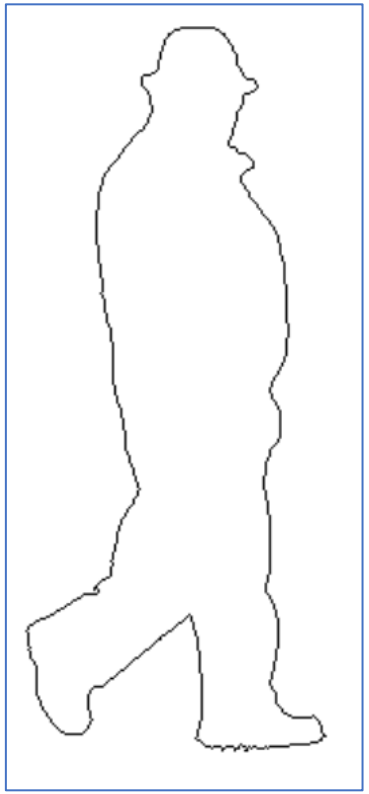
# PRS L4

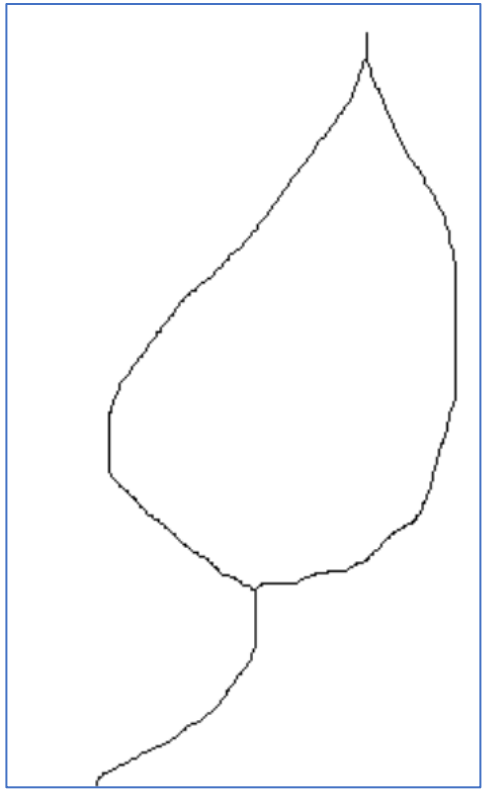
Distance Transform (DT). Pattern Matching using DT

# Distance Transform (DT)

Objective: DT is used to measure the similarity between two binary images



img1

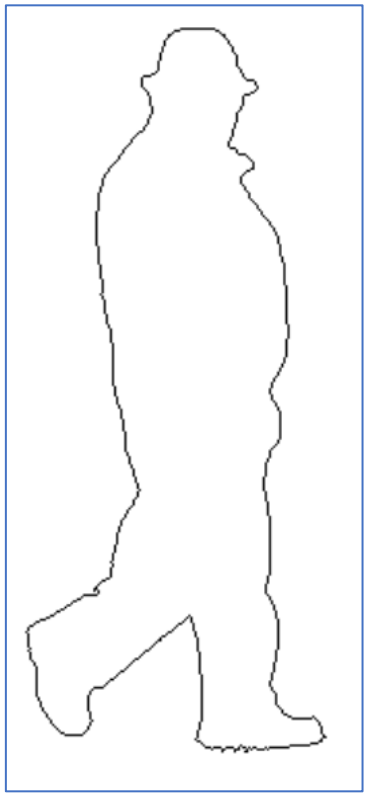


img2

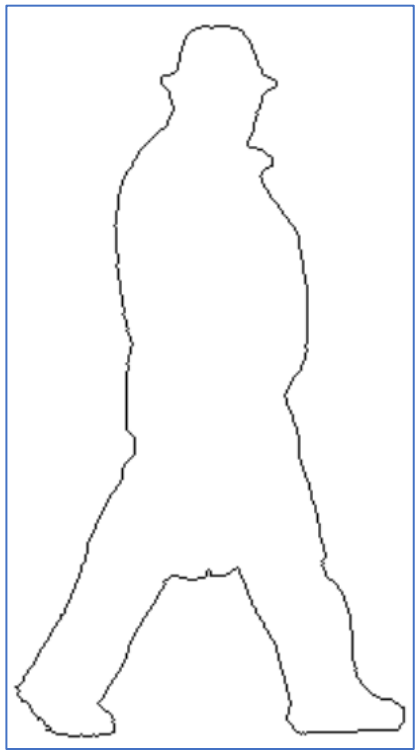
=> Large similarity cost -> they are not similar

# Distance Transform (DT)

Objective: DT is used to measure the similarity between two binary images



img1

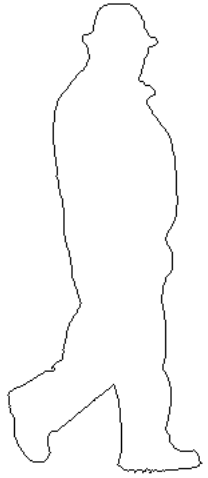


img2

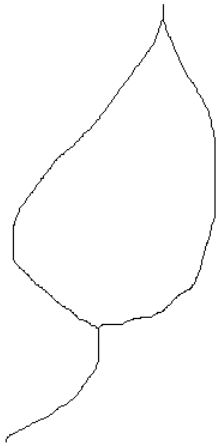
=> Small similarity cost -> they are similar

# Distance Transform (DT)

- Our goal is to evaluate the pattern matching score between a known template object (e.g. a pedestrian contour) and an unknown object (e.g. leaf)



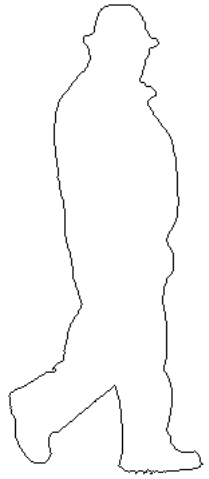
Template.bmp



Unknown_object2.bmp

# Distance Transform (DT)

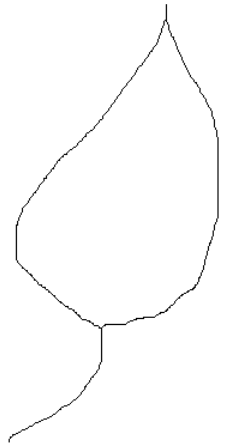
- Our goal is to evaluate the pattern matching score between a known template object (e.g. a pedestrian contour) and an unknown object (e.g. leaf)



Template.bmp

=> DT

score

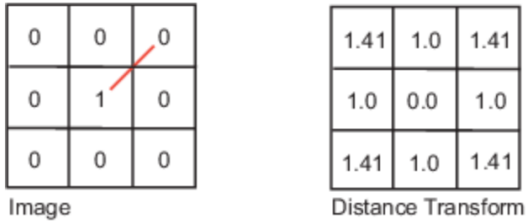
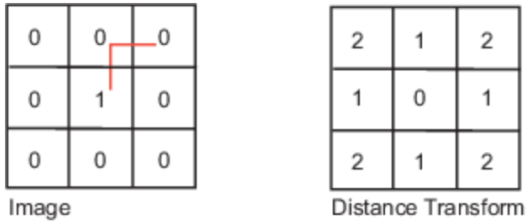
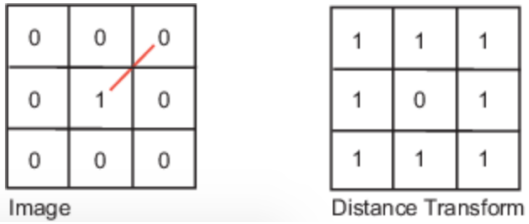


Unknown_object2.bmp

DT will contain at each pixel the distance to the nearest object pixel. (the contour of the object in our case)

# Distance Transform (DT)

- DT will contain at each pixel **the distance to the nearest object pixel**
- **What kind of distance?**

| Distance Metric | Description | Illustration |
|---|---|---|
| Euclidean | The Euclidean distance is the straight-line distance between two pixels. |   |
| City Block | The city block distance metric measures the path between the pixels based on a 4-connected neighborhood. Pixels whose edges touch are 1 unit apart; pixels diagonally touching are 2 units apart. |   |
| Chessboard | The chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood. Pixels whose edges or corners touch are 1 unit apart. |   |

# Distance Transform (DT)

- **How to compute this distance (e.g Euclidean)?**
  - Brute force: for each pixel p, its distance to each of the black pixels is computed. The distance map at p is equal to the smallest of these distances.
  - The brute force method is very inefficient

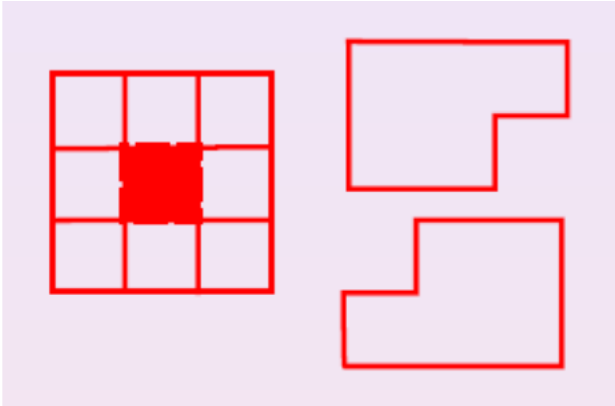
  - More efficient solution: **Chamfer based Distance Transform** which approximates the Euclidean distance

# Distance Transform (DT)

Compute the **Chamfer based DT**

1. Initialize dt - clone of the image (same size as the image, 0 – object pixel, 255 – background pixel), change type to int32

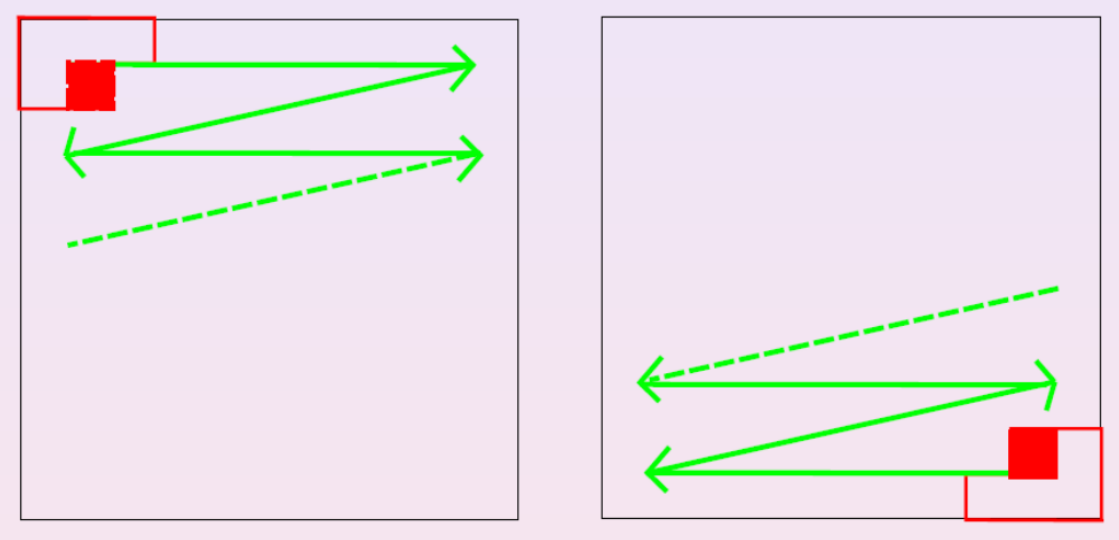
2. Select a mask of size 3x3, which is split into two parts (each submask has 5 values)



# Distance Transform (DT)

3. Scan the DT (top-down, left-right ) with the first submask and update DT
4. Scan the DT (bottom-up, right-left) with the second submask and update DT

$$DT(i,j) = \min_{(k,l)\in Mask}(DT(i+k, j+l) + weight(k,l))$$



Weight(k, l) (3 diagonal and 2 horizonal/vertical)

| | | |
|---|---|---|
| **3** | **2** | **3** |
| **2** | **0** | **2** |
| **3** | **2** | **3** |

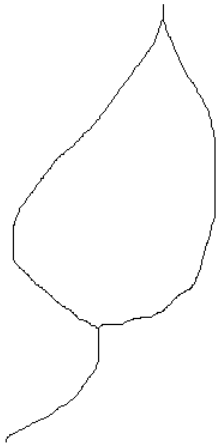
Other weights can be used:

- Chessboard (1 for all the neighbors)
- City block (2 diagonal and 1 horizontal/vertical)

# Pattern Matching

- Step 1: compute the distance transform for the template object
- Step 2: after computing the distance transform, we will compute the matching score

Template.bmp



Unknown_object2.bmp

The pattern matching score is the average of all the pixel values from the DT image that lie under the unknown object contour

# Pattern Matching

- How to compute the pattern matching score:
  1. Compute the DT for the template object
  2. **[Optional] Compute the center of mass for the template object and for the unknown object**
     - **The center of mass (mx,my) can be computed as the average coordinates of all object pixels (black pixels)**

  **3. [Optional] Translate the unknown object so that its center of mass is the same as the template object's center of mass**

  4. The pattern matching score is the average of all the pixel values from the DT image of the template object that lie under the unknown object contour

# Activitate practică

**1.** Implementați Transformata Distanță Chamfer. Calculați și vizualizați imaginea TD pentru imaginile de intrare: *contour1.bmp, contour2.bmp, contour3.bmp*. Rezultatele trebuie să coincidă cu cele prezentate în text. Pixelii de obiect sunt negri iar fundalul este alb.

**2.** Calculați imaginea TD pentru *template.bmp*. Evaluați costul de similitudine între imaginea model și cele două obiecte necunoscute: *unknown_object1.bmp* – pieton, *unknown_object2.bmp* – frunză. Costul de similitudine este media valorilor din imaginea TD de pe pozițiile punctelor de contur ale obiectului necunoscut.

**3.** Translatați obiectul necunoscut astfel încât centrul său să corespundă cu centrul obiectului model și recalculați costurile de potrivire.