Journal of Science Research International (JSRI)
ISSN: 2456-6365 (Online)

OPEN ACCESS

ANITRACK: A ROBUST MULTI-STAGE PIPELINE FOR ANIME CHARACTER RECOGNITION AND TRACKING IN VIDEOS

Omkar H. Sherkhane* and Princi Yadav

Department of Data Analytics,

Pillai College of Arts, Commerce & Science (Empowered Autonomous), Navi Mumbai, Maharashtra, India 410206

**Corresponding author E-mail: omkarsherkhane@mes.ac.in*

Received: 20 January 2026	Revised: 20 February 2026	Accepted: 15 March 2026	Published: 31 March 2026

DOI: **https://doi.org/10.5281/zenodo.19571564**

Abstract:

Identifying and tracking characters in anime videos automatically is hard because of the stylized 2D art, changing poses, inconsistent lighting, and frequent shot changes. Current methods mostly focus on classifying static images and don't have the ability to name things in an open set or keep things consistent over time. This paper talks about AniTrack, a Streamlit-based pipeline that does everything from shot detection to YOLOv8 person tracking, face-anchored cropping, anime-specific CCIP embeddings, HDBSCAN clustering with temporal pruning, cross-shot similarity merging, and reference-based open-set naming. For scene segmentation, the system uses PySceneDetect; for detection, it uses ultralytics YOLOv8m; for 768-dimensional identity features, it uses the deepghs/imgutils CCIP model; and for grouping tracklets across shots, it uses hierarchical density-based clustering. Face-anchored crops and temporal overlap constraints make sure that the system works well even when characters only show up for a short time or when there are multiple characters in a scene. AniTrack gets high silhouette scores (>0.65) on sample anime episodes, accurately names characters using cosine similarity thresholding (0.65 default), and makes interactive timelines and character galleries. The pipeline runs completely in the browser and has adjustable thresholds. It is available as open-source for research and fan use.

Keywords: *Anime Character Recognition, Video Tracking, Face Embedding, HDBSCAN Clustering, CCIP, Open-Set Identification, Deep Learning Pipeline.*

1. Introduction

Anime has thousands of frames per episode, so it takes a lot of work to manually label each character. Automated systems need to be able to deal with rendering that isn't photorealistic, over-the-top expressions, and quick cuts. Anime doesn't work with traditional computer vision pipelines made for live-action videos because the domains are different and there aren't any realistic facial landmarks.

© 2026 The Author(s). This is an open access article distributed under the terms of the Creative Commons Attribution (CC BY 4.0) License

AniTrack fills in these gaps by combining the best anime-specific tools with traditional video-processing steps. The system can handle MP4 files that have been uploaded or downloaded from YouTube. It can also create named character timelines, similarity diagnostics, and crops that can be exported. Some of the contributions are:

- Face-anchored cropping with anime-optimized face detection.
- Temporal-aware HDBSCAN clustering is followed by merging shots that don't overlap based on cosine similarity.
- Open-set naming for reference galleries with CCIP verification.
- A Streamlit dashboard that is fully interactive and shows the pipeline in real time, along with advanced configuration sliders.

The pipeline is written in Python 3 and works on regular computers (Google Colab is the best place to run it on a GPU).

2. Related work

Anime character recognition has evolved from static image classification to multimodal and video-based approaches, yet challenges remain in temporal consistency, domain-specific embeddings, and open-set recognition.

Early research focused mainly on image-based methods. Guo *et al.* (2021) evaluated LSTM, CNN, and residual networks for cartoon character recognition, showing that residual networks handle style variations more effectively. Similarly, Li *et al.* (2021) highlighted the importance of style-robust deep features for cartoon face recognition. Qin *et al.* (2019) proposed progressive deep feature learning using unlabeled data to improve generalization for manga characters, though the approach remained limited to single images. Naftali *et al.* (2022) explored transfer learning models such as EfficientNet-B7, achieving 85.08% top-1 accuracy on static anime face datasets.

Recent image-based advancements include Rios *et al.* (2022), who proposed a Vision Transformer with intermediate feature aggregation for improved anime character recognition. Yi *et al.* (2023) introduced a multimodal framework combining image and text information with curriculum learning, though it relies on annotated textual data. Li *et al.* (2022) also introduced the LSASRD benchmark for anime style recognition; even advanced ReID models such as TransReID achieved only 42.24% mAP, highlighting the difficulty of stylized visual domains. Additionally, Jin *et al.* (2025) investigated copyright protection through portrait feature extraction and Seq2Seq-based morphological fitting.

Video-based approaches are relatively limited. Nir *et al.* (2022) proposed CAST, which uses multi-object tracking and self-supervised learning to build character dictionaries for animation series. While effective within a single series, it lacks specialized embeddings, cross-shot clustering, and open-set recognition. Yanagisawa *et al.* (2019) demonstrated the use of DBSCAN clustering for manga character grouping, but without temporal modeling. Khan *et al.* (2025) released a dataset to support video-based re-identification tasks but did not provide a unified pipeline.

To address these limitations, AniTrack introduces an integrated video pipeline combining anime-optimized CCIP embeddings, HDBSCAN-based clustering with temporal pruning and cross-shot merging, and open-set character naming using reference galleries. The system also integrates end-to-end temporal tracking using YOLOv8 with

face-anchored crops and interactive visualization, providing capabilities not available in prior static or video-only approaches.

3. Proposed methodology

The pipeline consists of eight modular stages, each corresponding to a visualized progress step in the UI.

A. Shot detection (Stage 1)

PySceneDetect with ContentDetector (threshold tunable, default 27.0) segments the video into coherent shots. Each shot is stored with start/end frames and seconds.

B. Person detection and tracklet formation (Stage 2)

YOLOv8m (classes=[0], conf=0.45) detects persons per frame (skipping every 3 frames by default). IOU-based tracking (threshold 0.30) builds tracklets; brief tracks (<3 frames) are filtered after gap-15 merging.

C. Face-anchored crops (Stage 3)

For each tracklet, up to 20 frames are sampled. imgutils.detect.detect_faces locates the face; the crop is expanded upward (UCR=0.45) and padded (PAD=25). Fallback to upper-body crop if no face detected. Crops are saved per tracklet.

D. CCIP embedding (Stage 4)

imgutils.metrics.ccip_batch_extract_features produces 768-dim normalized embeddings. Batch size 8, mean-pooled per tracklet. Diagnostic plots show similarity distribution and PCA projection.

E. HDBSCAN Clustering + Temporal prune (Stage 5)

Embeddings are clustered with HDBSCAN (min_cluster_size=3, min_samples=2). Tracklet temporal intervals are recorded. A graph connects same-cluster tracklets only if they do **not** overlap in time, eliminating intra-shot false merges. Connected components yield initial cluster IDs.

F. Cross-shot similarity merge (Stage 6)

Cluster centroids are compared via cosine similarity (merge threshold 0.92). Union-Find merges non-overlapping clusters above threshold, preventing the "9625-merge" bug observed in earlier versions.

G. Reference-gallery open-set naming (Stage 7)

A user-provided reference folder (each subfolder = one character) is embedded with CCIP. Each cluster centroid is matched (threshold 0.65). ccip_same verifies the best crop against the reference. Named clusters receive "Character|sim|tracklets"; unknowns report the closest reference.

Reference dataset construction

The author made the reference gallery for open-set naming by hand in Google Drive. A root folder called character_dataset was made, and inside it was a separate subfolder for each character (for example, Dazai/, Loid/, etc.). There are only 6–7 high-quality reference images in each character subfolder. These pictures were chosen carefully to show a range of angles (front, profile, three-quarter), expressions, and lighting conditions, all while staying true to the anime's official art style. Using ccip_extract_feature, the pipeline (Stage 7) automatically processes every image in each subfolder, calculates the mean embedding, and normalizes it to create one strong 768-dimensional reference vector for each character. This simple folder-based structure lets you add new characters or series right away without having to change any code. It also makes sure that the matching is always high quality when the Name Threshold is set to 0.65.

H. Timeline and metrics (Stage 8)

Matplotlib's bar-chart timeline shows the frame spans for each character. We use valid clusters to calculate the silhouette (cosine), Davies-Bouldin, and Calinski-Harabasz scores.

4. Implementation details

A custom Streamlit app with dark anime-themed CSS (Cinzel Decorative fonts and radial gradients) wraps around the whole system. The sidebar shows real-time indicators and sliders for all thresholds in the pipeline stage. Video preview, shot keyframes, detection samples, crop gallery, embedding diagnostics, cluster previews, named character cards, and timeline are all shown one after the other. Temporary directories take care of crops, reps, and metrics. You can upload MP4 files or download YouTube videos (up to 480p) with the app.

All of the dependencies (ultralytics, scenedetect, hdbscan, networkx, imgutils) are imported when the program runs, and if any are missing, the program will fail gracefully.

V. Experimental Results and Evaluation

The pipeline was tested on multiple anime episodes (e.g., 22-minute clips, 2-minute clips). Typical outputs:

- Shots detected: 80–120
- Tracklets kept: 150–300
- Final clusters: 8–15 (after merge)
- Named characters: 5–10 with similarity >0.65

Quantitative metrics (average over 5 episodes):

- Silhouette score: 0.68 ± 0.04
- Davies-Bouldin: 1.12 ± 0.15
- Calinski-Harabasz: 2850 ± 420

Qualitative results include correctly merged cross-shot appearances of the same character despite pose/lighting changes, accurate suppression of background clusters (<3 tracklets), and precise naming when a reference gallery is supplied. Embedding PCA plots confirm clear separation; similarity histograms peak well above the merge threshold. The interactive dashboard allows instant reconfiguration (e.g., lowering MIN_TRACK_LEN to 3 rescues cameo characters) without re-running the entire pipeline.

Conclusion

AniTrack shows that using general detection and tracking tools along with anime-specific embeddings (CCIP) and temporal-aware clustering can create a useful, high-accuracy character recognition system for animated videos. In the future, there will be more features like splitting characters, adding voice, and exporting to subtitle formats (like ASS with character tags). The open-source Streamlit app makes it easier for researchers, archivists, and fans to look at large collections of anime.

Acknowledgement

The author would like to thank the Department of M.Sc. Data Analytics, Mumbai University, for providing the academic support and environment necessary to complete this research work. The author also acknowledges the developers of open-source tools and libraries used in this project, including YOLOv8 (Ultralytics), PySceneDetect, HDBSCAN, Streamlit, and the imgutils CCIP model by DeepGHS. Their contributions made the development of the AniTrack pipeline possible.

References

1. Yi, F., *et al.* (2023). Anime character identification and tag prediction by multimodality modeling: Dataset and model. *Proceedings of the IEEE International Conference*. https://ieeexplore.ieee.org
2. Rios, E. A., *et al.* (2022). Anime character recognition using intermediate features aggregation with vision transformer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. https://ieeexplore.ieee.org
3. Li, H., *et al.* (2022). A challenging benchmark of anime style recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 4721–4730. https://arxiv.org/abs/2204.14034
4. Nir, O., Rapoport, G., & Shamir, A. (2022). CAST: Character labeling in animation using self-supervision by tracking. *Computer Graphics Forum, 41*(2), 135–145. https://arxiv.org/abs/2201.07619
5. Guo, Z., *et al.* (2021). Cartoon figure recognition with the deep residual network. *Proceedings of the IEEE Conference*.
6. Qin, X., *et al.* (2019). Progressive deep feature learning for manga character recognition. *Proceedings of the ACM International Conference on Multimedia (MM)*, 332–340. https://dl.acm.org
7. Naftali, M. G., *et al.* (2022). *AniWho: A quick and accurate way to classify anime character faces in images*. arXiv. https://arxiv.org/abs/2208.11012
8. Khan, G. I., *et al.* (2025). *Attack on Titan (AoT): Anime image dataset for character, scene, emotion recognition and beyond*. arXiv. https://arxiv.org
9. Jocher, G., *et al.* (2023). *YOLOv8 by Ultralytics*. https://github.com/ultralytics/ultralytics
10. DeepGHS. (2024). *imgutils: Anime-style image data processing library and CCIP embeddings*. https://github.com/deepghs/imgutils