IMAGE BASED SOLAR POSITION EDITING AND RELIGHTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT TÜRE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2018

Approval of the thesis:

IMAGE BASED SOLAR POSITION EDITING AND RELIGHTING

submitted by **MURAT TÜRE** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ahmet Oğuz Akyüz _____
Supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Tolga Kurtuluş Çapın
Computer Engineering Department, TEDU _____

Assoc. Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Aykut Erdem
Computer Engineering Department, Hacettepe University _____

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MURAT TÜRE

Signature :

ABSTRACT

IMAGE BASED SOLAR POSITION EDITING AND RELIGHTING

TÜRE, MURAT
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Ahmet Oğuz Akyüz

June 2018, 54 pages

Capturing perfect photos is a challenging task that requires good composition and timing. Failure to achieve them may result in 'missing the moment'. For example, when shooting photos at sunset the sun quickly disappears behind the horizon. In this thesis, a new image editing algorithm is defined to let the user change the sun position in a photo. This is the first study in the literature which tries to solve this particular image editing problem. The algorithm can be divided into three parts. First, the image pixels are segmented into different categories, namely the sky, clouds, sun, horizon, and the foreground. Then, a real-time rendered sky is replaced with the old sky to allow changing the position of the sun. A color transfer algorithm is used to cope with the differences that cannot be accurately deduced from a single photo such as exposure, atmospheric parameters, and the camera response curve. Furthermore, techniques for relighting the cloud and foreground pixels are proposed. Finally, these relighted pixels are blended together by using the labels from the segmentation part of the algorithm to generate the final image. The proposed image editing technique works in real-time to provide instant feedback to the user. It also provides intutive user controls to allow overriding and fine-tuning various automatically detected parameters.

Keywords: Image Editing, Rendering of Atmospheric Scattering, Image Segmenta-

tion, Color Transfer, Fully Convolutional Neural Networks

ÖZ

GÖRÜNTÜ TABANLI GÜNEŞ POZİSYONU DEĞİŞTİRME VE YENİDEN IŞIKLANDIRMA

TÜRE, MURAT
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Ahmet Oğuz Akyüz

Haziran 2018 , 54 sayfa

Fotoğrafçılar için mükemmel anı yakalamak her zaman zor olmuştur. Fotoğrafın kontrol edilemeyen kısımları en iyi kompozisyonun yakalanmasını imkansız hale getirebilir. Bu çalışmada, kullanıcının fotoğraftaki güneş pozisyonunu değiştirebileceği yeni bir görüntü düzenleme tekniği önerilmiştir. İlk adım olarak fotoğraf piksel bazında gruplara ayrırlır. Bu gruplar gökyüzü, bulut, güneş, ufuk çizgisi ve ön plan olarak tanımlanmıştır. İkinci olarak fotoğraftaki eski gökyüzü gerçek zamanlı bir atmosfer ışıklandırma algoritmasının çıktısı ile değiştirilir. Kamera ve atmosfer gibi gökyüzü görüntüsünü değiştirebilecek farklılıkları önlemek için bir renk transfer tekniği bu yeni gökyüzüne uygulanır. Bunun ardından, güneş pozisyonu farklı gökyüzü fotoğraftaki karşılığı ile değiştirilir. Son olarak da fotoğraftaki ön plan ve bulut kısımları yeni gökyüzüyle uyumla olacak şekilde yeniden ışıklandırılır. Bu teknik gerçek zamanlı bir şekilde güneşin oynatılabilmesine olanak sağlarken aynı zamanda otomatik bulunan bir çok parametrenin de kullanıcı tarafından kolaylıkla değiştirilebilmesi için arayüz sunar.

Anahtar Kelimeler: Görüntü Düzenleme, Atmosferik Saçılmanın Işıklandırılması,Görüntü Segmentasyonu, Renk Transferi, Tamamen Katlamalı Sinir Ağları

ACKNOWLEDGMENTS

I would like to thank my advisor Assoc. Proc. Dr. Ahmet Oguz Akyuz for his huge support for my thesis. He was the one who not only teached me Computer Graphics but more important than that made me love Computer Graphics. I can not repay his contributions to my academic work. I will never forget that.

I would like to thank my thesis committee members Prof. Dr. Tolga Kurtulus Çapın, Assoc. Prof. Dr. Ahmet Oguz Akyuz, Assoc. Prof. Dr. Aykut Erdem, Assoc. Prof. Dr. Sinan Kalkan, Assoc. Prof. Dr. Yusuf Sahillioglu for their valuable feedback.

I would like to thank my mother and father who always be there for me when problems seem to be impossible to solve. I would not be here without their help.

I would like to thank my colleagues Gökhan Uras and Burak Dermanlı. They were always there when I had questions and it was always inspiring for me to see their passion about Computer Graphics. I know that they will always stay hungry and never stop being the individuals who improves their surroundings with their sheer will. I also thank my old mentor and lead Serdar Koçdemir for molding my character by being a role model for me.

The 'Presidente' and my fellow brother Burak. I hope you and Gülsüm will have the happiest life. I will never forget the conversations we had, meals we shared at late hours and homeless looking joggings we had in the winter. I can never thank you enough for the support you gave me.

Finally, I would like to thank my fellow Bard, engineer and scientist Serdar Yonar. Even though we can not see each other that frequently, I know that you are doing great out there. Never stop.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

Abbreviations

BRDF Bidirectional Reflectance Distribution Functions

CNN Convolutional Neural Network

CRF Conditional Random Field

FCN Fully Convolutional Network

GMM Gaussian Mixture Model

HDR High Dynamic Range

LDR Low Dynamic Range

CHAPTER 1

INTRODUCTION

Capturing the perfect photo is the ultimate aim of every professional and amateur photographer. Perfect photos are often taken when the alignment of many different aspects like the orientation and position of the foreground objects as well as an aesthetic and meaningful background are established. Obtaining the desired composition for the foreground objects are generally easier than obtaining a good composition for the background. Especially, in outdoor scenes, the background is typically dynamic and cannot be changed at the will of the photographer. In particular, photographs containing the sun are difficult to capture due to the constant motion of the sun in the sky and the appearance changes induced by this motion in the foreground. As an example, a photographer may desire to capture a partial sunset together with a particular dynamic foreground composition. However, despite the foreground being ready, there may still be some time for the sun to reach the desired elevation. In this case, the photographer must either wait for this moment risking the distortion of the foreground or settle for an image with a non-ideal background. Furthermore, when the scene is ready to be captured the photographer must make fast decisions to avoid missing the perfect picture.

There are many image editing tools that allow post-processing a captured photograph to obtain the desired look-and-feel. There are many filters in popular image editing software such as Photoshop to introduce various effects [28]. Similarly, popular photo sharing platforms, such as Instagram[1] allow the user to apply various effects to their pictures. In the literature, there are also many image editing techniques that enable advanced effects such as image based material editing [20], inserting new objects into

[1] http://www.instagram.com

photographs [22], and replacing the sky of an existing photograph [35].

This thesis proposes a similar advanced image editing algorithm particularly targeting an aspect of image editing that has been hitherto overlooked. The proposed approach involves making modifications to the position of the sun as a post-processing effect. The input to the algorithm is a single photograph. The user can make adjustments by selecting the sun and changing its position in the sky. The proposed approach allows not only simple changes but also drastic modifications such as setting the sun behind the horizon, bringing back up a partially set sun, changing its both azimuthal and elevation angles, entirely removing the sun, or adding the sun to a picture that is devoid of it. Such modifications entail overall changes to the photograph so that the entire photograph is consistent with respect to the new sun position. This thesis attempts to show that such advanced image editing effects are not only possible but can be applied in real-time to allow a fully interactive solution.

At the heart of the proposed algorithm lies a two-step sky segmentation in which a coarse labelling of sky pixels is obtained in the first step [35] followed by sky refinement [31] in the second. This segmentation permits re-rendering of the sky regions using an atmospheric scattering algorithm [16]. The result of this rendering is smoothly blended using a color transfer algorithm [29] with the original sky to stay faithful to the original picture. The foreground regions of the original sky (i.e. clouds) are extracted, relighted with respect to the new sun position, and blended back onto the image. Furthermore, the foreground regions of the original photograph are also relighted to obtain a consistent look with respect the current sun position. All of these operations are performed in real-time to allow instant feedback to the user. The proposed algorithm is mostly automatic. However, intutive controls are provided to the user to override certain parameters that are automatically extracted and provide fine-tuned control on the appearance of the final image.

To summarize, the primary contributions of this thesis are:

- First image-based sun position modification algorithm in the literature,
- Realistic handling of various appearance effects that stem from updating the sun position,

- A real-time implementation allowing all modifications to be fully interactive.

The rest of the thesis is organized as follows. Chapter 2 first describes the previous image editing studies and then overviews the fundamental algorithms that are used in the current study. In Chapter 3, the proposed algorithm is explained in detail. This is followed by the experimental results shown in Chapter 4. Limitations of the suggested approach are described in Chapter 5. Finally, the conclusions and potential future improvements are given in Chapter 6.

CHAPTER 2

RELATED WORK

Image editing is an extensively studied subject within the computer graphics literature, and as a result, a large body of research exists that may serve as the background for this thesis. In this chapter, this literature is organized into two categories. In the first category, image editing techniques that are related to the presented work are discussed. In the second category, various techniques that serve as the building blocks of the proposed method are reviewed.

2.1 Image Editing

Image editing refers to the notion of processing an input image to produce an output image with desired characteristics. Such processing may occur at the global level changing the entire appearance of the image or may be confined to local regions [27].

There are many image editing algorithms that are proposed in the literature. In Poisson image editing [27], different editing tools are used with guided interpolation by solving the Poisson equations. These tools vary from seamlessly transporting a part of one image between images to changing the affect of the texture, the illumination or making a texture tileable.

Khan et al. [20] devised an algorithm to change the BRDF of an object inside an high dynamic range image and relight it with respect to its surroundings in the image. This process is tailored to highly exploit the tolerance of human vision. The algorithm can be divided into several parts of finding the object shape, calculating the environ-

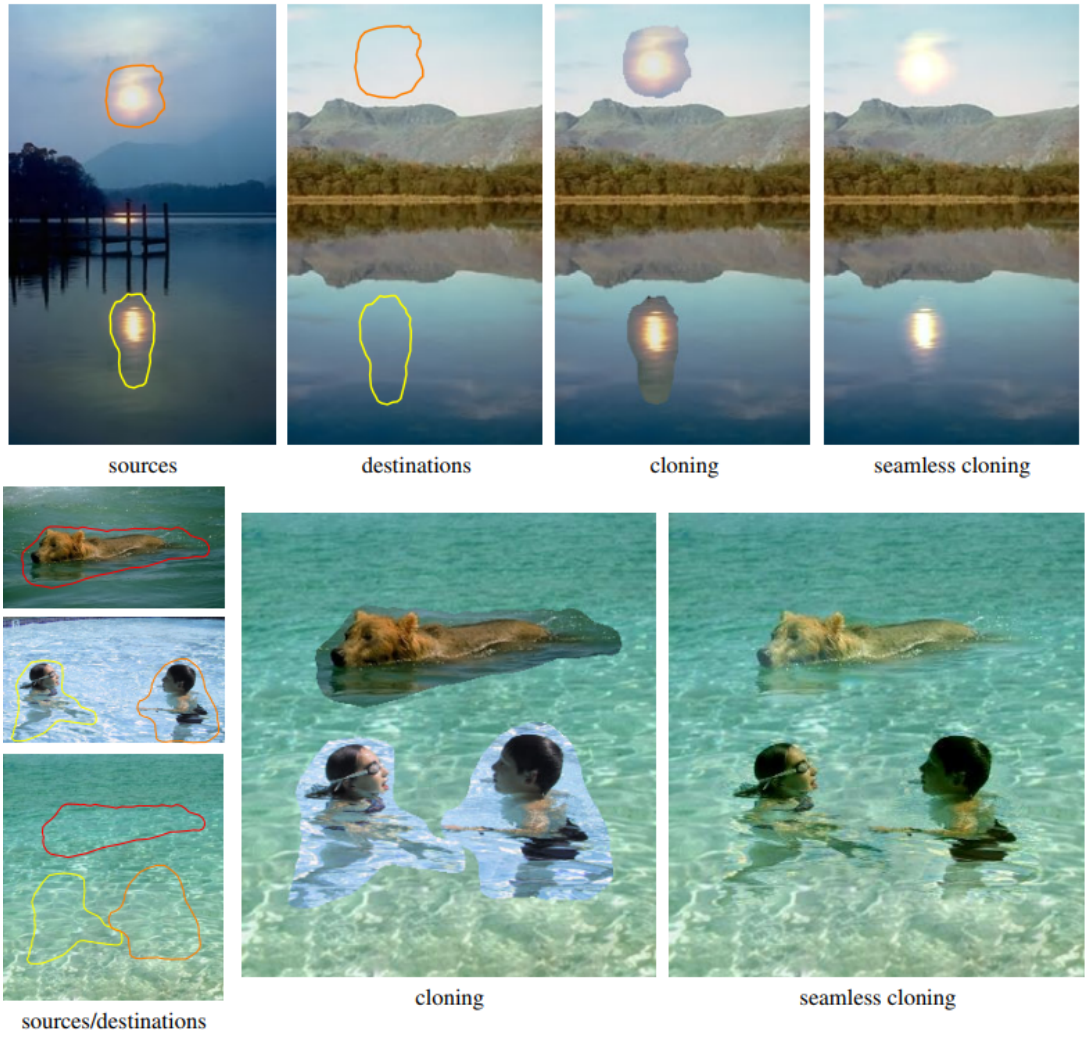


Figure 2.1: Poisson image editing allows the user to transport a part of one image to another seamlessly. Image taken from [27].

Figure 2.2: The material properties of the object in the input image on the left can be changed to be a transparent object or have a different BRDF. Image taken from [20].

ment lighting and with the new material properties relighting of the object. Firstly, the sigmoidal compression is used to capture the depth of the object pixels. Then, by using the screen space gradients of the depth values, the normal of the object is approximated. Furthermore, in order to find the lighting properties of the environment, the mesh pixels are removed and a hole filling algorithm is used to fill these pixels with other pixels from the image. The center of the new image without the object is placed into a two dimensional cartesian space and then warped to create a hemisphere which has the same center. This hemisphere is duplicated in the other side to create the HDR environment map. Finally, the BRDF of the image is changed and the lighting is sampled from the calculated HDR environment map. Also, a texture can be wrapped around the object with the gradient field calculated at the first step.

Konstantinos [30] used convolutional neural networks to find the shape, material and illumination of an object inside an image. A CNN is trained to find the reflectance map of the objects inside an image. Reflectance map is the mapping of surface orientation and properties of an object to its appearance. This is done with the assumptions of constant material, distance light source and viewer.

Shih et al. [32] proposes an algorithm to change the time of day of a photo. This algorithm uses a data-driven approach. The color transfer from one frame of the video (for example sun rise) to another (night) is learned by a model. A matching time lapse video is found for the input image and the affine color transfer for that frame is applied to the input image. The overall algorithm is depicted in Figure 2.4.

As one of the more related algorithms to the current thesis, Tsai et al [35] proposed an algorithm to change the sky with another one from a suitable image. First, the scene is segmented by an FCN. Afterwards, with a online classifier, the result of

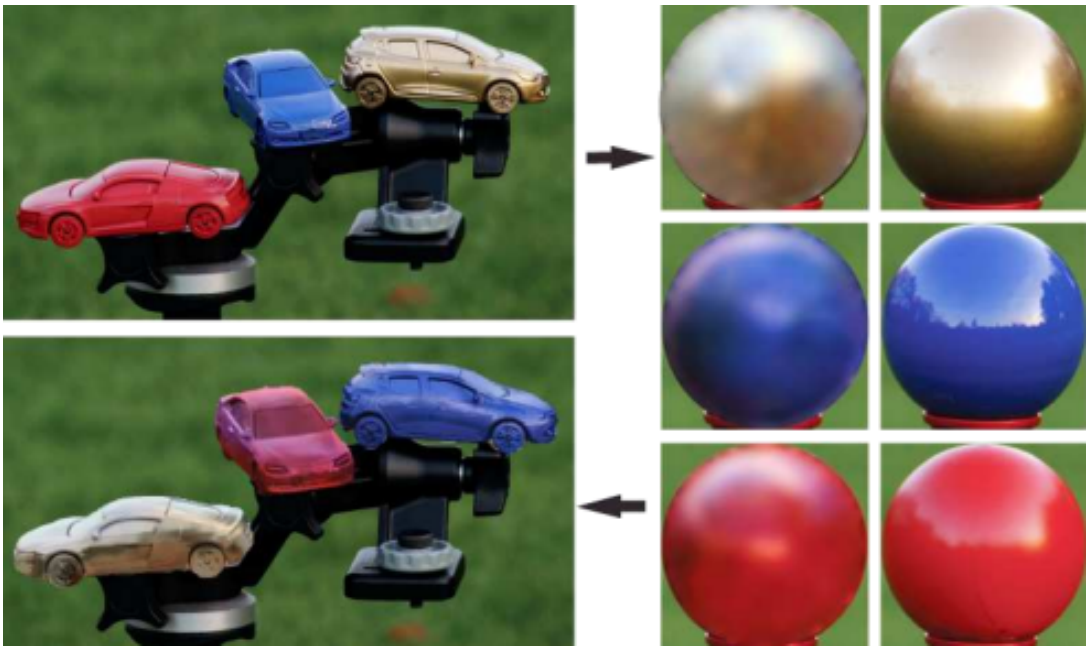


Figure 2.3: Using the input image in the top left, the reflectance maps (right side) can be found. Then the appearance of the cars can be changed by changing the reflectance maps of the objects. Image taken from [30].

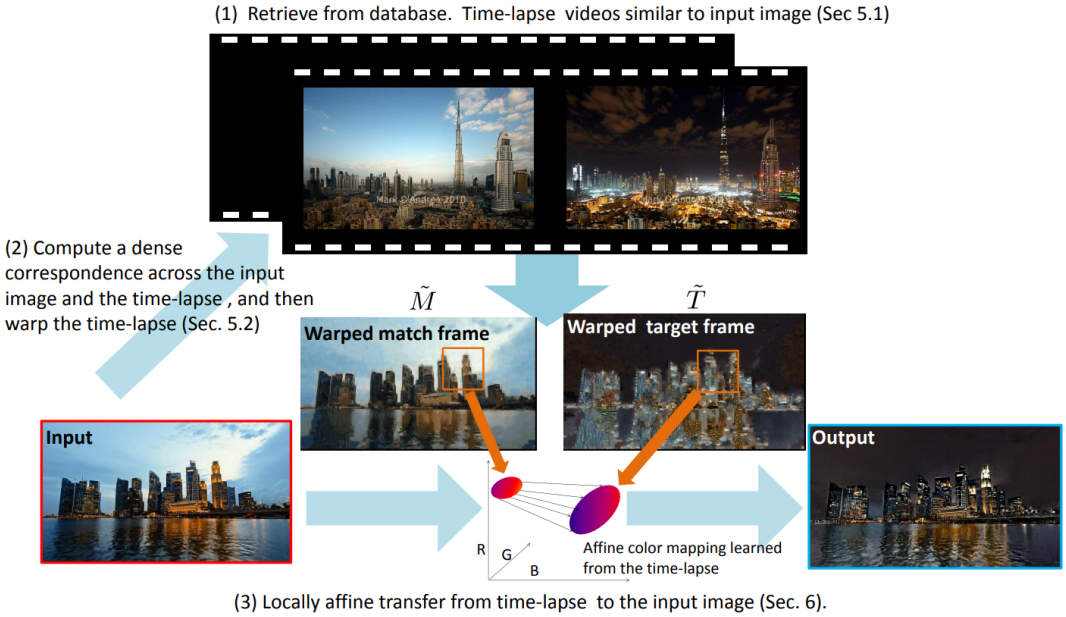


Figure 2.4: The overall algorithm of the Data-driven Hallucination of Different Times of Day from a Single Outdoor Photo. Image taken from [32].

(a) The first and the third images are input while the second and fourth are the sky replaced ones. Image taken from [35].

Figure 2.5: Results of the semantic aware sky replacement algorithm.

the FCN is refined to match with the input resolution. Then, by using the semantic information about the image, images with a matching sky box are found to be used as a replacement for the sky in the original image. To correctly update the color of the foreground, the semantic data from both images is used. Every pixel is updated with the mean colors of the pixels from the other images which are at the same labels. Soft blending of multiple label levels are also used in this relighting process to increase the visual quality.

2.2 Background

In this section, various techniques that are used in the current thesis and therefore serve as its backround are reviewed.

2.2.1 Scene Segmentation

Segmentation of an image into predefined labels is one of the problems that Computer Vision algorithms try to solve. Nowadays, learning based techniques dominate the computer vision research with their state-of-the-art performance on several benchmarks. In this section, Fully Convolutional Networks (FCN) [24], Grabcut [31], and color based cloud segmentation techniques [29] will be discussed.

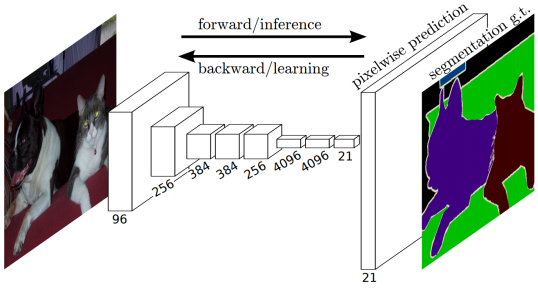


Figure 2.6: Fully convolutional networks can easily learn and perform per pixel segmentation tasks. Image taken from [24].

2.2.2 Fully Convolutional Neural Networks

Convolutional nueral networks are commonly used to solve semantic segmentation problems where the task is to predict a category label for each pixel in a given image.

Long et al. [24] used Fully Convolutional Neural Networks for the scene segmentation problem. These networks significantly increase the performance and robustness of the per pixel scene segmentation tasks. However, the output resolution is defined in the model and constant for every input image. The model and the overall algorithm of [24] can be seen in Figure 2.6.

2.2.3 Grabcut

Boykov et al. [15], in their seminal paper, define the segmentation problem as an energy minimization problem. Firstly, the image is transformed into a graph with the following rules. The initial foreground and background pixels are the sink nodes of the graph. Every other pixel in the image is inserted as an intermediate node. Connection between every pixel and their neighbours are defined as the edges in the graph. Then, for every node a probability of being in whether the foreground and background labels is computed. The energy of the edges which are the edge weights are then defined as the difference of these probabilities of the edge nodes. Thus, one can define finding the best segmentation of the images as finding the minimum cut for this graph. Finally, a globally satisfying graph cut can be found in linear time with max-flow/min-cut algorithm [18]. Rother et al. [31] improved the graph cut

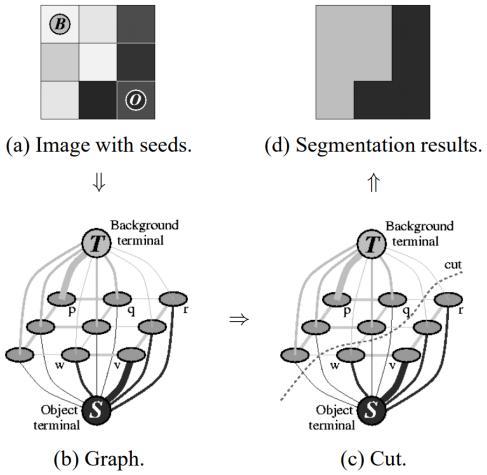


Figure 2.7: Representation of an image as a graph and solving the segmentation problem with min cut algorithm. Image taken from [15].

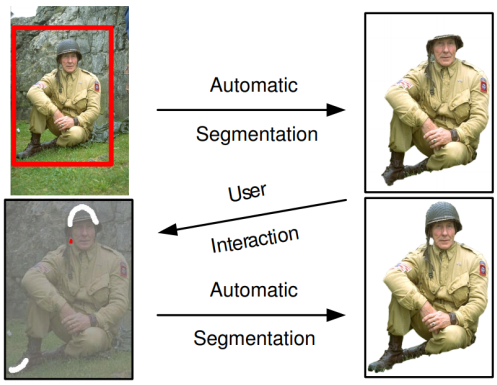


Figure 2.8: Grabcut takes the image and a rectangle as inputs. After the segmentation is finished, user can use strokes to further refine the result. Image taken from [31].

segmentation algorithm and named it Grabcut. Firstly, the gray scale color spaced used in the original method is replaced with Gaussian Mixture Model (GMM). A GMM value is calculated as a Gaussian mixture for both background and foreground. The algorithm starts with the bounding box for the foreground object as an user input. Then, the GMM values for labels are initialized with respect to the pixels that are inside and outside of this rectangle. For every iteration, the min cut is found and the result is used to better estimate the values of these GMM's. Finally, if there are errors in some areas of the image, user can draw some strokes to change values of some pixels. Then by using this information the minimum cut algorithm is run once more and the result is the final segmentation of the foreground and the background pixels.

2.2.4 Cloud Segmentation

Cloud segmentation is used frequently at the atmospheric observation algorithms. Robust segmentation of the clouds in an image is essential at the weather analysis processes. The most robust algorithms to solve this problem usually uses color based techniques.

Long et al. [23] calculates the ratio of red and blue channels. Then this value is compared against a threshold to determine whether a pixel is cloud or not. Heinle et al. [19] uses the difference of the red and blue channels to label the pixels. Souza et al. [34] computes the Saturation of the pixel color and use it to find the label. Finally, Dev et al. [17] uses a learning based approach to segment clouds without the fine tuning that the older techniques needed.

2.2.5 Sky Rendering

Accurate capture of atmospheric scattering effects is the key to render realistic skies and big landscapes. Lots of different techniques are published to capture the scattering effect of light while passing through a dense medium. The scientific notation from [16] is used.

Earth's atmosphere layer is a thick spherical layer around the planet. It has a different density of air molecules at every height and it is responsible for the atmospheric effects that is visible everyday and every hour. When light passes through a medium, two things happen. It can get absorbed or it can get scattered away in a different direction. Air molecules and aerosol particles are the two main medium that is the reason of these effects in our atmosphere.

The light that is scattered away at θ degrees away, can be found by the product of scattering coefficient B^s and phase function P. For the air molecules, Rayleigh theorem states that:

$$B_r^s(h, \lambda) = (5.8, 13.5, 33.1) \tag{2.1}$$

$$P_r = \frac{3}{16\pi}(1 + \pi^2) \tag{2.2}$$

Notice that the blue component of the scattered light is more than the other components. This is the reason for the sky appearing blue to our eyes. This happens most when the sun is at the top of the sky. However, when the sun is rising or setting, the most scattered wavelengths cannot reach our eyes resulting in an orange hue at these times of the day. Also the lack of an atmosphere results in a sky where only the direct sun light is visible. This is indeed the case at the moon surface as can be seen in Figure 2.9.

(a) Clear sky. (b) Sky of moon.

Figure 2.9: Blue is the dominant color that is scattered at the sky. Meanwhile, absence of a atmospheric layer results in a black moon sky. From left to right, images are taken from [8], [13].

For aeresol particles , Mie theorem defines the B^s and P as:

$$B_m^s(h,\lambda) = B_M(0,\lambda)exp(-h/H_M) \tag{2.3}$$

$$P_m = (3/8\pi)(1-g^2)(1+\mu^2)/(2+g^2)(1+g^2-2g\mu)^{3/2} \tag{2.4}$$

With these information, the light that is scattered to eye position x from the position x_0 can be defined. This x_0 can be at the surface of the planet or a point at the sky. v is the direction from x_0 to x and s is the direction of sun direct light.

First the absorption function $T(x, x_0)$ for the light that comes from x_0 to x is defined as:

$$T(x,x_0) = \exp\left(-\int_x^{x0} B^s(y)dy\right). \tag{2.5}$$

Note that the amount of sun light is L_{sun}. L_0 which is the contribution of direct sun reflected at the position x_0 and arrived at x is defined as follows:

$$L_0(x, x_0) = T(x, x0)L_{sun}, \tag{2.6}$$

and R be the sum of every light that is reflected from x_0 and arrived at the x:

$$R(x, x_0) = T(x, x_0)(a(x0)/\pi) \int_{2\pi} L(x_0, w, s)w.n(x_0)dw. \tag{2.7}$$

Also, the sum of the scattered light from $x0$ to x in the direction v is defined as S as follows.

$$S(x, v, s) = \int_x^{x0} T(x, y) \left(\int_{4\pi} B^s(y)P(v, w)L(y, w, s)dw \right) dy. \tag{2.8}$$

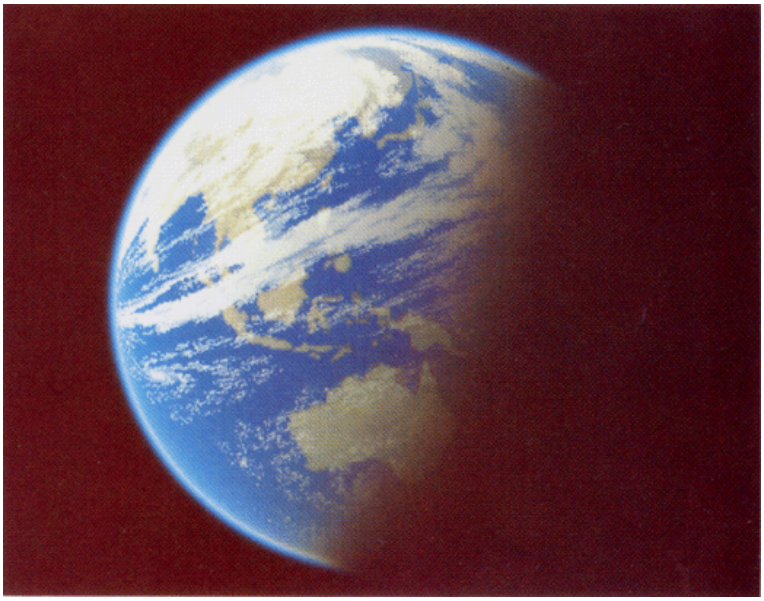
Finally, with all this formulas, total light that is reflected to x from x_0 namely $L(x, x_0)$, can be defined as:

$$L(x, v, s) = L_0(x, v, s) + R(x, v, s) + S(x, v, s). \tag{2.9}$$

Computationally, the most expensive term is S, as it requires the result of every scattering event along the way to x. Now, the proposed methods in the literature to solve L will be investigated.

2.2.5.1 Rendering Single Scattering Visible from Space

Nishita et al. [26] is the first to propose an algorithm for atmospheric rendering. This work tries to solve the single scattering equation for the air particles with a numerical approximation. The atmosphere layer is discretezed as thin spherical layers. Two directions for light is taken into account. The direction from x to x_0 and from x_0 at s direction to sun. The inner integral in S is computed iteratively. The result is the ability to render the earth and the sea from space.



(a) Earth viewed from space

Figure 2.10: Output of the algorithm of Nishita. Image taken from [26].

2.2.5.2 Rendering Multiple Scattering

In the following years, Nishita improved his algorithm to account for multiple scattering and the ability to view from inside the atmosphere layers.

In order to cope with the high order of integral to render the multiple scattering effect, Nishita et al. [25] proposed to divide the sky into voxels in which every voxel is defined as a viewing direction to sky with respect to the viewer. Then, the scattering effects at every reflection of the light ray is incrementally updated with respect to the previous scattering.

2.2.5.3 Sky Rendering in a Real time Game Engine

In a real-time game engine, Wenzel et al. [36] was one of the first to render realistic atmospheric scattering effect taking into account the Mie and Rayleight in-scattering. A 128×128 grid is generated for the sky and the formula L is computed at this resolution. This computation is done in pixel shaders at the GPU. This data is independent of camera direction and only gets updated once the sun direction changes. This pro-

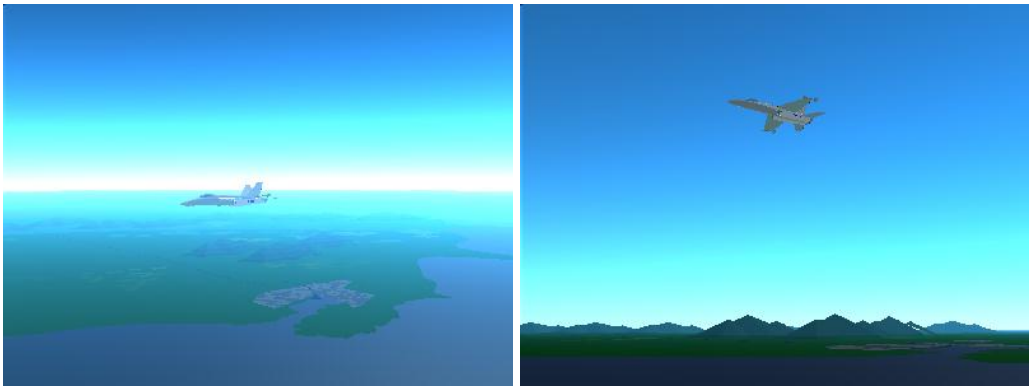


Figure 2.11: Computer generated images using multiple atmospheric scattering effect. Image taken from [25].

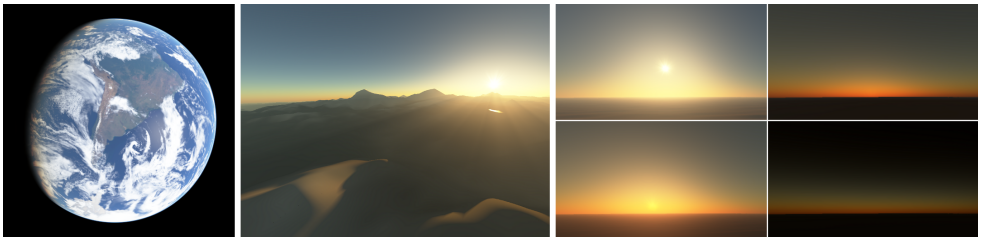


Figure 2.12: Precomputed 4D scattering data also allows the user to change the camera height without any new computation. Images are taken from [16].

cess of re computing is done in several frames to ensure smooth transition of time of day. The only assumption was that the camera is at ground level and infinitely far away from the sky.

2.2.5.4 Precomputed Atmospheric Scattering

Bruneton et al. [16] introduced their work to precompute the multiple scattering of atmospheric particles at a four dimensional spatial space. The results can be obtained in real-time and unlike previous methods, the height of the camera from the ground is taken into account. This enables the algorithm to render the atmospheric scattering at every height, from ground to space.

2.2.6 Color Transfer

Transferring the characterics one image to another is a widely encountered problem in many areas. These techniques can be used to enhance a photo to be more appealing to the human perception. Also, one can use to generate new photos by merging two photos together.

One of the pioneering color transfer techniques was proposed by Reinhard et al. [29]. In this method, the goal is to transfer the colors of one image to the second image. First, both images are converted into the $L\alpha\beta$ color space. Per channel mean and standart deviation of the two images are calculated. Denoting the mean of the images as M_0 and M_1 and the standard deviations as S_0 and S_1. R is the ratio of the standard deviation of the second image and the first image.

$$R = S_1/S_0 \tag{2.10}$$

In order to transfer the first image into the second image, the following equation is used : Let the initial pixel value of the images in LaB space are L_0 and L_1. The mean of the first image is subtracted from every pixel in the first image. Then, the intermediate pixel values found at the last step is multiplied by R. Finally, the mean of the second image is added to the current values of pixels:

$$L' = ((L_0 - M_0)R) + M_1 \tag{2.11}$$

This operation is applied to each color component and the result is converted back to the red green blue (RGB) color space.

CHAPTER 3

ALGORITHM

The proposed algorithm takes two inputs which are the input image to be edited and a user-click representing the position of the sun. The editing process is accomplished in three stages. These are

- The segmentation of the scene into labels such as sky, clouds, foreground, sun and the horizon,
- Sky rendering based on the new position of the sun,
- Relighting of the foreground and the clouds to make them consistent with the new sun position.

An overview of the proposed algorithm is shown in Figure 3.1. Also, the notation used throughout the algorithm can be seen in Table 3.1.

3.1 Image Segmentation

In order to correctly change the appearance of the image, the scene should first be segmented into its categories with meaningful labels. The different object categories important for this thesis are the sky, sun, clouds, foreground and the horizon (Figure 3.2).

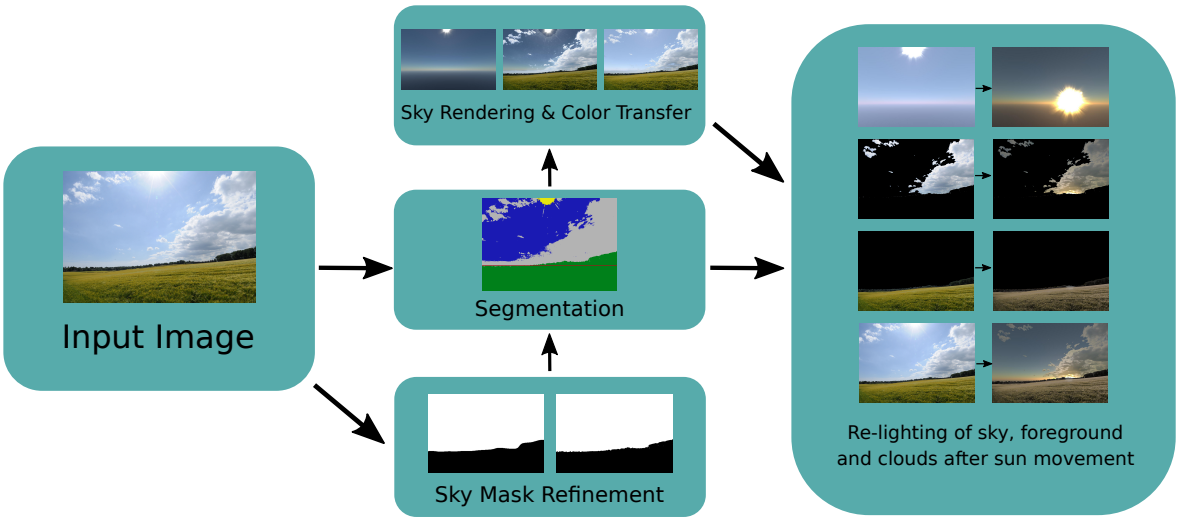
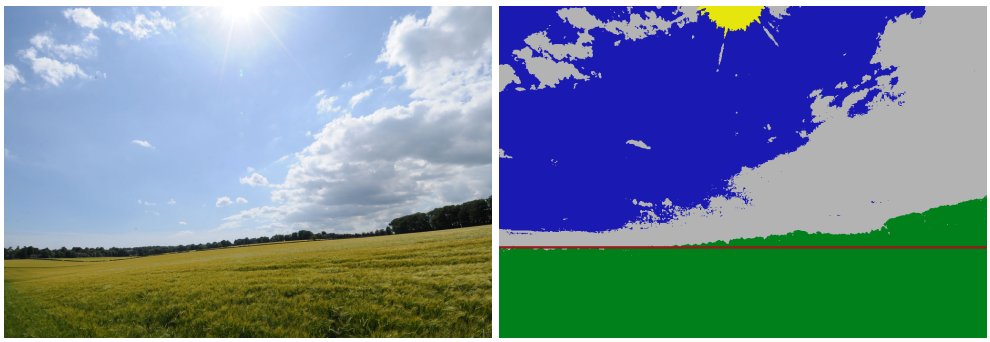


Figure 3.1: Overview of the proposed algorithm. Input image taken from [1].



(a) Input image (b) Computed labels

Figure 3.2: An input image and the corresponding segmentation. Input image taken from [1].

Each category requires a different segmentation approach as explained below.

3.1.1 Sky Segmentation

The proposed algorithm starts with finding a binary mask for the sky pixels. This mask will be used for further segmentation steps as well as the relighting the entire

Notation	Meaning	Dimension
M	Crude Sky Mask	WxHx1
M^u	Upsampled Crude Sky Mask	WxHx1
$M^{u,r}$	Refined Sky Mask	WxHx1
C^m	Cloud Mask	WxHx1
C^s	Cloud Softness Mask	WxHx1
C	Original Cloud Color	WxHx3
C^r	Relighted Cloud Color	WxHx3
C^r_{soft}	Relighted Soft Cloud Color	WxHx3
C^r_{thick}	Relighted Thick Cloud Color	WxHx3
S	Original Sky Color	WxHx3
S^r	Re-rendered Sky Color with Color Transfer	WxHx3
B^r	Relighted Background Color	WxHx3
I^r	Final Image	WxHx3

Table 3.1: List of the terms used in the proposed method

image with the new sun position.

3.1.1.1 Coarse Sky Segmentation

To segment the sky pixels of the input image, a fully convolutional network (FCN) that represents the state-of-the-art for semantic segmentation is used [24]. The already trained model of [38] and [39] is used. The output of this is the probability map for each pixel of the input image, $P(i, j, k)$, where i and j are the pixel coordinates and k is the label index. For every pixel, the most probable label index $\bar{K}_{i,j}$ is defined as follows:

$$\bar{K}_{i,j} = \arg\max_k P(i, j, k) \tag{3.1}$$

The coarse sky mask, $M(i, j)$, is then defined as follows:

$$M(i,j) = \begin{cases} 1 & \text{if } \bar{K}_{i,j} = K_{\text{sky}} \text{ ,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

For computational efficiency, the FCN produces the segmentation results in 384×384

resolution. Therefore, the sky mask computed in Equation 3.2 has also low resolution. This mask is upsampled to produce, M^u, which is the sky mask in the original image resolution. However, this upsampling process results in a crude sky mask that must be refined. A sample result of this algorithm is depicted in Figure 3.3

(a) Input image (b) Upscaled crude sky mask

Figure 3.3: Input image and the crude sky mask gathered with the FCN algorithm. Mask is scaled to the size of the input image. Input image taken from [1].

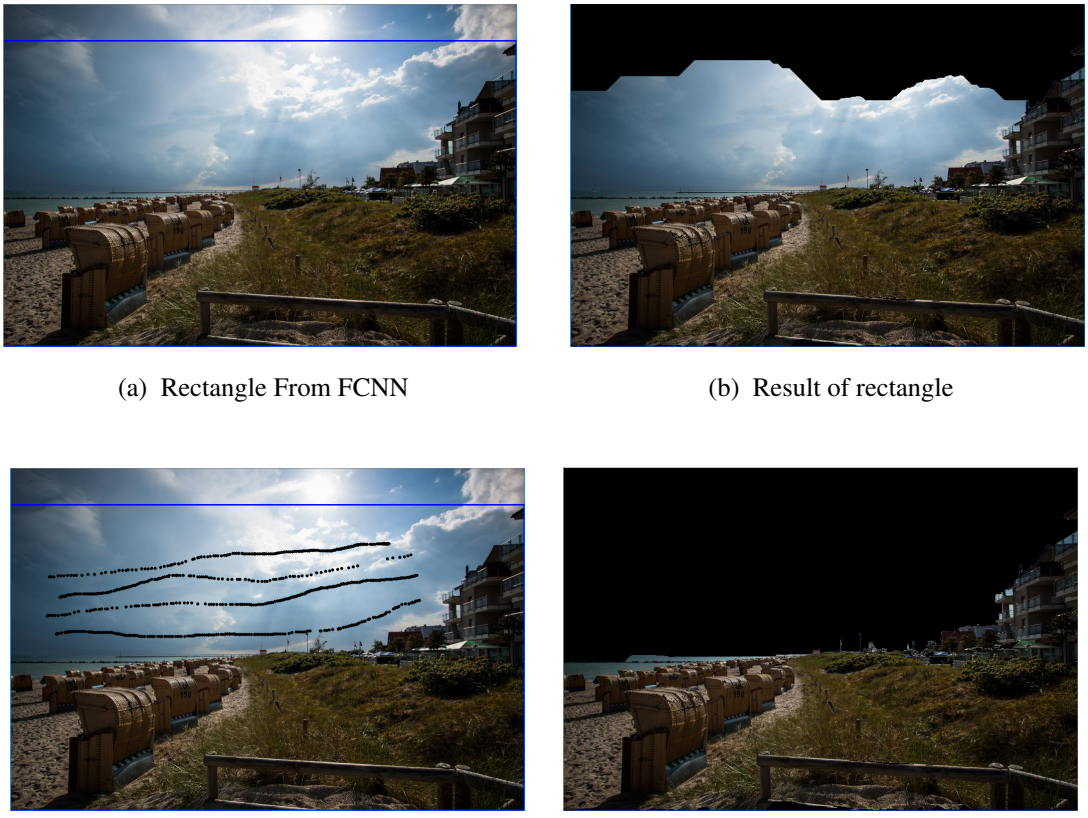
3.1.1.2 Sky Mask Refinement

In order to refine the sky mask various techniques were tested. Initially, custom heuristics based on detecting edges in the original image in the neighbourhood of the crude sky mask contours were employed. However, these did not provide satisfactory results in different images in a consistent manner. Next, experiments were conducted to use a dense conditional random field (CRF) [21]. This algorithm uses the initial segmentation results as unary potentials and computes pairwise potentials between all pairs of pixels. However, the outputs of this algorithm were not satisfactory in all cases as well. The best results were obtained using Grabcut segmentation [31]. Grabcut takes the original image as well as a screen-space bounding box of the foreground. This bounding box is computed using the minimum and maximum pixel coordinates of M^u matrix. The results after the Grabcut refinement are shown in Figure 3.4.

(a) Bounding box (b) Input Image (c) Refined sky mask

Figure 3.4: Using the input image and the foreground rectangle information as inputs, Grabcut refines the coarse sky mask generated by FCN. Input image taken from [1].

However, in pictures that have complex skylines, using the bounding box only may result in incorrect segmentation. For these cases, an interface is provided for user to draw strokes on the background to assist Grabcut in segmentation. The results of this approach are shown in Figure 3.5.



(a) Rectangle From FCNN (b) Result of rectangle

(c) User stroke input (d) Result with user stroke

Figure 3.5: An example showing a wrong segmentation with the bounding box of FCNN segmentation and the correction from user stroke. Input image taken from [2].

3.1.2 Horizon Line Detection

Detecting the horizon line is critical for the correct rendering of the sky when the position of the sun is changed by the user. For instance, when the sun is set, the most intense orange hues will be visible around the horizon in the regions close to the sun. Therefore if the horizon line is detected in an incorrect position this colouring effect will take place in an incorrect location. To find the horizon line, first an edge detection is performed along the vertical axis of the upsampled and refined sky mask, $M^{u,r}$:

$$F(i,j) = |M^{u,r}(i,j) - M^{u,r}(i,j-1)|, \tag{3.3}$$

where $F(i,j)$ represents the binary edge mask. The horizon line is assumed to be at the j coordinate which has the maximum number of edge pixels in this coordinate:

$$\arg\max_{j} \sum_{i=0}^{\text{width}-1} F(i,j). \tag{3.4}$$

This algorithm works well for landscape images (Figure 3.6) but may fail in images where objects that occlude the horizon are close to the camera (e.g. a building with a flat-roof). To cope with this, a user interface control is provided to allow the user to modify this incorrect horizon line.

Figure 3.6: Computed horizon line position for several inputs. From left to right images are taken from [5], [6], [7].

3.1.3 Sun Segmentation

In the previous part, the cloud mask is found by using the ratio of red and blue channels. This techniques also labels the sun pixels as cloud because of the high red

channel values at the sun pixels. To cope with this, the set of sun pixels should be found. Flood fill algorithm is used to label the sun pixels [33].

At the beginning of the application, user clicks the sun to move it. The flood fill algorithm starts with that pixel. Let the first pixel be P_0. Three structures are used. Firstly, a stack S to hold the non processed neighbours , an array F which holds the pixels already labelled as sun and a set P to hold the already processed pixels are defined. F and S is initialized with P_0. Then, at every iteration, a pixel is popped from S. If the luminance difference between P_0 and itself is lower then a threshold M, it is pushed into F. Also, in that case every neighbour which is not processed before is pushed into the S. Finally, current pixel is inserted into the processed set P. The algorithm is finished when the S is empty and F denotes the sun pixels. This technique can be seen in Algorithm 1.

Algorithm 1: Sun segmentation algorithm.

Input: I : input image, P_0 : coordinate of the starting pixel

Output: F : set of sun pixel coordinates

```
1  Function SegmentSun(I):
2      while P is not empty do
3          if |I(current) − I(P_0)| < M then
4              F.push(neihbour)
5              for neighbour ← current.neighbours do
6                  if P.contains(neighbour) is false then
7                      S.push(neighbour)
8                  end
9              end
10         end
11         P.push(current)
12     end
13     return F
```

(a) Input image (b) Sun segmentation mask

Figure 3.7: Sun segmentation mask in the second image is calculated with a flood fill algorithm from the first image and the user click on the sun. Image taken from [1].

3.1.4 Cloud Segmentation

In order to blend and relight the cloud pixels successfully, a cloud mask should be calculated. The segmentation technique from [23] is used to determine whether that pixel is cloud or not. The output of this method is a binary mask. First, the ratio of red and blue channels of the sky pixels are found. Then, the pixel is segmented as cloud if this ratio is bigger than a threshold m_{min}. The output mask C^m is defined as follows:

$$C^m = \begin{cases} 1 & \text{if } (r/b) > m_{\text{min}} \text{ ,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

The reflectance of the cloud pixels change rapidly with respect to their density. Thick clouds does not let the light pass through itself, so their bluish appearance governed by the light that is scattered between the clouds and the viewer. Using this observation, it can be assumed that while the soft clouds will be bright, the color of the thick cloud pixels will be more like a darker blue sky. To correctly relight thick cloud pixels, a per pixel softness map is needed. To find this softness mask C^s, just like the cloud mask, the ratio of the red and blue channels are used. However, this time, C^s is a gray scale mask, where the 0 value indicates that the ratio is at the minimum threshold m while a value bigger than 1 indicates that the ratio is equal or bigger than a maximum threshold which is defined as m_{max}. The calculation of the C^s is as follows:

$$C_s = ((r/b) - m_{min})/(m_{max} - m_{min}) \tag{3.6}$$

Figure 3.8: Left image is the cloud softness map M_s and the right one is the cloud mask M_c.

The output masks of the cloud segmentation algorithm can be seen in Figure 3.8

3.2 Sky Rendering

At this point, horizon line and the sun position is found. Using this information a camera can be calibrated to match the sky rendering algorithm to the original image as explained below.

3.2.1 Assumptions about the Input Image

Several assumptions are made about the camera that captured the input image. A fixed field of view for the sky rendering algorithm is used throughout the algorithm. When this assumption does not match with the reality, the shape of the scattering effect around the sun can be different than the original image. Also, the elevation of the camera from the ground is set to 0 throughout the technique.

3.2.2 Calibration and Blending

With the information gathered from photo and the assumptions made in Section 3.2.1, the camera and sun direction properties of the sky rendering algorithm can be adjusted. First, the sun position is found by getting the center pixel of the sun mask. Then, the camera is rotated up or down till the horizon line in the rendered sky

(a) Input image (b) Calibration of rendered sky

Figure 3.9: Using the sun position and horizon line, the rendered sky is calibrated to match with the input image. Input image taken from [1].

(a) Input image (b) Rendering result

Figure 3.10: The sky pixels of the left image are updated using sky rendering algorithm. Input image taken from [1].

matches with the horizon in the image. The result of this calibration can be seen in Figure 3.9.

At this point, the rendered sky can be blended with the first image by using the refined sky mask $M^{u,r}$. Note that this blending also uses the cloud mask C so that the cloud pixels are used from the original image. The result of this operation can be seen in Figure 3.10.

3.2.3 Matching Camera Properties

There are information about the image that can not be deduced from one photo. They are, exposure, camera response curve, and densities of atmospheric particles. To cope

with this, Reinhard's Color Transfer algorithm [29] is used to match the visual of rendered sky with the input image. Unlike the original algorithm, only the pixels that were segmented as sky pixels are used. This ensures that, without any sun movement, the new image looks as similar as possible to the old one. The result of the color transfer can be seen in Figure 3.11.

(a) Input image (b) Rendering result (c) Result with color transfer

Figure 3.11: Reinhard et al.'s color transfer algorithm [29] is used to reduce differences with the rendered and original sky. Input image taken from [1].

3.3 Foreground and Cloud Relighting

When the sun is moved by user and the appearance of the sky is changed, the color of foreground and cloud pixels should be changed to match with the new lighting environment. Foreground and cloud pixels are treated differently because of their differences in surface shape and reflectance.

3.3.1 Foreground Relighting

Let $\bar{S}$ and $\bar{F}$ represent the mean colors of the sky and foreground regions in the original picture, respectively. Denote their per-component ratio by $Q = \bar{S}/\bar{F}$. After updating the sky color as explained in Section 2.2.5.4, the mean foreground color is recomputed to preserve this ratio, i.e. $\bar{F}' = Q \times \bar{S}^r$ where $\times$ represents per-component multiplication. To achieve this, each foreground pixel is scaled by the factor of $\bar{F}'/\bar{F}$.

$$F^r = (\frac{\bar{F}'}{\bar{F}} \times F). \tag{3.7}$$

This effect can be seen in Figure 3.12.

Figure 3.12: After each movement of the sun, the new mean color of the sky is calculated and the foreground is relighted with the equation. It can be seen that as the sun is going down, the overall color of the foreground becomes more orange. Image taken from [8].

However, when the input image is LDR, the mean color of the sky pixels does not represent the lighting environment successfully. In real world, the illumination amount of direct sun pixels are very high with respect to the sky pixels. However, the sun pixels in the LDR input image are generally white while some clouds can be at the some color as well. In order to cope with this, the power of the sun pixels is amplified by a factor F^s. This ensures that the color change of the sun and occlusion of the sun pixels effects the overall color of the foreground at a much higher rate and in a more realistic way. The factor F^s is used as 1000 for all of the input images.

Another key observation about the outdoor lighting is that the direct sun contribution is much more powerful than the scattered sky colors and the transmitted sun rays through the clouds. With this, it can said that when the sun is occluded by the clouds, not only the color schema is changed to a more blue tint, the overall contrast of the image is lowered. This assumption generally holds as long as the image is not fully shadowed in the initial state.

To accommodate for this phenomena, the occlusion ratio for the sun pixels is found after every movement of the sun and the overall contrast of the foreground is reduced with respect to this occlusion value. A new variable s is introduced which is the cosine power percentage of the visible sun pixels. Then, a colourfulness modulation formula inspired from tone mapping to reduce the contrast of the foreground final colors is employed. The final foreground relighting algorithm for the pixels is modified as follows:

$$F^r = \left(\frac{C'_f}{L'_f}\right)^s L'_f, \tag{3.8}$$

where L'_f is the luminance computed from the color C'_f after Equation 3.7 and s is

the contrast term for the foreground relighting algorithm. As the sun is occluded with clouds , $s < 1$ will result in reduced colorfulness. The effect of this can be seen in Figure 3.13.

Figure 3.13: The contrast of the foreground pixels is reduced with the factor s. The left image is the relighted foreground without this effect while contrast reduction is present at the right one. Image taken from [1].

3.3.2 Cloud Relighting

The process of relighting the clouds can be divided into two parts. Changing the appearance of the soft and thick clouds. Following masks were found in the cloud segmentation chapter. S^m determines whether there that pixels is cloud or not. C^s is the mask for cloud softness and it's value is higher at the pixels with denser clouds.

The overall cloud relighting algorithm can be expressed with the following equation:

$$C^r = C^r_{soft} C^s + C^r_{thick}(1 - C^s). \tag{3.9}$$

C^r_{thick} and C^r_{soft} are the relighted colors of the thick cloud pixels and the soft cloud pixels. Once these are found, the final cloud color can be found by blending those together with the thickness mask C^s. According to the atmospheric scattering algorithm, one can assume that when the clouds does not pass any sun light, the color of that pixel is determined by only the scattered light from that point to the camera. Thus it can be stated that the color of that cloud will be proportional to the sky light from that same direction if there were no cloud at that direction. Using this observation, the formula for relighting a thick cloud pixel can be derived. Note that S is defined as the first color of the re-rendered sky pixel. Then, S^r is defined as the current color

of the rendered sky after the movement of the sun. Then, the relighted color of thick cloud pixels, namely C^r_{thick}, can be found by changing C by preserving the ratio of S_r and S after the movement of the sun. Now, $C^r_{th,ck}$ can be defined as follows:

$$C^r_{thick} = \frac{S^r C}{S}. \tag{3.10}$$

In order to relight the soft cloud pixels, some observations needs to be made. Firstly, the sun can be directly visible beneath a soft layer of clouds. F' is defined as the new sun mask after the movement of sun. With this information, C'_{s0} can be defined as follows:

$$C'_{s0} = SF' + C(1 - F'). \tag{3.11}$$

Secondly, sun light can increase the intensity of soft cloud layers around whole cloud clusters and this can be approximated with Lambert's cosine law. Since the normals of the cloud pixels are unknown, a blurred version of the sky rendering is used to approximate the color changes in clouds around the sun. The blending factor is defined with a simple variable which changes with respect to the distance to the sun. Blending factor B_f is defined as follows:

$$B_f = 1 - \min(\frac{d_c}{d_m}, 1). \tag{3.12}$$

d_c is the current distance to sun pixel in texture space while the d_m is the maximum distance where the blend factor becomes 0. Also, B_{fp} is the smoothness factor for the blending factor. The blurred version of the sky rendering is defined as S_b. Then, final soft cloud color C^r_{soft} can be calculated with equation:

$$C^r_{soft} = C'_{s0} S_b B f + C'_{s0}(1 - B_f) \tag{3.13}$$

Finally, the relighted cloud pixels, namely C^r can be found by blending the relighted soft and thick color pixels with the cloud softness mask C^s. This can be seen at the equation 3.14. Input and relighted colors of clouds can be seen in Figure 3.14.

$$C^r = C^r_{soft} C^s + C^r_{thick}(1 - C^s) \tag{3.14}$$

(a) Cloud pixels in the input image (b) Relighted cloud pixels

Figure 3.14: The output of the cloud relighting method.

(a) Input image (b) Output image

Figure 3.15: The sun in the input image is moved by the user and the image appearance is changed accordingly with the proposed algorithm. Image taken from [1].

3.3.3 Generating the Final Image

The final color of the background(sky) pixels, namely B^r, can be found by blending the relighted sky S^r and relighted cloud C^r pixels with respect to the cloud mask C^m. This process is shown at the equation 3.15.

$$B^r = C^r C^m + S^r(1 - C^m) \tag{3.15}$$

Finally, the pixel colors of the output image, namely I^r, can be found by blending the relighted foreground pixels F^r and the background pixels B^r using the background mask $M^{u,r}$:

$$I^r = M^{u,r} B^r + (1 - M^{u,r}) F^r. \tag{3.16}$$

The output of the proposed algorithm can be seen in Figure 3.15.

3.3.4 Performance

This technique heavily uses the Graphics Processing Unit. On the other hand, the Central Processing Unit usage is very low. The run time processing can be group into three categories. There are

- Computing the atmospheric scattering values
- Render the final sky, cloud and foreground pixels
- Computing the mean colors of foreground and sky to be used next frame

The first and the third items are computed in compute shaders while the second one is rendered with a pixel shader. The following performance measurements are done with a Nvidia Gtx 760 graphics card and with an input of 1024x698 resolution. The results can be seen at Table 3.2.

Atmospheric scattering	21.2 ms
Rendering final image	3.2 ms
Mean color computation	2.4 ms
Total	28.8 ms

Table 3.2: Run time performance counters of the technique. Units are in milliseconds.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, the results of the algorithm will be presented with a variety of input images. These images were collected with only the assumption that the sky is visible. The sun starts with its initial position and is moved across the sky by the user. If there is no sun in the image, one will be added by clicking at a random point in the sky. The results are presented in two categories with respect to the presence and abscence of clouds.

4.1 Clear Sky

In this section, the result of the input images that have no clouds in their sky region will be demonstrated. These images commonly have a blue-yellow lighting environment. The altitude of the sun will lowered and the results will be discussed.

Once the sun is moved towards the horizon by the user, the mean color of the sky transforms into a more orange hue. According to the proposed foreground relighting algorithm, the mean color of the foreground will be changed as well. This can be seen in Figures 4.1 - 4.5.

Another thing to note that is the importance of the sun segmentation results. In Figure 4.6 the scattering effect is powerful around the sun and the threshold used for the flood fill algorithm is too high for that input. Because of this, the sun mask is larger than what is supposed be. The effect of the bad segmentation is most visible at the sunset. Once the false positive sun pixels starts to get occluded by the horizon, the

Figure 4.1: Input image, and relighted outputs after incremental sun movements. Note that as the sun is moved towards horizon color of the foreground pixels is changed to a more orange hue. However, the relighting of the sea is not successful. Image taken from [9].

Figure 4.2: Input image, segmentation mask, and relighted outputs after incremental sun movements. The foreground relighting looks natural because of the foreground pixels with high diffuse reflection. Image taken from [8].

Figure 4.3: Input image, segmentation mask, and relighted outputs after incremental sun movements. When the sun is halfway to the horizon the lack of the scattering calculation in foreground pixels is visible as the scattering effect can not be modified correctly by the foreground relighting algorithm. Image taken from [10].

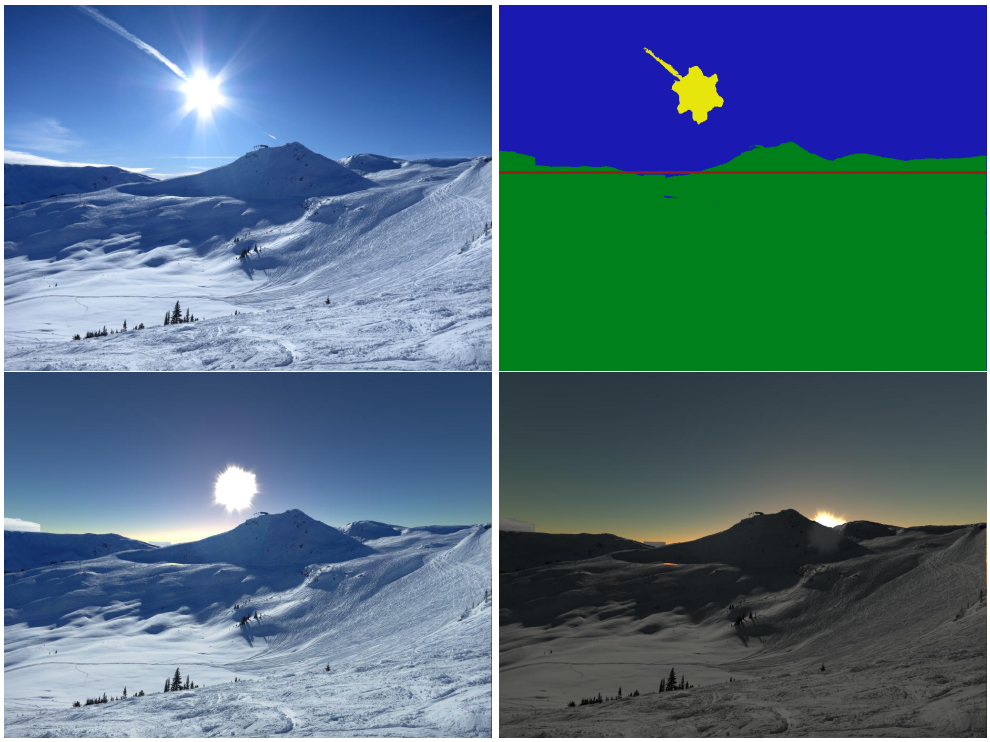


Figure 4.4: Input image, segmentation mask, and relighted outputs after incremental sun movements. The faulty segmentation of foreground lowers the visual quality of the output. Image taken from [6].

Figure 4.5: Input image, segmentation mask, and relighted outputs after incremental sun movements. Image taken from [7].

Figure 4.6: Input image, segmentation mask, and relighted outputs after incremental sun movements. Image taken from [11].

mean color of the sky reduces too rapidly with the sun amplification and this results in a darker foreground.

In Figure 4.2, the scene segmentation algorithm labels the cloud pixels as foreground. This negatively impacts the relighting result on these pixels. Soft cloud should pass more light and the general foreground relighting algorithm can not capture this phenomena.

While the distance to the camera increases or the altitude of the sun decreases the scattering effect on the earth grows stronger . Proposed algorithm does not handle this with the lack of the depth information. This results in a bad relighting at the horizon line because there should be a more blurry and strong orange hue at those pixels. This can be seen in Figure 4.3.

The foreground relighting algorithm works well in pixels with a strong lambertian reflection property. However, surfaces like sea or metal can not be accurately relighted. Also, the proposed technique is unable to add or move the specular highlights. This reduces the quality of the output images. This can be seen in Figures 4.1, 4.7.

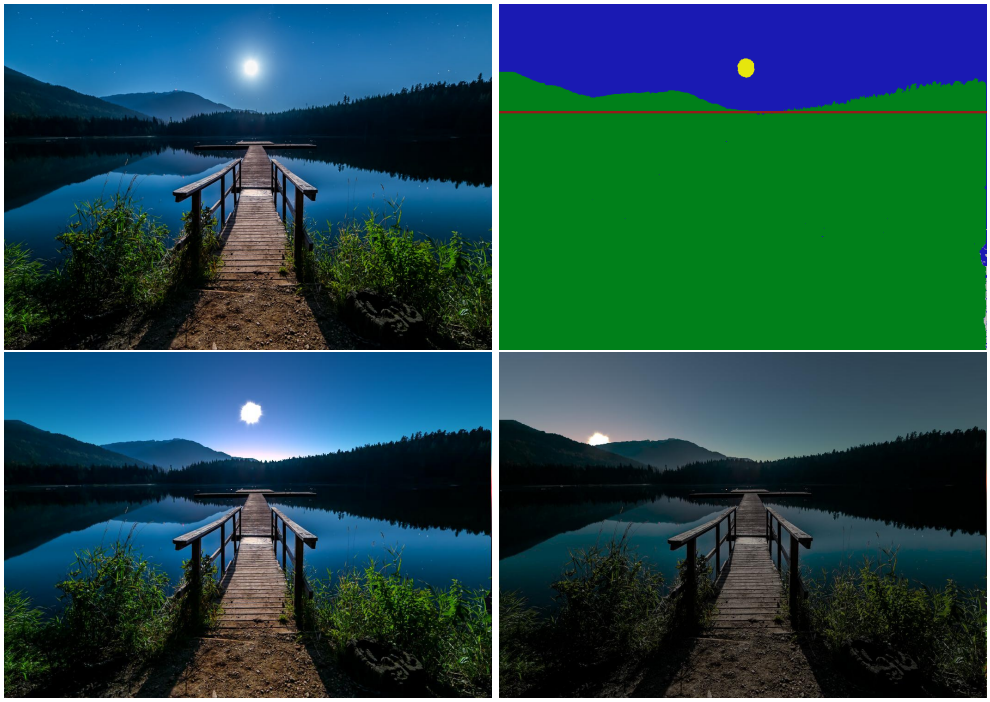


Figure 4.7: Input image, segmentation mask, and relighted outputs after incremental sun movements. A night scene can be turned into a sunset. Image taken from [12].

One interesting result can be seen in Figure 4.7. The photo is taken at night. The only visible light source in the scene is the moon. The proposed algorithm can segment it as sun. The overall color of the sky is dark. With the color transfer, the rendered sky is transformed to be dark, too. At the end, the output image looks like a dark sun set. However, the lack of the reflection change in the water highly reduces the output quality.

4.2 Cloudy Sky

The way clouds interact with light is very different because of their volumetric properties. The thickness of the volume directly impacts the light interaction. Soft cloud pixels transmit more light while the thick clouds pixels only reflects the scattered light on their surfaces. Handling this difference correctly is the most important point in the cloud relighting algorithm.

As the cloud segmentation uses the ratio of red and blue channels, all of the clouds can only be segmented correctly when this ratio is distinguishable in sky and cloud pixels.

Figure 4.8: To prevent false positive segmentation, some cloud pixels are sacrificed. This results in a less dense cloud field. Image taken from [3].

This happens only when there is no bright sky or dark cloud pixels. In these cases, to not have false positive cloud pixels in the scattered areas, a high cloud threshold is chosen at the cloud segmentation algorithm. This results in missing clouds, especially the thick ones. This can be seen in Figure 4.8.

In Figure 4.9, it can be seen that the best cloud relighting results can be obtained when the sky and cloud pixels are easy to segment. Also, a better visual quality can be achieved when the histogram of the cloud softness mask C^s is uniform which means that the input image has clouds with many levels of thickness. The cloud relighting algorithm works well with both very thick, very soft and also for clouds in between by using blending for the intermediate pixels. Also, the effect of the contrast reduction from sun occlusion for the foreground can be seen in Figure 4.9.

Figure 4.9: If the cloud softness mask C^s has enough contrast, cloud relighting algorithm works well.This is indeed the case in this input. Also, the effect of the foreground contrast reduction is visible. Image taken from [1].

Figure 4.10: A cloud mask with hard edges reduces the visual quality of the output.

As mentioned before, proposed algorithm can also insert sun into photos without one. Figure 4.10 is an example for that. However, the hard edges of the cloud mask C^m lowers the visual quality of the output.

CHAPTER 5

LIMITATIONS

The challenge in this thesis is that our problem requires solving many subproblems. Many of those problems are tackled by making many assumptions about the properties of the scene, atmosphere or camera. This forces the algorithm to have many parameters and the selecting the optimal parameters are crucial to the overall success of the algorithm. Also, some natural phenomena like the scattering effect on the foreground, direct shadows, specular reflections on water or metallic objects or correct calculation of indirect lighting was missing because of the lack of information like depth, geometry and BRDF of these surfaces. In this chapter, the limitations caused by aforementioned issues will be discussed.

5.1 Incorrectly Placed Horizon Line

The horizon line calculation algorithm reviewed at Subsection 3.1.2 finds the vertical line with the most edge pixels in the sky mask. This technique fails when the camera is tiled upwards or a big object is near the camera and occludes the sky. The vertical line is found on the upper sections of the occluder object. Using this horizon line results in a stretched sky rendering with a high scatter effect with a dominant orange color. As the color transfer technique [29] works in a global scale , it can not undo this error. Figure 5.1 demonstrates this issue.

Figure 5.1: In this input the horizon line is estimated to be higher than its correct level. This results in a premature sun set. This gives the false impression that the mountains in the image are actually a lot smaller. Image taken from [4].

Figure 5.2: High threshold value is used to segment the clouds. This prevents scattered pixels to be segmented. However, some parts of the cloud layer is lost. Image taken from [3].

5.2 Cloud Segmentation Parameters

The cloud segmentation algorithm defined at the Subsection 3.1.4 uses the ratio of red and blue channels of the sky pixels to segment the cloud pixels. A minimum and maximum value is defined for the ratio and the ratio is linearly transformed into this region to have a 0 value at the minimum and 1 at maximum. This makes the tweaking of these values paramount for the success of the cloud segmentation. Images that have sky pixels with a higher ratio then the thick cloud regions can not be segmented correctly this technique. In these cases, either false positive cloud pixels emerge from some sky pixels or some thick cloud pixels are labelled as sky pixels. Generally, the latter is chosen in the algorithm. Result of this can be seen in figure 5.2.

5.3 Foreground Depth

The geometric shape of the sky can be modelled without much information. This enables the sky rendering to achieve realistic results without any other information. However, lack of the surface depth forces the algorithm to disregard many effects in foreground relighting.

The scattering algorithm at Chapter 2 dictates that the scattering effect on an object increases with the distance to the camera. This effect also gets amplified when the sun's altitude is lower. Currently, the atmospheric scattering algorithm is only run for the sky pixels at which the depth can be estimated by simple arithmetic. However, if depth of the foreground pixels were available, the scattering effect could have been applied to the foreground as well. Lack of this information, reduces the visual fidelity of images with deep horizon lines.

Direct sun light is the most powerful light sources in the input images. This increases the contrast between the occluded and not occluded pixels very. Also, the direction and intensity of the shadows are a strong visual cue for the human perception. Like scattering, lack of depth and geometry information forces the proposed algorithm to neglect shadows. If the depth information were available, the foreground relighting algorithm would be very different. Firstly, the algorithm proposed by Xu et al. [37] can detect and remove the shadow pixels from an image. After that, screen space ray tracing techniques [14] can be used to insert both new shadows and calculate better ambient for the pixels according to the new sun direction. This would probably be one of the best improvement for the proposed algorithm.

5.4 Surface BRDF

The reflectance of a surface governs how that surface interacst with light. Bidirectional Reflection Distribution Function (BRDF is a four dimensional function which takes position, normal, light direction and view direction as input and outputs the color of that position. Many models are proposed to mimic the real life visuals of many different materials. Without the BRDF of the pixels, the relighting algorithm

Figure 5.3: After the sun movement the water pixels can not be relighted correctly without the BRDF. Image taken from [9].

is not capable of handling specific types of material. For example, the specular highlight on the sea, which is the result of a high gloss and high specular BRDF, does not change when the sun moves. This is easily recognized by the human perception and reduces the overall quality of the output. This can be seen in Figure 5.3.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, a brief summary of the algorithm that this thesis propose will be given. Furthermore, possible, and most important improvements that can be done in the future will be discussed.

6.1 Summary

This thesis proposes an algorithm to change the sun position in a photo. First, the image is segmented in per-pixel fashion into several categories. Then, with this information, the sky is re-rendered using atmospheric scattering rendering algorithms and blended with the old one. Using the color values of this new sky, the clouds and the foreground is relighted to match with the new lighting environment. All of these new information is blended by the segmentation masks calculated in the first part to produce the final image.

6.2 Conclusions

The results section shows that this algorithm can generate successful results in a wide section of inputs. Proposed technique can handle the clear sky images easily and does not require too many tweaking to the parameters. A photo that were captured at midday can be transformed into sunsets or sunrises. Even though the field of view or horizon line can be different from the estimations, the results are visually pleasing.

Meanwhile, the cloudy images are harder to edit. The success of these images depend on the cloud segmentation algorithm. Using the ratio of red and blue channels of the sky pixels to segment clouds does not work very well at many of the inputs. When the cloud softness mask has a uniform histogram the output of the cloud relighting is successful and visually pleasing. Also, not being able to capture a soft blending mask for the clouds can generate a hard lined cloud layer in the output.

Images with diffuse surfaces and soft shadows give better results. However the ones with materials like water or metal can not be relighted correctly without the reflectance information. Furthermore, the algorithm can not handle shadows which is easily detectable by human perception when the shadow is sharp and ambient lighting is low.

6.3 Future Work

The foreground relighting works by changing the mean of the foreground pixels with respect to the new rendered sky. However, there many different lighting phenomena that is not uniform at every pixel of the foreground. For example scattering effect can not be applied to the foreground pixels because of the lack of depth information. With a depth estimation algorithm , the atmospheric scattering technique can also calculate the scattering effect of the foreground as well. This would increase the visual quality of the results especially when the sun is setting or rising.

One other lighting parameter is the normal of the surfaces in the image. Light reflected from a surface is highly dependent on this parameter. The gradient of the depth can be used to generate a view space normal and then the foreground can be relighted by using this new information.

The least successful inputs for out algorithm is the images with high specular surfaces like water or metal. The proposed foreground relighting algorithm is not able to move the specular reflections on these images after the sun movement. To fix this, reflectance inference algorithms can be used to first segment these surfaces and the high specular pixels. Then, these pixels can be filled by the surface color and the new specular reflection can be added by taking into account the new sun position.

A user study can be made to test the output of the algorithm by a group of user. The images that were altered with the algorithm can be mixed with other landscape photos. The questions:

- Which photo looks more realistic
- Which photo looks more pleasing

can be asked.

Finally, the results of this algorithm can be compared with the ground truth. A frame from a time lapse video can be given as input and the output can be compared with the corresponding frame. Also, computer generated images can be used to check not only the final output of the algorithm, as well as the outputs from the intermediate steps.

REFERENCES

[1] Input 0. https://www.flickr.com/photos/lo_ise/4696937075/in/gallery-160672184@N05-72157689442588790/.

[2] Input 1. https://www.flickr.com/photos/stephanma/15562839041/in/faves-160672184@N05/.

[3] Input 10. https://www.flickr.com/photos/antonnovoselov/8719805389/in/gallery-160672184@N05-72157689442588790/.

[4] Input 11. https://www.pexels.com/photo/cold-daylight-landscape-mountain-279492/.

[5] Input 2. https://www.flickr.com/photos/125343440@N08/20063352700/in/faves-160672184@N05/.

[6] Input 3. https://www.flickr.com/photos/ruthanddave/39729273041/in/faves-160672184@N05/.

[7] Input 4. https://www.flickr.com/photos/armin_vogel/8401844416/in/faves-160672184@N05/.

[8] Input 5. https://www.pexels.com/photo/clouds-daylight-highway-landscape-461823/.

[9] Input 6. https://www.pexels.com/photo/seagull-soaring-on-top-of-pebble-field-at-beach-733292/./.

[10] Input 7. https://www.pexels.com/photo/countryside-daylight-grass-hd-wallpaper-568236/.

[11] Input 8. https://www.pexels.com/photo/man-wearing-gray-long-sleeved-polo-shirt-near-dock-989206/.

[12] Input 9. https://www.pexels.com/photo/beautiful-beauty-blue-bright-414612/.

[13] Moon atmosphere. https://www.pexels.com/photo/astronaut-standing-beside-american-flag-on-the-moon-39896/.

[14] Louis Bavoil and Miguel Sainz. Screen space ambient occlusion. *NVIDIA developer information: http://developers. nvidia. com*, 6, 2008.

[15] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[16] Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. *Computer Graphics Forum*, 27(4):1079–1086, 2008.

[17] Soumyabrata Dev, Yee Hui Lee, and Stefan Winkler. Color-based segmentation of sky/cloud images from ground-based cameras. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):231–242, 2017.

[18] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[19] Anna Heinle, Andreas Macke, and Anand Srivastav. Automatic cloud classification of whole sky images. *Atmospheric Measurement Techniques*, 3(3):557–567, 2010.

[20] Erum Arif Khan, Erik Reinhard, Roland W Fleming, and Heinrich H Bülthoff. Image-based material editing. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 654–663. ACM, 2006.

[21] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[22] Jean-François Lalonde, Derek Hoiem, Alexei A Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. *ACM transactions on graphics (TOG)*, 26(3):3, 2007.

[23] Charles N Long, Jeff M Sabburg, Josep Calbó, and David Pagès. Retrieving cloud characteristics from ground-based daytime color all-sky images. *Journal of Atmospheric and Oceanic Technology*, 23(5):633–652, 2006.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[25] Tomoyuki Nishita, Yoshinori Dobashi, Kazufumi Kaneda, and Hideo Yamashita. Display method of the sky color taking into account multiple scattering. In *Pacific Graphics*, volume 96, pages 117–132, 1996.

[26] Tomoyuki Nishita, Takao Sirai, Katsumi Tadamura, and Eihachiro Nakamae. Display of the earth taking into account atmospheric scattering. In *Proceedings*

of the 20th annual conference on Computer graphics and interactive techniques, pages 175–182. ACM, 1993.

[27] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 22(3):313–318, 2003.

[28] Adobe Photoshop. Adobe photoshop 5.0 limited edition, chapter 4: Making color and tonal adjustments. *Jan*, 1:67–89, 1998.

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

[30] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. Deep reflectance maps. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 4508–4516. IEEE, 2016.

[31] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[32] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Transactions on Graphics (TOG)*, 32(6):200, 2013.

[33] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[34] MP Souza-Echer, EB Pereira, LS Bins, and MAR Andrade. A simple method for the assessment of the cloud cover state in high-latitude regions by a ground-based digital camera. *Journal of Atmospheric and Oceanic Technology*, 23(3):437–447, 2006.

[35] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, and Ming-Hsuan Yang. Sky is not the limit: semantic-aware sky replacement. *ACM Trans. Graph.*, 35(4):149–1, 2016.

[36] Carsten Wenzel. Real-time atmospheric effects in games. In *ACM SIGGRAPH 2006 Courses*, pages 113–128. ACM, 2006.

[37] Li Xu, Feihu Qi, and Renjie Jiang. Shadow removal from a single image. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 2, pages 1049–1054. IEEE, 2006.

[38] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.

[39] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. CVPR*, 2017.