

Homework 2

Computer Vision 2024 Spring

2024.4.25

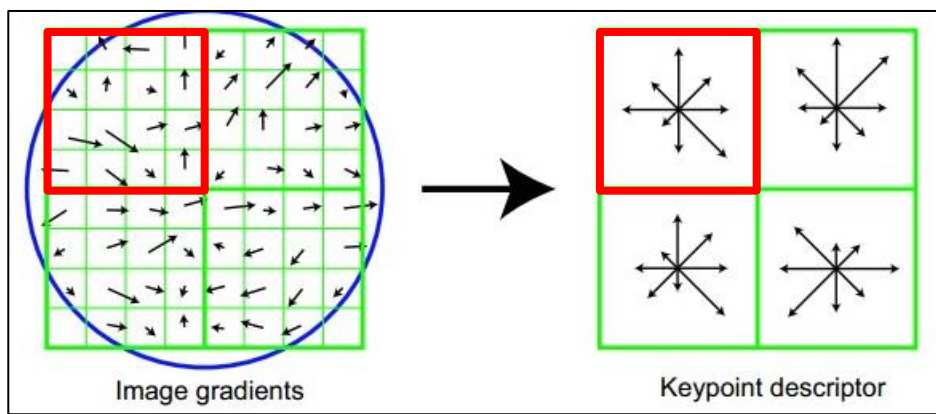
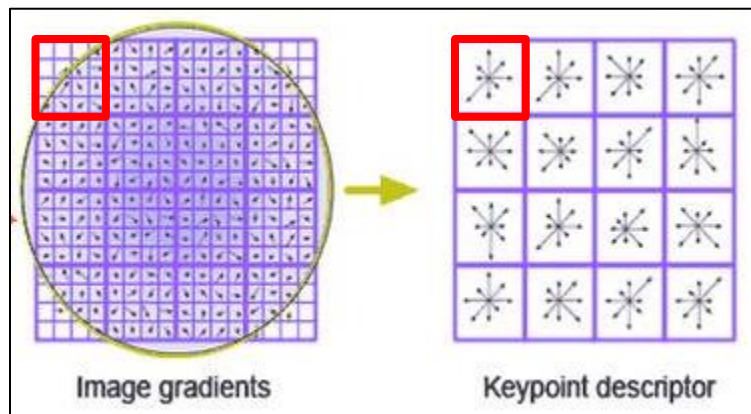
Image stitching

1. Detecting key point(feature) on the images
 - SIFT
2. Finding features correspondences (feature matching)
 - KNN
3. Computing homography matrix.
 - RANSAC
4. Stitching image (warp images into same coordinate system)
 - Homography



Feature Detection

- Finding features correspondences/compute homography matrix.
- SIFT – Scale Invariant Feature Detection
 - detect key points in the image and describe the points as 128-dimensional features ($4 * 4 * 8$).
- Check Ch.6 、 7 for more details of SIFT.



1. SIFT in OpenCV

- Using OpenCV to detect SIFT key points of two images
- Input : **gray scale** image

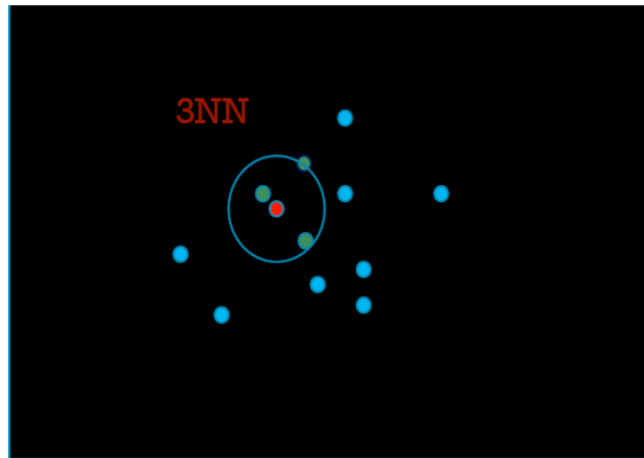
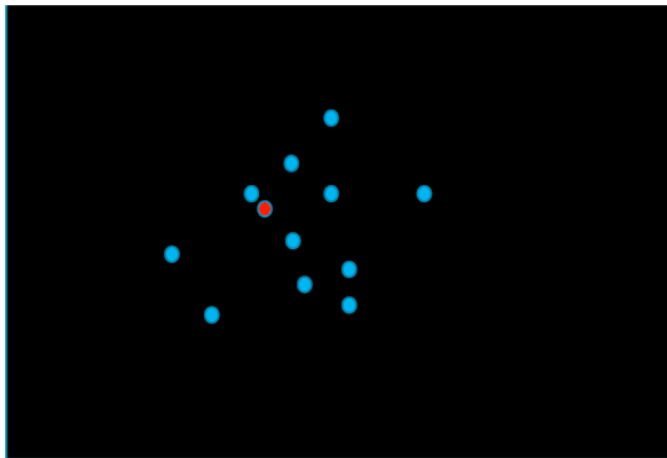
```
SIFT_Detector = cv2.SIFT_create()  
kp, des = SIFT_Detector.detectAndCompute(img, None)
```

- output : keypoints (array), Descriptors (array)
- Keypoints store feature points
 - for a single keypoint you can use “.pt” to get the position of this key point on image [[Ref](#)]
- Descriptors store the 128-dimensional features
- The **function name(detectAndCompute)** of SIFT may be different with the **version of OpenCV**

2. Feature matching - KNN

- K-Nearest Neighbor

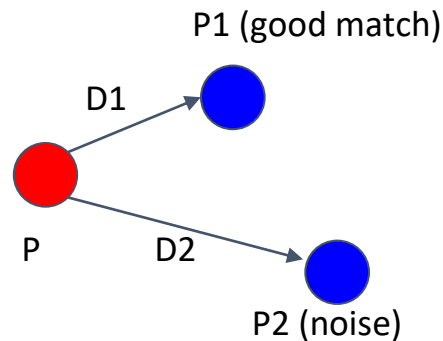
- Finding the K closest neighbors to the target.
- **Brute-force** : Comparing with the all **2-norm of SIFT feature (the 2-norm of descriptor)**



2. Feature matching - Lowe's Ratio test

- Lowe's Ratio test for eliminating bad match
 - A good match should be able to be distinguished from noise
1. For every key point **P** in image1 using 2NN to get 2 matched key points **P1** & **P2** in image2
 2. Computing the 2-norm of **P1** & **P2** between **P** named **D1** , **D2**
 3. If **D1** < **threshold** * **D2** then **P1** is a good match

(threshold is a programmer defined ration between 0 to 1 , the suggestion of OpenCV tutorial is 0.7~0.8)



3. Homography

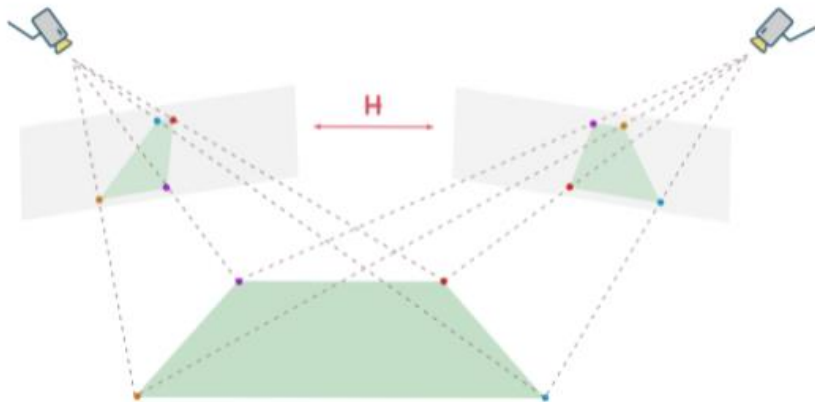
- Construct a linear system as: $\mathbf{P2} = \mathbf{H}\mathbf{P1}$, $\mathbf{P2} = (x_2, y_2, 1)$, $\mathbf{P1} = (x_1, y_1, 1)$ where $\mathbf{P2}$ and $\mathbf{P1}$ are correspondence points, \mathbf{H} is homography matrix.

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \Leftrightarrow \mathbf{x}_2 = \mathbf{H}\mathbf{x}_1$$

$$x'_2 = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1}$$
$$y'_2 = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1}$$

$$x'_2(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{11}x_1 + H_{12}y_1 + H_{13}$$
$$y'_2(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{21}x_1 + H_{22}y_1 + H_{23}$$

In homogenous coordinates ($x'_2 = x_2/z_2$ and $y'_2 = y_2/z_2$)



3. Homography

- If we restrict $h_{33} = 1$

$$\begin{aligned} x'_2 (H_{31}x_1 + H_{32}y_1 + 1) &= H_{11}x_1 + H_{12}y_1 + H_{13}z_1 \\ y'_2 (H_{31}x_1 + H_{32}y_1 + 1) &= H_{21}x_1 + H_{22}y_1 + H_{23}z_1 \end{aligned}$$



$$\begin{aligned} x'_2 &= H_{11}x_1 + H_{12}y_1 + H_{13}z_1 - H_{31}x_1x'_2 - H_{32}y_1x'_2 \\ y'_2 &= H_{21}x_1 + H_{22}y_1 + H_{23}z_1 - H_{31}x_1y'_2 - H_{32}y_1y'_2 \end{aligned}$$

- For perspective transformation, you can use **4 pairs of match result** to solve **8** unknown variable in homography matrix

$$\begin{bmatrix} \hat{x}_i z_a \\ \hat{y}_i z_a \\ z_a \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & \boxed{h_{33}} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1\hat{x}_1 & -y_1\hat{x}_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2\hat{x}_2 & -y_2\hat{x}_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3\hat{x}_3 & -y_3\hat{x}_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4\hat{x}_4 & -y_4\hat{x}_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1\hat{y}_1 & -y_1\hat{y}_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2\hat{y}_2 & -y_2\hat{y}_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3\hat{y}_3 & -y_3\hat{y}_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4\hat{y}_4 & -y_4\hat{y}_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = h_{33} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$

3. Homography

- Let $h_{33} = 1$
- You can solve the equation $Ah = b$ below by **pseudo inverse**

$$\begin{bmatrix} \hat{x}_i z_a \\ \hat{y}_i z_a \\ z_a \end{bmatrix} = \overset{\mathbf{H}}{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\overset{\mathbf{A}}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1\hat{x}_1 & -y_1\hat{x}_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2\hat{x}_2 & -y_2\hat{x}_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3\hat{x}_3 & -y_3\hat{x}_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4\hat{x}_4 & -y_4\hat{x}_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1\hat{y}_1 & -y_1\hat{y}_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2\hat{y}_2 & -y_2\hat{y}_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3\hat{y}_3 & -y_3\hat{y}_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4\hat{y}_4 & -y_4\hat{y}_4 \end{bmatrix}} \overset{\mathbf{h}}{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}} = \overset{\mathbf{b}}{h_{33} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}}$$

3. Homography

$$A = U\Sigma V^T$$

- Using **SVD decomposition** to find Least Squares error solution of **$Ah = 0$**
- the solution = eigenvector of $A^T A$ associated with the smallest eigenvalue (V stores the eigenvector of $A^T A$, Σ stores the singular value (root of eigen value))
find the **smallest number** in Σ and **H = corresponding vector in V^T**
- Remember to **normalize h33 to 1**

$$\begin{aligned} \mathbf{h} &= (H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33})^T \\ \mathbf{a}_x &= (-x_1, -y_1, -1, 0, 0, 0, x'_2 x_1, x'_2 y_1, x'_2)^T \\ \mathbf{a}_y &= (0, 0, 0, -x_1, -y_1, -1, y'_2 x_1, y'_2 y_1, y'_2)^T \end{aligned}$$

$$A = \begin{pmatrix} \mathbf{a}_{x1}^T \\ \mathbf{a}_{y1}^T \\ \vdots \\ \mathbf{a}_{xN}^T \\ \mathbf{a}_{yN}^T \end{pmatrix}.$$

You can multiply a minus to match the form in previous slide **A** is a 9 by 9 matrix
(It's similar to A in previous slide)

Reference :

SVD : https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

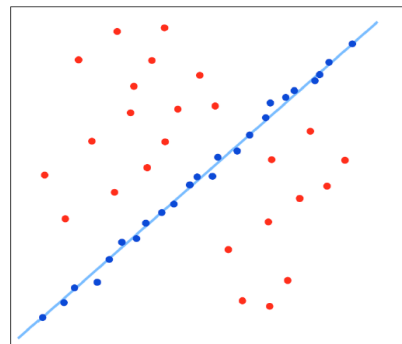
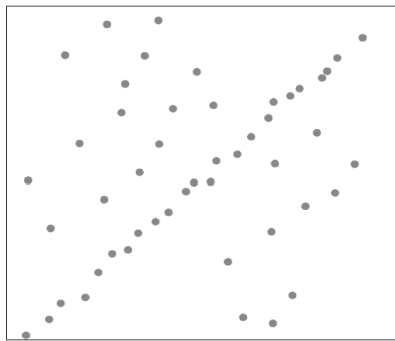
Homography : https://cseweb.ucsd.edu/classes/wi07/cse252a/homography_estimation/homography_estimation.pdf

3.RANSAC

Random Sample Consensus

Input : M match;

1. Randomly select 4 data points as inliers S . Find a homography matrix H to S .
2. Test all $\text{match}(p_1, p_2)$ against H , estimate $p_2' = p_1 * H$
- if the distance between p_2' and p_2 is small, add the match to S , which is called a consensus set.
3. If $|S|$ is larger than ever, mark H as the best estimated H^* .
4. If some stopping criterion is satisfied, end
5. Else go to step 1.



Note that you can re-estimate the models with the consensus sets.

4. Stitching image

1. Using homography matrix **H** to calculate the position of 4 corners of image1 in the perspective of image2
2. Using image1 after perspective transformation to analyze the **size** which we need to combine two image together of
3. Using **cv2.warpPerspective(src, M, dsize, ...)** to warp the whole image1
 - **src** is source **image1** , **M** is homography matrix **H** , **dsize** is output image **size**

```
warped_1 = cv2.warpPerspective(src=img1, M=H, dsize=size)
```
4. Concating two images (for better results you can use **blending** or some ways to improve the quality of overlap part)

For **stitching images** you can use **any function of OpenCV**

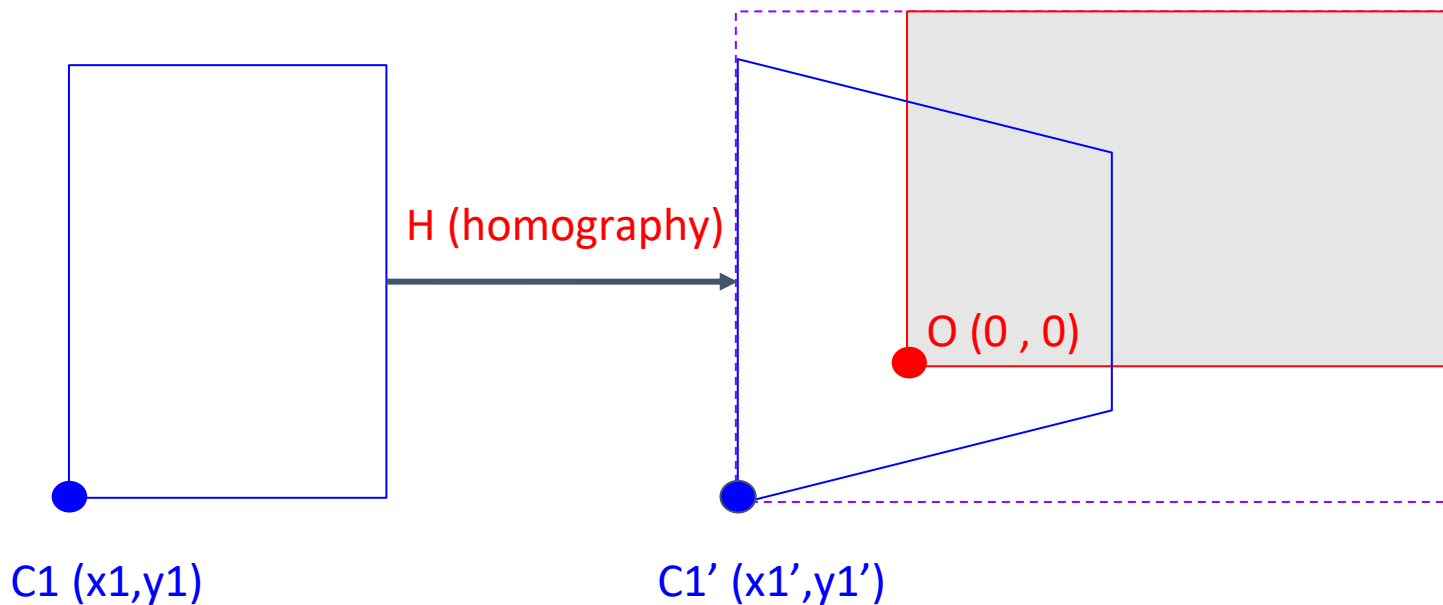
4. Stitching image - more detail

$$\begin{aligned}\text{corners}' &= \text{corners} * H \\ x1' &= \min(\min(\text{corners}'_x), 0) \\ y1' &= \min(\min(\text{corners}'_y), 0)\end{aligned}$$

- Assume **image1** is on left hand side and **image2** is on right hand side
- **Size** we need = ($w2 + \text{abs}(x1')$, $h2 + \text{abs}(y1')$)

width of image2 = $w2$

height of
image2 = $h2$

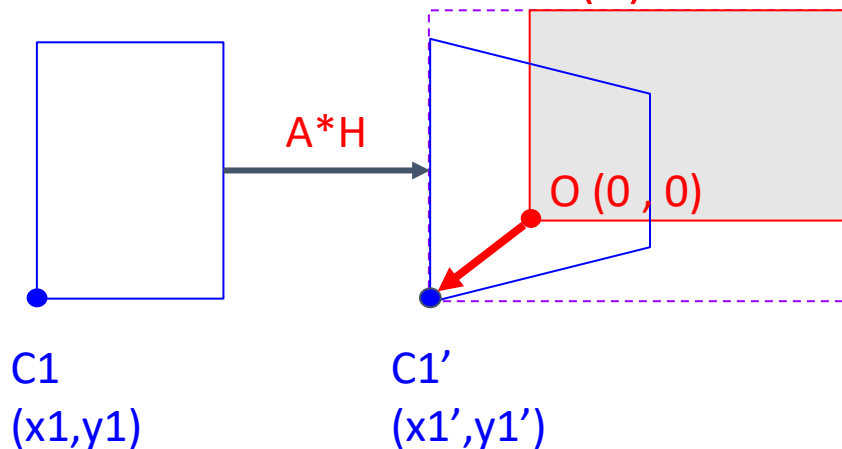


4. Stitching image - more detail

- For both images using **affine translation** to move the origin **O** to **C1'**. This way gives some space to image1 and it's easier to combine them in same image size.
- For image1 your homography matrix(H)s need multiply translation matrix (A) because we translate the perspective of image2
- For image2 you can directly warp with **affine translation matrix(A)**

Affine translation matrix (A) =
$$\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

In this case $\Delta x = -x1'$, $\Delta y = -y1'$
Because origin moves to down left,
image2 needs move to top right relatively



4. Stitching image - more detail

- Example for image1 applies perspective transformation

```
warped_1 = cv2.warpPerspective(src=img1, M=H, dsize=size)
```



$A * H$



4. Stitching image - more detail

- Example for using affine translation to move the image2 origin O to $C1'$

```
warped_r = cv2.warpPerspective(src=img2, M=A, dsize=size)
```



A



Requirements

- You are only allowed to use the function of OpenCV mentioned in previous slides. Please implement all (key point matching ,RANSAC , Homography , cylindrical projections,...) by yourself
 - For submission you can use :
 - SIFT
 - For **debugging** only:
 - KNN match : `BFMatcher()`
 - Homography : `findHomography()`
- But there is no limitation of “**image stitching**” only (You can use any function provided by OpenCV)

Grading

50% Image stitching

- SIFT (10%)
- KNN (10%)
- RANSAC (15%)
- Homography (15%)

30% Report (Don't just paste the code with comment)

- 1.Explain your implementation
- 2.Show the result of stitching “Base” images (and “Challenge image” if you did that part).
- 3.Discuss different blending method result.

(The below **two** part of grade you got may “vary” according to the stitched image “**quality**”, methods you used, and so forth.)

10% **Stitching 3 images** seamlessly with blending and show result in report

10% **Challenging task**: stitching 6 modified photos with gain compensation method applied, and then showing the result in the report and explaining what you did.

Deadline

- Deadline : 2024/05/12 (Sun.) 11:59 pm
- Please zip the all files and name it as {studentID}_HW2.zip :
ex 312550000_HW2.zip (wrong file format may get -10% penalty)
 - Zip file format:
 - 1. {studentID}_report.pdf
 - 2. your code
- Penalty of 3% of the value of the assignment per late day
 - late a day : $\text{your_score} * 0.97$
 - late two days : $\text{your_score} * 0.94$
- Feel free to send e-mails through E3 platform to all TAs for personal questions.

Example Result

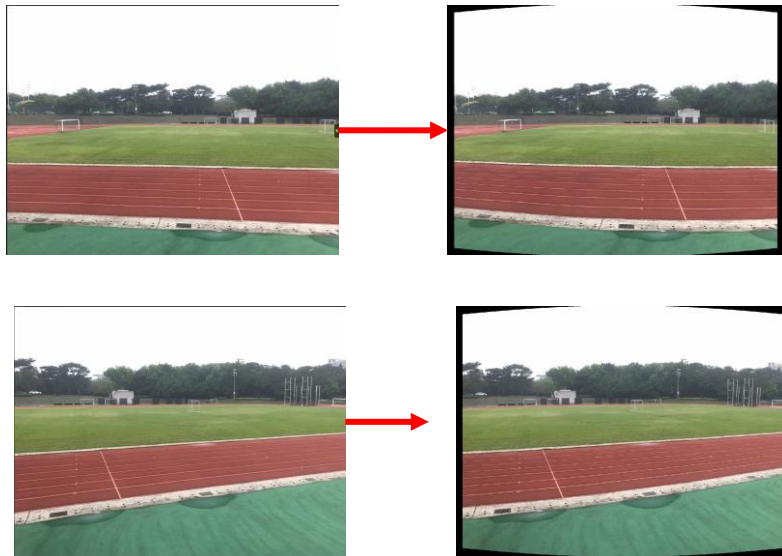


Sample of concatenating all 3 Base images

Others Tips

- Using different blending method to improve the quality
- Preprocessing for more easily concatenate multiple image :

Cylindrical projection



Cylindrical projection on images



Bad blending method example

Challenging Task-Gain compensation

$$e = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n N_{ij} ((g_i \bar{I}_{ij} - g_j \bar{I}_{ji})^2 / \sigma_N^2 + (1 - g_i)^2 / \sigma_g^2)$$



Challenging Task-Gain compensation

- Stitching them seamlessly and show how you find your optimizing “g” in the report

