The problems

- Visual surveillance
  - stationary camera watches a workspace – find moving objects and alert an operator
  - moving camera navigates a workspace – find moving objects and alert an operator
- Image coding
  - use image motion to perform more efficient coding of images
- Navigation
  - camera moves through the world - estimate its trajectory
    - use this to remove unwanted jitter from image sequence - image stabilization and mosaicking
    - use this to control the movement of a robot through the world
Surveillance example: Adding an object to the scene

Image Sequence Smoothing
Motion detection

Frame differencing
- subtract, on a pixel by pixel basis, consecutive frames in a motion sequence
- high differences indicate change between the frames due to either motion or changes in illumination

Problems
- noise in images can give high differences where there is no motion
  » compare neighborhoods rather than points
- as objects move, their homogeneous interiors don’t result in changing image intensities over short time periods
  » motion detected only at boundaries
  » requires subsequent grouping of moving pixels into objects
Image Differencing: Results

1 frame difference

5 frame difference

Motion detection

- Background subtraction
  - create an image of the stationary background by averaging a long sequence
  - for any pixel, most measurements will be from the background
  - computing the median measurements, for example, at each pixel, will with high probability assign that pixel the true background intensity - fixed threshold on differencing used to find “foreground” pixels
  - can also compute a distribution of background pixels by fitting a mixture of Gaussians to set of intensities and assuming large population is the background
  - adaptive thresholding to find foreground pixels
- difference a frame from the known background frame
  - even for interior points of homogeneous objects, likely to detect a difference
  - this will also detect objects that are stationary but different from the background
  - typical algorithm used in surveillance systems

Motion detection algorithms such as these only work if the camera is stationary and objects are moving against a fixed background
Background Subtraction: Results

Confidence corresponds to gray-level value.
High confidence – bright pixels, low confidence – dark pixels.

Background modeling: color-based

- At each pixel model colors \((r,g,b)\) or gray-level values \(g\). The following equations are used to recursively estimate the mean and the variance at each pixel:

\[
\mu_{t+1} = \alpha \mu_t + (1 - \alpha) z_{t+1}
\]

\[
\sigma_{t+1}^2 = \alpha (\sigma_t^2 + (\mu_{t+1} - \mu_t)^2) + (1 - \alpha)(z_{t+1} - \mu_{t+1})^2
\]

where \(z_{t+1}\) is the current measurement. The mean \(\mu\) and the variance \(\sigma\) can both be time varying. The constant \(\alpha\) is set empirically to control the rate of adaptation \((0 < \alpha < 1)\).

- A pixel is marked as foreground if given red value \(r\) (or for any other measurement, say \(g\) or \(b\)) we have

\[
|r - \mu_t| > 3 \max(\sigma_r, \sigma_{rcam})
\]
Background model

- \( \sigma_{\text{cam}} \) is the variance of the camera noise, can be estimated from image differences of any two frames.
- If we compute differences for all channels, we can set a pixel as foreground if any of the differences is above the preset threshold.
- Noise can be cleaned using connected component analysis and ignoring small components.
- Similarly we can model the chromaticity values \( r_c, g_c \) and use them for background subtraction:
  
  \[
  r_c = r/(r+g+b), \quad g_c = g/(r+g+b)
  \]

Background model: edge-based

- Model edges in the image. This can be done two different ways:
  - Compute models for edges in the average background image
  - Subtract the background (model) image and the new frame; compute edges in the subtraction image; mark all edges that are above a threshold.
  - The threshold can be learned from examples
  - The edges can be combined (color edges) or computed separately for all three color channels
Foreground model

- Use either color histograms (4-bit per color), texture features, edge histograms to model the foreground
- Matching the foreground objects between frames: tracking
- Can compare foreground regions directly: shift and subtract. SSD or correlation: $M, N$ are two foreground regions.

\[
SSD = \sum_{i=1}^{n} \sum_{j=1}^{n} [M(i, j) - N(i, j)]^2 \\
C = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} M(i, j)N(i, j)}{[\sum_{i=1}^{n} \sum_{j=1}^{n} M(i, j)^2 \sum_{i=1}^{n} \sum_{j=1}^{n} N(i, j)^2]^{1/2}}
\]

A 300-Frame Sequence with a “Busy” Background
Some Intermediate Maps Used in the Method

Color-based moving object detection

Edge-based moving object detection

Combined color and edge based detection

Detected human
Results for the sequence

Using histograms for background modeling

- Use histograms of small regions to model the background:
  - Color histograms computed for small regions of the “background” image and the current (new) image (reduced color/12 bit bit representation)
  - Color edge histograms computed for small regions of the “background” image and the current image (36 bin quantization)
Color Histograms

Reduced color representation =
\[ C = \left( \frac{R}{16} \right) \times 256 + \left( \frac{G}{16} \right) \times 16 + \left( \frac{B}{16} \right) \]
(This results in a 24 -> 12 bit color depth reduction)
This results in a 4096 bin histogram
- lowest 4 bits are less useful
- requires less storage
- faster implementation - easier to compare histograms

Color Edge Histograms

- Use edge detector to compute edges in each color band
  \((r_x, r_y, g_x, g_y, b_x, b_y)\)
- Combine the three color bands into the structure matrix, S, to compute the color edge response
- The edge strength is computed as the larger of the two eigenvalues of S, and the orientation is given by the corresponding eigenvector
- Histogram bin index is determined using edge orientation (36 bins total), and the bin count is incremented using the edge magnitude
Histogram Matching

- Histogram Intersection

\[ I(h_c, h_b) = \sum_i \frac{\min\{h_c(i), h_b(i)\}}{\max\{h_c(i), h_b(i)\}} \]

- Chi Squared Formula

\[ \chi^2(h_c, h_b) = \sum_i 2 \frac{(h_c(i) - h_b(i))^2}{h_c(i) + h_b(i)} \]

Overall control

- Divide each frame into 40x40 pixel blocks
- To make sure that we do not miss objects on grid block boundaries we tile the frame by overlaying two grids, one of which is shifted by 20 pixels in x and y directions
Criteria for block activation

- On a block by block basis, similarity measures between background and foreground histograms are computed.
- For histogram intersection: If the similarity is below a threshold, $T$, then the block contains a foreground object and is activated for display.
- For chi squared: If the $X^2$ measure is greater than a threshold, $T$, then the block contains a foreground object and is activated for display.

Examples of edge histograms

- Similar histograms: $\text{Similarity (inters.)} = 92\%$, $X^2 = 61$
- Different histograms: $\text{Similarity (inters.)} = 22\%$, $X^2 = 828$
Using edge histograms for detection

Moving person in a cluttered scene
Color histogram based detection

Edge histogram-based detection
**Surveillance: dropping an object**

![Image of dropping an object](image1)

**Surveillance: removing an object**

![Image of removing an object](image2)
Optic flow is the 2-D velocity field induced in a dynamic scene due to the projection of moving objects onto the image plane.

Three prevalent approaches to computing optic flow:
- **token matching or correlation**
  - extract features from each frame (grey level windows, edge detection)
  - match them from frame to frame
- **gradient techniques**
  - relate optic flow to spatial and temporal image derivatives
- **velocity sensitive filters**
  - frequency domain models of motion estimation
A 1-d gradient technique

- Suppose we have a 1-D image that changes over time due to a translation of the image.
- Suppose we also assume that the image function is, at least over small neighborhoods, well approximated by a linear function.
- completely characterized by its value and slope
- Can we estimate the motion of the image by comparing its spatial derivative at a point to its temporal derivative?
- example: spatial derivative is 10 units/pixel and temporal derivative is 20 units/frame
- then motion is (20 units/frame) / (10 units/pixel) = 2 pixels/frame

Gradient techniques

- Assume I(x,y,t) is a continuous and differentiable function of space and time.
- Suppose the brightness pattern is locally displaced by a distance dx, dy over time period dt.
  - this means that as the time varying image evolves, the image brightnesses of points don’t change (except for digital sampling effects) as they move in the image.
  - I(x,y,t) = I(x + dx, y + dy, t + dt)
- We expand I in a Taylor series about (x,y,t) to obtain
  - I(x + dx, y + dy, t + dt) = I(x,y,t) + dx ∂I/∂x + dy ∂I/∂y + dt ∂I/∂t + (higher order terms)
- dI/dt = [I(x+dx, y+dy, t+dt) - I(x,y,t)]/dt = dx/dt ∂I/∂x + dy/dt ∂I/∂y + ∂I/∂t = 0
  - valid only if temporal change is due entirely to motion.
- Can rewrite this as dI/dt = G_x u + G_y v + G_t = 0. The G’s are derivatives measured from the image sequence, and u and v are the unknown optic flow components in the x and y directions, respectively.
So, the spatial and temporal derivatives at a point in the image only provide a linear constraint on the optic flow.

If \( G_x \) and \( G_y \) are small, then motion information cannot be accurately determined.

If \( G_x = 0 \), then \( -G_t = G_y v \), so that \( v \) is determined, but \( u \) is unknown.

If \( H \) and \( L \) denote the gradient and level directions at a pixel then
- \( G_H = \| VG \| \)
- \( L \) is perpendicular to \( H \)
- \( G_L = 0 \)

Then \( G_t = -G_h \frac{dh}{dt} \), where \( n = \frac{dh}{dt} \) is the displacement in the gradient direction \( (h = VG / \| VG \|) \)
- \( \frac{dh}{dt} \) can be recovered by measuring \( G_t \) and \( G_H \). It is called normal flow.
- But \( \frac{dl}{dt} \) cannot be recovered, since \( G_L = 0 \).
- This is called the aperture problem.
Aperture problem

Motion Flow Example: Images
Recovering $u$ and $v$

- Compute for normal flow in a small image neighborhood
  - $n_j = -G_x/\|\nabla G\|$ 
- Solve system of linear equations corresponding to motion constraints in the small neighborhood
  - assume $u$ and $v$ will not vary in that small neighborhood
  - requires that neighborhoods have edges with different orientations, since slope of motion constraint line is determined by image gradient
Recovering u and v

- If the constraint lines in a neighborhood are nearly parallel (i.e., the gradient directions are all similar), then the location of the best fitting (u,v) will be very sensitive to errors in estimating gradient directions.

- More generally, one could fit a parametric form to local neighborhoods of constraint lines, finding parameters that bring constraint lines “nearest” to the estimated motion assigned to each pixel.
  - for example, if we assume that the surface we are viewing in any small image neighborhood is well approximated by a plane, then the optical flow will be a quadratic function of image position in that image neighborhood

A regularization approach

- Many vision problems such as stereo reconstruction of visible surfaces and recovery of optic flow are instances of ill posed problems.

- A problem is well posed when its solution:
  - exists
  - is unique, and
  - depends continuously on its initial data

- Any problem that is not well posed is said to be ill posed

- The optic flow problem is to recover both degrees of freedom of motion at each image pixel, given the spatial and temporal derivatives of the image sequence
  - but any solution chosen at each pixel that locally satisfies the motion constraint equation can be used to construct an optic flow field consistent with the derivatives measured
  - therefore, the solution is not unique - how to choose one?
A regularization approach

- Solution - add a priori knowledge that can choose between the solutions
- Formally, suppose we have an ill posed problem of determining $z$ from data $y$ expressed as
  - $Az = y$, where $A$ is a linear operator (e.g., projection operation in image formation)
- We must choose a quadratic norm $|| \cdot ||$ and a so-called stabilizing functional $||Pz||$ and then find the $z$ that minimizes:
  - $||Az-y||^2 + \lambda ||Pz||^2$
  - $\lambda$ controls the compromise between the degree of regularization and the closeness of the solution to the input data (the first term).

For optic flow:
- the first term is $[dx/dt \partial I/\partial x + dy/dt \partial I/\partial y + \partial I/\partial t]^2 = [dI/dt]^2$
  - this should, ideally, be zero according to the theory
- the second term enforces a smoothness constraint on the optic flow field:
  - $\varepsilon = (\partial u/\partial x)^2 + (\partial v/\partial x)^2 + (\partial u/\partial y)^2 + (\partial v/\partial y)^2$
- The regularization problem is then to find a flow field that minimizes
  - $[dI/dt]^2 + \lambda \varepsilon$
- This minimization can be done over the entire image using various iterative techniques
Token and correlation methods

- Gradient based methods only work when the motion is “small” so that the derivatives can be reliably computed
  - although for “large” motions, one can employ multiresolution methods
- Tracking algorithms can compute motion when the motion is “large”
  - correlation
  - feature tracking
- Correlation
  - choose a kxk window surrounding a pixel, p, in frame i.
  - compare this window against windows in similar positions in frame i+1
  - The window of best match determines the displacement of p from frame i to frame i+1

Correlation

- Correlation
  - sum of squared gray level differences
  - sum of absolute intensity differences
  - “robust” versions of these sensitive to outliers
- Drawbacks of correlation
  - matching in the presence of rotation is computationally expensive since all orientations of the window must be matched in frame i+1
  - if motion is not constant in the kxk window then the window will be distorted by the motion, so simple correlation methods will fail
    - this suggests using smaller windows, within which motion will not vary significantly
    - but smaller windows have less specificity, leading to matches more sensitive to noise
Tracking

- Apply a feature detector, such as an edge detector, to each frame of the sequence
  - want features to be distinctive
  - example: patterns of edges or gray levels that are dissimilar to their surrounds (image has a locally small autocorrelation)
- Match these features from frame to frame
  - might assume that nearby features move similarly to help disambiguate matches (but this is not true at motion boundaries)
  - integrate the matching with assumptions about scene structure - e.g., features are all on a plane moving rigidly

Motion estimation – token matching

- Extract features from each frame (grey level windows, edge detection)

\[
S = \begin{pmatrix}
\Sigma E_x^2 & \Sigma E_x E_y \\
\Sigma E_x E_y & \Sigma E_y^2
\end{pmatrix}
\]

- \( \lambda_1 \geq \lambda_2 \geq 0 \) are eigenvalues of \( M \)
- If \( \lambda_1 = \lambda_2 = 0 \), mean squared magnitude of the gradient is 0 (flat, unchanging area in the image)
- If \( \lambda_1 > \lambda_2 = 0 \), values do not change in the direction of the corresponding eigenvector (edge)
- If \( \lambda_1 > 0 \) and \( \lambda_2 > 0 \), gray values change in multiple directions (corner)
  \( \Rightarrow \lambda_2 > \tau \), where \( \tau \) is some threshold
Motion estimation – token matching

- Match them from frame to frame. Detect tokens in the next frame using lower threshold. Why?
  - Minimize SSD (sum of squared differences) over a neighborhood in the new image. $M$ is a small area around the token (5x5, 7x7, 11x11)

$$SSD = \sum_{i=1}^{n} \sum_{j=1}^{n} (M(i, j) - N(i, j))^2$$

- Maximize the correlation over a neighborhood in the new image

$$C = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} M(i, j) N(i, j)}{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} M(i, j)^2 \sum_{i=1}^{n} \sum_{j=1}^{n} N(i, j)^2\right)^{1/2}}$$

Multiresolution methods

- Consider using edges as features for a tracking algorithm for motion estimation. What should the scale of the edge detector be?
  - small scale
    - many edges are detected
    - easily confused with one another
    - computationally costly matching problem
  - coarse scale
    - relatively few edges identified
    - localized only poorly, so motion estimates have high errors
    - simple matching problem
Multiresolution methods

- Multiresolution - process the image over a range of scales, using the results at coarser scales to guide the analysis at finer scales
  - detect edges at a coarse scale
  - estimate motion by tracking
  - use these estimates as initial conditions for matching edges at next finest scale
- These are also called focusing methods or scale space methods
  - can also apply to gradient based motion estimators

3-D motion and optical flow

- Assume a camera moving in a static environment
- A rigid body motion of the camera can be expressed as a translation and a rotation about an axis through the origin.
- Let
  - \( \mathbf{t} \) be the translational component of the camera motion
  - \( \omega \) be the angular velocity
  - \( \mathbf{r} \) be the column vector \([X Y Z]^T\)
- Then the velocity of \( \mathbf{r} \) with respect to the XYZ coordinate system is
  \[
  \mathbf{V} = -\mathbf{t} + \omega \times \mathbf{r}
  \]
- Let the components of
  - \( \mathbf{t} = [U V W]^T \)
  - \( \mathbf{w} = [A B C]^T \)
3-D Motion and Optic Flow

- Rewrite in component form:
  \[ X' = -U - BZ + CY \]
  \[ Y' = -V - CX + AZ \]
  \[ Z' = -W - AY + BX \]
  where the differentiation is with respect to time.

- The optic flow at a point \((x, y)\) is \((u, v)\) where
  \[ u = x', \ x = fX/Z \]
  \[ v = y', \ y = fY/Z \]

- Differentiating \(x\) and \(y\) with respect to time, we obtain
  \[ u = X'/Z - XZ'/Z^2 = (-U/Z - B + Cy) - x(-W/Z - Ay + Bx) \]
  \[ v = Y'/Z - YZ'/Z^2 = (-V/Z - Cx + A) - y(-W/Z - Ay + Bx) \]

These can be written in the form
\[ u = u_t + u_r \]
\[ v = v_t + v_r \]

- \((u_t, v_t)\) denotes the translational component of the optic flow.
- \((u_r, v_r)\) denotes the rotational component of the optic flow.

\[ u_t = [-U + xW]/Z \]
\[ v_t = [-V + yW]/Z \]
\[ u_r = Axy - B(x^2 + 1) + Cy \]
\[ v_r = A(y^2 + 1) - Bxy - Cx \]

- Notice that the rotational part is independent of \(Z\) - it just depends on the image location of a point.
- So, all information about the structure of the scene is revealed through the translational component.
Special case of a plane in motion

- Suppose we are looking at a plane while the camera moves
  - \( Z = Z_0 + pX + qY \)
- Then for any point on this plane
  - \( Z - pX - qY = Z_0 \)
  - \( 1 - p(X/Z) - p(Y/Z) = Z_0/Z \)
  - \( 1/Z = [1-pX/Z - qY/Z]/Z_0 = [1- px - qy]/Z_0 \)
- So, we can rewrite the translational components of motion for a plane as:
  - \( u_t = [-U + xW]/Z \)
  - \( v_t = [-V + yW]/Z \)
- Consider the special point \((u, v) = (U/W, V/W)\).
  - This is the “image” of the velocity vector onto the image plane
  - The motion at this point must be 0 since the surface point along this ray stays on the ray as the camera moves (also our equations evaluate to 0 at \((U/W, V/W)\))
- Consider the line connecting any other \((x, y)\) to \((x + u_t, y + v_t)\)
  - The slope of this line is \(v_t/u_t = [x-u]/[y-v]\)
  - So, the line must pass through \((u, v)\)
- All of the optic flow vectors are concurrent, and pass through the special point \((u, v)\) which is called the **focus of expansion (contraction)**

---

Pure translation

- When camera motion is only translation, then we have
  - \( u_t = [-U + xW]/Z \)
  - \( v_t = [-V + yW]/Z \)
- Consider the special point \((u, v) = (U/W, V/W)\).
  - This is the “image” of the velocity vector onto the image plane
  - The motion at this point must be 0 since the surface point along this ray stays on the ray as the camera moves (also our equations evaluate to 0 at \((U/W, V/W)\))
- Consider the line connecting any other \((x, y)\) to \((x + u_t, y + v_t)\)
  - The slope of this line is \(v_t/u_t = [x-u]/[y-v]\)
  - So, the line must pass through \((u, v)\)
- All of the optic flow vectors are concurrent, and pass through the special point \((u, v)\) which is called the **focus of expansion (contraction)**
Another way to look at it

- Let $\Delta t = 1$, so that the image center at time $t$ moves from $(0,0,0)$ to $(U,V,W)$ at time $t+1$
- Think of the two images as a stereo pair
- The location of the projection of $(U,V,W)$, the lens center at time $t+1$ (the “right” image), in the image at time $t$ (the left image) is at location $(U/W, V/W) = (u,v)$
- All conjugate lines at time $t$ must pass through this point
- So, given a point $(x,y)$ at time $t$, the location of its corresponding point at time $t+1$ in the original coordinate system must line on the line connecting $(x,y)$ to $(u,v)$

So, if we know the optic flow at two points in the case of pure translation, we can find the focus of expansion

- In practice want more than two points

Can we recover the third component of motion, $W$?

No, because the same optic flow field can be generated by two similar surfaces undergoing similar motions ($U$, $V$, and $W$ always occur in ratio with $Z$).
Normal flows and camera motion estimation

- If we can compute optic flow at a point, then the foe is constrained to lie on the extension of the optic flow vector
- But the aperture problem makes it difficult to compute optic flow without making assumptions of smoothness or surface order
- Normal flow (the component of flow in the gradient direction) can be locally computed at a pixel without such assumptions
- Can we recover camera motion from normal flow?

Identifying the FOE from normal flow

- Assume that the foe is within the field of view of the camera
- For each point, \( p \), in the image
  - For each normal flow vector, \( \mathbf{n} \)
    - If \( p \) lies in the “correct” halfplane of \( \mathbf{n} \), then score a vote for \( p \)
  The FOE is the centroid of the connected component of highest scoring points (might be a single pixel, but ordinarily will not be).
- Alternative code - maintain an array of counters in register with the image
  - For each normal flow vector, \( \mathbf{n} \)
    - Increment the counters corresponding to all pixels in the “correct” halfplane of \( \mathbf{n} \)
    - Search the array of counters for the connected component of highest vote count
- For an image containing \( N \) normal flow vectors and \( mxm \) pixels, both algorithms are \( (m^2N) \), but (2) is more efficient
Identifying the FOE from normal flow

- What if the FOE is outside the field of view of the camera?
- The image plane is a bad place to represent the FOE to begin with
  - FOE indicates the direction of translational motion
  - Pixels in a perspective projection image do not correspond to equal angular samples of directions
    » in the periphery, a pixel corresponds to a wide range of directions
  - Solution - represent the array of accumulators as a sphere, with an equiangular sampling of the surface of the sphere
    » Each normal vector will then cast votes for all samples in a hemisphere
    » Simple mathematical relationship between the spherical coordinate system of the array of counters, and the image coordinate system