Stabilizing Object Lighting Under Variable Lighting Conditions In Video

Oscar Dadfar
Carnegie Mellon University
Pittsburgh, Pennsylvania
odadfar@andrew.cmu.edu

Figure 1. Our proposed algorithm takes a variably-unstable lighting sequence **[Top]** and relights the foreground and background scenes **[Bottom]** based on relighting weights. We attempt to approximate the illumination of discrete regions in the images based on hue and relight regions given their deviation from the global average illumination. For consistently-delectable objects in scene, we employ local relighting that samples a discrete uniform set of hsl values from the object's detected bounding box and relights each box based on the average hsl values.

Abstract

We present a novel light-stabilization algorithm that takes into account object BRDFs to relight sequences of frames subjected to variable lighting conditions. By assuming no objects in scene share the same BRDF, we calculate illumination changes in objects independently and apply a calculated relighting mask to a sequence of video frames to stabilize changes in illumination. OpenCV is used to calculate object bounding boxes to perform local lighting re-weightning. Pixels outside local objects are considered a part of the global lighting scale and are relight using k-means clustering to seperate hues in HSL space and perform relighting independently on separate clusters. Global relighting does not respect BRDF properties and assumes all objects in the same hue group share a uniform BRDF. We test our algorithm on manually-relit and real-life variable lighting scenes and discover that our algorithm is translation and scale-invariant but not rotation-invariant.

1 Introduction

Many sources of error can harm video capturing and cause abrupt, unwanted changes in scene. One such source is variable lighting which can lead to objects changing luminosity mid-scene. Environmental factors such as sunlight are elements that cause these changes that are out of reach of videographers. Often when filming indoors near windows, videographers have to take precautions and reduce their dependence on variable lighting by using other, more brighter ambient lights in the scene that they have control over such that abrupt changes in outdoor lighting will have a minimal effect on the illumination of objects in scene. Some artifacts of outdoor lighting are a lack of light during night shoots, overexposure in certain areas (also known as "hot spots"), harsh shadows from sunlight hitting only parts of a subject, and lens flare [3].

Using controlled lighting indoors is not always a feasible alternative. It often negatively impacts the aesthetics of a photo and only works at short ranges, due to the falloff of light as a function of distance [1]. Due to these issues, natural lighting is required for certain scenes, and can often create more lifelike environments than with indoor lighting, thus making it difficult for videographers to lessen their dependence on outdoor lighting.

In such scenarios where variable lighting is heavily present, post-processing lighting-stabilization techniques are used to reduce abrupt changes in lighting during a given scene. Professional Video Editing softwares like Adobe After Effects and Final Cut Pro offer videographers with plug-ins that can directly edit lighting conditions. These techniques allow videographer to traverse videos frame-by frame, assume some average luminosity value of each frame, and then the overall average luminosity value among all frames. Videographers will manually edit the the luminosity of a given frame, adding keyframes when illumination falls below a certain threshold from the mean. Procedures such as these act as a global relighting technique, where the brightness of the

entire scene is raised or lowered equally. Global relighting make heavy assumptions on the objects in view when brightening such objects, such as every object being equally as far from the camera and having the same BDRF properties. Computation time for these algorithms are quick, yet objects in scene will not always be lit correctly. By assuming uniform distance from the camera and constant BRDFs, more specular surfaces may appear darker and diffuse surfaces may end up brighter, ultimately leading objects to be lit incorrectly when light-stabilization is used.

A visual representation of these artifacts can be seen in Figure 2 where two static scenes with varying illumination are brightened to match in illumination. By using the green color patch of the image as a control point, the darker image is brightened until the green color patches match in both images. This result still leaves the red patch to differ in both images because the difference in illumination between the green patches and red patches are not equal, nor do they respond to variable lighting the same way.

Objects in the real world have varying responses to variable lighting depending on their geometry and exterior material composition. Light-stabilization algorithms should attempt to preserve these conditions as much as possible. Doing so requires independently analyzing the change in luminosity of each object and making the necessary alterations of illumination for each object.

2 Contributions

In an effort to preserve the varying illumination properties of objects, we propose a light-stabilization algorithm that periodically analyzes an object's illumination and modifies object illumination independent of other objects in scene. Object detection gives us the corresponding bounding boxes we need to sample an object's luminosity in a given frame. Given the bounding box of an object for a set of discrete frames, we can uniformly sample values from the object in each frame and average these values with other frames to get an average luminosity of the object over the course of the video. When the object differs from its average luminosity by a certain threshold, the algorithm will combine part of the average luminosity of the object with its current luminosity to bring it closer to the average.

3 Limitations

Due to the algorithm's high reliance on object detection and segmentation, the results of the research depend on how well vision algorithms are able to detect objects on scene. Jittery and imprecise bounding boxes will cause the algorithm to sample points on the screen that may not be related to the object at all, thus biasing the object's average illumination.

Object detection does not give any information on the orientation of the object either. The way we sample illuminations from the object will stay the same even if the scene

Figure 2. [Top] Original photos under low **[Left]** and high **[Right]** outdoor lighting. **[Bottom]** Increased low lighting photo **[Left]** to match high lighting photo **[Right]**. Center compares color patches between top two and bottom two photos. Brightness is increased in lower photos until green shades match while a difference between red shades remain.

rotates, thus biasing illumination sampling. Making a more robust sampling algorithm requires extracting orientation features from the bounding box.

An approximation algorithm to extract orientation features could be to calculate the average area of the bounding boxes across all frames of an object. If the area of an object's bounding box in a given frame is noticeably larger than the average, then we periodically rotate the frame in both directions and rerun the frame in the object detection algorithm until the bounding box area approaches the average area. While such an algorithm would ensure the bounding boxes in each frame are roughly the same size, this does not work in scenes that zoom and change object size. Thus, we constrain our algorithm for orientation scenes so as not to worry about rotations.

Object detection only provides a rough estimate as to where the object may be located. Since bounding boxes are rectangular and very few objects in real life exhibit rectangular faces, our algorithm would end up sampling luminosity from the background and neighboring objects contained in the bounding box. To reduce extraneous luminosity samplings, an extension of our algorithm could be to use object segmentation that constrains our luminosity sampling and re-weighting to the object itself. By using a segmentation, our algorithm would rely even more on the segmentation detecting algorithm, since incorrect meshes will force our algorithm to sample more from the backgrounds and nearby objects while less from the object we may be interested in. Thus our algorithm would heavily rely on the accuracy of object segmentation.

4 Related Works

Many algorithms exist for color correction and light stabilization that use various hardware and algorithmic techniques

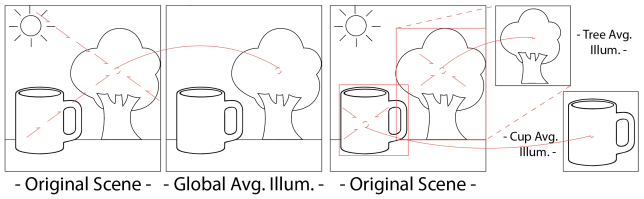

Figure 3. Global re-illumination **[Left]** attempts to relight one pixel (control point) to its corresponding pixel in the average illuminated case, and use the same relighting across the entire image. Local re-illumination **[Right]** uses multiple control points for each object. Relighting radius for each object is smaller and more precise in the later case.

to stabilize pixel quality. A meta review by Xu and Mulligan [11] tested 10 different state-of-the-art color correction algorithms on consecutive images from video. Each algorithm was evaluated based on two criteria: the similarity of color pallets and the structure similarity between the two images. The latter trait measured the degree of feature loss after color correcting. Algorithms with local color adjustments performed better overall than those with global shifts, signifying the improved performance quality of local relighting operations.

A common approach to the software side of color and lighting stabilization is to use neural networks to detect and transform colors. HaCohen et. al [12] used a SVM to target core color features of a set of images to make a graph of them. The SVM is user-definable and can be altered to be more or less sensitive to creating edges between nodes. The algorithm is a useful approach to visualizing systems of feature linkages in images and is user-controllable in how well the SVM classifies features and assigns weights.

Datasets can also be used to create a color mapping between images. By having a large quantity of variable lighting objects, it is easier to detect and transfer colors from their ground truth. Banic and Loncaric [2] plotted a histogram of color features for images in order to extract the most important color features. This leads to a substantial reduction in required variables, since knowing the red pixel can be used to calculate the blue and green pixels (colors are normalized to sum to 1). Simmilary Yang et. al [5] extracted illumination details from an image by looking at grey pixels in order to re-render the image under white lighting. For a known set of multiple lighting sources k, they then split the pixels into K groups using K-means clustering based on their positions and observe illumination properties (such as color and intensity) separately from these regions. Hussain and Akbari [4] also used color deconstruction to parse an image into a finite number of discrete lighting regions though K-means clustering of three pixel values, calculate the lighting at each point and using a weighted sum to get the overall lighting conditions of the image. Their approach also uses local relighting techniques, but creates pre-defined local regions based on neighboring color groups, not specific objects.

Color mappings can also be used to create a function that maps colors between different domains in order to create novel lighting conditions. Oskam et. al [9] sample color values between input and transfer images to create a color mapping that can recolor a transfer image. By placing colors in vector spaces, colors not identified in the original image can be mapped to other colors via a weighting function vectors pointing to known color regions.

Advancements in hardware have also emerged to provide color-consistence. Simao and Schneebeli [8] provided an iterative approach that applies a color matrix to pixels to recolor values. By using a sensor attaches to the camera, the sensor can grab a hold of colors from previous frames and determine a canonical color representative for each object (what the object would look like under stable lighting conditions) and transport those pixel values onto future frames. The transporting is done in iterations, where they attempt to minimize the difference between an object's pixels and its canonical color representation.

Much of color correction rely on being able to localize regions to perform color correction on. Some approaches define each object as a localized region, which requires sufficient object detection and segmentation algorithms. The research from Miksik et. al [7] can use information about previous frames and segmentations to make accurate and consistent predictions about current frame segmentations. By warping the current frame to match with the previous frame, they are able to get a warped segmentation of objects that, when warp-inverted, can produce segmentations for the current frame. This works well for large-scale objects with low motion and long screen time, but dips in accuracy for small objects with small pixel area and high motion.

By rendering video as a sequence of images through image-relighting algorithms, artifacts arise in the temporal domain between frames. Bonneel et. al [6] created an algorithm for preserving temporal consistency between frame-edited segments in video. By using gradient edge analysis, the researchers want to minimize the difference between the artifacts and post-processed video frames, where the operation is a weight of how much the original video frame differed from the mean of video frames. Lai et. al [10] use deep learning to minimize local and global temporal flickering to make the frames more consistent. Local temporal flickering attempts to warp frames to match the previous frame to provide smoothening between frames and its neighbors, while global temporal flickering compares every frame to the first, where the first frame acts as a stabilization point. This research is useful in reducing local and global deviations in color by comparing colors of frames to neighboring frames, as well as to the first frame of a video as a reference point.

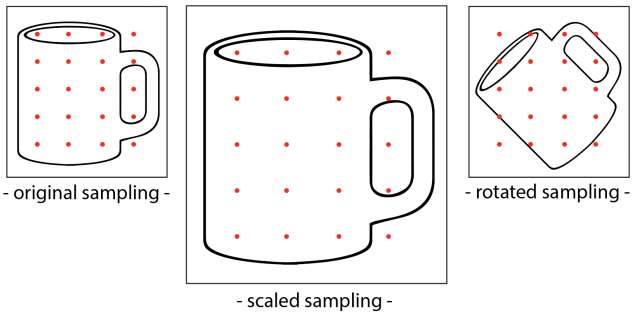

Figure 4. Scaled bounding boxes preserves pixel sampling. Rotation does not preserve sampling.

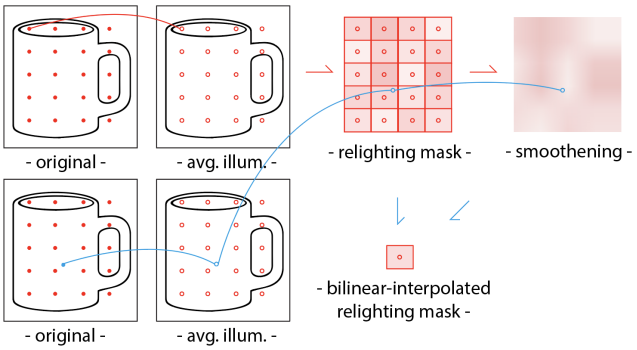

Figure 5. [Top] Iteratively capturing relighting weights for discrete sampling points across all frames. **[Bottom]** using bilinear interpolation to smoothen weights for points not originally sampled.

5 Methods

Our approach captures the average illumination for each identifiable object in scene as well as the overall illumination of all frames. This setup allows us to perform relighting on both the local and global scale by comparing each object with their average object illumination and the entire scene with the global average to relight pixels that deviate too much from their average illumination. This approach differs from standard relighting techniques by providing multiple illumination mappings rather than just a single global illumination as in Figure 3.

5.1 Pre-Weighting Local Illuminations

In order to have a heuristic for object relighting, we need an estimate for the average object illumination across the scene. We begin by using OpenCVs pre-trained YOLO v3 model to detect the classes and bounding boxes of objects among all frames. For each detected object, we record the object class and bounding box for average luminosity sampling. In this paper, we work in HLS space, where we attempt to discover hue, luminance, and saturation weights that we store as a relighting mask to relight our image.

The bounding boxes of any given object are not expected to remain constant, as scenes can zoom in and out, changing the scale of the object and the area of the frame it inhibits over time. Even if we were to keep the scene constant, an artifact of OpenCV is that it can never produce completely motionless bounding boxes, and as such, our program must account for changes in the dimensions of a bounding box for an object. To do so, we specify for each object the number of horizontal and vertical luminosity samples taken from each of its bounding boxes. We assume that such a model is robust for panning and scaling operations as shown in Figure 4 where we consistently target the same relative coordinates of an object invariant of translation or scaling. Rotation does not preserve relative sampling and thus is not supported by our model. As a result of our approach, we assume no orientation changes occur for any objects.

With each object, we specify a sampling rate across the horizontal and vertical fields that determines the resolution of the average-illuminated object. A sampling rate too small does not provide much illumination information, and a sampling rate too large can be too costly without providing much of a performance improvement. Upon detecting a new object, our algorithm assigns the sampling rate of the object to be the initial bounding box of the object scaled down by a factor of 10.

5.2 Local Relighting

We present pseudocode of our relighting algorithm in Algorithm 1. Given the average illumination of an object and all the instances where the object is present across the frames, we can individually compare the illumination of the cropped component of the object in each frame to the average illumination for that object. Since the bounding box $[W_i, H_i]$ of the object for each frame i is not guaranteed to match the dimensions of the average image-illuminated object $[W_o, H_o]$, we employ the same sampling technique of uniformly sampling $W_o \times H_o$ components in frame $i's$ bounding box and comparing them to their corresponding index in the average image-illuminated object (which we refer to as its corresponding pixel representative). Doing so will provide information on the illumination difference from the average illumination at each of the $W_o \times H_o$ discrete, spatially-uniform regions of frame $i's$ image.

To relight the image, we allow the user to specify a relighting strength coefficient that signifies how much the illuminations of each sampled pixel from each frame of an object is allowed to deviate from their corresponding pixel representative in the average-lit object. With each sampled object pixel, we can capture a variance between it and its pixel representative, and can relight the pixel closer to its representative as a function of the strength coefficient and variance. As in Figure 5, we assign to each sample pixel a

Algorithm 1 Local Relighting

```
imgs ← loadImgs(fileDir)
while n < numIters do
  for img in imgs do
    classes, bboxes ← openCV(imgs)
    for i in len(classes) do
      if illumObj.exists(classes[i]) then
        illum ← illumObj.find(classes[i])
      else
        illum ← illumObj.create(classes[i])
      end if
      illum.reweight(bboxes[i])
    end for
  end for
  for img in imgs do
    relight ← [ ]
    classes, bboxes ← openCV(imgs)
    for i in len(classes) do
      illum ← illumObj.find(classes[i])
      relight ← illum.relight(bboxes[i], relight)
      relight ← smoothen(relight)
      newImg ← relightImg(imgs[i], relight, strength)
      imgs[i] ← newImg
    end for
  end for
  n ← n + 1
end while
```

Algorithm 2 Global Relighting

```
imgs ← loadImgs(fileDir)
buckets ← kMeans(imgs[0].hues)
while n < numIters do
  for i in len(buckets) do
    globals[i] ← createGlobals(buckets[i])
  end for
  for img in imgs and g in globals do
    g.tryRelight(img)
  end for
  for i in len(imgs) do
    relight ← [ ]
    relight ← globalRelightWeight(imgs[i], relight)
    relight ← smoothen(relight)
    newImg ← relightImg(imgs[i], relight, strength)
    imgs[i] ← newImg
  end for
  n ← n + 1
end while
```

relighting weight that signifies how much of its luminance is to be changed in a given frame. For object pixels not sampled originally, we use bilinear interpolation over the pixel representatives to find the closest pixel value given its location and determine its relighting value. This provides a smoother relighting weight for objects where $[W_o, H_o] << [W_i, H_i]$.

5.3 Global Relighting

Not all objects can be discretely identified by openCV, so for portions of frames that were not given bounding boxes, we use global relighting similar to Hussain and Akbari to get their relighting weight [4]. We first want to make a heuristic for dividing the unclaimed pixels for each frame into N discrete color bins such that we minimize variance between the color values of any pixels in the bin. We sample the hue values from the first frame and use k-means clustering to sort the colors into N hue bins. For the remaining images, we add the pixels to the corresponding hue bins. Example pseudocode can be seen in Algorithm 2.

As new light is introduced into the scene, different color groups will respond differently to the additional lighting. For each bin, we calculate the average hue, illumination, and saturation of pixels retrieved from that frame and the overall average among all frames and use this as a heuristic for how much each color bin should be re-lit. Comparing a bin's average frame illumination to its overall average illumination gives us the same variance needed to calculate how much color re-weighting is needed to bring the average illumination of the pixels in that bin of a given frame closer to their bin average. This re-weighting can then be applied to all pixels in that bin for that particular frame.

Notice that this approach does not respect object BRDFs as it captures average illuminations across multiple objects in order to relight a scene. This is the best we can do given we use global relighting on areas where object locations and bounding boxes are not made clear.

6 Results

We tested our algorithm on various image sets with distinct foreground and background components, where we introduced known variable lighting to the images and attempted to correct for the lighting. In one trial we test an image sequence with stabilized positions while in another trial we apply positional and scale-based jitters to test the robustness of the algorithm to transformations on position and scale. We also test the effects of our algorithm on real-life video capture.

7 Discussion

Results in Figure 6 show the original and relighted image sequence. Our algorithm was able to detect and apply illumination masks to reduce illumination differences in images. Our algorithm is demonstrated to be position and scale invariant.

When initially constructing sampling dimensions for new local illumination object, we attempted to find optimal sampling dimensions proportional to $[w_{in}, h_{in}]$ which represent

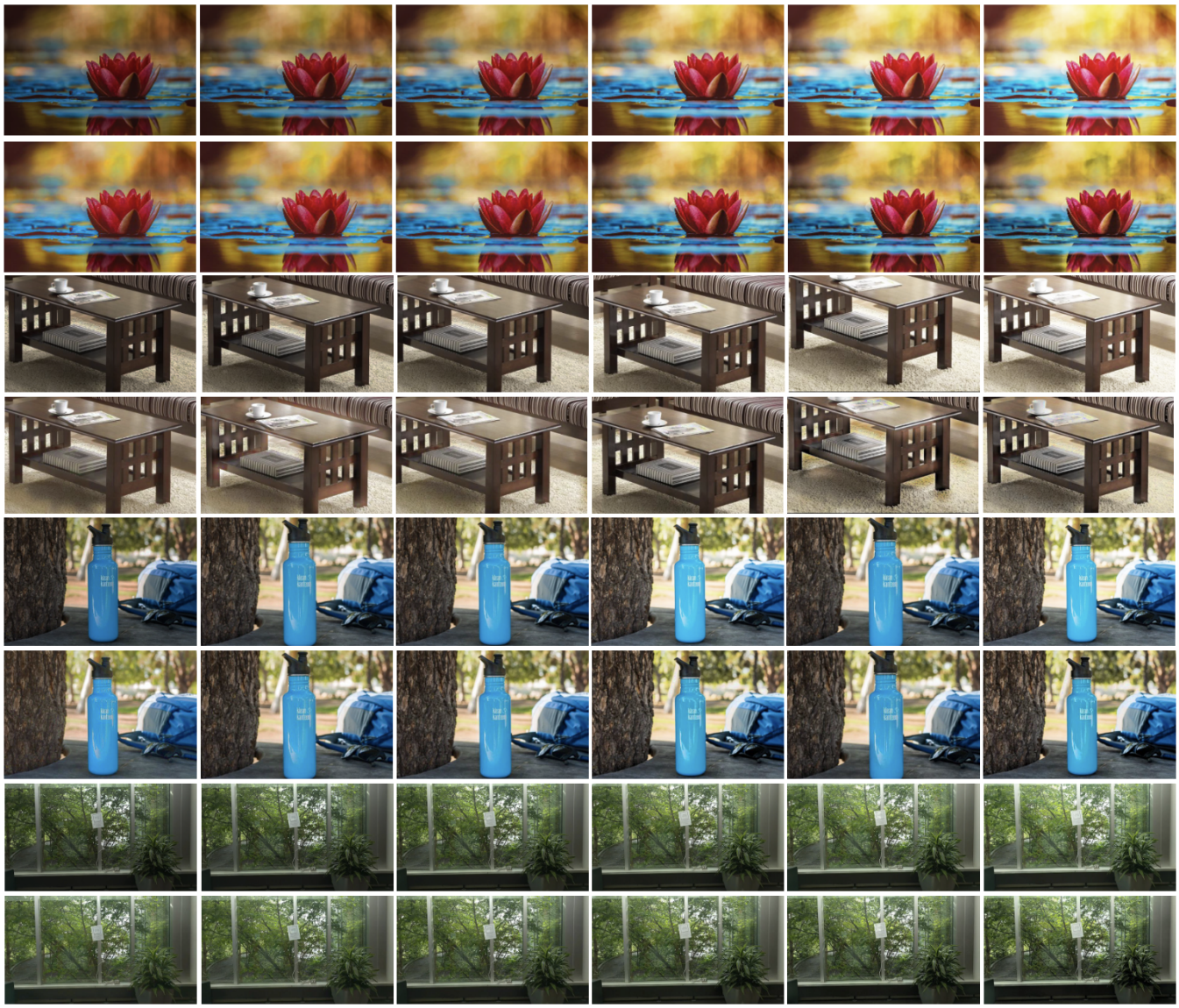

Figure 6. [Row 1.] Relighting on stable-transformation frames. **[Row 2.]** Relighting on position-variant frames. **[Row 3.]** Relighting on scale-variant frames. **[Row 4.]** Relighting on naturally-lit frames.

the initial width and height of the bounding box where the object was first detected. Through multiple trials, we found that sampling rates of $[w_{in}/div, h_{in}/div]$ for $div \in [1, 10]$ produced optimal results. For any value of div larger than 10, we would have too high pixels per sample resulting in very uneven results from too little sampling. By keeping div low, we ensure that we have enough samples to accurately capture the illumination of the object in each frame without missing any details. One case where this approach fails is when an object is first revealed when from far away before coming close to the viewer. In this case, we start with small sampling dimensions from its first appearance that overall leads to small illumination sampling as the object gets closer to the viewer. A possible resolve could be to adjust sampling dimensions over time if the bounding box of an object exceeds more than k times its sampling dimensions for some fixed k.

True constant illumination could not be achieved, as there is still some illumination difference present between the first and last frames of each sequence. This is because changes in illumination can also lead to changes in hues, and as such leads to differences in hue segmentations across images as visualized in Figure 7. That is, when a bright pixel of a certain hue class leaves to another hue class, the class it left receives a lower illumination and a higher illumination mask, while the class that now features the new pixel has a

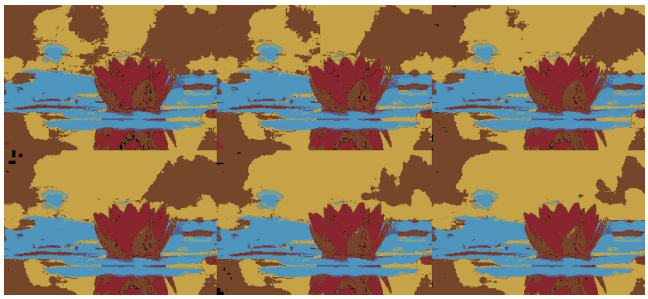

Figure 7. Hue Segmentation on 10 discrete hue buckets using k-means clustering for optimal partitioning. Segmentations do not remain constant across frames, as can be noted with the change in the surface area of brown.

higher illumination and a lower illumination mask. Solving such a problem would require adjusting hue ranges across images so that images with overall brighter illuminations share similar hue segmentations as those with lower overall illuminations.

Originally, a sharp relighting mask was used as seen in Figure 8. This led to really sharp edges and illumination artifacts, particularly in regions that should have been smoothed out. From this, we applied a 3x3 Gaussian blur matrix for 250 iterations on each of the 3 HSL relighting maps to reduce relighting artifacts that arose from sharp edges.

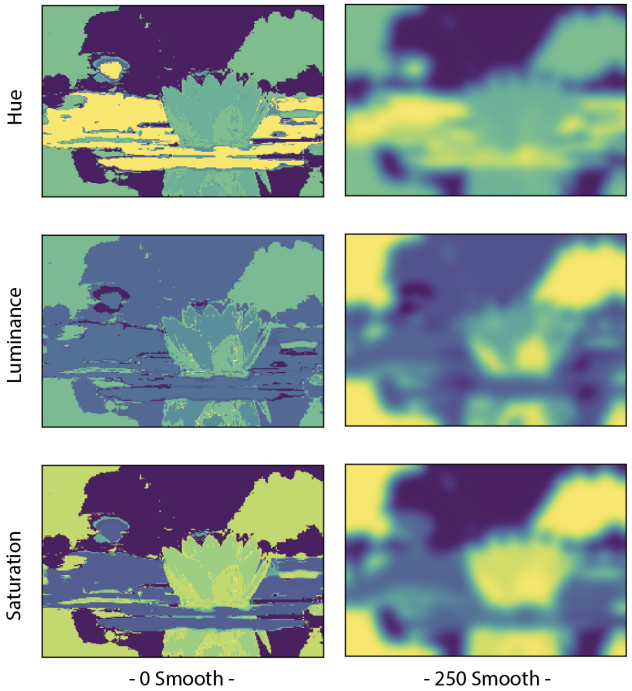

Figure 8. HSL relighting maps calculated on flower image set. Gaussian smoothing is used to resolve artifacts that arise from hard relighting lines.

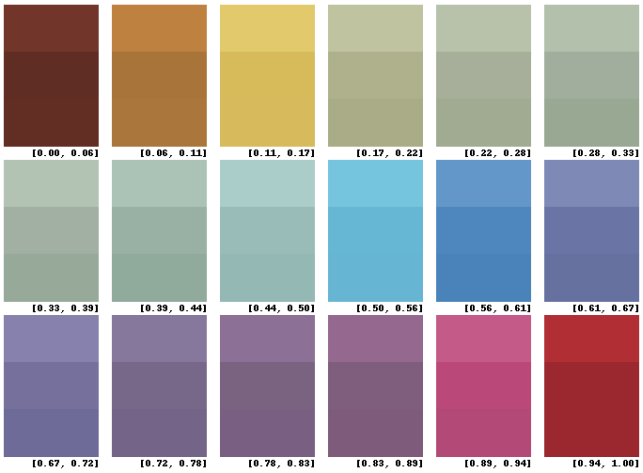

Figure 9. Global colors are broken down into 18 independent hue ranges. Average hue, illumination, and saturation are calculated for each color group. The top color in each patch represents the initial HSL colors of the first image in the flower set. The middle color represents the relighted image colors. The lower color is the global average of that color. Each color is labeled by its hue range. After relighting, each color group is lit closer towards its canonical representative.

We compare the HSL output between initial and re-lit frames in Figure 9. From this we can see that the hue divisions of each frame steadily reach the average hue values after each iteration. Since we use a relighting strength of a ones-vector in Algorithm 2, then our results converge on the first iteration, where subsequent iterations provide no changes to our HSL values.

8 Conclusion

We presented a novel relighting algorithm that individually relights foreground objects detectable by OpenCV in order to respect object BRDF properties. Not all objects can be re-lit independently, as our algorithm is limited by OpenCVs ability to detect objects in scene. With better object detection we can break our scene more consistently into discrete lighting regions that can more optimally relight the scene on an object-by-object basis.

References

[1] Shree Nayar Amit Agrawal, Ramesh Raskar and Yuanzhen Li. 2005. Removing Photography Artifacts using Gradient Projection and Flash-Exposure Sampling. *IEEE* (2005), 8.

[2] Nikola Banic and Sven Loncaric. 2015. Color Cat: Remembering Colors for Illumination Estimation. *IEEE SIGNAL PROCESSING LETTERS* 22, 6 (jun 2015).

[3] Amy Horton. 2017. Perfecting Outdoor Lighting.

[4] Akmol Hussain and Akbar Sheikh Akbari. 2018. Color Constancy Algorithm for Mixed-Illuminant Scene Images. *IEEE* 6 (mar 2018). https://doi.org/10.1109/ACCESS.2018.2808502

[5] Shao-Bing Gao Kai-Fu Yang and Yong-Jie Li. 2015. Efficient Illuminant Estimation for Color Constancy Using Grey Pixels. *CVPR* (2015), 2254 – 2263.

[6] James Tompkin Nicolas Bonneel and Kalyan Sunkavalli. 2015. Blind Video Temporal Consistency. (2015), 9.

[7] J. Andrew Bagnell Ondrej Miksik, Daniel Munoz and Martial Hebert. 2018. Efficient Temporal Consistency for Streaming Video Scene Analysis. (2018), 7.

[8] Josemar Simao and Hang Andreas Schneebeli. 2014. An Iterative Approach for Color Constancy. *SBR-LARS Robotics Symposium and Robocontrol* (2014), 130–135. https://doi.org/10.1109/SBR.LARS.Robocontrol.2014.34

[9] Robert W. Sumner Thomas Oskam, Alexander Hornung and Markus Gross. 2012. Fast and Stable Color Balancing for Images and Augmented Reality. *Second Joint 3DIM/3DPVT Conference* (2012), 49–56.

[10] Oliver Wang Wei-Sheng Lai, Jia-Bin Huang and Eli Shechtman. 2018. Learning Blind Video Temporal Consistency. *ECCV* (2018), 16.

[11] Wei Xu and Jane Mulligan. 2008. Performance Evaluation of Color Correction Approaches for Automatic Multi-view Image and Video Stitching. *IEEE* (2008), 8.

[12] Dan B Goldman Yoav HaCohen, Eli Shechtman and Dani Lischinski. 2018. Optimizing Color Consistency in Photo Collections. *IEEE* (2018), 9.