Image morphing via triangulation

This project focuses on image morphing techniques and specifically, image morphing via triangulation. A morph is a simultaneous warp of the image shape and a cross-dissolve of the image colors. The cross-dissolve is the easy part; controlling and doing the warp is the hard part. The warp is controlled by defining a correspondence between the two pictures. The correspondence maps eyes to eyes, mouth to mouth, chin to chin, ears to ears, etc., to get the smoothest transformations possible. The correspondences are provided to you. For the triangulations, we use Delaunay triangulation (see Matlab delaunay). Recall you need to generate only one triangulation and use it on both the point sets. The triangulation should be computed at the midway shape (i.e. mean of the two point sets) to lessen the potential triangle deformations.

You need to write a function that produces a warp between your two images using point correspondences. In particular, the two input images I and J are first warped into an intermediate shape configuration controlled by `warp_frac`, and then cross-dissolved according to `dissolve_frac`. For interpolation, both parameters lie in the range $[0, 1]$. These are the two and only input parameters to the function we are asking you to code. You can download the files using this link (https://courses.edx.org/asset-v1:PennX+ROBO2x+2T2017+type@asset+block@Lab_7.1.zip).

Given a new intermediate shape, the main task is to map the image intensity in the original image to this shape. This computation should be done in reverse:

1. For each pixel in the target intermediate shape, determine which triangle it falls inside. You should use Matlab `tsearchn` for this.

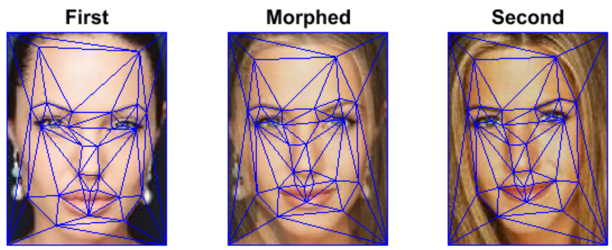
2. Compute the barycentric coordinate for each pixel in the corresponding triangle. Recall, the computation involves solving the barycentric equation:

$$\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where a, b, c are the three corners of triangle, (x, y) the pixel position, and α, β, γ are its barycentric coordinates. Note that you should only compute the matrix $\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix}$ and its inverse only once per triangle. Also, in this case, DO NOT use `tsearchn` for computing the barycentric coordinate.

3. Compute the cooresponding pixel position in the source image: using the barycentric equation, but with three corners of the same triangle in the source image $\begin{bmatrix} a^s_x & b^s_x & c^s_x \\ a^s_y & b^s_y & c^s_y \\ 1 & 1 & 1 \end{bmatrix}$, and plug in same barycentric coordinate (α, β, γ) to compute the pixel position (x^s, y^s, z^s). You need to convert the homogeneous coordinate (x^s, y^s, z^s) to pixel coordinates $(x^s/z^s, y^s/z^s)$.

4. Copy back the pixel value at (x^s, y^s) the original (source) image back to the target (intermediate) image. You should round the pixel location (x^s, y^s).



Your Function

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
function M = ImageMorphingTriangulation(warp_frac,dissolve_frac)

if nargin < 1
    warp_frac = .5;
end

if nargin < 2
    dissolve_frac= warp_frac;
end


% ream images
I = im2double(imread('a.png'));
J = im2double(imread('c.png'));

% load mat file with points, variables Ip,Jp
 load('points.mat');

% initialize ouput image (morphed)
M = zeros(size(I));

%  Triangulation (on the mean shape)
MeanShape = (1/2)*Ip+(1/2)*Jp;
TRI = delaunay(MeanShape(:,1),MeanShape(:,2));


% number of triangles
TriangleNum = size(TRI,1);

% find coordinates in images I and J
CordInI = zeros(3,3,TriangleNum);
CordInJ = zeros(3,3,TriangleNum);

for i =1:TriangleNum
  for j=1:3

    CordInI(:,j,i) = [ Ip(TRI(i,j),:)'; 1];
    CordInJ(:,j,i) = [ Jp(TRI(i,j),:)'; 1];

  end
end
```

```
end

% create new intermediate shape according to warp_frac
Mp = (1-warp_frac)*Ip+warp_frac*Jp;


% create a grid for the morphed image
[x,y] = meshgrid(1:size(M,2),1:size(M,1));

% for each element of the grid of the morphed image, find  which triangle it falls in
TM = tsearchn([Mp(:,1) Mp(:,2)],TRI,[x(:) y(:)]);


% YOUR CODE STARTS HERE


  % remove NaN :
  %in_tri = ~isnan(TM);
  %TM = TM(in_tri);
  %for num=1:length(TM);
  %  if isnan(TM(num))
  %    TM(num) = TM(num-1);
  %  end
  %end

  % get subindices
  [subI,subJ] = ind2sub(size(I), 1:size(I,1)*size(I,2));
  %subI = subI(in_tri);
  %subJ = subJ(in_tri);


  num_pix= numel(TM); % is same as m*n
  % note the factor of 3 for rgb
  IndI = zeros(1,3*num_pix);
  IndJ = zeros(1,3*num_pix);
  IndM = 1:3*num_pix;

  % create bary matrices and inverses for mp points
  bary_mat = zeros(3,3,TriangleNum);
  inv_mat = zeros(3,3,TriangleNum);
  for i = 1:TriangleNum
    current_tri = TRI(i,:);
    bary_mat(:,:,i) = [Mp(current_tri,:)'; 1 1 1];
    inv_mat(:,:,i) = inv(bary_mat(:,:,i));
  end

  % for each pixel get Bary coordinates
  for i = 1:num_pix

    % tri number
    current_tri = TM(i);
    % each pixel has a bary_matrix
    current_inv_bary_mat = inv_mat(:,:,current_tri);

    % each pixel has bary coordinates
    current_pt = [subJ(i); subI(i); 1];
    bary_coor = current_inv_bary_mat*current_pt;



    % now use bary_coor to get coor in I, J
    bary_matrix_I = CordInI(:,:,current_tri);
    pos_I = bary_matrix_I*bary_coor;
    pos_i_x = max(min(round(pos_I(1)/pos_I(3)),size(I,2)),1);
    pos_i_y = max(min(round(pos_I(2)/pos_I(3)),size(I,1)),1);
    % convert to index
    ind_I = sub2ind(size(I), pos_i_y, pos_i_x);

    % for red
```

```
    % for red
    IndI(i) = ind_I;
    % for green
    IndI(i+num_pix) = ind_I+num_pix;
    % for blue
    IndI(i+2*num_pix) = ind_I+2*num_pix;

    bary_matrix_J = CordInJ(:,:,current_tri);
    pos_J = bary_matrix_J*bary_coor;
    pos_j_x = max(min(round(pos_J(1)/pos_J(3)),size(J,2)),1);
    pos_j_y = max(min(round(pos_J(2)/pos_J(3)),size(J,1)),1);
    % convert to index
    ind_J = sub2ind(size(J), pos_j_y, pos_j_x);

    % for red
    IndJ(i) = ind_J;
    % for green
    IndJ(i+num_pix) = ind_J+num_pix;
    % for blue
    IndJ(i+2*num_pix) = ind_J+2*num_pix;


  end


% YOUR CODE ENDS HERE


  % cross-dissolve
  M(IndM)=(1-dissolve_frac)* I(IndI)+ dissolve_frac * J(IndJ);


  figure(100);
  subplot(1,3,1);
  imshow(I);
  hold on;
  triplot(TRI,Ip(:,1),Ip(:,2))
  hold off;
  title('First')

  subplot(1,3,2);
  imshow(M);
  hold on;
  triplot(TRI,Mp(:,1),Mp(:,2))
  hold off
  title('Morphed')

  subplot(1,3,3);
  imshow(J);
  hold on;
  triplot(TRI,Jp(:,1),Jp(:,2))
  hold off
  title('Second')



end
```

Code to call your function

Reset

```



```

```
warp_frac = 1/2;
```

Run Function

Previous Assessment: Incorrect

Submit

Test 1