Object Detection Using IR Camera

Aleksander Nawrat, Krzysztof Daniec, and Tomasz Warmuz

Abstract. The main goal of this paper is to implement and test various target detection algorithms in thermal imagery. The topic of target detection and moving target detection in infrared video streams is considered as a very challenging research topic in computer vision discipline. Although there is a huge number of different algorithms developed for target detection and target tracking in video streams generated by daylight cameras, there is still a limited number of solutions in field of infrared video streams.

This document introduces some of the basic techniques for image processing and some either classical approaches or state-of-the-art algorithms for target detection in IR imagery. Some of the chosen algorithms are implemented, tested and described in more details.

Keywords: object detection, IR video, blob detection, labeling.

1 Problem Description

The main problem faced during implementation of a target detection algorithm is such that computers perceives images differently than humans do. Human is able to perceive an image or at least a part of it at once and can immediately draw some conclusions about objects placement or target movement. On the opposite, computers, and all computer algorithms for image processing are working in pixel-by-pixel manner, thus each wrong assumption can cause bad results and wrong conclusions can be drawn. Therefore it is practically impossible to design a robust and 100%

Aleksander Nawrat · Krzysztof Daniec · Tobiasz Warmuz
Silesian University of Technology, Institute of Automatic Control,
Akademicka 16, 44-101 Gliwice, Poland
e-mail: {Aleksander.Nawrat,Krzysztof.Daniec}@polsl.pl,
tobiasz.warmuz@gmail.com

A. Nawrat et al. (Eds.): Advanced Technologies for Intelligent Systems, SCI 440, pp. 129–142.
springerlink.com

error-proof algorithm of target detection and even there are some relatively good and robust solutions, they have no chances while competing with a human vision system.

Thermal images characterizes with a significantly lower signal-to-noise ratio comparing to ordinary daylight images. This means that the power of the noise in the IR images is very high comparing to the power of the signal (information included in the image).

Huge advantage of thermal images is that even when the scene is poorly illuminated or not illuminated at all, thermal detector can sense the target due to the difference of the amount of thermal radiation. Thus it is the easiest case of tracking a mammal in a natural environment where the target (mammal) is much brighter on the thermal image comparing to the background due to its higher body temperature. This assumption was taken in [1, 2]. Unfortunately it is not always a good solution to follow with this assumption because there can be several real-life situations of tracking an object much colder that the background is. For example one can imagine a case of a highway on a warm and sunny day and the task is to track the air-conditioned cars driving on that motorway. Additional problem arises when we consider an ego-motion of the sensor, when the camera is mounted on a moving platform like in airborne IR imagery. In such cases the whole image scene moves either the targets or the background. In order to distinguish the background and the targets some motion compensation algorithms are needed. Such an algorithm for motion compensation needs to assume that the direction of the camera is pointed relatively perpendicularly to the scene because otherwise, when the scene includes more perspective, some of the further slowly moving targets may be not detected due to higher movement of the background on the front of the scene. According to the assumption of non-stationary sensor platform the algorithm needs to be additionally robust against basic transformations such as rotation, translation and scaling. The solution presented in [3] assumes a stationary sensor platform. Moreover, [3, 5, 6] provides also some solutions that assumes that the target feature do not change drastically over the course of tracking.

This kind of obstacles needs to be recognized at the very beginning stage of the algorithm implementation and all of the possible assumptions needs to be stated. This shows that algorithms for target detection may significantly differ in different applications and at the same time the wider the application we assume to be, the lower robustness of the algorithm we will get.

Nevertheless the ideal algorithm needs to work properly in different conditions it also needs to be computationally light and simple so it can work in real time and the result with a relatively smooth video stream (at least few frames per second).

2 Algorithm Decription

Recalling the assumptions for the algorithm stated in section 1 we know that the potential targets are assumed to be a regions of brighter pixels comparing to the

background. In order to enhance or to mark this kind of regions an algorithm called blob detection will be used.

2.1 Basic Idea

Blob detection results with a binary image that indicates the regions of brighter or darker pixel intensities comparing to the background. According to our assumptions we will focus on brighter regions and skip the case of darker regions. On this binary image, white regions indicates blobs. Using blob detection algorithm one can easily distinguish the objects and separate them from the background. In order to distinguish between different blobs another algorithm needs to be used which is blob labeling. This algorithm is responsible for proper labeling of blobs where each separate blob gets its own unique label. Finally having all blobs detected from the scene and properly labeled it is possible to get some characteristic features of each object like area, circularity factor, the mass center or others. Having the area for each blob known, one can also discard relatively small blobs that most likely results from inaccuracy of the IR detector.

Fig. 1 Visualization of blob detection and blob labeling algorithm

The fig. 1 presents the simplified scheme of operation and the basic idea of the blob detection and blob labeling algorithm. The first image to the left presents a sample grayscale IR image with a scene of two persons crossing the road. The algorithm of blob detection results with two blobs indicating the shape and position of the targets on the scene. The result of the blob detection algorithm is illustrated by the middle image. Finally the most right image illustrates the result of the blob labeling algorithm where both blobs gets their unique labels. Colors are used only for better visualization that both blobs are treated separately as independent blobs.

2.2 Algorithm Decription

As it was mentioned before, blob detection algorithms are focused on detection of brighter regions of image. The algorithm uses the fact that each flat region is separated from the background with a steep edge. Looking at the fig. 1 one could

think of an algorithm for blob detection which uses simple thresholding method but unfortunately it will not work correctly in case when the background is more significantly nonuniform. Apart from that, thresholding methods are very sensitive for any changes of background or object intensity. Fig. 2 presents results of applying thresholding method on sample IR image. Such an algorithm may give us satisfactory results in case of single image but when we consider a video stream, where the scene changes and objects moves it would be necessary to adapt threshold value according to changing conditions. The problem is what would that threshold depend on? The image on the left from the fig. 2 corresponds to the threshold value 100 while the right image corresponds to the threshold value 150.

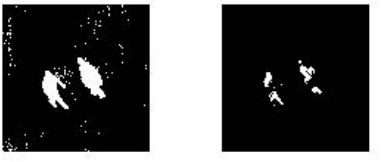

Fig. 2 Results of thresholding method

In section 1 it was assumed that the algorithm needs to deal with the nonuniform background and deal with changes of background and object intensity. The fig. 3 presents a situation where it is impossible to determine such a threshold value to make the thresholding method work for detecting each of the three objects from the scene.

That situation, mentioned above, motivates the usage of edge detectors in blob algorithms. Edge detection algorithms are focused on detection of image regions where there are drastic changes in luminance between relatively smooth regions while thresholding methods are only focused on each pixel luminance separately.

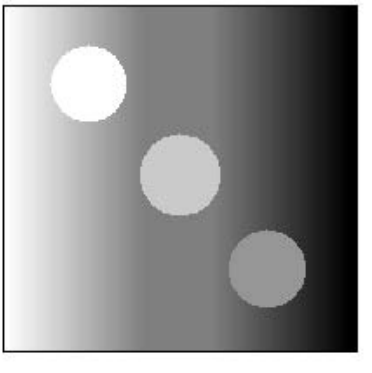

Fig. 3 Phantom image with three objects on the nonuniform background

3 Blob Detection

The results from edge detection algorithms motivates the idea of introducing more smoothing LoG filter comparing to the popular one described on [7]. Let us increase both parameters of LoG filter i.e. the size of the window and σ in order to gain a smooth and noise-free image. Let us introduce on a fig. 4 a 19×19 large filter window with $\sigma = 2.5$.

Fig. 5 presents the result of applying the modified LoG filter and due to those changes of LoG filter, the mid-cross-section of the resulting image from fig. 5 presented on fig. 6 has different (smoother) shape comparing to the result from edge detection algorithm. Yet again, red dots points the position of the edge of the object.

Recalling the assumptions for the algorithm stated in section 1 we know that the potential targets are assumed to be a bright regions (due to their higher temperature comparing to the background) observed from above (the IR camera is mounted on UAV). According to those assumptions we know that potential targets are most likely small bright regions on darker background as it can be seen on fig. 1. Since the potential targets are assumed to be relatively small it can be enough to apply a threshold method to get the inner part of the bright object.

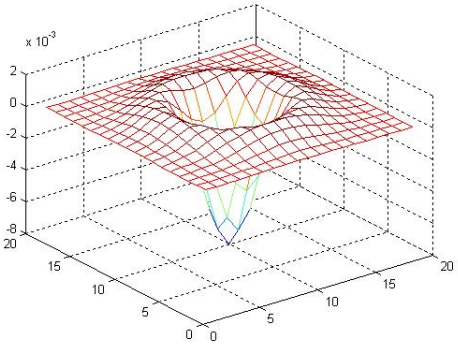

Fig. 4 The 19×19 version of LoG filter with $\sigma = 2.5$

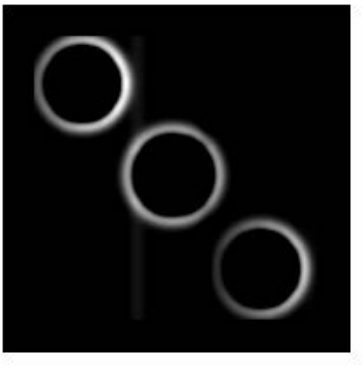

Fig. 5 Result of applying the LoG filter

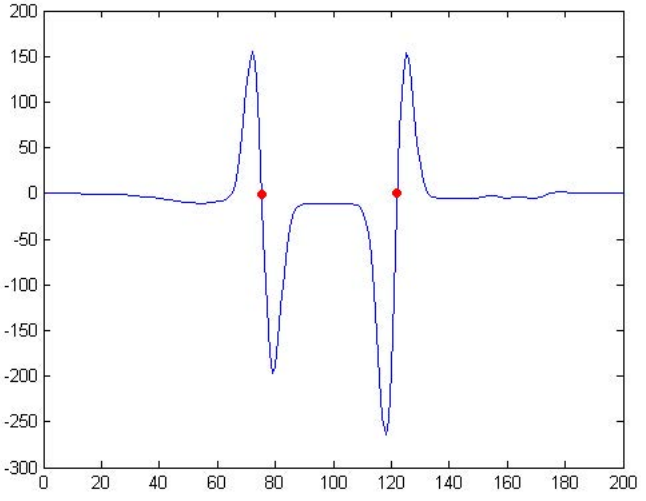

Fig. 6 The cross-section of the smoothed LoG

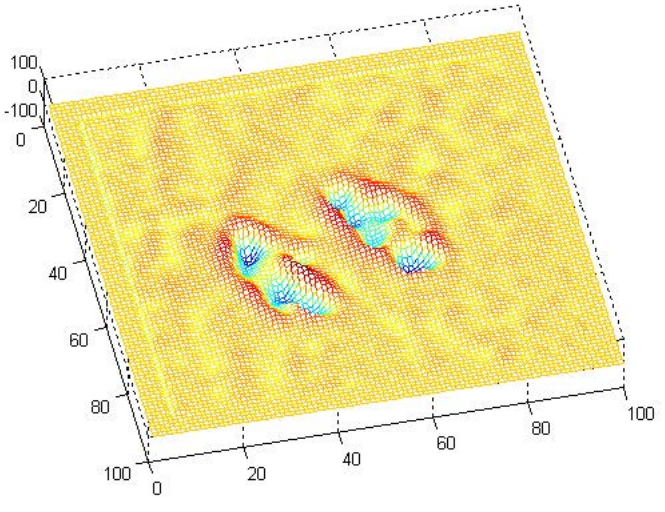

Fig. 7 The 3D visualization of smoothed LoG applied on real IR image

On the fig. 7 the result of applying LoG filter is presented. One can see that regions that corresponds to position of objects (brighter regions) has negative values (on the figure it can be seen as blue regions) due to occurrence of object edges. This is why for blob detection it may be sufficient to detect regions with negative values

of smoothed LoG using thresholding method. The fig. 8 presents the sample real IR image on the left-hand-side, the image filtered with LoG in the middle and on the right-hand-side the resulting binary image $B(x,y)$ applying thresholding method (white regions corresponds to regions of negative values). The resulting binary image shows two significant (the biggest) blobs that corresponds to the desired objects and a number of redundant small blobs.

Fig. 8 Results of blob detection algorithm applied on sample real IR image

The results seems to be satisfactory and takes us closer to the desired results as it was presented on fig. 1. Thus the next stage is to filter the binary image to discard small redundant blobs. In order to do this it is necessary to attach a unique label for each blob, then to calculate the area for each of blobs and next discard potentially non-significant blobs with area smaller than some value. Next section describes method for blob labeling.

4 Blob Labeling

The algorithm for blob labeling is needed to set a unique label for each separate blob. As a separate blob we understand a blob that has no other blob in 4-neighborhood. As 4-neighborhood of pixel for binary image B at $B(i,j)$ we understand pixels at $B(i-1,j)$, $B(i,j-1)$, $B(i+1,j)$ and $B(i,j+1)$. If in 4-neighborhood of pixel $B(i,j)$ there is a blob pixel it means that it is the same blob thus the same label for the neighboring blob needs to be assigned. There are different algorithms of blob labeling [4]. Fig. 9 presents the basic idea for the so called two-pass algorithm. In the first pass the algorithm goes pixel-wise and is looking for unlabeled blob pixels, where there are no labeled pixels in the neighborhood. This state of the algorithm is represented by dark gray arrows. While the algorithm finds a unlabeled pixel with no labeled pixels in the neighborhood (red arrows) it sets a unique label for the pixel. Finally, blue arrows represents the state of the algorithm when it finds an unlabeled pixel with other labeled pixel in the neighborhood. Then the label is set equal to the labeled neighboring pixel. Here, on the fig. 9 it is shown, that in case of concave objects it may happen that the same blob can get two separate labels (on

the figure it is depicted with two different pixel colors). In order to avoid such cases it is necessary to store the information about occurrences of situations where two neighboring pixels has different label. Different labels that corresponds to the same blob are stored in classes and in the second pass of the algorithm each pixel label is replaced with the minimum label from the class. More details for the two-pass algorithm can be found in [4].

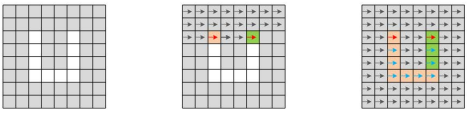

Fig. 9 Idea of the first pass from the two-pass algorithm of blob labeling

There is also another way for blob labeling that do not require to pass through the image more than once. On the fig. 10 one can see the simplified scheme of operation for the one-pass recursive algorithm of blob labeling.

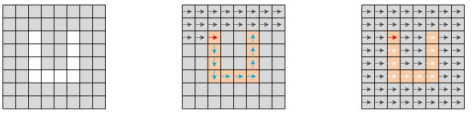

Fig. 10 Idea of the recursive algorithm of blob labeling

Similarly as in the algorithm presented on fig. 9 the dark gray arrows are looking for an unlabeled blob pixel. Once a new unlabeled blob pixel is found (red arrow) a unique label is set to the pixel and a recursive algorithm is ran for setting the same new label for all blob pixels in 4-neighborhood. As long there is a unlabeled blob pixel in the neighborhood the recursive algorithm is running (blue arrows). Finally, when whole blob is labeled with the same label, the algorithm goes back at the position of the first pixel from the blob and starts looking for another unlabeled blob pixel (dark gray arrows). White arrows corresponds to situations where an already labeled pixel is found. Since potential objects are assumed to be small bright regions, the recursive algorithm do not introduce too much burden for the processor. The result of recursive blob labeling algorithm for sample real IR image is presented on fig. 11.

The last stage of blob detection algorithm is to discard small redundant blobs that do not corresponds to desired objects. The algorithm used for that is also a two-pass method. First, we count the area for each blob by counting the number of pixels with the same label and next in the second pass, the pixels with labels that corresponds to small blobs (smaller than some threshold value) are set to zero thus they are

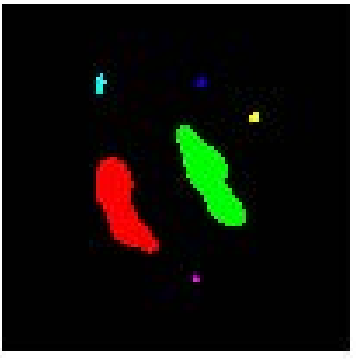

Fig. 11 Result of recursive blob labeling algorithm applied on sample real IR image

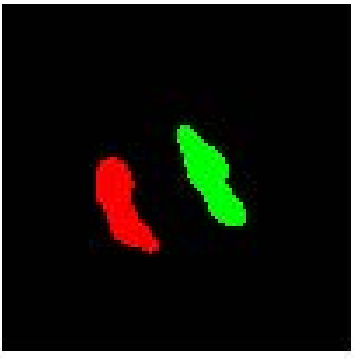

Fig. 12 Result of algorithm for discarding small and redundant blobs

Fig. 13 Result of the blob detection and blob labeling algorithm applied on a real IR image

no longer visible on the output screen. The result of such algorithm is presented on fig. 12.

Finally as a result, we are able to get relatively close to the desired result as presented on fig. 1. The fig. 13 presents the results of blob detection and blob labeling algorithms according to the results from fig. 1 and 8. Looking at the result from the fig. 13 one can notice that the algorithm enabled us to detect the lantern which has a relatively low temperature comparing to the pedestrians. In order to extend the algorithm, one can think of filtering out the low temperature objects in order to enhance the pedestrians. Furthermore, having such blobs, one can think of calculating some characteristic features for each blob i.e. some shape factors or mass center. For example it might be also possible to track the mass centers of each blob for further compensation of steady objects and enhancement of moving objects. Having the trajectory of the objects movement from number of the past frames one may try to apply some predictive algorithms in order to predict future movement of the object.

5 Results of the Algorithm

First fig. 14 presents the result of the blob detection algorithm on the sample IR video where three warmed batteries are rolled on the cold floor. The contrast between the batteries and the floor is very low and the signal-to-noise ratio is very low as well. The batteries are rolling on the floor rather rapidly but it do not cause any problem for the algorithm and each battery is properly detected as separate objects. The detected objects are enhanced by setting the R (red) channel to 255 for each pixel that to corresponds to the certain blob. the red channel is not accidentally chosen because red channel provides less information than green channel but more information than blue channel. Moreover red is better visible on graylevel images comparing to blue. Fig. 15 presents the result of blob detection algorithm on sample IR video. One can distinctly see that each pedestrian is properly detected as separate object and additionally some other objects has been detected i.e lanterns and road signs. Additionally fig. 16 presents the same scene as fig. 15 but shifted. The pedestrians are displaced and new object has appeared on the scene namely the car. On the bottom right corner of both frames there is a region of warmer ground due to some underground municipal waste pipings. Because the contrast of those piping is lower and the edges are not that sharp comparing to objects like pedestrians or cars the blobs are rather weak and small.

Fig. 14 Result of the blob detection algorithm on the sample IR video

Next two fig. 17 and 18 presents the another set of results for blob detection algorithm on sample IR video. On the scene there are several pedestrians and cars. Pedestrians are much warmer comparing to the cars thus they are much better detected by the algorithm. The video has been captured in winter thus it can be the reason that the cars are very cold and it is difficult for the algorithm to detect them. On the scene there are also several windows detected. The situation gets more

Fig. 15 Result of the blob detection algorithm on the sample IR video

Fig. 16 Result of the blob detection algorithm on the sample IR video

complicated if the scene on the IR video do not follow the second assumption stated in section 1 that the potential target is supposed to be much brighter comparing to the background. This situation happens very often in hot summer time on the street. One can imagine very hot asphalt street and pedestrians that are colder than the street what results in the scene where potential targets (pedestrians) are darker than the background (street). Cars are also darker than the street due to the fact that they use air conditioning very often. Results of such unfortunate case is presented on fig. 19 and 20. Fig. 19 presents the scene where the car and one pedestrian are in a shadow made by the tree nearby. Due to the very high sunlight illumination there

Fig. 17 Result of the blob detection algorithm on the sample IR video

Fig. 18 Result of the blob detection algorithm on the sample IR video

are plenty of false and weak blobs. Either the car or the pedestrian are detected properly. Next fig. 20 emphasizes the situation where the background is brighter comparing to the potential target (against second assumption stated in section 1) In such cases the blob regions are marked on the other side of the desired object what is obviously improper. One can see this effect on the bottom of the car on the fig. 20. The results would be more likely proper if the videos would be captured during the night when there is no influence of the strong sunlight radiation. This considerations shows that the algorithm results strongly depends on weather conditions and that it is very difficult to implement an universal algorithm.

Fig. 19 Result of the blob detection algorithm on the sample IR video

Fig. 20 Result of the blob detection algorithm on the sample IR video

6 Conclusions

The results for the algorithms presented in section 2 shows that it is very difficult to design such an algorithm that would provide proper results in various weather conditions, for various kind of potential targets observed from different distances. The presented algorithm for blob detection gives satisfactory results as long as the assumptions stated in section 1 are satisfied. The most problematic case is described in section 2 where the background is brighter comparing to the desired target. In such case, the blob detection algorithm doesn't mark the object but it only enhance its outer contour. Such case may happen in sunny day during summer. It shouldn't be a problem to modify the blob detection algorithm in such way that it would handle such case. Nevertheless for the Harris corner detection algorithm there is no difference whether the object is brighter than the background or not and this is an advantage.

References

1. Chen, J.Y., Reed, I.S.: A detection algorithm for optical targets in clutter. IEEE Transactions on Aerospace and Electronic Systems 23(1), 46–59 (1987)
2. Longmire, M.S., Takken, E.H.: Lms and matched digital lters for optical clutter suppression. Applied Optics 27(6), 1141–1159 (1988)
3. Davies, D., Palmer, P., Mirmehdi: Detection and tracking of very small low contrast objects. In: Proceedings of 9th British Machine Vision Conference, pp. 599–608 (1998)
4. Shapiro, L., Stockman, G.: Computer Vision. Prentice Hall (2002)
5. Strehl, A., Aggarwal, J.K.: Detecting moving objects in airborne forward looking infrared sequences. Machine Vision Applications Journal 11, 267–276 (2000)
6. Choudhary, M., Braga-Neto, U., Goutsias, J.: Automatic target detection and tracking in forward-looking infrared image sequences using morphological connected operators. Journal of Electronic Imaging 13(4), 802–813 (2004)
7. Wang, R.: Sharpening and Edge Detection (September 2009), `http://fourier.eng.hmc.edu/e161/lectures/gradient/gradient.html`