



Master in Computer Vision *Barcelona*

Module: Introduction to human and computer vision

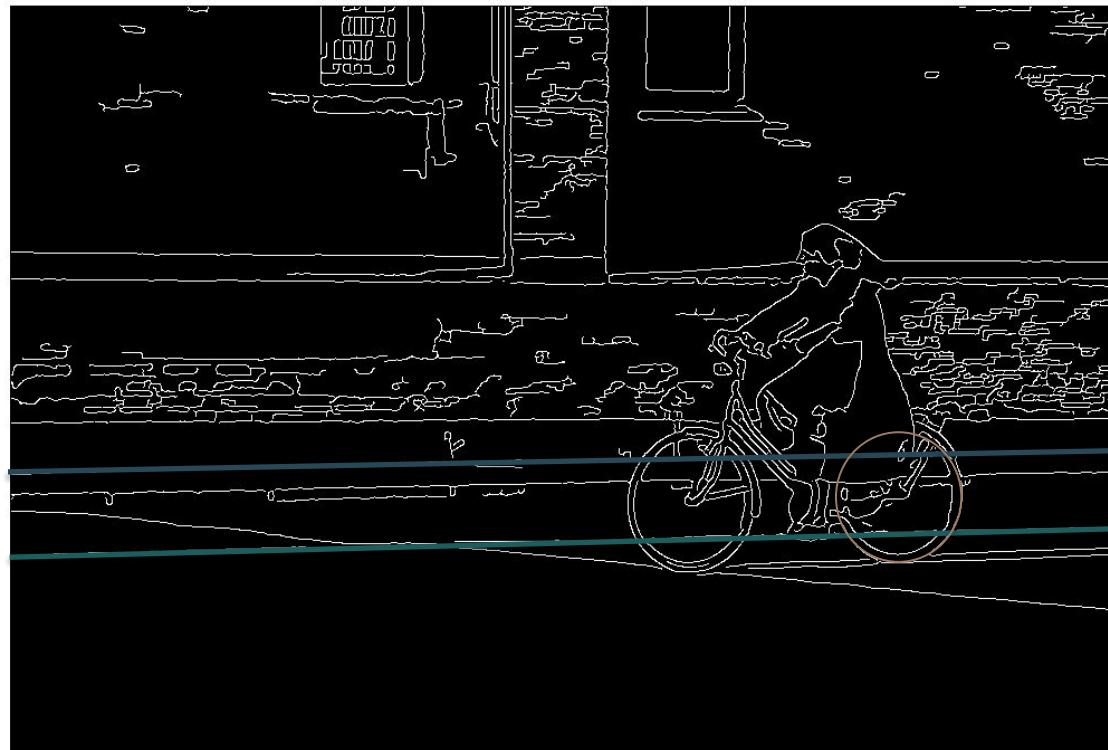
Lecture 9: Grouping, segmentation and classification (I)

Lecturer: Ramon Morros

GROUPING & MODELING: SHAPE

Shape

- Objective: find simple shapes in images (lines, circles, etc.)



Shape

- Objective: find simple shapes in images (lines, circles, etc.)
- Shapes that can be parametrized

Line

$$y = ax + b$$

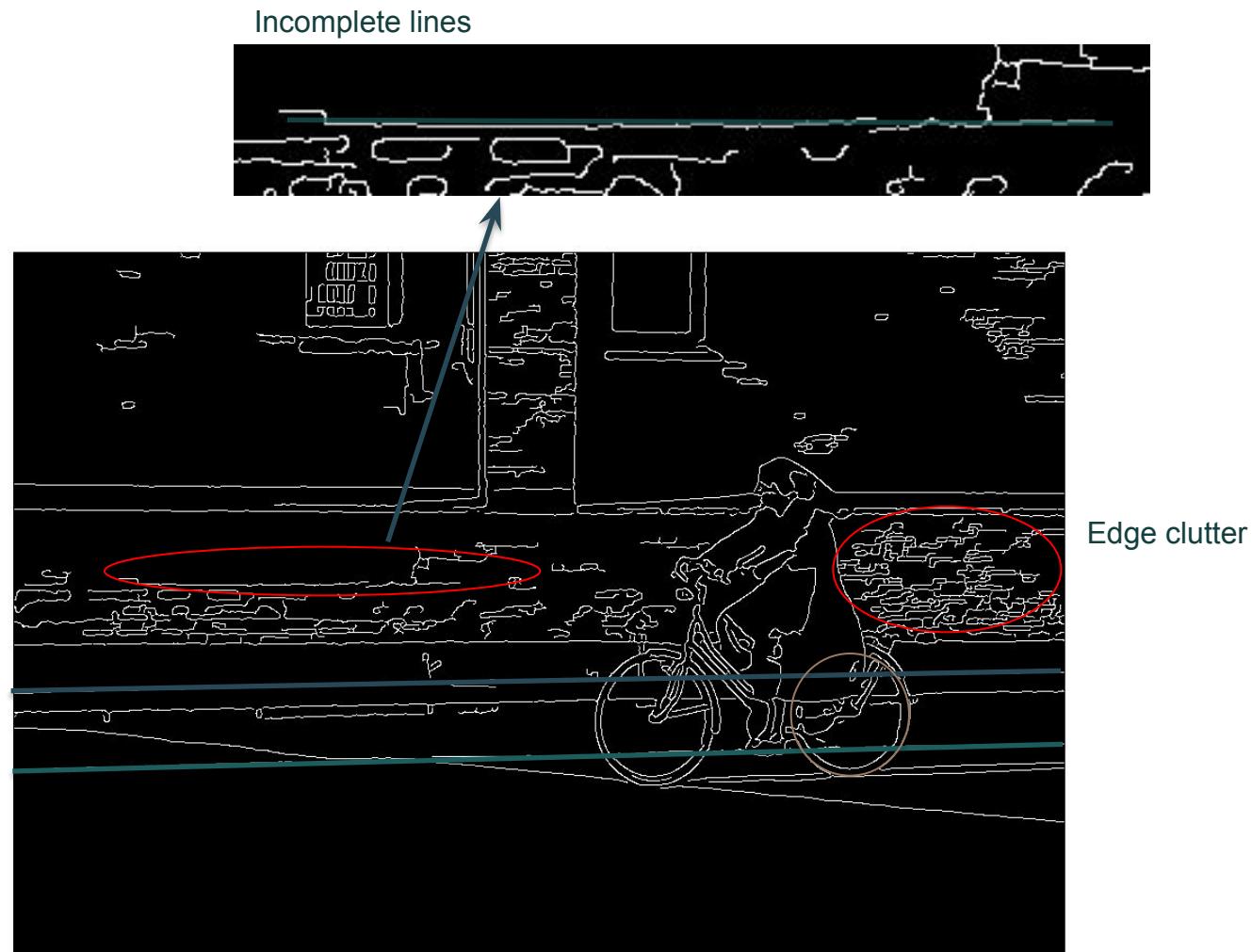
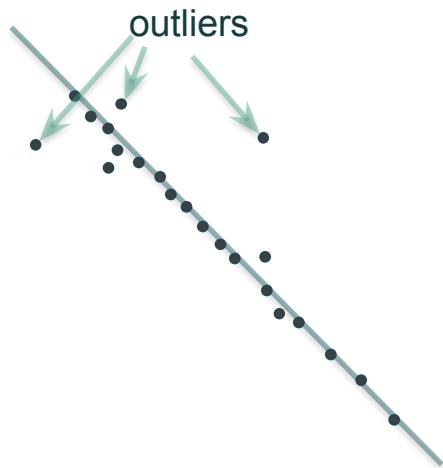
Circle

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$



Shape

Problems:



Outliers: points that do not fit to the model within a given small error

Shape

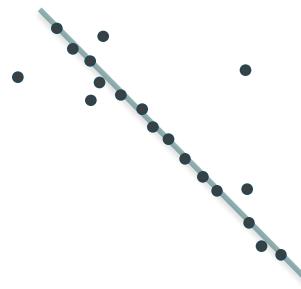
Canny, Pb, ...
(L8)

- Method
 - Obtain the data: edge detection, or another method to generate shape from images
 - Adjust data to a model / group the features
- Techniques
 - Linear regression: least squares
 - Hough transform : voting in parameter space
 - **RANSAC** : try randomly subsets of points to adjust the shape

Shape: Least Squares

- Goal: find the parameters that best fit a set of data to a model
- Over-determined system : more equations than parameters!

Example: line



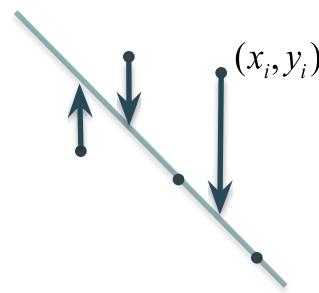
Line equation:

$$y = ax + b$$

$a?$, $b?$

- Method: minimize the sum of squared residuals

$$E = \sum_{i=0}^N (y_i - ax_i + b)^2$$



Shape: Least Squares

- Minimization: $\mathbf{p} = \underset{p \in \Re^2}{\operatorname{argmin}} \{E\}$ $\mathbf{p} = [a \ b]$

Energy

$$E = \sum_{i=0}^N (y_i - ax_i - b)^2$$

coordinates of contour points

In matrix form:

$$E = \sum_{i=0}^N \left([x_i \ 1] \begin{bmatrix} -a \\ -b \end{bmatrix} + y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{bmatrix} -a \\ -b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$
$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

Minimization:

$$\frac{dE}{dp} = -2\mathbf{A}^T \mathbf{y} + 2\mathbf{A}^T \mathbf{Ap} = 0$$

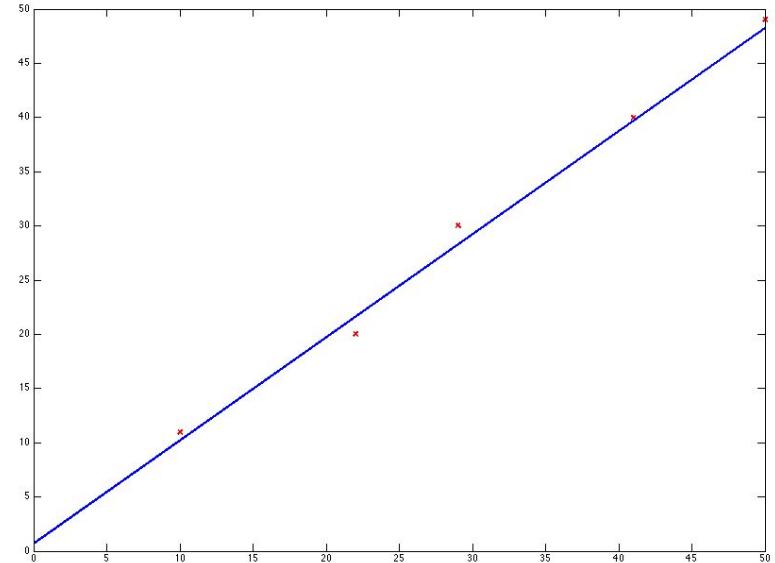
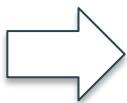
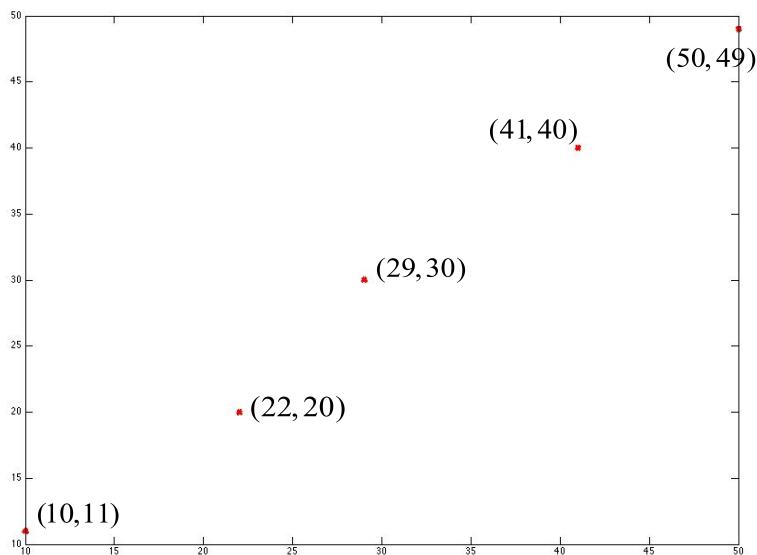
$$\mathbf{p} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{y} = \mathbf{A}^{-1} \mathbf{y}$$

Python:
`p = np.linalg.pinv(A)*y`

Source: S. Lazebnik, J. Hays

Shape: Least Squares

- Example: Line



$$\mathbf{A} = \begin{bmatrix} 10 & 1 \\ 22 & 1 \\ 29 & 1 \\ 41 & 1 \\ 50 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 11 \\ 20 \\ 30 \\ 40 \\ 49 \end{bmatrix}$$

$$p = A^{-1}y = [0.96, 0.69]$$

Shape: Least Squares

- Strengths:

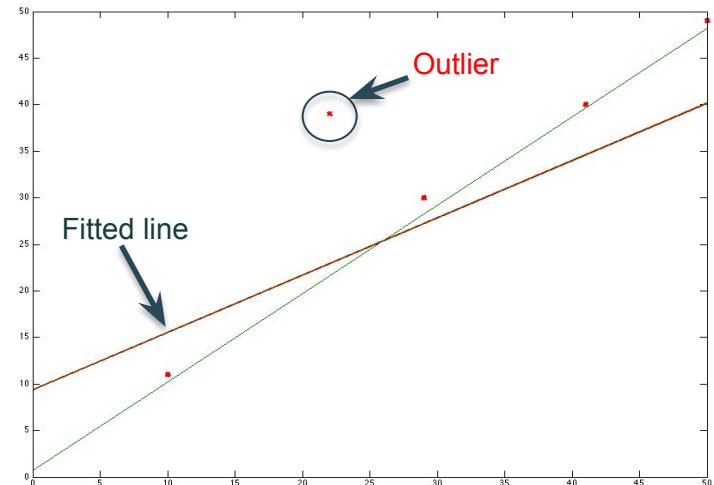
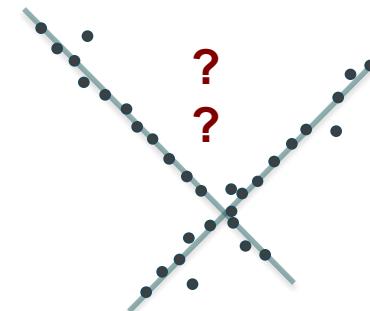
- Simple, closed form
- Fast!

- Problems:

- Do not allow multiple fits
- Not robust: Very sensible to outliers

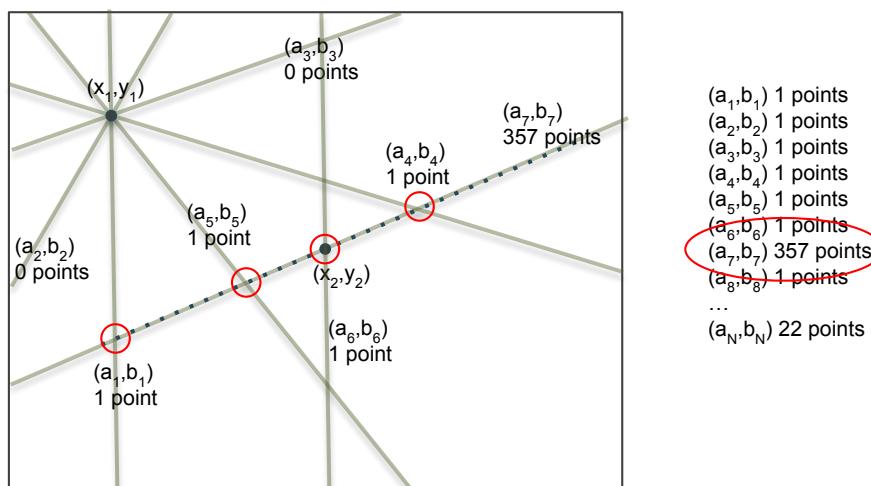
- Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)
- Modeling depth in depth estimation



Shape: Hough transform

- Method: generate shape candidates by varying shape parameters, try to match with data and vote best one(s)
- Select maximum points in parameter space
 - Naïve approach: for each possible line in the image, count the contour pixels that are compatible with this line



Shape: Hough Transform

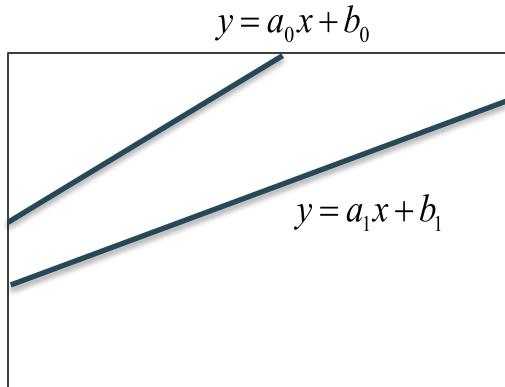
- Method: generate shape candidates by varying shape parameters, try to match with data and vote best one(s)
- Select maximum points in parameter space
- **Problem: computational complexity!**
 - Infeasible to generate all combinations of parameters and check on data
- The features (for instance, edge pixels) vote for the model instances that are compatible to them
 - Cycle through features, cast votes for model parameters.
 - Select model parameters that receive a lot of votes.
 - Noise & clutter do not follow the model and receive few votes

Shape: Hough Transform

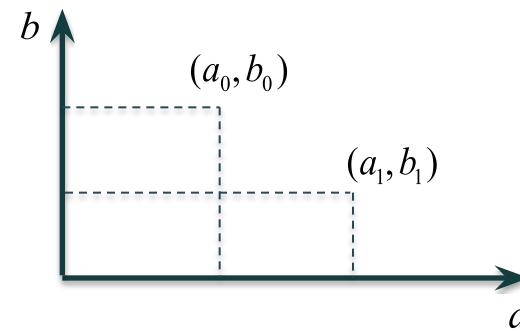
Line equation:

$$y = ax + b$$

Image space



Parameter space



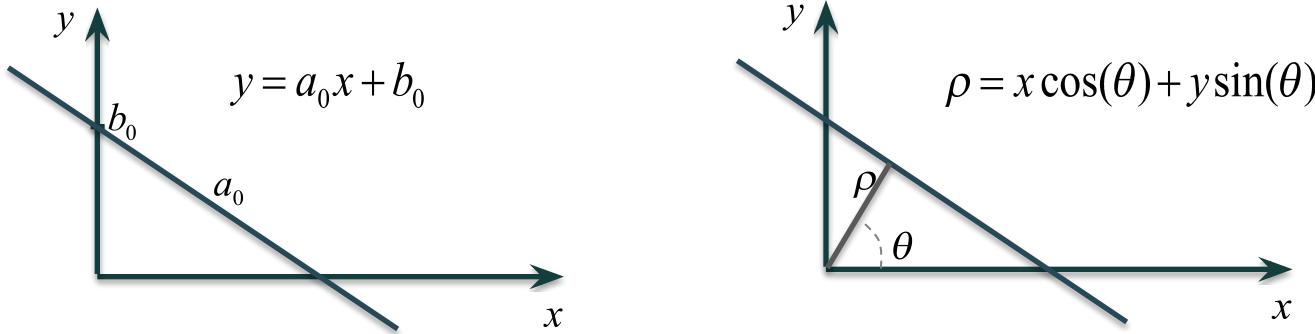
A line in image space corresponds to a point in parameter space



Shape: Hough Transform

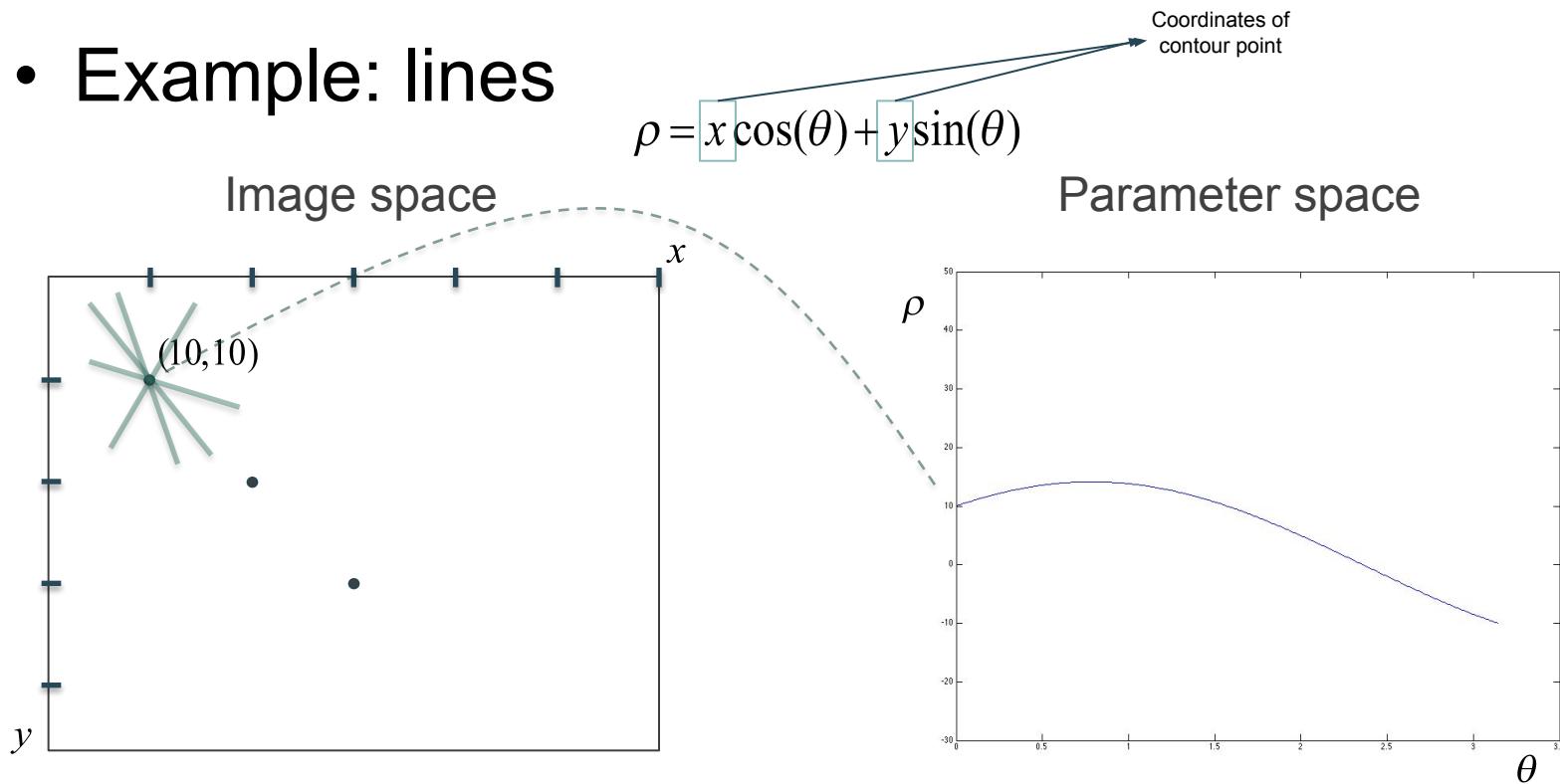
- Vertical lines → parameter a (line slope) goes to infinity!!
- Polar representation is better in this case

$$\rho = x \cos(\theta) + y \sin(\theta)$$



Shape: Hough Transform

- Example: lines

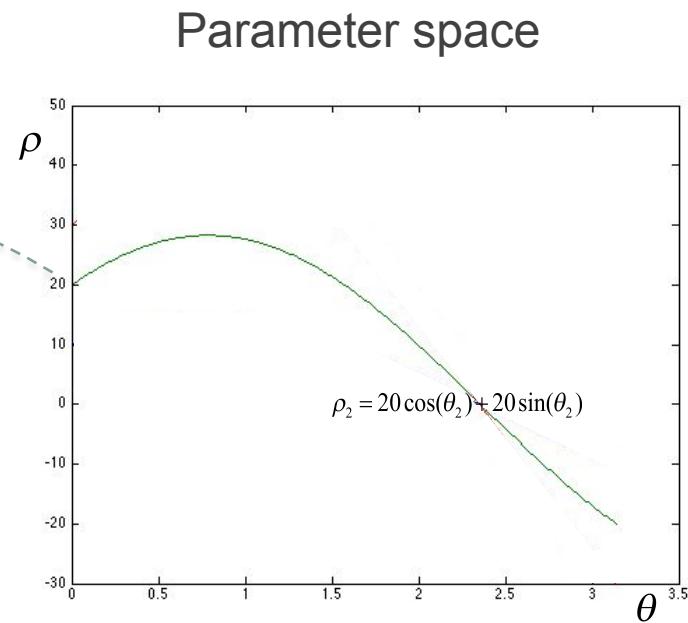
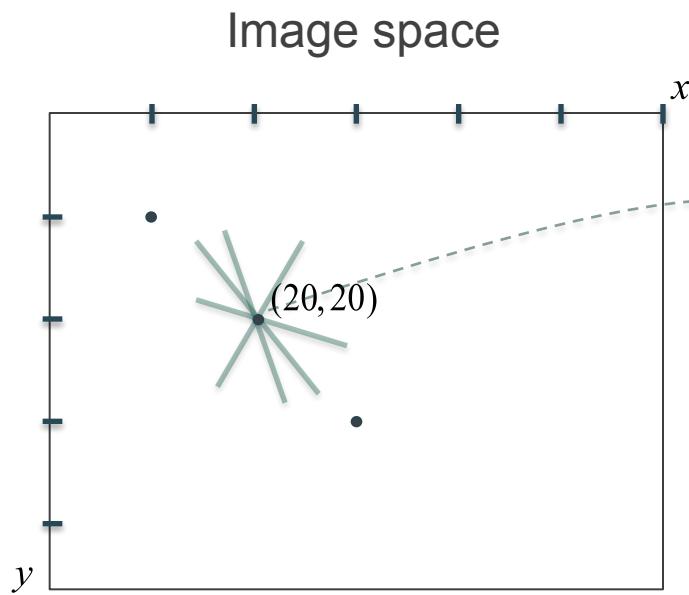


Each possible line passing through a point reflects into a ‘sinusoidal’ curve in parameter space



Shape: Hough Transform

- Example: lines



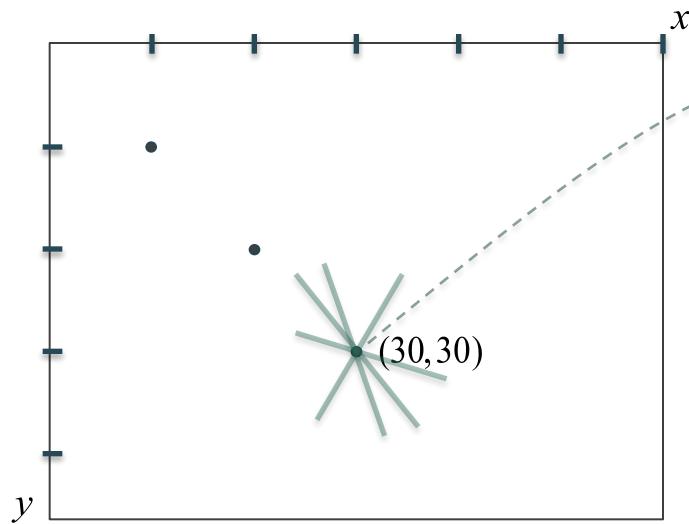
Each possible line passing through a point reflects into a ‘sinusoidal’ curve in parameter space



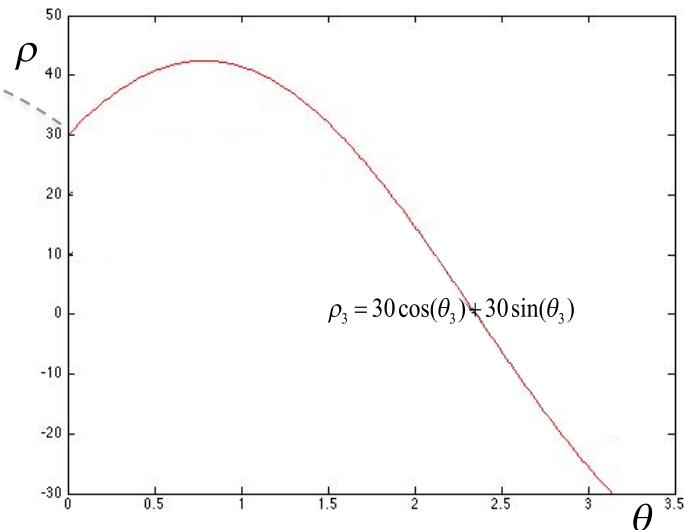
Shape: Hough Transform

- Example: lines

Image space



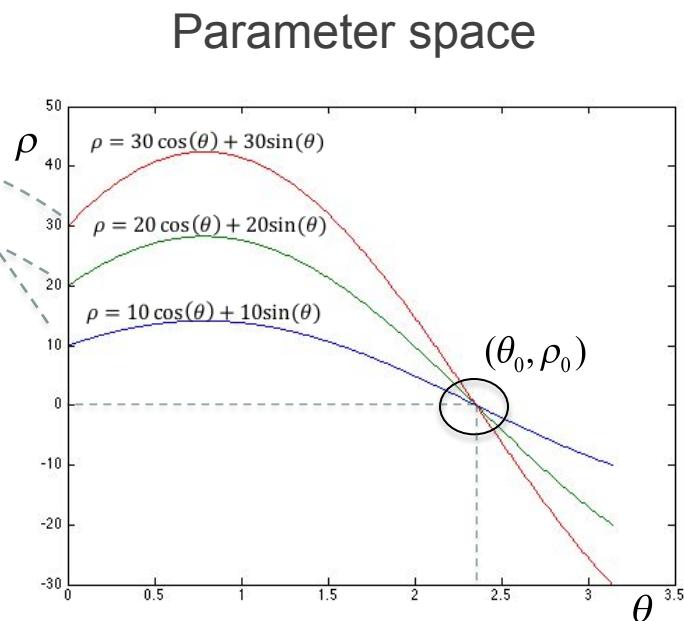
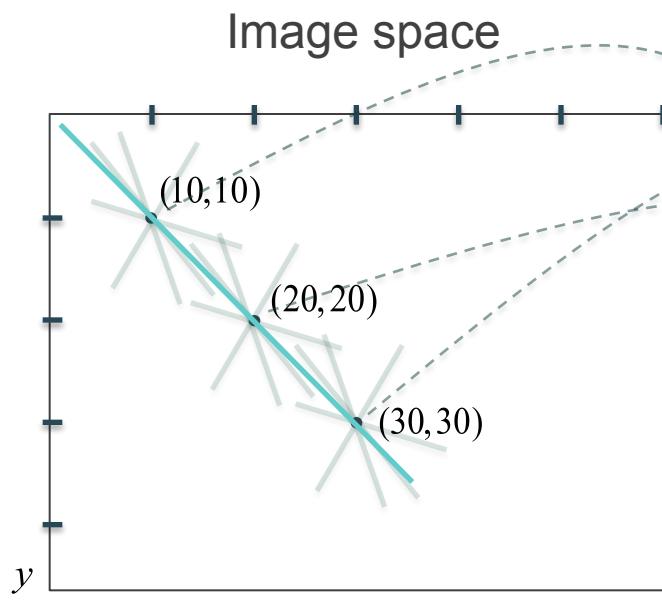
Parameter space



Each possible line passing through a point reflects into a ‘sinusoidal’ curve in parameter space

Shape: Hough Transform

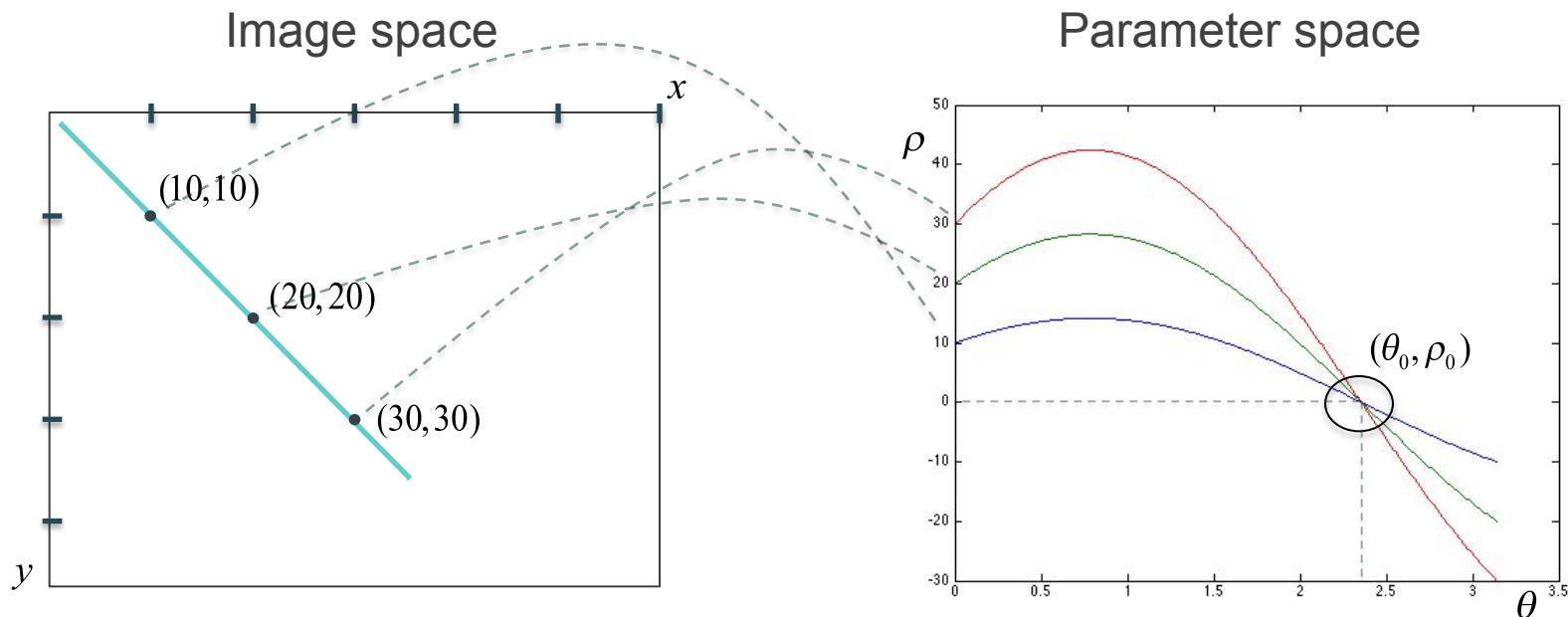
- Example: lines



Points on the same line in image space define curves in the parameter space that intersect at a single point.

Shape: Hough Transform

- Example: lines



Main idea: points in image space are represented by **curves** in parameter space. Find the points in parameter space that have many curves passing through.

Shape: Hough Transform

1. Quantize the parameter space

```
int P[pmax][θmax]; // accumulators
```

```
P = 0
```

2. Loop for all edge points

```
for each edge point (x, y) {
```

```
    // Compute parameters
```

```
    for (θ = 0; θ <= θmax; θ = θ + Δθ) {
```

```
        ρ = x · sin(θ) + y · cos(θ); // round off to integer (quantization)
```

```
        (P[ρ][θ])++;
```

```
}
```

```
}
```

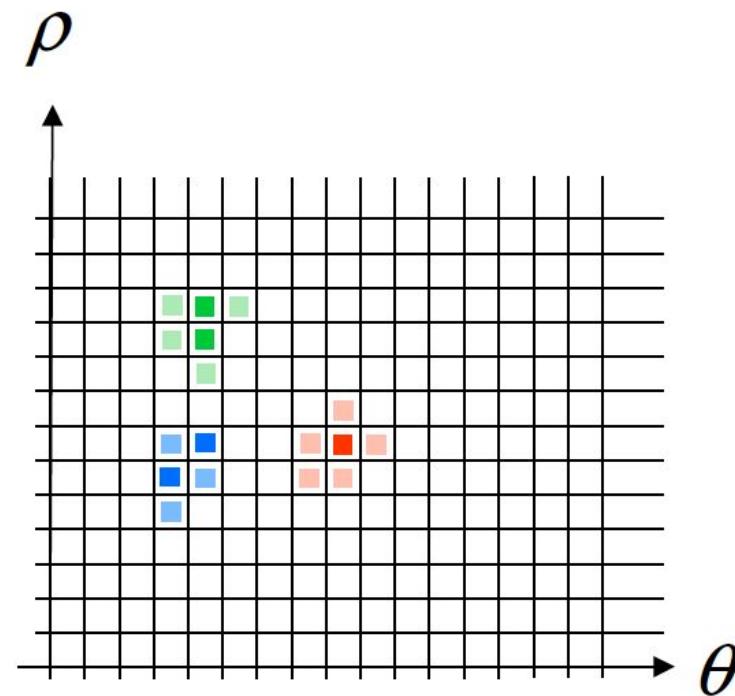
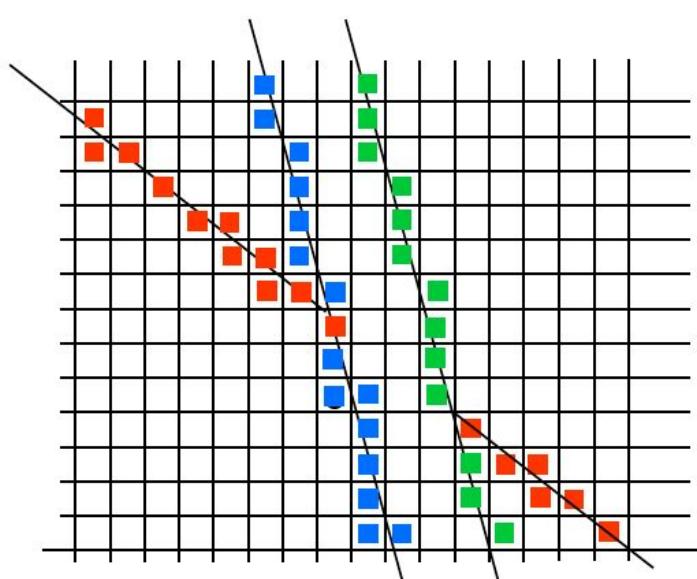
3. Find the peaks in P[ρ][θ].

Quantized angle

Source: Chang Shu

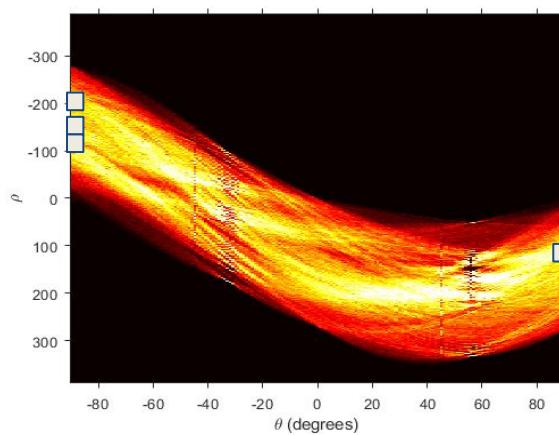
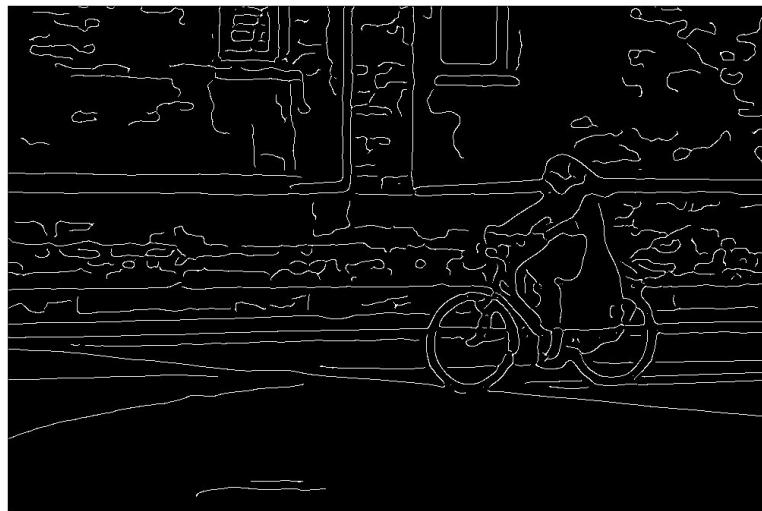
Shape: Hough Transform

- Select quantization of parameter space
 - Too coarse: Too many different lines correspond to a single bucket
 - Too fine: miss lines because points not exactly collinear vote for different buckets



Source: Chang Shu

Shape: Hough Transform



Matlab:

- `hough()`
- `houghpeaks()`
- `houghlines()`

Python:

- `scikit-image`
- `OpenCV`

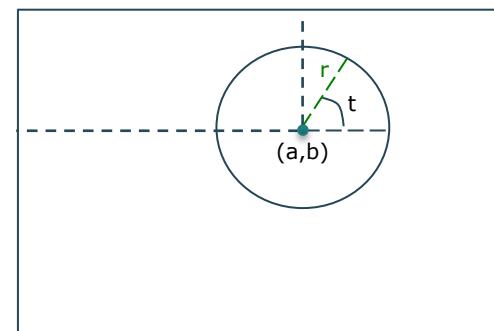
Shape: Hough Transform

Example: circumference:

```
int P[amax][bmax][rmax]; // accumulators  
for each edge point (x, y) {  
    // Compute parameters  
    for (r = rmin; r <= rmax; r = r+Δr) {  
        for (t = 0; t <= tmax; t = t+Δt) {  
            a = x - r·cos(t);  
            b = y - r·sin(t);  
            P[a][b][r]++;  
        }  
    }  
}
```

$$a = x - r \cos(t)$$

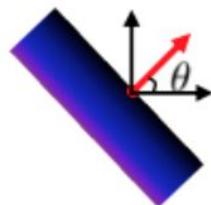
$$b = y - r \sin(t)$$



Shape: Hough Transform

Incorporating gradients:

- When detecting edges, gradient direction provides the orientation of the edge
- No need to loop over all possible angles, line is determined
- Faster computation!



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

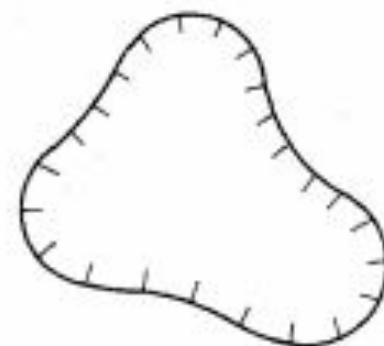
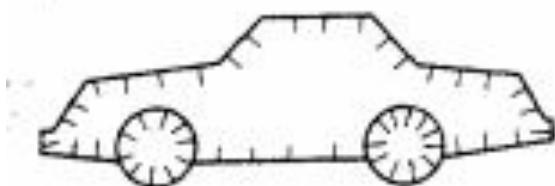
```
for each edge point (x, y) {  
    // Compute parameters  
    θ = gradient_orientation (x,y)  
    ρ = x·sin(θ)+y·cos(θ)  
    (P[ρ][θ])++  
}
```

Shape: Hough Transform

- Pros
 - All points are processed independently, so it can cope with occlusion
 - Moderate robustness to noise: noise points unlikely to contribute consistently to any single bin
 - Detection of multiple instances of a model in a single pass
- Cons
 - Search time increases exponentially with the number of model parameters
 - Non-target shapes can produce spurious peaks in parameter space
 - Quantization: hard to pick a good grid size

Shape: Generalized Hough Transform

- The generalized Hough transform can be used to detect arbitrary shapes (i.e., shapes having no simple analytical form).
- It requires the complete specification of the exact shape of the target object.
- Uses edge direction information



D.H. Ballard, **Generalizing the Hough transform to detect arbitrary shapes** Pattern Recognition, Volume 13, Issue 2, 1981, Pages 111–122

DEMO: <http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/GeneralHoughTransformation.html>

Shape: Generalized Hough Transform

- The generalized Hough transform can be used to detect arbitrary shapes (i.e., shapes having no simple analytical form).
- **It requires the complete specification of the exact shape of the target object.**

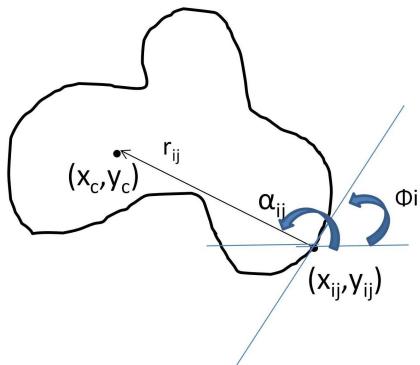
$$a = \{y, s, \theta\}$$

y is a reference origin for the shape, θ is its orientation, and $s = (s_x, s_y)$ describes two orthogonal scale factors

- Shape is described in terms of a template table called the **R table** of possible edge pixel orientations
- Computation of parameters s and θ is accomplished by straightforward transformations to this table

Shape: Generalized Hough Transform

- R table uses edge direction information
 - Choose a reference point y for the shape (typically chosen inside the shape).
 - For each boundary point x , compute $\phi(x)$, the gradient direction and $r = y - x$ as shown in the image. Store r as a function of ϕ
 - Each index of ϕ may have many values of r



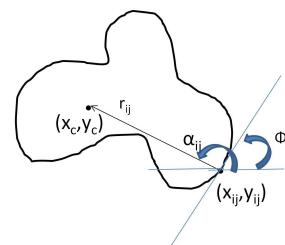
i	ϕ_i	R_{ϕ_i}
1	0	$(r_{11}, a_{11}) (r_{12}, a_{12}) \dots (r_{1n}, a_{1n})$
2	$\Delta\phi$	$(r_{21}, a_{21}) (r_{22}, a_{22}) \dots (r_{1m}, a_{1m})$
3	$2\Delta\phi$	$(r_{31}, a_{31}) (r_{32}, a_{32}) \dots (r_{3k}, a_{3k})$
...

Image by Skhushu - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=37263680>

Shape: Generalized Hough Transform

- **Code:**

```
def build_r_table(image, origin):  
    '''  
    R-table from the given shape image and a ref point  
    '''  
    edges = canny(image,  
                  low_threshold=MIN_CANNY_THRESHOLD,  
                  high_threshold=MAX_CANNY_THRESHOLD)  
    gradient = gradient_orientation(edges)  
  
    r_table = defaultdict(list)  
    for (i,j),value in np.ndenumerate(edges):  
        if value:  
            r_table[gradient[i,j]].append((origin[0]-i,  
                                           origin[1]-j))  
  
    return r_table
```



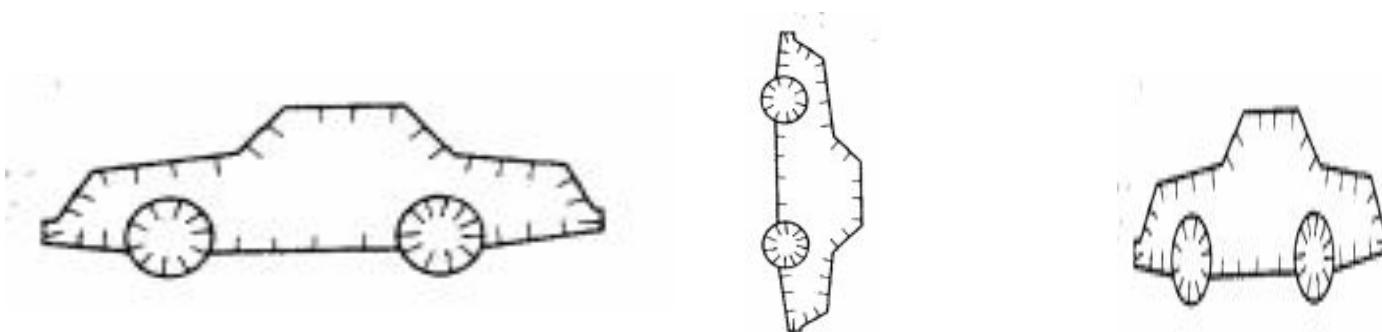
i	ϕ_i	R_{ϕ_i}
1	0	$(r_{11}, a_{11}) (r_{12}, a_{12}) \dots (r_{1n}, a_{1n})$
2	$\Delta\phi$	$(r_{21}, a_{21}) (r_{22}, a_{22}) \dots (r_{1m}, a_{1m})$
3	$2\Delta\phi$	$(r_{31}, a_{31}) (r_{32}, a_{32}) \dots (r_{3k}, a_{3k})$
...

```
def accumulate_gradients(r_table, grayImage):  
    '''  
    Perform a General Hough Transform with the given image and R-table  
    '''  
    edges = canny(grayImage, low_threshold=MIN_CANNY_THRESHOLD,  
                  high_threshold=MAX_CANNY_THRESHOLD)  
    gradient = gradient_orientation(edges)  
  
    accumulator = np.zeros(grayImage.shape)  
    for (i,j),value in np.ndenumerate(edges):  
        if value:  
            for r in r_table[gradient[i,j]]:  
                accum_i, accum_j = i+r[0], j+r[1]  
                if accum_i < accumulator.shape[0] and accum_j <  
                    accumulator.shape[1]:  
                    accumulator[accum_i, accum_j] += 1  
    return accumulator
```

Image by Skhushu - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=37263680>

Shape: Generalized Hough Transform

- **Advantages**
 - Robust to partial or slightly deformed shapes (i.e., robust to recognition under occlusion).
 - Robust to the presence of additional structures in the image.
 - Tolerant to noise.
 - Can find multiple occurrences of a shape in one pass.
- **Disadvantages**
 - Has substantial computational and storage requirements



Shape: RANSAC

- Goal: adjust models to noisy data
- Shapes with a mathematical model (lines, planes, circles, etc.)
- Idea:
 - Obtain random subsets of points and adjust to model
 - Store the parameters that minimize residual error by using discriminant function:

$$\min \sum_i f(r_i)$$

where $f(r_i) = \begin{cases} 1, & r_i > t \\ 0, & r_i \leq t \end{cases}$

r_i : residual of i-th sample
t: threshold

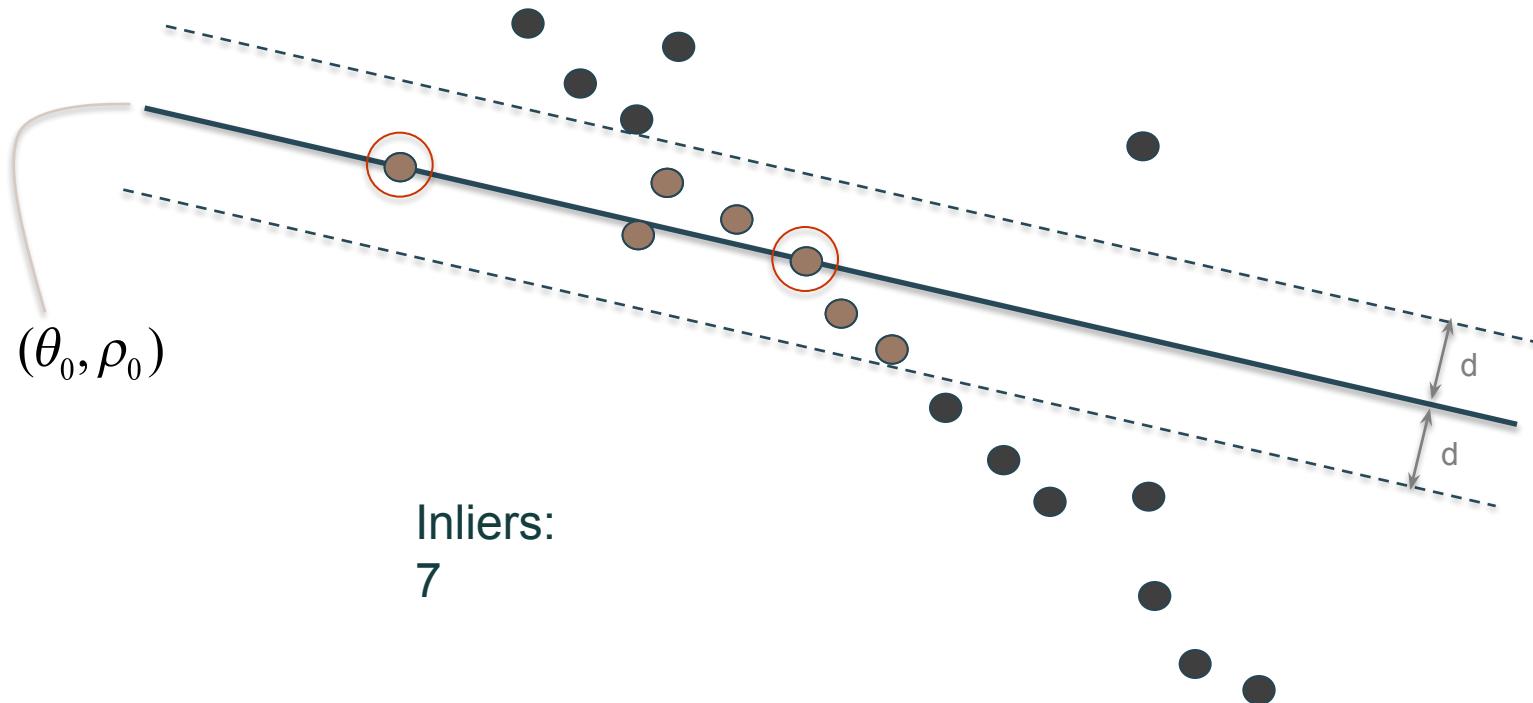
Shape: RANSAC

- Process:
 - Randomly select a minimal set of points necessary to estimate an instance of the model
 - For instance, two points in the case of a line, three for a circle, etc.
 - Compute the parameters and evaluate the model, measuring the number of inliers.
 - Re-compute the model using only inliers using LS (optional)
 - Iterate procedure K times and select the model that better represent the set of data (for instance, the one that gives the maximum number of inliers).

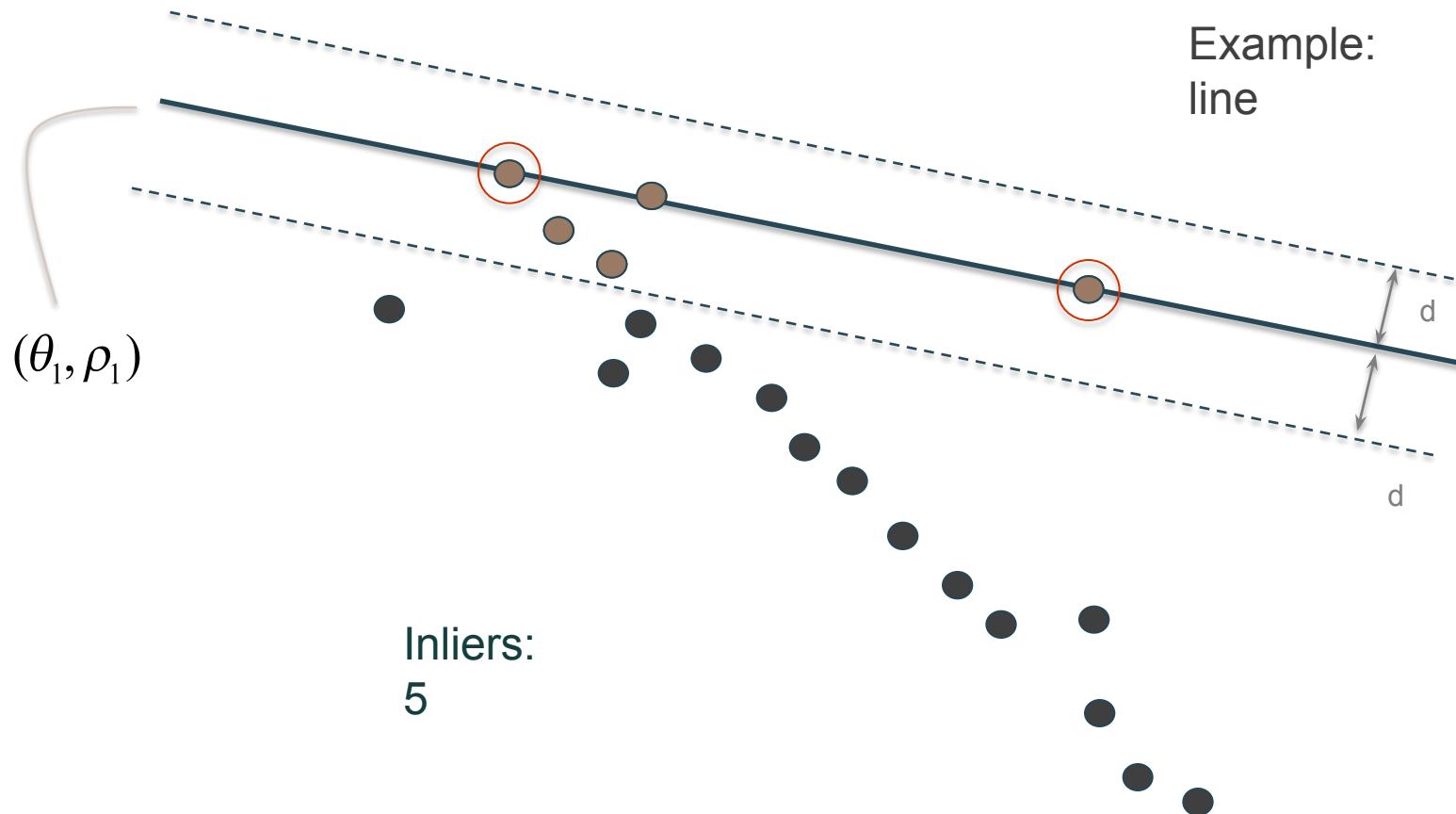
Inliers: points which approximately can be fitted to the model within a given small error

Shape: RANSAC

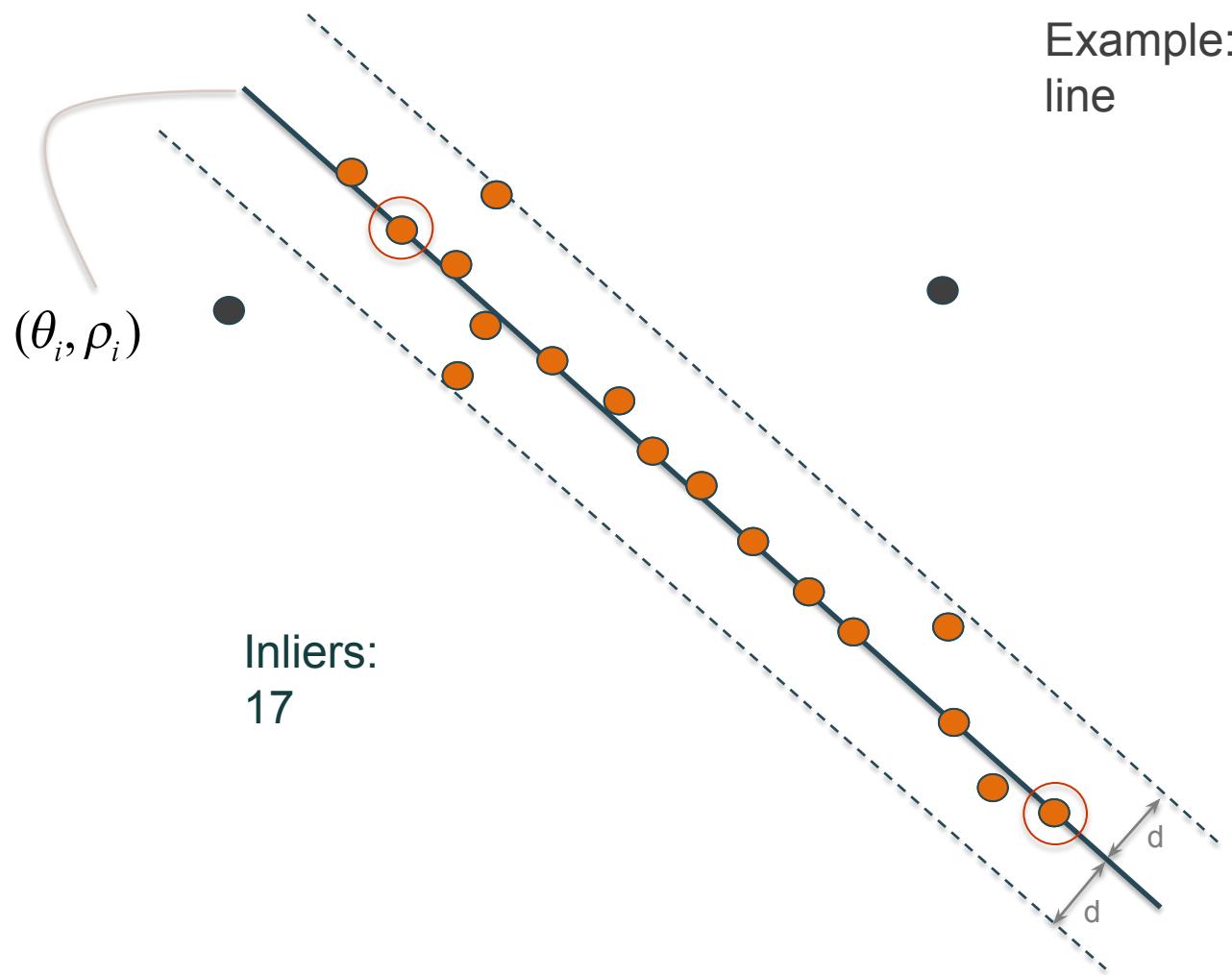
Example:
line



Shape: RANSAC

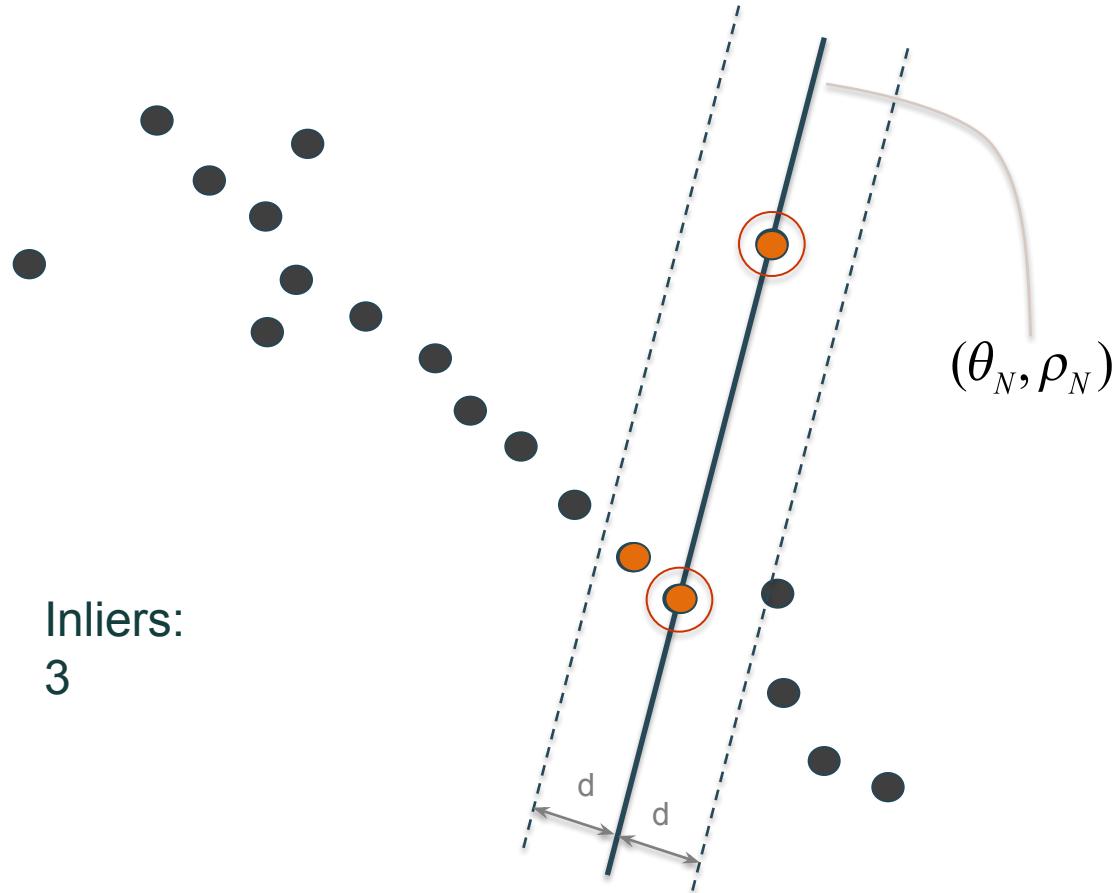


Shape: RANSAC



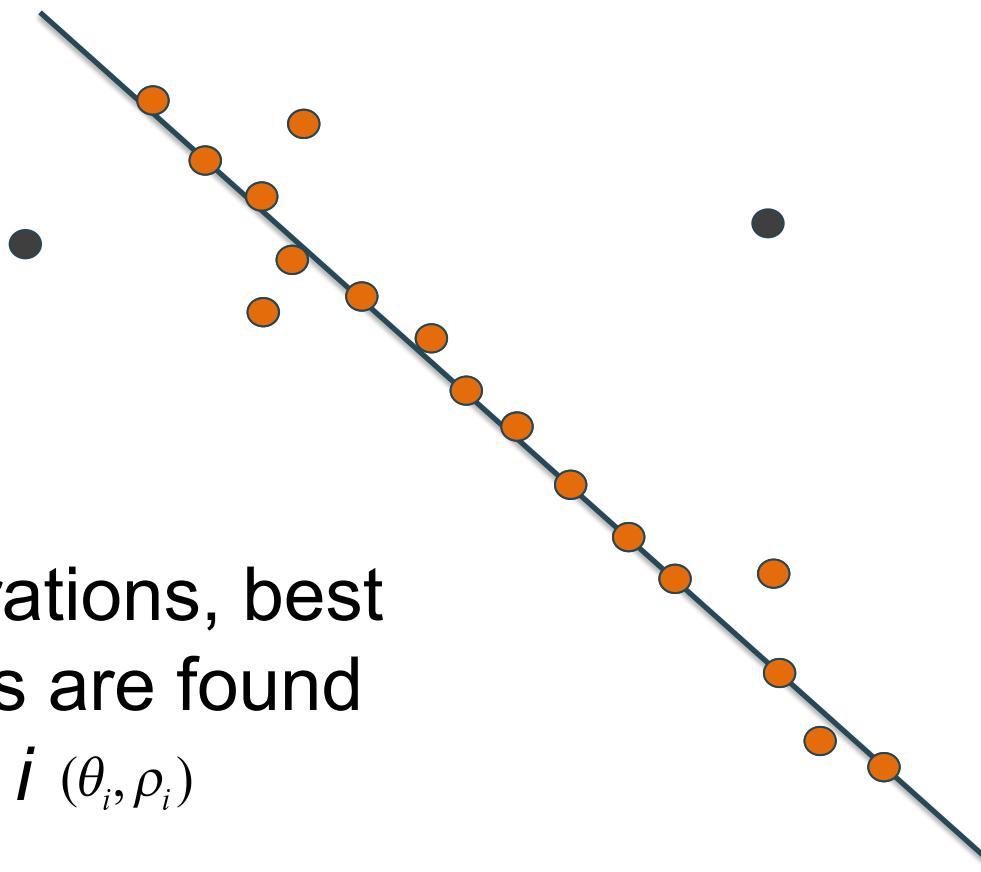
Shape: RANSAC

Example:
line



Shape: RANSAC

Example:
line



After K iterations, best
parameters are found
in iteration i (θ_i, ρ_i)

Shape: RANSAC

- Variants:
 - Define a threshold on the measure and stop iterating if the measure is reached
 - Saves computational time at the expense of another parameter
 - Change model evaluation: use fitting error instead of inliers
 - Randomly select a minimal set of points necessary to estimate an instance of the model
 - Compute the parameters and evaluate the number of inliers N_{inliers}
 - If $N_{\text{inliers}} < Th_{\text{inliers}}$, discard model. Otherwise re-compute model using LS and compute fitting error on the set of inliers. Store the fitting error.
 - Iterate procedure many times and select the model with minimal error

Shape: RANSAC

- RANSAC parameters:
 - Number of model parameters (n), number of iterations (k), threshold to consider a point an inlier (d)
- Selecting number of iterations (k):
 - Suppose we know the percentage of inliers (w)
 - Let p be the probability that RANSAC in some iteration selects only inliers (i.e. the probability to produce a useful result)

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

- Selecting the threshold for inliers/outliers (d)
 - Application/model dependent

Shape: RANSAC

- Number of iterations:

Sample size n	Proportion of outliers						
	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

Shape: RANSAC

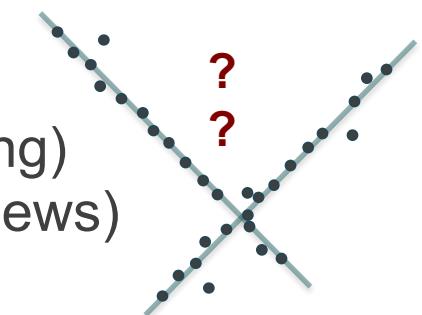
- Selecting number of iterations :
 - If we do not know the percentage of inliers: compute an estimate at each iteration:

```
if ninliers > bestscore      % Largest set of inliers so far...
    bestscore = ninliers;    % Record data for this model

    % Update estimate of N, the number of trials to ensure we pick,
    % with probability p, a data set with no outliers.
    fracinliers = ninliers/npts;
    k = log(1 - p)/log(1 - fracinliers^n);
end
```

Shape: RANSAC

- **Strengths:**
 - Robust to outliers
 - Applicable for larger number of parameters than Hough transform
 - Parameters are easier to choose than Hough transform
- **Weaknesses:**
 - Computational time (iterations) grows quickly with fraction of outliers and number of parameters
 - Do not allow multiple fits
- **Common applications**
 - Computing a homography (e.g., image stitching)
 - Estimating fundamental matrix (relating two views)
 - Modeling depth in depth estimation



Shape: RANSAC alternatives

- MSAC:
 - Inliers scored according to their fitness to the model, outliers are given a constant weight.
 - The solution that produces the smallest median of residuals is picked.
 - Similar results respect to RANSAC
 - Same disadvantages: requires an application dependent threshold and a similar computational cost

$$f(r_i) = \begin{cases} r_i, & r_i \leq t \\ t, & r_i > t \end{cases}$$

Source: Hartley and Zisserman

Shape: RANSAC alternatives

- LMedS:
 - Similar to MSAC: Minimize the median of the residuals
 - The threshold is determined automatically as a value proportional to the median of the residuals produced for each subset.
 - Typically produces better results than RANSAC/MSAC when the inliers contain some noise because in such cases RANSAC or MSAC might have thresholds set too large.

$$f(r_i) = \begin{cases} r_i, & r_i \leq t \\ t, & r_i > t \end{cases}$$

Navigation icons: back, forward, search, etc.

Shape: RANSAC alternatives

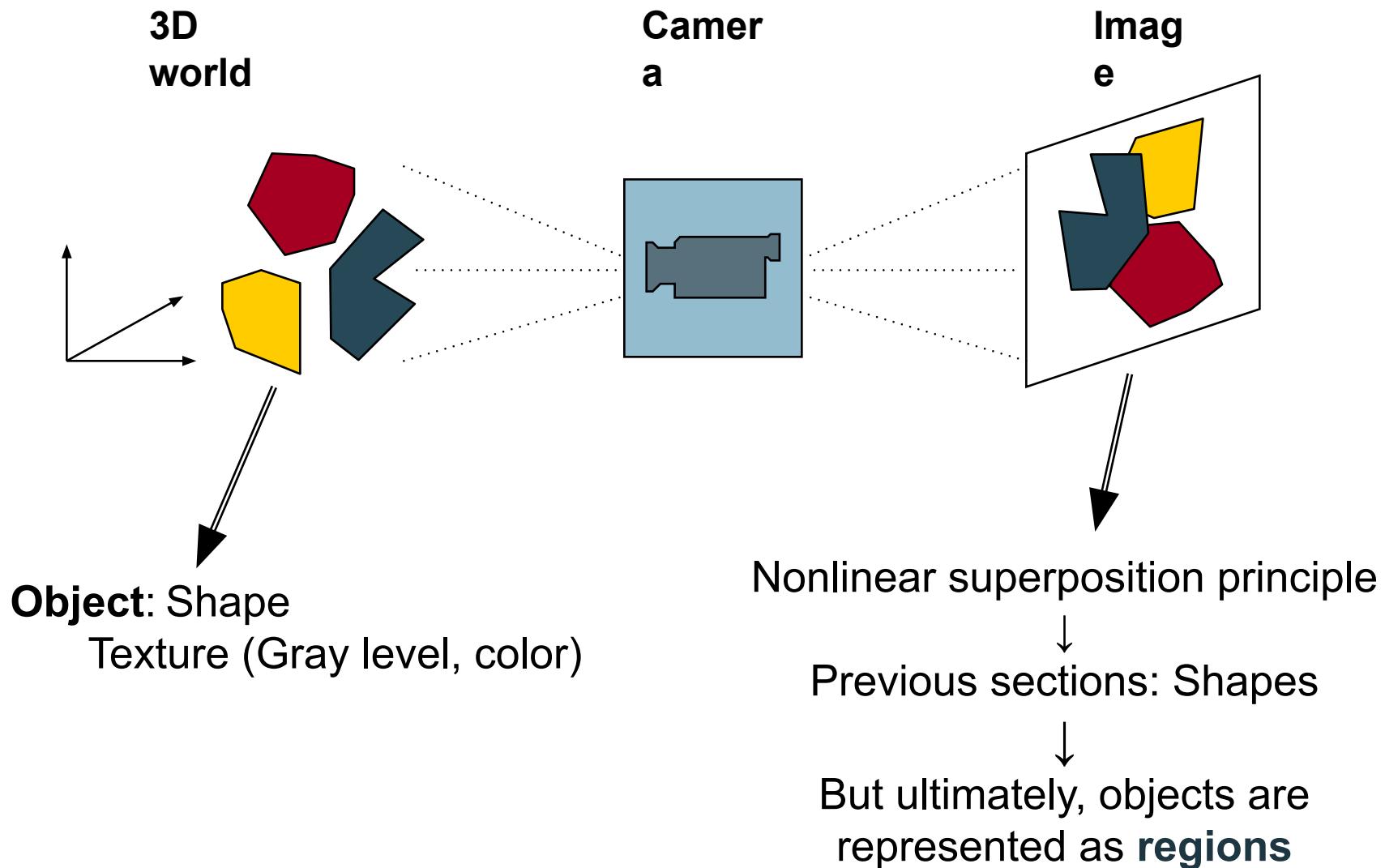
- PROMEDS:
 - Takes into account additional information of the **quality scores** of the points
 - Points are ordered from best to worse quality
 - The algorithm starts picking subsets of points among the points having the largest quality.
 - If a suitable solution having a minimum number of inliers is not found, the algorithm selects matches with lower quality.
 - Usually, solutions are found with a few iterations. Much faster than RANSAC, with similar quality
 - An application dependent threshold is required as in RANSAC
 - Quality scores may not be available in most applications

Shape: Conclusions

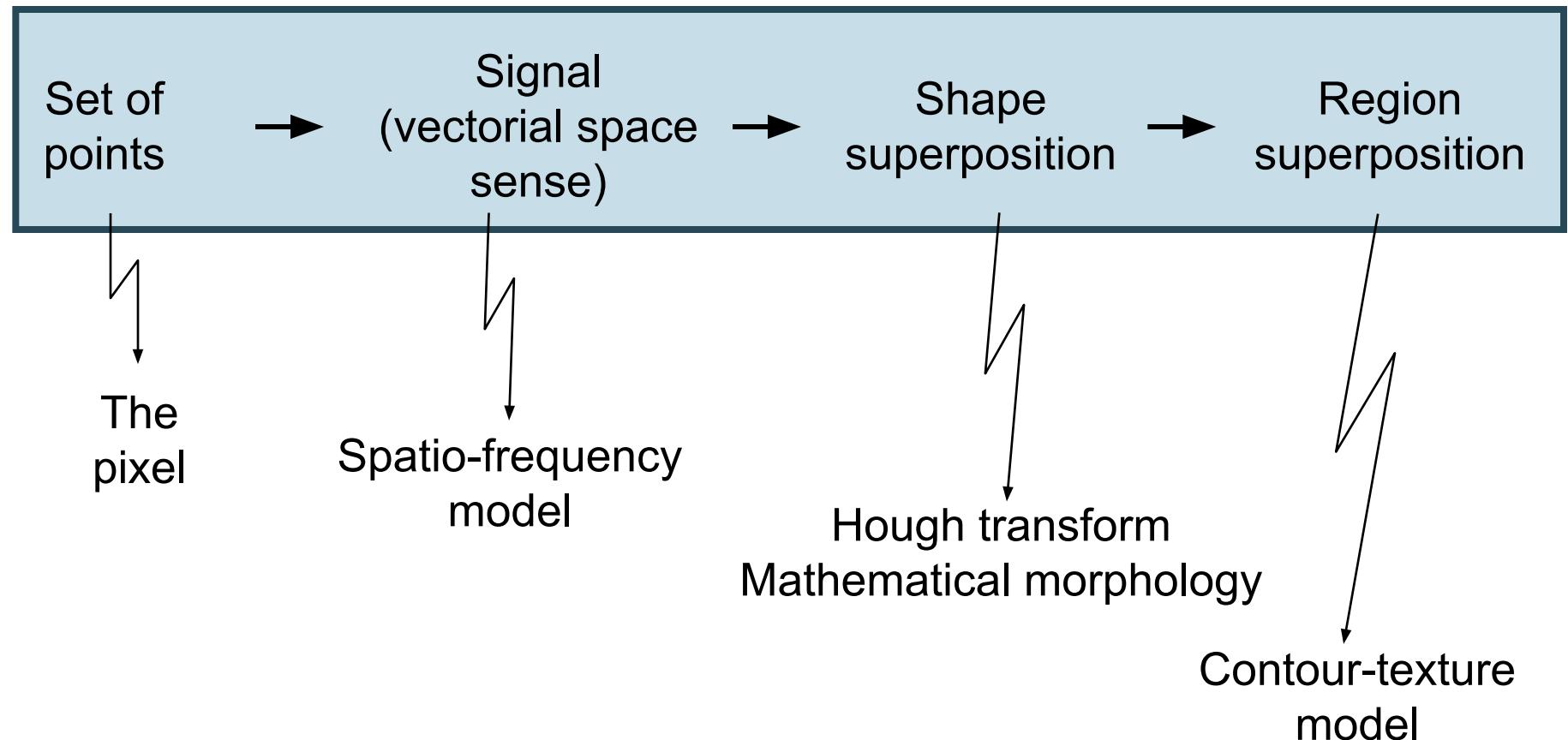
	LS	RANSAC	HOUGH
Multiple fits	No	No	Yes
Computational complexity	Low, closed form	High. Depends on % outliers	High
Robustness	Low	High	Medium/High
Scalability (model dimensionality)	Good	Good	Bad
Applications	Homography, fund. Matrix, modeling depth	Homography, fund. Matrix, modeling depth	Lines, circles, simple forms

SEGMENTATION & CLASSIFICATION

Segmentation: objects & superposition model



Segmentation: Image interpretation

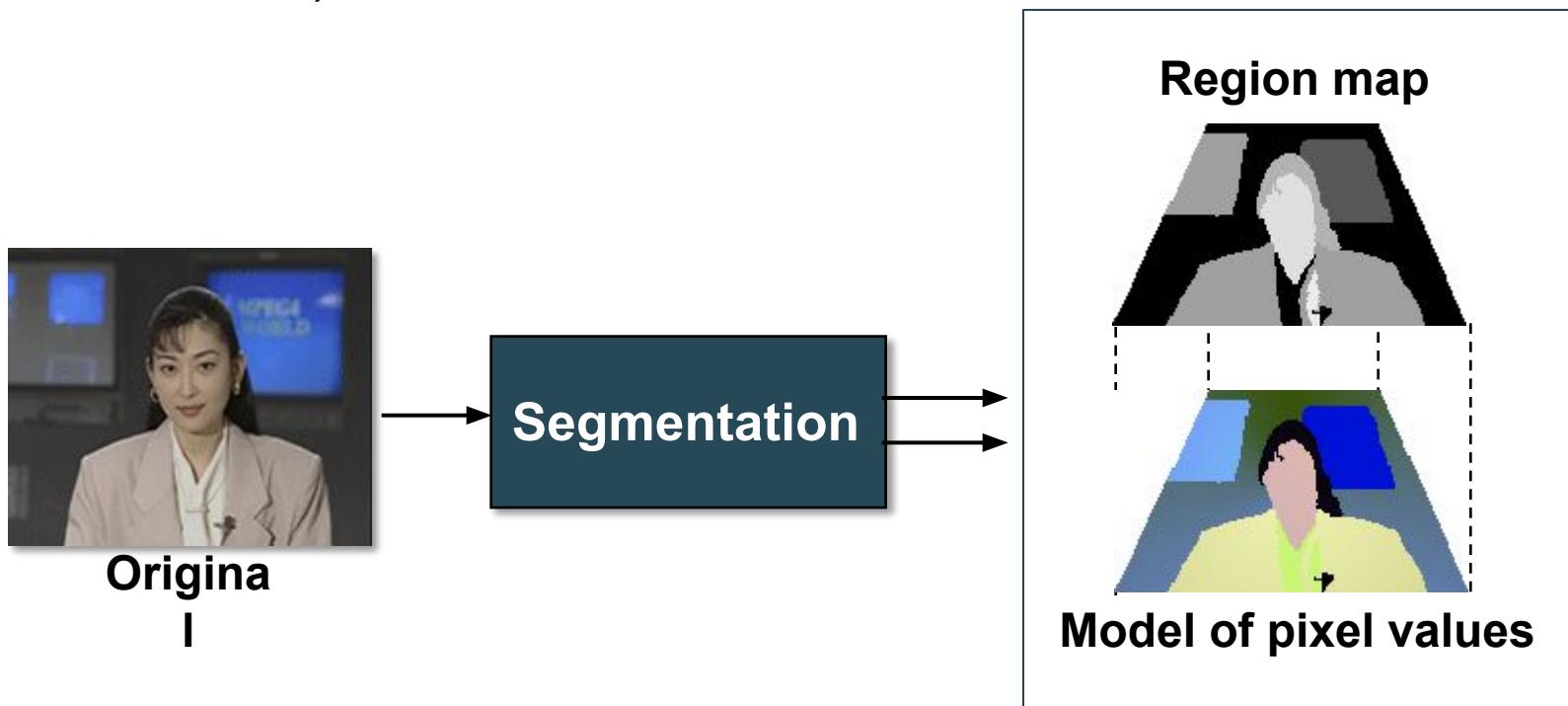


Segmentation

- The contour-texture model
- Segmentation definition
- General segmentation scheme
- Segmentation approaches:
 - Transition-based
 - Region-based

Segmentation: The contour-texture model

- Images can be understood as a set of patches forming a **region map (contour information)**, where the interior of each region can be represented by a **pixel model (texture information)**:



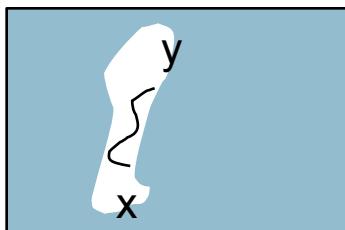
Segmentation: Regions

- A region is a **connected component** of the map:
 - Map value indicates the pixel membership (label image)
 - Regions have closed boundaries

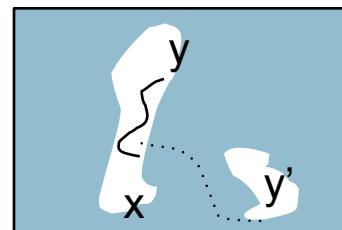


- **Connected component:**

- A set of pixels with the same label such that there exists a path, that is, a succession of **neighboring pixels, connecting them.**



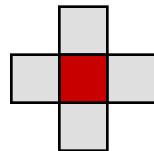
1(2) connected component(s)



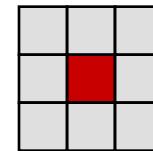
2(3) connected components

Segmentation: Regions

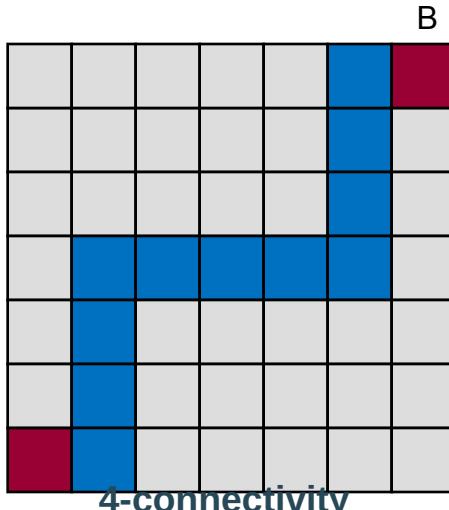
- What is a neighboring pixel in a discrete image?
 - **4-connectivity**
 - **8-connectivity**



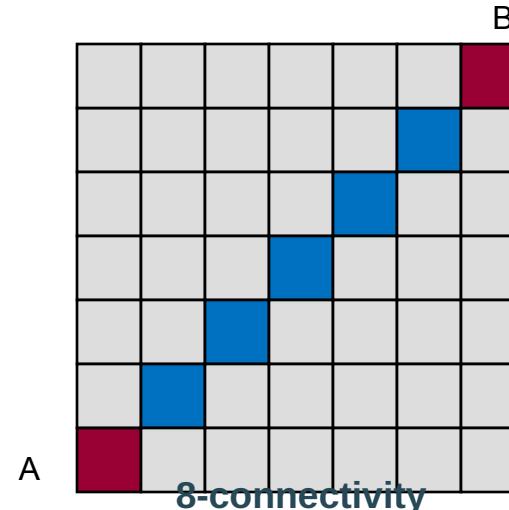
8-connectivity



- Examples of connected paths:

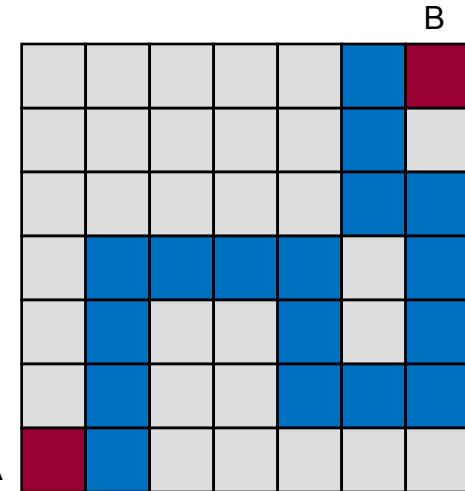


4-connectivity



8-connectivity

Also valid as 8-connected path



4-connectivity: 1 path

8-connectivity: various possible paths

Segmentation: Definition

- Create a **partition** of image I : $\{R_i\}$

$$\begin{cases} \bigcup R_i = \text{Image} \\ R_i \cap R_j = \emptyset, \quad \forall i, j \end{cases}$$

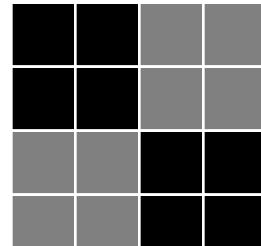
... of **connected regions** R_i, R_j that optimize a **criterion** C :

$$\begin{cases} C(R_i) = \text{True}, \quad \forall i \\ C(R_i \cup R_j) = \text{False}, \quad \forall i, j \end{cases}$$

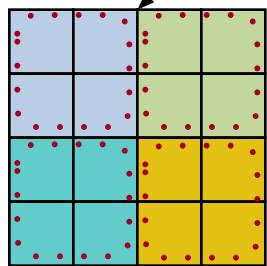
- What is a **good criterion** for segmentation
- How to **optimize** it?

Segmentation: Definition

Assuming images are defined over a **square grid**, two types of connectivity can be defined:

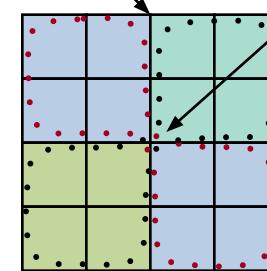


4 Connectivity



4 regions

8 Connectivity



2 regions

Overlapping

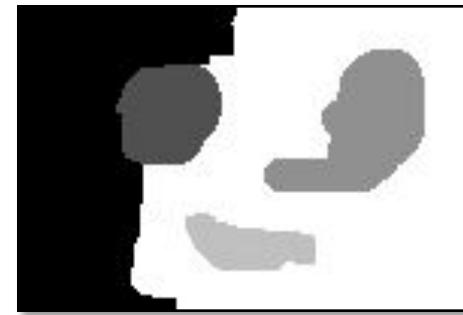
→ For segmentation, in general: 4 connectivity

Segmentation vs. classification

Classification: Segmentation with regions, called **classes**, that are not necessarily connected.



Classification



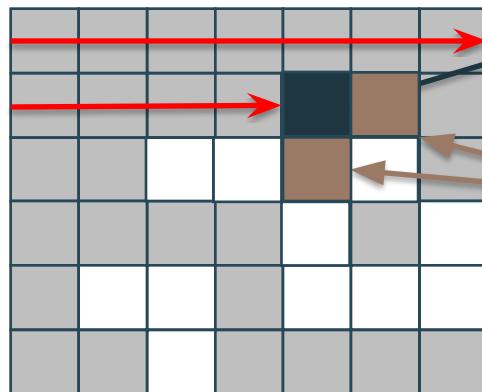
Segmentation
(image partition)

Labeling

Segmentation

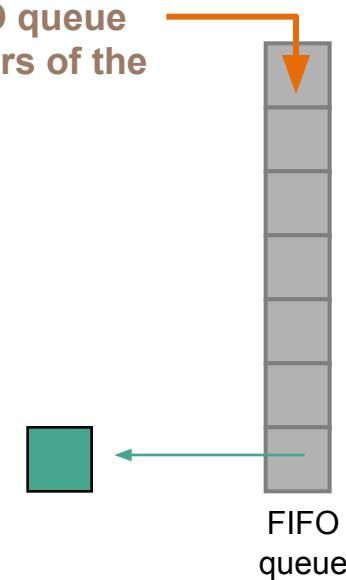
- **Labeling process:** Assign a different label to **each connected component**
 1. Search for a connected component
 2. Completely label the connected component

1. Scan to get the first pixel of the connected components



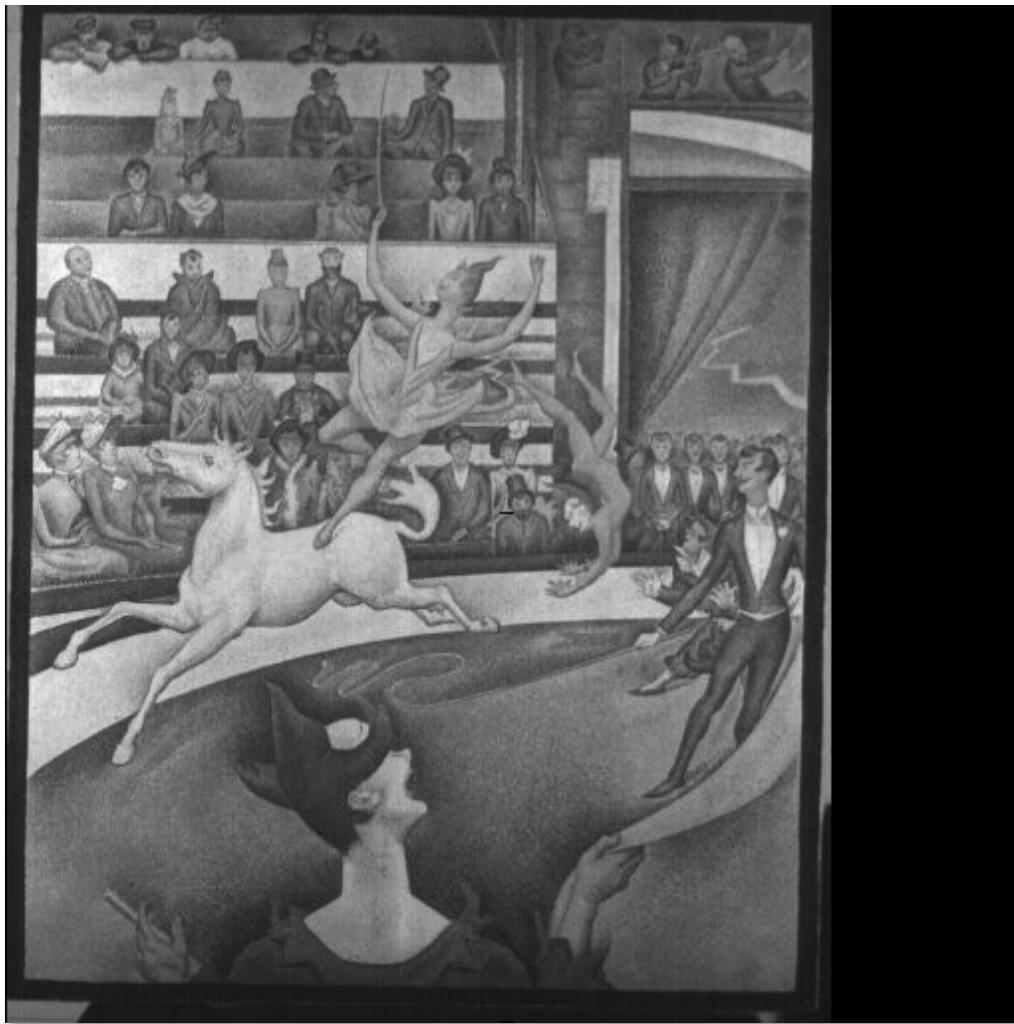
2. The first pixel of the connected component has been found. Insert in a FIFO queue and extract it.

3. Insert in a FIFO queue the pixel neighbors of the same connected component.



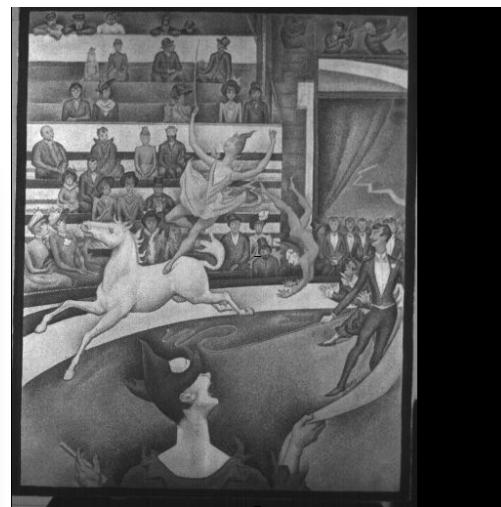
4. Extract the first pixel of the FIFO queue, assign the label and go to 3

Segmentation: Evaluation



- (Extremely) difficult
- Application dependant
- Often deceiving

Segmentation: Evaluation



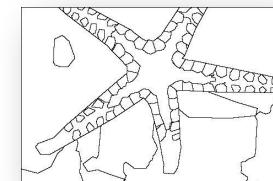
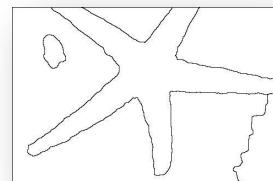
Best segmentation?

Segmentation

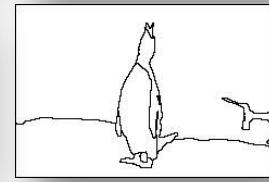
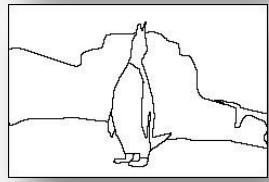
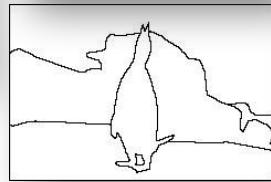
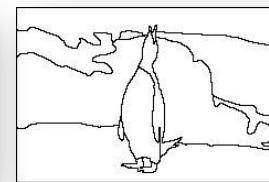
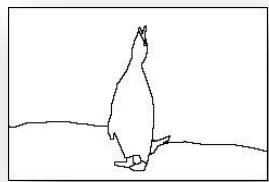
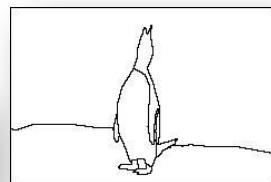
Segmentation is **an ill posed problem**:

- A unique solution does not exist and the partition definition depends on the final application

Different levels of detail



Same level of detail



Segmentation: General Scheme



- Remove useless or annoying information
- Preserve shape information
- Feature space:
 - || Gray level
 - || Color
 - || Variance
 - || Frequency component
- Pixels belonging to the same regions should correspond to homogeneous features
- Partition definition:
 - || Transition
 - || Homogeneity

Segmentation: Simplification

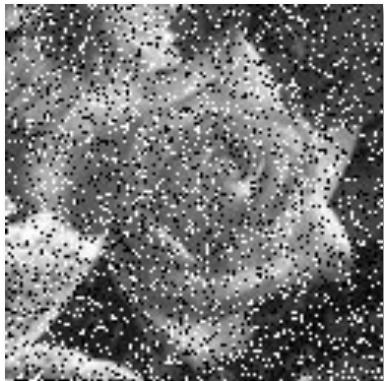


Image with
impulsive
noise



Image with artifacts



Median



Opening

Classical filters:

- Low-pass
- Median
- Opening / closing
- Op. reconstruction
- Alternating filters

Segmentation: Simplification

- In some cases, the application itself fixes a **minimum size** for the regions to be obtained:



Original



Mean



Median



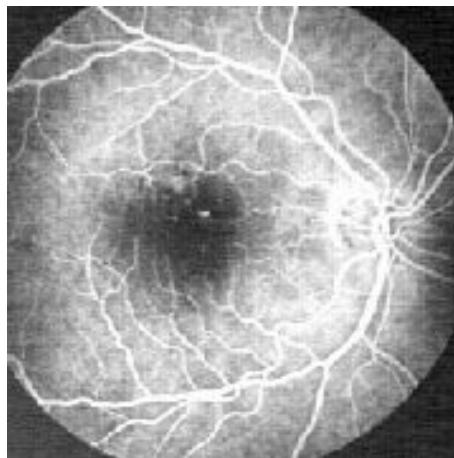
Open-Close



Open-Close
by
reconstruction

Segmentation: Simplification

Aneurisms

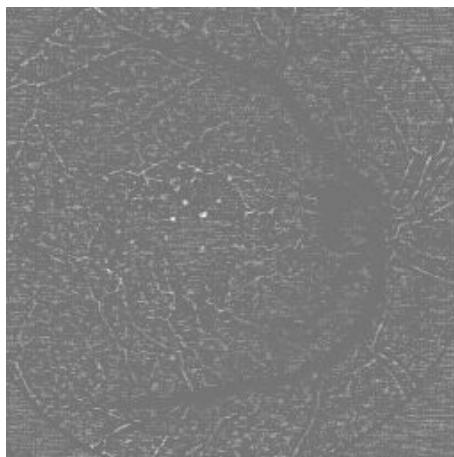


Text segmentation



Original

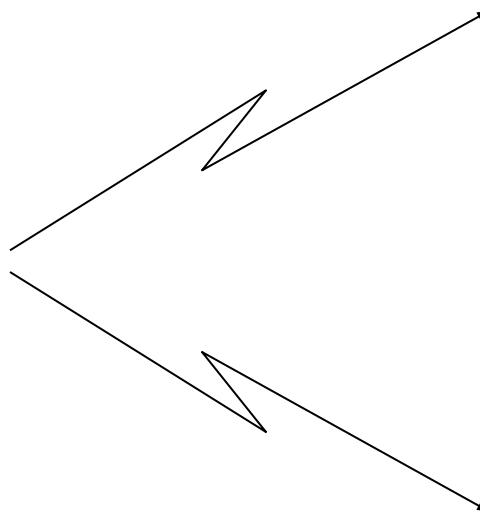
Top-hat



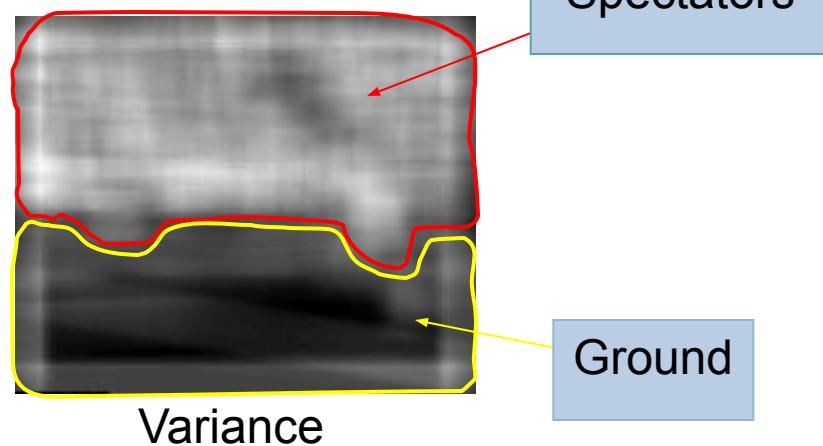
Segmentation: Feature extraction

Feature space:

- Gray level
- Color
- Variance
- Frequency
- Activity



Original

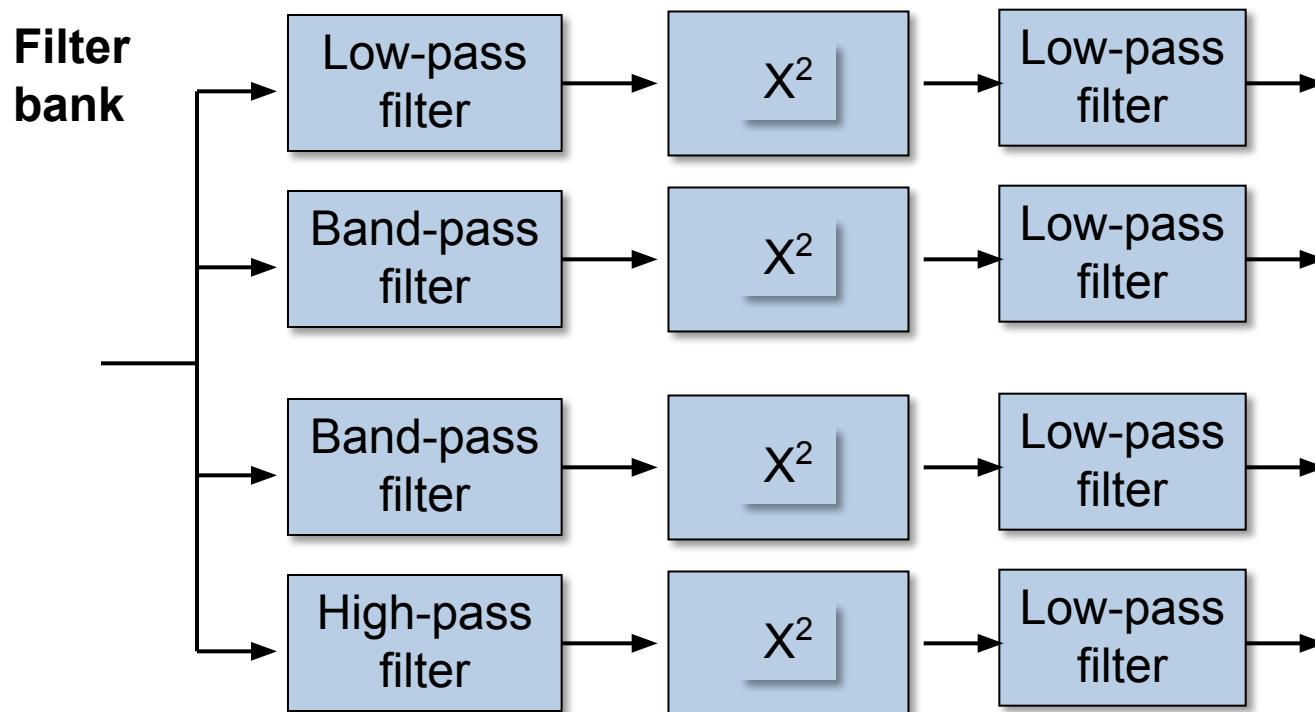


Variance

Segmentation: Feature extraction

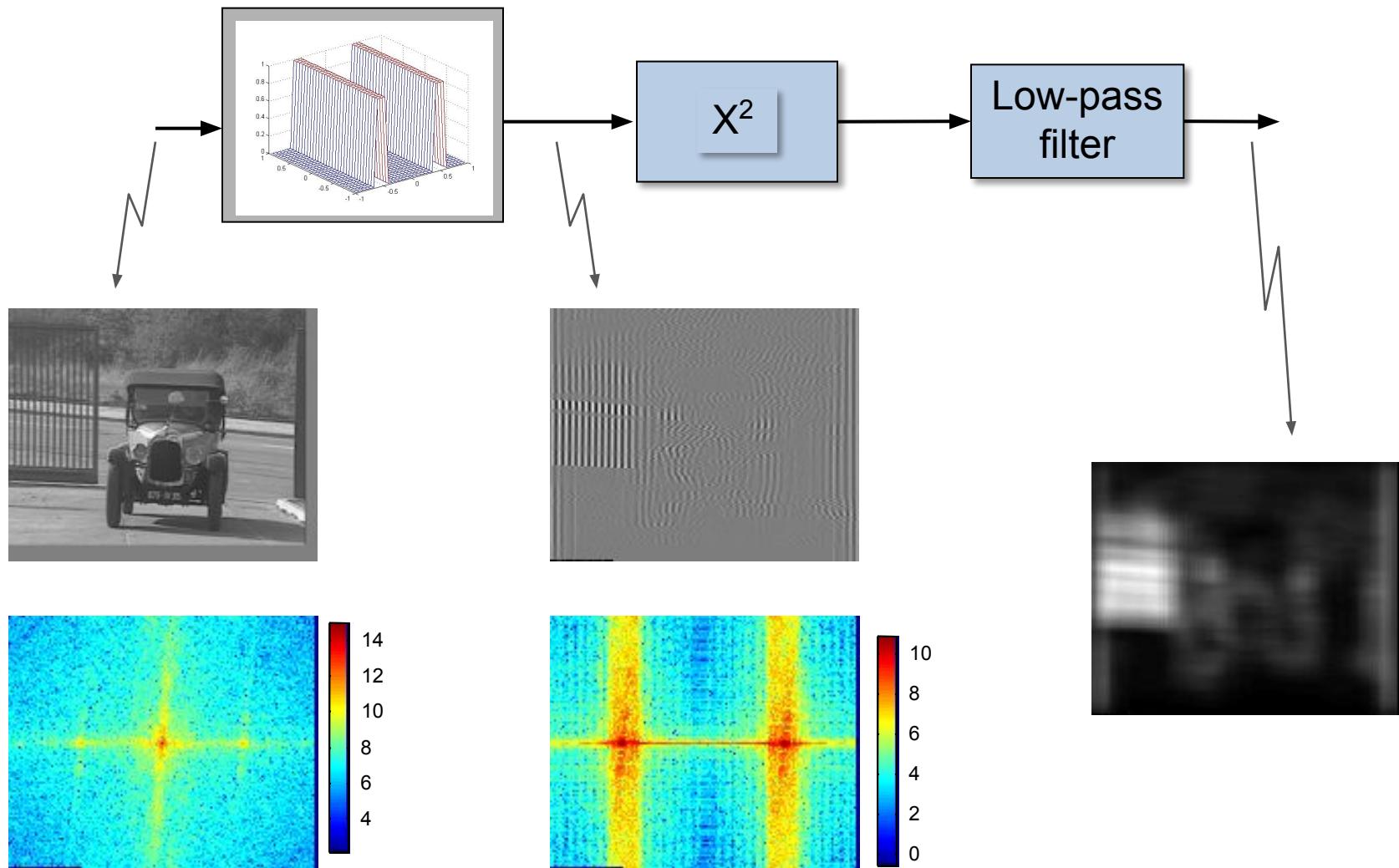
Frequency analysis

- Estimate the **energy** in a given frequency band



Segmentation: Feature extraction

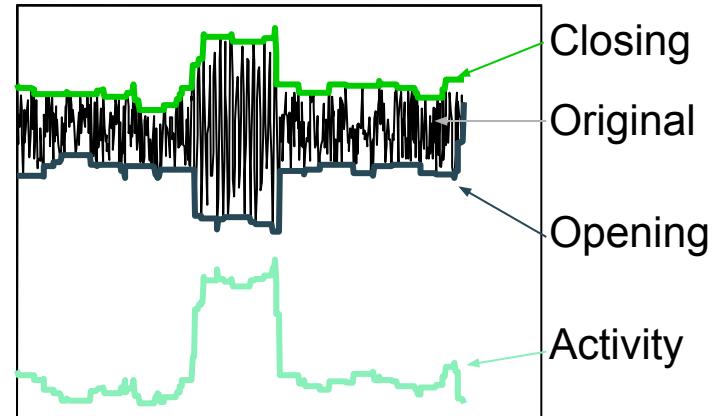
Frequency analysis



Segmentation: Feature extraction

Activity

- Gray level fluctuation
- Difference between opening and closing



Original



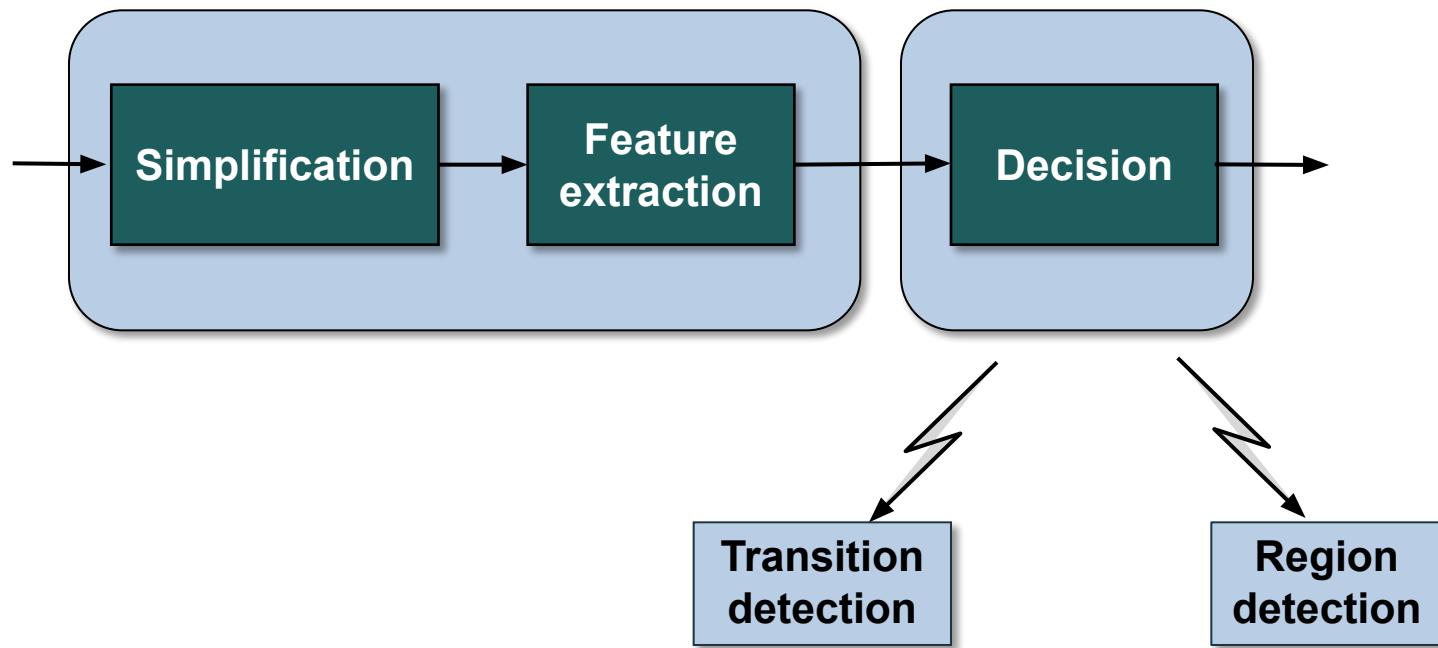
Closing



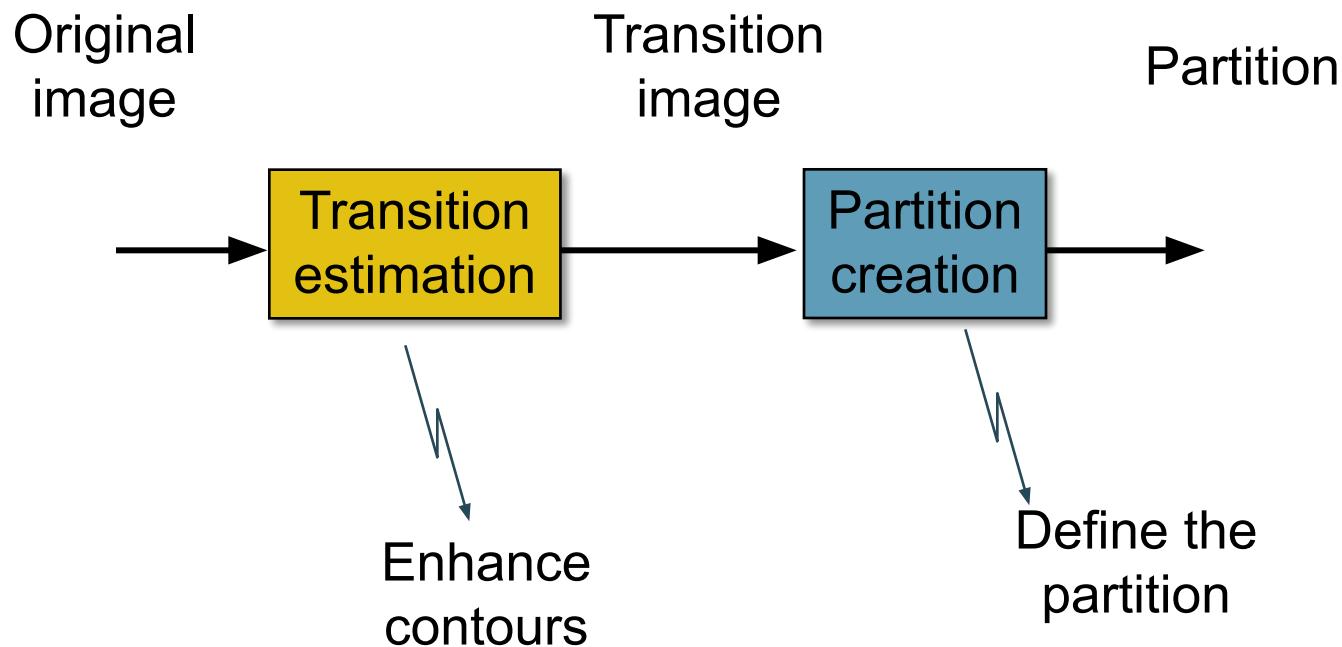
Activity
Opening



Segmentation: Overview

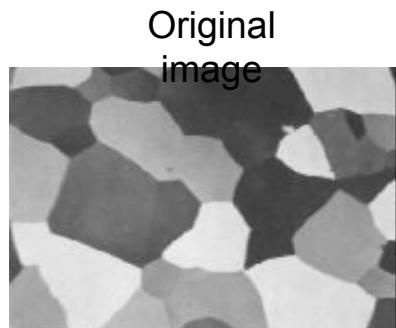


Segmentation: Transition based



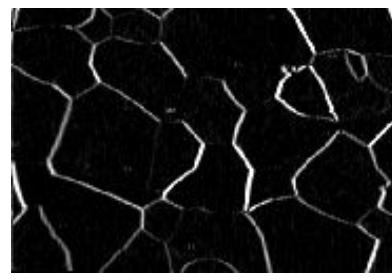
Segmentation: Transition based

Example: Sobel



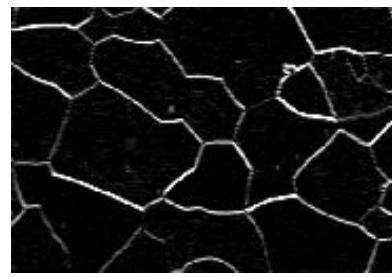
$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Vertical
contours

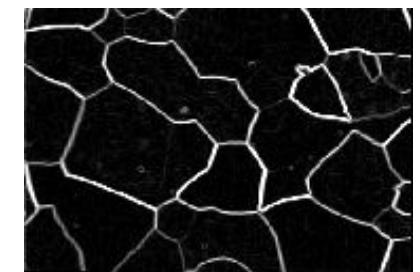


$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

Horizontal
contours



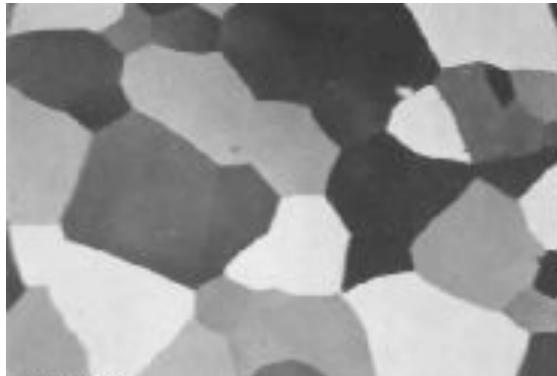
Combined information:
gradient



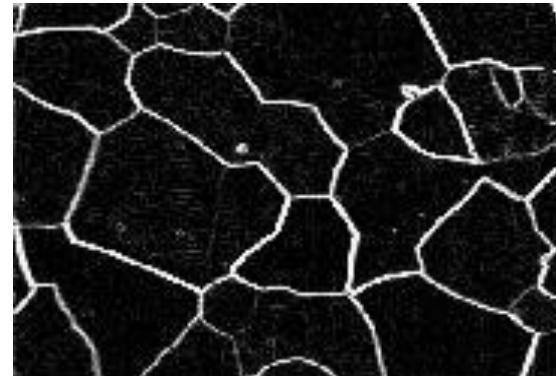
$$\text{Sqrt}(| h_1 * f |^2 + | h_2 * f |^2)$$

Segmentation: Transition based

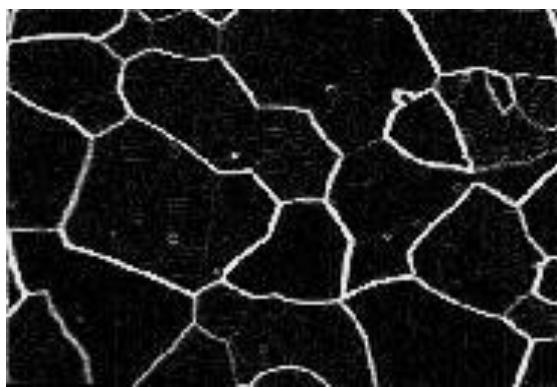
Example: Morphological gradient



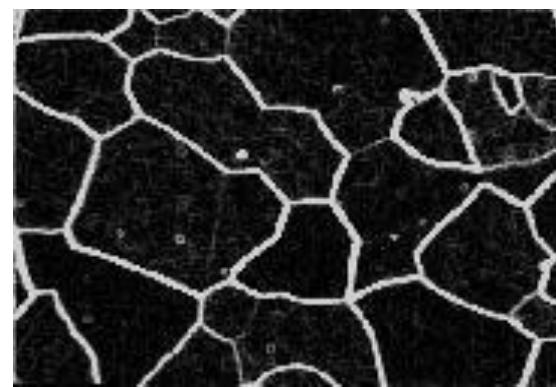
x



$x - (x \Theta b)$

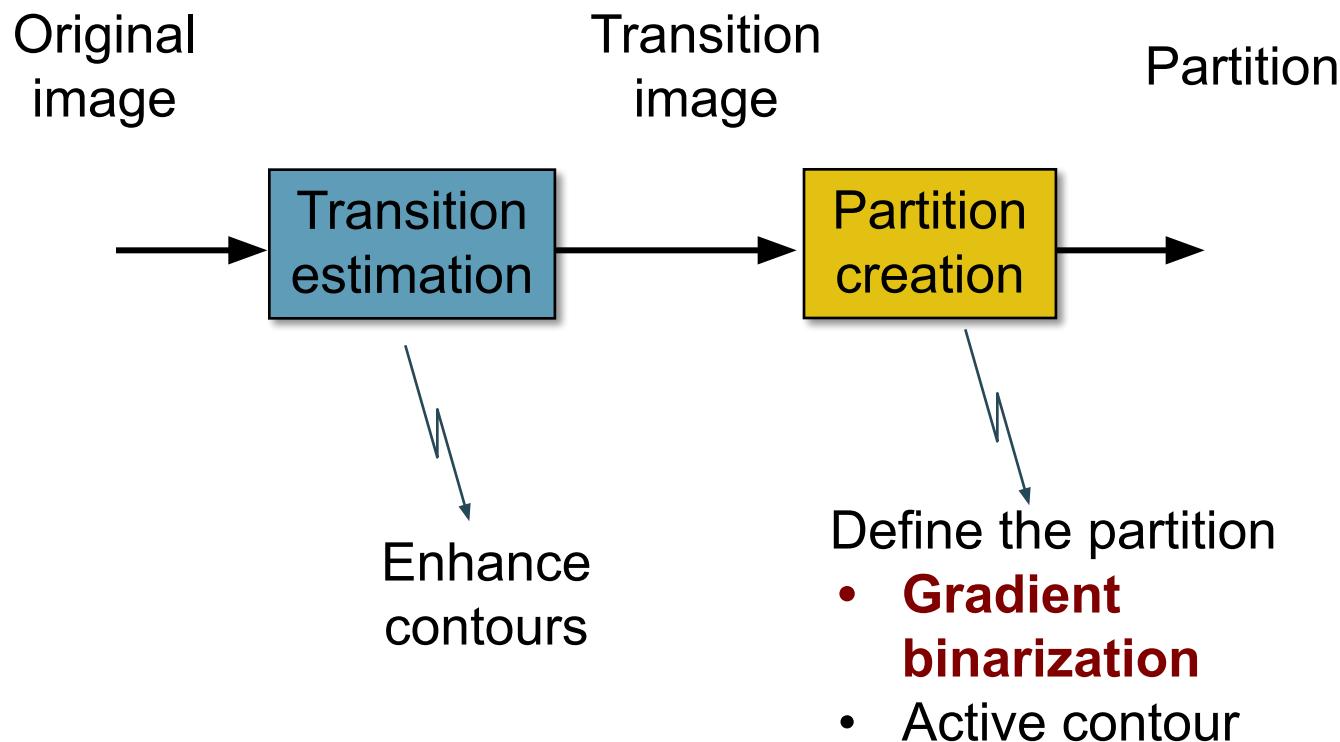


$(x \oplus b) - x$



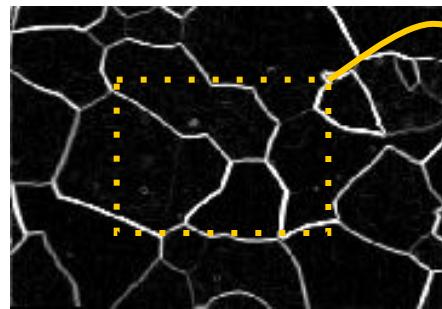
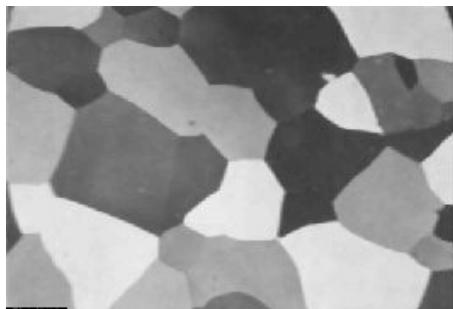
$(x \oplus b) - (x \Theta b)$

Segmentation: Transition based



Segmentation: Transition based

- Straightforward strategy: Gradient binarization



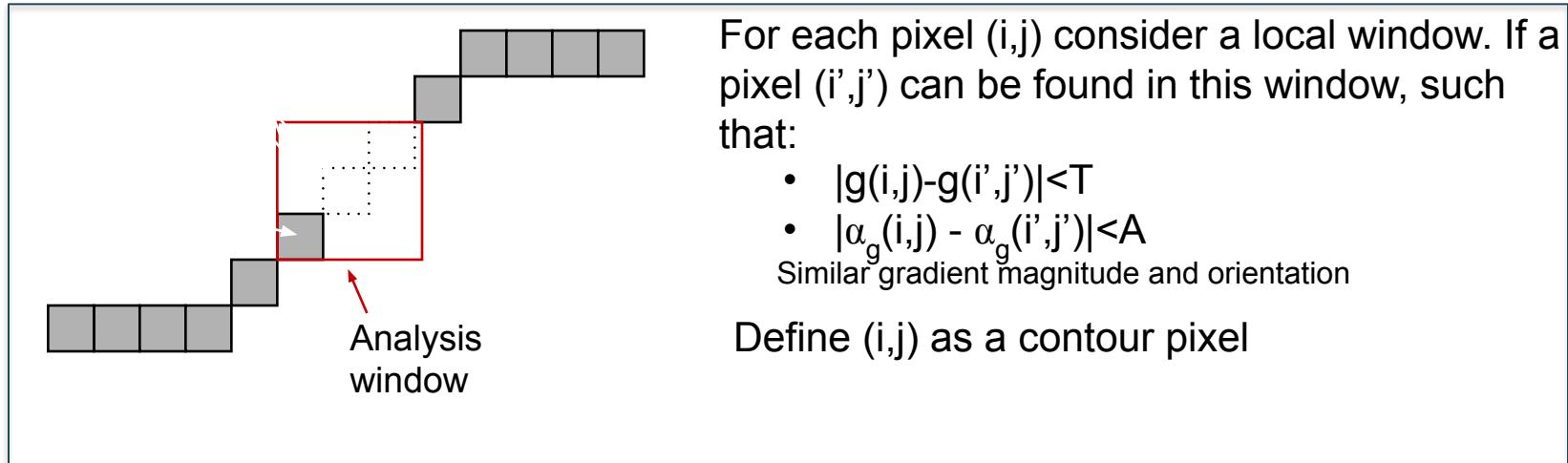
Issues to consider:

- Difficulty to define the threshold
- Contours may not be closed
- Noise may appear
- Contours may be thick

Segmentation: Transition based

Gradient binarization

- Contours may not be closed:
 - Morphological approach: Closing
 - Gradient extrapolation: Analyze gradient pixels at the end of contours

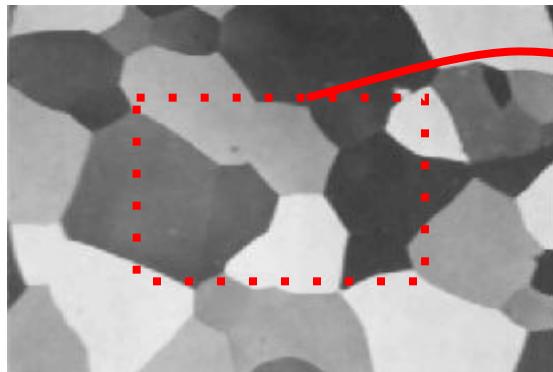


- Noise may appear:
 - Morphological approach: Opening
- Contours may be thick:
 - Remove pixels without breaking the contours: Thinning



Segmentation: Transition based

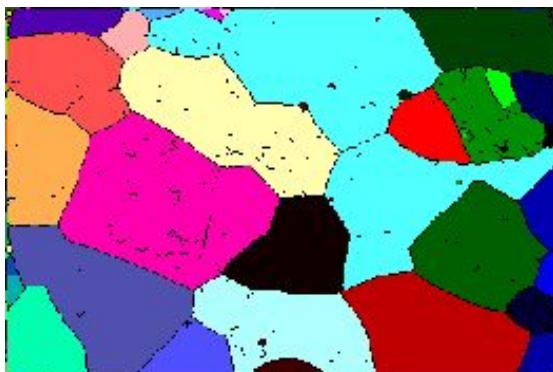
Gradient binarization



Original



Gradient
binarization



Labeling of regions



Closing + Thinning

Useful in simple cases, but lack of robustness