Real-Time Moving Object of Interest Detection in Multi-Sensor Imaging System

Miloš Pavlović, *Member, IEEE,* Nikola Stojiljković, Ivan Gluvačević,
Miljan Vučetić, *Member, IEEE,* Miroslav Perić, *Member, IEEE*

***Abstract*— The video surveillance systems usually require a constant user's observation of the scene in order to respond on time to situations that could be risky. In order to allow the system to understand and react to the environment and in that way to reduce user's need for manual intervention, objects of interest detection algorithms need to be integrated into the video systems. This paper proposes a real-time algorithm for moving object of interest detection, applicable to the multi-sensor imaging system. Algorithm introduces a combination of the method for motion detection in images - Optical flow and deep learning method for detecting objects of interest - YOLO algorithm. The objects of interest for detection in this paper are pedestrians and cars.**

***Index Terms*—Motion detection, Object detection, Real-Time, Multi-Sensor imaging**

I. Introduction

The video surveillance systems usually require a constant presence of user and observation of the scene in order to respond on time to situations that could be risky. In other words, a common problem is the lack of "intelligence" to function independently without the need of user's manual intervention.

Object detection and tracking algorithms are also an important part of the video surveillance systems. Most often, tracking algorithms implemented in video systems are algorithms from the class of semi-automated algorithms. This means that the user needs to manually select the target to be tracked, and then the algorithm takes over the tracking process. In the case when objects that are in motion need to be selected (especially when they are moving fast), the problem of manual selection becomes especially pronounced. For this reason, there is a need for algorithms for fully automated detection of moving objects.

The detection of moving objects in video as a problem of computer vision from early beginnings has been approached in various ways, which often depended on the computer processing power, but also very often of the problem understanding. The rapid improvements in video sensor quality and resolution and the dramatic increase in processing power of platforms running the algorithms in the previous period have favored the development of new algorithms and applications in this segment of computer vision.

The primary goal of a moving object detection algorithm is to achieve high detection accuracy in different conditions in the scene, as well as sufficient execution speed for real-time operation. In addition to work on visible light (color) images, in order to provide the ability to work in different weather and low light conditions, and also to work in complete darkness, using sensors that work in infrared (IR) range, the aim is development of algorithm for detection in Short-Wave IR (SWIR - 1.4-3μm), Medium-Wave IR (MWIR - 3-8μm) and Long-Wave IR (LWIR - 8-14μm) [1] imaging as well.

IR imaging sensors have the grayscale output and objects on these type of images have very specific features compared to the features of same type of objects in color images. Most of publicly available datasets for training neural networks for object classification and detection contain only color images, so neural networks trained only on color images are not able to detect objects in infrared images with high accuracy. On the other hand, color images are very sensitive to many types of visual obstacles (low light conditions, night conditions, haze problems, various atmospheric disturbances, etc.), especially in detection various objects at very large distances, where the influence of these various disturbances is very high, and all these problems prevent algorithm from reaching its maximum detection accuracy. As the task is to develop an algorithm for moving objects of interest detection for real-time application, the presented algorithm in this paper is a combination of conventional method for motion detection in images - Optical flow [2] and deep learning method for objects of interest detection - YOLO algorithm [3]. The objects of interest for detection in this paper are pedestrians and cars.

The paper is organized as follows. Section II describes the methodology of work, method for the motion detection - Optical flow, as well as the objects of interest detection method based on deep learning. Section III gives a description of the implemented system for the moving object of interest detection. Section IV describes the experimental work including the experimental setup, results and discussion. Section V lists conclusions and indicates directions for future work in this research area.

Miloš Pavlović is with Vlatacom Institute of High Technologies, Blvd. Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: milos.pavlovic@vlatacom.com).

Nikola Stojiljković is with Vlatacom Institute of High Technologies, Blvd. Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: nikola.stojiljkovic@vlatacom.com).

Ivan Gluvačević is with Vlatacom Institute of High Technologies, Blvd. Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: ivan.gluvacevic@vlatacom.com).

Dr Miljan Vučetić is with Vlatacom Institute of High Technologies, Blvd. Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: miljan.vucetic@vlatacom.com).

Dr Miroslav Perić is with Vlatacom Institute of High Technologies, Blvd. Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: miroslav.peric@vlatacom.com).

II. METHODOLOGY

A. Optical flow

Optical flow [2] represents the movement of objects between consecutive frames in a video sequence, caused by the relative movement of the object and the camera. The problem can be expressed as illustrated in Fig. 1.

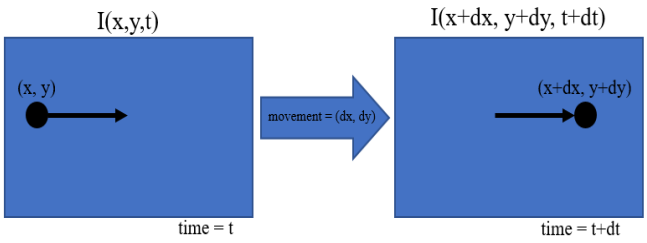

Fig. 1 Optical flow illustration

As shown in Fig. 1, between consecutive frames it is possible to express the intensity of image I as a function of the spatial coordinates (*x*, *y*) and time (t). In other words, if the pixels of image I(x, y, t) from the video sequence at time *t* are moved by (*dx*, *dy*) over time *dt*, result is the new image I(x + dx, y + dy, t + dt).

Several assumptions are made for the computation of Optical flow.

1. An object's pixel intensities are constant between consecutive frames.

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{1}$$

2. With Taylor Series Approximation, we have:

$$I(x + dx, y + dy, t + dt) = \\ = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \cdots \\ \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \tag{2}$$

3. By dividing the previous equation by δt, the equation of Optical flow is:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{3}$$

where: $u = \frac{\partial x}{\partial t}$ and $v = \frac{\partial y}{\partial t}$

$\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ present the image gradients along the horizontal and vertical axes and time. To solve the Optical flow problem, a solution of u ($\frac{\partial x}{\partial t}$) and v ($\frac{\partial y}{\partial t}$) are required to determine motion over time. It can be noted that the Optical flow equation for u and v cannot be solved directly, since there is only one equation for two unknown variables.

There are two different approaches to solving Optical flow problems: Sparse [4] and Dense [5]. Sparse gives the flow vectors of characteristic features in the image, while Dense Optical flow gives the flow vectors of all pixels in the image up to one flow vector per pixel. Dense Optical flow has better accuracy in motion detection but is more computationally complex [6].

B. Object Detection in Image

In recent years, much has been done in the development of object detection algorithms using a standard camera without additional sensors. State-of-the-art object detection algorithms use deep neural networks.

YOLO (YOU ONLY LOOK ONCE) is an object detection algorithm published in 2016 [3]. The algorithm was developed to perform the process involving detection and classification in one step. The idea behind the YOLO algorithm is different from other detection methods (RCNN, Fast RCNN, and Faster RCNN) in that the bounding boxes and the classes of the detected objects are determined after only one evaluation of the input image.

First, the input image is divided into a grid of S × S cells. Further, for each grid cell, B bounding boxes are defined together with a confidence score. The Confidence score is defined as:

$$\mathrm{C} = P_r(Object) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}} \tag{4}$$

In this case, C is the probability that an object exists in each bounding box, while the IOU represents the Intersection Over Union ratio and takes the value from the range [0, 1]. The intersection is the area of overlap between the predicted bounding box and the ground truth bounding box, while union represents the total area of the predicted bounding box and the ground truth bounding box. The IOU value closer to 1 indicating that the predicted bounding box is closer to the ground truth bounding box. The illustration of the IOU metrics is as shown in Fig. 2.

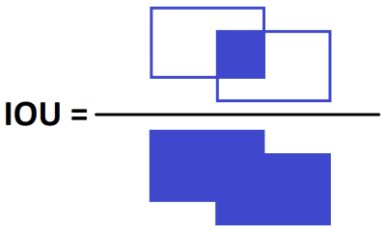

Fig. 2 Illustration of the IOU metrics

Simultaneously with the construction of bounding boxes, each grid cell also predicts a conditional probability of class. The class-specific probability [3] for each grid cell is defined as:

$$P_r(Class_i|Object) * P_r(Object) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}} \\ = P_r(Class_i) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}} \tag{5}$$

The following equation is used for loss calculation and ultimately confidence optimization:

$$Loss = \sum_{i=0}^{s^3}\sum_{j=0}^{A} 1_{ij}^{obj}\left[\left(b_{x_i} - b_{\hat{x}_i}\right)^2 + \left(b_{y_i} - b_{\hat{y}_i}\right)^2\right]$$

$$+\alpha_{coord}\sum_{i=0}^{s^3}\sum_{j=0}^{A} 1_{ij}^{obj}\left[\left(\sqrt{b_{w_i}} - \sqrt{b_{\hat{w}_i}}\right)^2 + \left(\sqrt{b_{h_i}} - \sqrt{b_{\hat{h}_i}}\right)^2\right]$$

$$+\sum_{i=0}^{s^3}\sum_{j=0}^{A} 1_{ij}^{obj}(C_i - \hat{C}_i)^2 \qquad (6)$$

$$+\alpha_{noobj}\sum_{i=0}^{s^3}\sum_{j=0}^{A} 1_{ij}^{obj}(C_i - \hat{C}_i)^2$$

$$+\sum_{i=0}^{s^3} 1_{i}^{obj}\sum_{c\in classes}(p_i(c) - \hat{p}_i(c))^2$$

For each prediction, the position of the center and the size of the bounding box are corrected using the given loss function. In loss function, parameter A is the number of bounding boxes for each grid cell of S × S image grid. The variables b_x and b_y present the center of each bounding box prediction, while b_w and b_h refer to the bounding box width and height. The importance of boxes with and without objects is controlled with the variables α_{coord} and α_{noobj}. C presents confidence, while *p(c)* refers to classification prediction. 1_{ij}^{obj} equals to 1 presents the responsibility for predicting the object of the j^{th} bounding box in the i^{th} grid cell and is equal to 0 otherwise. If the object is in the cell *i*, 1_{i}^{obj} is 1, 0 otherwise.

While the loss is used to evaluate model performance, the accuracy of model predictions in detecting objects is calculated by the equation of average precision, where P(k) refers to precision at threshold *k*, while $\Delta r(k)$ refers to recall change.

$$argPrecision = \sum_{k=1}^{n} P(k)\Delta r(k) \qquad (7)$$

The basic architecture of the YOLO model contains 24 convolution layers followed by two fully connected layers. YOLO is later improved with different versions such as YOLOv2 [7], YOLOv3 [8] or YOLO-LITE [9], YOLOv4 [10] in order to improve detection performance and reduce computational time.

III. Algorithm Description

Fig. 3 shows a block diagram of the moving objects of interest detection algorithm.

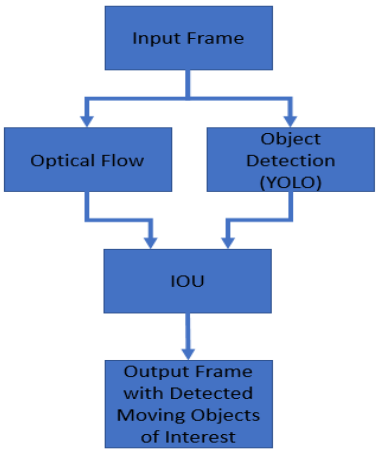

Fig. 3 Block diagram of the moving objects of interest detection algorithm

As can be seen from the block diagram shown in Fig. 3, the moving object of interest detection algorithm in this paper is implemented from two steps.

The input image is forwarded to two blocks running in parallel. The first block is responsible for motion detection in the image independently of which object from the scene caused that motion. This block uses the conventional method for motion detection in an image - Optical flow, primarily because of the high execution speed of this method in real-time. As the advantage of Dense over Sparse Optical flow has been presented in the literature [6], Dense optical flow has been applied in this work for motion detection. The result of calculating Optical flow is a flow matrix that is represented by its *hsv* (hue, saturation, value) model, and then converted into the RGB model. Converting the obtained RGB model to a grayscale image and then binarizing with a defined threshold *Tbin* produces a binary image in which the white pixels present the movements in the image. In such a binary image, it is then necessary to find the closed contours of the white pixels that actually represent the moving objects. For the obtained contours, the next step is comparing their surfaces with a defined threshold *Tcont*. In the set of moving objects, only those whose surface exceeds the defined threshold *Tcont* should be retained in order to eliminate very small objects or detected movements that present noise in the image. As a result of this procedure and the output of this block are bounding boxes around the regions detected as moving objects.

The second block to forward the input image is the block for detection objects of interest. Bounding boxes around the objects of interest are the output of this block. The detection method used in this work is the YOLO object detection method of version 3 [8], which has been shown to have very good performance in objects detection from the YOLO method class, and can be executed fast enough on a computer with a GPU for possible real-time implementation.

Detections of moving objects and detections of objects of interest are then forwarded to the block to calculate the IOU metric between them. By the IOU metric is actually determined whether the detected movements in the image

were caused by the objects of interest. If the overlap between detected movements and detections of objects of interest is sufficient and exceeds the defined threshold *Tiou*, it can be said that the detected objects of interest are in movement.

The output of the algorithm is an image with bounding boxes around moving objects of interest.

IV. Experimental work

The system setup used in this paper is based on vMSIS3 (Vlatacom Multi-Sensor Imaging System) [11]. Integrating visible and infrared imaging sensors and providing ultra-long-range target detection, recognition, and identification, vMSIS is a state-of-the-art monitoring and surveillance system. It contains of three video channels: visible light (FULL HD resolution: 1920x1080 pixels), thermal (resolution: 640x480) and short-wave infrared (resolution: 720x576 pixels).

The Algorithm is implemented in Python programming language using the OpenCV library to perform most of the functions related to image processing. The algorithm is applied to the compressed video signal. The stream from cameras was achieved via RTSP (Real-Time Streaming Protocol).

The algorithm is initialized by the first frame of the video sequence. For each subsequent frame that arrives from the camera, it is observed in relation to the previous one whether and what kind of movement occurred in the video sequence. After the input frame arrives, the Optical flow calculation for two successive frames is started. Dense Optical flow [12], *Farneback* implementation [13] from OpenCV library was applied. In parallel with the Optical flow calculation, the input frame is forwarded to the object of interest detector, YOLO algorithm, version 3. The YOLO method is implemented in the C programming language in the Darknet framework [14] using the CUDA acceleration library on the GPU. Object detection was performed on a reduced resolution image (448x448 pixels) primarily due to the execution speed to enable real-time operation. Using the NVIDIA RTX 2080 GPU for 448x448 pixel images, the processing time per frame is about 16ms. Reducing the resolution affects the detection performance (lower resolution - poorer detection), but the empirical assessment has shown that the resolution used is quite satisfactory for the quality of the detector in relation to the execution speed. Processing time per frame required for the Optical flow calculation on the same resolution (with parameters: *Tbin* = 50 and *Tcont* = 0.35% of the image area) is about 8ms, while the time consumption for the whole algorithm is about 27ms per frame ~ 37 frames per second (fps). The achieved speed is quite satisfactory for cameras that operate with 30 fps. The YOLO model pretrained on the COCO data set [15], which includes classes of pedestrians and cars, was used for object detection on visible light and SWIR images. Although the COCO dataset contains only the color images, the model was able to generalize well enough that the same model can be applied to the SWIR images. For detection on thermal images that are significantly different from color and SWIR images, pretrained model on COCO dataset was additionally trained with 9229 thermal images (8314 from FLIR dataset [16] and 915 taken from VMSIS thermal channel) to further improve detection performance on thermal images, especially for small objects at greater distances from the camera. Images from FLIR dataset were labeled for car and pedestrian classes, while for the purposes of this work we manually labeled images from VMSIS thermal channel for car and pedestrian classes using *Yolo mark* - application for marking bounded boxes of objects in images for training YOLO detector [17].

For the obtained detections of moving objects and detections of objects of interest, their overlap is then determined by applying the IOU metric, and if the overlap is sufficient and exceeds the threshold of 0.5, a decision is made that detected objects of interest are in motion and only bounding boxes around these objects are retained represents the end result of the algorithm.

In Fig. 4 are given three image sequences (Visible, SWIR and Thermal) that show the results of the algorithm, while Table I presents objective performance of the algorithm on the same sequences. In Table I Accuracy is the percentage of video sequence frames in which all moving objects of interest are detected, while false alarm is the percentage of frames in which some of the objects of interest that are not in movement are detected as moving.

In visible and thermal image sequences, cars were selected as objects of interest. The observed sequences present scene where pedestrians are beside the cars, and in the background is parking with static cars. The scene illustrates a situation with moving objects of interest, but also objects of interest that are not in movement. The algorithm distinguishes between objects of interest in movement (moving cars at the crossroads) and other objects in the scene (moving pedestrians and static cars in parking). In the sequence of SWIR images pedestrians were selected as objects of interest. The observed scene contains pedestrians standing, as well as those in motion. The algorithm detects when pedestrians who in the previous scenes were not in the frame appear, and if they are in motion, they are detected with a red bounding box.

Detection accuracy is mostly affected by the partial occlusions that are very often - a static or moving object covers an object of interest, image contrast changing, and always present noise especially in infrared images. The lowest accuracy is obtained for SWIR sequence, and the main reason for that is used detector of objects of interest, trained with visible images. The false alarm error occurred due to the appearance of any moving object in front of the object of interest that is not in movement, which is the main weakness of the simple IOU comparison between the objects of interest detections and detections of movement in the image.

TABLE I

	Number of frames	Accuracy [%]	False alarm [%]
Visible	622	68.3	2.03
SWIR	724	53.8	3.18
Thermal	600	80.5	3.32

V. Conclusion

The paper shows that the combination of the Optical Flow and the YOLO algorithm can be very successfully used to detect objects of interest in motion in a video sequence. As Optical flow is an exact method, regardless of the type of sensor and the moving object, using Optical flow, motion in a video sequence can be detected very successfully. However, the YOLO detector, like any other deep learning-based model, largely depends on the training set used for training. Object detection problem in color images is largely solved and the detection performances are very good. However, the problem of detection of objects of interest on infrared images remains open for further improvements. The most common problem is the lack of data of infrared images for training, primarily due to the cost of the infrared sensors themselves. Although insufficient, publicly available sets of LWIR and MWIR images can be found, but they are almost non-existent for SWIR sensors. As future work in this area and the improvement of the presented algorithm is seen in the creation of bigger infrared image datasets for detectors training to achieve better performance on LWIR and SWIR images.

References

[1] J. Lloyd, "Thermal imaging systems", Springer Science & Business Media, 2013.

[2] Beauchemin, S. S., & Barron, J. L., "The Computation of Optical Flow", *ACM computing surveys (CSUR)*, *27*(3), 433-466.

[3] Redmon J., Divvala S., Girshick R., & Farhadi A. (2016), "You Only Look Once: Unified, Real-Time Object Detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).

[4] T. Lim, B. Han, J. Han, "Modeling and Segmentation of Floating Foreground and Background in Videos", Pattern Recognition 45 (4) (2012) 1696-1706.

[5] Farnebäck, G. (June, 2003), "Two-frame Motion Estimation Based on Polynomial Expansion", Scandinavian Conference on Image analysis, (pp. 363-370). Springer, Berlin, Heidelberg.

[6] Chapel M. N., Bouwmans T. (2020), "Moving Objects Detection with a Moving Camera: A Comprehensive Review", *arXiv preprint arXiv:2001.05238*.

[7] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv preprint, 2017.

[8] Redmon J., Farhadi A. (2018), "Yolov3: An Incremental Improvement", *arXiv preprint arXiv:1804.02767*.

[9] Huang, R., Pedoeem, J., & Chen, C. (2018, December), "YOLO-LITE: a Real-Time Object Detection Algorithm Optimized for Non-GPU Computers", 2018 IEEE International Conference on Big Data (Big Data) (pp. 2503-2510).

[10] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020), "YOLOv4: Optimal Speed and Accuracy of Object Detection", *arXiv preprint arXiv:2004.10934*.

[11] Vlatacom Institute: Electro-optical systems VMSIS3, product brochures, available on-line: https://www.vlatacominstitute.com/electro-optical-systems, accessed on 09-July-2020.

[12] OpenCV Optical flow https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html, accessed on 09-July-2020.

[13] OpenCV Farnerback optical flow https://docs.opencv.org/3.4/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af, accessed on 09-July-2020.

[14] https://pjreddie.com/darknet/yolo/, accessed on 09- July -2020.

[15] COCO dataset, http://cocodataset.org/#home, accessed on 09-July-2020.

[16] https://www.flir.com/oem/adas/adas-dataset-form/, accessed on 09-July-2020.

[17] https://github.com/AlexeyAB/Yolo_mark, accessed on 04-September-2020.

Fig. 4 Examples of the moving objects of interest detection algorithm