Lecture 5
Template Matching and Feature Detection

Part 1

- Template matching
  - image matching
  - similarity and dissimilarity measures
  - object versus contour matching
  - invariance and speed
- Types of local image features
  - edges
  - lines
  - blobs
  - corners
- Blob detection
  - The 'Mexican Hat' operator
  - The 'Difference of Mean' operator
- Line detection

Image matching

Matching in general: Finding *correspondences* between two or more images.

Basic tasks of computer vision related to matching:

1. Given images of a scene taken by *different sensors*, bring them into registration. (Image registration.)
2. Given images of a scene taken at *different times*, find correspondences, displacements, or changes. (Motion analysis.)
3. Given images of a scene taken from *different positions*, find correspondent points to obtain 3D information about the scene. (Stereomapping, 3D object reconstruction.)
4. Find places in an image or on a contour where it matches a given pattern. (Template matching, feature detection.)
5. Match *two contours* for object recognition, measurement, or positioning. (Contour matching.)

Key issues of matching: invariance under imaging transformations (spatial and intensity) and sensitivity to noise and distortions.

Only tasks 4 and 5 will be discussed here.
Transformations considered: spatial 2D rotation and shift, linear intensity transformations. (Shift and scaling, that is, multiplication by a constant.)

Template matching: finding patterns in image

Task: Compare subimage (template) $w(r', c')$ with image $f(r', c')$ for all possible displacements (r, c). In other words, *match* $w(r', c')$ *against* $f(r + r', c + c')$ for all (r, c).

Measure of mismatch or *dissimilarity* between image f and template w in position (r, c):

$$D(r,c) = \sum_{\substack{(r',c')\in W \\ (r+r',c+c')\in F}} [f(r+r', c+c') - w(r', c')]^2$$

Here W is the coordinate domain of w, F that of f.

$D(r, c)$ is *not* invariant under intensity shift and scaling. To compensate for intensity shift (e.g., because of uneven illumination), *intensity shift-corrected dissimilarity* is used:

$$\delta(r,c) = \sum_{\substack{(r',c')\in W \\ (r+r',c+c')\in F}} \{[f(r+r', c+c') - \overline{f}(r,c)] - [w(r', c') - \overline{w}]\}^2$$

Here $\overline{w}$ is the average value of the template which is computed only once. $\overline{f}(r, c)$ the average value of the image in the region covered by the template; it is computed in each position (r, c).

A simple *measure of match* or *similarity* is the unnormalised cross-correlation of image f with template w:

$$C(r,c) = \sum_{\substack{(r',c')\in W \\ (r+r',c+c')\in F}} f(r+r', c+c') \cdot w(r', c')$$

$C(r, c)$ is *not* invariant under intensity shift and scaling. To compensate for linear intensity transformations, i.e., changes in both average level and amplitude, the *normalised* cross-correlation (correlation coefficient) is used:

$$\gamma(r,c) = \frac{\sum_{\substack{(r',c')\in W \\ (r+r',c+c')\in F}} [f(r+r', c+c') - \overline{f}(r,c)] \cdot [w(r', c') - \overline{w}]}{\sqrt{S_f \cdot S_w}}$$

where

$$S_f(r,c) = \sum_{\substack{(r',c')\in W \\ (r+r',c+c')\in F}} [f(r+r', c+c') - \overline{f}(r,c)]^2$$

and

$$S_w = \sum_{(r',c')\in W} [w(r', c') - \overline{w}]^2$$

Here again S_w is computed only once, while $S_f(r, c)$ is computed in each position (r, c).

If the average values are *not* subtracted, the normalised cross-correlation is only intensity scale-invariant (scale-corrected).

Template matching: Varying r and c, search for *high* C or γ, or *low* D or δ.

Examples of matching by unnormalised and scale-corrected normalised cross-correlation (CC):

template

0	0	0
1	1	1
0	0	0

image	*unnormalised CC*	*normalised CC*
1 1 1	1 2 3 2 1	1.00 1.41 1.73 1.41 1.00
1 1 1 1 1 1 1 1 1	1 2 3 2 1 1 2 3 2 1 1 2 3 2 1	1.00 1.22 1.00 1 1.00 1.22 1.00
1 1 1	1 1 1 1 1 1 1 1 1	
1 1 1 1 1 1 1 1	1 2 3 2 1 1 1 2 1 1 1 2 3 2 1	1.15 1.34 1.15 1.15 1.34 1.15

In the images, *pixels not shown are* 0*'s*. Values below 1 were discarded.

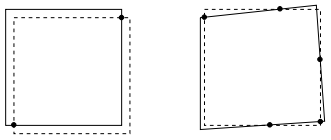
Note: The perfect match value (1.73) is not much better than the near misses in position and shape. *The match is not sharp!*

Matching of *outlines* yields *sharper* matches:

image	*template*	*normalised CC*
1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	1.22 1.33 2.00 1.33 1.22 2.00 3.00 2.00 1.22 1.33 2.00 1.33 1.22
1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0	1.34 1.41 1.34 1.41 2.82 1.41 1.34 1.41 1.34

Trade-off between localisation accuracy (contours of pattern) and reliability (interior of pattern):

- matching the contours is faster and yields sharp matches, but is very sensitive to distortions;
- matching the whole pattern is slower and less sharp, but more robust.



Contours matching versus interior matching. The template is the dashed rectangle. The circles mark the overlapping points of the contours. Left: Small shift of template results in drastic decrease of contour overlap and negligible descrease of area overlap. Contour matching is sharper. Right: Limited distortion of pattern has similar effect. Contour matching is less robust.

Problems:

- 'Noisy' matches: unexpected configurations may occur that yield high matching values.
- Sensitivity to pattern distortion, e.g., because of varying viewing angle.
- Sensitivity to changes in size and rotation.
- Heavy computational load.

Possible solutions:

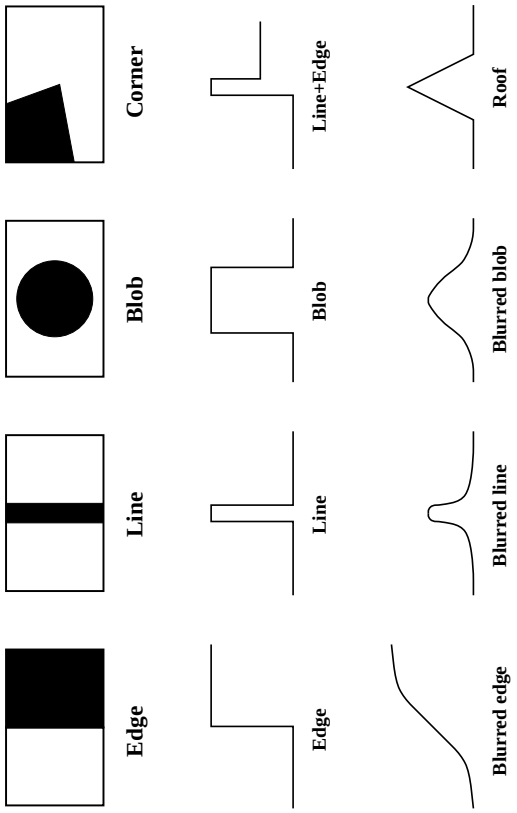
- Normalising for size:
  - normalise the image for size; works only if there is no size variation within the image;
  - spatially scale the template in each position; can cope with size variations, but slow.
- Normalising for rotation: rotate the template;
  - very slow at exhaustive rotations;
  - used only at limited rotations.
- Distortion-tolerant matching:
  - use flexible templates composed of spatially connected subtemplates with flexible links;
  - segment the image and template into regions and find correspondencies by comparing region properties; works when the segmentation is reliable.

Speeding it up:

- For large templates (more than 13×13 pixels), consider implementation in frequency domain via Fast Fourier Transform;
- Work with local features of images and templates, e.g., edges, rather that the patterns themselves; useful for sparse, reliable features.
- Use a fast procedure to select match candidates and reject mismatches rapidly, then test the selected candidates:
  - use a coarsely spaced grid of template positions, then rectify the candidates; works if peaks of cross-correlation are smooth and broad, rather than spikelike;
  - compute a simple property of template and image region, then select the candidate regions having value of the property close to that of the template; makes sense if computation of the property is faster than computation of cross-correlation;
  - use subtemplates to reject a mismatch rapidly when a subtemplate does not match;
  - if a cumulative measure of mismatch is used, reject a position when the mismatch exceeds a preset threshold.

Types of local image features

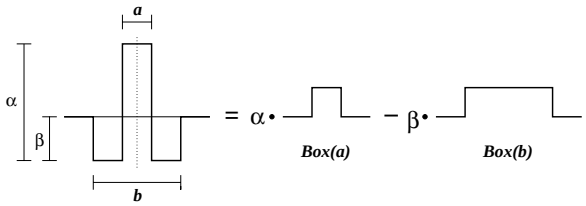
- *Edges* are drastic changes of *intensity* across object contours.
  - do not necessarily coincide with *physical* edges;
  - importance: physiological evidence exists that human eye detects edges 'in hardware', at initial level of processing.
- *Lines* are narrow, elongated objects of approximately the same intensity. Can be viewed as two close, parallel sequences of edges.
  - blurred lines and other linear objects may have different, e.g., roof-shaped, cross-section;
  - lines may be combined with edges;
  - humans may perceive texture borders or other structural changes as 'virtual lines'; example: border between two periodic sequences of parallel bars, one being shifted with respect to the other.
- *Blobs* are compact objects of approximately the same intensity.
- *Corners* are sharp turns of the contour.
  - corners are used in shape analysis and motion analysis;
  - human perception of 2D shapes relies on characteristic shape features: corners and other points of high curvature;
  - shapes can be approximately reconstructed from their characteristic points;
  - detection of corners on contours — connected and ordered sequences of points — is considered in another lecture; operators that detect corners in images are *not* considered here.



Blob detection

The 'Mexican Hat' blob detector:

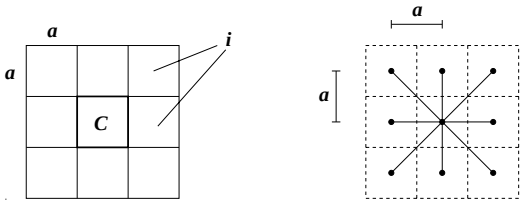
- Rotationally symmetric around the axis. In practice, approximated by rectangular filters in order to speed it up.
- a is the expected size of blobs: 'blob template'. $b > a$ averages vicinity of blob and removes background.
- Can be implemented as fast combination of two (running) box filters: speed is independent of a and b.
- Reminds the Laplacian filter.
- May respond to lines as well.



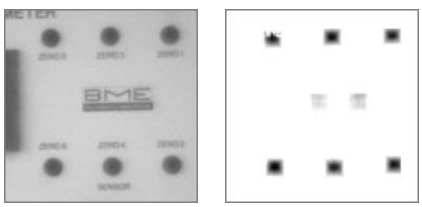
The 'Mexican Hat' blob detector and its fast implementation as a weighted sum of two mean filters.

The 'Difference of mean' (DoM) blob detector:

- A light blob is detected if $Mean(c) > Mean(i)$ for all $i = 0, 1, ..., 8$. Output blob contrast: Average of $|Mean(c) - Mean(i)|$. Similarly for dark blobs.
- Fast implementation: a size box filtering followed by spatial differencing between the corresponding points of the filtered image. Then speed is independent of a. (Run filtering.)



The 'Difference of Mean' blob detector. Left: a configuration of nine $a \times a$ size windows. Right: fast implementation with a $a \times a$ box filter followed by spatial differencing.



Example of blob detection by the DoM operator. The operator size is approximately equal to the size of the dark circular blobs.

Line detection

Line Detection Operators

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

(a) E-W (b) NE-SW (c) N-S (d) NW-SE

These masks detect lines of one pixel width. Each mask responds to a line of certain orientation. The mask with maximum response is selected, similarly to edge detection with compass masks. (See edge detection.)

Alternatives for detecting lines of larger width:

- Use larger masks that match the expected line width.
- Reduce image resolution until the above masks can be applied.
- Detect lines as parallel sequences of edges, that is, detect contours of lines.
- If the line shape is known (e.g., straight line), use a global technique rather than local line detection. Example: the Hough transform.