Problem Statement

- Given a set of sequential images, reliably track features across the sequence, while monitoring the quality of each feature.
Paper Overview

- Feature Selection
  - Fundamental definition of the Harris corner method
- Tracking System
  - Anandan’s Approach limited to only a pure translation model
  - Ability to monitor the goodness of a feature throughout tracking process
    - Anandan’s approach using full affine parameters (deformation and translation) to measure the dissimilarity between first and the current frame
    - Keep/Abandon features based on dissimilarity measure
- Detect occlusions, disocclusions, and features that do not have real-world correspondence
- Constraint: Inter-frame displacement is small

Terminology

- Occlusions
  - Shape to Detect
  - Shape not occluded
  - Shape is occluded

- Disocclusions:
  - Areas occluded in original reference frame but visible in current view
  - Detect “J”
  - Detected “J”
  - Disocclusion
  - More Disocclusion
**Terminology**

- Non-real world points

**Given Sequence**

Antenna and mirror support bar create a feature which does not correlate to a real-world feature

- Feature Detection is unable to discern depth
- Need to monitor features to track reliably

**Feature Selection**

- Many feature selection options being debated in early 1990’s
  - Most measure the amount of texturedness or cornerness in a window
  - Windows with high spatial frequency content
  - High standard deviation on the spatial intensity profile
  - Presence of zero crossings of the Laplacian of the image intensity
  - Regions where second-order derivatives are above a threshold
  - Corner detection
  - Even a window rich in texture can be a poor point to track
    - Non real-world point, occlusion/disocclusion, reflective surface, shadows, etc.
    - Tracking based solely on one of the above methods will most likely be unsuccessful and error-prone
- Paper proposes a fundamental definition for feature quality
  - i.e. Harris Corner Method
  - Used for initial feature selection, not for further tracking
Feature Selection

Basic Harris Corner Method

1. Given an image
2. Smooth image with Gaussian Filter
3. Compute derivatives \((g_x)\) and \((g_y)\) for smoothed image
4. Option: Smooth derivative images \((g_x)\) and \((g_y)\)
5. For each pixel in the image space, compute the gradient moment matrix, using the \(n \times m\) neighborhood of pixels (window) around current pixel.

\[
M = \int_{W} Zw dxdy \quad \text{where,} \quad Z = \begin{bmatrix}
    g_x^2 & g_x g_y \\
    g_x g_y & g_y^2
\end{bmatrix}
\]

\(W = \text{window (neighborhood)} = n \times m = \text{i.e.} \ 5 \times 5, 25 \times 25, \text{etc.}
\)

\(w = 1, \text{OR a 2D Gaussian weighting scheme}
\)

OR,

\[
M = \begin{bmatrix}
    \sum_i \sum_j g_x^2 w & \sum_i \sum_j g_x g_y w \\
    \sum_i \sum_j g_x g_y w & \sum_i \sum_j g_y^2 w
\end{bmatrix}
\]

6. Compute the two Eigen values for the gradient moment matrix \(M\)

- Two requirements must be upheld for the matrix \(M\)
- Above the Noise Level
  - Both Eigen values must be large
- Well-Conditioned
  - Eigen values cannot differ by several orders of magnitude

7. Select the minimum Eigen value

\[
\min (\lambda_1, \lambda_2) > \lambda_{\text{threshold}}
\]

- Smaller Eigen value meets noise-level-criterion
- Well-conditioned because intensity variations are bounded by image intensity range (i.e. 0-255).

8. Store the minimum Eigen value for each pixel in the image
9. Apply a type of Non-Maximum Suppression to the Eigen values
10. Threshold Suppressed Eigen value space to reduce amount of detected interest points

Alternative Computation to 6.7:

\[
R = \det(M) + k \text{trace}(M)^2 > \text{Threshold}
\]

Feature Selection

<table>
<thead>
<tr>
<th>(\lambda_1)</th>
<th>(\lambda_2)</th>
<th>Texturedness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Small</td>
<td>Constant intensity profile (nothing)</td>
</tr>
<tr>
<td>Small</td>
<td>Large</td>
<td>Unidirectional texture pattern (edge)</td>
</tr>
<tr>
<td>Large</td>
<td>Small</td>
<td>Unidirectional texture pattern (edge)</td>
</tr>
<tr>
<td>Large</td>
<td>Large</td>
<td>Corner, salt-and-pepper texture, (texture can be tracked reliably)</td>
</tr>
</tbody>
</table>
What is Next?

- Feature Selection used for initial detection only
- How to Track?
- Affine Motion Model
  - Last Semester Project: Anandan’s Approach

![Diagram showing starting image, warped image, and goal image with numerous iterations.]

Starting Image → Warped Image = Goal Image

- Inter-frame displacement is relatively small
- Brightness constancy constraint
- Uses
  - Image registration
  - Mosaics/Panoramic views
  - Morphing technology
  - Tracking (uses pure translation of affine motion model)
  - Measuring quality of tracked feature (complete affine model)
- Authors apply Anandan’s approach to neighborhood around features

Affine Motion Model

- Affine model for one pixel

![Diagram showing image at time t, image at time t+1, and the transformation formulae.]

\[
\begin{align*}
\text{Affine motion}: \\
&u(x, y) = a_1 x + a_2 y + b_1 \\
v(x, y) = a_3 x + a_4 y + b_2 \\
\text{Affine Transformation:} \\
x'' = (a_1 + 1)x + a_2 y + b_1 \\
y'' = a_3 x + (a_4 + 1)y + b_2
\end{align*}
\]

Affine motion parameters:

\[\{a_1, a_2, b_1, a_3, a_4, b_2\}\]
**Affine Motion Model**

- Affine model handles translation, rotation, rigid rotation and translation, affine, and shear.

<table>
<thead>
<tr>
<th>Translation</th>
<th>Rotation</th>
<th>Rigid</th>
<th>Shear</th>
<th>Affine</th>
</tr>
</thead>
</table>

\[
\begin{bmatrix}
  u \\ v
\end{bmatrix} = \begin{bmatrix}
  a_1 & a_2 \\ a_3 & a_4
\end{bmatrix} \begin{bmatrix}
  x \\ y
\end{bmatrix} + \begin{bmatrix}
  b_1 \\ b_2
\end{bmatrix}
\]

For a specific example:

\[
u(x, y) = a_x x + a_y y + b_1 \\
v(x, y) = a_{1x} x + a_{1y} y + b_2
\]

\[
\begin{bmatrix}
  u(x, y) \\ v(x, y)
\end{bmatrix} = \begin{bmatrix}
  x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & x & y & 1
\end{bmatrix} \begin{bmatrix}
  a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2
\end{bmatrix}
\]

\[
u(x) = X(x) a
\]

where,

\[
x = \begin{bmatrix}
  x \\ y
\end{bmatrix} \quad u(x) = \begin{bmatrix}
  u(x, y) \\ v(x, y)
\end{bmatrix}
\]

\[
a^T = \begin{bmatrix}
  a_1 & a_2 & b_1 & a_3 & a_4 & b_2
\end{bmatrix}
\]

\[
X(x) = \begin{bmatrix}
  x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & x & y & 1
\end{bmatrix}
\]
**Affine Motion Model**

- **Optical Flow Equation**
  \[ I_x u + I_y v = -I_t \rightarrow \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t \rightarrow \Delta I^T u = -I_t \]

- **Energy Functional**
  \[ E(u) = \sum_w (I_t + \Delta I^T u)^2 \]
  \[ E(a) = \sum_w (I_t + \Delta I^T Xa)^2 \]

- Minimize energy by taking derivative and setting it equal to zero

**Affine Motion Model**

\[ E(a) = \sum_w (I_t + \Delta I^T Xa)^2 \]

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

\[ \sum_w X^T \Delta I \Delta I_t + \sum_w X^T \Delta I \Delta I^T Xa = 0 \]

\[ \sum_w X^T \Delta I \Delta I^T Xa = - \sum_w X^T \Delta I \Delta I_t \]
Affine Motion Model

\[
\sum_{W} X^T \Delta I \Delta I^T X a = - \sum_{W} X^T \Delta I \Delta I, \quad \begin{array}{c} K \end{array}
\]

\[
K_{6 \times 6} a_{6 \times 1} = L_{6 \times 1} \quad \rightarrow \quad a = K^{-1} L
\]

- Update previous \(a\) with new \(a\)
  - Concatenation procedure
- Iteratively solve for affine parameters \(a\) until updates do not change or some iteration limit is reached

Affine Motion Model

- Author’s method similar to Anandan’s
  - Affine Motion
    \[
    \delta = D x + d \quad D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \quad d = \begin{bmatrix} d_x \\ d_y \end{bmatrix}
    \]
    equivalent to:
    \[
    \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}
    \]
  - Affine Transformation
    - A point \(x\) in the first image, \(I\), moves to a point \(Ax+d\) in the second image \(J\), where \(A = I + D\) and \(I\) is the \(2 \times 2\) identity matrix
      \[
      J \left( Ax + d \right) = I \left( x \right) \quad (2)
      \]
Tracking

- Given two images I and J
- Tracking means computing $D$ and $d$
- Quality of computation depends on
  - Size of feature window
  - Texturedness inside the feature window
  - Amount of camera/object motion between frames
- When window is small, or when inter-frame motion is small, $D$ is harder to estimate
  - Variations of motion within window are small
  - $D$ is not reliable
- However, small windows are preferred for tracking
  - Less likely to straddle depth discontinuity
- Therefore, a pure translational model is used for tracking
  - $D$ is assumed to be zero

$$\delta = d$$

Two Models of Image Motion

1. Affine Model ($D + d$)
2. Pure Translation Model ($d$)

- Use Pure Translation for tracking
  - Higher reliability
  - Higher accuracy
  - Inter-frame motion tends to be small
  - Less computations

- Use Affine Motion to monitor quality of features
  - Between first and current frame
  - Not computed every frame! Every $n^{th}$ frame
Computing Image Motion

- Both motion models measure **dissimilarity** between frames
  - Find an A and d that minimizes this dissimilarity
  - Increasing number of iterations for model can improve dissimilarity parameter

\[
\epsilon = \int \int_W \left[ J(Ax + d) - I(x) \right]^2 \cdot w(x) \, dx
\]  

- W = window (neighborhood) = n x m = i.e. 5 x 5, 25 x 25, etc.
- w = 1, OR a 2D Gaussian weighting scheme

To minimize (3), take derivative and set equal to zero

Linearize result by a truncated Taylor series
  - Due to this truncation, method must be solved iteratively

Linearization yields,

\[
T'_{6 \times 6} \mathbf{z}_{6 \times 1} = \mathbf{a}_{6 \times 1} \quad \text{(5) Affine motion Dissimilarity}
\]

where \( \mathbf{z} \) is comprised of affine parameters, D and d

\[
\mathbf{z}^T = \begin{bmatrix} d_{xx} & d_{yx} & d_{xy} & d_{yy} & d_x & d_y \end{bmatrix}
\]

and \( \mathbf{a} \) is the error vector,

\[
\mathbf{a} = \int \int_W \left[ I(x) - J(x) \right] \cdot \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} \cdot w(x) \, dx
\]

This method of calculation requires two images and is therefore not used
Computing Image Motion

- T can be computed from one image

\[
T = \int \int w \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w(x) dx
\]  

(6)

\[
U = \begin{bmatrix}
  x^2 g_x^2 & x^2 g_x g_y & x^2 g_y^2 & x^2 g_x g_y^2 \\
  x^2 g_x g_y & x^2 g_y^2 & x^2 g_x^2 & x^2 g_y^2 \\
  x^2 g_x g_y & x^2 g_y g_x & x^2 g_x^2 & x^2 g_y^2 \\
  x^2 g_x g_y & x^2 g_y g_x & x^2 g_y^2 & x^2 g_x^2
\end{bmatrix}
\]

\[
Z = \begin{bmatrix}
  g_x^2 & g_x g_y \\
  g_y g_x & g_y^2
\end{bmatrix}
\]

\[
V^T = \begin{bmatrix}
  x g_x^2 & x g_y g_x & y g_x^2 & y g_y g_x \\
  x g_y g_x & x g_y^2 & y g_x g_y & y g_y^2 \\
  x g_x g_y & x g_y g_x & y g_x g_y & y g_y g_x \\
  x g_y g_x & x g_y g_x & y g_x g_y & y g_y g_x
\end{bmatrix}
\]

D and d interaction in matrix V
\[
\therefore \text{errors in D seep into d}
\]

Computing Image Motion

- For Pure Translation Model

\[
Zd = e
\]  

(7) Pure Translation Dissimilarity

\[
Z = \begin{bmatrix}
  g_x^2 & g_x g_y \\
  g_y g_x & g_y^2
\end{bmatrix}
\]

\[
d = \begin{bmatrix}
  d_x \\
  d_y
\end{bmatrix}
\]

- Same Z used to compute Eigen values in corner detector
- Derivation by Stan Birchfield (developed KLT program)
Feature Selection
Harris Corner Detector
Tracking
Pure Translation

Select new feature

Feature Selection
Harris Corner Detector

Select new feature

Perform
\[ Zd = e \]

continue update?

no

More features?

no

Not n\(^{th}\) frame?

no

Select new feature

Monitor Quality of Feature
Affine Model

\[ Tz = a \]

continue update?

yes

no

Discard feature?

yes

no

Discard

More features?

yes

no

Select new feature

More features?

yes

no

no

START

A

A

Dissimilarity

- Not all features are good to track & some features are only good to track for a while
- **Dissimilarity** indicates possible change in feature (becomes a bad feature)
- Typical video spans a large number of frames
  - Pure translational model good for inter-frame tracking
  - Pure translation dissimilarity measure not good across a large number of frames
  - Affine dissimilarity better measures the quality of features across frame range

**Example 1: Woody Allen’s *Manhattan***

1\(^{st}\) frame 11\(^{th}\) frame 21\(^{st}\) frame

Sign mostly translates, but does increase size by 15%

Tracked

Affine warping

1 6 11 16 21

Dashed line = Pure Translation
Solid Line = Affine Transformation
Dissimilarity

Example 2: Woody Allen’s *Manhattan*

- Glass window becomes occluded in middle frame
- Dissimilarity spike in affine transformation curve at frame 5 indicates occlusion
- Affine warping tries to deform traffic sign into a window

Convergence

- Dissimilarity looked at an entire sequence of frames
  - Many affine dissimilarity measurements computed
- Convergence: comparing the first and current frames
  - Fitting current frame (source) to first frame (destination)
  - One dissimilarity measurement
  - Iterative method
  - Leftmost column: source
  - Rightmost column: destination
    - 16% Gaussian noise added
  - Middle cols: after 4, 8, & 19 iterations

<table>
<thead>
<tr>
<th>Source</th>
<th>4th iter</th>
<th>8th iter</th>
<th>19th iter</th>
<th>Dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Convergence

Comparisons for previous slide

<table>
<thead>
<tr>
<th></th>
<th>True Deformation</th>
<th>Computed Deformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.499 0.342 0.562</td>
<td>1.393 0.318 0.562</td>
</tr>
<tr>
<td>2</td>
<td>0.342 0.342 0.655</td>
<td>0.670 0.339 0.608</td>
</tr>
<tr>
<td>3</td>
<td>0.253 0.253 1.252</td>
<td>0.802 0.235 0.937</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>True Translation</th>
<th>Computed Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 0</td>
<td>3.8765 0.0007</td>
</tr>
<tr>
<td>2</td>
<td>2 0</td>
<td>2.0920 0.0155</td>
</tr>
<tr>
<td>3</td>
<td>3 0</td>
<td>3.8591 0.0317</td>
</tr>
</tbody>
</table>

Penny Example

Source 4th iter 8th iter 19th iter Dest.

Blobs to Cross Example

Real world image sequence
- 26 frame sequence
- Camera moves forward
- Objects become larger
- Due to depth issue, the following will occur
  - Occlusions
  - Disocclusions
  - Non-real points
- 102 features selected
- Limited # features by prohibiting overlapping feature windows during feature selection process
Monitoring Features

- Pure translation is sufficient for inter-frame tracking
  - Not for monitoring
  - All features, except two, have comparable dissimilarities
  - No way to distinguish good from bad features

Affine Motion Dissimilarity

- Good for monitoring
- Seven features have high dissimilarity, thus bad and are discarded
- Thick band of curves at bottom represents all good features (keep)

KLT Demo