

# Feature Correspondence

*Arthur Goshtasby*

Wright State University

Image Registration and Fusion Systems

# Correspondence methods

## Point pattern matching:

- 1) scene coherence
- 2) clustering
- 3) invariance

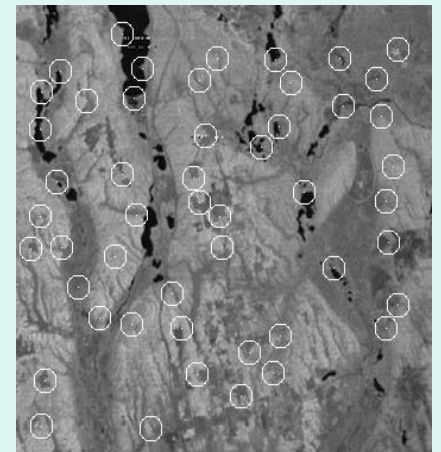
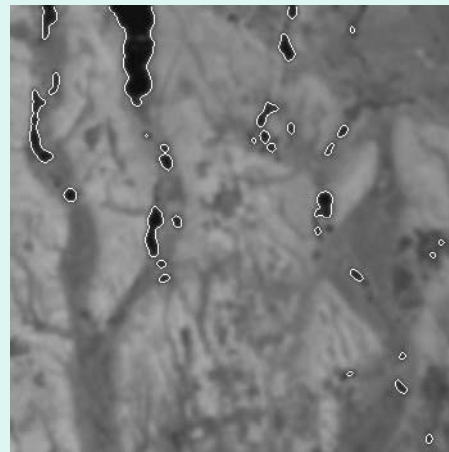
## Line matching

## Region matching:

- 1) shape matching
- 2) relaxation labeling
- 3) chamfer matching

## Template matching:

- 1) Similarity measures
- 2) Coarse-to-fine methods

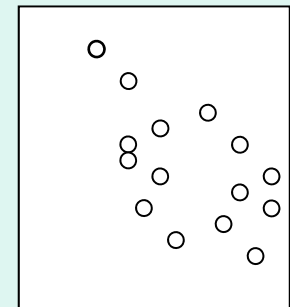
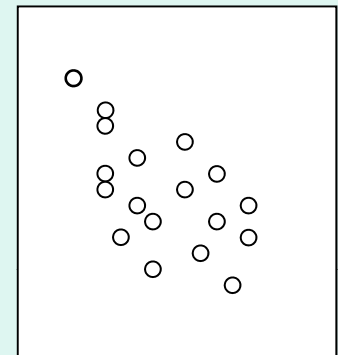


# Point pattern matching

**Problem:** Given two sets of points, determine the correspondence between them.

- Information about only the locations of the points is available.
- The points may contain positional error.
- Some points may exist in only one of the sets.

We will only consider the case where the two point sets are related by the affine transformation.



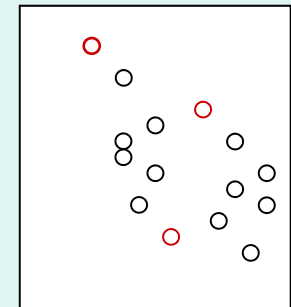
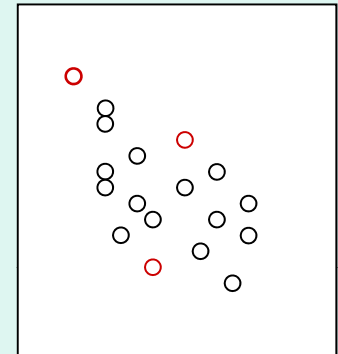
# Point pattern matching using scene coherence

If three points in the two sets are aligned, because of the scene coherence, the remaining points in the two sets will also align.

**RANSAC Algorithm: Find three corresponding points.**

- 1. Take 3 points from set 1 and 3 points from set 2 and find the affine transformation that aligns them.**
- 2. Count the number of other points in the two sets that also align with the obtained transformation.**
- 3. If the count is sufficiently high, stop the process. Otherwise, go to step 1.**

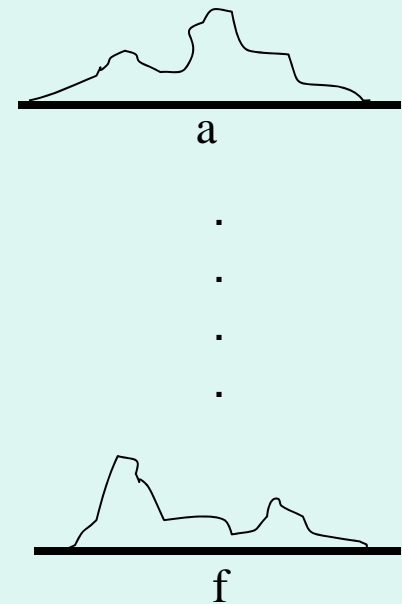
To speed up the search, limit the point combinations to those falling on the convex-hulls or the minimum-spanning trees of the two sets.



# Point pattern matching using clustering

## Algorithm:

1. Create accumulators  $a[ ] - f[ ]$  and initialize the entries to 0.
2. From point triples in the two sets calculate parameters  $a - f$  of
$$X = ax + by + c,$$
$$Y = dx + ey + f.$$
3. Increment entries  $a - f$  of accumulators  $a[ ] - f[ ]$ , respectively, by 1.
4. Repeat Steps 2 and 3 a sufficiently large number of times.
5. Locate the entry with the highest counts in  $a[ ] \dots f[ ]$ . They show parameters  $a - f$  of the registration.

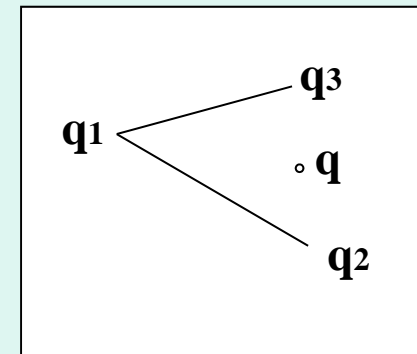
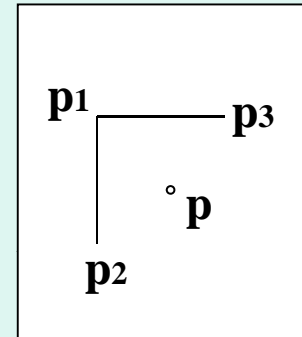


# Point pattern matching using affine invariance

If two point sets are related by an affine transformation, knowing three corresponding points in the two sets  $(p_1, q_1)$ ,  $(p_2, q_2)$ ,  $(p_3, q_3)$ , the relation between corresponding points  $(p, q)$  in the sets can be written as

$$p = p_1 + a_1(p_2 - p_1) + a_2(p_3 - p_1)$$

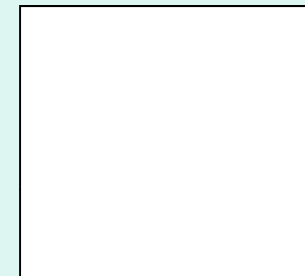
$$q = q_1 + a_1(q_2 - q_1) + a_2(q_3 - q_1)$$



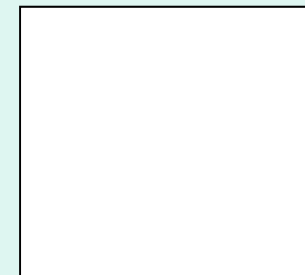
# Point pattern matching using affine invariance

## Algorithm:

1. Create two 2-D accumulator arrays  $\mathbf{H}_1[ ]$  and  $\mathbf{H}_2[ ]$ .
2. Select three points in set 1 and for each additional point in set 1 calculate  $a_1$  and  $a_2$  and increment entry  $[a_1, a_2]$  of  $\mathbf{H}_1[ ]$  by 1.
3. Select three points in set 2 and for each additional point in set 2 calculate  $a_1$  and  $a_2$  and increment entry  $[a_1, a_2]$  of  $\mathbf{H}_2[ ]$  by 1.
4. Find the similarity between  $\mathbf{H}_1$  and  $\mathbf{H}_2$ . If the similarity is sufficiently high, take the point triples selected in Steps 2 and 3 as corresponding points and stop. Otherwise, go to Step 2.



$\mathbf{H}_1$



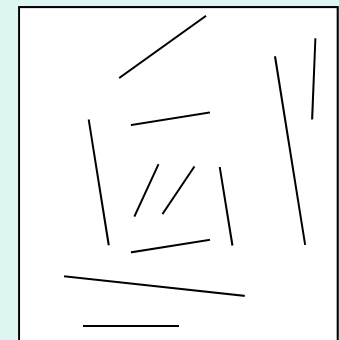
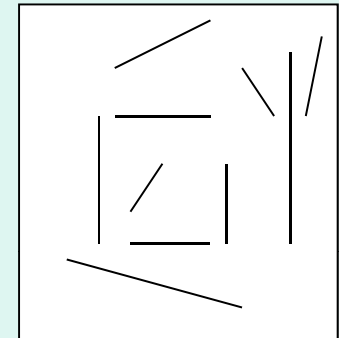
$\mathbf{H}_2$

# Line matching

**Assumption:** Images are related by a rigid transformation (unknown translation and rotation).

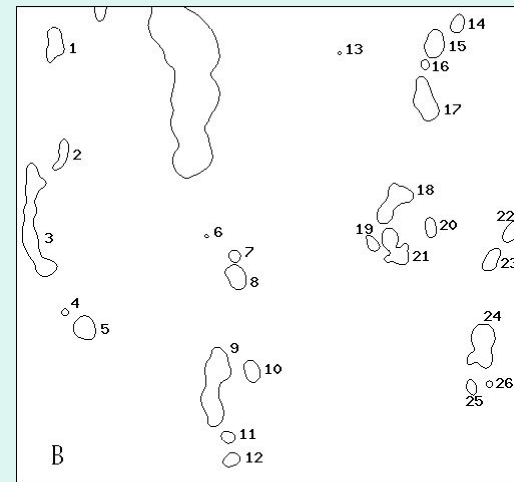
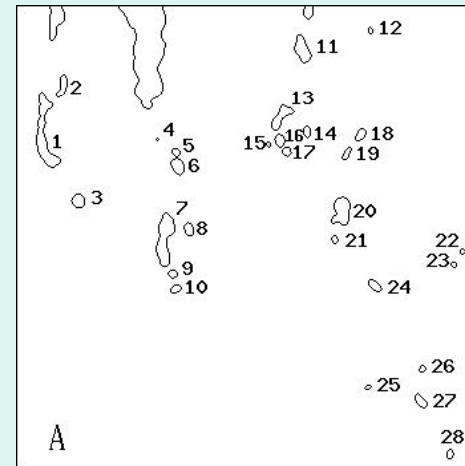
**Algorithm:**

1. Find the rotational difference between the line sets.
2. Correct the orientation of set 2 with respect to set 1.
3. Find the translational difference between the line sets.



# Region matching

- Shape matching
  - Fourier descriptors
  - Invariant moments
  - Shape matrices
  - Relaxation labeling
  - Chamfer matching
    - Distance transform



# Fourier descriptors

- Given pixels on the boundary of a region  $\{(x_i, y_i): i=0, \dots, N-1\}$ , the  $xy$  coordinates of the pixels along the boundary can be considered samples from a periodic signal. Letting  $z_i = x_i + jy_i$ , where  $j = \sqrt{-1}$ , Fourier descriptor is computed from

$$c_k = \frac{1}{N} \sum_{i=0}^{N-1} z_i \exp(-j2\pi ki / N), \quad k=0, \dots, N-1$$

# Invariant moments

- Given boundary pixels  $\{(x_i, y_i): i=0, \dots, N-1\}$ , the  $(p+q)$ th order moment is defined by

$$m_{pq} = \sum_{i=0}^{N-1} x_i^p y_i^q f_i$$

The  $(p+q)$ th order central moment of the boundary is defined by

$$u_{pq} = \sum_{i=0}^{N-1} (x_i - \bar{x})^p (y_i - \bar{y})^q f_i$$

- Invariant moments:

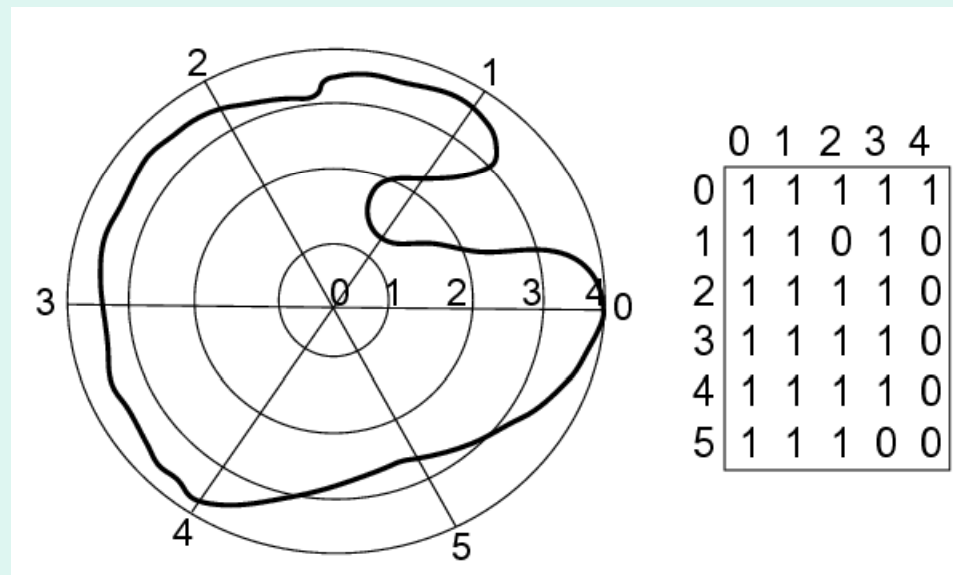
$$a_1 = u_{20} + u_{02}$$

....

$$a_7 = (3u_{21} - u_{03}) \dots$$

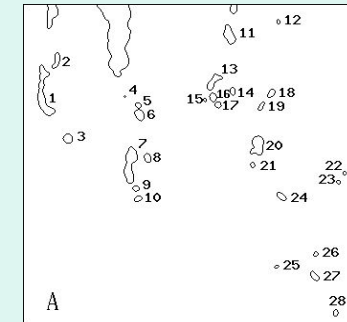
# Shape matrix

- Resampling a shape into a representation that is independent of its position, orientation, and scale.

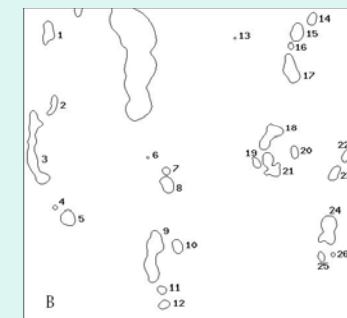


# Relaxation labeling

- Denote regions in the reference image by  $\{a_i: i=1, \dots, m\}$  and regions in the sensed image by  $\{b_i: i=1, \dots, n\}$ .
- If  $a_i$ 's denote objects and  $b_i$ 's denote labels, the correspondence problem becomes that of finding labels for the objects such that a compatibility condition is satisfied.
- Initially assign labels to the objects with probabilities proportional to their similarities.  
 $P_i(b_j)$ : similarity between regions  $a_i$  and  $b_j$ .
- Iteratively revise the label probabilities until they converge to either 0 or 1.



Set of objects



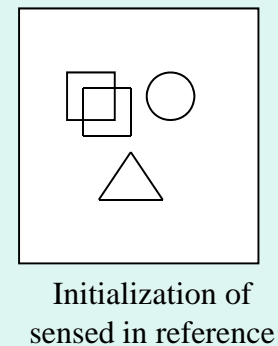
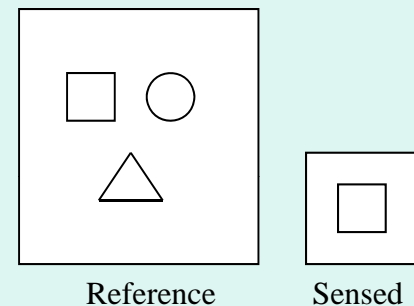
Set of labels

$$P_i^{k+1}(b_j) = \frac{P_i^k(b_j) [1 + q_i^k(b_j)]}{\sum_{j=0}^n P_i^k(b_j) [1 + q_i^k(b_j)]}$$

# Chamfer matching

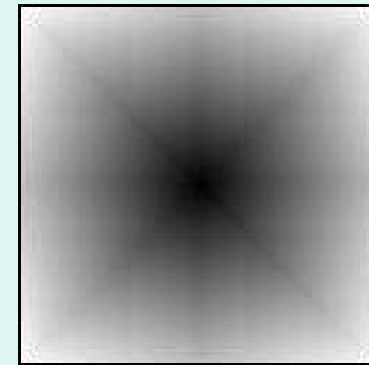
Given two binary images with *translational differences*:

1. Initially position the sensed image within the reference image at  $(i,j)$  based on some available information.
2. Determine the similarity between the two images using distances of closest object points.
3. Reposition the sensed image within the reference image at the eight neighbors of  $(i,j)$  and determine the image similarities at these eight positions.
4. If highest similarity is obtained when sensed image is at  $(i,j)$ , stop. Otherwise, move the sensed image to the neighbor of  $(i,j)$  that produces the highest similarity and go to Step 2.

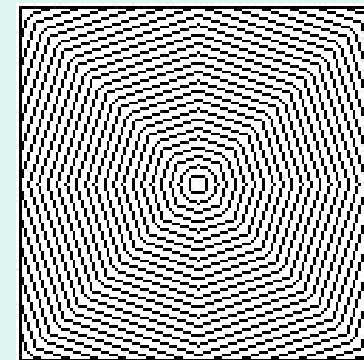


# Distance transform

- Given a binary image, assign to a background pixel a value proportional to its distance to the object point closest to it.
- To speed up the computations, integer distances may be used, but note that integer distances involve errors and make distances dependent on the orientation of the image.



Distance transform of a  
single point

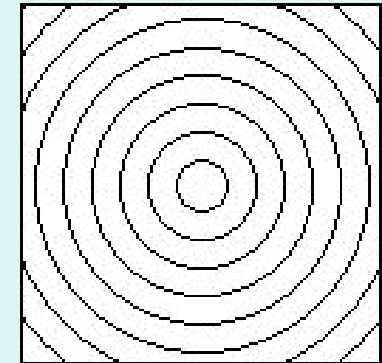
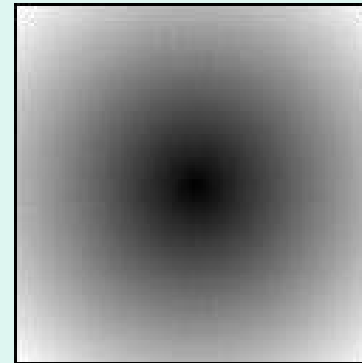


Isovalued distances

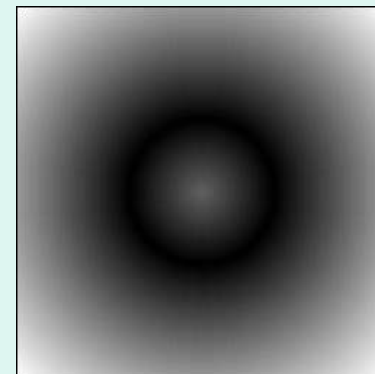
# Distance transform

To make distances rotationally invariant, use actual Euclidean distances.

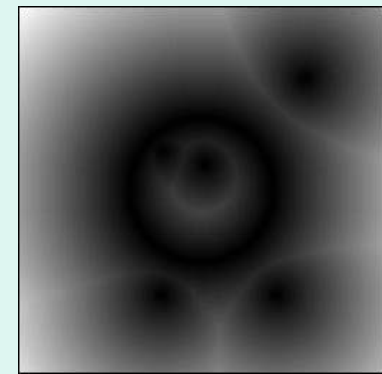
Distance transform in its current form is very sensitive to noise.



Euclidean distance transform of a single point and the isovalued distances.

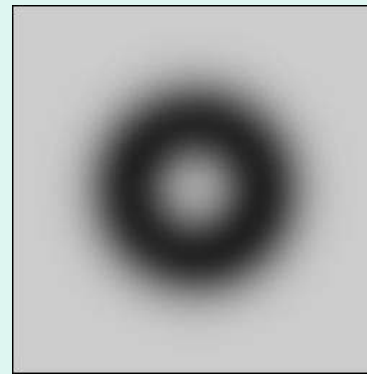


Euclidean distance transform of a circle

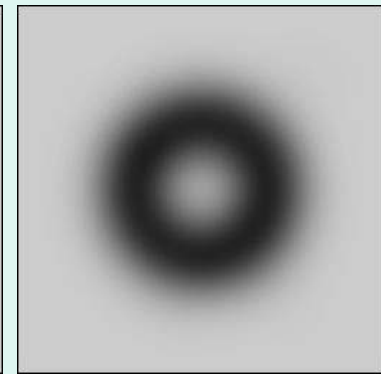


Euclidean distance transform of a circle and 5 noisy points

- Instead of saving a single distance at a pixel, save a weighted sum of distances.

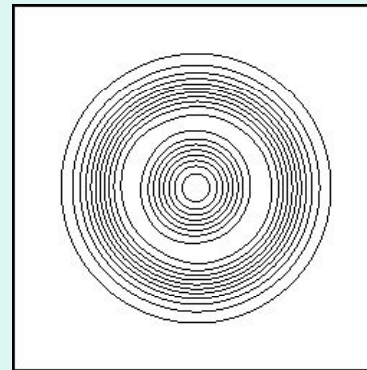


Circle

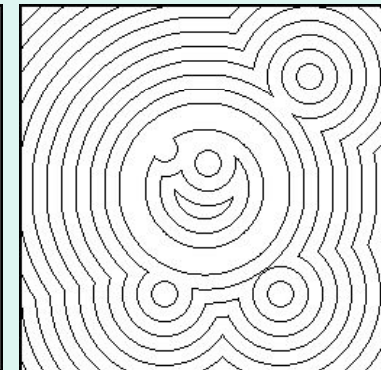


Circle + 5 points

- This can be implemented by smoothing the binary image with a Gaussian and inverting the values.



New DT of circle + 5 points



Traditional DT of circle + 5 points

# Template matching

- This is same as chamfer matching except that the images are no longer binary.
- Similarity measures
  - Sum of absolute differences
  - Cross-correlation coefficient
  - Mutual information
- Gaussian weighted templates
- Coarse-to-fine approaches

# Similarity measures

- Given two gray scale images with translational differences, shift one image over the other and at each shift position determine the similarity between the two.
- Sum of absolute differences

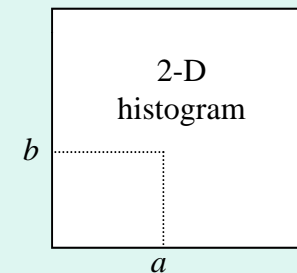
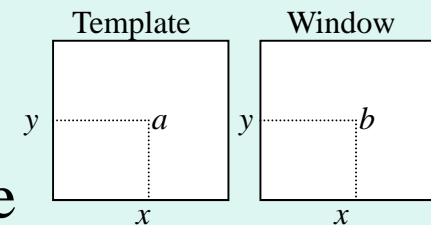
$$D(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |f_t[i, j] - f_w[i + x, j + y]|$$

- Cross-correlation coefficient  $(g = f - f_{mean})$

$$S(x, y) = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} g_t[i, j]g_w[i + x, j + y]}{\left\{ \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} g_t^2[i, j] \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} g_w^2[i + x, j + y] \right\}^{1/2}}$$

# Mutual information

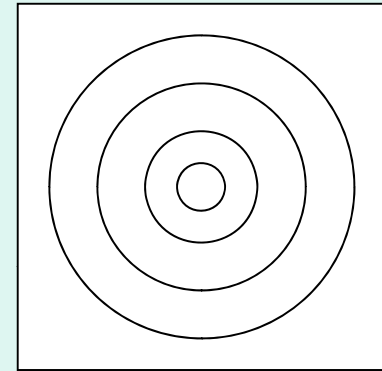
- Create a 2-D histogram with entry  $[a, b]$  showing the number of pixels in the template with intensity  $a$  that align with intensity  $b$  in the window.
- Divide the counts by the number of pixels in template to obtain joint probabilities.
- Then, compute:



$$I(t, w) = \sum_{a=0}^{255} \sum_{b=0}^{255} P_{tw}(a, b) \log \frac{P_{tw}(a, b)}{P_t(a)P_w(b)}$$

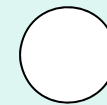
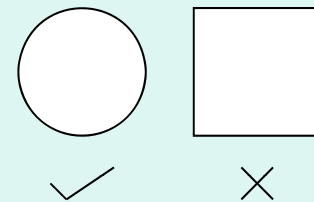
# Gaussian weighted templates

- Instead of treating all pixels in a template similarly, give higher weights to pixels that are closer to the template center.
- This makes the process less dependent on image orientation when using rectangular templates.
- It also reduces the effect of geometric difference between images.

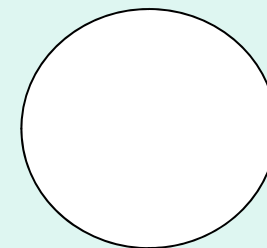


# On template shape and size

- Always take circular templates. This will make the process less dependent on image orientation.
- Set template size proportional to the information content in the template. A smaller template size is sufficient when the template is highly detailed compared to when it covers a rather homogeneous area.



For a detailed area



For a less detailed area

# Coarse-to-fine matching

- **At image level:** Reduce the size of images, find the correspondences, and determine the transformation parameters. Use the transformation to resample the images at one level higher resolution. Repeat the process until images at the highest resolution are registered.
- **At template level:** Either use smaller templates or cheaper similarity measures to find candidate match positions. Then find the best match position from among the candidates using a larger template and/or a more expensive similarity measure.

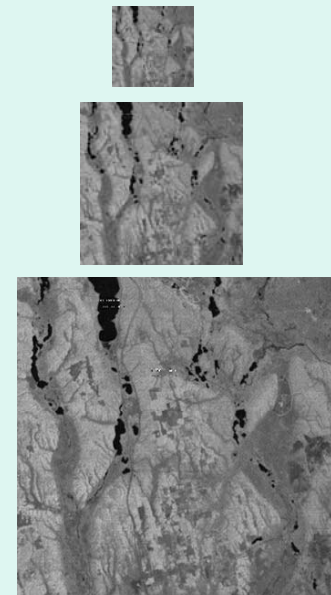
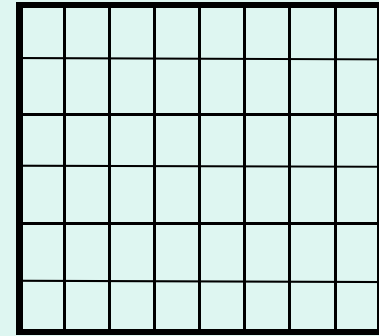


Image pyramid

- **During search:** Use large steps to find possible match positions. Then use finer steps in the neighborhood of the likely matches to refine the final match position.
- **Use partial information:** If a large set of landmarks is given, first use a subset of the landmarks to find approximate registration parameters and then verify the correctness of the registration and refine the parameters using all landmarks.



# Feature matching references

1. A. Goshtasby and G. C. Stockman, Point pattern matching using convex-hull edges, *IEEE Transactions on Systems, Man, and Cybernetics*, **15**(5):631–637 (1985).
2. W. J. Rucklidge, Efficiently locating objects using the Hausdorff distance, *Int'l J. Computer Vision*, **24**(3):251–270 (1997).
3. G. Stockman, S. Kopstein, and S. Benett, Matching images to models for registration and object detection via clustering, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **4**(3):229–241 (1982).
4. J. L. Mundy and A. Zisserman, *Geometric Invariance in Computer Vision*, The MIT Press, Cambridge, MA (1992).
5. J. Kittler and J. Illingworth, Relaxation labeling algorithms - A review, *Image and Vision Computing*, **3** (4):206–216 (1985).
6. L. S. Shapiro and J. M. Brady, Feature-based correspondence: An eigenvector approach, *Image and Vision Computing*, **10**(5): 283–288, 1992.
7. A. Goshtasby, S. H. Gage, and J. F. Bartholic, A two-stage cross correlation approach to template matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(3):374–378 (1984).
8. D. Holtkamp and A. Goshtasby, Precision registration and mosaicking of multicamera images, *IEEE Trans. Geoscience and Remote Sensing*, **47**(10):3446–3455, 2009.