Scientific Visualization, 2025, volume 17, number 5, pages 29 - 37, DOI: 10.26583/sv.17.5.04

Automated video stream search for research and content analysis of video fragments

N.A. Bondareva[1], A.E. Bondarev[2], S.V. Andreev[3], I.G. Ryzhova[4]

Keldysh Institute of Applied Mathematics RAS

[1] ORCID: 0000-0002-7586-903X, nicibond9991@gmail.com
[2] ORCID: 0000-0003-3681-5212, bond@keldysh.ru
[3] ORCID: 0000-0001-8029-1124, esa@keldysh.ru
[4] ORCID: 0000-0003-1613-3038, ryzhova@gin.keldysh.ru

Abstract

This paper examines the problem of targeted search for specific objects in a video stream on request and recording the timestamps of their appearance. Since the solutions currently available on the market were inadequate for the task, it was decided to implement such a tool independently as part of an ongoing research project conducted at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences

Keywords: video stream, video content, search query, timestamps.

1. Introduction

The growing role of video content in all fields of research, from social science to science, is increasingly driving the need to turn to quantitative methods for video content analysis. Researchers in various fields, such as media sociology and digital humanities, are faced with the need to analyze vast volumes of video material to identify patterns of representation of a given phenomenon, study character screen time, analyze gender balance, and other aspects of media content [1, 2].

However, traditional methods of searching for and analyzing the appearance of specific objects in video footage remain extremely labor-intensive. Researchers are forced to review hours of video footage, manually recording the time intervals of the appearances of the characters of interest. This approach not only requires significant time resources but is also prone to subjective errors, especially when working with long-form video or large media collections [3].

Existing commercial solutions for video analysis are either focused on professional video production (Adobe Premiere Pro [4], Final Cut Pro [5]) and are not aimed at video analysis tasks, or require significant financial investments and technical expertise (cloud services AWS Rekognition [5], Google Video Intelligence API [6]). Currently, there is a very limited range of tools that can enable researchers to analyze significant amounts of material according to their needs. Often, research tasks are so highly specialized that widely available software packages do not functionally meet the requirements of the tasks. This poses an additional challenge for researchers immediately prior to conducting scientific research: first selecting a suitable search or analysis tool, and if one does not exist, creating one themselves.

This paper examines the problem of targeted search for specific objects in a video stream on request and recording the timestamps of their appearance. Since the solutions currently available on the market were inadequate for the task, it was decided to implement such a tool independently as part of an ongoing research project conducted at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences.

The aim of this study is to develop and describe a practical tool for automated object search in video content, accessible for use by researchers without special technical skills.

The tasks set include:

1. Analysis of existing solutions for character search in videos and identification of their limitations for research applications;
2. Description of the architecture of the developed tool;
3. Development of software implementation
4. Identification of development prospects and possibilities for integrating the tool into various areas of research.

2. Overview of existing solutions

The modern software market offers a number of solutions for working with video content, but their applicability for research tasks is limited by a number of nuances that have a narrowly targeted application.

Professional video editors such as Adobe Premiere Pro and Final Cut Pro versions include automatic facial recognition features, but they are primarily aimed at video production. The main limitations of these solutions for research applications include: high licensing costs, difficulty learning for users without a technical background, and the lack of the ability to export structured data for subsequent statistical analysis [8].

Cloud services (AWS Rekognition Video, Google Video Intelligence API, Microsoft Azure Video Indexer) provide powerful capabilities for video content analysis, including facial recognition and tracking. However, their use in academic research can be challenging for a number of reasons, ranging from the high cost of processing large volumes of video materials to the complex political situation, which creates a number of restrictions for Russian scientists. It should also be taken into account that cloud technologies are not suitable for all scientific tasks, as some data requires exclusively local processing due to its specific nature and confidentiality requirements when uploading research materials to external servers [9, 10].

In the academic community, there are a number of specialized tools for video content analysis. For example, ELAN (EUDICO Linguistic Annotator is widely used by linguists for video annotation; however, it requires manual tagging and does not include automatic recognition of faces or objects [11]. OpenCV-based solutions provide a powerful technical foundation for creating video analysis systems; however, their use requires significant programming skills. Most ready-made implementations are aimed at technical specialists and do not provide a simple interface for humanities researchers [12].

VideoANT Project (Video Annotation Tool) from the University of Minnesota offers a web interface for video annotation, but also requires manual work and does not include automatic recognition features [13]. In the field of digital humanities, separate tools for media content analysis have been developed. The Cinemetrics project focuses on the analysis of film editing characteristics, but does not include features for finding objects or specific characters [14]. CLARIAH Media Suite provides researchers with access to large collections of media content with search and analysis capabilities, but its automatic object recognition features are also limited [15].

Among the existing software solutions presented in open Internet sources, several key gaps can be identified, in particular:

1. Lack of simple solutions for solving the problem of searching for specific objects/characters in a video stream;
2. Lack of technical means to launch a search based on a sample of specific video material provided by the user;
3. Unavailability of tools for researchers without technical expertise background - most solutions require programming skills or significant financial resources;
4. Limited ability to export structured data for subsequent statistical analysis and visualization;

These aspects confirm the need and relevance of developing a simple, accessible tool for automated search of specific objects in video content, aimed at the needs of researchers in various fields from experimental physics to media studies and digital humanities.

3. Technical implementation

The developed system is a tool for automated character search in video content, consisting of five main stages: pre-processing of the reference image, decomposition of the video material, detection of the object (in our case, the face) in the frames, comparison with the reference and generation of output data in the following form:

- directory with cut video fragments
- a file with timestamps of the appearance of the sought object in the frame.

To operate the program, an organized directory structure is created where the necessary materials are grouped.

After launching the executable file, the program automatically downloads the reference faces added by the user, then searches the user-uploaded video files, analyzing the frames for the target object and comparing the detected faces with the reference sample. Finally, it creates a directory with the results. This directory contains:

- video files with found fragments containing specified faces;
- A JSON file with scene timecodes, describing the start and end times of each fragment;
- log file with information about the processing process;
- settings file (for possible adjustment of parameters).

The user simply prepares the data (photos and videos) and launches the program. After that, the program automatically searches for faces and creates the results in the specified folder.

3.1. System architecture

The system is a modular application designed to automatically extract video fragments containing specific faces. The basic principle is to compare faces detected in the video stream with reference images submitted by the user as photographs.

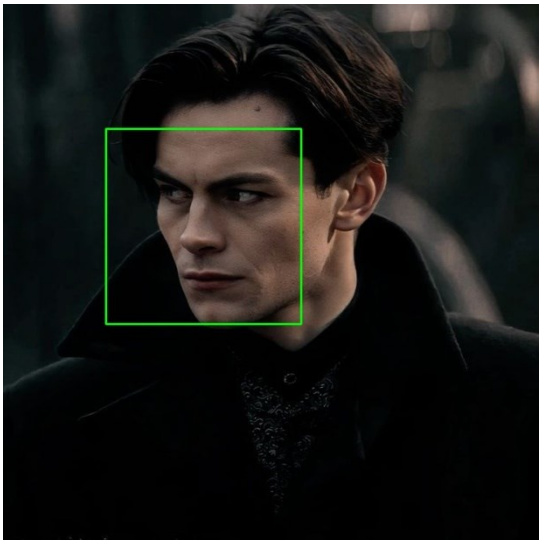

Figure 1. Example of a recognized face

The program's operating algorithm is presented in the diagram (Fig. 2):

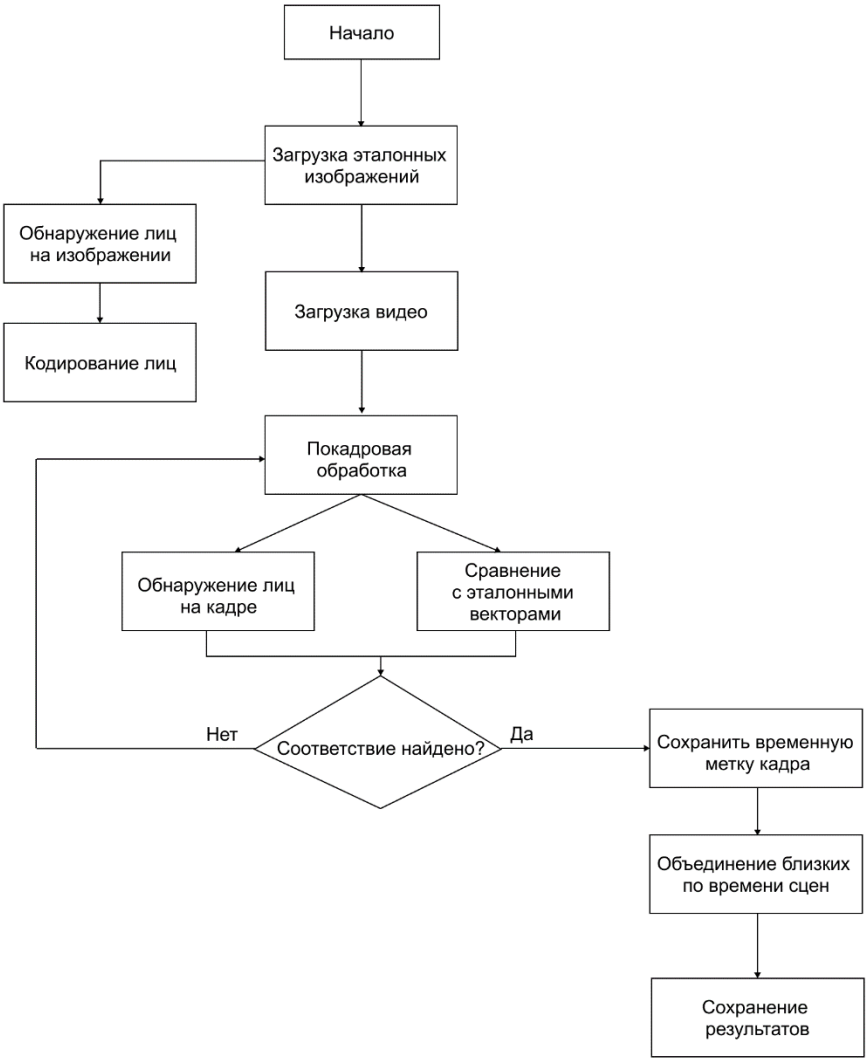

Figure 2. Algorithm diagram

The core component of the system is the FaceExtractor class, which encapsulates all video processing and face recognition logic. The key methods of the FaceExtractor class are:

- load_reference_faces (): Loads reference face images from the specified directory and encodes them into feature vectors.
- process_video (): Processes a video file frame by frame, detects faces in each frame, encodes them and compares them with reference vectors.
- _ merge_scenes (): Merge scenes with detected faces that are close in time into single fragments.
- _ save_results (): Saves video clips with detected faces and metadata to the specified directory.

3.2. Technical parameters

The system is developed in Python 3.9. The face_recognition library [16], a Python wrapper over the dlib library [17], is used for face recognition and matching. The algorithm is based on the FaceNet approach [18], which creates 128-dimensional feature vectors (embeddings) for each detected face using a deep neural network based on the ResNet architecture.

The following libraries were used to implement the main functions of the program:

- opencv-python (cv2) (version 4.8.1.78) - for working with video and images, used for loading, decoding and frame-by-frame processing of video.
- moviepy (version 1.0.3): - a video editing program used to combine scenes and save video fragments.

- numpy (version 1.24.3) - for working with arrays, used to store and process face feature vectors.

The choice of these software tools was due to their ease of use, wide availability of libraries for machine learning and video processing, and good performance.

3.3. Data preprocessing

To ensure high accuracy and speed of face recognition, a series of preprocessing steps are applied to the input data.

Reference face images must be submitted in JPG, JPEG, or PNG format with a resolution sufficient to clearly display the face. The process of loading reference images involves reading the files, detecting faces in the images, and encoding the faces into 128-dimensional feature vectors using the face_recognition library.

Figure 3. Reference facial samples

The system supports MP4, AVI, and MKV video files. Video processing involves loading the video file, decoding frames, and processing them frame by frame. To reduce the processing load, the frame size is adjusted to max_image_size pixels (this parameter is configured in the configuration file). To speed up processing, the frame_interval parameter is used, which determines how often frames are processed (for example, frame_interval = 5 means every fifth frame will be processed).

3.4. Output data format

The results of the system's work are presented in the form of video fragments containing scenes with detected faces, and metadata describing these fragments.

Video fragments are saved in MP4 format using the H.264 codec. Metadata is saved in JSON format and contains the following information: video_name (the name of the original video file); fragments (a list of fragments with detected faces and timestamps); settings (processing parameters), etc. Log files are also recorded (Figure 4).

```
2025-09-03 00:45:03,882 - INFO - Начало загрузки референсных изображений
2025-09-03 00:45:04,488 - INFO - Успешно загружено изображение: im1.jpg
2025-09-03 00:45:05,124 - INFO - Успешно загружено изображение: im2.jpg
2025-09-03 00:45:05,606 - INFO - Успешно загружено изображение: im3.jpg
2025-09-03 00:45:05,608 - INFO - Начало обработки видео: c:\Users\NiciBond\face_detection_project\input_videos\sara.mp4
2025-09-03 00:45:23,477 - INFO - Начало новой сцены: 3.30 сек
2025-09-03 00:45:34,524 - INFO - Конец сцены: 5.30 сек
2025-09-03 00:45:40,445 - INFO - Начало новой сцены: 6.30 сек
2025-09-03 00:45:45,913 - INFO - Конец сцены: 7.30 сек
```

Figure 4 – Example of a program report

The system was tested on the following equipment:

- CPU: Intel Core i7-8700K (6 cores, 12 threads).
- RAM: 16 GB DDR4.
- GPU: NVIDIA GeForce GTX 1070 (8 GB)
- Drive type: SSD.
- Software:
- Operating system: Windows 10 (64-bit).
- Python : 3.9.

3.5. Description of tests

The following tests were performed to evaluate the system performance:

1. Image Size Test: The purpose of this test is to determine the impact of image size on processing speed. During the test, the image size was varied (from 500 to 2000 pixels) and the processing time for one frame was measured.
2. Parallel Processing Test: The purpose of this test is to determine the optimal packet size for parallel processing. During the test, the packet size was varied (from 1 to 16) and the processing time for one frame was measured.
3. Video Processing Test: The purpose of this test is to determine the impact of frame interval on processing speed. During the test, the frame interval was varied (from 1 to 30) and the video processing speed (frames per second) was measured.

The following metrics were used to evaluate the test results:

- Processing time (in seconds).
- Processing speed (frames per second).
- Memory usage (in MB).
- CPU load (in percent).

The test results showed that the system successfully detects specified characters in a video stream, generates video fragments with their participation, and saves metadata describing these fragments.

In particular, the tests confirmed:

- Efficiency of using the face_recognition library for face recognition.
- Positive impact of data preprocessing (image resizing, frame skipping) on video processing speed.
- Ability to customize system parameters (e.g. max_image_size, frame_interval, recognition_threshold) to adapt to different performance and accuracy requirements.

Thus, it can be concluded that the first version of the software has been successfully implemented and fulfills its intended purposes. The system demonstrates stable operation and allows for the effective detection of specified characters in a video stream.

The obtained results open up broad prospects for further development and application of the system in various fields, such as:

- Automatic marking of video archives;
- Security and video surveillance systems.
- Analysis of media content and identification of key characters.
- Creating personalized video collections.

4. Discussion

The developed system has a number of advantages that make it effective for use in various tasks:

- Ease of use: The system is based on widely known and available Python libraries, which makes it easy to configure.

- Efficiency: Thanks to the use of a pre-trained neural network and optimized video processing algorithms, the system ensures high speed and accuracy of facial recognition.
- Flexibility: The system allows you to customize video processing parameters (e.g. image size, frame interval, recognition threshold) to adapt to different performance and accuracy requirements.
- Modularity: The modular architecture of the system simplifies its expansion and modification to solve new problems.

However, it's worth noting that the system's first version has narrowly focused functionality and is designed solely for searching for faces in a video stream. It also has some limitations that must be taken into account when using it:

- Dependence on the quality of reference images: Face recognition accuracy is directly dependent on the quality of the reference photos. Blurry, poorly lit, or rotated photos can reduce recognition accuracy.
- Limitations of the pre-trained model: The system uses a pre-trained neural network, which may not be optimal for face recognition under certain conditions (e.g., low lighting, strong changes in perspective, or the presence of occlusions).
- Computational complexity: Video processing requires significant computing resources, which can be a problem when using the system on low-power devices.

As we work to improve the system's performance and expand its functionality, the following areas of development can be identified:

- using more modern neural network architectures. Replacing a pre-trained model with a more modern architecture (e.g., ResNet, EfficientNet) can improve recognition accuracy and expand its areas of application, allowing it to recognize not only faces, but also more specific objects, such as certain phenomena in physics experiments.
- Development of adaptive algorithms. The development of algorithms that automatically adjust system parameters (e.g. image size, frame interval, recognition threshold) depending on the characteristics of the video stream will increase its efficiency and flexibility.
- GPU optimization. Using a GPU to accelerate computations can significantly improve video processing speed, especially when using complex neural networks.
- integration with other systems. Integration of the system with other systems (for example, video surveillance systems, face databases) will expand its functionality and scope of application.
- Development of a graphical user interface (GUI). Creating a user-friendly graphical interface will make it easier for users who do not have experience working with the command line to use the system.

The developed system has the potential for application both to applied tasks in the field of media analysis and in various fields of science: from experimental setups to social research.

5. Conclusion

This paper presents a system for automatic face detection in a video stream, based on the use of the face_recognition library and optimized to achieve high performance.

The developed system is a tool for searching faces in video streams, which can be used in various fields of science and engineering. Proposed development directions for the system, such as the use of more modern neural network architectures, the development of adaptive algorithms, and optimization for GPU operation, will expand its functionality and improve its efficiency, opening up new possibilities for its application in solving a wide range of problems in various fields of science and engineering.

The contribution of this work is to demonstrate the practical applicability of existing libraries and optimization methods for solving the real-world problem of face detection in video streams. The developed system can be used as a basis for creating more complex and functional applications in the field of video data analysis.

References

1. Moretti F. Distant Reading. - London: Verso, 2013. - 298 p.
2. Manovich L. Cultural Analytics: Analyzing Cultural Patterns in the Era of “More Media” // Domus, 2009. - No. 923. - P. 1-7.
3. Salt B. Film Style and Technology: History and Analysis. - London: Starword, 2009. - 398 p.
4. Adobe Premiere Pro - URL: https://www.adobe.com/products/premiere.html
5. Final Cut Pro - URL: https://www.apple.com/final-cut-pro/
6. Amazon Rekognition - URL: https://aws.amazon.com/rekognition/
7. Video Intelligence API documentation - URL: https://cloud.google.com/video-intelligence/docs
8. Adobe Systems Inc. Adobe Premiere Pro User Guide // Adobe Documentation. - 2023. - URL: https://helpx.adobe.com/premiere-pro/user-guide.html
9. Amazon Web Services. Amazon Rekognition Video Developer Guide. - 2023. - URL: https://docs.aws.amazon.com/rekognition/
10. Google Cloud. Video Intelligence API Documentation. - 2023. - URL: https://cloud.google.com/video-intelligence/docs
11. Wittenburg P., Brugman H., Russel A. ELAN: a Professional Framework for Multimodality Research // Proceedings of the 5th International Conference on Language Resources and Evaluation. - 2006. - P. 1556-1559.
12. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. - Sebastopol: O'Reilly Media, 2016. - 1024 p.
13. VideoANT Project. Video Annotation Tool // University of Minnesota. - URL: https://ant.umn.edu/
14. Cinemetrics. Film Measurement and Analysis // Cinemetrics Database. - URL: http://www.cinemetrics.lv/
15. CLARIAH Media Suite. Research Platform for Media Studies // CLARIAH Consortium. - URL: https://mediasuite.clariah.nl/
16. Geitgey A. Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning // Medium. - 2016. - URL: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78
17. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). - 2016. - P. 770-778.
18. Schroff F., Kalenichenko D., Philbin J. FaceNet : A Unified Embedding for Face Recognition and Clustering // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). - 2015. - P. 815-823