ECE661 Computer Vision - Fall 2014 - HW5

Fu-Chen Chen chen1623@purdue.edu

1. Feature extraction and matching

I use the SURF in OpenCV for feature extraction and matching. After matching, I also compare the distance between corresponding feature's descriptors. If the distance is too large, I will discard the correspondence to reduce the number of outliers.

2. RANSAC algorithm

1. Determine the parameter δ, ε, ρ where:
   - δ is the distance threshold for defining inlier and outlier.
   - ε is the expecting ration of outliers.
   - ρ is the targeting probability of at least picking all the inliers.

2. Compute the variable N, M where:
   - N is the expected # of iteration of RANSAC = $\frac{\mathrm{Ln}(1-\rho)}{\mathrm{Ln}(1-(1-\varepsilon)^4)}$
   - M is the expected # of inliers = (1-ε)*(Total # of correspondences)

3. Start the RANSAC iterations until the exit conditions is fulfilled (conditions will be mentioned in step 4). In each iteration, randomly pick 4 correspondences of SURF points and use them to derive the homography with the same manner in HW2. After getting the homography, use it to generate the corresponding points of features and get the distances between these corresponding points and the SURF points. If the distance is smaller than δ, the SURF points are the inliers. Otherwise, they are the outliers. Keep the homography with the most number of inliers as the best homography.

4. Exit the RANSAC iteration if # of iteration >= N and # of inliers >= M. These will ensure the quality of derived homography. Sometimes the condition may not be fulfilled since there are too many outliers than expected, for safety I set a iteration upper-bound for terminating the RANSAC if # of iteration is bigger than the upper-bound.

3. Least square method

After getting the best homography from RANSAC, we apply it again to all the SURF points and store all the inlier correspondences:

$$X_i \rightarrow X'_i \text{ for } i = 1 \sim n, \quad \text{where } X_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, X' = \begin{bmatrix} x'_i \\ y'_i \end{bmatrix}, n = \text{total \# of inlier}$$

Let $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$ is the homography we want to have, then we could have following equations:

$$x_i' = x_i h_{11} + y_i h_{12} + h_{13} - x_i' x_i h_{31} - x_i' y_i h_{32}$$
$$y_i' = x_i h_{21} + y_i h_{22} + h_{23} - y_i' x_i h_{31} - y_i' y_i h_{32}$$

Rewrite them in matrix form:

$$Ah = B$$

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1' x_1 & -x_1' y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1' x_1 & -y_1' y_1 \\ & \vdots & & & \vdots & & \vdots & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n' x_n & -x_n' y_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n' x_n & -y_n' y_n \end{bmatrix}_{2nx8}, h = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}_{8x1}, B \begin{bmatrix} x_1' \\ y_1' \\ \vdots \\ x_n' \\ y_n' \end{bmatrix}_{2nx1}$$

The solution h such that $||B - Ah||$ is minimized would be:

$$h = (A^T A)^{-1} A^T B$$

So we could use all the inlier correspondences to form the matrix A and B. Then we could derive the vector h so as to the homography H.

4. Dog-Leg algorithm

Let $p_k = [h_{11} \quad h_{12} \quad h_{13} \quad h_{21} \quad h_{22} \quad h_{23} \quad h_{31} \quad h_{32} \quad h_{33}]^T$ be a 9x1 vector (and with all the definitions of variables mentioned above) at the iteration k of Dog-Leg algorithm. We want to update the p_k to minimize the geometry error.

In each iteration, we computer the Jacobian and the error vector:

$$J(p_k) = \begin{bmatrix} x_1/\widehat{w_1} & y_1/\widehat{w_1} & 1/\widehat{w_1} & 0 & 0 & 0 & -\widehat{x_1}x_1/\widehat{w_1}^2 & -\widehat{x_1}y_1/\widehat{w_1}^2 & -\widehat{x_1}/\widehat{w_1}^2 \\ 0 & 0 & 0 & x_1/\widehat{w_1} & y_1/\widehat{w_1} & 1/\widehat{w_1} & -\widehat{y_1}x_1/\widehat{w_1}^2 & -\widehat{y_1}y_1/\widehat{w_1}^2 & -\widehat{y_1}/\widehat{w_1}^2 \\ & \vdots & & & \vdots & & & \vdots & \\ x_n/\widehat{w_n} & y_n/\widehat{w_n} & 1/\widehat{w_n} & 0 & 0 & 0 & -\widehat{x_n}x_n/\widehat{w_n}^2 & -\widehat{x_n}y_n/\widehat{w_n}^2 & -\widehat{x_n}/\widehat{w_n}^2 \\ 0 & 0 & 0 & x_n/\widehat{w_n} & y_n/\widehat{w_n} & 1/\widehat{w_n} & -\widehat{y_n}x_n/\widehat{w_n}^2 & -\widehat{y_n}y_n/\widehat{w_n}^2 & -\widehat{y_n}/\widehat{w_n}^2 \end{bmatrix},$$

$$e(p_k) = \left[(x_1' - \frac{\widehat{x_1}}{\widehat{w_1}}) \quad (y_1' - \frac{\widehat{y_1}}{\widehat{w_1}}) \quad \cdots \quad (x_n' - \frac{\widehat{x_n}}{\widehat{w_n}}) \quad (y_n' - \frac{\widehat{y_n}}{\widehat{w_n}}) \right]^T$$

where

$$\widehat{x_k} = h_{11}x_k + h_{12}y_k + h_{13}$$
$$\widehat{y_k} = h_{21}x_k + h_{22}y_k + h_{23}$$
$$\widehat{w_k} = h_{31}x_k + h_{32}y_k + h_{33}$$

Then we can have:

$$\delta_{GD}(p_k) = \frac{\left\| J(p_k)^T e(p_k) \right\|}{\left\| J(p_k) J(p_k)^T e(p_k) \right\|} J(p_k)^T e(p_k)$$

$$\delta_{GN}(p_k) = (J(p_k)^T J(p_k) + u_k I)^{-1} J(p_k)^T e(p_k)$$

with a constant u_k.

Now we can update p_k according to the rule:

$$p_{k+1} = p_k + \begin{cases} \delta_{GN}(p_k), \text{if } \|\delta_{GN}(p_k)\| < r_k \\ \delta_{GD}(p_k) + \beta(\delta_{GN}(p_k) - \delta_{GD}(p_k)) \\ \frac{r_k}{\|\delta_{GD}(p_k)\|} * \delta_{GD}(p_k), \text{otherwise} \end{cases}, \text{if } \|\delta_{GD}(p_k)\| < r_k < \|\delta_{GN}(p_k)\|$$

where β is the solution of

$$\|\delta_{GD}(p_k)\|^2 + \beta^2 \|\delta_{GN}(p_k) - \delta_{GD}(p_k)\|^2 + 2\beta \delta_{GD}(p_k)^T (\delta_{GN}(p_k) - \delta_{GD}(p_k)) = {r_k}^2$$

and with a constant r_k.

For the first iteration, we set $u_0 = \gamma * \max(\text{diag}(J(p_0)^T J(p_0)))$ with $0<\gamma<<1$ and set $r_0 = 0.5 \sim 6.0$. For the following iterations, we update them by:

$$u_{k+1} = u_k * \max\left(\frac{1}{3}, 1 - (2\rho LM - 1)^3\right)$$

where

$$\rho LM = \frac{C(p_k) - C(p_{k+1})}{{\delta_p}^T (u_k \delta_p + J(p_k)^T e(p_k))}, \qquad \delta_p = p_{k+1} - p_k, \qquad C(p_k) = e(p_k)^T * e(p_k)$$

$$r_{k+1} = \begin{cases} \frac{r_k}{4}, \text{if } \rho DL < 0.25 \\ r_k, \text{if } 0.25 \le \rho DL \le 0.75 \\ 2r_k, \text{otherwise} \end{cases}$$

where

$$\rho DL = \frac{C(p_k) - C(p_{k+1})}{2\delta_p{}^T J(p_k)^T e(p_k) - \delta_p{}^T J(p_k)^T J(p_k)\delta_p}$$

At the end of each iteration, check if the total error is decreasing or increasing by the sign of $C(p_k) - C(p_{k+1})$. If the sign is positive, it means the total error is decreasing. If not, it means we take the wrong step so we should undo this iteration and update the constants by:

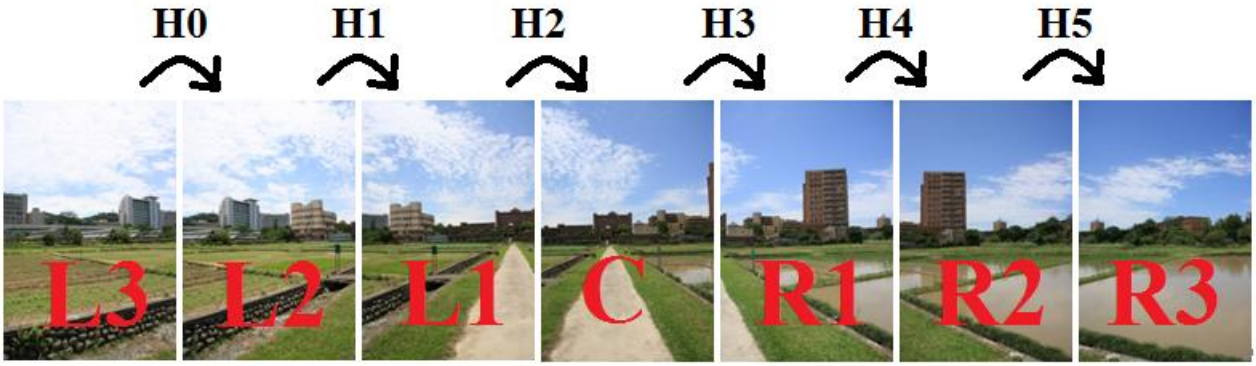
$$p_{k+1} = p_k$$
$$u_{k+1} = 2 * u_k$$
$$r_{k+1} = r_k/2$$

The whole Dog-Leg algorithm ends when the $C(p_k) - C(p_{k+1})$ is positive and lies within a small constant. This means we are very close to the optimal solution so the error only decrease a small value.

5. Image mosaicing method

a. Get homographies of center images

First we derive the homographies in a serial order (ex: from left to right) of images. Then we pick a image as a center image and find the corresponding homegraphies from center image to other images by multiplying the derived homographies and their inverse sucessively.



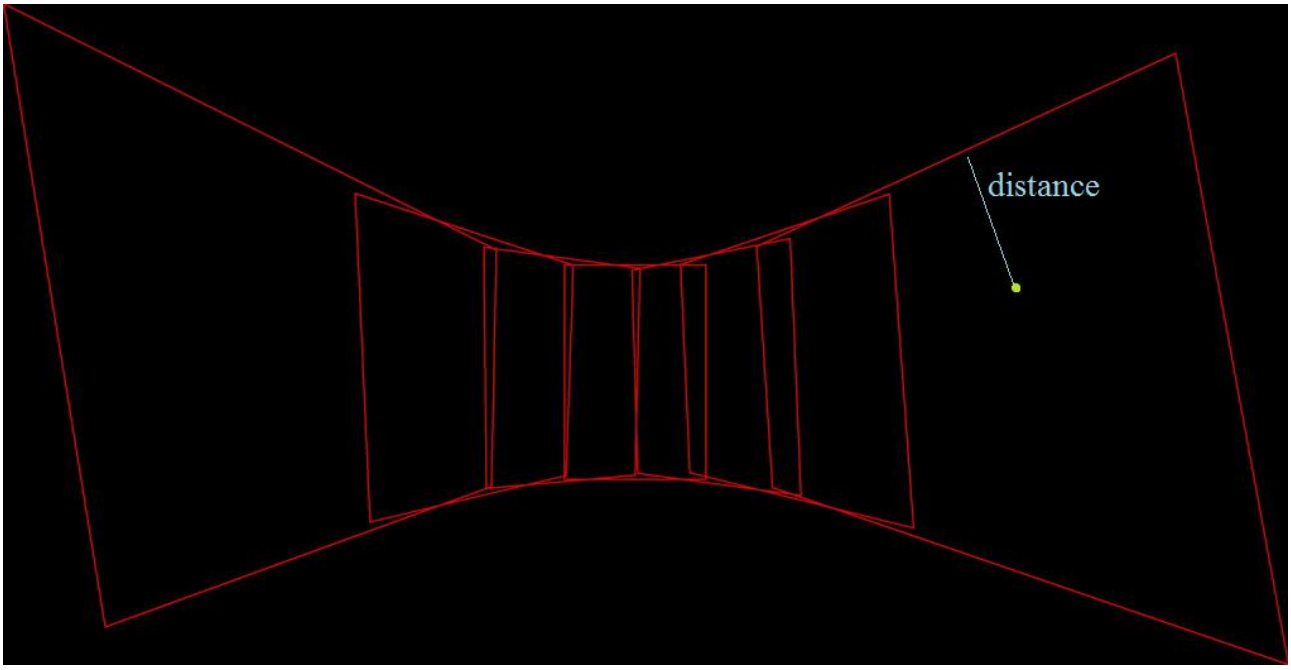
Use above images for example, we have a center image "C", its left, right images and the homographies H0~H5. We could get the corresponding homegraphies from center image to other images by:

For image C to image R1: HCR1 = H3
For image C to image R2: HCR2 = H4* HCR1
For image C to image R3: HCR3 = H5* HCR2

For image C to image L1: HCL1 = $H2^{-1}$
For image C to image L2: HCL2 = $H1^{-1}$HCL1
For image C to image L3: HCL3 = $H0^{-1}$HCL2

b. Perform back-warping with blending

Using the corresponding homegraphies of center image, we can create an expanded image that contain all the back-warped images. Each images' boundaries form a polygon in the expanded image, so we could scan every pixels in expanded image to check if they are within certain polygons, which means we could get the pixels value from corresponding images of the polygons by back-warping and bi-linear interpolation.



One pixel may belong to multiple polygons. If you only take one pixel value from one image of the polygons, there could be very obvious edges on the boundaries of polygons. Thus we perform a blending processing that use the weighted sum of pixel values by considering the distance of the pixel to the boundaries of polygons. The weight of each pixel value of polygon is:

$$\text{weight} = \text{distance} * \text{distance} + 0.001$$

By taking the quadratic form of distance, the weight would be small on boundaries and be very large near the center of boundaries. This let us could perform blending only on the boundaries, and make other regions of image remain un-blur.

Image without blending: Lots of obvious edges

Image with blending: The edge disappear while the image remain un-blur

6. Parameters

SURF:

SURFthr	= 2500	Threshold for extracting SURF feature
MatchThr	= 0.30	Threshold for SURF's feature matching.

RANSAC:

RANSAC_delta	= 4.5	Distance threshold for defining inlier and outlier
RANSAC_epsilon	= 50%	Ratio of outliers
RANSAC_p	= 99.9%	Probability of at least picking all the inlier
RANSAC_TIMEOUT	= 999	Iteration upper-bound to stop RANSAC if there is no good result

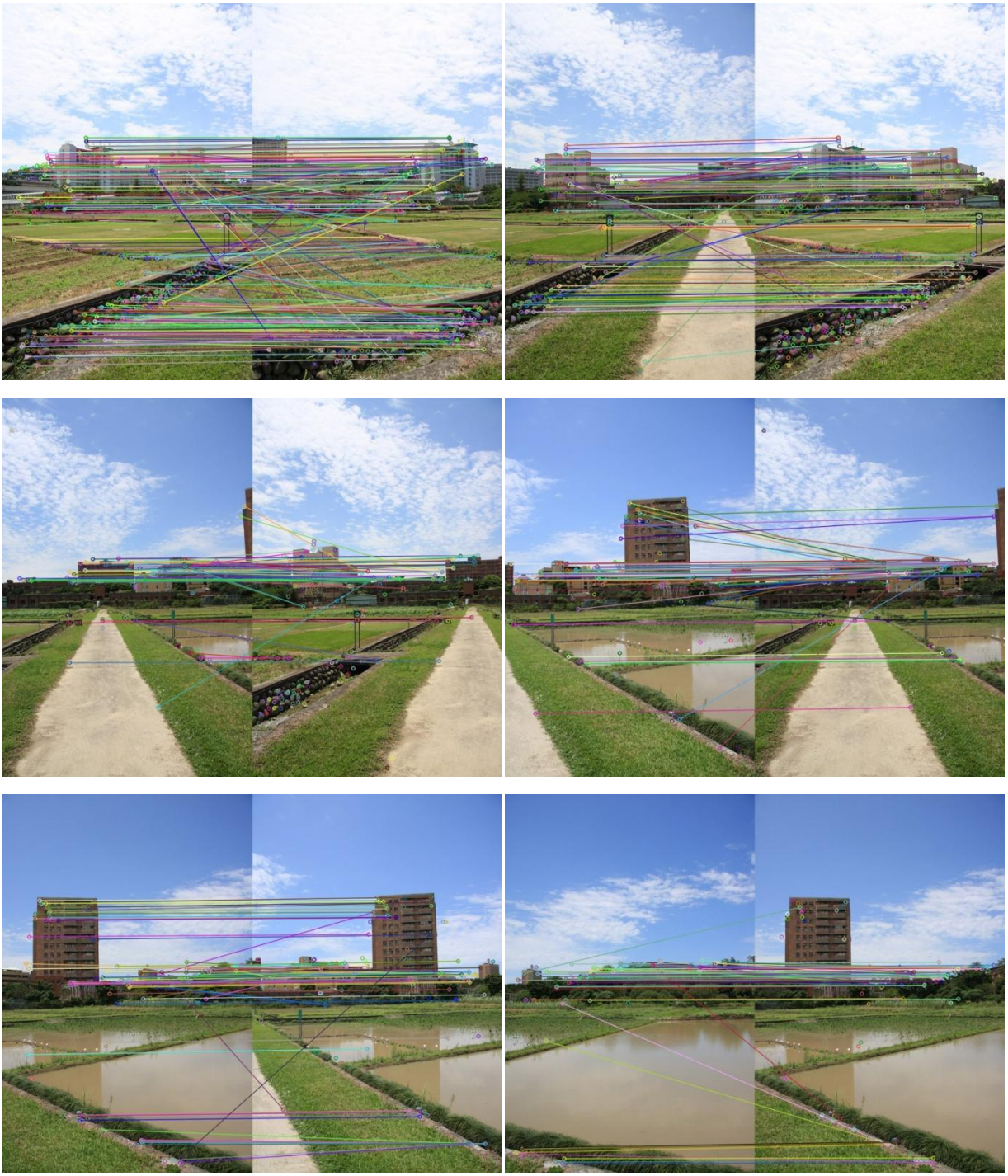
Dog-Leg:

coef_uk	= 0.0000005	Multiplier for initializing uk
coef_rk	= 3.0	Initial value of rk
err_con	= 0.01	Threshold of delta-error to stop the Dog-Leg

7. Results

Set1 - original images

Set1 - feature correspondences

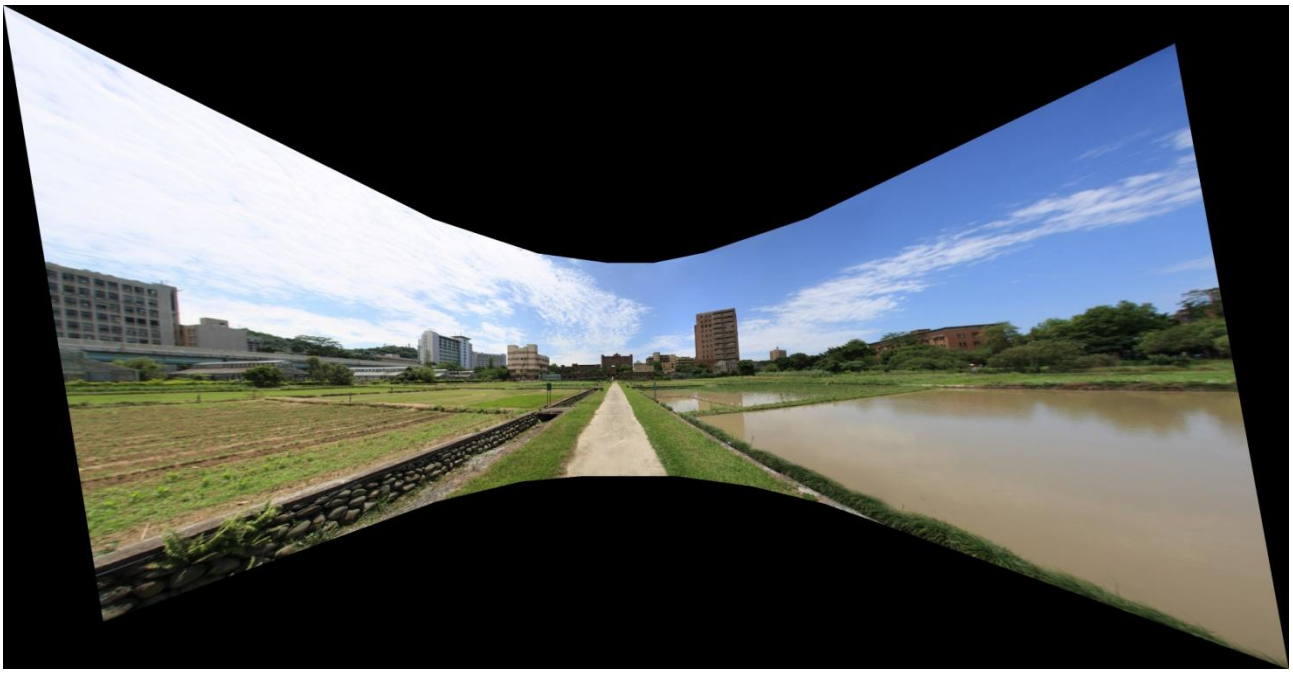


Set1 - outliers(red) and inliers(blue)

Set1 - final results

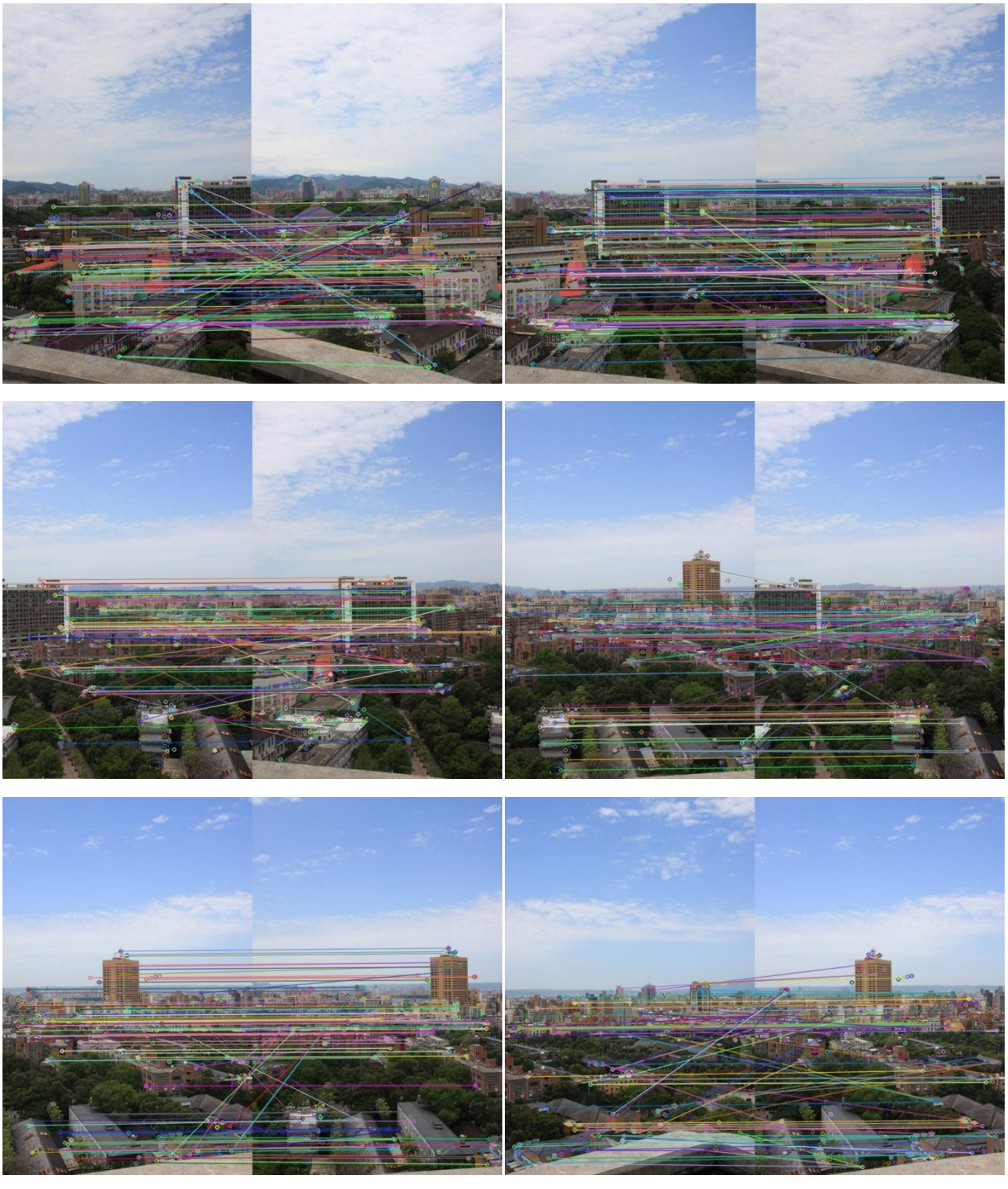
without blending

with blending



Set2 - original images

Set2 - feature correspondences



Set2 - outliers(red) and inliers(blue)

Set2 - final results

without blending

with blending

8. Discussion

- Almost all of the inliers lie on the overlap regions of two images, while the outliers lie on excluded regions.

- Lower the parameter "MatchThr" could significantly reduce the number of outliers. I set MatchThr=0.30 here in order to get more outliers and show the power of RANSAC. If I set MatchThr=0.25, then the ration of outlier could be 10%~20%.

- The geometry error reduce a little after Dog-Leg algorithm. I think it's because the homography comes from the RANSAC and linear-least-square is good enough. In the most cases, the average geometry error (total error / # of inliers) is below 1.0 before applying Dog-Leg, so the solution may be very close the optimal solution.