EdgeLabel: A Video Annotation Method for Moving Camera using Edge Devices*

Hai-Thien To[1], Khac-Hoai Nam Bui[2], and Chi-Luan Le[1,**]

[1] University of Transport and Technology, Hanoi, Vietnam
{thienth, luanlc}@utt.edu.vn
[2] Viettel Cyperspace Center, Hanoi, Vietnam
hoainam.bk2012@gmail.com

Abstract. Currently, AI edge device is becoming the trend in real-time object detection. However, one of the disadvantages of current object detection models is the limitation of objects or the lack of additional data for existing objects. Therefore, labeling data is an important task. This study proposes a new method for labeling object detection in video, which collects from moving cameras using edge devices. Specifically, our method is able to collect sharp resolution frames containing new objects and objects that are mis-detected during the real-time running of AI Edge devices. The application of this solution supports locating new objects and suggests adding data to existing data in a frame/image, which is able to save a lot of time and effort for labeling video data.

Keywords: Video labeling · Object detection · Edge devices · Moving camera.

1 Introduction

Recently, video understanding has received more attention since the availability of several large-scale video datasets [9]. However, annotating large-scale video datasets is cost-intensive and cumbersome. This suggests a need for a semi-automatic annotation method to improve this process [8]. Specifically, the core idea is using an automatic preprocessing method using a neural network to roughly annotate the image before the human review and revision [1]. Nonetheless, video annotation, especially data from moving cameras, is still a challenging issue because of variations in viewpoint, scale, and appearance within the video. Furthermore, Deep Neural Networks (DNN) grow with the complexity, executing the object detection method using on edge devices becomes a challenging problem [7].

** Corresponding Author

This paper presents EdgeLabel, a novel framework for labeling object detection of moving cameras by executing object detection methods in edge devices. Specifically, EdgeLabel obtains better results than using pretrained object detectors and focuses on incorporating object detection methods on edge devices to enable real-time processing. Fig. 1 illustrates the primary process of EdgeLabel. Notably, our method does not attempt to provide a better and general

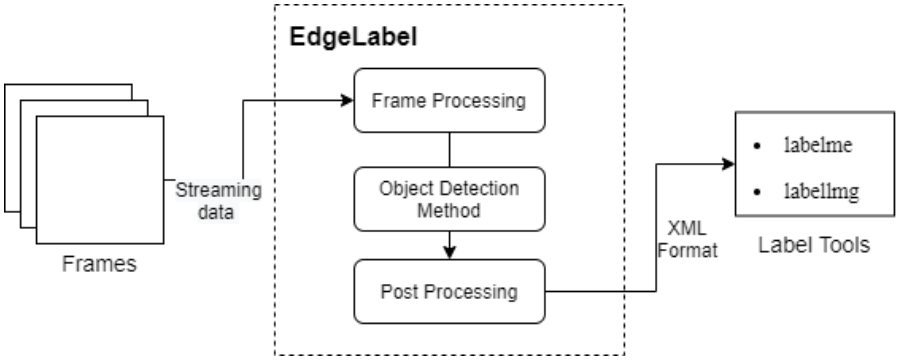

Fig. 1. Modules of EdgeLabel

object detector. Specifically, we capitalize on the redundancy and specificity of individual video streams that empirically perform better on that specific video. The main contribution of this study is twofold as follows:

- We propose a novel semi-automatic annotation method for videos, which are collected from moving camera in terms of efficient time consuming and better performers on specific video.
- The object detection method is executed on edge devices for enabling the real-time processing of streaming data.

The rest of this paper is organized as follows: In Section 2, we take a brief review of image labeling tools and object detection methods. Section 3 presents the main process of the EdgeLabel framework. Section 4 demonstrates preliminary results of the proposed framework on specific video data. Section 5 is the conclusion and future work.

2 Literature Review

2.1 Image Labeling Tools

Current models can label hundreds of thousands of objects, but the number of labeled objects cannot compare with the number of objects in daily life. Therefore, we always need to add new data to artificial intelligence (AI) models related

to object detection. Several well-known tools that support data labeling are labelImg [12], labelme [3] and so on. The inputs of these tools are images or videos. In the case of images, the objects in the image will be zoned and labeled. However, with the input video, the labelling process has to follow several steps, as shown in Fig.2. Specifically, the input video is extracted to many frames. For

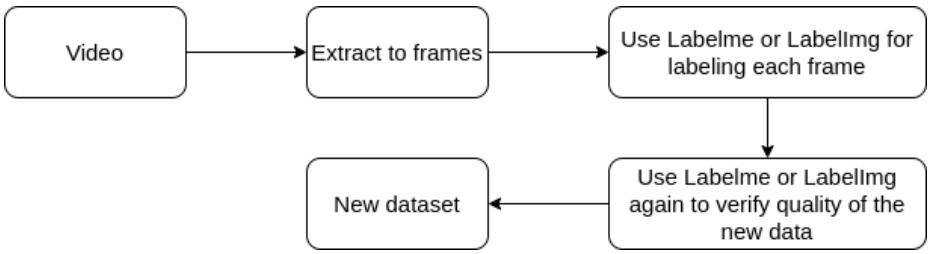

Fig. 2. The pipeline of Video anotation using image labeling tools for object detection

instance, if we want to label 1 hour video with the FPS is 24, we have to spend time considering 86400 frames. However, not all frames have new objects, which are not necessary to spend a lot of time checking all frames. Therefore, we need a solution that retrieves frames containing new data or suggests frames containing objects that may be misdetected. In this regard, Poorgholi et. al introduce t-SNE, a new method to speed up the annotation process by placing the same actions from different videos into two-dimensional space based on feature similarity [8]. Furthermore, authors in [4] proposed an automatic annotation method based on deep learning preprocessing which includes two stage: i) improving the results of CNN processing by combining with the object detection outcome; ii) using a sliding window in order to deal with a large number of outliers. Zhou et. al. [15] combining two complementary sources of knowledge using bounding box merging and model distillation to generate detections on the unlabeled portion.

2.2 Object Detection Method

In this study, we define a new object as an object that is not detected, or misrepresented by using neural network models. Specifically, modern object detection methods integrate feature generation, proposal, and classification into a single pipeline, which fall into two categories: i) single-stage architectures (e.g., YOLO [10] and SSD [6]) which directly predict the coordinates and class of bounding boxes; ii) two-stage detectors (e.g.,Faster R-CNN [11]) refine proposals produced by a region proposal network. Technically, comparing with two-stage methods, single-stage models are less accurate and less expensive, which are suitable for detection and tracking problems [3, 2].

[3] https://github.com/tzutalin/labelImg

Since we focus on executing the object detection process on edge devices, we fine-tune a compact single-stage object detector (i.e., MobileNet-SSD [13]), which is trained on Microsoft COCO [5] to generate labels.

3 EdgeLabel: Overview and Features

EdgeLabel is responsible for identifying and collecting useful frames while the Edge device is running other AI models. Specifically, Alg. 1 illustrates the main process of EdgeLabel framework. Particular, the framework includes three main processes such as frame processing, object detection method, and post processing, which are sequentially described as follows:

3.1 Frame Processing

This process focus on select the input frames. In streaming data of input video, there will be a lot blurred frames. Therefore, we adopt Laplacian method for selecting quality image based on the focus level. The peseudo code of this process is illustrated in Line 3-9 of the Alg. 1.

3.2 Object Detection Method

The selected input frames are put into an object detection model. In this study, we employ SSD MobilenetV2 trained over COCO dataset using Tensorflow API for the object detection. Furthermore, the object detection model is quantized for running real-time with the best performance inside edge devices [14]. After using the object detection model, all object-label and locations of misidentified-objects (low obj-score) are temporarily stored. This process is illustrated in lines 10-11 of the Alg. 1.

Furthermore, one of the main problems in this process is that there are many objects that cannot detect by object detection models. However, they have certain shapes such as triangles, squares, rectangles, and circles. Therefore, we adopt contour, a well-known image processing algorithm to extract meaningful objects. Sequentially, their locations and shape-name are stored for next process, which is expressed in line 12 of the algorithm.

3.3 Post Processing

This process is provided for selecting frames used for the annotation, which is expressed in lines 13-26 of the algorithm. Specifically, after detecting the image quality, misidentified-objects and meaningful objects of the frame in the previous process, the next step is to check whether the select frame is a new frame or not by calculating the structural similarity between two frames. Depends on the speed of the moving camera, we can select correspond threshold, and two frames that are used for structural similarity comparison. The process includes three

Algorithm 1 EdgeLabel framework for labeling video from moving camera

```
cap = CaptureVideo()                                  ▷ Initialize real-time camera
frame-count = 0
check-similarity-background = False
while cap.isOpened() do                    ▷ Check real-time camera is open or not
    frame = cap.read()
  ▷ Initialize a list to store useful frames
    ListFrame = []
  ▷ Initialize a list to location of each object in useful frames
    BBox = []
  ▷ Initialize the list to store pre-label of each object
    Pre-label = []
  ▷ Declare a variable to evaluate the image quality of each frame
    good-frame = Evaluate the image quality of frame.
  ▷ Collect misidentified object of object detection model
    object-label,points,obj-score = ObjectDetectionModel(frame)
    misidentified-object = considering the obj-score < fixed-threshold
  ▷ Beside, the objects is detected by using object detection model, we can evaluate
  the frame contain more new object or not
    Detect meaningful-object of input frame (the shape of objects are trian-
  gles,squares, rectangles, circles) by using contour algorithms.
    new-background = Evaluate the background of the frame is new or not.
    count-down = 5 (get 5 frames)
    if new-background and count-down > 0 then
        if good-frame == True then
            save frame to a path
            ListFrame.append(good-frame)
            BBox.append(location of misidentified-object)
            Pre-label.append(object-label)
            BBox.append(location of meaningful-object)
            Pre-label.append(shape-name of meaningful-object)
            Store ListFrame,BBox,Pre-label to xml file with frame path to DB
        end if
        count-down = count-down - 1
    end if
  ▷ Loading frame and xml file to LabelImg and Labelme. Sequentially, by using
  LabelImage and Labelme, we can re-label new objects easily
end while
```

steps as follows: i) First, we define compare-area inside of the frame for comparing two frames; ii) Then, we calculate the similarities using image processing algorithms. iii) Maximum five consecutive frames are stored with the best image quality, including information of misidentified-objects and meaningful objects. The output is XML files, which follow the formats of LabelImg or LabelMe for the labeling.

4 Demonstration and Results Analysis

We test our framework with an input video that are collected using the Coral Camera, which is a 5MP camera designed for use with the Coral Dev Board. Fig. 3 demonstrates the interface of our test. Specifically, EdgeLabel includes

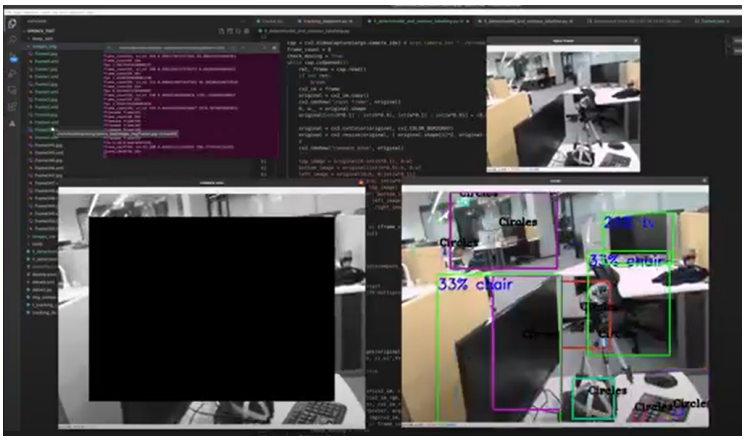

Fig. 3. The test interface of EdgeLabel

three main processes for labeling video data: i) Input data are preprocessed for removing blurred frames; ii) The selected frames are put into an object detection model. Notably, in this study, we employ SSD MobilenetV2 on edge devices to provide the real-time processing; iii) lastly, the post-processing is provided

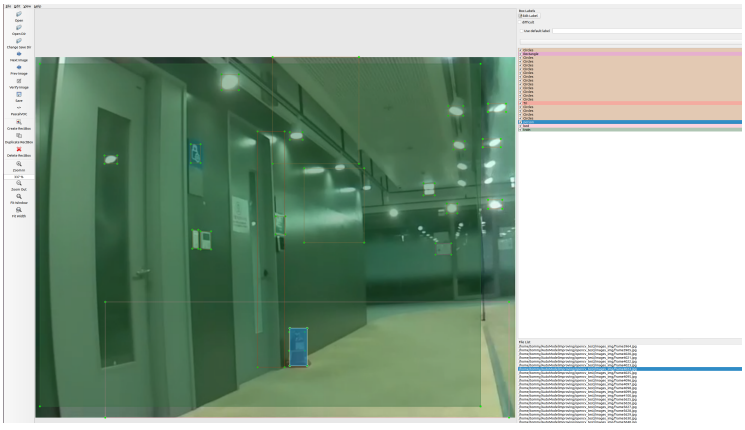

Fig. 4. Labeling process using LabelImg with extracted frames

for extracting frames, which include misidentified-objects for the annotation. Sequentially, the output are xml files, which are input into labeling tools such as labeImg or LabelMe for the annotation, as shown in Fig. 4.

Tab. 1 shows the results of our framework for the input video in which the video lengths equal 3m30s, and the speed of moving camera is around 3km/h. Particularly, we test the input video with various values of distance between two

Table 1. Results of EdgeLabel for the input video. Bold text are the appropriate results.

Total Frames	Frame Distance	Threshold	Extracted Frame	Mis-Objects
6286	10	90%	486	4224
	10	87%	108	865
	10	85%	12	78
	50	**85%**	**69**	**533**

frames to calculate the similarity and the threshold to extract frames for labeling. Accordingly, with the distance between two frame is 50 and the threshold is 85%, there is 69 frames including 533 mis-objects for the annotation. Note that the values of frame distance depend on the speed of moving the camera to get appropriate results.

5 Conclusion and Future work

In this paper, we present LabelEdge, a semi-automatic video annotation method, which focus on the video inputs are collected from moving camera. Specifically, the framework includes three main processes as i) frame processing for selecting the quality frame and remove redundant objects; ii) object detection method with a lightweight model (i.e., MobilenetV2) with SSD for identifying misidentified and unknown objects; and iii) post-processing for selecting frames that used for the annotation. In particular, LabelEdge enables real-time labeling processing by executing the object detection model on edge devices. Regarding the future work of this study, we are trying to exploit the capability of the proposed framework by providing a large-scale dataset of moving cameras for object detection using labeled. Furthermore, more studies on the quantization approach of object detection methods on edge devices for improving performance are also taken into account.

6 Acknowledgement

This research is funded by University of Transport Technology (UTT) under grant number ĐTTĐ2021-06

References

1. Berg, A., Johnander, J., de Gevigney, F.D., Ahlberg, J., Felsberg, M.: Semi-automatic annotation of objects in visual-thermal video. In: Proceeding of the IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019. pp. 2242–2251. IEEE (2019). https://doi.org/10.1109/ICCVW.2019.00277
2. Bui, K.N., Yi, H., Cho, J.: A multi-class multi-movement vehicle counting framework for traffic analysis in complex areas using cctv systems. Energies **13**(8), 2036 (2020). https://doi.org/10.3390/en13082036, https://www.mdpi.com/1996-1073/13/8/2036
3. Bui, K.N., Yi, H., Cho, J.: A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In: Proceeding of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020. pp. 2466–2474. Computer Vision Foundation / IEEE (2020). https://doi.org/10.1109/CVPRW50498.2020.00297
4. Jin, Y., Li, J., Ma, D., Guo, X., Yu, H.: A semi-automatic annotation technology for traffic scene image labeling based on deep learning preprocessing. In: Proceeding of the IEEE International Conference on Computational Science and Engineering, CSE 2017. pp. 315–320. IEEE Computer Society (2017). https://doi.org/10.1109/CSE-EUC.2017.63
5. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Proceeding of 13th European Conference on Computer Vision - ECCV 2014. Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer (2014). https://doi.org/10.1007/978-3-319-10602-1_48
6. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: Proceeding of the 14th European Conference on Computer Vision, ECCV 2016. Lecture Notes in Computer Science, vol. 9905, pp. 21–37. Springer (2016). https://doi.org/10.1007/978-3-319-46448-0_2
7. Marchisio, A., Hanif, M.A., Khalid, F., Plastiras, G., Kyrkou, C., Theocharides, T., Shafique, M.: Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges. In: Proceeding of the IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2019. pp. 553–559. IEEE (2019). https://doi.org/10.1109/ISVLSI.2019.00105
8. Poorgholi, S., Kayhan, O.S., van Gemert, J.C.: t-eva: Time-efficient t-sne video annotation. In: Proceeding of the International Workshops and Challenges Pattern Recognition, ICPR 2021. Lecture Notes in Computer Science, vol. 12664, pp. 153–169. Springer (2020). https://doi.org/10.1007/978-3-030-68799-1_12
9. Real, E., Shlens, J., Mazzocchi, S., Pan, X., Vanhoucke, V.: Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. pp. 7464–7473. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.789
10. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016. pp. 779–788. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.91
11. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031

12. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: A database and web-based tool for image annotation. Int. J. Comput. Vis. **77**(1-3), 157–173 (2008). https://doi.org/10.1007/s11263-007-0090-8
13. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00474
14. To, H., Bui, K.N., Le, V., Bui, T., Li, W., Cha, S.: Real-time social distancing alert system using pose estimation on smart edge devices. In: Proceeding of the 13th Asian Conference in Intelligent Information and Database Systems, ACIIDS 2021. Communications in Computer and Information Science, vol. 1371, pp. 291–300. Springer (2021)
15. Zhou, G., Dulloor, S., Andersen, D.G., Kaminsky, M.: EDF: ensemble, distill, and fuse for easy video labeling. CoRR **abs/1812.03626** (2018), http://arxiv.org/abs/1812.03626