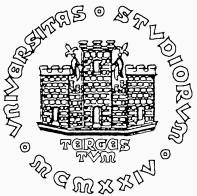
Computer Vision and Pattern Recognition

Course ID: 554SM – Fall 2018

Felice Andrea Pellegrino

University of Trieste
Department of Engineering and Architecture



554SM –Fall 2018

Lecture 4: Feature Detection

Key-point detection and matching

Features

Correspondences can be established by

1. detecting *features*;
2. finding **features whose appearance is similar across the images**.

Some classes of features:

- *keypoints* (or *interest points*, or *corners*): localized features, usually represented as the appearance of patches of pixels surrounding the point location;
- *edges*;
- *geometric primitives* such as curves and straight lines.

Tracking vs matching

There exist two main approaches to finding feature points and their correspondences:

- *detect-and-track*: find features **in one image** that can be accurately **tracked** using a local search technique (suitable when the images are taken from nearby viewpoints, or for video sequences);
- *detect-and-match*: find features **in all the images** under analysis and then **match** features based on their local appearance (more suitable when a large amount of motion is expected).

Feature detection and matching pipeline:

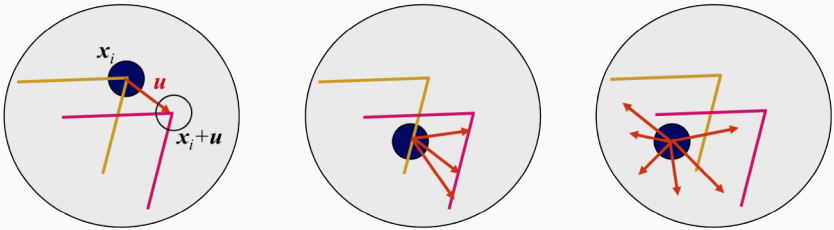
1. *feature detection*: each image is searched for locations that are likely to match well in other images;
2. *feature description*: each region around a detected keypoint is converted into a compact and possibly invariant[1] *descriptor*, to be matched against other descriptors;
3. *feature matching*: matching candidates are searched for in other images;
4. *feature tracking*: alternative to the previous stage. The search is restricted to a small neighborhood of the detected feature.

[1] to some transformations such as rotation, or illumination.

Harris corner detector

What kind of features are likely to provide reliable correspondences?

Intuitively, a necessary condition is to **exhibit intensity changes (gradients) in at least to different orientations** (figure below, left).
If the change occurs in just one direction (figure below, center) ambiguity occurs (*aperture* problem, or barber pole illusion).
When there is no change, matching is impossible (figure below, right).



Tracking corners (left), edges (center) and flat regions (right).

Harris corner detector (cont.)

Let's compute the variation (in terms of the sum of square differences) obtained by translating of $(\Delta x, \Delta y)$ a region $\mathcal{W}$ centered in the pixel (x, y) of the image I:

$$E(x, y, \Delta x, \Delta y) = \sum_{d_x, d_y \in \mathcal{W}} \left[I(x + d_x, y + d_y) - I(x + d_x + \Delta x, y + d_y + \Delta y) \right]^2 .$$

By Taylor series expansion we get

$$I(x + d_x + \Delta x, y + d_y + \Delta y) \approx I(x + d_x, y + d_y) + \nabla I(x + d_x, y + d_y)^\top \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix},$$

where $\nabla I = [\frac{\partial I}{\partial x} \;\; \frac{\partial I}{\partial y}]^\top \doteq [I_x \;\; I_y]^\top$, thus

$$E(x, y, \Delta x, \Delta y) = \sum_{d_x, d_y \in \mathcal{W}} [\Delta x \;\; \Delta y] \underbrace{\nabla I(x + d_x, y + d_y) \nabla I(x + d_x, y + d_y)^\top}_{\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{(x+d_x, y+d_y)}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

Harris corner detector (cont.)

Then

$$E(x, y, \Delta x, \Delta y) = [\Delta x \;\; \Delta y] \underbrace{\begin{bmatrix} \sum_{d_x, d_y \in \mathcal{W}} I_x^2 & \sum_{d_x, d_y \in \mathcal{W}} I_x I_y \\ \sum_{d_x, d_y \in \mathcal{W}} I_x I_y & \sum_{d_x, d_y \in \mathcal{W}} I_y^2 \end{bmatrix}}_{\doteq C \text{ (positive semidefinite by construction)}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

A smoother and rotationally invariant operator may be obtained by using a Gaussian weighting window:

$$E(x, y, \Delta x, \Delta y) = [\Delta x \;\; \Delta y] \underbrace{\begin{bmatrix} \sum I_x^2 w(d_x, d_y) & \sum I_x I_y w(d_x, d_y) \\ \sum I_x I_y w(d_x, d_y) & \sum I_y^2 w(d_x, d_y) \end{bmatrix}}_{\doteq \hat{C}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix},$$

where

$$w(d_x, d_y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{d_x^2 + d_y^2}{2\sigma^2}\right).$$

Harris corner detector (cont.)

For a displacement $(\Delta x, \Delta y)$ we get a weighted summed square difference of

$$E(x, y, \Delta x, \Delta y) = [\Delta x \;\; \Delta y]\, \hat{C} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

where $\hat{C}$ is symmetric and positive semidefinite. Thus $\hat{C}$ may be written as

$$\hat{C} = T \Lambda T^{\top} = T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} T^{\top}$$

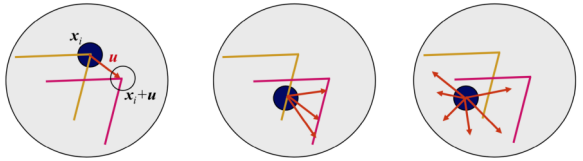
where $\lambda_2 \geq \lambda_1 \geq 0$. As a consequence, for unitary displacement:

$$\lambda_1 \leq E(x, y) \leq \lambda_2.$$

The matrix $\hat{C}$ is called the *second moment matrix*.

The eigenvalues of the second moment matrix reveal the structure of the neighborhood centered in (x, y).

Harris corner detector (cont.)



Corner (left), edge (center) and no structure (right).

Possible cases:

- corner: $\lambda_1, \lambda_2 \gg 0$;
- edge: $\lambda_1 \approx 0$, $\lambda_2 \gg 0$;
- no structure: $\lambda_1 \approx \lambda_1 \approx 0$.

Possible "cornerness" measures:

- λ_1 (minimum eigenvalue, (Shi and Tomasi, 1994));
- $\det(\hat{C}) - k\,\mathrm{tr}^2(\hat{C})$, with $k = 0.04$ (Harris and Stephens, 1988);
- $\dfrac{\det(\hat{C})}{\mathrm{tr}(\hat{C})}$ (harmonic mean (Noble, 1988)).

Feature detection procedure

Algorithm Basic feature detection

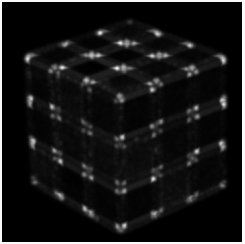
Input: Image

Output: Position of keypoints

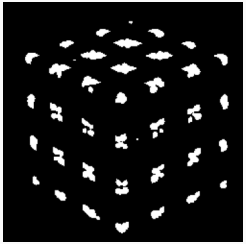
1: Compute I_x and I_y by convolving the image with derivatives of Gaussians.
2: Compute the 3 images $I_x^2, I_x I_y, I_y^2$ corresponding to the outer products of these gradients.
3: Convolve each of these images with a larger Gaussian.
4: Compute a scalar index using one of the formulas discussed above.
5: Find local maxima above a certain threshold.

Example

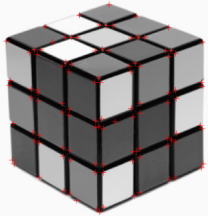
Original image



Cornerness measure (harmonic mean)



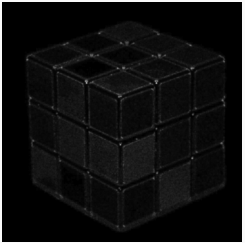
Pixels above a threshold



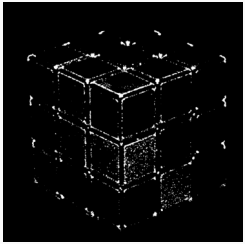
Markers indicate *local maxima* above the same threshold

Scale is important

Original image (4 times bigger than before)



Cornerness measure (same filters' size as before)



Pixels above a threshold

Markers indicate *local maxima* above the same threshold

Scale is important (cont.)

- Some features appear at a small scale;
- some features appear at a large scale;
- in the same location, feature at different scales may coexist.

Two approaches for dealing with scale:

1. extract features at many scales using a pyramid, and then matching features at the same level;
2. extract features that are stable (strong) in both location *and* scale.

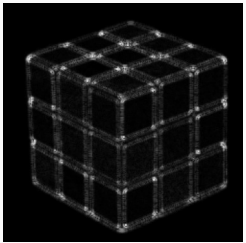
The second approach is based on the *scale-space representation* (Lindeberg, 1998) and will be described later on.

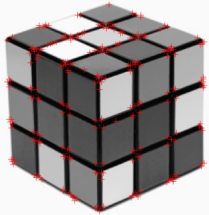
Hessian detector

The *Hessian detector* (Beaudet, 1978) exploits second derivatives, and precisely, the Hessian matrix.

$$H(x, y) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}.$$

- The strategy is to detect local maxima of $|\det H|$ above a certain threshold θ.
- Underlying idea: detect locations exhibiting **high curvature in two orthogonal directions**.
- In general, the Harris locations are more specific to corners, while the Hessian detector also returns many responses on regions with strong texture variation.





More on the second moment matrix

It is desirable to have

$$E(x, y, \Delta x, \Delta y) = [\Delta x \;\; \Delta y]\, \hat{C} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

as large as possible for all the directions of the vector $[\Delta x \;\; \Delta y]^\top$. The second moment matrix $\hat{C}$ is a weighted sum of matrices:

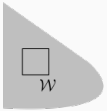
$$\hat{C} = \sum w(d_x, d_y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{(x+d_x, y+d_y)} .$$

$\hat{C}$ is symmetric and positive semidefinite, since the weights $w(d_x, d_y)$ are non-negative and each matrix in the sum is positive semidefinite, satisfying

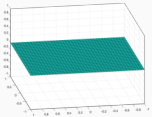
$$\begin{cases} \mathsf{tr} \geq 0 & \Rightarrow \; \lambda_1 + \lambda_2 \;\; \geq 0 \\ \mathsf{det} = 0 & \Rightarrow \; \lambda_1 \lambda_2 \;\;\;\;\;\; = 0 \end{cases} .$$

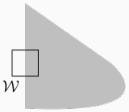
Thus, one eigenvalue is greater than or equal to zero and the other is zero.

More on the second moment matrix (cont.)

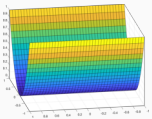


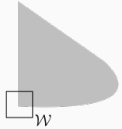
$$\hat{C} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$



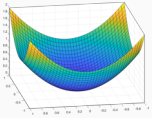


$$\hat{C} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

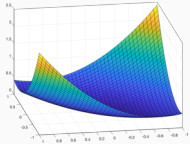




$$\hat{C} = \begin{bmatrix} A & 0 \\ 0 & C \end{bmatrix}$$



$$\hat{C} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$$



More on the second moment matrix (cont.)

Since $\hat{C}$ is symmetric, it is orthogonally similar to a diagonal matrix (it can be diagonalized by a rotation transformation):

$$\hat{C} = T \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} T^\top = [t_{\max} \;\; t_{\min}] \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} \begin{bmatrix} t_{\max}^\top \\ t_{\min}^\top \end{bmatrix}.$$

For unitary displacements

$$t = [\Delta x \;\; \Delta y]^\top, \quad ||t||_2 = 1,$$

$t_{\max}$ represents the direction along which E is maximal (and equal to $\lambda_{\max}$):

$$t_{\max}^\top \hat{C} t_{\max} = \begin{bmatrix} t_{\max}^\top t_{\max} & t_{\max}^\top t_{\min} \end{bmatrix} \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} \begin{bmatrix} t_{\max}^\top t_{\max} \\ t_{\min}^\top t_{\max} \end{bmatrix}$$

$$= [1 \quad 0] \begin{bmatrix} \lambda_{\max} & 0 \\ 0 & \lambda_{\min} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda_{\max}$$

Similarly, $t_{\min}$ represents the direction along which E is minimal, and equal to $\lambda_{\min}$. These directions are orthogonal.

Invariance and covariance

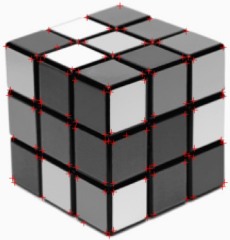
How do we expect a good detector to behave in the presence of *photometric* and *geometric* transformations?

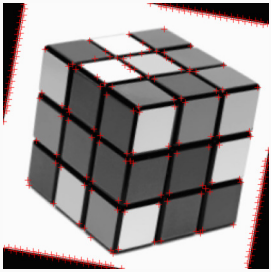
The keypoint locations should be *invariant* to photometric tranformations and *covariant* to geometric transformations.

- *Invariance*: image is transformed and keypoint locations do not change.
- *Covariance*: image is transformed and keypoint locations are transformed accordingly. More generally, if a detector detects regions of interest, **the regions detected after the transformation should be the same as the transformed versions of the regions detected in the original image** (image transformation and region detection commute).

Rotational covariance

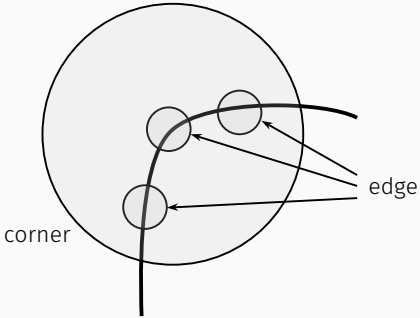
- Rotational covariance is covariance to rotations.
- A natural way to obtain rotation covariance is to design detectors that are rotationally invariant (i.e. that respond equally to the rotated versions of the same patch).
- Since the trace and the determinant are invariant to rotation, the Harris detector (and its variants that employ trace and determinant) provides rotational covariance.





Multi-scale approach

- Real scenes comprise structures that have different intrinsic scales: an object may appear in different ways at a different scales. For instance, and edge may become a corner.



- When no a priori knowledge is available about the proper scale to use in a given location of an image, a reasonable approach is to take into account a variety of scales.

Scale-space representation

- The *scale-space representation* of an image (Witkin, 1984; Lindeberg, 1998) is **a family of smoothed versions of the same image**, obtained by convolution with Gaussian kernels of various width $t = \sigma^2$.
- t is called the *scale parameter*.
- The triplet (x, y, t) represents **a point in the scale-space**.
- The members of the family take the form

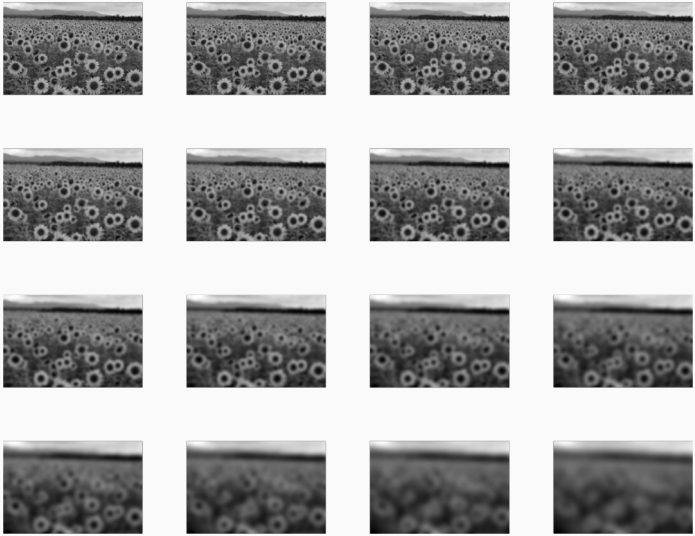
$$L(x, y, t) = G(x, y; \sqrt{t}) * f(x, y),$$

where

$$G(x, y; \sqrt{t}) = \frac{1}{2\pi t} \exp\left(-\frac{x^2 + y^2}{2t}\right).$$

As the scale t increases, the image becomes more and more blurred and finer details (i.e. details at smaller scales) are smoothed away.

Scale-space representation (cont.)



Example of scale-space representation.

Principle of scale selection

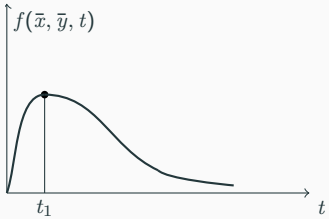
> `Principle for scale selection` (Lindeberg, 1998)
> *In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of* normalized *derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data.*

- Assumption: for a given scale, the keypoints can be detected as the (spatial) local maxima of a (possibly non-linear) combination of derivatives of various orders. As, for instance, the Harris corner detector does.
- Goal: in the whole scale-space, the keypoints should be detected as the local maxima (along space **and scales**) of the same combination of derivatives of various orders.
- Problem: derivatives at various scales need to be properly *normalized* (otherwise, performing a max search over the scales is meaningless).

Normalized derivatives

What is a reasonable way to normalize the derivatives?

- Ideally, a feature that has no intrinsic scale (such as a sharp edge infinitely long) should not exhibit a local maximum over the scales.
- On the other hand, a feature that *does have* an intrinsic scale, should exhibit a maximum at that scale.



Local maximum over the scales. The function f is a combination of (normalized) derivatives of various orders.

- In general, the proper normalization could depend on the kind of feature to be detected.

Normalized derivatives (cont.)

Consider a one-dimensional sinusoidal signal:

$$g(x) = \sin \omega_0 x.$$

By convolving with a Gaussian (it is straightforward by applying the convolution theorem) we get the scale-space representation:

$$L(x, t) = e^{-\omega_0^2 t/2} \sin \omega_0 x.$$

Thus, the amplitude decreases with scale; at scale t the local maxima w.r.t. space x are

$$L_{\max}(t) = e^{-\omega_0^2 t/2}.$$

The same holds for the m-th order smoothed derivative whose local maxima are

$$L_{\max}^{x^m}(t) = \omega_0^m e^{-\omega_0^2 t/2}.$$

None of the m-th order derivatives has a maximum over the scales. However, the signal has a characteristic length (the wavelength $2\pi/\omega_0$) at which it would be desirable to get a maximum.

Normalized derivatives (cont.)

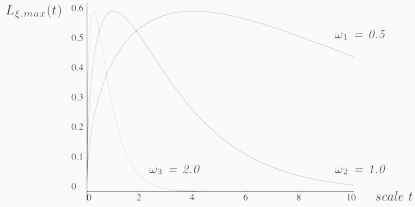
By introducing the γ-*normalized derivative operator*

$$t^{\gamma/2}\partial x,$$

which corresponds to the change of variable $\xi = x/t^{\gamma/2}$, we get a non monotonic behavior of the m-th order derivative as a function of scale:

$$L^{\xi^m}_{\max}(t) = t^{m\gamma/2}\omega_0^m e^{-\omega_0^2 t/2}.$$

There is a unique maximum at $t_{\max} = \gamma m/\omega_0^2$, thus the location of the maximum across the scales is a function of ω_0.



The amplitude of first order normalized derivatives as a function of scale for sinusoidal signals of different angular frequencies and $\gamma = 1$ (from (Lindeberg, 1998)).

Normalized derivatives (cont.)

The value of γ can be selected depending on the feature to be detected (and, thus, on the combination of derivatives employed). Example of features are

- edges;
- junctions;
- corners;
- ridges;
- blobs.

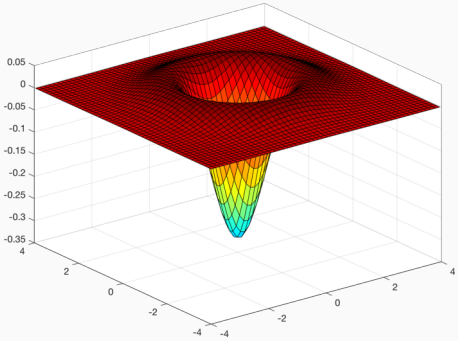
Blobs are particularly important, as one of the most effective keypoint detectors, the Scale Invariant Feature Transform (SIFT (Lowe, 2004)) relies on them.

Joan Miró, Ciphers and Constellations in Love with a Woman, 1941

Blob detection using Laplacian of Gaussian

A common blob detector is the Laplacian of Gaussian (LoG) filter:

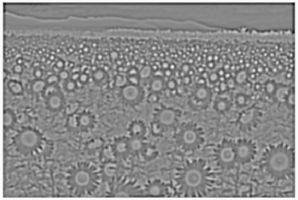
$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma).$$



The LoG filter:

- is not oriented;
- has a center-surround response;
- is a band-pass filter;
- is *scale-specific* because it provides strong positive responses for dark blobs of size $\sqrt{2}\sigma$ and strong negative responses for bright blobs of similar size.

Blob detection using Laplacian of Gaussian (cont.)


The example above shows the scale-specificity of the LoG filter:

- left: original image;
- center: response of a LoG filter having $\sigma = 3$;
- right: response of a LoG filter having $\sigma = 15$.

Peaks of the response correspond to dark regions of a given size, depending on σ.

Notice, however, that high response can be achieved along strong ridges of proper size.

Blob detection with automatic scale selection

A multi-scale approach to blob detection can be achieved as follows (Lindeberg, 1998):

1. compute the scale-space normalized Laplacian of Gaussian of the image

$$\nabla^2_{\text{norm}} L(x, y, t) = t\nabla^2 G(x, y; \sqrt{t}) * f(x, y),$$

   corresponding to $\gamma = 1$ and $m = 2$;
2. find the local extrema in the scale-space, i.e. local maxima and minima with respect **to both space and scale**.

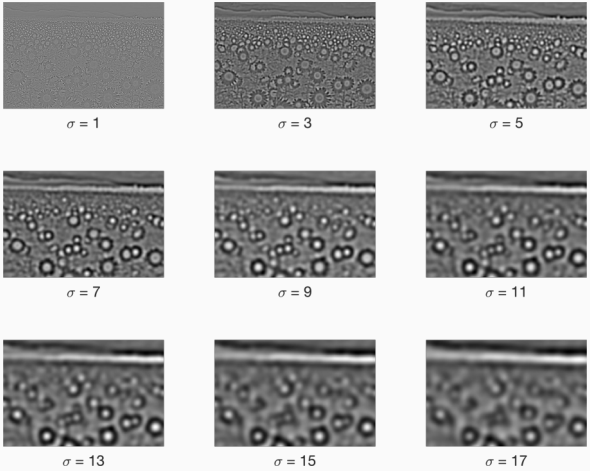
In practice, since both the scale and the space are discrete, the detection of the extrema is performed through a comparison within a neighborhood window (usually of size $3 \times 3 \times 3$).

The scale at which the extremum occurs, is a *characteristic scale* for a given image location.

- It can be shown that the approach guarantees **scale covariance**;
- due to the rotational symmetry of the LoG filter it has also **rotational covariance**.

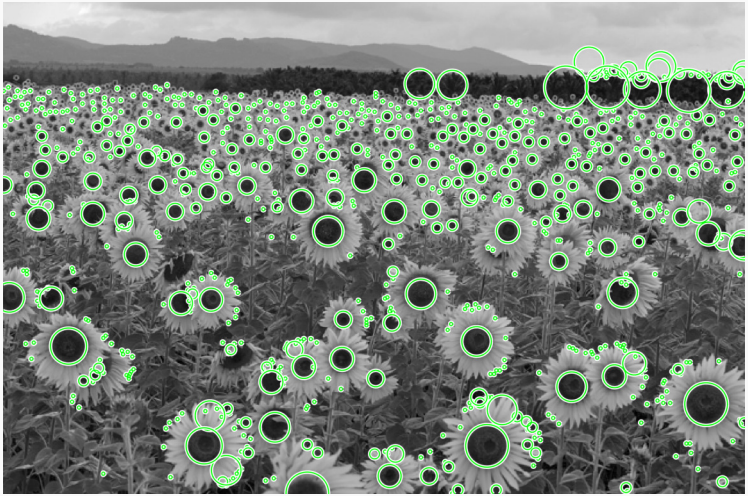
Example

The scale-space normalized Laplacian of Gaussian, for 9 different scales:



Example (cont.)

The local extrema above a threshold, in the scale-space. The center of the circles represents the location while the radius is proportional to the characteristic scale, precisely $\sqrt{2t} = \sqrt{2}\sigma$.



Affine covariance

For many applications, scale and rotation covariance may be not sufficient. When a scene undergoes viewpoint changes, local changes may be approximated with affine transformations. Thus, *affine covariance* is desirable.

There exist many affine-covariant region detectors (Mikolajczyk et al., 2005). Some of them use an *affine normalization* approach, based on the second moment matrix.

Let $x = [u - u_0 \quad v - v_0]^\top$ (where $[u_0 \quad v_0]^\top$ is the location of a keypoint) and A be the corresponding second moment matrix. The level curves of the quadratic form

$$x^\top A x$$

are ellipses, whose orientation and elongation depend on the local structure around the keypoint.

Affine covariance (cont.)

If the neighborhood undergoes the affine transformation

$$\tilde{x} = A^{1/2}x,$$

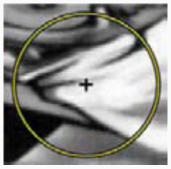
then, in the new coordinates the same quadratic form will be

$$x^\top Ax = (A^{-1/2}\tilde{x})^\top A(A^{-1/2}\tilde{x}) = \tilde{x}^\top \underbrace{A^{-1/2}AA^{-1/2}}_{I}\tilde{x} = \tilde{x}^\top I\tilde{x}.$$

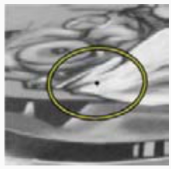
Thus, the local structure has been "stretched" (or "normalized") and the new second moment matrix is the identity.

$x_0 \rightarrow$
$A_0^{-1/2}\tilde{x}_0$

$\tilde{x}_0 \rightarrow$
$R\tilde{x}_1$



$A_1^{-1/2}\tilde{x}_1$
$\leftarrow x_1$



After normalization, the local neighborhoods of the same actual location, are related by a pure rotation R.

SIFT detector

The Scale Invariant Feature Transform (SIFT) detector (Lowe, 2004) is widely recognized as a robust and efficient detector. It can be considered a computationally efficient and, to some extent, improved version of the general LoG approach.

Characteristics of the SIFT approach:

- SIFT uses Difference of Gaussians (DoG) to approximate the Laplacian of Gaussian operator;
- SIFT uses a pyramid (scale are chosen at sub-octave resolution, and undersampling occurs at each octave);
- SIFT detects local extrema at sub-pixel and sub-scale level, by fitting a quadratic function of $\Delta x, \Delta y, \Delta \sigma$ to a raw scale-space location and setting its gradient to zero;
- SIFT removes the potential edges and ridges by using the Hessian matrix of the DoG.

Difference of Gaussians

The Laplacian of Gaussian can be approximated by a Difference of two Gaussians (DoG) at different scales. It is easy to check that

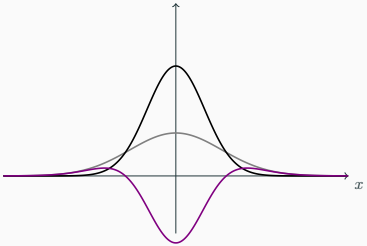
$$\frac{\partial G(x,y,\sigma)}{\partial \sigma} = \sigma \nabla^2 G(x,y,\sigma),$$

thus, by substituting the derivative with the difference quotient we get

$$\frac{G(x,y;\sigma+\Delta\sigma) - G(x,y;\sigma)}{\Delta\sigma} \approx \sigma \nabla^2 G(x,y,\sigma).$$

By taking $\sigma + \Delta\sigma = k\sigma$ (for $k > 1$) we get $\Delta\sigma = (k-1)\sigma$, thus

$$G(x,y;k\sigma) - G(x,y;\sigma) \approx (k-1)\sigma^2 \nabla^2 G(x,y;\sigma) = (k-1)\nabla^2_{\text{norm}} G(x,y;\sigma).$$



Two one-dimensional Gaussians of different σ and their difference.

- the approximation incorporates normalization;
- the closer is k to 1, the better the approximation;
- in a multi-scale framework, a convenient choice is $k = \sqrt[n]{2}$, to get n levels per octave;

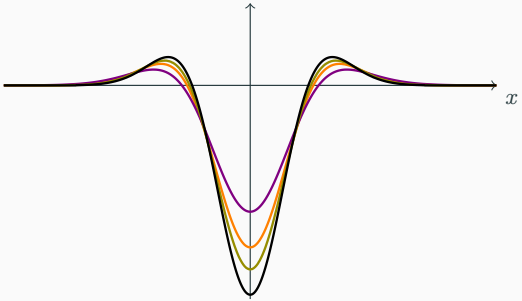
Difference of Gaussians (cont.)

Different approximations of a one-dimensional Laplacian of Gaussian (black), obtained as a Difference of Gaussians

$$G(x, y, k\sigma) - G(x, y, \sigma).$$

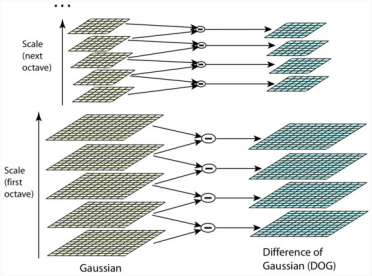
- $k = \sqrt{2}$ (violet);
- $k = \sqrt[4]{2}$ (orange);
- $k = \sqrt[8]{2}$ (olive).

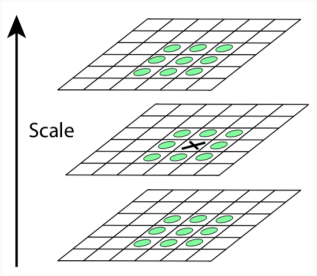
SIFT employs $k = \sqrt[4]{2}$, to get 4 levels per octave.



Pyramid with sub-levels

Scale are chosen at sub-octave resolution, and undersampling occurs at each octave.





For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated (Lowe, 2004).

Notice that to search for extrema over n levels per octave, $n+2$ DoG images are needed and therefore $n+2+1=n+3$ Gaussian filtered images are needed.

Sub-pixel and sub-scale precision

SIFT detects local extrema at sub-pixel and sub-scale level, by fitting a quadratic function of x, y, σ to a raw scale-space location and setting the gradient to zero.

Let $D(x, y, \sigma)$ denote the DoG approximation of the normalized Laplacian at scale σ, in the neighborhood of a pixel-level local extremum. By using Taylor expansion:

$$D(x + \Delta x, y + \Delta y, \sigma + \Delta\sigma) \approx D(x, y, \sigma) + \nabla^\top D(x, y, \sigma)\Delta + \frac{1}{2}\Delta^\top H(x, y, \sigma)\Delta,$$

which is a quadratic function of $\Delta = [\Delta x \;\; \Delta y \;\; \Delta\sigma]^\top$.

The gradient $\nabla D(x, y, \sigma)$ and the Hessian $H(x, y, \sigma)$ can be approximated by differences of neighboring sample points. Then, the sub-pixel and sub-scale extremum location can be obtained by setting to zero the gradient of the quadratic function:

$$0 = \nabla D(x, y, \sigma) + H(x, y, \sigma)\Delta,$$

hence

$$\Delta = -H^{-1}(x, y, \sigma)\nabla D(x, y, \sigma).$$

Eliminating edges responses

As the LoG, the DoG function will have a strong response along edges (and ridges), even if the location along the edge is poorly determined and thus noise-sensitive.

A keypoint drift due to noise may introduce errors in subsequent processing, thus it should be avoided.

SIFT rejects poorly defined peaks in the DoG by means of the **Hessian matrix of the DoG** computed in correspondence of the (x, y, σ) location of the keypoint.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}.$$

The rejection criterion is that the ratio $r = \alpha/\beta$ between the maximum and minimum eigenvalue (α and β, respectively) exceeds a certain threshold $\bar{r}$. Since

$$\frac{\det H}{\mathrm{tr}^2 H} = \frac{\alpha\beta}{(\alpha+\beta)^2} = \frac{\beta^2 r}{\beta^2(r+1)^2} = \frac{r}{(r+1)^2}$$

is monotonic with respect to r, the peak is rejected when

$$\frac{\det H}{\mathrm{tr}^2 H} \geq \frac{\bar{r}}{(\bar{r}+1)^2}.$$

SIFT (detector) algorithm

Algorithm SIFT detector

Input: Image

Output: Position, scale and orientation of keypoints

1: Compute the Gaussian pyramid.
2: Compute DoG by subtracting nearby levels of the same octave.
3: Find scale-space extrema at pixel level.
4: Refine location of scale-space by interpolation.
5: Reject extrema with absolute value below a threshold.
6: Reject extrema corresponding to edges, based on the ratio between maximum and minimum eigenvalue of the Hessian.
7: Compute the orientation of the keypoints by histogram of gradients.

The last step (computing the orientation) will be discussed next.

Example

Example of application of the SIFT detector. The detector has been applied to the image below, obtaining 3370 keypoints. A random selection of 1000 keypoints is shown. The size of the circle depends on the scale of the feature and the segment denotes the orientation.

Descriptors

Given a keypoint (found, for instance, using SIFT), many different descriptors can be computed, independent of the detector employed for finding the keypoint.

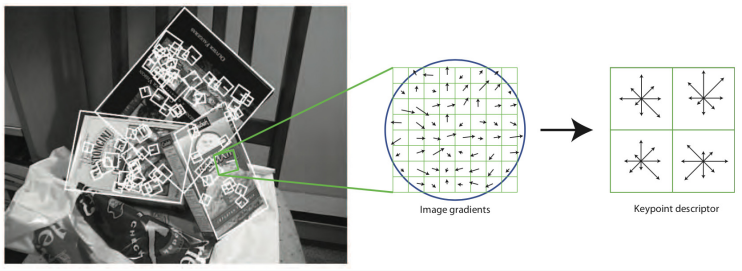
The simplest descriptor is the neighborhood of the keypoint as it is (a patch centered in the keypoint location). In some situations (video sequences or rectified images[2]) such a patch may be sufficient for an acceptable matching). In general, however, more sophisticated and invariant/covariant descriptors are needed. Possible descriptors are:

- SIFT;
- multi-scale oriented patches (MOPS, Brown et al. (2005));
- PCA-SIFT (Yan Ke and Sukthankar, 2004);
- Gradient Location Orientation Histogram descriptor (GLOH, (Mikolajczyk and Schmid, 2005));
- steerable filters (Freeman et al., 1991).

[2] see the Lecture on Stereopsis.

SIFT descriptor

To each keypoint, a *descriptor* is associated, which is basically a histogram of the gradient orientation in its neighborhood.



SIFT descriptor (cont.)

To achieve scale invariance, the size of the neighborhood should depend on the detection scale; thus, SIFT considers a fixed 16×16 window in the pyramid level where the point has been detected.

To achieve rotation invariance, a *dominant orientation* is computed as follows:

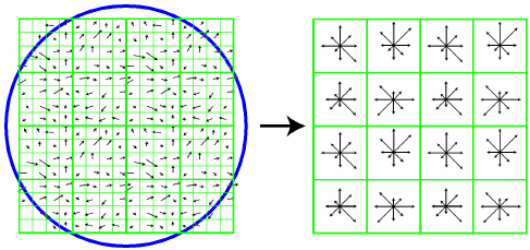
- the gradient is computed for each pixel of the 16×16 window;
- a 36 angular bins histogram is built, where the contribution to each bin is weighted by the gradient magnitude and a Gaussian centered in the window;
- the dominant orientation is the mode of the histogram;
- for increasing the angular resolution, the mode is estimated by quadratic interpolation with the two neighboring bins.
- the histogram is considered to be multi-modal (thus multiple dominant orientations are accepted) when there are local maxima of the histogram within the 80% of the mode. In that case, a descriptor is generated for each dominant orientation.

SIFT descriptor (cont.)

Given the dominant orientation, the descriptor is constructed as follows:

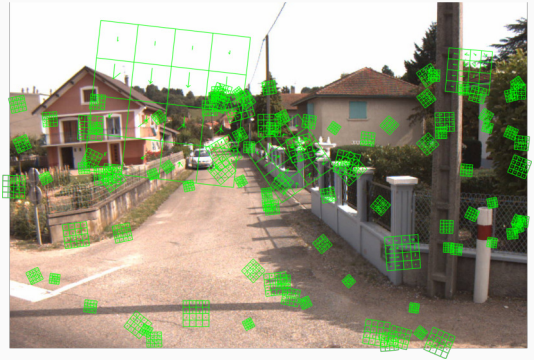
- a 16×16 window, rotated according to the dominant orientation is considered (at the same pyramid level of detection);
- a Gaussian-weighted gradient is computed for each pixel;
- the window is divided into 16 subwindows, each of size 4×4;
- an 8 bin histogram of orientations, weighted by gradient magnitude, is constructed for each subwindow;

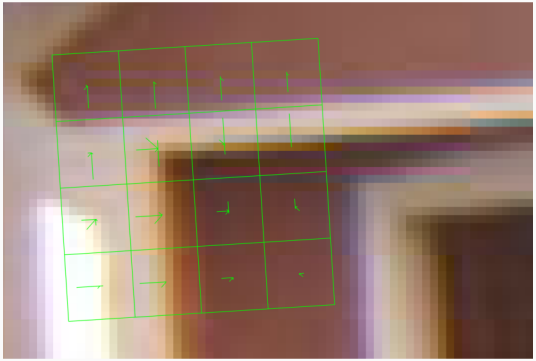
- the resulting 128 non-negative values are collected into a vector;
- the vector is normalized to unit length (to achieve contrast invariance);
- finally, values below a threshold (0.2) are trimmed to zero and the vector is normalized again.
- the obtained vector is the SIFT descriptor.



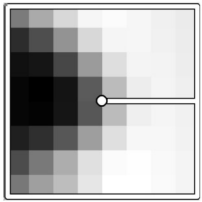
SIFT descriptor (cont.)

In the figures below, the representation of the descriptors of some randomly chosen keypoints are superimposed to the image. The rotation of the square denotes the dominant orientation.





MOPS descriptor



Multi-scale oriented patches (MOPS) descriptors (Brown et al., 2005) are 8×8 **intensity patches**, properly rotate, sampled and normalized.

- MOPS are obtained by sampling the neighborhood of the keypoint with a sample spacing of five pixels relative to the detection scale. The low frequency spatial sampling gives the features some robustness to interest point location error and is achieved by sampling at a higher pyramid level than the detection scale.
- The intensity values are standardized (i.e. rescaled so that the mean is zero and the variance is one).
- The patches are rotated with respect to the dominant orientation.

MOPS are simple to implement and perform reasonably well for feature matching whenever the feature do not exhibit too much variation across images (for instance for image stitching).

PCA-SIFT descriptor

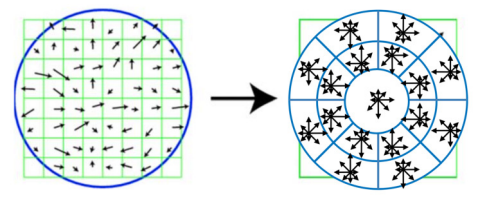
PCA-SIFT descriptors (Yan Ke and Sukthankar, 2004) are more compact descriptors than SIFT. Given the location, scale, and dominant orientations of the keypoint, they are computed as follows:

- a 41×41 patch at the given scale, centered over the keypoint is extracted;
- the patch is rotated to align its dominant orientation to a canonical direction;
- the x and y derivatives are computed, producing a vector of dimension 2×(41-2)×(41-2)=3042;
- **the descriptor is computed by projecting the 3042-dimensional vector onto a subspace of much smaller dimension** (good results are reported in (Yan Ke and Sukthankar, 2004) for a subspace of dimension 36);
- for keypoints with multiple dominant orientations, a representation for each orientation is built, in the same manner as SIFT.

How is the subspace chosen? The subpace is actually an eigenspace obtained by applying Principal Component Analysis (PCA); more precisely:

- 21K patches where collected on a diverse collection of images;
- each was processed to create a 3042-element vector;
- PCA was applied to the covariance matrix of these vectors;
- the matrix consisting of the top n eigenvectors was stored and used as the projection matrix for PCA-SIFT.

GLOH descriptor

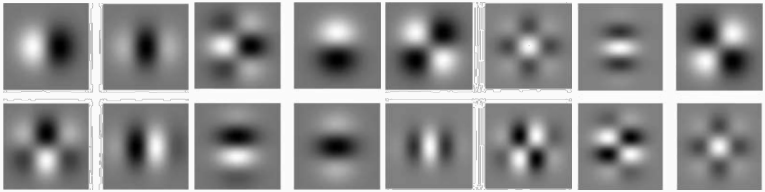


The Gradient Location Orientation Histogram descriptor (Mikolajczyk and Schmid, 2005) is a variant on SIFT that uses a **log-polar binning structure** instead of the quadrants employed in SIFT.

- The SIFT descriptor for a log-polar location grid with 3 bins in radial direction (the radius set to 6, 11 and 15) and 8 in angular direction is computed;
- 17 location bins are thus obtained (the central bin is not divided in angular directions);
- the gradient orientations are quantized in 16 bins, thus a 272 bin histogram is obtained.
- The descriptor is obtained by projecting the vector of bin values onto a subspace of dimension 128 obtained by PCA.
- Similarly to PCA-SIFT, the covariance matrix for PCA is estimated on 47 000 image patches collected from various images and the 128 largest eigenvectors are used for description.

GLOH has been reported (Mikolajczyk and Schmid, 2005) to slightly outperform SIFT in typical matching applications.

Steerable filters as descriptors



The steerable filters (Freeman et al., 1991) (or, more precisely, the basis sets for steerable filters) can be used as a compact descriptor.

- The descriptor is compact because it is composed of the filter responses corresponding to filters centered in the keypoint (n filters result in a descriptor of dimension n);
- in the figure are represented the x and y basis filters for computing derivatives up to the 4-th order;
- in that case the actual descriptor's length is 14 because two y-oriented filters are the same or the opposite of an x-oriented filter.

A noticeable feature of the steerable filters as descriptors is that they are suitable for representing even (e.g. ridge-type) features.

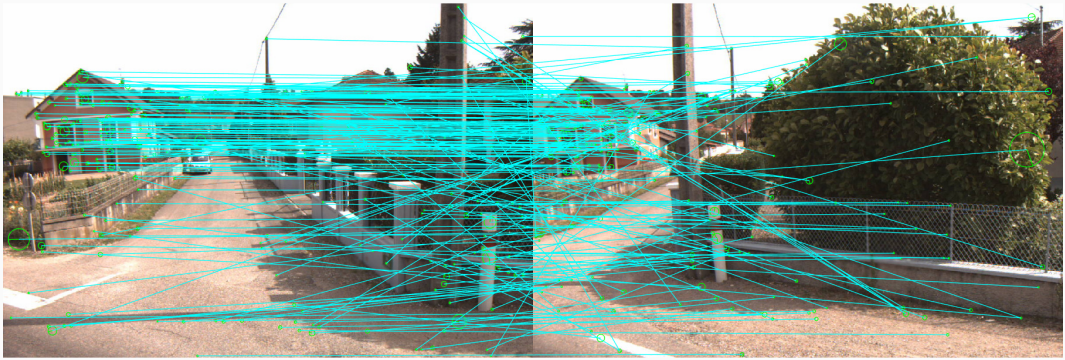
Steerable filters have been reported (Mikolajczyk and Schmid, 2005) as the most effective among the short length descriptors.

Matching

When the keypoints and the descriptors have been extracted from two or more images, the next step is to establish matches between these images. This is the *feature matching* step.

Matching

When the keypoints and the descriptors have been extracted from two or more images, the next step is to establish matches between these images. This is the *feature matching* step.



Distance

We assume in the following that the descriptor has been designed in such a way that **the Euclidean distance in feature space can be directly used for ranking potential matches**: i.e., given two descriptors D_A and D_B, their distance

$$d(D_A, D_B) = \sqrt{(D_A - D_B)^\top (D_A - D_B)},$$

is meaningful in evaluating the goodness of the match. In some cases, better results can be obtained by using a weighted distance:

$$d_W(D_A, D_B) = \sqrt{(D_A - D_B)^\top W (D_A - D_B)},$$

where W is a positive semidefinite matrix.

In some other cases, other distances can be employed. For instance, when matching histograms a flexible metric is the *earth mover's distance* (EMD, Rubner et al. (1998)).

`Fixed threshold`

The simplest matching strategy is fixing a threshold: D_A and D_B are matched whenever $d(D_A, D_B) \leq \theta$. However, θ may be difficult to set; the useful range of thresholds can vary a lot depending on the region of the feature space.

`Nearest neighbor`

A better strategy in some cases is to simply match the nearest neighbor in feature space. Since some features may have no matches, a threshold is still used to reduce the number of wrong matches.

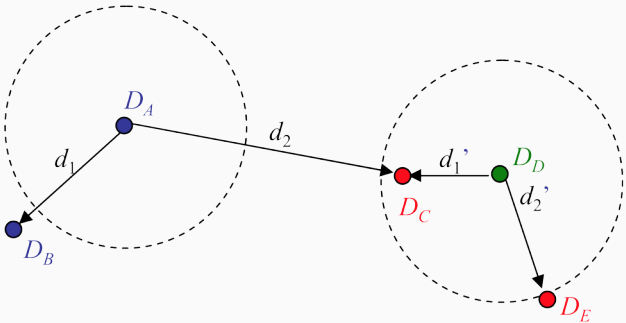
`Nearest neighbor distance ratio (NNDR)`

The NNDR is a heuristic consisting in comparing the distance of the nearest neighbor to that of the second nearest neighbor:

$$\text{NNDR} = \frac{d_1}{d_2} = \frac{d(D_A, D_B)}{d(D_A, D_C)}$$

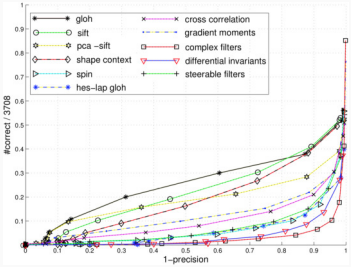
where D_A is the target descriptor and D_B and D_C are its closest two neighbors.

Matching strategies (cont.)

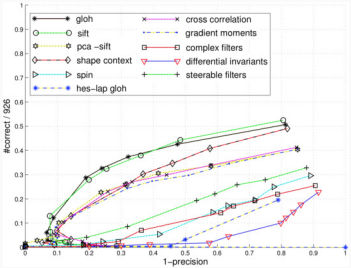


Comparison between fixed threshold, nearest neighbor, and nearest neighbor distance ratio matching. The NNDR correctly matches D_A with D_B, and correctly rejects matches for D_D.

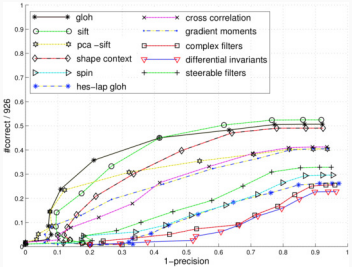
Matching strategies (cont.)



Fixed threshold.



Nearest neighbor.



Nearest neighbor distance ratio.

In the figures above (Mikolajczyk and Schmid, 2005), the performance of different feature descriptors are reported for the three matching strategies. The ranking of the descriptors is roughly the same, but the overall matching performance changes significantly between the different matching strategies.

Efficient matching

The simplest way to find all corresponding feature points is to compare all features against all other features in each pair of potentially matching images.

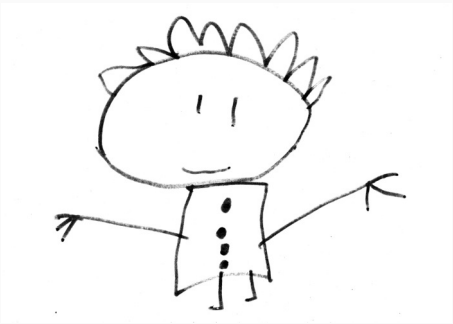
Unfortunately, this is **quadratic in the number of extracted features**, which makes it impractical for most applications.

Efficient **data structures** and **algorithms** are needed to perform the matching quickly. For efficient matching strategies, see Szeliski (2010), chapter 4.

Edge detection

Edges

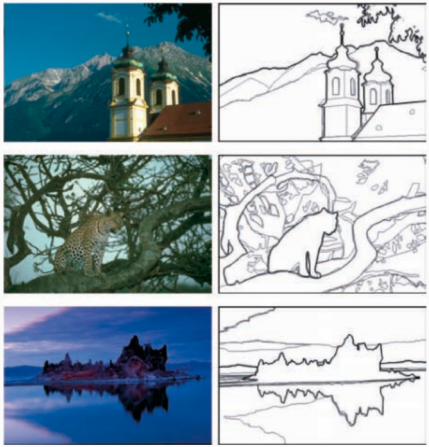
Cave drawing at Chauvet, France (about 30,000 B.C.)



Line drawing by 4-year old Greta (2018)

Edges (cont.)

- Edges are (spatially) rapid changes in intensity;
- edges often carry a lot of semantic information (for instance, they are typically associated to the boundary of the objects present in the image);
- the goal of *edge detection* is to find sudden changes in intensity (i.e. discontinuities) in an image;
- ideally, edge detection should provide results similar to human drawn boundaries (such as those in the figures), however:
  - when drawing edges, humans make selections using object-level information;
  - inspection reveals that perceptually important “edges” may be extremely weak or even do not exist.



Origin of edges

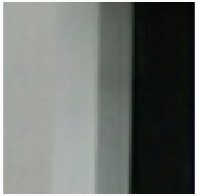
Edges originate from discontinuities:

- surface normal discontinuity;
- depth discontinuity;
- surface color discontinuity;
- illumination discontinuity.

Origin of edges

Edges originate from discontinuities:

- surface normal discontinuity;
- depth discontinuity;
- surface color discontinuity;
- illumination discontinuity.



Surface normal discontinuity.

Origin of edges

Edges originate from discontinuities:

- surface normal discontinuity;
- depth discontinuity;
- surface color discontinuity;
- illumination discontinuity.

Depth discontinuity.

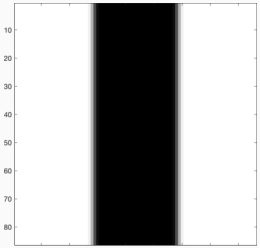
Origin of edges

Edges originate from discontinuities:

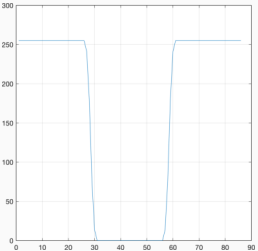
- surface normal discontinuity;
- depth discontinuity;
- surface color discontinuity;
- illumination discontinuity.

Surface color discontinuity.

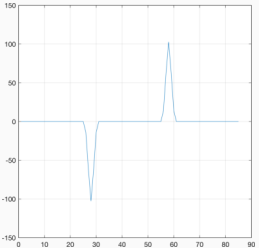
Edges and derivatives



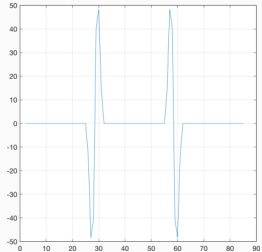
Image



Horizontal profile



First derivative (first order finite difference)

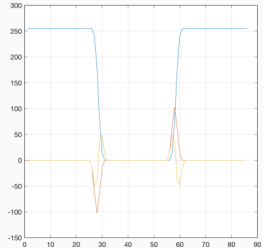


Second derivative (second order finite difference)

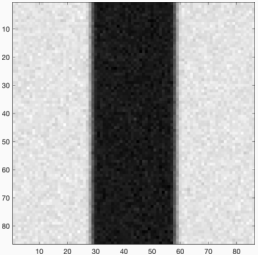
An analysis of a one-dimensional profile shows that:

- edges occur at **local extrema of the first derivative** (or, in other words, local maxima of the magnitude of the first derivative);
- edges occur at **zero crossings of the second derivative**;

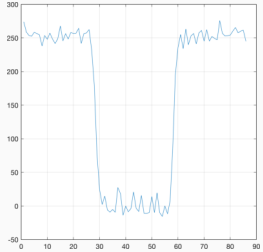
Remember that, although it is convenient reasoning in terms of a continuous domain, digital images have discrete domain hence “first derivative” is actually a first order difference (similarly, the second derivative is a second order filter).



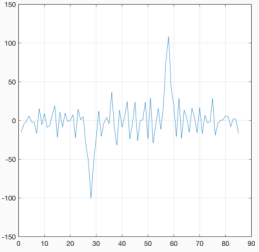
Effect of noise



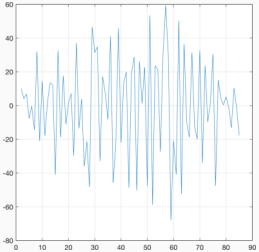
Image



Horizontal profile

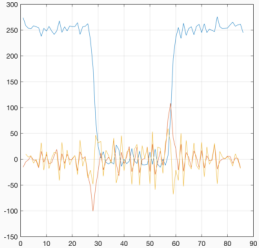


First derivative (first order finite difference)

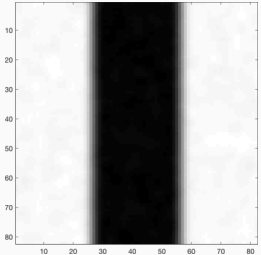


Second derivative (second order finite difference)

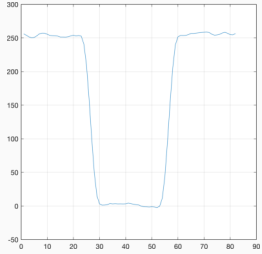
- Unfortunately, derivatives are very sensitive to noise (they are basically a high-pass filter).
- Adding some Gaussian noise as in the figure above:
  - the edges in the image are still clearly distinguishable by a human observer;
  - a lot of local extrema of the first derivative arise;
  - a lot of zero crossing of the second derivative arise.



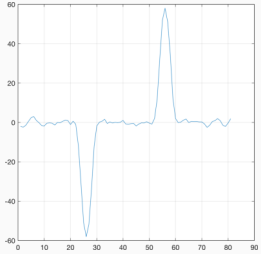
Filtering noise



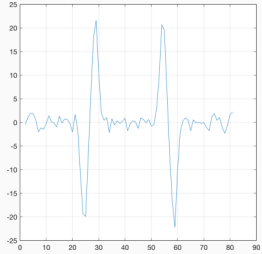
Filtered image



Horizontal profile

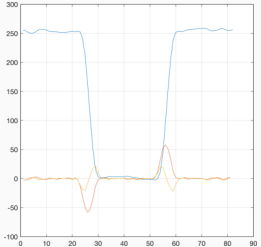


First derivative (first order finite difference)



Second derivative (second order finite difference)

- Solution: remove the noise (by low-pass filtering) before computing the derivatives.
- By using linear filtering, such a low-pass filtering amounts to a convolution of the signal f with a proper kernel g, for instance a Gaussian kernel.
- The figures show that the first and second derivative of the filtered image $f * g = g * f$ are smoother that before.



Derivatives and convolutions

- Consider a 1D signal $f(x)$;
- linearlty filtering the signal amounts to computing

$$f * g$$

  i.e. convolving f with a kernel g (e.g. a Gaussian kernel);
- by taking the first derivative we get

$$\frac{d}{dx}(g * f);$$

- since the differentiation is a linear operator, it commutes with other linear filtering operators, thus

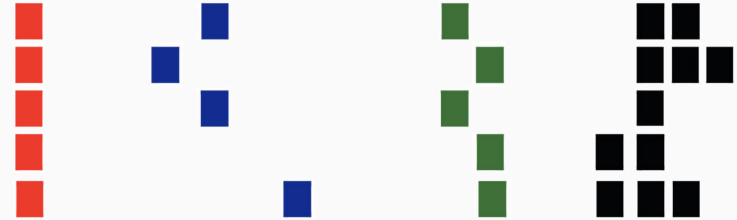
$$\frac{d}{dx}(g * f) = \left(\frac{d}{dx} g\right) * f.$$

- The same holds for 2D signals: it follows that the derivative of the filtered image can be computed by **convolving the image with the derivative of the kernel**.

Properties of an optimal edge detector

Criteria for an “optimal” edge detector:

- **Good detection**: the detector should minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
- **Good localization**: the edges detected should be as close as possible to the true edges.
- **Single response**: the detector should return one point only for each true edge point; that is, minimize the number of local maxima around the true edge.



Two main approaches

The two main strategies for detecting edges are:

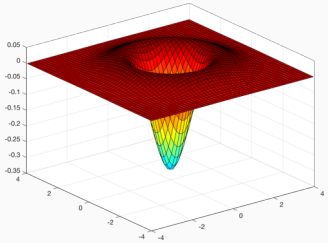
- find the **zero crossing of the 2D analogue of the second derivative**;
- find the **points where the magnitude of the gradient is maximal**.

The first approach (Marr and Hildreth, 1980) is no longer popular but historically important.

Laplacian of Gaussian approach

- A rotationally invariant 2D analogue of the second derivative is the Laplacian operator (also called the undirected second derivative):

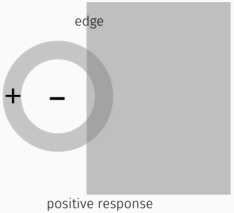
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

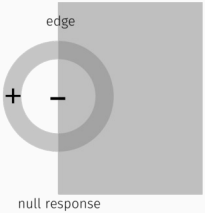


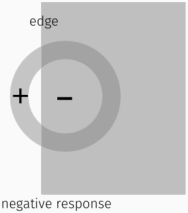
- When the smoothing is Gaussian, computing the Laplacian of the smoothed image amounts to convolving with the Laplacian of Gaussian filter (LoG filter):

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma).$$

- The LoG filter is not oriented, and has a center-surround response.

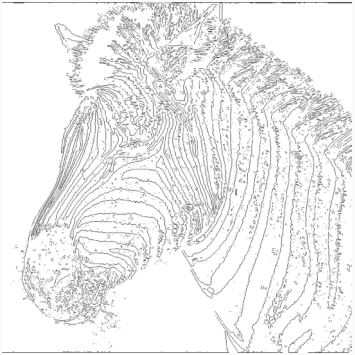




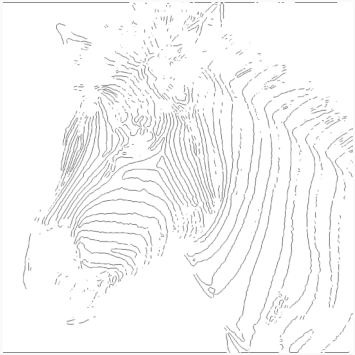


Laplacian of Gaussian approach (cont.)

Original image.



LoG edges with $\sigma = 1.5$.



LoG edges with $\sigma = 3$.

Gradient-based edge detection

- In a gradient-based edge detector, some estimate of the gradient magnitude is computed and used to determine the position of edge points.
- To get a single response, typically, the maximum magnitude is sought along the direction orthogonal to the edge.

Algorithm Gradient-based edge detection

Input: Image

Output: Edge points

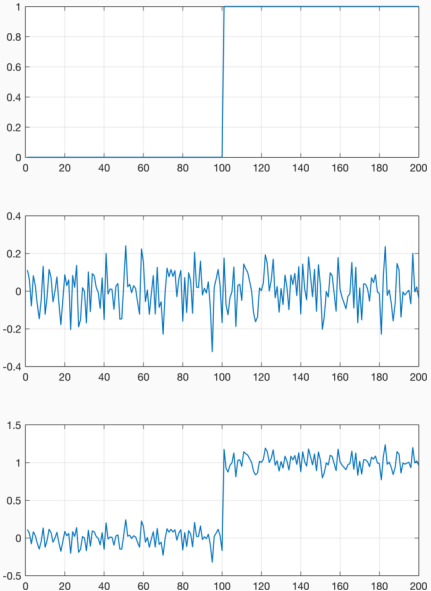
1: Form an estimate of the image gradient.
2: Obtain the gradient magnitude from this estimate.
3: Identify image points where the value of the gradient magnitude is maximal in the direction perpendicular to the edge and also large: these points are edge points.

Step 3 of the algorithm implies:

- *non-maximum suppression*;
- thresholding.

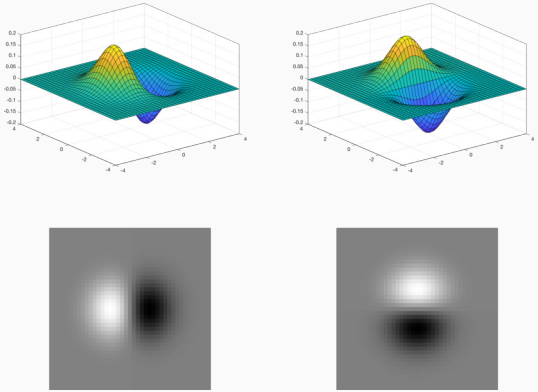
Canny edge detector

- *Canny edge detector* (Canny, 1986): probably the most widely used edge detector in computer vision;
- based on a theoretical model: **edges are modeled as step-edges corrupted by additive Gaussian noise**;
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.



Estimating the gradient

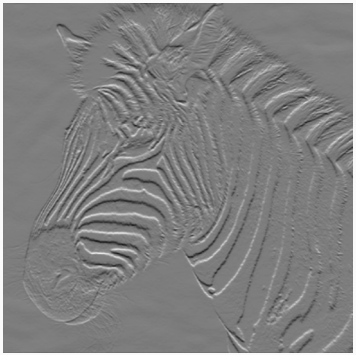
- The Canny edge detector employs the derivatives of Gaussian to estimate the gradient.
- The standard deviation σ of the Gaussian defines the bandwidth of the low-pass filter thus is related to the "scale" of edges to be detected.



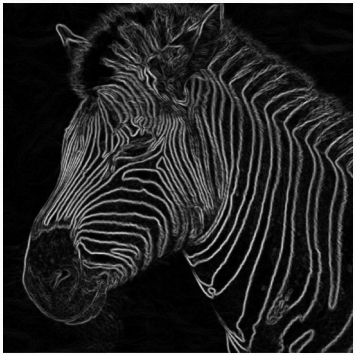
Estimating the gradient (cont.)

The gradient estimates below have been obtained with $\sigma = 1$.

$\left(\frac{\partial}{\partial x} G_\sigma\right) * f$ (x-derivative of Gaussian).



$\left(\frac{\partial}{\partial y} G_\sigma\right) * f$ (y-derivative of Gaussian).

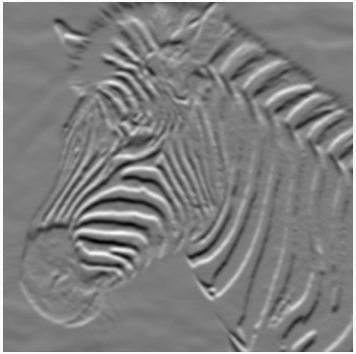


Gradient magnitude.

Estimating the gradient (cont.)

The gradient estimates below have been obtained with $\sigma = 4$.

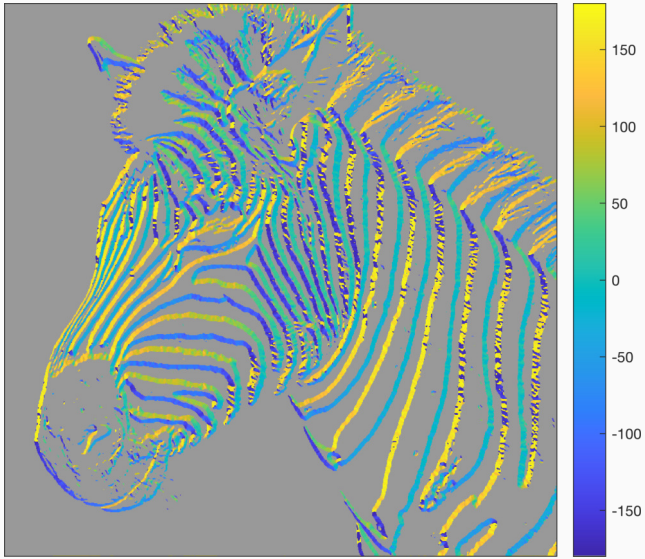
$\left(\frac{\partial}{\partial x} G_\sigma\right) * f$ (x-derivative of Gaussian).



$\left(\frac{\partial}{\partial y} G_\sigma\right) * f$ (y-derivative of Gaussian).

Gradient magnitude.

Computing orientation at each pixel

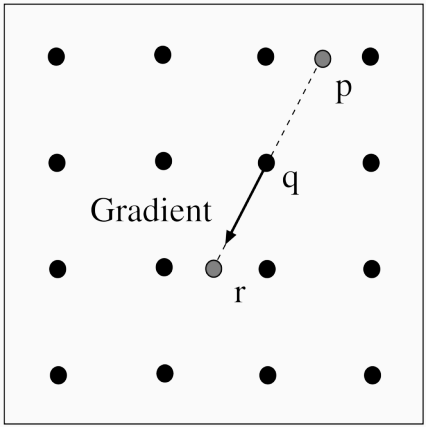


$$\theta = \text{atan2}(G_y * f, G_x * f)$$

Non-maximum suppression

Non-maximum suppression obtains points where the gradient magnitude is at a maximum along the direction of the gradient:

- consider the figure on the right, where dots represent pixel locations;
- we want to decide whether the gradient magnitude reaches a maximum at pixel q;
- find points p and r, by taking a segment passing through q, and parallel to the gradient, of a given length (which is a parameter);
- then, mark q as a maximum if its value is larger than those at both p and at r.



Notice that, in general, p and r are off the grid. Thus, we need to interpolate, for instance using bilinear interpolation of the four nearest pixels:

$$\hat{f}(x,y) = \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

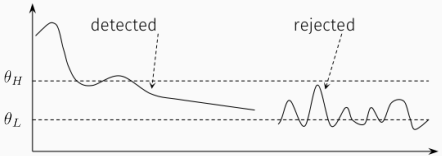
Non-maximum suppression (cont.)

Before non-max suppression.

After non-max suppression.

Hystheresis thresholding

- To get rid of the weak maxima, it is necessary to apply a threshold on the gradient magnitude.
- However, selecting an adequate value for the threshold may be difficult.
- Sometimes, a proper *global* value may even not exist (see figures).
- An effective way to alleviate the problem is *hystheresis thresholding*:
  - find **connected components**, starting from strong edge pixels;
  - use a **high threshold to start** edge curves and a **low threshold to continue** them.



Low threshold.

High threshold.

Hystheresis thresholding (cont.)

$f \geq \theta_L$.

$f \geq \theta_H$.

hyst_threshold(f, θ_L, θ_H).

Examples

Original image.

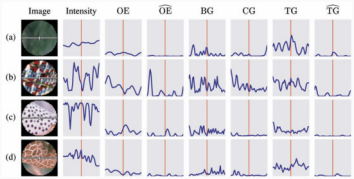
Canny edges with $\sigma = 1.5$.

Canny edges with $\sigma = 3$.

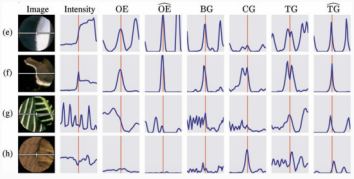
Color edge detection

- Most edge detection techniques have been developed for grayscale images;
- color images can provide additional information (for example, edges between iso-luminant colors fail to be detected by grayscale edge operators).
- Possible approaches:
  - *output fusion*: apply grayscale edge operators on each color band and combine the results to obtain the final edge map;
  - *multi-dimensional "gradient"*: compute a single estimate of the orientation and strength of an edge at a point. A possibility is to use the gradient magnitude, or the *local energy* (Morrone and Burr, 1988) of each band, sum up the results and detect the prominent pixels in the resulting map;
  - *color statistics*: estimate local color statistics in regions around each pixel (Ruzon and Tomasi, 2001; Martin et al., 2004). This has the advantage that more sophisticated techniques (e.g., 3D color histograms) can be used to compare regional statistics and that additional measures, such as texture, can also be considered (see next slide).

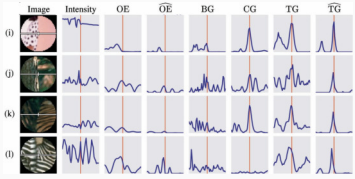
Edges and boundaries



Non-boundaries



Boundaries



Boundaries

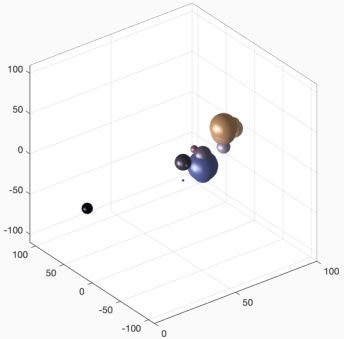
The examples above, from Martin et al. (2004) show that:

- some perceptually relevant boundaries are not edges (i.e. are not fast changes in intensity);
- some edges (fast changes in intensity) are not perceptually relevant boundaries;
- the most important example are discontinuities in texture (rows g,h,i,k).

Signatures

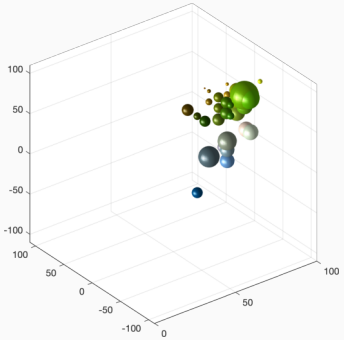
- Imagine to be given a collection of color pixels, for example pixels belonging to a certain region of the image.
- The pixel values are distributed is some way in the color space (for instance RGB).
- a possibility for describing such a distribution is using histograms, i.e. partitioning the color space in bins and counting the number of pixel values that fall within each bin.
- another possibility is using *signatures* (Rubner et al., 1998), which are essentially a generalization of histograms, in that they are a collection $\{(p_1, w_1), \ldots (p_l, w_l)\}$ of pairs of color centers p_i and a corresponding weight w_i (in other words, signature are collection of clusters).
- The number l may vary, depending on the distribution and also the cluster centers are not fixed.

Signatures (cont.)



- Example of signature for a whole image.
- Instead of using the RGB color space, the *CIE-Lab color space* (Wyszecki and Stiles, 1982), which is more "perceptually meaningful" has been used.
- The center of the spheres represent the cluster center.
- The volume of the spheres is proportional to the number of pixels belonging to that cluster.

Signatures (cont.)



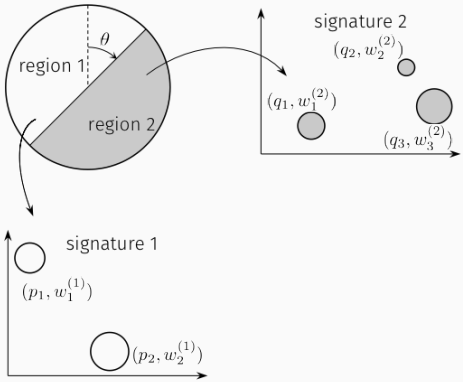
- Another example of signature for a whole image.
- Instead of using the RGB color space, the *CIE-Lab color space* (Wyszecki and Stiles, 1982), which is more "perceptually meaningful" has been used.
- The center of the spheres represent the cluster center.
- The volume of the spheres is proportional to the number of pixels belonging to that cluster.

Earth Mover's Distance (EMD)

In principle, the problem of color boundary detection could be faced as follows:

- consider a circular mask, divided into two regions;
- for a given location of the image, compute the two signatures for different orientations θ of the mask;
- for each orientation, compute the **distance between the two signatures**;
- the location is likely to belong to a boundary if a prominent local maximum of the distance over the orientation appears.

Thus, the question arises: **how to compare signatures**?



Earth Mover's Distance (EMD) (cont.)

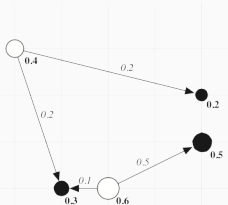
The earth mover's distance (EMD, Rubner et al. (1998)) is an effective way for comparing signatures (and histograms, too).

Take two signatures, say

$$\mathcal{S}_1 = \left\{(p_1^{(1)}, w_1^{(1)}), \ldots, (p_l^{(1)}, w_l^{(1)})\right\} \text{ and } \mathcal{S}_2 = \left\{(p_1^{(2)}, w_1^{(2)}), \ldots, (p_m^{(2}, w_m^{(2)})\right\}$$

When computing the EMD distance between $\mathcal{S}_1$ and $\mathcal{S}_2$:

- each $(p_i^{(1)}, w_i^{(1)})$ is thought of as a **pile of earth**, located in $p_i^{(1)}$ and of volume $w_i^{(1)}$;
- each $(p_j^{(2)}, w_j^{(2)})$ is thought of as a **hole**, located in $p_j^{(2)}$ and of volume $w_j^{(2)}$;
- a transportation problem (Hitchcock, 1941) is formulated: **move all the earth to the holes with minimum effort**.
- The effort is defined as $\sum f_{ij} d_{ij}$ where f_{ij} is the amount of earth moved from $p_i^{(1)}$ to $p_j^{(2)}$, and d_{ij} is the *ground distance* from $p_i^{(1)}$ to $p_j^{(2)}$.



The earth mover's distance in 2D between a signature with three centers (black) and one with two (white). Bold and italic numbers are the weights of the points and the weights moved between points, respectively.

Earth Mover's Distance (EMD) (cont.)

More formally, the transportation problem is formulated as that of finding the flows f_{ij} for minimizing the effort J:

$$\min_{f_{ij}} J = \sum_{i=1}^{l} \sum_{j=1}^{m} d_{ij} f_{ij}$$

subject to the following constraints:

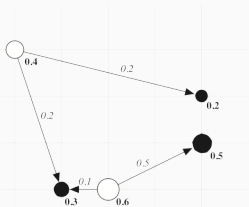
$$f_{ij} \geq 0 \qquad \forall i = 1, ..., l, \qquad \forall j = 1, ..., m$$

$$\sum_{j=1}^{m} f_{ij} \leq w_i^{(1)} \qquad \forall i = 1, ..., l,$$

$$\sum_{i=1}^{l} f_{ij} \leq w_j^{(2)} \qquad \forall j = 1, ..., m,$$

$$\sum_{i=1}^{l} \sum_{j=1}^{m} f_{ij} = \min \left(\sum_{j=1}^{m} w_j^{(2)}, \sum_{i=1}^{l} w_i^{(1)} \right).$$

The last constraint forces to move the maximum amount of earth possible.



Linear cost and linear constraints: Linear Programming problem.

Earth Mover's Distance (EMD) (cont.)

Once solved the LP problem, and found the optimal flows, the earth mover's distance is defined as the resulting effort normalized by the total flow:

$$\text{EMD} = \frac{\sum_{i=1}^{l} \sum_{j=1}^{m} d_{ij} f_{ij}}{\sum_{i=1}^{l} \sum_{j=1}^{m} f_{ij}}$$

where the flows f_{ij} are the optimal ones.

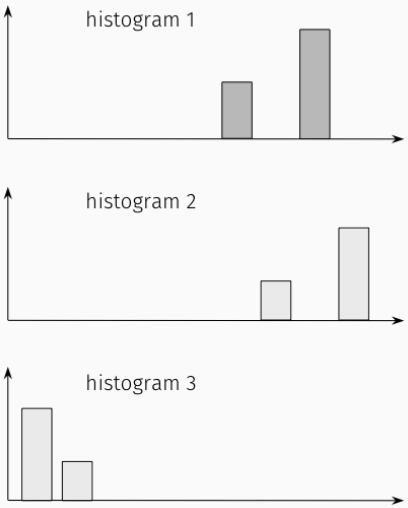
Notice that:

- The normalization is needed when the two signatures have different total weight, in order to avoid favoring smaller signatures.
- In general, the ground distance d_{ij} can be any distance and has to be chosen according to the problem at hand. For instance, when dealing with color distributions, a it could be suitable distance in a color space.

Earth Mover's Distance (EMD) (cont.)

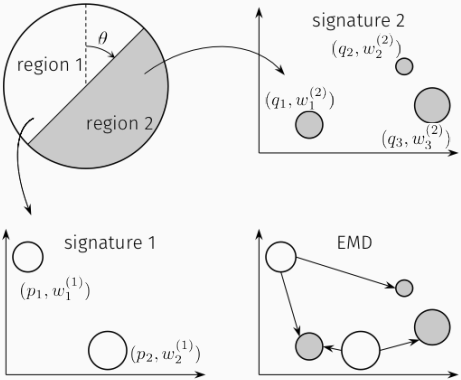
The EMD is useful also in comparing histograms:

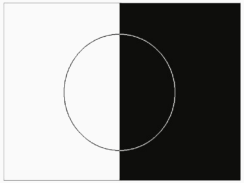
- histogram 2 and histogram 3 of the figure, have the same Euclidean distance from histogram 1;
- however, it is apparent that histogram 2 is "more similar" to histogram 1 than histogram 3 is;
- the earth mover's distance is able to capture such a similarity and provides a comparison between histograms that is sometimes more meaningful than others.

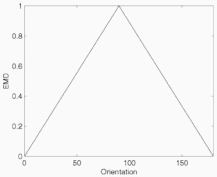


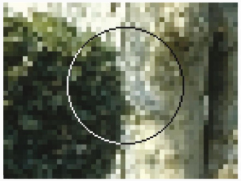
Boundary detection

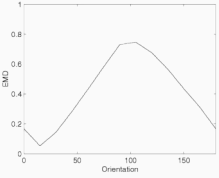
In Ruzon and Tomasi (2001) the EMD is applied to edge, junction and corner detection.





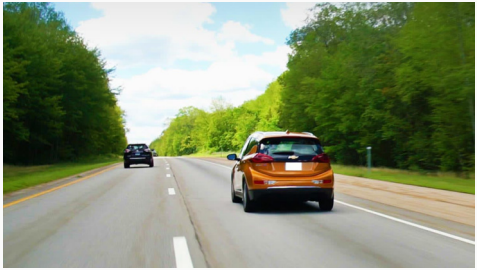






Fitting geometric primitives

Fitting

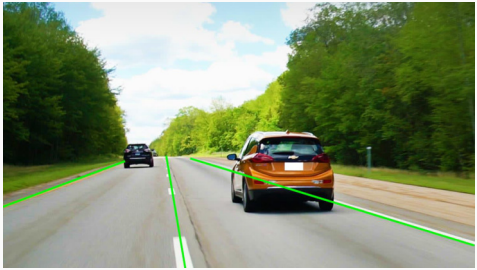


By *fitting* we mean **identifying a group of pixels that belong to the same *geometric primitive*** (for instance, a straight line). Such a geometric primitive is usually described as a parametric model, thus the goal is typically that of finding at the same time

- the **group of pixels** and
- the **parameter values**.

Fitting has obvious important applications.

Fitting



By *fitting* we mean **identifying a group of pixels that belong to the same *geometric primitive*** (for instance, a straight line). Such a geometric primitive is usually described as a parametric model, thus the goal is typically that of finding at the same time

- the **group of pixels** and
- the **parameter values**.

Fitting has obvious important applications.

Fitting as a search in parameter space

- Suppose that a parametric model has been chosen to represent the geometric entity.
- Three main questions arise:
  - What instance of model represents this set of pixels best?
  - How many model instances are there?
  - Which of several model instances gets which pixel?
- Computational complexity is important: it is unfeasible to examine every possible set of parameters and every possible combination of pixels.

Noise and clutter

Suppose we want find straight lines in an image, and suppose we have obtained an edge map as in figure.

- When multiple model instances may appear in the same image, we cannot use all the pixels of the edge map to fit a single model (because there are many model instances to be estimated);
- even in the case there is a single line to be fitted, **noise and clutter** prevent from using the whole edge map (because potentially many pixel do not belong to the line).

Voting

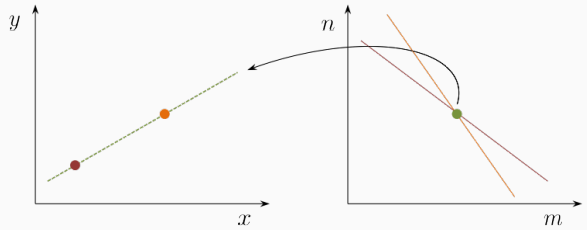
- We want to **find the subsets of pixels and the parameter values at the same time**, but...
- ...it's not feasible to check all combinations of pixels by fitting a model to each possible subset.
- *Voting* is a general technique in which each pixel "votes" for all models that are compatible with it:
  - cycle through pixels and collect votes for model parameters;
  - look for model parameters that receive more votes.
- Noise and clutter pixels will vote too, but typically their votes will be inconsistent with the majority of "good" pixels.
- using voting, there is no need to observe all the pixels belonging to a geometric entity (thus some degree of robustness to occlusions is achieved).

Hough transform

One of the most common voting techniques used in Computer Vision is the *Hough transform* (Hough, 1959), mainly employed for detecting straight lines.

The main idea is that of **transforming a line detection problem into a simple peak detection in the space of the parameters of the line**. The approach can be generalized to curves. The basic Hough transform for line detection comprises two steps.

Hough transform (cont.)

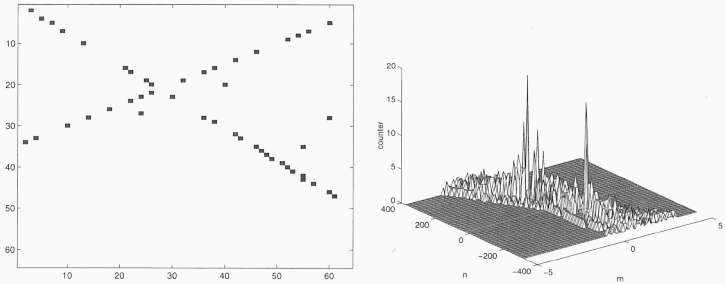


1. **Transform line detection into a line intersection problem**. Any line $y = mx + n$ is identified by a unique pair (m, n) which is a point in the parameter space (the n, m plane). Given a point $p_1 = [x_1 \quad y_1]^\top$, the family of the straight lines passing through p_1 is a line in the parameter space, of equation $n = x_1(-m) + y_1$. Given another point p_2, the family of the straight lines passing through p_2 has equation $n = x_2(-m) + y_2$. Thus, the single line passing through p_1 and p_2 is represented by the pair (m, n) satisfying both

$$\begin{cases} n = x_1(-m) + y_1 \\ n = x_2(-m) + y_2 \end{cases}$$

thus, it is the intersection of two lines in the parameter space. More generally, considering N points, we will search for the intersection of N lines in the m, n plane.

Hough transform (cont.)



Example of basic Hough transform, from Trucco and Verri (1998).

2. **Transform line intersection in a peak detection problem**. Divide the m, n into a finite grid of cells and associate an *accumulator* to each cell. Then, each point p_i votes for all the cells of the corresponding line in the parameter space. The local maxima of the accumulator map represent the lines.

Hough transform using polar representation of lines

The original ("basic") Hough transform has two problems:

- m and n can take on values in $[-\infty, \infty]$ thus it is impossible to sample the whole parameter space;
- the line bundle $x = k$ with constant k does not admit the explicit representation $y = mx + n$.

The **polar representation of lines**

$$\rho = x \cos\theta + y \sin\theta,$$

solves both problems. Here, $\rho \geq 0$ represents the distance between the image origin and the line, and $0 \leq \theta \leq 2\pi$ is the line orientation.

Hough transform using polar representation of lines

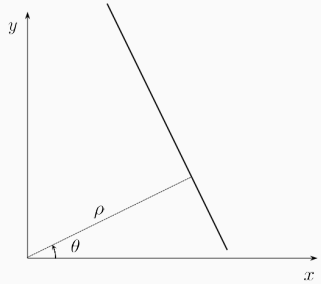
The original (“basic”) Hough transform has two problems:

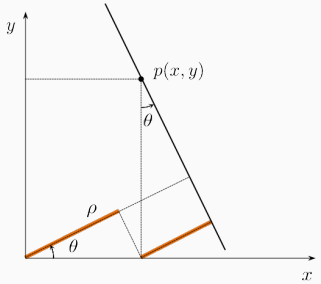
- m and n can take on values in $[-\infty, \infty]$ thus it is impossible to sample the whole parameter space;
- the line bundle $x = k$ with constant k does not admit the explicit representation $y = mx + n$.

The **polar representation of lines**

$$\rho = x \cos\theta + y \sin\theta,$$

solves both problems. Here, $\rho \geq 0$ represents the distance between the image origin and the line, and $0 \leq \theta \leq 2\pi$ is the line orientation.





Hough transform using polar representation of lines (cont.)

Given a point $p_1 = [x_1 \quad y_1]^\top$, the family of the straight lines passing through p_1 has the following equation, in the θ, ρ plane:

$$\rho = x_1 \cos\theta + y_1 \sin\theta.$$

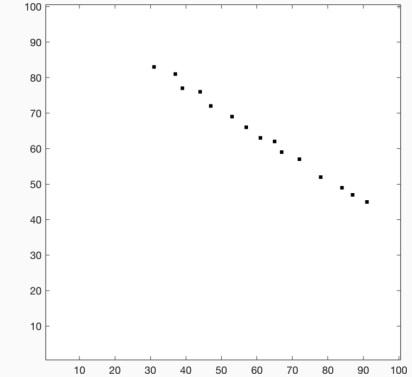
Thus, with the new representation, the family of the straight lines through a point is represented by a **sinusoid in parameter space**. As a consequence, when voting, each point increases the counter of all cells corresponding to a sinusoid.

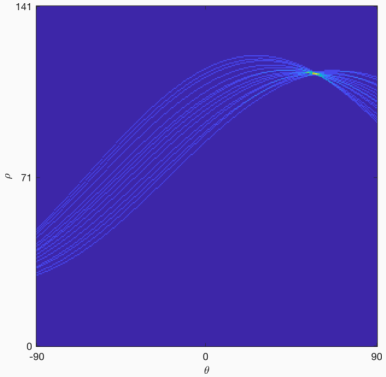
Hough transform using polar representation of lines (cont.)

Given a point $p_1 = [x_1 \quad y_1]^\top$, the family of the straight lines passing through p_1 has the following equation, in the θ, ρ plane:

$$\rho = x_1 \cos\theta + y_1 \sin\theta.$$

Thus, with the new representation, the family of the straight lines through a point is represented by a **sinusoid in parameter space**. As a consequence, when voting, each point increases the counter of all cells corresponding to a sinusoid.



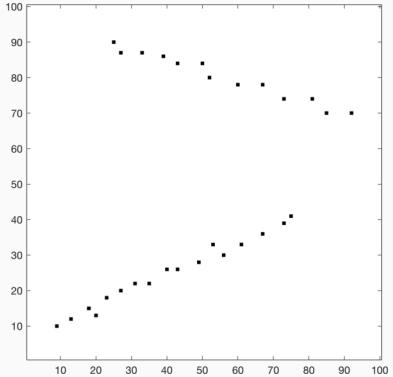


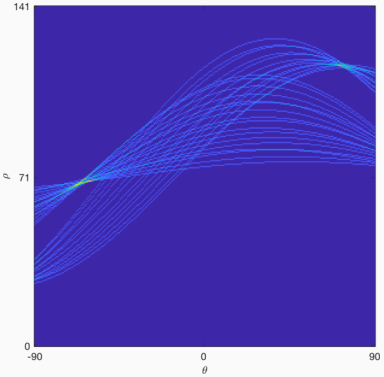
Hough transform using polar representation of lines (cont.)

Given a point $p_1 = [x_1 \quad y_1]^\top$, the family of the straight lines passing through p_1 has the following equation, in the θ, ρ plane:

$$\rho = x_1 \cos\theta + y_1 \sin\theta.$$

Thus, with the new representation, the family of the straight lines through a point is represented by a **sinusoid in parameter space**. As a consequence, when voting, each point increases the counter of all cells corresponding to a sinusoid.





Practical problems with the Hough transform

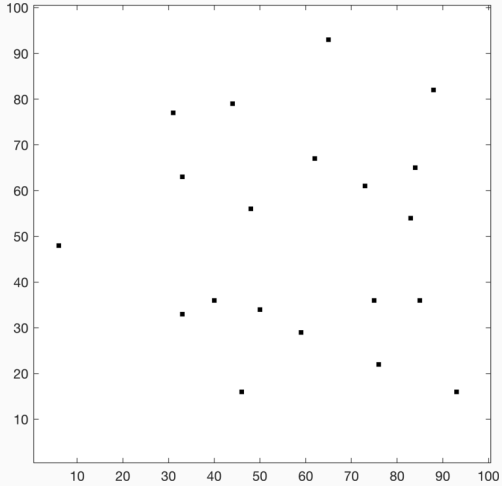
The Hough transform has some significant practical problems:

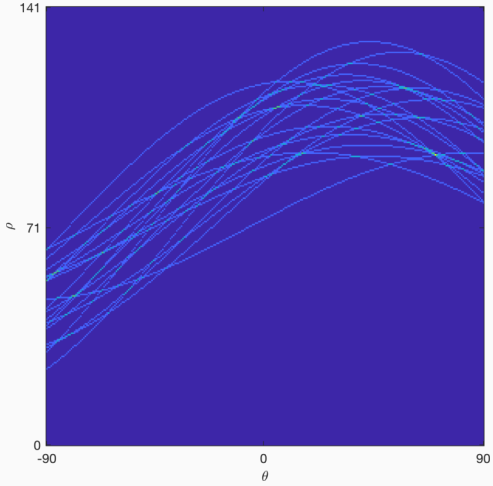
- **Quantization errors**: it is difficult to choose an appropriate grid:
  - too coarse a grid results in false voting (many different lines correspond to the same cell);
  - too fine a grid can lead to lines not being found, because votes are spread among different cells corresponding to similar lines.
- **Problems with noise**: the Hough transform has the advantage of connecting widely separated pixels belonging to the same straight line: this, however, may become a weakness when clutter and noise are present. Even a set of uniformly distributed pixel may lead to large peaks (see figure in next slide).

However, the Hough transform is widely used for finding lines in sets of edge points, especially in well-adapted problems, i.e. when a priori knowledge on the domain can be exploited to:

- get rid of irrelevant pixels;
- choose the grid carefully.

Counterexample





Hough transform for detecting circles

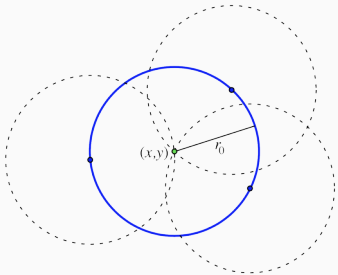
Let a circle be described by the triplet (x_c, y_c, r) where (x_c, y_c) is the center point and r the radius. Thus the parameter space is three-dimensional.

A given a point $p(x, y)$ should vote for all the circles passing through it.

What are the triplets corresponding to the family of all the circles through p?

- The circle of radius $r = 0$ and center (x, y) gives the triplet $(x, y, 0)$;
- the circles of generic radius r_0 passing through p have center in a circle of radius r_0 centered in (x, y);
- as a result, **the whole family of circles through $p(x, y)$ is the surface of the straight circular cone defined by (x, y) in the parameter space**.

Hence, each point votes for the cells belonging to a surface of a cone.



Generalized Hough transform

The Hough transform is easily generalized to detect curves $y = f(x, a)$ where $a = [a_1, \ldots, a_P]^\top$ is a vector of P parameters.

Algorithm Generalized Hough transform

Input: A pixel map $E(i, j)$

Output: A set of vectors $a_{(1)}, \ldots a_{(N)}$ describing the curve instances detected in E

1: Discretize the intervals of variation of $a_1 \ldots, a_P$. Let $s_1, \ldots, s_P$ be the number of discrete intervals for each parameter.
2: Let A be an $s_1 \times s_2 \times \cdots \times s_P$ array of integer counters.
3: For each pixel $E(i, j)$ such that $E(i, j) = 1$, increment all counters on the curve defined by $y = f(x, a)$ in A[3]. .
4: Find all local maxima $a_{(1)}, \ldots a_{(N)}$ of A above a certain threshold.

[3] all the p-tuples $a_1 \ldots, a_P$ satisfying $y = f(x, a)$ for the pair (x, y) corresponding to (i, j)

Observations

- Sometimes, a priori knowledge can be exploited to restrict the search to a subset of the parameter space. For instance:
  - **lines with bounded orientation**: if we know that the sought line is such that $\theta_{min} \leq \theta \leq \theta_{max}$, then the voting and the search for the peaks can be restricted to the corresponding region of the parameter space;
  - **pixels with associated orientation**: if the pixels $p_i(x_i, y_i)$ come with an associated orientation θ_i (a quite common case since edge detection implies gradient computation) then each pixel votes for one cell only, precisely the one corresponding to the pair (ρ, θ_i) where $\rho = x_i \cos\theta_i + y_i \sin\theta_i$;
  - **known radius**: when detecting circles, if the radius of the sought circles is known in advance, the parameter space becomes two-dimensional (and each pixel votes for the cells belonging to a circle).
- Due to the discretization of the parameter space, the detected geometric primitive may be an imperfect fit. Thus (depending on the application), a refinement may be necessary, for instance using some of the techniques described in the following.

Line fitting with least squares

Suppose we are given a set of points $\mathcal{S} = \{(x_i, y_i), i = 1 \dots N\}$. A traditional technique for fitting a straight line to this set is the *least squares*.

We represent a line as $y = px + q$ and we solve the unconstrained minimization problem

$$\min_{p,q} \sum_i r_i^2,$$

where the terms $r_i = y_i - (px_i + q)$ are called the *residuals*. Let $z = [p \quad q]^\top$, $b = [y_1 \dots y_N]^\top$ and

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}.$$

Then, the problem can be restated as

$$\min_z \|Az - b\|,$$

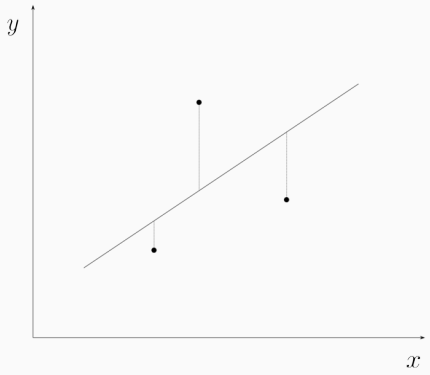
whose solution in well-known to be

$$z = (A^\top A)^{-1} A^\top b.$$

Line fitting with least squares (cont.)

However, this standard linear solution is not suitable for Computer Vision applications:

- the measurement error is dependent on the coordinate frame (it is sum of the squared **vertical** offsets from each point to the line);
- vertical lines lead to quite large values of the error, and can produce geometrically poor fits;
- on the other hand, rotating the reference frame leads do different results (i.e. different lines) for the same set of points, which is not desirable.
- the reason is that the underlying model of the least squares is that the pairs (x_i, y_i) are the outcome of a measurement process, in which only y (the dependent variable) is corrupted by noise.



Total least squares

A better approach is to **use the actual distance between the point and the line** (instead of the vertical offset). This leads to the *total least squares problem*.

If we represent a line as

$$ax + by + c = 0,$$

the distance from a point (u, v) to the line is given by

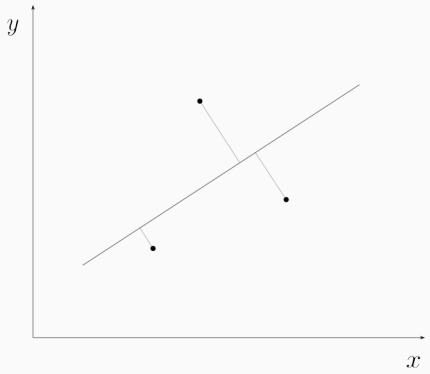
$$|au + bv + c| \qquad \text{if} \quad a^2 + b^2 = 1.$$

Thus, the following **constrained** minimization problem can be formulated:

$$\min_{a,b,c} \sum_i (ax_i + by_i + c)^2$$

s.t.

$$a^2 + b^2 = 1$$



Total least squares (cont.)

By introducing the Lagrange multiplier λ, we get the Lagrangian

$$\mathcal{L}(a, b, c, \lambda) = \sum_i (ax_i + by_i + c)^2 + \lambda(a^2 + b^2 - 1).$$

By imposing the stationarity conditions we obtain

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial a} = 2\displaystyle\sum_i x_i(ax_i + by_i + c) + 2\lambda a = 0 \\ \dfrac{\partial \mathcal{L}}{\partial b} = 2\displaystyle\sum_i y_i(ax_i + by_i + c) + 2\lambda b = 0 \\ \dfrac{\partial \mathcal{L}}{\partial c} = 2\displaystyle\sum_i (ax_i + by_i + c) = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = a^2 + b^2 - 1 = 0 \end{cases}$$

The third equation states that

$$c = -\frac{a}{N}\sum_i x_i - \frac{b}{N}\sum_i y_i.$$

Total least squares (cont.)

By substituting for c in the first two equations we get

$$\begin{cases} \sum_i x_i(ax_i + by_i - \frac{a}{N}\sum_i x_i - \frac{b}{N}\sum_i y_i) + \lambda a = 0 \\ \sum_i y_i(ax_i + by_i - \frac{a}{N}\sum_i x_i - \frac{b}{N}\sum_i y_i) + \lambda b = 0 \end{cases}$$

Simple computations lead to:

$$\begin{bmatrix} \overline{x^2} - \overline{x}\,\overline{x} & \overline{xy} - \overline{x}\,\overline{y} \\ \overline{xy} - \overline{x}\,\overline{y} & \overline{y^2} - \overline{y}\,\overline{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = -\lambda \begin{bmatrix} a \\ b \end{bmatrix}$$

where we use the notation

$$\overline{u} = \frac{\sum_i u_i}{N}.$$

Total least squares (cont.) (cont.)

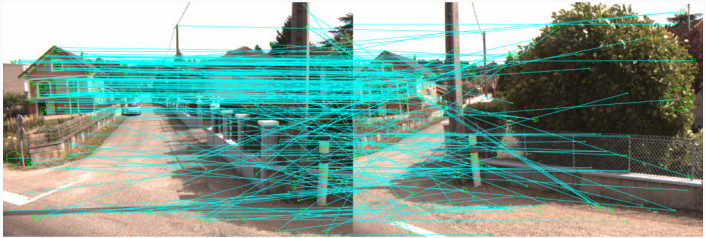
$$\underbrace{\begin{bmatrix} \overline{x^2} - \overline{x}\,\overline{x} & \overline{xy} - \overline{x}\,\overline{y} \\ \overline{xy} - \overline{x}\,\overline{y} & \overline{y^2} - \overline{y}\,\overline{y} \end{bmatrix}}_{\doteq M} \begin{bmatrix} a \\ b \end{bmatrix} = -\lambda \begin{bmatrix} a \\ b \end{bmatrix}$$

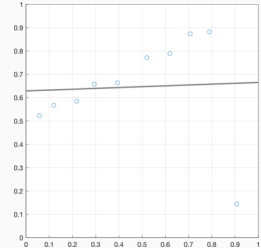
Observe that:

- it is an eigenvalue problem, since the solution $[a \quad b]^\top$ is an eigenvector of the matrix M, associated to the eigenvalue $-\lambda$.
- M is 2×2, thus two solutions up to a scale factor can be obtained in closed form. The scale is obtained from the constraint $a^2 + b^2 = 1$.
- M is symmetric, thus its eigenvectors are orthogonal. As a consequence, the two solutions to the problem are lines at right angles, one of which corresponds to the minimum.

Robustness

- Squared error terms provide nice analytic treatment but at the price of **high sensitivity to outliers**.
- Indeed, in typical Computer Vision applications, a data point may be "wrong" for two main reasons:
  - it is corrupted by **measurement noise**. For a reasonable amount of well-behaved noise (Gaussian, zero mean, not too high variance), squared error terms behave nicely (if the noise is Gaussian, the least squares is actually optimal in that it minimizes the variance of the residuals);
  - it is the result of a **wrong match** between features. Wrong matches are particularly prone to generating wild outliers (as in left figures below).
- Squared error terms are very sensitive to outliers. Even a single outlier can ruin a least squares estimate (see below, right).





M-estimators

M-estimators are among the most used robust regression techniques. The basic idea is that of **using a sub-quadratic loss function**, in order to reduce the influence of the outliers. In other words, denoting by r_i the i-th residual, instead of the problem

$$\min \sum_i r_i^2,$$

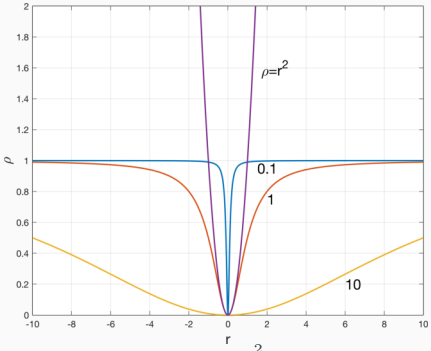
we formulate a problem

$$\min \sum_i \rho(r_i),$$

where ρ is a symmetric and non-negative function having a unique global minimum in $\rho(0) = 0$. A common family of such functions is:

$$\rho(r) = \frac{r^2}{k^2 + r^2},$$

where the parameter k controls the point at which the function flattens out.



Functions of type $\rho(r) = \frac{r^2}{k^2 + r^2}$, for $k = 0.1, 1, 10$ compared to the quadratic loss function.

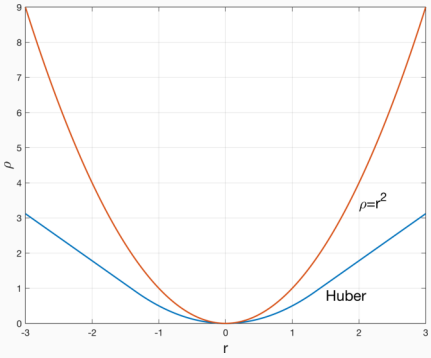
M-estimators (cont.)

Another of such functions is the *Huber loss function*:

$$\rho_H(r) = \begin{cases} \frac{1}{2}r^2 & \text{if} \quad |r| \leq k \\ k|r| - \frac{1}{2}k^2 & \text{if} \quad |r| > k \end{cases}$$

where the parameter k controls the point at which the function changes from quadratic to linear.

M-estimators lead to a **nonlinear minimization problem** that can be solved iteratively using the *Iteratively Reweighted Least Squares* method (IRLS). For details, see Section 6.1.4 of Szeliski (2010).



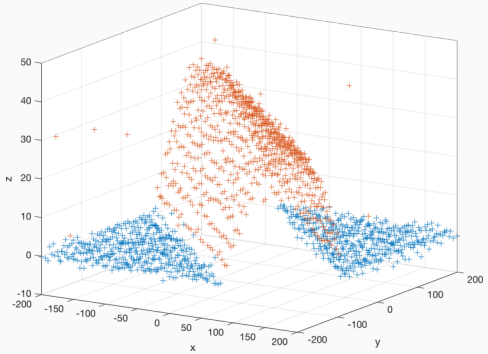
Huber loss function for $k = 1.34$, compared to the quadratic loss function.

Example

- Task: fitting a parametric surface to the orange points in figure.
- Although some outliers are present, it is rather clear how a good fitting surface should look.
- Based on a priori knowledge the model is chosen as

$$z(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} p_{ij} x^i y^j,$$

thus the parameters to be determined are p_{ij} $i, j \in \{0, 1, 2\}$.

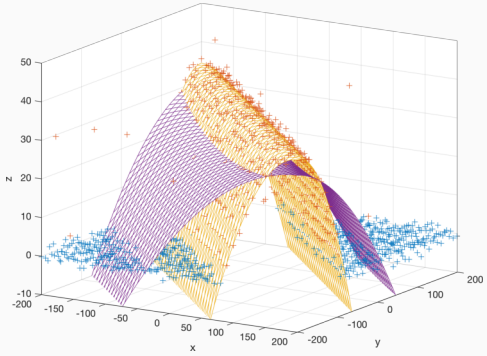


Example

- Task: fitting a parametric surface to the orange points in figure.
- Although some outliers are present, it is rather clear how a good fitting surface should look.
- Based on a priori knowledge the model is chosen as

$$z(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} p_{ij} x^i y^j,$$

  thus the parameters to be determined are p_{ij} $i,j \in \{0,1,2\}$.
- Violet surface: least squares fitting.
- Yellow surface: robust fitting obtained through IRLS using Huber loss function.

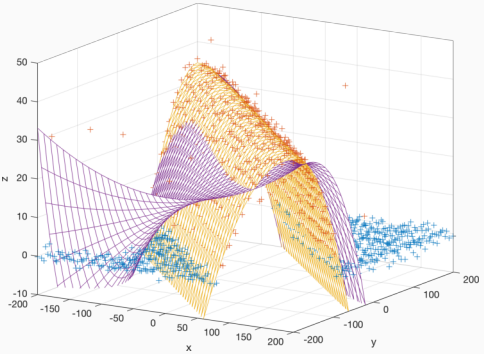


Example (cont.)

- If we consider an unnecessarily more expressive model:

$$z(x,y) = \sum_{i=0}^{2} \sum_{j=0}^{3} p_{ij} x^i y^j,$$

the behavior of the least squares is even worst, while the robust method is capable of producing an acceptable fit.



RANdom SAmple Consensus (RANSAC)

The random sample consensus (RANSAC, Fischler and Bolles (1981)) is a method widely used on Computer Vision for fitting geometrical primitives.

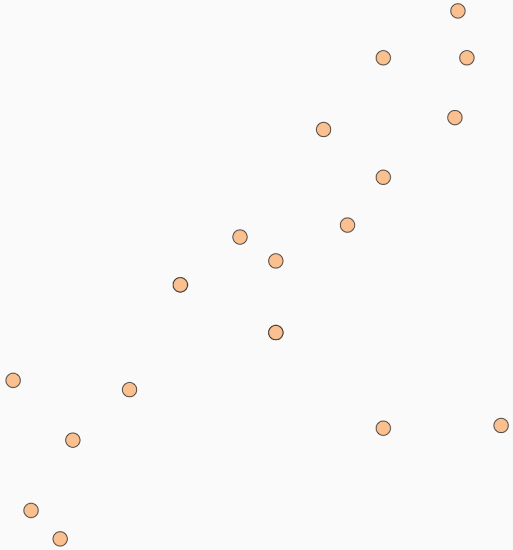
- Approach: outliers can ruin the fit, thus look for “inliers” and use only those.
- RANSAC is not a fitting technique per se. Instead, it relies on a fitting technique (for instance, least squares) and provides a mechanism for selecting inliers.
- Intuition: a random **minimal** sample containing outliers is likely to provide an estimate having poor consensus (agreement with remaining data).

Sketch of the RANSAC loop:

1. randomly draw a minimal sample from the set of data;
2. compute the fit based on the selected sample;
3. find inliers (data points that are well explained by the computed primitive);
4. repeat steps 1-3 until the number of inliers is sufficiently large;
5. perform a final fit on the inliers only.

Line fitting example

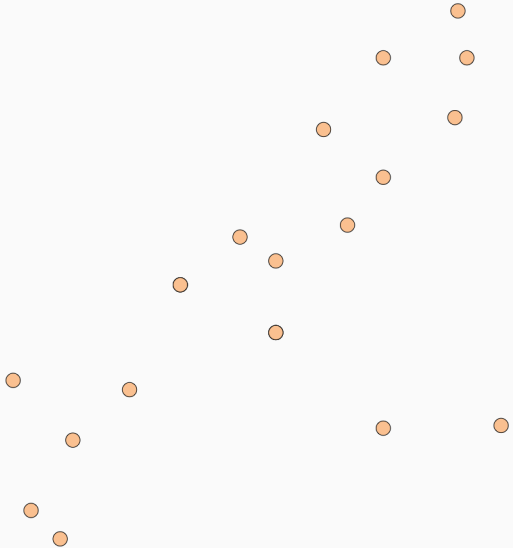
Suppose we want to fit a line to the points in figure.



Line fitting example

Suppose we want to fit a line to the points in figure.

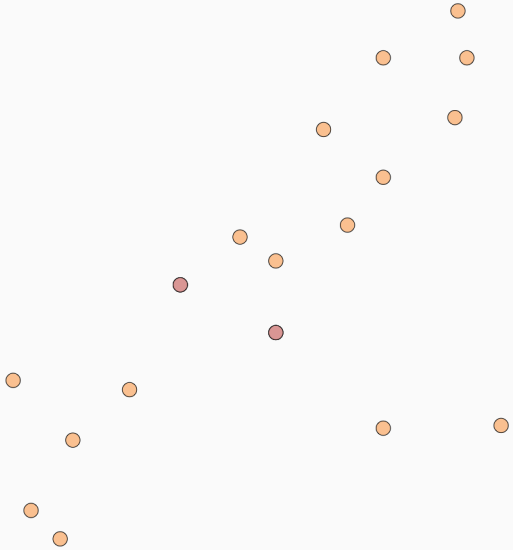
- The minimum number of points to estimate a line is two;



Line fitting example

Suppose we want to fit a line to the points in figure.

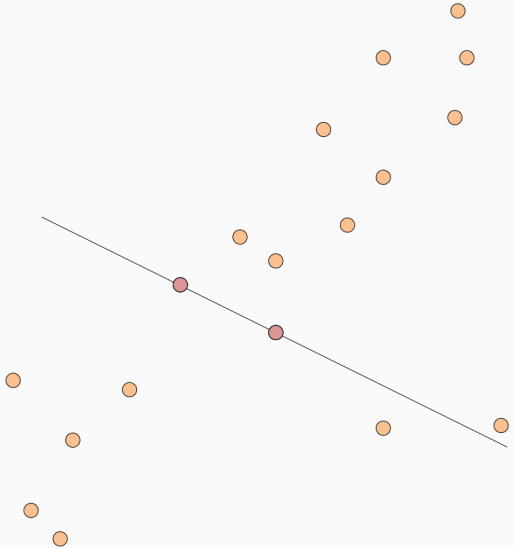
- The minimum number of points to estimate a line is two;
- sample two points;



Line fitting example

Suppose we want to fit a line to the points in figure.

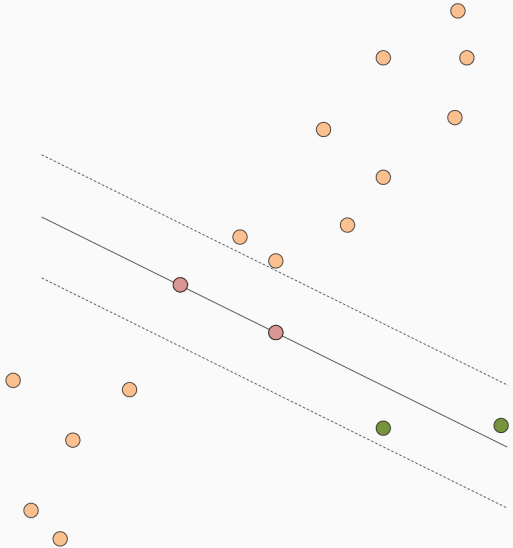
- The minimum number of points to estimate a line is two;
- sample two points;
- fit a line;



Line fitting example

Suppose we want to fit a line to the points in figure.

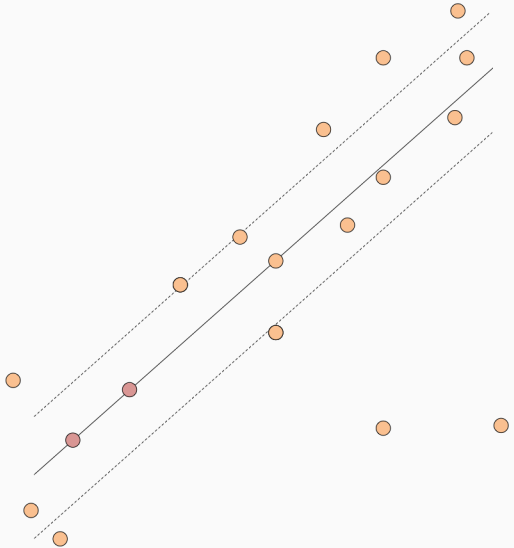
- The minimum number of points to estimate a line is two;
- sample two points;
- fit a line;
- find the total number of points that agree with the line (i.e. lie within a threshold);



Line fitting example

Suppose we want to fit a line to the points in figure.

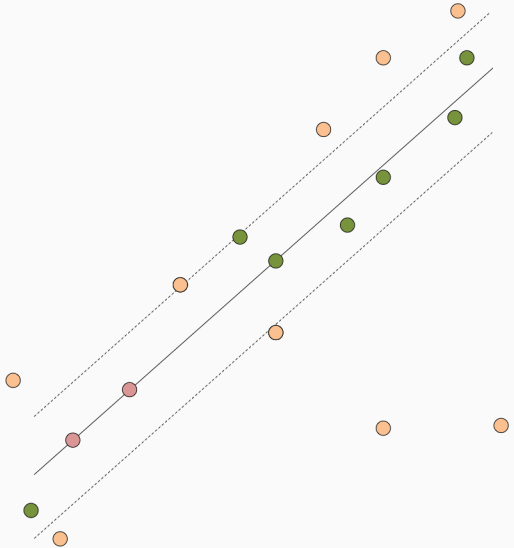
- The minimum number of points to estimate a line is two;
- sample two points;
- fit a line;
- find the total number of points that agree with the line (i.e. lie within a threshold);
- repeat sampling and fitting ...



Line fitting example

Suppose we want to fit a line to the points in figure.

- The minimum number of points to estimate a line is two;
- sample two points;
- fit a line;
- find the total number of points that agree with the line (i.e. lie within a threshold);
- repeat sampling and fitting ...
- ...until a good result is obtained.



RANSAC paradigm

Quoting from the original paper (Fischler and Bolles, 1981):

> *The RANSAC paradigm is formally stated as follows:*
>
> *Given a model that requires a minimum of n data points to instantiate its free parameters, and a set of data points $\mathcal{P}$ such that the number of points in $\mathcal{P}$ is greater than n ($\#(\mathcal{P}) > n$), randomly select a subset $\mathcal{S}_1$ of n data points from $\mathcal{P}$ and instantiate the model. Use the instantiated model $\mathcal{M}_1$ to determine the subset $\mathcal{S}_1^*$ of points in $\mathcal{P}$ that are within some error tolerance of $\mathcal{M}_1$. The set $\mathcal{S}_1^*$ is called the* consensus set *of $\mathcal{S}_1$.*
>
> *If $\#(\mathcal{S}_1^*)$ is greater than some threshold t, which is a function of the estimate of the number of gross errors in $\mathcal{P}$, use $\mathcal{S}_1^*$ to compute (possibly using least squares) a new model $\mathcal{M}_1^*$.*
>
> *If $\#(\mathcal{S}_1^*)$ is less than t, randomly select a new subset $\mathcal{S}_2$ and repeat the above process. If, after some predetermined number of trials, no consensus set with t or more members has been found, either solve the model with the largest consensus set found, of terminate in failure.*

A RANSAC algorithm

A variant of the original RANSAC consist of **performing an exact number of iterations** k (instead of a maximum number). The following variant is from **?**.

Algorithm Fitting lines using RANSAC

Input: the smallest number of points required (n), the number of iterations required (k), the threshold used to identify a point that fits well (t), the number of nearby points required to assert a model fits well (d).

Output: the parameters of the fitting line.

1: **repeat**
2: Draw a sample of n points from the data uniformly and at random
3: Fit a line to that set of n points
4: **for all** data points outside the sample **do**
5: Compute the distance from the point to the line; if it is is less than t, the point is close
6: **end for**
7: **if** there are d or more points close to the line **then**
8: Refit the line using all these points.
9: **end if**
10: **until** k iteration have occurred
11: Output the best fit from this collection.

How many iterations?

Q: Why it is convenient to use a fixed number k of iterations?

A: Because through k we can control the probability of failure.

Q: How many sampling iterations k are needed for obtaining a certain probability of failure?

A: See below.

Suppose that:

- the data contains a fraction w of inliers (points belonging to the line, possibly with some small noise).
- the model requires n points;
- we perform k iterations.

We can easily express the probability z of failure (the probability that none of the k samples contains inliers only) as a function of k. Indeed, the probability that a single sample is correct is w^n and the probability that all samples are wrong is $(1 - w^n)^k$.

Thus,

$$z = (1 - w^n)^k,$$

meaning that if we fix a desired failure rate $\bar{z}$, we should choose k higher than

$$\bar{k} = \frac{\log(\bar{z})}{\log(1 - w^n)}.$$

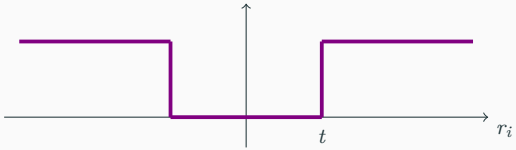
How many iterations? (cont.)

In the table, the minimum number of iterations as function of the sample size n and the fraction of outliers $1 - w$ is reported, for $z = 0.01$.

Sample size	Fraction of outliers $(1 - w)$						
n	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

RANSAC as an M-estimator

- As pointed out in Stewart (1999), RANSAC can be viewed as a particular M-estimator.
- The objective function that RANSAC maximizes is the number of data points having absolute residuals smaller than a predefined value t. This is equivalent to minimizing a binary loss function that is zero for small (absolute) residuals, and 1 for large absolute residuals r_i, with a discontinuity at t.



References

References

Beaudet, P. R. (1978). Rotationally invariant image operators. In *Proc. 4th Int. Joint Conf. Pattern Recog, Tokyo, Japan, 1978*.

Brown, M., Szeliski, R., and Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517. IEEE.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Freeman, W. T., Adelson, E. H., et al. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.

Hitchcock, F. L. (1941). The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20:224–230.

Hough, P. V. (1959). Machine analysis of bubble chamber pictures. In *Conf. Proc.*, volume 590914, pages 554–558.

References (cont.)

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proc. R. Soc. Lond. B*, 207(1167):187–217.

Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–49.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72.

Morrone, M. and Burr, D. (1988). Feature detection in human vision: a phase dependent energy model. *Proceedings of the Royal Society of London*, B235:221–245.

Noble, J. A. (1988). Finding corners. *Image and vision computing*, 6(2):121–128.

Rubner, Y., Tomasi, C., and Guibas, L. J. (1998). A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE.

References (cont.)

Ruzon, M. and Tomasi, C. (2001). Edge, junction, and corner detection using color distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1281–1295.

Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, number June, pages 593–600. IEEE.

Stewart, C. V. (1999). Robust parameter estimation in computer vision. *SIAM review*, 41(3):513–537.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Trucco, E. and Verri, A. (1998). *Introductory techniques for 3-D computer vision*. Prentice Hall Englewood Cliffs.

Witkin, A. (1984). Scale-space filtering: A new approach to multi-scale description. *ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 9:150–153.

Wyszecki, G. and Stiles, W. S. (1982). *Color science*, volume 8. Wiley New York.

Yan Ke and Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 506–513, Washington, DC.

554SM –Fall 2018

Lecture 4
Feature Detection

END