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, **template matching**

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

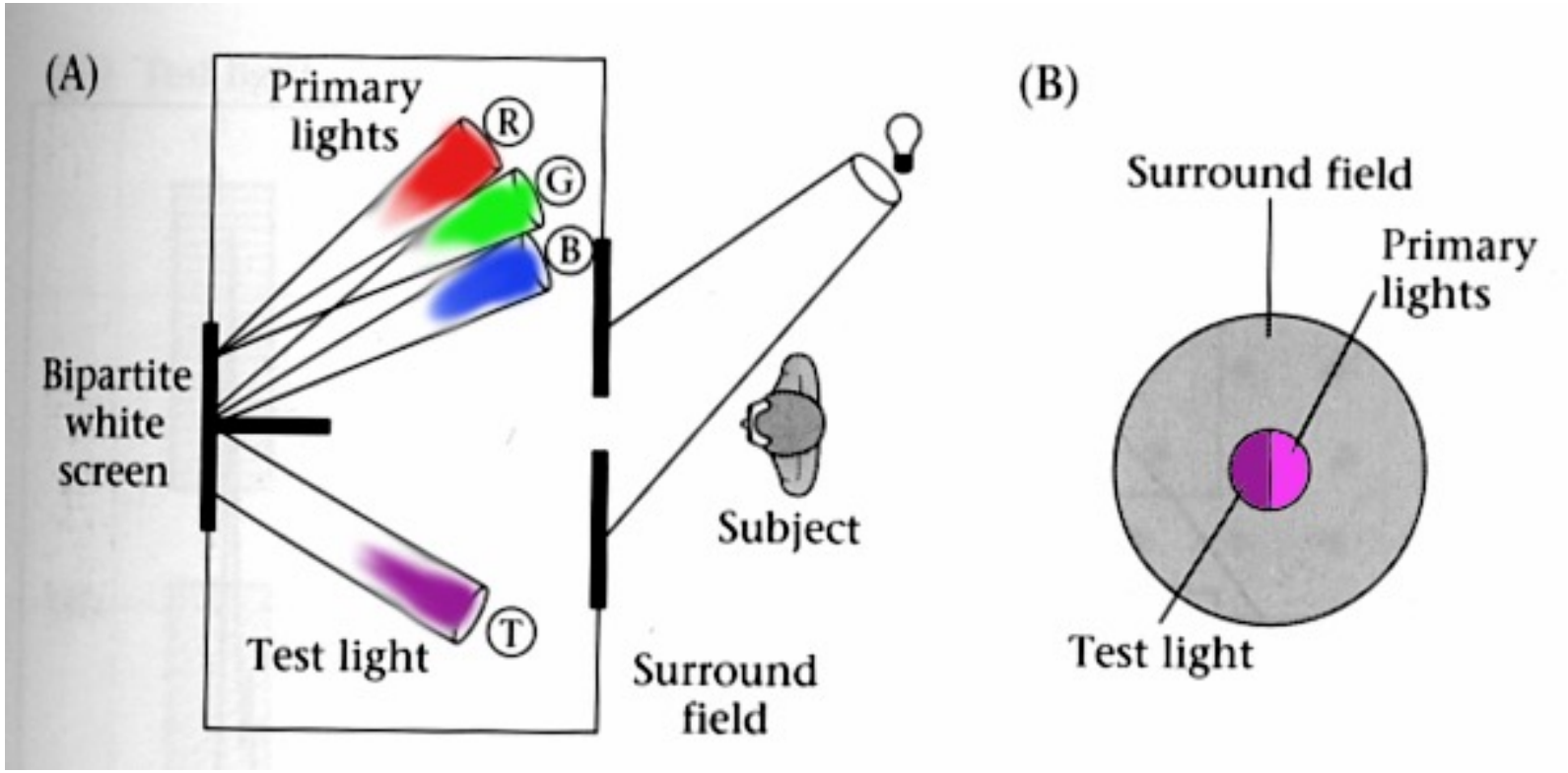
5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Color

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

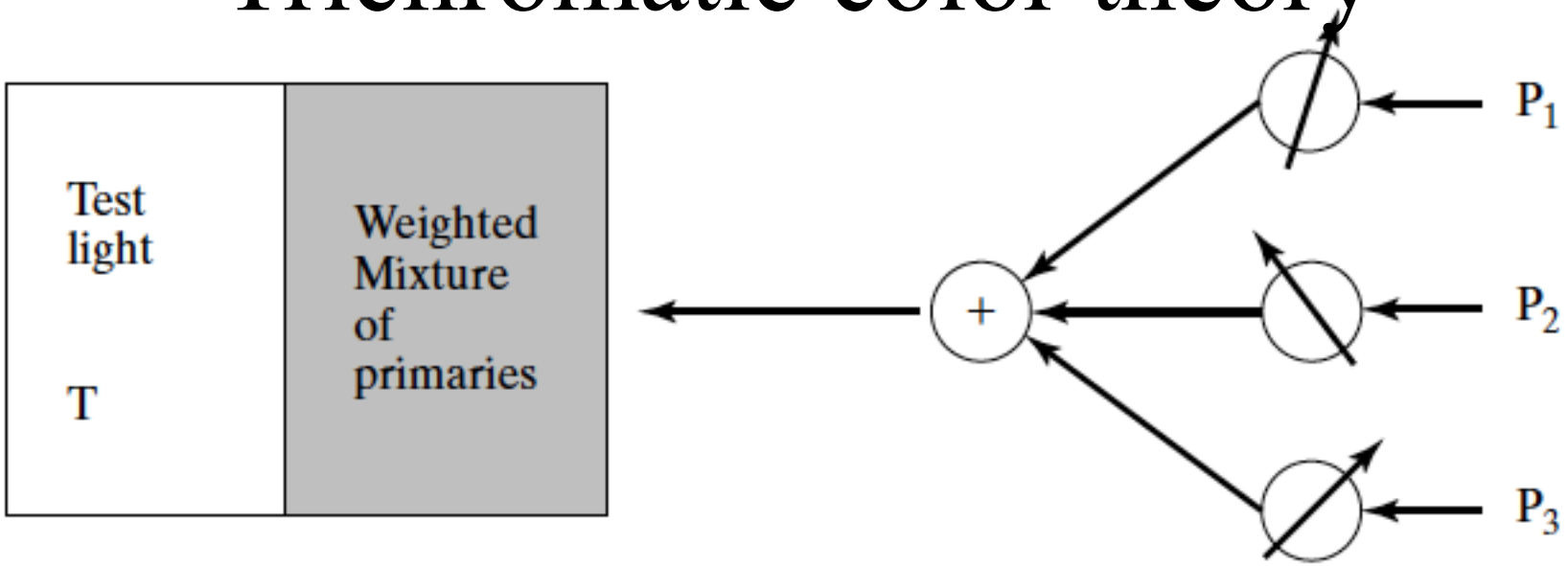
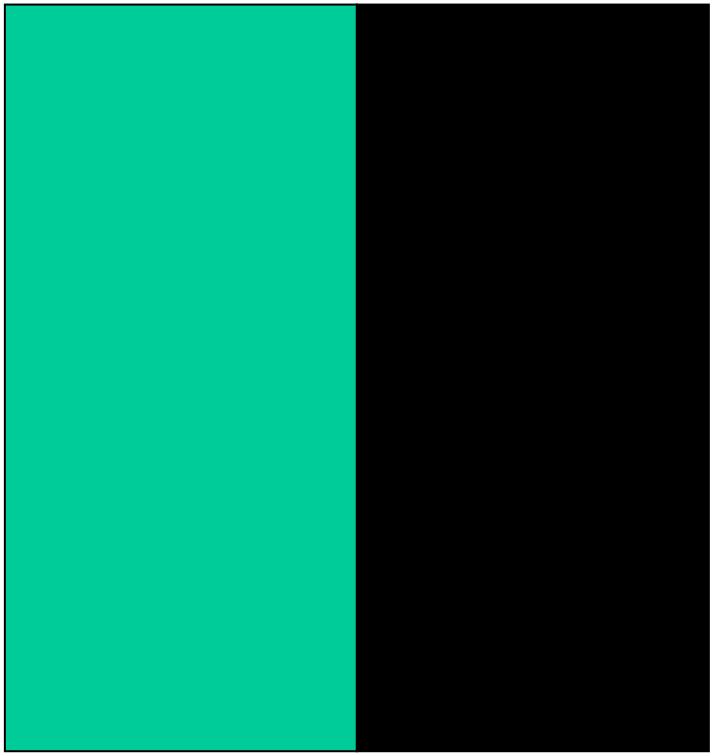
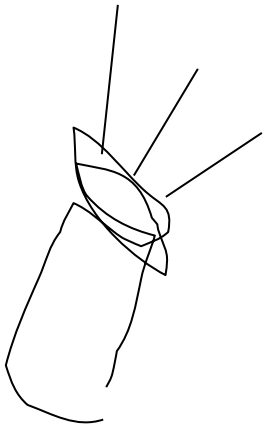


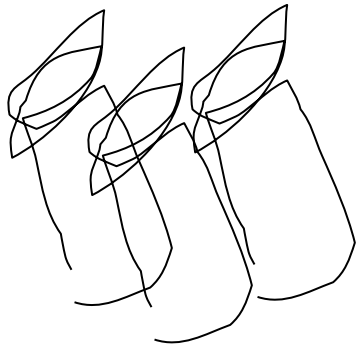
FIGURE 3.2: Human perception of color can be studied by asking observers to mix colored lights to match a test light shown in a split field. The drawing shows the outline of such an experiment. The observer sees a test light T and can adjust the amount of each of three primaries in a mixture displayed 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.

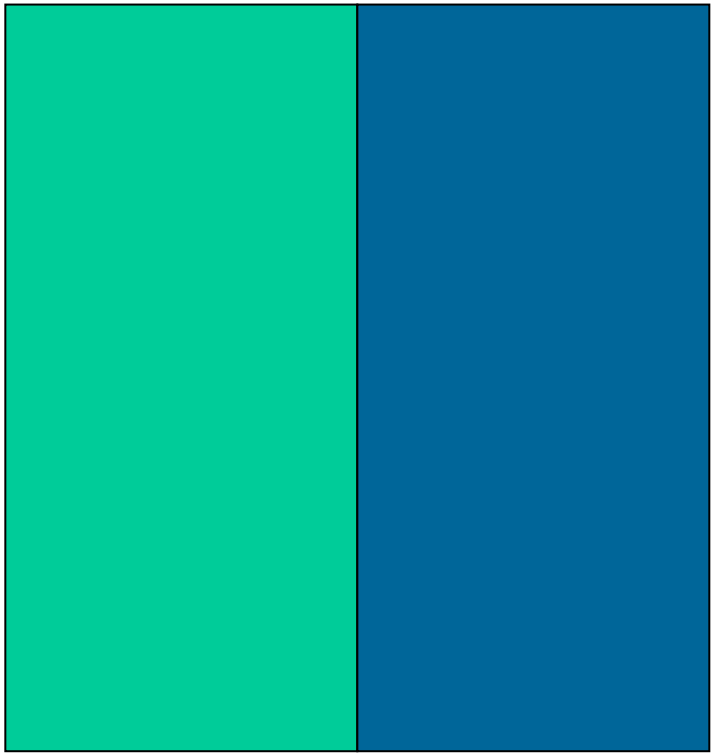
Trichromatic color theory

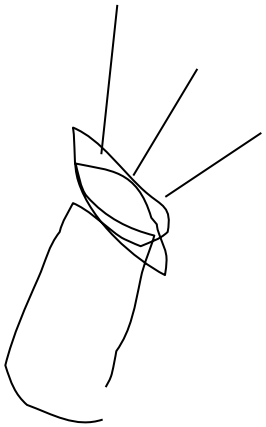
- In color matching experiments, most people can match any given light with three primaries
  - Primaries must be *independent*
- For the same light and same primaries, most people select the same weights
  - Exception: color blindness
- Trichromatic color theory
  - Three numbers seem to be sufficient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

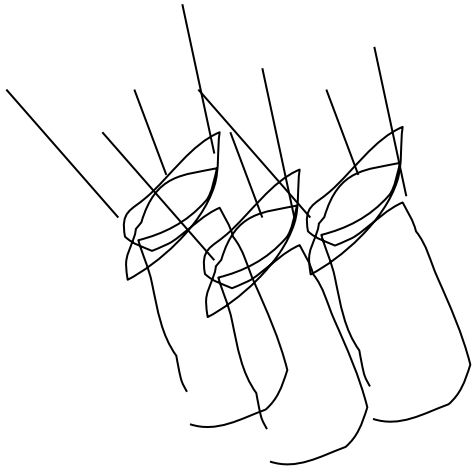


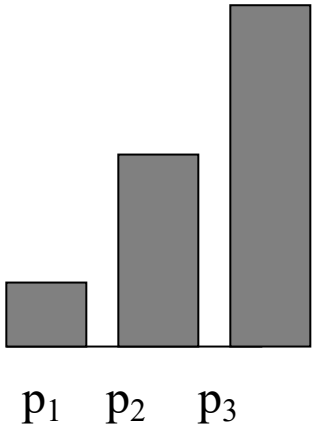


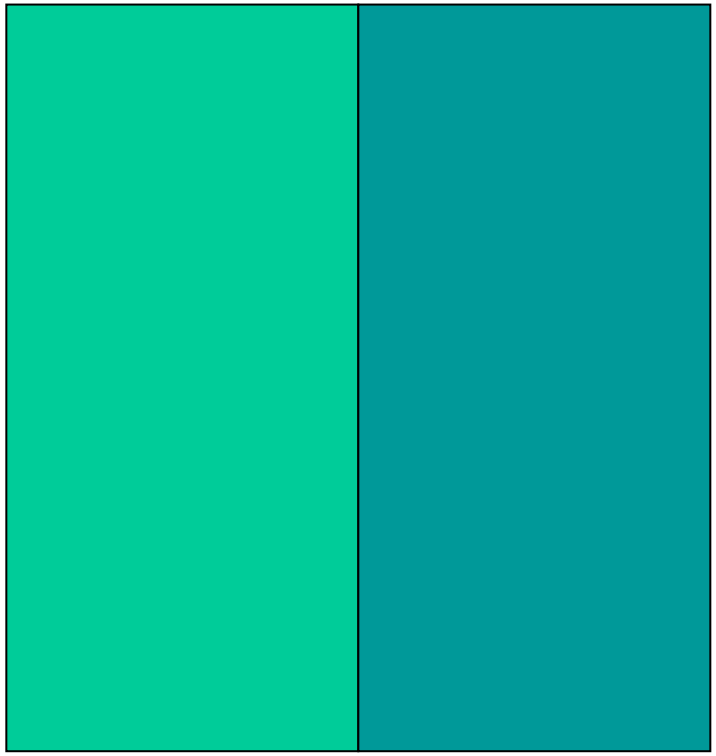


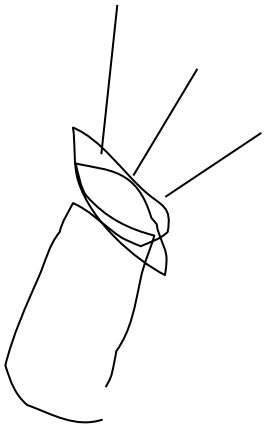


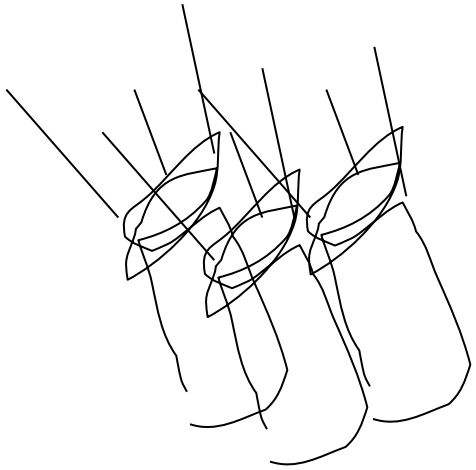


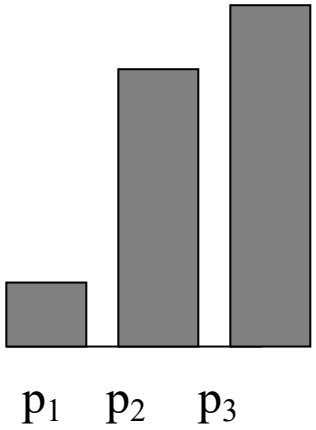


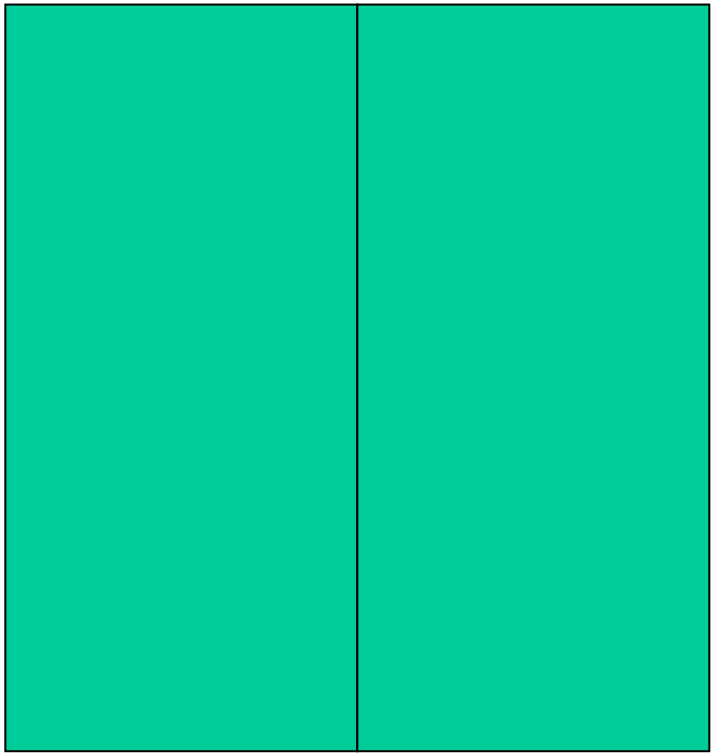

p1
p2
p3



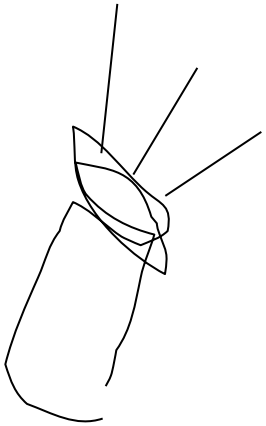


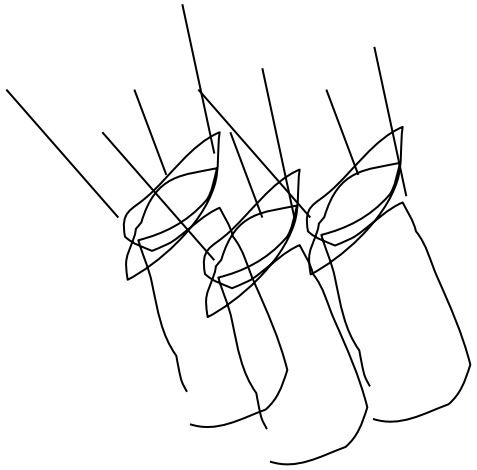


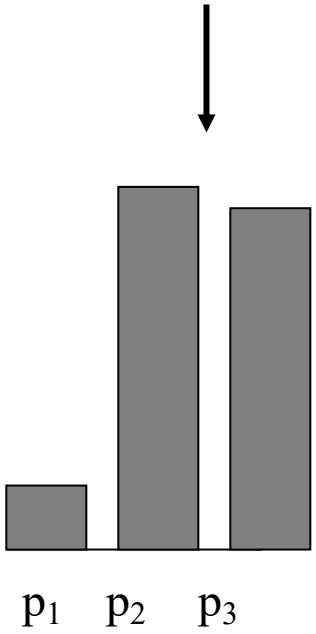

p1
p2
p3

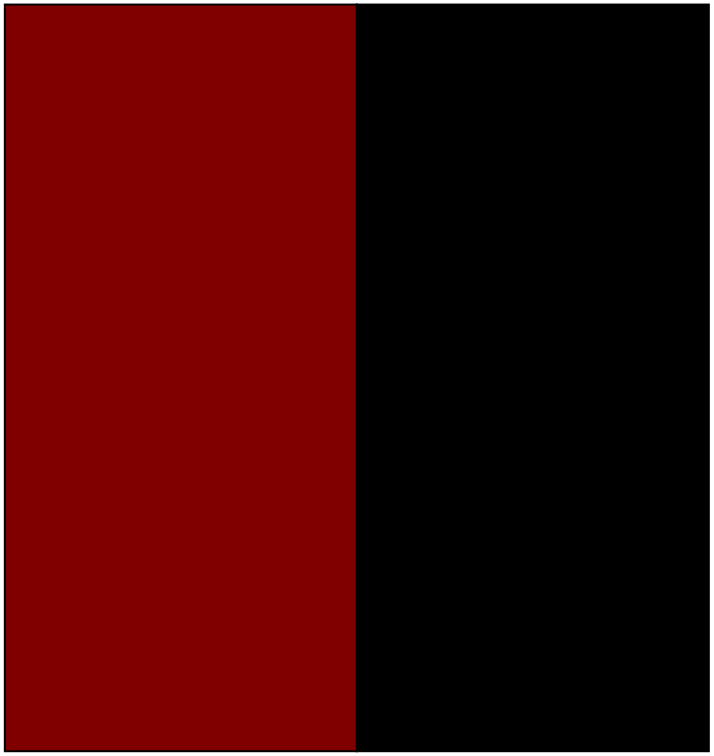


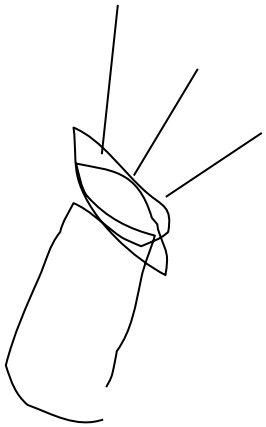
The primary color amounts needed for a match

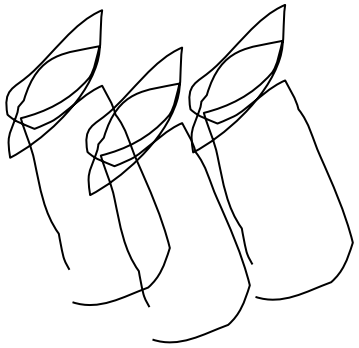


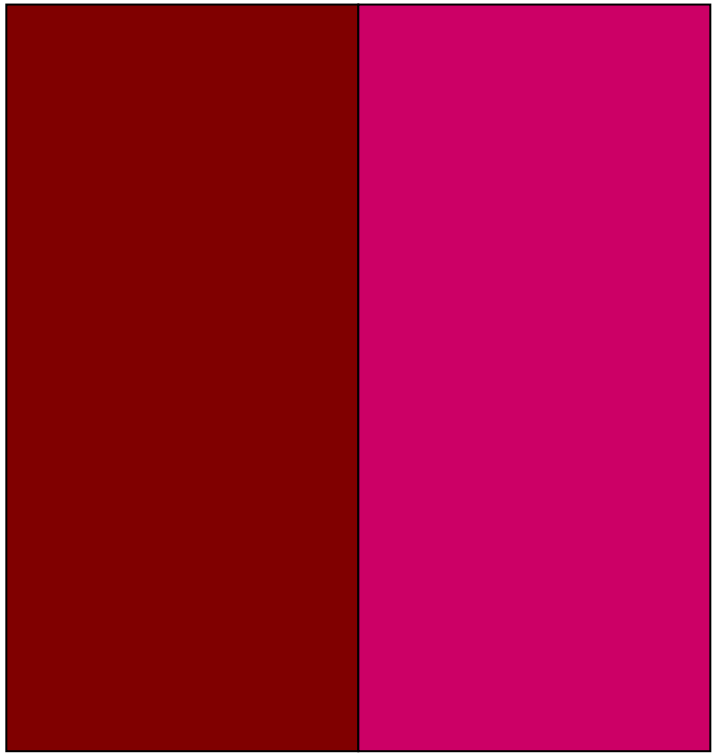


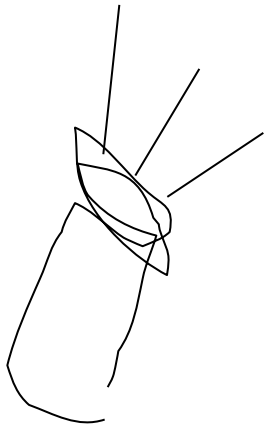


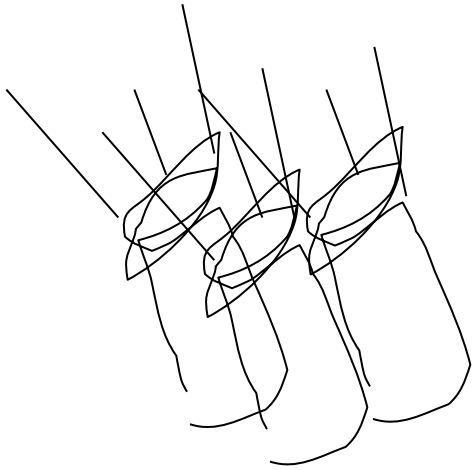


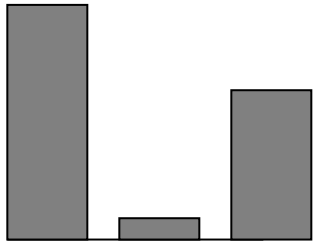




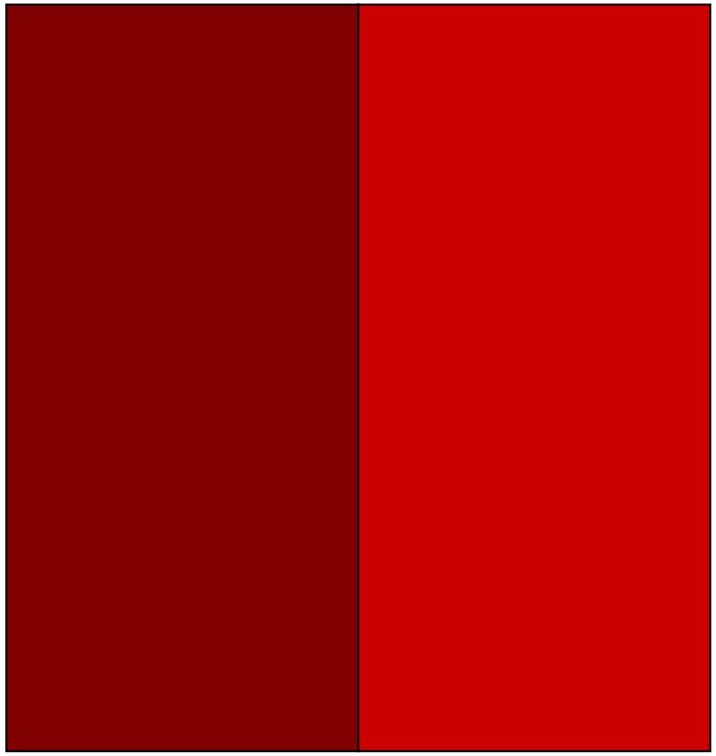


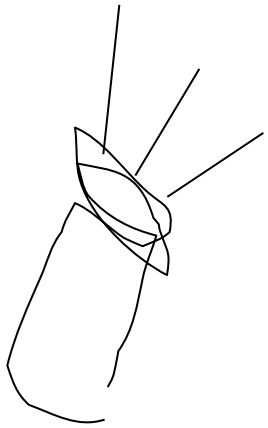


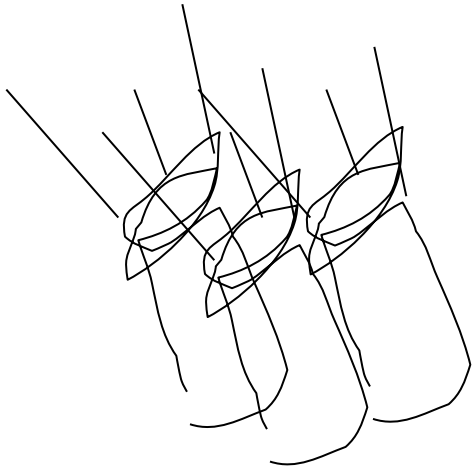


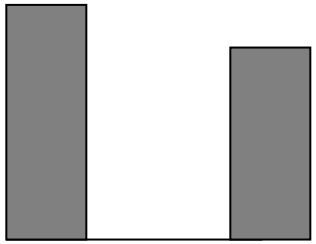


p_1 p_2 p_3



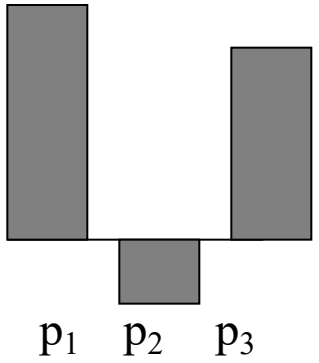




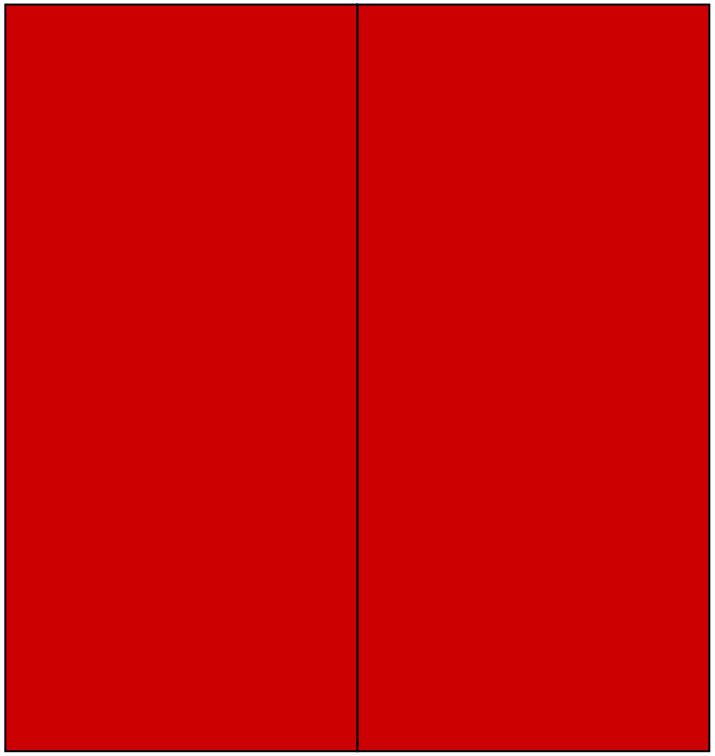


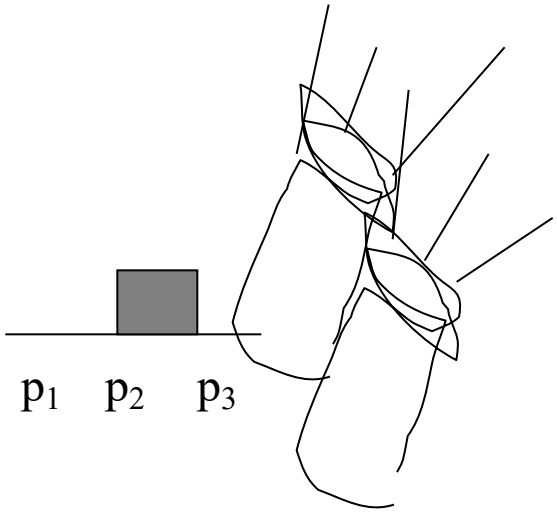
p_1 p_2 p_3

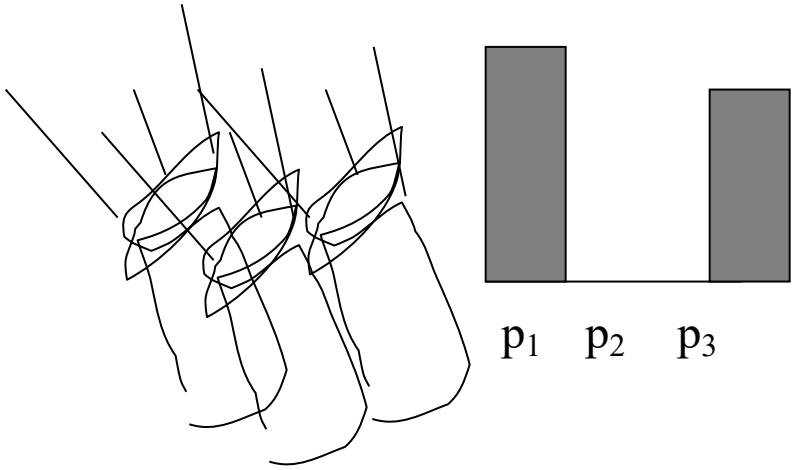
The primary color amounts needed for a match:



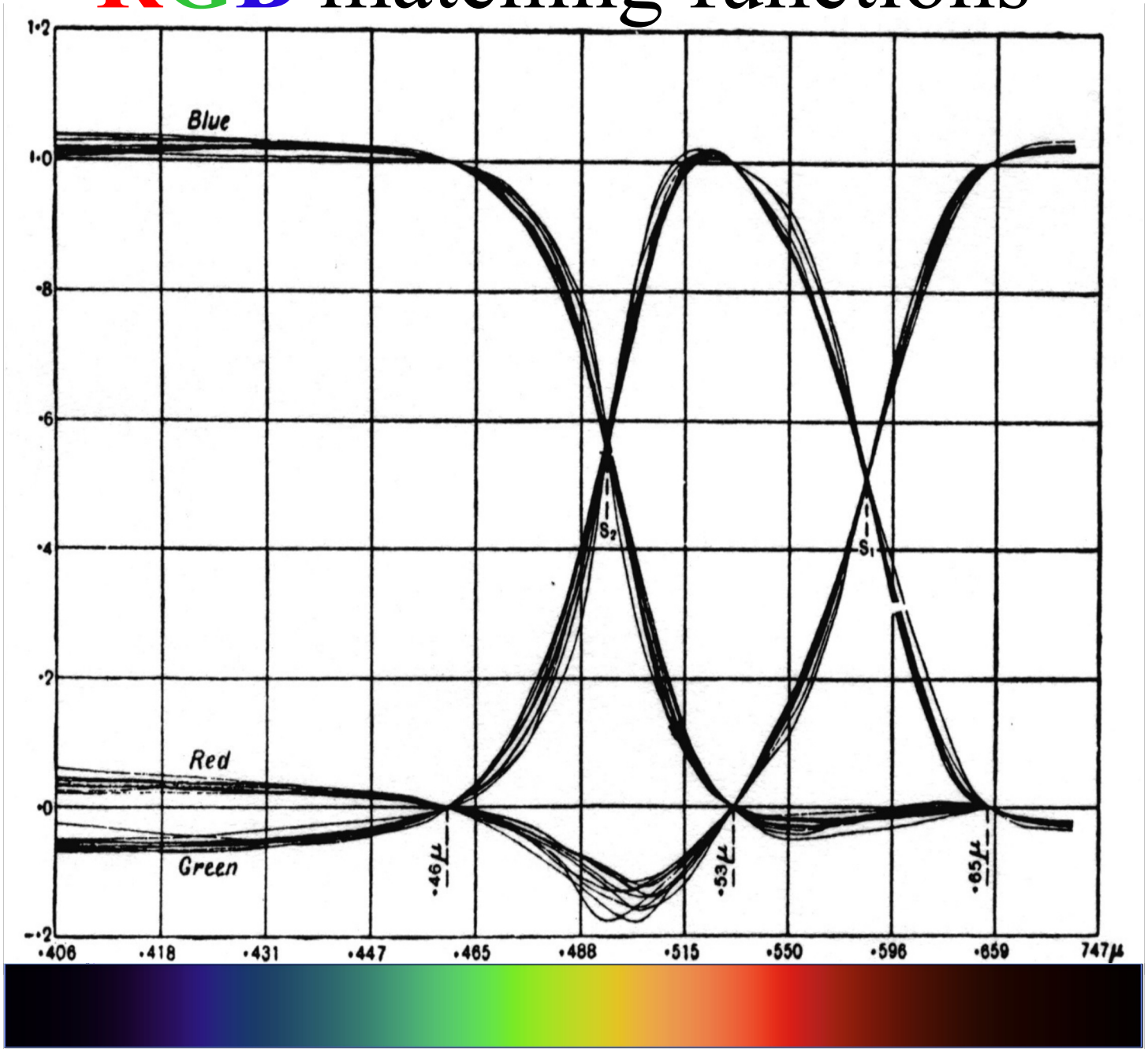
We say a "negative" amount of p_2 was needed to make the match, because we added it to the test color's side.



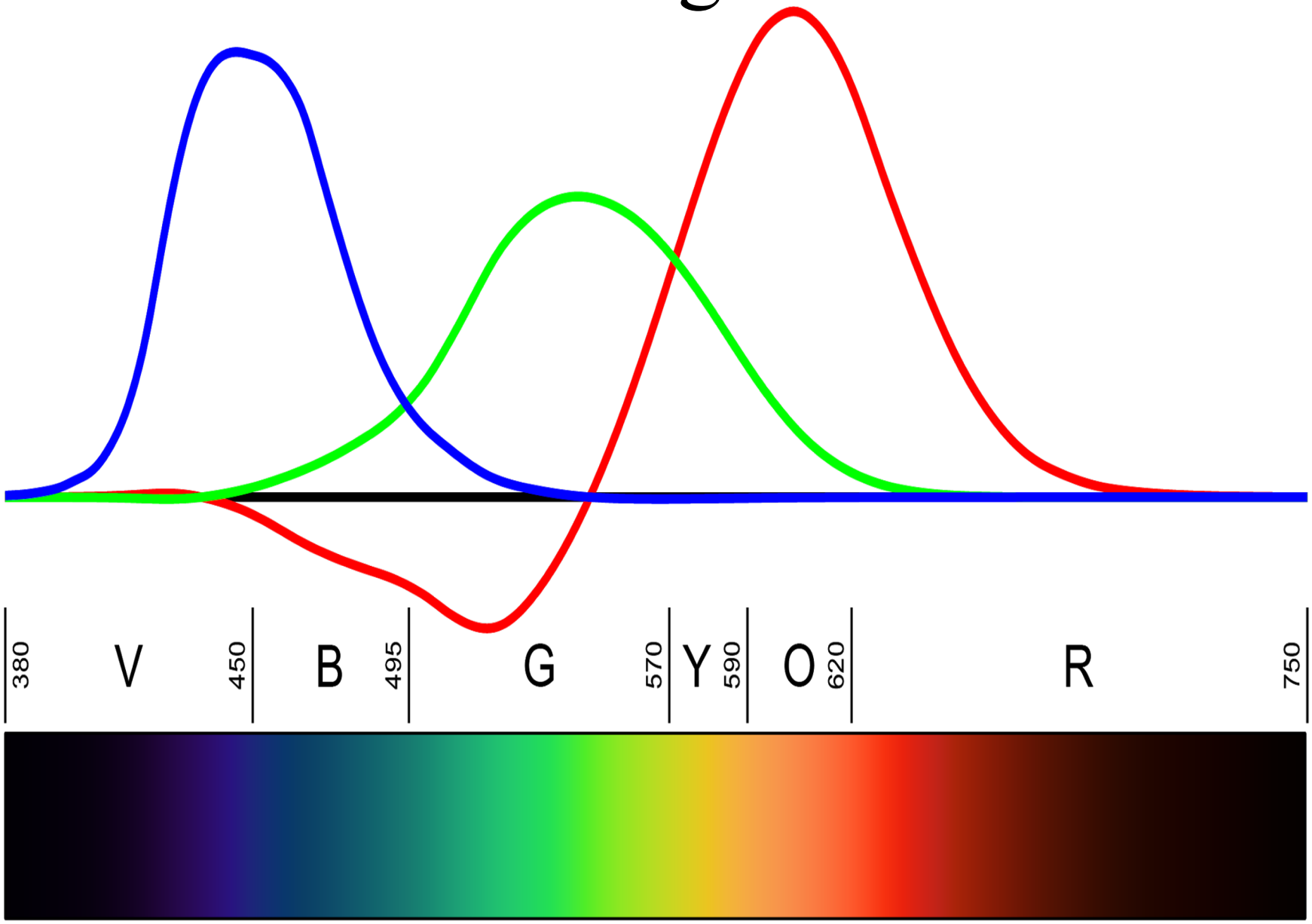




RGB matching functions



RGB matching functions

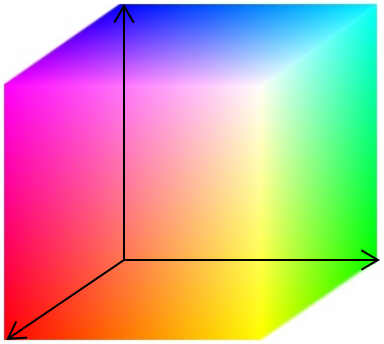


Linear color spaces

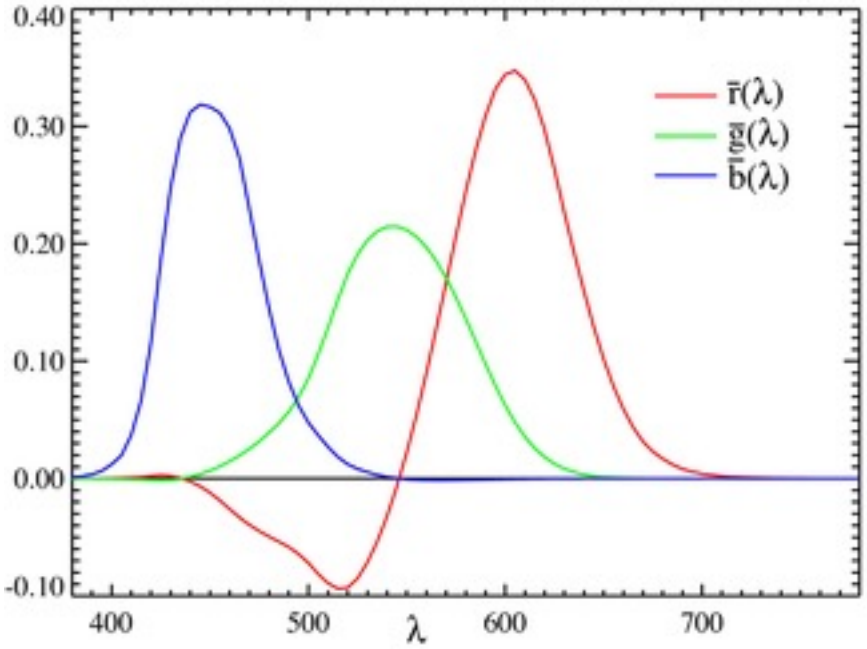
- Defined by a choice of three **primaries**
- The **coordinates** of a color are given by the weights of the primaries used to match it
- In addition to primaries, need to specify **matching functions:** the amount of each primary needed to match a monochromatic light source at each wavelength

Linear color spaces - RGB

- Defined by a choice of three **primaries**
- The **coordinates** of a color are given by the weights of the primaries used to match it
- In addition to primaries, need to specify **matching functions:** the amount of each primary needed to match a monochromatic light source at each wavelength



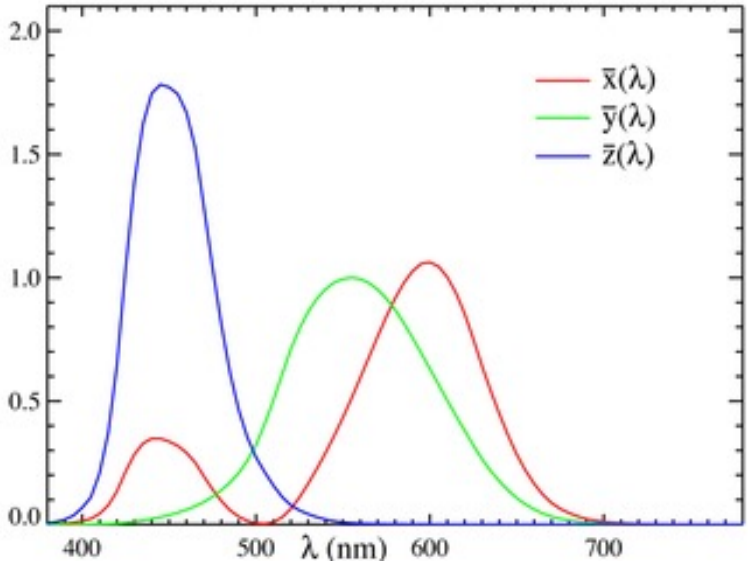
RGB matching functions



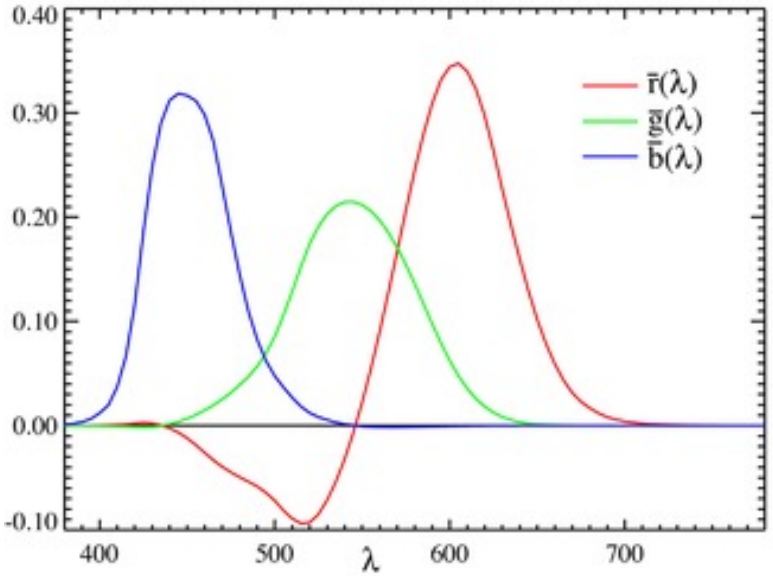
Linear color spaces: CIE XYZ

- Primaries are *imaginary*, but matching functions are everywhere positive
- The Y parameter corresponds to brightness or *luminance* of a color
- Z is blue
- X is a mix of response curves chosen to be nonnegative

XYZ matching functions



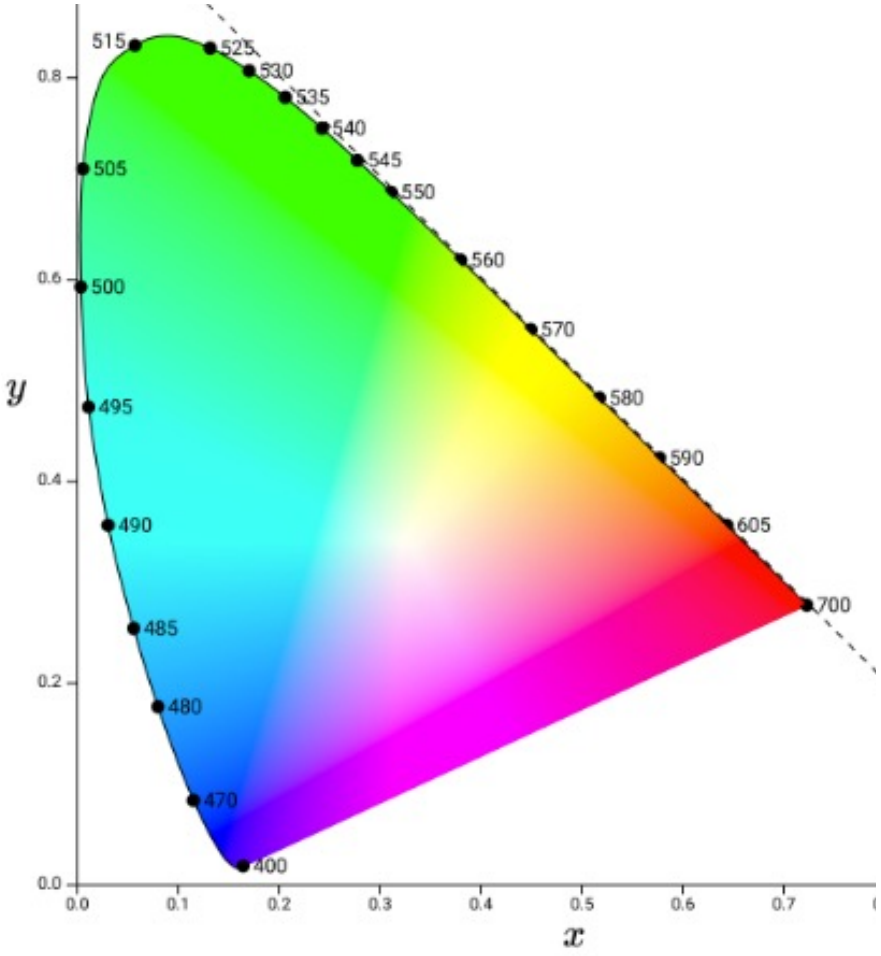
RGB matching functions



http://en.wikipedia.org/wiki/CIE_1931_color_space

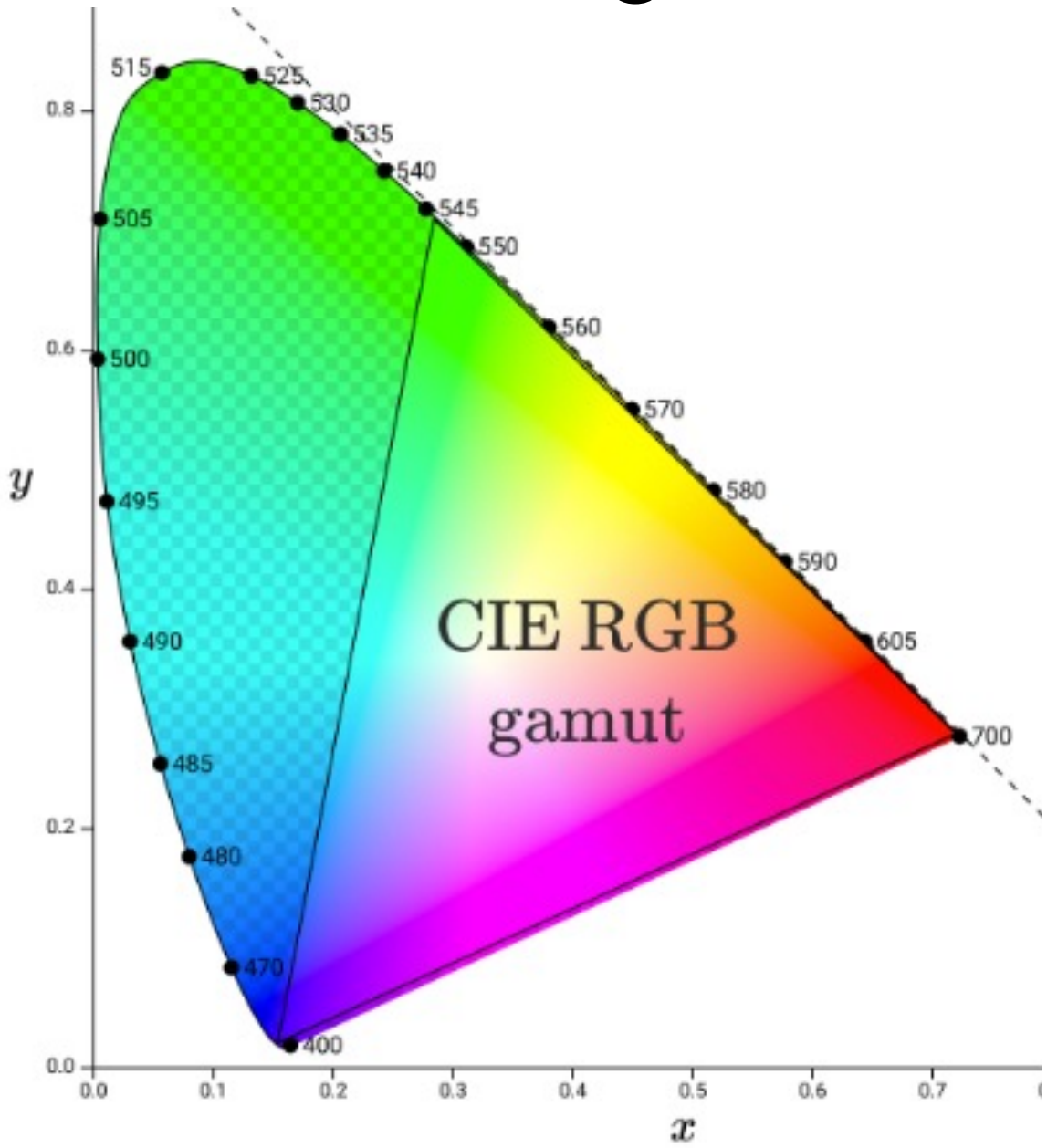
The xy chromaticity diagram

- the concept of color can be divided into two parts: brightness and chromaticity.
- white and grey have the same chromaticity while their brightness differs
- go from 3D (color) to 2D (chromaticity)
- 2D visualization: draw (x,y), where
  $x = X/(X+Y+Z)$, $y = Y/(X+Y+Z)$

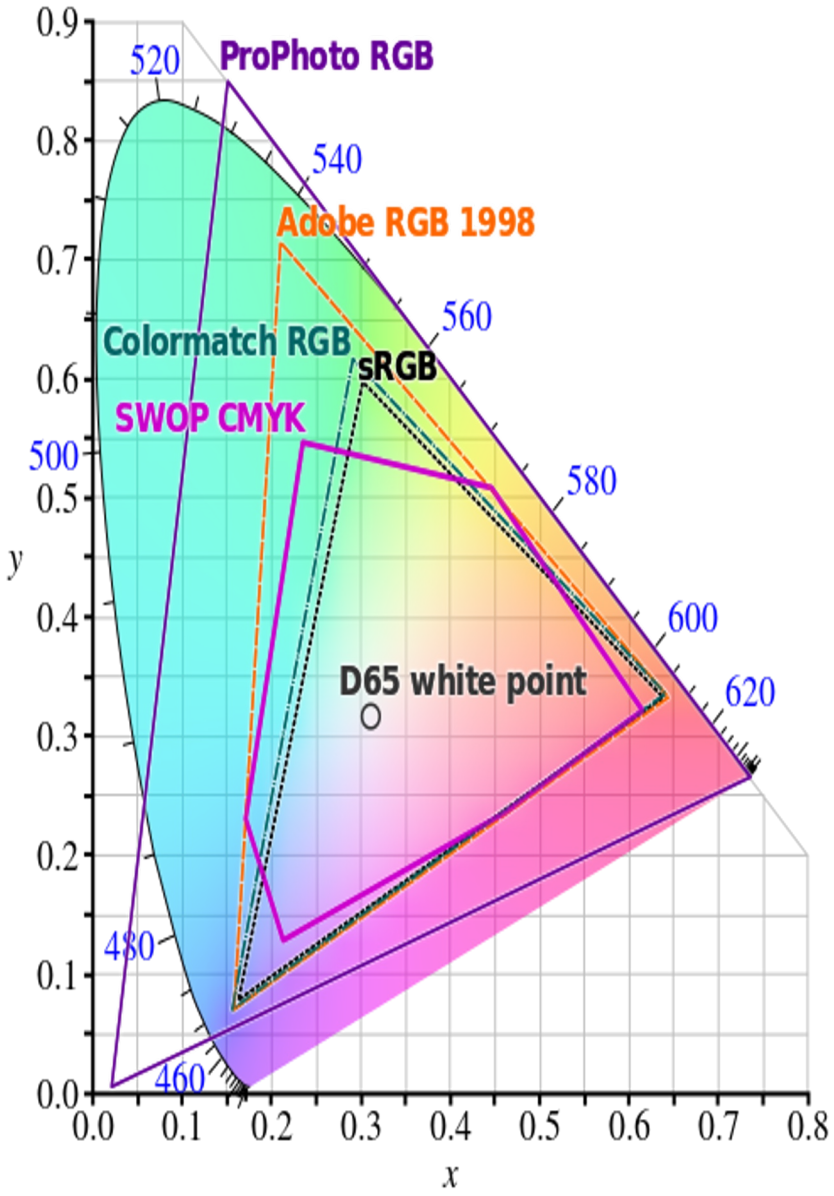


Chromaticity diagram

The RGB gamut

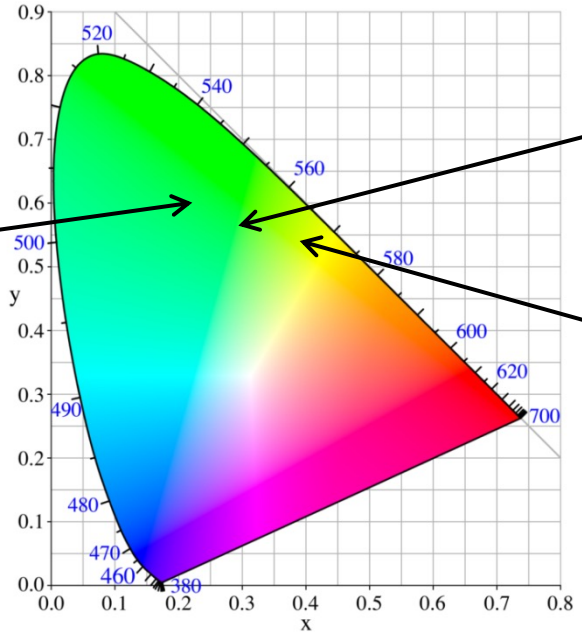


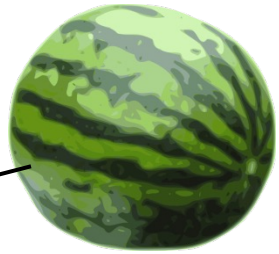
MANY different colorspaces



Distances in color space

- Are distances between points in a color space perceptually meaningful?

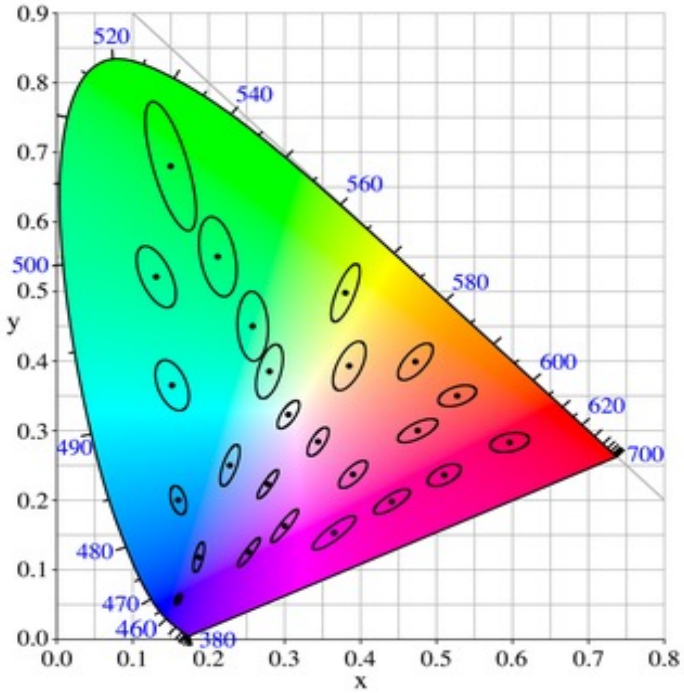




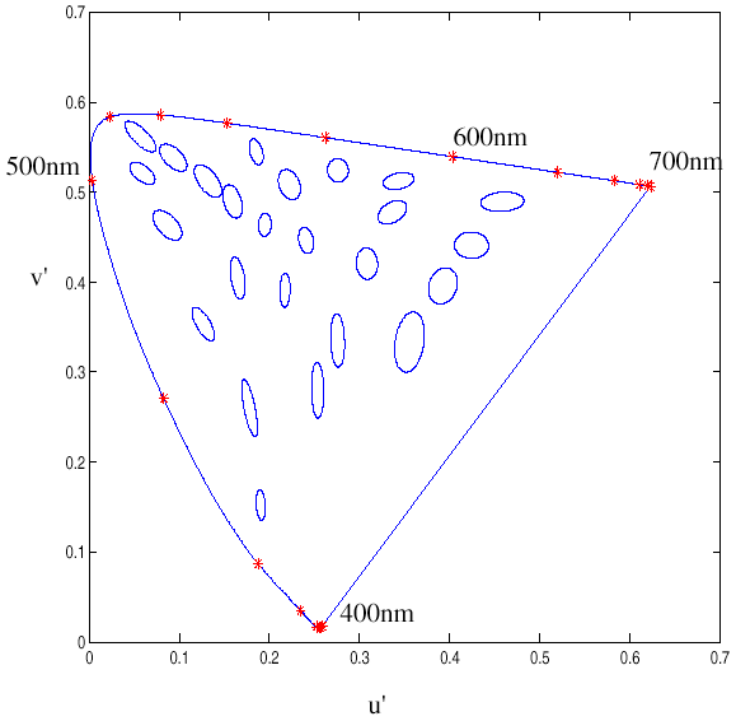
- Not necessarily: CIE XYZ is **not** a *uniform* color space, so magnitude of differences in coordinates are poor indicator of color "distance".

Uniform color spaces

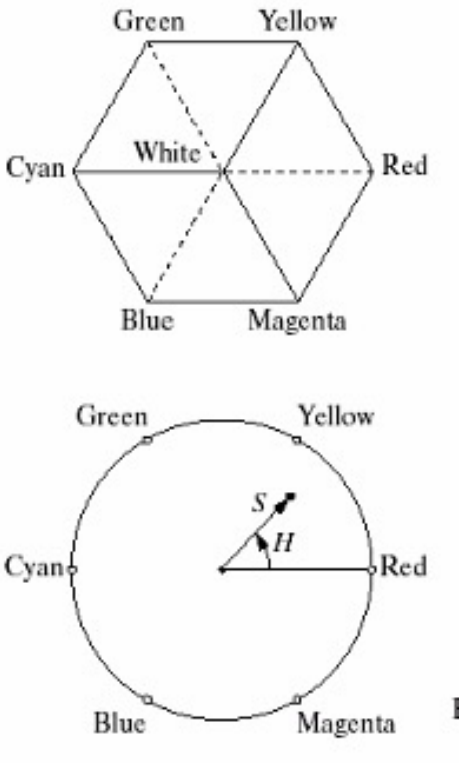
- Unfortunately, differences in x,y coordinates do not reflect perceptual color differences
- CIE u'v' is a projective transform of x,y to make the ellipses more uniform

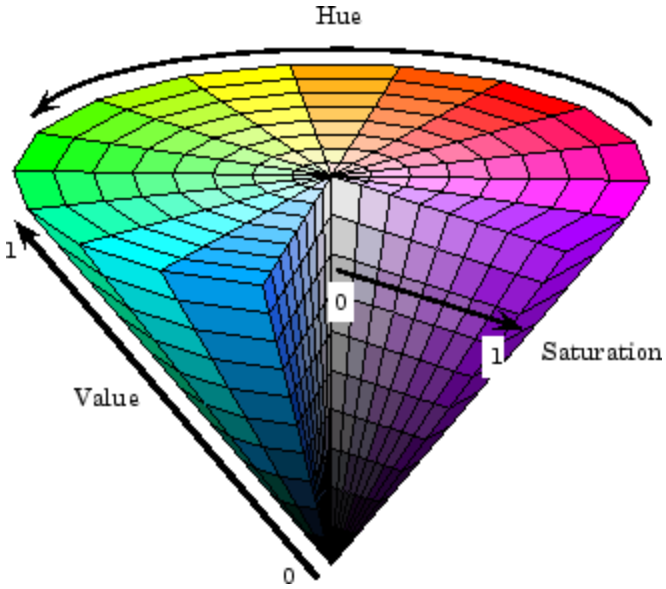


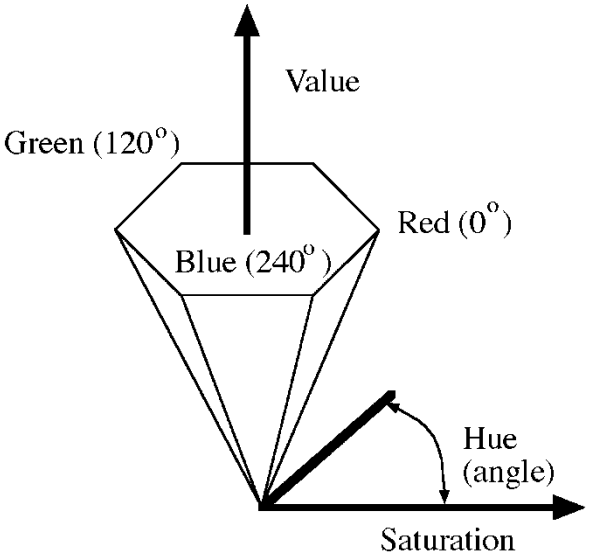
McAdam ellipses: Just noticeable differences in color



Nonlinear color spaces: HSV

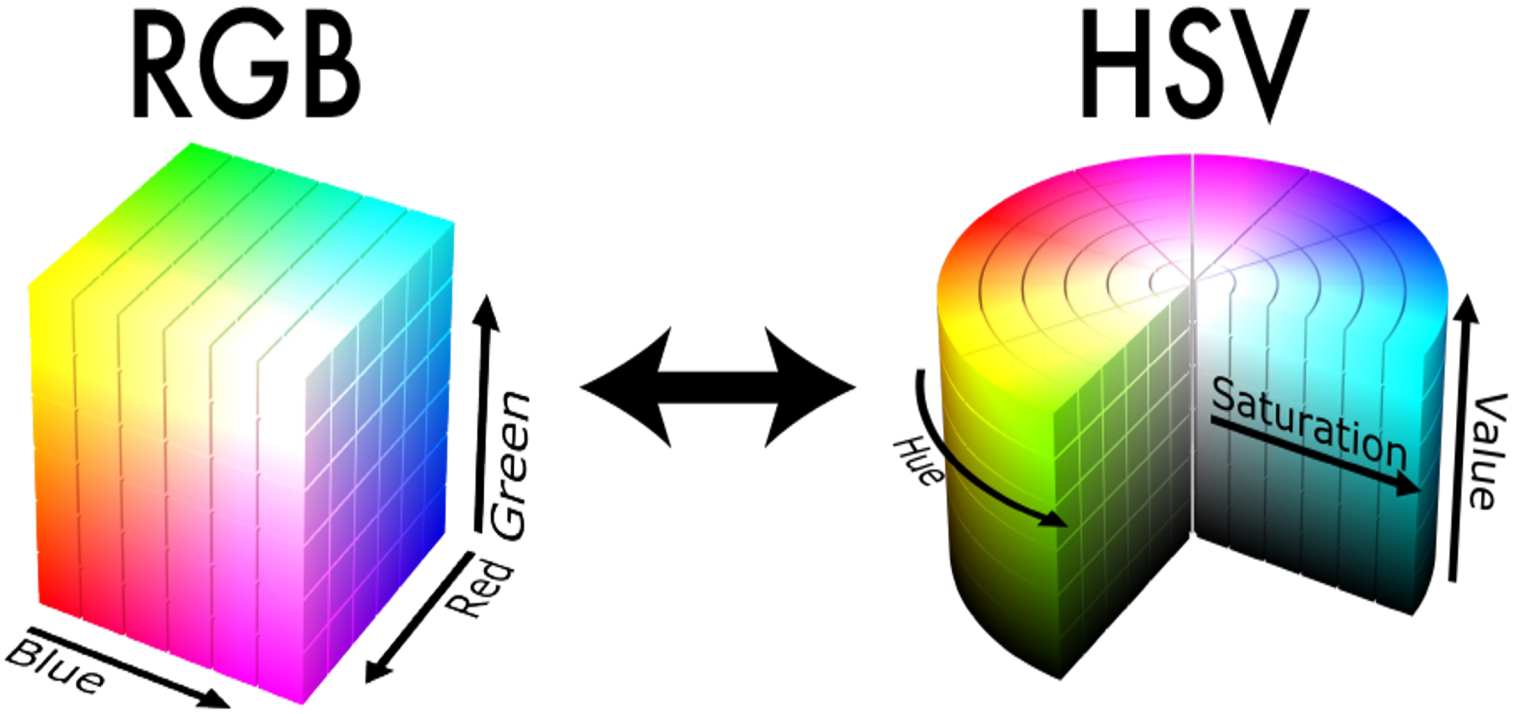






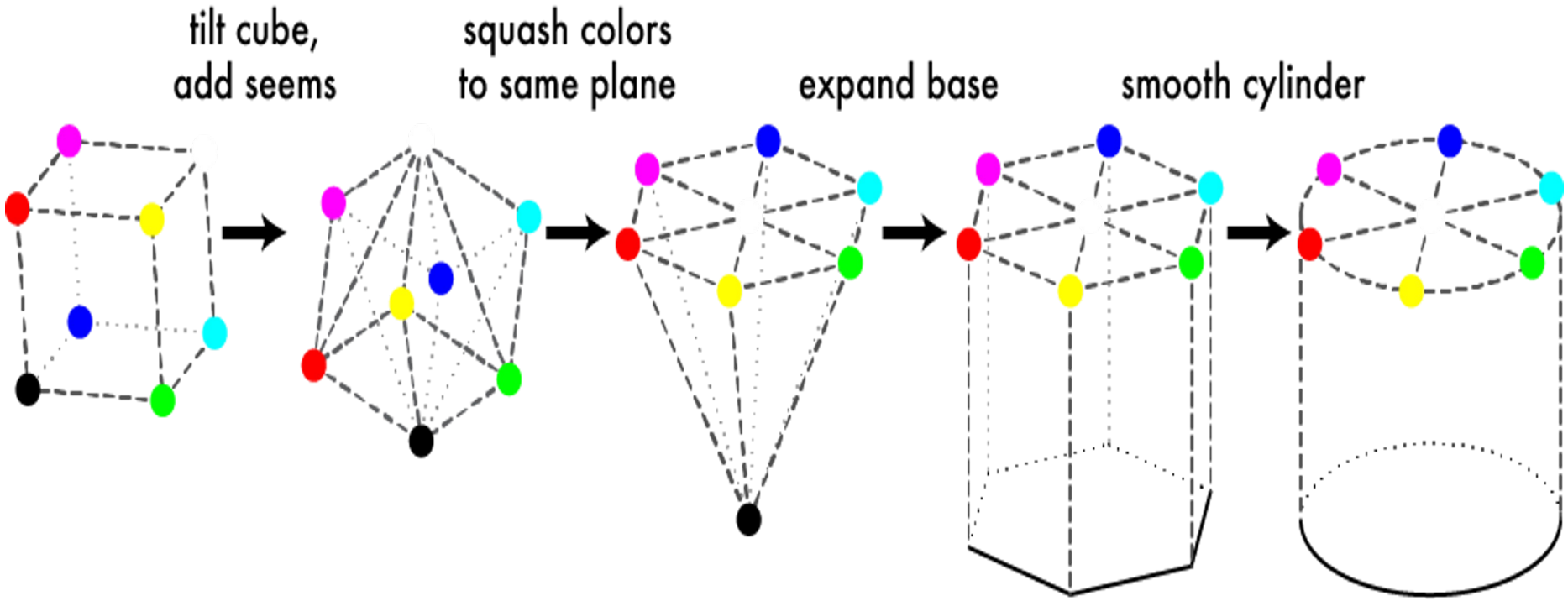
- Perceptually meaningful dimensions: Hue, Saturation, Value (Intensity)
- RGB cube on its vertex

RGB2HSV

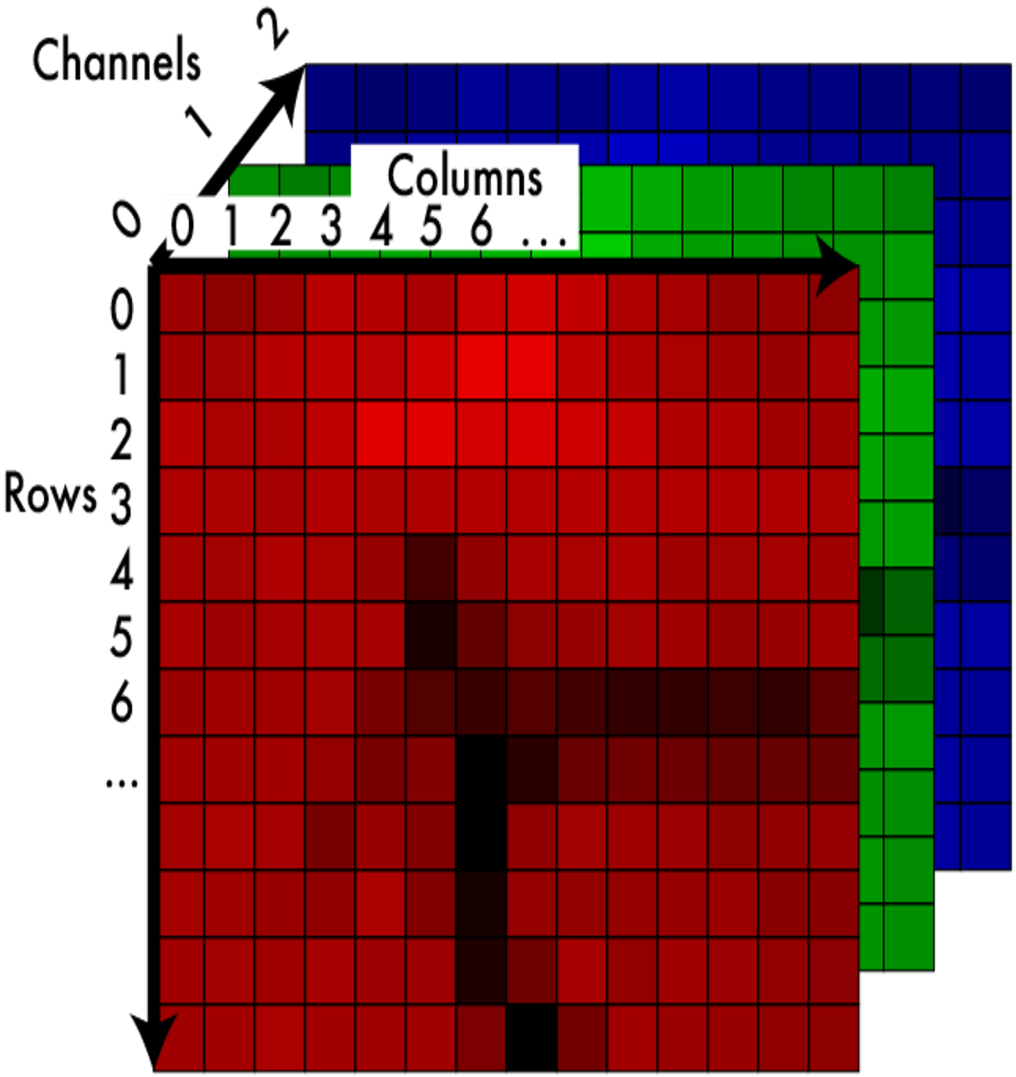


- Different model based on perception of light
- Hue: what color
- Saturation: how much color
- Value: how bright
- Allows easy image transforms: shift the hue, increase saturation

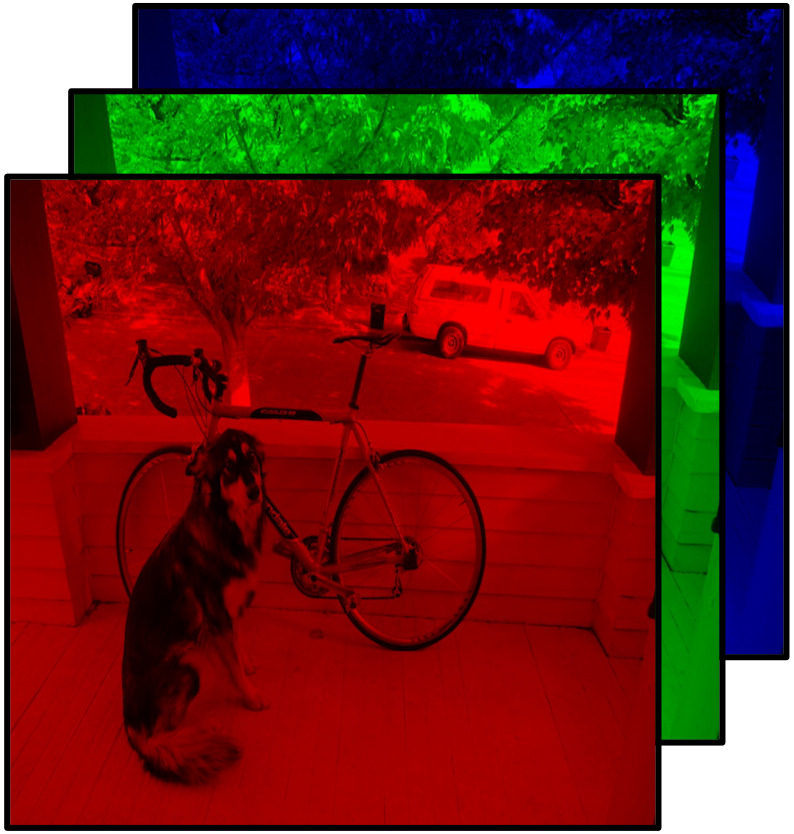
Geometric RGB to HSV

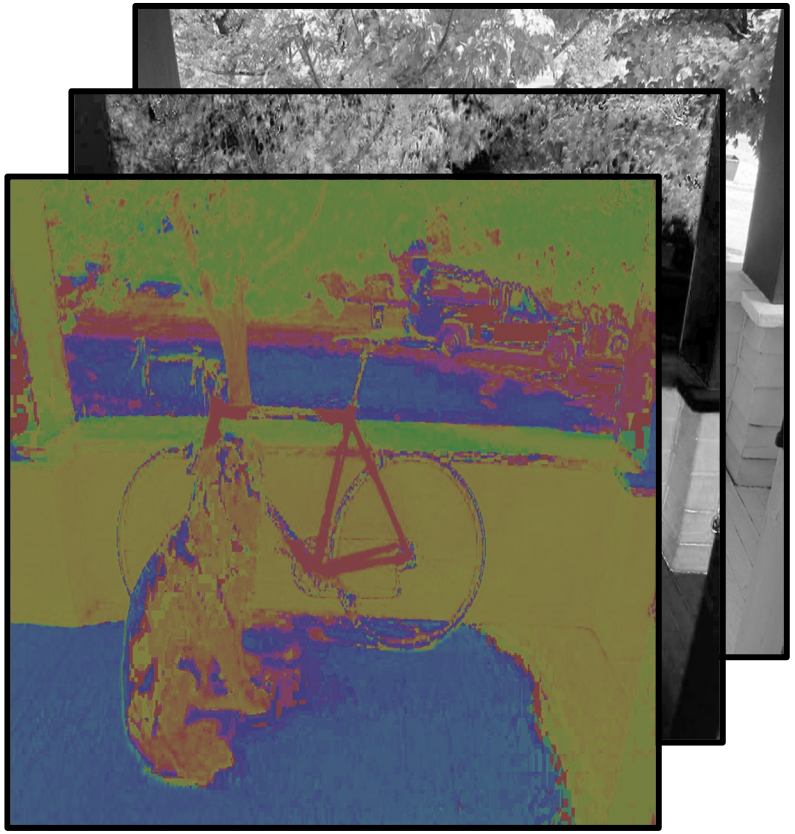


RGB Color image: 3d tensor in colorspace



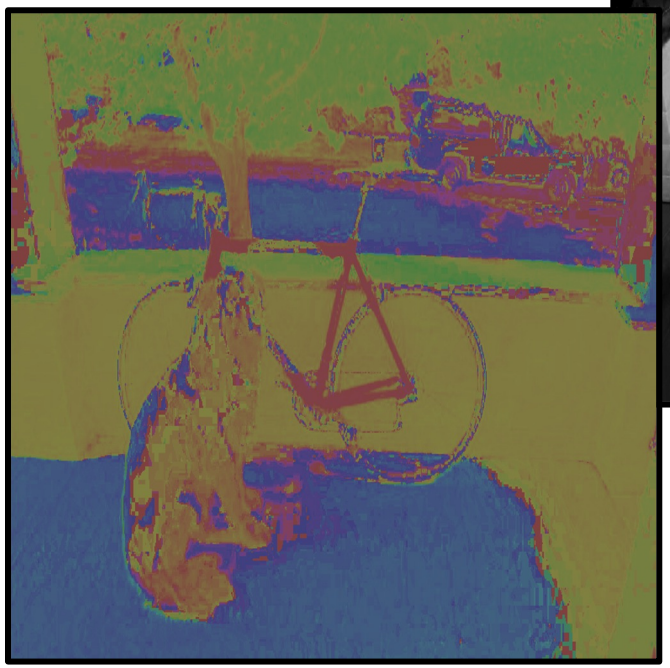
HSV Color image: 3d tensor, different info

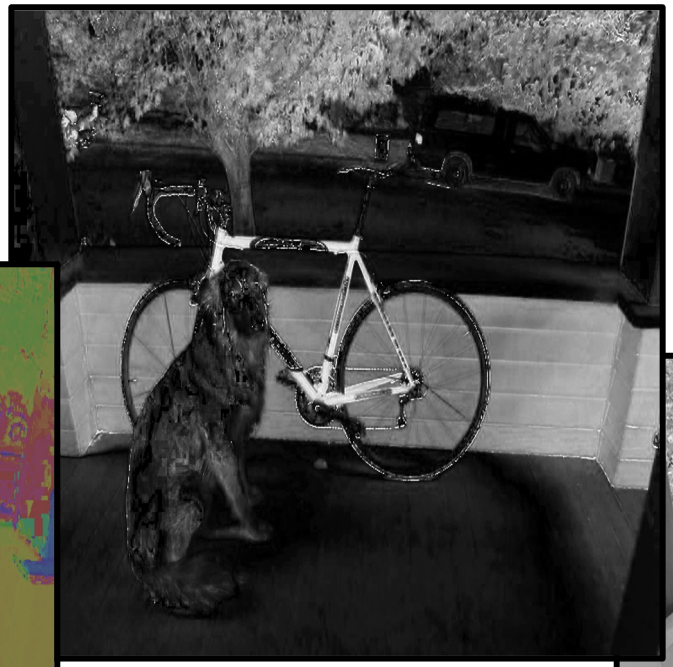




Saturation

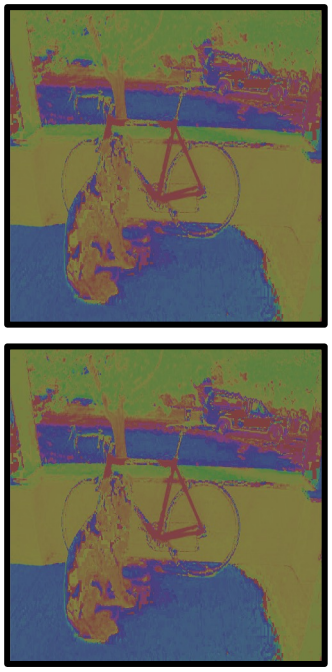
Hue





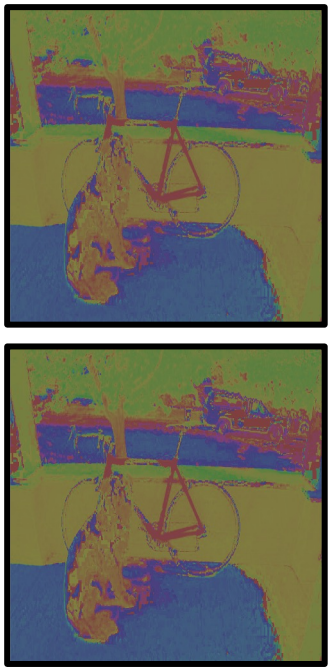
Value

More saturation = intense colors



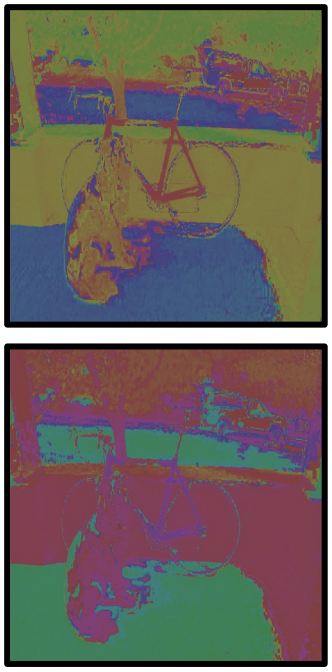
2x

More value = lighter image



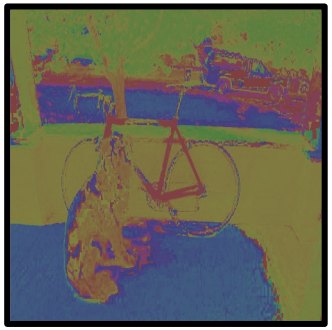
2x

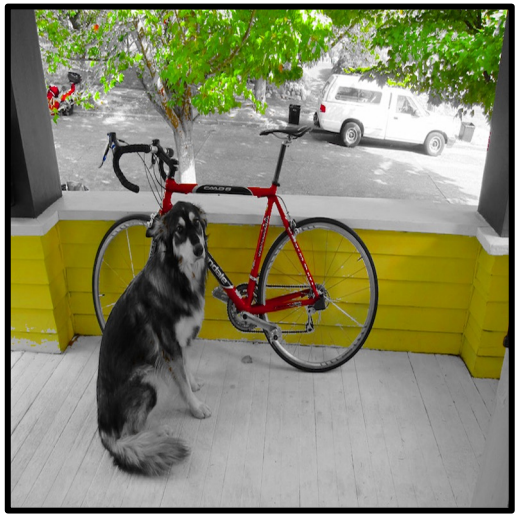
Shift hue = shift colors



-.2

Increase and threshold saturation





Template matching

Template matching - example

We want to localize in the image this template

template
one way traffic sign

- simplest method for object detection
- for every pixel in the image, compare the template with the window centered at that pixel: measure how well (in similarity or distance) the image window pattern resembles the template

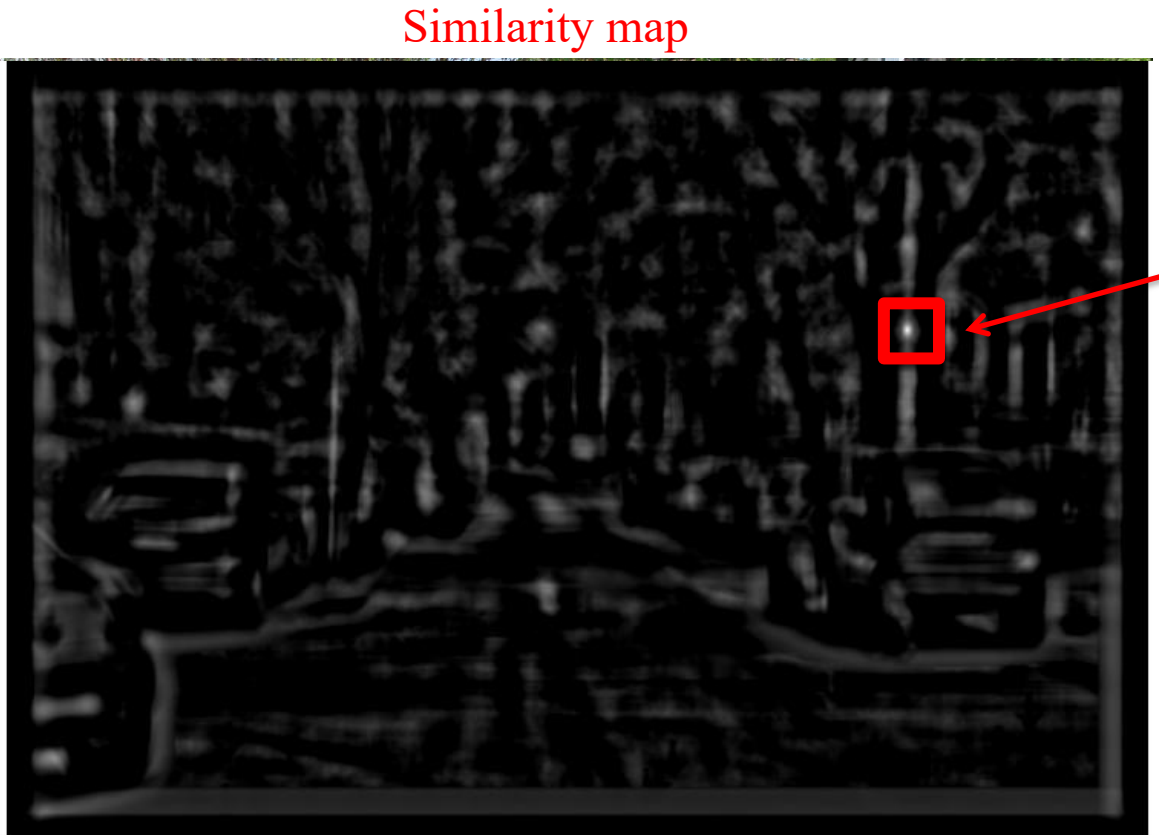
Template matching - example

We want to localize in the image this template

template

- simplest method for object detection
- for every pixel in the image, compare the template with the window centered at that pixel: measure how well (in similarity or distance) the image window pattern resembles the template

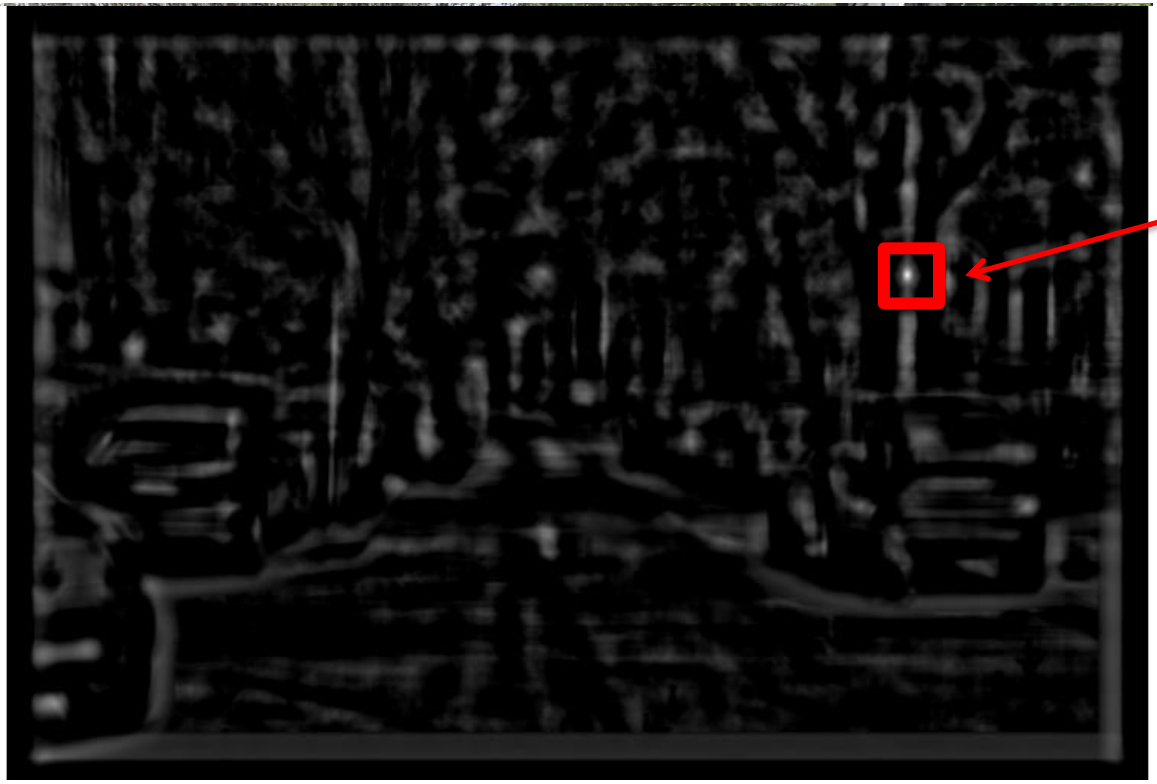
Template matching - example



- simplest method for object detection
- for every pixel in the image, compare the template with the window centered at that pixel: measure how well (in similarity or distance) the image window pattern resembles the template

Template matching - example

Similarity map



Highest value of the similarity map

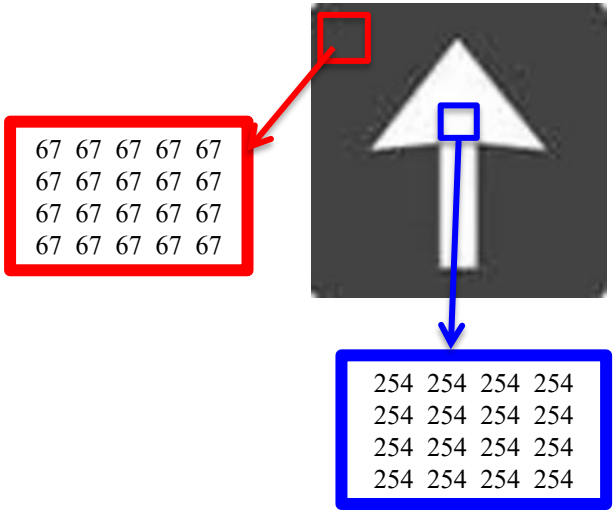
template

- similarity map = filtered image (template = filter)
- what is a good measure for similarity of two patches (windows)?
  - correlation?
  - sum of squared distances?
  - something else?

Template matching with filters

- template =

rgb2gray
cv.cvtColor



rgb2gray = cv.cvtColor

Slide credit Derek Hoiem

Template matching with filters

- filter f_1 = 
- correlation: we filter image with filter f_1

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_1(u,v) I(i+u, j+v)$$

I

$$O_1 = f_1 \otimes I$$

What is the result of applying the filter?

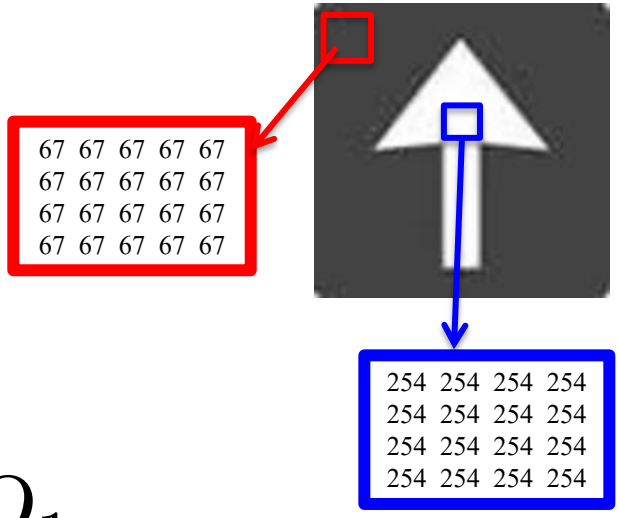
Template matching with filters

- filter f_1 = 
- correlation: we filter image with filter f_1

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_1(u,v) I(i+u, j+v)$$

Explanations?

Unnormalized filter (sum > 1) results in filtered image being much brighter



O_1

Filtered image = completely white

Template matching with filters

- filter f_2 =  normalized (sum = 1: 67→0.00007, 254→0.00003)
- correlation: filter image with filter f_2

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_2(u,v) I(i+u, j+v)$$

I

$$O_2 = f_2 \otimes I$$

What is the result of applying the filter?

Template matching with filters

- filter f_2 =  normalized (sum = 1: 67→0.00007, 254→0.00003)
- correlation: filter image with filter f_2

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_2(u,v) I(i+u, j+v)$$

O_2

Explanations?

The filter's response is large for high intensity values in the image. We obtain the maximum response for a white patch in the image.

Template matching with filters

- filter f_3 =  normalized (sum = 0, $f_3 = f_2 - \overline{f_2}$)
- correlation: filter image with filter f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$

I

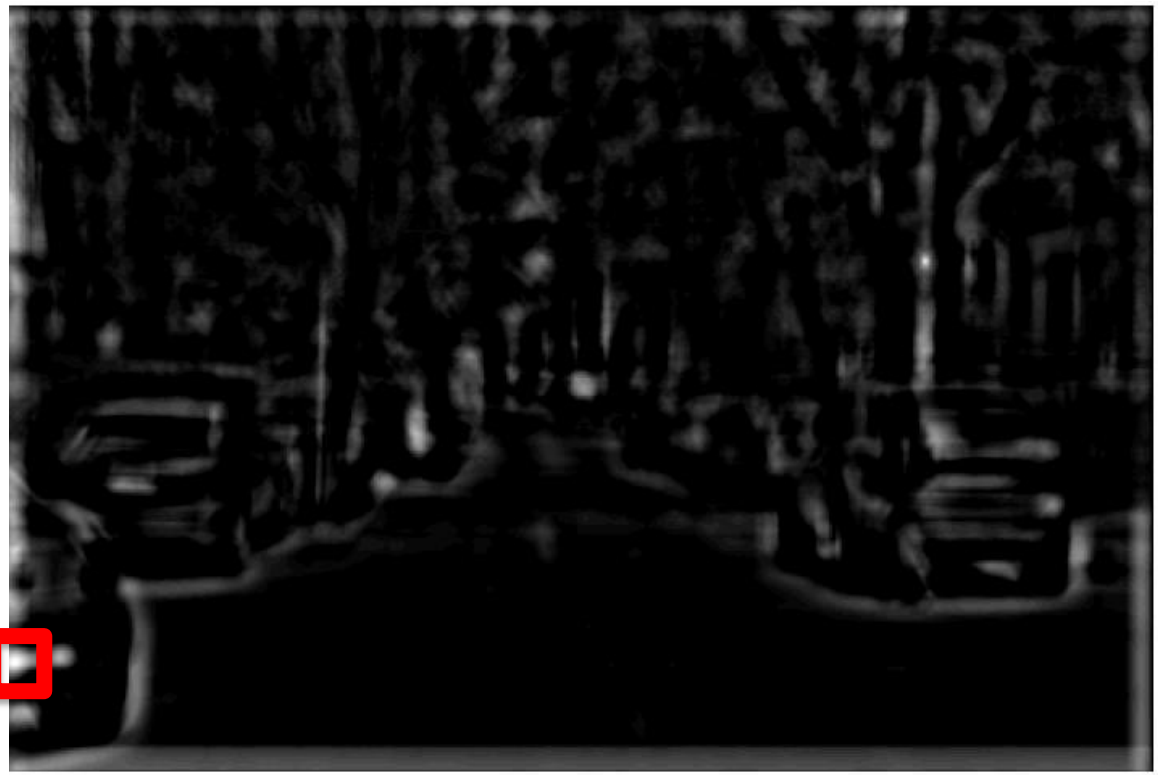
$$O_3 = f_3 \otimes I$$

What is the result of applying the filter?

Template matching with filters

- filter f_3 =  normalized (sum = 0, $f_3 = f_2 - \overline{f_2}$)
- correlation: filter image with filter f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

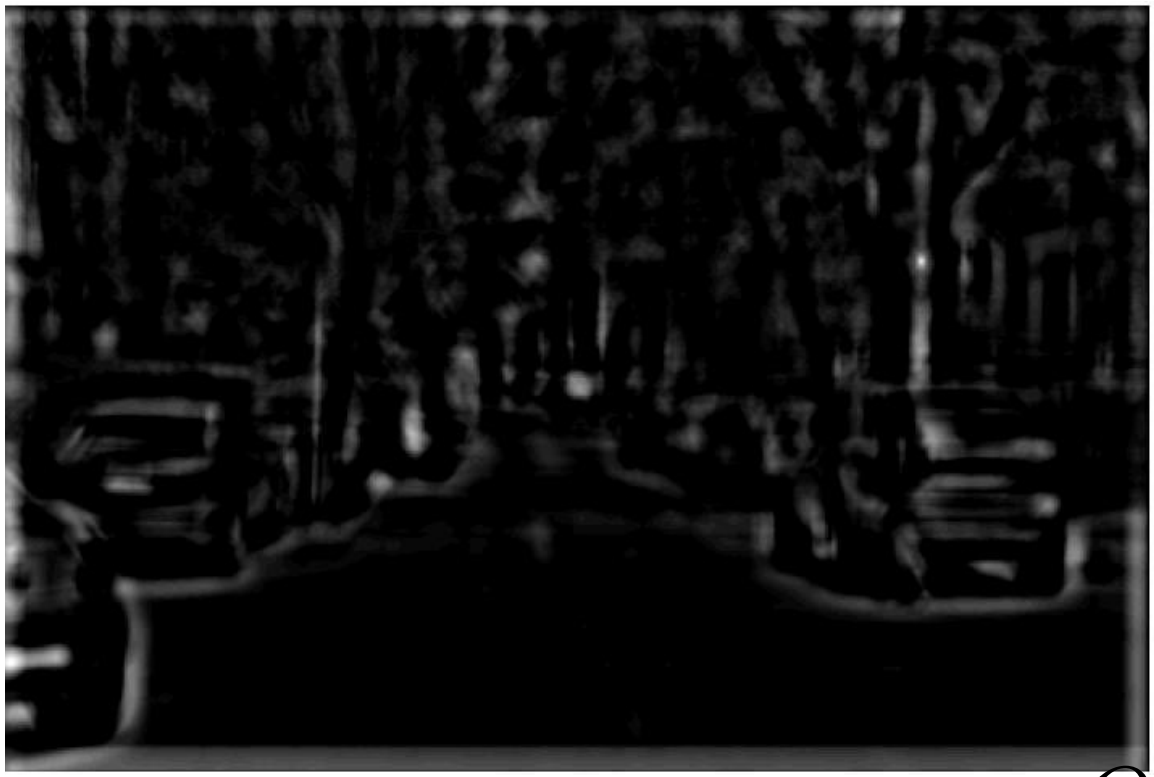
Explanations?

Response is large when:
-small values (negative numbers) from the filter correlate with small pixel intensities in the image
-large values (positive numbers) from the filter correlate with large pixel intensities in the image

Template matching with filters

- filter f_3 = ⬆ normalized (sum = 0, $f_3 = f_2 - \overline{f_2}$)
- correlation: filter image with filter f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

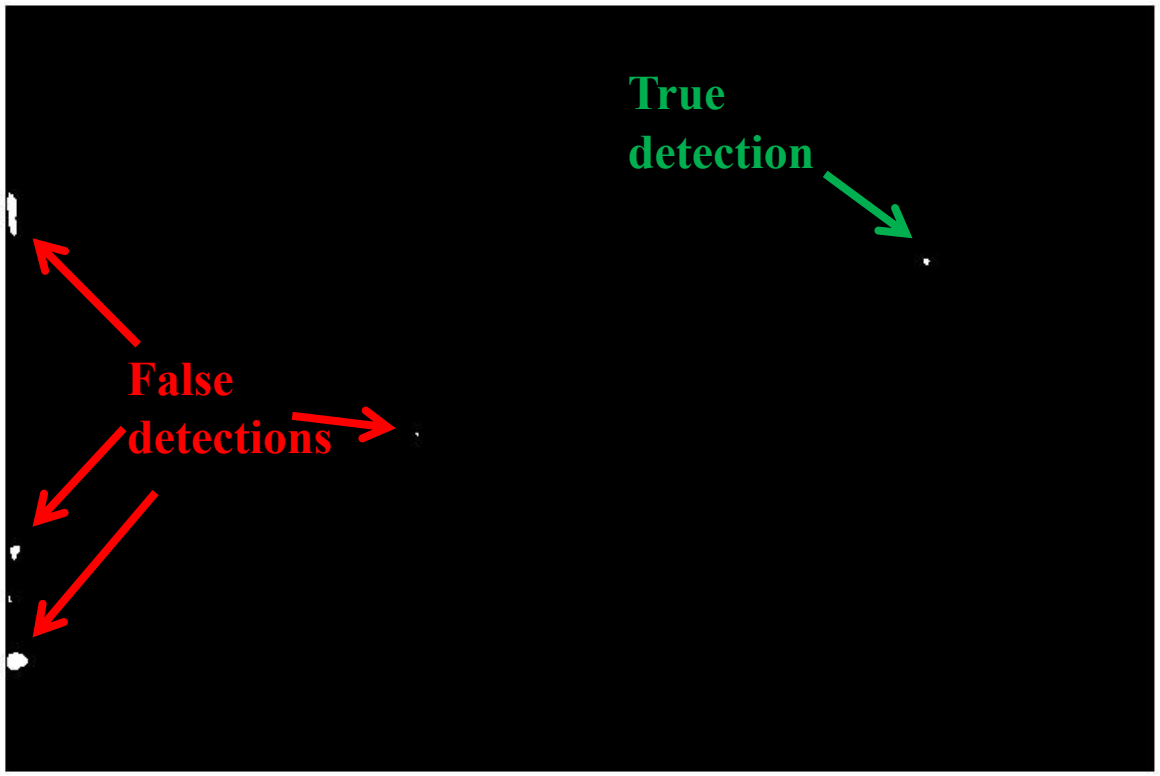
Explanations?

Response is maximum when:
-small values (negative numbers) from the filter correlate with black pixels
-large values (positive numbers) from the filter correlate with white pixels
-does not require pixel values in image to be near or proportional to values in filter.

Template matching with filters

- filter f_3 =  normalized (sum = 0, $f_3 = f_2 - \overline{f_2}$)
- correlation: filter image with filter f_3

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} f_3(u,v) I(i+u, j+v)$$



O_3

Thresholding

Apply a threshold to the filtered image: find all pixels with intensity > threshold

Template matching with filters

- filter $f_1 =$ 
- sum of squared distances

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

I

Template matching with filters

- filter $f_1 =$ 
- sum of squared distances

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

Find the minimum value of distance / maximum value of similarity and this should correspond to the template.

O_4 (1 – distance)

Template matching with filters

- filter $f_1 =$ 
- sum of squared distances

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

Shadow effects in the image will affect the template.

What to we obtain in this case?

Template matching with filters

- filter $f_1 =$ 
- sum of squared distances

$$o(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{l} (I(i+u, j+v) - f_1(u,v))^2$$

Explanations?

The filter's response is sensitive to the mean intensity in the window

O_5 (1 – distance)

Template matching with filters

- filter f_1 = 
- normalized correlation (of mean 0)

res = cv.matchTemplate(img,template,method)

mean intensity in the current window in the image

mean intensity of the filter

$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})(f_1 - \overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1 - \overline{f_1})^2}}$$

standard deviation of intensities in the current window in the image

standard deviation of intensities in the filter

Cosine of two normalized vectors

Template matching with filters

- filter f_1 = 
- normalized correlation

res = cv.matchTemplate(img,template,method)

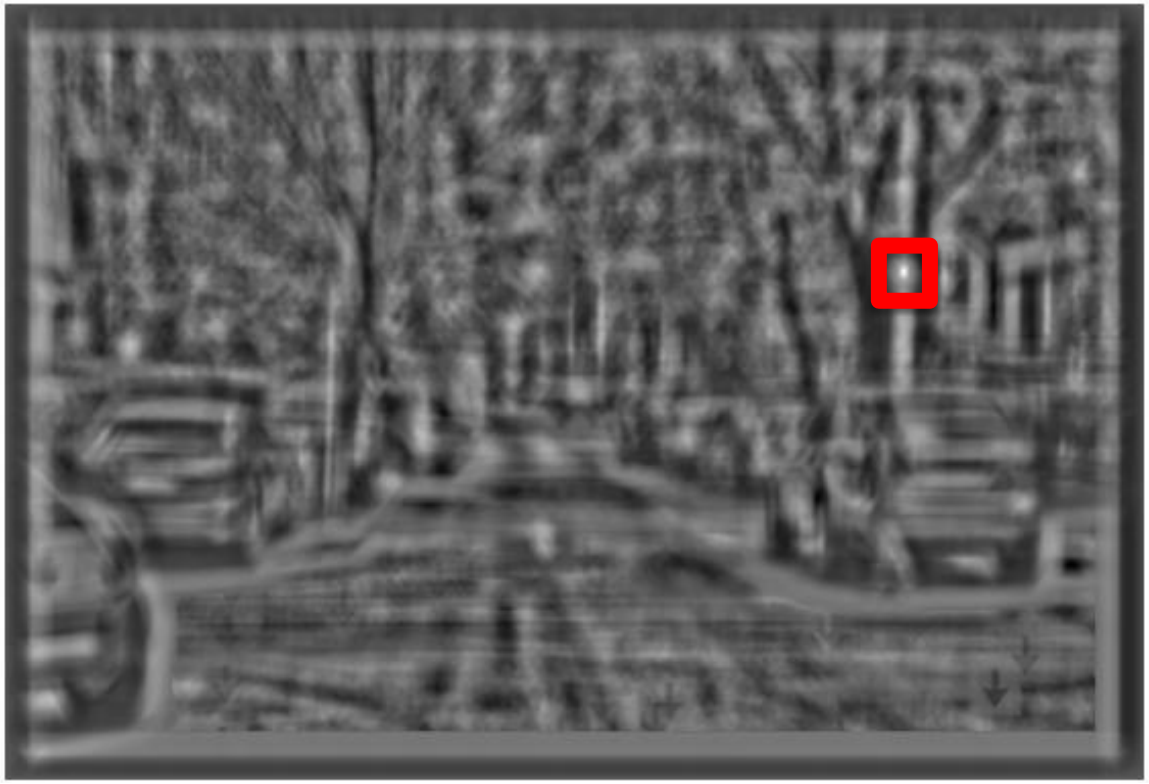
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})(f_1 - \overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1 - \overline{f_1})^2}}$$

Template matching with filters

- filter f_1 = 
- normalized correlation

res = cv.matchTemplate(img,template,method)

$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})(f_1 - \overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1 - \overline{f_1})^2}}$$



finding the minimum value = detect the template

Template matching with filters

- filter f_1 =
- normalized correlation

res = cv.matchTemplate(img,template,method)

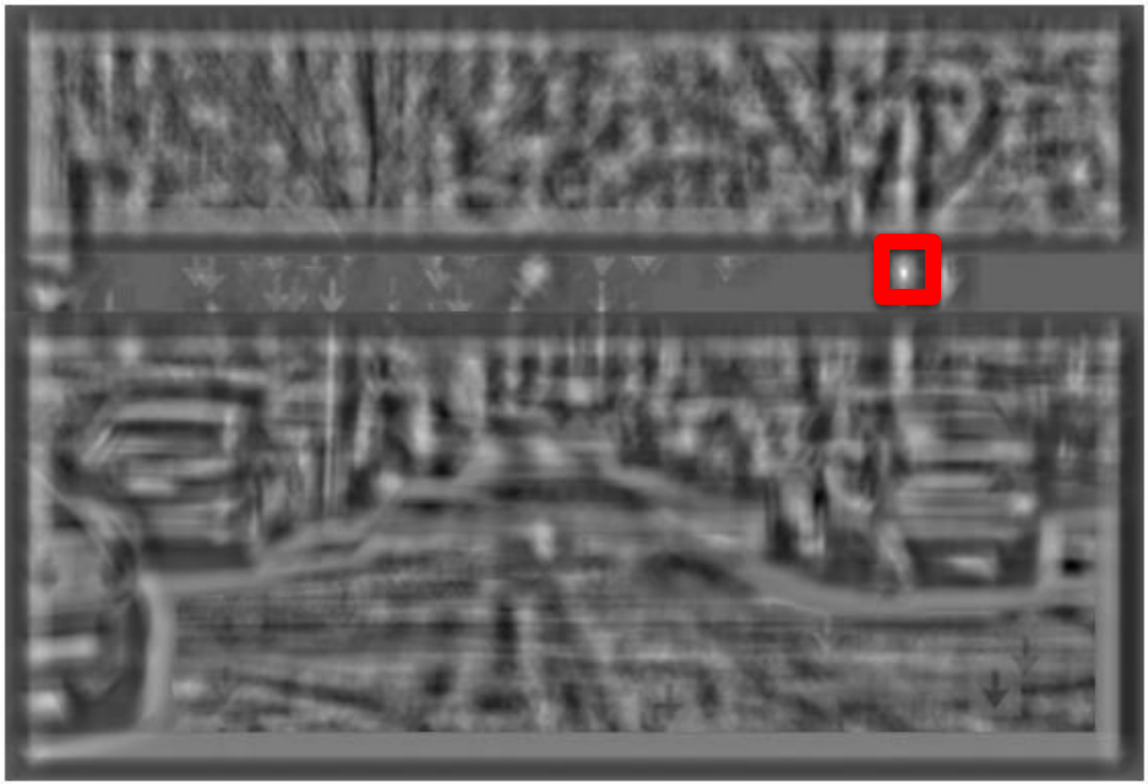
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})(f_1 - \overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1 - \overline{f_1})^2}}$$

Template matching with filters

- filter f_1 = 
- normalized correlation

res = cv.matchTemplate(img,template,method)

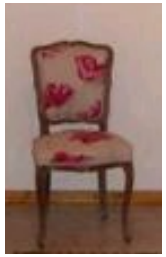
$$o(i,j) = \frac{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})(f_1 - \overline{f_1})}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(I(i+u,j+v) - \overline{I}_{i,j})^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-l}^{l}(f_1 - \overline{f_1})^2}}$$



finding the minimum value = detect the template

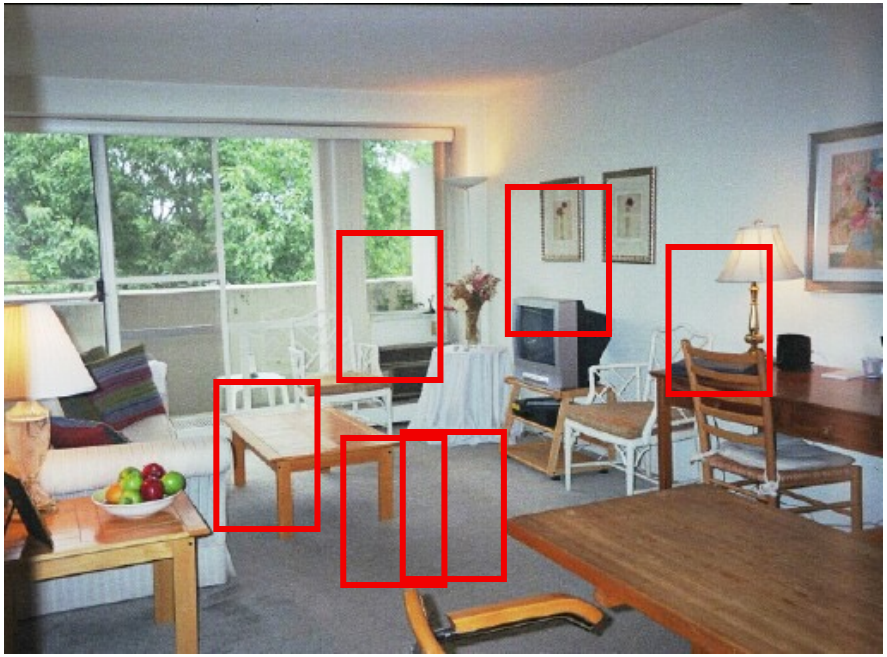
Best measure of similarity?

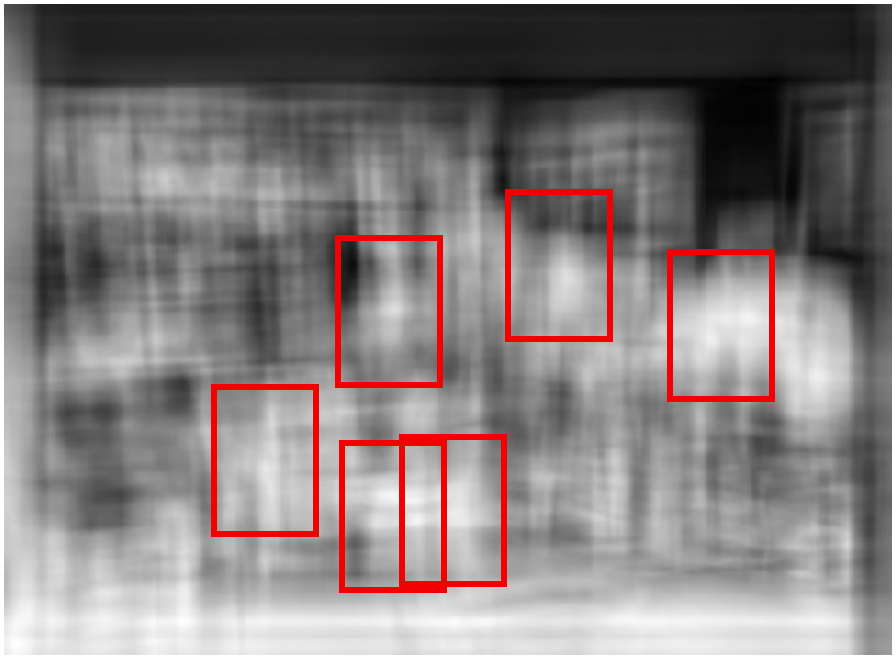
- correlation with filter of mean 0: results are not so good (false detections)
- sum of squared distances: sensitive to mean intensity
- normalized correlation: invariant to mean intensity and contrast



Object detection with template matching

Detect chairs in the image





Template matching doesn't solve the problem

A "popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts." Nevatia & Binford, 1977.

Object detection with template matching

Pros:

- simple method, easy to implement
- good results also in cases when the template/ something very similar to the template is in the image (some invariance to occlusions)

Cons:

- not invariant to scale, rotation (of the same template): if we resize the template or rotate it we obtain different results
- for object classes with large variability in appearance (cars, persons, faces, etc) we cannot apply template matching
- should work fine for logo detection, etc

Finding smaller or larger template?

initial template 

smaller template  alternative → build a pyramid with resized images

Course structure

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

   Linear filters, color, texture, edge detection, **template matching**

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

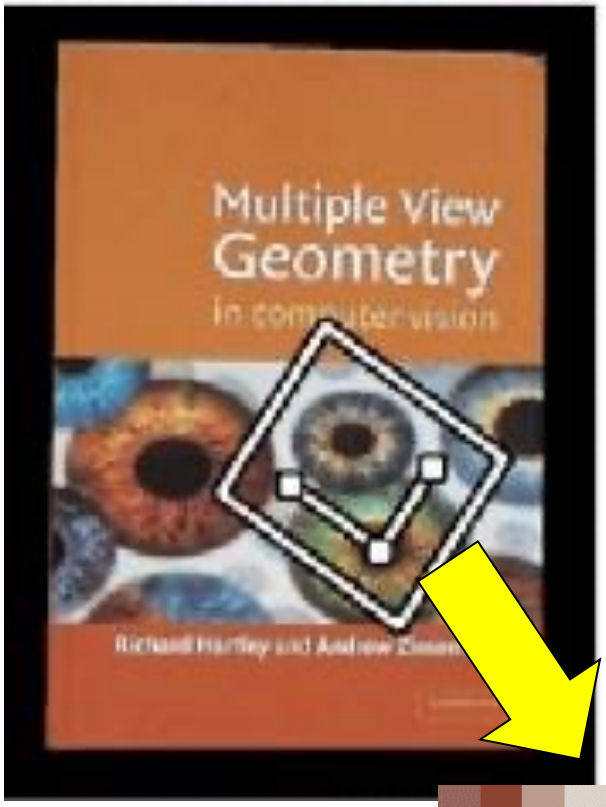
Local features

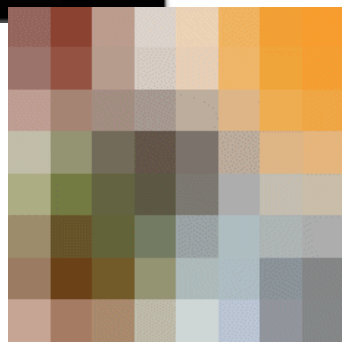
- local feature = interest point = keypoint + size + descriptor
- local image feature = a region in the image that
  - has a rich image content (brightness variation, color variation)
  - has a well defined representation for matching other similar features
  - has a well-defined position in the image
  - is invariant to image scaling, rotation and change in illumination. It is geometrically (translation, rotation, ...) and photometrically (brightness, exposure, ...) invariant.

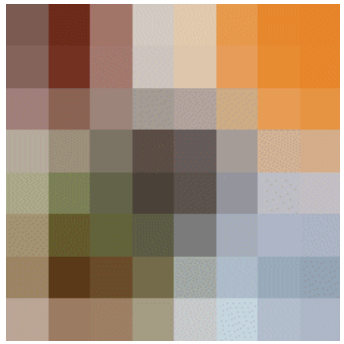
Local features

- local feature = interest points = keypoint + size + descriptor
- local image feature = a region in the image that
  - has a rich image content (brightness variation, color variation)
  - has a well defined representation for matching other similar features
  - has a well-defined position in the image
  - is invariant to image scaling, rotation and change in illumination. It is geometrically (translation, rotation, ...) and photometrically (brightness, exposure, ...) invariant.

Geometric transformations





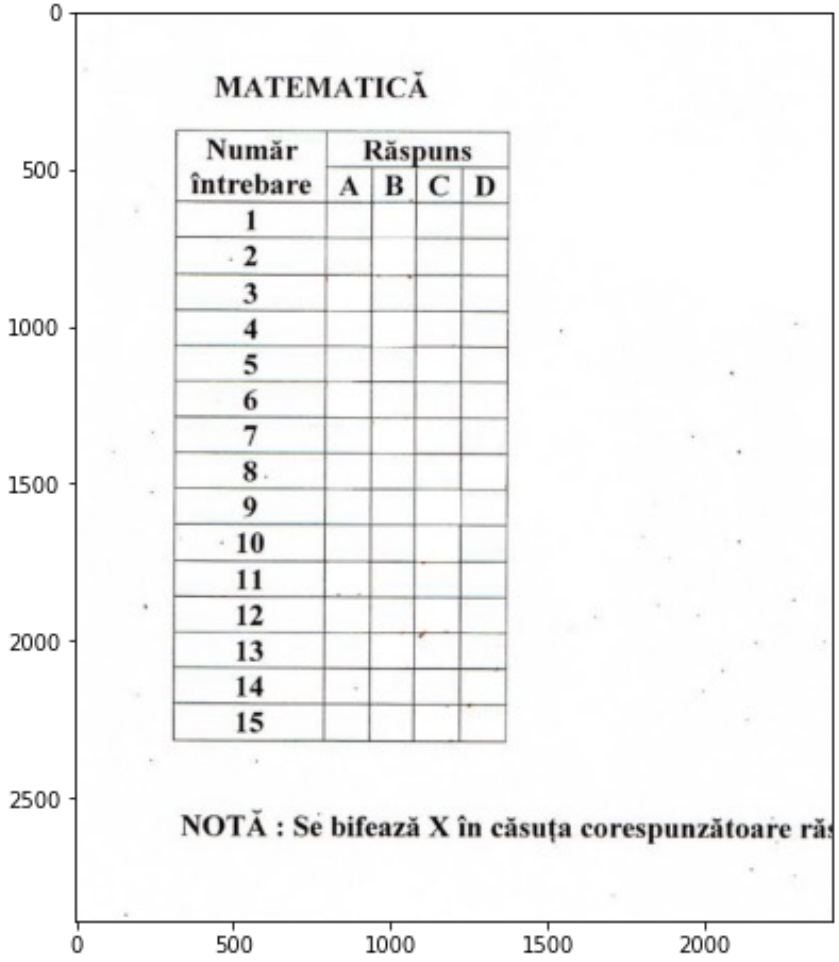


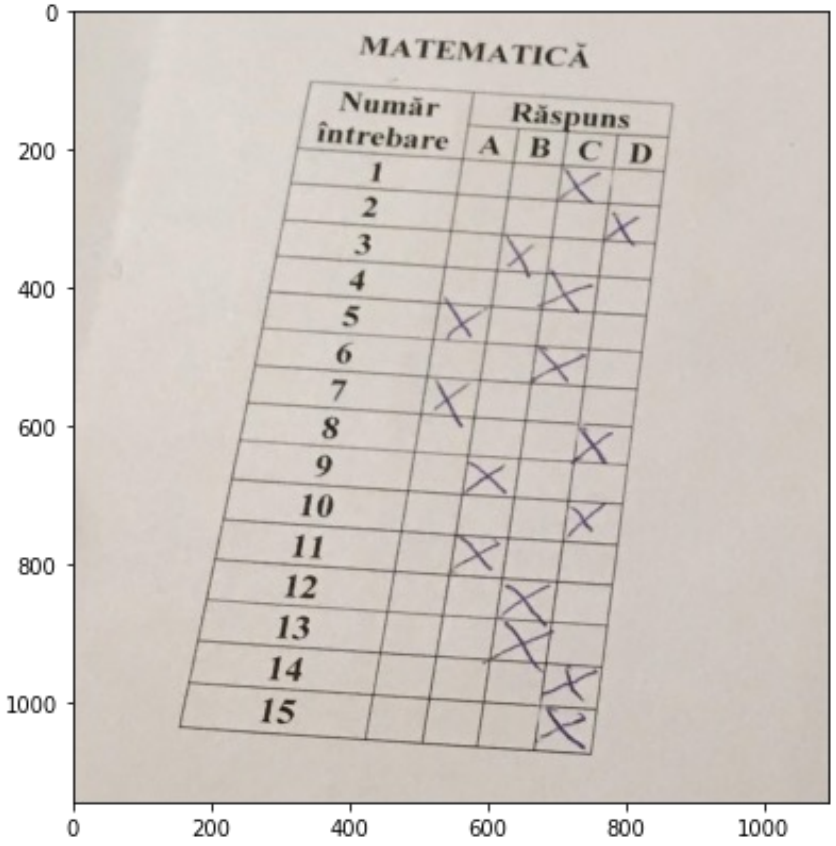
Exemples:
translation,
rotation,
scaling

Photometric transformations

Matching local features – past P1

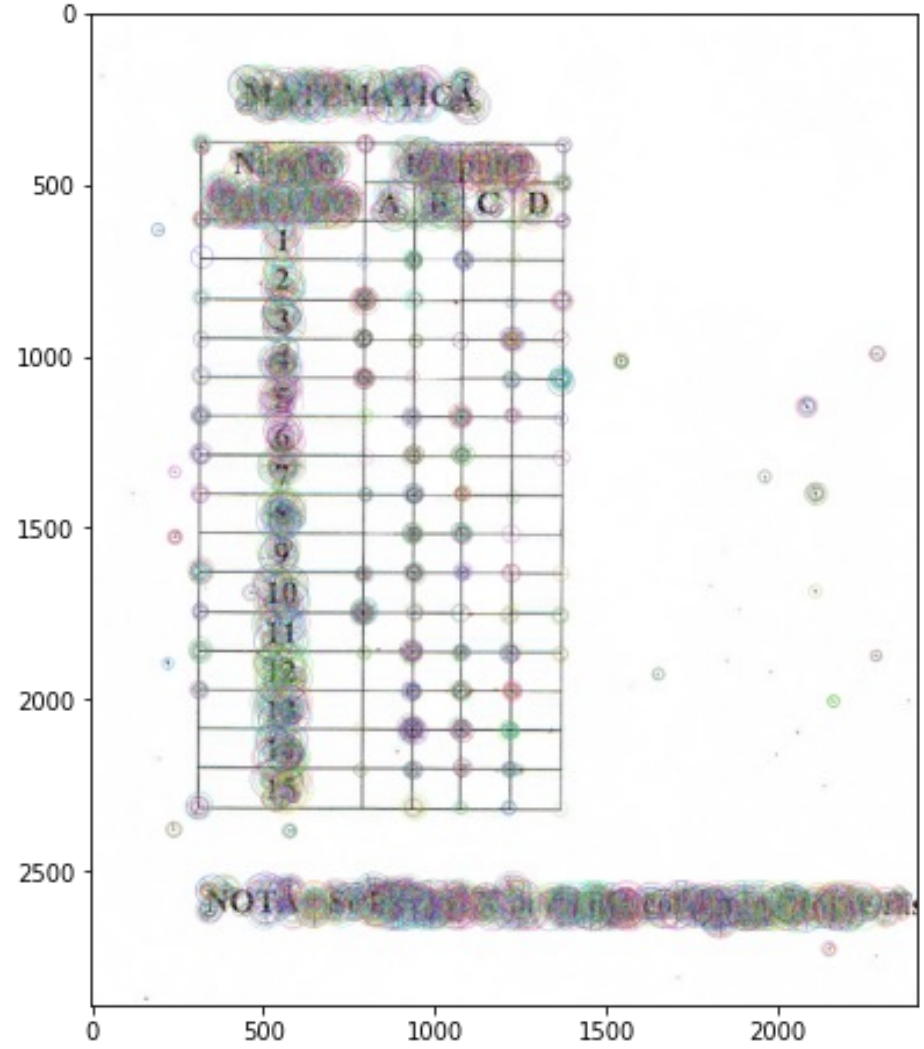
- matching local image feature = find correspondence points between images based on local features that look the same

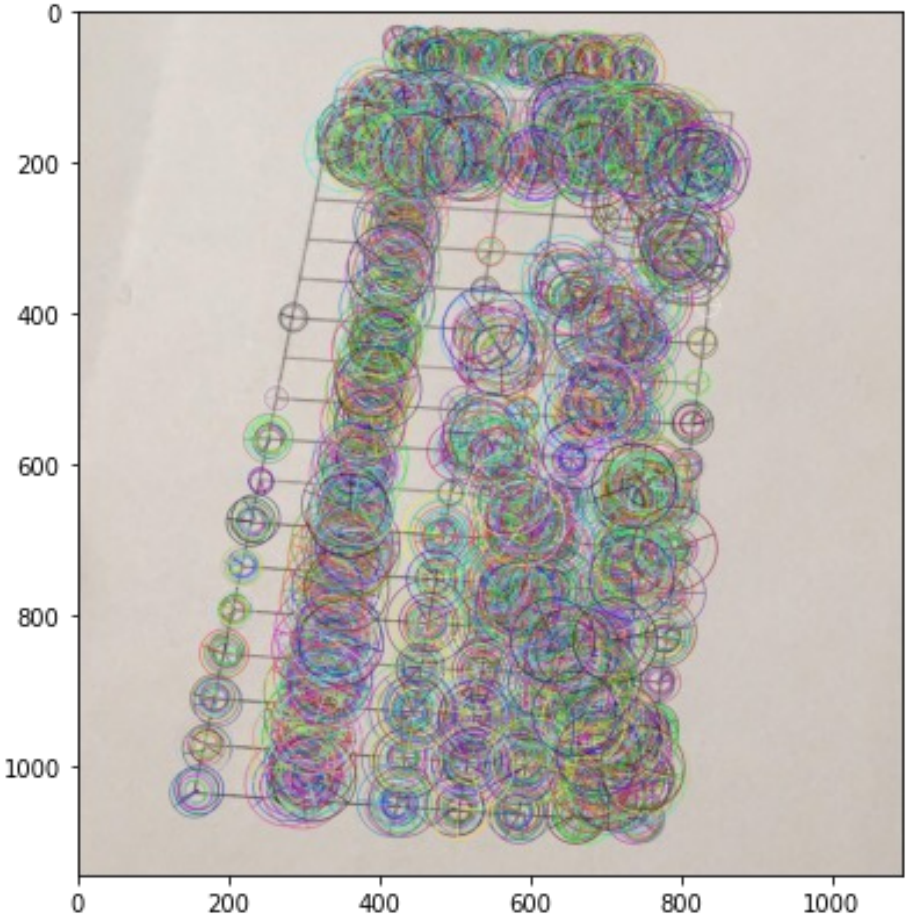




Matching local features – past P1

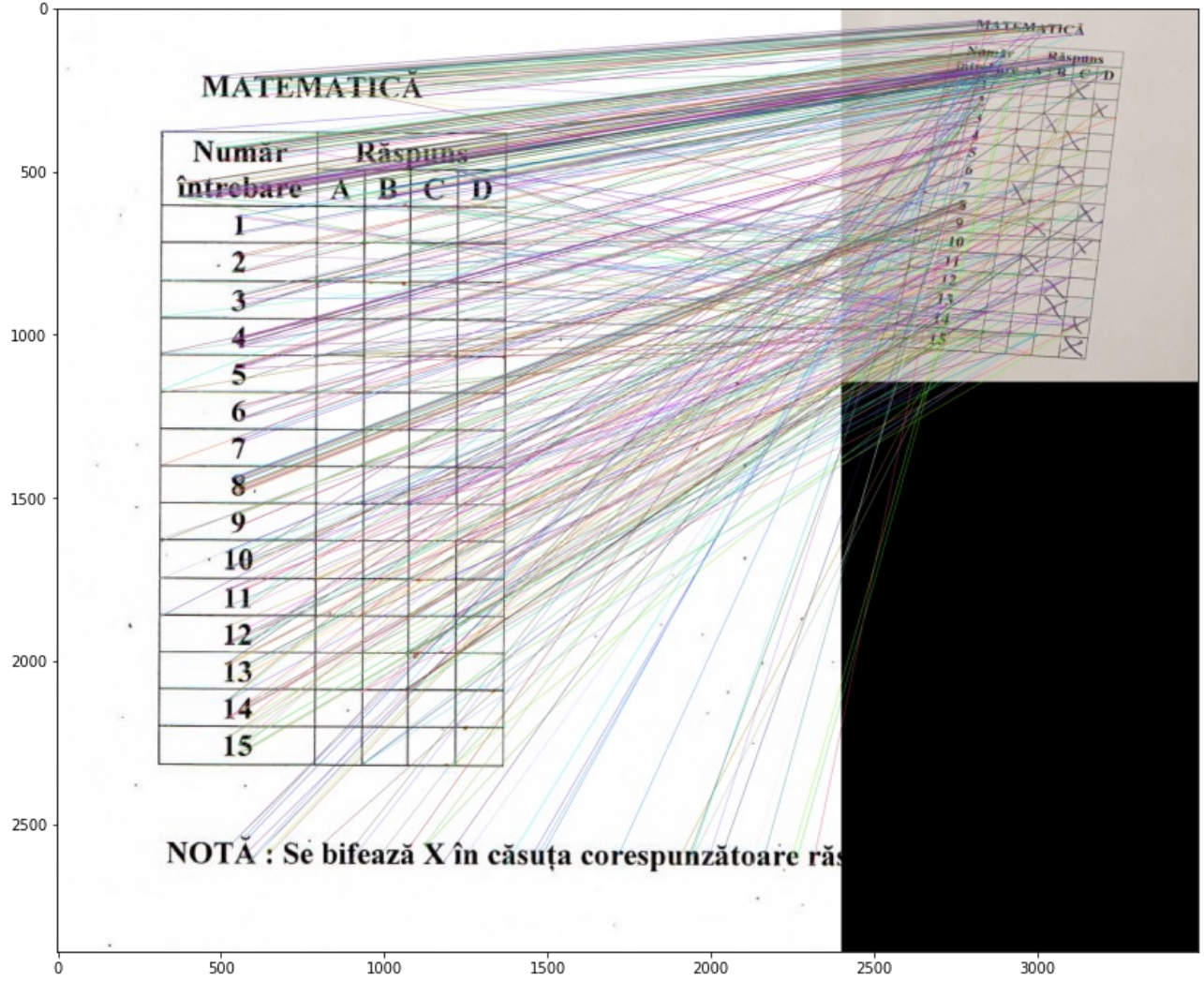
- matching local image feature = find correspondence points between images based on local features that look the same





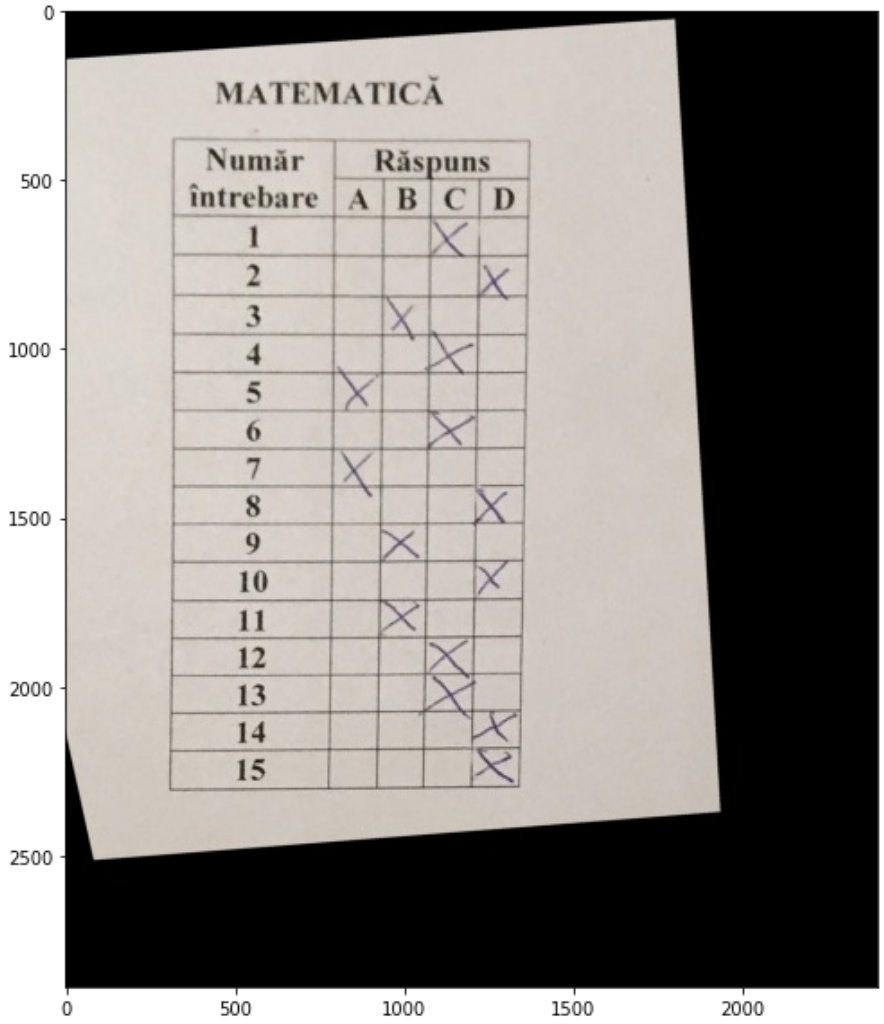
Matching local features – past P1

- matching local image feature = find correspondence points between images based on local features that look the same



Finding the homography transformation

- Find the matrix of the homography (perspective) transformation and use it to warp one image

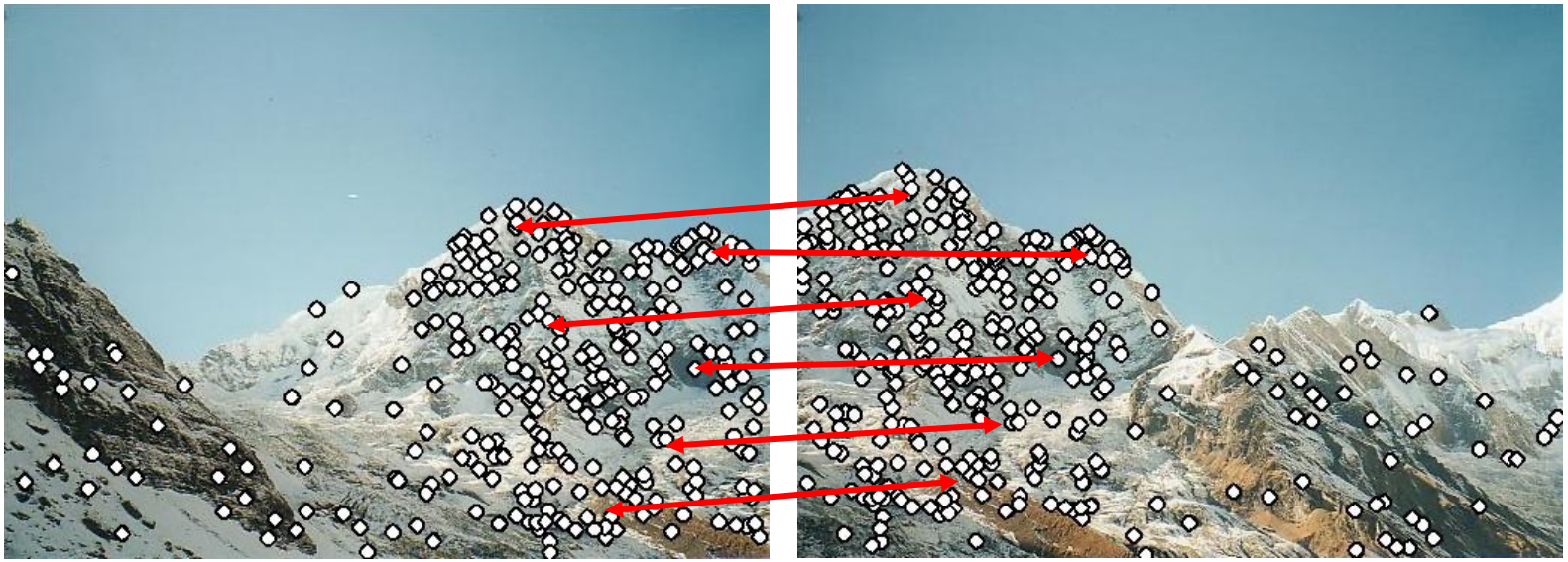


Why extract local features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?


Why extract local features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract local features

Step 2: match local features

Why extract local features?

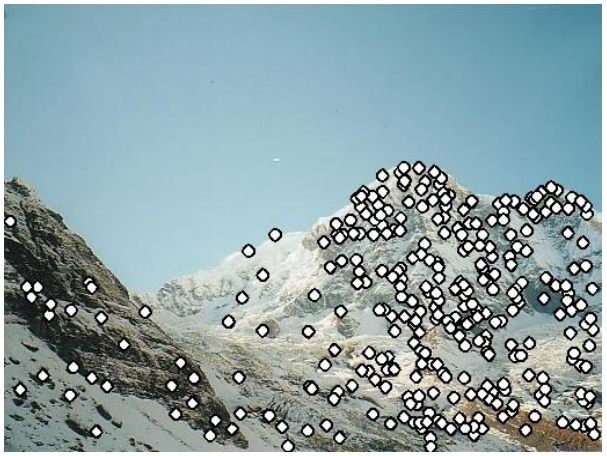
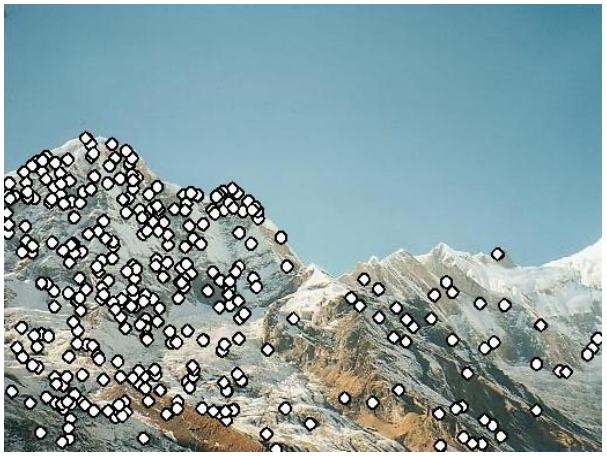
- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract local features

Step 2: match local features

Step 3: align images

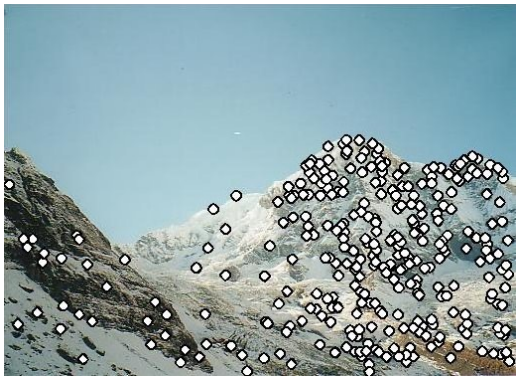
Characteristics of good local features

- Compactness and efficiency
  - Many fewer features than image pixels
- Saliency
  - Each local feature is distinctive
- Locality
  - A local feature occupies a relatively small area of the image; robust to clutter and occlusion
- Repeatability
  - The same local feature can be found in several images despite geometric and photometric transformations

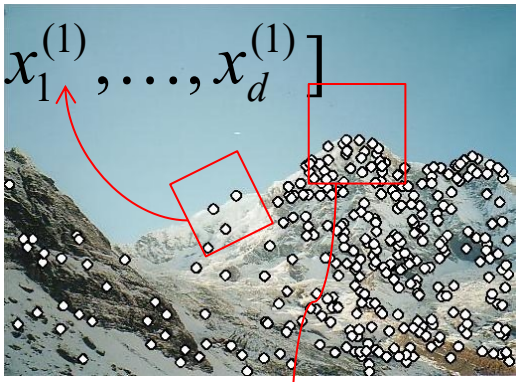
Local features: main components

1) **Detection**: Identify the interest points



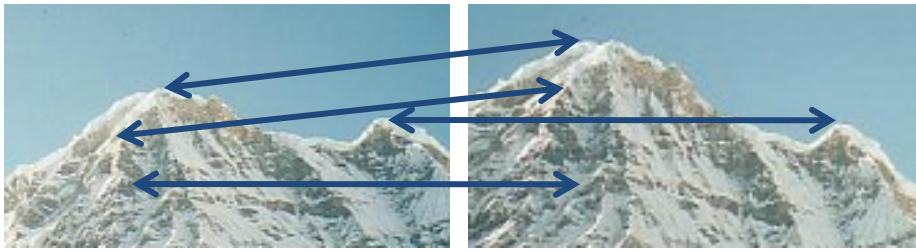
2) **Description**: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



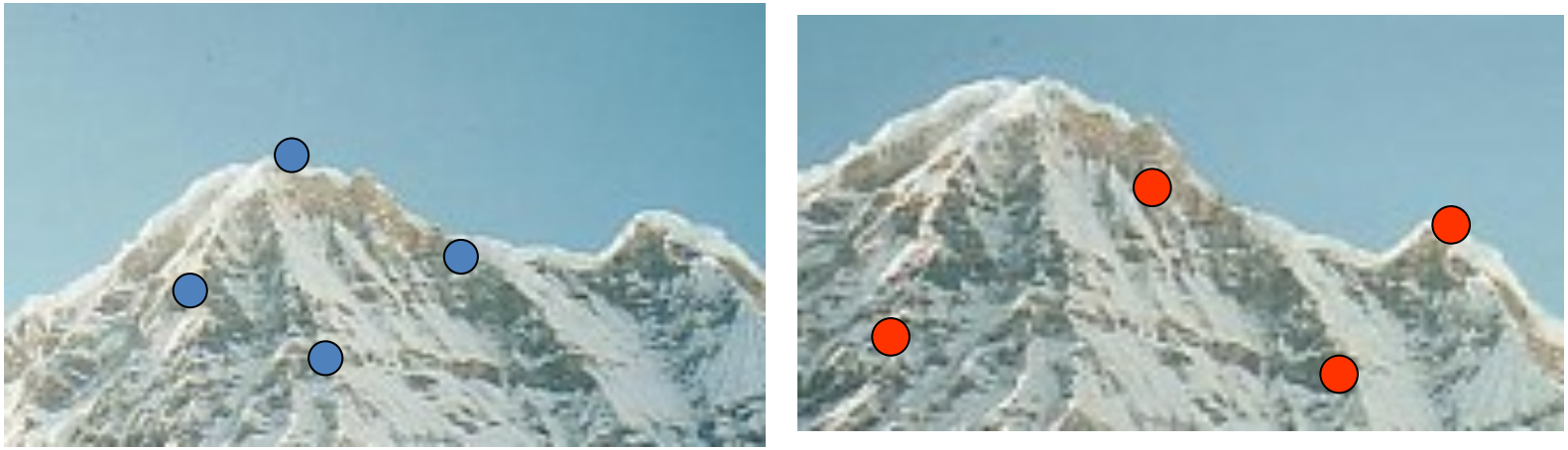
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching**: Determine correspondence between descriptors in two views



Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

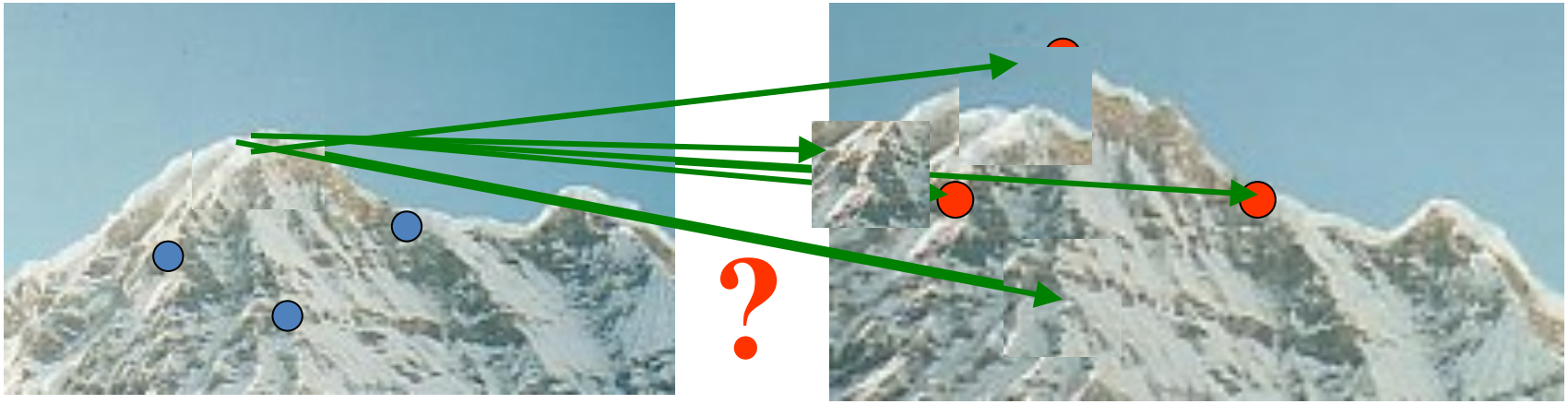


No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.



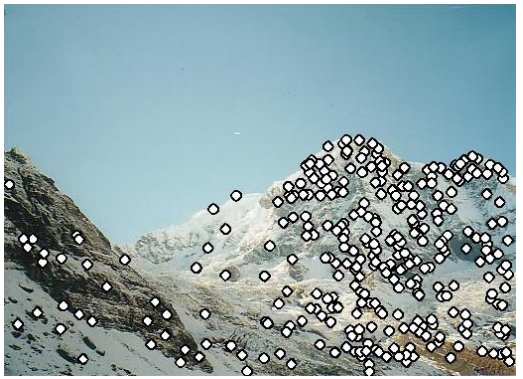
- Must provide some invariance to geometric and photometric differences between the two views.

Applications

- Local features are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Database indexing and retrieval
  - Object recognition (before Deep Learning)

Local features: main components

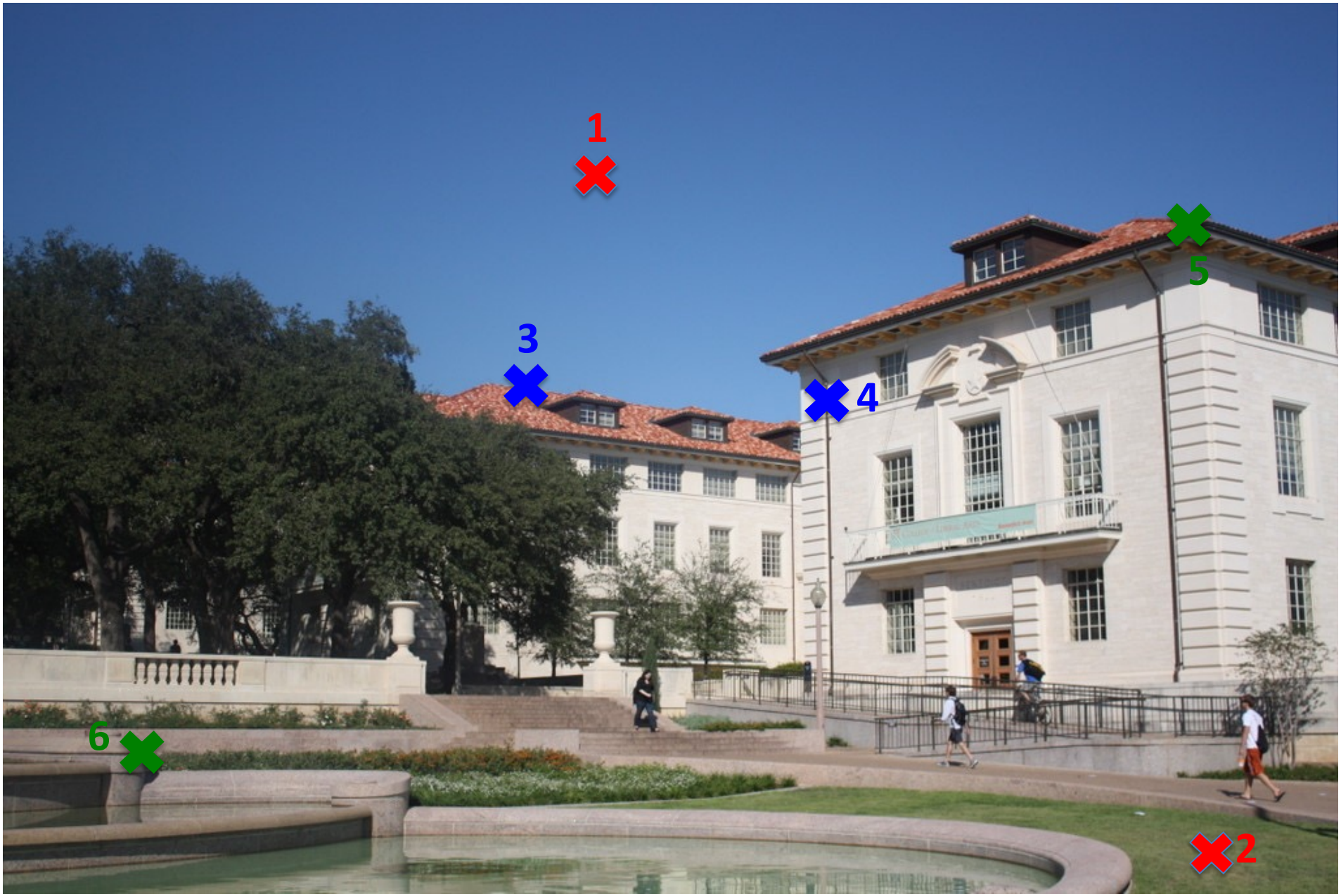
1) Detection: Identify the interest points



2) Description:Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

- What points would you choose?


1
2
3
4
5
6

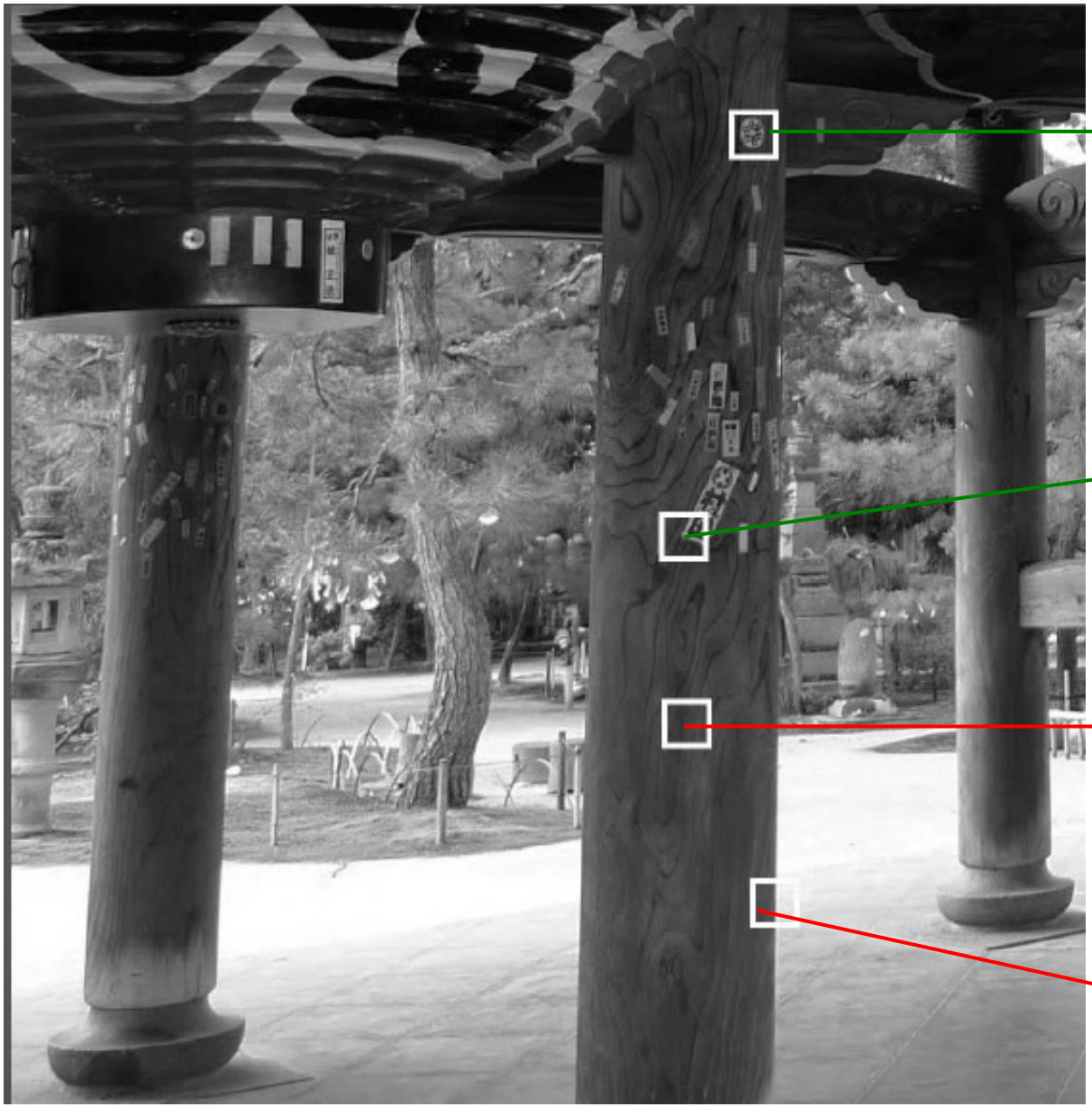
Detecting local invariant features

- Detection of interest points

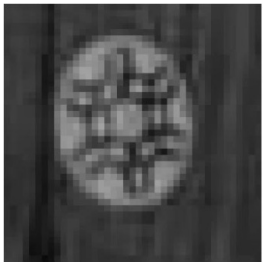
Uniform regions
Ambiguous for matching

Edges
Ambiguous for matching

Corners
Good pentru matching

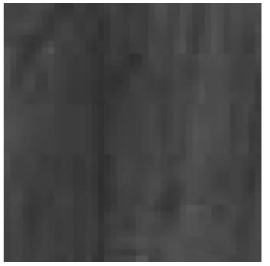


circular region (blob)

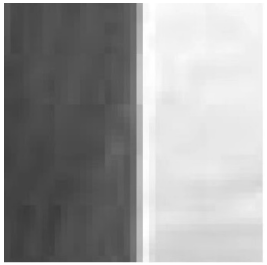


corner

uniform region



edge

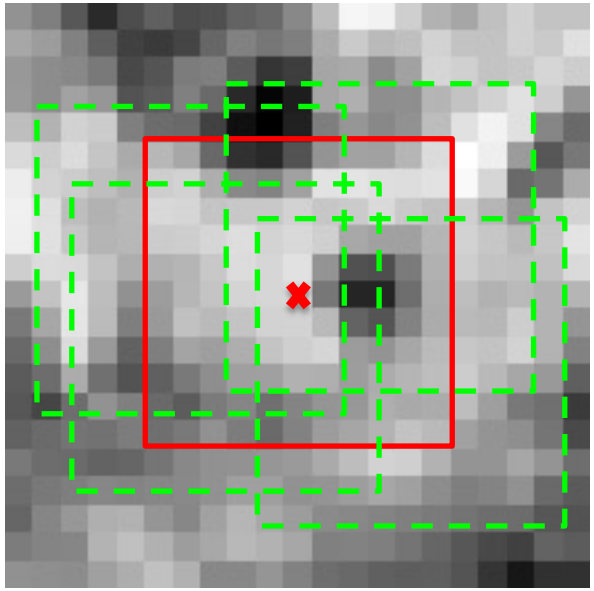


Detecting local invariant features

- Detection of interest points
  - Harris corner detection
  - Scale invariant blob detection: LoG
- Description of local patches

Detecting local invariant features

- How do we mathematically formulate what is a good local feature?

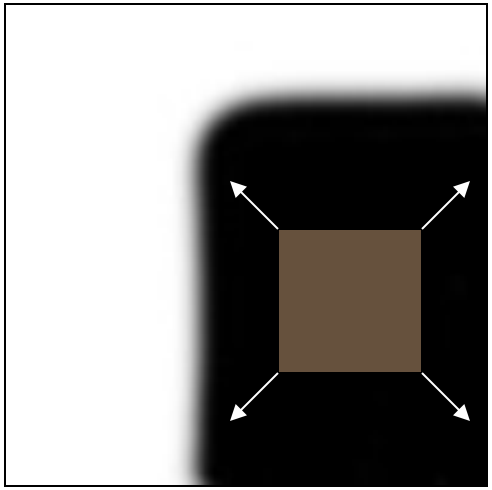


- Characterize the point by a small window centered in it

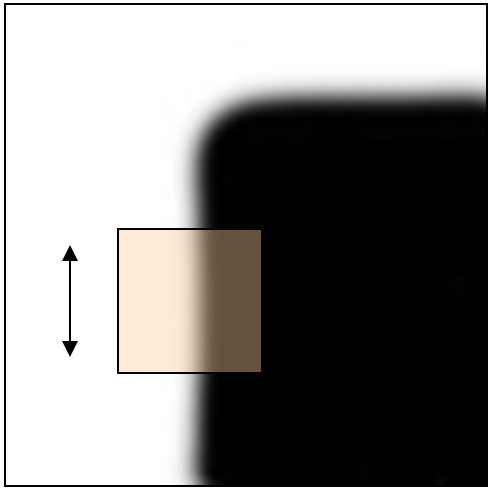
- Good local feature: shifting a window in *any direction* should give *a large change* in intensity (sum of squared distances)

Corner detection: Basic idea

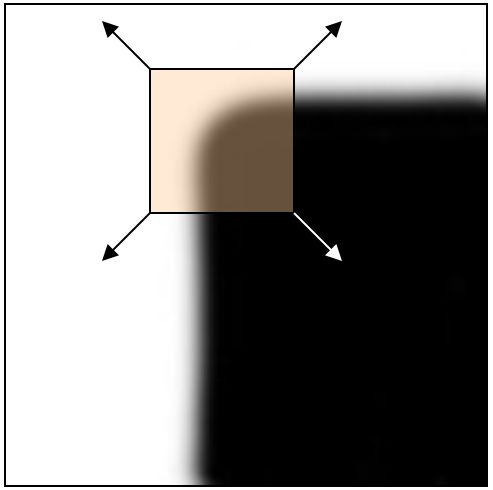
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions

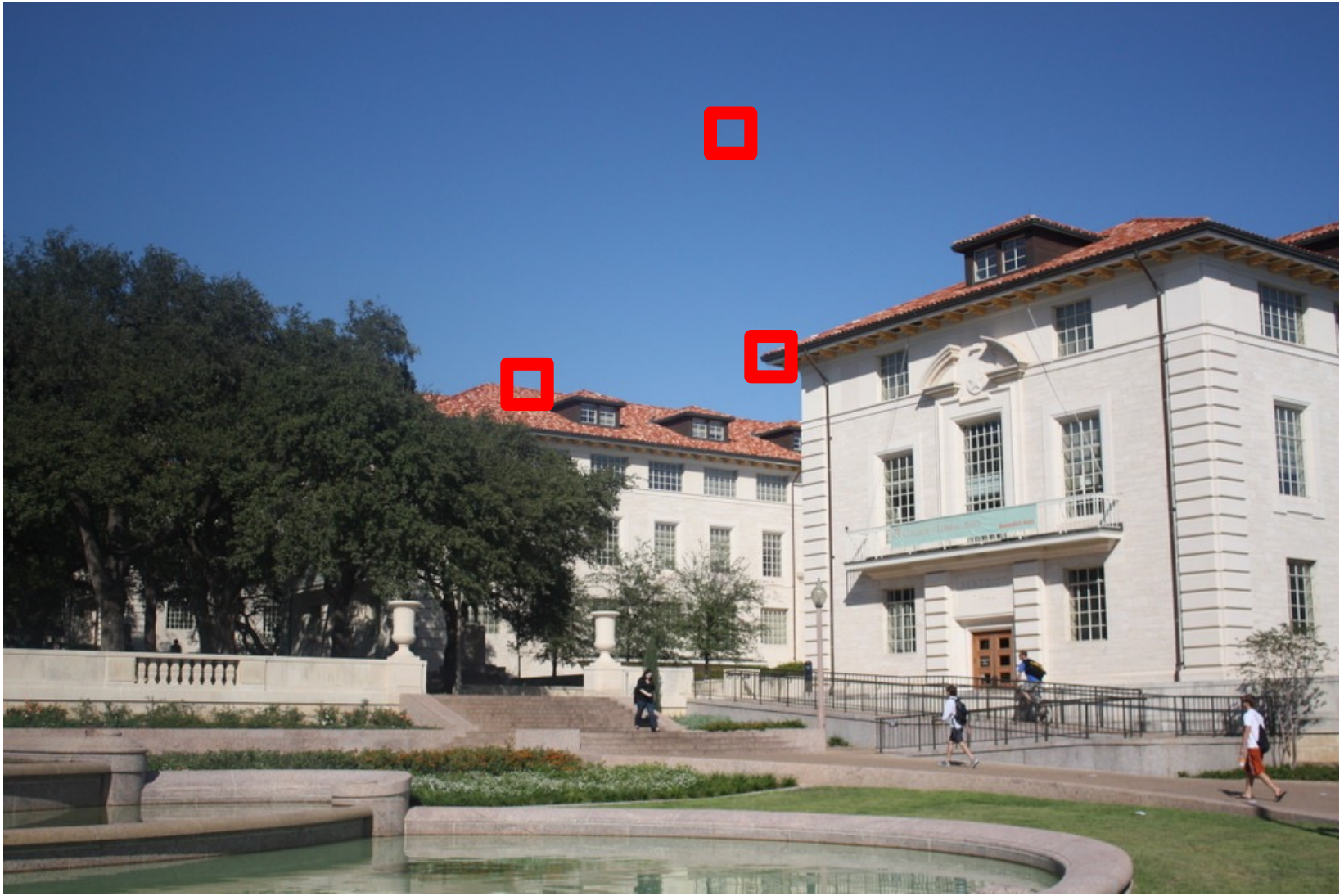


"edge":
no change along the edge direction



"corner":
significant change in all directions

Self-difference

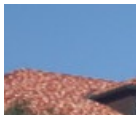


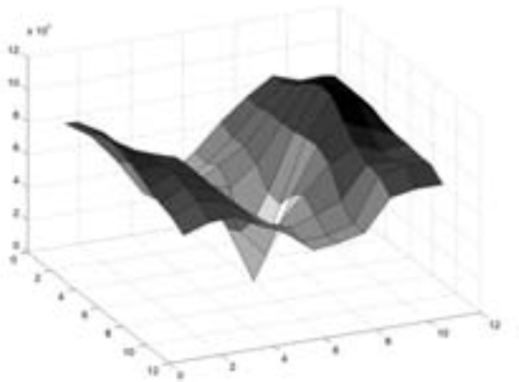
Self-difference

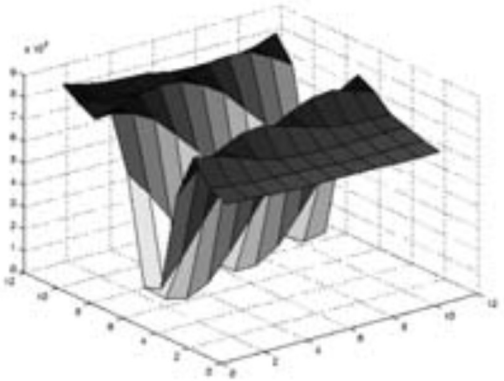
Sky: low everywhere

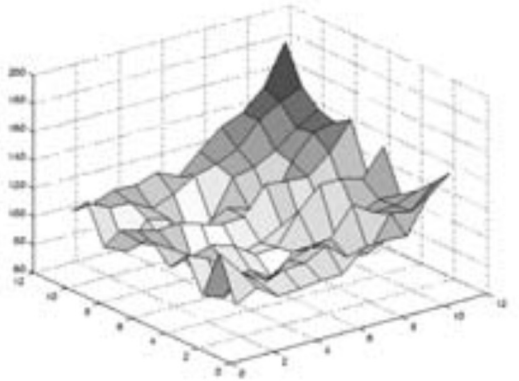
Edge: low along edge

Corner: mostly high







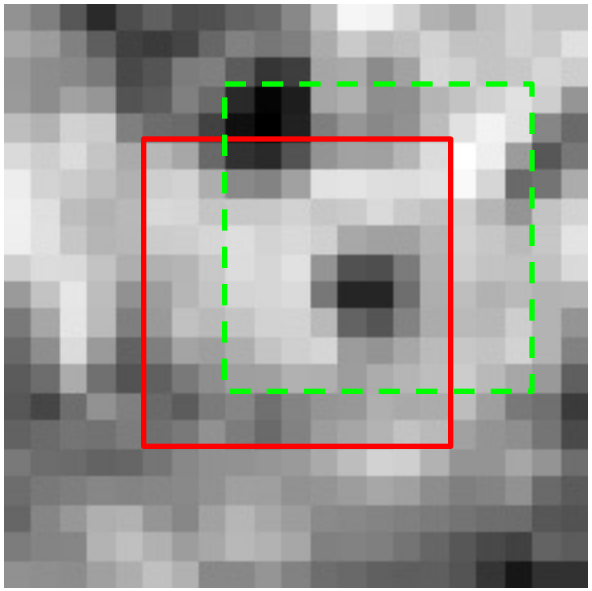


Corner Detection: Derivation

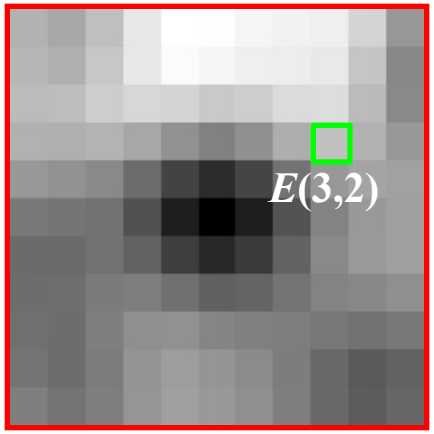
Change in appearance of window W for the shift $[u,v]$:

$$E(u,v) = \sum_{(x,y)\in W} (I(x+u,y+v) - I(x,y))^2$$

$I(x, y)$



$E(u, v)$

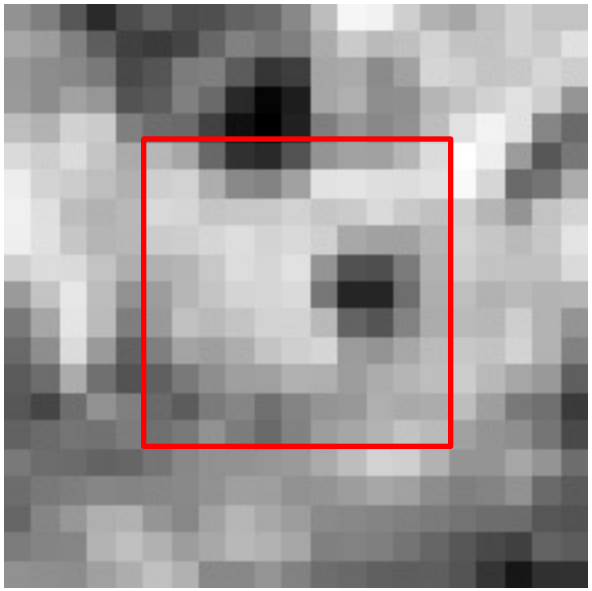


Corner Detection: Derivation

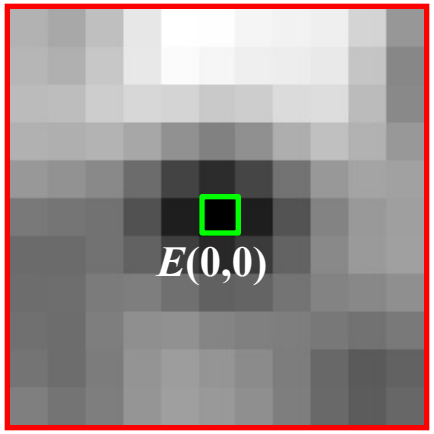
Change in appearance of window W for the shift $[u,v]$:

$$E(u,v) = \sum_{(x,y)\in W} (I(x+u,y+v) - I(x,y))^2$$

$I(x, y)$



$E(u, v)$



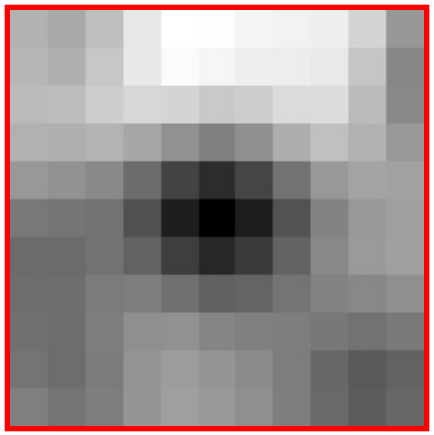
Corner Detection: Derivation

Change in appearance of window W for the shift $[u,v]$:

$$E(u,v) = \sum_{(x,y)\in W} (I(x+u, y+v) - I(x,y))^2$$

We want to find out how this function behaves for small shifts

$E(u, v)$



Corner Detection: Derivation

Change in appearance of window *W* for the shift [*u,v*]:

- First-order Taylor approximation for small motions [*u*, *v*]:

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

$$I(x+u, y+v) \approx I(x,y) + u \cdot I_x + v \cdot I_y$$

- Let's plug this into *E*(*u*,*v*):

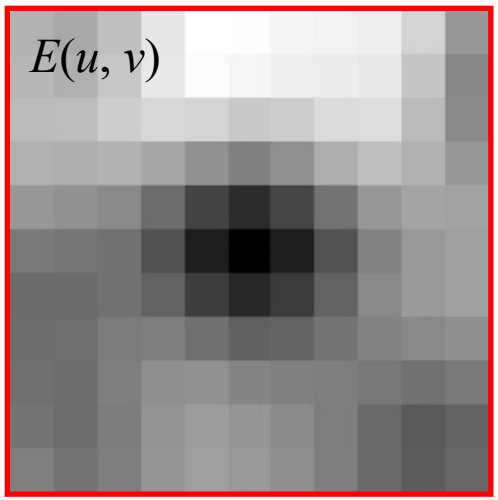
$$E(u,v) = \sum_{(x,y)\in W} \left(I(x+u, y+v) - I(x,y)\right)^2$$

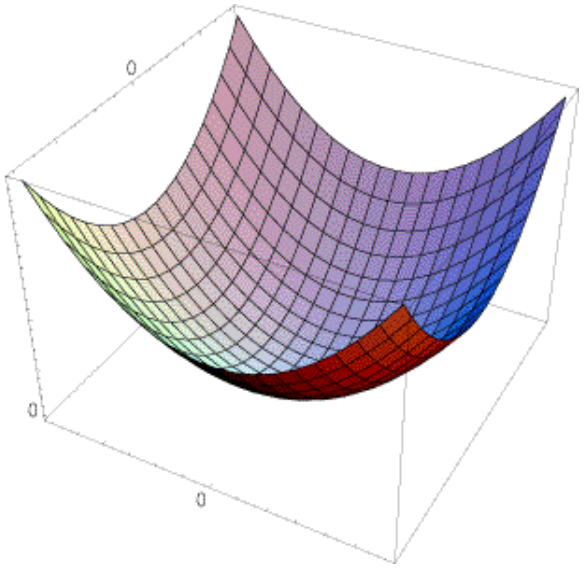
$$E(u,v) \approx \sum_{(x,y)\in W} (u \cdot I_x + v \cdot I_y)^2 = \sum_{(x,y)\in W} u^2 \cdot I_x^2 + 2uv \cdot I_x \cdot I_y + v^2 \cdot I_y^2$$

Corner Detection: Derivation

- $E(u,v)$ can be locally approximated by a quadratic surface:

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$



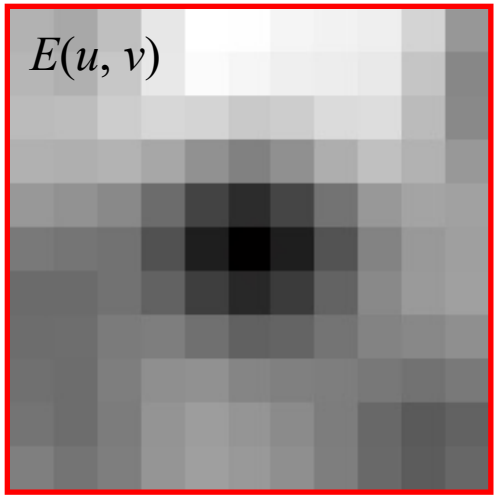


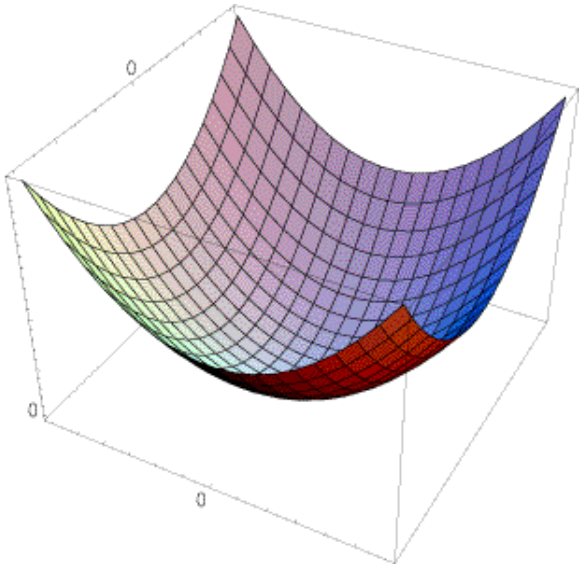
- In which directions does this surface have the fastest/slowest change?

Corner Detection: Derivation

- $E(u,v)$ can be locally approximated by a quadratic surface:

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$





- Eigenvectors given by the largest/smallest eigenvalue

Corner Detection: Derivation

- $E(u,v)$ can be locally approximated by a quadratic surface:

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$

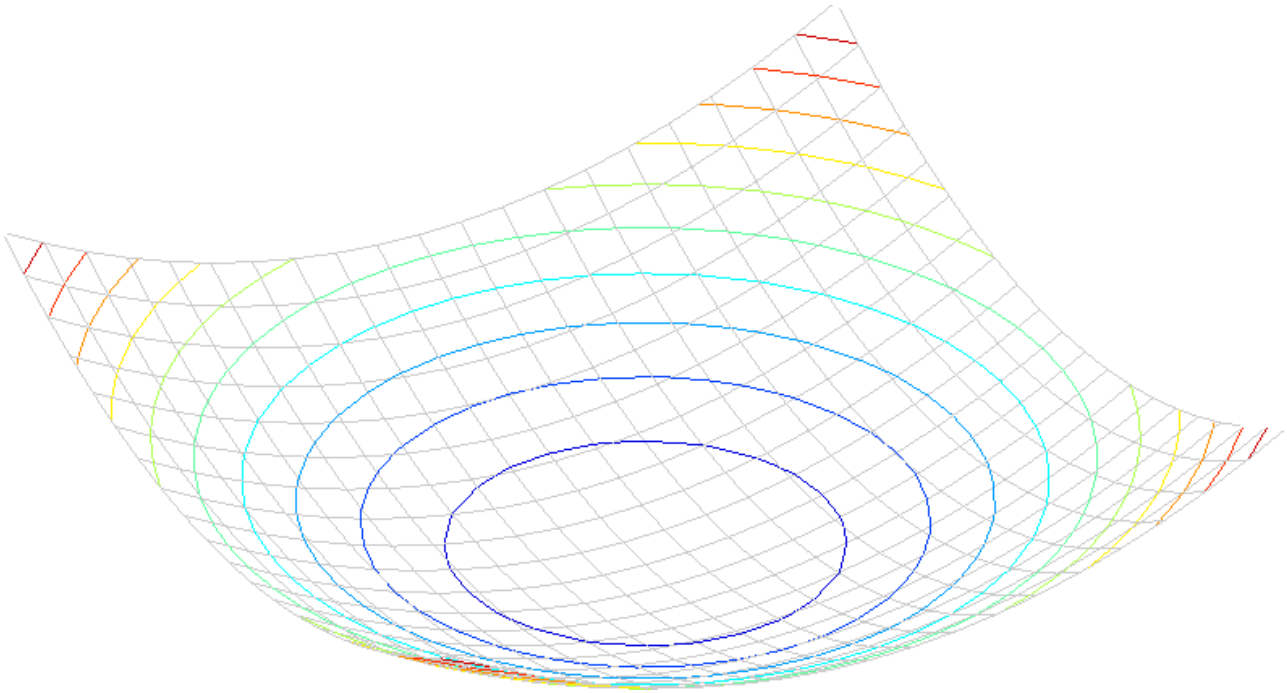
$$= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Second moment matrix M

Interpreting the second moment matrix

A horizontal "slice" of $E(u, v)$ is given by the equation of an ellipse:

$$[u \;\; v] \;\; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

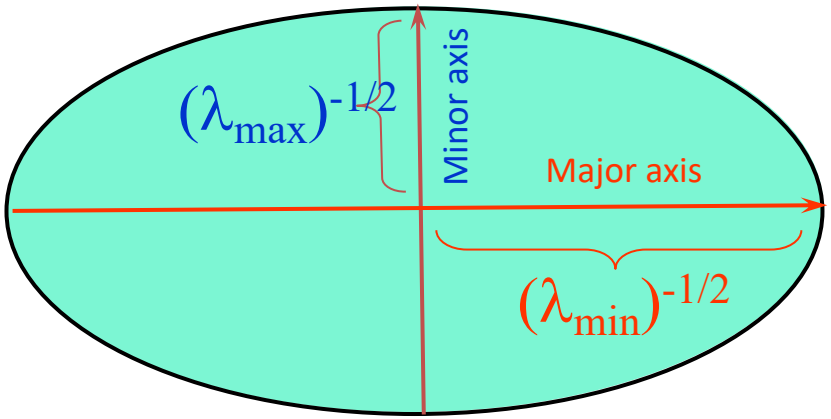


Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):

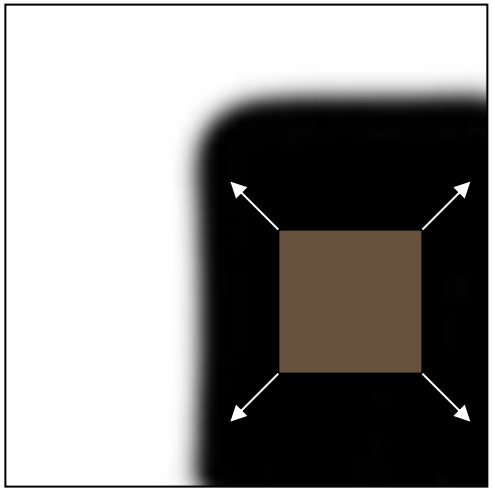
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} =$$

$$[u \;\; v] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 1$$



Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):



$$M = \begin{bmatrix} \sum_{x,y} I_x I_x & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y I_y \end{bmatrix}$$

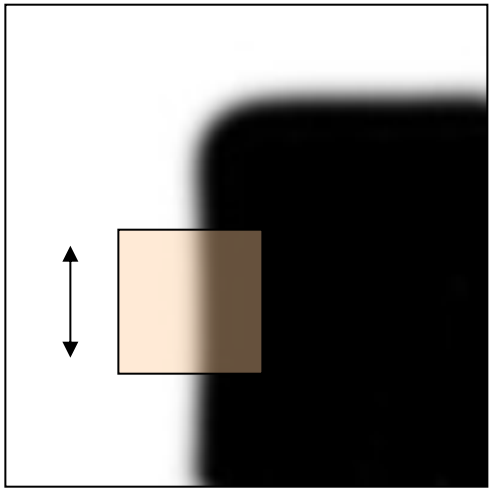
"flat" region:
no change in all directions

$$M \approx \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The eigenvalues of M = solutions of the equation $\det(M - \lambda I_2) = 0$ are $\lambda_1 \approx \lambda_2 \approx 0$

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):



$$M = \begin{bmatrix} \sum_{x,y} I_x I_x & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y I_y \end{bmatrix}$$

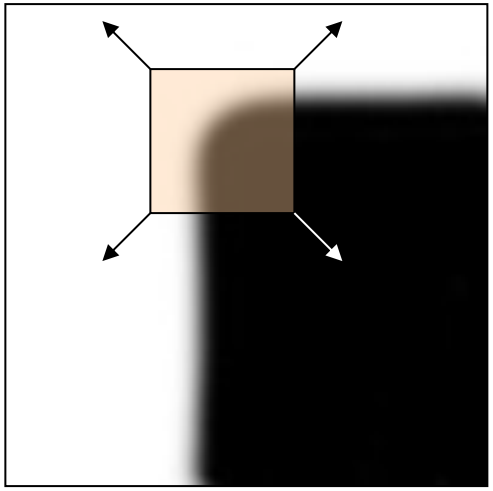
"edge":
no change along
the edge direction

$$M \approx \begin{bmatrix} \lambda & 0 \\ 0 & 0 \end{bmatrix}$$

The eigenvalues of M = solutions of the equation $\det(M - \lambda I_2) = 0$ are $\lambda_1 >> 0$, $\lambda_2 \approx 0$

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):



$$M = \begin{bmatrix} \sum_{x,y} I_x I_x & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y I_y \end{bmatrix}$$

"corner":
significant change in all directions

$$M \approx \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

The eigenvalues of M = solutions of the equation $\det(M - \lambda I_2) = 0$ are $\lambda_1, \lambda_2 >> 0$