Contour-based Object Detection in Dynamic Thermal Images with Noisy Background

Pia Lützen
Department of Natural Sciences and Industrial Engineering
Deggendorf Institute of Technology
Deggendorf, Germany
Email: pia.luetzen@stud.th-deg.de

***Abstract*—Recent breakthroughs in uncooled miniature bolometer systems enable to apply thermal imaging in low cost applications such as search-and-rescue. Object detection by means of infrared images promises usage in various scenarios. In this work, an image processing algorithm for the detection of rather small objects (covering 0.3 % - 1.5 % of sensor area) is presented. It is designed for moving devices, expecting a noisy back- and foreground still showing a steady distribution. Here, noisy means disturbing contours but also daylight noise. The key is a combination of background averaging and subtraction followed by image segmentation in order to find the target contour with minimum false alarm rate (FAR). Tests carried out in most complex environments yield FARs between 4 % - 16 %. Preconditions here are the awareness of the target size and a fix detection range.**

I. Introduction

Any object above 0 K emits thermal radiation. Here, the radiation's intensity is an indicator for the object's surface temperature. Infrared radiation is invisible for the human eye. However, Long Wave Infrared (LWIR) sensors (8-14 µm) enable to convert thermal radiation patterns into displayed infrared images [1]. This process is called thermal imaging, obtaining temperature mapping of a certain region of interest completely contactless [2]. Though, thermal sensors have been used in military applications (defense, surveillance, etc.) predominantly due to a high financial and a large form factor [1], [3].

Nowadays, breakthroughs in uncooled miniature bolometer systems enable the implementation of high quality and low cost thermal imaging in applications such as search-and-rescue (SAR) [1], [3]. With the different type of obtained information, thermal sensors are independent of illumination etc. and applicable in various scenarios. Due to the rising availability of small Unmanned aerial vehicles (UAVs) there is a growing interest in SAR applications as large areas can be mapped and missing objects be located [4]. Hereto, an efficient image processing algorithm is essential in order to detect a certain type of object. In [3], the ViBE background estimation [5] is used. This method is able to extract image regions that are hotter or colder than the environment. [4] also aims to detect objects from an aerial view. Here, complex background subtraction is not required as the application is established over the ocean which delivers a rather homogeneous image background. Object detection by means of thermal images is also done in [6]. Though, in this case the sensor is fixed and identifies moving objects.

In this paper a thermal image processing algorithm is presented which intends the sensor to be mounted on a moving device or vehicle. Thus, the images are dynamic. In contrast to previous work, the algorithm addresses equally-distributed but random backgrounds, or a mixture of back- and foreground (following simplified worded as background). An example are regions with strong and high growing vegetation. Moreover, the algorithm is conceived for objects covering a sensor area between 0.3 % - 1.5 %. Generally, the approach consists in the subtraction of an averaged value of an image in order to emphasize a target. Here, the sought-after targets are expected to show higher thermal emission than the environment. Additionally, the image is segmented into smaller regions of interest in order to find objects of a certain size. The single steps are explained in the beginning, coming to testing set-ups and evaluation of results in the end.

II. Method

The algorithm is realized by means of the computer vision library "OpenCV". The programming language is Python. However, in the following subsection, the software is not explained in detail but the single steps are constituted generally. Thereafter, the experimental set-up for software testing and corresponding conditions are described.

A. Algorithm Approach

The sequence of the algorithm's steps is explained. In figure 1 the image evolution process is depicted with an example image. Here, the numbering corresponds to the numbering of the single steps. The example image shows a piece of cloth midst a flower bed of palor palms. The detection range accounts approx. 16 cm.

1) **Convert image to grayscale**

   It is essential to have a single channel image, which means to consider simple incoming radiation intensities. There are grayscale infrared but also false color images that show three channels. However,

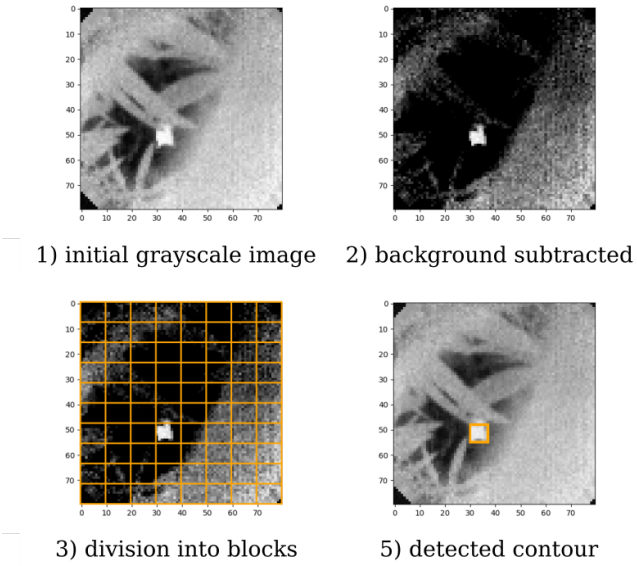

Fig. 1. Image evolution over detection process;
environment: flower bed of palor palms, indoor at 22°C; object: piece of cloth, 23°C, 4 cm^2; detection range: 16 cm

more channels increase computation effort as well as processing complexity. That is why the first step is to convert the image to grayscale, if it is not already.

2) **Background subtraction**

a) Calculate average intensity

For every received frame, the average intensity (= `current`) is calculated. Then, it is added up to the previous frames' intensity (= `previous`) in order to have a more reliable value for the average background intensity (= `overall`):

```
overall = (current + previous)/2
previous = overall
```

By doing this initially with some frames without an object, the value has a chance to stabilize. The goal is to generate a kind of basic noise as a scalar value that rises from the steady distribution of the background's thermal radiation.

b) Subtract average intensity from current frame

Following, the generated background noise is simply subtracted from the single pixels of every new received frame. To this, OpenCV's function "cv2.subtract()" is used. The function performs saturation which means that negative values are clipped to zero [7]. In contrast, results of regular subtraction are subject to integer overflow. Thus, by using "cv2.subtract()" also pixels showing a lower value than the basic noise are set to zero. Objects at higher temperatures than the average background are emphasized in the image.

3) **Divide image into blocks**

At this point, the object stands out in the image and holds the highest intensity. However, by definition, it covers a rather small area of the whole image. That is why the next step is to divide the image into smaller blocks at the object's size scale. The block size should be chosen to be approximately 2-3 times the target's area. This step corresponds to generating smaller regions of interest that can be investigated in more detail. Furthermore, large contours from the foreground are segmented and not seen as coherent structure anymore. Sought-after object however, will remain complete.

4) **Investigate mean intensity of each block**

The next step is to calculate the mean intensity of every single block in a current frame. Considering the block containing an object it will have a significant mean intensity. Figure 2 shows two blocks, (a) contains the object of interest and (b) shows a part of the background, more precise a part of a leaf. The mean intensity of a block with an object can be estimated by regarding the expected minimum object area and assuming that it shows the highest intensity value. For example, the minimum mean intensity of a block containing 80 pixels and an object of an expected minimum area of 20 pixels can be calculated as:

$$\frac{\text{min\_area} \cdot 0.5 \cdot \text{max\_value}}{80\,\text{px}} + \frac{(80 - \text{min\_area})\ \text{px} \cdot 0}{80\,\text{px}}$$

$$= \frac{20\,\text{px} \cdot 0.5 \cdot \text{max\_value}}{80\,\text{px}} + \frac{(80 - 20)\ \text{px} \cdot 0}{80\,\text{px}}$$

$$\approx 0.125 \cdot \text{max\_value}$$

Here, the max_value corresponds to the maximum value within the currently investigated block. As the contour

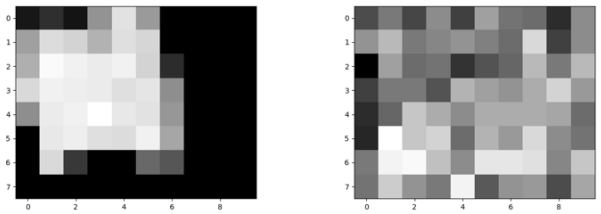

Fig. 2. Selection of blocks with and without object

will not show the highest value exclusively, its overall intensity is approximated as 0.5 · max_value times its area. Moreover, the minimum mean is targeted, that is why the rest of the block is assumed as zero. The estimated value is set as threshold for the blocks' mean intensity. Though, also high mean intensities may occur even without an object (see figure 2 (b)). That is why also a limit has to be set to the mean intensity value. If a block's mean intensity is within the set boundaries it is further investigated according to the subsequent steps.

5) **Contour Detection**

A block showing a high enough mean intensity is then investigated for existing contours. Hereto, image preprocessing steps are required in order to detect contours properly. This includes image thresholding, gaussian blurring and an opening as morphological transformation (refer to [8], [9], [10]). Then, OpenCV's function "cv2.findContours()" is executed to find coherent areas of similar intensity. If contours are found, they are passed to the instructions following below. The approach of observing mean intensities before contour detection intends to safe computation time as it is less effort to calculate an average than searching for contours in every single block.

6) **Check contour area**

Being aware of the observation height and the sensor's Field of View, the number of pixels covered by an object of certain size can be deduced. As soon as a contour is detected, its contour area can be investigated as a last step of verification. OpenCV's function "cv2.contourArea()" delivers the equivalent result to the spatial moment m_{00}. Spatial Moments are calculated by the following equation, where $\mathtt{a}_{m,n}$ is the array element at point (m, n):

$$m_{ij} = \sum_{m,n} (\mathtt{a}_{m,n} \cdot x^i \cdot y^j) \text{ , [11]}$$

Setting a threshold and a limit value for minimum and maximum contour area (e.g. expected target size ± 20 pixels), objects of a certain size are extracted.

7) **Detection success**

By also passing the area conditions, the image processing is finished and an object is considered as detected.

B. Experimental Setup

For testing the algorithm an infrared sensor with 80x80 pixels and a Field of View of 90° (horizontal and vertical) is used. Data are obtained in two different test scenarios.

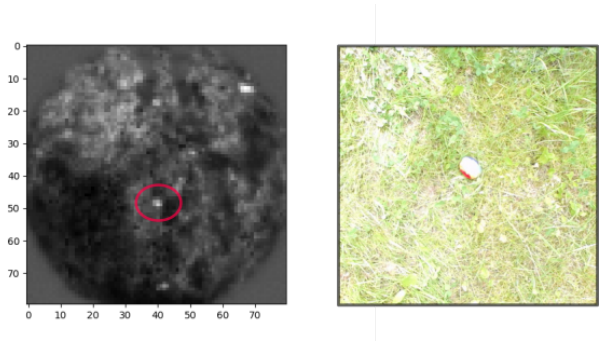

Fig. 3. Small object in infrared and visible spectrum;
environment: low growing meadow, sunny, outside at 23°C; object: fabric ball, 35°C, 35 cm^2; detection range: 1 m

In the first scenario, the images are taken outside in a low growing meadow. The sun is shining on the place. Diverse objects are spread on the ground, the objects' size ranges from 60 - 520 cm^2. By varying the detection range, the targets appear at different sizes in the image, which corresponds to the number of pixels covered by one object. Images are taken at 0.5 m, 1 m and 1.5 m height. Thus, four categories of target sizes arise: targets $\leq$ 15 pixel, 25 pixel, 40 pixel and $\geq$ 80 pixel. Moreover, the outside temperature accounts 24°C. The targets show temperatures between 20°C and 50°C. Single pictures are taken at a frame rate of approximately 3 fps, as the here applied serial interface is not able to reach higher values. To see the difference from the infrared to the visible spectrum, an additional camera is included that records the visual scenery.

The images shown in figure 1 are taken in the second scenario. As already mentioned, the test environment is a flower bed of palor palms, hence a high growing background which causes more disturbances. It is comparable to the conditions comprised by a maize field. A piece of cloth serves as target, its area accounts approximately 4 cm^2. With a detection range of 16 cm, the object is expected to cover a sensor area of 25 pixels. Furthermore, its temperature accounts 23°C, the leaves' temperature is 20°C and the room temperature approximately 22°C. In contrast to the first scenario, the measurement inside allows to record a video sequence with a frame rate of 50 fps, obtaining approximately 250 frames per video sequence.

III. Results and Discussion

A. Results from test scenario 1

Starting with the smallest object size (up to 15 pixels) it is found that those targets are not detected by the software. To ensure detection success, the target should cover a minimum of 0.3 % of the sensor area. Otherwise, objects are not distinguishable from the environment, like noticeable in figure 3. However, that may be improved by using a sensor with way higher resolution as structures will be presented in more detail.

TABLE I
DETECTION RESULTS 1

Object size	25 px	40 px	80 px
Covered sensor area	0.4 %	0.6 %	1.25 %
PoD	85 %	60 %	60 %
FAR	5 %	0 %	0 %

Considering objects of larger sizes it is proven that the algorithm works and is able to detect objects. For every size category the block sizes and hence the minimum mean intensity is adapted. Table I summarizes the received Probability of detection (PoD) with corresponding false alarm rate (FAR). It is to say that the PoD is rather small, as an object may be segmented by dividing the image into blocks, like shown in figure 4. Thus, contours are detected but do not fit the area limits. In the example, both contours have single areas of 17 and 16 pixels. Here, it is important to have a dynamic image and a high frame rate in order to ensure the acquisition of an image where the object is located within one block completely (see figure 5). Furthermore, a more steady detection range will increase the PoD values. Due to an inaccurate height during the measurements, the amount of covered pixels vary and do not correspond to the set area limits correctly. Another possibility is to widen the limits which, in turn, may increase the FAR. Apart from that, the FAR is good, the smaller the targets are the more likely some environmental disturbances may be considered as an object. However, the measuring conditions are relatively easy to detect an object as the background has a much lower temperature and shows no additional disturbing contours.

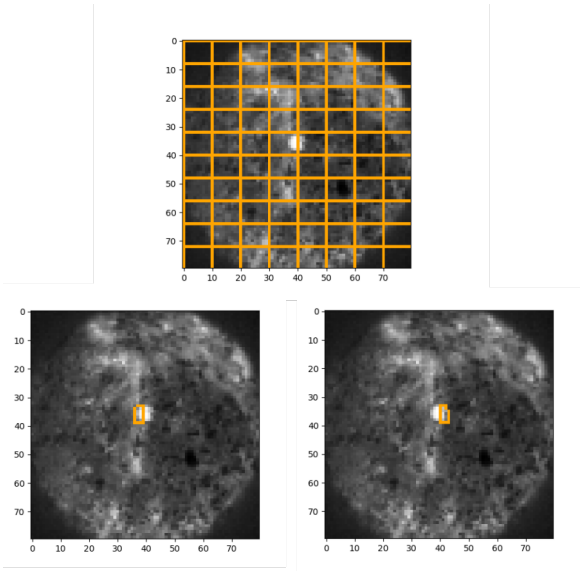

Fig. 4. Segmented object and resulting contours;
environment: low growing meadow, sunny, outside at 25°C; object: fabric ball, 33°C, 35 cm^2; detection range: 0.5 m

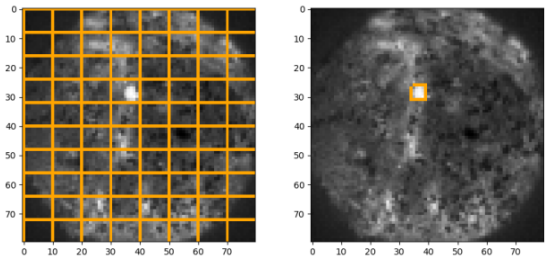

Fig. 5. Full contour detection;
environment: low growing meadow, sunny, outside at 25°C; object: fabric ball, 33°C, 35 cm^2; detection range: 0.5 m

For this reason, there is no need for a mean intensity limit, only a threshold. The objects may also be found by only searching for a contour in the picture. However, test scenario 2 shows a more complex environment.

B. *Results from test scenario 2*

In the second test environment, way more images containing an object exist due to a higher frame rate. It is to say that in every video the object is detected reliably. However, this scenario requires a limit for the mean intensity of each block.

It is found that the standard deviation of a block can be set as limit. In contrast to a block containing the target, blocks showing a part of a leaf will have a high intensity but low standard deviation.

Figure 6 shows some examples of detection successes. Here, the improving effect of a more dynamic image manifests, as there are more chances to have the target located within a block. Table II shows the results of detection hits. Although there are some false alarms, the different kind of environment requires the steps of background subtraction and individual investigation of smaller regions of interest to find a certain object. This is due to parts of the background that create own contours. As already shown in figure 1, the background subtraction helps to emphasize the object and fade background parts. However, contours still may be found in the large image. Dividing it into blocks at the target's size scale, large contours are not recognizable anymore, but the object's contour remains. This approach decreases the FAR to a minimum.

TABLE II
DETECTION RESULTS 2

	sequence 1	sequence 2
number of detection success	23	24
number of false alarms	1	4
FAR	4 %	16 %

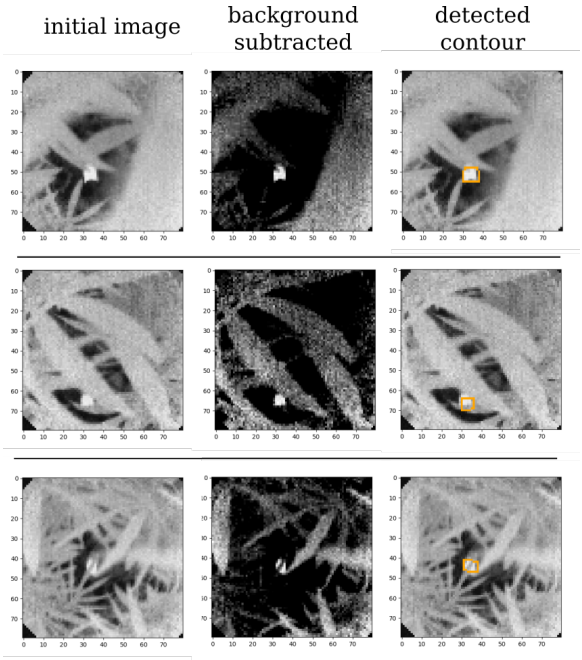

Fig. 6. Selection of detection success;
environment: flower bed of palor palms, indoor at 22°C; object: piece of cloth, 23°C, 4 cm^2; detection range: 16 cm

IV. Conclusions

Summing up, it is to say that the algorithm is reasonable at a certain kind of background which shows own contours. Important to know is that targets should cover a significant sensor area ($\geq$ 0.3 %), however be smaller than background disturbances. Depending on the target size, the algorithm's parameters of block size, mean intensity and area limits have to be adapted to ensure a reliable operation. Also a fix detection range is essential to be able to set the area limit values correctly.

Objects covering a small sensor area ($\leq$ 0.3 % of sensor area) are not detected yet. Though, the algorithm itself may show good performance at higher resolved images which can be investigated in further works.

References

[1] F. Pittaluga, A. Zivkovic, and S. J. Koppal, "Sensor-level privacy for thermal cameras," in *2016 IEEE International Conference on Computational Photography (ICCP)*, May 2016, pp. 1–12.

[2] R. Vadivambal and D. S. Jayas, "Applications of Thermal Imaging in Agriculture and Food Industry—A Review," *Food and Bioprocess Technology*, vol. 4, no. 2, pp. 186–199, Feb. 2011.

[3] J. Portmann, S. Lynen, M. Chli, and R. Siegwart, "People detection and tracking from aerial thermal views," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1794–1800.

[4] F. S. Leira, T. A. Johansen, and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera," in *2015 IEEE Aerospace Conference*, Mar. 2015, pp. 1–10.

[5] O. Barnich and M. Van Droogenbroeck, "ViBE: A powerful random technique to estimate the background in video sequences," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2009, pp. 945–948.

[6] Z. Yin and R. Collins, "Moving Object Localization in Thermal Imagery by Forward-backward MHI," in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, Jun. 2006, pp. 133–133.

[7] "Operations on Arrays — OpenCV 2.4.13.7 documentation," https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html\#subtract.

[8] "Image Thresholding — OpenCV-Python Tutorials 1 documentation," https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html.

[9] "Smoothing Images — OpenCV-Python Tutorials 1 documentation," https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html.

[10] "Morphological Transformations — OpenCV-Python Tutorials 1 documentation," https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html.

[11] A. Fernández Villán, *Mastering OpenCV 4 with Python: A Practical Guide Covering Topics from Image Processing, Augmented Reality to Deep Learning with OpenCV 4 and Python 3. 7*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2019.