Re-act: Compositing Yourself into Your Favorite Movies

Jay Shenoy
University of California, Berkeley
jayshenoy@berkeley.edu

Abstract

Digital compositing is a laborious process. Replacing one actor with another often involves re-shooting the performance under similar lighting conditions and manually performing color correction in post-production. In this project, I built a system to automatically relight and color-correct a person's face to match the lighting conditions in a target image, implementing a geometry-aware relighting scheme proposed by Shu et al. [9]. The algorithm can relight a photo to match another photo, painting, or even a video sequence with realistic results.

1. Introduction and Related Work

The task of replacing an actor in a video sequence with another recorded actor can be boiled down to histogram matching, a classic technique in image processing. The issue with simply matching histograms is that doing so fails to capture details in the facial lighting, which are dependent on the orientation of the face and the position of the light sources.

In addition to hue correction, it's necessary to properly relight the actor's face so that the composited image looks more realistic. There has been much research in the computer vision community in portrait relighting. A lot of work has been done in creating methods to relight human faces using high-definition environment maps and even simple light sources. Most of these methods have been data-driven, involving large datasets of human faces captured in complex light stage setups, as found in [10] and [11].

These methods are meant to be highly general, allowing one to relight portraits under any environment map. However, the problem of matching the lighting in an input portrait to that of a target portrait does not require such generality: it only requires that the two portraits have illumination conditions that appear similar. Thus, the method proposed by Shu et al. in [9], which solves the portrait relighting problem through an extension of the classic histogram matching technique, will suffice for the purpose of digital actor insertion.

(a) Input Image

(b) Target Image

(c) Relit Composite

Figure 1: Sample output of relighting algorithm

2. Overview of Technique

Shu et al. formulate portrait relighting as a mass transport problem that first represents each image as an eight-dimensional histogram: each pixel has a three-dimensional color, a three-dimensional normal vector, and a two-dimensional xy coordinate. In traditional histogram matching, each channel of the image is represented as a one-dimensional histogram, which is transferred to a target histogram by matching up the appropriate percentiles. Intuitively, if each one-dimensional histogram is visualized as a pile of sand, the act of histogram matching is analogous to finding the minimum amount of energy needed to move one pile of sand to match up with another.

The concept of moving piles of sand can be generalized to higher dimensions, and is formally referred to as the Wasserstein distance in probability theory. Concretely, to transfer the pixels of one image to match a target histogram, we wish to solve the following optimization problem that incorporates information about the positions and normal vectors:

$\arg\min_{\hat{f}} \sum_i \sum_j (||c_{ij} - \hat{f}_c(c_{ij})||^2 + ||p_{ij} - \hat{f}_p(p_{ij})||^2 + ||n_{ij} - \hat{f}_n(n_{ij})||^2) P(c_{ij}, p_{ij}, n_{ij})$

Here, c_{ij} represents the color of pixel (i, j), p_{ij} represents its position, and n_{ij} represents its normal vector. $\hat{f}_c$ represents the transfer function for color, $\hat{f}_p$ represents the transfer function for position, and $\hat{f}_n$ represents the transfer function for the normal vectors. Essentially, we are trying to minimize the distance that each eight-dimensional vector of each pixel is transported, subject to the constraint that the three $\hat{f}$ functions map the input image's histogram to the target image's histogram nearly perfectly.

Unlike in one-dimensional space, the eight-dimensional mass transport problem is computationally quite difficult to solve exactly. In the next section, I will discuss an approximation to computing the Wasserstein distance that the authors of [9] employ to perform portrait relighting.

3. Implementation

The relighting pipeline consists of several stages. 3D morphable models are first fit to the input and target images, producing per-pixel normals and positions that are extrapolated using a Poisson system. Stochasting sampling is used to compute multiple samples for each pixel and reduce noise in the final image. Next, I compute the mass transport between the eight-dimensional histograms with the Sliced Wasserstein Distance algorithm. The samples for each pixel are then averaged, and the entire algorithm is captured in a multiresolution pipeline that exploits the fact that lighting details tend to be of low-frequency. The details of this implementation are explained in the following subsections, and an illustration of the process is provided in 2.

3.1. 3D Face Fitting

Similar to project 4, I used a deep neural network to fit a 3D morphable model of a human face to both the input and target images. The deep network is built on top of the MobileNet architecture, and a pretrained implementation was kindly provided by the authors of the 3DDFA_V2 library [4]. The normal vectors of the fitted model had to be rasterized into image space. Note that the authors of [9] used a regression-based face-fitting technique, which is different from the one I used. The benefit of 3DDFA_V2 is that it is faster and more accurate than the regression approach; anecdotally, I noticed that the regression-based implementation provided in [1] took several minutes to fit each face and was highly inaccurate in low-light settings, whereas 3DDFA_V2 typically took less than 20 milliseconds to fit each face and worked well even parts of the face were out of frame.

3.2. Poisson Extrapolation

The normal vectors and position maps extracted from the previous step only contain geometric information for the image pixels corresponding to the facial region. To transfer the entire image, we need normal vectors and position maps for all the image pixels. While it is indeed possible to precisely estimate this information using other learning-based approaches in computer vision, for the relighting task it is sufficient to smoothly extrapolate the positions and normals in the masked face region to the rest of the image.

To do so, I set up a 2D discrete Poisson system in which the normals and positions in the facial region are fixed, and the gradients in the non-facial region are set to zero. Essentially, solving such a Poisson system smoothly extrapolates the x, y, and z coordinates of the normals and positions using a gradient field that encourages the values to gradually fade out. Since the Poisson matrix is inherently sparse, I used an algebraic multigrid solver [8] to compute the extrapolated maps.

3.3. Sliced Wasserstein Distance Algorithm

For each image pixel, I construct an eight-dimensional sample vector consisting of the three-dimensional color, three dimensional normal, and two-dimensional xy position coordinates. The authors of [9] note that incorporating the z position coordinate does not substantially alter the results.

As noted in the Overview section, computing the Wasserstein distance exactly in eight-dimensional space is computationally difficult. Therefore it is necessary to approximate this mass transport calculation using what's known as the Sliced Wasserstein Distance algorithm, which essentially computes random projections of the 8D samples onto one-dimensional axes and then performs the naive 1D histogram matching technique in an incremental fashion.

The algorithm operates as follows:

Algorithm 1: Mass Transport

Input: S_I, S_T, α, n
repeat n **times**
  Compute random orthonormal matrix P;
  $\tilde{S_I} \leftarrow S_I$ transformed by P;
  $\tilde{S_T} \leftarrow S_T$ transformed by P;
  for $i \leftarrow 1$ **to** 8 **do**
    $h_i(x) \leftarrow$ histogram matching function from $\tilde{S_I}$ to $\tilde{S_T}$ along i-th axis;
    $\tilde{S_I} \leftarrow \alpha h_i(\tilde{S_I}) + (1-\alpha)\tilde{S_I}$;
  end
  $S_I \leftarrow \tilde{S_I}$ transformed by P^T;
end

In the algorithm above, S_I and S_T correspond to the list of 8D samples of the input and target images, respectively. α is a hyperparameter akin to the learning rate of gradient descent in machine learning that tunes the speed at which mass transport is performed. n is another hyperparameter controlling the number of iterations for which mass transport is conducted. Empirically, setting $\alpha = 0.8$ and $n = 40$ generates realistic outputs.

3.4. Color Noise

The mass transport algorithm fundamentally creates a mapping between the histograms of the input and target images. Since this mapping transports one distinct sample per pixel in the input image, a nasty side effect is that samples that are close together in 8D space can be mapped to disparate regions. Practically, this means that it is possible for adjacent pixels in the input with similar color and normals can be transformed to noticeably different colors due to the inherent nature of discrete histogram matching. This produces the sorts of artifacts shown in 3.

To subvert this issue, I regularized the problem by constructing four additional 8D samples per pixel, where the color of each sample is set to the color of the original sample plus a Gaussian random variable with variance 0.1. Intuitively, this spreads out the samples in 8D space, allowing mass transport to map continuous regions. After transforming the input image, the samples for each pixel are averaged to produce the final output.

3.5. Multiresolution Approach

When the number of iterations is fixed, the time complexity of mass transport is $O(s \log s)$, where s is the number of samples to be transformed. As the input and target image sizes increase, mass transport slows down considerably. Thus, in order to speed up the process for high-resolution photographs, the authors of [9] suggest a two-scale approach that performs mass transport on a downsampled version of the input image, upscales the transformed input, applies a non-uniform high-pass filter on the original input, and finally adds these high-frequency details back to the transformed input.

The key insight of this approach is that face illumination is largely low-frequency, so it is reasonable to compute mass transport at a coarse scale and add back the high-frequency information later. To retrieve the high-frequency details, I implemented an edge-aware smoothing filter [5], which preserves sharp edges while smoothing out low-frequency regions, but runs much faster than the similar bilateral filter. This filter produces a low-frequency output, which I subtracted from the original image to retrieve the high-frequency information. The difference between the edge-aware smoothing filter and a simple Gaussian filter is shown in 4.

3.6. Video Inpainting

To remove an actor from a video sequence, I masked out their frame regions by hand and manually filled in the masked regions using similar parts of the video. The background plates were then composited with the input actor's green screen performance using standard chroma key.

I also tried automating the inpainting process using flow-edge guided video completion [3] to fill in the masked regions with temporally-consistent backgrounds. However, this technique did not work well for some scenes shot in low-light environments, so I stuck with the manual approach.

4. Results

The portrait relighting technique produces facial illumination that closely matches that of the target image. These results vastly outperform naive histogram matching, as shown in 5. Appendix A contains several examples of the technique in action, including light transfer from photographs to paintings, and the project website contains even more examples of the relighting algorithm, including composited video sequences.

Some of the images contain slight artifacts in the background, but these are to be expected due to way in which normals and position maps are extrapolated when solving the Poisson system. Since the goal of this project is to composite video sequences with realistic lighting, these background distortions do not matter as they are later masked out.

The generated video sequences look realistic when composited with the original movie scenes, but there is slight flicker in the illumination from frame to frame because the relighting step is computed for each frame separately. This issue is manually fixed using the technique described in [7], but one could potentially automate this process by employing temporal consistency methods as described in [2].

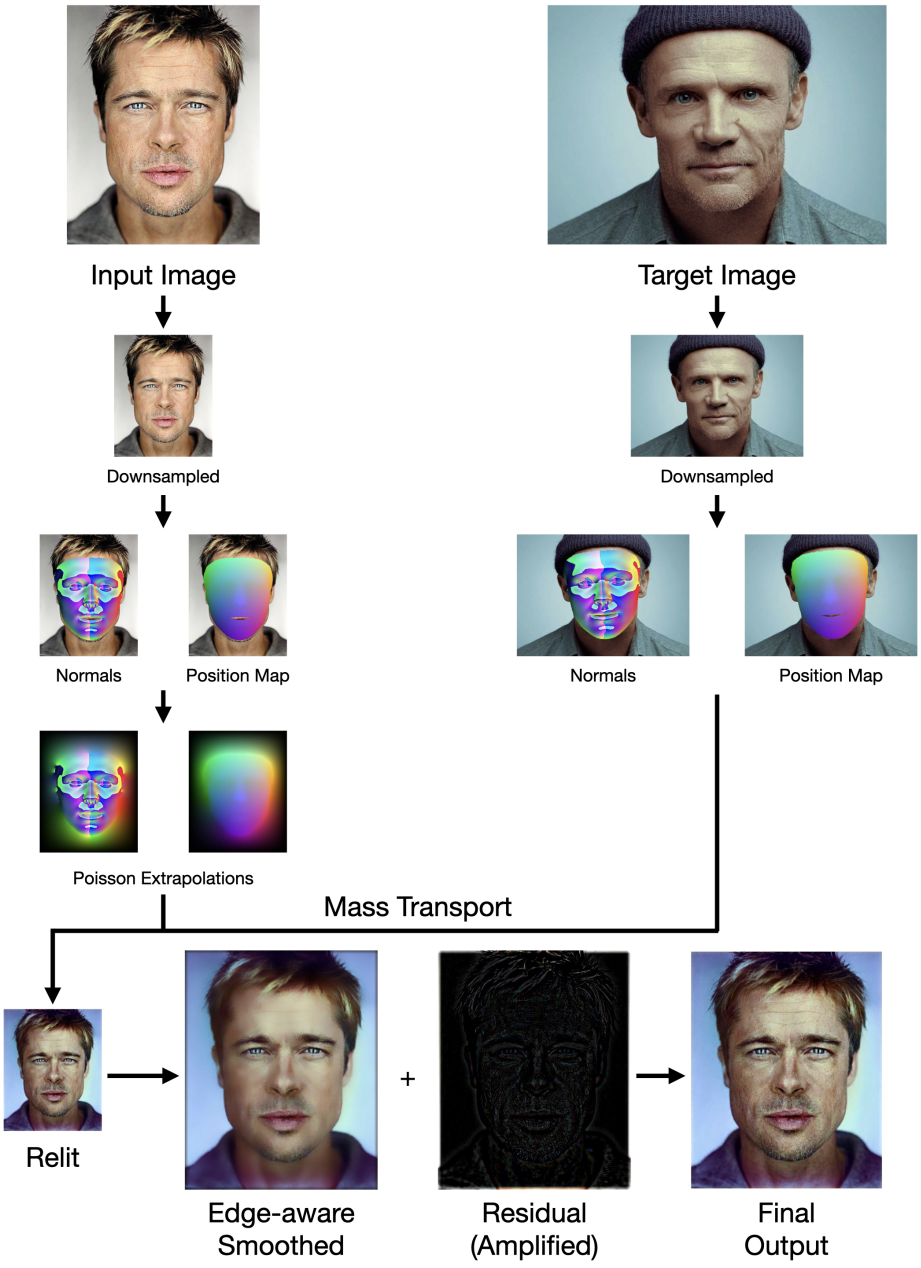


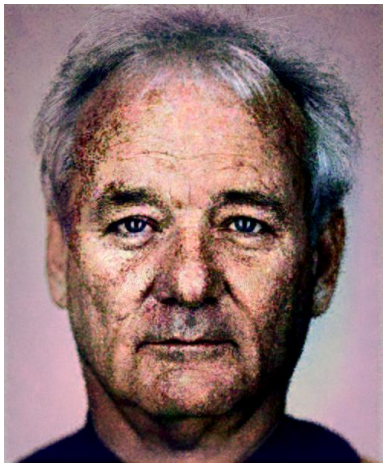
Figure 2: Pipeline diagram

5. Future Work

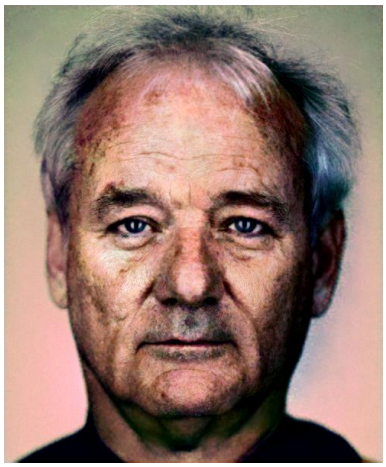
This project could be extended into a fully automatic digital compositing system that could use instance segmentation to select and replace actors without even using a green screen. It would be interesting to incorporate full-body mesh estimators as described in [6], which would provide better geometric information about the actors. In addition, the relighting technique implemented in this project only works when the actor is in the foreground, so allowing occlusion effects is another avenue worth exploring.

References

[1] Anil Bas, William A. P. Smith, Timo Bolkart, and Stefanie Wuhrer. Fitting a 3d morphable model to edges: A compari-



(a) Without stochastic sampling



(b) With stochastic sampling

Figure 3: Effect of stochastic sampling on noise reduction

son between hard and soft correspondences. 2016. 2

[2] Nicolas Bonneel, James Tompkin, Kalyan Sunkavalli, Deqing Sun Sylvain Paris, and Hanspeter Pfister. Blind video temporal consistency. 2015. 3

[3] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. Flow-edge guided video completion. 2020. 3

[4] Jianzhu Guo, Xiangyu Zhu, Yang Yang, Fan Yang, Zhen Lei, and Stan Z Li. Towards fast, accurate and stable 3d dense face alignment. 2020. 2

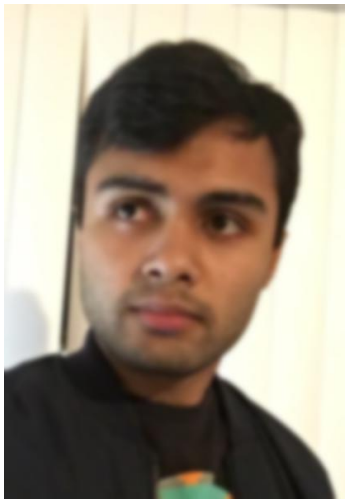
[5] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. 2013. 3

[6] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. 2018. 4

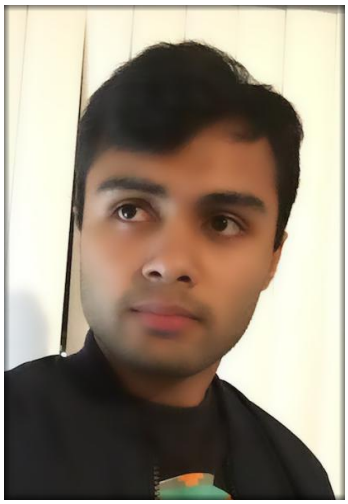
[7] Peter McKinnon. How to remove flicker from your videos!, 2018. 3

[8] L.N. Olson and J.B. Schroder. Pyamg: Algebraic multigrid solvers in python v4.0, 2018. 2

(a) Input Image

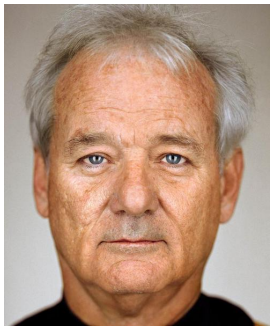


(b) Gaussian smoothing



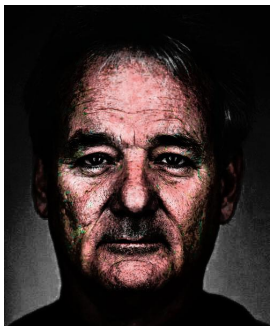
(c) Edge-aware smoothing

Figure 4: Difference between Gaussian and edge-aware smoothing

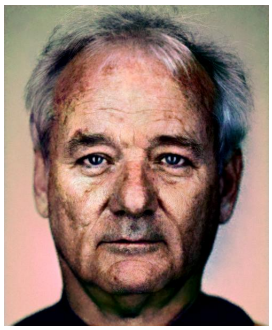


(a) Input Image

(b) Target Image



(c) Histogram Matching



(d) Mass Transport Relighting

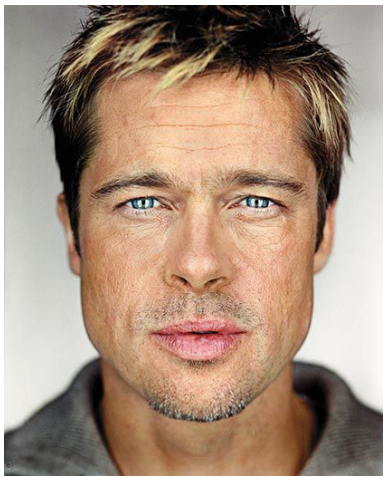
Figure 5: Histogram matching vs. mass transport

[9] Zhixin Shu, Sunil Hadap, Eli Shechtman, Kalyan Sunkavalli, Sylvain Paris, and Dimitris Samaras. Portrait lighting transfer using a mass transport approach. 2017. 1, 2, 3

[10] Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. 2019. 1

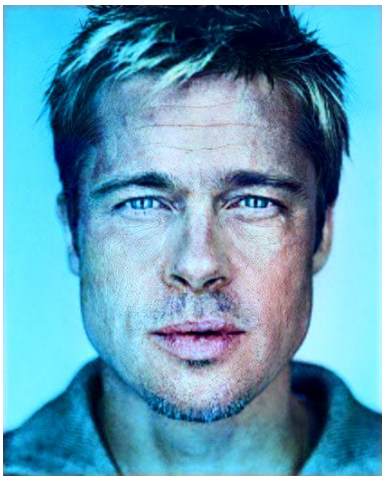
[11] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. Deep single image portrait relighting. 2019. 1

A. Relit Outputs

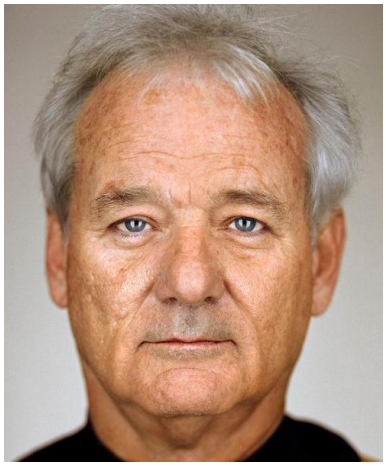


(a) Input Image

(b) Target Image

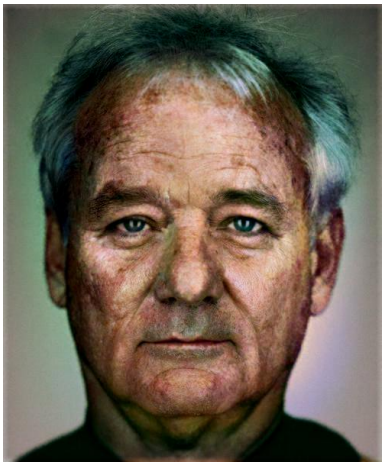


(c) Relit

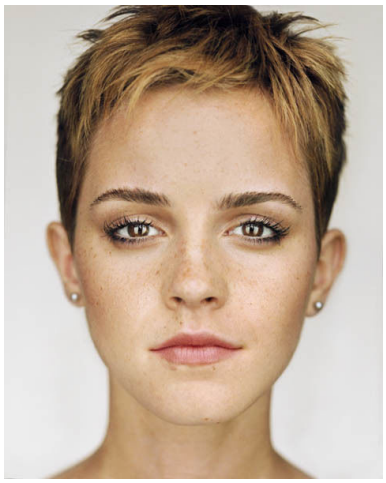


(a) Input Image

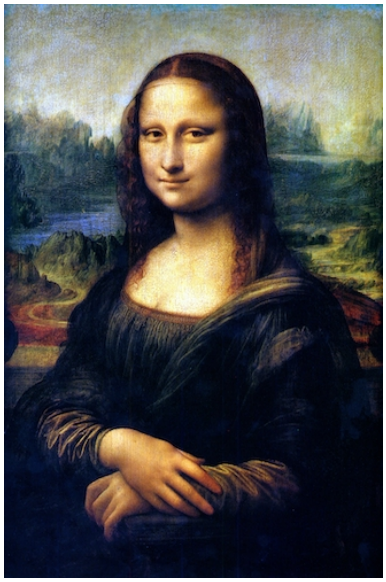
(b) Target Image



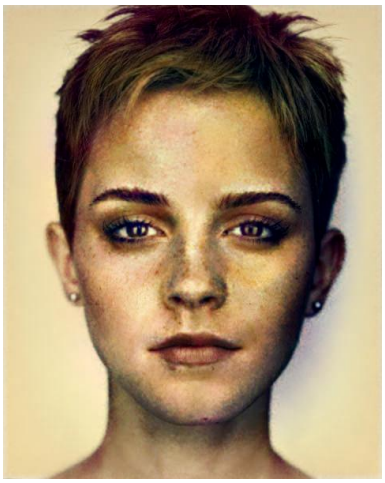
(c) Relit



(a) Input Image

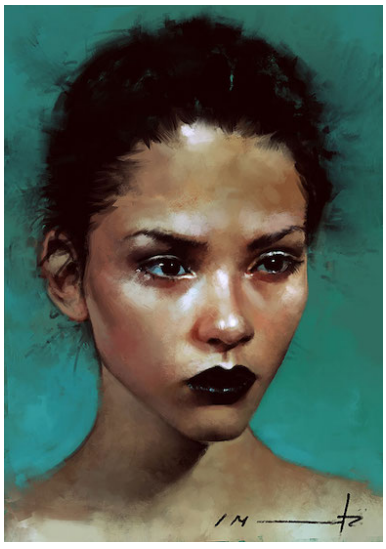


(b) Target Image



(c) Relit

(a) Input Image



(b) Target Image

(c) Relit