ATLANTIS PRESS

Advances in Computer Science Research, volume 93

2nd International Conference on Mathematics, Modeling and Simulation Technologies and Applications (MMSTA 2019)

Software for Detection of Objects while Video Sequences Handling

Iakov Korovin[1], Maxim Khisamutdinov[2] and Donat Ivanov[1,*]

[1]Southern Federal University, Rostov region, Taganrog, Chekhov str., 2, GSP-284., 347900, Russia

[2]"Neuronetwork technologies" Limited, Rostov region, Taganrog, Socialisticheskaya str., 150-g., 347900, Russia

***Abstract*—In this paper we describe a software implementation of computer vision methods for highlighting contrasting objects in noisy images. A user-friendly graphical interface is developed. The software program is intended for reading video sequences files, their processing in order to seek and recognize objects, present in a frame. After data handling the discovered objects' characteristics are stored in XML format files.**

Keywords—detection; blobs detector; image stabilization; optical flow; software

I. INTRODUCTION

One of the important tasks of machine vision is the search, recognition and tracking of objects in video images [1]. Currently, there are quite effective methods for detecting objects on a simple background. What for a complex background - is the task of detecting objects is considered to be vital. This task is complicated by the fact that images, obtained from video sensors, as a rule, are significantly noisy [2]. Therefore, it is necessary to get rid of noise before performing the procedure of detecting objects in a video sequence.

This paper shows the results of software implementation of the developed methods for highlighting contrasting objects in noisy images obtained from video sequences. The program is intended for reading files of video sequences, their processing in order to detect objects, present in the frame.

II. METHODS AND ALGORITHMS APPLIED

The basic software algorithm consists of four main stages:

- Input video input sequence
- Selection of detector parameters
- Video sequence processing
- Saving results

The program uses computer vision methods to highlight contrasting objects (blobs [3]–[7]) in the image. Consider an enlarged scheme for finding objects on a complex background. The input data for searching for objects on a complex background are *N* adjacent frames of the input video sequence - *[frame_0 .. frame_N]*. At the initial stage of the algorithm, the vector *vObj* is initialized, which will be the desired result of the found objects. Next, all input frames are processed, starting from the first. To do this, blobs are selected in each *frame[i*] frame and further analyzed handling their centers and dimensions. When intersecting with existing objects in the *vObj* vector, the object parameters are updated, if no intersection is found, a new object is added to the *vObj* vector with the found blob parameters.

Thus, when processing all *N* frames of the video sequence, a vector will be obtained, containing the characteristics of the present objects *vObj* of the input video sequence.

III. SOFTWARE STRUCTURE

The software consists of three main parts:

- Navigator for the video file;
- Blob detector;
- XML logging module.

The video navigator is used to view the video file. Getting started with the navigator is carried out after opening the video file using the "Open AVI" button. After opening the video file, the user can see the first frame of the video sequence and the total number of frames (1095).

The user can navigate through the video using the slider at the top of the screen, or frame by frame with the buttons "Prev. f "and" Next f ". To start viewing the video sequence, you must click the "Start" button, respectively, to stop viewing the video, click the "Stop" button.

The blob detector is used to search blobs on images of an open video sequence. To use it, you need to set the required detection parameters and click the "Update parameters" button, then select the detection options "Monitoring objects", "Analysis of the trajectory of objects" and "Display signatures of objects". After that, using the video file navigator, start playing the video sequence or frame-by-frame viewing. On-screen form will show the found objects in the corresponding frames (see Fig. 1).

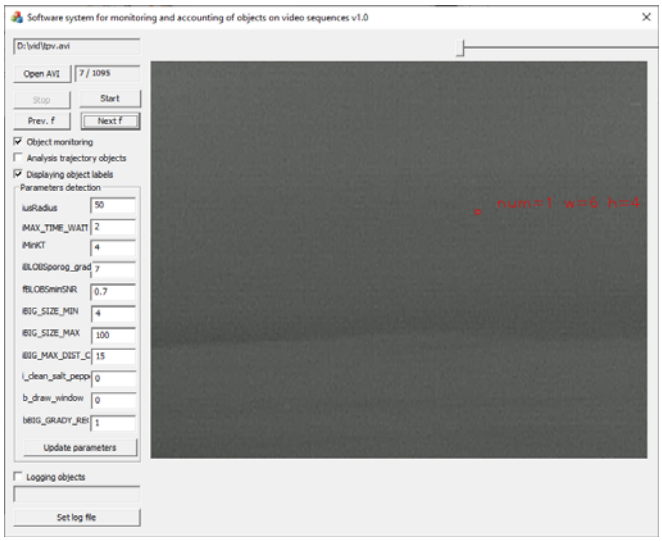

FIGURE I. BLOB DETECTOR.

The XML logging module is used to write information about found blobs to an xml file. To activate it, you need to click the "Set log file" button and set the name and path to the log file, then select the "Logging objects" option. After that, using the video file navigator to start playing the video sequence, the found blobs will not only be displayed on the screen, but also logged into an xml file, an example of the protocol file is shown in Fig. 2.

```
<?xml version="1.0"?>
<root>
  <frames>
    <frame num="5"/>
    <frame num="6">
      <object num="1" x="421" y="193" width="6" height="5" SNR="1.19" K="0"/>
    </frame>
    <frame num="7">
      <object num="1" x="421" y="194" width="6" height="4" SNR="1.15" K="1"/>
    </frame>
    <frame num="8">
      <object num="1" x="421" y="194" width="6" height="4" SNR="1.15" K="1"/>
    </frame>
    <frame num="9">
      <object num="1" x="421" y="194" width="4" height="4" SNR="0.90" K="2"/>
    </frame>
    <frame num="10">
      <object num="1" x="421" y="194" width="4" height="4" SNR="0.90" K="2"/>
      <object num="2" x="622" y="54" width="4" height="6" SNR="0.89" K="0"/>
    </frame>
    <frame num="11">
      <object num="1" x="622" y="54" width="4" height="6" SNR="0.89" K="0"/>
    </frame>
  </frames>
</root>
```

FIGURE II. BLOB PROTOCOL XML FILE.

IV. INPUT DATA PREPROCESSING

The input data is a set of video sequence files, stored on a PC hard drive or an external hard drive. Also, the input data for processing the video sequence are detector parameters (Fig. 3).

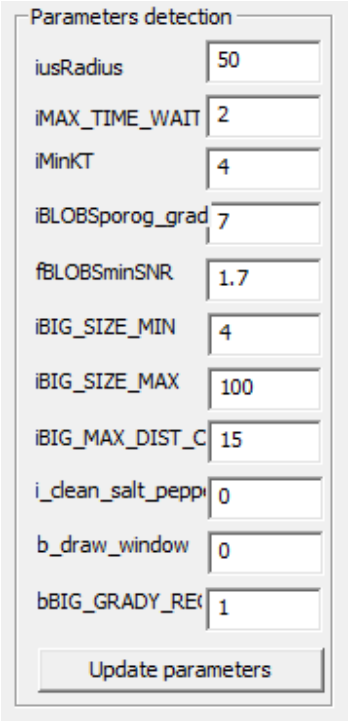

FIGURE III. SEQUENCE PROCESSING DETECTOR PARAMETERS

Let's consider in detail the name and purpose of each parameter:

iusRadius - radius of overlapping points (default value is 50);

iMAX_TIME_WAIT_TARGET - the maximum time (frames) to wait for an object to be updated, in case it has not been updated, deletion is performed (default value is 2);

iMinKT - the minimum number of object updates to make it visible (the default value is 4);

iBLOBSporog_grad - threshold of the gradient detector for binarization (default value is 7);

iBLOBSminSNR - minimum SNR of the blob found (blobs are cut off below this value) (the default value is 1.7);

iBIG_SIZE_MIN - minimum side of blob (default value is 4);

iBIG_SIZE_MAX - maximum side of blob (default value is 100);

iBIG_MAX_DIST_CLAST - distance for combining blobs into one (default value is 15);

i_clean_salt_pepper - flag for cleaning the salt / pepper noise (SP) 0 - do not clean, 1 - clean random SP, 2 - clean static SP (default value 0);

b_draw_window - output flag of the binarization debug window of the current frame (the default value is 0);

iBIG_GRADY_RECT - flag to use a less resource-intensive algorithm for allocating background areas (default value is 1).

Preliminary preparation of the input data consists in the selection of detector parameters. The selection of parameters should be carried out, focusing on the noise data of the environment. Below one can see an example of a very noisy video sequence (Fig. 4).

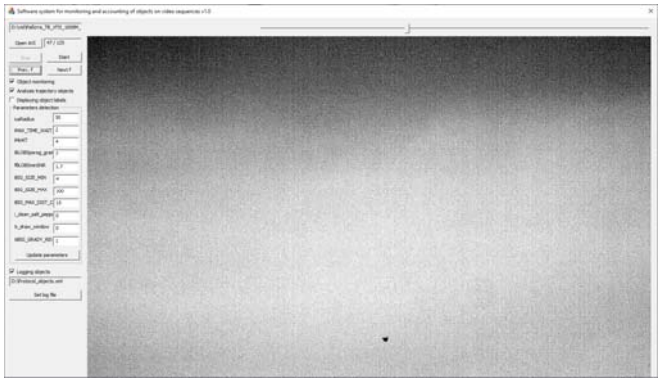

FIGURE IV. HIGHLY NOISY VIDEO SEQUENCE

It is clear from the figure that the threshold value of binarization is very small and needs to be increased. It is necessary to set the parameter b_draw_window = 1, click the "Update parameters" button on the screen form and click the "Next. f " for processing a frame with new detector parameters. As a result, an we get a binarized image of the frame of Fig. 5 highly noisy video sequences.

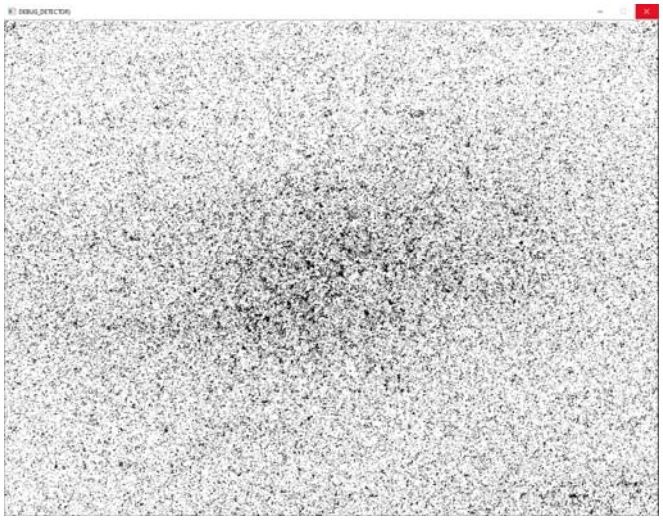

FIGURE V. BINARIZED IMAGE OF A HIGHLY NOISY VIDEO SEQUENCE FRAME

As can be seen on Fig. 5 default settings do not allow to select an object in the image. For reliable selection of an object, it is necessary to choose a binarization threshold. To do this, use the output of the binarized image. It is necessary to increase the parameter *iBLOBSporog_grad* = 70, press the button "Update parameters" on the screen form and click the button "Next. f" for processing a frame with new detector parameters. As a result, we can see a binarized image of an object of a very noisy video sequence with a new binarization parameter, it is well localized and detected by the program in Fig. 6.

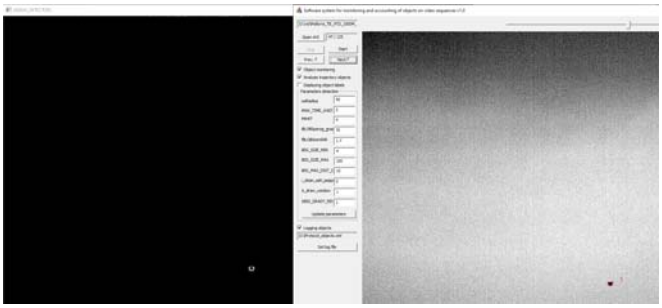

FIGURE VI. THE RESULT OF THE SELECTION OF BINARIZATION PARAMETERS

The program can work with files of video sequences of the avi or mp4 format. These files are a video sequence of monochrome or color images, our software can handle images not exceeding 2048 pixels on the larger side of the image. For encoding video sequence files, MPEG or H264 codecs are applied.

V. OUTPUT DATA

The output data is a XML format text files subset. These files can be stored on a PC hard drive either on an external data storage. The format and description of the output file is presented further. Below on fig. 7 one can see a software screenshot with an open video sequence containing only one object.

FIGURE VII. EXAMPLE OF A VIDEO SEQUENCE FRAME CONTAINING ONE OBJECT

Let's consider the output file, corresponding to this sequence (Fig. 8).

The root contains the *frames* element, which describes all the processed frames of the input video sequence. The *frames* element contains a set of single *frame* elements that describe each individual video sequence file. The frame element has a *num* data field that defines the frame number of the input video sequence (in the example of figure, frames with numbers from 39 to 47 are contained). In each processed frame, *objects* can

be found that are denoted by object in an XML document and have the following data fields that characterize the object and its properties:

num – object number in the current frame;

x,y – the position of the center of the object in the current frame horizontally in pixels;

width – object bounding box width;

height – object bounding box height;

SNR – the signal-to-noise ratio of the object to the background area on which it is located;

K – the number of frames on which the object is continuously detected.

```
<?xml version="1.0"?>
- <root>
  - <frames>
    - <frame num="39">
        <object num="1" x="1116" y="714" width="38" height="22" SNR="2.41" K="38"/>
      </frame>
    - <frame num="40">
        <object num="1" x="1115" y="714" width="37" height="22" SNR="2.46" K="39"/>
      </frame>
    - <frame num="41">
        <object num="1" x="1115" y="713" width="37" height="23" SNR="2.43" K="40"/>
      </frame>
    - <frame num="42">
        <object num="1" x="1114" y="714" width="37" height="22" SNR="2.50" K="41"/>
      </frame>
    - <frame num="43">
        <object num="1" x="1114" y="714" width="37" height="23" SNR="2.41" K="42"/>
      </frame>
    - <frame num="44">
        <object num="1" x="1114" y="714" width="37" height="22" SNR="2.39" K="43"/>
      </frame>
    - <frame num="45">
        <object num="1" x="1114" y="714" width="37" height="22" SNR="2.47" K="44"/>
      </frame>
    - <frame num="46">
        <object num="1" x="1114" y="714" width="37" height="22" SNR="2.42" K="45"/>
      </frame>
    - <frame num="47">
        <object num="1" x="1115" y="714" width="36" height="22" SNR="2.42" K="46"/>
      </frame>
    </frames>
</root>
```

FIGURE VIII. OUTPUT FILE EXAMPLE OF A VIDEO SEQUENCE CONTAINING ONE OBJECT

Fig. 9 presents an open video sequence containing several objects; the XML output file corresponding to this video is shown in Fig. 10.

FIGURE IX. EXAMPLE OF A VIDEO SEQUENCE FRAME CONTAINING SEVERAL OBJECTS

```
<?xml version="1.0"?>
- <root>
  - <frames>
    - <frame num="827">
        <object num="1" x="481" y="508" width="150" height="91" SNR="4.58" K="255"/>
        <object num="2" x="962" y="564" width="126" height="53" SNR="5.19" K="23"/>
        <object num="3" x="216" y="165" width="146" height="129" SNR="3.30" K="12"/>
        <object num="4" x="487" y="364" width="130" height="90" SNR="3.45" K="12"/>
        <object num="5" x="1147" y="585" width="119" height="41" SNR="5.05" K="6"/>
      </frame>
    - <frame num="828">
        <object num="1" x="481" y="508" width="150" height="91" SNR="4.58" K="256"/>
        <object num="2" x="962" y="564" width="126" height="53" SNR="5.19" K="24"/>
        <object num="3" x="216" y="165" width="146" height="129" SNR="3.30" K="13"/>
        <object num="4" x="487" y="364" width="130" height="90" SNR="3.45" K="13"/>
        <object num="5" x="1147" y="585" width="119" height="41" SNR="5.05" K="7"/>
      </frame>
    + <frame num="829">
    + <frame num="830">
    + <frame num="831">
    + <frame num="832">
    + <frame num="833">
    + <frame num="834">
    + <frame num="835">
    </frames>
</root>
```

FIGURE X. VIDEO SEQUENCE OUTPUT FILE EXAMPLE CONTAINING MULTIPLE OBJECTS

VI. CONCLUSION

In the paper we have described a software implementation of computer vision methods for highlighting contrasting objects in noisy images. A user-friendly graphical user interface is developed and depicted above.

The software that we have developed is applied for reading video files, their processing in order to seek and recognize objects present in the frame. Further research is aimed on its testing and improvement.

ACKNOWLEDGMENT

The reported study was funded by RFBR, project number 17-29-03407.

REFERENCES

[1] I. Korovin, M. Khisamutdinov, and D. Ivanov, “Improvement of a Video Sequence Singular Image,” in Proceedings of the 2nd International Conference on Advances in Artificial Intelligence, 2018, pp. 12–15.

[2] I. Korovin, M. Khisamutdinov, and D. Ivanov, “A Basic Algorithm of a Target Environment Analyzer,” in Proceedings of the 2nd International Conference on Advances in Artificial Intelligence, 2018, pp. 7–11.

[3] A. J. Danker and A. Rosenfeld, “Blob detection by relaxation,” IEEE Trans. Pattern Anal. Mach. Intell., no. 1, pp. 79–92, 1981.

[4] T. Lindeberg, “Feature Detection with Automatic Scale Selection,” Int. J. Comput. Vis., 1998.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” Comput. Vis. image Underst., vol. 110, no. 3, pp. 346–359, 2008.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006.

[7] K. Mikolajczyk and C. Schmid, “Scale & affine invariant interest point detectors,” Int. J. Comput. Vis., 2004.