LABORATORY #4: REPORT
A.A. 2015-2016

Roberto Costa

May 5, 2016

Subject:	Image and Video Analysis
Delivery date:	05-05-2016
Professor:	P. Zanuttigh
Student number:	1128285

1 Performing image stiching to create a panoramic image

The purpose of the lab is to develop an algorithm to perform the stiching of some images, based on SIFT features extraction, to create a panoramic image.
Two different algorithms have been used: the first one (non-recoursive) takes the first two images, compute the resulting image and repeat the steps by using as imput the resulting image and the next image, until all the images have been stiched; the second algorithm (recoursive) divides the set of images into two groups and recoursively apply the algorithm to the two groups created until it gets in input a couple of images; then compute the stiching recoursively to create the overall panoramic image.
Some optimizations have been adopted, like computing the SIFT features not to the whole image, but only to the right part of the image for the left image and to the left part of the image for the right image.
The following MATLAB code has been used:

```
clear all;
close all;
defineGlobals;
global transaction_type
global algorithm_type
global output

transaction={'linear','inverse-centerDistance'};
algorithm = {'non-recoursive','recoursive'};
addOutputTitle(char(strcat('Transaction: ',transaction(transaction_type))));
addOutputTitle(char(strcat('Algorithm: ',algorithm(algorithm_type))));
im_path = getImagesPaths;

t = cputime;
if (algorithm_type==1)
    imTot = stich(im_path);
```

```
else
    imTot = stich_recoursive(im_path);
end
elapsed_A = cputime-t;
s = char(strcat('Elapsed CPU time (Exposure: manual, transaction:',...
    transaction(transaction_type),', ...
        algorithm:',algorithm(algorithm_type),...
    ')=',num2str(elapsed_A)));
disp(s);
addOutputSubTitle('lab4_tester_v02.m:');
addOutput(s);
% Cropping
addOutputTitle('Cropping');
imTotMan = crop(im_path,imTot);
% Saving
imageFile1 = char(strcat('pano_man_',algorithm(algorithm_type),'_',...
    transaction(transaction_type),'.jpg'));
addOutputSubTitle(strcat('Saving: ',imageFile1));
imwrite(imTotMan,imageFile1);
disp(char(output));
dlmcell(strcat('out_',imageFile1,'_',datestr(now,'yyyy.mm.dd_HH.MM.SS'),'.txt'),output);
```

Where the key functions are 'stich', 'stich_recoursive' and 'crop'.

```
function I = stich(im_path)
    global angle
    global out
    n = size(im_path,1);
    img1 = imread(char(im_path(1)));
    im1 = projectIC(img1,angle);
    for i=2:n
        img2 = imread(char(im_path(i)));
        im2 = projectIC(img2,angle);
        out = strcat('left image: ',toString(im_path(1:i-1)),';\\',...
            'right image: ',int2str(i),'.');
        im1 = stich2_opt(im1,i,im2,1);
    end
    I = im1;
```

```
function I = stich_recoursive(im_path)
    global angle
    global out
    n = size(im_path,1);
    m = mod(n,2);
    if ((n>=2)&&(m==0))
        im1 = stich_recoursive(im_path(1:n/2));
        im2 = stich_recoursive(im_path(n/2+1:n));
        out = strcat('left image : (',toString(im_path(1:n/2)),');\\',...
            'right image : (',toString(im_path(n/2+1:n)),').');
        I = stich2_opt(im1,n/2,im2,n/2);

    else
        if ((n>2)&&(m>0))
```

```
            im1 = stich_recoursive(im_path(1:round(n/2)));
            im2 = stich_recoursive(im_path(round(n/2)+1:n));
            out = strcat('left image : ...
                (',toString(im_path(1:round(n/2))),');\\',...
                'right image : (',toString(im_path(round(n/2)+1:n)),').');
            I = stich2_opt(im1,round(n/2),im2,round(n/2-1));
        else
            if (n==1)
                %char(im_path(n))
                img1 = imread(char(im_path(n)));
                im1 = projectIC(img1,angle);
                I = im1;
            end
        end
    end
```

```
function im = crop(imPath,imToCrop)
    global debug
    im1 =stich(imPath(size(imPath,1)));
    im2 =stich(imPath(1));
    [trX, ~] = findTraslation_opt(im1,1,im2,1);
    limit = size(imToCrop,2)-ceil((size(im1,2)-trX));
    im = imToCrop(1:size(imToCrop,1),1:limit);
    if debug
        figure;
        imshow([im(1:size(im,1),size(im,2)-200:size(im,2)),im(1:size(im,1),1:200)]);
    end
```

The function used in both algorithms 'stich' and 'stich_recoursive' is 'stich2_opt'. The MATLAB code follows:

```
function I = stich2_opt(im1,l1,im2,l2)
global debug
global transaction_type
global out
    s = strsplit(out,'\\');
    addOutputSubTitle(strcat('stich2_opt.m:'));
    addOutput(char(s(1)));
    addOutput(char(s(2)));
    [im1, im2] = sameSize(im1,im2);
    [trX, trY] = findTraslation_opt(im1,l1,im2,l2);
    overlap = size(im1,2)-trX;
    im2 = imtranslate(im2,[trX, ...
        trY],'linear','FillValues',0,'OutputView','full');
    if (trY<0)
        % add zeros at the beginning of im1
        im1 = [zeros(ceil(trY),size(im1,2));im1];
    else
        % add zeros at the end of im1
        im1 = [im1;zeros(ceil(trY),size(im1,2))];
    end
    [im1, im2] = sameSize(im1,im2);
    % now im1 and im2 has the same vertical size
```

```
    % the limits of the transition area are
    i1 = size(im1,2)-ceil(overlap);
    i2 =  size(im1,2);
    % coordinates of the center of the 2 images
    c1 = size(im1,2)/2;
    c2 = i1+(size(im2,2)-i1)/2;
    if (i1<=c1), i1 = ceil(c1+1); end
    if (i2>=c2), i2 = ceil(c2-1); end

    for i=1:i1-1
        imTot(:,i) = im1(:,i);
    end
    if debug a=[];b=[]; end
    for i=i1:i2
        if (transaction_type==1)    % simple linear join
            alpha = (i2-i)/overlap; % line through (i1,1),(i2,0)
            beta = (i-i1)/overlap; % line through (i1,0),(i2,1)
        else    % inverse center distance join
            if (transaction_type==2)
                alpha = (i1-c1)/((i-c1)*l1);
                beta = (c2-i2)/((c2-i)*l2);
            else
                alpha = 1;
                beta = 1;
            end
        end
        alpha = alpha / (alpha + beta); % normalization
        beta = 1 - alpha;
        if debug a=[a;alpha]; b=[b;beta]; end
        imTot(:,i) = im1(:,i)*alpha + im2(:,i)*beta; % weighted average
    end
    if debug
        figure;
        stem(1:size(a,1),[a,b]);
        disp(strcat('a(first)=',num2str(a(1)),';b(last)=',num2str(b(size(b,1)))));
    end
    for i = i2+1:size(im2,2)
        imTot(:,i) = im2(:,i);
    end
    I = imTot;
```

Two different algorithms have been used for fusing the two images, one is by using the average of the intensity of the pixels of the two images, weighted by a linear function, the other is by using the average of the intensity of the pixels, weighted by a function proportional to the inverse of the distance between the X coordinate of the pixel fusing and the center of the image.
The two approaches produces a linear combination where the wheight are like the ones illustrated in the following two figures:

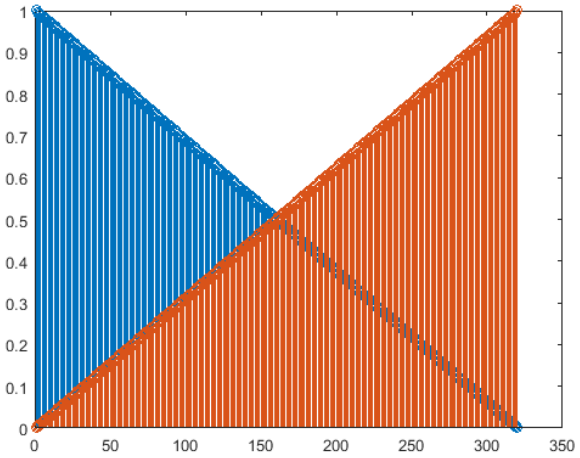

Figure 1: Example of linear blending scheme

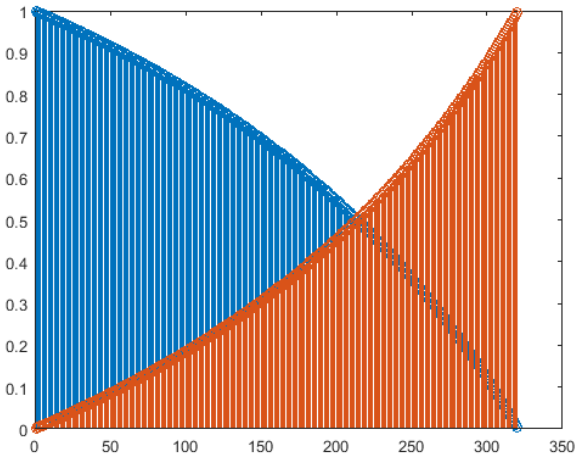

Figure 2: Example of inverse center distance blending scheme

One function used in 'stich2_opt' is 'sameSize', which takes in input two images and produces two images of the same size. The MATLAB code follows:

```
function [imA, imB] = sameSize(im1,im2)
    if (size(im1,1)>size(im2,1))
        % add zeros at the beginning of im2
        imA = im1;
        imB = [zeros(size(im1,1)-size(im2,1),size(im2,2));im2];
    else
        % add zeros at the beginning of im1
        imA = [zeros(size(im2,1)-size(im1,1),size(im1,2));im1];
        imB = im2;
    end
```

The key function used in 'stich2_opt' is 'findTraslation_opt'. The MATLAB code follows:

```
function [trX, trY] = findTraslation_opt(im1,l1,im2,l2)
    global debug
    global threshold1
    global threshold2
    global threshold3
    global threshold4
    global threshold5

    % OPTIMIZATION
    opt_index = 1;
    again = true;
    im1_original = im1;
    im2_original = im2;
    while (again)
        im1 = im1_original;
        im2 = im2_original;
        im1z1 = size(im1,2)-ceil(size(im1,2)/(2*l1))-1;
        im1z(1) = im1z1;
        im1z1 = (size(im1,2)-ceil(size(im1,2)/(l1))-1)*(l1>1);
        im1z(2) = im1z1;
        im1z1 = size(im1,2)-ceil(size(im1,2)/2)-1;
        im1z(3) = im1z1;
        im1z1=im1z(opt_index);

        im2z2 =  ceil(size(im2,2)/(2*l2))+1;
        im2z(1)=im2z2;
        im2z2 =  (ceil(size(im2,2)/(l2))+1)*(l2>1)+size(im2,2)*(l2==1);
        im2z(2)=im2z2;
        im2z2 =  ceil(size(im2,2)/2)+1;
        im2z(3)=im2z2;
        im2z2 = im2z(opt_index);

        %r = 1/2;
        r = 1/size(im1,1);

        h = ceil(size(im1,1)*r);
        % replace the useless part of the image with black
        im1 = [zeros(size(im1,1),im1z1),[zeros(h,size(im1,2)-im1z1);...
            im1(h+1:size(im1,1),im1z1+1:size(im1,2))]];
        im2 = [[zeros(h,im2z2-1);im2(h+1:size(im2,1),1:im2z2-1)],...
            zeros(size(im2,1),size(im2,2)-im2z2)];

        % END OF OPTIMIZATION
        if debug
            dt = datestr(now,'yyyy.mm.dd_HH.MM.SS');
            imageFile1 = strcat(dt,'_tmp_im1.bmp');
            imageFile2 = strcat(dt,'_tmp_im2.bmp');
        else
            imageFile1 = 'tmp_im1.bmp';
            imageFile2 = 'tmp_im2.bmp';
        end
        imwrite(im1,imageFile1);
        imwrite(im2,imageFile2);
        [im1, des1, loc1] = sift(imageFile1);
        [im2, des2, loc2] = sift(imageFile2);
```

```
56          [match, ~] = match2(im1, des1, loc1, im2, des2, loc2, threshold1);
57
58          trX = []; % traslation X
59          trY = []; % traslation Y
60          goodMatch = zeros(length(match),1);
61          for j=1:length(match)
62              if match(j)>0
63                  deltaX = loc1(j,2)-loc2(match(j),2);
64                  deltaY = loc1(j,1)-loc2(match(j),1);
65                  deltaS = abs(loc1(j,3)-loc2(match(j),3));
66                  deltaA = abs(loc1(j,4)-loc2(match(j),4));
67                  conditions = (abs(deltaY)<threshold3) ...
68                      * (abs(deltaS)<threshold5) * (deltaA<threshold4);
69                  if (conditions==1)
70                      trX=[trX;deltaX];   % store the X traslation
71                      trY=[trY;deltaY];   % store the Y traslation
72                      goodMatch(j)=match(j);
73                  else
74                      goodMatch(j)=0;
75                  end
76              end
77          end
78          if debug
79              showMatches(im1,im2,loc1,loc2,goodMatch);
80          end
81          indexes = {};
82          max_p = 0;
83          for j=1:length(trX) % for each value of X traslation
84              I = find(abs(trX(:)-trX(j))<threshold2/2); % find the traslations
85                                          % contained in the interval ...
                                                [-1.5,1.5]
86              if ((length(I)>max_p)&&(mean(trX(I))<size(im1,2)))
87                  max_p = length(I); % count them
88                  indexes = I; % save the indexes
89              end
90          end
91          if (size(trX,1)>1)
92              trX = mean(trX(indexes)); % compute the mean of the X traslation
93              trY = mean(trY(indexes)); % compute the mean of the Y traslation
94              again = false;
95          else
96              if (size(trX,1)==1)
97                  trX = trX(1);
98                  trY = trY(1);
99                  again = false;
100             else
101                 if opt_index>=3
102                     error(strcat('findTraslation_opt.m: Not enough ...
                            matching ',...
103                     'translation in findTraslation_opt.m'));
104                 else
105                     opt_index = opt_index + 1;
106                 end
107             end
108         end
109         %[trX, trY] = RANSAC(match,loc1,loc2);
```

```
110         s = strcat('Optimization index: ...
                ',int2str(opt_index-(again==1)),'); Average across (',...
111                 int2str(length(indexes)),') traslactions for RANSAC.');
112         if debug
113             disp(s);
114         end
115         addOutputSubTitle('findTraslation_opt.m:');
116         addOutput(s);
117     end
```

To find the traslation, the algorithm computes all the traslations between every couple of matching SIFT features; then for each traslation it counts how many other traslations have the X value closer than a threshold to the current traslation.
The optimization index is equal to 1 for the strongest optimization and is equal to 3 for no optimization.
The thresholds and some other useful variables have been defined by the function 'defineGlobals'. Tha MATLAB code follows:

```
1  function defineGlobals
2      global debug
3      global transaction_type
4      global algorithm_type
5      global angle
6      global path
7      global format
8      global length
9      global threshold1
10     global threshold2
11     global threshold3
12     global threshold4
13     global threshold5
14     global output
15     global out
16
17     debug = false;
18     transaction_type = 2; % transaction = ...
           {'linear','inverse-centerDistance'};
19     algorithm_type = 2; % algorithm = {'non-recoursive','recoursive'};
20
21     angle = 33;
22     path = '../images/lab_20_4_16_man/';
23     format = '.bmp';
24     length = 12;
25
26     %angle = 33;
27     %path = '../images/lab_20_4_16_man/';
28     %format = '.bmp';
29     %length = 12;
30
31     threshold1 = 0.7;
32     threshold2 = 3; % for X traslaction
33     threshold3 = 10; % for Y traslaction
34     threshold4 = 1/(2*pi); % for angle condition
```

```
35      threshold5 = 3; % for scale condition
36
37      output = {};
38      addOutputTitle('lab4_tester_v02.m');
39      addOutputTitle(strcat('out_',datestr(now,'yyyy.mm.dd_HH.MM.SS')));
40      addOutputTitle(strcat('Path:',path,';Angle:',num2str(angle)));
41      addOutputTitle(strcat('Thresholds:',...
42          num2str(threshold1),',',...
43          num2str(threshold2),',',...
44          num2str(threshold3),',',...
45          num2str(threshold4),',',...
46          num2str(threshold5)));
47      out = '';
```

The 'output' global variable of the algorithm for the images in the folder 'lab_20_4_16_man' follows:

```
1  file: out_lab_20_4_16_man_pano_recoursive_linear.jpg_2016.05.05_22.19.23.txt
2  ************************************lab4_tester_v02.m************************************
3  ***********************************out_2016.05.05_22.16.48***********************************
4  *************************Path:../images/lab_20_4_16_man/;Angle:33*************************
5  ******************************Thresholds:0.7,3,10,0.15915,3******************************
6  ************************************Transaction:linear************************************
7  ***********************************Algorithm:recoursive***********************************
8  -stich2_opt.m:
9  ------left image : (i1.bmp;);
10 ------right image : (i2.bmp;).
11 -match2.m:
12 ------Matches found=120
13 -findTraslation_opt.m:
14 ------Optimization index: (1); Average across (58) traslactions for RANSAC.
15 -stich2_opt.m:
16 ------left image : (i1.bmp;i2.bmp;);
17 ------right image : (i3.bmp;).
18 -match2.m:
19 ------Matches found=79
20 -findTraslation_opt.m:
21 ------Optimization index: (1); Average across (47) traslactions for RANSAC.
22 -stich2_opt.m:
23 ------left image : (i4.bmp;);
24 ------right image : (i5.bmp;).
25 -match2.m:
26 ------Matches found=148
27 -findTraslation_opt.m:
28 ------Optimization index: (1); Average across (67) traslactions for RANSAC.
29 -stich2_opt.m:
30 ------left image : (i4.bmp;i5.bmp;);
31 ------right image : (i6.bmp;).
32 -match2.m:
33 ------Matches found=88
34 -findTraslation_opt.m:
35 ------Optimization index: (1); Average across (74) traslactions for RANSAC.
36 -stich2_opt.m:
37 ------left image : (i1.bmp;i2.bmp;i3.bmp;);
38 ------right image : (i4.bmp;i5.bmp;i6.bmp;).
```

```
-match2.m:
-----Matches found=26
-findTraslation_opt.m:
-----Optimization index: (1); Average across (12) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i7.bmp;);
-----right image : (i8.bmp;).
-match2.m:
-----Matches found=100
-findTraslation_opt.m:
-----Optimization index: (1); Average across (42) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i7.bmp;i8.bmp;);
-----right image : (i9.bmp;).
-match2.m:
-----Matches found=144
-findTraslation_opt.m:
-----Optimization index: (1); Average across (88) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i10.bmp;);
-----right image : (i11.bmp;).
-match2.m:
-----Matches found=252
-findTraslation_opt.m:
-----Optimization index: (1); Average across (151) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i10.bmp;i11.bmp;);
-----right image : (i12.bmp;).
-match2.m:
-----Matches found=115
-findTraslation_opt.m:
-----Optimization index: (1); Average across (98) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i7.bmp;i8.bmp;i9.bmp;);
-----right image : (i10.bmp;i11.bmp;i12.bmp;).
-match2.m:
-----Matches found=35
-findTraslation_opt.m:
-----Optimization index: (1); Average across (18) traslactions for RANSAC.
-stich2_opt.m:
-----left image : (i1.bmp;i2.bmp;i3.bmp;i4.bmp;i5.bmp;i6.bmp;);
-----right image : (i7.bmp;i8.bmp;i9.bmp;i10.bmp;i11.bmp;i12.bmp;).
-match2.m:
-----Matches found=14
-findTraslation_opt.m:
-----Optimization index: (1); Average across (5) traslactions for RANSAC.
-lab4_tester_v02.m:
-----Elapsed CPU time (Exposure: manual, transaction:linear, ...
    algorithm:recoursive)=30.0156
****************************************Cropping****************************************
-match2.m:
-----Matches found=267
-findTraslation_opt.m:
-----Optimization index: (1); Average across (148) traslactions for RANSAC.
-Saving:lab_20_4_16_man_pano_recoursive_linear.jpg
```

The panoramic image produced follows:

Figure 3: Output image: algorithm type: recoursive; Fusing method: linear

The óutputǵlobal variable of the algorithm for the images in the folder 'lab_20_4_16_man' with different settings follows:

```
file: ...
    out_lab_20_4_16_man_pano_non-recoursive_linear.jpg_2016.05.05_22.22.53.txt
**********************************lab4_tester_v02.m**********************************
**********************************out_2016.05.05_22.19.25**********************************
************************Path:../images/lab_20_4_16_man/;Angle:33************************
*******************************Thresholds:0.7,3,10,0.15915,3*******************************
**********************************Transaction:linear**********************************
********************************Algorithm:non-recoursive********************************
...
-----Elapsed CPU time (Exposure: manual, transaction:linear, ...
    algorithm:non-recoursive)=31.6563
...
```

The panoramic image produced follows:

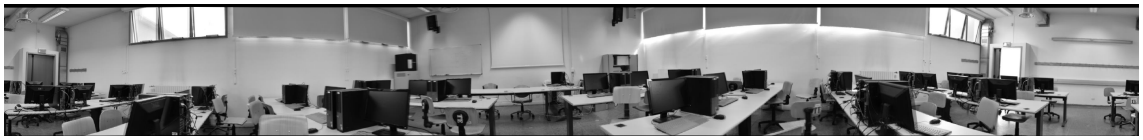

Figure 4: Output image: algorithm type: non recoursive; Fusing method: linear

The 'output' global variable of the algorithm for the images in the folder 'lab_20_4_16_man' with still different settings follows:

```
file: ...
    out_lab_20_4_16_man_pano_recoursive_inverse-centerDistance.jpg_2016.05.05_22.29.58.txt
**********************************lab4_tester_v02.m**********************************
**********************************out_2016.05.05_22.27.15**********************************
************************Path:../images/lab_20_4_16_man/;Angle:33************************
*******************************Thresholds:0.7,3,10,0.15915,3*******************************
**************************Transaction:inverse-centerDistance**************************
********************************Algorithm:recoursive********************************
...
-----Elapsed CPU time (Exposure: manual, ...
    transaction:inverse-centerDistance, algorithm:recoursive)=28.8281
...
```

The panoramic image produced follows:

Figure 5: Output image: algorithm type: recoursive; Fusing method: inverse center distance

From the above results we can see how the recoursive algorithm is faster and the difference between linear and non linear union is little.
The 'output' global variable of the algorithm for the images in the folder 'lab_20_4_16_auto' follows:

```
file: out_lab_20_4_16_auto_pano_recoursive_linear.jpg_2016.05.05_22.16.15.txt
**********************************lab4_tester_v02.m**********************************
**********************************out_2016.05.05_22.13.31**********************************
************************Path:../images/lab_20_4_16_auto/;Angle:33************************
*******************************Thresholds:0.7,3,10,0.15915,3*******************************
**********************************Transaction:linear**********************************
**********************************Algorithm:recoursive**********************************
...
------Elapsed CPU time (Exposure: manual, transaction:linear, ...
    algorithm:recoursive)=30.6563
...
```

The panoramic image produced follows:

Figure 6: Output image: algorithm type: recoursive; Fusing method: linear

The 'output' global variable of the algorithm for the images in the folder 'dolomites' follows:

```
file: out_dolomites_pano_recoursive_linear.jpg_2016.05.05_22.12.43.txt
**********************************lab4_tester_v02.m**********************************
**********************************out_2016.05.05_22.08.41**********************************
************************Path:../images/dolomites/;Angle:27.1************************
*******************************Thresholds:0.7,3,100,0.15915,3*******************************
**********************************Transaction:linear**********************************
**********************************Algorithm:recoursive**********************************
...
------Elapsed CPU time (Exposure: manual, transaction:linear, ...
    algorithm:recoursive)=72.8906
...
```

The panoramic image produced follows:

Figure 7: Output image: algorithm type: recoursive; Fusing method: linear

The 'output' global variable of the algorithm for the images in the folder 'kitchen' follows:

```
file: out_kitchen_pano_recoursive_linear.jpg_2016.05.05_19.17.45.txt
**********************************lab4_tester_v02.m**********************************
*********************************out_2016.05.05_19.10.27*********************************
*****************************Path:../images/kitchen/;Angle:22.6*****************************
*****************************Thresholds:0.6,3,5,0.15915,3*****************************
**********************************Transaction:linear**********************************
**********************************Algorithm:recoursive**********************************
...
------Elapsed CPU time (Exposure: manual, transaction:linear, ...
    algorithm:recoursive)=88.0313
...
```

The panoramic image produced follows:

Figure 8: Output image: algorithm type: recoursive; Fusing method: linear