

INTRODUCTION TO LOCAL FEATURES AND VIDEO ANALYSIS

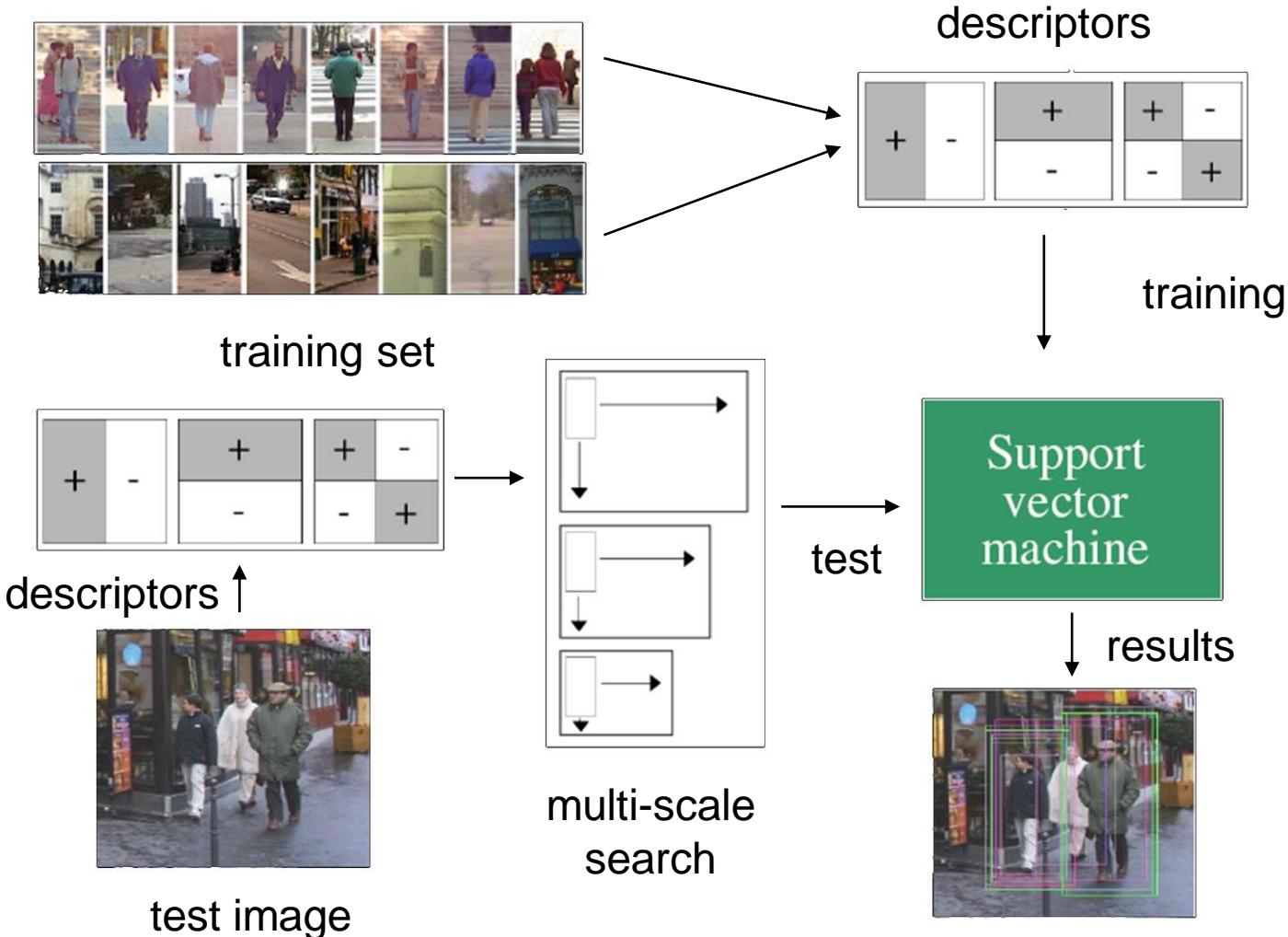
Donatello Conte

HISTOGRAMS OF ORIENTED GRADIENTS (HOG)

Navneet Dalal and Bill Triggs
CVPR '05



Support Vector Machine Detector (Papagergiou & Poggio, 1998)



HOG Steps

- HOG feature extraction
 - Compute centered horizontal and vertical gradients with no smoothing
 - Compute gradient orientation and magnitude
 - Divide de image into overlapping blocks
 - Quantize the gradient orientation into n bins
 - Concatenate histograms

Computing Gradients

- Centered $f' = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x - h)}{2h}$

- Masks in x and y directions

-1	0	1
----	---	---

-1
0
1

- In polar coordinates

- Magnitude $s = \sqrt{s_x^2 + s_y^2}$

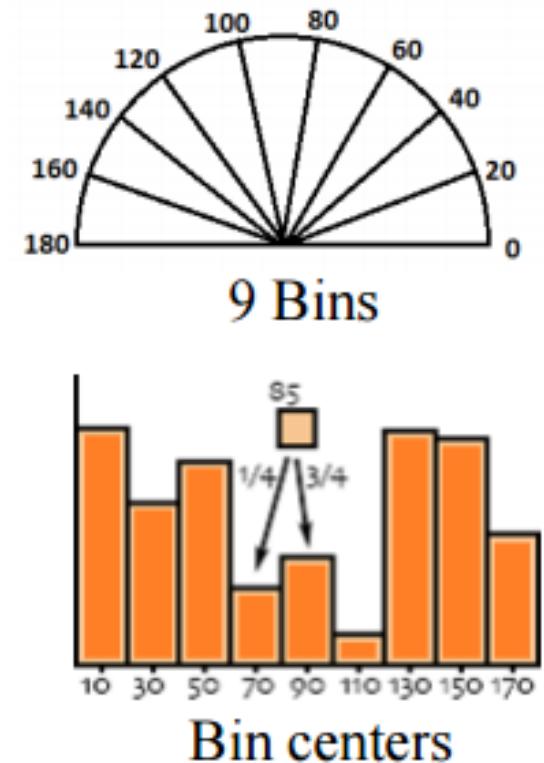
- Orientation $\theta = \arctan\left(\frac{s_y}{s_x}\right)$

Cell Orientation Histogram 1/2

- Divide the window into adjacent, non-overlapping *cells* of size $C \times C$ ($C=8$)
- In each cell, compute a histogram of the gradient orientation binned into B bins ($B=9$)
- Few bins \rightarrow quantization artifacts
- Interpolates votes linearly between neighboring bin centers
 - the vote is the gradient magnitude
 - a pixel with magnitude μ and orientation θ contributes a vote
- $v_j = \mu \frac{c_{j+1} - \theta}{\omega}$ to bin number $j = \left\lfloor \frac{\theta}{\omega} - \frac{1}{2} \right\rfloor \bmod B$
- where ω is the width of a bin

Cell Orientation Histogram 2/2

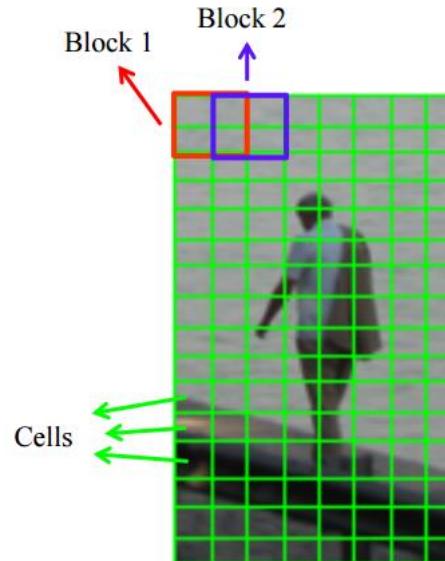
- Example: if $\theta=85$ degrees
- Distance to the bin centers Bin 70 and Bin 90 are 15 and 5 degree, respectively
- Hence, ratios are $5/20=1/4$ and $15/20=3/4$
- The resulting cell histogram is a vector with B nonnegative entries



Block normalization

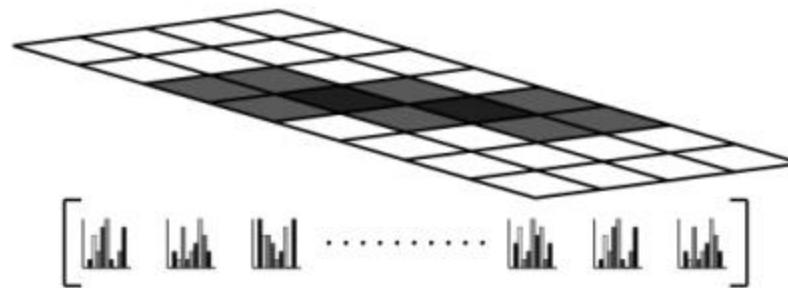
- Group cells into overlapping blocks of 2x2 cells each
- Concatenate the four cell histograms in each block into a single block feature vector \mathbf{b} and normalize the block feature vector by its Euclidean norm

$$\mathbf{b} \leftarrow \frac{\mathbf{b}}{\sqrt{\|\mathbf{b}\|^2 + \epsilon}}$$



HOG Features Vector

- Concatenate histograms



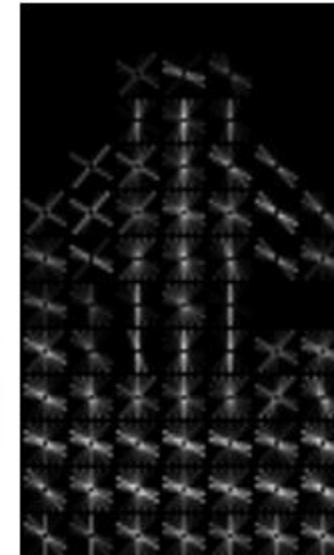
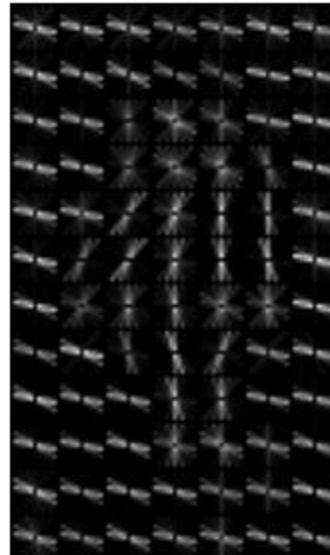
- Normalize and threshold results to make values independent of overall image contrast and to prevent big influence of very large gradients

$$\mathbf{h} \leftarrow \frac{\mathbf{h}}{\sqrt{\|\mathbf{h}\|^2 + \epsilon}} \longrightarrow h_n \leftarrow \min(h_n, \tau) \longrightarrow \mathbf{h} \leftarrow \frac{\mathbf{h}}{\sqrt{\|\mathbf{h}\|^2 + \epsilon}}$$

Some numeric details

- 128x64 window (8192 pixels)
- Cells of 8x8 pixels
- Blocks with 8-pixel overlapping and 4 cells per block
- 9 orientation bins
- 16 cells vertically and 8 horizontally
- 15 blocks vertically and 7 horizontally
- $|\mathbf{h}| = 15 \times 7 \times 4 \times 9 = 3780$
- can be viewed as $15 \times 7 \times 4 = 420$ histograms of 9 values so a 420×9 matrix

Visualization

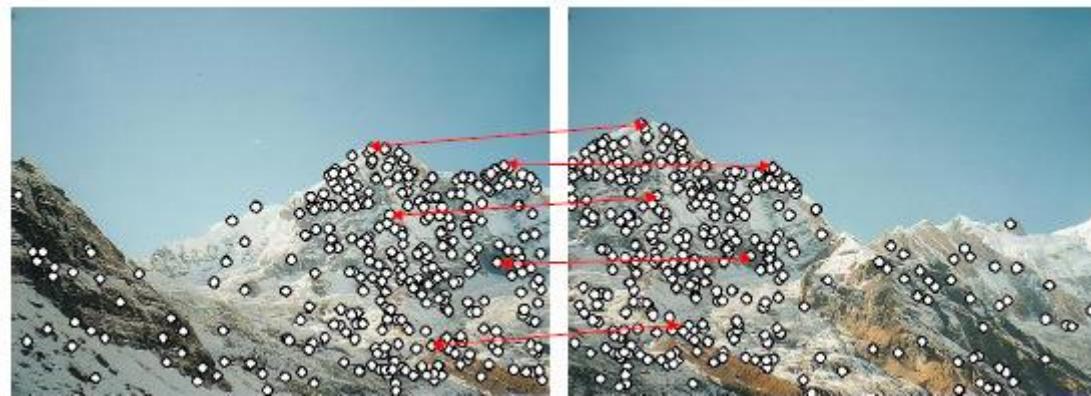


SCALE INVARIANT FEATURE TRANSFORM (SIFT)

David G. Lowe
IJCV '04

Features Detection

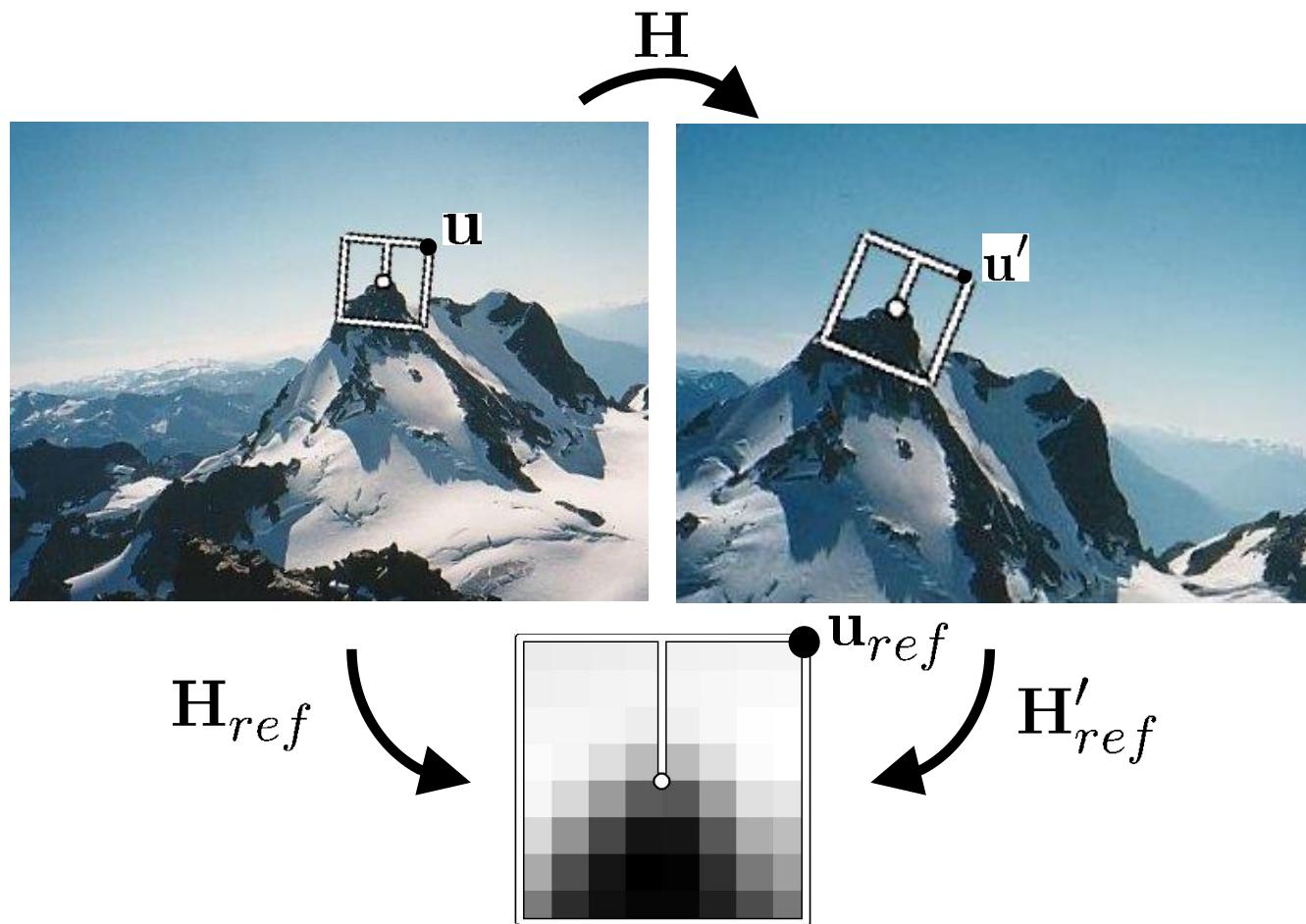
- Give comprehensive description of image.
- Enables matching!



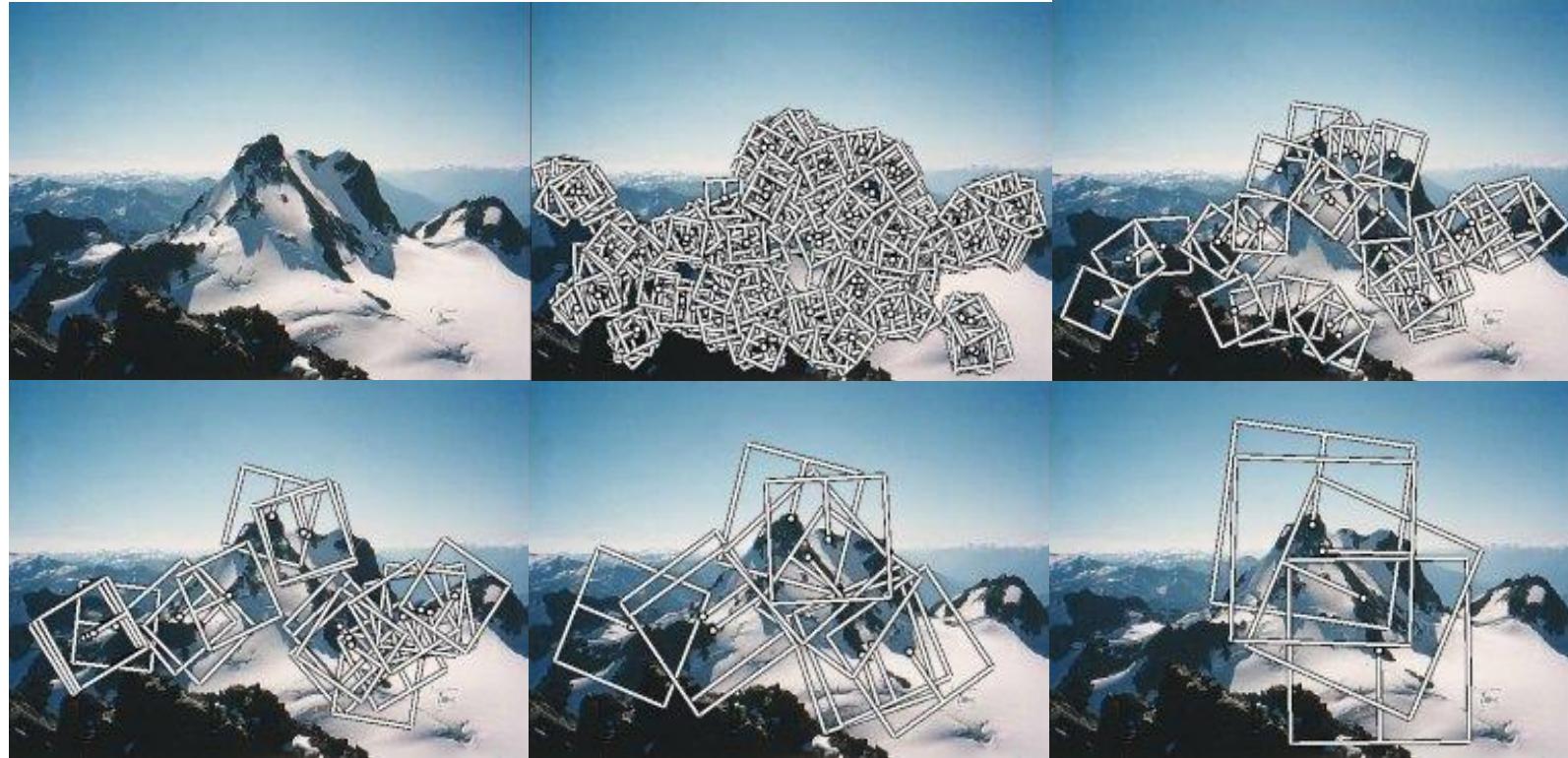
Features' Features

- Robustness => Invariance to changes in illumination, scale, rotation, affine, perspective.
- Locality => robustness to occlusion and clutter.
- Distinctiveness => easy to match to a large database of objects.
- Quantity => many features can be generated for even small objects.
- Efficiency => computationally “cheap”, real-time performance.

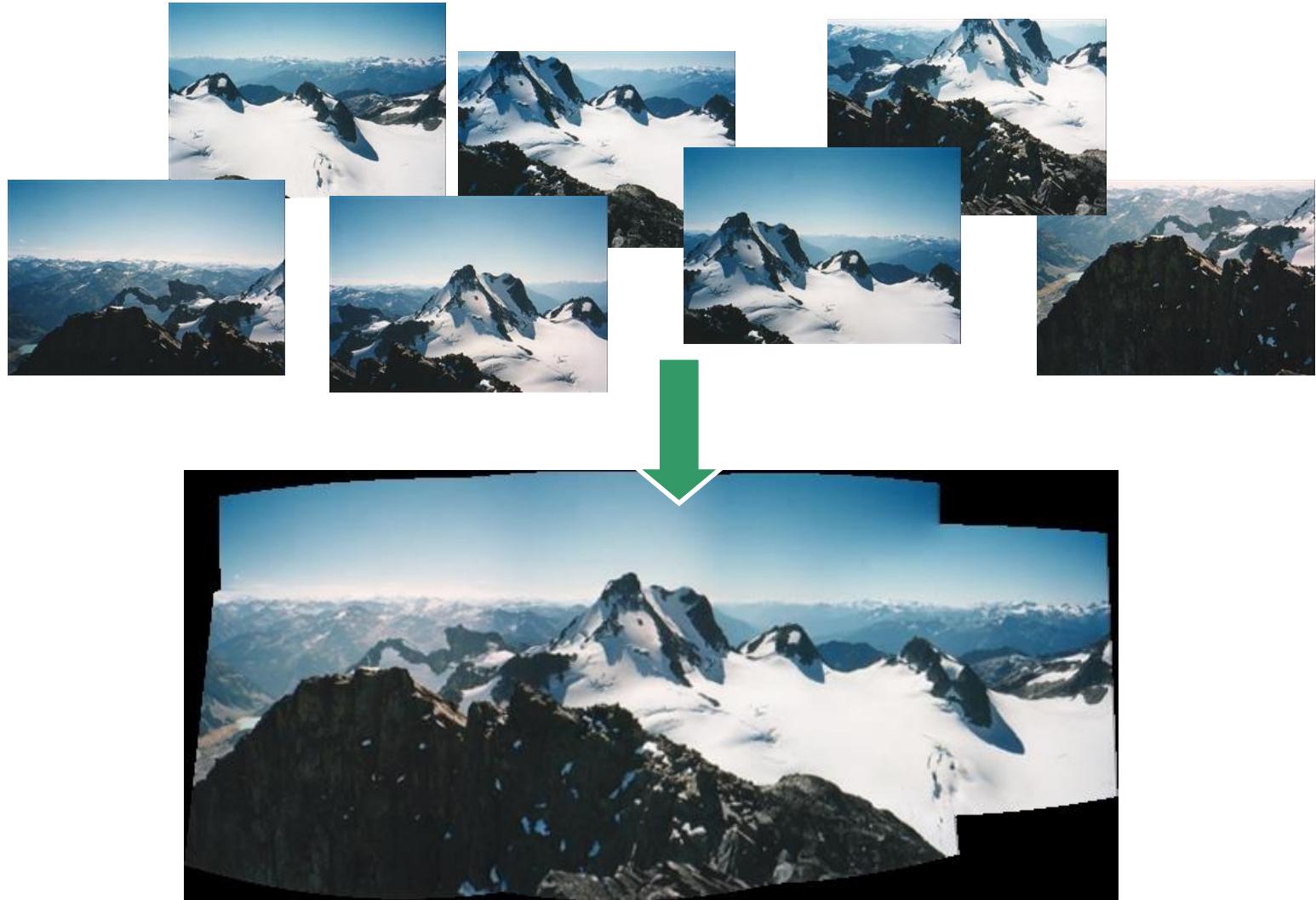
Invariance to rotation



Multi-Scale Oriented Patches



Application: Image Stitching



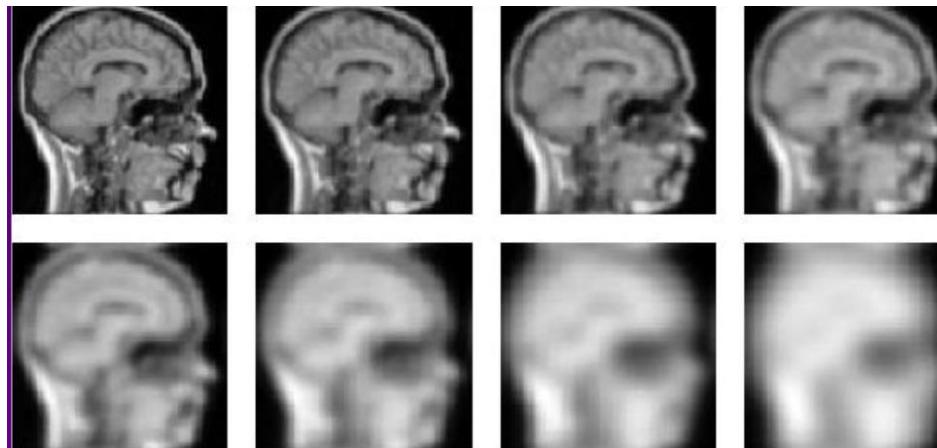
SIFT : the algorithm

- Input: Image ($n \times m$) Output: set of image descriptors

1. Constructing a scale space
2. LoG Approximation
3. Finding keypoints
4. Get rid of bad key points
5. Assigning an orientation to the keypoints
6. Generate SIFT features

1. Scale-space extrema detection

- **Goal:** Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- **Method:** search for stable features across multiple scales using a continuous function of scale.

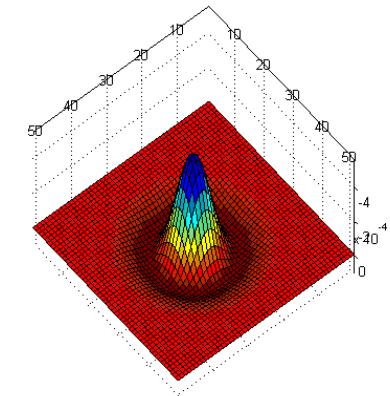
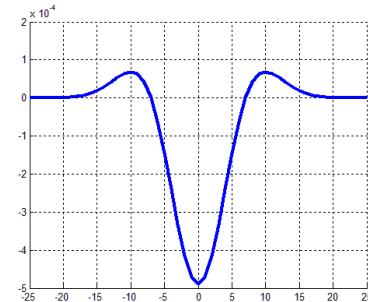
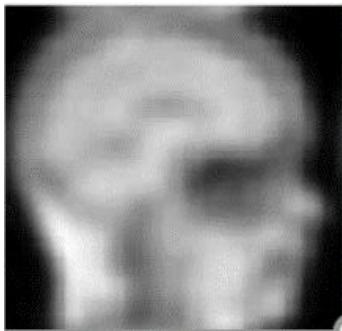


$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Laplacian of Gaussian

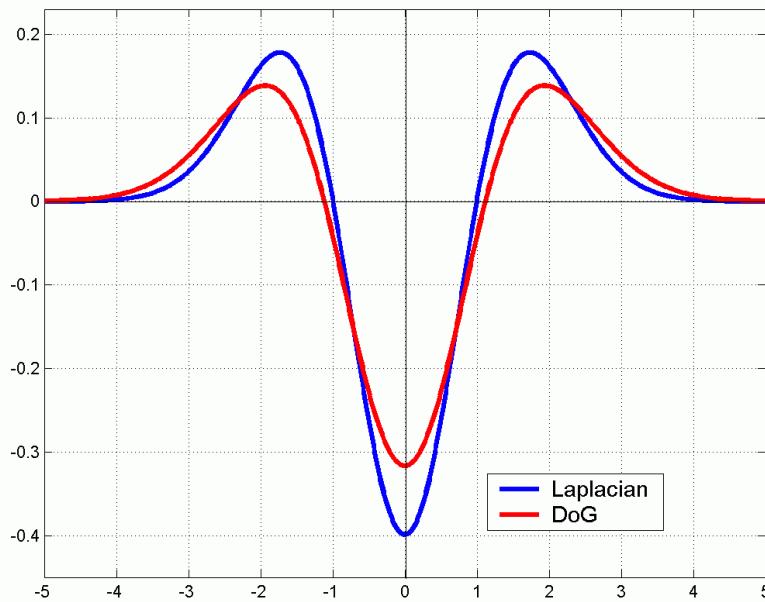
- **Laplacian of Gaussian kernel**
 - Scale normalized (x by scale 2)
 - Proposed by Lindeberg



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

[T. Lindeberg IJCV 1998]

Difference of Gaussians



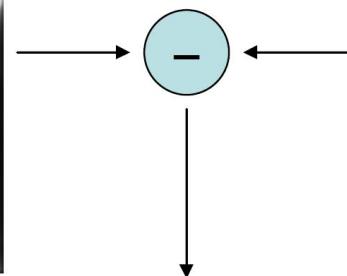
- Laplacian is not invariant to scale, hence

$$\sigma^2 \nabla^2 G$$

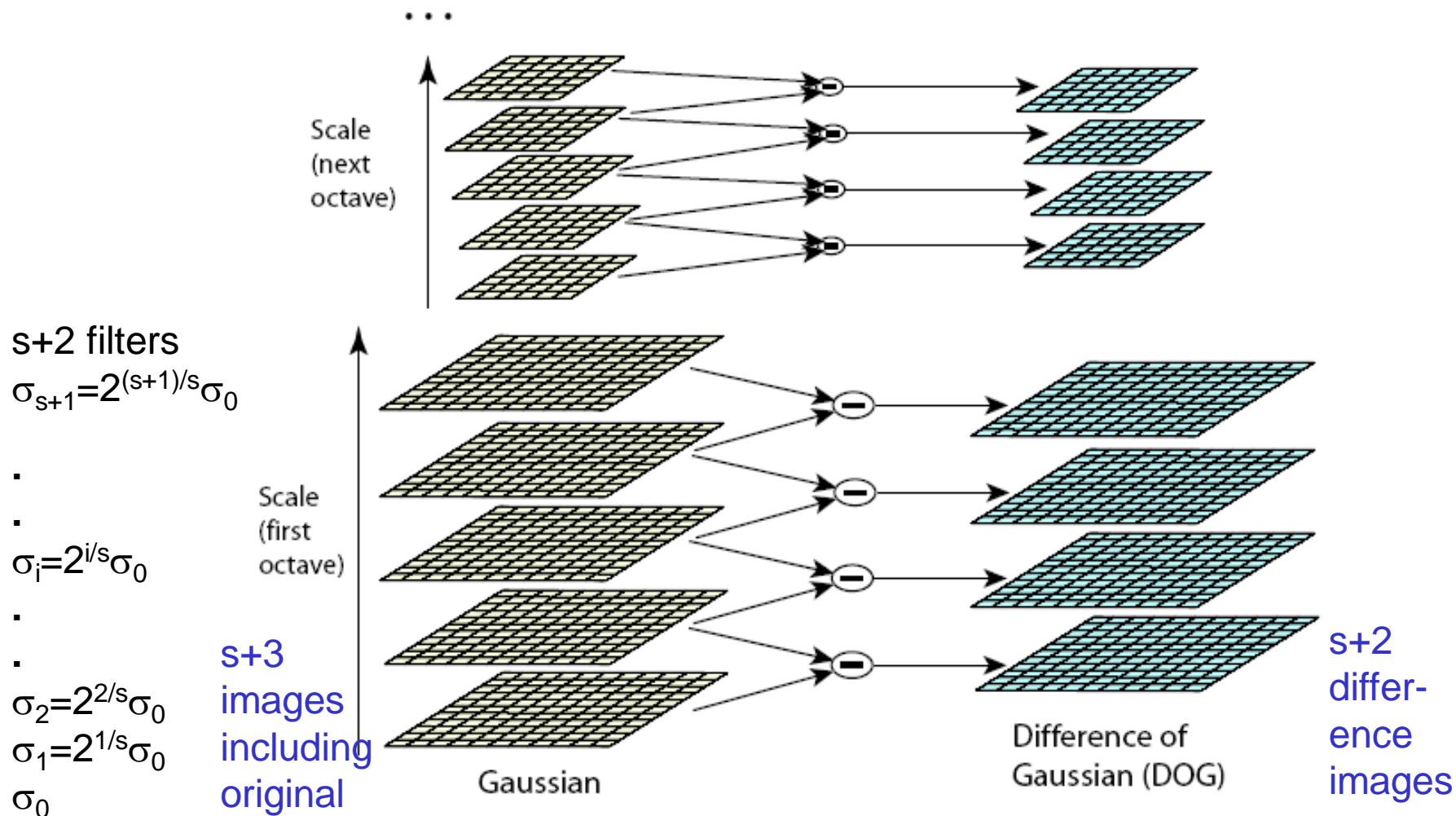
- Laplacian is computationally expensive, so

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G.$$

Difference of Gaussians

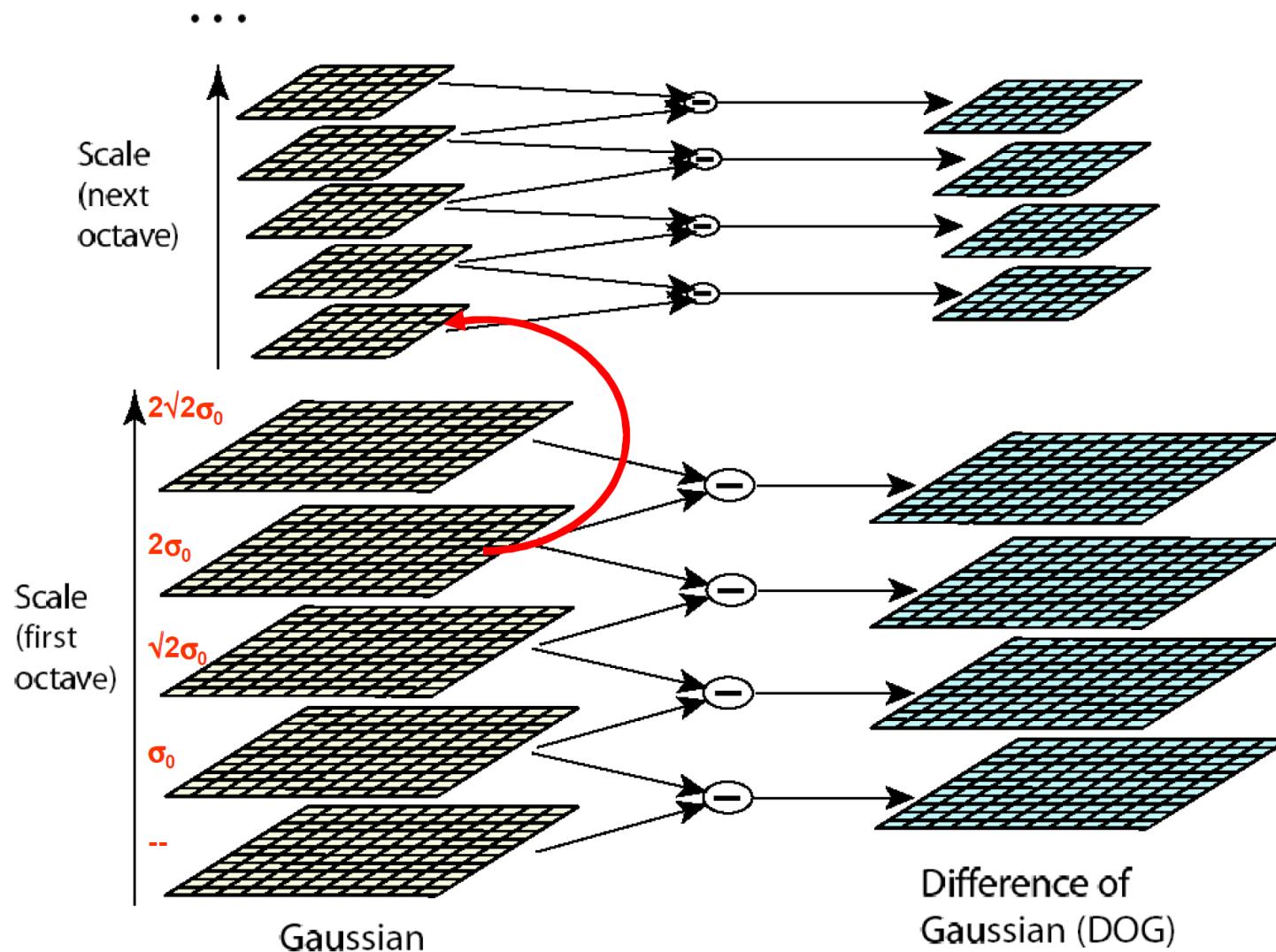


Lowe's Pyramid Scheme



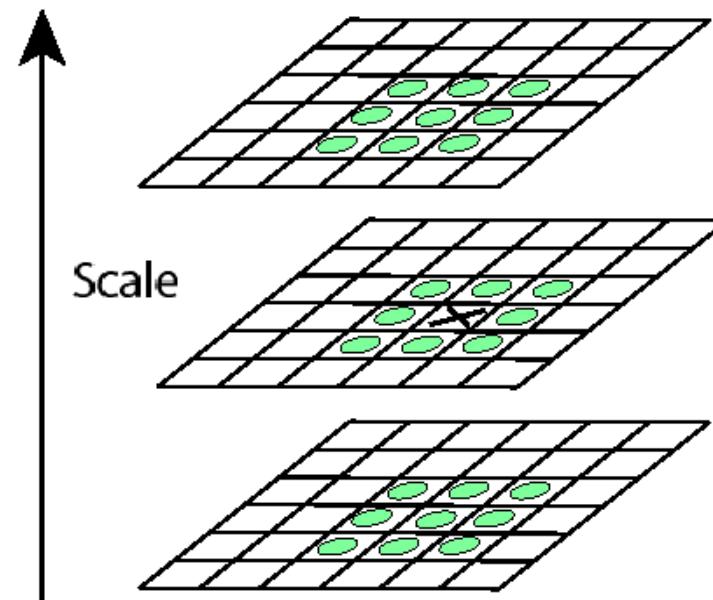
The parameter **s** determines the number of images per octave.

Lowe's Pyramid Scheme



Extracting Keypoints

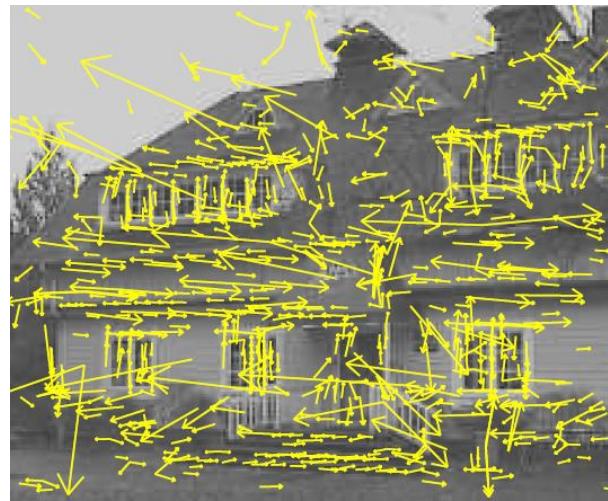
- X is selected if it is larger or smaller than all 26 neighbors.



Low cost - only several usually checked.

Extracting Keypoints

Extrema detection products



233x189 image => 832 DoG Keypoints

- Each Keypoint is represented as (x, y, σ)

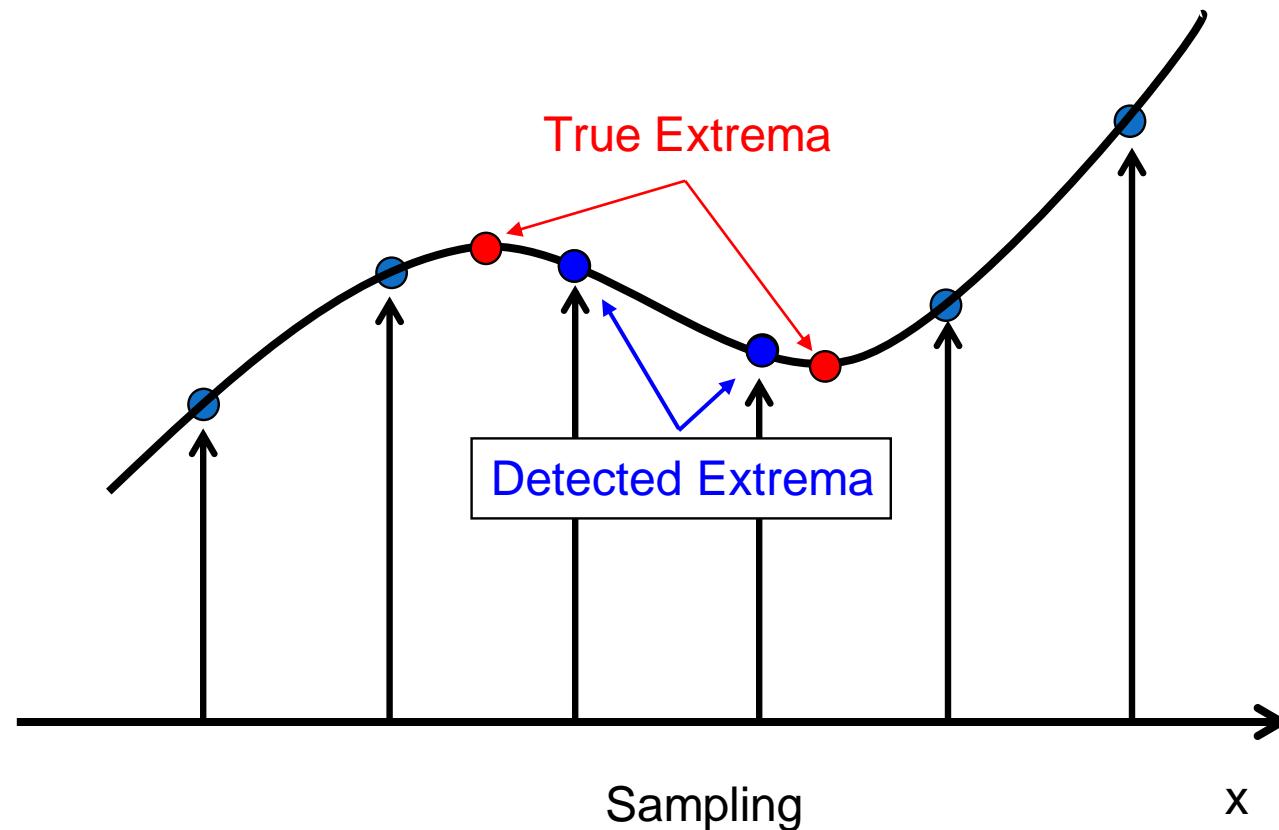
Not all of them are good...

Problematic Keypoints

- Inaccurate localization (due to scaling and sampling).
- Low contrast - sensitive to noise
- Strong edge responses

Inaccurate Keypoint Localization

- The Problem



Inaccurate Keypoint Localization

- The Solution:

- Taylor expansion:

$$D(\vec{x}) = D + \frac{\partial D^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 D^T}{\partial \vec{x}^2} \vec{x}$$

- Minimize to find accurate extrema:

$$\hat{x} = - \frac{\partial^2 D}{\partial \vec{x}^2}^{-1} \frac{\partial D}{\partial \vec{x}}$$

- If offset from sampling point is larger than 0.5 - Keypoint should be in a different sampling point.

Recall: discrete derivative

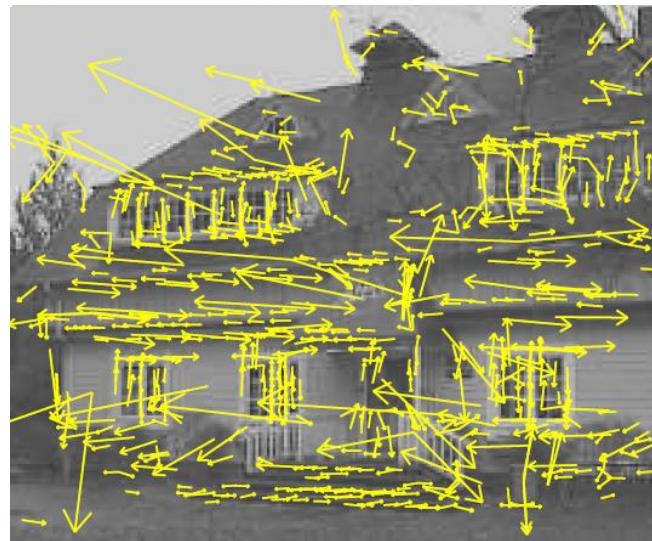
$$\frac{\partial D}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \sigma} \end{bmatrix} = \begin{bmatrix} \frac{D(x+1,y,\sigma) - D(x-1,y,\sigma)}{2} \\ \frac{D(x,y+1,\sigma) - D(x,y-1,\sigma)}{2} \\ \frac{D(x,y,\sigma+1) - D(x,y,\sigma-1)}{2} \end{bmatrix}$$

$$H(\mathbf{x}) = \begin{bmatrix} D_{xx} & D_{xy} & D_{x\sigma} \\ D_{yx} & D_{yy} & D_{y\sigma} \\ D_{\sigma x} & D_{\sigma y} & D_{\sigma\sigma} \end{bmatrix}$$

$$\frac{D(x+1,y+1,s) - D(x-1,y+1,s)}{2} - \frac{D(x+1,y-1,s) - D(x-1,y-1,s)}{2}$$

Low Contrast Keypoints

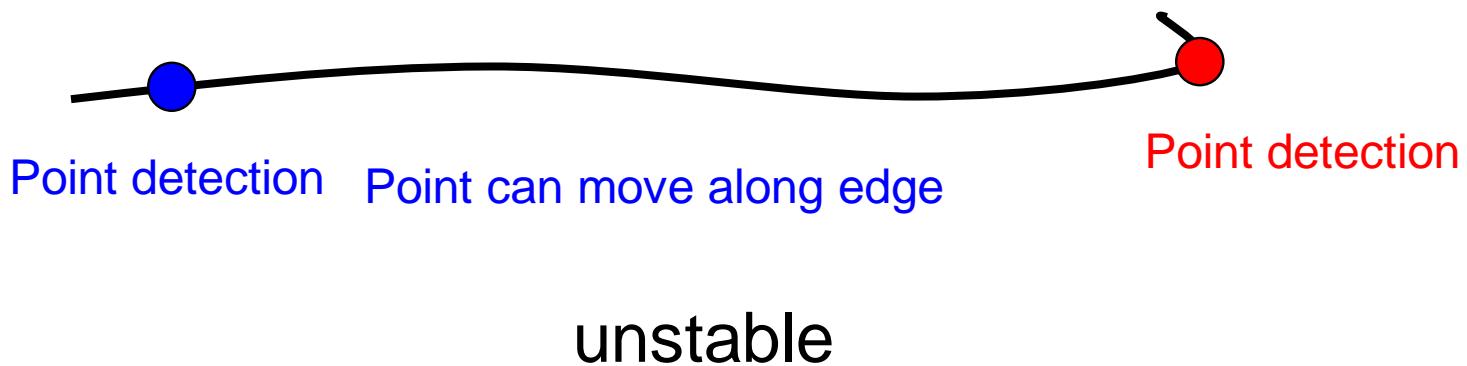
- Function value at the extremum $D(\hat{x})$
 - If $|D(\hat{x})| < 0.03$ (pixel values in range [0,1])
 - keypoint is discarded.



=> down to 729 Keypoints after min. contrast threshold.

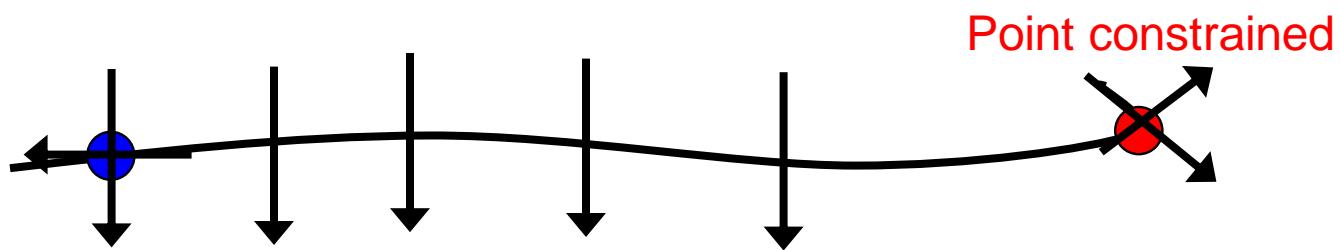
Eliminating Edge Responses

- The Problem
 - “Edge” keypoints are poorly determined.



Eliminating Edge Responses

- The Solution
 - Check Keypoints “cornerness”



- High “cornerness” \Leftrightarrow No dominant principal curvature component

Finding “Cornerness”

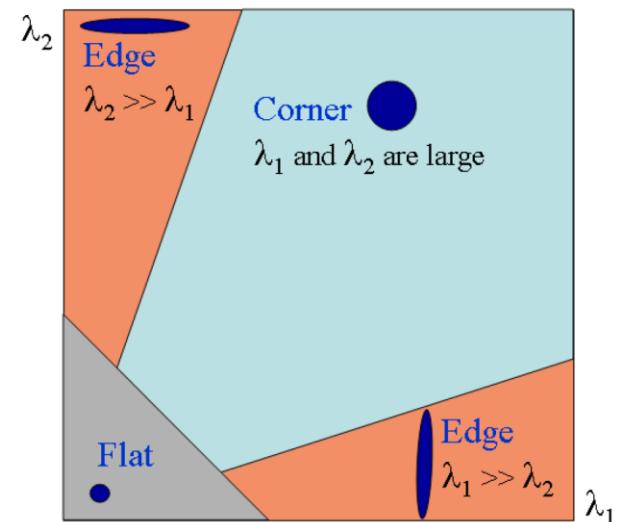
- Principal curvature are proportional to eigenvalues of Hessian matrix: $\lambda_{\max}, \lambda_{\min}$

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

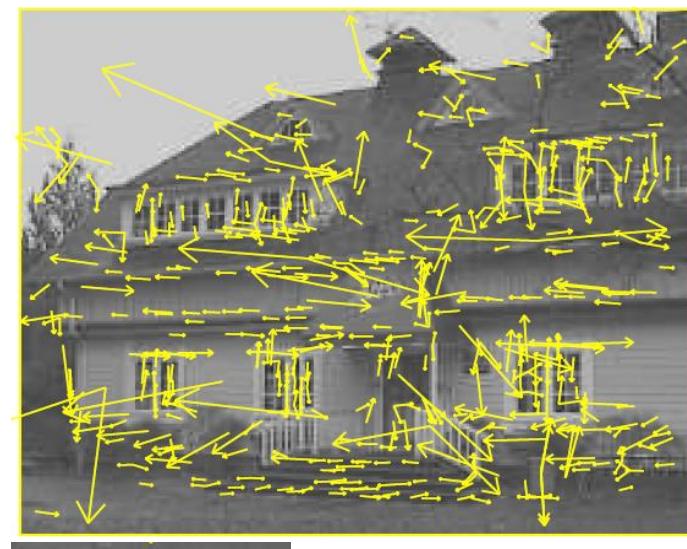
- Harris (1988) showed

$$\frac{\lambda_{\max}}{\lambda_{\min}} < r \Leftrightarrow \frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r+1)^2}{r}$$

- Threshold: if $r < 10$, ratio is too great => keypoint discarded



Stable Keypoints



=> down to 536 Keypoints after “cornerness” threshold.

Keypoint orientations

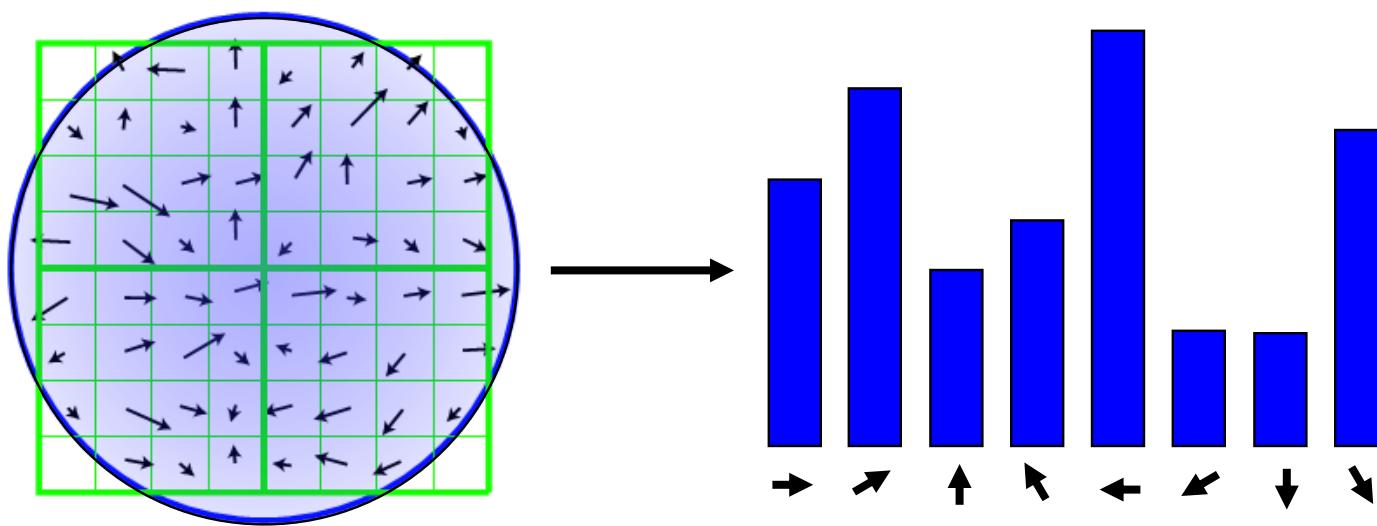
- For each sample point we compute gradient's magnitude and orientation:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1} \left(\frac{(L(x, y+1) - L(x, y-1))}{(L(x+1, y) - L(x-1, y))} \right)$$

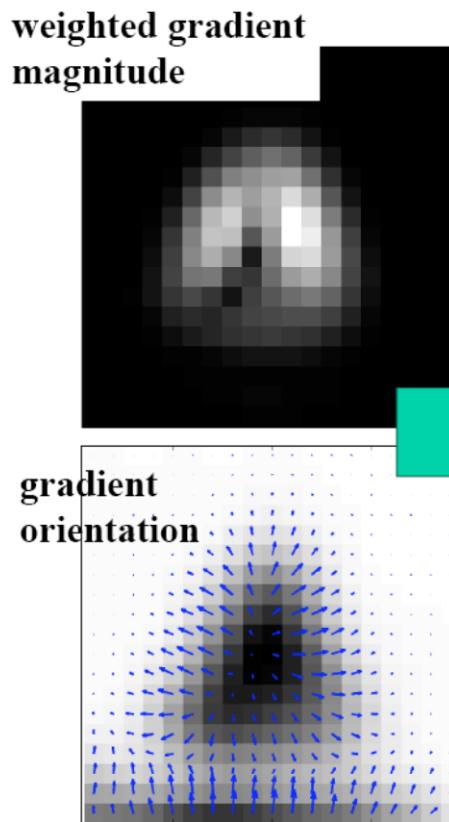
Keypoints Orientation

- Create gradient histogram (36 bins) weighted by magnitude and Gaussian window (σ is 1.5 times that of the scale of a keypoint)

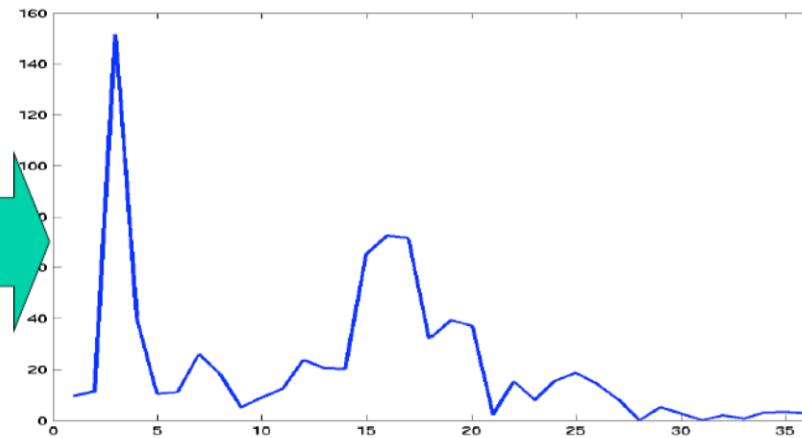


- Any histogram peak within 80% of highest peak is assigned to keypoint (multiple assignments possible).

Keypoint orientations

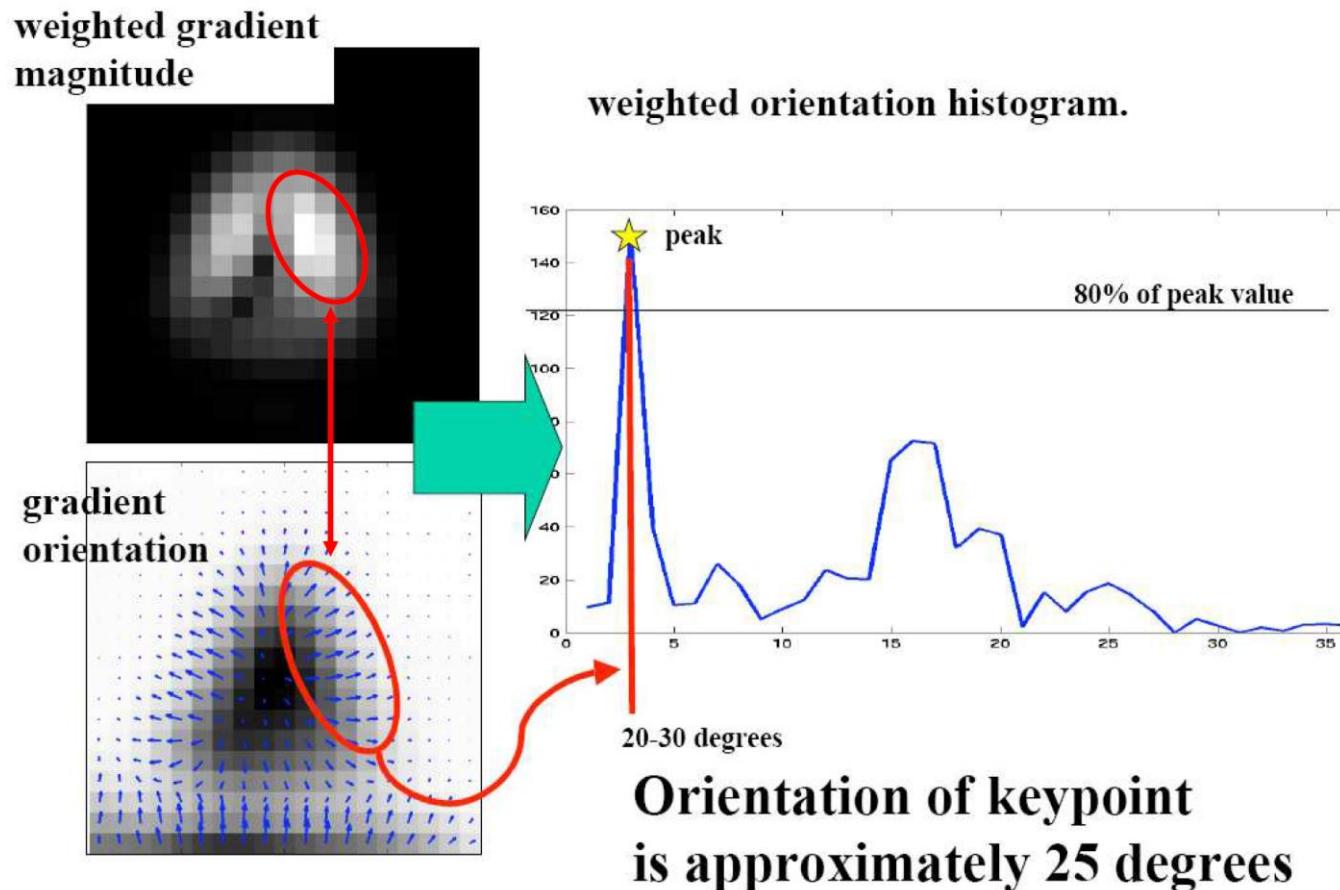


weighted orientation histogram.
Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.

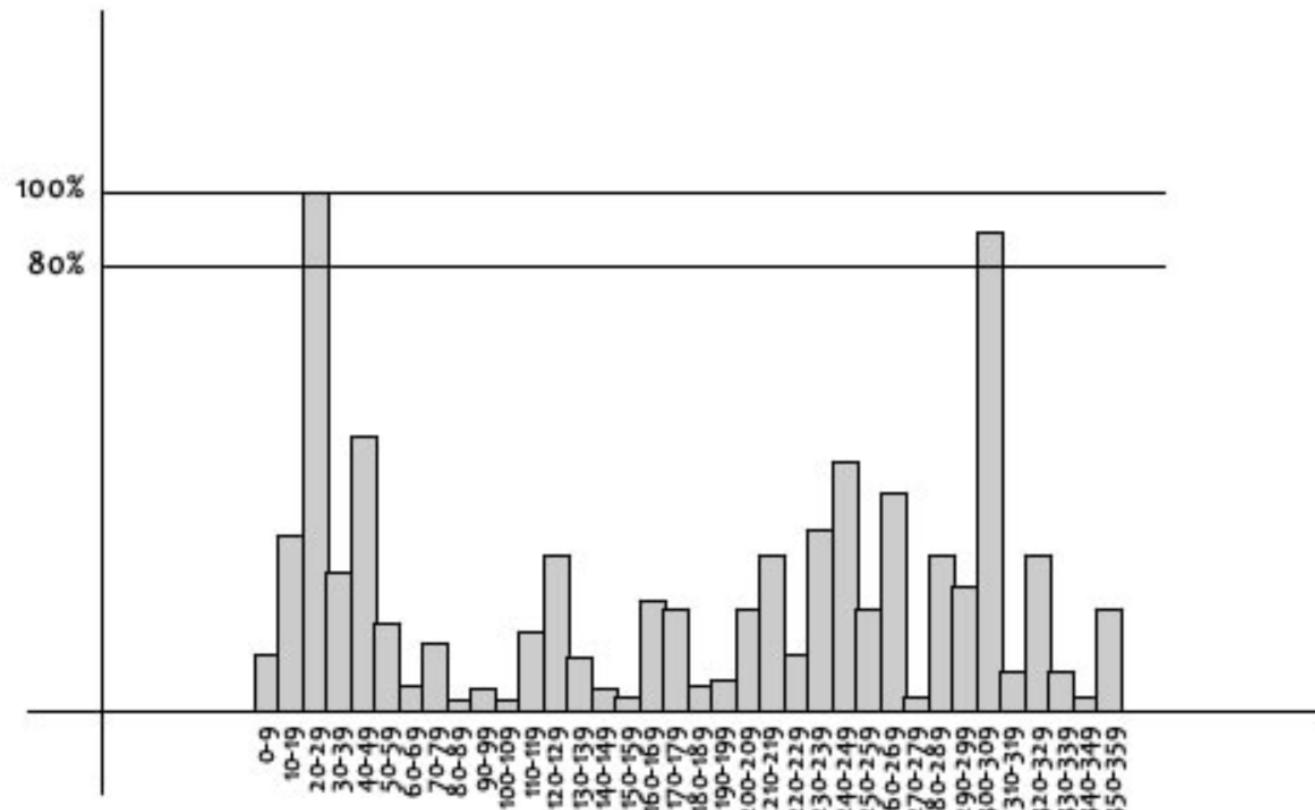


36 buckets
10 degree range of angles in each bucket, i.e.
 $0 \leq \text{ang} < 10$: bucket 1
 $10 \leq \text{ang} < 20$: bucket 2
 $20 \leq \text{ang} < 30$: bucket 3 ...

Keypoint orientations



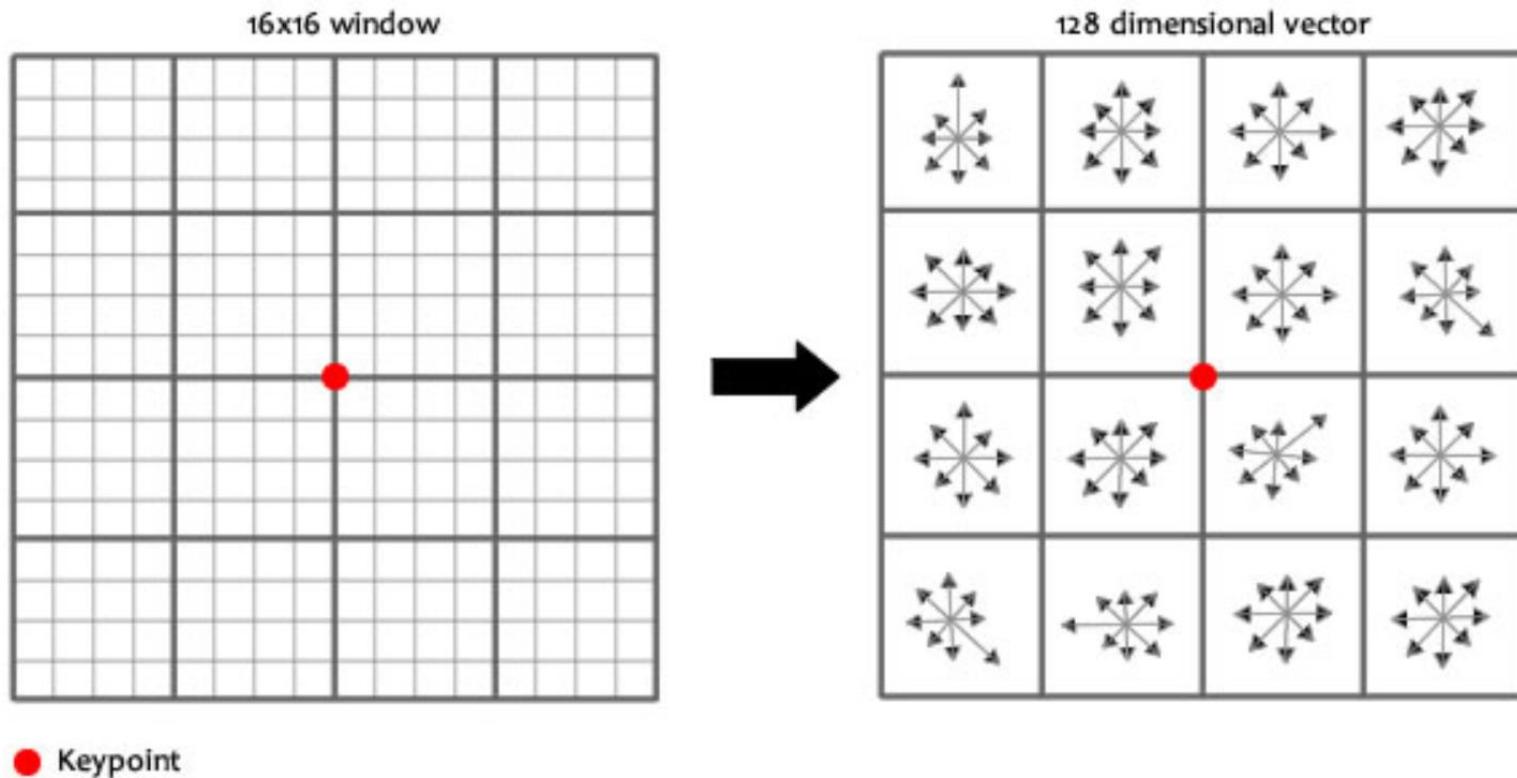
Keypoint orientations



Keypoint Descriptors

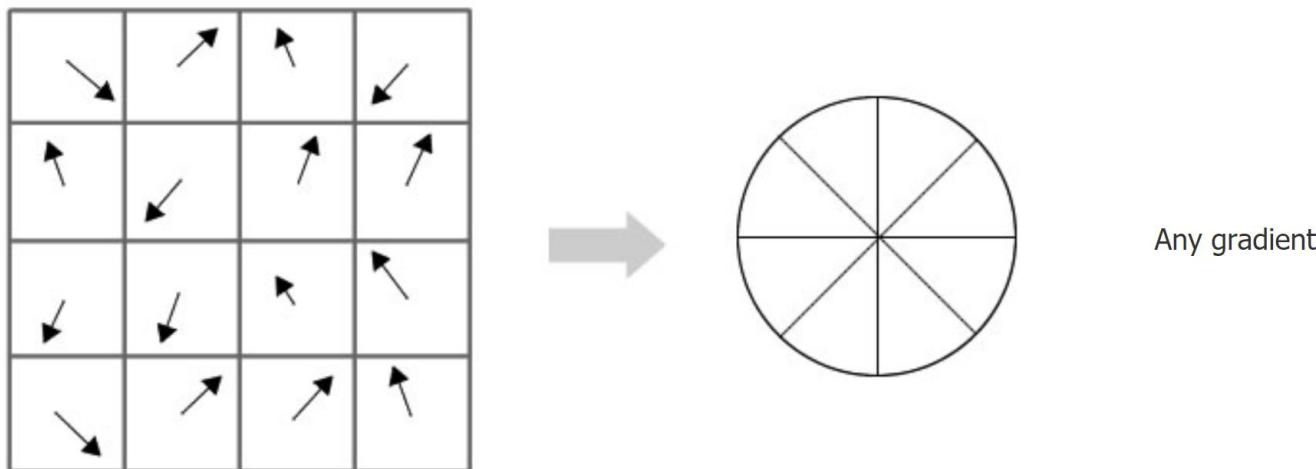
- At this point, each keypoint has
 - location
 - scale
 - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
 - highly distinctive
 - invariant as possible to variations such as changes in viewpoint and illumination

Lowe's Keypoint Descriptor



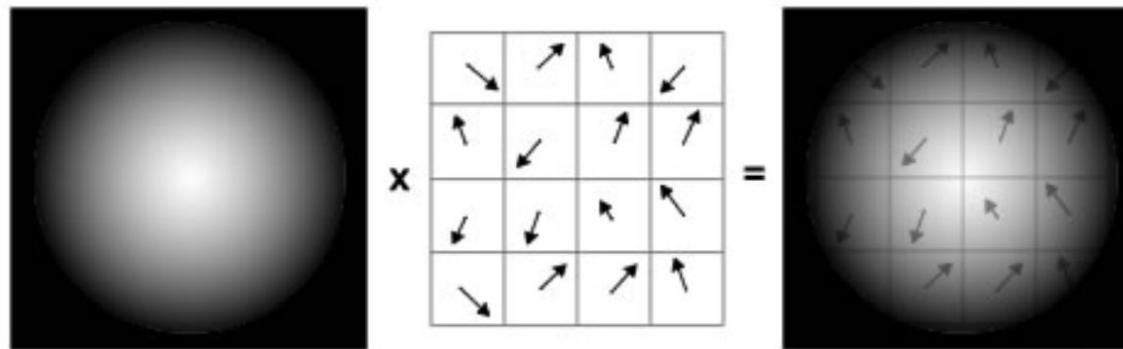
Lowe's Keypoint Descriptor

- compute gradient magnitude and orientation at each point in the region
- create an **orientation histogram** over the 4×4 subregions of the window



Lowe's Keypoint Descriptor

- weight them by a Gaussian window overlaid on the circle



- 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives a **vector of 128 values**.



Lowe's Keypoint Descriptor

- **Rotation dependence**

- to achieve rotation independence, the keypoint's rotation is subtracted from each orientation

- **Illumination dependence**

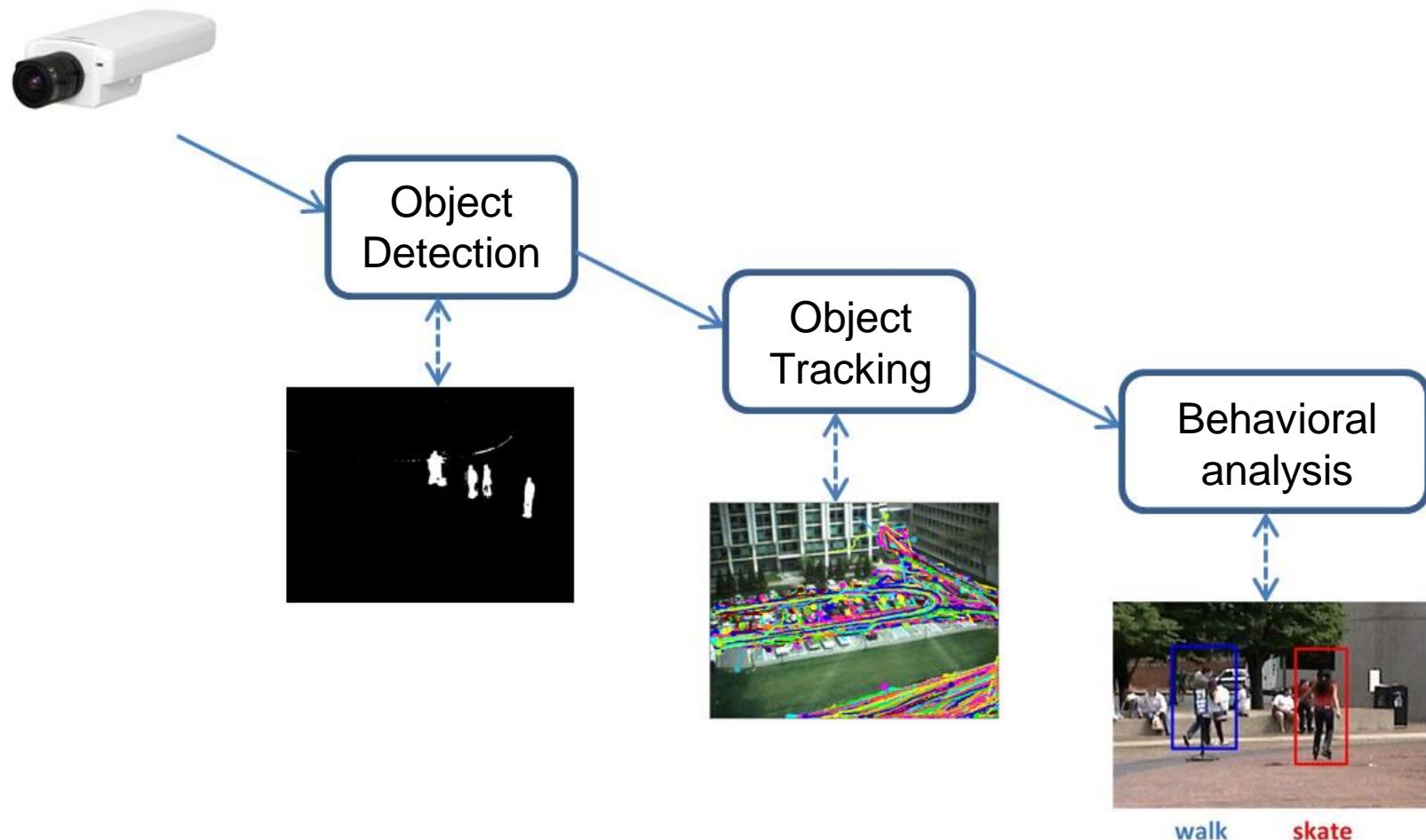
- to achieve invariance to intensity values the vector is normalized to 1 ($\hat{v} = \frac{v}{|v|}$)
 - to reduce the influence of large gradient magnitudes we threshold the values in the unit feature vector to each be no larger than 0.2
 - then renormalizing to unit length

INTRODUCTION TO VIDEO ANALYSIS

Real-time video analysis systems

- Detecting an event of interest in a video sequence
 - The event depends on the application
 - The response also
 - An alert
 - Some statistics
 - Visual Response

General architecture



Object Detection

- Object detection is the process of separating moving objects (or object of interest) from the background of the scene
- Many problems to solve: luminosity changes, camouflages, shadows, reflections ...



1 – from <http://www.telecom.ulg.ac.be/research/vibe/>

Problems



Waving trees



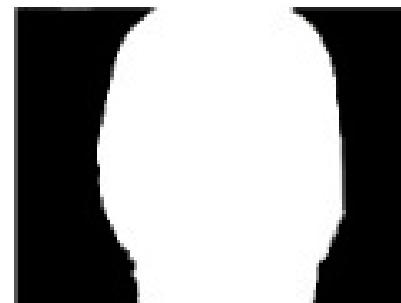
Light of day



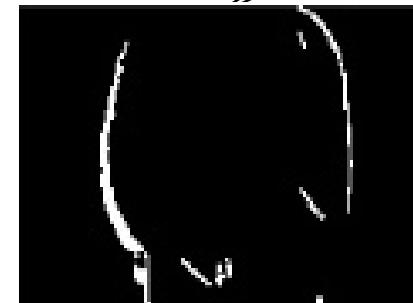
Problems



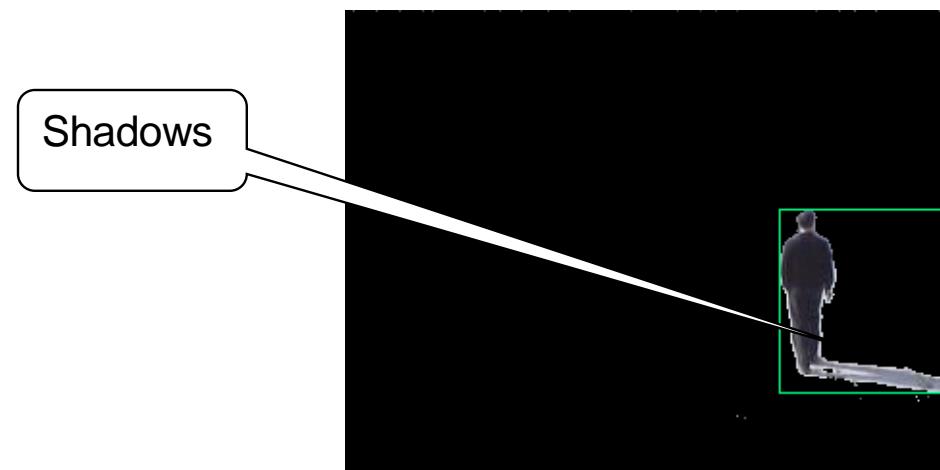
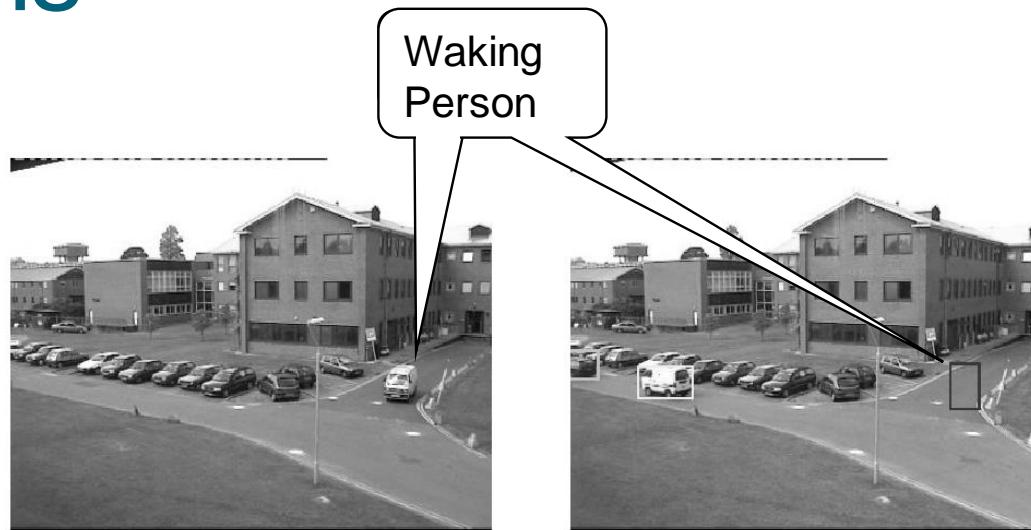
Camouflage



Foreground
aperture



Problems



Main approaches

- Model-based approaches
- Algorithms based on feature points
- Algorithms based on differences

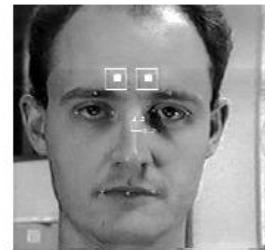
Model-based approaches

- The object to be detected is compared with a model in the current image
- It works well with detection of rigid objects

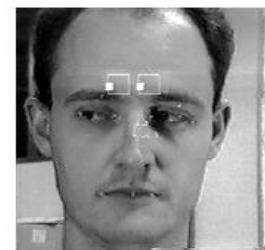


Algorithms based on features points

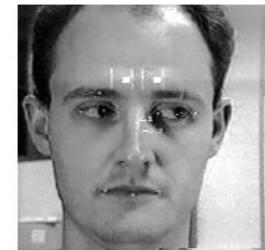
- These approaches detect (and follow) some features points belonging to the objects of interest
- They are good for detecting partially occluded objects
- They are appropriate for density estimation problems (but also for classification)



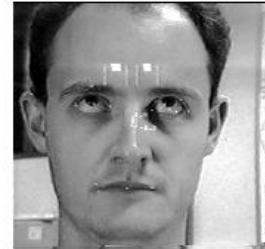
a)



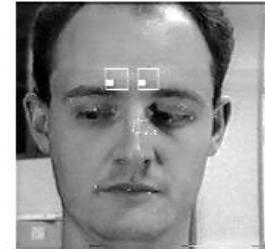
b)



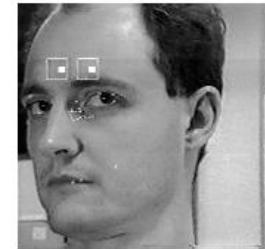
c)



d)



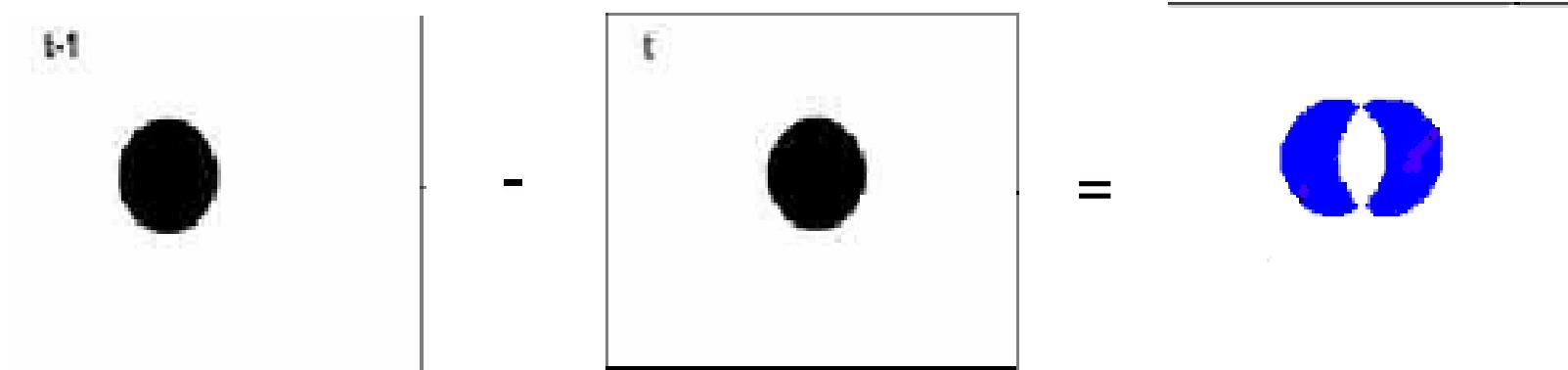
e)



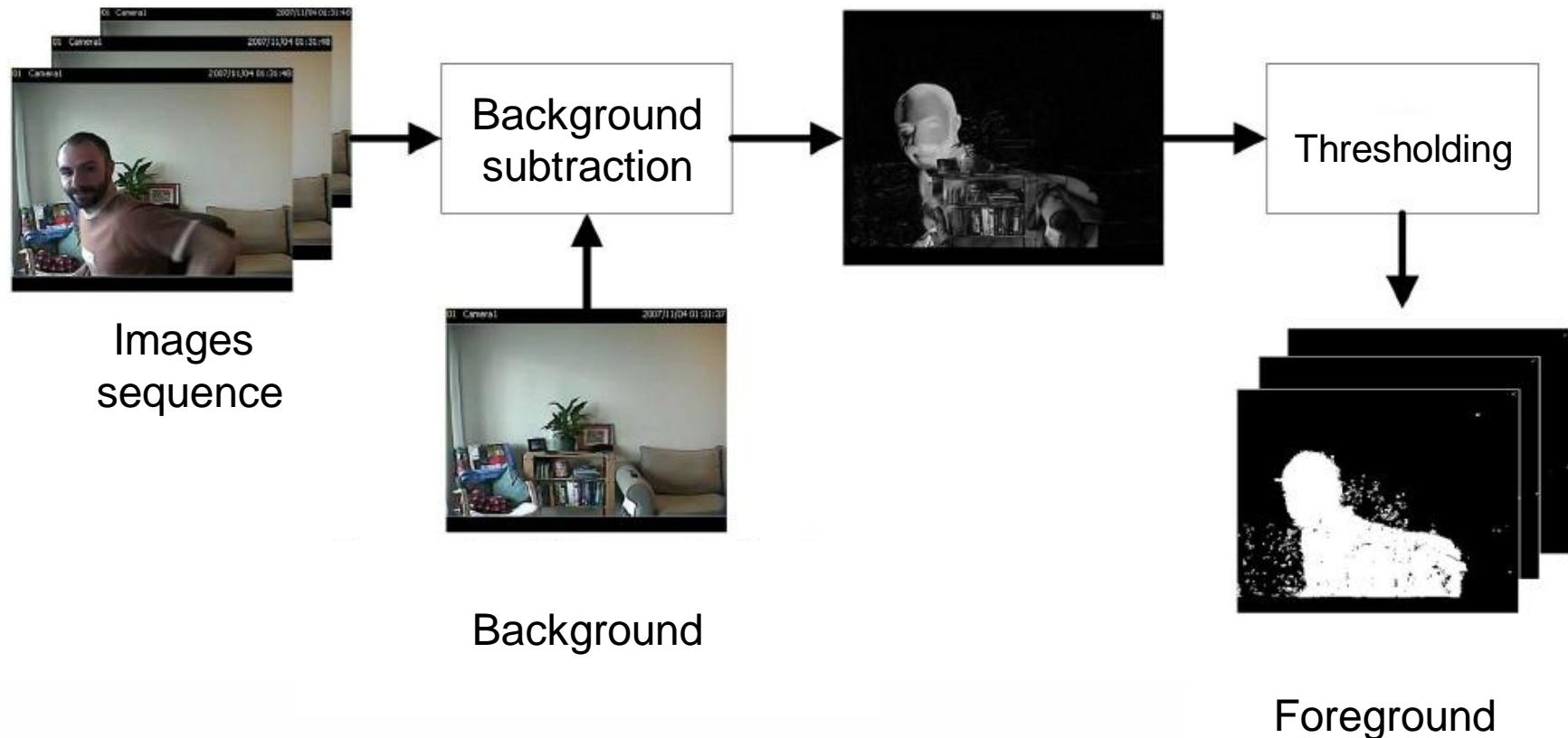
f)

Algorithms based on differences

- The objects of interest are detected by difference between two or more images
 - Difference between consecutive images



Background subtraction algorithm



Background representation

- One value per pixel
- A probability distribution for each pixel
 - One Gaussian¹
 - Mixture of Gaussians²
- Objects contours³

1. C. R. Wren, A. Azarbayejani, T. Darrel, and A. P. Pentland. Pfinder : Realtime tracking of the human body. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19-7 :780–785, 1997.
2. C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22-8 :747–757, 2000.
3. A. Cavallaro and T. Ebrahimi. Change detection based on color edges. Dans *The 2001 IEEE Intern. Symp. on Circuits and Systems, ISCAS*, 2001.

Mixtures of Gaussians

- Model the values of a particular pixel as a mixture of Gaussians
- We determine which Gaussians may correspond to background colors based on the persistence and the variance of each of the Gaussians
- Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them
- Update the Gaussians

Mixture Model

- At any time t , what is known about a particular pixel (x_0, y_0) , is its history

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

- This history is modeled by a mixture of K Gaussian distributions

$$P(X_t) = \sum_{i=1}^K \omega_{i,t} \cdot \mathcal{N}(\mathbf{X}_t | \mu_{i,t}, \Sigma_{i,t})$$

$$\mathcal{N}(\mathbf{X}_t | \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma_{i,t}|^{1/2}} e^{-\frac{1}{2}(\mathbf{X}_t - \mu_{i,t})^T \Sigma_{i,t}^{-1} (\mathbf{X}_t - \mu_{i,t})}$$

- What is the dimensionality of the Gaussian?

Explanations

- At time t we have k Gaussian distributions for each pixel
 - determined by the available memory and computational power (usually, 3-5 are used)
- For each Gaussian we have:
 - $\omega_{i,t}$ is an estimate of the weight of the i^{th} Gaussian in the mixture at time t
 - $\mu_{i,t}$ is the mean values of the i^{th} Gaussian in the mixture, at time t
 - $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian in the mixture, at time t
 - Often $\Sigma_{k,t} = \sigma_k^2 \mathbf{I}$ this assume that the red, green, and blue pixel values are independent and have the same variances

Model Adaptation 1/2

- On-line K-means
- If a new pixel value, X_{t+1} , can be matched to one of the existing Gaussians (within 2.5σ), that Gaussian's $\mu_{i,t+1}$ and $\sigma_{i,t+1}^2$ are updated as follows:

$$\mu_{i,t+1} = (1 - \rho)\mu_{i,t} + \rho\mathbf{X}_{t+1}$$

- and

$$\sigma_{i,t+1}^2 = (1 - \rho)\sigma_{i,t}^2 + \rho(\mathbf{X}_{t+1} - \mu_{i,t+1})^2$$

- where $\rho = \alpha \mathcal{N}(X_{t+1} | \mu_{i,t}, \sigma_{i,t}^2)$ and α is a learning rate

Model Adaptation 2/2

- None of the K distributions match → replace the least probable distribution with a new distribution (μ = current value, high variance, low prior weight)
- Prior weights of all Gaussians are adjusted as follows

$$\omega_{i,t+1} = (1 - \alpha)\omega_{i,t} + \alpha(M_{i,t+1})$$

- where $M_{i,t+1} = 1$ for the matching Gaussian and $M_{i,t+1} = 0$ for all the others

Background and Foreground Modeling

- Heuristic: the Gaussians with the most supporting evidence and least variance should correspond to the background (**why?**)
- The Gaussians are ordered by the value of ω/σ (high support & less variance will give a high value)
- The first B distributions are chosen as the background model

$$B = \operatorname{argmin}_b \left(\sum_{i=1}^b \omega_i > T \right)$$

- where T is minimum portion of the image with is expected to be background

Background updating : selectivity

- It is needing to take into account variations during the sequence
- It can be less or more complex based on goals to achieve

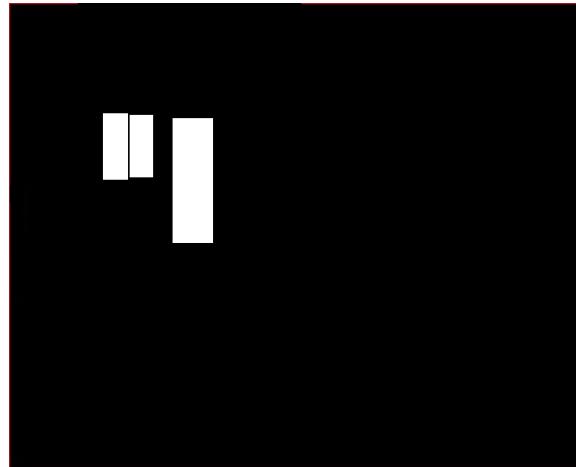
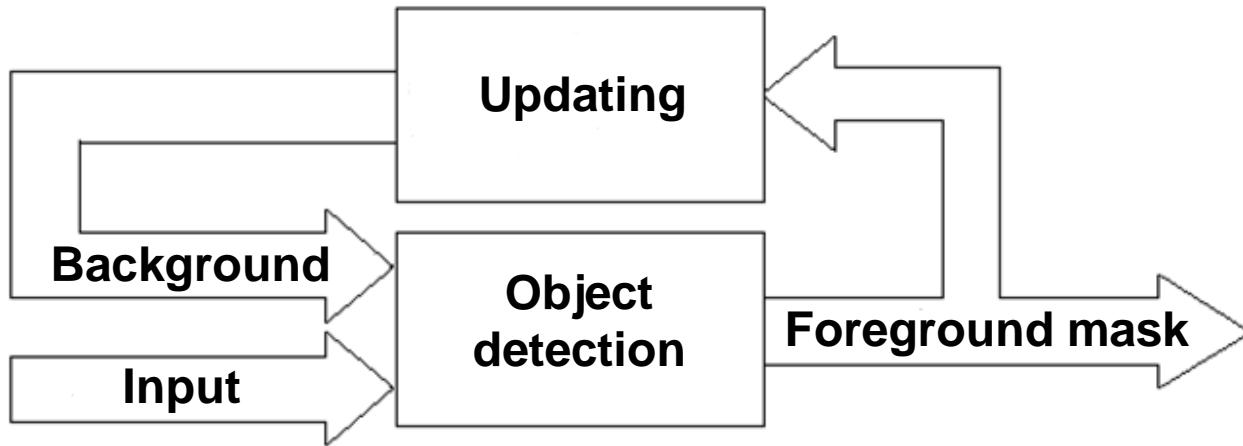
$$B_t(x, y) = \alpha \cdot B_{t-1}(x, y) + (1 - \alpha) \cdot I_t(x, y)$$

- The learning rate α is critic

Selectivity

- Different updating rate (values of α) based on the region to update
 - Slow updating for the foreground region
 - Fast updating for the background

Selectivity

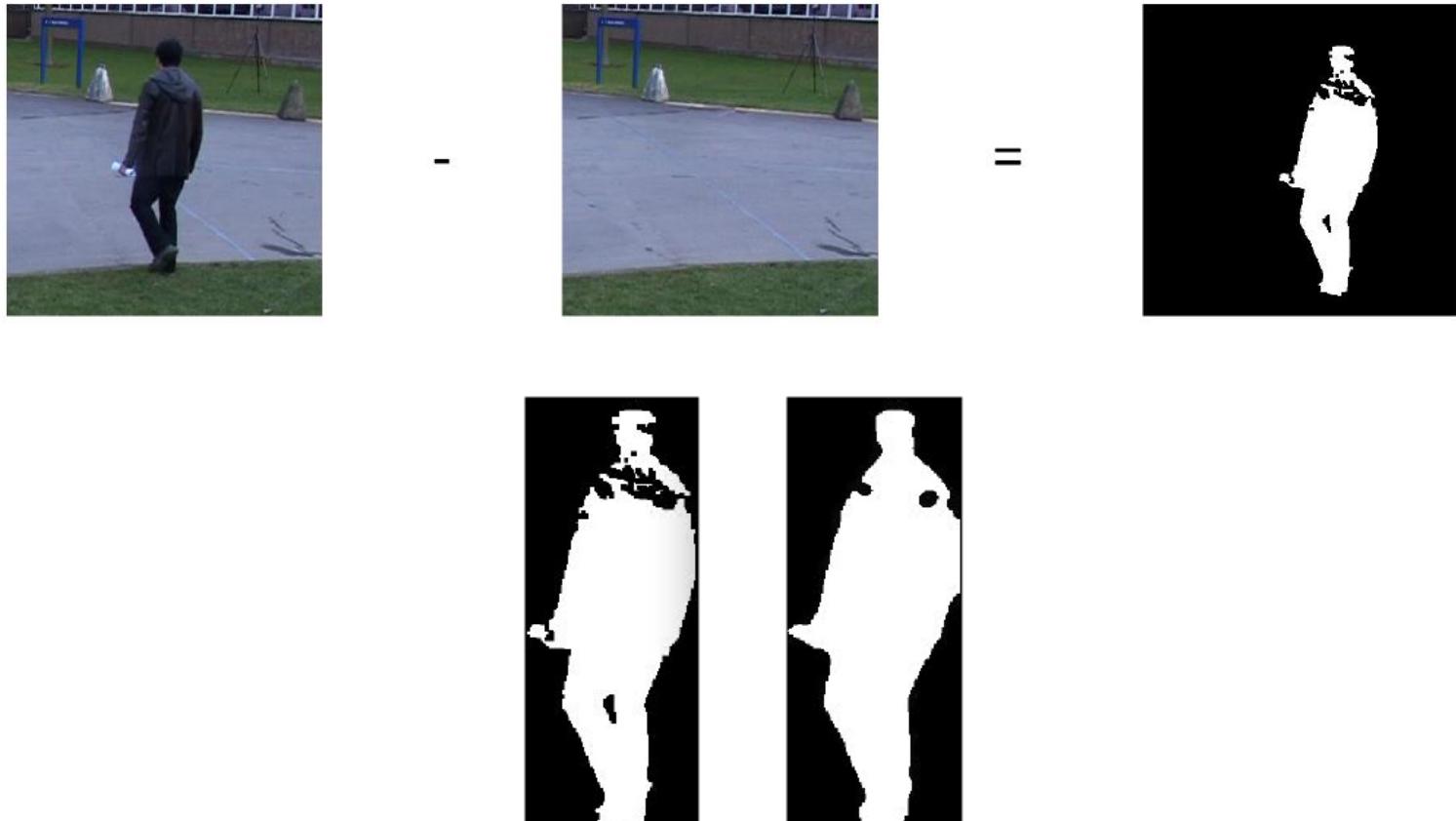


Summary

- No algorithm is globally the best
- Complex algorithms can solve some problems but are also slower
- Idea: we use a fast basic algorithm and we add some post-processing to solve some specific problems

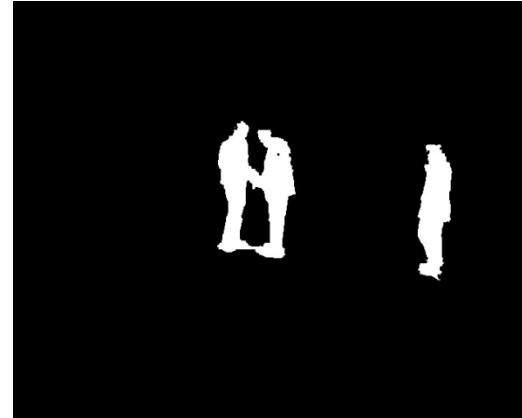
Filtering

- Morphological filtering

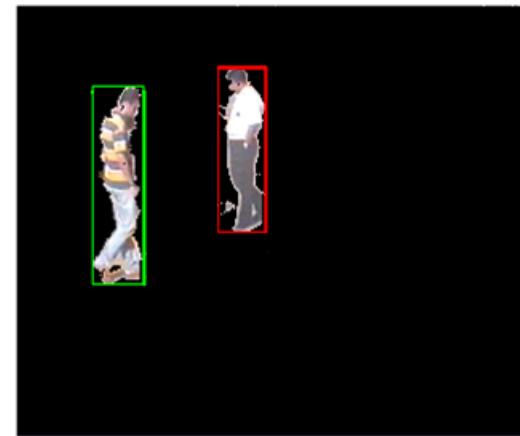
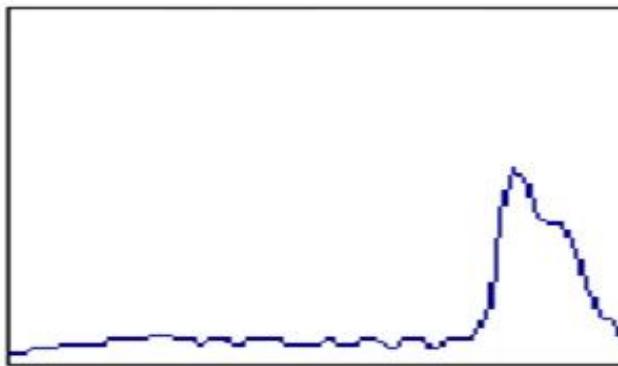


Filtering

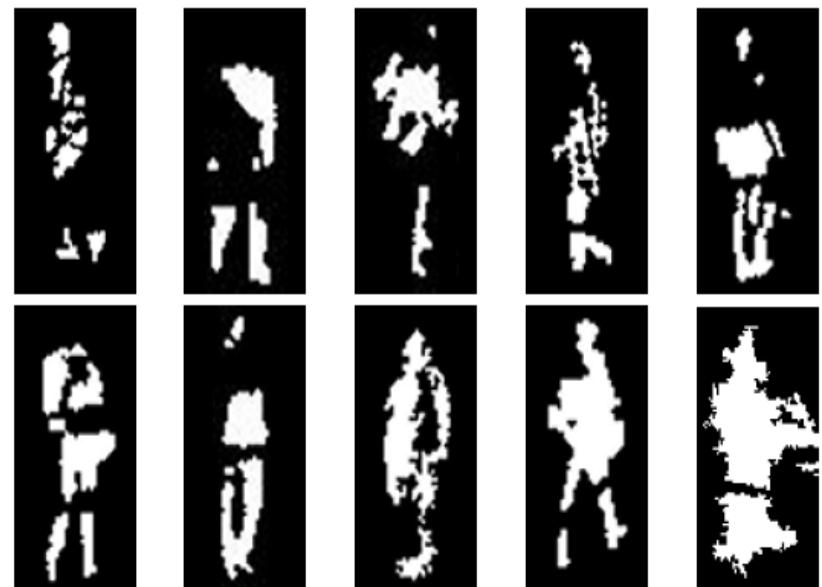
- Size-based filtering



Shadow removing



Camouflage problem



Dealing with camouflage : the idea

- We define
 1. A model for the object of interest
 2. An algorithm to group small regions in a unique region more similar to the model than small ones

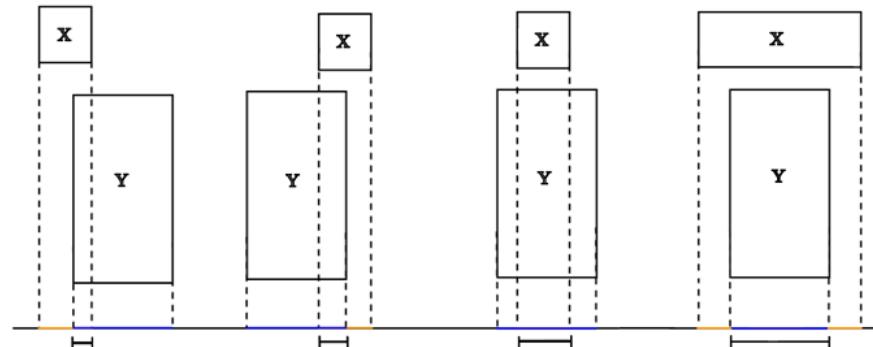
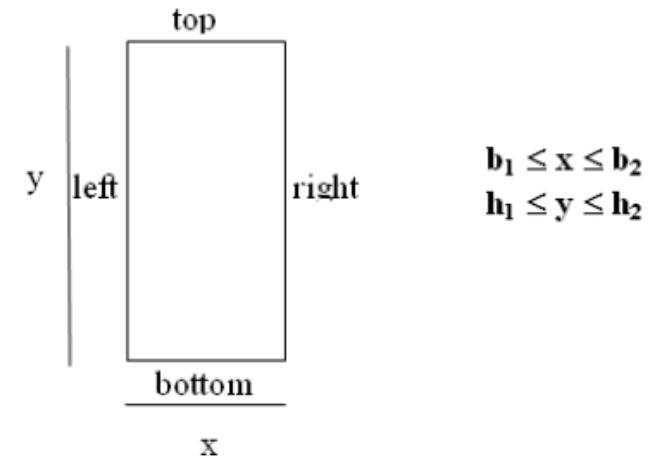
Dealing with camouflage : the model and the algorithm

Algorithm 1. The pseudo-code of the grouping algorithm

```

 $S \leftarrow$  all detected blobs
 $C \leftarrow S \times S$ 
while  $\exists(X, Y) \in C$  do
    comment: Perform and verify the conditions for grouping blobs  $X$  and  $Y$ 
     $R1 \leftarrow right_p(X) \geq left_p(Y) \wedge left_p(X) \leq right_p(Y)$ 
     $Z \leftarrow X \cup Y$ 
    <perform Inverse Perspective Mapping to calculate the actual size of  $Z$ >
     $R2 \leftarrow height_r(Z) \in [h_1, h_2]$ 
     $R3 \leftarrow width_r(Z) \in [b_1, b_2]$ 
    if  $R1 \wedge R2 \wedge R3$  then
        comment: Perform the grouping and update the set of blobs
        <connect the two foreground blobs  $X$  and  $Y$  by joining their barycenter>
         $S \leftarrow S - \{X, Y\}$ 
         $S \leftarrow S \cup \{Z\}$ 
         $C \leftarrow S \times S$ 
    else
         $C \leftarrow C - \{(X, Y)\}$ 
    end if
end while

```



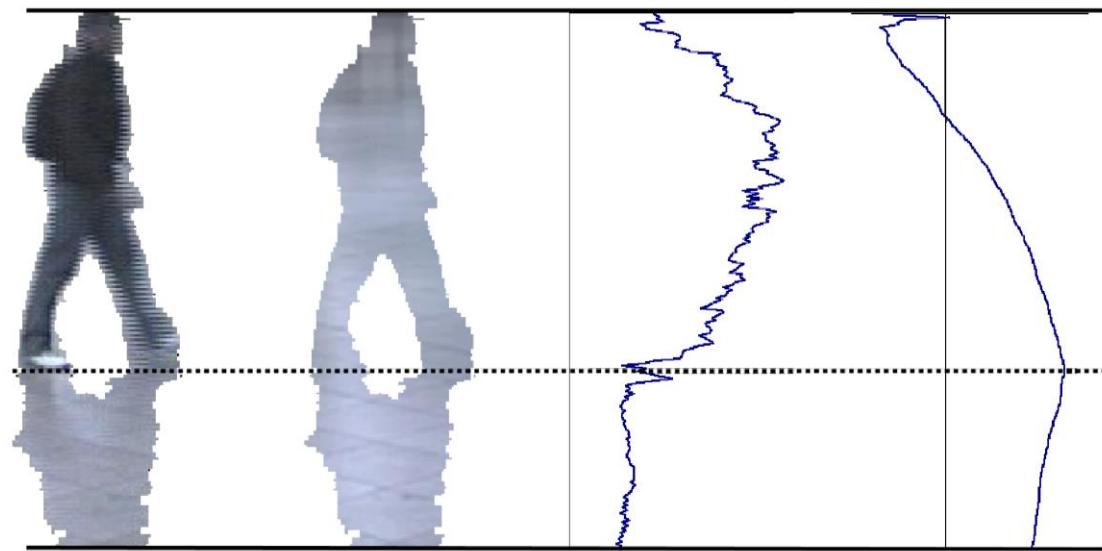
Dealing with camouflage : an example



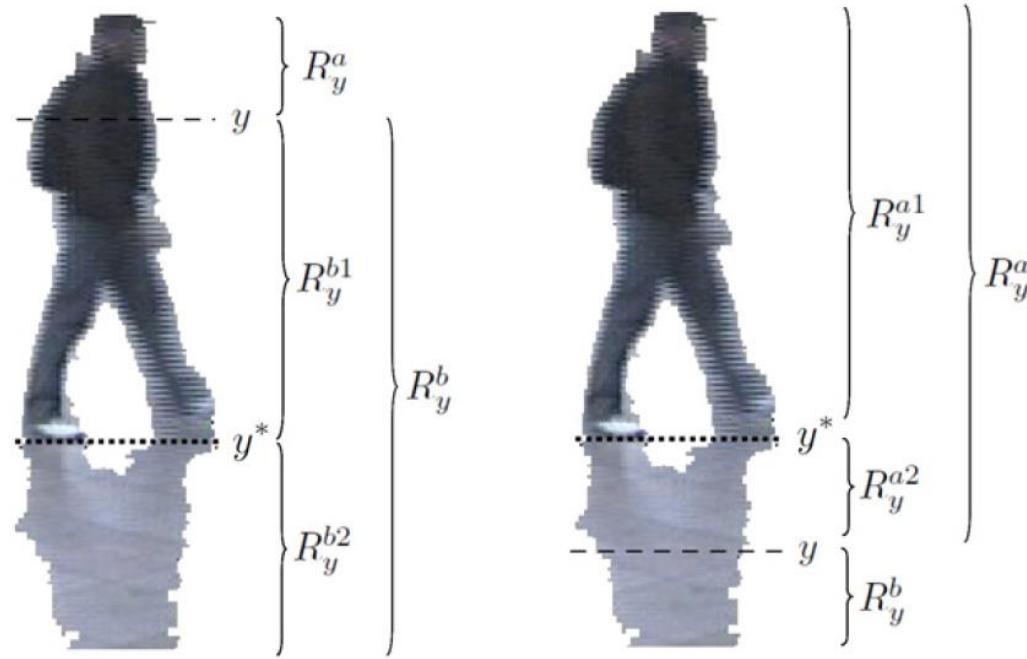
Current image and details of an object



Reflections removal



Reflections removal



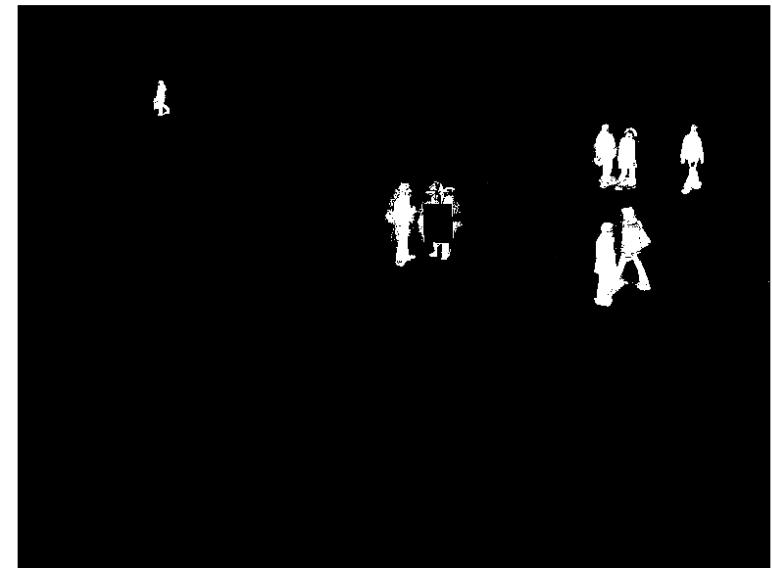
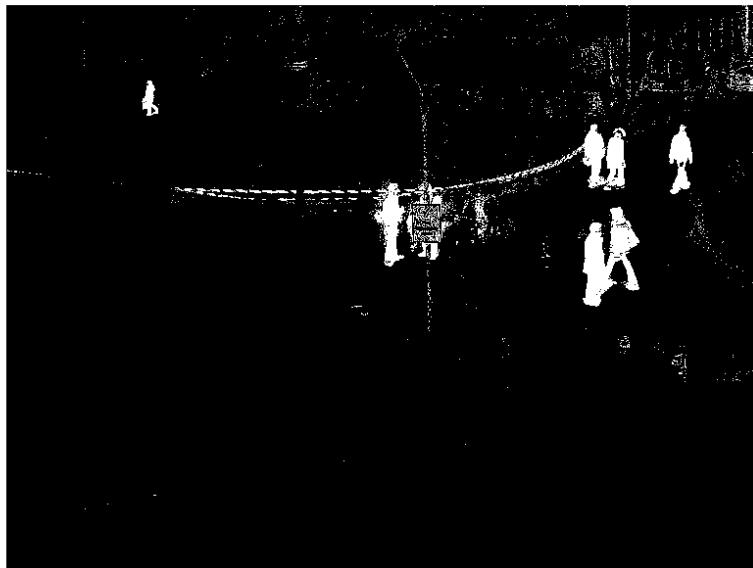
$$d_y = \frac{\sum_{(x,y) \in \mathcal{R}_y} \|F(x, y) - B(x, y)\|}{|\mathcal{R}_y|} \quad \Delta_y = D_y^a - D_y^b$$

$$D_y^a = \frac{1}{|\mathcal{R}_y^a|} \cdot \sum_{i \in \mathcal{R}_y^a} d_i$$

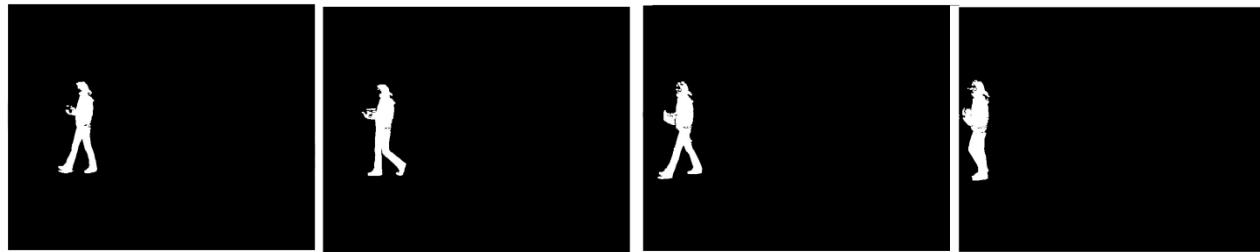
$$D_y^b = \frac{1}{|\mathcal{R}_y^b|} \cdot \sum_{i \in \mathcal{R}_y^b} d_i$$

A new approach

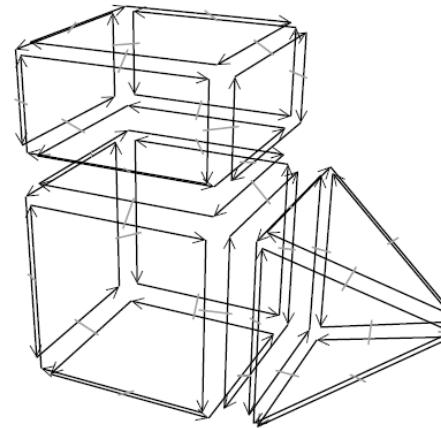
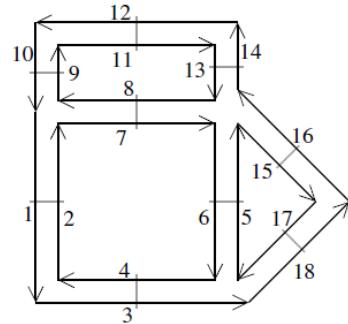
- In above heuristics temporal information are not exploited
- This algorithm is more generic



A structural representation of a video



Algebraic Definition of a nD combinatorial map

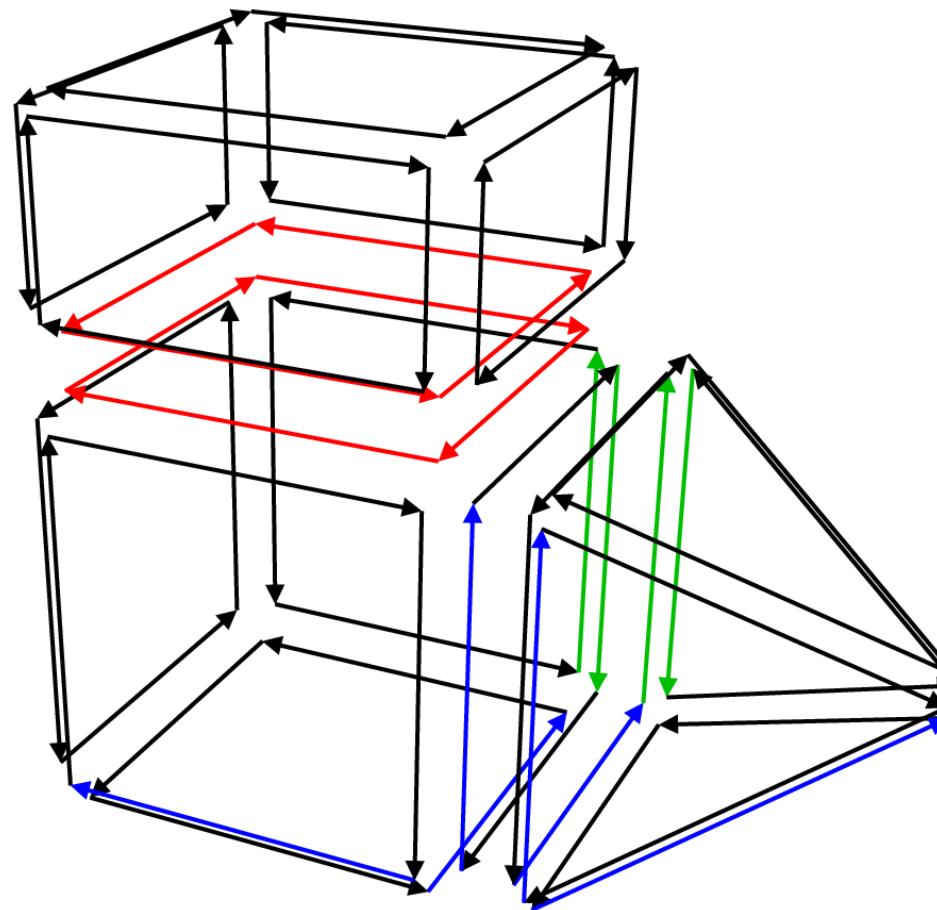


Definition (Combinatorial Map)

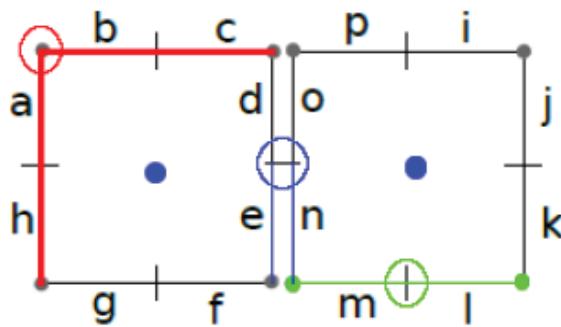
Let be $n \geq 0$. A n combinatorial map (or n -map) is an algebra $C = (D, \beta_1, \dots, \beta_n)$ where:

- ① D is a finite set of **darts**;
- ② β_1 is a **permutation** on D ;
- ③ $\forall i, 2 \leq i \leq n, \beta_i$ is an **involution** on D ;
- ④ $\forall i, 1 \leq i \leq n - 2, \forall j, i + 2 \leq j \leq n, \beta_i \circ \beta_j$ is an **involution**.

3D Maps and cells



Example



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
α_0	h	c	b	e	d	g	f	a	p	k	j	m	l	o	n	i
α_1	b	a	d	c	f	e	h	g	j	i	l	k	n	m	p	o
α_2	a	b	c	o	n	f	g	h	i	j	k	l	m	e	d	p

- α_0 sews each dart to the dart of the adjacent 0-cell (vertex)
- α_1 sews each dart to the dart of the adjacent 1-cell (edge)
- α_2 sews each dart to the dart of the adjacent 2-cell (face)
- ...
- α_n sews each dart to the dart of the adjacent n -cell (n -dimensional simplicial complex)

Principles of the method

- Compute the 3D topological map of a video of foreground masks (*nb* frames)
- Noise are white regions which are badly labeled
- Recognize these regions and merge them with the background

The algorithm

Algorithm 1: Reduce noise on foreground masks

Input: A video of foreground masks V ;
A boolean function $criterion(r_1, r_2)$.

Result: V is modified by merging all the adjacent pairs of regions satisfying $criterion$.

$T \leftarrow$ build the 3D topological map describing V ;

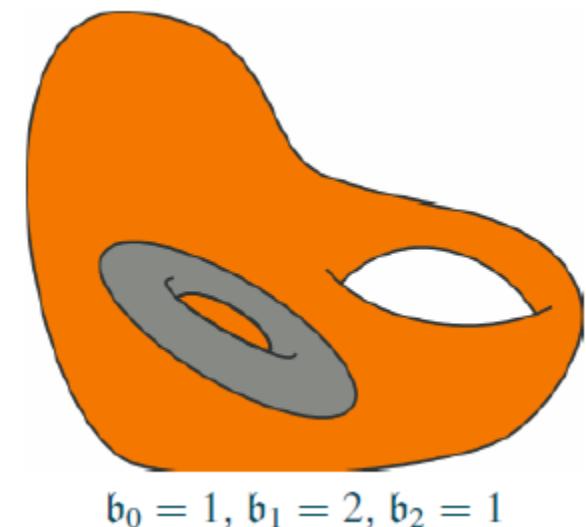
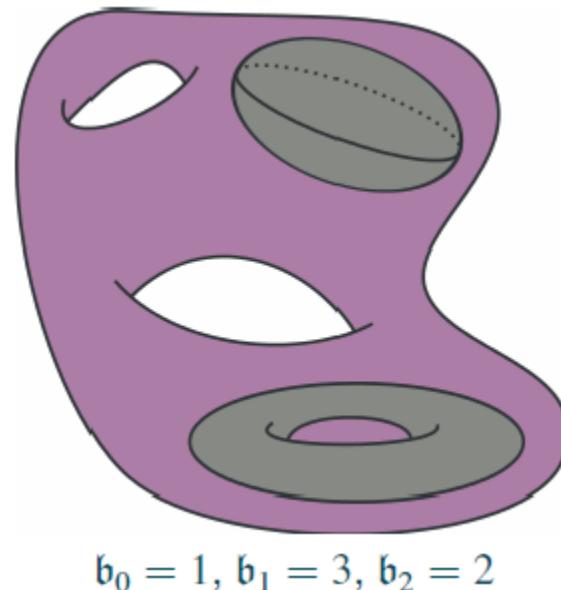
foreach region $R_1 \in T$ labeled 1, $R_1 \neq R_0$ **do**
 foreach region R_2 adjacent to R_1 , $R_2 \neq R_0$ **do**
 if $criterion(R_1, R_2)$ **then**
 $R \leftarrow \text{merge}(R_1, R_2)$;
 update region R ;

return the partition described by T ;

Criteria for merging

- 1. Based on the size of region ($\text{size}(R_1) < \tau$)
- 2. Based on a topological invariant: Betti numbers
- Intuitively: b_i number of the connected components of the i -surfaces
 - b_0 is the number of connected components

- b_1 is the number of tunnels
- b_2 is the number of voids



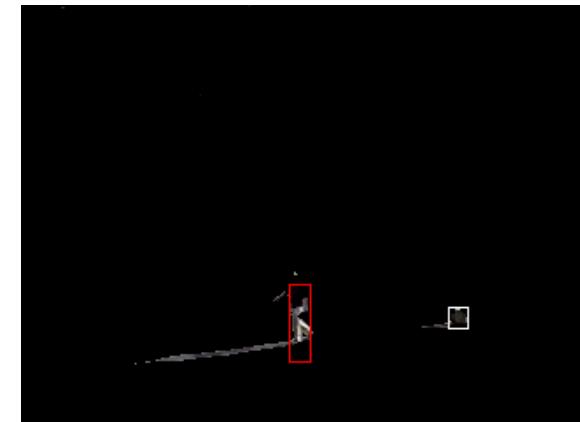
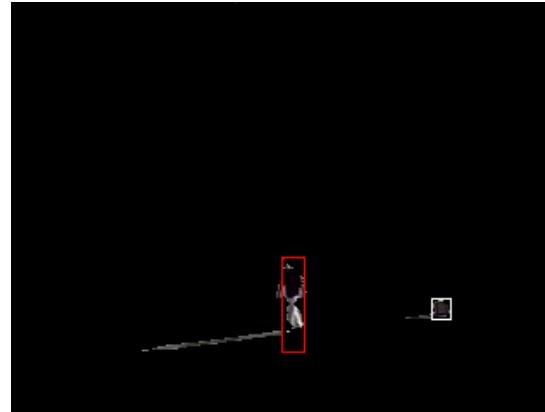
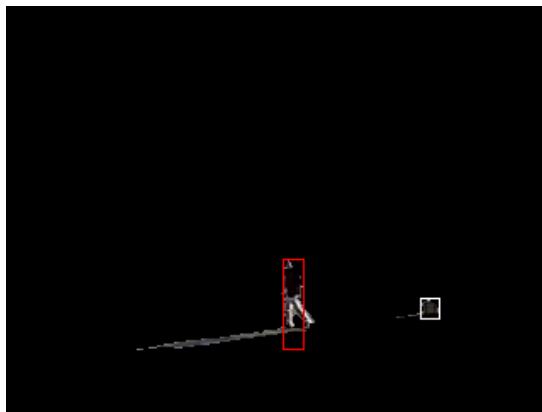
Merging criterion based on betti numbers

$$\text{size}(R_1) < \tau * (1 + \varphi * (\mathfrak{b}_1(R_1) + \mathfrak{b}_2(R_1)))$$

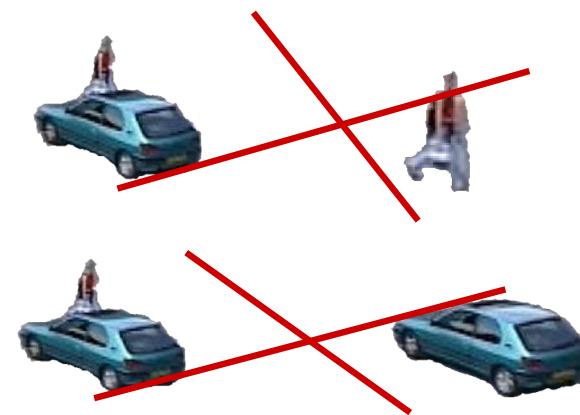
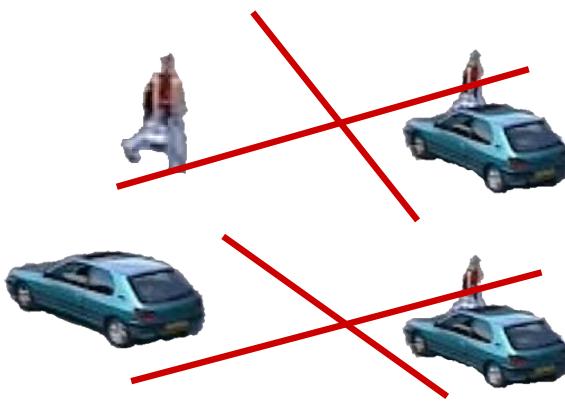
	v1 <i>Fsc</i>	v3 <i>Fsc</i>	v4 <i>Fsc</i>	v5 <i>Fsc</i>	v6 <i>Fsc</i>	v7 <i>Fsc</i>	v8 <i>Fsc</i>
<i>A1</i>	0.15	0.31	0.33	0.25	0.44	0.20	0.23
<i>A2</i>	0.35	0.26	0.40	0.37	0.47	0.37	0.46
<i>A3</i>	0.30	0.36	0.40	0.28	0.45	0.28	0.35
<i>t2000</i>	0.29	0.40	0.45	0.35	0.48	0.32	0.36
<i>t3000</i>	0.30	0.40	0.45	0.37	0.48	0.32	0.37
<i>t4000</i>	0.30	0.39	0.46	0.37	0.48	0.32	0.37
<i>t2000-p.05</i>	0.33	0.39	0.45	0.41	0.49	0.41	0.40
<i>t2000-p.1</i>	0.35	0.30	0.34	0.42	0.49	0.41	0.41
<i>t2000-p.15</i>	0.37	0.13	0.14	0.39	0.43	0.37	0.38
<i>t3000-p.05</i>	0.34	0.32	0.44	0.43	0.49	0.41	0.41
<i>t3000-p.1</i>	0.37	0.12	0.13	0.39	0.42	0.36	0.38
<i>t3000-p.15</i>	0.28	0.05	0.05	0.31	0.36	0.30	0.31
<i>t4000-p.05</i>	0.36	0.25	0.30	0.41	0.49	0.41	0.41
<i>t4000-p.1</i>	0.31	0.05	0.05	0.34	0.38	0.31	0.34
<i>t4000-p.15</i>	0.20	0.02	0.01	0.21	0.24	0.25	0.26

Object tracking

- The goal is to determine the trajectories of detected objects
- Most of the techniques are based on the principle of temporal continuity of the object appearance



Occlusion problem



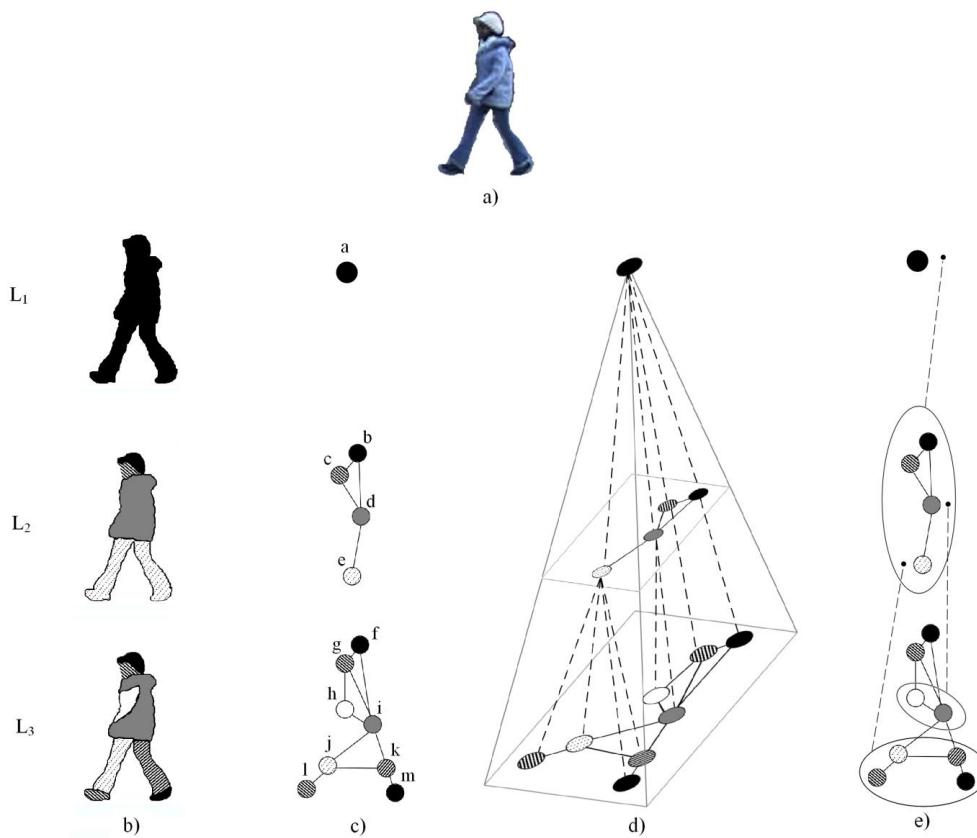
Ideal solution

1



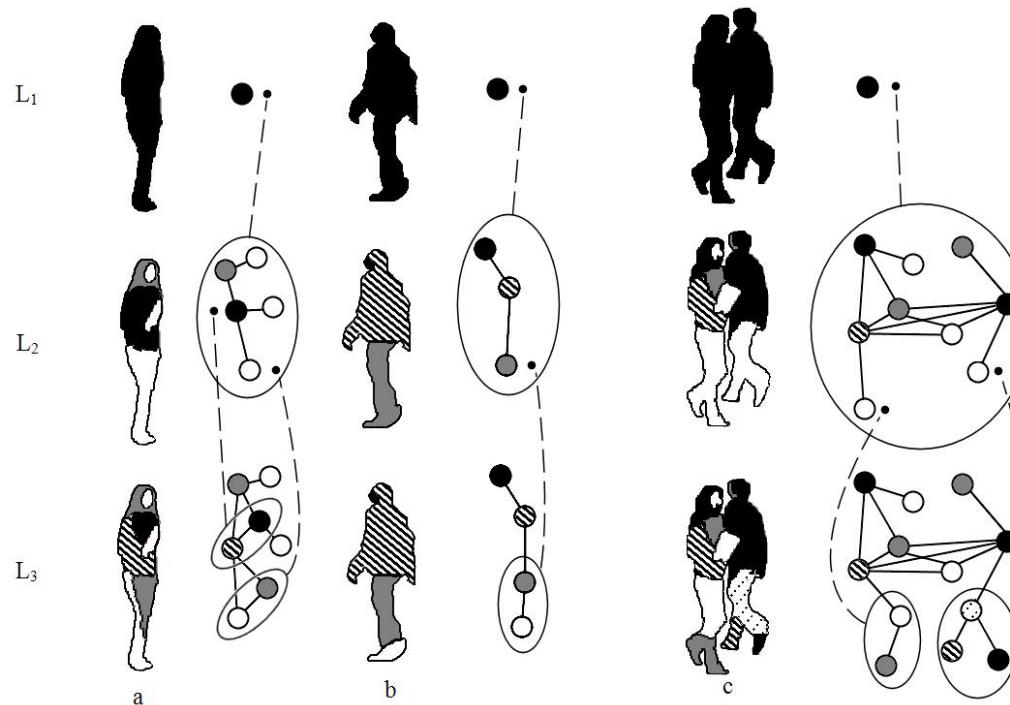
¹Image taken from Yeh et al. ICIP 2010

A structural representation

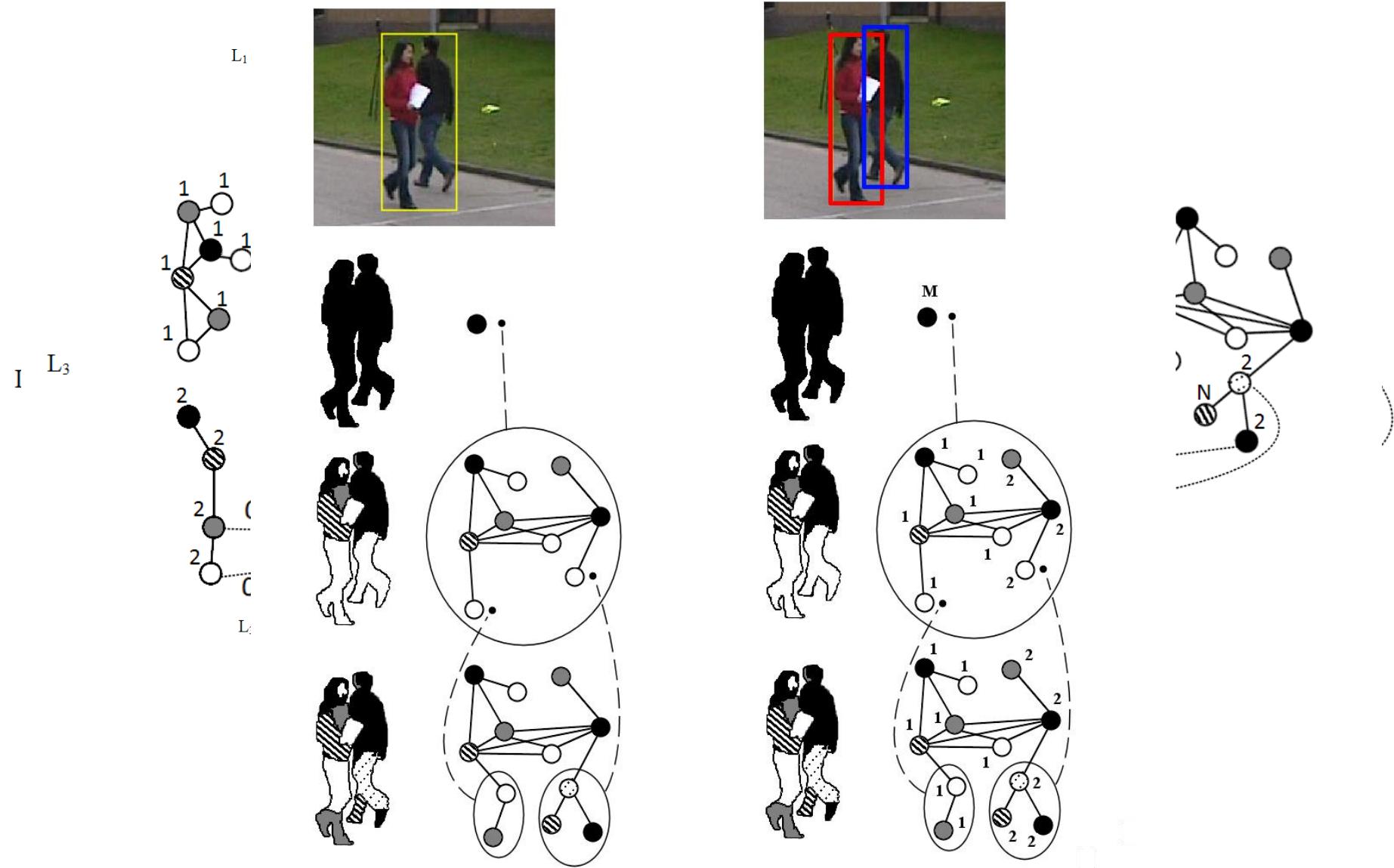


Node ID	c_x	c_y	w	h	μ_R	μ_G	μ_B
a	219	267	217	374	190	196	210
b	223	123	72	54	180	191	208
c	201	147	54	42	56	58	68
d	223	235	118	162	99	121	163
e	212	393	217	160	60	72	105
f	223	123	72	54	180	191	208
g	201	147	54	42	56	58	68
h	205	227	60	86	75	95	138
i	223	235	118	162	105	127	171
j	164	380	91	148	57	70	106
k	271	383	91	136	55	68	103
l	126	460	56	31	46	54	78
m	294	455	62	43	56	70	97

The algorithm by example



The algorithm by example



State of the art

- Large literature: many methods, but still an open problem
- Some examples:
 - KSP: Multiple Object Tracker Using K-Shortest Paths
 - Continuous Energy Minimization for Multi-Target Tracking
 - MonteCarlo methods and Particle Filters

A filtering system

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_t)$$

$$\mathbf{y}_t = h_t(\mathbf{x}_t, \mathbf{w}_t)$$

where \mathbf{x}_t : the hidden state at time t ;

\mathbf{y}_t : the measurement state;

f_t : the temporal evolution of \mathbf{x}_t ;

h_t : the measurement equation;

\mathbf{v}_t and \mathbf{w}_t : independent white noises.

Goal

Estimating the posterior density function

$$p(\mathbf{x}_t | \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t | \mathbf{x}_t) \cdot p(\mathbf{x}_t | \mathbf{y}_{1:t-1})}{\int_{\mathcal{X}} p(\mathbf{y}_t | \mathbf{x}'_t) \cdot p(\mathbf{x}'_t | \mathbf{y}_{1:t-1}) d\mathbf{x}'_t}$$

A solution: particles filters

- Approximating the density function by a weighted sum of N Dirac masses $\delta_{\mathbf{x}_t^{(n)}}(d\mathbf{x}_t)$ centered on hypothetic state realizations $\{\mathbf{x}_t^{(n)}\}_{n=1}^N$ of the state \mathbf{x}_t , also called particles
- $\mathbb{P}(\mathbf{x}_t | \mathbf{y}_{1:t})$ is recursively approximated by the empiric distribution

$$P_N(\mathbf{x}_t | \mathbf{y}_{1:t}) = \sum_{n=1}^N w_t^{(n)} \delta_{\mathbf{x}_t^{(n)}}$$

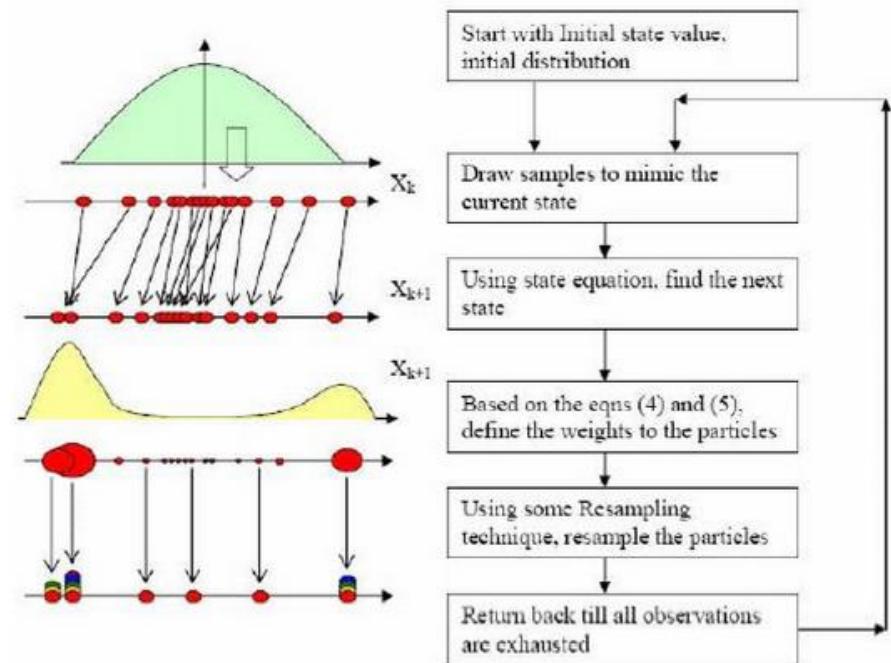
where $\mathbf{x}_t^{(n)}$ is the n^{th} particle and $w_t^{(n)}$ its weight

Particles Filters processing

- ① **Diffusion step:** propagating the particle swarm $\{\mathbf{x}_t^{(n)}, w_t^{(n)}\}_{n=1}^N$ using a probabilistic model for the state function;
- ② **Update step:** computes new particle weights using the new observation \mathbf{y}_t ;
- ③ **Resampling step:** generating new particles according to their weight.
- ④ **Output:**

- $\hat{\mathbf{x}}_t^{MMSE} = \sum_{n=1}^N w_t^{(n)} \mathbf{x}_t^{(n)}$

- $\hat{\mathbf{x}}_t^{MAP} = \operatorname{argmax}_{\mathbf{x}_t} \sum_{n=1}^N w_t^{(n)} \delta_{\mathbf{x}_t^{(n)}}$



Object Tracking and Particles Filters

- Widely used in the mono-object tracking, because of its reliability to deal with non-linear systems
 - usually the state of the system is the coordinates of the object to track

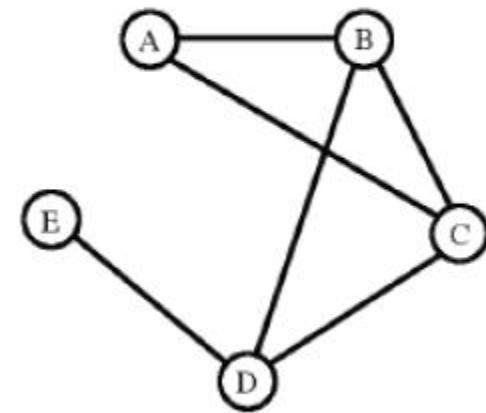
Object Tracking and Particles Filters

- Problems in multi-objects tracking
 - estimating more objects needs substantially more particles (curse of dimensionality: the number of particles increases exponentially with the number of objects)
 - the association problem between measures and objects has to be solved
 - interactions between objects should be modeled
- The problem is still open

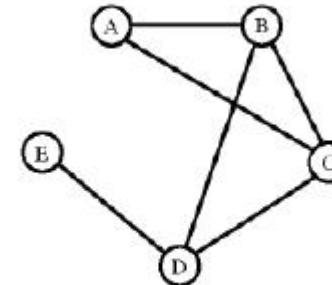
Graph-based particles filter

- Using graphs for representing particles
- One graph = One particle = The entire ensemble of objects in the scene
- Using graph kernels for measuring likelihood

The Representation

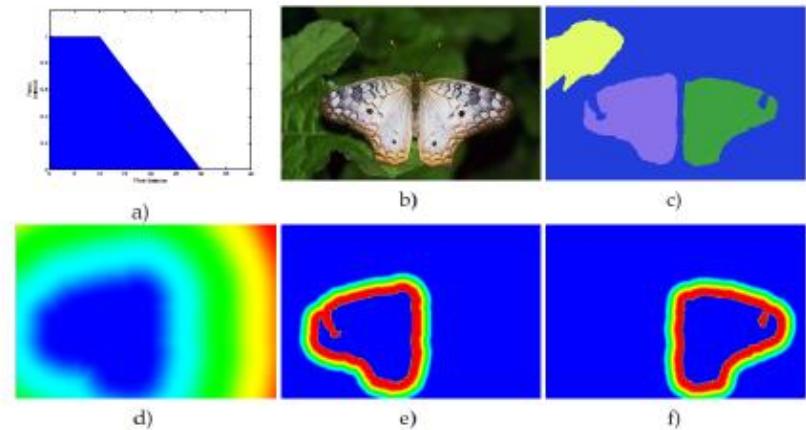
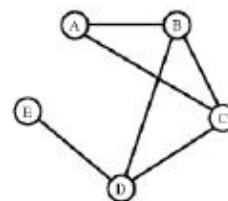


The Representation



Node attributes												
ID	label	visibility	w	h	c_x	c_y	$\dot{w}(\frac{px}{fr})$	$\dot{h}(\frac{px}{fr})$	$\dot{c}_x(\frac{px}{fr})$	$\dot{c}_y(\frac{px}{fr})$	$h_o(A)$	
A	1	1	41	104	94	207	3	0	15	0		
B	2	1	57	150	151	220	4	1	12	4		
C	3	1	94	197	235	233	4	2	8	8		
D	4	1	40	81	326	212	1	1	0	10		
E	5	1	105	253	505	308	0	0	0	0		

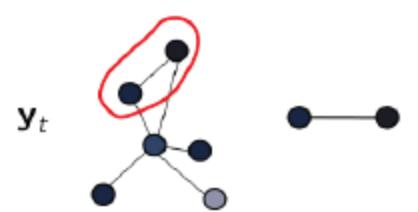
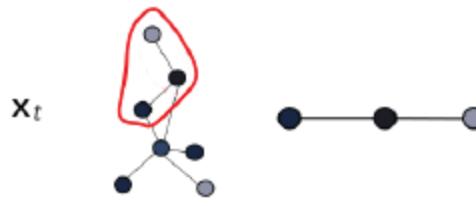
The Representation



<i>Edge attributes</i>	
ID	fuzzy adjacence measure adj (with $\tau = 0.1$)
e(A,B)	0.62
e(A,C)	0.43
e(B,C)	0.58
e(B,D)	0.32
e(C,D)	0.27
e(D,E)	0.12

$adj(near(R_1), R_2) = \frac{\sum_{x \in R_2} \mu_{near(R_1)}(x)}{\sum_{x \in S} \mu_{R_2}(x)}$

The likelihood: an Edition Path Kernel



$$p(y_t|x_t) \simeq K_{edit}(x_t, y_t)$$

Graph Kernel based on editions costs



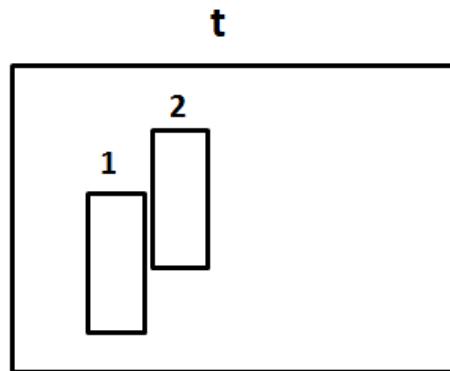
Edition paths: $(h, \kappa(h), \dots, \kappa^D(h))$

$$K_{edit}(h_1, h_2) = \frac{1}{D^2} \sum_{k=0}^D \sum_{l=0}^D e^{-\frac{cost_k(h_1) + cost_l(h_2)}{2\sigma_{cost}^2}} K_{classic}(\kappa^k(h_1), \kappa^l(h_2))$$

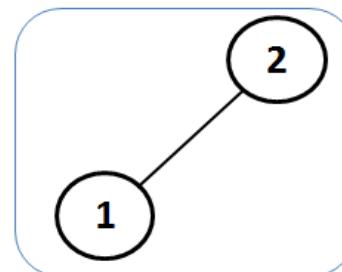
Graph-based Particles Filter

- **Diffusion step:** propagating the particle swarm $\left\{ \mathbf{x}_t^{(n)}, w_t^{(n)} \right\}_{n=1}^N$ using a probability model for the hidden state;
- **Updating step:** computing new particle weights using the new observation \mathbf{y}_t ;
- **Resampling step:** generating new particles based on old particles weights.
- **Output:**
 - $\hat{\mathbf{x}}_t^{MMSE} = \sum_{n=1}^N w_t^{(n)} \mathbf{x}_t^{(n)}$
 - $\hat{\mathbf{x}}_t^{MAP} = \operatorname{argmax}_{\mathbf{x}_t} \sum_{n=1}^N w_t^{(n)} \delta_{\mathbf{x}_t^{(n)}}$
- **Diffusion step:** Graph Edition (adding a node, changing an attribute, ...) according to a probabilistic model;
- **Updating step:** at time t the observation of the scene is also represented by a graph \Rightarrow the weights of “particle graphs” are computed by means of Graph Kernels;
- **Output:** MAP: the graph with the highest weight is taken as state representation at time t

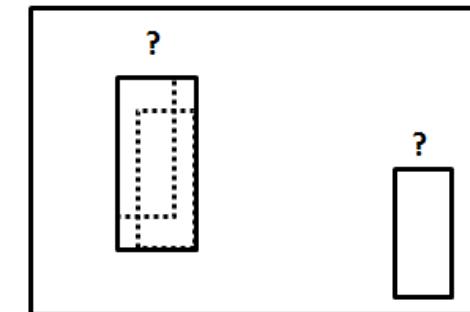
A toy example



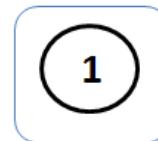
$p = 0.2$



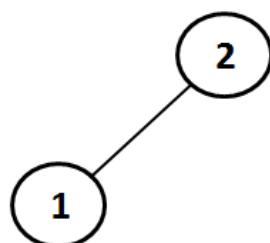
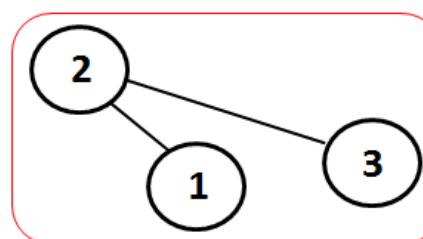
$t+1$



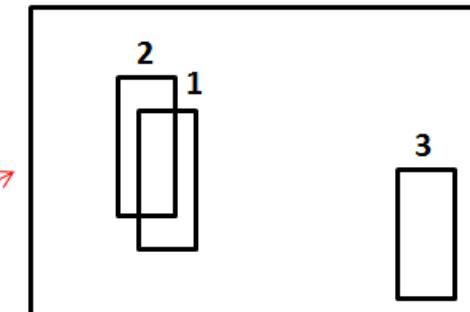
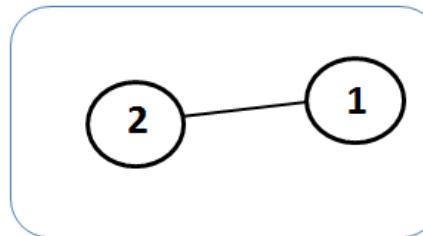
$p = 0.1$



$p = 0.5$



$p = 0.2$



Tracking algorithm evaluation

- Need of public, standard databases
 - e.g. : PETS, CAVIAR, ETHZ
- Need of standard evaluation indexes
 - e.g : Kasturi indexes

PETS 2010 dataset



Kasturi indexes

- R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, “Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 319–336, 2009.
- It is a reference for tracking algorithm evaluation

Kasturi indexes

Object Detection

$$FDA(t) = \frac{\text{Overlap_Ratio}}{\left[\frac{N_G^{(t)} + N_D^{(t)}}{2} \right]},$$

$$\text{Overlap_Ratio} = \sum_{i=1}^{N_{\text{mapped}}^{(t)}} \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|}.$$

$$SFDA = \frac{\sum_{t=1}^{t=N_{\text{frames}}} FDA(t)}{\sum_{t=1}^{t=N_{\text{frames}}} \exists(N_G^{(t)} \text{ OR } N_D^{(t)})}.$$

$$MODA(t) = 1 - \frac{c_m(m_t) + c_f(fp_t)}{N_G^{(t)}},$$

Kasturi indexes

Object Tracking

$$STDA = \sum_{i=1}^{N_{mapped}} \frac{\sum_{t=1}^{N_{frames}} \left[\frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|} \right]}{N_{(G_i \cup D_i \neq \emptyset)}}.$$

$$ATA = \frac{STDA}{\left[\frac{N_G + N_D}{2} \right]}.$$

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}^{(t)}} \left[\frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|} \right]}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}},$$

Conclusions

- A huge number of applications are based on video analysis
- Real-time processing, privacy laws, etc. impose severe constraints to algorithms
- Many problems are still open