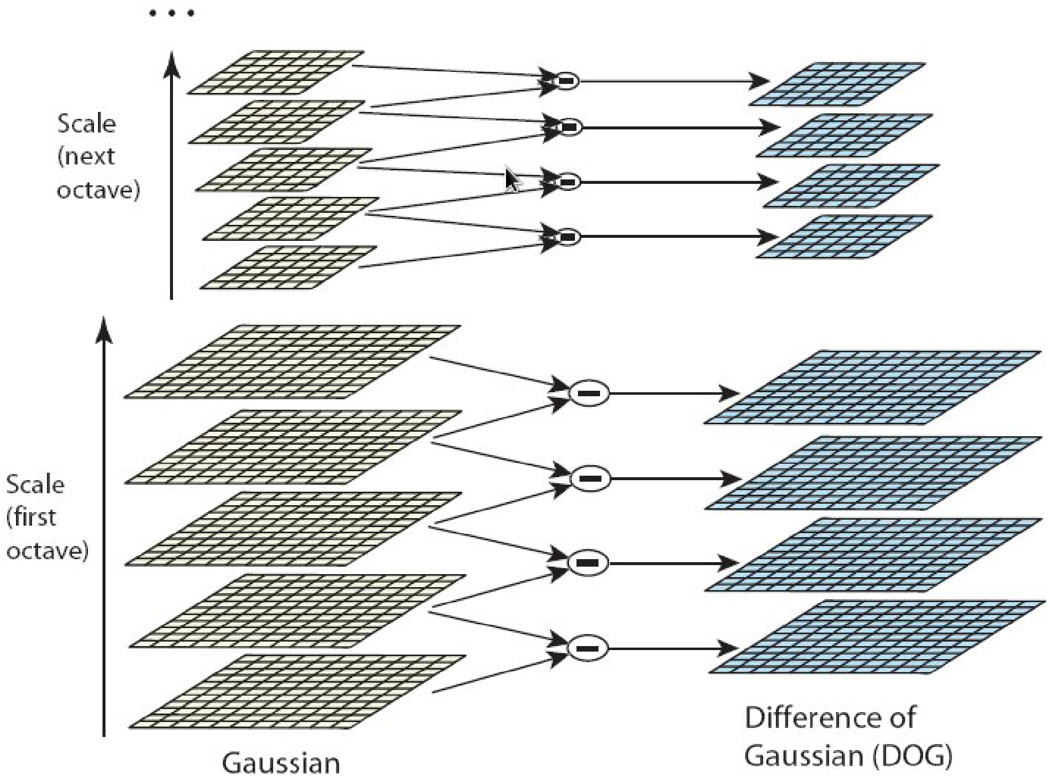
- This is termed a Difference of Gaussian (DOG)

Larger differences correspond to pixels that differ from their surroundings

- These locations are interesting
- Typically edges and corners



SIFT[8]

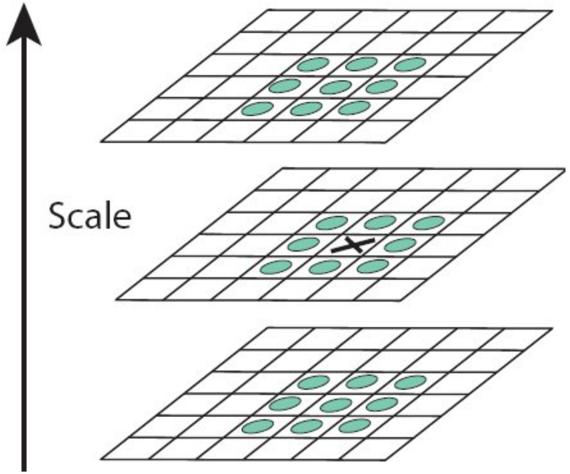


...
Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

SIFT[9]

Once we have the DOG for different scales, we need to select the local minima/maxima:

- Coarser scales are interpolated



Compare each pixel to:

- Its 8 neighbors on the same level
- Its 9 neighbors from scale above
- Its 9 neighbors from scale below

Pixel is selected if it is the maximum

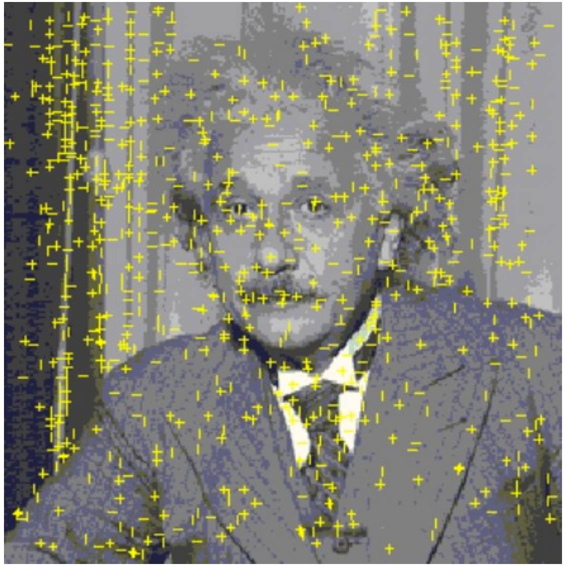
SIFT[10]

We get many keypoints, some of which are unstable

We can refine the selection:

- Remove keypoints with low contrast
- Remove keypoints along an edge

We use gradient orientation and magnitude again

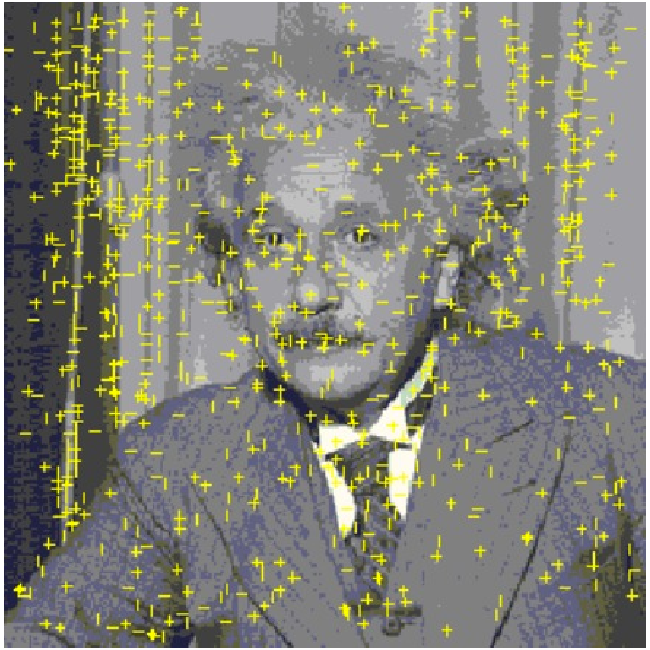


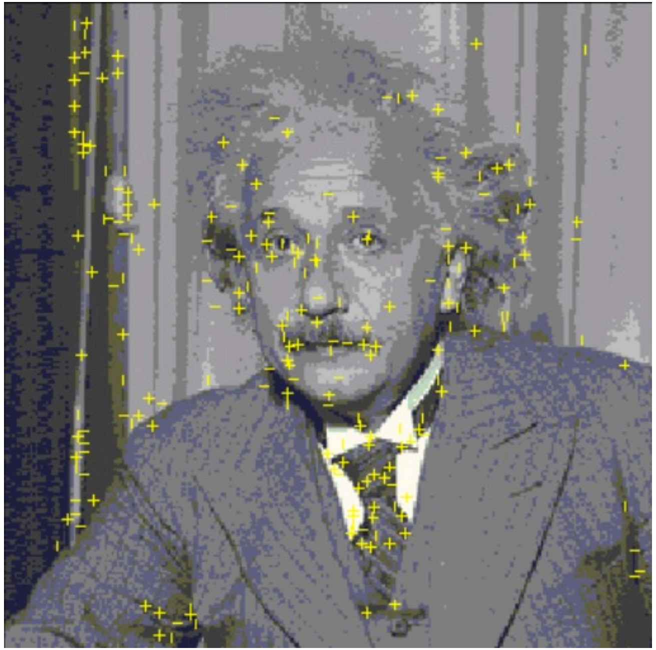
Remaining keypoints are at corners

- Repeatable!

SIFT[11]

After selection:

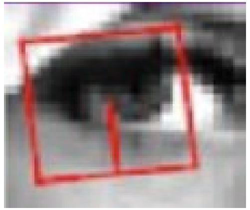


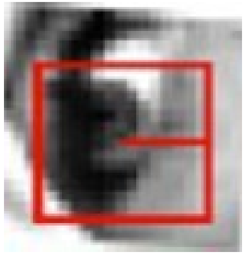


SIFT[12]

We now have a set of keypoints

- Each has a location
- Each has an orientation (determined from gradient)
- Each has a scale (determined from scale in which maximum was found)





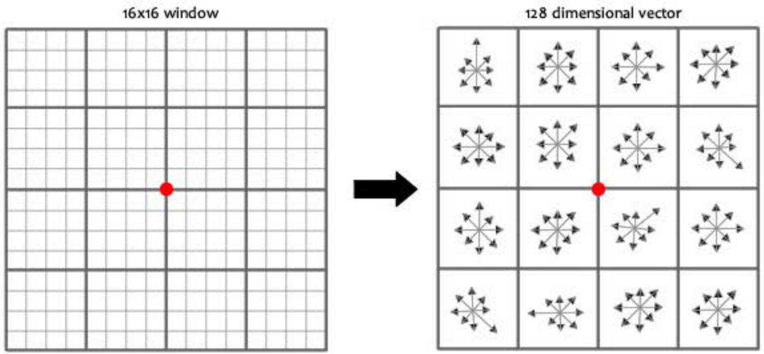
SIFT[13]

We can calculate the local descriptor:

- Histogram of orientations in a grid around the keypoint, with scale and orientation taken into account

Very similar to HOG. Small differences:

- Orientations are weighted with a Gaussian centered on the keypoint (pixels further away have less influence)
- Normalization is slightly different



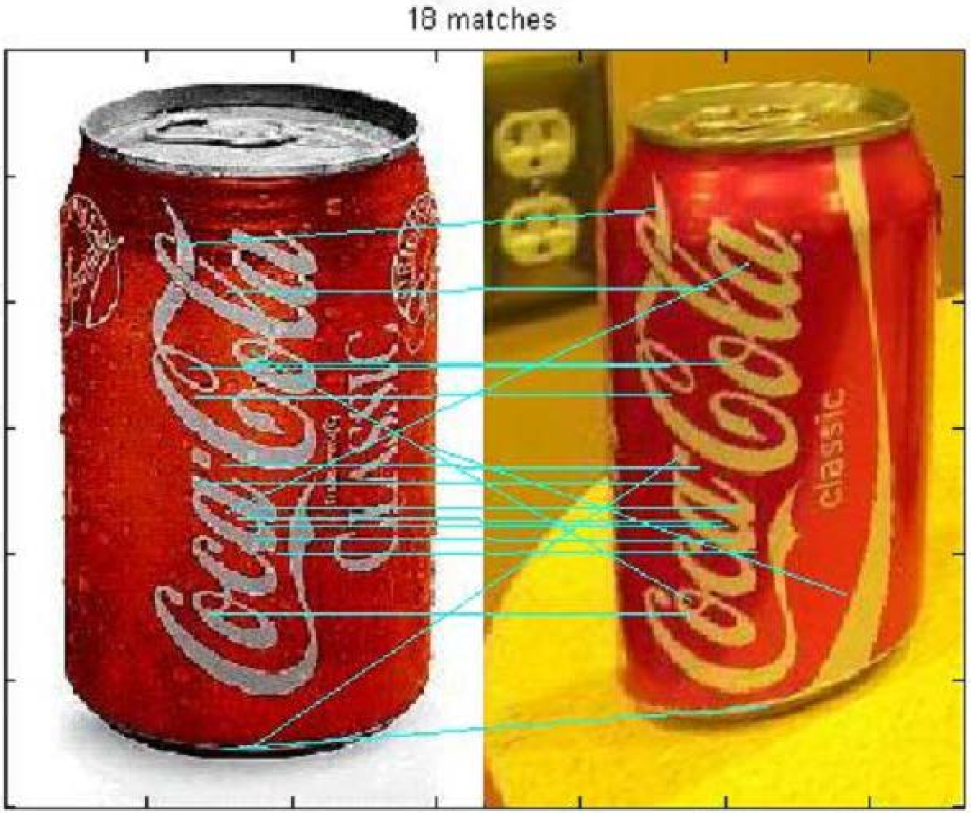
SIFT[14]

SIFT points can be matched based on Euclidian or Chi squared distance

For object matching, we want to find pairs of matching SIFT features:

- First find candidate features
- Then look at the distance, orientation and scales between the pairs
- Do some filtering and set a threshold on the matches
- The more matching pairs, the better the object match

SIFT[15]



RECAP

We looked in-depth at two state-of-the-art techniques

Histograms of oriented gradients are fast and can be matched densely in an image

- Ideal for pedestrian detection, but also for objects with similar orientation

Scale-invariant feature transforms are somewhat slower but can cope with differences in viewpoint, illumination and scale

- Ideal of object detection if the orientation of the object can vary

QUESTIONS?

ASSIGNMENT

ASSIGNMENT

Assignment 3:

- Tracking based on color models
- Don’t start too late, this assignment is more elaborate!

Deadline is Sunday March 10, 23:00

Assignment help session Thursday 11:00-12:45, RUPPERT-042

NEXT LECTURE

Optical flow

- Motion in video
- Also an image feature

Tuesday 13:15-15:00, BESTUURS-LIEREGG

QUESTIONS?