Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

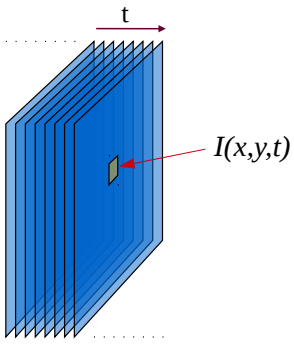
- **Image formation**
- **Low-level vision**
- **Mid-level vision**
  - grouping and segmentation of image elements
  - Biological
    - bottom-up influences (Gestalt cues)
    - top-down influences (knowledge, expectation, etc.)
  - Artificial
    - Thresholding, Region-based, Clustering, Fitting
  - Multi-View Vision
    - Stereo and Depth
    - Video and Motion ← Today
- **High-level vision**

Today

- Optic flow
  - – motion fields
  - – measurement
    - » correspondence problem
    - » aperture problem
  - – applications
    - » depth
    - » time-to-collision
- Tracking
- Segmentation
  - – image differencing
  - – background subtraction

Video

Video is a series of N images, or frames, acquired at discrete time instants $t_k=t_0+k\Delta t$, where Δt is a fixed time interval and k=0,1,…,N-1



In static images

The intensity of a pixel can be seen as a function of its spatial coordinates (x,y):
i.e. an image is $I(x,y)$

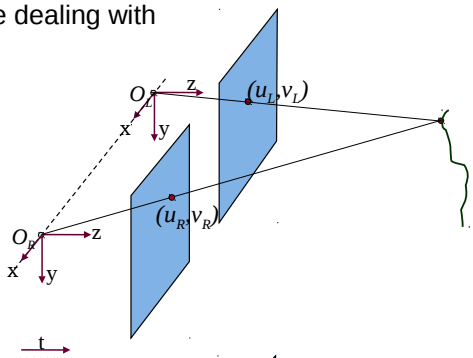
In video

The intensity of a pixel can be seen as a function of its spatial coordinates (x,y) and time (t):
i.e. a video is $I(x,y,t)$

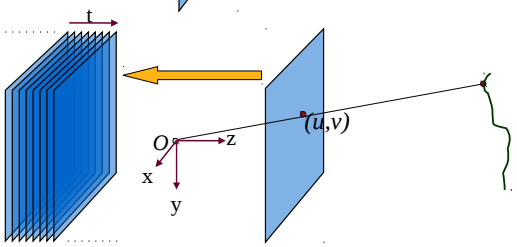
Video and Stereo

Similar to stereo in that we are dealing with more than one image

Stereo
- » multiple cameras
- » one time (images taken simultaneously)



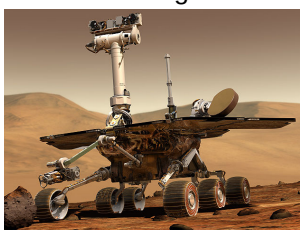
Video
- » one camera
- » multiple times (images taken at different times)



Video

Enables:
- inference of 3D structure (as with stereo).
- segmentation of objects from background without recovery of depth (unlike stereo)
- inference of self and object motion (unlike stereo)
  - ➔ essential for some applications, e.g.:

robot navigation



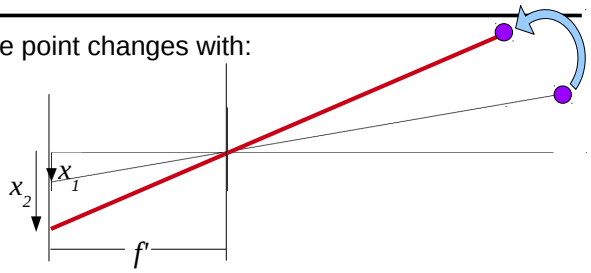
driver assistance

surveillance

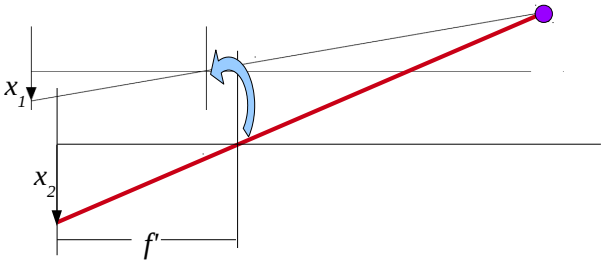
Motion Analysis

Projection of a scene point changes with:

1. object motion

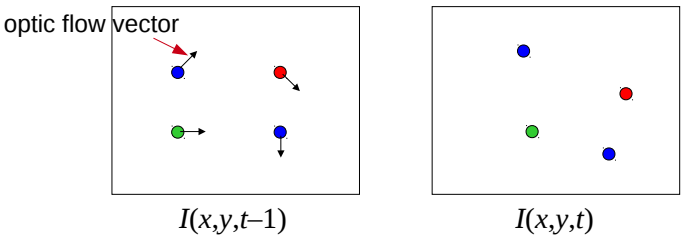


2. camera motion, or “ego motion”



Both types of movement give rise to “optic flow”.

Optic flow (OF)



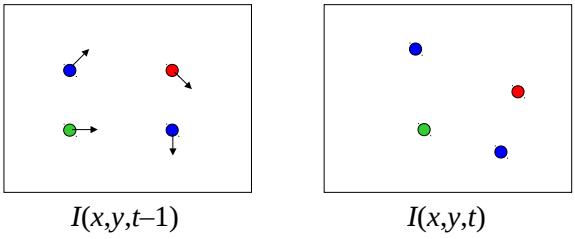
optic flow vector
the image motion of a scene point.

optic flow field
the collection of all the optic flow vectors
Optic flow fields can be sparse (vectors defined only for specified features) or dense (vectors defined everywhere).

optic flow vectors are analogous to disparity vectors in stereo vision

measuring optic flow requires finding correspondences between images

Motion field (MF)

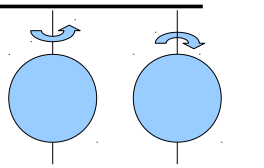


motion field
the true image motion of a scene point
i.e. the actual projection of the relative motion between the camera and the 3D scene

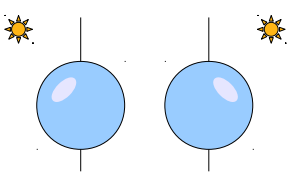
Optic flow provides an approximation to the motion field.
But it is not always accurate...

Motion field (MF) ≠ Optic flow (OF)

Consider a smooth, lambertian, uniform sphere rotating around a diameter:
MF ≠ 0 as points on the sphere are moving
OF = 0 as there are no changes in the images



Consider a stationary, specular, sphere and a moving light source:
MF = 0 as points on the sphere are not moving
OF ≠ 0 as there is a moving pattern in the images



Consider a barber's pole:
MF = horizontal
OF = vertical

Despite MF ≠ OF in all circumstances, MF cannot be observed, so we must estimate MF by observing OF.

The video correspondence problem

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

To solve the video correspondence problem, we can use:

- **Feature-based methods**

  Extract descriptors from around interest points find similar features in next frame (identical to method used for stereo correspondence)

  Sparse optic flow fields

  Suitable when image motion is large

- **Direct methods**

  Directly recover image motion at each pixel from temporal variations of the image brightness (by applying spatio-temporal filters)

  Dense optic flow fields, but sensitive to appearance variations

  Suitable when image motion is small

The video correspondence problem

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

To use feature-based methods

Basic requirements to be able to solve the correspondence problem:

1. Most scene points visible in both images
2. Corresponding image regions appear “similar”

Video Constraints on Correspondence

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

Constraints used to help find corresponding points:

Spatial coherence

Similar neighbouring flow vectors are preferred over dissimilar ones.

- The assumption is that the scene is made up of smooth surfaces, and hence, neighbouring points in the scene typically belong to the same surface, and hence, typically have similar motions and induce similar optic flow.

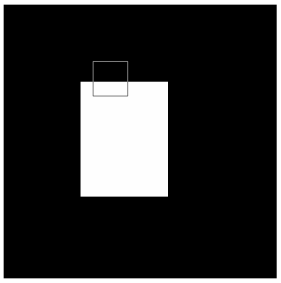
Small motion

Small optic flow vectors are preferred over large ones.

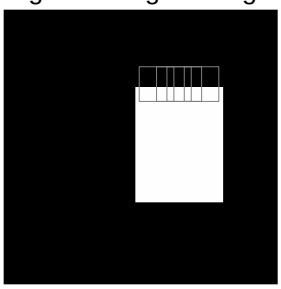
- The assumption is that relative velocities are slow compared to the frame rate, and hence, that the amount of motion between frames is small compared to the size of the image.

Aperture problem

Consider two consecutive frames showing a moving rectangle.



I(x,y,t)



I(x,y,t+1)

The image patch marked in the first frame could match any of the patches marked in the second frame.

Because the intensity varies across the edge but not along the edge a motion parallel to the edge can never be recovered.

The inability to determine optic flow along the direction of the brightness pattern is known as the "**aperture problem**".

Aperture problem

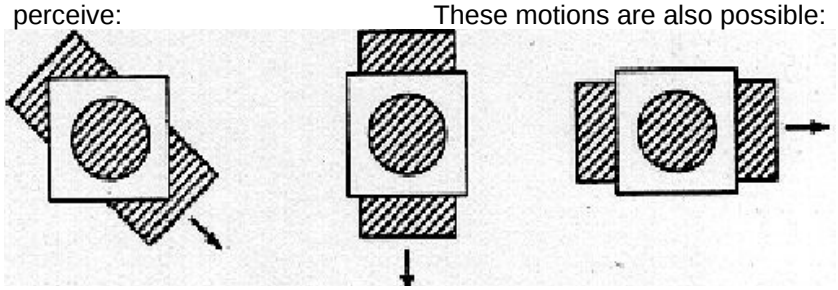
The brain is also faced with the aperture problem as each motion sensitive neuron sees only a small spatial region (i.e. its receptive field)

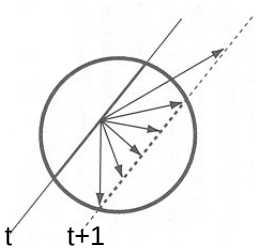
A demonstration.
What is the direction of motion here?

Aperture problem

This is what we perceive:

These motions are also possible:





Any movement with a component perpendicular to the edge is possible.

Locally the direction of motion is ambiguous.

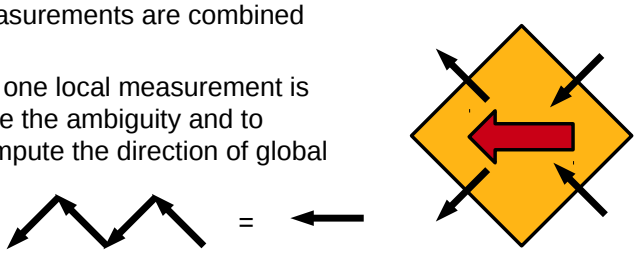
We may see motion perpendicular to edge because:

- it is average of all possibilities
- it predicts the slowest movement

Aperture problem: solutions

Local motion measurements are combined across space.

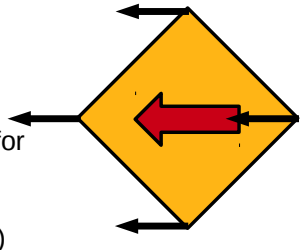
i.e. more than one local measurement is used to resolve the ambiguity and to accurately compute the direction of global motion.



Locations where the direction of motion is unambiguous are used.

i.e. corners

Note, SIFT and Harris corner detectors commonly used for finding interest points for solving stereo correspondence. The same features are also good for solving video correspondence (i.e. calculating optic flow)



Optic flow applications

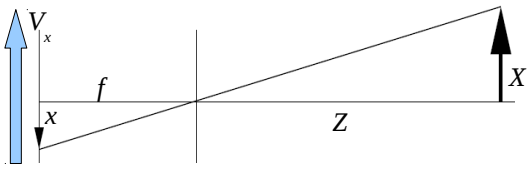
Optic flow can be used in various ways.

- To estimate the layout of the environment
  - depths and orientations of surfaces.
- To estimate ego motion
  - the camera velocity relative to a visual frame of reference.
- To estimate object motions
  - relative to the visual frame of reference, or relative to an environmental frame of reference.
- To obtain predictive information for the control of action. This information need not make layout or motion explicit.

Depth from optic flow and known ego-motion

Simple case 1:

- direction of motion is perpendicular to optical axis
- velocity V_x of camera is known



$$x = \frac{fX}{Z}$$

Given two images taken at times 1 and 2:

$$Z = \frac{fX_1}{x_1} = \frac{fX_2}{x_2} \quad \Rightarrow \quad X_1 x_2 = X_2 x_1$$

$$X_1 x_2 = (X_1 - V_x t) x_1$$

$$X_1 (x_2 - x_1) = -V_x t x_1$$

$$X_1 = \frac{-V_x x_1}{\dot{x}} \quad \text{where:} \quad \dot{x} = \frac{(x_2 - x_1)}{t}$$

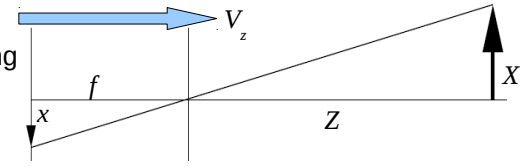
$$Z = \frac{fX_1}{x_1} = -\frac{fV_x}{\dot{x}}$$

Hence, by measuring the velocity of an image point, we can recover its depth.

Depth from optic flow and known ego-motion

Simple case 2:

- direction of motion is along camera optical axis
- velocity V_z of camera is known



$$x = \frac{fX}{Z}$$

Given two images taken at times 1 and 2:

$$fX = x_1 Z_1 = x_2 Z_2 \implies x_1(Z_2 + V_z t) = x_2 Z_2$$

$$x_1 V_z t = (x_2 - x_1) Z_2 \implies Z_2 = \frac{V_z x_1}{\dot{x}}$$

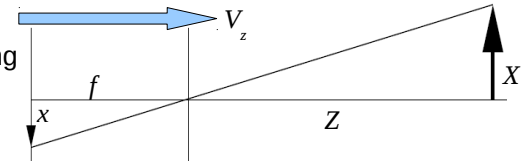
where $\dot{x} = \frac{(x_2 - x_1)}{t}$

Hence, by measuring the velocity of an image point, we can recover its depth.

Time-to-collision from optic flow

Simple case 2:

- direction of motion is along camera optical axis
- velocity V_z of camera is **unknown**



$$Z_2 = \frac{V_z x_1}{\dot{x}} \implies \frac{Z_2}{V_z} = \frac{x_1}{\dot{x}}$$

= time-to-collision (if the camera velocity is constant).

can be measured purely from the image.

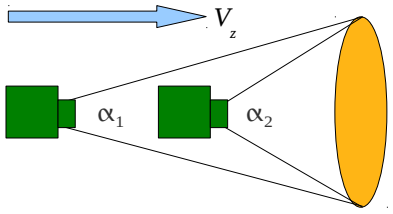
Time-to-collision from optic flow

Time-to-collision = $\frac{x_1}{\dot{x}} = \frac{\alpha_1}{\dot{\alpha}} = \frac{2A_1}{\dot{A}}$

where:

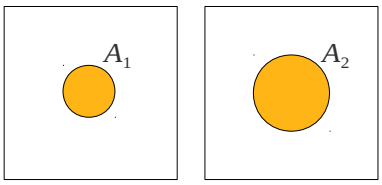
α is the angle subtended by the object, and

A is the area of the the object's image.



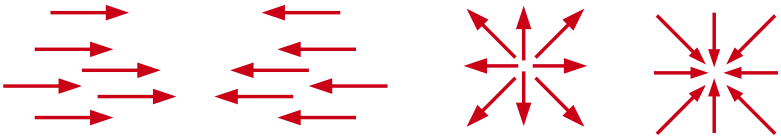
Hence, time-to-collision can be calculated without knowing anything about the speed of the camera, the size of, or distance from, the object.

Used in nature by birds and insects for catching prey, landing on a surface, etc.



Ego-motion from optic flow

Camera translations induce characteristic patterns of optic flow (for a static scene).



turn/translate left | turn/translate right | move forward | move backward

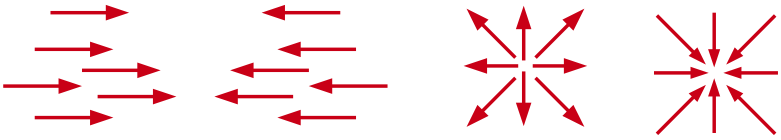
Parallel Optic Flow Field ($V_z = 0$)

- all optic flow vectors are parallel
- direction of camera movement opposite to direction of optic flow field
- speed of camera movement proportional to length of optic flow vectors

Radial Optic Flow Field ($V_z \neq 0$)

- all optic flow vectors point towards/away from a vanishing point (p_0)
- direction of camera movement determined by whether FOE (focus of expansion) or FOC (focus of contraction)
- destination of movement is FOE

Relative depth from optic flow



turn/translate left | turn/translate right | move forward | move backward

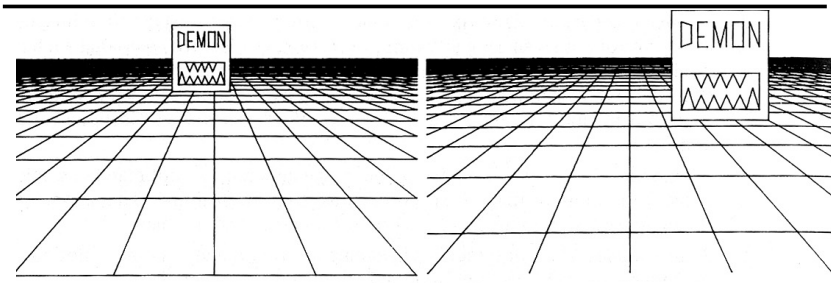
Parallel Optic Flow Field ($V_z = 0$)

- depth inversely proportional to magnitude of optic flow vector
- This is the same as motion parallax with fixation on infinity

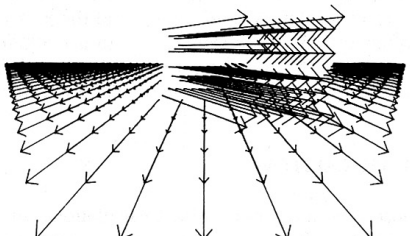
Radial Optic Flow Field ($V_z \neq 0$)

- depth of point p inversely proportional to magnitude of optic flow vector, and also proportional to distance from p to p_0

Segmentation from optic flow



I(x,y,t) I(x,y,t+1)



Discontinuities in optic flow field indicate different depths, and hence, different objects.

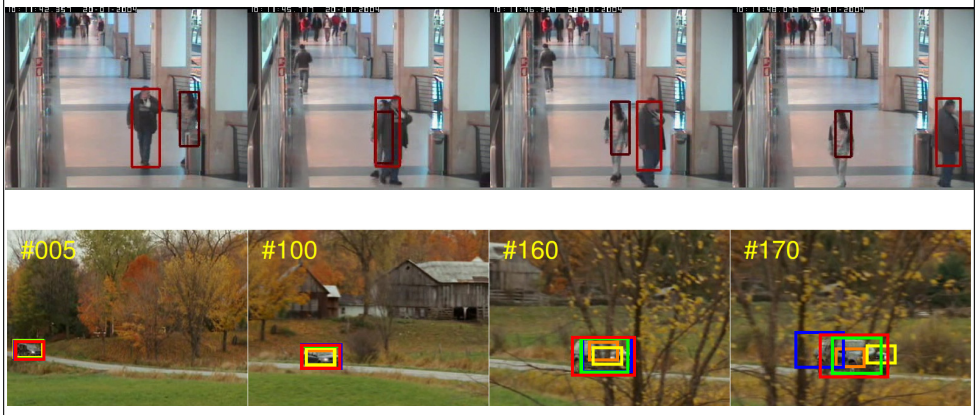
Optic flow applications

Using optic flow you can

- get to a destination by moving so that the destination is the FOE
- judge relative depths by relative magnitudes of optic flow vectors (points closer to the camera move more quickly across the image plane)
- measure absolute depths (with knowledge of camera velocity)
- judge camera speed by the rates of expansion/contraction
- measure time-to-collision
- judge direction of ego-motion
- judge directions and speeds of object motions
- segment objects at different depths / determine orientation of surfaces

Tracking

When high-level features, such as objects, are matched across several frames, the algorithms are usually referred to as tracking algorithms rather than optic flow algorithms.



Tracking

When high-level features, such as objects, are matched across several frames, the algorithms are usually referred to as tracking algorithms rather than optic flow algorithms.

Tracking algorithms use previous frames to PREDICT location of object in next frame

Intent:

– Do less work looking for the object, restrict the search.

– Get improved estimates since measurement noise is averaged out.

Methods:

– Kalman filtering

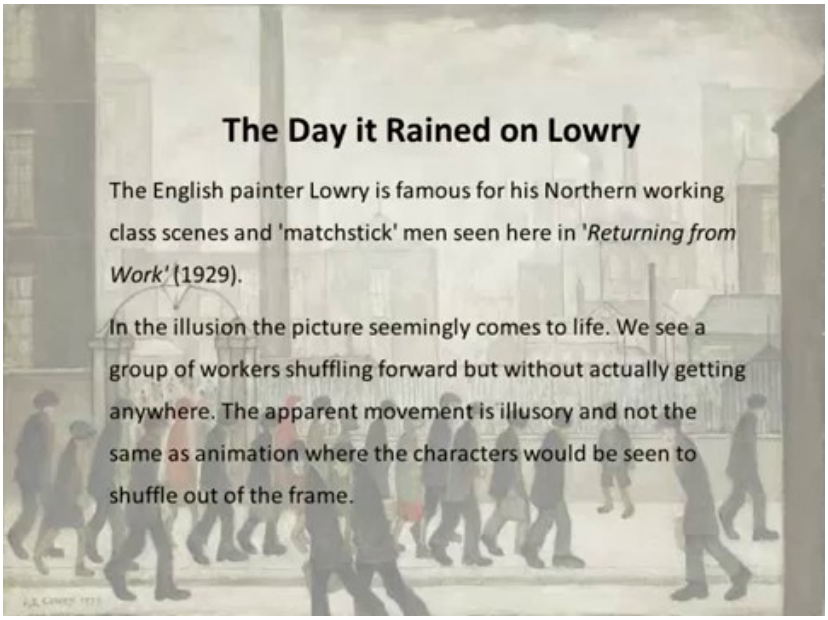
– Particle filtering

Tracking

Humans PREDICT movement

- Extrapolating object trajectory from previous locations
- But also, from high-level knowledge of typical movement

Tracking



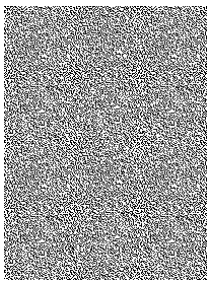
Segmentation from motion

Motion → optic flow discontinuities → segmentation

Motion → optic flow → depth → segmentation

Motion → segmentation

e.g. camouflaged objects are more easily seen when they move



i.e. Gestalt law of common fate

Segmentation: image differencing

For a static camera, image differencing can be used to segment moving objects from static ones.

Successive image are subtracted pixel by pixel, and a binary image containing absolute differences exceeding some threshold is generated.

The intensity level changes most in regions where there is motion, and hence these regions are detected.

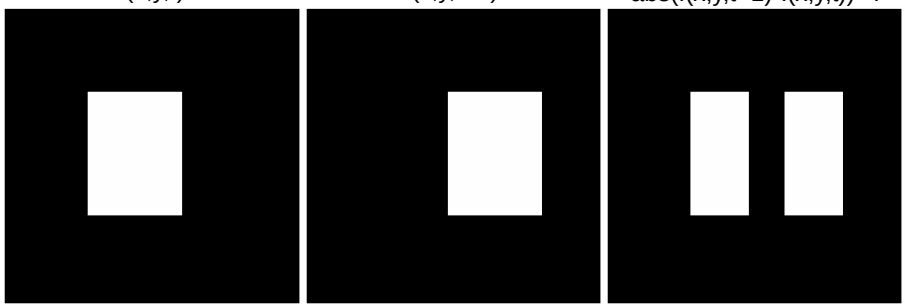
I(x,y,t) I(x,y,t+1) abs(I(x,y,t+1)-I(x,y,t))>T

Segmentation: image differencing

One problem with this technique is that the grey level changes most where background is replaced by the object (or vice versa), and may not change in locations occupied by the object in both fames.

Thus there is a kind of double view of each object.

I(x,y,t) I(x,y,t+1) abs(I(x,y,t+1)-I(x,y,t))>T



Segmentation: background subtraction

To avoid this problem learn the static part of the scene and use this reference image as the background to be subtracted.

Now objects that are temporarily stationary are still seen as foreground (e.g. car at bottom-left of example).

Still missing pixels and noise – morphological operations (e.g. dilation and erosion – see lecture on image segmentation) can be used to clean up result.

B(x,y) I(x,y,t) abs(I(x,y,t)-B(x,y))>T

Segmentation: background subtraction

How to learn the static part of the scene?

Adjacent Frame Difference $B(x,y)=I(x,y,t-1)$

- Each image is subtracted from previous image in sequence (i.e. image differencing)

Off-line average $B(x,y)=\frac{1}{N}\sum_{t=1}^{N} I(x,y,t)$

- Pixel-wise mean values are computed during separate training phase (also called Mean and Threshold)

Moving average $B(x,y)\leftarrow(1-\beta)\,B(x,y)+\beta\,I(x,y,t)$

- Background model is linear weighted sum of previous frames. Can overcome problems of slow illumination changes, so most typically used.

Segmentation: background subtraction

Hence, a typical algorithm for motion segmentation is:

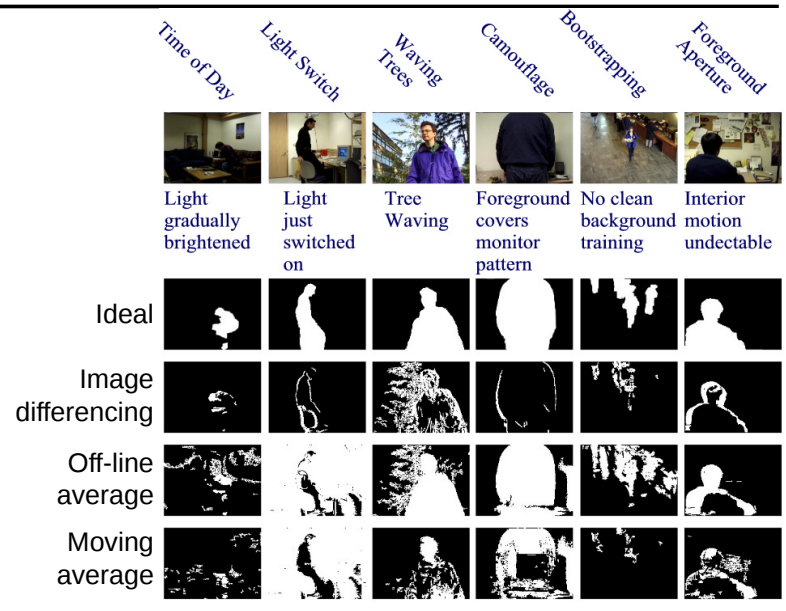
for each frame (t = 1:N)

Update background model	$B(x,y)\leftarrow(1-\beta)\,B(x,y)+\beta\,I(x,y,t)$
Compute frame difference	$\delta(x,y,t)=\|I(x,y,t)-B(x,y)\|$
Threshold frame difference	$\delta_T(x,y,t)=\delta(x,y,t)>thres$
Noise removal	$\delta_T(x,y,t)=close(\delta_T(x,y,t))$

end

Objects are detected where $\delta_T(x,y,t)$ is non-zero

Segmentation: examples



Segmentation: background subtraction

How to learn the static part of the scene?

Ideally background image should:

1. exclude “interesting” changes so that these are not subtracted from the image:
   • e.g. objects (even if these stop temporarily)

2. incorporate “uninteresting changes” so that these are subtracted from image:
   • illumination changes (e.g. due to sun/clouds, lights being switched on/off, shadows)
   • static objects that deform (e.g. trees blowing in wind)

Summary: Optic flow

Aim: to infer properties of the 3D scene from 2 or more images taken at different times. Two sub-problems:

1. Correspondence. Determining which points in the different frames are projections of the same point in the scene.

Hence, calculate optic flow (e.g. motion disparities).

Constraints:

- Small motion (extent of search reduced by expectation that motion between frames is small)
- Spatial coherence (optic flow varies smoothly assuming continuous surfaces)

Summary: Optic flow

2. Reconstruction. Given the correspondence between points, calculate 3D structure and motion of the observed scene.

- with knowledge of ego-motion: calculate absolute depth
- without knowledge of ego-motion:
  - calculate relative depths
  - time-to-collision
  - direction of ego-motion
  - heading of ego-motion

Summary: Segmentation

Segment moving objects from stationary background.

Calculate: *abs(I(x,y,t)-B(x,y))>T*

- *image differencing* $B(x,y)=I(x,y,t-1)$
- *background subtraction*
  - » *off-line average* $B(x,y)=\frac{1}{N}\sum_{t=1}^{N} I(x,y,t)$
  - » *moving average* $B(x,y)\leftarrow(1-\beta)B(x,y)+\beta I(x,y,t)$