UDC 004.75

VIDEO OBJECT DETECTION PROGRAM DESIGN UNDER TWO DIFFERENT CLIENT-SERVER ORGANIZATIONS

H. GAO, O.G. SHEVCHUK

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus

Received February 17, 2023

Abstract. The article presents video object detection program design under two different client-server organizations based on YOLOX model. It introduces the N-tier architecture and shows how to design programs through sequence diagrams based on client-server organization cases. And the processing process of the target detection part of the program is given. The test results show that the program design can be applied to Android system and Web scene, and analyze the actual application conditions.

Keywords: software design, object detection, image preprocessing, model inference.

Introduction

Object detection is a classic task in the field of computer vision, and it is the basic premise for advanced vision tasks such as scene content analysis and understanding. The target detection task in the video is closer to the needs of real life. In real life, intelligent video surveillance, robot navigation and other application scenarios need to process the video and detect the target in the video. The target detection in the video needs to deal with the various changes of the target due to the movement on the basis of the static image target detection, which is the difficulty. With the development of deep learning, deep convolutional neural networks are rapidly applied to every field of computer vision, and have made relatively great progress compared with traditional methods. In the context of deep convolutional networks, YOLO is an advanced single-stage target detection algorithm. It has undergone the evolution of version 1 ~ version 4, and has developed to YOLOX that does not rely on anchor boxes. The proposal of YOLO aims to improve the detection efficiency of target detection, trying to make target detection reach the level of real-time detection. This article proposes two implementations of video object detection based on the YOLOX model under different client-server organizations, uses UML diagrams to describe the functions of the modules, and explains how to infer with the model. In order to verify the effectiveness, the program is implemented according to the design and tested in a real network environment.

N-tier architecture understanding

In the client-server model, a client is a piece of computer hardware or software that accesses services provided by a server as part of a computer network client-server model. The server is usually (but not always) on another computer system, in which case the client accesses the service over the network. A client can be any device – computer, tablet or mobile phone. Thus, client-server represents the relationship between collaborating programs in an application, consisting of a client that initiates a request for a service and a server that provides that function or service. There are three main categories of client-server:

1. One-tier architecture. It is the simplest one as it is equivalent to running the application on the personal computer. All of the required components for an application to run are on a single application or server.

2. Two-tier architecture. It consists of a client, a server, and a protocol that connects the two layers. The GUI code resides on the client host, and the domain logic resides on the server host. Domain

logic or business logic is the part of a program that encodes real-world business rules that determine how data is created, stored, and changed. Business logic should be distinguished from business rules. Business logic is the part of an enterprise system that determines how data is transformed or calculated, and how it is routed to people or software. Business rules are the formal expression of business policy. Anything that is a process or a procedure is business logic, and anything that is neither a process nor a procedure is a business rule. For example, welcoming a new visitor is a process consisting of steps to be taken, while saying that every new visitor must be welcomed is a business rule. In addition, business logic is procedural while business rules are declarative.

3. Multitier architecture (often referred to as an N-tier architecture) is a client-server architecture in which presentation, application processing, and data management functions are physically separated. The most widely used multi-tier architecture is the three-tier architecture. Three-tier architecture is a client-server software architectural pattern in which the user interface (presentation layer), application (functional process logic), computer data storage and data access (data layer) are developed and maintained as independent modules, usually on a different platform.

For multi-tier architecture, it offers the possibility to physically distribute client-server applications across multiple machines. For client-server organization, there is the simplest organization: a client machine containing only the programs implementing the user-interface level; a server machine containing the rest, that is the programs implementing the processing and data level. In addition, there are alternative client-server organizations, shown in Figure 1.

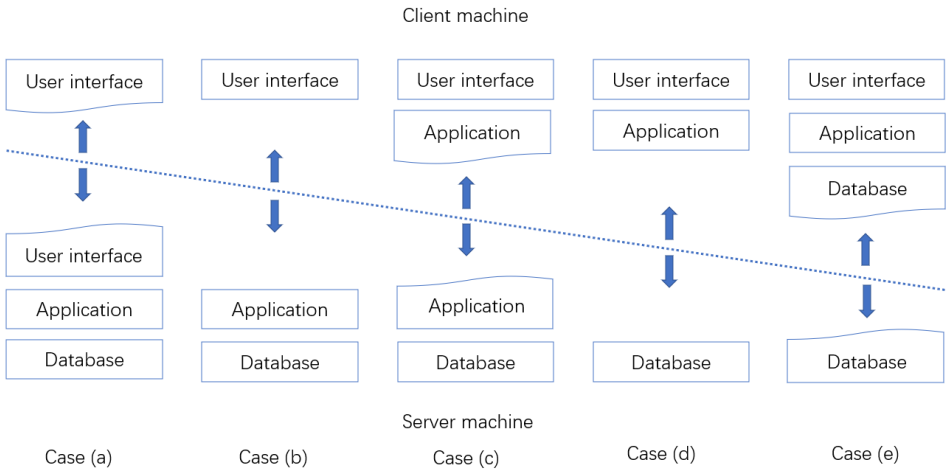

Figure 1. Alternative client-server organizations

Video object detection program design under different client-server organizations

Based on alternative client-server organization, video object detection program will be design under Case (b) and Case (c) respectively (shown in Figure 1). For Case (b), since the client only provides the user interface, the main function of the user interface is to upload videos, and the processing logic related to video processing and object detection is completely handed over to the server. For Case (c), since the client not only provides the user interface, but also provides some application processing logic, the client is mainly responsible for uploading video and video processing logic, and the processing related to object detection is handed over to the server.

1. Video object detection program design under Case (b)

Based on client-server organization Case (b), all processing logic is placed on the server side, and the client only provides interfaces for uploading and playing videos. The video object detection program under Case (b) is designed in the scenario where the user accesses the browser. Figure 2 shows the interactive process of realizing the video object detection function through a sequence diagram. In Figure 2, Browser represents User interface that provides the function of uploading video and playing video. Server and Live Video Server represent Application that provides all processing logic for video object detection, including video segmentation, object detection of frames, real-time transmission of frames by streaming. Database mainly records the processed video information, such as processing time, frame size, frame rate, etc.

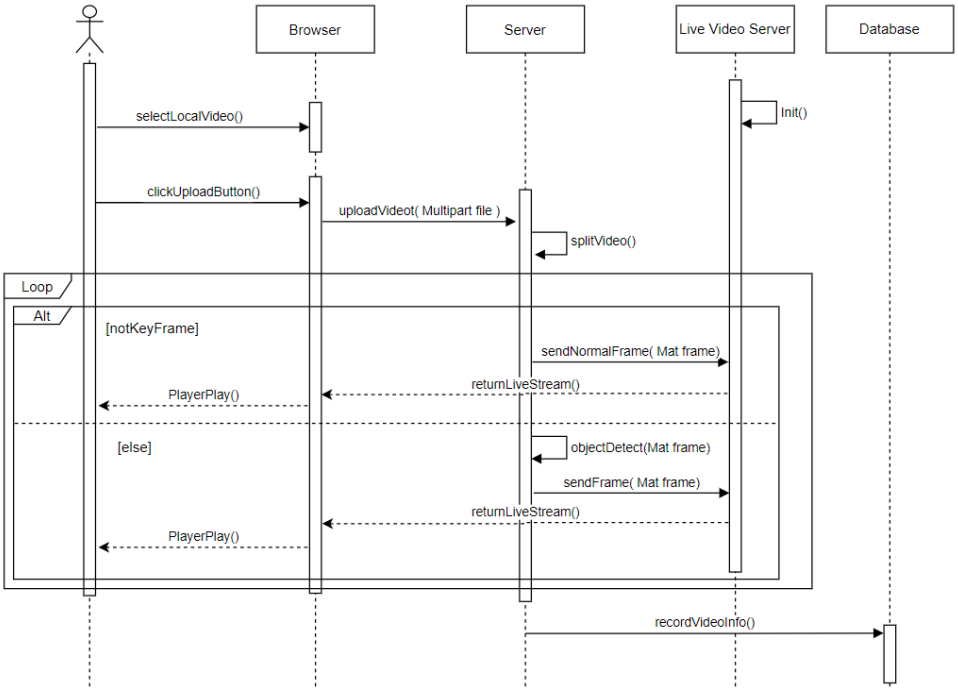

Figure 2. Sequence diagram of Case (b)

2. Video object detection program design under Case (c)

Based on client-server organization Case (c), the client side mainly includes interfaces for uploading and playing videos, video processing logic and frame processing logic, while the server side mainly handles frame object detection. The video object detection program under Case (c) is designed for mobile devices. Users can access the server through mobile devices to request image object detection, and cooperate with the processing logic on the mobile device to complete video object detection. Figure 3 shows the interactive process of realizing the video object detection function through a sequence diagram. In Figure 3, APP represents User interface and Application of client side that provides the functions of local video selection, video segmentation, key frame selection, image processing, composite video and video playback. Server represent Application of server side that provides image object detection. Database mainly records the processed image information, such as processing time, image size, etc.

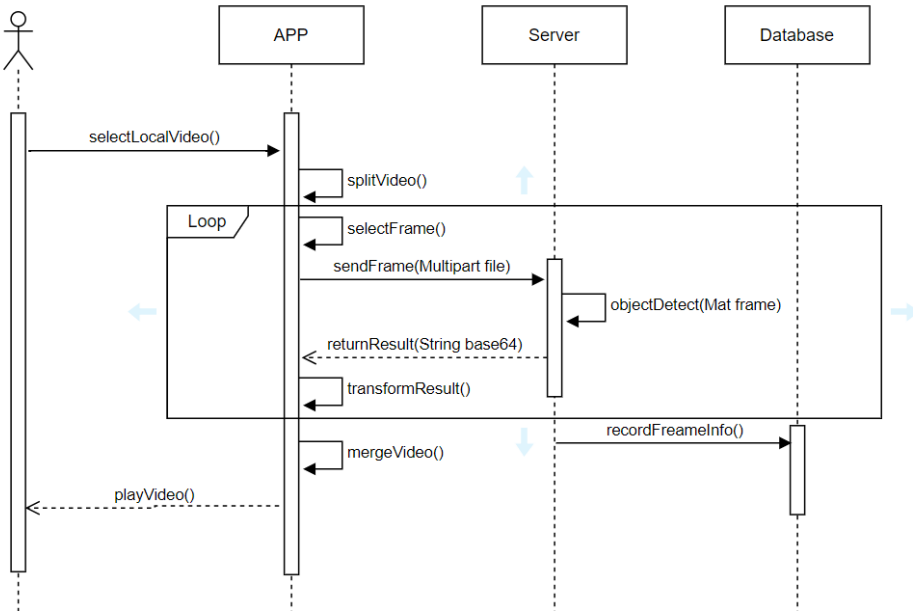

Figure 3. Sequence diagram of Case (c)

3. The process of image object detection

In the process of video object detection program design, it involves object detection of images (object detection of frames). The image object detection method based on YOLOX mainly involves image preprocessing, output decoding and the strategy for choosing prediction boxes. The process of image object detection mainly includes the following steps:

Step 1. Image transformation and normalization.

The purpose of transforming and normalizing the images is to obtain images that satisfies the input format of the YOLOX model. The YOLOX model requires the size of the input image to be 640×640 pixels. According to the affine transformation theory, image transformation can be done with the help of transformation matrix M. For image normalization, using the mean and std of ImageNet is a common practice. All models expect input images normalized in the same way, mini-batches of 3-channel RGB images of shape (3×H×W), where H and W are expected to be at least 224. The images have to be loaded in to a range of [0, 1] and then normalized using mean with [0,485; 0,456;0,406] and std with [0,229; 0,224; 0,225].

Step 2. Decode outputs, generate proposals.

The output of the model is the tensor of 8400×85 elements, meaning that the model predicts 8400 prediction boxes and every prediction box has 85 properties. Among the 85 properties, the first five properties represent the horizontal and vertical coordinates of the center point of the prediction box, the length and width of the prediction box, and the probability of objects in the prediction box. The remaining 80 attributes represent the object category probability (the model can judge 80 object categories, and each object category will correspond to a probability).

For an image, there are often only a few objects that need to be identified. However, there are 8400 boxes based on the output data. Obviously, the prediction boxes need to be further selected. In this step, it is necessary to manually set threshold to generate proposals with more accurate prediction boxes.

Before manually setting the threshold, it is necessary to know that the output 8400 prediction boxes are merged from three different sizes of anchors. Because in the decoding process, the coordinates of the prediction boxes are related to the corresponding anchor. Among the 8400 prediction boxes, the anchor size corresponding to 6400 prediction boxes is 8×8. This means that the original image of size 640×640 is divided into 80×80 anchors with stride of 8. Similarly, the anchor size corresponding to 1600 prediction boxes is 16×16, and the anchor size corresponding to 400 prediction boxes is 32×32.

Through the anchor, the coordinates of the prediction box can be obtained, and the probability threshold is further set manually. Here the probability threshold is set to 0,6. Then the confidence C that each prediction box has the corresponding category of objects can be given

$$C = p \cdot \max(class\ probability) \tag{5}$$

where p represents the probability that there is an object in the predicted box. Class probability represents the probability of being that class. Then if the confidence C is greater than 0,6, save the proposal (the coordinates of the prediction box) and the corresponding the confidence C. After generating the proposals, the possibly correct prediction boxes will be saved.

Step 3. Perform non-maximum suppression.

The purpose of performing the non-maximum suppression algorithm is to eliminate redundant overlapping boxes with lower confidences.

Step 4. Accomplish object detection.

According to the affine transformation theory, based on the transformation matrix M^{-1} from Image transformation, the coordinates of the final boxes can be converted to the coordinates in the original image. It accomplishes the object detection on the original image.

Test Result

The developed method of image object detection is implemented in C++ using library of OpenCV 4.5.4 and the part of splitting video is implemented using library of FFmpeg. For the program designed based on Case (b), the front-end program is developed based on Vue3 and deployed on Amazon Elastic Compute Cloud (EC2) together with the back-end server. For the program designed based on Case (c), the front-end application is developed on the Android system, and the back-end server is

deployed on EC2. EC2 is physically located in Frankfurt. The EC2 has the following specifications: vCPU: 1 core; Memory: 1GB; for bandwidth, the available network bandwidth of an instance depends on the number of vCPUs it has.

Examples of test result are shown in Figure 4 and Figure 5.

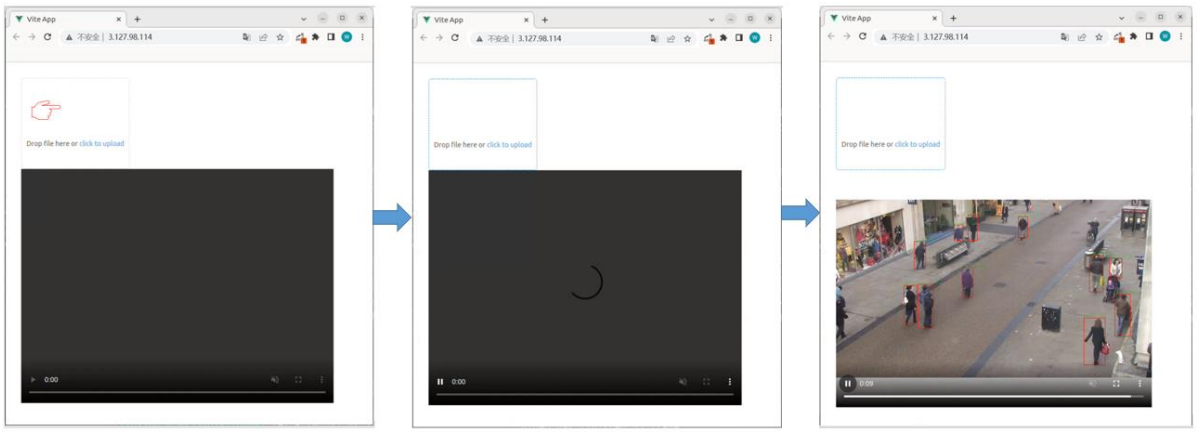

Figure 4. Test Result of Case (b)

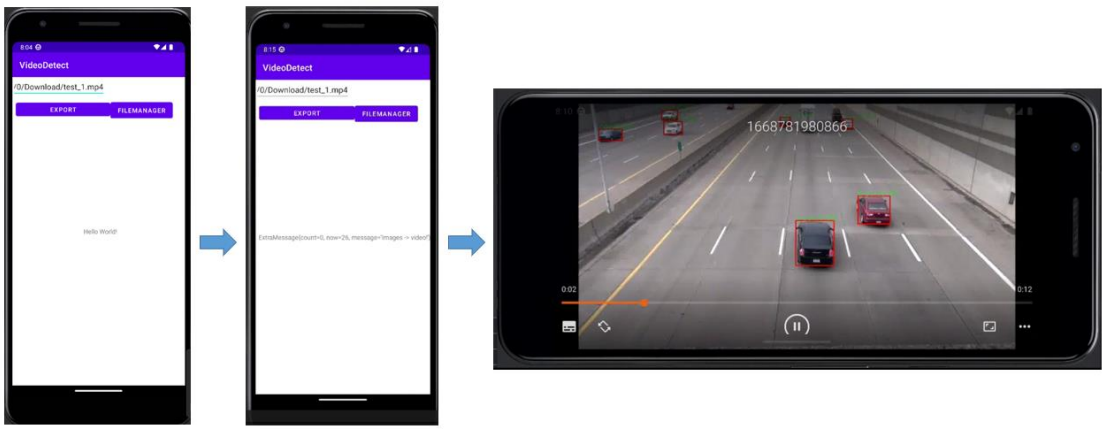

Figure 5. Test Result of Case (c)

In Figure 4, video object detection program is designed based on client-server organization Case (b). The user can open the browser to upload the video, wait for the buffering of the live stream, and after some data is processed in the background, the user can watch the result of video object detection. In Figure 5, video object detection program is designed based on client-server organization Case (c). Through the application on the mobile phone, the user can select a local video, or input the path of the video, and then clicks the button. The application starts to process the video and interacts with the background server to complete object detection, and finally synthesizes the video and plays it.

For Case (b), two sets of tests are carried out. The test object metadata is given in Table 1 and the test data are shown in the Table 2.

Table 1. **Metadata**

Video ID	Resolution, pixel	Frame Rate, FPS	BPS, kb/s	Size, Mb
1	1280×720	30	9818	39
2	1280×720	30	4321	17

Table 2. **The test data**

Video ID	Upload Time, s	Wait Response Time, s	Steam Content, Mb	Transmission Time, s
1	30,56	1,63	10,1	174
2	7,46	1,43	11,4	168

In table 2, Upload Time represents the upload time from client to server. Wait Response Time represents the time from when the server returns the message of receiving the uploaded video to when it starts to receive streaming data. Steam Content represents the size of the stream data received. Transmission Time represents the time to receive streaming data. For Upload Time, it depends on the

local upstream bandwidth. If the upstream bandwidth is smaller, the upload time will be longer. For Wait Response Time, during this period, the server needs to pre-process the first few frames, and then start streaming. For Steam Content and Transmission Time, they not only depend on the local bandwidth, but also depend on the upstream bandwidth of the server. Because the local bandwidth is 10M, the general network speed is 1,25 Mb/s. However, the test data does not meet expectations. The main reason is that the actual uplink speed of the server is too low. The part of real-time monitoring data of the server network speed is shown in the Figure 6.

Time	lo		eth0	
HH:MM:SS	KB/s in	KB/s out	KB/s in	KB/s out
19:01:36	0.00	0.00	105.46	204.65
19:01:37	0.00	0.00	24.72	47.45
19:01:38	0.00	0.00	86.74	168.45
19:01:39	0.00	0.00	56.77	109.53
19:01:40	0.00	0.00	28.02	54.51
19:01:41	0.00	0.00	61.47	118.79
19:01:42	0.00	0.00	65.35	126.04
19:01:43	0.00	0.00	52.00	99.58
19:01:44	0.00	0.00	50.13	97.26
19:01:45	0.00	0.00	81.20	156.07
19:01:46	0.00	0.00	7.95	16.15
19:01:47	0.00	0.00	57.97	111.48
19:01:48	0.00	0.00	97.50	188.95
19:01:49	0.00	0.00	17.52	34.01
19:01:50	0.00	0.00	99.33	192.20
19:01:51	0.00	0.00	68.11	132.00
19:01:52	0.00	0.00	41.64	80.95
19:01:53	0.00	0.00	113.67	219.96
19:01:54	0.00	0.00	38.09	73.07
19:01:55	0.00	0.00	64.28	125.32
19:01:56	0.00	0.00	101.81	196.64

Time	lo		eth0	
HH:MM:SS	KB/s in	KB/s out	KB/s in	KB/s out
19:03:00	0.00	0.00	52.31	101.05
19:03:01	0.00	0.00	48.93	94.57
19:03:02	0.00	0.00	56.64	109.04
19:03:03	0.00	0.00	73.63	142.28
19:03:04	0.00	0.00	26.86	52.89
19:03:05	0.00	0.00	32.48	61.57
19:03:06	0.00	0.00	142.48	276.33
19:03:07	0.00	0.00	70.49	136.40
19:03:08	0.00	0.00	32.13	62.74
19:03:09	0.00	0.00	110.47	214.26
19:03:10	0.00	0.00	52.47	102.53
19:03:11	0.00	0.00	58.75	112.22
19:03:12	0.00	0.00	74.14	143.50
19:03:13	0.00	0.00	29.33	57.85
19:03:14	0.00	0.00	59.72	114.31
19:03:15	0.00	0.00	114.18	221.43
19:03:16	0.00	0.00	26.20	50.65
19:03:17	0.00	0.00	110.90	215.15
19:03:18	0.00	0.00	76.73	147.93
19:03:19	0.00	0.00	27.85	54.99
19:03:20	0.00	0.00	120.19	233.06

Figure 6. Server real-time network speed monitoring data

In Figure 6, eth0 represents the network card of the server, and "in" and "out" represents download speed and uplink speed respectively. It can be seen from the Figure 6 that the uplink speed of the service is too low and unstable, which directly causes the client to freeze or drop frames when playing video. This degrades the experience of watching live.

For Case (b), the upload network speed of the client and the download network speed of the server determine the start time of the user waiting for the live broadcast. The download speed of the client and the upload speed of the server determine the user experience of watching the object detection video. For users, 10M bandwidth is enough to upload video and watch live broadcast. However, if the upstream bandwidth of the server is not large enough, it will directly cause the live video to freeze or drop frames. In this case, Case (b) is not recommended. For Case (c), the network transmission part only involves the transmission of images. The requirement for server bandwidth is not as high as Case (b). Therefore, when the bandwidth of the client and the bandwidth of the server cannot be guaranteed, Case (c) can be selected. In this case, the waiting time may be too long, but this case is more general with less material hardware and bandwidth requirements.

Conclusion

Under two client-server organization cases, two designs of video object detection programs are proposed. One is to put all the processing logic of video object detection on the server side, and the client only provides the interface; the other is to put only the function of object detection on the server side, and other video processing logic on the client side. Then two designs are implemented and tested by building a cloud server. The result is that the proposed program design can complete video object detection. In addition, the use conditions of the two designs are analyzed through the test data.

References

1. Jing J, Helal A. S., Elmagarmid A. // ACM Computing Surveys. 1999. Vol. 31. P. 117-157.
2. Redmon J., Divvala S., Girshick R., et al. You Only Look Once: Unified, Real-Time Object Detection. J. 2016.
3. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. J. 2018.
4. Ge Z., Liu S., Wang F., et al. YOLOX: Exceeding YOLO Series in 2021. J. 2021.
5. Neubeck A., Gool L. Efficient Non-Maximum Suppression. C. 2006.
6. Stearns C., Kannappan K. Method for 2-D affine transformation of images. P. 1995.