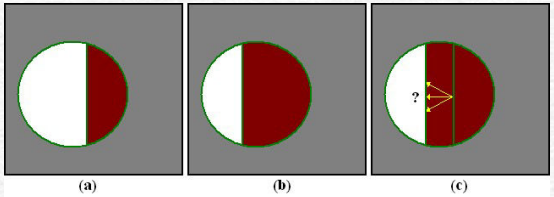
Features

- Introduction
- Moravec Corner Detection
- Harris/Plessey Corner Detection
- Scale Invariant Feature Transform (SIFT)

1

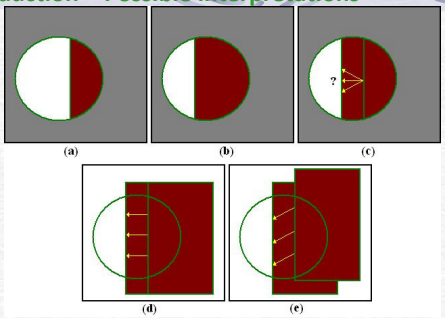
Introduction – Where have the edges gone?

- Given two images (a) and (b) taken at different times determine the movement of edge points from frame to frame…



2

Introduction – Possible interpretations

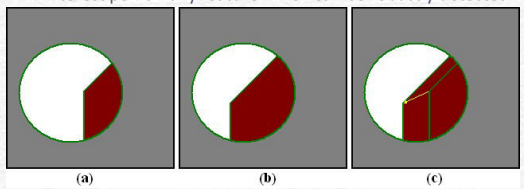


- The Aperture Problem.

3

Introduction – Using Features / Corners instead

- Use corners / image features / interest points
  - Corner = intersection of two edges
  - Interest point = any feature which can be robustly detected



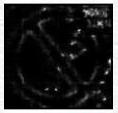
- Reduces number of points
- Easier to establish correspondences
- Spurious features

4

Introduction – Steps for corner detection

1. Determine cornerness values.
   - For each pixel
   - Main difference
   - Produces a Cornerness map.

2. Non-maxima suppression.
   - Multiple responses
   - Compare to local neighbours



3. Threshold the cornerness map.
   - Significant corners.

Moravec corner detection

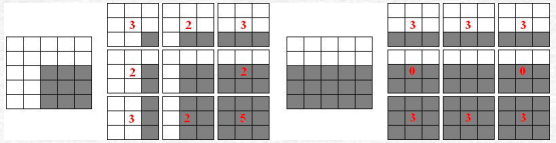
- Looks at the local variation around a point
  - Compares local image patches

$$V_{u,v}(i,j) = \sum_{\forall a,b \in Window} \left(f(i+u+a, j+v+b) - f(i+a, j+b)\right)^2$$

  - where *(u,v)* $\in$ { (-1,-1), (-1,0), (-1,1), (0,-1), (0,1) (1,-1), (1, 0), (1,1,) } and the Window is typically 3x3, 5x5 or 7x7
  - Select the Minimum value of $V_{u,v}(i,j)$

Moravec – Binary Example

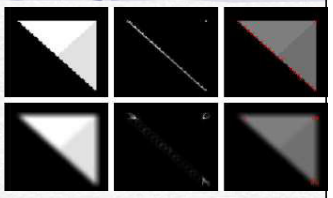
Corner: Edge:



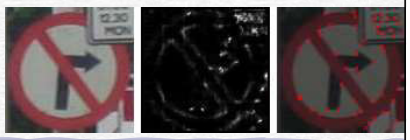
Minimum difference: 2 Minimum difference: 0

Moravec – Flaws.

- Anisotropic response
  - Diagonal lines
  - Smoothing



- Noisy response
  - Larger area
  - Smoothing



Harris / Plessey corner detection

- Difference – Cornerness determination
- Uses
  - Partial derivatives
  - Gaussian weighting
  - Matrix Eigenvalues

```
Ptr<FeatureDetector> harris_detector =
                GFTTDetector::create( 1000, 0.01, 10, 3, true );
vector<KeyPoint> keypoints;
harris_detector->detect( gray_image, keypoints );
```


Harris / Plessey corner detection

- Consider the intensity variation for an arbitrary shift (Δi, Δj) as

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} (f(i,j) - f(i-\Delta i, j-\Delta j))^2$$

- If $f(i-\Delta i, j-\Delta j) \approx f(i,j) + \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta i} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix}$
- Then $SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left(f(i,j) - f(i,j) - \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)^2$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left(\begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)^2$$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left(\begin{bmatrix} \Delta i & \Delta j \end{bmatrix} \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} \\ \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)$$

$$SSD_W(\Delta i, \Delta j) = \begin{bmatrix} \Delta i & \Delta j \end{bmatrix} \begin{bmatrix} \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta i}\right)^2 & \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} & \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta j}\right)^2 \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix}$$

Harris / Plessey corner detection

- From the matrix we can compute the Eigenvalues
  - Both high => corner
  - One high => edge
  - None high => constant region
- Harris & Stephens proposed the following cornerness measure:

$$M = \begin{bmatrix} \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta i}\right)^2 & \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i} \frac{\delta f(i,j)}{\delta j} & \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta j}\right)^2 \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

0.04	**0.12**	**0.04**
0.12	**0.36**	**0.12**
0.04	**0.12**	**0.04**

$\det(M) = \lambda_1 \lambda_2 = AC + B^2$

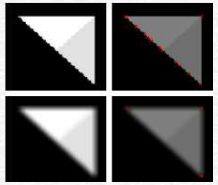
$\mathrm{trace}(M) = \lambda_1 + \lambda_2 = A + C$

$C(i,j) = \det(M) - k(trace(M))^2$

Harris / Plessey corner detection

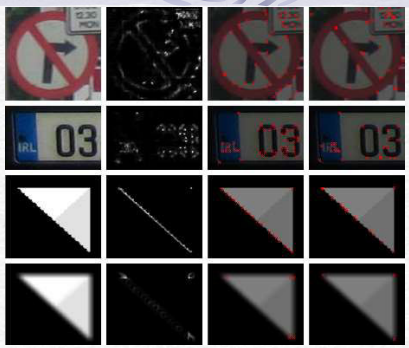
Harris / Plessey – Pros and Cons

- Cons:
  - More expensive computationally
  - Sensitive to noise
  - Somewhat anisotropic
- Pros:
  - Very repeatable response
  - Better detection rate



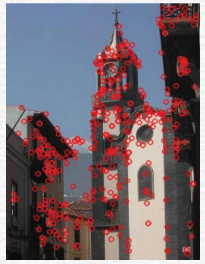
```
Mat display_image;
drawKeypoints( image, keypoints, display_image,
                                    Scalar( 0, 0, 255 ) );
```


Moravec & Harris/Plessey



Scale Invariant Feature Transform (SIFT)

- FROM "Distinctive Image Features from Scale-Invariant Keypoints",
- by David G. Lowe in International Journal of Computer Vision, 60, 2 (2004), pp.91-110
- Motivation:
  - Providing repeatable robust features for
    - Tracking,
    - Recognition,
    - Panorama Stitching, etc.



- Features:
  - Invariant to scaling, & rotation.
  - Partly invariant to illumination and viewpoint changes

SIFT – Overview & Contents

1. Scale Space Extrema Detection
   - Scale Space
   - Difference of Gaussian
   - Locate Extrema
2. Accurate Keypoint Location
   - Sub-pixel locate
   - Filter response – remove low contrast and features primarily along an edge
3. Keypoint Orientation assignment
4. Keypoint Descriptors

- Matching Descriptors – including dropping poor ones
- Applications

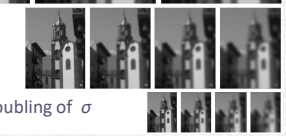
SIFT – OpenCV code

```
// The following code is valid up to v2.4.11:
Ptr<FeatureDetector> feature_detector =
                                    FeatureDetector::create("SIFT");
vector<KeyPoint> keypoints;
feature_detector->detect( gray_image, keypoints );
```


SIFT – Scale Space Extrema Detection

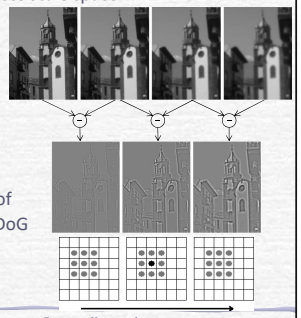
- For scale invariance, consider the image at multiple scales
- $L(x,y,\sigma)$ $L(x,y,k\sigma)$ $L(x,y,k^2\sigma)$ $L(x,y,k^3\sigma)$

- $L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$
- Applied in different octaves of scale space
- Each octave corresponds to a doubling of σ



SIFT – Scale Space Extrema Detection

- Stable keypoint locations defined to be at extrema in the Difference of Gaussian (DoG) functions across scale space…
- $D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$



- Extrema..
  - Centre point is Min or Max of
  - Local 3x3 region in current DoG
  - and in adjacent scales
  - IN any octave

SIFT – Accurate Keypoint Location

- Originally location and scale taken from central point
- Locate keypoints more precisely
  - Model data locally using a 3D quadratic
  - Locate interpolated maximum/minimum

SIFT – Accurate Keypoint Location

- Discard low contrast keypoints
  - If the local contrast is too low discard the keypoint
  - Evaluated from the curvature of the 3D quadratic…

SIFT – Accurate Keypoint Location

- Discard poorly localised keypoints (e.g. along an edge)

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

$$\alpha = r\beta$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

SIFT – Keypoint Orientation

- For scale invariance, the keypoint scale is used to select the smoothed image with the closet scale
- For orientation invariance we describe the keypoint wrt. the principal orientation
  - Create an orientation histogram (36 bins)

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

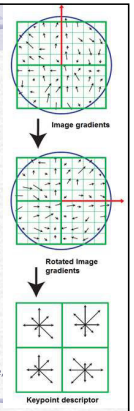
  - Weight by gradient magnitude
  - Sample points around the keypoint
  - Highest peak + peaks within 80%
    - Oriented keypoint(s)
  - Stable results….

SIFT – Keypoint Description

- Could sample image intensity at relevant scale
  - Match using normalized cross correlation
  - Sensitive to
    - affine transformations,
    - 3D viewpoint changes and
    - non-rigid deformations
- A better approach (Edelman et al. 1997)
  - Based on a model of biological vision
  - Consider gradients at particular orientations and spatial frequencies
  - Location not required to be precise

SIFT – Keypoint Description

- Use blurred image at closest scale
- Sample points around the keypoint
- Compute gradients and orientations
- Rotate by keypoint orientation
- Divide region into subregions
- Create histograms (8 bins) for subregions
  - Weight by gradient
  - Weight by location (Gaussian)
  - Distribute into bins (trilinear interpolation)

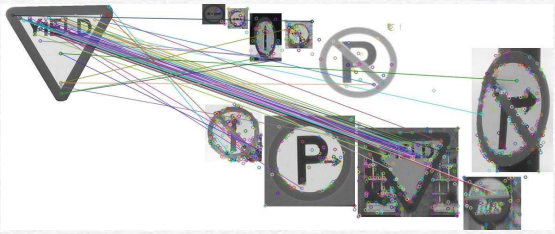


Images derived from those by ©David Lowe, reproduced with permission

SIFT – Matching Keypoints

- Nearest neighbour matching
  - Euclidean distance between keypoints
  - Use a global distance threshold?
  - Better to…
    - Compare distance to closest neighbour to distance to 2nd closest neighbour (from a different object)
    - Distance ratio > 0.8 eliminates
      - 90% false matches
      - 5% correct matches

SIFT – Matching Keypoints



SIFT – OpenCV code

```
// The following code is valid up to v2.4.11:
SiftFeatureDetector sift_detector;
vector<KeyPoint> keypoints1, keypoints2;
sift_detector.detect( gray_image1, keypoints1 );
sift_detector.detect( gray_image2, keypoints2 );
// Extract feature descriptors
SiftDescriptorExtractor sift_extractor;
Mat descriptors1, descriptors2;
sift_extractor.compute( gray_image1, keypoints1, descriptors1 );
sift_extractor.compute( gray_image2, keypoints2, descriptors2 );

…
```

SIFT – OpenCV code

```
…
// Match descriptors.
BFMatcher sift_matcher(NORM_L2);
vector< DMatch > matches;
matcher.match( descriptors1, descriptors2, matches );
// Display SIFT matches
Mat display_image;
drawMatches( gray_image1, keypoints1, gray_image2,
                                keypoints2, matches, display_image );
```

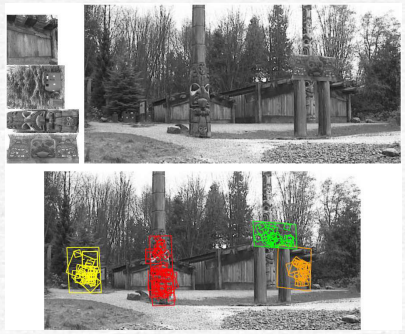

SIFT – Recognition

- Match at least 3 features
- Cluster matches
  - Hough transform
    - Location (2D)
    - Scale
    - Orientation
  - Really a 6D problem
    - Use broad bins
      - 30° for orientation
      - Factor of 2 for scale
      - 0.25 times image dimension for location
  - Consider all bins with at least 3 entries
    - Determine affine transformation

SIFT – Application: Recognition

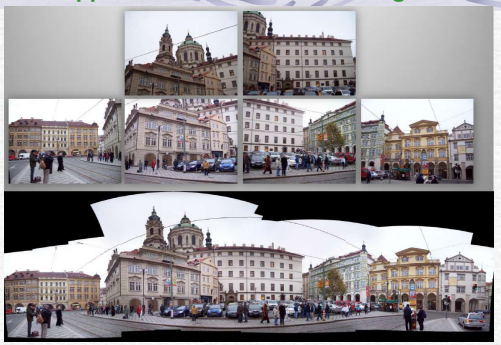
Images ©David Lowe, reproduced with permission

SIFT – Application: Recognition



Images ©David Lowe, reproduced with permission

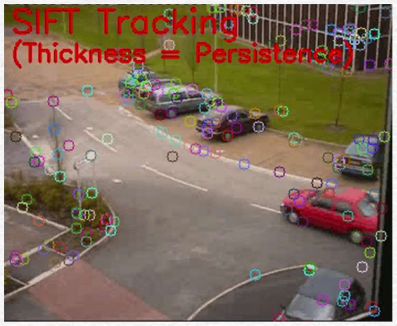
SIFT – Application: Panorama Stitching



Images © Lucas Mach, released under CC-BY 3.0

33

SIFT – Application: Tracking



34