MTRN4110

Laboratory Exercise

S1 2018 Week 7--10

This laboratory instruction sheet contains experiment procedures corresponding to the topic 'Image processing and computer vision'. These are Matlab based programming exercises to allow you to explore the theories behind algorithms discussed in the lecture.

You will not be required to submit a laboratory report but you have to demonstrate your work to tutors on/before the laboratory meeting in week 10. Note that this is an individual exercise.

A. Image Processing

Before processes can be made on image, they have to be read into the software, Matlab. It is also beneficial that you manage your data storage in an ordered manner. For example, you can put your codes in a directory called 'Codes', and test image in another directory called 'Images'. The following code shows how to read in an image, resize it, and display it. You can add your functions inside the 'if bn==1' segment.

```
function []=MTRN4110S12018Lab()

clc; clear all; close all; dbstop if error;

global fig; fig=[];

global img;
img.V=300; img.U=400; img.L=256; img.N=img.V*img.U;

[fn,pn,bn]=uigetfile('..//Images//*.jpg');
if bn==1,
  JPG=imread([pn fn]);
  [img,RGB]=MyImResize(img,JPG);
  fig(end+1)=MyFigure(); MyImShow(RGB);
end;

function [fig]=MyFigure()
global fig;
fig=figure('units','normalized');
set(gcf,'position',[rand*0.1+0.2 rand*0.1+0.2 0.3 0.3]);

function []=MyImShow(jmg)
global fig img;
figure(fig(end+1)); imshow(jmg); drawnow;
set(gca,'position',[0 0 1 1]);

function [jmg]=MyImResize(img,jmg)
[v,u,w]=size(jmg);
if v<u,
  jmg=imresize(jmg,[img.V img.U],'bilinear');
elseif v>u,
```

```
  jmg=imresize(jmg,[img.U img.V],'bilinear');
else
  jmg=imresize(jmg,[img.U img.U],'bilinear');
end;
[v,u,w]=size(jmg); img.V=v; img.U.u; img.N=img.V*img.U;
if max(jmg(:))>1,
  jmg=double(jmg)/img.L;
end;
```

1. *Image Enhancement*

It is required to have an output image specified by the range $a_{low} = 0$, $a_{high} = 1$. Implement Automatic Contrast Adjustment. Try out different range settings, e.g., $a_{low} = 0.3$, $a_{high} = 0.7$, and comment on the results. Hint: The function interface may look like this, `[jmg]=MyAutoContrastAdj(jmg,rng)`, where `rng` is the required intensity range.

Write another function to implement Modified Auto-Contrast. Again, try out different settings and comment on the results. You can use the same interface form as above.

Implement Histogram Equalization. Comment on the results with particular attention on the effect of colour distortion. Hint: The Matlab command `histeq` is available for this task.

Implement Histogram Specification. You have to decide on the specification, e.g. use `sin(linspace(0,1,img.L)*pi)` to generate a sine-like histogram. Comment on the choice of specified histogram on the result.

Write a function for Histogram Matching. In this task, you have to input a second image where its histogram is to be matched by the first image. Hint: The Matlab command `hist` can be used to generate a histogram.

Implement Gamma Correction. Comment on how the gamma value would change the brightness and contrast of the resultant image.

You might have encountered colour distortions in some of the enhanced results. Consider how colour distortion can be mitigated. Hint: You can try converting RGB into other colour spaces, e.g., HSV, conduct enhancement and then re-convert back to RGB for display. In Matlab, use `rgb2hsv` and `hsv2rgb`.

2. *Image Filtering*

Implement the following image filtering tasks (i) linear filter, (ii) Gaussian, (iii) difference filter. In Matlab, you can use the command `H=fspecial('gaussian',sze,sig);` to specify the filter type and parameters (read the help manual in Matlab), and the command, e.g., `jmg=imfilter(jmg,H);` to carry out filtering. When visualizing the output from the difference filter, use the command `surf(flipud(mean(filterout,3)),'edgecolor','none');` to inspect the magnitude of the output. You will see that some pixels have negative magnitudes, explain why?

Write a function to perform Median Filtering. The Matlab command `medfilt2` can be used (see Matlab help manual). Comment on the results for different filter sizes. From the output image, you will see that objects are blurred; implement a function, by using the difference

filter to restore object edges. How do you balance the smoothing effect of the Median Filter and the sharpening effect of the Difference Filter?

3. *Image Segmentation*

Implement Simple Thresholding. The goal of segmentation/thresholding is to produce a binary image that separated pixels in two classes. Hence, you can first convert a colour image into a grey image by the Matlab command `gry=rgb2gray(img)`, and then carry out thresholding.

Write a function to implement thresholding by using Otsu's method. The Matlab command `q=graythresh(gry);` can let you do this task. How this method is compared to Simple Thresholding?

Implement ISODATA thresholding. This task demands some advanced Matlab programming skills. Hint: searching for the threshold may be easier if the operation is carried out on integers, you just need to iteration the threshold through 0 to 255.

Modify the code to implement Maximum Entropy Thresholding. You can obtain the entropy by using the Matlab command `entropy(img)`. Discuss the performances of the above thresholding methods.

Implement Bernsen's and Niblack's Local Thresholding methods. Suggest how to obtain the best size of local regions. How the choices of regions size affect the resultant output?

The following function demonstrate wavelet decomposition, extraction and display of approximate, horizontal, vertical, diagonal coefficients respectively. The reconstruction process is also illustrated.

```
function [jmg]=MyWavelet1(jmg)
global fig;
gry=rgb2gray(jmg);
fig(end+1)=MyFigure(); MyImShow(gry);
W=2; ahvd={'a','h','v','d'}; wname='db5'; % wavelet name
[C,S]=wavedec2(gry,W,wname); % decomposition
for w=1:W,
  fig(end+1)=MyFigure();
  set(gcf,'position',[rand*0.1+0.2 rand*0.1+0.1 0.5 0.7]);
  for x=1:4,
    if x==1,
      % approximation coefficient
eval(sprintf('%s%d=appcoef2(C,S,''%s'',%d);',ahvd{x},w,wname,w));
    else
      % detail coefficient
eval(sprintf('%s%d=detcoef2(''%s'',C,S,%d);',ahvd{x},w,ahvd{x},w));
    end;
    % extended pseudocolor matrix scaling
    subplot(2,2,x);
eval(sprintf('image(wcodemat(%s%d,256));',ahvd{x},w));
  end;
end;
for w=1:W,
  % single-level inverse discrete 2-D wavelet transform
  eval(sprintf('gry=idwt2(a%d,h%d,v%d,d%d,''%s'');',w,w,w,w,wname));
  fig(end+1)=MyFigure(); MyImShow(gry);
end;
```

```
jmg=waverec2(C,S,wname); % multilevel 2-D wavelet reconstruction
```

Use the following code to gain an idea of the shapes of different wavelets.

```
% Set wavelet name.
fig(end+1)=MyFigure(); wname = 'db5';
% Compute the four filters associated with wavelet name given
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(wname);
subplot(221); stem(Lo_D);
title('Decomposition low-pass filter');
subplot(222); stem(Hi_D);
title('Decomposition high-pass filter');
subplot(223); stem(Lo_R);
title('Reconstruction low-pass filter');
subplot(224); stem(Hi_R);
title('Reconstruction high-pass filter');
```

B. Computer Vision

Computer vision procedures assume that the given image has been enhanced using image processing algorithms. The goal is to extract/identify objects at a higher hierarchy. However, most algorithms are too complicated for the laboratory exercise. In the following, you will be directed to observe examples in Matlab. The purpose is to let you aware of how algorithms behave with different parameter settings.

1. Object Detection

Write a function to calculate the gradient of a grey level image. The Matlab command `[Fx,Fy]=gradient(X)` returns the numerical gradient of input X. Also calculate the orientation of the gradient.

Use the Matlab command `edge()`, with difference methods, e.g., `jmg=edge(gry,'Canny');`, to extract object edges. Comment on the differences in the resultant edge maps.

The following function code demonstrates line detection using Hough transform.

```
function []=MyHoughLine(jmg)
global fig;
gry=rgb2gray(jmg); z=5; gry=gry(z:end-z,z:end-z);
edg=edge(gry,'canny');
fig(end+1)=MyFigure(); MyImShow(edg);
[H,theta,rho] = hough(edg);
fig(end+1)=MyFigure();
imshow(imadjust(mat2gray(H)),[],'XData',theta,'YData',rho,...
        'InitialMagnification','fit');
xlabel('\theta (degrees)'), ylabel('\rho');
axis on, axis normal, hold on;
P=houghpeaks(H,9,'threshold',ceil(0.1*max(H(:))));
x=theta(P(:,2)); y=rho(P(:,1));
plot(x,y,'s','color','black');
lines = houghlines(edg,theta,rho,P,'FillGap',5,'MinLength',15);
fig(end+1)=MyFigure(); imshow(gry), hold on;
set(gca,'position',[0 0 1 1]);
```

```
for k = 1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
   % Plot beginnings and ends of lines
   plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
end
```

However, the algorithm requires several parameters to be properly determined (it is in fact a very difficult problem because the ground truth is not available in real-world scenarios).

In the command, `P=houghpeaks(H,9,'threshold',ceil(0.1*max(H(:))));` try out different values in 'peak number =9' and 'threshold=ceil(0.1*max(H(:)))'. For detecting lines by the command `lines=houghlines(edg,theta,rho,P,'FillGap',5,'MinLength',15);`, try different settings for 'FillGap=5' and 'MinLength=15'.

The code below illustrates how circles are detected from a grey image (after converted from a colour image).

```
function []=MyFindCircle(jmg)
gry=rgb2gray(jmg); Rmin=5; Rmax=50;
[centers, radii, metric] = imfindcircles(gry,[Rmin Rmax]);
Thr=0.5; if min(metric)>Thr,
  C=length(radii); centersStrong = centers(1:C,:);
  radiiStrong = radii(1:C); metricStrong = metric(1:C);
  viscircles(centersStrong, radiiStrong,'EdgeColor','y');
end;
```

Again, as in the line detection case, some algorithmic parameters have to be tuned. Try different values for `Rmin`, `Rmax` and `Thr`. How they affect the detected results?

The following function illustrates how to detect corners in an image by the Harris corner detection algorithm.

```
function []=MyFindCorner(jmg)
global fig img;
gry=rgb2gray(jmg); jmg=zeros(size(gry));
Thr=graythresh(gry); jmg(gry>Thr)=1;
C=corner(jmg,'qualitylevel',0.3,'Sensitivityfactor',0.02);
fig(end+1)=MyFigure(); MyImShow(gry);
hold on; plot(C(:,1),C(:,2),'r*');
```

Determine the effects of parameters for 'qualityleel=0.3' and 'sensitivitylevel=0.02'. How they affect the results?

2. Object Tracking

Implement the Laplacian of Gaussian filter in a Matlab function. Specify kernel size and sigma (of the Gaussian) value as inputs. How is the output related to these settings? Hint: use the `imfilter` command with the following options: `'replicate'`, `'same'`, `'conv'`. You also have to use an enhancement method, Automatic Contrast Adjustment, to obtain better visualization of the result.

Write a function to implement Difference of Gaussian filtering. The approach is similar to Laplacian of Gaussian, but here you can use Matlab command `fspecial` to generate the Gaussian kernels. Also discuss the effect of kernel size and Gaussian parameter (sigma) on the result.

The following code calls the Matlab function `detectSURFFeatures` to extract SURF keypoints (if you are still using the uigetfile() structure, you do not have to select any image, when prompted by the `uigetfile` command, just select cancel). Observe results from using difference options in `MetricThreshold, NumOctaves, NumScaleLevels`. How these options affect the result?

```
function []=MyKeyPoint(jmg)
global fig img;
gry=rgb2gray(jmg);
points=detectSURFFeatures(gry,...
  'MetricThreshold',1000,'NumOctaves',3,'NumScaleLevels',4);
[features,valid_points]=extractFeatures(gry, points);
fig(end+1)=MyFigure(); MyImShow(gry); hold on;
plot(valid_points.selectStrongest(10),'showOrientation',true);
```

3. Tree-dimensional Construction

The following code (a large number of lines) demonstrates the working of stereo vision.

```
function []=MyStereo()
global fig;
% Load the stereoParameters object.
load('handshakeStereoParams.mat');
% Visualize camera extrinsics.
showExtrinsics(stereoParams);
% Create System Objects for reading and displaying the video
videoFileLeft = 'handshake_left.avi';
videoFileRight = 'handshake_right.avi';
readerLeft=vision.VideoFileReader(videoFileLeft,'VideoOutputDataType'
,'uint8');
readerRight=vision.VideoFileReader(videoFileRight,'VideoOutputDataTyp
e','uint8');
player = vision.VideoPlayer('Position', [20, 400, 850, 650]);
% Read and Rectify Video Frames
frameLeft = readerLeft.step();
frameRight = readerRight.step();
[frameLeftRect, frameRightRect] = ...
    rectifyStereoImages(frameLeft, frameRight, stereoParams);
fig(end+1)=MyFigure();
imshow(stereoAnaglyph(frameLeftRect, frameRightRect));
title('Rectified Video Frames');
% Compute Disparity
frameLeftGray  = rgb2gray(frameLeftRect);
frameRightGray = rgb2gray(frameRightRect);
disparityMap = disparity(frameLeftGray, frameRightGray);
fig(end+1)=MyFigure();
imshow(disparityMap, [0, 64]);
title('Disparity Map');
colormap jet; colorbar;
% Reconstruct the 3-D Scene
point3D = reconstructScene(disparityMap, stereoParams);
% Limit the range of Z and X for display.
z = point3D(:,:,3);
z(z < 0 | z > 8000) = NaN;
x = point3D(:,:,1);
x(x < -3000 | x > 3000) = NaN;
point3D(:,:,3) = z; point3D(:,:,1) = x;
fig(end+1)=MyFigure();
```

```
pcshow(point3D, frameLeftRect, 'VerticalAxis', 'Y', 'VerticalAxisDir',
'Down');
xlabel('X (mm)');ylabel('Y (mm)');zlabel('Z (mm)');
set(gca, 'CameraViewAngle',10, 'CameraUpVector',[0 -1 0],...
    'CameraPosition',[16500 -13852 -49597], 'DataAspectRatio',[1 1
1]);
title('Reconstructed 3-D Scene');
% Detect People in the Left Image
% Create the people detector object. Limit the minimum object size
for speed.
peopleDetector = vision.PeopleDetector('MinSize', [166 83]);
% Detect people.
bboxes = peopleDetector.step(frameLeftGray);
% Determine The Distance of Each Person to the Camera
% Find the centroids of detected people.
centroids = [round(bboxes(:, 1) + bboxes(:, 3) / 2), ...
    round(bboxes(:, 2) + bboxes(:, 4) / 2)];
% Find the 3-D world coordinates of the centroids.
centroidsIdx = sub2ind(size(disparityMap), centroids(:, 2),
centroids(:, 1));
X = point3D(:, :, 1); Y = point3D(:, :, 2); Z = point3D(:, :, 3);
centroids3D = [X(centroidsIdx)'; Y(centroidsIdx)'; Z(centroidsIdx)'];
% Find the distances from the camera in meters.
dists = sqrt(sum(centroids3D .^ 2)) / 1000;
% Display the detected people and their distances.
labels = cell(1, numel(dists));
for i = 1:numel(dists)
    labels{i} = sprintf('%0.2f meters', dists(i));
end
fig(end+1)=MyFigure();
imshow(insertObjectAnnotation(frameLeftRect, 'rectangle', bboxes,
labels));
title('Detected People');
% Process the Rest of the Video
while ~isDone(readerLeft) && ~isDone(readerRight)
    % Read the frames.
    frameLeft = readerLeft.step();
    frameRight = readerRight.step();
    % Rectify the frames.
    [frameLeftRect, frameRightRect] = ...
        rectifyStereoImages(frameLeft, frameRight, stereoParams);
    % Convert to grayscale.
    frameLeftGray  = rgb2gray(frameLeftRect);
    frameRightGray = rgb2gray(frameRightRect);
    % Compute disparity.
    disparityMap = disparity(frameLeftGray, frameRightGray);
    % Reconstruct 3-D scene.
    point3D = reconstructScene(disparityMap, stereoParams);
    % Detect people.
    bboxes = peopleDetector.step(frameLeftGray);
    if ~isempty(bboxes)
        % Find the centroids of detected people.
        centroids = [round(bboxes(:, 1) + bboxes(:, 3) / 2), ...
            round(bboxes(:, 2) + bboxes(:, 4) / 2)];
        % Find the 3-D world coordinates of the centroids.
        centroidsIdx = sub2ind(size(disparityMap), …
            centroids(:, 2), centroids(:, 1));
        X = point3D(:, :, 1);
        Y = point3D(:, :, 2);
        Z = point3D(:, :, 3);
        centroids3D = [X(centroidsIdx), Y(centroidsIdx),…
```

```
            Z(centroidsIdx)];
        % Find the distances from the camera in meters.
        dists = sqrt(sum(centroids3D .^ 2, 2)) / 1000;
        % Display the detect people and their distances.
        labels = cell(1, numel(dists));
        for i = 1:numel(dists)
            labels{i} = sprintf('%0.2f meters', dists(i));
        end
        dispFrame = insertObjectAnnotation(frameLeftRect, …
            'rectangle', bboxes, labels);
    else
        dispFrame = frameLeftRect;
    end
    % Display the frame.
    step(player, dispFrame);
end
% Clean up
reset(readerLeft);
reset(readerRight);
release(player);
```