Discussion 3

CS188 - Fall 19

Objectives

- Image representations
  - Why not pixels?
- (R)eview of image features
  - Why are they needed in computer vision?
  - Descriptors
  - Keypoints
- Computing image features
  - SIFT
  - Some Image processing tools
- Bag-Of-Words Model
  - Why does it work?

Goals and constraints

- The goal of computer vision could be summed up as helping computers interpret and understand their environment
  - One could define intelligence as ‘capacity for survival’
  - This implies ability to interact with your environment
  - Which implies an ability to understand it
  - No understanding => no interaction => no intelligence

- The only visual data that computers can leverage are images
  - 2D arrays of numbers
  - This is the data we can build intelligence from

Understanding our environment

- Humans recognize scenes (3D) through **semantics**
- From an evolutionary perspective, you need to be able to determine if you’re in danger

Understanding our environment

- Humans recognize scenes (3D) through **semantics**
- From an evolutionary perspective, you need to be able to answer:

Should I start running ?

Understanding images

- Humans recognize scenes through the objects within it
  - Eg: a blackboard, chair, tables, projector, people ->
  - Eg: sink, table, chairs, oven, cupboards ->

Understanding images

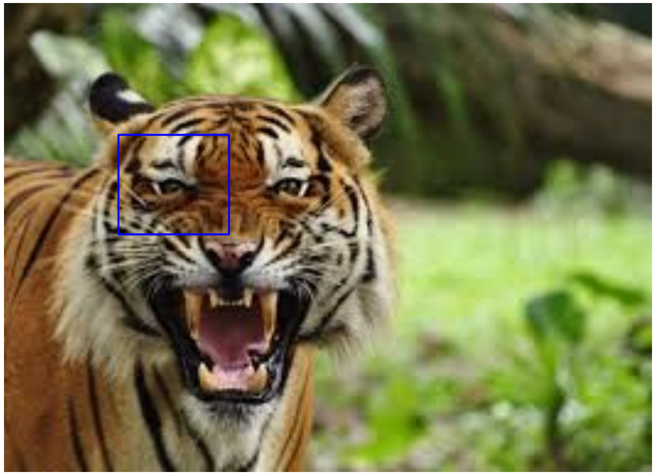
- Humans recognize scenes through the objects within it
  - Eg: a blackboard, chair, tables, projector, people -> classroom
  - Eg: sink, table, chairs, oven, cupboards -> kitchen
- If you can characterize objects in a scene, you can recognize the scene
  - If we replace ‘scene’ by ‘image’, we replace ‘object’ by ???

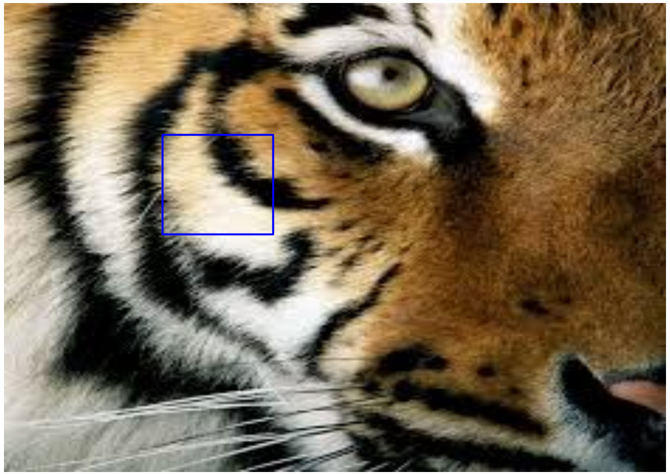
Understanding images

- Humans recognize scenes through the objects within it
  - Eg: a blackboard, chair, tables, projector, people -> classroom
  - Eg: sink, table, chairs, oven, cupboards -> kitchen
- If you can characterize objects in a scene, you can recognize the scene
  - If we replace ‘scene’ by ‘image’, we replace ‘object’ by ‘patch’
  - A patch is just a subset (a small square) of the image
- If you can recognize patches in an image, you can recognize the image
  - Can we do this by comparing pixel values?

Using pixels

- Let's say that the computer understand that the left picture is a tiger. How do I use pixels to recognize the other tiger?
  - Looking at same pixel locations -> no match!
  - We need another **representation**



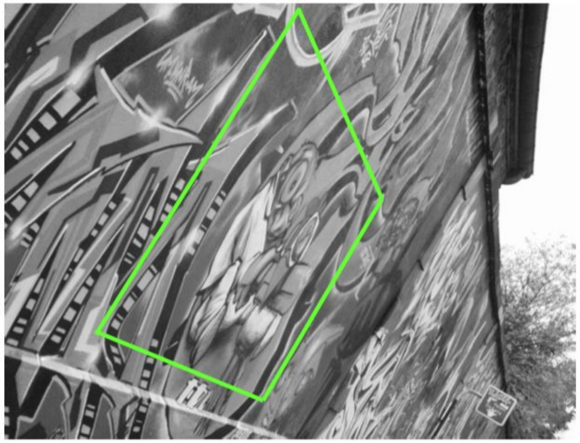


Conclusion

- We would like to represent images as a collection of patches
- If we were able to match patches in different images, we could recognize objects, and therefore scenes
- This could be used for image classification (and a lot of other things we'll talk about later)
- We cannot use pixels to match patches
  - Changing lighting would change pixel values
  - Changing scale would change pixel values
  - Changing viewpoint changes pixel values
- How do we do patch matching?

Object recognition and patch matching

- Assume that you're trying to recognize the same object in different images:



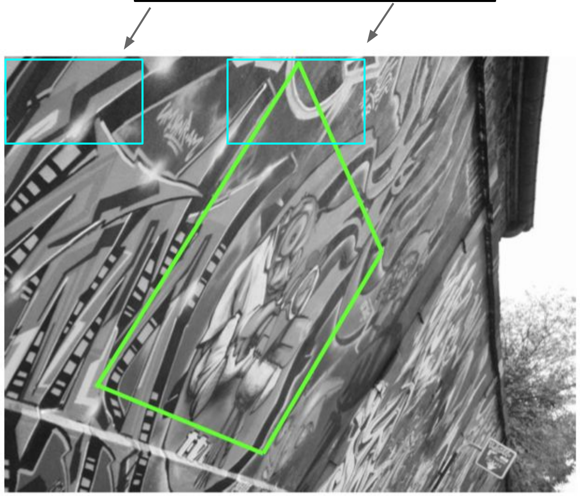
Mikolajczyk sequence

Patch matching: we would like to find the left image in the right image

Patch matching

- How do we proceed?
- Comparing pixel values

Here ? Or there?



Mikolajczyk sequence

Which pixels to compare? -> This won’t work

Matching pixels: a bad idea

- A change of scale will change pixel values in the image!
  - The patch will get larger or smaller in the image
  - Comparing pixel values won't work
- A change of orientation will change pixel values in the image!
  - The patch will deform, and move in the image
  - Comparing pixel values won't work
- A change of illumination will change pixel values in the image!
  - All pixel values in the image will increase (brighter image) or decrease (darker image)
  - Comparing pixel values won't work

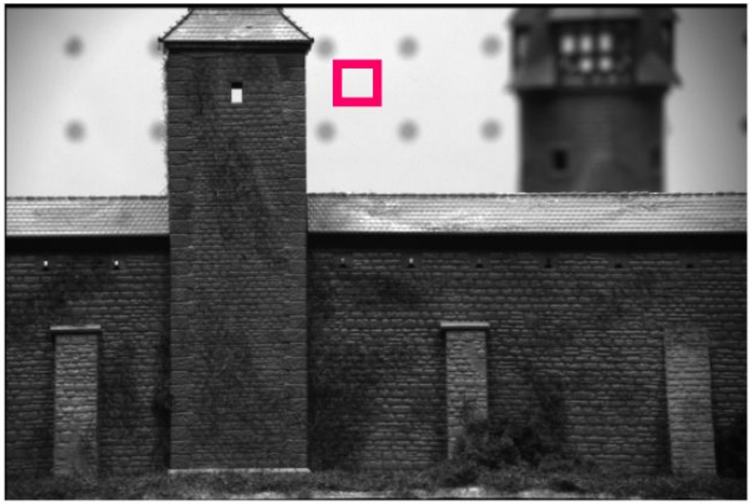
A better solution

- In the real world, you are identified by fingerprints
  - This uniquely identifies you as a person
  - You can change clothes, wear glasses, change haircut , you'll still be recognized
- We would like to do the same for patches
  - Extract a 'fingerprint' for that patch, that would remain the same in different images in which there was a change of:
    - Illumination
    - Scale
    - Viewpoint
  - That way, we could compute the fingerprint in image 1, computer the fingerprint in image 2 : if the fingerprints are the same, we're looking at the same patch!
  - Fingerprints are called **descriptors**
  - Researchers came up with different ways of obtaining them

Keypoints

- We’re not interested in describing **all** patches

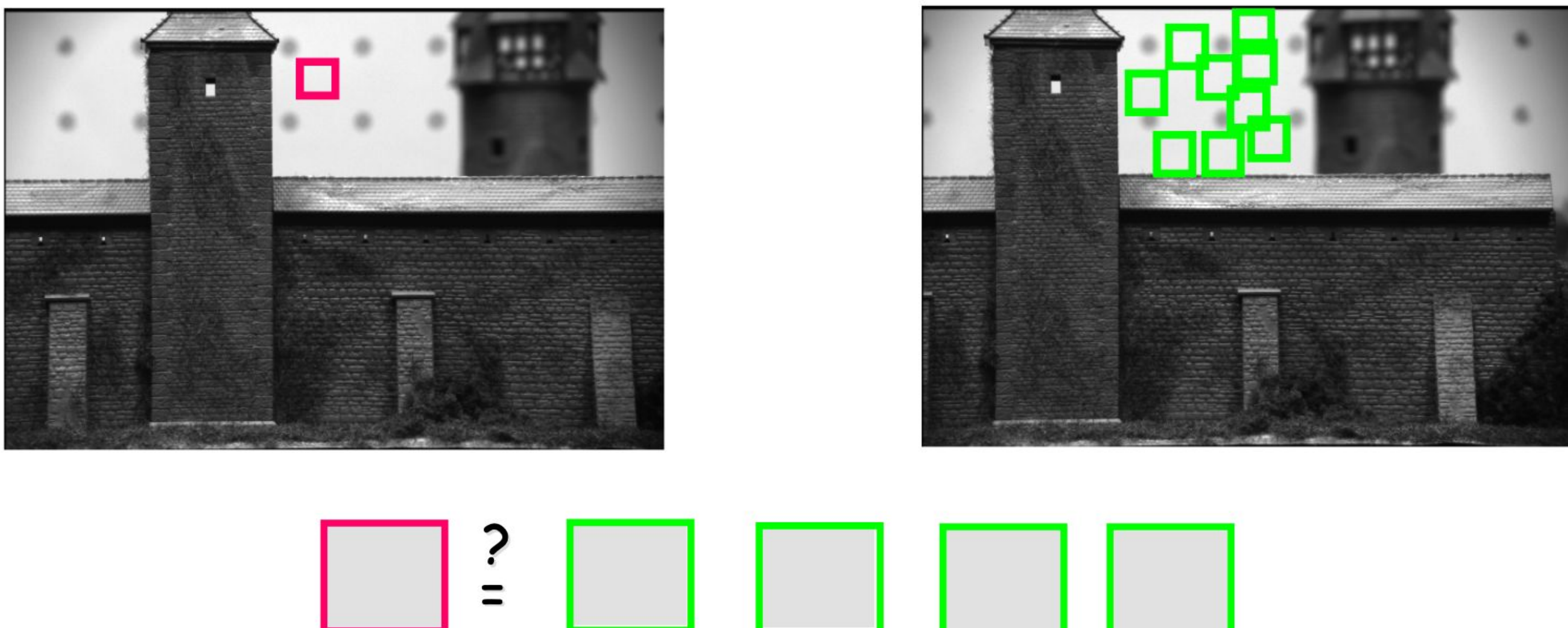
We’re interested in *characteristic* patches in the image



Keypoints

- We’re not interested in describing **all** patches

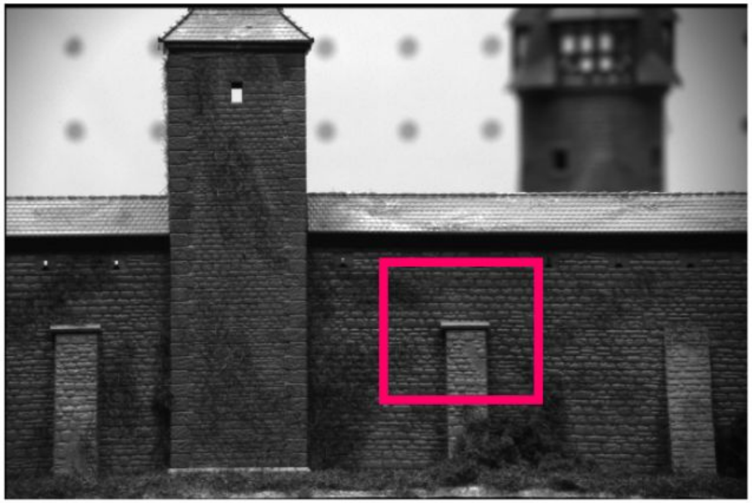
We’re interested in *characteristic* patches in the image



Keypoints

- We're not interested in describing **all** patches

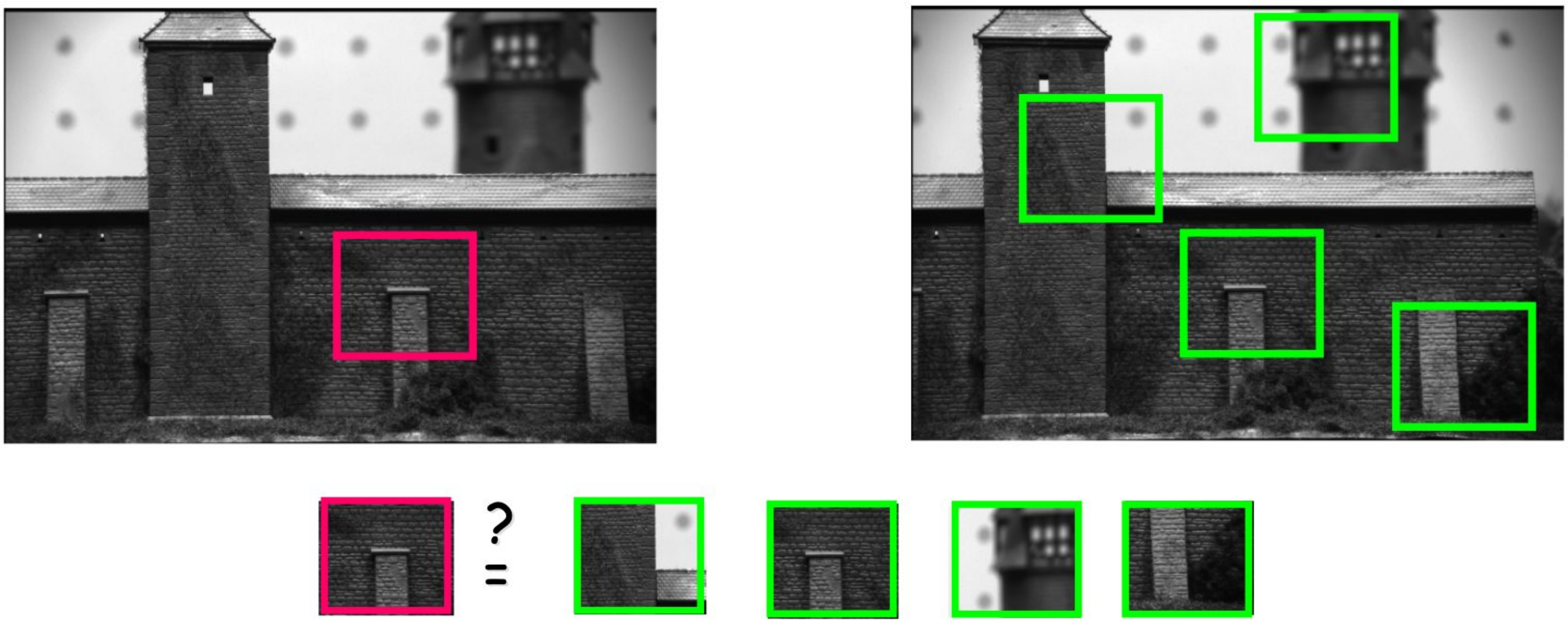
We're interested in *characteristic* patches in the image



Keypoints

- We're not interested in describing **all** patches

We're interested in *characteristic* patches in the image

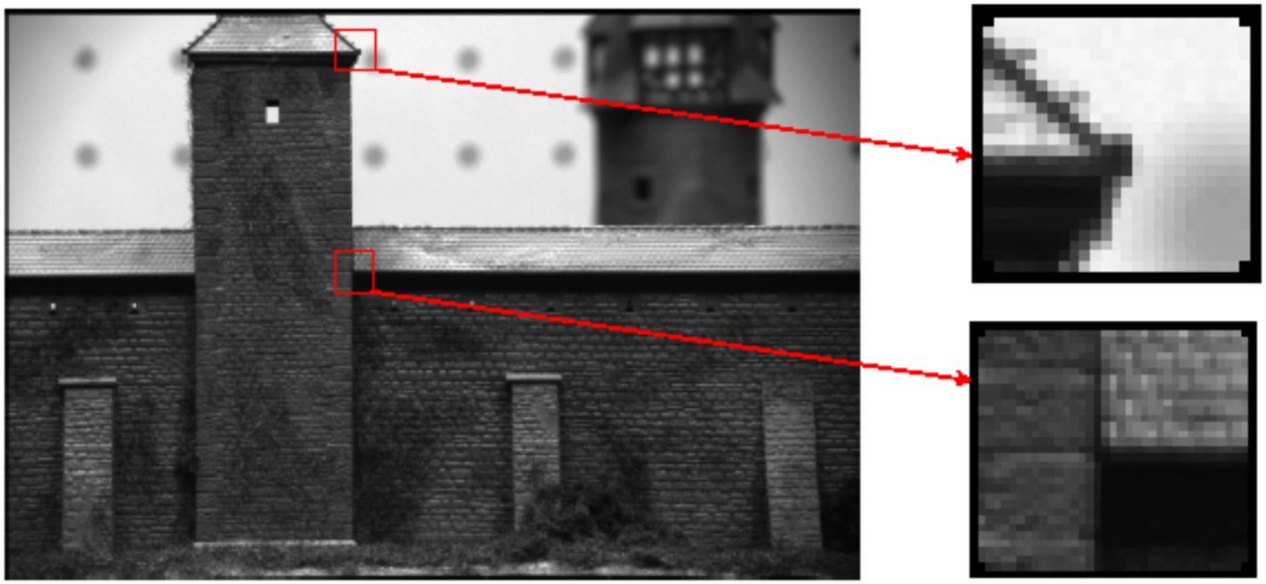


Keypoints

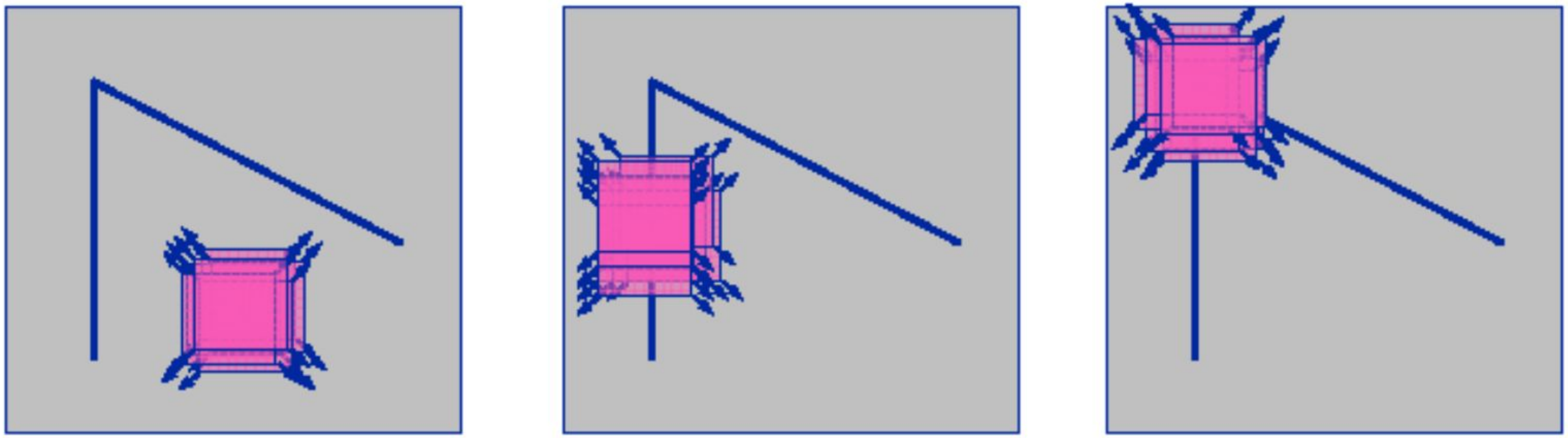
- We’re not interested in describing **all** patches

We’re interested in *characteristic* patches in the image

These patches contain a **keypoint**

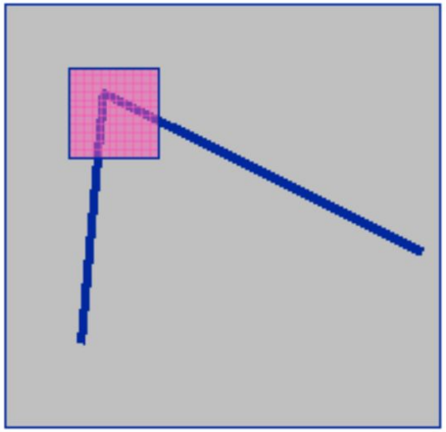


Why Corners?



Why Corners?

- How could we define a corner?
  - 2 edges in different orientations
  - 2 strong gradients in different directions



Conclusion

- Image features are the characteristic subsets of an image
- Image features are made up of:
  - A keypoint: a characteristic **location**, generally containing a corner
  - A descriptor: which describes the patch around the keypoint, and acts like a fingerprint for that patch
- Given enough image features, you can describe an image

- Comparing images can now be done without pixels by comparing their features

SIFT: a TLDR

- SIFT is a method to extract image features
- Typically, 50~200 features can be extracted from a single image
  - This depends on the image contents, the 'richer' the image, the more features can be extracted
  - A good rule of thumb for an image being rich is the number of corners
- How are SIFT features computed?
- First, we find a set of candidate keypoints
  - Maxima in the **scale-space** (image is blurred with Gaussian Kernels of different standard deviation, and downsampled)
  - These maxima can be thought of as strong corners at a particular scale
- Second, some keypoints are discarded:
  - Those that might have been created by noise
  - Those that lie on an edge
- Third, keypoints that don't have a strong **orientation** are discarded
  - The orientation is the mode of the distributions of gradient in the patch
- Fourth, the descriptor is computed
  - It is a histogram of gradients, in different subsets of the patch
  - The full descriptor is 128 dimensional

Some Code for image processing

Laplacian Kernel:

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Gaussian Kernel:

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Prewitt and Sobel Kernels:

Image Neighbourhood

a_1	a_2	a_3
a_4		a_6
a_7	a_8	a_9

Sobel, G_x

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

Sobel, G_y

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

Prewitt, G_x

-1	-1	-1
0	0	0
1	1	1

Prewitt, G_y

-1	0	1
-1	0	-1
-1	0	-1

From image features to Bag-Of-Words

A SIFT descriptor is a way to recognize the same point **in the scene** (3D) from different **images** (2D) with different scale, illumination, viewpoint

- It describes an image **patch**
- A single SIFT feature is not enough to represent an image - it just contains local information
  - How do we represent the entire image with SIFT?
  - Take a lot of image features! Within a single image, extract hundreds of SIFT features
- Using this, how do we compare images?
  - If most of their features match, the images are probably related

Bag-Of-Words

- Words here are “visual”
  - BOW actually comes from Natural Language Processing, text documents would be represented by their frequency of use of certain words
  - With images, “words” are image features (eg: obtained through SIFT) - ie ‘iconic’ image patches

- We do not want them to be ‘ordered’
  - The order in which the features appear/are detected should not matter
  - Therefore all the features are recorded in random order, like you would place them in a bag

A Natural Language Processing Idea

- Let's see how this works in NLP
  - Where computer vision deals with images, NLP deals with text
- How could you classify text documents?
  - Let's say, between economic papers and a car magazine ...
- First, look at all documents that are available, and extract the most 'meaningful' words
  - These are the words that appear the most
  - GDP, engine, horsepower, budget, driving, Dow-Jones, bitcoin ...
- These meaningful words create a **vocabulary**

A Natural Language Processing Idea

- Second, for each document, count how many vocabulary words are used
    - You go through all documents *again*, and just record which words are used (every time a vocabulary word is spotted, add one to a counter)
    - You end up with a histogram **for each document**
        - Describing how that document used the vocabulary

Now if I tell you:

- 1.txt => GDP:0, Engine:3, Horsepower:7,budget:2,Dow-J:0,driving:8
- 2.txt=> GDP:13,Engine:3,Horsepoer:0,budget:8,Dow-J:5,driving:3

A Natural Language Processing Idea

- Second, for each document, count how many vocabulary words are used
  - You go through all documents *again*, and just record which words are used (every time a vocabulary word is spotted, add one to a counter)
  - You end up with a histogram **for each document**
    - Describing how that document used the vocabulary

Now if I tell you:

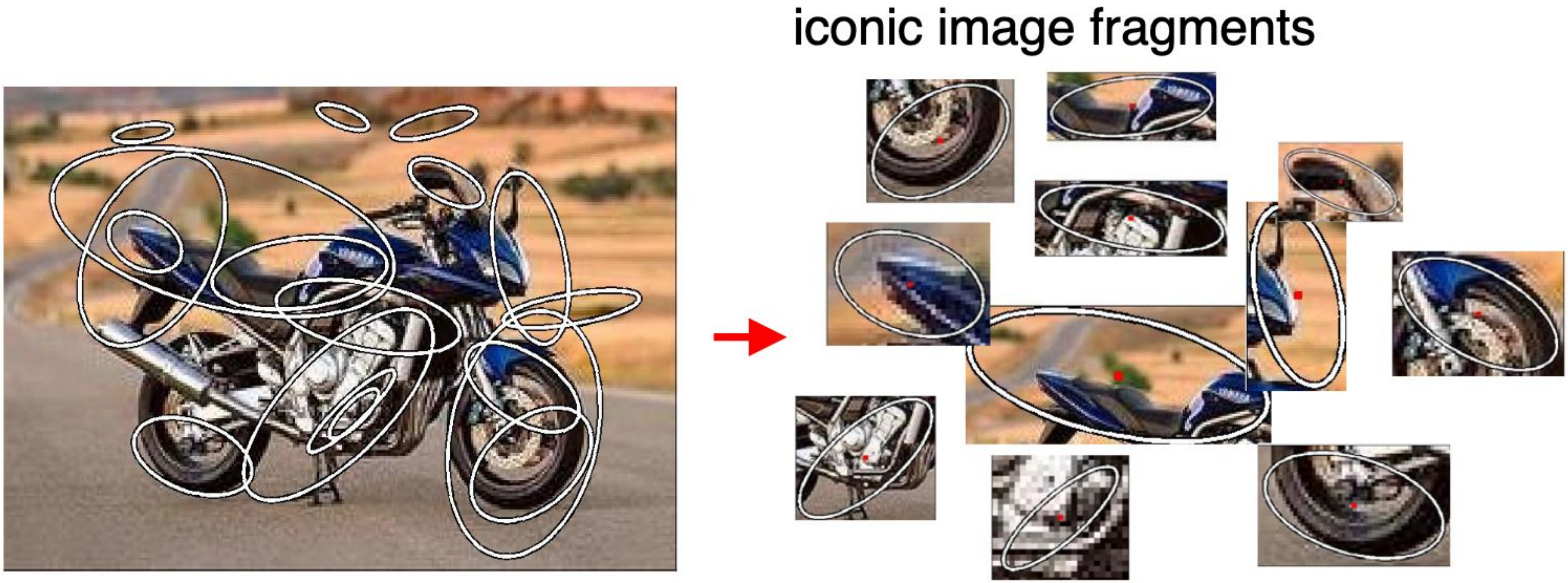
- 1.txt => GDP:0, Engine:3, Horsepower:7,budget:2,Dow-J:0,driving:8
- 2.txt=> GDP:13,Engine:3,Horsepoer:0,budget:8,Dow-J:5,driving:3
- 1 was in a car magazine, 2 in an economic journal!

- Why is it called a 'Bag' ?
  - That representation of the document doesn't depend on the order of appearance of the words in the document!
    - They are all thrown, unordered, in a 'bag'

NLP to computer vision

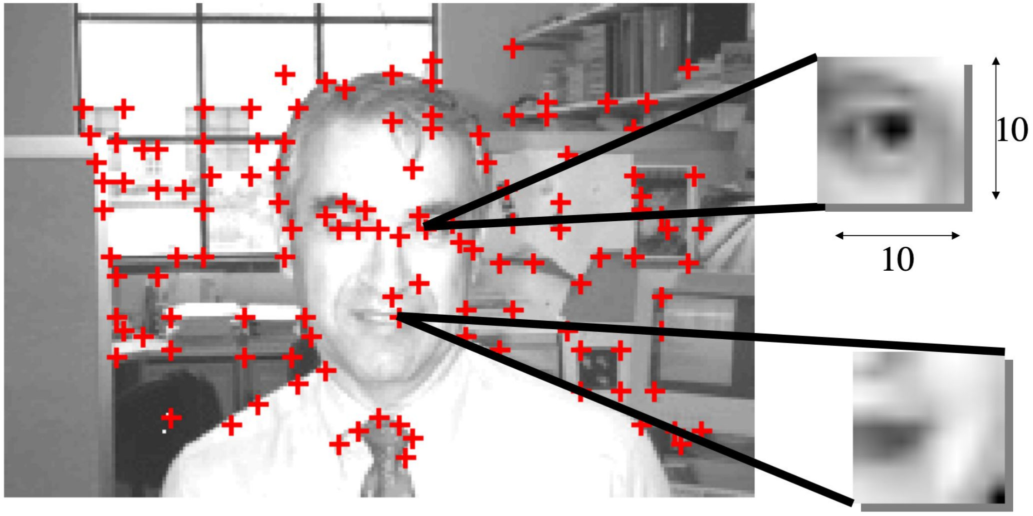
- The idea is the same in CV
  - Replace ‘word’ with ‘image feature’ and ‘text document’ with ‘image’
- First, extract features from all images at your disposal
- Second, **cluster** them to find the most representative image features => vocabulary
- Third, represent your training set as a histogram of vocabulary words, and train a classifier
  - The classifier learns what the histogram of a particular class looks like
- To test => feed histograms representing your test set to the classifier

In practice



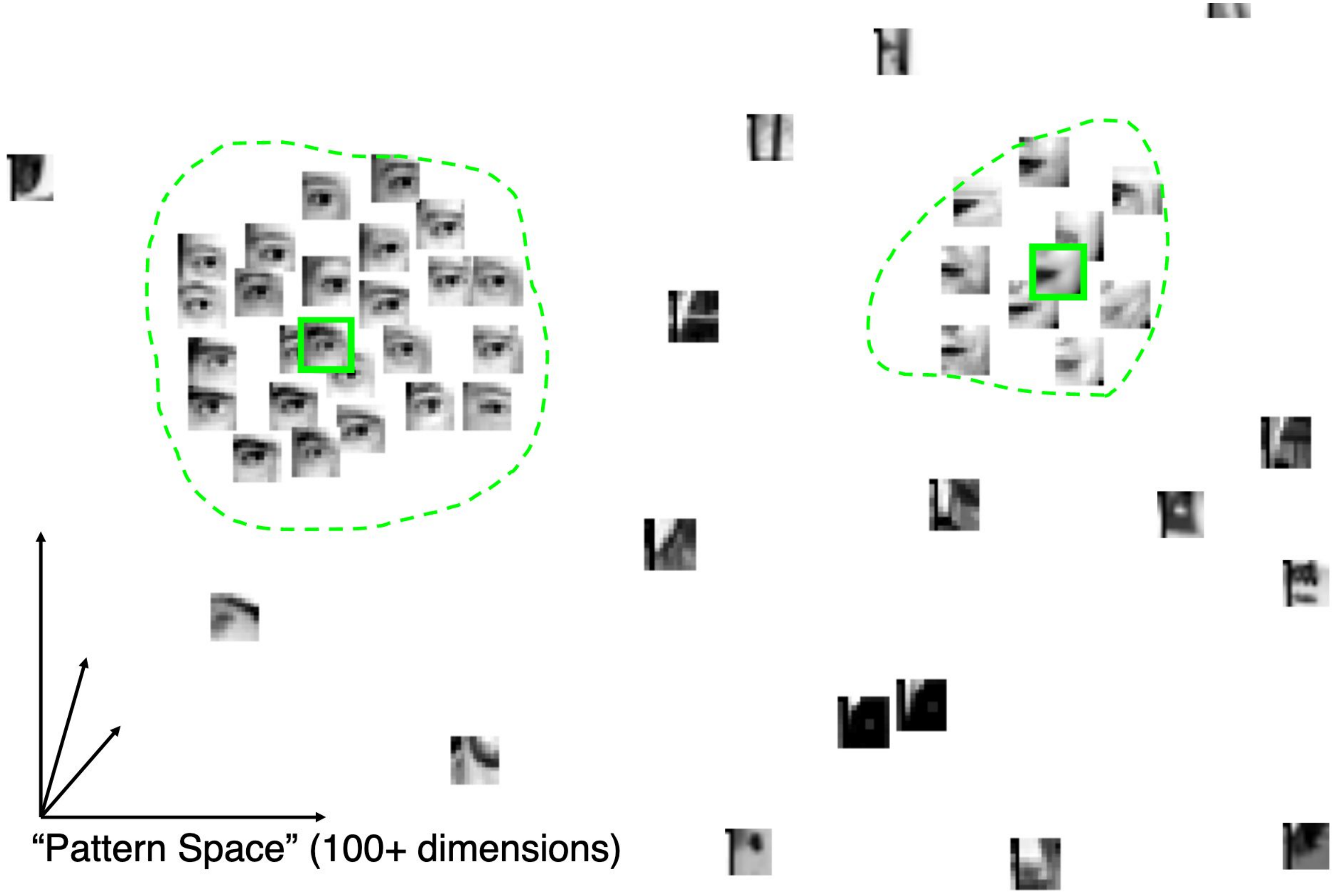
Vocabulary

- You will build a set of features that you can compare every image against
  - This is your **vocabulary**
- To do so, you will extract 100’s of features from different images, and **cluster** them together
  - That way your vocabulary will contain only relevant features that occur frequently
- An example with faces:

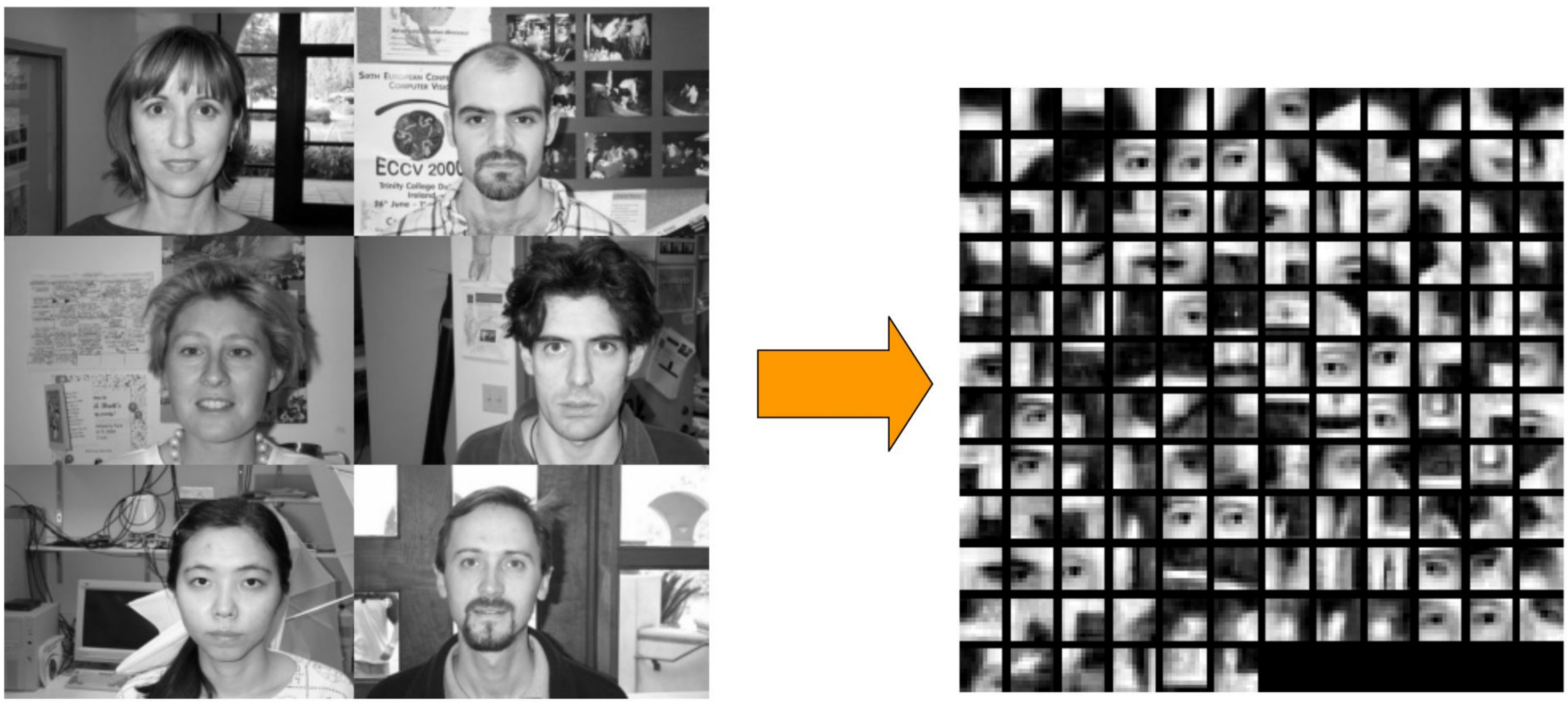


Vocabulary

I can do the same process for several images:



Vocabulary: representation

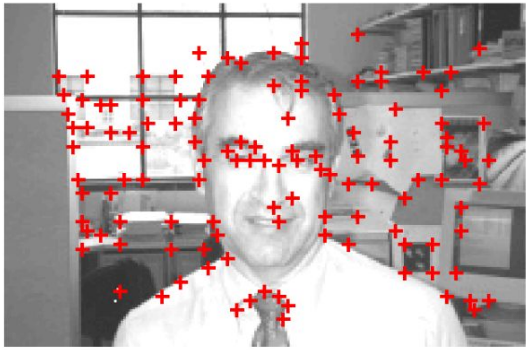


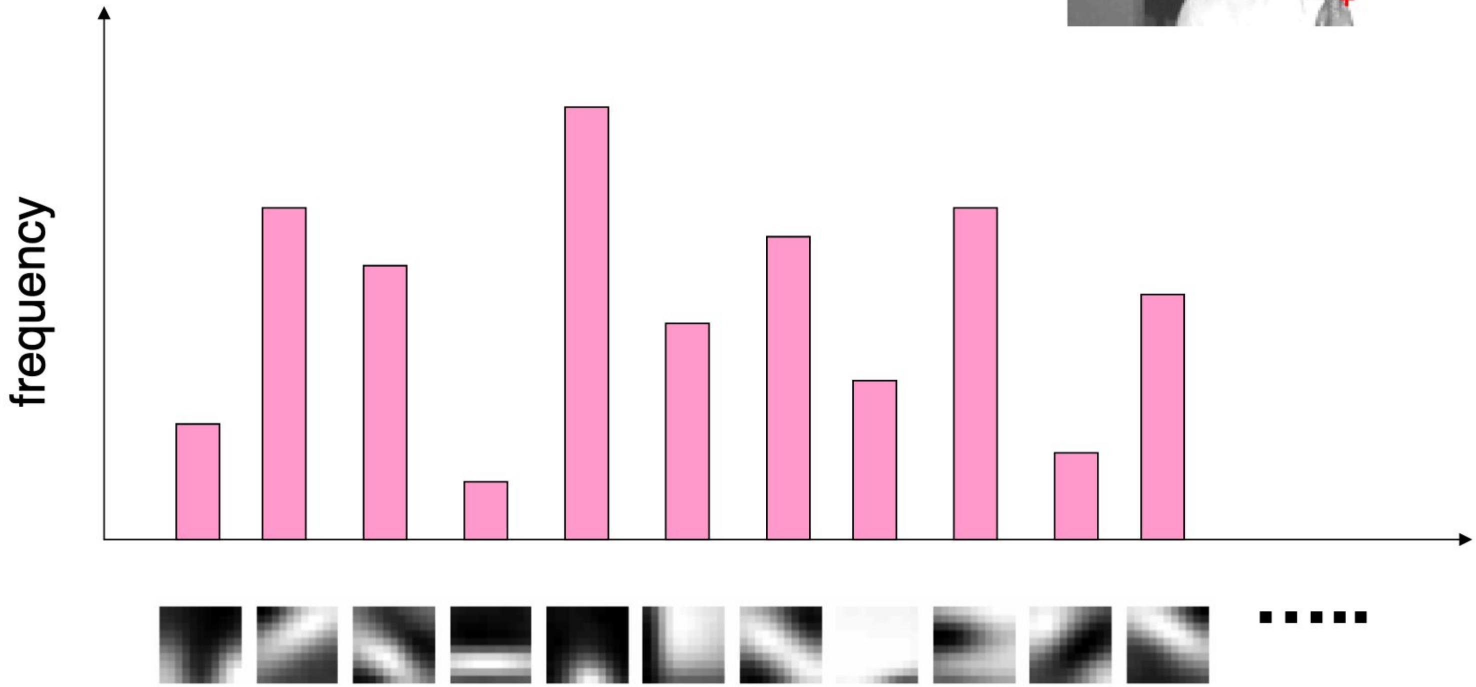
100-1000 images

~100 visual words

The BOW representation of an image

- detect interest point features
- find closest visual word to region around detected points
- record number of occurrences, but not position





How is this useful for classification?

- For your homework, you have categories like streets and forests
- Step 1: Represent all images in your training set using the BOW model
  - Your image will now be a vector of features
  - Hopefully, pictures of streets will share a lot of similar features, that are different from the features forests share
- Step 2: Train a classifier
  - What does the histogram of a street picture look like? What does the histogram of a picture of a forest look like? They won't be represented by the same features
- Step 3: Apply the classifier to the test image
  - Extract its BOW representation
  - Does its histogram look like that of a forest, or a street?