Corners
Sum of squared differences

• Say we want to track a patch from image $I_0$ to image $I_1$
  – We’ll assume that intensities don’t change, and only translational motion
  – So we expect that
    $$I_1(x_i + u) = I_0(x_i)$$
  – In practice we will have noise, so they won’t be exactly equal
• But we can search for the displacement $u=(x, y)$ that minimizes the sum of squared differences
  $$E(u) = \sum w(x_i) [I_1(x_i + u) - I_0(x_i)]^2$$
Stability of SSD

• We would like the SSD score to have a distinct minimum at the correct value of $u$
  – If not, there will be uncertainty in the value of $u$

• The image texture in the patch will determine the stability of the SSD score
  – Bland, featureless patches will not be able to be matched uniquely

• We want to choose patches to track that will be stable

• We can predict how stable a patch will be by comparing its SSD score with itself, at various displacements $\Delta u$

$$E(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2,$$
where $\Delta u = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$

• If the surface $E(\Delta u)$ has a distinct minimum, it is a good feature to track

$w(x)$ is optional weighting function
Figure 4.5 Three auto-correlation surfaces $E_{AC}(\Delta u)$ shown as both grayscale images and surface plots: (a) The original image is marked with three red crosses to denote where the auto-correlation surfaces were computed; (b) this patch is from the flower bed (good unique minimum); (c) this patch is from the roof edge (one-dimensional aperture problem); and (d) this patch is from the cloud (no good peak). Each grid point in figures b–d is one value of $\Delta u$.

Stability of SSD Score

• We can estimate the stability of the SSD score
• First, take the Taylor series expansion of the image
  – Recall Taylor series expansion of a function $f(x)$ near $x_0$
    $$f(x) = f(x_0) + \left[ \frac{df}{dx} \right]_{x_0} (x - x_0) + \cdots$$
  – For vectors $\mathbf{x}, \mathbf{u}$
    $$I_0(\mathbf{x}_i + \Delta \mathbf{u}) \approx I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u}$$

• Then
  $$E(\Delta \mathbf{u}) = \sum_i w(\mathbf{x}_i) \left[ I_0(\mathbf{x}_i + \Delta \mathbf{u}) - I_0(\mathbf{x}_i) \right]^2$$
  $$\approx \sum_i w(\mathbf{x}_i) \left[ I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u} - I_0(\mathbf{x}_i) \right]^2$$
  $$= \sum_i w(\mathbf{x}_i) \left[ \nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u} \right]^2$$

where
• $\mathbf{x}_i = (x_i, y_i)^T$ is some image point
• $\Delta \mathbf{u} = (\Delta x, \Delta y)^T$ is the displacement from that point
• $\nabla I_0(\mathbf{x}_i)$ is the gradient of the image at $\mathbf{x}_i$
Stability of SSD Score (continued)

• Expand the SSD score

\[ E(\Delta \mathbf{u}) = \sum_i w(\mathbf{x}_i) [\nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u}]^2 \]

\[ = \sum_i w(\mathbf{x}_i) \left[ \Delta x \frac{\partial I_0(\mathbf{x}_i)}{\partial x} + \Delta y \frac{\partial I_0(\mathbf{x}_i)}{\partial y} \right]^2 \]

\[ = \sum_i w(\mathbf{x}_i) \left[ \Delta x \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial x} \right)^2 \Delta x + 2 \Delta x \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial x} \right) \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial y} \right) \Delta y + \Delta y \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial y} \right)^2 \Delta y \right] \]

\[ = \Delta x \left[ \sum_i w(\mathbf{x}_i) \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial x} \right)^2 \right] \Delta x + 2 \Delta x \left[ \sum_i w(\mathbf{x}_i) \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial x} \right) \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial y} \right) \right] \Delta y + \Delta y \left[ \sum_i w(\mathbf{x}_i) \left( \frac{\partial I_0(\mathbf{x}_i)}{\partial y} \right)^2 \right] \Delta y \]

\[ = \Delta x \left( w* I_{xx} \right) \Delta x + 2 \Delta x \left( w* I_{xy} \right) \Delta y + \Delta y \left( w* I_{yy} \right) \Delta y \]

\[ = \left( \begin{array}{cc} \Delta x & \Delta y \end{array} \right) \left( \begin{array}{cc} w* I_{xx} & w* I_{xy} \\ w* I_{xy} & w* I_{yy} \end{array} \right) \left( \begin{array}{c} \Delta x \\ \Delta y \end{array} \right) \]

• where

\[ I_{xx}^2 = \left( \frac{\partial I}{\partial x} \right)^2, \quad I_{xy} = \left( \frac{\partial I}{\partial x} \right) \left( \frac{\partial I}{\partial y} \right), \quad I_{yy}^2 = \left( \frac{\partial I}{\partial y} \right)^2 \]
Stability of SSD Score (continued)

- The SSD score is
  \[
  E(\Delta u) = (\Delta x \quad \Delta y) \begin{pmatrix}
  w * I_{xx} & w * I_{xy} \\
  w * I_{yx} & w * I_{yy}
  \end{pmatrix} (\Delta x \quad \Delta y)
  \]
  \[
  = \Delta u^T A \Delta u
  \]

- where

\[
A = w * \begin{pmatrix}
  I_x^2 & I_{xy} \\
  I_{yx} & I_y^2
  \end{pmatrix}
\]

- \( w \) is a small \( N \times N \) averaging filter such as a box filter
- it essentially averages the values over a small local neighborhood

- \( A \) can tell us how stable the SSD score is, at a given point
Example Patches

\[
A = w \begin{pmatrix}
I_x^2 & I_{xy} \\
I_{xy} & I_y^2
\end{pmatrix}
\]
Stability of SSD Score (continued)

- \( E = \Delta u^T A \Delta u \) is a second order polynomial
  - Has the form \( E(x,y) = ax^2 + bxy + cy^2 \)
  - The matrix \( A \) gives the curvature of the surface

\[
A = \begin{pmatrix}
a & b \\
b & c \\
\end{pmatrix}
\]

- A large value of \( a \) means that \( E \) curves up sharply as \( x \) varies
  - Similarly, large \( c \) means that \( E \) curves up sharply as \( y \) varies
  - Both \( a,c \) large mean that we have a good minimum in \( E \)
- Problem – if \( b \) is not zero, the surface may be flat along the diagonal
Estimating stability (continued)

• Recall eigenvalues
  \[ A \mathbf{v}_i = \lambda_i \mathbf{v}_i \]
  where \( \lambda_i \) is the \( i \)th eigenvalue, \( \mathbf{v}_i \) is the \( i \)th eigenvector
  – We can write

\[ A \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} \]

  – Or \( AR = \Lambda R \)

• Now, \( R = (\mathbf{v}_1, \mathbf{v}_2) \) is an orthonormal matrix (its columns are mutually orthogonal, and all unit vectors). So \( R^T = R^{-1} \)

• So \( R^T A R = \Lambda \)
  – Where the columns of \( R \) are the eigenvectors of \( A \)
  – \( \Lambda \) is a diagonal matrix whose elements are the eigenvalues of \( A \)
Estimating stability (continued)

- **$R$** is a rotation matrix
  - If we rotate the coordinate system to $\Delta u = R \Delta v$
  - Then $E = \Delta u^T A \Delta u = (\Delta v^T R^T) A (R \Delta v) = \Delta v^T \Lambda \Delta v$
  - Which has the form $E(v_1, v_2) = \lambda_1 v_1^2 + \lambda_2 v_2^2$

- So if both eigenvalues are large, we have a good minimum (there is no diagonal component)

- If one eigenvalue is low, surface is flat along that direction
Possible interest point measures

• Look at the smallest eigenvalue – if it is sufficiently large, then we have a candidate point (Shi-Tomasi)

• Alternative measures that don’t require square roots:

\[
\text{(Harris)} \quad \det(A) - \alpha \, \text{tr}(A)^2 = \lambda_0 \lambda_1 - \alpha (\lambda_0 + \lambda_1)^2
\]

\[
\text{(Harmonic mean)} \quad \frac{\det(A)}{\text{tr}(A)} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}
\]

Matlab Corner Interest Operator

% Find corner points
clear all
close all

I = double(imread('test000.jpg'));

% Apply Gaussian blur
sd = 1.0;
I = imfilter(I, fspecial('gaussian', round(6*sd), sd));

imshow(I, []);

% Compute the gradient components
Gx = imfilter(I, [-1 1]);
Gy = imfilter(I, [-1; 1]);

% Compute the outer product g'g at each pixel
Gxx = Gx .* Gx;
Gxy = Gx .* Gy;
Gyy = Gy .* Gy;
Matlab Corner Interest Operator (continued)

% Size of neighborhood over which to compute corner features.
N = 13;

w = ones(N);  % The neighborhood

% Sum the G's over the window size.
A11 = imfilter(Gxx, w);
A12 = imfilter(Gxy, w);
A22 = imfilter(Gyy, w);

% At each pixel (x,y), we have the 2x2 matrix
% [A11(x,y)  A12(x,y);
%  A21(x,y)  A22(x,y)]
% Of course, A21 = A12.

% Compute the interest score: det(A)/trace(A)
detA = A11.*A22 - A12.^2;  % Computes det(A) at each pixel
traceA = A11 + A22;  % Computer trace(A) at each pixel
s = detA./traceA;  % The interest score

% If trace(A)=0 anywhere, we get "not a number".
s(isnan(s)) = 0;  % If any NaN's, just set them to zero
Finding Local Maxima

- Simply finding all points greater than a threshold will yield too many points

- We should find points that are
  - greater than the threshold
  - a local maximum
Finding Local Maxima

• If peaks are smooth, you can test each point to see if it is higher than its neighbors

\[
\text{Lmax} = \text{false(size(S));}
\]
\[
\text{Lmax}(S(2:end-1, 2:end-1)) = \ldots 
\]
\[
( S(2:end-1, 2:end-1) > S(3:end, 2:end-1) \ldots 
\] & \[
S(2:end-1, 2:end-1) > S(1:end-2, 2:end-1) \ldots 
\]
\[
\& S(2:end-1, 2:end-1) > S(2:end-1, 3:end) \ldots 
\]
\[
\& S(2:end-1, 2:end-1) > S(2:end-1, 1:end-2) 
\]
Finding Local Maxima (continued)

- Problem – what if you don’t have smooth peaks?
  - Too many detected features

- Instead, find points that are local **regional** maxima – they are the largest points within a region (e.g., square of size $W$)

- Can do by calculating the largest value within distance $N$ (this is a **dilation**)

- Then find those points that equal the local maxima

$$S_{\text{max}} = (S == \text{imdilate}(S, \text{ones}(N)))$$
Finding Local Maxima (continued)

Score array

$S$

Points where $S = S_d$

$S_{\text{max}}$

Score array dilated with square of side 15

$S_d$

Points where $S = S_d$ and are $> \text{threshold}$
% Choose the suppression radius, for non-maxima suppression
r = N;

% Find local maxima within each neighborhood of radius 2r
Lmax = (s==imdilate(s, strel('disk',2*r)));

% Note - we don't want to detect points too close to the border, so just
% zero out everything near the border.
Lmax(1:N,:) = false;
Lmax(:,1:N) = false;
Lmax(end-N:end,:) = false;
Lmax(:,end-N:end) = false;

% Get a list of the indices of all the potential interest points
[rows cols] = find(Lmax);

% Get the values of those interest points
vals = s(Lmax);

Look at range of scores
Identifying interest points

- Once you have the scores, you can decide which points to identify as interest points
  - Could choose the points with the top $M$ scores
  - Or, choose all points with scores greater than a threshold

```matlab
% Identify interest points above a threshold, and draw
% a box around them, of size N x N
for i=1:length(vals)
    if vals(i) > threshold
        x = cols(i);
        y = rows(i);

        rectangle('Position', [x-N/2 y-N/2 N N], ...
        'EdgeColor', 'r', ... 
        'Linewidth', 2.0); % default is 0.5

        text(x,y, sprintf('%d', i), ... 
        'Color', 'r', ... % label with id number 
        'FontSize', 16); % default is 10
    end
end
```
Corner interest points:

- Window size = 15
- Threshold = 100
- Local maxima within 15 pixels
Tracking Example

- Tracked points from truck-mounted camera (Hoff 2010)
- Size of window = 19x19
- Red = tracked point in 2D
- Yellow = have 3D information
Handling More General Viewpoint Changes

- Cross correlation assumes translation only
  - But appearance of image patches changes with rotation, scale, viewpoint
- Need features that are invariant to translation, rotation, scale, and viewpoint
Affine Corner Detector

• Affine transformation

\[ I_2 (A\mathbf{x} + \mathbf{d}) = I_1 (\mathbf{x}) \]

• where

\[ A = \begin{pmatrix}
    a_{xx} & a_{xy} \\
    a_{yx} & a_{yy}
\end{pmatrix} \]

  \( \mathbf{d} \) is the translation, \( A \) is the deformation matrix

• This can account for rotation and scale changes
• Also viewpoint changes, if planar patch is small compared to distance from camera
KLT Tracker

- Find interest points in the first frame
- Track (assuming translation only) points from each frame to subsequent frame
- Test consistency with original frame by computing affine transformation

from: Shi and Tomasi, “Good Features to Track”, IEEE CVPR 1994