Virtual Exposure Control for Creative Image and Video Editing

Ziliotto Salamon, N.

DOI
10.4233/uuid:f26ac568-7d3c-4dbe-a215-1e79bc00069f

Publication date
2021

Document Version
Final published version

Citation (APA)
Ziliotto Salamon, N. (2021). *Virtual Exposure Control for Creative Image and Video Editing*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:f26ac568-7d3c-4dbe-a215-1e79bc00069f

Virtual Exposure Control for Creative Image and Video Editing

Virtual Exposure Control for Creative Image and Video Editing

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
Chair of the Board for Doctorates
to be defended publicly on
Thursday, March 4th, 2021 at 3pm

by

Nestor ZILIOTTO SALAMON

Master of Science in Computer Science,
Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil.
Born in Ipê - RS, Brazil.

This dissertation has been approved by the promotor.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. E. Eisemann,	Technische Universiteit Delft
Independent members:	
Prof. dr. S. C. Pont,	Technische Universiteit Delft
Prof. dr. D. Gutierrez,	Universidad de Zaragoza
Prof. dr. T. Thormählen,	Philipps-Universität Marburg
Dr. C. R. Jung,	Universidade Federal do Rio Grande do Sul
Dr. R. Marroquim,	Technische Universiteit Delft
Prof. dr. ir. R.L.I. Lagendijk,	Technische Universiteit Delft, reserve member

This work was funded by the Brazilian agency CNPq (Process No 202551/2015-6) and carried out in the ASCI graduate school.

Keywords: multimedia; image processing; interactive editing; computer graphics

Front & Back: composition on a filmstrip with results of this thesis over a self-annotated diffused background.

ASCI dissertation series number: 414

ISBN: 978-94-6419-136-3

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Now shake the dust off and keep on rocking.

Contents

Summary

Post-processing has become a major component in movie and image production. This step is no longer a simple cleanup and cutting step, but it involves important manipulations that contribute to the atmosphere of a movie and the perception of a still image. Several movie studios spend a great part of their budget on it, as managing the post-processing parameters is a cumbersome task, requiring costly and specialized tools and skills. For photography, while several software packages provide automatic adjustments and filters, the fine grained editing is difficult to achieve for novice users.

The reason for post-processing is that many parameters are difficult to set correctly during the actual capture of the scene. An example is exposure time. Imagine you are in a car race and want to register that moment. To convey a sense of motion in your photograph, you adjust the camera exposure time: not too short to freeze all cars, nor too long to blur the image completely. To find the threshold, other camera parameters such as aperture and sensor sensitivity must be taken into account. Even the speed of the cars needs to be considered.

A much more suitable solution would be to adjust the motion blur after the acquisition. Nevertheless, this is not a simple task. Typically, it requires skill and involves manipulating the image by hand, which is time consuming and highly prone to artifacts. For videos, such edits are even more complex as the spatio-temporal coherence must be observed, especially when temporal warping occurs. In this dissertation, we present efficient solutions for exposure control in post-production to enable high-quality visual content generation.

Next to image-manipulation algorithms, we explore acquisition-based solutions and intuitive interaction metaphors to support expressive content production. Our outcome is not only intended for professionals to reach their design visions regarding atmosphere and storytelling, but also includes semi-automatic approaches to enable novice users to achieve impactful and realistic images. Consequently, the presented results have the potential of inspiring new artists, while the methods described can also be employed to simplify complex visual content creation tasks.

Samenvatting

N*abewerking* is een belangrijk onderdeel geworden van de productie van films en afbeeldingen. Waar deze stap vroeger bestond uit het wegwerken van oneffendheden of het bijsnijden en aan elkaar knippen van beelden, omvat het vandaag de dag uiteenlopende bewerking die bijdragen aan de sfeer en perceptie van het beeld. Filmstudio's besteden er een groot deel van hun budget aan: het beheren van het nabewerkingsproces is een omslachtige taak. Hiervoor zijn gespecialiseerde instrumenten en vaardigheden nodig, die dikwijls zeer kostbaar zijn. Hoewel er softwarepakketten zijn die automatische aanpassingen en filters aanbieden, blijft gedetailleerde bewerking moeilijk voor beginnende gebruikers.

Beelden worden vaak nabewerkt, omdat camera-instellingen moeilijk af te stellen zijn tijdens de opname van een beeld. Een voorbeeld is de belichtingstijd. Stel je voor dat je bij een autorace staat en je dat moment wilt vastleggen. Om een gevoel van beweging in je foto over te brengen, pas je de belichtingstijd van de camera aan: niet te kort, zodat de auto's niet bevriezen, maar ook niet te lang, zodat het beeld niet compleet vervaagt. Om de juiste waarde te vinden, wordt ook rekening gehouden met andere camera-instellingen zoals het diafragma en de sensorgevoeligheid. Zelfs de snelheid van de auto's moet worden overwogen.

Een betere oplossing zou zijn om bewegingsonscherpte na de opname aan te passen. Dit is echter geen eenvoudige taak en vereist een hoge mate van bedrevenheid. Daarnaast moet de afbeelding handmatig worden aangepast, wat tijdrovend is en makkelijk kunstmatige fouten oplevert. Voor video's zijn dergelijke bewerkingen nog complexer, omdat de samenhang van ruimte en tijd in acht moet worden genomen, vooral wanneer vervorming in de tijd optreedt. In dit proefschrift presenteren we efficiënte oplossingen voor het aanpassen van belichting tijdens de postproductie om het scheppen van hoogwaardige visueel materiaal mogelijk te maken.

Naast algoritmen voor beeldmanipulatie verkennen we manieren die gebaseerd zijn op het werven van beelden en intuitieve interactievormen om de productie van beeldend materiaal te ondersteunen. Ons resultaat is niet alleen bedoeld voor professionals om hun ontwerpvisie voor sfeer en verhaal te realiseren, maar omvat ook halfautomatische benaderingen die beginnende gebruikers in staat stellen om indrukwekkende en realistische afbeeldingen te maken. Hiermee hebben de gepresenteerde resultaten de potentie om nieuwe kunstenaars te inspireren, terwijl de beschreven methoden ook kunnen dienen om complexe taken voor het maken van visueel materiaal te vereenvoudigen.

Preface

This work is one of the outcomes of the not-initially-planned academic journey started in 2015. One week after my master defense, I decided to focus on my industry job. My former advisor, Prof. Dr. Soraia Musse, shared an email sent by Dr. Rafael Bidarra talking about the possibilities of academic collaboration between Brazil and The Netherlands via different scholarships. Reading about the exciting research done in the Computer Graphics and Visualization group at TU Delft, the industry focus became blurred. *Should I stay or should I go?*.

I contacted Dr. Bidarra, sharing my research interests on image processing, which put Prof. Dr. Elmar Eisemann in the loop. After a skype meeting with Prof. Eisemann, we started working on a project proposal for a scholarship from the Brazilian government. I recall the last minute iteration, with several changes that I had just a few hours to translate and submit. Apparently, it worked! I got the scholarship and, in 2016, moved to Delft to pursue the doctorate degree.

After four hard-working years, I returned to Brazil to finish writing this dissertation. During the valuable time there, I learned a lot working on different projects. The main projects, which now compose this dissertation, describe several experiments, tools and evaluations I implemented. While some projects branched from the original proposal due to the research findings, all still address the main research topic by exploring post-processing parameters towards image and video editing and content creation.

Furthermore, I also had the opportunity to assist on several courses and supervise bachelor and masters students. While these projects are not presented here, the ones resulting in publications are listed at the end of the dissertation. This certainly contributed positively to my experiences.

Without further ado, I would like to thank everyone who participated in this journey. I now let you with the pretty pictures I have been staring at for so long. Hope you enjoy the reading.

Nestor ZILIOTTO SALAMON
Porto Alegre, November 2020

1

Introduction

Visual content is the major element of the modern internet. Traffic predictions indicate that the future data transfer will be governed by images and videos. Cisco pointed out that annual individual network traffic (per IP) will exceed three zettabytes by 2021, with video viewing being responsible for 73-82% of this traffic[1].

Not only the availability of higher bandwidth, but also the tremendous amount of visual information that users can choose from and produce today contributed to the increased media consumption. Indeed, whole communities started to provide content, which is distributed over the web. These communities often consist of non-professionals, yet the produced content can sometimes rival professional productions, which is achieved by improved software and widely diffused by services such as Instagram and TikTok. Still, not all aspects of image processing are equally well captured and a large variety of challenges remain, in particular the adjustment of exposure time in a post-process. This existing gap becomes even more evident when looking at videos. For the most recent display trends, such as high framerate devices, even the professional movie industry relies on costly in-house developments and no suitable tools are accessible to common users. It might not come as a surprise that most of the budget of a movie is spent in post-production.

A major challenge is that many adjustments should have been addressed during acquisition. Hence, it becomes important to rethink the capture process and provide solutions to allow the artist to perform a large amount of post-production steps. For example, a standard 2D movie recorded at 24 frames per second, there is little room for temporal adjustments, while a higher frame rate might have been a better choice to give more freedom in post-production. Mid-range cellphones nowadays already record high framerate videos and can, therefore, be used to ac-

[1] https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1853168

quire extended data. On the technical and artistic knowledge to adjust and create visual content, the wide freedom in the choice of parameters makes it possible to also imagine novel interaction metaphors. In this realm, we are able to explore how novice user interact with editing tools, learning patterns and improving the pipeline for easier manipulations. Nonetheless, to better deal with the increased amount of visual information, there is a strong opportunity for developing optimization methods aiming real-time response rates.

Overall, we observe that: i) we have inflexible input (often the post-production has to work with standard videos as input in which the recording parameters are already fixed), ii) insufficient understanding and solutions to perform the intended manipulations (tools are non-existent or adhoc and the usage is cumbersome) and/or iii) the process is strongly linked to artistic skills (whereas the computer should actually provide support and suggestions).

This dissertation investigates visual-content processing and production methods based on virtual exposure control. Exposure, in photography, together with ISO and Aperture, influences the sensor response to light. A short exposure is able to freeze moving elements on the scene, whilst a longer exposure will capture the movement of such elements, creating blurred trails. In cinematography, the exposure behavior is analogous, and also related to frame rate. Together, both can be used to influence motion perception and the viewer's direction. During the shooting, however, the exposure setting is defined on the camera per take, i.e., different exposures need to be recorded with multiple equipment and/or multiple shots. Such constraints cause non-experienced photographers to often fail capturing a shot. We focus on new and accessible image-based exposure control methods that shift the creative control to post-processing.

Globally, we examine approaches to record additional information in order to give more artistic freedom to the users during post-production. Thus, the recording itself does not have to be perfect, but it should provide information that can then be exploited to adjust the final shot. Artistic choices, like shutters, lighting, etc., which usually require careful design already during the recording stage, can be later adjusted and mixed in real time to create a tailored result.

Further, we explore alternative pipelines to facilitate content authoring for non-professional users. For example, complex tasks such as Light Painting, which are developed on a trial-and-error basis even by professional artists, can be created in post-process by amateurs, delivering professional results. Combining different exposures in a single shot also requires complex masking and compositing tasks. Here we tackle such problems by developing tools to assist users during post-processing. For a widespread use, all images and videos that serve as input can be acquired on commodity hardware, such as cellphones and low-tier cameras.

1.1. Dissertation Overview

During my studies, we developed exposure control tools for synthesizing both still images and videos. Targeting still-image creation, we focus on Light Painting and Motion Blur control. For videos, we explore how to integrate different exposure times in the same video stream, as well as with a time-editing approach. An overview of the tools and personal contributions is presented in the following. The main chapters of this dissertation are based on articles written for each approach.

Computational Light Painting

Light painting is an artform, where a light source is moved during a long-exposure shot, creating trails resembling a stroke on a canvas. The first uses of light painting date back to the 1880s and gained significant attention in early 1900s, including works of Pablo Picasso. Even with the advanced technology of today, light paintings are still very difficult to create because the light source needs to be moved at the intended speed and along a precise trajectory. Moreover, it usually involves many attempts to get the desired result: the user cannot see the past stroke and there is no feedback during the drawing.

We developed a computational light painting solution, where the actual painting is performed in a post-process by directly drawing on the screen. Consequently, the precision is high and it is easy to control the output, which makes even animations possible. We make use of an extended acquisition, using as input to our approach a video of a person moving a light source along arbitrary trajectories through the scene. Our method processes this input to derive a structure which allows for querying an approximation of the relit environment for any given light position. Further processing also removes artifacts such as the person moving the light source and specularities on reflective surfaces. In consequence, it becomes possible to paint virtual light trajectories using our real-time interface and manage extra exposure controls, as well as precisely pick colors and create animated sequences. Fig. 1.1 shows light paintings created in real time using our solution.

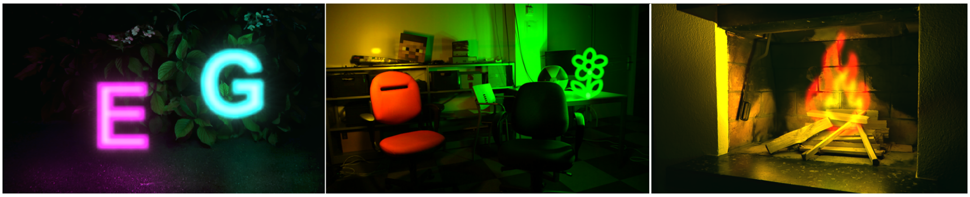


Figure 1.1: Light Paintings created with our approach. The data acquisition requires a light source to be moved through the scene - a process that takes only a minute. All drawings and color adjustments are performed in real time, avoiding the cumbersome trial and error usually employed for creating light paintings.

This project started from an idea by Prof. Elmar Eisemann and Dr. Marcel Lancelle and I implemented the methods and extensions during my first year. The final article, written as a collaborative effort, was published on Computer Graphics Forum (Vol 36, No 2) and I presented it at Eurographics 2017. Chapter 2 is *verbatim* of the publication, describing the whole project.

Controllable Motion-Blur

Motion blur in a photo is the result of object motion during the image acquisition. It creates a visible trail along the motion of a recorded object and can be used by photographers to convey a sense of speed and of how dynamic a scene feels. A photographer can create such effects by opening the camera shutter for an extended period of time, while potentially moving the camera to balance out object motion. This, however, is very challenging to acquire as intended and requires experience: faster objects require the camera to follow the focus point and use a short exposure time whilst slower objects need to be exposed for a longer period of time, while avoiding over- and under-exposure. To achieve actual control over the motion blur, one could add it in a post-process, but current solutions require complex manual intervention and can lead to artifacts that mix moving and static objects incorrectly.

We developed a novel method to add motion blur to a single image that generates the illusion of a photographed motion. Here, we aim at making the editing intuitive while relying on a standard image captured by any consumer camera. On a given image, from the user point of view, minimal input is required to create a blur effect: a rectangle defines the fore- and background segmentation and a motion vector defines the desired blur intensity and direction. Internally, the motion vector is converted to a blur kernel, convolved with the segmented area. Artifacts raised from the naive kernel-image convolution (e.g. color leakage and sharp edges) are automatically handled by our approach. The method was also extend to accept multi-directional blur, allowing an extra level of expressiveness with individual motion per-pixel. Additional effects, such as HDR blur and motion trails were also implemented. Fig. 1.2 illustrates the usage of our solution as an alternative to directly capturing the intended real-world motion blur: given the shot obtained with automatic settings, we enable fine-grained control of the motion-blur effect for each object or element in the scene.

This project started as a master thesis research from Xuejiao Luo, who I co-supervised together with Prof. Elmar Eisemann during her internship at TU Delft. After the thesis report was written, we wrapped up the project as a submission to Graphics Interface 2018. In this step I assisted the implementation and validation of the blur methods. The paper was accepted and I presented it at the conference. Our work was selected among the best papers, and we were invited to submit an extended journal version. For this extension, I contributed by developing the original multi-directional blur method and its effects, as well as creating manifold

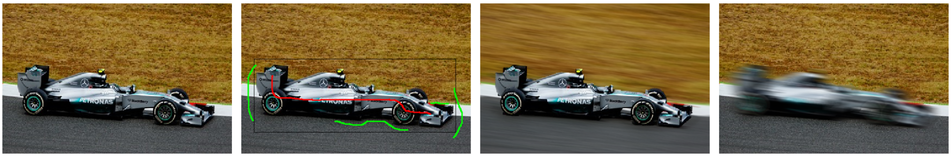


Figure 1.2: Given a single input image (left) captured with short exposure (usually also on automatic mode), users can segment content with scribble annotations (center-left). Next, a motion-blur effect is created to give the illusion of object motion during capture, also using different photo techniques (movable (center-right) and static (right) cameras). Original image source: pixabay.com

results, comparisons and tests. The extension was published on Transactions on Visualization and Computer Graphics (Vol 26, No 7). Chapter 3 is based on these publications and fully describes the method.

Spatio-temporal Exposure Control for Videos

The exposure time in a camera is defined by the shutter device. A faster shutter creates a short exposure, while the opposite yields a long exposure. In cinematography, shutter speed is intrinsically related to frame rate when dealing with motion control. While controlling the shutter has been established as a useful measure to influence the perception of a video sequence and to enable different visual styles, a physical camera is limited to a single shutter setting at any given moment. To combine different shutters on the same scene (mix different motion patterns together), the available options are rather limited and require multiple cameras and laborious compositing techniques.

We developed a novel technique to virtually simulate and combine different exposures over space and time with shutter functions. Here, we again make use of the extended acquisition process: a footage with higher frame rate can lead to more freedom on controlling the exposure. We built upon the observation that combining two neighboring frames is basically equivalent to a doubling in exposure time. Thus, we let artists define spatio-temporally-varying virtual shutters in a post-process using a simple user interface and ensure seamless interpolation of these. All results are presented in real time, which supports the shutter design process. Unlike previous work that require masking and compositing steps, a key advantage of our solution is that only sparse spatial and temporal annotations are needed, which then define varying per-pixel shutter functions for the entire video. The result is a mixed exposure video, as illustrated in Fig. 1.3.

This project began as a branch from our main goal of providing directors with intuitive techniques for spatio-temporal exposure control on videos, which I started researching in my second year. The results were published on Computer Graphics Forum (Vol 38, No 7) and I presented it at Pacific Graphics 2019. For this project, I developed the framework and evaluation and created the results, being helped by

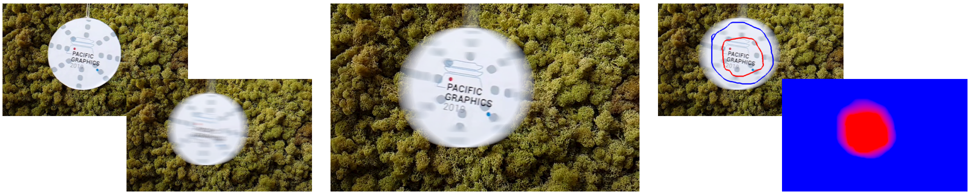


Figure 1.3: Left: single frame from a 240Hz short-exposure video and a simulated long exposure at 30Hz by averaging 8 frames. Middle: Using the 240Hz input, our method enables mixing a long exposure in the periphery with a short exposure for the details on the pendulum. Via user annotations in the video, different shutter functions can be defined (top right). Annotations and shutter functions can be keyframed over time. Based on the annotations, our method defines an interpolated shutter function for each pixel (bottom right).

Dr. Markus Billeter on the GPU acceleration methods. Writing was a collaborative effort and the entire approach is presented on Chapter 4.

Video Editing with Spatio-temporal Object Control

Video editing is a very time-consuming task: raw material is edited and combined until it matches the director's vision. As re-shoots are expensive, temporal editing operations such as looping, repeating or speeding up/slowing down existing sequences are common ways to better match the narrative in post-production. However, such operations can become overly complicated if applied to selected elements rather than the whole frame, as spatio-temporal conflicts such as overlaps and disocclusions start arising.

We developed an end-to-end solution for assisted spatio-temporal video editing and conflict resolution. Given an input video, users can segment objects of interest, which are automatically identified throughout the sequence. To allow a user to influence the timing of a video locally, we rely on an intuitive user-interface with real-time feedback, which enables even novice users to author new scenes using a few annotations. Inconsistencies introduced during the editing process, both spatial and temporal, are visualized and resolved using our tailored methods. In Fig. 1.4 we illustrate the selective time editing by slowing down the basketball and reconstructing the disoccluded area below its original position.

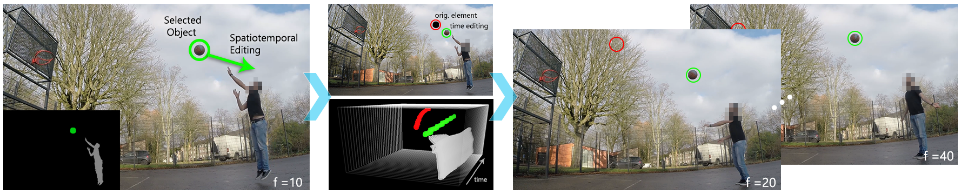


Figure 1.4: Left: A user segments the objects in the input video (bottom-left inset) and adjusts their timing. Middle: Real-time previews and visualizations support the editing process. Right: real-time synthesis of the final video with an adjusted basketball throw.

In this project we re-introduced the concept of objects and temporal manipulation from our original spatio-temporal exposure control goal, coupled with an interactive end-to-end pipeline for video editing. I acted as the main developer and the final text was written as a collaborative effort. The complete project is presented on Chapter 5.

The projects described here summarize the main results during my doctoral studies. All together, we developed solutions to virtually control exposure in images and videos, from simplified acquisition pipelines to real-time post-processing methods.

2

Computational Light Painting Using a Virtual Exposure

N. Z. Salamon, M. Lancelle and E. Eisemann

Though I'm going, going,
I'll be coming home soon,
Long as I can see the light.

Light painting is an artform where a light source is moved during a long-exposure shot, creating trails resembling a stroke on a canvas. It is very difficult to perform because the light source needs to be moved at the intended speed and along a precise trajectory. Additionally, images can be corrupted by the person moving the light. We propose computational light painting, which avoids such artifacts and is easy to use. Taking a video of the moving light as input, a virtual exposure allows us to draw the intended light positions in a post-process. We support animation, as well as 3D light sculpting, with high-quality results.

This chapter has been published on Computer Graphics Forum, **36**, 2 (2017) [1].

2.1. Introduction

In recent years, light painting has gained significant popularity in photography. There are two variants, both sharing a slow shutter speed to capture light trajectories on the image plane. One option is to move the camera, while keeping light sources fixed - known as Kinetic Light Painting, e.g., Lorenzi and Francaviglia [2] installed a large set of light sources during the Generative Art 2007 conference. In this paper, we focus on the second option, where the camera is static and the lights are dynamic. While some large-scale attempts have been made with hundreds of people [3], most creations are performed by a single person, who needs to skillfully move the light source with the goal of producing a target light trail on the camera sensor. Such a process is tedious, requires high precision, and usually involves many attempts because the user cannot see the past stroke and there is no feedback while drawing. Moreover, artists usually wear black cloth and move fast to not corrupt the result by appearing in the background. We propose computational light painting, where the light painting is made in a post-process by drawing on the screen. Consequently, the precision is high and the output is easy to control, which makes precise animation possible.

The input to our approach is a video of a person moving a light source along arbitrary trajectories through the scene, which is usually captured within a minute. Our method processes the input to remove artifacts caused by the artist and removes the light source. The latter allows us to replace it by a smaller synthetic light, which can then be drawn at any position. Removing the person from the background avoids ghosting effects, which are typical for real light paintings. We then produce a data structure to query an approximation of the relit environment for any light position. It is, hereby, possible to paint light trajectories and accumulate the illumination contributions. We show various results, including simulated professional light brushes [4].

Specifically, our work makes the following contributions:

- a simple video acquisition methodology;
- a preprocess to remove acquisition artifacts;
- a robust light interpolation method; and
- a design interface for light painting.

2.2. Background and Related Work

Light painting dates back to 1880: the *Pathological Walk From in Front* by Étienne-Jules Marey and Georges Demeny. The authors, both psychologists, used the technique to understand human motion. Only in the early 1900s, the first artistic light drawings were produced by Man Ray and later Picasso. The theoretical background was presented in 1940 by Leslie Walker [5]. Afterwards, the artform gained momentum, as summarized by Lance Keimig [6]. Today, we see a renewed interest in the topic. The availability of digital cameras makes it cheap to rely on trial-and-error attempts, which motivated many hobby artists. Nonetheless, the process remains time consuming and requires a high level of skill for convincing results.

The most recent development has been presented by the Austrian company FilmSpektakel. They propose a light painting pipeline named Holopainting [7]. They first acquire a subject with a ring of 24 connected cameras, which serves as a 3D scanner to recover a moving subject. The imagery is then transferred to an LED stick, which illuminates the 3D scene automatically during the long-exposure shot. From recording to final result, the process takes days, including manually cropping each captured image to fit the subject in the LED stick.

Faster, but less accurate applications (e.g., [8, 9]) simplify the process by accumulating many short exposed images to a single long-exposure shot. The principle is similar to Telleen et al. [10], who introduced the virtual exposure, which also forms the basis of our approach. Given this collection of frames, the user can delete or replace time periods. Nonetheless, artifacts due to the user's presence in the scene remain and the painting still needs to be performed actively. The latter also holds for recent approaches using virtual reality [11].

Related to our approach are also relighting scenarios. In virtual environments, it has been a long established concept [12], but similar principles of basis images for lighting can also be found for real-world applications. An image-based approach to light design was presented in [13]. Manders and Mann [14] proposed a solution to add light on the fly by employing an adapted flash equipment that triggers the camera shutter to accumulate the lit results. Similarly, Boyadzhiev et al. [15] capture a set of flash photographs with distinct light positions, then used in a composition method guided by the user. Product relighting is addressed analogously in [16]. The authors record a video, illuminating a product from different angles and find image snippets, which match design principles. Later, the user can choose and compose them.

Low-frequency environmental lighting can be achieved via Bayesian relighting [17], which involves a light probe to estimate the environmental illumination. Knowing the illumination can be useful for scene reconstruction (e.g., [18]), and material estimates (e.g., [19]), as well as composition [20]. An advanced light installation, supporting precise capture and relighting, is the light stage [21]. An overview of image-based lighting can be found in [22].

Our work is inspired by the effectiveness of low-cost image-based relighting solutions. It requires no calibration and provides access to a difficult-to-master artform. This latter aspect, we share in spirit with work around spray-can images[23]. Nonetheless, a major difference is that our solution is compatible with the entire image-processing toolbox, such as Photoshop, or even advanced supporting solutions, such as ShadowDraw [24].

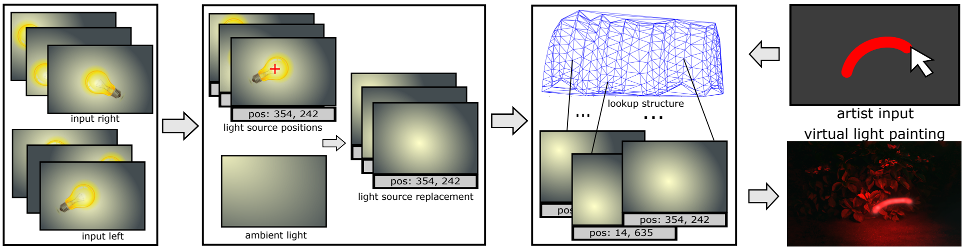


Figure 2.1: Overview: The input frames are from a static scene with a moving light source. For each frame, the light position is obtained and its illumination of the scene extracted. These contributions are stored in a lookup structure, which can be queried to design light paintings in real time.

In the following, Sec. 2.3 describes our approach, including acquisition and processing (Sec. 2.3.1), light interpolation (Sec. 2.3.2), and the user interface (Sec. 2.3.3). We will then present our results (Sec. 2.4) before concluding (Sec. 2.5).

2.3. Our Approach

An overview of our approach is shown in Fig. 2.1. The input is a video recorded with a moving light source. First, we process the video to remove artifacts related to the capturing process; we want to obtain *lighting images*, which encode the illumination of the scene. Neither the light source nor the person carrying the light source should be included (Sec. 2.3.1). Next, these frames are organized to provide a basis for new light positions, which are approximated by interpolation (Sec. 2.3.2). Finally, we will explain our interface and how the actual light painting is performed (Sec. 2.3.3).

2.3.1. Acquisition and Processing

The acquisition process is simple and allows for a robust derivation of the scene illumination. We use a white light source, which allows us to tint the illumination in different colors in a post-process. The user is asked to move through the scene from left to right and then back, waving the light source in front. In our setup, we relied on a standard light bulb attached to a stick or a flashlight in a thin plastic cup acting as a diffuser. To avoid automatic camera adjustments, the recording camera is in manual mode.

Our goal is to transform each frame of the input sequence into an image that only encodes the light added by the moving source and free of artifacts from the person moving it. The first step linearizes the response curve of the camera [25]. We convert our images by fitting a gamma correction followed by a normalization step. The so-linearized images are then processed further.

Ambient Lighting

In order to obtain only the added light, we subtract the ambient lighting A from each frame. To determine A, we rely on the first few frames of the video - before the moving light source appears in the scene. Accumulating these frames gives a good estimate of A for an exposure time equivalent to the summed frames [10]. In consequence, we need to divide by their number to obtain the background illumination per frame. Subtracting A from all frames then gives us a new video containing only the added light.

Removing the Light Source

We are very forgiving when estimating light-source sizes, shape, and positions. Hence, even if the captured source is slightly larger or some positions have not been properly sampled, we can produce a convincing result by interpolating between the images. Nonetheless, leaving the light source (or the person manipulating it) visible in the image produces unrealistic ghosting. We first focus on light-source removal before addressing the person ghosting in Sec. 2.3.2.

Light-source tracking

In all of our experiments the directly visible light source fully saturates the sensor. This greatly simplifies the detection. We employ a high threshold (1.0 on all channels) and then grow these regions up to a threshold of 0.9 before applying a morphological closure to remove noise. Still, this process is insufficient in the presence of reflections. To robustly estimate the light position, we leverage the temporal coherency of the video with a consistently moving light source. In consequence, we perform a tracking over time to derive a mask that we can use for the light-source removal.

We choose a standard tracking solution for the light position. The tracking is initialized on each detected region in the thresholded image. Next, we match each region to the following frame and verify i) a low size variation $\Delta s < 20\%$ and ii) a small-distance movement $\Delta d < 5\% s$, where s is the light's estimated size; the numbers have been determined experimentally. As the user is asked to move from left to right during acquisition, we can trigger the tracking when the light enters the scene from the left. Further, the movement being parallel to the image plane, the light will keep a constant size. To match two succeeding frames, we rely on a sum of squared differences and determine the best translation vector. Tracking results are illustrated in Fig. 2.2.

Light-source replacement

Having a mask that covers the light source, we now inpaint the corresponding region to remove the light source from each frame. While a standard inpainting [26] could

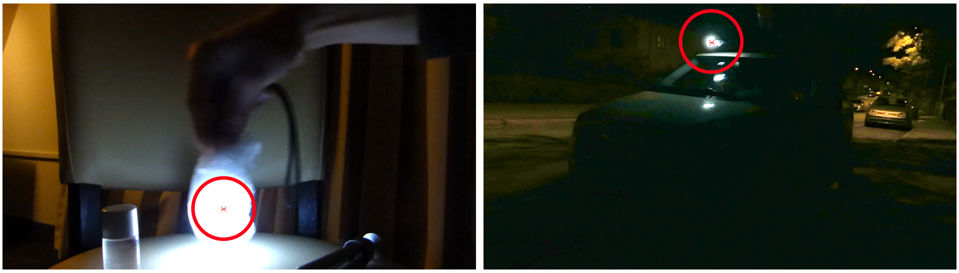


Figure 2.2: Our tracking algorithm automatically finds the center of the light sources with different sizes, occlusions and reflections.

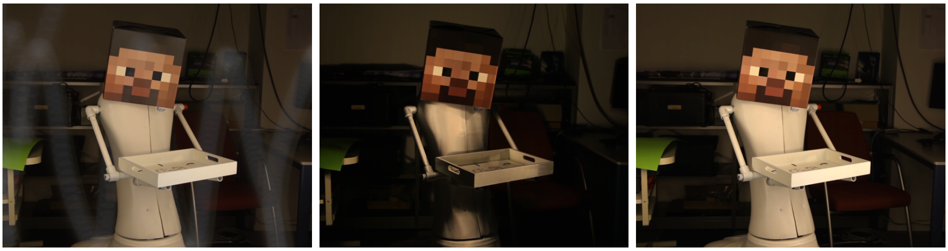


Figure 2.3: Methods to compute a neutral image for light source replacement. From left to right: Mean, median and trimmed mean.

be used, it is only a coarse estimate because the light source bleached all structural information. Instead, we inpaint the region based on the input-video frames.

To recover the image structure behind the light source, one could compute a mean over all images, but this process causes strong ghosting artifacts due to the high-contrast light source. A second option is a median, but it is generally too dark due to the short illumination of each part. We found that a trimmed mean (10% of the darkest and brightest samples removed) provides a robust trade-off between the artifacts in the mean and the darkness of median (Fig. 2.3).

Copying the patch directly from the trimmed-mean image T would result in visible seams. A solution to this problem is a Poisson reconstruction with the pixels around the masked region serving as constraints and T as the guide image [27]. Hereby, elements that are lit, extend their illumination naturally into the uncovered region. In practice, we use Tanaka et al.'s variant [28], which leads to less color bleeding. Still, the solution exhibits smaller artifacts because the gradients in T might not have a consistent magnitude. To adapt T, we need to scale it with a correcting factor. As the patch lies directly underneath the light position, it would appear fully illuminated, if the view was not blocked by the light source. For this reason, we are interested in the maximal possible magnitude of gradients for this area. We estimate it by using the average magnitude of the top 1% gradients in this region over all images, excluding the ones covered by the source. We then scale T

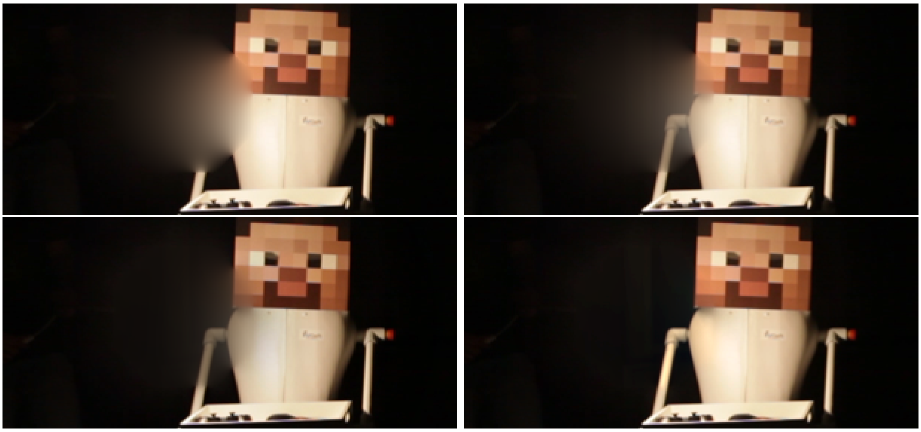


Figure 2.4: Light source replacement. Top: Poisson without and with guide [27]. Bottom: Tanaka et al. [28] and our result.

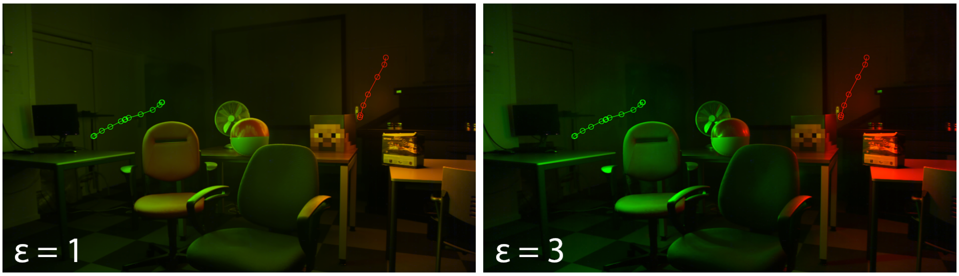


Figure 2.5: Shepard interpolation of captured sources varying ε.

to have its gradients match the result. A comparison of the inpainting processes is shown in Fig. 2.4. We refer to the resulting images as the *lighting images*.

2.3.2. Light Interpolation

The pre-processing results in a set of lighting images $\{L_i\}$ corresponding to the added light of the moving source at position p_i. Here, we describe how to approximate the lighting image L_n of a new virtual point light at position p_n with an emission E. E is assumed to be given relative to the light source used in the capturing process. If the virtual light is at a captured position, there is a p_k equal to p_n, and we obtain $L_n := E * L_k$. If not, we will define $L_n := E * \sum_i w_i L_i$, for a suitable set of weights $\{w_i\}$.

One weight definition is the inverse distance metric (Shepard interpolation); $w_i = \hat{w}_i / \sum_i \hat{w}_i$, where $\hat{w}_i = 1/(|p_n - p_i| + \varepsilon)$. The parameter ε has a strong impact on the outcome and can lead to a diffuse solution (Fig. 2.5).

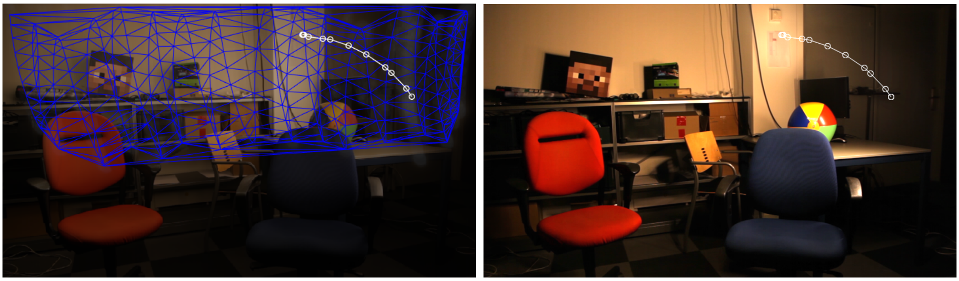


Figure 2.6: A Delaunay triangulation of light centers defines the weights to interpolate the captured light images.

A more suitable and local interpolation uses a Delaunay Triangulation of $\{p_i\}$, where the weights are defined via barycentric coordinates (Fig. 2.6). Hereby, only the surrounding three vertices are used for interpolation. Consequently, locality is increased, while being continuous over the entire domain.

Person-aware weights

While the interpolation allows us to define new light positions, we still need to handle the artifacts caused by the person moving the light source. To this extent, we introduce *person-aware weights*.

As the capturing process is executed twice (from left to right and vice versa), it is easy to determine the side (left/right) of the image contaminated by the person. Please notice that also all shadows cast by this person lie on this side. Consequently, we can remove the part of the image that contains the contaminations, resulting in the sets $\{\hat{L}_i^l\}$ and $\{\hat{L}_i^r\}$. L_i^l is equal to the original input frame, except that all pixels left of p_i^l are zero. Unfortunately, we cannot assume that the light source will visit the exact same locations during both passes, i.e., for a position p_k^r there might not be a corresponding position p_i^l. Thus, we cannot directly combine the image parts to produce one artifact-free set.

To fuse the images, we rely on the weights. First, we perform the Delaunay triangulation for the right $\{p_i^r\}$ and left pass $\{p_i^r\}$ independently, and derive corresponding weights $\{w_i^r\}$ and $\{w_i^l\}$. Next, we compute two result images using only left ($L_n^l := \sum w_i^l L_i^l$) and right image parts ($L_n^r := \sum w_i^r L_i^r$). Additionally, we produce corresponding weight images $W^r = \sum w_i^r \chi_i^r$ and $W^l = \sum w_i^l \chi_i^l$, where χ_i^x is the characteristic function that is one if the pixel lies in the valid part (left/right) of the image. For example, $w_i^l \chi_i^l$ is an image where all pixels left to the light position p_i^l contain w_i^l, while the rest is zero. We define $L_n := (L_n^r + L_n^l)/(W^l + W^r)$, which is a blended image without gaps. To avoid seams, we define χ_i^l (χ_i^r) not as a step but as a smooth ramp function to blend the image contributions. Fig. 2.7 shows the result before and after person-aware weights.

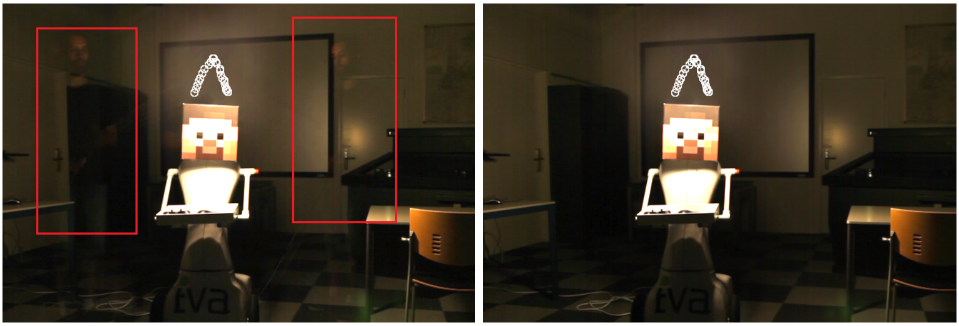


Figure 2.7: Ghosts by the light-handling person (left) are avoided by our approach (right).

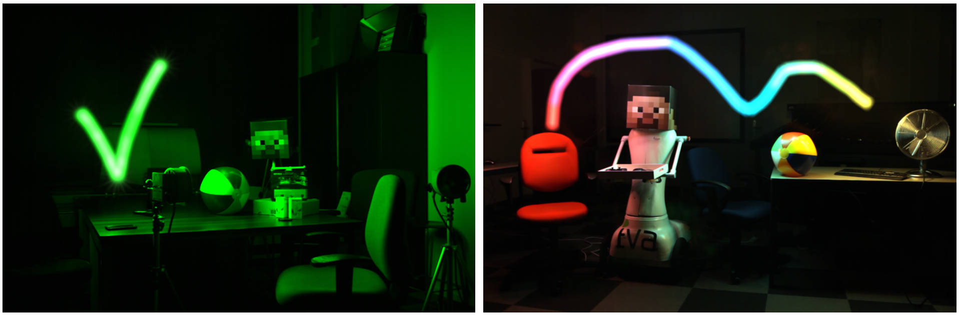


Figure 2.8: Intensity (left) and color (right) variation along a stroke.

2.3.3. Light Painting Interface

We have seen how to query a lighting image L_n for a virtual point light. Based on this result, we can reconstruct an approximation of the corresponding illuminated scene by $L_n + A$, where A is the ambient light image. To extend the principle to general virtual sources, we exploit the linearity of light transport; we approximate the stroke via a set of point lights and sum the corresponding light images. Consequently, we can use arbitrary pixel input.

Our interface enables the user to define strokes of varying thickness, color, and intensity. Intensity is defined with respect of the exposure time of a frame. If one wants to simulate a different exposure time, all intensities and the ambient image should be scaled before addition. Strokes can either be drawn as sampled freeform curves (represented by dense polylines) or splines. We support a special drawing mode, which keeps track of the speed at which the mouse moves - slow meaning brighter, which mimics the effect during a real exposure. The properties and control points of the curve can still be modified after its creation and values are interpolated between the control points (exemplified by a color gradient in Fig. 2.8, right). Additionally, vector or bitmap images are supported (Fig. 2.9).

Our interface also allows the user to define key-framed animation and path

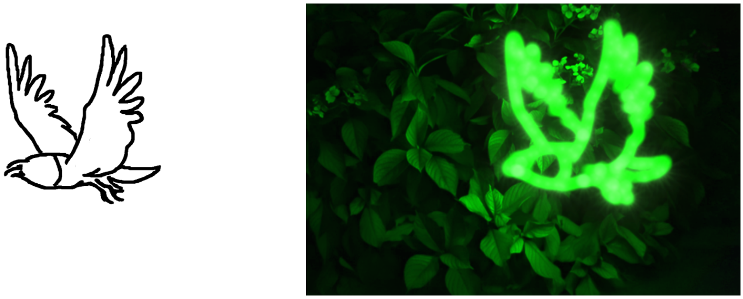


Figure 2.9: Vector input (based on Whitaker and Halas [29]) and the resulting light painting.

definitions. The light is then interpolated according to the user-provided animation. There are two options for the output: assuming a short-exposure in each frame, or an accumulation (resulting in an increasingly long exposure) along the trajectory.

Extensions

There are several extensions to our basic approach, explained so far, which are also exposed in the interface of our prototype implementation. They can improve the quality of the results drastically.

Lens Flare and Starburst Pattern. To increase realism and an increased sense of light [30], we can add common camera artifacts [31–33]. To this extent, we convolve a starburst pattern [31] induced by the aperture with the stroke input.

Depth Capture. Our approach can be extended to 3D light sculpting. The capturing process acquires one depth layer at a time. These layers are combined in form of a 3D Delaunay triangulation to enable 3D interpolation. To determine the 3D position of each light source, we rely on its relative size. This depth/size parameter is then exposed in the interface. A user can choose a value for each key point of a stroke and also animate these values.

Specularities. For very specular objects, we require a detailed acquisition to avoid undersampling artifacts. Or existing algorithms for artifact-free specularity interpolation [34] could be applied. As an alternative, we propose to remove specularities by thresholding the images again, while excluding the light source, followed by our inpainting algorithm.

GPU Acceleration. We mapped the light painting accumulation efficiently on the GPU. We first precompute two images, a weight image, where each pixel contains the barycentric weights of the triangle that contains it, and an index image, which contains the indices of the triangle's vertices. For each input pixel on the stroke, we can thus retrieve the weights and index of the light images that need to be accumulated. The composition can then be achieved via alpha blending.

2.4. Results

Aside from the GPU painting interface, we implemented our method in Matlab. The GPU painting interface runs at 60 fps for 960x540 images on a GTX 980 but the pre-processing takes around 5/20 seconds per frame (960x540/full HD). The latter could be improved but our focus was on showing the high potential of computational light painting. We provide many examples, ranging from indoor scenes, to outdoor examples, on a large and small scale. The scenes were captured by different artists with different equipment. The videos were recorded at frame rates from 24 to 30 fps and with manual focus; ISO and aperture parameters were set according to environmental characteristics (strong/weak environment light, distance camera-light source, etc.). All videos were processed as obtained and recorded in a single attempt.

While light painting animations can take months [35], our approach can provide results in the order of seconds, using key frame animation coupled with a path trajectory (Fig. 2.10).

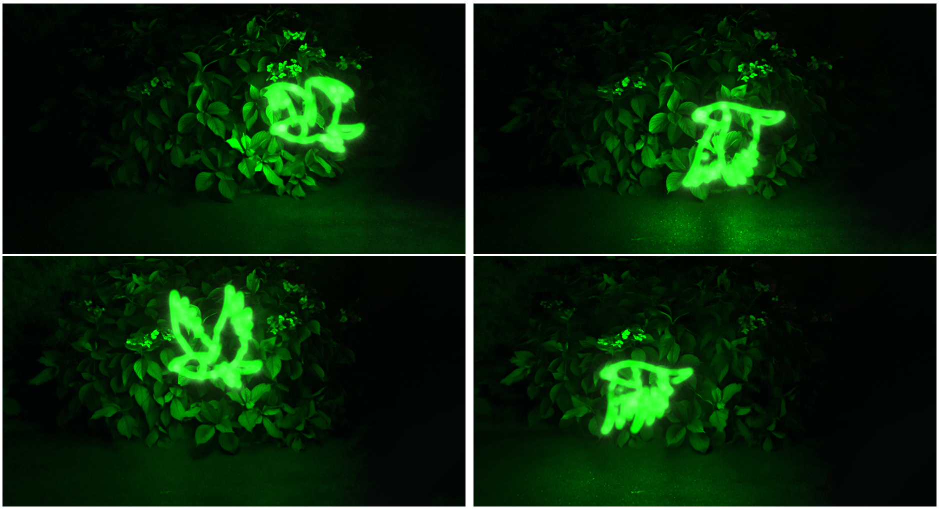


Figure 2.10: Frames of an animation.

Predefined drawings or single strokes (Fig. 2.11), potentially augmented with flares for more realism (Fig. 2.12), are basically done instantly (no example took more than a minute of design).

Complex drawings with layers, several shapes, or strokes took us up to five minutes to design. A layered result is shown in Fig. 2.13. The same has been applied for the Eurographics logo in the theatre (Fig. 2.18, top left).

For cases where the scene was not consistently sampled or recorded from only

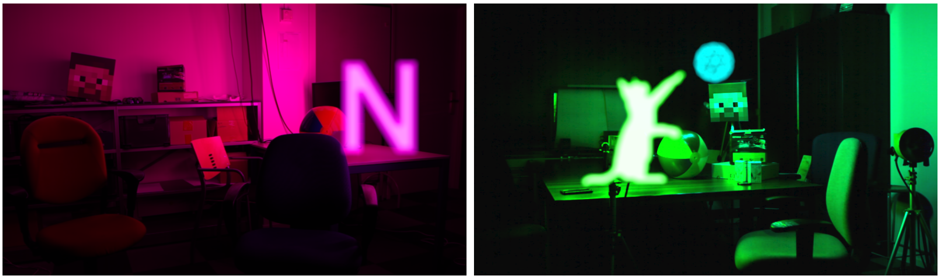


Figure 2.11: Using drawings and fonts as input points.

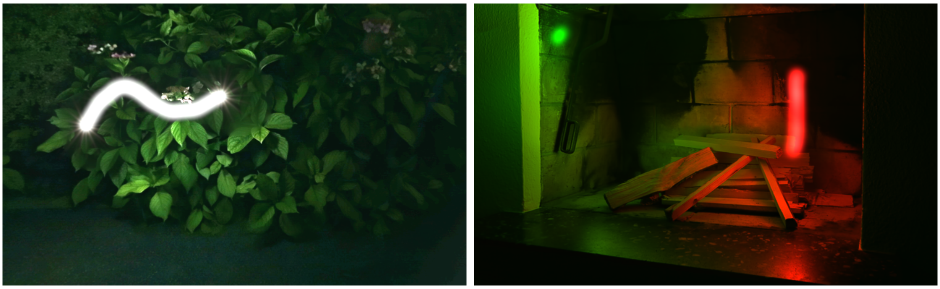


Figure 2.12: Different strokes with and without lens flare.

one side, our approach still produces convincing results, but cannot remove all artifacts. Fig. 2.14 (left) shows our result for a person standing behind the light source and, Fig. 2.14 (right), uses only few light samples. Moreover, if the light source is weak, we cannot scale the brightness without introducing noise. These artifacts occur also in traditional light paintings. Similarly, the behavior for non-static scenes is identical and results in motion blur. As an example, Fig. 2.18 (top, right) was recorded with a strong wind moving the leaves and Fig. 2.18 (bottom, right) shows a moving cow in the background.

Fig. 2.15 illustrates a challenging example and shows the effectiveness of our

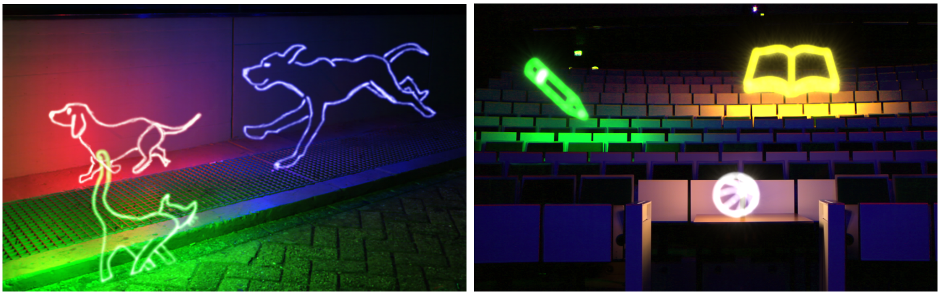


Figure 2.13: 3D drawing in different layers.

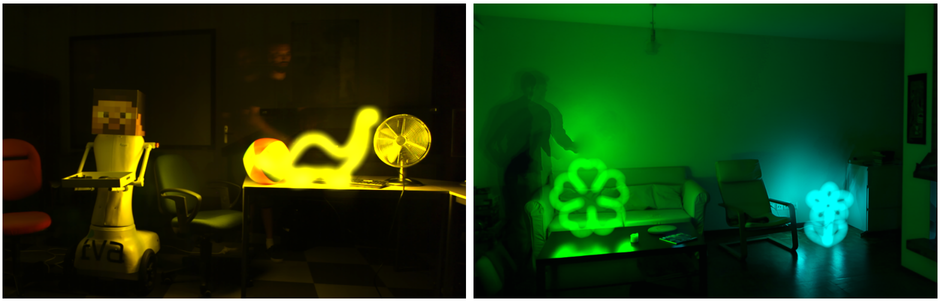


Figure 2.14: Results with inputs recorded from one side, with the person stands behind the light source or in a sparsely sampled scene.

solution in handling specular surfaces. The coarse sampling could lead to artifacts, but our work removes these problems via inpainting of the specularities. Large mirrors can still lead to problems and ghosting effects cannot be completely avoided. In the future, using polarized filters could help detect such cases.

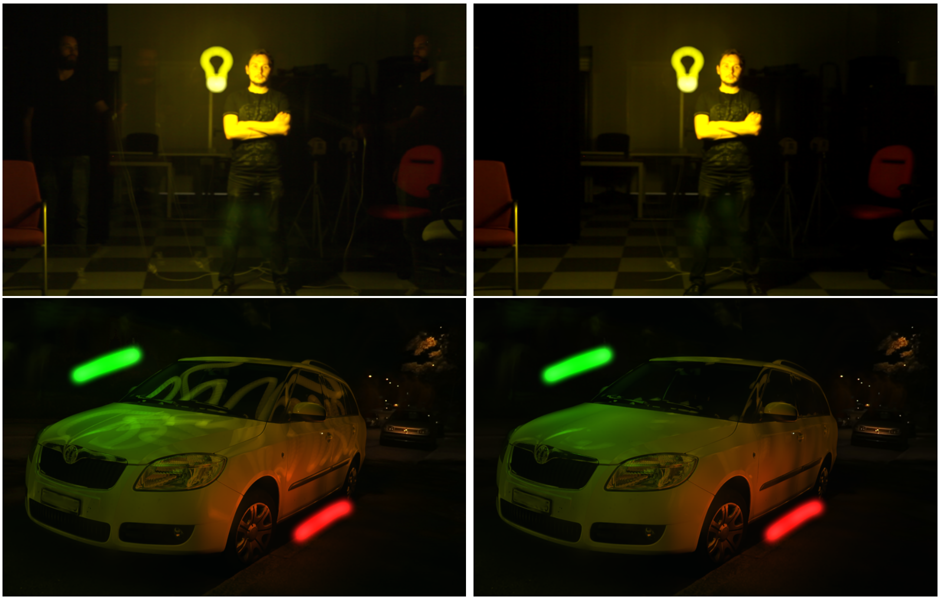


Figure 2.15: Ghost removal (top) and specularity treatment (bottom).

Besides light painting, our method can also be used for product light design or relighting. In Fig. 2.16, an artificial lamp is placed in an apartment scene and the illumination steered with our solution. The wide variety of scenes that can be handled is also illustrated in the overview (Fig. 2.18).

To examine precision, we tested a synthetic scene. Fig. 2.17 (top, left) shows

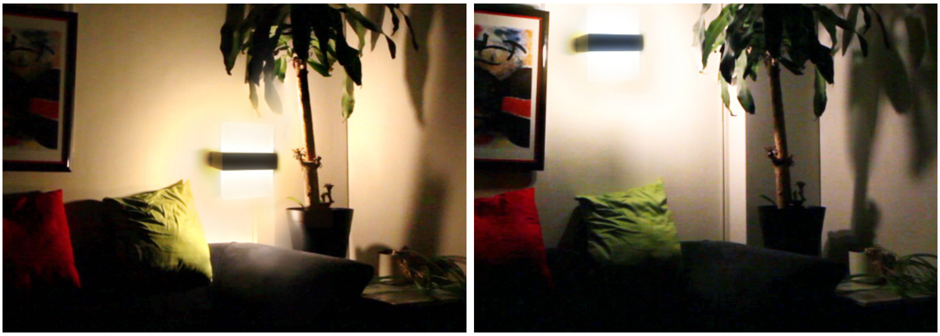


Figure 2.16: Virtual placement of a wall lamp with different positions and color temperatures.

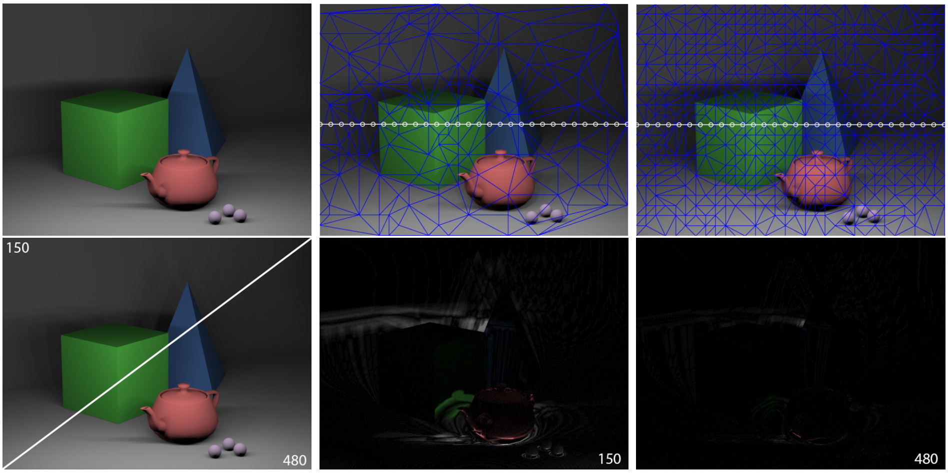


Figure 2.17: Top: synthetic ground truth (left), a sparse (150 points, middle) and a dense (480 points, right) light sampling. Bottom: our results and its respective absolute differences to the ground truth (differences increased 800% for better visualization).

the reference long exposure for a light source moving horizontally in the middle of the scene. Fig. 2.17 (bottom) shows the results obtained using our approach interpolated via Delaunay Triangulation.

2.5. Conclusion

We have presented an easy-to-use pipeline for light painting. It usually requires skill and is tedious but our solution simplifies the creation process. Our method addresses common artifacts with a novel inpainting solution, and avoids ghosts resulting from the artist being present in the scene. It requires a minimal and low-cost hardware setup, while delivering high-quality results and increased artistic freedom by moving creative choices to post-production.

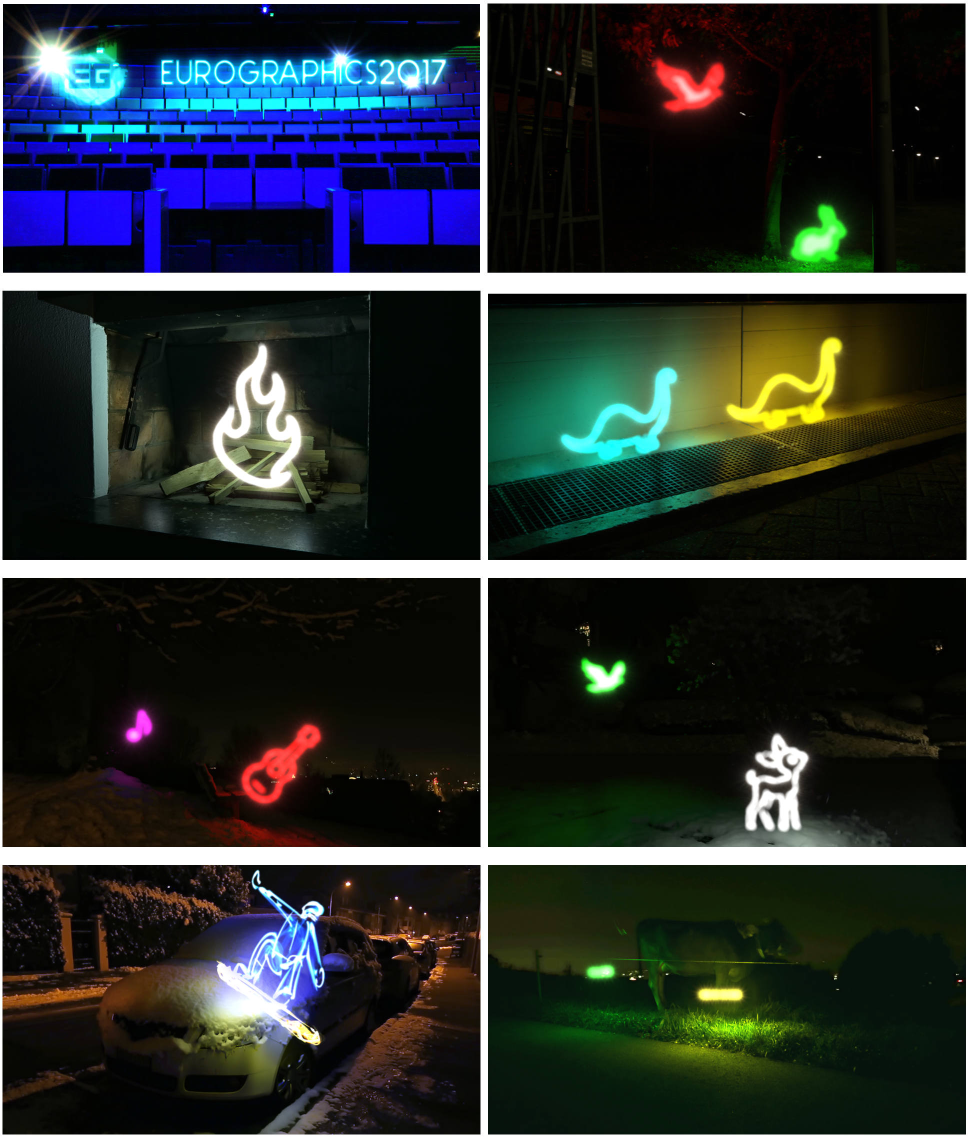


Figure 2.18: Several challenging examples illustrating our solution.

In the future, we would like to investigate new ways of scene capture, e.g., via a drone. Alternatively, an accelerometer can lead to good position estimates, even when the source is occluded. Further, polarization filters could help in removing reflections in mirror surfaces, for which we could then recompute the light reflection during reconstruction.

References

[1] N. Z. Salamon, M. Lancelle, and E. Eisemann, *Computational light painting using a virtual exposure,* Computer Graphics Forum (Eurographics) **36** (2017).

[2] M. G. Lorenzi and M. Francaviglia, *Painting with light: generative artworks or setting in motion the fourth dimension,* in *Proceedings of the 10th Generative Art Conference* (2007) pp. 12–14.

[3] M. Peres and A. Savakis, *Crowdpainting with Light: Participatory Imaging at the Big Shot,* in *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2015) pp. 32–36.

[4] M. Carlsen, *Light painting with the world's most powerful flashlights,* https://www.youtube.com/watch?v=U_iOqM8b55g (2011), accessed: 2016-09-12.

[5] L. C. Walker, *The technique of painting with light* (Nash-Jones, 1940).

[6] L. Keimig, *Night Photography and Light Painting: Finding Your Way in the Dark* (CRC Press, 2015).

[7] FilmSpektakel, *Holopainting combines light painting, stop motion, and hyperlapse,* http://bit.ly/2dOpq0t (2016), accessed: 2016-09-12.

[8] J. Miedza and J. Beckwith, *Lightpaint live,* http://lightpaintlive.com/ (2015), accessed: 2016-09-05.

[9] M-Apps, *Light painting/strokes camera,* http://bit.ly/2dU9BX5 (2013), accessed: 2016-09-05.

[10] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, and J. Davis, *Synthetic shutter speed imaging,* Computer Graphics Forum **26**, 591 (2007).

[11] A. Robertson, *Painter of light: my awkward adventures in virtual reality art,* http://bit.ly/1KV9rM1 (2016), accessed: 2016-09-26.

[12] J. S. Nimeroff, E. Simoncelli, and J. Dorsey, *Efficient re-rendering of naturally illuminated environments,* in *IN FIFTH EUROGRAPHICS WORKSHOP ON RENDERING* (1994).

[13] F. Anrys, P. Dutre, and Y. D. Willems, *Image-based lighting design,* in *Proceedings of the 4th IASTED International Conference on Visualization, Imaging, and Image Processing* (2004) p. 15.

[14] C. Manders and S. Mann, *Handheld electronic camera flash lamp as a tangible user-interface for creating expressive visual art works,* Proceedings of the 14th annual ACM international conference on Multimedia , 509 (2006).

[15] I. Boyadzhiev, S. Paris, and K. Bala, *User-assisted image compositing for photographic lighting,* ACM Trans. Graph. **32** (2013).

[16] I. Boyadzhiev, J. Chen, S. Paris, and K. Bala, *Do-it-yourself lighting design for product videography,* in *International Conference on Computational Photography (ICCP)* (2016).

[17] M. Fuchs, V. Blanz, and H.-P. Seidel, *Bayesian relighting,* in *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques* (2005).

[18] R. J. Woodham, *Shape from shading,* (MIT Press, 1989) Chap. Photometric Method for Determining Surface Orientation from Multiple Images.

[19] W. Matusik, M. Loper, and H. Pfister, *Progressively-refined reflectance functions from natural illumination,* in *Eurographics Workshop on Rendering* (2004).

[20] P. Debevec, *Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography,* in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98 (1998).

[21] P. Debevec, *The Light Stages and Their Applications to Photoreal Digital Actors,* in *SIGGRAPH Asia* (Singapore, 2012).

[22] P. Debevec, *Image-based lighting,* in *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05 (2005).

[23] R. Prevost, A. Jacobson, W. Jarosz, and O. Sorkine-Hornung, *Large-scale painting of photographs by interactive optimization,* Computers and Graphics **55** (2016).

[24] Y. J. Lee, C. L. Zitnick, and M. F. Cohen, *Shadowdraw: Real-time user guidance for freehand drawing,* ACM Trans. Graph. **30** (2011).

[25] U. Artmann, *Linearization and normalization in spatial frequency response measurement,* Electronic Imaging **2016** (2016).

[26] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, *Edge-preserving decompositions for multi-scale tone and detail manipulation,* (2008).

[27] P. Perez, M. Gangnet, and A. Blake, *Poisson image editing,* ACM Transactions on Graphics (TOG) **22**, 313 (2003).

[28] M. Tanaka, R. Kamio, and M. Okutomi, *Seamless image cloning by a closed form solution of a modified poisson problem,* in *SIGGRAPH Asia 2012 Posters* (ACM, 2012) p. 15.

[29] H. Whitaker and J. Halas, *Timing for animation* (Focal Press, 2013).

[30] T. Ritschel, M. Ihrke, J. R. Frisvad, J. Coppens, K. Myszkowski, and H.-P. Seidel, *Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye,* Computer Graphics Forum (2009).

[31] M. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, *Physically-based real-time lens flare rendering,* ACM Transactions on Graphics **30** (2011).

[32] H. Joo, S. Kwon, S. Lee, E. Eisemann, and S. Lee, *Efficient Ray Tracing Through Aspheric Lenses and Imperfect Bokeh Synthesis,* Computer Graphics Forum (Proc. EGSR) **35** (2016).

[33] S. Lee and E. Eisemann, *Practical real-time lens-flare rendering,* Computer Graphics Forum (Proc. of EGSR) **32**, 1 (2013).

[34] M. Fuchs, H. Lensch, V. Blanz, and H.-P. Seidel, *Superresolution reflectance fields: Synthesizing images for intermediate light directions,* in *Computer Graphics Forum*, 3 (2007) pp. 447–456.

[35] J. Wu, *Rhythm of light,* https://vimeo.com/184667838 (2016), accessed: 2016-10-06.

3

Controllable Motion-Blur Effects in Still Images

X. Luo, N. Z. Salamon and E. Eisemann

Sprinkled by the trappings of words that make the outlines;
Blur on the showplace of made history.

Motion blur in a photo is the consequence of object motion during the image acquisition. It results in a visible trail along the motion of a recorded object and can be used by photographers to convey a sense of motion. Nevertheless, it is very challenging to acquire this effect as intended and requires much experience from the photographer. In this paper, we propose a novel method to add motion blur to a single image that generates the illusion of a photographed motion. Relying on a minimal user input, a filtering process is employed to produce a virtual motion effect. It carefully handles object boundaries to avoid artifacts produced by standard filtering methods. We illustrate the effectiveness of our solution with various complex examples, including multi-directional blur, reflections and multiple objects. Our post-processing solution is an alternative to capturing the intended real-world motion blur directly and enables fine-grained control of the motion-blur effect.

This chapter has been published on Transactions on Visualization and Computer Graphics (2020) [1], and is an extended version of the conference paper published at the Graphics Interfaces (2018) [2].

3.1. Introduction

Motion blur, as an artistic effect, is able to convey a sense of motion in still images. A photographer can create such effects by opening the camera shutter for an extended period of time. During the exposure, moving objects with respect to the camera will result in a different projection location on the camera sensor, which produce visible trails in the final image [3]. While being an important technique [4], it is very challenging to control or acquire an intended result. Parameters such as the shutter speed, camera motion, illumination, lens and filter configurations strongly influence the result but their effect is difficult or even impossible (e.g., if the object is moving irregularly) to estimate. Further, capturing slowly moving objects, such as stars or clouds, requires a very long exposure to convey even a small sense of motion. In some other cases, a photographer might want to keep a fast moving object in focus, which results in only the background being affected by the motion blur. To achieve this, the photographer needs to precisely follow the object with the camera to keep the position perfectly stable, which is very challenging. Similarly difficult is the production of a precise camera/object trajectories on the image plane.

Easier than capturing the result directly, is to produce it in a post-process. However, current image editing software provides mostly blur tools for general purposes, which requires users to manually extract regions of interest and experiment with different effects. Additionally, standard filters typically result in artifacts at object boundaries, such as undesired color leakage.

Our method enables a user to easily add motion blur to a single still image with only little user intervention. To this extent, we propose a segmentation tool to select objects of interest to then define the intended motion blur. Multiple objects can be extracted and motion paths freely chosen. Our method relies on an edge-aware filtering to deliver convincing results, while keeping the user interaction simple and avoiding additional scene information. The algorithm in this paper builds upon our original method presented in [2]. This extended version provides a user evaluation and several new contributions, namely, a multi-directional motion blur to support more complex object and camera motions, as well as several approaches to produce artistic effects based on commonly applied shutter techniques.

This article is organized as follows. In the next section, we revisit previous work. We then describe our approach (Sec. 3.3), including methods to produce various artistic effects (Sec. 3.4), before presenting the results of our method (Sec. 3.5) and concluding (Sec. 3.6).

3.2. Related Work

Blur in photography is often used for artistic purpose, to guide the observer, emphasize important elements, or achieve a desired look and composition. The two most common sources of camera blur are motion blur and depth of field.

Depth of field has received much attention and is also a perceptually well explored effect [5]. Several hardware and algorithmic solutions have been proposed. Light-field cameras [6], special sensors [7], coded apertures [8], stereo setups [9], or synthetic reconstruction [10], enable post-processing of the depth-of-field effect.

Motion blur conveys a sense of motion but is often considered an undesirable artifact, as it can result from camera shake. In consequence, modern cameras usually involve stabilization systems [11] to avoid the effect. Our goal is to allow a user to control motion blur for artistic purpose.

For an image sequence, a computational solution to reconstruct motion blur has been proposed in form of the virtual exposure [12]. Here, short exposure shots are combined to simulate a long-exposure result. Originally conceived to simulate high-dynamic range photography, the work addresses also moving objects. Direct accumulation of the images would result in ghosting artifacts due to an exposure gap between the individual shots. They rely on an optical flow algorithm to fill in the missing transitions to obtain a motion-blurred output. Commercial systems, such as Reel Smart Motion Blur [13] rely on optical flow to estimate the movement between images to then apply a directed blur kernel. A virtual exposure was also used for light painting, which is able to describe the motion blur caused by a moving light source [14].

When relying on video input, a pixel-wise blur can be generated from an estimated motion trajectory of the object [15]. The authors create a blur kernel from the disparity along motion vectors in the stabilized video. However, background and camera motion cannot be decoupled and the blur is limited to moving areas. Moreover, the video input can grow significantly when dealing with slow moving objects, such as clouds or stars. Our method can be applied to any object/region in a single image.

Commercial solutions for computational motion blur on a single image exist. One example is the GIMP [16] motion blur tool and the Adobe Photoshop [17] motion-blur effect. These require significant manual intervention; object segmentation, inpainting and manual organization of layers need to be performed beforehand. Our integrated solution works directly on a single image and facilitates control and definition of motion blur.

Motion depiction has also been investigated for virtual scenes, even with a programmable interface for expressive results [18]. For efficient motion blur computation, perceptual factors can be integrated in the computations [19]. Most real-time 3D applications, such as [20–23], make use of deferred shading [24] to derive information such as per-pixel motion, depth, or object id, which are used in a filtering process. Our method shares ideas regarding post-processing but relies on a single photograph.

3.3. Our Approach

Our algorithm adds motion-blur effects to a single image based on a few simple user annotations. Fig. 3.1 shows an overview of our solution.

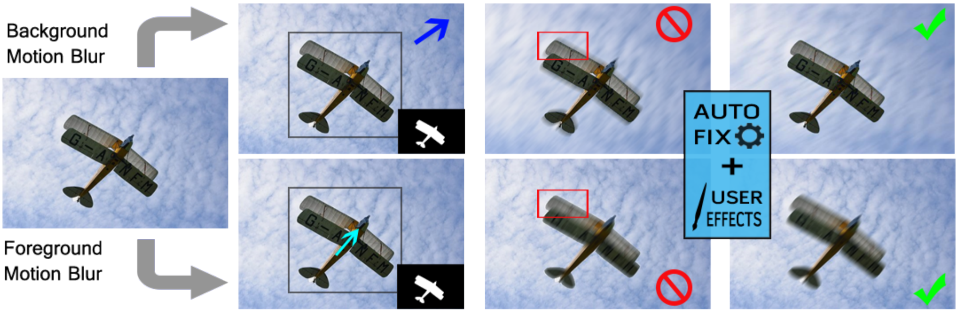


Figure 3.1: Overview. From left to right: the user can select target regions of an input image using simple annotations (bounding box, scribbles). A motion can be defined for the desired target region, which launches a filtering process to add plausible motion blur. (Input image source: pixabay.com)

The user can select objects via an image-segmentation method (Sec. 3.3.1) and can then define the motion of the selected object or area. The algorithm produces a motion-blurred result while avoiding artifacts around object boundaries (Sec. 3.3.2). First, we will describe our solution for linear motion per object before addressing general motion paths per pixel (Sec. 3.3.3). Additionally, we present several extensions of our approach to add high dynamic range (Sec. 3.4.1), and artistic effects inspired by photographic techniques (Sec. 3.4.2 and Sec. 3.4.3).

3.3.1. Object Selection

A moving object in the foreground blends with the background, while a moving background does not blend into a static foreground. The differing visibility relationships lead to very different outcomes; a static foreground object will cover the background during the entire exposure and will maintain crisp boundaries, while a moving foreground object will result in a fuzzy boundary. This difference makes it necessary to distinguish the order of the objects present in the image. Consequently, we will first focus on how to extract objects from the image.

For the segmentation of objects, a variety of methods could be used. Recently, machine learning techniques (e.g. [25–27]) are increasingly successfully employed for segmentation tasks. Still, for creative content creation, users might desire segmentation masks that differ from the typical automatic segmentation inferred from a learning process. For this reason, we opted for a constrained segmentation based on user annotations. One of the most user-friendly segmentation methods is GrabCut [28]. The user defines a rectangle containing the potential foreground object. Additional scribbles can be provided to refine the mask segmentation. GrabCut then partitions the image into foreground and background pixels. We then separate the foreground pixels into connected components [29] to define the different objects.

Fig. 3.2 shows two examples of the GrabCut extraction. The user defined an object's bounding box and, if necessary, scribbles, which assign potential foreground and background regions. The extracted mask defines the foreground object (the thumbnails in the lower right corner).

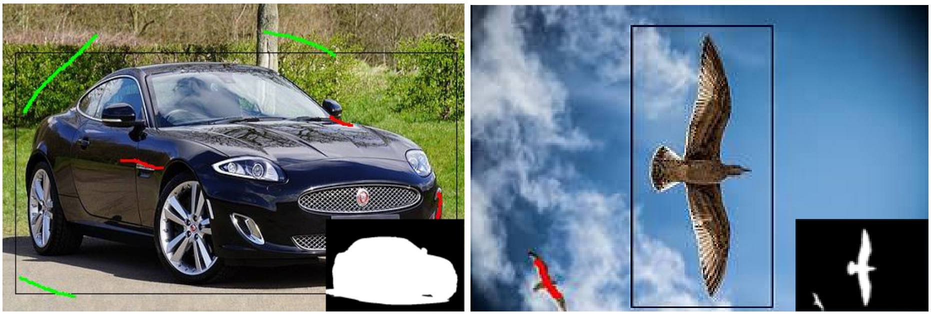


Figure 3.2: GrabCut foreground extraction. (Images source: pixabay.com)

In case that foreground objects overlap, the algorithm can be recursively applied by running the GrabCut on the previously extracted foreground regions. In each step, the output results in a fore- and a background label, which induces an ordering of the objects, which can be adapted by the user. These objects can then be processed individually with their own motion path.

During our experiments, we found that we rarely need to distinguish more than four objects. Hence, we allow the user to determine directly four levels of ordering in the interface by drawing corresponding scribbles. Fig. 3.3 illustrates a multi-object labeling.

In order to ease explanations, we will drop the foreground and background labels and refer to all extracted regions as objects, which are ordered from back to front.

3.3.2. Uniform Motion Blur

A linear motion can be defined by a motion direction and length. We simulate the motion blur by convolving an object with a motion-blur kernel (*psf*) defined by the motion trajectory. For example, a horizontal translation by n pixels results in a horizontal kernel of n pixels, which is normalized such that its integral (sum of all pixels) is equal to one. The latter avoids creating energy when convolving the input. To compute the kernel for a general linear motion we use the formulation proposed in Matlab. First the bounding box of the provided segment indicating the motion is determined and extended by two pixels. Then, for each bounding-box pixel, we compute one minus the distance of the pixel center to the segment. Next,

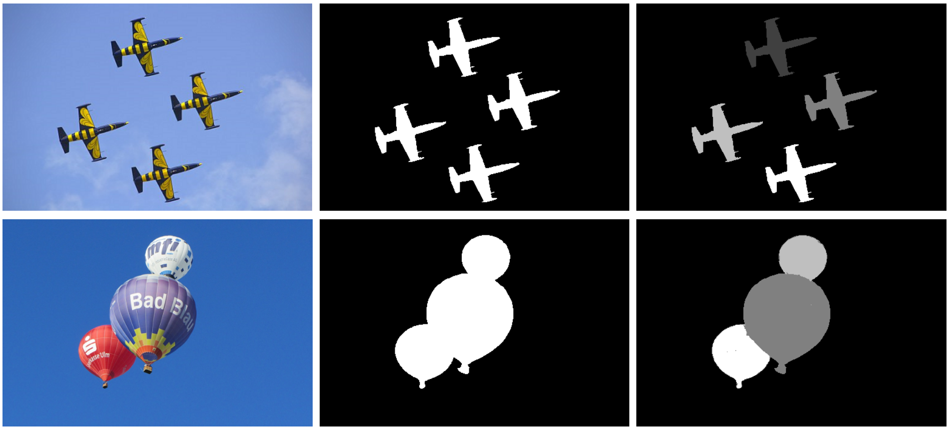


Figure 3.3: Multiple objects segmentation. Distinct target regions can be segmented, allowing localized and distinct effect control. (Images source: pexels.com (top) and pixabay.com (bottom))

all values are clamped between zero and one to eliminate negative values. Finally, the resulting kernel is normalized by the total sum of all pixels. For general motion paths, the provided path is decomposed into linear segments, which are individually handled as before.

Having derived a blur kernel per object, it seems tempting to visit every pixel of the labeled input image and simply apply the corresponding *psf* kernel. Unfortunately, this results in color bleeding artifacts, as illustrated in Fig. 3.4.

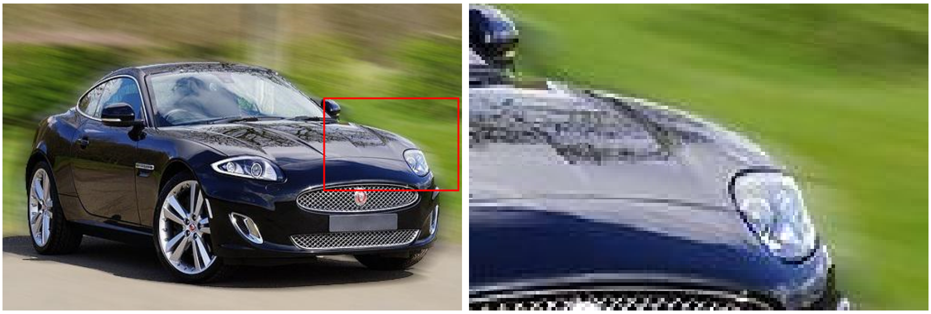


Figure 3.4: Color leakage when not respecting object boundaries.

Similarly, when applying edge-aware filtering, which avoids blurring across object boundaries, the result is unrealistic. Sharp boundaries are maintained for moving foreground objects (Fig. 3.5), while one would have expected a fuzzy boundary.

For a more plausible result, we will derive blending masks to composite the objects from back to front, one by one. In other words, a given object is motion

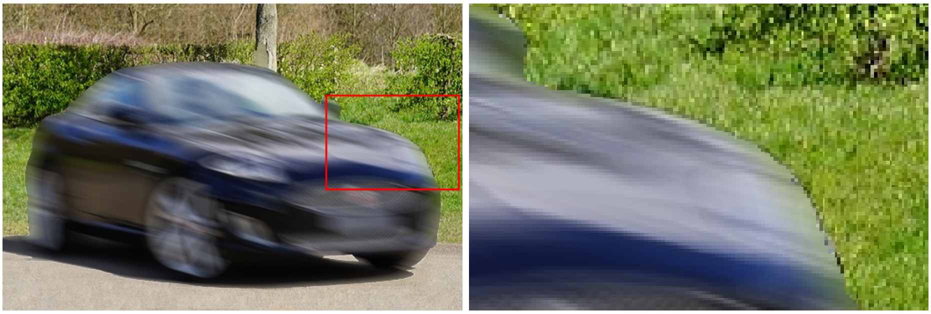


Figure 3.5: Unrealistic sharp edges from edge-aware filtering.

blurred, and then composed with the current background, which handles the problems in Fig. 3.5. After explaining the corresponding details, we will show how to address the issues of Fig. 3.4.

Composing Fore- and Background

To describe the algorithm to compose fore- and background, we will focus on the steps for a single object. Let B_k be the current background image (B_0 is initialized with zero). For an object O_k, we produce an image I_k, which is a black image into which we copy all pixels labeled with O_k from the input. Further, we produce a mask M_k, which is black except having ones in pixels that correspond to those labeled with O_k (one can interpret this step as adding an alpha channel). We then convolve both images with the *psf* of O_k. Intuitively, the image $psf * M_k$, where $*$ denotes the convolution, corresponds to a mask that indicates how much the foreground will occlude the background. For example, if the object is not moving, *psf* is by construction a Dirac (a single pixel equal to one), which implies that the mask describes exactly the pixels of the original object. Given the convolved results and the background B_k, we compute the new background B_{k+1} as $B_{k+1} = psf * (M_k I_k) + (1 - psf * M_k)B_k$.

In practice, it is possible to avoid the actual derivation of the masks by performing an integration along the kernel directly on the input image. Further, we do not need intermediate background images and it is enough to incrementally compose the motion-blurred objects in a single resulting image.

While conceptually simple, the above approach is still imperfect. It relies on the assumption that each object is entirely visible in the original input image. Unfortunately, this is rarely the case. As soon as an object is moving, visibility relationships change and parts previously-occluded by the object will be revealed. A challenge is to estimate the content that is disoccluded. Neglecting disoccluded regions and assuming that they look like the original image will result in the artifacts shown in Fig. 3.4. Similarly, assuming the disoccluded pixels are simply black results in a dark halo.

Handling Disocclusions

To correct for the disocclusion artifacts, we propose an inpainting procedure. While more advanced solutions could be employed (i.e. [30–32]), we found the simpler strategy usually sufficient. The reason is that the part will either be in motion itself or overlapped by a moving element, which naturally hides many of the details in the inpainted area.

Adding inpainting to our solution, the main algorithm remains the same; objects are treated front to back, but before filtering with their *psf*, an inpainting procedure is applied. For an object O_k, we will examine its boundary to find pixels adjacent to an object O_j that is nearer (i.e., $j > k$, as objects are ordered). If there is none, O_k does not require any inpainting. If there is an overlap, we want to extend O_k beneath the potentially uncovered region of O_j. Inspired by recent real-time methods [23], we mirror the content of O_k into the area that is covered by O_j. We choose the mirror direction d based on the motion direction of O_k (or of O_j in case that O_k is static). The value of a pixel p in O_j is then defined by finding a pixel q from which we copy the value. We determine the position of q by walking from p towards O_k along d, one pixel at a time. On the way, we maintain a counter, initialized at one, that is incremented whenever an encountered pixel is inside O_j and decremented when outside O_j. When the counter reaches zero, we have found q. The zero counter indicates that we have traveled the same distance inside O_j as outside, it is then located at a reflected position with respect to the object boundary. If we encounter an object O_l ($l > j$) while following d, we reverse d, which performs a *ping-pong* inpainting. Using this simple inpainting leads to a significant improvement (Fig. 3.11).

3.3.3. Multi-Directional Motion Blur

Now that we have discussed the case of per object motion, more complex motion paths for each pixel are addressed with our multi-directional motion blur. This extension increases the expressiveness of the algorithm significantly. For example, when photographers move a camera forward or sideways, the perspective foreshortening leads to blur effects that can enhance the impression of depth, yet they cannot be described with a single linear path per object. In other cases, such as a spinning wheel, a linear motion trajectory fails in reproducing a plausible motion-blur effect. In this section, we propose an extension to the previous principle. Here, unlike convolving the original image region with a single motion-blur kernel, each pixel will receive its own motion direction. In order to facilitate the annotation, the user defines a few motion paths, whose properties are propagated throughout the image using a diffusion process [33, 34]. This diffusion process results in an image with minimal gradient variation, under the constraint that the original annotations are maintained.

To describe this process formally, we will derive an image, in which each pixel will contain a motion vector defined by a length l and a direction $d := (\cos(\theta), \sin(\theta))$ for a given angle θ. Initially, the user will only sparsely annotate a few pixels. Let $\mathcal{I}$ be the set of pixels for which the user provided an input. For an index $(i,j) \in \mathcal{I}$, we denote the user-defined motion annotation as $m_{i,j}$. The image M containing the diffused three-component motion vectors is defined by $\Delta M = 0 \quad with \quad M(i,j) = m_{i,j} \ \ \forall (i,j) \ \in \ \mathcal{I}$

The resulting image M is smooth, as the equation $\Delta M = 0$ implies that the gradient is minimized, while the user annotations are maintained. This equation system can be solved with an iterative process. To this extent, one can iteratively average neighboring pixels via $M(i,j) = (M_{i-1,j} + M_{i+1,j} + M_{i,j-1} + M_{i,j+1})/4$, while maintaining the values of the pixels in $\mathcal{I}$, until convergence. More efficient solutions involve using multi-grid [33], sparse-system solvers [34], or even optimized methods for diffusion curves [35].

The initial user annotations of the pixels in $\mathcal{I}$ are done using a special annotation tool. The user defines the motion direction by drawing a line segment whose orientation and length define the desired values (d,l). By default, the pixels below the segment will be added to $\mathcal{I}$ but it is also possible to mark an area, or single pixel, with a brush to then associate the drawn segment to this area.

Given the diffused motion vectors M, the next step is to derive the corresponding motion-blurred image I_m from the input image I. The idea is to start in each pixel and walk along the defined motion trajectory. This path line integration is similar to the process of visualizing flow [36]. For each pixel p, we compute the motion-blurred result $I_m(p)$ by averaging the values along a path of the motion length $\lfloor M(p).l \rfloor$ over the input image I, guided by the direction $M(p).d$:

$$I_m(p) = \frac{1}{\lfloor M(p).l \rfloor} \sum_{i=0}^{\lfloor M(p).l \rfloor} I(p_i),$$

where

$$\begin{aligned} p_0 &= p \\ p_{i+1} &= p_i + M(p).d \end{aligned}$$

Disocclusions during this multi-directional motion blur are handled similarly to before, only now the mirror-padding is applied according to the motion path. Fig. 3.6 shows a result (bottom row, right). The user-provided motion paths and the diffused result are illustrated as well.

3.4. Extensions

Our method is able to add the illusion of a standard motion blur to a still image. In this section, we will describe extensions of our solution. First, we show how

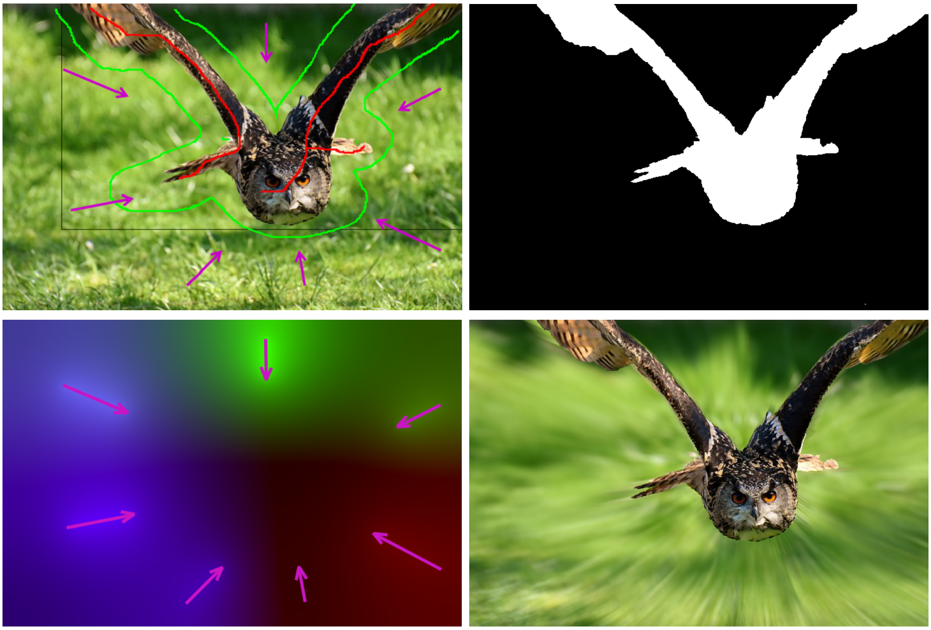


Figure 3.6: Multi-directional motion blur. Top row: the input image with user scribbles and multiple motion paths (left) and the segmentation mask (right). Bottom row: the diffusion map M with the colors encoding the direction and length of each motion path (left), and the final result (right). (Image source: pixabay.com)

to increase the realism of the motion blur for very bright sources by hallucinating high-dynamic range content, then we propose two artistic additions, which are often used in practice, the Harris shutter and addition of motion trails.

3.4.1. High Dynamic Range Motion Blur

For strong light sources exceeding the range of the sensor sensitivity and thus the pixel values, the impact on the appearance of the motion blur can be easily underestimated. In real-world environments luminance can span a wide range. While our human eyes can adapt to large intensity variations, with standard photography, values in the sensor might saturate. Bright and glowing elements are often clipped, e.g., a bright car headlight in a night scene. High-dynamic-range (HDR) imagery is produced by recording several images with different exposure times, which are fused to capture a larger range of intensities. Working with a high-dynamic range representation has a significant effect on the result. A clipped value when blurred will lead to a dimmed result in comparison with its original version. Having values that exceed the limits of the display will still be dimmed by a blur, but will maintain a higher intensity and potentially even still saturate after the blur is applied. To achieve this effect, we propose to virtually produce HDR content.

In our solution, a user can indicate regions in which values were potentially clipped by placing a bounding rectangle around them. Then our solution expands the values in this region from the range of $[t, 1]$ to a range of $[t, 2^T]$, where t and T are user-defined thresholds (per default, $t = 0.98$, $T = 2$) using the function $f(x) = t * pow(1 + (x - t)/(1 - t), T)$. While very coarse, the accuracy of such an expansion is of lower perceptual significance, still advanced conversions would be possible [37].

Fig. 3.7 shows an example of hallucinated HDR content created with our solution. The light blue rectangles illustrate the selected the region for the HDR expansion. The change affects the look of the motion blur, which results in a more realistic effect (right) compared to the standard approach (middle).

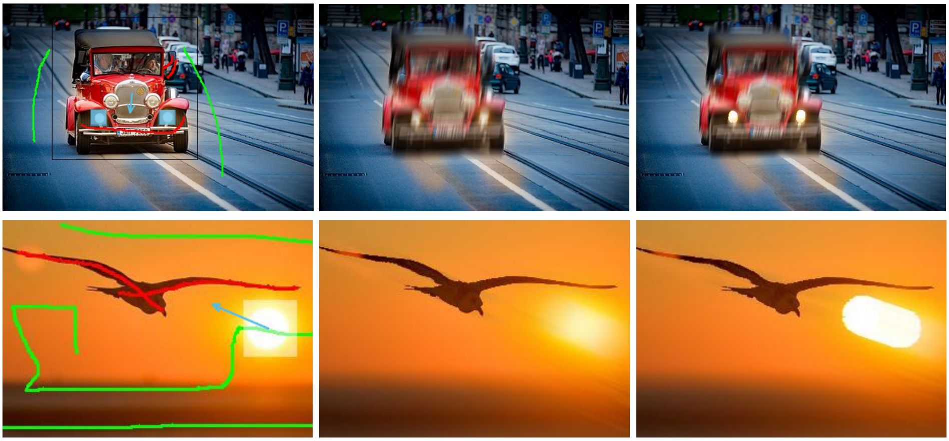


Figure 3.7: Hallucinated HDR expanding high intensity values. The bright lights of the car create visible trails (top), as does the sun when applying a strong background blur (bottom). (Images source: pixabay.com)

3.4.2. Harris Shutter using Motion Blur

To show the flexibility of our solution and ability to also reproduce artistic effects used in photography, we added support for the simulation of a Harris shutter effect. The effect conveys an appealing and colorful outcome by masking certain channels over time. Typically, the shutter effect consist of capturing the scene in different time intervals, using only a single channel for each exposure [38]. Alternatively, it can also be achieved by recording a video and using the complementary channels from different frames to composite the final image. In other words, motion in a scene will result in several differently-colored projections.

In our solution, the motion blur is used to create the time-shifted frames. The target region is defined by the user and we manipulate each channel separately. The green channel is used as reference (middle) point. The red and blue channels contain the result after following half the length of the provided motion blur in opposite directions. Results of the Harris shutter simulation are shown in Fig. 3.8. With this artistic effect, one can steer attention to scene composition.

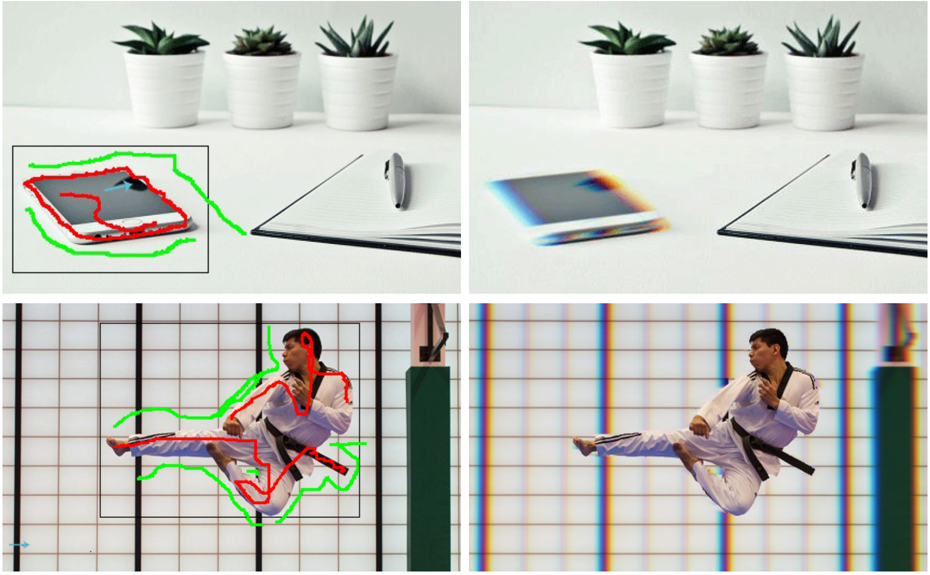


Figure 3.8: Harris shutter motion blur results in a colorful outcome to steer attention. (Images source: pixabay.com)

3.4.3. Motion Trails

Another artistic means to illustrate movement are motion trails. These are used in photography but usually shot in front of a dark background. Here, a flash or strong light is used at the end of the capturing process. Hereby, the object in its final position will be more visible than during the previous time frame. As a consequence, it seems as if the object leaves a trail behind that is easily understood by the observer as a displacement. We can simulate such an effect by compositing the foreground object on top of the motion-blurred result.

An example is shown in Fig. 3.9, using the multi-directional blur to create motion trails simulating the punch impact. Please notice that this effect can also be selectively applied on different parts of an object by using the diffused multi-directional motion blur.

3.5. Results

We have implemented our framework using OpenCV/C++. All results were created on a laptop with an Intel i5 2.2GHz and 8GB RAM. We did not optimize the per-

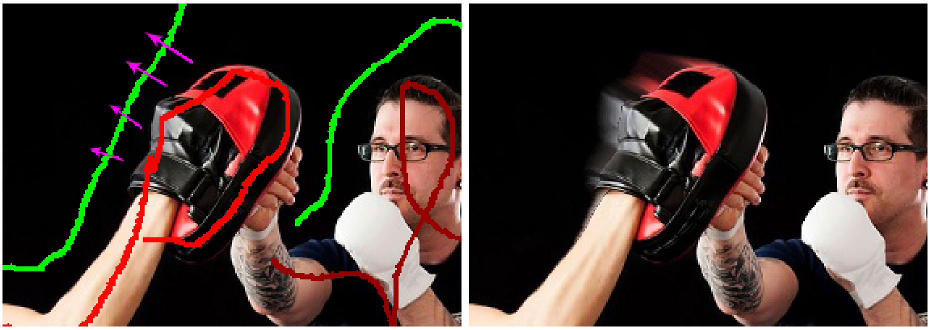


Figure 3.9: Motion trails indicating the action while keeping the final position on focus. (Image source: pixabay.com)

formance of our solution. The uniform motion blur is linear in terms of complexity with respect to the image dimensions. It takes 3 seconds to segment the content and apply a motion path on a 960×540 image. Since the input is simple, novice users can create convincing results in less than 10 seconds. For the multi-directional motion blur, the computation time is directly linked to the diffusion process. The image size, number of motion paths, and number of iterations until convergence do govern the cost. While it would be possible to use hierarchical [33] or advanced solvers [34, 35] on the GPU , which can run at interactive rates, we use a standard CPU solver to increase compatibility. In practice, users took around 2 minutes to obtain the results shown in this paper.

A large variety of examples are illustrated in Fig. 3.25. The top row (a-b) shows a boy with a ball, where the motion blur on the ball adds activity to the scene and guides the observer's focus. The same row (c-d) adds a clear sense of speed to the horse movement that was missing from the original shot. On the second row, a similar result is obtained: in (a-b) the background was blurred to underline the stormy sea and sky, which leads to an increased focus on the surfer; (c-d) show that the gradient in the sky remains almost perfectly unaltered. Row three illustrates how motion blur can emphasize actions to underline the semantics of a photo. The fourth row (a-b) illustrates the smoothness of the motion-blurred results, even in the presence of a complex path, which adds to the calm atmosphere of the photo. The same row (c-d) shows how our HDR effect can add to the apparent brightness of the back lights. The motion blur is used to add a sense of danger with regard to the slippery road in the image. The fifth row, illustrates the effects of the Harris Shutter, where the originally simple scenes are enriched by the vivid color additions. Finally, the sixth and the seventh rows show applications of the multi-directional motion blur involving diffused motion vectors. In the sixth row, we show how the motion blur can support the emphasis on the main character (a-b), or create the impression of vertigo (c-d). The seventh row shows a subject emphasized using motion lines (a-b) and also a spinning motion to illustrate the generality of the solution (c-d). The images exemplify the large variety of options for the controlled use of motion

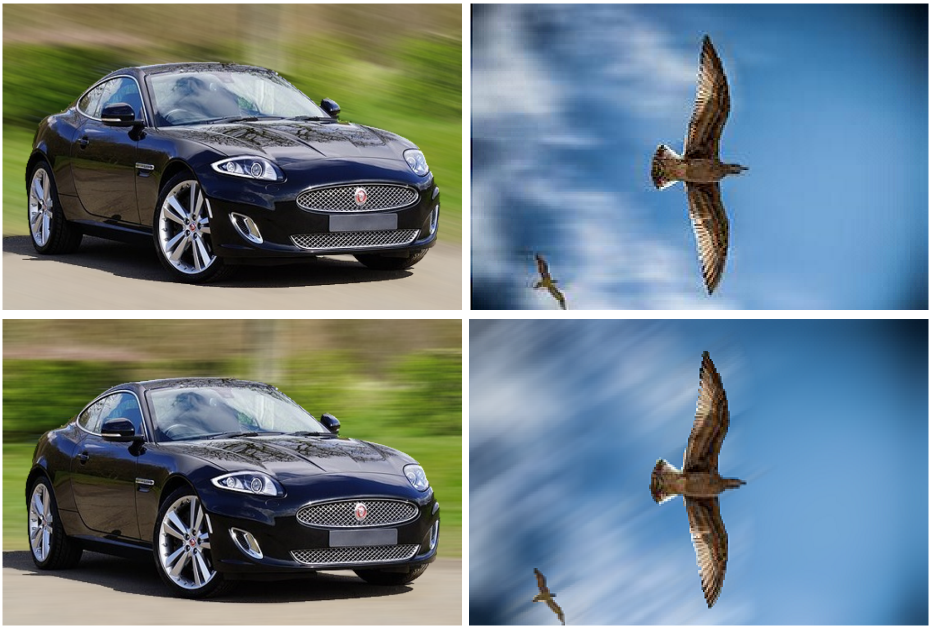


Figure 3.10: Background motion blur results. Rows differ on the motion vector chosen by the user for the same objects.

blur. In the following, we demonstrate key features of our algorithm.

The user defines objects with very little effort, as evidenced by the simple input previously shown in Figs. 3.2 and 3.3. To apply motion blur to the background, a user can draw a motion vector over the desired area. The color leakage artifacts seen in Fig. 3.4 are minimized by our approach, creating a natural transition along object and background edges, as shown in Fig. 3.10 (top). The user can decide to change the motion path at any time to explore the resulting effect and also can apply to more cases, such as in Fig. 3.10 (bottom). Fig. 3.11 illustrates cases where objects are set in motion, like the car (left) and eagle (right). Fig. 3.11 (top) illustrates the corrected result from Fig. 3.5. For a matter of comparison, Fig. 3.11 (bottom) shows a different motion direction applied to the target objects.

For scenes with more than one object, each object can be motion blurred with different motion paths and intensities. Fig. 3.12 (left) shows one motion-blurred target (one balloon, one dice), while Fig. 3.12 (right) shows the result when simulating different motion directions and speeds. Analogously, Fig. 3.13 illustrates how the impression of a scene can be influenced when switching the motion targets; here, either to the player (left) or to the ball (right). Similarly, motion blur can be used to guide an observer, such as in Fig. 3.14, where the hand motion influences the way an observer analyzes the scene.

When using general motion paths, they can be treated as several linear seg-

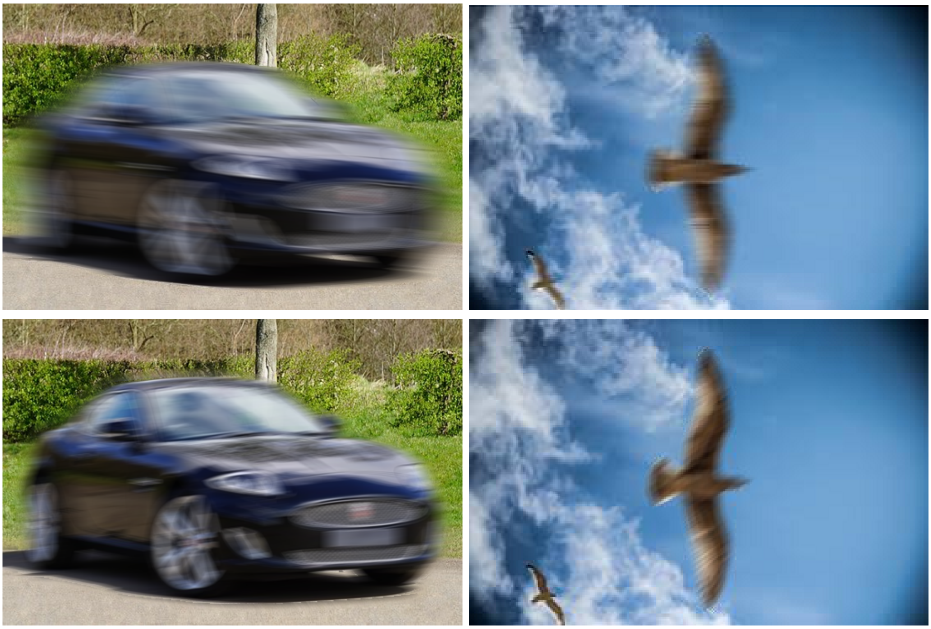


Figure 3.11: Foreground motion blur results. Our artifact minimization blending is applied to all results. Rows differ on the motion vector chosen by the user for the same objects.

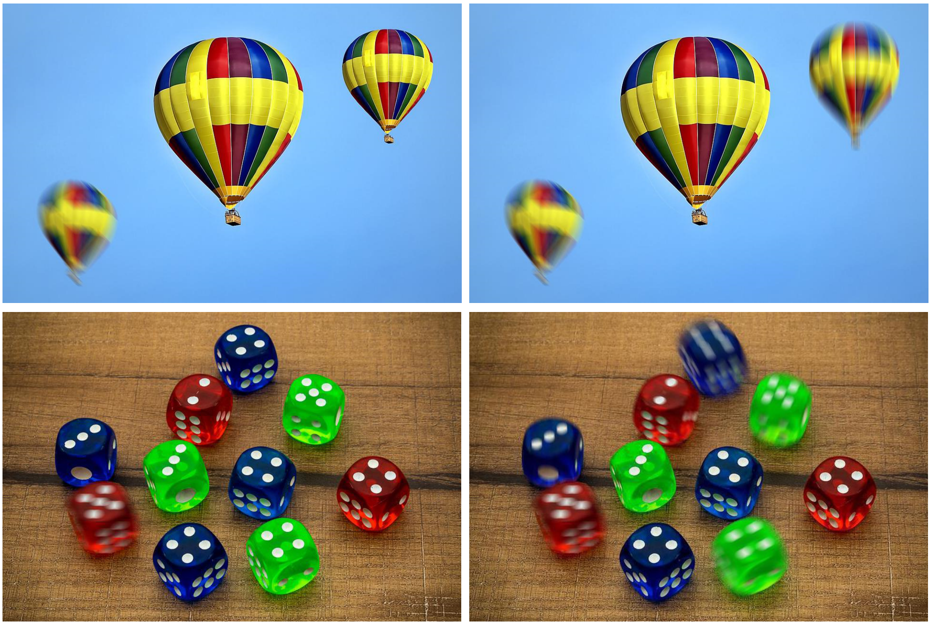


Figure 3.12: Multiple objects with distinct motion directions. (Images source: pixabay.com)

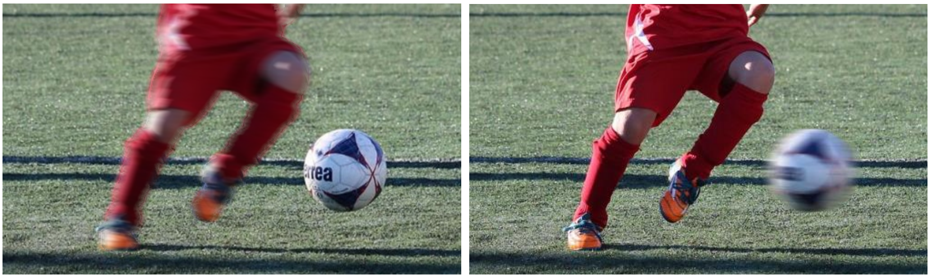


Figure 3.13: Motion blur on different targets, changing scene impression. (Image source: pixabay.com)

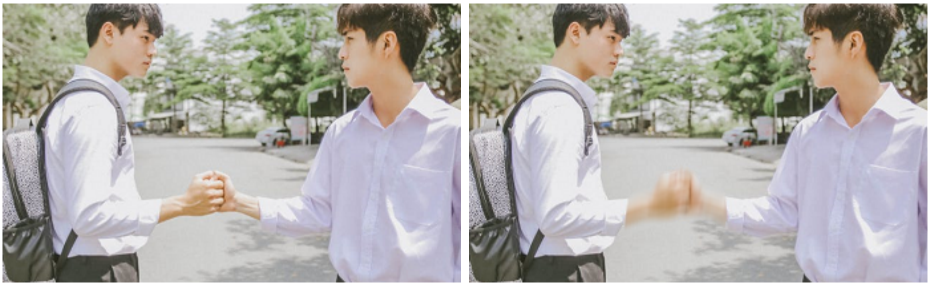


Figure 3.14: Motion blur indicating the semantics of the scene. (Image source: pixabay.com)

ments. In practice, it is typically sufficient to directly apply a convolution with the whole path and only perform a vertical and horizontal occlusion handling, depending on the local orientation, which is more efficient to evaluate. General motion curves are well suited to simulate a long exposure with non-linear motion, e.g., due to a hand-held acquisition. Fig. 3.15 demonstrates a curved motion paths on the ball to simulate a non-linear bounce (top) and a time-lapse sequence (bottom).

Fig. 3.16 shows multi-directional motion blur results with a complex motion per pixel. The purple arrows indicate the different directions of the motion paths. The results artistically emphasize the character (top), increase the perception of speed (middle), and simulate the spinning motion on a time-lapse (bottom).

Fig. 3.17 uses hallucinated HDR content to maintain the brightness of the sun. Fig. 3.18 adds the Harris shutter effect, while Fig. 3.19 shows expressive motion trails.

Discussion

Our approach introduces novel effects that are not easily reproducible in standard software. Motion blur can also be applied to an object using Photoshop, but it requires an extended workflow explained in the following. Fig. 3.20 shows the outcome of our method and a manual manipulation in Photoshop. Here, the background received a linear motion blur. Just applying Photoshop's Motion Blur and

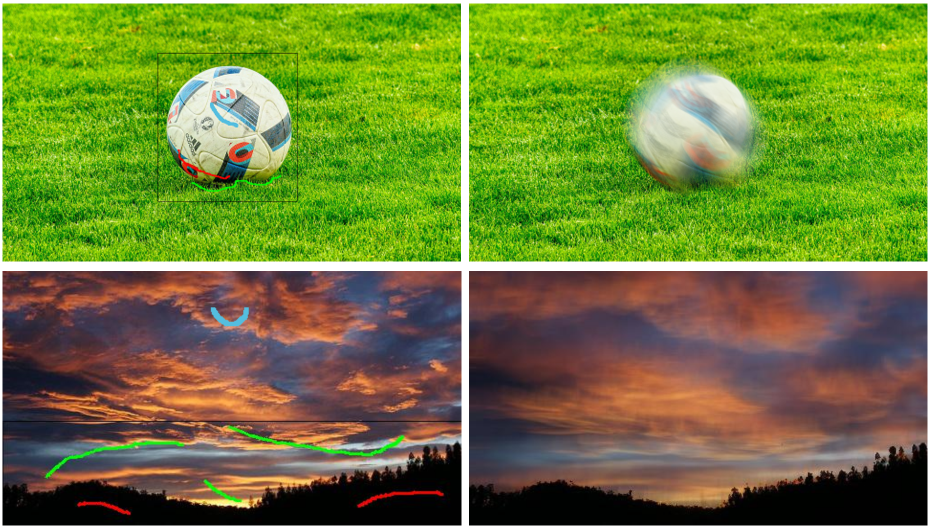


Figure 3.15: Uniform motion blur with a non-linear path. (Images source: pixabay.com (top) and freegreatpicture.com (bottom))

Path Blur tools (left, center-left) leads to similar artifacts as in Figs. 3.4 and 3.5. In consequence, an artist needs to first derive layers, which can be non-trivial. Further, each layer requires manual inpainting, which can be a difficult task and require experience, especially for complex content. After processing the layers, a manual compositing is needed to derive an acceptable result (center-right). Our approach leads to similar results (right), while avoiding manual layering and compositing automatically.

Despite its simplicity, our method's inpainting typically enables natural looking motion-blurred result for moderately strong motion blur and requires no user interaction for occlusion handling. The inpainting mechanism provided by Photoshop [30, 39] is more general and often provides a very detailed infilling. Nevertheless, it does not take the motion direction into account and might create unwanted structures that cannot be found in close proximity of the inpainted area. A comparison is shown in Fig. 3.21.

A direct comparison of our method to manually recreating motion blur effects in professional software tools is difficult, as the expertise of the users plays a significant role. To still provide some insights on practical usage, we chose to perform a simple evaluation with 10 users. These users had varying degrees of expertise in Photoshop, but used our solution for the first time. We gave them three photos that were relatively easy to segment and decompose in Photoshop to not require much expertise. They were asked to mimic a motion-blurred result. Overall, our system was still considered easier to use (4.3 (our) vs. 2.9 (PS) on a Likert scale of 5 - higher being better). Less time was spent using our method to create the desired

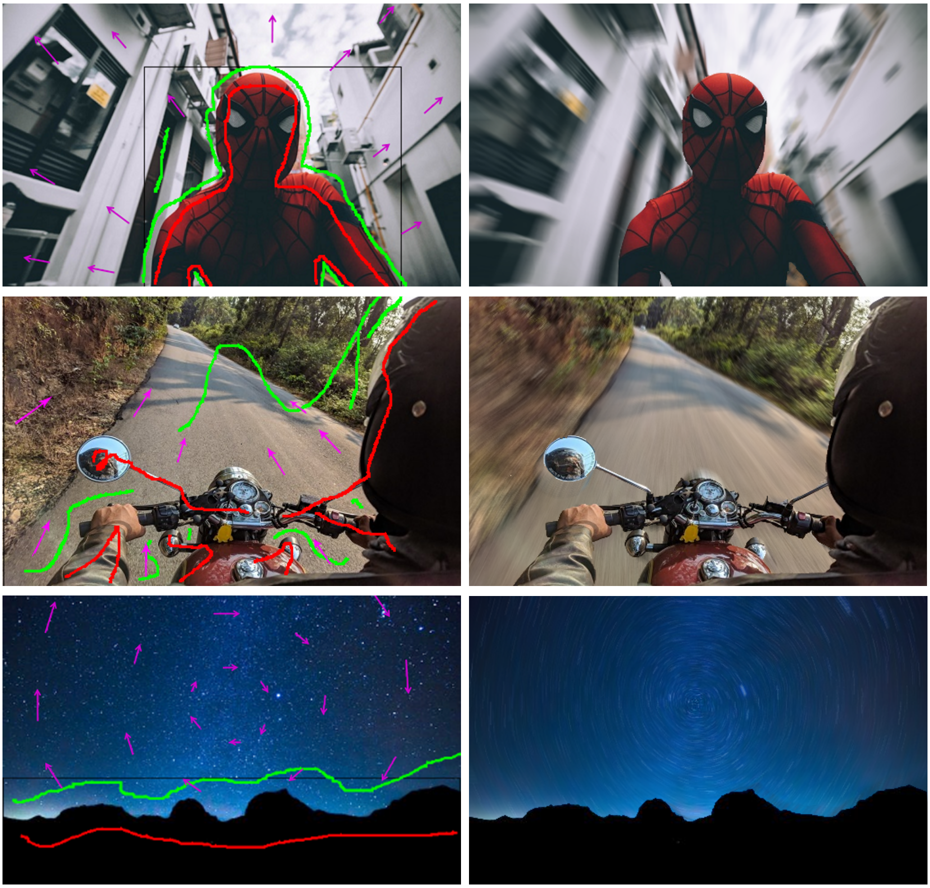


Figure 3.16: Multi-directional motion blur results. (Images source: unsplash.com (top), pexels.com (middle), and pixabay.com (bottom))

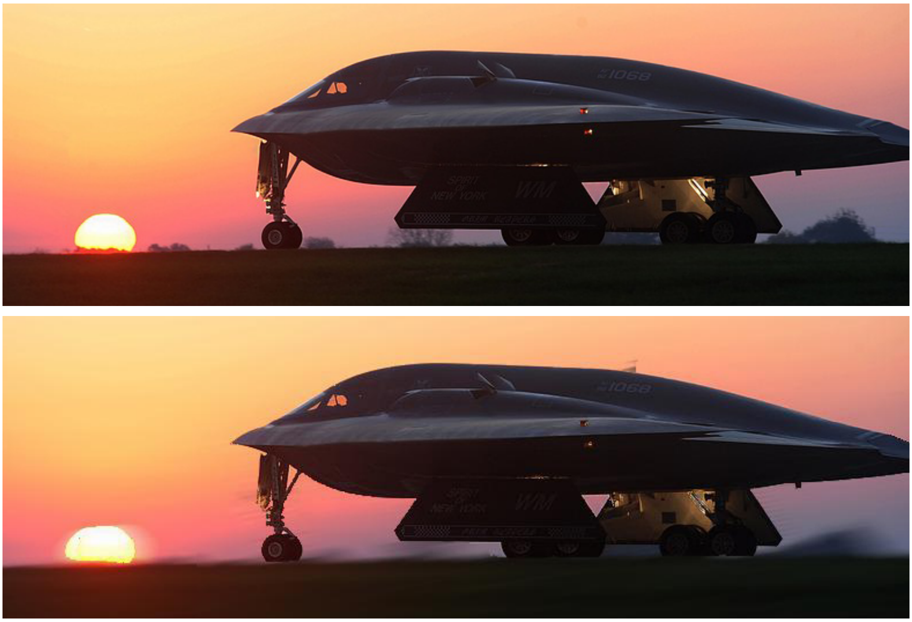


Figure 3.17: High Dynamic Range motion blur keeping high intensity values. (Image source: pixabay.com)

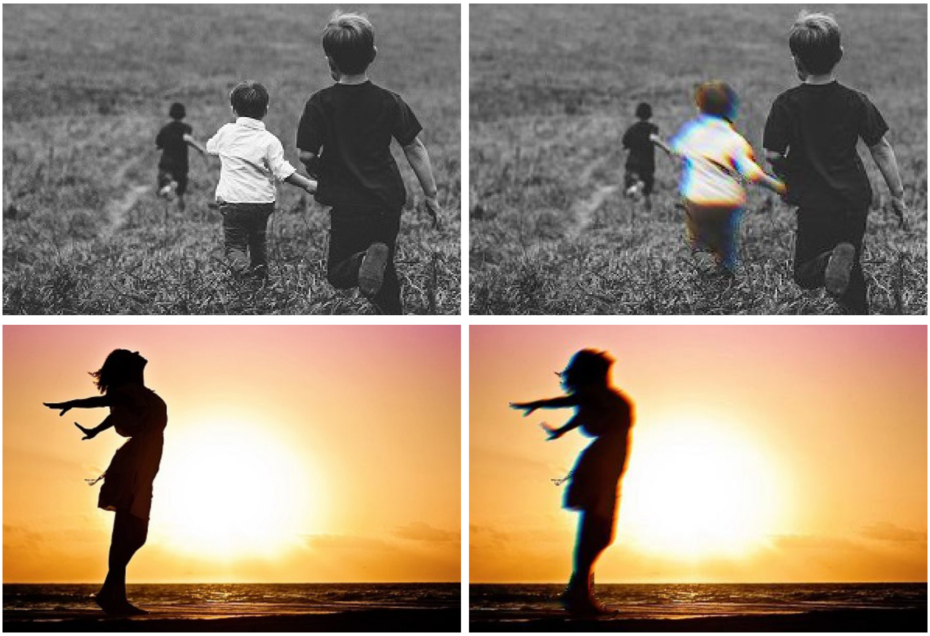


Figure 3.18: Harris Shutter motion blur results creating subtly colored motion and vintage effects. (Images source: pixabay.com)

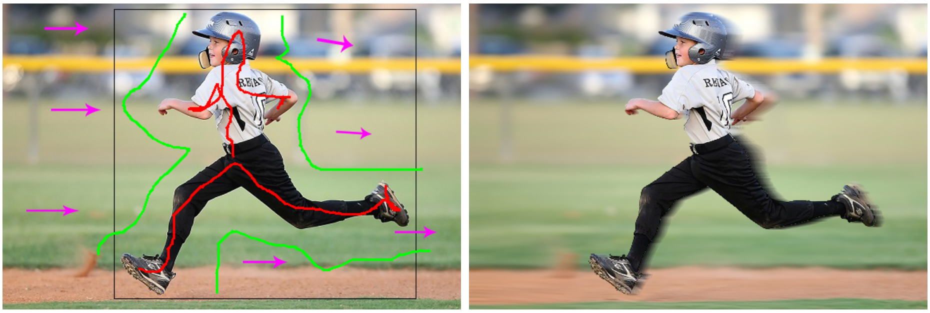


Figure 3.19: Motion trails simulating distinct movements while keeping the subject on focus. (Image source: pixabay.com)

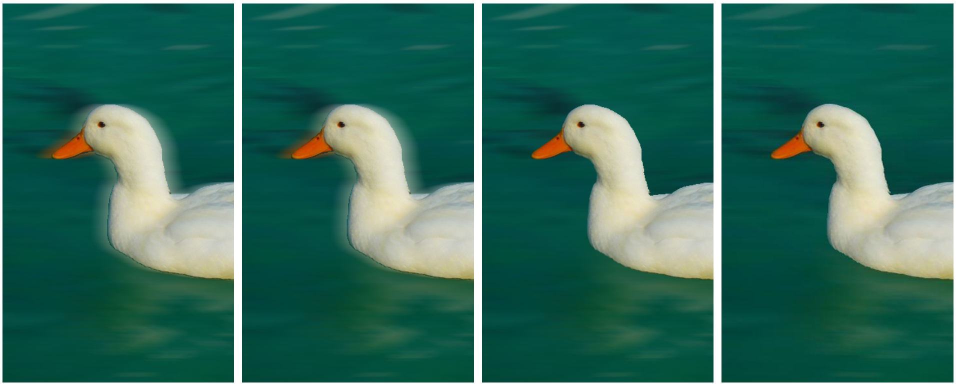


Figure 3.20: Comparison with Photoshop tools for the background motion blur. From left to right, Photoshop results using Motion Blur, Path Blur, Content Aware Fill with layer compositing, and our approach. (Image source: pixabay.com)

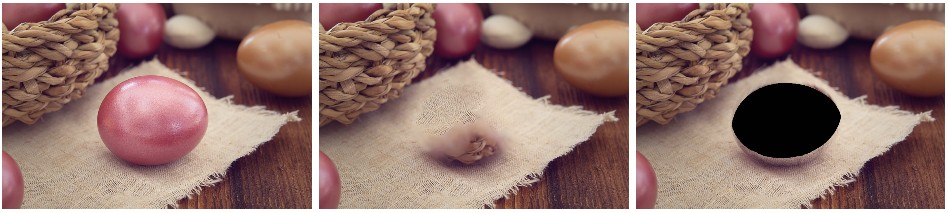


Figure 3.21: Inpainting results from Photoshop (center) and our solution (right) when a large object is removed. (Image source: pixabay.com)

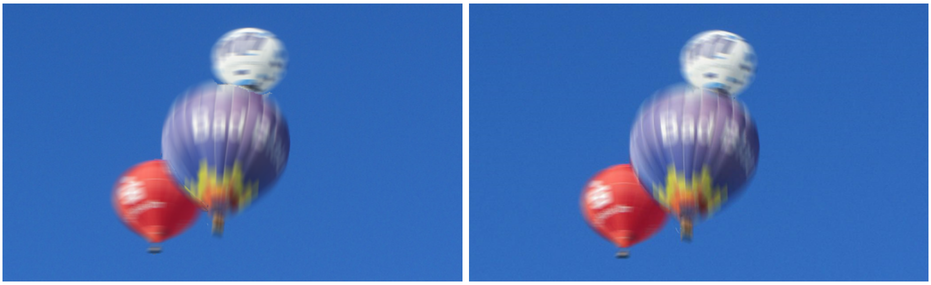


Figure 3.22: Motion blur with overlapping objects. Arbitrary depth assignment with artifacts (left) and our back-to-front ordering (right). (Image source: pixabay.com)

results (average: 6 minutes 13 seconds (our) vs. 15 minutes 25 seconds (PS)) and the users were more satisfied with their results (4.4 (our) vs. 3 (PS), on a Likert scale of 5 - higher being better). While giving an indication, this evaluation did not even cover all aspects of our solution. A more extensive study remains future work. The user-evaluation details are presented on Appendix A.1.

While our approach can produce convincing results, it also has its limitations. When moving objects overlap, a decision is needed to determine which element is to be considered in front, as illustrated in Fig. 3.22. However, our solution does not support object motion that would lead to several encounters of two objects changing their respective order.

Very strong motion is problematic because the area behind the moving object can become entirely visible (in the limit, the moving object would be completely transparent). A second problem can also occur when a moving object is initially partially covered. If the object has a shape that is easy to estimate for an observer, a discrepancy might arise. Fig. 3.23 illustrates such case. The head of the soccer player (left), for example, was enlarged in the inpainting, which darkens the motion trail. However, extreme motion blur is rarely attractive and usually not employed by a user. For most other cases, our inpainting is sufficient, as evidenced by the examples in this paper.

Finally, a challenge when applying motion blur is to deal with transparent and reflective surfaces. Fortunately, a human observer is typically not very strong in interpreting physical effects correctly and with ease. Regarding reflections, if the blur of the original object and its reflected counterpart do not perfectly match, the illusion might still be sufficient. To facilitate adding plausible reflections, we use also created a simple extension to our interface that allows a user to scribble a mirror axis, which is used to copy the annotations from one side of the reflection to the other when using uniform motion blur. An example is shown in Fig. 3.24.

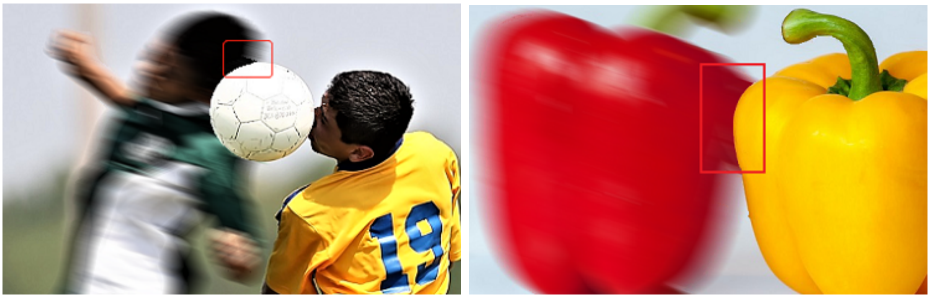


Figure 3.23: Inpainting artifacts with strong motion and small occluded areas. (Images source: pixabay.com)

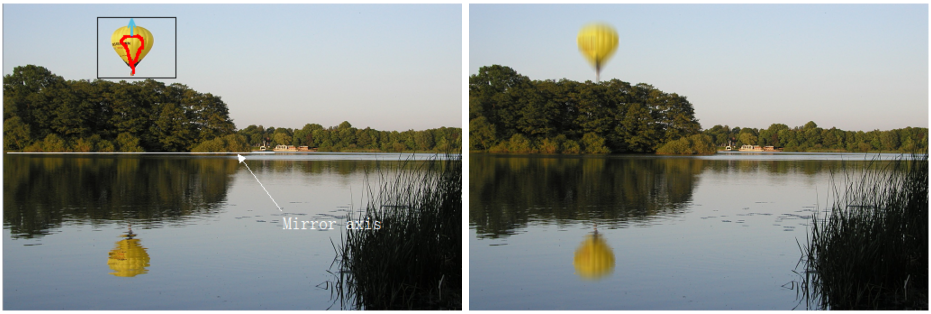


Figure 3.24: Motion applied to target object and its reflection. (Image source: pixabay.com)

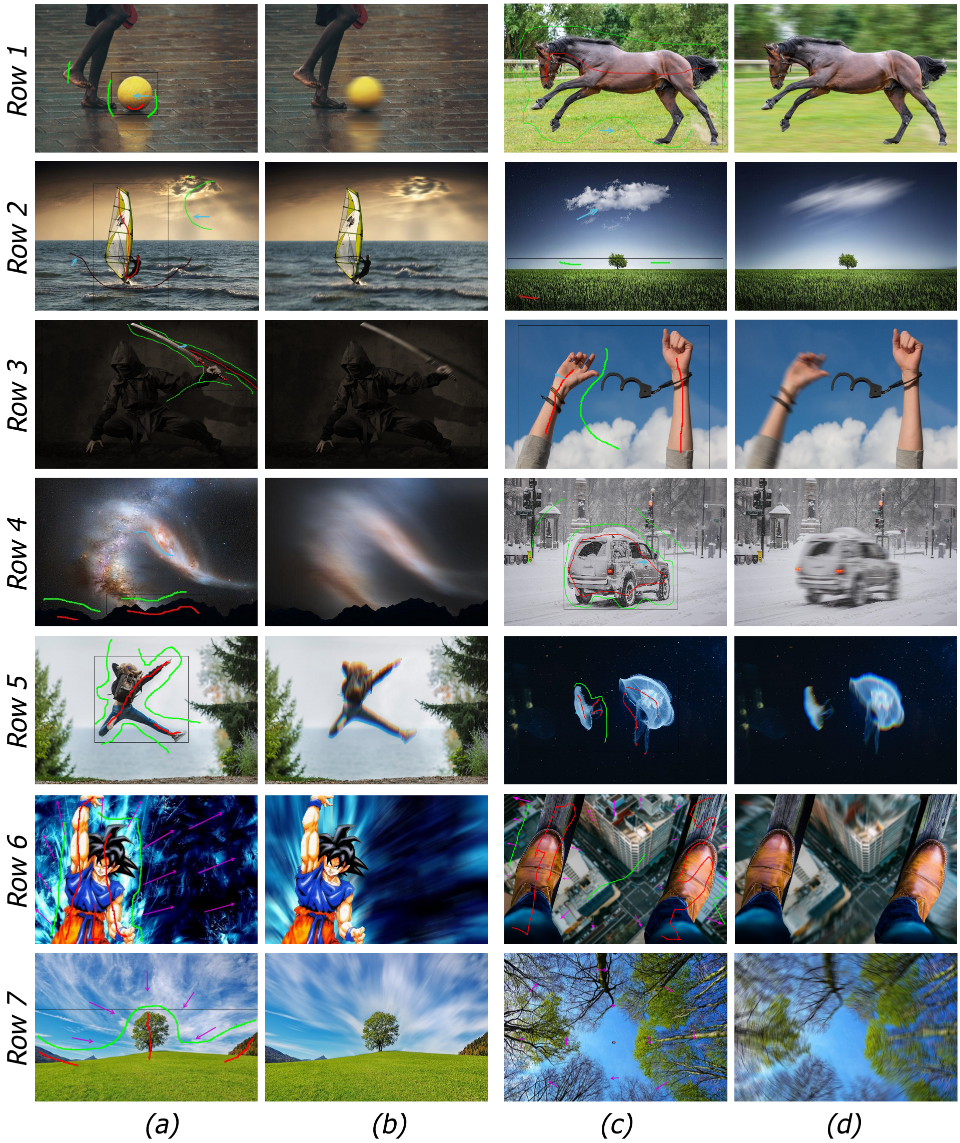


Figure 3.25: The diversity of scenes exploit by our framework. Blue and purple arrows indicate, respectively, the motion path of uniform and multi-directional motion blur. (Images from: unsplash.com, pexels.com, pixabay.com and easylife-online.com)

3.6. Conclusion

We presented a solution to add motion-blur effects to a single image in a post-process. It allows for a simple user interaction and requires only little user effort. Despite the method's simplicity, convincing results can be obtained in seconds and the outcome is easier to control than with a real-world capture. We illustrate that our solution provides support for complex motion paths and is able to reproduce several motion-blur-related photographic effects. Our algorithm can be applied to any image and does not require a specialized acquisition routine, which eases its use and increases its applicability.

References

[1] X. Luo, N. Z. Salamon, and E. Eisemann, *Controllable motion-blur effects in still images,* IEEE Transactions on Visualization and Computer Graphics (TVCG) (2018).

[2] X. Luo, N. Z. Salamon, and E. Eisemann, *Adding motion blur to still images,* in *Proceedings of Graphics Interface (GI)* (2018) pp. 108–114.

[3] F. Navarro, F. J. Seron, and D. Gutierrez, *Motion blur rendering: State of the art,* Computer Graphics Forum **30** (2011).

[4] B. Peterson, *Understanding Shutter Speed: Creative Action and Low-Light Photography Beyond 1/125 Second* (Amphoto Books, 2008).

[5] R. T. Held, E. A. Cooper, J. F. O'brien, and M. S. Banks, *Using blur to affect perceived distance and size,* ACM Trans. Graph. (TOG) **29** (2010).

[6] Lytro Camera, https://www.lytro.com/ (2017), accessed: 2017-10-20.

[7] H. Nagahara, S. Kuthirummal, C. Zhou, and S. K. Nayar, *Flexible depth of field photography,* in *European Conference on Computer Vision* (Springer, 2008) pp. 60–73.

[8] M. J. Cieslak, K. A. Gamage, and R. Glover, *Coded-aperture imaging systems: Past, present and future development–a review,* Radiation Measurements **92**, 59 (2016).

[9] J. T. Barron, A. Adams, Y. Shih, and C. Hernandez, *Fast bilateral-space stereo for synthetic defocus,* in *Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[10] SynthCam, http://bit.ly/2FQy4LZ (2011), accessed: 2017-12-09.

[11] C. MacManus, *The technology behind sony alpha dslrs steadyshot inside,* http://bit.ly/2CWbMDw (2009), accessed: 2017-12-20.

[12] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, and J. Davis, *Synthetic shutter speed imaging,* Computer Graphics Forum **26**, 591 (2007).

[13] RealSmart Motion Blur, *Realsmart motion blur,* http://revisionfx.com/products/rsmb/ (2017).

[14] N. Z. Salamon, M. Lancelle, and E. Eisemann, *Computational light painting using a virtual exposure,* Computer Graphics Forum **36** (2017).

[15] S. Liu, J. Wang, S. Cho, and P. Tan, *Trackcam: 3d-aware tracking shots from consumer video.* ACM Trans. Graph. (TOG) **33** (2014).

[16] GNU Image Manipulation Program (GIMP), https://www.gimp.org/ (2017), accessed: 2017-12-15.

[17] Adobe Photoshop, https://adobe.ly/1g8lSDp (2017), accessed: 2017-12-15.

[18] J. Schmid, R. W. Sumner, H. Bowles, and M. H. Gross, *Programmable motion effects,* ACM Trans. Graph. (TOG) **29** (2010).

[19] F. Navarro, S. Castillo, F. J. Seron, and D. Gutierrez, *Perceptual considerations for motion blur rendering,* ACM Trans. on Applied Perception (TAP) **8**, 20 (2011).

[20] G. Rosado, *Motion blur as a post-processing effect,* in *GPU Gems 3*, edited by H. Nguyen (Addison-Wesley, 2008) pp. 575–581.

[21] M. McGuire, P. Hennessy, M. Bukowski, and B. Osman, *A reconstruction filter for plausible motion blur,* Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (2012).

[22] J.-P. Guertin, M. McGuire, and D. Nowrouzezahrai, *A fast and stable feature-aware motion blur filter,* in *Proc. of High Performance Graphics*, HPG '14 (2014) pp. 51–60.

[23] J. Jimenez, *ACM Siggraph courses: Advances in real-time rendering - next generation post processing in call of duty,* (2014).

[24] T. Saito and T. Takahashi, *Comprehensible rendering of 3-d shapes,* in *Proc. of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90 (1990) pp. 197–206.

[25] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,* IEEE Trans. on Pattern Analysis and Machine Intelligence **40** (2018).

[26] A. Romero, M. Drozdzal, A. Erraqabi, S. Jegou, and Y. Bengio, *Image segmentation by iterative inference from conditional score estimation,* arXiv preprint arXiv:1705.07450 (2017).

[27] G. Lin, A. Milan, C. Shen, and I. D. Reid, *Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,* in *Computer Vision and Pattern Recognition (CVPR)* (2017).

[28] C. Rother, V. Kolmogorov, and A. Blake, *Grabcut: Interactive foreground extraction using iterated graph cuts,* ACM Trans. Graph. (TOG) **23**, 309 (2004).

[29] R. Laganiere, *OpenCV 3 Computer Vision Application Programming Cookbook* (Packt Publishing Ltd, 2017).

[30] Y. Wexler, E. Shechtman, and M. Irani, *Space-time completion of video,* IEEE Trans. on Pattern Analysis and Machine Intelligence **29** (2007).

[31] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, *Semantic image inpainting with deep generative models,* in *Computer Vision and Pattern Recognition (CVPR)* (2017).

[32] C. Barnes and F.-L. Zhang, *A survey of the state-of-the-art in patch-based synthesis,* Computational Visual Media **3**, 3 (2017).

[33] A. Orzan, A. Bousseau, H. Winnemoller, P. Barla, J. Thollot, and D. Salesin, *Diffusion curves: A vector representation for smooth-shaded images,* ACM Trans. Graph. (TOG) **27** (2008).

[34] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot, *Diffusion constraints for vector graphics,* in *Proc. of the 8th International Symposium on Non-photorealistic Animation and Rendering* (2010).

[35] S. Jeschke, D. Cline, and P. Wonka, *A gpu laplacian solver for diffusion curves and poisson image editing,* Transaction on Graphics (Siggraph Asia 2009) **28** (2009).

[36] J. J. van Wijk, *Image based flow visualization,* ACM Trans. Graph. (TOG) **21** (2002).

[37] B. Masia, S. Agustin, R. W. Fleming, O. Sorkine, and D. Gutierrez, *Evaluation of reverse tone mapping through varying exposure conditions,* ACM Trans. Graph. (TOG) **28** (2009).

[38] D. D. Busch, *Nikon D200 Digital Field Guide* (John Wiley and Sons, 2006).

[39] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, *Patchmatch: A randomized correspondence algorithm for structural image editing,* ACM Trans. Graph. (TOG) **28** (2009).

4

Spatio-temporal Exposure Control for Videos

N. Z. Salamon, M. Billeter and E. Eisemann

Another time, another place;
A hollow universe in space.

A camera's shutter controls the incoming light that is reaching the camera sensor. Different shutters lead to wildly different results, and are often used as a tool in movies for artistic purpose, e.g., they can indirectly control the effect of motion blur. However, a physical camera is limited to a single shutter setting at any given moment. ShutterApp enables users to define spatio-temporally-varying virtual shutters that go beyond the options available in real-world camera systems. A user provides a sparse set of annotations that define shutter functions at selected locations in key frames. From this input, our solution defines shutter functions for each pixel of the video sequence using a suitable interpolation technique, which are then employed to derive the output video. Our solution performs in real time on commodity hardware. Hereby, users can explore different options interactively, leading to a new level of expressiveness without having to rely on specialized hardware or laborious editing.

This chapter has been published on Computer Graphics Forum, **38**, 7 (2019) [1].

4.1. Introduction

In photography, the shutter controls when incoming light reaches the image sensor. Together with sensor sensitivity (ISO) and lens opening (aperture), the shutter speed defines the image exposure. In the first cameras, the shutter was a simple mechanic device manually moved in front of the lenses. Later, shutter devices with different shapes were created, from rotary-discs to blinds and diaphragms. Nowadays, most digital cameras implement an electronic shutter, which simply blocks or lets photons pass to an active sensor element.

In cinematography, many directors still use rotary-disc shutter devices for creative choices, typically turning synchronously to the 1/24th of a second frame time. The exposure of a frame is then controlled by an angular cut on the rotary-disc, while its spinning speed controls the video framerate. The ratio between the open and closed angles influences how sharp or smooth the scene motion is registered. A common practice is to use a 180° angle cutout, which is typically perceived as a natural motion blur by the audience [2]. A faster shutter (smaller open angle) leads to crisp content and sharp motion. A good example is the movie *Saving Private Ryan*, which uses a 45° shutter to convey a frightening ambiance [3]. Smaller angles mimic the effect of newsreels. Slower shutters (large open angle) result in motion blur, often used to give a sense of fast motion. Longer exposure can also smooth perceived motion, as used by David O. Russell's in *The Fighter* to subtly correct jittery movements [4]. Finally, ramping shutter speeds contributed to the energetic atmosphere of *Mad Max: Fury Road* [5].

While controlling the shutter has been established as a useful measure to influence the perception of a video sequence and to enable different visual styles, the available options are rather limited. Even mixing different motion patterns on the same take requires multiple cameras and compositing techniques. Our solution addresses this topic and we provide a novel technique to virtually simulate and combine different shutters in space and time. Unlike previous work, we let artists define spatio-temporally varying virtual shutters in a post-process using a simple user interface.

All results are presented in real time, which supports the shutter design process. A key advantage of our solution is that only sparse spatial and temporal annotations are needed, which then define varying per-pixel shutter functions for the entire video sequence. We demonstrate the reproduction of common real-world shutters and illustrate various options for artistic choices. In this context, we made the following technical contributions:

- an efficient shutter interpolation procedure;
- a technique to extend sparse shutter definitions; and
- a real-time interface for shutter design and compositing.

4.2. Related Work

The impact of shutter functions depends greatly on the motion in the captured scene, which can be spatially varying. When combining sharp and blurred objects in the scene, it is natural to consider matting to define target areas and to composite shots from different cameras. While recent approaches (e.g. [6–8]) can segment the masks, the process is still costly and challenging, as motion blur trails will jut out of the masks. Because of this, a single-frame motion blur requires inpainting to fill in the partially visible background underneath the motion trails [9]. Consequently, such spatial mixing is difficult to obtain for real-world footage.

Glassner [10] investigated the behavior of different shutter shapes on synthetic scenes, including a virtually-simulated "Slit Scan" shutter used in the iconic stargate scene from Stanley Kubrick's *2001: A Space Odyssey* [11]. The slit scan effect is analogous to nowadays electronic camera sensor: the image is composed by partial sensor scans. The scan occurs line by line and in fractions of a second, i.e., while the sensor is exposed to light at a given shutter speed. If the speed is not as fast as the moving objects in the scene, the *rolling-shutter* effect is observed. While such effect is occasionally applied for artistic reasons, for many applications it is not desirable and spatial and angular warpings have been proposed to correct for image distortions [12].

When the scene has little motion, the effects of varying shutters can be subtle, and their perception varies with different viewers' preferences and expertise [13]. Nevertheless, it has been shown that spatially varying exposure times can influence gaze motion [14]. Further, extreme examples, such as a stop-motion look, as seen in Cooper and Schoedsack's *King Kong*, is perceived as very unnatural. The effect is due to the complete lack of motion blur, as the ape model was recorded via still imagery. Interestingly, Brostow and Essa [15] proposed a solution to simulate fast shutter speeds and create a stop-motion look from blurred videos.

Post-processed motion blur simulates a long exposure shutter by accumulating video frames. Such frame aggregation has been explored by tracking image features to backproject the motion onto a single background image [16] or by stabilizing the video on the focus object and averaging the moving pixels [17]. In both cases, the blur effect is created from camera motion. Other approaches use optical flow and/or 3D motion vectors to globally track and create a per-pixel motion blur [18]. Spatially-varying blur requires masking and layer compositing via external software [18] or soft brushes [17]. Nonetheless, the temporal motion blur strength is constant after each keyframe. Our method not only supports frame aggregation for motion blur, but uses a general shutter function description that allows us to achieve many more effects (including rolling shutters, stuttered motion, or ghosting). In addition, we allow shutter functions to change both over space and time, and ensure seamless interpolation of these.

When processing high frame rate videos, another prominent context of shutter design is temporal filtering. Downsampling can be used for display on conventional screens [19] or to adapt framerate according to image content [20]. Nonetheless, the process has to be carefully applied, as framerate was shown to be more appreciated than resolution under budget constraints [21]. Disney's short movie Lucid Dreams of Gabriel [22] experiments with different spatio-temporal expressive effects to enhance storytelling. Our solution enables a wealth of temporal manipulations via its shutter-design interface.

4.3. Our Approach

For our algorithm, we assume an input video that is *fully exposed*, meaning that the shutter is open during the whole frame time and no delay is induced from one frame to the next. While such input is not standard, modern devices and computational approaches enable the construction of such a sequence relatively easily. For standard videos, we prepare in-between frames using optical flow [23]. Unlike the remaining steps of our method, this preparation is an offline pre-processing step and we discuss the details in Appendix B.1.

Given the input video, users can interact in real time to add simple shutter annotations in form of scribbles, defining regions that will be using the according shutter. Shutter definitions are interpolated over time and space using a diffusion mechanism, creating a shutter function for each pixel of the video. See Fig. 4.1 for an illustration of the full pipeline.

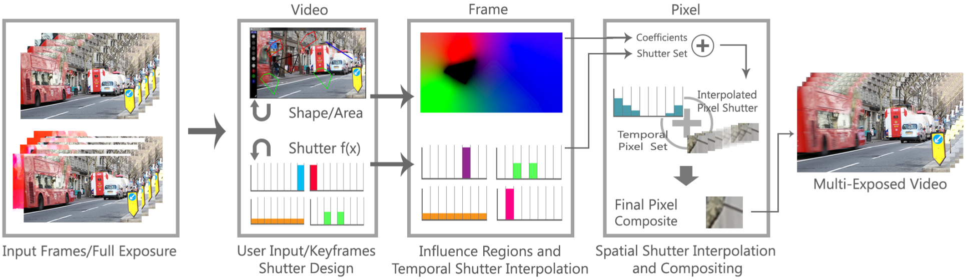

Video
Frame
Pixel
Shape/Area
Shutter f(x)
Coefficients
Shutter Set
Interpolated Pixel Shutter
Temporal Pixel Set
Final Pixel Composite
Multi-Exposed Video
Input Frames/Full Exposure
User Input/Keyframes Shutter Design
Influence Regions and Temporal Shutter Interpolation
Spatial Shutter Interpolation and Compositing

Figure 4.1: Our system accepts an input video. If this video is not already fully exposed, the full exposure is created in a pre-processing step. The user can then design custom shutter functions, create shutter areas and define keyframes for temporal changes. The resulting multi-exposed video is immediately visible, as the back-end pipeline runs in real time.

In the following, we first explain the mathematical model to simulate and interpolate shutter functions (Sec. 4.3.1 and Sec. 4.3.2). We then discuss the implemented user interface (Sec. 4.3.3) and how to provide sparse input to create shutter definitions for the entire video sequence. The latter is achieved via a diffusion process and an interpolation of the diffused shutter information. The efficiency of these two important steps is crucial for achieving real-time feedback and the details are described in Sec. 4.3.4.

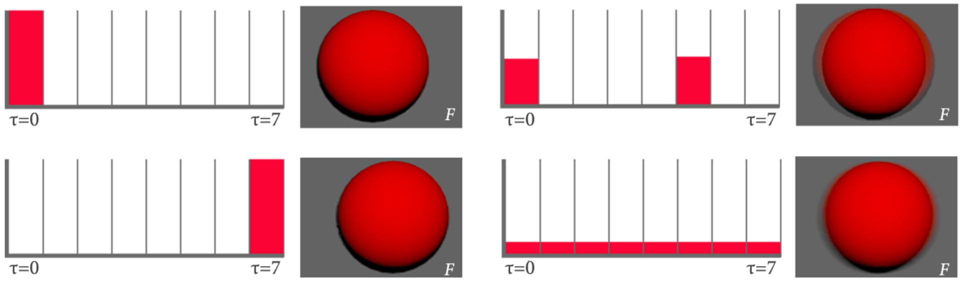


Figure 4.2: Different shutter functions yield distinct results on the same input video of a horizontally moving sphere. Left: A shutter function equal to $\delta_{0\tau}$ reproduces the original video (top). By changing the location of the delta function, the video is time shifted (bottom). Right: a simulated shutter device with two cuts (top) results in a compositing of two frames, while a constant exposure (bottom) creates a motion-blurred result.

4.3.1. Image formation with a Virtual Shutter

The traditional camera image-formation model defines a frame as the integration of the incoming light over time while the shutter is open. For a virtual shutter defined on a video with a fixed frame rate, time is discretized. The shutter function becomes a set of weights, one for each discrete quantum of time.

Formally, the input is a fully-exposed video V consisting of $T \in \mathbb{N}$ frames, such that $V\,(t)$, with $t < T \in \mathbb{N}_0$, is the t^{th} frame. For a frame t, a *shutter function* s_t acts as a filter and attributes weights to $\mathrm{T}+1$ frames of the input video from frame t onwards, leading to a *filtered frame* $F\,(t)$:

$$F\,(t) = \sum_{\tau=0}^{\mathrm{T}} s_t\,(\tau)\; V\,(t+\tau)\,.$$

The simple case of reproducing the input video $O\,(t) = V\,(t)$ would define $s_t\,(\tau) = \delta_{0\tau}$, where δ_{ij} is the Kronecker delta (one if both indices match, otherwise zero). Fig.4.2 illustrates the result of different shutter functions for the same video input. As the frame rate of an output video O does not have to match the frame rate of the input video V, a monotonic *frame-mapping function* $M : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ can be used to define the i^{th} output frame $O(i) = F(M(i))$. The video O can then be played using the suitable frame time.

4.3.2. Shutter-function Interpolation

In our approach, shutter definitions will be specified on a per-pixel basis, thus enabling different shutters in different frame locations and at different times. To transition between the shutter definitions, we need to provide an interpolation among them. A trivial choice would be a linear value interpolation. Unfortunately, this solution does not result in a meaningful outcome.

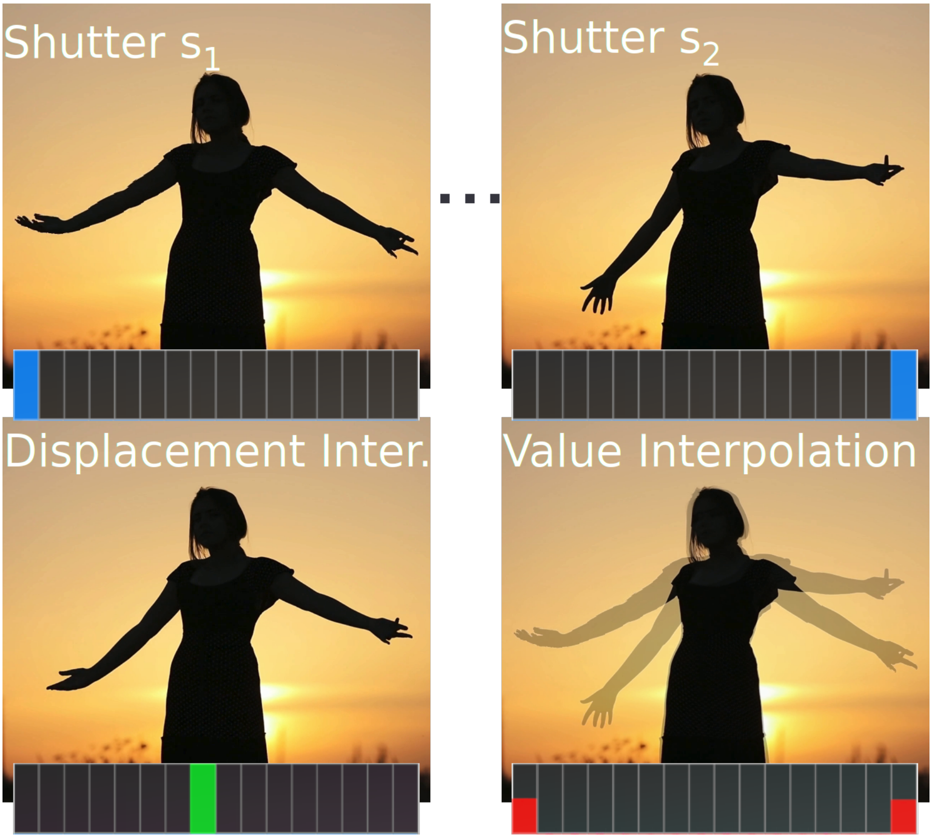


Figure 4.3: Comparison of displacement and value interpolation. Interpolating the input shutter functions (displayed in the top row) at roughly 50-50 results in the images in the bottom row, depending on the interpolation method. Displacement interpolation (bottom left) smoothly shifts time across the transition, resulting in a sharp output image constructed from the in-between frames. Value interpolation of the shutters results in a shutter function that just mixes the input images (bottom right). Video source: pixabay.com.

Consider the shutter functions $s_1(\tau) = \delta_{0\tau}$ and $s_2(\tau) = \delta_{(15)\tau}$ (see Fig. 4.3, top), which means that s_2 results in a frame that occurs 15 frames after the frame produced by s_1. The linearly interpolated shutter function (Fig. 4.3, bottom right) would just blend both frames. For a natural transition, one would expect that intermediate frames between both time steps are obtained (Fig. 4.3, bottom left). This can be achieved via displacement interpolation [24].

Displacement interpolation is typically applied to probability distribution functions and can be done by value interpolation of their inverse cumulative distribution functions [25]. We will present our efficient implementation in Sec. 4.3.4.

4.3.3. Interface

In order to define and apply different shutter functions to an input video, we need to indicate their regions of influence, both spatially and temporally. Our application provides a graphical interface (Fig. 4.4) to support users in this task.

To apply shutter effects to the video, the first step is to define a default shutter function that will be applied to all frames, unless additional shutters are defined. To this extent, the user first chooses the number of shutter weight entries $\mathtt{T}$, which triggers a graph-bar editor. Here, the individual shutter-function values are set by dragging the corresponding bars up and down, representing values between zero or one. Alternatively, the user can choose among shutter functions from a predefined library.

To create an additional shutter, the user draws a scribble on a wanted frame. Again, the user defines a corresponding shutter function. The new shutter is then active for this frame and applied to the area covered by the scribble. To use the shutter over several frames, the user advances in the video and can place key frames. A keyframe enables the user to redefine the scribble (position, size, orientation) or the shutter function. For all intermediate frames between keyframes, the shutter is interpolated (meaning its shutter function, as well as the defining scribbles attributes). Once the shutter definition is complete, the procedure can be repeated.

For now, the shutter definitions would only be applied directly to the pixels underneath the scribble annotation. Instead, we actually want to extend the shutter definitions to the entire frame. To this extent, we first collect all active shutters at time τ and derive their interpolated representation from the user defined keyframes. Using their scribble annotations, a diffusion process is executed to find shutter interpolation weights for all pixels in the frame. We express this diffusion process as the solution to a heat transport problem, where the shutter annotations are used as local constraints, similar to the work of Orzan et al. [26]. In each pixel, the resulting interpolation weights define a shutter function that is used to process the input video. We rely on an efficient implementation, which we detail in Sec. 4.3.4, such that all steps can be executed in real time. Hereby, the user has immediate feedback and can then adjust, delete or add new shutters until the desired result is obtained.

An illustration of the interface is shown in Fig. 4.4, which contains the graph-bar edit of a shutter function, as well as scribble annotations for several shutters. Here, a shutter with a longer exposure time is applied to the wood block that is moving towards the characters, where a short exposure is used.

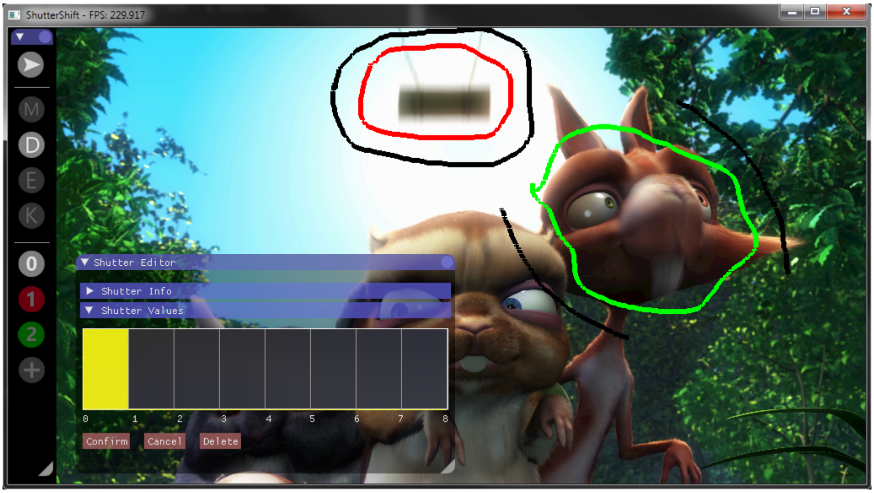


Figure 4.4: Interface for interactively defining shutter functions and their influence regions. Users design shutter functions using the bar graph editor, whilst additional details, such as the number of frames in the function, can be changed using one of the collapsed fold-outs. Influence regions are defined by drawing scribbles using the corresponding color. The results are immediately visible. The sequence is from the *Big Buck Bunny* movie by the Blender Institute [27].

4.3.4. Efficient Implementation

To achieve real-time performance with high accuracy, our solution relies on suitable algorithmic choices and an efficient GPU implementation. The two main performance bottlenecks are the interpolation of shutter functions and the diffusion of the shutter annotations. While diffusion accelerations exist [28, 29], we opted for an alternative solver that is easy to implement, does not require geometric or curve primitives, and extends to more complex diffusion annotations [30] without sacrificing quality.

Efficient Shutter-Function Interpolation

To describe the interpolation procedure, given N shutter functions s_i, we define the total exposure $e(s_i) = \sum_{k=0}^{\mathrm{T}} s_i\,(k)$ and the normalized accumulation, denoted by a capital letter: $S_i\,(\tau) = \sum_{k=0}^{\tau} s_i(k)/e(s_i)$.

Given the interpolation coefficients c_i for shutter s_i ($\sum c_i = 1$ and $c_i \geq 0$), we wish to find the interpolated shutter q. We make use of the observation that $q(\tau) = (Q(\tau+1) - Q(\tau))e(q)$, where $e(q) = \sum_{i=1}^{N} c_i e(s_i)$. Hence, having Q allows us to find q. Q is indirectly defined via its inverse, which is, in turn, given by a linear combination of the inverses of the accumulated shutter functions [25]: $Q^{-1} = \sum_{i=1}^{N} c_i\, S_i^{-1}$. This relationship provides a solution to determine q (Fig. 4.5): first compute all S_i, invert them to derive Q^{-1}, then invert this function to find Q and use it to determine q.

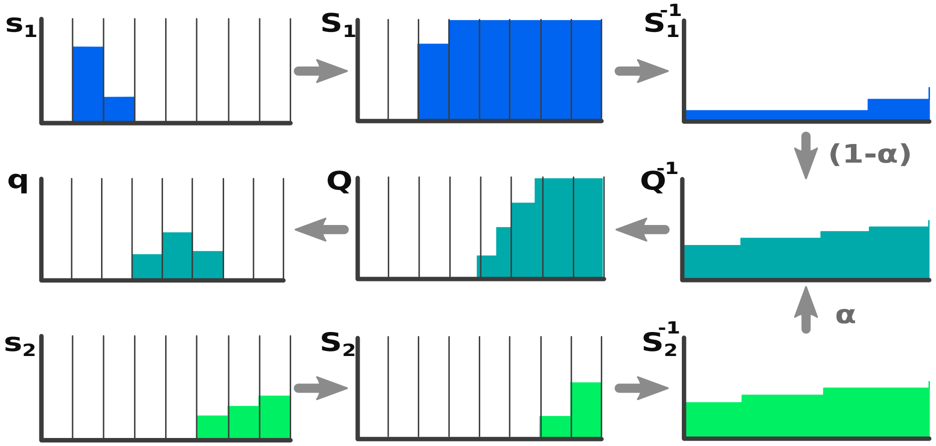


Figure 4.5: The process of interpolating shutter functions s_1 and s_2 to produce a new shutter function q requires several steps. First, the respective accumulation functions are derived (S_1, S_2). Next, these functions are inverted and then interpolated. The resulting function represents Q^{-1}, which is inverted so q can be obtained by deriving and discretizing the representation.

The functions Q and Q^{-1} depend on the per-pixel and per-frame coefficients c_i, requiring an efficient method for evaluating these functions in real time. In the following, we describe our approach. To simplify, but without loss of generality, we will assume that for all shutter functions $e(s_i) = 1$.

We consecutively determine the value for $q(\tau)$, with $\tau = 0 \dots \mathrm{T}$, retrieve each time the corresponding frame pixel, apply the weight and accumulate the result. In this way, the full function q does not need to be stored in memory. As $q(\tau) = Q(\tau + 1) - Q(\tau)$, we can instead solve for $Q(\tau)$, with $\tau = 0 \dots \mathrm{T}$ and only need the previous and current value of Q in memory. For now, we will assume that we have access to Q^{-1}. If we localize z such that $\tau = Q^{-1}(z)$, we have $z = Q(\tau)$. As $Q < 1$, we can deduce that $z \in [0, 1]$ and as Q^{-1} is monotonically increasing, we can employ a bisection method to solve for z. By default, we use a fixed number of nine search iterations on this interval, which, due to the interval search, yields an error of at most 2^{-10}. This error is sufficiently small, as the precision loss of an 8 bit video is magnitudes larger. However, the iterations can be adapted for higher dynamic range content.

To complete the calculation of the values of q, we still need to be able to compute $Q^{-1}(z)$ during the bisection method. We recall $Q^{-1} = \sum_{i=1}^{N} c_i\, S_i^{-1}$, which thus means that we need to invert S_i. We can again solve an equation of the form $z = S_i(\tau)$. Instead of a bisection, a binary search over discrete elements is more suitable, since S_i is efficiently represented by an array of T elements.

For numerical robustness, we consider the function S_i to be piecewise linear. In fact, this reflects that the values of S_i stem from an integration of constant exposure over the frame time. Based on this, we first search for the first element $w + 1$, such that $S(w+1) > z$. The value $S(w)$ is guaranteed to be smaller or equal to z. We then compute the refined result $\tau = w + (z - S(w))/(S(w+1) - S(w))$, linearly interpolating w and $w+1$ based on z.

To reduce the average cost of the searches, we successively shrink the search space. In each iteration k of the bisection method, a z_k will be updated to a new location z_{k+1}. Assume in iteration k, we found the set of $w_{k,i}$ in the binary searches. Due to monotonicity, if $z_{k+1} \geq z_k$, then each $w_{k+1,i} \geq w_{k,i}$ and if $z_{k+1} < z_k$, then $w_{k+1,i} \leq w_{k,i}$. Hereby, the search space for the next $w_{k+1,i}$ shrinks.

A similar optimization can be applied when evaluating consecutive values of Q. We can restrict the lower bound of $w_{0,i}$ based on the previous values after convergence because $Q(\tau) \leq Q(\tau+1)$. In principle, a last option exists to shrink the interval during the bisection but this case turned out to be inefficient because we already apply the method with only a few steps. A pseudo-code of our efficient shutter interpolation implementation is presented in Appendix B.2.

Efficient Shutter Diffusion

To extend the shutter annotation scribbles to the entire frame, we rely on a diffusion process. Similar to Orzan et al. [26], we express the diffusion as a heat transport problem, where user annotations are local constraints. We follow their approach and rely on a multi-grid solver but perform a customized downsampling to achieve a high quality diffusion only from the pixel image, without resorting to geometric primitives.

Specifically, we use two images, a mask image identifying the constraint locations, and an image defining the values of the constraints at those locations. We reduce the problem size by consecutively halving the resolution until we reach a size of 2×2. For the smallest image, the solution can be solved immediately and it is then repeatedly upsampled and recombined with the constraints at the next resolution level, while applying a small number of diffusion steps (Jacobi iterations). This process is repeated until we reach the original image size.

To faithfully maintain the constraints during downsampling, we analyze each 2×2-pixel block before collapsing and actually store four values representing an approximation of the accumulated values that are emitted into the four axis-directions. We refer to the four values along the axis as an *influence block*. The first influence blocks are formed by 2×2-pixel groups (Fig. 4.6, left). Starting from an edge of this block, if both adjacent pixels are filled, they both contribute evenly and we average their values. If one pixel is empty but the next behind is filled, the closer one contributes with a weight of one, the farther one with a weight of 0.5. If there are no constraints in the nearby pixels but both farther pixels contain constraints, then they contribute evenly. If there is only one pixel, it contributes alone. If the block contains no constraints, it does not emit anything.

In the following steps, 2×2 groups of influence blocks are combined safely by

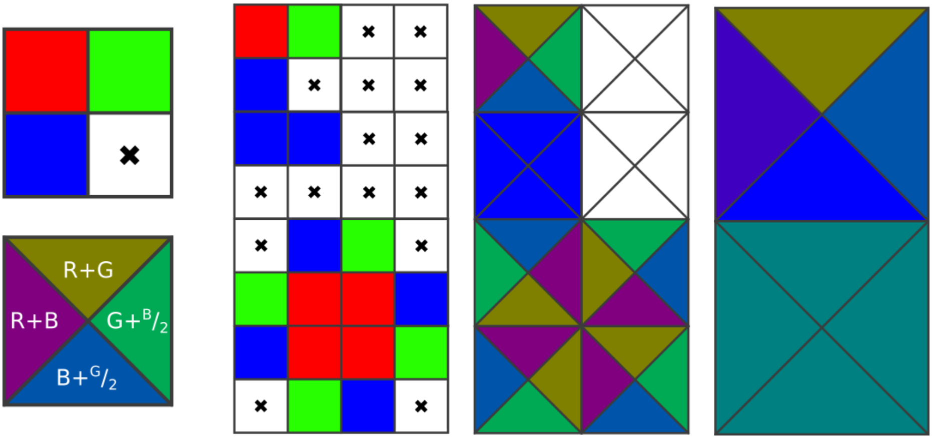


Figure 4.6: On the left, we show the diffusion influences arising from the 2×2 pixel block containing a red, a green and a blue constraint as well as one constraint-free location. On the right, we illustrate the downsampling of influence regions. In particular, the four fully-enclosed red pixels do not affect the outside at all, and their contribution disappears after downsampling once.

maintaining their outward influences and discarding interior ones (Fig. 4.6, right). We employ the same weighting scheme as in the initial step. However, instead of relying on the single color value at each location in the block, we fetch the location's influence in the currently-considered direction.

During the diffusion process, we naively upsample without any filtering, replicating pixel values across whole blocks. Upsampling with linear filtering would introduce additional complexity due to having to consider surrounding constraints. We are able to limit the number of Jacobi iterations performed at each level significantly: we perform six iterations for sizes up to 64×64, two iterations for higher resolutions. The Jacobi iterations must consider both constraint influences (if present) and the diffused values. Intermediate images storing diffused values use 16 bits of precision per channel, as an 8 bit color depth results in small gradients vanishing very early. Fig. 4.7 illustrates a set of user constraints along with diffused results. We compute the results for up to four color channels (RGBA) simultaneously; constraint locations are stored in a separate binary map. Note that constraints in close proximity remain properly separated without bleeding into each other.

4.4. Results

We implemented the described system as a desktop software that enables users real-time editing and exploration of different shutter functions in various video clips. All our experiments were performed on a standard desktop system running Windows 7, with a Intel i7 3820 CPU, 16GB of RAM, and a NVIDIA GTX 1080 GPU with

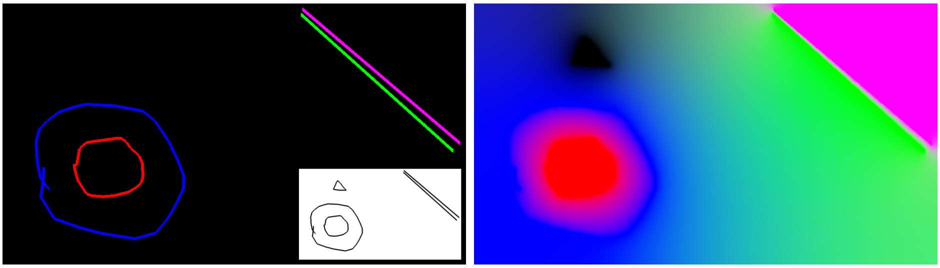


Figure 4.7: The input image (left) shows the user-drawn scribbles that create the constraints. The mask (white inset at the bottom) defines the locations of the constraints. Note the zero-valued (=black) constraint in the top-left of the input. The result of the diffusion process (right) form the per-pixel coefficients for the shutter interpolation. Importantly, the red influence is fully contained by the outer blue constraint, and the green and pink regions stay well separated.

8GB of VRAM. Our input videos were acquired in-house (using a Samsung Galaxy S8, recording at 120-240Hz), unless a different source is noted in the image captions (typically recorded at 24-30Hz). While the usage of high frame rate videos is a desirable choice to improve the physical accuracy of the time integration, this is not a requirement. Hence, the input frame rates vary depending on the video source. The output frame rate is determined from the input video and user selections, such as the size of the shutter function.

We can achieve a number of different effects with our system. For example, an object or region can be highlighted by applying a short shutter, producing a sharp output, while the rest of the image can use a long shutter that results in motion blur. Fig. 4.8 shows an example where the movement of one hand is made less visible, to focus on the precise movements of the right hand. For these effects, the user draws annotations on the image around the areas of interest; the results are shown in real time.

The effect can be extended to vary over time. In the video sequences shown in Fig. 4.9, keyframes are used to change the shutter functions and areas over time. At different moments, a single moving individual is made sharp to produce a contrast against the background, which is abstracted using a long exposure. The sharp areas are further keyframed to follow the subjects. Before switching to a new person, the shutter function fades out to smoothly merge the previously highlighted person into the crowd, illustrating the advantage of our interpolation method. The effect can be reversed, and attention can be removed from a person instead.

In Fig. 4.10 we demonstrate a time-varying rolling shutter. As the car drives past, the direction of the rolling shutter switches, causing the skewing arising from the rolling shutter to reverse. We realize the rolling shutter in our framework by defining two shutter functions at the top and bottom of the image, selecting the first and last frames inside the shutter function's window, respectively. The spatial interpolation ensures that the shutter function shifts smoothly across the window

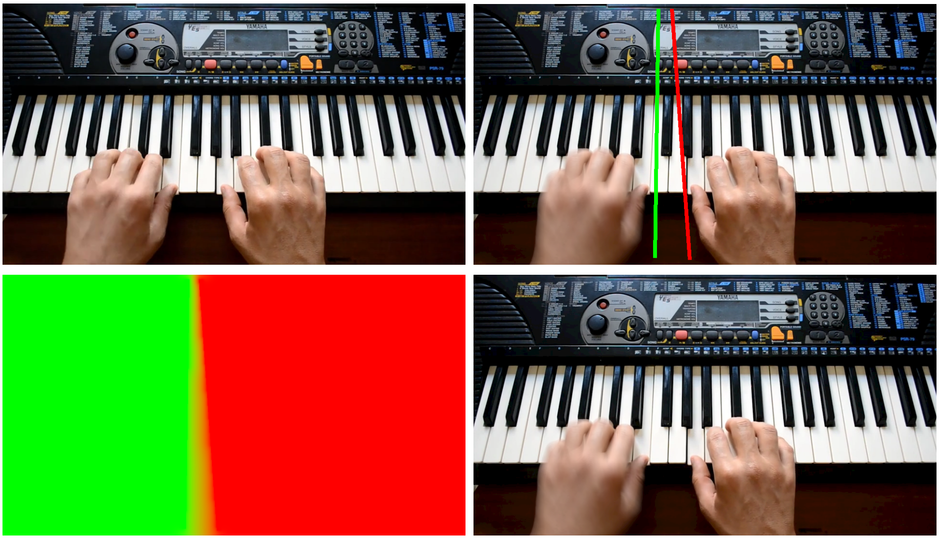


Figure 4.8: Top: Original frame (left) and user-drawn annotations (right). Bottom: Diffused per-pixel coefficients and the resulting frame after applying the shutter functions. Note the blur on the left hand contrasting with the sharp motion on the right side. Video source: pixabay.com at 25Hz. Rendering time (avg): 0.59ms in 960×540.

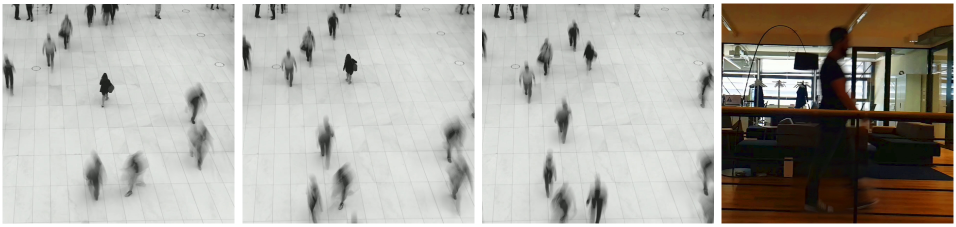


Figure 4.9: A motion blur shutter can be applied selectively to different regions of the image. Here, the keyframed regions track a person in the images. In the left images, we keep one moving person sharp over a short time span, while blurring the rest. This highlights and draws attention to the person. On the right, we do the opposite, and remove attention from the person crossing the hall. Left video source: pixabay.com at 25Hz. Rendering time (avg): 1.93ms in 960×540. Right image sequence captured in house at 240Hz. Rendering time (avg): 1.58ms in 1280×720.

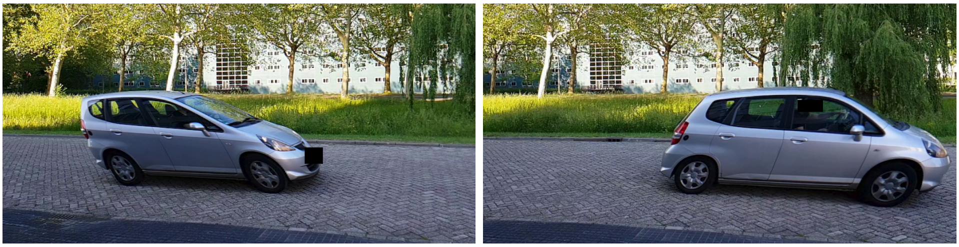


Figure 4.10: Rolling shutters are realized by defining two time-shifted shutter functions and interpolating between these. Shutters spans 64 frames. By keyframing the shutter functions, we can cause the effect to flip midway, reversing the skewing as the car drives past. Black squares added for anonymization. Video captured in house at 240Hz. Rendering time (avg): 2.24ms in 1280×720.

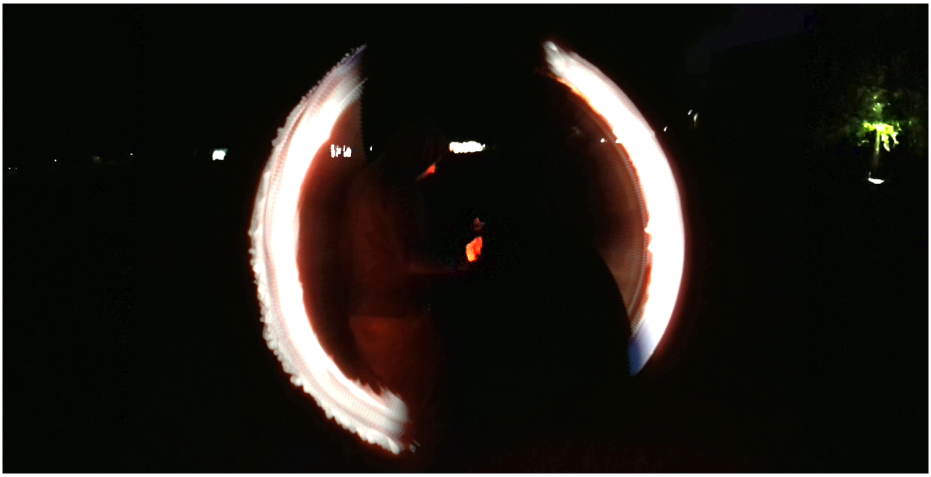


Figure 4.11: Light trails created with a long exposure shutter (spanning 0.5 seconds = 120 frames). The remainder of the background and the person holding the fire pole are kept sharp. Video captured in house at 240Hz. Rendering time (avg): 8.71ms in 1280×720.

size. Another artistic effect is illustrated in Fig. 4.11, where the light trails were composited to resemble a light painting while keeping the person sharp.

We list the per-frame total rendering time for all results in their respective figure captions. Total time measures wall-time, and thus includes overheads from other processing. For the shown results, we defined two shutter functions of varying size. The number of shapes and keyframes vary across examples.

To isolate performance on the two core steps of our approach (Diffusion and Interpolation), we specified a set of shapes (two lines and two lassos) and up to four shutters. The results are evaluated in 1920×1080 (full HD) and 3840×2160 (4K) resolution. Tables 4.1 and 4.2 present our results for different numbers of shutter functions (N) and different lengths (T) under *Ours (full resolution)*. We compare our method to a simplified interpolation implementation (see column *Naive Histogram*

			Naive Hist. Interp.		**Ours (full resolution)**		**Ours w/ aggreg. (2 × 2)**	
full HD	Diffusion		–		*0.88 ms*		*0.35 ms*	
	Other		–		*0.42 ms*		*0.39 ms*	
	Shutter Interp. & Comp.	$N \backslash T$	8	16	8	16	8	16
		2	*5.69 ms*	*12.85 ms*	*3.50 ms*	*6.88 ms*	*0.92 ms*	*1.79 ms*
		3	*8.05 ms*	*17.94 ms*	*4.91 ms*	*9.87 ms*	*1.35 ms*	*2.71 ms*
		4	*10.40 ms*	*23.20 ms*	*6.48 ms*	*12.95 ms*	*1.74 ms*	*3.46 ms*
4K	Diffusion		–		*3.04 ms*		*0.90 ms*	
	Other		–		*0.57 ms*		*0.44 ms*	
	Shutter Interp. & Comp.	$N \backslash T$	8	16	8	16	8	16
		2	*22.86 ms*	*51.01 ms*	*14.04 ms*	*27.22 ms*	*3.59 ms*	*6.96 ms*
		3	*31.76 ms*	*72.04 ms*	*19.78 ms*	*39.07 ms*	*5.36 ms*	*10.59 ms*
		4	*39.39 ms*	*88.53 ms*	*25.32 ms*	*51.37 ms*	*6.83 ms*	*13.68 ms*

Table 4.1: Run-time performance at full HD and 4K on shutters of sizes 8 and 16.

			Naive Hist. Interp.		**Ours (full resolution)**		**Ours w/ aggreg. (2 × 2)**	
full HD	Diffusion		–		*0.88 ms*		*0.35 ms*	
	Other		–		*0.42 ms*		*0.39 ms*	
	Shutter Interp. & Comp.	$N \backslash T$	32	64	32	64	32	64
		2	*28.69 ms*	*64.80 ms*	*13.85 ms*	*28.10 ms*	*3.57 ms*	*7.31 ms*
		3	*40.07 ms*	*88.84 ms*	*20.33 ms*	*42.05 ms*	*5.54 ms*	*11.61 ms*
		4	*52.26 ms*	*117.35 ms*	*27.49 ms*	*57.13 ms*	*7.27 ms*	*15.01 ms*
4K	Diffusion		–		*3.04 ms*		*0.90 ms*	
	Other		–		*0.57 ms*		*0.44 ms*	
	Shutter Interp. & Comp.	$N \backslash T$	32	64	32	64	32	64
		2	*107.82 ms*	*244.04 ms*	*54.47 ms*	*111.13 ms*	*13.77 ms*	*28.37 ms*
		3	*154.93 ms*	*365.18 ms*	*80.15 ms*	*167.51 ms*	*21.50 ms*	*44.83 ms*
		4	*203.04 ms*	*456.94 ms*	*108.50 ms*	*226.09 ms*	*28.36 ms*	*58.46 ms*

Table 4.2: Run-time performance at full HD and 4K on shutters of sizes 32 and 64.

Interpolation), i.e. one without our search-space optimizations. For completeness, times for intermediate steps (shape and shutter interpolation for key frames (CPU) and shape rasterization) are aggregated and listed as *Other*. Since the diffusion process is image-based, performance only depends on resolution, and is listed as *Diffusion*.

The shutter interpolation step is further sped-up by aggregating coefficients. Here, we rasterize shapes and compute diffusion in half resolution, giving each 2 × 2 block of pixels the same coefficients for interpolation. We perform the interpolation only once for such a block, which significantly reduces the computation time for the interpolation step (see column *Ours w/ aggregation (2x2)* in Tables 4.1 and 4.2). We thereby achieve real-time performance even at 4K resolution, while minimally impacting image quality: we obtained SSIM [31] differences of 0.9982 (average) and 0.9777 (worst) compared to the full-resolution results (this difference is considered high quality for video compression [32]).

4.5. Conclusion

We presented a novel method to influence the shutter functions of a video in a post-process as well temporally, as also spatially. We propose a real-time interface, which allows artists to preview their modifications. It is possible to produce a large variety of results with little user interaction, making it possible to explore many alternatives before settling on the desired effect. The quick feedback being key, we propose optimized algorithms to achieve spatial interpolation of the shutter definitions. Our solution enables even novice users to explore many different opportunities to enhance, stylize and also impact the gaze of observers. ShutterApp is a step forward in shifting typical settings that need to be defined during recording to a post-process and even offers many more possibilities beyond standard shutter systems.

References

[1] N. Z. Salamon, M. Billeter, and E. Eisemann, *Shutterapp: Spatio-temporal exposure control for videos,* Computer Graphics Forum (Pacific Graphics) **38**, 675 (2019).

[2] T. Hoser, *Introduction to Cinematography: Learning Through Practice* (Taylor & Francis, 2018).

[3] J. Leirpoll, *Debunking the 180-degree shutter rule,* https://bit.ly/2Gr2Jif (2019), accessed: 2019-07-30.

[4] B. Desowitz, *Keeping 'The Fighter' Authentic,* https://bit.ly/2JyQrGo (2011), accessed: 2019-07-30.

[5] J. Paul, *Capture Intense Cinematic Action With High Shutter Speed,* https://bit.ly/2JBh3qd (2016), accessed: 2019-07-30.

[6] S. W. Oh, J. Lee, N. Xu, and S. J. Kim, *Fast user-guided video object segmentation by deep networks,* DAVIS Challenge on Video Object Segmentation - CVPR Workshops (2018).

[7] X. Chen, R. Girshick, K. He, and P. Dollár, *Tensormask: A foundation for dense object segmentation,* arXiv preprint arXiv:1903.12174 (2019).

[8] X. Li and C. C. Loy, *Video object segmentation with joint re-identification and attention-aware mask propagation,* DAVIS Challenge on Video Object Segmentation - CVPR Workshops (2018).

[9] X. Luo, N. Z. Salamon, and E. Eisemann, *Controllable motion-blur effects in still images,* Transactions on Visualization and Computer Graphics (2018).

[10] A. Glassner, *An open and shut case,* IEEE Computer Graphics and Applications **19** (1999).

[11] Behind The News, *How did Douglas Trumbull make the Stargate sequence in 2001 A Space Odyssey?* https://youtu.be/P1gn06np-7g (2015), accessed: 2019-07-30.

[12] S. P. Nicklin, R. D. Fisher, and R. H. Middleton, *Rolling Shutter Image Compensation,* in *RoboCup 2006: Robot Soccer World Cup X*, Vol. 4434 LNAI (Springer, 2007).

[13] R. S. Allison, L. M. Wilcox, R. C. Anthony, J. Helliker, and B. Dunk, *Expert viewers' preferences for higher frame rate 3d film,* Journal of Imaging Science and Technology **60** (2016).

[14] M. Stengel, P. Bauszat, M. Eisemann, E. Eisemann, and M. Magnor, *Temporal video filtering and exposure control for perceptual motion blur,* IEEE Transactions on Visualization and Computer Graphics **21** (2015).

[15] G. J. Brostow and I. Essa, *Image-based motion blur for stop motion animation,* Proc. ACM SIGGRAPH (2001).

[16] S. Liu, J. Wang, S. Cho, and P. Tan, *Trackcam: 3d-aware tracking shots from consumer video.* ACM Transactions on Graphics **33** (2014).

[17] M. Lancelle, P. Dogan, and M. Gross, *Controlling motion blur in synthetic long time exposures,* in *Computer Graphics Forum (Eurographics)*, Vol. 38 (2019).

[18] RSMB, *ReelSmart Motion Blur,* http://revisionfx.com/products/rsmb/ (2019), accessed: 2019-05-04.

[19] M. Fuchs, T. Chen, O. Wang, R. Raskar, H. P. Seidel, and H. P. A. Lensch, *Real-time temporal shaping of high-speed video streams,* Computers and Graphics **34** (2010).

[20] K. Templin, P. Didyk, K. Myszkowski, and H.-P. Seidel, *Emulating displays with continuously varying frame rates,* ACM Transactions on Graphics **35** (2016).

[21] K. Debattista, K. Bugeja, S. Spina, T. Bashford-Rogers, and V. Hulusic, *Frame rate vs resolution: A subjective evaluation of spatiotemporal perceived quality under varying computational budgets,* Computer Graphics Forum **37** (2018).

[22] M. Gross and S. A. Schriber, *Lucid dreams of gabriel,* Short Movie (2014).

[23] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, *Deepflow: Large displacement optical flow with deep matching,* in *IEEE International Conference on Computer Vision* (2013).

[24] N. Bonneel, M. van de Panne, S. Paris, and W. Heidrich, *Displacement interpolation using lagrangian mass transport,* ACM Transactions on Graphics **30** (2011).

[25] A. L. Read, *Linear interpolation of histograms,* Nuclear Instrumuments and Methods in Physics Research **A425** (1999).

[26] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin, *Diffusion curves: a vector representation for smooth-shaded images,* ACM Transactions on Graphics (2008).

[27] S. Goedegebure, *Big Buck Bunny,* Short Movie (2008).

[28] N. Goodnight, C. Woolley, G. Lewin, D. Luebke, and G. Humphreys, *A multigrid solver for boundary value problems using programmable graphics hardware,* in *SIGGRAPH Courses* (2005).

[29] S. Jeschke, D. Cline, and P. Wonka, *A GPU laplacian solver for diffusion curves and poisson image editing,* ACM Transactions on Graphics **28** (2009).

[30] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot, *Controllable diffusion curves,* in *INRIA Technical Report* (2010).

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et al.*, *Image quality assessment: from error visibility to structural similarity,* IEEE Transactions on Image Processing **13** (2004).

[32] W. Loh and D. B. L. Bong, *Video quality assessment method: Md-ssim,* in *IEEE Intl. Conference on Consumer Electronics* (2015).

5

Video Editing with Spatio-temporal Object Control

N. Z. Salamon, M. Billeter, S. Lee and E. Eisemann

Every year is getting shorter
never seem to find the time.

We present a real-time end-to-end solution for spatio-temporal assisted video editing to allow a user to influence the timing of a video locally and even fuse content from different moments spatially. Such operations are common in movie productions but often involve tedious and manual intervention, in particular due to inconsistencies that arise during the manipulation. Our solution facilitates this task with a targeted interface and algorithms to manipulate time, track objects, and handle inconsistencies.

This chapter was submitted to a peer-reviewed conference and is currently under review.

5.1. Introduction

Editing can be time-consuming; raw material is recorded in segments, which are combined into a movie. For example, *First Man*'s editor Tom Cross says the movie editing process took almost a year with extended work shifts [1]. Nevertheless, the final product must match the director's vision [2]. As re-shoots are expensive, looping, repeating or speeding up/slowing down existing sequences are common ways to better match the narrative in post-production. In a scene in *Star Wars Episode IV - A New Hope*, a Tusken raider attacks the protagonist Luke Skywalker. The raider celebration was too short and had its movement looped to emphasize the successful attack [3]. However, editing operations can be much more complex if applied to selected elements rather than the whole frame and spatio-temporal conflicts can arise. For example, imagine a choreographed scene, where the movements are perfectly aligned in post-production. Yet, if two protagonists overlap, using different moments in time for the final composite leads to inconsistencies. Even just manipulating the speed of a single scene object requires infilling of its original locations.

This paper presents a real-time end-to-end framework to support interactive spatio-temporal editing and compositing of videos. Users can influence the timing in a video locally on selected elements or the whole video. Simple editing examples include time-shifts or speed-ramping of scene elements. We support additional editing options to shift/clone elements and to handle object overlap. Our approach enables quick experimentation and involves specialized tools and visualization solutions to facilitate the workflow. To this extent, our contributions can be summarized as follows:

- a solution for localized spatio-temporal adjustments in videos;
- a set of tools and visualizations for common edits; and
- an interactive implementation for a quick iterative workflow.

5.2. Related Work

Our work builds upon advances in image and video processing. In this section, we give a short overview of related work.

Expressive video editing A recent trend goes towards synthesizing videos from small input sets, such as still images or semantic maps [4, 5]. The photorealistic results are impressive but there is little room for artistic control.

Several methods aid users with editing the overarching video structure and timeline, primarily by placing cuts and transitions [6–8], combining different shots [9] or summarization [10, 11]. In contrast, we focus on adjustments and edits that are applied to a single segment of consecutive frames.

Past video manipulations target people's behavior [12], object transfer [13] and compositing from different time quanta into a single image [14, 15]. When dealing with spatio-temporal modifications inside a video stream, time can be warped at distinct locations, both accelerating [16] and de-animating spatial segments[17]. Templin et al. [18] explore spatially and temporally varying framerates employed independently of the actual display frame rate. Stengel et al. [19] propose a perceptual and localized motion blur that takes the predicted eye motion into account. ShutterApp [20] models shutter functions and allows a user to define them at key points over space and time; functions for remaining frames and positions are then interpolated and applied to each pixel in each frame. Despite fine-level controls, making adjustments to specific elements is not possible and may result in artifacts and spatio-temporal conflicts.

Segmentation Objects need to be identified and masked, which is an active research area [21] and many commercial packages exist (e.g., [22]). The yearly DAVIS challenges [23–25] provide excellent comparisons between video segmentation methods in realistic settings, but none has perfect accuracy. We settled on SiamMask [26], which uses a pre-trained neural network and bases the segmentation on an initial bounding rectangle. It does not require a pre-segmented initial mask and its interactive processing rates enable on-the-fly segmentation for the entire video.

Reconstruction Modifying video content, e.g., shifting an object, can lead to disocclusions and holes. Video reconstruction addresses this longstanding problem [27]. Recently, the focus is on machine-learning approaches that automatically infill content [28–30] and patch-based solutions [31–33]. Specialized approaches can remove static objects from videos in real time [34]. However, our focus lies on enabling fast experimentation with different user-driven editing choices in general scenes.

5.3. Our Framework

Our method enables users to influence the timing of objects in a video sequence. Starting with an input video, objects of interest are segmented semi-automatically (Section 5.3.1). Each of these objects forms its own video, to which users can apply temporal adjustments (Section 5.3.2). Resulting conflicts are highlighted in a visualization tool and can be resolved by, for example, infilling disoccluded frame content using our semi-automated tools (Section 5.3.3).

Each step and tool is designed with simple interaction (i.e., brushing, selections),

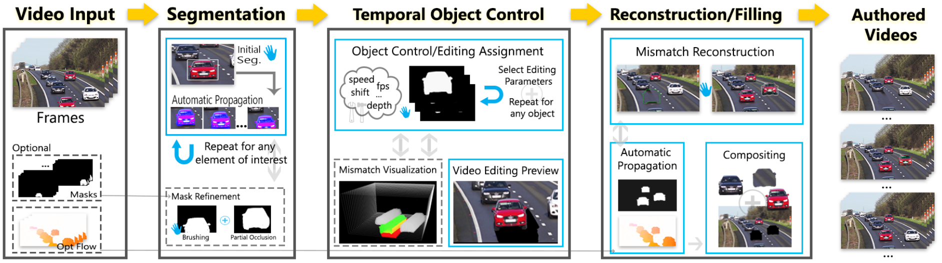


Figure 5.1: Overview. The user segments objects in the input video and manipulates their timings. Resulting conflicts are visualized and resolved with reconstruction tools. The new video is rendered in real time. Hand icons indicate interventions, dashed boxes optional steps.

interactivity and real-time feedback in mind. A majority of the intermediate results are computed on the fly, and do not have to be stored in memory, which is helpful for high-resolution videos. Fig. 5.1 shows an overview of the complete pipeline.

5.3.1. Video Object Selection

Initially, the objects of interest are identified and isolated. To select an object, users navigate to the time when the object first appears in the input video V and draw a bounding rectangle around the object. The rectangle is passed to SiamMask [26] to perform an automatic video segmentation using this selection.

Users can make corrections to the returned segmentation, either directly, or at any future frame by pausing the video. For this, we provide a brushing metaphor, as well as a lasso-tool for free-form used-defined segmentations with GrabCut [35]. The adjusted masks can optionally be fed back to segment the following frames, effectively propagating the corrections. Fig. 5.2 illustrates this process. As propagation methods, we support ROAM [36], VS-ReID [37] or CapsuleVOS [38]. Lassos can also be keyframed to define morphing segmentation regions over time.

Segmentation is repeated for all objects of interest, resulting in the segmentation masks M_o for each object o. Conceptually, each segmented object forms its own video $\mathcal{V}_o$ that only contains the pixels selected by M_o (a binary image; 1 on object o, 0 otherwise). Nevertheless, the video $\mathcal{V}_o$ can be computed on the fly from the input video V and the mask M_o, and is therefore never explicitly materialized in memory.

5.3.2. Video editing

Next, users edit the timing of the masked object videos $\mathcal{V}_o$ independently. The resulting transformed per-object videos, $\mathcal{F}_o$, are then fused to create the final output video F.

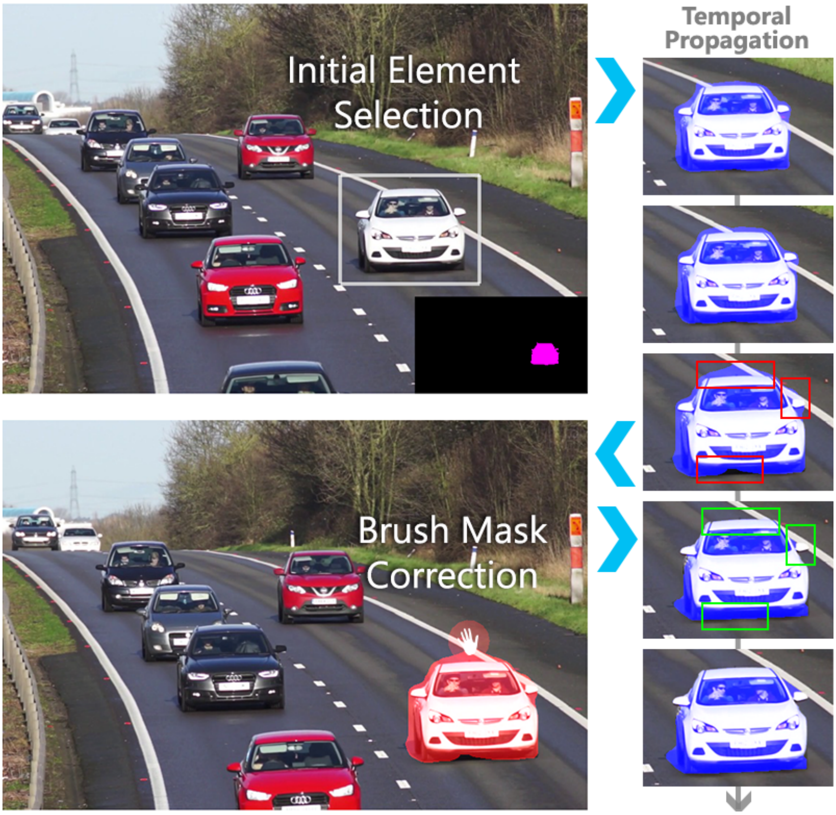


Figure 5.2: Segmentation. Top left: a user-provided rectangle to select the object. Right: masks are automatically generated for the current and following frames; red and green rectangles indicate, respectively, miss-segmented areas and corrected ones after adjustments. Bottom left: users can correct masks via brushing tools. Adjusted annotations are fed back to segment subsequent frames. Source: pixabay.com

Temporal Object Control

For each object, users may define a time mapping function, $\tau_o(t)$. This function specifies, for each output frame t, a corresponding input frame time in the source video $\mathcal{V}_o$. The transformed frames $\mathcal{F}_o$ and alpha masks $\mathcal{A}_o$ are then defined as:

$$\mathcal{F}_o\,(t) \quad = \mathcal{V}_o\,(\tau_o(t))\,, \;\mathcal{A}_o\,(t) \quad = M_o\,(\tau_o(t))$$

The transformation can be evaluated on the fly, meaning that neither $\mathcal{F}_o$ nor $\mathcal{A}_o$ needs to be pre-computed and stored in memory. Fig. 5.3 (top left) illustrates a frame from a time-shifted video $\mathcal{F}_o$.

The function $\tau_o(t)$ is represented as a piecewise linear function. Users define the function by selecting keyframes $t_{o,k}$ and specifying a time shift $\delta_{o,k}$ and playback speed $\rho_{o,k}$ for the segment following the keyframe:

$$\tau_o(t) = \delta_{o,k} + t\,\rho_{o,k} \quad \text{if} \quad t_{o,k} \leq t < t_{o,k+1}$$

The resulting function $\tau_o(t)$ can be discontinuous.

Time edits can also be created on-the-fly by dragging objects around the scene. After clicking on an object, the mouse movement direction is compared to the optical flow to define, together with moving distance, the parameters ρ and δ.

Compositing

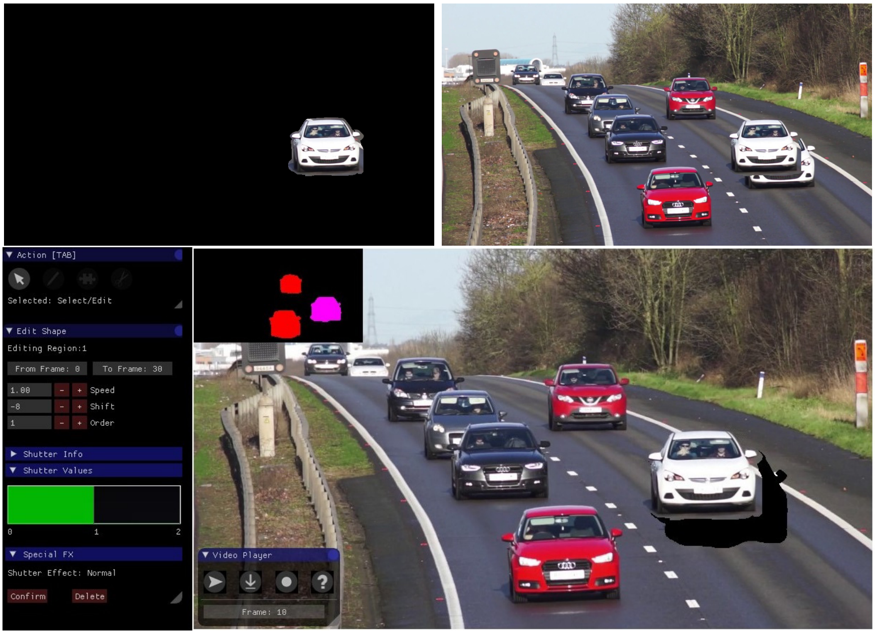
To create a final video frame $F(t)$, we need to carefully recombine the transformed object video frames $\mathcal{F}_o(t)$ together with a suitable background B and reconstructed content R. The reconstructed content R is used to fill in disocclusions created when the transformed object frames do not cover the same regions of the original input.

The background mask A_B is $B = \prod_{objects}(1 - M_o)$, in order to avoid duplicate object instances (Fig. 5.3 (top-right)). We generate the mask A_B just in time for each frame, meaning that we do not need to store more than one background instance at any time.

For an early preview ($P(t)$), we composite the transformed object frames $\mathcal{F}_o(t)$ using the masks $\mathcal{A}_o(t)$ directly onto the incomplete background $B = A_B * V$ (showing black regions, where objects were originally located). As, during compositing, the order matters, we follow a user-defined order that, by default, equals the order in which the objects were extracted. However, users can change the order, if needed, for a different editing narrative. The preview frames $P(t)$ are rendered in real time, enabling users to quickly experiment with different parameters (Fig. 5.3, bottom). For each pixel p, we loop over the objects in composition order. We take $\mathcal{F}_o(t)(p)$ of the topmost object o for which $\mathcal{A}_o(t)(p)$ is one. If there is no object, we use $B(t)$.

Mismatch Visualization

Mismatches correspond to pixels that are missing due to occlusion, such as those in B, where A_B was zero, but might also stem from an occlusion of a chosen object itself. These gaps require filling. To help users locate and identify mismatches in the edited video, we create a 3D visualization. It shows a volume, where Z-slices correspond to a frame at time t. Mismatches are drawn in red. To give some context of the whole scene, the objects' $\mathcal{A}_o(t)$ are shown in light gray. Users can optionally highlight a specific object by selecting it, which is then shown in green (Fig. 5.4).



Selected: Select/Edit

Editing Region:1

From Frame: 0 | To Frame: 30

1.00 − + Speed

-8 − + Shift

1 − + Order

Shutter Effect: Normal

Frame: 10

Figure 5.3: Compositing. Top: frame from the edited object video F_o (left). Direct fusion with the original sequence V creates a duplicate (right). Bottom: Real-time preview in our interface. The current instance is removed, and the black pixels on the removal region represent a mismatched region whose content must be reconstructed.

5.3.3. Mismatch Reconstruction

Mismatches are resolved by reconstructing the disoccluded areas. The best choice of reconstruction method varies depending on the target area. While static background can be filled with still content from a different frame, filling parts of deformable objects will require content to be warped at the destination position. The desired infilled content might depend on user-preferences and artistic directions. Further, we want to achieve a fast feedback loop to support the timing adjustments. We therefore provide semi-automatic tools to reconstruct areas by replicating content from different places in time and space. These tools create an image R, which contains content to fill regions with mismatches. In each frame, R and B are combined via Poisson inpainting [39] to achieve a seamless fill of the mismatches. In the following, we describe our tools to build R. For now, we will focus on a single frame. Users select the reconstruction method and paint annotations over the mismatches at the current frame. We will discuss how annotations can be propagated to the following frames in the next subsection.

All tools start with the user choosing a mismatch area. The most simple solution is to directly copy background pixels from the nearest moment in time, when they became visible at this location. If a mismatch is short (around one second), this

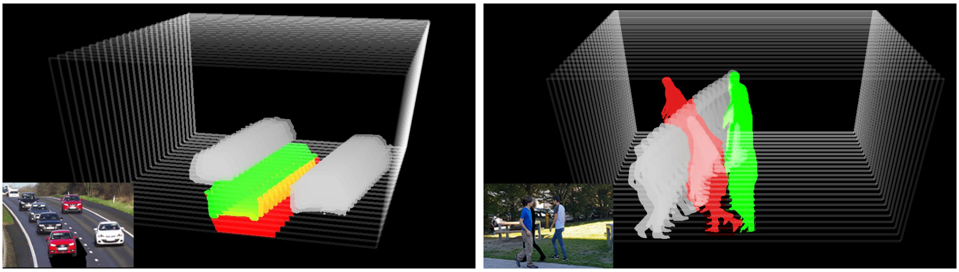


Figure 5.4: Mismatches. Original objects are gray, a scene preview is shown in the bottom-left inset. Left: three cars in the foreground were segmented; the red car is shifted back in time, creating a mismatch (highlighted by the red mask) and a projection of its new positions (green mask). Right: By slowing down one person on the right, it is removed from its original position (red), delaying the encounter in the authored scene. Note that the yellow color stems from green and red areas from different frames/slices overlapping in the 3D view.

simple approach can produce good reconstruction results in a variety of scenes, as illustrated in Fig. 5.5 (bottom). Alternatively, users can select source data on any frame at any position in the input video. For this, the outline of the selected area is shown as a mouse cursor and the user can navigate the video to place it (Fig. 5.6). Such solutions fall short for complex, non-static, and deformable content. For example, if an object is occluded in the original sequence but then disoccluded due to a timing adjustment, the exact information cannot be found in the sequence and we want to offer the possibility to choose a plausible source.

For complex cases, we enable users to specify a few locations within the mismatch and associate them to other locations in a different location and moment in time in the video. These correspondences form pairs of points that work as anchors, allowing us to derive a per-pixel offset map that interpolates these constraints to define the necessary infill operation. To this extend, we create an image D of the same resolution as R, where each pixel will hold a spatio-temporal offset (o, s) such that for a pixel p in frame t, we would fetch the value of pixel $p + o$ in frame $t + s$. Initially D is set to zero, then the anchor points will be filled with the corresponding offsets and used as constraints. Next, we find a smooth interpolant respecting these hard constraints using a multi-grid solver [20], which robustly solves the Poisson equation. The resulting map D then contains in each pixel the information from where to copy.

Fig. 5.7 shows the use of the anchor-point method to easily incorporate a perspective change in scale. In general, anchor points I_s and I_d can optionally be keyframed and propagated to following frames to track movement and deformations in the source or destination images, respectively. A suitable propagation approach is presented in the next subsection.

For all tools, users can select the pixels that are considered valid during the copy operation. They can be restricted to pixels of a segmented object (e.g., to reconstruct partially occluded selected objects) or to background pixels avoiding

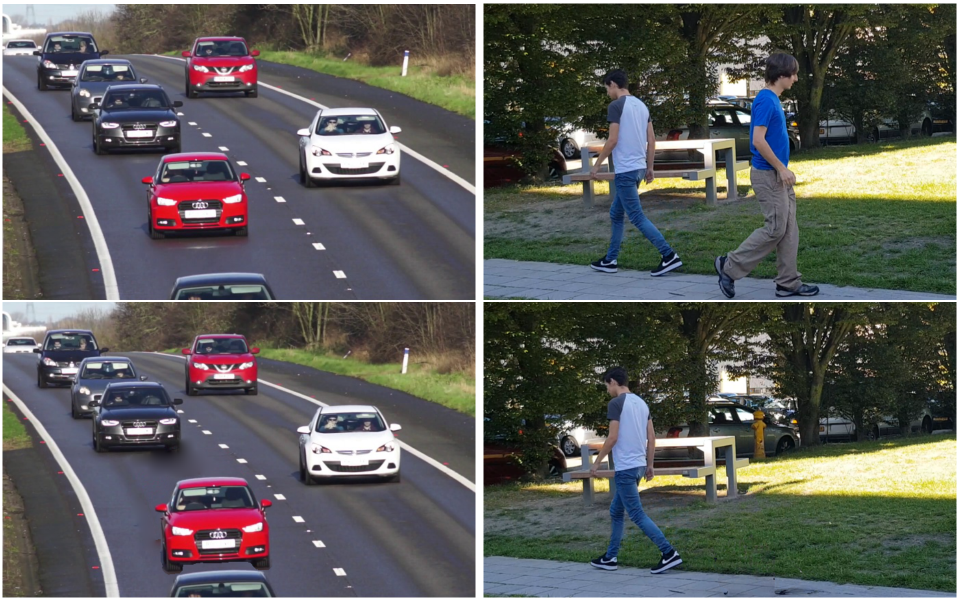


Figure 5.5: Temporally-driven hole filling. Top: original frames (cropped). Bottom: our approach recovers the background information for the disoccluded area when the red car is moved forward (left) and the person on the right is removed from the scene (right).

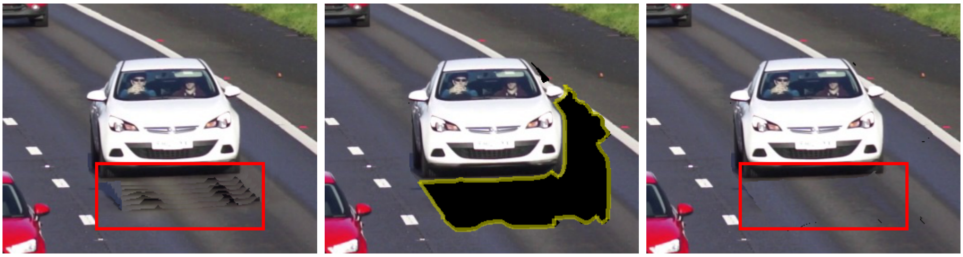


Figure 5.6: Patch-based filling. Left: temporally-driven filling fails to find a suitable background content due shadow cast by the car. Middle: the user clicks on the area to be reconstructed and the mouse cursor takes the shape of the patch needed for filling. The source patch can now be selected from any position inside the video stream. Right: the mismatch is resolved by replicating the asphalt patch from a previous frame.

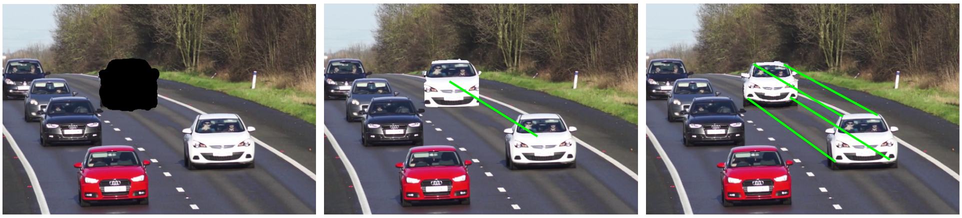


Figure 5.7: Anchor-point filling. Left: car in the background is removed. Middle: white car is replicated to reconstruct the mismatch area. A patch copy causes perspective artifacts. Right: using anchor points adjust the source content to the appropriated scale.

selected objects (e.g., to infill gaps in the background). One catch is that objects with filled-in holes still write the missing content in R for efficiency. To correctly composite, the mask $A_o(t)$ is extended with a special value that indicates to directly use the value in R.

Reconstruction Propagation

The reconstruction tools operate on a single frame. To aid with editing of a sequence of frames, we provide an automated method to map user-defined reconstruction operations to future frames. Users can manually intervene if adjustments are necessary. Multiple tools can be combined and the feedback is immediate, as the reconstruction is performed in real time.

In the simplest case, operations can just be replicated as-is to future frames. However, when matching patches or using anchor points, it is often desirable that the user-defined annotations move with the content. To do so, we allow the user to specify that annotation points are moved based on optical flow [40].

Users can define whether both or only destination or source points are moved. Additionally, if segmentations masks are defined using the keyframed lasso-tool, annotations can be set to follow the keyframes. For patches, we rely on the averaged optical flow of the region; for anchor points, the average flow in a 5×5 pixel region around them is applied.

5.3.4. Framework and Interface Extensions

In addition to the temporal editing, we provide a few additional options for manipulating objects. In particular, we allow the user to specify spatial shifts to individual objects o, which translate $\mathcal{F}_o(t)$ and $\mathcal{A}_o(t)$. Spatial shifts can be keyframed such that the translation changes over time, making it possible to introduce additional movements in the video. Similarly, we support duplications with potentially timing adjustments and object removals ($\mathcal{A}_o(t) := 0$)

Besides direct time adjustments, we also support shutter functions [20, 41] to control exposure parameters per individual objects. When using a shutter function, $F_o(t)$ and mask $\mathcal{A}_o(t)$ are virtually filtered - the images are not stored but computed on the fly with a modified compositing process; objects are alpha-blended onto the (reconstructed) background in the composition order using $\mathcal{A}_o(t)$ as blending weights.

5.4. Results

Our implementation has a modular architecture bridging $C++$ and $Python$, running on a Windows 10 laptop with a Intel i7-7700 CPU and GTX1070/8GB GPU. Timings were measured for full HD (1920x1080) resolution videos, which were recorded with a Samsung S8 and a GoPro Hero 3 or downloaded (see captions).

Segmentation [26] and optical flow [40] run at 8 and 6 frames per second, respectively, and both can be pre-loaded. We also experimented with ground-truth masks to evaluate the editing process separately (see captions). The editing interaction and final result computation ran above 60 frames per second for all results in this paper. In the following, we present a variety of scenes authored using different time editing choices. For most examples, the users took from one to two minutes to perform the intended time adjustments and reconstructions, with edge cases of 20 seconds for Fig. 5.13 and 8 min for Fig. 5.10.

In Fig. 5.8, the patient originally looks back at the camera then looks away when noticing the camera. The scene is edited for two different narratives. First, we skip the frames when she looks at the camera, aiming to focus on the conversation (left). Skipping frames, however, would shorten the video and, to compensate, we slow down later movements. We can also make the patient hold the look at the camera by looping the movement, for the opposite narrative, as asking the camera operator to stop recording (right).

Scene narrative can be further manipulated by changing the speed of objects. In Fig. 5.9 (top), impression of two people crossing the road together is voided when we modify the walking speed of one individual. The same editing choice is applied to Fig. 5.9 (bottom), where the kart on the inside is accelerated to take over the position.

In Fig. 1.4, the basketball is slowed down, while the scene keeps its timing. The original ball positions are filled in with the simple filling approach from neighboring frames. A more challenging example is Fig. 5.10. We want to remove the person in a blue shirt. A simple filling approach is not enough (top, right). Instead, we recreate the occluded person using anchor points from previous frames; at the torso, leg and feet positions. We set the propagation to shift the annotated points using only the flow of the source anchor. This warps the content and replicates the movement hidden by the occluder from previous frames (bottom, left). With the same steps, we can also change the relative depth and bring the background person

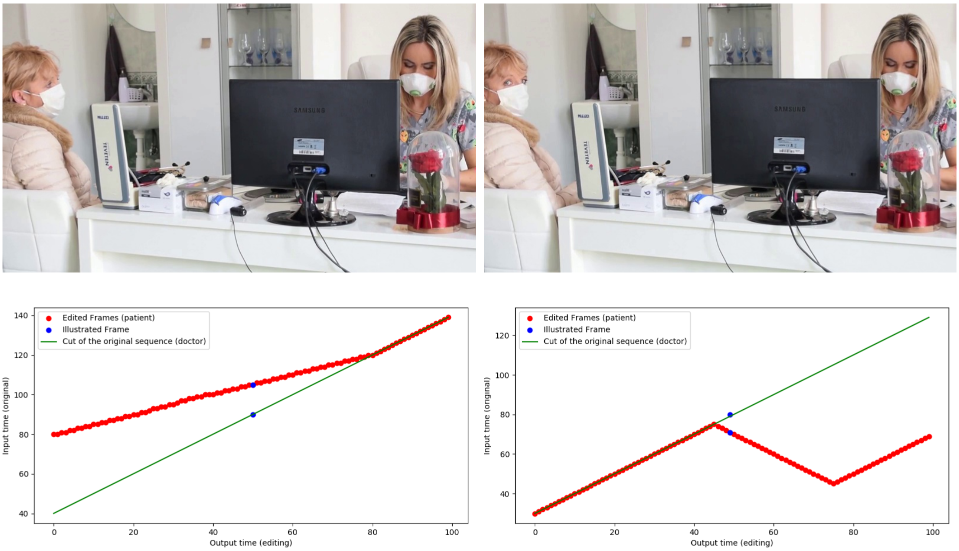


Figure 5.8: When the scene is cut shorter to focus on an event, slowing down or looping the content fills the necessary time. Top: the same output frame of the two authored narratives, slowing down (left) and looping (right) exclusively the patient. Bottom: time mapping function used for the examples. Lines describe the mapping between output time (x-axis) and the input time (y-axis) from where the frame's data is sourced. Blue marks indicate the current frame, which is a compositing of the edited and original video stream cuts. Source: pixabay.com.

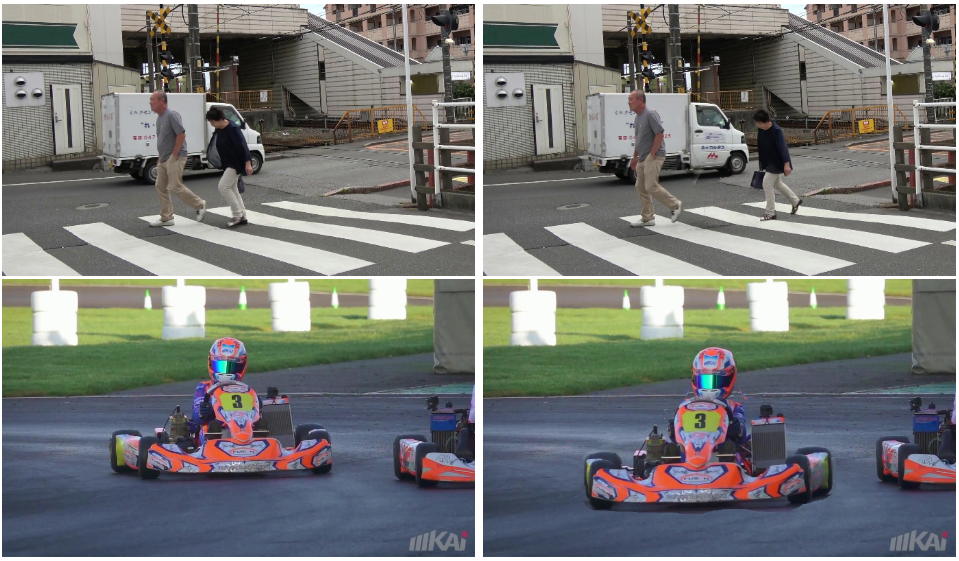


Figure 5.9: Narrative manipulation. Left: original scenes. Right: the couple now walks apart at different speeds (top); the kart on the inside is sped up so that positions are changed after the turn (bottom). Video sources: DAVIS dataset[23].

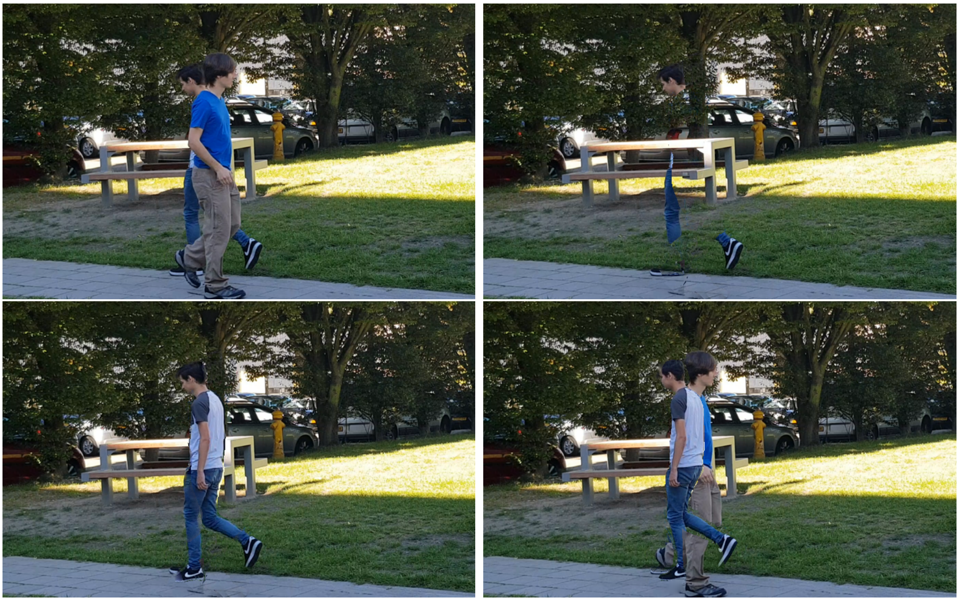


Figure 5.10: Editing multiple scene objects and their depth order. Top: original frame (left); the person in blue is removed from the scene, revealing information not originally acquired, which is temporally filled with background pixels (right). Bottom: the correct reconstruction is performed to keep a single person in the scene (left); a new narrative is created by switching relative depth between persons (right).

forward (bottom, right). A similar edit in Fig. 5.11 moves the clownfish behind the black fish.

Keyframes can change the trajectory of scene objects. In Fig. 5.12, we dispel one skydiver by drawing a lasso around the original position and keyframing the destination as the video's border. Inside the lasso, anchor points guide the reconstruction and shift along with the keyframes. Original positions are infilled using our temporally-driven approach.

Effects with custom shutter functions, such as motion blur, can be added seamlessly. For example, in order to guide the gaze direction onto the players, in Fig. 5.13 (left), a motion blur is applied to the referee. In Fig. 5.13 (right), a Harris shutter is applied to expressively highlight the attack. In both examples, the proper segmentation and compositing ensures a valid output, which was previously not handled by other approaches ([20]).

Other temporal effect particularly difficult to illustrate on paper is a time adjustments of the framerate for single objects. Such an appearance has been used in movies for artistic purposes (e.g., *Spiderman: Into the Spiderverse*, where the inexperienced protagonist was rendered with only 12 fps, while its alter ego appeared at 24 fps). Our approach makes such manipulation possible, on real-world footage.

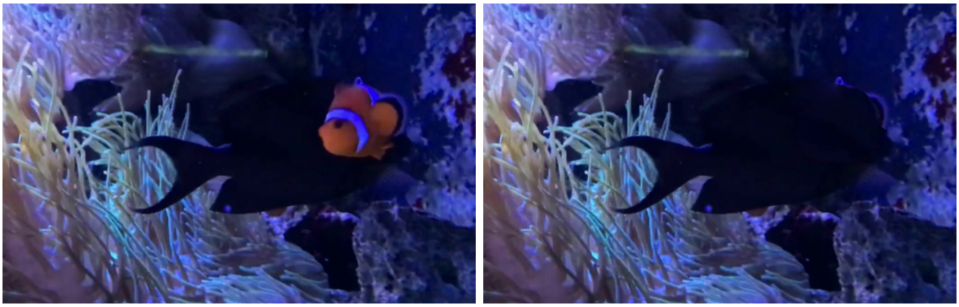


Figure 5.11: The clownfish is originally on top of the black fish (left), which can be changed by switching the compositing order and applying reconstruction (right). Source: Need for Speed dataset[42].

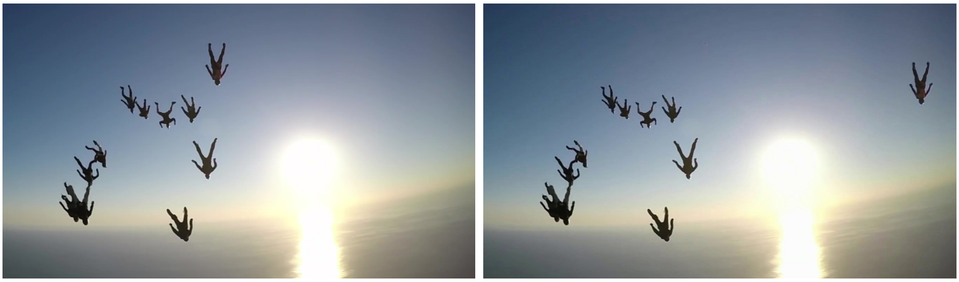


Figure 5.12: Left: original frame. Right: the same frame, with a new trajectory created for the person on top by using interpolated keyframes upon reconstruction. Source: Need for Speed dataset[42].

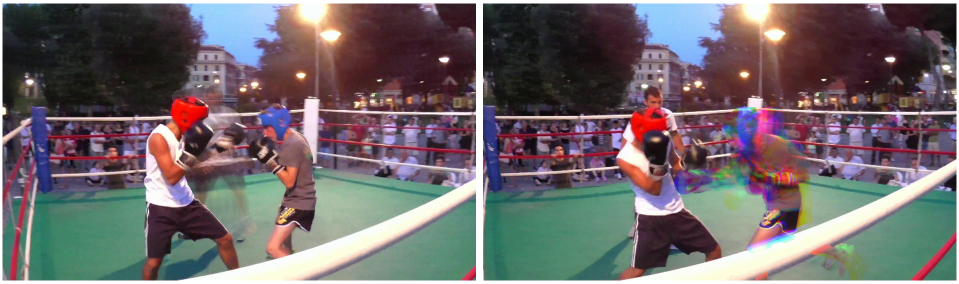


Figure 5.13: Motion blur (left, on the referee in the background) and Harris Shutter (right, on the player in the foreground) effects applied to single elements in the scenes at different depths. Video and masks source: DAVIS dataset[23].

5.5. Conclusion

Post-production is complex because one needs to work mostly with the footage that has been already captured with fixed parameters. Our solution allows a user to make more out of the original sequences and enables a large variety of timing adjustments. With real-time preview and visualization capabilities, we enable efficient exploration of different narratives and editing options.

Specifically, our solution supports adjusting the timing of individual objects via spatio-temporal video editing. The laborious correction of inconsistencies during editing is simplified by our semi-automatic reconstruction and visualization tools. Edits work with sparse annotations and only a few parameters, which facilitates fast prototyping and exploration of temporal adjustments for both novice and experienced users.

References

[1] S. Kouguell, *INTERVIEW: Academy Award®-Winning Film Editor Tom Cross Discusses First Man,* https://bit.ly/2ZKH3q4 (2019), accessed: 2020-07-06.

[2] Z. Sharf, *Christopher Nolan's Editor Is Aware Viewers 'Completely Misunderstand' Nolan Movies,* https://bit.ly/3iGvdWK (2020), accessed: 2020-07-06.

[3] C. Taylor, *How Star Wars conquered the Universe: The Past, Present, and Future of a Multibillion Dollar Franchise* (Hachette UK, 2015).

[4] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro, *Few-shot video-to-video synthesis,* in *Conference on Neural Information Processing Systems (NeurIPS)* (2019).

[5] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, *Video-to-video synthesis,* in *Conference on Neural Information Processing Systems (NeurIPS)* (2018).

[6] M. Leake, A. Davis, A. Truong, and M. Agrawala, *Computational video editing for dialogue-driven scenes.* ACM Trans. Graph. (TOG) **36** (2017).

[7] A. Truong, F. Berthouzoz, W. Li, and M. Agrawala, *Quickcut: An interactive tool for editing narrated video,* in *ACM Symposium on User Interface Software and Technology (UIST)* (2016).

[8] H.-Y. Wu, T. Santarra, M. Leece, R. Vargas, and A. Jhala, *Joint attention for automated video editing,* in *Workshop on Cinematography and Editing* (2020).

[9] J. Rüegg, O. Wang, A. Smolic, and M. Gross, *Ducttake: Spatiotemporal video compositing,* Computer Graphics Forum (CGF) **32** (2013).

[10] A. Rav-Acha, Y. Pritch, and S. Peleg, *Making a long video short: Dynamic video synopsis,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Vol. 1 (2006).

[11] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, *Video summarization with long short-term memory,* in *European Conference on Computer Vision (ECCV)* (2016).

[12] T. K. Shih, N. C. Tan, J. C. Tsai, and H.-Y. Zhong, *Video falsifying by motion interpolation and inpainting,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).

[13] Y. Li, J. Sun, and H.-Y. Shum, *Video object cut and paste,* ACM Trans. Graph. (TOG) **24** (2005).

[14] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz, *Video object annotation, navigation, and composition,* in *ACM Symposium on User Interface Software and Technology (UIST)* (2008).

[15] Y. Pritch, A. Rav-Acha, and S. Peleg, *Nonchronological video synopsis and indexing,* IEEE transactions on pattern analysis and machine intelligence **30** (2008).

[16] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, *Evolving time fronts: Spatio-temporal video warping,* (2005).

[17] J. Bai, A. Agarwala, M. Agrawala, and R. Ramamoorthi, *Selectively deanimating video,* ACM Trans. Graph. (TOG) **31** (2012).

[18] K. Templin, P. Didyk, K. Myszkowski, and H.-P. Seidel, *Emulating displays with continuously varying frame rates,* ACM Trans. Graph. (TOG) **35** (2016).

[19] M. Stengel, P. Bauszat, M. Eisemann, E. Eisemann, and M. Magnor, *Temporal video filtering and exposure control for perceptual motion blur,* Transactions on Visualization and Computer Graphics (TVCG) **21** (2015).

[20] N. Z. Salamon, M. Billeter, and E. Eisemann, *ShutterApp: Spatio-temporal exposure control for videos,* Computer Graphics Forum (Pacific Graphics) **38** (2019).

[21] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, *Image segmentation using deep learning: A survey,* arXiv preprint arXiv:2001.05566 (2020).

[22] Adobe Inc., *After Effects,* Computer Software. Available: https://adobe.ly/3c8hvra (2020).

[23] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, *The 2017 davis challenge on video object segmentation,* arXiv:1704.00675 (2017).

[24] S. Caelles, A. Montes, K.-K. Maninis, Y. Chen, L. Van Gool, F. Perazzi, and J. Pont-Tuset, *The 2018 davis challenge on video object segmentation,* arXiv:1803.00557 (2018).

[25] S. Caelles, J. Pont-Tuset, F. Perazzi, A. Montes, K.-K. Maninis, and L. Van Gool, *The 2019 davis challenge on vos: Unsupervised multi-object segmentation,* arXiv:1905.00737 (2019).

[26] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, *Fast online object tracking and segmentation: A unifying approach,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[27] S. Ilan and A. Shamir, *A survey on data-driven video completion,* Computer Graphics Forum (CGF) **34** (2015).

[28] D. Kim, S. Woo, J.-Y. Lee, and I. So Kweon, *Deep video inpainting,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[29] S. Lee, S. W. Oh, D. Won, and S. J. Kim, *Copy-and-paste networks for deep video inpainting,* in *International Conference on Computer Vision (ICCV)* (2019).

[30] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu, *Contextual residual aggregation for ultra high-resolution image inpainting,* (2020), arXiv:2005.09704 .

[31] Y. Wexler, E. Shechtman, and M. Irani, *Space-time completion of video,* IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **29** (2007).

[32] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, *Video inpainting of complex scenes,* SIAM Journal on Imaging Sciences **7** (2014).

[33] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, *Temporally coherent completion of dynamic video,* ACM Trans. Graph. (TOG) **35** (2016).

[34] J. Herling and W. Broll, *High-quality real-time video inpaintingwith pixmix,* IEEE Transactions on Visualization and Computer Graphics (TVCG) **20** (2014).

[35] C. Rother, V. Kolmogorov, and A. Blake, *Grabcut: Interactive foreground extraction using iterated graph cuts,* ACM Trans. Graph. (TOG) **23** (2004).

[36] O. Miksik, J.-M. Pérez-Rúa, P. H. Torr, and P. Pérez, *Roam: a rich object appearance model with application to rotoscoping,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[37] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy, *Video object segmentation with re-identification,* arXiv preprint arXiv:1708.00197 (2017).

[38] K. Duarte, Y. S. Rawat, and M. Shah, *Capsulevos: Semi-supervised video object segmentation using capsule routing,* in *IEEE Conference on Computer Vision (ICCV)* (2019).

[39] P. Pérez, M. Gangnet, and A. Blake, *Poisson image editing,* ACM Trans. Graph. (TOG) **22** (2003).

[40] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, *Fast optical flow using dense inverse search,* in *European Conference on Computer Vision (ECCV)* (2016).

[41] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, and J. Davis, *Synthetic shutter speed imaging,* Computer Graphics Forum (CGF) **26** (2007).

[42] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, *Need for speed: A benchmark for higher frame rate object tracking,* arXiv preprint arXiv:1703.05884 (2017).

6

Conclusion

This work presented several methods and corresponding tools to control exposure in a post-process for both images and videos. The tools are optimized to achieve real-time response rates and designed to support novice users. Using simple acquisition steps, involving no special equipment, our approaches can assist the fast-growing visual content producing communities.

In Chapter 2, our virtual light painting approach changes the trial-and-error capturing process into a controlled process, allowing artists to test parameters and create new light arts without recording a new scene. Improvements over traditional light-painting methods, such as person and specularity removals were also presented. In Chapter 3, we introduced an easy-to-use motion-blur tool in which a single image is given and, with a few clicks, long-exposure blur effects can be added to different objects in the scenes. Artifacts that raised from disocclusion and composition are automatically handled. A multi-directional motion blur method, not achievable with common hardware, was also presented. In Chapter 4, we smoothly propagate and integrate different exposures in a single video scene. In addition to the optimized shutter interpolation and the image-based diffusion process, the intuitive annotations make the compositing easier, whereas the real-time feedback allows for new stylizing at no cost. Finally, in Chapter 5, we presented an end-to-end pipeline for spatio-temporal control of individual objects in videos. An interactive implementation allows for segmenting the objects of interest and applying time and space adjustment functions. The inconsistencies raised during the editing process are visualized and resolved in real time with our tailored reconstruction tools.

Together, we believe that the developed tools and methods can bring a new level of expressiveness to novice users and also contribute to speed up the post-processes in the visual-content industry. We showed several results of high-quality, created in a matter of seconds on commodity hardware and without special acquisition hardware. This is a step forward on moving the creativity control to post-processes instead of relying on multiple captures and real-world adjustments.

In the future, we would like to study how different parameters influence perception on photos and movies. To that extent, one can classify scenes and camera parameters to derive presets and mimic style choices manually set by photographers and directors. The styles can later be replicated to different scenes with similar content, smoothing the learning curve for the final user.

Epilogue

It's a long way to the top. The hilly PhD journey comes to an end with this dissertation. Apart from the valuable experience I gained, I believe this work can inspire other researchers and also be employed on different fronts.

In practice, our findings can be further deployed to help novice producers realizing their creative visions. Together with the Brazilian Culture Incentive Law (#8.313/91)) fostering audiovisual productions, for example, our methods can encourage movie producers and artists to post-process their artworks and also author novel content. In this sense, it can assist smaller and middle-sized companies in catching up with the quality level of large enterprises and giving access to modern cinematic technology.

Last but not least, the most important aspect of this work is to give many users the possibility to express themselves in form of self-made visual content, which will be the central content of the internet evolution. It is my sincere hope that you, the reader, feel inspired and can also take part in this. In the end, research needs to meet the final user.

The italic highlights on the preface and here are, respectively, from the songs "Should I stay or should I go?" by The Clash and "It's a long way to the top (If you wanna Rock 'n' Roll)" by AC/DC. The epigraph quotes on Chapters 2 to 5 are from: "Long as I can see the light" by Creedence Clearwater Revival (Chapter 2), "Thousandfold" by Eluveitie (Chapter 3), "Speed of light" by Iron Maiden (Chapter 4) and "Time" by Pink Floyd (Chapter 5).

A

Motion Blur Evaluation

This Appendix contains additional information to Chapter 3.

A.1. User Evaluation

To evaluate our method in practice, we carried out an evaluation against a workflow of adding motion blur using Photoshop. Users were provided input images and asked to mimic the provided output using both methods. Our focus is on evaluating: i) time to execute the task, ii) user interaction, and iii) satisfaction with the outcome of each method.

It is important to highlight that a comparison among different tools can always be affected by the user expertise. To ensure a fair comparison, we tried to avoid the requirement for a high level of expertise in Photoshop. For this reason, we selected three test scenarios that are relatively easy to segment and decompose in Photoshop and exhibiting a single object to add motion blur to. Further, we provided each user with tutorials of the workflow, which they could reference at any time if needed (see Sec. A.2 and Sec. A.3).

The instructions were given as follow:

> "*Given the input images below, mimic the given result. The process should be done twice: using Photoshop and the proposed tool, in any order. Explore the tools and use prior knowledge. If the desired result cannot be achieved, follow the given tutorials. At the end, indicate whether or not tutorials were used and fill in the questionnaire. Time will be measured during the experiments.*"

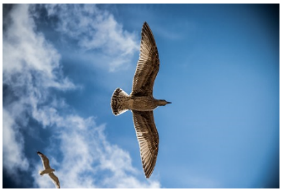
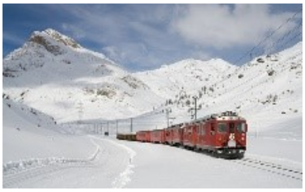
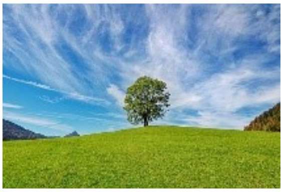
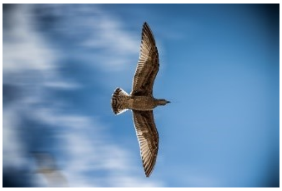
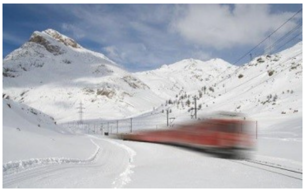
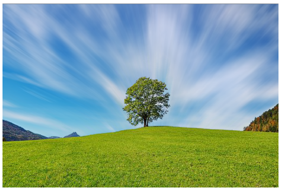
Background motion blur	**Foreground motion blur**	**Multi-directional motion blur**
Input: 	Input: 	Input: 
Result: 	Result: 	Result: 

Table A.1: Inputs (top) and the desired results (bottom) for the user evaluation.

In total, 10 users participated in our experiment. They had varying levels of experience with Photoshop, but have not used our tool before the experiment. We measured the total time to create all three results. At the end of the experiment, they answered the evaluation questionnaire on Sec. A.4. The results are shown in Table A.2 (being our approach *CMB* and Photoshop *PS*). The evaluation metrics were defined as a 5 point scale as follows.

- **Previous Experience:** from 1 = No experience to 5 = Professional.
- **User Interaction:** Very unsatisfied: 1; Unsatisfied: 2; Moderate: 3; Satisfied: 4; Very satisfied: 5.
- **Result Quality:** Very unsatisfied: 1; Unsatisfied: 2; Moderate: 3; Satisfied: 4; Very satisfied: 5.

We notice that the average time to obtain the results is substantially reduced when using our approach. Despite one similar time (user #9), all users drastically improved their total manipulation time, including the single user who reported that performing motion blur in Photoshop provides a better user interaction (#6). On average, users also rated the interaction of our method as higher. From free responses the users provided on the questionnaire, the main reason is the fact that the main tasks of segmenting the content and defining the motion vectors were easier in our approach.

Table A.2 also shows that the quality ratings of our results is more homogeneous, with all users reporting that they are either satisfied or very satisfied with their

User	PS previous Experience	PS time (in minutes)	CMB time (in minutes)	Score on PS Interaction	Score on CMB Interaction	Score on PS Results Quality	Score on CMB Results Quality
1	3.5	10:38	03:30	4	4	4	4
2	2	12:40	07:30	3	4	2	4
3	1	18:10	03:56	1	5	1	5
4	2	19:25	04:37	1	5	2	5
5	2	14:21	02:56	3	5	4	5
6	1	11:59	05:29	5	4	3	4
7	3	08:29	05:20	3	3	3	4
8	3	28:00	07:00	3	4	4	4
9	3	06:43	06:10	4	4	4	4
10	3	23:44	15:43	2	5	3	5
Average	2.35	15:25	06:13	2.9	4.3	3	4.4

Table A.2: Evaluation results. Numerical scores are on a 5 level Likert scale, where higher means better.

results. Upon further evaluation of each produced image, we noticed that imprecise initial segmentation and inpainting on Photoshop caused most artifacts that users complained about, which resulted them to give on average a lower score. 30% of the users stated that they were unsatisfied or very unsatisfied with their results in Photoshop.

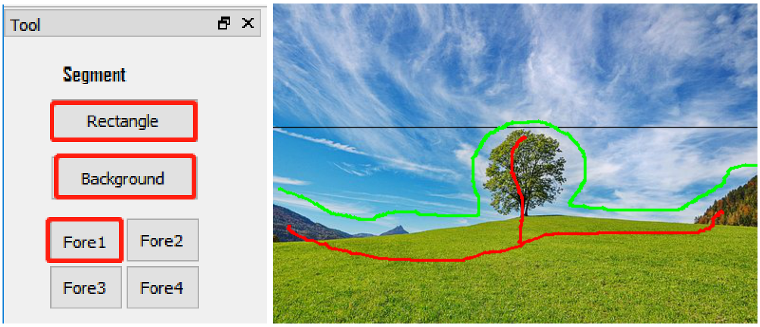
Moreover, we received valuable feedback on improving the overall user experience. Users stated that they would like additional features such as highlighting the segmented areas, keeping the tool selected after interaction, and adding hints to the action buttons. The incorporation of these improvements is left as future work.

While a more elaborate user study with a greater number of participants would be necessary to better evaluate the user interaction and experience, we believe that this preliminary evaluation already shows that the users can benefit from our approach and are able to create convincing results within less time using our method than relying on alternatives.

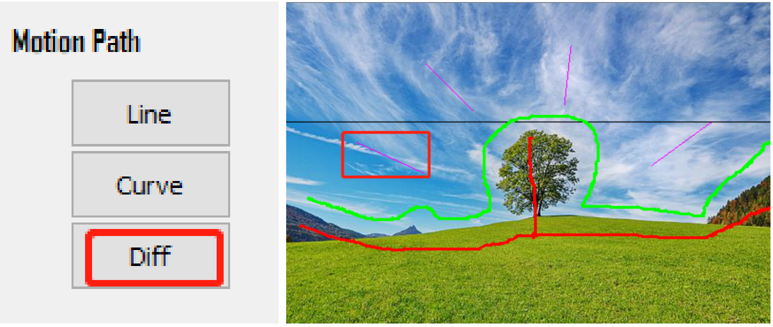
A.2. Controllable Motion Blur Tutorial

1. Open an image: File, Open;

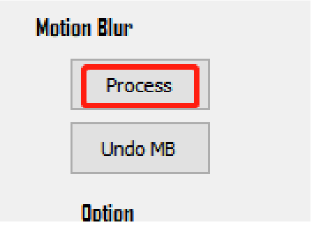
2. To extract the foreground/target object, use a rectangle to draw a frame containing the object. Use scribbles of “background/fore” to improve the segmentation;



3. Select the Motion Path (Line, Curve, or Diff), and draw the motion inside the region to be blurred;



4. Press the button "Process" to obtain the final result;

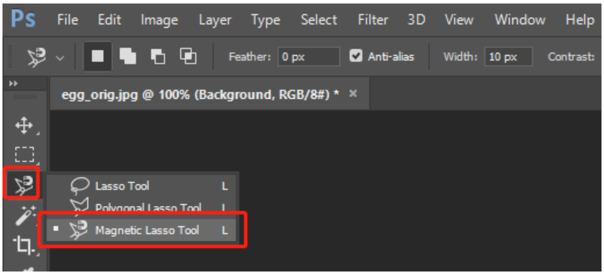


5. Save the resulting image.

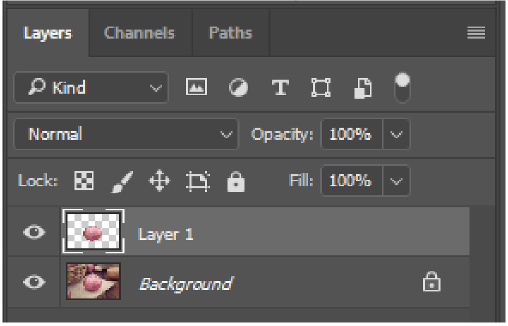
A.3. Photoshop Tutorial

1. Open an image: File, Open;

2. Select Lasso, Polygonal Lasso or Magnetic Lasso tools to extract the foreground/-target object;



3. Copy the extracted object to a new layer by (Ctrl+C) and (Ctrl+V);

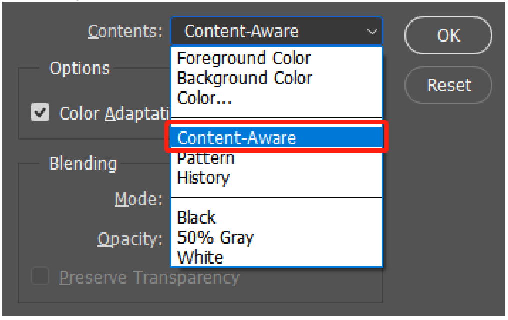


4. If you want to apply motion blur to the foreground (segmented object), go directly to step 8;

5. If the selection disappeared after the copy, press (Ctrl+Shift+D) to recover;

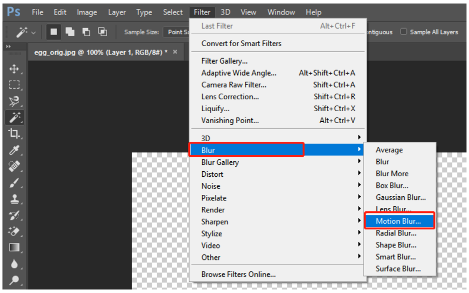
6. Go back to the layer "Background" and press Delete to remove the segmented object;

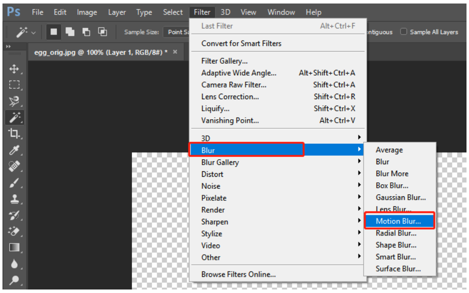
7. A popup should appear asking how to fill the deleted part. Choose Content-Aware;



8. With the layer to be blurred selected, go to Filter, Blur, and Motion Blur;

9. Define the desired Angle and Distance and press OK;





10. If not satisfied with the result, press Ctrl+Z and repeat steps 8-9;

11. Save the resulting image.

A.4. Questionnaire

Fill the questions denoted by *. Other questions are counted/timed by the applicants.

1. Did you use the provided tutorial? *
Photoshop: ______________ **Controllable Motion Blur:** ______________

1a. If tutorial is not used, how many steps does it take to get the desired result?
Photoshop: ______________ **Controllable Motion Blur:** ______________

2. How long does it take to get the desired result?
Photoshop: ______________ **Controllable Motion Blur:** ______________

3. Please score on user interaction/ease to use. *
(Very unsatisfied: 1; Unsatisfied: 2; Moderate: 3; Satisfied: 4; Very satisfied: 5)
Photoshop: ______________ **Controllable Motion Blur:** ______________

Optional: Comments on the process. Which steps do you think is complicated or time consuming?

4. Please rate the perceived quality of the result you obtained. *
(Very unsatisfied: 1; Unsatisfied: 2; Moderate: 3; Satisfied: 4; Very satisfied: 5)
Photoshop: ______________ **Controllable Motion Blur:** ______________

Optional: If your rate is 3 or below, which part of the result do you think is not good enough?

5. From 1 to 5, being 1 = No experience and 5 = Professional, rate your previous experience with Photoshop: *
Your rate:

B

Full Exposure and Shutter Interpolation Code

This Appendix contains additional information to Chapter 4.

B.1. Full Temporal Exposure

We assume as input a video with full temporal exposure, i.e., where the shutter of the recording camera does not close. Standard videos will not always fulfill this requirement, e.g., a 180° shutter only records half of the frame time. Nevertheless, digitally recorded high framerate videos often come close.

Furthermore, if the sequence consists of short exposed frames, one can also rely on optical flow [1–3] from neighboring frames to form virtual intermediate images that fill the gaps. Telleen et al. [4] compute the in-between frames by aligning images and calculating pixel movement to simulate different photo exposures. Jiang et al. [5] train and employ an encoder-decoder network that can create a virtually unlimited number of plausible intermediate frames.

Combining the original video frames (Fig. B.1, top left) with the computed intermediate frames (Fig. B.1, bottom) results in a fully exposed video (Fig. B.1, top right). The frame rate of the fully exposed video dictates the discretization of the shutter functions for which videos may be reconstructed.

For longer exposures, motion blur can occur. Here, a direct interpolation is insufficient to produce frames with a full temporal exposure, as the time intervals of the frames might overlap after adding the intermediate frames. A remedy can

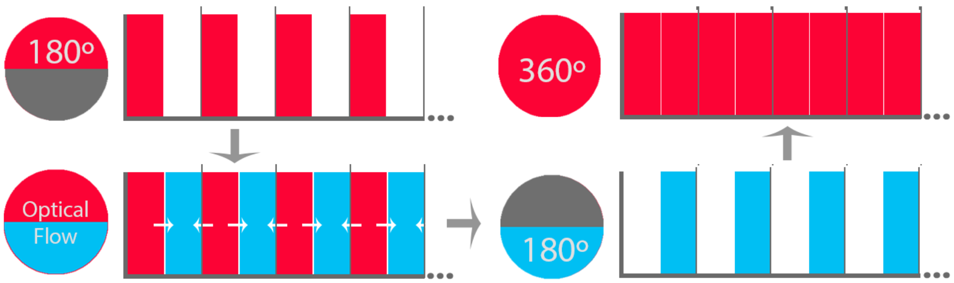


Figure B.1: Top: Shutter open (red): 180° (left) and 360° (right). Bottom: neighboring frames used to create the in-between frames (blue).

be deblurring technique [6, 7] that can transform the long exposure frame into an approximate short exposure.

B.2. Shutter Interpolation Method Implementation

Pseudo code for evaluation of the functions Q, Q^{-1} and S_i^{-1}, introduced in Sec. 4.3.4. We use the same notation and names for variables and functions here and in the text. Arrays of size N (the number of shutter functions) are denoted with a []-postfix. Values relating to the search space optimization are highlighted in color. Our implementation placed the values of S_i in shared memory for better random-access performance.

```
Q( τ, c[], searchBoundsLo[] )
    -> (float, uint[])
{
    // Define the search bounds. Lower bounds are based on previous evaluations of Q. Upper
      bounds always start at the maximum value (T+1). The bounds are updated each iteration
      of the bisection (below).
    los = searchBoundsLo[];
    his = [T+1, ..., T+1];

    // Bisection. The bisection performs a fixed number of steps, halving the remaining search
      interval each iteration. After 9 steps ⇒ $dz = 2^{-11}$, and the "true" $|z^* - z| \leq 2\,dz$.
    bounds = [];
    z = 0.5f, dz = 0.25f;

    for(j = 0; j < SEARCH_ITERATIONS; ++j)
        (val, bounds) = $Q^{-1}$( z, c, los, his );

        if( val ≤ τ )
            z += dz;
            los = bounds; // Adj. lower search bounds
        else
            z -= dz;
            his = bounds; // Adj. upper search bounds
        dz *= .5f;

    // Return result and indices (bounds) at which the value was found. The next call to Q
      (with $\tau \geq$ the current $\tau$) will receive the returned bounds as searchBoundsLo.
    return (z, bounds);
}

$Q^{-1}$( z, c[], searchBoundsLo[], searchBoundsHi[] )
    -> (float, uint[])
{
    res = 0.f;
    bounds = [];

    // $Q^{-1} = \sum_{i=0}^{N} c_i S_i^{-1}$. Evaluate each $S_i^{-1}$ in turn, using the restricted search space. In
      addition, propagate the indices from the search in order to update the search space.
    for( i = 0; i < N; ++i )
        (t,k) = $S_i^{-1}$( z, searchBoundsLo[i],
                searchBoundsHi[i] );

        res += c[i]*t;
        bounds[i] = k;

    return (res, bounds);
}

$S_i^{-1}$( z, searchBoundLo, searchBoundHi )
    -> (float,uint)
{
    // Binary search. Find the index of the first element greater than or equal to z, The search
      is restricted to the range [searchBoundLo, searchBoundHi]. Compare to, e.g., the C++
      standard function std::upper_bound.
    lo = searchBoundLo;
    hi = searchBoundHi;

    while( lo < hi )
        mid = (lo+hi)/2;
        if( !(z < $S_i$[mid]) )    lo = mid+1;
        else                      hi = mid;

    // Linear interpolation. Interpolate between the found value ($\geq z$) and the previous value
      ($< z$). If the search converges to 0, return 0.f (avoid out-of-bounds accesses).
    if( lo == 0 )  return (0.f, 0);

    vhi = $S_i$[lo];
    vlo = $S_i$[lo-1];
    vinterp = (lo-1) + (z-vlo)/(vhi-vlo);

    // Return the linearly interpolated value and the index at which it was found. The latter is
      used to restrict the search space in future searches.
    return (vinterp, lo);
}
```

Usage (e.g., in the body of a fragment or compute shader):

```
// ... (initialization, etc.) ...
boundsLo = [0, ..., 0];
```

```
$Q_\tau$ = $Q_{\tau-1}$ = 0.f;
for( $\tau$ = 1; $\tau \le$ T; ++$\tau$ )
    $Q_{\tau-1}$ = $Q_\tau$;
    ($Q_\tau$,boundsLo) = $Q$( $\tau$/T, $c$, boundsLo );
    contrib$_\tau$ = ($Q_\tau$−$Q_{\tau-1}$) * texelFetch( ... ).rgb;
    // ... (use/accumulate contrib$_\tau$) ...
// ... (finalize and output) ...
```

References

[1] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, *Deepflow: Large displacement optical flow with deep matching,* in *IEEE International Conference on Computer Vision* (2013).

[2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, *Flownet 2.0: Evolution of optical flow estimation with deep networks,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[3] T. Kroeger, R. Timofte, D. Dai, and L. van Gool, *Fast optical flow using dense inverse search,* in *European Conference on Computer Vision* (2016).

[4] J. Telleen, A. Sullivan, J. Yee, O. Wang, P. Gunawardane, I. Collins, and J. Davis, *Synthetic shutter speed imaging,* Computer Graphics Forum **26** (2007).

[5] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, *Super slomo: High quality estimation of multiple intermediate frames for video interpolation,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

[6] W. Ren, J. Pan, X. Cao, and M.-H. Yang, *Video deblurring via semantic segmentation and pixel-wise non-linear kernel,* in *IEEE International Conference on Computer Vision* (2017).

[7] T. Hyun Kim and K. Mu Lee, *Generalized video deblurring for dynamic scenes,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

Curriculum Vitæ

Nestor ZILIOTTO SALAMON

Education

2016–2021 PhD in Computer Science
Delft University of Technology (TU Delft)
Delft, The Netherlands
Thesis: Virtual exposure control for creative image and video editing
Promotor: Prof. dr. E. Eisemann

2013–2015 MSc in Computer Science
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
Thesis: People re-identification in still images through color and group belonging descriptors
Promotor: Prof. dr. S. R. Musse & Dr. J. C. S. J. Junior

2008–2012 BSc in Computer Science
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
Thesis: A sound representation of numerical algorithms complexity
Promotor: Prof. dr. B. R. T. Franciosi

Professional Experience

2020–	Appus HR Analytics & BI Engineer Porto Alegre, Brazil
2016–2020	Delft University of Technology (TU Delft) Researcher Delft, The Netherlands
2015–2016	Pmweb Marketing Cloud Services Senior Programmer Analyst Porto Alegre, Brazil
2011–2015	Hewlett-Packard (HP) Software Designer Porto Alegre, Brazil
2010–2011	Hewlett-Packard (HP) Intern Porto Alegre, Brazil
2008–2010	Pontifical Catholic University of Rio Grande do Sul (PUCRS) Undegraduate Researcher Porto Alegre, Brazil

List of Publications

18. O. Dikken, K. Prakash, B. Roseboom, A. Rubio, S. Ostvik, M. Bueno, **N. Z. Salamon**, R. Bidarra, *A serious game for changing mindsets about loans for home retrofitting*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 12517, (2020)

17. W. Raateland, K. Chronas, T. Wissel, T. Bruyn, B. Konuralp, M. Bueno, **N. Z. Salamon**, R. Bidarra, *A serious game for students to acquire productivity habits*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 12517, (2020)

16. A. Sjölund, M. Straatman, M. van Osch, O. Findra, P. Patil, M. Bueno, **N. Z. Salamon**, R. Bidarra, *Misusing mobile phones to break the ice: the tabletop game Maze Maestro*, International Conference on the Foundations of Digital Games (FDG), (2020)

15. X. Luo, **N. Z. Salamon**, E. Eisemann, *Controllable Motion-Blur Effects in Still Image*, IEEE Transactions on Visualization and Computer Graphics (TVCG), Vol. 26, No. 7 (2020)

14. S. Alaka, M. L. Cunha, J. Vermeer, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *Stimulating ideation in new teams with the mobile game Grapplenauts*, International Journal of Serious Games, Vol. 6, No. 4 (2019)

13. **N. Z. Salamon**, M. Billeter, E. Eisemann, *ShutterApp: Spatio-temporal Exposure Control for Videos*, Computer Graphics Forum (Pacific Graphics), Vol. 38, No. 7 (2019)

12. Y. Appel, Y. Dimitrov, S. Gnodde, N. van Heerden, P. Kools, D. Swaab, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *A serious game to inform young citizens on canal water maintenance*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 11899 (2019)

11. L. Mac an Bhaird, M. Al Owayyed, R. van Driel, H. Jiang, R. A. Johannessen, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *Learning geothermal energy basics with the serious game HotPipe*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 11899 (2019)

10. B. Baas, D. van Peer, J. Gerling, M. Tavasszy, N. Buskulic, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *Loud and Clear: the VR game without visuals* (Best Paper Award), Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 11899 (2019)

9. N. Numan, A. Kolster, N. Hoogerwerf, B. Kreynen, J. Romeijnders, T. H. Huala, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *Star Tag: a superhuman sport to promote physical activity*, IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Superhuman Sports Workshop (2019)

8. M. Kegeleers, R. Bruens, M. Liefaard, **N. Z. Salamon**, R. Bidarra, *IMOVE: A Motion Tracking and Projection Framework for Social Interaction Applications*, International Journal of Computer Games Technology, Vol. 2019 (2019)

7. N. Khalass, G. Zarnomitrou, K. I. Haque, S. Salmi, S. Maulini, T. Linkermann, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *Musicality: A Game to Improve Musical Perception*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 11385 (2019)

6. D. Alderliesten, K. Valečkaitė, **N. Z. Salamon**, J. T. Balint, R. Bidarra, *MainTrain: a serious game on the complexities of rail maintenance*, Proceedings of the Games and Learning Alliance conference (GaLA), LNCS 11385 (2019)

5. M. Kegeleers, S. Miglani, G. M. W. Reichert, **N. Z. Salamon**, J. T. Balint, S. G. Lukosch, R. Bidarra, *Star: superhuman training in augmented reality*, Proceedings of the First Superhuman Sports Design Challenge: First International Symposium on Amplifying Capabilities and Competing in Mixed Realities (2018)

4. X. Luo, **N. Z. Salamon**, E. Eisemann, *Adding Motion Blur to Still Images*, Proceedings of Graphics Interface (GI) (2018)

3. **N. Z. Salamon**, M. Lancelle, E. Eisemann, *Computational Light Painting Using a Virtual Exposure*, Computer Graphics Forum (Eurographics), Vol. 36, No. 2 (2017)

2. **N. Z. Salamon**, J. C. S. J. Junior, S. R. Musse, *A user-based framework for group re-identification in still images*, IEEE International Symposium on Multimedia (ISM) (2015)

1. **N. Z. Salamon**, J. C. S. J. Junior, S. R. Musse, *Seeing the Movement through Sound: Giving Trajectory Information to Visually Impaired People*, Proceedings of Brazilian Symposium on Computer Games and Digital Entertainment (SBGAMES) (2014)