MAPI – Computer Vision

Motion Analysis and Object Tracking

Motion Analysis

- Practical computer vision
  - motion analysis requires the storage and manipulation of *image sequences* rather than single images.
  - Image sequences can be stored as lists of arrays, vectors of arrays or 3-D arrays.
  - Each image in a sequence is often called a *frame*, and the time which elapsed between each frame being digitized the *frame interval*.
  - The inverse of the frame interval, that is, the number of frames per second, is the *frame rate*.

- The main issue in motion analysis is to recover information about the motion in the scene from the variation of the intensities I over time.

- For many real applications such as robot control
  - the ideal is to process each frame as it arrives (in *real time* or *online*)
- Scope techniques
  - Background Subtraction
  - Optical Flow
  - Object Tracking

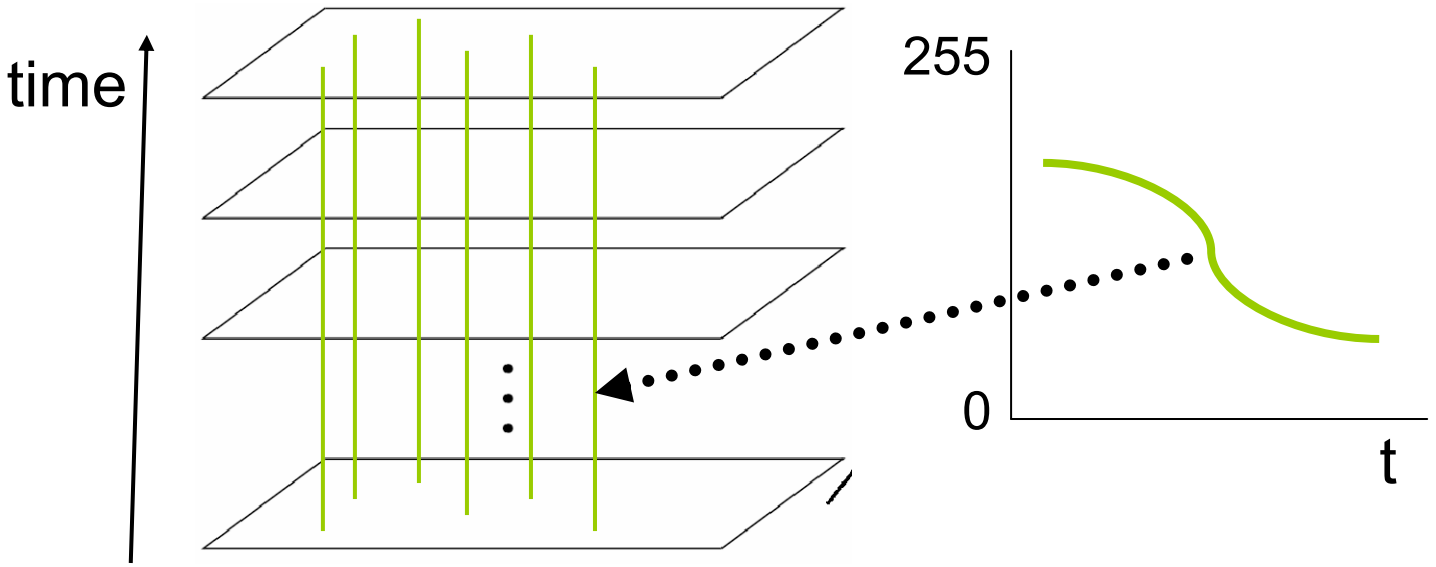
Motion Analysis

- A lot of information can be extracted from time varying sequences of images:
  - camouflaged objects are only easily seen when they move.
  - the relative sizes and position of objects are more easily determined when the objects move.
  - Even simple image differencing provides an edge detector for the silhouettes of texture-free objects moving over any static background.

- The analysis of visual motion divides into two stages:
  - the measurement of the motion, and
  - 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.
- There are two types of motion to consider:
  - movement in the scene with a static camera,
  - and 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 and specularities effects need to be dealt with

Motion Analysis



- video data as a spatio-temporal volume
  - If camera is stationary, each line through time corresponds to a single ray in space
  - We can look at how each ray behaves

Motion Analysis

Background Subtraction

- Background subtraction is a commonly used class of techniques for segmenting out objects of interest in a scene for applications such as:
  - Surveillance
  - Robot vision
  - Object tracking
  - Traffic applications
  - Human motion capture
  - Augmented reality

Motion Segmentation:
Special case for fixed background

Background Subtraction

- It involves comparing an observed image with an estimate of the image if it contained no objects of interest.
- The areas of the image plane where there is a significant deference between the observed and estimated images indicate the location of the objects of interest.
- The name *background subtraction* comes from the simple technique of subtracting the observed image from the estimated image and thresholding the result to generate the objects of interest.

Background Subtraction

- Generic Algorithm
  - Create an image of the stationary background by averaging a long sequence
  - Difference a frame from the known background frame
- Motion detection algorithms such as these only work if the camera is stationary and objects are moving against a fixed background

Background Subtraction

- Three important issues
  - foreground detection
    - how the object areas are distinguished from the background;
  - background maintenance
    - how the background is maintained over time;
  - post-processing
    - how the segmented object areas are post-processed to reject false positives,

Background Subtraction

One video frame

Estimated background

Difference Image

Background Subtraction

- Difficulty
  - Maintain performance under a variety of deferent operating conditions.
    - Sensitive to the background changes
      - Acquisition noise
      - Illumination drifts
      - Shadows
      - ...

Background Subtraction

- A largely unsolved problem…

One video frame | Estimated background | Difference Image

Background Subtraction

- Several algorithms have been developed
  - For a variety of different operations conditions
    - **Heikkila and Olli**
    - **Adaptive Mixture of Gaussians**
    - Pfinder
    - W4
    - LOTS
    - Halevy
    - Cutler
    - **Codebook**
    - **Wallflower**
    - …

Background Subtraction

- J. Heikkila and O. Silven: A real-time system for monitoring of cyclists and pedestrians in: Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado (Jun. 1999) pp. 74-81.

A pixel is marked as foreground if

$$\left|\mathrm{I}_t - \mathrm{B}_t\right| > \tau$$

where τ is a “predefined" threshold.

The thresholding is followed by morphological closing with a 3x3 kernel and the discarding of small regions.

Background Subtraction

- J. Heikkila and O. Silven: A real-time system for monitoring of cyclists and pedestrians in: Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado (Jun. 1999) pp. 74-81.

The background update is

$$\mathrm{B}_{t+1} = \alpha \mathrm{I}_t + (1-\alpha)\mathrm{B}_t$$

where α is kept small to prevent artificial “tails” forming behind moving objects.

Background Subtraction

- J. Heikkila and O. Silven: A real-time system for monitoring of cyclists and pedestrians in: Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado (Jun. 1999) pp. 74-81.

Two background corrections are applied:

- If a pixel is marked as foreground for more than m of the last M frames, then the background is updated as $B_{t+1} = I_t$.
  - This correction is designed to compensate for sudden illumination changes and the appearance of static new objects.
- If a pixel changes state from foreground to background frequently, it is masked out from inclusion in the foreground.
  - This is designed to compensate for fluctuating illumination, such as swinging branches.

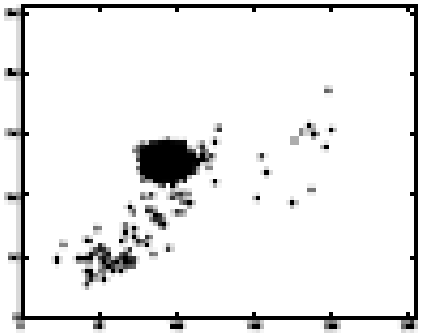
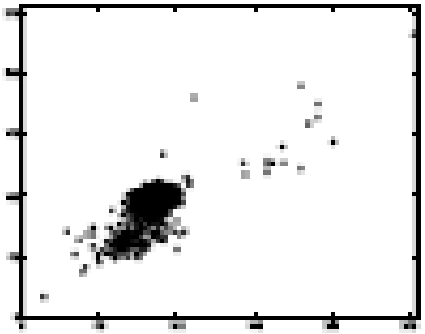
Background Subtraction

- Adaptive Mixture of Gaussians (MoG)
  - Each background pixel is modeled separately by a mixture of *K* Gaussians.
    - Typically values for *K:* 3, 4, 5

  - W. E. L. Grimson, C. Stauer, R. Romano and L. Lee: Using adaptive tracking to classify and monitor activities in a site in: Computer Vision and Pattern Recognition Santa Barbara, California (Jun. 1998) pp. 1-8.
  - Y. Ivanov, C. Stauer, A. Bobick and W. E. L. Grimson: Video surveillance of interactions in: Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado (Jun. 1999) pp. 82-90.
  - C. Stauer and W. E. L. Grimson: Adaptive background mixture models for real-time tracking in: Computer Vision and Pattern Recognition Fort Collins, Colorado (Jun. 1999) pp. 246-252.

Background Subtraction

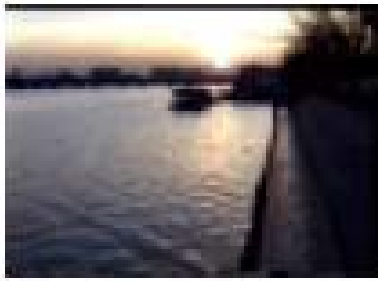
- Adaptive Mixture of Gaussians (MoG)
- Why MOG
  - some difficulties involved in background subtraction
  - (R,G) scatter plot of the same pixel taken 2 minutes apart
    - This would require 2 threshold for the segmentation at different times

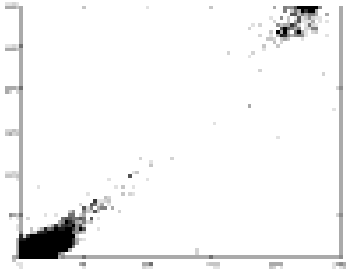




Background Subtraction

- Adaptive Mixture of Gaussians (MoG)
- Why MOG
  - some difficulties involved in background subtraction

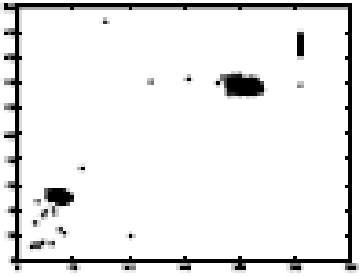
  - (R,G) scatter plot of the same pixel
    - Bi-Model distribution
      - This would require the application of 2 thresholds for each new frame





Resulting from specularities on the surface of the water

Resulting from monitor flicker



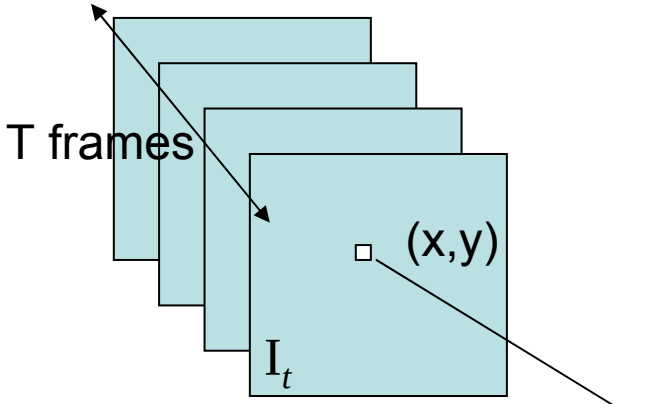
Background Subtraction

- Adaptive Mixture of Gaussians

- Each pixel as an independent statistical process
  - Each pixel intensity is observed during n initial frames
  - The observed samples is then optimally fit with a mixture of K Gaussians
- This reflects
  - the expectation that samples f the same scene point are likely to display normal noise distribution
    - Ex. Acquisition noise
  - the expectation that more than one process may be observed over time
    - Ex. Branches rusting in the wind

Background Subtraction

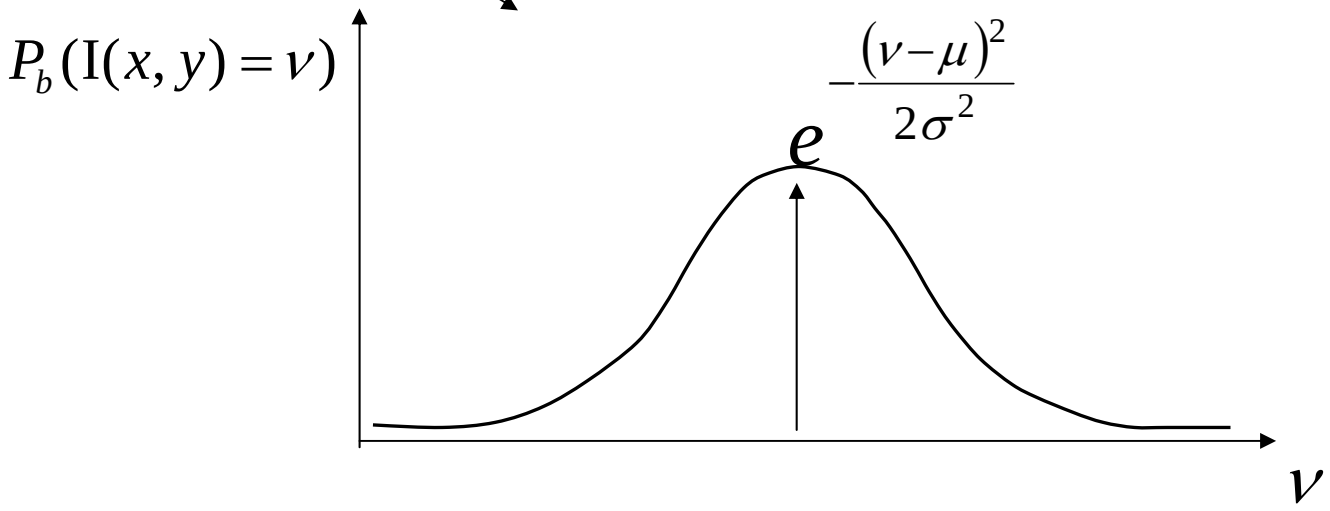
- Adaptive Mixture of Gaussians



Background model

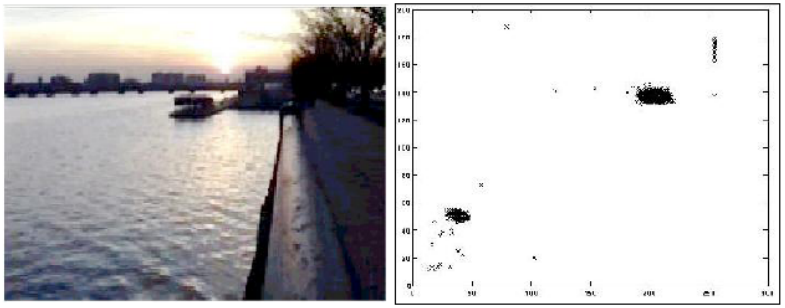
$$\mu(x,y) = \frac{1}{T}\sum_{t} I_t(x,y)$$

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



Background Subtraction

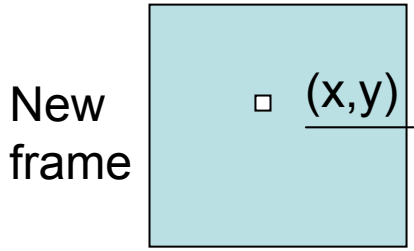
- Adaptive Mixture of Gaussians



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

Background Subtraction

- Adaptive Mixture of Gaussians



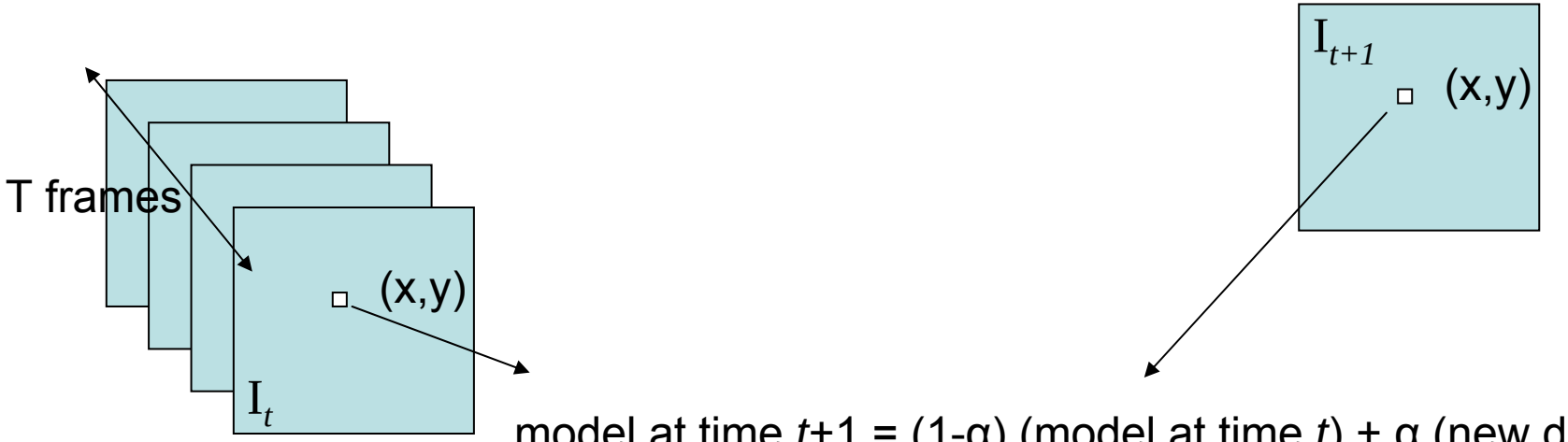
Segmentation - moving objects:

$$\left|\mathrm{I}(x,y)-\mu(x,y)\right|>2\sigma$$

Estimated background
The most probable background image
dominant Gaussian mean for each pixel's mixture model

Background Subtraction

- Adaptive Mixture of Gaussians



model at time *t*+1 = (1-α) (model at time *t*) + α (new data)
With α= “learning rate”

Update Gaussian model

$$\rho = \alpha P_b(\nu_{t+1})$$

$$\mu_{t+1} = (1-\rho)\mu_t + \rho\nu_{t+1}$$

$$\sigma^2_{t+1} = (1-\rho)\sigma^2_t + \rho(\nu_{t+1} - \mu_{t+1})^2$$

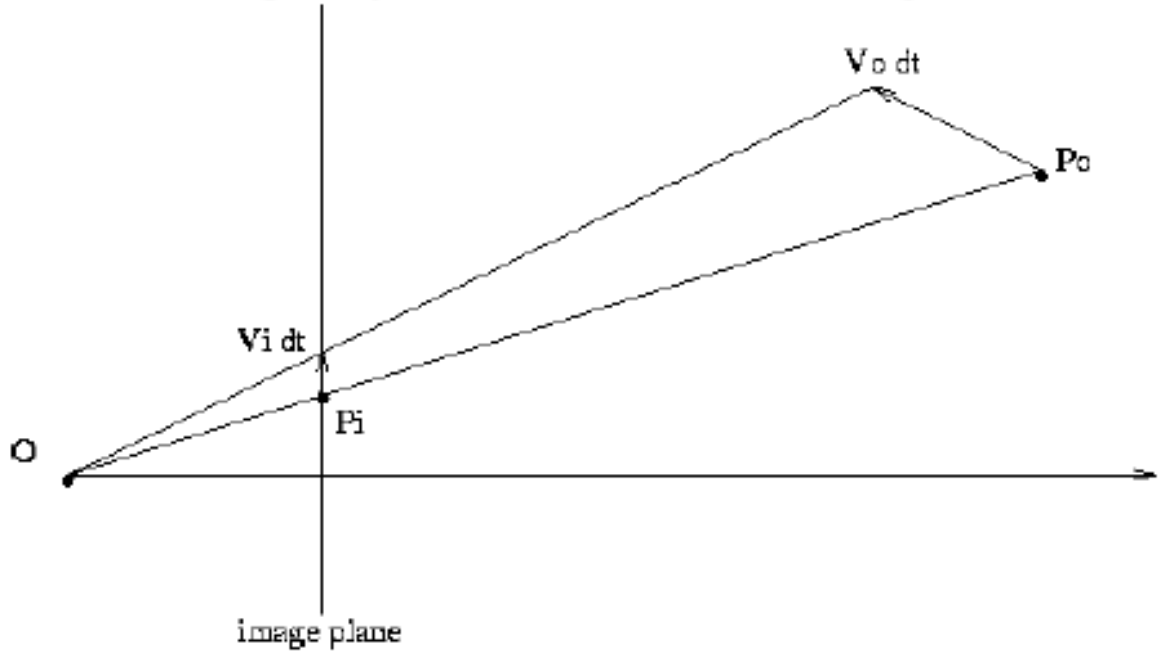
Background Subtraction

- Adaptive Mixture of Gaussians

Motion

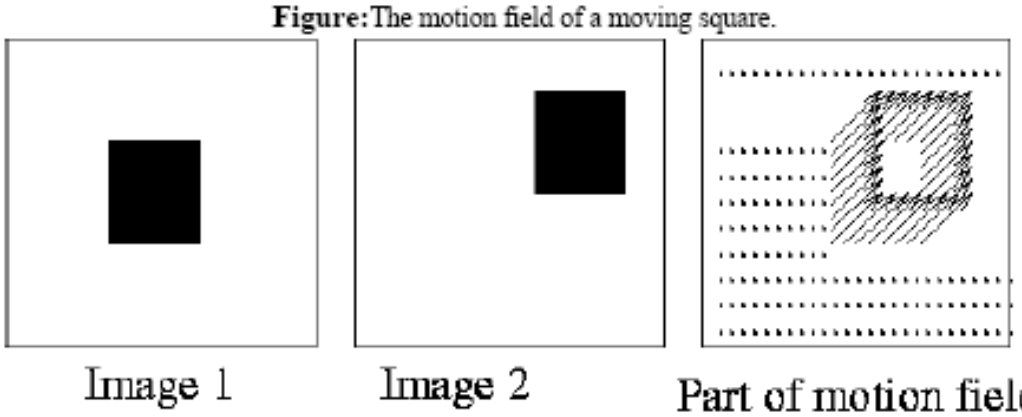
- Motion field
  - When an object moves in front of a camera, there is a corresponding change in the image.
  - if a point *po* on a object moves with a velocity v_o, then the imaged point p_i can be assigned a vector v_i to indicate its movement on the image plane. The collection of all these vectors forms the *motion field*.

Figure: Object motion creates a motion field in the image.



Motion

- 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).
  - Thus, in the case of divergence we have forward motion of the camera, and in the case of convergence, backwards motion.

Motion

- 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 we will assume that optical flow corresponds to the motion field.

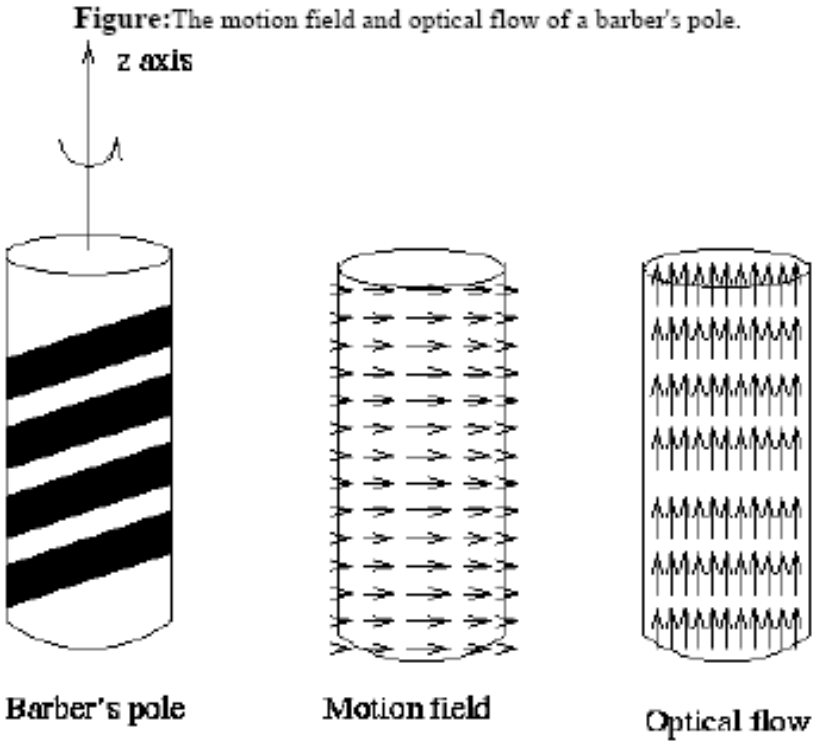


Figure: The motion field and optical flow of a barber's pole.

Motion

- Optical Flow
  - One problem is that we are only able to measure the component of optical flow that is in the direction of the intensity gradient.
  - We are unable to measure the component tangential to the intensity gradient.
  - This problem –> aperture problem

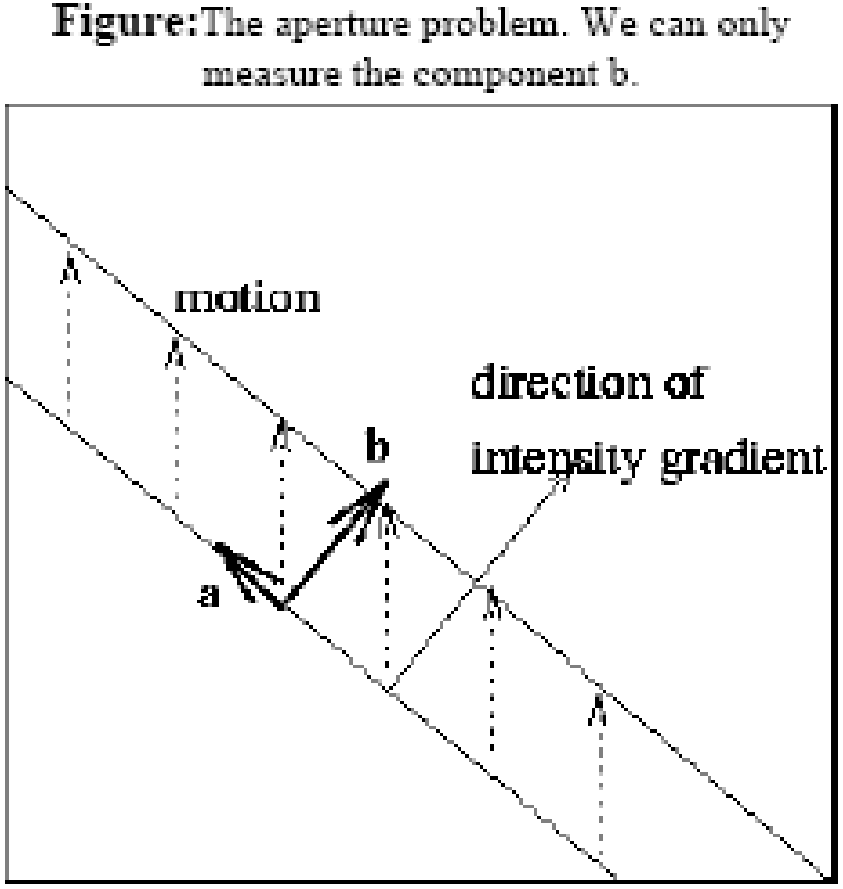


Figure: The aperture problem. We can only measure the component b.

Motion

- Optical Flow
  - Denote the intensity by $I(x,y,t)$.
    - A function of three variables as we now have *spatiotemporal* variation in our signal.
  - To see how I changes in time, we differentiate with respect to t:

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}$$

    - assuming that the image intensity of each visible scene point is unchanging over time (for example, shadows and illuminations are *not* changing due to any motion):

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

Motion

- Optical Flow
  - This assumption of *brightness conservation* implies that :

$$\frac{dI}{dt} = 0 \quad \Rightarrow \quad I_x u + I_y v + It = 0$$

where the partial derivatives of *I* are denoted by subscripts, and *u* and *v* are the *x* and *y* components of the optical flow vector.

  - This last equation is called the *optical flow constraint equation* since it expresses a constraint on the components *u* and *v* of the optical flow.

Motion

- Optical Flow
  - The optical flow constraint equation can be rewritten as

$$(I_x, I_y).(u, v) = -I_t$$

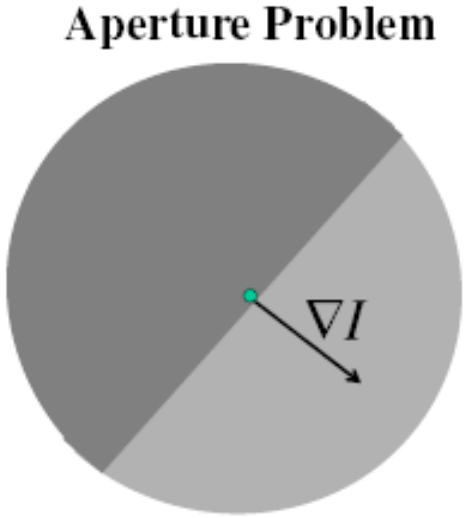
  - Thus, the component of the image velocity in the direction of the image intensity gradient at the image of a scene point is

$$(u, v) = \frac{-I_t}{\sqrt{I_x^2 + I_y^2}}$$

Motion

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

- 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.



Motion

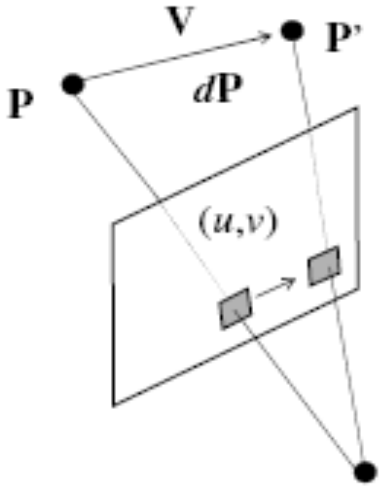
- Optical Flow
  - Aperture problem
    - We could try to use an entire region of the image W (rectangular window) and minimize:

$$\sum_{(x,y)\in W}\left(u(x,y)I_x + v(x,y)I_y + I_t\right)^2$$

    - However, the problem remains under-constrained.

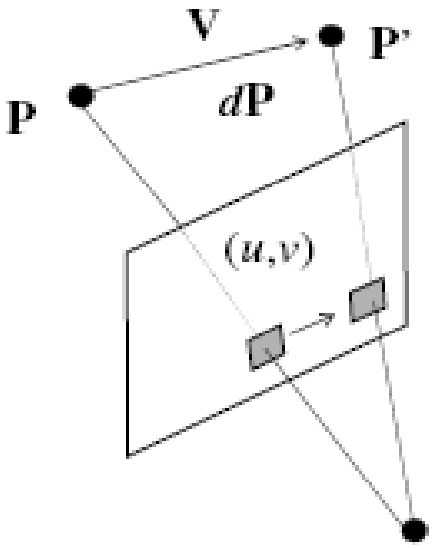
- Consider 2 images separated by a small rotation and translation motion (6 parameters).
  - The projection of a point in the image is given by $x=X/Z$ and $y=Y/Z$.
  - Taking the derivative with respect to time:

$$u = \frac{V_x - xVz}{Z} \qquad v = \frac{V_y - yVz}{Z}$$



Motion

- Optical Flow

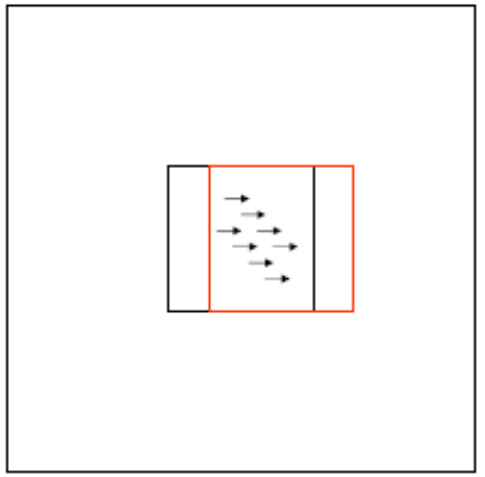


  - **V** is the velocity of the point in space
  - *u* and *v* depend on the UNKNOWN depth Z
  - This illustrates the parallax effect
    - The image motion is a function of:
      - Motion in space
      - Depth
  - It is therefore impossible to recover (*u,v*) in the general case

Motion

- Optical Flow
  - There are two main approaches to reconstructing three dimensional motion from motion in the image plane:
    - Convert the motion problem to a stereo problem and find the correspondence between a number of points in the image at time *t* to the image at time $t+\delta t$.
    - Computer the optical flow and use its geometrical properties to deduce three dimensional information about the scene and the motion.

Motion



- Optical Flow
  - Constant Flow – translation
    - Suppose that we assume that the flow is constant (u,v) over a small region W in the image. 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:

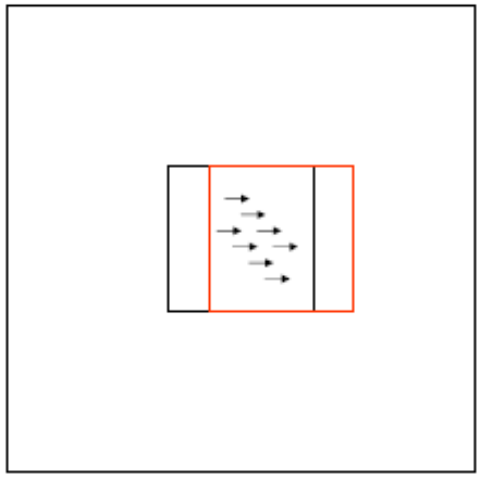
$$E(u,v) = \sum_{(x,y)\in W} \left(uI_x(x,y) + vI_y(x,y) + I_t \right)^2$$

    is close to 0.

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

$$\min_{u,v} E(u,v)$$

Motion



- Optical Flow
  - Constant Flow
    - *E* is a simple quadratic expression in *u* and *v*.
      The minimization can be expressed as a simple leastsquares problem:

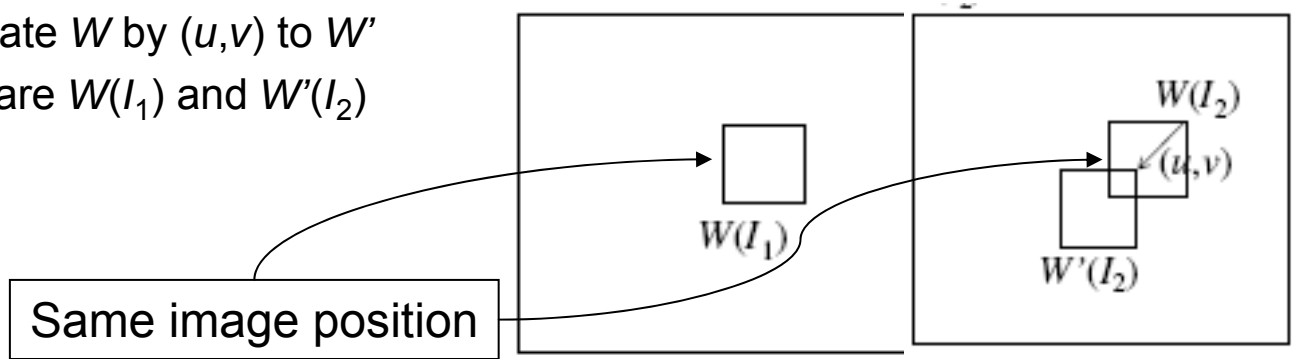
$$\mathrm{Min}\left\| \mathrm{A}\begin{bmatrix} u \\ v \end{bmatrix} + \mathrm{b} \right\|^2$$

$$\mathrm{A} = \begin{bmatrix} I_x(x_0, y_0) & I_y(x_0, y_0) \\ \vdots & \vdots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} I_t(x_0, y_0) \\ \vdots \\ I_t(x_n, y_n) \end{bmatrix}$$

Motion

- Optical Flow
  - Constant Flow
    - Feature Tracking
      - Based on the previous result, a feature tracking algorithm involves solving for the constant motion inside *W* by inverting the Harris matrix. This gives the offset (*u*,*v*) to the position in the next image.
    - Algorithm:
      - Given Images I_1 and I_2
      - Determine Feature at (x_o,y_o) in I_1
      - Window *W* at (x_o,y_o)
      - **Compute displacement in image (*u*,*v*)**
      - 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 $W'(I_2)$

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

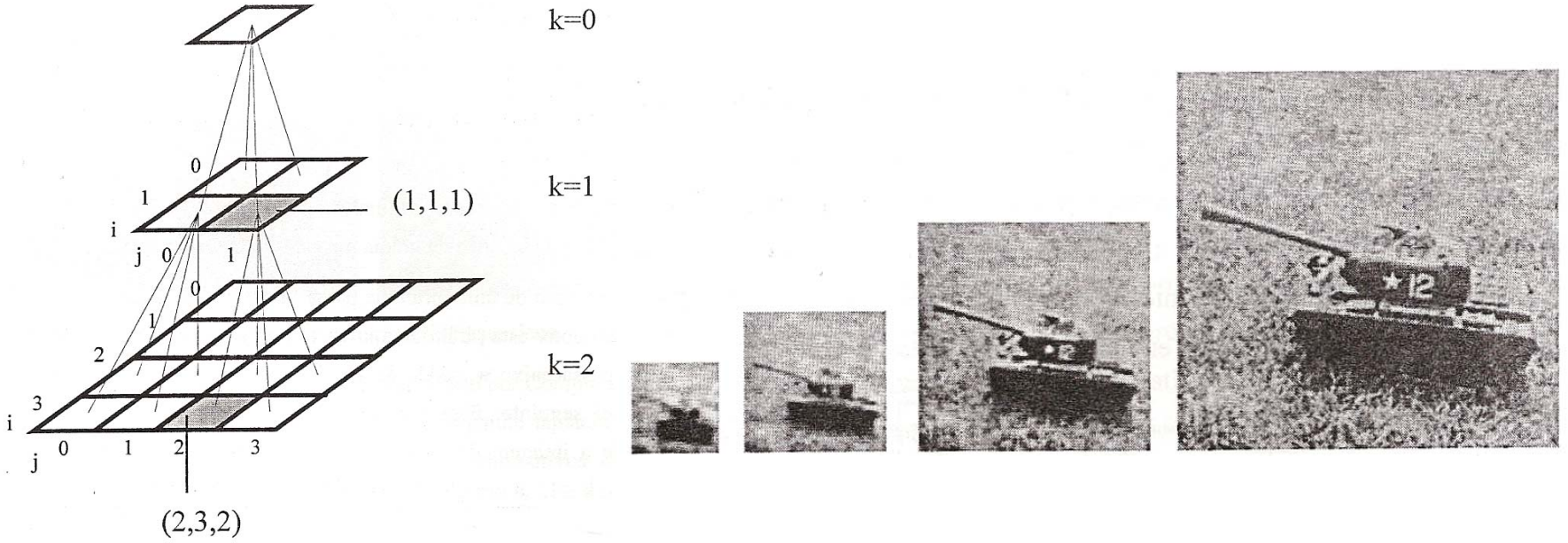


Motion

- Optical Flow
  - Constant Flow
    - Feature Tracking – improvement - iterate the same algorithm.
      - At each iteration the window *W* is shifted by the recovered motion to *W'*. The pixels $W(I_1)$ and $W'(I_2)$ are then compared (using a pattern matching technique).
      - 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) became very small or the windows match.
    - Since we are interested in the motion of the center point (x_0,y_o) 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 $I_x(x,y)$ and $I_y(x,y)$ by a weight $w(x,y)$ which is maximum at the center and tapers off at the edges of *W* (a Gaussian, for example.)
    - *Lucas-Kanade* tracker.

Motion

- Optical Flow
  - Constant Flow
    - To deal with larger motions, first reduce the images (after smoothing) and recover the motion at a coarse scale before recovering motion at full resolution – pyramid decomposition

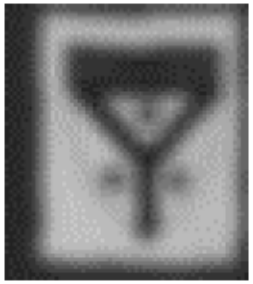


Motion

- Optical Flow
  - Constant Flow
    - Template Tracking:
      - The same idea can be used to track a fixed template through a sequence of images. In that case, I_1 is a reference template, let's call it T. Let **M** be the matrix:

$$\mathrm{M} = \begin{bmatrix} \sum T_x^2 & \sum T_x T_y \\ \sum T_x T_y & \sum T_y^2 \end{bmatrix}^{-1}$$

      - Then we can form 2 new images from the template T



Sx

Sy

$$\begin{bmatrix} S_x \\ S_y \end{bmatrix} = -M \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Motion

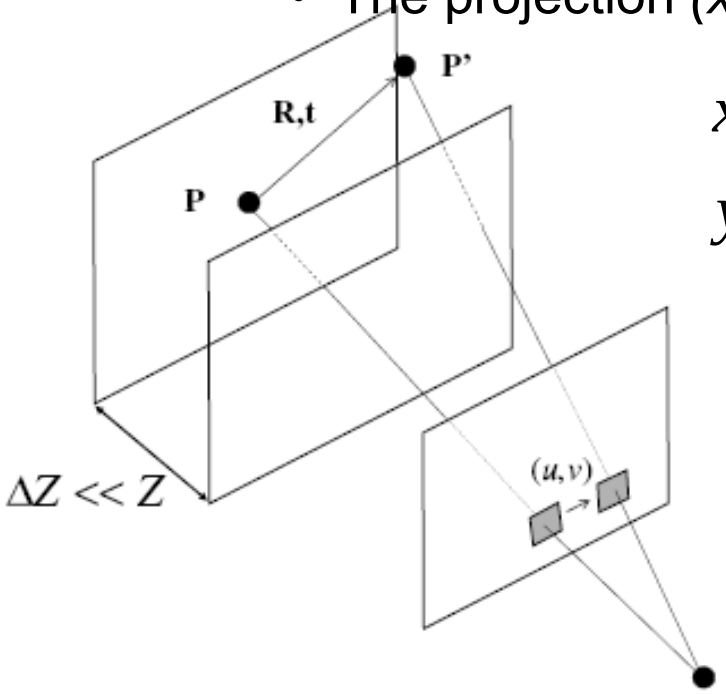
- Optical Flow
  - Constant Flow
    - Template Tracking:
      - From the previous equations, the motion (u,v) is recovered by taking the sums over the window W of:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_W S_x (I - T) \\ \sum_W S_y (I - T) \end{bmatrix}$$

      - These 2 sums are the correlation between the current image *I* and the two images *S*x and *S*y.
      - **Tracking is reduced to a simple image correlation.**
      - This suggests the **general tracking algorithm**:
        - » Pre-compute the 2 images *S*x and *S*y
        - » For a new image *I*, extract the window *W* at the current position of the template
        - » Correlate the image with *S*x and *S*y, yielding the 2 components of the image motion *u* and *v.*

Motion

- Optical Flow
  - Distant motion -> affine flow
    - If we assume now a general motion (rotation+translation) of the scene points, a point of coordinates *X,Y,Z* is transformed by a small rotation **R** and a translation **t.**
    - The projection *(x,y)* and *(x',y')* in two consecutive images are:



$$x = X / Z$$

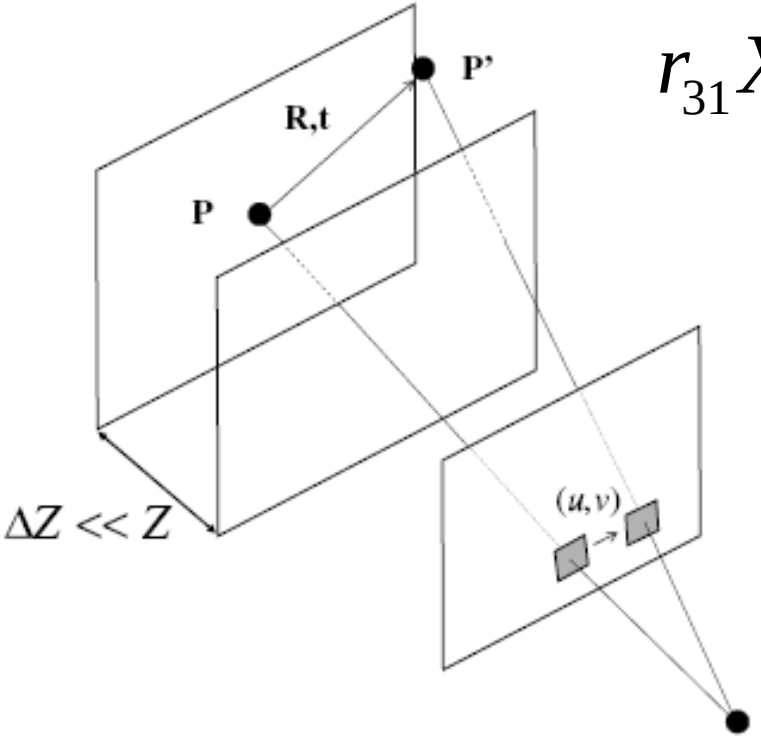
$$y = Y / Z$$

$$x' = \frac{r_{11}X + r_{12}Y + r_{13}Z + t_x}{r_{31}X + r_{32}Y + r_{33}Z + t_z}$$

$$y' = \frac{r_{21}X + r_{22}Y + r_{23}Z + t_y}{r_{31}X + r_{32}Y + r_{33}Z + t_z}$$

Motion

- Optical Flow
  - Distant motion -> affine flow
    - Assuming that the points are far from the camera, we can assume that the depth remains approximately constant:



$$r_{31}X + r_{32}Y + r_{33}Z + t_z \approx Z \approx Z_0$$

$$x' = r_{11}X / Z_0 + r_{12}Y / Z_0 + r_{13} + t_x / Z_0$$

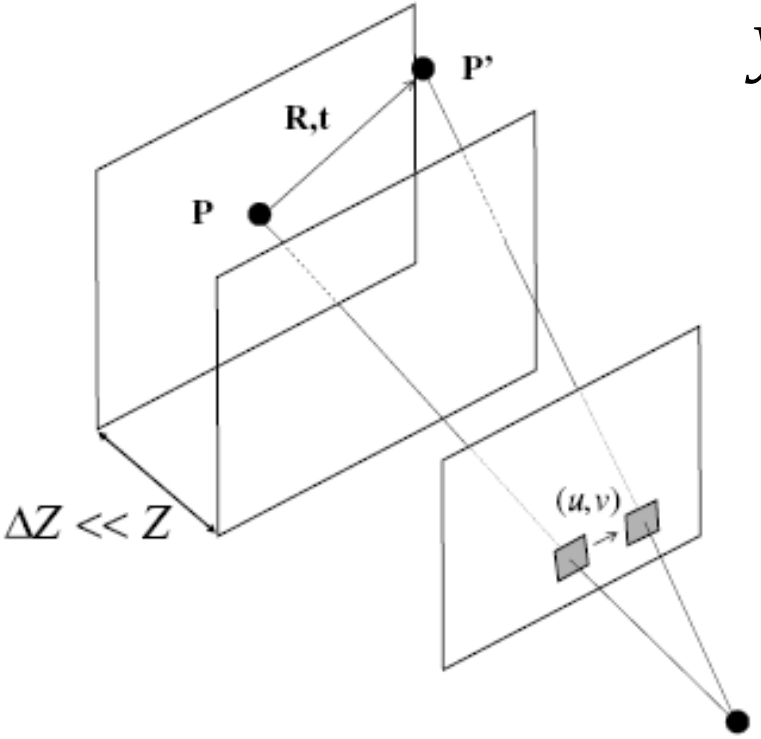
$$y' = r_{21}X / Z_0 + r_{22}Y / Z_0 + r_{23} + t_y / Z_0$$

Motion

- Optical Flow
  - Distant motion -> affine flow
    - Knowing that

$$x = X / Z_0$$

$$y = Y / Z_0$$



- We can eliminate the coordinates of the scene point to get a direct relation between the image coordinates:

$$x' = r_{11}x + r_{12}y + r_{13} + t_x / Z_0$$

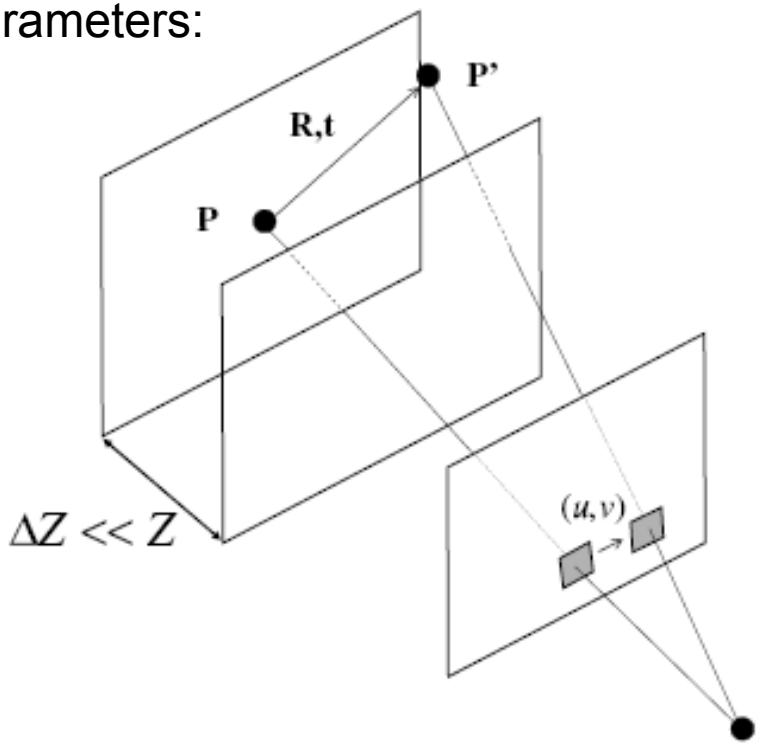
$$y' = r_{21}x + r_{22}y + r_{23} + t_y / Z_0$$

Motion

- Optical Flow
  - Distant motion -> affine flow
    - The motion from one image to the next has a linear expression
      - an *affine* model defined by 6 parameters:

$$u(x, y) = ax + by + c$$

$$v(x, y) = dx + ey + f$$



Motion

- Optical Flow
  - Distant motion -> affine flow
    - The idea of constant flow can be generalized by assuming that *u* and *v* are not constant but are (locally) linear functions of the image coordinates *x* and *y*. That is:

$$u(x, y) = ax + by + c$$

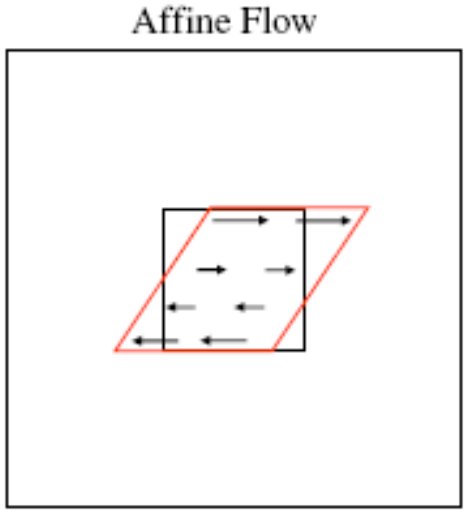
$$v(x, y) = dx + ey + f$$

    - In that case, the optical flow equation becomes:

$$E(u, v) = \sum_{(x,y)\in W} \left((ax + by + c) I_x(x, y) + (dx + ey + f) I_y(x, y) + I_t \right)^2$$

    can be solved directly by a matrix inversion

Motion



- Optical Flow
  - Distant motion -> affine flow
    - Geometrically, this assumption means that a window centered at a pixel is transformed in the next frame into a new window by an affine transformation of parameters *a*,*b*,*c*….
    - In the scene, this assumption on the image flow is equivalent to saying that the scene is locally planar and that the depth variation within a window is small with respect to the distance to the camera.

Motion

- Optical Flow
  - The steps followed for the translation and affine case can be generalized as follows
    - Suppose that the image at time t is related to the image at time 0 by the following relation:
      - $I(p(t),t) = I(0,0)$
    - Where $p(t)$ is a set of parameters. For example, $p(t)$ could be the set of parameters of an affine transformation. In that case, $I(p(t),t)$ is the image at time t warped to image 0.
    - At time t+1:
      - $p(t+1) = p(t) + \delta t$
    - And, as a first-order approximation:
      - $I(p(t+1),t+1) = I(p(t),t+1) + A\ \delta p$

Motion

- Optical Flow
  - Where A is the matrix of derivatives of $I(p(t),t+1)$ with respect to p and $I(p(t),t+1)$ is the image at time $t+1$ warped using the parameters computed at time t.
  - Since we want $I(p(t+1),t+1) = I(0,0)$, we want to find dp that minimizes:
  - $||A\delta p + I(p(t),t+1) - I(0,0)||^2$
  - The solution is: $\delta p = (A^T A)^{-1} A^T I_t$, where $I_t = I(p(t),t+1) - I(0,0)$

  – In general, the derivatives in A have to be recomputed every time. In fact, in many cases, like in the affine case, it turns out that the computation can be done on the template $I(0,0)$ instead and no re-computation is needed.

Object tracking

- Object tracking is a crucial research issue in computer vision, especially for the applications where the environment is in continuous changing:
  - Robot Vision
    - mobile robot navigation,
    - applications that must deal with unstable grasps
  - Surveillance
  - Traffic applications
  - Human motion capture

Object tracking

- The most common approaches for object tracking are based mainly on the detection of one of following cues:
  - Edges
  - Colour
  - Texture

Object tracking

- The most common approaches for object tracking are based mainly on the detection of one of following cues:
  - Edges
    - concerns the extraction of a number of features of the object
      - Points
      - Lines
      - Distances
      - Models of the contours.
    - These features allow to have fast tracking process and also to estimate the pose of the object.
    - This is generally based on the analysis of the gradients intensity
  - the edge based techniques are not suitable
    - for applications with highly texture environments or objects.
    - for applications where the light conditions are not stable or its interaction with the objects produces shadows.

Object tracking

- The most common approaches for object tracking are based mainly on the detection of one of following cues:
  - Colour
    - When color is the main different characteristic of the object in relation with the environment,
    - Extraction of several characteristics based on different color spaces
      - RGB
      - HSV
      - …
    - Have been proved to be efficient for situations where the light conditions are not uniform and are changing during the tracking procedure
      - HSV colour space

Object tracking

- The most common approaches for object tracking are based mainly on the detection of one of following cues:
  - Texture
    - Recently research application field
    - Used to object tracking especially as a complement to a multi-cue tracker.
    - The most common texture segmentation techniques tend to be computationally intensive, and use classification methods that require a time expensive off-line learning phase. For these reasons such approach isn’t consistently used for tracking purposes.

Object tracking

- Typically application has to deal with the following main problems:
  - The clutter environment, specifically non-uniform light conditions and different objects with the same color pattern.
  - The presence of the object at different distances relative to the camera during the tracking procedure.
  - Real time processing.
  - Image noise.
- Colour tracker
  - CAMSHIFT

Object tracking

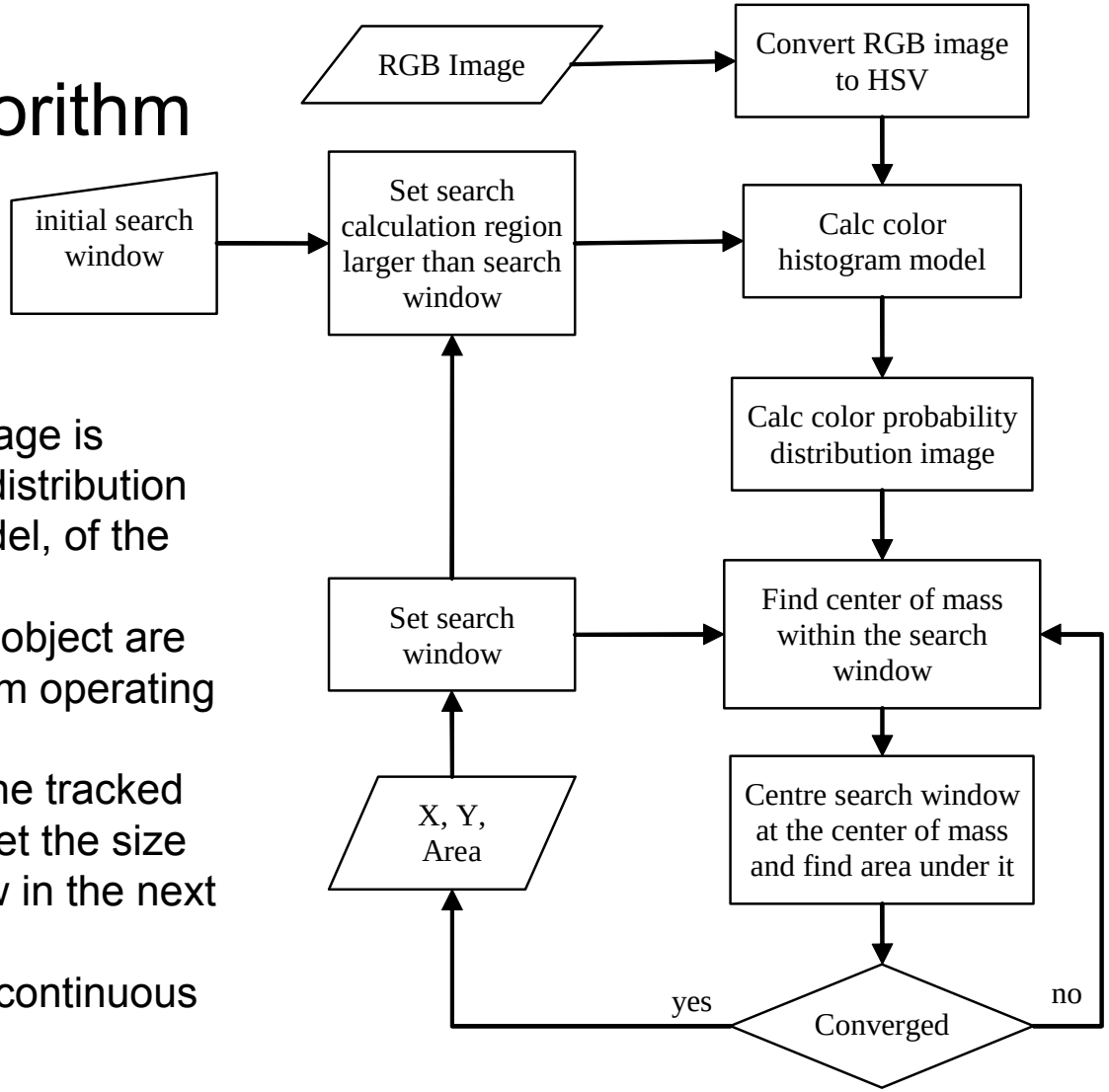
- CAMSHIFT Algorithm

Continuously Adaptive Mean Shift (CAMSHIFT)

  - Colour tracking procedure
  - This algorithm uses a search window to track the moving object, ignoring objects outside this search window.
  - Scales the search window to object size thus allowing different distance between the object and the camera.
  - The color space used is the HSV which is less sensitive to lighting changes.
  - The color model is mainly based on the hue histogram which eliminates much of the noise present in the image.

Object tracking

- CAMSHIFT Algorithm



For each video frame, the raw image is converted to a colour probability distribution image via a colour histogram model, of the colour being tracked.
The center and size of the colour object are found via the CAMSHIFT algorithm operating on the colour probability image.
The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image.
The process is then repeated for continuous tracking.

Object tracking

- CAMSHIFT Algorithm
  - A problem can occur when using HSV space
    - When brightness is low (V near 0), saturation is also low (S near 0). Hue then becomes quite noisy, since in such a small hexcone, the small number of discrete hue pixels cannot adequately represent slight changes in RGB.
    - This then leads to wild swings in hue values.
  - To overcome this problem
    - Simply ignore hue pixels that have very low corresponding brightness values.
    - With sunlight, bright white colours can take on a object hue so use an upper threshold to ignore object hue pixels with corresponding high brightness.
    - At very low saturation, hue is not defined so also ignore hue pixels that have very low corresponding saturation.
  - This means that the camera must auto-adjust or be adjusted for brightness

Object tracking

- CAMSHIFT Algorithm
  - Illumination Drift
  - Presence of other object