DOI: 10.2507/28th.daaam.proceedings.163

OBJECT RECOGNITION IN IMAGES

Adámek Milan, Neumann Petr, Pospíšilík Martin,
Španko Adrian, Vašek Vladimír

This Publication has to be referred as: Adamek, M[ilan]; Neumann, P[etr]; Pospisilik, M[artin]; Spanko, A[drian] & Vasek, V[ladimir] (2017). Object Recognition in Images, Proceedings of the 28th DAAAM International Symposium, pp.1178-1184, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/28th.daaam.proceedings.163

Abstract

Perception of image perceptions is one of the most important features of every person. With the help of our eyesight, we can perceive the world around us, acquiring information about our surroundings and recognize different objects in them. Thanks to the recipes of our eye, we can capture it clear information and transferred information to the brain where this information is further processed and evaluated. Based on that, we can recognize what we see.The detection of various objects or shapes in an image is a very important element of all operations involving Image Processing. It is not only used in the processing of Medical Imagery data, (e.g. searching for structures in tomographic images); but also, in the Security field, (e.g. access control); as well as in the human sector – i.e. in cameras and cameras, (e.g. face detection, smiles, etc.). This article deals with possible detection methods of objects in images.

Keywords: image processing; neural network; Matlab; pre-processing; training data

1. Introduction

Computer Vision Systems are used in many areas of human activities. In the Security Technology field, it is possible to detect objects; or, eventually, even to identify individuals, as captured by a security camera. In most cases, this often involves the detection of faces, their physical build, or the identification of a person by means of studying and comparing their physio-motoric mannerisms and manifestations. In the Industrial Production field, one can – with the assistance of video-recording, to assure product quality control; video-recordings are very often components of Robot Control Systems.

Prior to detecting objects in a recorded image, the "captured record" must first be converted from an analogue to a digital form. Before the actual segmentation of the image into sub-segments - the recorded image is very often adjusted in such a way as to make it possible to better search for the object, (that is), being searched for. This adjustment very often includes geometric transformations, brightness transformations, filtration, and focusing.

2. Image Segmentation

Image Segmentation is a set of processes, the course of which leads to the subdivision of the image into its sub-components, which - at least to a certain extent, are related to reality. Mutual Indivisibility is a requisite for all areas of an area in a given set. The degree of matching is sought between the segments and the actual objects that are discovered. Depending on their degree of accord, the segmentation can be further sub-divided into Complete and Partial. It is logical that Partial Segmentation has a lower degree of consistency, unlike Complete Segmentation [1].

An image is obtained in the course of Partial Segmentation, which is divided into several independent parts. The greatest advantage of this type of segmentation is the ability to elaborate even complex scenes; and further, to reduce the volume of the processed data. The result of this type of segmentation is a set of homogeneous regions, which have a certain colour and brightness. The fundamental disadvantage of this type of segmentation is the need to use subsequent steps, which help to acquire the relevant results [1, 4].

Image Segmentation can be broken-down, according to principles used, into three basic methods. The first, is based on an overall general knowledge of images, or their parts. This method is represented with the aid of histograms. The second, exploits algorithms that specify the algorithms between objects. The third, is based on the algorithms that these areas create [3, 6].

2.1. Object Description

It is possible to describe an object in an image in two ways. The first, is based on the Quantitative approach - this approach uses a set of numerical characteristics. As a rule, under the term “set of numerical characteristics”, it is possible to consider and evaluate the object size, colour scattering. or comparability. The second, is to describe the object using a Qualitative approach. This approach to description is based on the description of the relationships between selected objects; and describes their shape properties and characteristics. When comparing and recognising objects, these descriptions are considered as input information [2, 5].

2.2. Classification

The task of classification is to include an object found in the image into a group of known classes. The Classification methods are divided into two groups; Symptomatic and Structural. Symmetric Classification, uses groups of characteristic object numbers, which describe its properties; like, for instance, location or size. Quantitative Description of the object is used for the description of the object. Cluster Analysis can also be considered as a form of Symptom Analysis. This analysis classifies objects into cluster groups in such a way as that objects in a group have identical, or very similar properties/characteristics.

Structural Classification works on the basis of the assignment of certain properties that are characteristic for the given object. This type of classification works with the Qualitative Description of the given object. The properties of the object are, through the intermediary of algorithms subjected to by word-breakdown processing, which describe the object - and subsequent to control, the language, grammar, and alphabet can be defined [7, 8].

3. A Programme Application Designed to Search Objects in Images

The Matlab programme environment was used for the creation of a programme designed for searching for an object in an image. The following Image/Fig. (1), shows how to create a GUI in Matlab.

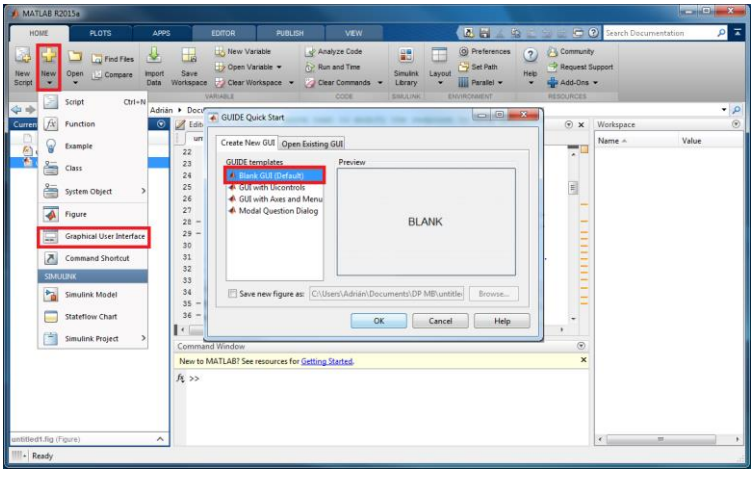

Fig. 1. Creating a graphic user environment in the Matlab programme

4. Programme Application Functions

4.1. Loading/Reading Images

The recorded image is loaded/read in this application by using the button, which uses the loadBotton_Callback function. It is necessary to define the place where the image is to be opened; on a standard basis, in a new window. The performance of the above steps is accomplished through the intermediary of the following function:

```
function loadButton_Callback ( hObject , eventdata , handles )
[ filename , pathname ] = uigetfile({'*.bmp' ; '*.jpg' ; '*.gif ' ; ' *.*'},
'Vyber obrázok ');
S~= imread ( [ pathname , filename ] ) ;
uiwait (msgbox ( ' Trénovacie dáta načítané ! ' , 'Dáta načítané ! ' ) ) ;
axes ( handles . mainAxes ) ;
imshow(S ) ;
handles.S~= S ;
set ( handles.vyberButton , 'enable' , ' on ' ) ;
guidata ( hObject , handles ) ;
```

The following function can be used to record/capture images:

```
kamButton_Callback:
 function kamButton_Callback ( hObject , eventdata , handles )
vid = videoinput ( ' winvideo ' , 1 , 'RGB24 640x480 ' ) ;
axes ( handles . mainAxes ) ;
vidRes = get ( vid, ' VideoResolution ' ) ;
nBands = get ( vid, 'NumberOfBands ' ) ;
hImage = image( zeros ( vidRes ( 2 ) , vidRes ( 1 ) , nBands ) ) ;
```

4.2. Selecting and Cropping Images

The Object Selection function initiates the process of selecting an object in an image, in such a way as one would know what object to search for. This function can also be used in the case where there are multiple objects in the image. The Display a Selection Rectangle function serves to indicate the object. The following functions can be used for selection purposes:

```
function vyberButton_Callback ( hObject , eventdata , handles )
S~= handles.S ;
axes ( handles.mainAxes ) ;
if isfield ( handles , 'api ' )
handles . api . delete ( ) ;
rmf ield ( handles , ' api ' ) ;
rmf ield ( handles , ' hRect ' ) ;
axes ( handles.mainAxes ) ;
imshow(S ) ;
end
axes ( handles . mainAxes ) ;
s z = size (S ) ;
handles.hRect = imrect (gca , [ round( sz ( 2 ) / 2 ) round( sz ( 1 ) / 2 ) 50 50 ] ) ;
handles.api = iptgetapi ( handles . hRect ) ;
set ( handles.orezButton , ' enable ' , ' on ' ) ;
guidata ( hObject , handles ) ;
```

4.3. Pre-processing

Another function is the Pre-processing feature, i.e. Image Pre-processing. After starting it, the Boot_Callback command is called. After recording and loading the cropped image, this is converted into shades of grey. Thereby, colour tones and saturation are removed; but image brightness is conserved/retained. This is followed by the “thresholding” of the image using the graythresh and im2bw commands. This approach encapsulates the following features:

```
function pripravaButton_Callback ( hObject , eventdata , handles )
img crop = handles.img crop ;
imgGray = rgb2gray ( img_crop ) ;
prah=graythresh ( imgGray ) ;
bw = im2bw( img_crop,prah ) ;
```

This is followed by an automatic cropping feature, which crops the object right up to its limits in such a way that there are no faint surfaces on it. This is accomplished by the following script:

```
 bw2 = auto_orez (bw) ;
axes ( handles.mainAxes ) ;
imshow(bw2 ) ;
handles.bw2 = bw2 ;
```

This is followed by cycles for detecting white spots in the image. The cycles function on the principle of the sum of the values of the elements in the matrix columns. The white space search cycles are as follows: The following are the cycles for detecting the white spots in the image. Cycles work on the sum of values of the elements in the matrix columns. The white gap space search cycles are as follows:

```
pocB=1;
while (sum(bw( : , pocB))==y2pom)
x1=x1+1;
pocB=pocB+1;
end
pocB=1;
while (sum(bw(pocB, : ) )==x2pom)
y1=y1+1;
pocB=pocB+1;
end
pocB=x2pom;
while (sum(bw( : , pocB))==y2pom)
x2=x2-1;
pocB=pocB-1;
end
pocB=y2pom;
while (sum(bw(pocB, : ) )==x2pom)
y2=y2-1;
pocB=pocB-1;
end
```

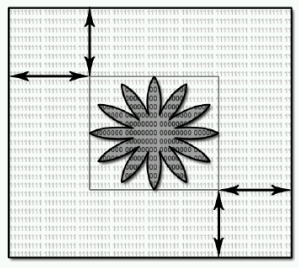

Fig. 2. Searching for White Spaces and the Creation of an Automatic Clipping of Sectors.

Subsequently, using the imcrop command, the final version of the crop is realised, and the image is the subsequent output of the given script. The cropping function is as follows:

```
bw2=imcrop (bw, [ x1 , y1 , ( x2-x1 ) , ( y2-y1 ) ] )
```

The NNdata variable creates a matrix of an image, which must be the same size for Neural Network Training purposes. When inserting the image into a database, the image is rotated by 360 °; and the rotations take place every 45°.

Each rotation is saved as a new image, the reason is for improving the learning of the neural network purposes. The whole function appears as follows:

```
if get ( handles.radiotren , ' value ' ) == 1
for i ~= 1 : 8
imshow(bw2)
pause ( 1 ) ;
handles.NNdata ( : , : , handles.pocobj )=imresize (bw2 , [ 80 , 80 ] ) ;
bw2 = (bw2 == 0 ) ;
bw2=imrotate (bw2 , 4 5 , ' bilinear ' ) ; bw2 = (bw2 == 0 ) ;
bw2 = auto_orez (bw2 ) ;
handles.pocobj=handles.pocobj+1;
end
else
handles.NNdata ( : , : , handles.cnt )=imresize (bw2 , [ 80 , 80 ] ) ;
end
handles.cnt=handles.cnt+1;
```

4.4. Training Neuron Networks

After processing and inserting the objects into a database, one can begin to train a neural network. The *treningButton_Callback* function can be used to train a neural network:

```
function treningButton_Callback ( hObject , eventdata , handles )
hasField = isfield ( handles , 'NNdata ' )
if hasField
NNdata = handles.NNdata ;
[ NNinput NNtarget ]=traindata (NNdata , handles.cnt , handles.pocobj ) ;
```

The "traindata" script contains two parameters at input; the first is the NNdata variable, (which contains the matrix of embedded images) - and the other is the total number of objects in the database. The script is composed of three cycles. The first distinguishes individual image matrices; the second determines the processing of the lines; and the third controls the processing of the columns in the given row. In essence, this functions by moving the whole matrix, sequentially, into blocks - and counting all of the values in each block. The dimensions of the matrix determine the total number of learning objects, (i.e. number of columns), and number of object types, i.e. (number of rows). Processing the matrix block by block, is shown in the following figure.

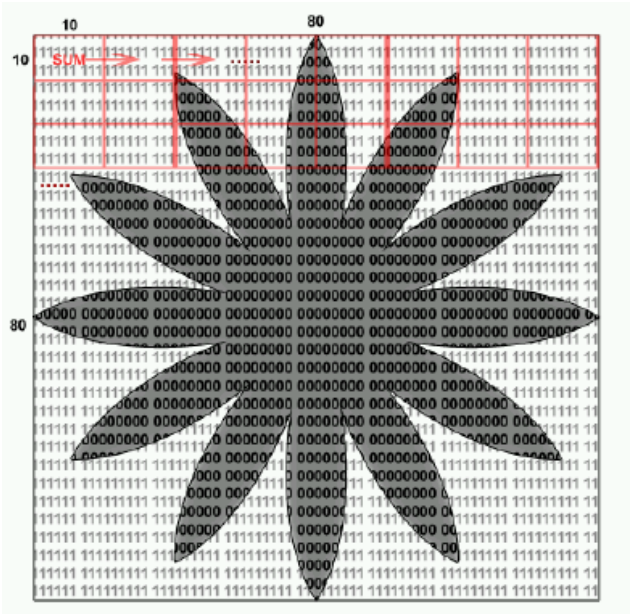

Fig. 3. Processing the matrix block by block [8]

The script entitled "*createnn*" creates the neural network and trains it in using the input data:

```
[ net , tr ] = creatnn (NNinput , NNtarget , handles.cnt )
handles . net=net ;
```

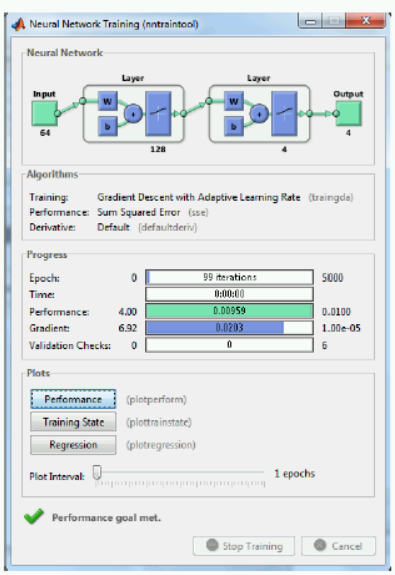

Fig. 4. The Neural Network Training Tool

5. Object Recognition

When using the designed GUI, one needs to first select the Recognition Mode. The "*radiotest_Callback*" function will then start. The function verifies whether there are NNdata and net variables, i.e. whether there are database objects and a neural network. Recognition Mode. Here is the function for switching over into the object recognition regime:

```
function radiotest_Callback ( hObject , eventdata , handles )
set ( handles.radiotest , ' value ' , 1 ) ;
set ( handles.radiotren , ' value ' , 0 ) ;
set ( handles.radiotren , ' enable ' , ' off ' ) ;
ha sField = isfield ( handles , 'NNdata ' )
hasField2 = isfield ( handles , ' net ' )
i f hasField && hasField2
handles = rmf ield ( handles , 'NNdata ' )
handles.cnt=1;
set ( handles.treningButton , ' enable ' , ' off ' ) ;
set ( handles.ulozbutton , ' enable ' , ' off ' ) ;
set ( handles.nacitajbutton , ' enable ' , ' off ' ) ;
else
warningMessage = sprintf ( ' Varovanie : Rozpoznávací režim je neprístupný,
najskôr uskutočni tréning siete. ' ) ;
uiwait ( warndlg ( warningMessage ) ) ;
set ( handles.radiotren , ' enable ' , ' on ' ) ;
```

6. Saving and Retrieving Training Data

Data from the training set can be saved using the Save function, which is executed by running the "*ulozButton_Callback*" function:

```
function ulozButton_Callback ( hObject , eventdata , handles )
state .NNdata= getfield ( handles , 'NNdata ' )
state. cnt= getfield ( handles , ' cnt ' )
state. pocobj= getfield ( handles , ' pocobj ' )
state. menaObj= getfield ( handles , ' menaObj ' )
[ filename , pathname ] = uiputfile ({' *.mat ' }, 'Vyber objekty ' ) ;
save ( [ pathname , filename ] , ' state ' )
uiwait (msgbox ( ' Trénovacie dáta sú uložené ! ' , 'Dáta uložené ! ' ) ) ;
guidata ( hObject , handles ) ;
```

In order to retrieve data from the training set, the Data Retrieval function was designed – which uses the following function:

```
function nacitajButton_Callback ( hObject , eventdata , handles )
[ filename , pathname ] = uigetfile ({ ' *.mat ' } , 'Vyber objekty ' ) ;
load ( [ pathname , filename ] , ' state ' )
uiwait (msgbox ( ' Trénovacie dáta sú načítané ! ' , 'Dáta načítané ! ' ) ) ;
handles .NNdata=state .NNdata ;
```

7. Creating a Standalone Executable Application

You can use the "mcc" command to create a standalone executable application. Parameter "m" in the "mcc" command creates a standalone executable application from the "main.m" file.

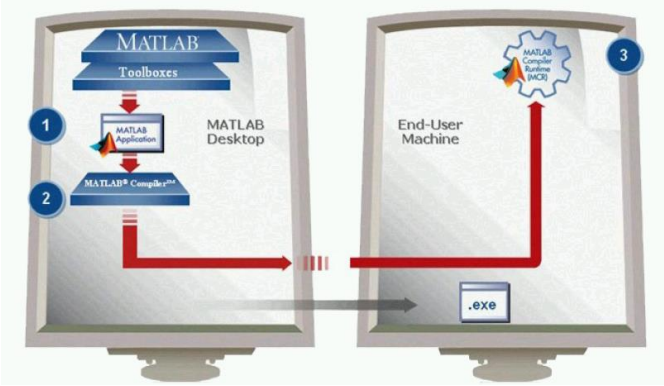

Fig. 5. Schema of the creation of a standalone executable application [8]

8. Conclusion

In the course of creating a programme, it is also necessary to take possible error events into account. These may be caused by incorrect handling of the programme application. The majority of states that could lead to the origin of error events are resolved by means of a GUI that does not allow some buttons to run in the programme. Sin view of the fact that it is not possible to resolve all the situations by means of the buttons in the GUI, the programme application also has error messages. An example of this is, for example, an error message that appears in the case where a user starts the recognition mode without prior training of the neural network.

9. Acknowledgments

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme project No. LO1303 (MSMT-7778/2014) and also by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

10. References

[1] Gonzales, R.; Woods, R. (2002). Digital Image Processing. 2nd edition. Upper Saddle River, New Jersey 07458 : Prentice-Hall, 2002. 793 p. ISBN 0-20-118075-8
[2] Šonka, M,; Hlaváč, V. & Boyle, R. (2008). Image processing, analysis, and machine vision. 3rd ed.. Toronto: Thomson. 829 p. ISBN 978- 0-495-08252-1.
[3] Parker J. R.(2011) Algorithms for Image Processing and Computer Vision. Wiley Publishing, Indianapolis. ISBN 978–0–470–64385–3.
[4] Dobeš, M. (2008). Image processing and algorithms in C#. Prague. BEN. 144 p. ISBN 978-80-7300-233-6.
[5] Hlaváč, V. & Sedláček, M. (2005). Processing of signals and images. Prague. 255 p. ISBN 80-010-3110-1.
[6] Říha, K.(2007). Advanced image processing techniques. Brno. 109 p.
[7] Karban, P. (2006). Calculations and simulations in Matlab and Simulink. Brno, Computer Press. 220 p. ISBN 80-251-1301-9.
[8] Žára, J. (2004). Modern computer graphics. Praha. Computer Press. p. 542-546. ISBN 80-251-0454-0.