Computer Vision Course
Lecture 04

Template Matching
Image Pyramids

Ceyhun Burak Akgül, PhD
cba-research.com

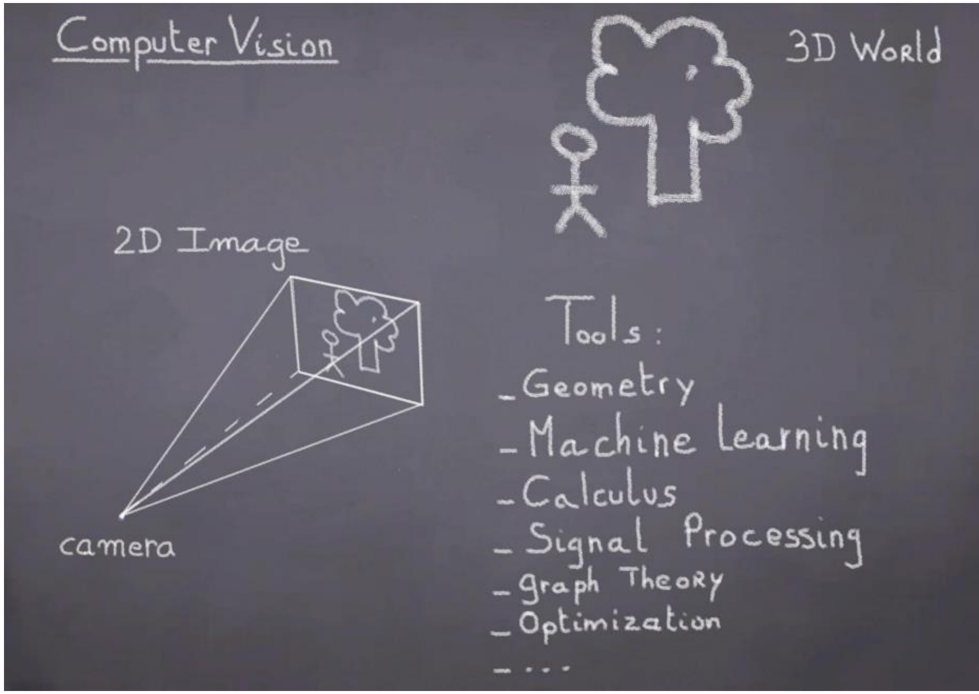


Photo credit: Olivier Teboul
vision.mas.ecp.fr/Personnel/teboul

Spring 2015
Last updated 11/03/2015

These slides have been adapted from James Hays's 2014 Computer Vision course slides at Brown University.

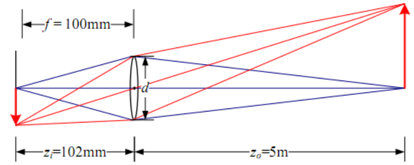
Course Outline

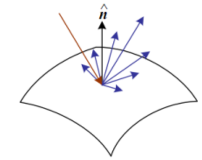
Image Formation and Processing

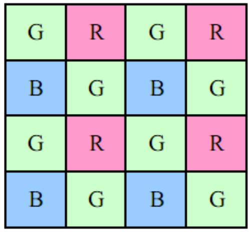
Light, Shape and Color

The Pin-hole Camera Model, The Digital Camera

Linear filtering, Template Matching, Image Pyramids







Feature Detection and Matching

Edge Detection, Interest Points: Corners and Blobs

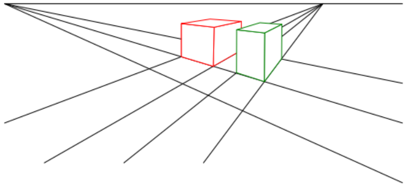
Local Image Descriptors

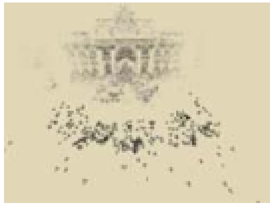
Feature Matching and Hough Transform

Multiple Views and Motion

Geometric Transformations, Camera Calibration

Feature Tracking , Stereo Vision





Segmentation and Grouping

Segmentation by Clustering, Region Merging and Growing

Advanced Methods Overview: Active Contours, Level-Sets, Graph-Theoretic Methods

Detection and Recognition

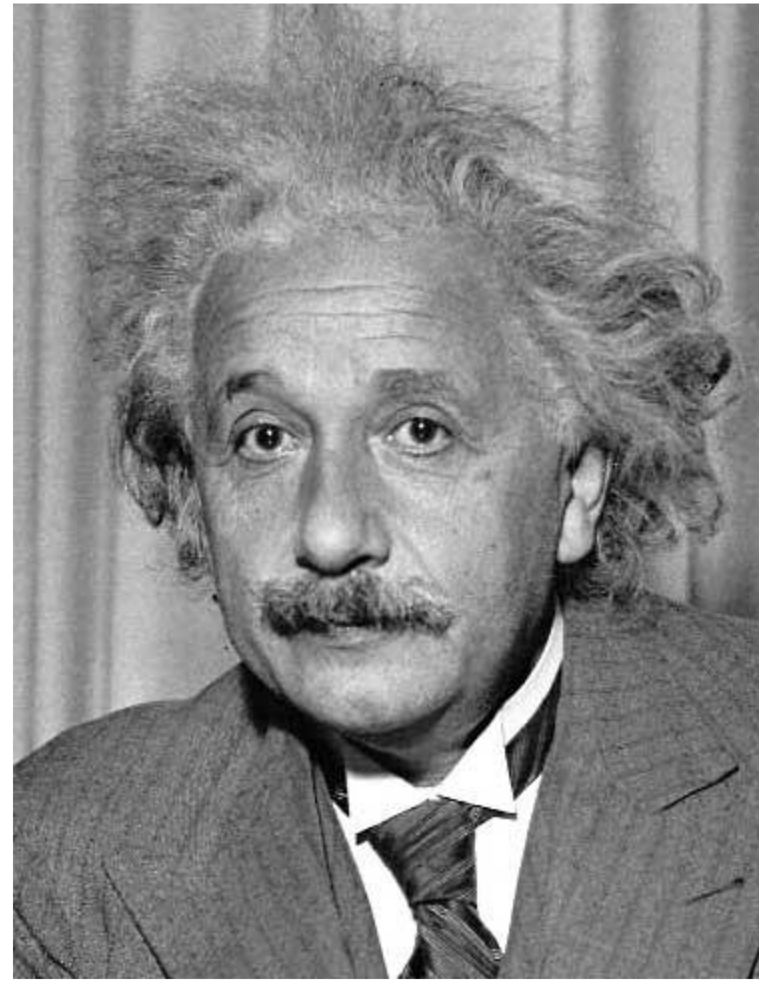
Problems and Architectures Overview

Statistical Classifiers, Bag-of-Words Model, Detection by Sliding Windows

Template Matching

- Goal: find  in image

- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation

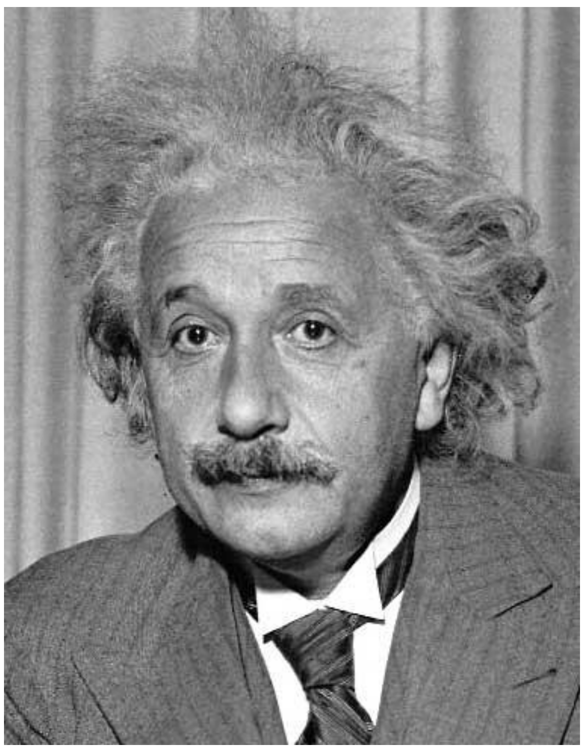


Matching with Filters

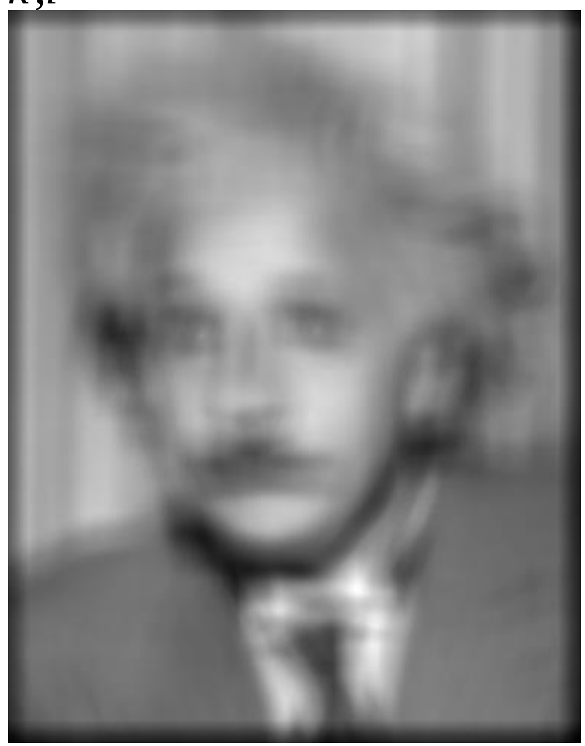
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

f = image
g = filter



Input



Filtered Image

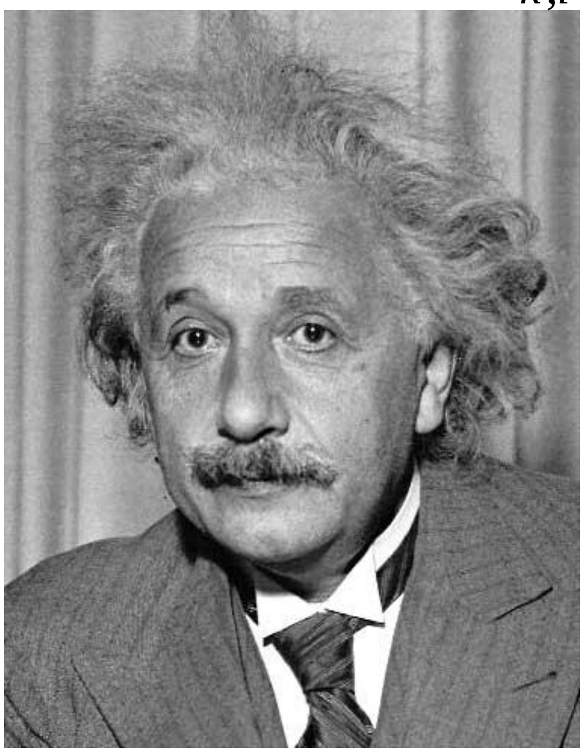
What went wrong?

Matching with Filters

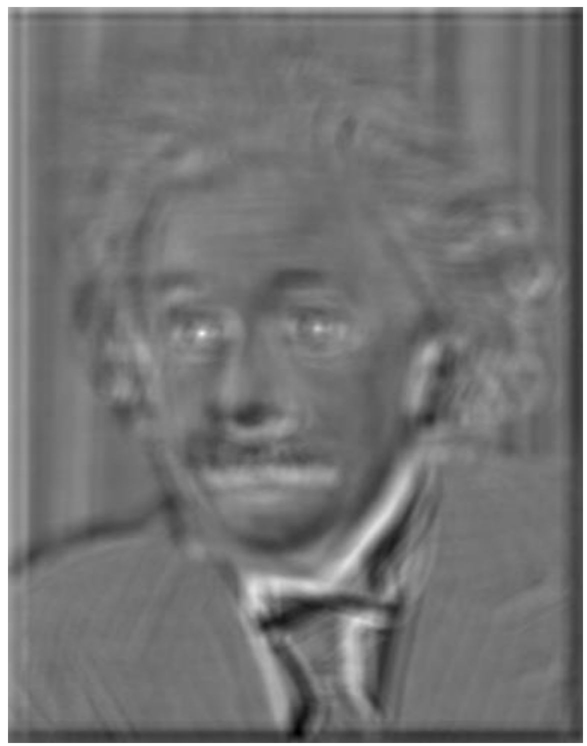
- Goal: find  in image
- Method 1: filter the image with zero-mean eye

$$h[m,n]=\sum_{k,l}(f[k,l]-\bar{f})\,(g[m+k,n+l])$$

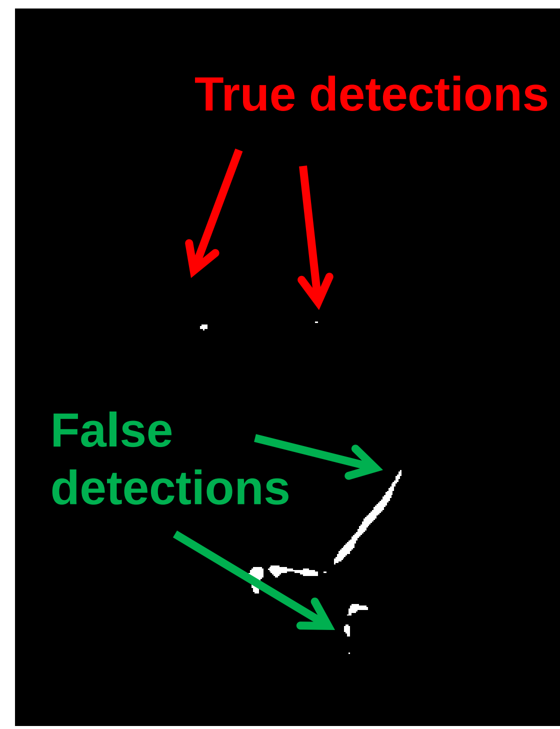
mean of f



Input



Filtered Image (scaled)

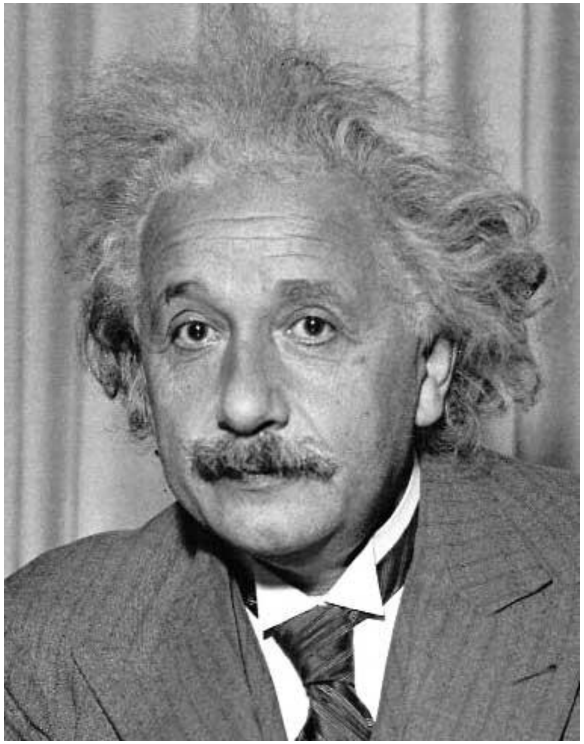


Thresholded Image

Matching with Filters

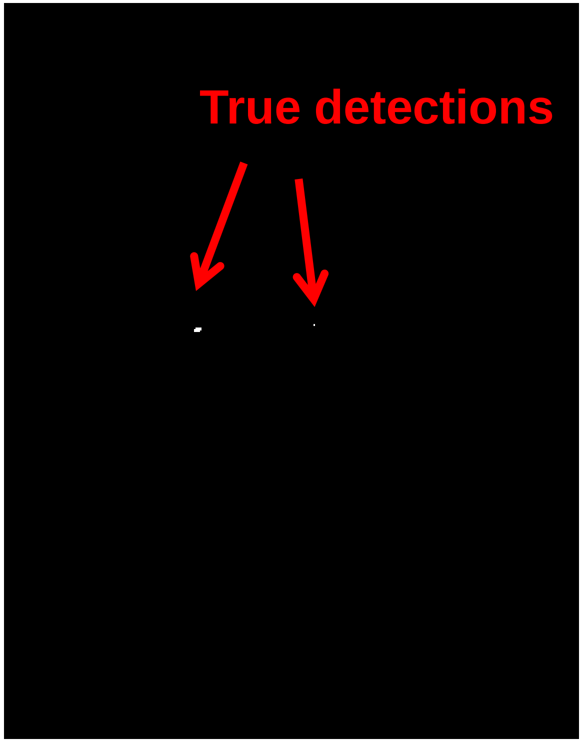
- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input

1- sqrt(SSD)



Thresholded Image

Matching with Filters

Can SSD be implemented with linear filters?

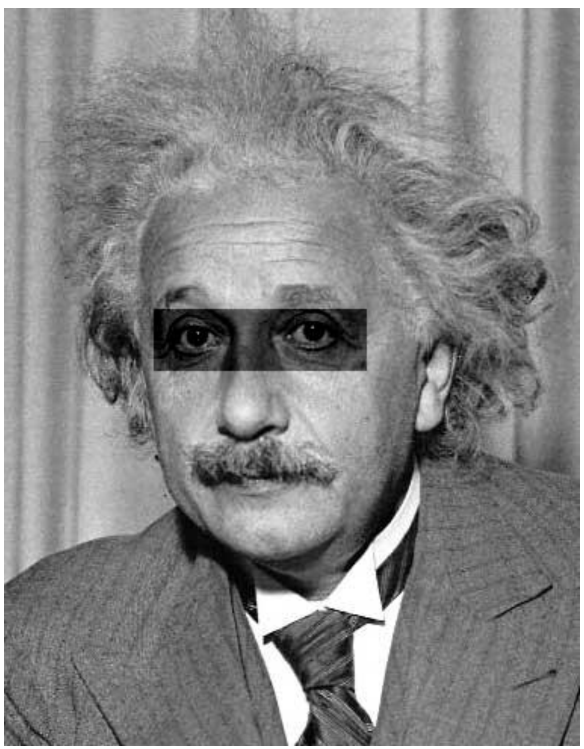
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

Matching with Filters

- Goal: find  in image
- Method 2: SSD

What's the potential downside of SSD?

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)

Matching with Filters

- Goal: find  in image
- Method 3: Normalized cross-correlation

mean template

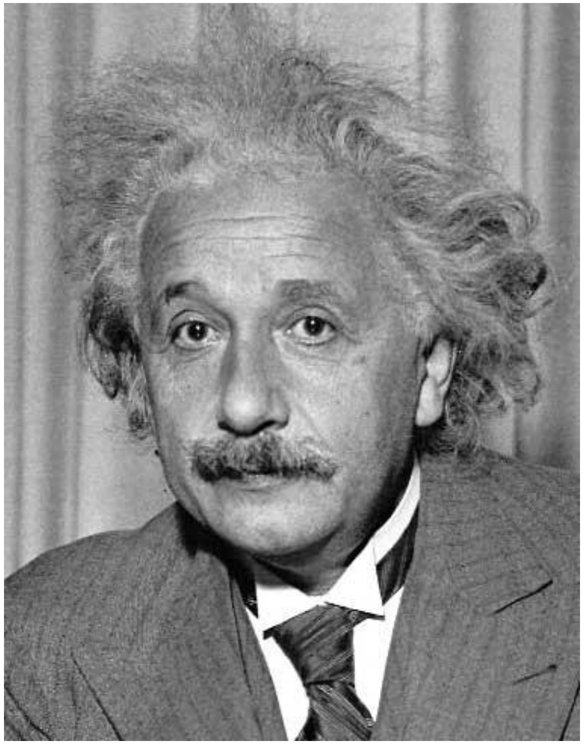
mean image patch

$$h[m,n]=\frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2\sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Matlab: `normxcorr2(template, im)`

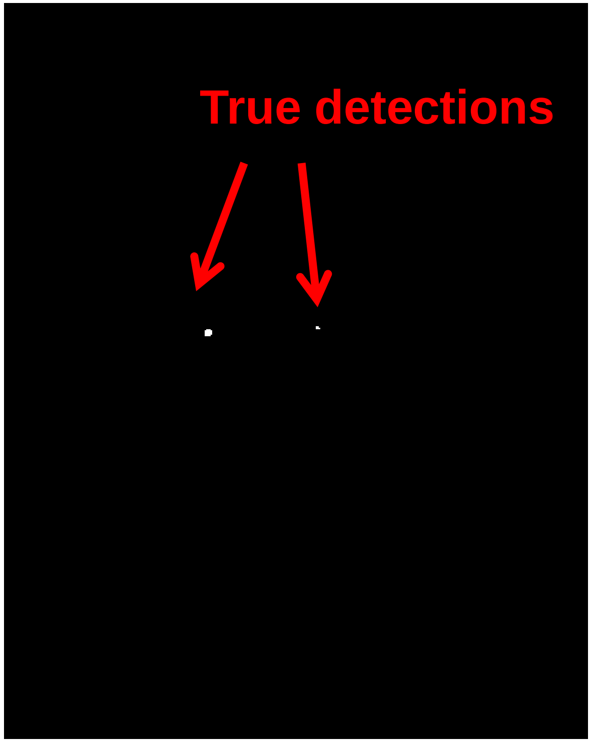
Matching with Filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

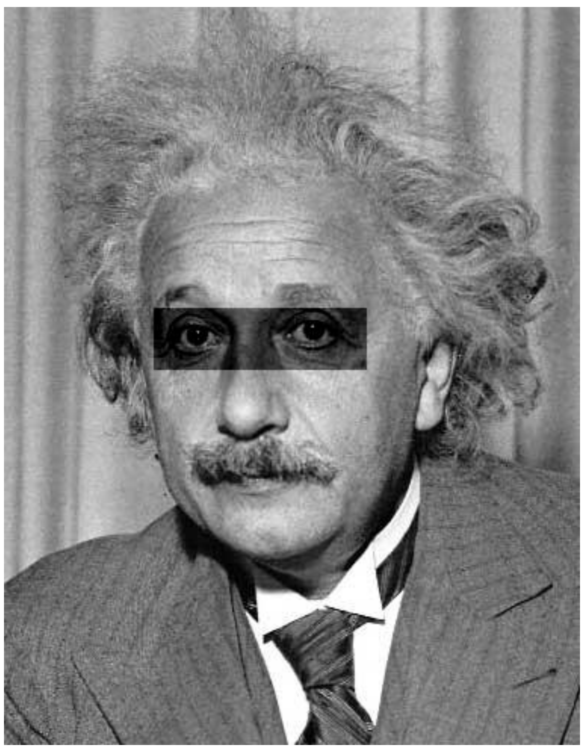
Normalized X-Correlation



Thresholded Image

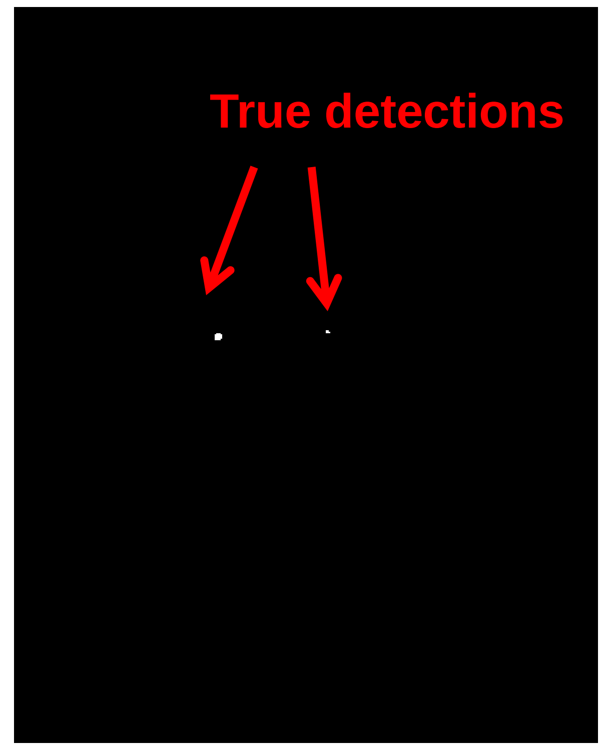
Matching with Filters

- Goal: find  in image
- Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



Thresholded Image

Q: What is the best method to use?

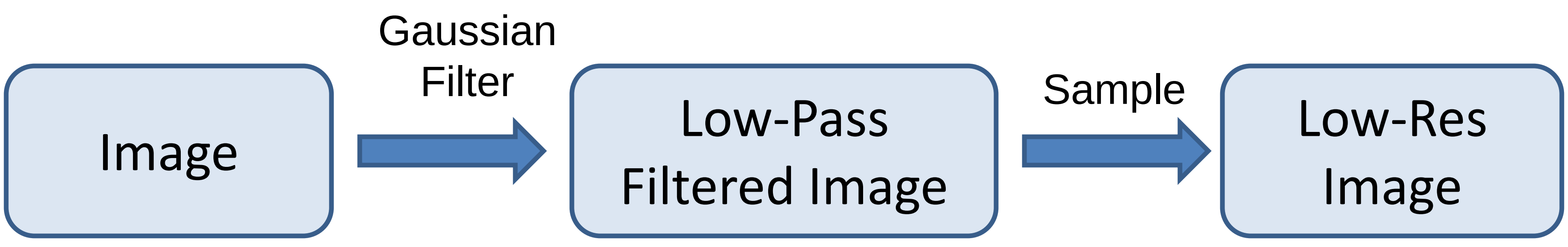
A: Depends

- SSD: faster, sensitive to overall intensity
- Normalized cross-correlation: slower, invariant to local average intensity and contrast
- But really, neither of these baselines are representative of modern recognition.

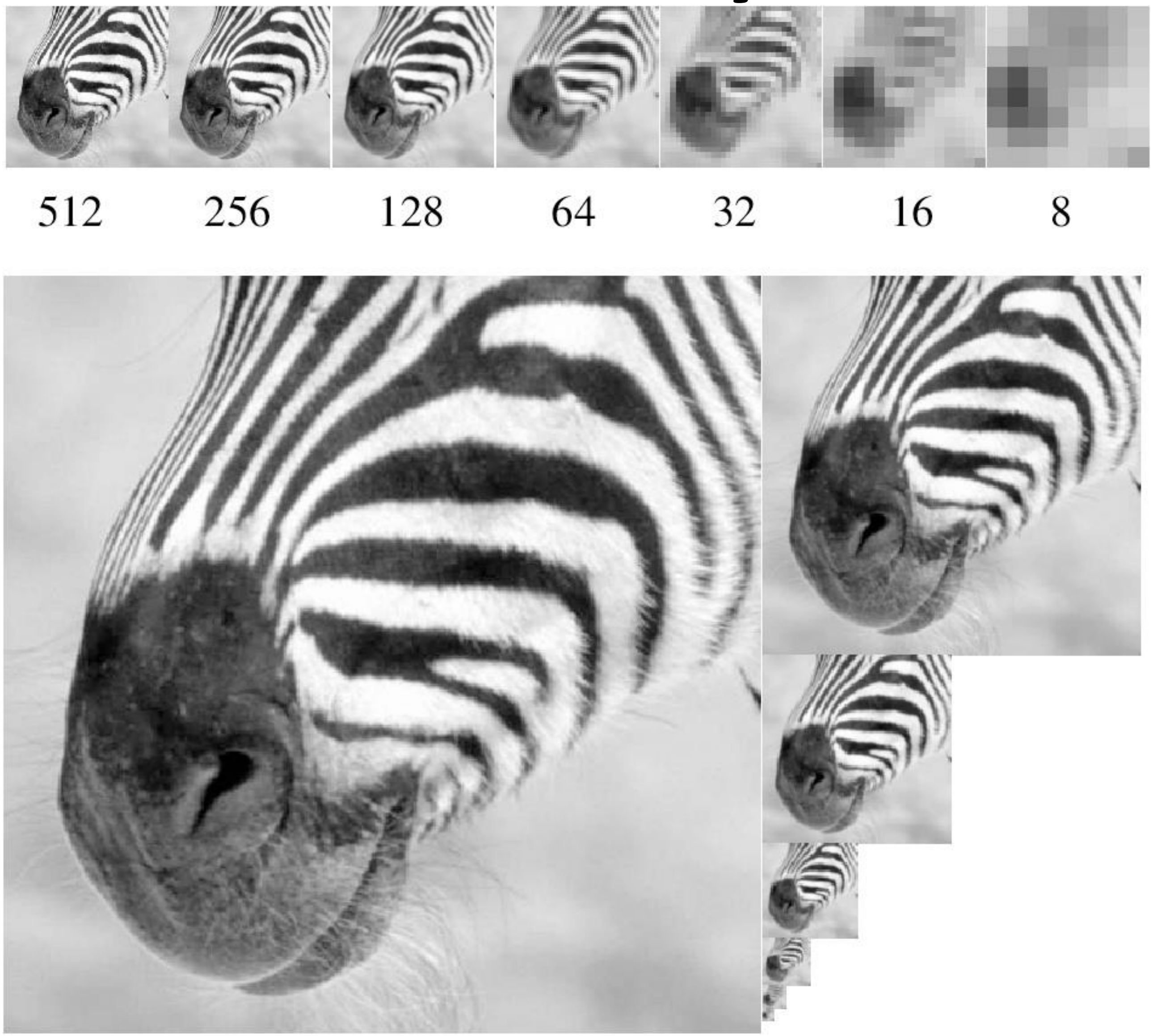
Q: What if we want to find larger or smaller eyes?

A: Image Pyramid

Review of Sampling



Gaussian Pyramid



Source: Forsyth

Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale

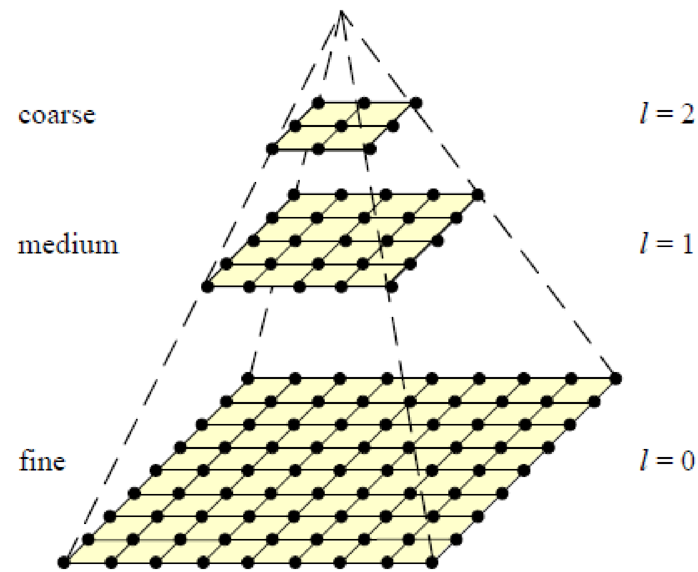
2. Downsample image

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression

Coarse-to-Fine Image Registration

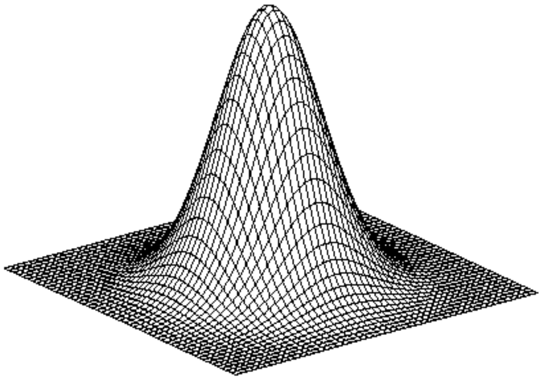
1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range



Why is this faster?

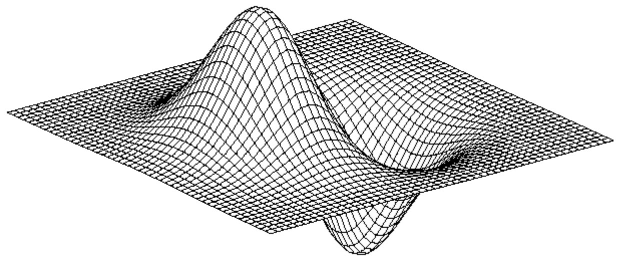
Are we guaranteed to get the same result?

2D Edge Detection Filters



Gaussian

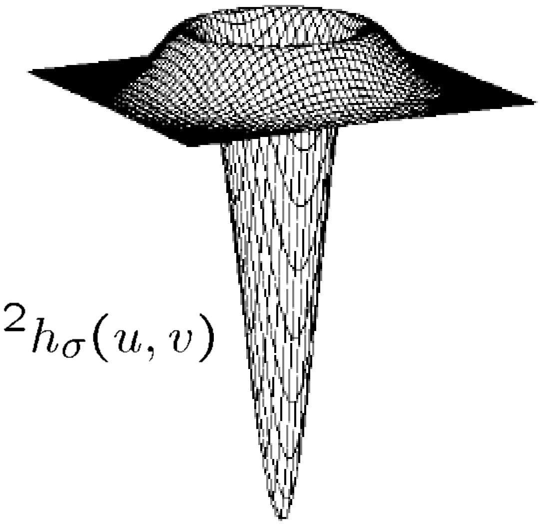
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

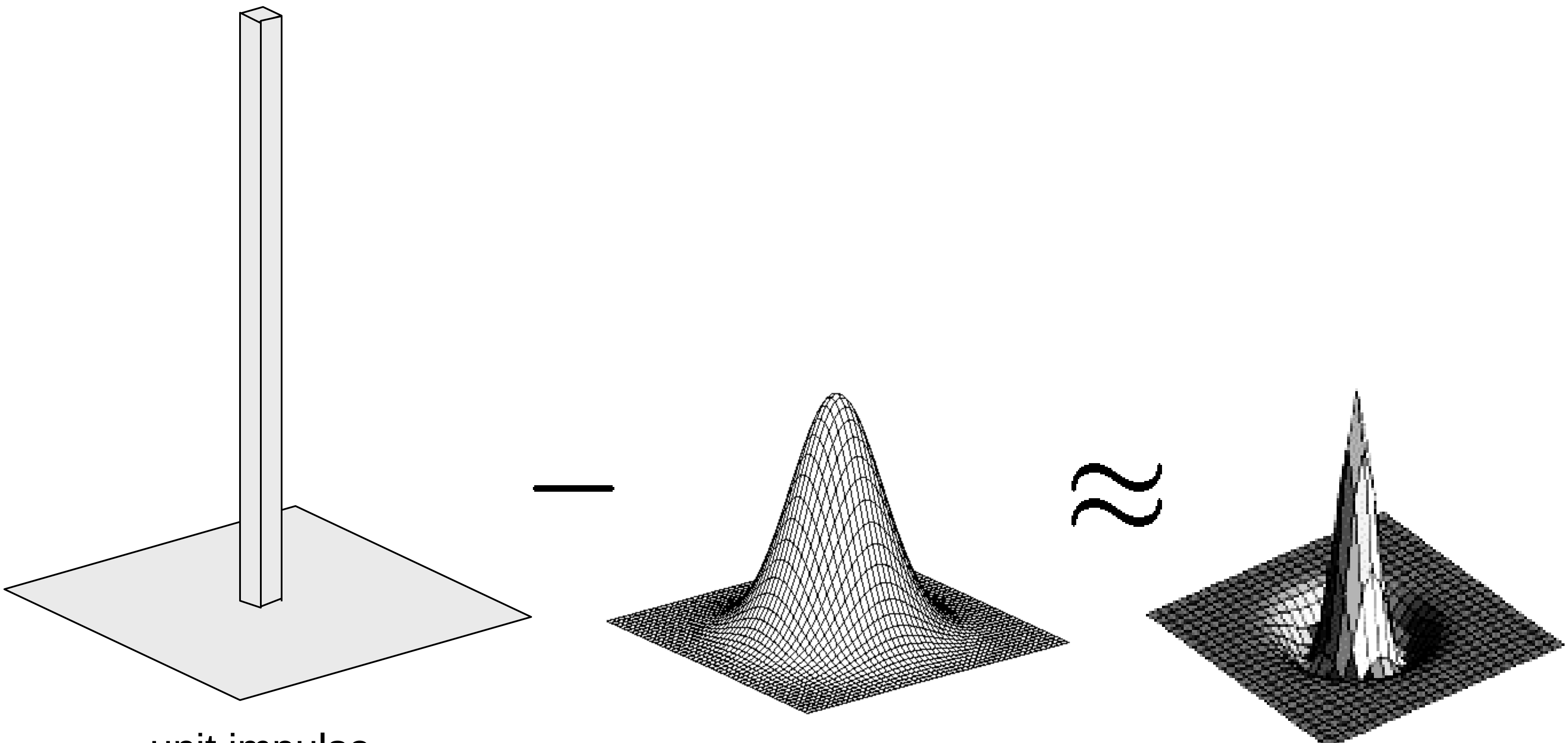


$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian Filter



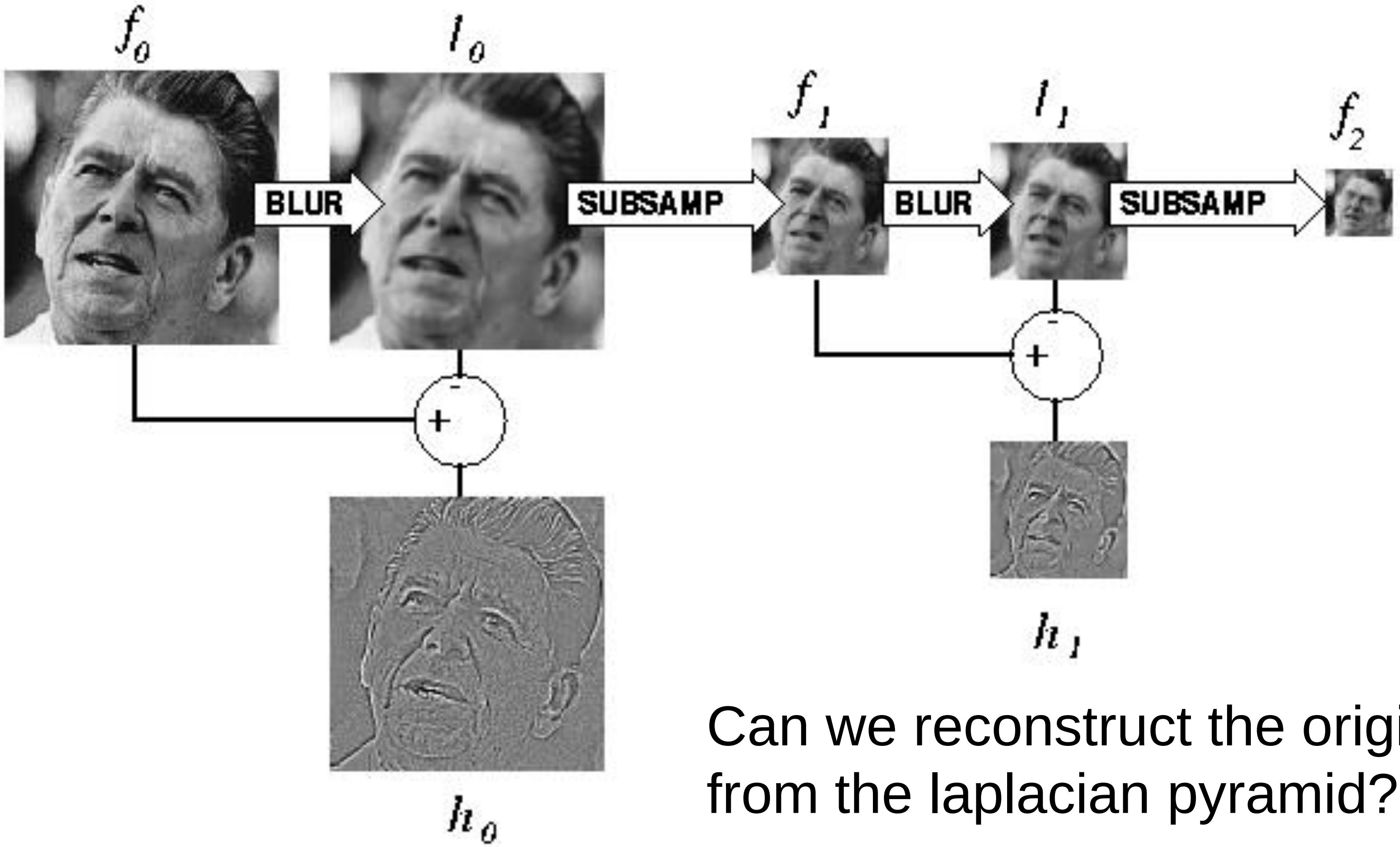
unit impulse

Gaussian

Laplacian of Gaussian

Source: Lazebnik

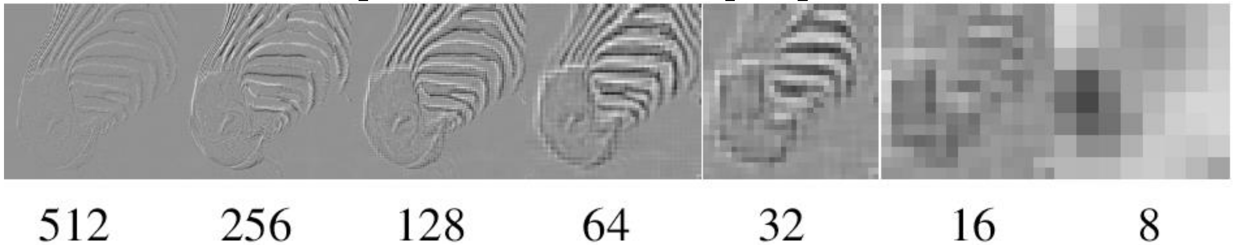
Computing Gaussian/Laplacian Pyramid

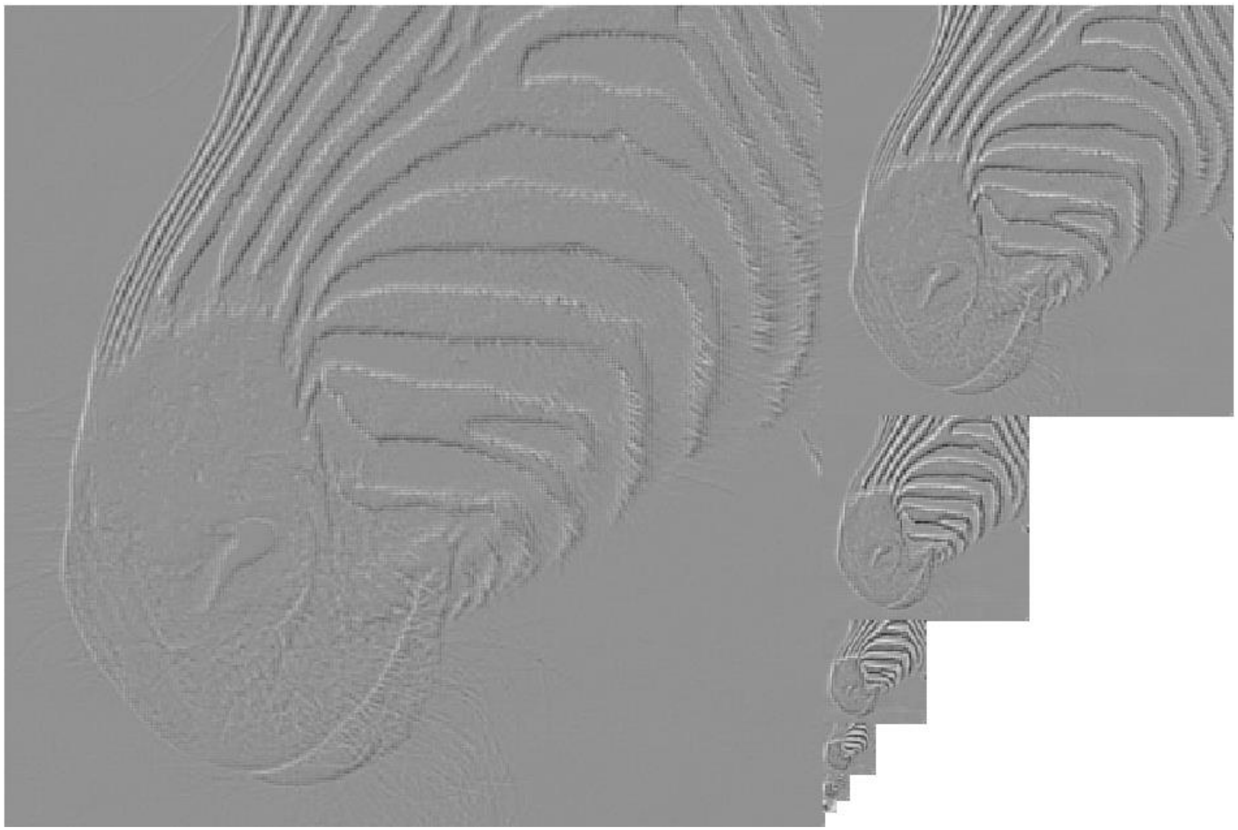


Can we reconstruct the original from the laplacian pyramid?

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Laplacian pyramid





Source: Forsyth

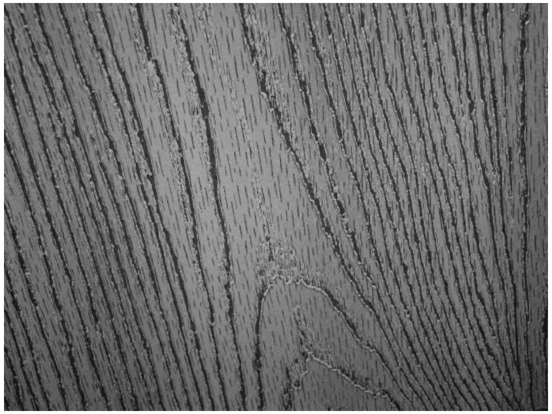
Major uses of image pyramids

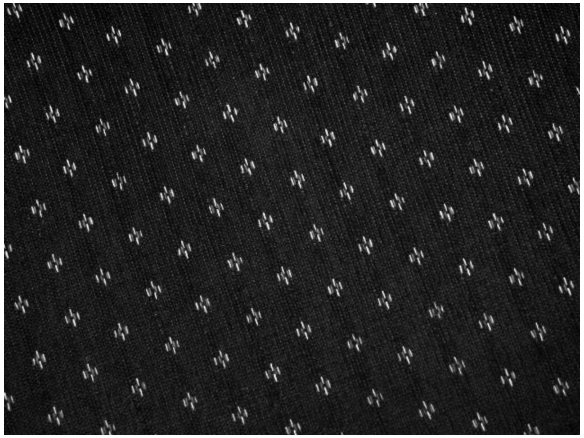
- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Course-to-fine

Application: Representing Texture

Source: Forsyth

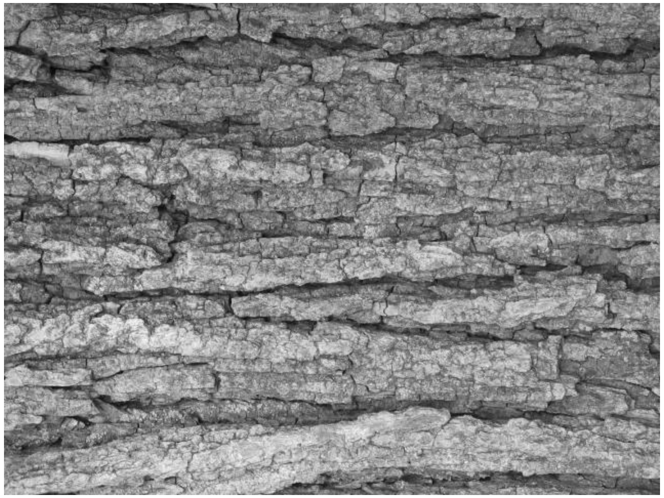
Texture and Material





http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation



http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

What is texture?

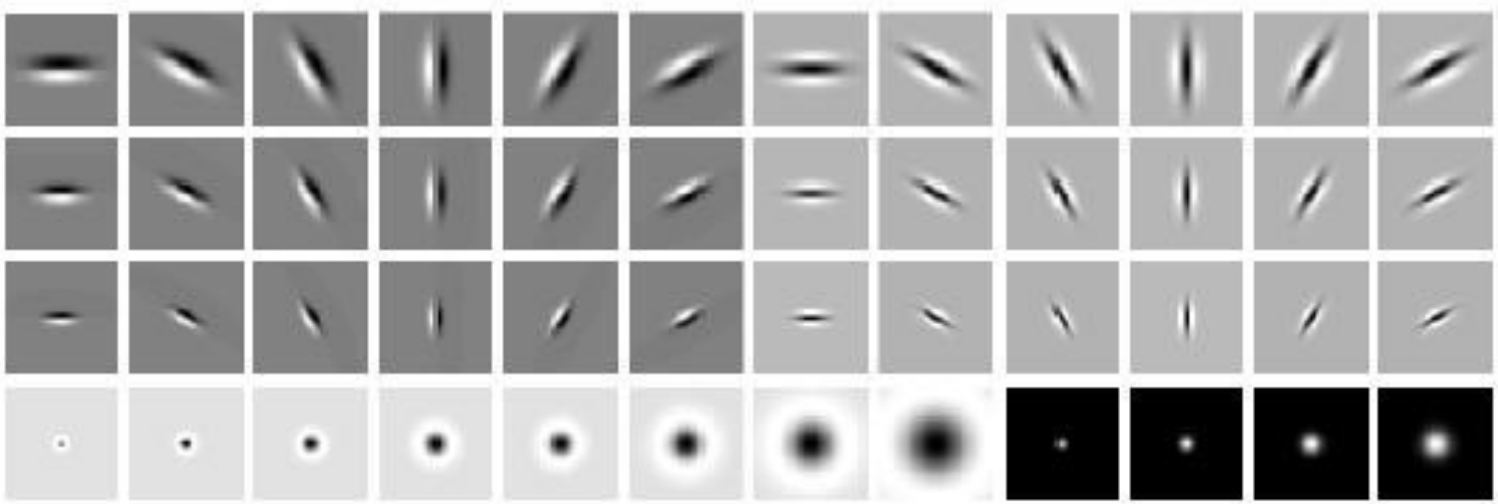
Regular or stochastic patterns caused by bumps, grooves, and/or markings

How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales

Overcomplete representation: filter banks

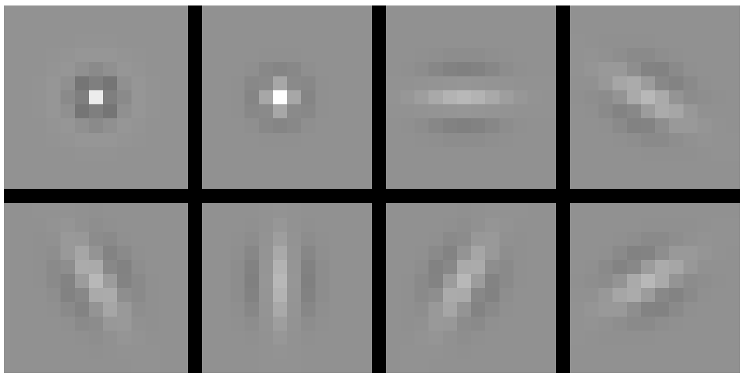
LM Filter Bank

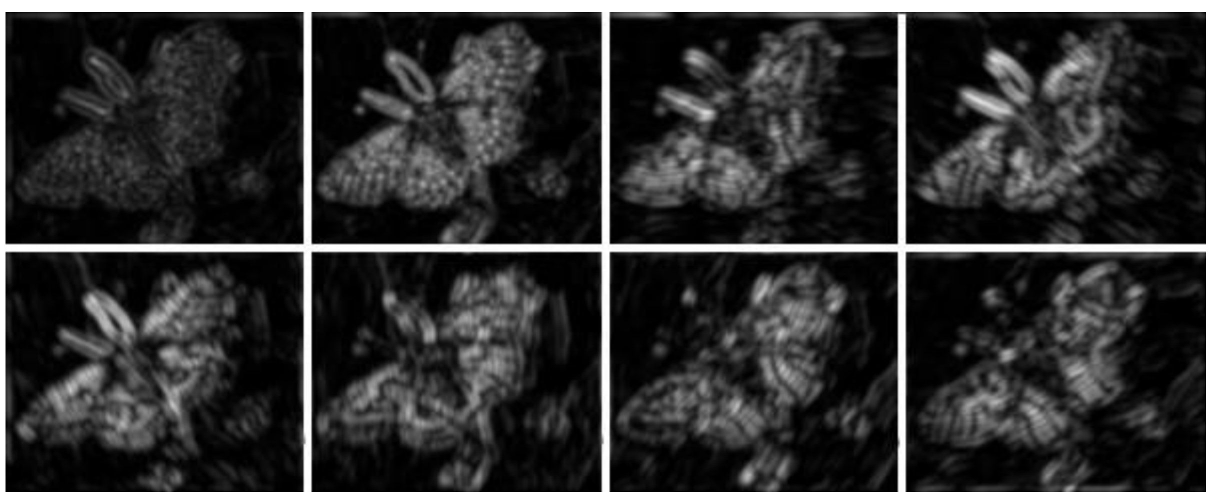


Code for filter banks: *www.robots.ox.ac.uk/~vgg/research/texclass/filters.html*

Filter banks

- Process image with each filter and keep responses (or squared/abs responses)



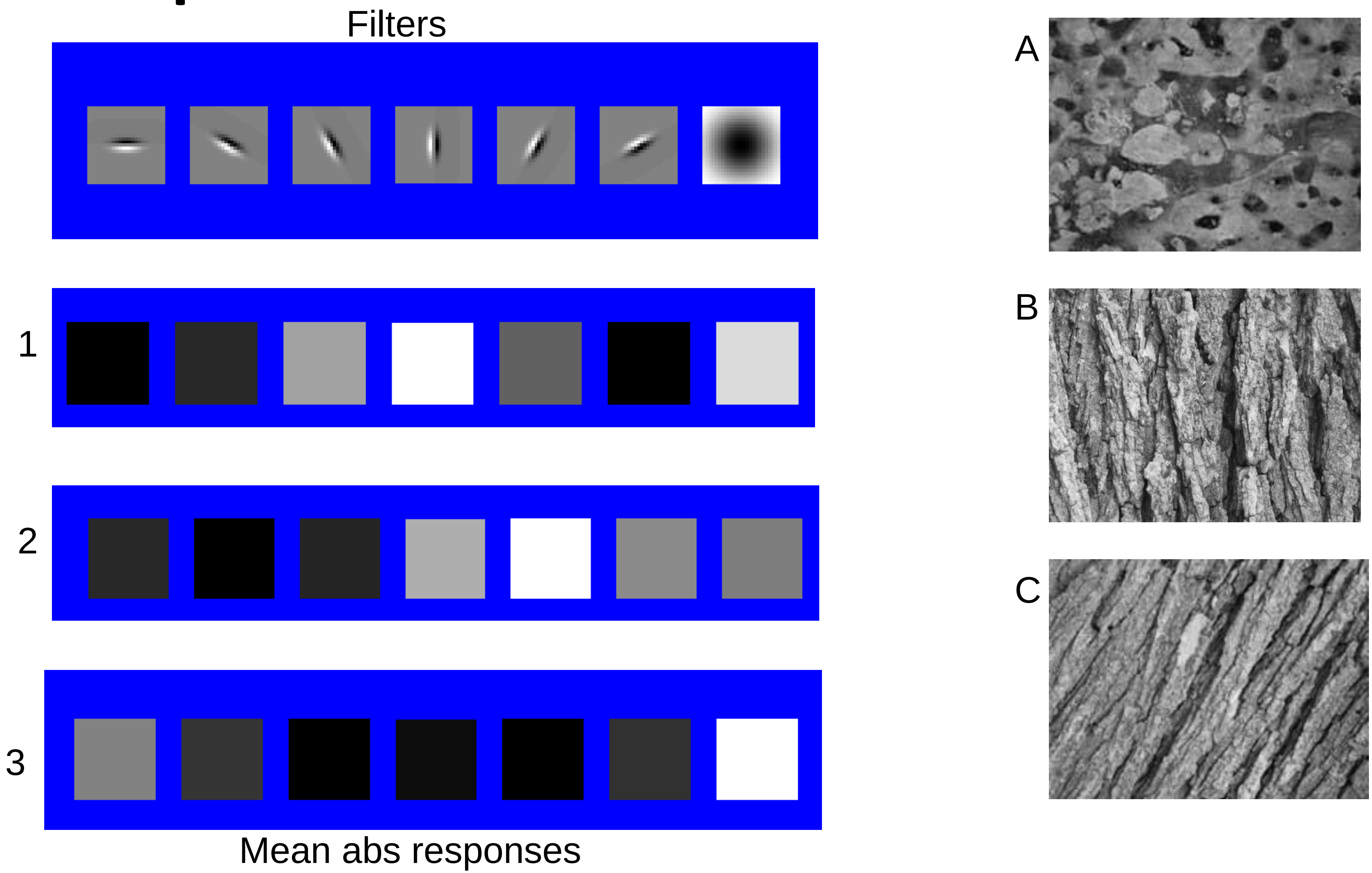


How can we represent texture?

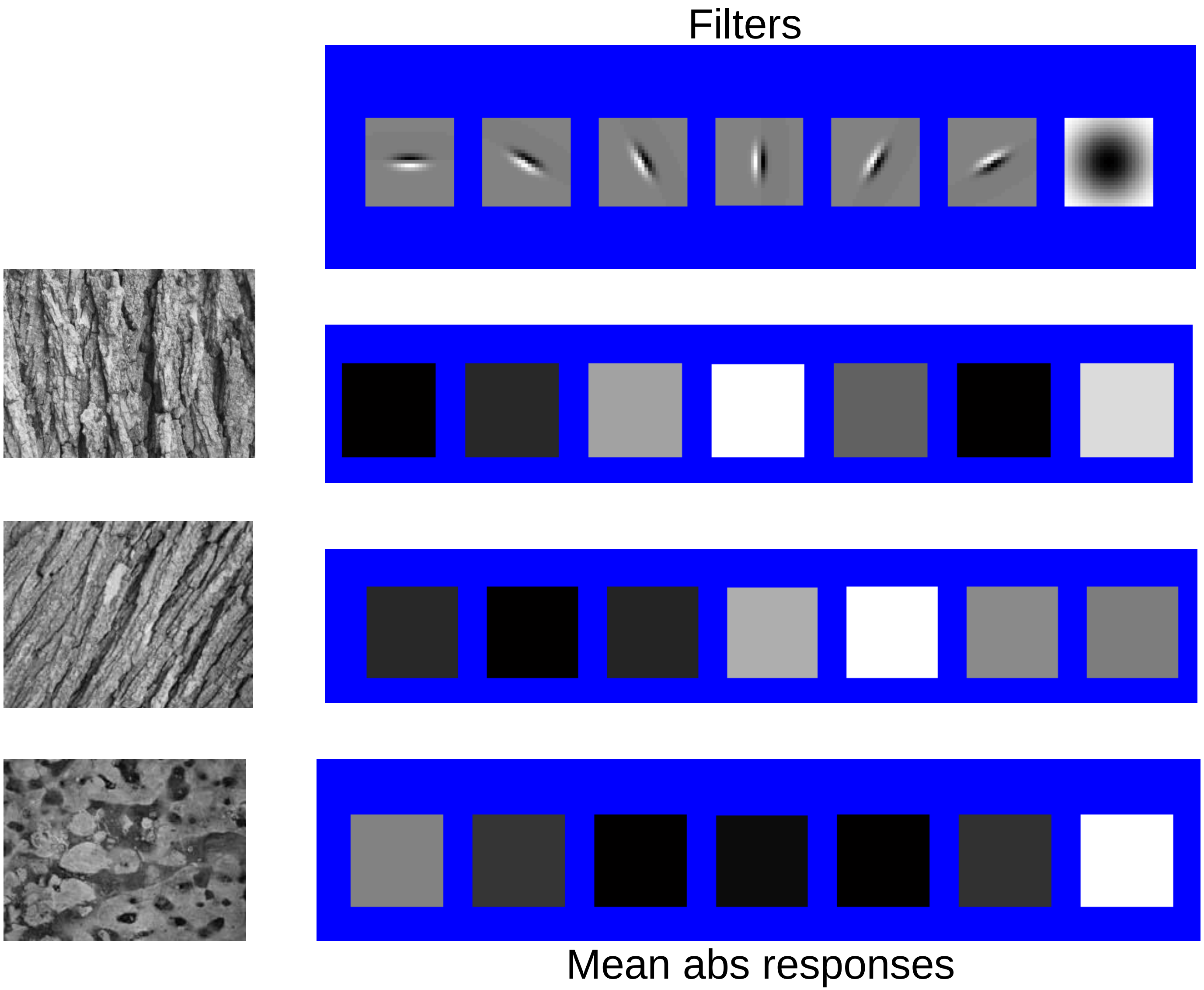
- Measure responses of blobs and edges at various orientations and scales

- Idea 1: Record simple statistics (e.g., mean, std.) of absolute filter responses

Can you match the texture to the response?

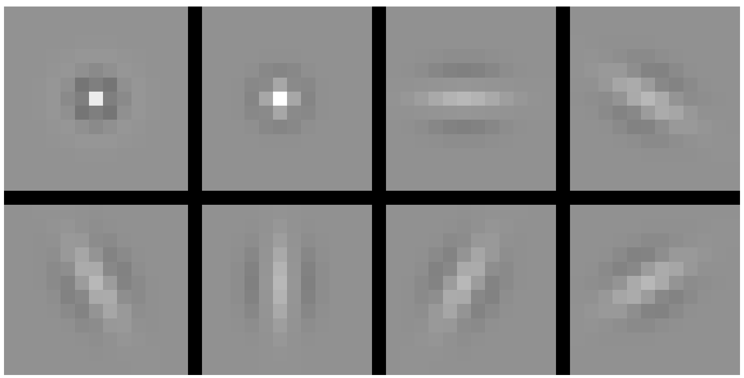


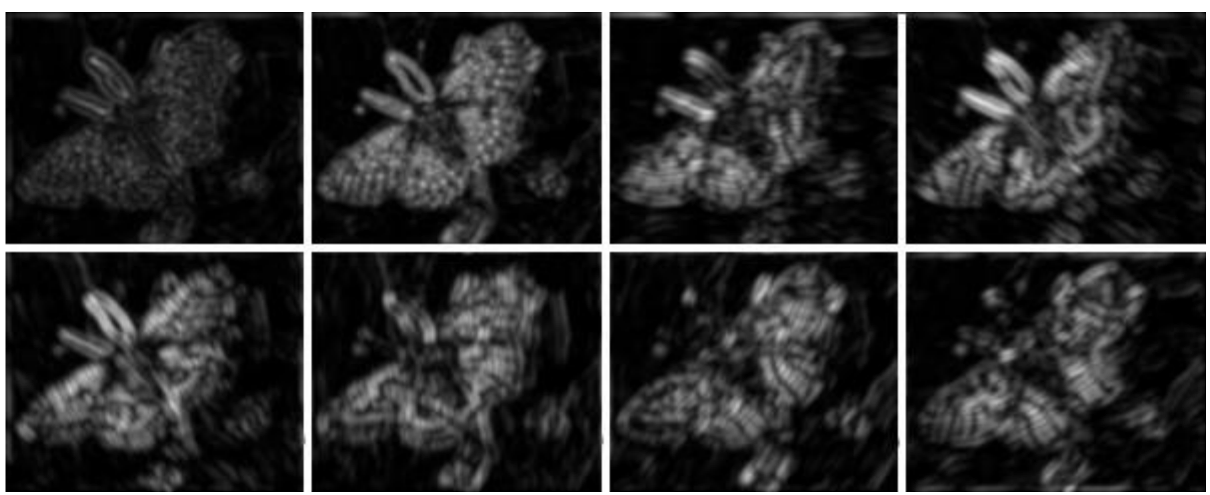
Representing texture by mean abs response



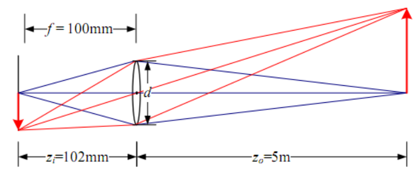
Representing texture

- Idea 2: take vectors of filter responses at each pixel and cluster them, then take histograms (more on in later weeks)





Course Outline



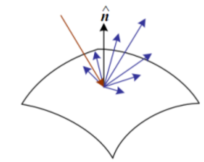
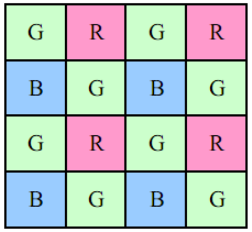


Image Formation and Processing

Light, Shape and Color

The Pin-hole Camera Model, The Digital Camera

Linear filtering, Template Matching, Image Pyramids

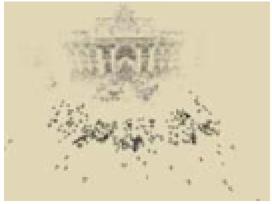


Feature Detection and Matching

Edge Detection, Interest Points: Corners and Blobs

Local Image Descriptors

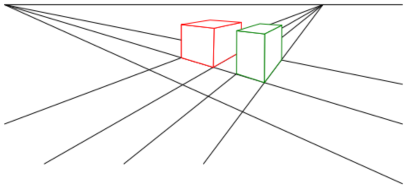
Feature Matching and Hough Transform



Multiple Views and Motion

Geometric Transformations, Camera Calibration

Feature Tracking , Stereo Vision



Segmentation and Grouping

Segmentation by Clustering, Region Merging and Growing

Advanced Methods Overview: Active Contours, Level-Sets, Graph-Theoretic Methods

Detection and Recognition

Problems and Architectures Overview

Statistical Classifiers, Bag-of-Words Model, Detection by Sliding Windows

Resources

Books

R. Szeliski, Computer Vision: Algorithms and Applications, 2010 – *available online*

D. A. Forsyth and J. Ponce, Computer Vision: A Modern Approach, 2003

L. G. Shapiro and G. C. Stockman, Computer Vision, 2001

Web

CVonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision

http://homepages.inf.ed.ac.uk/rbf/CVonline/

Dictionary of Computer Vision and Image Processing

http://homepages.inf.ed.ac.uk/rbf/CVDICT/

Computer Vision Online

http://www.computervisiononline.com/

Programming

Development environments/languages: Matlab, Python and C/C++

Toolboxes and APIs: OpenCV, VLFeat Matlab Toolbox, Piotr's Computer Vision Matlab Toolbox, EasyCamCalib Software, FLANN, Point Cloud Library PCL, LibSVM, Camera Calibration Toolbox for Matlab