INF584 - StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings

Wenqing QU

March 24, 2023

1 Motivation

In the following, I will summarize the paper "StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings" and implement parts of its research.

2 Introduction

The paper introduces a new approach to example-based stylization of 3D renderings that takes into account illumination effects. The paper propose a two-stage process that first extracts illumination information from 3D rendering files and then applies this information to synthesize a stylized image using an example-based approach.

The paper use a set of style exemplars provided by trained artists to guide the synthesis process. These exemplars are used to generate a set of texture patches that capture the visual characteristics of each style. The texture patches are then combined with the illumination information extracted in the first stage to synthesize a stylized image that better preserves the visual richness of hand-created style exemplars.

Following are the implement details for the two process.

1.Light Propagation Calculation: The first component involves the calculation of light propagation in the 3D scene. This is achieved by rendering the scene using a global illumination algorithm, which captures the illumination effects such as soft shadows, color bleeding, and glossy reflections. The resulting rendered image is then printed on

paper with low contrast and alignment marks, and the artist paints on this paper to create the exemplar image.

2.Example-based Algorithm: The second component involves the use of an example-based algorithm to synthesize the target image while preserving the stylization of individual illumination effects in the source exemplar. The algorithm uses the light propagation information calculated in the first step as guidance to synthesize the target image. Specifically, the algorithm employs Light Path Expressions (LPEs) to capture the contribution of individual illumination effects in the exemplar and uses these LPEs to guide the synthesis of the target image. The algorithm also uses color and normal information to ensure that the shading and highlights are placed correctly in the target image.

3 Previous work

In the following, I will briefly describe the previously proposed theory and its corresponding shortcomings. These theories will be discussed in categories according to the guidance channels, which are normals, colors and light path expressions (LPEs).

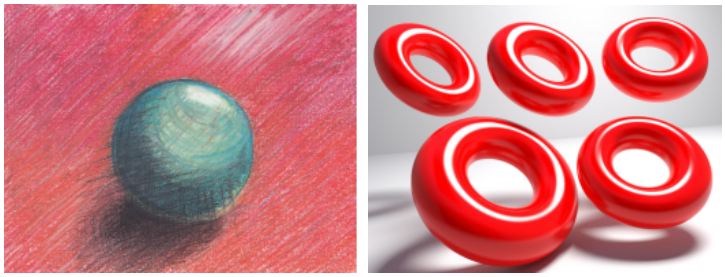

Figure 1: a source exemple and a rendering of a target scene

3.1 Normals

The Lit Sphere[SMGG01] uses normals to guide the rendering.

The lit sphere model involves rendering a sphere that is evenly lit from all directions. The sphere is then painted with the same material as the object to be shaded, and the resulting image is used as a reference for shading the object.

To use the lit sphere model, the artist or designer first creates a texture map that is based on the lit sphere image. This texture map is then applied to the object being rendered, with the shading of each point on the object determined by the corresponding point on the texture map.

However, using normals causing flaws in the stylization and position of the highlights in the rendered scene.

Because it just uses texture mapping to transfer the appearance in a pixel-wise fashion, there is no patch-wise consistency of the rendered output. The result is that the texture details of the exemplar are not reproduced correctly.

Figure 2: Normals

3.2 Colors

Image analogies[HJO+01] and extension[BCK+13] use color for guidance.

The Image Analogies algorithm works by establishing a mapping between two images, which allows for the transfer of style or content from one image to another. The algorithm follows three steps:

1.Analysis: In the first step, the algorithm analyzes the reference image and identifies its visual features, such as color and texture patterns. It then establishes correspondences between these features and those of the target images.

2.Synthesis: In the second step, the algorithm synthesizes a stylized version of the target images by applying the style of the reference image to them. This is done by mapping the visual features identified in the analysis step to those of the target images, and applying the appropriate transformations.

3.Transfer: In the final step, the synthesized image is refined and adjusted to ensure that it accurately represents the intended style or content transfer. This step involves

fine-tuning the parameters of the transformations used in the synthesis step.

However, these techniques fail to retrieve patches from appropriate locations in the exemplar, most visibly by mistakenly using patches from the background for the highlights. The greedy nature of the algorithm leads to texture details being corrupted by artificial seams that break the fidelity of the synthesized image.

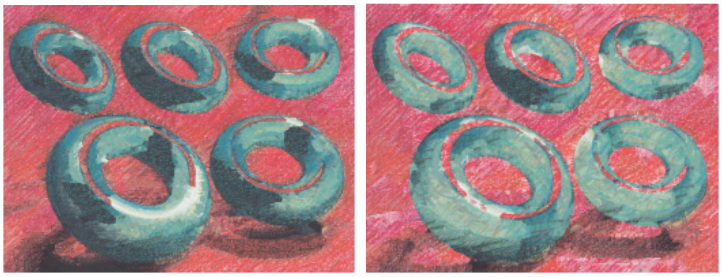

Figure 3: Colors, correspond to [HJO+01] and [BCK+13]

3.3 LPEs

The image synthesis methods described in the following use LPE as a means of guiding the synthesis process. Here is the basic idea of how it works:

First compute LPE channels for both the source and target images. These channels encode information about the lighting and shading of each image, and they can be used to compare the two images and guide the synthesis process.

Then during synthesis, the methods use the LPE channels in combination with color information to guide the placement of image patches. The LPE channels help to preserve the lighting and shading effects of the target scene by ensuring that the resulting synthesis is consistent with the light paths in the target image.

Following are several image synthesis methods and their respective drawbacks:

1.Texture optimization[WSI07]: The use of low-frequency content can result in a "wash-out" effect, where the resulting synthesis appears overused and lacking in detail.

2.Bidirectional similarity[SCSI08]: This method only provides a slight improvement over texture optimization and still suffers from the "wash-out" effect.

3.Image Melding[DSB+12]: While this method allows for changes in color, rotation, and scaling of patches, it still suffers from the "wash-out" effect, and the use of image

gradient channels in addition to LPEs may not be sufficient to guide the synthesis effectively.

4.Uniform patch usage[JFA+15]: Enforcing uniform patch usage can result in good texture quality but can completely break the overall structure of the target scene.

5.parameter λ method[KNL+15]: While this method provides more flexibility by providing a parameter λ to control the strength of the uniformity enforcement, manually tuning λ per scene is necessary to produce compelling results.

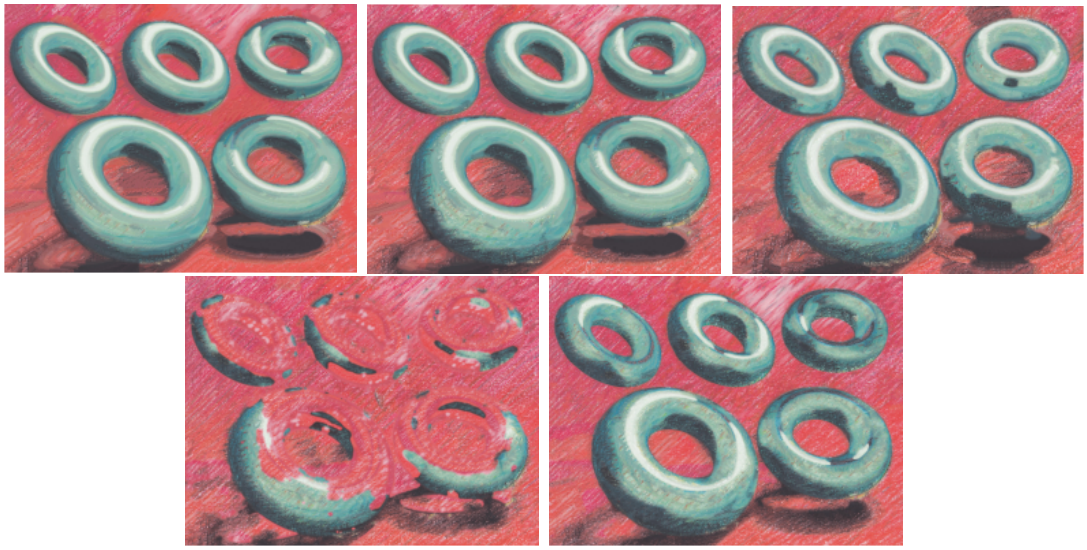

Figure 4: LPEs, correspond to [WSI07],[SCSI08],[DSB+12],[JFA+15] and[KNL+15]

4 Implementation

For all the implementations below, I have chosen to use the model spheres and rabbits that come with the OpenCV library. And the implements are mainly based on Image analogie from [HJO+01]

Figure 5: Light Path Expression images for Sphere

Figure 6: Light Path Expression images for Rabbit

The input are 3 images: an exemplar scene A rendered with LPE channels; a stylized exemplar A' aligned to the exemplar scene and a target scene B rendered with LPE channels.

The output is a new synthesized image B' such that the style from A is transferred to B according to the similarity between A and B.

[HJO+01] gives the basic idea: for each pixel p in B',try to find a match p' in A' such that p' could minimize

$$E(A, B, p, p', \mu) = ||A'(p') - B'(p)||^2 + \mu||A(p') - B(p)||^2$$

An optimization is proposed by [FLJ+14], showing that minimizing

$$\sum_{p \in B} min_{p' \in A} E(A, B, p, p', \mu)$$

gives better results.

This could be done by using EM-like iteration executed multiple times from coarse to fine resolution:

input : multi-channel images $\mathbf{A} = \{A, A'\}$ and $\mathbf{B}_k = \{B, B'_k\}$
output: synthesized target image B'_{k+1}

for *each pixel* $q \in \boldsymbol{B}_k$ **do**
  $NNF(q) = \underset{p \in \mathbf{A}}{\operatorname{argmin}} E(\mathbf{A}, \mathbf{B}_k, p, q, \mu)$

for *each pixel* $q \in \boldsymbol{B}_k$ **do**
  $B'_{k+1}(q) = Average(\mathbf{A}, NNF, q)$

Algorithm 1: EM-like iteration used to minimize energy (3).

Where B'_k denotes the current pixel values stored in B' and B'_k+1 the updated values, NNF is the nearest neighbour field that assigns source patch to each target patch and function Average computes the average color of colocated pixels in neighbour patches.

5 Result

My final result is fully based on NNF and run by GPU only, using 5 channels, which are 1. full global illumination render, 2.direct diffuse (LDE), 3. direct specular (LSE), 4. first two diffuse bounces(LD12E), and 5. diffuse interreflection (L,DDE). The stylization algorithm is run on 6 pyramid levels, with 8 light path tracing. I fix the size for the image as 860*650, the result can be run in 3 min.

6 Further Research

In computing the NNFs for a block of pixels centred on pixel p, I used a simple traversal algorithm. A more efficient method "PatchMatch algorithm" is mentioned in the paper[BSFG09], which can be the subject of more in-depth research later.

References

[BCK+13] Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. Stylizing animation by example. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[DSB+12] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012.

[FLJ+14] Jakub Fišer, Michal Lukáč, Ondřej Jamriška, Martin Čadík, Yotam Gingold, Paul Asente, and Daniel Sỳkora. Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. In *Computer Graphics Forum*, volume 33, pages 1–10. Wiley Online Library, 2014.

[HJO+01] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[JFA+15] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. Lazyfluids: Appearance transfer for fluid animations. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

[KNL+15] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[SCSI08] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[SMGG01] Peter-Pike Sloan, William Martin, Amy Gooch, and Bruce Gooch. The Lit Sphere: A Model for Capturing NPR Shading from Art. pages 143–150, June 2001.

[WSI07] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007.