A Framework for Art-directed Augmentation of Human Motion in Videos on Mobile Devices

R. Debski, O. Schmitt, P. Trenz, M. Reimann, J. Döllner, M. Trapp
Hasso Plattner Institute, Digital Engineering Faculty, University of Potsdam, Germany

A. Semmo, S. Pasewaldt
Digital Masterpieces, Potsdam, Germany

Figure 1: Stills from videos using different motion effects applied to user-generated content in Lumo (from left to right): Chalky Silhouette, Dotted Silhouette, Light Trails, Dr. Strange, Square, Marionette, Glow Sticks, Glow Sticks with black background

ABSTRACT

This paper presents a framework and mobile video editing app for interactive artistic augmentation of human motion in videos. While creating motion effects with industry-standard software is time-intensive and requires expertise, and popular video effect apps have limited customization options, our approach enables a multitude of art-directable, highly customizable motion effects. We propose a graph-based video processing framework that uses mobile-optimized machine learning models for human segmentation and pose estimation to augment RGB video data, enabling the rendering and animation of content-adaptive graphical elements that highlight and emphasize motion. Our modular framework architecture enables effect designers to create diverse motion effects that include body pose-based effects such as glow stick or light trail effects, silhouette-based effects such as halos and outlines, and layer-based effects that provide depth perception and enable interaction with virtual objects.

Keywords: Video Stylization, Mobile Processing, Human Motion, Depicting Dynamics, Video Effects

1 INTRODUCTION

User-generated short-form videos are one of the most influential formats on social media. While platforms such as TikTok and YouTube (Shorts) offer a variety of filters and visual effects, users still like using their imagination to create their own. Typically, industry-standard software such as Adobe After Effects or Resolve Fusion is used to create video effects and stylization, but their effective usage often requires in-depth knowledge and time. To produce especially motion-based video effects without such software, users require a significant amount of time and effort, such as in the #Glowstickdance trend [Cha20], in which participants adhered glow lights to their bodies to dance in the dark.

The objective of our work is to analyze, design, and implement a mobile application framework that facilitates the development and synthesis of motion visualizations suitable for user-generated content, such as those shown in Fig. 1. Our primary focus is on enabling users to film, edit, and share content instantly using mobile devices. To allow for rapid editing, we aim to automate the effect generation process, while at the same time retaining creative control over significant portions of the effect design. Such creative control should manifest in parameterized control over effect variables such as adjustments of color, range of the glow, or background modifications. Our ultimate goal is to open up the world of effect design to new audiences. Additionally, we aim to make the implementation and variation of effects easier for developers by creating a simple and flexible framework.

Problem Statement. To achieve our objective, we implement a framework that processes multidimensional video data to produce art-directed motion visualiza-

tions. In this endeavor, we identified the following challenges that our framework should address:

Interactivity / Usability: Most professional video editing programs, such as Adobe After Effects, are rich in functionality but also have a steep learning curve. Creating motion-based effects is often accomplished through time-consuming frame-by-frame editing. To ensure our app's suitability for user-generated content, we seek to create an application enabling the creation of complex effects with excellent visual quality, even by nonprofessional users. Nonprofessional video editing of user-generated videos is already possible with popular mobile apps such as TikTok or Instagram that offer customization options that enable users to adjust the size and order of effects. However, these options are often hidden behind several menus with different interfaces, leading to a complicated workflow. To address this issue, we aim to provide a user-friendly interface that allows users to apply, customize, and delete effects quickly and easily. To achieve this, we integrate all application options into a single menu that displays all options at once thereby avoiding complicated workflows while retaining a high level of creative control. Users can apply effects in sequence, reorder them, and adjust their timing. They can preview the effects in real-time and make adjustments to effect settings while observing the results. We achieve real-time performance and rendering to ensure that the preview is accurate and responsive. As our effect control options are specifically designed for motion effects, our interface can restrict the number of available options compared to industry-standard software.

Exchangeability/Adaptability: Motion effects in existing mobile apps such as TikTok offer limited customization options for the effect designer. To address this limitation, we introduce a highly adaptable framework that provides users with a wide range of customization options. Users have control over numerous parameters, such as color scheme, background color, and stylization. This adaptability is also reflected in the framework's design and enables the developer to create new and exciting effects or combine existing ones for the user to experiment with. The framework's different components, or processing nodes, can be easily interchanged and combined to achieve various visual outputs. If a modification is necessary, the code is simple to modify for new data and usage scenarios.

Approach & Contributions. Our technical approach consists of a framework for building video processing pipelines that allows for the real-time synthesis of content-aware motion visualizations. To this end, the Red-Green-Blue (RGB) video data is extended by extracted information such as depth, segmentation, or human body poses. To achieve real-time performance, our framework enables concurrent processing and presentation of such multidimensional data. To summarize, this paper presents

1. a framework for automatically creating content-aware video effects to visualize motion. Our approach uses semantic data such as human body pose and segmentation information to create content-aware effects that focus on a subject's movement.
2. a software system that processes multi-dimensional data simultaneously on a frame-by-frame basis. The framework hides complexity by encapsulating processing functionality in easily extendable processing nodes, organized in a graph-based structure to form processing pipelines.
3. an intuitive timeline-based user interface for art directing, applying, customizing, and removing content-aware effects on multi-dimensional videos without requiring domain knowledge. Our supplemental video[1] demonstrates the interface and interaction with the app.

The remainder of this work is structured as follows. Sec. 2 reviews related work with respect to motion visualization in general and motion-based video effects in particular. Sec. 3 analyzes popular real-world motion effects and derives semiotic aspects and designs for our video effects. Sec. 4 presents a conceptual overview of the proposed approach. Sec. 6 details implementation aspects of the framework. Sec. 7 evaluates the system's performance, presents a post-deployment user study on the intuitiveness and usability of the application, and discusses results and limitations. Finally, Sec. 8 concludes this work and presents ideas for future research.

2 BACKGROUND & RELATED WORK

Motion Visualization. The visualization of motion is a challenge to artists, scientists, and image creators alike, as Cutting *et al*. [Cut02] state in their analysis of contemporary artistic motion visualization. They suggest several effective ways to visualize motion, which include dynamic balance, multiple stroboscopic images, affine shear, photographic blur, and action lines. These techniques are based on principles of physics, perception, and visual design, and have been widely used in various applications. Nienhaus *et al*. [Nie05] propose an automated depiction system for visualizing dynamics in 3D scenes. They follow design principles found in visual art, such as those used in comic books, and

[1] https://drive.google.com/file/d/1ERgXif9aQDC_Ug_fVfxAxnUokNfKp89w

introduce visual metaphors and symbolization, such as object bending or squashing to depict motion in scenes. Bouver-Zappa *et al*. [Bou07] use 3D skeletal motion capture data to generate motion cues in a still image of the captured character using arrows, noise waves, and stroboscopic motion. Kwon *et al*. [Kwo12] further utilize the skeletal structure of the motion capture for improved motion cues. Schmid *et al*. [Sch10] introduce a method for visualizing motion by extending surface shaders to programmable motion effects with knowledge of the global spatio-temporal trajectory. Their approach enables the creation of complex motion effects, such as speed lines and blurring, that can be customized to different styles.

Motion-based Video Effects. In addition to their use in 3D scenes, motion visualization techniques, such as speed and focus lines, motion blur, ghosts, motion lines, and halos, are also utilized as photo and video effects. Collomosse *et al*. [Col03; Col05] present a system to render such motion cues using a variety of traditional feature-tracking-based computer vision techniques, whereas Nienhaus *et al*. [Nie08] use background subtraction and focus on the forward lean affine shear as a motion cue. Umeda *et al*. [Ume12] employ such techniques for real-time manga-like depiction in videos and the Vivid [Sem19] mobile app applies similar visual metaphors for depicting dynamics in live-photos. Lu *et al*. [Lu13] convert video sequences into movement depictions of annotated body parts using arrows and motion particles. However, these approaches are either one-shot non-interactive methods and thus not suitable to perform art-direction [Ume12; Col05; Lu13] or are only suitable for creating stylized images [Col05; Nie08; Lu13; Sem19]. Mayer *et al*. [May21] propose a mobile application for artistic silhouette visualization in videos that allows for interactive effect editing in a timeline-based graphical interface. We adopt a similar interface design for our Lumo mobile app. However, we enable a much wider range of content-adaptive effects and customization options, that allows for the interactive visualization of scenes, silhouettes, and human body poses. Our proposed processing pipeline design furthermore allows for easy extension and customization of new effects.

3 EFFECT ANALYSIS AND DESIGN

Analysis of Real-world Examples. To characterize motion-based video effects, we first examined actual instances of artwork that use additional motion graphics features in their artwork such as those shown in Fig. 2. We collected a large sample of videos and photos as reference, identified common semiotic aspects, and analyzed their composition into the video effects. In the analyzed video effects, common themes include playing with light, video annotations, and addition of artificial elements into the scene. All of these add new or

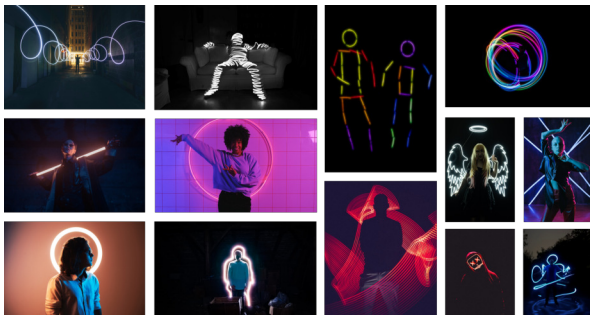


Figure 2: Overview of selected artworks that use motion-based video effects.

enhance already existing strong visual focus points to the image or video and often make it visually more appealing and striking.

Semiotic Aspects. Based on the analysis results, 11 semiotic aspects distinguishing motion-based elementary primitives and mechanisms were identified. They are visually summarized in Fig. 3 and described in the following.

Motion Effect-Type: The effect may be of a silhouette, skeleton or action line/light-trail, or generic annotation type.

Graphical Elements: Outlined shapes (e.g., triangles, squares, and circles) are formed by (poly) lines and curves, which can be represented by Bézier curves. Lines can also be dashed or dotted. Lines and shapes can appear once or in bigger numbers.

Stroke Weight: Lines may have different widths but are mostly of medium weight.

Base Color: Lines have saturated colors and hues mainly in reds, turquoise, blue, purple, greens, and yellows, which dominate the color palettes. Lines may exhibit varying degrees of transparency.

Texture: Lines are solid or can have patterns, such as a sketchy line.

Neon Glow: Lines may have the appearance of neon lights. This look is achieved by a brighter core in the line's center and a transparency gradient fall-off.

Background Brightness: The depicted scene may appear fully illuminated, dusky, or deeply dark.

Position: Annotations and lines can be offset from a position such as a point on or outline of a person.

Animation: Effect elements and their attributes can be animated meaning they change position, shape, color, etc. over time in a controlled manner.

Attachment: Effect elements such as shapes or lines appear to be affixed to anchor points on individuals in the scene, with their position dynamically animated to track the motion of the subjects. As a result, the elements remain temporally coherent,

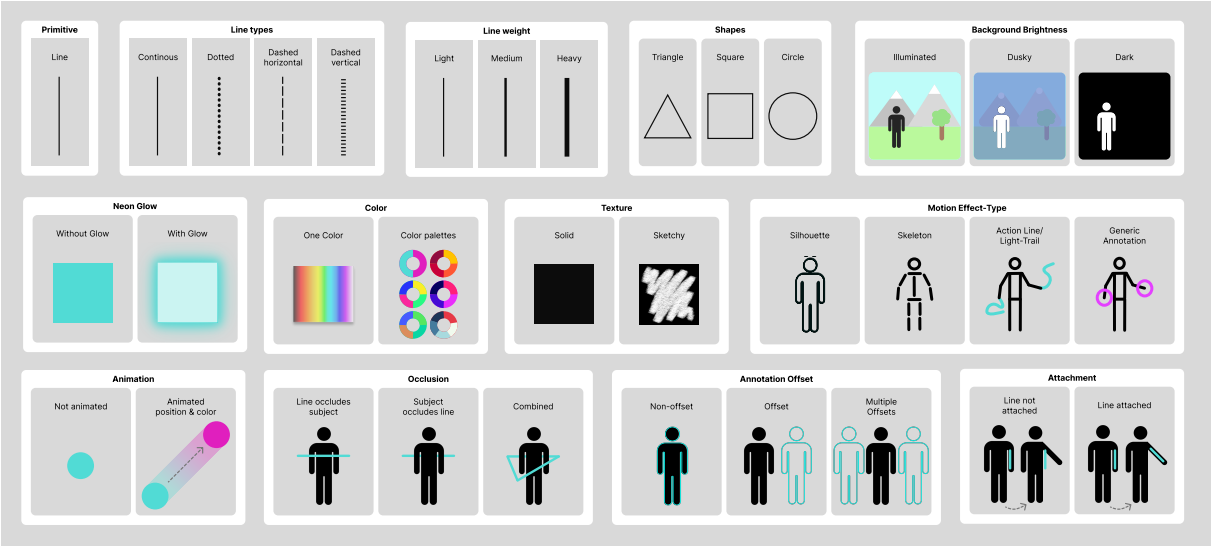


Figure 3: Summarizing presentation of semiotic aspects.

maintaining consistent attributes such as position, shape, and color, and do not change abruptly or unpredictably over time.

Occlusion: Effect elements can be occluded by subjects and objects in the depicted scene. The element thus appears to be behind the subjects or objects resulting in the illusion of depth.

Exemplary Effect Design. Combining the different semiotic aspects an exemplary effect design can be achieved as follows. Neon-colored lines are positioned on the limbs of a person in the video, anchored to the respective joints. Each line features a neon glow and medium stroke weight with a solid texture and opaque opacity. The lines are not occluded by the person, move coherently over time, and are not animated. The scene appears to be set at night. This results in an effect denoted as "Glowstick Effect", shown in Fig. 4, that is similar to glow sticks attached to the limbs of a person (e.g., [Cha20]).

Figure 4: "Glowstick Effect" Design.

4 FRAMEWORK

Based on the previous motion-based effect analysis and design, this section describes preliminaries and requirements (Sec. 4.1), the fundamental graph-based processing framework (Sec. 4.2), and the conceptual modeling of motion-based effects (Sec. 4.3).

4.1 Preliminaries & Assumptions

This section summarizes the functional and non-functional requirements for the framework's implementation.

Functional Requirements. We identified the following functional requirements for our framework: The application should be able to capture, process, and export multi-dimensional videos, i.e., RGB video streams with additional information channels. Those multi-dimensional videos should be loaded and stored persistently. Furthermore, work snapshots should be savable during effect creation and should be subsequently restorable and editable. The framework should enable the creation of silhouette and human pose-based video visualizations. To provide a user-friendly interface to the user, the application should present a Graphical User Interface (GUI) that allows to create, edit, and delete effects. Furthermore, the application should provide a real-time preview while editing.

Non-functional Requirements. We also identified the following non-functional requirements: The framework should be reliable in creating, modifying, deleting, and consistently reproducing effects. The framework must prioritize high usability, with an intuitive interface that requires no professional knowledge of video editing. From the developer's perspective, the framework must be maintainable, easily adaptable to different data and use cases, and efficiently extendable with new visualizations and additional parameters. Finally, the effects generated by the framework should be visually appealing, with a fluid and artistic appearance that enhances the overall visual characteristics of the video.

4.2 Graph-based Video Processing

To efficiently process our captured video frames we use a pipeline-based approach, i.e., a directed graph of pro-

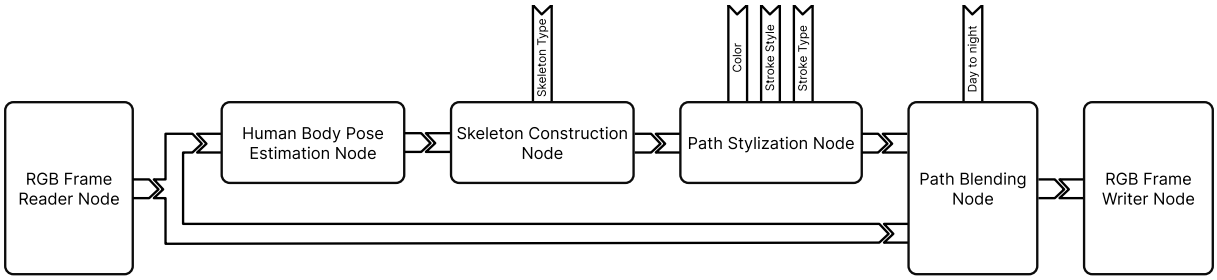


Figure 5: An exemplary pipeline that constructs the "Glowstick Effect".

cessing components similar to Lugaresi *et al*. [Lug19]. The processing graph allows a high degree of flexibility and easy implementation of different effects.

Node Port Reactivity. To allow interactivity, the inputs and outputs of the nodes are reactive, i.e., when a value at the input changes, the node processes it and passes the result forward via the output port. This results in a cascading change in the pipeline, leading to sequential or concurrent processing of nodes in the graph. To minimize processing requirements, a node only processes when an input value changes. A node can have more than one input and output port to increase flexibility. The processing of such multi-input nodes can be configured by the effect developer using a triggering rule. Common rules include the triggering based on either the change in any input port, a specific input port, a group of input ports, or all input ports since the last time the node underwent processing.

Graph Nodes for Specialized Processing Tasks. To illustrate the flexibility of the graph-based processing concept, we provide an exemplary processing pipeline that creates a stylized Bézier path from an RGB frame. To this end, we connect the processing nodes as shown in Fig. 5, and briefly describe them in the following:

Human Body Pose Estimation Node: It receives an RGB frame as input and transforms it into estimated human joint point positions in the frame, which are returned via the output port.

Skeleton Construction Node: It receives estimated joint point positions at its input port, filters out the relevant points, and subsequently creates a Bézier path, which is passed on via the output port.

Path Stylization Node: It receives a Bézier path at the input port, renders it, and stylizes it using parameters such as color, stroke type, or stroke style, which are provided at the other input ports, and returns the rendered and styled path. As the nodes are reactive, changing the color input of the line styling node will immediately change the line color in the resulting image after a cascade of processing.

Blending node: It receives two RGB frames as input, and returns the blended result. One of the inputs can be the output of the path stylization node and the other input can be the initial RGB frame.

To fetch input data and write output results of the processing graph to video file, read and write nodes are employed. The reader node reads multi-dimensional video from the disk frame by frame and provides the individual data as typed frame data at its output port. As nodes are reactive, the read results in a processing pass. Thus, the reader node can determine the frequency of processing passes and thus affects the video frame rate. To achieve the desired video frame rate, a scheduler is added to the read node. The write node ends a pipeline and writes the processed image to disk or allows displaying it in real-time.

Asynchronous processing. By offloading image data processing from the Central Processing Unit (CPU) to specialized hardware, such as Graphics Processing Units (GPUs) or Neural Processing Units (NPUs), and using the asynchronous programming paradigm, CPU resources can be freed up. While waiting for the GPU or NPU to finish processing and return results, the CPU can continue processing. The reactive node ports ensure a sequential flow of data, as calculation results are passed to subsequent nodes as soon as data is available. This asynchronous processing approach allows for flexibility in effect development and performance optimization. We could further improve performance by implementing a caching mechanism for output ports that allows results to be retrieved without re-processing the node if there have been no changes to its input port.

Precomputation Task. Processing-intensive steps, such as segmentation mask estimation in a video frame, can result in non-interactive processing performance if a user's hardware resources are insufficient. To solve this problem, precomputed human segmentations can be stored in the multi-dimensional video file before effect processing. The multi-dimensional video frame reader node can then read these frames along with the RGB frame and a segmentation estimation node is no longer required. We note that modern Apple mobile devices already run at interactive frame rates without such precomputations.

4.3 Modeling Motion-based Video Effects

A video effect is temporally divided into *Intro*, *Main*, and *Outro* stages (Fig. 6). Keyframes K_0 to K_3 are used

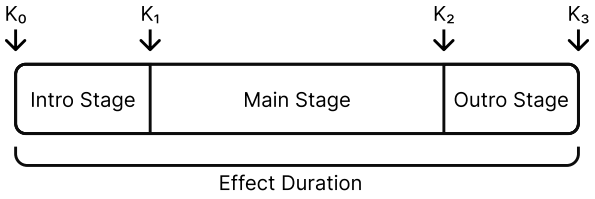


Figure 6: Keyframe-based parameterization of a motion-based video effect.

to define the start and end of an effect and the transitions between effect stages. The total duration of an effect is the time difference between K_0 and K_3. Animations of video effects can be created by interpolating visual variables between keyframes. The interpolation functions can be chosen from standard presets or defined using parameterized curves.

The additional video data generated from the Computer Vision (CV) framework, e.g., joint points, and segmentation mask can be used in several different ways to create a motion-based video effect, of which four will be explained in the following.

Connecting: We utilize joint point data and connect the points using Bézier curves, resulting in a skeleton-like effect. By adding an offset between the lines and post-processing the Bézier curves with a glow effect, an effect similar to attaching glow-sticks to the persons can be achieved.

Buffering: To create time-dependent effects, we buffer data over multiple frames, such as the data from the left and right wrists. We then draw a line through all of the joint points detected from previous frames. This results in a visual effect that resembles a light painting, with the line capturing the motion of the hand over time.

Constructing: We utilize detected human body poses as anchor points for further effect placement. For example, we use the detected root and neck points of a person to estimate the head placement, creating a convincing stick figure effect without the need for knowledge about the actual head rotation and placement. This is necessary because head rotation and placement might be partially obscured, making it difficult to place and size an ellipse around the head accurately.

Animating: We calculate metrics such as the distance between two points in the view space using joint points. The distance can then be used as an offset to change the position of Bézier curves already existing for the effect in the processed image. This results in an animation effect driven by the movement of the person in the video and thus motion visualization.

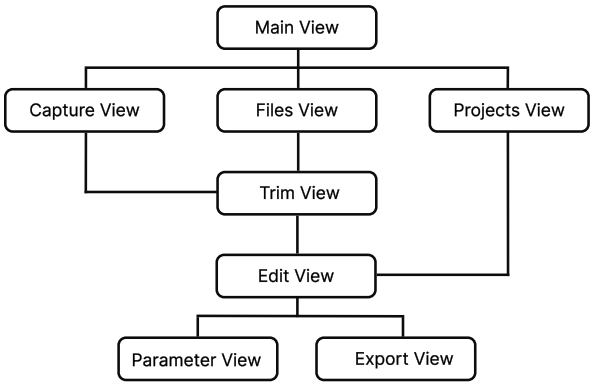


Figure 7: Diagram of the view hierarchy for individual views of our mobile application.

5 GRAPHICAL USER INTERFACE

To provide access to different functionality, the mobile app is structured into several views. Fig. 7 shows the view hierarchy of the mobile application, which we detail in the following.

Entry Point: The Main View, shown in Fig. 8, represents the entry point of the app, which provides buttons to navigate to the subordinate views. The Files View allows the user to load an existing video from the camera roll for editing. Saved projects can be opened again in the Projects View, accessed by the load projects button. Furthermore, the user can record a new video using the device's camera in the Capture View.

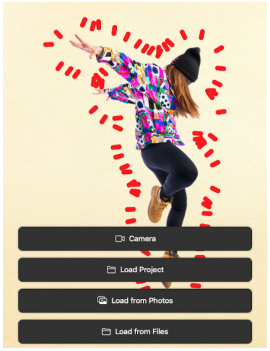
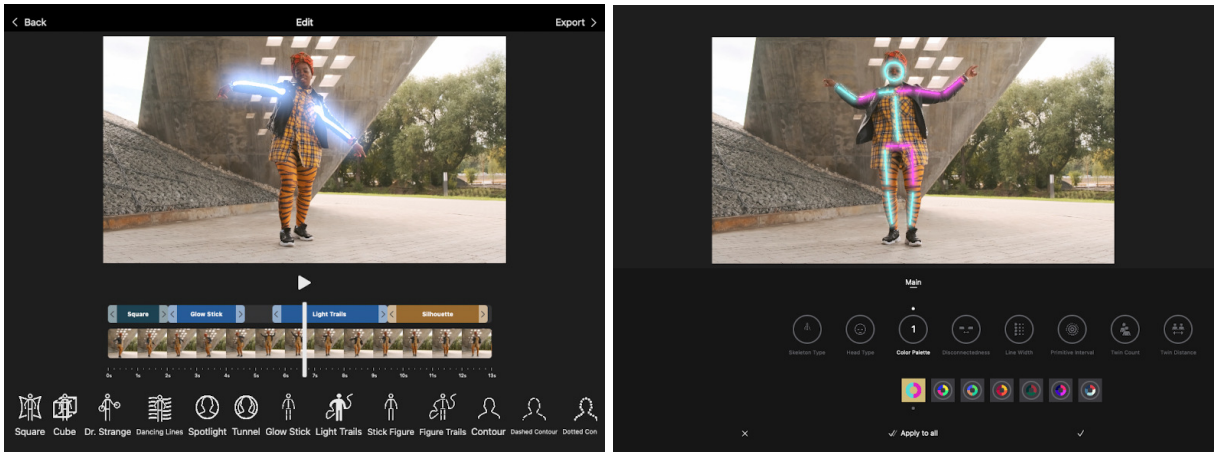


Figure 8: Main View

Trim View: Provides a video preview interface and allows to shorten a captured or loaded video.

Edit View: Provides a timeline-based video preview interface and allows to place video effects on the video (Fig. 9(a)). The top half of the screen displays the video preview and applied effects in real time allowing for interactive tweaking of effects. On the bottom of the screen the user can find various buttons to apply effects, represented by an icon and effect name. The user can find effects easily by horizontal scrolling of the effect button bar. Pressing and holding an effect button applies the effect to the video starting at the current video position for the duration of the button press while live previewing the video effect. The applied effect appears as a colored bar in the video effect timeline, found above the effect button row. To aid mobile usability there is a minimum effect length, that ensures effect bars are still easily visible and clickable in the timeline. Since only one effect can be active at a given time, an effect can only be applied if there is enough



(a) Edit View

(b) Parameter View

Figure 9: Screenshots of the user interface. The Edit View allows the user to apply, shorten, expand or delete effects. The Parameter View enables changing the effect parameters.

space in the video timeline. Applied effects can be prolonged or shortened by adjusting the end handles of the effect bars in the timeline. Clicking or tapping on an effect bar opens a modal allowing the user to delete the effect from the timeline or to switch to the parameter view. Pressing the export button in the top right screen results in a high quality rendering of the video with the applied effects for exporting and saving to the device.

Parameter View: Allows the user to customize effect appearance by adjusting effect parameters (Fig. 9(b)) and preview the results in real time. Every parameter is symbolized by an icon and a label with the parameter's name, where adjusted parameters are highlighted in a different color while the default setting is marked with a point below the value. Depending on the parameter type, different user interface elements for parameter adjustments, such as integer, float and color sliders, switches, icon buttons or effect icon buttons, are used for interaction. The parameter changes can be discarded, applied to the current effect, or applied to all effects of the same type using the three buttons on the bottom of the screen.

Export View: Allows the user to export the resulting video and save or share it with other users.

6 IMPLEMENTATION ASPECTS

Our mobile processing framework requires sensor and processing hardware to capture and process multidimensional videos, such as provided by the iPad Pro 3rd generation, which we used as our main validation platform. The mobile app is developed in the Swift programming language (version 5) and makes use of several Apple software libraries and frameworks. Our system is encapsulated in several components, see the supplementary material for a component diagram, they are briefly described as follows:

User Interface component. The GUI component implements the graphical representation and is implemented using the SwiftUI framework and a reactive programming paradigm in the form of the Apple Combine Framework. The AVKit library is utilized to enable and control media playback.

Pipeline component. The pipeline implements the multidimensional video processing architecture. It represents the core component of the framework, enabling the video frames to step through all executing steps and use their multidimensional video data to create different effects. The pipeline consists of multiple components to implement nodes and effects. The Apple Vision, CoreImage, and AVFoundation libraries are used in the processing nodes to implement computer vision techniques, Input/Output (IO)-related operations, and image manipulation. Furthermore, the Combine framework is employed to implement the data flow between processing nodes, which empowers the reactivity of interfaces between nodes and the implementation of asynchronous processing routines.

IO & Processing Components. The IO component offers data access to the device's media collection, local memory, and sensors, enables the recording and storing of multidimensional data and enables the user to permanently save processed content as a compressed project. The processing component includes custom-developed processing routines to create varied motion-based effects and uses Apple's MetalKit library for GPU-accelerated processing.

7 RESULTS & DISCUSSION

This section qualitatively evaluates our approach by means of different application examples (Sec. 7.1) and a performed User Study (Sec. 7.2) as well as quantitatively regarding runtime performance (Sec. 7.3). Furthermore, it discusses limitations (Sec. 7.4)

(a) Silhouette-based Effects (b) Body Pose-based Effects (c) Combined Effects

Figure 10: Stills from exemplary videos generated using the presented mobile application.

7.1 Application Examples

Fig. 10 shows stills from selected application examples generated using Lumo.

Silhouette-based Effects. These effects emphasize the contour of an individual's silhouette (Fig. 10(a)). They are produced by identifying the silhouette of an individual in the frame as a binary bit-mask. The effects may be portrayed as a continuous, dotted, dashed, or animated line. Additionally, attributes such as the color or glow of the line can be adjusted.

Body Pose-based Effects. To create body pose-based effects, we detect the joint positions of an individual as screen coordinates and connect them in a specific sequence with lines (Fig. 10(b)). Different sequences can produce various interpretations, such as a "Glow-stick effect", where the person's body joints are disconnected, or a "stick-figure effect", where all body lines are connected. Furthermore, altering properties such as line color and background brightness can modify the appearance of the effects. Additionally, different sequences for connection enable us to create differing body types. We are also able to create effects by buffering joint information over several frames to create effects like the light trails effect.

Combined Effects. We also created combined effects, that simulate a person being placed in a 3D scene, by fusing silhouette and joint point detection. First, a person's silhouette is detected as a bit-mask and used to crop the individual from the scene. Layering methods are then applied to create the illusion that the subject is situated within a three-dimensional setting ((Fig. 10(c), right). Adding the joint detection allows us to animate the scene by positioning scene objects depending on body coordinates and movements. This allows for the creation of various effects, such as "laser eyes" or "Dr. Strange"-like effects (Fig. 10(c), left).

7.2 Usability Evaluation

One of our non-functional requirements for Lumo is usability. We aim to ensure that the software is both error-free and user-friendly, even for non-professional users. To achieve this, we conducted a user study focused on determining the following: (1) whether the users found the GUI intuitive, (2) whether the effect appliance was perceived as fast and easy and (3) whether users were satisfied with the level of creative control they had over the effect design.

Participants & Apparatus. For our study, we recruited 18 volunteers (9 female and 9 male) between the ages of 17 and 55 years to test our Lumo application for the first time. The participants had no or little prior experience with video editing and stylization, and only four of them reported editing videos professionally or as a leisure activity. All participants were conceptually familiar with video filters and effects. We conducted a supervised in-person study where every participant used an iPad Pro (11", 3rd generation with M1 GPU and 8 GB Random Access Memory (RAM), running iPad OS 15.6.1) and was given a document providing a list of tasks. Each study session followed the same structure: (*i*) a quick overview of the study's methodology was provided; then, (*ii*) the participant received a document containing a brief app description, as it would be typically seen in an App/Play Store; following that, (*iii*) the participant received three tasks to complete sequentially. The study sessions lasted approx. 20 min.

User Tasks. We designed three tasks to cover the main functionalities of Lumo, arranged in increasing levels of difficulty.

Task 1 (T1): The user was given a 7 s video, which he was asked to load and edit. The given task involved applying a specific effect ("Glow Stick Effect") for a certain amount of time and extending or shortening the application time afterwards. The task concluded with exporting the video. The objective of this task was to help participants understand the concept of the effects in Lumo and the app workflow. On average, the participants took 143 s (excluding export time) for this task.

Task 2 (T2): The user was given a video with one applied effect and was tasked to change its parame-

ter values. We provided two GUI elements to modify these parameters: a slider and a switch, both of which users were asked to utilize to customize the given effect. The objective of this task was to help participants understand different kinds of effect customization. On average, participants completed this task in 112 s.

Task 3 (T3): For the last challenge we asked the participants to delete the applied effect and apply at least two different new effects to the video. Participants were also required to switch between the two effects when the dancing person did an eye-catching move and adjust the parameters of both effects according to their judgment. The objective of this task was to test whether participants could reuse their learnings from the previous tasks and understand the variability of effects. On average, participants completed this task in 151 s.

Data Collection and Analysis. After the study each user was asked to fill out a questionnaire, based on the Questionnaire for User Interface Satisfaction (QUIS) [Chi88] and the Computer System Usability Questionnaire (CSUQ) [Lew95] without time constraints.

Task 1 was successfully completed by all participants. They were satisfied with the visual results and expressed interest in exploring more of the app's functionalities. However, it was noticeable that nearly all of the users tried to apply the effects using a drag-and-drop metaphor instead of tap-and-hold, even after being shown a help message. In the qualitative part of the questionnaire, participants expressed a desire for a drag-and-drop functionality to be implemented.

Task 2 was completed the fastest and the option to adjust effects was found easily by the participants. They were fascinated by the control they had over the appearance of the effect. Nevertheless, some participants remarked that the terminology of the effect parameters is too technical for the non-professional target user group. Further, 33.3 % of the participants rated the application's flexibility with 5/5 points on the Likert scale, 27.8 % with 4 points, and the remaining gave 2 or 3 points. This might indicate that users wish for more options to customize effects, while also desiring greater clarity in the existing options.

Task 3 was the most enjoyable for participants, who relished the opportunity to try out different effects and parameters. This challenge took the longest on average, not because of its complexity, but due to the users' tendency to explore all the options for customizing the effects. Furthermore, participants expressed a desire for a feature that enables layering of effects, which is currently possible in the software with the use of blending but not available in the GUI.

The primary goal of the user study was to evaluate the usability of Lumo. In terms of usability, (*i*) all of the users were able to find the app's functionalities relatively fast, 72.2 % of the participants agreed that performing tasks in Lumo is straightforward, and 66.7 % rated the interface as pleasant. However, most of the users were dissatisfied with the number of help and error messages. Further, the size of the text and icons was too small for the majority of the users. Regarding learnability, (*ii*) 83.3 % of the users found it easy to learn how to use the application and 83.3 % were satisfied with the reliability of the app. Finally, (*iii*), 50 % rated the application's experience as stimulating. Most participants were able to find all the functions in the app that they expected to find (66.7 %). The overall satisfaction was rated 55.6 % with 5/5 points on the Likert scale and 44.4 % with 4/5 points.

7.3 Run-time Performance Evaluation

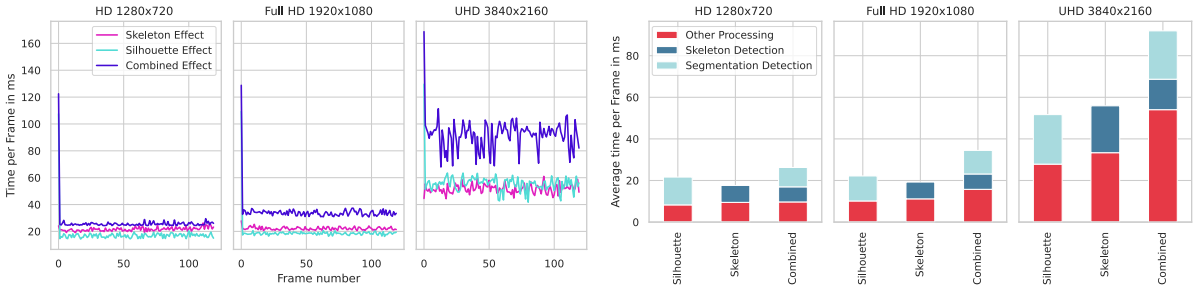
We tested the performance of Lumo with the following setup. Tests on mobile were performed in live preview mode, tested on a 3rd generation Apple iPad Pro 11-inch, equipped with an Apple M1 Chip and 8 GB RAM.

We conducted a run-time analysis using a test dataset comprising three 7-second-long, H.264/MPEG-4 AVC encoded RGB-videos with a frame rate of 24.86 Frames-per-Second (FPS) in three different resolutions: High Definition (HD) at 1280×720 pixels, Full High Definition (FHD) at 1920×1080 pixels and Ultra High Definition (UHD) at 3840×2160 pixels.

Tests were performed by applying a Skeleton effect, a Silhouette effect, and a Combined effect, which uses both body pose and segmentation estimation in its pipeline. For the final evaluation, we sampled 120 frames uniformly from the measurements. This step was necessary because the app's preview system skips frames to ensure interactivity when processing times exceed a certain threshold. Please view the supplementary material for details on the measurement setup and a per-node breakdown of timings.

The performance measurement in Fig. 11 demonstrates the real-time processing capabilities of the prototype app in HD and FHD, while maintaining interactivity at UHD resolution by dropping frames. We observe an initial spike in processing time in Fig. 11(a) attributed to loading the neural networks for body pose and segmentation detection. Processing time per frame is not strictly linear with increased input resolution, and could be influenced by proprietary Apple libraries used for up- or downsampling, tile-based rendering, and caching mechanisms.

In Fig. 11(b) we observe that processing time per frame increases with the effect complexity and detection of features required. Our approach is implemented in a pipeline-parallel manner. Thus, during the processing and rendering of one frame the segmentation and pose detection can already be performed for the next frame, which results in a noticeably more fluid experi-



(a) Processing time per frame

(b) Average processing time

Figure 11: Processing performance graphs for the applied effects by resolution. In (b), the processing time per frame is broken down by processing step. The processing nodes included in the "Other processing" category depend on the effect, and include nodes responsible for neon glow, day-to-night, line rendering, skeleton construction, silhouette detection, and blending and compositing.

Table 1: Peak RAM consumption of the app for different video resolutions (in pixels) and applied effects.

Video Resolution	Silhouette Effect	Skeleton Effect	Combined Effect
1280×720	113.96 MB	150.30 MB	133.23 MB
1920×1080	175.50 MB	125.80 MB	278.93 MB
3840×2160	444.96 MB	460.77 MB	2.43 GB

ence for the user, particularly when applying complex effects. We observe a shorter average segmentation detection time in the Combined Effect compared to the Silhouette effect, possibly due to internal resource optimizations in the Apple Vision Framework.

In Tab. 1 we observe, that the peak RAM consumption of the app increases with the input video resolution and effect complexity overall, but may display unexpected variations due to device-internal optimizations. Processing easily fits into the test system's RAM even at UHD resolutions.

7.4 Limitations

The presented approach has conceptual and technical limitations that can be addressed in future work.

The quality of the achieved effect is dependent on the detection accuracy of the computer vision models used for body pose estimation and segmentation provided by the Apple Vision framework. However, the detection performance for human body pose may be reduced under certain conditions. For instance, if some limbs are not visible in the frame, if the scene is was created from an unusual camera perspective (e.g. top-down), if subjects wear flowy garments (e.g. a wedding dress) or if the input video footage is very dark, body pose estimation accuracy is often degraded as Apple's developer documentation [Doc23] states. In some cases, shadows are mistakenly detected as humans resulting in undesirable artifacts when the effect is applied. In the case of body pose-based effects, the order of bone rendering is not depth-aware, which can result in skeleton joints or light trails being rendered in front of the body, even if they are occluded by other body parts.

8 CONCLUSIONS & FUTURE WORK

We presented a mobile framework for capturing and processing multi-dimensional videos to synthesize motion-based video effects. Our approach provides an exceptional level of artistic control and flexibility in creating motion-based video effects. We believe that our framework and mobile app will prove to be a valuable tool for professionals and amateurs alike, opening up new possibilities for artistic expression and, in particular, enabling non-professional users to design and share their own content-aware video effects.

While we currently extract segmentation and pose information from videos, the advanced sensor and processing capabilities of modern mobile devices offer exciting avenues for future work. Incorporating 3D data and (estimated) depth information is one such area that we plan to explore. By leveraging depth data obtained from sensors or depth estimation techniques, we can expand our range of visual effects by using the depth information to blend layers and simulate interactions with 3D elements. This addition has the potential to provide new opportunities for artistic expression and further variation in the possible visual effects.

REFERENCES

[Bou07] Simon Bouvier-Zappa, Victor Ostromoukhov, and Pierre Poulin. "Motion cues for illustration of skeletal motion capture data". In: *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. 2007, pp. 133–140.

[Cha20] Sammi Chan. "Latest trend in China: Glow Stick Dance Challenge". In: *South China Morning Post* (May 2020).

[Chi88] John P. Chin, Virginia A. Diehl, and Kent L. Norman. “Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’88. Washington, D.C., USA: Association for Computing Machinery, 1988, pp. 213–218. ISBN: 0201142376. DOI: 10.1145/57167.57203.

[Col03] J.P. Collomosse and P.M. Hall. “Cartoon-Style Rendering of Motion from Video”. In: *Vision, Video, and Graphics (VVG) 2003*. Ed. by Peter Hall and Philip Willis. The Eurographics Association, 2003. ISBN: 3-905673-54-1. DOI: 10.2312/vvg.20031016.

[Col05] John P Collomosse, David Rowntree, and Peter M Hall. “Rendering cartoon-style motion cues in post-production video”. In: *Graphical Models* 67.6 (2005), pp. 549–564.

[Cut02] James E Cutting. “Representing motion in a static image: constraints and parallels in art, science, and popular culture”. In: *Perception* 31.10 (2002), pp. 1165–1193.

[Doc23] Apple Developer Documentation. *Detecting Human Body Poses in Images*. Feb. 2023.

[Kwo12] Yunmi Kwon and Kyungha Min. “Motion Effects for Dynamic Rendering of Characters”. In: *Lecture Notes in Electrical Engineering* 181 (2012), pp. 331–338.

[Lew95] James Lewis and James R. “IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use”. In: *International Journal of Human-Computer Interaction* 7 (Feb. 1995), pp. 57–. DOI: 10.1080/10447319509526110.

[Lu13] Yijuan Lu and Hao Jiang. “Human movement summarization and depiction from videos”. In: *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2013, pp. 1–6.

[Lug19] Camillo Lugaresi et al. “Mediapipe: A framework for building perception pipelines”. In: *arXiv preprint arXiv:1906.08172* (2019).

[May21] Maximilian Mayer et al. “MotionViz: Artistic Visualization of Human Motion on Mobile Devices”. In: *ACM SIGGRAPH 2021 Appy Hour*. SIGGRAPH ’21. Virtual Event, USA: Association for Computing Machinery, 2021. ISBN: 9781450383585. DOI: 10.1145/3450415.3464398.

[Nie05] M. Nienhaus and J. Döllner. “Depicting dynamics using principles of visual art and narrations”. In: *IEEE Computer Graphics and Applications* 25.3 (2005), pp. 40–51. DOI: 10.1109/MCG.2005.53.

[Nie08] Marc Nienhaus, Holger Winnemöller, and Bruce Gooch. “Forward Lean–Deriving Motion Illustrations from Video”. In: *Proc. SIGGRAPH Asia Sketches* (2008).

[Sch10] Johannes Schmid et al. “Programmable Motion Effects”. In: *ACM Trans. Graph.* 29.4 (July 2010). ISSN: 0730-0301. DOI: 10.1145/1778765.1778794.

[Sem19] Amir Semmo et al. “ViVid: Depicting Dynamics in Stylized Live Photos”. In: *ACM SIGGRAPH 2019 Appy Hour*. SIGGRAPH ’19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450363068. DOI: 10.1145/3305365.3329726.

[Ume12] Daiki Umeda, Tomoaki Moriya, and Tokiichiro Takahashi. “Real-Time Manga-like Depiction Based on Interpretation of Bodily Movements by Using Kinect”. In: *SIGGRAPH Asia 2012 Technical Briefs*. SA ’12. Singapore, Singapore: Association for Computing Machinery, 2012. ISBN: 9781450319157. DOI: 10.1145/2407746.2407774.