EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

Lecture 08
130214

http://www.ee.unlv.edu/~b1morris/ecg795/

Outline

- Review
  - Points and Patches
- Feature Detectors
- Feature Descriptors
- Feature Matching
- Feature Tracking

Feature Detection and Matching

- Essential component of modern computer vision
  - E.g. alignment for image stitching, correspondences for 3D model construction, object detection, stereo, etc.
- Need to establish some features that can be detected and matched
  - Points and patches
  - Edges
  - Lines
- Which features are best?
  - Depends on the application
  - Want features that are robust
    - Descriptive and consistent (can readily detect)

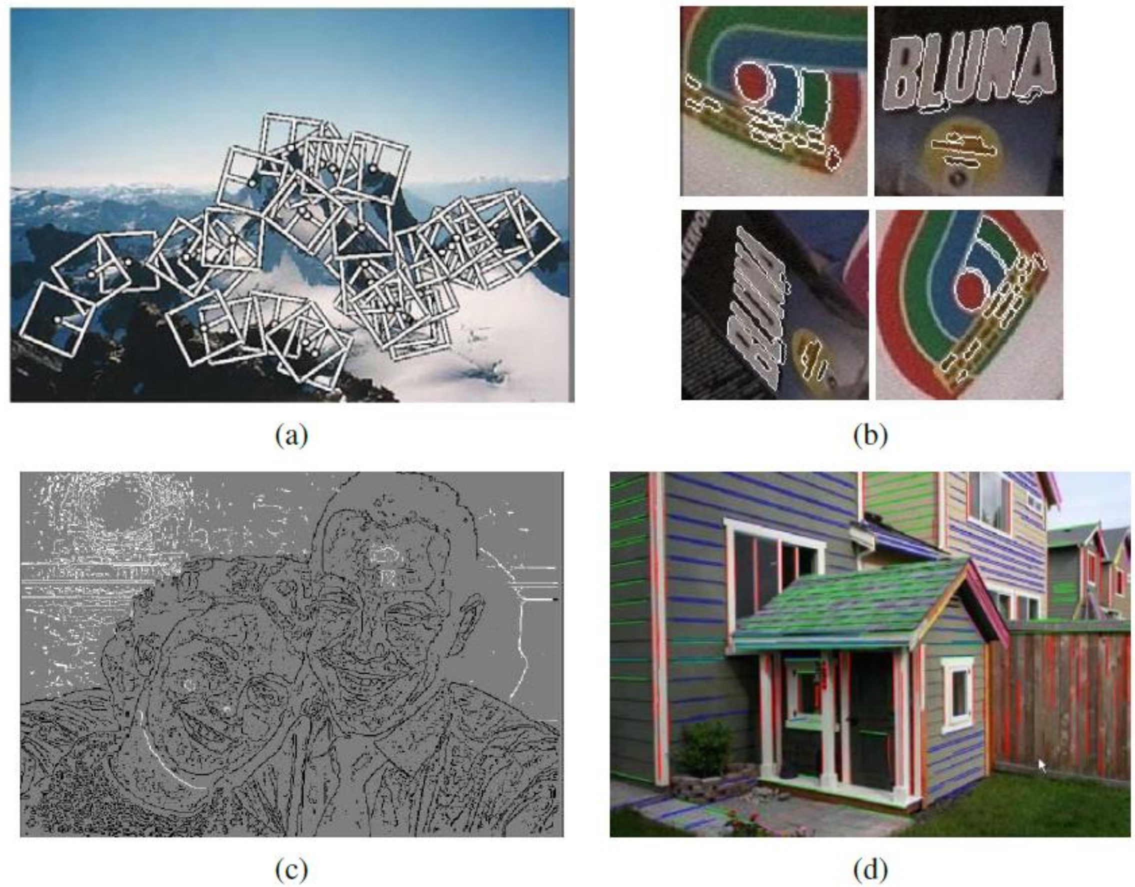


Figure 4.1 A variety of feature detectors and descriptors can be used to analyze, describe and match images: (a) point-like interest operators (Brown, Szeliski, and Winder 2005) © 2005 IEEE; (b) region-like interest operators (Matas, Chum, Urban *et al.* 2004) © 2004 Elsevier; (c) edges (Elder and Goldberg 2001) © 2001 IEEE; (d) straight lines (Sinha, Steedly, Szeliski *et al.* 2008) © 2008 ACM.

Points and Patches

- Maybe most generally useful feature for matching
  - E.g. Camera pose estimation, dense stereo, image stitching, video stabilization, tracking
  - Object detection/recognition
- Key advantages:
  - Matching is possible even in the presence of clutter (occlusion)
  - and large scale and orientation changes
- 2 General techniques
  - Detect and track – initialize features in a single image and look for them close by in next image (video)
  - Detect and match – find features in all images separately and match based on local appearance similarity (large motion or appearance change)

Keypoint Pipeline

- Feature detection (extraction)
  - Search for image locations that are likely to be matched in other images
- Feature description
  - Regions around a keypoint are represented as a compact and stable descriptor
- Feature matching
  - Descriptors are compared between images efficiently
- Feature tracking
  - Search for descriptors in small neighborhood
  - Alternative to matching stage best suited for video

Feature Detectors

- Must determine image locations that can be reliably located in another image

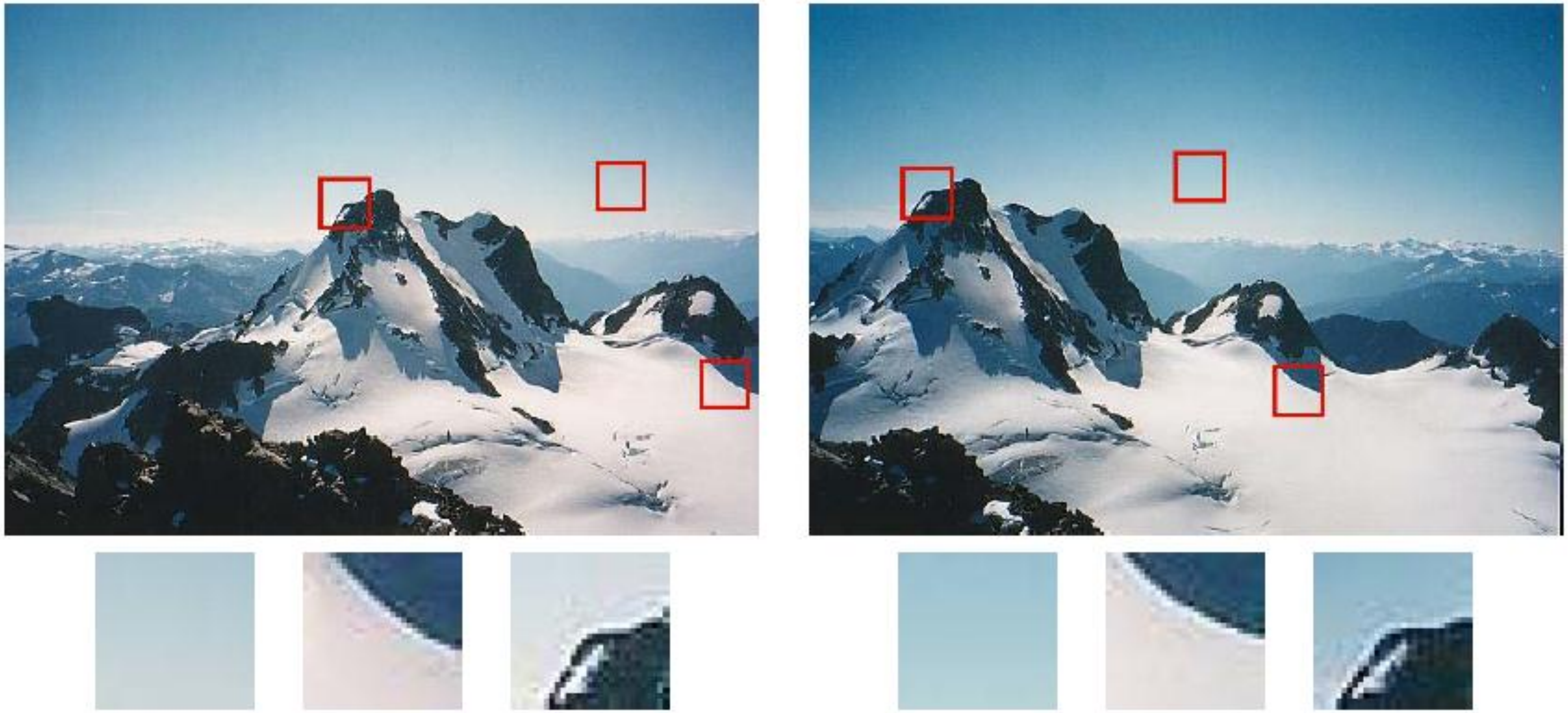


Figure 4.3 Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.

Comparison of Image Patches

- Textureless patches
  - Nearly impossible to localize and match
    - Sky region “matches” to all other sky areas


- Edge patches
  - Large contrast change (gradient)
  - Suffer from aperture problem
    - Only possible to align patches along the direction normal the edge direction


- Corner patches
  - Contrast change in at least two different orientations
  - Easiest to localize


Aperture Problem

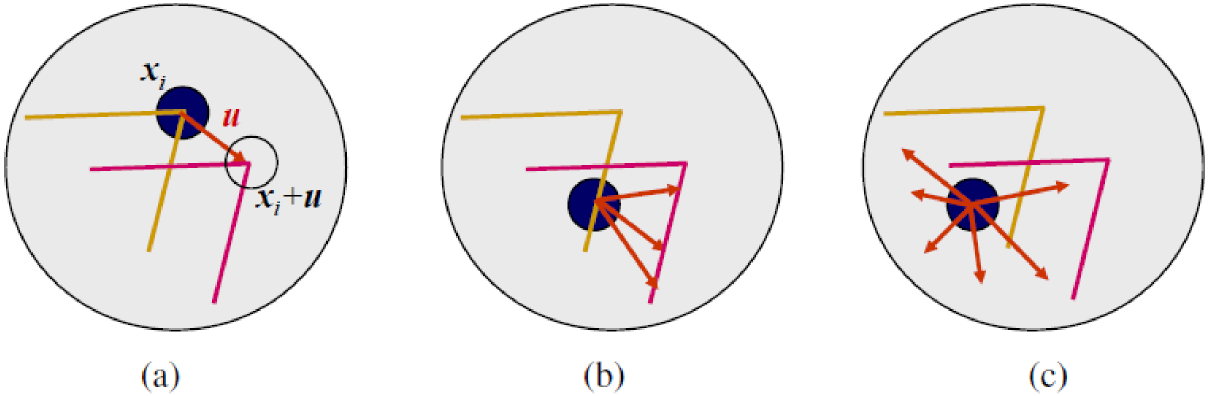


Figure 4.4 Aperture problems for different image patches: (a) stable ("corner-like") flow; (b) classic aperture problem (barber-pole illusion); (c) textureless region. The two images I_0 (yellow) and I_1 (red) are overlaid. The red vector $\boldsymbol{u}$ indicates the displacement between the patch centers and the $w(\boldsymbol{x}_i)$ weighting function (patch window) is shown as a dark circle.

- Only consider a small window of an image
  - Local view does not give global structure – causes ambiguity
- Corners have strong matches
- Edges can have many potential matches
  - Constrained upon a line
- Textureless regions provide no useful information

WSSD Matching Criterion

- Weighted summed squared difference
  - $E_{WSSD}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)\,[I_1(\boldsymbol{x}_i - \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$
    - I_1, I_0 - two image patches to compare
    - $\boldsymbol{u} = (u, v)$ – displacement vector
    - $w(\boldsymbol{x})$ - spatial weighting function
- Normally we do not know the image locations to perform the match
  - Calculate the autocorrelation in small displacements of a single image
    - Gives a measure of stability of patch – how well can a patch be distinguished
  - $E_{AC}(\Delta\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)\,[I_0(\boldsymbol{x}_i - \Delta\boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$

Image Patch Autocorrelation

$$E_{AC}(\Delta \boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)\,[I_0(\boldsymbol{x}_i - \Delta \boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$
$$= \sum_i w(\boldsymbol{x}_i)\,[\nabla I_0(\boldsymbol{x}_i) \cdot \Delta \boldsymbol{u}]^2$$
$$= \Delta \boldsymbol{u}^T A \Delta \boldsymbol{u}$$

- $\nabla I_0(\boldsymbol{x}_i)$ - image gradient
  - We have seen how to compute this
- A – autocorrelation matrix

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

  - Compute gradient images and convolve with weight function
  - Also known as second moment matrix

- Example autocorrelation

Image Autocorrelation II

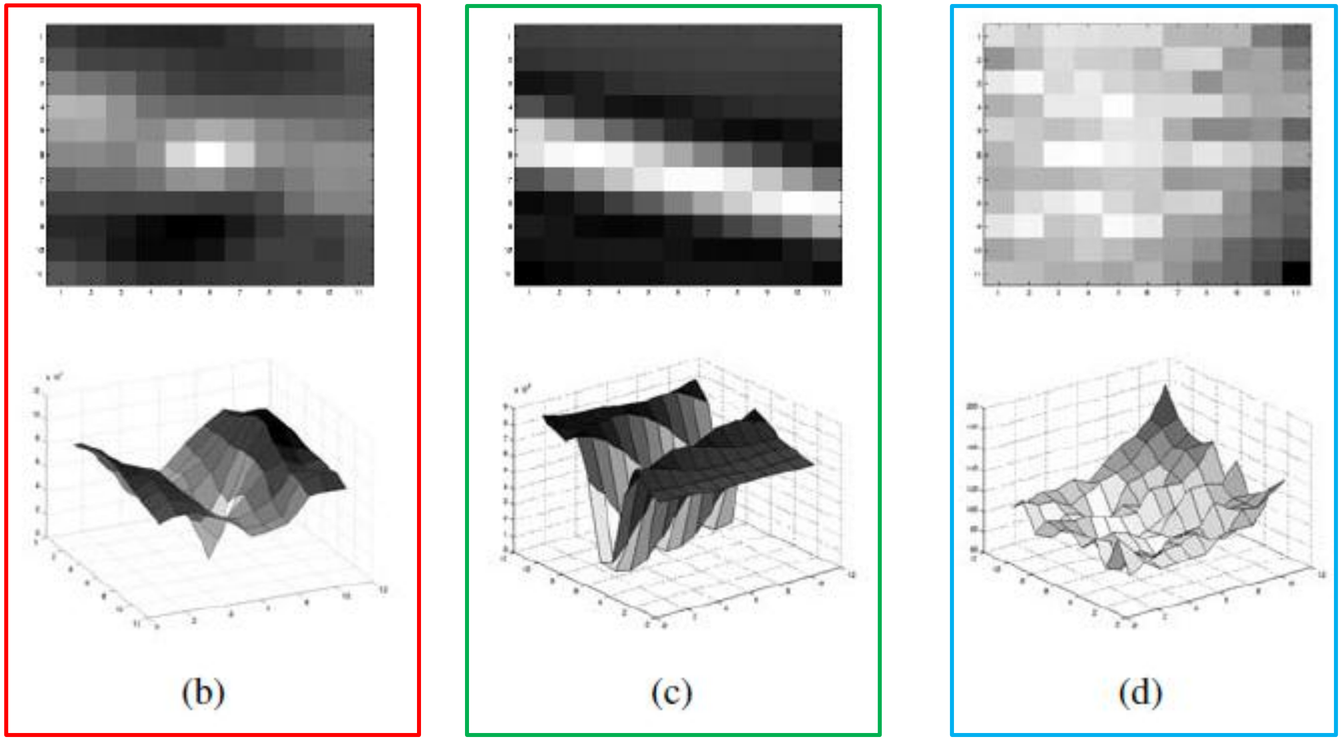
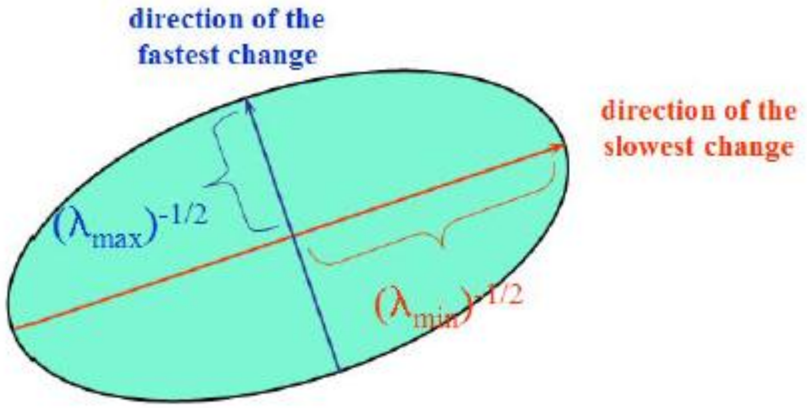


Figure 4.5 Three auto-correlation surfaces $E_{\mathrm{AC}}(\Delta u)$ shown as both grayscale images and surface plots: (a) The original image is marked with three red crosses to denote where the auto-correlation surfaces were computed; (b) this patch is from the flower bed (good unique minimum); (c) 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 Δu.

Image Autocorrelation III

- The matrix A provides a measure of uncertainty in location of the patch
- Do eigenvalue decomposition
  - Get eigenvalues and eigenvector directions
- Good features have both eigenvalues large
  - Indicates gradients in orthogonal directions (e.g. a corner)

- Uncertainty ellipse



- Many different methods to quantify uncertainty
  - Easiest: look for maxima in the smaller eigenvalue [Shi and Tomasi]
  - $\det(A) - \alpha\,\mathrm{trace}(A)^2$ [Harris]
  - See book for other methods

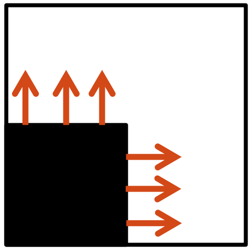
Basic Feature Detection Algorithm

1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix $\boldsymbol{A}$ is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

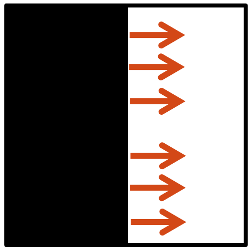
Algorithm 4.1 Outline of a basic feature detection algorithm.

Interest Point Detection

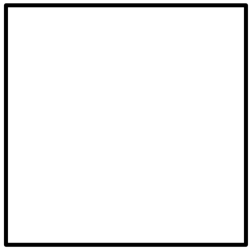
- The correlation matrix gives a measure of edges in a patch
- Corner
  - Gradient directions
    - $\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$



- Edge
  - Gradient directions
    - $\begin{bmatrix}1\\0\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$

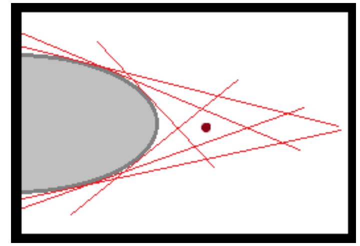


- Constant
  - Gradient directions
    - $\begin{bmatrix}0\\0\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$



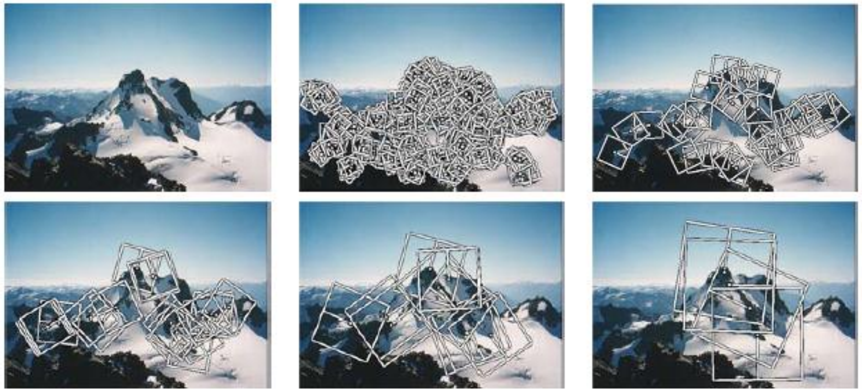
Improving Feature Detection

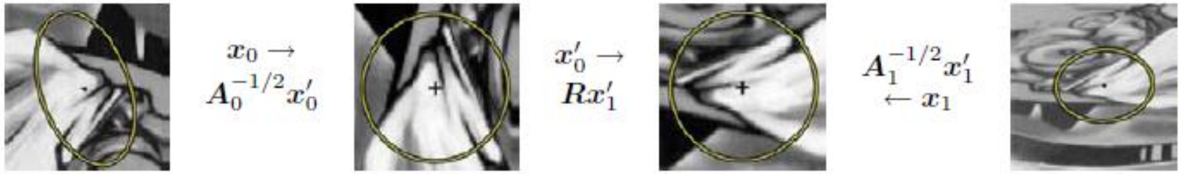
- Corners may produce more than one strong response (due to neighborhood)
  - Estimate corner with subpixel accuracy – use edge tangents
  - Non-maximal suppression – only select features that are far enough away
    - Create more uniform distribution – can be done through blocking as well
- Scale invariance
  - Use an image pyramid – useful for images of same scale
  - Compute Hessian of difference of Gaussian (DoG) image
  - Analyze scale space [SIFT – Lowe 2004]
- Rotational invariance
  - Need to estimate the orientation of the feature by examining gradient information
- Affine invariance
  - Closer to appearance change due to perspective distortion
  - Fit ellipse to autocorrelation matrix and use it as an affine coordinate frame
  - Maximally stable region (MSER) [Matas 2004] – regions that do not change much through thresholding



(a) Strongest 250

(c) ANMS 250, $r = 24$





Feature Descriptors

- Once keypoints have been detected the local appearance needs to be compactly represented
  - The representation should enable efficient matching
- Why not use the image patch itself as the descriptor?
  - The descriptor should remain the same in any image
    - Robust to photometric effects, lighting, orientation, scale, affine deformation
  - The patch intensity can be used in cases where the isn't much appearance change between images (e.g. stereo images, satellite images, video)
- The definition of descriptors to deal with the aforementioned issues is still very active

Bias and Gain Normalization (MOPS)

- Simple process to use normalized patch intensities
  - Tasks that do not have large amounts of foreshortening (perspective distortion causing differences in relative size of an objects parts)
- Patch intensities are re-scaled to be zero-mean and unit variance
- Descriptor computation:
  - Normalization of image intensity

Scale Invariant Feature Transform (SIFT)

- One of the most popular feature descriptor [Lowe 2004]
  - Many variants have been developed
- Descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes
- Descriptor computation:
  - Compute gradient 16×16 grid around keypoint
    - Keep orientation and down-weight magnitude by a Gaussian fall off function
      - Avoid sudden changes in descriptor with small position changes
      - Give less emphasis to gradients far from center
  - Form a gradient orientation histogram in each 4×4 quadrant
    - 8 bin orientations
    - Trilinear interpolation of gradient magnitude to neighboring orientation bins
    - Gives 4 pixel shift robustness and orientation invariance
  - Final descriptor is $4 \times 4 \times 8 = 128$ dimension vector
    - Normalize vector to unit length for contrast/gain invariance
    - Values clipped to 0.2 and renormalized to remove emphasis of large gradients (orientation is most important)

SIFT Schematic

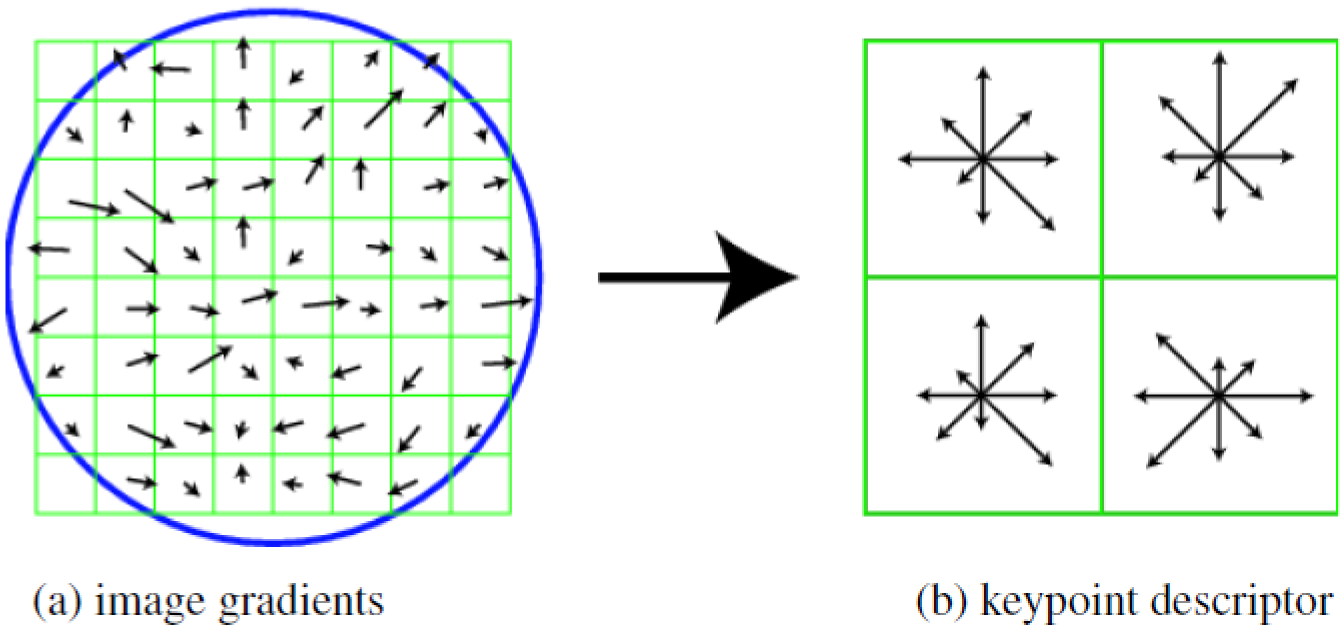
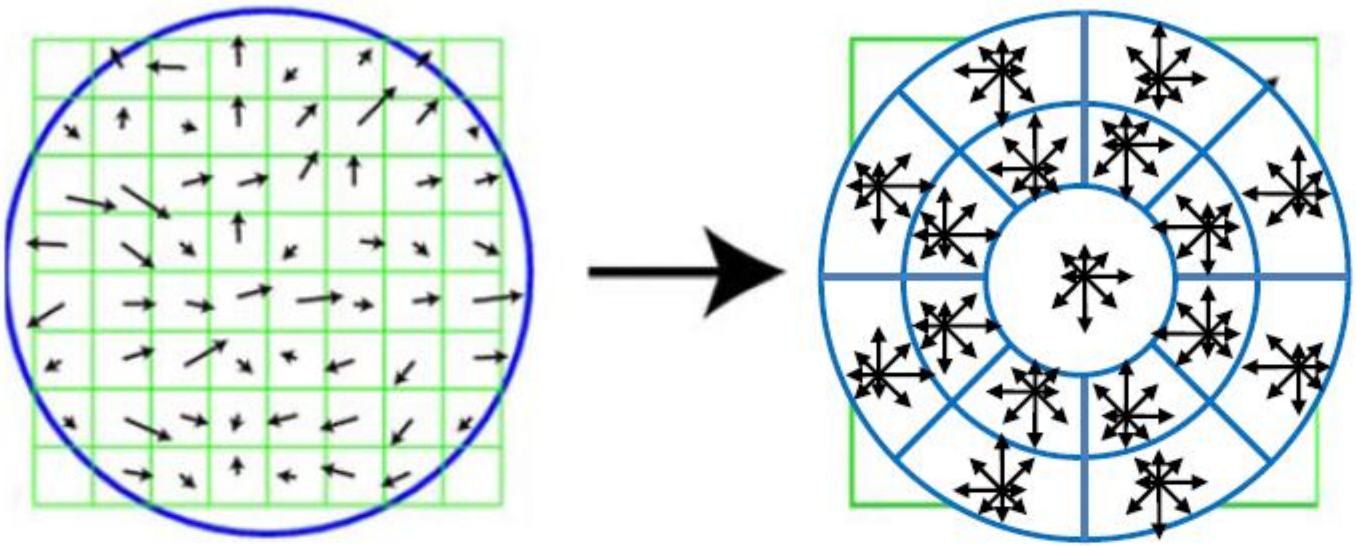


Figure 4.18 A schematic representation of Lowe's (2004) scale invariant feature transform (SIFT): (a) Gradient orientations and magnitudes are computed at each pixel and weighted by a Gaussian fall-off function (blue circle). (b) A weighted gradient orientation histogram is then computed in each subregion, using trilinear interpolation. While this figure shows an 8×8 pixel patch and a 2×2 descriptor array, Lowe's actual implementation uses 16×16 patches and a 4×4 array of eight-bin histograms.

Gradient Location-Orientation Histogram (GLOH)

- Variant on SIFT to use log-polar binning rather than 4 × 4 quadrant
  - Slightly better performance than SIFT
  - 272D histogram is projected onto 128D



(a) image gradients (b) keypoint descriptor

Figure 4.19 The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).

Other SIFT Variants

- Speeded up robust features (SURF) [Bay 2008]
  - Faster computation by using integral images (Szeliski 3.2.3 and later for object detection)
  - Popularized because it is free for non-commercial use
    - SIFT is patented
- OpenCV implements many
  - FAST
  - ORB
  - BRISK
  - FREAK
- OpenCV is maintained by Willow Garage, a robotics company
  - Emphasis on fast descriptors for real-time applications

Feature Matching

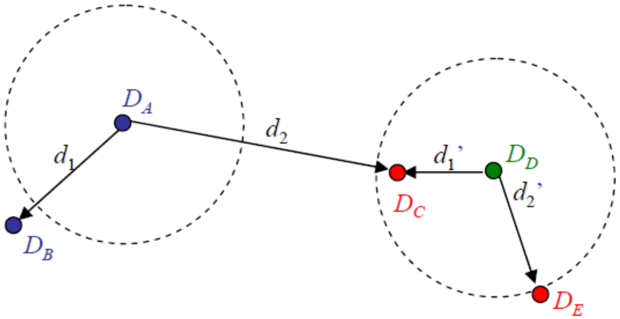
- Given descriptors from images, determine correspondences between descriptors
- Two parts to the problem
    - Matching strategy – how to select “good” correspondences
    - Efficient search – data structures and algorithms to perform matching quickly

Matching Strategy

- Generally, assume that the feature descriptor space is sufficient
  - Perform whitening of vector to concentrate on more interesting dimensions
- Use Euclidean distance as the error metric
- Set threshold to only return potential matches that are within some predefined "similarity"
  - Returns all patches from the other image that are similar enough
  - Threshold must be set appropriately to ensure matches are detected without introducing too many erroneous ones

Improved Threshold Matching

- Fixed threshold is difficult to set
  - Shouldn't expect different regions in feature space to behave the same
- Nearest neighbor matching
  - Only return the closest matching feature
  - A threshold is still required to restrict matching to "good" matches
- Nearest neighbor distance ratio
  - Adapt threshold for each feature
  - $NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$
    - Best if d_2 is a known not to match



Quantifying Performance

- Confusion matrix-based metrics
  - Binary {1,0} classification tasks

		actual value		
predicted outcome		p	n	total
	p’	TP	FP	P’
	n’	FN	TN	N’
	total	P	N	

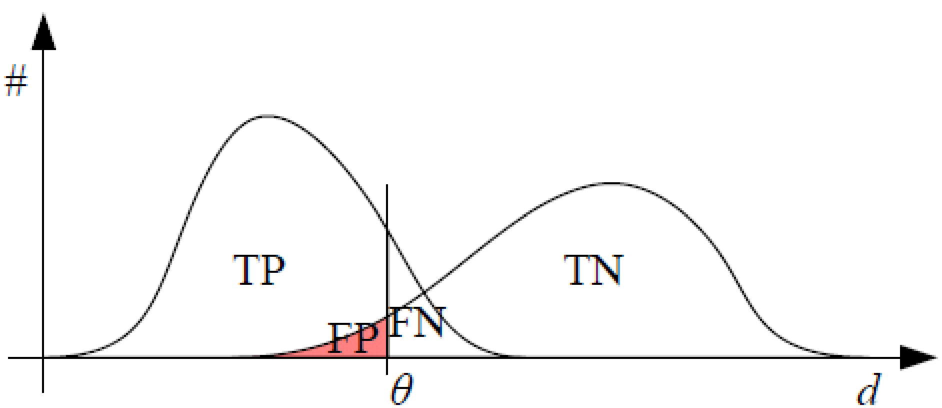
- True positives (TP) - # correct matches
- False negatives (FN) - # of missed matches
- False positives (FP) - # of incorrect matches
- True negatives (TN) - # of non-matches that are correctly rejected

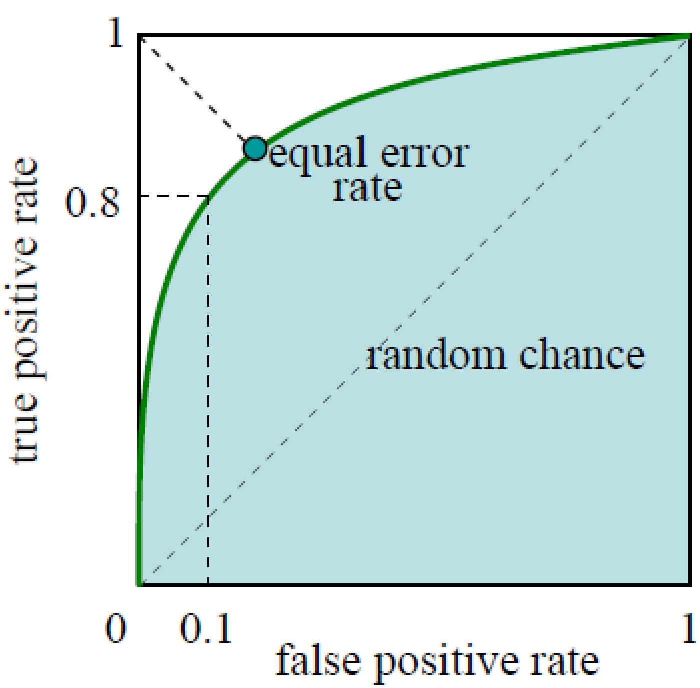
- A wide range of metrics can be defined

- True positive rate (TPR) (sensitivity)
  - $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$
  - Document retrieval → recall – fraction of relevant documents found
- False positive rate (FPR)
  - $FPR = \frac{FP}{FP+TN} = \frac{FP}{N}$
- Positive predicted value (PPV)
  - $PPV = \frac{TP}{TP+FP} = \frac{TP}{P'}$
  - Document retrieval → precision – number of relevant documents are returned
- Accuracy (ACC)
  - $ACC = \frac{TP+TN}{P+N}$

Receiver Operating Characteristic (ROC)

- Evaluate matching performance based on threshold
  - Examine all thresholds θ to map out performance curve
- Best performance in upper left corner
  - Area under the curve (AUC) is a ROC performance metric





Efficient Matching

- Straight forward matching compares all features with every other feature in every image
  - Quadratic in the number of features
- More efficient matching is possible with an indexing structure
  - Structure enables quick location of similar features
  - Can remove many potential search candidates quickly
- Popular methods are multi-dimensional trees or hash tables
  - Locality sensitive hashing, parameter-sensitive hashing
  - k-d trees

After Matching

- Matching gives a list of potential correspondences
  - Must determine how to handle these maybe matches
- Different approaches depending on task
  - Object detection – enough matching points constitutes a detection
  - Image level consistency (e.g. rotation) – determine inliers/outliers to estimate image transformation
- Random sampling (RANSAC) is very popular when there is a model to fit
  - Take a small random subset of matches, compute the model, and verify on the remaining matches

Feature Tracking

- Detect then track approach useful for video processing
- Use the same features we have already seen
- Tracking accomplished by SSD or NCC
  - Usually appearance is sufficient
- Large motions require hierarchical search strategies
  - Match in lower-resolution to provide an initial guess for speeded up search
- Must adapt the appearance model over longer time periods
  - Kanade-Lucas-Tomasi (KLT) tracker estimates affine transformation of the patch in question