Motion

CS 554 – Computer Vision
Pinar Duygulu
Bilkent University

Motion

- A lot of information can be extracted from time varying sequences of images, often more easily than from static images.
  - e.g, camouflaged objects are only easily seen when they move.
  - the relative sizes and position of objects are more easily determined when the objects move.

- For example, given an image of a moving car, deciding which pixels in the image represent motion can help to decide which pixels belong to the car, and which to the static background.

Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

Biological Motivation for Studying Motion

- Estimation of the motion of predators advancing at a mobile animal is important to its ability to take flight away from the predator and survive
- Human beings do it all the time without even realizing it, for example, this is why we have saccadic eye movements (that is our eyes jump from focusing at one spot to another). Thus, if the scene has no motion, and we are still our eyes are moving.
- Fixating on something close and very far away and moving your head (either sideways or forward and backward), you can notice that the retinal image of the close by tree moves more than the one of a distant tree, i.e. the motion in the retinal plane is inversely proportional to the distance from the retinal plane.
- Animals obtain some information about the environment structure, e.g. pigeons move their necks to get the so called ``motion parallax".

Motion

- Studying the motion in detail, we can answer such questions as
  - How many moving objects there are?
  - Which directions they are moving in?
  - Whether they are undergoing linear or rotational motion?
  - How fast they are moving?

Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

Motion

- The analysis of visual motion consists of two stages:
  - the measurement of the motion,
  - the use of motion data to segment the scene into distinct objects and to extract three dimensional information about the shape and motion of the objects.

Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

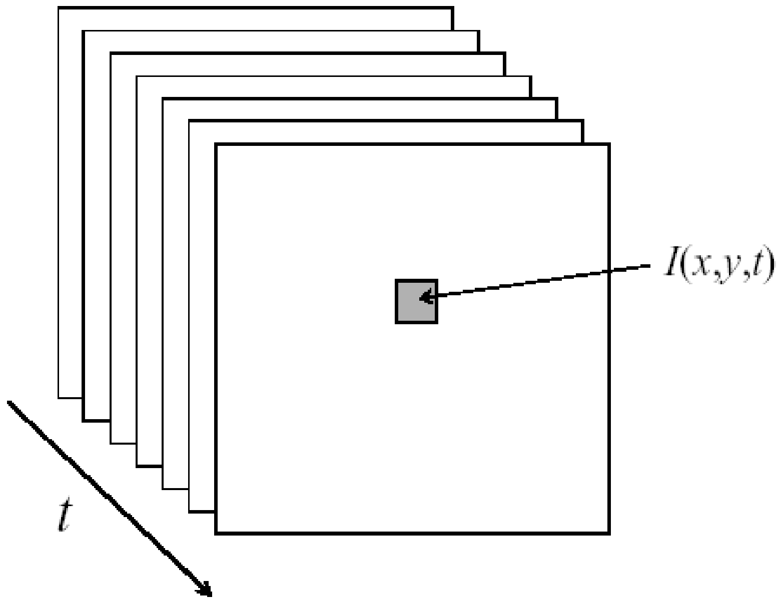
Motion

- There are two types of motion to consider:
  - movement in the scene with a static camera,
  - movement of the camera, or ego motion.

Since motion is relative anyway, these types of motion should be the same.

However, this is not always the case, since if the scene moves relative to the illumination, shadow effects need to be dealt with. Also, specularities can cause relative motion within the scene. For this lecture, we will ignore all such complications.

Motion



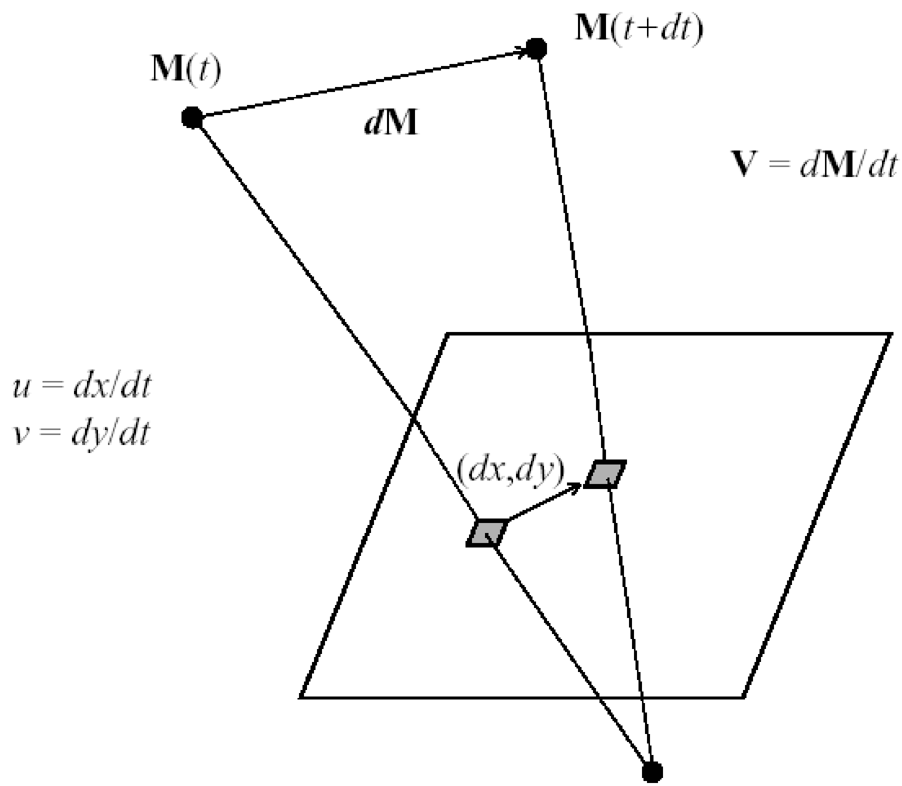
We are now considering a sequence of images captured over time.

This can be viewed as adding a third dimension, the time t, to the image. That is, each pixel is not only a function $I(x,y)$ of the spatial coordinates (x,y) but also a function $I(x,y,t)$ of time.

The main issue in motion analysis is to recover information about the motion in the scene from the variation of the intensities $I(x,y,t)$ over time.

Adapted from Martial Hebert, CMU

Motion field



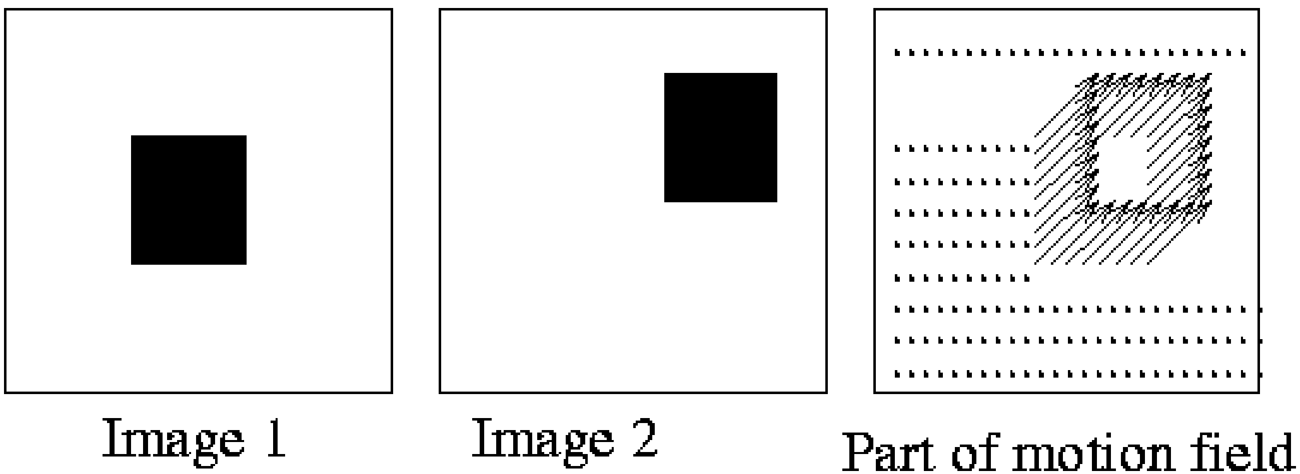
When an object moves in front of a camera, there is a corresponding change in the image

- Given a point in the scene at position **M** at time t, the point moves to **M**(t+dt) after a small interval dt.
- The corresponding velocity vector is **V** = d**M**/dt. The points **M**(t) and **M**(t+dt) project at points **m**(t) and **m**(t+dt) in the image.
- If $x(t)$ and $y(t)$ are the camera coordinates of **m**(t), the apparent velocity in the image has components: $u = dx/dt$ and $v = dy/dt$
- Motion filed is the set of values $u(x,y)$ and $v(x,y)$ over the image
- The problem of motion recovery is then to estimate u and v at the image pixels

Adapted from Martial Hebert, CMU

Motion field

- If we are only dealing with rigid body translations and rotations, then the motion field will be continuous except at the silhouette boundaries of objects.



The motion field of a moving square.

Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

The motion field

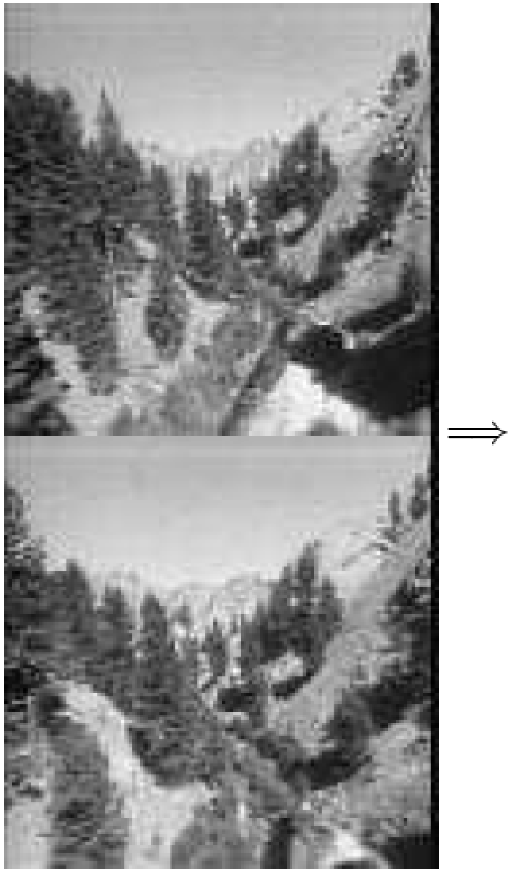
•In the case of pure camera translation, the direction of motion is along the projection ray through that image point from which (or towards which) all motion vectors radiate.

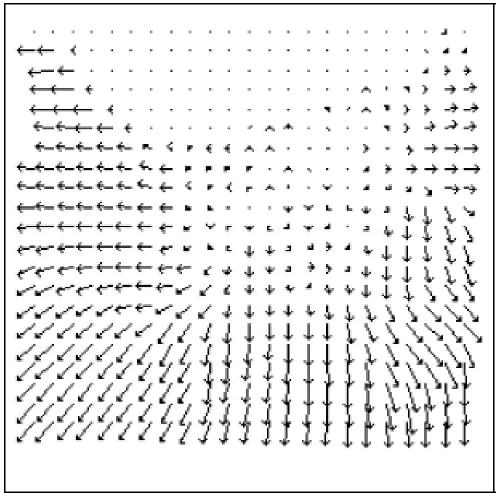
•The point of divergence (or convergence) of all motion field vectors is called the *focus of expansion* FOE (or *focus of contraction* FOC).

Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

Focus of Expansion – for translating camera

- In the case of divergence we have forward motion of the camera



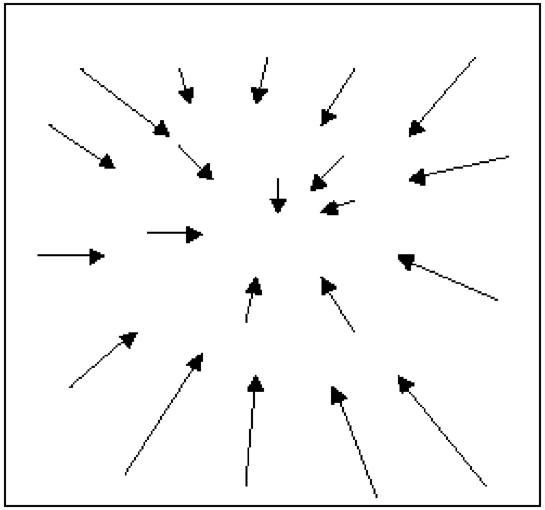


points closer to the camera move more quickly across the Image plane

Adapted from Michael Black

Focus of Expansion – for translating camera

and in the case of convergence, backwards motion.



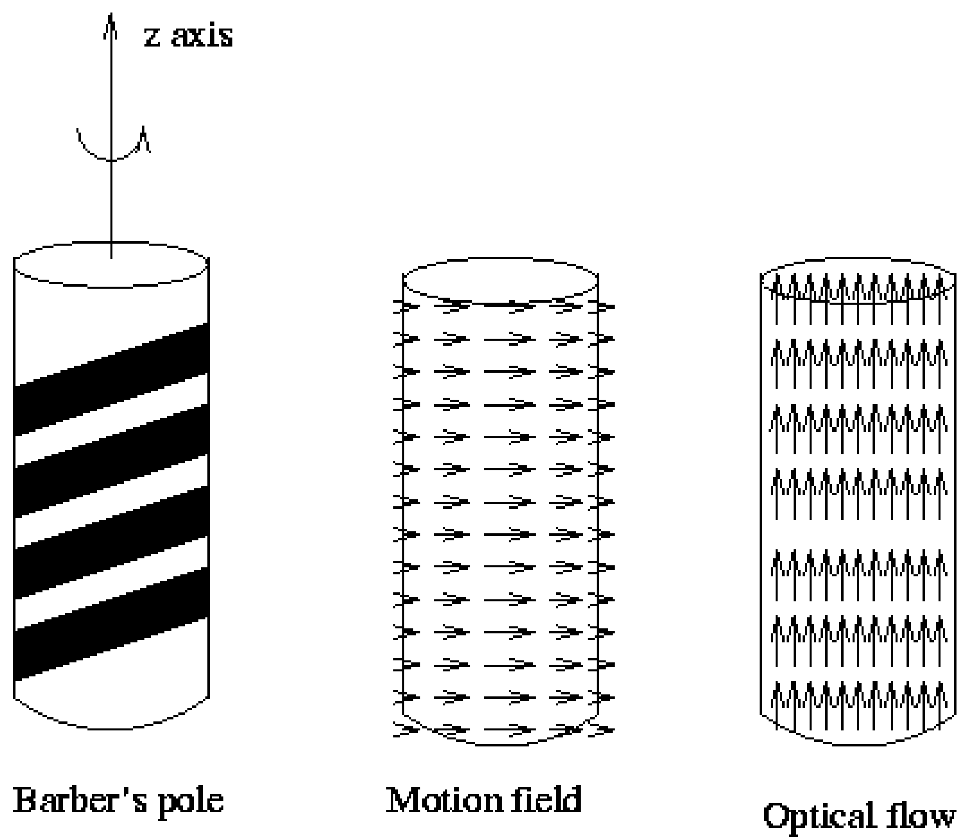
Adapted from Trevor Darrell, MIT

Optical flow

Optical flow is the apparent motion of brightness patterns in the image. Generally, optical flow corresponds to the motion field, but not always.

For example, the motion field and optical flow of a rotating barber's pole are different.

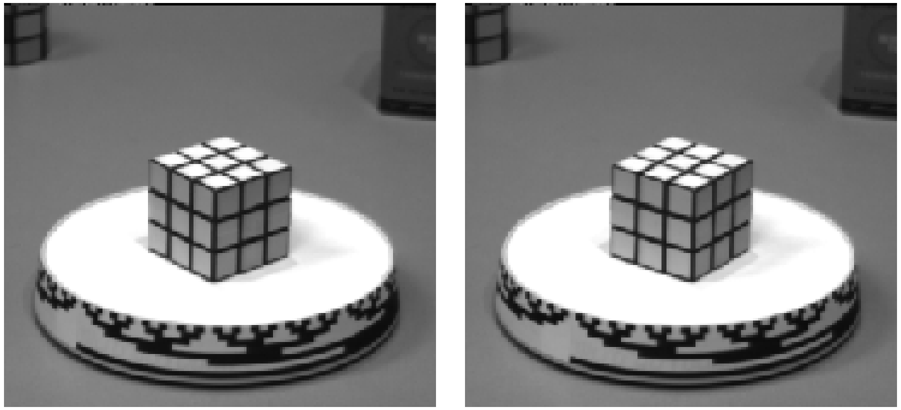
In general, such cases are unusual, and for this lecture we will assume that optical flow corresponds to the motion field.

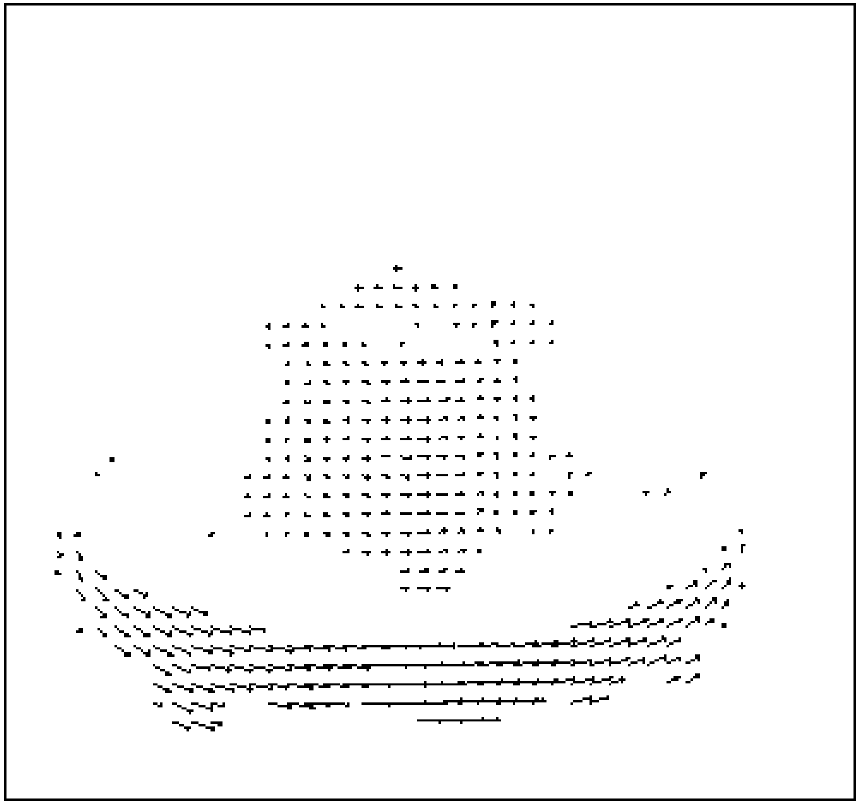


Adapted from Robyn Owens, http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html

Optical flow

The optical flow describes the direction and the speed of motion of the features in the image.



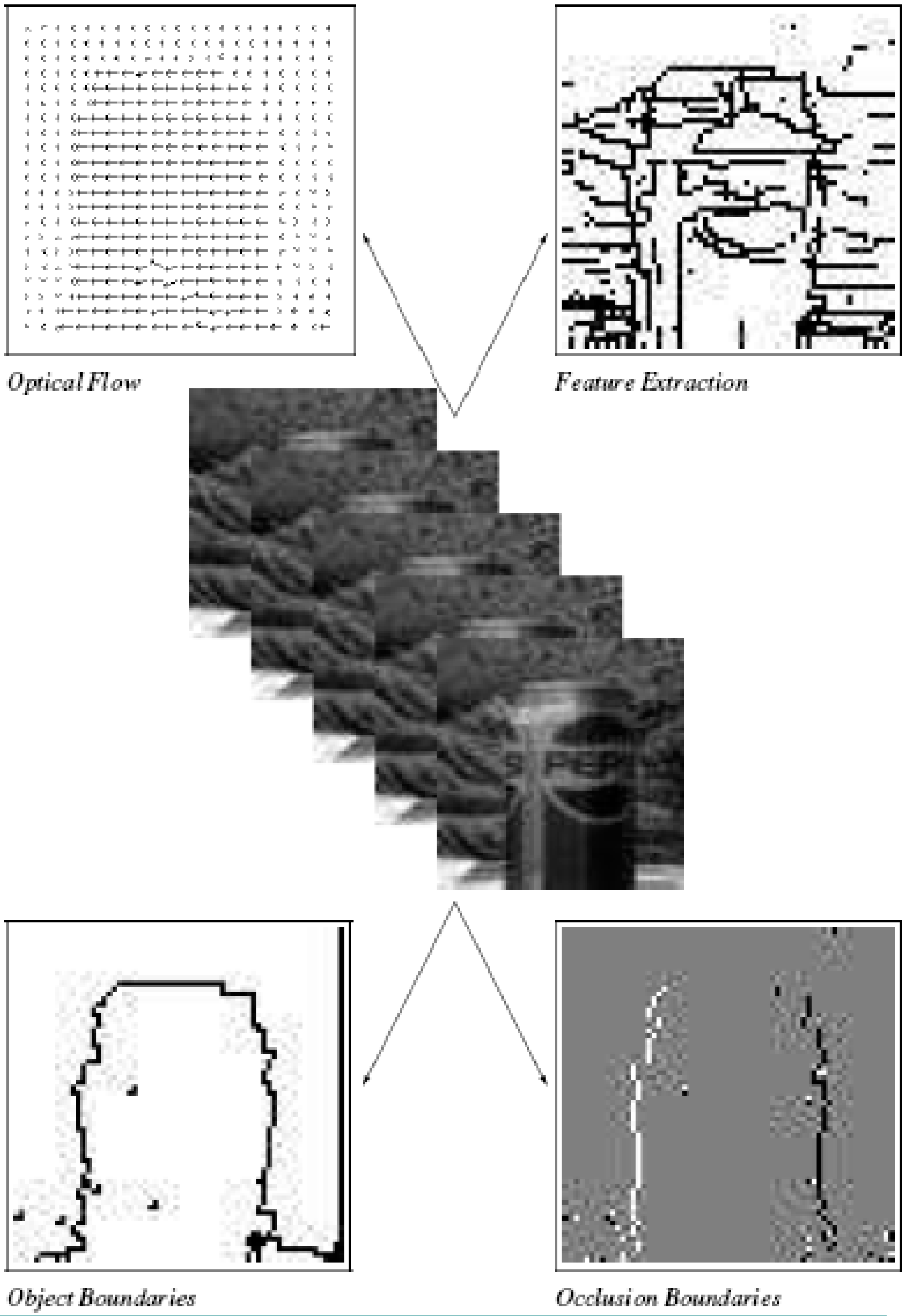


A Rubik's cube on a rotating turntable and Flow vectors calculated from comparing the two images of a Rubik's cube , taken from Russell and Norvig, ``AI, A Modern Approach'', Prentice Hall, 1995,

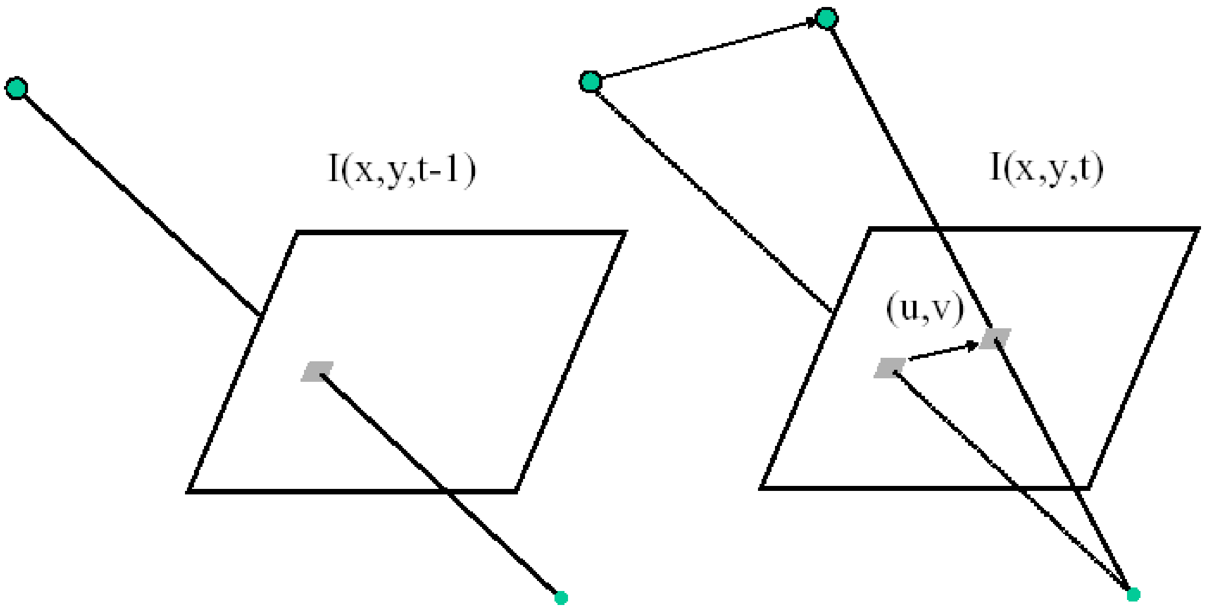
Adapted from http://robotics.eecs.berkeley.edu/~sastry/ee20/vision3/node2.html

Optical Flow

- Figure contains two distinct motions:
- The can is moving more rapidly than the background
- This can be used for detecting object boundaries



Optical flow



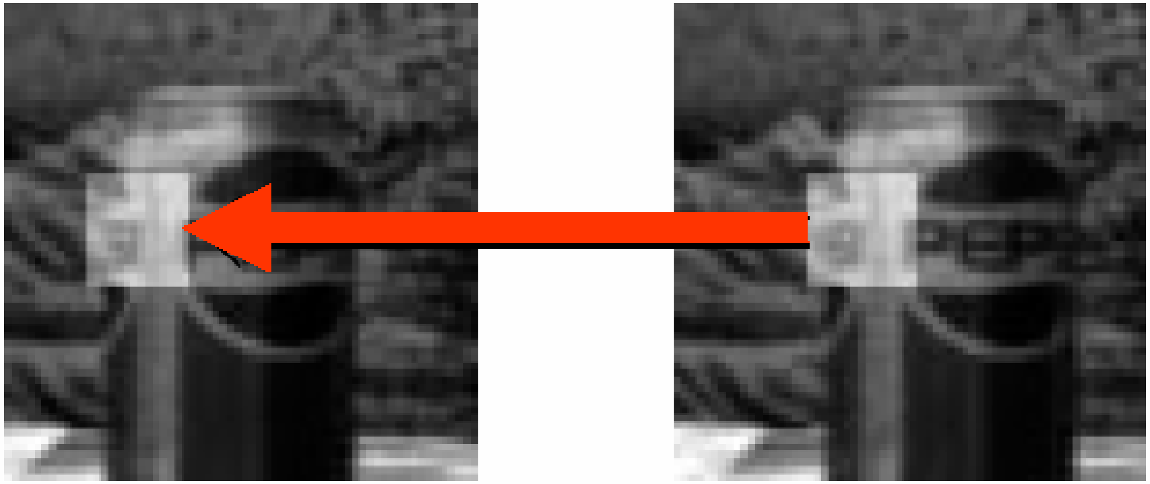
Let us consider two frames at times t-1 and t.

At a given pixel (x,y), the flow components are given by u(x,y) and v(x,y).

At time t, pixel (x,y) has moved to position (x+u(x,y), y+v(x,y)) with intensity I(x+u(x,y), y+v(x,y), t).

Adapted from Martial Hebert, CMU

Brightness Constancy



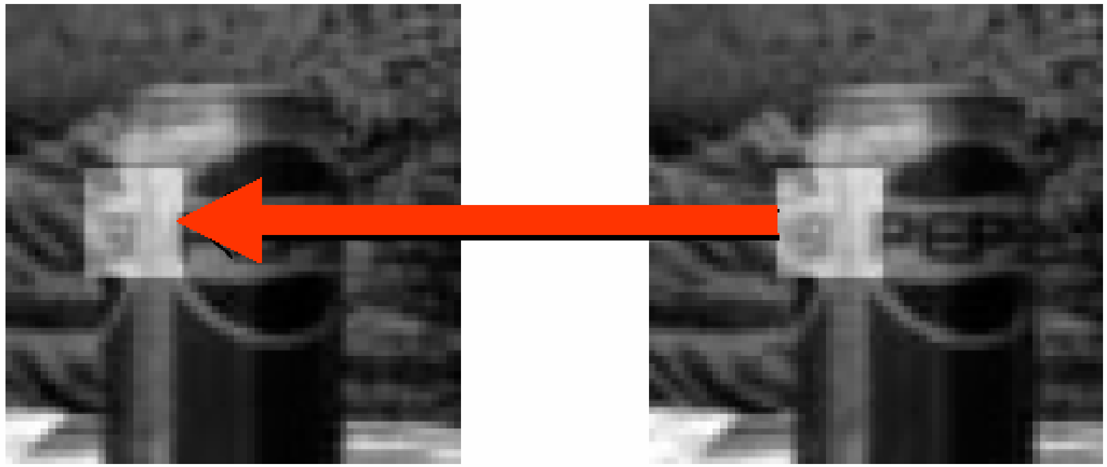
Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

i.e. the intensity of a scene point does not change over time. This implies that we assume Lambertian sufraces and no change in illumination during the interval dt.

Adapted from Michael Black, Brown University

Brightness Constancy



$$I(x+u, y+v, t+1) = I(x, y, t)$$

(assumption)

This assumption of *brightness conservation* implies that : assuming that u and v are small, we can use a first order approximation:

I(x+u(x,y), y+v(x,y), t+1) = I(x,y,t)

Adapted from Michael Black, Brown University

Optical Flow Equation

- Assuming that u and v are small we can make a first-order approximation

$$I(x+u(x,y), y+v(x,y), t) \approx I(x,y,t) + u(x,y)\frac{\partial I}{\partial x} + v(x,y)\frac{\partial I}{\partial y}$$

The two derivatives are the components of the image gradient at (x,y) which are denoted by Ix and Iy .

I_t as the difference I(x,y,t)-I(x,y,t-1)

Substituting this approximation of I(x+u(x,y), y+v(x,y), t)in the brightness constancy equation, we end up with the fundamental equation of motion:

$$uI_x + vI_y + I_t = 0$$

Adapted from Martial Hebert, CMU

Notation

$$uI_x + vI_y + I_t = 0$$

- The component of the image velocity in the direction of the image intensity gradient is

$$u = \frac{-I_t I_x}{I_x^2 + I_y^2} \quad v = \frac{-I_t I_y}{I_x^2 + I_y^2}$$

Adapted from Martial Hebert, CMU

Optical flow

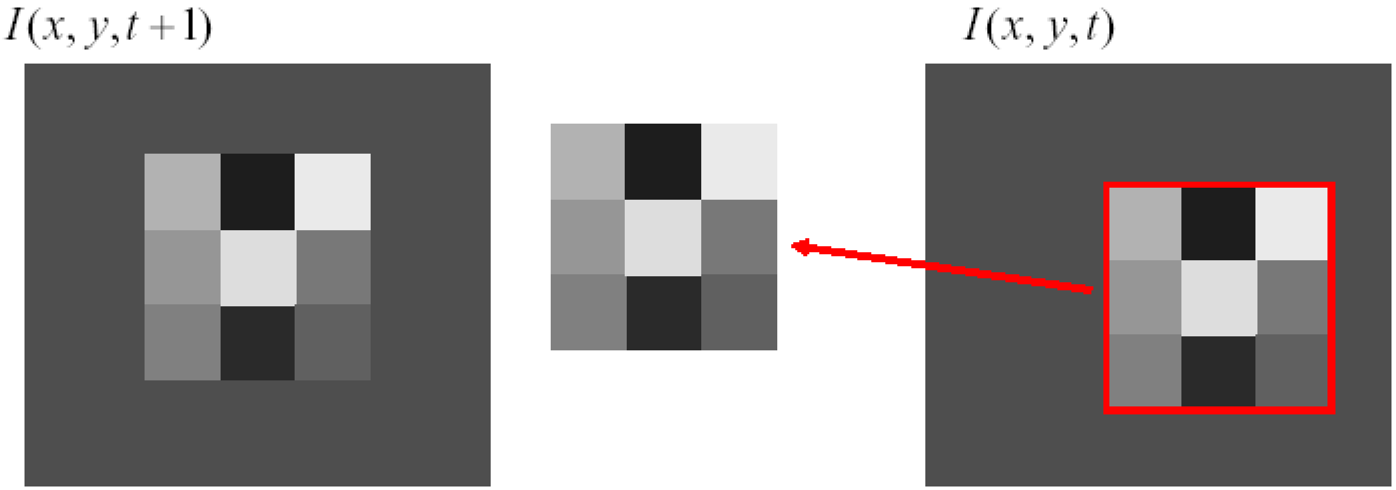
In order to be able to measure optical flow we need to find corresponding points between two frames.

One possibility would be to exploit the similarity between the image patches surrounding the individual points.

Two measures of similarity:
- Sum of squared differences (SSD)
- Cross-correlation

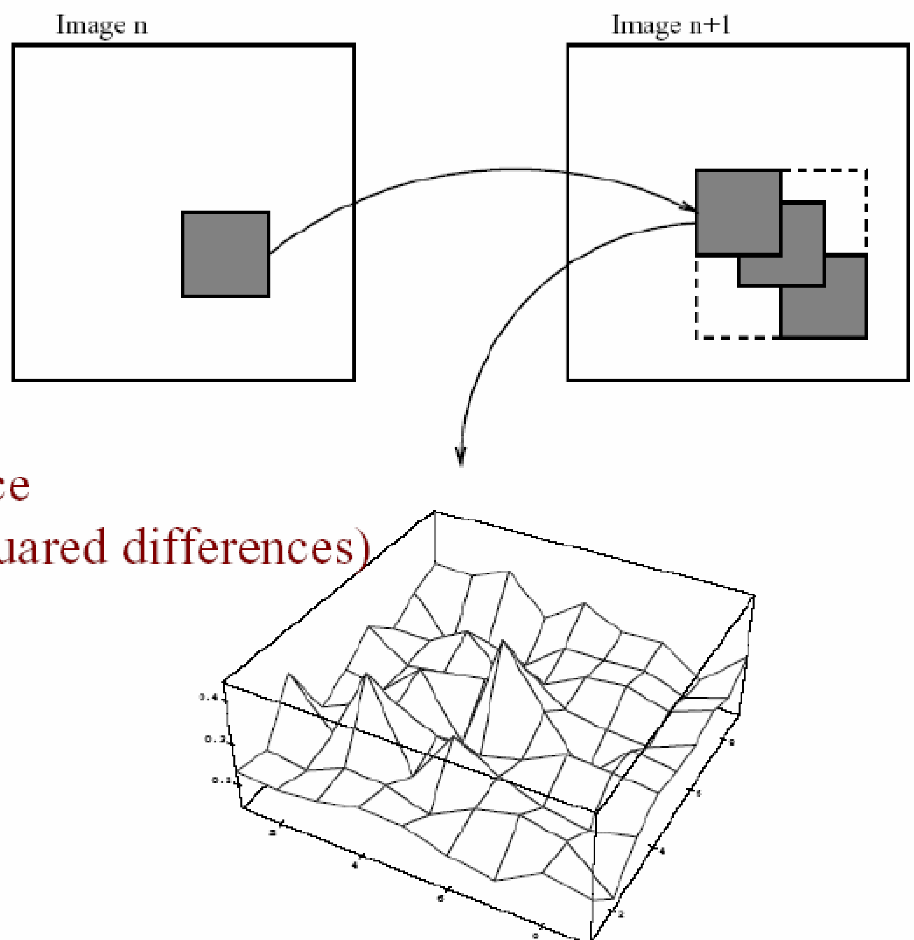
Adapted from http://robotics.eecs.berkeley.edu/~sastry/ee20/vision3/node2.html

Minimize Brightness difference



$$E_{SSD}(u,v) = \sum_{x,y\in R} (I(x+u, y+v, t+1) - I(x, y, t))^2$$

Minimize Brightness difference



SSD surface
(sum of squared differences)

$$E_{SSD}(u,v) = \sum_{x,y \in R} \left(I(x+u, y+v, t+1) - I(x,y,t) \right)^2$$

Adapted from Michael Black, Brown University

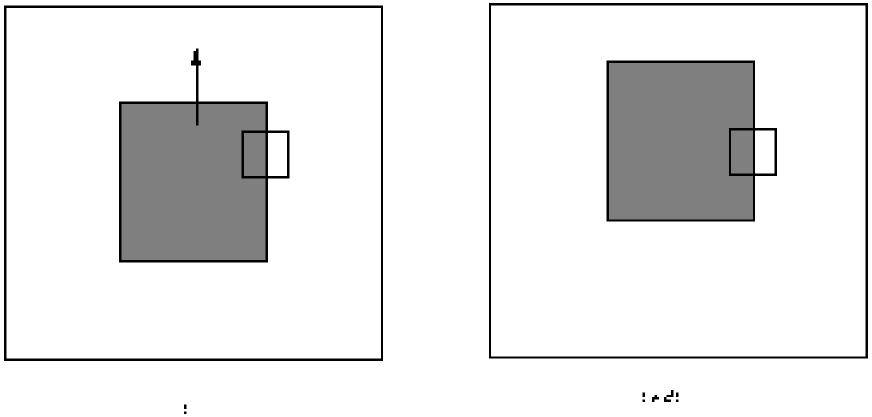
Simple SSD algorithm

```
For every pixel (i,j) in the image at t-1
    Cut out a neighborhood of pixels
    For every pixel (k,l) in image at t
        u=k-i; v=l-j;
        Compute the E(u,v)
    End for
    Choose the (u,v) that maximize E(u,v)
End for
```

Problems?

1. not very efficient
2. (u,v) defined on a pixel grid

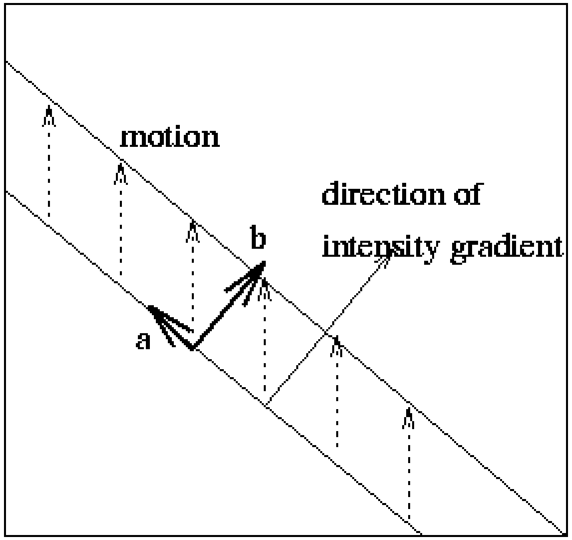
Optical flow

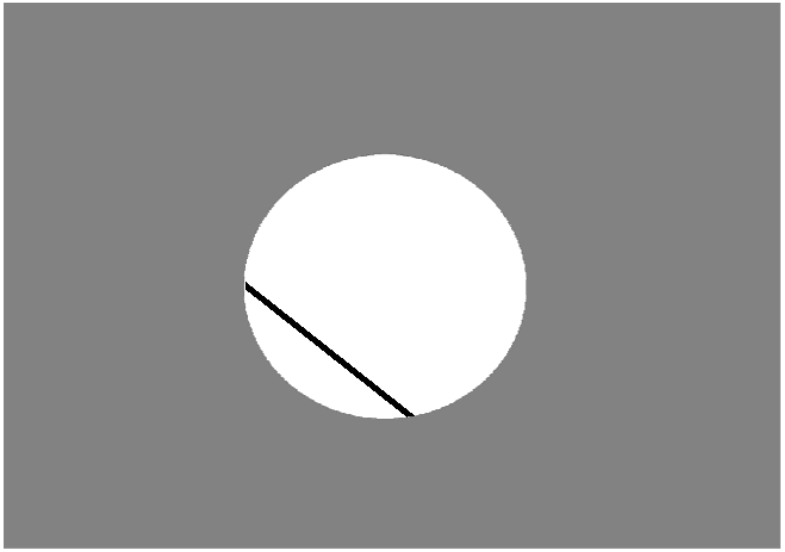


In spite of the fact that the dark square moved between the two consecutive frames, observing purely the cut-out patch we cannot observe any change, or we may assume that the observed pattern moved arbitrarily along the direction of the edge. The fact that one cannot determine the optical flow along the direction of the brightness pattern is known as *aperture problem*

Adapted from http://robotics.eecs.berkeley.edu/~sastry/ee20/vision3/node2.html

Aperture Problem





Consider one point in the image. We are computing the gradient in a small window around this point (the "aperture").

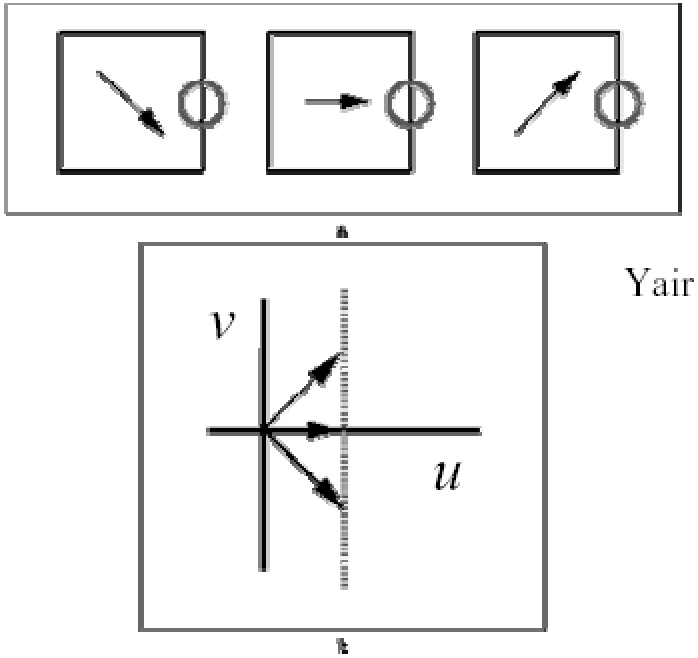
Within this small window, the intensity varies in the direction of the gradient, but not in the direction perpendicular to the gradient.

In terms of edges: the intensity varies across the edge but not along the edge.

As a result, a motion (u,v) that is parallel to the edge can never be recovered. In other words, even though the flow has two coordinates, only one of them (in the direction of the gradient) can be recovered.

Adapted from Martial Hebert, CMU

Aperture Problem



Yair Weiss

$$I_x u + I_y v = -I_t$$

$$I_x u + 0v = -I_t$$

$$I_x u = -I_t$$

$$u = -I_t / I_x$$

v could be anything

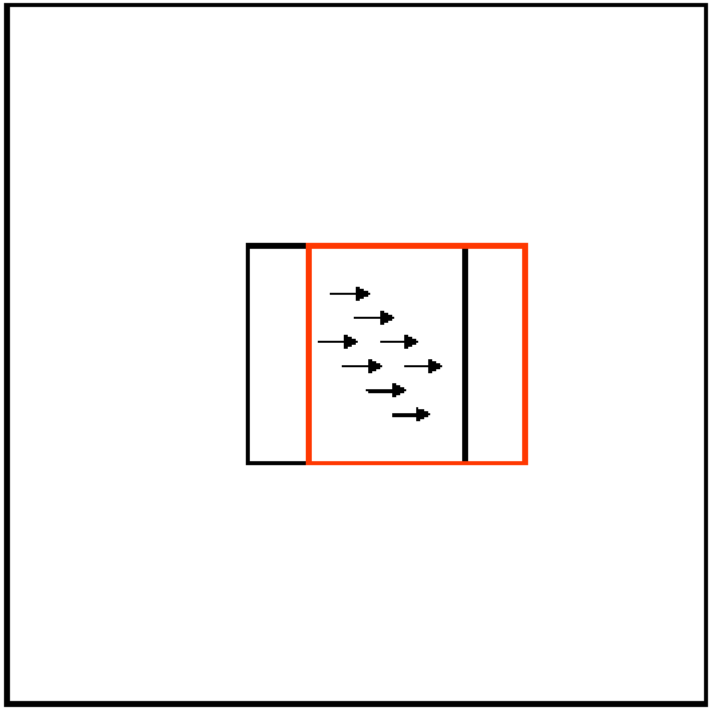
Adapted from Michael Black, Brown University

Aperture Problem

The optical flow equation gives us one constraint per pixel, but we have two unknowns u and v. Therefore, the flow cannot be recovered unambiguously from this equation alone. The problem is under-constrained.

Because of the under-constrained nature of the problem. One can solve the optical flow only by adding more constraints to the optical flow equation.

Constant flow



Suppose that we assume that the flow is constant (u,v) over a small region W in the image. The idea being that there is enough variation of gradient within W in order to recover the flow. We can express the fact that the optical flow equation is satisfied at every point of W by saying that the function E defined by:

$$E(u,v) = \sum_{(x,y)\in W} \left(uI_x(x,y) + vI_y(x,y) + I_t\right)^2$$

is close to 0. More formally, the constant flow (u,v) most consistent with the intensity values in W is found as:

$$\operatorname*{Min}_{u,v} E(u,v)$$

Constant flow

The minimization can be expressed as a simple least-squares problem:

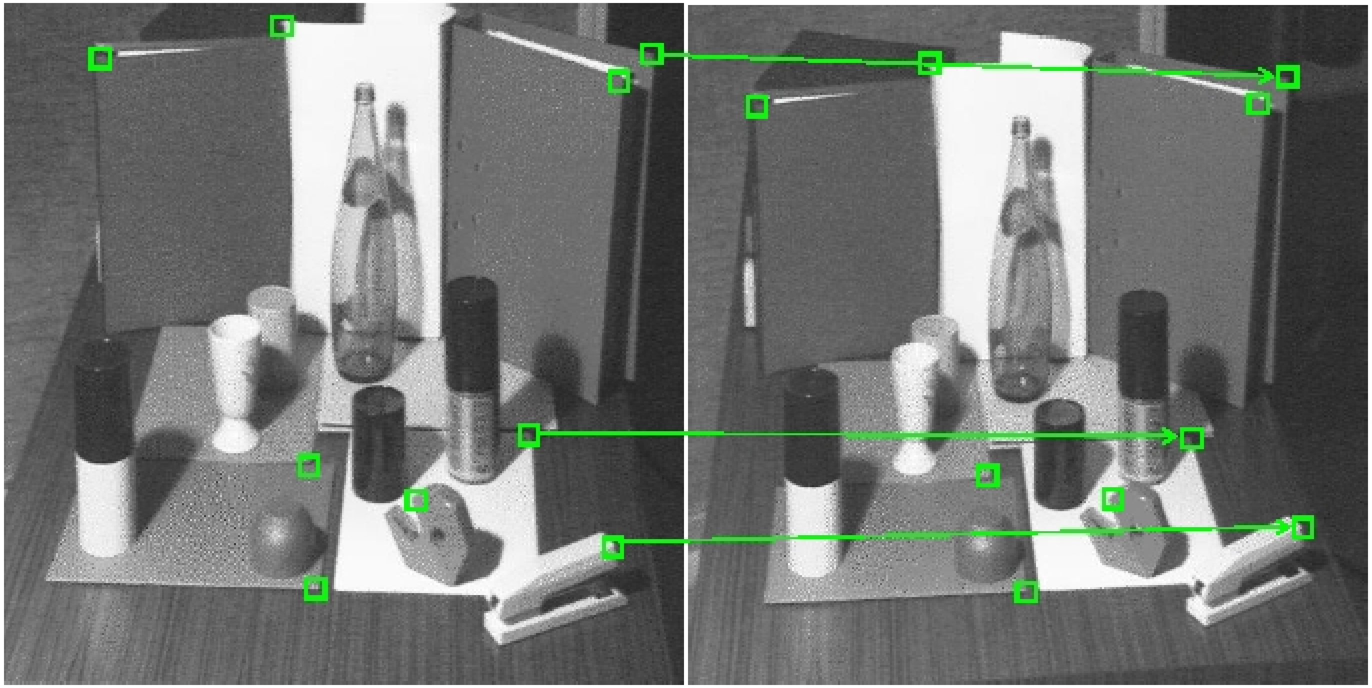
$$\mathrm{Min}\left\| \mathbf{A}\begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b} \right\|^2$$

with

$$\mathbf{A} = \begin{bmatrix} I_x(x_o, y_o) & I_y(x_o, y_o) \\ \vdots & \vdots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} I_t(x_o, y_o) \\ \vdots \\ I_t(x_n, y_n) \end{bmatrix}$$

Therefore, assuming constant flow over a region, motion can be estimated directly by a simple least squares estimation.

Feature Tracking



1) Extract features - The features that are most often used are the corner features

2) Track the features from frame to frame. For this, we are going to assume that the frame are spaced closely enough in time that the motion between frames is small.

Adapted from Martial Hebert, CMU

Feature Tracking

Given a feature at location (*x*o,*y*o) in one image I1, the tracking problem is to basically find the position (*x*o+*u*,*y*o+*v*) of the corresponding feature in the next image in the sequence I2, where *u* and *v* are the components of the motion field at (*x*o,*y*o). Let us define a window *W* around (*x*o,*y*o)

The set of pixels inside *W* in *I*1 is denoted by *W*(*I*1). Assuming that the window is not too large, we can assume that the motion field is constant inside *W*. Therefore, we can use the constant flow algorithm to recover *u* and *v*. This is recovered by minimizing in *u* and *v* the quantity:

$$E(u,v) = \sum_{(x,y)\in W} \left(uI_x(x,y) + vI_y(x,y) + I_t\right)^2$$

It turns out that the solution is given by:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Feature Tracking

Remarkably, the matrix involved in this calculation is exactly the same as the matrix used in the corner detector!!

This is not surprising: The matrix characterizes the 2-D distribution of the gradient vectors in W, and this relation states that the variation of intensity over time is the product of the motion vector by the gradient distribution.

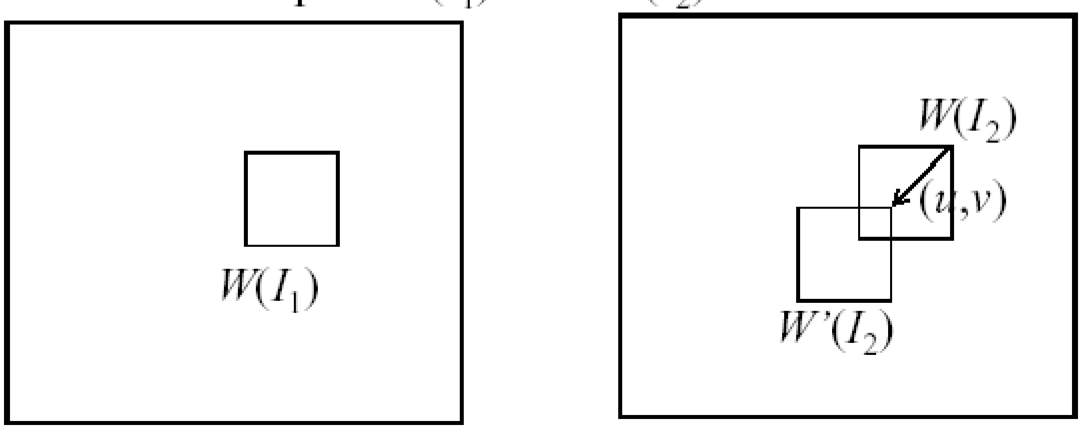
Note in particular that the matrix is singular when W contains an edge (gradient in one direction only) which is exactly when the motion vector cannot be recovered.

Feature Tracking – Iterative algorithm

- Given:
  - Images I_1 and I_2
  - Feature at (x_o, y_o) in I_1
  - Window W at (x_o, y_o)
- Compute displacement in image (u,v):

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- I_t is the pixel difference between $W(I_1)$ and $W(I_2)$
- Translate W by (u,v) to W'
- Compare $W(I_1)$ and $W'(I_2)$



the feature tracking algorithm involves solving for the constant motion inside W by inverting the "corner" matrix. This gives the offset (u,v) to the position in the next image.

At each iteration the window W is shifted by the recovered motion to W'. The pixels *W(I1)* and *W'(I2)* are then compared (using SSD for example). If they are too different, that means that the motion *(u,v)* is not quite correct and another iteration of the constant flow is applied, replacing W by W'. The iterations continue until (u,v) because very small or the windows match.

Adapted from Martial Hebert, CMU

Limits of the local gradient method

1. Fails when intensity structure within window is poor
2. Fails when the displacement is large (typical operating range is motion of 1 pixel per iteration!)
   - *Linearization of brightness is suitable only for small displacements*

Also, brightness is not strictly constant in images

- *actually less problematic than it appears, since we can pre-filter images to make them look similar*

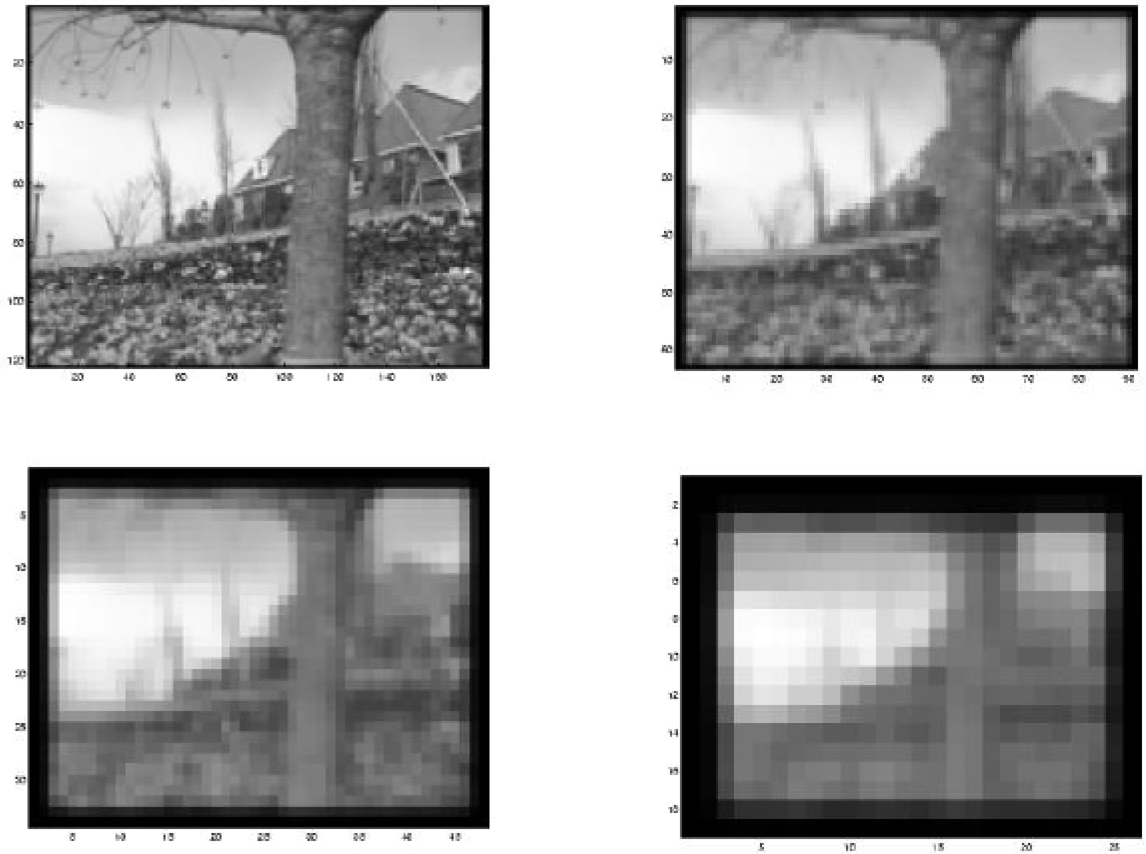
Adapted from Trevor Darrell, MIT

Feature Tracking

Since we are interested in the motion of the center point (xo,yo) of W, it might be useful to give more weight to the pixels at the center of W than to those at the periphery. This is achieved by multiplying the values Ix(x,y) and Iy(x,y) by a weight w(x,y) which is maximum at the center and tapers off at the edges of W (a Gaussian, for example.)

To deal with larger motions, it can be advantageous to first reduce the images (after smoothing) and recover the motion at a coarse scale before recovering motion at full resolution. Incidentally, this class of trackers is called the *Lucas-Kanade* tracker.

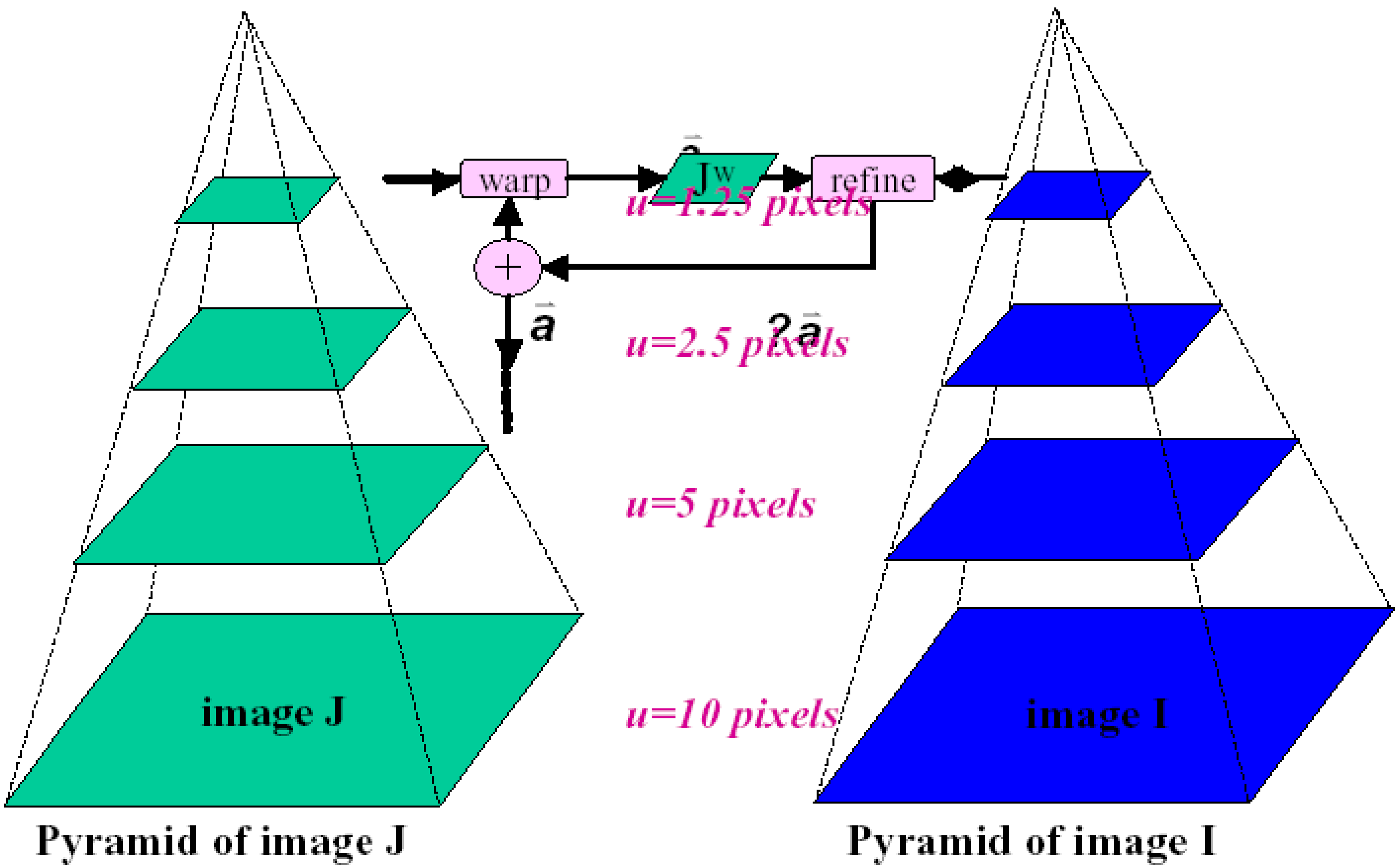
Pyramid : Coarse to fine



Adapted from Trevor Darrell, MIT

Pyramid : Coarse to fine

$$\boxed{I_x \cdot u + I_y \cdot v + I_t \approx 0}$$ ==> **small *u* and *v* ...**



Visual Tracking

Tracking is the problem of generating an inference about the motion of an object given a sequence of images.

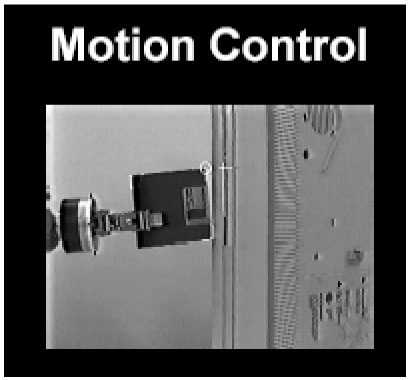
-Motion Capture: if we can track a moving person accurately, then we can make an accurate record of their motions. Once we have this record, we can use it to drive a rendering process; for example, we might control a cartoon character, thousands of virtual extras in a crowd scene, or a virtual stunt avatar. Furthermore, we could modify the motion record to obtain slightly different motions. This means that a single performer can produce sequences they wouldn't want to do in person.

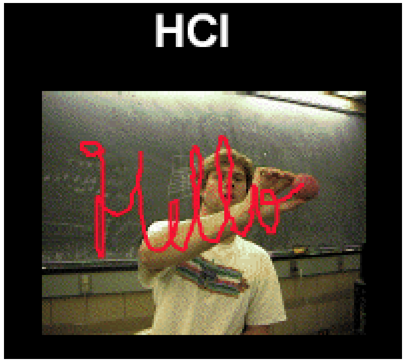
-Recognition From Motion: the motion of objects is quite characteristic. We may be able to determine the identity of the object from its motion; we should be able to tell what it's doing. –

-Surveillance: knowing what objects are doing can be very useful. For example, different kinds of trucks should move in different, fixed patterns in an airport; if they do not, then something is going very wrong. Similarly, there are combinations of places and patterns of motions that should never occur (no truck should ever stop on an active runway, say). It could be helpful to have a computer system that can monitor activities and give a warning if it detects a problem case.

-Targeting: a significant fraction of the tracking literature is oriented towards (a) deciding what to shoot and (b) hitting it. Typically, this literature describes tracking using radar or infra-red signals (rather than vision), but the basic issues are the same — what do we infer about an object's future position from a sequence of measurements? (i.e. where should we aim?)

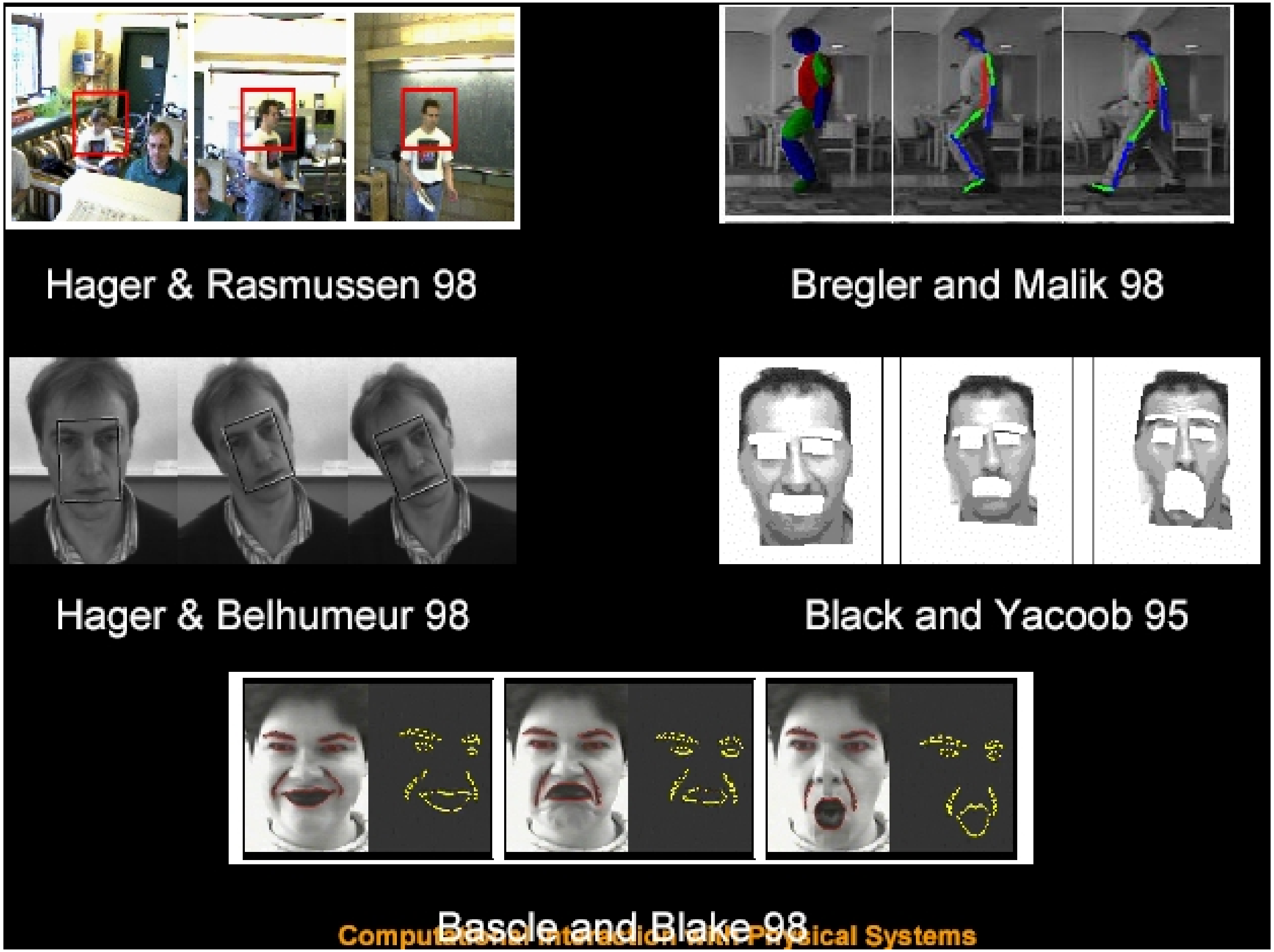
Visual Tracking





Adapted from G. Hager, JHU

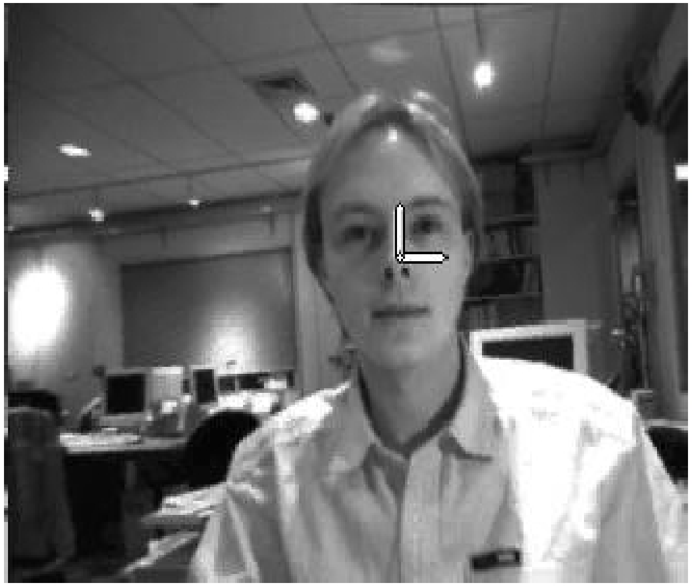
Visual Tracking



Adapted from G. Hager, JHU

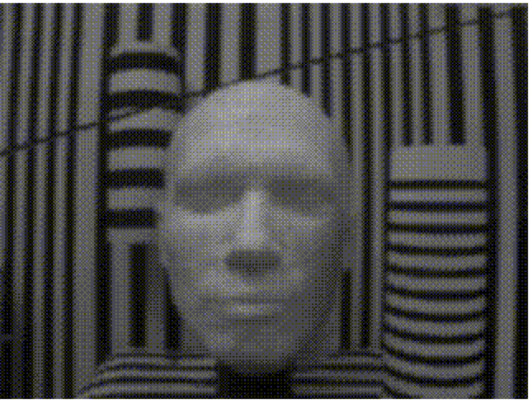
Application

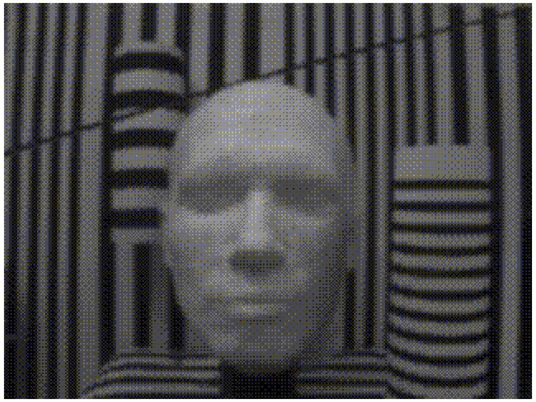
Track users' head gaze for hands-free pointing…

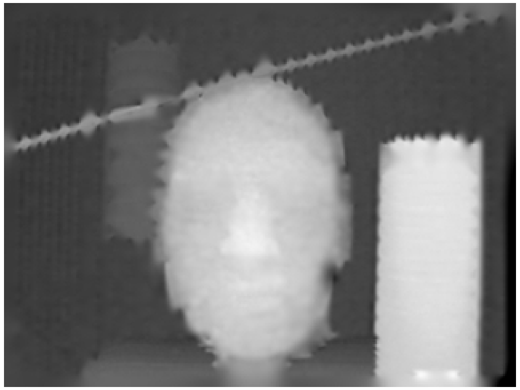


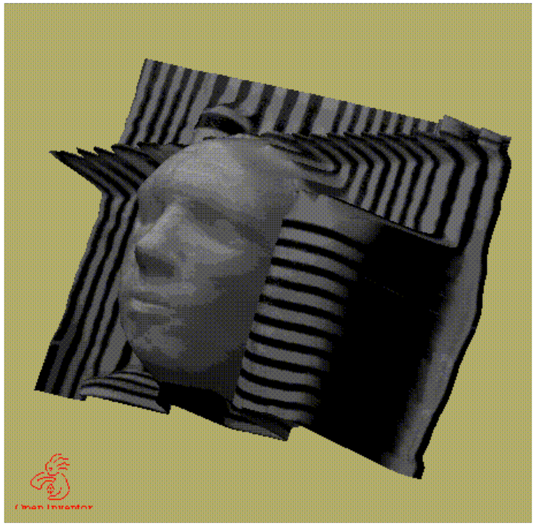
Adapted from Trevor Darrell, MIT

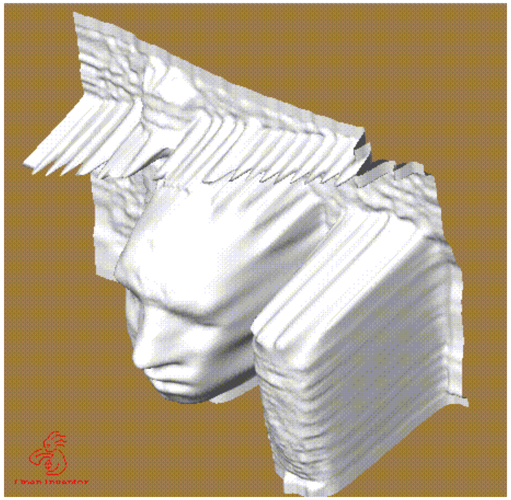
Model-based Brightness Constraints: on Direct Estimation of Structure and Motion











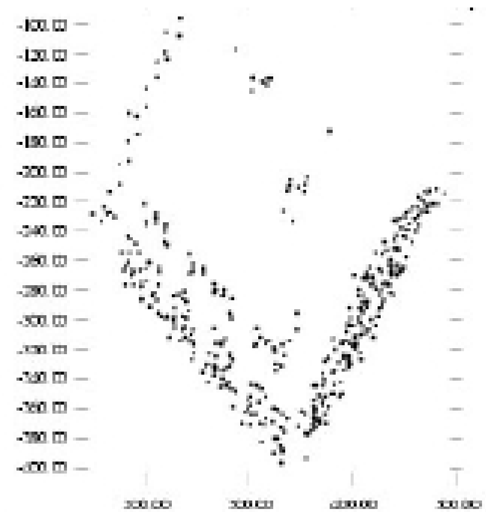
http://www.ai.mit.edu/people/gideon/Demos/DirectMethods/Demo1.html

Structure from Motion

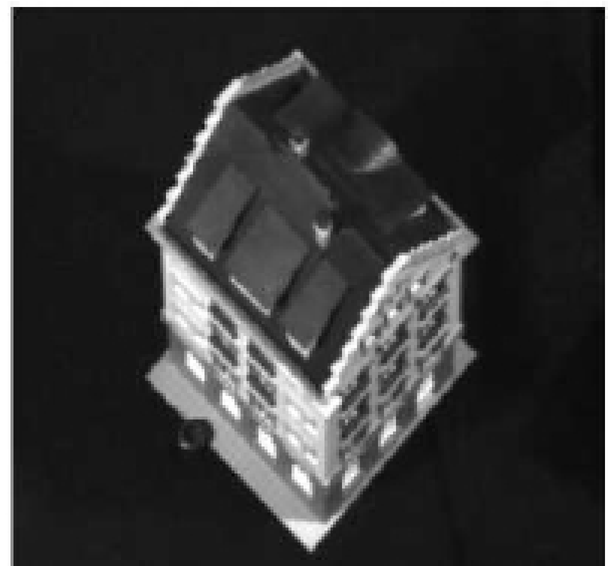
1

60

Input



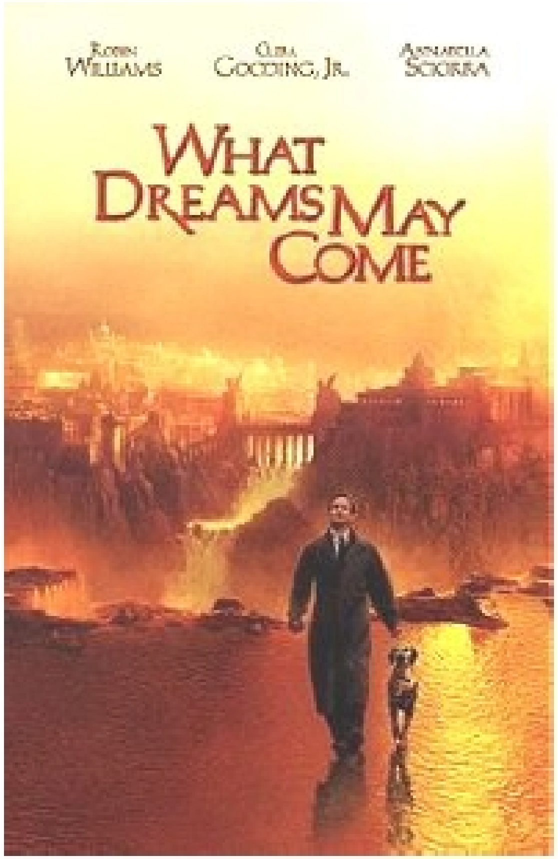
result



comparision

Structure from Motion

Applications of Optical flow



Impressionist effect.
Transfer motion of real world to a painting

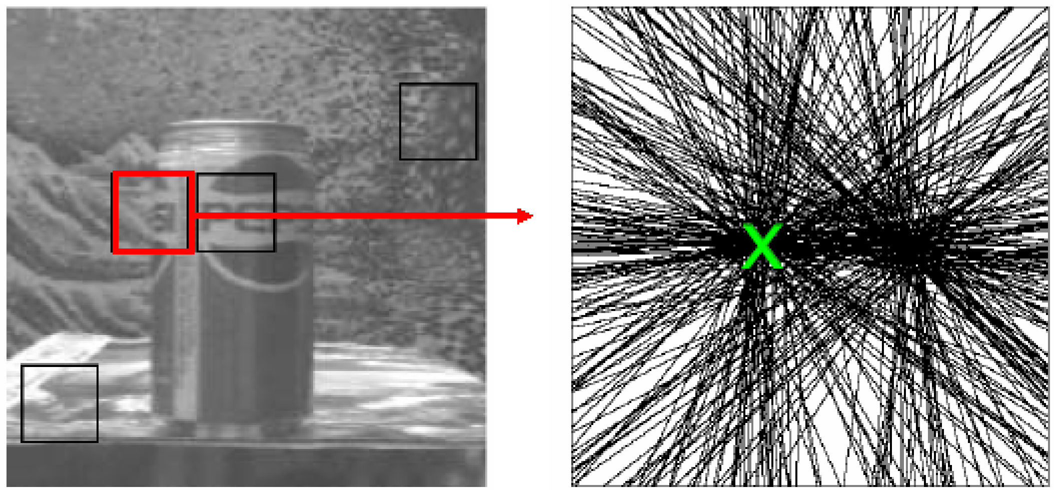
Adapted from Michael Black, Brown University

Applications of Optical flow

Use optical flow to compute correspondence between different camera views. Allows smooth interpolation between views.

Adapted from Michael Black, Brown University

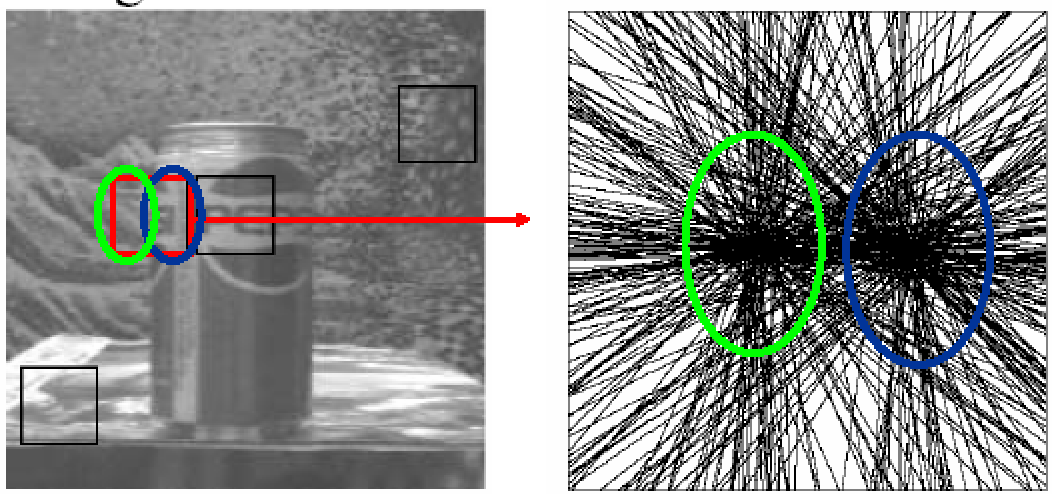
Multiple Motions



Find the dominant motion while rejecting outliers.

Multiple Motions

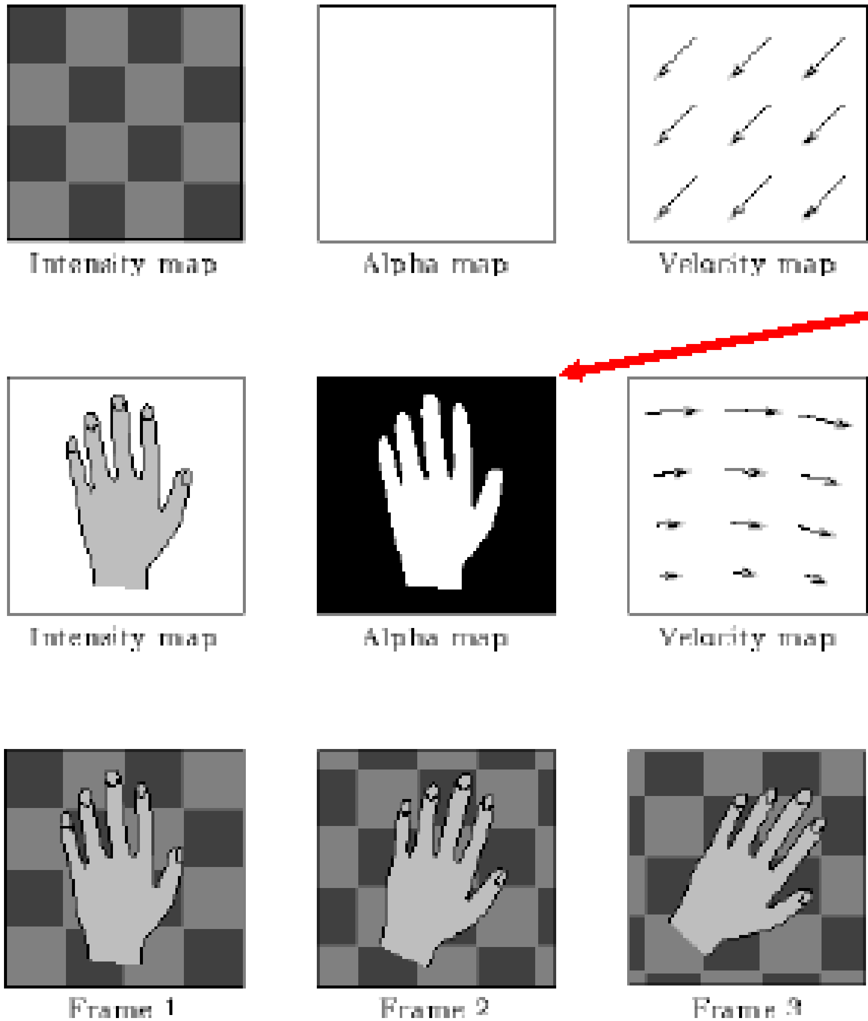
"What goes with what?"



The constraints at these pixels all "go together."

Adapted from Michael Black, Brown University

Layered Representation

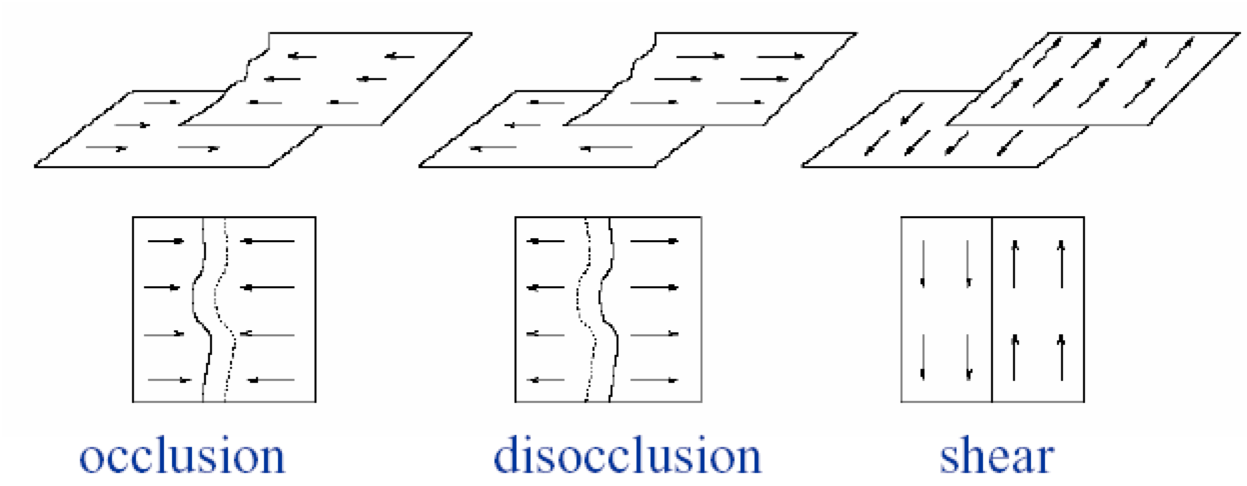


segmentation

Wang & Adelson

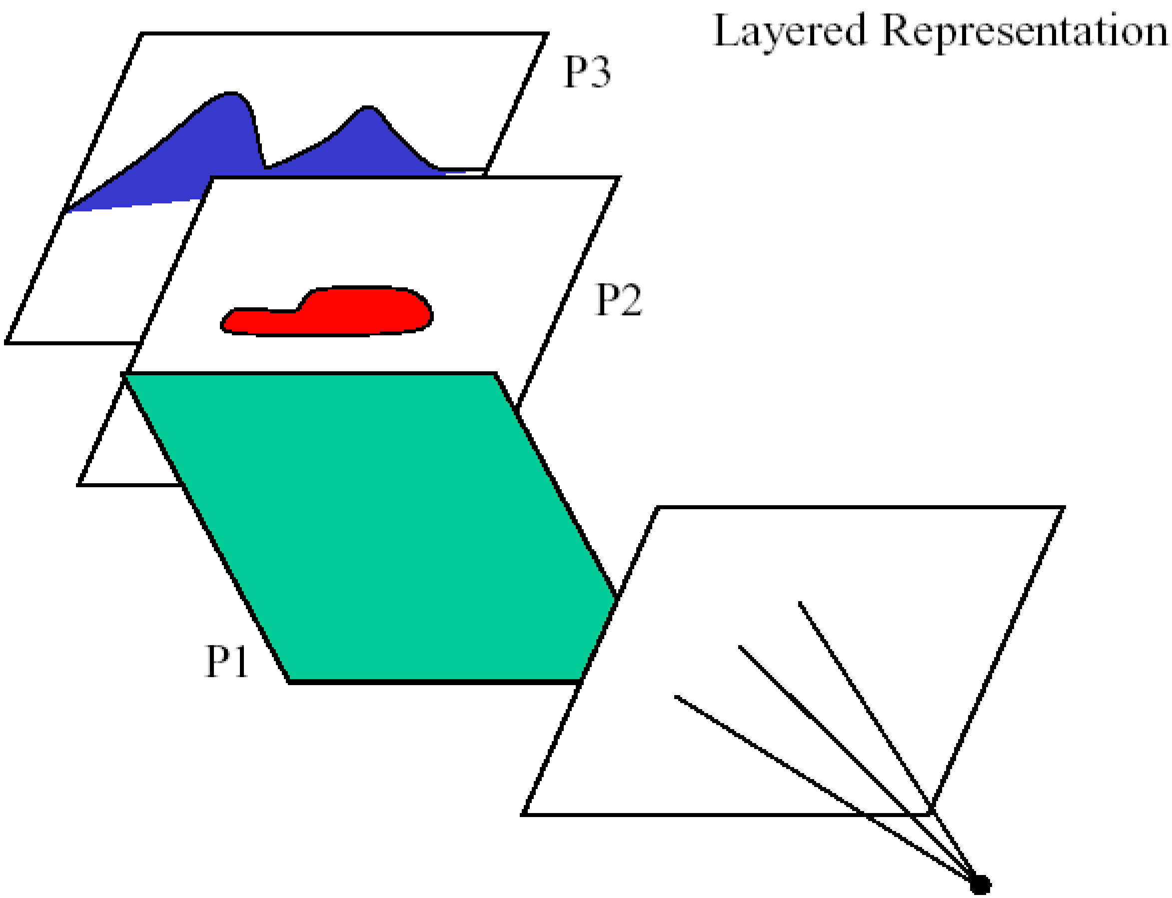
Adapted from Michael Black, Brown University

Multiple Motions



Multiple motions within a finite region.
Violates the assumption of Gaussian noise.

Layered Image Representation



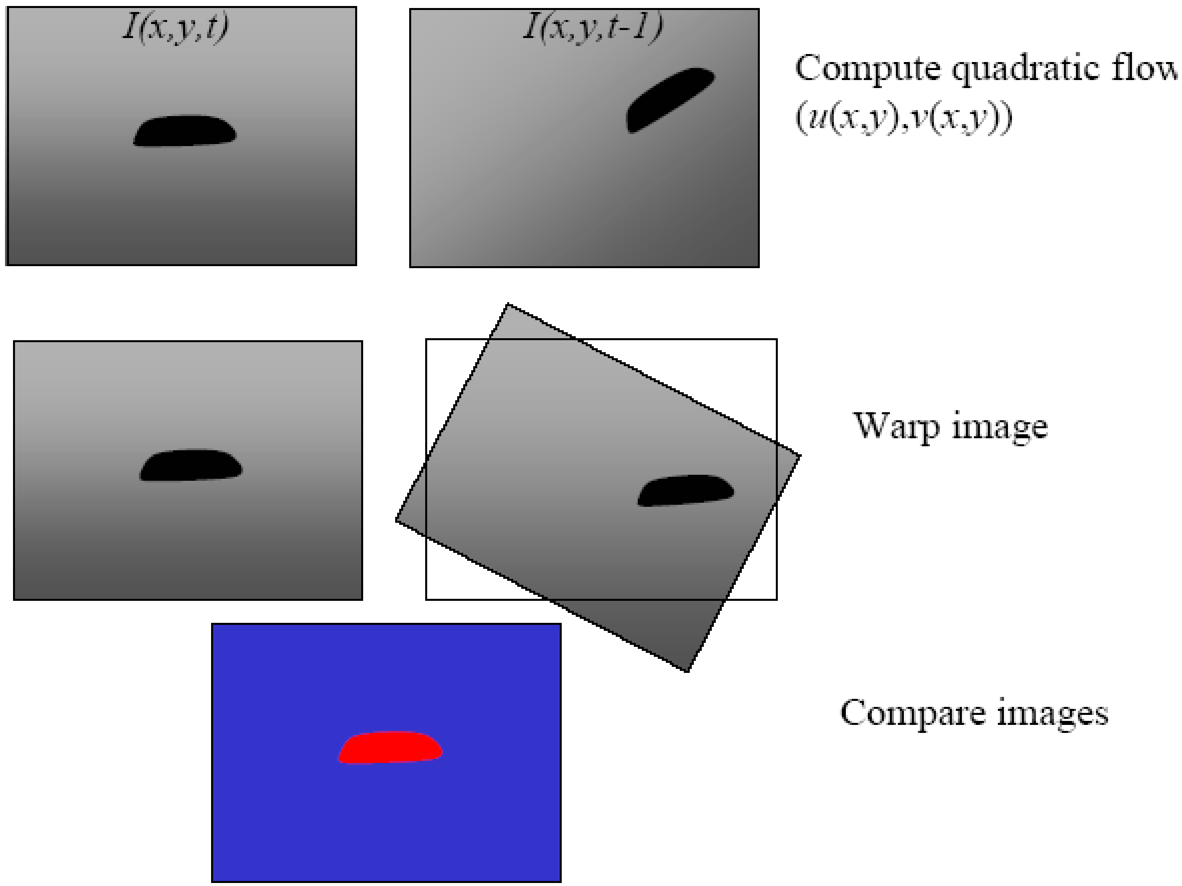
Instead of using only the egomotion estimated over the entire image, we need to segment out the various layers from the image.

Adapted from Martial Hebert

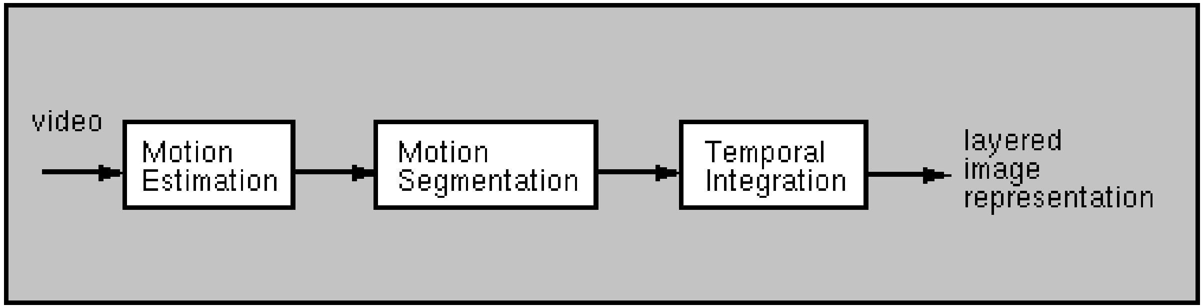
Layered Image Representation

- Estimate the dominant motion a over the entire image.
- Warp the images according to the dominant motion.
- Extract the regions that do not agree with the dominant motion.
- For each region:
    - Estimate the dominant motion again over that region
    - Extract the moving objects in the region by warping and differencing.

Dominant Motion



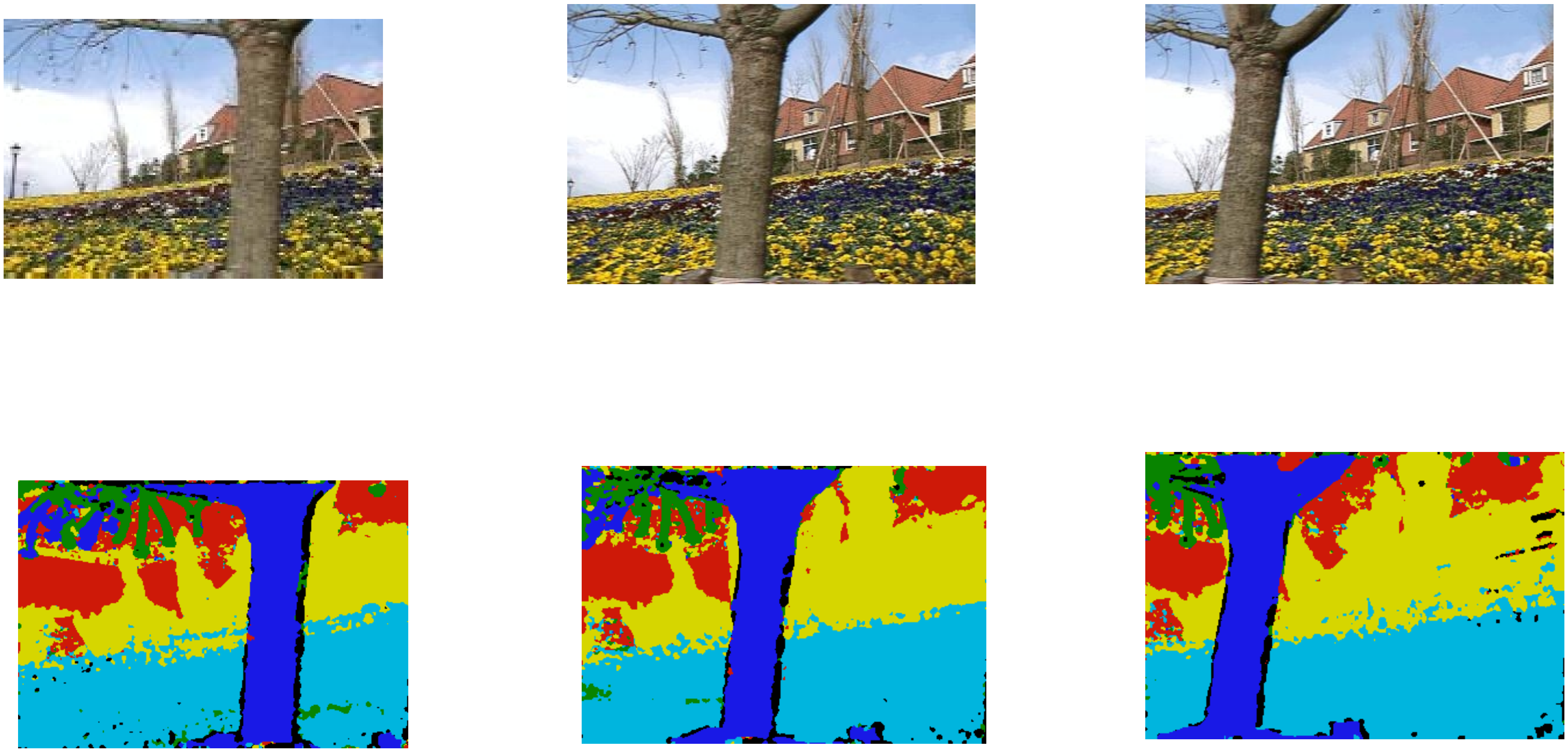
Layered Image Representation



Wang & Adelson

http://www-bcs.mit.edu/people/jyawang/demos/garden-layer/layer-demo.html

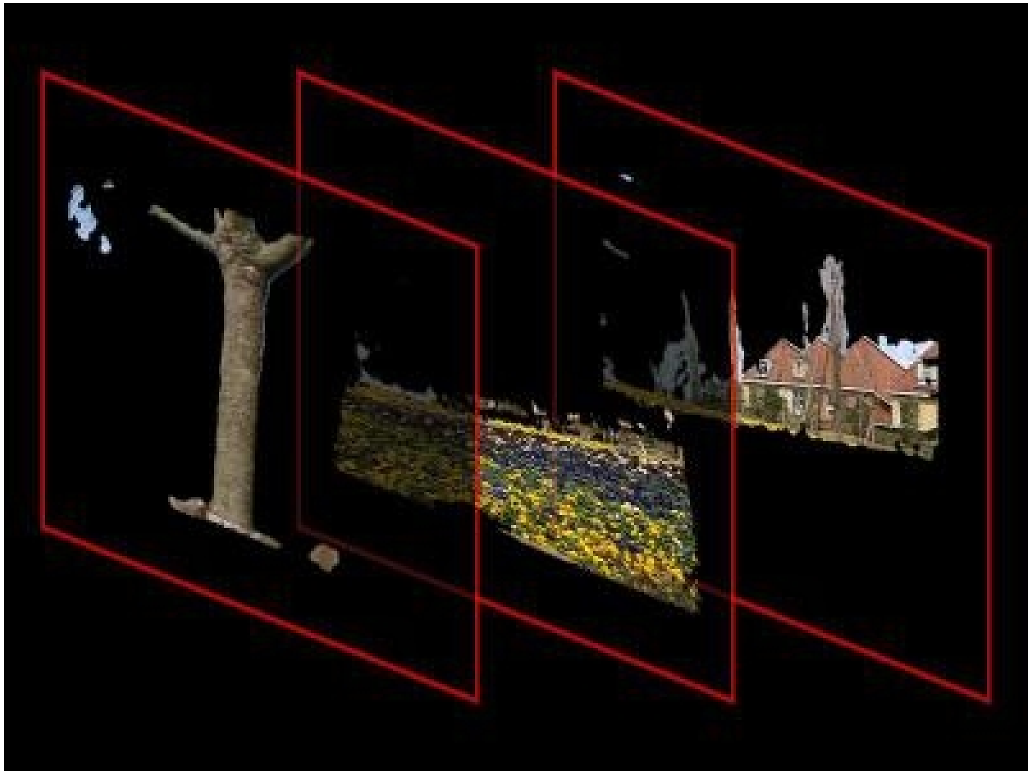
Layered Image Representation



http://www-bcs.mit.edu/people/jyawang/demos/garden-layer/layer-demo.html

Wang & Adelson

Layered Image Representation



http://www-bcs.mit.edu/people/jyawang/demos/garden-layer/layer-demo.html

Wang & Adelson

Layered Image Representation - Application

video editing and video manipulation

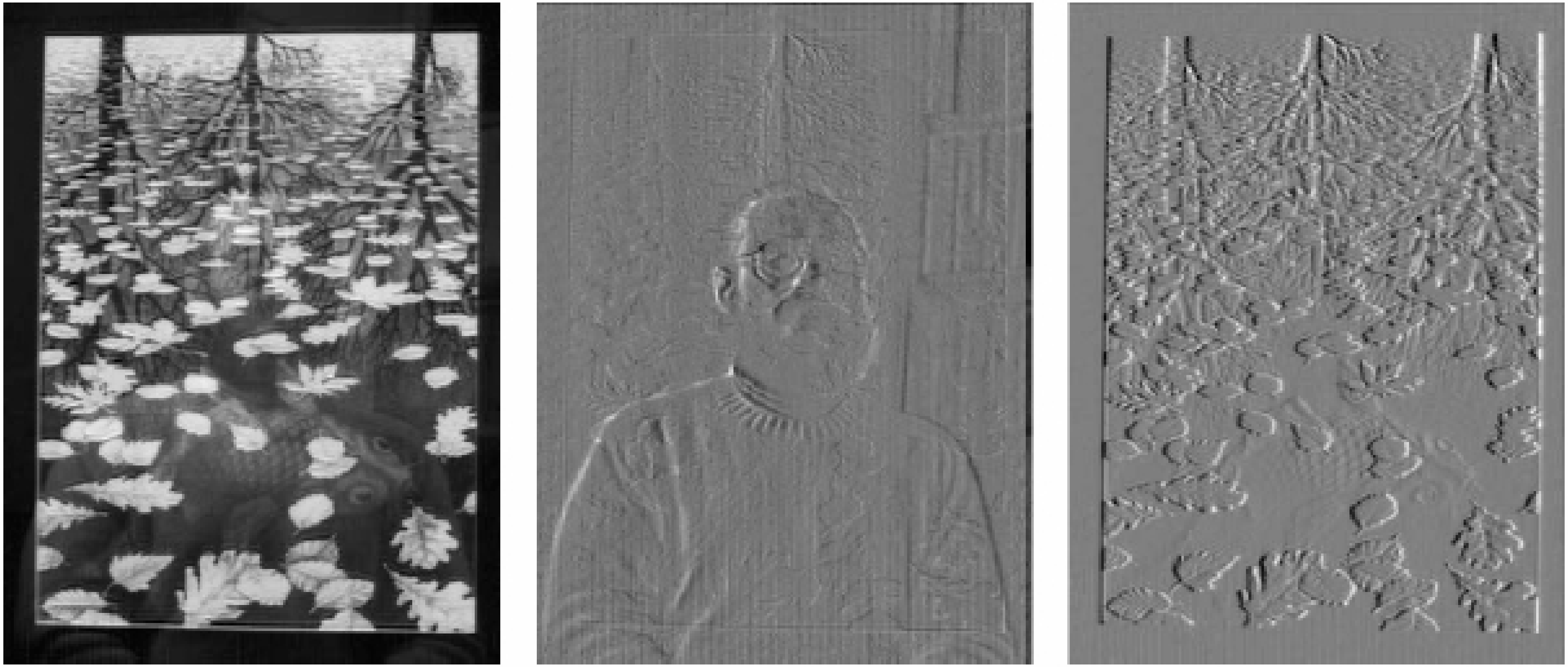
http://www-bcs.mit.edu/people/jyawang/demos/garden-layer/layer-demo.html

Wang & Adelson

Transparency

Adapted from Michael Black, Brown University

Transparency



Adapted from Michael Black, Brown University

Background Subtraction

It is important to segment the moving objects from the background either when viewing a scene from a fixed camera or after stabilization of the camera motion.

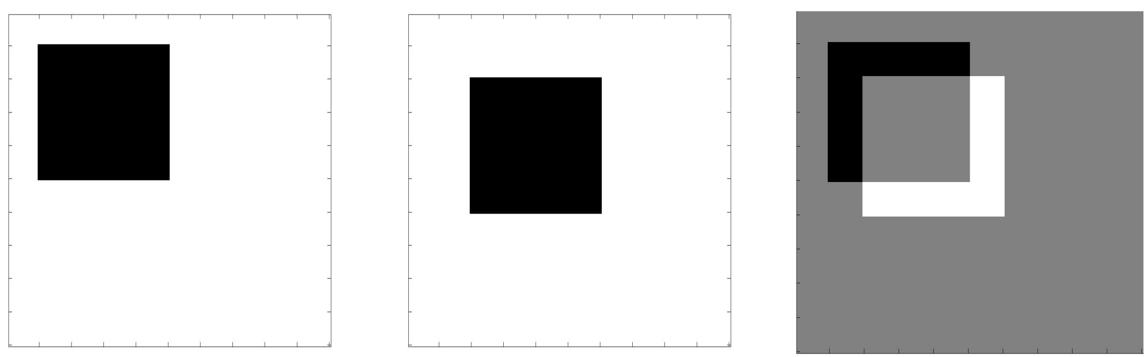
very important in many applications: surveillance, intelligent rooms, compression (MPEG-7), etc.

$$|I(x,y) - I_b(x,y)| > \tau \text{ ??}$$

Adapted from Martial Hebert, CMU

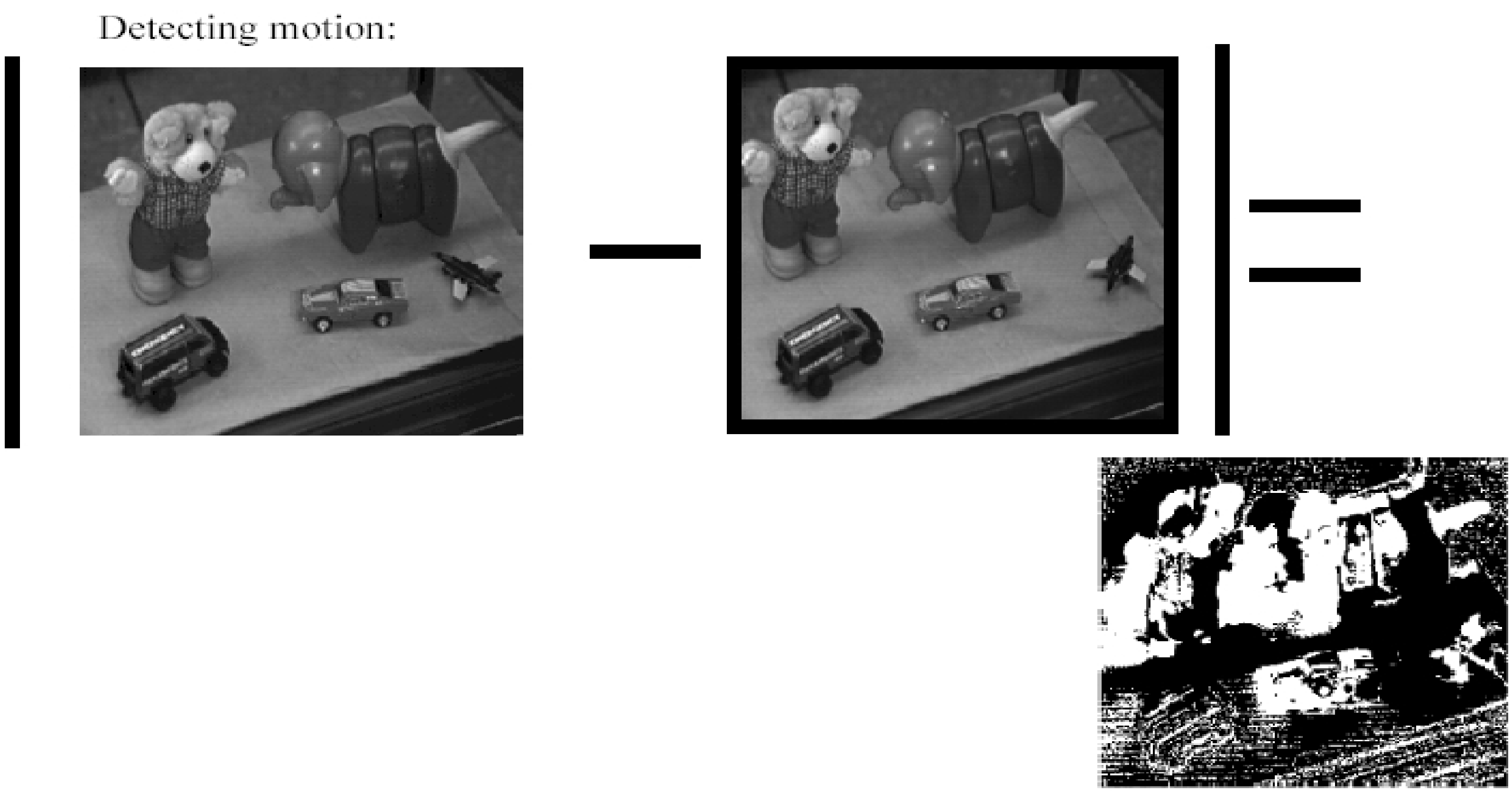
Background Subtraction

- Even simple image differencing provides an edge detector for the silhouettes of texture-free objects moving over any static background.



$I(x,y,t)$ $\quad$ $I(x,y,t+1)$ $\quad$ $I(x,y,t+1) - I(x,y,t)$

Motion in real images



Adapted from G. Hager, JHU

Motion in real images

−

> 50

Candidate areas for motion

Adapted from G. Hager, JHU

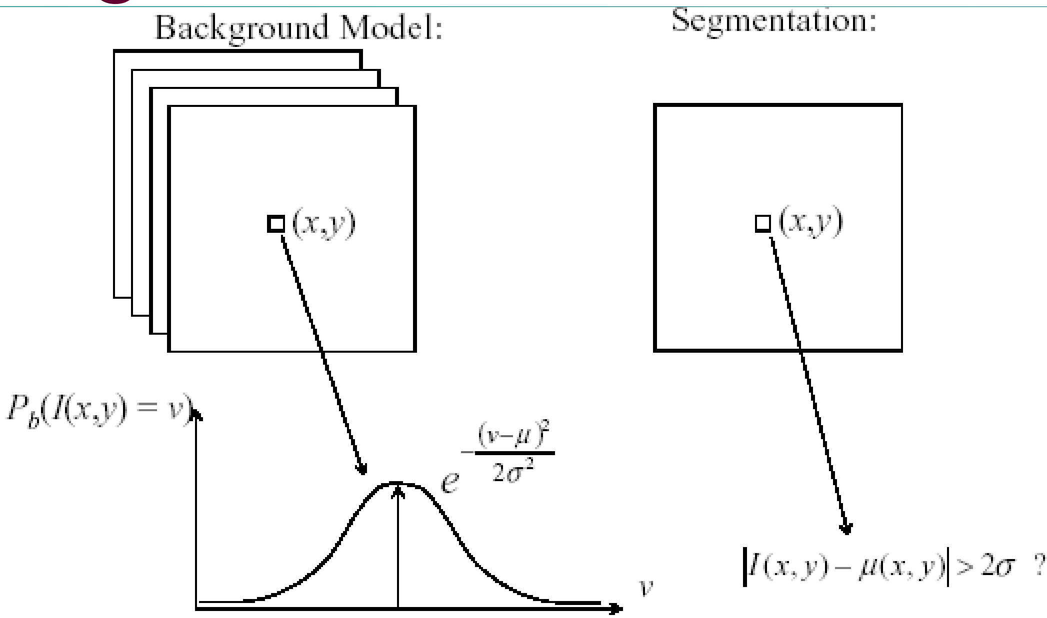
Background Subtraction

The first simple idea would be to identify those pixels that are on moving objects by simply thresholding the difference of intensity $I_t(x,y) - I_{t-1}(x,y)$ between the pixel values at times t and $t-1$, or to threshold the difference between the current image and the background image $I(x,y) - I_g(x,y)$.

This approach will not work in general because of the key problems in motion segmentation. In particular the changing background: On a pixel by pixel basis, the background is never stationary. The pixel values change constantly due to gradual changes in illumination, and small motion in the environment (for example tree leaves in the wind.)

Adapted from Martial Hebert, CMU

Background Model



$$\mu(x,y) = \frac{1}{T}\sum_t I_t(x,y)$$

$$\sigma^2(x,y) = \frac{1}{T}\sum_t (I_t(x,y) - \mu)^2$$

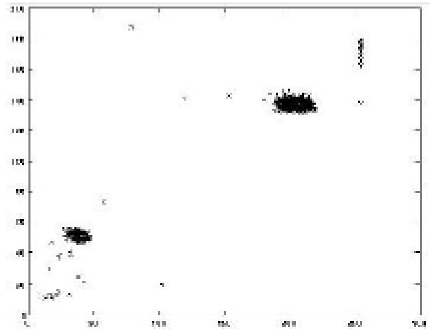
The main problem is that each pixel in the background varies in a different way over time. We need to represent the variation pattern of each pixel as computed from observing sequences of training images from the background. We'll assume for now that we have a set of T images It of the background to use for training. We can characterize the variation of each pixel over time by a probability distribution *P*b, meaning that *P*b($I(x,y) = v$) is the probability that pixel (x,y) has value v if it is in the background.

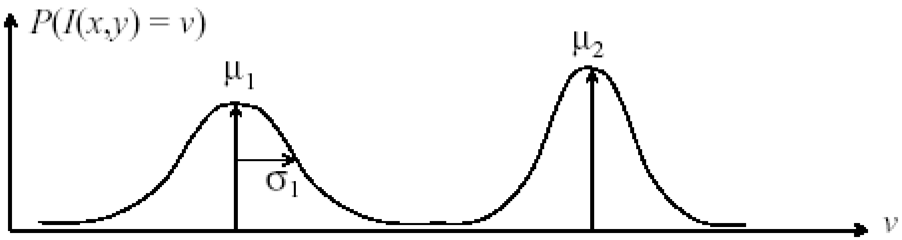
The background is modeled on a pixel-by-pixel basis by computing at each pixel: the mean m of the pixel value over time, and the standard deviation s. Each pixel is then assumed to have a normal (Gaussian) distribution N(μ,σ).

A pixel in the new image is classified as a foreground pixel if $|I(x,y) - \mu| > k\sigma$. k controls the confidence with which we want to detect foreground pixels. k = 3 implies 99% confidence.

Adapted from Martial Hebert, CMU

Mixture Models





$$P_b(I(x,y)=v)=\sum_i w_i e^{-\frac{(v-\mu_i)^2}{2\sigma_i^2}}$$

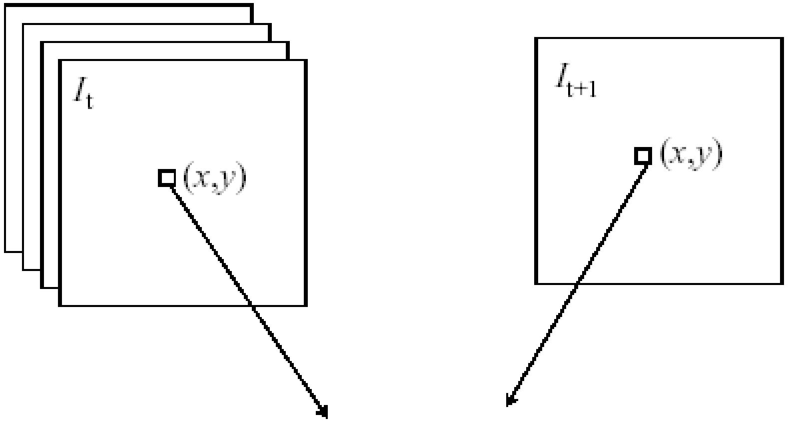
The problem with this approach is that the normal distribution may be too simplistic of a model. For example, because of small motions in the background scene, a given pixel may be on, for example, three different regions. In that case the sigma of the pixel values may be artificially large and lead to wrong segmentation results. A improved approach is to represent the distribution of each pixel in the background as a weighted sum of normal distributions (called a "mixture of Gaussians"). Each Gaussian is weighted according to the frequency with which it explains the background.

$$P(I(x,y)=v)=\sum_i w_i e^{-\frac{(v-\mu_i)^2}{2\sigma_i^2}}$$

Pixels that are more than two standard deviations away from all the components of the mixture model are classified as background pixels.

Adapted from Martial Hebert, CMU

Linear Prediction and Adaptation



model at time t+1 = (1-α) (model at time t) + α (new data)

α = "learning rate"

Gaussian model:

$$\rho = \alpha P_b(v_{t+1})$$
$$\mu_{t+1} = (1-\rho)\mu_t + \rho v_{t+1}$$
$$\sigma_{t+1}^2 = (1-\rho)\sigma_t^2 + \rho(v_{t+1} - \mu_{t+1})^2$$

the background may change gradually over time, for example, because of gradual changes in lighting. As a result, the model computed from initial training images is no longer valid after observing the scene for some time.

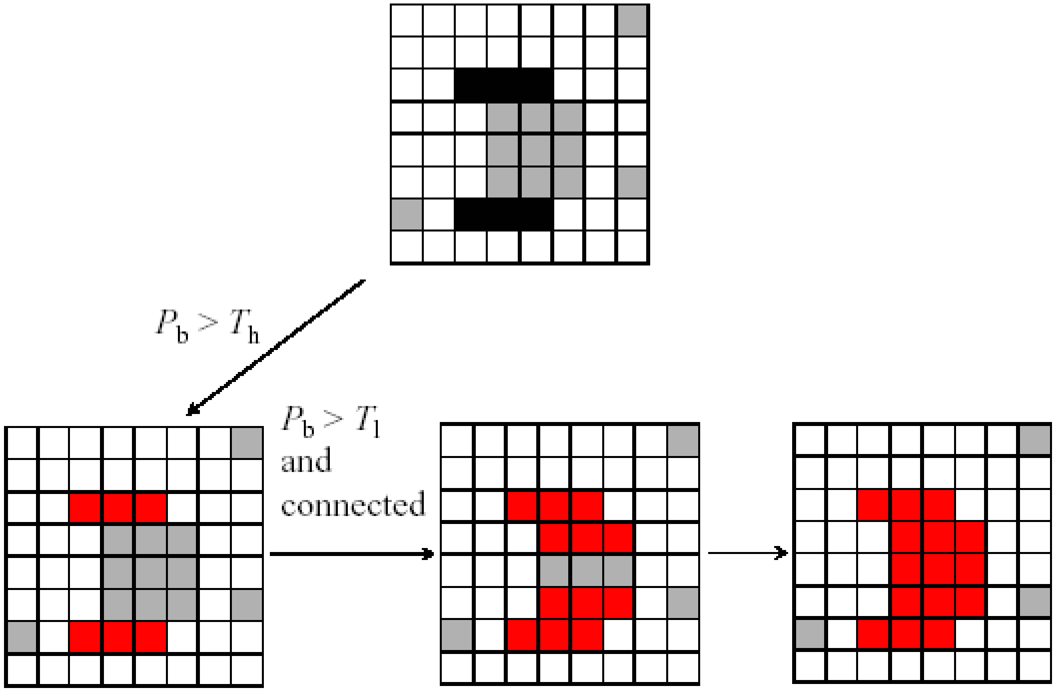
Therefore, the background model needs to be updated over time. In general, a linear update isused, of the form:

model at time t+1 = (1-α) (model at time t) + α (new data)

α is the learning rate. If α is close to 1, the model is re-created everytime new data arrives; if α is close to 0, the background model changes very slowly over time.

Adapted from Martial Hebert, CMU

Hysteresis Thresholding



moving objects typically form connected regions in the image. It is important to perform the segmentation in a way that favors connected regions.

The previous techniques tend to generate fragmented regions because some pixels that are part of the moving object may have values that are too close to the background values and are therefore not classified as foreground pixels.

Pixels are first classified as foreground using a conservative threshold Th. Those pixels are part of foreground objects with high confidence. Then, any pixel that is connected to a foreground pixel and satisfies a less conservative threshold Tl is also classified as foreground. This approach tends to generate connected regions and to compromise between detection of spurious foreground points and fragmentation of regions.

Adapted from Martial Hebert, CMU