

LUMINATING SCENES: A STUDY OF LIGHT AUTOMATION USING DEEP LEARNING IN VIRTUAL PRODUCTION ENVIRONMENTS

Achilleas Apostolou
Aalborg University Copenhagen
11th of October 2024
aapost22@student.aau.dk

ABSTRACT

This study introduces a tool that automates light management in Virtual Production (VP), enhancing communication between key roles and improving lighting techniques in Unreal Engine 5 (UE5) through deep learning (DL) algorithms. Preliminary interviews were conducted in the form of focus groups with 10 participants attending VP workshops, with varying levels of expertise in cinematography and VP. Results showed that the limitations of a VP set, specifically when it comes to lighting, created a steep learning curve on their role as VP Directors of Photography (DoP), as well as their communication with the Virtual Art Department (VAD). The proposed tool aims to automate a very basic procedure that would normally require extensive planning and collaboration between departments. Using DL algorithms, three models were created: One predicting the light position for an input image, one predicting the light color for an input image, and one predicting both. All three models were created and trained, yet the strong focus point was the color predictor model, as this was easier to test and verify. By creating a short film with this tool, the study evaluates the effectiveness of streamlining the VP process.

1. INTRODUCTION

Emerging innovative technologies have always been used to advance cinematic storytelling and push the boundaries of what was considered possible.

Lighting in cinema is an essential tool for storytelling, creating atmosphere and directing audience focus. Early pioneers such as Georges Méliès and Oscar Rejlander invented methods and create visual illusions that could not exist in real life (1). Additionally, technological advancements are now pushing the boundaries of what is possible on a narrative and psychological level as well as physical.

In this new era of filmmaking, there is now more than ever a vast unexplored area of creative decisions that can help

guide both the creators as well as the audiences, towards a cinematic future of real-time virtual environments and immersive storytelling.

Despite the growing reliance on visual effects (VFX) in modern film production, the VFX artists have become increasingly marginalized, as claimed by Michael Curtin after studying the infamous case of Rhythm and Hues regarding Life of Pi¹ (2). Curtin studies the "Broken Studio Contract System" which results in VFX artists often being paid for work that makes it into the final cut of the film.

The case to be made here, is that Virtual Production (VP) offers a solution to many of the issues caused by the "Broken Studio Contract System". One key benefit is its real-time visualization capabilities, which allow directors, cinematographers, and production teams to see the final visual effects directly on set, during filming, rather than having to wait for post-production. This enables real-time decision-making and collaboration, which reduces the need for costly revisions later in the process (3) (4).

To fully streamline real-time decision-making, it is essential to eliminate all technical and communication bottlenecks, enabling different departments to focus solely on meeting creative criteria.

In this study, the challenges of VP were identified by professional filmmakers by conducting focus group interviews, leading to the development of a tool that aims to explore light management automation in VP, in order to streamline the communication between Director of Photography (DoP) and the Virtual Art Department (VAD), with the help of advanced Machine Learning (ML), and specifically Deep Learning (DL) algorithms.

A short film was then created using this tool to evaluate the impact on the VP pipeline and assess improvements in the overall process.

2. LITERATURE REVIEW

2.1 Technology affecting Storytelling

In traditional cinema, lighting was largely a manual and creatively driven process, where cinematographers used physical light sources to craft mood, texture, and atmosphere (1).

Copyright: © 2024 Achilleas Apostolou, et. al. This is an open-access article distributed under the terms of the [Creative Commons Attribution 3.0 Unported License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

¹ "Life of Pi": dir. Ang Lee, 20th Century Fox, 2012

Dynamic lighting systems are becoming increasingly prominent, reducing the need for static light setups. Dynamic lighting opens new avenues for creative expression, particularly in interactive environments like video games and immersive virtual worlds (5). Additionally, as El-Nasr explains, dynamic lighting not only improves visibility but also sets mood and atmosphere in ways static setups could never achieve (5). This evolution highlights the ever increasing integration between creative and technological possibilities in the film industry.

The Uncanny Valley, a concept proposed by Masahiro Mori in 1970, describes the negative emotional response to almost human-like characters (6). This phenomenon has been observed in various media, including video games and films, where near-photorealistic virtual humans can appear eerie or creepy (7).

Similarly, the use of lighting can create a sense of unease or uncanniness, as can the importance of virtual environments, such as the use of space liminality.

Liminal Spaces, characterized by deviations from expected design norms, can evoke feelings of discomfort and unease in individuals (8). Research suggests that the absence of people and objects in these spaces contributes significantly to the sense of discomfort (9). Artistically, liminal spaces challenge existing perceptions and ways of knowing, creating opportunities for introspection and meaning-making (10). The feeling of ambiguity and heightened reflexivity can leave the audience in a state of emotional uncertainty, fostering engagement with new perspectives (11).

The existence and artistic usage of such effects, causing deep emotional responses, proves that the virtual scene in this new era of filmmaking is more than just a background.

2.2 Current VP Pipelines

The rise of real-time rendering engines and VP techniques have shifted how filmmakers approach lighting, enabling them to adjust light and scene composition on the fly during production (12) (3).

VP is revolutionizing filmmaking by enabling real-time integration of live-action and digital elements, offering new possibilities for narrative scope beyond realism. VP allows filmmakers to visualize and manipulate digital assets during shooting, enhancing production value and reducing cost (4).

Director Yorgos Lanthimos uses VP combined with traditional cinematic techniques, allowing him to paradoxically bend the boundaries of realism. A good example is that of *Poor Things*² in which a world was crafted that feels immersive yet deliberately disorienting (13).

Lanthimos evokes the visual language of early cinema by deliberately employing techniques reminiscent of rear projection and cycloramas, commonly used in the nascent days of film. He does this by intentionally creating environments that appear artificial, with a slightly surreal

and whimsical quality (13).

On the VES Handbook of VP by Zwerman and Okun (1) it is noted that contrary to popular belief that Pixel Pitch is the *Holy Grail* of Lighting for VP, a more accurate assessment would be that the Holy Grail is a combination of things: The light should be good color and quality, match the LED wall and practical light on an actor's face, and be designed with lighting in mind (1) pp.324.

Double Negative (DNEG) offers a method for managing colors on LED walls during in-camera VFX in VP (12). This system allows filmmakers and visual effects artists to create virtual environments accurately reflecting their intended colors, adjusting the imaging process in real time. It accommodates cinematographers' preferences for exposure and white balance, allowing them to focus on storytelling.

The authors note that even though all productions are presumably using similar techniques for on-set light manipulation, there is not enough public information to back that claim.

Furthermore, it is noted that the authors' goal is to enable seamless integration between digital images, LED wall displays, and camera recordings, ensuring accurate color reproduction and flexibility in adjusting exposure and white balance for creative and technical purposes without compromising visual quality. It is important to note that even though they provide a complete solution for image and color adjustments, they do not focus on light direction. Additionally, their solution is tailored to LED-walls and not a broader hardware-agnostic setup.

For DNEG purposes, this is useful, as they create lighting setups and pipelines in their studios that can be calibrated by themselves, based on the needs of each production. However, a tool that aims to work in any set, on any hardware and operated by any technician, should not be limited to specific setups.

2.2.1 Color Framework in UE5

OpenColorIO (OCIO) is an open-source color management system that integrates with ACES (Academy Color Encoding System) in Unreal Engine 5 (UE5) to ensure consistent color reproduction across different platforms and displays (14). OCIO, developed under the Academy Software Foundation, represents a major update with significant engineering efforts to enhance color management capabilities (15).

Understanding the pipeline between OCIO and ACES is vital in handling any lighting setup, as well as colors and materials accurately in UE5. Additionally, since ACES is a suite of technical standards and tools designed to enhance digital image interchange, color management, and long-term archiving in the film and television industry, having a good understanding of it proves beneficial from both a technical and a creative standpoint when working in any part of the VP pipeline. (16).

² "Poor Things", dir. Yorgos Lanthimos, Searchlight Pictures, 2023

2.3 AI powered Computer Graphics in Cinema

Sial et. al. have published a method for estimating the direction and color of a scene's light source from a single image (17). This method utilizes a synthetic dataset with strong shadow effects and a DL architecture trained on this dataset to accurately estimate the light source properties. The approach demonstrates testing on synthetic images and extends to real scenes by proposing a procedure to obtain light positions from the Multi-Illumination dataset, showcasing the model's effectiveness in real-world applications.

The strongest application of this deep regression model is to accurately predict light direction and color in complex scenes for a wide variety of subjects. This model is a complex inception-based encoder-decoder architecture that predicts pan, tilt and RGB values using sin/cos transformations for differences in angles, as well as combine different loss functions for pan, tilt and color estimations, as it was trained using 45.000 synthetic (SID2) images. However, the model was not made to be used in real-time applications. Additionally, it can be assumed that while this model performs well given a wide variety of images, it would be more accurate had it been trained to be used with a specific subject.

This is something that a model trained specifically for VP usage, would prove beneficial for. In VP, the most common subject that is lit up is the actor(s). Thus, a model to specifically predict how light acts on the human face, should be trained on image datasets of human faces, and not that of random objects. In case a VP shoot has different subjects than human characters (e.g. commercial product), new models can be trained and tailored depending on the desired accuracy.

When it comes to training models specifically using the human face as reference in order to calculate dynamic lighting, Meka et al. proposed an innovative technique allowing for photorealistic relighting of the human face under any lighting condition. The authors used a U-Net Convolutional Neural Network (CNN) designed to predict an OLAT (one-light-at-a-time) image corresponding to a specific lighting configuration. This novel technique was developed for applications such as relighting in augmented reality, virtual reality, and visual effects, yet its real-time capabilities make it suitable for use in filmmaking, especially for scenarios requiring dynamic lighting (18).

The future of VFX and Artificial Intelligence (AI) in CGI is rapidly evolving, with AI technologies transforming various aspects of creative industries. AI tools are being integrated into VFX pipelines, enhancing efficiency and realism (19).

ML algorithms, including CNNs, GANs, and RNNs, are being applied across content creation, information analysis, and post-production workflows (20). In their research on AI in the creative industries, N. Anantrasirichai and D. Bull conclude that: *"In the context of creative industries, maximum benefit from AI will be derived where its focus is human centric – where it is designed to augment rather than replace, human creativity"* (20).

Neural style transfer (NST) has evolved to enable real-time video applications on mobile devices. Huang et al. introduced adaptive instance normalization (AdaIN) for arbitrary style transfer in real-time (21). NST has since then been used in a variety of Real-Time applications, ranging from videos (22) to, more recently, game engines. As of Unreal Engine 5.2 Neural Network Engine (NNE) went from an experimental plugin (NNI), to a beta, built-in Unreal Engine Plugin, able to perform real-time Neural Style Transfer and other ML Model operations using the 2017 Microsoft open-source AI ecosystem Open Neural Network Exchange (.ONNX) (23).

3. PRELIMINARY INTERVIEWS

The preliminary interviews were conducted before the development of the prototype. The interviews were conducted in the form of 3 focus groups, containing 3 or 4 participants in each and one facilitator (24) pp. 271-272. Participants were cinematographers varying in experience, expertise and tech-savviness.

All of the interviewees were attending the VP Workshop "ViZARTS" at Aalborg University Copenhagen, and the interviews were held on the last day of the workshop - and thus, after an entire VP experimentation set. Participants were encouraged to share and discuss their experience with VP as cinematographers.

The basis of the conducted interviews were 3 generic questions, followed by follow-up questions and a discussion between the participants. This way, each of those focus groups had a semi-structured interview format. The initial questions asked to all participants were:

1. "What differences did you notice on the lighting setup of a VP set, in comparison to that of a traditional film making set?"
2. "Did you notice any restrictions when lighting a VP Set?"
3. "If you could automate one of the procedures, what would it be? - In relation to the VAD or on set"

The test participants (TP) professional backgrounds in each focus group (FG) were:

- FG1:
 - TP1: Professional cinematographer. The Workshop was their first time experimenting with VP.
 - TP2: Professional cinematographer. Has worked with green screen in the past but not with VP.
 - TP3: Professional cinematographer. Has attended a VP set in the past, but did not have any role in the crew.
- FG2:

- TP1: Professional cinematographer. Has worked with green screen but never on a VP set.
- TP2: Professional cinematographer. Has worked with back-projection before, but never with camera tracking.
- TP3: Professional cinematographer. Has worked once on a green screen VP set in the past.
- FG3:
 - TP1: Semi-professional cinematographer. Has never worked with VP Before.
 - TP2: Professional cinematographer. Has never worked with VP Before.
 - TP3: Professional cinematographer. Has worked with green screen but never on a VP set.
 - TP4: Professional cinematographer. Has never worked with VP Before.

3.1 Results

All interviews were coded and analysed using qualitative content analysis (25). Even though the questions were the same for all groups, the conversations that emerged slightly varied. To be able to categorize the most important topics and points of interest that were discussed, the arguments and quotes were categorized in three categories:

- Lighting in VP
- Green Screen Pipeline vs LED Panel and Rear Projection
- Communication with the VAD

Within these topics, TPs discussed the current state of and their experience with VP, their hopes for future improvements and barriers they encountered during the workshop.

The results and quotes were then color coded and divided based on the results. See appendix 11.1 for the full analysis and all citations.

3.1.1 Positives and Opportunities within VP

As the test participants mostly had limited experience with VP but have been working in the industry for a varying amount of years, the opportunity to experiment with this new technology proved to be an interesting look into the potential future of film making.

A common theme across the interviews when focusing on the positives of VP, was to speak about the general advancements in technology and compare it to traditional film making, as well as previous studio techniques such as using green screens. When talking about lighting in VP, TP1 from FG1 noted:

"I [...] enjoy [...] that you can change the light in the scene and then change the lights in the physical world as well. But I think we still need to learn in which order to do it."

Similarly, TP1 from FG2, while discussing the same subject, went into more detail on the technical aspects of film making in VP, quoting:

"How do you match the projectors contrast with the contrast of the lighting that [...] we made on set [...] that was good fun."

Apart from the technical aspects of VP, a common re-emerging theme that kept concerning the participants, was the communication between the film crew and the VAD. When discussing the cooperation between themselves and the VP technician of that day's set, TP2 from FG1 claimed it was: *"[...] kind of like having a set designer, operator and gaffer in one."*

Most of the participants conceptualized the DoP in VP pipeline by trying to figure out solutions that would make the process more streamlined; some of which delved into the possibility of an automated solution for lighting. TP1 from FG2 explained:

"It it would be really interesting if there was a program that [...] fit in with the way a camera works. [...] like, OK, how much light is coming? Does this room have? How many stops are we looking at?"

On a similar note, TP2 from FG2 continued:

"Because then you can talk about color correcting. Then you can [work] on the look of the final picture while working on the look of the background and kind of complementing each other and [...] then you have to keep the shadows clean or what's the Max, my highlight, where does that fall on the curve that we want for this look and stuff like that might make a lot of sense."

3.1.2 Learning Curve and Communication

A second recurring theme was the participants identifying parts of the process that require for adjustment and learning, usually through improving the crew's communication. TP1 from FG1 wondered:

"If you should change the physical light first or you should change the light in the digital (Virtual Background) [first]."

On a more technical standpoint, TP2 from FG2 appeared to have struggled with the contrast and the exposure whilst attempting to match the subject's shades to the virtual background's colors. They noted:

"The contrast ratio was really hard to get. You had to

kind of bake in the contrast a lot more than you would when you'd be shooting normally. I think [...], normally we had a contrast ratio on the screen of [...] three to four stops. So like the blacks [...] weren't actually black. They were like just four stops under. [...] I found really hard matching was that you didn't have the same kind of roll off in your exposure on your curve."

Similarly, TP1 from FG1 struggled to achieve true color, but they pinpointed it to the communication with the VAD:

"I think you just need to do it a couple of times with the person and learn how they interpret. Like, should I just say numbers or should I just say I want something warmer? Or should I say I wanted A2, 3200 Kelvin? [...] It's like more intense? Or is it the temperature? [...] it's still the same terms we use, it's just a new way of communicating it?", they noted. To what TP2 from the same group continued in saying:

"[...] The most new thing about this [is] the language [used to] communicate with the person about this." To which they added: *"[...]Also [I found confusing] the steps. First you need to find. OK, this is going to be the background you think, but you need to like turn the things and they will find the background after?"*

3.1.3 Limitations and Barriers

Apart from the general interest the participants showed in the experiment and the opportunity they got to understand this new technology, there was also a lot of skepticism on whether VP is at a production-ready state. A common discussion topic throughout all three focus groups was the counter arguments, limitations and requirements from a DoP perspective.

An example of this skepticism was well described by TP1 from FG2, saying:

"I don't know if [VP is usable] where it is right now. I don't think I'd want to use the technology to do that. I think I'd much prefer to be on set in on a location and work with that." to which TP3 added: *"I couldn't contain as much light as there was in the the film like [I would] in real life."*

TP2 from FG2 also had certain thoughts about the limitations of lighting in VP. Regarding the technical aspect, they wondered:

"You have to keep your your exposure quite low to be able to expose for the screen. I find it quite limiting that you had to expose for the screen also because it didn't give off light.", continued by: *"There's one of the big problems is that if you get close to the screen, you have the screen that has like a it doesn't have focus roll off. It's just like everything is in focus right now. [...]"*

However, they agreed that these issues might have been specific to using VP with a back-projection, and could have been solved, or become less apparent when working

with a Volume (LED) Panel.

TP2 from FG2 had another concern regarding the art direction of the experiment:

"In regards to like creating an environment that lighting environment that follows around, it needs to be super flexible to work because there's also a lot of creative decisions in it."

3.2 Conclusion

Through the interviews, data was extracted on the participants' opinion on VP as a technology from a photographic perspective. These proved to divide the common consensus in three major categories:

- 1) The Positives; Opportunities of VP, the similarities with traditional film making and the improvements of the craft through this advanced technology
- 2) The "loose ends" of the DoP process in VP. Adjustments that need to happen in order for this technology to be "production ready" with automated solutions and how the crew can communicate with the VAD seamlessly.
- 3) The negatives, limitations and requirements of The process. What scared them, annoyed them or made them feel that traditional film making is not going away any time soon.

In general, the participants seemed to have attained a good understanding of the technology after being part of each respective workshop. The opportunities that the technology creates were quite clear, with certain limitations bothering them retroactively in how they can be solved and improved. The technical limitations were mostly focused on the inability of the back projection to emit light and they were positive that a LED panel would not have the same restrictions. For the most part they agreed that creatively, VP allows for more creative freedom, as long as the process is streamlined to the extent that the technology does not stand in the way of the artistic expression.

It became apparent that VP as a process seemed to them more intuitive in general than working with a Green Screen, as "you can be more creative" without "giving the footage to someone and hoping for the best" (TP1 from FG1).

Additionally, most of them commented on the new layer of communication created between the DoP, the camera department and the VAD. Most of the participants agreed that whilst within the limits of the workshop the communication was smooth and intuitive, a larger set with a bigger VAD, the communication could exponentially become problematic, and thus there is an immense need for figuring out a "mutual language" that translates the terms between those two drastically different departments.

4. METHODS

Methods, Hardware and Software used for the implementation as well as the creation of the short film:

The base of the implementation of the prototype was created in **Unreal Engine 5.2**, using both the blueprint system and the C++ system, creating custom classes in **Visual Studio 2022**.

Testing the prototype was done using a **Blackmagic Design Cinema Pocket Camera 4K** and an **Elgato Cam-Link**.

The Photoscanned 3D model was created by using **Polycam** on Android for photogrammetry, and **Autodesk Maya** for cleanup.

After the lighting dataset was created in UE5, the models were created, built and trained using **Python 3.11** and Jupyter Notebooks in **Visual Studio Code**.

The movie screenplay was written and edited using the **Microsoft Office suite**.

The "Egg" short film was shot on a **Blackmagic Design Cinema Pocket Camera 4K**, using a **Samyang T1.5, 30mm** lens.

During post production, the editing was done in **Adobe Premiere Pro** and the color grading was done in **Blackmagic Design DaVinci Resolve**. The files were exported from Premiere to DaVinci using the XML format, and after grading they were sent back to premiere as exported LUTs.

5. TECHNICAL IMPLEMENTATION

Considering the interviews of the DoP and specifically their concerns about the constant color-manipulation to correct the visual outcome, there is a need for tools used to streamline the process are needed, since the lighting setup in a VP set is different from that of a traditional film set.

The industry-standard way of achieving such uniformity between physical and virtual worlds is manually adjusting them in every shot or using Digital Multiplex (DMX) lights to automate part of the procedure. The constraint that comes with current automation techniques using DMX lights is that they light up the set based on the Virtual Scene lighting. While this gives a lot of flexibility to the VAD, it limits the DoP in the way they choose to emit light on the actors. Building a system which would achieve the opposite of that procedure (i.e. having the DoP set up the physical lights based on the actors and camera and having the virtual scene adjust to it automatically) would be useful for many low or high budget productions. It could be argued that a tool like this, unlike the DNEG system (12), could be completely hardware-agnostic and perform well in many different lighting scenarios.

5.1 First Iteration

The way that the original implementation was set up, the process involved two main parts: the Computer Vision phase and the Look-Up Table (LUT) generation phase.

5.1.1 Computer Vision Phase

5.1.1.1 Camera Footage Input

Began by capturing footage in Media Stream using the Blackmagic Cinema Pocket Camera 4K yet any camera could be used in this system. This footage served as the input for subsequent processes. To do this in real time, additional hardware was needed, such as the BlackMagic Design Decklink, or in this case the Elgato CamLink 4K.

5.1.1.2 ML-Based Segmentation

Utilizing ML algorithms within UE5 to segment the camera footage and specifically the OpenCV library for UE5.

5.1.1.3 Color Profile Extraction

Extracted color profiles from the segmented elements. Depending on the depth of information that one wants to store, the more complex it could prove to be. For instance, trying to account for other visual information such as the NITS of the LED Panel would change the way that the color extraction is performed. In this situation, only the Color profile and scheme were

5.1.2 LUT Generation Phase

According to Selan (26), three-dimensional LUT enable for real-time color processing of high-resolution imagery and could be considered "production-ready" even at the moment of publishing. When it comes to LUTs for Color Grading Unreal Engine uses 3-Dimensional LUTs (27), by using built-in drag and drop techniques that make it "easy" for users to modify on demand (27).

5.1.2.1 LUT Generation

Based on the extracted color profiles, generate a LUT using dedicated algorithms or plugins. The LUT serves as a mapping between the current color representation and the desired color correction.

5.1.2.2 Applying LUT to Virtual Scene

Implement the generated LUT within UE5 to adjust the virtual scene's lighting and post-processing effects. This ensures that the virtual elements seamlessly integrate with the real-world footage in terms of color accuracy. Unreal has a drag-n-drop LUT import system that can even be adjusted on runtime, by manipulating the default LUT into the new changed color values.

5.1.2.3 Real-Time Color Correction

Enable real-time color correction within UE5, allowing DoP and VP technicians to visualize and adjust the scene's colors on the fly.

It is important to note that this iteration could prove beneficial for a wide variety of applications. The existence of LUT Generators is important when it comes to color grading in any aspect of real-time computer graphics. However, LUTs are applied to post-process materials and post-process volumes, as they transform the colors of the entire

Figure 1. Custom LUT Generator

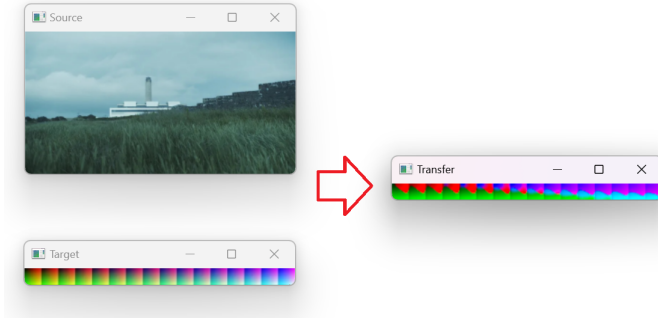
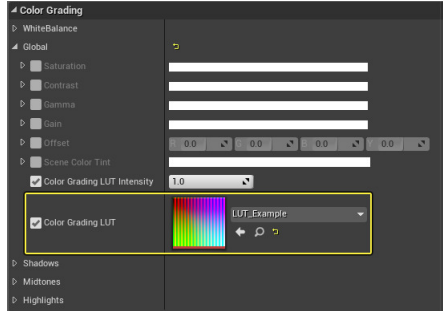


Figure 2. Applying an LUT in UE5.2



scene by remapping them according to pre-defined color look-up textures.

Because of this, LUT's are part of the post-processing pipeline to adjust the **final** appearance of the rendered image. Hence, it is something that is very useful for altering the final rendered image, but that can be an issue when it comes to a VP setup that is aiming to achieve Final Pixel in Camera. The reason for that is that the VP Background is not supposed to have a "cinematic" look similar to the desired final result; yet to simulate how the background would be in a real-life setup. Zwerman and Okun (2024) describe:

"It is important to recall that the LED wall [...] is supposed to produce light similar to the real world. The LED wall is not a large movie screen. (1), pp. 302. Zwerman and Okun go on explaining how it would be a mistake to apply non-linear curves such as film emulation on the LED wall, resulting to "double LUTing". This effect derives from the fact that any artistic look applied on the LED wall, would also be applied to the output of the camera.

5.2 Second Iteration

The basis of the second iteration was a Virtual Production oriented approach, creating a framework that calculates light direction and estimates color values from every frame (or whenever the user chooses to enable it). Similar approaches have been implemented in the past; Sial et. al. created a similar model that uses Deep Regression to calculate light direction and estimate color with high accuracy in a wide variety of input images and scene setups (17).

Since this prototype's aim is to be used solely in VP Environments with a strong focus on lighting on actors' faces,

the way the model was trained was handled in a less abstract manner, while still being trained on a synthetic image dataset that was generated.

The models created for this tool were made using deep learning, specifically a CNN, to perform deep regression, predicting continuous values in the form of positional coordinates (X, Y, Z) and color values (R, G, B) from images. The CNN trained on complex patterns from image data through supervised learning, with the goal of minimizing the error between predicted and actual values using a loss function, such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). Additionally, normalization was ensured in order to balance predictions.

Three models have been created for two different uses:

1. Light Direction Predictor,
2. Color Estimation,
3. Combination of both.

The models were trained by using 3 different synthetic image datasets, for each of the three individual models. The first and second model were trained on 1000 individual images each, whereas the third model was trained on a merged dataset containing the first two, plus another 1000 images (3000 in total). It is important to note that even though these datasets were big enough to train the models for this prototype with acceptable results, in order for a system like this to be used in an actual production and in a variety of different lighting situations the datasets would need to be considerably larger, as well as trained on a variety of faces and assets.

The images contained in the datasets have been captured from the UE5 view port, simulating a media stream, by using a photo scanned face model. Then, a random light was placed in the scene, and captured the shadows that were casted on the photoscanned face, as well as the light's current position and color. These were stored as X, Y, Z float values for the position, and R, G, B float values for the color.

After training the models, they were exported as .onnx files, to be used in Unreal Engine using Unreal's Neural Network Inference built-in plugin.

Additionally, a more lightweight version of the color predictor has been created, that is able to change the color of an Unreal Engine object (whether Material or Light) by reading the Render Texture pixels directly on screen. This process, whilst less accurate as it has to manually adjust for Color Profiles and Difference in objects in the scene, is much lighter to run, as it does not incorporate the real-time usage of any Neural Network or any other Computer Vision directed approach.

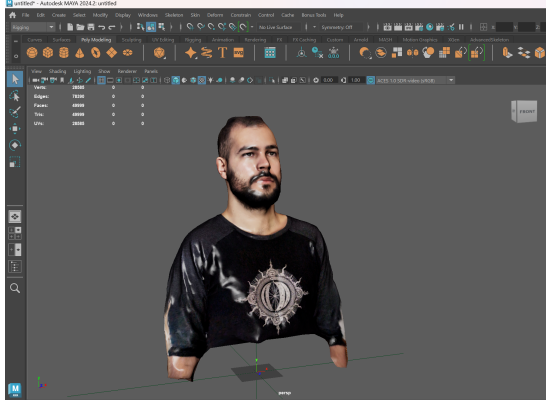
5.2.1 Model Training Process

5.2.1.1 Photogrammetry

An important aspect of training this light predictor with a small dataset in order to produce accurate results, would be to emulate the light on the actors' face and train it accordingly. This predictor would not be highly accurate

when it comes to different objects and setups, but in a VP environment, the model should cover most of the cases in which it is needed. As shown in the interviews, the most common concern of the DoPs was how to dynamically light an actor's face. It can be assumed that this is a strong point of focus, as for static objects and scenes or in any scenario that the acting part is not affecting the scene, the lighting setup should also remain relatively unchanged.

Figure 3. Photoscanned model in Autodesk Maya 2024.2



Polycam for Android was used to create the 3D model. The 3D asset was then cleaned and retopologized in Autodesk Maya, in order to have more accurate real-time shadows in UE5. It was not important to optimize it for skinned and animated use, as long as the shadows casted on the asset were accurate and without any artifacts.

5.2.1.2 LightPosition Dataset Creation

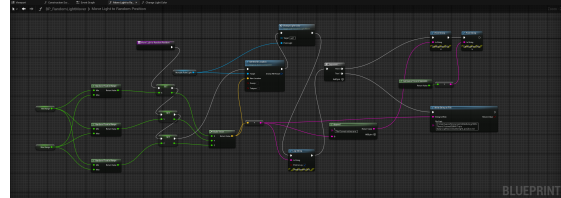
In order for the dataset to be functional and be used for training an ML model, there was a need to create an automated way to produce a large number of pictures with a randomized lighting setup, and their corresponding lighting information. When it comes to the Light Position predictor, a blueprint in UE5 was created to change the light setup, capture the current viewport frame of how that lighting setup affect the 3D Asset (Actor's face) and print on a text file the corresponding Position of the Spotlight object for that given frame.

The logic of the blueprint was implemented as follows (also seen in figure 4):

- A random position is being generated, by creating three float variables (X, Y, Z) and assigning a random value to each, ranging from -150.0 to 150.0. For the use of this prototype, a span of 3 cubic meters was enough, but the range of the values could change depending on the dimensions of the target studio.
- The three float values are being casted to a 3-dimensional position vector.
- The Set World Location node is used to apply the newly generated vector to the Movable Point Light in the scene. The generated location is passed as the New Location input to move the light to the new random position.

- The generated X, Y, and Z values are printed to the screen using Print String for debugging purposes. The position values are also saved to a text file (light-position.txt) by appending them into a string and writing it to the file.

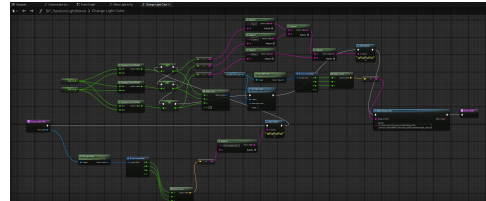
Figure 4. UE5 Blueprint to change Light Position and Print the values.



5.2.1.3 LightColor Dataset Creation

In a similar manner, the Blueprint to modify and store the light color's attributes also randomized values, stored them in a light-color.txt file and synchronised the output to the viewport capture that happened every one second. Both blueprints share a similar structure: They generate random values (RGB for color or X, Y, Z for position). They apply these random values to a light component. They log the results using Print String and write them to external text files.

Figure 5. UE5 Blueprint to change Light Color and Print the values.



5.2.1.4 LightPosition Model Configuration

This model was designed to predict the position of the light source based on an input image. Specifically, it aimed to estimate the X, Y, and Z coordinates of the light's location in 3D space. The model was tasked with taking an image as input and predicting the light color in terms of X, Y, and Z values.

The model was built in Visual Studio Code using Python and Jupyter Notebooks as well as the TensorFlow libraries.

Description of model architecture as seen in figure 11.4:

- Input Layer: The input to the model is an image of size 256x256 with 3 color channels (RGB). All input images were captured at a Full High Definition (FHD) in UE5, and are downsampled to 256x256 during the model training process, for easier training and the ability to handle bigger datasets.
- Convolutional Layers: The model uses 3 sets of convolutional layers (Conv2D), each followed by a MaxPooling2D layer. These layers are designed to:

- Extract important visual features from the input image.
 - Gradually reduce the spatial dimensions (downsample the image) while maintaining the most important information.
- **Flattening Layer:** After the convolutional layers, the output is flattened into a one-dimensional vector to be fed into the fully connected (Dense) layers.
 - **Dense Layers:** The model uses two fully connected (Dense) layers with 128 and 64 neurons respectively, both activated by a Rectified Linear Unit (ReLU). These layers further process the extracted features from the image to learn non-linear combinations of the features.
 - **Output Layer:** The final output layer consists of 3 neurons with no activation function. Each neuron corresponds to one of the X, Y, or Z color values.

In the process of training the model, it was compiled with the Adam optimizer and the mean squared error (MSE) loss function. The model was trained for 20 epochs³ using the image dataset and its corresponding X, Y, Z (Light Position) values for each.

After training, the model was able to be used to predict the X, Y, Z color values for a new image. The model was able to be tested both inside of Visual Studio Code by inputting a new image with an unknown lighting setup for predictions. After the training, the model was exported as an .onnx file to be used in UE5 with the NNE plugin.

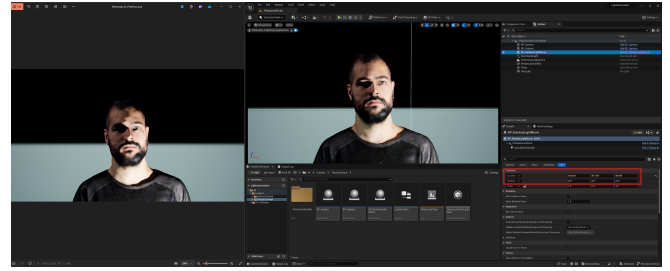


Figure 6. Result of Light Position Predictor

Pseudocode for TrainLightPosition model:

```
1 # Imports
2 import tensorflow as tf, os, numpy as np,
  matplotlib.pyplot as plt
3 from PIL import Image
4
5 # Dataset class
6 class ImageDataset:
7     def __init__(self, folder, labels_file):
8         self.image_filenames =
9             generate_filenames()
10            self.labels = load_labels(labels_file)
11
12     def parse_label(self, label_str): # Parse X
13         , Y, Z (position)
14         return np.array([float(v.split('=')[1])
15             for v in label_str.split()])
16
17     def load_image(self, filepath): # Load,
18         resize (256x256), normalize
19         return np.array(Image.open(filepath).
20             resize((256, 256))) / 255.0
21
22 # Dataset and model setup
23 dataset = ImageDataset(image_folder, labels_file)
24 tf_dataset = create_tf_dataset(dataset,
25     batch_size=32)
26
27 # Model: Conv2D + MaxPooling + Dense
28 model = create_model()
29 model.compile(optimizer='adam', loss='mse',
30     metrics=['mae'])
31 history = model.fit(tf_dataset, epochs=20)
32
33 # Predict and evaluate
34 predicted = model.predict(new_image)
35 loss, mae = model.evaluate(tf_dataset)
```

The Mean Absolute Error of this model is 7.198.

For the full Jupyter Notebook python file, along with the output logs and predictions, see Appendix 11.3

5.2.1.5 LightColor Model Configuration

This model was designed to predict the color of a light source based on an input image. The goal was to train the network to estimate three key parameters: the X, Y, and Z components of the color of the light (represented in a color space like RGB). Similarly to LightPosition, the model is trained with an image dataset, and its corresponding X,Y,Z values. After training, the model takes a new image as input and predicts three new values. These values could correspond to some representation of color such as RGB in this case.

³ Each complete pass of the entire training dataset through the network, during which the model's weights are updated based on the error between predicted and actual outputs.

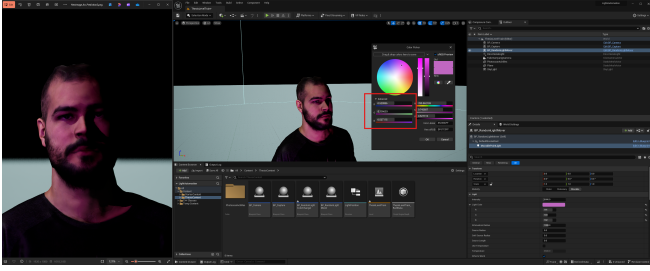


Figure 7. Result of Light Color Predictor

Pseudocode for the TrainLightColor Model:

```

1 # Imports
2 import tensorflow as tf, os, numpy as np,
  matplotlib.pyplot as plt
3 from PIL import Image
4
5 # Dataset class
6 class ImageDataset:
7     def __init__(self, folder, labels_file):
8         self.image_filenames =
  generate_filenames()
9         self.labels = load_labels(labels_file)
10
11     def parse_label(self, label_str): # Parse X
12         , Y, Z (color)
13         return np.array([float(v.split('=')[1])
14         for v in label_str.split()])
15
16     def load_image(self, filepath): # Load,
17         resize (256x256), normalize
18         return np.array(Image.open(filepath).
19         resize((256, 256))) / 255.0
20
21 # Dataset and model setup
22 dataset = ImageDataset(image_folder, labels_file
23 )
24 tf_dataset = create_tf_dataset(dataset,
25     batch_size=32)
26
27 # Model: Conv2D + MaxPooling + Dense
28 model = create_model()
29 model.compile(optimizer='adam', loss='mse',
30     metrics=['mae'])
31 history = model.fit(tf_dataset, epochs=20)
32
33 # Predict and evaluate
34 predicted = model.predict(new_image)
35 loss, mae = model.evaluate(tf_dataset)

```

The Mean Absolute Error for the Color variation of the model, built for 20 epochs, is 0.124 (as color values are normalized to range between 0.0 and 1.0), by dividing the absolute color number (range between 0.0 and 255.0) by the number 255.

5.2.1.6 LightPositionColor Configuration

A third model was created that was trained to calculate position and color given a single image. The main difference between the combined model to the previous two, is that it was trained using all the datasets, and the fact that it has 6 labels per image (X, Y, Z, R, G, B float values). In addition, 3 (R, G, B) of the 6 labels were normalized to range between 0.0 and 1.0.

The datasets were merged by training the model on each of them and concatenating the images, position-values and color-values to create a single combined dataset. When training using the Color dataset, the position labels were filed with the fixed $[X,Y,Z] = (150.0, 150.0, 150.0)$ values, since the color dataset was created with the light hav-

ing that fixed position. Similarly, when training the combined model using the position dataset, the color values were fixed at $[R, G, B] = (1.0, 1.0, 1.0)$ as the position dataset was created with a fixed white light.

The architecture of the combined model can be seen in the appendix 11.4.

5.2.2 Model Testing Process

In figure 6, on the left is the input test image, and on the right is the result of the light position that the model predicted in UE5. It safe to assume that the difference in lighting is of little to no importance, considering that in a real world scenario the position of the lighting would only affect the virtual background, based on the physical changes of lighting.

In figure 7, on the left is the input test image, and on the right is the result of the light color that the model predicted in UE5. Similarly to figure 6, the difference would not be noticeable.

It is important to note that the only reason the 3D model exists in the right image in the scene in figures 7 and 6, is to see how well it matches the reference. In a real application, the scene on the right would only consist of the target Virtual Background as well as an NDisplay renderer. The light would take the same color and Position and would affect the scene as it should, based on the physical light on the face of the actor.

Additionally, even though both the Position and the Color predictors have been implemented, in this prototype the focus is on the Color Predictor, as it was the one planned to be used during the production of the Short Film "The Egg".

5.2.2.1 Real Time Color Picker

A different way to test would be using a Color Picking in Real Time setup. This setup could be used either by using the pre-trained ONNX models, or by creating a self-adjusting system from an input stream directly connected to a UE5 material or light's attributes.

As seen in figures 8 and 9, a media stream feeding the color picker images at runtime changes directly the R, G, B color values of the virtual lamp. This, is exactly the same technique used in the one-shot scene transition of "The Egg" short film.

5.2.2.2 Models as ONNX files

To export the trained models as .ONNX files, in order to be used with the NNE plugin in UE5, the tf2onnx and onnxruntime libraries for python were used, as seen in figure 10.

6. SHORT FILM IMPLEMENTATION

In order to test and evaluate the implementation of the prototype, a short film production was conducted. The crew consisted of current and former students of Aalborg University Copenhagen as well as external film makers with a strong interest in VP.

An important environmental transition shot came to life using the prototype to dynamically change the lighting. The DoP of the movie was asked to choose a way to modify the physical lighting during the scene transition which was shot using a one-shot take. The virtual light adjustment tool was then used for adjusting the background lighting setup accordingly.

6.1 Original Series and Backstory

The Short Film created was named "To avgo"; Greek for "The Egg". The film is a remake of a scene from the Greek short film series Ekeinos kai Ekeinos (Him and the Other), released in 1972 which marked a pivotal moment in Greek cinema, written by Kostas Mourselas. Released during the military junta, it stood out for its subtle yet critical political commentary. The series used dark humor and absurdist elements to highlight themes of oppression and resistance, which resonated with the Greek audience during a time of censorship and political turmoil (28).

The series was a ground-breaking look into the Greek society of the early 1970's with the characters contemplating whether they should accept, embrace and adopt the reality of an emerging "modern and new" world as described by Elissavet Georgiadou (28).

The series' influence was profound in both its artistic style and its thematic content, challenging the norms of traditional Greek cinema by adopting a more abstract and allegorical style. It helped push Greek cinema towards a more politically conscious narrative, while also expanding the boundaries of the absurd and surreal as storytelling tools. Its minimalist aesthetic also allowed lighting to take on a symbolic role, using shadows and stark contrasts to emphasize the bleakness of the characters' situations and the broader political context. By doing so, Ekeinos kai Ekeinos not only influenced subsequent generations of Greek filmmakers but also contributed to a growing wave of European political cinema that challenged authoritarianism through art (28).

6.2 "Egg" Short Film

6.2.1 Short Film Adaptation

The script has been modified to match the same fears and understandings of people in 1972, with those of the people today. "The Egg" takes the form of an Extended Reality (XR) device, and the "new modern world" becomes that of a potentially digital, parallel reality. The characters from "Ekeinos kai Ekeinos" would find themselves in 2024 to contemplate the dangers and fears of this new reality. Implemented using advanced VP techniques and an ML-powered advanced media technology workflow, the script results into being a meta-cinematic experience



Figure 8. Color Picking in Real Time 1

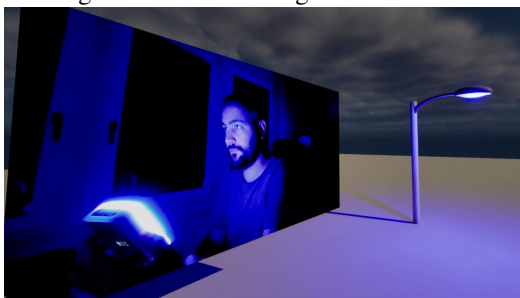


Figure 9. Color Picking in Real Time 2

```
spec = (tf.TensorSpec([None, 256, 256, 3], tf.float32, name="input"),)
onnx_model_path = "light_posCol_model.onnx"
# Convert the model to ONNX format
model_proto, _ = tf2onnx.convert.from_keras(model, input_signature=spec, output_path=onnx_model_path)
```

Figure 10. Simple python block to convert a trained TensorFlow model to an .ONNX file



Figure 11. Behind the Scenes 1



Figure 12. Behind the Scenes 2

that delves into criticizing the advancement of technology, whilst incorporating said advanced technologies in a way that deems it to be self-actualized.

What stayed true to the original was the general abstract and allegorical aesthetic, as well as parts of the dialogue between the two main protagonists, who have a more vague dynamic, compared to the original, yet still depict a long-lasting friendship.

6.2.2 Short Film Shooting

The shootings of the short film took place in August 2024 at Base.M studio in Aalborg University Copenhagen. The crew consisted of a team of 10 people whilst many more helped with their contribution before, during and after the shooting days.

The short film was filmed entirely on a VP set. The virtual background was being rendered in real-time using Unreal Engine 5.2 and the camera tracking was performed using an outside-in system, as described by Zwerman and Okun(1) pp.289, with Vive trackers and base stations positioned around the environment.

6.2.3 Practical use of Light Automation

In order to evaluate the Light Automation prototype created for this study in an actual production, the shotlist was made in a way to incorporate difference lighting setups, as well as push the limits of dynamic lighting in a VP environment. The film consists of two virtual scenes:

- An undefined western virtual city in 2024. The lighting is cold, and there are people and cars in the background, showcasing the feeling of a busy, lively city.
- The inside of a huge shell that vaguely resembles the inside of an egg, but could be mistaken for an eerie,

dark environment that leaves the viewer with a feeling of solitude and isolation. There are no dynamic elements in the background and the characters feel detached from reality. In order to enhance that feeling of loneliness that the characters may be affected with, the scene is made in a way that assimilates a liminal space, as mentioned in 2.1.

The movie follows the two characters, through their conversation, in their journey from the city environment, to the inside of the egg and back at the streets of the city. The transitions happened seamlessly, without the dialogue ever stopping, in a way that the characters were never seemingly aware of the environment changing. Thus, the lighting needed to play an important role in the transition between the scenes, in order for the visual narrative to follow along their arcs.

The decision to make the transition between the two scenes take place during a one-shot, was in order to experiment with dynamic lighting, both in the physical and the virtual setup. The Automatic lighting prototype was incorporated during shooting for that specific shot, in order to evaluate the usability and results of such system. As seen in figure 13, the actor's face at the beginning of the shot was lit by the original physical lighting setup, whilst on 14 the actor's face has changed color, as the DoP chose to physically light up his face mid-shot, and following that the background lighting setup automatically adjusted based on the color of the actor's skin as seen on 15.

6.2.4 Results

The final draft release of this short film ended up being 6 minutes and 33 seconds in length.

Semantically, it follows the dilemma of two longtime friends on whether they should give up their daily routine for a more polished, seemingly well refined and stress-free reality by giving up on each other.

"The egg" is an abstract and allegorical term, used to describe this "new reality" that the characters are considering entering. This new reality could be perceived as a plethora of conditions and states of being. The movie hints into the idea of the new reality being a completely virtual, XR world, in which there is no "room" for companionship and solidarity.



Figure 13. Actor before light transition (One-shot).



Figure 14. Actor after light transition (One-shot).



Figure 15. Background adjusting to light transition (One-shot).



Figure 16. Character1.

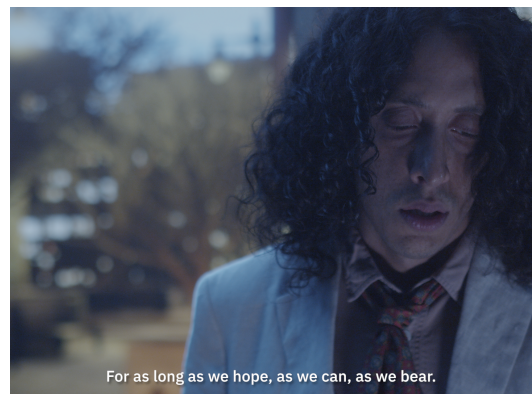


Figure 17. Character2.

Character 1 - seen in figure 16 - is a friendly and sensitive man who is not easily convinced about changes in his routine, and certainly not in letting go of anything he holds dear.

Character 2 - seen in figure 17 - is an ambitious and head-strong man who aims to change his life for the better, whatever that entails. His only weak spot, is for his best friend, as he indulges his every whim.

The two characters are eager to discover new beginnings, in order to be part of something bigger. However, their long lasting friendship is either the thing that will push them into the unknown, or keep them away from an isolated circumstance that neither of them can escape.

The Final Draft of the film, released for this paper, can be watched in the link that can be found in the Appendix 11.5.

7. DISCUSSION

According to the VES handbook of VP *Visual effects have constantly evolved since their perception, [...] yet today the opportunity has never been more significant for visual effects artists [as] VP with real-time animation **changes the game*** (1).

However, advanced real-time workflows are designed to fit into a non-siloed production pipeline (1). Thus, the spirit of collaboration and spontaneity is of utmost importance. Real-Time VP tools can assist this effort and

empower filmmakers and technicians to deliver their best work.

7.1 Automating Light for VP

One of the ways that VP has revolutionized filmmaking is the flexibility and real-time control over lighting and environments. It is safe to assume that filmmakers - and especially producers - who created films before the rise of VP would envy the ability of planning a 10-hour shooting day for a single Magic Hour ⁴ Scene.

However, the flexibility that VP brings to modern film sets, makes the collaboration between departments as important as ever. Communication between filmmakers can prove to be difficult, especially when they come from different departments and fields, such as the DoP and VAD. The role of VP technologists is to eliminate any technical "bottlenecks" in the production pipeline, allowing the artists focus exclusively on Creative decisions.

Lighting, one of the most critical aspects of filmmaking, becomes significantly more complex in VP, where digital and physical elements must be seamlessly integrated. The cinematographers that took part in the focus groups expressed the difficulties of achieving consistent lighting across virtual and physical environments and highlighted how automated tools can streamline these workflows. As described in 3.1.1, participants were excited about the possibility of doing color correction in real-time. This would result in an iterative creative process of adjusting background and foreground until they compliment each other. However, participants also noted that there is still a learning curve and a need to refine communication between the DoP and the VAD as referred to in 3.1.2. It is important to remember that innovative automation tools are created to assist the artists, instead of limit them. Certain participants mentioned that they would prefer to be on set on a location, than trying to simulate "real life" light in a VP studio. 3.1.3

AI and especially DL systems, can open up new possibilities for automating tasks that were traditionally performed manually, such as lighting adjustments. In this study, a tool was developed that addresses a key challenge in VP - the real-time management of lighting setups - by utilizing computer vision and ML models. The results, both from the short film created (The Egg) and feedback from cinematographers, suggest that such automation can improve efficiency on set and provide new creative possibilities, if used carefully.

7.2 Practical Application

The prototype was put to the test during the production of the short film "The Egg", a modern representation of the Greek short film series "Ekeinos kai Ekeinos". The decision to adapt a film that originally had a minimalist

⁴ i.e. the time just after sunset and just before sunrise, producing warm colors of gold, pink, and blue.

and politically charged aesthetic allowed the study to explore the symbolic role of lighting in more abstract terms. Lighting was not only used to set the mood, but also to emphasize the thematic tension between reality and illusion.

The automated lighting tool proved particularly useful during the dynamic transitions between scenes, especially in a one-shot sequence that required continuous adjustments to both physical and virtual lighting. This experiment demonstrated that ML-driven automation could effectively support real-time lighting changes, particularly in complex shooting environments like VP, where traditional lighting setups might slow down the production process or create inconsistencies.

7.3 Limitations

Despite the clear advantages of using light automation in VP, several challenges remain. The focus group interviews identified the need for better communication between the film crew and the VAD, especially in terms of understanding the technical aspects of light calibration across digital and physical sets. Cinematographers often struggled with matching the contrast of virtual environments with that of physical lighting, a task that requires both artistic skill and technical expertise. This points to a broader challenge in VP, where the boundaries between digital and physical filmmaking processes are still being navigated.

Additionally, as much as this system is designed to be hardware-agnostic, the current state of VP does not allow for universal solutions when it comes to lighting, as setups vary vastly. An LED volume setup, a back projection setup and even a VP using green-screen setup, all need to be treated in unique and individually calibrated ways. Even if the tool managed to work perfectly and predict light color and position with maximum accuracy, the real-world setups would vary. For instance, an LED volume acts as a physical light source as well as a background. Thus, the physical lighting would be affected by the adjusted virtual lighting for every cycle, resulting in a self-adjusting infinite loop. Presumably, the best solution for a problem like this would be to integrate in a system like this physical light manipulation techniques such as DMX light adjusting system. This way, the user would have complete control over the lighting setup of both virtual and physical assets.

Furthermore, regardless of the success rate of any automated light manipulation system, the focus groups participants expressed skepticism about relying entirely on automation for creative decisions. Cinematographers emphasized that while such tools can enhance the workflow, they should augment rather than replace human intuition and creativity. This echoes broader concerns in the industry about the role of AI in creative fields.

7.4 The future of Filmmaking

The results of this study suggest that ML powered light automation could play an increasingly important role in the filmmaking, especially in VP. The development of tools that can streamline parts of the process and focus on creative decisions will allow filmmakers to push the boundaries of what is possible in terms of dynamic storytelling as well as immersive environments. However, for technologies like this to be fully adopted in the industry, there is a need for further refinement in terms of purposeful automation and user experience (UX) design, to assist creatives who might not have extensive technical knowledge.

Additionally, as the tool proposed in this study was a prototype, the models used were trained for a very specific setup. A tool like this would need to be much better trained on a way larger image dataset. As described in 5.2, the model would need to be trained on a larger dataset with a bigger variety of inputs. H. Sial et al. created a deep-learning regression system that predicted with high success rate the light direction and color given a single image as input (17). Their system was not made to run in real-time, but was quite successful with a large variety of input images, such as individual assets and complex compositions. To train their model, they used a complex architecture as well as a 45000 images-long synthetic image dataset. In order to simulate such a variety of different scenarios in this study's models, it would be needed to match the length of their dataset as well as train it using different actors and compositions. A possibility would be to train the model using EpicGames' MetaHumans⁵ instead of photoscanned models in order to "feed" it with a larger synthetic image dataset.

A different direction would be to train the model for every production, specifically using the actors and assets that would take part in it. Subsequently, this would result in extra costs and planning, as the actors would need to spend extra time before the shooting days for scanning. However, virtual productions are increasingly willing to invest more time and money in pre-production so that they can reduce costs and uncertainties during the shooting and post-production phases.

Moreover, as filmmakers continue to explore the artistic possibilities of VP, there is a growing need for tools that can balance the technical demands of virtual environments with the creative freedom that is essential to storytelling. Additionally, for any new technology to be deemed future-proof, it needs not to collide with traditional filmmaking methods in any way. In this context, automated lighting tools have the potential to not only streamline production processes but also open up new avenues for creative expression. The combination of ML and VP offers a glimpse into a future where filmmakers can experiment with complex lighting setups and dynamic environments

⁵ Framework that allows to create fully customizable photorealistic digital humans and use in UE5.

in real-time, without compromising on the artistic quality of their work

8. CONCLUSION

In conclusion, this study demonstrates the potential for integrating ML techniques into VP environments to address key challenges in light automation. The tool developed offers promising results, though further refinements are needed to address the challenges identified by the cinematographers involved. Showcasing the technology to a larger variety of cinematographers could prove beneficial to better understand its general applicability.

The experimental short film shows that in practice, while automation can significantly enhance efficiency and consistency, it should ultimately serve to support rather than replace human creativity.

9. ACKNOWLEDGEMENTS

This study would not have been possible without the crew and the cast that helped bring "The Egg" to life, Henrik Schønau Fog, Mathias Ramsø Thomsen, Aishah Hussain and the rest of SMILE Lab at AAU Copenhagen, as well as everyone else who in any way participated, tested, watched and contributed to this project.

10. REFERENCES

References

- [1] S. Zwerman and J. A. Okun, *The Ves Handbook of Virtual production*. Routledge, 2024.
- [2] M. Curtin and J. Vanderhoef, "A vanishing piece of the pi: The globalization of visual effects labor," *Television new media*, vol. 16, no. 3, p. 219–239, 2015. [Online]. Available: <http://dx.doi.org/10.1177/1527476414524285>
- [3] Priadko and M. Sirenko, "Virtual production: a new approach to filmmaking," *Bulletin of Kyiv National University of Culture and Arts. Series in Audiovisual Art and Production*, vol. 4, pp. 52–58, 06 2021.
- [4] G. W. Perkins and S. Echeverry, "Virtual production in action: A creative implementation of expanded cinematography and narratives," in *ACM SIGGRAPH 2022 Posters*, ser. SIGGRAPH '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: <https://doi.org/10.1145/3532719.3543231>
- [5] M. El-Nasr and I. Horswill, "Intelligent lighting for game environments," *Journal of Game Development*, vol. 1, pp. 17–50, 01 2005.
- [6] D. Ratajczyk, "Uncanny valley in video games: An overview," *Homo Ludens*, pp. 135–148, 12 2019.

- [7] R. McDonnell and M. Breidt, "Face reality: investigating the uncanny valley for virtual faces," in *ACM SIGGRAPH ASIA 2010 Sketches*, ser. SA '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: <https://doi.org/10.1145/1899950.1899991>
- [8] A. E. O. Knowlton, E. O. Knowlton, F. M. Scalzo, and J. C. Hulbert, "Something is lurking: The impact of liminality on the emotional something is lurking: The impact of liminality on the emotional valence of buildings valence of buildings," 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:260802398>
- [9] A. Sonoda and K. Ueki, "Discomfort analysis of liminal space images," in *Other Conferences*, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:269549576>
- [10] A. MacDonald, "The liminal space," 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:151895980>
- [11] C. Schnugg, *Spaces In-Between: Liminality*, 03 2019, pp. 55–72.
- [12] O. James, R. Achard, J. Bird, and S. Cooper, "Colour-managed led walls for virtual production," in *ACM SIGGRAPH 2021 Talks*, ser. SIGGRAPH '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3450623.3464682>
- [13] N. Kadner. (2024) Led wall evokes early-cinema effects for poor things. [Online]. Available: <https://theasc.com/articles/virtual-world-virtual-effects-poor-things>
- [14] E. Hasche and R. Creutzburg, "Aces color workflow in unreal engine 5," *Electronic Imaging*, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:270294429>
- [15] D. Walker, C. Payne, P. Hodoul, and M. Dolan, "Color management with opencolorio v2," *ACM SIGGRAPH 2021 Courses*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:236162441>
- [16] A. Maltz, "The academy color encoding system: Standards for digital image interchange, color management and long-term archiving," *Smpte Motion Imaging Journal*, vol. 125, pp. 34–39, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:227010364>
- [17] H. Sial, R. Baldrich, M. Vanrell, and D. Samaras, "Light direction and color estimation from single image with deep regression," *London Imaging Meeting*, vol. 2020, pp. 139–143, 09 2020.
- [18] A. Meka, C. Häne, R. Pandey, M. Zollhöfer, S. Fanello, G. Fyffe, A. Kowdle, X. Yu, J. Busch, J. Dourgarian, P. Denny, S. Bouaziz, P. Lincoln, M. Whalen, G. Harvey, J. Taylor, S. Izadi, A. Tagliasacchi, P. Debevec, C. Theobalt, J. Valentin, and C. Rhemann, "Deep reflectance fields: high-quality facial reflectance field inference from color gradient illumination," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: <https://doi.org/10.1145/3306346.3323027>
- [19] D. Korolov, D. Roble, J.-C. Bazin, R. Zioma, R. Pieke, J. Kember, and D. Luebke, "Future of artificial intelligence and deep learning tools for vfx," in *ACM SIGGRAPH 2018 Panels*, ser. SIGGRAPH '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3209621.3219782>
- [20] N. Anantrasirichai and D. Bull, "Artificial intelligence in the creative industries: A review," 07 2020.
- [21] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu, "Real-time neural style transfer for videos," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7044–7052.
- [22] C. Iwuanyanwu. (2019) Real-time style transfer. [Online]. Available: <https://medium.com/@chimezie.iwuanyanwu/real-time-style-transfer-caffa3393833>
- [23] W. Frames. (2022) Bringing deep learning to ue5 — pt. 2. [Online]. Available: <https://medium.com/@weirdframes/bringing-deep-learning-to-unreal-engine-5-pt-2-51c1a2a2c3>
- [24] J. Preece, Y. Rogers, and H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*, 4th ed. Hoboken, NJ: Wiley, 2015.
- [25] P. Mayring, "Qualitative content analysis," *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research [On-line Journal]*, <http://qualitative-research.net/fqs/fqs-e/2-00inhalt-e.htm>, vol. 1, 06 2000.
- [26] J. Selan, "Using lookup tables to accelerate color transformations," 2004. [Online]. Available: <https://developer.nvidia.com/gpugems/gpugems2/part-iii-high-quality-rendering/chapter-24-using-lookup-tables-accelerate-color>
- [27] [Online]. Available: <https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/UsingLUTs/>
- [28] E. Georgiadou, "Komodies stin periodo tis chountas: I periptosi tis seiras 'ekeinos kai ekeinos'», 02 2020.

11. APPENDIX

11.1 Appendix 1: Color Coded Interviews

Full color coded data from Focus Group interviews:

Green: pros and opportunities
Dark blue circle: general VP pros + similarities with "normal" film making
Light blue circle: technical opportunities and ideas
Orange: adjustments and learning
Red: cons, limitations and requirements
Circle: communication

thesis test participants

QUOTES	Lighting in VP	Greenscreen / LED vs Rear Projection	Communication with the VAD
Me	How did you find it specifically from the DP point of view and the lighting point of view? What were the main light constraints or difficulties that you wouldn't find on a traditional set?	What about the difference between shooting in the studio with a green screen and shooting in the studio with a virtual production background? Because on the green screen you just care about the actors being evenly lit and everything.	What about the communication with the visual production crew? Like, was it limiting for you that you had to go to somewhere, to someone like Matthias and ask him to change things? Were there any communication issues?
G1P1	<ul style="list-style-type: none">I really enjoy the thing that you can change the light in the scene and then change the light in the physical world as well. But I think we still need to learn in which order to do it.<ul style="list-style-type: none">If you should change the physical light first or you should change the light in the digital.And lighting scenes in physical world is quite easy and think it's just the gesture they question which is more.I think what we think of is always like where does the light come from? [...]<ul style="list-style-type: none">It's like the reference of how we light everything. And then it's just the matter of like controlling the light.And I think so it was, it would depend on the scene. The digital scene, always on how you wanted to light the thing.	<ul style="list-style-type: none">you can see the background and you can be more creative in the moment as as a photographer where if you shoot a green screen you're just handing it over to somebody else to hope for the best.	<ul style="list-style-type: none">I didn't find it limiting at all. And I think you just need to do it a couple of times with the person and learn how they interpret. Like, should I just say numbers or should I just say I want something warmer? Or should I say I want A2 3200 Kelvin? Or is it the way you need to learn to communicate with the crew because it's the same things. It's like more intense? Or is it the temperature? Or is it like the soft the same terms we use. It's just a new way of communication?
G1P2	<ul style="list-style-type: none">And I think like when do how do we shoot that and where should the light be now instead of when you're on set or you're in real location, you just it just comes very natural because you can see everything at once. At every angle at once. But here you have to change it around.Were just starting out just like trial and error of it. But yeah, with this there's I think it's also like trying to wrap your head around it because there's so many possibilities that it's it can be kind of like overwhelming where in the real world it's just kind of you have to work with what you got.	<ul style="list-style-type: none">you always have to remember what the image is on aAnd I think that's really limiting especially if it's not learning that the actor actually has to interact with the environment.	<ul style="list-style-type: none">It's kind of like having a get down get up and get a camera.communication is what's the most new thing about and also just the language communicate with the person about this.You have to think about what angles you want.

	<ul style="list-style-type: none">So it was really interesting to see how to match, how we can get the lighting to match out here and but also just to remove yourself of it because we know it's not real. So how do you can you even convince yourself that it's real?<ul style="list-style-type: none">So it kind of runs the magic in doing yourself.		
G1P3	--	--	<ul style="list-style-type: none">[Also I found confusing the steps. First you need to find OK. This is going to be the background you think, but you need to like turn the things and they will find the background after?
G2P1	<ul style="list-style-type: none">Real life, like the highlights. We talked a lot about the highlights.And we suggested a little bit with the with the projector because it couldn't quite reach the white point. Which meant that it kind of looked slightly off because we were making it look like it was overexposed, but it wasn't quite white, so it didn't feel overexposed.<ul style="list-style-type: none">So we we played around with that and we used some filters in front of the lenses, some prismes and and Climmer glass to kind of give it a sort of texture of a softness and a bit of glow around the edges of the windows.How do you match the projector's contrast with the contrast of the lighting that you have made on set that was really that was good for.So I think we just decided to push it more kind of like the like way (laughter) and going in the contrast.<ul style="list-style-type: none">It would have been much more of a challenge if we tried to light it to be almost like this sort of soft daylight.I don't know if the where where it is right now, I don't think I'd want to use the technology to do that. I think I'd much prefer to be on set in a location and work with that.	<ul style="list-style-type: none">you can see the background and you can be more creative in the moment as as a photographer where if you shoot a green screen you're just handing it over to somebody else to hope for the best.	<ul style="list-style-type: none">It's kind of like having a get down get up and get a camera.communication is what's the most new thing about and also just the language communicate with the person about this.You have to think about what angles you want.

	<ul style="list-style-type: none">The contrast ratio was really hard to get. You had to kind of take in the contrast a lot more than you would when you'd be shooting normally.I think like the big thing for me was like, normally we had a contrast ratio on the screen of three stops, three to four stops. So like the blacks that look black weren't actually black. They were like just four stops under [...]<ul style="list-style-type: none">I found really hard matching was that you didn't have the same kind of roll off in your exposure on your camera.You have to keep your your exposure quite low to be able to expose for the screen. I find it quite limiting that you had to expose for the screen also because it didn't give off light.There's one of the big problems is that if you get close to the screen, you have the screen that has like a it doesn't have focus roll off, it's just like everything is in focus right now [...]<ul style="list-style-type: none">But there's lenses that have the focus data output. If you had that data you could simulate the roll off through the back of the camera. Something that that would make it more like a focus and then you could basically choose like up to like even right up to the screen because you had a realistic roll off.I think like if you had the roll off, that would you could you'd be much more flexible in your selection of focal ranges and you can do bigger shots and you could it would help a lot.	<ul style="list-style-type: none">Like screen would be better because they give the lighting of the environment like for our suit reflection stuff like that.<ul style="list-style-type: none">It requires like creating an environment that follows around it needs to be super flexible to work because there's also a lot of creative decisions in it.	<ul style="list-style-type: none">background? Because it's not quite out of focus.<ul style="list-style-type: none">the closer you are to the and the more out of focus the background is, the more believable it is at the moment.Yeah, like having having overgrading or having someone working live on one feed so you don't just have an output and a camera reacting to it capturing, but also have a OAT (?) setup so someone who is able to you know work with that signal afterwards.because then you can talk about color correcting, then you can have the working on the look of the final picture while working on the look of the background and kind of complementing each other and knowing that if you're going to put green in the background, then you have to keep in the shadows, then you have to put keep the shadows clean or what's the Max my night vision where does that fall on the curve that we want for this look and stuff like that might make a bit of sense.
G2P2	<ul style="list-style-type: none">It wouldn't contain much as much light as there was in the the film like in the real life.		
G2P3			
G3P1			
G3P2			
G3P3			
G3P4			

11.2 Appendix 2: LUT Generator

Entire Python Scripts from Implementation 1 of custom LUT-Generator:

```
import numpy as np
import cv2
def color_transfer(source, target):
    # converts the images from the RGB to L*ab* color space.
    # (note: OpenCV expects floats to be 32-bit, so use that instead of 64-bit)
    source = cv2.cvtColor(source, cv2.COLOR_BGR2LAB).astype("float32")
    target = cv2.cvtColor(target, cv2.COLOR_BGR2LAB).astype("float32")

    # computes color statistics for the source and target images
    (lMeanSrc, lStdSrc, aMeanSrc, aStdSrc, bMeanSrc, bStdSrc) = image_stats(source)
    (lMeanTar, lStdTar, aMeanTar, aStdTar, bMeanTar, bStdTar) = image_stats(target)
    # subtracts the means from the target image
    (l, a, b) = cv2.split(target)
    l -= lMeanTar
    a -= aMeanTar
    b -= bMeanTar
    # scales by the standard deviations
    l = (lStdTar / lStdSrc) * l
    a = (aStdTar / aStdSrc) * a
    b = (bStdTar / bStdSrc) * b
    # adds in the source mean
    l += lMeanSrc
    a += aMeanSrc
    b += bMeanSrc
    # clips the pixel intensities to [0, 255] if they fall outside this range
    l = np.clip(l, 0, 255)
    a = np.clip(a, 0, 255)
    b = np.clip(b, 0, 255)
    # merges the channels together and convert back to the RGB color
    # space, being sure to utilize the 8-bit unsigned integer data
    # type
    transfer = cv2.merge([l, a, b])
    transfer = cv2.cvtColor(transfer.astype("uint8"), cv2.COLOR_LAB2BGR)

    # returns the color transferred image
    return transfer

def image_stats(image):
    # computes the mean and standard deviation of each channel
    (l, a, b) = cv2.split(image)
    (lMean, lStd) = (l.mean(), l.std())
    (aMean, aStd) = (a.mean(), a.std())
    (bMean, bStd) = (b.mean(), b.std())
    # returns the color statistics
    return (lMean, lStd, aMean, aStd, bMean, bStd)
```

11.3 Appendix 3: Training DL Models

Full TrainPosition Jupyter Notebook

```
In [1]: #Imports
import tensorflow as tf
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:2876: The name tf.nn.sparse_softmax_cross_entropy is deprecated. Please use tf.nn.sparse_softmax_cross_entropy instead.

In [11]: class ImageDataset:
def __init__(self, image_folder, labels_file):
    self.image_folder = image_folder
    self.labels_file = labels_file

    # Load the labels from the text file
    with open(labels_file, 'r') as file:
        self.labels = [self.parse_label(line.strip()) for line in file]

    # Get the list of image filenames, assuming they are in order
    self.image_filenames = ['img%d%screenshots' % (i, self.labels[i].img) for i in range(len(self.labels))]

    def parse_label(self, label_str):
        # Parse a label string like '9x92.233 9x96.128 2x18.996' into a list of floats.
        values = label_str.split()
        x = float(values[0].split('-')[1])
        y = float(values[1].split('-')[1])
        z = float(values[2].split('-')[1])
        return np.array([x, y, z])

    def load_image(self, filepath):
        # Load the image, resize to 256x256, and normalize to [0, 1].
        image = Image.open(filepath).convert('RGB')
        image = image.resize((256, 256)) # Resize to 256x256 for uniformity
        image = np.array(image) / 255.0 # Normalize pixel values to [0, 1]
        return image

    def __len__(self):
        # Return the number of images in the dataset.
        return len(self.image_filenames)

    def __getitem__(self, idx):
        # Get the image and corresponding label at the given index.
        # Get the image path and load the image
        img_path = os.path.join(self.image_folder, self.image_filenames[idx])
        image = self.load_image(img_path)

        # Get the corresponding label (x, y, z values)
        label = self.labels[idx]

        return image, label

# Initialize the dataset
image_folder = r'C:\Users\jagost\Documents\Media\log\HLS Thesis\TrainData\LightCoordinates\light_position.txt'
labels_file = r'C:\Users\jagost\Documents\Media\log\HLS Thesis\TrainData\LightCoordinates\light_position.txt'
dataset = ImageDataset(image_folder, labels_file)

In [11]: dataset_length = len(dataset)
print("length of the dataset:", dataset_length)


length of the dataset: 385

In [11]: #To use if the dataset is initialized properly, I will try to access the 385th element (index 385) and print the image and its corresponding x,y,z values
image, label = dataset[385]

# Print the corresponding x, y, z values
print("x={label[0]}, y={label[1]}, z={label[2]}")

# Visualize the image
plt.imshow(image) # Show the image
plt.axis('off') # Hide axes for clarity
plt.show()

x=177.226, y=177.342, z=462.673



In [ ]: #Now lets convert the dataset to tensorflow:

In [11]: def generator():
for i in range(len(dataset)):
    image, label = dataset[i]
    yield image, label

# Convert to tf.data.Dataset
tf_dataset = tf.data.Dataset.from_generator(generator,
output_types=(tf.float32, tf.float32),
output_shapes=((256, 256, 3), (3,)))

# Shuffle and batch the data
batch_size = 32
tf_dataset = tf_dataset.shuffle(buffer_size=len(dataset)).batch(batch_size)

WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:2720: calling DatasetV2.from_generator (from tensorflow.python.data.ops.dataset_ops) with output_types is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead
WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:2720: calling DatasetV2.from_generator (from tensorflow.python.data.ops.dataset_ops) with output_shapes is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead

In [ ]: #And now I'll try to define the light direction guesser model.

In [6]: from tensorflow.keras import layers, models

def create_model():
    model = models.Sequential([
        layers.Conv2D(16, (3, 3), activation='relu', input_shape=(256, 256, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(32, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(3) # Output 3 values for x, y, z
    ])
    return model

# Instantiate the model
model = create_model()

# Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mse'])

WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:161: The name tf.nn.softmax is deprecated. Please use tf.nn.softmax instead.

WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:189: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

In [7]: # Train the model, 20 epochs used to avoid overfitting
epochs = 20
History = model.fit(tf_dataset, epochs=epochs)

Epoch 2/20
WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:402: The name tf.nn.softmax is deprecated. Please use tf.nn.softmax instead.
WARNING:tensorflow:From c:\Users\jagost\AppData\Local\Programs\Python\Python311\lib\site-packages\tensorflow\tensorflow.py:184: The name tf.nn.softmax is deprecated. Please use tf.nn.softmax instead.

32/32 [=====] - 51s 42ms/step - loss: 1594.7346 - mse: 52.4844
Epoch 2/20
32/32 [=====] - 51s 42ms/step - loss: 6873.8818 - mse: 78.4688
Epoch 3/20
32/32 [=====] - 54s 42ms/step - loss: 5172.8504 - mse: 59.8256
Epoch 4/20
32/32 [=====] - 62s 43ms/step - loss: 3671.2144 - mse: 47.8728
Epoch 5/20
32/32 [=====] - 62s 40ms/step - loss: 2348.5081 - mse: 36.9512
Epoch 6/20
32/32 [=====] - 62s 40ms/step - loss: 1893.9462 - mse: 32.4552
Epoch 7/20
32/32 [=====] - 62s 40ms/step - loss: 976.6188 - mse: 22.8998
Epoch 8/20
32/32 [=====] - 62s 41ms/step - loss: 689.8735 - mse: 18.4587
Epoch 9/20
32/32 [=====] - 61s 412ms/step - loss: 682.7635 - mse: 18.1466
Epoch 10/20
32/32 [=====] - 61s 407ms/step - loss: 387.2935 - mse: 14.1768
Epoch 11/20
32/32 [=====] - 61s 412ms/step - loss: 263.8909 - mse: 11.5955
Epoch 12/20
32/32 [=====] - 61s 412ms/step - loss: 255.8945 - mse: 11.7284
Epoch 13/20
32/32 [=====] - 64s 412ms/step - loss: 193.9849 - mse: 9.8998
Epoch 14/20
32/32 [=====] - 61s 405ms/step - loss: 182.4133 - mse: 9.7388
Epoch 15/20
32/32 [=====] - 55s 405ms/step - loss: 151.7756 - mse: 8.6446
Epoch 16/20
32/32 [=====] - 54s 418ms/step - loss: 137.7937 - mse: 9.4056
Epoch 17/20
32/32 [=====] - 54s 407ms/step - loss: 131.2584 - mse: 8.4088
Epoch 18/20
32/32 [=====] - 54s 412ms/step - loss: 129.1344 - mse: 8.2873
Epoch 19/20
32/32 [=====] - 54s 409ms/step - loss: 186.6531 - mse: 7.2637
Epoch 20/20
32/32 [=====] - 54s 409ms/step - loss: 182.5466 - mse: 7.1299

In [11]: # Load the test image (for example)
new_image_path = r'C:\Users\jagost\Documents\Media\log\HLS Thesis\TrainData\Train Data\NewImage_for_Prediction.png'
new_image = dataset.load_image(new_image_path)

# Add batch dimension (since model expects batches)
new_image = np.expand_dims(new_image, axis=0)

# Predict the x, y, z values
predicted_values = model.predict(new_image)
print("Predicted x, y, z values: (predicted values)")

2/1 [=====] - 0s 24ms/step
Predicted x, y, z values: [[98.108475 164.185735 46.889080]]

In [12]: # Evaluate the model
loss, mse = model.evaluate(tf_dataset)
print("Mean Absolute Error: (mse)")

32/32 [=====] - 41s 926ms/step - loss: 96.1878 - mse: 7.1383
Mean Absolute Error: 7.138288917541384

In [ ]: #The m.s.e of ~7m is acceptable, as the expected difference in lighting between two positions 7m apart, will not be visible in different objects.
```

Full TrainColor Jupyter Notebook

```
In [2]: #Imports
import tensorflow as tf
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:2076: The name tf.nn.softmax_cross_entropy is deprecated. Please use tf.nn.softmax_cross_entropy_with_logits instead.

In [3]: class ImageDataset:
def __init__(self, image_folder, labels_file):
    """
    Initialize the dataset with an image folder and labels file.
    """
    self.image_folder = image_folder

    # Load the labels from the text file
    with open(labels_file, 'r') as file:
        self.labels = [self.parse_label(line.strip()) for line in file]

    # Get the list of image filenames, ensuring they are in order
    self.image_filenames = [f'{i}{str(i).zfill(5)}.jpg' for i in range(len(self.labels))]

def parse_label(self, label_str):
    """
    Parse a label string like '7x02.231 7x36.128 2x18.990' into a list of floats.
    """
    values = label_str.split()
    x = float(values[0].split('.')[1])
    y = float(values[1].split('.')[1])
    z = float(values[2].split('.')[1])
    return np.array([x, y, z])

def load_image(self, filepath):
    """
    Load the image, resize to 128x128, and normalize to [0, 1].
    """
    image = Image.open(filepath).convert('RGB')
    image = image.resize((256, 256)) # Resize to 256x256 for uniformity
    image = np.array(image) / 255.0 # Normalize pixel values to [0, 1]
    return image

def len_(self):
    """
    Return the number of images in the dataset.
    """
    return len(self.image_filenames)

def __getitem__(self, idx):
    """
    Get the image and corresponding label at the given index.
    """
    # Get the image path and load the image
    img_path = os.path.join(self.image_folder, self.image_filenames[idx])
    image = self.load_image(img_path)

    # Get the corresponding label (x, y, z values)
    label = self.labels[idx]

    return image, label

# Initialize the dataset
image_folder = r'C:\Users\apost\Documents\Unreal Projects\LightAutomation\Scenes\Screenshots\WindowEditor'
labels_file = r'C:\Users\apost\Documents\Unreal\Projects\LightAutomation\Scenes\Screenshots\WindowEditor\train_data\light_coordinates\light_color.txt'
dataset = ImageDataset(image_folder, labels_file)

In [4]: dataset_length = len(dataset)
print('Length of the dataset:', dataset_length)


Length of the dataset: 188

In [5]: #Now use if the dataset is initialized properly, I will try to access the 188th element (index 188) and print the image and its corresponding x,y,z values
image, label = dataset[188]

# Print the corresponding x, y, z values
print(f"x={label[0]}, y={label[1]}, z={label[2]}")

# Visualize the image
plt.imshow(image) # Show the image
plt.axis('off') # Hide axes for clarity
plt.show()

x=0.165, y=0.362, z=0.127


```

```
In [ ]: #Now lets convert the dataset to tensorflow

In [6]: def generator():
for i in range(len(dataset)):
    image, label = dataset[i]
    yield image, label

# Convert to tf data.Dataset
tf_dataset = tf.data.Dataset.from_generator(generator,
                                            output_types=(tf.float32, tf.float32),
                                            output_shapes=(256, 256, 3), (3,)))

# Shuffle and batch the data
batch_size = 32
tf_dataset = tf_dataset.shuffle(buffer_size=len(dataset)).batch(batch_size)

WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:873: calling DatasetV2.from_generator (from tensorflow.python.data.ops.dataset_ops) with output_types is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead.
WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:873: calling DatasetV2.from_generator (from tensorflow.python.data.ops.dataset_ops) with output_shapes is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead.
WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:873: calling DatasetV2.from_generator (from tensorflow.python.data.ops.dataset_ops) with output_shapes is deprecated and will be removed in a future version.
Instructions for updating:
Use output_signature instead.

In [ ]: #And now I'll try to define the light direction guesser model.

In [7]: from tensorflow.keras import layers, models

def create_model():
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)),
        layers.MaxPooling2D(2, 2),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D(2, 2),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D(2, 2),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(3) # Output 3 values for x, y, z
    ])
    return model

# Instantiate the model
model = create_model()

# Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mse'])

WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:341: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:380: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

In [8]: # Train the model, 20 epochs used to avoid overfitting
epochs = 20
history = model.fit(tf_dataset, epochs=epochs)

Epoch 1/20
WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:402: The name tf.nn.softmax_cross_entropy_with_logits is deprecated. Please use tf.nn.softmax_cross_entropy_with_logits_v2 instead.
WARNING:tensorflow:From c:\Users\apost\AppData\Local\Programs\Python\Python311\site-packages\tensorflow\tensorflow.py:384: The name tf.nn.softmax_cross_entropy_with_logits is deprecated. Please use tf.nn.softmax_cross_entropy_with_logits_v2 instead.
1/2 [=====] - loss: 0.1074 - mse: 0.4383
Epoch 2/20
1/2 [=====] - loss: 0.0708 - mse: 0.2425
Epoch 3/20
1/2 [=====] - loss: 0.0707 - mse: 0.2289
Epoch 4/20
1/2 [=====] - loss: 0.0625 - mse: 0.2128
Epoch 5/20
1/2 [=====] - loss: 0.0554 - mse: 0.1988
Epoch 6/20
1/2 [=====] - loss: 0.0504 - mse: 0.1834
Epoch 7/20
1/2 [=====] - loss: 0.0447 - mse: 0.1780
Epoch 8/20
1/2 [=====] - loss: 0.0424 - mse: 0.1633
Epoch 9/20
1/2 [=====] - loss: 0.0404 - mse: 0.1595
Epoch 10/20
1/2 [=====] - loss: 0.0382 - mse: 0.1539
Epoch 11/20
1/2 [=====] - loss: 0.0378 - mse: 0.1534
Epoch 12/20
1/2 [=====] - loss: 0.0367 - mse: 0.1580
Epoch 13/20
1/2 [=====] - loss: 0.0354 - mse: 0.1464
Epoch 14/20
1/2 [=====] - loss: 0.0328 - mse: 0.1405
Epoch 15/20
1/2 [=====] - loss: 0.0325 - mse: 0.1398
Epoch 16/20
1/2 [=====] - loss: 0.0354 - mse: 0.1468
Epoch 17/20
1/2 [=====] - loss: 0.0305 - mse: 0.1342
Epoch 18/20
1/2 [=====] - loss: 0.0298 - mse: 0.1318
Epoch 19/20
1/2 [=====] - loss: 0.0291 - mse: 0.1306
Epoch 20/20
1/2 [=====] - loss: 0.0286 - mse: 0.1271

In [9]: # Load the test image (for example)
new_image_path = r'C:\Users\apost\Documents\Unreal\Projects\LightAutomation\Scenes\Screenshots\WindowEditor\train_data\new_image_for_prediction2.jpg'
new_image = Image.open(new_image_path)

# Add batch dimension (since model expects batches)
new_image = np.expand_dims(new_image, axis=0)

# Predict the x, y, z values
predicted_values = model.predict(new_image)
print(f'Predicted x, y, z values: {predicted_values}')

1/1 [=====] - 8s 116ms/step
Predicted x, y, z values: [[0.185069 0.139669 0.1266424]]
1/1 [=====] - 8s 116ms/step
Predicted x, y, z values: [[0.185069 0.139669 0.1266424]]

In [10]: # Evaluate the model
loss, mse = model.evaluate(tf_dataset)
print(f'When Absolute Error: {mse}')

1/1 [=====] - 26s 90ms/step - loss: 0.8201 - mse: 0.1341
Mean Absolute Error: 0.1146793526889747

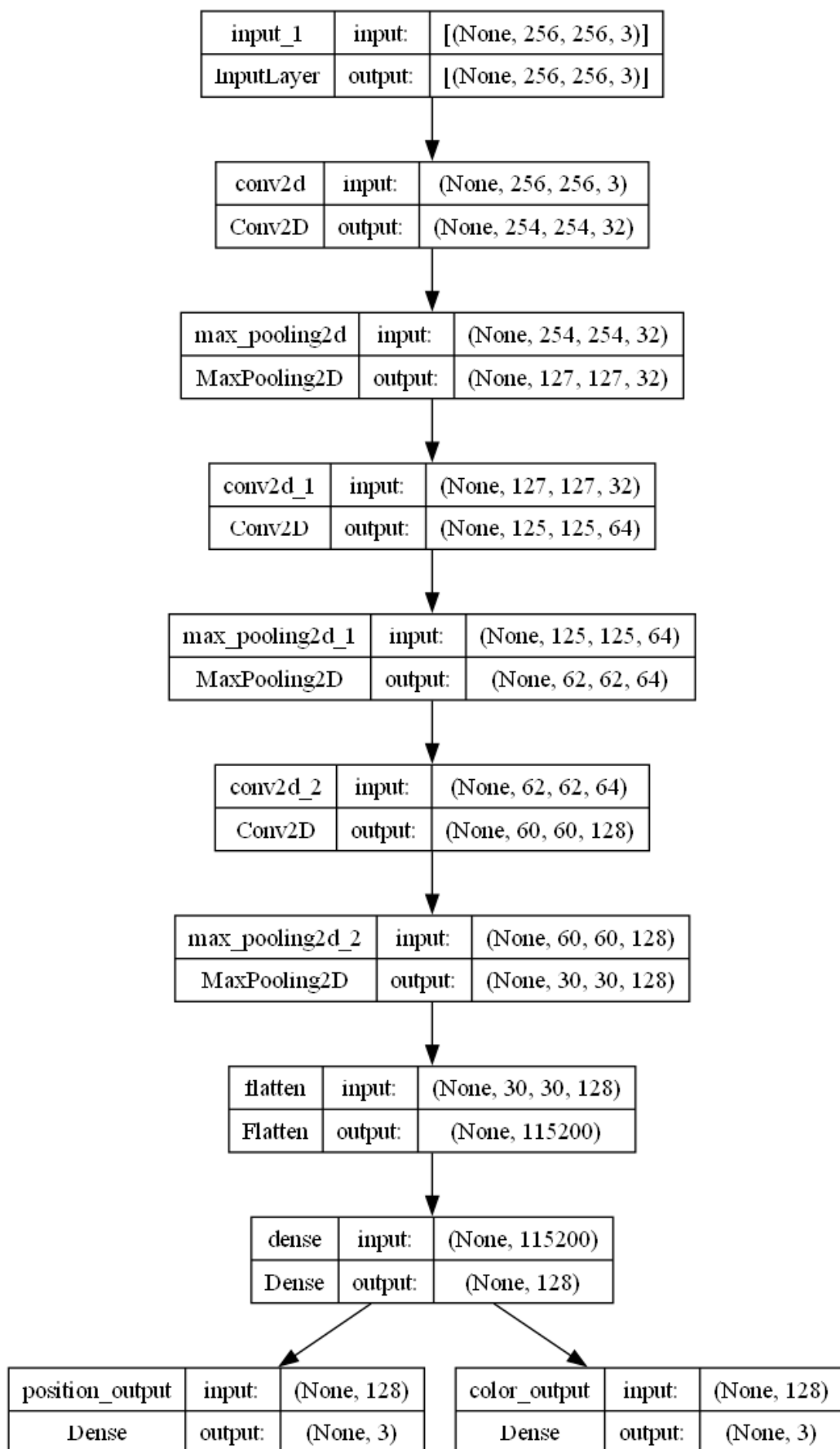
In [ ]:

In [1]: #Exporting model as an .onnx file:

In [ ]: import tf2onnx
import onnx

In [ ]:
```


11.4 Appendix 4: Model Architecture



11.5 Appendix 5: "The Egg" short Film

11.5.0.1 Watch the short film

The VP short film can be watched in the following un-listed link on YouTube.com:

- <https://www.youtube.com/watch?v=AnOrfVIRIC8>

Note: At the time of writing this paper, this cut is the final draft for the MSc Thesis submission. For any later releases, please contact the author.

11.5.0.2 Additional Behind the Scenes



Figure 18. BTS1



Figure 19. BTS2

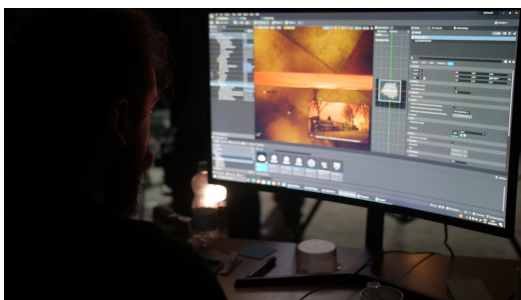


Figure 20. BTS3



Figure 21. BTS4



Figure 22. BTS5



Figure 23. BTS6