Evaluating Film Projects Considering Feasibility for Virtual Production – An LLM-based Assistant at Pre-Production

Can Bilgin*†, Julian Geheeb*, Dr. Jonathan Partecke†, Lena Fischer†, Prof. Dr. Georg Groh*
* Technische Universität München
Munich, Germany
e-mail: julian.geheeb@tum.de
† CreatiF Center
Hochschule für Fernsehen und Film München
Munich, Germany
e-mail: {c.bilgin | j.partecke | l.fischer}@hff-muc.de

***Abstract*—Virtual production workflows require the identification and preparation of digital assets based on screenplay content, a process that is typically performed manually and can be labor-intensive. This research investigates whether Large Language Models (LLMs) can assist in automating the extraction and structuring of production-relevant asset information from screenplays. To address this problem, a software application was implemented that analyzes film script, generates structured asset requirements, and matches them with available assets within an Unreal Engine environment. The proposed workflow was evaluated using multiple LLMs and assessed with quantitative metrics including precision, recall, and execution consistency. The experimental results show that high precision in screenplay analysis and asset extraction can be achieved and that structured output generation is feasible for capable models. However, the reliability of the pipeline varies significantly across different LLMs. The findings indicate that LLM-based screenplay analysis can support asset preparation in virtual production workflows, although practical deployment remains strongly dependent on model capability.**

Keywords-virtual production; generative AI; LLM; comparison of different LLMs; automated screenplay analysis; game engine.

I. Introduction

With recent advancements in game engines and real-time rendering techniques, film shooting environments can increasingly be created virtually using game engine technologies. Virtual production encompasses a variety of approaches; however, across these variations, the environment is typically generated within a game engine or a comparable real-time system. The actual filming process may then be conducted using either a virtual camera or a physical camera that is synchronized with the virtual environment [1].

In the 1990s, Computer-Generated Imagery (CGI) was introduced into the film industry, marking a significant shift in visual effects production. The adoption of computer-based visual effects fundamentally transformed filmmaking workflows by enabling films to be processed and enhanced in digital formats. This advancement allowed filmmakers to create increasingly realistic and dynamic visual effects, thereby reducing dependency on the physical constraints of the real world. As a result, entirely new fantasy worlds and science fiction environments could be realized on screen [2] [3].

Finally, in the 2020s, a significant advancement was introduced to the film industry with the emergence of Light-Emitting Diode (LED) volumes for virtual production. In virtual production workflows, traditional green screens are replaced by large-scale LED walls on which digitally rendered environments are displayed in real time. These environments are generated using a game engine, while filming is conducted with conventional physical cameras that are synchronized with the virtual scene [1].

As the camera position or orientation changes, the content displayed on the LED wall is dynamically updated to reflect the corresponding perspective, thereby maintaining spatial and visual consistency. This real-time synchronization enables an interactive filming process. Actors perform in front of the LED wall and are able to interact with the virtual environment directly. In contrast to green screen–based production, actors are no longer required to imagine the surrounding scene; instead, they can perceive the virtual environment in real time and adapt their performance accordingly [4] [5].

The majority of film projects that employ virtual production utilize Unreal Engine as the industry-standard game engine [6]. Unreal Engine provides built-in assets and supports the integration of additional assets through external asset stores. During this stage, the visual effects (VFX) artist must determine whether the assets already available within the Unreal Engine project are sufficient, whether suitable assets can be sourced from external repositories, or whether new assets must be created specifically for the project. Although this decision is inherently subjective and dependent on the expertise and creative judgment of the VFX artist, the availability of a supporting software tool could facilitate and inform this decision-making process.

For this reason, the objective of this research is to develop a LLM–based tool that supports VFX artists regarding asset selection for film projects. This paper investigates how LLMs can support the transformation of screenplay text into structured data usable in virtual production workflows. The central research question is whether LLM-based approaches can reliably extract relevant production information from screenplay text.

Section II reviews related research and existing tools and

compares them with the proposed solution. The system architecture is then briefly described alongside with the framework for the experiments in Section III. Section IV presents the comparative evaluation of different LLMs, followed by a discussion in Section V. Finally, Section VI concludes the paper and outlines directions for future work.

II. Related work

This section presents a comparative analysis of relevant tools identified during the course of this research. Accordingly, a comparison of these tools is provided in Table I. Furthermore, the basic functionalities of the tools are also given explicitly:

- **FilmUstage** [7]: A tool for analyzing screenplays and generating shooting schedules.
- **Das Element** [8]: A tool for managing asset libraries.
- **Kubrick** [9]: A tool designed to automate the entire film production workflow.
- **FilmAgent** [10]: Another tool aimed at automating the complete film production workflow similar to Kubrick [9].
- **ScreenWriter** [11]: A tool for analyzing and summarizing complete film projects.
- **Movie Script Evaluator (MSE)**: The software solution proposed in this research to automatize a pipeline for specifically virtual production.

The aforementioned tools are analyzed across several dimensions, including whether they are developed specifically for virtual production (VP), whether they perform screenplay analysis using LLMs, whether they implement mechanisms within game engines or virtual environments for asset management, and whether they incorporate LLMs or other artificial intelligence techniques in their implementation.

Table I
Comparison of different tools.

Software Tool	Specific for VP	Screenplay Analysis	Asset Management	Usage of AI
FilmUstage	No	Yes	No	Yes
Das Element	Yes	No	Yes	Yes
Kubrick	Yes	No	No	Yes
FilmAgent	Yes	No	No	Yes
Screen Writer	No	Yes	No	Yes
Movie Script Evaluator (MSE)	Yes	Yes	Yes	Yes

When examining tools such as FilmUstage [7], Das Element [8], and ScreenWriter [11], no direct end-to-end workflow from screenplay analysis to virtual production can be observed. These tools typically focus on isolated aspects of the filmmaking process, such as screenplay analysis or asset and production management using LLMs and machine learning techniques, rather than providing an integrated pipeline.

In contrast to Kubrik [9] and FilmAgent [10], which target full automation of the film production pipeline, the proposed approach focuses on a specific task within virtual production, namely, screenplay analysis and the evaluation of relevant existing assets.

III. Methods

The proposed software solution is composed of multiple interacting components. The central component of the system is the Pix:e application [12]. Pix:e is an open-source project, which is developed at Technical University of Munich for improving workflows in game development. Pix:e functions as an orchestration layer that coordinates the various system actions and provides a graphical user interface. In addition, it records user-triggered actions and persists the associated data within its internal database.

On the right-hand side of the architecture diagram shown in Figure 1, the Custom Unreal Engine Plugin is depicted. This plugin was developed as part of the implementation phase of this work and enables the extraction and filtering of asset data from the Unreal Engine project. The plugin must be installed and configured by the user within the Unreal Engine environment. Configuration requires authentication credentials obtained from the Pix:e web application.

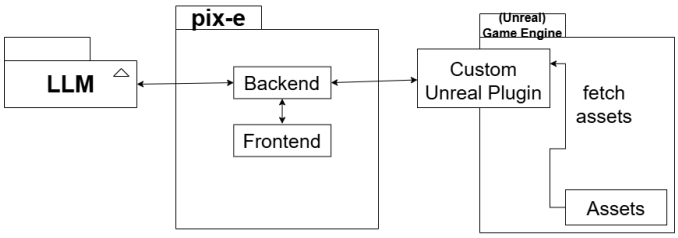

Figure 1. Software architecture schema.

Once the Custom Unreal Engine Plugin is properly configured, the user can initiate the transmission of asset data to Pix:e. On the Pix:e side, a dedicated Application Programming Interface (API) endpoint is responsible for handling incoming asset data. This endpoint receives the transmitted information and stores it in the Pix:e, which uses a Structured Query Language (SQL) based database, thereby making the assets available for subsequent processing within the system.

In addition, Pix:e is responsible for managing communication with LLMs in the background via RESTful APIs. To support this functionality, Pix:e implements a unified abstraction layer that enables interaction with different types of LLMs and model providers. This design allows the system to integrate both cloud-based models, such as Gemini [13] and ChatGPT [14], as well as locally hosted models accessed through Ollama [15].

One of the key considerations in this research is the selection of LLMs for evaluation. The investigation focuses on models that were among the most widely used as of January 2026. For locally deployable models, popularity data was obtained from the official Ollama platform [15], where models are ranked according to usage indicators such as the number of downloads and community adoption. For cloud-based models, offerings

Table II
MODELS USED FOR EVALUATION.

Model Name	Local/Cloud	Number of Parameters
GPT-4.1	Cloud	NA
Gemini 3 Flash	Cloud	NA
Llama3.1	Local	8b
DeepSeek-R1	Local	14b
Gemma3	Local	4b
Gemma3	Local	12b

from OpenAI [14] and Google [13] were considered. In Table II the selected models are given.

This study also evaluates a set of film screenplays. The selected screenplays, their corresponding film projects, and the rationale for their inclusion are listed below:

- **Tokyo At Night** [16]: The narrative is set in Tokyo, and the environments described in the screenplay would be challenging to recreate in Germany using conventional production methods. Furthermore, the screenplay was specifically developed with virtual production in mind.
- **September 5** [17]: The storyline is set in Munich, Germany, in the year 1972. Although this project was not produced using virtual production techniques [18], the depiction of historical locations and events suggests that such a project could potentially be realized using virtual production approaches.
- **The Seed of the Sacred Fig** [19]: The narrative takes place in Iran. The film was shot on location in Iran while facing significant political constraints [20]. Under these circumstances, the use of virtual production techniques could have provided an alternative means of production, allowing the film to be realized in a different location without encountering such restrictions.

Due to the length of the screenplays for September 5 and The Seed of the Sacred Fig, three scenes from each screenplay were selected and combined into a single plain text (TXT) file for analysis. Limiting the analysis to three scenes per screenplay reduced the complexity of result interpretation and ensured manageable experimental runtimes. In contrast, the complete screenplay of Tokyo At Night was provided to the system, as this project is comparatively shorter and consists of only four scenes.

Separate Unreal Engine projects were configured for each selected film project. Within these projects, a subset of the required assets was deliberately downloaded and integrated from the Fab store [21], while other necessary assets were intentionally omitted. This controlled setup enabled a systematic evaluation of the system's ability to identify missing assets. The analysis is therefore based on a pre-defined ground-truth classification of present and absent items.

Film screenplays follow a standardized and well-defined format that is commonly adopted across film projects [22]. In defining the ground-truth annotations for this research, both the conventional structure of film screenplays and established techniques used in the creation of shooting plans were taken into consideration. The resulting annotations serve as the reference ground truth for evaluating the accuracy and completeness of the LLM-based asset extraction.

IV. RESULTS

First, the screenplay analysis capabilities of different LLMs are compared. Next, the models are evaluated on their ability to identify missing elements by comparing screenplay information with existing assets in Unreal Engine. The pipeline is then executed multiple times for each LLM and film project to analyze variations across inference runs.

Extraction accuracy is evaluated with respect to different screenplays and LLMs. The results are presented and analyzed, and subsequently, precision, recall, and F1-score metrics are calculated for each LLM to provide a quantitative assessment of performance.

Considering the overall performance of the evaluated LLMs, as illustrated in Figure 2, GPT-4.1 achieved the highest detection performance across all screenplays. Figure 2 presents exclusively the number of correctly matched elements between the generated asset lists and the manually defined scene breakdown tables, thereby providing a direct comparison of detection accuracy among the models.

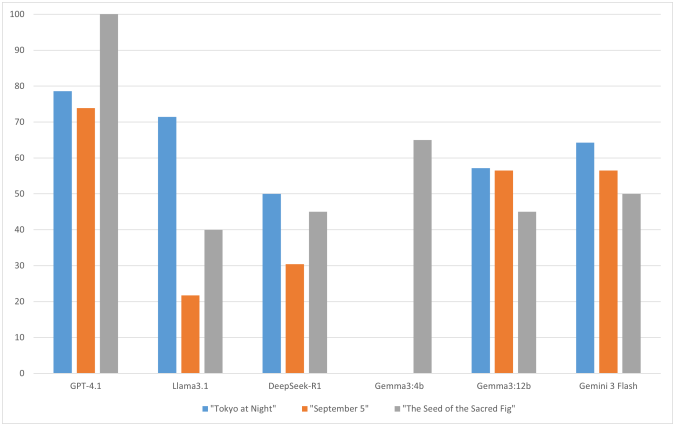

Figure 2. Percentage of successfully detected items per LLM.

In terms of classical information extraction metrics, detected items correspond to true positives, undetected items to false negatives, and hallucinated items to false positives. Following the evaluation, precision, recall, and F1-score metrics are calculated for each LLM, as presented in Table III. Since Gemma3:4B did not produce results for all screenplays (Figure 2), only successful runs were included in Table III.

The pipeline, implemented in the Movie Script Evaluator, first analyzes the screenplay and extracts a table of relevant assets. These extracted items are then compared with the assets available in the specified Unreal Engine project. Based on this comparison, the LLM indicates which screenplay elements can be realized with existing assets and labels each item as "FOUND" or "NOT FOUND". Based on the automatically generated tagging produced by the LLMs, the evaluated items were manually classified into four categories according to the ground-truth annotation: *Valid Found, Valid Not Found,*

Table III
CONFUSION MATRIX COMPONENTS (TP, FP, FN) AND DERIVED PRECISION, RECALL, AND F1-SCORE FOR EACH EVALUATED LLM REGARDING SCREENPLAY ANALYSIS.

	GPT-4.1	Llama3.1	DeepSeek-R1
True Positives	48	23	23
False Positives	0	0	0
False Negatives	9	34	34
Precision	1	1	1
Recall	0.84	0.4	0.4
F1-Score	0.91	0.58	0.58
	Gemma3 4B	Gemma3 12B	Gemini 3 Flash
True Positives	13	30	32
False Positives	1	0	1
False Negatives	7	27	25
Precision	0.93	1	0.97
Recall	0.65	0.53	0.56
F1-Score	0.76	0.69	0.71

Invalid Found, and *Invalid Not Found*. The definitions of these evaluation categories are as follows:

- **Valid Found**: The item is marked as "FOUND" by the LLM, and a corresponding asset exists within the Unreal Engine project.
- **Valid Not Found**: The item is marked as "NOT FOUND" by the LLM, and no corresponding asset exists within the Unreal Engine project.
- **Invalid Found**: The item is marked as "FOUND" by the LLM, although no corresponding asset exists within the Unreal Engine project.
- **Invalid Not Found**: The item is marked as "NOT FOUND" by the LLM, despite the existence of at least one suitable asset within the Unreal Engine project.

For the purpose of quantitative evaluation, these categories were mapped to standard classification metrics. Specifically, Valid Found corresponds to a *True Positive*, Invalid Found corresponds to a *False Positive*, and Invalid Not Found corresponds to a *False Negative*.

Based on the classification framework described above, precision, recall, and F1-score values were calculated for the models that generated valid outputs. The results of this evaluation are presented in Table IV.

Table IV
CONFUSION MATRIX COMPONENTS (TP, FP, FN) AND DERIVED PRECISION, RECALL, AND F1-SCORE FOR EACH EVALUATED LLM REGARDING MISSING ITEMS ANALYSIS.

	GPT-4.1	Gemma3 12B	Gemini 3 Flash
True Positives	28	12	11
False Positives	9	2	3
False Negatives	4	10	4
Precision	0.76	0.86	0.79
Recall	0.88	0.54	0.73
F1-Score	0.81	0.67	0.76

Notably, only three LLMs were able to produce valid outputs for the missing-item analysis: Gemma3:12B, GPT-4.1, and Gemini 3 Flash. According to the results presented in Table IV, Gemma3:12B achieved the highest precision but the lowest recall among the evaluated models, resulting in the lowest overall F1-score. In contrast, GPT-4.1 exhibited the lowest precision but the highest recall. Despite generating more false positives than the other models, its ability to correctly identify a larger proportion of relevant items led to the highest F1-score overall. Gemini 3 Flash demonstrated intermediate performance across all three metrics, indicating a more balanced but not leading performance compared to the other two models.

To evaluate the consistency of the LLMs, the entire pipeline was executed three times for each screenplay–LLM combination. 3 screenplays were evaluated with 6 LLMs yielding at 18 different configurations. Consistency was assessed using quantitative stability metrics based on the number of items generated at each execution. In Table V and Table VI, the arithmetic mean, median, and standard deviation values, calculated over the three execution attempts, are reported for each LLM–film project combination. This results in a total of 18 rows per table, providing a structured overview of the quantitative performance and consistency metrics across models and screenplays.

Table V
COMPARISON OF RELEVANT ITEMS GENERATION BASED ON LLM AND FILM SCREENPLAY.

LLM/Film Project	**Arithmetic mean ($\bar{x}$)**	**Median ($\tilde{x}$)**	**Standard Deviation (σ)**
GPT-4.1/Tokyo At Night	18.67	17	3.09
GPT-4.1/September 5	21.67	21	0.94
GPT-4.1/SSF	19.67	18	2.36
Gemini 3 Flash/Tokyo At Night	10.33	10	0.47
Gemini 3 Flash/September 5	13.33	13	0.47
Gemini 3 Flash/SSF	11.33	11	0.47
Llama 3.1/Tokyo At Night	10.67	9	3.09
Llama 3.1/September 5	3	4	2.16
Llama 3.1/SSF	5.67	8	4.03
DeepSeek-R1/Tokyo At Night	4.33	6	3.09
DeepSeek-R1/September 5	5.67	7	4.19
DeepSeek-R1/SSF	7.33	9	5.44
Gemma3:4B/Tokyo At Night	-	-	-
Gemma3:4B/September 5	-	-	-
Gemma3:4B/SSF	17	18	1.41
Gemma3:12B/Tokyo At Night	9.33	10	0.94
Gemma3:12B/September 5	12.33	12	1.25
Gemma3:12B/SSF	9.67	10	0.47

When examining the results, it can be observed that Gemini 3 Flash generates similar outputs at screenplay analysis with an average standard deviation of 0.47 across runs, making it the model with the lowest variability in this stage. In contrast, DeepSeek-R1 exhibits the highest average standard deviation, with a value of 4.24, indicating comparatively greater fluctuation in output quantities across repeated executions. Notably, the highest deviation within this category is observed for the combination of DeepSeek-R1 and September 5, where the variability between runs is most pronounced.

GPT-4.1 is the model that generated the highest number of items compared to the other evaluated LLMs. Across all executions and screenplays, GPT-4.1 produced an average of 20 items per screenplay analysis as can be seen at

Table VI
COMPARISON OF RECOMMENDATION LIST GENERATION BASED ON LLM AND UNREAL ENGINE PROJECT.

LLM/Film Project	Arithmetic mean ($\bar{x}$)	Median ($\tilde{x}$)	Standard Deviation (σ)
GPT-4.1/Tokyo At Night	381.33	382	12.26
GPT-4.1/September 5	53.33	53	12.66
GPT-4.1/SSF	33.33	33	2.05
Gemini 3 Flash/Tokyo At Night	-	-	-
Gemini 3 Flash/September 5	69.33	102	49.05
Gemini 3 Flash/SSF	32	43	22.99
Llama 3.1/Tokyo At Night	-	-	-
Llama 3.1/September 5	-	-	-
Llama 3.1/SSF	-	-	-
DeepSeek-R1/Tokyo At Night	-	-	-
DeepSeek-R1/September 5	-	-	-
DeepSeek-R1/SSF	-	-	-
Gemma3:4B/Tokyo At Night	-	-	-
Gemma3:4B/September 5	-	-	-
Gemma3:4B/SSF	-	-	-
Gemma3:12B/Tokyo At Night	292	316	47.24
Gemma3:12B/September 5	58	60	12.33
Gemma3:12B/SSF	24.33	26	2.36

Table V. Furthermore, in the recommendation generation stage, where the existing game engine assets are evaluated, GPT-4.1 generated an average of 156 recommended assets across all film projects and attempts, indicating a comparatively higher output volume relative to the other models, which can be calculated using the results at Table VI.

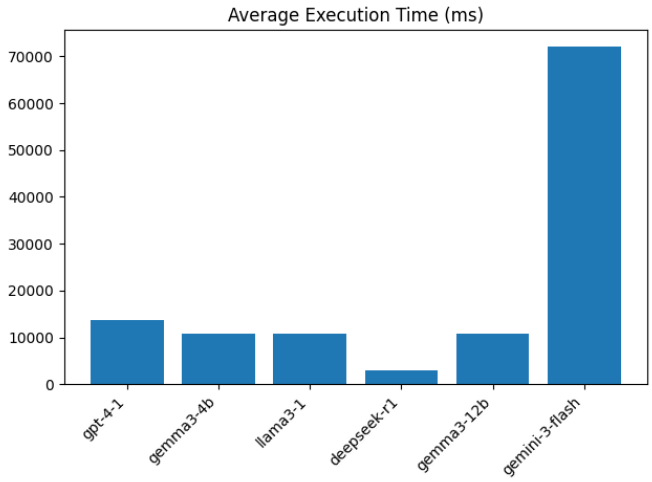

Figure 3. Average execution time per LLM in milliseconds.

Furthermore, the evaluated LLMs are analyzed with respect to additional metrics, including valid response generation rate and average execution time in milliseconds per model. According to Figure 3, Gemini 3 Flash exhibits an average execution time of 72 seconds per interaction. However, this value must be interpreted with caution. In order to comply with the rate limits imposed by the Gemini cloud service, an additional waiting mechanism was implemented within the system. The design of this throttling mechanism directly influenced the measured execution time, thereby inflating the observed average duration for this model.

The second-highest average execution time was observed for GPT-4.1, with approximately 13.7 seconds per interaction. Both versions of the Gemma 3 model (12 billion and 4 billion parameters) exhibited comparable performance, closely aligned with Llama 3.1 in terms of execution time. Specifically, Gemma3:12B achieved an average processing time of approximately 10.9 seconds, while Gemma3:4B averaged 10.8 seconds. Llama 3.1 demonstrated a similar average execution time of 10.7 seconds per interaction. With an average processing time of approximately 3 seconds per interaction, DeepSeek-R1 was the fastest model among those evaluated in the experimental testbeds.

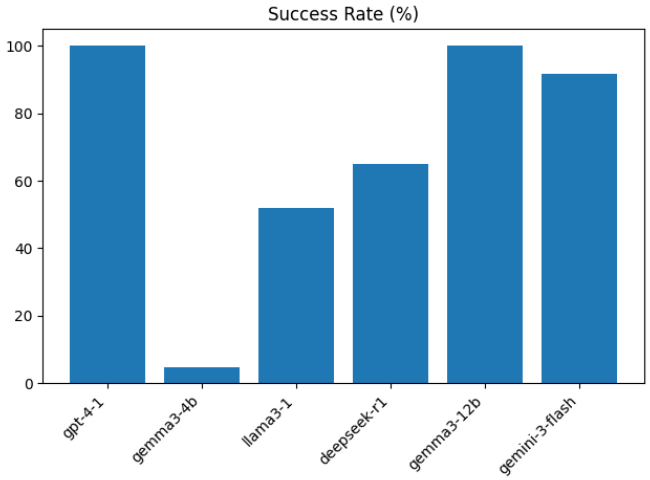

Figure 4. Successful response rate per LLM.

Figure 4 shows that Gemma3:4B achieves the lowest schema conformity success rate (5%), whereas Gemma3:12B and GPT-4.1 both reach 100% under the tested conditions. The remaining models achieve intermediate performance: Llama 3.1 52%, DeepSeek-R1 65%, and Gemini 3 Flash 92%.

V. DISCUSSION

The proposed solution is designed as an assistive tool for virtual production professionals. By partially automating the extraction and structuring of asset requirements from screenplays, the system reduces repetitive manual tasks and supports more efficient early-stage production planning.

Compared to established manual workflows in VFX production, the presented approach offers measurable practical benefits, particularly in terms of time efficiency and process consistency. The average processing time per LLM interaction is approximately 10 seconds, with total pipeline execution time depending on screenplay length and asset library size. While screenplay analysis typically requires a single LLM call, later stages, such as recommendation generation, require multiple calls due to batch processing of assets, resulting in cumulative execution time.

The evaluation results show that GPT-4.1 achieves the highest performance in terms of output validity and extraction quality. As reported in Table III and Table IV, it attains the highest F1-scores across both screenplay analysis and missing-item detection tasks, largely due to its strong recall. These findings indicate that GPT-4.1 produces more complete outputs while maintaining high accuracy under the tested conditions.

In contrast, locally deployed models demonstrate lower latency, with average response times ranging from approximately 3 to 10.9 seconds (mean: 8.85 seconds), suggesting a performance advantage in terms of inference speed.

Overall, the results indicate that LLM-based approaches can effectively support the extraction and structuring of production-relevant information from screenplays. However, performance depends on model capability, pipeline design, and the availability of production assets.

VI. Conclusion and Future Work

One notable limitation of the proposed software solution is that the assets extracted from Unreal Engine are not analyzed visually within the automated pipeline. Visual inspection is only possible manually at the final stage, after the execution of the complete workflow. By incorporating a vision–language model (VLM), the extracted assets could be further described or semantically tagged and subsequently integrated into the existing pipeline. Providing richer metadata about the visual and contextual characteristics of each asset would likely enhance the quality of the recommendation process and potentially increase precision and recall during asset matching.

After implementing and evaluating the proposed software tool, it became evident that the overall performance of the system is highly dependent on the underlying LLM used within the architecture. The experimental results demonstrate that, when suitable models are employed, such a tool can support virtual production workflows by reducing certain manual tasks during the pre-production phase. Consequently, the system has the potential to decrease the time required for asset identification and preparation.

ACKNOWLEDGMENTS

We would like to thank the Innovative Hochschule, funded by the German Federal Ministry of Research, Technology, and Space (BMFTR), for supporting this research. This study has been conducted as part of the CreatiF Center research mission.

References

[1] J. Swords and N. Willment, "'it used to be fix-it in post production! now it's fix-it in pre-production': How virtual production is changing production networks in film and television", *Creative Industries Journal*, vol. 0, no. 0, pp. 1–17, 2024. DOI: 10.1080/17510694.2024.2430025. eprint: https://doi.org/10.1080/17510694.2024.2430025.

[2] S. Das, "The evolution of visual effects in cinema: A journey from practical effects to cgi", *Journal of Emerging Technologies and Innovative Research*, vol. 10, e303–e309, Nov. 2023.

[3] P. Palanimurugan, "Challenges and solutions in green screen post- production: A narrative review", *IRE Journals*, vol. 7, pp. 147–155, Feb. 2024.

[4] P. Chanpum, "Virtual production: Interactive and real-time technology for filmmakers", *Humanities, Arts and Social Sciences Studies*, vol. 23, pp. 9–17, Jan. 2023. DOI: 10.14456/hasss.2023.2.

[5] C. Cremona and M. Kavakli, "The evolution of the virtual production studio as a game changer in filmmaking", in *Creating Digitally: Shifting Boundaries: Arts and Technologies—Contemporary Applications and Concepts*, A. L. Brooks, Ed. Cham: Springer International Publishing, 2023, pp. 403–429, ISBN: 978-3-031-31360-8. DOI: 10.1007/978-3-031-31360-8_14.

[6] H. Hogan, "Virtual production in unreal engine 5: Tools for the game and film industry", Master of Fine Arts in Digital Media Culminating Experience, East Tennessee State University, 2024.

[7] Filmustage, *Filmustage: Ai-powered pre-production software*, https://filmustage.com/, Accessed: 2026-02-05, 2026.

[8] Das Element, *Das Element: Virtual production and digital art studio*, https://das-element.com/, Accessed: 2026-02-05, 2026.

[9] L. He *et al.*, *Kubrick: Multimodal agent collaborations for synthetic video generation*, 2025. arXiv: 2408.10453 [cs.CV].

[10] Z. Xu *et al.*, "Filmagent: Automating virtual film production through a multi-agent collaborative framework", in *SIGGRAPH Asia 2024 Technical Communications*, ser. SA '24, Association for Computing Machinery, 2024, ISBN: 9798400711404. DOI: 10.1145/3681758.3698014.

[11] L. Mahon and M. Lapata, *Screenwriter: Automatic screenplay generation and movie summarisation*, 2024. arXiv: 2410.19809 [cs.AI].

[12] G. D. L. Geheeb Julian, "Pix-e", GitHub repository, 2025, [Online]. Available: https://github.com/gamedevlabs/pix-e (visited on 10/31/2025).

[13] Google DeepMind, *Gemini*, https://deepmind.google/technologies/gemini/, Large language model developed by Google, Google DeepMind, 2023.

[14] OpenAI, *Chatgpt*, https://chat.openai.com, Large language model accessed via web interface, OpenAI, 2022.

[15] Ollama Team, *Ollama*, https://ollama.com/, Accessed: 26 January 2026, 2024.

[16] M. Schulz, "Tokyo at night", Directed by Malte Schulz, IMDb page, accessed: 30.03.2026, [Online]. Available: https://www.imdb.com/de/title/tt37749777/.

[17] M. Binder, T. Fehlbaum, and A. David, "September 5 - the day terror went live", Directed by Tim Fehlbaum, IMDb page, accessed: 30.03.2026, 2024, [Online]. Available: https://www.imdb.com/de/title/tt28082769/?ref_=nv_sr_srsg_0_tt_8_nm_0_in_0_q_september%205.

[18] *Screenplay of the fff-funded cinema films „september 5" nominated for the academy award*, https://www.fff-bayern.de/pressemitteilungen/drehbuch-des-fff-gefoerderten-kinofilms-september-5-fuer-den-academy-award-nominiert/, Accessed: 2026-02-07, FilmFernsehFonds Bayern, 2026.

[19] M. Rasoulof, "The seed of the sacred fig", Directed by Mohammad Rasoulof, IMDb page, accessed: 30.03.2026, 2024, [Online]. Available: https://www.imdb.com/title/tt32178949/?ref_=nv_sr_srsg_0_tt_3_nm_5_in_0_q_the%20seed%20of%20the%20sacred%20fig.

[20] *The seed of the sacred fig*, https://www.polyfilm.at/film/the-seed-of-the-sacred-fig/, Accessed: 2025-02-07, Polyfilm, 2026.

[21] *Fab*, https://www.fab.com/, Accessed: 2025-01-28, Epic Games, Inc., 2025.

[22] O. Babalola, "Organizing, planning and developing visual style in screen directing during pre-production", *International Journal of Current Research in the Humanities*, vol. 26, pp. 349–376, Feb. 2023. DOI: 10.4314/ijcrh.v26i1.21.