I. J. Engineering and Manufacturing, 2026, 3, 345-364
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijem.2026.03.21

User Object of Interest Based Video Clip Extraction using Pretrained YOLOv7

Md. Mahmudul Hasan
Computer Science and Engineering Discipline, Khulna University, Khulna, 9208, Bangladesh
E-mail: mahmudul2029@cseku.ac.bd
ORCID iD: https://orcid.org/0009-0007-5848-6647

Md. Titas Ahmmed
Computer Science and Engineering Discipline, Khulna University, Khulna, 9208, Bangladesh
E-mail: adtitas.me@gmail.com
ORCID iD: https://orcid.org/0009-0001-2485-4399

Md. Farhan Sadique*
Computer Science and Engineering Discipline, Khulna University, Khulna, 9208, Bangladesh
E-mail: farhan@cse.ku.ac.bd
ORCID iD: https://orcid.org/0000-0002-7454-0638
*Corresponding Author

Received: 02 April, 2026; Revised: 28 April, 2026; Accepted: 20 May, 2026; Published: 08 June, 2026

Abstract: Video clip extraction is the process of generating shorter, focused video segments by identifying and retaining frames that contain a user-specified object of interest (UOoI). Such targeted extraction allows users to access relevant portions of a video without watching the entire recording, with practical use in surveillance review, content management, and educational settings. In this work, we present an object-conditioned video clip extraction framework that uses the pretrained YOLOv7 detector to perform frame-level analysis of an input video. For each frame, the detector produces a set of class labels, which are matched against the user-selected UOoI to produce a binary per-frame detection signal. A one-dimensional temporal-window voting filter is then applied to this signal to suppress isolated false positives and recover isolated false negatives, addressing the single-frame detection noise that produces visible discontinuities in naive frame-by-frame approaches. The voted-positive frame indices are mapped back to source timestamps, and the corresponding audio segments are extracted directly from the source video using ffmpeg, preserving the original audio track in the output clip. The framework uses a dictionary of 80 object categories drawn from the MS COCO label set, and a graphical user interface allows users to select an input video, choose a target object, preview the input, and obtain the extracted clip with audio. We evaluate the framework on the SumMe benchmark, which we re-annotated at the frame level for object presence, and on a newly annotated set of 39 videos collected from public sources. Both datasets were independently labelled by two annotators, with Cohen's kappa of 0.85 and 0.83, respectively, and disagreements resolved by a third adjudicator. At the default voting configuration of W=5, K=3, the framework attains an F1-score of 70.88% with 90.12% accuracy on SumMe and an F1-score of 69.89% with 85.13% accuracy on the custom dataset. An ablation over voting parameters shows monotonic gains on SumMe across the full sweep, and a smaller, dataset-dependent gain on the custom set. We discuss the remaining limitations of the pipeline, including single-UOoI conditioning, dependence on the MS COCO label vocabulary, and abrupt transitions between non-adjacent extracted segments, and outline directions for addressing them. abstract.

Index Terms: User object of interest, Video clip extraction, Pretrained yolov7, temporal-window voting, Object detection in video, video shortening

1. Introduction

The volume of video data produced and consumed today has grown to a scale where manually locating relevant content within long recordings is impractical. Surveillance footage, lecture recordings, sports broadcasts, online tutorials, and social-media uploads routinely run for hours, while users typically need only the segments in which a particular object, person, or event of interest appears. This mismatch between video length and user attention motivates

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

the problem of video clip extraction: producing a shorter video that retains only the segments of an input video relevant to a user-specified criterion. We focus on object-conditioned clip extraction. Given an input video and a target object label — which we refer to as the User Object of Interest (UOoI) — the goal is to output a shorter video composed of the frames in which the UOoI is present, preserving their original temporal order. This formulation is distinct from generic video summarization, which produces a representative overview of the entire video without reference to user intent, and from query-based video retrieval, which returns time-stamps rather than a playable clip. The output of our framework is a dynamic clip (a playable video segment), in contrast to static representations such as keyframe collages or textual summaries. Dynamic output preserves motion information and is directly usable in downstream review tasks (e.g., a security operator scanning for a specific vehicle, or a student isolating laboratory-equipment usage from a long lecture). We adopt YOLOv7 [1] as the per-frame object detector. The choice is driven by its strong speed–accuracy trade-off on MS COCO [17] and by the availability of pretrained weights covering 80 everyday object categories, which we expose to the user as the Dictionary of Objects (DoB). Frame-level detections are matched against the UOoI label, and matched frames are concatenated chronologically to produce the output clip.

1.1. Problem Formulation

Let an input video v_i consist of n ordered frames $FRAMES_n = \{f_1, f_2, f_3, f_4, \ldots\ldots, f_n\}$ and let the user select a target object $UOoI \in DoB$. The task is to produce a subset $FRAMES_m = \{f_{j1}, f_{j2}, f_{j3}, f_{j4}, \ldots\ldots, f_{jm}\} \subseteq FRAMES_n$ with $0 \leq m \leq n$ and $j_1 < j_2 < j_3 < \ldots\ldots < j_m$ such that every f_{jk} contains at least one instance of the UOoI as determined by the detector. The output clip is formed by concatenating these frames in their original temporal order.

1.2. Contributions

The contributions of this work are:

- A frame-level, object-conditioned video clip extraction pipeline built on pretrained YOLOv7, exposing 80 MS COCO [17] categories as user-selectable targets.
- Frame-level annotations, with respect to MS COCO object categories, of 22 videos from the SumMe [18] benchmark, which was originally released with shot-level summary annotations rather than object-presence labels.
- A new dataset of 39 videos collected from public sources, annotated at the frame level for object presence, intended to support evaluation of object-conditioned extraction methods.
- A desktop graphical user interface that allows users to select an input video, choose a UOoI from the DoB, preview the input, and obtain the extracted clip.
- A temporal-window voting step that suppresses isolated single-frame detections, improving clip continuity over naive frame-by-frame filtering.

1.3. Scope and Applications

The framework is applicable in any setting where a user wants to isolate appearances of a known object class within long-form video. Examples include surveillance review (locating segments containing specific vehicle types), educational content (extracting the segments of an instructional video that show a particular instrument or apparatus), wildlife and underwater footage analysis (isolating the appearances of a target species), and sports footage (isolating segments containing a specific piece of equipment). The pipeline assumes that the target object class is part of the detector's training vocabulary; behavior for out-of-vocabulary targets is outside the scope of this work.

The remainder of the paper is organised as follows. Section 2. reviews related work in object-conditioned video summarization and positions our approach with respect to prior pipelines. Section 3. details the methodology. Section 4. describes the datasets, evaluation protocol, and experimental results. Section 5. presents the graphical user interface. Section 6. discusses limitations and directions for future work.

2. Literature Review

Today, the field of video-related work based on user object of interest (UOoI) is gaining significant popularity. Hence, different types of approaches to the video processing field task have been made since the start of video processing using UOoI works [3]. Though each method differs from the other's works in some perspective, these works used UOoI with different names. There are different types of video processing tasks in a broad sense. Firstly, Non-interest-oriented video processing refers to methods that, while processing a video in a specific procedure, do not incorporate any form of user interest, such as an object or micro-questions in the text. This type of video processing only focuses on the video as a whole. A lot of work falls under this broad category as compared to interest-oriented video processing. There are several approaches that are made by researchers, such as supervised, unsupervised and reinforcement learning based [4, 5, 6, 7, 8, 9, 10, 11, 12, 13].

Secondly, interest-oriented video processing is in the sense that while processing the video, the methods include user- interested objects or give the user the opportunity to ask micro queries as text. This type of video processing

processed video with the higher priority of user-given interest. Very few works fall under this category as compared to the non- interest-oriented video processing [2, 3, 14, 15].

All of these works only involve interest-oriented video processing work, which is described below as our method that falls under that category. We described some of the works that go with UOoI. Those works actually do video summarization, not video clip extraction. Video summarization is a process where the result gives the whole overview of that video without watching it, whereas video clip extraction is a process that cuts the original video based on a specific object and accumulates those parts, and gives another video with shorter, more effective parts. These remarkable video summarization approaches are described below:

This paper by [3] introduces a novel method for video summarization that emphasizes user-specific interests by incorporating an Object of Interest (OOI) into the summarization process. Addressing the limitations of traditional video summarization methods, the authors propose a system that generates dynamic, interest-oriented summaries for diverse applications such as surveillance, sports highlights, and video storage optimization. Unlike conventional approaches, which often fail to cater to user-specific preferences, this methodology combines keyframe extraction and object re-identification to produce summaries tailored to individual user requirements.

The proposed system employs two parallel pipelines for summarization. The first pipeline (Pipe-1) segments the video into shots using colour histogram-based techniques and extracts static and dynamic features to generate a generalized keyframe summary. The second pipeline (Pipe-2) identifies frames containing the OOI using a deep learning-based Siamese network architecture for object comparison. The final summary is derived from the intersection of outputs from both pipelines, ensuring content representation and user interest alignment.

Key contributions of the paper include a colour histogram-based video segmentation technique for generalized summarization with a supervised deep learning model for OOI-based frame selection using a Siamese network architecture. This methodology also includes combining generalized and interest-specific summaries to support user-specific preferences. The approach is notable for its adaptability across multiple domains, offering solutions for applications like anomaly detection in surveillance, personalized sports video highlights, and efficient video content management. Additionally, the use of supervised and unsupervised learning techniques and dynamic content analysis positions this method as an advancement over existing single-summary systems.

Despite its innovation, the method's reliance on pre-defined OOIs and supervised learning raises concerns about scalability and generalizability. Future work could explore incorporating more dynamic features such as actions or behaviours, skimming-based summaries, and audio analysis for enhanced summarization accuracy.

This paper [15] proposes a robust framework for video summarization using deep learning techniques tailored to user- selected objects of interest (OoIs). The proposed framework addresses challenges such as computational efficiency and personalized content extraction in the context of the massive data produced by surveillance systems and other video sources. The study combines object detection with video summarization to provide meaningful summaries tailored to user-defined requirements.

Key contributions of the paper include a dictionary-based approach for selecting OoIs, allowing users to define specific objects for summarization while filtering out irrelevant noise with the framework employs the YOLOv3 [16] deep learning model, known for its speed and accuracy, to detect objects in video frames effectively. These papers also give a stepwise algorithm that ensures that only frames containing the OoI are retained and combined into a summarized output.

The framework demonstrated robust performance across diverse scenarios, such as lectures, news, documentaries, and surveillance footage. The approach is particularly noted for its ability to detect static and dynamic objects, ensuring relevance in applications like anomaly detection, personal video highlights, and efficient data management.

However, the reliance on YOLOv3 [16] introduces limitations related to detecting small objects in low-quality videos, as highlighted in the paper. Furthermore, the framework's dependence on predefined object categories may restrict flexibility in real-world applications requiring broader generalization. This paper [14] presents an advanced video summarization technique tailored to user-defined Objects of Interest (OoIs). The framework addresses the challenges of processing large-scale video data generated by surveillance systems and other sources, focusing on customized summaries for improved usability and efficiency. Leveraging YOLOv3 [16] for object detection, the method demonstrates significant advancements in accuracy, processing speed, and adaptability. Key contributions of the study include a customizable approach allowing users to specify objects (e.g., cars, bicycles, persons) as points of interest. The methodology employs a dictionary of 80 classes based on the MS COCO [17] dataset for defining potential OoIs. Utilizing the speed and accuracy of YOLOv3 [12], the framework efficiently detects and tracks user-specified objects across video frames. Its single-stage detection mechanism enhances processing capabilities for real-time applications.

Frames containing user-defined objects with a confidence score above a specified threshold are identified and retained to form a concise video summary.

The proposed system also includes a graphical user interface (GUI) for intuitive user interaction, allowing easy selection of input videos and objects of interest. The evaluation metrics, including precision, recall, F1 score, and overall accuracy, highlight the framework's superiority over existing techniques.

Despite its strengths, the reliance on YOLOv3 [16] introduces limitations in detecting small objects in low-

resolution or noisy video frames. Furthermore, the fixed dictionary of object classes may restrict the system's generalizability in scenarios involving undefined or emerging object categories.

From the above discussion, we can conclude that Interest-oriented video processing tasks outperforms non-interest-oriented methods by tailoring the summary to user-specific preferences, significantly enhancing the relevance and usability of the output. These studies are closely aligned with ongoing research in user-specific video clip extraction process and provide a solid foundation for further exploration in interest-based content extraction, making it highly relevant to the thesis topic of user-object-oriented video clip extraction.

3. Methodology

This section will discuss our video clip extraction procedure with the given UOoI in detail. The whole procedure is described step by step. Each step is described in each sub-section. At the end, we provide an algorithm of the entire procedure:

3.1. Method Overview

A short overview of our proposed method is shown in the Fig.1. There is a dictionary of objects from where user choose an object by which video clip to be extracted. Let that user chosen object named as UOoI (User object of Interest). At the same time user supposed to give an input video for which clip will be attained, giving priority to UOoI. For obtaining the UOoI based clip of the whole video we are going through the fore mentioned in the Fig.1. As per the Fig.1. first, we extract the whole video into a number of images named frames. After that, all these frames obtained from the video are sent to YOLOv7 [1] module to detect objects one by one. In a frame, there is a list of detected objects. Now, after detection of objects in each frame, we will compare those object classes with the class of UOoI. If the UOoI class is in a frame, then the frames are considered positive to be in resulted clip; otherwise, excluded as negative to be in resulted video clip. Like this, all other frames are also classified as positive or negative for clip extraction. The matched-positive signal is then passed through a temporal-window voting step (detailed in sub-section 3.6) before the final clip is assembled. Following sub sections is given for a detailed description of each step.

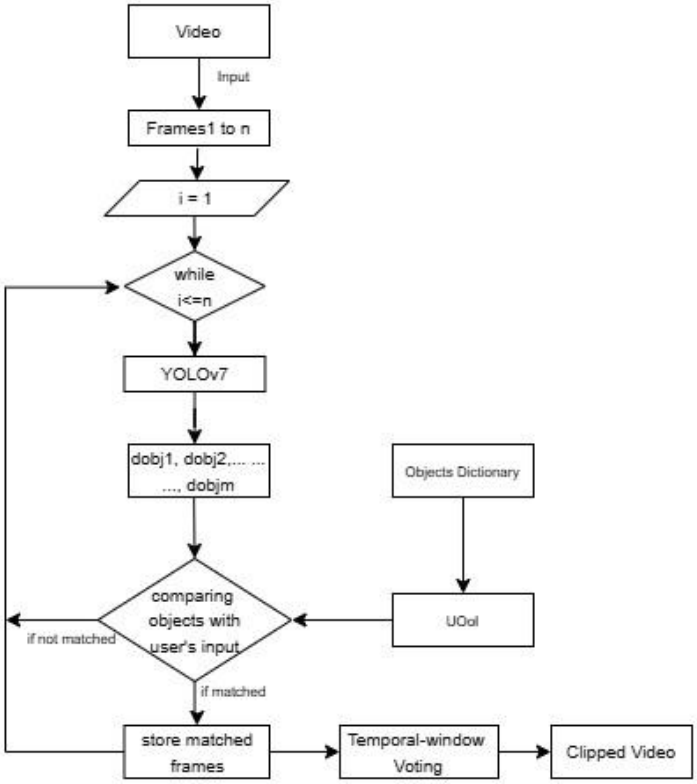


Fig. 1. Overview of Proposed Methodology

3.2. Take Input Video and User Choice from UOoI Dictionary

A user interface is developed to get the input for the whole video clip extraction process. In the user interface, there is a drop-down menu of the object dictionary or UOoI dictionary where the user can select an object. Let there is *w*

number of objects that can be chosen by the user as UOoI. So, the set of all objects is named as $OBG = \{obj_1, obj_2, obj_3, obj_4, ... , obj_{w-1}, obj_w\}$. For each object in the set OBG, there is a label and a number associated with it. So, the set of these objects can be rewritten as:

$$OBG=\{obj_1^{label1},\ obj_2^{label2},\ obj_3^{label4}, \ldots\ldots\ldots, obj_w^{labelw}\}$$

Where obj_i^{labeli}, i in ranges $1 \leq i \leq w$ is the number and $label_i$ is the label of that associated object, respectively. Among all of these objects, users will select one object. That user selected object is called as UOoI (User Object of Interest). obj_{UOoI}^{label} is notation for UOoI with its label. At the same time, the user needs to give a video as input for the clip extraction process. Let the user-selected video be v, and the class label of the user-selected object is obj_{UOoI}^{label}. The video v is passed to the next stage where video to frame conversion occur.

3.3. Video to Frames Conversion

YOLOv7 [1] detect object in an image. To use YOLOv7 [1] in video clip extraction that user selected video v should be converted to a number of frames. That is a very handy process with some built in function. Let user selected video v have m number of frames, so the set of frames for the given video is:

$$F_m=\{f_1,f_2,f_3,f_4, \ldots\ldots\ldots,f_{m-2},f_{m-1},f_m\}$$

Where F_m is set of all frames and f_i is single frame with sequence number i and i belongs to $1 \leq i \leq m$, and m is the total number of frames in video v.

3.4. Object Detection in Each Frames

Now each frame is passed to YOLOv7 [1] module to detect object. YOLOv7 [1] accepts those frames as input and detects objects and also gives the bounding box with classification score one by one frame. The YOLOv7 [1] module also gives the class label of each detected object in each frame. With those processes, all frames has own set of detected objects with class label. For a single frame let there is d number of detected objects. So, the set of detected objects in single frame can be written as:

$$D_i^d=\{dobj_i^1, dobj_i^2, dobj_i^3, \ldots\ldots\ldots, dobj_i^d\}$$

Where D_i^d is set of detected objects in frame i and $dobj_i^j$ represent the j^{th} detected object of the i^{th} frame and $1 \leq i \leq m$ and $1 \leq j \leq d$: m is the total number of frames and d is the number of objects detected in frame i. But each detected object should have its class label. Now, the above set D_i^d can be rewritten with their class label as follows:

$$D_i^d=\{dobj_i^{label_1}, dobj_i^{label_2}, dobj_i^{label_3}, \ldots\ldots\ldots, dobj_i^{label_d}\}$$

Where $dobj_i^{label_j}$, i represents j^{th} detected object of i^{th} frame with its label.

3.5. Determination of Positive and Negative Frames

Now, in this step of the proposed method will select each frame as a positive or negative frame. To select each frame as positive or negative, the detected object sets with class label from the previous step are

$$D_i^d=\{dobj_i^{label_1}, dobj_i^{label_2}, dobj_i^{label_3}, \ldots\ldots\ldots, dobj_i^{label_d}\}$$

Positive of negative frame determination process of each frame are shown in Fig. 2. Now, refer to the sub-section 3.2. where the user-selected object with class label is obj_{UOoI}^{label}. If the user-selected object label matches with any object label from the set D_i^d, then that frame is marked as a positive frame for the clip extraction. Otherwise, those frames will be considered as negative frames for clip generation. Referring to Fig.2. for better understanding. In this way, all the frames are marked as positive or negative frames for clip extraction.

3.6. Temporal-Window Voting for Detection Smoothing

The procedure of sub-section 3.5. produces, for each frame f_i of the input video, a binary indicator $s_i \in \{0,1\}$ where $s_i = 1$ if any detected object label in frame f_i matches the user-selected UOoI, and $s_i = 0$ otherwise. Concatenating frames in the order they appear, keeping only those with $s_i = 1$, would yield a valid extracted clip. In practice, however, the raw per-frame signal exhibits two characteristic noise patterns that hurt both quantitative recall and the perceptual quality of the output clip:

- **Isolated false positives.** A single frame, or a short run of two or three consecutive frames, is marked positive because of momentary motion blur, partial occlusion of an unrelated object, or near-class confusion (for example, a brief misclassification of a *horse* as a *cow* along a class-decision boundary). Including such frames in the output produces short "blips" that are visually jarring.
- **Isolated false negatives.** The detector briefly misses the target object for one or two frames inside a longer stretch in which the object is clearly present, due to a transient pose change, fast camera motion, or temporary occlusion.

These misses punch small holes in otherwise continuous true-positive segments, reducing recall and introducing visible discontinuities.

To suppress both error modes, we apply a one-dimensional majority-vote filter to the binary signal before clip assembly. Given a window size W and a vote threshold K, the filter produces a smoothed signal $\hat{s}$ defined as

$$\widehat{s_i} = \mathbb{1}\left[\sum_{k=-\lfloor \frac{W}{2} \rfloor}^{\lfloor \frac{W}{2} \rfloor} s_{i+k} \geq K\right] \tag{1}$$

where $\mathbb{1}[\cdot]$ is the indicator function and out-of-range indices (i.e. $i + k < 1$ or $i + k > n$, where n is the total frame count) are treated as zero. The parameters satisfy $1 \leq K \leq W$, and we adopt $K = \lceil W/2 \rceil$ in this work, corresponding to a simple majority vote within the window. Frame f_i is retained in the output clip if and only if $\hat{s}_i=1$.

Equation (1) has a direct interpretation. An isolated false positive at frame f_i (i.e. $s_i = 1$ surrounded by zeros) yields a window sum below K and is removed: $\hat{s}_i=0$. Conversely, an isolated false negative inside a true segment (i.e. $s_i = 0$ but most of its neighbours are positive) yields a window sum at or above K and is recovered: $\hat{s}_i=1$. The filter is therefore symmetric: it suppresses both single-frame insertions and single-frame deletions, in proportion to how isolated they are relative to the window length. A naive implementation of (1) requires $O(W)$ work per frame and $O(nW)$ work over the full video. We compute it in $O(n)$ time using a running-sum trick: denoting $C_i = \sum_{j=1}^{i} S_j$ the cumulative sum of the raw signal (with $C_0 = 0$ and out-of-range $s_j = 0$), the window sum at frame i is

$$\sum_{k=-\lfloor \frac{W}{2} \rfloor}^{\lfloor \frac{W}{2} \rfloor} s_{i+k} = C_{i+\lfloor \frac{W}{2} \rfloor} - C_{i-\lfloor \frac{W}{2} \rfloor - 1} \tag{2}$$

The cumulative sum itself is built in a single pass over s. Since the filter operates only on the per-frame binary signal and not on image data, its cost is negligible compared to the YOLOv7 [1] inference: for an n-frame video at 30 fps, the entire smoothing step takes on the order of milliseconds.

The parameter W trades two competing properties. Larger W produces smoother output and suppresses noisier detectors, but also erases legitimate short appearances of the target object whose length, in frames, is less than K.

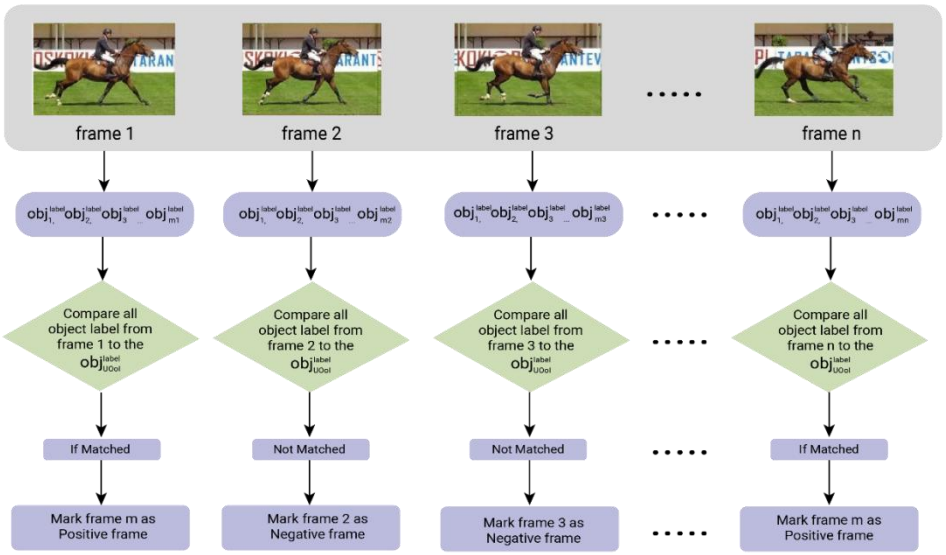


Fig. 2. Determination of positive and negative frames

Smaller W preserves short appearances but lets more single-frame noise through. The appropriate setting therefore depends on the expected length of true positive segments in the source content. We adopt $W = 5, K = 3$ as the default reported throughout Section 4; at 30 fps, this corresponds to a temporal window of approximately 167 ms and a simple-majority vote. The empirical sensitivity of all four evaluation metrics to the choice of (W,K) is reported in sub-section 4.7., which sweeps W from 1 (no voting) to 35 on both datasets.

The end-to-end procedure is summarised in Algorithm 1. below, with the addition of the temporal-voting step shown in lines 8–10.

3.7. Extraction of Video Clip with Positive Shots

For constituting the extracted clip of the user-selected video v with respect to the given obj^{label}_{UOoI}, only the positive frames are considered. All the positive frames are chronologically added one after another which results in another video named as the clipped video of the original video v. The resulting video is the clipped video of the user-selected original video, prioritizing the user-selected object.

3.8. An Algorithm Describing Entire Process

Firstly, video v_i taken with UOoI as input. So video v_i is decomposed to n numbers of frames where n is the total number of frames. Now each frame f_i is passed to the YOLOv7 [1] module for object detection where $1 \leq i \leq n$. From the detected object list object is matched with UOoI for each frames detected object lists. If any matched occurs then marked that frame f_i as positive frame otherwise marked as negative frame if no match is found. After that clipped video is made arranging all the positive frames in chronological order.

The total complexity of the algorithm is $O(n \cdot T_{yolo})$, where n is the total number of frames and T_{yolo} is the per frame inference cost of YOLOv7 at the fixed input resolution of $640 \times 640 \times 3$. The label-matching step contributes $O(|D|)$ per frame, where $|D|$ is the number of detections in that frame (bounded by a small constant in practice). The temporal-voting step adds $O(n)$ via the running-sum implementation of (2), which does not change the asymptotic cost. Frame decomposition and final concatenation are both $O(n)$. The pipeline is therefore dominated by the per-frame detector inference, as expected.

Table 1. YOLOv7-based video clip extraction with temporal-window voting.

Algorithm 1: YOLOv7-based Video Clip Extraction with Voting	
Input: Video V_i, UOoI, window size W, vote threshold K	
Step	Cost
1. Decompose Vi into n frames $F = \{f_1, f_2, \ldots, f_n\}$	$O(n)$
2. **For** each f_i in F:	$O(n)$
3. Detect object set $D = \{d_1, d_2, \ldots, d_j\}$ in f_i via YOLOv7	$O(T_{yolo})$
4. **For** each $d \in D$:	
5. **If** label(d) = UOoI: set $s_i \leftarrow 1$ and break	$O(\|D\|)$
6. **Else**: set $s_i \leftarrow 0$	
7. Build cumulative sum C of s	$O(n)$

Algorithm 1: YOLOv7-based Video Clip Extraction with Voting	
Input: Video V_i, UOoI, window size W, vote threshold K	
Step	Cost
8. **For** each $i \in \{1,\ldots\ldots, n\}$:	$O(n)$
9. Compute $w_i \leftarrow C_{i+\lfloor w/2 \rfloor} - C_{i-\lfloor w/2 \rfloor - 1}$	$O(1)$
10. Set $\hat{s_i} \leftarrow \mathbb{1}[w_i \geq K]$	
11. Concatenate frames $\{f_i : \hat{s}=1\}$ in temporal order	$O(n)$
Output: Extracted video clip V'	

4. Experimental Result

To evaluate our methodology, we set an environment for the experiment. Algorithms are coded in Python 3.12, with Microsoft Windows 11 pro using an Intel(R) Core(TM) i9-13900k CPU of 13th generation processor clocked at 3.00 GHz , 64.00 GB of memory and NVIDIA GeForce RTX 4090 GPU. The libraries used are cv2, Numpy, Torch and CUDA. The IDE (Integrated Development Environment) is used for this experiment is PyCharm Community Version (2024.1.4). The object detection in each frame mentioned (see sub-section 3.4) done by modification of code given with this paper [1]. We do not need to training as we are using a pretrained model on same dataset MS COCO [17]. We have used the default weights of pretrained YOLOv7 [1] in this experiment. These include, fixed size image of 640×640 as a input with three channels (RGB) making 640×640×3, Confidence score threshold as (conf_thres = 0.25) and IoU (Intersection over Union) threshold as (iou_thres = 0.45).

In this methodology, we need two datasets, plus the MS COCO [17] label vocabulary inherited from the pretrained detector, and the other two are for evaluating our procedure performance.

4.1. DoB Objects Lists from MS COCO Datasets

As mentioned before, we used YOLOv7 [1] for object detection. With our procedure the user needs to select the UOoI (User Object of Interest) from a DoB (Dictionary of object). YOLOv7 [1] is trained on MS COCO [17] datasets. MSCOCO[17] dataset has 200,000 labelled images, which includes annotation with 80 different object categories such as people, animals, vehicles, and everyday objects like cups, chairs and laptops. More than 1.5 million object instances are annotated. So, in our methods, DoB offers the same 80 categories of objects to be selected by the user. Because this version of YOLOv7 [1] is trained to detect 80 different categories of objects. So, DoB has same 80 categories that can be selected as UOoI.

4.2. Preparation of SumMe and own custom datasets

Our method needs two inputs: a video to be clipped and a UOoI selected from the dictionary. To assess the framework we use two datasets covering complementary content.

The first is the SumMe [18] benchmark, which consists of 25 videos covering diverse scenarios including travel, sports, and social events, with durations ranging from 1 to 6 minutes. Each video in SumMe [18] is released with ground truth annotations from 15 human summaries; however, those annotations target keyframe selection, highlight detection, and shot-based summarization, and do not include the frame-level object-presence labels required for our evaluation. We therefore re-annotated 22 videos from SumMe [18] at the frame level, marking, for every frame, the presence or absence of each object class that (i) appears in the video and (ii) belongs to the MS COCO [17] category set used by the YOLOv7 [1] detector.

To support evaluation beyond SumMe [18], we additionally introduce a custom dataset of 39 videos collected from public sources, including YouTube, Facebook, and a subset drawn from the TVSum [19] benchmark. Video durations range from 0.5 minutes to 7 minutes, covering a wider range of content than SumMe [18] alone. Each video is annotated at the frame level for the presence of every MS COCO [17] object class that appears in the footage, following the same protocol as the SumMe [18] re-annotation.

4.3. Annotation Protocol and Inter-Annotator Agreement

For both datasets, every video was annotated independently by two annotators using identical guidelines: a frame is labelled positive for a given object class if any visible instance of that class is present in the frame, regardless of bounding box size or partial occlusion. For each video, the annotation output is a plain-text file pairing every object class observed in the video with the frame ranges in which it occurs.

To quantify the reliability of the resulting labels, we compute Cohen's kappa [21] between the two independent annotators. For a binary labelling task, Cohen's kappa is defined as

$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \tag{3}$$

where p_o is the observed agreement between the two annotators (the fraction of frames on which they assign the same

label) and p_e is the agreement expected by chance, computed from the marginal class distributions of each annotator as

$$p_e = \sum_{c \in \{0,1\}} p_1(c)\, p_2(c), \tag{4}$$

with *pk(c)* denoting the proportion of frames assigned label *c* by annotator *k*. Aggregated over all (video, object) pairs, the two independent annotation passes yielded $\kappa = 0.85$ on the SumMe [18] re-annotation and $\kappa = 0.83$ on the custom dataset, both of which fall in the "almost perfect" range of the standard interpretation scale [22].

Disagreements between the two annotators were adjudicated by a third independent annotator who reviewed only the frames in dispute and issued a binding label. The resulting consolidated annotations constitute the final ground truth used in all experiments reported in sub-section 4.5. and sub-section 4.6.

4.4. Evaluation Criteria

To measure the performance of our methods we use precision, recall, f1-score and accuracy to determine the rate of performance. F1-score defines precision and recall in a single value. Before the introducing the formulas for these four-measurement standard we need to define something else that is essential. Recalling the process of assigning positive or negative marks to a frame for a given UOoI (see sub section 3.5). For implementation we take that models prediction positive as 1 and negative mark as 0. Now each video in data set can be passed several times with different UOoI. So, for different UOoI there is different prediction list for clip extraction. Those predicted list is compared with ground truth values with respective UOoI. The following equations provide the mathematical expressions for various evaluation parameters:

$$precision^i_{UOoI} = \frac{TP^i_{UOoI}}{TP^i_{UOoI} + FP^i_{UOoI} + \epsilon} \tag{5}$$

$$recall^i_{UOoI} = \frac{TP^i_{UOoI}}{TP^i_{UOoI} + FN^i_{UOoI} + \epsilon} \tag{6}$$

$$f1\text{-}score^i_{UOoI} = \frac{2 \times precision^i_{UOoI} \times recall^i_{UOoI}}{precision^i_{UOoI} + recall^i_{UOoI} + \epsilon} \tag{7}$$

$$accuracy^i_{UOoI} = \frac{TP^i_{UOoI} + TN^i_{UOoI}}{TP^i_{UOoI} + TN^i_{UOoI} + FP^i_{UOoI} + FN^i_{UOoI} + \epsilon} \tag{8}$$

Where $precision^i_{UOoI}$, $recall^i_{UOoI}$, $f1\text{-}score^i_{UOoI}$, $accuracy^i_{UOoI}$, TP^i_{UOoI}, TN^i_{UOoI}, P^i_{UOoI} and FN^i_{UOoI} means respectively precision, recall, f1-score, accuracy, TP, TN, FP and FN for i^{th} video with UOoI. TP, TN, FP and FN are true positive, true negative, false positive and false negative respectively. Again, ϵ is a very small value which is very close to zero but not zero. Usually, ϵ is *0.000001*, used to avoid the division by zero occurred in any case. Now for a single video *i*, final precision, recall, f1-score and accuracy are calculated by taking the means/average of all UOoI's respective precision, recall, f1-score and accuracy. Let i^{th} video be judged by q number of UOoI's. So now mathematically precision, recall, f1-score and accuracy for a video single video *i* with q number of UOoI's is given below:

$$precision^i = \frac{1}{q} \times \sum_{j=1}^{q} precision^i_{UOoI_j} \tag{9}$$

$$recall^i = \frac{1}{q} \times \sum_{j=1}^{q} recall^i_{UOoI_j} \tag{10}$$

$$f1\text{-}score^i = \frac{1}{q} \times \sum_{j=1}^{q} f1\text{-}score^i_{UOoI_j} \tag{11}$$

$$accuracy^i = \frac{1}{q} \times \sum_{j=1}^{q} accuracy^i_{UOoI_j} \tag{12}$$

Where j belongs in the range of $1 \leq j \leq q$. Now, for the whole dataset's final precision, recall and accuracy measurement, we take the mean of all video's precision, recall and accuracy. However, the F1-score is calculated with the whole dataset's final precision and recall. Mathematically, if in a data set, there is g number of videos, then a dataset's final precision, recall and accuracy are calculated by following formulas:

$$precision^{final} = \frac{1}{g} \times \sum_{i=1}^{g} precision^{i} \tag{13}$$

$$recall^{final} = \frac{1}{g} \times \sum_{i=1}^{g} recall^{i} \tag{14}$$

$$accuracy^{final} = \frac{1}{g} \times \sum_{i=1}^{g} accuracy^{i} \tag{15}$$

$$F1\text{-}score^{final} = \frac{2 \times precision^{final} \times recall^{final}}{precision^{final} + recall^{final} + \in} \tag{16}$$

The reason for not taking the $F1\text{-}score^{final}$ as a mean of all $F1\text{-}score^{i}$ is that the f1-score is to gives one number representation of precision and recall. So, representing $F1\text{-}score^{final}$ as the harmonic mean of $precision^{final}$ and $recall^{final}$ is more sensible than the average of all $F1\text{-}score^{i}$.

4.5. Results of SumMe Dataset

We measure our method performance on SumMe [18] datasets. Performance for each video is shown on Fig.3 which is the best results among all the temporal-window voting with *W=31, K= 16*. In the graph Fig.3. the performance of our proposed method for each video is shown in terms of precision, recall, F1 score and accuracy. As expected of the video clip extraction process, video length is reduced in clipped parts. For each video, the duration of the clipped video with respect to different UOoI. The following Table 2. shows the duration of clipped video for the corresponding UOoI for each video.

Table 2. Reduced duration of each video with respective UOoI

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
1	Air Force One	179.76	airplane	170.88
2	Base jumping	133.4	person	29.16
			car	0
			Clock	0
3	Bearpark climbing	133.64	bear	123.6
			person	9.52
			backpack	2.92
4	Bike Polo	102.13	person	91.43
			bicycle	82.13
			bench	19.03
5	Bus in Rock Tunnel	171.03	bus	54.36
			person	30.33
			car	29.77
6	Car over camera	146.21	car	19.51
			person	0.73
7	Car railcrossing	169.34	train	17.95
			car	151.38
			person	165.60
			bus	55.32
8	Cockpit Landing	301.84	airplane	24.82
9	Cooking	85.73	person	85.33
			bottle	6.8
			wine glass	72.27
			fork	6.8

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
10	Eiffel Tower	198.84	person	59.6
			bus	5.56
			car	11.92
11	Excavators river crossing	388.84	person	272.56
			handbag	34.32
			truck	24.48
			car	2.72
			backpack	28.36
12	Jumps	38	person	29.16
13	Kids playing in leaves	106.34	person	106.34
			car	69.64
			bicycle	15.48
14	Notre Dame	192	person	42.67
			stop sign	4.5417
			bus	1.2917
			bicycle	0
			car	16.375
15	Paintball	254.25	person	36.54
16	Paluma Jump	85.89	person	85.88
17	Playing Ball	104	person	3.5
			bird	55.17
			dog	44.6
			chair	33.63
			bowl	7.94
			potted plant	12.9
18	Playing on Water Slide	102.27	person	92.39
			bicycle	0
			chair	1.40
			car	1.13
19	Saving Dolphins	222.99	person	160.46
20	Scuba	74.03	person	17.5
21	St Maarten Landing	70.04	person	44.92
			airplane	42.24
			backpack	8.36
			car	9.56
22	Statue of Liberty	154.52	person	115.92
			handbag	38.88
			backpack	18.4
			clock	0[1]

There are some confusion matrices for a single video with different UOoI given in Fig.4. Finally, we get a precision of about 75.26% , a recall of about 75.05%, an F1-score of about 70.88%, and an accuracy of about 90.12%.

4.6. Results of Own Custom Dataset

We measure our method performance on our dataset. Performance for each video is shown in Fig.5. and Fig.6. where the performance of our proposed method for each video is shown in terms of precision, recall, F1 score, and accuracy. As expected of the video clip extraction process, the video length is reduced to include clipped parts. For each video, the duration of the clipped video with respect to different UOoI.

Table 3. shows the duration of clipped video for the corresponding UOoI for each video.

Table 3. Reduced duration of each video with respective UOoI

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
1	Bear Attacked	82.84943333	cup	4.771433333
			bottle	6.673333333
			bear	0.633966667
			backpack	1.2012
2	bicycle accident	27.06036667	person	18.18483333

[1] means the object present at ground truth but not detected by YOLOv7

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
			bench	0
			traffic light	27.06036667
			car	27.06036667
			bicycle	1.9019
			motorcycle	13.64696667
			truck	12.4124
3	Bike Ride	22.36666667	cow	0.333333333
			person	1.166666667
			car	0
			dog	1.5
			bicycle	0
4	bird on cctv	11.65	bird	0.3
			car	11.65
5	Bus Motorbike Entry	12.2	bus	9.366666667
			car	9.333333333
			motorcycle	4.033333333
			person	12.2
6	CCTV Footage1	14.32	bench	14.32
			person	14.32
			bicycle	0
			car	0
			dog	2.48
7	change your tire	111.0109	person	76.94353333
			car	9.943266667
8	Cycling Accident	20.02	bicycle	4.537866667
			person	8.942266667
			car	8.641966667
9	dog ear clening	144.1333333	bottle	53.83333333
			person	78.23333333
			dog	63.16666667
10	Dog nail cutting	141.28	person	111
			dog	61.96
			book	5.68
			couch	24.12
			vase	5.24
			scissors	5.92
			chair	5.04
			car	2.68
11	Electric Car News	133.2998333	person	105.2718333
			car	78.01126667
			tv	23.65696667
			handbag	11.6116
			tie	47.68096667
			mouse	6.573233333
			bus	12.6126
			backpack	5.5055
12	Killer bee guy	138.8666667	person	89.23333333
			tv	14.96666667
			truck	0
13	kites	10.73333333	person	7.9
			kite	5.133333333
14	Lock Your Bike Properly	143.6101333	person	102.6358667
			handbag	0.467133333
			bicycle	90.55713333
			car	17.45076667
			truck	8.274933333
			traffic light	1.1011
15	Sheep Attack	42.08	sheep	39
			stop sign	0

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
			person	16.44
16	tiger attack zebra	15.88	zebra	11.72
17	Tom Cruise	7.590916667	person	5.880875
			motorcycle	0
			airplane	1.126125
18	traffic jam	25.88	car	19.72
			person	19.64
19	train and bear	15.23333333	train	5.833333333
			bear	0
			traffic light	3.766666667
20	youtube Su6gnOloV U	38.00463333	person	23.79043333
			backpack	0
			teddy bear	0
			bus	9.075733333
			car	12.54586667
			stop sign	4.237566667
			cell phone	5.5055
21	youtube ZH7mBZeGLMM	40.8408	person	32.76606667
			tv	0
			car	24.89153333
			bus	16.08273333
			fire hydrant	1.568233333
22	VID0526 01	14.86666667	airplane	5.9
			car	9.433333333
			person	10.53333333
			traffic light	2.666666667
23	VID0526 02	14.7	bus	4.333333333
			car	9.7
			person	5.5
			traffic light	4.133333333
			train	0
			truck	6.6
24	VID0526 03	22.4363047	cell phone	6.543922204
			person	17.52836305
25	VID0526 04	23.86666667	banana	6.833333333
26	VID0526 05	15.68	baseball bat	0
			person	7.4
27	VID0526 06	17.03333333	bicycle	12
			person	14.8
28	VID0526 7	17.91666667	bicycle	5.791666667
			person	8.125
29	VID0526 08	20.3	bird	11.03333333
30	VID0526 09	14.13333333	bird	6.066666667
			dog	2.566666667
			person	2.566666667
			cow	0
31	VID0526 10	19.4	bottle	8.1
32	VID0526 11	11.08	person	7
			sports ball	0.48
33	VID0526 12	25.66666667	bus	6.133333333
			car	11.36666667
			motorcycle	4.633333333
			person	6.433333333
			traffic light	14.83333333
34	VID0526 13	57.057	car	24.024
			cell phone	13.30495833
			couch	1.042708333
			cup	1.042708333
			dining table	2.002

Video No.	Video Name	Full Video Duration (sec)	UOoI	Clipped Video Duration (sec)
			person	30.23854167
35	VID0526 14	16.08	car	0
			person	14.36
36	VID0526 15	23.53333333	cell phone	19.86666667
37	VID0526 16	54.2	book	5.92
			cell phone	32.72
			microwave	12.36
			person	23.8
			refrigerator	12.36
38	VID0526 17	18.1	cell phone	3.3
			person	11.33333333
			potted plant	6
39	VID0526 18	28.69533333	car	8.717041667
			laptop	9.175833333
			person	15.390375

There are some confusion matrices for a single video with different UOoI given below in Fig.7. Finally, we get a precision of 71.99%, recall is 75.85%, F1-score is 69.89%, and accuracy is about 85.13% with temporal-window voting parameters at W = 5, K = 3.

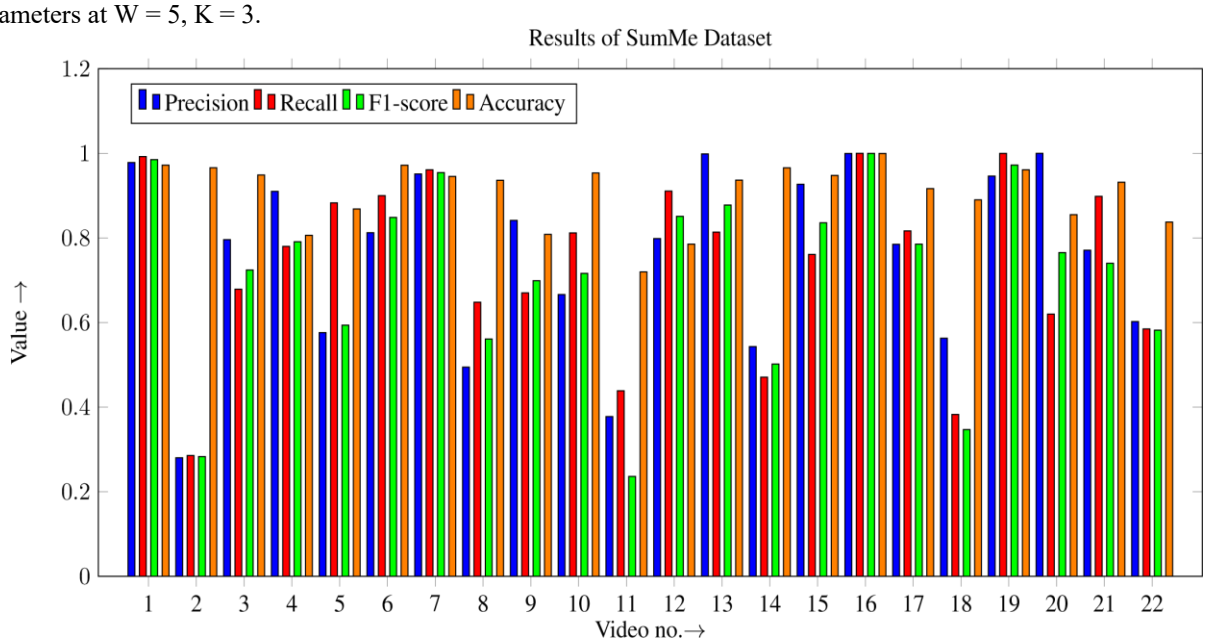


Fig. 3. Results of SumMe Dataset with temporal-window voting (W =5, K =3).

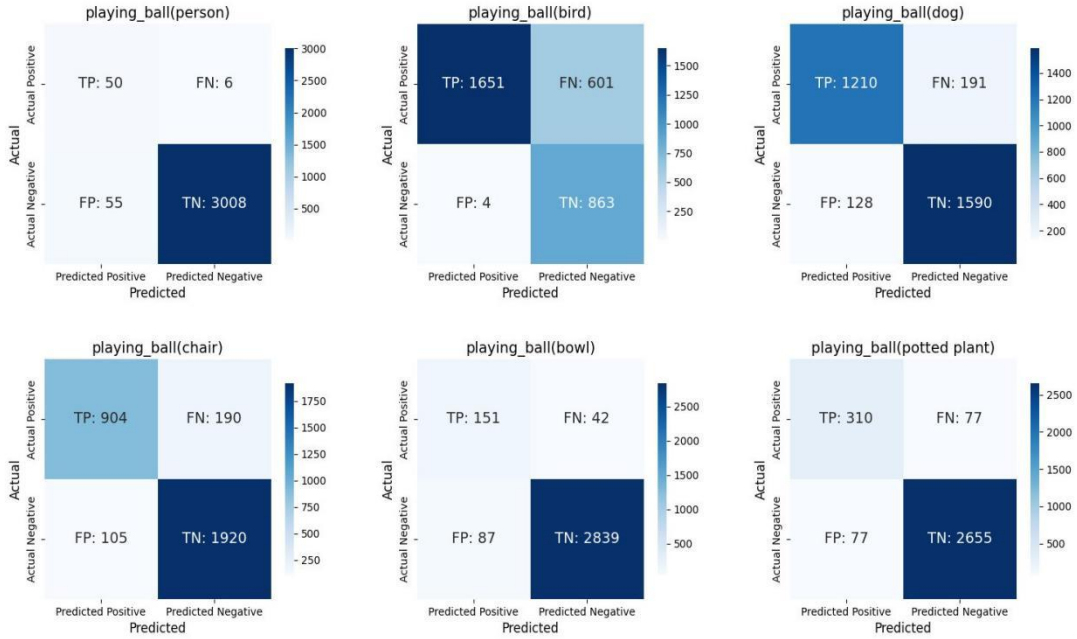


Fig. 4. Confusion Matrix playing ball.mp4 with all occurred UOoI

4.7. Effect of Temporal-Voting Parameters

The temporal-window voting step sub-section 3.6. introduces two hyperparameters: the window size W and the vote threshold K. To characterise their effect we ran the full pipeline at a range of (W,K) settings on both datasets, holding all detector parameters fixed at the values reported in Section 4. We use $K = \lceil W/2 \rceil$ throughout the sweep, corresponding to a simple majority vote within the window. The configuration $W=1,K=1$ recovers the raw per-frame detection signal with no smoothing and serves as the no-voting baseline.

Table 4. reports per-dataset precision, recall, F1, and accuracy across nine voting configurations from W=1 (no voting) to W=35. All values are means across videos using the evaluation protocol of sub-section 4.4.

The two datasets respond differently to temporal smoothing, and we report both behaviours.

On the SumMe dataset[18], temporal voting produces a consistent improvement on all four metrics across the entire sweep. Precision rises monotonically from 0.7325 at $W=1$ to a peak of 0.7659 at $W=29$, accuracy from 0.8976 to a peak of 0.9056 at $W=31$, and F1 from 0.6985 to a peak of 0.7188 at $W=29$. Recall is essentially flat across the sweep. The default configuration $W=5, K=3$ already captures most of the available gain, with +2.0pp precision, +1.0pp F1, and +0.4pp accuracy over the no-voting baseline; the marginal improvement from larger windows is small (an additional +1.0pp F1 going from W=5 to W=29). This consistent gain across metrics on SumMe indicates that the SumMe videos contain enough single-frame-blip detection noise for voting to suppress, and that the typical ground-truth segment is long enough that voting at moderate W does not erase legitimate appearances.

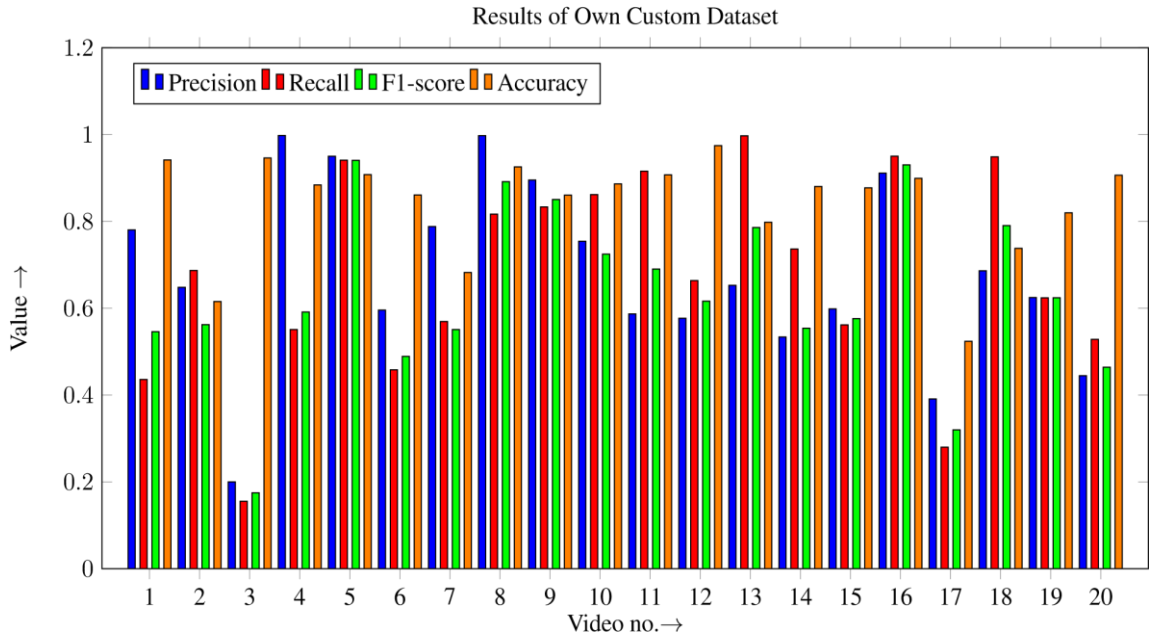


Fig. 5. Graph of performance for each video of our own dataset

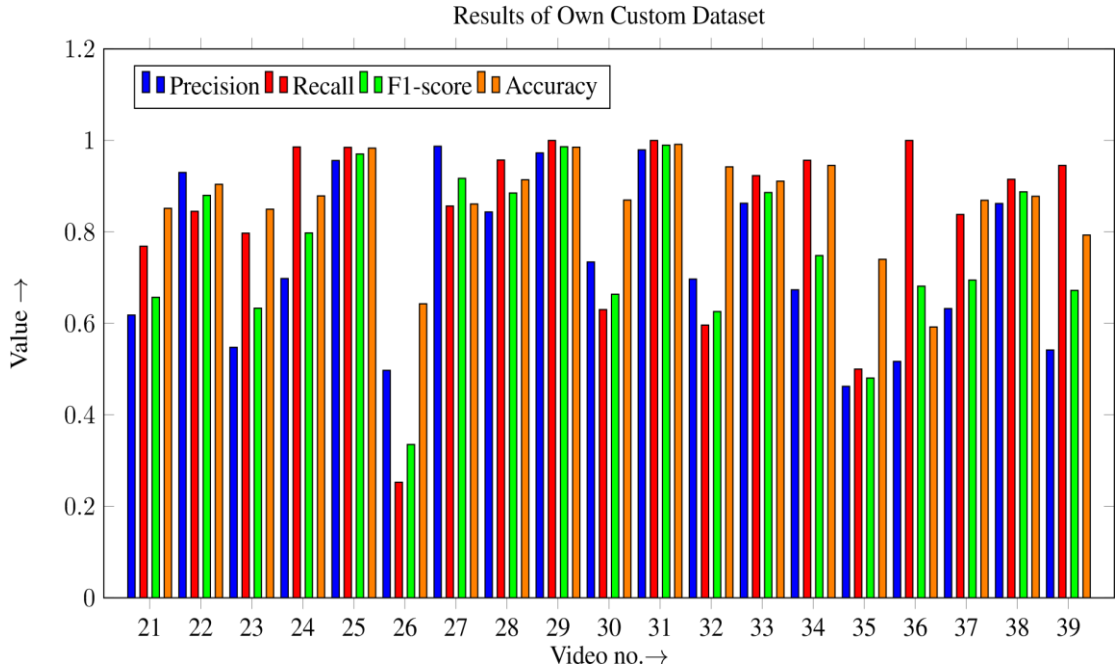


Fig. 6. Graph of performance for each video of our own dataset

On the custom dataset, voting produces negligible movement in F1-score (peak 0.6992 at W=3 vs. baseline 0.6983) and a small accuracy gain that saturates around W=13 (peak 0.8523 vs. baseline 0.8499). Precision and recall both trend downward as W grow, which we attribute to the shorter mean ground-truth segment lengths in this dataset: short legitimate appearances of the target object are erased by larger voting windows whenever the number of positive detections within the window falls below K.

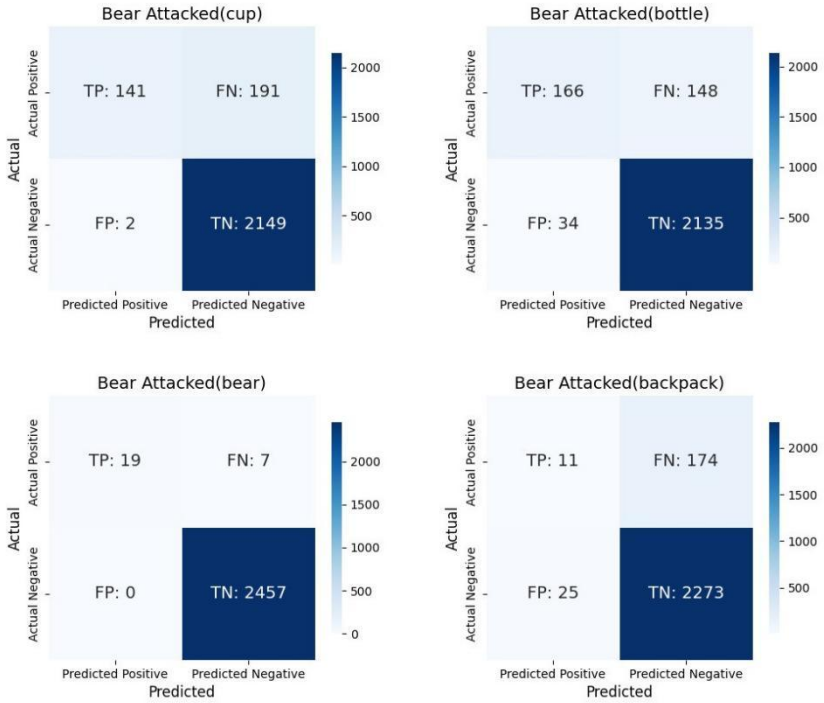


Fig.7. Confusion matrix for Bear Attacked video with all occurred UOoI

The asymmetric behaviour across datasets is informative. Temporal voting helps most when (i) ground-truth segments are long relative to the window size, so no genuine appearance is erased, and (ii) the raw detector produces frequent single- frame false positives, which voting suppresses. The SumMe videos satisfy both conditions strongly:

Table 4. Effect of temporal-window voting on both datasets. *W* denotes the window size and *K* the vote threshold; *W*=1,*K*=1 corresponds to the no-voting baseline. Best value in each column is in **bold**.

	Custom dataset (39 videos)				**SumMe dataset (22 videos)**			
(*W*,*K*)	P	R	F1	Acc	P	R	F1	Acc
(1,1)	**0.7289**	0.7588	0.6983	0.8499	0.7325	0.7467	0.6985	0.8976
(3,2)	0.7268	**0.7592**	**0.6992**	0.8507	0.7433	0.7491	0.7041	0.8999
(5,3)	0.7199	0.7586	0.6989	0.8513	0.7527	0.7505	0.7088	0.9012
(7,4)	0.7140	0.7577	0.6981	0.8520	0.7465	0.7504	0.7098	0.9019
(9,5)	0.7089	0.7561	0.6969	0.8516	0.7492	0.7509	0.7116	0.9025
(13,7)	0.7084	0.7505	0.6939	**0.8523**	0.7491	0.7502	0.7110	0.9032
(19,10)	0.7033	0.7437	0.6876	0.8499	0.7559	**0.7529**	0.7147	0.9046
(25,13)	0.7026	0.7423	0.6863	0.8497	0.7591	0.7518	0.7165	0.9053
(29,15)	0.6889	0.7395	0.6850	0.8501	**0.7659**	0.7477	**0.7188**	0.9054
(31,16)	0.6891	0.7387	0.6845	0.8501	0.7553	0.7412	0.7113	**0.9056**
(35,18)	0.6850	0.7334	0.6799	0.8495	0.7557	0.7386	0.7099	0.9048

object-of-interest segments often span hundreds of frames, and unconstrained outdoor footage produces detection noise from motion blur and partial occlusion. The result is monotonic improvement on all metrics across the entire *W* sweep. The custom dataset includes a higher proportion of brief object appearances (some lasting fewer than 10 frames), for which voting cannot improve precision without first sacrificing recall on the legitimate short segments. The result is much flatter sensitivity to *W*, with the gains saturating quickly and reversing at larger windows. This dataset-dependent trade-off is consistent with the design of the filter: a fixed window cannot simultaneously serve long and short ground-truth segments optimally, motivating adaptive window selection as future work.

5. Graphical User Interface

In the proposed study, a desktop application was developed using Tkinter, a Python-based GUI library, to provide users with an interactive platform for performing video clip extraction (VS) based on object detection. The interface facilitates user engagement by offering an intuitive layout for selecting input videos and objects of interest (UOoI). An image of GUI Fig.8. is given below:

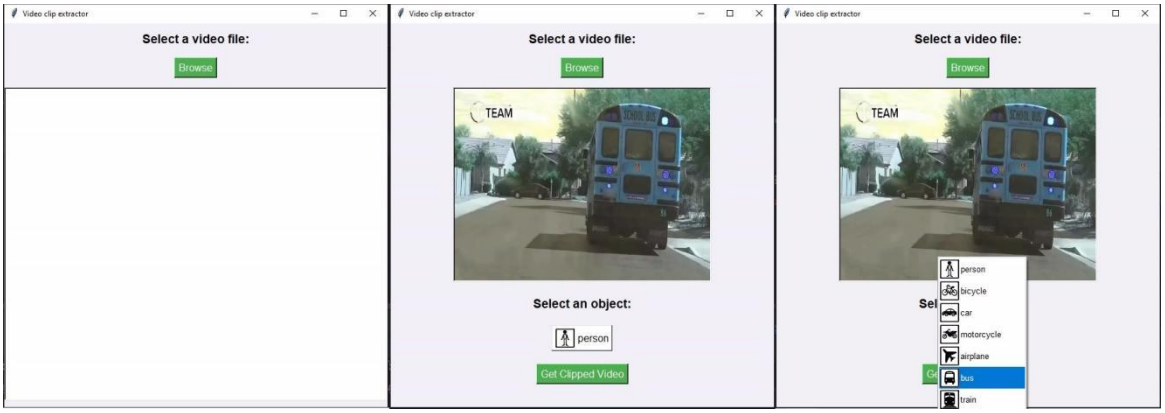


Fig.8. Showing different stage of UI. First appearance (left), After a video selection (middle), Selecting UOoI (right)

5.1. Input Video Selection

The GUI includes a dedicated section for input video selection, where users can browse and select video files in supported formats such as MP4, AVI, MOV, and MKV. The application validates the chosen file to ensure it adheres to the required format. If the input file has fewer than two frames or is not a valid video, the system prompts the user with a warning message. This validation ensures the robustness of the application and prevents errors during processing.

5.2. Object Selection

After selecting the input video, users can choose their desired UOoI from a predefined dictionary of objects. The GUI employs a visually enriched dropdown menu, displaying object names alongside representative icons for easy recognition. Examples include objects like ”person,” ”car,” ”bicycle,” and ”airplane.” This feature enhances usability, allowing users to select UOoI according to their preferences effortlessly.

5.3. Additional Features

Our graphical interface has some additional features too. Those are described below:

1. Preview Section: A preview window is incorporated to display the first frame of the selected video, providing users with a quick visual confirmation of their input.
2. Clip Extraction: A dedicated button, ”Get Clipped Video,” allows users to initiate the clip extraction process. The extracted video is dynamically generated, containing only the segments with the selected object, preserving the original audio.
3. Interactive Feedback: The system provides real-time feedback during the processing phase and displays a completion message upon successful video extraction.

5.4. User-Friendly Design

The GUI employs a modern design, featuring clearly labeled buttons, dropdowns, and labels. Color-coded elements enhance visibility and guide user actions. The graphical interface ensures accessibility and usability, catering to users of varying technical expertise. This GUI implementation highlights the practicality and usability of the proposed method, making it adaptable for diverse applications such as surveillance analysis, sports highlights, and educational content extraction.

6. Concluding Remarks

We have presented a frame-level, object-conditioned video clip extraction framework built around the pretrained YOLOv7 [1] detector. The system takes a video and a user-specific object class as input, identifies frames containing the target object via per-frame detection, applies a temporal-window voting filter to suppress isolated detection errors, and assembles a playable clip with the original audio preserved through ffmpeg-based segment-level extraction.

- A YOLOv7-based video clip extraction pipeline exposing the 80 MS COCO object categories as user-selectable targets, distinguishing the approach from prior YOLOv3-based summarization frameworks [14, 15] by use of a stronger detector backbone and by producing playable dynamic clips rather than static keyframe summaries.
- A temporal-window voting step (window W, vote threshold K) that filters the per-frame binary detection signal before clip assembly, addressing the isolated false positives and single-frame holes that hurt both quantitative recall and perceived continuity in naive frame-by-frame approaches.
- An ffmpeg-based audio preservation step that maps voted-positive frame indices back to source timestamps and extracts the corresponding audio segments, eliminating the audio-loss limitation common to frame-decode/re-encode pipelines
- Frame-level annotations, with respect to MS COCO object categories, of 22 videos from the SumMe [21] benchmark, which was originally released with shot-level summary annotations rather than object-presence labels.
- A new dataset of 39 videos collected from public sources, annotated at the frame level for object presence, intended to support evaluation of object-conditioned extraction methods.
- A desktop graphical interface enabling video selection, target-object choice from the dictionary, input preview, and clip export.

Our experimental results are summarised as follows:

- On the SumMe [18] benchmark, the framework attains an F1-score of 70.88% with 90.12% accuracy.
- On the newly annotated 39-video dataset, the framework attains an F1-score of 69.89% with 85.13% accuracy.
- Temporal-window voting improves F1-Score from 69.83% to 69.89% on the custom dataset when compared with the no-voting baseline *(W=1,K=1)*.
- The pipeline does not require additional training: the pretrained MS COCO [17] weights are used directly, and the only hyperparameters introduced by the framework are the voting window *W* and threshold *K*.

Several limitations are listed below:

- The framework extracts clips conditioned on a single User Object of Interest at a time; multi-object conditioning (e.g. ”frames containing both a bicycle and a person”) would require extending the matching step beyond single label comparison.
- Evaluation reports per-frame precision, recall, and F1-score against frame-level ground truth; we do not measure perceptual clip quality or downstream user-task performance.

Natural extensions include: adaptive selection of *W* and *K* based on detector confidence statistics or estimated motion content; multi-object query formulation (conjunction, disjunction, and ordered queries over the dictionary); replacing the MS COCO [17] vocabulary with an open-vocabulary detector (for example YOLO-World or CLIP-conditioned detectors) to remove the fixed-class constraint; transition smoothing via short cross-fades or detected shot boundaries; and user studies measuring perceived clip quality alongside the frame-level metrics reported here.

In our proposed video clip extraction work, we aim to develop a video clip extraction system which extract clip of a video by detecting object in each frame of the video given that an object label.

All the Declarations and Statements

Author Contributions Statement

M. M. Hasan — Writing (Original Draft): Drafted the initial manuscript, conducted the literature review, Methodology, Software, Validation, Formal analysis, Investigation, and data curation.
M. T. Ahammed — Data Curation and Software Development: Performed data acquisition, dataset preprocessing, Dataset Validation, and Performance Evaluation: validated the results using standard evaluation metrics.
M. F. Sadique — Conceptualization, Methodology, Resources, Supervision, and Formal Analysis: Conceived the research idea, supervised project execution, and dataset curation.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding Declaration
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement
This study utilized a publicly available dataset cited in this paper. The re-annotated version of the public dataset, as well as the custom dataset developed for this study, is currently kept confidential for use in future research. However, the re-annotated datasets may be made available upon reasonable request to the authors.

Ethical Declarations
This study involved the analysis of large-scale, predominantly unlabeled videos obtained from publicly available sources. The research did not involve direct interaction with human participants or animals, and therefore, formal ethical approval and informed consent were not required.

Acknowledgments
We would like to express our sincere gratitude to the reviewers for their constructive feedback and valuable suggestions, which have helped improve the quality of this work and the reliability of its results.

Declaration of Generative AI in Scholarly Writing
During the preparation of this work, the authors used ChatGPT in order to improve readability and grammar. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Abbreviations
The following abbreviations are used in this manuscript:

UOoI - User Object of Interest
MS COCO - Microsoft Common Object in Context
DoB - Dictionary of Objects
YOLO – You Look Only Once

Appendix A\B\C…, with appendix tile
None.

References

[1] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object,” in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, 2023. https://doi.org/10.1109/CVPR52729.2023.00721

[2] D. Potapov, M. Douze, Z. Harchaoui and C. Schmid, “Category-specific Video Summarization,” in ECCV- Euro pean Conference On Computer Vision, Zurich, Switzerland, 2014. https://doi.org/10.1007/978-3-319-10599-4_35

[3] P. Gunawardena, H. Sudarshana, O. Amila, R. Nawarante, D. Alahakoon, A. S.Perera and C. Chitraranjan, “Interest Oriented Video Summarization with Keyframe Extraction,” in 2019 19th International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka, 2019. https://doi.org/10.1109/ICTer48817.2019.9023769

[4] R. J. R, P. Nimmagadda, K. Sudhakar, B. C. J, P. Rajasekar and S. M. A, “Perceptual Video Summarization Using Keyframe Extraction Technique,” in 3rd Internation Conference on Innovative Practices in Technology and Manage ment (ICIPTM), Uttar Pradesh, India, 2023. https://doi.org/10.1109/ICIPTM57143.2023.10118236

[5] M.Shim, T. Kim, J. Kim and D.Wee, “MaskedAutoencoder for Unsupervised Video Summarization,” ArXiv, 2023. https://arXiv:2306.01395v1

[6] B. P. A. Prabhu, T. Sharma, R. Dani and M. S. G. Prasad, “A Novel Approach to Video Summarization Using AI-GPT and Speech Recognition,” in Data Science and Applications, Singapore, 2024. https://doi.org/10.1007/978-981-99-7817-5_16

[7] H. Khan, T. Hussain, S. U. Khan, Z. A. Khan and S. W. Baik, “Deep multi-scale pyramidal features network for supervised video summarization,” Expert System with Applications, vol. 237, 2024. https://doi.org/10.1016/j.eswa.2023.121288

[8] A. Singh and M. Kumar, “Bayesian fuzzy clustering and deep CNN-based automatic video summarization,” Multi media Tools and Applicationss, vol. 83(1), pp. 963-1000, 2024. https://doi.org/10.1007/s11042-023-15431-9

[9] A. SahuandA.S.Chowdhury, “Egocentric video co-summarization using transfer learning and refined random walk on a constrained graph,” in Pattern Recongnition, 2023. https://doi.org/10.1016/j.patcog.2022.109128

[10] J. Qin, H. Yu, W. Liang and D. Ding, “Video Summarization Using Knowledge Distillation-Based Attentive Net work,” in Cognitive Computation, 2024. https://doi.org/10.1007/s12559-023-10243-3

[11] P. Narwal, N. Duhan and K. K. Bhatia, “A novel multi-modal neural network approach for dynamic and generic sports video

summarization," in Engineering Applications of Artificial Intelligence, 2023. https://doi.org/10.1016/j.engappai.2023.106964
[12] G. Wang, X. Wu and J. Yan, "Progressive reinforcement learning for video summarization," Information Sciences, 2024. https://doi.org/10.1016/j.ins.2023.119888
[13] Liu, Ling & Özsu, M. Tamer. (2009). Encyclopedia of Database Systems. https://doi.org/10.1007/978-0-387-39940-9
[14] H. B. U. Haq, W. Suwansantisuk and K. Chammongthai, "An Optimized Deep Learning Method for Video Summa rization Based on the User Object of Interest," International Journal of Advanced Computer Science and Applica tions(IJACSA), vol. 14, no. 10, 2023. https://doi.org/10.14569/IJACSA.2023.0141027
[15] H. B. U. Haq, M. Asif, M. Ahmed, R. Ashraf and T. Mahmood, "An Effective Video Summarization Framework Based on the Object of Interest Using Deep Learning," Mathematical Problems in Engineering, pp. 1-25, 2022. https://doi.org/10.1155/2022/7453744
[16] J. Redmon and AliFarhadi, "YOLOv3: An Incremental Improvement," 2018. https://arXiv:1804.02767v1
[17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Doll´ ar, "Microsoft COCO: Common Objects in Context," 2015. https://arXiv/abs/1405.0312
[18] M. Gygli, H. Grabner, H. Riemenschneider and L. VanGool, "Creating Summaries from User Videos," in Computer Vision–ECCV 2014, 2014. https://doi.org/10.1007/978-3-319-10584-0_33
[19] Y. Song, J. Vallmitjana, A. Stent and A. Jaimes, "TVSum: Summarizing web videos using titles," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015. https://doi.org/10.1109/CVPR.2015.7299154
[20] M. Afham, S. N. Shukla, O. Poursaeed, P. Zhang, A. Shah and S. Lim, "Revisiting Kernel Temporal Segmentation as an Adaptive Tokenizer for Long-form Video Understanding," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023. https://arxiv.org/abs/2309.11569
[21] J. Cohen, "A coefficient of agreement for nominal scales," Educ. Psychol. Meas., vol. 20, no. 1, pp. 37–46, Apr. 1960. https://doi.org/10.1177/001316446002000104
[22] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," Biometrics, vol. 33, no. 1, pp. 159-174, Mar. 1977. https://doi.org/10.2307/2529310

Authors' Profiles



Md. Mahmudul Hasan received his B.Sc. degree in computer science and engineering from Khulna University, Khulna, Bangladesh, in 2025. He is currently pursuing an M.Sc. degree in the same discipline at Khulna University. His major field of study is computer science.
He is interested in research related to video summarization, deep learning, and computer vision. He has worked on projects involving object detection and search engine development. His research interests also include deep learning, video processing, and information-retrieval.



Md. Titas Ahmmed (born August 10, 2000) received the B.Sc. degree in Computer Science and Engineering from Khulna University, Bangladesh. He developed projects including a drone-based surveillance system and the C++ game Traping, and interned at Khulna University ICT Cell, specializing in networking and resource management.
His research focuses on image generation, segmentation, and video analysis. His work bridges practical software development with advanced computer vision solutions.
Mr. Ahmmed's interests span deep learning applications for automated multimedia analysis.



Md. Farhan Sadique is a Lecturer in the Computer Science and Engineering Discipline at Khulna University, Khulna, Bangladesh. He earned both his Bachelor and Master of Science degrees in Computer Science and Engineering from Khulna University.

How to cite this paper

Md. Mahmudul Hasan, Md. Titas Ahmmed, Md. Farhan Sadique, "User Object of Interest Based Video Clip Extraction using Pretrained YOLOv7", International Journal of Engineering and Manufacturing (IJEM), Vol.16, No.3, pp.345-364, 2026. DOI:10.5815/ijem.2026.03.21