University of Trento

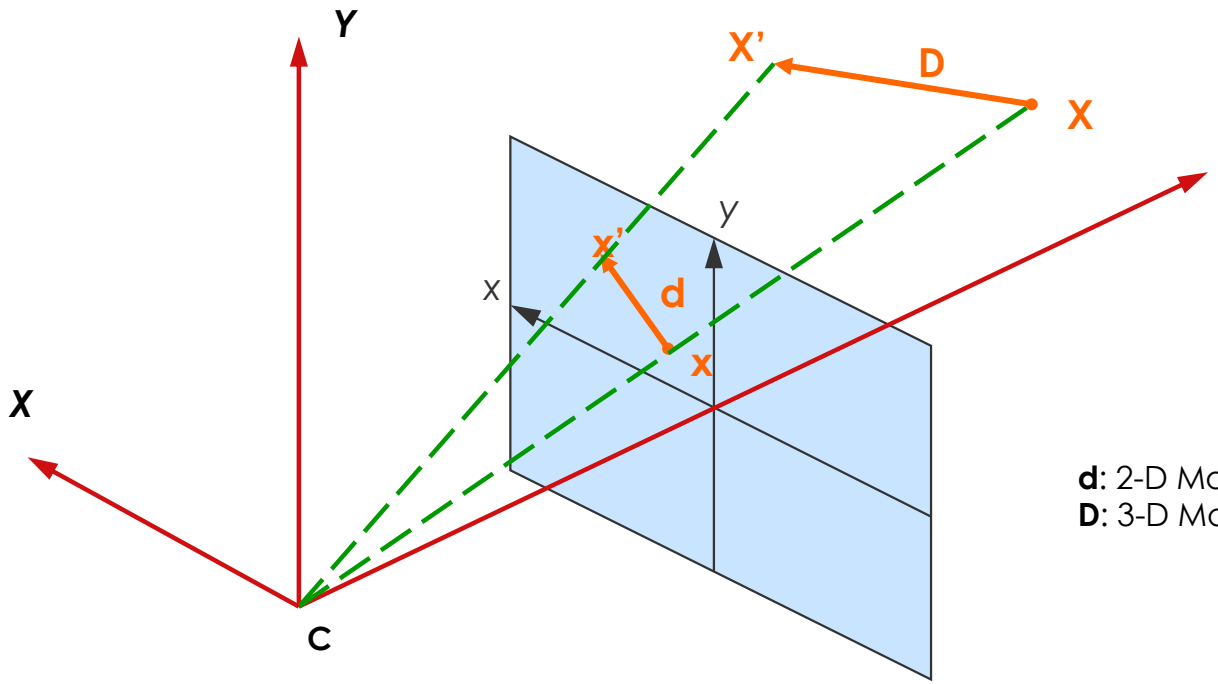
Part III: Motion Detection

Nicola Conci
nicola.conci@unitn.it

Motion

- In video analysis and processing motion is a fundamental feature
- We'd like to have an accurate description of
  - Objects motion in the scene
  - Camera motion
- Projection of 3D motion onto the 2D image plane:
  - 3D: $\mathbf{D}(\mathbf{X};t_1;t_2) = \mathbf{X}' - \mathbf{X} = [Dx,Dy,Dz]$
  - 2D: $\mathbf{d}(\mathbf{x};t_1;t_2) = \mathbf{x}' - \mathbf{x} = [dx,dy]$

2D Motion of a 3D Object

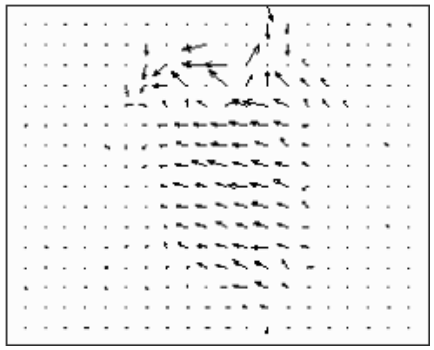


d: 2-D Motion Vector
D: 3-D Motion Vector

An example

Frame 1

Frame 2



Motion Field

- Questions:
  - How can I extrapolate the motion field?
  - Does it really reflect the object displacement?
  - Is the motion due to the camera or to the object?

2D motion of a rigid object

- The camera is supposed to be fixed
- Objects move in the 3D space
  - Motion is generated by a complex combination of arbitrary movements
  - The 2D projection we obtain is ambiguous (combinations of different movements can generate the same image)
  - Through an a posteriori analysis it's very difficult to determine the different types of movements that affect the scene

Motion Detection

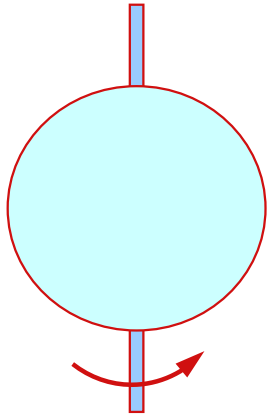
- (Some) Applications:
  - Detection
  - Segmentation
  - Recognition
  - Filtering (deblurring, noise suppression)
  - Compression (transmission and storage)
- *In general*: detect changes over time of the image intensity
- *Ill-posed problem*, the solution is not unique

Problem

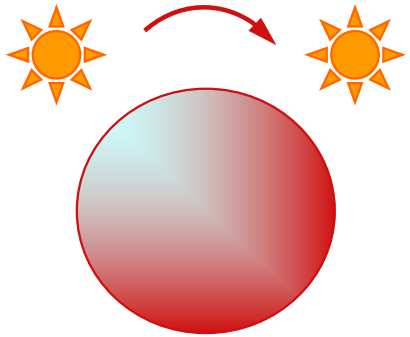
- It is based on the velocity $v(x,y,t)$ of the pixels over two consecutive frames
- **Does not reflect real motion**
- Can be caused by:
  - Object motion
  - Camera motion
  - Illumination changes

Real vs Apparent movement

- 2D motion perceived by changes in the scene
- No intensity change, no motion



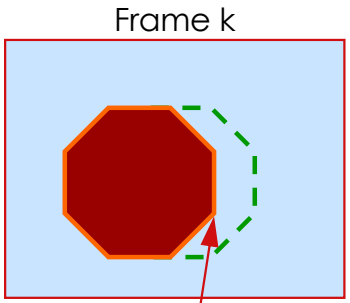
Constant illumination:
Sphere rotates, no change



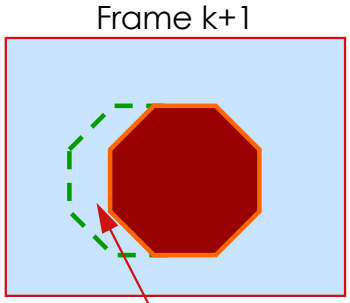
Illumination source rotates around the sphere:
It looks like the sphere rotates!

Occlusions

- The surface is covered/uncovered by the movement of an object



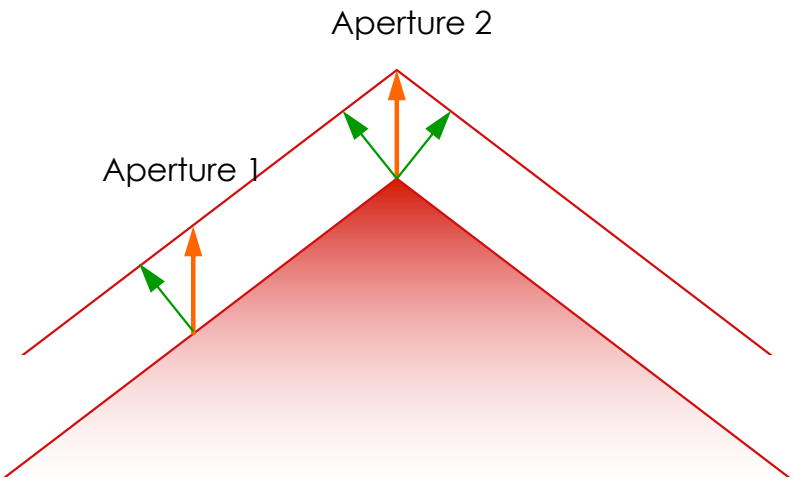
- Portion of the region that will be covered by the foreground object
- In the next frame there won't be such a region



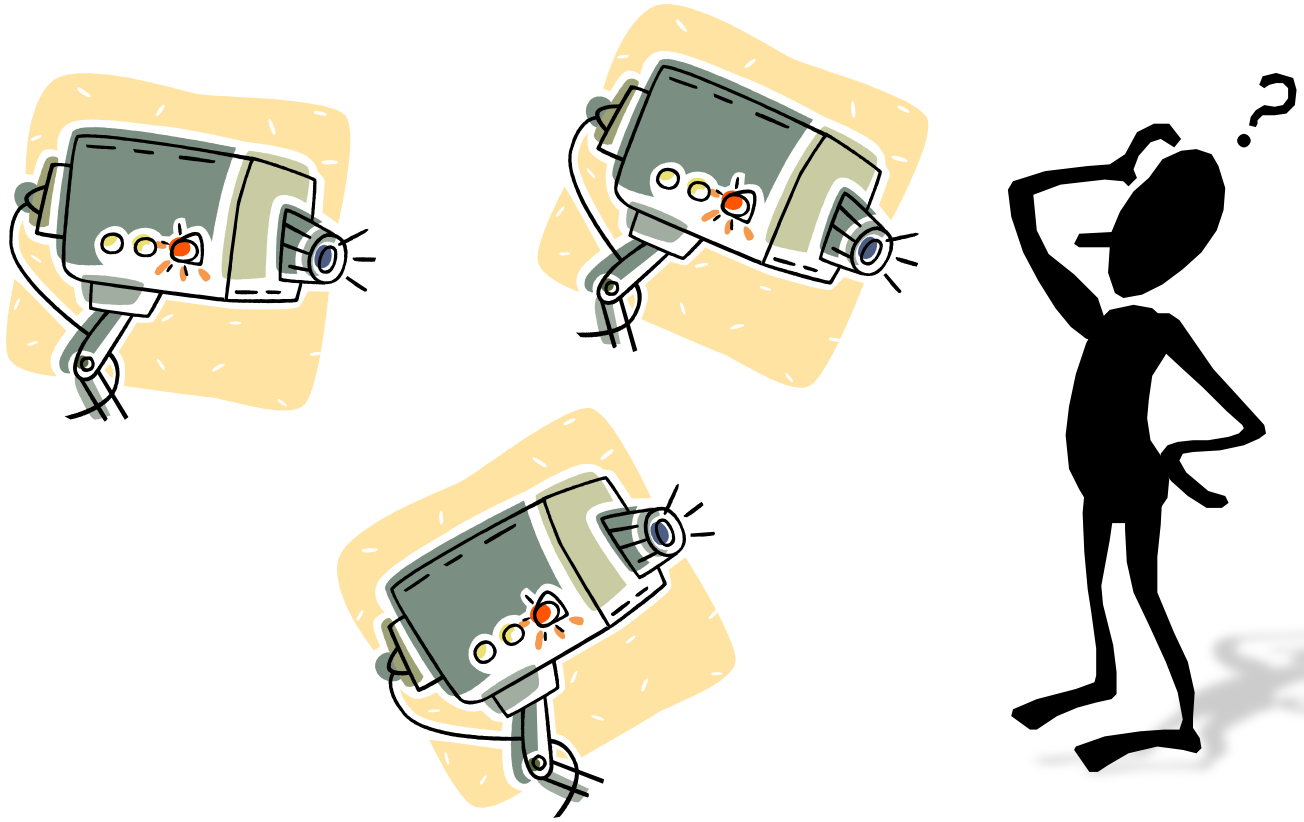
- Region uncovered
- There did not exist such a region before

Aperture

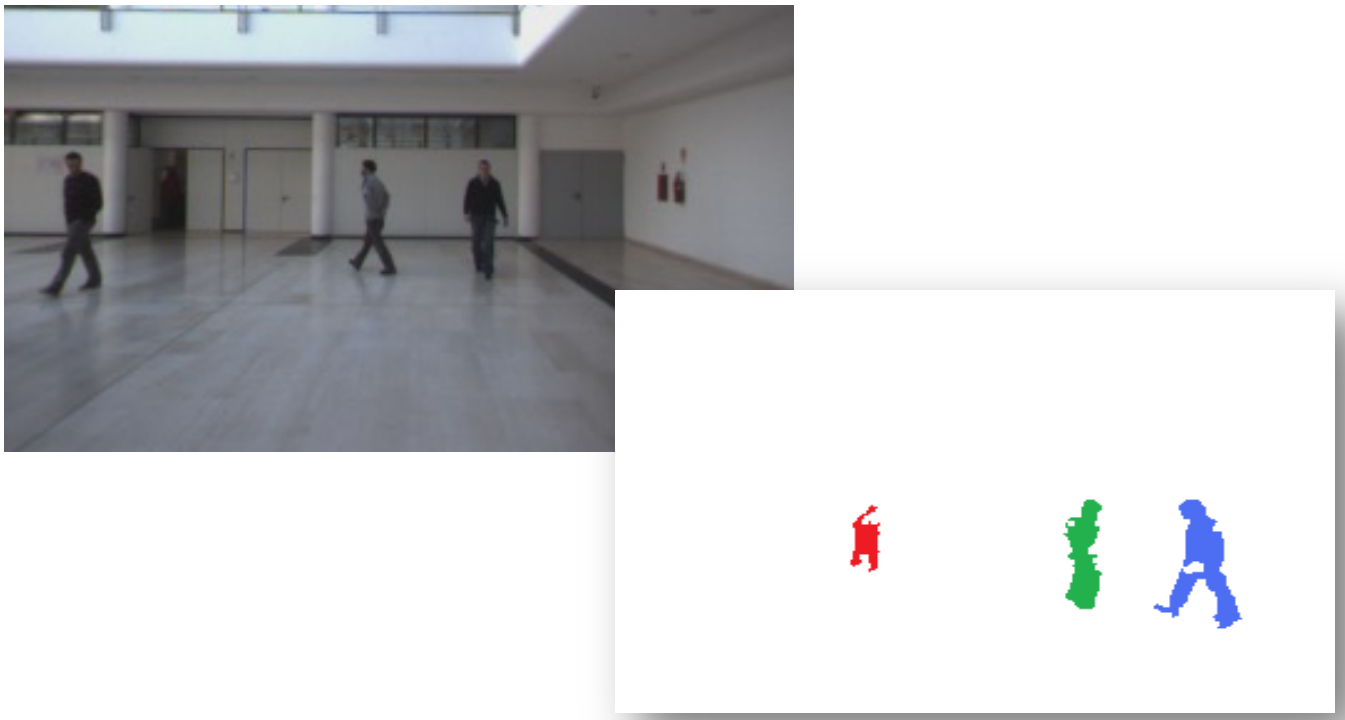
- Only displacements of the borders can be detected
- **Plus**, only in the direction of the intensity gradient



Things look complicated...



What we want to obtain



Optical flow/Image flow

- Assumptions:
  - Object illumination does not change in $[t, t+dt]$
  - Distances do not change significantly
  - Each point $[x, y]$ is shifted in $[x+dx, y+dy]$
- Not true in real video, but good approximation!

Optical Flow Equation

- Hypothesis: object points keep the same intensity

$$\psi(x+d_x, y+d_y, t+d_t) = \psi(x,y,t)$$

- The Taylor expansion states that: $f(x+\Delta x) = f(x) + f'(x)\Delta x$
- For small *dx, dy, dt,* the Taylor series representation becomes:

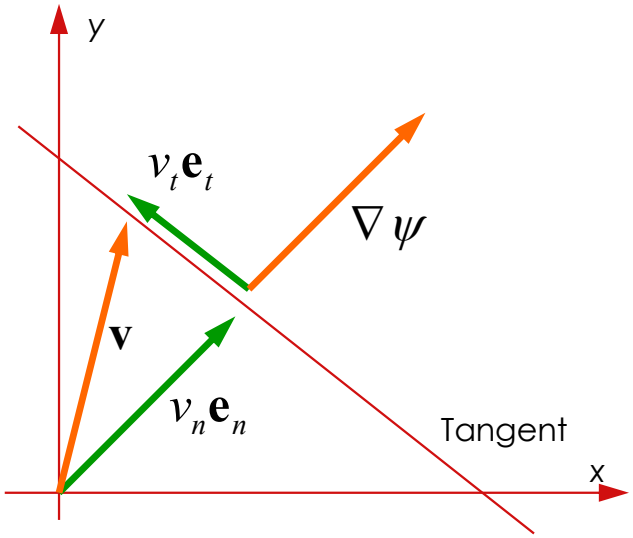
$$\psi(x+d_x, y+d_y, t+d_t) = \psi(x,y,t) + \frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t$$

- Comparing the two:
- Dividing by d_t, we get:

$$\frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t = 0$$

$$\frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} = 0 \qquad \text{or} \quad \nabla \psi^T \mathbf{v} + \frac{\partial \psi}{\partial t} = 0$$

Comments



$$v_n\|\nabla\psi\|+\frac{\partial\psi}{\partial t}=0$$

- Decomposing **v** in the orthogonal components:

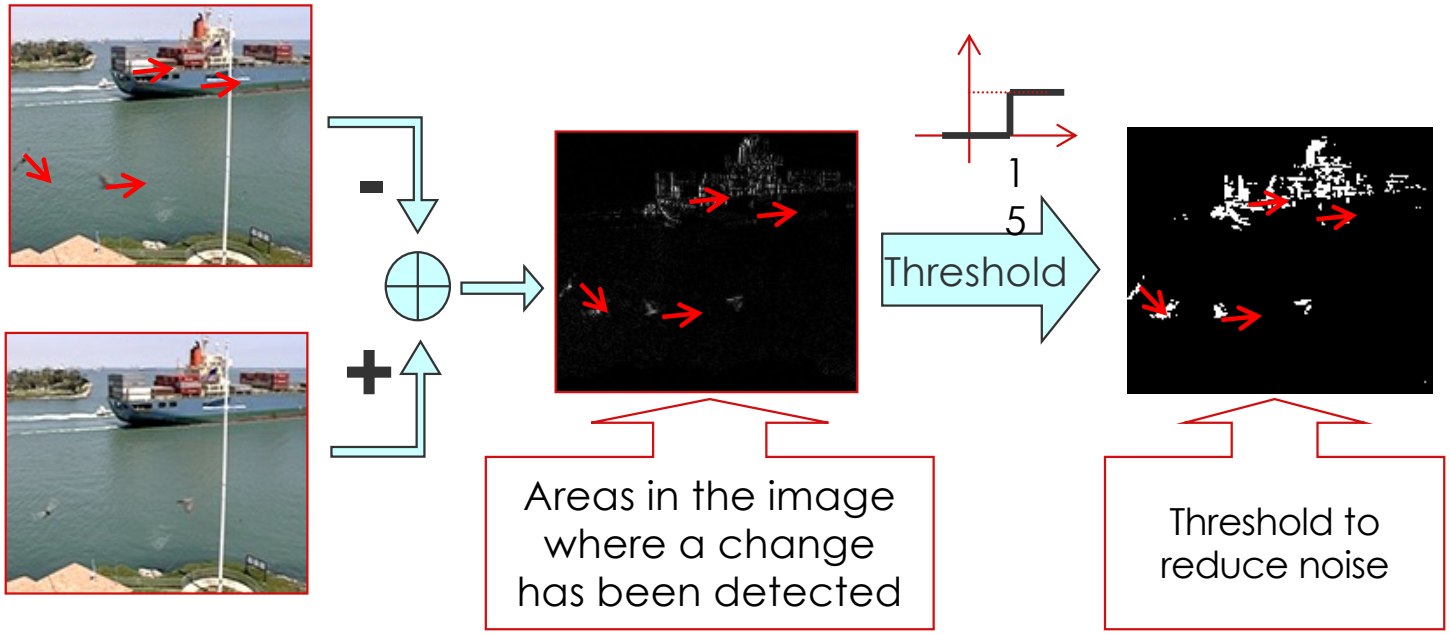
$$\mathbf{v}=v_n\mathbf{e}_n+v_t\mathbf{e}_t$$

- v_t is undetermined: every value satisfies the Optical Flow (aperture problem)
- As an additional constraint, also intensity changes slowly: to determine the variations we consider movements in a neighborhood
- If the area has a constant intensity, motion is undetermined

Motion detection in practice

- Two possibilities
  - Intensity based
  - Feature based
- Problems
  - How to represent the motion field?
  - What do I use as discriminant parameter?

Change Detection



Change Detection

- Can work if illumination is (almost) constant
- Frame difference corresponds to: I_k-I_{k-1}

$$FD_{k,k-1}(x_1,x_2) = s(x_1,x_2,k) - s(x_1,x_2,k-1)$$

- If FD is non-null, a change has occured
- Change can be due to noise. A threshold is applied to control it:

$$z_{k,k-1}(x_1,x_2) = \begin{cases} 1 & \text{if} \quad FD_{k,k-1}(x_1,x_2) > T \\ 0 & \text{otherwise} \end{cases}$$

- A good strategy to avoid noise is to use the so-called cumulative difference

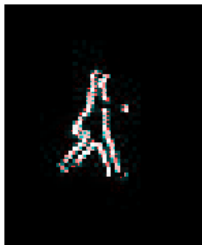
Frame Differencing

- Replace background at each frame
- Example:

```
B(0) = I(0);
...
loop time t
        I(t) = next frame;
        diff = abs[B(t-1) – I(t)];
        M(t) = threshold(diff);
        ...
        B(t) = I(t);
end
```

Frame Differencing: comments

- Quickly adapts to changes in the background.
- Once an objects stops, it is not detected anymore
- Can detect contours of objects
- Only few pixels are usually detected



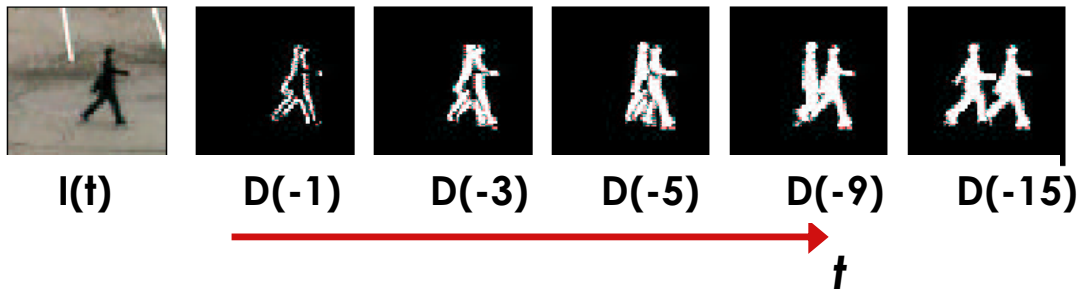
(a)

(b)

Frame Differencing: comments

- By changing the time scale, results can be significantly different

$$D(N) = \|I(t) - I(t+N)\|$$



The contour of the object becomes clearer, until it doubles: departure and arrival points.

Background subtraction

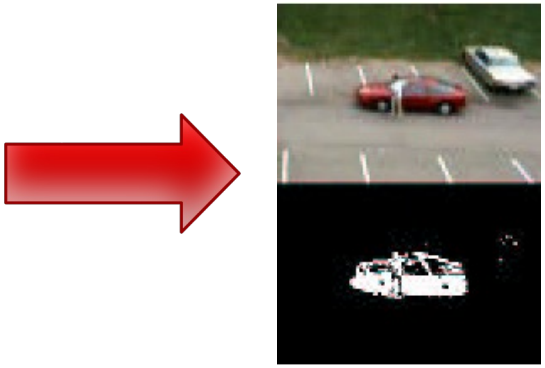
- Background is modeled as a static image (in general acquired without moving objects).
- Calculate difference between current frame and the background and assign a label:
  - (1) if foreground
  - (0) if background
- Example:

```
B = I(0);
…
loop time t
     I(t) = next frame;
     diff = abs[B – I(t)];
     Map(t) =
threshold(diff);
                …
end
```

Background subtraction: comments

- Good results if the color of objects differs from the background.
- If an object enters the scene, it will always be regarded as foreground
- If another object enters in the foreground areas, it won't be detected (a).
- If a background object moves, both the real object and its "ghost" are detected (b).

The background is acquired when the parking lot is empty



(a)

(b)

Background subtraction: comments

- Sensitive to illumination and background changes (trees, reflections on glossy surfaces like cars and water) (a)

- Not robust to camera motion! (b)

(a)

(b)

Adaptive background subtraction

- Use a parameter α to weight the contributions
- α is called the *learning rate*

- $B_t = \alpha I_t + (1-\alpha) B_{t-1}$;
  - $\alpha = 0$ → bg sub, no update
  - $\alpha = 1$ → frame differencing

Gaussian average

- May use a training sequence (possibly empty scene)
- Take:
  - Current pixel value
  - Previous average and standard deviation
- Similar to adaptive bg subtraction, but more stable

$$\mu_t = \alpha I_t + (1-\alpha)\mu_{t-1}$$

M. Piccardi - Background subtraction techniques: a review, IEEE Int. Conf. on System, Man and Cybernetics, 2004

Gaussian average

- Fast to compute
- Low memory requirements
- Pixel goes to foreground if:

$$\left|I_t - \mu_t\right| > k\sigma_t$$

- Quality depends on a
- a depends on the application
- Slow update = slow response to changes

Improving the Gaussian average

- Use a binary operator to update values selectively
- Update only background values
- M=1 if the pixel is foreground
- M=0 if the pixel is background

$$\mu_t = M\mu_{t-1} + (1-M)(\alpha I_t + (1-\alpha)\mu_{t-1})$$

Mixture of Gaussians

- With most techniques new objects are sooner or later absorbed in the background
- There are changes that appear and disappear faster than the update rate
- Example:
  - Tree leaves
  - Rain/snow
  - Water of a fountain
- It can be that the same pixel represents at different time instants:
  - Tree leaves
  - Sky/building behind

C. Stauffer and W.E.L. Grimson - Adaptive background mixture models for real-time tracking, CVPR'99

Mixture of Gaussians

- Idea: use ***K*** Gaussians instead of a single one!

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

- Each of them used to model a foreground or background object
- Theoretical model is complex:
  - Multi-variate gaussians to model RGB, or YUV
  - In practice:
    - Components are independent (matrix is diagonal)
    - Variance of components is the same → collapse to σ^2

Mixture of Gaussians

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

- $\omega_{i,t}$ = weight for the current Gaussian
- Select ***K***
- Rank the Gaussians on the basis of
  - Peak amplitude
  - Weight
  - Standard deviation
- Hint: the higher the peak and the more compact the distribution (low variance), the more the pixel is likely to belong to the background → strong evidence

Mixture of Gaussians

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

- Among the ***K*** Gaussians,
  - Some belong to the background
  - Some belong to the foreground
- The first ***B*** Gaussians are associated to the background if

$$\sum_{i=1}^{B} \omega_i > T$$

Mixture of Gaussians

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

- Each incoming pixel is checked against the available models
- Example: a match is found if the pixel is within $(x_t - \mu_{i,t}) < 2.5\sigma$
- In case no match is found, the less probable Gaussian is replaced with a new one centered in x_t with low weight and high variance

Mixture of Gaussians

- Also here, update of the weights is necessary

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

- α is still the learning rate
- M is one for the matching model and 0 otherwise
  - → if it is not the matching model, the weight is decreased