Shape, scene, and object recognition

IN4393 – Computer Vision

TU Delft

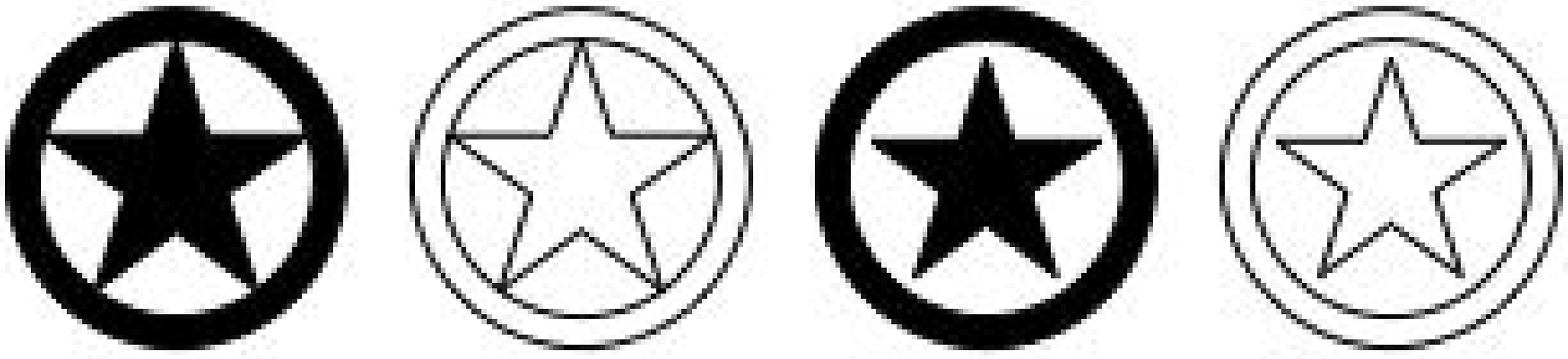
Shape recognition

Shape recognition

- *Region-based descriptors* describe region occupied by the shape
- *Contour-based descriptors* describe the contour of the shape
- How do the two types of descriptors differ on the following two shapes?

Shape recognition

- *Region-based descriptors* describe region occupied by the shape
- *Contour-based descriptors* describe the contour of the shape
- How do the two types of descriptors differ on the following two shapes?



Shape contexts

- Contour-based method to measure shape dissimilarity that works as follows:

1) Sample points from the contours of two shapes

2) Describe points using distance-angle histograms

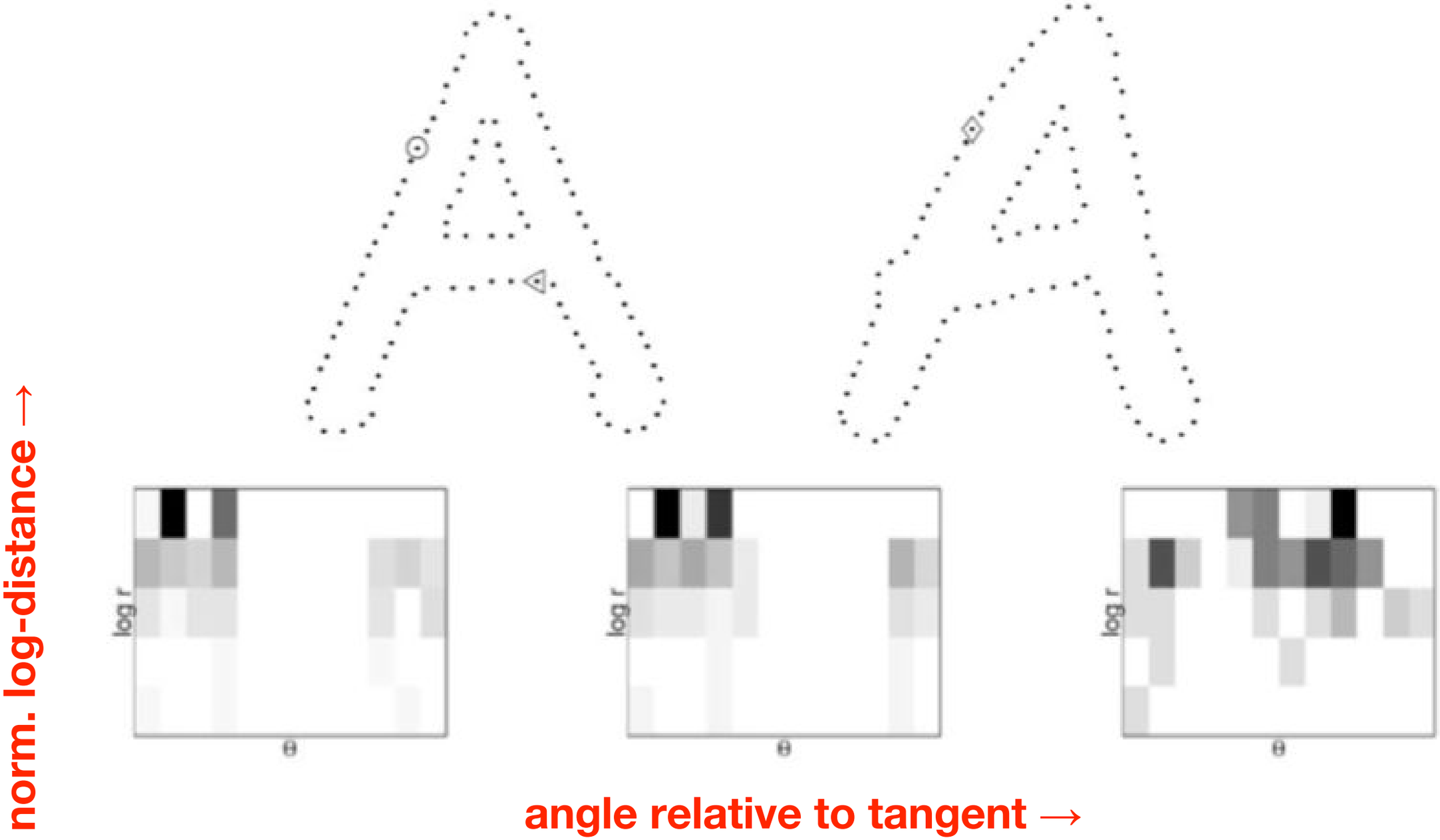
3) Match points by solving an assignment problem between descriptors

4) Warp target shape using the assignments; return to step 3)

5) Final shape dissimilarity is the energy of the warp plus the residual error

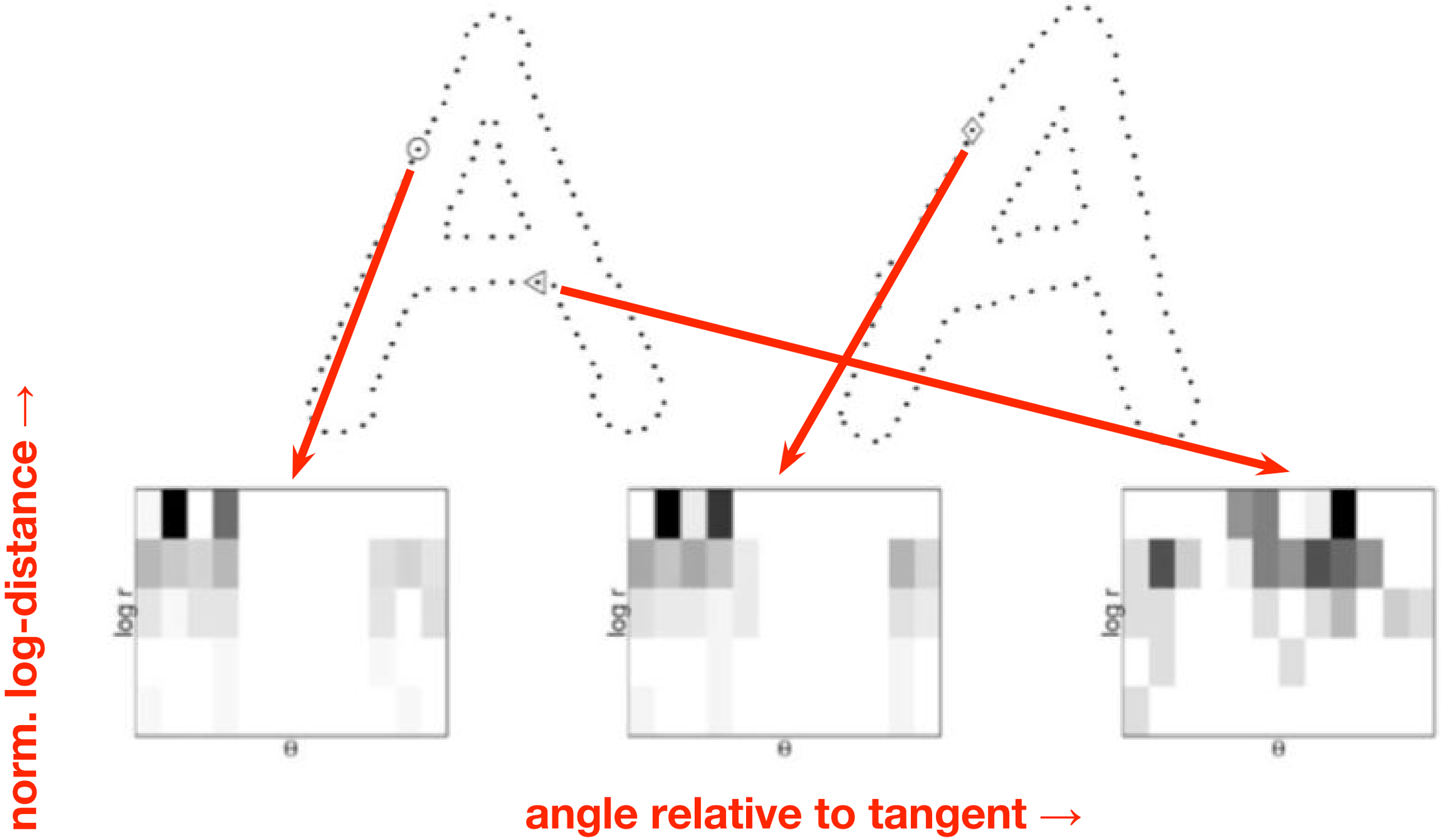
Shape contexts

- Histogram of *log-distance* and *relative angle* to all other points:



Shape contexts

- Histogram of *log-distance* and *relative angle* to all other points:



Shape contexts

- Compute distances between shape-context descriptors (Euclidean)

- Solve assignment problem between the two point sets (using Hungarian algorithm):

	Bathroom	Floor	Windows
Jim	1$	2$	3$
Steve	3$	3$	3$
Allan	3$	3$	2$

Shape contexts

- Compute distances between shape-context descriptors (Euclidean)

- Solve assignment problem between the two point sets (using Hungarian algorithm):

	Bathroom	Floor	Windows
Jim	1$	2$	3$
Steve	3$	3$	3$
Allan	3$	3$	2$

Optimal: Jim cleans bathroom, Steve cleans floor, and Allan cleans windows

Shape contexts

- Compute distances between shape-context descriptors (Euclidean)

- Solve assignment problem between the two point sets (using Hungarian algorithm):

	SC2.1	SC2.2	SC2.3
SC1.1	1	2	3
SC1.2	3	3	3
SC1.3	3	3	2

- We now know which points correspond to each other

Shape contexts

- Compute distances between shape-context descriptors (Euclidean)

- Solve assignment problem between the two point sets (using Hungarian algorithm):

	SC2.1	SC2.2	SC2.3
SC1.1	1	2	3
SC1.2	3	3	3
SC1.3	3	3	2

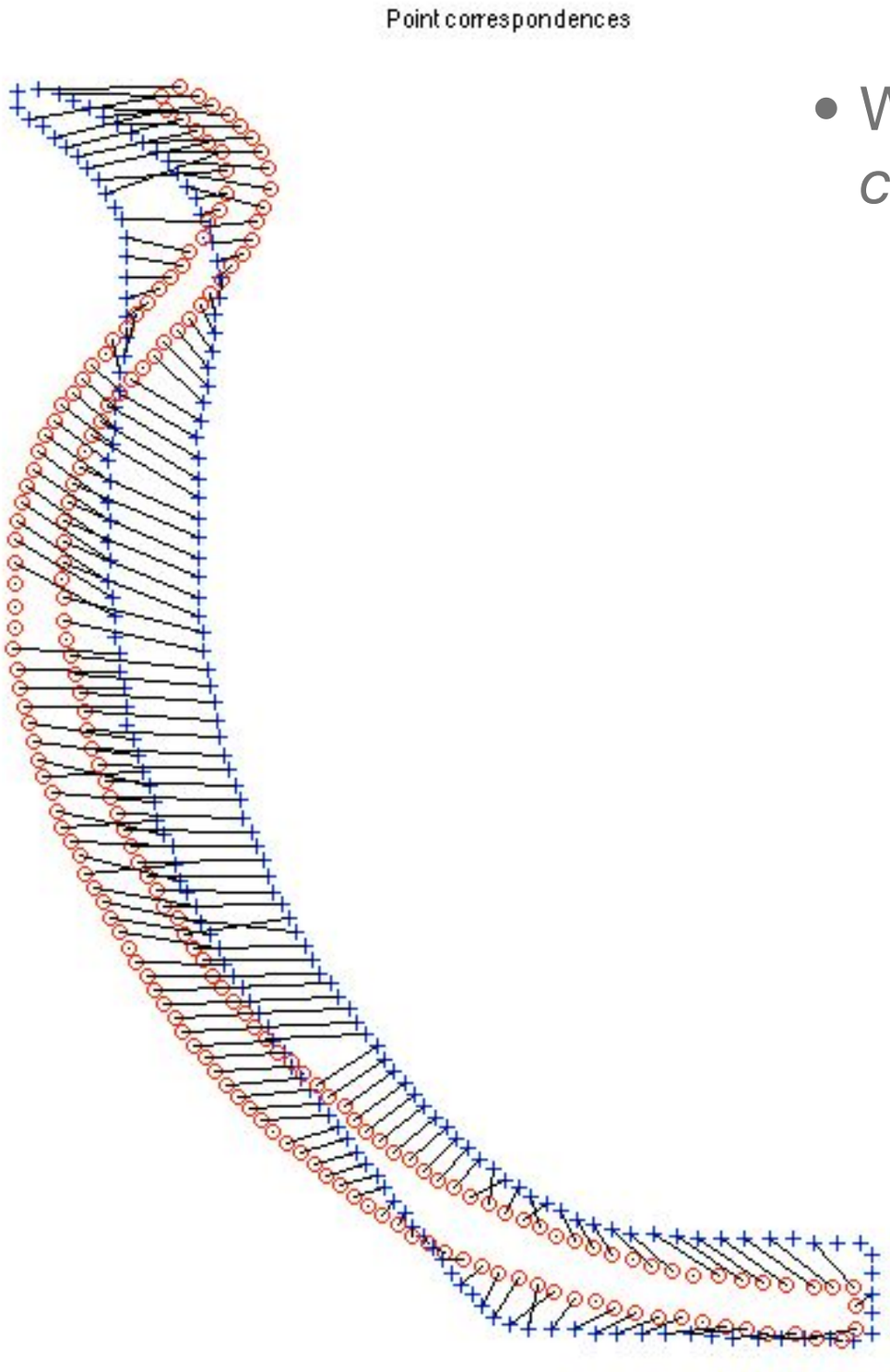
- We now know which points correspond to each other

  - We could have used RANSAC to find the correspondences

*RANSAC (Random Sample Consensus)

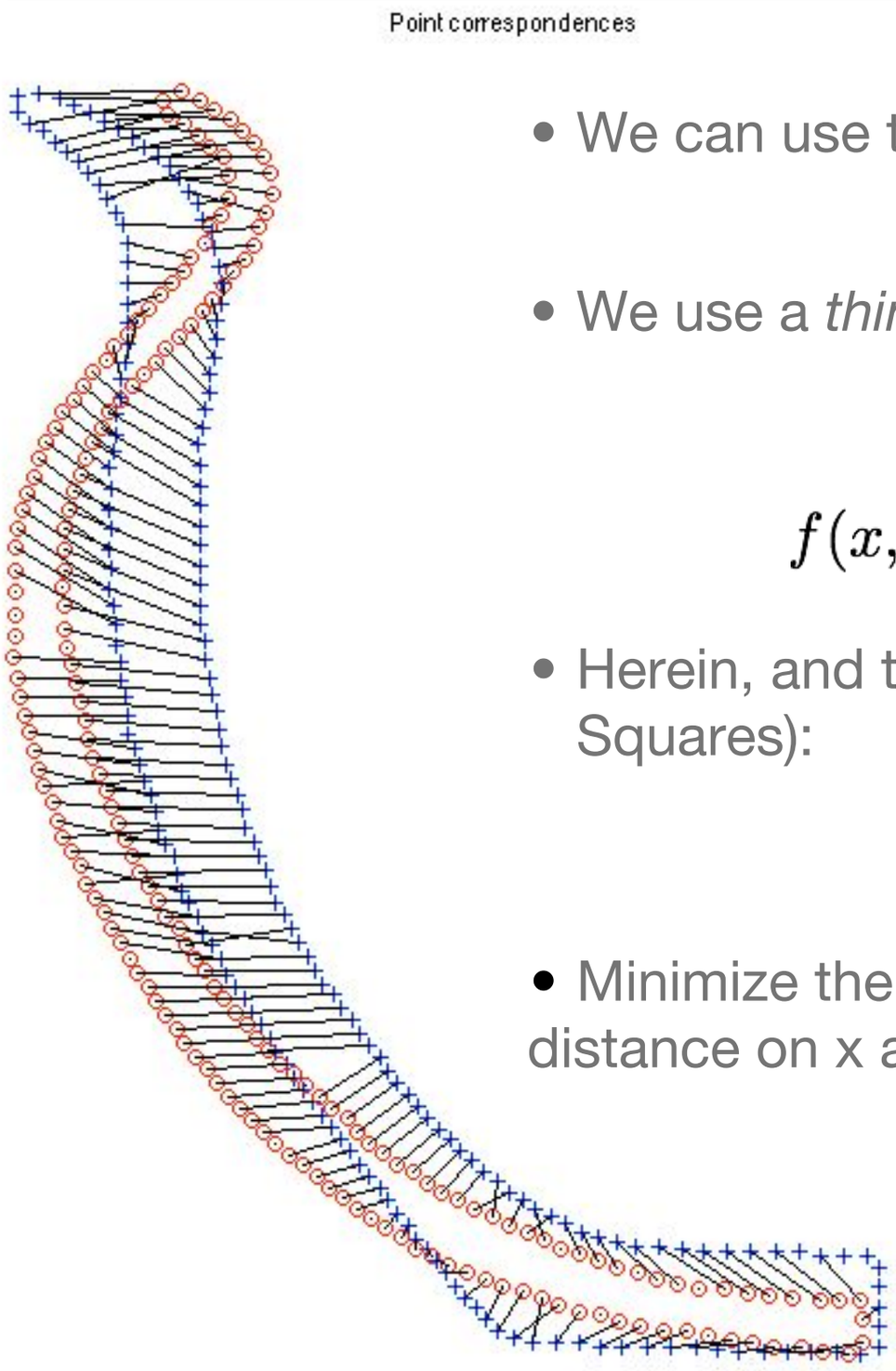
1. Iteratively select the minimum set of correspondences needed to learn the transformation or fit a model.
2. Estimate how well the other correspondences fit the transformation/model.
3. Keep the transformation/model that fits the most correspondences.

Shape contexts



- We can use the correspondences as *control points* to find a *warp.*

Shape contexts



- We can use the correspondences as *control points* to find a *warp*

- We use a *thin-plate spline warp*:

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{K} w_i U(\|(x_i,y_i) - (x,y)\|)$$

- Herein, and the parameters are learned via LLS (Linear Least Squares):

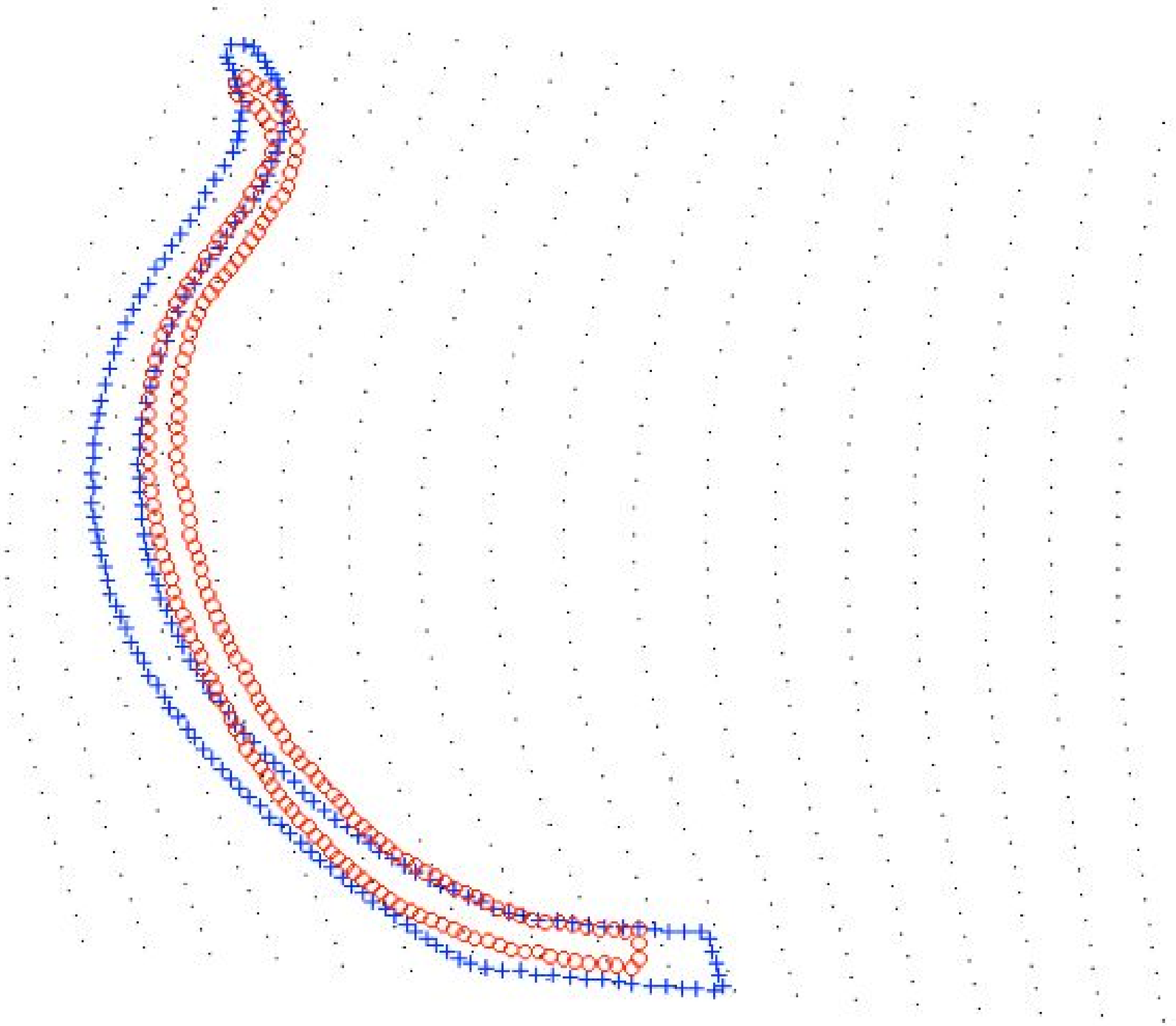
$$U(r) = r^2 \log r^2$$

- Minimize the squared distance on x and y:

$$\min \sum_{i=1}^{K} (x'_i - f_x(x_i,y_i))^2$$

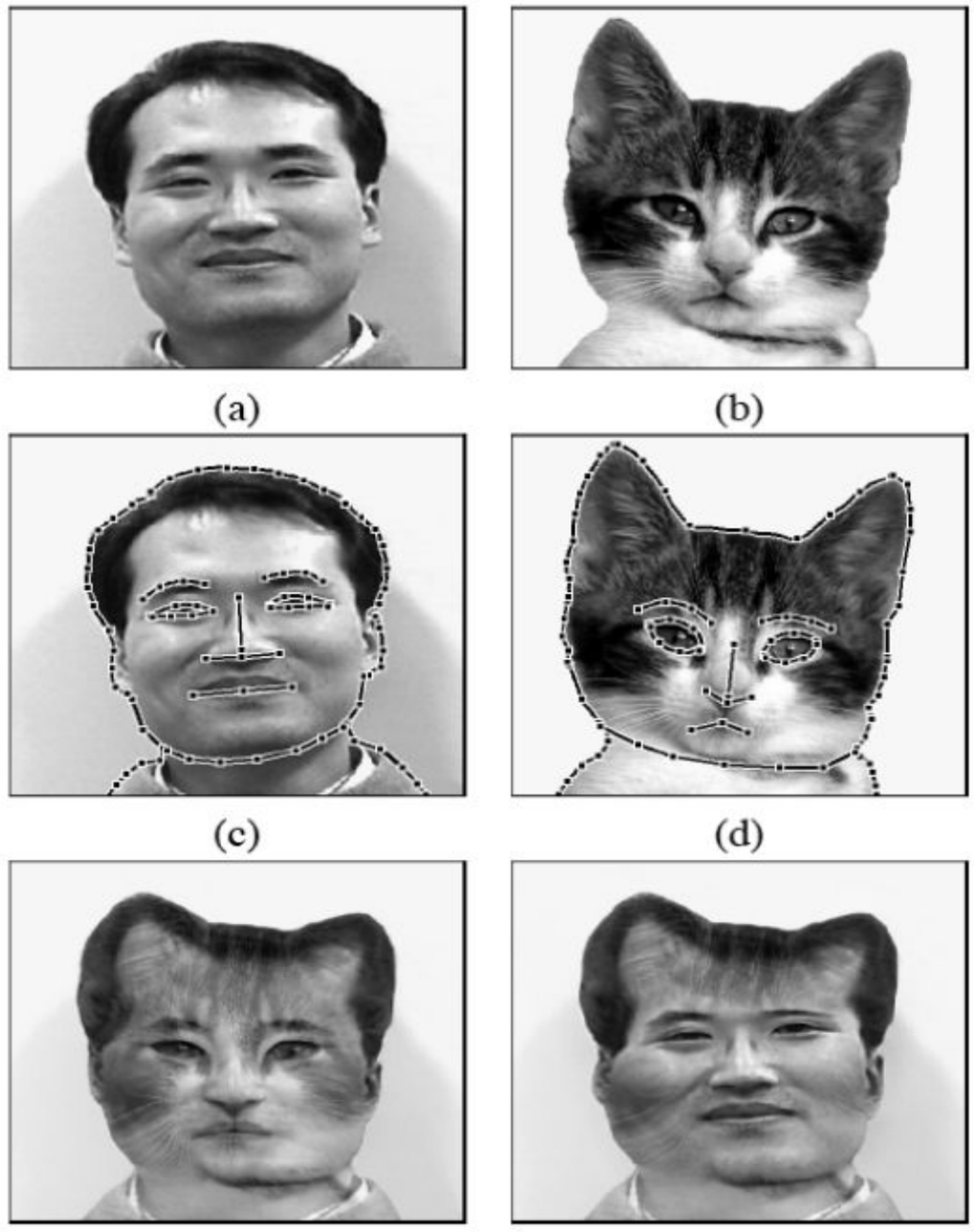
$$\min \sum_{i=1}^{K} (y'_i - f_y(x_i,y_i))^2$$

Shape contexts



Shape contexts

Thin-plate spline warping is also used for image morphing.



https://alliance.seas.upenn.edu/~cis581/wiki/Lectures/CIS581-10-10-thinplate.pdf

Shape contexts

- Chicken-and-egg problem, the warp changed the point correspondences:
    - Iterate for a few iterations or until convergence

Shape contexts

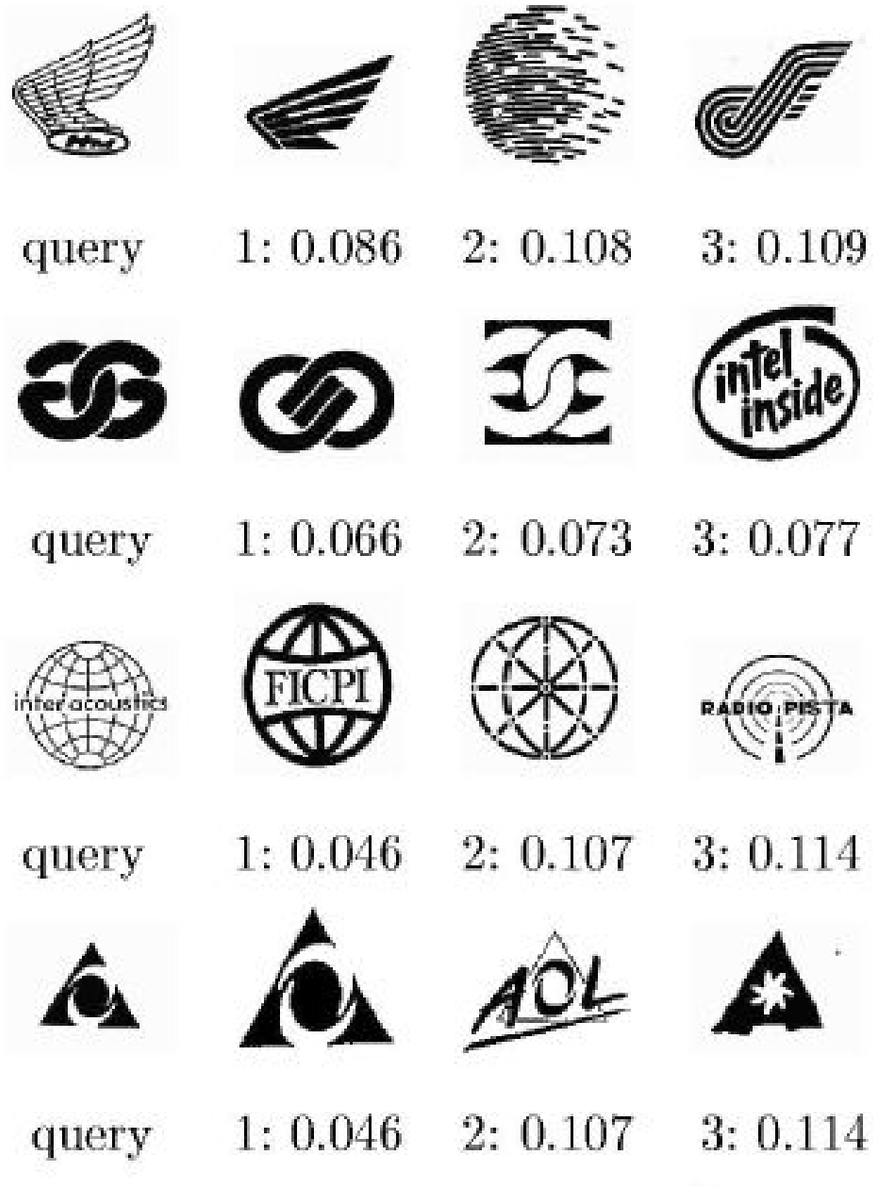
- Chicken-and-egg problem, the warp changed the point correspondences:
  - Iterate for a few iterations or until convergence
- The final dissimilarity between the two shapes is now a sum of:
  - The sum of distances between corresponding points after the final warp
  - The bending energy of the final warp

Shape contexts

- Chicken-and-egg problem, the warp changed the point correspondences:
  - Iterate for a few iterations or until convergence
- The final dissimilarity between the two shapes is now a sum of:
  - The sum of distances between corresponding points after the final warp
  - The bending energy of the final warp
- Note that shape contexts are translation-, rotation-, and scale-invariant

Shape contexts

- Example of using shape contexts to retrieve logos:



- Example of using shape matching from 2D to 3D:

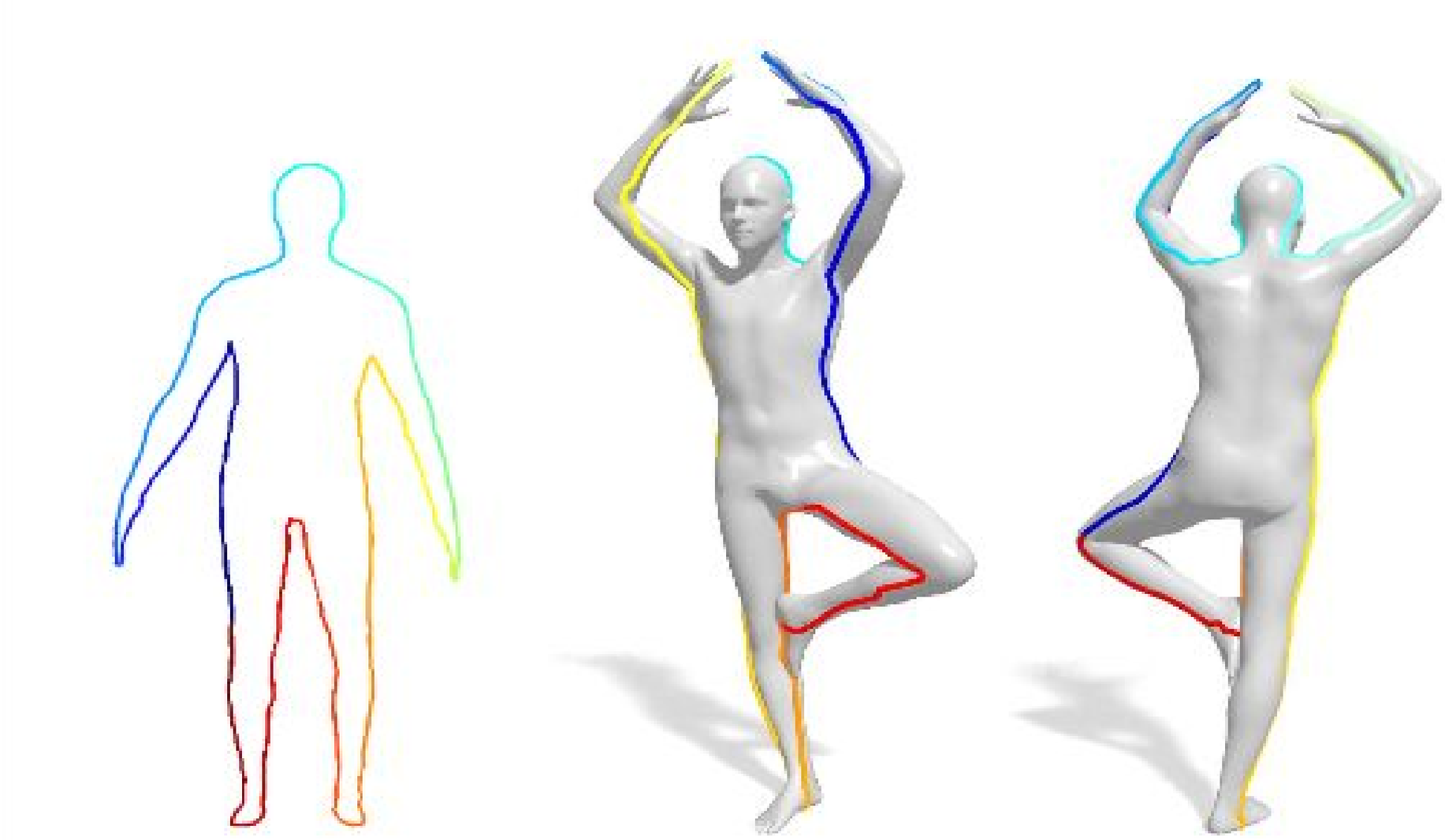


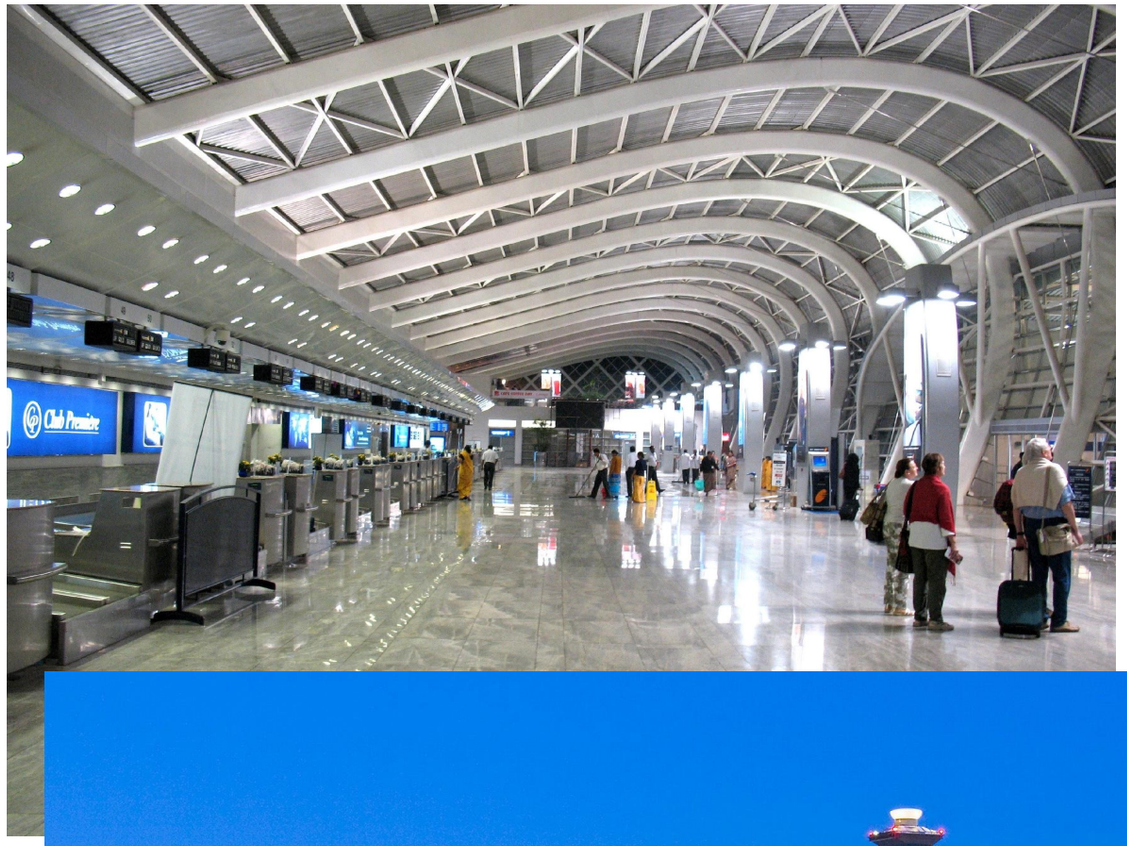
Figure 1. We propose the first shape matching method between a 2D query shape (left) and a 3D target shape (right), both of which are allowed to deform non-rigidly. The globally optimal matching (shown on top of the 3D target) is guaranteed to be continuous.

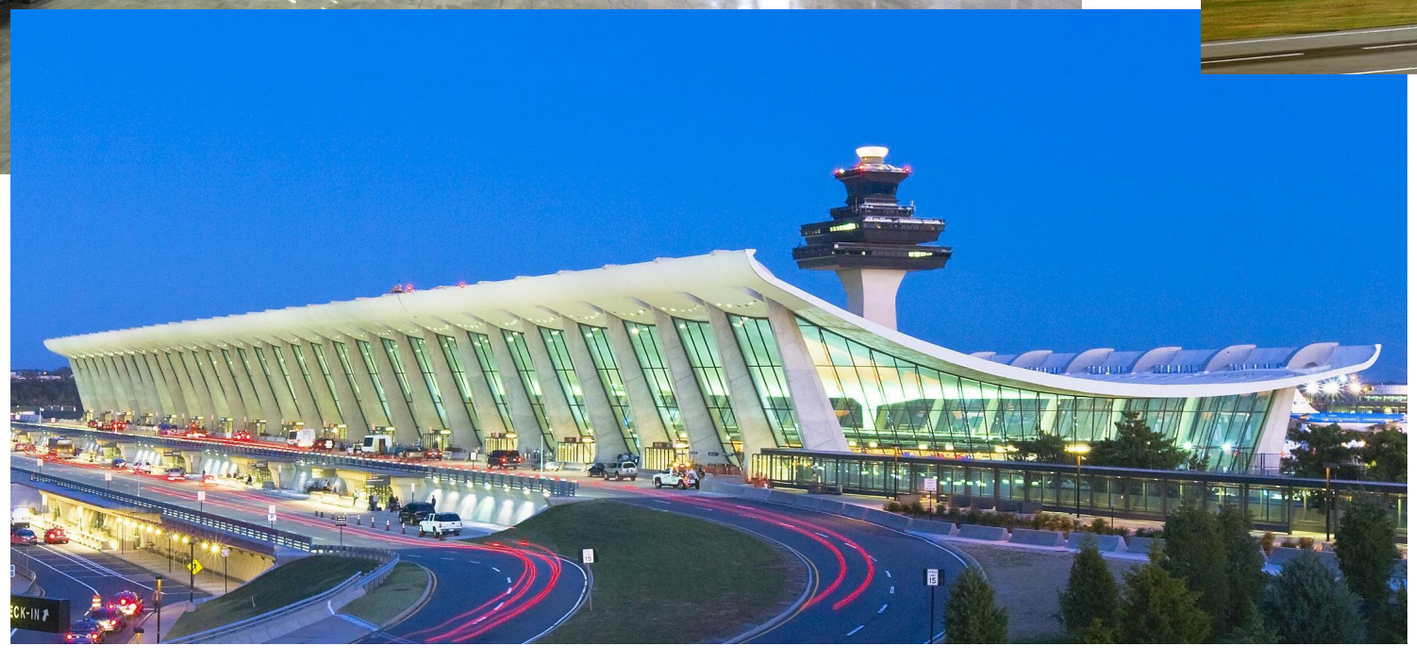
Z Lahner, et al. "Efficient globally optimal 2d-to-3d deformable shape matching." CVPR. 2016.

Scene recognition

Scene recognition

- *Scenes* are constellations of different objects in somewhat arbitrary locations:

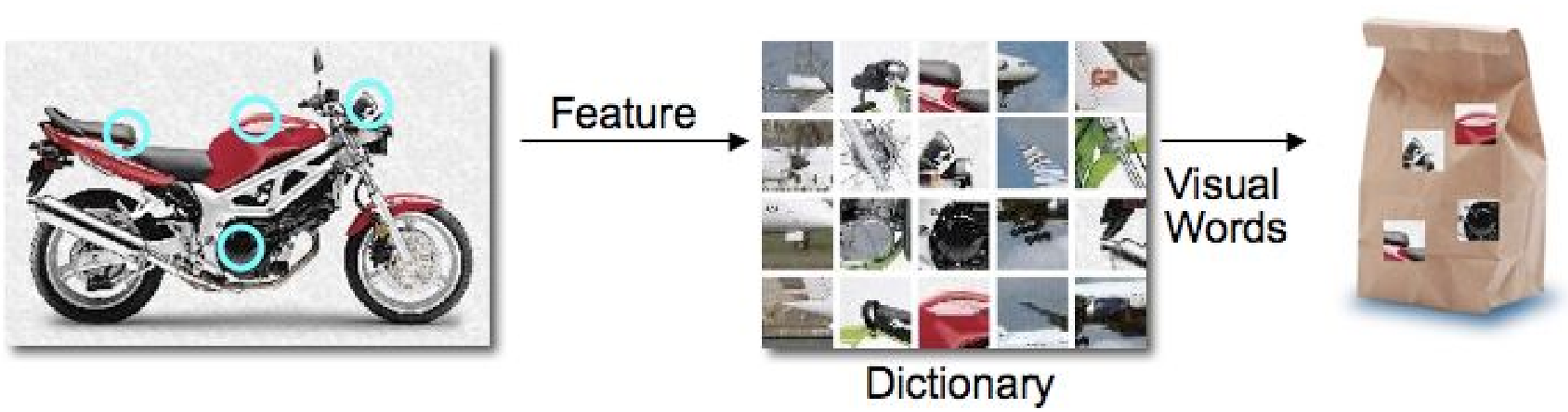




Bag of visual words

- We have already seen descriptors for local image patches (*e.g.*, SIFT).

- Bag-of-visual-words features consider an image as a *bag* of image patches, *ignoring* the spatial relations between those images:



- How do we obtain the *dictionary* of visual words?

Bag of visual words

- Construction of bag-of-visual-words features proceeds in four main steps:

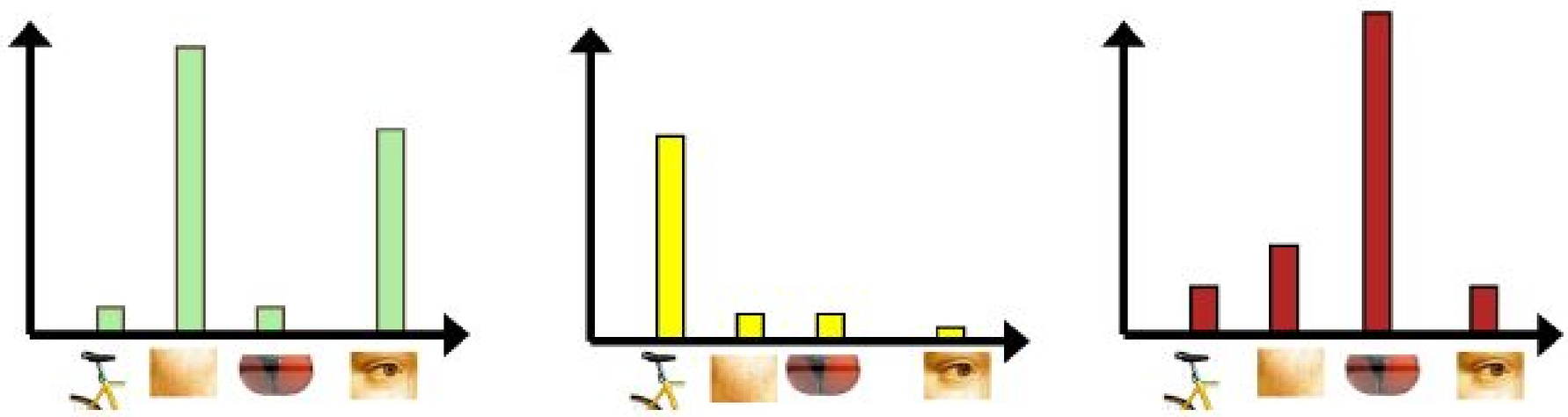
1. Gather a collection of randomly selected image patches

2. Perform k-means clustering on the image patches (using some descriptor for the patch) to obtain a *codebook* or *dictionary* of visual words

3. Gather image patches from each image (using keypoint detector or dense sampling)

4. Describe each image by counting how often each word appears in the image

Bag of visual words



K-means algorithm

- The *k*-means algorithms finds clusters via an iterative process:
  - Given *K* cluster centers, determine current assignments
  - Given cluster assignments, compute the *K* cluster centers
- This procedure minimizes the sum of squared Euclidean distances between each point and its corresponding cluster center ($c_n \in \{1, \ldots, K\}$):

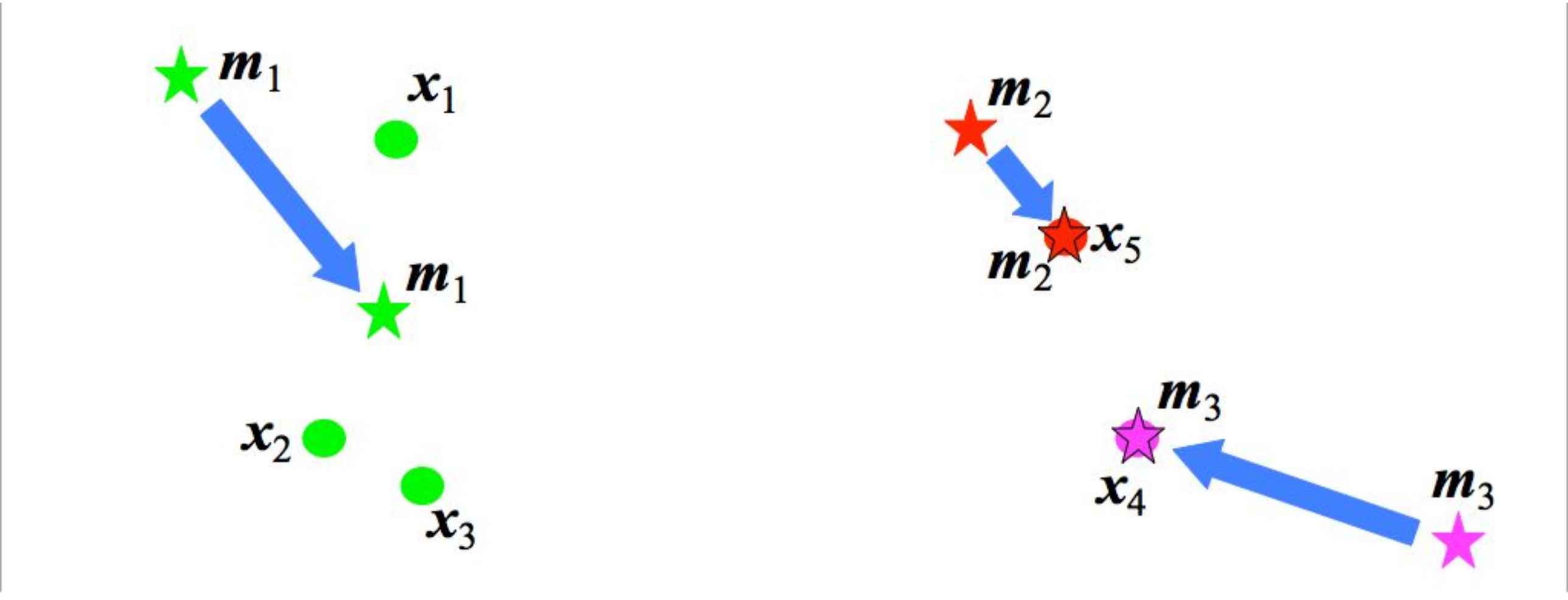
$$\sum_{n=1}^{N} \| \mathbf{x}_n - \boldsymbol{\mu}_{c_n} \|^2$$

K-means algorithm

K-means algorithm

K-means algorithm

K-means algorithm



K-means algorithm

K-means algorithm

- Overall objective: $$\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

K-means algorithm

- Overall objective: $\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$

centers of all *K* clusters

K-means algorithm

- Overall objective: $\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$

centers of all *K* clusters

cluster assignments: vector of length *N* with values between *1* and *K*

K-means algorithm

- Overall objective: $$\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

centers of all *K* clusters

cluster assignments: vector of length *N* with values between *1* and *K*

minimize squared Euclidean distance between each data point and its corresponding cluster center

K-means algorithm

- Overall objective: $$\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

- Step 1: Assign data to clusters: $$\min_{\mathbf{c}} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

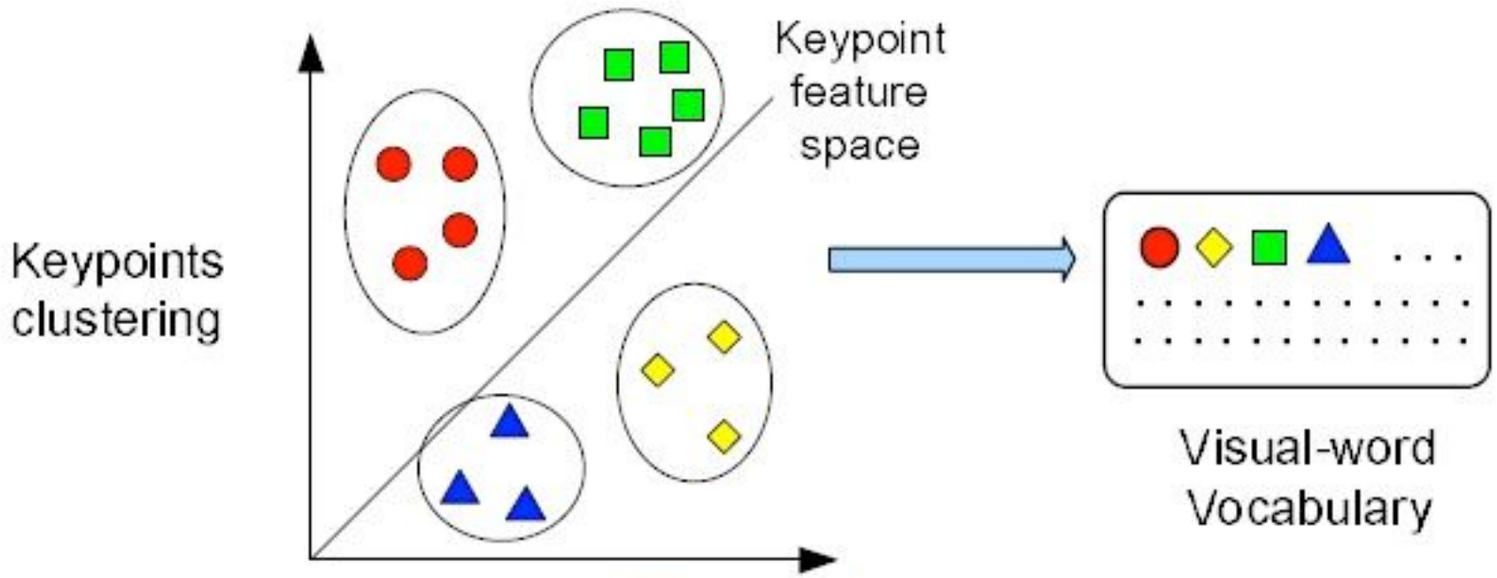
K-means algorithm

- Overall objective: $$\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K,\mathbf{c})} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

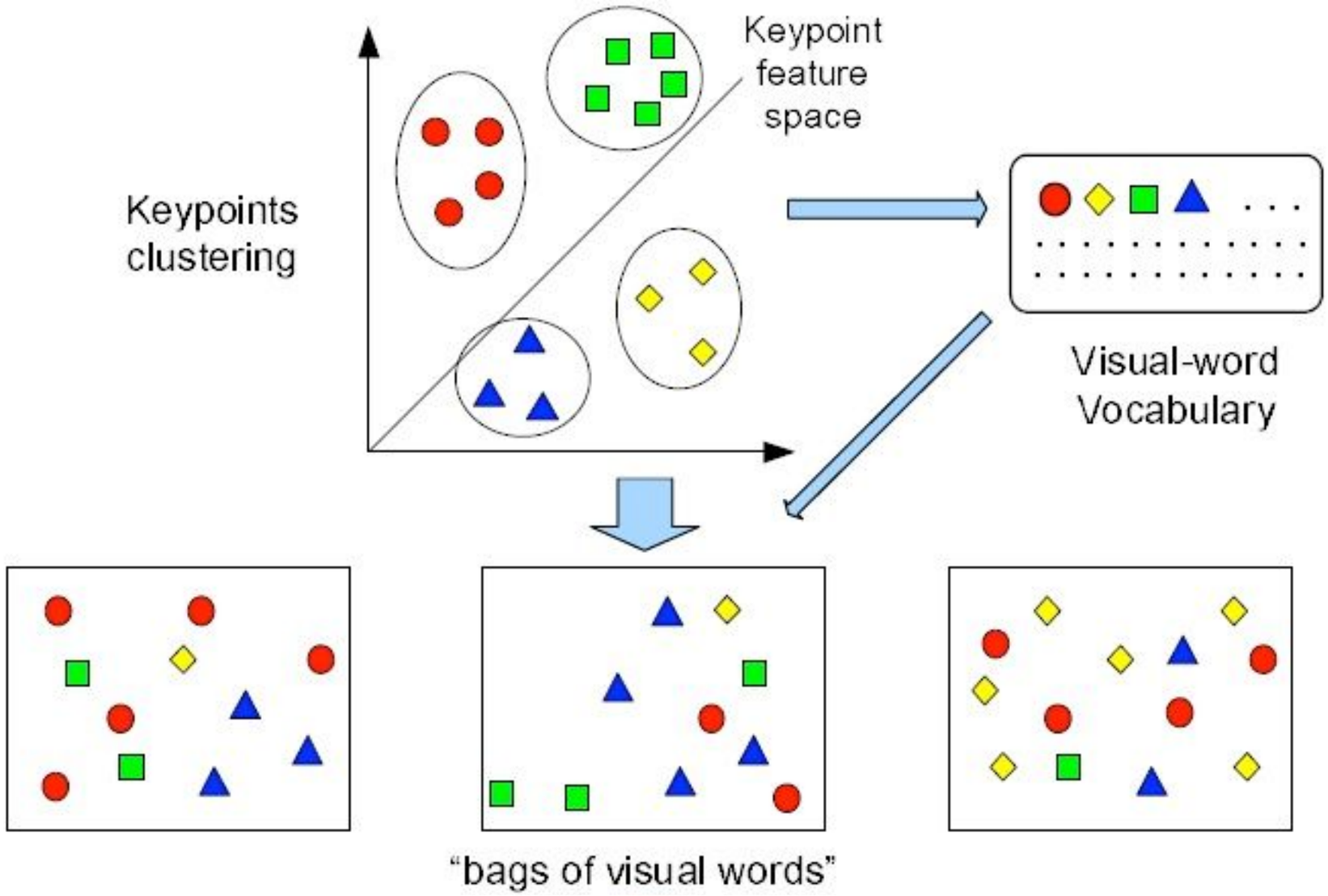
- Step 1: Assign data to clusters: $$\min_{\mathbf{c}} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

- Step 2: Compute cluster means: $$\min_{(\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K)} \sum_{n=1}^{N} \|\mathbf{x}_n - \boldsymbol{\mu}_{c_n}\|^2$$

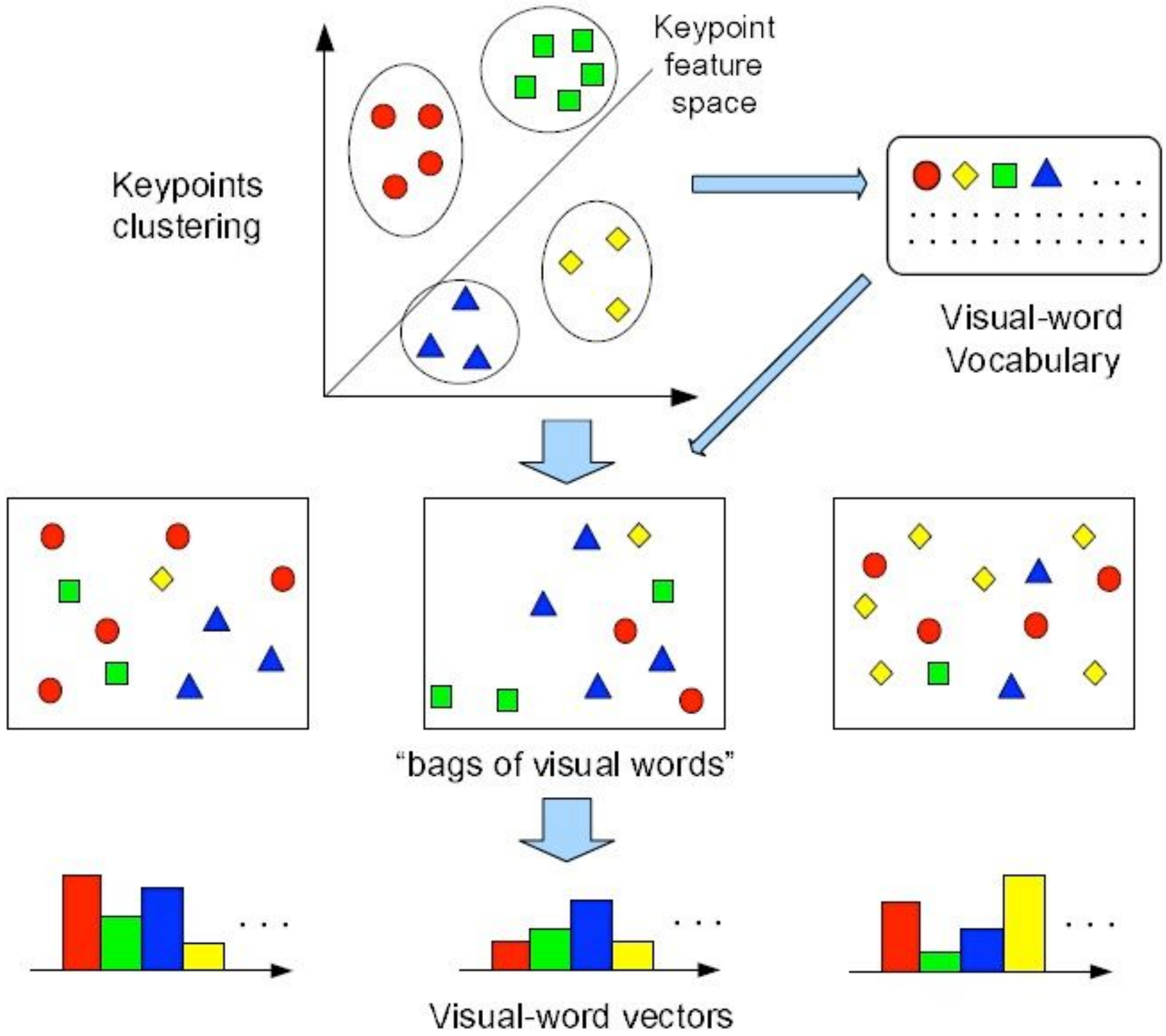
Bag of visual words



Bag of visual words

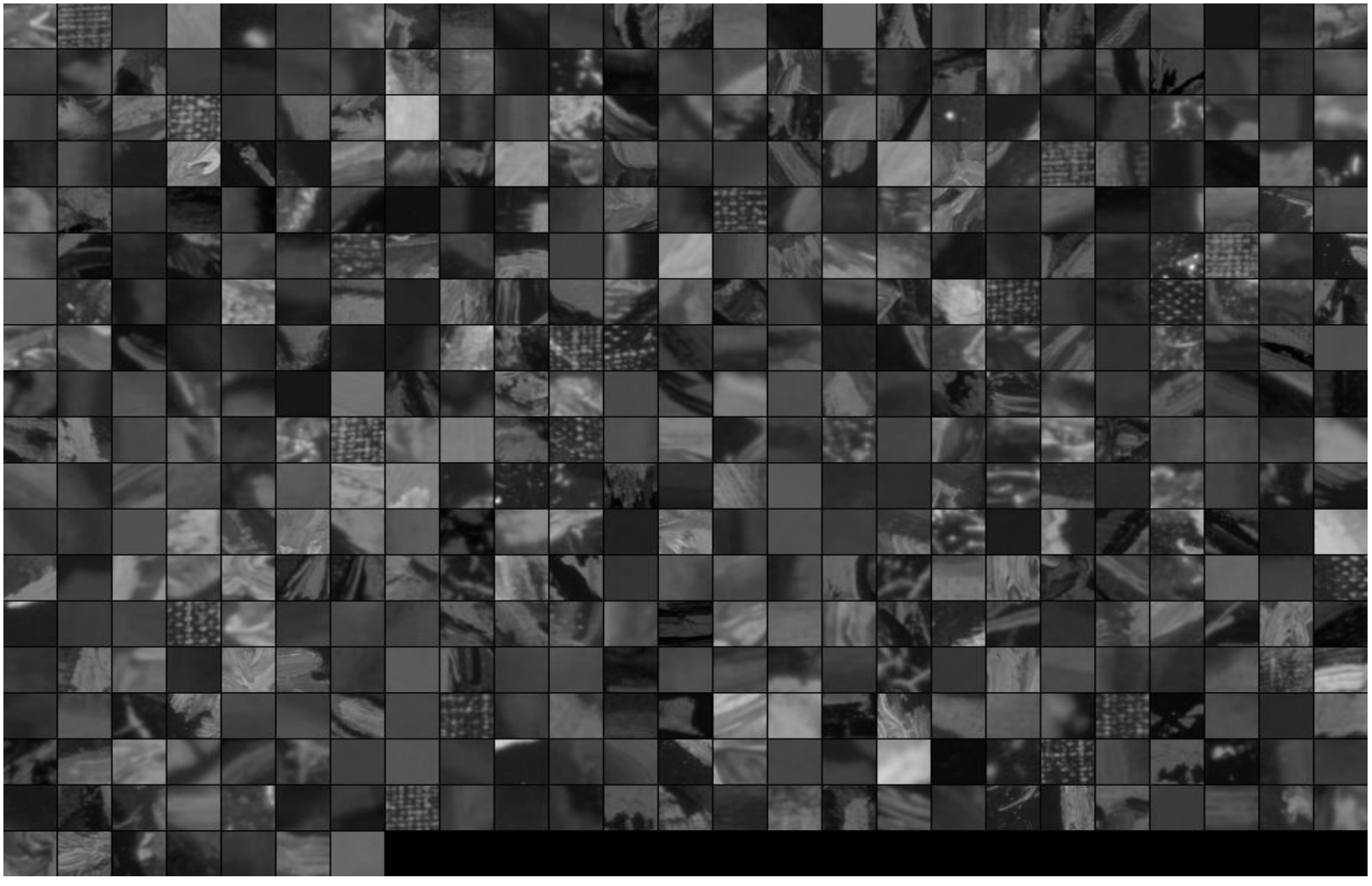


Bag of visual words



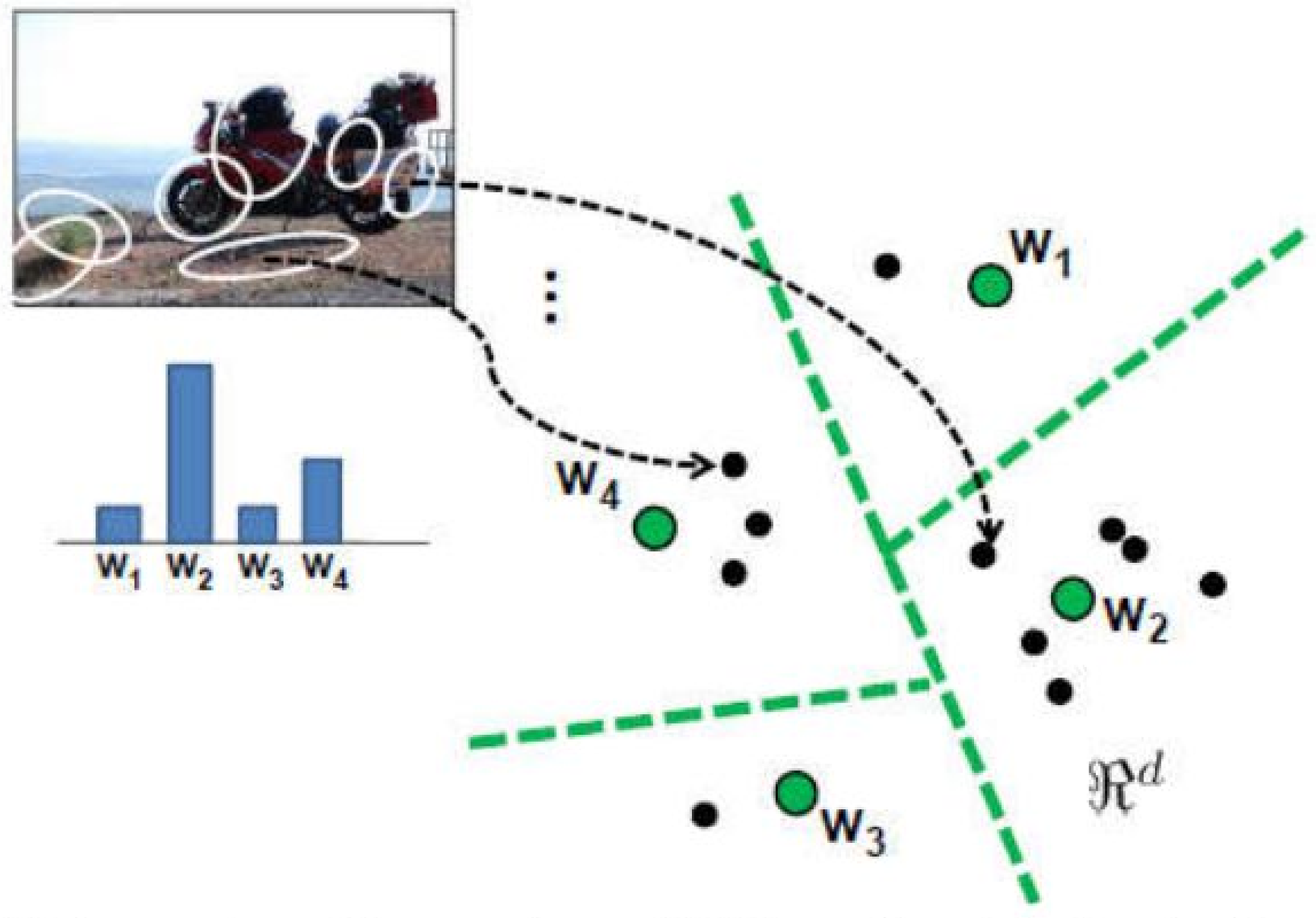
Bag of visual words

- Example of a *(texton) codebook* build using k-means on small patches:



Hard assignment

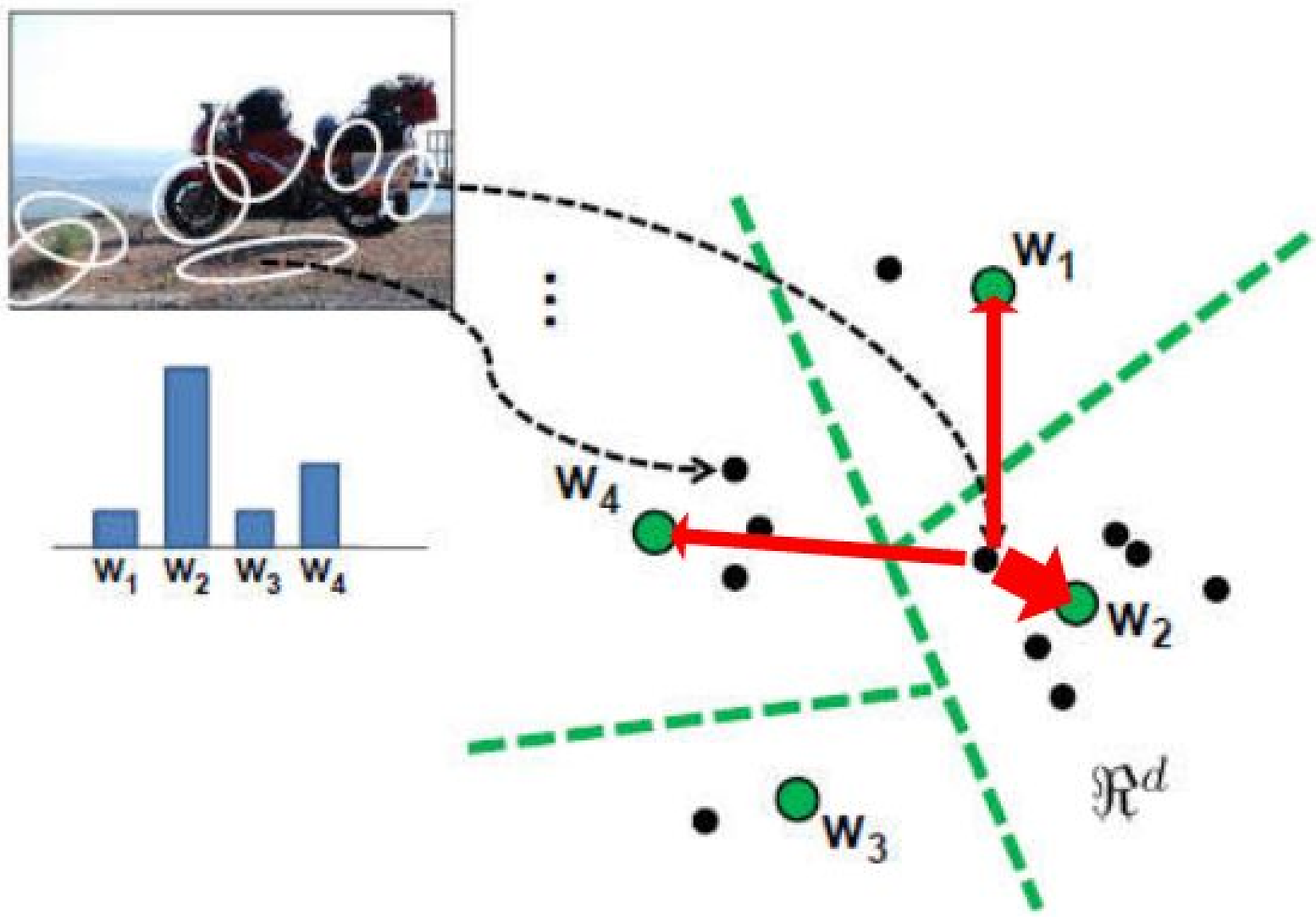
- Each descriptor is assigned to 1 codeword, only:



http://www.cs.utexas.edu/~grauman/courses/fall2009/papers/bag_of_visual_words.pdf

Soft assignment

- Each descriptor is assigned to a weighted combination of codewords:



Soft assignment

- A gaussian kernel is set over each codeword (similar to a Gaussian Mixture Model):

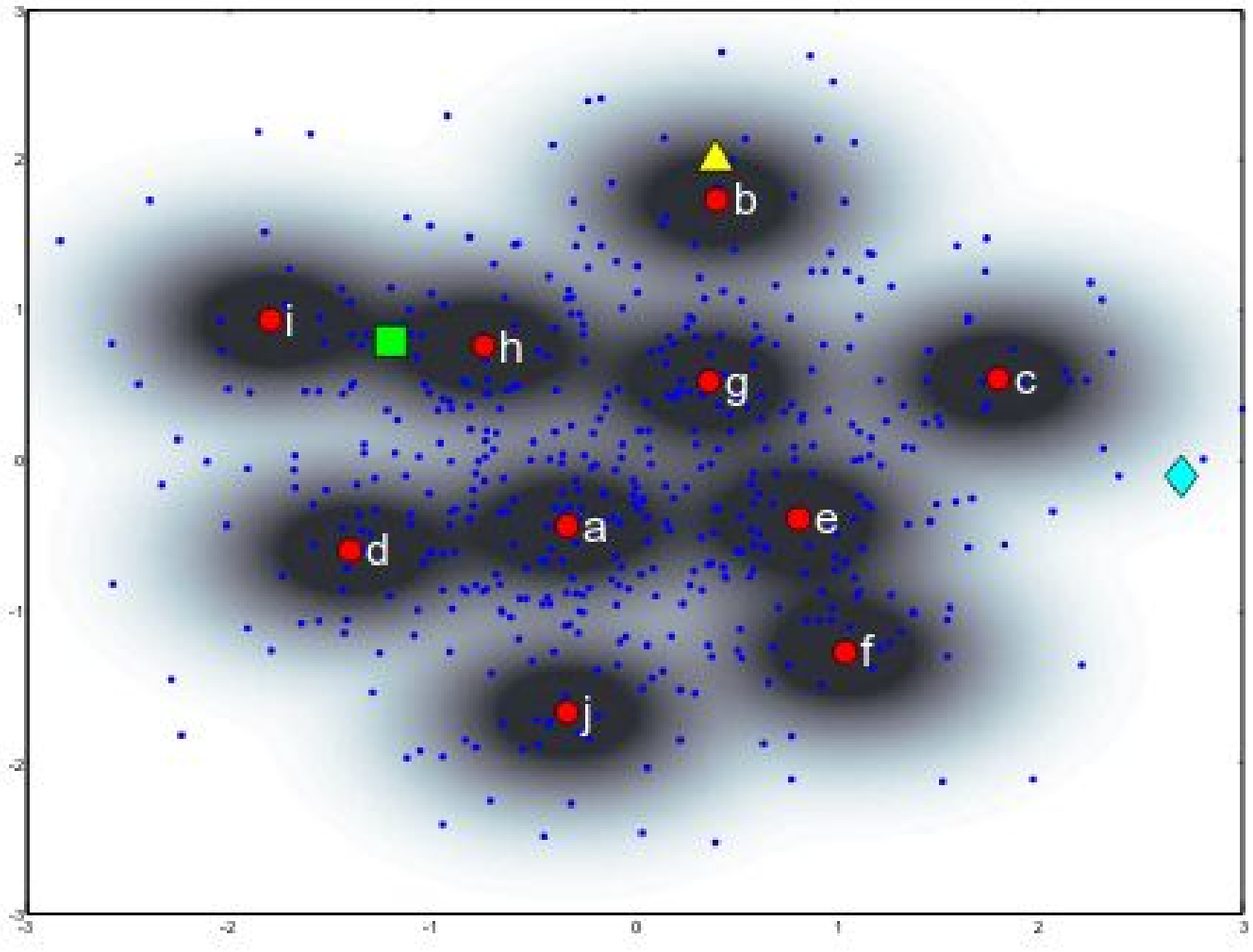
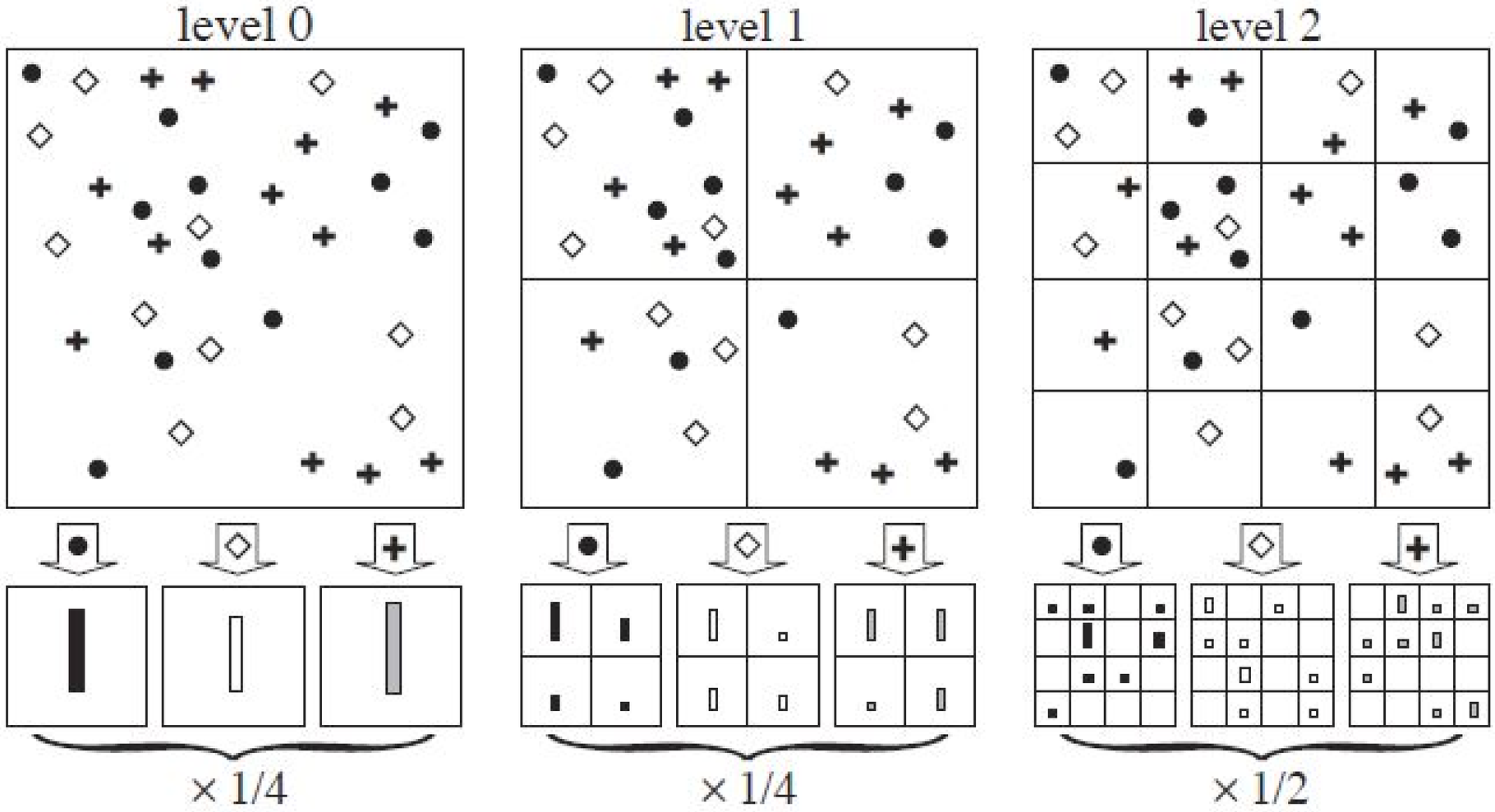


Fig. 2. An example of the weight distribution of a kernel codebook with a Gaussian kernel, where the square, diamond and triangle represent the image features taken from figure 1.

J. C. van Gemert, et al. "Visual word ambiguity." PAMI. 2010.

Bag of visual words

- Spatial *pyramid matching* incorporates spatial structure in bag-of-words:



- Histograms are constructed per segment (on multiple scales).

VLAD embedding (pooled image descriptors)

- Adds rough second-order statistics (not only the mean but also the deviation from the mean):

$$q : \mathbb{R}^d \rightarrow \mathcal{C} \subset \mathbb{R}^d \tag{1}$$

$$\mathbf{x} \mapsto q(\mathbf{x}) = \arg\min_{\boldsymbol{\mu} \in \mathcal{C}} \|\mathbf{x} - \boldsymbol{\mu}\|^2, \tag{2}$$

$$\boldsymbol{v}^i = \sum_{\mathbf{x}: q(\mathbf{x}) = \boldsymbol{\mu}_i} \mathbf{x} - \boldsymbol{\mu}_i. \tag{3}$$

The concatenation $\boldsymbol{v} = [\boldsymbol{v}^1 \dots \boldsymbol{v}^k]$ is a D-dimensional vector, where $D = k \times d$.

J. Herve, et al. "Aggregating local descriptors into a compact image representation." CVPR 2010.

Fisher embedding (pooled image descriptors)

- Adds second and third-order statistics by setting assuming a GMM (Gaussian Mixture Model):

$$p(x|\lambda) = \sum_{i=1}^{N} w_i p_i(x|\lambda) = \sum_{i=1}^{N} w_i \mathcal{N}(x|\mu_i, \Sigma_i). \quad (\Sigma_i) = ((\sigma_i^1)^2, \ldots, (\sigma_i^D)^2)$$

$$\gamma_i(x_t) = \frac{w_i p_i(x_t|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_t|\lambda)}.$$

then the statistics are:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \left(\gamma_t(k) - w_k\right), \tag{16}$$

$$\mathcal{G}_{\mu_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{x_t - \mu_k}{\sigma_k}\right), \tag{17}$$

$$\mathcal{G}_{\sigma_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} \left[\frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1\right]. \tag{18}$$

J Sánchez, et al. "Image classification with the fisher vector: Theory and practice." IJCV 2013.

NetVlad

- A more new deep learning approach:

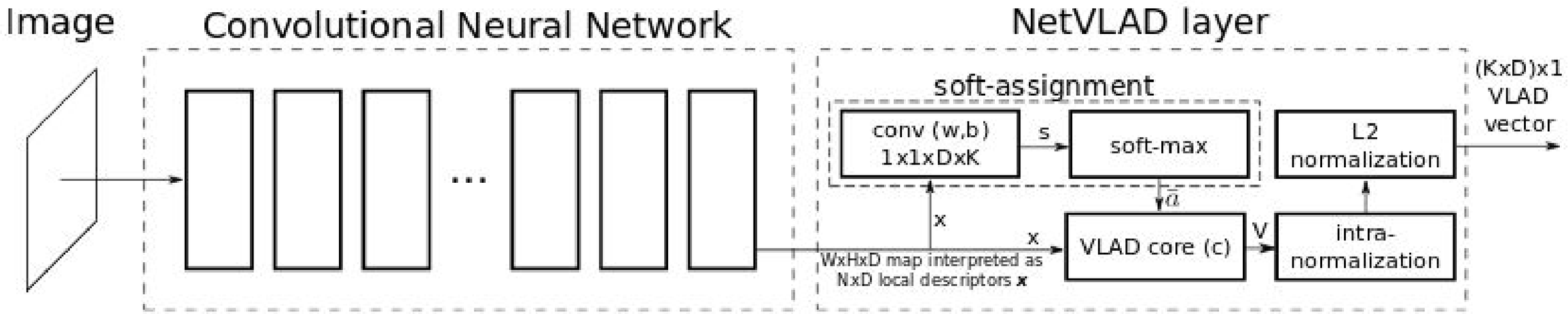
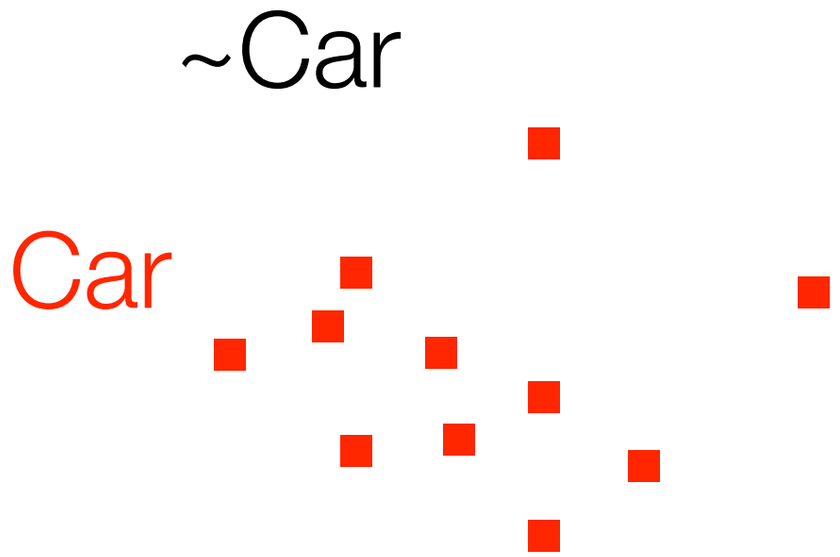


Figure 2. **CNN architecture with the NetVLAD layer.** The layer can be implemented using standard CNN layers (convolutions, softmax, L2-normalization) and one easy-to-implement aggregation layer to perform aggregation in equation (4) ("VLAD core"), joined up in a directed acyclic graph. Parameters are shown in brackets.

R. Arandjelovic, et al. "NetVLAD: CNN architecture for weakly supervised place recognition." CVPR. 2016.

Object / scene recognition

- Bag-of-words features can be used as input into a classifier:



Object / scene recognition

- Bag-of-words features can be used as input into a classifier:

~Car

Car

Object / scene recognition

- Bag-of-words features can be used as input into a classifier:

~Car

Car

Object / scene recognition

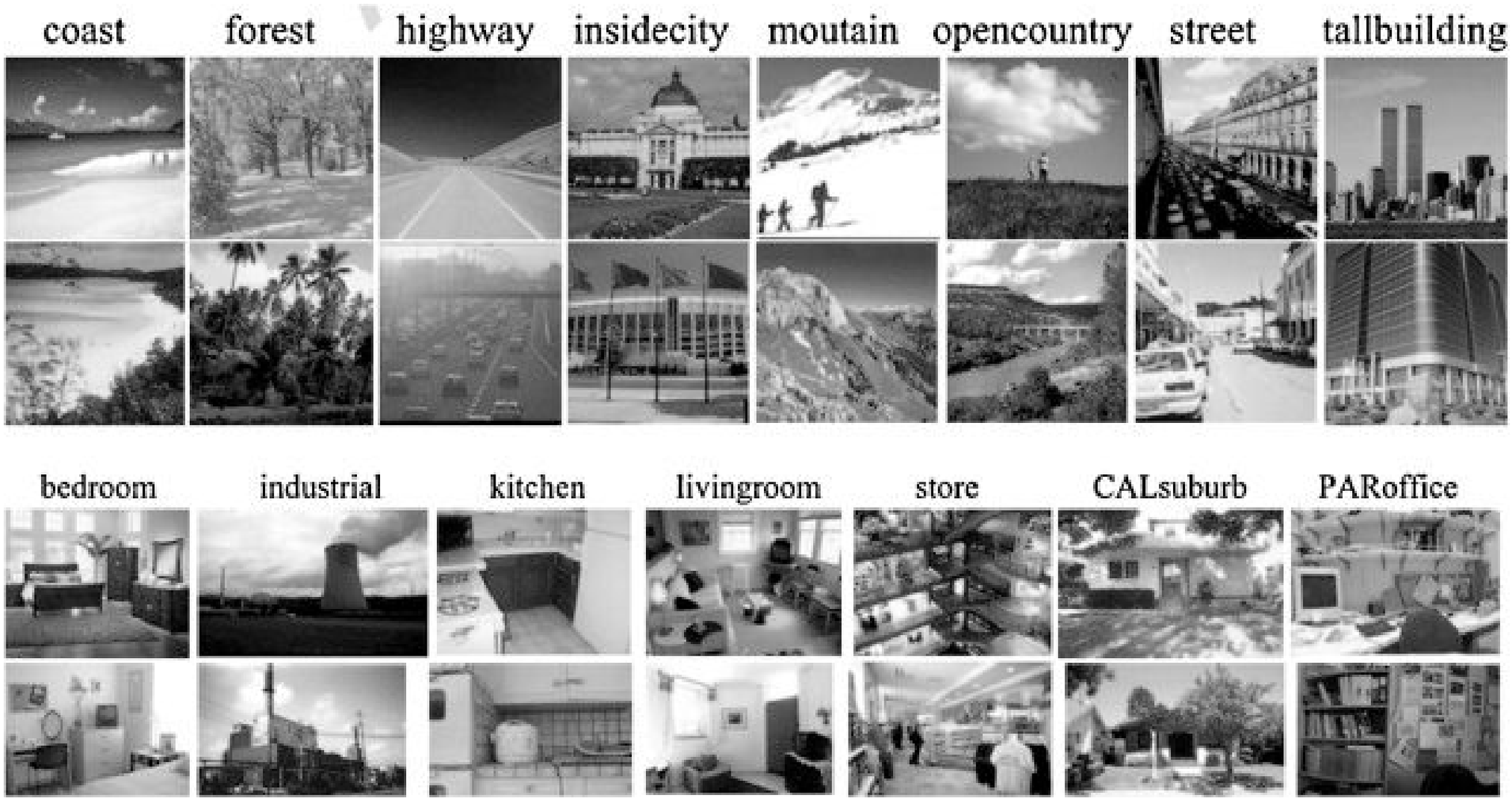
- Bag-of-words features can be used as input into a classifier:

~Car

Car

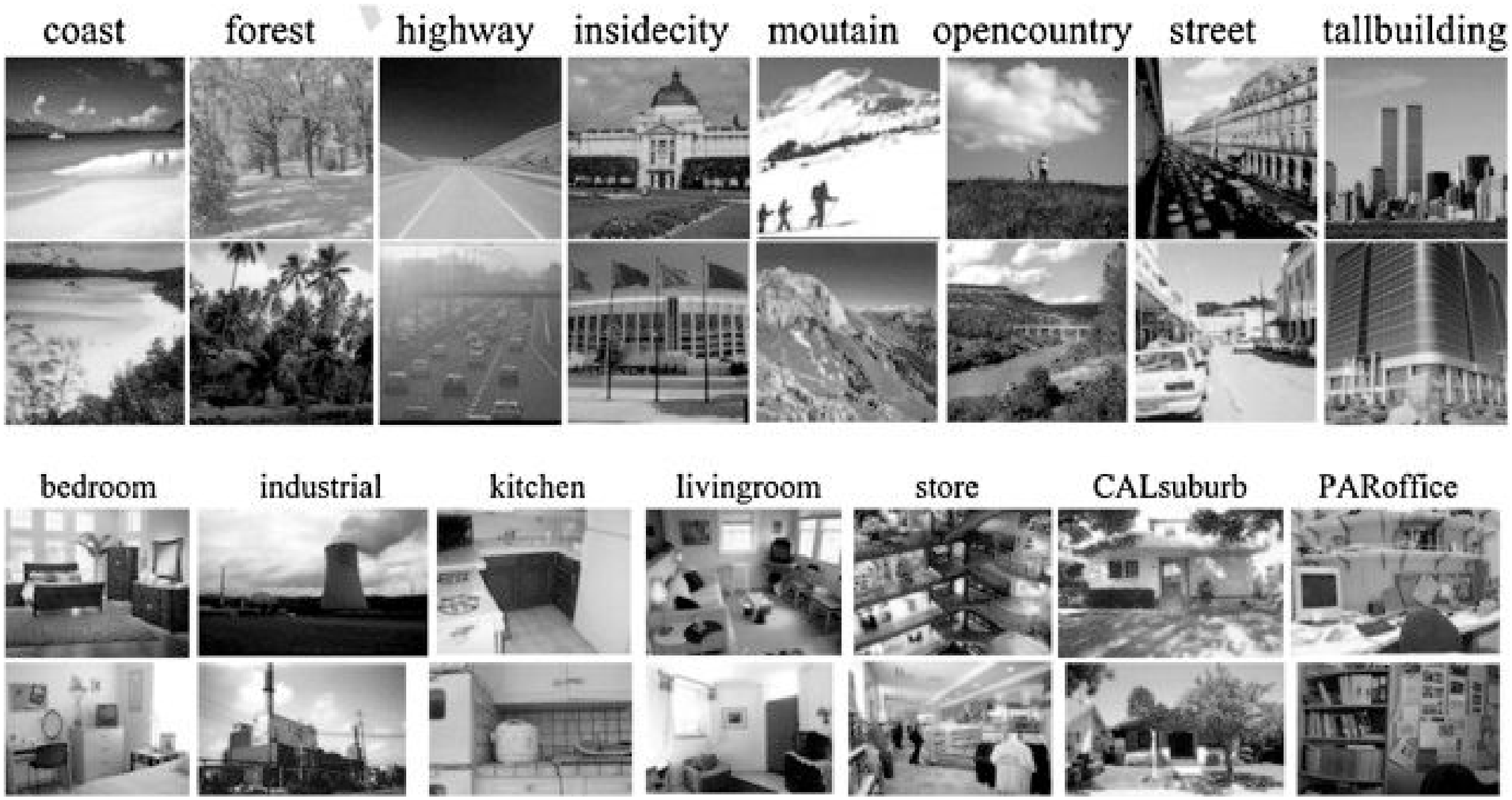
Example: Scene understanding

- Bag-of words representations are particularly good for *scene understanding*:



Example: Scene understanding

- Bag-of words representations are particularly good for *scene understanding*:

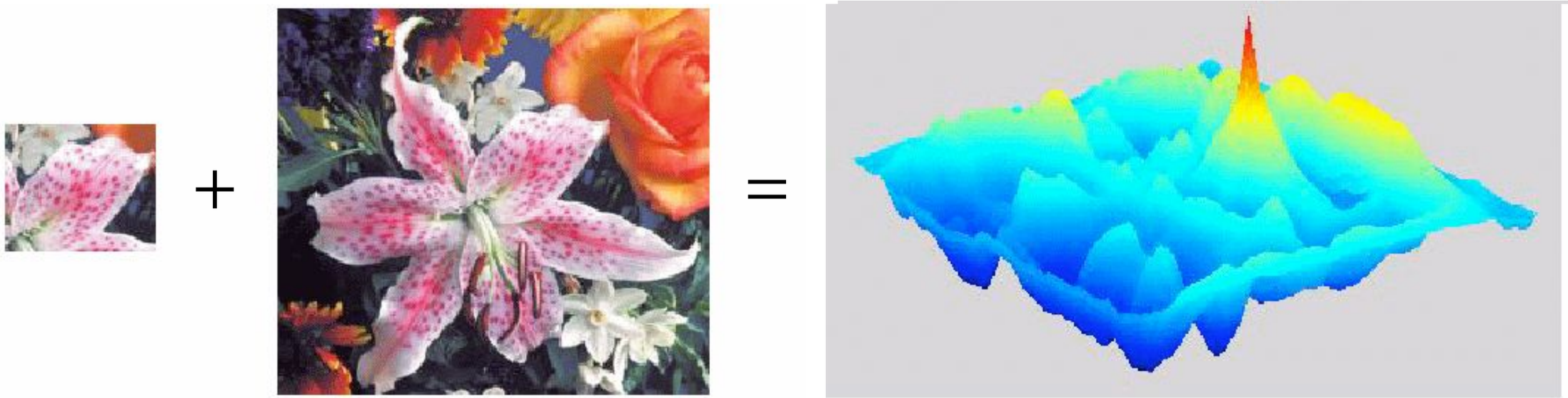


- Scene is not determined by specific objects, but by a *constellation* of features

Object recognition

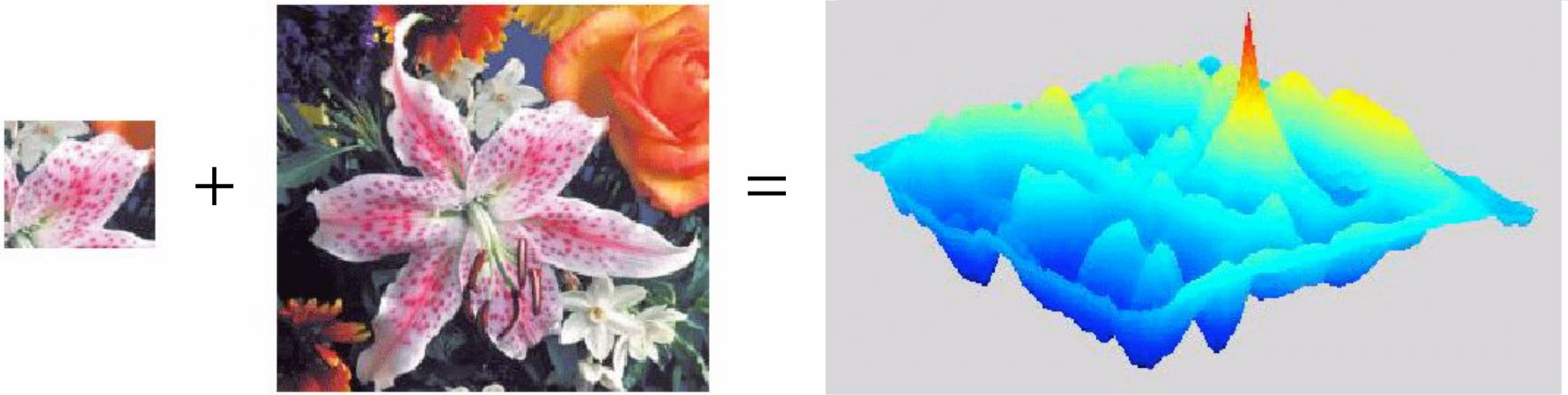
Object recognition

- Simple approach matches a *template* with the image to find its location:
  - E.g., use *linear filter* or *normalized cross-correlation*: $$\frac{1}{N}\sum_{x,y}\frac{(I(x,y)-\bar{I})(T(x,y)-\bar{T})}{\sigma_I\sigma_T}$$



Object recognition

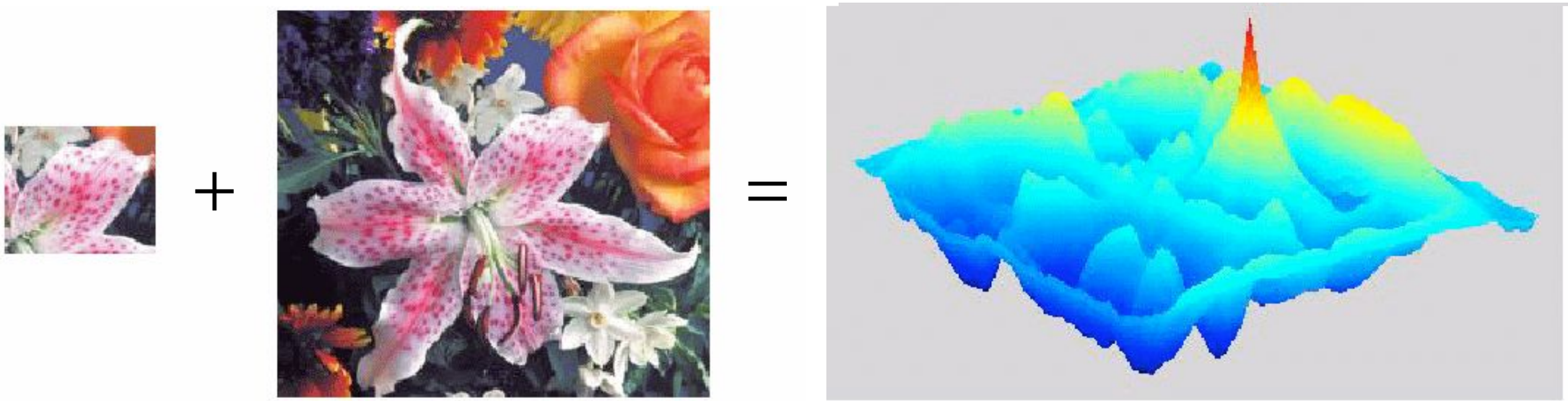
- Simple approach matches a *template* with the image to find its location:
  - E.g., use *linear filter* or *normalized cross-correlation*: $\frac{1}{N}\sum_{x,y}\frac{(I(x,y)-\bar{I})(T(x,y)-\bar{T})}{\sigma_I\sigma_T}$



- How do we obtain the template?

Object recognition

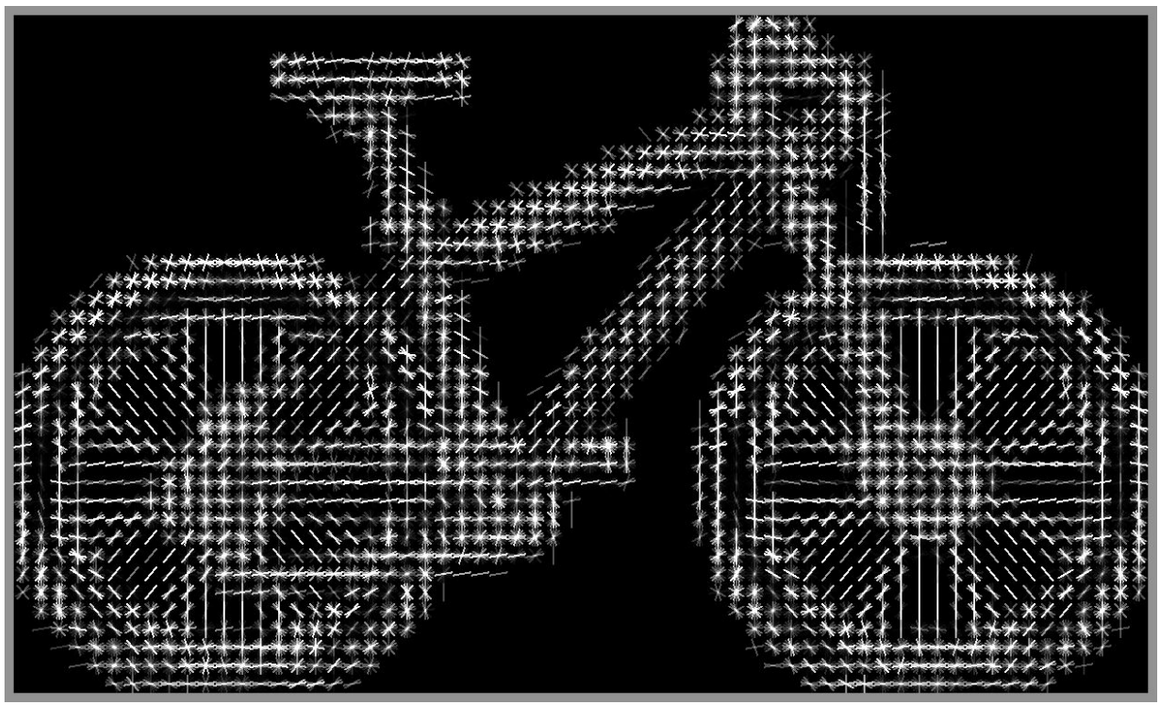
- Simple approach matches a *template* with the image to find its location:
  - E.g., use *linear filter* or *normalized cross-correlation*: $\frac{1}{N}\sum_{x,y}\frac{(I(x,y)-\bar{I})(T(x,y)-\bar{T})}{\sigma_I\sigma_T}$



- How do we obtain the template? Pattern recognition: train a classifier!

Dalal-Triggs detector

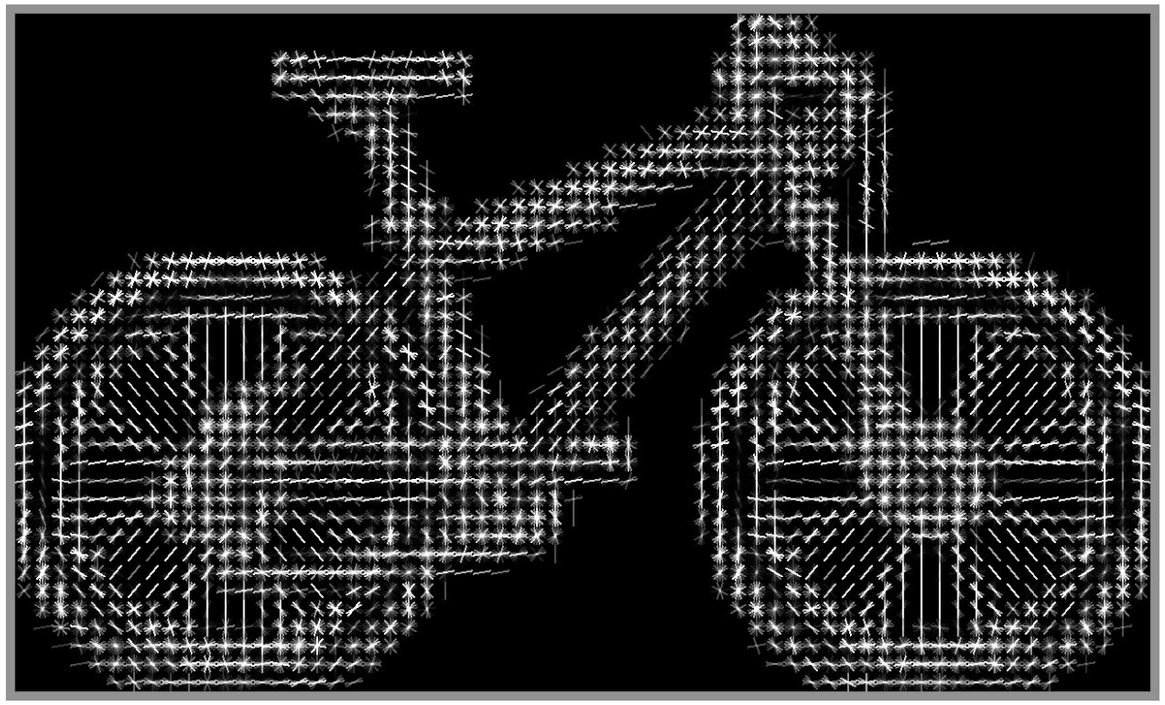
- Extract histogram of gradients (HOG) features from the image patch:

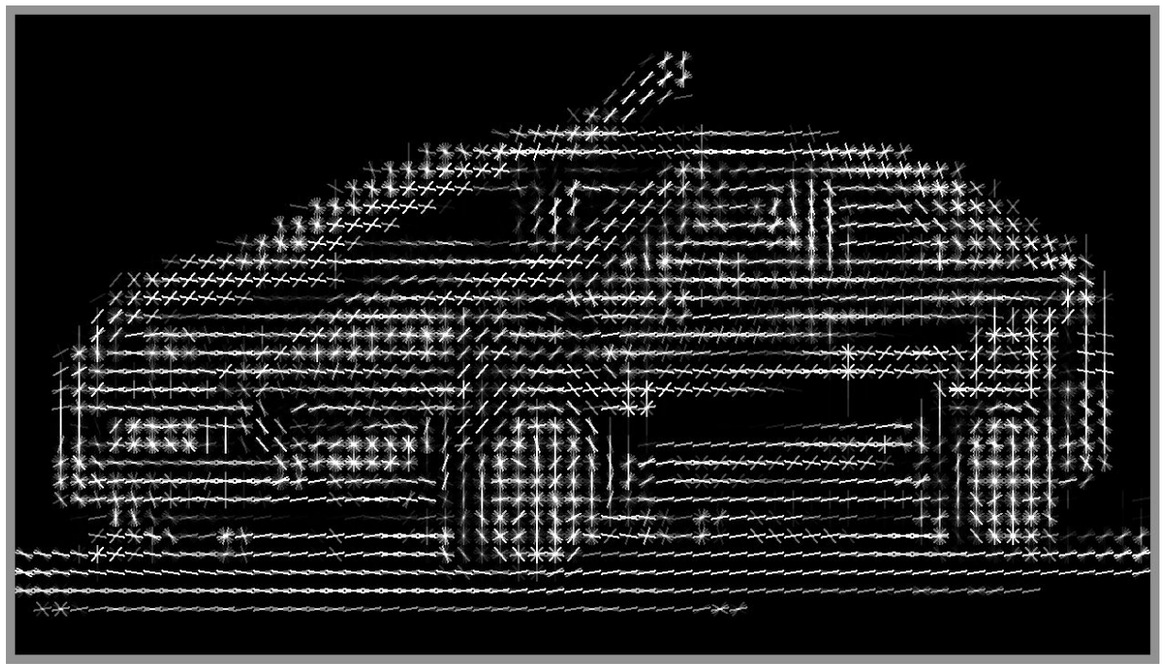


- HOG features divide an image into small (8x8) *blocks*, and measure the *gradient orientations* in each of the blocks using a histogram (almost like SIFT).

Dalal-Triggs detector

- Different objects have different HOG features:



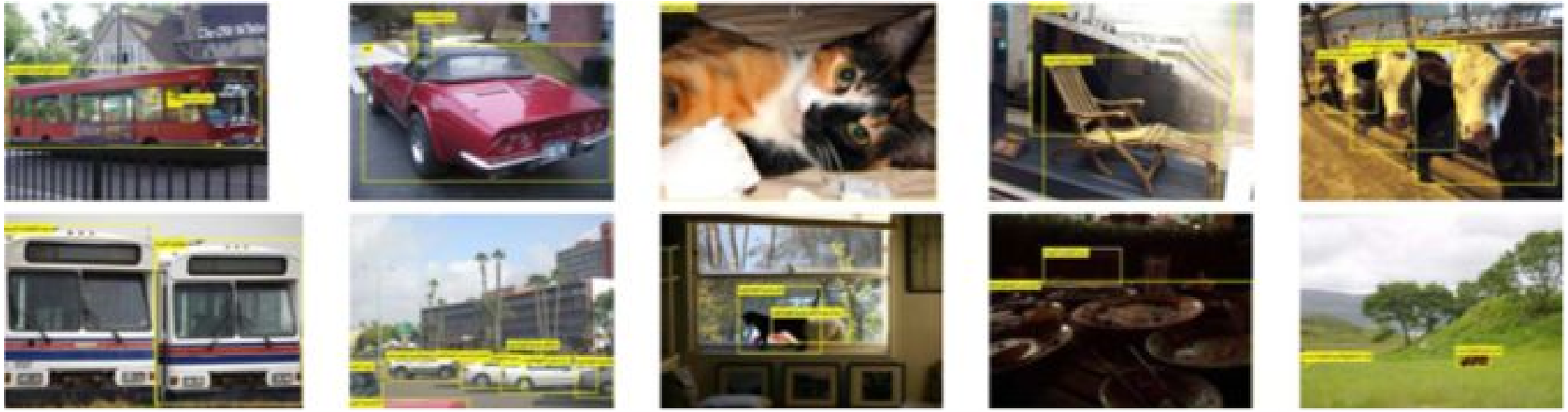


Dalal-Triggs detector

- Train a linear SVM on annotated images to predict object presence:

Training: $$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \max\left(0, 1 - y\mathbf{w}^{\mathrm{T}}\phi(\mathbf{I};\mathbf{x})\right)$$

Detection: $$s(\mathbf{I};\mathbf{x}) = \mathbf{w}^{*\mathrm{T}}\phi(\mathbf{I};\mathbf{x})$$

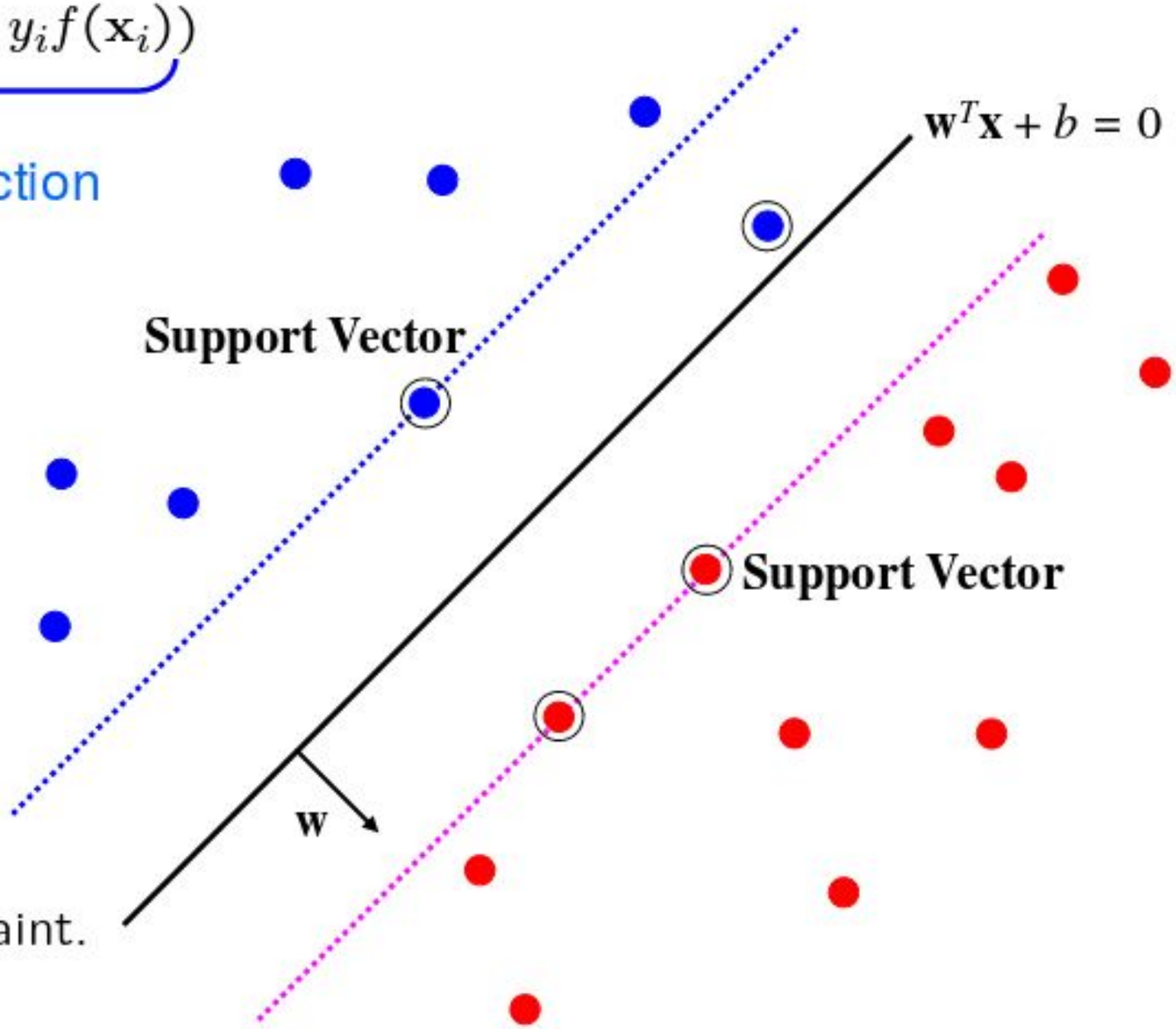


*SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_i^N \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss

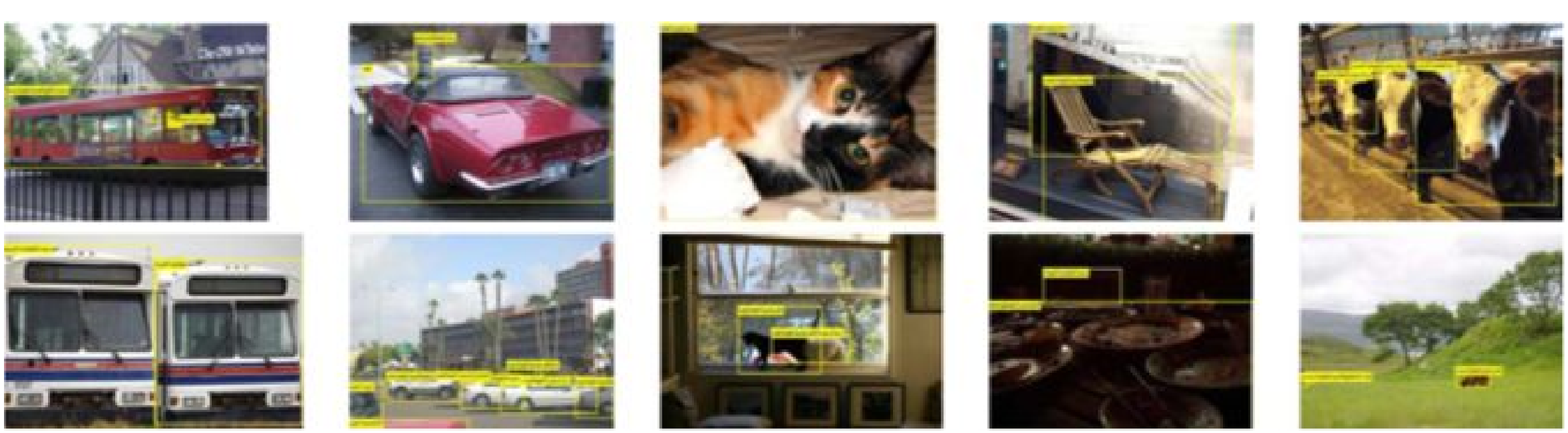


Dalal-Triggs detector

- Train a linear SVM on annotated images to predict object presence:

Training: $$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \max\left(0, 1 - y\mathbf{w}^{\mathrm{T}}\phi(\mathbf{I};\mathbf{x})\right)$$

Detection: $$s(\mathbf{I};\mathbf{x}) = \mathbf{w}^{*\mathrm{T}}\phi(\mathbf{I};\mathbf{x})$$



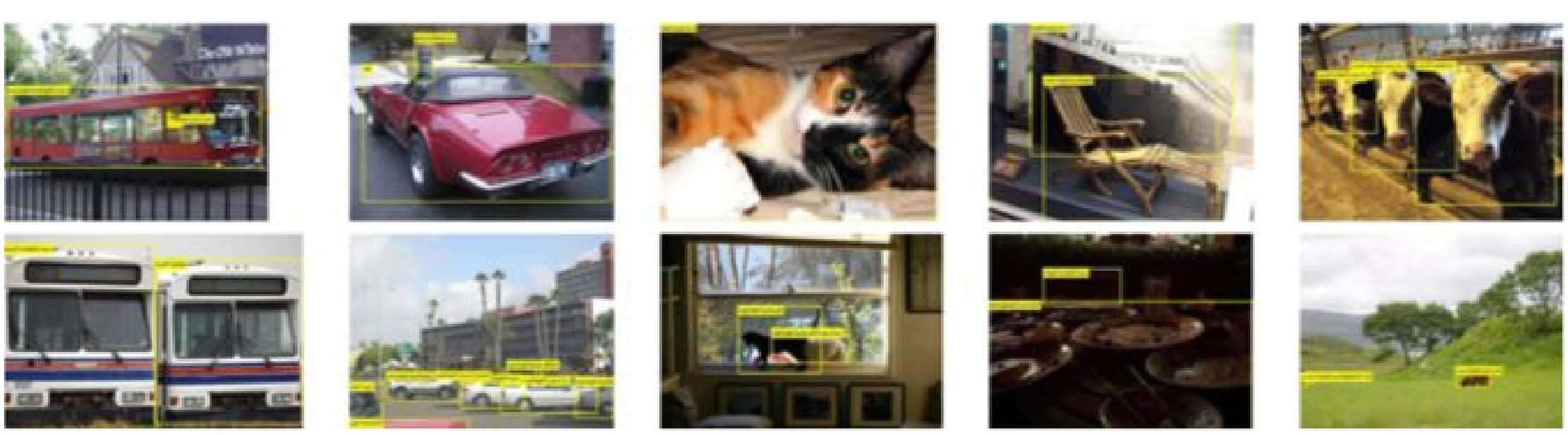
- How do we get the *negative examples* to train the SVM?

Dalal-Triggs detector

- Train a linear SVM on annotated images to predict object presence:

Training: $$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \max\left(0, 1 - y\mathbf{w}^{\mathrm{T}}\phi(\mathbf{I};\mathbf{x})\right)$$

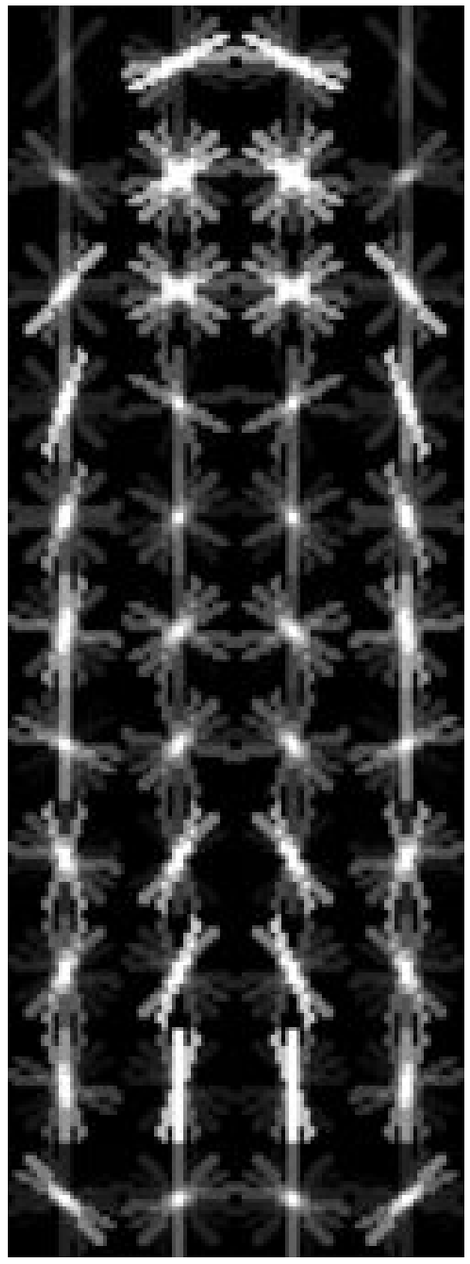
Detection: $$s(\mathbf{I};\mathbf{x}) = \mathbf{w}^{*\mathrm{T}}\phi(\mathbf{I};\mathbf{x})$$



- How do we get the *negative examples* to train the SVM? Random patches!

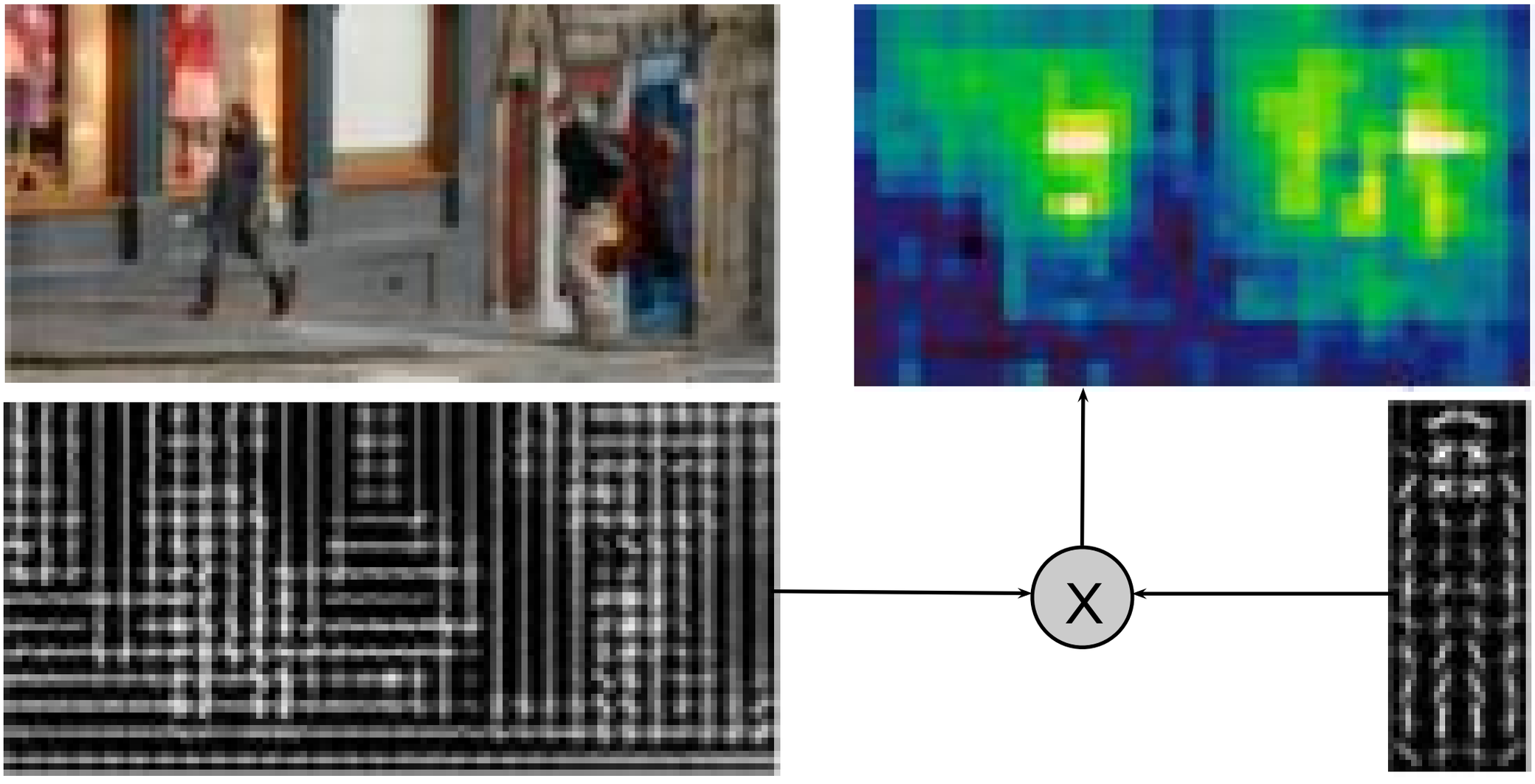
Dalal-Triggs detector

- HOG visualization of the SVM weights for a pedestrian detector:



Dalal-Triggs detector

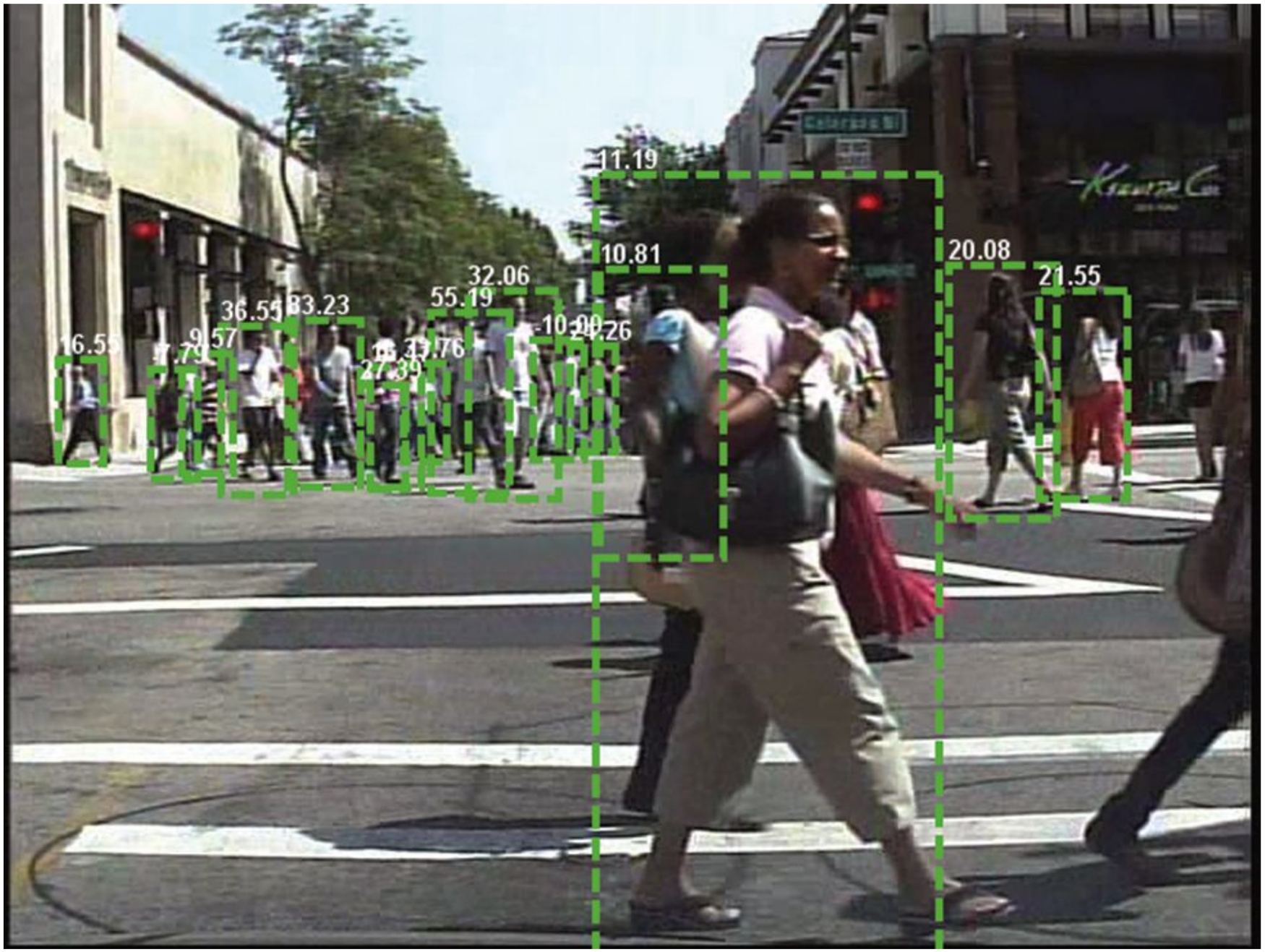
- Applying the detector at each location leads to a *confidence map*:



- *Non-maxima suppression* can be used to obtain the final detections.

Dalal-Triggs detector

- Example of pedestrian detections using Dalal-Triggs detector:



Pictorial structures

- What can we do when a part of the object to be detected is occluded?

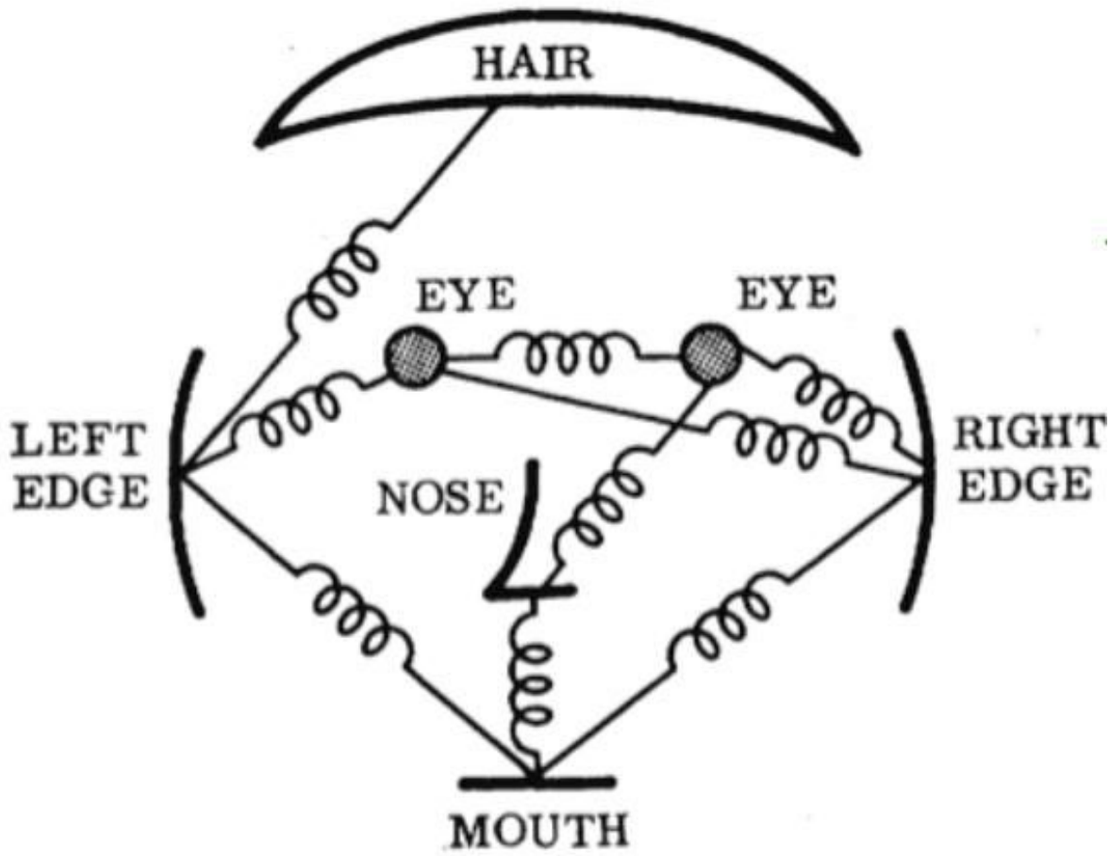
Pictorial structures

- What can we do when a part of the object to be detected is occluded?

- Exploit the fact that other parts of the object are still visible!

Pictorial structures

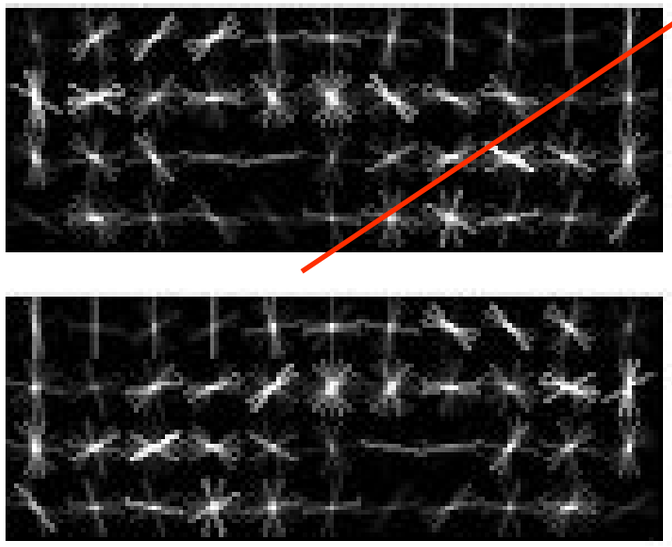
- What can we do when a part of the object to be detected is occluded?
- Exploit the fact that other parts of the object are still visible!
- *Pictorial structures* does this by modeling objects as a constellation of parts:



Deformable template models

- Defines a *score function* that involves *parts* and *part deformations*:

$$s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|}) = \mathbf{w}_0^{\mathrm{T}} \phi(\mathbf{I}; x_0, y_0)$$

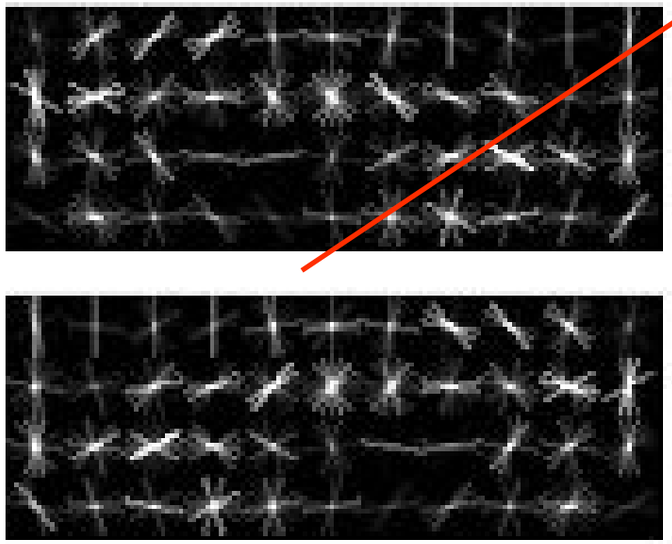


Global object model

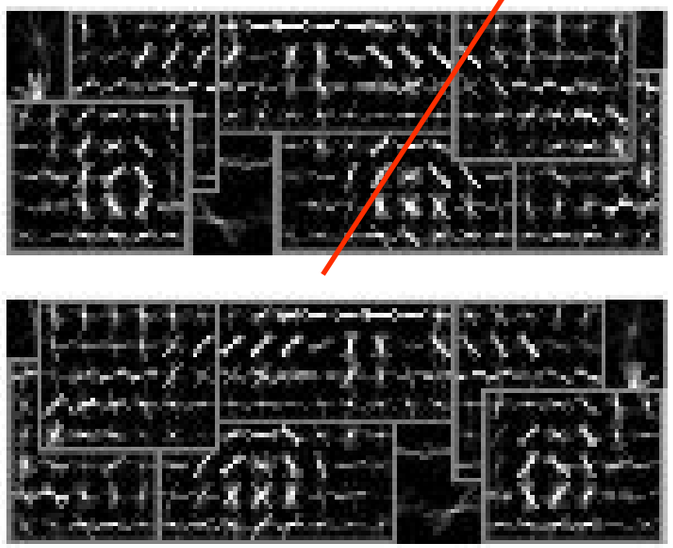
Deformable template models

- Defines a *score function* that involves *parts* and *part deformations*:

$$s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|}) = \mathbf{w}_0^{\mathrm{T}} \phi(\mathbf{I}; x_0, y_0) + \sum_{i \in V} \mathbf{w}_i^{\mathrm{T}} \phi(\mathbf{I}; x_i, y_i)$$



Global object model

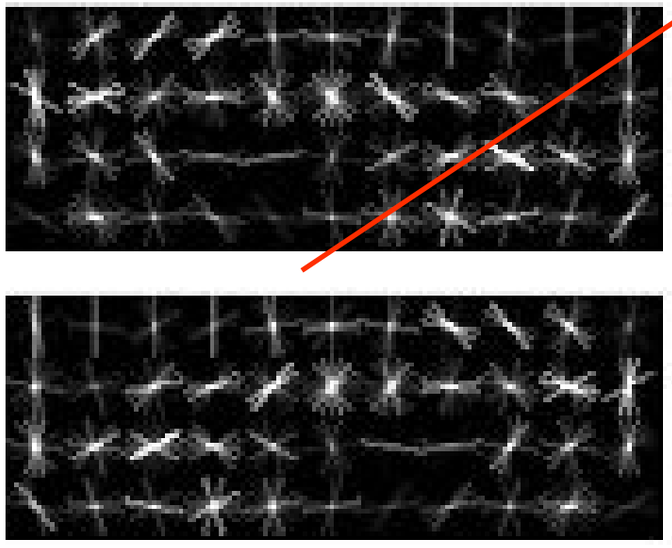


Object part models

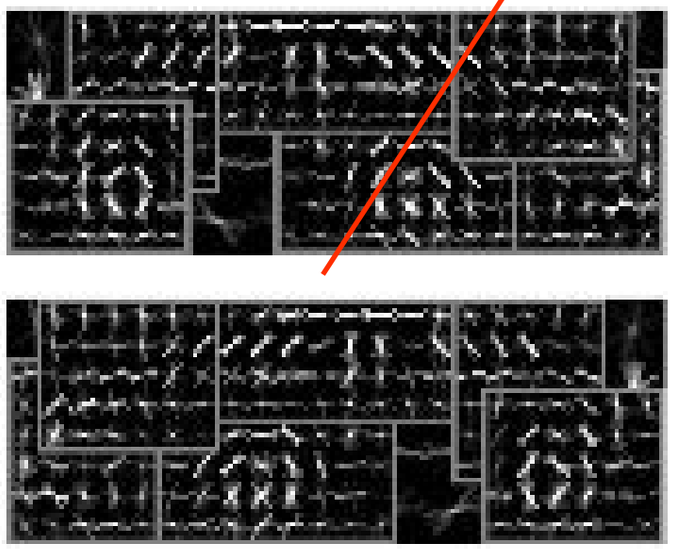
Deformable template models

- Defines a *score function* that involves *parts* and *part deformations*:

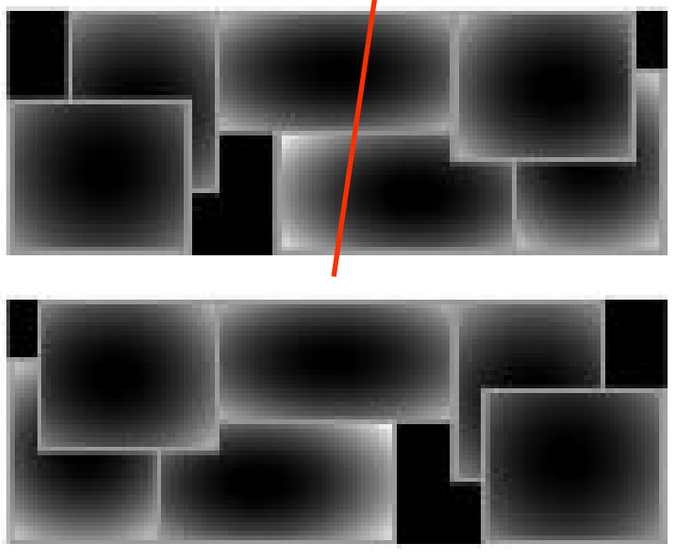
$$s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|}) = \mathbf{w}_0^{\mathrm{T}} \phi(\mathbf{I}; x_0, y_0) + \sum_{i \in V} \mathbf{w}_i^{\mathrm{T}} \phi(\mathbf{I}; x_i, y_i) + \sum_{(i,j) \in E} d_{ij} \phi_d(x_i - x_j, y_i - y_j)$$



Global object model



Object part models

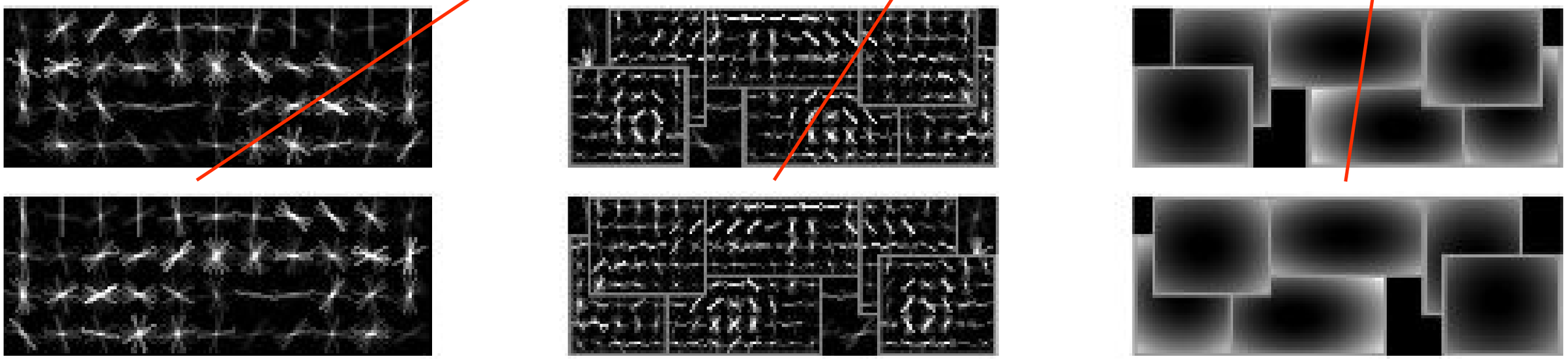


Deformation model

Deformable template models

- Defines a *score function* that involves *parts* and *part deformations*:

$$s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|}) = \mathbf{w}_0^{\mathrm{T}} \phi(\mathbf{I}; x_0, y_0) + \sum_{i \in V} \mathbf{w}_i^{\mathrm{T}} \phi(\mathbf{I}; x_i, y_i) + \sum_{(i,j) \in E} d_{ij} \phi_d(x_i - x_j, y_i - y_j)$$

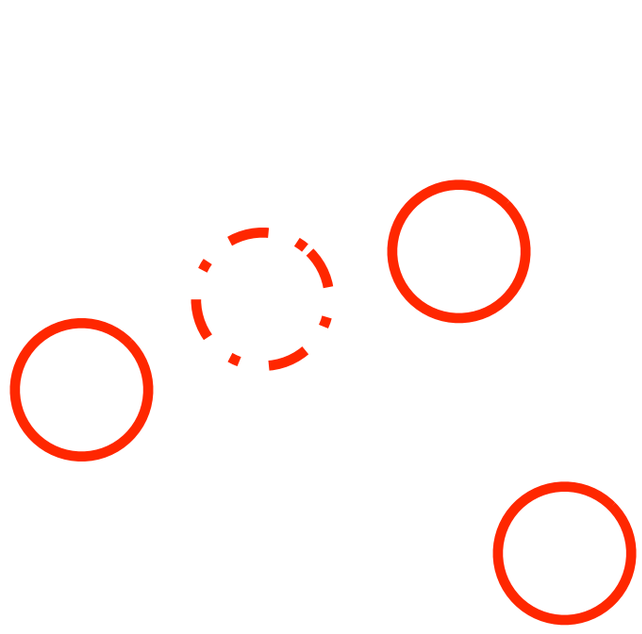


Global object model | Object part models | Deformation model

- Deformable template models are much more robust against *partial occlusions* and *deformations* of non-rigid objects

Graph structure

- The graph structure for this model is a star-shaped tree:



Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

- For *squared-error* deformation models, this can be done very efficiently:

$$g(x_i) = \min_{x_j} (f(x_j) + (x_i - x_j)^2)$$

Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

- For *squared-error* deformation models, this can be done very efficiently:

$$g(x_i) = \min_{x_j} (f(x_j) + (x_i - x_j)^2)$$

final score with deformations

Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

- For *squared-error* deformation models, this can be done very efficiently:

$$g(x_i) = \min_{x_j}(f(x_j) + (x_i - x_j)^2)$$

final score with deformations

negative part model score

Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

- For *squared-error* deformation models, this can be done very efficiently:

$$g(x_i) = \min_{x_j} (f(x_j) + (x_i - x_j)^2)$$

final score with deformations

negative part model score

deformation penalty

Pictorial structures

- Find configuration of pict. structures model by maximizing over part locations:

$$\max_{x_0, y_0, \ldots, x_{|V|}, y_{|V|}} s(\mathbf{I}; x_0, y_0, \ldots, x_{|V|}, y_{|V|})$$

- For *squared-error* deformation models, this can be done very efficiently:

$$g(x_i) = \min_{x_j} (f(x_j) + (x_i - x_j)^2)$$

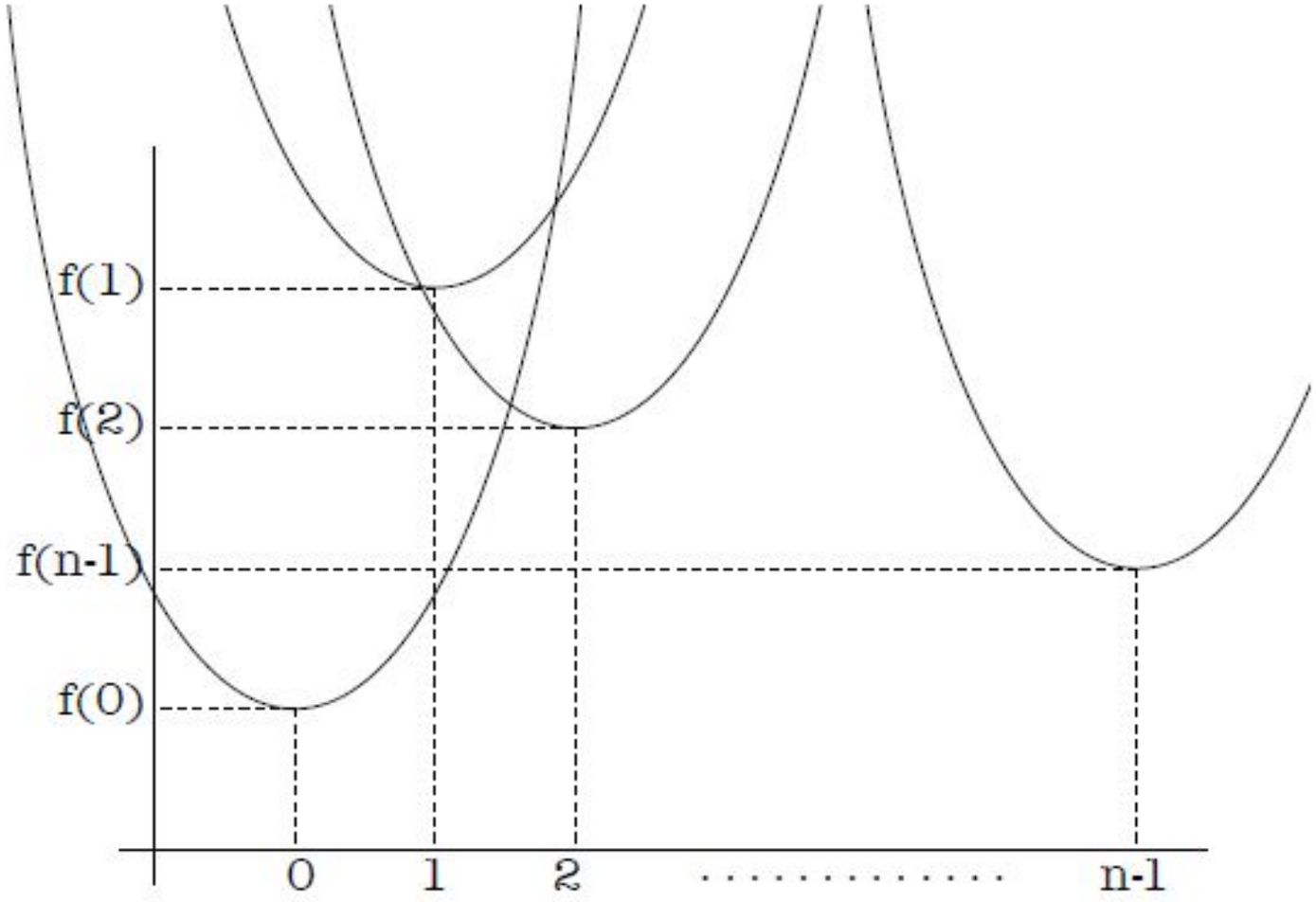
final score with deformations

negative part model score

deformation penalty

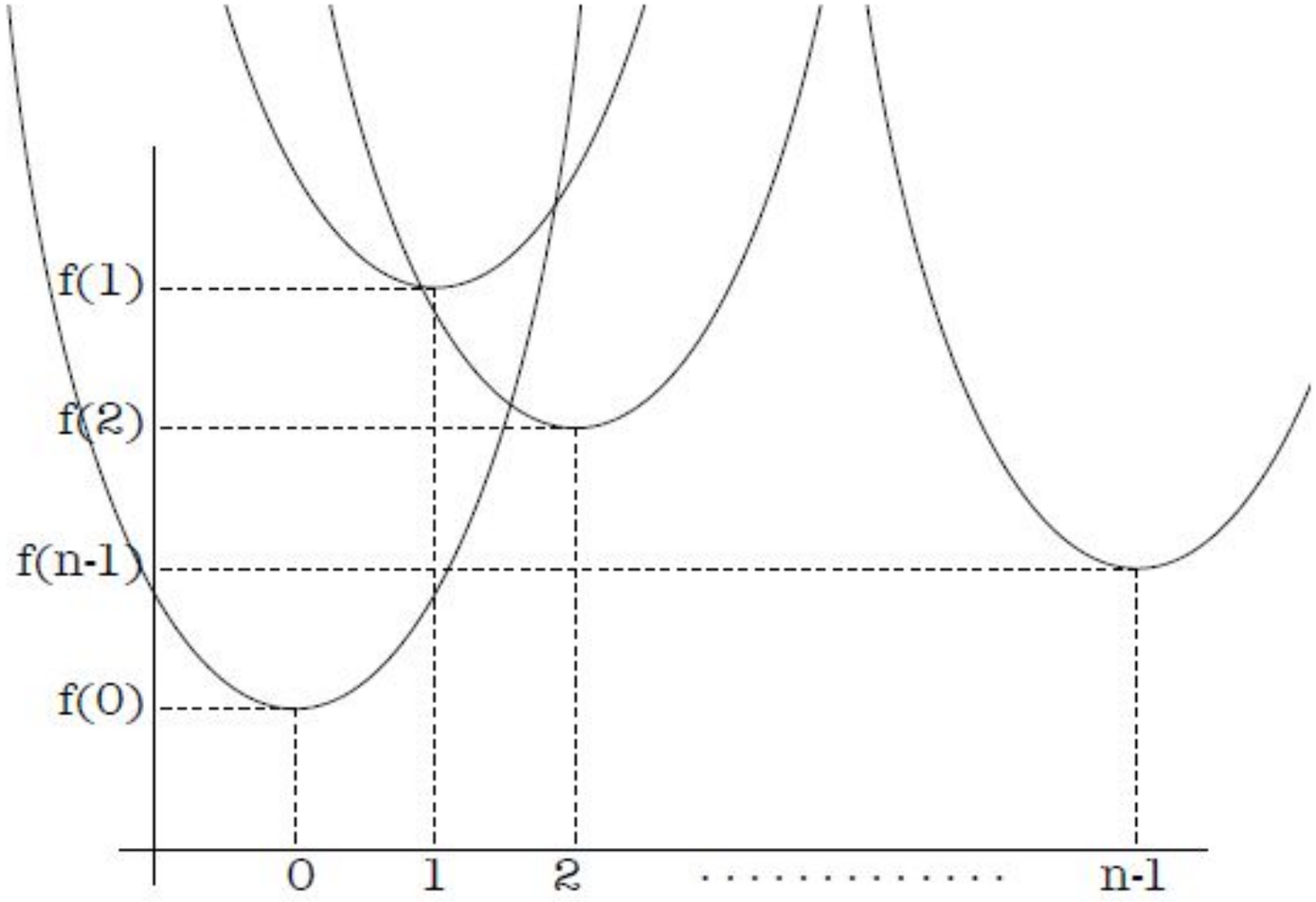
- Hence, we have a parabola for every pixel x_j rooted at $(x_j, f(x_j))$

Pictorial structures



* Felzenszwalb & Huttenlocher, 2004

Pictorial structures



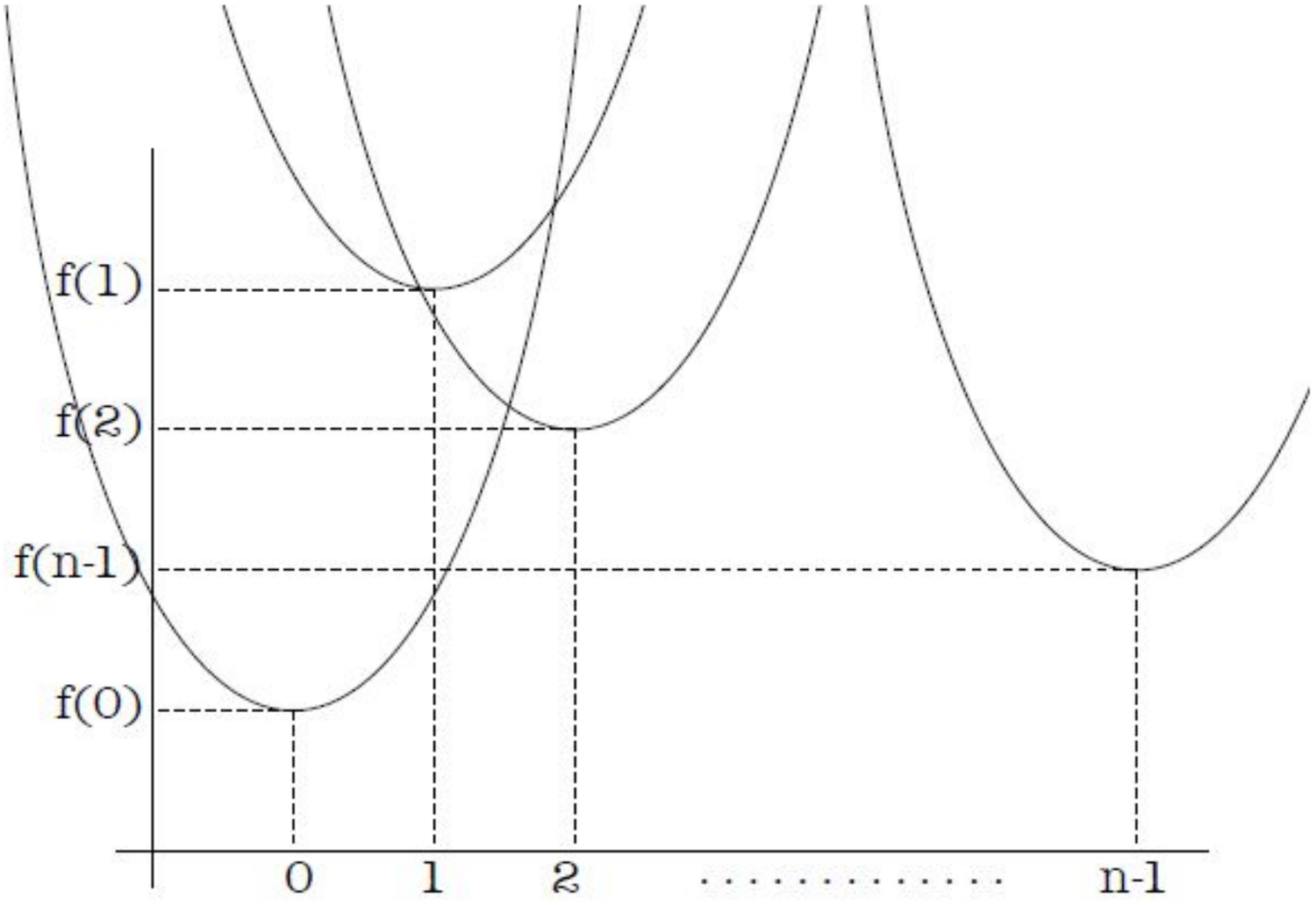
- It is straightforward to compute the *intersection* between two parabolas:

$$\frac{(f(x_i) + x_i^2) - (f(x_j) + x_j^2)}{2x_i - 2x_j}$$

* Felzenszwalb & Huttenlocher, 2004

Pictorial structures

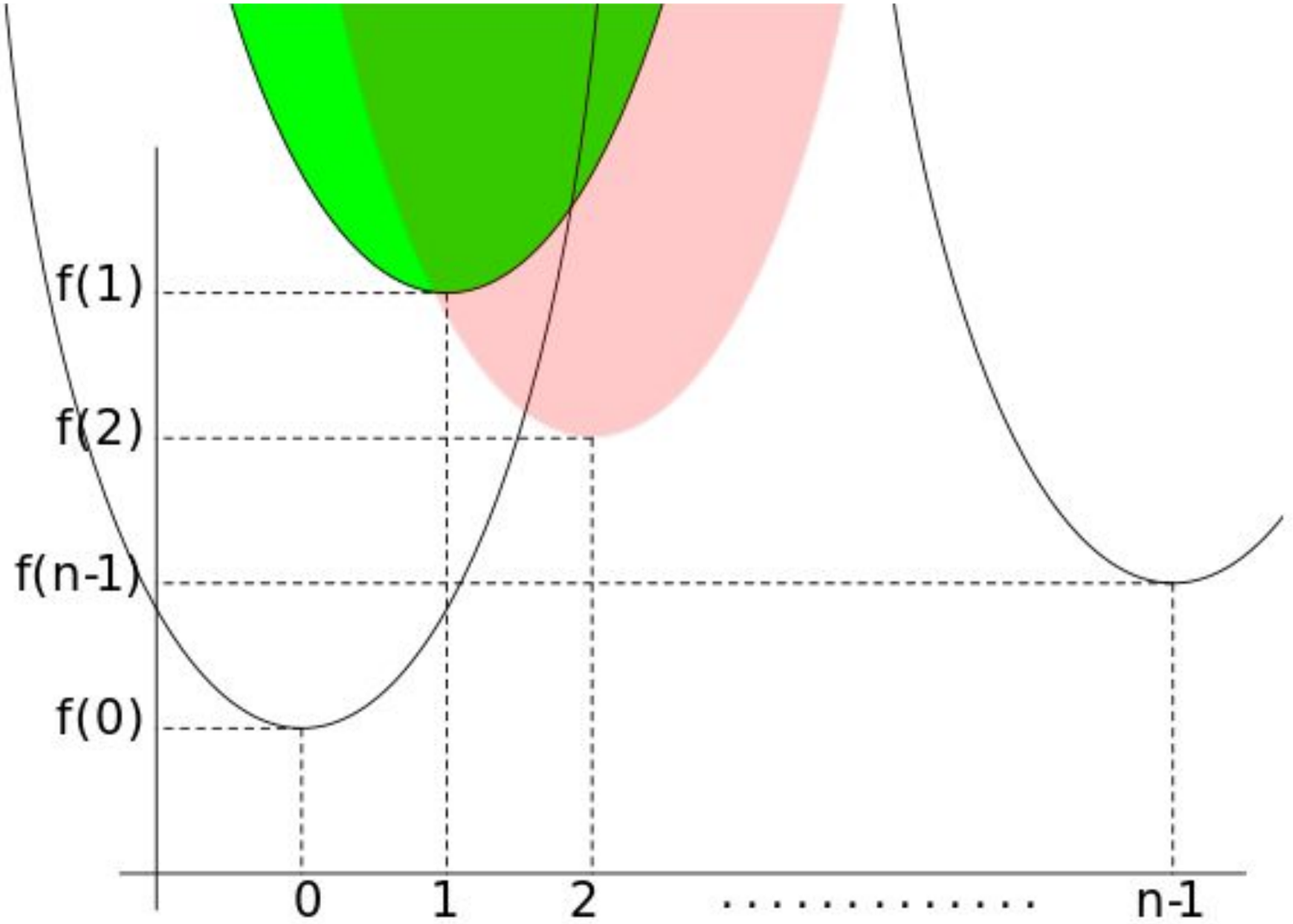
- If $x_j < x_i$: parabola corresponding to x_j is *below* that of x_i *left* of the intersection, and above it *right* of the intersection



* Felzenszwalb & Huttenlocher, 2004

Pictorial structures

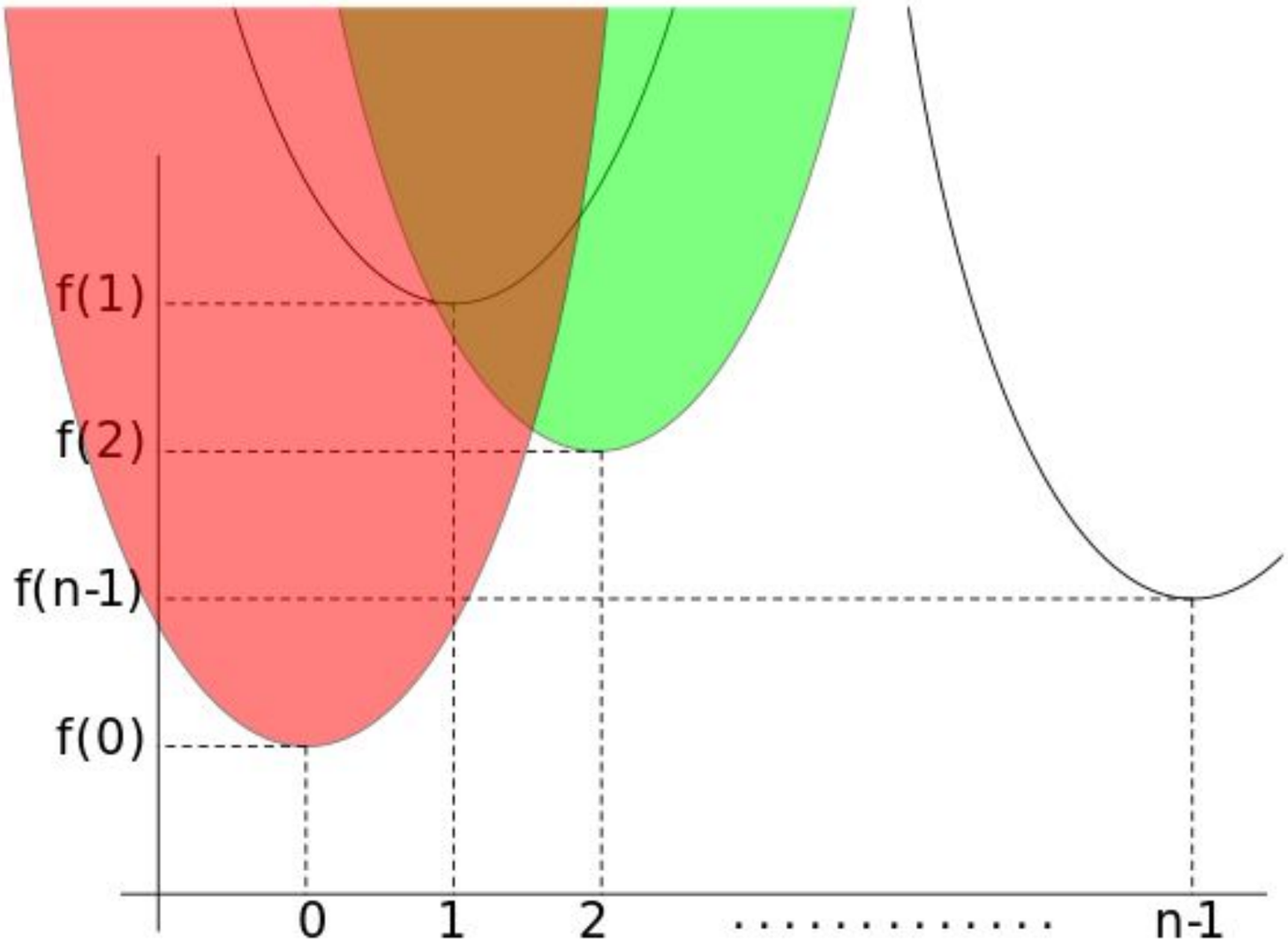
- If $x_j < x_i$: parabola corresponding to x_j is *below* that of x_i *left* of the intersection, and above it *right* of the intersection



* Felzenszwalb & Huttenlocher, 2004

Pictorial structures

- If $x_j < x_i$: parabola corresponding to x_j is *below* that of x_i *left* of the intersection, and above it *right* of the intersection

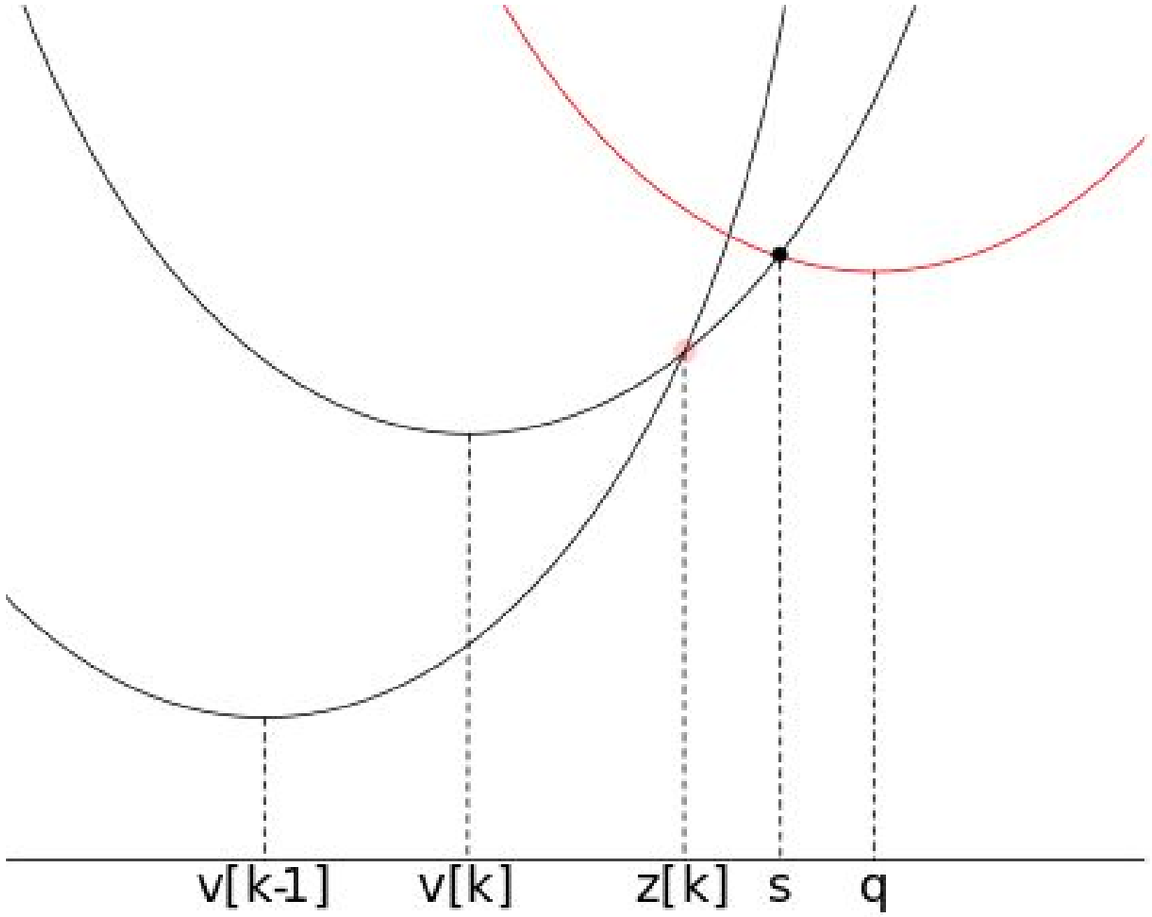


* Felzenszwalb & Huttenlocher, 2004

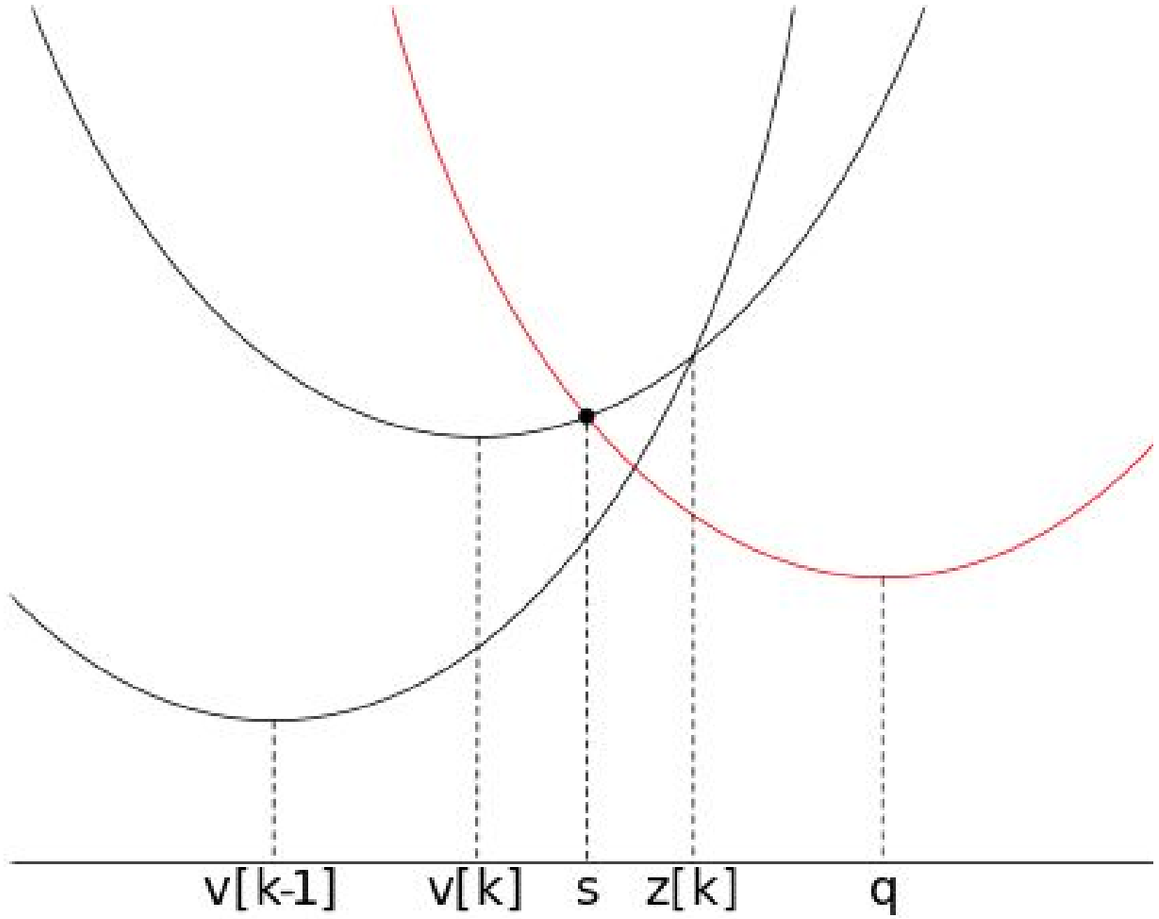
Pictorial structures

- Maintain the *lower envelope* of the parabolas (parabolas and intersections)

- When adding a new parabola, there are two possibilities:



new intersection right of last intersection:
maintain last parabola in the envelope

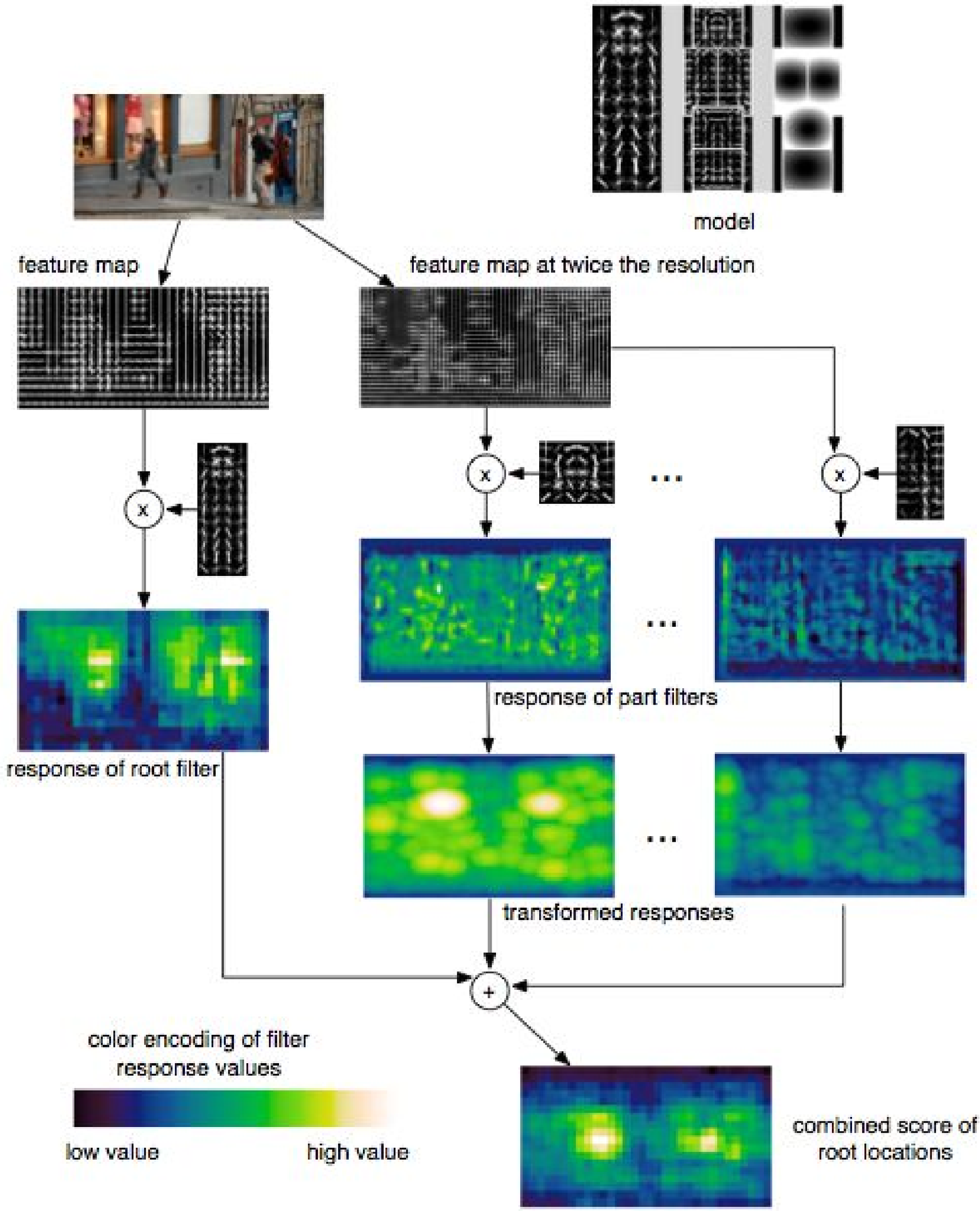


new intersection left of last intersection:
remove last parabola from the envelope

Pictorial structures

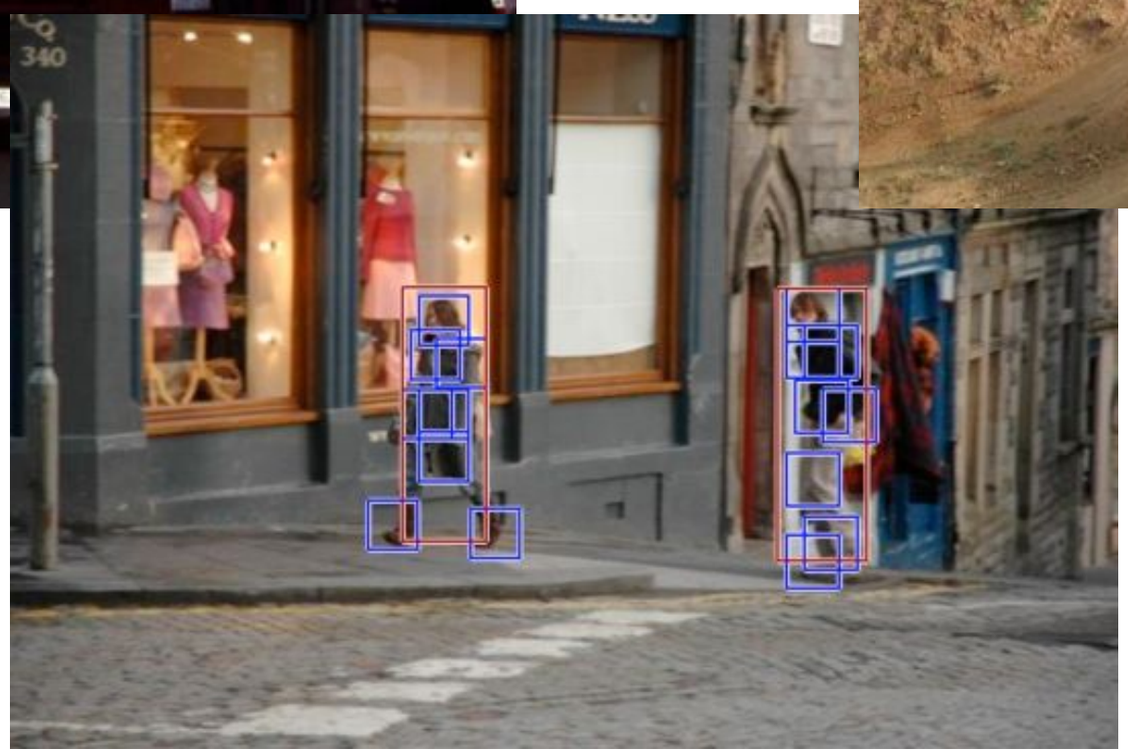
- This suggests a simple algorithm that is *linear* in the number of pixels:
    - Maintain list with the *lower envelope* of the parabolas (indices and intersections)
    - Move from *left to right* through all parabolas; and do for each parabola:
        - Find intersection of parabola with the last parabola in lower envelope
        - If intersection is left of last intersection in lower envelope: remove last parabola from lower envelope, and go back one step
        - Add parabola to lower envelope, starting from intersection

* Felzenszwalb & Huttenlocher, 2004

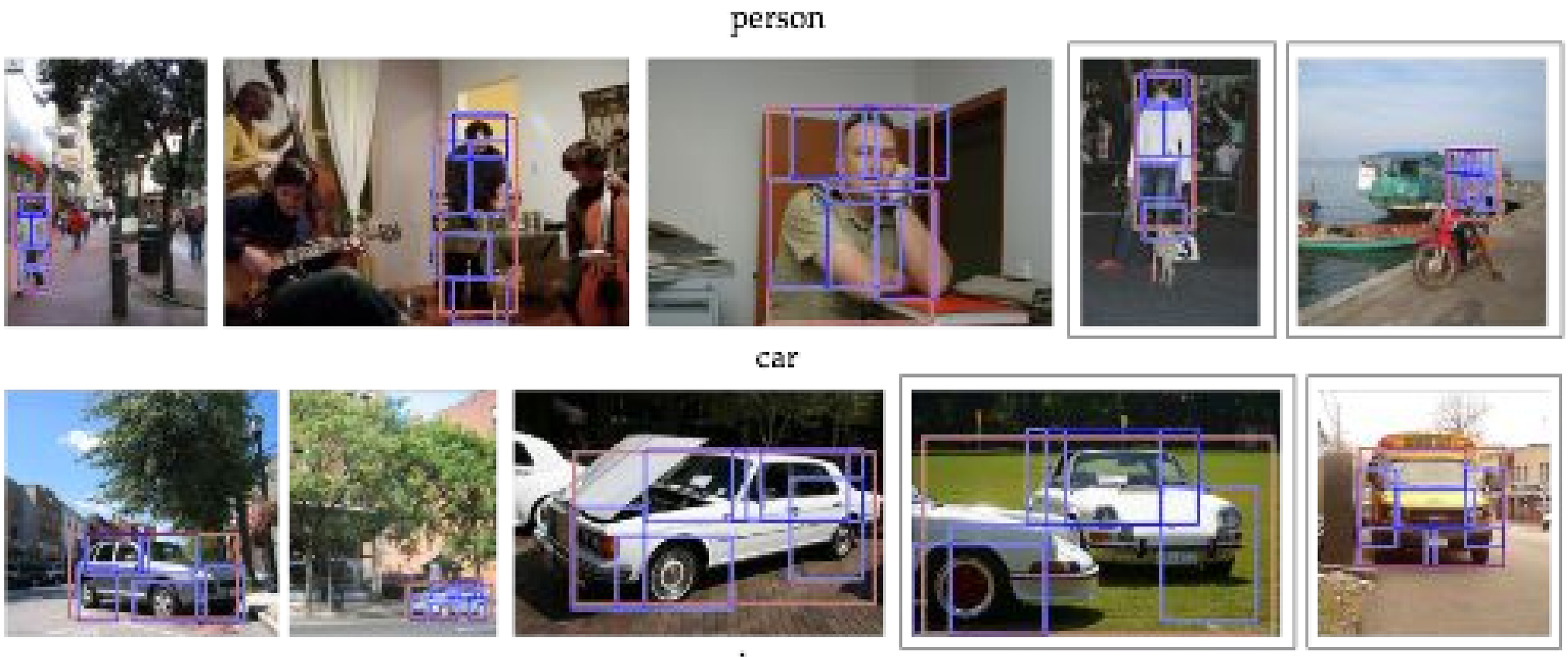

model
feature map
feature map at twice the resolution
response of root filter
response of part filters
transformed responses
color encoding of filter
response values
low value
high value
combined score of
root locations

Deformable template models

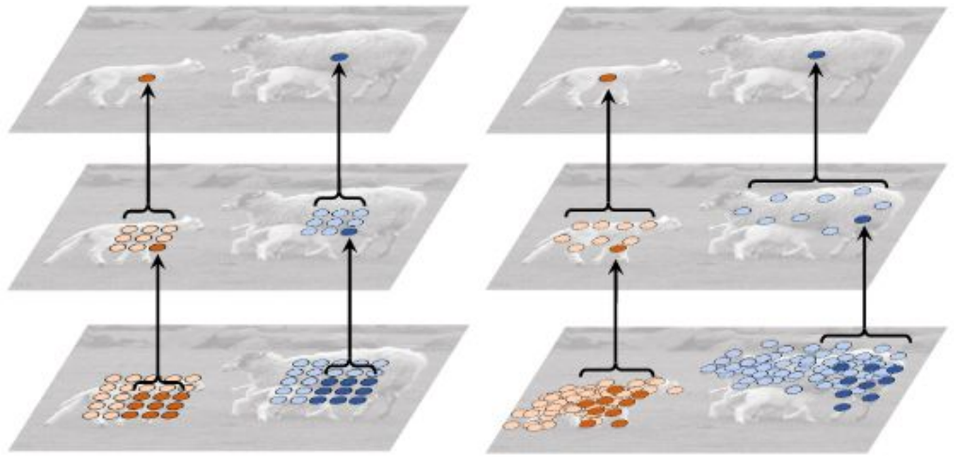
- Examples of object detections by deformable template models:



Example detections

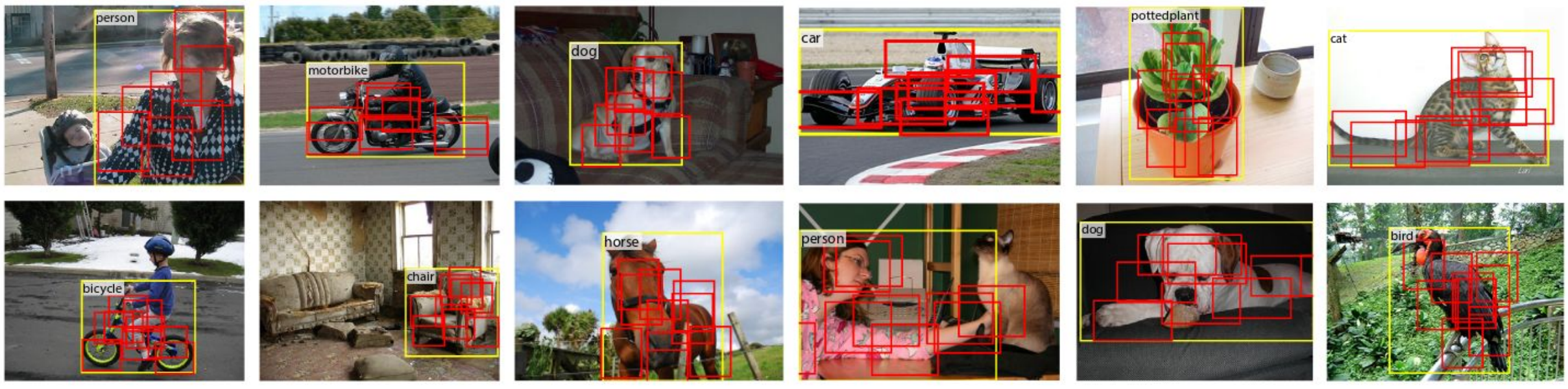


*New deformable convolutions/pooling CNNs



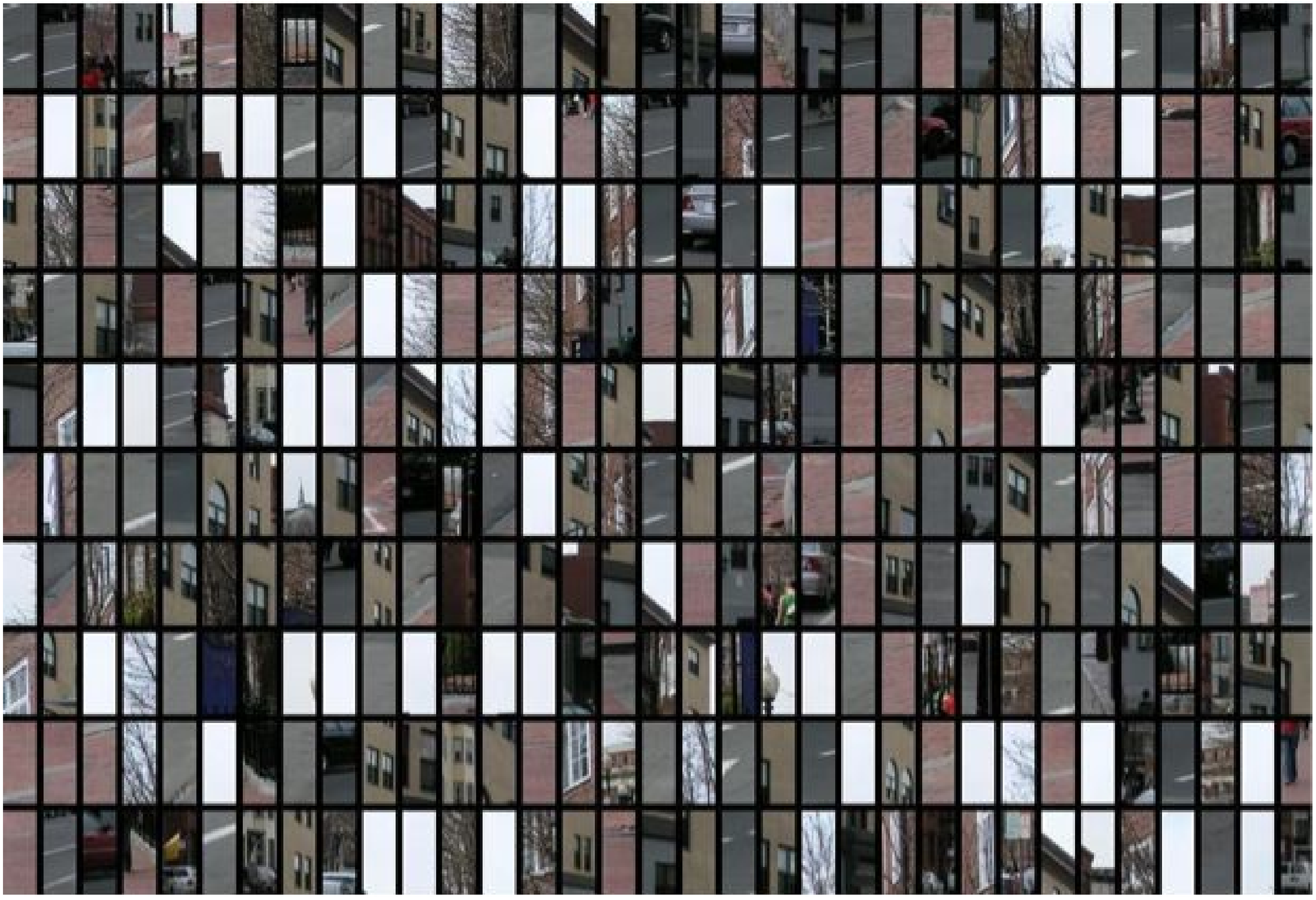
(a) standard convolution (b) deformable convolution

Dai, Jifeng, et al. "Deformable Convolutional Networks." CoRR 2017.



The miserable life of a person detector...

The miserable life of a person detector...

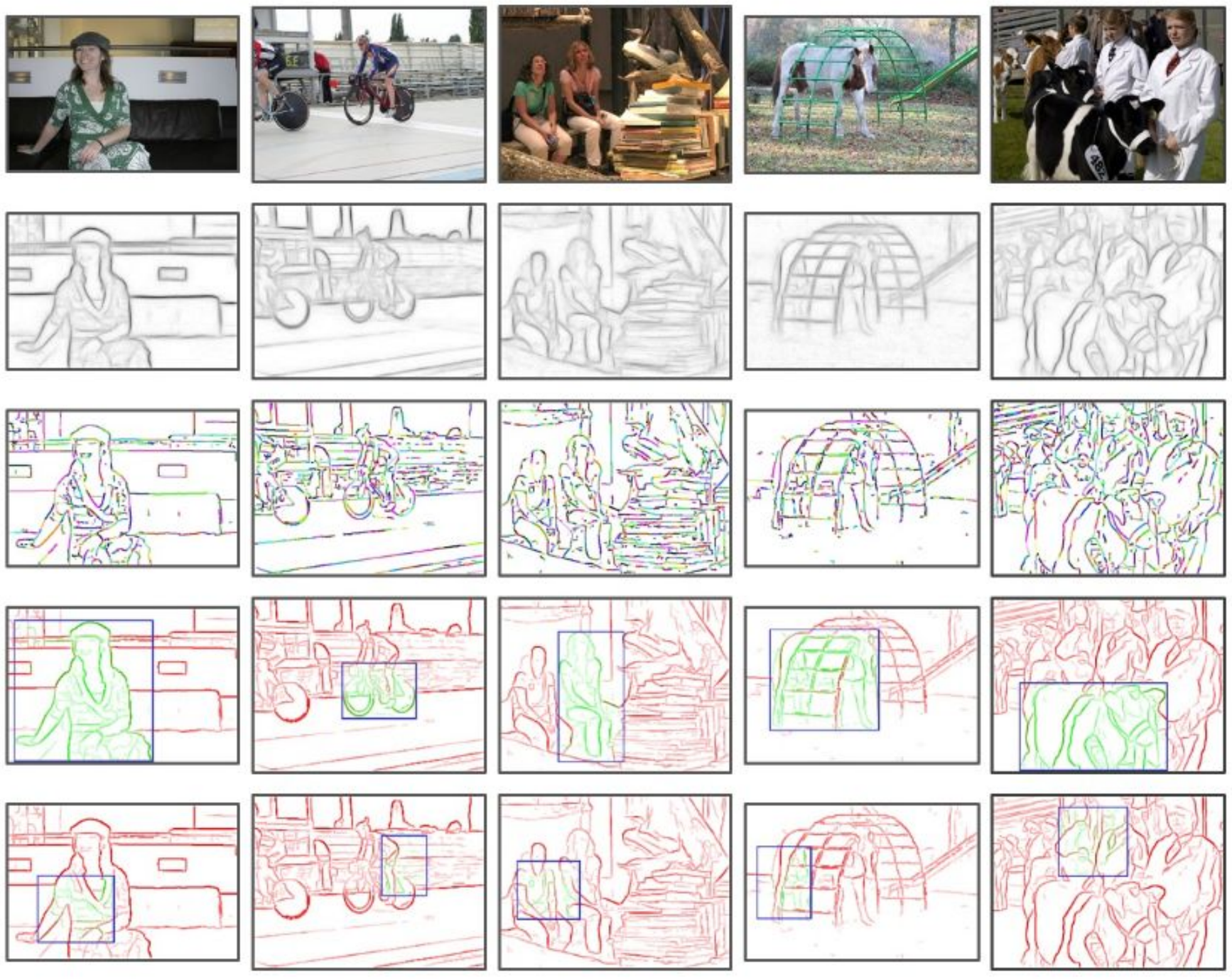


Object proposals to rescue

Fig. 2 Two examples of our selective search showing the necessity of different scales. On the *left* we find many objects at different scales. On the *right* we necessarily find the objects at different scales as the girl is contained by the tv

JRR Uijlings, et al. "Selective search for object recognition." IJCV 2013.

Object proposals to rescue



C. L. Zitnick, and P. Dollár. "Edge boxes: Locating object proposals from edges." ECCV, 2014.

Object proposals to rescue

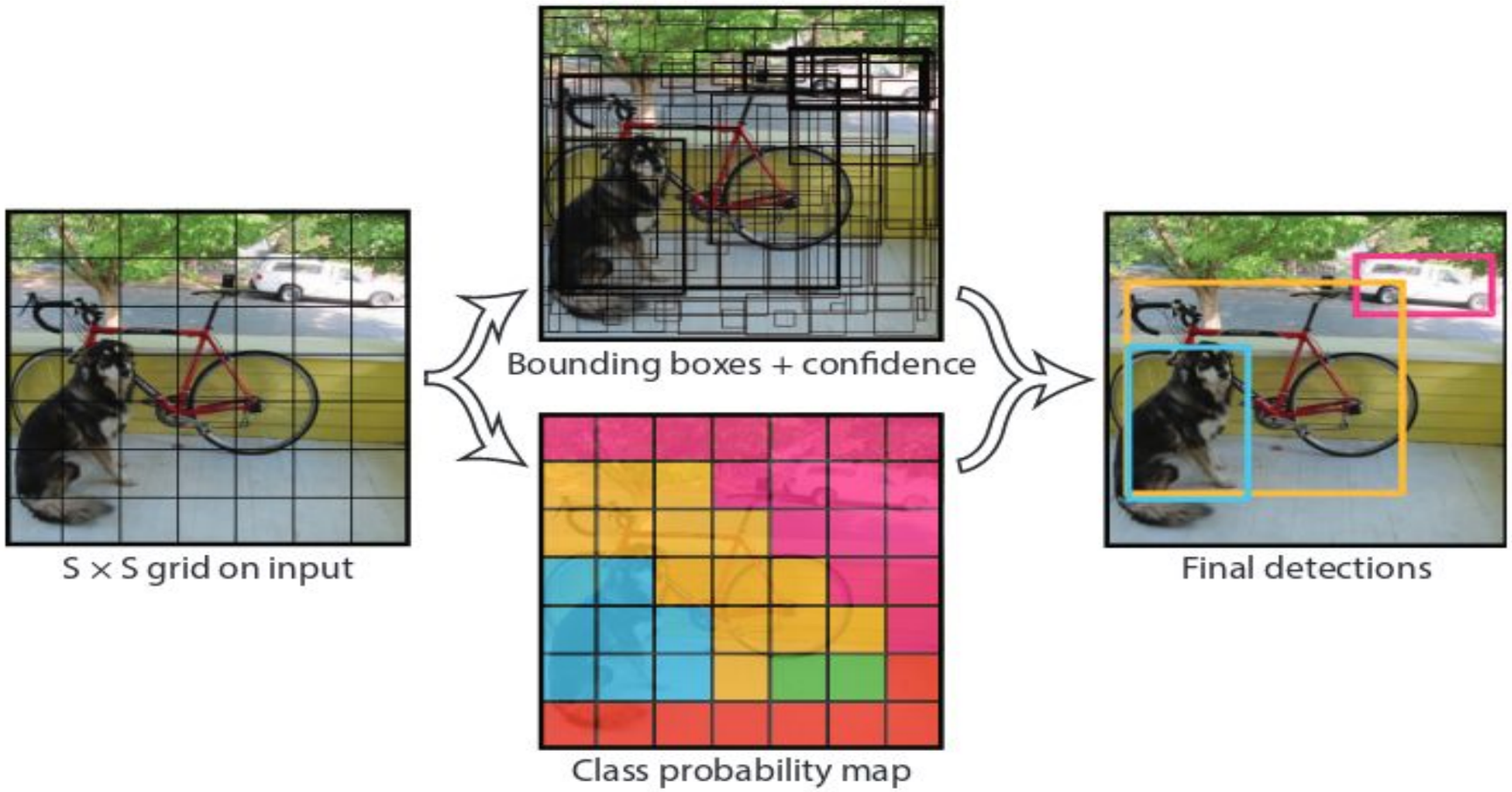
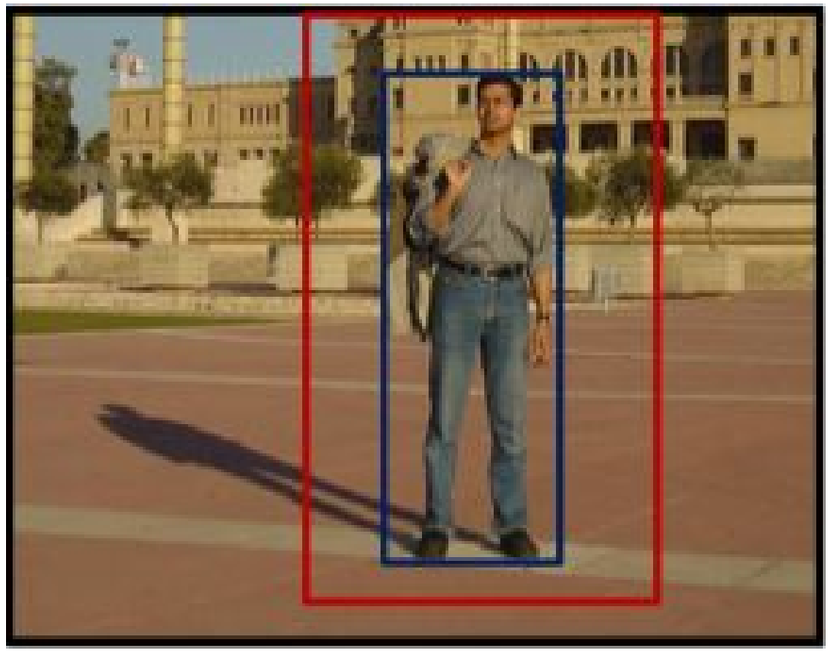


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

J Redmon, et al. "You only look once: Unified, real-time object detection." CVPR. 2016.

Incorporating context

- Incorporating context can help to partially solve recognition problems:
  - Local pixel context (larger bounding box)
  - Semantic context (scene category, other objects present)
  - Geographic context (GPS location, landmarks)
  - Temporal context (objects do not change rapidly)
  - Etcetera...



President George W. Bush makes a statement in the Rose Garden while Secretary of **Defense Donald Rumsfeld** looks on, July 23, 2003. Rumsfeld said the United States would release graphic photographs of the dead sons of **Saddam Hussein** to prove they were killed by American troops. Photo by Larry Downing/Reuters

Reading material: Section 14 and Belongie paper

Optional: Object detection with discriminatively trained part-based models