BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en Vol. 29 No. 2. 2024

UDC 004.91
DOI: 10.62660/bcstu/2.2024.45

Cross-platform adaptation of algorithmic editing techniques

Xiang Li*
Postgraduate Student
Chiang Mai University
50200, 239 Huay Kaew Rd., Chiang Mai, Thailand
https://orcid.org/0009-0000-7073-3646
Tamprasirt Anukul
PhD
Chiang Mai University
50200, 239 Huay Kaew Rd., Chiang Mai, Thailand
https://orcid.org/0009-0005-5424-1170
Fangli Ying
PhD
Chiang Mai University
50200, 239 Huay Kaew Rd., Chiang Mai, Thailand
https://orcid.org/0000-0001-8390-3229

Abstract. The research relevance is determined by the rapid development of technology and the growing need for efficient data processing on various platforms. The study aimed to address methods that would enable the use of data editing algorithms on various operating systems and hardware platforms. A methodology was developed for studying cross-platform adaptation technologies, including cross-compilation, virtualisation, the use of universal libraries and Application Programming Interface, as well as methods for testing and optimising algorithm performance. The study addressed various approaches to implementing cross-platform compatibility, including the use of cross-compilation, virtualisation and containerisation. The main technical challenges are managing resources, optimising performance and ensuring compatibility with hardware from different platforms. The principles included selecting the most appropriate technology for the task at hand, considering performance and security requirements, and ensuring effective integration of existing systems and infrastructure. The workflows are focused on creating modular and extensible solutions that can easily adapt to changes in the technological environment and user requirements. In the context of this study, artificial intelligence software plays a key role in improving the efficiency and accuracy of data processing across different platforms. The results showed that artificial intelligence software can automate various stages of the video and audio editing process. Artificial intelligence is used to analyse large amounts of content data, such as video files, images and audio recordings. The study determined that artificial intelligence is increasingly relevant in various aspects of movie production. Artificial intelligence can analyse scripts, predict their potential success and suggest improvements using data from previous films and their commercial success

Keywords: optimisation; software; efficiency; compatibility; interface

Article's History: Received: 14.12.2023; Revised: 26.03.2024; Accepted: 27.05.2024.

INTRODUCTION

Modern multimedia production and cinematography require the efficient use of algorithmic editing technologies to create high-quality content. However, the diversity of operating systems and hardware platforms poses a challenge for developers to ensure that algorithms are compatible and efficient across devices. In

Suggested Citation:
Li, X., Anukul, T., & Ying, F. (2024). Cross-platform adaptation of algorithmic editing techniques. *Bulletin of Cherkasy State Technological University*, 29(2), 45-56. doi: 10.62660/bcstu/2.2024.45.

*Corresponding author

this context, the cross-platform adaptation of algorithmic editing methods becomes a necessary component to ensure the versatility and reliability of media content processing and editing solutions. The modern requirements to produce media content such as films, videos and animation imply the use of highly efficient algorithms for processing and editing video, audio and graphics. However, the diversity of devices and technology platforms creates challenges in ensuring the same quality and performance across all target devices. The issue includes several key aspects, each of which is important and requires special attention when developing and implementing multimedia technologies.

Firstly, differences in the hardware characteristics of different platforms are a critical factor. Each operating system and device have unique features in terms of processor, memory, graphics and input/output. This requires in-depth optimisation of algorithms for the specific technical conditions of each platform to ensure optimal performance and quality of media content processing. Secondly, the compatibility and interoperability of algorithms between different operating systems and devices are equally relevant. From Windows and mac Operation System (OS) to Linux and mobile platforms such as iOS and Android, it is worth addressing the differences in software and ensuring that algorithms work without losing functionality and performance on each platform. The third aspect concerns the efficient use of resources. Algorithms for algorithmic editing should be designed to efficiently use the available computing and memory resources of devices. This is especially relevant for large data volume processing, where it is necessary to ensure fast and stable information processing without significant resource consumption. Ensuring data security and protection is a critical aspect of adapting algorithmic editing methods. When dealing with confidential information or personal data, security standards must be strictly adhered to prevent data leaks or unauthorised access to sensitive information.

The problem in this area is the need for effective integration and compatibility of algorithms on different platforms. J.Z. Blanco & D. Lucrédio (2021) proposed a holistic approach to cross-platform software development. Y. Ma *et al.* (2022) addressed support for cross-platform real-time collaborative programming, describing the architecture, methods, and prototyping system. Areas requiring further research include effective methods for integrating algorithmic editing across platforms, optimising performance and ensuring interoperability. The need to optimise algorithms to ensure uniform performance across different hardware and software platforms is also an important task. I. Bieda & T. Panchenko (2022) analysed various artificial intelligence (AI) tools, their functionalities, as well as advantages and limitations when used on different platforms. R. Eugeni & P. Pisters (2020) explored how modern algorithms are changing approaches to video creation and editing, and how these changes affect the perception and use of video content in different environments. Ensuring the security and protection of data when using algorithms on different platforms requires further study. Another pressing issue in this area is the need to create efficient and compatible algorithms that can run seamlessly on different platforms. M. Sajbidor *et al.* (2023) considered the creation of a cross-platform application in Java and C++, which demonstrates the opportunities and challenges associated with the implementation of algorithms on different systems. H. Nguyen (2022) studied the compatibility of programming languages in the context of cross-platform development, emphasising the difficulties that arise when trying to ensure the smooth operation of software on different platforms. The development of universal standards for cross-platform interaction of algorithms requires further study. Another issue that requires further study is the need to develop effective strategies for testing and debugging algorithms on different platforms. Y. Zhang (2021) examined cross-platform methods in computer graphics that facilitate the creation of experimental films. N. Dey (2021) has studied cross-platform development using Qt 6 and modern C++, which allows implementing algorithms on different systems. The development of methods for automatic detection and elimination of errors in the cross-platform operation of algorithms requires additional consideration. One of the key issues is the need to develop effective strategies for managing resources and optimising performance when working with algorithms on different platforms. C. Rieger & T.A. Majchrzak (2019) explored ways to create a definitive framework for evaluating approaches to cross-platform application development, which focuses on the complexities and challenges of this process. A.A. Menegassi & A.T. Endo (2020) addressed automated tests of cross-platform mobile applications in various configurations, emphasising the importance of testing algorithms on different devices and environments. Another issue that requires further study is the need to develop effective methods for adapting the user interface and visualising algorithmic editing for different devices and screens.

The study aimed to analyse and create approaches that would allow data editing algorithms to be effectively applied on different operating systems and hardware platforms. The following tasks were set as part of this goal: analysing existing technologies and tools that support cross-platform data editing and studying the technical features and challenges associated with adapting algorithms for different platforms.

MATERIALS AND METHODS

The study is conducted on an analysis of the advantages and limitations of each technology, such as performance losses, and difficulty in integrating or maintaining functionality across different platforms.

The study included a detailed analysis of existing approaches to cross-compilation, virtualisation, the use

of universal libraries and Application Programming Interface (APIs), as well as methods for testing and optimising algorithm performance on different platforms. The analysis of technologies included an assessment of their applicability and effectiveness for ensuring that data editing algorithms work on different operating systems and devices. Particular attention is devoted to an analysis of the limitations of technologies, such as performance losses, difficulty in maintaining functionality across platforms and the need for additional resources for integration and use.

The study examined cross-platform tools such as Qt, Xamarin and Flutter. The capabilities and limitations of these tools for developing cross-platform applications were assessed. Qualitative analysis methods, such as case studies, were used to gain an in-depth understanding of the experiences and perspectives of people involved in the development of algorithmic editors. The following experts took part in the study:

1. Lead developer (software developer), 10 years of experience.

2. Project manager (project manager), 8 years of experience.

3. Test engineer (software tester), 6 years of experience.

4. Chief developer (software developer), 12 years of work experience.

5. Integration specialist (integration engineer), 7 years of work experience.

These interviews were used to gather detailed information about practical experience of using cross-platform adaptation technologies, identify their strengths and weaknesses, and receive recommendations on how to improve and optimise cross-platform application development processes.

The study included a survey that helped to determine the current state of cross-platform adaptation of algorithmic editing methods. The survey involved 50 professionals, including 23 women and 27 men, working in programming and software development, specialising in algorithmic editing. The survey included 4 blocks of questions (Table 1).

Table 1. Blocks of questions in the questionnaire

Block of questions	Questions
The first block	Do you use algorithmic editing methods in your work?
The second block	Have you encountered any difficulties in adapting algorithmic methods for different operating systems?
The third block	Have you ever had to change your algorithms depending on the platform on which they are executed?
The fourth block	Do you think it is important to ensure cross-platform compatibility in your project?

Source: compiled by the authors

In addition to these methods, the study adhered to strict ethical standards. All participants were informed about the purpose of the study and how their answers would be used. Confidentiality and anonymity of respondents were guaranteed. The study was conducted voluntarily, without discrimination based on religion, race or gender. The ethicality of the study was supported by references to relevant standards, including professional codes of ethics and the European Commission's Guidelines on Ethics and Data Protection (2021).

RESULTS

Cross-platform adaptation technologies play a key role in modern software development, enabling the creation of applications that run on different operating systems and hardware platforms. In a rapidly changing technological landscape where users expect applications to work seamlessly across all devices, cross-platform solutions are becoming essential. These technologies not only simplify the development process but also significantly reduce the time to market, providing a competitive advantage. The main technologies for cross-platform adaptation are cross-compilation, virtualisation and containerisation, as well as the use of cross-platform libraries and frameworks.

Cross-compilation is the process of compiling software code on one platform (called the host platform) for execution on another platform (called the target platform). This approach allows developers to create software for a variety of devices and operating systems, even if they do not have direct access to the target hardware. Cross-compilation is especially useful when developing applications for mobile devices or embedded systems. The benefits of cross-compiling include reduced development time, as developers can use powerful host systems to compile code, which significantly speeds up the process compared to compiling on less powerful target devices. The ability to test and debug code on the host platform makes it easy to identify and fix errors without the need for access to the target hardware. In addition, cross-compilation allows developers to create software for many different architectures and operating systems, such as ARM, x86, Windows, macOS and Linux.

The use of cross-compilers such as GNU Compiler Collection (GCC) and Low Level Virtual Machine (LLVM) is a common practice. GCC supports a wide variety of programming languages and platforms, providing flexibility and broad applicability. LLVM, being a modular and flexible compiler, also supports various architectures

and is used in Xcode for macOS and iOS development. Cross-compiling reduces development costs and simplifies software support, as the same code can be used for different platforms with minimal changes. This is especially relevant in the modern software development environment, where applications must be available on a variety of devices and operating systems, ensuring the same quality and performance.

Virtualisation and containerisation can be used to create isolated environments for running applications, which simplifies portability and compatibility between different operating systems and hardware platforms (Casalicchio & Iannucci, 2020). Virtualisation is the creation of virtual machines (VMs) that emulate separate hardware, allowing multiple operating systems to run on a single physical server. This provides flexibility and isolation, as each virtual machine operates independently, minimising the impact of one VM on the others. Containerisation, in turn, uses containers that provide lightweight, isolated environments for applications. Unlike VMs, containers do not emulate an entire operating system but use the kernel of the host OS, which makes them more efficient in terms of resource usage. Containers contain all the necessary components to run an application, including code, libraries, and dependencies, which ensures consistent behaviour in any environment.

Examples of virtualisation tools include VMware and VirtualBox, which can be used to run VMs with different operating systems on the same physical server. Examples of containerisation tools include Docker and Kubernetes. Docker provides a platform for creating, distributing, and running containers, ensuring their isolation and ease of deployment. Kubernetes, in turn, automates the deployment, scaling and management of containerised applications, providing high availability and scalability.

The main benefits of virtualisation and containerisation include the isolation of environments, which avoids conflicts between dependencies of different applications and simplifies the deployment and scaling of applications. These technologies improve safety, as isolated environments minimise the risk of impacting other parts of the system. They also provide high resource efficiency, as containers consume less memory and central processing unit (CPU) resources than VMs. The use of virtualisation and containerisation is key to modern Information Technology (IT) infrastructures, especially in the context of cloud computing and DevOps practices. These technologies enable developers and system administrators to effectively manage complex, multi-component applications, ensuring their reliability, security and performance across multiple platforms and environments.

Cross-platform libraries and frameworks allow developers to create applications that run on different operating systems with minimal changes to the code (Ameen & Mohammed, 2022). This is achieved using common APIs and development tools, which greatly simplifies the process of developing and maintaining applications for different platforms. Such libraries and frameworks provide unified interfaces and abstractions that hide the differences between operating systems, allowing developers to focus on the logic of the application rather than the specifics of each platform.

Examples of cross-platform tools include Qt, Xamarin, and Flutter. Qt is a powerful framework for creating graphical user interface applications that support multiple platforms, including Windows, macOS, Linux, iOS, and Android. It provides a set of tools and libraries for developing graphical interfaces, working with files, networking and much more, creating complex applications that look and work the same on all supported platforms.

Xamarin is a C# mobile application development platform that can be used to create applications for iOS and Android using a common code (Toasa *et al.*, 2023). Xamarin provides access to the native APIs of each platform, which provides all the features and functions of the operating systems, providing high performance and a native user experience.

Flutter is a framework from Google for creating native applications for mobile, web and desktop platforms using a single code in the Dart language (Kishore *et al.*, 2022). Flutter includes a substantial set of tools for creating animations and user interfaces, thus providing tools to create beautiful and responsive applications. One of the key advantages of Flutter is its ability to compile code into native machine code, which ensures high application performance.

The main benefits of using cross-platform libraries and frameworks include reduced development time and costs due to code reuse, a consistent user experience across platforms, and ease of maintenance and updating of applications. These tools allow developers to create high-quality applications that run on a variety of devices and operating systems while ensuring high performance and reliability. The study created a diagram showing the importance of different technologies in the process of cross-platform adaptation of algorithmic editing methods (Fig. 1).

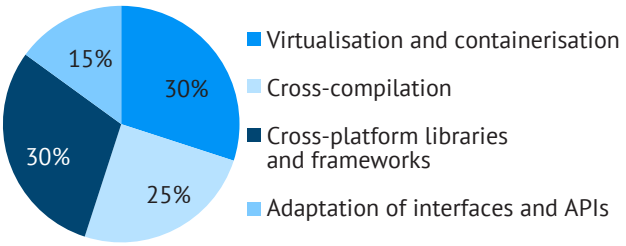

Figure 1. The importance and contribution of various technologies to the process of cross-platform adaptation of algorithmic editing methods
Source: compiled by the authors based on K. Vassallo *et al.* (2019)

The technical features and challenges of cross-platform adaptation of algorithmic editing methods are

complex and multifaceted tasks that require careful analysis and optimisation at different levels of development. Each platform has unique features in terms of processor, memory, graphics and data input and output. These differences require the optimisation of algorithms for the specific technical conditions of each platform. For instance, applications designed for desktop computers with powerful processors and large amounts of Random Access Memory may experience performance issues on mobile devices with more limited resources. Optimisation includes adapting algorithms to make efficient use of multithreading, minimising memory usage and improving graphics performance.

Ensuring that algorithms work on different operating systems, such as Windows, macOS, Linux, and mobile platforms (iOS, Android), requires implementation of compatibility and interoperability between different systems. This includes support for various system calls, file systems, network protocols and interfaces. Interoperability can be achieved using standard libraries and frameworks that abstract low-level details and provide unified APIs. Algorithms must efficiently use the available computing and memory resources of devices to ensure fast and stable operation when processing large amounts of data. This includes optimising code to reduce memory and CPU time, using hardware acceleration (e.g. GPUs) and ensuring low latency for interactive applications. Efficiencies can be achieved by profiling and analysing application performance across platforms and making changes accordingly. Data protection and security standards are essential when adapting algorithmic editing techniques, especially when dealing with sensitive information or personal data. This includes protecting data in transit and storage, managing access and authenticating users, and protecting against vulnerabilities and attacks. Security can be enhanced using encryption, multi-factor authentication and regular updates and patches to address vulnerabilities.

The results of the first block of the survey on the use of algorithmic editing methods show that out of 50 survey participants, 46% (23 respondents) regularly use algorithmic editing methods in their work. This confirms the high level of prevalence and importance of such approaches among professionals in the field of programming and software development. The remaining 54% (27 respondents) do not use algorithmic editing methods. This observation is relevant as it indicates the diversity of approaches and preferences in software development. Some professionals may prefer other methods or see no need to use algorithmic editing in their current work. These data indicate the need for further study and optimisation of algorithmic editing methods.

The results of the second block of the survey show that 62% (31 respondents) reported that they had encountered difficulties in adapting algorithmic methods for different operating systems. This shows the prevalence of cross-platform compatibility issues in software development. Such challenges may include API incompatibilities, differences in programming language standards, or hardware platforms. The remaining 38% of participants (19 respondents) did not encounter such difficulties, indicating fewer cross-platform compatibility issues.

In the third block of the survey, 50% (25 respondents) noted that they had to change algorithms depending on the platform on which they were executed. This highlights the importance of adapting algorithms to the specific technical features and requirements of different platforms, which can affect their efficiency and performance. The remaining 50% (25 respondents) did not express this need, which indicates that they were working on projects where the algorithms remained unchanged regardless of the platform. The results of the fourth block of the survey show that 78% (39 respondents) consider cross-platform compatibility to be an important or very important aspect of their project. The remaining 22% (11 respondents) stated that this is a less important aspect of software development for them. Based on the survey, a histogram was developed to visualise the key results and reflect the distribution of respondent opinions and experiences (Fig. 2).

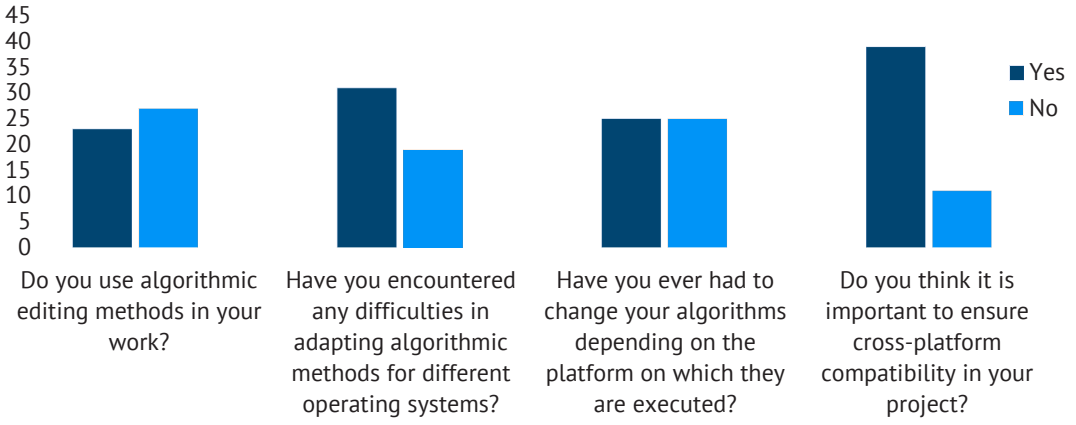

Figure 2. Results of answers to questions regarding the use of algorithmic editing methods
Source: compiled by the authors

Cross-platform adaptation of algorithmic editing methods requires addressing several technical features and challenges that need to be considered for the effective operation of applications on different operating systems and hardware platforms. One of the key aspects is the modularity and extensibility of solutions. Creating modular systems allows developers to update or replace individual software components without having to rewrite the entire system. This is especially relevant for cross-platform applications, where the need to adapt to different technical specifications and changes in the technology stack can arise at any time.

Another aspect is unification and standardisation. The use of unified standards and protocols, such as POSIX for operating systems, OpenGL for graphics, or JSON for data exchange, simplifies development and ensures application compatibility across platforms. This reduces development and support costs by reducing the number of adaptations and tests required on different devices. Another important aspect is performance optimisation. Cross-platform applications often require the adaptation of algorithms to different computing resources and hardware accelerators. For example, the use of Compute Unified Device Architecture for GPU-accelerated computing or specialised hardware accelerators for performing complex operations in real-time. This ensures high application performance across devices while maintaining functionality and efficiency. All these aspects combined ensure successful cross-platform adaptation of algorithmic editing methods, addressing a wide array of technical platforms and user requirements. This is necessary to create universal and stable solutions for processing media content and other applications running on different devices and operating systems.

Content analysis and optimisation using AI is a process that involves the automatic study and processing of large amounts of content data to improve its quality, style and relevance to the target audience across multiple platforms. Key examples of algorithms used for content optimisation include:

- Automatic colour correction – this algorithm automatically adjusts the colour palette of images or videos to achieve the desired visual effect. It can be used to eliminate colour deviations, balance hues, and adjust brightness and contrast, which is important for ensuring consistency and aesthetic perception of content across devices.
- Noise reduction – this algorithm aims to reduce noise in images or videos. Noise can occur due to insufficient lighting, poor sensor quality, or other technical reasons. The use of noise reduction techniques improves the clarity and quality of images, which is particularly important for maintaining the detail and clarity of images in different lighting conditions.
- Image quality enhancement – this algorithm includes various techniques aimed at increasing the clarity, contrast and detail of images (Li *et al.*, 2022). It can use resampling, filtering and edge enhancement techniques to enhance the overall visual experience of content, which also contributes to its aesthetic quality and appeal to viewers.

These algorithms work by analysing large amounts of content data to automatically optimise content for various purposes, from improving visuals to making it more relevant to audience requirements and platform specifications.

AI in the media is used in various aspects of the production, processing, distribution and consumption of multimedia content. The main areas of AI use in media include process automation, improving content quality, personalising user experience, creating new content products and optimising content management. The first area is content analysis. AI is used to recognise and classify media data such as images, video and audio files. This includes recognising objects and faces in images, analysing video content and detecting audio files, making it easier to automate the indexing and organisation of large volumes of content. The second important area is algorithmic editing. AI is used to automatically process and improve the quality of media content. Examples include automatic colour correction, noise reduction, sharpening and image detail, which help to achieve the desired visual effects and deliver high-quality content across multiple platforms and devices. The third area is content personalisation. AI is used to adapt media content to individual user needs and preferences. This includes the development of recommender systems that offer personalised content recommendations based on the analysis of user behaviour and preferences, which increases audience engagement and satisfaction with media services. The fourth area is content creation. AI is used to automatically create new media content, such as generating texts, images, music and videos. This includes the creation of various forms of multimedia content by algorithms, which promotes innovation in the field of creativity and expands the possibilities for creating unique content products. The fifth aspect is content management and distribution. AI helps to improve digital library management, copyright control, audience analysis, and optimisation of content distribution across various platforms and channels, which increases the efficiency of media resources and increases their commercial value.

The use of AI in the media significantly increases the efficiency of production processes, improves user experience and contributes to the creation of innovative solutions aimed at meeting the growing needs and expectations of modern users of multimedia technologies (de-Lima-Santos & Ceron, 2022). Synchronisation problems on TV can occur for several reasons related to the technical characteristics of different platforms and devices. These problems can affect the quality of content perception and require special measures to prevent them. One of the main reasons is the difference in hardware characteristics of the devices. Each

device has unique technical features, such as processor, memory, graphics capabilities and audio systems. These differences may result in inconsistencies in the playback of audio and video content. Network delays also play an important role in synchronisation problems. Differences in Internet connection speed and network stability can cause delays in data transmission, which is especially critical when streaming video and audio. Differences in the codecs and compression formats used to compress and decompress media files can lead to incompatibilities and additional delays when playing content on different platforms.

Another reason is software errors and bugs that occur in media content processing and playback systems. These errors can lead to incorrect synchronisation of audio and video signals, which can degrade the viewing experience. Adapting content for different devices and screens can also pose challenges. For example, content optimised for one type of screen may not display or play properly on another device. Various measures are taken to prevent these problems. One approach is to use unified standards and protocols for encoding and transmitting media data. This helps to ensure compatibility between different platforms and devices. Optimising algorithms for processing media content is also important. The use of hardware accelerators such as GPUs and FPGAs can significantly increase data processing speeds and reduce latency. Testing and debugging software on different platforms helps to identify and fix synchronisation issues before the product is released. This includes the use of automated tests and emulators of various devices and operating systems. It is also necessary to implement adaptive streaming mechanisms that can automatically adjust video and audio quality depending on the speed of the Internet connection and device characteristics.

Deep Fake is a technology that allows the creation of realistic fakes of videos and images using AI (Kwok & Koh, 2021). The process of creating a deep fake usually involves the following steps. The first stage is data collection. Deep fake creation requires large amounts of data, such as images or videos of faces to be replaced or synthesised. This data serves as the basis for model training, so its quality and diversity play a key role in the final result. This is followed by the data pre-processing stage. At this stage, the images and video are aligned and normalised to improve the quality of the model training. This can include cropping faces, adjusting lighting and colour, and removing noise. The purpose of pre-processing is to prepare the data so that the neural network can train efficiently, minimising errors. After that, the model training phase begins. Generative adversarial networks (GANs) are used in this stage, which consists of two neural networks – a generator and a discriminator. The generator creates fake images, and the discriminator tries to distinguish them from the real ones. These two networks are trained in a competitive mode: the generator tries to deceive the discriminator by creating more and more realistic images, and the discriminator improves its skills in recognising fakes. The training process continues until the generator learns to create sufficiently realistic images that the discriminator cannot distinguish them from real ones.

The next step is to create fake images or videos. The trained generator is used to create realistic images or videos by replacing the face in the original with the face of another person. In the case of video, this also requires lip movement and facial expressions to be synchronised to achieve maximum realism. At this stage, additional algorithms are often used to improve the quality, such as anti-aliasing filters and artefact correction. The final stage is post-processing. Includes final adjustments and enhancements to the video or image quality to eliminate artefacts and improve realism. This can include manual editing, the use of specialised tools to enhance images and audio, and additional checks for realism. Post-processing brings a fake video or image to a level where it will be difficult to distinguish it from the real material, even with careful analysis. Thus, the process of creating a deep fake involves several key stages, each of which plays an important role in ensuring the high realism of the final product. Data collection and pre-processing, training of GANs, creation of fake images and videos, and post-processing all combine to create high-quality fakes that can deceive even the most careful observers.

Creating Deep Fakes requires the use of various technologies, each of which is required in the process of generating realistic fakes. GANs are the main technology used to create Deep Fake (Revi *et al.*, 2021). GANs consist of two neural networks: generator and discriminator. Autoencoders also play an important role in the creation of Deep Fake. They are used to reduce data dimensionality and extract key facial features. Autoencoders consist of two parts: an encoder that converts the input data into a compact representation, and a decoder that recovers the original data from this representation. This simplifies the face replacement process, as autoencoders can effectively learn from face characteristics and then reproduce them with high accuracy.

Open-Source Computer Vision Library (OpenCV) is a computer vision library that is widely used in the process of creating Deep Fake. OpenCV provides a variety of tools for image and video processing, such as face detection, object tracking, lighting and colour correction, and noise filtering. These tools are necessary for pre-processing data to prepare it for training neural networks. TensorFlow and PyTorch are popular machine-learning frameworks used to create and train neural networks, including GANs and autoencoders (Novac *et al.*, 2022). These frameworks provide powerful tools and libraries that make it easy to develop and customise machine learning (ML) models. They support parallel computing and the use of graphics processing units (GPUs), which significantly speeds up the learning

process and enables large data amount processing. Dlib is a machine-learning library that includes tools for detecting faces and other objects in images (Xu *et al.*, 2020). It is widely used for pre-processing data and preparing it for training. Dlib provides highly accurate algorithms for face detection, image alignment and normalisation, which improves the quality of training and increases the accuracy of the fake images created. FaceSwap and DeepFaceLab are specialised tools and software for creating Deep Fake. They integrate the technologies described above and provide a user-friendly interface for working with them. These tools automate many aspects of Deep Fake creation, such as data collection, model training, fake image and video creation, and post-processing. FaceSwap and DeepFaceLab use GANs, auto-encoders, OpenCV, TensorFlow and PyTorch to ensure high realism and quality of the final product. Used together, these technologies and tools can be used to create Deep Fake with a high degree of realism, rendering them substantial in the hands of developers. However, they also raise important ethical and security issues, as they can be used to create misinformation and fraud.

Deep Fake detection and recognition is a complex task that requires the use of various technologies and methods. When creating Deep Fake, artefacts such as blurring, unnatural object boundaries, unrealistic lighting and shadows often occur. Deep Fake detection algorithms determine such anomalies in images and videos. Fake videos can have unnatural movements or behaviour of objects, such as inconsistent lip movements when words are spoken or facial expressions that do not match the words spoken. These features can be used to identify counterfeits. Deep Neural Networks (DNNs) are trained on large datasets, including real and fake videos and images. They study the characteristics that distinguish fakes from real content, such as skin textures, movement patterns and facial proportions. To protect the authenticity of content, digital signatures and hashing can be used to record and verify the originality and integrity of data. This can be used to track any changes or manipulations of media content. There are specialised software tools and platforms, such as DeepFaceLab, FaceForensics++, Reality Defender and others, that provide opportunities for analysing and detecting Deep Fake. They use a combination of the above methods to ensure accuracy and efficiency in detecting counterfeits. Together, these methods and technologies form a powerful toolkit to combat the problem of Deep Fake, providing protection against manipulation of media content and maintaining trust in digital data.

DISCUSSION

Cross-platform adaptation of algorithmic editing methods is a complex and multifaceted task that requires consideration of many aspects, from technical implementation to user experience. The main problem is the need to create algorithms that can run efficiently on different operating systems and hardware platforms without losing performance and data quality.

One of the key approaches to solving this problem is cross-compilation. This method can be used to generate executable code that can be executed on a variety of target platforms, providing versatility and independence from the source system. However, cross-compilation often faces incompatibility issues and requires significant resources to support different platforms. Similar conclusions were reached by M. Dempewolf (2020) in a study on the perception and experience of mobile developers when using cross-platform methods. As noted, the main reasons for choosing cross-platform solutions are cost reduction and accelerated development. The author emphasised the importance of ensuring compatibility and adequate testing when using cross-compilation, which is a key factor in successful cross-platform development.

Universal libraries and APIs are also significant in the cross-platform adaptation of algorithmic editing methods. These libraries and APIs provide a common programming interface that abstracts platform-specific details, making it much easier to develop and maintain algorithms. Despite some limitations, such as functionality and performance, they provide compatibility and integration with various platforms. The optimal use of universal libraries and APIs, combined with native platform-specific solutions, can provide the balance between versatility and performance needed for successful cross-platform adaptation. S. De (2021) also addressed the approach to designing a unified service API modelling, although for the semantic compatibility of inter-corporate automotive applications. The study results demonstrate that the use of a unified service API can significantly improve the interaction between different corporate systems, providing a higher degree of integration and compatibility.

The study identified several cross-platform mobile app development tools that have proven effective in adapting algorithmic editing methods. Flutter, developed by Google, supports iOS, Android and web applications, providing high performance using the Dart language and native user interfaces rendering. Flutter also provides a powerful set of development and debugging tools. Xamarin supports iOS, Android, and Windows, allowing developers to use C# and .NET to build cross-platform applications. The benefits of Xamarin include access to native APIs and integration with the .NET ecosystem, although performance may vary depending on the task. These tools were evaluated according to various criteria, including supported platforms, performance, versatility, available libraries and development tools. The choice of the right tool depends on the specific requirements of the project and the target platforms. Similar studies were conducted by M. Işıtan & M. Koklu (2020). The authors compared and evaluated various cross-platform mobile application development tools. This study demonstrated that

cross-platform tools have unique strengths and weaknesses, and developers should carefully choose a tool depending on the requirements of their projects, accounting for performance, ease of use, support for native features, and community accessibility. Y. Kyelem *et al*. (2021) investigated a hybrid approach to developing a cross-platform mobile interface for Infrastructure as a Service (IaaS). As concluded, a hybrid approach to developing a cross-platform mobile interface for IaaS can significantly improve flexibility, reduce costs, and speed up the development process. However, for this approach to be successful, potential integration and interoperability challenges need to be considered.

This study determined that the application of AI in media is an important tool used for the production, processing, distribution and consumption of multimedia content. Key areas of AI use in media include task automation, improving media quality, adapting content to user preferences, creating new forms of content and optimising content management processes. AI can automate many routine and labour-intensive processes in the media industry. In the field of image processing, AI automatically improves image quality, removes noise and corrects lighting, which simplifies the process of preparing visual content. AI plays a key role in creating a personalised experience for users. AI-powered recommendation systems analyse user behaviour and preferences to offer relevant content, which increases audience engagement and improves user experience. K. Nader *et al*. (2024) drew similar conclusions but did so through a different approach. The scientists analysed how AI is represented in entertainment media and how these representations influence public perception and understanding of AI. J.V. Pavlik (2023) addressed the impact of using generative AI, such as ChatGPT, on the fields of journalism and media education. The results demonstrated the potential benefits and challenges of using generative models in journalism and media education. They can also help to develop guidelines for the effective and responsible use of such technologies, contributing to the development of more sustainable and ethical practices in these industries.

The study revealed how deep fakes are created and what technologies are used for this purpose. Deep fake is a technology based on the use of AI and ML that can be used to create fake videos or images by replacing the faces of some people with the faces of other people with a high degree of realism. The study determined that various technologies such as DNNs, Autoencoders, GANs and software platforms such as TensorFlow and PyTorch are used to create deep fakes. The study also highlighted that the ability to create such fake videos and images raises significant ethical and legal issues. Potential abuses include spreading disinformation, defamation and fraud. This underscores the need to develop new legislative measures and technologies to detect and prevent the use of deep fakes for malicious purposes, protect personal information and ensure security in the digital space. C.-C. Hsu *et al*. (2020) drew similar conclusions, however, they focused on detecting fake images and analysing threats related to deep fakes. The authors addressed the development and analysis of deep learning methods for detecting fake images created using GANs and other similar approaches.

The conducted study also identified which methods and neural network architectures are best suited to detecting different types of fake images. S. Negi *et al*. (2021) addressed and analysed the phenomenon of deep fakes – technologies that allow the creation of realistic fake images and videos using ML and computer vision. As such, the features and potential threats associated with the distribution of such content in the digital environment were identified.

In general, study results demonstrate how cross-platform algorithmic editing methods contribute to the creation of universal tools available for different operating systems and devices. This makes editing technologies more accessible to users with different preferences and needs. In addition, the study not only expands the technological horizons in the field of content editing but also helps to improve the user experience and develop new innovative solutions.

CONCLUSIONS

The study identified key technologies and technical features related to algorithmic video editing, the use of AI in media, and methods to improve the accuracy of working with large amounts of data. The issue of creating and detecting deep fake technologies was also considered. Cross-platform adaptation of algorithmic video editing methods allows algorithms to be effectively applied on different devices and platforms, ensuring the same high quality of processing and editing. AI in media is used to automate tasks, improve content quality, personalise user experience and optimise content management processes. This includes automatic image and video enhancement, user behaviour analysis to offer personalised content and other features. AI software is a key tool for a variety of media tasks, including content creation, editing, analysis and trend prediction.

The survey results highlight the need for further development and improvement of cross-platform adaptation technologies in the field of algorithmic editing. They also point to the current challenges faced by specialists in this field, including integration difficulties and the need for constant optimisation of algorithms. Deep Fake is a technology for generating fake images and videos using deep learning. It uses neural networks to create realistic fakes, which pose security and trust risks. Specialised software and technologies are being developed to detect and recognise Deep Fake, including methods of analysing deep textures and behavioural patterns, as well as the use of computer vision and ML technologies. Further research includes the development of universal libraries and frameworks that can simplify the integration of algorithmic editing

across platforms, minimising the need for changes to the application source code. The study is limited to the selection of certain technologies and frameworks, which excludes other potentially effective solutions for cross-platform adaptation.

ACKNOWLEDGEMENTS
None.

CONFLICT OF INTEREST
None.

REFERENCES

[1] Ameen, S., & Mohammed, D. (2022). Developing cross-platform library using flutter. *European Journal of Engineering and Technology Research*, 7(2), 18-21. doi: 10.24018/ejeng.2022.7.2.2740.
[2] Bieda, I., & Panchenko, T. (2022). A systematic mapping study on artificial intelligence tools used in video editing. *International Journal of Computer Science and Network Security*, 22(3), 312-318. doi: 10.22937/IJCSNS.2022.22.3.40.
[3] Blanco, J.Z., & Lucrédio, D. (2021). A holistic approach for cross-platform software development. *Journal of Systems and Software*, 179, article number 110985. doi: 10.1016/j.jss.2021.110985.
[4] Casalicchio, E., & Iannucci, S. (2020). The state-of-the-art in container technologies: Application, orchestration and security. *Concurrency and Computation: Practice and Experience*, 32(17), article number e5668. doi: 10.1002/cpe.5668.
[5] De, S. (2021). *Design approach to unified service API modeling for semantic interoperability of cross-enterprise vehicle applications*. Plzeň: University of West Bohemia.
[6] de-Lima-Santos, M.-F., & Ceron, W. (2022). Artificial intelligence in news media: Current perceptions and future outlook. *Journalism and Media*, 3(1), 13-26. doi: 10.3390/journalmedia3010002.
[7] Dempewolf, M. (2020). *Perceptions of mobile developers adopting cross-platform methods: A generic qualitative inquiry*. (Doctoral dissertation, Capella University, Minneapolis, USA).
[8] Dey, N. (2021). *Cross-platform development with Qt 6 and Modern C++: Design and build applications with modern graphical user interfaces without worrying about platform dependency*. Birmingham: Packt Publishing.
[9] Eugeni, R., & Pisters, P. (2020). The artificial intelligence of a machine: Moving images in the age of algorithms. *European Journal of Media Studies*, 1, 91-100. doi: 10.25969/mediarep/14325.
[10] European Commission. (2021). *Ethics and data protection*. Retrieved from https://ec.europa.eu/info/funding-tenders/opportunities/docs/2021-2027/horizon/guidance/ethics-and-data-protection_he_en.pdf.
[11] Hsu, C.-C., Zhuang, Y.-X., & Lee, C.-Y. (2020). Deep fake image detection based on pairwise learning. *Applied Sciences*, 10(1), article number 370. doi: 10.3390/app10010370.
[12] Işıtan, M., & Koklu, M. (2020). Comparison and evaluation of cross platform mobile application development tools. *International Journal of Applied Mathematics, Electronics and Computers*, 8(4), 273-281. doi: 10.18100/ijamec.832673.
[13] Kishore, K., Khare, S., Uniyal, V., & Verma, S. (2022). Performance and stability comparison of react and flutter: Cross-platform application development. In *2022 international conference on cyber resilience (ICCR)* (pp. 1-4). Dubai: Institute of Electrical and Electronics Engineers. doi: 10.1109/ICCR56254.2022.9996039.
[14] Kwok, A.O.J., & Koh, S.G.M. (2021). Deepfake: A social construction of technology perspective. *Current Issues in Tourism*, 24(13), 1798-1802. doi: 10.1080/13683500.2020.1738357.
[15] Kyelem, Y., Kabore, K.K., & Bassole, D. (2021). Hybrid approach to cross-platform mobile interface development for IAAS. In S. Shakya, R. Bestak, R. Palanisamy & K.A. Kamel (Eds.), *Mobile computing and sustainable informatics* (pp. 225-238). Singapore: Springer. doi: 10.1007/978-981-16-1866-6_16.
[16] Li, C., Zhu, J., Bi, L., Zhang, W., & Liu, Y. (2022). A low-light image enhancement method with brightness balance and detail preservation. *PLoS ONE*, 17(5), article number e0262478. doi: 10.1371/journal.pone.0262478.
[17] Ma, Y., Yang, Z., Chiu, B., Zhang, Y., Jiang, J., Du, B., & Fan, H. (2022). Supporting cross-platform real-time collaborative programming: Architecture, techniques, and prototype system. In H. Gao & X. Wang (Eds.), *Collaborative computing: Networking, applications and worksharing* (pp. 124-143). Cham: Springer. doi: 10.1007/978-3-030-92638-0_8.
[18] Menegassi, A.A., & Endo, A.T. (2020). Automated tests for cross-platform mobile apps in multiple configurations. *IET Software*, 14(1), 27-38. doi: 10.1049/iet-sen.2018.5445.
[19] Nader, K., Toprac, P., Scott, S., & Baker, S. (2024). Public understanding of artificial intelligence through entertainment media. *AI & SOCIETY*, 39(2), 713-726. doi: 10.1007/s00146-022-01427-w.
[20] Negi, S., Jayachandran, M., & Upadhyay, S. (2021). Deep fake: An understanding of fake images and videos. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(3), 183-189. doi: 10.32628/CSEIT217334.
[21] Nguyen, H. (2022). *Programming language interoperability in cross-platform software development*. (Master's thesis, Aalto University, Espoo, Finland).

[22] Novac, O.-C., Chirodea, M.-C., Novac, C.-M., Bizon, N., Oproescu, M., Stan, O.P., & Gordan, C.E. (2022). Analysis of the application efficiency of TensorFlow and PyTorch in convolutional neural network. *Sensors*, 22(22), article number 8872. doi: 10.3390/s22228872.
[23] Pavlik, J.V. (2023). Collaborating with ChatGPT: Considering the implications of generative artificial intelligence for journalism and media education. *Journalism & Mass Communication Educator*, 78(1), 84-93. doi: 10.1177/10776958221149577.
[24] Revi, K.R., Vidya, K.R., & Wilscy, M. (2021). Detection of deepfake images created using generative adversarial networks: A review. In M. Palesi, L. Trajkovic, J. Jayakumari & J. Jose (Eds.), *2nd international conference on networks and advances in computational technologies* (pp. 25-35). Cham: Springer. doi: 10.1007/978-3-030-49500-8_3.
[25] Rieger, C., & Majchrzak, T.A. (2019). Towards the definitive evaluation framework for cross-platform app development approaches. *Journal of Systems and Software*, 153, 175-199. doi: 10.1016/j.jss.2019.04.001.
[26] Sajbidor, M., Vesely, P., & Krajewski, M. (2023). Creating cross-platform application in Java and C++. In N. Kryvinska, M. Greguš & S. Fedushko (Eds.), *Developments in information and knowledge management systems for business applications* (pp. 495-540). Cham: Springer. doi: 10.1007/978-3-031-27506-7_19.
[27] Toasa, R.M., Egas, P.F.B., Recalde, H., & Saltos, M.G. (2023). Mobile development with Xamarin: Brief literature, visualizations and important issues. In Á. Rocha, C. Ferrás & W. Ibarra (Eds.), *Information technology and systems* (pp. 299-307). Cham: Springer. doi: 10.1007/978-3-031-33261-6_26.
[28] Vassallo, K., Garg, L., Prakash, V., & Ramesh, K. (2019). Contemporary technologies and methods for cross-platform application development. *Journal of Computational and Theoretical Nanoscience*, 16(9), 3854-3859. doi: 10.1166/jctn.2019.8261.
[29] Xu, M., Chen, D., & Zhou, G. (2020). Real-time face recognition based on Dlib. In *Innovative computing* (pp. 1451-1459). Singapore: Springer. doi: 10.1007/978-981-15-5959-4_177.
[30] Zhang, Y. (2021). *Cross-platform methods in computer graphics that boost experimental film making*. (Master's thesis, Rochester Institute of Technology Rochester Institute of Technology, Rochester, USA).

Кросплатформна адаптація алгоритмічних методів редагування

Сян Лі
Аспірант
Чіангмайський університет
50200, дорога Хуай Каев, 239, м. Чіангмай, Таїланд
https://orcid.org/0009-0000-7073-3646
Тампрасірт Анукуль
Доктор філософії
Чіангмайський університет
50200, дорога Хуай Каев, 239, м. Чіангмай, Таїланд
https://orcid.org/0009-0005-5424-1170
Фанлі Ін
Доктор філософії
Чіангмайський університет
50200, дорога Хуай Каев, 239, м. Чіангмай, Таїланд
https://orcid.org/0000-0001-8390-3229

Анотація. Актуальність дослідження зумовлена стрімким розвитком технологій та зростаючою потребою в ефективній обробці даних на різних платформах. Метою дослідження була розробка методів, які б дозволили використовувати алгоритми редагування даних на різних операційних системах та апаратних платформах. Було розроблено методологію дослідження технологій кросплатформної адаптації, включаючи крос-компіляцію, віртуалізацію, використання універсальних бібліотек та інтерфейсу прикладного програмування, а також методи тестування та оптимізації продуктивності алгоритмів. У дослідженні були розглянуті різні підходи до реалізації крос-платформної сумісності, включаючи використання крос-компіляції, віртуалізації та контейнеризації. Основними технічними проблемами є управління ресурсами, оптимізація продуктивності та забезпечення сумісності з апаратним забезпеченням різних платформ. Принципи включають вибір найбільш підходящої технології для вирішення поставленого завдання, врахування вимог до продуктивності та безпеки, а також забезпечення ефективної інтеграції існуючих систем та інфраструктури. Робочі процеси зосереджені на створенні модульних і розширюваних рішень, які можуть легко адаптуватися до змін у технологічному середовищі та вимог користувачів. У контексті цього дослідження програмне забезпечення зі штучним інтелектом відіграє ключову роль у підвищенні ефективності та точності обробки даних на різних платформах. Результати показали, що програмне забезпечення зі штучним інтелектом може автоматизувати різні етапи процесу редагування відео та аудіо. Штучний інтелект використовується для аналізу великих обсягів даних контенту, таких як відеофайли, зображення та аудіозаписи. Дослідження показало, що штучний інтелект стає все більш актуальним у різних аспектах кіновиробництва. Штучний інтелект може аналізувати сценарії, прогнозувати їхній потенційний успіх і пропонувати покращення, використовуючи дані попередніх фільмів та їхній комерційний успіх

Ключові слова: оптимізація; програмне забезпечення; ефективність; сумісність; інтерфейс