Exercise #2: Geometric Transform of Image

Align multiple images. Then, Generate a “stitched” image from the aligned multiple images.

1. Take a series of interesting overlapping photos.
2. Detect feature points from the images and match the feature points.
   You can use a library function, such as AKAZE, SIFT, ORB, etc.
3. Assign a unique index i to each track.
   Then you can have $\{x_i, y_i\}$ for N feature points for the series of input images.
4. Estimate a transform for each image **using the linear solution**.
   Try every transform: translation, similarity, and affine as the motion model.

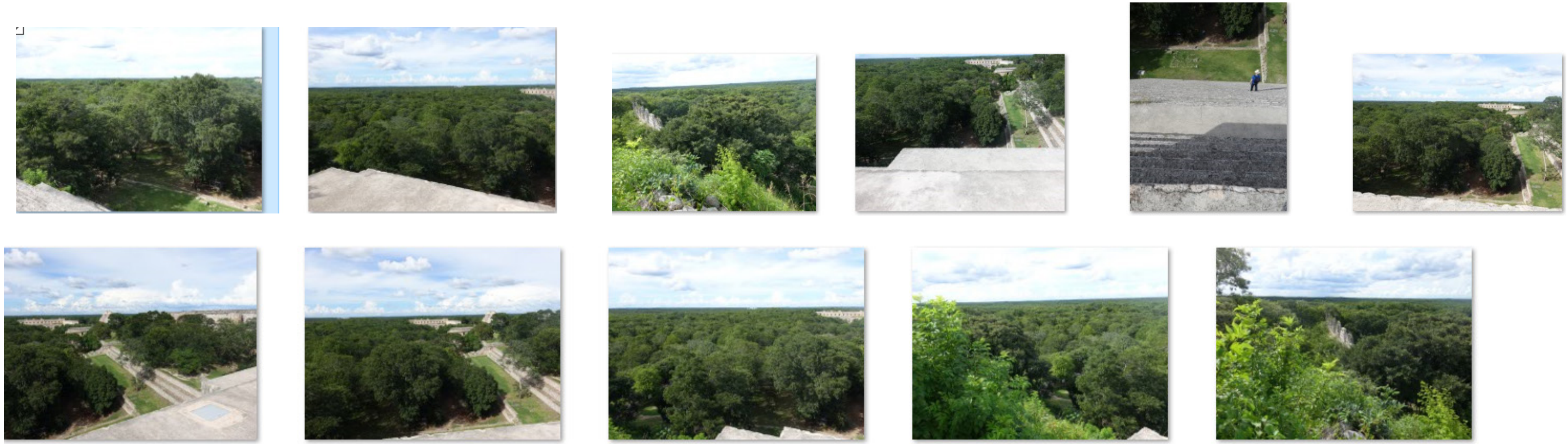
4’. For perspective transform (homography), use OpenCV function

cv2.getPerspectiveTransform

5. Compute the size of the resulting composite canvas.
6. Warp each image into its final position on the canvas.

   cv2.warpPerspective
7. Average all of the images. Think about what kind of averaging way is the best

1. Take a series of interesting overlapping photos



2. Detect feature points from the images and match the feature points.

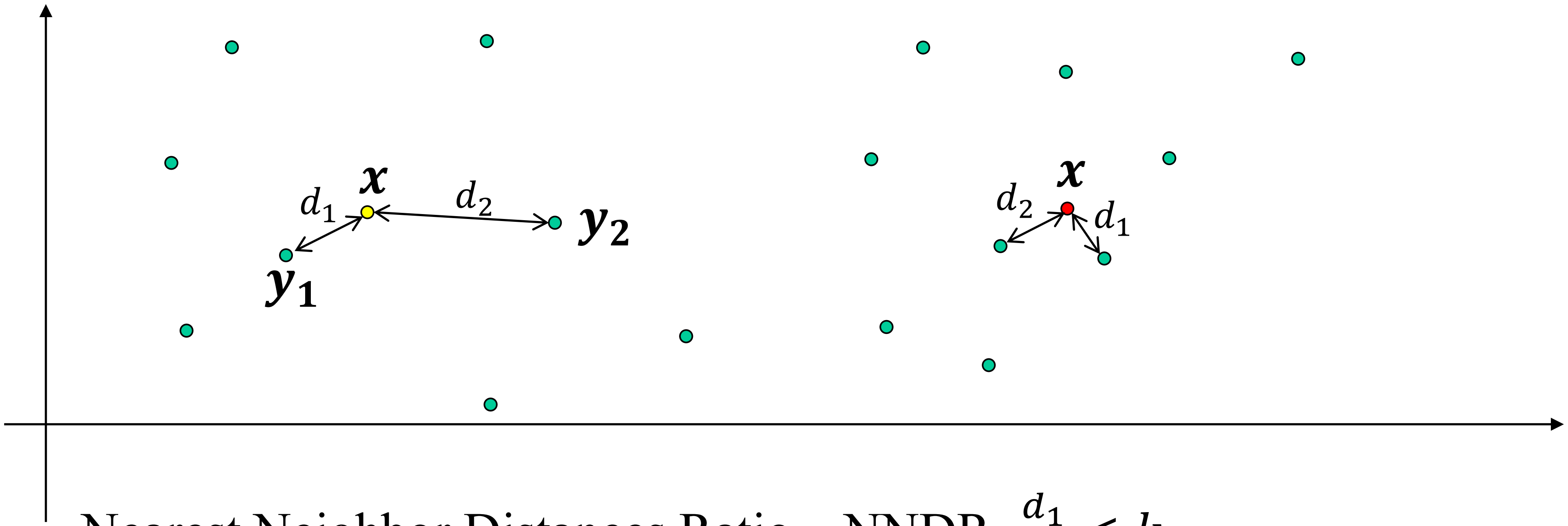
You can use a library function, such as AKAZE, SIFT, ORB, etc.

AKAZE https://docs.opencv.org/3.4/db/d70/tutorial_akaze_matching.html
ORB https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html
SUPERPOINT https://github.com/rpautrat/SuperPoint
https://github.com/magicleap/SuperPointPretrainedNetwork

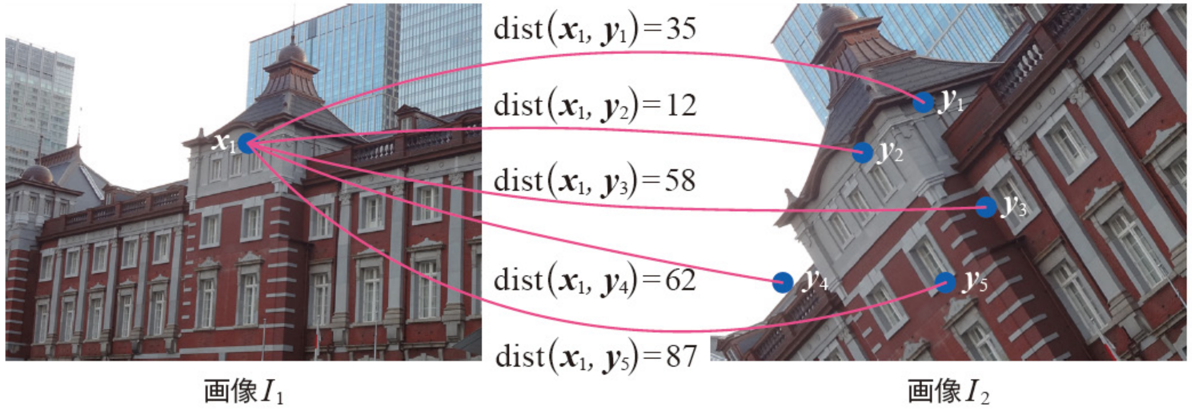
Matching strategy and error rates

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=0}^{N-1}(x_i - y_i)^2}$$

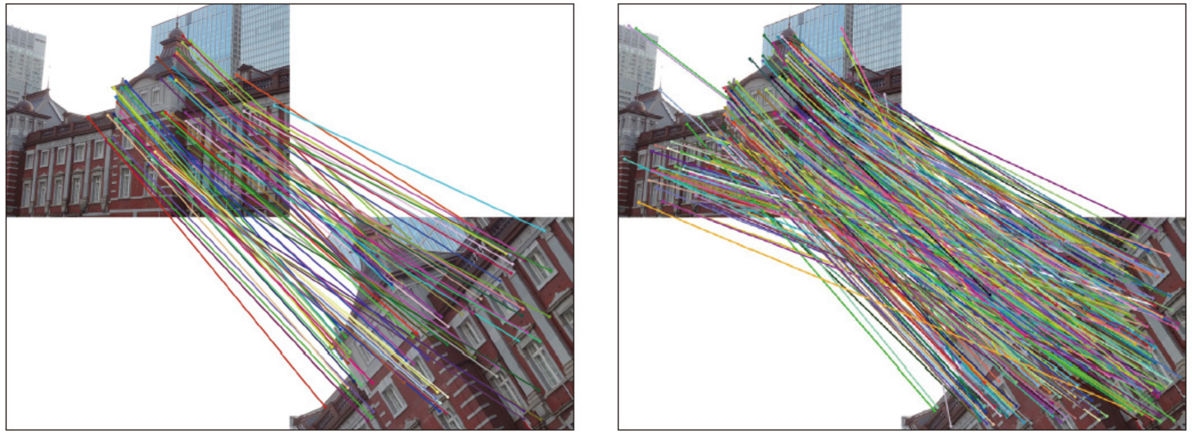
the feature descriptor space (N dim)



Nearest Neighbor Distances Ratio NNDR$=\dfrac{d_1}{d_2} < k$



■図 11.21――対応点の探索

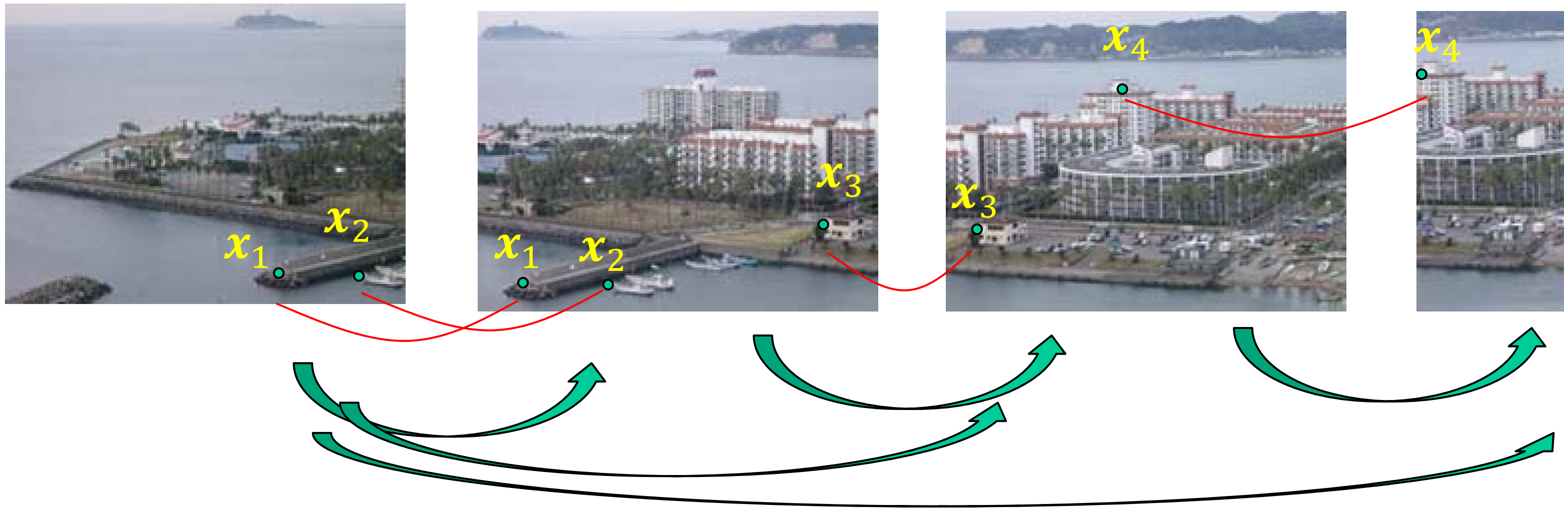


[a] k=0.25 の場合の対応づけ結果　[b] k=0.8 の場合の対応づけ結果

■図 11.22――2画像間の対応点とマッチング

3. Assign a unique index i to each track.

Then you can have $\{x_i, y_i\}$ for N feature points for the series of input images.

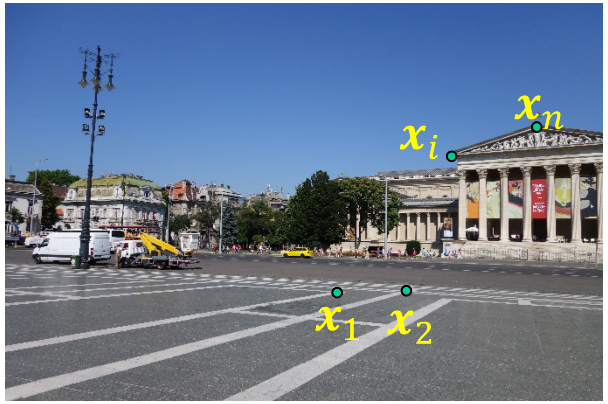


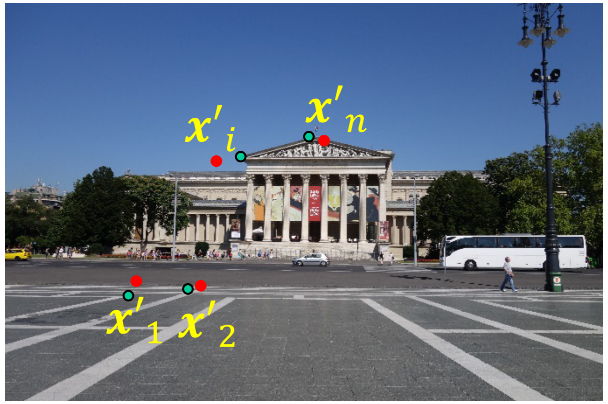
4. Estimate a transform for each image using the linear solution.

Transform	Matrix M	
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	without openCV
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	
Homography projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	with openCV

2D alignment using least squares (LS)

Given a set of matched feature points $\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}$





$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

Estimate $\boldsymbol{p}$ by minimizing the sum of squared residuals: E_{LS}

$$E_{LS} = \sum_i \|\boldsymbol{r}_i\|^2 = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}_i'\|^2$$

Predicted location by transformation model $\boldsymbol{p}$

measurement

Matched Feature Points $\{(\boldsymbol{x}_i, \boldsymbol{x'}_i)\}_1^N$

If we can assume a linear relationship
between the amount of motion $\Delta\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{x}$ and the unknown parameters $\boldsymbol{p}$

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x};\boldsymbol{p}) = \left(\frac{\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{p})}{\partial \boldsymbol{p}}\right)\boldsymbol{p} + \boldsymbol{x} = \underbrace{\boldsymbol{J}(\boldsymbol{x})}_{\text{Jacobian}}\boldsymbol{p} + \boldsymbol{x}$$

$$\boldsymbol{x}' - \boldsymbol{x} = \Delta\boldsymbol{x} = \boldsymbol{J}(\boldsymbol{x})\boldsymbol{p}$$

$$\sum_i \|\boldsymbol{J}(\boldsymbol{x}_i)\boldsymbol{p} - \Delta\boldsymbol{x}_i\|^2 \to 0$$

$$E_{LLS} = \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i)\boldsymbol{p} - \Delta\boldsymbol{x}_i\|^2$$

$$= \boldsymbol{p}^T\left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i)\right]\boldsymbol{p} - 2\boldsymbol{p}^T\left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i\right] + \sum_i \|\Delta\boldsymbol{x}_i\|^2$$

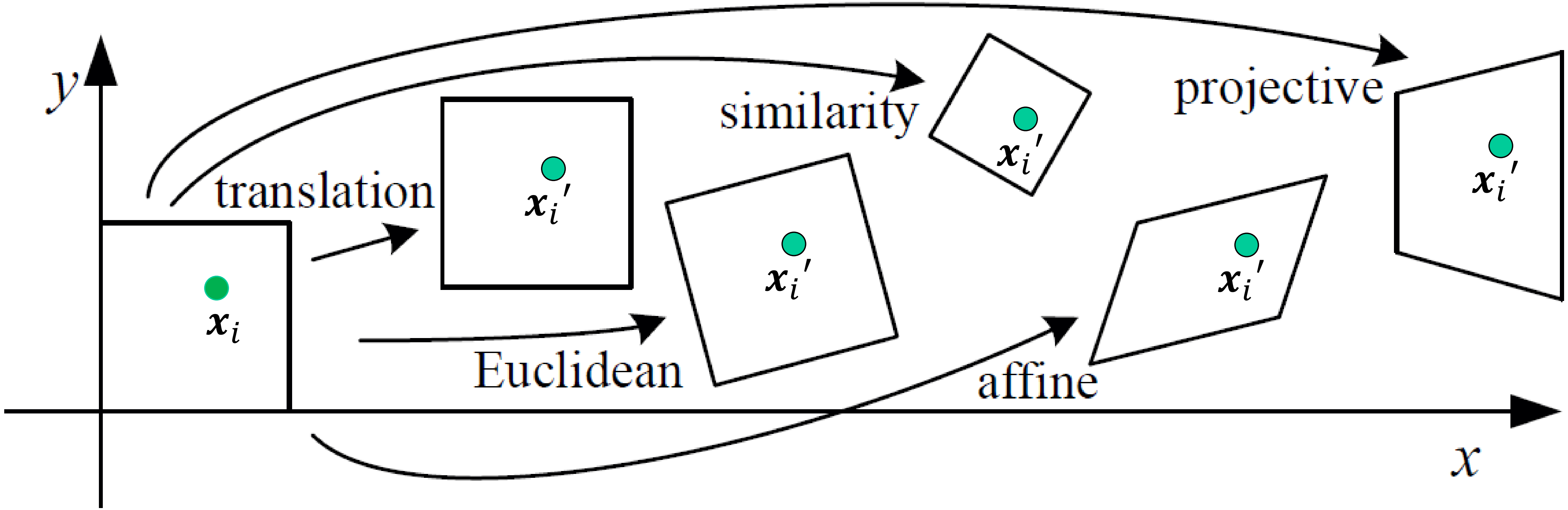
$$= \boldsymbol{p}^T\boldsymbol{A}\boldsymbol{p} - 2\boldsymbol{p}^T\boldsymbol{b} + c$$

LLS: linear least squares

Minimum E_{LLS} can be found by solving symmetric positive definite (SPD) system :

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \qquad \boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i$$

Matched Feature Points $\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}_1^N$



Jacobians of the 2D coordinate transformations

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 8.1.3)

re-parameterized the motions so that they are identity ***I*** for ***p*** = ***0***.

Jacobians of the 2D coordinate transformations

Translation

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{p}) = \boldsymbol{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} t_x \\ t_y \end{bmatrix}\right) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad \boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial x'}{\partial \boldsymbol{p}} \\ \frac{\partial y'}{\partial \boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Similarity

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{p}) = \boldsymbol{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} t_x \\ t_y \\ a \\ b \end{bmatrix}\right) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \qquad \boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial x'}{\partial \boldsymbol{p}} \\ \frac{\partial y'}{\partial \boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} & \frac{\partial x'}{\partial a} & \frac{\partial x'}{\partial b} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} & \frac{\partial y'}{\partial a} & \frac{\partial y'}{\partial b} \end{bmatrix} = \begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$$

Affine

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{p}) = \boldsymbol{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} t_x \\ t_y \\ a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{bmatrix}\right) = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial x'}{\partial \boldsymbol{p}} \\ \frac{\partial y'}{\partial \boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} & \frac{\partial x'}{\partial a_{00}} & \frac{\partial x'}{\partial a_{01}} & \frac{\partial x'}{\partial a_{10}} & \frac{\partial x'}{\partial a_{11}} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} & \frac{\partial y'}{\partial a_{00}} & \frac{\partial y'}{\partial a_{01}} & \frac{\partial y'}{\partial a_{10}} & \frac{\partial y'}{\partial a_{11}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$$

Motion Alignment in Translation Case

Matched Feature Points $\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}_1^N = \left\{\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix}, \begin{bmatrix} x'_i \\ y'_i \end{bmatrix}\right)\right\}_1^N$ $\quad \{\Delta \boldsymbol{x}_i\}_1^N = \left\{\begin{bmatrix} x'_i - x_i \\ y'_i - y_i \end{bmatrix}\right\}_1^N$

$$\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \qquad \boldsymbol{A} = \sum_{i=1}^{N} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} N & 0 \\ 0 & N \end{bmatrix}$$

$$\boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta \boldsymbol{x}_i \qquad \boldsymbol{b} = \sum_{i=1}^{N} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x'_i - x_i \\ y'_i - y_i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N}(x'_i - x_i) \\ \sum_{i=1}^{N}(y'_i - y_i) \end{bmatrix}$$

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \qquad \begin{bmatrix} N & 0 \\ 0 & N \end{bmatrix}\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N}(x'_i - x_i) \\ \sum_{i=1}^{N}(y'_i - y_i) \end{bmatrix}$$

use python library to solve this

Motion Alignment in similarity transform

Matched Feature Points $\{(\boldsymbol{x}_i, \boldsymbol{x'}_i)\}_1^N = \left\{\left(\begin{bmatrix} x_i \\ y_i \end{bmatrix}, \begin{bmatrix} x_i' \\ y_i' \end{bmatrix}\right)\right\}_1^N$ $\quad \{\Delta\boldsymbol{x}_i\}_1^N = \left\{\begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix}\right\}_1^N = \left\{\begin{bmatrix} x_i' - x_i \\ y_i' - y_i \end{bmatrix}\right\}_1^N$

$$\boldsymbol{A} = \sum_{i=1}^{N} \begin{bmatrix} 1 & 0 & x_i & -y_i \\ 0 & 1 & y_i & x_i \\ x_i & y_i & x_i^2 + y_i^2 & 0 \\ -y_i & x_i & 0 & x_i^2 + y_i^2 \end{bmatrix} \qquad \boldsymbol{b} = \sum_{i=1}^{N} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ x_i & -y_i \\ y_i & x_i \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ x_i \Delta x_i - y_i \Delta y_i \\ y_i \Delta x_i + x_i \Delta y_i \end{bmatrix}$$

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \qquad \begin{bmatrix} N & 0 & \sum_{i=1}^{N} x_i & -\sum_{i=1}^{N} y_i \\ 0 & N & \sum_{i=1}^{N} y_i & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & \sum_{i=1}^{N} x_i^2 + y_i^2 & 0 \\ -\sum_{i=1}^{N} y_i & \sum_{i=1}^{N} x_i & 0 & \sum_{i=1}^{N} x_i^2 + y_i^2 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \Delta x_i \\ \sum_{i=1}^{N} \Delta y_i \\ \sum_{i=1}^{N} x_i \Delta x_i - y_i \Delta y_i \\ \sum_{i=1}^{N} y_i \Delta x_i + x_i \Delta y_i \end{bmatrix}$$

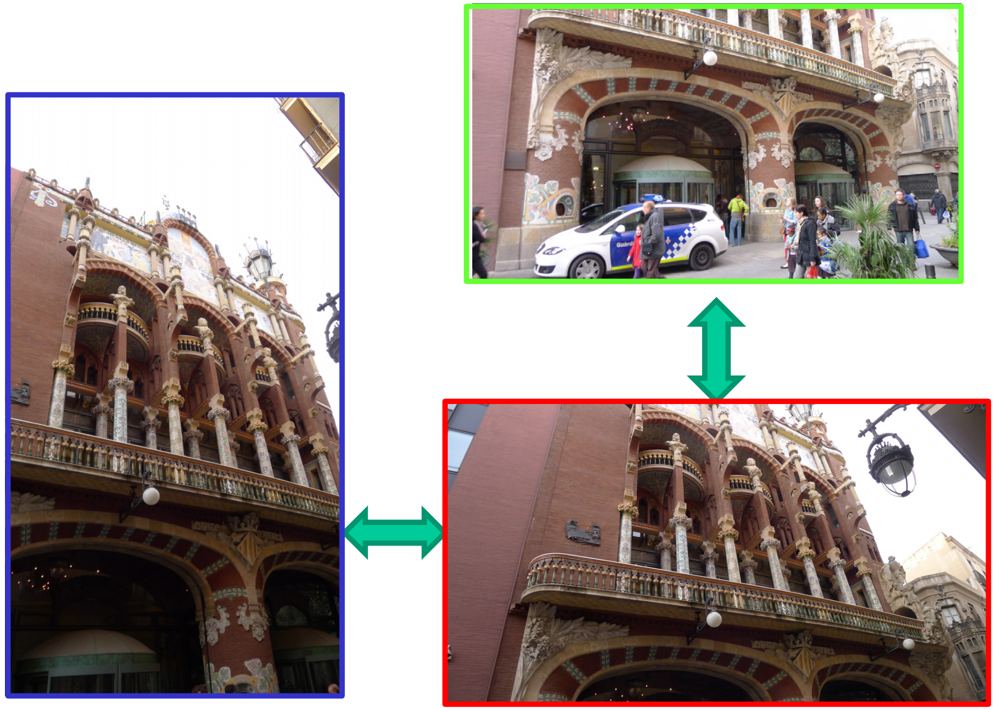
4’. Use OpenCV function for estimating homography.

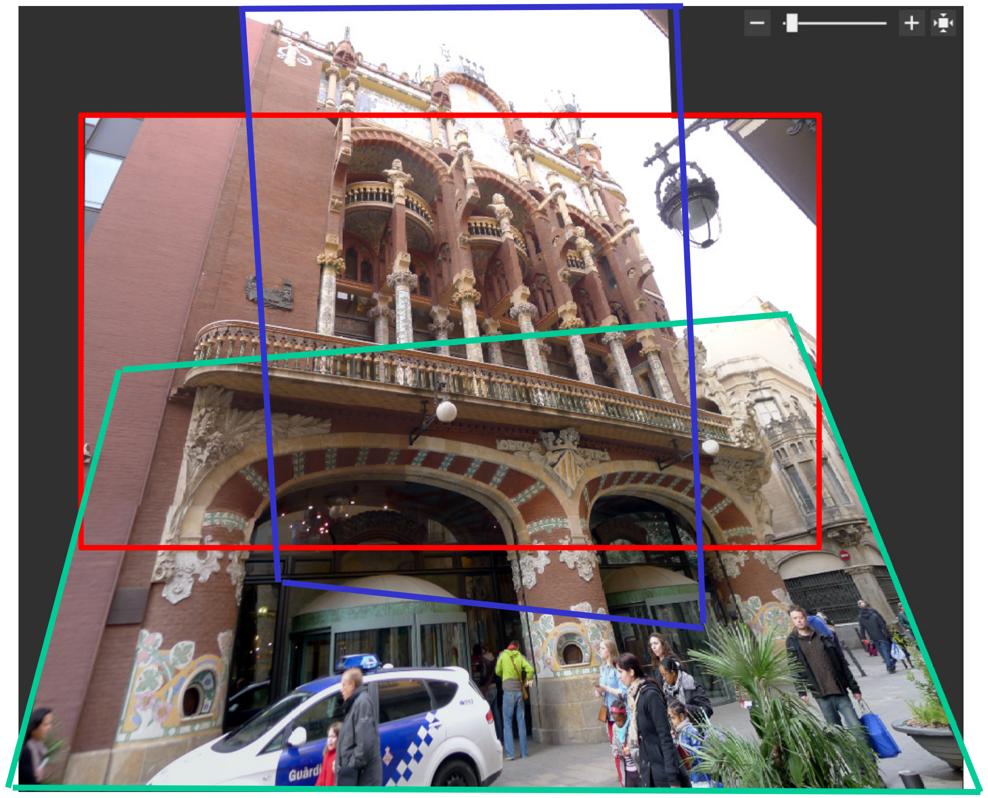
cv2.getPerspectiveTransform

https://www.devdoc.net/linux/OpenCV-3.2.0/da/d6e/tutorial_py_geometric_transformations.html

```
pts1 = np.float32([[176,187],[465,94],[98, 411],[572,356]])
pts2 = np.float32([[100,200],[500,200],[100,700],[500, 700]])
M = cv2.getPerspectiveTransform(pts1,pts2)
```

5. Compute the size of the resulting composite canvas.





6. Warp each image into its final position on the canvas.

cv2.warpPerspective

```
dst = cv2.warpPerspective(img, M, (600, 800))
```

Stitching and Panograph

This exercise is a similar as Ex 8.2: Panography in the reference book p.549.

Panograph and **stitching** are both methods of combining multiple photographs to create a larger image, but with different objectives and results.

A **panograph** (https://edelberto-cabrera.pixels.com/featured/panograph-of-a-beach-edelberto-cabrera.html)
- an artistic technique where overlapping photos are arranged in a non-linear, mosaic-like fashion, emphasizing their individual perspectives and angles, resulting in a creative, fragmented appearance.
- celebrates the differences between individual photos

stitching (or photo stitching)
- aims to produce a seamless, cohesive image.
- aligns and blends photos carefully to ensure continuity in perspective, scale, and lighting, commonly used for panoramic images in landscape and architectural photography.
- seeks to make these differences invisible.

Report submission

- Deadline: June 9
- Language: English or Japanese
- Content: Goal, Method, Results, Discussion, Conclusion

Discussion required to deal with at least these 2 issues in the discussion

- Comparison of different geometric transforms pros and cons
- Comparison with recent applications for stitching
  - https://paperswithcode.com/task/image-stitching
  - https://mattabrown.github.io/autostitch.html
  - https://www.microsoft.com/en-us/research/project/image-composite-editor/