COMPUTATIONAL AESTHETICS USING MACHINE LEARNING FOR VIDEO GAME CAMERA DIRECTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ NACİ ERDEM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MODELING AND SIMULATION

AUGUST 2015

Approval of the thesis:

COMPUTATIONAL AESTHETICS USING MACHINE LEARNING FOR VIDEO GAME CAMERA DIRECTION

submitted by **ALİ NACİ ERDEM** in partial fulfillment of the requirements for the degree of **Master of Science in Game Technologies Department, Middle East Technical University** by,

Prof. Dr. Nazife Baykal _____________
Director, **Informatics Institute, METU**

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu _____________
Head of Department, **Modeling and Simulation, METU**

Prof. Dr. Uğur Halıcı _____________
Supervisor, **Electrical and Electronics Engineering, METU**

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar _____________
Electrical and Electronics Engineering, METU

Prof. Dr. Uğur Halıcı _____________
Electrical and Electronics Engineering, METU

Assoc. Prof. Dr. Ahmet Oğuz Akyüz _____________
Computer Engineering, METU

Assoc. Prof. Dr. Hüseyin Hacıhabiboğlu _____________
Modeling and Simulation, METU

Assist. Prof. Dr. Murat Yılmaz _____________
Computer Engineering, Çankaya University

Date: **27.08.2015**

ABSTRACT

COMPUTATIONAL AESTHETICS USING MACHINE LEARNING FOR VIDEO GAME CAMERA DIRECTION

Erdem, Ali Naci

M.S., Department of Game Technologies

Supervisor: Prof. Dr. Uğur Halıcı

August 2015, 109 pages

Computational aesthetics is a developing field which employs computational approaches either for generating or evaluating aesthetic values. In the scope of this thesis, visual aesthetic quality of computer generated images was aimed to be improved using a computational aesthetics approach. An appropriate machine learning algorithm was selected and trained on a set of reference images collected online. Using the trained model, a novel video game camera direction method predicting the aesthetic quality of the real-time graphics and changing the virtual camera position accordingly was developed. In order for the proposed approach to be effective, a regression analysis assigning aesthetic quality values to images was utilized instead of high and low quality classification. Rather than dealing with semantic context, color distribution and compositional properties affecting aesthetic appeal were preferred and to make quicker aesthetic score predictions, faster and more efficient features were selected, considering their aesthetic foundations. Some of the existing features were improved, and some were tailored to be applied to regression analysis. Aesthetics being a highly subjective topic, only outdoor scene and landscape visuals were targeted in this work in order to reduce complexity. The proposed method on the other hand, can be extended to other environments by changing the training data. The prediction performance of the machine learning model was not very significant when compared to the previous works, yet promising considering the challenges and limitations involved and showed that a near-real time aesthetic analysis and visual improvement was possible through a “virtual” camera director.

Keywords: Computational Aesthetics, Machine Learning, Computer Graphics, Virtual Camera

ÖZ

VİDEO OYUNU KAMERA YÖNETİMİ İÇİN MAKİNE ÖGRENMESİ İLE HESAPLAMALI ESTETİK

Erdem, Ali Naci

Yüksek Lisans, Oyun Teknolojileri Bölümü

Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Ağustos 2015, 109 sayfa

Hesaplamalı estetik, hesaplamalı yaklaşımları, estetik değer üretimi ve değerlendirmesi için kullanan gelişmekte olan bir konudur. Bu tez kapsamında, bilgisayar tarafından oluşturulan görüntülerin görsel estetik kalitesinin hesaplamalı estetik kullanılarak iyileştirilmesi hedeflenmektedir. Uygun bir makine öğrenmesi algoritması seçilmiş ve çevrim içi olarak toplanmış bir takım görüntüler üzerinde eğitilmiştir. Eğitilen model kullanılarak gerçek zamanlı grafiklerin estetik kalitesi öngörülerek buna bağlı sanal kamera konumunun değiştirildiği yenilikçi bir kamera yönetim yöntemi geliştirilmiştir. Önerilen bu yöntemin etkin olabilmesi için yüksek ve düşük kalite olarak sınıflandırma yerine görüntülere estetik kalite değerleri veren bir regresyon analizi uygulanmıştır. Anlamsal bağlam ile ilgilenmek yerine estetik çekiciliği etkileyen renk dağılımları ve kompozisyon özellikleri kullanılmış ve daha çabuk estetik puan öngörüsü yapılabilmesi için estetik dayanaklarını da gözeterek daha hızlı ve verimli öznitelikler seçilmiştir. Mevcut bazı öznitelikler iyileştirilmiş ve regresyon analizine uygulamak üzere uyarlanmıştır. Estetiğin ileri seviyede öznel bir konu olması sebebiyle karmaşıklığı azaltmak adına bu çalışmada sadece dış mekan sahneleri ve manzara görselleri hedeflenmiştir. Öte yandan önerilen yöntem eğitim verisinin değiştirilmesi ile farklı ortamlar için de genişletilebilir. Makine öğrenmesi modelinin öngörü performansı önceki çalışmalar ile kıyaslandığında çok dikkat çekici olmamakla beraber, konunun ihtiva ettiği zorluklar ve kısıtlamalar dikkate alındığında yine de ümit verici bir şekilde “sanal” bir kamera yönetmeni vasıtası ile neredeyse gerçek zamanlı estetik analiz ve görsel iyileştirme yapılmasının mümkün olduğunu göstermiştir.

Anahtar Kelimeler: Hesaplamalı Estetik, Makine Öğrenmesi, Bilgisayar Grafiği, Sanal Kamera

To My Family

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Prof. Dr. Uğur Halıcı for her guidance and supervision throughout the thesis preparation and for suggesting the original research idea.

I would like to thank to Prof. Dr. Gözde Bozdağı Akar, Assoc. Prof. Dr. Ahmet Oğuz Akyüz, Assoc. Prof. Dr. Hüseyin Hacihabiboglu, Prof. Dr. Uğur Halıcı and Assist. Prof. Dr. Murat Yılmaz for being on my thesis committee.

I would also like to thank Oğuz Can Aydın, Ayda Çiğlez, Özgür Hamat, Hazal Suna, Caner Kahvecioğlu, and Kayra Kurt for their support during the thesis writing period.

Finally, I owe a debt of gratitude to my family and Sıla Erdem for their support, endless patience, encouragements and especially for their love.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
GPU	Graphics Processing Unit
ML	Machine Learning
SVM	Support Vector Machine
GLCM	Gray Level Co-occurrence Matrix
DoF	Depth of Field
RANSAC	Random Sample Consensus
ANN	Artificial Neural Network
HSV	Hue Saturation Value
HSL	Hue Saturation Lightness
RBF	Radial Basis Function
MSE	Mean Squared Error
RGB	Red Green Blue
FFT	Fast Fourier Transform
PCA	Principal Component Analysis
ADAboost	Adaptive Boosting
CUHK	The Chinese University of Hong Kong
RMS	Root Mean Square
EMD	Earth Mover's Distance
SIFT	Scale Invariant Feature Transform
SMS	Slope of the Magnitude Spectrum
KNN	K Nearest Neighbors
CV	Cross-Validation
AVA	Aesthetic Visual Analysis
CART	Classification and Regression Tree
Bagging	Bootstrap aggregation
OOB	out-of-bag
RF	Random Forests
HDR	High Dynamic Range

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The video game (or interactive entertainment) industry is growing at an ever increasing pace and already competing with other entertainment mediums such as film and music. At this high throughput, fulfilling the needs of the end-users is becoming harder and harder and as the industry pushes the limits of current technology, criteria other than gameplay and realistic graphics start to stand out to a greater extent. Eventually, video games incorporating various art forms are prone to aesthetic requirements and this is a well-known and studied subject. On the other hand, the man-hours needed to attain a high aesthetic quality are costly, it requires lots of manual labor, and still does not completely resolve the issue when the user is controlling the protagonist in real-time. In order to improve visual aesthetics of video games, novel methods need to be developed and applied to the field. Although mainly considered for multimedia communication services, the term "quality of experience" was defined as "the degree of delight or annoyance of the user of an application or service" [1] in the Qualinet white paper. This notion can also be extended to the interactive entertainment area and this work approaches the problem from a visual aesthetics viewpoint as a step towards improving the quality of experience in video games.

The word aesthetics originates from the Greek word "aisthanesthai" roughly translating into "perceiving" and "sensing" [2]. It is widely agreed upon as a division of philosophy that deals with beauty and taste [3], considering questions like how art is interpreted, what constitutes art and what is beautiful or ugly [4]. Even though it is tempting to classify aesthetics as a scientific study, it is a highly subjective matter. It is influenced by many factors such as culture, experience, personal preferences [5] and can even change for a given person over time and depending on the circumstances [4]. Understanding aesthetic value involves meanings that are meant and perceived, the purpose of the artist, genre and experience of the observer [6]. On the other hand, it is also claimed that there exists a global aesthetic understanding and current technology is not sufficient yet to analyze it completely [7]. Whether or not there exists an absolute aesthetic judgement is a highly controversial topic and the truth is yet to be discovered. Although the term is applicable to any art form, in the context of this thesis work the word “aesthetics” is used instead of “visual aesthetics” since the main concern here is aesthetics related to visual stimuli, more specifically photographs and scenes in video games. Aesthetics exists as a concept on its own but does not come to light and have a real meaning without criticism. In their book “Algorithmic Aesthetics”, Stiny et al. suggested that aesthetics cares about how art forms can be “described”, “interpreted” and “evaluated” and defined criticism as any attempt to describe, interpret and evaluate an art piece [8].

Although not completely independent from each other, there are roughly two main parts to be evaluated when considering aesthetics; content and visual quality [9]. Content of an image is in close relationship to semantics and meaning. On the other hand, it is not always possible to infer these higher level properties. This is even true for humans, for example the artist may be aiming to induce a specific feeling/emotion to the viewer through some semantic context and metaphor but the viewer can easily interpret the scene in a different way. It is sometimes referred as transparency where the scene in consideration is interpreted as-is, like it has a real equivalent meaning [8]. Sometimes transparency is intended by the artist on purpose and sometimes the viewer seeks a real meaning, preventing the intended alternative emotion to be transferred. On the other hand, when visual quality is of concern, there are widely accepted principal techniques and “rule of thumbs” that are known to improve the aesthetic quality of images [10]. These are not always applicable to every style of photography [5] but can be beneficial when considering a general understanding of aesthetics. Commonly accepted criteria for visual aesthetics are; composition, lighting/exposure/contrast, color usage and distribution, and simplicity/subject emphasis [11], [10], [6].

Hoenig defines computational aesthetics as “the research of computational methods that can make applicable aesthetic decisions in a similar fashion as humans can.” [3] It is a multidisciplinary field that spans computer science, cognitive sciences, neuroscience and art. It deals with the computational interpretation of aesthetics, trying to determine or generate aesthetic value using computer technology. Computational aesthetics was first originated with George David Birkhoff, who built the basics in his book “Aesthetic Measure” in 1933 [3], [8]. He developed the formula $M = O/C$ (where, M is the aesthetic measure, O is order and C is complexity) and applied it to various aesthetic classes of objects.

As arts integrate into the everyday living with the increase in computer-aided design tools, high numbers of daily photographs taken by mobile phones, excessive sharing of these on social media without explicit attention to aesthetics led to an “aesthetic pollution” [3]. Due to the limited labor and the time-consuming nature of generating high aesthetic value designs, trying to understand and explain aesthetics through software and computational methods without the need for human intervention became a necessity. This need to evaluate and possibly generate aesthetic value makes up the main driving force of computational aesthetics [3]. Computational aesthetics uses various methods including machine learning to achieve this goal and there is still much progress to be made.

Search engines, image retrieval systems, digital cameras and computer image editing software all contributed to the improvements on the field [12], [13]. Video games on the other hand were lightly considered in conjunction with computational aesthetics, especially visually and constitute a field of application awaiting further interest. The main objective of this work is to apply current state of computational aesthetics - through machine learning and image processing methods- to video games on the graphics that are rendered real-time on a graphics processing unit (GPU). A near-real time aesthetic camera direction method to be used in a third person camera setup is aimed, to improve the perceived aesthetic quality of rendered computer graphics by further inducing emotion by a virtual "camera director". The main emphasis is on the run-times of available methods due to the real-time constraints of the final application. Therefore, rather than dealing with face detection and semantic context, color distribution and composition cues affecting aesthetic appeal (corresponding to visual quality, not content) are preferred although aesthetics and semantics are closely related to each other [6]. In the literature, there are also attempts to capture this semantic information and consider aesthetics accordingly [11], which was left out of the scope of this work. To be able to apply the final system on a real-time rendered environment effectively (for better camera placement analysis) a regression analysis is required rather than classification of high and low aesthetic quality images.

This work covers computer graphics and computer vision as well as aesthetic interpretation of imagery which is considered a result of the cognitive processes in the human mind. Aesthetics is a highly subjective topic, making a computational approach even more difficult. Datta et al. [14] described this problem by defining the aesthetic gap as;

> *The aesthetics gap is the lack of coincidence between the information that one can extract from low-level visual data (i.e., pixels in digital images) and the interpretation of emotions that the visual data may arouse in a particular user in a given situation.*

To further narrow down the application area, considering its applicability to a greater extend of video games; only outdoor scenes/landscapes were targeted in this work. Application of computational aesthetics to a single category of images (in this case landscape) is known to perform better [10] and using a simpler aesthetic model targeting a single class of photography is intended. A novel aesthetics and optimization oriented feature selection method is applied together with some further improvements.

The thesis is divided into five chapters including this introduction. Currently used computational aesthetics methods dealing with visual aesthetics and frequently used aesthetic image features are discussed in chapter 2 together with some preliminary on virtual camera direction and examples of computational aesthetics applied to video games. These are followed by the proposed approach to successfully incorporate some of these methods and features to video game camera direction, in chapter 3. In chapter 4, numerical results concerning the aesthetic prediction quality of the system are given as well as the development process details. And in the final chapter, conclusions are made with probable future research directions.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, some background information on visual aesthetics will be presented and currently used computational aesthetics practices applied to various forms of visual media will be investigated. These include methods (machine learning and rule based approaches), features, and pre-processing steps used to apply computational aesthetics for value evaluation and improvement/generation. Furthermore, the principle foundations behind machine learning algorithms used to achieve the objective of the thesis will be explained as stated in the previous studies. The features and methods applied in this thesis work will be emphasized in section 2.4 and performance criteria used to measure the performance of the model will be presented in section 2.5. Other features that are not included in this work will only be summarized. Finally various methods used in video game camera direction will be introduced shortly, followed by the use cases of computational aesthetics in video games, although they fundamentally differ from the one presented in this work.

2.1 Aesthetic Primitives

Aesthetical foundations of the features and methods used in section 2.2 are presented in this section to give a general understanding of visual aesthetics. There are some commonly known and accepted building blocks of visual aesthetics and these are summarized here, establishing some background information for better understanding of the features described in the following sections. Grouping of features and aesthetic properties heavily depend on the context of the research and vary widely among existing works. There are mainly two main classes of features, namely global features and local features. Global features are low-level features calculated on the entire image without considering local regions with the exception of some global features that may encode spatial information. Local features on the other hand, are the features calculated specifically for (or in comparison to) a given region of the image found by face detection [15], image segmentation [12], saliency analysis [10] etc. and aims at extracting higher level properties of the images. Features corresponding to the properties of these regions (such as center of mass, area, etc.) can also be grouped under local features. A more aesthetic-oriented classification of features mainly involves composition, lighting/exposure/contrast, color distribution/saturation, texture, sharpness, and simplicity/subject distinction. These groups relate to different aspects of visual aesthetics and they are intertwined in some cases. Yet they are generally well defined on their own to categorize the features used in the previous works appropriately.

2.1.1 Color

Colors used in a picture relates to the aesthetic appeal of photography. One of the important aspects of color usage is often called "color harmony". There are some sets of colors, "harmonic colors", that are known to improve visual aesthetic quality when used together [16]. Colors in an image can be represented by means of a color wheel as shown in Figure 1. The wheel represents the "hue" of a color in a circularly repeating manner and was a result of early theory of color harmony [16]. The main effect of colors on image aesthetic quality arises from the distribution of colors on this circle and relative positions of hues used.

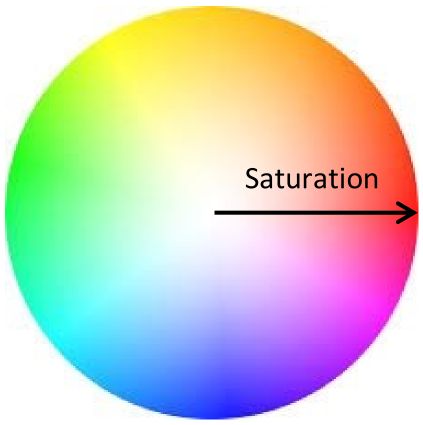

Figure 1: Color wheel

Although there are cultural differences on the perception of color, some color combinations have a global effect on aesthetic perception [17]. One specific rule for improving color distribution quality of an image is employing complementary colors [6] that coincide to opposite ends in the color wheel (see Figure 2). These color pairs reinforce each other [18]. Another important aspect of color is saturation in photography and color psychology and it determines the purity of a given color [12]. Colors relate to the wavelength of the visible electromagnetic spectrum and saturation reaches 100% when only a single wavelength of light is present. Fully saturated colors appear more vivid. By adding complementary hues, saturation decreases. In Figure 1, at the center, there is no saturation and at the peripheral the colors are fully saturated. There are also various color schemes that can be represented on the hue wheel other than the complementary combination as summarized in Figure 2 where the black circle represents the hue wheel [19]. When there is only a single hue with different saturation and lightness values, it is called a monochromatic color scheme. In a split complementary scheme, one of the complementary colors is used as a pair of hues close to each other. These neighboring colors on the wheel are also called analogous colors. In triadic and four-way split configurations, hues corresponding to the vertices of an equilateral triangle and square on the hue wheel are utilized. These color schemes can be rotated on the wheel and still have the same aesthetic effect.

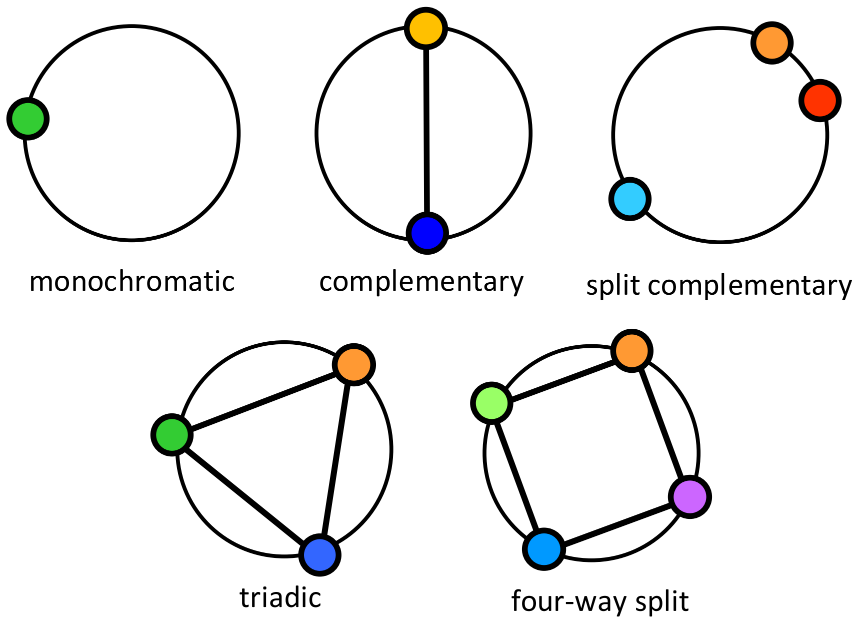

Figure 2: Color schemes

Professional photographers take pictures at specific times during the day or use filters to adjust the color in an image and simplicity of an image is related to the number of colors used, a lower count being preferable [20], [18].

2.1.2 Lighting and Exposure

Brightness of an image is directly related to lighting conditions and exposure settings. An over-exposed image will be rendered brighter whereas an under-exposed image will be darker [12]. Additionally the brightest point and the darkest point in an image represent its dynamic range and add up to its aesthetic value. Representing brightness values on a greater range and representing darker and lighter regions at the same time is important on controlling the contrast in a pleasant image [18]. Lighting of a scene also improves the three dimensional perception of objects when well executed [17].

2.1.3 Composition

Composition is related to the placement of various objects and subjects in the photographic frame considering visual balance. Visual balance states that some arrangements of objects in the frame are more aesthetically pleasing than others [21]. There are various ways of achieving better quality in terms of composition by applying simple rules such as the rule of thirds [6]. Local features discussed above are generally used in conjunction with compositional properties of the images since they are intended to detect and evaluate various regions of the images under consideration.

Rule of thirds is a photographic principle that is used to improve aesthetic quality and suggests that placing important subjects at the intersection of lines that divide the frame into nine equal parts by equally spaced vertical and horizontal lines have a better aesthetic appeal [12]. An example of usage of the rule of thirds is given in Figure 3. Here, the sunflower was positioned at the intersection of these lines. These intersection points are called “power points” [22]. Another similar composition principle is the use of golden ratio $\varphi = (1 + \sqrt{5})/2 = 1.618$. In this case, the photographic frame is partitioned vertically and horizontally such that each dividing line’s position separates the frame into two sections with sides x and y subject to $x/y = y/(x + y) = 1/\varphi$ and important subjects are preferably positioned at these locations [21]. The rule of thirds in reality, is a simplified version of this technique [12]. Furthermore, the placement of the boundary between sky and ground elements (horizon) is also important when analyzing visual balance [21].

Long lines appearing in the image also add to the composition of the photographs, and hold semantic meanings such as horizon and sea surface [23]. Perception of lines in the visual system can be caused by contrast gradients and color changes in the image and their orientations has an effect on the aesthetic perception especially when there are a few dominant lines with specific directions [18]. The relative positions of horizontal and vertical lines can also influence aesthetics and prominent lines in the direction of the main diagonals of an image can have a strong effect, which is sometimes referred as “diagonal dominance” [24]. Furthermore lines in the images can be used to direct the attention of the viewer.

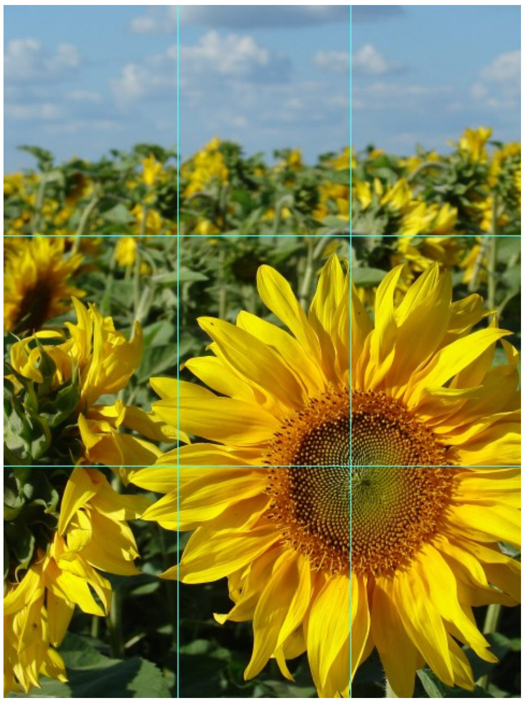

Figure 3: Example rule of thirds usage

2.1.4 Figure-Ground Separation

Not being completely isolated from compositional properties, figure-ground separation is related to the distinction of the main figure from the remaining part of the image (ground) [20]. Keeping the attention of the viewer on the main subject is a well-known photographic practice and can be satisfied by a good usage of depth of field (DoF) [10] or using complementary colors (see section 2.1.1) for subject and background [6]. Depth of field is the distance range that a lens can clearly focus. The objects that are away from the focus position appear blurred when wider apertures are used (a lower depth of field). This property is generally used to make the main subject sharp while keeping the background blurred to fulfill figure separation [18]. The effect of depth of field, although not very prominent, can also be observed in Figure 3. To keep distractions to a minimum, generally images with a clear subject distinction have simpler backgrounds [25].

Similar to the use of complementary colors (hue contrast) for the subject and background regions, another method used frequently utilizes brightness contrast among them [20]. There is evidence that contrast among regions is effective on visual cognition. For human visual system, it is difficult to separate objects without hue contrast and it is difficult to determine locations of objects without brightness contrast [26]. Furthermore the spatial distribution of edges in an image is also important when determining aesthetic quality and edges are generally concentrated near the center of a good quality image [20]. In any case, a distinction/contrast between the subject and background areas is wanted for higher aesthetic quality [17] and many different objects and details in the frame distract the viewer [18].

2.1.5 Texture

Texture can say a few things about the photograph such as the graininess of the used film/medium and smoothness of the image [12]. It also holds the structural properties of surfaces. Patterns, repetition and rhythm support the aesthetic understanding of an image in a positive manner unless it is completely monotonous [18].

2.1.6 Sharpness and Clarity

Photographs with a high sharpness (except the use of blur for other purposes as in DoF) and without an overall blur, are accepted of higher quality [20]. Missing high frequency content in an image is an indication of low sharpness [18]. In landscape photography, an overall sharp and crisp image including the foreground and background regions is generally preferred [27].

2.2 Computing Aesthetics

In this section, features and methods used in the literature to evaluate visual aesthetics are introduced. There are a variety of computational aesthetics approaches used in the literature covering computer vision techniques, machine learning methods and optimization procedures. In most of them modeling and exploitation of the above introduced aesthetic primitives were the main concern and machine learning was the tool of the trade for a large portion of these works. Many of the works dealt with the classification problem by separating the images into high and low quality classes and made predictions for newly introduced ones. A few of the existing works incorporate regression analysis and ranking ([28], [10], [15], [29], [30], [31], [32]) and some of these were either meant to show other findings such as method comparison or presented as proof-of-concept.

In some of the existing works rather than using machine learning techniques, rule based methods were adopted, for example to perform image retargeting. Image retargeting aims at finding a higher aesthetic quality view inside a given image by re-framing and in one particular example [13] it was executed by introducing the input image to Itti's saliency extraction [33]. The extracted saliency map was thresholded to determine subject locations (using thresholded segments' bounding boxes), and their center of mass and area. Additionally after determining pixels corresponding to the edges in the image, prominent lines were detected using random sample consensus (RANSAC) algorithm and least squares method. Using this information, a rule of thirds score built upon distances of prominent lines and primary salient objects to power points, a diagonal dominance score based on distances of diagonal lines to image's main diagonals, and a visual balance score based on the dispersion of salient objects around image center were calculated. These scores were then combined and optimized using gradient descend and the higher quality frame found inside the input image was cropped.

Another automated image retargeting method similar to the previous one was presented in [24]. Lines extracted via segmentation based line detection, center of mass and size of the salient regions after Itti's saliency [33] analysis were used to optimize image aesthetic appearance and re-frame the image. A rather interesting approach was introduced in [22] in which the image was warped by moving salient objects and prominent lines into target locations to improve aesthetic quality, using predefined rules.

As another example of rule-based aesthetic quality improvement works, Cohen-Or et al. [16] used widely accepted color templates shown in Figure 4 to improve color harmony. The gray areas on these templates show the relative hue combinations that can be used together in an aesthetically pleasing way. These areas can be rotated freely and they will still keep their harmonic properties as long as their relative orientations do not change. The last template represents a monochromatic image. They used these hue relations to improve the color harmony properties of the images by manipulating the color distributions after analyzing the hues present in an image.

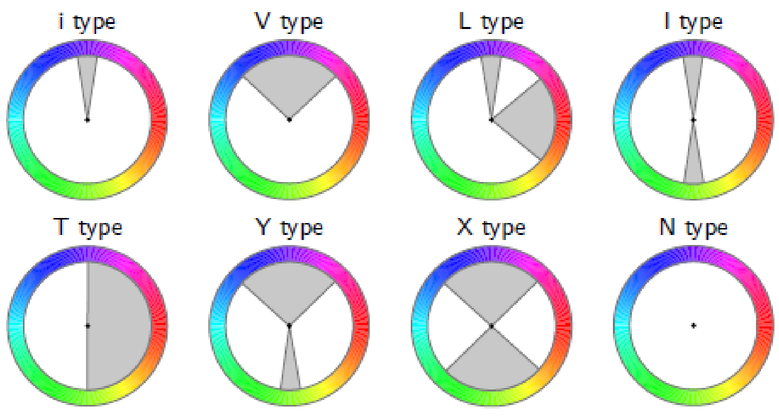

Figure 4: Color templates

Baluja et al. [28] trained an artificial neural network (ANN) and used it as a fitness measure on a human assisted genetic image generation system. 48 by 48 pixels 256 color images were fed into a neural network with various input and output connections and a better performing neural net architecture was developed that can capture the user's aesthetic preferences relatively well and generate new images using genetic algorithms. Their intent was using the scoring sub-system as a fitness function in the genetic algorithm and they reported the performance for various neural network configurations.

Machado et al. [9] relates visual aesthetics to visual acquisition system of human brain and developed formulations that try to explain aesthetics based on "image complexity" and "processing complexity". These two metrics were mainly based on jpg compression (discrete cosine transform, quantization and Huffman coding) and fractal compression, encoding simplicity and self-repetition properties of images respectively. In their experiments the computer evaluation system made better selections among given set of images than fine arts graduates [9]. A similar approach was also introduced in [29], relating complexity and aesthetics.

Moorthy et al. [34] described an aesthetic evaluation method to be used in videography. Focus/sharpness, color variety, luminance, hue harmony, rule of thirds, block artifact quality, motion related, and frame rate features were utilized to describe aesthetics of consumer videos. The color harmony feature was calculated by convolving the hue distribution of the image on the color wheel with seven color templates presented in [16] as explained previously. Peaks of the convolutions were used to determine the fit of each template and highest convolution peak template was used as a categorical feature (see section 2.4.10).

Luo et al. [25] used various region detection methods relying on the fact that low-level features calculated on image segments being superior to global features. Three region detection methods used include; "clarity based region detection" which involves over-segmenting the image based on clearness or blur amount of pixels; "layout based region detection" by finding sky, ground and vertical objects and "human based region detection" involving face detection. Furthermore they have used photographs belonging to different classes of photographs ("animal", "plant", "static", "architecture", "landscape", "human" and "night") and showed that various features perform differently on different classes. They have used images collected from photography websites and only the ones with a higher consensus were used to train a linear support vector machine (SVM). Their local (calculated on image patches) features include dark channel feature calculated on the previously found subject regions by averaging the normalized dark channel image (see section 2.4.2). Dark channel feature assesses the clarity and color distribution of the image at the same time. Also a hue composition feature inspired by [16] was calculated, fitting the color scheme of the image to the learned color wheel templates from high and low quality images instead of using pre-determined templates. In order to capture the line information in the image, they implemented Hough transform based line detection and calculated vertical and horizontal average orientations and locations of prominent lines extracted from the image. Maximum high/low quality classification rate was 0.9631 for landscape category and 0.9273 for face category. Other features used by Luo et al. [25] were; proportion of face areas, average face lighting, proportion of shadow areas, clarity of faces, and a complexity feature based on numbers of segmented regions in the image.

Datta el al. [12] developed aesthetic quality classification and regression computational models to be used to suggest a composition for digital camera users or to be used together with content based image retrieval systems favoring higher aesthetic quality images. 56 features were developed to generate the model using primarily the HSV color system. They introduced an image segmentation based approach, in which the images were converted into LUV color system. Using the K-means algorithm in this three dimensional LUV color space, color based clusters were found. Using these clusters the images were segmented into various regions and after connected component analysis the largest 5 regions were selected to be used in local feature calculations. To assess the lighting conditions, average values of the brightness channel of the HSV image was used. To measure colorfulness, earth mover's distance (EMD) between a hypothetical colorful image and the input image was implemented. To assess saturation of the image, average saturation was calculated on the saturation channel of the HSV image. Although it does not map to a linear space, average hue was also calculated in a similar manner (see section 2.4.1). Furthermore, the region of the image corresponding to the center rectangle formed by rule of third lines' was also used to calculate compositional features. Considering the fact that if rule of thirds were to be used, this region would be different than the remaining image for a high quality photograph. They calculated the average H, S and V values also for this center section as additional features. To measure textural properties of the image, wavelet based features were used (see section 2.4.3) and to determine use of depth of field, rule of thirds' inner region and the whole image's texture properties were compared. This time, instead of using the exact rule of thirds rectangle explained above, a larger area determined by dividing the frame into 16 rectangles and using the center 4 blocks was used. As simplicity features, the number of image regions that are larger than $image\ area/100$, and the total number of clusters found by K-means were used where the number of clusters was determined dynamically for each individual image. A shape convexity feature was calculated by finding the convex hull of segmented image regions and comparing these areas with the real shape of the segments. Average hue, saturation and brightness of largest five image segments and their relative sizes were also used as local composition features. They also utilized the size and aspect ratio of the input images (section 2.4.17). Other features calculated include; a "familiarity measure" comparing an image with other images using integrated region matching to find original images and complementary color features calculated among the largest local image segments. Using all these features, support vector machine (SVM) with a radial basis function (RBF) kernel and a linear regression model was trained achieving 5-fold cross validation correct classification rate of 70.12%. In case of regression, the performance was measured using residual sum of squares, or mean squared error (MSE) (see section 2.5.1). Using a linear regression model involving squared, cubed and 1/3, 2/3 powered features, they reached a 28% reduction from the variance (see section 2.5.2).

In [35], using the features from [12], a weighting scheme was introduced to improve classification accuracy. Datta et al. [14] further investigated various problems of computational aesthetics and described some of the previously used methods and introduced real datasets to be used in the field.

Ciesielski et al. [7] used features of Datta et al. [12] to analyze which of these features relates to aesthetic quality better by applying them on photographs and abstract images using a variety of classification techniques. They found that global color features were the most significant among this set. It was also stated that findings of Datta et al. [12] differs from their results and only a few of the features reported to be best, overlap [7].

Ke et al. [20] developed high-level image aesthetic quality features. To describe simplicity of the image, the center area of the image that edges were concentrated was used as a feature (see section 2.4.5). Furthermore for professional and snapshot quality images, Laplacian edge images were averaged to generate two different edge image templates. Then for a given image, the quality was calculated by finding the L_1 distance between the new edge image and these two template images (see section 2.4.6). Color distribution of the image was calculated by quantizing the image RGB values into a 4096 bin histogram (16 bin for each channel), and then a KNN search space was generated using the professional and snapshot images. Finding the 5 nearest neighbors for a given image using L_1 distance metric, the color quality was calculated as the difference between the number of professional and snapshot neighbor counts (see section 2.4.7). After generating a hue histogram, number of bins with values less than the 5% of the maximum hue count was used as a color simplicity feature (see section 2.4.8). To measure the sharpness of an image, a fast Fourier transform (FFT) based feature was used (see section 2.4.12). A contrast measure was calculated by finding the center 98% mass of the red, green, blue combined histogram as shown in section 2.4.13. Using these features and training a Real-ADAboost classifier, they achieved a professional-snapshot classification accuracy of 72% [20].

Li et al. [26] investigated the quality of paintings using a computational aesthetics approach. Both global and local features were implemented and Naive Bayes' classifier and ADAboost were used for classification. Local image patches were found using an adaptive segmentation. They also employed features that measure the contrast between features calculated on these image segments. To calculate the color relations, hue templates from [16] were used to fit the colors in the image using a customized hue distance between the hue distribution of the image and the templates (see section 2.4.10). Using the HSL color space and only considering pixels with brightness values between 0.15-0.95 and saturation over 0.2 (called “effective pixels”), a 20 bin hue histogram was generated. On this histogram hue counts higher than 10% of the maximum count was used to calculate a “hue count” feature and the number of pixels with the highest count was also calculated (see section 2.4.8). A “hue contrast” (section 2.4.9) feature was also introduced. Average saturation, hue and brightness features were computed on each respective channel. To measure brightness contrast, the method presented in section 2.4.13 was utilized, in which the breadth of a given amount of center mass covered on the brightness histogram is calculated. The blur amount of the image and area ratio of the box enclosing a certain ratio of edge energy was calculated in a similar manner to [20] (see section 2.4.5). With a similar approach to [12], Li et al. [26] also used the center rectangle (called the “focus region”) of the image encapsulating the power points to calculate further local features. This region was selected to be a little larger than the center rule of thirds rectangle in order to capture minor misplacements. Additionally, they used saturation-lightness model fits, logarithmic brightness average, shape features for segments; center of mass, variance, skewness, color features of segments (averages for three biggest regions); hue, saturation, lightness, contrast features between segments (maximum difference between top five biggest regions); hue contrast, saturation contrast, brightness contrast, and sharpness contrast as discriminative features [26].

Tong et al. [36] applied machine learning on COREL and Microsoft Office Online datasets to relate image features to aesthetic class as, taken by “photographers” or “home-users”. To select more important features, principle component analysis (PCA) and feature decorrelation was applied to the features to be fed into a Bayesian classifier and SVM. Also an ADAboost classifier was utilized without pre-selecting features, since boosting performs an internal feature importance assessment when training the classifier. Features used are in summary; saliency metrics: saliency map average, variance and third order moment, color: band difference, color moment, color histogram, coherence; energy: discrete Fourier transform coefficient and discrete cosine transform coefficient based features, texture: Tamura features [37] (section 2.4.15), wavelet based features, blur, contrast, and color variety measures. Sobel histograms describing gradient directions were used to describe shapes in the input images (see section 2.4.14). Canny and Laplace edge histograms were also calculated in a similar manner. This set of low level features was used to classify pictures with a lowest test misclassification error of 4.9% using the Bayesian classifier [36].

In [17] and [11], subject regions were extracted using a clarity measure and, clarity contrast, lighting contrast, 50 bin per-channel HSV histogram color harmony between detected regions and background were computed. Rule of thirds properties for these subject areas were also utilized. To measure the color simplicity of the background, RGB colors of the background region was quantized in a 4096 bin histogram (16 bin for each channel) and counts were calculated (see section 2.4.8).

Experiments were done on the CUHK [25] dataset using SVM, ADAboost and a Bayesian Classifier. The best achieved classification error rate was 5%.

Wong and Low [38] used a visual attention model (based on Itti's visual saliency model [33]) and segmentation to determine object regions in the image in order to classify images as high and low aesthetic quality. 98% center mass of luminance histogram was calculated to measure global image contrast (see section 2.4.13). Some of the features from [12] were also used. Lightness, sharpness, saturation and wavelet based texture features (only the sum of wavelet coefficients) on salient regions were calculated. Area of the salient region, salient region count, standard deviation of salient regions, mean and standard deviation of saliency map were among saliency related features. For many of the features, the squared difference of low-level features between the salient region and background was used to describe the contrast between them. Using a linear kernel they trained a SVM and reached a classification accuracy of 78.8%.

Another example use of visual attention and segmentation was given in [30]. Their work aimed at improving the composition of images and a software to assist users doing so was implemented. The software uses the learned aesthetic properties to find a better composition by replacing the position of the objects in an image, given some constraints. When determining aesthetic quality, features like distance of subject region’s centroid to each power point and proportion of pixels in the sky region to the pixels in the ground/sea region were utilized.

Desnoyer et al. [39], developed computational aesthetics methods proposed to be used with an independent real-life camera agent that can create aesthetically pleasing imagery similar to the one introduced in this thesis, but they did not elaborated or implemented it. To measure visual aesthetic quality, they employed image segmentation and divided the images into blocks to calculate features relating to simplicity, contrast, texture/blur and familiarity. They represented the hue distribution of an image by using the shifted hue histogram (see section 2.4.11). They reached a classification error rate of 18.1% on the Apollo dataset. Other features used were colorfulness from [12], harmonic colors (harmony among largest extracted segments), number of largest regions after segmentation, area covering 75% of edge energy similar to [20] (section 2.4.5), FFT blur feature as in [20] (section 2.4.12), Gabor filters, and region saliency. Many of these features were applied on: 3x3 and 5x5 divided image blocks and mean shift color segmented image patches. As contrast features, Michelson contrast and RMS of intensity values (summarized in section 2.4.1) were used on these patches and blocks.

In [32], an improved regression analysis was done by building upon the RankBoost ranking algorithm. The method they proposed as “Diff-RankBoost” was used that is based on the pairs of images and their pairwise rank relations. Features used include colorfulness, contrast, edge intensity on horizontal and vertical axes, normalized contrast, brightness, symmetry, sharpness, vanishing point histogram, Ke's features

[20], and face-related features such as; face count, size of most prominent face and its position. Many of the used features were calculated globally on the whole image, on a 3 by 3 grid and face areas. Their ranking method was shown to be better than various other approaches.

Machajdik and Hanbury [40] dealt with a classification problem to classify images into groups of emotions. Initially the images were pre-processed by cropping borders and resizing the images to have approximately 200k pixels. Waterfall segmentation was applied to find local image areas. A wide variety of image features were used, some borrowed from other works including average saturation and brightness, emotional coordinates built on saturation and brightness, vector based mean hue, colorfulness measure based on EMD as in [12], amount of various pre-defined colors, and Itten contrasts. To describe texture properties, [40] used Tamura features [37] (coarseness, contrast, directionality – section 2.4.15), wavelet based features similar to [12] for each HSV channels (section 2.4.3) and gray level co-occurrence matrix (GLCM) features (section 2.4.4). Other features used by [40] were; segment count after waterfall segmentation, wavelet coefficient ratio of center rule of thirds rectangle and whole image to measure depth of field, static/dynamic line slopes, lengths of detected lines, average saturation, brightness and hue for the inner rectangle, face count, size of the biggest face, skin pixels count (calculated by thresholding colors in a specialized color space), and ratio of skin areas to the faces.

The work on computational aesthetics of Lo et al. [41] stands-out at computational efficiency as they did not use any saliency or segmentation based features. To measure color quality of an input image, they built a 4096 bin histogram using the HSV color space (each channel quantized into 16 bins) representing the colors in the image similar to the method of Ke et al. [20], but instead of using the whole histogram, only the 5 extracted dominant colors were used in K-nearest Neighbor analysis (see section 2.4.7). To represent composition of the image, again they used a method similar to the one used by [20], where L_1 distance between a given image and the templates were used, but instead of using only the Laplacian edge image, H, S, V and H + S +V images were used together with their Laplacian filtered images adding up to 8 templates (section 2.4.6). The edge image templates also served as a saliency assessment [41] when compared with a given image. To represent the global texture of the image, the input image was sliced into 6, both vertically and horizontally and each neighboring pair was used to calculate sum of differences (see section 2.4.16). Other than these features, [41] used the dark channel feature (Luo et al. [25], section 2.4.2) on the entire image, FFT blur metric similar to [20] (section 2.4.12), and calculated brightness contrast as in section 2.4.13 finding the center mass width of the brightness histogram. Non-zero bins of H, S and V channels were also utilized. Using the CUHK database, they have trained an SVM classifier and achieved a classification rate of above 90% for some of the photographic classes [41].

Guo et al. [42] used Locality constrained linear coding in accordance with scale invariant feature transform (SIFT) descriptors of image blocks, spatial pyramid segmentation, and slope of the magnitude spectrum (SMS) to analyze visual aesthetics efficiently. They further utilized an image clarity based subject region extraction. Features used were; rule of thirds: average values and histograms of SMS in inner thirds region and whole image, color harmony: subject, background and whole image histograms in HSV color space, geometrical features: proportion of pixel counts of subject region to whole image, contrast feature: Michelson contrast calculated over the entire image (section 2.4.1), and semantic features using SIFT. CUHK [17] and Photo quality dataset [11] were used to train SVM, KNN and ADAboost classifiers and a maximum classification performance of 95.01% was achieved in the human category.

Obrador et al. [21] utilized compositional features via segmentation and aimed at finding "accent" regions in an image to classify images with respect to their aesthetic value. A relevance value for the segments was calculated using their size and relative brightness and to decide on accent regions, color contrasts were calculated. By means of a SVM based classification they achieved 66.5% classification accuracy using the top and bottom 8% of the images from the dataset of [12]. Features used were; simplicity: segment count, relevant segment count, accent region count, appeal: mean distance between centers of relevant regions and its standard deviation calculated directly among centroids and among the bounding circles of the regions. To measure visual balance, pre-defined template images that were generated considering rule of thirds and golden ratio were compared against the relevant region positions.

Dhar et al. [43] classified aesthetic quality of images using saliency analysis, Viola-Jones face detection, and SIFT. The saliency analysis was executed by introducing a multi-scale contrast map, a center surround histogram map, and a center weighed color spatial distribution map to a conditional random field and training it on images with highly salient objects. Wavelet based features similar to [12] together with third level wavelet coefficients of inner third region divided by coefficients of entire image, product of min distance between centroid of the saliency map to the power points and third-lines, existence of faces, animal existence, and sky lighting properties after detecting sky regions were also used.

Zhang et al. [44] analyzed aesthetics of abstract art using SVM based machine learning, trying to classify them as exciting - boring, relaxing - irritating. Haralick texture features, multiscale histograms, Tamura properties [37], and Radon transform features were used to describe texture. Zernike features, edge statistics features, and Gabor filters were used to describe shape and edges in the image. Furthermore, features like Chebyshev statistics, and Chebyshev-Fourier were used to achieve the objective.

Solli and Lenz [45] used colors' emotional interactions (namely "activity", "weight" and "heat") as a part of a content based image retrieval system using bags-of-emotions to demonstrate how high level semantic information (like aesthetics or emotions) can be introduced. Utilizing histograms of color emotions, they found images inducing similar feelings based on an L_2 distance metric. This paper relates to the subject as visual cues like colors used in an image were related to the emotions they evoke.

Marchesotti et al. [46] aimed at finding more generic features to evaluate visual aesthetics instead of hand-crafted and heuristic features. They used bag-of-visual-words (BOV) descriptor, Fisher vector, SIFT, and gist descriptors. They reduced the feature count using PCA and utilized photoNET and CUHK [20] datasets reaching 89.9% classification accuracy using a linear kernel SVM.

The work of Obrador et al. [10] aimed at improving prediction performance by finding contrasting regions in the image with respect to the low-level features and thus improving low-level feature prediction power. For this reason, a graph-based image segmentation method on a scaled down version of the image was incorporated. On these segmented regions, hue, relevance, maximum saliency and maximum sharpness were calculated. These local features were then thresholded to determine two groups of segments (ten in total) that have high and low values. Using these thresholded segments, weighted average of hue, maximum sharpness and saliency as well as mean, standard deviation, minimum, maximum and 1st , 2nd, 3rd quartiles of CIELab luminance were calculated as "contrasting region features". Simplicity features used include the number of segments larger than a specified percentage of the image, following the intuition that a less segmented image is less cluttered and simpler. As global features, luminance (average, maximum, minimum), luminance RMS contrast, color variety measures, and color harmony metrics were used. The luminance values for each pixel in the image were calculated by transforming the image into the CIELab color space. Color harmony feature was calculated in a similar fashion to [34] by convolving the hue histogram of the image with seven color templates presented in [16] (Figure 4). Peaks of the convolutions were used to determine the fit of each template. In contrast to [34] that calculate the best matching (highest convolution peak) template as a single feature, Obrador et al. used the values for each template as independent features [21] (see section 2.4.10). To be able to capture compositional properties of the image, pre-calculated templates as in [21] considering rule of thirds and golden ratio were used and instead of finding centroid of objects on these templates, image edge maps were intersected with these templates. Furthermore, different aesthetic systems for seven different classes (architecture, animals, cityscape, floral, landscape, portraiture, seascapes) of photographs were built with different selected feature subsets. The machine learning algorithm used mainly was support vector machine regression. For each class of photographs, filter and wrapper based feature selection was applied using 5-fold cross validation (CV). First, top 50% performing features on their own were selected and then using the highest performing feature as the initial model, features that increase the performance of the model the most, were introduced until some criteria is achieved. The 5-fold CV performance for the “landscape” category was 24.4% reduction from the sample variance, being highest achieved performance among photographic classes [10].

There were also some studies specifically focused on human portraiture. Jin et al. [47] investigated the visual aesthetics of lighting on portrait photography and [15] evaluated aesthetic quality using bag of poses and bag of expressions on images with human faces. The latter reported a 25% reduction from the score sample variance using the predictive model. Additionally in [5], aesthetic quality of human portraits was investigated using various composition, color, texture and statistical features. The centroid of a face was compared with the predefined templates similar to the method in [21]. KNN classifier, support vector machine (SVM), random forests, classification via regression, and multi-boosting ADAboost was used in this study reaching 61.1% classification accuracy. Evaluation of face images is not directly related to the scope of this thesis but these papers were included here for completeness.

In [31], sharpness and colorfulness features were used together with text based information related to the images in online communities. The attractiveness of the images was predicted and ranked to aid searching. Using only visual features, Kendall’s Tau-b (section 2.5.4) score of 0.2523 was attained. When used together with textual features the performance was increased to 0.4841.

Based on their influences on some drawbacks of other datasets, Murray et al. [48] developed an aesthetic-oriented large-scale Aesthetic Visual Analysis (AVA) dataset. Above 250k images were collected from the photography website www.dpchallenge.com, in which various photography challenges are held. Using the titles and explanations of these challenges, semantic, aesthetic and photographic tags were generated for many of the images in the dataset. The individual images with a lower score variance were reported to generally employ a more conventional photographic style or picture a more conventional subject. Furthermore, by using machine learning algorithms with extreme scored images (scored delta above and delta below 5) was shown to perform better.

Faria et al. [49] applied random forests classification using many of the features that were previously used on visual aesthetics analysis and achieved higher classification rates than some of the previous methods, showing the better performance of random forests.

Fedorovskaya et al. [50] did a subject study to evaluate the effectiveness of image features on image harmony interpretation. In CIELab color space, standard deviation of luminance, mean luminance, number of segmented patches (as a complexity metric), symmetry measure using SIFT (diagonal, horizontal, and vertical) features were incorporated. Some of the top features that relate to image harmony better were; edge contrast, extent of lightness, contrast between segments and the “good figure" factor. A similar study was done in [51] using principal component and cluster analysis, in which images with various properties were rated by the participants and technical image quality, vanishing point location, face expressions and subject location were found to be the most prominent aesthetic related properties.

2.3 Machine Learning

Many of the existing works utilized some sort of a machine learning algorithm on the features extracted from the input images. In this section, a very short description of machine learning is given and the machine learning algorithms used throughout this thesis work (tree bagging, random forests and least squares boosting) are explained in detail.

Machine learning is a field of computer science that develops and applies methods to learn from a given data (called the training set), a relation to a variable to be predicted when a new sample is introduced. This can be done through supervised or unsupervised learning. In supervised learning, the training set from which the relation is to be learned already has the appropriate labels representing the true response (ground truth) for given set of parameters (or features). Unsupervised learning on the other hand do not require the labels of the data and is out of the scope of this work as almost all previous works used a supervised learning to predict aesthetic quality.

There are basically two kinds of supervised prediction methods; classification and regression. While classification aims at finding the response to the given data as a label selected from various classes of items, regression responds the data with a numerical value. In the case of function approximation, for a given set of input variables (feature vector) $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ the aim is to find the response y. When there is a training set of N, $\{\boldsymbol{x}_i,\ y_i\}$ pairs then the objective is finding a function that map $\boldsymbol{x}$ to y minimizing a loss function; which is generally squared error for regression tasks [52]. There are many approaches to solving these general prediction problems and those employed in this thesis are introduced below.

2.3.1 Ensemble Methods

Ensemble learning methods combine various classification or regression models (base predictors) to form a new fused model that improves accuracy with respect to each individual model [53]. Ensemble learning algorithms has various approaches which can be classified roughly as averaging and boosting methods. In averaging methods, all the models in the ensemble are trained using the data separately and vote for the result for classification or their results are averaged for regression. In boosting methods on the other hand, a new model is introduced in the ensemble one by one trying to adjust the new model correcting the mistakes (reducing the loss function) of the previous models in the ensemble [53].

2.3.2 Classification and Regression Tree

One of the commonly used base predictors in ensemble learning methods is the classification and regression tree (CART). CARTs are basically decision tree structures that split the various inputs of the learning model at each node into subtrees using one of the inputs $\boldsymbol{x_i}$ and report the final predicted value in the terminal node at the bottom of the tree. The best split at each node is found by iterating over each variable in the input vector and calculating an impurity value for each of them [54]. CARTs can be used in both regression and classification configuration as the name implies. In the case of classification, the terminal node holds the class and in the case of regression, it holds the output value to be reported for the given input vector $\boldsymbol{x}$. One advantage of using CARTs is that it allows usage of categorical (not real valued) inputs. In Figure 5, an example regression tree was presented. $x_1, x_2, \ldots$ represent the input variables where x_4 is a categorical predictor. The values are split at each node with respect to the variables and predictions are made at the terminal nodes.

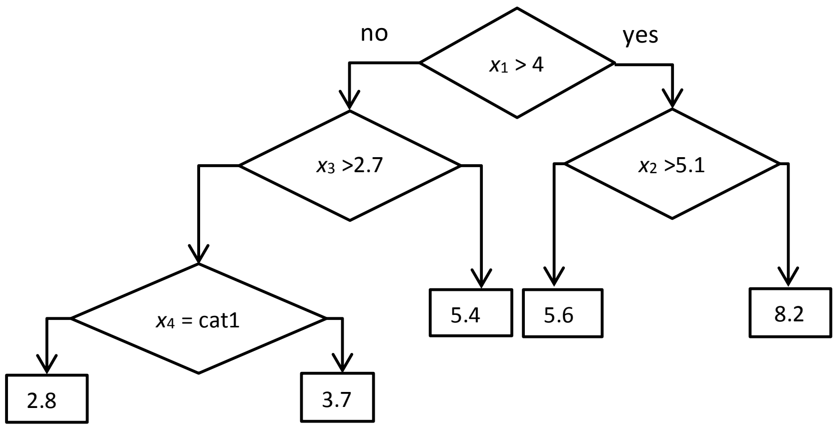

Figure 5: Example regression tree

Another property of classification and regression trees is that using the splits in the tree, it is possible to predict the relative importance of each input variable $\boldsymbol{x_i}$. At each node of the tree, the amount of error that was reduced with respect to not splitting at that node is used to determine the relative effect of that candidate feature. When these probable error reductions are summed for each node of the tree, in which this particular variable is used as the splitting criterion, the influence (or importance) of each predictor can be calculated repeating this process for every $\boldsymbol{x_i}$ [55].

2.3.3 Bagging

CARTs are generally not preferred on their own since small variations in the training data causes the outputs to change unpredictably. They are considered unstable learners and used in conjunction with some form of ensemble learning. Breiman introduced the concept of bootstrap aggregation or "bagging" to the ensembles of such unstable methods (originally ensembles of classification and regression trees) [56]. When a bagging methodology is used in an ensemble learning algorithm, each model in the ensemble is trained on a different partition of the data which is selected with replacement out of the original training data. If there are N samples in the training data and N random selections with replacement are performed, there is the probability of using same samples for the current model more than once as well as the probability of not using some fraction of the training data in any sub-model [53]. Statistically for each learner, the expected number of uniquely selected samples is about 66% of the whole set when N selections are made out of N samples, meaning that about one third of the samples will not be selected on average [57]. With this bagging strategy, it is possible to improve the robustness of the ensemble [56]. The training samples that are not selected -because of the bootstrap nature of the selection process- for a particular sub-model in the ensemble are called out-of-bag (OOB) samples for that model. Using these as a test set for that particular sub-model, an estimate for the prediction error of that learner can be calculated. When these errors are averaged over the entire ensemble, the resulting value can be used to estimate the out of sample performance of the ensemble [58]. In this thesis, ensemble of regression trees is used in a bagging configuration.

2.3.4 Random Forests

Random forests, in essence are ensemble learners that use CARTs as the individual models. In addition to randomly sampling the training set with replacement as it is in bagging, Random Forests introduced by Breiman [57], also uses a randomly selected subset of predictor variables in the training of individual trees in the ensemble. When the random variable selection is not applied, the working principle is the same as a regular bagged tree ensemble described above. Additionally, the trees in the ensemble are not pruned according to Breiman's instructions.

Another variable ranking method was introduced by Breiman in his random forests paper that is applicable specifically to bagging ensembles. With this approach, the OOB samples are used to estimate variable importance. By randomly permuting a single variable's values among the OOB samples of a tree (replacing the values of the ith input ($\boldsymbol{x}_i$) randomly across samples, thus breaking the relation of this variable to the output y_i), a prediction error is computed. Then by comparing this value with the actual (without permuting anything) OOB prediction error, it is possible to predict the performance loss caused by that specific variable being scrambled for that tree. By averaging the same calculation for each tree in the ensemble and repeating the procedure for each feature, the relative importance of each particular variable is predicted [57].

2.3.5 Least Squares Boosting

In least squares boosting using regression trees, a new tree is introduced to the ensemble whose effect is closest to the negative gradient direction found using the squared error loss function's steepest descent [55]. This criterion is fulfilled by introducing trees that are split accordingly and the ensemble model is built. With each new tree added, the prediction performance of the overall model is expected to increase. Since CARTs are used in the building of a least squares boosting ensemble, it is possible to predict variable importance using the tree split criterion explained above.

2.3.6 Feature Selection

When dealing with machine learning it is sometimes required to eliminate some of the features in order to improve prediction performance, reduce computational cost or better understand the data at hand/visualize data. There are three main classes of feature selection processes; filter methods that depend on the dependence and relation of the features, wrapper methods using the learning algorithm in analysis and embedded methods. In filter methods, the features are pre-processed in order to find correlated features to be eliminated. In wrapper methods, the selection process is "wrapped" in the learning algorithm and resulting performance of the model by adding or removing features are measured by validation. When features are added to the model, it is called forward selection and when they are eliminated it is called backward elimination. Forward selection/backward elimination are much more computationally friendly methods than trying every possible combination of input variables. The final class of feature selection, namely embedded methods, depends on the learning algorithm at hand which must be capable of evaluating features while training the model [59].

There exist various works that uses variable importance measures for feature selection using an embedded approach. Boosted trees' squared-error decrease induced by the splits was used in a wrapper-based feature selection in [60] to reduce the number of used features due to computational requirements. The OOB permuted error metric explained above was used to select more important features to be fed into a support vector machine (SVM) in [61]. A wrapper based method using random forests algorithm and its embedded OOB permuted error metric was described in

[58] using a recursive and non-recursive setup. By eliminating features based on a wrapper approach, discarding half of the features that have lowest predictor importance in a cross-validation loop, it was possible to improve the prediction performance to some extent [58] possibly eliminating some possible source of over fitting that may be caused by high-feature count or redundant features.

2.4 Features

In this section, some of the features introduced in section 2.2 are elaborated and consolidated with the emphasis on the ones that have been used throughout this work.

2.4.1 Image Statistics

These are the simplest among all features considered and give general information on a given image. Yet they were used extensively on various channels of different color transformed images [12], [38], [50], [10].

Average value (μ) for a single channel grayscale image I (or region of interest) is calculated as in equation (1);

$$\mu = \frac{1}{N}\sum_{i \in I} V_i \tag{1}$$

In equation (1), i is the index of a single pixel in the image I, V_i is the value at pixel i and N is the total number of pixels in the image.

Maximum and minimum values are the highest and lowest values of V_i over the entire image I.

Root mean squared (RMS) contrast (or equivalently standard deviation of luminance values as used in [50]) of image I is calculated as in equation (2);

$$\sqrt{\frac{1}{N}\sum_{i \in I} (V_i - \mu)^2} \tag{2}$$

Michelson contrast is calculated as;

$$Michelson\ Contrast = \frac{V_{max} - V_{min}}{V_{max} + V_{min}} \tag{3}$$

Where V_{max} and V_{min} are the maximum and minimum values of the quantity under consideration [42].

2.4.2 Dark Channel

Dark channel of an image is calculated using equation (4).

$$I_{dark}(i) = \min_{c \in RGB}\left(\min_{i' \in \partial(i)} I_c(i')\right) \tag{4}$$

Where I_{dark} (i) is the dark channel value for pixel i, ∂(i) is the neighboring pixels (10 by 10 neighborhood in this work, using the 5th pixel from left and top as the center point) of current pixel i and I_c is the currently processed channel of the image (R = red, G = green, B = blue).

To summarize the process, each pixel in the image is assigned the minimum value of red, green or blue value around its neighborhood. After calculating this dark channel image, the values are averaged over the region of interest using equation (1) and used as a feature [25], [41].

2.4.3 Wavelet Features

A three level wavelet transform for the H, S and V channels are calculated. The sum of the wavelet coefficients (HH, HL, LH) in levels 1, 2, and 3, normalized with the size of the wavelet transforms are used as 9 texture features respectively. Furthermore for each channel H, S, and V the sum of the wavelet coefficients are also calculated as three new features summing up to 12 features [12], [38], [36]. The intricate details of the wavelet transforms are not introduced here.

2.4.4 Gray Level Co-Occurrence Matrix

Gray level co-occurrence matrix is used to detect repetition patterns and texture in a gray image. First the image is quantized on a given number of discrete levels N and then an N by N matrix is generated and filled with the numbers of found patterns with a given offset. This offset determines the second pixel relative to the pixel of interest, for which the pattern is searched for. For example if the offset is set as the right adjacent pixel, whenever a value **k** is followed by a value **n** on its right, **k**$^{\mathbf{th}}$ row and **n**$^{\mathbf{th}}$ column of the matrix is incremented [62]. After calculating the matrix, contrast, correlation, energy and homogeneity properties are calculated using the counts in this matrix. These properties are used as image features to describe texture [40] using a single pixel offset and 8 discrete levels on the hue, saturation and brightness channels.

2.4.5 Edge Concentration Area

To calculate this feature, first an edge intensity image is calculated. The input image's R, G and B channels were convolved with a 3 by 3 Laplacian filter with $\propto = 0.2$ shown in Figure 6 [20].

$$\begin{bmatrix} 0.1667 & 0.6667 & 0.1667 \\ 0.6667 & -3.3333 & 0.6667 \\ 0.1667 & 0.6667 & 0.1667 \end{bmatrix}$$

Figure 6: 3 by 3 Laplacian filter

Then the mean of the absolute values (to neglect gradient directions) of Laplacian filtered images of three channels is calculated, followed by a scaling operation down to 100 by 100 pixels. Finally, the image is normalized to make the sum equal to 1 [20].

The area of the bounding box that encapsulates top 96.04% of edge energy is calculated by projecting (summing) the resulting edge image vertically and horizontally. The breadth of 98% mass of these projections in two directions are found with a similar approach that will be introduced in section 2.4.13. The spans of the two projection's center mass are used as the width and height of the edge concentration region and are multiplied to calculate the concentration area.

2.4.6 Image Template Comparisons

A set of images are collected to represent two opponent classes of images (e.g. high aesthetic quality and low aesthetic quality) and they are averaged to generate two "template" images T^H and T^L. Given a new (or probe) image L, the quality of the new image with respect to these templates was calculated by the difference of two L_1 distances between the new image and two templates as in equation (5).

$$Q = \sum_{i \in I} \left| L_i - T_i^L \right| - \sum_{i \in I} \left| L_i - T_i^H \right| \qquad (5)$$

In equation (5), Q is the template based image quality; L_i is the value of ith pixel in the image in question, T_i^L is the value of ith pixel of the low quality template and T_i^H is the value of ith pixel of the high quality template. Ke et al. only used this metric on the 100 by 100 pixel Laplacian edge images of professional and snapshot images [20] and Lo et al. [41] also used this metric on H, S, V, H+S+V channels in addition to their Laplacian filtered edge images on high and low quality images generating 8 independent features.

2.4.7 Color Quality

To measure color quality of an input image, instead of using template based approaches as will be described in section 2.4.10, [41] built a 4096 bin histogram using the HSV color space (quantizing each channel into 16 bins) representing the candidate colors in the image. In this three dimensional quantized space, weighted K-means algorithm with K = 5 is executed to find 5 clusters of colors. To determine the dominant colors in the image, instead of using the centers of clusters directly, colors with higher weights are preferred by means of equation (6).

$$Dom(j) = argmax_{C_i \in cluster\ j} \left(\alpha h(i) + \frac{(1-\alpha)}{\left\| C_i - V_j \right\|} \right) \qquad (6)$$

For each cluster j, the dominant color $Dom(j)$ is calculated with the above equation, where $h(i)$ is the number of pixels in the ith bin of the histogram, C_i is a candidate color in cluster j and V_j is the center of cluster j. The candidate color that maximizes the quantity in equation (6) was selected as the jth dominant color. α is a parameter that balances the high histogram weight requirement and distance to the cluster center. After calculating these 15 dimensional (3 channels with 5 dominant colors) vectors for the high and low quality images, a K-nearest neighbor search space is built and for a new image color quality feature is calculated as;

$$Color\ Quality = n_H - n_L \qquad (7)$$

In equation (7), n_H is the number of neighbors in the high class and n_L is the number of neighbors in the low class [41]. Following a different approach, Ke et al.

used the whole RGB 4096 bin histogram instead of the dominant colors as the K-nearest neighbor search space (4096 dimensional) in a similar manner [20].

2.4.8 Bin Counts

These features utilize various histograms by finding the number of bins that are empty or very low in count and were generally used as simplicity features.

To measure the simplicity of the background, RGB colors of the background region are quantized in a 4096 bin histogram (16 bins for each channel) as was used in [17].

$$Simplicity = \left(\frac{\|S\|}{4096}\right) \times 100\% \tag{8}$$

In equation (8), color simplicity of the background was meant to be calculated. $\|S\|$ is the number of bins of the histogram, the bin count being greater than γh_{max}. γ is a threshold chosen as 0.01 and h_{max} is the maximum bin count of the histogram.

Number of hues in the image was calculated as a simplicity feature on the hue channel, only considering pixels with brightness values between 0.15-0.95 and saturation over 0.2 since the values outside this range were indistinguishable to the eye. After generating a histogram, number of bins with values less than the 5% of the maximum count was used as a color simplicity feature, named “hue count” [20], [26].

Additionally, number of pixels belonging to the most frequent hue was also calculated using the maximum bin count of this filtered hue histogram [26].

2.4.9 Hue Contrast

A “hue contrast” feature is defined on the filtered hue histogram as;

$$Hue\ Contrast = \max(\|\overline{I_H}(i) - \overline{I_H}(j)\|) \tag{9}$$

In equation (9), $\overline{I_H}(i)$ and $\overline{I_H}(j)$ represents the hue values in the 10% thresholded hue histogram described in section 2.4.8 and the largest distance among them is used as a hue contrast feature. Here $\|\cdot\|$ operator calculates the arc-length distance between two hue values since the hues are represented as angles on the hue wheel [26].

2.4.10 Hue Template Fits

In [34] and [10] the templates in Figure 4 were convolved with the hue distribution of the image in question. For each template, the peak of the convolution operation was found and used as the fit value for that template. Two different approaches were using the best fitting template as a categorical feature or using individual fit values as independent features.

Another approach to fitting these templates was introduced in [26]. On the hue channel of the image, $E_{T_k(\alpha)}(i)$ is defined as the closest hue model boundaries (gray regions in Figure 4) of the ith pixel in the hue image ($I_H(i)$) as shown in equation (10), where $T_k(\alpha)$ is the hue model rotated by an angle α. When the ith value is inside a template region G_k, its value $[I_H(i)]$ is used as the boundary value and when it is outside, the value of the nearest border ($H_{rearest}$) is used.

$$E_{T_k(\alpha)}(i) = \begin{cases} I_H(i) & if\ I_H(i) \in G_k \\ H_{rearest} & if\ I_H(i) \notin G_k \end{cases} \tag{10}$$

Then the distance ($F_{k,\alpha}$) between the image hue distribution and the template k rotated by α becomes;

$$F_{k,\alpha} = \sum_{i \in I_H} \left\| I_H(i) - E_{T_k(\alpha)}(i) \right\| . I_S(i) \tag{11}$$

In equation (11), ||·|| operator calculates the arc distance between two hue values and $I_S(i)$ is the ith pixel value of the saturation image, used to weight pixels with higher saturation values. At this point the α value that minimizes this equation should be found for each template k using equation (12).

$$\alpha(k) = argmin_{\propto}(F_{k,\propto}) \tag{12}$$

After calculating the α minimizing distance sums for each template, the best fitting template is the one that gives the smallest $F_{k,\alpha}$. This template might not be the best solution since some of the templates contains other ones (for example i-type is included in V-type) therefore the selection is done using a threshold. If some of the template fit values are smaller than this threshold, the strictest one is selected in the order of i-type, I-type, V-type, Y-type, L-type, X-type, T-type with respect to strictness. When all of them are above the threshold, the one with the best fit (minimal $F_{k,\alpha}$) is selected as the hue template for the current image.

2.4.11 Adjusted Hue Histogram

Desnoyer and Wettergreen [39] used histogram of hues adjusted so that the dominant color (the bin with the highest count) is at 0 degrees. To achieve this, the image's hue histogram was cyclically shifted to bring the highest hue count on the initial bin. The driving force behind this operation is leaving the hue relations in an image to the machine learning algorithm, which may learn their interactions (e.g. complementary colors etc.) on its own, thus capturing a more detailed understanding of hue distribution relations.

2.4.12 FFT Blur Metric

A blurry image is assumed to be an image convolved with a Gaussian smoothing filter G_σ with the smoothing parameter σ. The relation of the smoothed image can be represented by equation (13).

$$I_b = G_\sigma * I \tag{13}$$

In equation (13), * is a two dimensional convolution operation. Applying a two dimensional fast Fourier transform (FFT), the high frequency content of the image I_b can be estimated counting the number of frequencies greater than some threshold $\theta = 5$. Since the G_σ filter removes only the high frequencies, the ratio of remaining lower frequencies above the threshold can be used to approximate $1/\sigma$ and therefore the sharpness of the image [20].

2.4.13 Middle Mass of an Histogram

This feature aims to find the spread of the brightness throughout the image. The width of the center part of the brightness histogram was used as a contrast measure by finding the value range covering center 98% of the histogram counts [20]. It is calculated by starting from the maximum bin of the brightness histogram and moving towards each end until the sum reaches 98% of all counts combined [26]. In Figure 7, the process was visualized. The horizontal axis represents the values and vertical axis represents the counts at each of these values. The area under the curve between the two vertical bounds is equal to 98% of the area under the whole distribution curve and the value range between these bounds was used as a contrast feature.

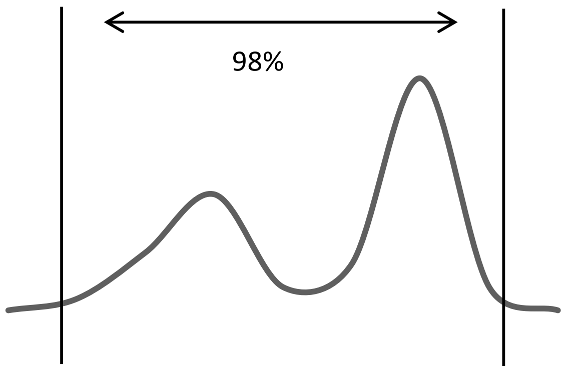

Figure 7: Calculating 98% center mass for a histogram

2.4.14 Edge Histograms

Edge histogram of an image gives clues about the directionalities of the gradients and edges in the image. A Sobel histogram is calculated by first convolving the gray image data (*I*) using the horizontal and vertical Sobel operators to approximate local derivatives Δ_H and Δ_V using equation (14).

$$\Delta_H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad \Delta_V = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \tag{14}$$

In equation (14), * represents the two dimensional convolution operation. Then using these derivatives in two different directions, it is further possible to approximate the local direction (θ) using four quadrant inverse tangent as in equation (15) and magnitude (M) using equation (16).

$$\theta = \text{atan2}\ (\Delta_V, \Delta_H) \tag{15}$$

$$M = \sqrt{\Delta_V + \Delta_H} \tag{16}$$

Using the calculated directions for each pixel in the image, they are separated into 15 bins to form a histogram of directions in the image [36].

2.4.15 Tamura Features

Tamura features were used extensively in the previous works and Tamura's measures were formulated in 1978 in order to understand texture properties of the images [37]. There are three mainly used Tamura texture features; directionality, contrast and coarseness.

Tamura directionality is calculated on a direction histogram very similar to the one introduced by equation (14), (15) and (16). In this case instead of the Sobel operator, Prewitt operator is used for convolution as shown in equation (17). To generate the histogram, the values are thresholded with the values calculated by equation (16). The directionality measure is calculated by the sharpness of the peaks in the direction histogram [37].

$$\Delta_H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad \Delta_V = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * I \tag{17}$$

Tamura contrast on the other hand is calculated on the brightness histogram as shown in equation (18).

$$Tamura\ Contrast = \sigma / \left(\frac{\mu_4}{\sigma^4}\right)^{1/4} \tag{18}$$

Here, σ^2 is the standard deviation and μ_4 is the fourth moment about the mean, of the brightness histogram.

Third Tamura feature, namely Tamura coarseness is calculated by taking the averages of $2^k \times 2^k$ neighborhoods of each pixel where k is a power of 2. Then differences between non-overlapping neighborhoods are calculated in horizontal and vertical directions. At each point the size 2^k giving the highest difference value is recorded and named S_{best}. These values are averaged over the entire image giving the coarseness measure [37].

2.4.16 Global Texture

To represent the texture of the image, the input (probe) image is sliced into 6 both vertically and horizontally and each neighboring pair is used to calculate sum of differences. The slicing operation is visualized in Figure 8 [41].

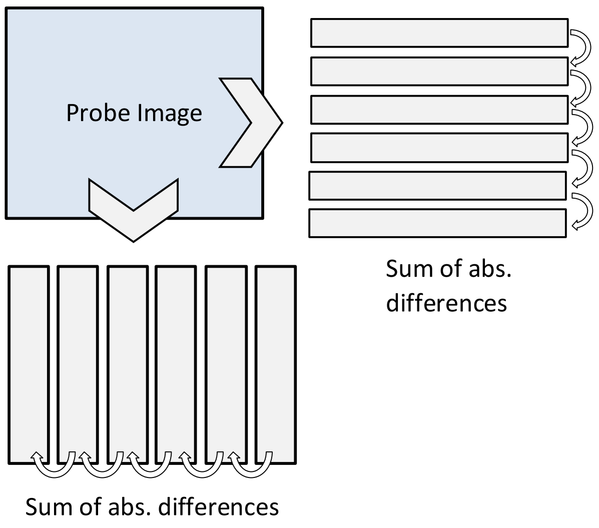

Figure 8: Slicing the input image to calculate global texture features

This process is repeated for each channel (H, S, V and H+S+V) including the respective Laplacian edge images and a set of 8 texture features is calculated.

$$Global\ Texture = \sum_{k=1}^{5} \sum_{i \in V^k} \left| V_i^k - V_{i+1}^k \right| + \sum_{k=1}^{5} \sum_{i \in S^k} \left| H_i^k - H_{i+1}^k \right| \qquad (19)$$

In equation(19), V^k and H^k are the kth vertical and horizontal slices of images respectively. The subscript i represents the pixel number in that slice and V_i^k and H_i^k are the ith pixel values of the slices [41].

2.4.17 General Features

To deal with size and aspect ratio of the images, two features are utilized. In equation (20) and (21), X is the width, and Y is the height of the original (not scaled) image in pixels [12].

$$Aspect\ Ratio = \frac{X}{Y} \qquad (20)$$

$$Size\ Feature = X + Y \qquad (21)$$

2.5 Performance Criteria

In this section, the performance metrics that were used in the prediction performance analysis of the regression machine learning models are summarized. In order for the proposed method to be effective at analyzing the virtual scene, a regression analysis was required. To have a meaningful regression performance measure, mainly five criteria are utilized. Since in the final application, a difference in the aesthetic quality for different rendered scenes is desired, it is not critical for the trained model to predict the scores spot on. Rather, it is important to be able to differentiate two images and determine which one is better. Therefore to evaluate the relative rankings of the predicted and actual (ground truth) aesthetic scores, two different rank correlation coefficients were also incorporated. In all of the following metrics, for a two sets of values $\{y_1, \dots, y_n\}$ and $\{\hat{y}_1, \dots, \hat{y}_n\}$ containing n elements (here, the ground truth values y_i and corresponding predictions $\hat{y}_i$), it is aimed to measure how well the second set follows the first.

2.5.1 Mean Squared Error

This criteria measures how much the predicted scores deviate from the actual (ground truth) scores of the data and used as a performance metric frequently when evaluating regression models. This metric was also called residual sum-of-squares error and the squared distances between the predicted and ground truth values are averaged as shown in equation (22) [12].

$$MSE = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad (22)$$

In equation (22), N is the number of samples used for error calculation, $\hat{y}_i$ is the predicted score for the ith item and y_i is the ground truth score for that item. Lower MSE values mean better performance.

2.5.2 Reduction from the Variance

Since the value range for different regression models are possibly different, a more objective method for reporting performance is called reduction from the variance [12]. If a primitive regression model was established that assigns every sample the mean (μ) of the ground truth scores as its prediction then, the MSE of this constant model would become the variance (σ^2) of the sample;

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu)^2 \tag{23}$$

If the trained regression model is resolving some of the aspects of the underlying data with its predictions then the MSE of the model should be smaller than the variance of the sample. Based on this proposition, the new performance criterion becomes reduction from the variance (24), measuring how much smaller is the MSE. In equation (24), σ^2 is the sample variance for the whole ground truth data labels and *MSE* is the mean squared error of the model as calculated by equation (22) [12]. A 100% reduction from variance means that the two sets follows each other exactly and as this number decreases, performance also diminishes.

$$reduction\ from\ the\ variance\ = \frac{\sigma^2 - MSE}{\sigma^2} \times 100\% \tag{24}$$

2.5.3 Correlation Coefficient

For the two sets of values, another way of measuring regression performance is evaluating the correlation between them. The simplest form of correlation analysis is through Pearson product-moment correlation coefficient, which reports the linear dependence of these two collections of values. The sample Pearson correlation coefficient represented by *r* is calculated as;

$$r = \frac{\sum_{i=1}^{n}(y_i - \mu_y)(\hat{y}_i - \mu_{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \mu_{\hat{y}})^2}} \tag{25}$$

In equation (25), μ_y and $\mu_{\hat{y}}$ are the sample mean for the two sets of values respectively and y_i and $\hat{y}_i$ are the ith values in each set [63].

2.5.4 Kendall's Rank Correlation Coefficient

Kendall's rank correlation coefficient (also called Kendall's Tau-b) is calculated as shown in equation (26).

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + T_1)(P + Q + T_2)}} \tag{26}$$

In equation (26), P is the number of concordant pairs, Q is the number of discordant pairs and T_1 and T_2 are the number of tied pairs in the first and second set of values respectively as summarized in [31]. For each pair of values $\{(y_i, \hat{y}_i), (y_j, \hat{y}_j)\}$; if the relative ordering of ith and jth values are the same ($y_i > y_j$ and $\hat{y}_i > \hat{y}_j$ or $y_i < y_j$ and $y_i < y_j$), they are counted as concordant pairs. If relative order do not agree ($y_i > y_j$ and $\hat{y}_i < \hat{y}_j$ or $y_i < y_j$ and $\hat{y}_j > \hat{y}_j$) these pairs are counted as discordant pairs. If orders are the same in any of the sets, they are considered tied in the first set (if $y_i = y_j$) or second set (if $\hat{y}_i = \hat{y}_j$) [64].

2.5.5 Spearman's Rank Correlation Coefficient

The second metric to evaluate relative rankings among the two sets is Spearman's rank correlation coefficient (also called Spearman's rho), and is calculated using the formula;

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(y_i^r - \hat{y}_i^r)^2}{n(n^2 - 1)} \tag{27}$$

In equation (29), y_i^r and $\hat{y}_i^r$ are the ranks of ith values in the first and second set respectively and n is the number of elements. These ranks are calculated by ordering each set of values separately and assigning a number to each of the values incremented one by one starting from 1 and ending at n. If more than one value is the same, the average of their ranks is used instead. This is similar to calculating Pearson correlation coefficient using the relative ranks of the two sets of values [65].

All of the above metrics (25), (26), (27) report correlation on a scale of -1 to 1 inclusive. A value of 1 means perfect correlation, -1 means perfect negative correlation and 0 means no correlation.

2.6 Video Game Camera Control and Computational Aesthetics

In the video game industry, the game camera is an essential part of the equation for three dimensional games. It renders the scene by transforming the three dimensional position of the meshes through world, view and projection matrices. This process uses various texture, color and position data related to the vertices and processes them in a rendering pipeline on the GPU, resulting in a two dimensional image on the screen. Once the camera's position is determined these calculations do not relate to the scope of this work. On the other hand there are various ways of positioning the virtual camera in the virtual scene and these constitute what information is transferred to the user and how [66].

One of the commonly used alternatives is a tracking camera which follows the game's "avatar". A regular third person camera follows the controlled player's rotation and position in regard to some constraints and dollies behind the virtual character. The avatar is generally displayed from the back and a little above (and hence the name third-person). In various different modes, the camera can always look at the direction the player is heading at that instant or the camera can rotate smoothly following this direction to reduce motion sickness. Another alternative is rotating the camera as soon as the player stops moving while following its position continuously. This final kind of a camera always keeps the player in the screen but not guaranteed to display the section of the 3D virtual environment that is intended to be viewed (headed) by the player and generally used in game environments without too many threats [67]. As an advantage, this kind of a loose tracking makes it possible to see the avatar's front side for some time, instead of continuously displaying it from the back. An example third person view is given in Figure 9.

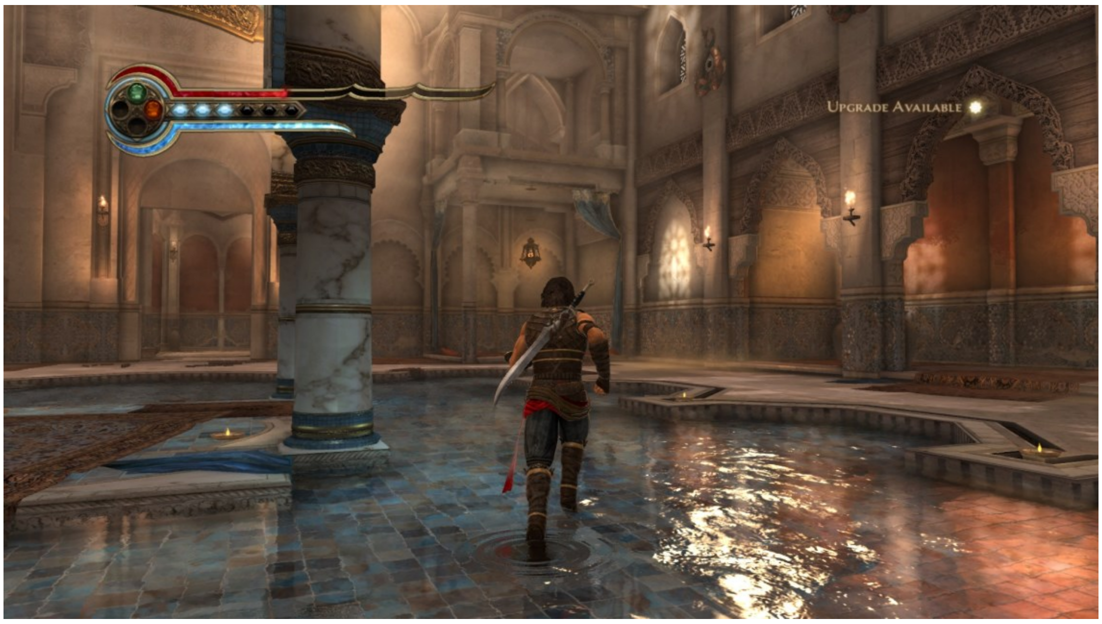

Figure 9: Prince of Persia: The Forgotten Sands as an example third person view

The virtual cameras are not only used during gameplay but also used for cut-scenes and other narrative elements. These kinds of camera positioning and motion are generally difficult to design and realize. Steven Mark Drucker [68] in his Ph.D. dissertation developed methods to determine camera position using constrained optimization and path planning. His virtual camera system was capable of introducing accepted cinematographic techniques to real-time rendered graphics using the various rendered element's positions in the scene and on the two dimensional projection of the camera. Another similar work was done by Tomlinson et al. [69], in which the camera position, and various lights' intensity and color was changed according to the emotions of the virtual three dimensional characters in the scene, called a "behaviour-based autonomous cinematography system". By calculating various parameters and weighting them across the scene, the virtual camera was able to determine which character to keep in the shot and the shooting style. It was possible to automatically decide on close-ups and the angle of the camera, depicting a character more dangerous by a low angle shot or frightened with a high angle shot.

The motion of the camera was controlled by adjusting the smoothing factors, giving the shot an angry or happy look. Furthermore camera transitions such as "cut" and "whip-pan" were also implemented, the former being an instantaneous camera transition and latter being a quick sweep to the new position. They have also implemented a ray-cast based occlusion detection to utilize a different camera position when the target is occluded by the other objects in the scene. They further anticipated camera systems that can modify themselves to the taste of the viewer and that can learn [69]. Although these are considered computational aesthetics methods, as they aim to enhance aesthetics trough computational methods, none of these methods utilized the actual rendered image to predict the aesthetic quality of the shot. As stated in [66], compositional properties of the displayed images are difficult to characterize algorithmically and accurate placement of the camera is critical in order to adjust these properties. Christie et al. [66] further stated that compositional elements such as lines and simple shapes were neglected although a wide variety of techniques on camera planning were implemented.

Further examples of computational aesthetics in video games, although not directly related to the visual aesthetics of the real-time displayed content, include [70] in which they experimented on a personalized content creation method considering visual aesthetics of the space-ships in the game. Introducing a set of initial space-ship models to the online system, newer space-ship models were evolved based on fitness functions adapted to the users' visual likings through neuro-evolutionary constrained optimization. The players of the game were to choose the designs they liked from the initial space-ship set and their visual preferences were transformed into personalized visual aesthetic models. Alternatively in [71], Shaker et al. used gameplay data collected online to generate a computational model of the relation between a two dimensional platform game's level design and player experience. Artificial neural networks were configured using artificial evolution and forward feature selection was used to determine the sequential features that result in a higher predictive performance. They have extracted features relating to engagement, frustration and challenge and related game aesthetics to these features.

CHAPTER 3

PROPOSED APPROACH

In this chapter, the methodology that is used to apply computational aesthetics on video game camera direction in an efficient manner is introduced. To explain the current approach, the process is summarized in Figure 10. First, a dataset appropriate for the task is collected and the images are pre-processed. Before training the machine learning algorithms, data is separated into two parts, leaving a separate test set, and remaining images are used in the learning tasks using various machine learning algorithms. These learning algorithms are initially selected considering their out-of-the-box strength in prediction and their ability to rank features with respect to relative importance. The best performing algorithm is determined using a validation sub-set in the training partition. Some of the commonly used features relating to aesthetic inference in the literature were collected, considering various aesthetic subclasses of properties and are improved to better suit the needs. Calculation times of all these features are measured to be able to select the ones with less computational cost.

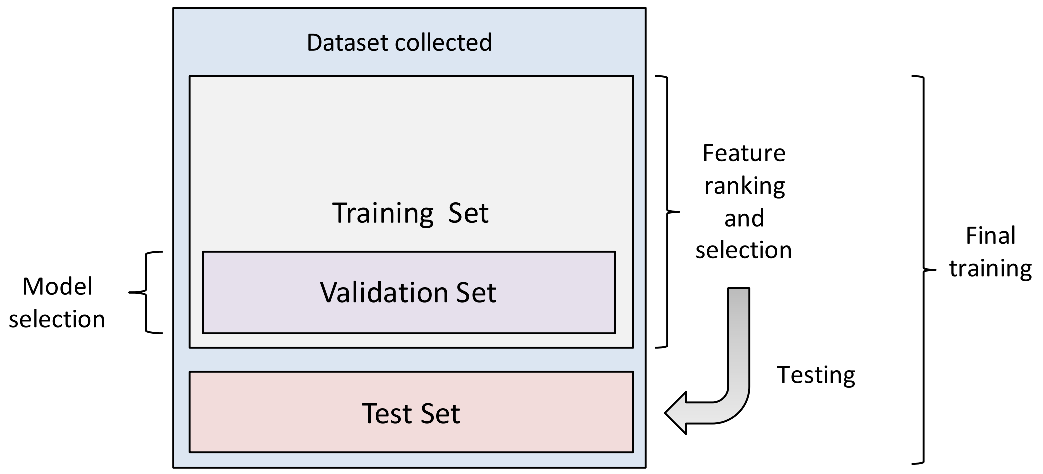

Figure 10: The proposed machine learning setup

The best model selected by validation is used to rank the features using the whole training set and a feature selection, involving the run-times and rankings of the features, is performed. The performance of the selected model and features is tested on the separate test set. And at the end, the final model is trained on the whole dataset using the selected model and features to be used in the video game application.

3.1 Dataset Acquisition

There are a few publicly available datasets to be used in computational aesthetics ([14], [25]) but some of them are not appropriate for the problem at hand. They either have a limited number of mixed category images or do not have the image scores (only have high and low classes). The AVA dataset [48] on the other hand, has many images to choose from (more than 250k images), it includes all the vote counts given to each image and has semantic tags (66 in total) for many of the images. It is a highly manageable dataset and is adopted for this work.

The dataset was shared as a list of image IDs acquired from dpChallenge web site [48] and these were downloaded with a simple crawler software. In Table 1, the structure of the dataset is exemplified. Each image in the dataset has an image ID tabulated in the first row, and at most two semantic tags presented in the second row. The numbers of votes corresponding to scores (selected among 1 to 10) are listed in the third row. The weighted averages of these ratings for each image are calculated to be used as the ground truth labels, tabulated as "Score" (fourth row). Also for each image in the dataset, a score variance is also calculated (fifth row). These should not be confused by the variance calculated on the ground truth labels of the whole dataset (to calculate variance reduction metric as explained in section 2.5.2) in the following sections. The variance of the individual image's votes will always be called "*per-image score variance*".

Table 1: AVA dataset structure

Image ID	**Semantic Tags**	**Vote counts 1-10**	***Score***	***Score Var.***
953619	Abstract, Macro	0, 1, 5, 17, 38, 36, 15, 6, 5, 1	5.637	2.022
953958	Abstract, B&W	10, 7, 15, 26, 26, 21, 10, 8, 1, 2	4.698	3.941
…	…	…	…	…
…	…	…	…	…

When using this dataset, it is possible to improve consensus on the aesthetic values of the selected images using a *per-image score variance* bound. In this work, since regression on human ratings is inherently a tough job due to the possible noise introduced in the voting process, only the images with a *per-image score variance* below 2.8 and voted by at least 150 people are used.

It is more challenging to predict aesthetic scores of mixed categories; therefore image dataset is further diminished considering the actual application into account, which is rendered outdoor landscape scenes. Furthermore, the given scores may not be completely due to the aesthetic properties and rather the semantic content of subject photographs. The aim here is to capture as much aesthetic properties as possible to further relieve the computational cost. By eliminating the subject images and selecting only landscape images, it is possible to further reduce semantic cues in the photographs and find scores that better relate to the aesthetic quality of landscape scenes. On the other hand, it is not completely possible to get rid of all semantic content since there may still be induced meanings in the landscape photographs by metaphor or usage of a subject.

Rather than dealing with photographic style or learning various aesthetic styles from these images, it is required to train the system to learn the aesthetics of landscape imagery. Selecting mainly the images having the "landscape" tag and excluding images having any of the tags listed in Table 2, 8515 images were selected out of this 250k image dataset.

Table 2: AVA tags excluded from image selection.

Abstract	Water
Cityscape	Studio
Fashion	Political
Family	Advertisement
Humorous	Persuasive
Interior	Digital Art
Sky	Seascapes
Sports	Traditional Art
Urban	Diptych / Triptych
Vintage	Floral
Emotive	Transportation
Performance	Food and Drink
Candid	Science and Technology
Portraiture	Wedding
Still life	Astrophotography
Animals	Military
Architecture	History
Black and white	Infrared
Macro	Self Portrait
Travel	Textures
Action	Children
Photojournalism	Blur
Nude	Photo-Impressionism

The remaining tags (out of 66 available tags) are "Snapshot", "Nature", "Rural", "Panoramic", "HDR" (high dynamic range), "Camera Phones" and "Analog". These tags are not excluded from the selection since nature and rural images will still fulfil our requirements and other tags introduce a wider variety of photography techniques and devices to the dataset. After applying the semantic tag filter and vote bound to the original dataset, per-image variance versus image count is plotted in Figure 11. The images with a variance greater than 2.8 are eliminated out of this subset and this threshold is represented with a dashed line in the same figure. The mean of the ground truth scores of the final selected set was 5.518 with a ground truth score variance of 0.499.

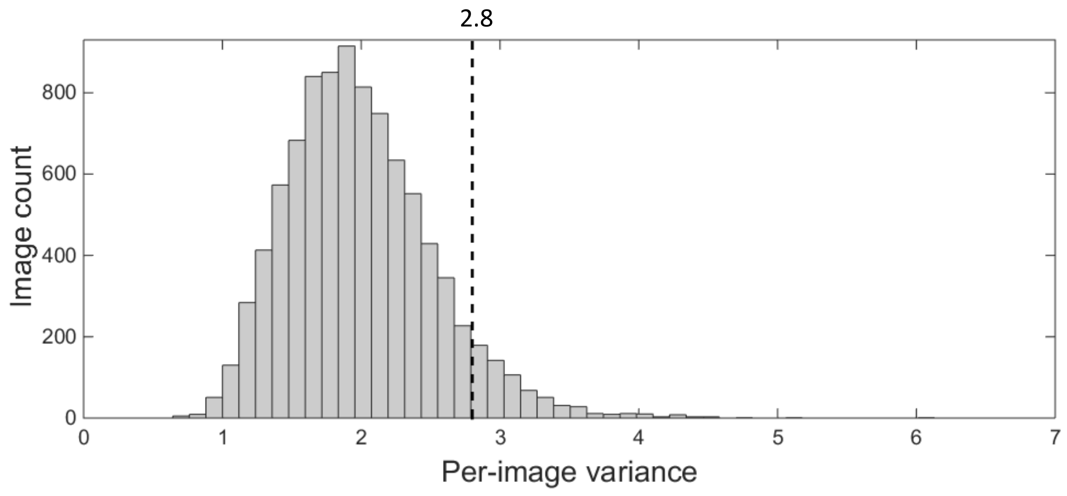

Figure 11: Per-image variance distribution for landscape images

Additionally images with extreme scores and variance are also exemplified in Figure 12 to Figure 15 with their vote distributions, average score and per-image variance. Similar to the above figure, these images are also selected after applying the semantic tag and vote count filter.

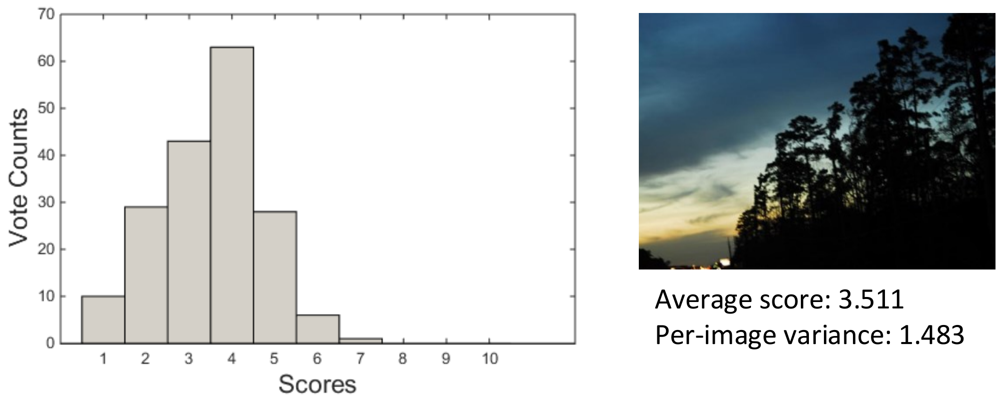

Figure 12: Example low quality, low variance image and its vote distribution

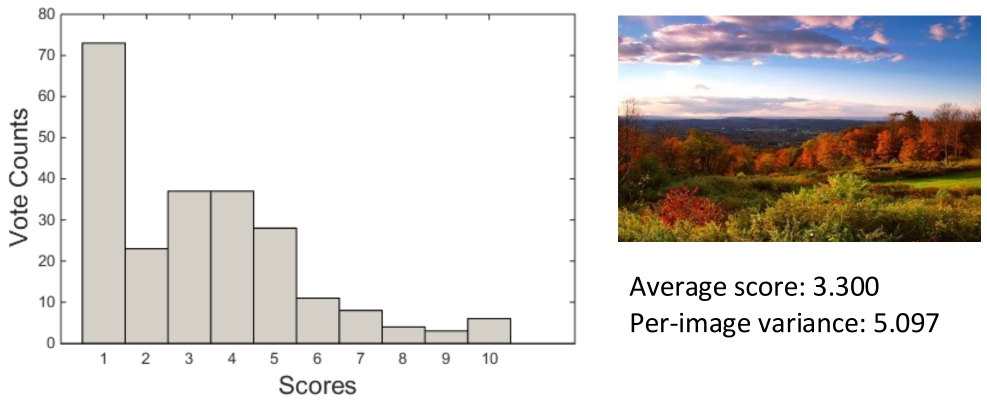

Figure 13: Example low quality, high variance image and its vote distribution

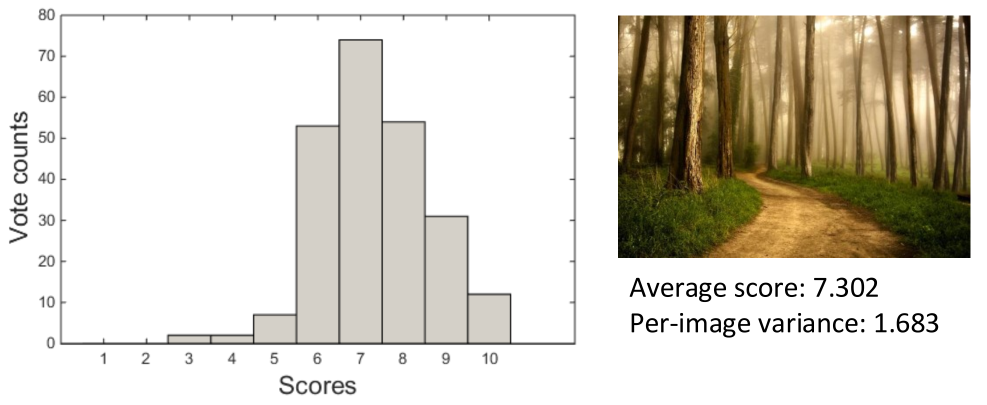

Figure 14: Example high quality, low variance image and its vote distribution

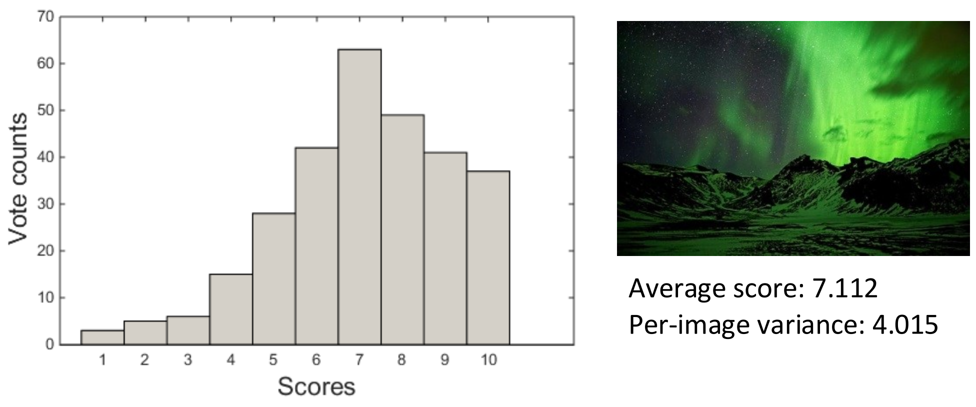

Figure 15: Example high quality, high variance image and its vote distribution

The semantic tags in the AVA dataset [48] are not always perfect and sometimes there are outlier images with other semantic meanings and unrelated objects. For example in the photograph shown in Figure 16 (ground truth score: 6.325), there are houses, water and a lonely man, which is not the exact type of photograph desired in the dataset. On the other hand, it is expected that the ground truth score of these images still somewhat relates to basic aesthetic properties such as color distributions, contrast etc. and they are not completely ineffective.

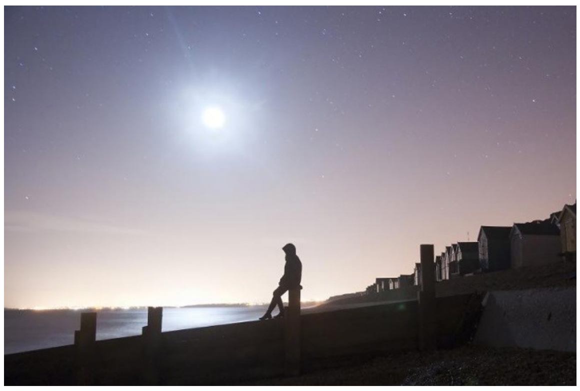

Figure 16: A photograph depicting a man sitting at night

From this set of selected landscape photographs, four examples having higher and four examples having lower scores are shown in Figure 17. Below each of these photographs, their ground truth scores are tabulated and they are ordered with respect to these scores, the highest quality image being on the upper left.

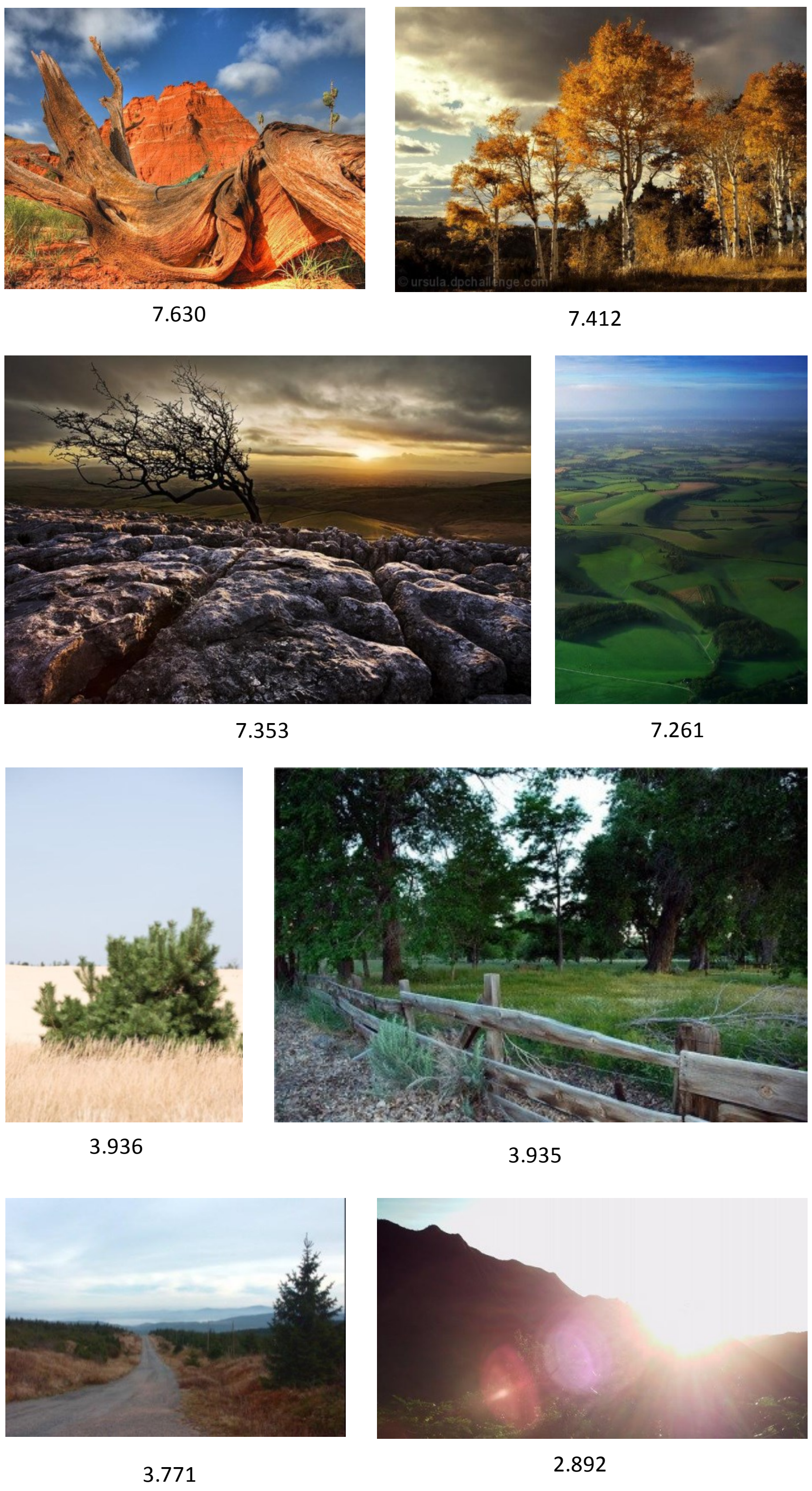

Figure 17: Example high and low quality photographs with their scores

3.2 Pre-processing

In this study, the learning algorithm serves a very specific purpose so some pre-processing is necessary to improve the final performance and predictability of the model.

Borders absolutely serve an aesthetic purpose and add to the value of an art piece [46] but also interfere with any edge detection component heavily and for the computer it is hard to differentiate between real edges and border-related edges. In fact the learning algorithm may learn the borders and map scores relatively, but this would be a risky decision and might introduce noise in the data. Furthermore, in the final aesthetic improvement software, it will not be possible (or rational) to use frames or borders in the output to modify aesthetic value.

Although there may be a relation between borders and public scores of images, it is an ultimately tough job to make sure if this relation is due to high (or low) quality photographers preferring borders or raters have a bias towards framed images. Furthermore, this relation may actually be due to the used border color(s)/style and even the web site layout and colors at the time of rating (frames may also be used to direct attention to the photograph from the website itself). Humans may prefer having a border around the subject, to be able to better differentiate it from its surroundings but the computer will already have the full image data during training. Furthermore, for template-based features, templates would also be affected heavily from borders.

Considering the difficulties involved, it is better try to capture as much image native data as possible. There is already possible noise sources including watermarks, multiple images stitched together with extreme framing (even without any aesthetic consideration). A border removal algorithm similar to the ones utilized by [20], [40] and [46] is applied to all of the images used for learning and testing. Although the exact method used was not disclosed in their publications, an alternative method is developed inspired from them.

It is not a trivial task to detect borders with perfect accuracy without sacrificing some image data. Sometimes it is not easy even for a human to say if there is a border or not. Some photographs can picture a scene with a frame like element in it. Computer border removal with as low a fail rate as possible is applied using a combination of canny edge detection and color counting. Still it may recognize false frames on perfectly centered subjects with very symmetrical vertical / horizontal lines or will not detect not so confident borders. Sometimes, images - especially with texts in borders - were not cropped perfectly. The CUHK dataset [25] also have border removal artifacts and has similar irregularities.

The algorithm detects three main kinds of borders, which are observed frequently in the selected dataset. First, the images with (almost) symmetrical borders on all four sides; second, images with borders not equally placed on four sides and finally images with single symmetrical borders (i.e. up and down or left and right). When the borders are not equally positioned, almost always the bottom border is higher (closer to the image center). An example image with symmetrical borders was shown in Figure 18 together with its border removed version.

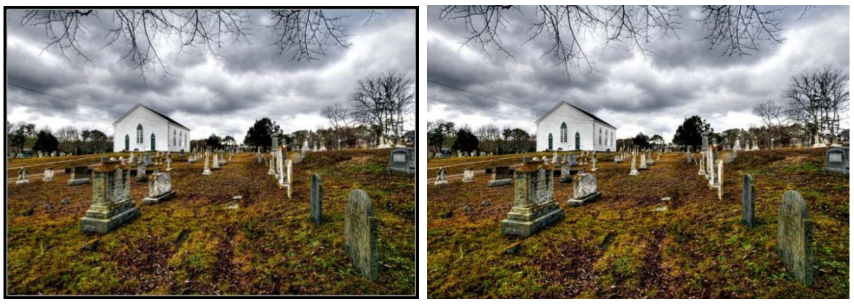
Figure 18: An example bordered image and its border removed version.

To summarize the applied border removal process; the edges in the images are extracted using Canny edge detection. Then the binary edge image representing edges is projected on the horizontal and vertical axes. Starting from outer portions of the image, the algorithm moves inwards from each side (top, bottom, left and right) one row/column at a time, as long as the color variations of that row/column have a low standard deviation. When the projected edge counts exceeds a threshold value, that location for the corresponding side is considered a candidate border and recorded. If the standard deviation exceeds the threshold, no further candidates are collected in that side of the image.

After this process is repeated for all sides, the candidate borders' distance to image edges on each side are compared. If all of them are close to each other within a given margin, a four-side symmetrical border is detected. Considering the border types introduced above, this process is repeated with various thresholds looking for different types of borders and checking their validity. Once a border is detected, the image is cropped using the matching candidate border distances. The failed attempts are small in ratio (a few in thousands) and the benefit will possibly overcome this added noise. Images are re-saved with lossless compression to maintain their quality and prevent re-introducing extra compression artefacts.

Additionally all images are scaled to improve feature extraction times before training, make the images in the dataset consistent and limit the calculation times in the real-time application. This scaling is done to have approximately 200k pixels as suggested in [40], without changing the aspect ratio (within a pixel error). Instead of scaling the image into a fixed size square, this method would preserve spatial details in horizontal and vertical directions relatively better. For example, an input image having a wide ratio (landscape aspect ratio) may have more information on horizontal axis and vice versa (more detail in vertical axis on a portrait aspect ratio). As it is apparent, it is not always possible to have 200k pixels for every image due to the original image aspect ratio constraints. There may be few extra pixels since an integer line/row count is required [40]. To handle such cases, although the difference is small, pixel count normalization was applied to some of the features that are directly related to pixel counts. For a commonly used screen resolution of 1024 by 768 (about 786k pixels), reducing the pixel count to almost ¼ would reduce the computation times significantly. This scaling will affect the high frequency content of the input images but both the ML model and the final application will be using images of the same size and they will be consistent.

This scaling process is bypassed when calculating templates [41] for the HSV images to be able to capture the full detail of the input images and generate better quality templates. The images were already being resized down to 100 by 100 pixels when calculating the individual L_1 distance features. At this point, resizing is eliminated only to increase the template details. On the other hand, if the templates were generated on the scaled image data, there might be some accumulation of impurities.

3.3 Feature Short-Listing

Some of the existing features in the literature are collected with various aesthetic considerations. Some of them are improved considering the needs and some of them are combined. The same feature names used in this section will be used throughout the remaining part of the thesis. Most of the features in this section were introduced in section 2.4 and exact sections are referenced below. Although some of the features are not extracted in the same way as they are introduced in the literature (for example they are extracted on different image channels), the basic principles apply. Other than these listed features, saliency analysis using Itti's saliency [33] in which the images were analyzed using color, intensity and direction filters, Hough transform to detect lines and image segmentation as in [12] are implemented. These features will be eliminated on the performance observations in section 4.3 and not discussed further since they are not included in the final model.

The 55 features are grouped under eight aesthetic classes as; composition, line composition, brightness, color simplicity, hue distribution, saturation, sharpness, and texture. These classes are built so that feature selection process will have a variety of features covering a wide range of properties. Below, the selected features under these groups are summarized. All of these features are used as they were implemented in the literature. It is stated whenever a different approach is followed. Many of the features listed here are implemented using only the HSV color space's brightness channel to minimize different color conversion operations whenever applicable. As a side note, GLCM correlation metrics were giving indefinite results on hue and saturation channels (constant hue image on black and white images in the dataset) and was excluded from the model early-on.

3.3.1 Composition Features

Image composition has a significant effect on the visual aesthetics especially of landscape imagery [72]. Leaving out segmentation and saliency analysis as will be explained in section 4.3 due to their computational cost, the remaining compositional features must be strong and efficient. Composition features are selected as the features representing the spatial distribution of different regions having different color and brightness in an image. Also the distribution of edges and preferred edge locations in the frame are considered in this subset since the edge information is related to the salient regions in an image [41]. The main driving force behind this subset is describing spatial distribution of lower level features which were grouped as local features in section 2.1.

There are three main feature groups represented in this class, namely template based composition features of [41], squared differences of global features between the rule of thirds region and outer region influenced by [12], [38] and [26], and edge concentration area feature [20]. All of these features relate to the spatial distribution of color, saturation, brightness and edge information in the image.

The template based features explained in section 2.4.6 are utilized as in [41] on H, S, V, H+S+V channels and their Laplacian filtered edge images. They were named in the same order, *L_1 Distance Hue, L_1 Distance Saturation, L_1 Distance Brightness, L_1 Distance Summed, L_1 Distance Hue Edge, L_1 Distance Saturation Edge, L_1 Distance Brightness Edge,* and *L1 Distance Summed Edge.*

These template based features (section 2.4.6) require a selection of high and low quality set of photographs. In the original paper [41], the CUHK dataset [25] was utilized and being a high-low quality separated dataset, it is not clear how much score difference exists among these groups. In this implementation, the high and low quality images were selected using the ground truth score distance of the training images to their mean, eliminating the images around the mean. Images that have scores higher than $\mu + \delta$ and lower than $\mu - \delta$ were selected as the "high" and "low" groups, where μ is the mean ground truth score of the images and δ is the selection parameter. In the experiments δ is chosen to be 1.3. The applied procedure is summarized in Figure 19.

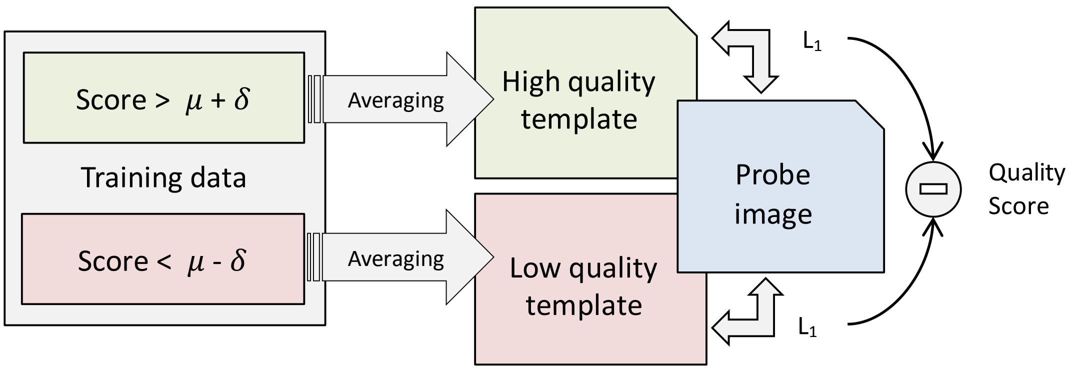

Figure 19: L_1 distance template comparison features

L_1 Distance Improved Hue: This feature is an improvement over the original *L_1 Distance Hue* feature. When calculating the averaged templates for the high and low classes, the ith pixel of the template image is instead calculated by;

$$T_i = atan2\left(\frac{1}{N}\sum_k \sin(V_i^k), \quad \frac{1}{N}\sum_k \cos(V_i^k)\right) \tag{28}$$

In equation (28) N is the number of images used to calculate the template, V_i^k is the ith pixel of image k and atan2 is the four-quadrant inverse tangent function. Since the hue is a cyclic function, it is aimed to approximate average of the hue values by converting them into vectors, finding the average of these vectors and converting back to an angle (hue value) using atan2 function. An example vector based hue average calculation is illustrated in Figure 20. H_1 and H_2 are the two hue values to be averaged, converted into vectors. They are summed to yield the vector shown in green and the resulting angle α is calculated.

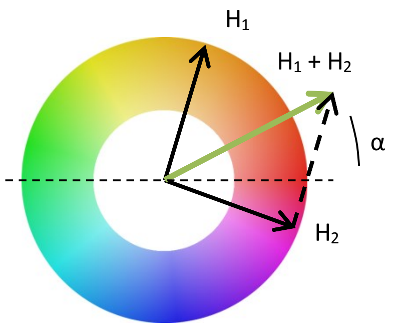

Figure 20: Vector hue average

Machajdik and Hanbury [40] also used vector based average hue in their work in a similar manner. Since the templates are calculated beforehand to be compared against probe images, the wrong values would accumulate without a proper hue average function. Additionally, when the image quality (Q) is being evaluated, equation (29) is used instead of equation (5) to calculate the real hue distance between the two images.

$$Q = \sum_{i \in I} \|L_i - T_i^L\| - \sum_{i \in I} \|L_i - T_i^H\| \tag{29}$$

Here, $\|\cdot\|$ is the arc-length distance between the hue values of two images, L_i is the ith pixel in of the image in question, T_i^L is the ith pixel value of the low quality template and T_i^H is the ith pixel value of the high quality template.

Edge Energy Concentration Area: The area of the bounding box encapsulating 98% of edge energy (section 2.4.5). This feature was originally deduced by looking at the templates extracted from high and low quality images [20] and it measures edge distribution and therefore is categorized together with template based features.

Other composition features are calculated using some of the global image statistics calculated on the center region of the input image. Datta et al. [12] and Dhar et al. [43] determined the inner rule of thirds region by dividing the frame into 16 equal parts and selecting the 4 center sections covering the power points as illustrated in Figure 21.

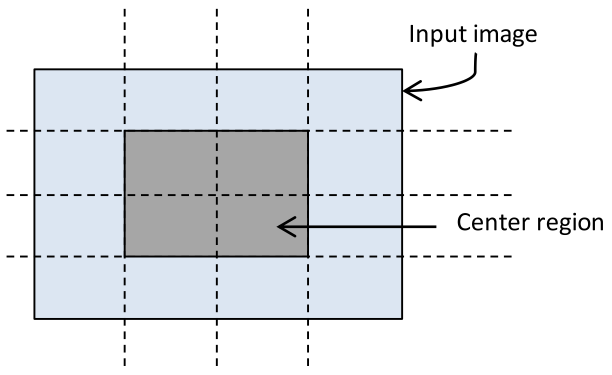

Figure 21: Determining the center region

The same division is utilized in this study when the features related to the center regions are to be extracted. Squared differences of the values in this center region and outer region are calculated as in [38], to capture the contrast between the possible subjects in the rule of thirds area and the remaining "background" regions. These features are center region squared differences of; *Average Hue, Average HSV Saturation, Average Intensity.* In the following sections, these features will be summarized as *Diff("Feature Name").*

3.3.2 Line Composition Features

Lines and gradients have a strong effect on visual aesthetics as introduced in section 2.1.3. Line composition feature group was separated from the composition features since the regular composition features deal only with the spatial distribution of various low-level properties. Line composition on the other hand, also deals with the directions of the lines and gradients in an image which is an important aesthetic consideration [11]. Similar to segmentation and saliency analysis, Hough transform to detect lines in the image was also found inefficient due to its computational cost as to be later explained in section 4.3 and alternative features that can discriminate lines and gradients in the image are essential. Therefore, the features that utilize image gradients and edge histograms are introduced instead of calculating the Hough transform.

Furthermore there may be differences between the rule of thirds inner region and the outer region of the image. For example, differences in the directions of tree branches inside and outside the inner thirds region may influence aesthetics. Center region squared difference features are also employed to cover these effects. The collected features are;

Tamura Directionality: As explained in section 2.4.15

Diff(Tamura Directionality): Squared difference of Tamura directionality between the inner third and outer region.

Sobel Edge Histogram (15 dimensions): Calculated as was explained in section 2.4.15

Diff(Sobel Edge Histogram) (15 dimensions): Squared difference of *Sobel Edge Histogram* between the center and outer region.

3.3.3 Texture Features

These features measure the textural properties of the entire image. Some of the GLCM and wavelet features are grouped together for ease of reading in the following chapters.

Tamura Coarseness: Calculated over the entire image's HSV brightness channel as described in section 2.4.15.

GLCM Hue Features (3 dimensions): Energy, homogeneity and contrast metrics are calculated over the entire image, on the hue channel (see section 2.4.4).

GLCM Saturation Features (3 dimensions): Energy, homogeneity and contrast metrics are calculated over the entire image, on the saturation channel (see section 2.4.4).

GLCM Brightness Features (4 dimensions): Energy, correlation, homogeneity and contrast metrics are calculated over the entire image, on the hue channel (see section 2.4.4).

Wavelet Features (12 dimensions): 12 features based on wavelet coefficients explained in section 2.4.3.

Global Texture: Calculated on H, S, V and H+S+V channels and respective edge images as described in section 2.4.16. The features in this set are named *Absolute Sum Hue, Absolute Sum Saturation, Absolute Sum Brightness, Absolute Sum Summed, Absolute Sum Hue Edge, Absolute Sum Saturation Edge, Absolute Sum Brightness Edge, Absolute Sum Summed Edge* in the order given above.

Additionally, to resolve the cyclic nature of the hue channel, *Absolute Sum Improved Hue* feature is calculated using equation (30).

$$Improved\ Hue = \sum_{k=1}^{5} \sum_{i \in V^k} \left\| V_i^k - V_{i+1}^k \right\| + \sum_{k=1}^{5} \sum_{i \in S^k} \left\| H_i^k - H_{i+1}^k \right\| \tag{30}$$

Here, ||·|| is the arc-length distance between the hue values and other parameters are the same as equation (30).

3.3.4 Hue Distribution Features

These features measure the global color distribution of the image and hue relations on the color wheel.

Color Quality: 5 dominant colors are found and a KNN search is performed among high and low quality images as explained in section 2.4.7.

Color quality features (section 2.4.7) require a selection of high and low quality set of photographs and a k-nearest neighborhood of K. In the original paper [41], only a high-low quality separated dataset was used. Furthermore, the parameter K used for dominant color KNN search was not disclosed either. In this implementation, the high and low quality images are selected using the ground truth score distance of the training images to their mean (μ) similar to the method explained in section 3.3.1 (see Figure 22). In the experiments K was chosen to be 10 and δ was 1.3. Furthermore, for the color quality feature introduced in section 2.4.7, the α parameter –again not being disclosed in the original paper- is selected as 0.5.

Real Color Quality: After building the KNN space, instead of finding the difference between the high and low quality color sets as in equation (7), the ground truth scores of the neighbors determined by KNN is averaged as a color quality feature. With this approach, calculating a finer color quality score is aimed, as it would serve regression analysis better. In the original work [41], only classification was intended. This process was summarized in Figure 22.

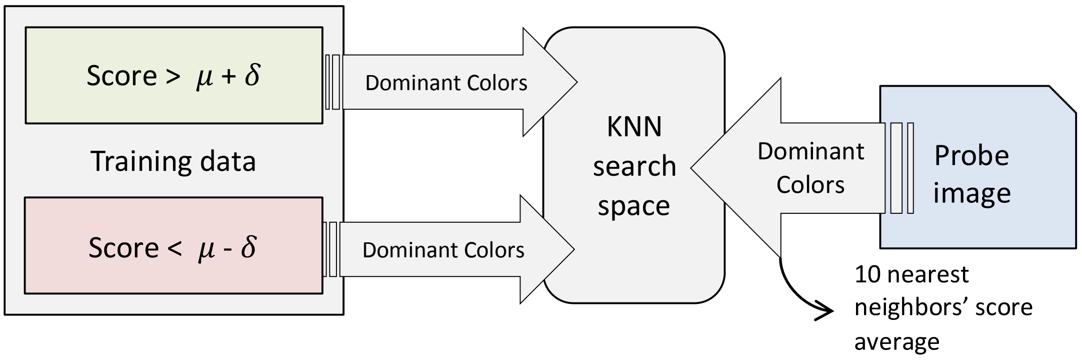

Figure 22: Calculating color quality for a probe image

Adjusted Hue Histogram (16 dimensions): 16 bin hue histogram shifted such that highest count bin moved to the initial bin as explained in section 2.4.11.

Convolved Template Fit Values (7 dimensions): Hue templates in Figure 4 convolved with the image hue histogram using each independent fit value as a feature (section 2.4.10).

Hue Template Convolution: Only the best fitting convolved template used as a categorical predictor (section 2.4.10).

Template Fit Values (7 dimensions): Hue templates are fit with the customized fitting functions explained in section 2.4.10. In addition to using the best fitting, strictest template as a feature as in [26], each of the 7 values are also used as features to generate this feature in a similar approach to [10].

Hue Template: Only the best fitting, strictest template found when calculating *Template Fit Values* is used as a categorical predictor (section 2.4.10).

Hue Contrast: The maximum distance between hues in the image as explained in section 2.4.9.

These final two color related features do not exactly measure hue distribution on the color wheel but still they are categorized under hue distribution since they capture the color distribution on the image. And above features possibly include the following ones since they all also deal with the hue counts.

Average Hue: A regular average of hue channel values of the image using the averaging method in section 2.4.1.

Most Frequent Hue Pixel Count: The pixels belonging to the hue bin with the highest count as explained in section 2.4.8.

3.3.5 Color Simplicity

These features are related to the number of colors used in an image. Although the features included in the above section capture the color distribution, they do not compute the number of colors used in an explicit way, which is an important aesthetic consideration. Therefore these two features are separately considered.

Simplicity Using 4096 bin RGB: Non-empty bins of 4096 bin RGB histogram (section 2.4.8).

Hue Count: Number of bins higher than a threshold calculated on the filtered hue histogram (section 2.4.8).

3.3.6 Saturation Feature

This feature is the only feature that describes saturation properties of the image.

Average HSV Saturation: Average of saturation channel S in HSV is calculated over the entire image (Section 2.4.1).

3.3.7 Brightness Related Features

In almost all of the previous works, brightness features were used to describe the lighting and exposure properties of images. Here, a few were selected.

Average Intensity: Average of brightness values (section 2.4.1).

Maximum Intensity, Minimum Intensity: The maximum and minimum values of brightness values (section 2.4.1).

Intensity RMS Contrast: RMS contrast (or standard deviation) of brightness values (section 2.4.1).

Tamura Contrast: Tamura contrast of the whole image, using the brightness channel (section 2.4.15).

Intensity Center Mass: Width of the center portion of the brightness histograms (section 2.4.13).

Michelson Contrast: Michelson ratio calculated over the whole image, using *Maximum Intensity* and *Minimum Intensity* (section 2.4.1).

The features except *Average Intensity* explain the variations in the brightness information that resolve contrast properties and measure the distribution of brightness values on the histogram. *Maximum Intensity* and *Minimum Intensity* represent use of dynamic range [18] and relate to overall brightness and contrast of the image.

3.3.8 Sharpness Features

Although overall sharpness is not related to the use of depth of field, in landscape photography, generally a narrower aperture and a wide depth of field is used [27] to render both distant and closer objects clearly unless a special effect is targeted. Therefore it is expected that overall sharpness to be more effective than depth of field usage and only sharpness measures were incorporated.

FFT Blur Metric: Calculated as explained in section 2.4.12 on the HSV brightness channel.

Dark Channel Feature: Calculated over the whole image as explained in section 2.4.2 and utilized in [41]. This feature is expected to encode clarity and colorfulness information [25], but here it was grouped under sharpness features. This situation will be evaluated further in section 4.4.9.

3.3.9 General Features

These features (*Aspect Ratio* and *Image Size* as described in section 2.4.17) do not directly relate to any of the aesthetic primitives introduced in section 2.1 and describe dimensional properties of the image under consideration. These can still have an influence on the aesthetic quality; especially the aspect ratio of the medium induces some aesthetic value [12]. These are introduced in the analysis in order to remove any aesthetic bias introduced even though there will not be direct control over these parameters (screen resolution) in the final application.

3.4 Model Selection

The motivation of this work is relating images with aesthetic score predictions that will be compared among various alternative camera views and a regression machine learning algorithm is best suited to this task. As commonly implemented in the previous works, separating the images to high and low aesthetic quality classes would not be applicable to video game camera direction since the high quality camera locations would occur at specific locations, limiting the camera movement. Furthermore, finding these high quality camera locations requires a tedious search that is not feasible for a real time application.

Before applying the machine learning algorithms, data is separated into two parts, leaving out 10% of the images as a final test set. This set is never used in training or validation, and is used only for performance analysis after all experiments and modifications are complete. The remaining 90% of images are the basis for the learning tasks using various machine learning algorithms. Since the aim is reducing the final run-time of the predictive regression model, a measure of embedded feature importance is required to eliminate less effective features. Although a wrapper based feature selection would also be appropriate, due to the complex aesthetic interactions between features and interdependent run-times, an embedded approach is adopted.

In this respect, three powerful regression methods (namely least squares boosting, bagged tree ensembles and random forests) that are capable of predicting feature importance are utilized and they are trained to select the best performing method by validation. The reason behind selecting the best model lies at feature importance rankings and final performance. The ML model that can explain the underlying data better (having a better validation performance) is expected to predict feature importance better as it would be able to fit these features to the ground truth labels better. This selection is done by training each model on the training set by keeping 20% of the data as a hold-out validation set, training on the remaining samples and checking the performance on the validation set. This process is repeated 5 times for each model and the average validation performance is calculated. The 20% is selected and separated at random differently at each repetition.

When validating the models, template feature calculations (explained in section 3.3.1) are repeated for each training sub-set to prevent optimistic performance estimates. Extracting compositional templates using the whole training data would result in optimistic figures even if the images used for template generation was not used in validation. This is because the high quality and low quality images would be determined using the distance from the mean calculated on the whole training data.

Although bagging methods are known to reduce variance and perform better using higher numbers of features due to their bootstrap nature [73], the reduction in feature count in this case, makes it possible to run faster real time analysis/scoring and must be considered seriously.

The separate test set is used for evaluating final model performance. Without a separate test set, the predicted performance of the model would be influenced towards a more optimistic one and generalization might suffer. This is mainly due to various parameter adjustments and feature selection done on the course of the experiments which are all learned from this selected data sub set and thus influence the predicted performance metrics.

3.5 Feature Run-Times

In order to improve the final model's prediction run-times, time spent for feature extraction are measured to be able to select quicker features. Two different measurements are made, one with a coarser analysis and second with a more detailed investigation considering the possible common subset of calculations.

First, short-listed features are timed to have a general understanding of the required time to calculate them. Namely a coarser analysis is done to determine the slowest features only considering the time required generating the base information for them. At this point, the features with highest computational requirements are eliminated with a maximum-time threshold.

As the second run-time analysis, a more detailed measurement is made also considering the calculations that are common among various features such initial transformations. These detailed run-times are used in the actual feature selection process in the next section.

3.6 Feature Selection

In the final model, as small a feature count as possible is targeted due to computational restrictions and therefore a feature elimination process is needed. The feature selection is principally based on the feature importance rankings of the learning algorithm chosen in the previous step. Predictor importance is calculated using whole training data in order to capture all of the details of currently selected training data. This process would of course over-fit the training data therefore final performance measures are examined using the previously separated independent test set. Since the results are being compared and feature selection is done on a given data set, the better features to be found still depend on this set. Better features (or feature importance) are "learned" in this process although the machine learning method itself does not seem to be learning directly. For example, a specific feature may be performing well on our specific dataset (and we are over-fitting) or vice-versa. Since the actual prediction performance of the model will be evaluated on a separate test set, the final performance will still be reported correctly. It is also important to note that, after measuring performance on the test set, the results are not used to select a better method, but to fulfill scientific curiosity and see relative performance of various methods.

Two alternative feature selection schemes targeting this problem are proposed. In the first one, which is a rather heuristic feature selection, the features are grouped under aesthetically coherent sub-groups and among these subsets, features are selected considering the different aspects they represent, their relative importance as reported by the learning algorithm, and the time required calculating them. Selection is applied on the features that are already known to be similar to each other using the knowledge on aesthetics and it is important to keep at least one feature out of each class. This selection method is named "computation-centric, aesthetics-aware selection". With this first approach, the minimum required features to describe aesthetics are determined using the background in aesthetics evaluation. The used aesthetic classes of features are as described in section 3.3.

In the second selection method, which is implemented rather out of scientific curiosity, the first feature subset is kept as a baseline for maximum time requirements. A computation budget is calculated and a new selection is made only considering relative feature importance of the features in a similar manner to [60], the difference being the learning algorithm. The computation-centric, aesthetics-aware selection would determine the minimum time required encoding a wide variety of aesthetic aspects that are known to be important in evaluation. Based on this, the second method is employed to measure the relative performance of a second feature set, depending solely on the relative importance metric (without taking aesthetics into consideration) in the same given time-box.

It is true that, features that are rather irrelevant on their own can make up good predictors when used together with other predictors [59]. Therefore when using the feature importance rankings, it is possible that some masking among features may occur and the overall effect of a specific feature can be influenced through feature interactions. On the other hand being on a computational budget, these rankings are followed as guidelines instead of definite rules (in order to not to select many features to describe an aesthetic primitive) assuming that every feature's rank can be influenced from other ones in a similar manner. If a feature's rank is higher than others, it is highly probable that it fits the underlying data, but this does not prove that it is a better feature. At this point, aesthetic considerations and run-time requirements gain importance.

Below, the general rules applied during the selection process are summarized. These are not exact rules followed, but instead the motivation of the process is emphasized. In section 4.4, specific details of why and how each feature is selected are given following the feature importance calculation.

When the highest ranking feature takes longer (more than twice as long) to compute and have another similarly ranked alternative (rank difference within ~5% of the overall variable count of 128), the ones with the lower computational times are selected trying to describe as much of the same properties. Eliminating the feature with the highest computational cost would not induce much performance loss as long as the ones with lower computation times will compensate for it.

If only a single feature ranks exceptionally well in a subset (rank difference of more than ~30% of feature count), it is selected. No matter how long the calculations take, there is no other metric to measure this quantity as effectively.

When features with more than one component (such as histograms) are in question, if any component ranks high, the whole values are incorporated into the model since the additional ones would not introduce much extra computational overhead as they are generally calculated by searching every value in the image. Additionally, in order to keep the integrity of the originally proposed coherent feature groups in the literature, if some of the components for these features rank high, the whole feature group is included (for example global texture features).

3.7 Application to a Real-Time Game Environment

After the feature selection process, the ML model is trained on the whole data (including training and test) to be applied on real-time rendered graphics. The aesthetic quality of the game's visuals can be improved using different approaches, adjusting different parameters and trying to follow generally accepted aesthetic rules given in section 2.1 must be the main starting point.

Considering the computational requirements of the problem at hand, instead of trying to optimize many parameters such as lighting/exposure, color filtering, field of view and depth of field, only the composition of the rendered scene is desired to be improved, knowing that composition is essential for landscape imagery [72].

To improve the composition, framing is intended to be changed since the elements in the virtual scene are relatively static (except for wind effect etc.) and it would not be possible to change their locations in a meaningful way. Furthermore, by interfering with the framing of the rendered graphics, differently lit and colored regions can be evaluated and it is possible to fulfill other aesthetics considerations at the same time. There are still many degrees of freedom to determine the camera positon, such as its three dimensional position, roll, yaw and pitch [68] illustrated in Figure 23, where the cone represent the camera's field of view. In this implementation, only the camera position and yaw axis is considered similar to a third person camera.

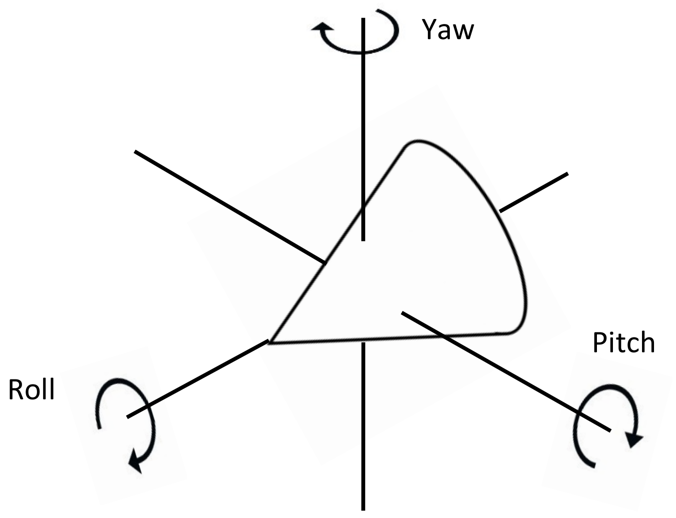

Figure 23: Symbolic camera and its three axis of rotation

First a 3D environment involving a terrain, grass, trees and other natural elements is built in the popular Unity game engine as seen in Figure 24.

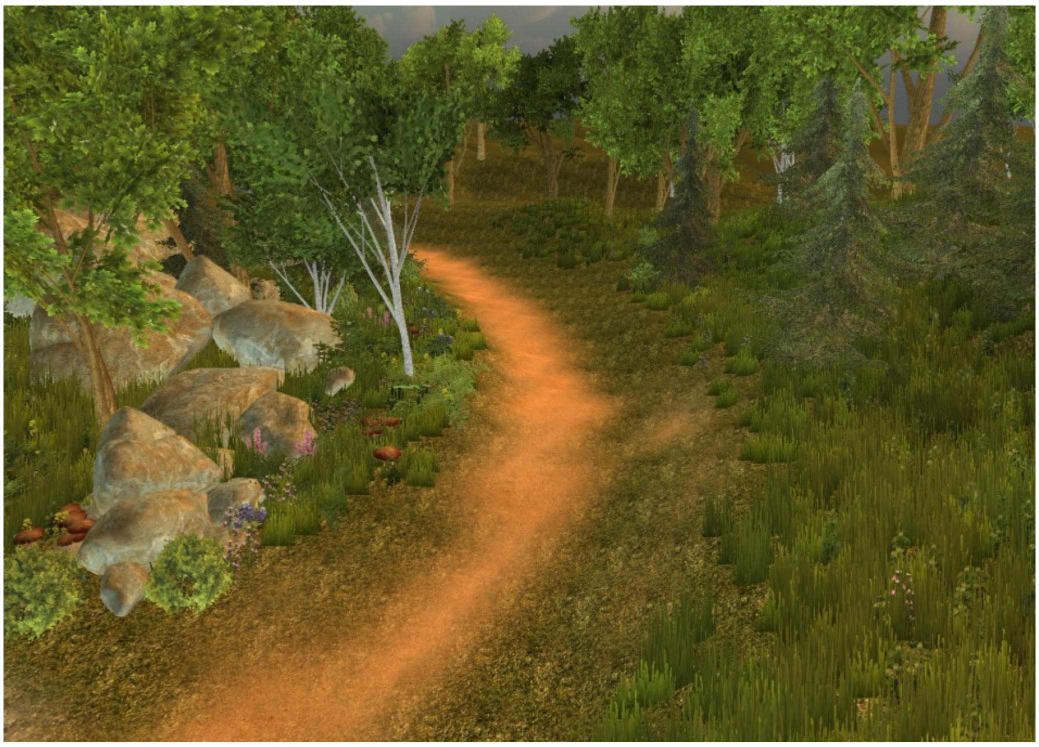

Figure 24: Example scene built using Unity game engine.

A standard third person camera setup is also incorporated that follows the controlled player’s position and rotation in a smoothed manner. This kind of a camera always keeps the player in the screen and rotates towards the direction headed by the virtual character. It is sometimes important to show the section of the 3D virtual environment that is wanted to be viewed by the player and this approach serves the purpose well when used in continuous rotation configuration. But this is not a critical consideration for a non-threatening game world and the camera’s rotation can be changed on a specific event such as the player becoming stationary [67]. On the other hand, a third person camera does not care whether the displayed scene is aesthetically pleasing or not.

Without completely discarding the player’s needs (the direction and location that is wanted to be seen), the classical third person camera is used as an aesthetic evaluation camera that always follows the virtual character’s viewpoint. But the image rendered on this camera is not used to display the scene to the user directly. Instead, it is called the “secondary camera” and is used for aesthetic evaluation of this alternative viewpoint. The actual displayed image on the screen (presented to the user) is rendered on a camera named “main camera”. This camera setup is depicted in Figure 25. The red arrow represents the avatar (or virtual character) and its heading direction. Two triangles represent the cameras with their field of view shown by the dashed lines. The secondary camera follows the player’s position and direction whereas the main camera renders the image to the screen.

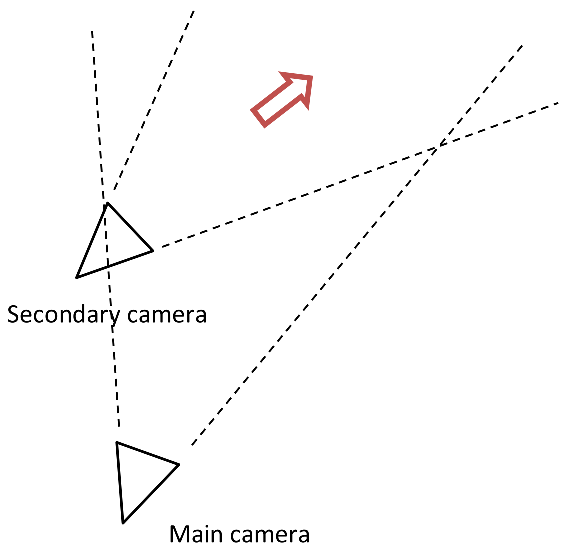

Figure 25: Camera setup

Additionally when using this main camera for presentation, a camera relative control mode is adopted in which the player controls the character relative to the camera. For example, when the user wants to go left, the avatar moves to the left with respect to the current view, whatever its original heading direction is.

In order to manage two rendering jobs simultaneously, a secondary thread (called the aesthetic evaluation thread) is launched in parallel to the "main thread" in Unity. Otherwise, the main camera's rendering operations would be interrupted by the intensive feature calculations. The main thread executes all the controls and mechanics of the game, render the actual image to be displayed, positions the secondary camera with respect to the virtual character (in a similar fashion to the third person camera) and communicates with the secondary thread to retrieve aesthetic evaluation information. All processes in the main thread should be executed real-time at the maximum frame rate. The summarized system diagram is given in Figure 26.

In Figure 26, the basic components of the real-time aesthetic camera direction system is illustrated. The Unity game engine runs two separate threads with inter-thread communication. The aesthetic evaluation thread calls MATLAB's [74] dlls that actually wrap the MATLAB scripts and interpret them as if they were running in a MATLAB workspace. The image from the Unity camera is transformed into a byte array and passed. The wrapper returns the aesthetic prediction for the input image as a real valued number.

In the secondary thread, the image of the secondary camera (that is positioned as if it is a third person camera – see Figure 25) is rendered on a surface on the GPU and image data is transferred to the MATLAB's interpreter. Once the feature calculations are complete, the machine learning algorithm generates its prediction and the main thread reads this score information from the aesthetics thread.

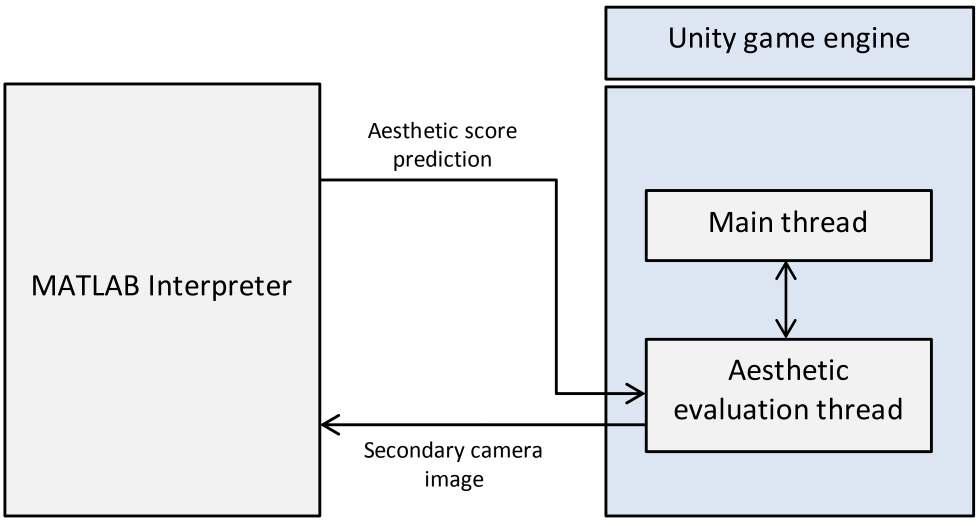

Figure 26: Real-time aesthetic evaluation system diagram

Initially the system is in an “unknown” state and the “initial” position of the main camera is used to render the scene (this position was selected as the third person camera position for this initial state). As the player moves the virtual character, secondary camera (aesthetic evaluation camera) continues to follow it. At the same time, aesthetic evaluation thread continues to evaluate the images captured by the secondary camera and predict the aesthetic scores. After receiving the first aesthetic score from the secondary evaluation thread, the new score is kept as a reference, which was initially set to zero. When a new aesthetic score that is higher than the previously recorded score is reported by the aesthetics thread, the main thread uses this new viewpoint as the “target” state of the main camera and smoothly translates and rotates it to this viewpoint from which the score prediction was made using the secondary camera image. This transition is similar to a “whip-pan” [69]. In the initial “unknown” state, the first reported score is always expected to be higher than the initially set value of zero and the camera moves to this new viewpoint immediately.

When the player wants to move to a position outside the current viewpoint, gets too far away from the main camera or becomes occluded (player is not visible anymore), the process restarts by moving the main camera to the actual third person camera location (“initial state”) and clearing the recorded score. When there is not a better viewpoint, the main camera behaves like a fixed camera. The process is summarized in Figure 27. The occlusion control is done via a simple ray-cast from the camera to the avatar’s middle position. If the cast ray intersects with a scene object, the player is assumed to be occluded similar to the method used in [69].

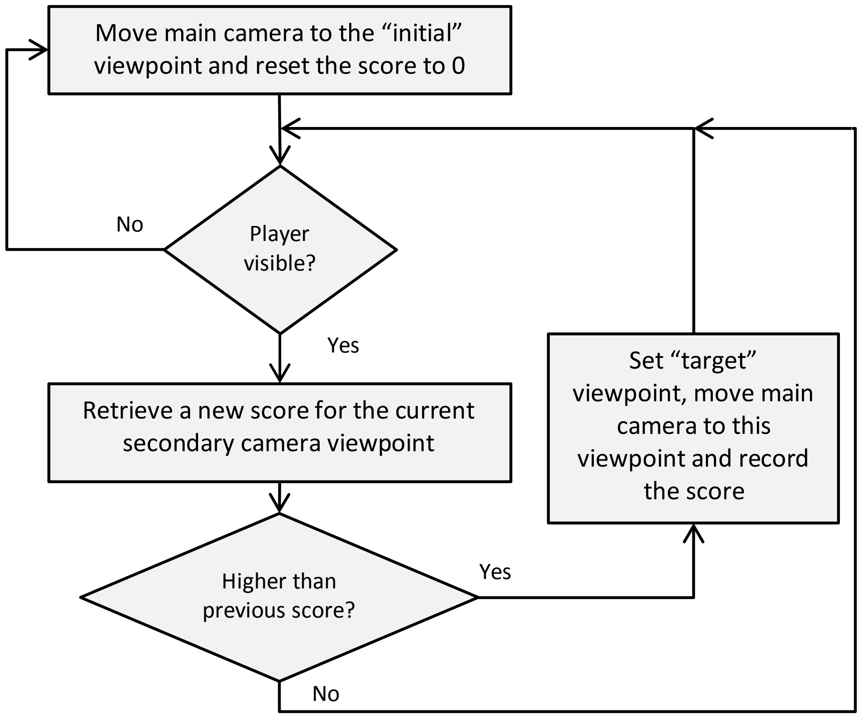
Move main camera to the “initial” viewpoint and reset the score to 0
Player visible?
No
Yes
Retrieve a new score for the current secondary camera viewpoint
Higher than previous score?
Yes
No
Set “target” viewpoint, move main camera to this viewpoint and record the score

Figure 27: Flowchart for aesthetics improvement camera setup.

The secondary thread is allowed to run near real-time and reports back scores for various viewpoints acquired. During this whole process, the main camera is not placed at the exact position that the player wanted it to be, but still it is known that the various compared viewpoints are actual candidate viewpoints that are generated using a conventional third person camera. The expected behavior of the viewpoint is not completely discarded. Without an extensive search or optimization for the "best" viewpoint, this method allows a near-real time aesthetic improvement by successively finding aesthetically pleasing viewpoints using the ones that would already been preferred by the player through classical third person views. Even if a better viewpoint cannot be found while the player is moving, this means that the current viewpoint is a better one and it is already being used. As a final note, the secondary camera will not render the avatar as its position will change with respect to the main camera. The evaluation rate would not be fast enough to take the avatar, which is continuously changing position, into aesthetic score prediction and only capturing the composition of the displayed background scenery is intended.

In the video game application, there may be overheads such as transferring the image data to the feature calculation module, thread management, running a MATLAB workspace etc. Therefore to adjust the frame rate of the aesthetic evaluation thread, the image is rendered on a smaller surface than the one presented to the user. This pre-scaling will introduce some errors when predicting aesthetic scores if it becomes smaller than 200k pixels. On the other hand, the final model should be able to report aesthetics scores at an acceptable rate for the procedure to work smoothly. The whole purpose of finding quicker features and feature selection is increasing this resolution of the aesthetic evaluation camera to an acceptable size (preferably to 200k pixels).

In order for the proposed improvement method to be effective, a regression analysis is required. Indeed, rather than an exact match between ground truth scores and machine learning predictions, it is still useful to have some rank relation for our purposes and rank correlation coefficients are also reported when determining the final performance.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, the main numerical results of the proposed approach described in the previous chapter are presented. Additionally, some of the decisions made based on these results are also included here. Finally the behavior of the final model in the real-time rendered environment is presented with short videos and screenshots.

4.1 Selected Model

When the *per-image score variance* bound, vote count bound and tag filter described in section 3.1 is applied on the AVA dataset [48], 8515 images were obtained. Of this dataset, 856 of the images (about 10%) were selected at random to keep out as an independent test set. The small deviation from the exact 10% image count of 851 is caused by the fact that the test set separation being done on the entire AVA dataset before all the experiments including sub-set selection. On the remaining 7659 images, three alternative machine learning algorithms were applied with the validation scheme explained in section 3.4.

All training processes were done using MATLAB's Machine Learning Toolbox [74]. Bagging tree ensemble and random forests were trained using 100 individual trees which resulted in an asymptotically stabilized performance (see section 4.2). The default value of minimum leaf size of 5 was used when training these two ensembles and Random Forests were trained by using only one thirds of the input variables.

When predicting the score for a new feature vector, all trees in the ensemble are used to vote for the final score and increasing the tree count would induce additional computational cost in the final implementation. The prediction time of the both ensembles is independent of the image size and proportional to tree count. Keeping in mind that the validation performance is comparable to previous regression analyses in the literature, the tree count is not increased further since the increase in prediction performance would increase by a small amount although the prediction time would increase in a linear manner.

Least squares boosting required some parameter optimizations to maximize its performance. The maximum validation performance was obtained at a learning rate of 0.05 and using maximum number of splits of 128 at the default minimum leaf size of 5, using about 31 individual trees. The performance of this algorithm is not asymptotical as the above methods and a minimum squared error occurs at a specific tree count. This phenomenon can be observed in Figure 28, in which a mean squared error versus number of ensemble trees graph is plotted for a single validation run of least squares boosting. In this graph the lowest attained MSE was 0.459.

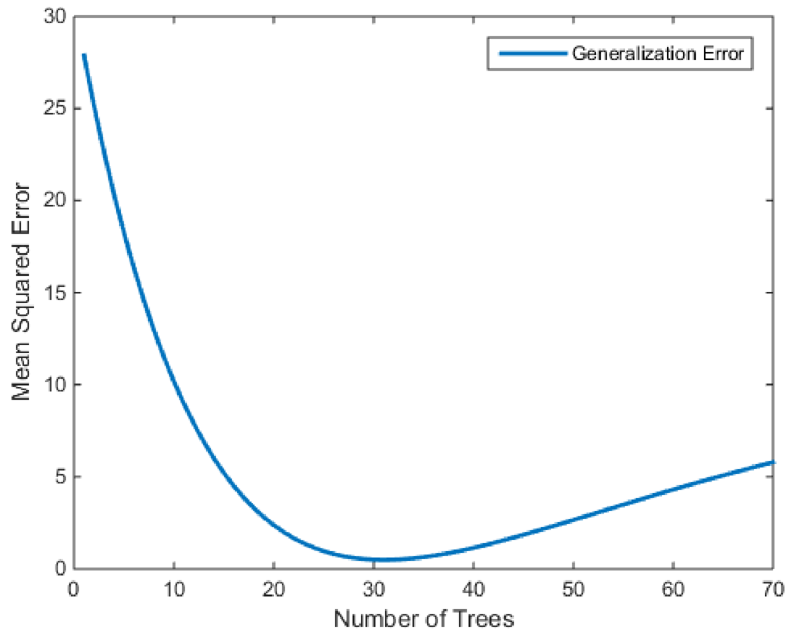

Figure 28: Number of trees versus mean squared error (MSE) plot of a single validation run of least squares boosting algorithm.

The final validation performances of tree bagging, random forests and least squares boosting are presented in Figure 29. The error bars represent the standard deviation of the values obtained by the validation repetitions and the actual value is the average. For convenience, the reduction from variance figures are reported as ratios instead of percentages in the graph, 1 being the highest achievable (100%) performance. Similarly, correlation metrics represent a better performance as they approach 1, and there is no correlation if they are 0. The vertical axis in Figure 29 is dimensionless and used to represent the variance reduction ratio and correlation coefficients.

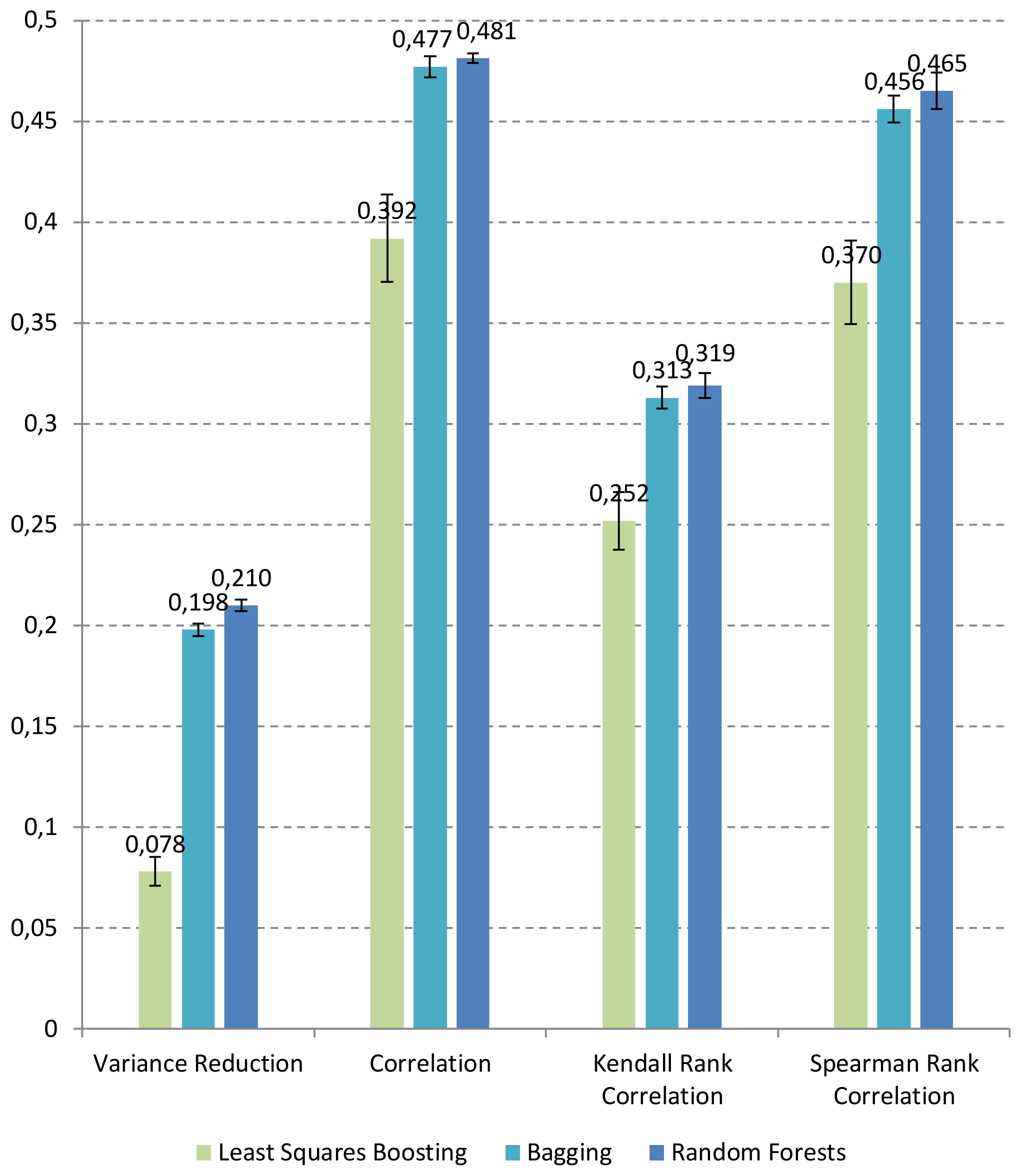

Figure 29: Relative performance of the trained machine learning algorithms.

The “Correlation” column represents Pearson correlation coefficient and the two other reported correlation coefficients are Kendall’s Tau-b and Spearman's rho. The reduction from the variance was calculated using the sample score variance of the whole ground truth labels ($\sigma^2 = 0.499$) as described in section 2.5.2.

The validation MSE values for the three models are; 0.394 for RF, 0.400 for bagging and 0.460 for least squares boosting. All of the performance values are combined in Table 3 for convenience.

Table 3: Validation performance of the three ML models

	MSE Attained	Variance Reduction	Spearman Rank Correlation	Kendall's Rank Correlation	Correlation Coefficient
Least Squares Boosting	0.460	7.8%	0.370	0.252	0.392
Bagging	0.400	19.8%	0.456	0.313	0.477
Random Forests	0.394	21.0%	0.465	0.319	0.481

The prediction performance of least squares boosting was behind the other two approaches especially at reduction from variance metric with a maximum of 7.8% reduction. The performance of this model is not acceptable. The performance difference among the two other models is not very prominent but random forests perform slightly better. This difference was more significant in reduction from the variance metric. Among the three proposed machine learning algorithms, random forests gave the highest validation performance and in the rest of the study only the random forests algorithm is used.

4.2 Trained Model

Having selected the learning model to rank the features, the model is re-trained using all of the training data and OOB permuted delta errors for each predictor are calculated as explained in section 2.3.4. In Figure 30, the mean squared error versus number of trees in the final ensemble (trained on the whole training set) is plotted. The blue and orange curves show the change in OOB error (averaging OOB samples' MSE over every tree) and in sample error (MSE when predicting the scores of the images in the training set) with respect to the tree count, respectively.

The minimum OOB MSE (which is an estimate of the out of sample performance) achieved with random forests at 100 trees was 0.392. There are two observations to make in Figure 30, related to both curves. First, it is possible to roughly approximate the out of sample variance reduction of the model (although to an optimistic extent due to the model selection process). Assuming ground truth score variance of the OOB samples are roughly equal to the variance of the whole sample (they were "sampled" from the training set), using the OOB MSE value, the out of sample variance reduction can be estimated as $((0.499 - 0.392)/0.499) \times 100 \sim 21.4\%$

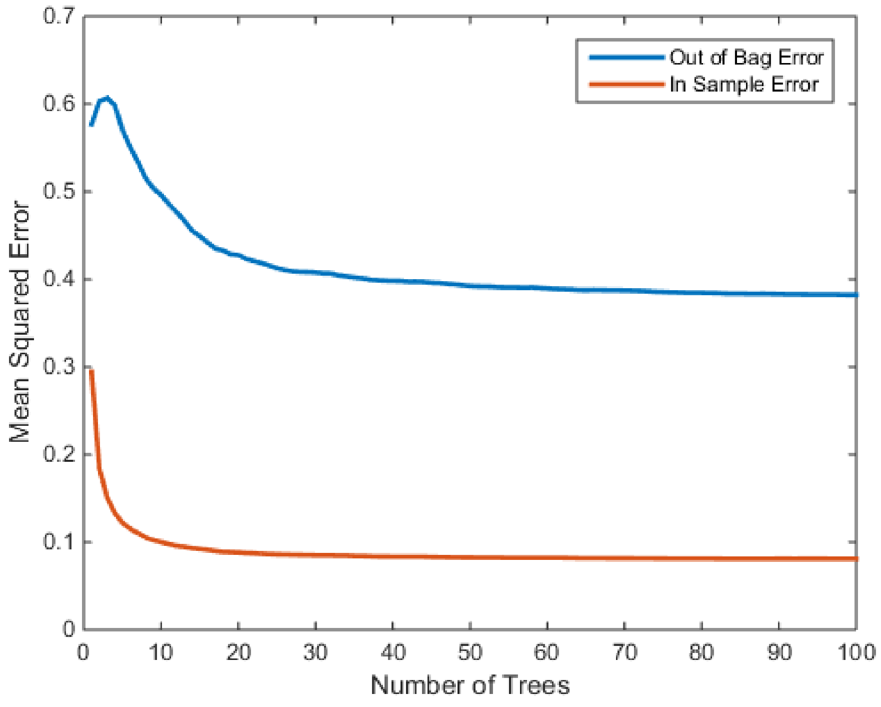

Figure 30: Mean squared error versus trees in the random forests

Second, predicting the scores for the training samples for reference, over 80% in-sample reduction from the variance (from 0.499 to 0.080) was possible which proves that the used features are able to fit score relations of the training data. Since the error rate is not zero, the hypothesis still have a bias predicting scores, but in an acceptable amount due to the inherent noise in the peer-voting process and it is not over-fitting the underlying data either. For comparison, the average *per-image score variance* is around 1.9 for the current dataset which is much larger than the best OOB prediction mean squared error of 0.392. This *per-image score variance* in the dataset makes up the inherent noise of the model, and it is not desired to model this into the hypothesis. There is a further source of hidden bias in our model that is the personal tastes of the dpChallenge community, but the important point here is to generate a relative scoring model, accepting their preferences as the ground truth.

4.3 Feature Run-Times

Run-times for feature extraction reported in this section are measured on a test machine with an AMD 4GHz 6 cores CPU. All these measurements are made on a random 100 image subset of the selected dataset and done without any parallel processing. All images are pre-processed (border removal) and scaled to yield 200k pixels. Calculations are timed using MATLAB's profiler and the total time spent for 100 images is reported. All code is highly optimized and uses MATLAB's array structures where possible for bulk processing.

These values do not represent the final implementation's run-time exactly and they are rather relative metrics. In the video game application, there may be other overheads such as passing the image data to the evaluation module etc. Furthermore, initial scaling operations (except for 100 by 100 template comparison image scaling operations) are not included in run-time measurements since in the final application, the scale of the image to be processed would be changed on the GPU by rendering on a surface of a smaller size.

As explained in section 3.5, the coarse feature run-times for 100 images are summarized in Table 4. Here, only the top seven slowest features are listed for brevity. Bottommost two entries are reported with an additional significant figure and others are rounded to the nearest integer. Segmentation operation as executed by [12] is divided into its two components which are detection of optimum number of segments and connected component analysis adding up to 210 seconds.

Table 4: Coarse feature calculation run-times in seconds per 100 images

Image Segmentation	**210+**
Finding Optimum Number of Clusters	100
Connected component analysis	110
Hough Transform	**113+**
Itti's Saliency Map	**20+**
Tamura Coarseness	**17**
Color Quality	**12**
Dark Channel	**5.2**
GLCM for a Single Channel	**3.7**

For segmentation, Hough transform and saliency analysis, only the initial calculations are reported without considering the time required to calculate the features based on the extracted base data and they are tabulated with an additional "+" sign. The remaining features on the other hand already represent image features on their own.

To elaborate the above statement, Hough transform applied on an image extracts the lines in the image but these lines should be converted into meaningful features adding up to the reported figure. When segmentation is utilized, after finding an image segment, further feature calculations on these image patches are required. And finally, after the saliency map is calculated further operations such as thresholding to find salient image regions, connected component analysis and calculating other features on these regions are required to generate meaningful features. Even with their base run-times these features take a long time to compute and there is no need to consider additional calculation run-times when finding the slowest features.

Out of the features listed in Table 4, the slowest three methods are eliminated by thresholding the feature calculation times above 20 seconds per 100 images. One may argue that using a further down-sampled version of the input image for these slow calculations, but Hough line detection was not confident enough on smaller images in the experiments. Furthermore segmentation features of Datta et al. [12] are not guaranteed to work as intended if the image is further down-scaled as no scaling information was given in their paper. Also it is known that with subject region detection, computational requirements increase very quickly as stated by [41] due to additional calculations that generate the real features out of the saliency map. To exemplify, after calculating the saliency map, thresholding and connected component analysis took about 0.97 seconds, a single geometric analysis on each of these segmented regions (center of mass, skewness etc.) took about 0.13 seconds and a single local image statistic (average brightness, average hue etc.) on a single region took 0.25 second per 100 images on average. Using the largest three salient regions and calculating only two geometric, and three image statistics features on these, the sum quickly increases to 22 seconds per 100 images, without even considering more time-consuming features such as dark channel feature.

The next slowest feature, i.e. Tamura coarseness is somewhat a borderline feature and will not be included in the model unless there is not any alternative high ranking feature in the feature selection process. Furthermore, even if it was to be included in the model, it would define a feature on its own without extra calculations (like saliency analysis, Hough transform, and segmentation) making this split a legitimate one.

After eliminating these three calculations and related features, a finer computation time analysis is done and the results are reported in Table 5. In this table, the first column represents the features used in the analysis, second column is the time required to calculate that feature (self-time), except the time required to calculate common operations. Common operations are the calculations shared among some of the features and these must be considered when adding a new feature to the model since a previously added feature can reduce the time required by a following feature calculation. These are tabulated on the rightmost two columns, with their short explanation and run-time. Again, all of the run-times in Table 5 are reported for 100 images as explained above. When the tabulated value is ~0, it means that the additional time requirement is negligible for 100 images once the common operation is done. Some of the common operations are dependent on other features such as *Wavelet Features* 10 to 12, which require the features 1-3, 4-6, 7-9 respectively. The second example is the *Michelson Contrast* feature which is trivial to compute once the *Maximum Intensity* and *Minimum Intensity* features are calculated. Additionally, features with negligible calculation times (such as *Image Size* and *Aspect Ratio*) are not included in the table.

Table 5: Detailed run-times of the calculated features.

	Self Time (s)	Common Operation	Time (s)
Tamura Coarseness	17.01		
Color Quality	0.04	Finding dominant colors	12.25
Real Color Quality	0.04		
Dark Channel Feature	5.12		
GLCM Hue Features	3.68		
GLCM Saturation Features	3.68		
GLCM Brightness Features	3.68		
Diff(Sobel Edge Histogram)	1.3	Local directions (Sobel operator)	2.11
Sobel Edge Histogram	0.35		
Wavelet Features (10)	~0	*Wavelet Features (1-3)*	~0
Wavelet Features (11)	~0	*Wavelet Features (4-6)*	~0
Wavelet Features (12)	~0	*Wavelet Features (7-9)*	~0
Wavelet Features (1-3)	2.12		
Wavelet Features (4-6)	2.12		
Wavelet Features (7-9)	2.12		
Simplicity Using 4096 bin RGB	2.3		
Hue Template	~0	Calculating best template fits	1.88
Hue Template Fit Values	~0		
Diff(Tamura Directionality)	0.4	Local directions (Prewitt operator)	1.31
Tamura Directionality	0.07		
Edge Energy Concentration Area	1.2		
L_1 Distance Hue Edge	0.5	Laplacian filter on hue	0.73
Absolute Sum Hue Edge	0.29		
L_1 Distance Saturation Edge	0.5	Laplacian filter on saturation	0.73
Absolute Sum Saturation Edge	0.29		
L_1 Distance Brightness Edge	0.5	Laplacian filter on brightness	0.73
Absolute Sum Brightness Edge	0.29		
L_1 Distance Summed Edge	0.5	Laplacian filter on H+S+V	0.73
Absolute Sum Summed Edge	0.29		
FFT Blur Metric	1.03		
L_1 Distance Improved Hue	0.88		
L_1 Distance Hue	0.5		
L_1 Distance Saturation	0.5		

Table 5: (continued)

L_1 Distance Brightness	0.5		
L_1 Distance Summed	0.5		
Hue Contrast	0.05		
Hue Count	~0	Histogram of filtered hue pixels	0.38
Most Frequent Hue Pixel Count	~0		
Absolute Sum Improved Hue	0.39		
Intensity RMS Contrast	0.38		
Absolute Sum Hue	0.29		
Absolute Sum Saturation	0.29		
Absolute Sum Brightness	0.29		
Absolute Sum Summed	0.29		
Diff(Average Intensity)	0.25		
Diff(Average HSV Saturation)	0.25		
Diff(Average Hue)	0.25		
Convolved Template Fit Values	~0	Convolution	0.11
Hue Template Convolution	~0		
Adjusted Hue Histogram	0.1		
Tamura Contrast	0.08		
Intensity Center Mass	0.04		
Average Intensity	0.02		
Michelson Contrast	~0	*Minimum, Maximum intensity*	~0
Minimum Intensity	0.02		
Maximum Intensity	0.02		
Average HSV Saturation	0.02		
Average Hue	0.01		
Total	55.34		20.96
Grand Total	76.30		

Important examples of use of common operations are: L1 distance edge calculations and absolute sum edge features on each separate channel were calculated on respective Laplacian edge images. *Hue Contrast, Hue Count, and Most Frequent Hue Pixel Count* features all use the filtered hue histogram described in section 2.4.8. *Sobel Edge Histogram* and *Tamura Directionality* features use the same direction data among their inner third difference and global variants. For the hue template fit features, once the fits are calculated (by convolution or custom distance metrics introduced in section 2.4.10), the fit values for individual templates and best fitting template calculations are trivial. For some of the features having multiple components such as *Adjusted Hue Histogram,* using different values on the histogram does not introduce a significant computation overhead and they were listed as a single entry.

Summing up self-times and common operation times of the features, time requirements were calculated as 55.34 and 20.96 seconds per 100 images. So the total time required to calculate all of the listed features was 76.30 seconds per 100 image. It is important to note that these figures have some duplicate features that are not intended to be used together. And yet, calculating the saliency map eliminated above takes 26% of the time required to calculate all these remaining features listed in Table 5, further demonstrating the inefficiency of saliency analysis with respect to other low level features.

Also, the overheads that are present in the final application are not included in these figures such as prediction time of the machine learning algorithm and HSV color conversion operations. HSV color transformation takes about 5.41 seconds for 100 images. With a rough calculation, using the whole feature set would take 0.82 seconds for an in-game frame which is roughly translates to only 1.2 frames per second excluding any other overhead. Even though some down-scaling could have been applied at the run time and features would be calculated on smaller images, this figure is required to be minimized without losing much performance, in order to be able to work with larger images (as close to 200k pixels as possible) in the final application in order for the model to work as expected.

4.4 Feature Importance and Selected Features

After training the random forests on the whole training data, OOB permuted delta errors of all the features are calculated, repeating the training for 5 times and averaging the variable importance. The process is repeated due to the bootstrap nature of the random forests algorithm. In the following sections, these importance ranks are tabulated and two feature selection approaches (one to be used in the final application and an alternative selection) are explained referencing these rankings.

4.4.1 Computation-Centric, Aesthetics-Aware Selection

In the following tables, the rankings of the features presented under aesthetic groups introduced in section 3.3 are listed. In each of these tables, the first columns are the feature names and the second column is the rank of the feature among 55 features (with 128 dimensions). The selected features among each set is marked with an asterisk (*) and listed in boldface type. For the properties with more than one dimension, minimum and maximum rank is tabulated and the highest ranking dimension is used as a comparison reference. As explained previously, considering features in aesthetically coherent sub groups, features with high feature importance are selected from each group. The total run-time of each group after the selection is also reported.

4.4.2 Composition Subset

Since the segmentation and saliency features were eliminated from the model in section 4.3, remaining composition features are important for aesthetic evaluation and investigated first. In Table 6, the feature rankings of the composition elements are listed. L_1 distance features of Lo et al. [41] were ranked on top of remaining composition features (above 5% of the remaining features with comparable calculation times) and the newly introduced *L1 Distance Improved Hue* ranked higher than its counterpart.

Keeping the integrity of features of Lo et al. [41], all of the L_1 distance features are selected out of this set. Instead of using the regular *L_1 Distance Hue* feature, *L_1 Distance Improved Hue* is preferred knowing that it involves a more realistic hue difference calculation although it takes a little more time to compute (0.88 versus 0.50 seconds for 100 images - Table 5). Furthermore, the edge distance features would also take care of the *Edge Energy Concentration Area* feature since the edge distributions were utilized when originally building this feature [20]. Selected features (marked with an asterisk) took 7.30 seconds in total (see Table 5). It is also important to note that the Laplacian transformations are introduced as a common operation, not to be recalculated again, by this selection.

Table 6: Importance ranking of composition features, also indicating selected features to be used in the final video-game application

L_1 Distance Brightness Edge*	7
L_1 Distance Saturation Edge*	11
L_1 Distance Brightness*	14
L_1 Distances Summed Edge*	15
L_1 Distance Hue Edge*	19
L_1 Distance Saturation*	31
L_1 Distance Summed*	42
L_1 Distance Improved Hue*	51
L_1 Distance Hue	64
Edge Energy Concentration Area	77
Diff(Average Hue)	115
Diff(Average HSV Saturation)	117
Diff(Average Intensity)	128

4.4.3 Line Composition Subset

In Table 7, feature rankings of the line composition features are tabulated. *Tamura Directionality* and *Diff(Tamura Directionality)* ranked worse than both histograms. *Sobel Edge Histogram* and *Diff(Sobel Edge Histogram)* features are in fact are more detailed versions of Tamura directionality capturing all different gradient directions instead of stating how directed the image is and the rankings support this idea. Among this set, it is required to select two features that can describe both overall directionality and center region relative directionality since they describe different properties and are independent.

Although *Tamura Directionality* for the entire image and the center part takes 1.78 seconds (lower than 3.76 seconds of *Sobel Edge Histogram*) they are eliminated being more than 30% behind the best bins of the histograms in rankings. Therefore *Sobel Edge Histogram* and *Diff(Sobel Edge Histogram)* are selected out of this set. After calculating the global directions using the Sobel operator (2.11 seconds to compute), it takes 1.30 seconds to filter and calculate center region histogram based on the existing data and the two selected features take 3.76 seconds for 100 images.

Table 7: Importance ranking of line composition features, also indicating selected features to be used in the final video-game application

Sobel Edge Histogram (15 dimensions)*	3(min)-99(max)
Diff(Sobel Edge Histogram) (15 dimensions)*	25(min)-74(max)
Diff(Tamura Directionality)	112
Tamura Directionality	122

4.4.4 Texture Subset

In Table 8, relative rankings of the texture related features are tabulated. *Tamura Coarseness* ranked at the top, mainly followed by the *Global Texture* features. Only the GLCM features calculated on hue channel ranked higher than *Absolute Sum Hue* feature. *Absolute Sum Improved Hue* ranked higher than its traditional counterpart.

Tamura Coarseness feature has a very significant high rank but it takes 17.01 seconds to compute for 100 images (see Table 5), much longer than the other alternatives. In order to replace it with a more cost-effective feature, the features within 5% rank difference are investigated. The best candidates are the *Global Texture* features (section 2.4.16), with comparable feature importance ranks. Additionally, they also capture other repetition properties such as the hue, saturation and edge texture and take only 2.32 seconds for 100 images all H, S, V, H+S+V, and edge components combined. Already introducing Laplacian edge images for all these channels in section 4.4.2, Laplacian convolution times are not included in this value.

For *Wavelet Features*, the rankings are generally low and it would be unnecessary to include wavelet transforms considering their total 2.12 seconds per channel computation time. As stated above, *GLCM Hue Features* rank higher than the hue component of the *Global Texture* features but with a calculation time of 3.68 seconds, it is not preferred either. Furthermore it is also aimed to keep the integrity of the global texture features of Lo et al. [41].

Therefore *Global Texture* features of Lo et al. [41] with much lower computation times with comparable feature rankings are selected. Instead of using *Absolute Sum Hue, Absolute Sum Improved Hue* feature is preferred based on its expected better hue texture resolution although it has a slight computational overhead (0.39 seconds versus 0.29 seconds). Selecting 8 features costs 2.42 seconds for 100 images in total, again excluding the Laplacian transformations as they are already calculated for composition features.

Table 8: Importance ranking of texture features, also indicating selected features to be used in the final video-game application

Tamura Coarseness	2
Absolute Sum Saturation Edge*	4
Absolute Sum Brightness Edge*	6
Absolute Sum Brightness*	9
GLCM Brightness Features (4 dimensions)	10(min)-37(max)
GLCM Hue Features (3 dimensions)	13(min)-63(max)
Absolute Sum Saturation*	18
Absolute Sum Summed Edge*	35
Absolute Sum Summed*	44
Absolute Sum Improved Hue*	49
GLCM Saturation Features (3 dimensions)	57(min)-72(max)
Wavelet Features (12 dimensions)	68(min)-121(max)
Absolute Sum Hue	73
Absolute Sum Hue Edge*	78

4.4.5 Hue Distribution Subset

In Table 9, Hue distribution related feature rankings are listed. *Real Color Quality* feature ranked highest followed by the original *Color Quality* feature. Template fit features are ranked way worse than *Color Quality* features of Lo et al. [41], however they took much less time to compute (12.29 seconds versus 1.88 for *Hue Template* and 0.11 for *Hue Template Convolution*). Similarly, convolution template features were calculated quicker than their [26] counterparts, but they were ranked lower on average. Furthermore, per-template fit scores (in total 7 for convolution and 7 for distance based fit score) were ranked higher than just using only the corresponding categorical template features for both fitting approaches, but still behind *Color Quality* features.

Considering the exceptional importance rank of the *Real Color Quality* feature (more than 30% rank difference between its best alternative *Adjusted Hue Histogram*) and the fact that example based color feature is expected to perform better than its rule-based counterparts [41], it is selected as the only feature out of this set with a 12.29 seconds calculation time (see Table 5).

Average Hue and *Most Frequent Hue Pixel Count* features, are already covered in *Color Quality* features due to the utilization of color histograms weighted by pixel counts (see section 2.4.7) and are not selected also considering their relatively low rankings.

Table 9: Importance ranking of hue distribution features, also indicating selected feature to be used in the final video-game application

Real Color Quality*	8
Color Quality	24
Adjusted Hue Histogram (16 dimensions)	54(min)-119(max)
Average Hue	59
Template Fit Values (7 dimensions)	65(min)-101(max)
Convolved Template Fit Values (7 dimensions)	76(min)-127(max)
Most Frequent Hue Pixel Count	110
Hue Contrast	123
Hue Template	125
Hue Template Convolution	126

4.4.6 Color Simplicity Subset

Two color simplicity measures are tabulated in Table 10. *Simplicity using 4096 bin RGB* feature has the highest importance (with a significant margin) in this set, and it is selected among the two color simplicity features with a calculation time of 2.30 seconds.

Table 10: Importance ranking of color simplicity features, also indicating selected feature to be used in the final video-game application

Simplicity Using 4096 bin RGB*	17
Hue Count	124

4.4.7 Saturation Subset

This set already has a single feature, tabulated in Table 11 and it takes 0.02 seconds per 100 images. This feature is also included in the model as it is the only feature relating to overall saturation of the image and ranked in the top 25% of the features.

Table 11: Importance ranking of saturation feature, being the only selected feature to be used in the final video-game application

Average HSV Saturation*	27

4.4.8 Brightness Subset

In Table 12, brightness related feature rankings are listed. *Average Intensity* and *Intensity RMS Contrast* were the highest ranking features in this set. Among this set, it is required to select two features that can describe both overall brightness and contrast since contrast is also an important and independent metric.

Average Intensity is selected being the highest ranking feature. To measure contrast, *Tamura Contrast* is also selected from this set. Although *Intensity RMS Contrast* ranked a little higher, it has a higher computational cost than other features (0.38 seconds versus 0.02 seconds for *Tamura Contrast*). The one with a slight computational advantage is selected also being within 5% rank distance to *Intensity RMS Contrast*. With these two 0.10 seconds worth of features (see Table 5), it is expected to describe contrast, dynamic range and overall brightness properties of the image.

Table 12: Importance ranking of brightness related features, also indicating selected features to be used in the final video-game application

Average Intensity*	12
Intensity RMS Contrast	16
Tamura Contrast*	22
Intensity Center Mass	55
Maximum Intensity	94
Michelson Contrast	108
Minimum Intensity	114

4.4.9 Sharpness Subset

In Table 13, rankings of sharpness related features are listed. *FFT Blur Metric* ranked higher than its counterpart *Dark Channel Feature.* It is important to note that the ranking of the *Dark Channel Feature* was worse than any highest ranking color related features in the above tables. It was originally used to both measure clarity and colorfulness [25] but did not rank comparable to any color related features selected above. Therefore it is eliminated at this point also considering its relatively high computation requirement of 5.12 second per 100 images. *FFT Blur Metric* on the other hand took only 1.03 seconds and is selected.

Table 13: Importance ranking of sharpness features, also indicating selected feature to be used in the final video-game application

FFT Blur Metric*	5
Dark Channel Feature	38

4.4.10 General Features Subset

Finally in Table 14, feature rankings of the general features are given. This high importance of *Image Size* is not guaranteed to be caused by the effect of image size directly on aesthetics and rather possibly influenced by the fact that higher resolution images perceived better and are preferred. Template based features are calculated on 100 by 100 square images, and do not have the aspect ratio information, therefore *Aspect Ratio* feature possibly complements them and other features. Even though there will not be direct control over these parameters in the real application and all images are scaled to have 200k pixels when extracting other features, if there is some interaction of other features with these two features, the system may optimize the aesthetic quality of the scene according to the monitor size and aspect ratio of the end-user. If this assumption is not true, the model will still be using other features on a constant *Image Size* and *Aspect Ratio* hyperplane, giving an aesthetic score prediction under these conditions.

Table 14: General feature rankings

Image Size*	1
Aspect Ratio*	39

4.4.11 Total Time Requirements for Selected Features

With this computation-centric, aesthetics-aware approach, 29.22 seconds per 100 images worth of 26 features (with 54 dimensions) are selected using the aesthetic classes. This time-box is the basis for the secondary approach explained below.

4.4.12 Secondary Approach

As an alternative, in this section, the features are selected only according to their global ranking within a time budget, without considering any grouping, as explained in section 3.6. The full list is not included here since it is very long and the selection process ends when only a few features are selected. Including the whole table would be unnecessary. The relevant part of the list was given in Table 15.

Table 15: Top 8 highest importance features, selected in the second approach

Image Size	1
Tamura Coarseness	2
Sobel Edge Histogram	3(min)-99(max)
Absolute Sum Saturation Edge	4
FFT Blur Metric	5
Absolute Sum Brightness Edge	6
L_1 distance Brightness Edge	7
Real Color Quality	8

To summarize the process, the features with negligible calculation times are initially included, namely *Image Size* and *Aspect Ratio* to be fair.

Following the list in Table 15 and run-times in Table 5;

Image Size ranking first is already included. *Tamura Coarseness* (took 17.01 seconds for hundred images) ranking second is selected.

A single bin of the *Sobel Edge Histogram* ranked third and the whole histogram (15 dimensions) is selected since the additional bins do not increase computation time. With these two features total time becomes 19.47 seconds.

Absolute Sum Saturation Edge requires the Laplacian filtered saturation image and together with its calculation makes the sum 20.49 seconds.

With *FFT Blur Metric* and *Absolute Sum Brightness Edge* features total time becomes 22.54 seconds.

L1 distance Brightness Edge feature only counts 0.5 seconds since the Laplacian brightness channel was already calculated with previously selected *Absolute Sum Brightness Edge* feature making the sum 23.04 seconds.

Real Color Quality feature is calculated in 12.29 seconds on its own and exceeds the 29.22 second budget but regardless, it is also selected making the total 35.33

seconds. Finally only 8 features (with 22 dimensions) are selected which took a little longer to compute than the previously selected 26 features.

4.5 Test Set Performance

In this section, the performance of the above developed machine learning model is evaluated on the previously separated independent test set to give unbiased performance estimates. Doing the performance analysis on the previous training set would yield optimistic results since we have utilized all the training data to select features in the previous discussion. The main model that is intended to be tested is the one using random forests as the learning algorithm and the proposed computation-centric, aesthetics-aware selection of features. To fulfill scientific curiosity, the test performance of the whole 55 feature set (128 dimensions) and the alternative selection method are also presented. The mean squared error of the trained model, and its resulting reduction from the sample score variance (of 0.499) are tabulated. Similar to the above results, Kendall's rank correlation, Spearman's rank correlation and correlation coefficients are also calculated on the test set to make the results comparable to future works. All of the values are reported in Table 16 by averaging performance of 10 independent runs on the same test set due to the bootstrap nature of the algorithm.

Table 16: Test set performance of the proposed methods.

	MSE Attained	**Variance Reduction**	**Spearman Rank Correlation**	**Kendall's Rank Correlation**	**Correlation Coefficient**
Full Feature Set	0.403	19.3%	0.468	0.322	0.478
computation-centric, aesthetics-aware selection	0.410	17.8%	0.444	0.305	0.455
Budget Selection	0.416	16.6%	0.426	0.292	0.441

The highest performance was attained using the full feature set, followed by the heuristic feature selection and budget selection respectively.

To further investigate the results on the test set, two scatter plots are given for the aesthetics-aware selection. In Figure 31, predicted scores tend to follow the real (ground truth) score values but there is not an exact match on the score range (real scores are between ~3 and ~8.5 whereas predictions are between ~4 and ~6.5). This is not very critical for this application as long as some relative scoring among images persists. The scatter plots for the other two feature sets are very similar in appearance and are not plotted.

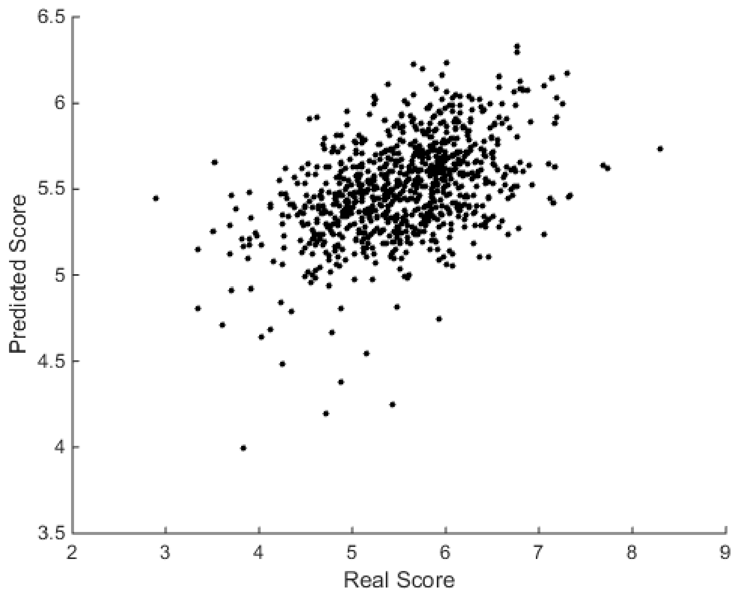

Figure 31: Scatter plot of real scores versus predicted scores.

When real scores are plotted against MSE in Figure 32, another observation can be made, that is at the extreme ends of the score spectrum, MSE increases.

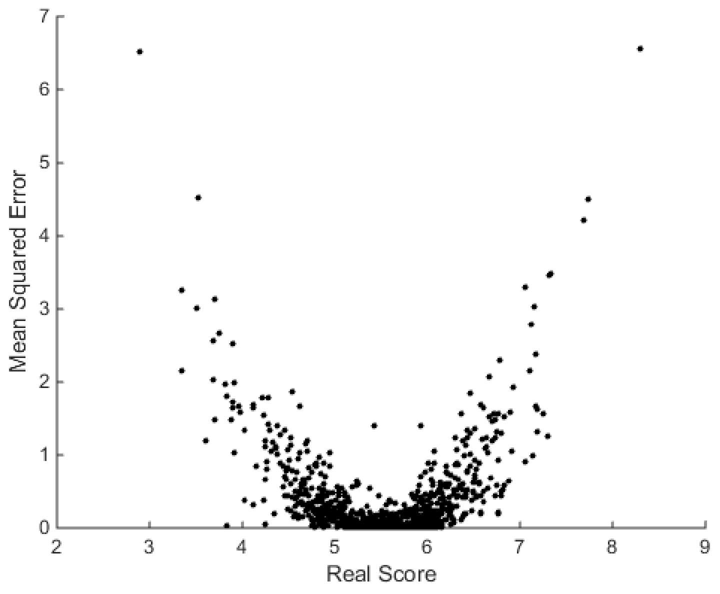

Figure 32: Scatter plot of real scores versus mean squared error.

Still the aim in this work is not generating scenes with very good scores but rather differentiate between worse and better. To better understand that these figures are indeed promising, it is important to note that randomly permuting the original scores as was done by [12] before training and calculating the above metrics gives almost zero correlation and very high MSE values which proves that the model is indeed learning from the data and is not over fitting an unnecessarily complicated hypothesis.

4.6 Final Training and Video Game Application

The model is finally trained using the whole data set to be used in the final application using the aesthetics-aware features. Computational times were reduced almost in half and the performance of 17.8% reduction from variance was attained using this model. By reducing the computational cost (number of features), some prediction performance loss was expected but in return, the size of the image to be processed by the aesthetic evaluation unit can be increased resulting in more detailed and faster aesthetic analysis. Another thing to consider is the fact that the machine learning algorithm was trained on real photographic images and the performance was estimated using a similar set of images. In the final application on the other hand, this performance is not guaranteed since the model is now predicting scores for computer generated images. In case of computer generated images, the output is smoother without noise and the objects are approximated with polygons, which may introduce a difference when evaluating aesthetics. Nevertheless, some prediction capability is expected to persist as color and edge distributions can still be approximated.

Aspect Ratio and *Image Size* are calculated using the actual screen size in the final application since the real image seen by the end-user has a high resolution, which in turn would affect his/her aesthetic appreciation. Then it is required to run the aesthetic evaluation module using 200k pixels to achieve the optimum evaluation performance that the learning algorithm was built with. Since the ML algorithm is also using *Aspect Ratio* and *Image Size* features based on the original image size (before scaling to 200k pixels), using the actual screen size for these features make the final model consistent with the trained model although it is not completely clear whether these will be beneficial in the final application. When the application was run by scaling the virtual image to 200k pixels by rendering on a small surface, evaluation of a single aesthetic camera frame took 0.546 seconds on average on the test machine. That is, while 60 frames per second (with vertical sync) are displayed to the user in game as the output of the main camera, frames to be compared concurrently at background for aesthetic quality comparison are selected from the secondary camera at a rate of 1.83 frames per second.

The feature calculations take about 0.292 seconds (was calculated as 29.22 seconds for 100 images in section 4.4.1), HSV color space transformation takes 0.054 second and the machine learning algorithm’s prediction takes 0.134 second on average for a single frame. The remaining time of 0.066 is the overheads caused by the GPU scaling, texture transfer to main memory and MATLAB byte array conversions. These values are summarized in Table 17.

Table 17: Real-time computation run-times (in seconds per 200k pixel image)

Feature Calculation	0.292
Prediction	0.134
Overhead	0.066
HSV conversion	0.054
Total	0.546

Using a 200k pixels rendering surface, the achieved 1.83 frames per second evaluation rate resulted in an acceptable camera direction performance and resulted in fluid camera movement between found high quality locations. As the aesthetic evaluation rate decreases, the viewpoint alternatives that can be evaluated also drops. The rate at which the evaluation is performed can further be increased using smaller render surfaces but doing so will not guarantee a correct aesthetic score prediction.

Another thing to investigate on the final model is the image templates that were used when determining template based features (section 2.4.6). The resulting high quality and low quality templates for the brightness, hue and brightness edge images are shown in Figure 33, Figure 34 and Figure 35 respectively. In all of them there is a significant difference at the top part of the template, which corresponds to the sky region for landscape images. In any case there is a visible difference in distributions between the high and low templates, which in return makes the analysis effective. Better quality photographs' templates have a more prominent horizon location.

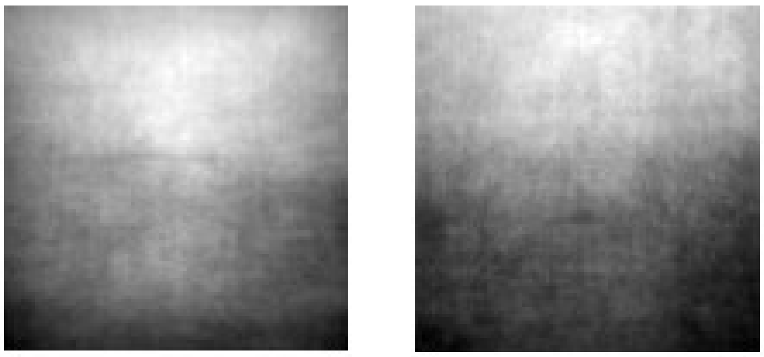
Figure 33: High (left one) and low (right one) quality brightness templates

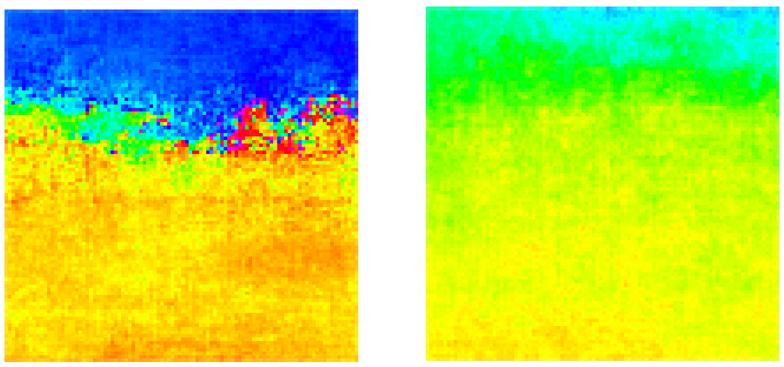

Figure 34: High (left one) and low (right one) quality hue templates

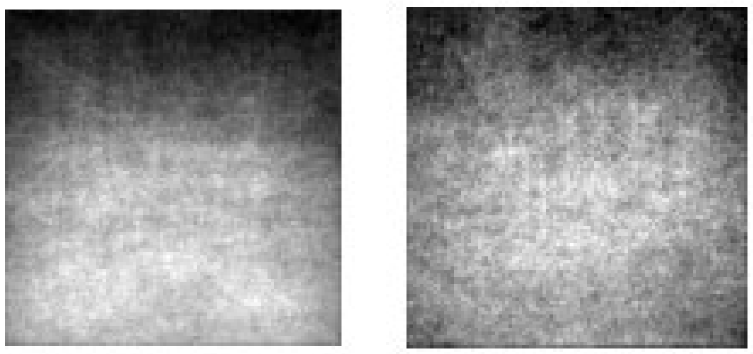

Figure 35: High (left one) and low (right one) quality brightness edge templates

Finally using the camera direction method described in section 3.7, some screenshots and videos are captured showing the various states of the described system. In Figure 36 and Figure 37, the overlaid view on the bottom right is the view of the secondary camera that is concurrently being evaluated. Figure 36 shows the initial state of the aesthetically improved camera setup. Since there are not enough collected scores yet, the camera assumed its initial third person position. When the avatar (the white character) was moved, the secondary camera looked for a higher aesthetic score viewpoints and after switching between some intermediate viewpoints, the view in Figure 37 was determined to be a higher quality viewpoint. After such a view is found, the camera generally stabilizes and moving the avatar does not affect the camera position since there are not any better views around this region of the scene when looked from the avatar's third person view. If the automatic camera direction is switched off, the third person view similar to the one in Figure 36 is followed in the classical manner.

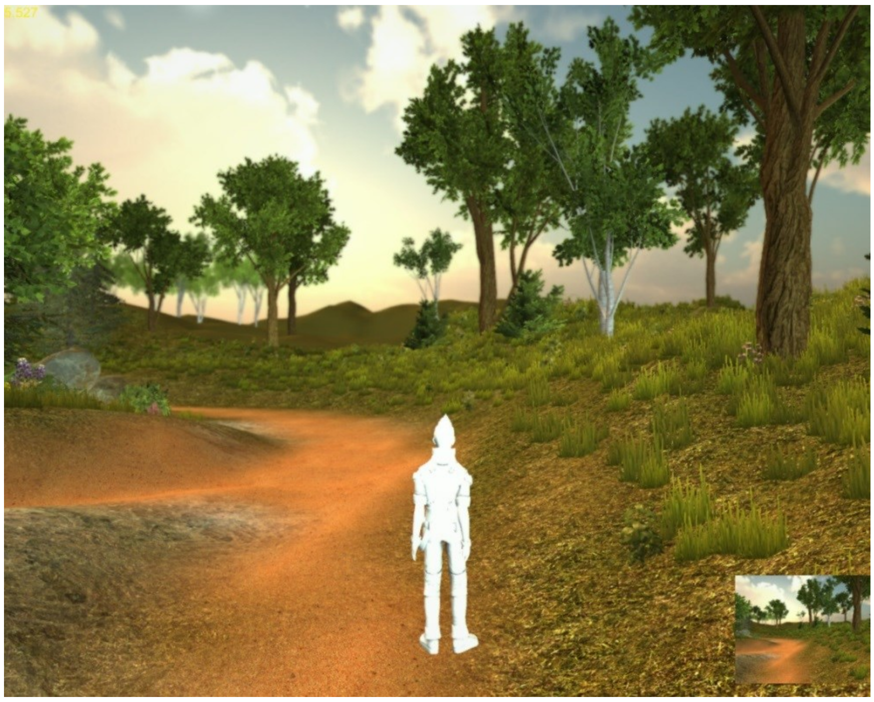

Figure 36: Initial state of the aesthetic improvement system.

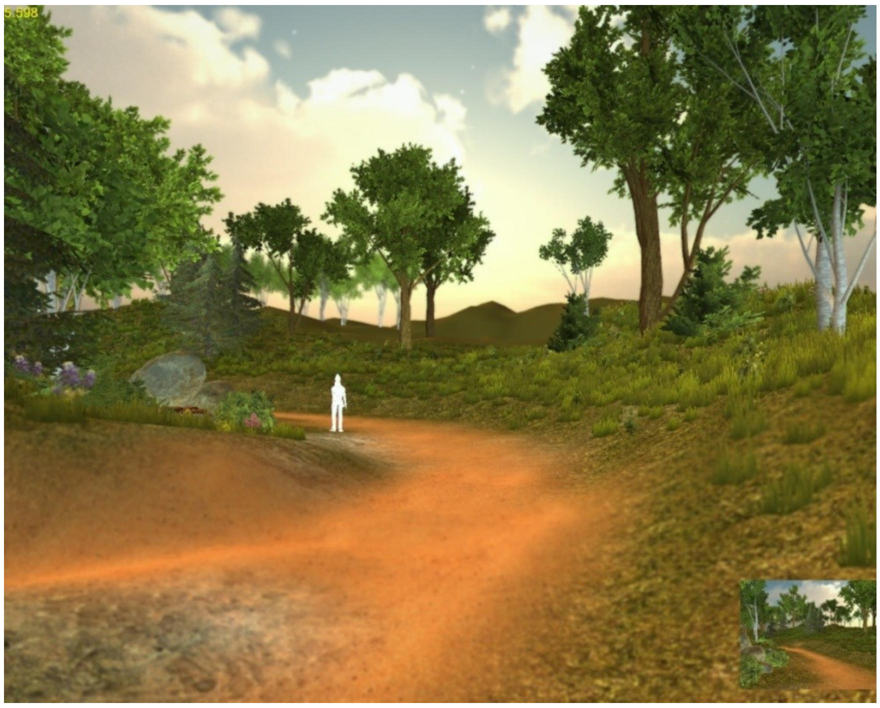

Figure 37: Aesthetic improvement system after the aesthetic score was stabilized.

Two separate short videos of the classical third person camera[1] and the proposed aesthetic improvement camera[2] are accessible online. In the camera direction video, the frame at the bottom right shows the actual image seen by the secondary aesthetic camera and the number on the top left is the aesthetic score of the currently "targeted" camera viewpoint. It takes a little time for the main camera to reach this target view since the motion was smoothed. In this video, different camera direction phases can be observed. As long as the player stays in the frame, the camera moves to better viewpoints and when the player gets too far away, leaves the frame from one of the sides or gets occluded, the score gets reset as can be observed on the top left corner of the video frame, and the process starts over.

The second video of the third person camera only had a camera smoothly following the player, on a path similar to the one followed in the first video. This view is the base reference for the aesthetic improvement camera.

Finally, various different viewpoints from the eye of the aesthetic camera were collected by moving the evaluation camera around the virtual scene using a 16:9 aspect ratio camera. It is computationally intensive to analyze every possible meaningful viewpoint in the constructed scene but a random sampling was used instead. Score range of these collected images ranged from 5.11 to 5.61 (relative to the dataset used in the training). Below figures (Figure 38 to Figure 47) shows two sets of captured images chosen from the high and low ends with their respective predicted aesthetic scores. Lower quality scenes were presented following the high quality images. Each set of five images were ordered according to their relative predicted scores, higher quality images being first.

Although it is possible to find scenes with scores outside 5.11-5.61 range, assuming this range is the absolute score range of possible viewpoints; a relative scale is also constructed to better discriminate the relative qualities of the rendered scenes. Normalizing the scores in this range, a "predicted normalized score" is calculated giving the top percentile of the image as;

$$predicted\ normalized\ score = \left(1 - \frac{score - 5.11}{(5.61 - 5.11)}\right) \times 100\% \qquad (31)$$

These values are also indicated in parentheses in the following figures, using the top percentile for the higher quality viewpoints. For the images of lower quality, the bottom percentile is also reported.

[1] Available at https://www.youtube.com/watch?v=jizX3_nTEO0
[2] Available at https://www.youtube.com/watch?v=jyopKrsLz7M

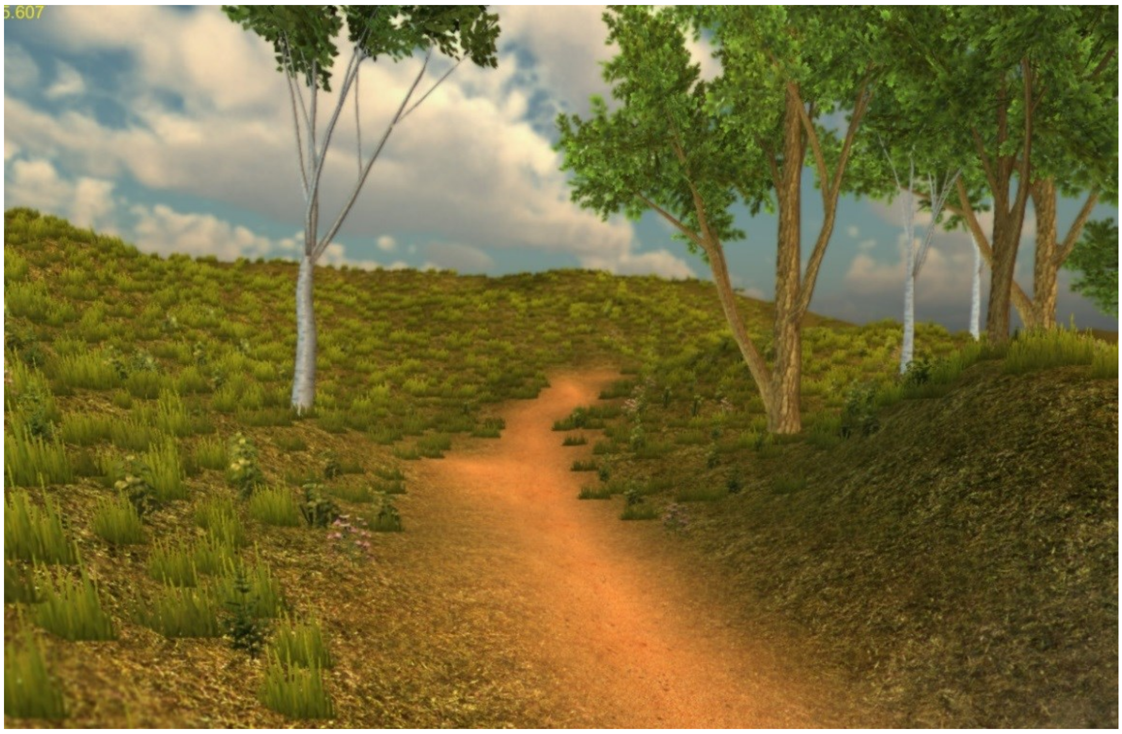

Figure 38: Example scene, predicted score: 5.61 (top 1%)

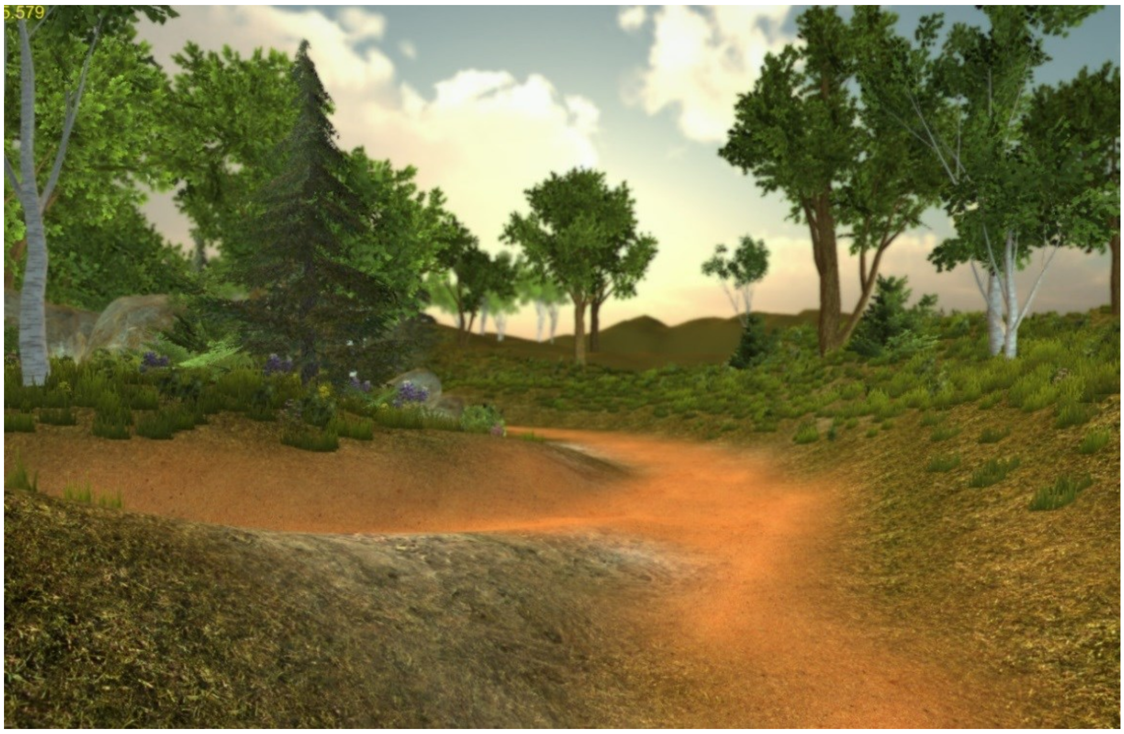

Figure 39: Example scene, predicted score: 5.58 (top 6%)

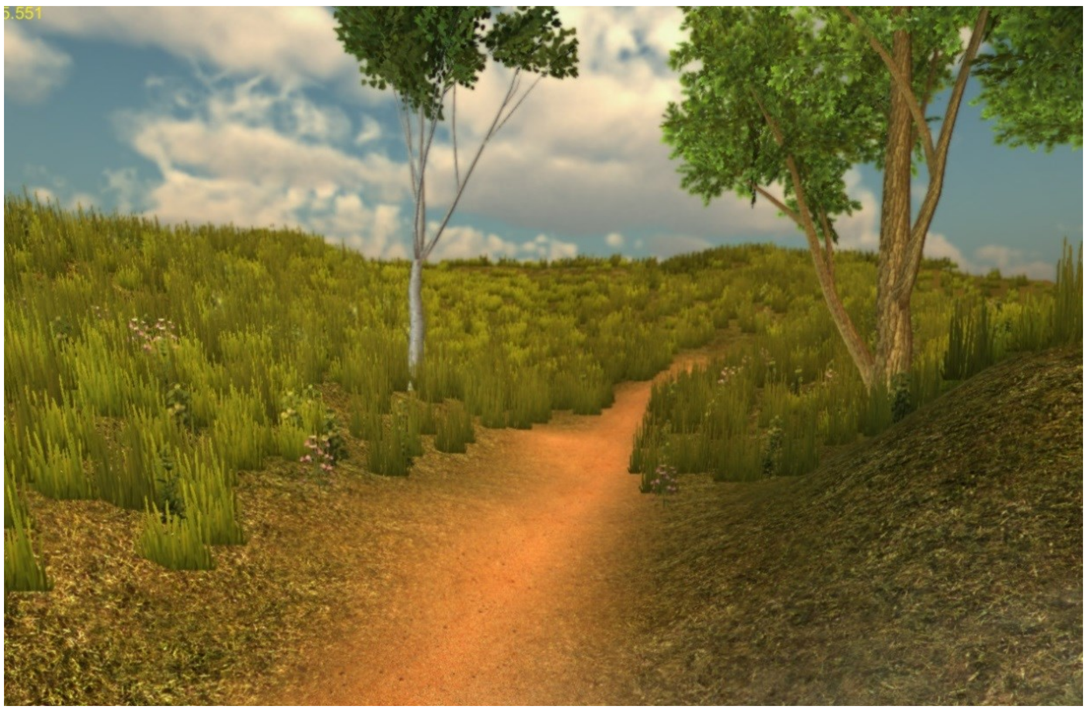

Figure 40: Example scene, predicted score: 5.55 (top 12%)

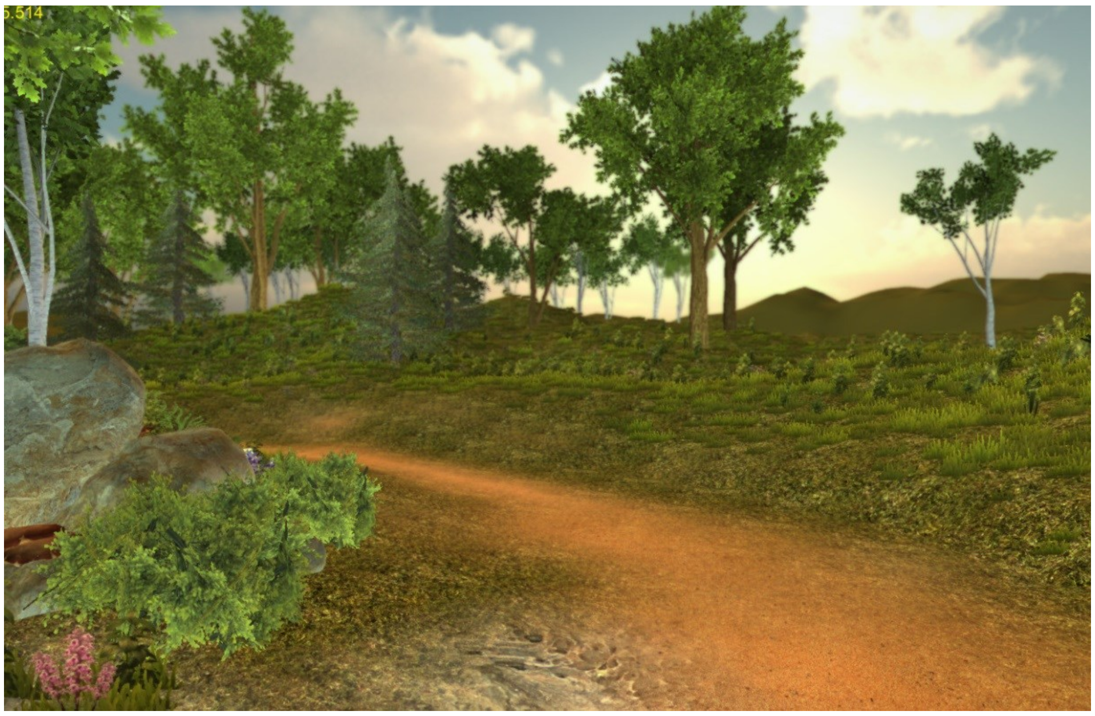

Figure 41: Example scene, predicted score: 5.51 (top 20%)

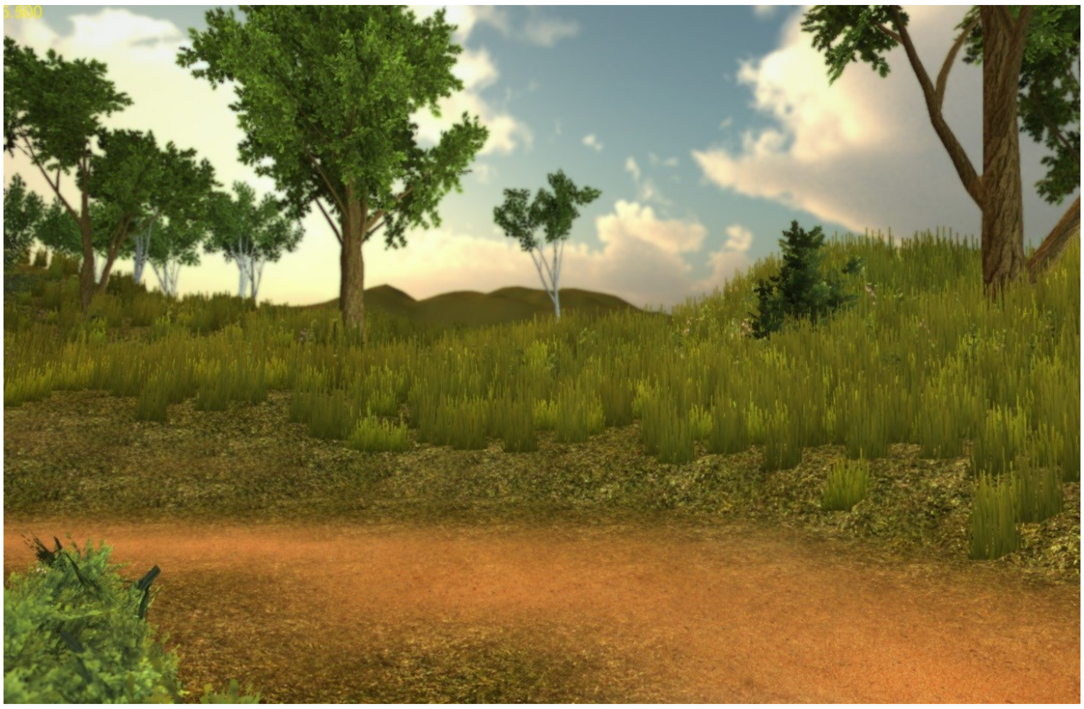
Figure 42: Example scene, predicted score: 5.50 (top 22%)

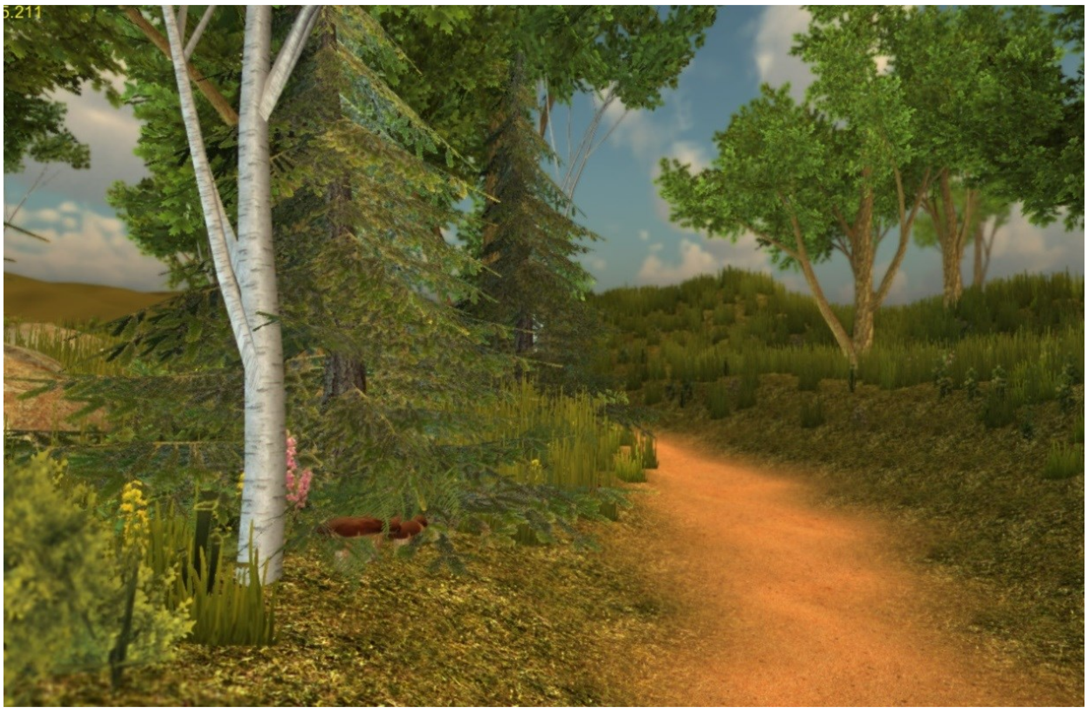
Figure 43: Example scene, predicted score: 5.21 (top 80%, or bottom 20%)

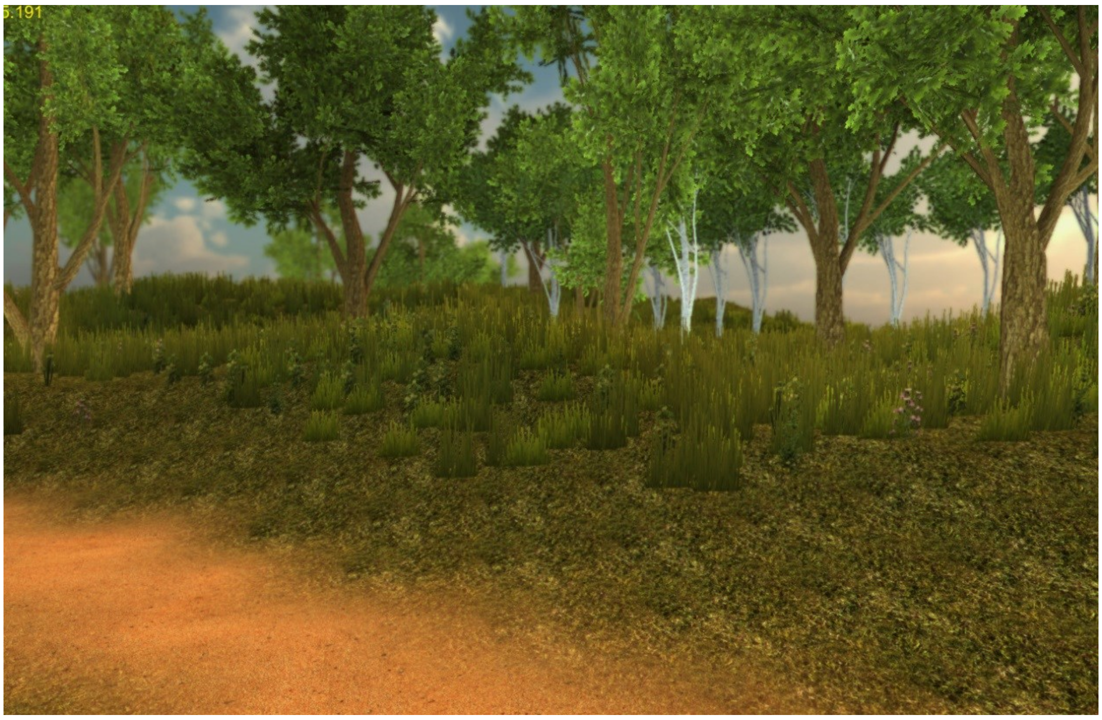

Figure 44: Example scene, predicted score: 5.19 (top 84%, or bottom 16%)

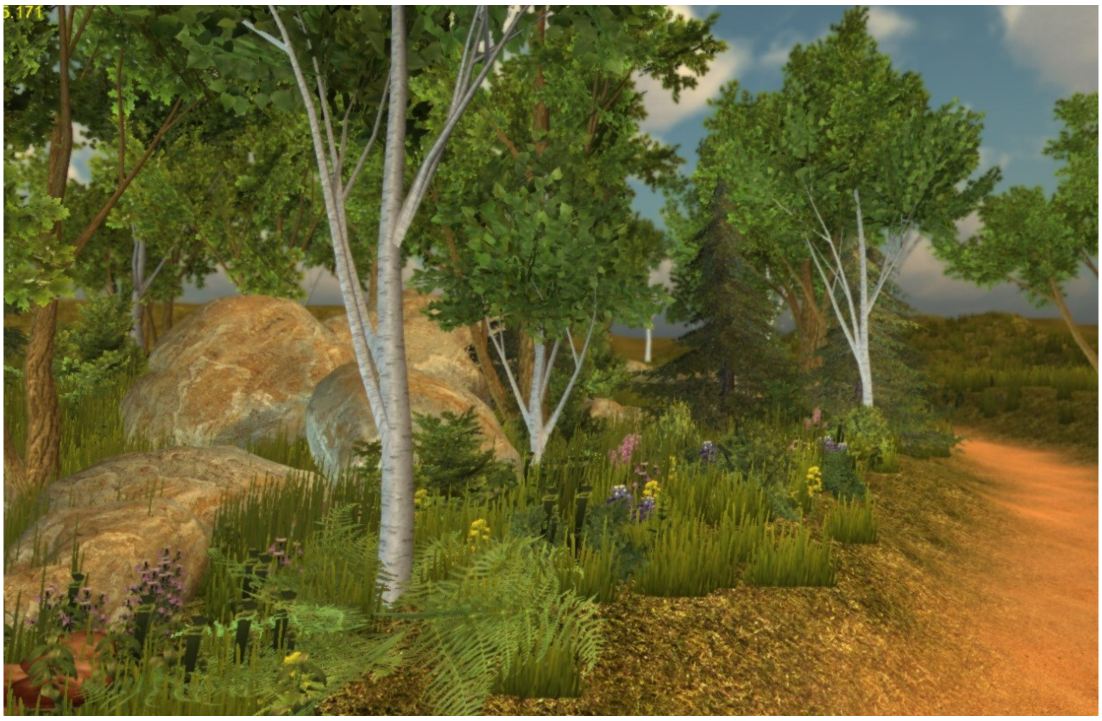

Figure 45: Example scene, predicted score: 5.17 (top 88%, or bottom 12%)

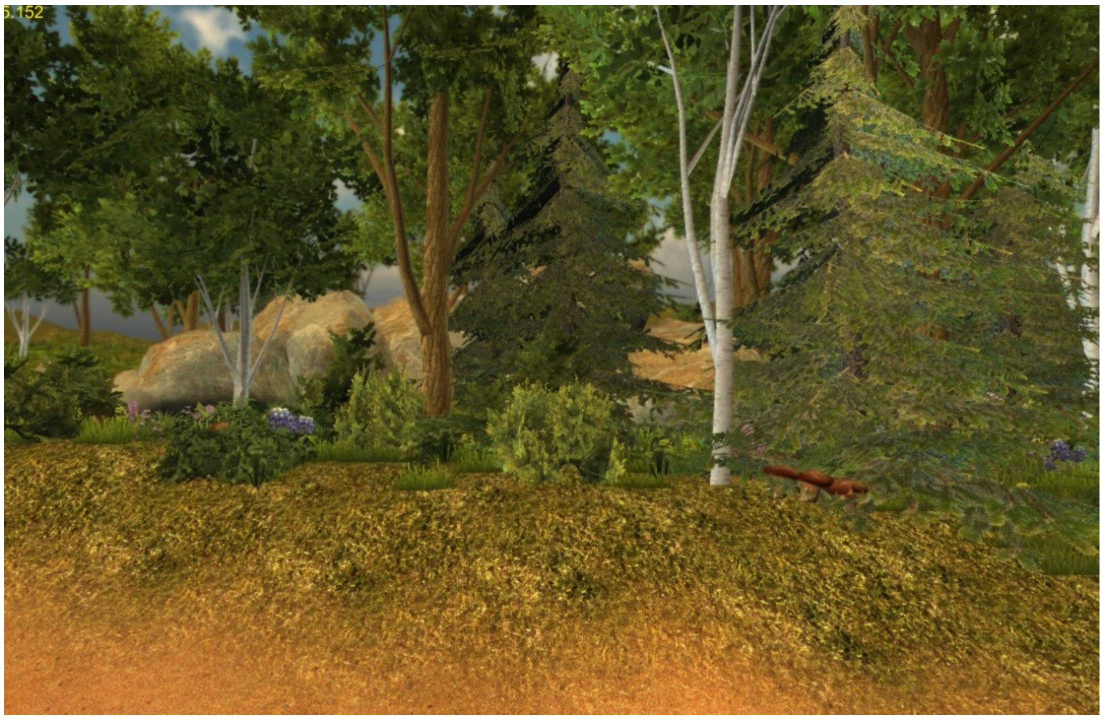

Figure 46: Example scene, predicted score: 5.15 (top 92%, or bottom 8%)

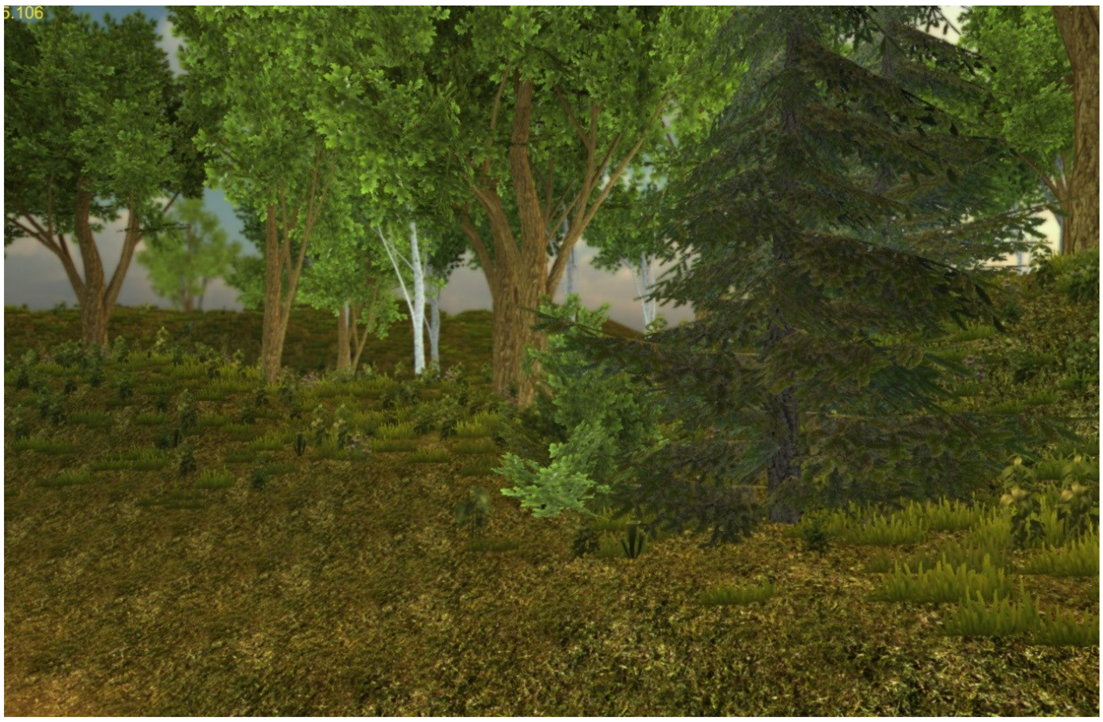

Figure 47: Example scene, predicted score: 5.11 (top 99%, or bottom 1%)

CHAPTER 5

CONCLUSION AND FUTURE WORK

Computational aesthetics is a newly developing interdisciplinary field uniting many fields such as computer science, cognitive science, and art. In this work, computational aesthetics is applied on a video game application with outdoor scenes/landscapes. A novel approach is developed in which the aesthetic quality of actual rendered game scenes are evaluated for camera direction in real time while the game is being played, for the first time in literature. Even though there are various studies in literature in computational aesthetics area, only a few of them were related to video games. In [70], visual aesthetics of space-ships present in the game environment were improved using computational methods and in [71], the level design of platform games was evaluated. In [69] and [68], virtual camera systems was developed that can make aesthetic decisions based on the positions and states of virtual three dimensional objects, without considering the final rendered pixels. The main difference here is that the actual "presented" raw image's aesthetic quality is evaluated using visual aesthetic features covering a wide range of visual aesthetics properties such as color, brightness and composition. A real-time virtual camera director that is aware of the final pixels in the image and capable of modifying the position of the camera accordingly is proposed, without depending solely on the rough positions of three dimensional objects.

During the course of this work the AVA dataset [48] is tailored to the needs of the problem at hand using the available tags, *per-image score variance* and implementing a further border removal process. Using machine learning algorithms, a relation between the raw image pixels and aesthetic value is attained. Considering the real time requirements of the task, a clever feature selection method is applied and rather than dealing with face detection and semantic context, computationally more efficient methods are incorporated in a regression analysis setup. When determining better features, an embedded feature selection method, after finding a powerful machine learning algorithm is used, keeping in mind the actual meanings of these features and their aesthetic foundations. Many of the features of Lo et al. [41] are found computationally efficient and discriminative as they claimed in their original paper.

Being a highly subjective topic, predicting aesthetic value is a challenging topic and predicting scores rather than image classification makes the evaluation more difficult. Some of the features in the previous works originally used for classification are used for regression instead. Lo et al. [41] applied their features on the CUHK dataset [25], which was only intended for high and low quality class separation. Adopting some of their features, this work uses them in a regression setup by separating high and low quality images out of the AVA dataset and using them as the building blocks for these features. Doing so, the KNN color quality feature of Lo et al. [41] is modified to capture the ground truth scores of the images by averaging the KNN scores to improve regression performance by generating a finer granularity feature. Furthermore their template based and texture features [41] are improved to capture the hue relations better, by calculating more realistic hue averages and differences.

In the previous works, up to 28% [12], 24.4% [10] and 25% [15] reduction from variance and Kendall's Tau-b score of ~0.25 [31] were reported for various different approaches. When compared, the prediction performance of the developed model is not very significant but highly encouraging considering the limitations involved. Segmentation method of Datta et al. [12] takes an order of magnitude longer to compute than all the features utilized in this work combined. The final model with a test set variance reduction of 17.8% and Kendall rank correlation coefficient of 0.305 shows that a near-real time visual aesthetic improvement is possible with some prediction capability using the newly introduced "virtual" camera direction framework. By reducing computational complexity, a quicker visual scene analysis is achieved. The camera director acts as a lazy optimization process trying to find better viewpoints by successively comparing aesthetic scores of different perspectives in the virtual scene. This is a novel work introducing visual computational aesthetics to the video game industry using an original approach to evaluate and improve real-time graphics. Furthermore it would be possible to apply this method on other rendered environments such as cityscapes for games taking place in a city or sports games by changing the used dataset and further tailoring the features.

Still, the resulting virtual camera director is not specifically applicable for high-paced, action-packed games where a fast tracking camera may be preferable but can be successfully applied to other genres such as adventure or exploration where the environment is less threatening and movements are rather slow and calm. Furthermore, similar to a loosely tracking third person camera, it is possible to see the avatar's front side using the proposed camera, for a longer time period. If the current viewpoint is of a high aesthetic quality, it will remain constant as long as the player stays in the field of view and the avatar can look directly into the camera. Although the introduced method only considers the avatar related camera direction, a similar approach can be applied to cut-scenes and long camera transitions to reduce the labor required to construct them. Even though a similar approach can be applied to a real-life camera setup as was anticipated by [39], using a virtual environment makes the process straightforward since the virtual camera can "see" wherever it needs to without blocking or interfering with other objects. On the other hand there is much further work to do and this is only a small step towards the synergy between video games and visual computational aesthetics.

Doing the feature calculations on the GPU and reducing the computational times of these features is one of the most important future directions of this work. Instead of eliminating features, better discriminating features can further be introduced in the model without increasing the computation times. Another alternative improvement is running multiple aesthetic cameras on various threads and analyzing a wider range of viewpoints, lighting/exposure conditions, field of views and depth of fields although it would be much more complicated to implement than the one introduced here. This approach requires even more performance optimizations and GPU parallel computing, considering the limited processing power of the current home-user personal computers.

Although the reported performance figures are comparable to previous works and it is clear that the model indeed resolve aesthetic quality to some degree, the actual effect of applying this evaluative model to the rendered graphics on the user remains unclear. A carefully prepared user survey to evaluate the proposed method in terms of aesthetics and gameplay should be considered. At this stage this work remains as a proof-of-concept and without constructing a full playable game, the survey would not be informative especially in terms of gameplay evaluation using the proposed camera direction method. Similarly, since the original model is trained on real images but the evaluation is done on computer generated images, the estimated performance on the image dataset may not be the same in the final application but it is expected to approximate aesthetic quality.

In this current state, the application only evaluates scene aesthetics and finds statistically better viewpoints without considering the position of the virtual character. Further improvements on the perceived aesthetics can be achieved via character position analysis (to better use rule of thirds for example) similar to the work of [68], while still continuing to evaluate raw image aesthetics. Another thing to consider is the effect of moving images (or videos) on the perceived aesthetic quality. In this approach only the individual frames are analyzed as still images without considering the successive frame's interactions and movement related aesthetic effects. Furthermore commonly used camera transitions in cinematography such as jump cuts etc. can also be integrated, subject to various constraints for different camera positions. Additionally, there are many features already found to be related to visual aesthetics in the literature and only a limited amount of them are considered and evaluated. Doing a far-reaching analysis of aesthetic features is essential, considering the features with low computation times, and newer features should also be engineered. Also, the used dataset is not very clean and have images other than pure landscape scenes. A better dataset should also be generated possibly rated by a group of photography experts, establishing a gold standard. Furthermore, to make the performance analysis realistic in the computer generated image domain, establishing an image dataset consisting of computer generated landscape images (video game scenes) with their aesthetic ratings should be considered to be used as the ground truth when building the model. Doing so, the effect of screen size and aspect ratio can be further investigated using various alternative datasets collected for different computer screens. Keeping the screen size constant for a single model, the ML algorithm would not have to deal with many different image sizes as it was the case with the collected dataset.

Using real time graphics, it is already possible to have an idea on the semantic elements since the predefined objects are being rendered. Using training images with semantic tags and incorporating these tags to the machine learning algorithm, it may be possible to improve this camera system with tagged virtual objects. By checking whether these tagged virtual objects are in the rendered frame or not (or how much of them is visible) and evaluating accordingly with respect to the learned tags and other low-level image properties, a semantic aesthetic evaluation becomes a possibility, without the full price of extracting object from the raw image data. Another exciting idea, inspired by the success of template-based features, is using a full deep-learning scheme via convolutional neural networks on image data and building a similar system presented in this thesis to improve the predictive performance and eliminate the complicated feature selection process.

REFERENCES

[1] European Network on Quality of Experience in Multimedia Systems and Services, "Qualinet White Paper on Definitions of Quality of Experience," Lausanne, 2013.

[2] İ. Tunalı, Estetik, İstanbul: Remzi Kitabevi, 2002.

[3] F. Hoenig, "Defining Computational Aesthetics," in *Computational Aesthetics in Graphics, Visualization and Imaging*, Girona, 2005.

[4] J. Dhiraj, D. Ritendra, F. Elena, L. Quang-Tuan, W. J. Z., L. Jia and L. Jiebo, "Aesthetics and Emotions in Images," *IEEE Signal Processing Magazine,* pp. 94-115, September 2011.

[5] S. S. Khan and D. Vogel, "Evalutaing Visual Aesthetics in Photographic Portraiture," in *Computational Aesthetics in Graphics, Visualization, and Imaging*, 2012.

[6] D. Joshi, R. Datta, E. Fedorovskaya, Q.-T. Luong, J. Z. Wang, J. Li and J. Luo, "Aesthetics and Emotions in Images," *Signal Processing Magazine, IEEE,* vol. 28, no. 5, pp. 94-115, 2011.

[7] V. Ciesielski, P. Barile and K. Trist, "Finding Image Features Associated with High Aesthetic Value by Machine Learning," in *International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design*, Copenhagen, 2013.

[8] G. Stiny and J. Gips, Algorithmic Aesthetics, California: University of California Press, 1978.

[9] P. Machado and A. Cardoso, "Computing Aesthetics," in *Advances in Artificial Intelligence*, Brazil, 1998.

[10] P. Obrador, M. A. Saad, P. Suryanarayan and N. Oliver, "Towards Category-Based Aesthetic Models of Photographs," in *18th International Conference, MMM 2012*, Klagenfurt, 2012.

[11] X. Tang, W. Luo and X. Wang, "Content-Based Photo Quality Assessment," *IEEE Transactions on Multimedia,* vol. 15, no. 8, pp. 1930-1943, 2013.

[12] R. Datta, D. Joshi, J. Li and J. Z. Wang, "Studying Aesthetics in Photographic Images Using a Computational Approach," in *European Conference on Computer Vision*, Graz, 2006.

[13] R. Gallea, E. Ardizzone and R. Pirrone, "Automatic Aesthetic Photo Composition," in *Image Analysis and Processing – ICIAP 2013*, Naples, Springer Berlin Heidelberg, 2013, pp. 21-30.

[14] R. Datta, J. Li and J. Z. Wang, "Algorithmic Inferencing Of Aesthetics And Emotion In Natural Images: An Exposition," in *IEEE International Conference on Image Processing*, San Diego, 2008.

[15] C. Li, A. Gallagher, A. C. Loui and T. Chen, "Aesthetic Quality Assessment Of Consumer Photos With Faces," in *Image Processing (ICIP), 2010 17th IEEE International Conference*, Hong Kong, 2010.

[16] D. Cohen-Or, O. Sorkin, R. L. T. Gal and Y.-Q. Xu, "Color Harmonization," in *ACM SIGGRAPH*, New York, 2006.

[17] Y. Luo and X. Tang, "Photo and Video Quality Evaluation: Focusing on the Subject," in *European Conference on Computer Vision*, Marseille, 2008.

[18] G. Peters, "Aesthetic Primitives of Images for Visualization," in *IEEE International Conference Information Visualization*, Zurich, 2007.

[19] S. Quiller, Color Choices, Watson-Guptill, 2002.

[20] Y. T. X. Ke and F. Jing, "The Design of High-Level Features for Photo Quality Assessment," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[21] P. Obrador, L. Schmidt-Hackenberg and N. Oliver, "The Role Of Image Composition In Image Aesthetics," in *International Conference on Image Processing*, Hong Kong, 2010.

[22] L. Liu, Y. Jin and Q. Wu, "Realtime Aesthetic Image Retargeting," in *International Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, 2010.

[23] L. Wei, X. Wang and X. Tang, "Content-Based Photo Quality Assessment," in *IEEE International Conference on Computer Vision*, Barcelona, 2011.

[24] L. L. Renjie, C. L. Wolf and D. Cohen-Or, "Optimizing Photo Composition," in *International Conference*, Naples, 2010.

[25] W. Luo, X. Wang and X. Tang, "Content-Based Photo Quality Assessment," in *IEEE International Conference on Computer Vision*, Barcelona, 2011.

[26] C. Li and T. Chen, "Aesthetic Visual Quality Assessment of Paintings," *IEEE Journal of Selected Topics in Signal Processing,* vol. 3, no. 3, pp. 236-252, 2009.

[27] R. Sheppard, Landscape Photography: From Snapshots to Great Shots, Berkeley: Peachpit Press, 2012.

[28] S. Baluja, D. Pomerleau and T. Jochem, "Towards Automated Artificial Evolution for Computer-generated Images," *Connection Science,* vol. 6, no. 2 & 3, pp. 325-354, 1994.

[29] J. Romero, P. Machado, A. Carballal and O. Osorio, "Aesthetic Classification and Sorting Based on Image Compression," in *Applications of Evolutionary Computation*, Springer Berlin Heidelberg, 2011, pp. 394-403.

[30] S. Bhattacharya, R. Sukthankar and M. Shah, "A framework for photo-quality assessment and enhancement based on visual aesthetics," in *International conference on Multimedia*, New York, 2010.

[31] J. S. Pedro and S. Siersdorfer, "Ranking and Classifying Attractiveness of Photos in Folksonomies," in *International conference on World wide web*, Madrid, 2009.

[32] W. Jiang, A. C. Loui and C. D. Cerosaletti, "Automatic Aesthetic Value Assessment In Photographic Images," in *Multimedia and Expo (ICME), 2010 IEEE International Conference*, Suntec City, 2010.

[33] L. Itti, C. Koch and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1254 - 1259, 1998.

[34] A. K. Moorthy, P. Obrador and N. Oliver, "Towards Computational Models of the Visual Aesthetic Appeal of Consumer Videos," in *11th European Conference on Computer Vision*, Heraklion, 2010.

[35] R. Datta, J. Li and J. Z. Wang, "Learning the Consensus on Visual Quality for Next-Generation Image Management," in *MM*, Augsburg, 2007.

[36] H. Tong, M. Li, H.-J. Zhang, J. He and C. Zhang, "Classification of Digital Photos Taken by Photographers or Home Users," in *Pacific Rim Conference on Multimedia*, Tokyo, 2004.

[37] H. Tamura, S. Mori and T. Yamawaki, "Textural Features Corresponding to Visual Perception," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 8, no. 6, pp. 460-473, 1978.

[38] L.-K. Wong and K.-L. Low, "Saliency-Enhanced Image Aesthetics Class Prediction," in *IEEE International Conference on Image Processing (ICIP)*, Cairo, 2009.

[39] Desnoyer, Mark; Wettergreen, David, "Aesthetic Image Classification for Autonomous Agents," in *International Conference on Pattern Recognition (ICPR)*, Istanbul, 2010.

[40] J. Machajdik and A. Hanbury, "Affective Image Classification using Features Inspired by Psychology and Art Theory," in *International conference on Multimedia*, Firenze, 2010.

[41] K.-Y. Lo, K.-H. Liu and C.-S. Chen, "Assessment of Photo Aesthetics with Efficiency," in *International Conference on Pattern Recognition*, Tsukuba, 2012.

[42] L. Guo, Y. Xiong, Q. Huang and X. Li, "Image esthetic assessment using both hand-crafting and semantic features," *Neurocomputing,* vol. 143, pp. 14-26, 2014.

[43] S. Dhar, V. Ordonez and T. L. Berg, "High Level Describable Attributes for Predicting Aesthetics and Interestingness," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, 2011.

[44] H. Zhang, E. Augilius, T. Honkela, J. Laaksonen, H. Gamper and H. Alene, "Analyzing Emotional Semantics of Abstract Art Using Low-Level Image Features," in *International Symposium, IDA*, Porto, 2011.

[45] M. Solli and R. Lenz, "Color Based Bags-of-Emotions," in *International Conference, CAIP*, Münster, 2009.

[46] L. Marchesotti, F. Perronnin, D. Larlus and G. Csurka, "Assessing the aesthetic quality of photographs using generic image descriptors," in *IEEE International Conference on Computer Vision*, Barcelona, 2011.

[47] X. Jin, M. Zhao, X. Chen, Q. Zhao and S.-C. Zhu, "Learning Artistic Lighting Template from Portrait Photographs," in *European conference on Computer vision*, 2010.

[48] N. Murray, L. Marchesotti and F. Perronnin, "AVA: A Large-Scale Database for Aesthetic Visual Analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, 2012.

[49] J. Faria, S. Bagley, S. Rüger and T. Breckon, "Challenges Of Finding Aesthetically Pleasing Images," in *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Paris, 2013.

[50] E. Fedorovskaya, C. Neustaedter and W. Hao, "Image Harmony For Consumer Images," in *IEEE International Conference on Image Processing*, San Diego, 2008.

[51] C. D. Cerosaletti and A. C. Loui, "Measuring The Perceived Aesthetic Quality Of Photographic Images," in *International Workshop on Quality of Multimedia Experience*, San Diego, 2009.

[52] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics,* vol. 29, no. 5, pp. 1189-1232, 1999.

[53] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research,* vol. 11, pp. 169-198, 1999.

[54] L. Breiman, J. Friedman, C. J. Stone and R. Olshen, Classification and Regression Trees, Boca Raton: Wadsworth Publishing, 1984.

[55] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning Data Mining, Inference, and Prediction, Stanford: Springer, 2008.

[56] L. Breiman, "Bagging Predictors," in *Machine Learning*, Boston, Kluwer Academic Publishers, 1996, pp. 123-140.

[57] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[58] V. Svetnik, A. Liaw, C. Tong and T. Wang, "Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules," in *International Workshop, MCS*, Cagliari, 2004.

[59] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research,* vol. 3, pp. 1157-1182, 2003.

[60] F. Pan, T. Converse, D. Ahn, F. Salvetti and G. Donato, "Feature Selection for Ranking using Boosted Trees," in *ACM conference on Information and knowledge management*, New York, 2009.

[61] A. Chen, S. Yuan and D. Jiang, "Bagging Based Feature Selection for Dimensional Affect Recognition in the Continuous Emotion Space," in *International Conference on Multimedia Technology*, Guangzhou, 2013.

[62] H. Robert M., K. Shanmugan and I. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 3, pp. 610-621, 1973.

[63] G. W. Snedecor and W. G. Cochran, Statistical Methods, Ames: Iowa State University Press, 1989.

[64] M. G. Kendall, Rank correlation methods, New York: Hafner Publishing Co., 1955.

[65] J. L. Myers, A. D. Well and R. F. Lorch, Research Design and Statistical Analysis: Third Edition, New York: Routledge, 2010.

[66] M. Christie, R. Machap, J.-M. Normand, P. Olivier and J. Pickering, "Virtual Camera Planning: A Survey," in *Smart Graphics*, Frauenwörth Cloister, Springer Berlin Heidelberg, 2005, pp. 40-52.

[67] E. Adams, Fundamentals of Game Design, Pearson Education, Inc., 2014.

[68] S. M. Drucker, *Intelligent Camera Control for Graphical Environments, Phd Thesis,* Cambridge: Massachusetts Institute of Technology, 1994.

[69] B. Tomlinson, B. Blumberg and D. Nain, "Expressive Autonomous Cinematography for Interactive Virtual Environments," in *International conference on Autonomous agents*, Barcelona, 2000.

[70] A. Liapis, G. N. Yannakakis and J. Togelius, "Adapting Models of Visual Aesthetics for Personalized Content Creation," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 4, no. 3, pp. 213-228, 2012.

[71] N. Shaker, G. N. Yannakakis and J. Togelius, "Crowd-Sourcing the Aesthetics of Platform Games," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 5, no. 3, pp. 276-290, 2012.

[72] K. Svobodova, P. Sklenicka, K. Molnarova and J. Vojar, "Does the composition of landscape photographs affect visual preferences? The rule of the Golden Section and the position of the horizon," *Journal of Environmental Psychology,* vol. 38, pp. 143-152, 2014.

[73] M. A. Munson and R. Caruana, "On Feature Selection, Bias-Variance, and Bagging," in *European Conference, ECML PKDD*, Bled, 2009.

[74] The MathWorks, Inc., "MathWorks - MATLAB and Simulink for Technical Computing," The MathWorks, Inc., 2015. [Online]. Available: http://www.mathworks.com/. [Accessed 16 7 2015].