AGH University of Science and Technology

Department of Telecommunication

Object Tracking

Introduction to image processing using Matlab

Zbigniew Hulicki **©**

1. **Theoretical fundamentals**.

Object tracking is a fundamental pre-processing step in computer vision computation and one of the essential elements of many systems and devices based on image processing. When added to e.g. object detection or the process of modelling the changes in an object's look, we can get a broad spectrum of applications, like for example:

- human-computer interaction (gesture recognition, eye gaze tracking for data input to computers);
- traffic monitoring;
- surveillance systems, which use video cameras;
- medical imaging.

Object tracking can be defined as getting the location of a particular object in subsequent frames of a video or the images in an image sequence and its purpose is to track, given the initialized state, its position during the video. The object can be anything from a point, through a geometrical figure, silhouette to more complicate shapes. Due to the complicity of the problem, its solutions have been being improved throughout many years. This complicity is influenced by for example:

- information loss, caused by transferring a three-dimensional world image to a two-dimensional computer image;
- image noise;
- changes in lighting in the image sequence;
- computational complicity of the algorithms.

Although the problem can be approached in many ways, some of the most popular ones are:

- mean shift tracking;
- contour tracking.

The first one makes use of kernels in order to find the peak of a confidence map near to the previous location of the searched object by computing the probability of a particular pixel being included in the object in the previous frame.

The second one minimizes the energy of the object, using gradient descent, and by doing so iteratively evolves an initial contour initialized from the previous frame to its new position in the current frame.

During the laboratory exercises we will use two different methods for object tracking. The first one is called frame differencing and is about comparing two subsequent images and computing differences between them, then using a threshold to distinguish noises from the

real movement. This method is especially useful in situations when the background stays still and unchanged while we are interesting in the path of all the moving objects.

The second one is background subtraction, also known as foreground detection. As its name may suggest, the method is about extracting the foreground of an image for further processing, since we are only interested in the objects in the front. As above, this method also should be applied only when the background is static. It is useful for example in processing recordings from static cameras.

The functions for simple image processing operations (including object tracking) are involved in the Image Processing Toolbox of the Matlab software. In order to ensure if the installed version of the software contains the IPT library, you should enter ***ver*** in the command window, which will result in displaying the information of the version followed by the alphabetical list of the libraries included, and then find ***Image Processing Toolbox***. In order to study the functions included in the IPT as well as their brief description, enter ***help images/contents*** and to get more specific information on their syntax and usage write ***help function_name***.

2. List of functions used in the exercise and their descriptions.

Bwareaopen()	Morphologically open binary image (remove small objects)
Diff()	Differences and approximate derivatives
Figure ()	Create figure graphics object.
Get(h,'default')	Returns all default values currently defined on object *h* in a structure array. The field names are the object property names and the field values are the corresponding property values.
Graythresh()	Global image threshold.
Imabsdiff()	Absolute difference of two images.
Imdilate()	Dilate image.
Imshow()	Displays the grayscale image.
Im2bw()	Convert image to binary image, based on threshold
Int32()	Convert to 32-bit signed integer
Length()	Length of vector or largest array dimension
Logical ()	Convert numeric values to logica
Max()	Largest elements in array
Mean ()	Average or mean value of array
Mmfileinfo ()	Information about multimedia file. As a parametr this function take a filename.
Montage()	Display multiple image frames as rectangular montage
Mov()	
Movie()	The movie function plays the movie defined by a matrix whose columns are

	movie frames.
Read ()	Read data from processor memory.
Regionprops()	Measure properties of image regions
Rgb2gray()	Convert RGB image or colormap to grayscale
Subplot()	Subplot divides the current figure into rectangular panes that are numbered rowwise
Title()	add title to current axes.
VideoReader()	Read video files. Use the VideoReader function with the *read* method to read video data from a file into the MATLAB workspace.
Zeros()	Create array of all zeros.

3. Example of object tracking in Matlab.

a. Create a new catalog “IM_TRACK” on the Desktop.

b. Copy the .avi files which are located in the same catalog as the instruction to IM_TRACK catalog.

c. Run Matlab software and do not forget to change the current catalog (above the command window) to the newly created IM_TRACK.

d. Create a new script and name it *frame_differencing*. Following the instruction and reading the comments which are included in the code, copy the subsequent fragments of the code to the script. After each part of the code, the script should be saved and run (by typing *frame_differencing* in the command window). The effects of most of the actions will be visible in subsequent windows (figures) displayed on the screen. After having studied figures 2,3, 5 and 6 please remove the lines of code which implements their creation, because these figures are interesting to be seen once but are not essential for the presentation of the functionality of the completed program.

e. The first fragment of the code presents loading a video file of an .avi type, splitting it into subsequent frames and creating data arrays and color maps to each one of them. As a result we see the original movie in figure (1) and a montage of 6 frames in figure (2).

```
%FRAME DIFFERENCING EXAMPLE 1
clc
clear all
close all

VideoReader('sun.avi');
mmfileinfo('sun.avi');
obj=VideoReader('sun.avi');
vidFrames=read(obj);

numFrames=get(obj, 'numberOfFrames');
%getting the individual frames
for k=1:numFrames
```

```
    mov(k).cdata=vidFrames(:,:,:,k);
    mov(k).colormap=[];
end

figure(1),
movie(mov,1,obj.FrameRate),title('Original
movie');
figure(2),montage(vidFrames(:,:,:,10:15));
%%end of part1------------------------------
```

f. Now we need to change all the frames to shades of gray in order to compute the differences between the subsequent ones and we will see the 10th frame in shades of grey in figure (3), the difference between frames no 1 and 2 in figure (4) and the difference between frames no 1 and 21 in figure (5).

```
%changing images from rgb to grey
for k = numFrames:-1:1
grey(:, :, k) = rgb2gray(vidFrames(:, :, :,
k));
end
figure(3),imshow(grey(:,:,10));

%computing the difference between each frame
for k = numFrames-1:-1:1
diff(:, :, k) = imabsdiff(grey(:, :, k),
grey(:, :, k+1));
end
figure(4),imshow(diff(:, :, 1), []);

%difference between 20 frames, just for
visibility - delete later
for k = numFrames-20:-1:1
diff2(:, :, k) = imabsdiff(grey(:, :, k),
grey(:, :, k+20));
end
figure(5),imshow(diff2(:, :, 1), []);
%%end of part 2-----------------------------
```

g. We use a threshold of the gray to make only the moving parts white and the noise and background black and deleting the areas that are too small to be of any interest of ours. In figure (6) we see the difference between frames no 1 and 2 in b&w and then the same image after deleting the small areas in figure (7). Examine the syntax of bwareaopen and changing its arguments no 2 and 3 try to observe the effects on figure (7).

```
%creating a black and white image in which
only the changing parts are
%white and the background is black

for k = numFrames-1:-1:1
  bw(:,:,k)=
im2bw(diff(:,:,k),graythresh(diff(:,:,k)));
end
figure(6),imshow(bw(:, :, 1));

%deleting the small area that are not
connected to the actual shape of moon
```

```
bw2 = bwareaopen(bw, 8, 8);

figure(7), imshow(bw2(:,:,1));
%%end of part 3----------------------------
```

h. The next step is calculating the assumed location of the object of interest in each frame. Then we will plot this position on two subplots in figure (8).

```
%calculating the avarage location of the
changing object in each frame
for k= numFrames-1:-1:1
   s = regionprops(logical(bw2(:,:,k)),
   'centroid');
   centroids = [s.Centroid];
   xavg = mean(centroids(:,1:2:end));
   yavg = mean(centroids(:,2:2:end));
   position(:,k)=[xavg,yavg];
end

%ploting it
figure(8), subplot (2,1,1); title('X');
plot([1:numFrames-1], position(1,:)),
ylabel('position x');
subplot (2,1,2); title('Y');
plot([1:numFrames-1], position(2,:)),
ylabel('position y');
%%end of part 4---------------------------
```

i. The final step is displaying the original movie with the assumed position of the object on it.

```
%displaying the movie with the assumed
position of the object
for k=1:length(position)
    I=mov(k).cdata;
    xpos=int32(position(1,k));
    ypos=int32(position(2,k));
    I(ypos-5:ypos+5,xpos-5:xpos+5,1:2)=255;
    mov(k).cdata=I;
end;
figure(9), movie(mov,1), title ('position');
%end of the code
```

4. Exercise to practice frame differencing.

a. Create a new script and call it *frame_differencing2*. Copy the code below, read it carefully and then follow the instruction to understand and improve its functionality.

```
%FRAME DIFFERENCING EXAMPLE 2
clc
clear all
close all

VideoReader('bb8.avi');
mmfileinfo('bb8.avi');
obj=VideoReader('bb8.avi');
```

```
vidFrames=read(obj);

numFrames=get(obj, 'numberOfFrames');
%get the individual frames
for k=1:numFrames
    mov(k).cdata=vidFrames(:,:,:,k);
    mov(k).colormap=[];
end

figure(1),
movie(mov,1,obj.FrameRate),title('Original
movie');

%changing images from rgb to grey
for k = numFrames:-1:1
grey(:, :, k) = rgb2gray(vidFrames(:, :, :,
k));
end

%computing the difference between each frame
for k = numFrames-1:-1:1
diff(:, :, k) = imabsdiff(grey(:, :, k),
grey(:, :, k+1));
end
figure(2),imshow(diff(:, :, 1), []);

for k = numFrames-1:-1:1
    bw(:,:,k)=
im2bw(diff(:,:,k),graythresh(diff(:,:,k)));
end

figure(3),imshow(bw(:, :, 1));


for k= numFrames-1:-1:1
    s = regionprops(logical(bw(:,:,k)),
'centroid');
    centroids = [s.Centroid];
    xavg = mean(centroids(:,1:2:end));
    yavg = mean(centroids(:,2:2:end));
    position(:,k)=[xavg,yavg];
end

figure(4), subplot (2,1,1); title('X');
plot([1:numFrames-1], position(1,:)),
ylabel('position x');
subplot (2,1,2); title('Y');
plot([1:numFrames-1], position(2,:)),
ylabel('position y');

for k=1:length(position)
    I=mov(k).cdata;
    xpos=int32(position(1,k));
    ypos=int32(position(2,k));
    I(ypos-5:ypos+5,xpos-5:xpos+5,1:2)=255;
    mov(k).cdata=I;
end;
figure(5), movie(mov,1), title ('position');
```

```
background = imdilate(grey,ones(1,1,10));
figure(6), imshow(background(:,:,15));
```

b. Save and run the script. It should throw an error informing us that we are trying to access elements of `I` array expressed as `ypos-5:ypos+5,xpos-5:xpos+5` that are not real numbers. In order to solve this problem we should take a closer look to the subplots at figure (4) and the object in the original movie itself. There are moments when the robot stops and is still – the subplots then have no real value. Try to fix it by editing the range of the for loop (line 57). Try to determine to which frame (X axis in figure (4)) the function is continuous and put this index as the ending condition in the loop:

```
57 - for k=1:length(position)
```

This way we will save the problem of the error but the marker will disappear after reaching the frame given in this condition.

c. Unlike the sun in the first example, we can see the marker well on BB-8. Change it by editing line 61. Make its edge's length 20 instead of current 10 and its color darker. Tip: see how it changes depending on the value you will put after the assignment operator:

```
61- I(ypos-5:ypos+5,xpos-5:xpos+5,1:2)=255;
```

d. Let us move back to the problem of the discontinuous functions in figure (4) in cases of stillness of our object (that is when the k^{th} frame is the same as $(k+1)^{th}$ and thus vidFrames(k)=vidFrames(k+1)). Since the object does not move, then its position changes or stays the same as in the previous frame – thus the function value should change or retain its previous value?
Write a conditional statement after line 43 (in the loop that fills the *position* array), remembering that it is filled 'from the top down' (`for k= numFrames-1:-1:1`).

5. Example of background subtraction.

a. Create a new script and call it *background_subtraction*. Copy the code below and read it carefully. To notice the difference between the two algorithms we will use the same movie as in the first code. Compare these two methods and decide which one works better.

```
%%BACKGROUND SUBTRACTION EXAMPLE 1
clc
clear all
close all

VideoReader('sun.avi');
mmfileinfo('sun.avi');
obj=VideoReader('sun.avi');
vidFrames=read(obj);

numFrames=get(obj, 'numberOfFrames');
%get the individual frames
```

```
for k=1:numFrames
    mov(k).cdata=vidFrames(:,:,:,k);
    mov(k).colormap=[];
end

figure(1),
movie(mov,1,obj.FrameRate),title('Original
movie');

%changing images from rgb to grey
for k = numFrames:-1:1
grey(:, :, k) = rgb2gray(vidFrames(:, :, :,
k));
end
figure(3),imshow(grey(:,:,10));

%exctracting the background
background = imdilate(grey, ones(1, 1, 5));

%other way to use the threshold
imshow(background(:,:,1));
d = imabsdiff(grey, background);
thresh = graythresh(d);
bw = (d >= thresh * 255);
centroids = zeros(numFrames, 2);
for k = 1:numFrames
s = regionprops(logical(bw(:, :, k)), 'area',
'centroid');
area_vector = [s.Area];
[tmp, idx] = max(area_vector);
centroids(k, :) = s(idx(1)).Centroid;
end
subplot(2, 1, 1)
plot(1:numFrames, centroids(:,1)),
ylabel('x')
subplot(2, 1, 2)
plot(1:numFrames, centroids(:, 2)),
ylabel('y')
xlabel('time (s)')

for k=1:length(centroids)
    I=mov(k).cdata;
    xpos=int32(centroids(k,1));
    ypos=int32(centroids(k,2));
    I(ypos-5:ypos+5,xpos-5:xpos+5,1:2)=255;
    mov(k).cdata=I;
end;
figure(4), movie(mov,1), title ('position');
```

6. Solutions.

2.b. – For example:

```
57 - for k=1:150
```

2.c. – For example:

```
61- I(ypos-10:ypos+10,xpos-
10:xpos+10,1:2)=10;
```

2.d. – For example:

```
if vidFrames(:,:,:,k)==vidFrames(:,:,:,k+1)
    position(:,k)=position(:,k+1);
end
```