EE5806: Topics in Digital Image Processing

Lecture Notes: Landmark registration

Motivation

- Wish to compare *source* image with *target* image (Fig. 1).
- Need to align (i.e., register) source with target before comparison.

Approach

- We will deal with cases where the source image is a rotated, translated and scaled version of the target. Therefore, we need to find a geometric transformation consisting of rotation, translation and scaling to align the source with the target.
- In landmark registration, we identify landmarks in the source image and corresponding landmarks in the target image.
  - o Landmarks: Distinctive features that can be accurately and reproducibly identified in both the source and target images.
- Landmarks are used to find an "optimal" geometric transformation to register the two images. In the following, we will define "optimality" mathematically.

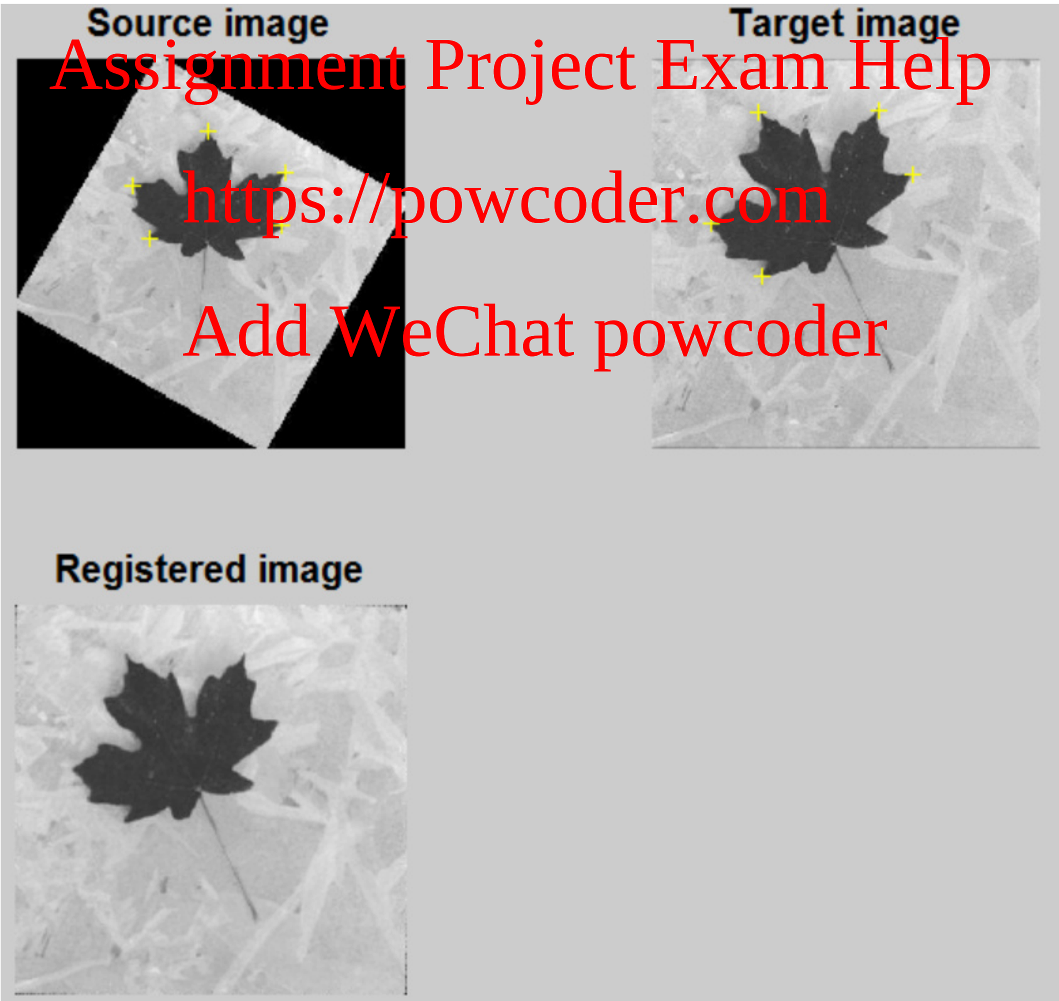

Fig. 1

Mathematical Derivation

Given: N landmarks p_n (n = 1, 2, …., N) in source image and corresponding landmarks in the target image p'_n (n = 1, 2, …., N). In 2D images:

$$p_n = \begin{bmatrix} i_n \\ j_n \end{bmatrix} \quad \text{and} \quad p'_n = \begin{bmatrix} i'_n \\ j'_n \end{bmatrix}$$

Problem: Find a geometric transformation (specifically, one consisting of rotation, translation and scaling) that aligns the two images based on the two sets of landmarks. We will assume the scaling in i and j directions are the same (i.e., $s_i = s_j = s$)

Solution:

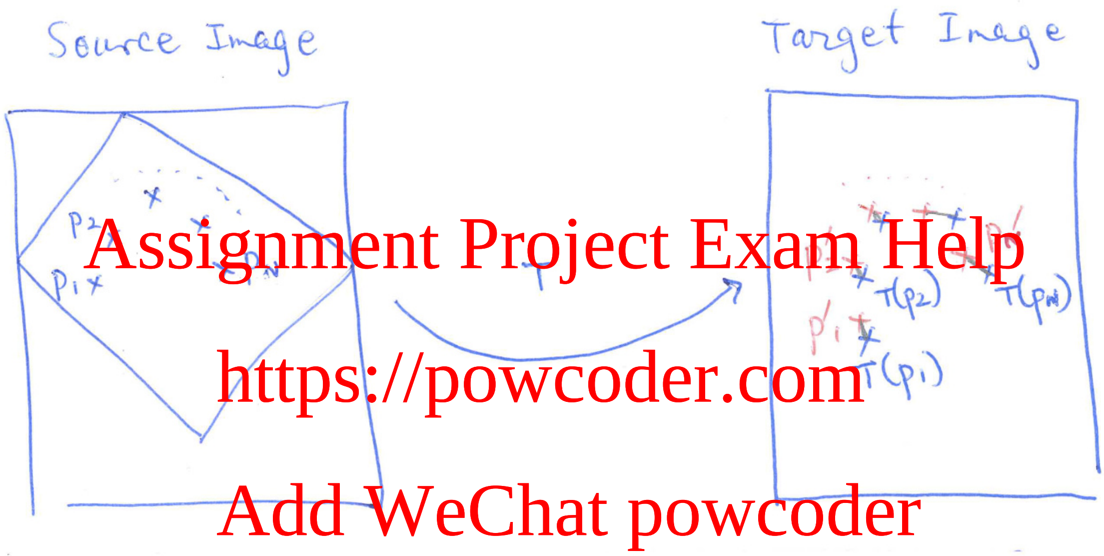

Fig. 2. Source image before and after transformation and the target image. $\{p_n\}_{n=1}^{N}$ denotes the landmarks chosen in the source image (blue crosses in left panel) and $\{p'_n\}_{i=1}^{N}$ denotes the corresponding landmarks chosen in the target image (red crosses in the right panel). After the transformation T has been applied, $\{p_n\}_{n=1}^{N}$ becomes $\{T(p_n)\}_{n=1}^{N}$ (blue crosses in the right panel).

So, there are infinitely many transformation, how do we choose a "best" transformation to perform? Before answering this question, you would ask: "After applying a certain transformation T, how close are the transformed source landmarks $\{T(p_i)\}_{i=1}^{N}$ to the target landmarks $\{p'_n\}_{i=1}^{N}$." The "closeness" is quantified by the following cost function:

Define a cost function, C, that is the mean-squared distance between the target landmarks, p'_n, and transformed source landmarks, $T(p_n)$. Our goal is to minimize this cost function:

$$C = \frac{1}{N} \sum_{n=1}^{N} ||p'_n - T(p_n)||^2$$

where $||\mathbf{v}||$ denotes the magnitude of the vector v, and

$$T(p_n) = s\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} i_n \\ j_n \end{bmatrix} + \begin{bmatrix} t_i \\ t_j \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}\begin{bmatrix} i_n \\ j_n \end{bmatrix} + \begin{bmatrix} t_i \\ t_j \end{bmatrix}$$

where $a = s\cos\theta$ and $b = s\sin\theta$.

Now, we express C in terms of $i_n, j_n, i'_n \text{ and } j'_n$:

$$C = \frac{1}{N}\sum_{n=1}^{N}\left\|\begin{bmatrix} i'_n \\ j'_n \end{bmatrix} - \begin{bmatrix} ai_n + bj_n + t_i \\ -bi_n + aj_n + t_j \end{bmatrix}\right\|^2$$

$$C = \frac{1}{N}\sum_{n=1}^{N}(i'_n - ai_n - bj_n - t_i)^2 + (j'_n + bi_n - aj_n - t_j)^2$$

To minimize this cost function, compute its derivative with respect to a, b, t_i and t_j and set to zero. Example for a:

$$\frac{\partial C}{\partial a} = 0$$

$$\frac{1}{N}\sum_{n=1}^{N} 2(i'_n - ai_n - bj_n - t_i)(-i_n) + 2(j'_n + bi_n - aj_n - t_j)(-j_n) = 0$$

$$a\sum_{n=1}^{N}(i_n^2 + j_n^2) + t_i\sum_{n=1}^{N} i_n + t_j\sum_{n=1}^{N} j_n = \sum_{n=1}^{N}(i'_n i_n + j'_n j_n)$$

Performing differentiation with respect to b, t_i and t_j as well gives the following set of linear equations:

$$\begin{array}{ccccccccc} a(S_{ii}+S_{jj}) & & & + & t_iS_i & + & t_jS_j & = & S_{ii'} + S_{jj'} \\ & & b(S_{ii}+S_{jj}) & + & t_iS_j & - & t_jS_i & = & S_{i'j} - S_{ij'} \\ aS_j & - & bS_i & & & + & t_jN & = & S_{j'} \\ aS_i & + & bS_j & + & t_iN & & & = & S_{i'} \end{array}$$

where

$$\begin{array}{llll} S_i = \sum_{n=1}^{N} i_n & S_j = \sum_{n=1}^{N} j_n & S_{i'} = \sum_{n=1}^{N} i'_n & S_{j'} = \sum_{n=1}^{N} j'_n \\ S_{ii} = \sum_{n=1}^{N} i_n^2 & S_{jj} = \sum_{n=1}^{N} j_n^2 & S_{ii'} = \sum_{n=1}^{N} i_n i'_n & S_{jj'} = \sum_{n=1}^{N} j_n j'_n \\ S_{ij'} = \sum_{n=1}^{N} i_n j'_n & S_{i'j} = \sum_{n=1}^{N} i'_n j_n & & \end{array}$$

This system of linear equations can be expressed in matrix notation $Bv = k$ where

$$B = \begin{bmatrix} S_{ii}+S_{jj} & 0 & S_i & S_j \\ 0 & S_{ii}+S_{jj} & S_j & -S_i \\ S_j & -S_i & 0 & N \\ S_i & S_j & N & 0 \end{bmatrix}, \quad v = \begin{bmatrix} a \\ b \\ t_i \\ t_j \end{bmatrix}, \quad k = \begin{bmatrix} S_{ii'} + S_{jj'} \\ S_{i'j} - S_{ij'} \\ S_{j'} \\ S_{i'} \end{bmatrix}$$

Solving for v will define our optimal geometric transformation T.

Python implementation

```
import numpy as np
import cv2
import matplotlib.pyplot as plt


def landmark_register(fnameS='imS.jpg', fnameT='imT.jpg'):
    """
    landmark_register: Landmark registration of two images. Takes into
account translation, rotation and scaling only.
    :param fnameS: a string containing the filename of the SOURCE image
    :param fnameT: a string containing the filename of the TARGET image
    Note: If you use Pycharm and cannot get the landmarks successfully,
please uncheck 'Show Plots in Tool window'
    Windows: Settings \ Tools \ Python Scientific \ Show Plots in Tool
window
    MacOS: Preferences \ Tools \ Python Scientific \ Show Plots in Tool
window
    """
    # Read in images and display side-by-side; get landmarks
    source = cv2.imread(fnameS, cv2.IMREAD_GRAYSCALE)
    target = cv2.imread(fnameT, cv2.IMREAD_GRAYSCALE)
    height, width = target.shape
    plt.ion()
    plt.subplot(131)
    plt.imshow(source, cmap='gray', vmin = 0, vmax = 255)
    plt.axis('off')
    plt.title('Source image', {'fontsize': 15, 'fontweight': 'bold'})
    print('Select landmarks in source image. Press ENTER key when done.')
    ps = plt.ginput(n=-1, timeout=0, show_clicks=True)
    ps = np.asarray(ps)
    plt.plot(ps[:, 0], ps[:, 1], 'r+')
    plt.subplot(132)
    plt.imshow(target, cmap='gray', vmin = 0, vmax = 255)
    plt.axis('off')
    plt.title('Target image', {'fontsize': 15, 'fontweight': 'bold'})
    print('Select landmarks in target image. Press ENTER key when done.')
    pt = plt.ginput(n=-1, timeout=0, show_clicks=True)
    pt = np.asarray(pt)
    plt.plot(pt[:, 0], pt[:, 1], 'r+')
    plt.ioff()
    # Set up the matrix B and the vector k
    N = len(ps)
    Sj = np.sum(ps[:, 0])
    Si = np.sum(ps[:, 1])
    Sjj = np.sum(ps[:, 0] ** 2)
    Sii = np.sum(ps[:, 1] ** 2)
    Siip = np.sum(ps[:, 1] * pt[:, 1])
    Sjjp = np.sum(ps[:, 0] * pt[:, 0])
    Sjip = np.sum(ps[:, 0] * pt[:, 1])
    Sjpi = np.sum(pt[:, 0] * ps[:, 1])
    Sjp = np.sum(pt[:, 0])
    Sip = np.sum(pt[:, 1])
    B = np.array([[Sii + Sjj, 0, Si, Sj], [0, Sii + Sjj, Sj, -Si], [Sj, -
Si, 0, N], [Si, Sj, N, 0]])
    k = np.array([Siip + Sjjp, Sjip - Sjpi, Sjp, Sip]).T
    # Solve least-squares problem: Bv = k using the inv(B)*k
    v = np.dot(np.linalg.inv(B), k)
    a = v[0]
    b = v[1]
    ti = v[2]
```

```
tj = v[3]
# Perform the transformation using built-in function.
# Step 1: Define affine matrix, T
T = np.array([[a, -b, tj], [b, a, ti]])

# Step 2: Perform the transformation.
im2 = cv2.warpAffine(source, T, (width, height))
plt.subplot(133)
plt.imshow(im2, cmap='gray', vmin = 0, vmax = 255)
plt.axis('off')
plt.title('Registered image', {'fontsize': 15, 'fontweight': 'bold'})
plt.show()
```