Journal of Electronic Research and Application, 2025, Volume 9, Issue 6
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Design and Implementation of a Python-Based LIVP Live Photo Conversion Tool

Jiaming Wang*

School of Artificial Intelligence and Computer Science, North China University of Technology, Beijing 100144, China

**Corresponding author: Jiaming Wang, 2264649885@qq.com*

Abstract: This paper presents the design and implementation of a Python-based conversion tool for Apple Live Photo files (.livp). The tool converts .livp files into standard image formats (JPEG/HEIC/PNG) by renaming them to .zip and extracting the static images while optionally removing the embedded short video (.mov). It supports one-click batch processing of all .livp files within a folder. The software adopts a modular design, consisting of file renaming, batch decompression, temporary file cleaning, and a Tkinter-based graphical user interface (GUI). A multithreading mechanism ensures a smooth user experience without interface freezing. With a compact size of approximately 10.7 MB, the software is easy to deploy and suitable for users who need to batch-convert Live Photos into static images. This paper introduces the system design, key implementation details, performance analysis, and potential improvements, providing references for developing lightweight format conversion applications.

Keywords: LIVP; Live photo; File conversion; Python; Tkinter

Online publication: December 16, 2025

1. Introduction

1.1. Background and motivation

With the rapid advancement of smartphone photography, Apple devices have popularized the use of Live Photos (commonly stored in the .livp format) [1,2]. This format encapsulates both a static image and a short video clip [3,4]. When no user interaction occurs, the file displays as a still image; upon a long press or activation of a specific control, the dynamic portion is revealed by playing the embedded video segment. Although Live Photos enhance the expressive quality of images, the .livp format suffers from poor compatibility outside the Apple ecosystem, such as on Windows or Android platforms, where it cannot be opened directly with standard image viewers. This limitation complicates image management, bulk archiving, and cross-platform utilization.

1.2. Research significance and objectives

Batch conversion of .livp files into common static image formats such as JPG, HEIC, or PNG can greatly enhance

image accessibility and cross-platform interoperability, allowing users across different systems to view images with ease [5]. The objective of this study is to design and implement a lightweight, stable, and user-friendly tool capable of performing one-click, directory-level batch conversions. The software features a clean and intuitive graphical interface that ensures both high operational efficiency and ease of use for non-technical users.

1.3. Main contributions

Based on an in-depth analysis of the LIVP file structure, a simple yet reliable batch conversion workflow is proposed and implemented using Python, and the resulting tool supports both command-line and graphical user interface modes with comprehensive logging and robust exception handling to ensure stable operation and improved user experience. The graphical interface is implemented with Tkinterto provide a concise, easy-to-use front end that allows non-technical users to perform directory-level, one-click conversions with minimal learning effort [6]. The application is packaged with PyInstaller into a compact standalone executable that embeds the Python runtime and necessary dependencies, removing the requirement for a preinstalled Python environment and simplifying distribution; by selecting only essential libraries and excluding unnecessary modules, the packaged release maintains a small footprint (approximately 10.7 MB) while retaining full functionality [7].

2. Analysis of LIVP file structure and conversion principle

2.1. LIVP file format analysis and conversion process

As shown in Figure 1, the LIVP file is essentially a container format that typically includes a static image (commonly in JPEG format) and a short MOV video clip. From the perspective of binary structure and file encapsulation, the LIVP format can be regarded as a ZIP-compatible compressed container. By simply renaming the file extension from '.livp' to '.zip', the Live Photo file can be transformed into a standard compressed archive. Extracting this archive reveals the contained static image and video files.

After analyzing the structure of LIVP files, this study introduces an automated and efficient conversion workflow. The program scans the selected directory for ".livp" files, batch-renames them to ".zip," and extracts their contents into organized subfolders. During extraction, it identifies static images (JPEG or HEIC) and, based on user settings, may delete the embedded MOV videos and temporary ZIP files. Real-time logs record progress, duration, and any exceptions. This streamlined process ensures lossless conversion, efficient file management, and convenient one-click batch processing for an improved user experience.

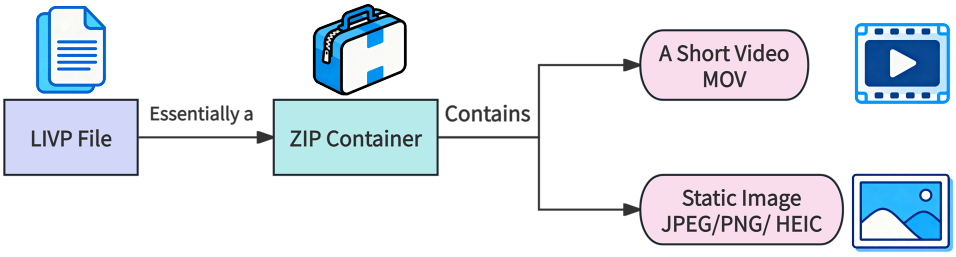

Figure 1. Structure and conversion process of a LIVP file.

2.2. Safety and legal considerations

In developing this tool, safety and legal compliance were considered. This software is intended solely for educational and personal data processing purposes, and users must comply with applicable data protection and

copyright regulations. As the program performs batch renaming, extraction, and deletion, it is recommended to back up all original files before use to prevent data loss. The tool is designed to improve the accessibility and cross-platform compatibility of Live Photos rather than to modify or commercially exploit the original media.

3. System design and implementation

The overall architecture of the system is composed of two main components: the core processing module and the GUI module. Following a modular design principle, the structure ensures maintainability, scalability, and clear separation of functionality. The program supports both command-line and graphical operation modes, allowing users to either perform automated batch conversions or operate through an intuitive interface. The tool is developed in Python 3, leveraging standard libraries such as os, zipfile, and threading for file management and multithreaded execution. The GUI is implemented using Tkinter, and the entire application is packaged with PyInstaller to generate a standalone executable that runs on systems without a preinstalled Python environment. This design achieves high portability and cross-platform compatibility while maintaining a compact footprint.

The table and workflow diagram together illustrate the overall architecture and logic of the LIVP conversion tool. Each module performs distinct yet connected tasks that ensure efficiency and stability. The process_livp_files module acts as the core controller, coordinating renaming, extraction, and cleanup operations described in **Table 1**. As shown in **Figure 2**, the system follows a sequential modular workflow that forms a clear processing pipeline, from renaming and extracting .livp files to optional video deletion and image generation, achieving high-speed batch conversion with minimal user input.

The design emphasizes exception handling and user interaction. Invalid paths and corrupted files are automatically detected and reported via logs and prompts, while backup reminders reduce data loss risks. The integrated logging system provides continuous feedback for progress tracking and troubleshooting. For deployment, the software is packaged with PyInstaller into a single 10.7 MB executable containing only essential dependencies, ensuring both efficiency and portability.

Table 1. Core modules and functional descriptions of the LIVP conversion tool

Module name	Function overview	Key features / notes
livp_to_zip()	Renames all .livp files to .zip format for further extraction	Handles naming conflicts and permission errors; records renaming logs
extract_all_zips()	Extracts all ZIP archives in the target folder	Supports subfolder extraction; detects corrupted files automatically
delete_mov_files()	Removes all .mov video files from extracted folders	Provides detailed deletion logs and ensures safe cleanup
process_livp_files()	Controls the entire conversion workflow and logs progress	Integrates renaming, extraction, and cleanup operations in sequence
LivpConverterGUI	Provides the graphical interface for folder selection and conversion	Maintains interface responsiveness using multithreading

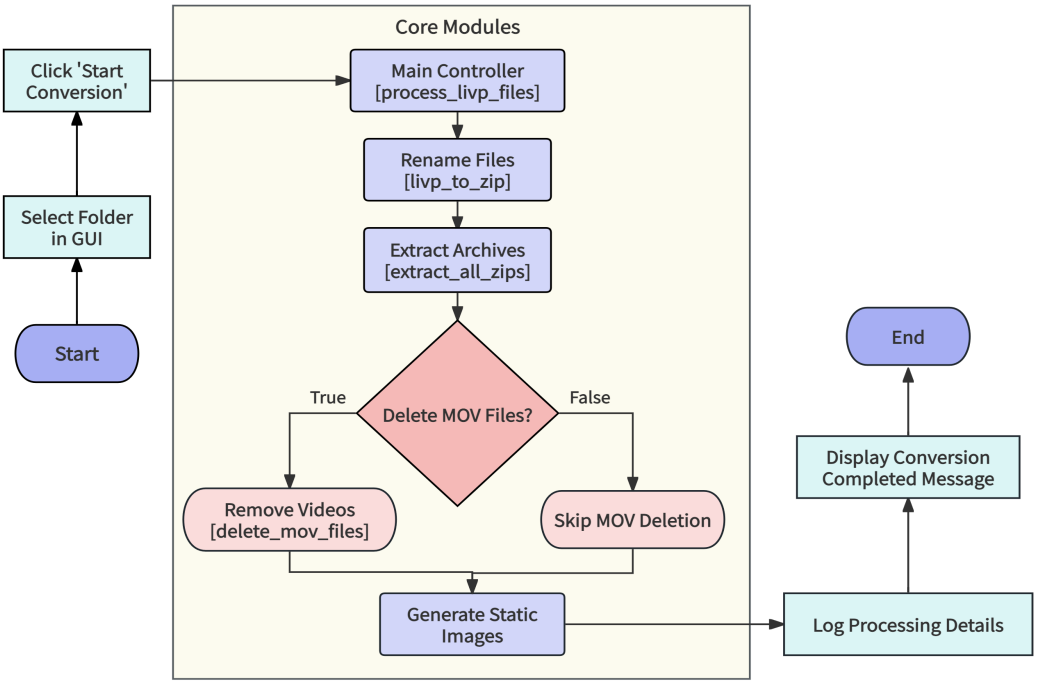

Figure 2. Workflow of the LIVP file conversion tool.

4. Functional testing and performance evaluation

4.1. Testing environment and methodology

The testing was carried out on a Windows 11 environment using folders containing multiple .livp files of different sizes. The test dataset included both normal and corrupted files to evaluate the robustness of the conversion process. The evaluation mainly focused on several performance indicators, such as conversion success rate, average processing time per file, memory usage, interface responsiveness, and exception handling capability.

4.2. Experimental example

During the experiment, a sample folder containing 300 .livp files with a total size of approximately 2.50 GB was selected for testing. Each file was automatically renamed, extracted, and processed through the conversion pipeline. **Figure 3** provides a visual demonstration of the actual software interface and the overall conversion process, showing how multiple .livp files are efficiently converted into standard image formats such as JPG or HEIC. After conversion, only the static image was retained, and all temporary files were deleted according to user configuration. The tool achieved a 100% success rate with an average processing time of about 0.062 seconds per file. The entire process was completed in 18.76 seconds, and detailed progress and error logs were recorded throughout execution.

Figure 3. Visualization of the LIVP file conversion process.

5. Conclusion

The test results show that the program can reliably process standard .livp structures and automatically skip invalid or damaged files without interruption. When encountering naming conflicts or insufficient access permissions, the system records the issue in the log file and continues processing other items. Although the current version is designed primarily for Windows platforms, it demonstrates strong stability and efficiency. Future versions will consider cross-platform adaptation for macOS and Linux systems.

Acknowledgments

The author would like to express sincere gratitude to the instructors and peers from the School of Artificial Intelligence and Computer Science, North China University of Technology, for their valuable guidance and support during the completion of this study.

Funding

The 2025 Beijing College Students' Innovation and Entrepreneurship Training Program (Project No.: S202510009001)

Disclosure statement

The author declares no conflict of interest.

References

[1] Apple Inc., n.d., Live Photos. https://developer.apple.com/design/human-interface-guidelines/live-photos

[2] Apple Inc., n.d., “LivePhoto” (UTType). https://developer.apple.com/documentation/uniformtypeid entifiers/uttype-swift.struct/livephoto

[3] Filext, n.d., LIVP File Extension: What is it? How to Open a LIVP File? https://filext.com/file-extension/LIVP

[4] Stack Overflow Contributors, 2015, Apple Live Photo File Format. https://stackoverflow.com/questions/32508375/apple-live-photo-file-format

[5] Library of Congress Preservation, 2020, High Efficiency Image File Format, HEIC/HEIX Brands. https://www.loc.gov/preservation/digital/formats/fdd/fdd000526.shtml

[6] Python Software Foundation, n.d., Tkinter: Python Interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html

[7] PyInstaller Developers, n.d., PyInstaller Manual. https://www.pyinstaller.org/