****

**AGH University of Science and Technology**

**Department of Telecommunication**

HOW TO DETECT AND TRACK OBJECTS USING MATLAB

**r**

Mikołaj Leszczuk, Zbigniew Hulicki

*Introduction to image processing using MATLAB*

# Introduction

Object tracking is one of the most critical tasks for image processing and recognition. The statement We find in the picture required to determine what the object and where to find the place. The detection of items can not only be used to determine what is in the image but also to see what colour and how many of these objects are located. The purpose of this presentation is to present the algorithm and policy of detection of colours and objects and object tracking. We can get a broad spectrum of applications, like for example:

* finding lost things,
* traffic monitoring in cars,
* medical imaging,
* security system with detecting intruders.

# Image Detector

The first part of the tracking object is detection these objects in the picture chosen by the user.  
We chose a jar of jam; we took a shot of jar solo and after on the background of another object.

## Reading

First, we must read our image containing the object of interest and the scene with this object. This method of object detection works best for items that exhibit non-repeating texture patterns, which give rise to unique feature matches. This technique is not likely to work well for uniformly coloured objects; that is why we must convert to black and white. Also, our code includes changing colours so that we can use colour images.

RGB = imread('jam.jpg');

objectImage = rgb2gray(RGB);

figure(1);

imshow(objectImage);

title('Image of an object');

RGB2 = imread('background.jpg');

backgroundImage = rgb2gray(RGB2);

figure(2);

imshow(backgroundImage);

title('Image of a background');

## Finding Points

Next step is to find and show some (e.g., 100) specific points in both pictures.

objectPoints = detectSURFFeatures(objectImage);

backgroundPoints = detectSURFFeatures(backgroundImage);

figure(3);

imshow(objectImage);

title('100 Strongest Feature Points from object Image');

hold on;

plot(selectStrongest(objectPoints, 100));

figure(4);

imshow(backgroundImage);

title('300 Strongest Feature Points from background Image');

hold on;

plot(selectStrongest(backgroundPoints, 300));

## Extracting Descriptors

Extract feature descriptors at the interest points in both images.

[objectFeatures, objectPoints] = extractFeatures(objectImage, objectPoints);

[backgroundFeatures, backgroundPoints] = extractFeatures(backgroundImage, backgroundPoints);

## Finding and Displaying Matching Features

On the next step we have to find matching the features and display matched features.

objectPairs = matchFeatures(objectFeatures, backgroundFeatures);

matchedObjectPoints = objectPoints(objectPairs(:, 1), :);

matchedBackgroundPoints = backgroundPoints(objectPairs(:, 2), :);

figure(5);

showMatchedFeatures(objectImage, backgroundImage, matchedObjectPoints, ...

matchedBackgroundPoints, 'montage');

title('Putatively Matched Points with Others');

## Locating Object

Locating object in the scene using matches from point 2.5 and displaying pairs of these matches.

[tform, inlierObjectPoints, inlierBackgroundPoints] = ...

estimateGeometricTransform(matchedObjectPoints, matchedBackgroundPoints, 'affine');

figure(6);

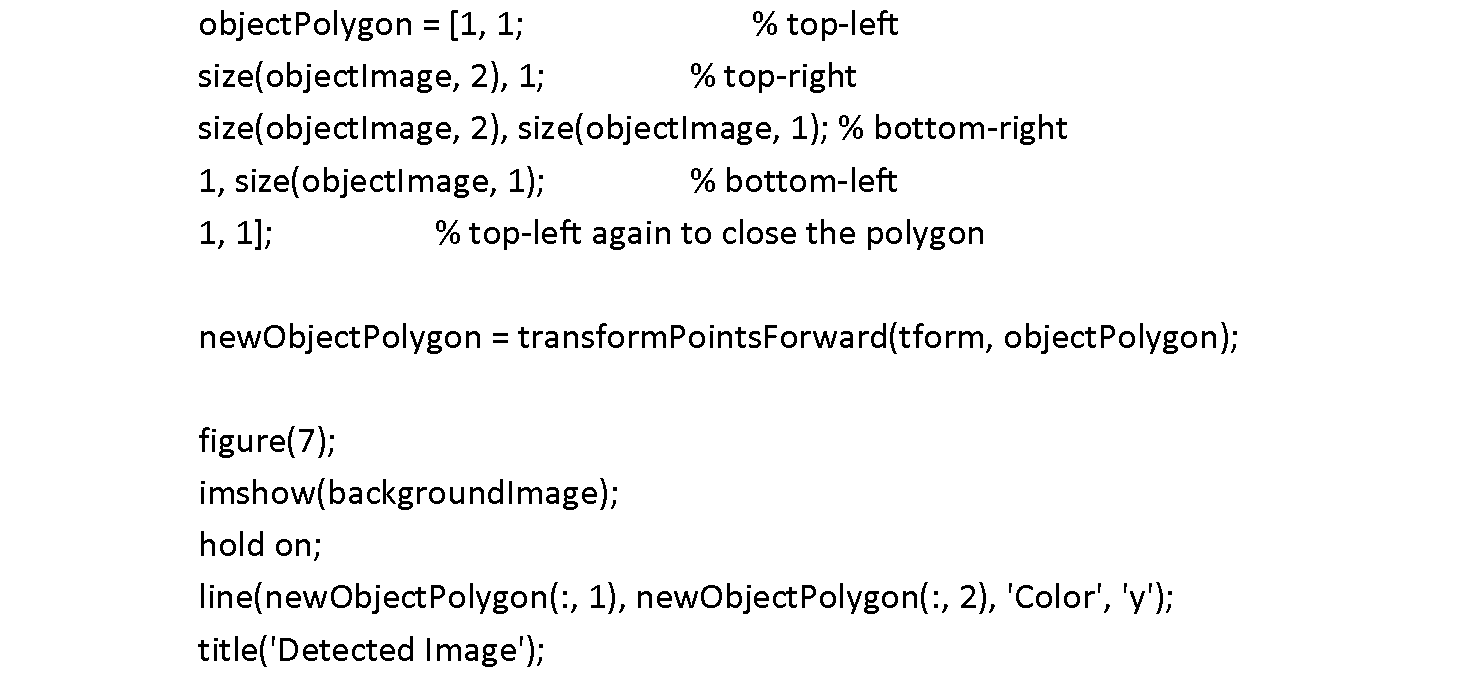
showMatchedFeatures(objectImage, backgroundImage, inlierObjectPoints, ...

inlierBackgroundPoints, 'montage');

title('Matched Points without Others');

## Showing Polygon

The last step shows a polygon of the reference image and display.



## Figures

Our figures:

|  |  |  |  |
| --- | --- | --- | --- |
| 1.png | | | 2.png |
| **Figure 1** | | | **Figure 2** |
|  |  |  | |
|  | **3.png**  **Figure 3** | **4.png**  **Figure 4** | |
|  | 5.png **Figure 5**  6.png **Figure 6**  7.png **Figure 7** | | |

# Colour Detection

**Warning! Several people have reported a problem that they cannot perform the exercises presented in Section 3 of the instructions. It is because MATLAB they installed does not have the appropriate operating system adapters that are needed to work with the "video input" function. Here are more details on this topic from MathWorks:**

[**https://www.mathworks.com/help/imaq/installing-the-support-packages-for-image-acquisition-toolbox-adaptors.html**](https://www.mathworks.com/help/imaq/installing-the-support-packages-for-image-acquisition-toolbox-adaptors.html)

**The problem arises from the MATLAB installation architecture.**

**If the problem persists, please report it to the tutor and move on to performing further exercise points.**

Many objects have customized shapes such as the face. Human face shape is very diverse, in addition to the face can always occur moustache, glasses or hat. It is some impediment to the detector face. Therefore, face detection we need colour. When detecting objects based on the colour, it is necessary to specify the colour precisely and the scope to be covered. Detection based on colour is not limited to the face. An example of this may be a notebook, which is red. By which can be easily detected in the movie.

In everyday use object detection using colour is not very useful. Because of the shade and the colour depending on the lighting, which is different in the day (e.g., the sun), the other at night (e.g., the moon or artificial light) by which colour of light is varied. If the colour range we give too little, and when the lights change significantly, we cannot detect the object.

## Camera

Get your computer camera, set parameters, and start it.

vid = videoinput('winvideo',1,'YUY2\_320x240');

%properties

set(vid, 'FramesPerTrigger', Inf);

set(vid, 'ReturnedColorspace', 'rgb');

vid.FrameGrabInterval = 5;

start(vid)

## Frames

To get frames from the camera, you must put this into the Main loop.

while(vid.FramesAcquired<=200)  %stop after 100 frames

      data = getsnapshot(vid); %snapshot of the current frame

## Track

Now to track red objects in real-time, we must subtract the red component from the grayscale image to extract the red elements in the picture.

diff\_im = imsubtract(data(:,:,1), rgb2gray(data));

If you want to detect other colours than red, you can set another parameter, e.g.:

* + - data(:,:,2) - green
    - data(:,:,3) - blue

## Filter

Median filter to filter out noise, convert the resulting grayscale image into a binary image and remove all those pixels less than 300px.

diff\_im = medfilt2(diff\_im, [3 3]);

diff\_im = im2bw(diff\_im,0.18);

diff\_im = bwareaopen(diff\_im,300);

## Label

Label all the connected components in the image and get a set of properties for each labelled region.

 bw = bwlabel(diff\_im, 8);

 stats = regionprops(bw, 'BoundingBox', 'Centroid');

## Display

Display as a coloured image or as black and white in which only the colour parts are white, and the background is black.

  imshow(data); %color

  imshow(diff\_im); %black and white

  hold on

## Loop

It is a loop to bound the red objects in a rectangular box.

    for object = 1:length(stats)

        bb = stats(object).BoundingBox;

        bc = stats(object).Centroid;

        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)

        plot(bc(1),bc(2), '-m+')

a=text(bc(1)+15,bc(2), strcat('X:', num2str(round(bc(1))), 'Y:', num2str(round(bc(2)))));

set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');

end

hold off

We also display position and set properties of the bound.

* 1. At this point, we end the Main loop, stop and clear data to release memory.

stop(vid);

flushdata(vid);

# Object Tracking

Detection of moving objects is done by comparing each frame with the stored background. The background is taken automatically when the image processing and updated to be running changes in the field of view of the camera. When comparing each frame with the background image is divided into background and foreground. Then, foreground pixels are combined into groups and give the position and dimensions of the moving objects in the picture. At the end of these objects, they are classified and tracked.

We can define object tracking as getting the location of a particular item 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 complicated shapes. Due to the collaboration of the problem, its solutions have been being improved throughout many years. This collaboration 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.

## 4.1 Video

Get video from the file.  
  
mmfileinfo('Snow.avi');

obj=VideoReader('Snow.avi');

vidFrames=read(obj);

## 4.2 Frames

Get the individual frames from the movie and display the original film.  
  
numFrames=get(obj, 'numberOfFrames');

for k=1:numFrames

mov(k).cdata=vidFrames(:,:,:,k);

mov(k).colormap=[];

end

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

## 4.3 Transform

Now we transform image frames from RGB to grey.

for k = numFrames:-1:1

grey(:, :, k) = rgb2gray(vidFrames(:, :, :, k));

end

## 4.4 Extract

Extract background from the video

background = imdilate(grey, ones(1, 1, 5));

## 4.5 Find

Find path of the moving object and fill data of it into the matrix.

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

## 4.6 Show

Show positions X and Y of the object on the graphs depending on time.

figure(3);

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)')

## 4.7 Display

Display original movie with the following dot after moving object.

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');

1. **Bibliography**

* <https://en.wikipedia.org/wiki/Background_subtraction>
* <https://en.wikipedia.org/wiki/Video_tracking>
* <https://en.wikipedia.org/wiki/Background_subtraction>
* <https://en.wikipedia.org/wiki/Video_tracking>
* <http://www.kkiem.agh.edu.pl/dydakt/obrazy/Matlab_obrazy.pdf>
* <https://uk.mathworks.com/company/newsletters/articles/tracking-objects-acquiring-and-analyzing-image-sequences-in-matlab.html>
* <http://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html>
* <https://www.mathworks.com/help/matlab/ref/rgb2gray.html>