1 Feature point extraction by SIFT

SIFT algorithm is used to extract feature points in the two images. For easy understanding, let one image be \textit{im}1 and the other be \textit{im}2. Euclidean distance between the descriptor of each feature point on \textit{im}1 and the descriptor of each feature point on \textit{im}2 is measured. Then, the feature point on \textit{im}2 having the smallest distance to a feature point on \textit{im}1 is a corresponding point to the feature point on \textit{im}1. By taking an appropriate threshold to the measured Euclidean distances, some erroneous correspondences can be removed. The remained correspondences of the feature points are used to compute the homography between the two images. The MATLAB built-in function \texttt{vl_sift} from VLFeat open source library is used for the feature extraction in this implementation.

2 RANSAC algorithm

A set of 7 images is taken by a cellphone camera to mosaic the set of images. The RANSAC algorithm is used as the following steps to compute the homography of each pair of images.

2.1 Parameter setting

\( n = 6 \) correspondences are randomly chosen at each trial to compute a homography in Linear Least Square sense, and the repeated chosen is not permitted. The 10\% rule is used to set \( \delta \) for this implementation. Therefore, the correspondences having the top 10\% values of \( \|x' - Hx\|^2 \) are regarded as the outliers when the correspondences \((x, x')\) are ranked in ascending order. The number of iteration \( N \) is

\[
N = \frac{\ln(1-p)}{\ln[1-(1-\epsilon)^n]},
\]

which yields \( N = 6 \) for \( \epsilon = 0.1 \) and \( n = 6 \) in our case. The number of inliers \( M \) is

\[
M \simeq (1-\epsilon) \cdot n_{total},
\]

where \( n_{total} \) is the number of detected correspondences by SIFT in Sec. 1.
2.2 Homography obtained by inliers

For $N$ trials, $N$ homographies are obtained by the 6 correspondences chosen at each trial. The most fittable homography should be selected among the $N$ homographies in order to distinguish the outliers. The most fittable homography is the homography having the smallest $\sum \|x' - Hx\|^2$. Then, the correspondences having the top 10% values of $\|x' - Hx\|^2$ are regarded as the outliers and the others are inliers where $H$ is the most fittable homography. Finally, the robust homography is estimated in Linear Least Square sense using the $M$ inlier correspondences. That is, the eigenvector corresponding to the smallest singular value in the SVD decomposition of the matrix $A$ is the column vector having the elements of the robust homography where $A$ is

\[
A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_M \end{bmatrix}, \quad A_k = \begin{bmatrix} \vec{0}^\top & -w'x^\top & y'x^\top \\ w'x^\top & \vec{0}^\top & -x'x^\top \\ \end{bmatrix}, \quad \text{for } k\text{-th } x' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}.
\]

3 Refined homography by Levenberg-Marquardt method

The Levenberg-Marquardt method is used to refined the robust homography with Non-linear Least-Squares Minimization. For the $M$ inlier correspondences given by the RANSAC algorithm, a refined $H$ that minimizes $\sum_{i=1}^{M} \|x_i' - Hx_i\|^2$ can be found by the Levenberg-Marquardt algorithm. The MATLAB built-in function `lsqcurvefit` function is used to apply LM method for this implementation. Following steps and parameters are needed to apply `lsqcurvefit` to the refinement:

\[
\text{h = lsqcurvefit(@fun, h0, x, x', option)}
\]

1. `fun` is a user-defined function which can define the relation between $x$ and $\tilde{x}' = Hx$.
2. `h0` is a starting point to search the minimizer $h$:
   - The column vector form of the robust homography $H$ obtained by LLS is used as $h_0$.
3. $x$ is a column vector having the coordinates of inliers in the domain plain.
4. $x'$ is a column vector having the coordinates of inliers in the range plain.
5. `option` is the option to use Levenberg-Marquardt algorithm to search the minimizer:
   - `option = optimset('Algorithm','Levenberg-Marquardt')` is used to set the algorithm.
6. $h$ is the minimizer found by Levenberg-Marquardt algorithm.

Finally, the transformation of $h$ to a $3 \times 3$ matrix yields the refined homography.
4 Image Mosaicking

We use a set of 7 images for the image mosaicking as mentioned previously. Let the most left image be $im_1$ and the most right image be $im_7$, and let middle images be sequently numbered. Then, the boundaries of the mosaiced image should be set first. The transformed boundary corners of $im_1$ and $im_7$ to the center image plan, $im_4$, contains proper boundary corners of the mosaiced image since they have the most geometric distortions. Therefore, the boundary corners of the mosaiced image should be set to the minimum and maximum $(x,y)$ coordinates among the transformed boundary corners of $im_1$ and $im_7$. After the boundary corners are set, a shifting factor should be measured since image coordinates cannot have negative values. Then, appropriate pixel values can be assigned to the mosaiced image plane using the bilinear interpolation. In this implementation, $im_1$ and $im_7$ are mapped to the mosaiced plane first, and then $im_2$ and $im_6$ are mapped, and last $im_3$ and $im_5$ are mapped. Finally, the center plane $im_4$ is put into the appropriate position of the mosaiced plane.

4.1 Homography to the center plane

For successive mosaicking, right homographies should be used to map every single image to the center plane. Every homography between a pair of two neighbored single images is already obtained using the feature extraction, RANSAC algorithm, LLS, and Levenberg-Marquardt algorithm in the previous sections. We have $H_{12}$, $H_{23}$, $H_{34}$, $H_{54}$, $H_{65}$, and $H_{76}$ where $H_{IJ}$ is the homography mapping $im_I$ to $im_J$. By the chain rule of homography, $H_{24} = H_{23}H_{34}$ and $H_{14} = H_{12}H_{24}$ and $H_{64} = H_{65}H_{54}$ and $H_{74} = H_{76}H_{64}$. Therefore, the 6 homographies ($H_{14}$, $H_{24}$, $H_{34}$, $H_{54}$, $H_{64}$, $H_{74}$) can be applied to properly map every single image to the center plane.

5 Matching results and Parameter values

Following parameters are used and matching results are obtained as shown in the tables below.

<table>
<thead>
<tr>
<th>( T_{ED} )</th>
<th>Threshold for Euclidean distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{total} )</td>
<td>Number of correspondences</td>
</tr>
<tr>
<td>( n )</td>
<td>Number of correspondences chosen to compute a homography at each trial</td>
</tr>
<tr>
<td>( N )</td>
<td>Number of trials in the RANSAC algorithm</td>
</tr>
<tr>
<td>( M )</td>
<td>Number of inliers</td>
</tr>
<tr>
<td>( O )</td>
<td>Number of outliers</td>
</tr>
</tbody>
</table>
Table 2: Matching Results

<table>
<thead>
<tr>
<th>Image Pair</th>
<th>im1 &amp; im2</th>
<th>im2 &amp; im3</th>
<th>im3 &amp; im4</th>
<th>im5 &amp; im4</th>
<th>im6 &amp; im5</th>
<th>im7 &amp; im6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ED}$</td>
<td>35</td>
<td>27</td>
<td>35</td>
<td>32</td>
<td>40</td>
<td>32</td>
</tr>
<tr>
<td>$n_{total}$</td>
<td>59</td>
<td>65</td>
<td>61</td>
<td>60</td>
<td>64</td>
<td>61</td>
</tr>
<tr>
<td>$n$</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$N$</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$M$</td>
<td>53</td>
<td>58</td>
<td>54</td>
<td>54</td>
<td>57</td>
<td>54</td>
</tr>
<tr>
<td>$O$</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

6 Inliers & Outliers

Figure 1: im1.jpg (left) and im2.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.
Figure 2: im2.jpg (left) and im3.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.

Figure 3: im3.jpg (left) and im4.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.
Figure 4: im5.jpg (left) and im4.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.

Figure 5: im6.jpg (left) and im5.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.
Figure 6: im7.jpg (left) and im6.jpg (right): Green lines represent inlier and Red lines represent outlier correspondences.

7 Mosaiced result

Figure 7: Mosaiced image created from all the single images using parameters in the tables.
Figure 8: Additional result of the mosaiced image.

8 Source Codes
Source Code for the SIFT, RANSAC, and Levenberg-Marquardt algorithm

clear; close;

%% Preprocess %%
% Read images
im1 = imread('apart1.jpg');
im2 = imread('apart2.jpg');

% Convert RGB image to grayscale
im1g = single(rgb2gray(im1));
im2g = single(rgb2gray(im2));
[R,C] = size(im1g);

%% SIFT of im1 and im2 by VL_SIFT function %%
[F1,D1] = VL_SIFT(im1g);   % Each column of F is a feature frame
[F2,D2] = VL_SIFT(im2g);   % Each column of D is a discriptor
d = dist(D1',D2);          % Distance between D1's column and D2's column
[Y I] = min(d);
count = 0;                 % Number of non-overlapped correspondences
c1 = zeros(1,2);           % Corresponding feature coordinates of im1
c2 = zeros(1,2);           % Corresponding feature coordinates of im2

%% Find correspondences between two images using Euclidean distance %%
img = [im1,im2];
for k = 1:length(Y)
    ind = 1;               % Indicator to avoid overlapped correspondences
    for l = 1:length(I)
        if l~=k && I(l)==I(k)
            ind = 0;
            break;
        end
    end
    if ind && Y(k) < 35    % Threshold for Euclidean distance
        count = count + 1;
        c1(count,:) = round(F1(1:2,I(k)));
        c2(count,:) = round(F2(1:2,k));
    end
end

%% RANSAC algorithm %%
nc = 6;                  % Number of correspondences used to find a homography
N = fix(log(1-.99)/log(1-(1-.1)^nc));     % Number of trials by 10% rule
M = fix((1-.1)*count);                   % Minimum size for the inlier set
d_min = 1e100;
for n = 1:N
    lcv = 1;                             % Loop control variable
    while lcv                             % To avoid repeated selection
        r = randi(count,nc,1);
        r = sort(r);
for k = 1:nc-1
    lcv = lcv*(r(k+1)-r(k));
end
lcv = -lcv;
end
A = zeros(2*nc,9);
for k = 1:nc
    A(2*k-1:2*k,:)=...
        [0,0,0,-[c1(r(k),:),1],c2(r(k),2)*[c1(r(k),:),1];
        [c1(r(k),:),1],0,0,0,-c2(r(k),1)*[c1(r(k),:),1];
    end
[U,D,V] = svd(A);
h = V(:,9);
H = [h(1),h(2),h(3);h(4),h(5),h(6);h(7),h(8),h(9)];
d2 = zeros(count,1); % d^2(x_measured, x_true)
for k = 1:count
    x_true = H*[c1(k,:),1]'; % x_true in HC
    temp = x_true/x_true(3);
    x_true = temp(1:2); % x_true in image plane
    d = c2(k,:)-x_true';
    d2(k) = d(1)^2+d(2)^2;
end
[Y I] = sort(d2);
if sum(Y(1:M)) < d_min
    d_min = sum(Y(1:M));
inliers = I(1:M);
    outliers = I(M+1:end);
end

% Visualize the inliers and outliers
figure; image(img); true_size; hold on;
for k = inliers'
    plot([c1(k,1),C+c2(k,1)], [c1(k,2),c2(k,2)], '-og', 'linewidth',1);
end
for k = outliers'
    plot([c1(k,1),C+c2(k,1)], [c1(k,2),c2(k,2)], '-or', 'linewidth',1);
end
plot([C,C], [1,R], '-k'); hold off;

%% Linear Least Squares %%
A = zeros(2*M,9);
for k = 1:M
    A(2*k-1:2*k,:) =...
        [0,0,0,-[c1(inliers(k),:),1],c2(inliers(k),2)*[c1(inliers(k),:),1];
        [c1(inliers(k),:),1],0,0,0,-c2(inliers(k),1)*[c1(inliers(k),:),1];
end
[U,D,V] = svd(A);
% Homography estimated by LLS with all inliers

%% Non-linear Least Square (Levenberg-Marquardt) %%
c1 = c1(inliers,:);  
c1 = c1(:);  
c2 = c2(inliers,:);  
c2 = c2(:);  
opt = optimset('Algorithm','levenberg-marquardt');  
h2 = lsqcurvefit(@fun,h1,c1,c2,[],[],opt);  
% Refined homography by L.M. 
H = [h2(1),h2(2),h2(3);h2(4),h2(5),h2(6);h2(7),h2(8),h2(9)];

Source Code for Image Mosaicking

Mosaic function:

```matlab
function [ IMG ] = mosaic( IM, im, H, x_min, y_min )
% This function maps an original image 'im' in a plane to a new image 
% 'img' in the mosaiced plane transformed by a homography H 
% Arguments: 
%        im  - single image 
%        H  - homography 
%        IM - current mosaiced image 
%       x_min - shift to x-direction 
%       y_min - shift to y-direction 
%        IMG - updated mosaiced image 

% Size of single image 
[M,N,C] = size(im);

% Size of mosaiced image 
[height,width,C] = size(IM); 
IMG = IM; 
% Assign pixel values 
x_new = [0 0 1]';  
% Homogeneous coordinate in new plane 
for m = 1 : height
    x_new(2) = m + y_min - 1;
    for n = 1 : width
        x_new(1) = n + x_min - 1;
        for c = 1 : C
            x_org = H \ x_new; 
            x = x_org(1) / x_org(3); 
            fx = x - fix(x); 
            y = x_org(2) / x_org(3); 
            fy = y - fix(y); 
            if (1 <= x && x <= N && 1 <= y && y <= M)
                % Use bilinear interpolation
```

\[
\begin{align*}
IMG(m,n,c) &= (1 - fx) \times (1 - fy) \times im(\text{fix}(y), \text{fix}(x), c) + \\
&\quad (1 - fx) \times fy \times im(\text{ceil}(y), \text{fix}(x), c) + \\
&\quad fx \times (1 - fy) \times im(\text{fix}(y), \text{ceil}(x), c) + \\
&\quad fx \times fy \times im(\text{ceil}(y), \text{ceil}(x), c);
\end{align*}
\]

end
end
end
end
end
IMG = uint8(IMG);
end

Fun function:

\[
\begin{align*}
\text{function } [ F ] &= \text{fun}( h, x ) \\
&\text{This is a user-defined function to express the homography mapping} \\
&\text{and 'fun' will be used as a parameter of the non-linear least square} \\
&\text{fitting function lsqcurvefit().} \\
&\text{Arguments :} \\
&\quad h = [h11,h12,h13,h21,h22,h23,h31,h32,h33]' \\
&\quad F = \text{Transformed coordinate by } h \text{ in the range plane} \\
&\quad x = \text{Coordinate in the domain plane} \\
L &= \text{length}(x); \quad \% \text{Length of xdata} \\
F &= \text{zeros}(L,1); \quad \% \text{Initialize function values} \\
\text{for } k = 1:2:L \\
F(k) &= (h(1) \times x(k) + h(2) \times x(k+1) + h(3)) / (h(7) \times x(k) + h(8) \times x(k+1) + h(9)); \\
F(k+1) &= (h(4) \times x(k) + h(5) \times x(k+1) + h(6)) / (h(7) \times x(k) + h(8) \times x(k+1) + h(9)); \\
\text{end}
end
\]

Main Script:

clear; close; clc;
\% Preprocess \%
\% Load single images. 
im1 = imread('apt1.jpg');
im2 = imread('apt2.jpg');
im3 = imread('apt3.jpg');
im4 = imread('apt4.jpg');
im5 = imread('apt5.jpg');
im6 = imread('apt6.jpg');
im7 = imread('apt7.jpg');
[M,N,C] = size(im2);
% Load estimated and refined homographies in previous steps.
% All the refined homographies were saved as mat files.
H12 = load('H12');   H12 = H12.H;     % Homography of im1 to im2
H23 = load('H23');   H23 = H23.H;     % Homography of im3 to im2
H34 = load('H34');   H34 = H34.H;     % Homography of im1 to im2
H54 = load('H54');   H54 = H54.H;     % Homography of im3 to im2
H65 = load('H65');   H65 = H65.H;     % Homography of im1 to im2
H76 = load('H76');   H76 = H76.H;     % Homography of im3 to im2

% im1 im2 im3 im4 im5 im6 im7 <- order of single images : im4 is in center.
H14 = H12*H23*H34;
H24 = H23*H34;
H64 = H65*H54;
H74 = H76*H65*H54;

%% Boundary Condition of Mosaiced Image %
% h14 = H14';   h14 = h14(:);           % Change homography to a vector form.
% h24 = H24';   h24 = h24(:);
% h34 = H34';   h34 = h34(:);
% h54 = H54';   h54 = h54(:);
% h64 = H64';   h64 = h64(:);
% h74 = H74';   h74 = h74(:);
% c14 = fun(h14,[1,1,N,1,1,M,N,M]);     % Transformed boundaries of im1
% c74 = fun(h74,[1,1,N,1,1,M,N,M]);     % Transformed boundaries of im7
x = [1,3,5,7];
y = [2,4,6,8];
xmin = round(min([c14(x);c74(x)]));
xmax = round(max([c14(x);c74(x)]));
ymin = round(min([c14(y);c74(y)]));
ymax = round(max([c14(y);c74(y)]));

%% Assign pixel values into the mosaiced image %%
img = zeros(ymax-ymin+1,xmax-xmin+1,C);  % Initialize mosaiced image
img = mosaic(img,im1,H14,xmin,ymin);       % Mosaicking im1
img = mosaic(img,im7,H74,xmin,ymin);       % Mosaicking im7
img = mosaic(img,im2,H24,xmin,ymin);       % Mosaicking im2
img = mosaic(img,im6,H64,xmin,ymin);       % Mosaicking im6
img = mosaic(img,im3,H34,xmin,ymin);       % Mosaicking im3
img = mosaic(img,im5,H54,xmin,ymin);       % Mosaicking im5
img(2-ymin:M+1-ymin,2-xmin:N+1-xmin,:) = im4;   % Mosaicking im4
figure; imshow(img); imwrite(img,'Mosaic_apt');