Time-varying image analysis

- Motion detection
- Motion estimation
- Egomotion and structure from motion

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
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

- 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
Motion estimation - optic flow

- Optic flow is the 2-D velocity field induced in an image 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 (gray 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
- \( l(x) \) then motion is \( \frac{20 \text{ units/frame}}{10 \text{ units/pixel}} = 2 \text{ pixels/frame} \)
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 brightness 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

\[
I(x + dx, y + dy, t + dt) = I(x, y, t) + dx\frac{\partial I}{\partial x} + dy\frac{\partial I}{\partial y} + dt\frac{\partial I}{\partial t} + \text{higher order terms}
\]

\[
d\frac{I}{dt} = \frac{[I(x + dx, y + dy, t + dt) - I(x, y, t)]}{dt} = \frac{d}{dt}(dx\frac{\partial I}{\partial x} + dy\frac{\partial I}{\partial y} + dt\frac{\partial I}{\partial t})
\]
- 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.

- Places in the image where the gray level is almost constant are difficult places to estimate motion.
  - $G_t$ will also be small in these places.

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

- If $G_x = 0$, we have a horizontal edge, so we can’t measure its motion along the edge.
If $H$ and $L$ denote the gradient and level directions at a pixel then
- $H = \tan^{-1} \frac{G_y}{G_x}$
- $L$ is perpendicular to $H$
- $G_L = 0$

Then $G_t = -G_H \frac{dH}{dt}$, where $dH/dt$ is the displacement in the gradient direction
- $dH/dt$ can be recovered by measuring $G_t$ and $G_H$. It is called normal flow
- but $dL/dt$ cannot be recovered, since $G_L = 0$
- this is called the aperture problem
Recovering u and v

- Solve for u and v separately, ignoring their coupling through 2-D motion
  - \( u = -G_t/G_x \)
  - \( v = -G_t/G_y \)
- Solve system of linear equations corresponding to motion constraints in a small image 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

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 model 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.
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, once 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
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
- 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

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 - 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
Multiresolution methods

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
- \( \mathbf{\omega} \) be the angular velocity
- \( \mathbf{\tau} \) be the column vector \([X \ Y \ Z]^T\)

Then the velocity of \( \mathbf{\tau} \) with respect to the XYZ coordinate system is
\[
\mathbf{V} = -\mathbf{t} + \mathbf{\omega} \times \mathbf{\tau}
\]

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
  
  \[ u = X'/Z - X(Z'/Z^2) = (-U/Z - B + Cy) - x(-W/Z - Ay + Bx) \]
  \[ v = Y'/Z - Y(Z'/Z^2) = (-V/Z - Cx + A) - y(-W/Z - Ay + Bx) \]

- 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.
Mosaicing from a rotating camera

- If we take a camera and rotate it, we can combine all of the images into a panoramic mosaic.

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:
  - \( u_t = [-U + xW][1 - px - qy]/Z_0 = [-U/Z_0 + xW/Z_0]/Z_0 \)
  - \( v_t = [-V + yW][1 - px - qy]/Z_0 = [-V/Z_0 + xW/Z_0]/Z_0 \)
- These are quadratic equations in \( x \) and \( y \)
- So, if we can compute the translational component of the optic flow at “enough” points from a planar surface, then we can recover the translational motion (with unknown scaling) and the orientation of the plane being viewed.
When camera motion is only translation, then we have
\[ u_t = \frac{-U + xW}{Z} \]
\[ v_t = \frac{-V + yW}{Z} \]

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
**Pure translation**

- 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, \( n \):
    - If \( p \) lies in the “correct” halfplane of \( n \) 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, \( n \):
    - Increment the counters corresponding to all pixels in the “correct” halfplane of \( n \).
  - 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.

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.
If we can compute the 3D motion parameters of an image sequence then we can compute the (scaled) range to visible points in the scene.

- So, if the camera motion is a simple translation, then the Z coordinate of a point is inversely proportional to the length of the optical flow vector - just like disparity for stereo.

Practical problems
- motion is not simple translation, but also includes rotation
  » small rotations about the y axis are easy to confuse with translations in x
- computing optical flow more difficult than normal flow

More practical problems
- discontinuities in range
  » optical flow algorithms integrate information over small image neighborhoods. If those neighborhoods overlap a boundary between an object and the background, then the assumptions on which the algorithm is based (e.g., planar surface) are violated and the result will be wrong.

- Independently moving objects
  » will confuse the algorithms that estimate 3D motion parameters because their motion is inconsistent with the rigid camera motion
Structure from motion
Many vision problems such as stereo reconstruction of visible surfaces and recovery of optic flow are instances of \textit{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?

Solution - add \textit{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 $\| \|$ 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).

A regularization approach

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