Visual Motion

- Over sequence of images can determine which pixels move where
- Differs from motion in the world
  - Camera motion
    - Pan, tilt, zoom
  - Motion parallax
    - Information about depth from camera motion
  - Scene motion
    - Reveals independent objects and behaviors
  - Un-detectable motion
    - No/low intensity variation
Some Uses of Visual Motion

- Human-machine interaction
  - Animation, gestures, facial expressions
- Surveillance and monitoring
  - Tracking and analyzing behaviors
    - Collision detection and avoidance
- Camera stabilization
  - Remove jitter
- Autonomous navigation
  - Path finding and depth from parallax
- Constructing panoramic mosaics
Motion Analysis in Video

- **Video insertion**
  - Compute motion in one image sequence
  - Use to transform frames of another sequence and superimpose
  - Today used to insert signs and markings into sporting events

- **Panoramic mosaics**
  - Synthesized views from video sequence
Estimating Visual Motion

- Historically two different approaches
  - Direct methods, based on local image derivatives at each pixel
  - Feature based methods, sparse correspondence

- We will focus on direct methods
  - Used most in practice
  - Recover image motion from spatio-temporal variations in brightness
  - Dense estimates but can be sensitive to variations in appearance
Direct Motion Estimation Methods

- Based on the following assumptions
  - Every pixel in image I goes to some location in subsequent image J
  - Overall brightness of images I,J does not change (much)
- Called brightness constancy equation
  \[ I(x,y) \approx J(x+u(x,y), y+v(x,y)) \]
Using Brightness Constancy

- Minimization formulation
  - Seek \((u(x,y), v(x,y))\) minimizing error 
    \([I(x,y) - J(x+u(x,y),y+v(x,y))]^2\)
  - Not practical to search explicitly!

- Linearization
  - Relate motion to image derivatives
    - Gradient constraint
  - Assuming small \(u,v\) (on order of a pixel)
  - First order term of Taylor series expansion of brightness constancy
Gradient Constraint

- One-dimensional example – linearization
  - Estimate displacement \( d \) using derivative
    - Two functions \( f(x) \) and \( g(x) = f(x-d) \)
  - Taylor series expansion
    \[ f(x-d) = f(x) - d f'(x) + E \]
    - Where \( f' \) denotes derivative
  - Now write difference as
    \[ f(x)-g(x) = d f'(x) + E \]
  - Neglecting higher order terms
    \[ \delta = (f(x)-g(x))/f'(x) \]
  - Note only for small \( d \)
Gradient Constraint (or Optical Flow Constraint)

- Same approach extends naturally to 2D:
  \[ I(x,y) \approx J(x+u,y+v), \quad u=u(x,y), \quad v=v(x,y) \]
  - Assume time-varying image intensity well approximated by first order Taylor series:
  \[ J(x+u,y+v) \approx I(x,y)+I_x(x,y)\cdot u+I_y(x,y)\cdot v+I_t \]
  - Substituting:
  \[ I_x(x,y)\cdot u+I_y(x,y)\cdot v \approx -I_t \]
  - Using gradient notation:
  \[ \nabla I \cdot (u,v) \approx -I_t \]
  - Linear constraint on motion \((u,v)\) at each pixel
  - Can only estimate motion in gradient direction
Aperture Problem (Normal Flow)

- Can only measure motion in direction normal to edge (along gradient)
Aperture Problem (Normal Flow)

- Gradient constraint defines line in (u,v) space
  \[ \nabla I \cdot (u,v) \approx -I_t \n\]
- Methods based solely on per pixel estimates don’t work well
Combining Local Constraints

- Each pixel defines linear constraint on possible \((u,v)\) displacement
  - For set of pixels with same displacement combine constraints to get estimate
  - For pixels with different displacements, somehow identify that is case
Translational Motion

- Assume single displacement \((u,v)\) for all pixels within some region of image
- Over-constrained system of linear equations \(I_x(x,y) \cdot u + I_y(x,y) \cdot v = -I_t\)
- Find least squares solution
  - In matrix form: \(\min_z \| Dz - t \|

\[
D = \begin{bmatrix}
I_x(x_1,y_1) & I_y(x_1,y_1) \\
\vdots & \vdots \\
I_x(x_n,y_n) & I_y(x_n,y_n)
\end{bmatrix}
\]

and \(t = [I_t(x_1,y_1) \ldots I_t(x_n,y_n)]^T\)
Least Squares Solution

- \( z^* = (D^TD)^{-1} D^Tt \)
  - Method of normal equations, can derive from setting partial derivatives to zero

\[
D^TD = \begin{pmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{pmatrix} \quad D^Tt = \begin{pmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{pmatrix}
\]

- Inverse of 2x2 closed form

\[
A = \begin{pmatrix}
a & b \\
c & d
\end{pmatrix} \quad A^{-1} = 1/(ad-bc) \begin{pmatrix}
d & -b \\
-c & a
\end{pmatrix}
\]

Where \( \text{det}(A) = ad-bc \) not (near) zero
Translational Motion

- Can estimate small translation over local patch around each pixel
  - Fast using box sums
  - Note relation to corner detection
  - Poor estimate if A nearly singular
  - Also poor if patch contains more than one underlying motion

- Better handling of multiple motions
  - Robust statistical techniques

- Handling larger translations
  - Pyramid method
Multiple Motions

- Robust statistical techniques for finding predominant motion in a region
- Consider approach of iteratively reweighted least squares (IRLS)
  - As illustration of robust methods
- Generalize minimization problem to
  \[ \min_{z} \| W(Dz - t) \| \]
  - Weight matrix W is diagonal
  - Lessen importance of pixels that don’t match
  - Iterate to find “good” weights
  - Note in unweighted case W is identity matrix
Finding Predominant Motion

- Minimization generalizes in obvious way
  \[ z^* = (D^TW^2D)^{-1} D^TW^2t \]
- Determining good weights to use
  - Start by computing least squares solution, \( z^0 \)
  - Iteratively compute better solutions
    - Compute error for each pixel based on previous solution \( z^{k-1} \) and use that to set weight per pixel
    - Depends on initial solution being good enough to allow “bad pixels” to have largest error
    - Have to measure error based on image intensity matches, it’s the only thing we can measure
Updating Weights

To solve for $z^k$ given $z^{k-1}$

- Create weights $W^k = \text{diag}(w_1^k \ldots w_n^k)$ where

$$w_i^k = \begin{cases} 1 & \text{if } r_{i}^{k-1} \leq c \\ c/r_{i}^{k-1} & \text{otherwise} \end{cases}$$

- Where $r_{i}^{k-1}$ is measure of error at i-th pixel with motion estimate from iteration $k-1$
  - Compare i-th pixel value to matching pixel of other image (using $z^{k-1}$ for correspondence)
  - And $c$ is set based on robust measure of good versus bad data, such as median
    - Common value is $1/\.6745 \ \text{median}(r_{i}^{k-1})$
Weights Example

\[ z^{k-1} \]

\[ r_i^{k-1}: 0,0,1,0,1,1,6,5,6 \]
\[ w_i^k: 1,1,1,1,1,1,.24,.29,.24 \]

\[ \text{median} = 1 \]
\[ c \approx 1.48 \]
Global Motion Estimation

- Estimate motion vectors that are parameterized over some region
  - Each vector fits some low-order model of how vectors change
- Affine motion model is commonly used
  \[ u(x,y) = a_1 + a_2 x + a_3 y \]
  \[ v(x,y) = a_4 + a_5 x + a_6 y \]
- Substituting into grad. constr. equation
  \[ I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) \approx -I_t \]
  - Each pixel provides a linear constraint in six unknowns
Affine Transformations

- Consider points \((x,y)\) in plane rather than vectors for the moment
  - Linear transformation and translation
    \[
    x' = a_1 + a_2 x + a_3 y \\
    y' = a_4 + a_5 x + a_6 y
    \]
  - In matrix form \(A(z)=Lz+b\)
    \[
    \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_2 & a_3 \\ a_5 & a_6 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_1 \\ a_4 \end{pmatrix}
    \]
  - Maps any triangle to any triangle
    - Defined by three corresponding pairs of points
Why Affine Transformations

- Simple (and often inaccurate) model of projection
  - Point \((x,y,z)\) in space maps to \((x,y)\) in image
  - Orthographic or parallel projection
- Somewhat reasonable model for telephoto lens
- Yields affine transformation of plane for viewing “flat objects”
  - 3D rotation, translation followed by orthographic projection and scaling
Affine Motion Estimation

- Minimization problem become that of estimating the parameters $a_1, \ldots, a_6$
  - Rather than just two parameters $u,v$
- Still (over-constrained) linear system but in more unknowns
  - Again use least squares to solve
- Separable into two independent 3 variable problems
  - $a_1, a_2, a_3$ reflect only $u$-component of motion
  - $a_4, a_5, a_6$ reflect only $v$-component of motion
Affine Motion Equations

- Again compute \((D^TD)^{-1} DT_t\)
  - Or (re)weighted version for IRLS
- Now two 3x3 problems, one for \(I_x\) and one for \(I_y\), as opposed to single 2x2 problem
- Problem for \(I_x\) and \(u\) motion (\(I_y\) analogous)
  - \(T\) remains same, \(D\) changes

\[
D = \begin{pmatrix}
I_{x1} & x_1 & I_{x1} & y_1 & I_{x1} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
I_{xn} & x_n & I_{xn} & y_n & I_{xn}
\end{pmatrix}
\]
Multiple (Layered) Motions

- Combining global parametric motion estimation with robust estimation
  - Calculate predominant parameterized motion over entire image (e.g., affine)
  - Corresponds to largest planar surface in scene under orthographic projection
    - If doesn’t occupy majority of pixels robust estimator will probably fail to recover its motion
  - Outlier pixels (low weights in IRLS) are not part of this surface
    - Recursively try estimating their motion
    - If no good estimate, then remain outliers
Other Global Motion Models

- The affine model is simple but not that accurate in some imaging situations
  - For instance “pinhole” rather than “parallel” camera model for closer objects
  - Non-planar surfaces
  - Explicit modeling of motion parallax

- Projective planar case
  \[ x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y} \]
  \[ y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y} \]
  and \( u = x' - x, \ v = y' - y \)

- 3D models such as residual planar parallax
Handling Larger Motions

- Methods based on image gradients are restricted to small displacements
- Two different approaches
  - Abandon gradient method and explicitly search over possible translations
    - Computationally expensive to do for every pixel
      - Consider shifts and products of image patch
    - Block motion provides estimates just for certain pixels, used in compression (e.g., MPEG)
  - Pyramid to guarantee small motions
    - At top level small motion
    - At each level small deviation from one above
Coarse to Fine Motion Estimation

- Estimate residual motion at each level of Gaussian pyramid

Original

Pyramid of image I

Pyramid of image J

1/2\(^k\) res

\(I^1, J^1\)

\(I^0, J^0\)
Coarse to Fine Estimation

- Compute $M^k$, estimate of motion at level $k$
  - Can be local motion estimate $(u^k,v^k)$
    - Vector field with motion of patch at each pixel
  - Can be global motion estimate
    - Parametric model (e.g., affine) of dominant motion for entire image
  - Choose max $k$ such that motion about one pixel

- Apply $M^k$ at level $k-1$ and estimate remaining motion at that level, iterate
  - Local estimates: shift $I^k$ by $2(u^k,v^k)$
  - Global estimates: apply inverse transform to $J^{k-1}$
Global Motion Coarse to Fine

- Compute transformation $T_k$ mapping pixels of $I^k$ to $J^k$
- Warp image $J^{k-1}$ using $T^k$
  - Apply inverse of $T^k$
  - Double resolution of $T^k$ (translations double)
- Compute transformation $T^{k-1}$ mapping pixels of $I^k$ to warped $J^{k-1}$
  - Estimate of “residual” motion at this level
  - Total estimate of motion at this level is composition of $T^{k-1}$ and resolution doubled $T^k$
    - In case of translation just add them
Affine Mosaic Example

- Coarse-to-fine affine motion
  - Pan tilt camera sweeping repeatedly over scene
- Moving objects removed from background
  - Outliers in motion estimate, use other scans
**SSD**

- An alternative to gradient based methods is template matching
  - Treat a rectangle around each pixel as a “template” to find best match in other image
  - Search over possible translations minimizing some error criterion (or maximizing quality)
  - Generally use sum squared difference (SSD)
    \[ \sum \sum (I(x,y)-J(x+u,y+v))^2 \]
  - Sometimes compute cross correlation
  - Compute over local neighborhood