Head Pose Changer

Loading Data

```
clear; close all;
sample_image = imread('test_images/sample.jpg');
load('test_images/sample_landmarks.mat');
face_image = imread('test_images/001.jpg');
addpath('find_face_landmarks/interfaces/matlab/')
```

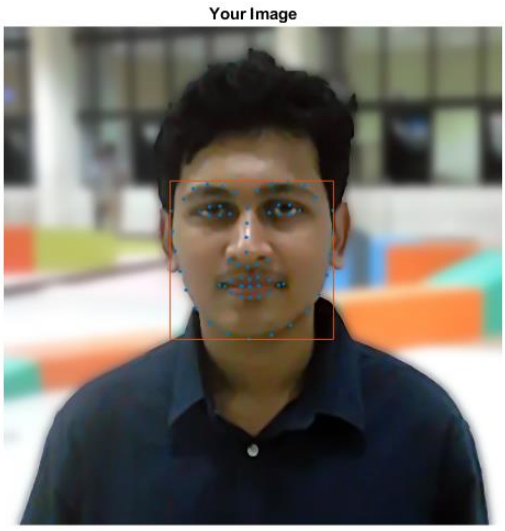
Prompt for Landmark Extraction Method

```
manual_flag = false;
answer = questdlg(['Select facial landmark extraction method. '  ...
    'Choose Manual if automatic fails to find proper landmarks'], ...
       'Landmark Extaction Method', ...
       'Automatic','Manual','Automatic');
switch answer
    case 'Automatic'
        manual_flag = false;
    case 'Manual'
        manual_flag = true;
        sample_landmarks = sample_landmarks_m;
end
```

Launching GUI for Landmark Extraction

```
f = figure('units','normalized','outerposition',[0 0 1 1]);
subplot(1, 2, 2);
imshow(sample_image); hold on;
scatter(sample_landmarks(:, 1), sample_landmarks(:, 2), '.');
if (manual_flag)
    text(double(sample_landmarks(:, 1)), double(sample_landmarks(:, 2)), ...
        num2str((1:size(sample_landmarks, 1))'), 'color', 'white');
end
title('Sample Image');
```

```
subplot(1, 2, 1);
imshow(face_image); hold on;
title('Your Image');
if (~manual_flag)
    f.Name = 'Press return if landmarks match';
    landmark_model = 'models/shape_predictor_68_face_landmarks.dat';
    frame = find_face_landmarks(landmark_model, face_image);
    landmarks = frame.faces.landmarks;
    x_min = int32(min(landmarks(:, 1))-1);
    x_max = int32(max(landmarks(:, 1))+1);
    y_min = int32(min(landmarks(:, 2))-1);
    y_max = int32(max(landmarks(:, 2))+1);
    scatter(landmarks(:, 1), landmarks(:, 2), '.');
    plot([x_min x_max x_max x_min x_min], [y_min y_min y_max y_max y_min]);
    pause;
else
    f.Name = 'Select landmarks as shown and press return';
    [landmarks(:, 1), landmarks(:, 2)] = getpts;
    f.Name = 'Press return if landmarks match';
    scatter(landmarks(:, 1), landmarks(:, 2), '.');
    text(landmarks(:, 1), landmarks(:, 2) , ...
        num2str((1:size(landmarks, 1))'), 'color', 'white');
    pause;
end
landmarks = landmarks_001;
close(f);
```



Processing Landmarks (filtering, depth assignment, triangulation)

```
landmarks = double(landmarks);
num_landmarks = size(landmarks, 1);
landmark_weights = [16, 8, 8, 8, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]';

triangulation = [   1, 2, 3;
                    1, 3, 4;
                    1, 4, 5;
                    1, 5, 2;
                    2, 16, 6;
                    3, 2, 6;
                    3, 6, 7;
                    3, 7, 8;
                    3, 8, 9;
                    3, 9, 10;
                    3, 10, 11;
                    3, 11, 4;
                    4, 11, 12;
                    4, 12, 13;
                    4, 13, 5;
                    5, 13, 14;
                    5, 14, 15;
                    5, 15, 2;
                    2, 15, 16];
num_triangles = size(triangulation, 1);

triangles.P1 = landmarks(triangulation(:, 1), :);
triangles.P1(:, 3) = 0;
triangles.P2 = landmarks(triangulation(:, 2), :);
triangles.P2(:, 3) = 0;
triangles.P3 = landmarks(triangulation(:, 3), :);
triangles.P3(:, 3) = 0;

triangles.P21 = triangles.P2 - triangles.P1;
triangles.P32 = triangles.P3 - triangles.P2;
triangles.P13 = triangles.P1 - triangles.P3;
```

Getting Input for Head Movement

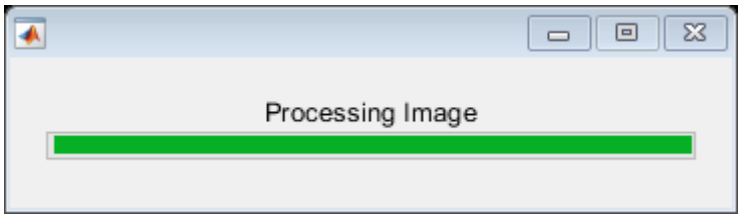
```
movement_multiplier = 2.0;
landmark_weights = landmark_weights*movement_multiplier;
f = figure;
imshow(face_image); hold on;
f.Name = 'Press arrow key to tilt head';
[~,~,button]=ginput(1);
if (button == 28)       % left
    landmarks_t(:, 1) = landmarks(:, 1) - landmark_weights;
    landmarks_t(:, 2) = landmarks(:, 2);
elseif (button == 29)   % right
    landmarks_t(:, 1) = landmarks(:, 1) + landmark_weights;
    landmarks_t(:, 2) = landmarks(:, 2);
elseif (button == 30)   % up
    landmarks_t(:, 2) = landmarks(:, 2) - landmark_weights;
    landmarks_t(:, 1) = landmarks(:, 1);
```

```
elseif (button == 31)   % down
    landmarks_t(:, 2) = landmarks(:, 2) + landmark_weights;
    landmarks_t(:, 1) = landmarks(:, 1);
end

% Same triangulation for transformed landmarks
triangles_t.P1 = landmarks_t(triangulation(:, 1), :);
triangles_t.P1(:, 3) = 0;
triangles_t.P2 = landmarks_t(triangulation(:, 2), :);
triangles_t.P2(:, 3) = 0;
triangles_t.P3 = landmarks_t(triangulation(:, 3), :);
triangles_t.P3(:, 3) = 0;
triangles_t.P21 = triangles_t.P2 - triangles_t.P1;
triangles_t.P32 = triangles_t.P3 - triangles_t.P2;
triangles_t.P13 = triangles_t.P1 - triangles_t.P3;
```

Finding polygon corresponding to each pixel and transforming it to its new location

```
[X, Y] = meshgrid(1:size(face_image, 2), 1:size(face_image, 1));
X_t = X; Y_t = Y;
x_min = int32(min(landmarks(:, 1))-1);
x_max = int32(max(landmarks(:, 1))+1);
y_min = int32(min(landmarks(:, 2))-1);
y_max = int32(max(landmarks(:, 2))+1);
t = waitbar(0, 'Processing Image');
for x = x_min:x_max
    for y = y_min:y_max
        P = repmat(double([x y 0]), num_triangles, 1);
        cpt = cross(-triangles.P13, triangles.P21);
        cpp = cross(P - triangles.P1, triangles.P21);
        s1 = sign(cpt(:, 3).*cpp(:, 3)) > 0;
        cpt = cross(-triangles.P21, triangles.P32);
        cpp = cross(P - triangles.P2, triangles.P32);
        s2 = sign(cpt(:, 3).*cpp(:, 3)) > 0;
        cpt = cross(-triangles.P32, triangles.P13);
        cpp = cross(P - triangles.P3, triangles.P13);
        s3 = sign(cpt(:, 3).*cpp(:, 3)) > 0;
        i = find(s1 & s2 & s3);
        if (~isempty(i))
            i = i(1);
            comp = pinv([triangles.P21(i, [1 2])', -triangles.P13(i, [1 2])'])*(double([x
y])' - triangles.P1(i, [1 2])');
            P_t = triangles_t.P1(i, [1 2])' + [triangles_t.P21(i, [1 2])', -
triangles_t.P13(i, [1 2])']*comp;
            X_t(y, x) = P_t(1);
            Y_t(y, x) = P_t(2);
        end
    end
    waitbar(double(x-x_min)/double(x_max-x_min), t);
end
```



Generating Output Image

```
R = face_image(:, :, 1);
R_interpolant = scatteredInterpolant(X_t(:), Y_t(:), double(R(:)));
R_t = uint8(R_interpolant(X, Y));
G = face_image(:, :, 2);
G_interpolant = scatteredInterpolant(X_t(:), Y_t(:), double(G(:)));
G_t = uint8(G_interpolant(X, Y));
B = face_image(:, :, 3);
B_interpolant = scatteredInterpolant(X_t(:), Y_t(:), double(B(:)));
B_t = uint8(B_interpolant(X, Y));
face_image_t = cat(3, R_t, G_t, B_t);
close(t);
imshow(face_image_t);
pause;
```

Visualizing results

```
close(f);
f = figure('units','normalized','outerposition',[0 0 1 1]);
subplot(1, 3, 1);
imshow(face_image);
title('Old Image');

subplot(1, 3, 2);
imshow(face_image_t);
title('New Image');

subplot(1, 3, 3);
imshowpair(face_image, face_image_t);
title('Difference');
```

Published with MATLAB® R2019a