TU WIEN Informatics

Approaching Under-Explored Image-Space Problems with Optimization

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Eng. João Afonso Liborio Cardoso, Master of Science
Registration Number 11937133

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn Michael Wimmer

The dissertation has been reviewed by:

First Reviewer	Second Reviewer

Vienna, 1, 2024

João Afonso Liborio Cardoso

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Eng. João Afonso Liborio Cardoso, Master of Science

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. 2024

João Afonso Liborio Cardoso

Acknowledgements

I am deeply grateful to my current supervisor, Dr. Michael Wimmer, for his exceptional guidance, mentorship, and for providing me with the opportunity to work on this fascinating research project. I am also grateful to my co-authors Bernhard Kerbl, Francesco Banterle, Paolo Cignoni, Jarret Martin, Lei Yang and Yury Uralsky for their invaluable contributions, insightful guidance, and for the collaborative and enjoyable experience of working together. I would like to express my gratitude to my colleagues, who provided a supportive research environment.

I am also grateful to the EVOCATION project for providing me with the resources and support needed to complete this work and for the opportunities to collaborate with experts in the field.

I would like to acknowledge and thank my students for their hard work and dedication. I am fortunate to have had the opportunity to work with such talented and motivated individuals.

I would also like to express my gratitude to my previous supervisors, Dr. Paul Kry and Dr. Dinesh Pai, for their invaluable guidance during my Bachelor and Master's degree, respectively. Their teachings and support have laid the foundation of my education. I am also grateful to the Canadian Government and European Commission for funding the research and educational opportunities that made this foundation possible.

Finally, I would like to extend my heartfelt appreciation to my family and friends for their unwavering love, support, and encouragement throughout my academic journey. Their belief in me and my abilities has been the foundation of my success. I am truly grateful to all those who have supported me throughout this journey and made this achievement possible.

Abstract

This doctoral dissertation delves into three distinct yet interconnected problems in the realm of interactive image-space computing in computer graphics, each of which has not been tackled by existing literature.

The first problem centers on the prediction of visual error metrics in real-time applications, specifically in the context of content-adaptive shading and shading reuse. Utilizing convolutional neural networks, this research aims to estimate visual errors without requiring reference or rendered images. The models developed can account for 70%–90% of the variance and achieve computation times that are an order of magnitude faster than existing methods. This enables a balance between resource-saving and visual quality, particularly in deferred shading pipelines, and can achieve up to twice the performance compared to state-of-the-art methods depending on the portion of unseen image regions.

The second problem focuses on the burgeoning field of light-field cameras and the challenges associated with depth prediction. This research argues for the refinement of cost volumes rather than depth maps to increase the accuracy of depth predictions. A set of cost-volume refinement algorithms is proposed, which dynamically operate at runtime to find optimal solutions, thereby enhancing the accuracy and reliability of depth estimation in light fields.

The third problem tackles the labor-intensive nature of hand-drawn animation, specifically in the detailing of character eyes. An unsupervised network is introduced that blends inpainting and image-to-image translation techniques. This network employs a novel style-aware clustering method and a dual-discriminator optimization strategy with a triple-reconstruction loss. The result is an improvement in the level of detail and artistic consistency in hand-drawn animation, preferred over existing work 95.16% of the time according to a user study.

Optimization techniques are the common thread that ties these problems together. While dynamic optimization at runtime is employed for cost volume refinement, deep-learning methods are used offline to train global solutions for the other two problems. This research not only fills gaps in the existing literature but also paves the way for future explorations in the field of computer graphics and optimization, offering new avenues for both academic research and practical applications.

Contents

Abstract **vii**

Contents **ix**

1 Introduction **1**
1.1 Motivation . . . 3
1.2 Problem Statement . . . 5
1.3 Challenges . . . 6
1.4 Contributions and Publications . . . 7

2 Background **11**
2.1 Optimization . . . 11
2.2 Applications of Optimization . . . 19
2.3 Human Vision . . . 25
2.4 Basics of Drawn Animation . . . 28

3 Predicting Perceptual Error in Real-Time Applications **33**
3.1 Metric Prediction . . . 35
3.2 Real-Time Render Mode Selection . . . 43
3.3 Evaluation . . . 46

4 Refining Cost-Volumes for Depth Prediction **53**
4.1 Minimal Pipeline . . . 55
4.2 Refinement Algorithms . . . 56
4.3 Evaluation . . . 60

5 Context-Aware Translation **65**
5.1 Method . . . 68
5.2 Ablation Experiments . . . 76
5.3 User Study . . . 81

6 Conclusion **89**

List of Figures **93**

List of Tables **95**

Bibliography **97**

CHAPTER 1

Introduction

Optimization, in the mathematical sense, is the process of finding the most effective set of parameters that yields the optimal solution to a problem, among a set of feasible alternatives. In this context, 'parameters' broadly encompass not only numerical values but also choice of algorithms, model structures, loss functions or pipelines — anything that can be adjusted to enhance the outcome or efficiency of the solution. The objective is to minimize or maximize a real-valued function, often called the objective" or cost" function, subject to certain constraints. Historical records suggest that the seeds of optimization were sown in antiquity, with early civilizations grappling with challenges in geometry and number theory using primitive optimization strategies. In 1827, Joseph Fourier first introduced a technique for resolving systems of linear inequalities, a method that later inspired the Fourier–Motzkin elimination approach [SZ15]. However, modern optimization as we know it today can be argued to have been born in the 20th century: this period witnessed the advent of linear and nonlinear programming, along with a suite of other mathematical methodologies [Sch98, Dan82]. These innovations not only reshaped the landscape of optimization but also cemented its role as an indispensable asset across diverse sectors. Today, optimization has a wide range of applications, from engineering and physics to the logistics behind global supply chains, the algorithms powering financial markets, or the research propelling healthcare advancements. Of particular significance to this work, optimization plays a crucial role in the field of applied mathematics and computer science.

In the realm of visual computing, the role of optimization already had an established foothold in computer vision, but is now also gaining prominence within computer graphics, especially when it comes to neural network training and differential rendering. Training a neural network entails fine-tuning its parameters to align closely with a specific dataset. The objective is to minimize a given measure of success, often referred to as the loss function, which quantifies the discrepancy between the network's output and the expected results. Commonly, algorithms like stochastic gradient descent are employed to

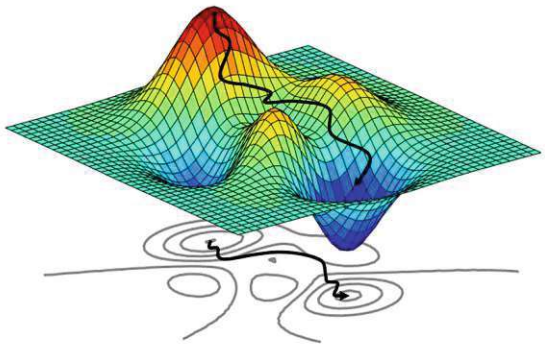


Figure 1.1: **Optimization Pervasiveness**. Searching for the arguments that result in the lowest value of a function - known as minimizing the function - is one of the computational problems commonly addressed using optimization. Figure from [Guo21].

iteratively refine these parameters and reduce this loss (see Figure 1.1). In the case of differential rendering, optimization aids in identifying the best settings for generating visuals, including factors like lighting conditions, shading techniques, and camera angles. Additionally, optimization methods have been leveraged to enhance various computer graphics algorithms, from simplifying the geometric complexity of 3D models to elevating the visual quality of rendered images.

Expanding upon these foundational applications, this thesis will delve into three specific, yet largely unexplored, problems in computer graphics where optimization techniques have great potential to be applied.

The first area of focus in this thesis is adaptive rendering mode selection in the context of increasing display resolutions and refresh rates. As hardware capabilities continue to grow, so does the complexity of shading and rendering effects. This has led to the development of both software and hardware solutions aimed at dynamically and locally reusing rendering information from previous frames or optimizing the shading resolution based on the content displayed. As neural networks are increasingly executed on GPUs, optimization-based techniques now have further potential to be crucial for the dynamic allocation of resources, ensuring that high-quality rendering is achieved without unnecessary computational overhead.

The second focus is depth reconstruction in light-field cameras, also known as plenoptic cameras. These specialized cameras offer unique post-capture refocusing capabilities but present a complex challenge in depth reconstruction. The narrow baselines of the micro-lenses make traditional techniques unsuitable, creating a need for specialized depth-reconstruction methods tailored for light-field imagery. By leveraging iterative refinement optimization strategies, there is the potential to improve upon these existing specialized methods.

The third focus is the automation of detail in hand-drawn animation, particularly in

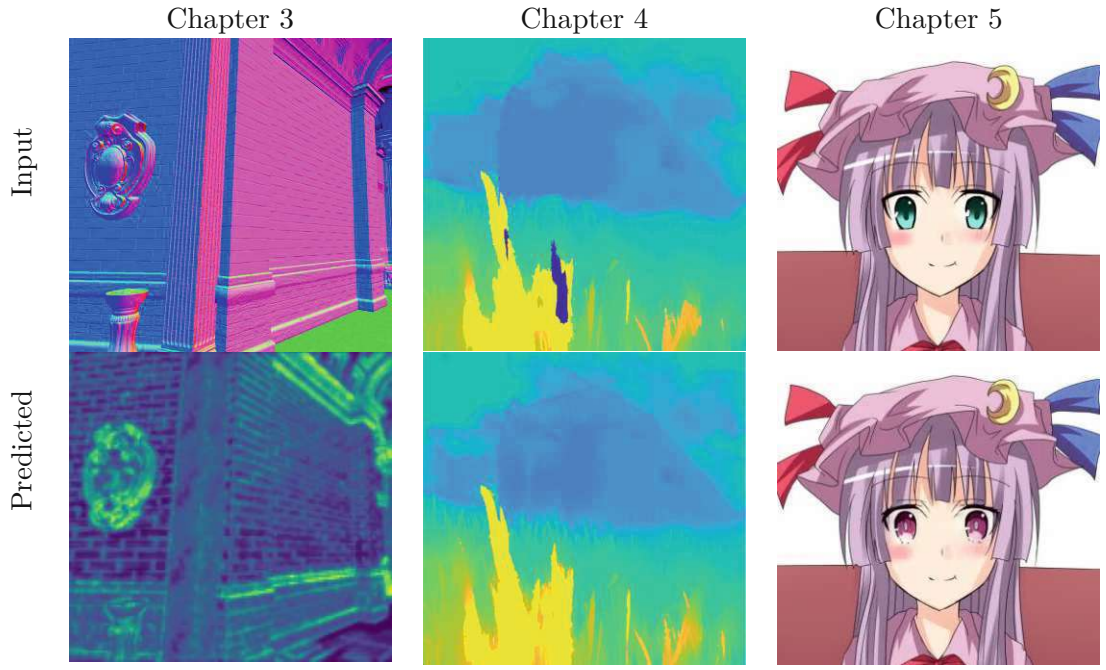


Figure 1.2: **Problem Overview**. This thesis focuses on the key theme of extrapolating new information from existing data. In Chapter 3, arbitrary perceptual metrics are predicted from screen-space material information (view-space normals and LPIPS [ZIE+18] shown). In Chapter 4, better cost volumes are inferred from existing ones (shown are the depth predictions from these volumes, due to the complexities in visually representing these 3D entities). Finally, in Chapter 5 art details are redrawn according to arbitrary provided designs.

character faces and eyes. Despite its resurgence in popularity, traditional animation has not significantly benefited from recent advances in computer graphics. The labor-intensive nature of drawing character details often leads to compromises in design complexity and artistic consistency, highlighting the need for optimization-based techniques that can streamline the workflow. Among these techniques, deep learning is particularly promising, as it has the potential to understand the abstract representation and replicate the nuances of hand-drawn styles, where traditional algorithms might struggle.

1.1 Motivation

Optimization techniques have the potential to address a broad spectrum of challenges in the field of computer graphics, ranging from algorithmic performance improvements to the generation of high-quality visuals. While optimization has long been a cornerstone in various research domains, there has been a notable intensification in its application and development in recent years (particularly with the advent of more computationally and data-intensive methods), and it is important to acknowledge that many potential use cases

for optimization techniques remain largely unexplored, as discussed in Chapter 2. By identifying and investigating different untapped use cases, this thesis seeks to uncover new opportunities and the limitations inherent to the application of optimization techniques in computer graphics problems and thus contribute to the expansion of knowledge in the field.

One such use case that exemplifies the need for further research is adaptive rendering mode selection. The quest for more realistic and interactive rendering has led to a race for ever-increasing display resolutions and refresh rates in the hardware market. At the same time, shading costs also keep increasing due to higher software shading complexity and the intricacy of effects being used. Neither trend is expected to halt, especially considering the recent introduction of GPUs with real-time ray-tracing capabilities and the surge in popularity of virtual reality (VR) headsets, as visual quality for VR requires much higher resolutions and framerates than regular screens for the same level of perceived visual fidelity.

In light of these developments, the perceptual relevance of each pixel can change drastically depending on the view configuration and the content being viewed. To exploit this fact in real-time applications, both software and hardware solutions have recently been proposed for dynamically and locally reusing rendering information from previous frames [MNV+21] or changing the shading resolution across the screen depending on the displayed content [YZK+19, Dro20]. However, the key question then becomes: how does one choose how to render each region of the screen, without knowing the end result of the shading operations for the current frame? Current methods for dynamically adjusting rendering settings are limited by their reliance on information from previous frames. This approach has two significant limitations:

1. Only metrics that can be estimated from previous renderings in a computationally efficient manner may be used. This rules out most image metrics, with simple estimates becoming the norm among perceptual problems.

2. Estimation is only possible for previously seen content. The higher the amount of motion in the scene (and thus the frequency of disocclusion of previously unseen regions), the smaller the impact of these methods becomes.

Transitioning to a second use case, depth reconstruction in light-field cameras is motivated by the limitations of current methods. Light-field cameras first became popular among professional photographers due to their ability to precisely refocus images after acquisition [NLB+05, VRA+07], but depth reconstruction remains a complex and unresolved issue. These cameras capture both the light direction and intensity emanating from a scene simultaneously. Typically, this is implemented as an array of micro-lenses placed in front of a conventional image sensor [GYLG13, PW12].

This data redundancy from the multiple micro-lenses also allows, in theory, to predict the depth of the scene from the camera. Yet, when compared to multi-camera systems,

the major limitation of light-field cameras for depth reconstruction is that all their lenses are extremely close together. This results in very narrow baselines [YGL+13, BJK17]. Thus, typical multi-camera depth-reconstruction techniques are not appropriate to use with light-field imagery. On the other hand, light-field cameras are generally cheaper than multi-camera setups, more portable, and need no synchronization between different cameras. This has sparked an interest in depth-reconstruction methods specific for light-fields.

Finally, the third use case we focus on is the automation of detail in hand-drawn animation, particularly in the eye region of characters. Traditional hand-drawn animation has seen a massive resurgence in the last decade [MSR+19] due to the increased control and freedom it affords artists, granting the medium a distinctive look. However, it has struggled to benefit from advances in computer graphics: techniques used in production remain largely the same, with productions relying on repetitive manual labor from a large workforce. As a result, studios often have to compromise character design complexity and art consistency. Character faces, and especially the eyes, are the most time consuming to draw, and thus are generally the first to be significantly simplified.

Recent advances in deep learning offer a promising avenue for breaking this impasse. Unlike traditional computer graphics methods, which have struggled to integrate seamlessly into the hand-drawn animation workflow, deep learning algorithms have the potential to adapt to the unique artistic styles and complexities inherent in this medium. By leveraging data-driven techniques, deep learning could offer more flexible and automated solutions for intricate details like character faces and eyes, thereby reducing manual labor without sacrificing artistic integrity.

In essence, each of the challenges presented poses complex problems that defy straightforward solutions. Optimization techniques stand out as a powerful tool capable of navigating these complexities to discover optimal solutions. Whether it's determining the most efficient rendering settings, fine-tuning depth estimates, or automating intricate artistic details, optimization provides a structured yet flexible framework for problem-solving. By employing optimization-based techniques, this thesis aims to unlock new avenues for innovation and efficiency in the realm of computer graphics.

1.2 Problem Statement

The purpose of this work is to advance computer graphics systems by addressing key challenges that remain unexplored in existing literature, in particular depth prediction refinement, pre-rendering visual error estimation, and character eye redrawing. By jointly focusing on these three research areas, we aim to enhance the accuracy, efficiency, and realism of computer graphics algorithms, enabling improved scene understanding, error detection, and artistic representation in various domains.

To accomplish this goal, we propose addressing three distinct problems leveraging optimization techniques, illustrated in Figure 1.2, that have not yet been adequately

investigated in the existing literature:

1. Improving rendering-resource saving techniques such as content-adaptive shading and shading reuse, by being able to accurately and consistently predict in real time the visual error incurred by future renders under different settings. The proposed method is capable of using any arbitrary perceptual metric from existing literature to predict the visual error. The goal is to achieve a balance between resource saving and visual quality, and the proposed method will be evaluated in a variety of rendering scenarios and settings.

2. Improving depth estimation in light-fields by moving estimate refinement from the traditional depth-map refinement techniques to refinement of cost volumes. Cost volumes are popular intermediary products for depth estimation, and they contain more information regarding the light-field than the traditional depth-maps. The proposed method aims to leverage this additional information to improve the accuracy of depth estimation in light-fields.

3. Improving the quality of hand-drawn animation by developing a method that provides automatic and plausible suggestions for redrawing character eyes, notorious for being the most laborious elements. The goal is to improve the level of detail and art consistency while preserving the animation artist's intended pose and expression. The proposed method will be evaluated using a dataset of hand-drawn animation frames and compared against state-of-the-art methods, in order to demonstrate its effectiveness in improving the visual quality of hand-drawn animation.

By addressing these issues, this research endeavors to fill gaps in existing literature, providing novel solutions that have the potential to advance computer graphics systems and to push the boundaries of the field of visual computing.

1.3 Challenges

In the context of this thesis, several notable overarching challenges emerge, each bearing significant implications for the research outcomes and practical applications. Addressing these challenges is crucial to ensure that the goals of this thesis are met.

First, the problems explored in this thesis offer compelling practical applications, particularly in domains where real-time or near real-time performance is of paramount importance. For example, visual error prediction can be used to optimize rasterization, but only if the savings make up for the cost of prediction; character redrawing can be used while hand-drawing, but only if results are provided at a responsive rate. **Striking an equilibrium between satisfactory output quality and the desired performance** thus becomes a central concern in the choice and design of algorithms, as allocating excessive resources may prove wasteful, while insufficient resources may compromise

the efficacy of the solutions. Therefore, navigating this delicate balance emerges as a significant challenge throughout the research process.

Another challenge arises from the **lack of established datasets**. In emerging problem spaces, acquiring appropriate datasets for ablation or optimization through training can be particularly challenging. This scarcity may arise due to limited access to relevant data sources or simply the novelty of the problem itself. For instance, while there are available open source datasets of 3D environments [Ama17, NHB17, Epi17], we were unable to find datasets specifically encompassing uniformly sampled viewpoint collections or deferred shading data. Similarly, although labeled datasets of illustrations in animation style are well established [AcB22], datasets of animation production data are not readily available. Consequently, overcoming these obstacles required the use of various strategies. Transfer learning, for instance, was employed to introduce domain-specific data in Chapter 4. Data augmentation was utilized to enhance object detection accuracy in Chapter 5. The creation of novel synthetic datasets was crucial to provide sufficient resources for training and ablation studies, and is described in detail in Chapters 3 and 5.

Furthermore, **the scarcity of existing work to draw upon for comparative analysis**, commonly known as the "ablation of results", represents a significant hurdle. In this thesis, by definition, the explored problems limit the availability of direct benchmarks and often even established methodologies for comparison. For example, a standardized approach to evaluate the efficacy of individual stages in cost volume based depth prediction is lacking, as existing work primarily compares different pipelines directly. This challenge demands the identification and implementation of suitable evaluation approaches for each contribution to demonstrate the significance and efficacy of the proposed solutions.

In addition to these overarching challenges, each of the three tackled problems encompasses a set of additional hurdles that must be overcome. These specific challenges, carefully described in later chapters of this thesis, further contribute to the depth and complexity of the research endeavor.

1.4 Contributions and Publications

The main contributions of this thesis lies in the development and evaluation of our proposed optimization solutions for three novel problems in computer graphics, as outlined in the problem statement. These solutions are not only rigorously evaluated across a range of scenarios and rendering settings but also compared against existing state-of-the-art methods.

The first set of contributions of this thesis is in the area of adaptive rendering mode selection. The work on this topic has been published in the *Proceedings of the ACM in Computer Graphics and Interactive Techniques*, under the title "*Training and Predicting Visual Error for Real-Time Applications*" [CKY+22]. The research introduces a compact Convolutional Neural Network (CNN) specifically designed for the real-time prediction of error metrics. This is a significant advancement over traditional methods, which often

rely on heuristics or are computationally expensive. Additionally, the work introduces novel metric transforms that facilitate a more balanced training loss, making the model easily generalizable across new metrics and scenes. A unique correction method is also proposed, which eliminates the need for measuring perceptual thresholds explicitly, thereby embedding these thresholds into the trained model's predictions. The work also conducts a comprehensive analysis of which current-frame screen-space data is most valuable for predicting error metrics, providing guidelines for future research in this area. Finally, the research evaluates the quality, performance, and generalizability of this learning-based approach in the context of Variable Rate Shading (VRS), setting a new standard in the field.

Following this, the thesis delves into the problem of depth reconstruction in light-field cameras using cost volumes. This second set of contributions has been published in the *25th International Conference on Pattern Recognition*, under the title "*Cost Volume Refinement for Depth Prediction*" [CGW21]. The work introduces a modular framework for cost-volume refinement, a step forward in making depth reconstruction more accurate and efficient in light-field cameras. The framework is versatile and can be applied to both regular and multi-view stereo imagery. Additionally, the research proposes a floating-point method for artifact-removal on cost volumes, which is robust to smooth surfaces and object complexity. A fast local smoothing technique is also introduced for noise and discontinuity reduction on cost volumes, which is robust to sharp depth changes. Furthermore, the work presents a method for combining different types of depth prediction methods before regression, offering a more holistic approach to depth estimation.

The third area of investigation is in the automation of detail in hand-drawn animation. This research is to be presented at the *33rd International Joint Conference on Artificial Intelligence* under the title "*Re:Draw - Context Aware Translation as a Controllable Method for Artistic Production*". Meanwhile, its contributions have already been made available on arXiv [CBCW24]. The work introduces a dual-discriminator-based training structure for adversarial training that takes into account both the original pose and design translation requirements, which we call *context-aware translation*. This is a significant advancement over traditional methods, which often struggle with maintaining the integrity of the original design. The research also introduces a triple-reconstruction loss, a novel approach that yields greater unsupervised level-of-detail generation than traditional loss functions. Furthermore, the work presents a character design recognition network that uses a production style-aware latent space. This network not only outperforms existing work but also generates a robust training dataset for adversarial supervision, setting a new benchmark in the field.

In addition to these primary contributions, the thesis also offers a review of relevant background knowledge and existing literature in Chapter 2, detailed discussions and solutions for each problem in Chapters 3, 4, and 5, and a thorough summary and discussion of the findings in Chapter 6. This final chapter also highlights potential future directions and areas for further exploration, thereby laying the groundwork for subsequent

research in this field.

CHAPTER 2

Background

This chapter explores fundamental concepts and techniques in computer graphics and optimization. It aims to provide a foundation for the research and findings discussed in later chapters. These fields encompass various theories, methodologies, and applications, which have evolved significantly over the years.

The chapter is organized into four main sections, each focusing on a specific area of knowledge that is crucial to understanding the problems we tackle in this work. The bulk of this chapter are Sections 2.1 and 2.2, which form the central focus of this discussion: Section 2.1 delves into the fundamental concepts of optimization, along with the principles and techniques that hold specific relevance to this thesis; Section 2.2 discusses the application of these optimization techniques in work related to this thesis. The remaining sections provide foundational knowledge for some of the subjects discussed: Section 2.3, offers an overview of how human perception has been studied and modeled computationally, with a focus on perceptual metrics and depth cues; Section 2.4 transitions to an exploration of drawn animation, introducing its fundamental principles and the essential components of the prevalent animation pipelines.

By the end of this chapter, readers should have a solid understanding of the critical areas and state of the art relevant to this thesis, the theoretical and practical foundations of our research, and the gaps in the current body of knowledge that our research aims to fill.

2.1 Optimization

Optimization has found applications in diverse fields. In engineering, it is used to design efficient systems, minimize costs, and improve performance. In economics, it guides resource allocation, portfolio optimization, and supply chain management. Environmental optimization focuses on waste management, energy optimization, and more. The

applications are vast and varied, and the impact of optimization is profound, shaping the way we live, work, and interact with the world around us. Current trends and challenges in optimization include big data optimization, multi-objective optimization, and real-time optimization. Handling large-scale optimization problems requires innovative algorithms and computational techniques. Multi-objective optimization deals with problems where multiple conflicting objectives must be optimized simultaneously. Real-time optimization is concerned with optimizing systems that operate in dynamic and uncertain environments.

Optimization is a mathematical field that encompasses a broad range of concepts, which has led to the development of a wide array of algorithms to solve optimization problems. These algorithms can be categorized into various groups, such as linear, nonlinear, integer-based, and multi-objective optimization. Constraints must be defined in these problems to carve out a specific feasible solution space, and it is within this delineated boundary that a search for the optimal solution unfolds. Linear programming algorithms, such as Simplex Method and the Interior Point Method, deal with the problem of optimizing a linear objective function, subject to linear equality and linear inequality constraints, while non-linear programming algorithms, such as Gradient Descent, tackle problems where either the objective function, the constraints, or both are non-linear. Metaheuristic algorithms, inspired by natural processes, are more uncommon methods used to solve complex optimization problems when traditional methods are not applicable or prove to be too inefficient. Yet, in the realm of optimization, perhaps the most groundbreaking advancement has been the integration of machine learning, particularly deep learning.

2.1.1 Regression

Regression analysis a statistical method used primarily for modeling and exploring relationships between variables and one of the more straightforward applications of optimization. It is used for several reasons: firstly, it enables the identification of significant predictors or features in large datasets; secondly, it assists in forecasting trends, which is particularly beneficial in dynamic optimization scenarios where future states of the system need to be anticipated; thirdly, regression models can be used to optimize the parameters of a system, ensuring that the system's output is maximized or minimized as per the defined objective function.

In regression analysis, the selection of appropriate models and estimation methods is crucial. Initially, a suitable model is chosen for estimation. Subsequently, a loss function and minimization method is employed for estimating the model's parameters. The components integral to regression models include:

1. Unknown parameters to be determined and used to describe the relationships between variables, commonly denoted as a vector θ.
2. Independent variables, which are observed in the data and usually represented by a vector X.

3. Dependent variables, also observed in the data and often expressed using Y.

4. Error terms, not directly observable in the data, typically denoted by ϵ.

The effectiveness of these models relies heavily on the data quality, the relevance of the selected model to the data, and the precision of the parameter estimation methods used. For example, while linear regression often utilizes least squares for analytical parameter estimation, more complex methodologies are required for non-linear models.

Linear Regression The simplest form of regression, linear regression, models the relationship between a dependent variable y and an independent variable x as a linear function:

$$y = \theta_0 + \theta_1 x + \epsilon \tag{2.1}$$

where θ_0 and θ_1 are the intercept and slope of the regression line, respectively, and ϵ represents the error term. This model can be extended to multiple linear regressions when there are multiple independent variables.

Non-Linear Regression Non-linear regression, on the other hand, is used when the relationship between the variables is not linear. This might involve polynomial terms, exponential functions, logarithms, or other non-linear transformations of the independent variables. For example, the nth degree polynomial can be represented as:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n + \epsilon \tag{2.2}$$

where θ_0, θ_1, θ_2, $\ldots$ θ_n are the coefficients of the model, and ϵ is the error term. This model can capture the curvature in the data, which a linear model (i.e., a first-degree polynomial) cannot. The choice between linear and non-linear regression depends on the nature of the data and the specific requirements of the optimization problem at hand. For example, while a polynomial regression can fit a wide range of curvature, over-fitting is more likely to occur when using higher-degree polynomials, especially when dealing with a limited number of data points.

2.1.2 Iterative Methods

When dealing with complex, non-linear, or large-scale problems where analytical solutions are either infeasible or non-existent iterative methods can be employed. These methods work by repeatedly refining an approximation of the solution, using an iterative process to gradually converge towards an optimal or near-optimal solution. The choice of an iterative method depends on the nature of the optimization problem, including the type of objective function, the presence and type of constraints, and the size of the problem.

Gradient Descent Gradient Descent is traditional method of iterative optimization, utilized for minimizing a differentiable objective function. It operates by iteratively moving in the direction of the steepest descent, defined by the negative gradient of the function, with the aim to progressively update a set of parameters θ to a better solution. Each step is defined as:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) \tag{2.3}$$

where $J(\theta)$ is a differentiable cost function, θ_t represents the parameters of the model in the current step, and $\nabla J(\theta_t)$ the current derivative of the cost function. The goal of gradient descent is to find the values of θ that minimize $J(\theta)$. Stochastic Gradient Descent is a variant of the more traditional Gradient Descent algorithm. The primary difference between gradient descent and the stochastic variant lies in the computation of the gradient. In traditional gradient descent, the gradient is computed using the entire dataset, which can be computationally expensive for large datasets. In contrast, stochastic gradient descent estimates this gradient using a single data point (or a mini-batch) randomly chosen from the dataset. Each step thus becomes:

$$\theta_{t+1} = \theta_t - \alpha \nabla J_i(\theta_t) \tag{2.4}$$

where J_i is the cost for the i^{th} data point. While this stochastic nature introduces variability, with the right hyper-parameters and learning rate schedules, it can converge to a solution effectively, making the algorithm much more efficient and the preferred choice for many large-scale learning tasks. This feature made SGD the standard algorithm for optimizing parameters in neural networks.

Conjugate Gradient The Conjugate Gradient method is particularly effective for large-scale linear systems, and generally improves convergence rates over traditional Gradient Descent in those scenarios. It selects search directions that are mutually conjugate, optimizing the process particularly for sparse systems. The update rule in its basic form is given by:

$$\theta_{t+1} = \theta_t + \alpha_t d_t \tag{2.5}$$

where x_t is the solution vector at iteration t, α_t is the step size, and d_t is the conjugate direction.

Newton's Method and Quasi-Newton Methods Newton's method uses second-order derivatives for optimization, involving the Hessian matrix H. The update rule is:

$$\theta_{t+1} = \theta_t - H^{-1}(\theta_t) \nabla J(\theta_t) \tag{2.6}$$

Here, $H^{-1}(\theta_t)$ is the inverse of the Hessian matrix at θ_t. Quasi-Newton methods, like BFGS, approximate H^{-1} to improve computational efficiency:

$$B_{t+1} = B_t + \frac{y_t y_t^T}{y_t^T \Delta x_t} - \frac{B_t \Delta x_t \Delta x_t^T B_t}{\Delta x_t^T B_t \Delta x_t} \tag{2.7}$$

with B_{t+1} being the approximation of H^{-1}, Δx_t the change in θ, and y_t the change in the gradient.

2.1.3 Deep Learning

As a subset of machine learning, deep learning has redefined the boundaries of what is achievable using iterative optimization. The essence of deep learning lies in training neural networks, a process underpinned by the backpropagation algorithm. This algorithm computes the gradient of a loss function with respect to each weight, and iterative optimization algorithms, notably Stochastic Gradient Descent, tweak these weights to minimize the loss. The synergy between deep neural networks and optimization algorithms enables the modeling of complex intricate patterns and relationships in data, bridging traditional optimization techniques with the avant-garde, data-driven methodologies of today.

Convolutional Neural Networks Given the image-centric challenges addressed in this thesis, when it comes to deep learning, in this work we are particularly interested in a specific type of network: the convolutional neural network. This category of neural networks has proven to be highly effective in a wide variety of image problems, such as image recognition, classification, or generation, becoming the de-facto standard in computer graphics and vision. While vision transformers, a type of neural network architecture inspired by natural language processing techniques and adapted for image recognition tasks [DBK+21], have gained traction as a promising approach for image processing, their computational demands are generally more intensive. This makes them unsuited for real-time scenarios and limiting in high performing or interactive ones, which are the type of problems we deal with in this thesis. Consequently, this section focuses on convolutional networks, exploring their origins, basic concepts, mathematical principles, and applications particularly relevant to this work.

The concept of convolutional processing dates back to the biological studies conducted by Hubel and Wiesel in the 1960s, which were inspired by the visual cortex in the brain. The first practical implementation of a convolutional network was by LeCun *et al.* [LBD+89], who applied backpropagation to train a network on handwritten digit recognition. A convolutional network is comprised of an input layer, several hidden layers, and an output layer. The hidden layers consist of a series of convolutional layers, interlaced by activation layers, pooling layers, normalization layers. Sometimes, fully connected layers, also known as linear layers, are also used. However, the cornerstone of this category of network is the convolutional layer, which applies a convolution operation to the input, passing the result to the next layer. This convolution operation can be represented as:

$$f_{\text{conv}}(x, y) = I(x, y) \circledast K(x, y) = \sum_m \sum_n I(x - m, y - n) \cdot K(m, n) \tag{2.8}$$

where $f_{\text{conv}}(x, y)$ is the output, $\circledast$ denotes the convolution operation, $I(x, y)$ is the input and $K(x, y)$ is the kernel, which is composed of weights to be learned during training – optimization through backpropagation. The kernel slides over the input image and multiplies its values with the corresponding values in the input, summing them up to produce the output – see Figure 2.1 for reference.

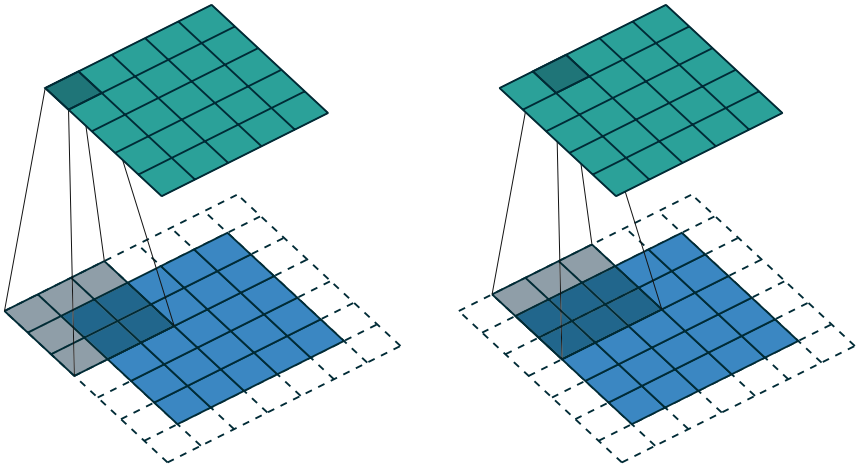


Figure 2.1: **Convolution Layer**. This illustration [DV16] depicts the convolution operation, where each pixel in the output matrix (top) is computed as a weighted sum of its corresponding local neighborhood in the input matrix (bottom). The dimensions of this neighborhood are determined by the size of the weight matrix $\mathbf{W}$. Optional padding can be applied to the input to ensure that output pixels can be centered around a greater number of input pixels. In this illustration, padding is used to preserve the dimensions of the input.

Linear Layers Linear layers, commonly known as fully connected layers, are a cornerstone in various neural network architectures, which includes convolutional networks. These layers are responsible for learning a linear transformation that maps the input space to the output space. The output is computed as a weighted sum of its input vector $\mathbf{x}$, along with a bias term b. For a weight matrix $\mathbf{W}$, composed of weight vectors for each neuron in x, the output is given by:

$$f_{\text{linear}}(x) = \mathbf{x}\mathbf{W}^T + b \tag{2.9}$$

Despite their apparent simplicity, linear layers are incredibly versatile and effective. They find applications in a myriad of tasks, ranging from traditional regression problems to the latest text transformer models.

Activation Functions While convolutional and linear layers serve as foundational elements in neural architectures, using them in isolation has inherent limitations. Specifically, the composition of multiple linear or convolutional transformations remains a linear or convolutional transformation. This means that stacking multiple such layers, would still result in a model that can only capture relationships in data a single layer could. This is often insufficient for tackling complex tasks that require understanding intricate patterns and relationships.

Activation functions are thus a pivotal component in convolutional neural networks, introducing non-linearity into the model. This non-linearity allows the network to learn from the error and make adjustments, which is essential for learning complex patterns. We will now discuss some of the most commonly used activation functions and their mathematical expressions. Rectified Linear Unit function (ReLU) has become the default activation function for neural networks because of its simplicity and effectiveness in various contexts. The function returns 0 if the input x is negative and returns the input itself if it is positive. It is given by:

$$f_{\text{relu}}(x) = \max(0, x) \tag{2.10}$$

Despite its popularity, the ReLU function has some drawbacks, such having no derivative for $x < 0$, where it can sometimes get *stuck* during training and stop updating. Another common activation function is the sigmoid, an S-shaped curve that can take any real-valued number and map it between 0 and 1. It's given by:

$$f_{\text{sigmoid}}(x) = \frac{1}{1 + \exp(-x)} \tag{2.11}$$

However, the sigmoid function is often avoided in deep networks due to the vanishing gradient problem. This challenge arises particularly in networks using gradient-based optimization methods like gradient descent and is exacerbated by network depth. The issue refers to the shrinking of gradient values to infinitesimally small numbers as one propagates backward from the output layer to the input layer during training. Specifically for the sigmoid function, its output is $\in [0, 1]$, and derivatives can become very small, causing the network to effectively *stall* or becoming *stuck* during the training process.

Normalization Functions Normalization functions play a pivotal role in the realm of machine learning and data science, serving as essential preprocessing steps to condition the data for optimal performance. These functions aim to scale, center, and sometimes even standardize the data, thereby making it easier for algorithms to learn the underlying patterns. The primary objective is to transform the features so that they conform to a standard range or distribution, which in turn helps in accelerating the convergence of optimization algorithms like gradient descent.

One of the most straightforward normalization techniques is min-max scaling, which rescales the features to lie in a given range, usually $\in [0, 1]$. Mathematically, given $\min(X)$ and $\max(X)$ as the minimum and maximum values of a feature X, the min-max normalization for a data point x is given by:

$$f_{\text{min-max}}(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{2.12}$$

Another commonly used normalization technique is Z-score normalization or zero-mean normalization. This method transforms the data into a distribution with a mean of zero

and a standard deviation of one. GIven μ as the mean and σ as the standard deviation of the a feature X, the formula for Z-score normalization is:

$$f_{\text{z-score}}(x) = \frac{x - \mu}{\sigma} \tag{2.13}$$

Batch Normalization has gained significant attention. It normalizes the output from a hidden layer in a way that the mean output activation is close to zero, and the output standard deviation is close to one. Given a mini-batch B of size m, the mean μ_B and the variance σ_B of this mini-batch and a small constant ϵ added for numerical stability, the output y is computed as:

$$f_{\text{batch-norm}}(x_i) = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2.14}$$

Normalization functions are not just limited to feature scaling and data conditioning. They also find applications in the regularization of models, preventing overfitting, and even aiding in better generalization. The choice of normalization function often depends on the specific requirements of the problem at hand, the nature of the data, and the algorithm being used. Overall, normalization functions serve as indispensable tools in the machine learning pipeline, contributing significantly to the performance and robustness of models.

Pooling Functions Pooling functions serve as a critical component in the architecture of convolutional neural networks, aiding in the reduction of spatial dimensions and computational complexity. These functions operate on each feature map separately and are designed to aggregate information in a local neighborhood. By doing so, pooling layers introduce a form of translation invariance and reduce the risk of overfitting.

The most commonly used pooling function is max pooling. Given a $n \times m$ region R in a feature map, max pooling selects the maximum value within this region as the representative value. Mathematically, max pooling is defined as:

$$f_{\text{max-pool}}(R) = \max_{x \in R} x \tag{2.15}$$

Another widely used pooling function is average pooling, which computes the average value of all the pixels in the $n \times m$ region R. The formula for average pooling is:

$$f_{\text{avg-pool}}(R) = \frac{1}{n \times m} \sum_{x \in R} x \tag{2.16}$$

In addition to max and average pooling, there are more specialized pooling functions like min pooling, which selects the minimum value in the region, or global-average pooling, which takes the entire feature map as its region to average over, and its often used as a replacement for the linear layers in classification tasks. Pooling functions are not merely dimensionality-reducing mechanisms; they also contribute to the model's ability to generalize by focusing on the most salient features and ignoring irrelevant details. The choice of pooling function can significantly impact the performance and interpretability of the convolutional neural network.

(a) Full rate (stationary) (b) Adaptive (stationary) (c) Shading Rate (stationary) (d) Adaptive (in motion) (e) Shading Rate (in motion)

Figure 2.2: **Content Adaptive Shading**. Image quality comparison of adaptive shading on and off (Wolfenstein II scene) as shown in the work of Yang *et al.* [YZK+19]. Different colored tiles in the top right image represent different shading rates.

2.2 Applications of Optimization

This section examines the areas of research relevant to this thesis where optimization plays a role. The first subsection, Real-Time Shading, discusses how optimization helps in balancing image quality with computational demands during rendering. Next, in the Cost Volumes subsection, we explore how optimization contributes to effective depth estimation in light field imaging. The section then moves to Imaging, highlighting advancements in image editing and manipulation driven by deep learning. In essence, each of these section roughly corresponds to work related to each of the three problems covered in this thesis.

2.2.1 Real-Time Shading

In the context of real-time graphics, the concept of optimization manifests itself mostly as a quest to minimize computational resources while maximizing image quality – a multi-objective optimization problem. In particular, methods for reducing the amount of shading computation required are a well-studied concept. Mixed-resolution shading [Sho09, YSL08] renders expensive and low-frequency shading components at low resolution and bilaterally upsamples the results to full resolution. This is akin to constraint optimization, where the objective function aims to maximize image quality subject to computational and memory constraints. Decoupled shading [CTM13, LD12, RKLC+11] separates the shading rate from the visibility sampling rate by establishing a mapping between the visibility samples and shading samples and sharing or reusing shading samples where possible. Texture-space shading [AHTAM14, BFM10, CTH+14, HY16] computes shading in texture or patch space in an appropriate density controlled by the mip level. These software-based techniques are available for use on a wider variety of hardware but require more complicated implementation and maintenance due to their significant deviation from the hardware rasterization pipeline.

Variable-rate shading (VRS) does not suffer from this issue, as it fits within traditional rasterization pipelines. VRS can be seen as a generalization of multi-sample anti-aliasing,

by which a single shading operation can be used to color not only multiple samples within a single pixel but multiple pixels. Software-based VRS implementations commonly divide the screen into $n \times n$ pixel tiles (where n is an integer number) and assign shading rates—the ratio of actual pixels to the number of shading operations—independently to each tile. Current hardware implementations are even more specific and operate on 16×16 tiles, with a fixed set of possible shading rates [NVI18, Int19]. Some use cases for VRS have been targets of growing interest, such as foveated rendering [Bho19, FFM+21, TAKW+19], a technique which uses eye-tracking hardware to direct rendering resources to the region the user focuses on [PSK+16], or lens-optimized shading [Kra18, YZ19], which aims at warping screen space to more closely match the final lens-corrected image [Liu17]. However, these techniques are only usable with specific peripherals, such as a VR display with eye-tracking capabilities, and do not take advantage of scene-dependent information.

Content-adaptive shading, first proposed by Yang *et al.* [YZK+19], can be seen as a real-time optimization technique that provides a more general solution, being usable in the rendering of any 3D scene. It does so by dynamically varying the shading rate across the screen according to the perceivable level of detail of the content being rendered (see Figure 2.2): the rendering result of the previous frame and the previous shading rate choices are reprojected into the current screen space and used as cues to choose the required shading rate. Drobot *et al.* [Dro20] developed a variant of this concept, designed with software-based VRS in mind. Mueller *et al.* [MNV+21] showed that shading information from previous frames can be reused for quite some time if properly sampled. Jindal *et al.* [JWMM21] proposed a more elaborate VRS-specific metric that adapts to known texture IDs. However, these techniques share several common limitations: First, they rely solely on analyzing the content from previous frames. Thus, they are unable to make predictions where reprojection data isn't available. Further, they are unable to make any predictions regarding how a surface's light response or texture aliasing might change over time, which can be especially problematic with visual edges, shiny and animated materials. Finally, due to the constraints of real-time rendering, image quality needs to be measured using a computationally efficient estimator, and some form of Just-Noticeable-Difference (JND) [TG15] threshold. Thus, these methods have to rely on multiple approximations, leading to imprecise shading-rate decisions, which, in theory, could accumulate error over time. In practice, adaptive shading is only used after significant engine- and scene-specific tuning, such as ensuring it is only enabled in highly diffuse materials.

2.2.2 Cost Volumes

Cost volumes are a tool that can be used when an optimization problem seeks to balance computational efficiency with required precision. In particular to this thesis, their use for depth estimation from light fields has been an active research topic over the past few years, and for which a large body of recent research already exists.

We now explain the basic concepts surrounding cost volumes and how they are generally used. Cost volumes serve as a three-dimensional objective function to be minimized in

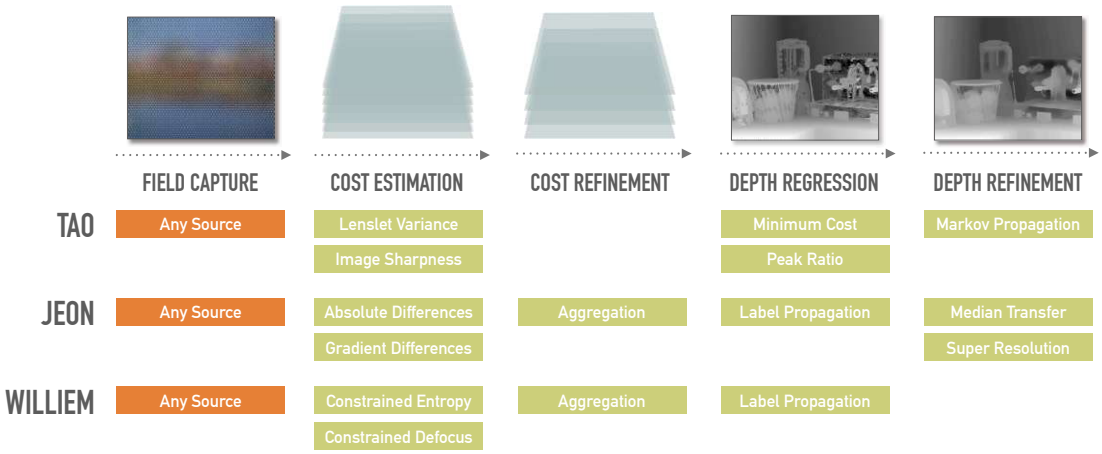


Figure 2.3: **Depth Estimation Pipelines**. Comparison of the pipelines proposed by Tao *et al.* [THMR13], Jeon *et al.* [JPC+15] and Williem *et al.* [WWL18], and how they fit within the main five stages for depth prediction using cost volumes from light-field images. Images are only illustrative.

this optimization problem, and are typically generated by measuring a visual cue across a range of depths, thereby defining the feasible solution space for depth estimation. Let $C(\mathbf{u}, z)$ be some cost volume, a three-dimensional function parameterized by the image coordinates u and depth z that, when minimized along the depth axis, should result in an accurate prediction D_C of ground-truth depth D:

$$D(\mathbf{u}) \simeq D_C(\mathbf{u}) = \underset{z}{\operatorname{argmin}}\, C(\mathbf{u}, z) \tag{2.17}$$

The pipelines used by cost-volume based depth prediction methods can be generalized to 5 stages: first, a cost volume is estimated from depth-cue(s) in the light-field – see Section 2.3.2 for information on common cues. Second, the cost volume might be refined to improve further predictions. Third, a depth-map is estimated from the cost volume. While directly minimizing the cost volume, as in Equation 2.17, is the simplest possible depth estimation optimization method, more complex methods have been proposed. Fourth, the depth prediction itself might be refined using refinement methods for range images. Finally, camera properties and calibration are taken into account to compute a map of calibrated depth from the depth prediction. We illustrate all stages in Figure 2.3 and how some of the existing work we will mention fit into them.

Yet, our primary focus with cost volumes lies not in how these are generated, but rather in how they are processed and how depth is regressed from them. Depth estimations from existing work often exhibit sharp discontinuities, which are particularly problematic in flat and smooth surfaces. This occurs due to a strict trade-off between computational efficiency and cost precision, as both are tied to the number of layers of the volume. For

example, classification methods can solve ambiguities in the cost volume, in particular in out-of-focus background regions, and thus remove unwanted artifacts. However, they can also further exacerbate discontinuity artifacts by reducing the number of possible depths to a limited set of classes, or create new artifacts of their own due to mislabeling. Both Jeon *et al.* [JPC+15, JPC+19] and Williem *et al.* [WWL18] suffer from this issue, as they perform multi-label optimization, using graph cuts [KZ02], to propagate SIFT [Low04] feature matches.

To compensate limitations of the regression methods, previous work often performs refinement operations on the disparity or depth maps obtained from regression. After their multi-label regression, both Jeon *et al.* [JPC+15, JPC+19] and Williem *et al.* [WWL18] perform median weight transfer, followed by an iterative spatial-depth super-resolution method first proposed by Yang *et al.* 2007 [YYDN07].

Different cues can also be integrated. For example, Jeon *et al.* [JPC+19] use four different cues, which result in a total of 16 cost volumes, to compensate for each other's shortcomings. Defocus and correspondence are the most often combined [LCBKY15, KGB95, SYT97]: Tao *et al.* [THMR13] first combined these two cues using Markov random field propagation with the Peak Ratio, introduced by Hirschmˋuller *et al.* [HIG02], as the confidence measure. Later, Tao *et al.* [TSM+15] improved it by also using a shading constraint as the regularization term.

However, depth refinement methods are unable to take full advantage of the light-field properties, as they are reduced to work in 2D color and depth space. Due to the lack of information, we found they are prone to creating artifacts or misreading the shape of the scene. It is also important to point out that recent deep learning based methods have shown promise. For example, Shin *et al.* [SJY+18] proposed a fully convolutional network capable of estimating depth from epipolar images. Zhou *et al.* [ZZL+19] proposed three unsupervised loss functions, which remove the need for large amounts of ground-truth data for training.

2.2.3 Imaging

In the rapidly evolving field of using convolutional neural networks for visual content, remarkable strides have been made that are revolutionizing the way we can edit and manipulate images. One of the seminal works that ignited this transformative journey was the work by Gatys *et al.* [GEB16], which showed how to edit an image by varying its overall style by transferring the style of a target image using deep learning optimizations. This work has generated a prolific field [HB17, KLA19, KLA+20, KAL+21, KLA21].

Image-to-image translation, the task of converting an image from one representation to another, has been a topic of significant interest in recent years. This task can involve a wide range of functions, such as converting a daytime scene to a nighttime scene, changing the season in a landscape photo, or converting a photo into a painting in the style of a famous artist. Lee *et al.* [LTH+18] proposed a method for producing diverse outputs without the need for paired training images by embedding images onto two

(a) Input (b) Inpainted (c) Input (d) Inpainted

Figure 2.4: **Inpainting**. Images with marked missing regions and the corresponding inpainted results, as shown in the work of Liu *et al.* [LRS+18].

spaces: a domain-invariant content space and a domain-specific attribute space. Liu *et al.* [LHM+19] introduced FUNIT, a few-shot unsupervised image translation model based on MUNIT [HLBK18]. Compared to earlier image-to-image translation methods [ZPIE17, IZZE17], this approach has the ability to translate from an unseen domain with unpaired data. FUNIT achieves this by coupling an adversarial training scheme with a novel multi-task adversarial discriminator. Park *et al.* [PLWZ19] introduces a new layer for synthesizing photo-realistic images given an input semantic layout. Lee *et al.* [LLWL20] presented a framework for diverse and interactive facial image manipulation. Park *et al.* [PEZZ20] proposed maximizing mutual information between corresponding input and output patches, using a framework based on contrastive learning.

Image inpainting, the task shown in Figure 2.4 of predicting missing parts of images, has also seen significant advancements with the application of deep learning techniques. Liu *et al.* [LRS+18] proposed a method for image inpainting using partial convolutions, where the convolution is masked and normalized to be conditioned on only valid pixels. They also introduced a mechanism to automatically generate an updated mask for the next layer as part of the forward pass, demonstrating the efficacy of training image-inpainting models on irregularly shaped holes. Yi *et al.* [YTA+20] presented a residual aggregation mechanism for ultra high-resolution image inpainting, which reduces the cost of memory and computing power, and also alleviates the need for high-resolution training datasets. Nazeri *et al.* [NNJ+20] proposed a two-stage adversarial model that consists of an edge generator followed by an image completion network. The edge generator hallucinates edges of the missing region of the image, and the image completion network fills in the missing regions using these hallucinated edges as a priori. Liu *et al.* [YTA+20] proposed a mutual encoder-decoder network for joint recovery of both structures and textures, effectively removing blur and artifacts caused by inconsistent structure and texture features.

Illustration and Animation Deep learning techniques have started to be applied to comic book [AIK18] and illustration editing; most literature focused on the colorization of either sketches or shaded manga drawings and main line extraction. Regarding this latter

topic, Simo-Serra *et al.* [SSISI16] proposed one of the first deep learning methods based on a simple encoder-decoder CNN architecture for sketches. Li *et al.* [LLW17] generalized the approach for patterns in drawings by employing a U-Net, a type of convolutional neural network known for its 'U'-shaped architecture, which includes skip connections to merge low-level detail information from early layers with high-level features in deeper layers. This design enhances the network's ability to capture both local and global patterns effectively. Line extraction methods can be improved when they are paired with user inputs [SSII18b] or adversarial training [SSII18a] or unpaired data with the synthesis of paired ones [LKY+19].

Concerning colorization, Yuanzheng *et al.* [CMW+18] employed a conditional GAN to color illustration line art using scribbles from the user. This approach was improved by Zhang *et al.* [ZLW+18], who proposed a two-stage sketch colorization for illustration: first, the user marks colors as points on a sketch and the system generates a colorized draft using a GAN; the user can then correct mistakes on the draft with further color points which are propagated using a refinement GAN. The illustration dataset by Zhang *et al.* [AcB22] was used for training, and it has also become a staple for many other methods. A different approach for manga colorization was proposed in [SdCJM19, SFO+21], where the user, instead of scribbling or splashing colors, provides an input image with basic colors. In particular, Shimizu *et al.* [SFO+21] showed that providing a flat colored image of the sketch image can generate high-quality colorization with little training data. Note that this flat-filling colorization can also be automated using deep learning [ZLS+21] and user inputs for complex line art. In this domain, researchers have also focused on specific parts to colorize using user inputs [AMT20], or combined with text tags [KJPY19]. While at first, this might appear appropriate for animation, as they do not require input per frame, they suffer from limitations when it comes to artistic control. Both remove control of lighting conditions, and the first removes control over pose and expression by virtue of being an in-painting method, while the second entirely removes control over shading. Akinobu *et al.* [MKS+21] proposed the first colorization method tailored for anime production, using an ad-hoc sampling method for patches that is capable of associating patches throughout frames in the same shot and thus transferring coloring on manually colored reference frames to other frames in the same shot – see Figure 2.5.

An emerging topic is manga generation using adversarial training and some character parameters [JZL+17], photos [WY20, SNL+21], and sketches [YYP19]. Although they can generate high-quality results, they lack precise artistic control; especially for modifying existing drawings.

Recently, researchers have applied classic methods such as segmentation [ZJL20] and clustering [NRS22] to illustration and cartoon art. Nir *et al.* [NRS22] proposed a method to learn a style-specific semantic for animated content using self-supervision. They introduce a semantic clustering based on a CNN and self-supervision where style is encoded in the training. However, it needs to be trained separately on each production and is more suited for tracking characters and not their designs.

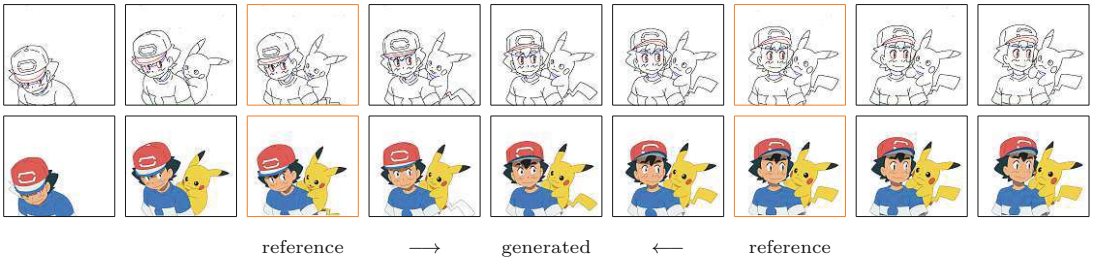


Figure 2.5: **Colorization**. The method proposed Akinobu *et al.* [MKS+21], as shown here, is a rare example of colorization that fits within the hand-drawn animation needs and constraints. Final line drawings are colorized based on manual reference colorings performed on select frames of the same shot.

Drawing these threads together, it's evident that while there has been a significant amount of research dedicated to comic books and illustrations, the field of hand-drawn animation is comparatively underrepresented. Despite its unique production challenges, very few methods have been developed that are compatible with the nuances of anime production. Even rarer are generalizable methods that do not require fine-tuning to adapt to each production's distinct style.

2.3 Human Vision

One of the main challenges that face the field of computer vision is replicating human visual perception, enabling algorithms to interpret and interact with the world in a way similar to human cognition. Thus, modeling human perception in computer vision goes beyond recognizing patterns or objects; it involves understanding how humans perceive, interpret, and respond to visual stimuli.

While human perception is a widely studied topic, it is also a complex interplay of physiological processes and cognitive interpretations. The human eye captures light, processes it through a series of photoreceptor cells, and then transmits this information to the brain. The brain, in turn, deciphers this data, extracting meaningful patterns, colors, motion, and depth.

2.3.1 Perceptual Metrics

Computationally measuring the multifaceted nature of human perception has led to the development of perceptual metrics. These metrics aim to quantify the visual quality of images and videos in a manner that aligns closely with human judgments. General error metrics, such as Mean Squared Error (MSE), have been found lacking in capturing the nuances of human perception. As a result, the concept of the perceptual metric has emerged, considering factors like luminance, contrast, and structural integrity, to better resonate with how humans evaluate visual quality.

(a) Reference (b) Degraded (c) Aggregated Users Markings

Figure 2.6: **Perceptual Metrics**. Image pair from the dataset proposed by Wolski *et al.* [WGY+18], displaying a reference image, its distorted counterpart and annotations of user-identified visual discrepancies. Perceptual metrics attempt to computationally approximate precise per-pixel markings or overall score of the visual difference perceived by humans of arbitrary image pairs.

Conventionally, a reference perceptual metric $f(I, J)$ aims to compute the perceptual difference between a reference image I and a candidate image J, while no-reference methods $f(J)$ guess perceptual issues given the expected proprieties and common distortions in natural images. Values may be computed for the entire image domain or regions thereof. The most commonly referenced metrics in image quality assessment include Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). PSNR, while not a perceptual metric, is frequently used for its simplicity and historical significance in signal processing. It is derived from the Mean Squared Error and applies a logarithmic scale, coincidentally aligning with the logarithmic nature of human vision. For a color value range $\in [0, 1]$, it can be defined as:

$$f_{\text{PSNR}}(I, J) = 10 \cdot \log_{10} \left(\frac{N}{\sum_i^N [I_i - J_i]^2} \right) \tag{2.18}$$

where I_i and J_i are pixels in the original and reconstructed images, respectively, and N is the number of elements (pixels) in the images. A higher PSNR value generally indicates better quality, but it may not always correlate well with human perception. SSIM is a metric that considers luminance, contrast, and structure to evaluate the similarity between two images. Unlike MSE and PSNR, which focus on pixel-wise differences, SSIM aims to compare the structural information of the images as a whole:

$$f_{\text{SSIM}}(I, J) = \frac{(2\mu_I\mu_J + 0.01\lambda_r)(2\sigma_{IJ} + 0.03\lambda_r)}{(\mu_I^2 + \mu_J^2 + 0.01\lambda_r)(\sigma_I^2 + \sigma_J^2 + 0.03\lambda_r)} \tag{2.19}$$

where μ_I and μ_J are the means, σ_I^2 and σ_J^2 are the variances, $\sigma_{I,J}$ is the covariance, λ_r is the dynamic range of the color values and the constants avoid instability when denominators are close to zero.

However, by expanding upon the foundational concepts of perception and traditional metrics, recent research has introduced more advanced options, paving the way for

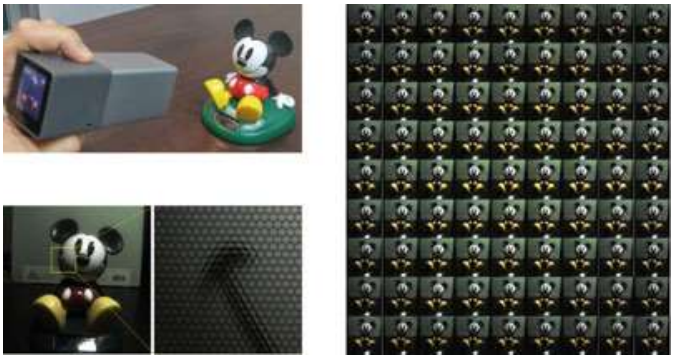


Figure 2.7: **Light-Field Imagery**. The raw photos taken by light-field cameras show an array of lenses focusing on different parts of a scene. Yet, by slicing them into sub-aperture images, we can see these images are equivalent to multi-view images with very narrow baselines. Figure from [JPC+19].

more nuanced and accurate evaluations of visual quality. The following publications are highlighted: Anderson *et al.* [ANAM+20] presented the FLIP estimator, inspired by models of the human visual system and designed with particular focus on the differences between rendered images and corresponding ground truths. Zhang *et al.* [ZIE+18] discovered that, during neural image classification, the intermediate image representation produced by the classification network could be used for comparison with other images. Wolski *et al.* [WGY+18] created a data set of image pairs with user markings of where they perceive distortions (example shown in Figure 2.6) and a convolutional network trained on it capable of predicting markings in new images. However, all of these elaborate metrics share the drawback of remaining inaccessible in real-time environments, with more naive metrics such as the one proposed by Yang *et al.* [YZK+19] being used in those scenarios instead.

This ongoing development of more advanced techniques to more closely mirror human perception underscores the complexity of human vision and the ongoing challenge of balancing perceptual accuracy with computational efficiency, a theme central to the first use-case of optimization approached in this thesis.

2.3.2 Depth Cues

Depth perception plays a crucial role in how we interpret the world around us. When capturing multiple images of the same scene, the interrelationship among these images provides valuable insights into the spatial structure of the scene itself. Drawing parallels to the human ability to gauge depth through stereo vision, computer vision methodologies have been developed to mirror this innate capability. Commonly adopted heuristics, such as defocus, correspondence, and epipolar plane analysis, serve as pivotal cues for discerning depth in multi-view imagery.

In this section, we delve into these depth cues, placing special emphasis on their use in

light-field images, a topic of particular significance to the second problem tackled in this thesis. As highlighted in Chapter 1, light-field images capture both the direction and intensity of light from a scene. Essentially, they are multi-view images with minuscule disparities between each perspective (see Figure 2.7).

Defocus serves as a depth cue by measuring the optimal local sharpness at various focus distances. This is achieved by capturing images refocused at different depths, as detailed by Tao *et al.* 2013 [THMR13]. In traditional photography, this implies taking the same photo with the camera focused at different depths, while light-field photos have the advantage that a single image can be refocused through computation alone. An adaptive defocus response, proposed by Tao *et al.* 2015 [TSM+15], extends this cue to enhance robustness against occlusion. Taking it a step further, Williem *et al.* 2018 [WWL18] introduced the concept of constrained adaptive defocus, which offers invariance to noise and occlusion, making it even more effective for depth estimation.

Correspondence refers to the process of finding matching points on different images that represent the same point in the scene. However, in the case of light-field images, these generally have very narrow baselines, which cause stereo correspondence matching to obtain sub-par results due to sub-pixel shift [LLC+10]. Thus, correspondence generally refers to angular patch-based estimation methods, even though standard multi-view stereo data cost, calculated from the sum of absolute differences, is also used [TSM+15]. Jeon *et al.* [JPC+15] used the phase-shift theorem to estimate sub-pixel shift in the image frequency domain. Tao *et al.* 2013 [THMR13] estimated correspondence as the variance in the angular patch of refocused images. As they did for the defocus cue, Williem *et al.* 2018 [WWL18] proposed constrained angular entropy, a cue invariant to noise and occlusion.

Epipolar plane image analysis, related to the concept of angular estimation, is specific to light-field images, as it refers to slicing an image along the epipolar planes, and taking advantage of properties of the resulting images to estimate properties [BBM87]. In particular, light-field images tend to form diagonal lines whose angles are linearly related to depth [MTLP96]. However, not only is this method only usable in problems that deal with light-fields, these diagonal lines only allow for sparse estimations, making them sub-optimal for problems that require dense depth information.

2.4 Basics of Drawn Animation

Animation, in its essence, is the art of breathing life into static images, allowing them to move and interact in a dynamic sequence. Historically, this art form has been deeply intertwined with human culture, evolving from primitive cave paintings that hinted at motion to the intricate hand-drawn animations we recognize today. Understanding the basics of drawn animation is crucial to this thesis as it lays the foundation for exploring the optimization of animation production pipelines – the third use case approached in this thesis.

Drawn animation, unlike other forms of animation that rely on stop-motion or computer-generated imagery (either 3D or 2D), is predicated on the sketching of each frame to create movement. This creates a visual storytelling medium that is both evocative and distinctly unique. The production of this type of animation is complex, time consuming and requires the combined efforts of many professional artists and technicians, specializing in various fields. This results in the need for pipelines with precisely defined steps to allow the easy exchange of work between different artists.

At the core of drawn animation lies the depiction of characters, and two critical aspects define a character's appearance and interaction in any frame: pose and expression. A character's pose refers to the arrangement or posture of their body, highlighting their stance, orientation, and position in relation to other objects or characters in the scene. It provides crucial context about the character's current activity or intention. On the other hand, a character's expression pertains to the portrayal of their emotions or sentiments, primarily through facial features but also influenced by body language. Expressions offer a direct window into a character's mindset, feelings, and reactions, making them pivotal for conveying narrative elements and driving viewer empathy.

Nowadays, the vast majority of drawn animation produced is limited animation, which is a sub-category within drawn animation that represents an ingenious blend of efficiency and artistry. Instead of redrawing every frame from scratch, as done in full animation, limited animation re-purposes the moving parts, also known as cels, across multiple frames, as shown in Figure 2.10. While full animation is generally more fluid, limited animation not only economizes the production process but also lends itself to more stylistic direction choices and characteristic animation styles.

A foundational step in the production of limited animation encompasses the creation of a storyboard and color guides. The storyboard is a sequence of rough sketches that sets the narrative trajectory, acting as a comprehensive blueprint for the animation's progression. In parallel, color guides are meticulous documents that communicate how the characters should be drawn, as to standardize their appearance, including their shape, colors, poses and expressions, ensuring consistency throughout the production. Integral to the process, both the storyboard and color guides collectively serve to articulate the intended visual and thematic objectives to the entire team. This alignment is essential to maintain a consistent art style – a distinctive combination of design, color, shading, and animation techniques that gives each production its unique visual signature. Typically, each production adheres to a singular art style, ensuring a cohesive visual experience for the viewer.

The standard limited animation pipeline can be roughly divided into 4 main stages: drawing keys, inbetweening, coloring and compositing, as illustrated in Figures 2.8 and 2.9. Initiated by the storyboard, the layout emerges, setting the groundwork for planning frame sequences and drafting key frames. These key frames undergo multiple reviews and alterations, and might be passed between various artists. Key frames are pivotal frames that define the start and end points of any motion or transformation. By focusing on these frames, animators can create the illusion of movement without having to draw

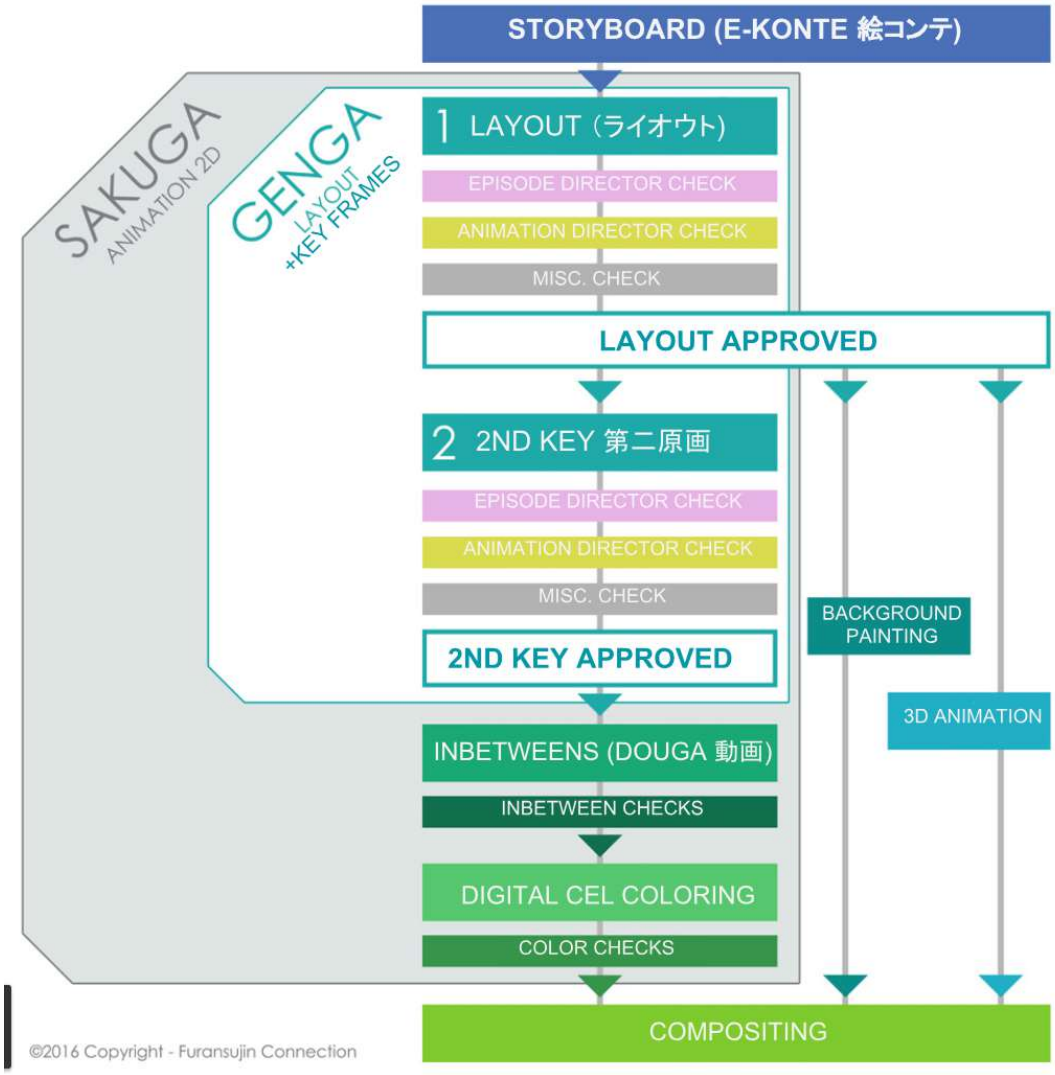
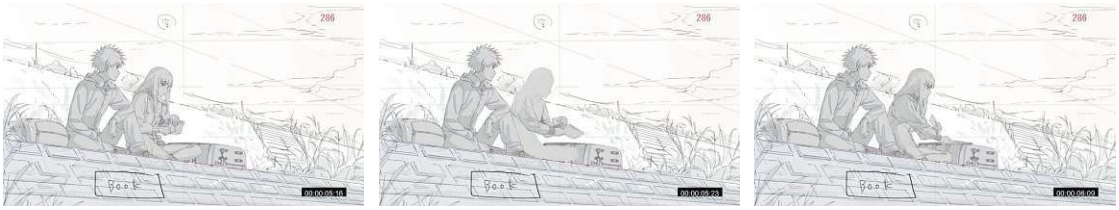


Figure 2.8: **Animation Pipeline**. Diagram of the standard production workflow for limited drawn animation [Fur16]. In this thesis, we mostly interact with the color check and compositing stages.

(a) Colored Cell (b) Final Composition

Figure 2.9: **Animation Frames**. Each frame in drawn animation is composed of two major components: the foreground, also known as cell, which is generally composed of characters and objects they interact with; and the background, which contains the scene and static objects. Artwork provided by OtakuVS *et al.* [OT21].



(a) Fully Drawn Key (b) Partially Drawn (c) Fully Drawn Key

Figure 2.10: **Limited Animation**. In this type of drawn animation, characters might only be partially drawn, with the intention that parts of drawings from previous frames will be re-purposed. In this example from [IKT23], only the arms and book were drawn for the inbetweens, as the face and body remained static.

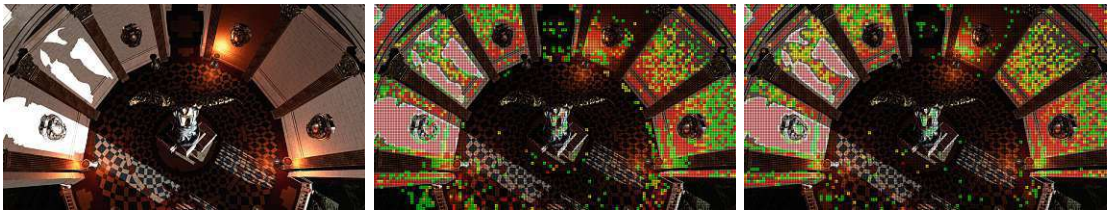
every intermediate frame. The inbetween frames, which are frames that fill in the gaps between the key frames, are then added and together with the key frames are drawn in a strict format that contains information on how elements should be handled by the coloring team. Starting with coloring, all operations are digital, while paper is still prevalent in the previous stages (see Chapter 5 for more information from our industry survey). Finally, compositing assembles the drawn animations with other elements, such as background or 3D digital effects.

CHAPTER 3

Predicting Perceptual Error in Real-Time Applications

In this chapter, we delve deeper into the topic of adaptive rendering mode selection, initially introduced in Chapter 1. As previously discussed, much of the content in this chapter has been published in the Proceedings of the *ACM in Computer Graphics and Interactive Techniques* – titled '*Training and Predicting Visual Error for Real-Time Applications*' [CKY+22]. We explore the pressing need for innovative solutions in dynamic rendering adjustments. This need is fueled by the escalating demands for higher display resolutions and refresh rates, particularly in applications involving real-time ray tracing and virtual reality (VR). Traditional methods for rendering adjustments often fall short, especially when faced with challenges such as rapid scene motion or fluctuating view configurations. It is here that optimization-based techniques could come into play, offering the potential to more accurately adapt rendering modes in real time, thereby improving both performance without compromising visual fidelity.

Our goal is to enable the use of arbitrary screen-space visual metrics in real-time applications and their efficient prediction, even for previously unseen regions of the scene resulting from, e.g., fast camera movement. This forms an optimization problem where the objective function is the minimization of the error when predicting these metrics. To find a solution to this minimization problem, we propose a convolutional neural network (CNN) that takes as input both reprojected renders, similarly to previous work and current-frame screen-space information that is often readily available in G-buffers before final shading, such as material properties or light visibility buffers. We demonstrate this approach by applying our network to solve the broad problem of adaptive rendering mode selection: given a viewport that is divided into equally sized tiles, select the suitable fidelity mode for each one. By approaching the problem in this way, our solution becomes applicable as a selection mechanism for most rendering settings that could be varied across the screen. Possible examples in current hardware include variable-rate shading



(a) Suntemple scene (full render)

(b) VRS with JNFLIP (network predicted)

(c) VRS with JNFLIP (ground-truth)

Figure 3.1: **Chapter Overview**. The network described in this chapter predicts image error metrics for real-time use cases, such as variable rate shading (VRS). The shading rate (□ full, ■ fine, ■ medium, ■ coarse) is selected for each image region based on a neural network's prediction from G-buffer data. In addition to established metrics (e.g., FLIP), we can learn Weber-corrected variants (JNFLIP) that respect perceptual context.

(VRS), software multi-sampling, temporal shading reuse, and hybrid rendering. By enabling consistent prediction of arbitrary metrics on the entire screen regardless of scene motion, we also open the door for new methods, use cases, and perceptual metrics to appear in a real-time context.

Metric prediction for seen and unseen regions as a learning effort confronts us with novel challenges: balanced selection of training samples becomes non-trivial since conventional data preparation methods cannot be applied. Furthermore, for many practical use cases (including VRS), perceptually correct threshold values may be required, which cannot be measured for unseen regions. In this chapter, we present solutions to these challenges and, as a proof of concept, use this approach to implement content-adaptive VRS. The discussed contributions are:

1. A compact CNN for learning and predicting error metrics in real-time applications for seen and unseen regions.
2. Two metric transforms to produce a more balanced training loss that easily generalizes for new metrics and scenes.
3. A correction to metrics that removes the need for explicitly measuring perceptual thresholds, embedding them into the trained models' predictions.
4. Analysis and discussion of which current-frame screen-space data is most valuable for predicting error metrics.
5. An evaluation of achievable quality, performance, and ability to generalize our learning-based approach for VRS with the current state of available hardware support.

Section 3.1 describes our network and how to train it to consistently achieve high-accuracy image-error estimation in real time. Section 3.2 describes how to use the network in the

context of adaptive rendering mode selection, including a concrete example for application to VRS (see Figure 3.1). Finally, Section 3.3 considers the performance and quality aspects of our approach and provides an analysis of the obtained results.

3.1 Metric Prediction

Conventionally, a reference image metric $f(I, J)$ computes the perceptual difference between a reference image I and a candidate image J. No-reference methods $f(J)$ guess perceptual issues given the expected proprieties and common distortions in natural images. Values may be computed for the entire image domain or regions thereof. In this thesis, we aim instead to estimate $f(I, I')$, where I' represents an informed approximation of the reference I, such as a lower-resolution rendering of I. Our goal is to predict $f(I, I')$ directly, without explicitly computing either I or I' by exploiting other, more easily available screen-space scene information instead.

Our deep learning-based approach enables fast prediction of complex metrics that would otherwise incur significant computational overhead. However, one challenge to overcome is the sensitivity of machine learning to unbalanced training data sets; another is that the practical applications of $f(I, I')$ often involves spatially varying parameters, e.g., the local just noticeable difference (JND) at each point in I [YZK+19]. These challenges and performance requirements can be seen as the constraints in our optimization problem, where the objective is to minimize the prediction error while satisfying these constraints. In this section, we introduce our network architecture, discuss which input data should be used to predict metrics, and present our solution to the output imbalance problem. Furthermore, we show how the spatially varying JND threshold can be integrated directly into the trained model. For the sake of brevity, the visual illustrations of our approach will focus exclusively on the example of predicting the error when I and I' vary in shading rate. In the figures displayed in this article, plotted or color-coded values of $f(I, I')$ show the difference between reference I and corresponding I' obtained with coarser 2×2 shading rate for a given metric f.

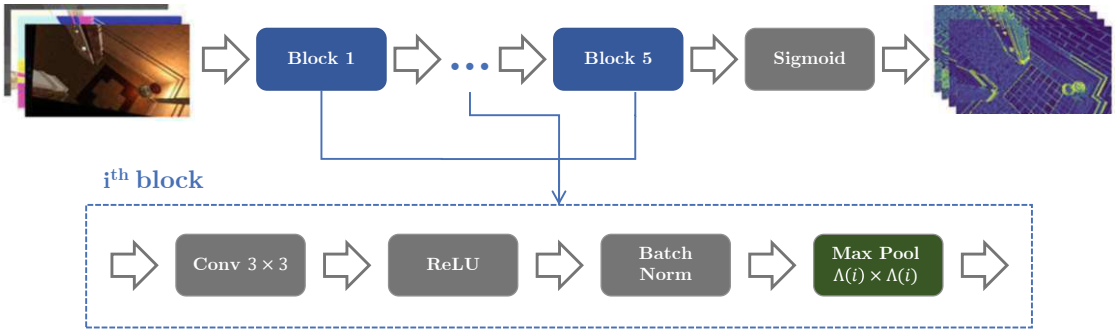


Figure 3.2: **Predictor Architecture**. Proposed network architecture for prediction of perceptual metrics. Each block i performs down-pooling at size $\Lambda(i)$.

Table 3.1: **Predictor Parameters**. Example of the dynamic parameters values for the layers of the predictor network given $w = 16$.

Layer	In Channels	Out Channels	Groups	Pooling Factor
1	# input data	16	1	1
2	16	16	1	2
3	16	16	4	2
4	16	16	8	2
5	16	# predictions	1	2

3.1.1 Convolutional Network Architecture

Figure 3.2 shows the schematic of our convolutional network architecture. It consists of 3×3 convolutions, interlaced with rectified linear units (ReLU) and batch normalization, and with a stride, padding, and dilation equal to 1 to ensure the image output size of the convolution matches the input size. We optimize for prediction performance by pooling as early as possible in the network and maintain a consistent amount of parallelism by dividing hidden channels into independent groups at the same rate at which down-pooling is performed—that is, we try to keep the number of independent groups times the number of pixels remains the same. We chose a latent channel dimension of 16 as our testing showed us that it is the lowest one can pick before there is an evident loss of prediction quality. However, one can go as low as 8 latent channels before the network becomes unusable. A single final sigmoid layer is used to constrain the output to the range $[0, 1]$.

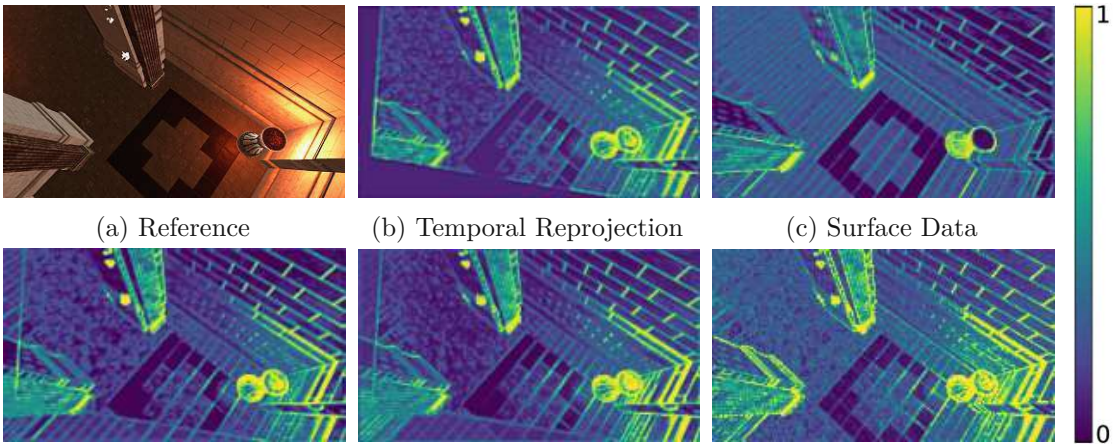
To support optimized generation of (conservative) predictions for arbitrarily-sized image regions (e.g., for application to hardware VRS), maximum pooling is done depending on an intended region size $w \times w$ (for per-pixel predictions, $w = 1$). The size $\Lambda(i)$ in down-pooling layer i is:

$$\Lambda(i) = \begin{cases} 2 & \text{if } \frac{w}{2^i} > 2 \text{ and } i < 5 \\ \lceil \frac{w}{2^i} \rceil & \text{otherwise} \end{cases} \tag{3.1}$$

As an illustrative example, Table 3.1 shows how grouping and max-pooling is used throughout the network five layers when the intended output tiling size is 16×16 pixels ($w = 16$).

The design of our network is governed by its intended use in real-time applications: given sufficient training time, the network is capable of learning sophisticated features while prediction remains fast. Its layout makes it compatible with optimized, massively parallel inferencing solutions, such as TensorRT. Furthermore, the per-region predictions for $w > 1$ can be passed on directly to tile-based procedures. Training was performed using Root Mean Square Propagation [Hin12].

We converged on our eventual design after comparing more complex alternatives, which all underperformed or provided no visible benefit over the simpler solution. These



(a) Reference (b) Temporal Reprojection (c) Surface Data

(d) Both (e) Optimal 4 Channels (f) Target

Figure 3.3: **Input Comparison**. Network predictions for FLIP error between the full-resolution reference image and a coarser, 2×2-shaded versions. Results were obtained from networks trained with different screen-space input sets. Number of input channels used are 3, 16, 19 and 4 in (b), (c), (d) and (e), respectively.

alternatives included using partial convolutions—with and without data masking—and rendering-aware denormalization. We also decided on maximum pooling as it provided higher accuracy than downscaling purely through convolution.

3.1.2 Input Data

Our solution aims to leverage as input any screen-space information that becomes available in real-time rendering pipelines prior to expensive stages that can benefit from accurate metric predictions. Hence, it presents an ideal fit for ubiquitous deferred shading pipelines, which provide a range of screen-space information via the G-buffer. Outputs of previous frames are also commonly obtained as a byproduct of rendering or at little additional cost through temporal reprojection. The question then becomes which of these resources to choose as inputs for the network to yield high accuracy while keeping the input set compact. We assessed commonly available G-buffer contents and statistically analyzed how influential each is on the prediction of perceptual error metrics. Our reference rendering pipeline uses deferred shading, with cascading shadow maps, screen-space ambient occlusion, fast approximate anti-aliasing, and tone mapping with automatic exposure selection. The pipeline was implemented on top of Falcor [BYC+20] and the network trained on established ORCA scene assets (Amazon Bistro [Ama17] and Unreal Engine 4's Suntemple [Epi17]).

We found that directly available information in the G-buffer—such as view-space normals, diffuse color, or roughness—enables reasonable predictions across the entire screen. However, it lacks a myriad of information that otherwise would have to be explicitly encoded, such as lighting, tone mapping, or other effects. As shown in Figure 3.3, we

Table 3.2: **Input Contribution**. Analysis using DeepLIFT of the contribution of each potential network input towards prediction accuracy.

Channels	Format	Seen Regions	Unseen
Reprojected Color	RGB	31.37%	—
Reprojection Mask	Bool	17.26%	8.67%
View Normals	RGB	16.89%	33.63%
Diffuse Color	RGB	14.8%	42.59%
View Normal Z	Float	10.12%	20.15%
Shadowing	Float	7.48%	9.36%
Roughness	Float	5.41%	6.77%
Specular Color	RGB	5.73%	10.61%
Reflect Product	Float	1.06%	1.33%
Emissive Color	RGB	0.01%	0.03%

found the temporal reprojection of final color from previous frames to be a valuable asset (similar to [YZK+19] and [Dro20]), as it contains most of this missing information. However, color reprojection is spatially limited to previously seen regions only and thus presents decreasing benefits in use cases with more obstructions, animated scenes or fast-paced camera movement. Figure 3.3 proves that using temporal reprojection with a quickly changing view or scene does not suffice to produce adequate predictions for the current frame. Hence, a good prediction solution should weight available inputs differently, depending on whether it is predicting for recently seen or newly disoccluded, unseen regions. We assumed (and experimentally confirmed) that the network's prediction quality is highest if reprojected color is paired with a binary mask (seen = 1, unseen = 0).

To quantify the contribution of each input candidate, we used DeepLIFT [SGK17, ACÖG17] on a model trained on all pre-selected candidate inputs and computed attribution scores on a large validation set from our test suite. Table 3.2 lists the mean absolute attribution score of each candidate input, as identified on the FLIP metric. As expected, reprojected color contributes the most, but even more so if masked (accepted if previously seen, zero otherwise). The contribution of diffuse material colors is highest for unseen regions. Other inputs are less important, such as emissive material color, for which we found no anecdotal or statistical benefit, or the dot product between the surface normal and the reflection vector, which is redundant if view-space normals are provided directly. Most RGB channels are relatively redundant, with whichever channel being first in the input order becoming the dominant one and representing the majority of the accuracy of the whole group. The only exception was normals, where the Z-axis is always the dominant one. We also found no advantages of training with HDR for RGB input data instead of 8-bit color channels. Using this knowledge, we can derive effective yet compact input data sets. For real-time applications, we propose to use a single 4-channel texture containing the reprojection mask, one RGB channel (any) of

(a) Helmet scene (b) Ground-Truth (c) MSE Loss (d) MAE Loss (e) Clamped MAE

Figure 3.4: **Transforms Motivation**. Example of the network predicting FLIP on an extremely unbalanced scenario [Car16], containing mostly background and highly reflective surfaces. Training this network to predict this error (b) with traditional losses causes it to underestimate the metric (c,d). Applying our transform on the parameter space remedies the issue (e).

the reprojected color, one RGB channel of diffuse material color, and the Z-axis of the view-space normals. This provides a good tradeoff between desired low inference time and prediction quality since these four account for 52.08% of the network's prediction capability, according to DeepLIFT.

3.1.3 Reparameterization

For a given perceptual metric, its output value distribution can change drastically with different environments and rendering settings. We noticed in our experiments that, for most metrics, the tested scenes produced mostly low output values and only a few very high outliers. Such an unbalanced target distribution might prevent the network from converging to a reasonable solution altogether when trained on arbitrary scenes. In theory, this problem becomes less noticeable the more data and a greater variety of scenes are provided. However, our goal is to provide a solution that can be efficiently trained with a limited training set, as well as arbitrary metrics, scenes, and rendering settings, yet still, generalize across them well.

We note that for many real-time applications, high metric prediction accuracy is most relevant within a limited range of values that drive performance-related optimizations, such as render mode selection. Thus, we choose to tackle the data imbalance issue by using a modified parameter space that balances the data distribution while preserving the relevant information in it.

Let $\mathcal{L}(Y, \hat{Y})$ be a given loss function, where $\hat{Y} \in [0, 1]$ is a set of predicted values in transformed space, and $Y = f(I, I') \in [0, 1]$ the corresponding target values. We define a new loss function that measures the difference between predictions $\hat{Y}$ and targets Y after transforming them to a new parameter space according to a function $\mathcal{T}$:

$$\mathcal{L}_{adaptive}(Y, \hat{Y}) = \mathcal{L}(\mathcal{T}(Y), \hat{Y}) \tag{3.2}$$

We then use mean absolute error (MAE) as our $\mathcal{L}$ loss function:

$$\mathcal{L}_{MAE\ adaptive}(Y, \hat{Y}) = \frac{1}{n} \sum_{i}^{N} |\mathcal{T}(Y_i) - \hat{Y}_i| \tag{3.3}$$

If predictions in non-transformed space are required (e.g., for comparison with perceptual thresholds), they can be obtained as $\mathcal{T}^{-1}(\hat{Y})$. In the following, we describe our two proposed different reparameterization transforms. Figures 3.5, 3.6 demonstrate how they compare to each other and to non-transformed space.

Clamped Transform

A computationally efficient but lossy reparameterization solution is to re-scale the metric, so its output distribution is centered at 0.5, and clamp outlier values to $[0, 1]$ —existing work on HDR imagery shows us this is not unreasonable [LAK18]. Let Y_i be a value to be transformed and μ_Y the mean value of all target values in the data set. We define the clamped transform:

$$\mathcal{T}_{clamped}(Y_i) = \max(\min(\frac{Y_i}{2\mu_Y},\ 1),\ 0) \tag{3.4}$$

where μ_Y can either be precomputed before training or estimated on-the-fly as a running average of previously seen values for Y. We found this transform to improve prediction efficacy on all of our tests, exemplified by Figure 3.4, and suggest it as the default choice. Note that due to its lossy nature, $\mathcal{T}^{-1}$ only exists in the $[0, 1]$ range.

Logistic Transform

Due to its assumptions, the clamped transform may fail to generalize in special cases. This could occur when training on a data set with higher values—and thus, higher μ_Y—or if using a metric with a vastly different distribution. Additionally, it also removes information and zeroes out derivatives for the high outlier values. In cases where this becomes an issue, we propose using instead a transform based on the logistic function $\mathcal{S}$, which is a bounded function with a bell-shaped derivative that is defined everywhere and has its peak at the curve's midpoint. We use the standard logistic function, centered on μ_Y:

$$\mathcal{S}(Y_i) = \frac{1}{1 + e^{-k(Y_i - \mu_Y)}} \tag{3.5}$$

This function allows us to re-center the prediction distribution for any value of μ_Y, while establishing increased importance of accuracy for values near μ_Y, without zeroing any derivatives. Furthermore, it allows adjusting the relative impact of outliers using the logistic growth rate hyperparameter k. In practice, we found that $k = 10$ works best across the evaluated data sets. $\mathcal{S}$, as defined above, does not produce values $\in [0, 1]$. Hence, we normalize it as:

$$\mathcal{T}_{logit}(Y_i) = \frac{\mathcal{S}(Y_i) - \mathcal{S}(0)}{\mathcal{S}(1) - \mathcal{S}(0)} \tag{3.6}$$

3.1.4 Weber Correction

Several use cases of real-time perceptual metrics, such as content-adaptive shading [YZK+19, Dro20] or decoupled shading [MNV+21] use the just-noticeable difference

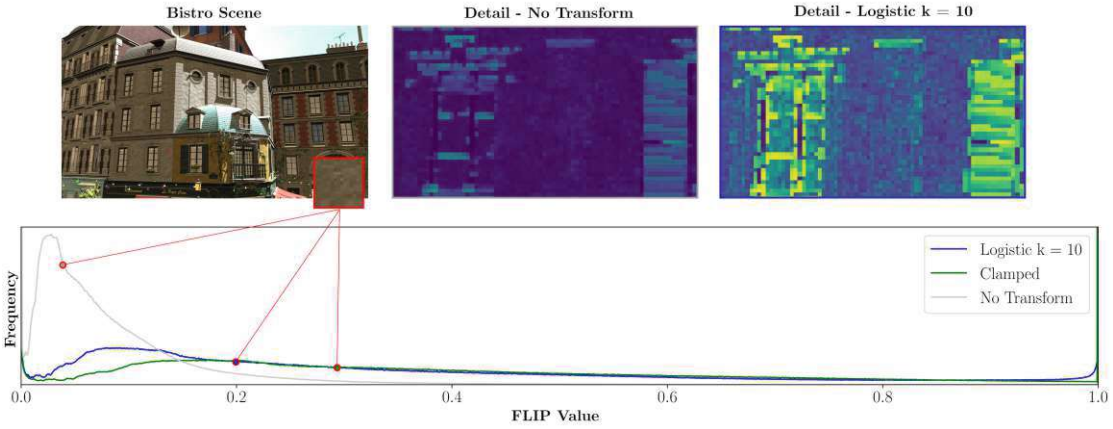


Figure 3.5: **Metric Distribution**. Visualization of region in the training set distribution, according to different FLIP parameter spaces. The clamping transform approaches a uniform distribution, at the cost of ignoring differences between the highest values (notice the spike at $\hat{Y} = 1.0$). The logistic transform achieves a similar result but gives greater importance to nuanced decisions. Both methods preserve all available information.

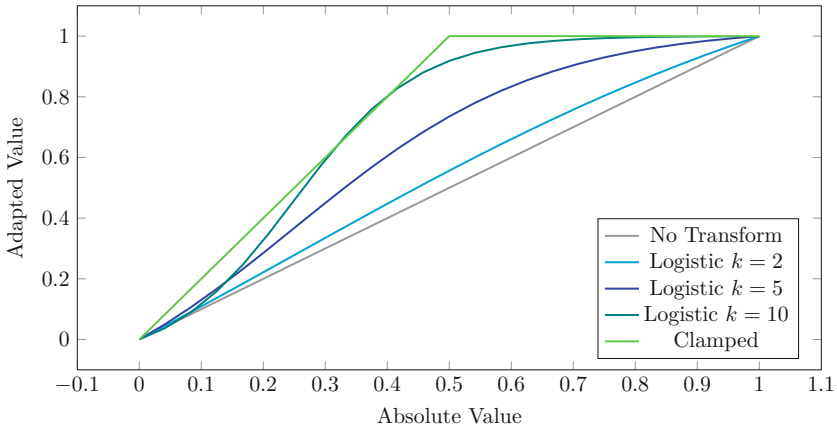
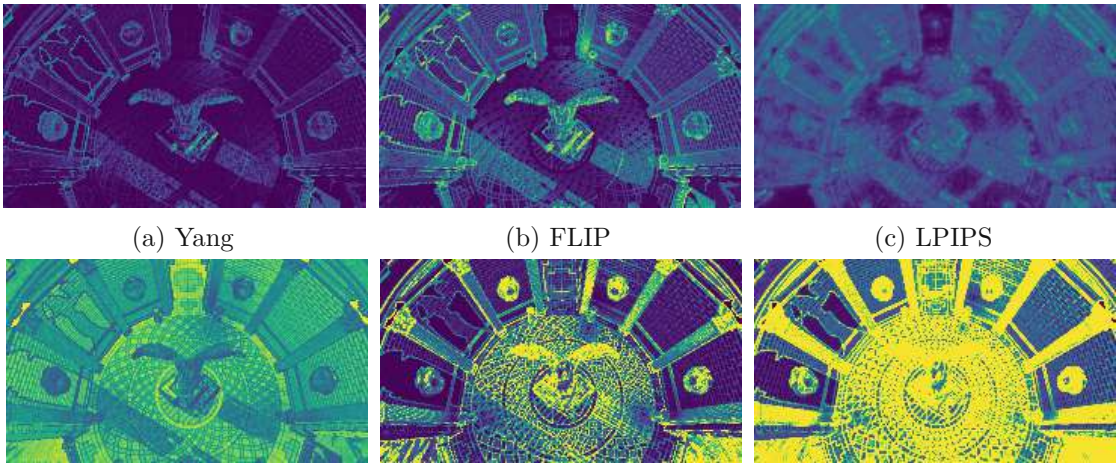


Figure 3.6: **Parameter Spaces**. Visualization of the parameter spaces created as a result of our transforms, for a training distribution average value $\mu_Y = 0.25$.



(a) Yang (b) FLIP (c) LPIPS

(d) PSNR (e) JNYang (f) JNFLIP

Figure 3.7: **Perceptual Metrics**. Ground-truth side by side comparison of the metrics mentioned in this Chapter, including two proposed novel metrics, shown for the same frame when considering half-resolution rendering.

(JND) threshold to inform performance decisions, like render mode selection. However, state-of-the-art approaches rely on explicit computation or reprojection to obtain this—spatially varying—value. Hence, it is only available *after* rendering or for previously seen regions, but not for unseen regions before shading. We solve this issue for our learning-based approach by embedding the visual component of the prediction directly into the model. To enable visually-based decisions, for the current frame, we must estimate the final image error E and compare it with the JND threshold $\mathcal{W}$. Based on Weber's law [TG15], Yang *et al.* [YZK+19] define this threshold $\mathcal{W}_i$ and its applied relation to the visual image error E_i at each location i as:

$$E_i \leq \mathcal{W}_i = t \cdot (L_i + l) \tag{3.7}$$

where L_i is the average luminance at location i, t is a user-selected sensitivity threshold, and l is the environment luminance, which affects the sensitivity to dark areas. This definition is only valid assuming a metric whose output values E directly represent visual error on a luminance scale (e.g., FLIP). The relation is equivalent to:

$$\frac{E_i}{L_i + l} \leq t \tag{3.8}$$

Hence, instead of computing the perceptually-corrected threshold in real time, we can train the network to estimate an already corrected metric Y', enabling the model to specialize for its eventual real-time application and reducing computational cost at runtime:

$$Y' = \frac{E}{L + l} \tag{3.9}$$

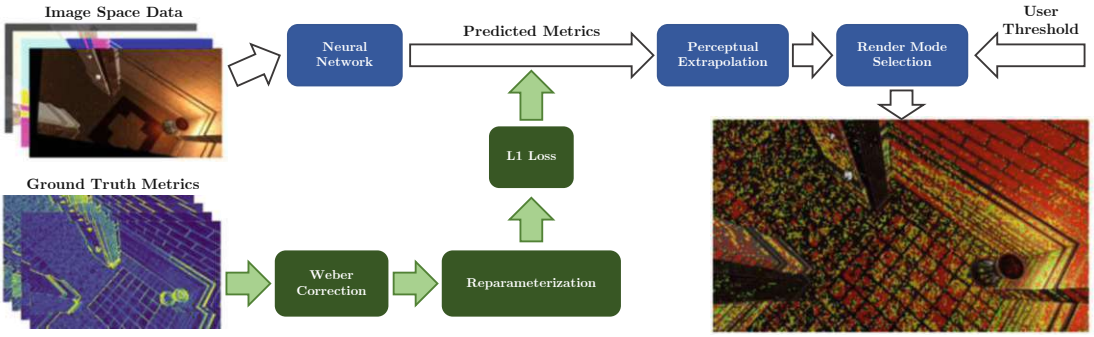


Figure 3.8: **VRS Pipeline**. Proposed rendering mode selection pipeline for content-adaptive shading with VRS. Inputs are provided to a network that has been trained to predict a perceptual metric, with Weber correction and reparameterization applied. At run-time, the network predicts the implied image error for selecting different shading rates. Based on these predictions and a user-defined threshold, the final shading rate (■ full, ■ fine, ■ medium, ■ coarse) is selected for each image region.

Our experiments include Weber-corrected variants derived from existing metrics: just-noticeable FLIP (JNFLIP) and the just-noticeable variant of the image error estimation used in [YZK+19] (JNYang). A comparison of all the metrics used in this Chapter is shown in Figure 3.7 .

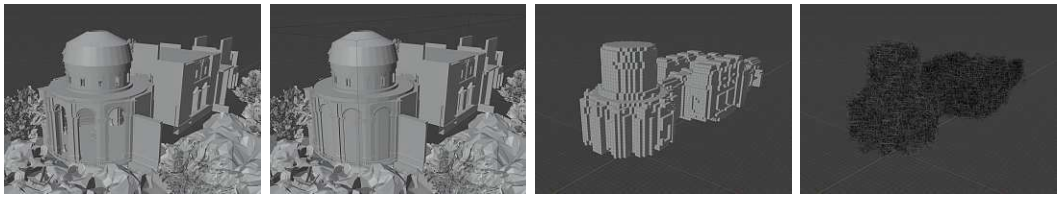
3.2 Real-Time Render Mode Selection

We describe how our metric prediction network can be used for render mode selection. Furthermore, we describe optimizations for applying it to content-adaptive shading with VRS (see Figure 3.8).

3.2.1 Dataset Generation

In order to train a metric prediction network for render mode selection, capturing of training data should be performed with the same render engine that the model is intended to be used with eventually. For the computation of perceptual metrics, reference images for different rendering modes generally must be rendered simultaneously for each captured frame. This is necessary for generating the training and validation targets of any metric that relies on I, I' image pairs. Further, they should be captured only after all post-processing and image effects have been applied, since the computed errors should capture the perceived visual difference.

We capture the environments at representative viewpoints and then compute the metric between each render mode and the reference image obtained without any optimizations active. We also capture the corresponding network input data for each rendered frame, both temporarily reprojected and from the current frame.



(a) Suntemple Scene (b) Added Boundaries (c) Volume Marching (d) Chosen Viewpoints

Figure 3.9: **Viewpoint Capture Pipeline**. The different stages in our viewpoint data capture pipeline, which runs on Blender [Com18]. (b) is the only manual step required, and is only required on maps that are not self-enclosed.

While there is not a single methodology for selecting representative scene viewpoints/states, and its perfectly reasonable to obtain the training data by capturing it during normal application usage, we chose to implement a pipeline to automate the generation of our datasets, as to avoid bias in the training data and ensure consistency in dataset generation during development and analysis. It runs as a combination of a Blender [Com18] script tool and custom render jobs on our Falcor renderer. See Figure 3.9 for an overview of its stages.

For any given scene, we define a cube in the scene centered on a valid viewpoint. Then, we perform flood fill using the cube geometry to create a voxel domain for potentially valid viewpoints. Flooding alone works in closed environments but would leak on any open environment to an infinite domain. To solve this, we take inspiration from game-level design and manually add invisible walls to the scene to limit the valid voxel domain (see Figure 3.9b). Finally, we select random 3D points in the voxel space paired with random 3D directions and test whether they would make valid viewpoints.

Different criteria could be used to validate viewpoints. We filter them based on two:

1. Whether a template camera geometry placed on the viewpoint position and direction intersects with the scene.
2. Whether it renders a minimum amount of visible geometry, measured in percentage of rendered pixels (80%).

The first requirement prevents the dataset from containing viewpoints where the camera intersects with the scene geometry, while the latter avoids an exterior dataset being filled with viewpoints looking at the skybox, for example.

Having a predefined amount of valid viewpoints selected, we randomly select for each a corresponding "previous viewpoint". This simulates the environment being explored by the player and is necessary to generate reprojection training data. To do so, we select random valid viewpoints just as before, but each in very close proximity to its corresponding "next frame". Besides the two aforementioned criteria, we also check whether a raycast from one viewpoint to the next intersects with geometry to verify there

is an open path between the two. Finally, for each previous/next viewpoint pair entry, we render the set of g-buffers, reprojected color, and final renders at different shading rates in Falcor.

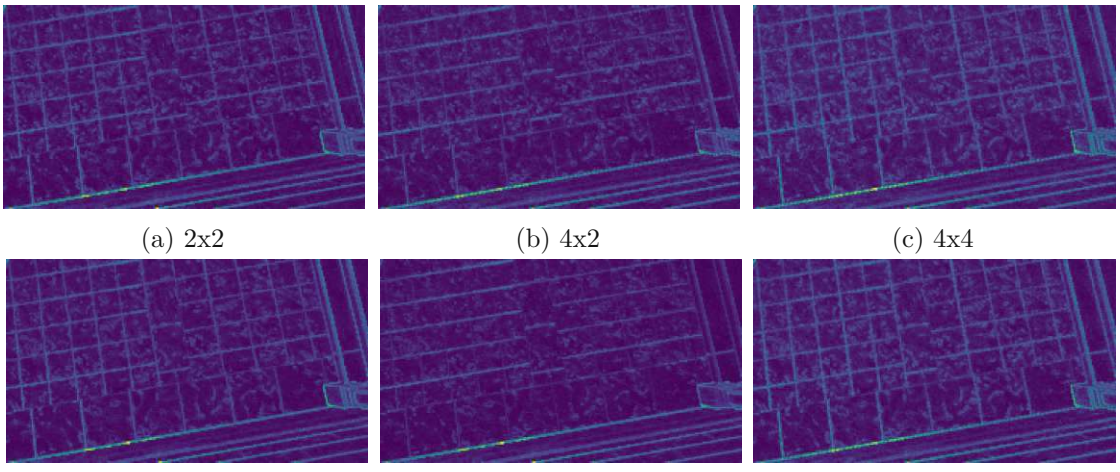
3.2.2 Mode Inference

Rather than predicting render modes directly, we suggest producing a continuous error prediction and perform mode selection based on user-defined thresholds, e.g., the JND threshold, as this allows for greater control by artists and application users alike. Consequently, we can exploit our metric prediction network for this task. We set the layout for our network such that the predicted metric between a render mode and the reference image is computed in a separate output channel for each available mode. We can therefore iterate these channels in order of increasing computational cost and check if any presents a perceptual loss lower than the defined threshold. If no available channel presents an acceptable value for a given tile, we apply the highest quality mode instead:

```
chooseMode(metric, tile)
  for each mode in increasing cost
    if metric[mode, tile] < threshold
        return mode
  return reference mode
```

3.2.3 Rate Extrapolation for VRS

Many modern real-time graphics solutions offer support for VRS, which allows selecting different shading rates for individual objects or image regions to economize on expensive



(a) 2x2 (b) 4x2 (c) 4x4

(d) 2x2 Extrapolated (e) 4x2 Extrapolated (f) 4x4 Extrapolated

Figure 3.10: **Extrapolation for VRS**. FLIP at different shading rates. Ground truth versus extrapolation of them as described in Equation 3.10.

fragment shader invocations. Commonly supported shading rates include fundamental squares (1×1, 2×2 and 4×4) and rectangles with conforming side lengths. For this particular use case, the metric values for similar shading rates are strongly correlated: similar to Yang *et al.* [YZK+19], we can reduce the number of output channels by extrapolating the outputs of multiple channels from just a few. Let $\hat{Y}_{u\times v}$ be an output channel of the network, where u and v are its corresponding horizontal and vertical shading strides, respectively. Let $k = 2.13$ capture the constant relative change in error when switching from a shading rate to its half (e.g., $2 \times 2 \rightarrow 4 \times 4$), as derived by Yang *et al.* [YZK+19]. We can approximate lower shading rates from higher ones, allowing for using only two output channels—the network predictions for 1×2 and 2×1 shading rates:

$$\hat{Y}_{u\times v} \approx \begin{cases} \max(\hat{Y}_{\frac{u}{2}\times v}, \hat{Y}_{u\times\frac{v}{2}}) & \text{if } u = v \\ \max(\hat{Y}_{\frac{u}{2}\times\frac{v}{2}} \cdot k, \hat{Y}_{\frac{u}{2}\times v}) & \text{if } u > v \\ \max(\hat{Y}_{\frac{u}{2}\times\frac{v}{2}} \cdot k, \hat{Y}_{u\times\frac{v}{2}}) & \text{if } u < v \end{cases} \tag{3.10}$$

The values for shading rate 2×2 can be extrapolated from 1×2 and 2×1. Following Equation 3.10, 2×4 can further be obtained from 1×2 and 2×2, 4×2 from 2×1 and 2×2, 4×4 from 2×4 and 4×2, and so on. We found that square rates are approximated with higher precision than non-square rates (see Figure 3.10 for an example). Thus, in practice, we recommend using 4 output channels (1×2, 2×1, 2×4, 4×2) and extrapolating the others for good quality/performance trade-off.

3.3 Evaluation

We assess our approach based on prediction quality, performance, and robustness, (its reliability to handle various scenarios, including edge cases and new scenes). We utilize the 4-channel input set suggested in Section 3.1.2 and simulate camera movement across a total of 8 different scenes by randomly capturing 12,820 viewpoint pairs. The results are computed on a Windows 10 PC equipped with an i7 CPU @ 3.40GHz, 16GB RAM, and an NVIDIA RTX 2080TI GPU.

3.3.1 Metric Prediction

To evaluate the network's prediction capability, we trained and tested it with three established error metrics (PSNR, FLIP, and LPIPS), as well as the Weber-corrected variants (JNFLIP and JNYang). Validation was performed for each scene from Section 3.1.2, using 64 random viewpoint pairs that were withheld during training, as well as on three scenes the network was never trained on: Emerald Square (day/dusk) [NHB17] and Sibenik Cathedral [McG17]. For the approximation I' of I that the network should learn, we chose images rendered for the same frames at full resolution (I) and at four different reduced shading rates (I').

Table 3.3 shows the measured statistics per scene for predicting each metric between reference images and their reduced versions on each scene's test set. Its consistent high

Table 3.3: **Ablation of Metric Prediction**. Prediction quality on test sets across six different scenes. The network has only been trained on the three scenes in the left column (Suntemple and Amazon Bistro). For each scene, we give the number of triangles (Δ), unique materials (⊛), and the achieved R^2 score (coefficient of determination), mean average error (MAE, i.e., the discrepancy between measured and predicted perceptual metric, both total and underestimation only) and variance (σ_{MAE}) of the total MAE.

	Suntemple, 606k Δ, 48 ⊛				Amazon Bistro (Exterior), 2.8M Δ, 132 ⊛			
	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}
FLIP	**90%**	5.99e-2	4.40e-2	7.30e-2	**81%**	8.55e-2	4.40e-2	1.08e-1
PSNR	**92%**	3.15e-2	1.42e-2	3.21e-2	**82%**	4.06e-2	1.98e-2	4.55e-2
LPIPS	**79%**	4.32e-2	2.46e-2	4.54e-2	**77%**	4.15e-2	2.23e-2	4.51e-2
JNYang	**87%**	7.75e-2	5.02e-2	1.11e-1	**84%**	7.93e-2	4.15e-2	1.29e-1
JNFLIP	**88%**	7.37e-2	4.78e-2	9.56e-2	**82%**	9.52e-2	4.10e-2	8.21e-2
	Amazon Bistro (Interior), 1M Δ, 71 ⊛				Emerald Square (Day), 10M Δ, 220 ⊛			
	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}
FLIP	**78%**	7.88e-2	4.46e-2	9.87e-2	**94%**	4.98e-2	2.51e-2	7.56e-2
PSNR	**80%**	3.85e-2	1.25e-2	4.71e-2	**83%**	5.45e-2	3.22e-2	5.72e-2
LPIPS	**72%**	3.39e-2	1.52e-2	3.83e-2	**80%**	4.15e-2	2.29e-2	4.55e-2
JNYang	**78%**	8.59e-2	4.72e-2	1.32e-1	**94%**	5.03e-2	1.83e-2	9.50e-2
JNFLIP	**79%**	9.51e-2	4.12e-2	8.21e-2	**90%**	7.30e-2	2.37e-2	8.15e-2
	Emerald Square (Dusk), 10M Δ, 222 ⊛				Sibenik Cathedral, 75k Δ, 15 ⊛			
	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}	R^2	MAE_{total}	$\text{MAE}_{under.}$	σ_{MAE}
FLIP	**94%**	6.18e-2	1.24e-2	6.99e-2	**91%**	4.07e-2	2.50e-2	6.18e-2
PSNR	**92%**	4.95e-2	6.4e-3	4.30e-2	**88%**	2.95e-2	1.15e-2	4.05e-2
LPIPS	**81%**	3.56e-2	1.38e-2	3.85e-2	**70%**	3.28e-2	2.09e-2	3.68e-2
JNYang	**92%**	5.15e-2	1.55e-2	1.04e-1	**86%**	6.61e-2	2.34e-2	9.61e-2
JNFLIP	**82%**	9.62e-2	0.61e-2	1.53e-1	**81%**	9.18e-2	1.06e-2	9.17e-2

accuracy, high coefficient of determination, and low variance in each scene's test set indicate that the network generalizes rather well: the model is capable of explaining most of the variance in each metric (high R^2) without over-fitting to specific scenes or states (visual examples of predictions are provided in Figure 3.11). We did not find a direct correlation between triangle/material count and the network's ability to predict perceptual metrics. In fact, the highest prediction accuracy was achieved on the most demanding scene in terms of geometry and the number of unique materials, Emerald Square at dusk, despite the network having only been trained on daylight scenes. The lowest scores were obtained in Bistro (Interior), which can be explained by the large number of specular objects it contains: since light sources are not explicitly encoded in the input, the network struggles to produce accurate predictions in previously unseen regions with specular materials.

To test this theory, we created two variations of this scene, shown in Figure 3.12: one with highly specular chrome materials and one with flat checkerboard textures applied everywhere (see supplemental material). As expected, the prediction quality, as indicated by R^2, is lower for the completely specular scene (FLIP: 71%, PSNR: 76%, LPIPS: 64%, JNYang: 70%, JNFLIP: 70%). However, for the same scene with only flat, checkered textures, the opposite is true: prediction quality rises, conversely, bringing it closer to

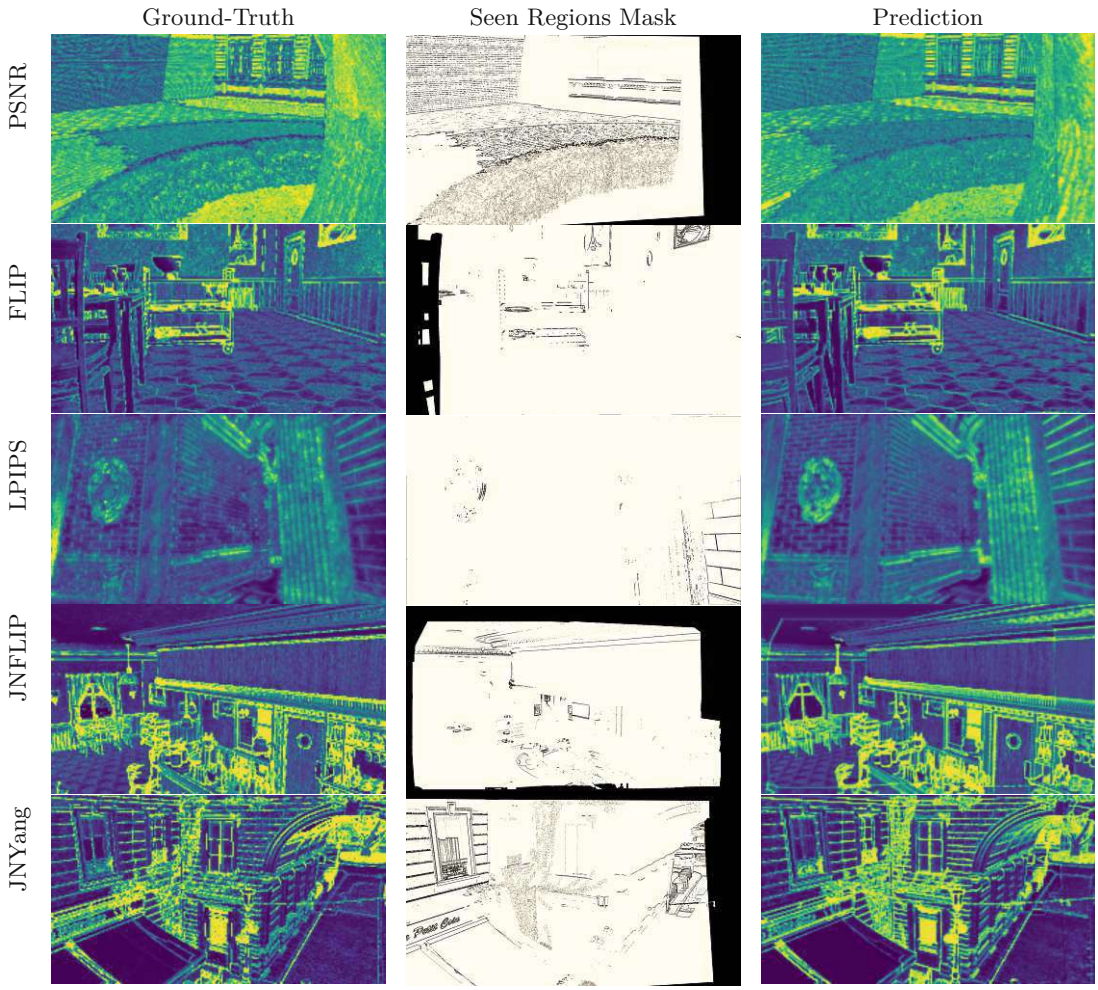
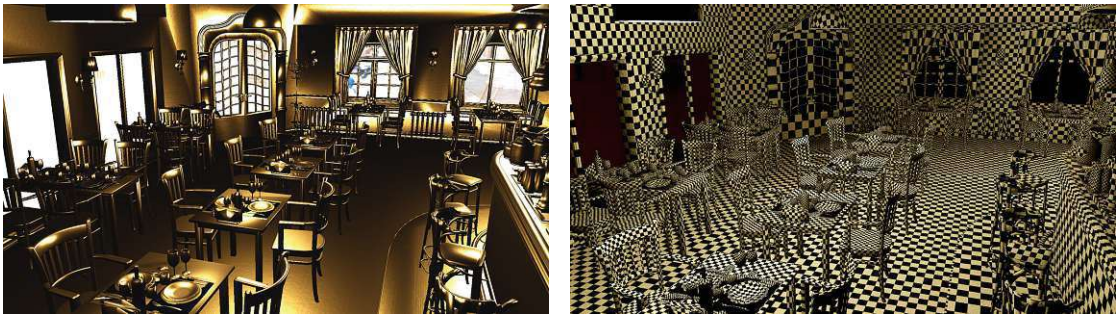


Figure 3.11: **Prediction Results**. Examples of our network predicting metrics in tested scenes. Black in the center column indicates unseen regions in the current frame. All metrics performed similarly across tested scenes, with no obvious outliers or catastrophic failures.

the other scenes.

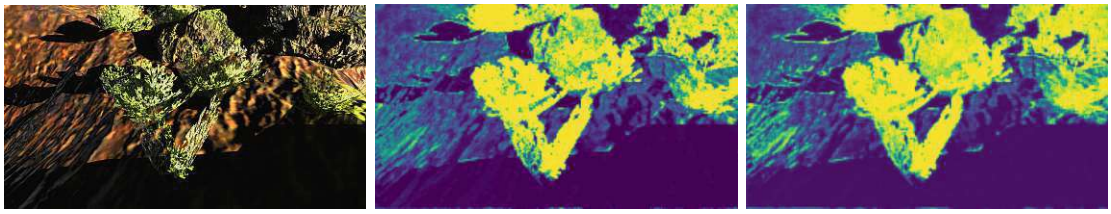
As demonstrated in Figure 3.13, the network does not require a large number of training samples to achieve generalization: in our experiments, we found a negligible decrease in test accuracy—0.04%—when an environment is not included as part of the training and found no benefit in using more than $500 - 2000$ captured frames on any environment (the exact number depends on the scene size).

Finally, we recorded the run time for prediction and compared it against reference implementations of the corresponding metrics on the CPU and on the GPU (Python+PyTorch). For our neural network, timings are independent of the metric it was trained on since



(a) Bistro, interior, highly specular (b) Bistro, interior, high-frequency textures

Figure 3.12: **Modified Scenes**. We modified a scene to be used for evaluating the influence of (a) highly specular materials and (b) simple checkerboard textures on the prediction quality and performance of our approach.



(a) Suntemple, exterior (b) Ground-Truth FLIP (c) Predicted FLIP

Figure 3.13: **Predictor Generalizability**. Our prediction network, when trained strictly indoors in the Suntemple scene, still produces accurate FLIP predictions for bushes and rocks on the exterior.

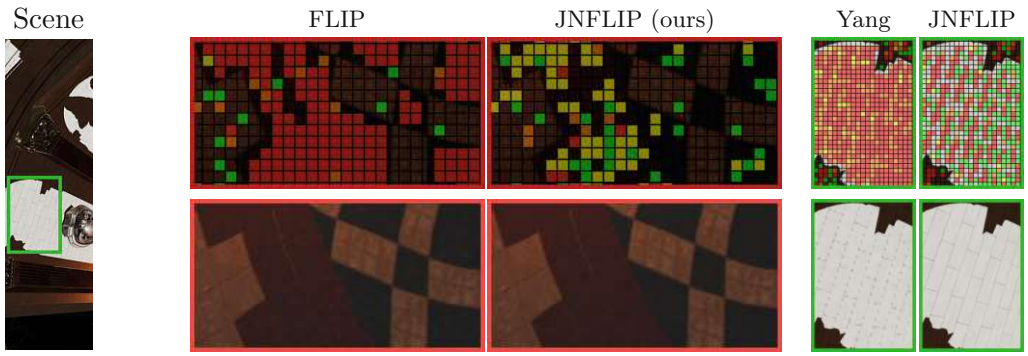


Figure 3.14: **Adaptive Shading Comparison**. While FLIP is not normalized for local brightness and thus underestimates dark regions, and the method by Yang *et al.* [YZK+19] can struggle in shiny or overly exposed regions, JNFLIP handles both cases gracefully which can provide visual benefits in content-adaptive shading.

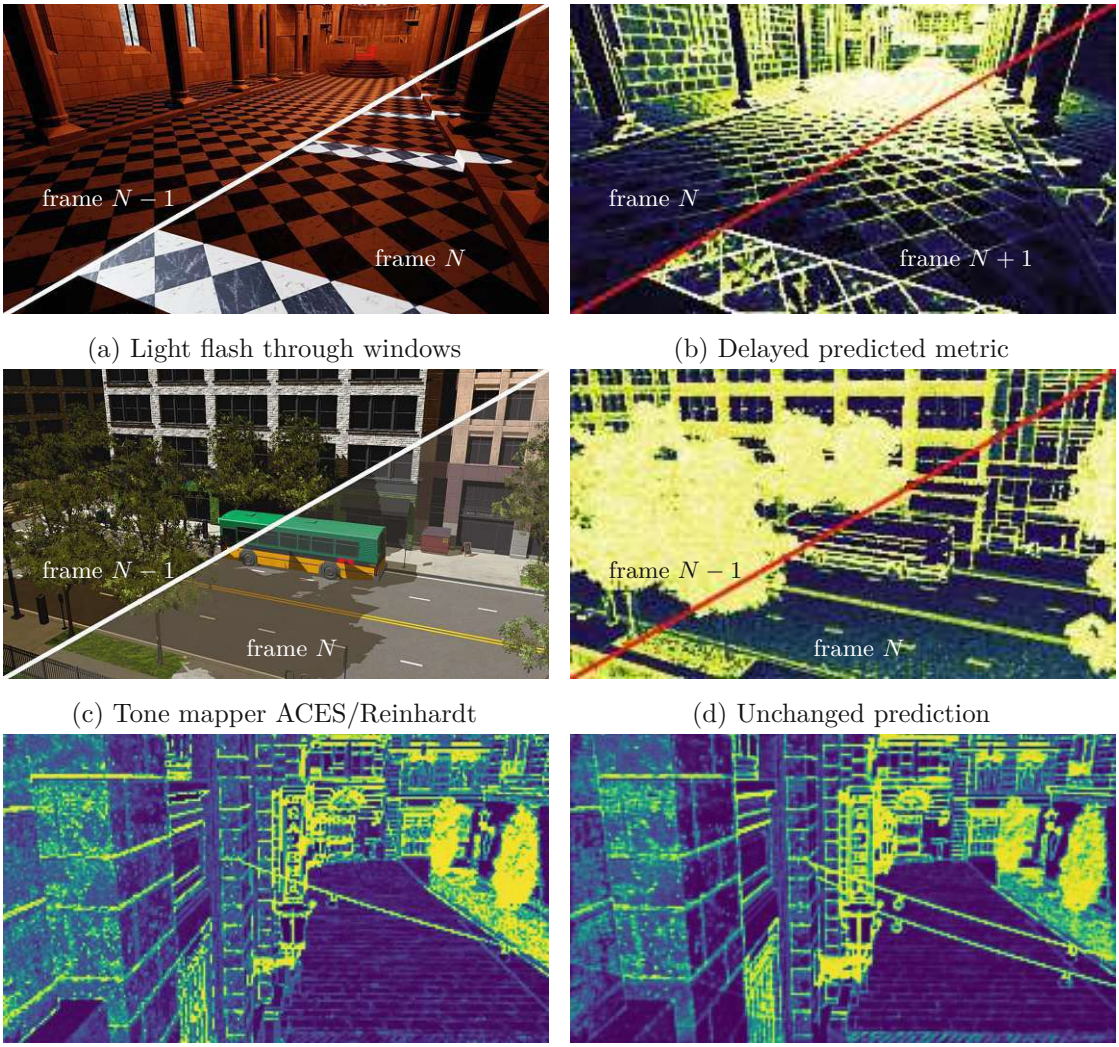
it does not influence its architecture. Inference with our network took $0.58s/2ms$ on CPU/GPU, respectively. It is thus significantly faster than explicitly computing FLIP ($2.46s/190ms \rightarrow 4.24\times/95.9\times$) and LPIPS ($13.6s/16.4ms \rightarrow 23.5\times/8.2\times$). For the much simpler PSNR, our approach is between $2\times$ and $10\times$ slower.

3.3.2 Content-Adaptive Shading Application

To assess a real-time use case, we implemented content-adaptive deferred shading in Falcor [BYC+20] using our network, trained on JNYang and running on 16×16 tiles at $1080p$ resolution. We load the network into TensorRT and provide it with GBuffer-texture inputs in Falcor directly. For comprehensive results, we ran performance evaluation on five scenes that exhibit varying complexity in terms of geometry and materials: Suntemple, Bistro (Exterior), and the regular/specular/checkered Bistro (Interior). Frame times of our approach was compared with full-rate shading and a state-of-the-art VRS method [YZK+19]. We considered two types of camera motion between frames: slow (resulting in 14% previously unseen pixels per frame on average), and fast (31% unseen on average), and evaluated 15 corresponding viewpoint pairs per scene and speed.

Inference with TensorRT requires a constant ≈ 2.3 ms per frame. For our approach to provide a benefit, it must amortize this overhead, which can only occur under appreciable fragment shader load. To simulate a pipeline comparable to interactive graphics applications (e.g., AAA video game titles), we created a synthetic load (50:1 arithmetic to memory) in the deferred fragment shader to bound full-rate shading performance to 60 FPS. In combination with our network's prediction, GPU hardware support for VRS yields a considerable performance gain across the board. For a slow-/fast-moving camera between frames, we achieved a $1.12/1.14\times$ speedup for Suntemple, $1.17/1.18\times$ for Bistro (Exterior), and $1.42/1.41\times$ for the regular Bistro (Interior). The purely specular and checkered versions of the latter performed slightly better ($1.5/1.54\times$ and $1.48/1.52\times$, respectively): in both cases, this can be explained by the reduction of sharp features and high-frequency visual details in the scene, which enables the network to choose lower shading rates. In summary, VRS using our network reduced average frame times by at least 10% compared to full-rate shading in all examined scenarios. The relative performance gain is boosted by the reduction of high-frequency features, permitting the use of lower shading rates.

For comparison with Yang *et al.* [YZK+19], we used the same setup and configured the synthetic load so to have their approach match the target frame rate. Since their base overhead is significantly lower than our network's inference time, our method trails behind Yang *et al.* 's at 60 FPS with slow camera motion on static scenes (52.1 FPS on average across all scenes $\rightarrow 0.87\times$). For fast camera motion, however, our method performs better ($1.03\times$) due to its ability to predict and use lower shading rates in unseen regions, rather than defaulting to full resolution. Using an even heavier load (30 FPS target), our method prevails as soon as camera motion occurs ($1.11\times$ at slow, and $2.16\times$ at fast motion). Hence, even given the early state of dedicated GPU inferencing hardware, our learning-based approach can provide clear benefits in such demanding scenarios.



(a) Light flash through windows

(b) Delayed predicted metric

(c) Tone mapper ACES/Reinhardt

(d) Unchanged prediction

(e) Ground-Truth

(f) Predicted

Figure 3.15: **Predictor Limitations**. (a,b) Reliance on reprojection can cause the network to react to sudden lighting changes in previously seen regions with a delay of one frame. (c,d) Changing tone-mapping method also does not result in immediate different predictions. (e,f) Incorrect previous frame reprojections can cause our network to hallucinate duplicated objects due to surface information mismatch.

Finally, we compared using Yang *et al.* [YZK+19], FLIP and JNFLIP as metrics for use within content-adaptive shading. We found some benefits to the use of JNFLIP, as FLIP is not normalized for local brightness and Yang *et al.* [YZK+19] can struggle with shiny or overly exposed regions. An example comparison is shown in Figure 3.14.

3.3.3 Limitations

The key purpose of our approaches is to enable optimizations in real-time applications by predicting the—otherwise expensive—pixel shading result. This naturally impedes its ability to account for factors that are unknown prior to pixel shading. We circumvent this issue by relying on reprojection and G-buffer data, the latter of which may not contain all information affecting the final color generation (e.g., light source position, cf. Figure 3.3). Hence, similar to other state-of-the-art methods [YZK+19], the network is bound to make assumptions about such effects based on previously seen regions. If an effect cannot be predicted from G-buffer data alone, it may only react to it in the next frame, when its reprojection becomes available. This includes temporal inconsistencies in the scene (e.g., sudden disocclusion of a strong light source), reflections, and modifying of rendering settings or post-processing effects (Figure 3.15). However, in this Chapter, we have shown that our approach can be trained to discard reprojected color and substitute information derived from G-buffer data instead. Hence, it may be trained to adapt to sudden changes immediately. For instance, in the case of a disoccluded light source, this could be achieved by providing additional input tracking changes in the binary screen-space shadowing information between frames. For more complex effects, like reflections or fog, more sophisticated solutions may be needed to provide suitable, inexpensive approximations of the required information to the network. The decision of trading a single-frame delay of predicted effects for larger input sets should then depend on the user's expected attention to them. Future work may explore under which circumstance reprojection may be omitted and instead replaced by additional, equally expressive encodings or estimates of important scene features, such as light sources and reflections. Tackling this challenge would come with the advantage of providing a unified solution for both seen and unseen regions.

Although the achieved performance in real-time applications is acceptable with our approach, it incurs an overhead that limits its applicability. For slow-moving changes, selective reuse of predictions could significantly alleviate this issue, which we aim to pursue in future work. In our proof-of-concept, the naive screen-space reprojection used is not precise, which can sometimes cause our network to hallucinate thin objects' duplicates due to material and reprojection data inconsistency (Figure 3.15). This could be improved upon by using state-of-the-art, non-screen-space reprojection.

CHAPTER 4

Refining Cost-Volumes for Depth Prediction

In this chapter, we turn our attention to the issue of depth reconstruction in light-field cameras, the second subject initially broached in Chapter 1. The research presented here has been published at the *25th International Conference on Pattern Recognition* under the title '*Cost Volume Refinement for Depth Prediction*' [CGW21]. As we previously outlined, light-field cameras offer a unique set of capabilities, including the ability to refocus images after they have been captured. However, they also present specific challenges that are not encountered in traditional single-camera or multi-camera systems, particularly when it comes to depth estimation. The narrow baselines of the micro-lenses in light-field cameras make conventional depth-reconstruction techniques unsuitable, necessitating the development of new methods tailored to this technology. This need aligns with the discussions in Section 2.2.2, where we explored cost volumes and their application in light fields.

It is worth noting that, to the best of our knowledge, the majority of existing research in this area focuses on generation of cost volumes and depth estimation from this cost, which are arguably the most important steps. Often, these methods resort to existing range-image refinement techniques to compensate for their limitations. We argue that this is a sub-optimal approach, as it disregards most of the rich cost-volume information (and thus, the inherent data redundancy provided by light-field cameras). Instead, our work proposes a shift in focus towards the cost-refinement stage, aiming to leverage this untapped potential for more accurate depth estimation. We employ iterative optimization to refine cost volumes, as it provides a computationally efficient solution compared to alternatives like deep learning methods. Additionally, it removes the necessity for extensive datasets of light-field images and their corresponding ground-truth depths. Figure 4.1 provides a preview of how our refinements improve upon existing methods.

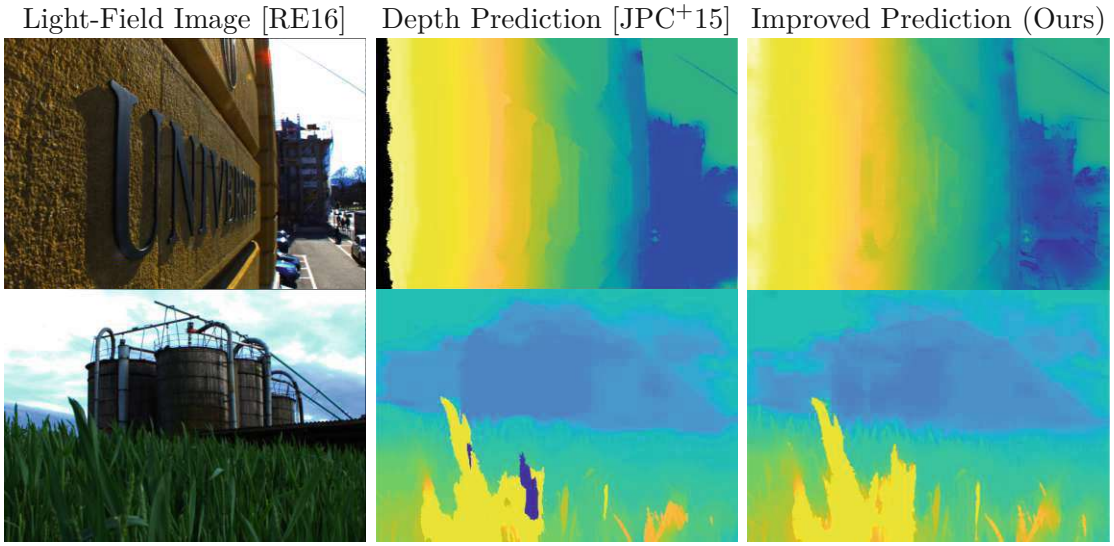


Figure 4.1: **Chapter Overview**. The refinements described in this chapter improve upon existing methods for depth prediction using light-field cost volumes. Notice that our proposed cost volume refinements are less prone to artifacts and better preserve details of far away objects (the street and far away building on the top, the grass and metallic cylinders on the bottom).

In summary, our innovation in this chapter lies squarely within the **novel iterative-optimization algorithms** proposed and **their modular application** within the cost-volume stage of depth prediction, resulting in the following contributions:

1. A modular framework for cost-volume refinement, which can be applied for depth reconstruction on light-fields, regular and multi-view stereo imagery.

2. A floating-point method for artifact-removal on cost volumes based on classification methods robust to smooth surfaces and object complexity.

3. A fast local smoothing method for noise and discontinuity reduction on cost volumes robust to sharp depth changes.

4. A method for combining cost-volume based depth prediction with other prediction methods before regression.

Section 4.1 describes the standard stages for depth reconstruction from light-fields using cost volumes. Section 4.2 describes our proposed refinement framework and methods. Furthermore, in Section 4.3 we present an extensive testing of the importance of cost-volume refinement and of the efficacy of our methods on multiple previously proposed cost cues.

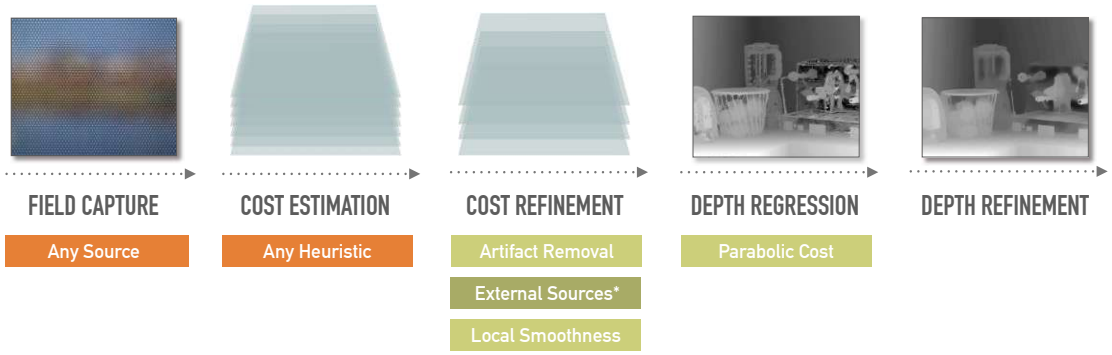


Figure 4.2: **Minimal Depth Estimation Pipeline**. Overview of our minimal pipeline for depth estimation using cost volumes. Images are only illustrative. See Figure 2.3 for a comparison with Tao *et al.* [THMR13], Jeon *et al.* [JPC+15] and Williem *et al.* [WWL18] pipelines. Note that, due to its specificity, the results in this thesis and the accuracy comparisons in Section 4.6 do not use combination of external predictions or domain-specific knowledge as outlined in Section 4.2.1, unless specifically noted.

4.1 Minimal Pipeline

To test our proposed cost-volume refinements, we present a simplified pipeline, which abstains from complex depth regression and depth refinement techniques. This is done for three reasons: first, one of the advantages of cost refinement is the reduced necessity of such techniques, which we want to show. Second, some existing regression methods, such as graph cuts label propagation [JPC+15] [WWL18], cause a great loss of detail and a portion of our improvements could be lost. Third, we want to refrain from giving the results of our refinements any unfair advantage by working with the most traditional and naive pipeline as possible. This approach also aligns itself with the well-tested optimization principle of parsimony, focusing on achieving the best results with the least complexity.

All of our results are computed using this minimal pipeline, illustrated in Figure 4.2. For comparison, results of previous work are always computed using their respective original pipelines. As explained in Section 2.2.2 and shown in Figure 2.3, all of these pipelines used by cost-volume based depth prediction methods can be generalized into 5 stages, and ours is no exception. First, for cost-volume generation, we use cues from existing work. Then, for each volume, our refinement methods are applied in succession: obvious artifacts are removed using our classification-based global artifact-removal, described in Section 4.2.2. Noise and unwanted sharp discontinuities are vastly reduced using our iterative local smoothness refinement, described in Section 4.2.3. Optionally, depth predictions from non cost-volume based methods can also be combined in this stage, a form of ensemble learning we describe in Section 4.2.1.

After our proposed refinements have been applied, we make use of a classical solution to

estimate depth from the cost. The theoretical depth regression, described in Equation 2.17, assumes that cost is a continuous function. However, cost volumes are computed and stored in discrete steps. Reducing this step ξ increases depth precision but at the cost of computational performance, and precision is required to effectively use cost refinement. Thus, we use parabolic interpolation [AS17] [KOK14], which takes into account information from the immediate neighbors of the minimum step $D_C(\mathbf{u})$:

$$D(\mathbf{u}) \simeq \mathcal{D}_C(\mathbf{u}) = D_C(\mathbf{u}) - \left(1 + 2 \cdot \frac{C(\mathbf{u},\, D_C(\mathbf{u}) - \xi) - C(\mathbf{u},\, D_C(\mathbf{u}))}{C(\mathbf{u},\, D_C(\mathbf{u}) + \xi) - C(\mathbf{u},\, D_C(\mathbf{u}) - \xi)}\right)^{-1} \tag{4.1}$$

This interpolation method minimizes the error in depth estimation by utilizing local cost information to interpolate between the volume's discrete steps, effectively optimizing the depth value at each point without increasing computational costs.

4.2 Refinement Algorithms

Let $k \in [0, n_r[$ be the number of cost-volume refinements that have been performed by a pipeline, where n_r is the total number of refinement algorithms being used. We generalize modular refinement as a function R_k that takes as input the current cost volume C_k and outputs a refined volume C_{k+1}:

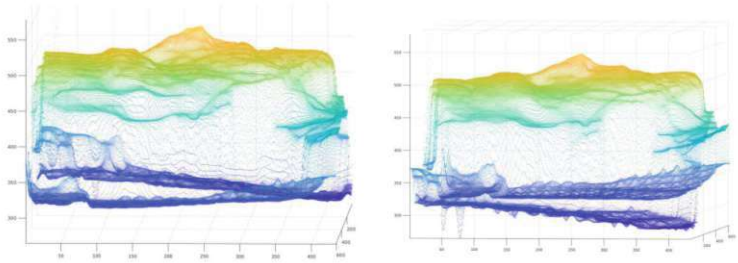
$$C_{k+1}(\mathbf{u}, z) = R_k(C_k,\, \mathbf{u}, z,\, \lambda_k) \tag{4.2}$$

In this thesis, we present three different modular refinement algorithms, all of which follow the definition of Equation 4.2 and can be used interchangeably in any order. The strength of refinement R_k can be controlled with hyper-parameter $\lambda_k \in \mathbb{R}$. That is, each refinement R_k serves as a regularizer in the optimization problem, and λ_k allows to fine-tune the balance between the different objectives (the original cost function and the applied refinements) during minimization.

4.2.1 Independent Predictor Combination

We first describe our simplest refinement. Our goal is to define a generic method that can make use of any independent depth prediction to inform our own. Independent predictions can be obtained from non cost-based light-field methods, from existing methods for monocular or stereo imagery, or from any domain specific knowledge.

To do so, we increase the original cost according to the difference between depth and the independent method's prediction. There should be no increase when in agreement, but cost should increase as the two diverge. Additionally, the increased cost should have



(a) Not Refined (b) Refined

Figure 4.3: **Neural Network Combination**. Depth reconstruction from a light-field portrait picture with and without refinement from Section 4.2.1. Here, we use our refinement to combine information from the facial neural network proposed by Sela *et al.* 2017 [SRK17].

a known maximum $\in \mathbb{R}$, so that the refinement impact can be controlled. Given these properties, we define the increased cost $G(t) \in [0, 1]$ as normalized inverted Gaussian distributions centered around the independent predictions:

$$G(t) = 1 - e^{-\frac{t^2}{2\sigma^2}} \tag{4.3}$$

where σ is a chosen deviation. We choose this Gaussian-based cost adjustment to serve as a soft constraint, optimizing the agreement between independent depth predictions and the cost volume. For each image point u for which the independent method has a prediction P_u, the refinement operation becomes:

$$C_{k+1}(\mathbf{u}, z) = C_k(\mathbf{u}, z) + \lambda_k G(P_\mathbf{u} - z) \tag{4.4}$$

As an example, we explore the case of facial reconstruction. We make use of the trained neural network presented by Sela *et al.* 2017 [SRK17] for facial reconstruction from color images as our domain-specific prior knowledge. We estimate prior depth P from the neural network depth prediction and, for each pixel that we have a prior $P_\mathbf{u}$ for, we apply the refinement. The result can be seen in Figure 4.3.

4.2.2 Classification Artifact Removal

We propose a variation of the previous refinement for the purposes of artifact removal. In particular, as described in Chapter 2, we found that some multi-label classification methods (where discrete steps in depth correspond to labels) are robust to artifacts, but tend to reduce accuracy due to lack of precision or miss-assignment between close labels. To take advantage of the artifact detection while maintaining accuracy, we define a increased cost different from Section 4.2.1.

As before, we increase cost according to the difference between depth and the multi-label classification. However, it should not be bound to a known maximum and should instead quickly rise with divergence. As such, we define the increased cost as a polynomial of degree m, where m is an even number. Additionally, artifacts are more likely the higher the difference between predicted parabolic depth $\mathcal{D}$ and multi-label classification L is. Thus, we scale the increased cost according to the difference between these two predictions computed from the current cost volume C_k.

Our refinement thus becomes:

$$C_{k+1}(\mathbf{u}, z) = C_k(\mathbf{u}, z) + \lambda_k |L_\mathbf{u} - \mathcal{D}(C, \mathbf{u})| \cdot (L_\mathbf{u} - z)^m \tag{4.5}$$

As an example, we use the graph-cuts implementation of Jeon *et al.* [JPC+15] for propagation of SIFT feature matches to estimate each label L_u at pixel u with $m = 2$. A direct comparison of our refinement to the original algorithm can be seen in Figure 4.4.

4.2.3 Iterative Local Smoothness

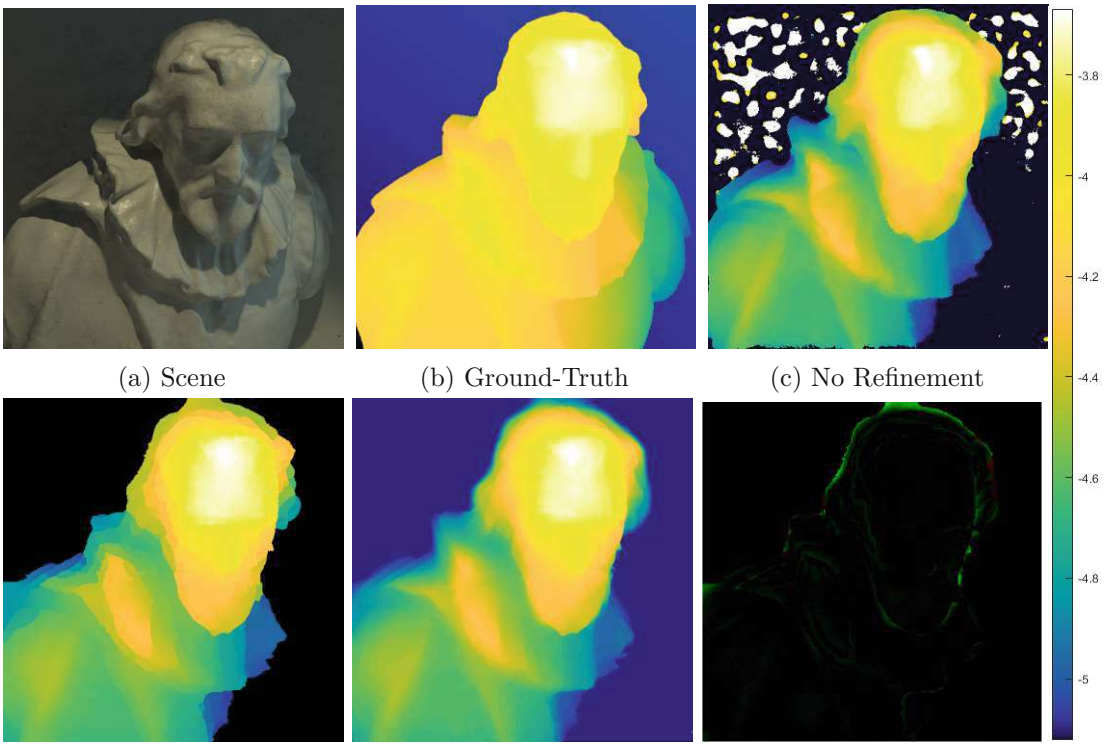
Two common issues with depth predictions from cost volumes are noise and local artifacts. We vastly reduce these by looking at the neighborhood $\mathcal{I}_u$ of each image point u. For each neighbor $v \in \mathcal{I}_u$, we create an added cost based on the difference between the depth predictions at v and u. To weight the importance of each neighbor, we estimate point confidence using the peak ratio coefficient W (as proposed by Hirschmüller *et al.* [HIG02]), which produces lower values when multiple local minima are similar:

$$W_C(\mathbf{u}) = \frac{C(\mathbf{u}, \mathcal{D}_C(\mathbf{u}))}{C(\mathbf{u}, \operatorname{argmin}_{z \neq \mathcal{D}_C(\mathbf{u})} C(\mathbf{u}, z))} \tag{4.6}$$

Just as for the refinement in Sections 4.2.1, we want agreeing predictions to have no additional cost, but to increase cost as local differences raise up to a chosen maximum. As such, we use the same normalized inverted Gaussian distribution G. However, we want to define the increased cost as a function of depth prediction at the neighbors, which is in turn dependent on the cost increase. To deal with the conundrum, we take inspiration from traditional optimization techniques like gradient descent, which systematically update parameters to minimize a given cost function. In our case, we compute our refinement iteratively to achieve more accurate depth predictions. Let $j \in [0, n_i[$ be the current iteration number. We define new temporary volumes S as:

$$\begin{cases} S_0 = C_k \\ S_{j+1}(\mathbf{u}, z) = C_k(\mathbf{u}, z) + \\ \qquad\qquad \lambda_k \sum_{v \in \mathcal{I}_\mathbf{u}} G(\mathcal{D}_{S^j}(v) - z) \cdot W_{S^j}(v) \end{cases} \tag{4.7}$$

where $\mathcal{D}_{S^j}$ is the parabolic depth prediction and W_{S^j} the Peak Ratio coefficient computed from temporary cost volume S_j, according to Equations 4.1 and 4.6 respectively. n_i is the



(a) Scene (b) Ground-Truth (c) No Refinement

(d) Graph Cuts (e) Artifact Removal (ours) (f) Difference of (d) and (e)

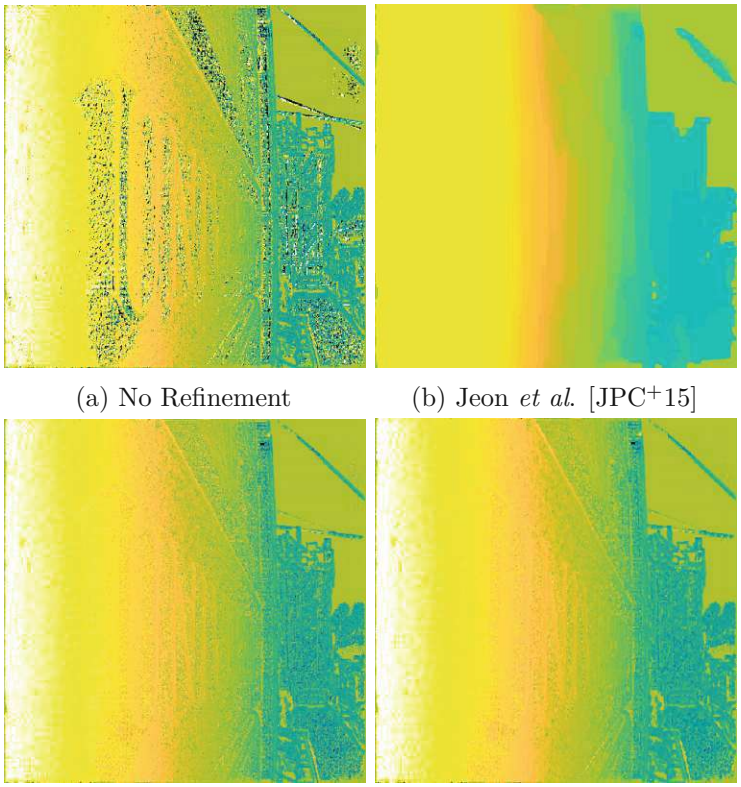
Figure 4.4: **Artifact-Removal**. Reconstructions of a bust on an intentionally poorly estimated cost volume, which overestimates depth and is unable to predict depth of far away objects, resulting in multiple visible artifacts. Graph cuts prediction (d) removes the background artifacts, but at the cost of depth accuracy of the bust itself. The artifact-removal refinement (e) from Section 4.2.2 outperforms previous methods, being able to both remove artifacts and reduce depth overestimation without decreasing accuracy.

total number of iterations to be performed, which can be either statically or dynamically controlled by the pipeline. We make use of the Peak Ratio because different neighbors might have more or less reliable cost predictions than others, and thus should be weighted differently.

Having the last iteration been performed, we define our refinement as:

$$C_{k+1}(\mathbf{u}, z) = S_{n_i}(\mathbf{u}, z) \tag{4.8}$$

In our implementation, we set a static maximum number of iterations, but we also monitor the ratio of change between depth maps $\mathcal{D}_{S^j}$ and $\mathcal{D}_{S^{j+1}}$. Once the ratio is below a predefined threshold, we stop iterating. This stopping criterion is a form of early termination, an optimization technique to save computational resources when further iterations yield diminishing returns. We found that our implementation never requires more than 2 iterations before converging.



(a) No Refinement (b) Jeon *et al.* [JPC+15]

(c) Artifact-Removal (ours) (d) Iterative Smoothing (ours)

Figure 4.5: **Iterative Smoothing**. Depth regression before any image-based refinement is applied on a real-life light-field image from a dataset [RE16]. Previous work often forfeits detail and accuracy (b) to reduce artifacts and noise. Our use of cost volume refinement methods (c,d) from Sections 4.2.2 and 4.2.3 solve these issues while preserving detail. Depth from volumes was predicted using parabolic interpolation (a,c,d).

4.3 Evaluation

To test the effectiveness of our refinement algorithms, we look at three very different cost-volume generation cues proposed by three different authors: Tao *et al.* [THMR13] lenslet variance (LV), Jeon *et al.* [JPC+15, JPC+19] sum of absolute differences computed using sub-pixel phase-shift (SAD) and Williem *et al.* [WWL18] constrained angular entropy (CAE). We compare the depth maps regressed from these volumes with our minimal pipeline (as described in Section 4.1), which includes cost volume refinement, to the ones regressed using the pipelines publicly provided by their authors (see Figure 2.3).

To do the comparisons, we use the synthetic dataset by Honauer [HJKG16], which contains 30 pairs of light-field images and corresponding ground-truth depth maps of different scenes. We also present a visual comparison using the dataset by Rebarek and Ebrahimi [RE16], as shown in Figures 4.1 and 4.5. For each scene, we generate cost

volumes according to the three mentioned cues. Then, for each volume, we regress depth maps using our minimal pipeline and the originally corresponding one. This results in a total of 6 different depth maps per scene.

Note that both Tao *et al.* [THMR13] and Jeon *et al.* [JPC+15, JPC+19] combine multiple cues using weighted sums of different cost volumes in their works. However, we are not proposing an end-to-end depth prediction method, but a set of operations that, given an arbitrary cost volume, are able to generate a better and more consistent volume. Thus, we analyze the performance of refinement on different cues individually.

Additionally, the absolute error between predictions and ground-truth of a specific example are not relevant, as they are largely constrained by the quality of the input cost volume. Instead, we look at how this error changes with the introduction of refinements. Thus, Figure 4.6 displays the change of error when our refinements (with our minimal pipeline) are used, color coded in green for reduced and red for increased error. Color is normalized to the highest change.

We do not tweak the configurable variables λ_k and σ from Equations 4.2 and 4.3 for each scene and use the same for all tests. We also do not perform any additional operations, such as vignetting and distortion estimation and correction, as these should affect all 6 cases equally, and operations before cost-volume generation are out of scope of this thesis.

Statistical Analysis We calculate the mean squared error and the structural similarity index between the ground-truth maps and the regressed ones. Table 4.1 shows the average of these metrics for each of the 6 combinations. Our minimal pipeline outperforms the original ones in all cases, presenting a lower average error and higher similarity, even though it is not performing complex depth regression or depth refinement. We also found that our refinements are the more effective the worse the cost prediction is. For example, the differences are more visible in real photographs than synthetic (perfect) images, or in intentionally poor reconstructions. As such, the real error and similarity differences might be higher than suggested by our synthetic dataset.

Detail Preservation As displayed in Figure 4.5, alternative proposed methods often oversimplify depth predictions. Most detail can be lost and only regained through depth-map refinement, which does not take advantage of the light-field properties over traditional images. Our algorithm is able to vastly reduce unwanted artifacts and noise, while preserving the details present in the scene by performing operations at a cost volume level. As such, the shape of objects in the final results more closely resembles the ground-truth than previous methods, as shown in Figure 4.6.

Limitations While functional, our smoothing refinement is still not able to remove all noise without removing details. As shown in Figure 4.5, the refined result still exhibits some noise. We also do not have a solution for optical effects such as flares, which can mislead predictions locally, as also shown in Figure 4.5. However, we are not aware of any

Pipeline	MSE	SSI
LV (Tao *et al.* [THMR13])	2.1672%	9.6871
Refined Lenslet Variance	1.5297%	10.8989
SAD (Jeon *et al.* [JPC+15])	1.2829%	11.3914
Refined Sum of Absolute Differences	0.7165%	11.6619
CAE (Williem *et al.* [WWL18])	3.2723%	8.6063
Refined Constrained Angular Entropy	2.1083%	9.9078

Table 4.1: **Ablation of Refinements**. Comparison of the average of mean squared error (MSE) and structural similarity index (SSI) for each tested pipeline when predicting on the synthetic dataset by Honauer [HJKG16]. Lower MSE and higher SSI are better [HJKG16].

existing depth refinement method to deal with this issue, and previous work frequently suffers from the same issue.

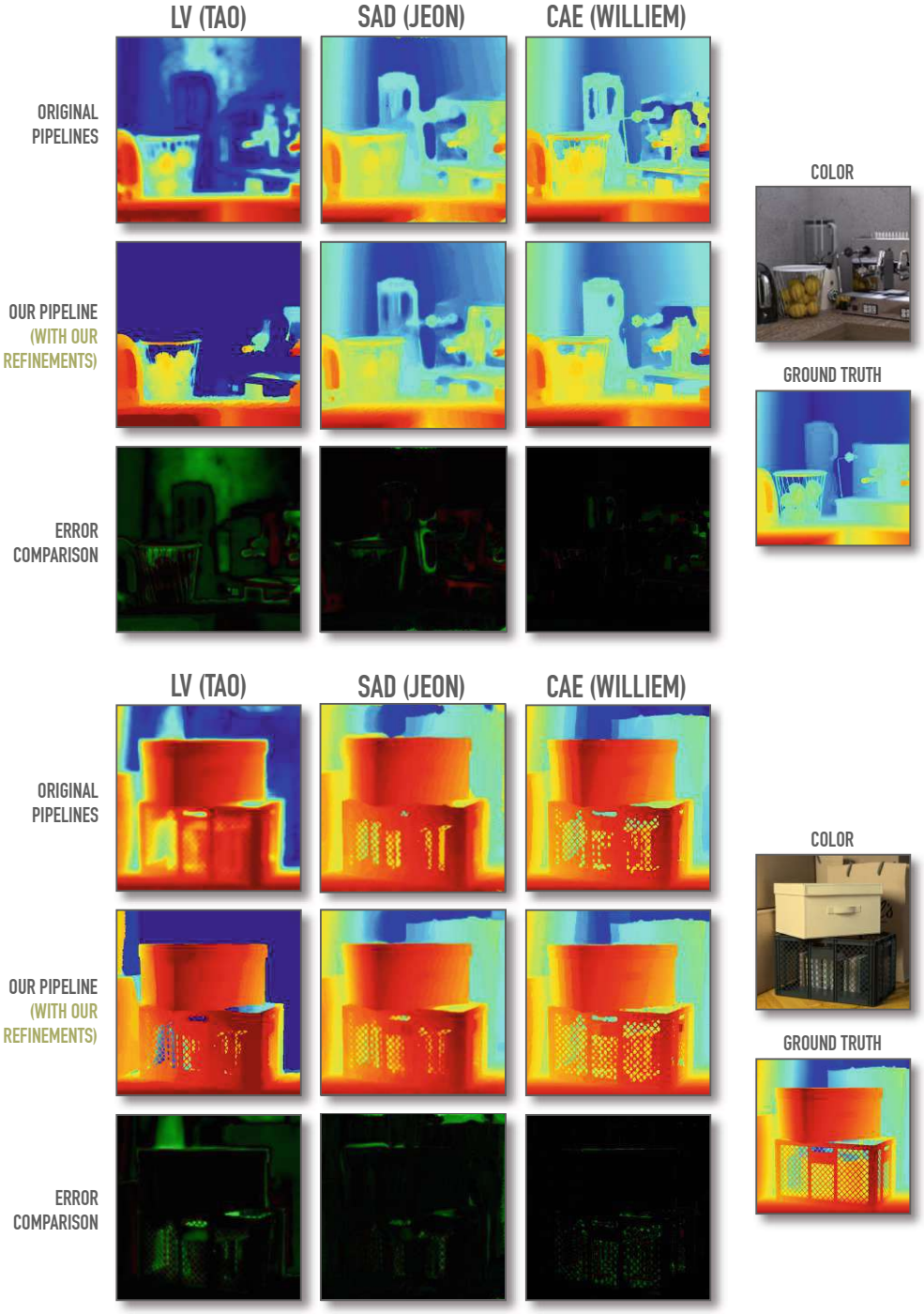


Figure 4.6: **Refinement Results**. Analysis of refinement performance in two scenes from synthetic dataset [HJKG16]. Top rows show depth predictions using three existing cost volume estimation methods, each processed exactly as in their original work (see Figure 2.3). Middle rows show predictions of the same cost volumes, but processed with our pipeline instead (which includes refinements described in Sections 4.2.2 and 4.2.3). Finally, we compare the per pixel ground-truth reconstruction errors between each top and middle row pair, as explained in Section 4.6. Green means a lower error, red a higher.

CHAPTER 5

Context-Aware Translation

In this chapter, we delve onto the subject of automation of detail in hand-drawn animation, the final topic first introduced in Chapter 1. The research findings discussed will be published at the *33rd International Joint Conference on Artificial Intelligence* under the title '*Re:Draw - Context Aware Translation as a Controllable Method for Artistic Production*". Meanwhile, it is publicly available at arXiv [CBCW24]. Despite the resurgence of traditional hand-drawn animation over the past decade, the labor-intensive techniques that have remained largely unchanged despite advances in computer graphics. Specifically, we focus on the time-consuming process of drawing character faces, particularly the eye region of characters, and how this often leads to compromises in design complexity and artistic consistency. Our aim is to introduce computational methods, particularly optimization techniques, that can alleviate some of these challenges without sacrificing the artistic integrity of the medium. Figure 5.1 provides an illustrative example of how

Figure 5.1: **Chapter Overview.** The context-aware translation proposed in this chapter is capable of automatically redrawing parts of images according to any provided design, without the need for fine-tuning. Unlike image-to-image translation [LHM+19], which neglects surrounding context, our approach considers the entire frame. Unlike inpainting [Lah23], which lacks artistic control by ignoring the original content, our method honors the artist's input. This facilitates the production of more consistent artwork and allows for more complex design choices.

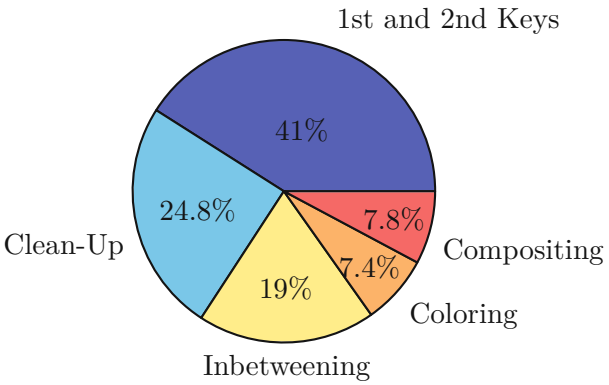


Figure 5.2: **Industry Survey**. Average reported time spent in the different stages of production of a cut, in percentage.

our method can automatically redraw the eyes of characters to produce consistent and detailed designs.

We conducted a survey among 17 professional animators, of which 29.4% (5/17) work at established studios, and the rest either freelanced or worked for smaller studios. We asked them multiple questions about time consumption, (not mandatory answer), of which the full breakdown is shown in Figures 5.2 and 5.3. Character faces were reported to be the most complex part of animation, with 50% (8/16) reporting it as the element they spend the most time on. Of the remaining animators, 75% (6/8) voted either anatomy or hair as the most time-consuming element. Drawing was estimated to constitute the vast majority of the work (84.8%), and doing it at the highest level of detail was estimated to take 1.7 times the amount of work than on average, for a total of 66m of additional human effort per key frame from 1st key through coloring. Sadly, the fact that 52.9% (9/17) still use paper drawings in their studios, despite 100% preferring to draw digitally, indicates that using computational tools during the early drawing stages might not be possible yet in practical terms.

Deep learning, powered by stochastic gradient descent, is a promising solution for applications within hand-drawn animation. This approach allows the training of models that can learn the meaning of animation frames and the different styles and techniques of different artists. Yet, existing deep learning methods present significant limitations within artistic applications. Inpainting, while capable of generating detailed art that fits within existing content, offers little control over the generated content, making it unsuitable for most precise artistic endeavors [AMT20]. Image-to-image translation, while being able to take artistic input, is constrained by only being applicable to entire images, as it does not take into account the context surrounding target regions.

We propose context-aware translation as the solution to these limitations. We then apply it in a novel pipeline that automates increasing consistency and amount of detail in the eyes of hand-drawn animation characters. It effectively mimics the work of cleanup

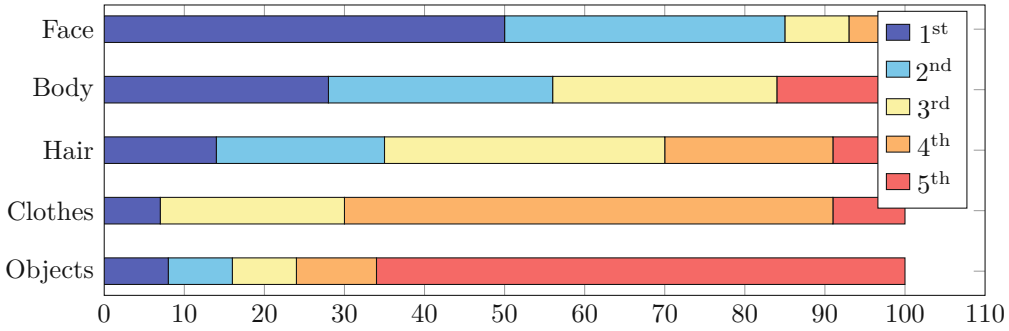


Figure 5.3: **Industry Survey**. Reported ranking of common elements in animation from most to least time consuming. Labels '1st', '2nd', etc., indicate how often it was ranked as the most time-consuming, second-most time-consuming, etc.

Figure 5.4: **Color Guide**. Also known as a model sheet, this type of document depicts all the information an artist is expected to need to know how to draw a character in accordance with the production direction and style. Example from [KSP11].

animators, who redraw frames to fix mistakes and better match the character color guides – despite the misleading name, color guides, also known as model sheets, depict all the information an artist would need to draw a character while remaining true to its intended design and the art style of the production (see Figure 5.4 for an example). We also tackle an additional problem this use-case raises: the lack of training datasets of anime production, which we address by proposing methods to negate the need for production data entirely, including a novel character recognition method.

In this chapter, we present several key innovations spread across different areas of optimization. These include a novel model structure, the development of new reconstruction and adversarial loss functions, and a new training method that we call context-aware translation, which leverages both the aforementioned loss functions to offers significant advantages over existing methods. Here's a breakdown of our core contributions:

1. Context-aware translation, a **novel general deep-learning method** that avoids the limitations of both inpainting and image-to-image translation. This includes:
   - A **dual discriminator structure** and **novel adversarial losses** that enforce simultaneous respect for input content, translation requirements, and context constraints.
   - A **triple-reconstruction loss** that yields greater generation capabilities than traditional loss.
2. A **novel network architecture** for character design recognition, that employs a production style-aware latent space to outperform existing work, and is able to generate a robust training dataset for adversarial supervision.
3. A **novel pipeline** that takes advantage of the aforementioned contributions to automatically increase the consistency and amount of detail in the eye region of characters, and without the need of production data during training.

Furthermore, we present ablation and a large user study on the perceived quality of our method in Section 5.2. Section 5.1 describes the proposed pipeline and its contributions.

5.1 Method

The goal of our method is to ensure the design and level of detail of animation frames match the desired look in subpar drawn regions. In this chapter, we concentrate on the eye regions, as they are often the most salient features in an animation and, as character identifiers, artists treat them differently, not following the same anatomical rules as the rest of the artwork. As input, our approach takes the animation frames to be improved and a character color guide (a set of cels of the character drawn by the character director in high quality and in different poses and/or expressions); see Figure 5.5. We use an

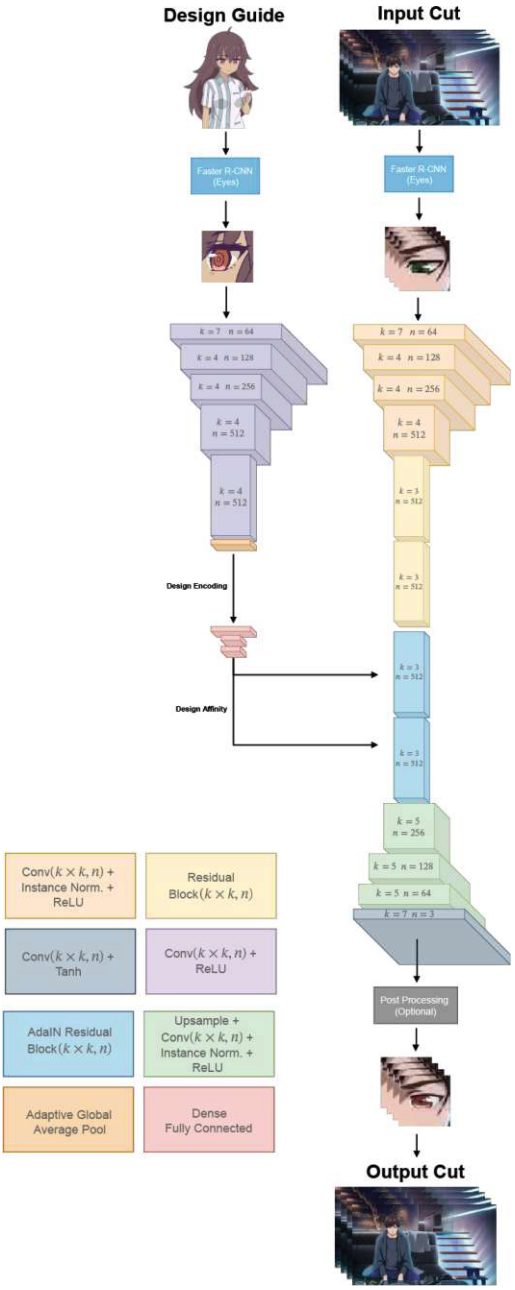


Figure 5.5: **Redrawing Pipeline**. Starting from a sequence of frames and a color guide, eyes are detected, fed to our neural redrawer generator G, and the resulting styled eyes are post-processed into the original art.

unsupervised convolutional network trained alongside classification networks, capable of telling designs apart, as its adversaries.

Using such a model would require artists to manually associate regions to redraw and color guides manually, which is not practical. Even more problematic, to train this type of adversarial structure, one would normally use pairs of these images, labeled into different classes (character designs). In particular, to ensure our model is capable of generalizing to new designs, we need to train on a large enough variety of them. Yet, art direction is not easily available and generally not created in high enough quantities that would be needed for a robust training. Moreover, manually tagging and cropping this data would be extremely labor intensive and hard to replicate. Instead, we solve the association problem, making Re:Draw both practical and only require a set of randomly sampled frames from different productions for training, with no internal production data being needed; see Figure 5.6. In Section 5.1.1, we propose a deep learning approach, reliant on a novel character design clustering method, and use it to automatically infer enough training data from random frames.

Image in-painting has shown to be capable of completing missing regions, yet predictions based only on the surrounding of the area to be redrawn, do not allow artists to finely control the output results using art or style direction examples. Image-to-image translation and style transfer are capable of using both of these inputs, yet existing work is incapable of generating art that fits and correctly matches within the actual context of the drawing: they can be very unreliable in preserving the artwork pose. For these reasons, in Section 5.1.2 we introduce a novel approach distinct from both image translation and inpainting. We make use of two adversarial discriminators built using partial convolutions, allowing them to weight images differently and independently, and a novel triple reconstruction loss based on the concept of the generation of image triplets. To our knowledge, this approach for style-guided detail enriching is different from existing literature.

5.1.1 Dataset Generation

We aim at translating art to any given design by leveraging a high variety of target classes during training, which allows the multi-classification adversarial networks to learn to tell apart. We will now describe how we automatically cluster art by character design to create this high variety of classes and how we split them into low- and high-levels of detail.

Object Detection We first train an object-detection network – we use the well-established Faster R-CNN network [RHGS15] – to identify character faces and details in them (such as eyes) and run it on the randomly sampled frames. This results in a dataset of character faces in a variety of poses, split by the sources they were sampled from.

Style-Aware Clustering Although re-identification of human faces is a long studied topic [Bal15], we found existing work to be ineffective at automatically identifying

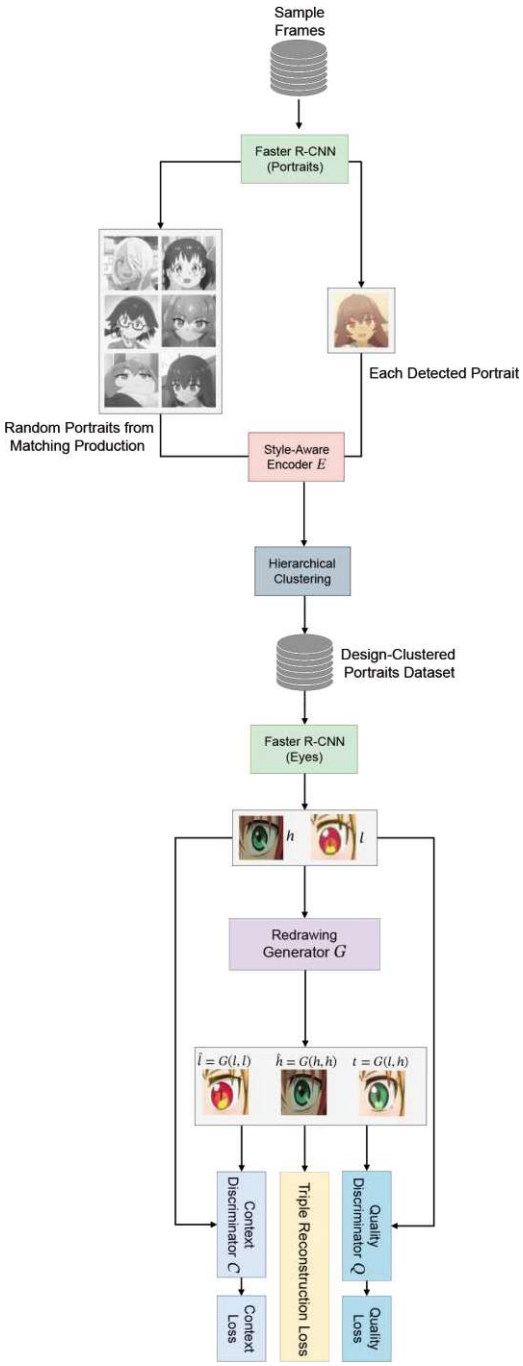


Figure 5.6: **Redrawer Training**. Detected portraits are clustered using our style-aware encoder (left), with additional portraits as style guides. Content and style images are extracted from portraits of different designs (center). These are used to generate three redrawings from different combinations of these input pairs, which are judged by multiple losses, including two multi-class discriminators (right).

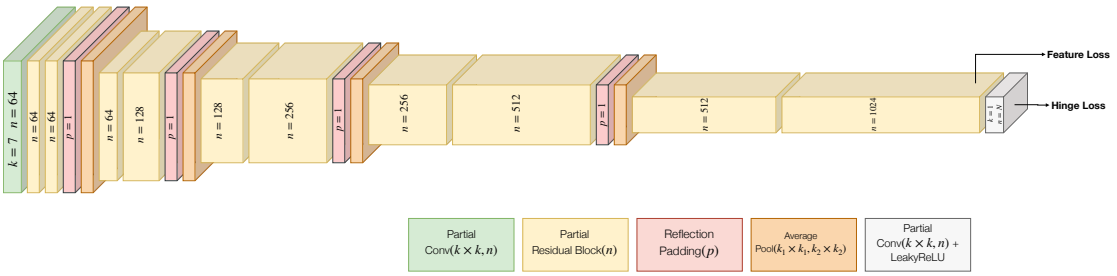


Figure 5.7: **Discriminator Architecture** Both the quality and context discriminators described in this Chapter share this architecture.

animated characters not seen during training. We attribute this to the fact that, while human faces have a consistent and predictable structure, animated characters are not restrained by the laws of reality and thus present a much higher variance: in a given production, characters might have a very similar look and feel, while in another production they might vary widely in structure and shape(see Figure 5.8).

To solve this issue, we improve upon the state of the art of character recognition by combining ideas from facial recognition and image-to-image translation. We propose a supervised network that, unlike existing work, maps character portraits to an art-style normalized Euclidean space, where distances between these portraits correspond to a measure of character design similarity within its production. It takes as content input a character portrait and as normalization input a collection of random portraits from the same production.

As shown in Figure 5.9, latent representations of both inputs are estimated: we compute the content representation using a ResNet [HZRS16] encoder – which is well established for object recognition – and the production representation using a convolutional encoder. The latter is done using only the lightness in $l\alpha\beta$ color space [RAGS01], as we found that the normalization input works better if it only contains the main shape information, so we use a color space to decorrelate it from color variation. This style-latent representation is then used to compute a set of affine transformations, with the goal of mapping the encoding from an absolute Euclidean-space representation of portraits to the style-normalized one. This mapping is done using Adaptive Instance normalization [HB17] on the content-input latent representation. This finally results in 32 parameters per portrait thanks to the linear layers, which are to be used by a clustering algorithm. We found using traditional hierarchical clustering, using unweighted pair group method with arithmetic mean and Euclidean distance, to give the best results across the compared methods.

To train this network E and ensure the content-encoding output respects the desired intra and inter-class proprieties of the normalized Euclidean space, we use the option [HBL17] of Triplet Margin Loss [BRPM16] – that is, given a pair of portraits from the same design $\{P_1, P_2\}$ and one from another P_3 but from the same production $\mathbb{P}$, we minimize the distance from the first two while maximizing the distance of the third:

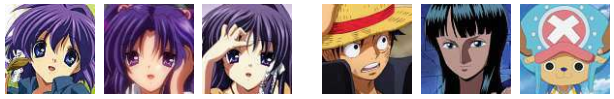


Figure 5.8: **Design Variety**. While characters in some productions have very similar structures (left [KK07]), yet others may present a high variety of designs (right [OT97]). This hinders traditional facial recognition.

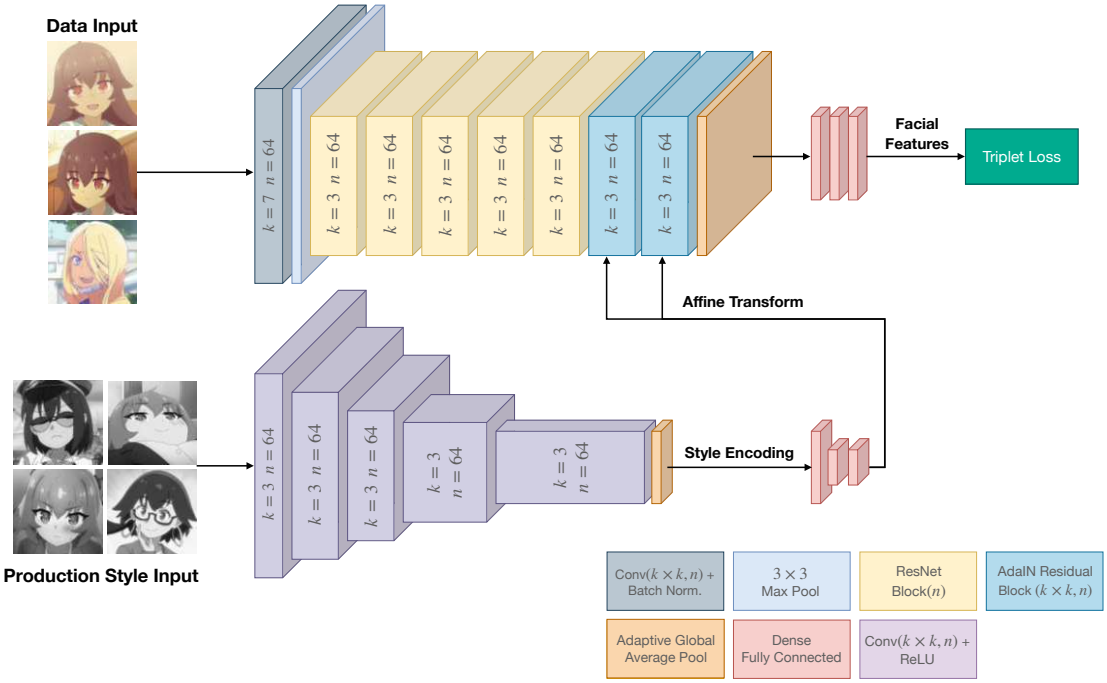


Figure 5.9: **Style-Aware Encoding** We train a style-aware encoder network capable of generalized character design recognition using triplet loss. Unlike traditional facial recognition, where only a set of labeled portraits is seen, we additionally input random unlabeled portraits to account for production style.

$$\underset{E}{\operatorname{argmin}}\ max\{\|E(P_1,\mathbb{P})-E(P_2,\mathbb{P})\|^2- \|E(P_1,\mathbb{P})-E(P_3,\mathbb{P})\|^2+1,0\} \tag{5.1}$$

We also ensure that the total training weight of each production style and of each character design within each style is the same, to further help with generalization. While it is technically possible to train E in conjunction with the redrawer and its adversaries, it is much more computationally efficient to train E first and then freeze it to train the remainder networks.

Level-of-Detail Split Having the portraits organized by character design, we extract the intended art details (eyes in our case) from them using again a Faster R-CNN network,

Table 5.1: **Generated Dataset**. Statistics of our animation frames dataset generation.

Selected Productions	48	Predicted Elements	35884
Marked Designs	23	Element Mean Size	14395 $pixels^2$
Sampled Frames	14338	Content Images	20374
Predicted Designs	476	Style Images	15510

but now with the knowledge of their corresponding designs. We standardize all the extracted art details to the same size. Then, we exploit the fact that characters are often drawn with different levels of detail, depending on their prominence on screen to discriminate between low and high details regions. So, after extensive empirical observation, regions representing less than 0.31% pixels of the frame were assumed to be low-detail and to be redrawn, while regions with more than 0.48% pixels were used as art direction examples (see Table 5.1).

5.1.2 Redrawing

Having generated the dataset, we can now train the redrawing model. We want to optimize a function $x^+ = g(x, \mathbb{S})$ that, given a low-detail content image x and color guide $\mathbb{S}$ (equivalent to style images in a style transfer context), is capable of outputting a higher detail version x^+ of x. This leaves us with conflicting goals: we want the translated artwork x^+ to match the provided design $\mathbb{S}$ and its level of detail, but to still fit within the original drawing of x. Our problem can thus be viewed as a multi-objective optimization problem, where we aim to find a Pareto-optimal solution that balances between design fidelity, original drawing fit, and detail enhancement.

We define an image-to-image generator network G, with the purpose of approximating g. As illustrated in Figure 5.5, it is composed of a convolutional encoder-decoder structure with an additional style encoder. The latter matches exactly the encoder described in Section 5.1.1 and is used to compute a set of affine transformations that control the Adaptive Instance normalization in the decoder.

Triple Reconstruction Let l be a low-detail image and h a high-detail one, each sampled from different designs $\mathbb{L}$ and $\mathbb{H}$, respectively. Our approach is to train the generator as an image translation problem such that $t = G(l, h)$ outputs the result of applying design $\mathbb{H}$ to l.

To help G learn a translation model and ensure it maintains the local structure of l, a second output $\hat{l} = G(l, l)$ is frequently used as part of a reconstruction loss [HLBK18]. However, this is not appropriate for our problem, as we are not interested in the network producing low detail images. We propose a novel reconstruction loss that analyses a total of three generated images:

$$\mathcal{L}_R = [\|h - G(h, h)\| + \|F(l) - F(t)\| + \|F(l) - F(\hat{l})\|]_1^1, \tag{5.2}$$

where $F(x)$ is a low-pass image filter applied on the lightness of the image x, implemented by converting to frequency space using fast Fourier transform and remove any frequencies

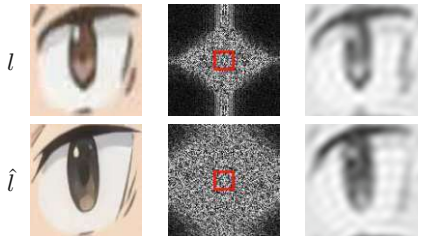


Figure 5.10: **Low-Pass Filter** $\hat{l}$ is computed during training from enhancing l. Right shows the result of our low pass filter.

above a set threshold (0.06, as shown in Figure 5.10). The basis is that, by removing high frequencies and color changes, differences between low-detail and high-detail images are ignored as well, allowing us to create a reconstruction loss in low-detail images.

Adversarial Discriminators Unsupervised networks are generally trained using one adversarial discriminator tasked with telling real and generated images apart. To address our aforementioned conflicting goals, we instead employ a dual-discriminator optimization strategy, by using two independent image multi-task classifiers: a *quality discriminator* Q judging whether the output is high detail and matches the intended design, and a *context discriminator* C judging whether it fits within the original artwork and its own design, irrespective of detail-level.

To achieve this purpose, as we show in Section 5.2, we need to train each discriminator differently, despite sharing many commonalities: both have the same partial convolutional structure shown in Figure 5.7 and are trained using hinge loss with R1 regularization [MGN18], to prevent over-fitting and mode collapse. This results in the following losses for penalizing wrong classifications of positive $\mathcal{L}_P$ and negative $\mathcal{L}_N$ examples:

$$\begin{aligned} \mathcal{L}_P(x, \mathbb{S}) &= [\max(0, 1 - D(x)_{\mathbb{S}}) + \gamma \|\nabla D(x)_{\mathbb{S}}\|^2]_1 \\ \mathcal{L}_N(x, \mathbb{S}) &= [\max(0, 1 + D(x)_{\mathbb{S}})]_1 \end{aligned}, \tag{5.3}$$

where $D \in \{Q, C\}$ can be any one of the discriminators, $D(x)_{\mathbb{S}}$ is the discriminator's score of image input x for design class $\mathbb{S}$, $\nabla D(x)_{\mathbb{S}}$ its derivative used for R1 regularization and $\gamma = 10$ (R1 standard weight). That is, D should converge to $[0, 1]$ for positive entries and to $[-1, 0]$ otherwise. The discriminators are then given different input masks to weight disparate regions of images differently: the quality discriminator Q focuses on the interior of the redrawn region, while the context discriminator C focuses on the opposite, including an outer border that is not redrawn. They meet and oppose each other in the intersection of their two regions. Finally, they are trained to judge the training image pairs $\{l, h\}$ and the generated triplets $\{t, \hat{l}, \hat{h}\}$ such that Q looks for high detail output, while C tries to tell real and generated art apart:

$$\begin{aligned} &\underset{Q}{\operatorname{argmin}}\, \mathcal{L}_P(h, \mathbb{H}) + \frac{\mathcal{L}_N(l, \mathbb{H}) + \mathcal{L}_N(t, \mathbb{H})}{2} \\ &\underset{C}{\operatorname{argmin}}\, \mathcal{L}_P(h, \mathbb{H}) + \mathcal{L}_N(t, \mathbb{H}) + \mathcal{L}_P(l, \mathbb{L}) + \mathcal{L}_N(\hat{l}, \mathbb{L}) \end{aligned} \tag{5.4}$$

That is, Q attempts to learn to identify real high-detail images as positive examples, and low-detail or generated art as negative ones; while C attempts to identify real as positive and generated as negative, independently of detail. Images are always judged for the design class they are supposed to belong to. Then, to train the generator network G using these discriminators, we use hinge loss with a latent feature loss to regularize the adversarial training. Let D^F be the latent features computed by a discriminator D in a hidden layer, and s a sampled image (either l or h) from the given design class $\mathbb{S}$. The adversarial loss function of each discriminator D becomes:

$$\mathcal{L}_D(x, \mathbb{S}) = [1 - D(x)_{\mathbb{S}}]_1^1 + [D^F(x)_{\mathbb{S}} - D^F(s)_{\mathbb{S}}]_1^1 \tag{5.5}$$

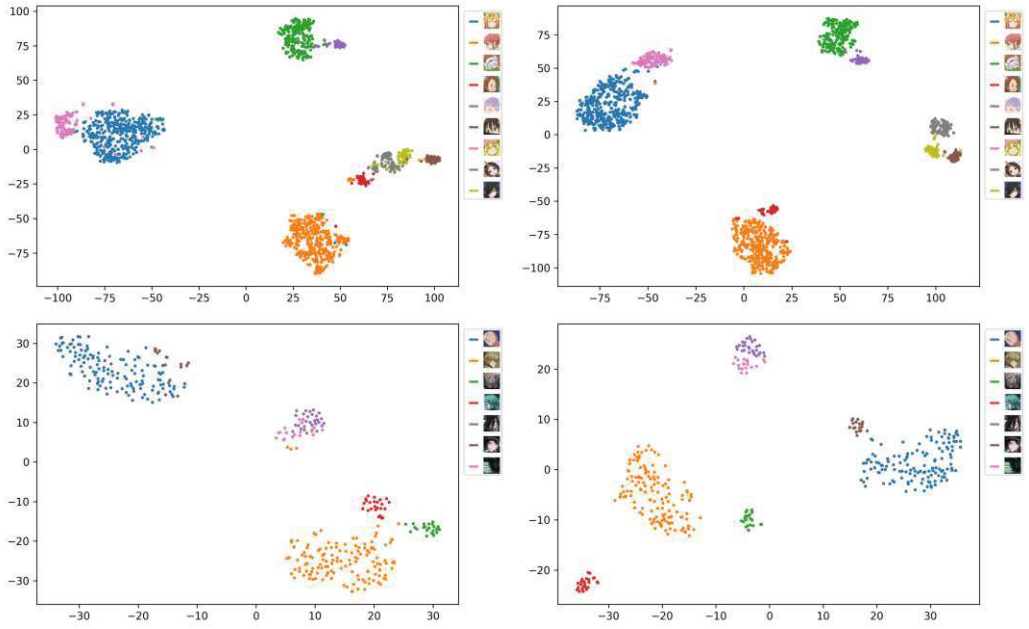
The use of hinge loss, R1 regularization and feature matching loss have been used in different forms in image-to-image translation problems [LHM+19, SSL20]. Just as with training the discriminators themselves, our contribution is how these are then used to train a generator capable of addressing our problem. We combine our novel reconstruction loss with two adversarial losses to verify discriminator conditions and another two to ensure reconstruction persistence, where $\mathcal{L}_Q$ and $\mathcal{L}_C$ are the adversarial functions of each discriminator, defined in Equation 5.5

$$\underset{G}{\operatorname{argmin}}\, \mathcal{L}_R + \mathcal{L}_Q(t, \mathbb{H}) + \mathcal{L}_Q(\hat{l}, \mathbb{L}) + \mathcal{L}_C(t, \mathbb{L}) + \mathcal{L}_C(\hat{h}, \mathbb{H}) \tag{5.6}$$

Post-Processing Figure 5.13 shows that regions generated on a model trained without a context loss will not necessarily fit perfectly with the rest of the art. The losses described in Section 5.1.2 largely fix this issue, but to make our approach more robust, we apply a few image processing operations: after re-sampling the network output to the original resolution, we apply color transfer [RAGS01] and place it into the original frame using Poisson image editing [PGB03].

5.2 Ablation Experiments

Clustering We compare our style-aware clustering approach with FaceNet [SKP15], trained on the same labeled animated character faces. We statistically analyze how effective the latent representations learned from either method work on a validation dataset of productions styles not seen during training: we measure the ratio of the average squared norm distance between each point of the same character, and the average squared norm distance between the mean points of each character. The lower this value, the better



(a) FaceNet (b) Ours

Figure 5.11: **Clustering Validation** 2D visualization of characters faces from productions not seen during training [CD17, OMH15], generated using T-Distributed Stochastic Neighbor Embedding [VdMH08] on the latent spaces learned using FaceNet [SKP15] and our proposed encoder network. Colors correspond to ground-truth labels. As shown, character differentiation is clearer in our learned space.

(a) Input (b) Ours (c) Input (d) Ours (e) Input (f) Ours (g) Input (h) Ours

Figure 5.12: **Redrawing Results**. The method proposed in this Chapter succeeds at the intended goal of automatically associating color guides with corresponding cells and performing enhancement for productions and characters unseen during training [OT22].

Figure 5.13: **Ablation of Contributions**. We compare our method with traditional image translation by progressively introducing our contributions (rows 4 to 6). The synthetic data extrapolated using our style-aware clustering prevents over-fitting (a,e) and allows generalization to unseen designs (g,h,i). Further introducing our dual discriminators allows redrawn areas to maintain artwork fit, but sacrifices detail (a,b,c,e,i) and ability to redesign shape or color (g,h). Finally, introducing the triplet reconstruction loss brings that expressiveness back.

is the latent space representation in principle. Our method measured a ratio of $1.212e^{-4}$ in the validation data, which outperformed FaceNet's $1.494e^{-4}$ ratio. Interestingly, our method performs similarly to a FaceNet network trained on the validation dataset, whose ratio was $1.209e^{-4}$. This shows our method presents better generalization to unseen data. Furthermore, we show in Figure 5.11 how well split the character faces from a validation production styles are. The colors of the points represent their ground-truth labels, and there is a visible improvement with our method.

Redrawing We evaluate our redrawing approach against neural cross-domain image translation with content and style inputs. While multiple variations of these networks exist [SSL20, OLL+21], they mostly compete in translation ability and do not address our problems. Thus, we chose as a baseline for comparison Liu *et al.* [LHM+19] method (FUNIT). We progressively introduce each of our novelties to show the importance of each one (Figure 5.13): we introduce our style-aware clustering by training FUNIT only on the few manually labeled faces versus the generated dataset. We then add our double-discriminator method, but maintain FUNIT's traditional reconstruction loss.

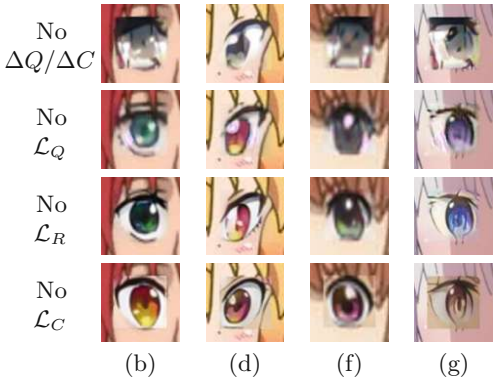


Figure 5.14: **Ablation of Losses**. We test removing each loss during training of redrawer G on examples from Figure 5.13. It shows $\mathcal{L}_Q$ and $\mathcal{L}_R$ are crucial for useful results. Most notably, it reveals $\mathcal{L}_C$ makes output much more predictable than otherwise expected, and this stability is what enables the use of our less explicit $\mathcal{L}_R$.

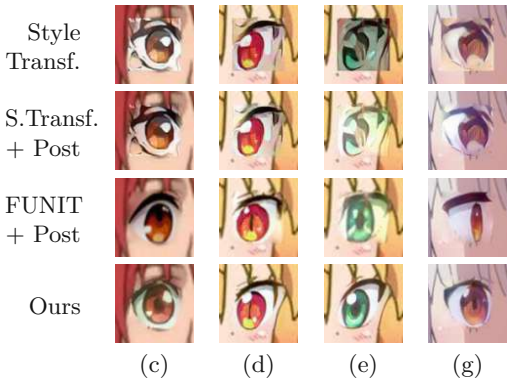
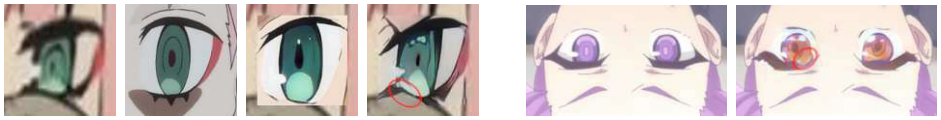


Figure 5.15: **Related Work Ablation**. We compare our method with style transfer and test using our post-processing on image to image translation.

(a) First Design (b) Second Design (c) Second Design (d) Third Design

Figure 5.16: **Ablation of Diffusion**. We try unsuccessfully to recreate Figure 5.1 using Dall-E [RPG+21]. The biggest challenge in our use case is not to generate believable eyes, but to be able to match artistic specifications, and despite the recent advances of diffusion methods for text to image generation, they haven't proved adequate yet for use in drawn animation.



(a) Input (b) Design (c) FUNIT (d) Ours (e) Input (f) Ours

Figure 5.17: **Redrawing Limitations**. Our method vastly improves upon existing work on handling occlusions; however, it still can struggle with uncommon scenarios (top). As it was trained mostly with upright eyes, fails if input is not pre-rotated to upright position (bottom).

Finally, we introduce the proposed triplet reconstruction, followed by the post-processing step. FUNIT fails to respect pose/expression and context. Without our large-scale dataset, it often fails to generate realistic art and cannot generalize to designs not seen during training. Our dataset and double discriminator approach solve all of these issues, but reduce the ability of the network to generate highly detailed art. Our novel triplet reconstruction fixes that.

Our method is also very stable and demonstrates temporal coherence, shown in Figure 5.21, despite not specifically incorporating it in the loss. We believe the reason for this consistency is its emphasis on preserving the intended pose and context of the drawing: as shown in Figure 5.14, removing the regularizer from the adversarial losses results in mode-collapse, and removing the quality loss results in a poor auto-encoder as expected. Yet, removing the reconstruction loss does not result in unpredictable output as expected, just an inability to learn useful transformations, and removing the context loss does not result in output reminiscent of FUNIT. Our explanation is that the network, by following the context loss, is being incentivized to exhibit spatial and temporal consistency in its output. This stability is what enables our less explicit form of reconstruction loss to direct the training toward useful results.

We critically evaluate the performance of existing techniques—namely style-transfer, image-to-image translation, and text-to-image diffusion—against our proposed context-aware translation. Figures 5.15 and 5.16 clearly illustrate the limitations of these existing approaches, thereby reinforcing why our method is the most apt solution for this particular type of problem. Additionally, Figures 5.12, 5.18, 5.19 and 5.20 provide compelling evidence of our method's robustness and versatility in handling a variety of challenging scenarios, further substantiating its suitability for this application.

The limitations of the network we found boiled down to two cases: uncommon occlusions and strong rotations. As most occlusion to anime eyes is hair and the vast majority of are drawn up-right, the network can generate artifacts outside of those expected conditions. As shown in Figure 5.17, the shape of occluders below the eyeline might be distorted as if was a skin tattoo. Eyes will be drawn upright if a character is reversed. Adding a network capable of estimating eye rotation to the pipeline would automate this process and further improve the generated dataset. But outside of these two unusual spaces, artifacts we did find were created by Poisson-blending, not our models, which leads us to

Figure 5.18: **Increasing Detail**. The method behaves as intended when prompted to increase the amount of detail in character eyes, despite challenges such as unusual face proportions, adverse color balance due to lighting effects, head tilts, minor hair occlusions and character designs from productions that have not been seen during training.

conclude post-processing is the current main limitation of the method.

5.3 User Study

To validate our work, we conducted a user study with 63 participants and three different tasks with images coming from an anime production. Participants applied voluntarily to the study and the majority of them classified themselves as highly familiar with anime, as shown in Figure 5.2. Three visual tasks were presented:

Realness (T1) The user had to watch eight images and pick the ones with drawing problems. In this test, four were original un-retouched images (*Market, Tent, Cold Night, Field of Leaves*), and four had the eyes replaced by our approach (*Dormitory, Bar, Bathroom, Crying*). Images were presented in random order, and the user did not know how many could have problems. This study aims to determine if our method generates visual artifacts or if its result is not distinguishable from a production. To analyze these

Figure 5.19: **Redesign Results**. A side-effect of how our proposed networks are trained is that our method is also capable of applying entirely different designs to characters. While not the focus of our work, it demonstrates the versatility of our method and the robustness of the learned model, even when applied to high-resolution imagery.

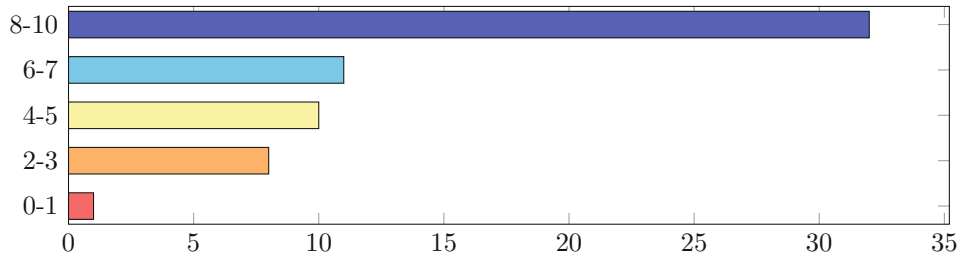
results, we used the χ^2 test with Yates correction [SC88], and our null hypothesis is H_0:*"Original and retouched images are equally probable to be considered as wrong"*. The number of times users decide that an image is wrong has a very similar distribution for images generated by our approach and original ones (see Table 5.4 and Figure 5.3). The $\chi^2(1, N=504)=0.0405, p=0.840$, therefore with a $p$value larger than 0.05, we tested

Figure 5.20: **Redesign Results**. Further examples of character redesign which demonstrate that our method is robust enough to handle characters in a variety of challenging scenarios. This includes dealing with oblique camera angles, irregular lighting conditions, head tilts and occlusions caused by hair.

the independence between groups (*Our vs Real*) and categories (*Wrong vs Right*), or in simpler words, our generated eyes do not have evident artifacts and are indistinguishable from real production.

Figure 5.21: **Temporal Coherence**. Example of redesigning the eyes of a character [NKR16] in a particularly lengthy shot. Output is temporally consistent.

Table 5.2: **Domain Knowledge**. More than half of the users classified themselves on the high end of an anime (viewer) knowledge spectrum.



Level of Detail (T2) The user had to select between two images the one with more details in the eyes. Each user evaluated eight image pairs: one unedited (U) and one with eyes replaced by our approach (O). This study aims to determine if we effectively produce images with higher detail than the original production. To understand if the expressed preferences are statistically significant, we employed the formula for the multiple comparison test[Dav88], where the score significance of a method against another one is present if the difference in scores between the two methods is greater than the critical value R. We also tested if the agreement $u \in [-1, 1]$ (where $u = -1$ strong disagreement and $u = 1$ strong agreement), among observers; with the following null hypothesis; H_0: *"There is no agreement among participants on one category being better"*. The failure of this hypothesis implies that categories are perceptually equivalent causing difficulties in judging (e.g. we check if users alloted their preferences at random). The overall result (Table 5.5) is that our method effectively produces images with higher detail than the original production artwork with statistical significance.

Preference (T3) The user had to select the better looking image from a pair. For each of the 8 scenes, there were three pairs cross-comparing 3 methods: FUNIT[LHM+19] (F), FUNIT with our clustering (FC), and our method (O); in total, 24 pairs for each user. This study aims to determine if our method is more effective in producing results better looking than F and FC. In short, our method was preferred to (F) 95.16% of times. In Tab. 5.6 we used the same analysis of **T2**, showing that images generated by our method are more preferred than the ones generated by the previous work (F), even when it is enhanced with our dataset (FC).

Table 5.3: **Realness Test (T1)**: The number of times that our redrawn art frames ■ and original unmodified ones ■ have been picked as fake is not significantly different.

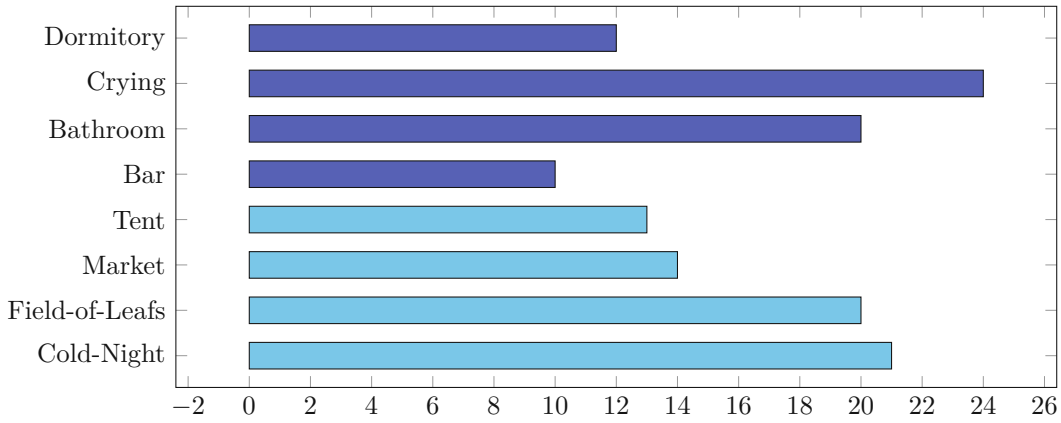


Table 5.4: **Realness Test (T1)**. We test independence between groups (Our vs Real) and categories (Wrong vs Right), i.e. our generated eyes are indistinguishable from real artwork.

	Wrong	**Right**	**Marginal Row Totals**
Ours	66	186	252
Real	69	183	252

Table 5.5: **Level-of-Detail Test (T2)**. The R values (threshold to determine statistical difference) are, respectively, $R = 45$ for the overall and $R = 16$ for each single image at a significance level $p < .05$. The χ^2 critical value is 3.84 to pass the test at a significance level $p < .05$. Techniques not statistically significant different are circled together.

Image	Agree. Coeff. u	χ^2	Sign. u	Groups
1	0.230	15.25	OK	U - 16 O - 47
2	0.041	3.57	NO	U - 24; O - 39
3	0.119	8.40	OK	U - 20 O - 43
4	0.334	21.73	OK	U - 13 O - 50
5	0.373	24.14	OK	U - 12 O - 51
6	0.297	19.44	OK	U - 14 O - 49
7	0.058	4.59	OK	U - 23 O - 40
8	0.144	9.92	OK	U - 19 O - 44
Overall	0.192	97.79	OK	U - 141 O - 363

Table 5.6: **Preference Test (T3)**. The R values (threshold to determine if a score is statistically different) are, respectively, $R = 65$ for the overall and $R = 24$ for each single image at a significance level $p < .05$. The χ^2 critical value is 7.82 to pass the test at a significance level $p < .05$. Techniques not statistically significant different are circled together.

Image	Agree. Coeff. u	χ^2	Sign. u	Groups
1	0.460	73.38	OK	FC - 32; F - 40 O - 117
2	0.808	126.61	OK	F - 5 FC - 63 O - 121
3	0.856	134.00	OK	F - 2 FC - 68 O - 119
4	0.830	129.92	OK	F - 4 FC - 63 O - 122
5	0.396	63.53	OK	F - 20 FC - 67 O - 102
6	0.307	49.92	OK	F - 32 FC - 59 O - 98
7	0.526	83.53	OK	F - 17 FC - 56 O - 116
8	0.413	66.23	OK	F - 23; FC - 52 O - 114
All	0.530	802.93	OK	F - 143 FC - 460 O - 909

CHAPTER 6

Conclusion

This thesis embarked on a journey to explore novel optimization approaches in the field of computer graphics, with a particular focus on three critical areas: image-to-image translation, perceptual metrics prediction, and depth prediction for light-field data. The motivation behind this exploration was to address specific challenges that have long plagued these domains, thereby contributing to both the academic and practical bodies of knowledge.

In Chapter 3, we attempted to address the computational challenges associated with prediction of perceptual metrics in real time, particularly in the context of high-demand applications. The motivation for this exploration was rooted in the limitations of existing methods, which often rely on heuristics based on previous frames, or require reference or rendered images and are not optimized for real-time performance. Our goal was to develop a learning-based approach that could efficiently predict these metrics in real-time settings based on current information of the scene, and we successfully designed and implemented a compact convolutional neural network architecture designed for this very task. Specialized solutions were also developed to tackle unique challenges, such as unbalanced training data and the integration of visually based decision making with just-noticeable differences, which were essential for practical real-time application. These innovations also significantly enhanced the robustness and generalizability of our solution across various metrics and scenes.

By leveraging recent advancements in GPU hardware, our method demonstrated its capability for real-time inference, paving the way for new uses of visual error metrics in real-time rendering applications. Furthermore, our contributions extend beyond the academic sphere; our contributions were further supported by the open-sourcing of our codebase and datasets, available at `jaliborc.github.io/rt-percept`, encouraging experimentation and research in this domain. This chapter not only showcased optimization as an intrinsic facet of training neural networks and formulating novel loss functions but also demonstrated how our method can be employed to optimize real-time

rasterization by dynamically selecting optimal shading rates across the screen. The potential benefits of our approach were further validated through its implementation at highly interactive frame rates, showcasing its applicability in complex and dynamic environments, as well as in scenarios requiring predictions to be amortized over multiple frames. As for future work, the newfound ability to use a variety of perceptual metrics in real-time applications presents a fertile ground for exploration. Potential avenues include identifying novel applications that can benefit from these metrics and experimenting with amortization techniques to further optimize performance.

In Chapter 4, we turned our focus to the specific challenges of depth prediction in light-field cameras, a subject of active research due to the unique architecture of these devices. The driving force behind this research was to enhance the existing cost-volume based approaches for depth prediction, which often rely on computationally expensive and limited image-refinement methods. Our aim was not to reinvent the wheel but to improve upon it, by introducing a novel refinement approach that is capable of leveraging the inherent redundancy in light-field data. To this end, we introduced a modular framework for cost-volume refinement.

This framework demonstrated its effectiveness across a variety of cost cues and was rigorously evaluated using a publicly available dataset, thereby confirming its robustness, adaptability, and effectiveness. This modular approach not only improved upon existing methods but also demonstrated that there is room for further research and development in the field of cost volume regression. We also took the initiative to open-source our codebase, available at `github.com/cg-tuwien/CostRefinement`, to encourage further experimentation and research in this domain. From promising directions to explore, two avenues we want to highlight is the refinement of noise and optical effects. Our current smoothing method, while effective, does not utilize color information, a potential factor for further improvement. This could lead to more accurate and robust depth prediction techniques, thereby contributing to the ongoing advancements in the field.

Lastly, in Chapter 5, we ventured into challenges that have long constrained the traditional hand-drawn animation, specifically targeting the drawing of highly detailed character eyes. We were motivated by the industry's reliance on labor-intensive manual techniques, the resulting compromises in character design complexity and artistic consistency and the struggle in this art form to benefit from recent advances in computer graphics. Our survey of professional animators substantiated these concerns, revealing not only the time-consuming nature of drawing character faces but also the industry's slow adoption of digital tools.

With these challenges in mind, our aim was to create a method that could adapt to various design directions, allowing for dynamic changes in shape, color, and the addition of various features, all without the need for manual work, labeled samples or internal production data. We introduced the concept of context-aware translation, a novel method for unsupervised image-to-image translation based on convolutional networks. Our approach employed two adversarial classification networks and partial convolutions,

and it was further enhanced by a novel triple-reconstruction loss. This loss function contributed to generating a greater amount of unsupervised detail than traditional loss methods, fulfilling one of our key objectives. The model demonstrated its capability to automatically enhance the detailing of anime character eyes based on specific design guidelines. This allowed for dynamic alterations in iris shape, color, and the addition of various features, fulfilling our initial goal of elevating detail and consistency without compromising artistic intent. The versatility of our model negated the need for specialized production data or labeled samples, making it a universally applicable solution across different animation environments.

Beyond its primary functions, our model displayed emergent capabilities, such as precise image segmentation, which offer promising directions for future research. While we identified areas for refinement, especially in post-processing to eliminate artifacts and achieve real-time interactivity, the effectiveness of our approach was corroborated through a large user study and extensive ablation. These confirmed a preference for our style-driven enhancements in non-photo-realistic imagery applications. Although the paper detailing this work is still under conference review, it has already been made publicly available on arXiv under the title *Re:Draw - Automatically Redrawing Anime Eyes* [CBCW24], inviting further academic scrutiny and application.

In summary, this doctoral thesis has highlighted the critical role of optimization in the field of visual computing. Through each chapter, we've seen how applying optimization techniques can significantly improve processes and outcomes across various applications. From enhancing real-time performance to refining depth prediction and automating drawn details, optimization has been a key driver in our research. This work demonstrates that, regardless of the specific domain, optimization methods are essential tools for addressing complex challenges, improving efficiency, and pushing the boundaries of what's possible in computer graphics. As the field continues to grow and change, we hope the optimization strategies explored here will inspire and inform future research and development.

List of Figures

1.1 Optimization Pervasiveness 2
1.2 Problem Overview 3

2.1 Convolution Layer 16
2.2 Content Adaptive Shading 19
2.3 Depth Estimation Pipelines 21
2.4 Inpainting 23
2.5 Colorization 25
2.6 Perceptual Metrics 26
2.7 Light-Field Imagery 27
2.8 Animation Pipeline 30
2.9 Animation Frames 31
2.10 Limited Animation 31

3.1 Chapter Overview 34
3.2 Predictor Architecture 35
3.3 Input Comparison 37
3.4 Transforms Motivation 39
3.5 Metric Distribution 41
3.6 Parameter Spaces 41
3.7 Perceptual Metrics 42
3.8 VRS Pipeline 43
3.9 Viewpoint Capture Pipeline 44
3.10 Extrapolation for VRS 45
3.11 Prediction Results 48
3.12 Modified Scenes 49
3.13 Predictor Generalizability 49
3.14 Adaptive Shading Comparison 49
3.15 Predictor Limitations 51

4.1 Chapter Overview 54
4.2 Minimal Depth Estimation Pipeline 55
4.3 Neural Network Combination 57
4.4 Artifact-Removal 59

4.5 Iterative Smoothing 60
4.6 Refinement Results 63

5.1 **Chapter Overview.** The context-aware translation proposed in this chapter is capable of automatically redrawing parts of images according to any provided design, without the need for fine-tuning. Unlike image-to-image translation [LHM+19], which neglects surrounding context, our approach considers the entire frame. Unlike inpainting [Lah23], which lacks artistic control by ignoring the original content, our method honors the artist's input. This facilitates the production of more consistent artwork and allows for more complex design choices. 65
5.2 Industry Survey 66
5.3 Industry Survey 67
5.4 Color Guide 67
5.5 Redrawing Pipeline 69
5.6 Redrawer Training 71
5.7 Discriminator Architecture 72
5.8 Design Variety 73
5.9 Style-Aware Encoding 73
5.10 Low-Pass Filter 75
5.11 Clustering Validation 77
5.12 Redrawing Results 77
5.13 Ablation of Contributions 78
5.14 Ablation of Losses 79
5.15 Related Work Ablation 79
5.16 Ablation of Diffusion 79
5.17 Redrawing Limitations 80
5.18 Increasing Detail 81
5.19 Redesign Results 82
5.20 Redesign Results 83
5.21 Temporal Coherence 84

List of Tables

3.1 Predictor Parameters 36
3.2 Input Contribution 38
3.3 Ablation of Metric Prediction 47

4.1 Ablation of Refinements 62

5.1 Generated Dataset 74
5.2 Domain Knowledge 85
5.3 Realness Test (T1) 86
5.4 Realness Test (T1) 86
5.5 Level-of-Detail Test (T2) 86
5.6 Preference Test (T3) 87

Bibliography

[AcB22] Anonymous, Danbooru community, and Gwern Branwen. Danbooru2021: A large-scale crowdsourced and tagged anime illustration dataset. `https://www.gwern.net/Danbooru2021`, January 2022. Accessed: DATE.

[ACÖG17] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[AHTAM14] Magnus Andersson, Jon Hasselgren, Robert Toth, and Tomas Akenine-Möiler. Adaptive texture space shading for stochastic rendering. In *Computer Graphics Forum*, volume 33, pages 341–350. Wiley Online Library, 2014.

[AIK18] Olivier Augereau, Motoi Iwata, and Koichi Kise. A survey of comics research in computer science. *J. Imaging*, 4(7):87, 2018.

[Ama17] Amazon Lumberyard. Amazon lumberyard bistro, open research content archive (orca), July 2017.

[AMT20] Kenta Akita, Yuki Morimoto, and Reiji Tsuruno. Deep-Eyes: Fully Automatic Anime Character Colorization with Painting of Details on Empty Pupils. In Alexander Wilkie and Francesco Banterle, editors, *Eurographics 2020 - Short Papers*. The Eurographics Association, 2020.

[ANAM+20] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. Flip: a difference evaluator for alternating images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):1–23, 2020.

[AS17] Yuriy Anisimov and Didier Stricker. Fast and efficient depth map estimation from light fields. In *2017 International Conference on 3D Vision (3DV)*, pages 337–346. IEEE, 2017.

[Bal15] Stephen Balaban. Deep learning and face recognition: the state of the art. *Biometric and surveillance technology for human and activity identification XII*, 9457:68–75, 2015.

[BBM87] Robert C Bolles, H Harlyn Baker, and David H Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55, 1987.

[BFM10] Christopher A Burns, Kayvon Fatahalian, and William R Mark. A lazy object-space shading architecture with decoupled sampling. In *Proceedings of the Conference on High Performance Graphics*, pages 19–28. Citeseer, 2010.

[Bho19] Swaroop Bhonde. Easy vrs integration with eye tracking, 2019. Accessed: 2021-03-05.

[BJK17] Yunsu Bok, Hae-Gon Jeon, and In So Kweon. Geometric calibration of micro-lens-based light field cameras using line features. *IEEE transactions on pattern analysis and machine intelligence*, 39(2):287–300, 2017.

[BRPM16] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, page 3, 2016.

[BSD19] Boichi, Shueisha, and Dr.STONE Production Committee. Dr.stone. `https://dr-stone.jp/`, 2019. [Online; accessed 4-August-2023].

[BYC+20] Nir Benty, Kai-Hwa Yao, Petrik Clarberg, Lucy Chen, Simon Kallweit, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. The Falcor rendering framework, 08 2020.

[Car16] Leonardo Carrion. Battle damaged sci-fi helmet - pbr, 2016.

[CBCW24] Joao Liborio Cardoso, Francesco Banterle, Paolo Cignoni, and Michael Wimmer. Re:draw – context aware translation as a controllable method for artistic production, 2024.

[CD17] Coolkyousinnjya and Dragon Life Improvement Committee. Miss kobayashi's dragon maid, 2017. [Online; accessed 4-August-2023].

[CGW21] João L Cardoso, Nuno Gonçalves, and Michael Wimmer. Cost volume refinement for depth prediction. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 354–361. IEEE, 2021.

[CKY+22] Joao Liborio Cardoso, Bernhard Kerbl, Lei Yang, Yury Uralsky, and Michael Wimmer. Training and predicting visual error for real-time applications. *Proc. ACM Comput. Graph. Interact. Tech.*, 2022.

[CMW+18] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1536–1544, New York, NY, USA, 2018. Association for Computing Machinery.

[Com18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[CTH+14] Petrik Clarberg, Robert Toth, Jon Hasselgren, Jim Nilsson, and Tomas Akenine-Möller. Amfs: adaptive multi-frequency shading for future graphics processors. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[CTM13] Petrik Clarberg, Robert Toth, and Jacob Munkberg. A sort-based deferred shading architecture for decoupled sampling. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[Dan82] George B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1(2):43–48, April 1982.

[Dar18] Darling in the Franxx Production Committee. Darling in the franxx. `https://darli-fra.jp/staffcast/`, 2018. [Online; accessed 4-August-2023].

[Dav88] H. A. David. *The Method of Paired Comparisons, 2nd ed.* Oxford University Press, 1988.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[Dro20] Michal Drobot. Software-based variable rate shading in call of duty: Modern warfare. The 47TH International Conference and Exhibition on Computer Graphics and Interactive Techniques, 2020.

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

[Ede09] Eden of the East Production Committee. Eden of the east. `https://juiz.jp/`, 2009. [Online; accessed 4-August-2023].

[Epi17] Epic Games. Unreal engine sun temple, open research content archive (orca), October 2017.

[FFM+21] Linus Franke, Laura Fink, Jana Martschinke, Kai Selgrad, and Marc Stamminger. Time-warped foveated rendering for virtual reality headsets. In *Computer Graphics Forum*, volume 40, pages 110–123. Wiley Online Library, 2021.

[Fur16] Furansujin Connection. Les étapes de fabrication. `https://web.archive.org/web/20220401075414/http://www.furansujinconnection.com/les-etapes-de-fabrication/`, 2016. [Online; accessed 2022-March-21].

[GEB16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2414–2423. IEEE Computer Society, 2016.

[Guo21] Shuai Guo. An introduction to surrogate optimization: intuition, illustration, case study, and the code. `https://towardsdatascience.com/5d9364aed51b`, 2021. Online; accessed: 4-August-2023.

[GYLG13] Todor Georgiev, Zhan Yu, Andrew Lumsdaine, and Sergio Goma. Lytro camera technology: theory, algorithms, performance analysis. In *Multimedia Content and Mobile Devices*, volume 8667, page 86671J. International Society for Optics and Photonics, 2013.

[HB17] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1510–1519. IEEE Computer Society, 2017.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[HIG02] Heiko Hirschmüller, Peter R Innocent, and Jon Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1-3):229–246, 2002.

[Hin12] Geoffrey Hinton. Neural networks for machine learning, lecture 6a, overview of mini-batch gradient descent. Coursera, 2012. Available online: [URL].

[HJKG16] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*, pages 19–34. Springer, 2016.

[HLBK18] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[HY16] Karl E Hillesland and JC Yang. Texel shading. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: Short Papers*, pages 73–76, 2016.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[IKT23] Masakazu Ishiguro, Kodansha, and Tengoku Daimakyo Production Committee. Heavenly delusion. https://tdm-anime.com/staffcast/, 2023. [Online; accessed 4-August-2023].

[Int19] Intel Corp. Intel processor graphics gen11 architecture. Technical report, 03 2019.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5967–5976. IEEE Computer Society, 2017.

[JPC+15] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1547–1555, 2015.

[JPC+19] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Depth from a light field image with learning-based matching costs. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):297–310, 2019.

[JWMM21] Akshay Jindal, Krzysztof Wolski, Karol Myszkowski, and Rafał K Mantiuk. Perceptual model for adaptive local shading and refresh rate. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[JZL+17] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *CoRR*, abs/1708.05509, 2017.

[KAL+21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 852–863, 2021.

[KGB95] William N Klarquist, Wilson S Geisler, and Alan C Bovik. Maximum-likelihood depth-from-defocus for active vision. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 374–379. IEEE, 1995.

[KJPY19] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *2019*

IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 9055–9064. IEEE, 2019.

[KK07] Key and Kyoto Animation. Clannad. `https://www.tbs.co.jp/clannad/clannad1/02staffcast/staffcast.html`, 2007. [Online; accessed 4-August-2023].

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4401–4410. Computer Vision Foundation / IEEE, 2019.

[KLA+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8107–8116. Computer Vision Foundation / IEEE, 2020.

[KLA21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4217–4228, 2021.

[KOK14] Min-Jung Kim, Tae-Hyun Oh, and In So Kweon. Cost-aware depth map estimation for lytro camera. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 36–40. IEEE, 2014.

[Kra18] Manuel Kraemer. Accelerating your vr games with vrworks. In *NVIDIAs GPU Technology Conference (GTC)*, 2018.

[KSP11] Shoji Kawamori, SATELIGHT, and Project AQUARION Evol. Aquarion evol. `http://aqevol.com/staff_cast/index.php`, 2011. [Online; accessed 4-August-2023].

[KZ02] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *European conference on computer vision*, pages 82–96. Springer, 2002.

[Lah23] LahIntheFutureland. Mysteriousv4 sdxl 1.0. `https://civitai.com/models/118441/lah-mysterious-or-sdxl`, 2023. [Online; accessed 19-October-2023].

[LAK18] Siyeong Lee, Gwon Hwan An, and Suk-Ju Kang. Deep recursive hdri: Inverse tone mapping using generative adversarial networks. In *proceedings of the European Conference on Computer Vision (ECCV)*, pages 596–611, 2018.

[LBD+89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[LCBKY15] Haiting Lin, Can Chen, Sing Bing Kang, and Jingyi Yu. Depth recovery from light field using focal stack symmetry. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3451–3459, 2015.

[LD12] Gábor Liktor and Carsten Dachsbacher. Decoupled deferred shading for hardware rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 143–150, 2012.

[LHM+19] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10551–10560, 2019.

[Liu17] Edward Liu. Lens matched shading and unreal engine 4 integration, 2017. Accessed: 2021-03-05.

[LKY+19] Gayoung Lee, Dohyun Kim, Youngjoon Yoo, Dongyoon Han, Jung-Woo Ha, and Jaehyuk Chang. Unpaired sketch-to-line translation via synthesis of sketches. In *SIGGRAPH Asia 2019 Technical Briefs*, SA '19, page 45–48, New York, NY, USA, 2019. Association for Computing Machinery.

[LLC+10] Jianguo Li, Eric Li, Yurong Chen, Lin Xu, and Yimin Zhang. Bundled depth-map merging for multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2769–2776. IEEE, 2010.

[LLW17] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Trans. Graph.*, 36(4), jul 2017.

[LLWL20] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5549–5558, 2020.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LRS+18] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 85–100, 2018.

[LTH+18] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51, 2018.

[McG17] Morgan McGuire. Computer graphics archive, July 2017.

[MGN18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[MKS+21] Akinobu Maejima, Hiroyuki Kubo, Seitaro Shinagawa, Takuya Funatomi, Tatsuo Yotsukura, Satoshi Nakamura, and Yasuhiro Mukaigawa. *Anime Character Colorization Using Few-Shot Learning*. Association for Computing Machinery, New York, NY, USA, 2021.

[MNV+21] Joerg H Mueller, Thomas Neff, Philip Voglreiter, Markus Steinberger, and Dieter Schmalstieg. Temporally adaptive shading reuse for real-time rendering and virtual reality. *ACM Transactions on Graphics (TOG)*, 40(2):1–14, 2021.

[MSR+19] Hiromichi Masuda, Tadashi Sudo, Kazuo Rikukawa, Yuji Mori, Naofumi Ito, Yasuo Kameyama, and Megumi Onouchi. Anime Industry Report. Technical report, The Association of Japanese Animations, 2019.

[MTLP96] JP Mellor, Seth Teller, and Tomás Lozano-Pérez. Dense depth maps from epipolar images. 1996.

[NHB17] Kate Anderson Nicholas Hull and Nir Benty. Nvidia emerald square, open research content archive (orca), July 2017.

[NKR16] Tappei Nagatsuki, KADOKAWA, and Re: Life in a Different World Starting from Zero 1 Production Committee. Re: Life in a different world starting from zero. `http://re-zero-anime.jp/tv/`, 2016. [Online; accessed 4-August-2023].

[NLB+05] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, Pat Hanrahan, et al. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11):1–11, 2005.

[NNJ+20] K Nazeri, E Ng, T Joseph, FZ Qureshi, and M Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. arxiv 2019. *arXiv preprint arXiv:1901.00212*, 2020.

[NRS22] Oron Nir, Gal Rapoport, and Ariel Shamir. CAST: Character labeling in Animation using Self-supervision by Tracking. *Computer Graphics Forum*, 2022.

[NVI18] NVIDIA Corp. Nvidia turing gpu architecture. Technical report, 09 2018.

[OLL+21] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2021.

[OMH15] ONE, Yusuke Murata, and Hero Association Headquarters. One punch man. `https://onepunchman-anime.net/staff/`, 2015. [Online; accessed 4-August-2023].

[OT97] Eiichiro Oda and Toei Animation. One Piece. `https://one-piece.com/anime/`, 1997. [Online; accessed 4-August-2023].

[OT21] OtakuVS and Tonari Animation. Otachan! `https://www.youtube.com/watch?v=OWRsQT0pI-U`, 2021. [Online; accessed 4-August-2023].

[OT22] Monyreak Oum and Team RWBY Project. RWBY: Ice Queendom. `https://anime.team-rwby-project.jp/staff/`, 2022. [Online; accessed 4-August-2023].

[PEZZ20] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 319–345. Springer, 2020.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, pages 313–318. 2003.

[PLWZ19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[PSK+16] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[PW12] Christian Perwass and Lennart Wietzke. Single lens 3d-camera with extended depth-of-field. In *Human Vision and Electronic Imaging XVII*, volume 8291, page 829108. International Society for Optics and Photonics, 2012.

[RAGS01] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[RE16] Martin Rerabek and Touradj Ebrahimi. New light field image dataset. Technical report, 2016.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA, 2015. MIT Press.

[RKLC+11] Jonathan Ragan-Kelley, Jaakko Lehtinen, Jiawen Chen, Michael Doggett, and Frédo Durand. Decoupled sampling for graphics pipelines. *ACM Transactions on Graphics (TOG)*, 30(3):1–17, 2011.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[SC88] Sidney Siegel and N. John Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGrall-Hill International, 1988.

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[SdCJM19] Felipe Coelho Silva, Paulo André Lima de Castro, Hélio Ricardo Júnior, and Ernesto Cordeiro Marujo. Mangan: Assisting colorization of manga characters concept art using conditional GAN. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 3257–3261. IEEE, 2019.

[SFO+21] Yugo Shimizu, Ryosuke Furuta, Delong Ouyang, Yukinobu Taniguchi, Ryota Hinami, and Shonosuke Ishiwatari. Painting style-aware manga colorization based on generative adversarial networks. In *2021 IEEE International Conference on Image Processing, ICIP 2021, Anchorage, AK, USA, September 19-22, 2021*, pages 1739–1743. IEEE, 2021.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[Sho09] Jeremy Shopf. Mixed resolution rendering. In *Game Developers Conference*, 2009.

[SJY+18] Changha Shin, Hae-Gon Jeon, Youngjin Yoon, In So Kweon, and Seon Joo Kim. Epinet: A fully-convolutional neural network using epipolar geometry for depth from light field images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4748–4757, 2018.

[SKA18] Makoto Sanada, Kadokawa, and Angels of Death Production Committee. Angels of death. http://www.jcstaff.co.jp/sakuhin/nenpyo/2018/06_satsuriku/satsuriku.htm, 2018. [Online; accessed 4-August-2023].

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[SNL+21] Hao Su, Jianwei Niu, Xuefeng Liu, Qingfeng Li, Jiahe Cui, and Ji Wan. Mangagan: Unpaired photo-to-manga translation based on the methodology of manga drawing. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 2611–2619. AAAI Press, 2021.

[SRK17] Matan Sela, Elad Richardson, and Ron Kimmel. Unrestricted facial geometry reconstruction using image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1576–1585, 2017.

[SSII18a] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: Adversarial augmentation for structured prediction. *ACM Trans. Graph.*, 37(1), jan 2018.

[SSII18b] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Real-time data-driven interactive rough sketch inking. *ACM Trans. Graph.*, 37(4), jul 2018.

[SSISI16] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph.*, 35(4), jul 2016.

[SSL20] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. COCO-FUNIT: few-shot unsupervised image translation with a content conditioned style encoder. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12348 of *Lecture Notes in Computer Science*, pages 382–398. Springer, 2020.

[SYT97] Murali Subbarao, Ta Yuan, and JennKwei Tyan. Integration of defocus and focus analysis with stereo for 3d shape recovery. In *Three-Dimensional Imaging and Laser-Based Systems for Metrology and Inspection III*, volume 3204, pages 11–24. International Society for Optics and Photonics, 1997.

[SZ15] Gerard Sierksma and Yori Zwols. *Linear and Integer Optimization: Theory and Practice*. CRC Press, 3 edition, 2015.

[TAKW+19] Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. Luminance-contrast-aware foveated rendering. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[TG15] Ee-Leng Tan and Woon-Seng Gan. Computational models for just-noticeable differences. In *Perceptual Image Coding with Discrete Cosine Transform*, pages 3–19. Springer, 2015.

[THMR13] Michael W Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 673–680, 2013.

[TSM+15] Michael W Tao, Pratul P Srinivasan, Jitendra Malik, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Depth from shading, defocus, and correspondence using light-field angular coherence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1940–1948, 2015.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[VRA+07] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. In *ACM transactions on graphics (TOG)*, volume 26, page 69. ACM, 2007.

[WGY+18] Krzysztof Wolski, Daniele Giunchi, Nanyang Ye, Piotr Didyk, Karol Myszkowski, Radosław Mantiuk, Hans-Peter Seidel, Anthony Steed, and Rafał K Mantiuk. Dataset and metrics for predicting local visible differences. *ACM Transactions on Graphics (TOG)*, 37(5):1–14, 2018.

[WWL18] In Kyu W. Williem, Park and Kyoung Mu Lee. Robust light field depth estimation using occlusion-noise aware data costs. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2484–2497, 2018.

[WY20] Xinrui Wang and Jinze Yu. Learning to cartoonize using white-box cartoon representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[YGL+13] Zhan Yu, Xinqing Guo, Haibing Lin, Andrew Lumsdaine, and Jingyi Yu. Line assisted light field triangulation and stereo matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2792–2799, 2013.

[YMT17] Kore Yamazaki, Mag Garden, and The Ancient Magus' Bride Production Committee. The ancient magus bride. https://mahoyome.jp/staffcast/#season1, 2017. [Online; accessed 4-August-2023].

[YS07] Hajime Yatate and Sunrise Bandai Visual. Idolmaster: Xenoglossia. https://www.sunrise-inc.co.jp/idolmaster/, 2007. [Online; accessed 4-August-2023].

[YSL08] Lei Yang, Pedro V Sander, and Jason Lawrence. Geometry-aware framebuffer level of detail. In *Computer Graphics Forum*, volume 27, pages 1183–1188. Wiley Online Library, 2008.

[YTA+20] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7508–7517, 2020.

[YYDN07] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-depth super resolution for range images. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[YYP19] Sheng You, Ning You, and Minxue Pan. PI-REC: progressive image reconstruction network with edge and color domain. *CoRR*, abs/1903.10146, 2019.

[YZ19] Lei Yang and Dmitry Zhdan. Adaptive shading overview, 2019.

[YZK+19] Lei Yang, Dmitry Zhdan, Emmett Kilgariff, Eric B Lum, Yubo Zhang, Matthew Johnson, and Henrik Rydgård. Visually lossless content and motion adaptive shading in games. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(1):1–19, 2019.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[ZJL20] Lvmin Zhang, Yi JI, and Chunping Liu. Danbooregion: An illustration region dataset. In *European Conference on Computer Vision (ECCV)*, 2020.

[ZLS+21] Lvmin Zhang, Chengze Li, Edgar Simo-Serra, Yi Ji, Tien-Tsin Wong, and Chunping Liu. User-guided line art flat filling with split filling mechanism. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9889–9898. Computer Vision Foundation / IEEE, 2021.

[ZLW+18] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM Trans. Graph.*, 37(6), dec 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251. IEEE Computer Society, 2017.

[ZZL+19] Wenhui Zhou, Enci Zhou, Gaomin Liu, Lili Lin, and Andrew Lumsdaine. Unsupervised monocular depth estimation from light field image. *IEEE Transactions on Image Processing*, 29:1606–1617, 2019.