Computer Vision

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2023-2024

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

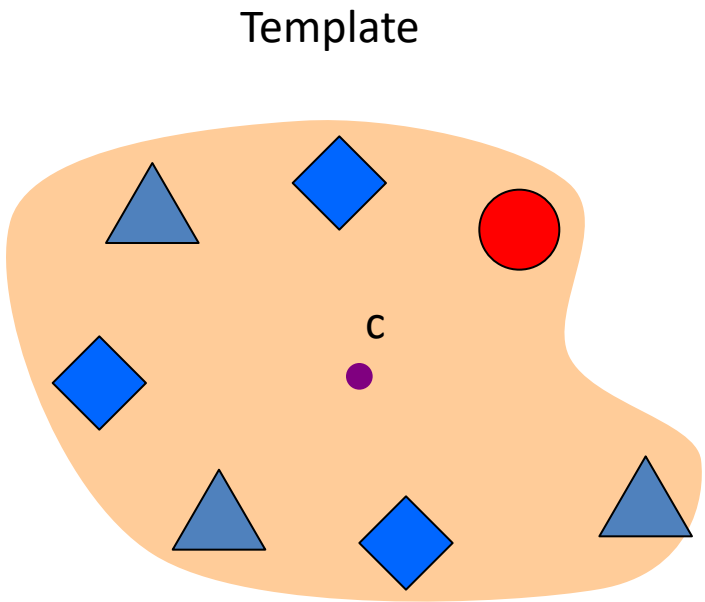
   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Generalized Hough transform

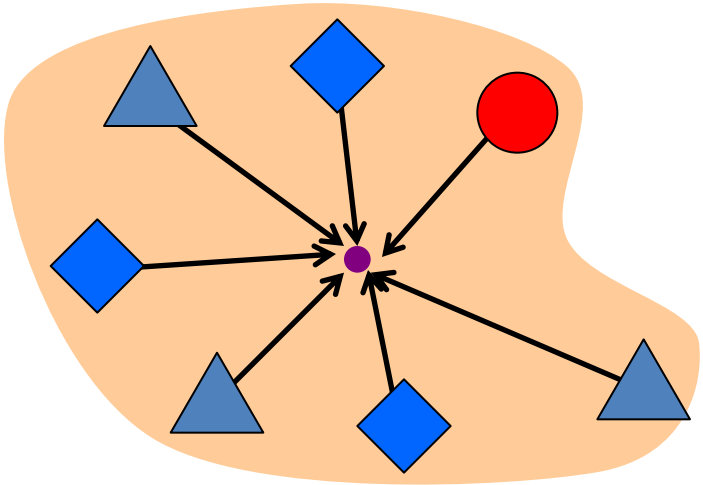
- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



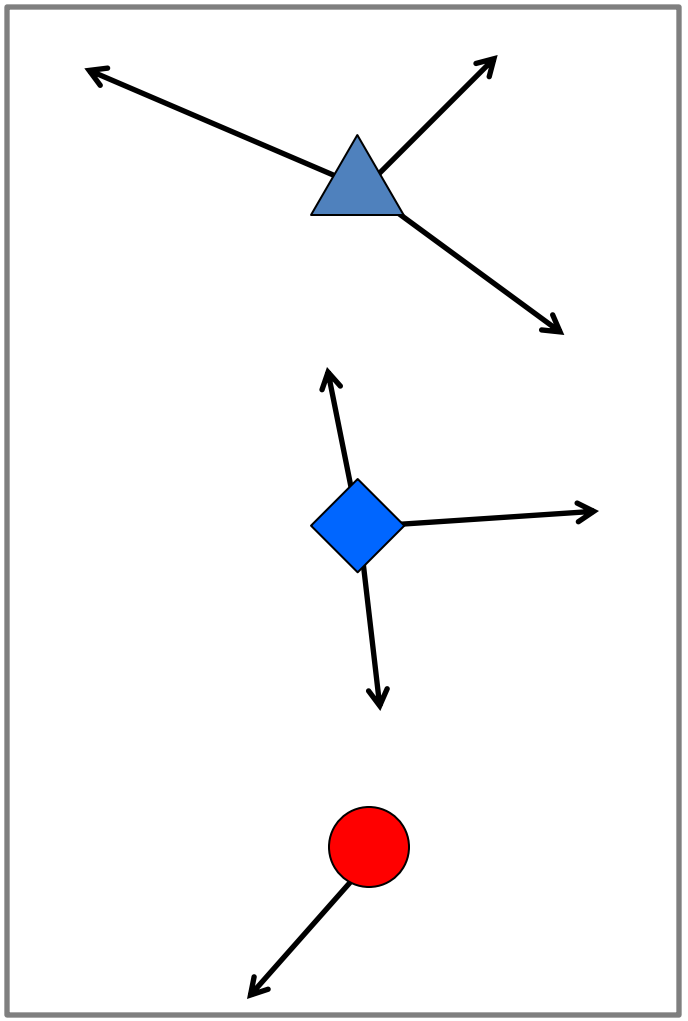
Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center

Template



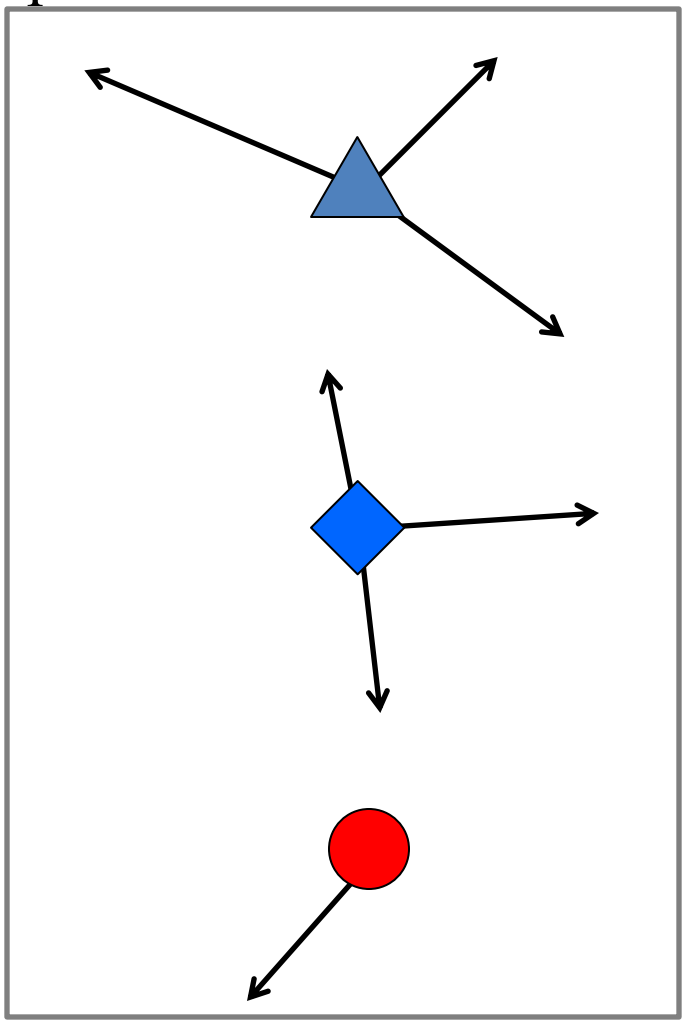
Model



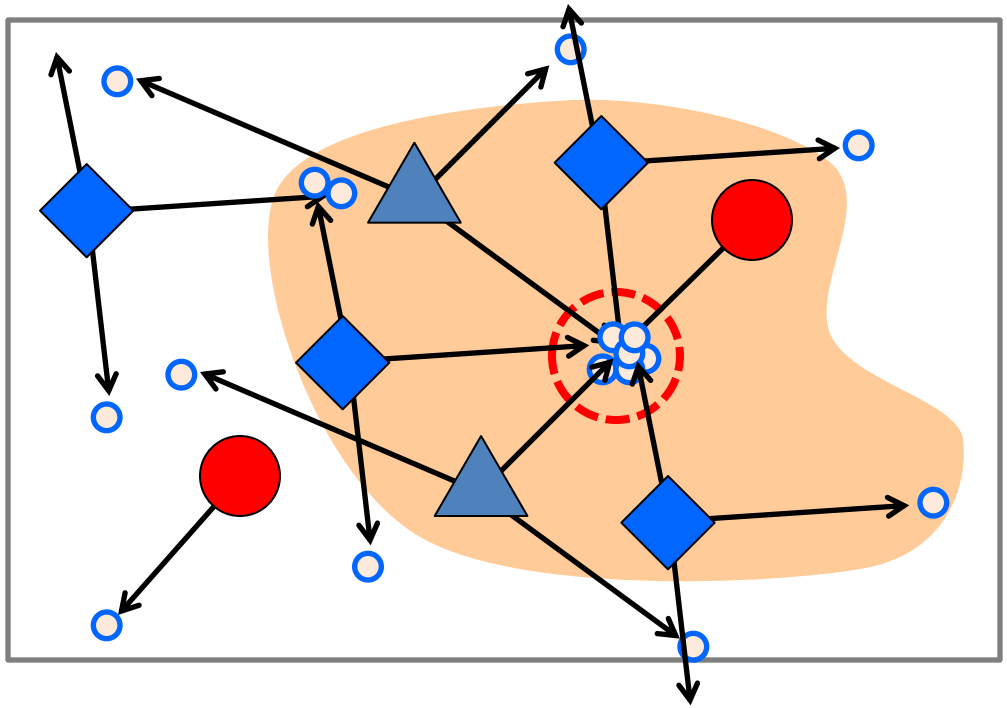
Generalized Hough transform

- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model

Model

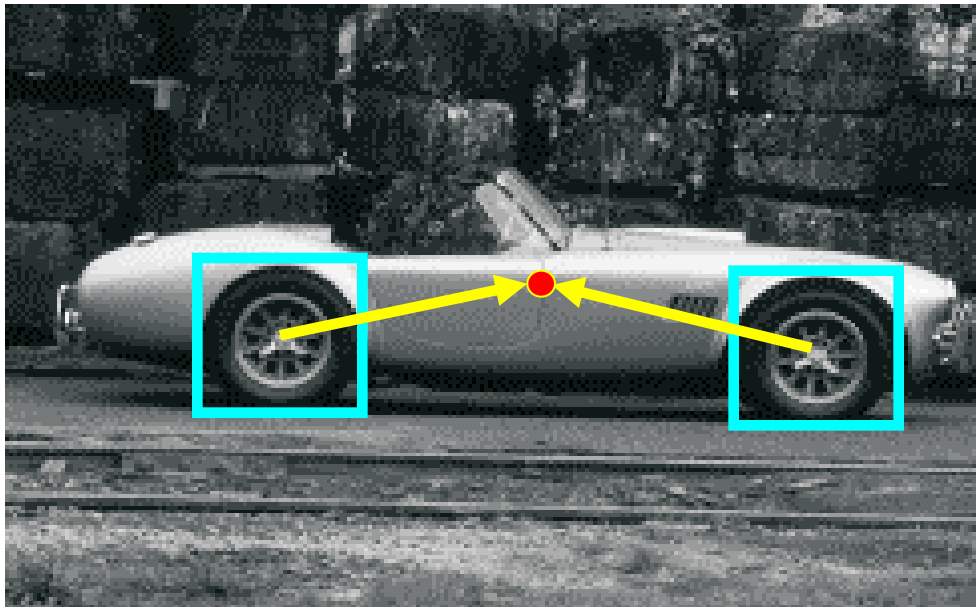


Test image



Application in recognition

- Index displacements by “visual codeword”



training image

visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

Application in recognition

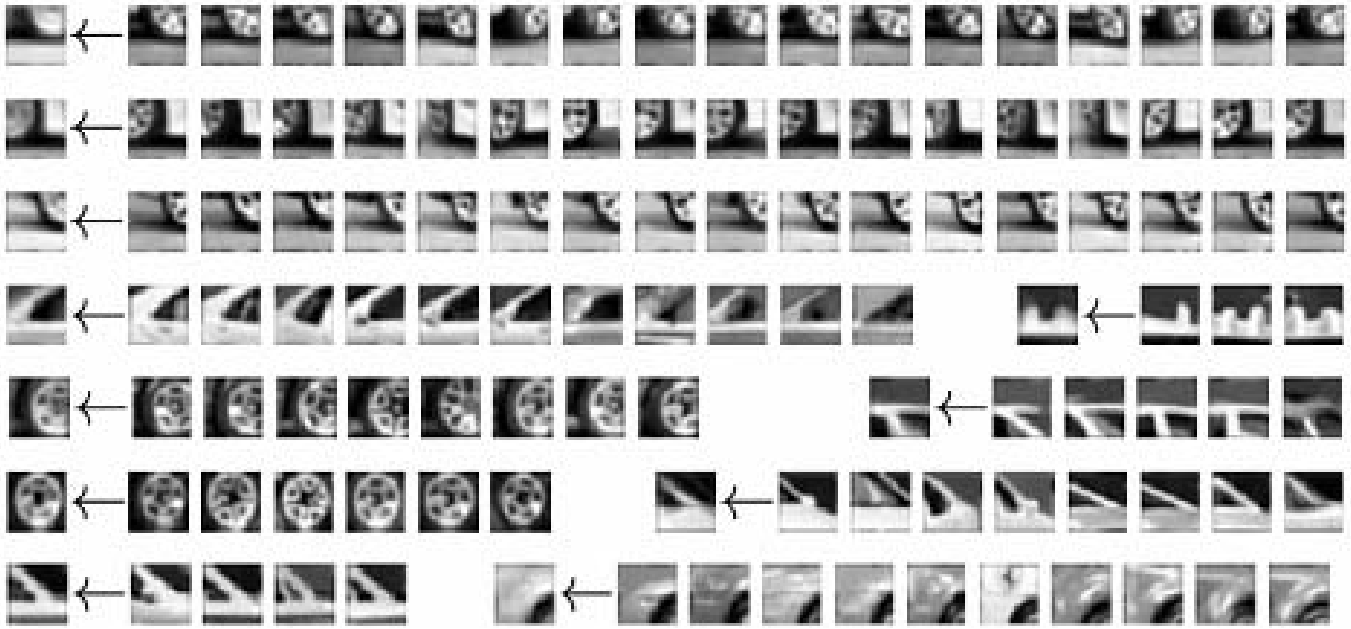
- Index displacements by “visual codeword”

test image

B. Leibe, A. Leonardis, and B. Schiele, Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV Workshop on Statistical Learning in Computer Vision 2004

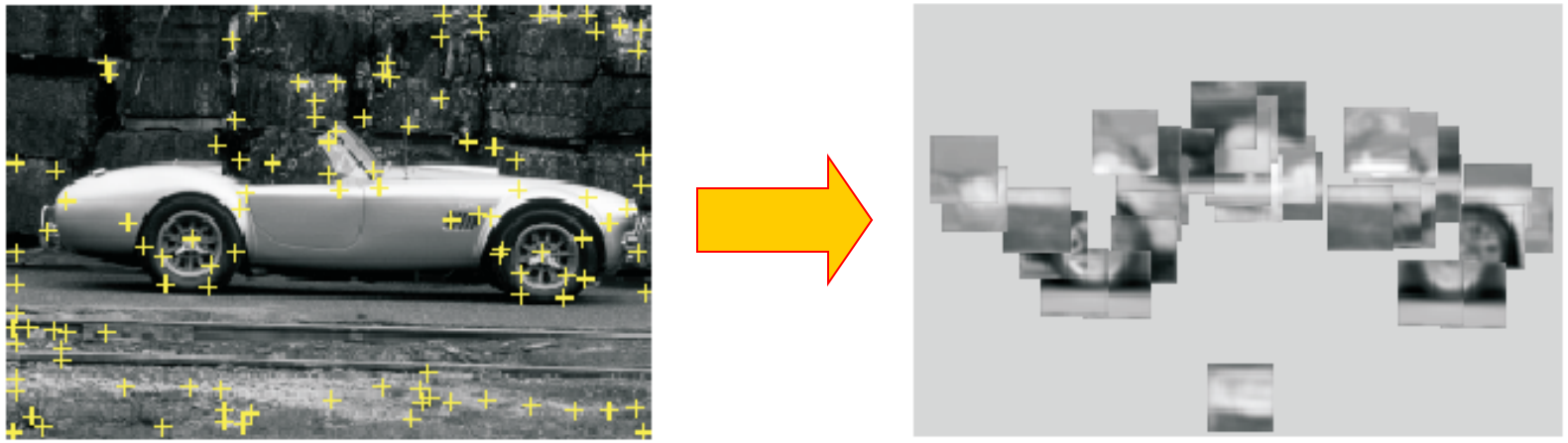
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering



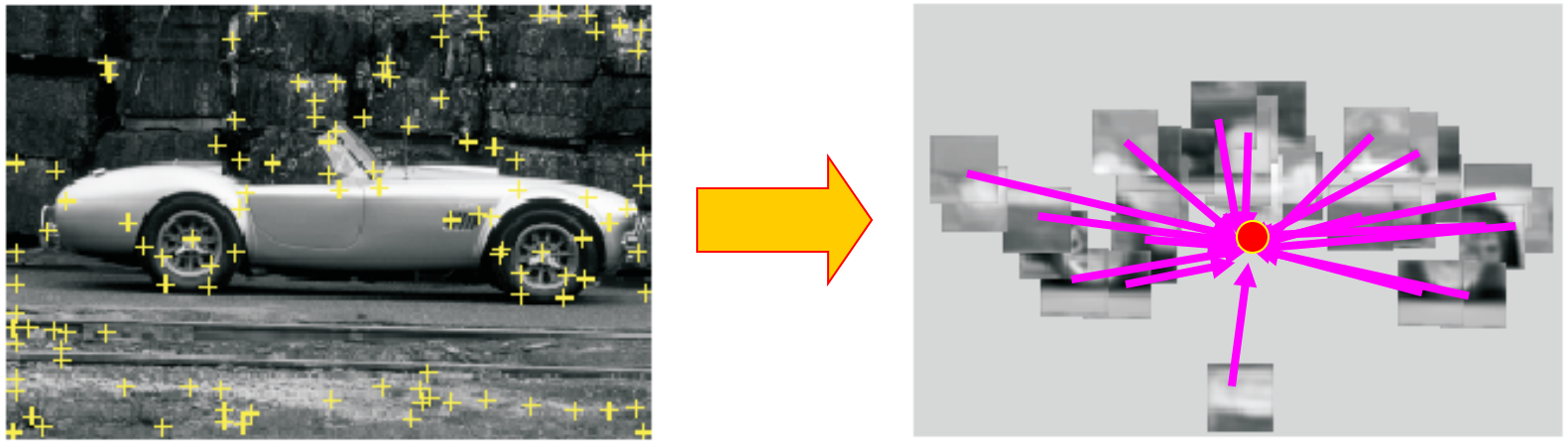
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry



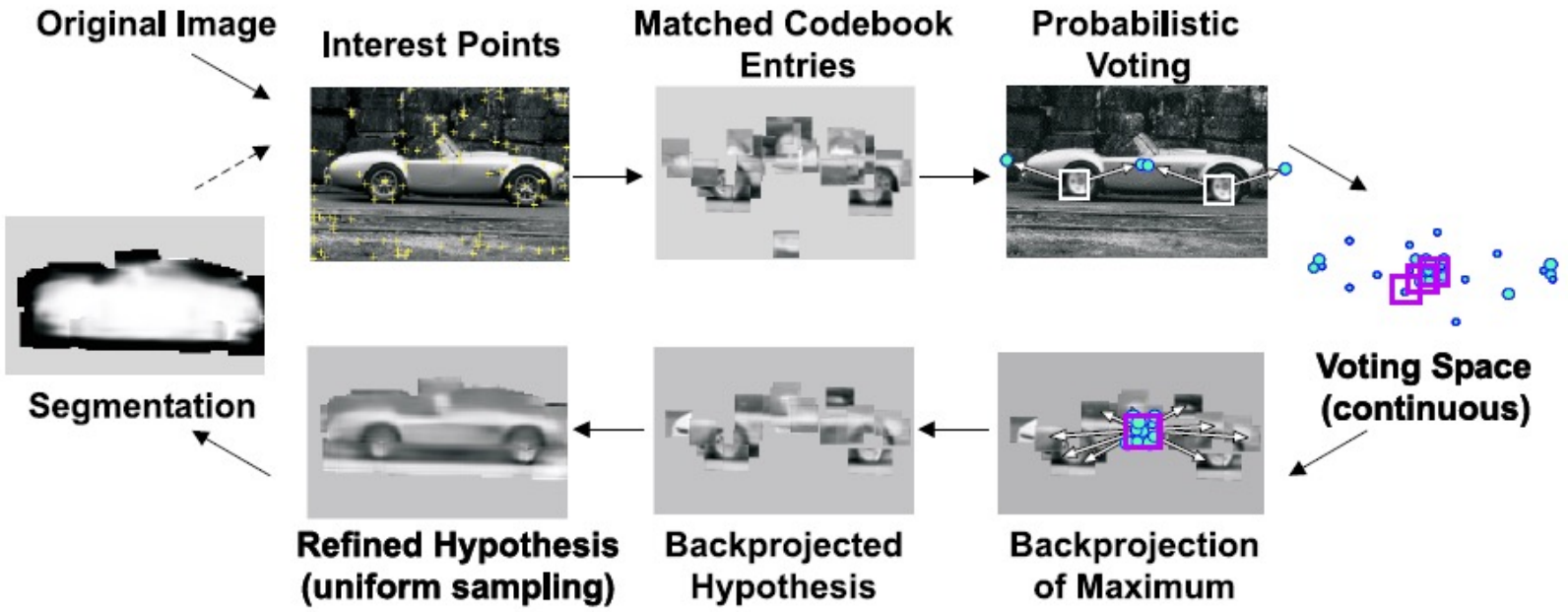
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions it was found, relative to object center

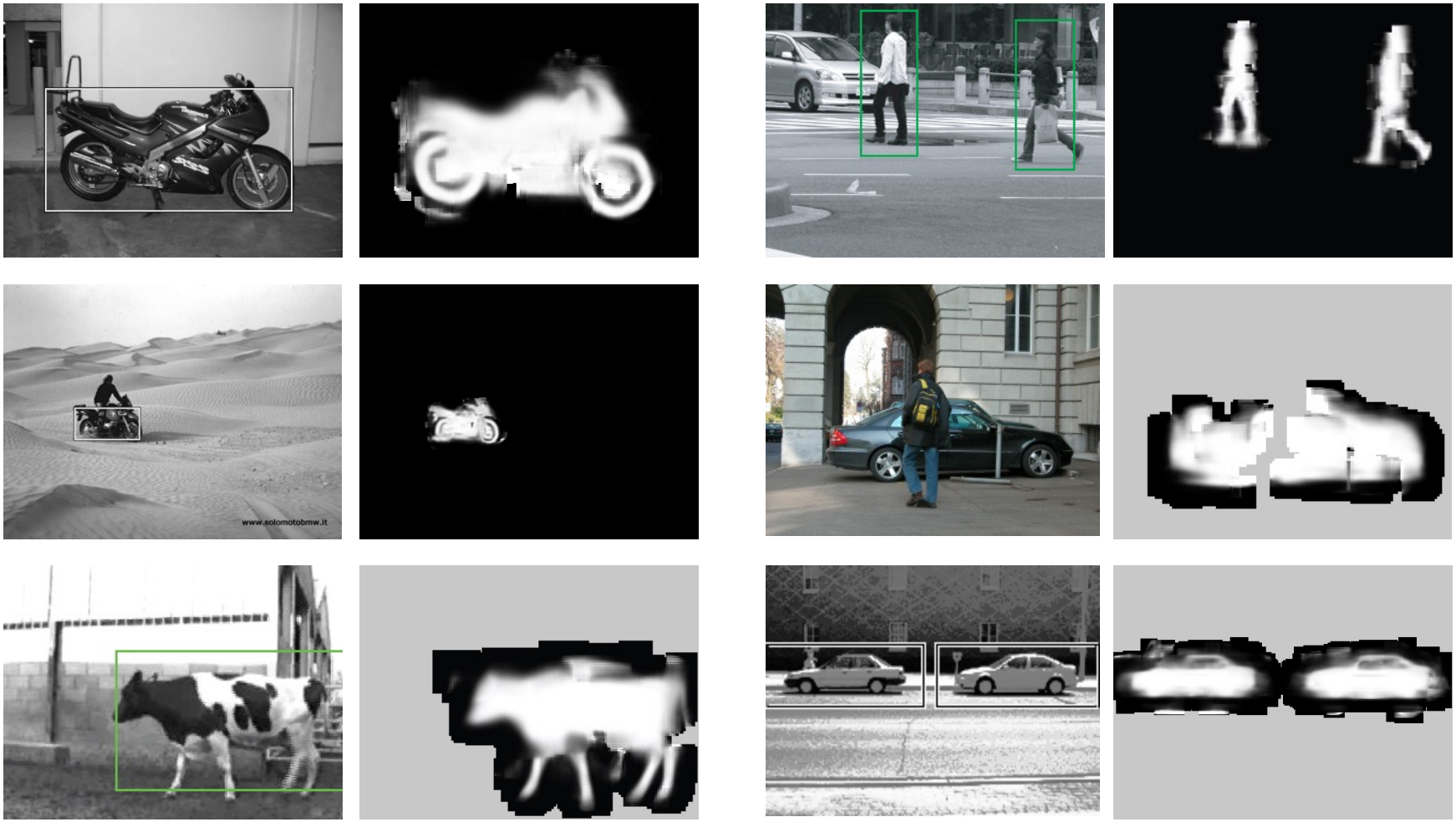


Implicit shape models: Testing

1. Given test image, extract patches, match to codebook entry
2. Cast votes for possible positions of object center
3. Search for maxima in voting space
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences

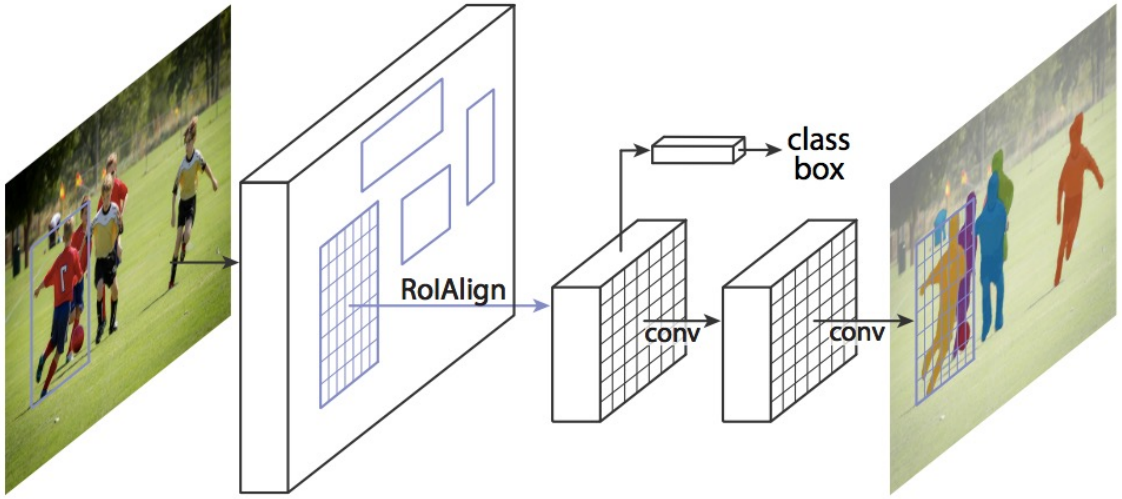


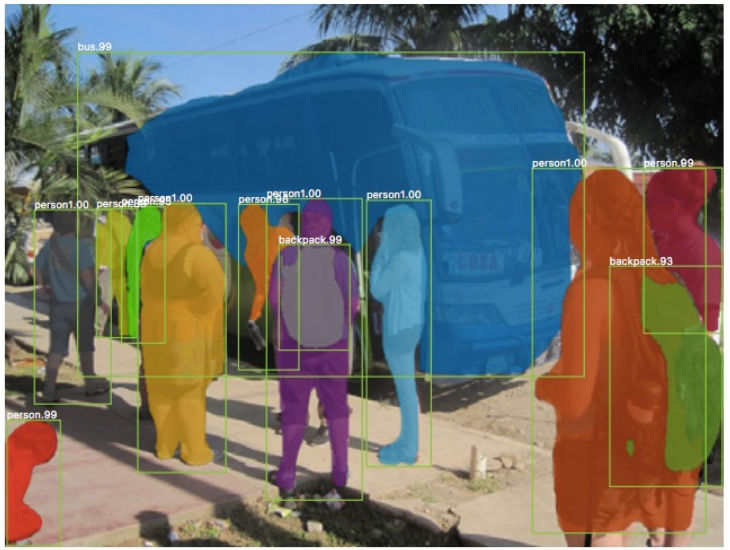
Additional examples

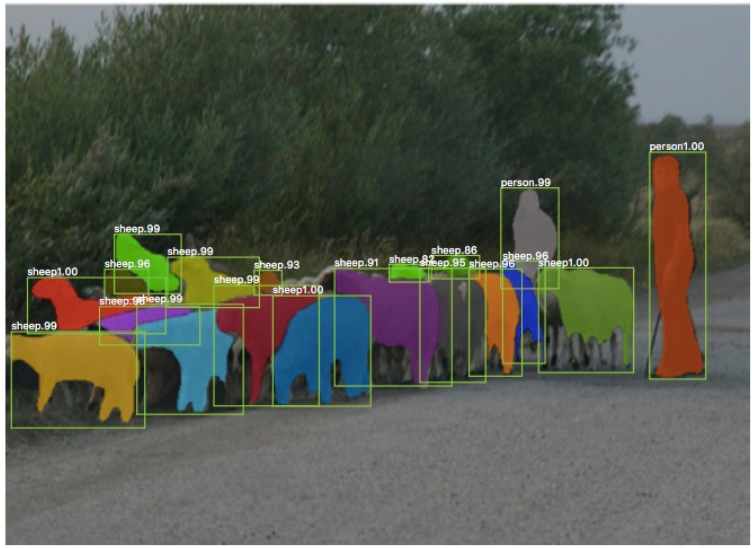


B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation,
IJCV 77 (1-3), pp. 259-289, 2008.

These days: Mask R-CNN







K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017

RANSAC

RANSAC

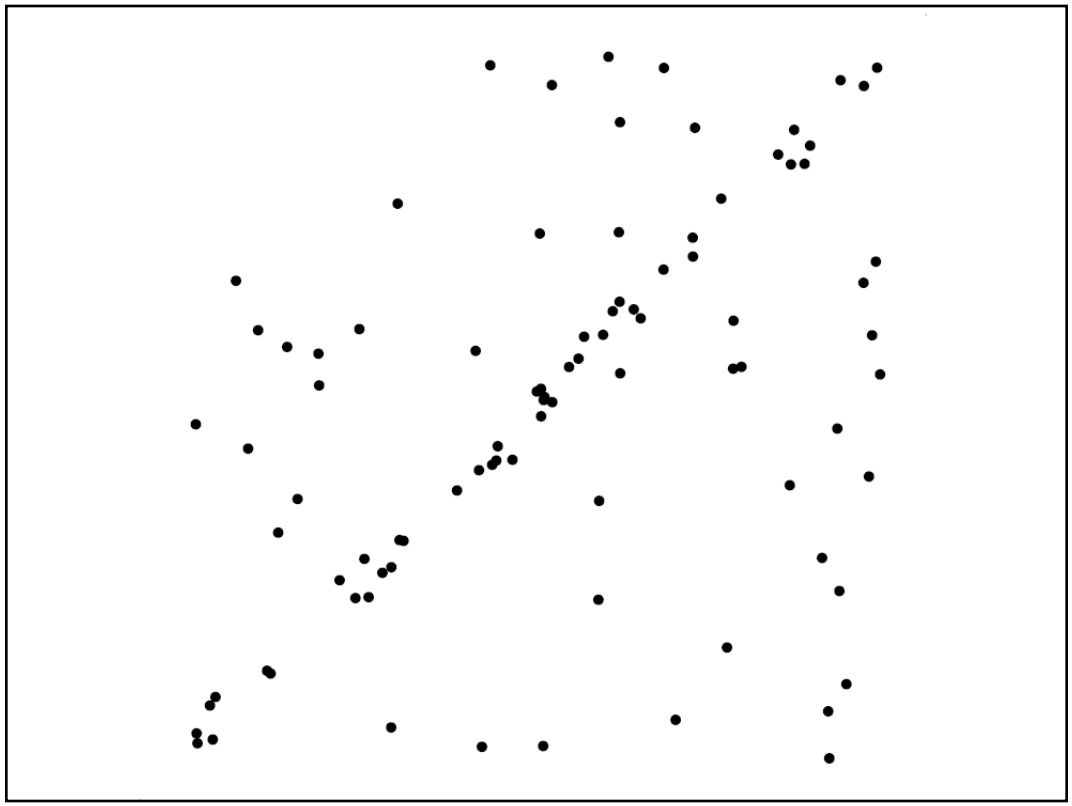
- Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers

- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are “close” to the model and reject the rest as outliers
  - Do this many times and choose the best model

M. A. Fischler, R. C. Bolle
Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.
Comm. of the ACM, Vol 24, pp 381-395, 1981.

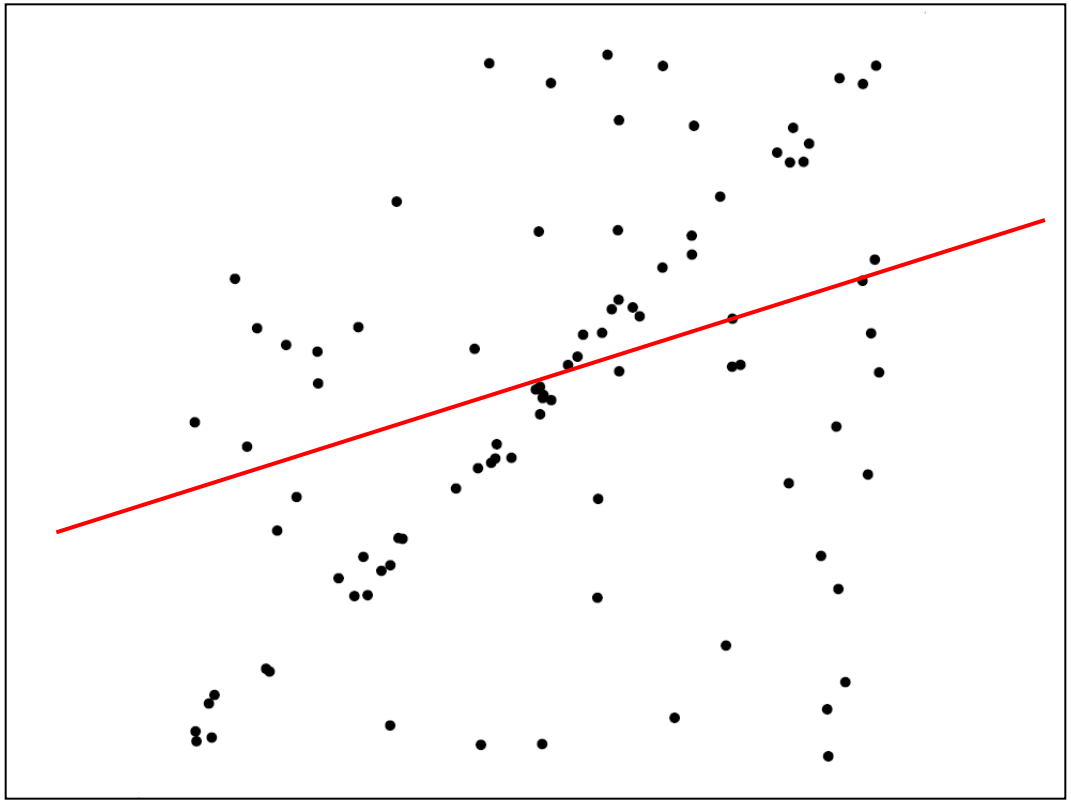
Slide credit : S. Lazebnik

RANSAC for line fitting example



Slide credit: R. Raguram

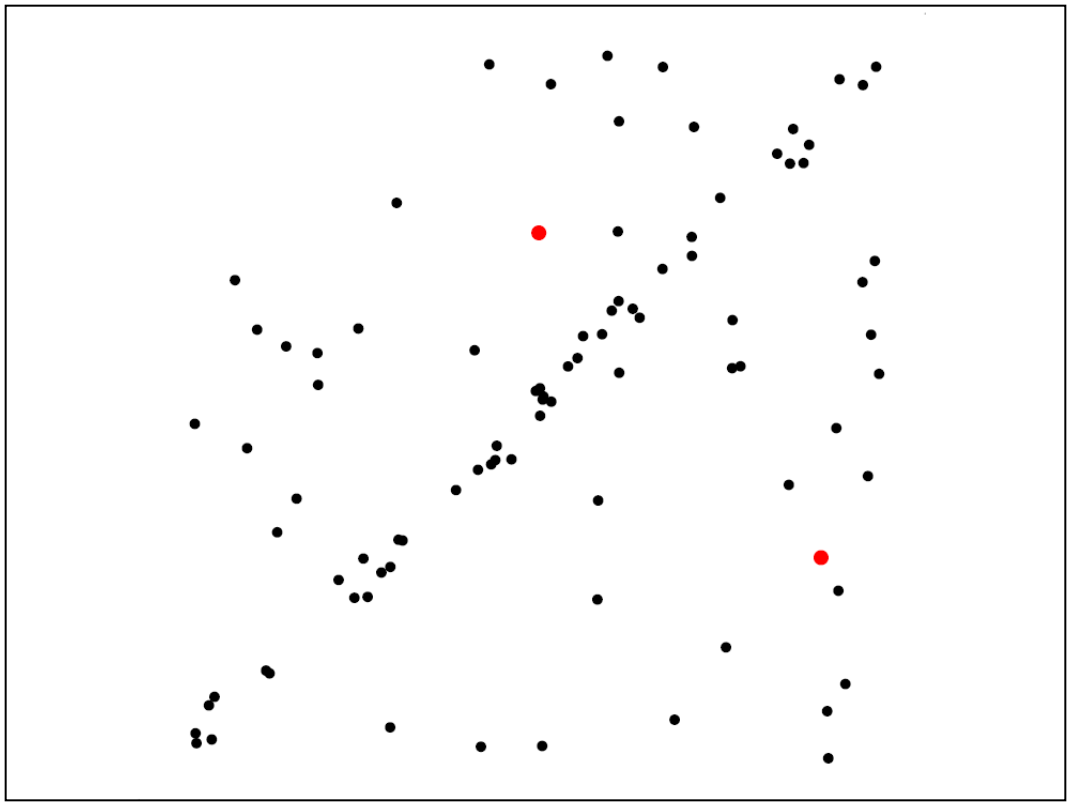
RANSAC for line fitting example



Least-squares fit

Slide credit: R. Raguram

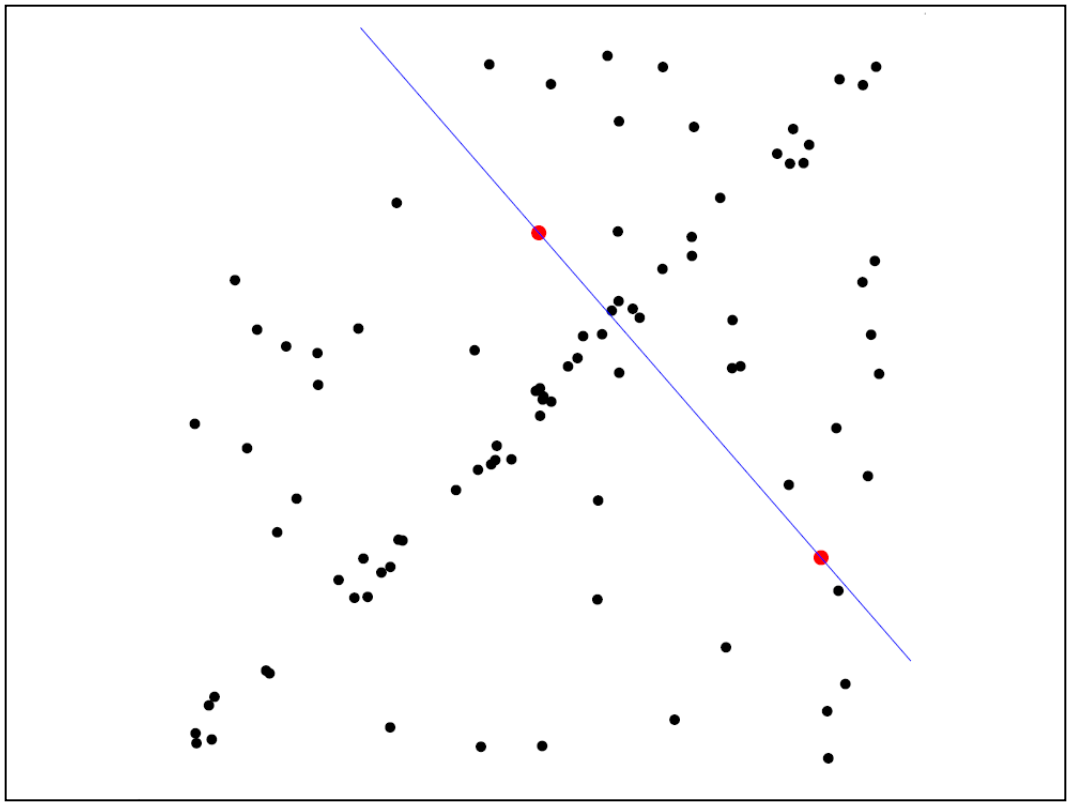
RANSAC for line fitting example



1. Randomly select minimal subset of points

Slide credit: R. Raguram

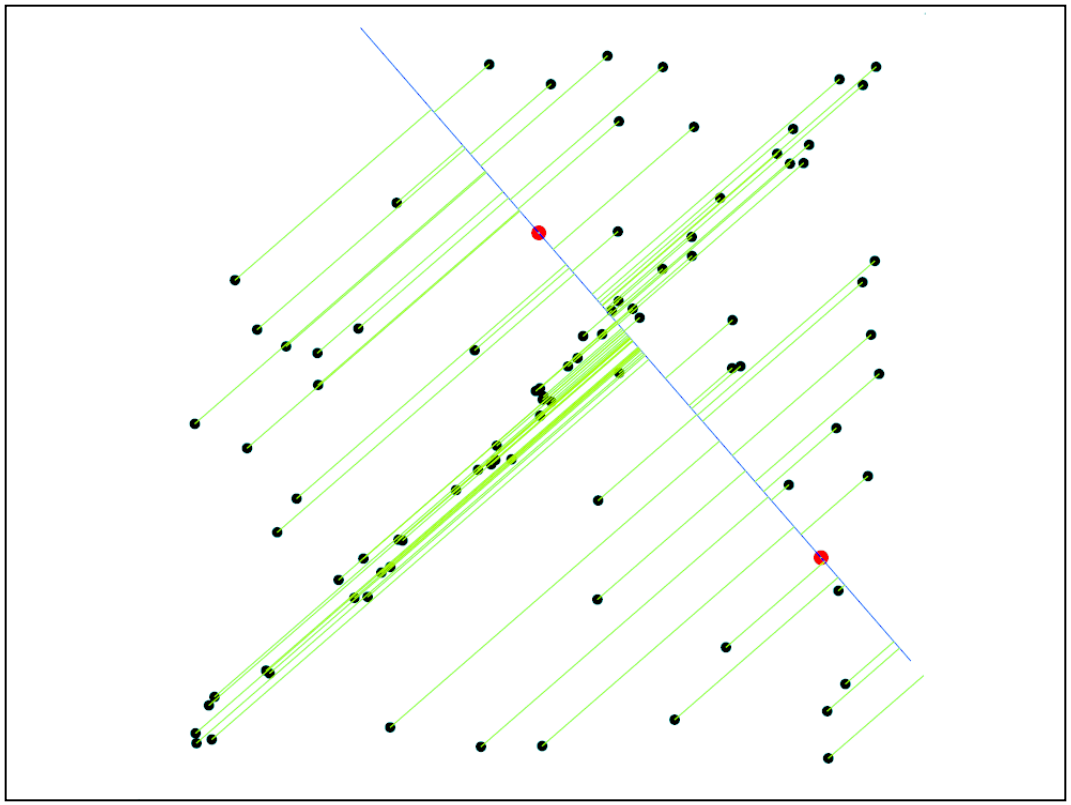
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Slide credit: R. Raguram

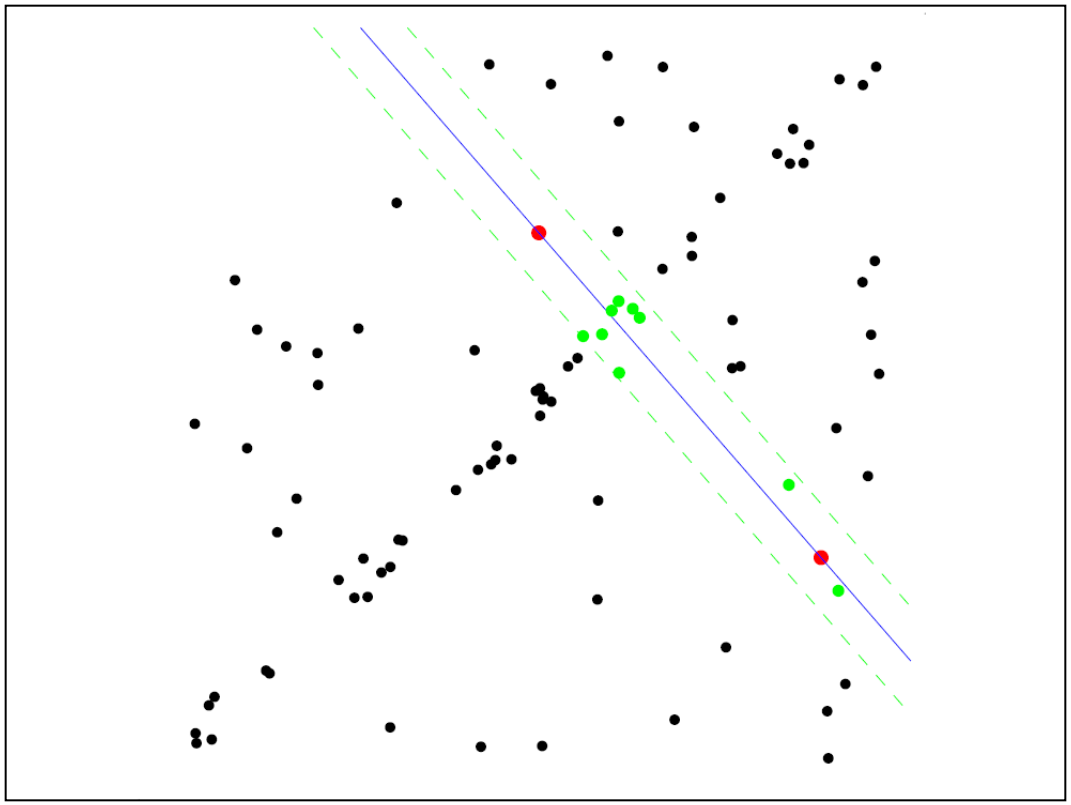
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Slide credit: R. Raguram

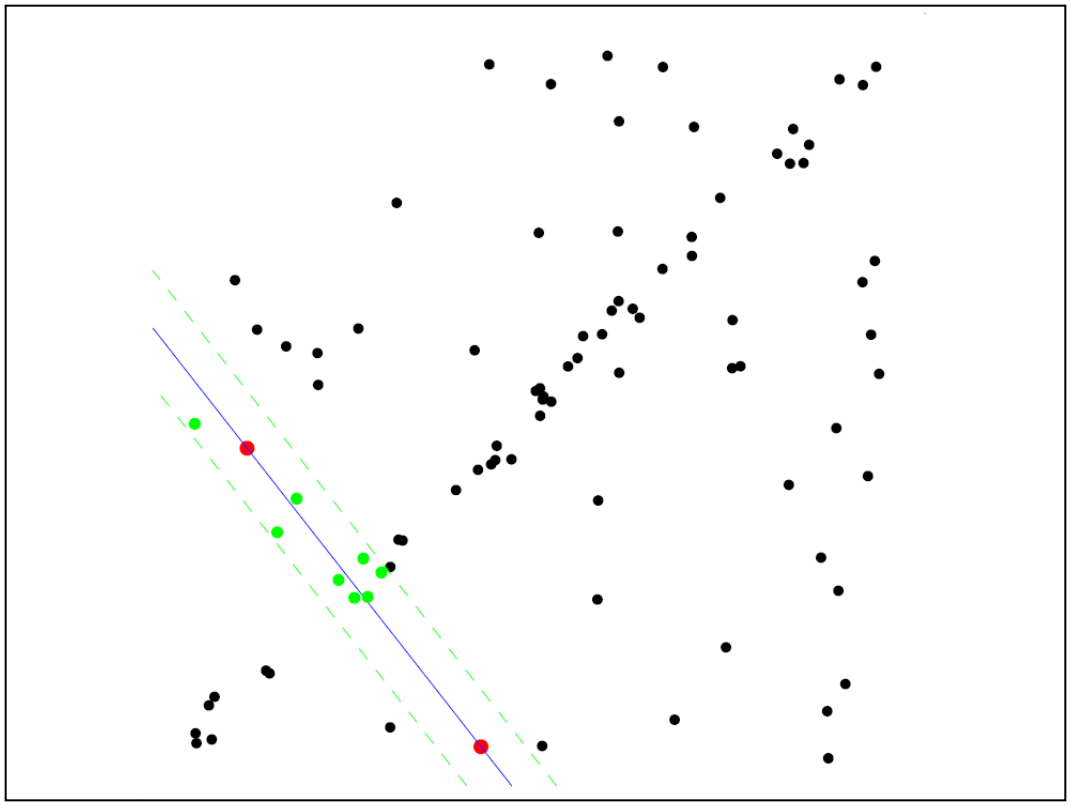
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Slide credit: R. Raguram

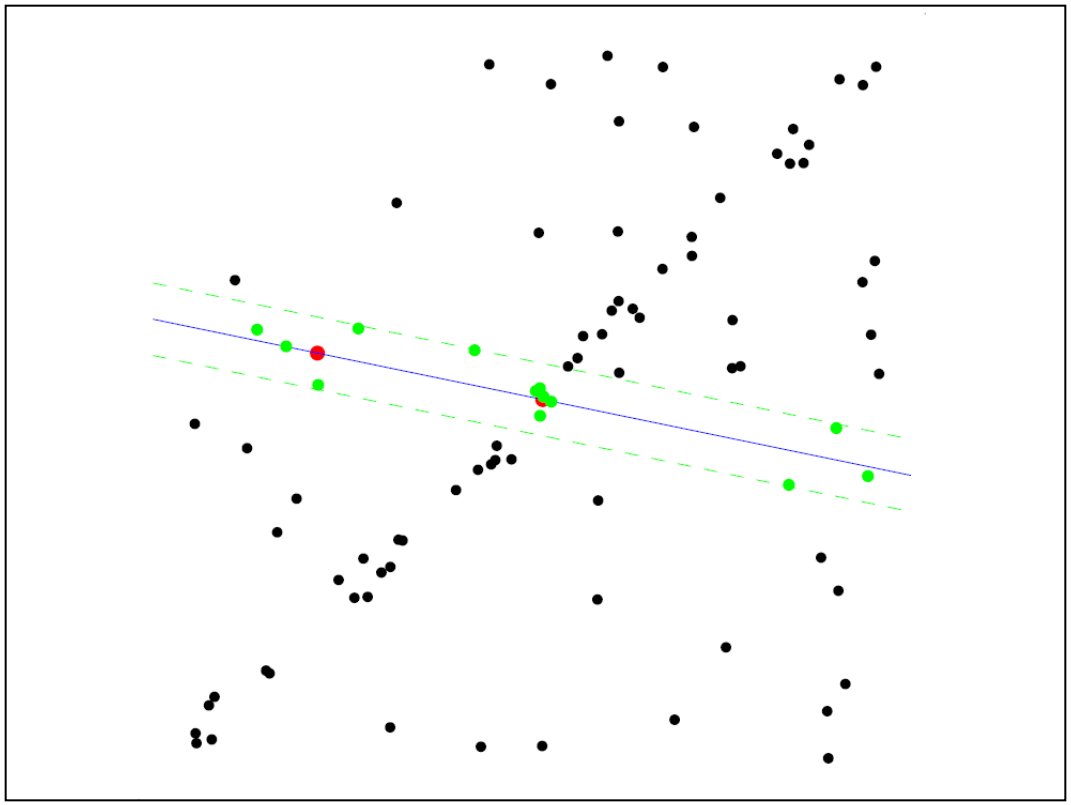
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example

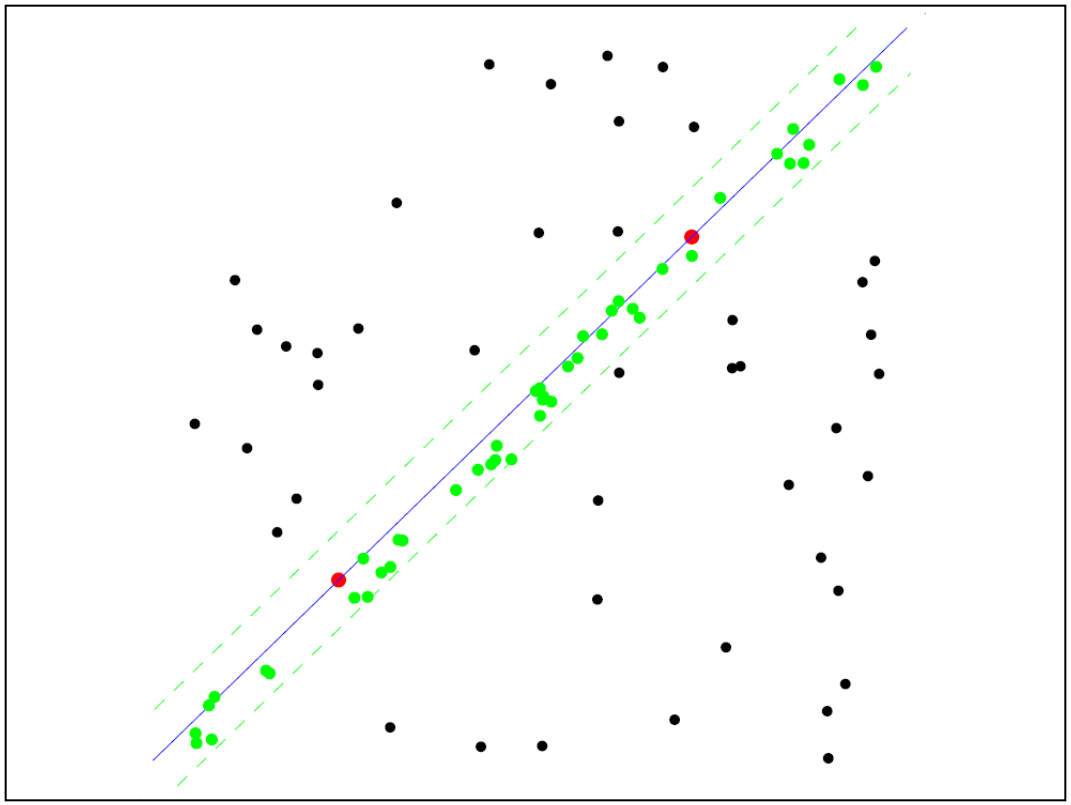


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example

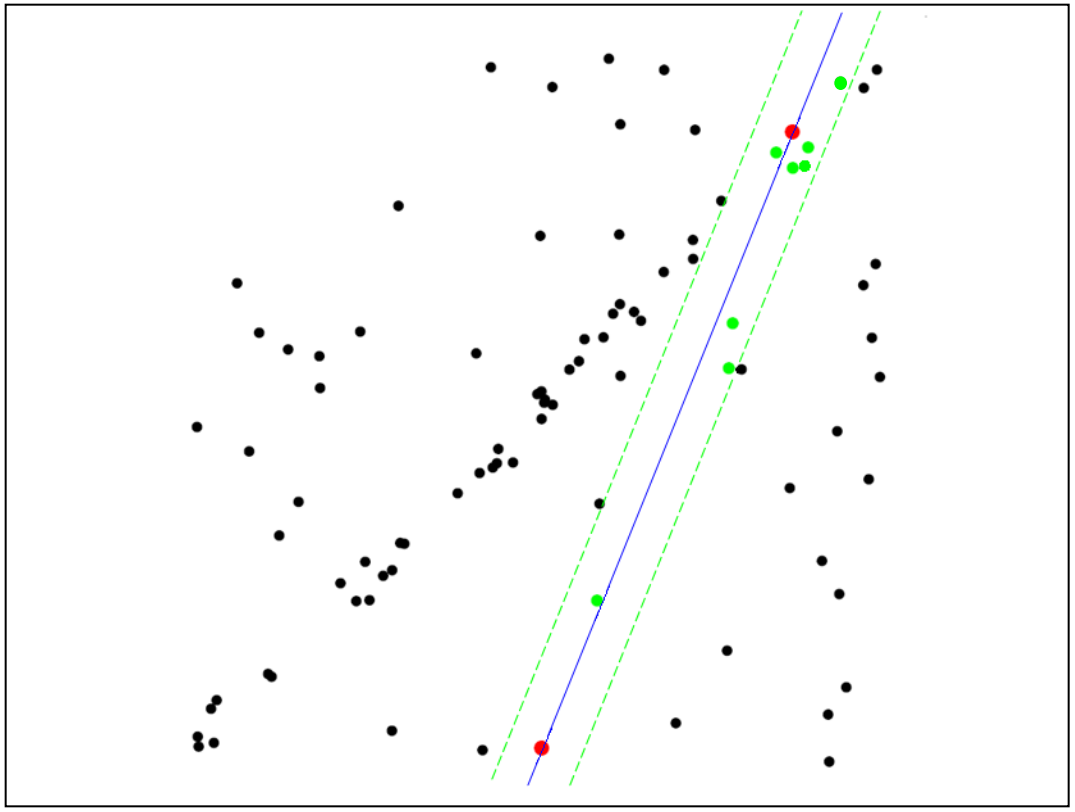
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Slide credit: R. Raguram

RANSAC for line fitting

- Repeat ***N*** times:
  - Draw ***s*** points uniformly at random
  - Fit a line (model) to these ***s*** points
  - Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
  - If there are ***d*** or more inliers, accept the line and refit using all inliers

Slide credit: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$, $t=1.9\sigma$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: *e*)

Slide credit: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is p (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right)/\log\left(1-\left(1-e\right)^s\right)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

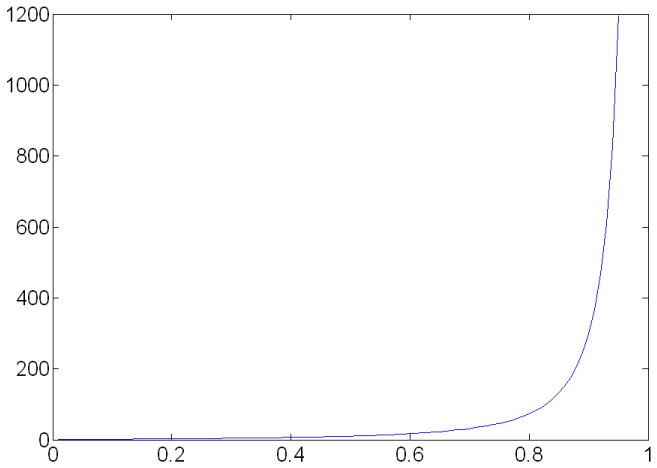
Slide credit: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right)/\log\left(1-\left(1-e\right)^s\right)$$



Slide credit: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Consensus set size ***d***
  - Should match expected inlier ratio

Slide credit: M. Pollefeys

Adaptively determining the number of samples

- Outlier ratio *e* is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield *e*=0.2

- Adaptive procedure:
  - $N = \infty$, *sample_count* =0
  - While $N >$ *sample_count*
    - Choose a sample and count the number of inliers
    - If inlier ratio is highest of any found so far, set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

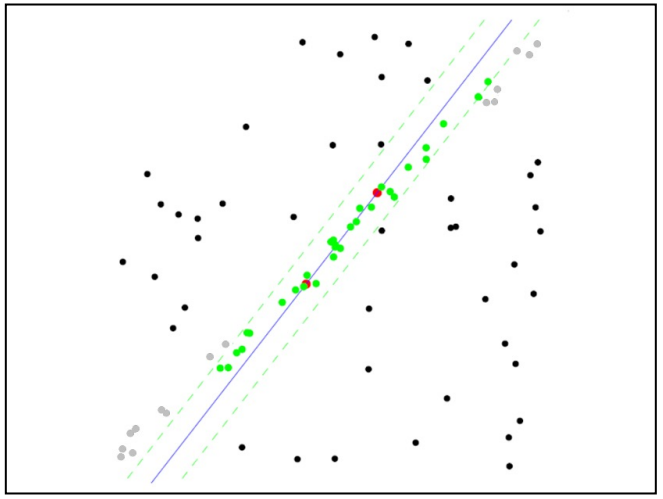
$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

    - Increment the *sample_count* by 1

Slide credit: M. Pollefeys

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
  - Can't always get a good initialization of the model based on the minimum number of samples



Slide credit : S. Lazebnik

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

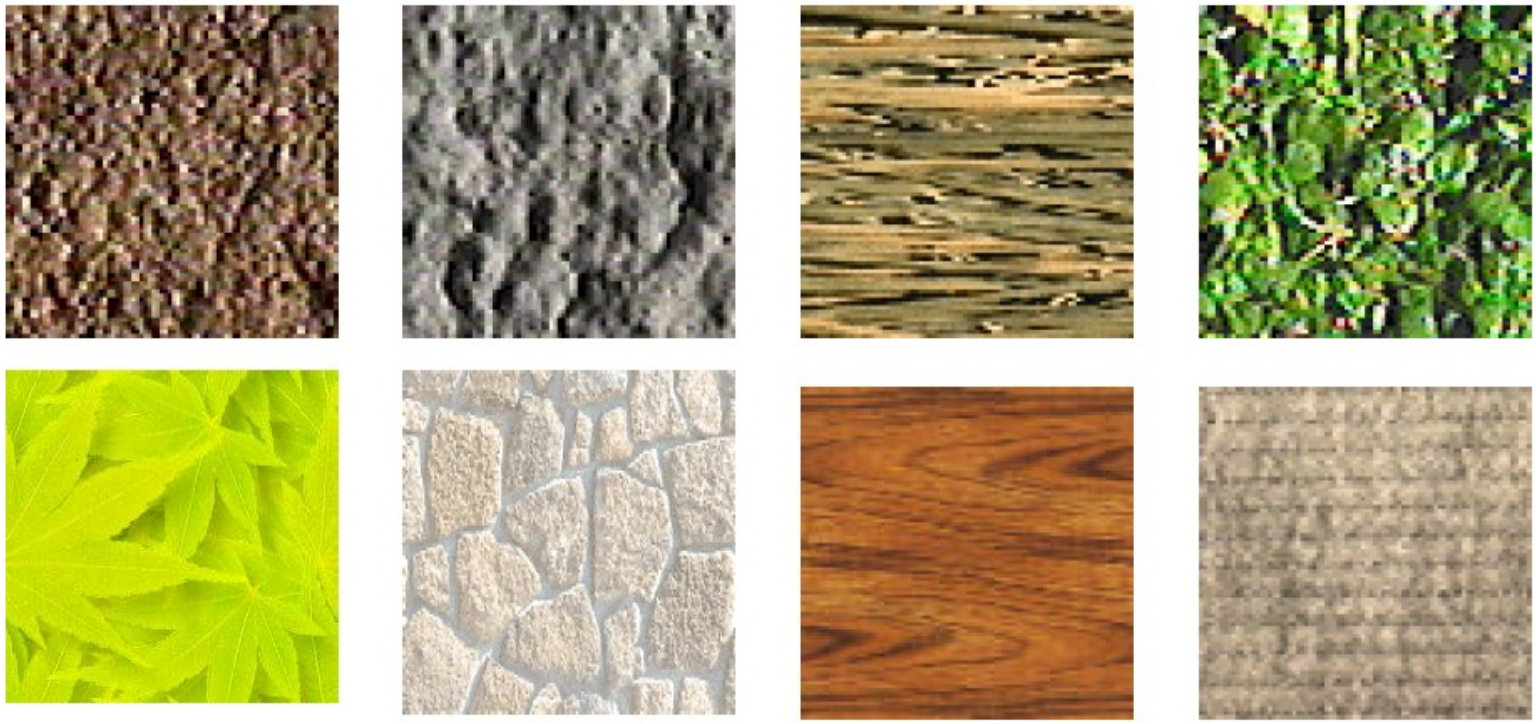
   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

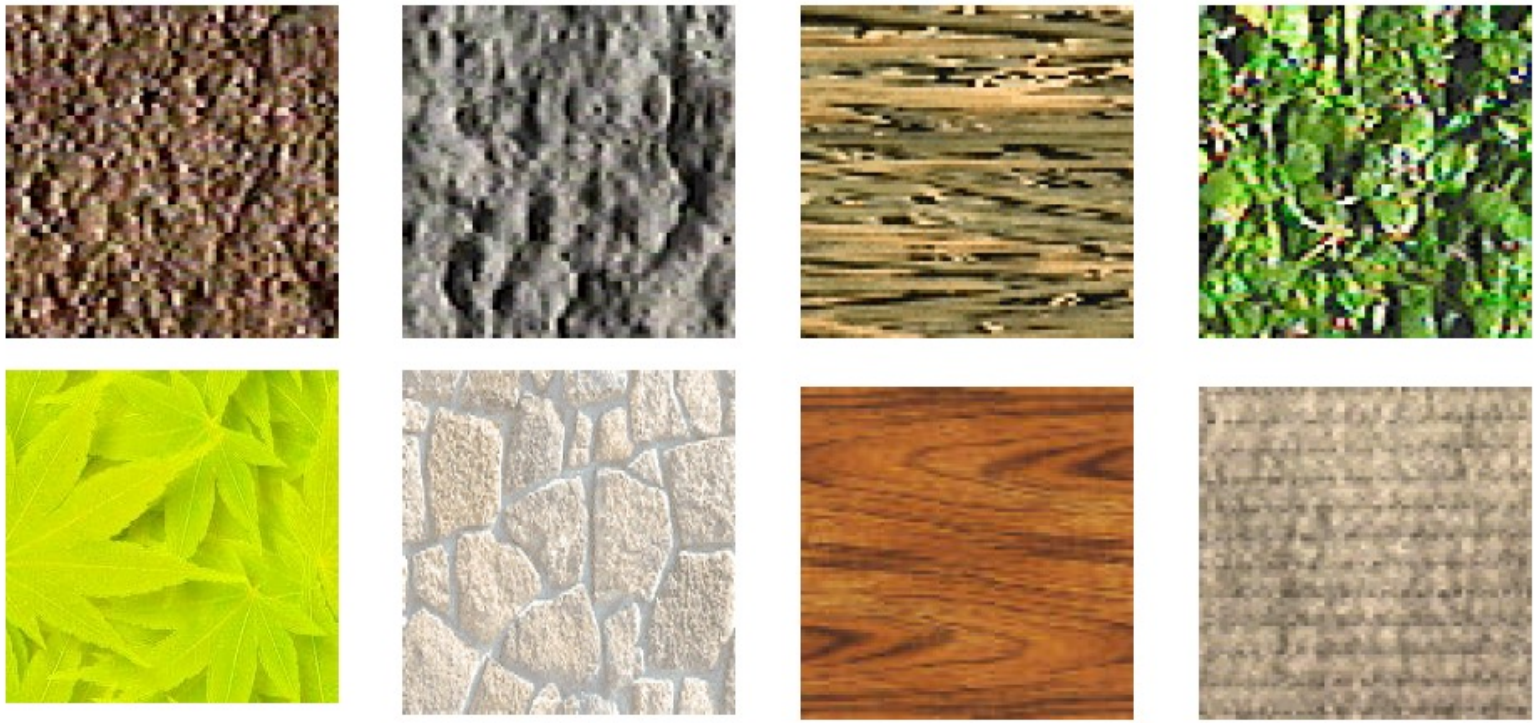
Texture

Texture



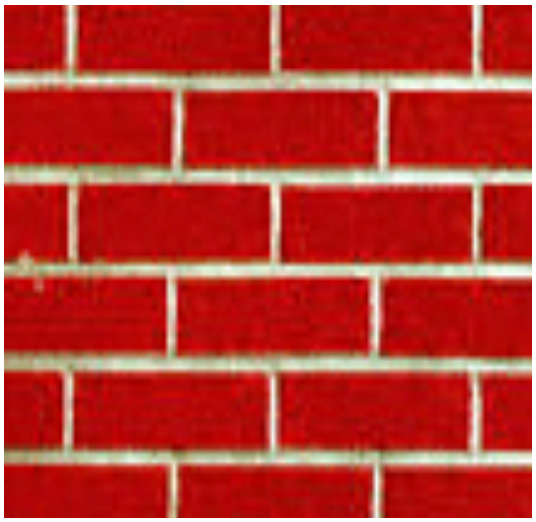
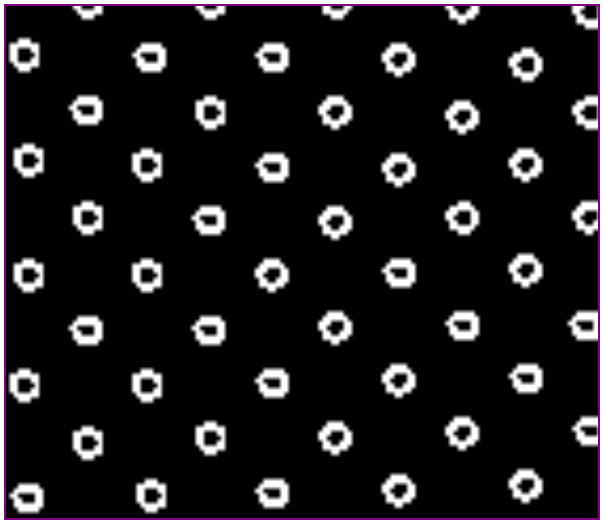
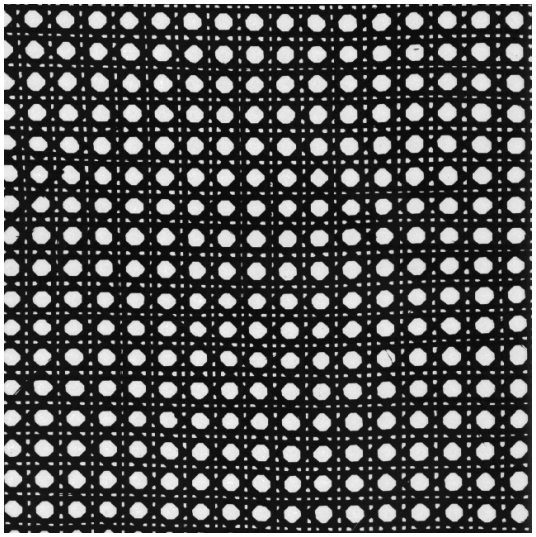
What defines a texture?

Texture



Similar structures of pixels (patterns) that repeat themselves
Similar aspect at a fixed scale
Follow some statistics properties

Includes: more regular patterns

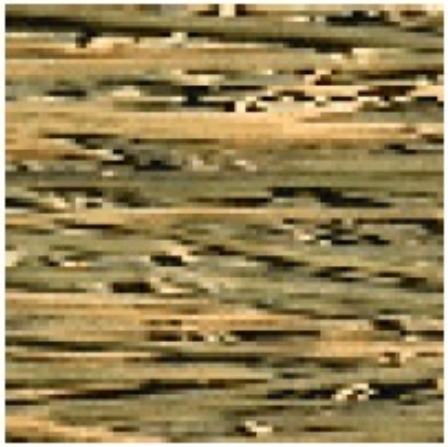
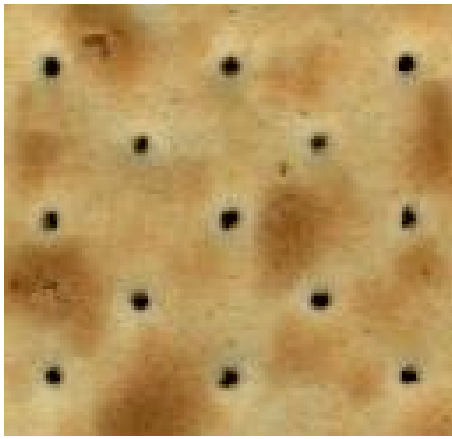




Texture = set of ***texels*** in some regular pattern

Texels = texture elements

Includes: more random patterns



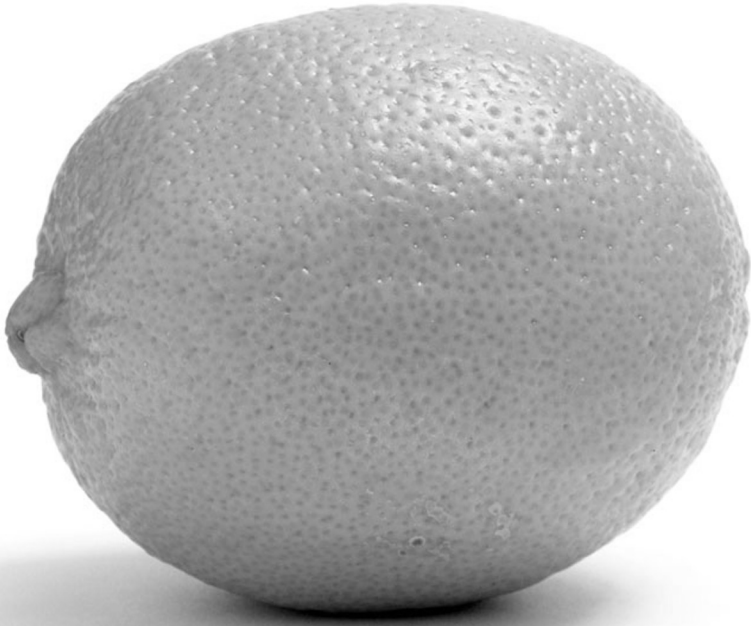
Texture = set of ***texels*** in some regular or repeated pattern

Texels = texture elements

Why analyze texture?

Importance to perception:

- Often indicative of a material's properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture



Slide credit: Kristen Grauman

http://animals.nationalgeographic.com/

Slide credit: Kristen Grauman

Why analyze texture?

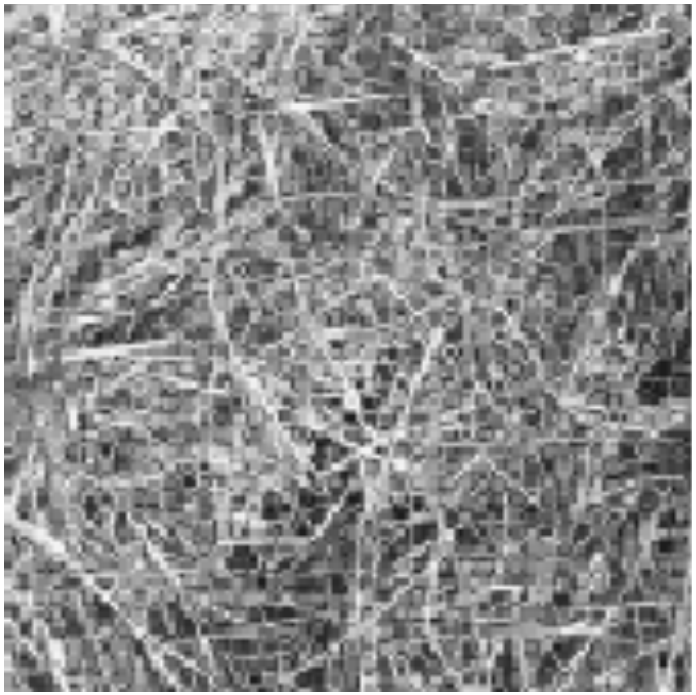
Importance to perception:

- Often indicative of a material's properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture

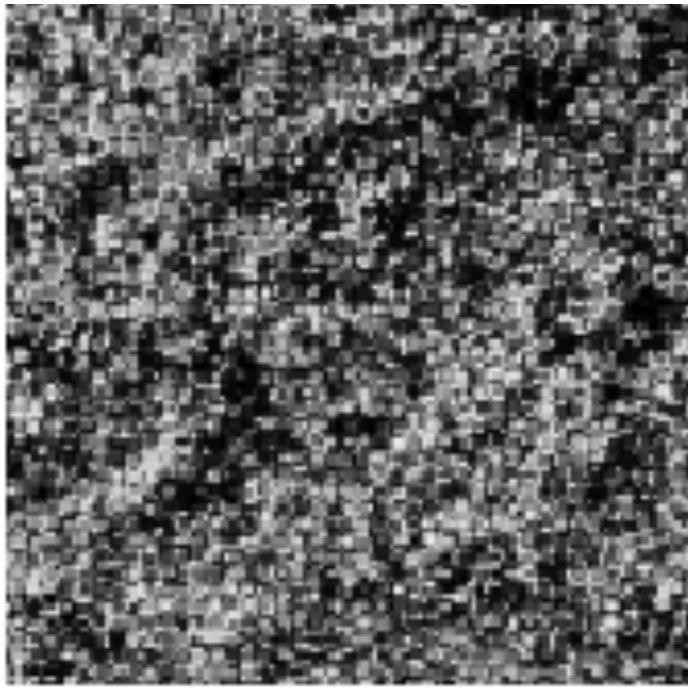
Technically:

- Representation-wise, we want a feature one step above "building blocks" of filters, edges.

Natural Textures



grass



leaves

What/Where are the texels?

The Case for Statistical Texture

- Segmenting out texels is difficult or impossible in real images.
- Numeric quantities or statistics that describe a texture can be computed from the grayscale (or colors) images alone.
- This approach is less intuitive, but is computationally efficient.
- It can be used for both classification and segmentation.

Some Simple Statistical Texture Measures

Edge Density and Direction

- Use an edge detector as the first step in texture analysis.
- The number of edge pixels in a fixed-size region tells us how busy that region is.
- The directions of the edges also help characterize the texture

Edge-based Texture Measures

1. edgeness per unit area

$$F_{edgeness} = |\{ p \mid gradient\_magnitude(p) \geq threshold \}| / N$$

where N is the size of the unit area

2. edge magnitude and direction histograms

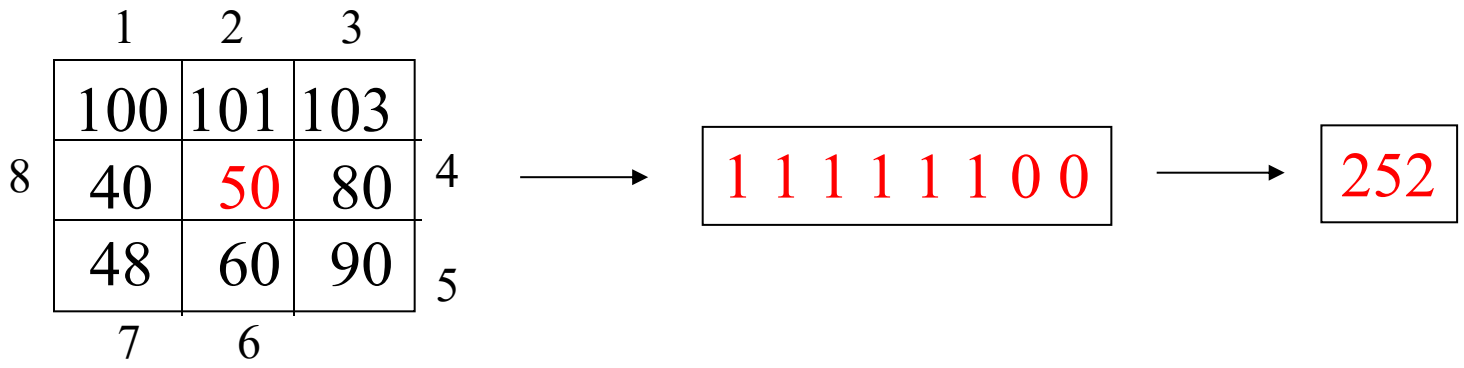
$$F_{magdir} = (H_{magnitude}, H_{direction})$$

where these are the normalized histograms of gradient magnitudes and gradient directions, respectively.

Local Binary Pattern Measure

- For each pixel p, create an 8-bit number $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8$, where $b_i = 0$ if neighbor *i* has value less than or equal to p's value and 1 otherwise.

- Represent the texture in the image (or a region) by the histogram of these numbers.

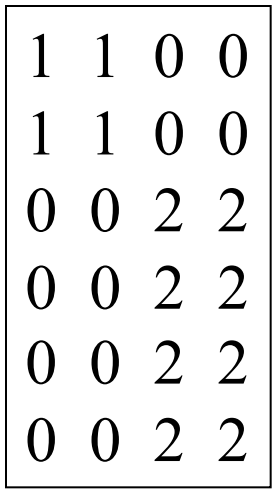


Co-occurrence Matrix Features

A co-occurrence matrix is a 2D array C in which

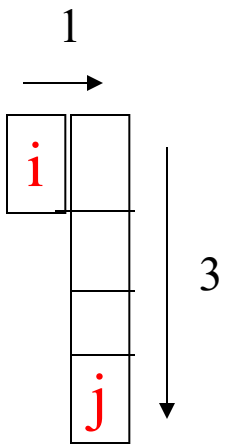
- Both the rows and columns represent a set of possible image values.
- $C_d(i,j)$ indicates how many times value i co-occurs with value j in a particular spatial relationship d.
- The spatial relationship is specified by a vector d = (dr,dc).

Co-occurrence Example



1	1	0	0
1	1	0	0
0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

grayscale image



d = (3,1)

	0	1	2
0	1	0	3
1	2	0	2
2	0	0	1

C_d

co-occurrence matrix

From C_d we can compute N_d, the normalized co-occurrence matrix, where each value is divided by the sum of all the values.

Co-occurrence Features

- extract numerical features from the co-occurrence matrix which can be used for representing and comparing textures

$$
\begin{aligned}
Energy &= \sum_i \sum_j N_d^2(i,j) \\
Entropy &= -\sum_i \sum_j N_d(i,j) log_2 N_d(i,j) \\
Contrast &= \sum_i \sum_j (i-j)^2 N_d(i,j) \\
Homogeneity &= \sum_i \sum_j \frac{N_d(i,j)}{1+\mid i-j \mid} \\
Correlation &= \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_d(i,j)}{\sigma_i \sigma_j}
\end{aligned}
$$

where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column

But how do you choose d?

- This is actually a critical question with **all** the statistical texture methods.
- Are the "texels" tiny, medium, large, all three …?
- Not really a solved problem.

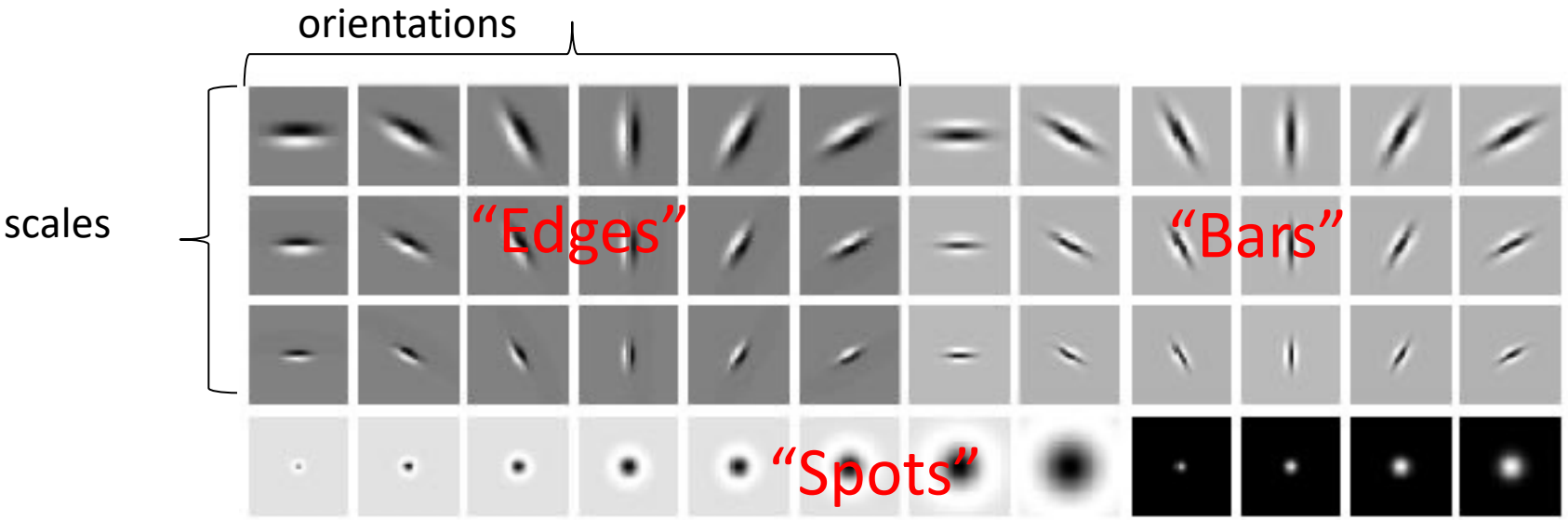
Alternative approach

Signal-processing-based algorithms use texture filters applied to the image to create filtered images from which texture features are computed.

Textures are made up of repeated local patterns, so:

- Find the patterns
  - Use filters that look like patterns (spots, bars, raw patches…)
  - Consider magnitude of response
- Describe their statistics within each local window, e.g.,
  - Mean, standard deviation
  - Histogram
  - Histogram of "prototypical" feature occurrences

Filter banks

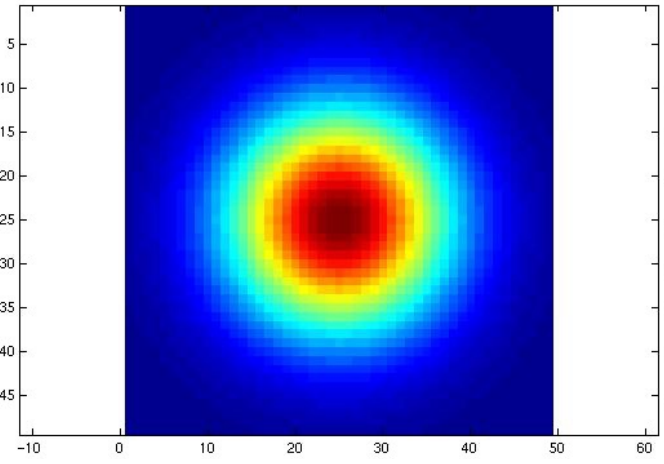


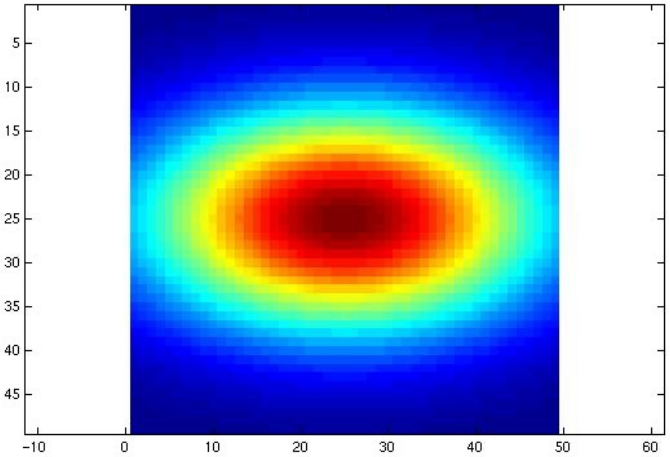
- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

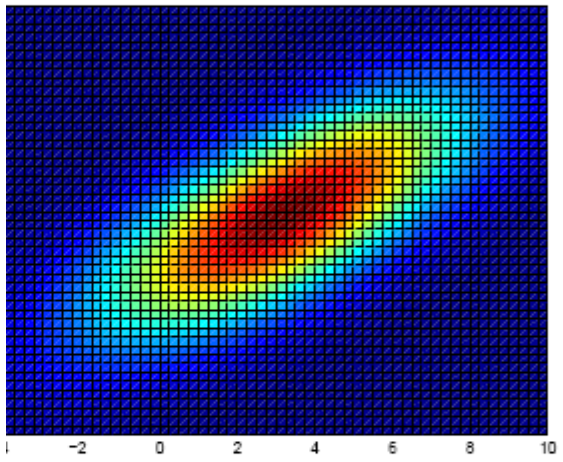
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Multivariate Gaussian

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$



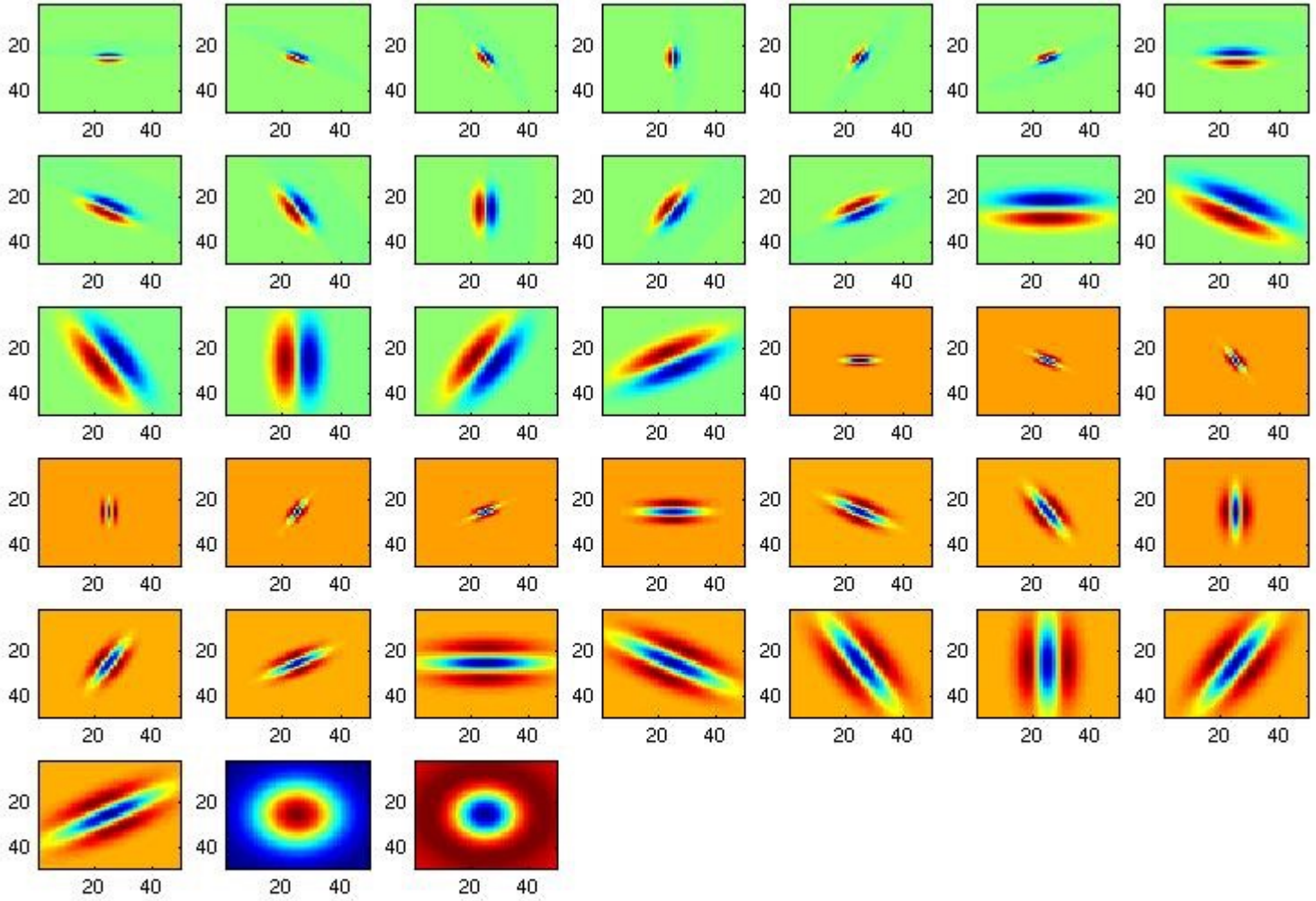




$$\Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

Slide credit: Kristen Grauman

Filter bank

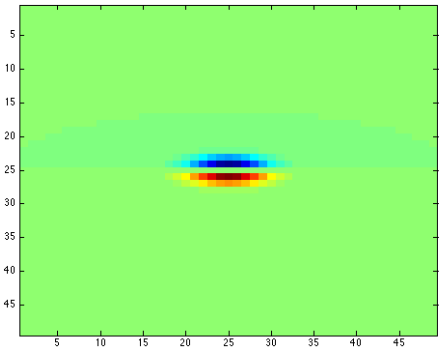


Slide credit: Kristen Grauman

Initial images

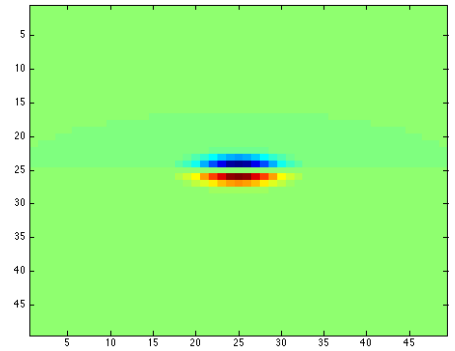
Slide adaptat după K. Grauman

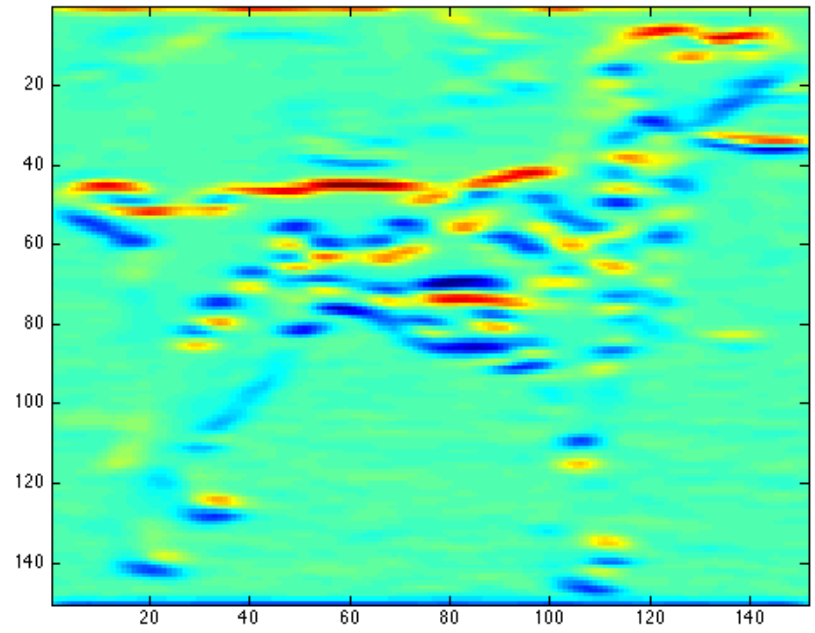
Initial images + filter

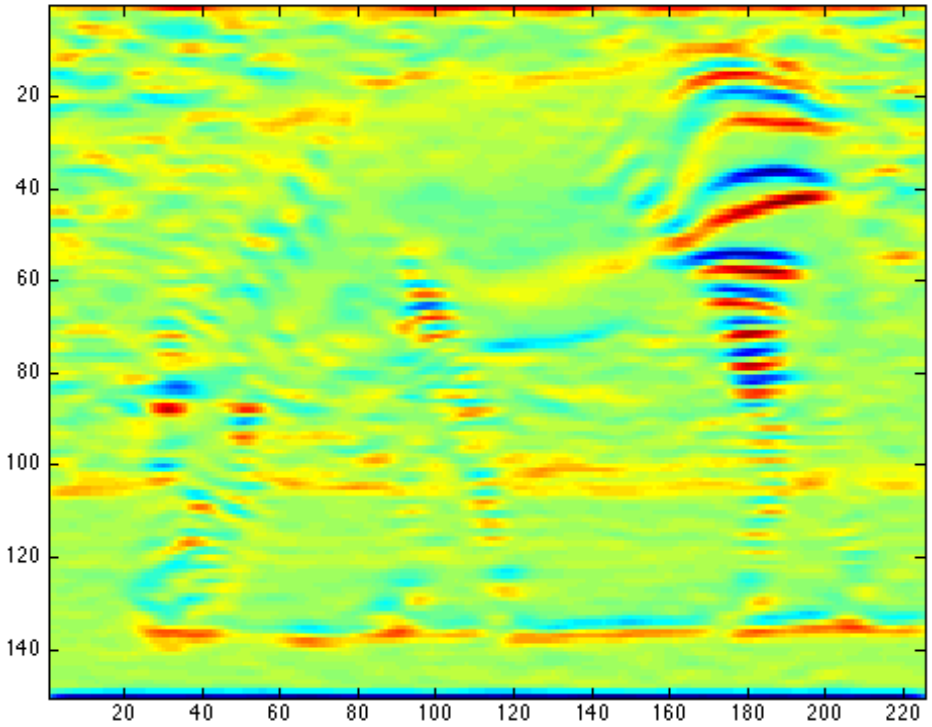


Slide adaptat după K. Grauman

Filtered images

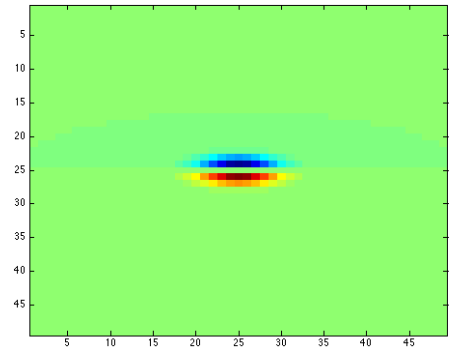


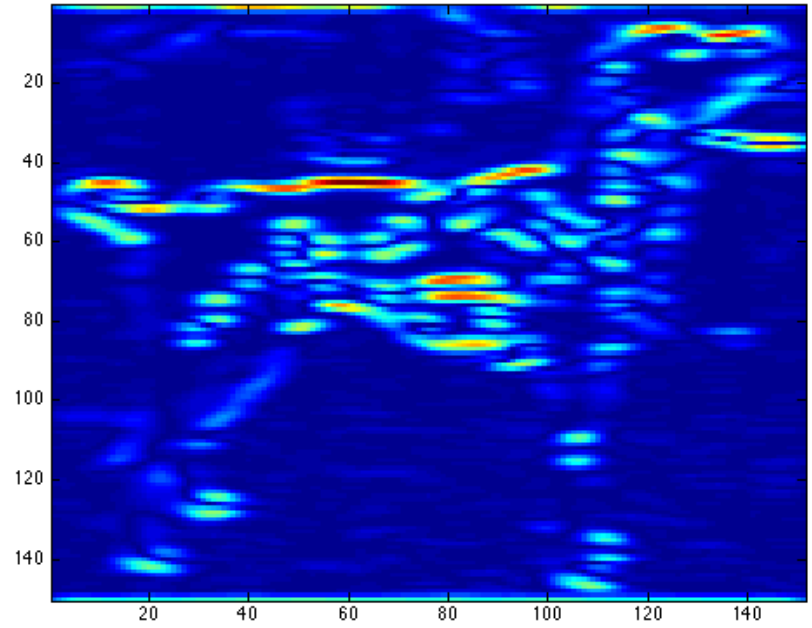


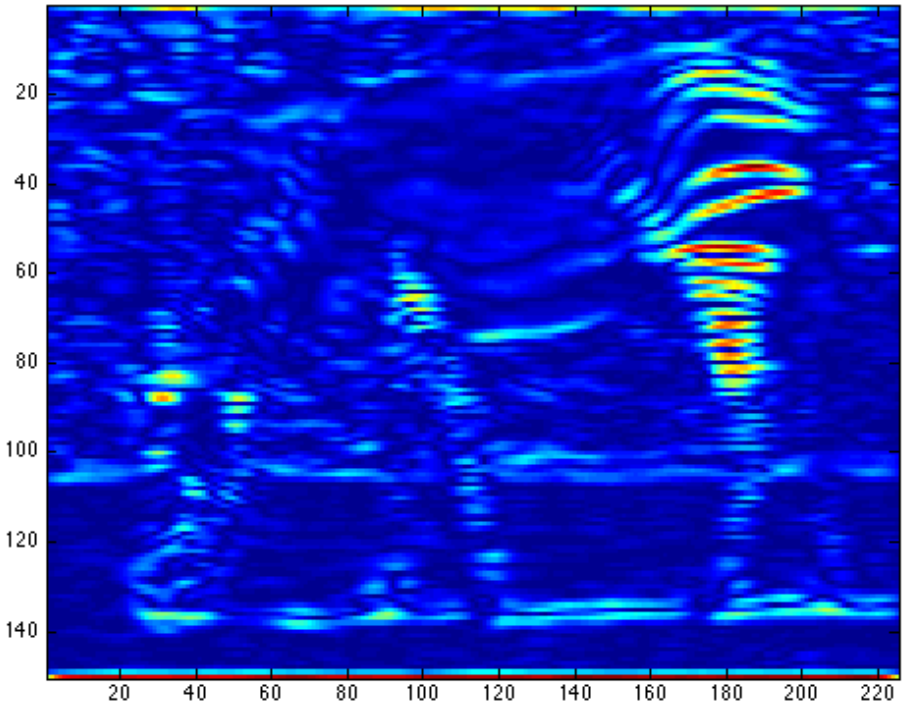


Slide adaptat după K. Grauman

Filtered images with absolute response

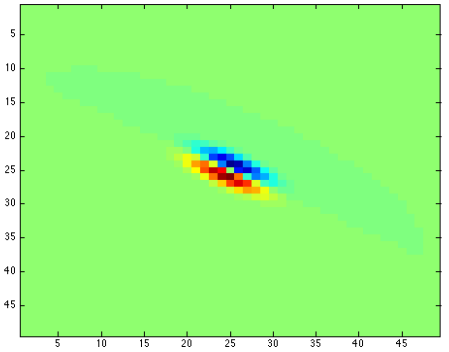






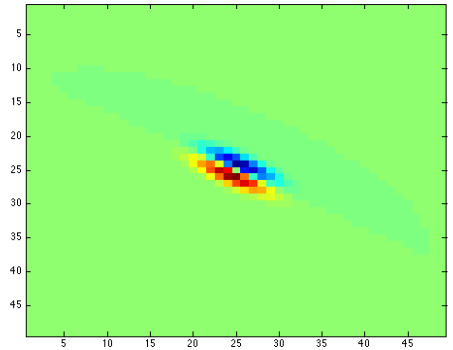
Slide adaptat după K. Grauman

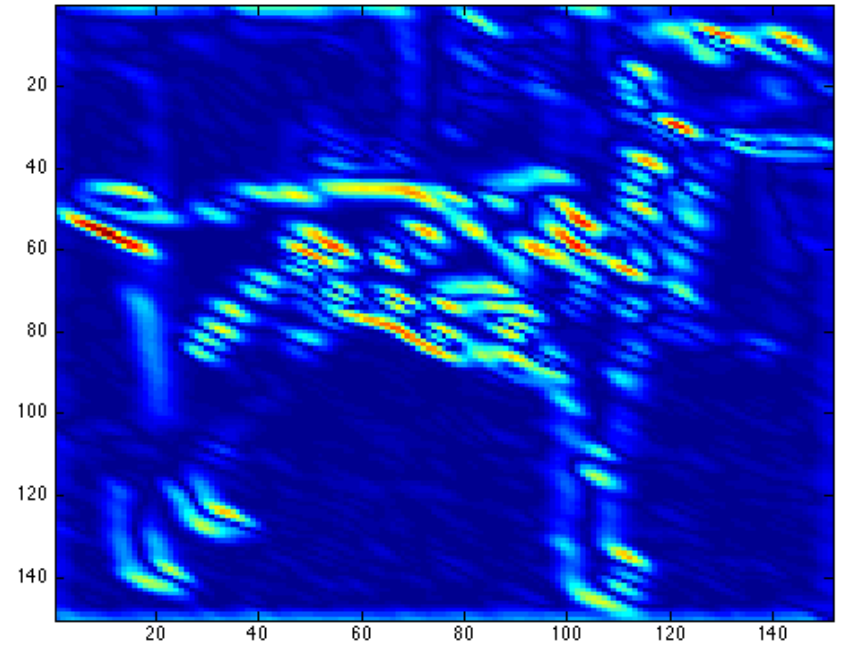
Initial images + filter

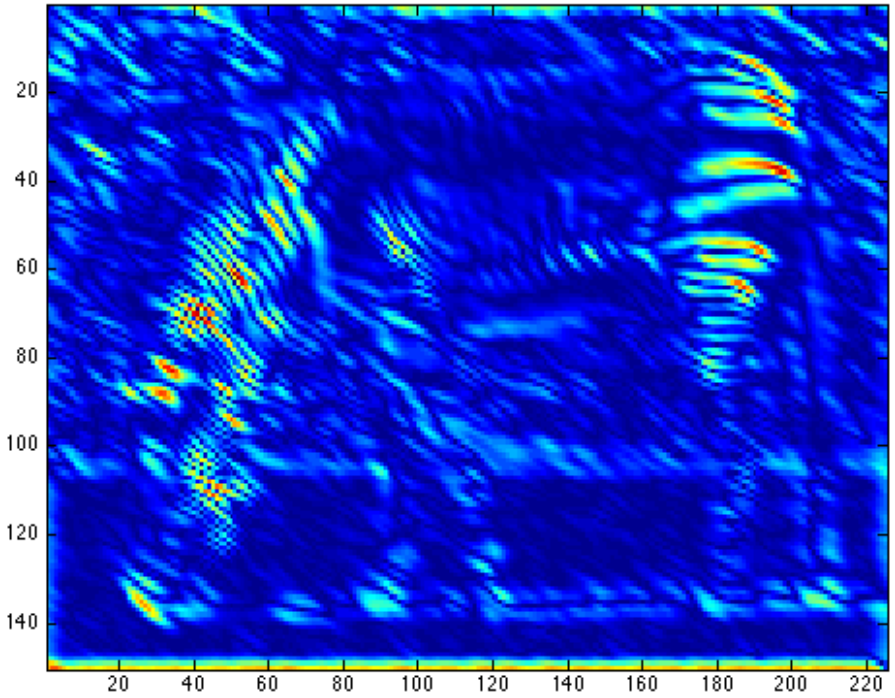


Slide adaptat după K. Grauman

Filtered images with absolute response

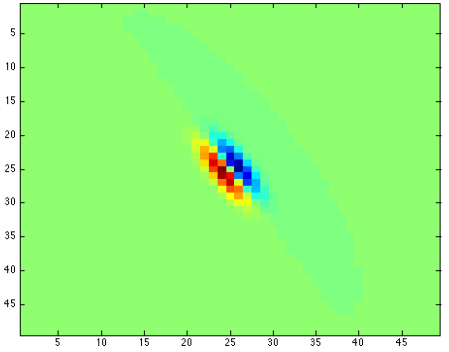






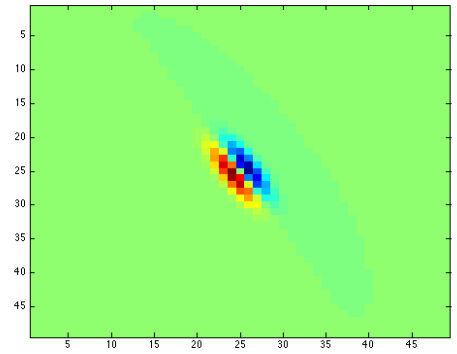
Slide adaptat după K. Grauman

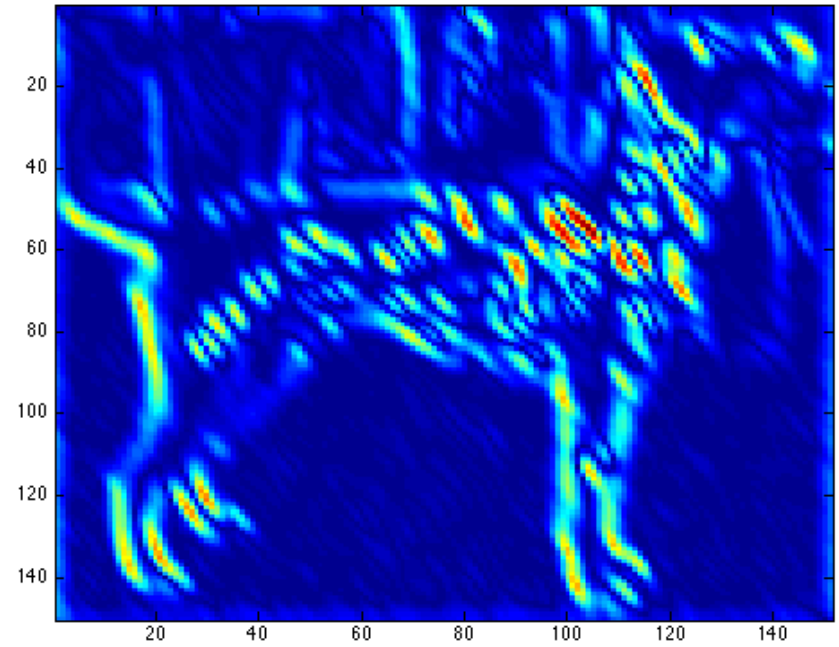
Initial images + filter

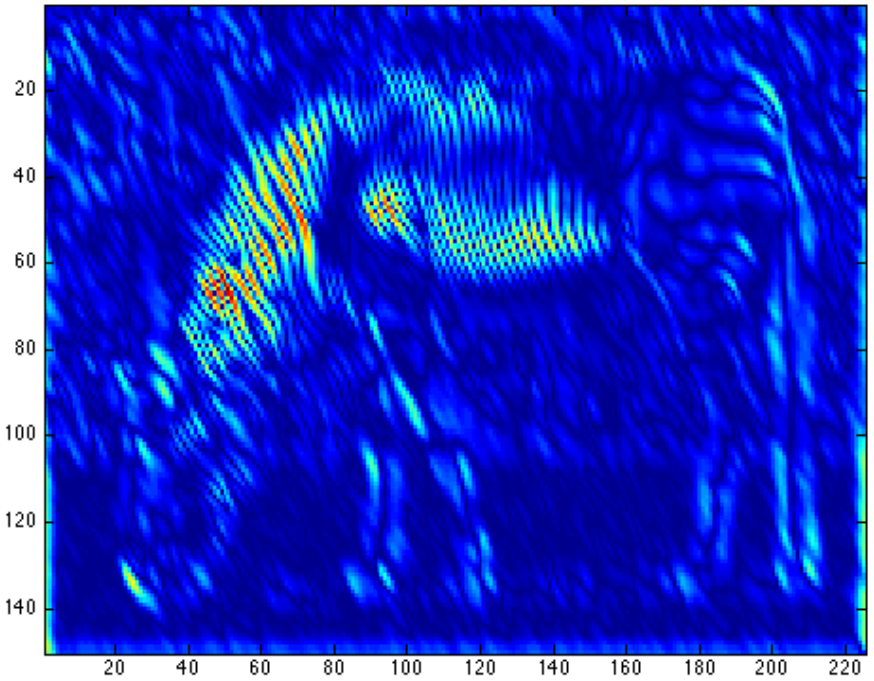


Slide adaptat după K. Grauman

Filtered images with absolute response

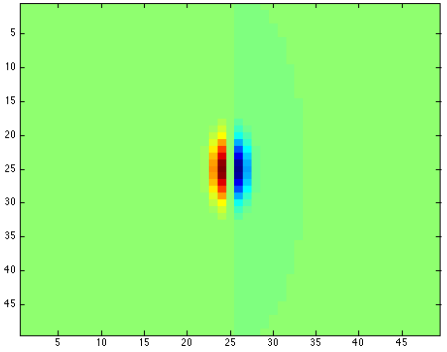






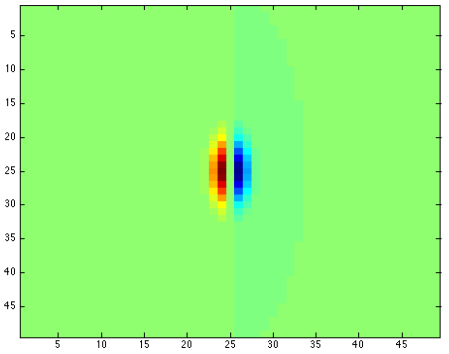
Slide adaptat după K. Grauman

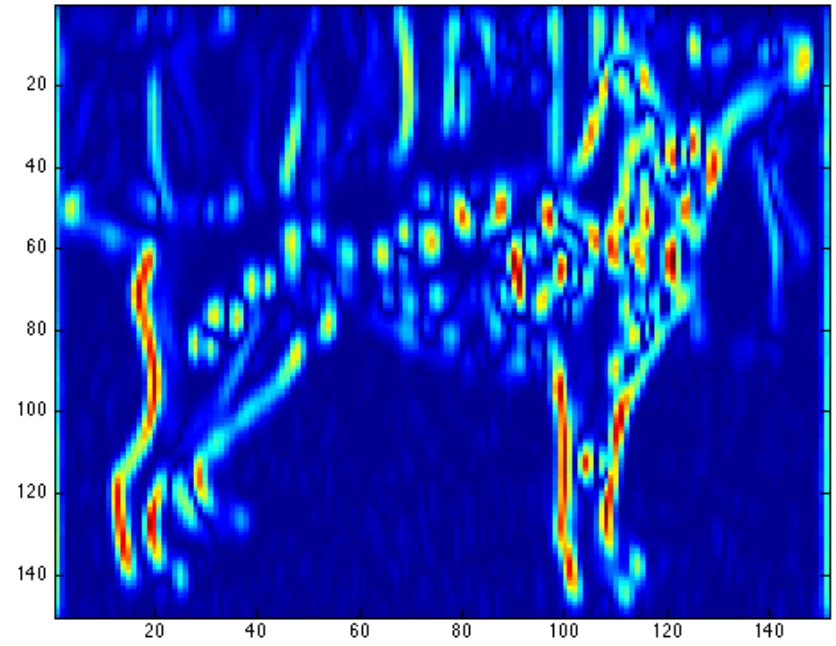
Initial images + filter

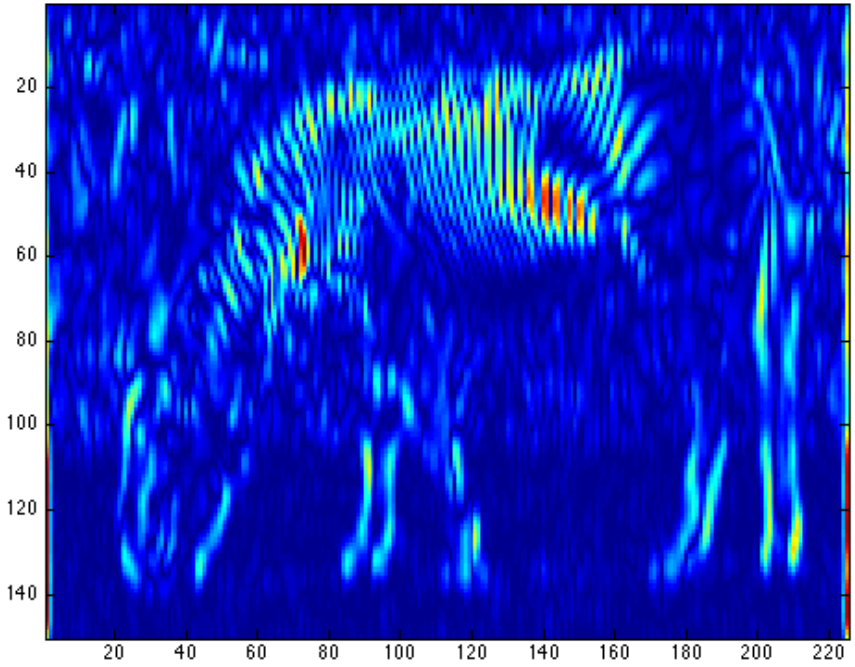


Slide adaptat după K. Grauman

Filtered images with absolute response

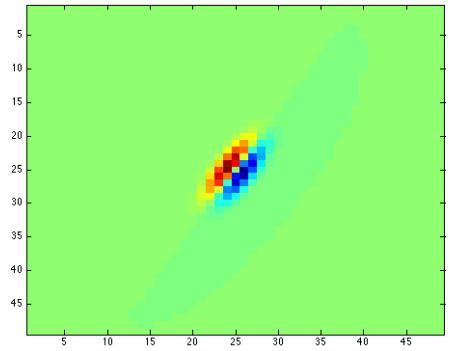






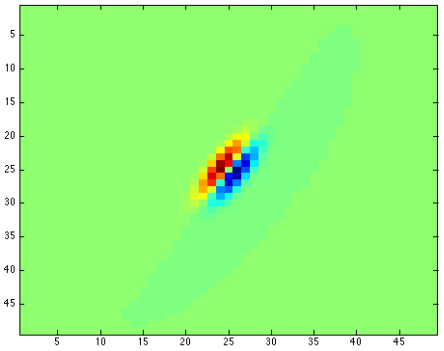
Slide adaptat după K. Grauman

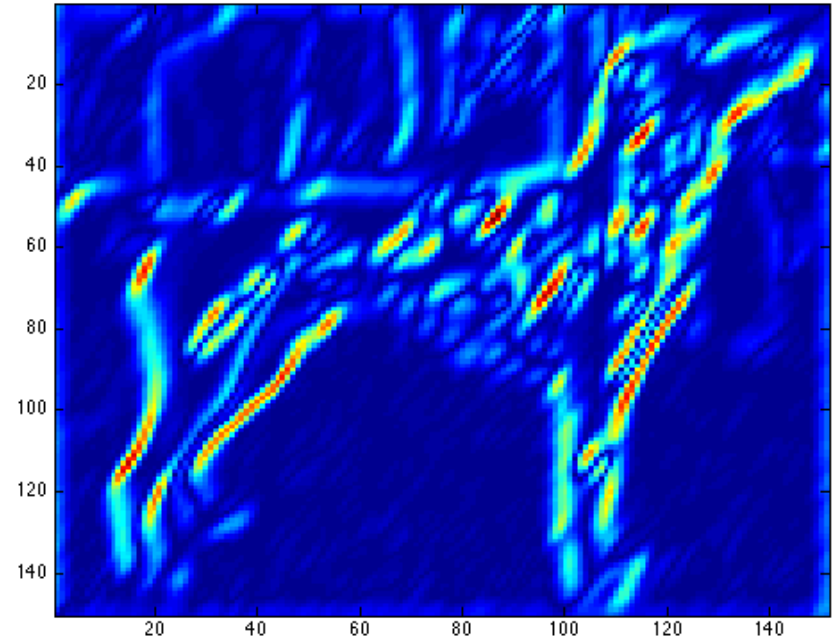
Initial images + filter

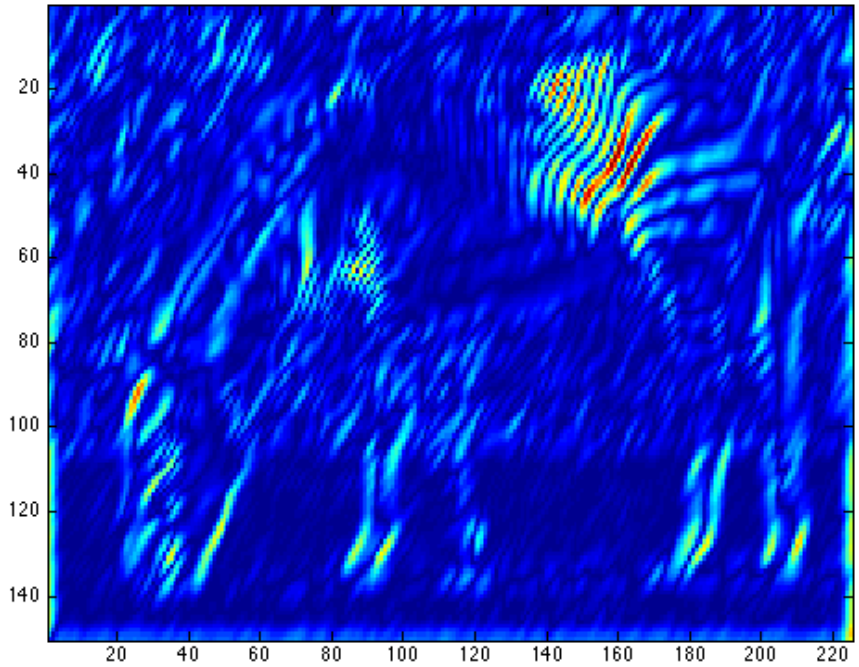


Slide adaptat după K. Grauman

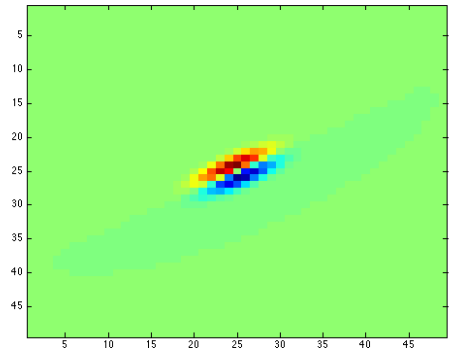
Filtered images with absolute response





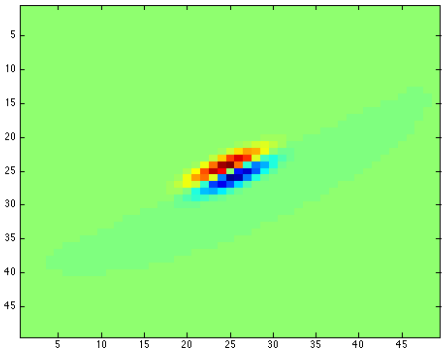


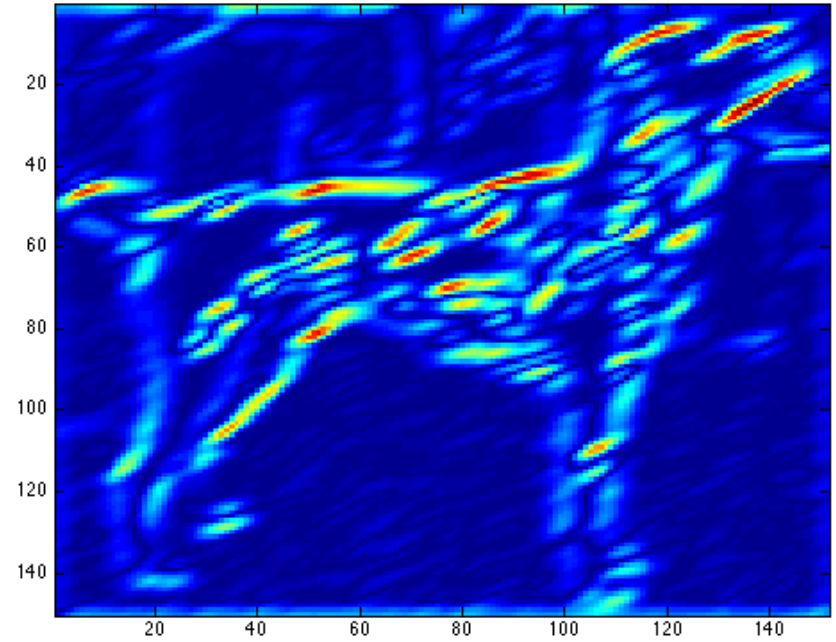
Initial images + filter

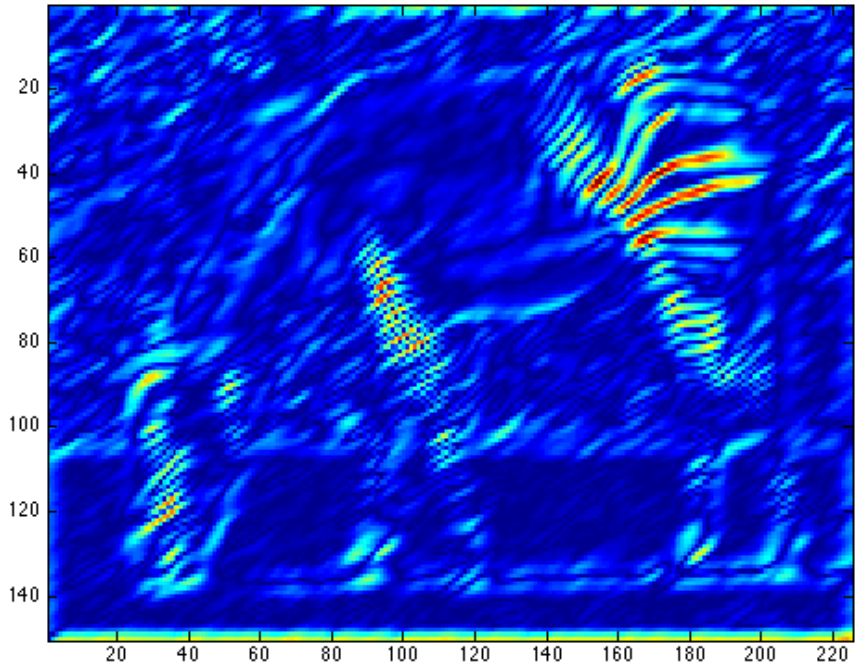


Slide adaptat după K. Grauman

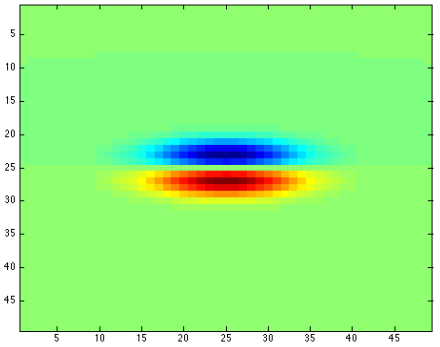
Filtered images with absolute response





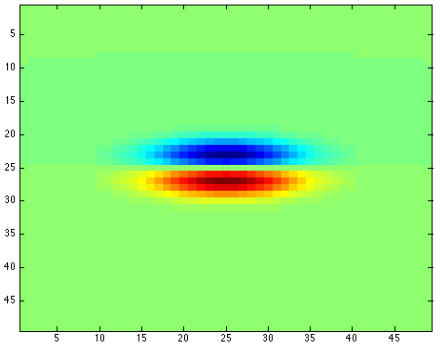


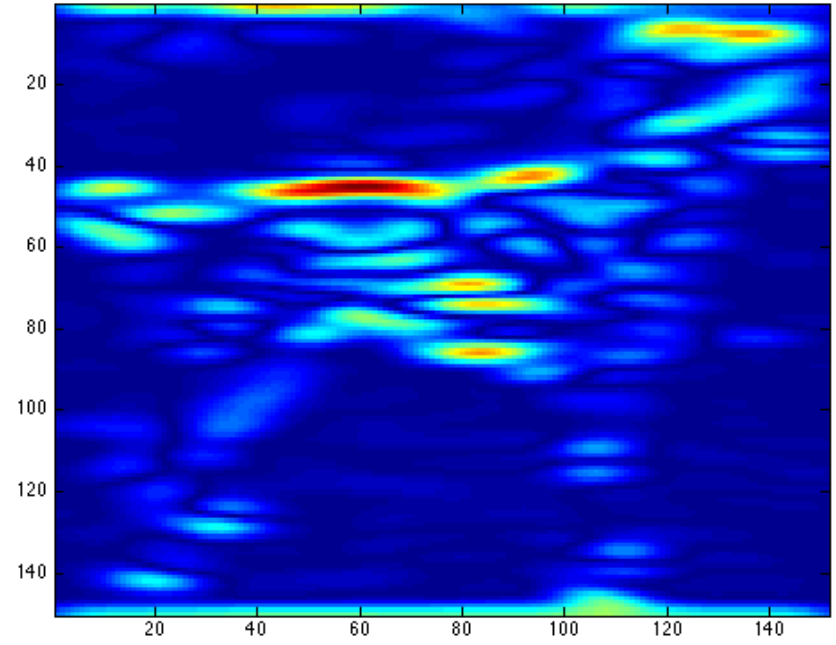
Initial images + filter

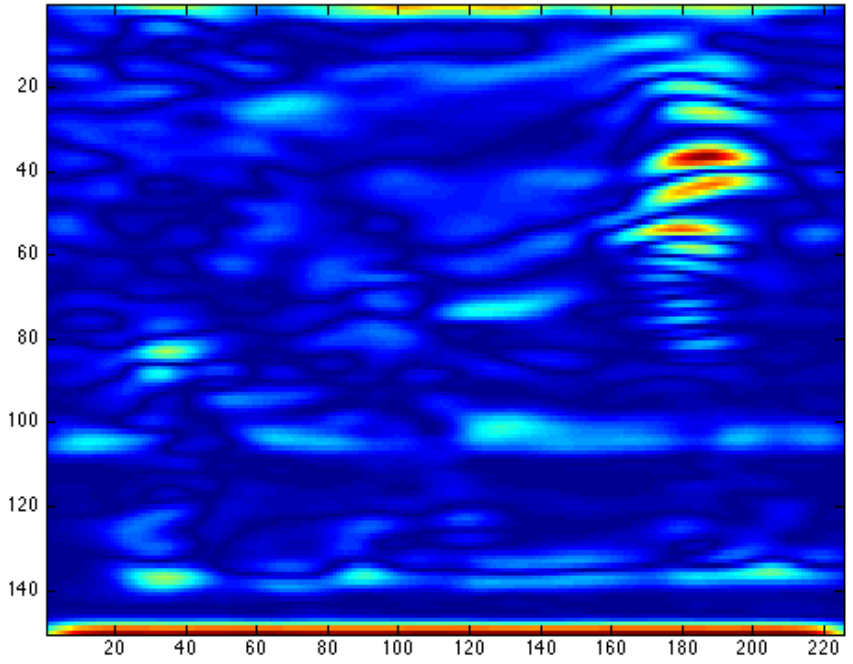


Slide adaptat după K. Grauman

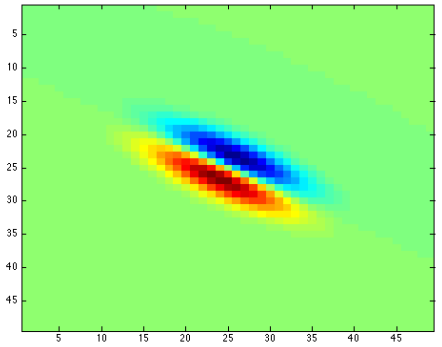
Filtered images with absolute response





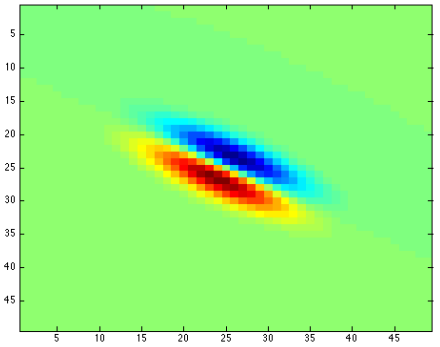


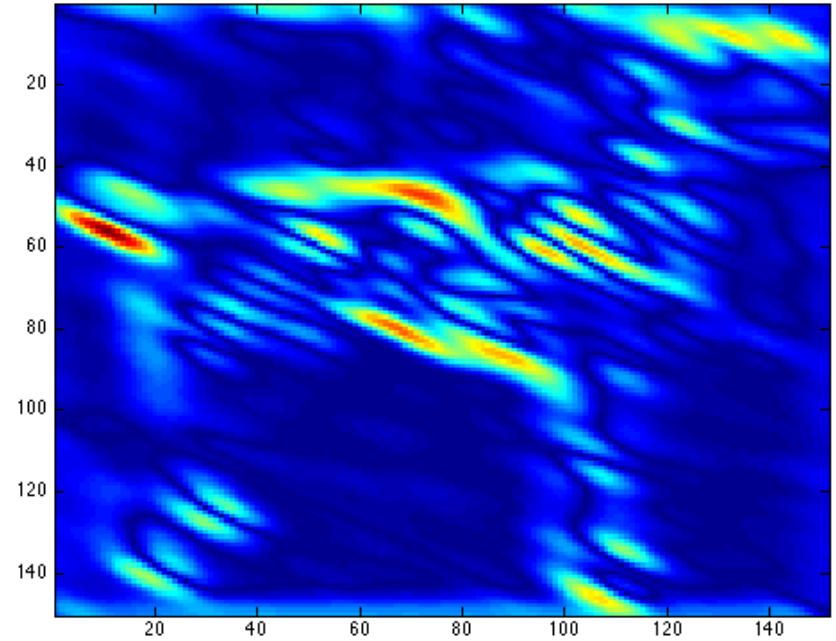
Initial images + filter

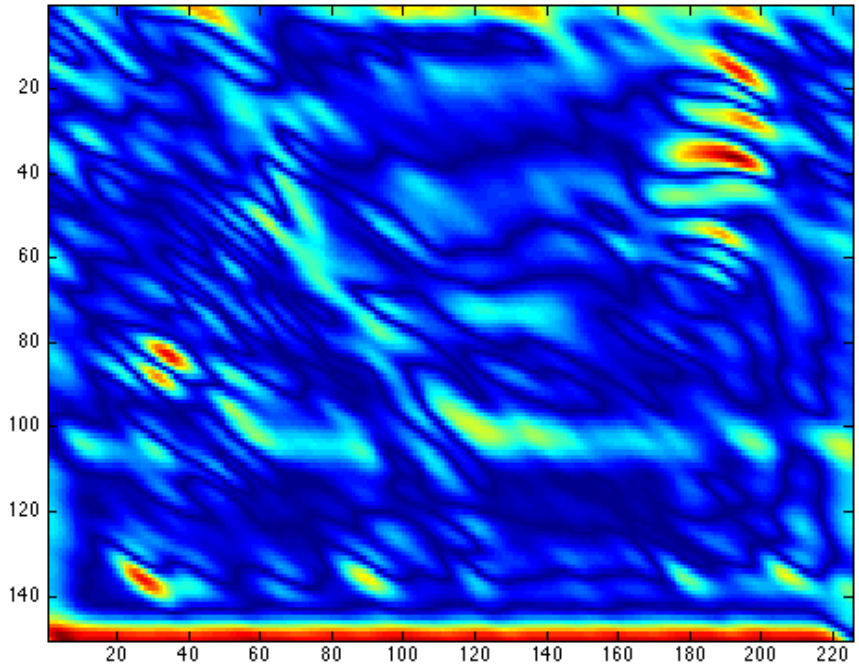


Slide adaptat după K. Grauman

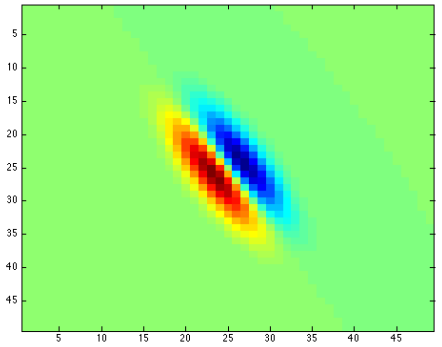
Filtered images with absolute response





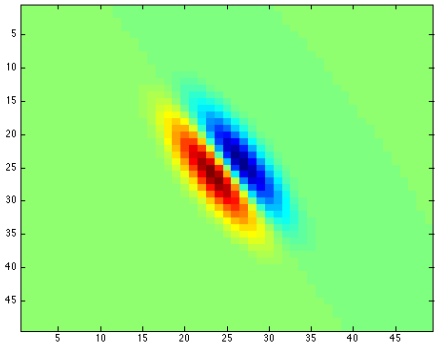


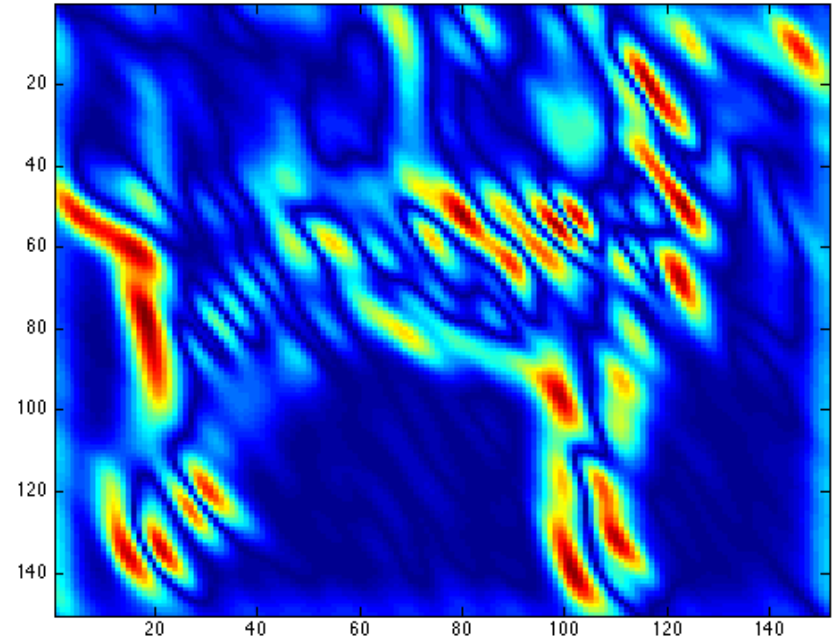
Initial images + filter

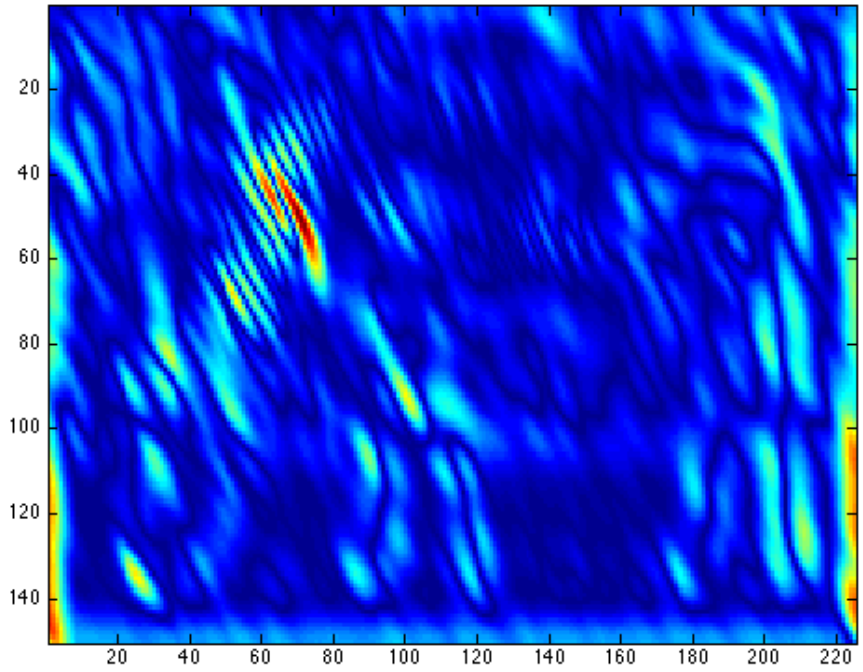


Slide adaptat după K. Grauman

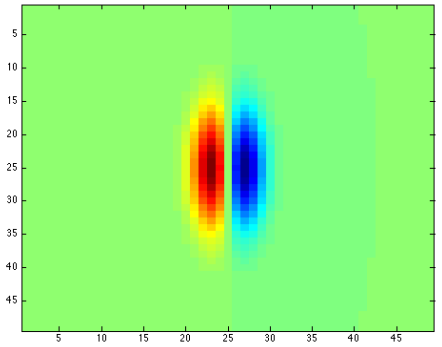
Filtered images with absolute response





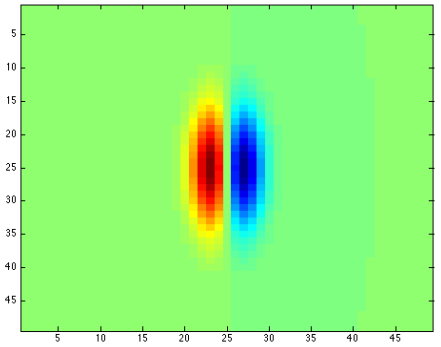


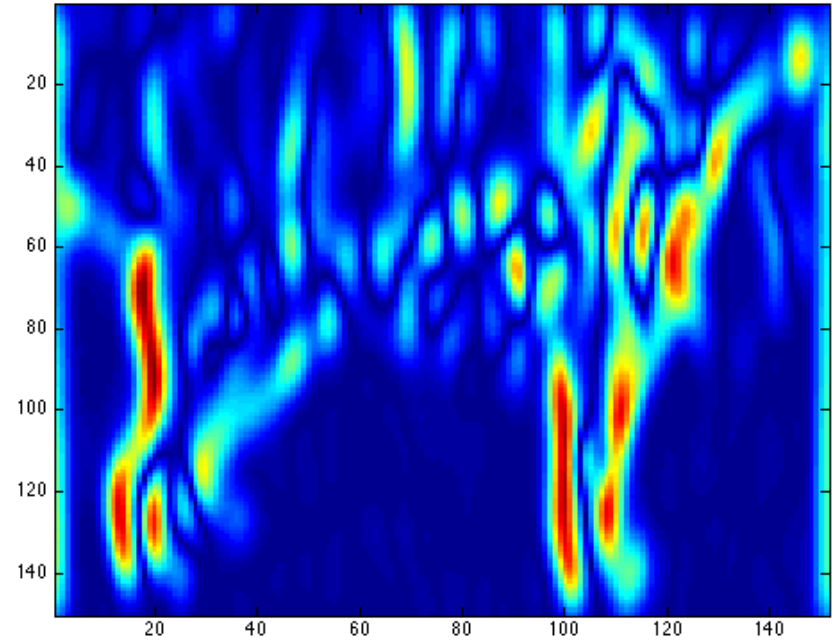
Initial images + filter

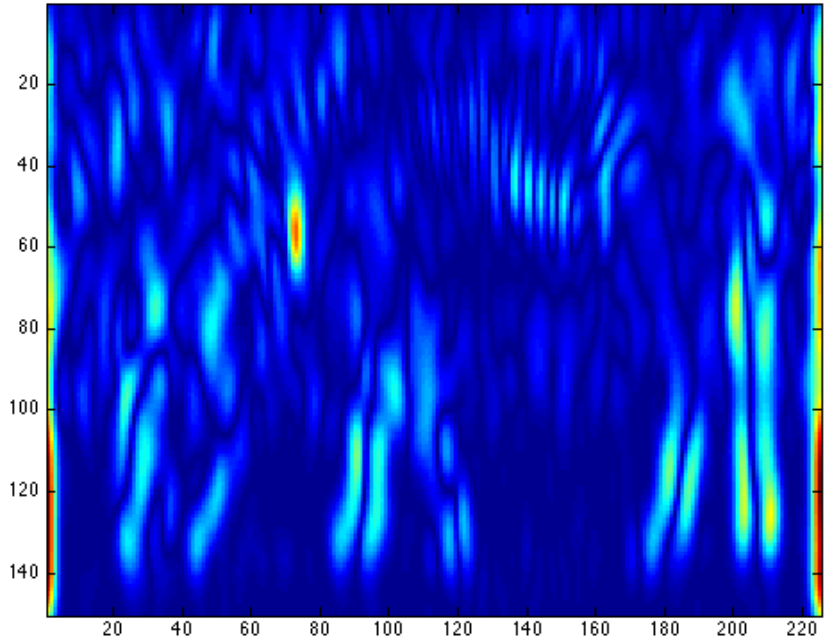


Slide adaptat după K. Grauman

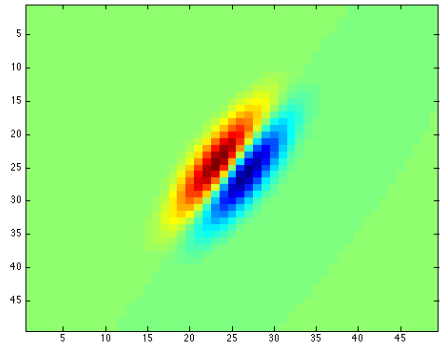
Filtered images with absolute response





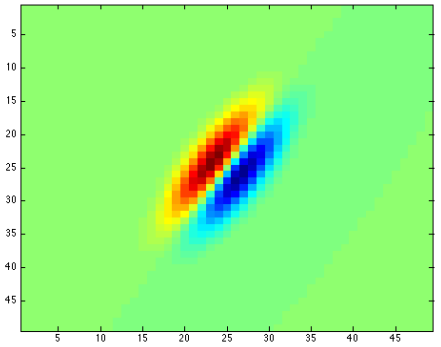


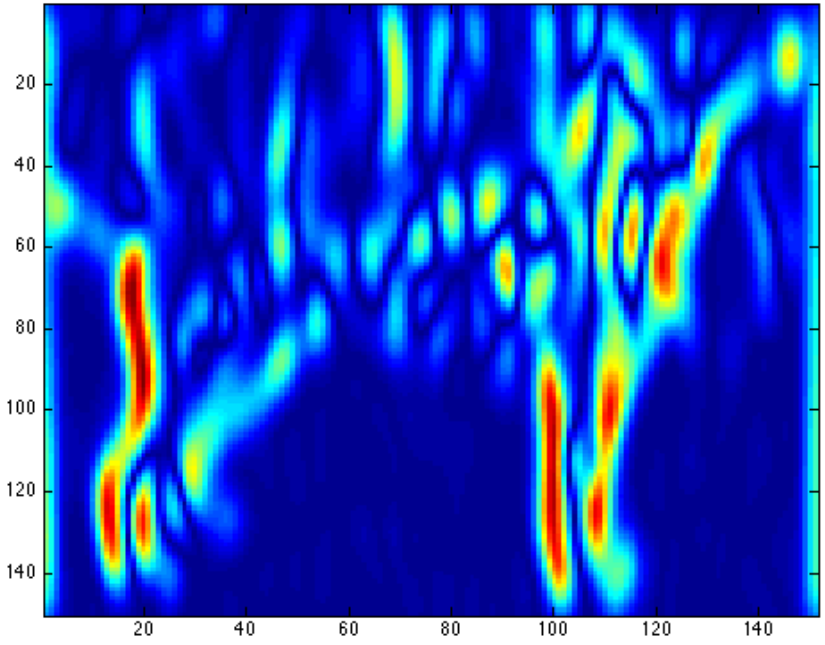
Initial images + filter

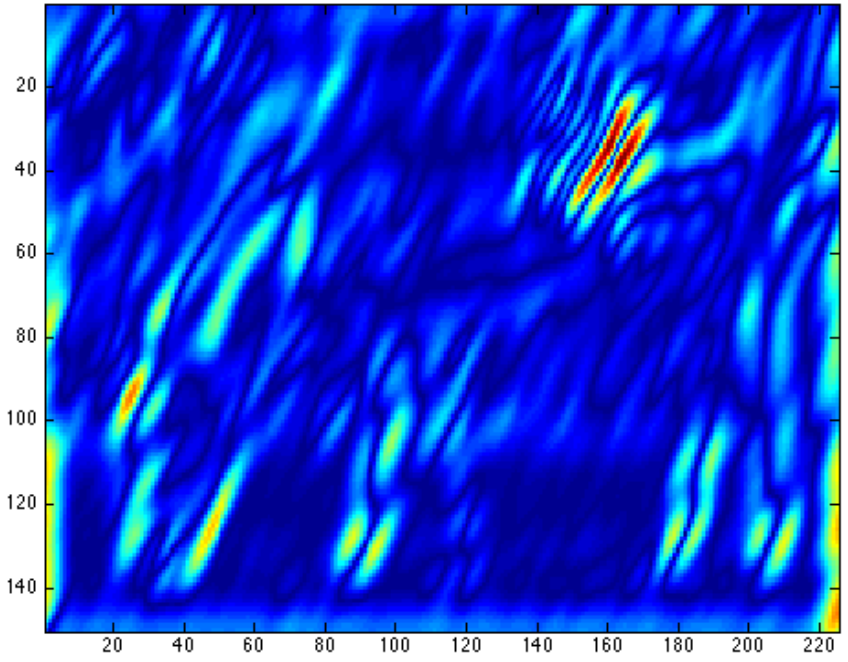


Slide adaptat după K. Grauman

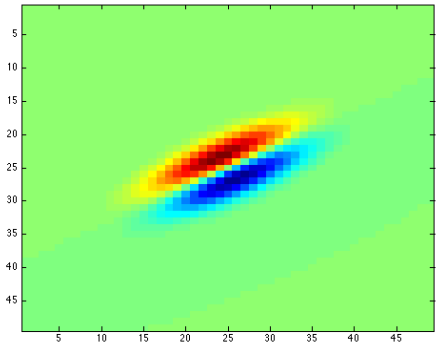
Filtered images with absolute response





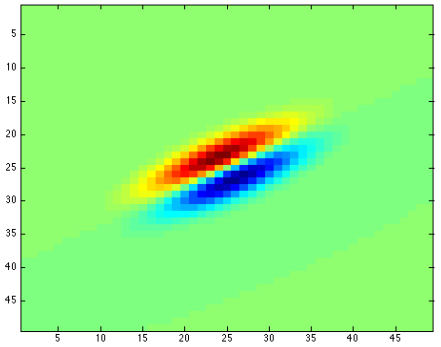


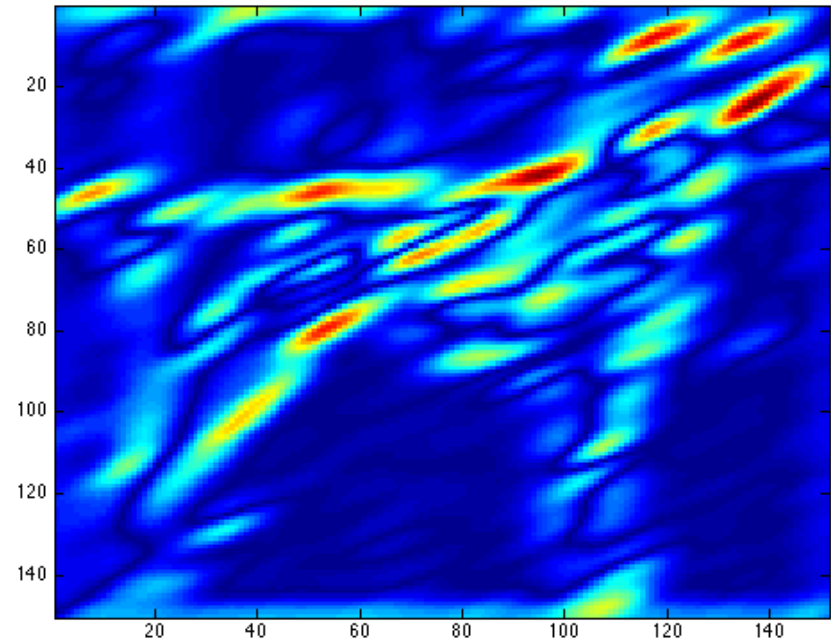
Initial images + filter

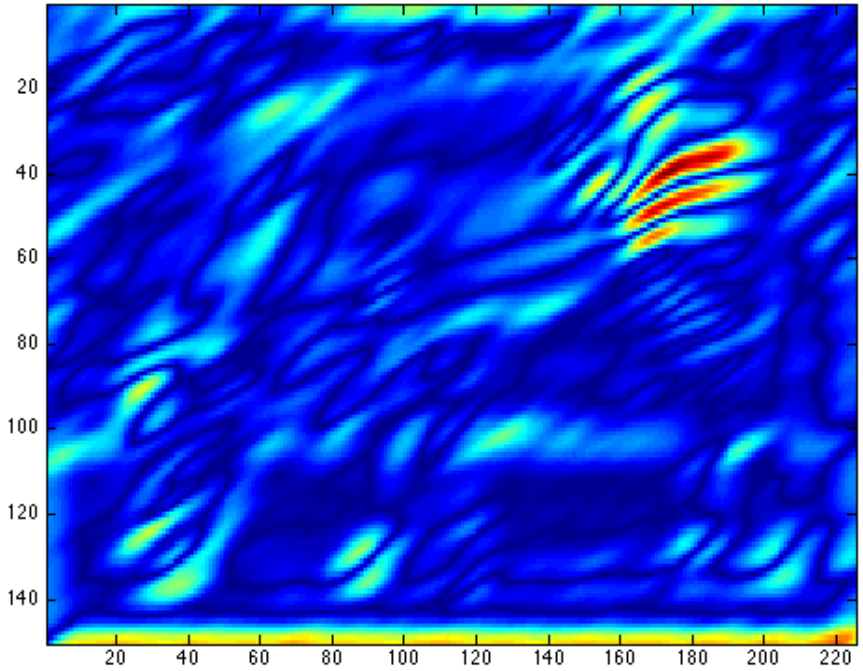


Slide adaptat după K. Grauman

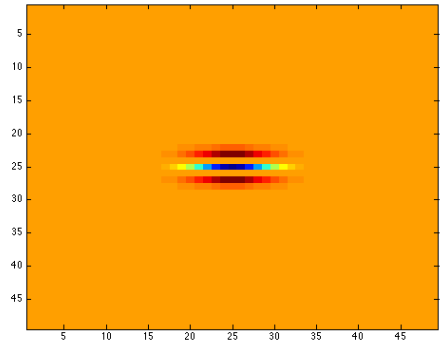
Filtered images with absolute response





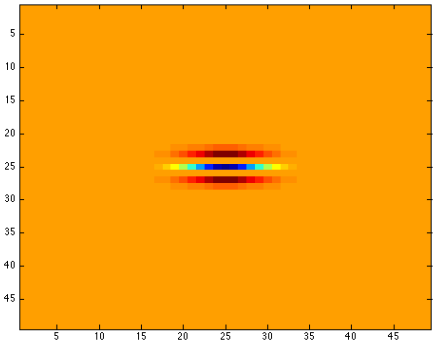


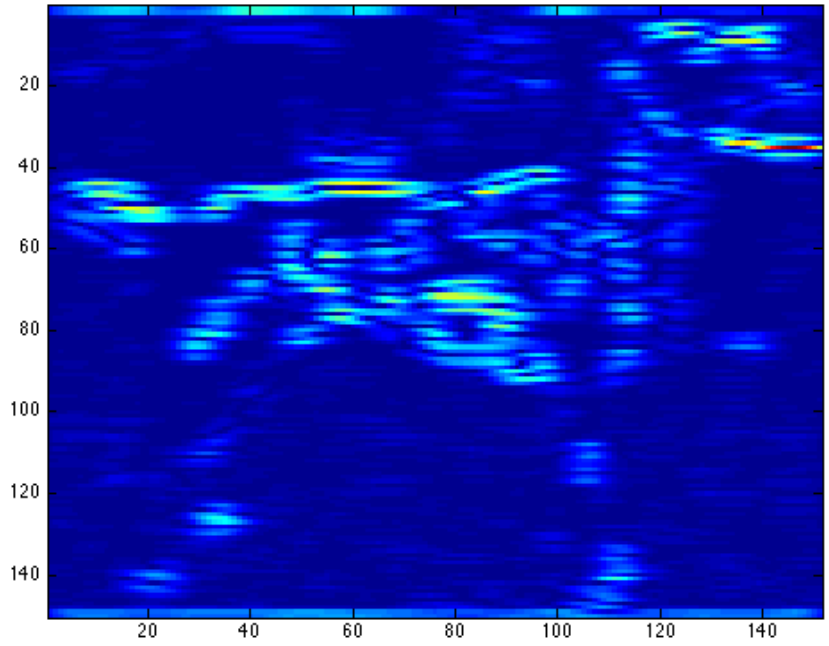
Initial images + filter

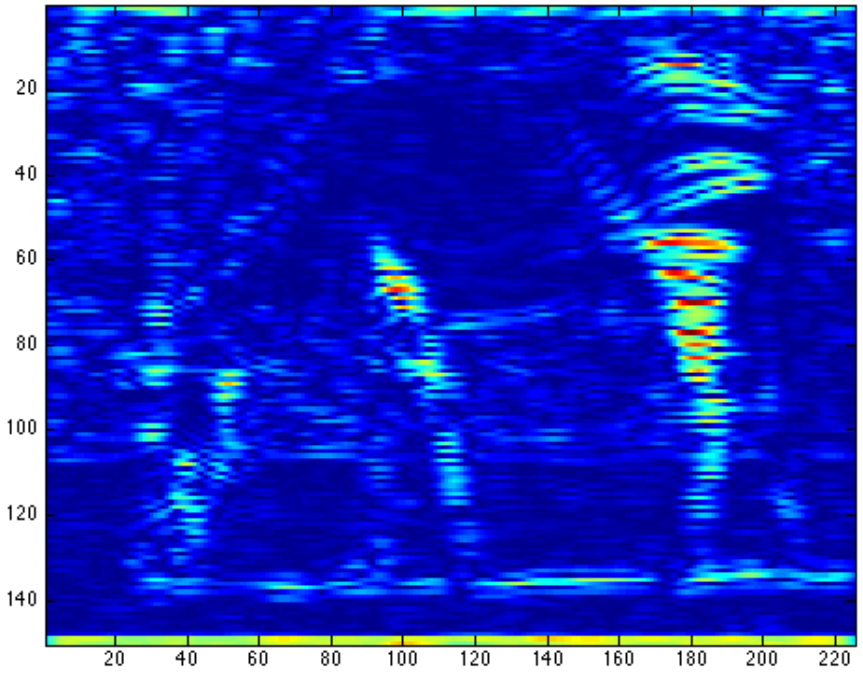


Slide adaptat după K. Grauman

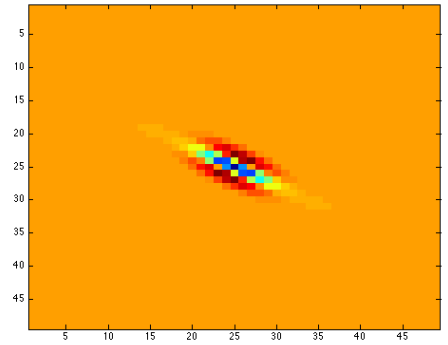
Filtered images with absolute response





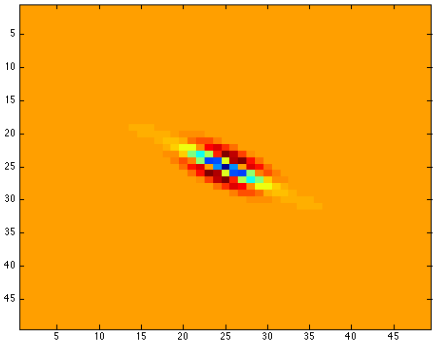


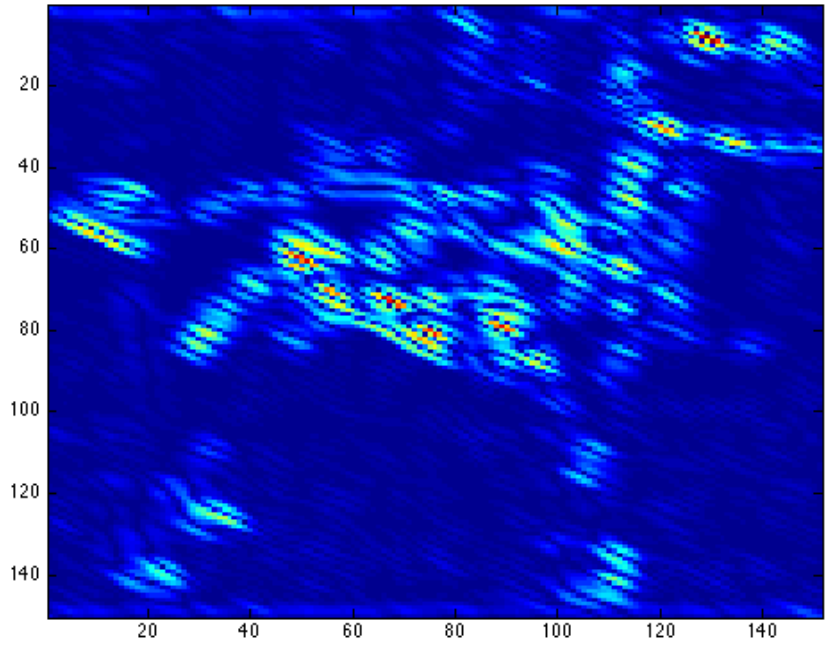
Initial images + filter

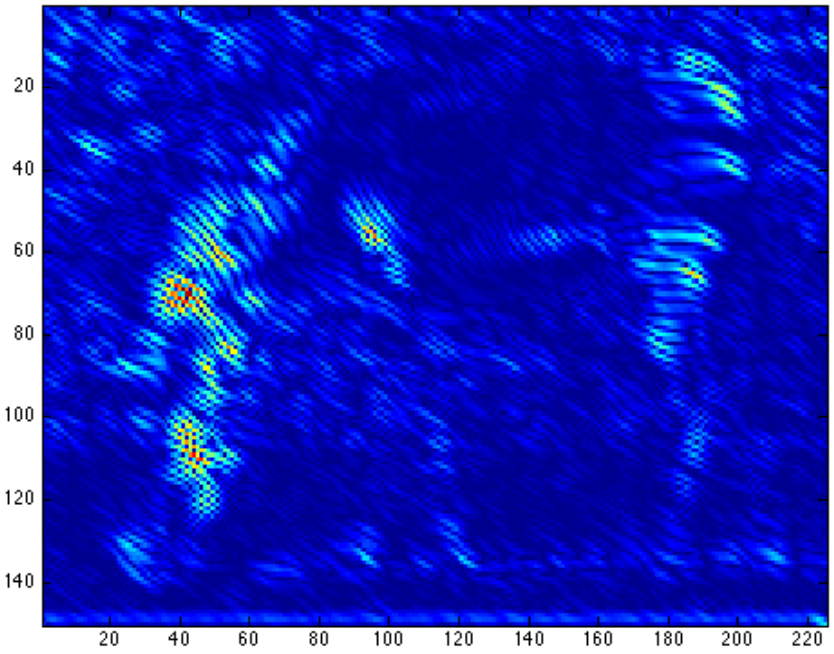


Slide adaptat după K. Grauman

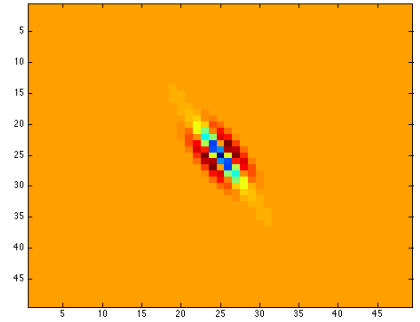
Filtered images with absolute response





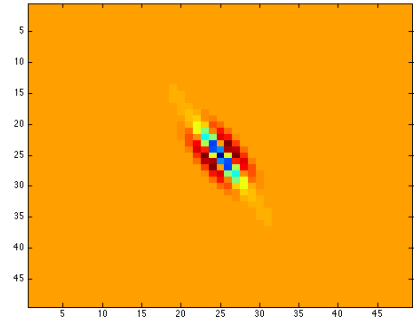


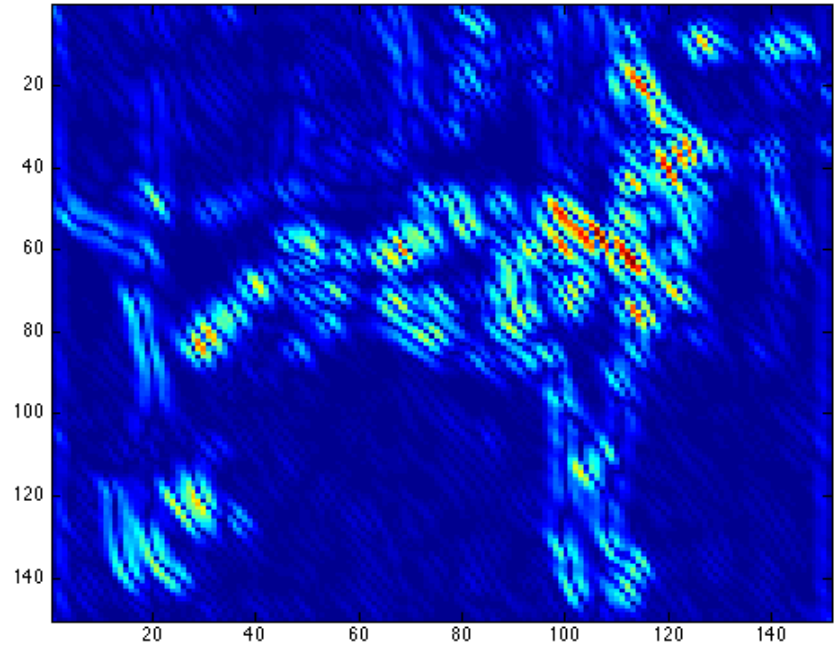
Initial images + filter

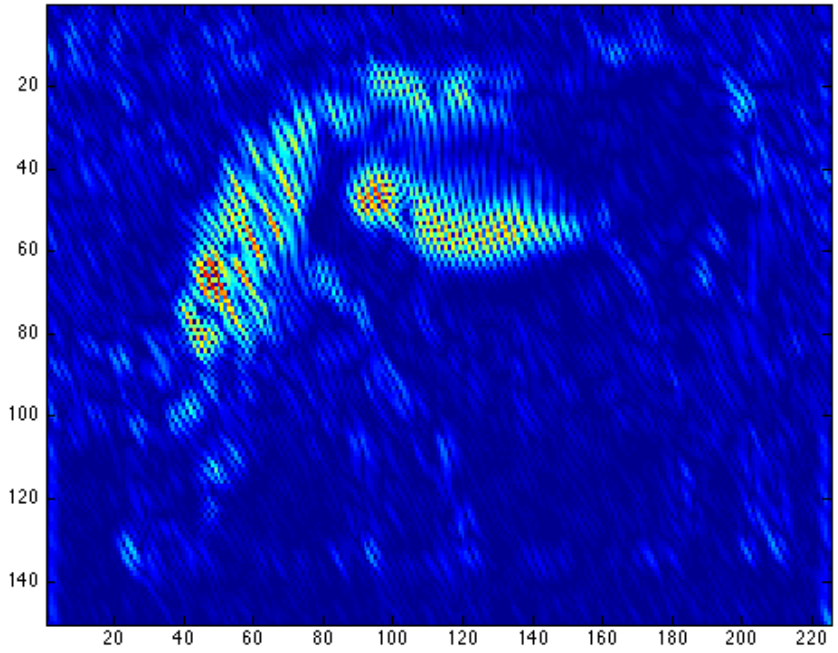


Slide adaptat după K. Grauman

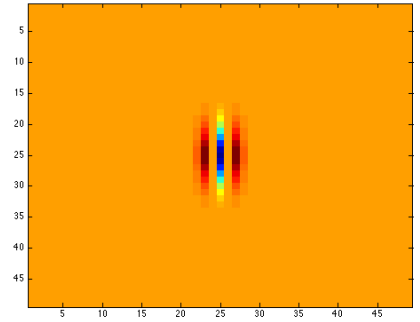
Filtered images with absolute response





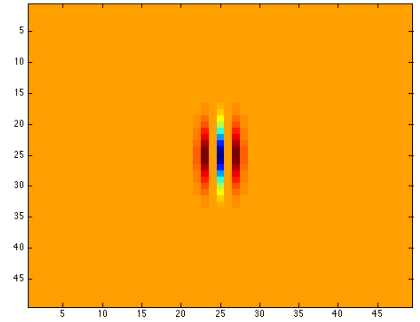


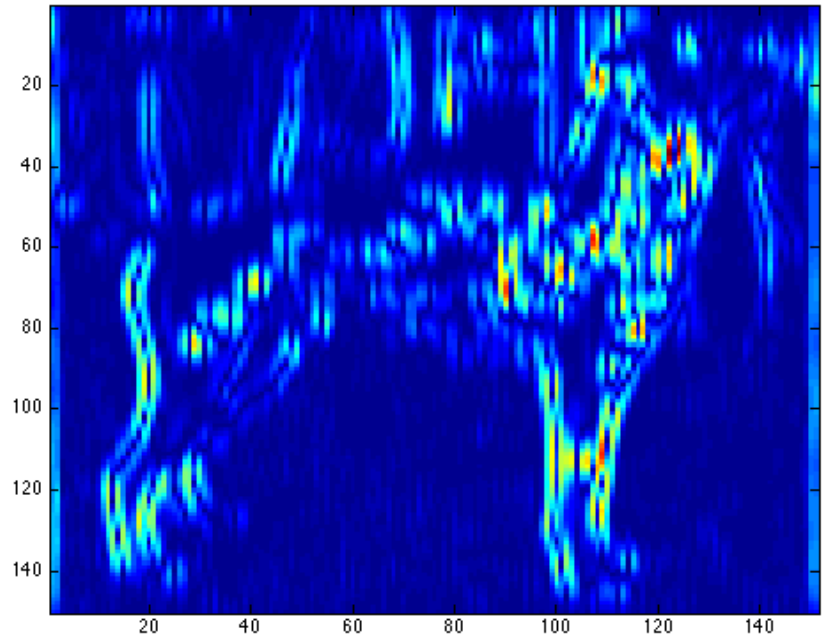
Initial images + filter

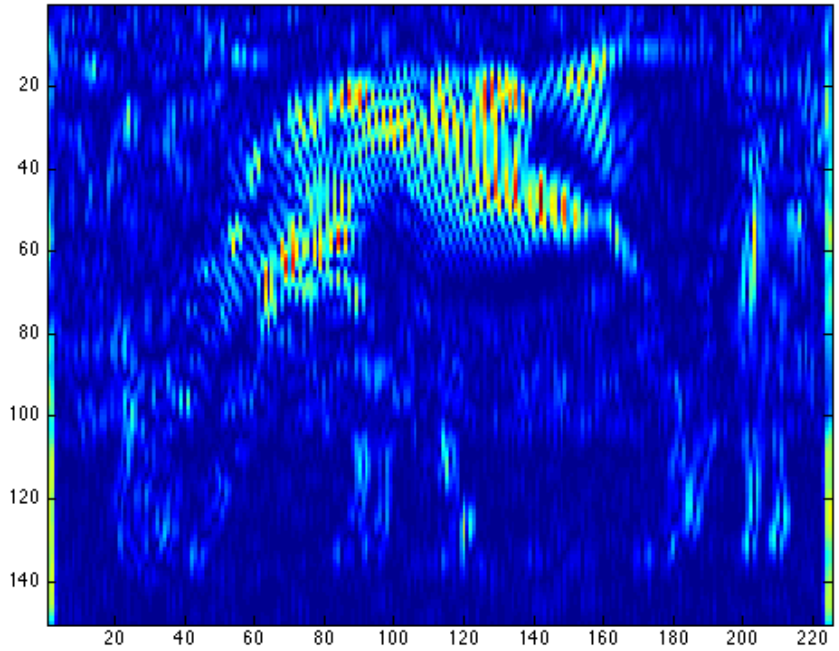


Slide adaptat după K. Grauman

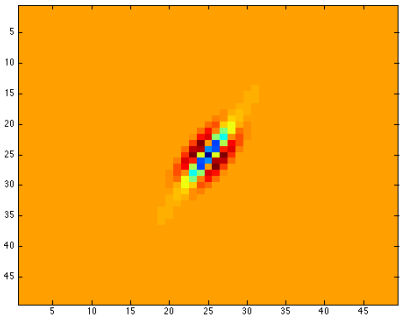
Filtered images with absolute response





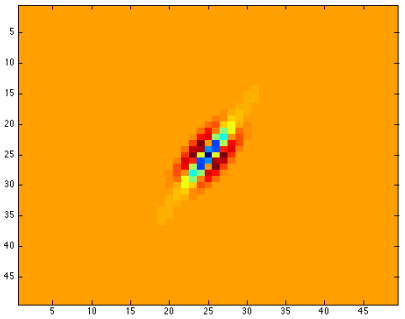


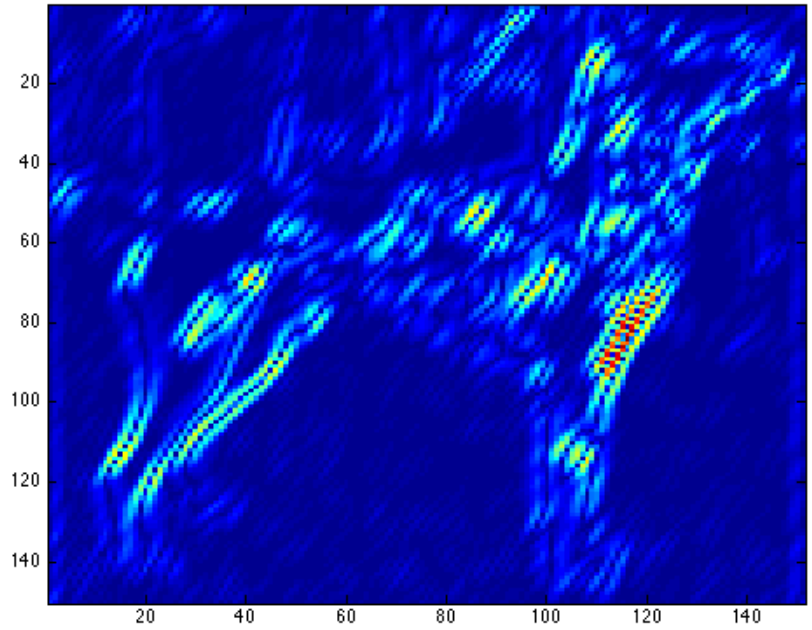
Initial images + filter

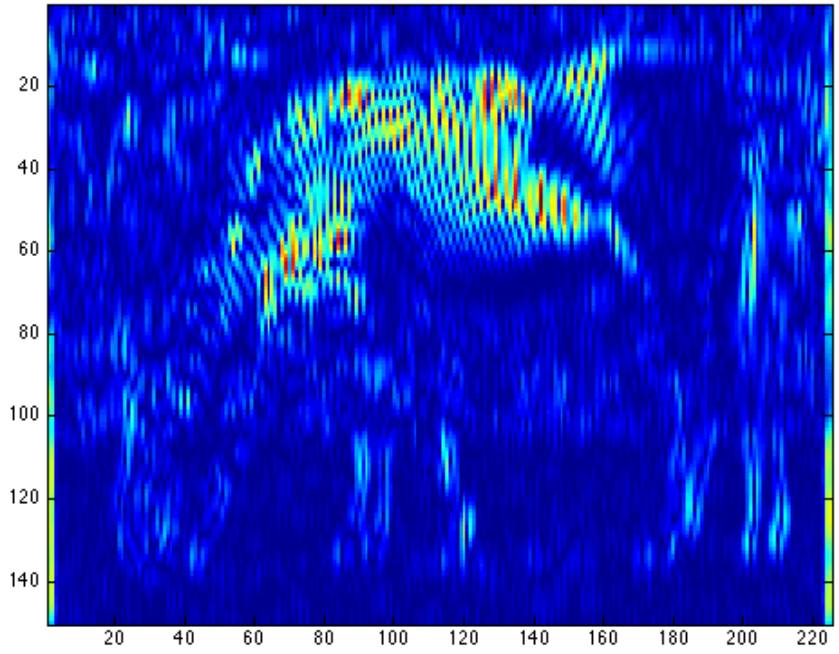


Slide adaptat după K. Grauman

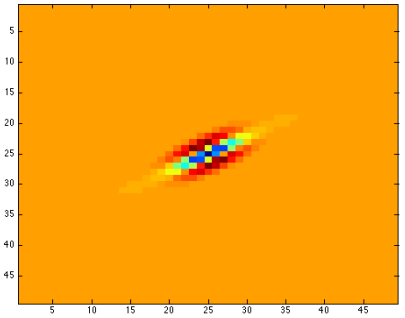
Filtered images with absolute response





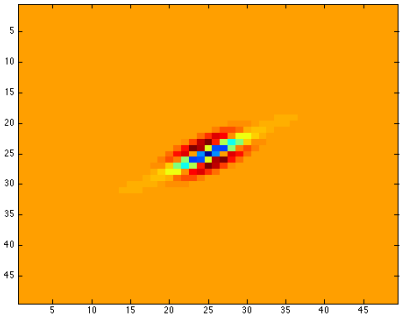


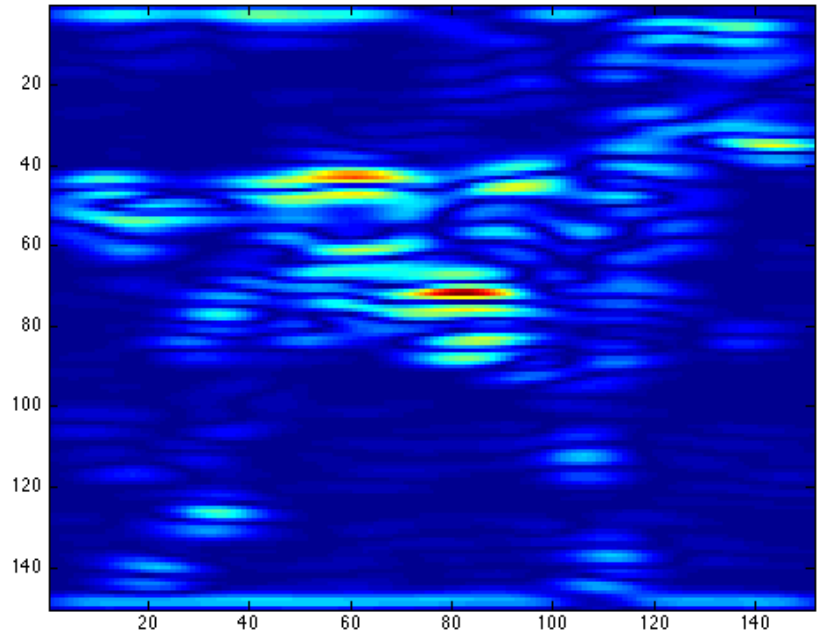
Initial images + filter

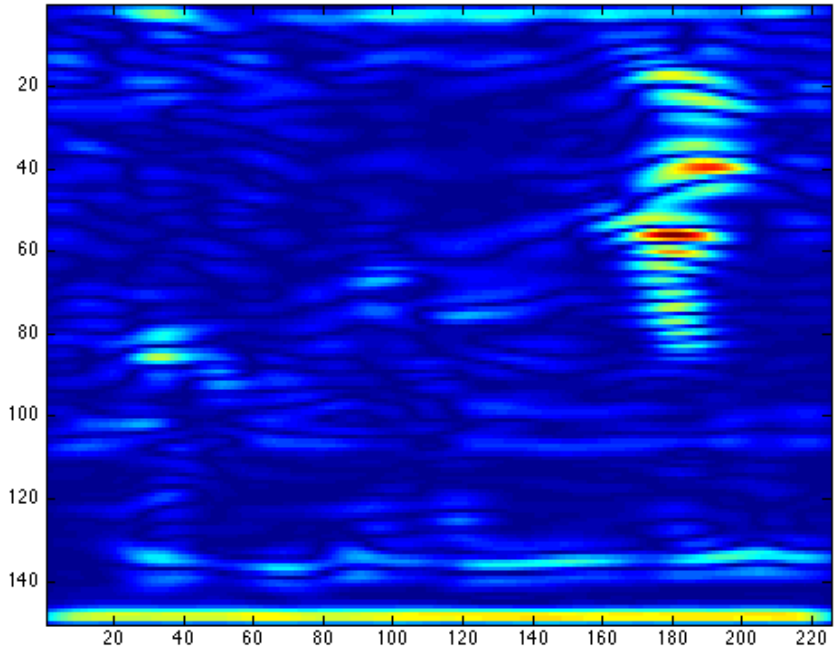


Slide adaptat după K. Grauman

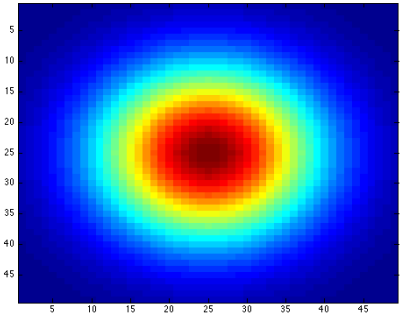
Filtered images with absolute response





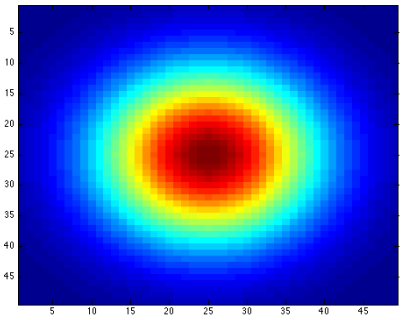


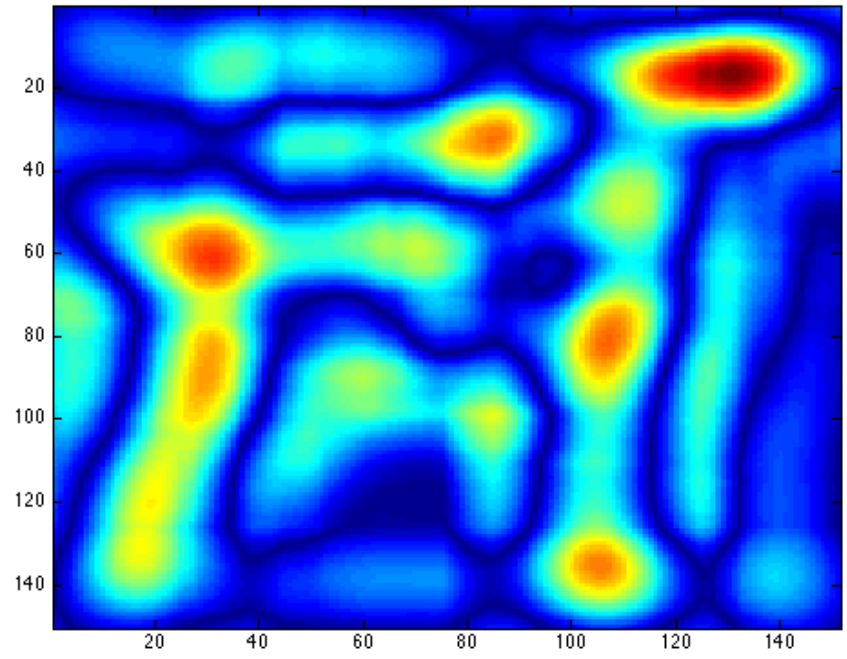
Initial images + filter

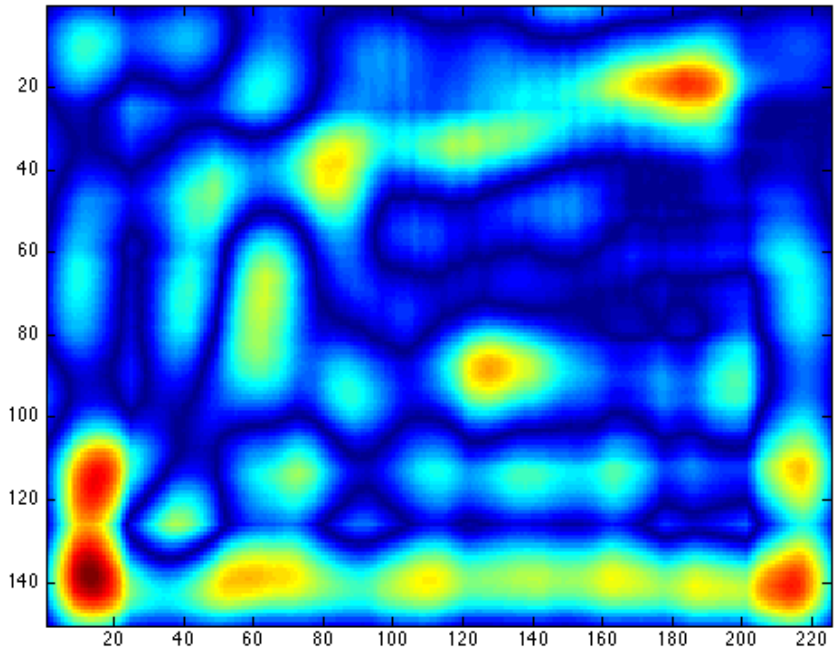


Slide adaptat după K. Grauman

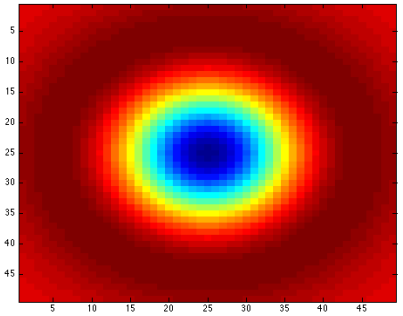
Filtered images with absolute response





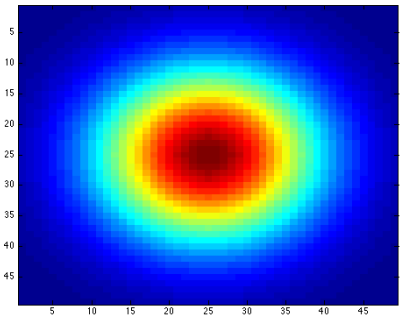


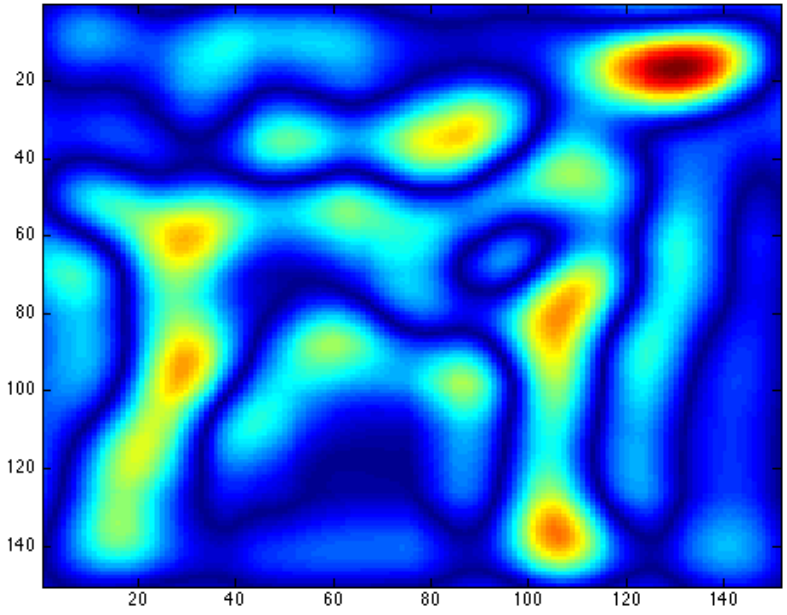
Initial images + filter

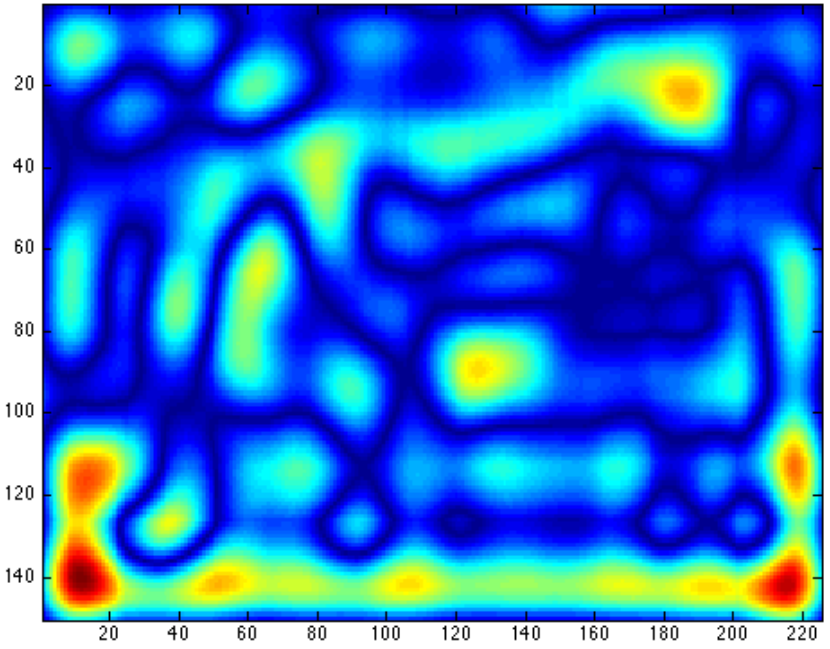


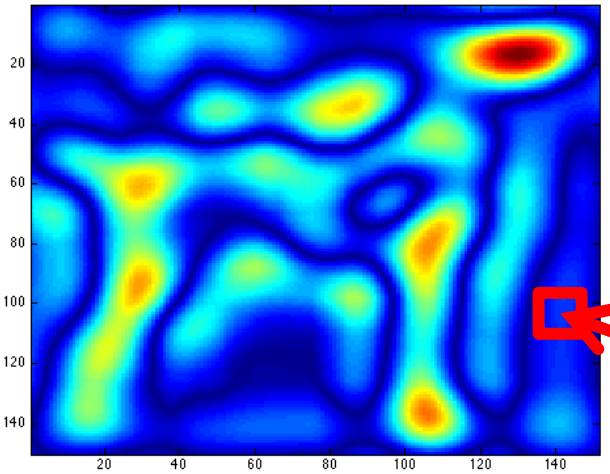
Slide adaptat după K. Grauman

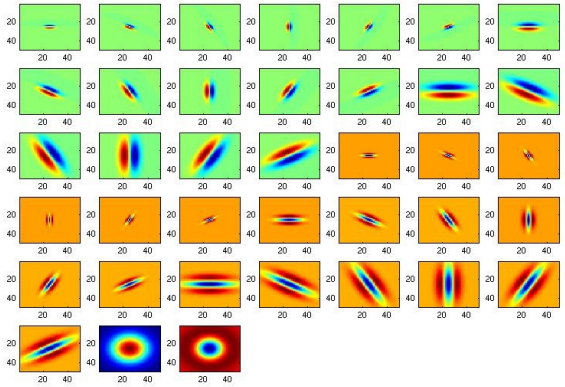
Filtered images with absolute response



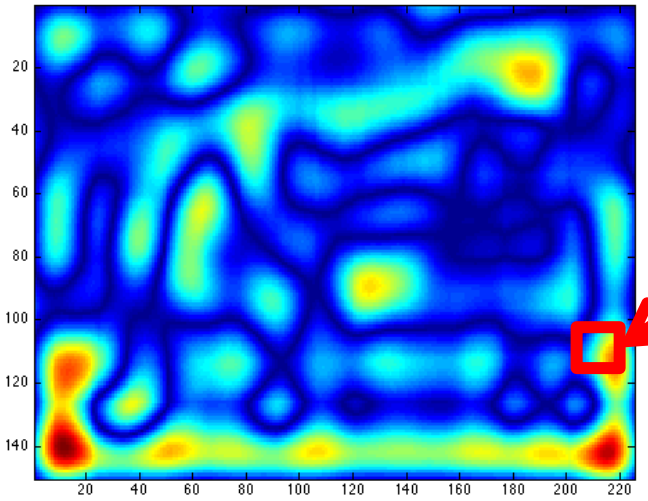








[r1, r2, …, r38]

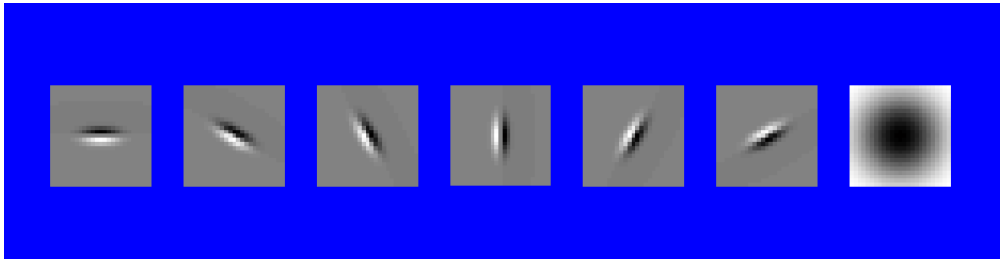


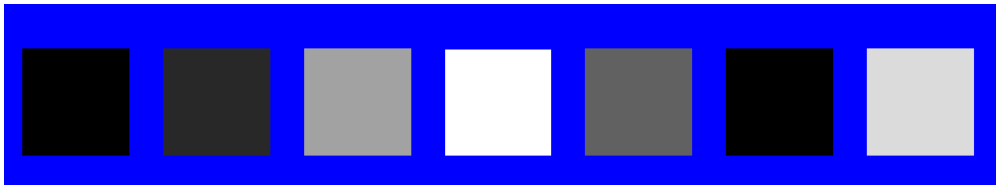
We can form a feature vector from the list of responses at each pixel.

Slide credit: Kristen Grauman

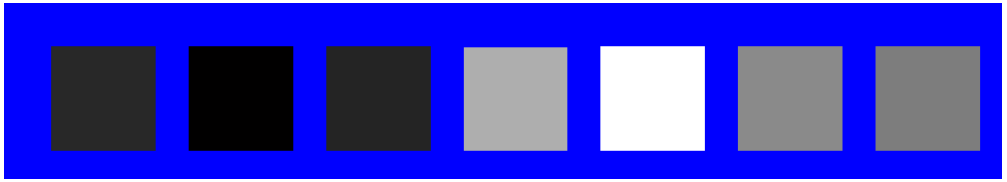
You try: Can you match the texture to the response?

Filters

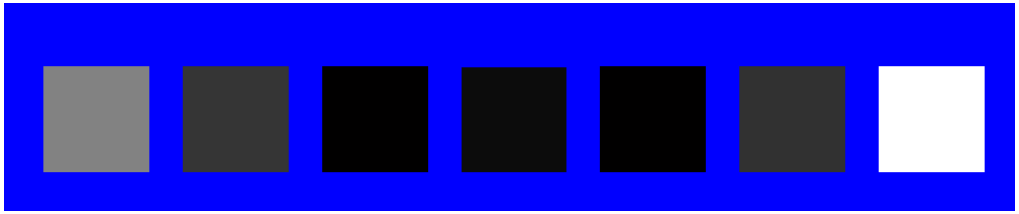




1



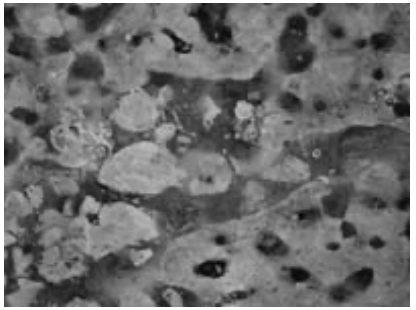
2



3

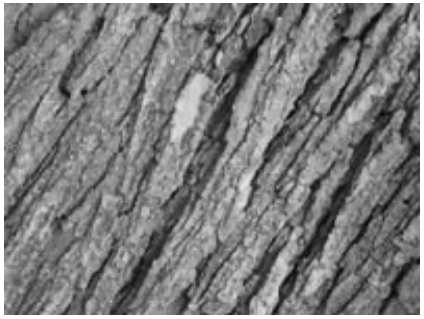
Mean abs responses

A



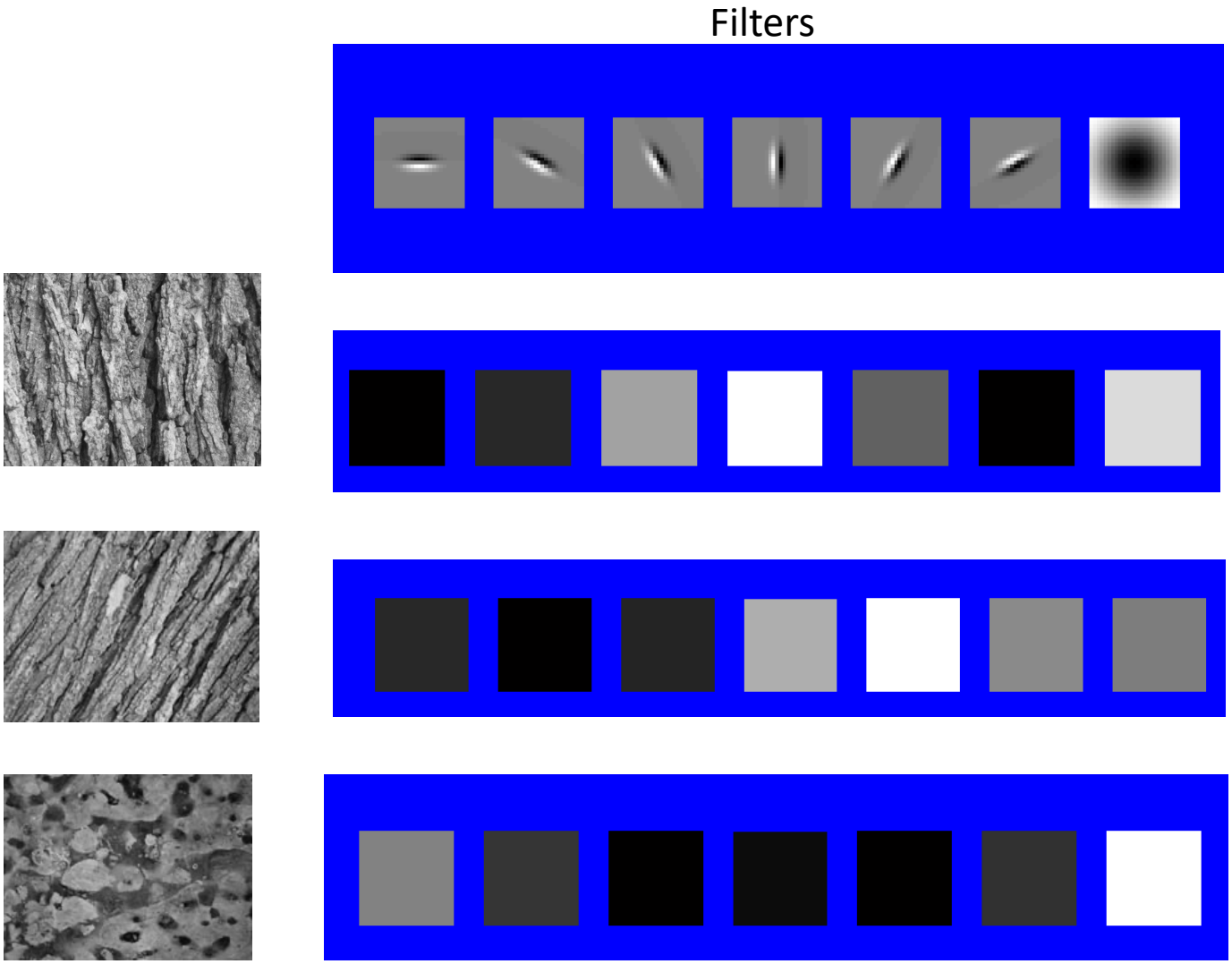
B

C



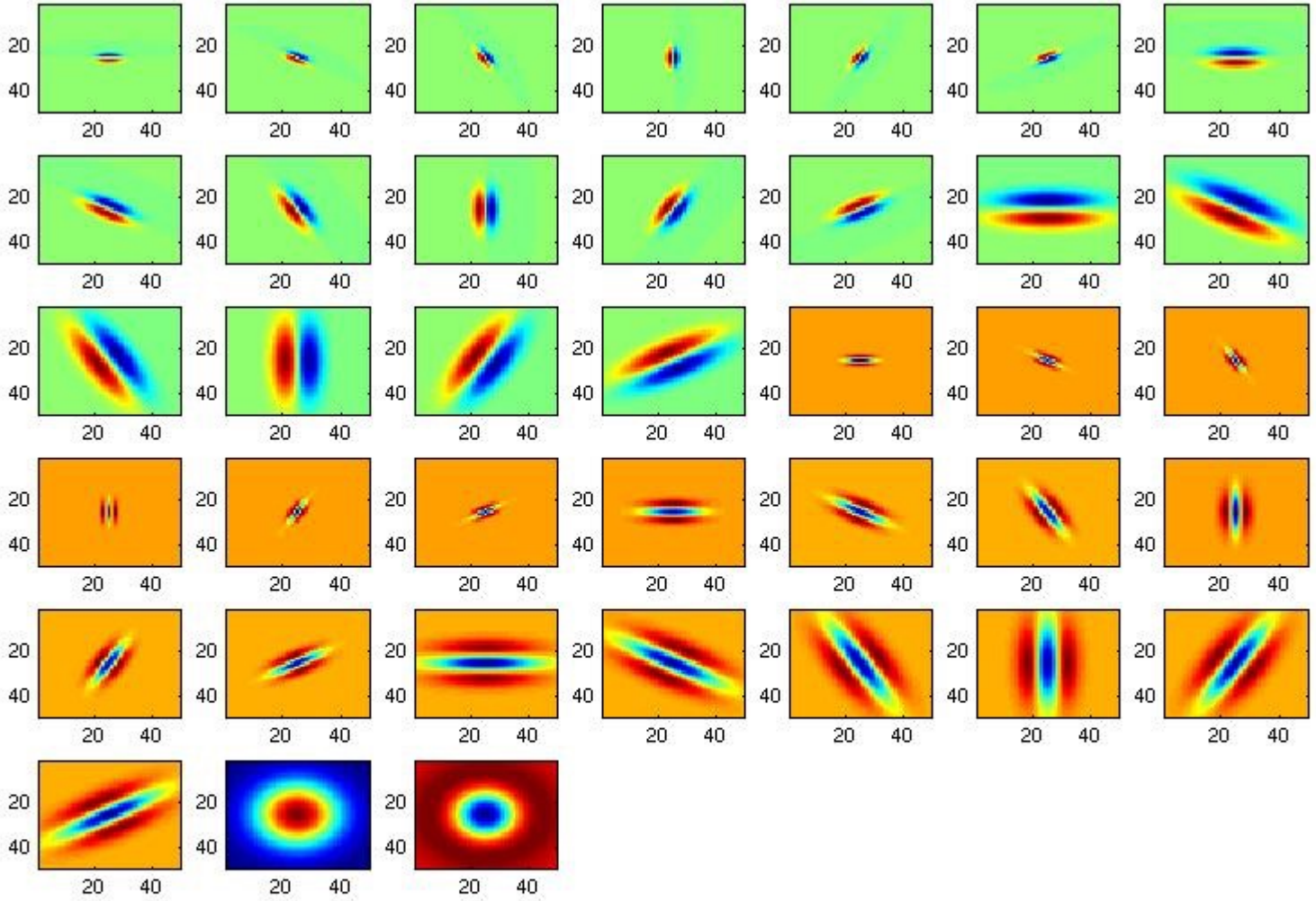
Slide credit Derek Hoiem

Representing texture by mean abs response



Slide credit Derek Hoiem

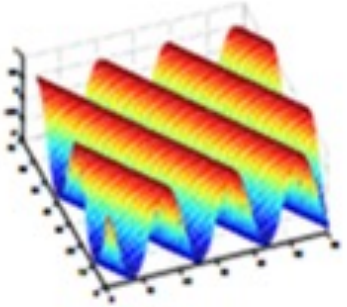
Filter bank



Slide credit: Kristen Grauman

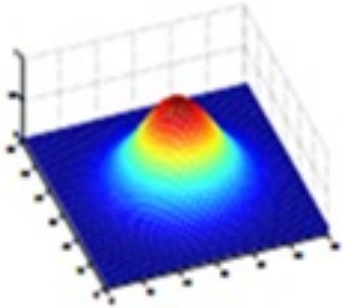
Gabor Filters

A Gabor filter is a combination of a Gaussian filter and a sinusoidal term.



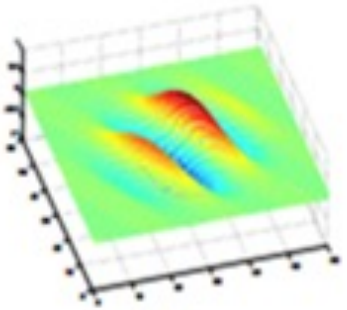
(a)

A Sinusoid oriented 30^0 with X-axis



(b)

A 2-D Gaussian

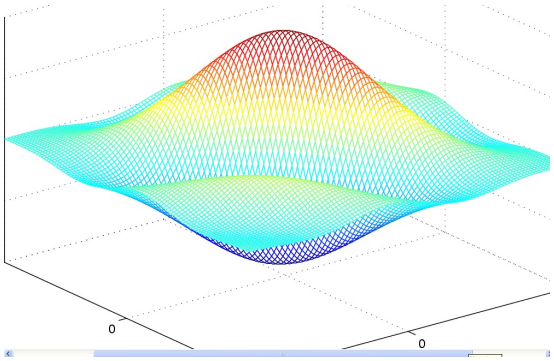


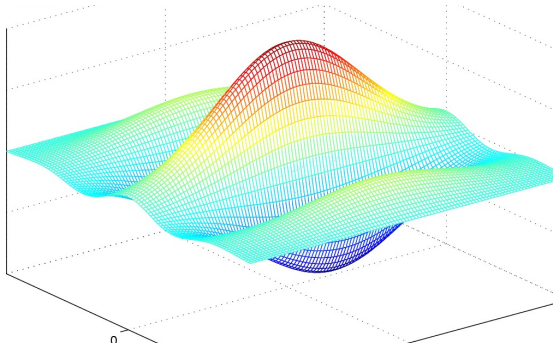
(c)

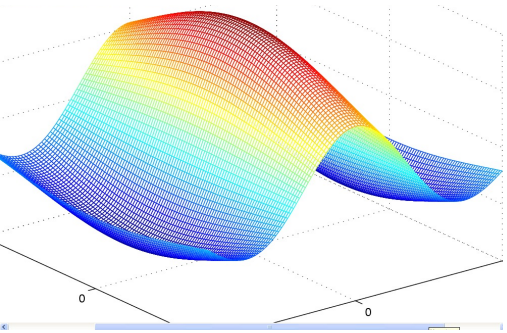
The corresponding 2-D Gabor filter

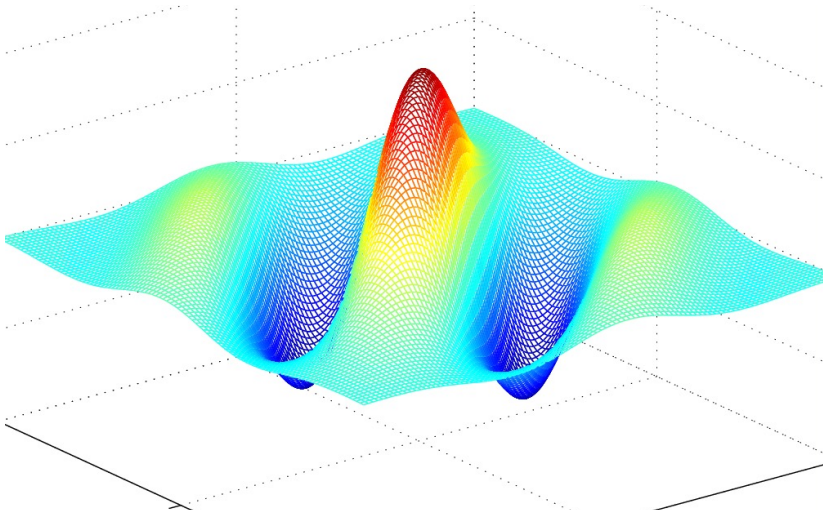
Gabor Filters

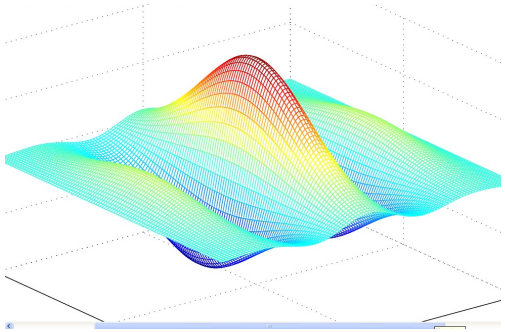
- Wavelets at different frequencies and different orientations





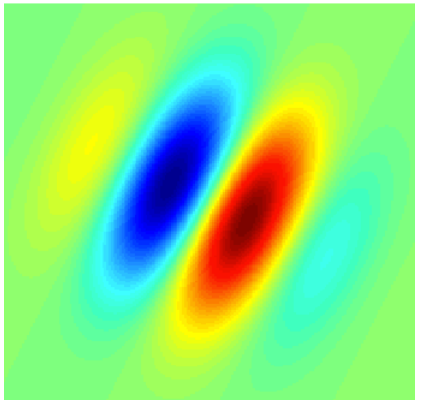






Gabor Filters

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$



where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$.

The parameters λ, θ, ψ are parameters for the sinusoidal part (or factor).

λ and ψ are basic parameters of a sine wave.

λ - controls the Wavelength (https://en.wikipedia.org/wiki/Wavelength) of this sinusoid. A higher λ would "widen" the ellipses you see in the image above, while a lower λ would have the opposite.

θ - controls the rotation of the ellipses you see in the image above. These ellipses are the "parallel stripes". A value of θ = 0 indicates no rotation and would make the ellipses vertical.

ψ - is the phase shift of the sinusoid i.e., how much the ellipses need to be shifted with respect to the center. It's a parameter that has to be specified.

σ - is the standard deviation (i.e., the spread) of the Gaussian part.

γ - controls the aspect ratio of the ellipses (roughly, how elongated the ellipses need to be).

Gabor filter values visualization by Juergen Mueller.

variable naming conventions as used by http://en.wikipedia.org/wiki/Gabor_filter.
Red: positive values. Blue: negative values

1. Wavelength is changed.
1. wavelength (lambda) : 5.00
2. orientation (theta) : 0.79
3. gaussvar   (sigma) : 20.00
4. phaseoffset (phi)   : 0.00
5. aspectratio (gamma) : 0.50

min gabor val : -0.91
max gabor val : 1.00

Color

Important cue for recognition

Important cue for segmentation

"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother* Vladimir Kolmogorov† Andrew Blake‡
Microsoft Research Cambridge, UK

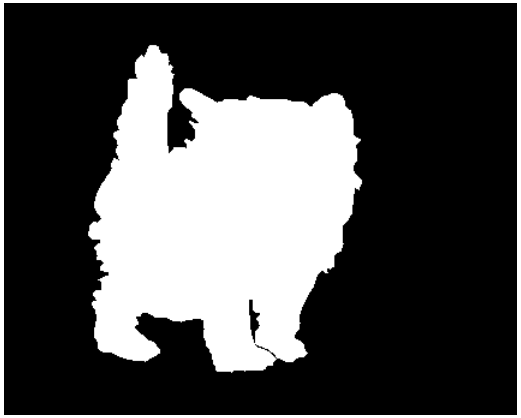
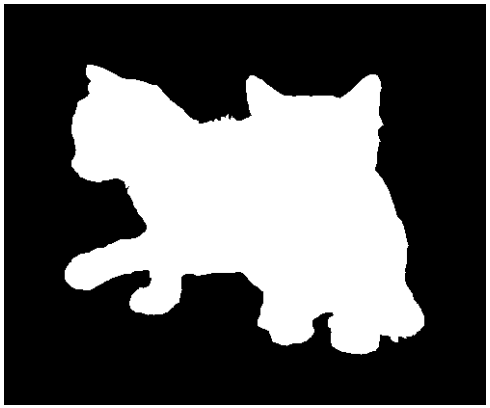
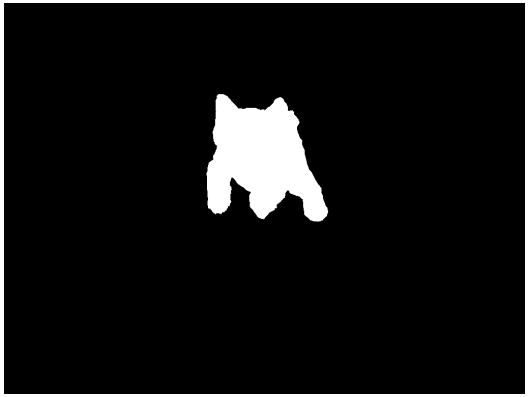
Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

Exercise lab 3
Grabcat - Foreground/background segmentation of cats in images

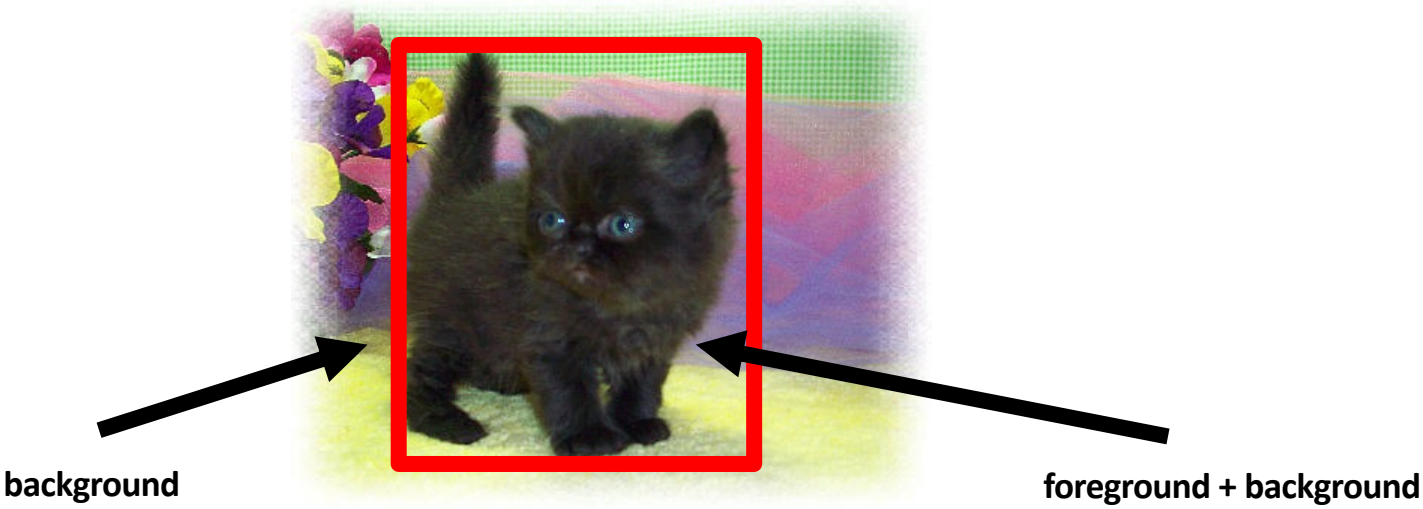
Input images



Ground-truth binary masks

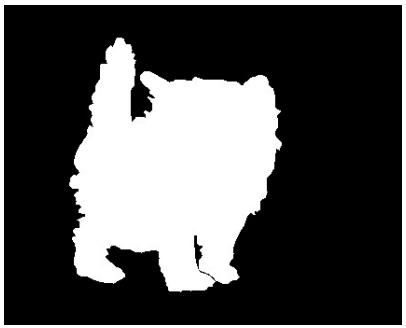




Exercise lab 3
Grabcat - Foreground/background segmentation of cats in images

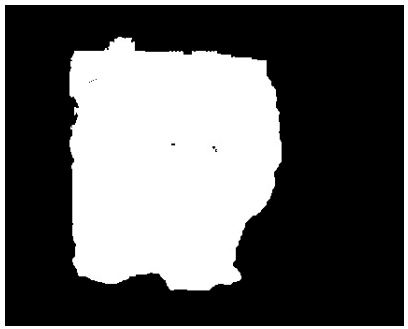


Exercise lab 3
Grabcat - Foreground/background segmentation of cats in images

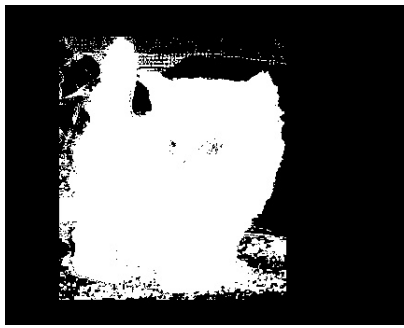
Input image



Ground-truth binary mask



Grabcut

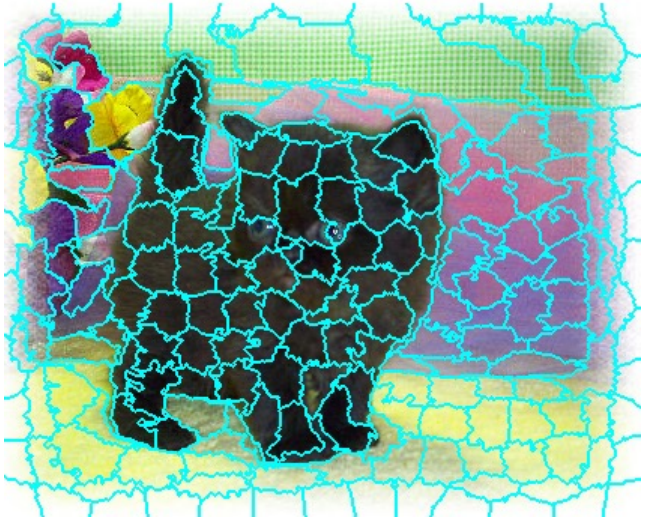


Grabcat

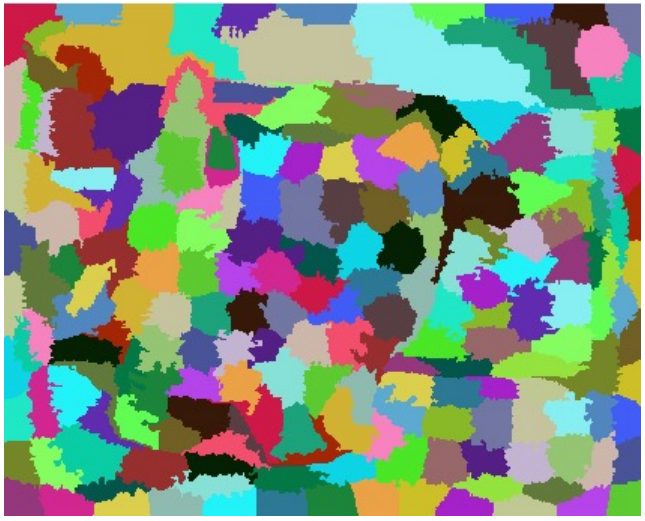
Exercise lab 3 - Grabcat

Input image

Superpixels



Superpixels color labeling

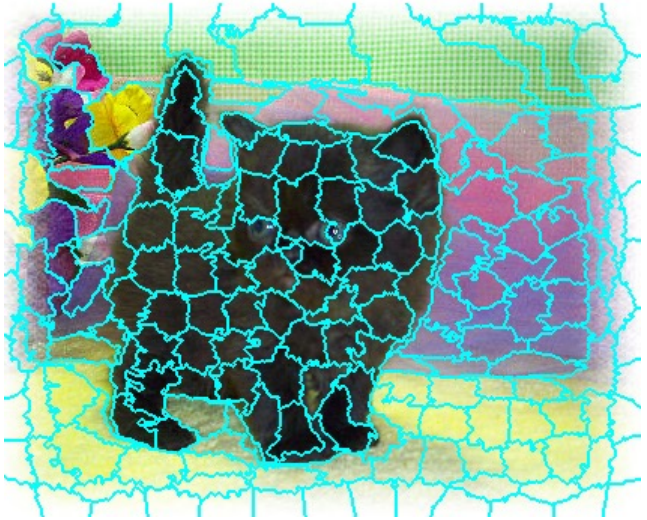


Best achievable mask using superpixels

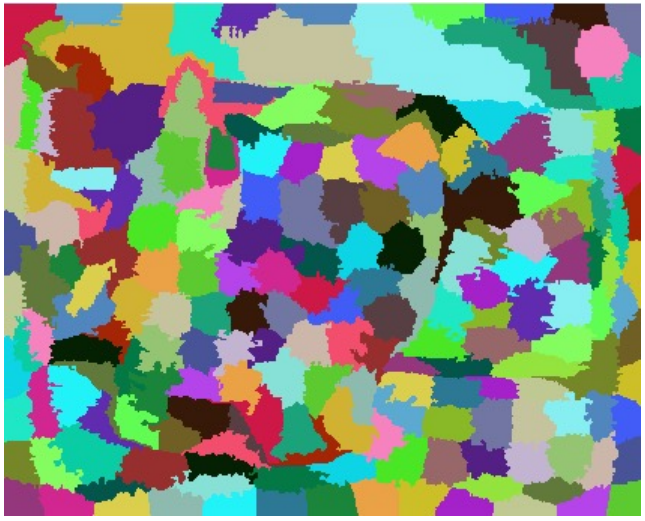
Exercise lab 3 - Grabcat

Input image

Superpixels



Superpixels color labeling



Result obtained using superpixels

SLIC Superpixels Compared to State-of-the-art Superpixel Methods

Radhakrishna Achanta, Appu Shaji, Kevin Smith,
Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk

***Abstract*—Computer vision applications have come to rely increasingly on superpixels in recent years, but it is not always clear what constitutes a good superpixel algorithm. In an effort to understand the benefits and drawbacks of existing methods, we empirically compare five state-of-the-art superpixel algorithms for their ability to adhere to image boundaries, speed, memory efficiency, and their impact on segmentation performance. We then introduce a new superpixel algorithm, *simple linear iterative clustering* (SLIC), which adapts a k-means clustering approach to efficiently generate superpixels. Despite its simplicity, SLIC adheres to boundaries as well as or better than previous methods. At the same time, it is faster and more memory efficient, improves segmentation performance, and is straightforward to extend to supervoxel generation.**

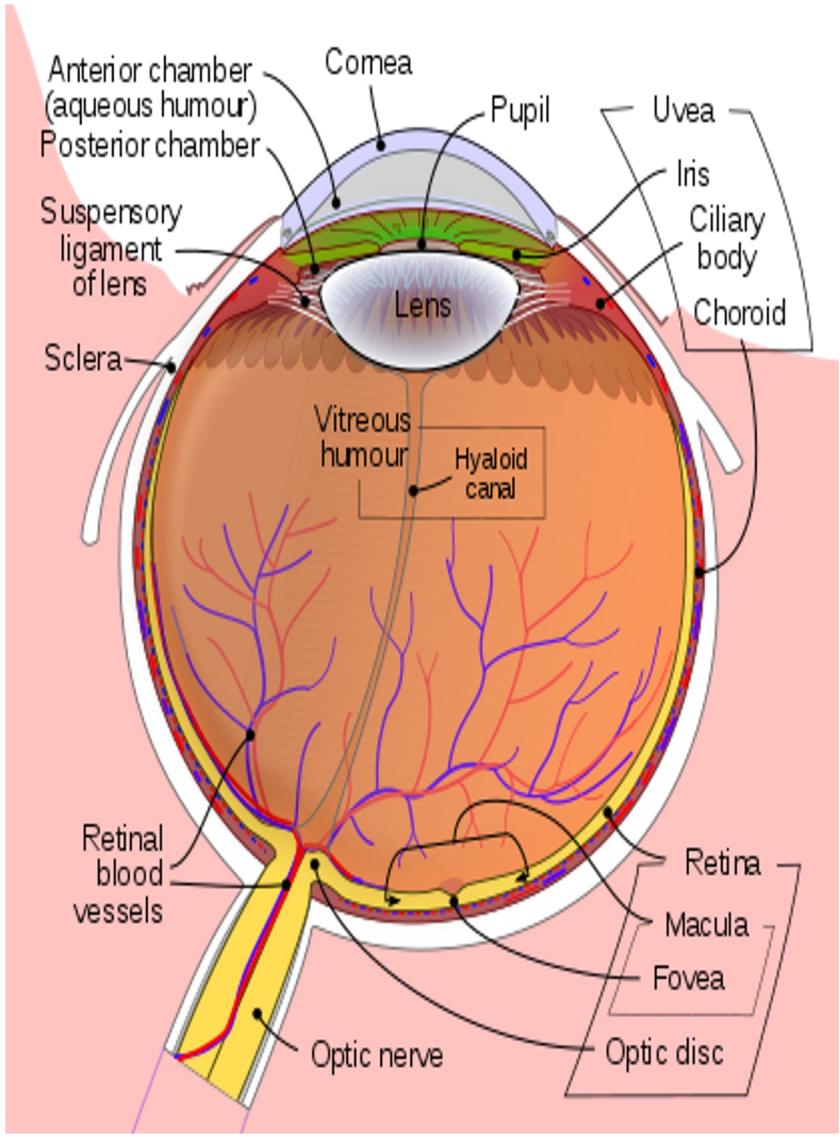
Fig. 1: Images segmented using SLIC into superpixels of size 64, 256, and 1024 pixels (approximately).

regular lattice, such as [23], is probably a better choice. While it is difficult to define what constitutes an ideal approach for all applications, we believe the following properties are generally desirable:

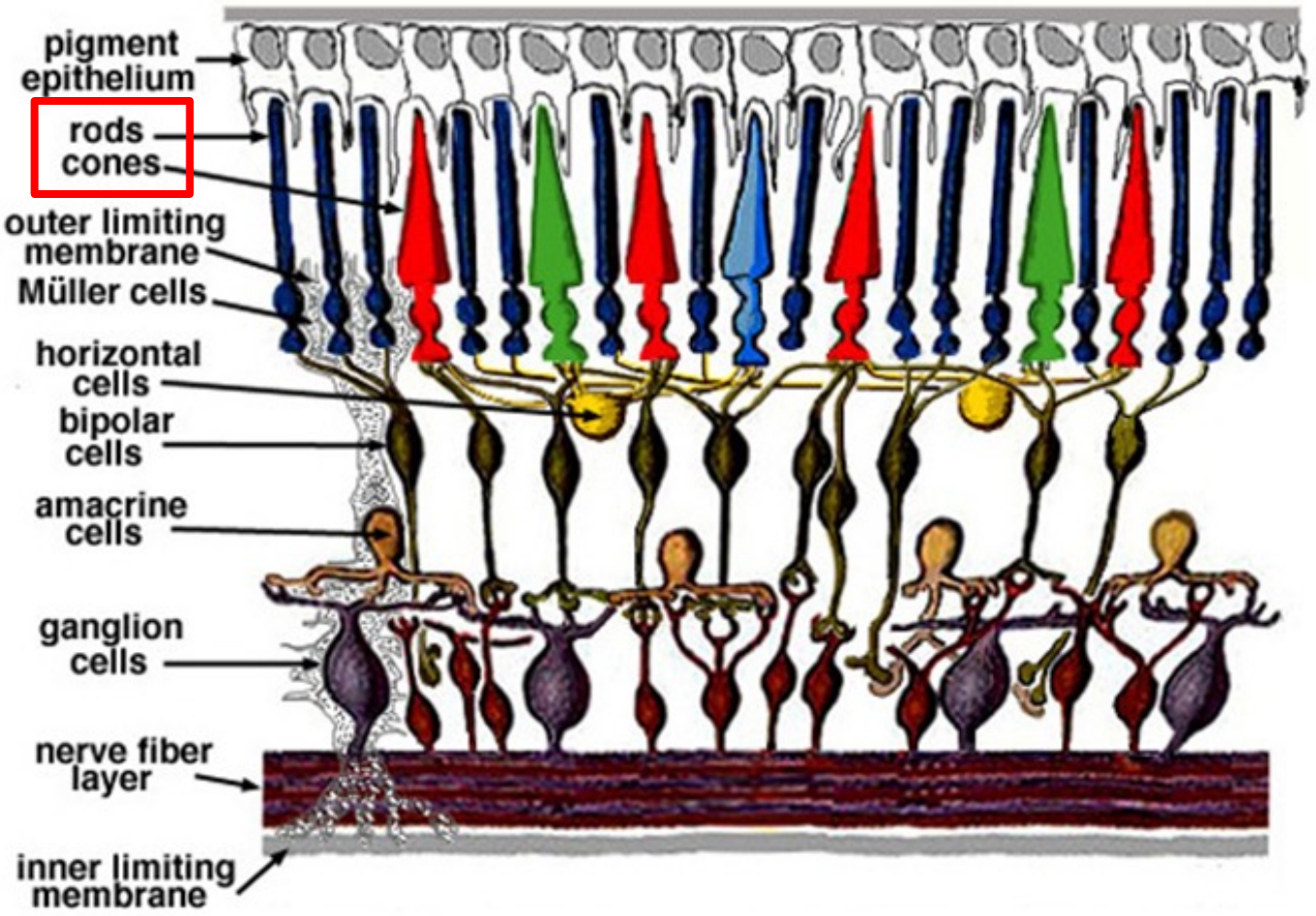
1) Superpixels should adhere well to image boundaries.
2) When used to reduce computational complexity as a pre-processing step, superpixels should be fast to compute, memory efficient, and simple to use.
3) When used for segmentation purposes, superpixels should both increase the speed and improve the quality of the results.

So how do human eyes work?

- Complex!
- Light passes through
  - Cornea, humours, lens refract light to focus
- Hit the retina
- Absorbed by photosensitive cells
- Info transmitted through optic nerve, processed by visual cortex

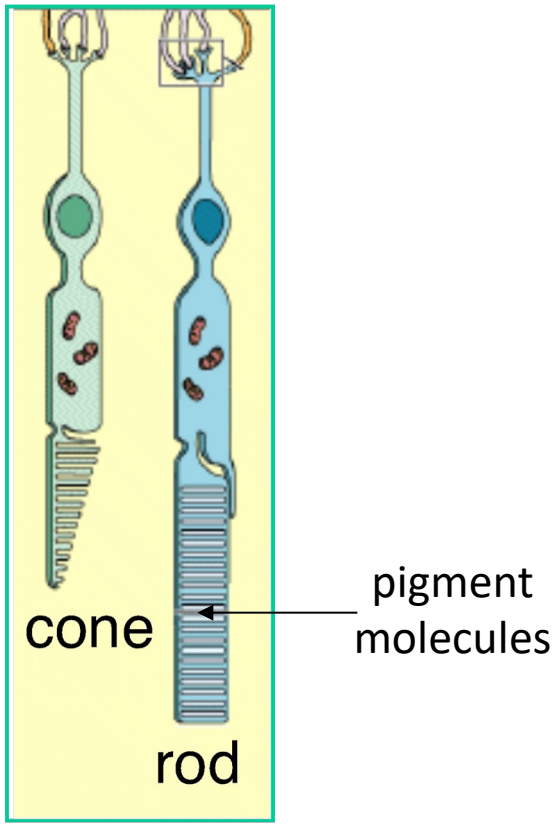


Retina



Simple diagram of the organization of the retina.

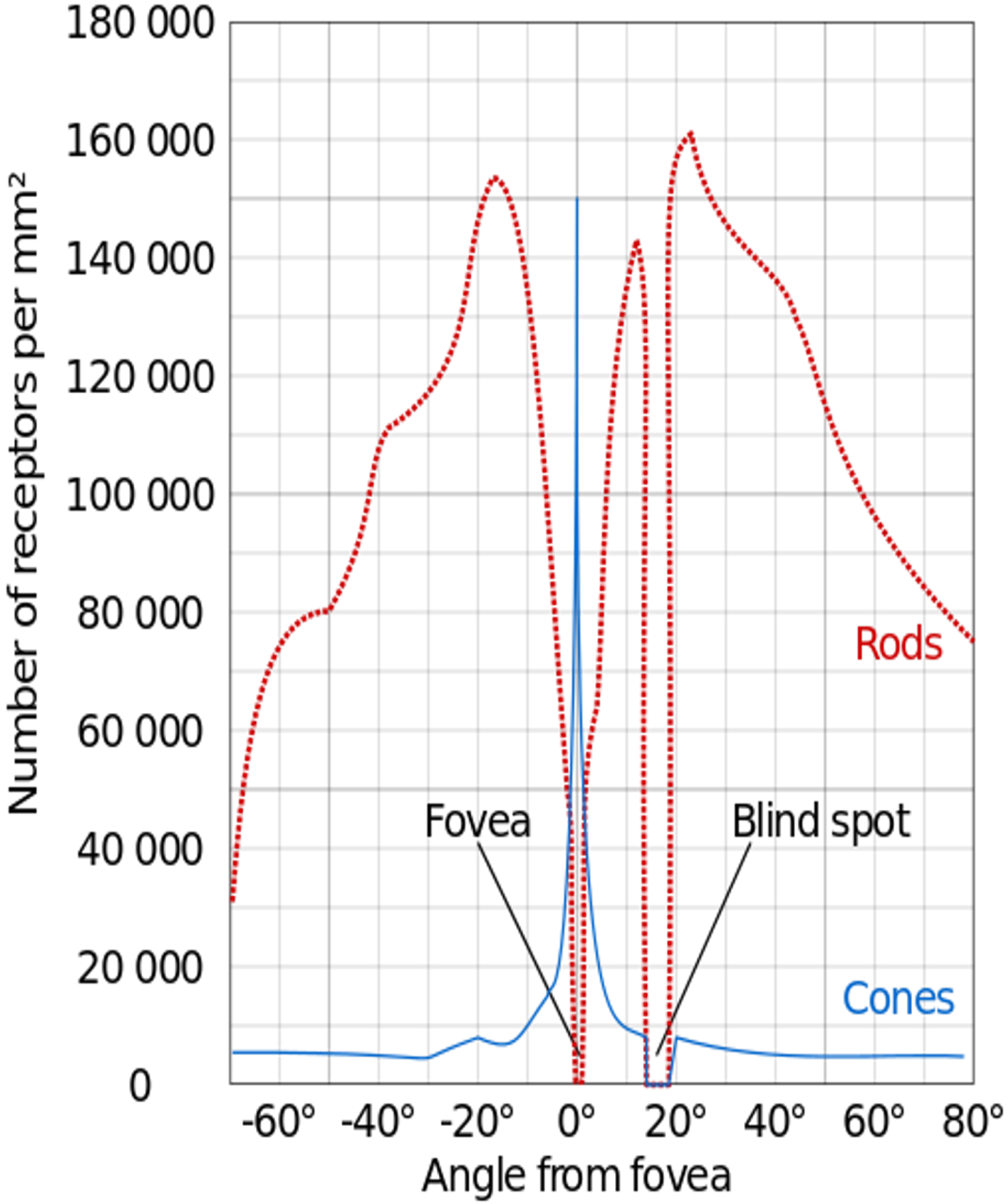
Photoreceptive cells - rods and cones

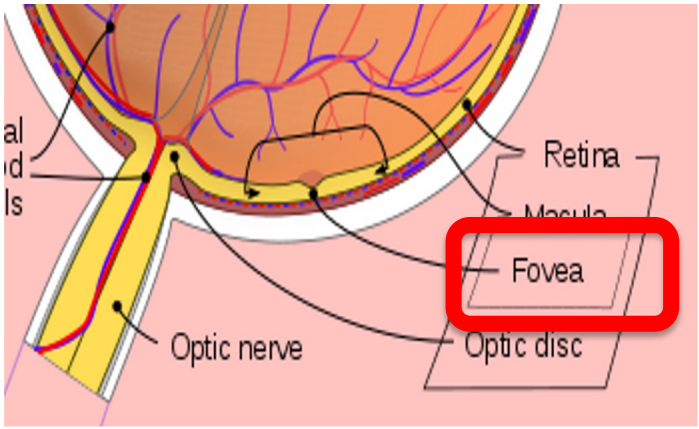


- Cones (RO: conuri)
  - around 6 million
  - need many photons to activate, bright light
  - fast response time
  - fine details
  - fast changes over time
  - responsible for most daytime vision (color vision)
  - mostly packed into one region: Fovea

- Rods (RO: bastonașe)
  - ~120 million
  - sensitive to 1 photon
  - can pool responses
  - slow response time
  - only operate in low-light conditions
  - saturate quickly in lots of light
  - take ~7 minutes to adjust (night vision)

Fovea: where it's all happening





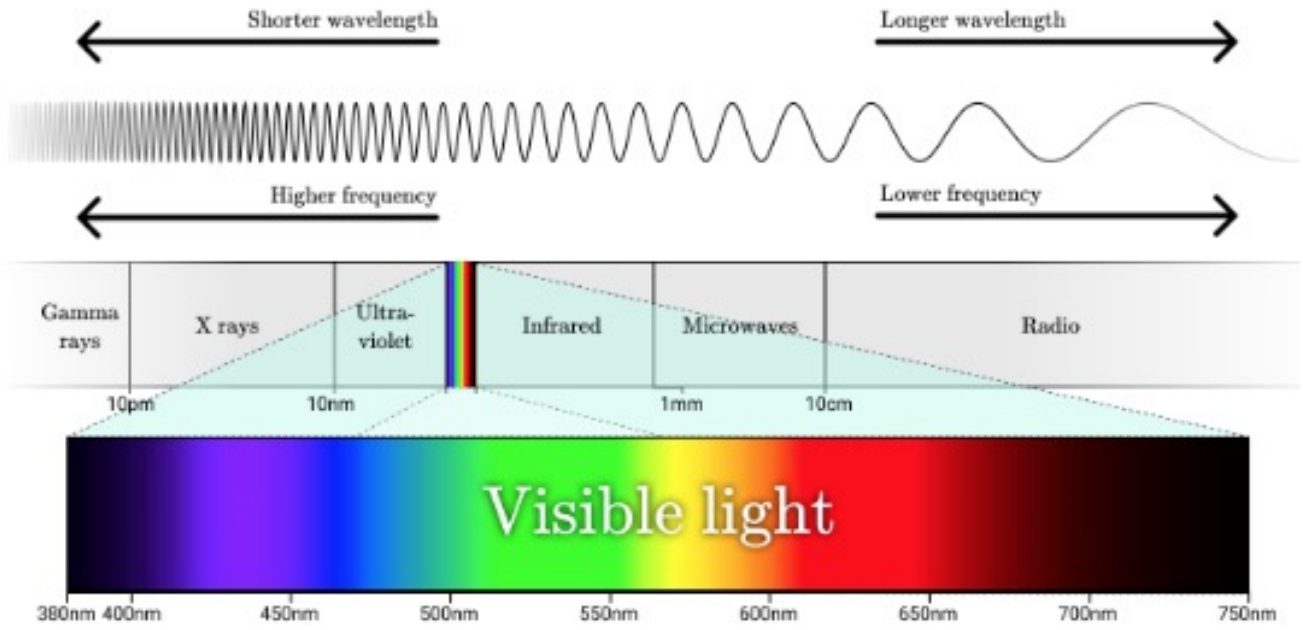
Fovea

- Small circle on the retina, 1.5mm^2
- Densely packed with cones
  - 200,000 cones/mm2
- Highest visual acuity
- Reading: move your eyes so the text is centered in fovea
- Responsible for sharp central vision;

What is color?

Color = the result of interaction between physical light in the environment and our visual system

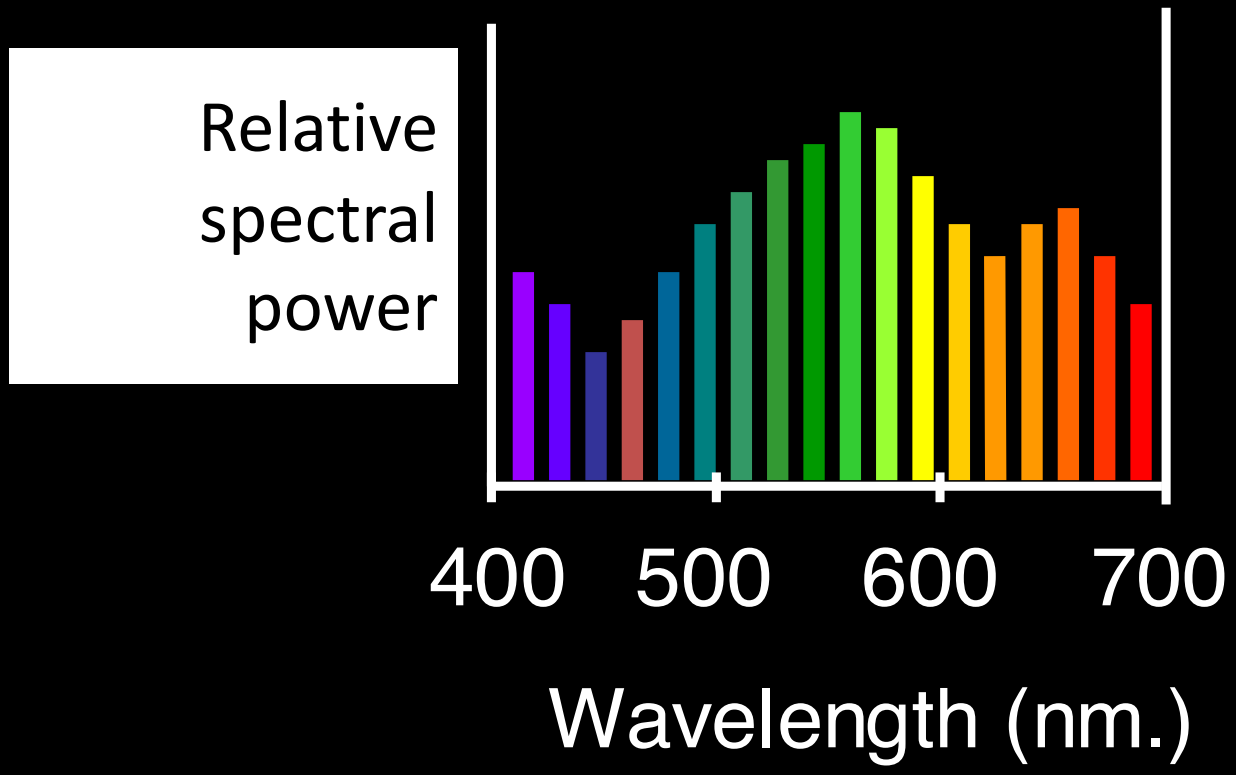
Color = electromagnetic wave detected by the eye and interpreted by the brain. The visible light is composed of a relatively narrow band of the electromagnetic spectrum. The visible spectrum is in the wavelength range: 380nm - 780nm.



The electromagnetic spectrum

The Physics of Light

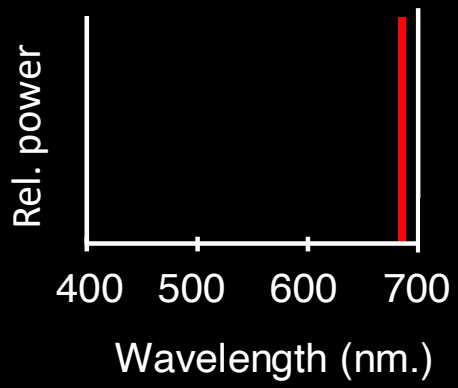
Any source of light can be completely described physically by its spectrum: the amount of energy emitted (per time unit) at each wavelength 400 - 700 nm.



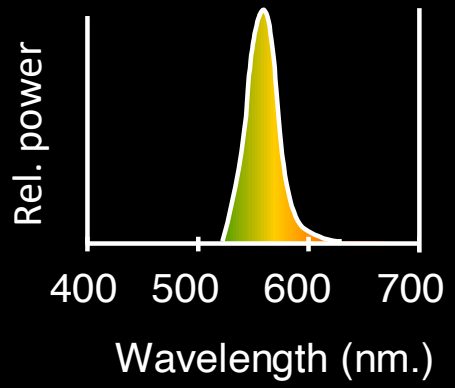
Spectra of Light Sources

Some examples of the spectra of light sources

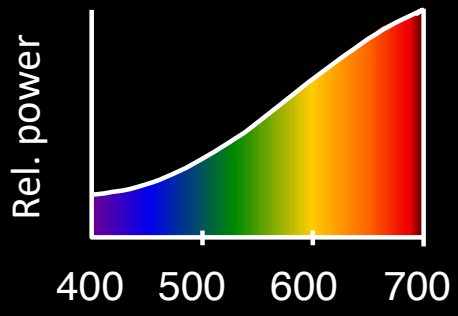
A. Ruby Laser



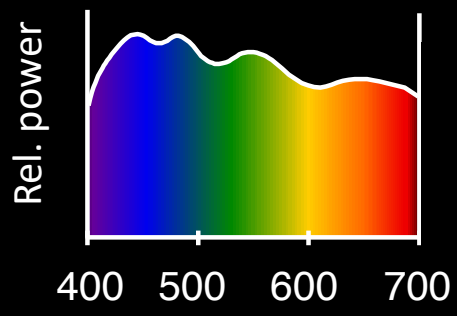
B. Gallium Phosphide Crystal



C. Tungsten Lightbulb



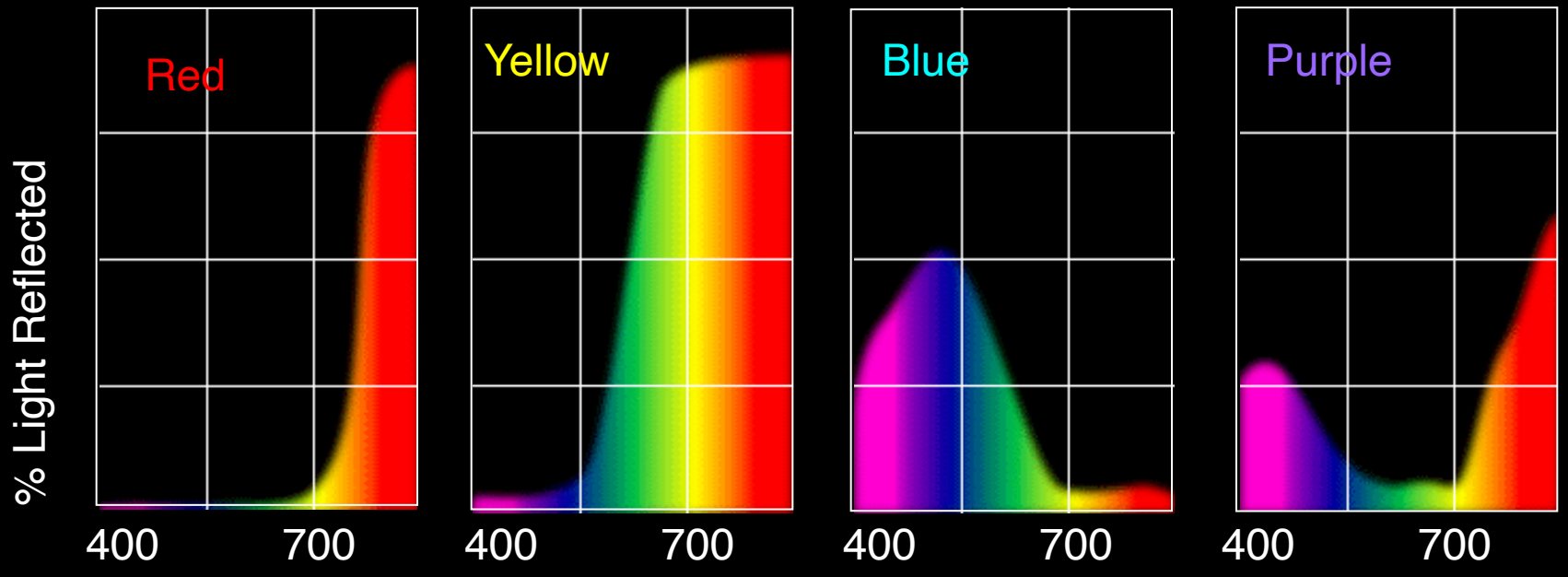
D. Normal Daylight



Reflectance Spectra of Surfaces

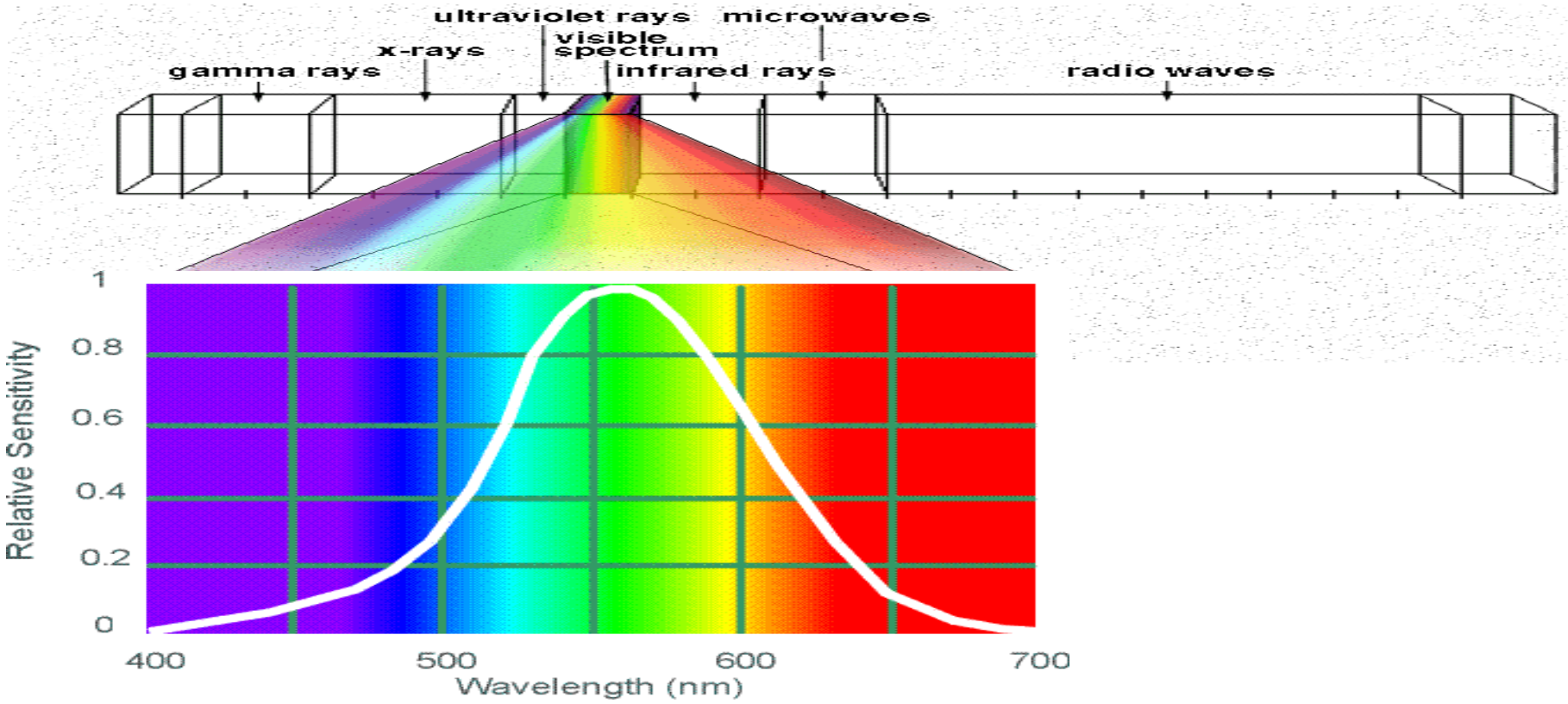
Some examples of the reflectance spectra of surfaces



Human Luminance Sensitivity Function

Light is composed of a spectrum of wavelengths. If we measure a human's eye sensitivity to every wavelength we get the standard human luminance sensitivity function



Human Luminance Sensitivity Function

Slide Credit: A. Efros

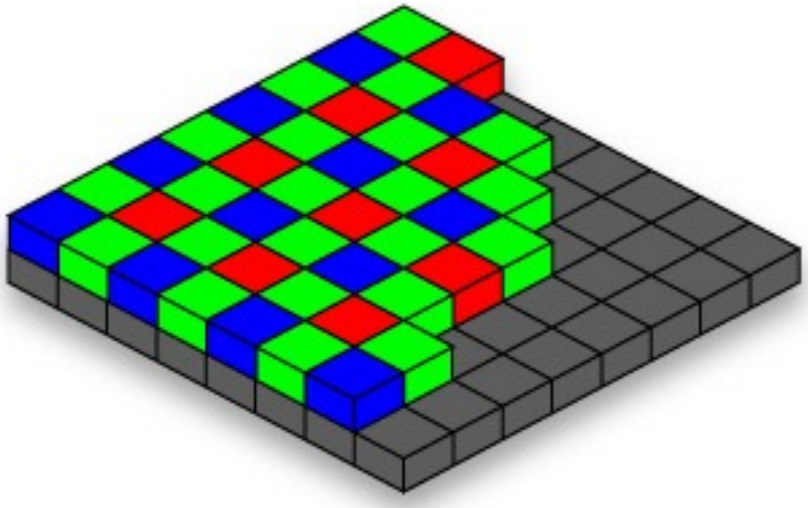
This is hard to read

This is easier to read

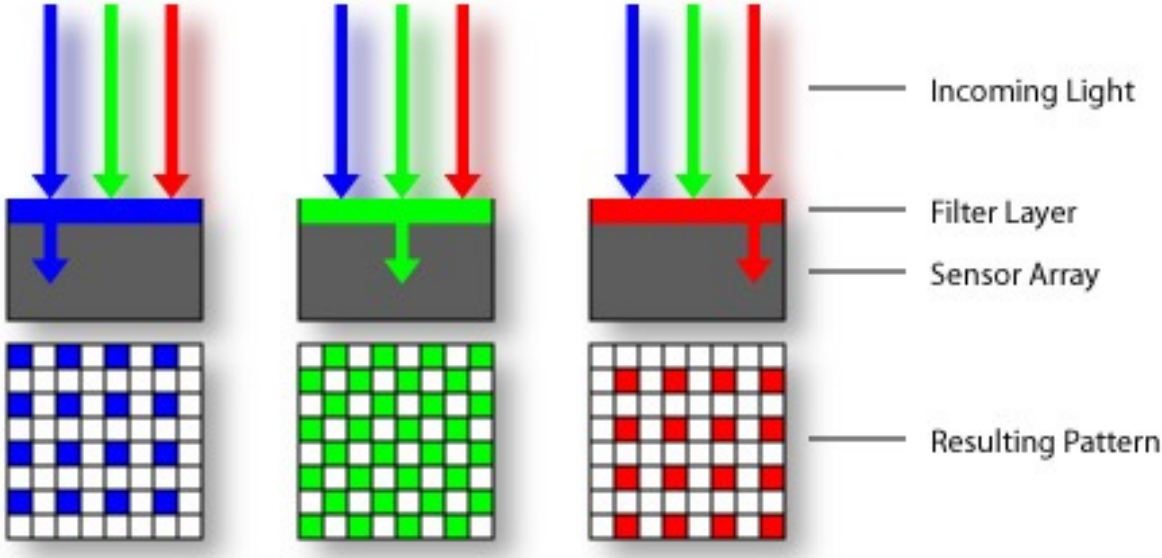
Why RGB?

If light is a spectrum, why are images RGB?

Color Sensing: Bayer Grid



- Estimate RGB at each cell from neighboring values

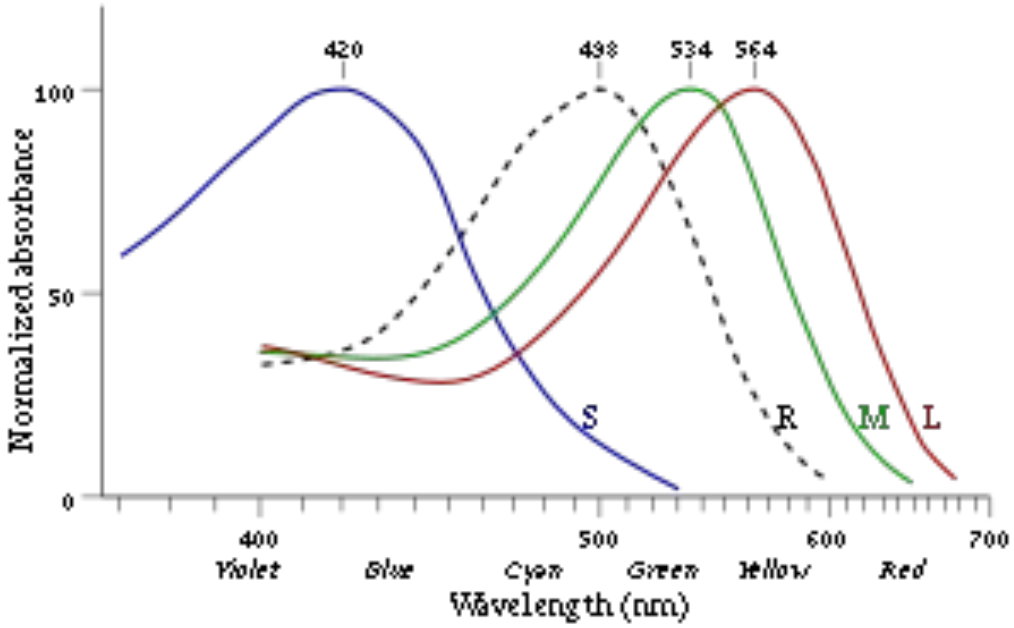


http://en.wikipedia.org/wiki/Bayer_filter

Slide credit: Steve Seitz

Photoreceptors and light

- Each receptor has a responsiveness curve
- Receptors more responsive to some wavelengths, less responsive to others
- Rods: peak around 498 nm
- Cones: 3 kinds
  - Short: peak around 420 nm
  - Medium: peak around 530 nm
  - Long: peak around 560 nm



Cones and color

Our perception of color comes from cones

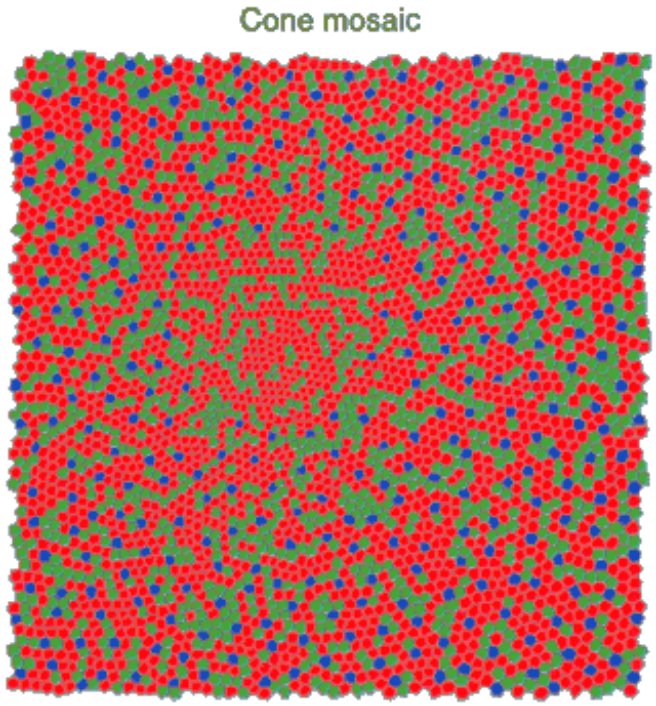
Different waveforms provoke different responses

Each cone has essentially one “output”

To calculate:

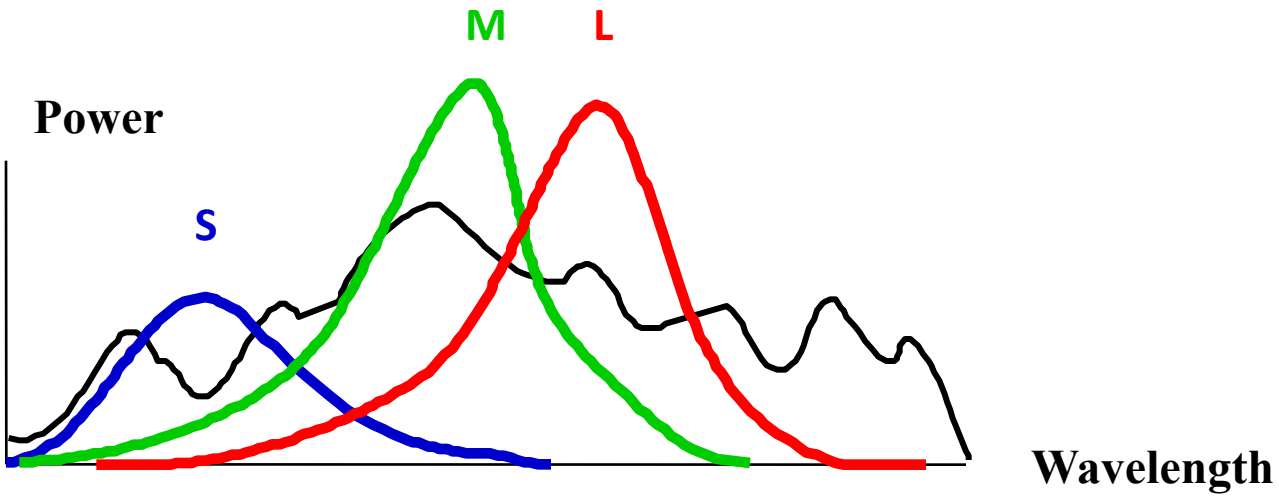
- Multiply input waveform by response curve
- Integrate area under the curve

The “color” we see is the relative activation of the 3 kinds of cones



- Ratio of Long to Medium to Short cones: approx. 10:5:1
- Almost no S cones in the center of the fovea

Color perception



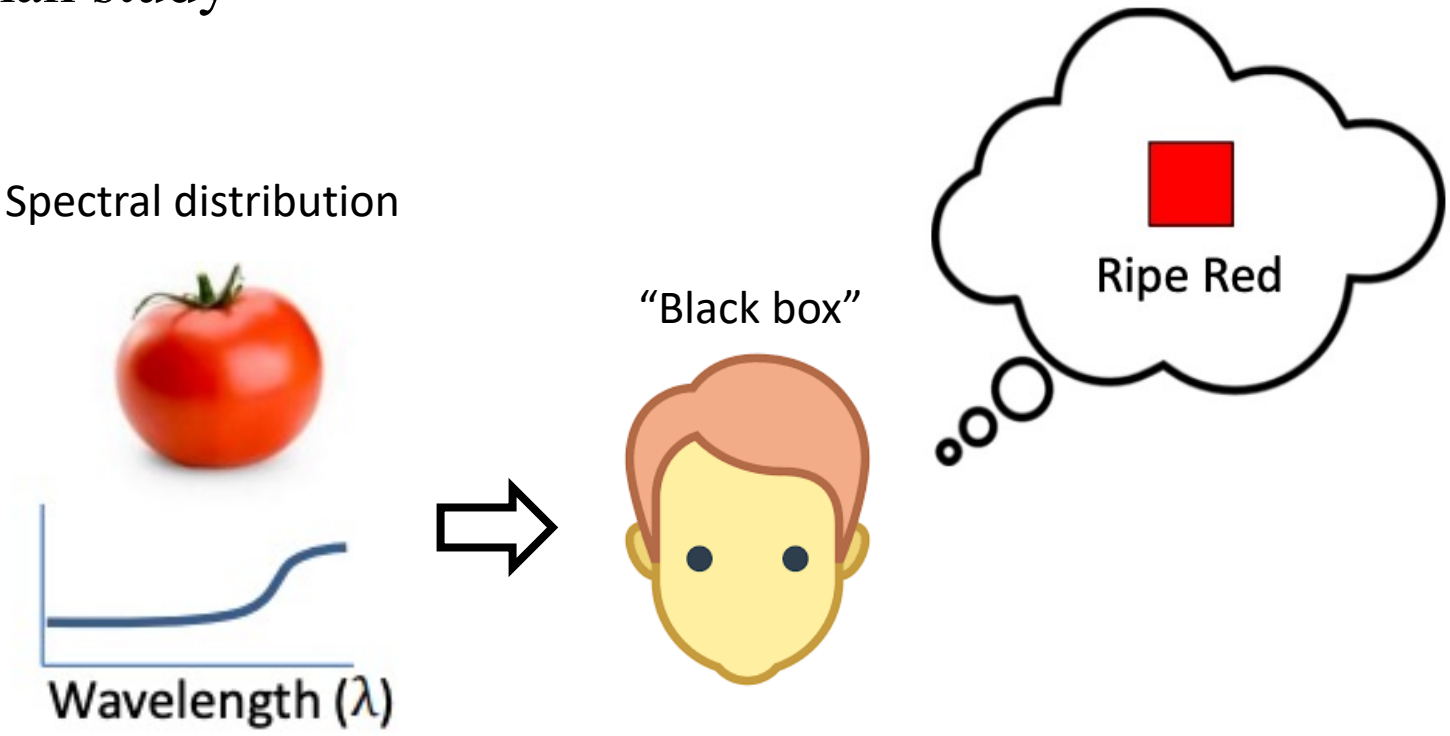
Rods and cones act as *filters* on the spectrum

- To get the output of a filter, multiply its response curve by the spectrum, integrate over all wavelengths
  - Each cone yields one number
- How can we represent an entire spectrum with three numbers? We can't! Most of the information is lost
  - As a result, two different spectra may appear indistinguishable (such spectra are known as **metamers**)

Slide credit: Steve Seitz

Quantifying color

- Spectral distributions go through a “black box” (human visual system) and are perceived as color
- The only way to quantify the “black box” is to perform a human study



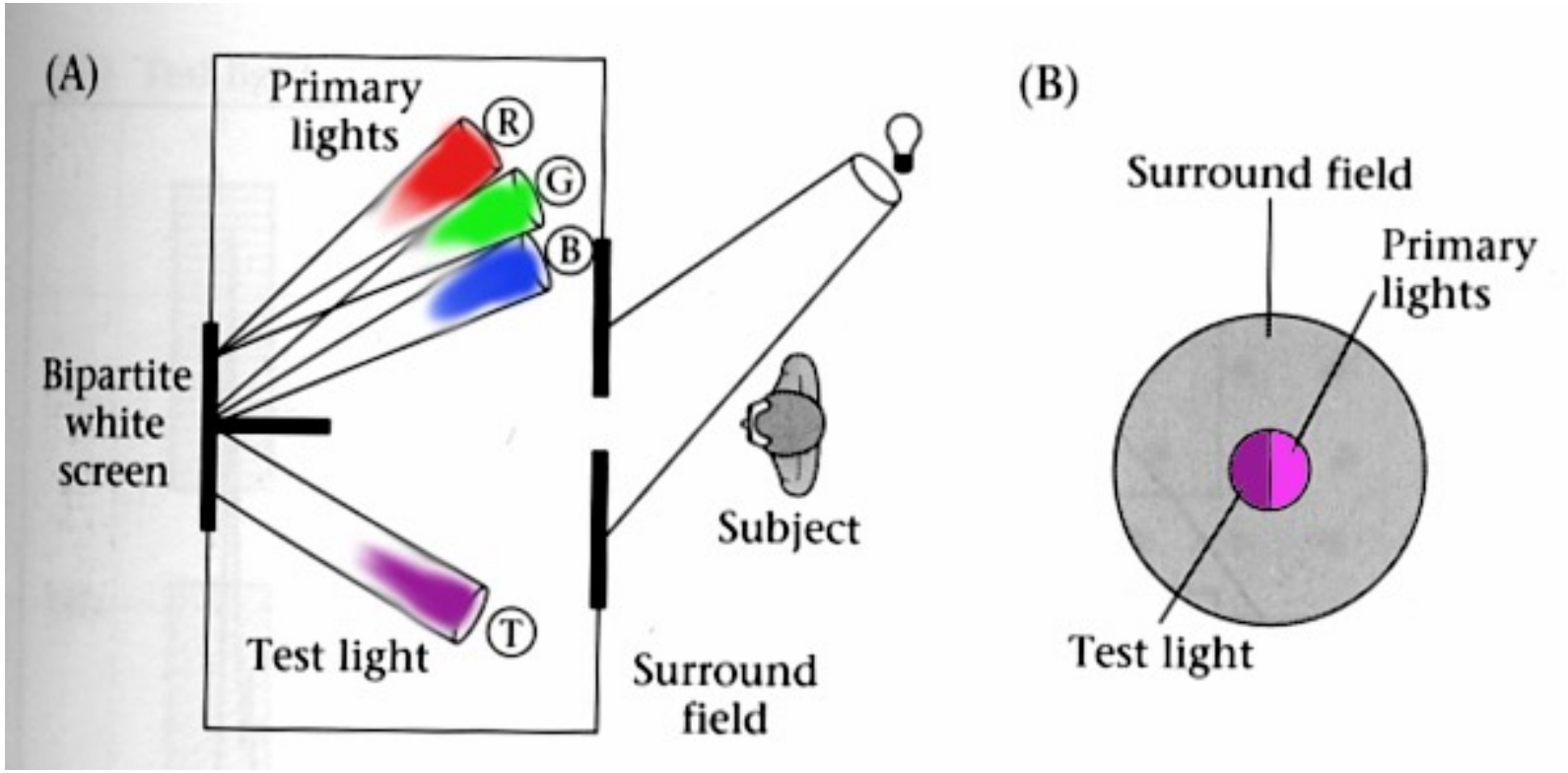
Color matching experiments

- We would like to understand which spectra produce the same color sensation in people under similar viewing conditions.

- Late 1920s experimented with colors! (and people)
  - Subjects get controls to 3 “primary” lights
  - Show them a light
  - Subject adjusts their lights to match the given light

Color matching experiments

- Late 1920s experimented with colors! (and people)
  - Subjects get controls to 3 “primary” lights
  - Show them a light
  - Subject adjusts their lights to match the given light



Trichromatic color theory

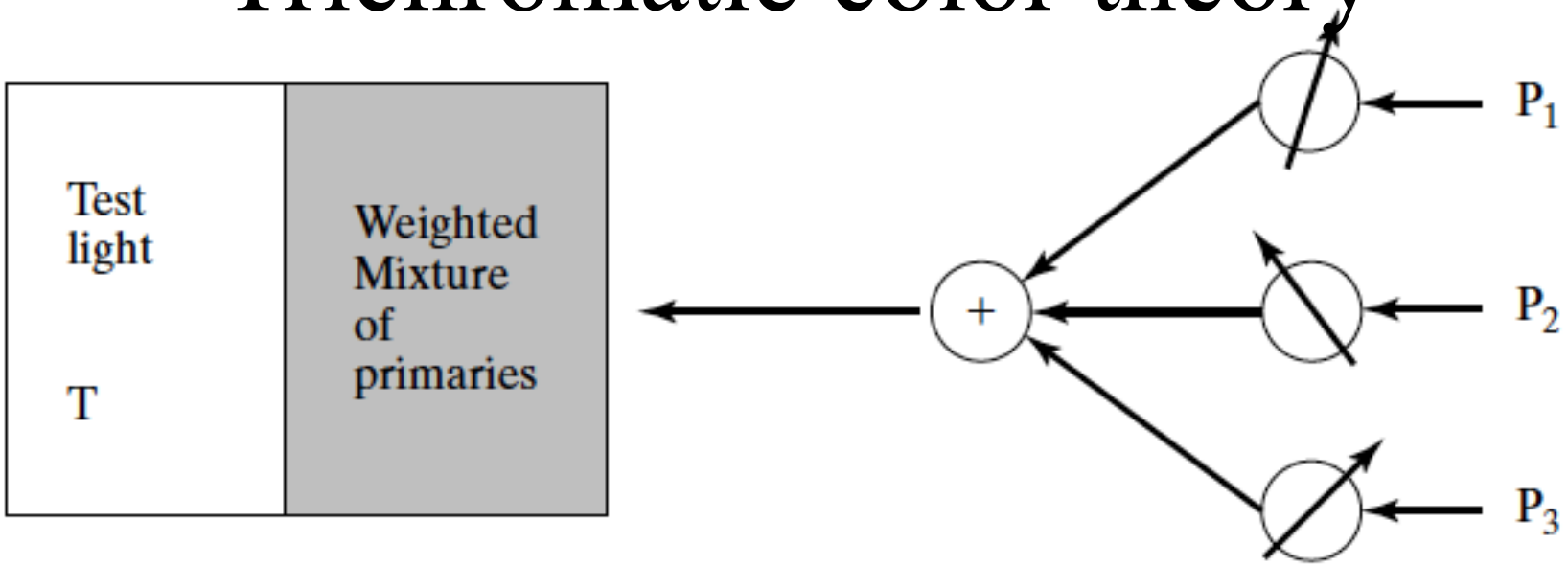
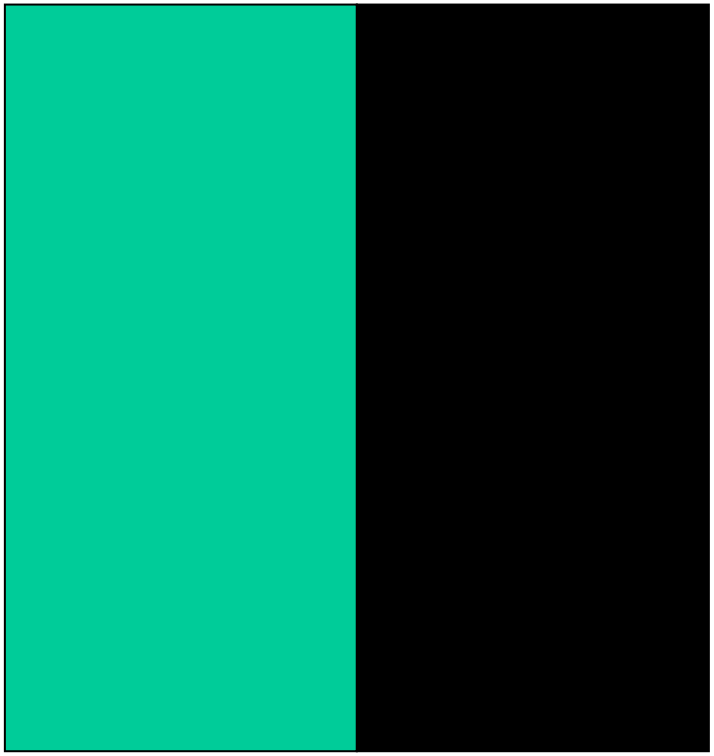
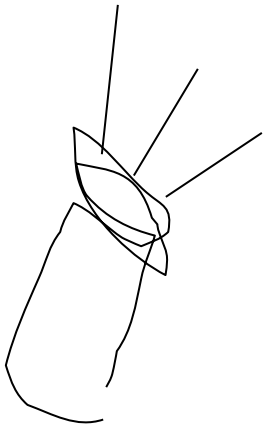


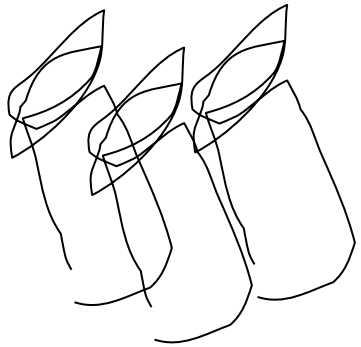
FIGURE 3.2: Human perception of color can be studied by asking observers to mix colored lights to match a test light shown in a split field. The drawing shows the outline of such an experiment. The observer sees a test light T and can adjust the amount of each of three primaries in a mixture displayed next to the test light. The observer is asked to adjust the amounts so that the mixture looks the same as the test light. The mixture of primaries can be written as $w_1P_1 + w_2P_2 + w_3P_3$; if the mixture matches the test light, then we write $T = w_1P_1 + w_2P_2 + w_3P_3$. It is a remarkable fact that for most people three primaries are sufficient to achieve a match for many colors, and three primaries are sufficient for all colors if we allow subtractive matching (i.e., some amount of some of the primaries is mixed with the test light to achieve a match). Some people require fewer primaries. Furthermore, most people choose the same mixture weights to match a given test light.

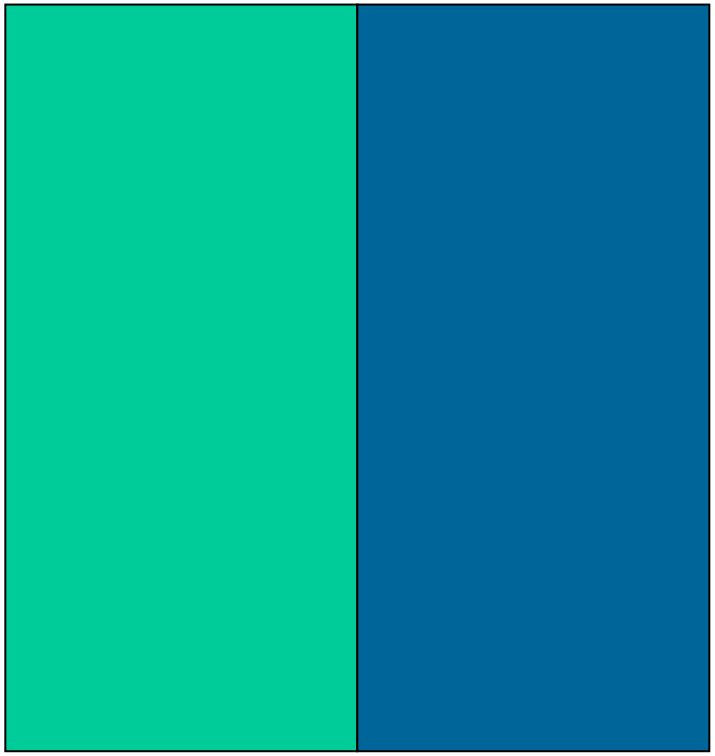
Trichromatic color theory

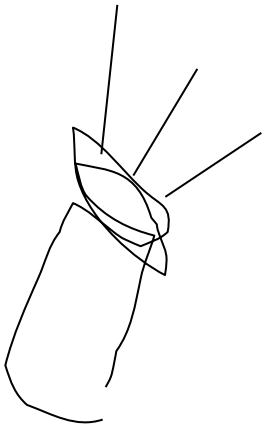
- In color matching experiments, most people can match any given light with three primaries
  - Primaries must be *independent*
- For the same light and same primaries, most people select the same weights
  - Exception: color blindness
- Trichromatic color theory
  - Three numbers seem to be sufficient for encoding color
  - Dates back to 18^{th} century (Thomas Young)

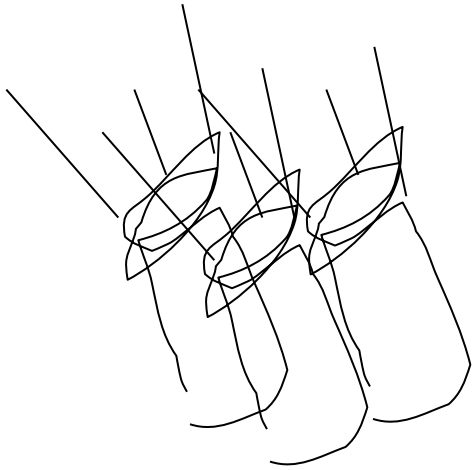


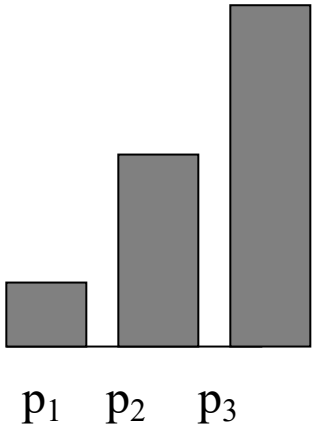


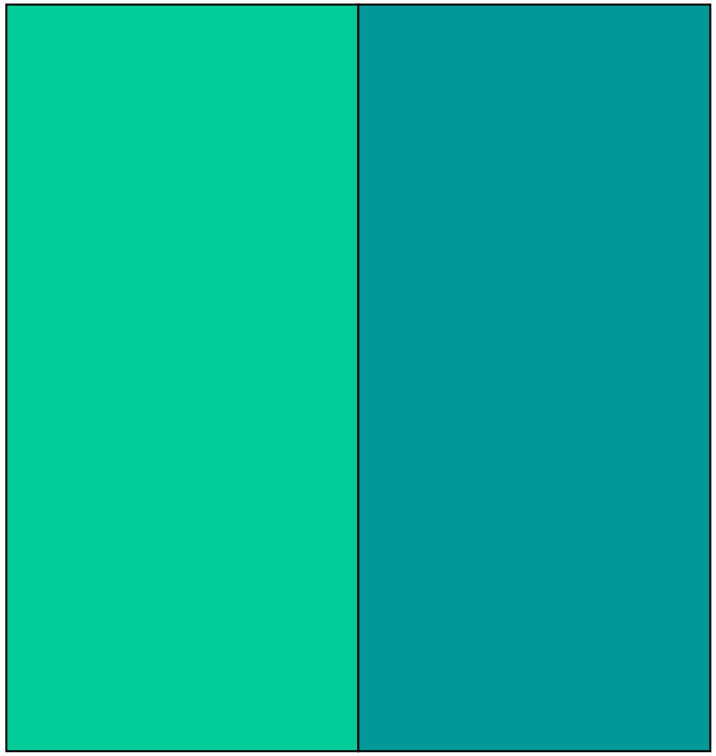


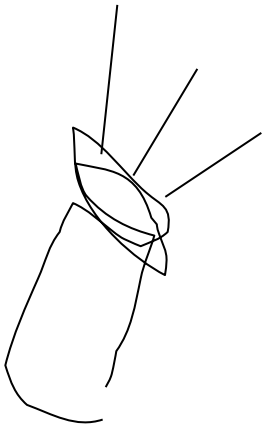


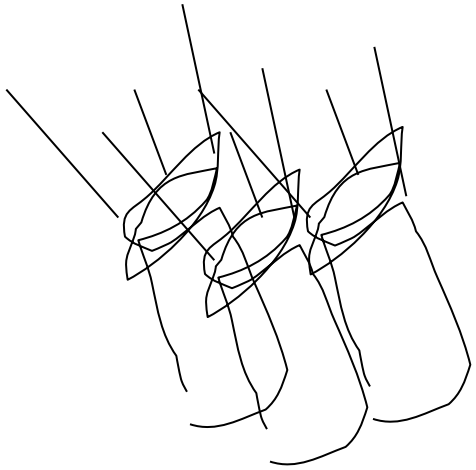


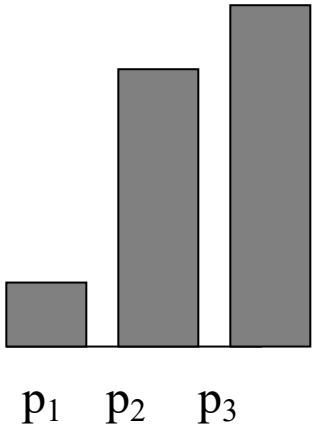


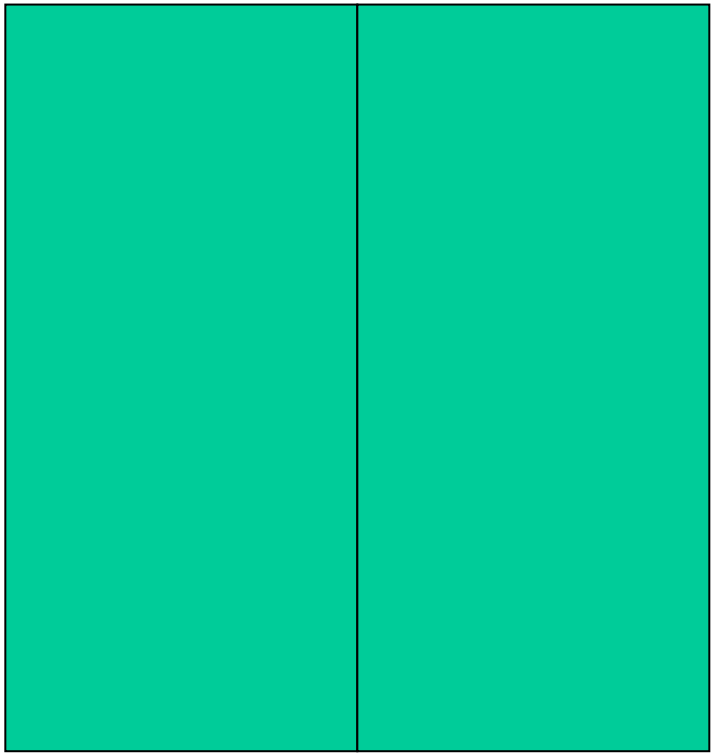

p1
p2
p3



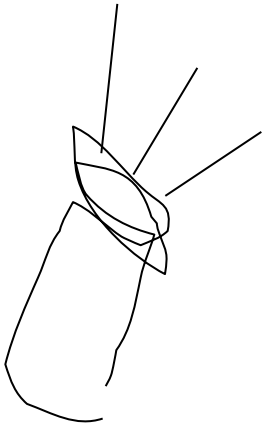


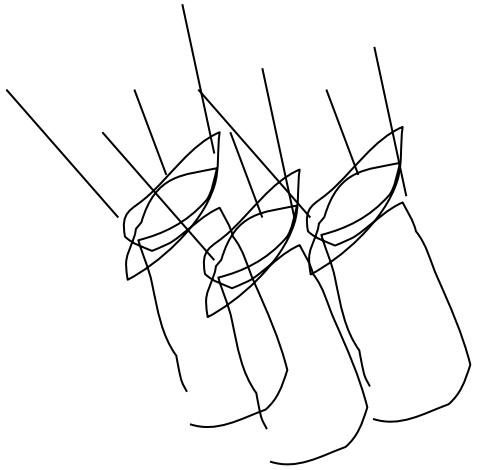


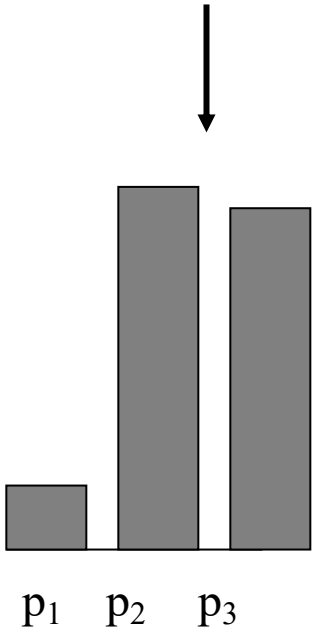

p1
p2
p3

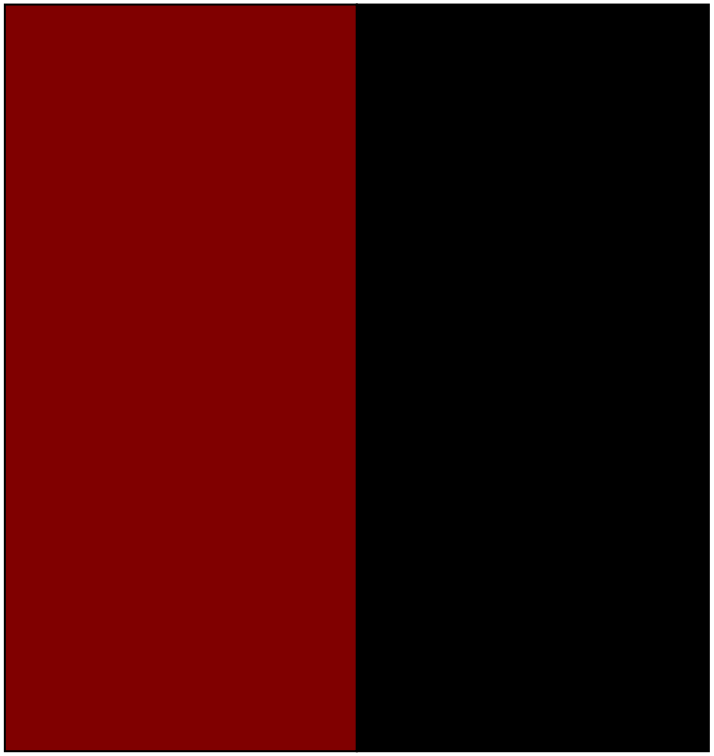


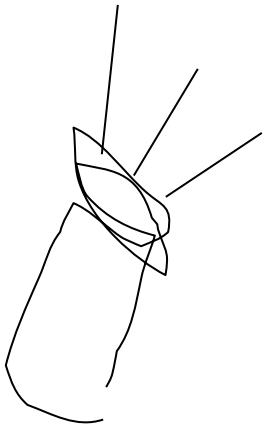
The primary color amounts needed for a match

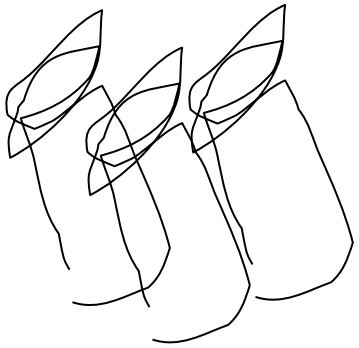


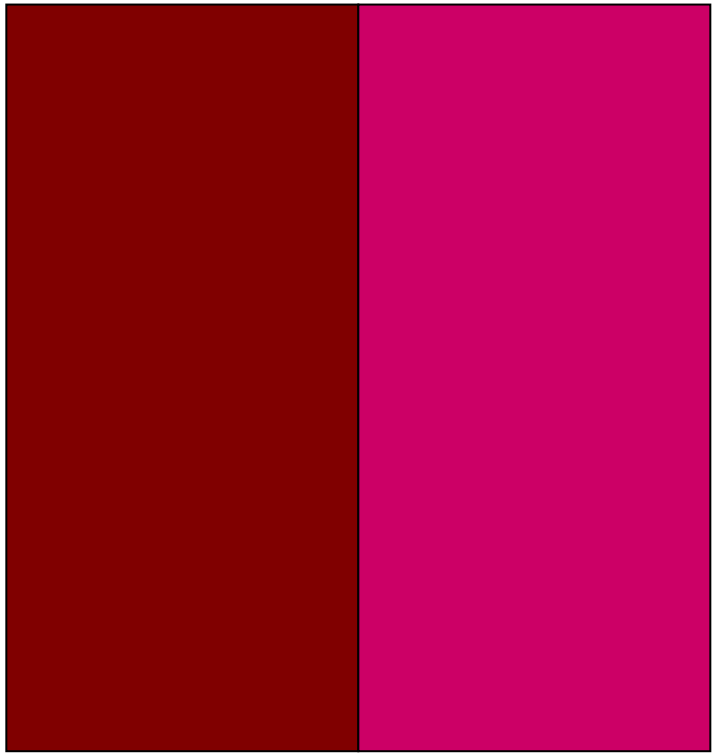


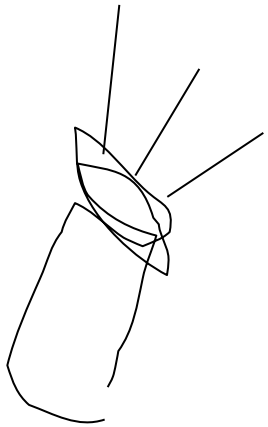


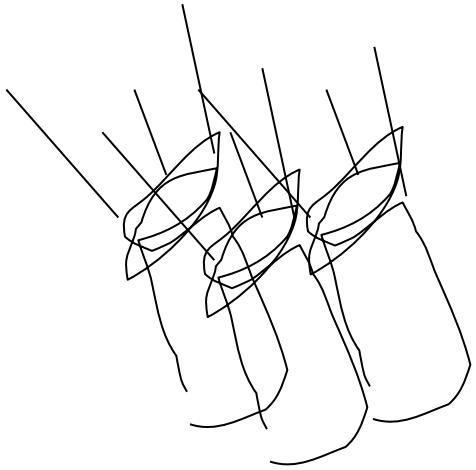


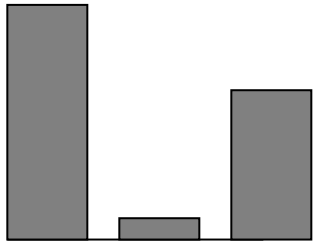




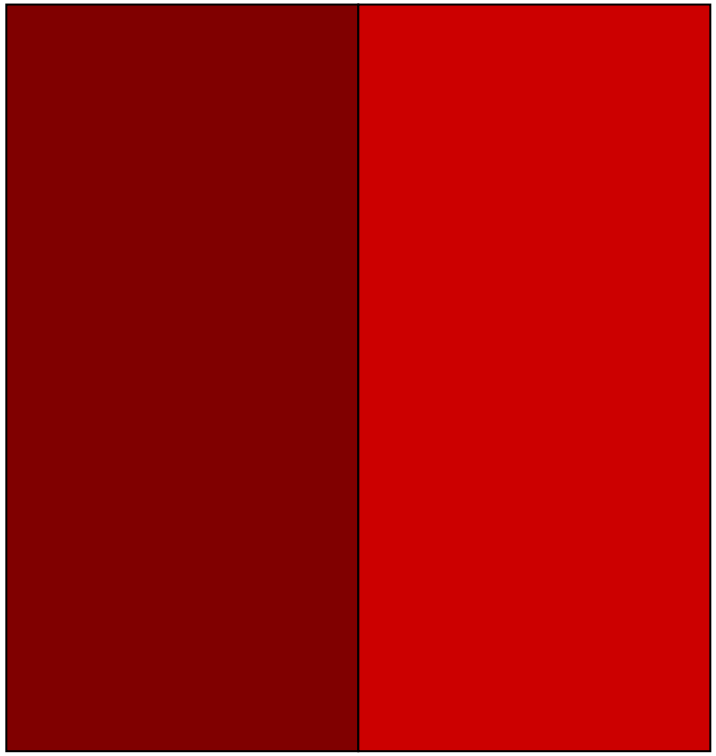


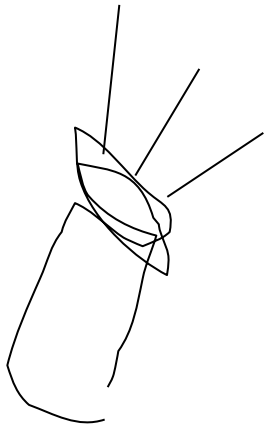


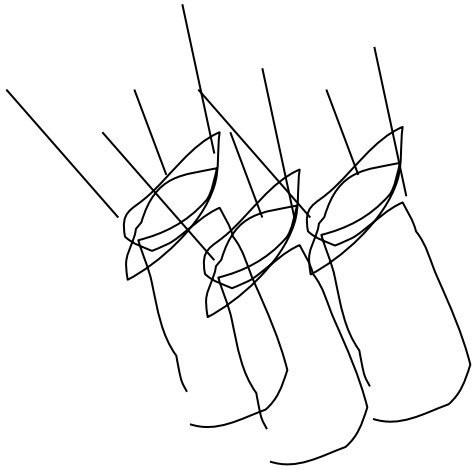


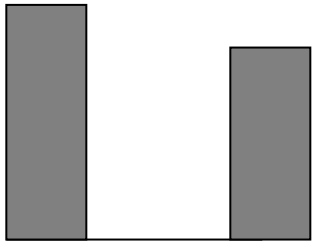


p_1 p_2 p_3



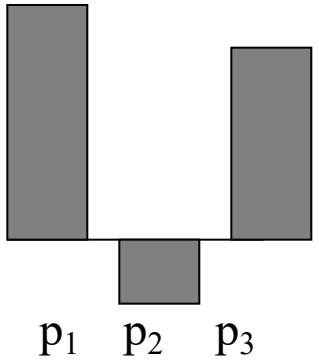




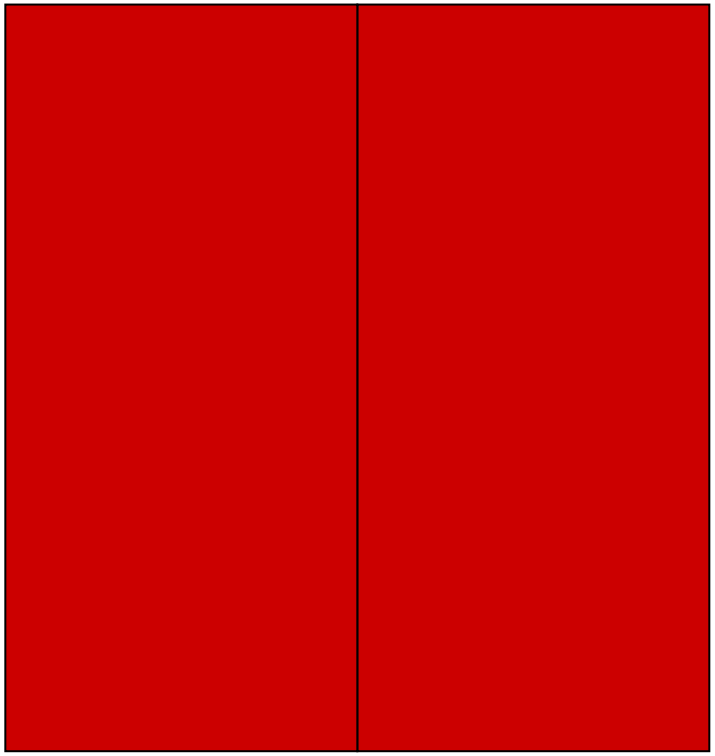


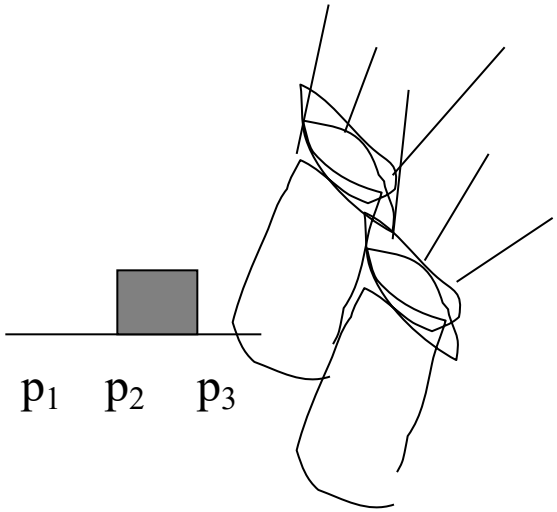
p_1 p_2 p_3

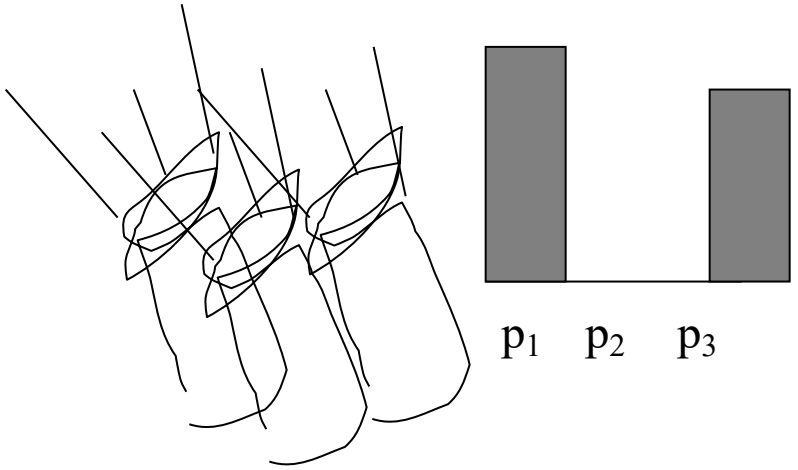
The primary color amounts needed for a match:



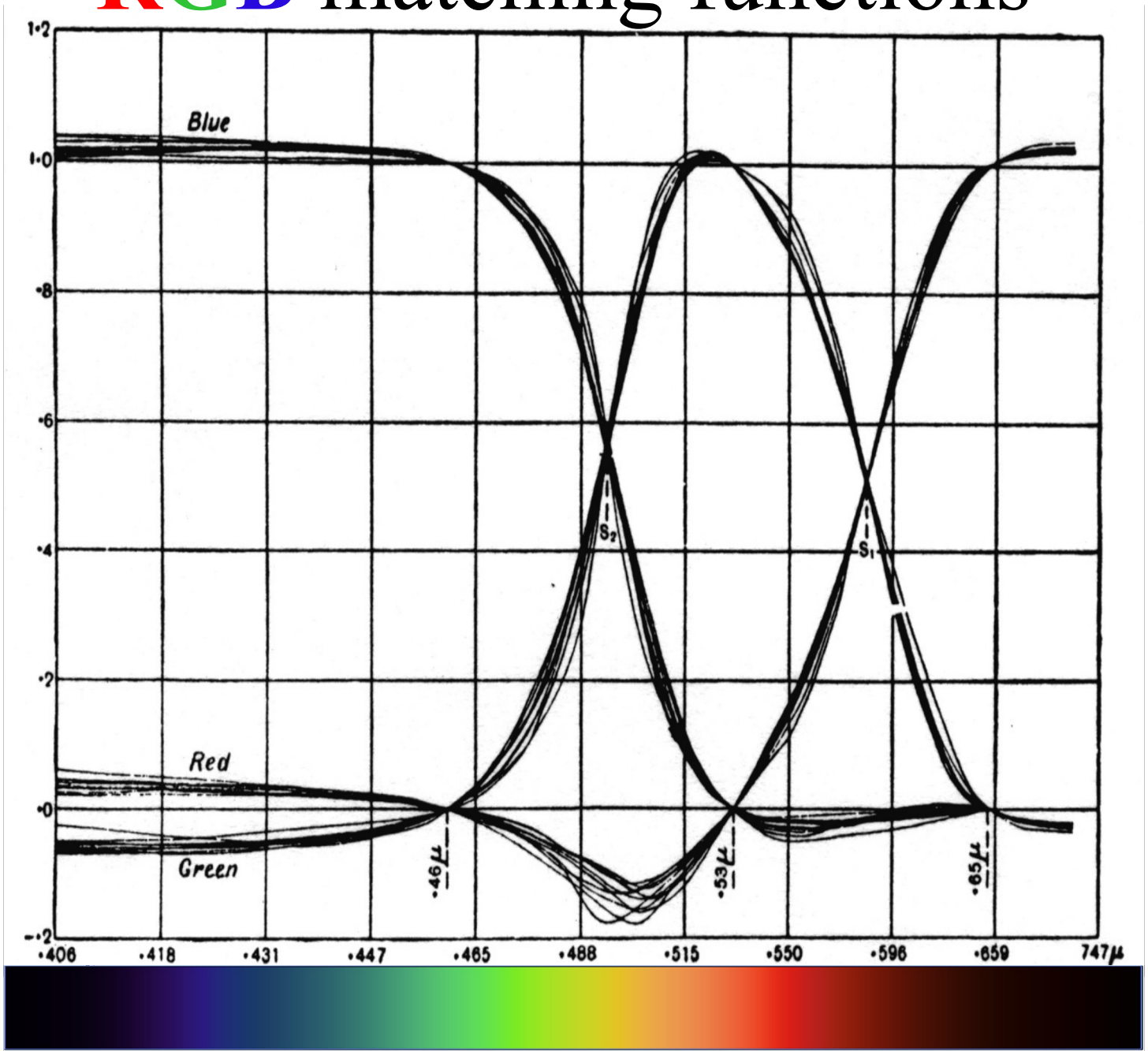
We say a "negative" amount of p_2 was needed to make the match, because we added it to the test color's side.







RGB matching functions



RGB matching functions