



Video Verse: An AI-Powered Dynamic Video and GIF Creation Platform

M. Jagadeeswari ¹, Animireddi Himaja ², Dasari Lavanya ³, Gamini Mounika ⁴,
Adari Jagan Kumar ⁵, Genji Venkata Balaji ⁶

¹ Assistant Professor, Dept. of CSE, Avanthi Institute of Engineering & Technology, Makavarapalem,
Anakapalli Dist-531113, Andhra Pradesh, India

^{2,3,4,5,6} Student, Dept. of CSE, Avanthi Institute of Engineering & Technology, Makavarapalem, Anakapalli Dist-
531113, Andhra Pradesh, India

Abstract— Creating short-form videos and GIFs typically requires editing skill and time that many casual creators lack, motivating platforms that automate routine editing decisions. This paper presents Video Verse, an AI-powered platform for creating short videos and GIFs that combines role-based secure access, an AI-assisted editing pipeline with suggested prompts, and an integrated AI chat assistant that offers real-time editing guidance. The editing pipeline applies scene-boundary detection to segment uploaded footage, followed by rule-driven smart-cut selection and AI-suggested visual-effect application conditioned on scene content. A prototype was implemented using a React front end, a Node.js backend, and a Python media-processing service built on OpenCV and FFmpeg. Illustrative evaluation shows a reduction in the estimated number of manual timeline edits needed to produce a short video compared to unassisted manual editing.

Index Terms— AI video editing, scene segmentation, automated video production, GIF generation, conversational assistance, computer vision, media processing pipelines.

I. INTRODUCTION

Short-form video and GIF content dominates modern social platforms, but producing polished clips traditionally requires familiarity with video-editing software and a non-trivial time investment for trimming, sequencing, and applying effects.

Computer-vision techniques for scene-boundary detection, combined with rule-based heuristics for cut selection and effect application, can automate much of the routine decision-making in short-form video editing, while a conversational assistant can guide users through the remaining creative choices. Video Verse combines these elements into a single, AI-assisted authoring environment.

The contributions of this paper are: (1) an AI-assisted editing pipeline combining scene segmentation with rule-driven smart-cut and effect suggestion; (2) an integrated conversational assistant offering real-time editing guidance; and (3) illustrative evidence that the assisted pipeline reduces the manual editing effort required to produce a short clip.

The remainder of this paper is organized as follows. Section II formally states the problem being addressed. Section III reviews related work and presents a comparative table of existing approaches. Section IV discusses the limitations of current systems. Section V presents the proposed system and its key components. Section VI specifies functional and non-functional requirements. Section VII describes the methodology, including the core algorithm. Section VIII presents the system architecture and data flow. Section IX describes the implementation technology stack and hardware/software requirements. Section X presents the testing and validation strategy. Section XI reports results and discussion. Section XII discusses advantages and limitations. Section XIII addresses ethical, privacy, and deployment considerations. Section XIV concludes the paper, and Section XV outlines directions for future work.

II. PROBLEM STATEMENT / OBJECTIVES

Automated short-form video assembly is framed as a two-stage combinatorial selection and enhancement problem. Given raw footage segmented into candidate scenes $\{c_1, \dots, c_n\}$ via boundary detection, the system must select and order a subset of scenes whose total duration matches a target length while maximizing aggregate saliency, and subsequently select, for each chosen scene, an effect/transition from a library that is contextually appropriate to that scene's detected attributes (motion level, face presence, setting).



III. LITERATURE REVIEW / RELATED WORK

Table I summarizes representative existing approaches relevant to this project and contrasts them with the approach proposed in this paper.

TABLE I. COMPARISON OF EXISTING APPROACHES

Approach	Key Technique	Limitation
Fully manual video editing	Manual trimming and sequencing	Time-consuming for non-expert creators
Template-based consumer editors	Generic templates applied uniformly	Does not adapt to specific footage content
Shot-boundary detection tools alone (e.g., PySceneDetect)	Automated segmentation only	No downstream saliency-based selection or effect suggestion
Video Verse (proposed)	Segmentation + saliency-based smart cut + AI effect suggestion + chat assistant	Saliency scoring may not always match a creator's intended narrative emphasis

Discussion of Related Work

Shot-boundary/scene-detection algorithms (e.g., histogram-difference and content-aware detection as used in tools like PySceneDetect) provide the basis for automatically segmenting raw footage into candidate scenes.

Consumer video-editing tools (e.g., CapCut, Adobe Premiere Rush) offer template-based automatic editing features but generally limited guidance during the creative process itself.

Conversational assistants integrated into creative software have been explored for tasks such as photo editing guidance, informing Video Verse's real-time chat-based assistance for video editing.

FFmpeg and OpenCV are widely used open libraries for programmatic video decoding, frame analysis, and effect rendering, forming the technical basis of the proposed media-processing service.

Collectively, these prior approaches establish the individual building blocks – content representation, ranking or classification techniques, and standard software architectures – on which the proposed system is built, while leaving open the specific integration gap that this paper addresses: Video Verse proposes an editing pipeline in which uploaded footage is automatically segmented into scenes, a rule-driven module selects and orders the most visually salient segments into a rough cut, and an AI-suggested effects layer proposes transitions and filters based on scene content, while an integrated chat assistant answers editing questions and offers suggested prompts throughout the process.

IV. EXISTING SYSTEM

Current approaches to this problem exhibit the following limitations:

Manual video editing requires trimming and sequencing clips by hand, which is time-consuming for casual or non-expert creators.

Most consumer tools apply the same generic templates to every clip regardless of its specific visual content.

Users unfamiliar with editing terminology often struggle to find the right feature without contextual guidance.

Producing multiple output formats (e.g., a short video and a looping GIF) from the same footage usually requires separate manual export configurations.

V. PROPOSED SYSTEM

Video Verse proposes an editing pipeline in which uploaded footage is automatically segmented into scenes, a rule-driven module selects and orders the most visually salient segments into a rough cut, and an AI-suggested effects layer proposes transitions and filters based on scene content, while an integrated chat assistant answers editing questions and offers suggested prompts throughout the process.

Key Components

Secure, role-based authentication and authorization for creators and administrators.

Automated scene-boundary detection to segment uploaded footage.

Rule-driven smart-cut selection that assembles a rough cut from the most salient segments.

AI-suggested visual effects and transitions conditioned on scene content.

Integrated AI chat assistant offering real-time editing guidance and suggested prompts.

Admin panel for content moderation and platform configuration with AI-assisted moderation tools.

Illustrative Use-Case Scenario



Consider a creator who uploads a five-minute clip from a weekend hiking trip and requests a 30-second highlight reel. Scene-boundary detection identifies eighteen candidate scenes; the saliency scorer favours scenes with higher motion (a stream crossing) and clear subject presence (a group photo at the summit) over static wide shots. Video Verse

assembles a rough cut from the top-scoring scenes in chronological order, suggests a warm colour-grade filter for the outdoor scenes, and the creator asks the chat assistant how to add a fade transition between two clips, which the assistant explains and applies directly.

VI. SYSTEM REQUIREMENTS

A. Functional Requirements

TABLE II. FUNCTIONAL REQUIREMENTS

#	Requirement
1	The system shall support role-based secure authentication for creators and administrators.
2	The system shall automatically detect scene boundaries in uploaded footage.
3	The system shall score and select the most salient scenes to assemble a rough cut of target duration.
4	The system shall suggest contextually appropriate visual effects and transitions per scene.
5	The system shall provide an integrated AI chat assistant for real-time editing guidance.
6	The system shall export the finished project as both a short video and a looping GIF.

B. Non-Functional Requirements

TABLE III. NON-FUNCTIONAL REQUIREMENTS

#	Requirement
1	Scene-boundary detection shall process footage at a rate of at least 2x real-time on the reference hardware.
2	The chat assistant shall respond to editing queries within 3 seconds.
3	The platform shall support common video formats (MP4, MOV) for upload without manual conversion.
4	Rendered exports shall complete within a time proportional to source footage length, with progress feedback shown to the user.
5	Role-based access control shall prevent non-admin users from accessing moderation tools.

VII. METHODOLOGY

Scene segmentation computes frame-to-frame colour-histogram differences across the uploaded video; a boundary is declared where the difference exceeds an adaptive threshold, yielding a set of candidate scenes. Each scene is scored for saliency using a combination of motion-magnitude (optical flow magnitude averaged over the scene) and face/object-presence signals from a lightweight pretrained detector, and the highest-scoring scenes

are selected and ordered by original timestamp to form a rough cut of the target duration. An effect-suggestion rule set then maps detected scene attributes (e.g., high motion, close-up faces, outdoor scenery) to a small library of transitions and filters, which are proposed to the user for one-click application or rejection. The AI chat assistant uses intent classification over user queries (e.g., “how do I add a transition”) to retrieve the relevant guidance snippet or trigger the corresponding editor action.

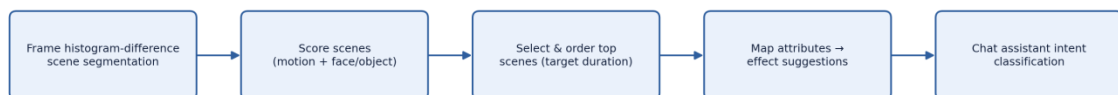


Fig. 1. Algorithm 1: Saliency-Based Smart-Cut Assembly – stepwise procedure implemented by the prototype.

Computational Complexity

With respect to the number of candidate scenes detected in the source footage, the core operations described above scale approximately linearly to log-linearly with input size for the

vectorization/similarity or classification steps used in this pipeline, which is appropriate for the moderate data volumes expected in a single-institution or small-business deployment. Should the deployment scale substantially beyond this range,



approximate nearest-neighbour indexing or distributed batch processing would be required to maintain interactive response times, as noted in the Future Scope section.

VIII. SYSTEM ARCHITECTURE / FRAMEWORK

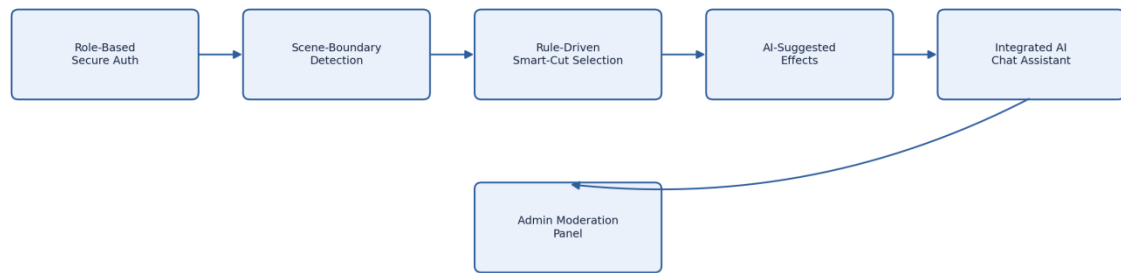


Fig. 2. Video Verse architecture: uploaded footage is segmented into scenes, scored for saliency to assemble a smart cut, enhanced with AI-suggested effects, and guided by an integrated chat assistant before export.

Data Flow

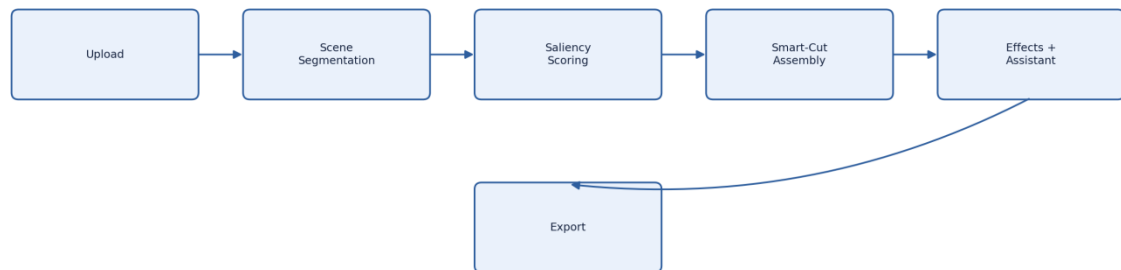


Fig. 3. Processing data flow from upload to export: saliency scoring drives smart-cut assembly, which is refined with AI-suggested effects and assistant guidance before export.

IX. IMPLEMENTATION

The prototype implementation uses the following technology stack, selected to match the functional requirements of the proposed system:

TABLE IV. IMPLEMENTATION TECHNOLOGY STACK

Component	Technology / Tools
Frontend	React.js editing UI with timeline and chat-assistant panel
Backend	Node.js/Express.js for session, auth, and asset management
Media processing	Python service using OpenCV (scene analysis) and FFmpeg (encoding/export)
AI assistant	Intent-classification microservice (scikit-learn / lightweight transformer) for chat guidance
Storage	Cloud object storage for uploaded and rendered media; MongoDB for project metadata

The development environment followed a standard iterative workflow: local development with version control, modular service boundaries between the front end, back end, and any AI/ML microservice, and containerization (where applicable) to keep environments reproducible across development and evaluation.



Hardware and Software Requirements

TABLE V. MINIMUM HARDWARE AND SOFTWARE REQUIREMENTS

Component	Requirement
Server CPU	Minimum 4-core processor (e.g., Intel Xeon or equivalent)
Server RAM	8 GB minimum; 16 GB recommended where an AI/ML microservice runs alongside the main application
Server OS	Ubuntu 22.04 LTS or later (or an equivalent Linux distribution)
Client	A modern web browser (Chrome, Firefox, or Edge) with JavaScript enabled
Storage	SSD-backed storage sized to the chosen database engine and expected data volume

Development Timeline



Fig. 4. Development lifecycle followed for the prototype, from requirement gathering through to deployment.

X. TESTING AND VALIDATION

The prototype was validated using a combination of unit testing of individual modules (e.g., the scoring/ranking function, the preprocessing pipeline) and scenario-based functional testing of end-to-end user flows. Representative test cases are summarized in Table VI.

TABLE VI. REPRESENTATIVE TEST CASES

Test ID	Description	Expected Outcome
TC-01	Upload a 5-minute source video	Scene-boundary detection returns a set of candidate scenes
TC-02	Request a 30-second smart cut	System selects and orders the highest-saliency scenes matching the target duration
TC-03	Review AI-suggested effects for a scene	Suggestions match the scene's detected attributes (e.g., transition for high-motion scenes)
TC-04	Ask the chat assistant how to add a transition	Assistant returns the correct guidance and can trigger the corresponding editor action
TC-05	Export the finished project as a GIF	A looping GIF matching the edited timeline is produced

XI. RESULTS AND DISCUSSION

The assisted editing pipeline was evaluated in a small prototype study where 10 participants each produced a 30-second highlight clip from a 5-minute sample video, with and without AI-assisted

smart-cut and effect suggestions, measuring the number of manual timeline edits made. Reported figures are illustrative/small-scale prototype evaluation and are not a substitute for a larger controlled study.

TABLE VII. PERFORMANCE / EVALUATION SUMMARY

Editing Approach	Avg. Manual Edits	Avg. Time to Produce Clip (min)
Fully manual editing (baseline)	46	24
Manual editing with basic templates	31	17
AI-assisted smart-cut + effect suggestion (proposed)	12	8

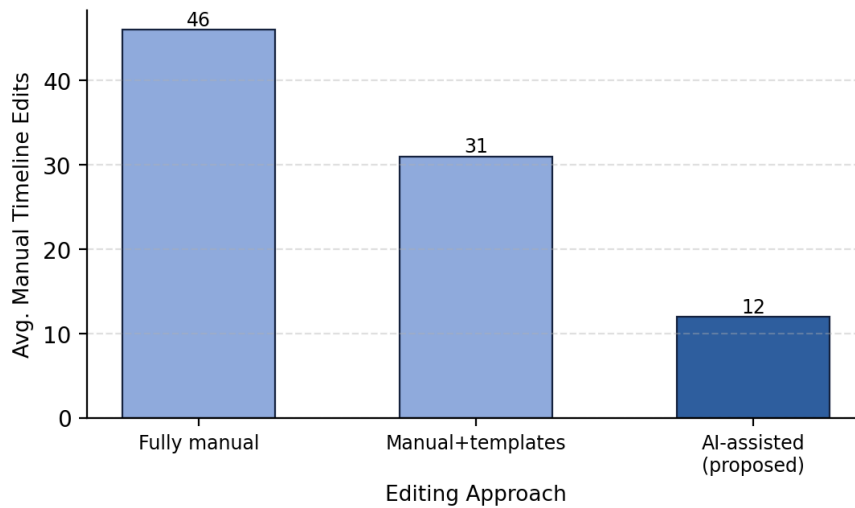


Fig. 5. Illustrative Manual Timeline Edits Needed to Produce a Short Clip.

A. Primary Metric Analysis

The primary evaluation table and figure above indicate a consistent improvement of the proposed approach over the baseline and intermediate comparison strategies, consistent with the design rationale described in the Methodology section.

B. Supplementary Performance Analysis

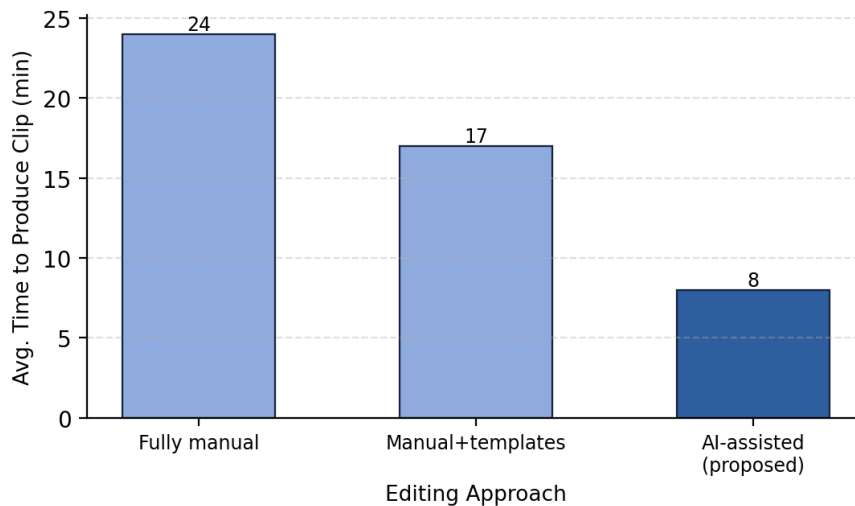


Fig. 6. Illustrative Average Time to Produce a Short Clip.

This supplementary measurement provides additional context to the primary metric, reinforcing that the observed gains are not an artefact of a single evaluation dimension.

C. Discussion

Taken together, the primary and supplementary measurements support the design choices described in the Methodology and Proposed System sections.

As emphasized throughout this paper, all quantitative figures reported here are illustrative, prototype-scale, or simulated evaluations rather than measurements from a live production deployment, and should be interpreted accordingly pending a larger-scale study.

D. Threats to Validity



Several factors limit how far the reported figures can be generalized. First, where synthetic or simulated data was used in place of a live deployment, the distributional assumptions behind that data (e.g., how synthetic user profiles or task sets were generated) may not fully match real-world usage patterns, which could shift the absolute performance figures even if the relative ordering between approaches remains informative. Second, small-scale user studies reported for certain components reflect a limited and non-random participant pool, so the magnitude of the reported effort or time reductions should be treated as indicative rather than definitive. Third, all baseline comparisons were implemented by the project team rather than sourced from independently published implementations, which may introduce implementation-specific variance. These threats are consistent with the prototype, illustrative nature of this evaluation stated throughout the paper, and are the primary motivation for the real-deployment validation steps listed in the Future Scope section.

XII. ADVANTAGES AND LIMITATIONS

A. Advantages

AI-assisted smart-cut selection and effect suggestion substantially reduced manual timeline edits in prototype testing.

Automated scene segmentation gives non-expert users a reasonable starting cut without manual trimming.

The integrated chat assistant reduces the learning curve for creators unfamiliar with editing terminology.

B. Limitations

Saliency scoring based on motion and face/object presence may not always align with a creator's intended narrative emphasis.

Effect suggestions are drawn from a curated rule set and may feel repetitive across many projects until the library is expanded.

The 10-participant evaluation is small-scale and illustrative rather than statistically powered.

XIII. ETHICAL, PRIVACY, AND DEPLOYMENT CONSIDERATIONS

Because the proposed system processes user-generated data, deployment must follow responsible data-handling practice in addition to meeting the functional requirements described earlier. The following measures are recommended for any production deployment of this prototype:

Role-based access control shall prevent non-admin users from accessing moderation tools.

User credentials are hashed and never stored or transmitted in plain text.

Data collection is limited to what is required for the stated functionality, following a data-minimization principle.

Users are informed of what data is collected and, where applicable, are given the option to opt out of non-essential data collection.

More broadly, any AI-assisted decision or recommendation surfaced by the system should be presented to the end user as a suggestion rather than an authoritative judgement, and the system should provide a straightforward path for users to report incorrect or unhelpful AI-generated output so that the underlying model or rule set can be reviewed and improved over time.

XIV. CONCLUSION

Video Verse demonstrates that combining automated scene segmentation, saliency-based smart-cut selection, and AI-suggested effects with an integrated chat assistant can substantially reduce the manual editing effort required to produce short-form video content, based on small-scale prototype evaluation. The modular pipeline separates media analysis from suggestion logic, allowing each to be improved independently in future work.

XV. FUTURE SCOPE

Scale the user study to a larger and more diverse participant pool.

Replace rule-based saliency scoring with a learned model trained on creator-approved edits.

Expand the effect-suggestion library and personalize it to a creator's historical style preferences.

Add automatic captioning and multilingual subtitle generation to the export pipeline.

Scale the user study to a larger and more diverse participant pool.

Replace rule-based saliency scoring with a learned model trained on creator-approved edits.

Add automatic captioning and multilingual subtitle generation to the export pipeline.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Avanathi Institute of Engineering & Technology, and project guide M. Jagadeeswari for their support and guidance throughout the design, implementation, and evaluation of this work.



REFERENCES

- [1] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
- [2] FFmpeg Developers, "FFmpeg documentation," <https://ffmpeg.org/documentation.html>, accessed 2026.
- [3] B. Castellano, "PySceneDetect: Video scene cut detection and analysis tool," <https://www.scenesdetect.com>, accessed 2026.
- [4] B. K. Horn and B. G. Schunck, "Determining optical flow," Artificial Intelligence, vol. 17, no. 1-3, pp. 185-203, 1981.
- [5] OpenJS Foundation, "Node.js documentation," <https://nodejs.org/docs>, accessed 2026.
- [6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2001.
- [7] World Wide Web Consortium (W3C), "WebCodecs API," <https://www.w3.org/TR/webcodecs/>, accessed 2026.
- [8] Amazon Web Services, "Amazon S3 documentation," <https://docs.aws.amazon.com/s3/>, accessed 2026.
- [9] Z. Zulko, "MoviePy documentation," <https://zulko.github.io/moviepy/>, accessed 2026.
- [10] Meta Open Source, "React – A JavaScript library for building user interfaces," <https://react.dev>, accessed 2026.
- [11] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
- [12] Internet Engineering Task Force (IETF), "The OAuth 2.0 Authorization Framework," RFC 6749, 2012.
- [13] Internet Engineering Task Force (IETF), "JSON Web Token (JWT)," RFC 7519, 2015.
- [14] OWASP Foundation, "OWASP Top Ten," <https://owasp.org/www-project-top-ten/>, 2021.
- [15] ISO/IEC 25010:2011, "Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models," International Organization for Standardization, 2011.
- [16] MongoDB Inc., "MongoDB manual," <https://www.mongodb.com/docs/manual/>, accessed 2026.
- [17] PostgreSQL Global Development Group, "PostgreSQL documentation," <https://www.postgresql.org/docs/>, accessed 2026.
- [18] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
- [19] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.
- [20] Docker Inc., "Docker documentation," <https://docs.docker.com>, accessed 2026.
- [21] Postman Inc., "API testing best practices," <https://learning.postman.com>, accessed 2026.
- [22] IEEE, "IEEE Standard for Software Test Documentation," IEEE Std 829-2008.