Scale Invariant Feature Point Detection

Problem Description

Get two images, taken from the same scene but with scale transformations. Detect the scale invariant points on the two images. You can use the center of the circle to indicate the spatial position of the point and use the radius of the circle to indicate the characteristic scale of the point, just like the following example.

Development Environment

- **Operating System**: macOS Catalina 10.15.4
- **IDE**: MATLAB_R2019a
- **Programming Language**: matlab

How to Run

1. Put the following matlab script into your working path
   - `main.m`: sample driver for the project
   - `LoGDetector.m`: LoG detector for scale invariant point detection
   - `FeaturePointsVisualization.m`: Visualize for feature points detected by LoGDetector
2. Put your testing image into the folder `img/`

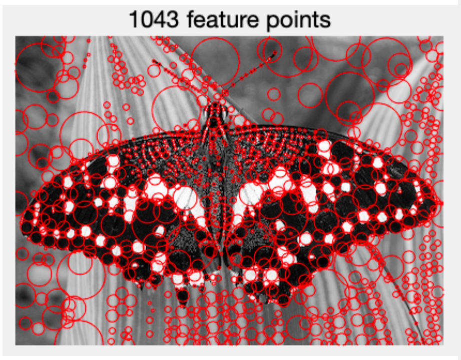
3. Change the `file_name` in `main.m`
4. Run `main.m` and waiting for output image with feature points.

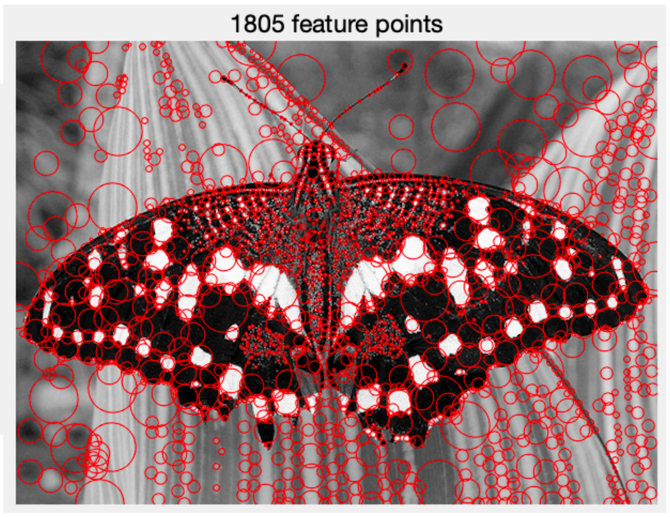
Algorithm

1. Build scale spatial space using different LoG filter.
2. Find local extremum in scale space of Laplacian of Gaussian.*(approximate)*
    - Find local maximum in image(space) scale
    - Find local maximum in sigma scale
3. Perform non-maximum suppression in scale spatial space
4. Visualize for feature points detected by LoGDetector.
    - Use the center of the circle to indicate the spatial position of the point .
    - Use the radius of the circle to indicate the characteristic scale of the point.

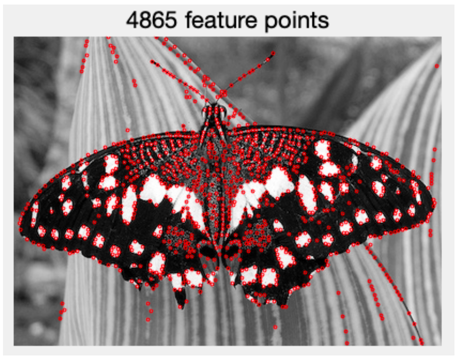
Realization

I use the image similar to the sample image, and I process the image with scale transformations. Results through my realization are showing below*(I don't spend great effort in parameter optimization, so the result cannot represent the algorithm completely)*:





> I also found something interesting, when setting the hyperparameter into sigma0=1, total_num=5, scale_ratio=1, threshold=0.005. The feature points detected by blob detector are all corner point of the object, it is well represent the edge of objects in the image. It is really interesting.



Build Scale Spatial Space

```
%% build scale spatial space using different LoG filter
scale_spatial_space = zeros(M, N, total_num);
for i = 1:total_num
    sigma = sigma0 * scale_ratio^(i-1);

    filter_size = 2 * ceil(3 * sigma) + 1;     % increase pattern size as sigma
    LoG_filter = (sigma ^ 2) * fspecial('log', filter_size, sigma);     %
create LoG filter with specific pattern size and sigma
    img_log_filtered = imfilter(img, LoG_filter, 'replicate');    % implement
LoG filter
                                                                  % replicate -
avoid false blob detection at the outer pixels

    scale_spatial_space(:, :, i) = img_log_filtered;
end
```

Find Local Maximum

```
%% find local maximum in image(space) scale
scale_spatial_space = scale_spatial_space .^ 2;      % increase the difference
max_values = zeros(M, N, total_num);
for i = 1:total_num
    max_values(:, :, i) = ordfilt2(scale_spatial_space(:, :, i), 5^2,
ones(5,5));   % ascending order for scale spatial space

% domain is 5*5

% take 5^2th as output pixel
end
```

Non-maximum Suppression

```
%% find local maximum in sigma scale
blobs = zeros(M, N, total_num);
for i = 1:M
    for j = 1:N
        blobs(i, j, :) = max(max_values(i, j, :));
    end
end
```

```
%% non-maximum suppression in scale spatial space
for i = 1:M
    for j = 1:N
        if blobs(i, j, :) < threshold
            blobs(i, j, :) = 0;
        end
    end
end
blobs = blobs .* (blobs == scale_spatial_space);
```

Record Information from (x,y,sigma) Space

```
rows = [];
cols = [];
radiuses = [];
for i=1:total_num
    [row, col] = find(blobs(:, :, i));
    rows = [rows; row];
    cols = [cols; col];
    temp_redius = sigma0 * scale_ratio.^(i-1) * sqrt(2);    % usually, the
ratio between adjacent is set to sqrt(2)
    radius = repmat(temp_redius, [size(row,1),1]);          % copy rows times
    radiuses = [radiuses; radius];
end
```

Feature Point Visualization

```
function FeaturePointsVisualization(img, rows, cols, radiuses)
% Visualize for feature points detected by LoGDetector
% use the center of the circle to indicate the spatial position of the point
% and use the radius of the circle to indicate the characteristic scale of the
point
%
% @param
% img: origin image which use to display the circle
% rows: x coordinate of feature points
% cols: y coordinate of feature points
% radiuses: characteristic scale of the point
```

```
img = rgb2gray(img);
figure;
imshow(img);
hold on;

theta = 0:pi/40:2*pi;
X = rows + sin(theta) .* radiuses;
Y = cols + cos(theta) .* radiuses;
line(Y', X', 'Color', 'r');

title(sprintf('%d feature points', size(radiuses, 1)));

end
```

About the Author

Item	Info
Name	Zhe Zhang(张喆)
ID	1754060
Adviser	Prof. Lin Zhang
Course Name	Digital Image Processing
Course Time	Mon. 2-4 [1-17]
Email	dbzdbz@tongji.edu.cn

Project Structure

```
.
├── README.md
├── doc
│   └── Report of Scale Invariant Feature Point Detection.pdf
├── img
│   ├── butterfly.jpg
│   ├── butterfly@large.jpg
│   ├── result.png
│   ├── result@large.png
│   └── result_special.png
└── src
    ├── FeaturePointsVisualization.m
    ├── LoGDetector.m
    └── main.m

3 directories, 10 files
```