Image features

Global features – global properties of an image, including intensity histogram, frequency domain descriptors, covariance matrix and high order statistics, etc.

Local features – local regions with special properties, including edges, corners, lines, curves, regions with special properties, etc.

Depending on applications, various features are useful. We will focus on edges and corners in this chapter.
Edges

Edge points are pixels at or around which the image values undergo a sharp variation – pixels with large gradient

Figure 4.1  (a) A 325 × 237-pixel image, with scanline $i = 56$ highlighted. (b) The intensity profile along the highlighted scanline. Notice how the main intensity variations indicate the borders of the hair region along the scanline.
Three type of edges

- Step edges
- Ridges edges
- Roof edges
Some edge detection results

Figure 4.6 Output of HYSTERESIS_THRESH run on Figure 4.5, showing the effect of varying the filter’s size. Left to right: $\sigma_f = 1, 2, 3$ pixel. The grey levels has been inverted (black on white) for clarity.
Edge detection

The general three-step approach

- Denoise
  - Suppress image noise without loosing the real edges
- Edge enhancement
  - Use filters that give only large responses to edges
- Edge localization
  - Distinguish large filter responses caused by real edges and noise. This is usually a two-step process: finding local maximums and reasoning about noise and true edges
Canny edge detector overview

Given an image $I$

- Apply CANNY_ENHANCER to $I$, output $I_E$
  - Denoise and gradient computation to enhance edges
- Apply NONMAX_SUPPRESSION to $I_E$, output $I_M$
  - Find local maximum in $I_E$, along edge normal. “Suppress” all the other pixels
- Apply HYSTERESIS_THRESH to $I_M$, output $I_H$
  - Remove edge pixels caused by noise
Step edge model used in Canny edge detector

- A step edge can be described by
  - edge position
  - edge normal
  - edge strength

An ideal 1D step edge is modeled as

\[
G(x) = \begin{cases} 
0 & x < 0 \\
A & x \geq 0 
\end{cases}
\]

We also assume the edge is corrupted by additive white Gaussian noise.

Based on this model and criteria for good edge detectors, optimal filter can be designed.
Canny edge enhancer - theory

- Criteria for optimal edge detection
  - High detection rate and low false alarm rate - maximize signal to noise ratio, the ratio between the filter response to the step function and the filter response to the noise. If the filter is $f(x)$, the filter length is $2W$, the root mean square (RMS) noise amplitude per unit length is $n_0$, then
  \[
  \text{SNR} = \frac{A \left\| \int_{-W}^{0} f(t) dt \right\|}{n_0 \sqrt{\int_{-W}^{W} f^2(t) dt}}
  \]

- Good localization of edges
  \[
  \text{LOC} = \frac{A \left\| f'(0) \right\|}{n_0 \sqrt{\int_{-W}^{W} f'^2(t) dt}}
  \]

- The best filter to maximize $\text{SNR} \times \text{LOC}$ is $f(x) = G(-x)$
- Another constraint is that there should be only a few local maximum responses
Canny edge enhancer - theory

- In general, a larger filter improves detection, but worsen localization, this is called the localization-detection tradeoff
  - Larger W results larger SNR (why ?)
  - Larger W results smaller LOC (why ?)

- A first derivative of Gaussian proves to be a good filter that approximates the optimal filter using the previous three criteria
  - So we should compute \( \nabla G_{1D} \ast I \) along all possible directions and find the largest response in all directions
  - A good approximation is using 2D Gaussian filter and estimating the normal and strength

\[
s(i, j) = \| \nabla G \ast I \|
\]

\[
\mathbf{n} = \frac{\nabla (G \ast I)}{\| \nabla G \ast I \|}
\]

- Notice that \( \nabla G \ast I = \nabla (G \ast I) \)
Canny edge enhancer - algorithm

Algorithm CANNY_ENHANCER

- Apply Gaussian smoother to $I$, obtaining $J = G * I$
- For pixel $(i,j)$
  - Compute gradient component $J_x, J_y$
  - Estimate the edge strength as
    $$e_s(i, j) = \sqrt{J_x^2(i, j) + J_y^2(i, j)}$$
  - Estimate the edge normal as
    $$e_o(i, j) = \arctan\left(\frac{J_x}{J_y}\right)$$
  - The output is the strength image $e_s(i, j)$ and an orientation image $e_o(i, j)$
Canny nonmaximum suppression

- The result of CANNY_ENHANCER contain wide ridges around local maximums, nonmaximum suppression will produce 1-pixel wide edges

Algorithm NONMAX_SUPPRESSION

- Quantize $e_o(i,j)$ into four directions, $0^\circ$, $45^\circ$, $90^\circ$, $135^\circ$
- For each pixel $(i,j)$
  - If $e_s(i,j)$ is smaller than at least one of its two neighbors along the normal direction, set $I_N(i,j) = 0$ (suppression), otherwise assign $I_N(i,j) = e_s(i,j)$
- The output is an image $I_N(i,j)$
Canny edge enhancer - results

- CANNY_ENHANCER + nonmaximum suppression

Figure 4.5 Strength images output by CANNY_ENHANCER run on Figure 4.1, after nonmaximum suppression, showing the effect of varying the filter’s size, that is, the standard deviation, \( \sigma_f \), of the Gaussian. Left to right: \( \sigma_f = 1, 2, 3 \) pixel.
Canny hysteresis thresholding - algorithm

- Goal: find local maximums that are true edges
- Assumption
  - true edges should have large strength in general
  - pixels belong to true edges are connected to each other

Based on the first assumption, we should threshold $I_N(i, j)$
- Problem: some edge pixels has lower values than false edge pixels
- Solution: hysteresis algorithm – a pixel belongs to a true edge if its edge strength is at least $\tau_l$ and is linked to some points with edge strength larger than $\tau_h$, where $\tau_h > \tau_l$
Canny hysteresis thresholding - algorithm

Algorithm HYSTERESIS_THRESH

- For all the edge points in $I_N$, and scanning $I_N$ in a fixed order:
  - Locate the next unvisited edge pixel $I_N(i, j)$ such that $I_N(i, j) > \tau_h$
  - Starting from $I_N(i, j)$, follow the chains of connected local maxima, in both direction perpendicular to the edge normal, as long as $I_N > \tau_l$. Mark all visited points and save a list of the locations of all points in the connected contours found

- The output is a set of lists
Canny hysteresis thresholding - results

Figure 4.6  Output of HYSTERESIS_THRESH run on Figure 4.5, showing the effect of varying the filter’s size. Left to right: $\sigma_f = 1, 2, 3$ pixel. The grey levels has been inverted (black on white) for clarity.
**Sobel edges detector - algorithm**

**Algorithm SOBEL_EDGE_DETECTION**

- Apply noise smoothing to the original image
- Filter the resultant image with the two kernels respectively

\[
\begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\quad
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\]

Obtaining \( I_1 \) and \( I_2 \)

- Estimate the gradient magnitude at each pixel as

\[
G(i, j) = \sqrt{I_1^2(i, j) + I_2^2(i, j)}
\]

- Mark a pixel as edge points if \( G(i, j) > \tau \)
Sobel edges detector - results

Figure 4.7  Left: output of Sobel edge enhancer run on Figure 4.1. Middle: edges detected by thresholding the enhanced image at 35. Right: same, thresholding at 50. Notice that some contours are thicker than one pixel (compare with Figure 4.5).
Corner features

- Sources: intersection of image lines, corner patterns in the images, etc
- Stable across sequence of images

Figure 4.8 Corners found in a 8-bit, synthetic checkerboard image, corrupted by two realizations of synthetic Gaussian noise of standard deviation 2. The corner is the bottom right point of each 15 x 15 neighbourhood (highlighted).
Harris corner detector

For a pixel $p$, use its neighborhood (e.g. 7x7) to form the following matrix

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Where $I_x$, $I_y$ are image gradient components

If the smaller eigen-value of this matrix $\lambda_2$ is larger than a certain threshold, it is considered a corner

![Figure 4.9](image_url) (a): original image of a building. (b): the 15 × 15 pixel neighbourhoods of some of the image points for which $\lambda_2 > 20$. (c): histogram of $\lambda_2$ values across the image.
Harris corner detector

For each pixel \( q \) in the neighborhood of \( p \), the gradient is \( [I_x^q, I_y^q]^T \). \( C \) is the covariance matrix of all the gradient vectors in the neighborhood. The eigenvalues represent the major and minor axis of the elliptical approximation of the gradient vector distribution.
Harris corner detector - algorithm

- Compute the image gradient
- For each pixel $p$
  - Form matrix $C$ in a $(2N+1) \times (2N+1)$ neighborhood
  - Compute $\lambda_2$, the smaller eigenvalue of $C$;
  - If $\lambda_2 > \tau$, save the coordinates of $p$ into a list $L$
- Sort $L$ in decreasing order of $\lambda_2$
- Scanning the sorted list top to bottom: for each current point $p$, delete all points appearing further on in the list which belong to the neighborhood of $p$
- The output is a list of corner points that do not overlap
- Remarks: to make the result robust, Gaussian filter $I_x^2$, $I_y^2$, and $I_xI_y$ before computing $C$
Harris corner detector - result

Figure 4.10  (a): image of an outdoor scene. The corner is the bottom right point of each $15 \times 15$ neighbourhood (highlighted). (b): corners found using a $15 \times 15$ neighbourhood.
Homework

- Prove $\nabla G*I = \nabla (G*I)$ for the 2D case

- Corners can also be detected if $R = \det C - k(\text{trace } C)^2$ is larger than a certain threshold, where $k$ is set to a small number 0.04 (Suggestion of Harris). Explain why?