Recap

Motion

- Brightness constancy constraint
  \[ I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \]

- Optical flow equation, brightness constancy equation
  - Normal flow can be computed
  - Parallel flow cannot

\[ u I_x + v I_y + I_t = 0 \quad v = u \frac{I_x}{I_y} + \frac{I_t}{I_y} = 0 \quad \text{line equation} \]
Recap
Computing Optical Flow

- Define energy function
  - Brightness constancy + smooth solution (Horn&Schuck)
  - Common flow (Schuck)
  - Brightness constancy (Lucas&Kanade)
    - Least squares fitting

Recap
Global Motion

- Common motion of pixels observed in frame
  - Camera motion or rigid object motion

- Affine Model
  - Affine Transformation
  - Affine Motion
Recap
Affine Transformation

- Direct relation between pixel positions

\[ x'' = a_1 x + a_2 y + b_1 \]
\[ y'' = a_3 x + a_4 y + b_2 \]

Recap
Affine Motion

\[ u(x, y) = a_1 x + a_2 y + b_1 \]
\[ u = x'' - x \]
\[ x'' - x = a_1 x + a_2 y + b_1 \]
\[ x'' = (a_1 + 1) x + a_2 y + b_1 \]

\[ v(x, y) = a_3 x + a_4 y + b_2 \]
\[ v = y'' - y \]
\[ y'' - y = a_3 x + a_4 y + b_2 \]
\[ y'' = a_3 x + (a_4 + 1) y + b_2 \]
Recap
Affine Motion and Transformation

- Transformation
  \[ x'' = a_1 x + a_2 y + b_1 \]
  \[ y'' = a_3 x + a_4 y + b_2 \]

- Motion
  \[ x'' = (a_1 + 1) x + a_2 y + b_1 \]
  \[ y'' = a_3 x + (a_4 + 1) y + b_2 \]

Recap
Anandan’s Approach (Affine Motion)

\[
\begin{align*}
    u(x, y) &= a_1 x + a_2 y + b_1 \\
    v(x, y) &= a_3 x + a_4 y + b_2 \\
    \mathbf{u} &= \mathbf{Xa} \\
    \Delta I^T \mathbf{u} + I_t &= 0 \\
    \Delta I^T \mathbf{Xa} + I_t &= 0
\end{align*}
\]

minimize \( E = \sum (\Delta I^T \mathbf{Xa} + I_t)^2 \)

\[
\frac{\partial E}{\partial \mathbf{a}} = 2 \sum (\Delta I^T \mathbf{X})^T (I_t + \Delta I^T \mathbf{Xa}) = 0
\]

\[
\sum \mathbf{X}^T \Delta I \Delta I^T \mathbf{Xa} = - \sum \mathbf{X}^T \Delta I \mathbf{I}_t
\]
Block Based Optical Flow
Block Based

- Select a patch $P$ image at time $t$
- Search for $P$ at frame $t+1$ in a larger neighborhood

Block Search

- Search for patch in overlapping windows
- Compute similarity of intensity values between original patch and search patch
- Select the location with highest similarity
- Distance vector between centroids give optical flow vector
Computing Similarity

- Sum of square differences: $\text{SSD} = \sum \sum (I_{i,i} - I_i)^2$
- Absolute difference: $\text{AD} = \sum \sum |(I_{i,i} - I_i)|$
- Cross correlation: $\text{CC} = \sum \sum I_{i,i} I_i$
- Normalized correlation: $\text{NC} = \frac{\sum \sum (I_{i,i} - \mu_i)(I_i - \mu_i)}{\sqrt{\sum \sum I_{i,i} I_i}}$
- Mutual correlation: $\mu, \sigma$ are region mean and stdv

Correlation Surface

- Using Cross correlation: $\text{CC} = \sum \sum I_{i,i} I_i$

Find peak in correlation surface

$\mu, \sigma$ are region mean and stdv
Issues With Correlation

- Patch Size
- Search Area
- How many peaks
- Computationally expensive
  - Same operations in Fourier domain takes less time
    - Take FFT of image patch and search area
    - Multiply Fourier coefficients to construct corr. surface
    - Find maximum
- Should use pyramids here too for large displacements

Token Based Optical Flow
Tokens

- Interest points
  - Movarec’s operator
- Corners
  - Harris corner detector
- Edges
  - Any edge detection algorithm

Overview

- Given two consecutive images
- Find tokens in both images
- Find token correspondences
A graph $G(V,E)$ is a triple consisting of a vertex set $V$ an edge set $E$ and a relation that associates two vertices with an edge.

Bipartite graph: A graph $G$ is bipartite if its vertex set can be partitioned in two subsets such that no two vertex in same set have a common edge.
Finding Correspondence

- Finding matching: Matching is a set of edges such that no two of them have a common vertex.

Token Based Optical Flow

- Tokens correspond to vertices in the bipartite graph.
- Tokens at time instants $t$ and $t+1$ form partite sets of graph.
- The cost of corresponding a point at instant $t$ to a point at instant $t+1$ is the weight of edge between the corresponding vertices.
Defining Weights

Maximum Speed  Common Motion  Minimum Velocity

Consistent Match  Model

Weights

Ullman: \[ w_y = \| y_j - x_i \| \]  (Absolute distance between points)

Sethi & Jain: \[
\begin{align*}
w_y &= c \left[ \frac{1 - (y_j - x_i) \cdot v_{xy}}{\| y_j - x_i \| \cdot \| v_{xy} \|} \right] + (1 - c) \left[ 1 - \frac{GM(\| y_j - x_i \|, \| v_{xy} \|)}{AM(\| y_j - x_i \|, \| v_{xy} \|)} \right]
\end{align*}
\]

Rangarajan & Shah: \[
\begin{align*}
w_y &= \frac{\sum_{p=1}^{n_y} \sum_{q=1}^{n_j} \| y_q - (y_j - x_i) \|}{\sum_{p=1}^{n_y} \sum_{q=1}^{n_j} \| y_q - x_i \| + \sum_{k=1}^{n_x} \sum_{l=1}^{n_k} \| x_l - y_i \|}
\end{align*}
\]

geometric mean \( GM(a, b) = \sqrt{ab} \)

arithmetic mean \( AM(a, b) = \frac{a + b}{2} \)

Alper Yilmaz, Fall 2005 UCF
Algorithm

- Find initial correspondences using correlation
- Compute costs $w_{ij}$ for each pair of points $x_i, y_j$
  - Construct a bipartite graph based on computed costs
  - Prune all edges having weights exceeding certain threshold
    - Define cost matrix
- Find the minimum matching of constructed graph.
  - Hungarian Algorithm
  - Greedy search

Greedy Algorithm

Algorithm A

1. For $k = 2$ to $n - 1$ do
   
   (a) Construct $M$ an $(m \times m)$ matrix, with the points from $k$th frame along the rows and points from $(k + 1)$th frame along the columns. Let $M[i, j] = \delta(\chi_{k}^{(i)} \chi_{k+1}^{(j)})$, when $\delta^{(i)}(p) = i$.
   
   (b) for $a = 1$ to $m$ do
   i. Identify the minimum element $[i, l_a]$ in each row $i$ of $M$.
   ii. Compute priority matrix $P$, such that $B[i, l_a] = \sum_{j=1}^{m} M[i, j] + \sum_{j=i}^{m} M[i, j]$ for each $i$.
   iii. Select $[i, l_a]$ pair with highest priority value $B[i, l_a]$, and make $\delta^{(i)}(i) = l_a$.
   iv. Mask row $i$ and column $l_a$ from $M$.

Figure 5.8: Motion correspondence using multiple frames.