Real-time Referred Video Object Segmentation (RVOS)-based Video Summarization using Multimodal Transformers

Shashwat Srivastava (2021EEB1210)   Ranjeet Singh (2021EEB1203)

Department of Electrical Engineering
Indian Institute of Technology Ropar

Under the guidance of Dr. Santosh Kumar Vipparthi

May 13, 2025

Outline

Project at a Glance

- **Goal:** Create short, relevant video summaries.
- **Problem with Old Methods:** Often generic, don't understand user interest.
- **Our Approach:** Use Referring Video Object Segmentation (RVOS).
  - User describes an object in text (e.g., "person playing guitar").
  - System (MTTR model) finds and highlights this object.
- **Summarization:**
  - Use these highlighted segments to create the summary.
  - Two methods: simple uniform sampling, and advanced feature-based analysis.
- **Outcome:** System that generates query-focused video summaries.

The Challenge: Video Overload

- We have too much video content to watch.
- Video summarization creates short versions to save time.
- **Key Question:** What is "important" information in a video?
  - This is subjective and depends on the user's needs.
- Traditional methods often use basic visual cues (motion, scene cuts).
  - They might miss the actual meaning or user-specific interests.

A Smarter Way: Referring Video Object Segmentation (RVOS)

- RVOS lets users guide the summarization:
  - Describe an object/action in text (e.g., "the dog catching the frisbee").
  - The RVOS system finds and highlights (segments) that object.
- This enables summaries directly related to user interest.
- Powerful AI models like MTTR (Multimodal Tracking Transformer) [1] excel at this task.

Our Project's Aim: The "RVOS-Summarizer"

- We built a system using RVOS (with MTTR) for video summarization.
- **How it works:**
  1. **Input:** A long video + text query (e.g., "car turning left").
  2. **RVOS:** MTTR segments the queried object in frames.
  3. **Summarization:** A module processes these segments to create a short summary video.
- This query-based method produces highly relevant summaries.
- We explored two summarization techniques:
  - Uniform sampling (basic).
  - Feature-based analysis (advanced).

Key Steps in Our Project (1/2)

Our project involved several main development stages:

- **MTTR Model Setup:**
  - Prepared the Multimodal Tracking Transformer (MTTR).
  - Installed necessary tools and loaded a pre-trained version.
- **Video Processing Pipeline:**
  - Created an automated system (using Python tools like ‘yt_dlp‘, ‘MoviePy‘).
  - This system can:
    - Download YouTube videos.
    - Trim them to desired lengths.
    - Prepare them for the MTTR model.

Key Steps in Our Project (2/2)

Continuing with our development stages:

- **RVOS Inference:**
  - Used the pre-trained MTTR model.
  - It segments objects in videos based on text queries.
  - This process outputs "masks" that highlight the target objects.
- **Summarization Modules:**
  - Implemented two distinct methods to create summaries:
    - **Uniform Sampler:** A simple approach; picks frames at regular intervals.
    - **Feature-based Analyzer:** More advanced; calculates video features (like motion, scene changes, object size, audio events) and uses these features to create summaries and informative plots.
- **Integration:**
  - Combined all components into an end-to-end system.
  - It takes a video URL and query, then outputs: an annotated video, a summary clip, and analytical plots.

Our System's Two-Stage Process

Our RVOS-Summarizer uses a two-stage approach:

1. **RVOS with MTTR:**
   - The MTTR model first identifies and segments (highlights) the object described in your text query throughout the video.
2. **Video Summarization Module:**
   - The output from MTTR (video frames with the object highlighted) is then used to create a summary.

Stage 1: Finding Objects with MTTR

- We use the Multimodal Tracking Transformer (MTTR) [1].
- MTTR understands both video (visuals) and text (language query).
- **Simplified MTTR Process:**
  - Takes video frames + text query as input.
  - Processes visual and text features separately.
  - Fuses these features using a multimodal Transformer.
  - Outputs segmentation masks showing the queried object over time.
- We use a pre-trained MTTR model (no re-training in this project).

MTTR Architecture Overview

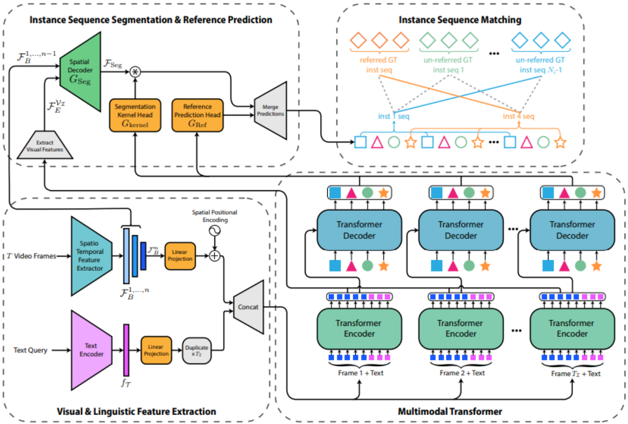


Figure 1: Architecture of MTTR [1]: Uses encoders for video/text, then a Transformer to combine them for segmentation.

Image Source: Botach et al. (2022) [1]

Stage 2: Making Summaries - Two Ways

After MTTR creates an annotated video (frames + object masks), we use one of two summarization methods:

1. **Uniform Sampling Summarizer (Simple Baseline)**
   - Opens the annotated video from MTTR.
   - User specifies summary length (e.g., 50
   - Calculates a sampling interval (N).
   - Picks every Nth frame for the summary.
   - *Result:* A shorter version of the annotated video. Can miss quick events.

Stage 2: Feature-based Analyzer (Advanced)

2. **Feature-based Analyzer**

- Aims for more content-aware summaries.
- Calculates key video features frame-by-frame:
  - Optical Flow (motion)
  - Scene Change Scores
  - Mask Coverage (object size/prominence)
  - Frame Shannon Entropy (visual complexity)
  - Audio Onset Strength (sound events)
- **Uses these features to select keyframes and create a summary video.**
- Summary length is user-controllable (default: 50
- *Result:* Analytical plots of features AND a content-aware summary video.

Feature-based Analyzer: The Process

Algorithm 1 Feature-based Analysis Summarization (Conceptual)

Require: Annotated Video V_{ann} (Frames F_i, Masks M_i), Audio A, Target Summary Percent P_{sum}
Ensure: Feature plots, Summarized Video V_{sum}

Initialize feature lists (motion, scene, mask, entropy, audio)
Process A for $Feat_{audio}$
for each frame F_i in V_{ann} **do**
    Calculate c_i (Mask Coverage); Add to $Feat_{mask}$
    Calculate e_i (Entropy); Add to $Feat_{entropy}$
    if $i > 1$ **then**
        Calculate Optical Flow magnitude m_i; Add to $Feat_{motion}$
        Calculate Scene Change d_i; Add to $Feat_{scene}$
    end if
end for
Generate plots from feature lists.
Analyze features to score/select keyframes to meet P_{sum}.
Assemble V_{sum} from selected keyframes.

This provides rich data and creates a more meaningful summary.

Overall System Flow

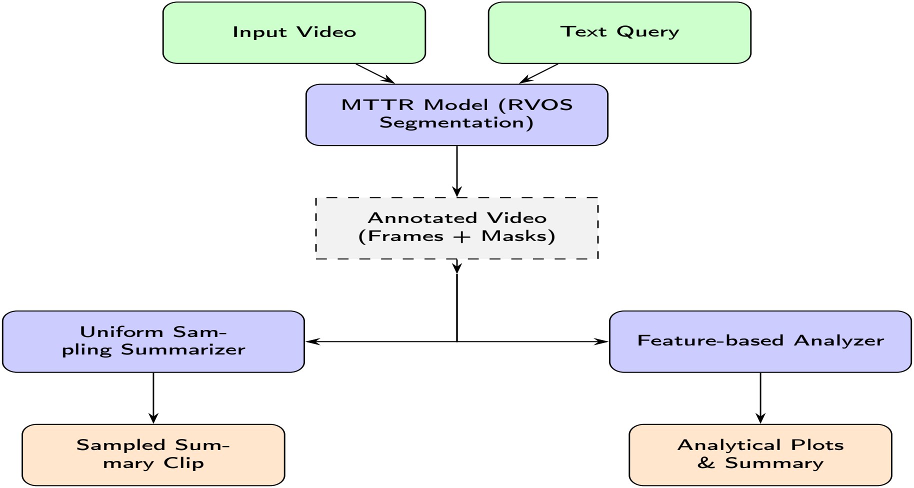


Figure 2: Our RVOS-Summarizer: Processes video and text, MTTR segments objects, then two paths for summarization/analysis.

Key Observations

From our work, we noticed:

- **MTTR Effectiveness:** Very good at finding and highlighting the queried object, even in complex scenes. This is vital.
- **Query Specificity:** Clear, specific text queries give the best results. Vague queries can cause errors.
- **Uniform Sampling Limits:** Simple and fast, but can easily miss short, important events if they fall between samples.
- **Feature Analysis Insights:** The generated plots (motion, object size, etc.) are very informative. They show great potential for guiding smarter summary creation.
- **Computational Cost:** RVOS with MTTR is computationally heavy. Feature analysis (especially optical flow) also adds to processing time.

Limitations of Current Approaches

Traditional video summarization often struggles with:

- **Semantic Understanding:** Difficulty prioritizing content by meaning or specific user interest if relying only on low-level features.
- **Generic Summaries:** Unsupervised methods often create general summaries, not tailored to individual needs.
- **RVOS Complexity:** Some older RVOS methods use complex, multi-stage pipelines, hard to integrate.

Our RVOS-based approach also has limitations:

- **Depends on MTTR:** The quality of the summary is highly dependent on MTTR's segmentation accuracy. Errors propagate.
- **Computational Cost:** As noted, the process is resource-intensive.

Results: MTTR's RVOS Performance

- The MTTR model itself is highly capable.
- It performs well on standard benchmarks like A2D-Sentences.
- The table (from original MTTR paper [1]) shows its strong performance compared to other methods.

Table 1: MTTR performance on A2D-Sentences benchmark [2, 1].

	Precision @ K					**IoU**		
Method	50%	60%	70%	80%	90%	Overall	Mean	mAP
MTTR ($w = 8$)	72.1	68.4	60.7	45.6	16.4	70.2	61.8	44.7
MTTR ($w = 10$)	75.4	71.2	63.8	48.5	16.9	72.0	64.0	46.1

Higher "Precision", "IoU", and "mAP" scores are better.

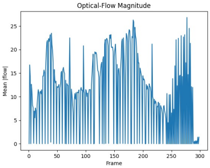
Query: **"a guy riding a bicycle"**

(a) Input video frame

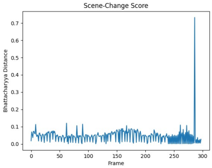
(b) Output frame with RVOS mask

Figure 3: The system successfully segmented the "guy riding a bicycle".

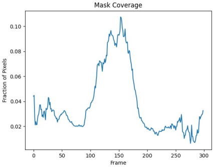
Results: Feature Analysis Plots (Set 1 of 2)



(a) (a) Mean Optical Flow (Motion)



(b) (b) Scene Change Score

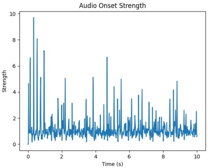


(c) (c) Mask (Object) Coverage

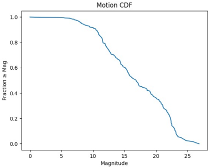
Figure 4: Example plots from Feature Analyzer (1/2): Motion, scene changes, object prominence.

Results: Feature Analysis Plots (Set 2 of 2)

(a) (a) Frame Shannon Entropy



(b) (b) Audio Onset Strength



(c) (c) Motion Distribution Hist.

Figure 5: More analytical plots (2/2): Frame complexity, audio events, motion patterns.

Conclusion

In Summary:

- Successfully built an RVOS-based video summarizer using MTTR.
- It performs query-specific object segmentation, then summarization.
- Two summarization methods explored:
  - Basic uniform sampling.
  - Advanced feature-based analysis (provides insights and summary).
- Demonstrates promise of RVOS for semantic, query-relevant summaries.

Future Work Ideas

- **Smarter Keyframe Selection:** Further develop algorithms using the extracted features for even better summary frame selection.
- **Efficiency Improvements:** Optimize the pipeline for faster processing, aiming towards real-time capabilities (e.g., model distillation).
- **User Studies:** Conduct studies to evaluate the perceptual quality and usefulness of the generated summaries from a user perspective.
- **Explore Different RVOS Models:** Test with newer or alternative RVOS architectures as they emerge.
- **Broader Applications:** Investigate use in other areas like video search or content moderation.

Demo

References I

Botach, A., Zheltonozhskii, E., & Baskin, C. (2022). *End-to-End Referring Video Object Segmentation with Multimodal Transformers*. arXiv preprint arXiv:2111.14821.

Gavrilyuk, K., Ghodrati, A., Li, Z., & Snoek, C. G. M. (2018). *Actor and action video segmentation from a sentence*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8166-8175.

Thank You! Questions?