Introduction

In this lab assignment, we empirically study a subset of topics within low-level vision. In particular, in Section 1, we begin with photometric stereo where we try to estimate a 3D surface from multiple images of a static object captured with varying illumination and a fixed camera. Next, in Section 2, we explore conversions among various color spaces in which images can be represented. In Section 3, we look at the fundamental problem of intrinsic image decomposition. Finally, in Section 4, we look at some color constancy algorithms. We conclude with a summary and some learnings.

1 Photometric Stereo

1.1 Estimating Albedo and Surface Normal

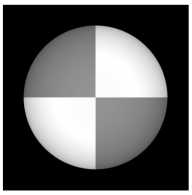
Answers to Question 1.

1. Albedo is essentially the reflectance of the surface. Through visual inspection, we can say that two quarters of the sphere are highly reflective (north-east and south-west) while the remaining two are less reflective (north-west and south-east). In the albedo obtained, as shown in Figure 1a, we see the result in-line with our expectation.

2. Typically, we need $n \geq 3$ images to reliably estimate albedo and normals because we are trying to estimate 3 variables (i.e. $\mathbf{g}(x, y) \in \mathbb{R}^3$) in a linear system of equations via least-squares fitting. With $n = 3$, if images are such that the point light source kept in the 3 settings is not co-planar, then $\mathcal{V} \in \mathbb{R}^{3\times3}$ has linearly independent columns and we could get a unique solution $\mathbf{g}(x, y) = \mathcal{V}^{-1}\mathbf{i}(x, y)$. We experiment with varying n and observe that the normals obtained for $n < 3$ are unreliable while for $n >= 6$, we tend to get better estimates of albedos and normals as n increases as shown in Figure 2. This is because with more images, we tend to average out the measurement noise.

3. Shadowed regions may be problematic for photometric stereo since $\mathbf{i}(x, y) = 0$ for pixels in such regions, thus, we may end up with the trivial solution $\mathbf{g}(x, y) = 0$. With shadow trick, we multiply both sides by $\mathcal{I} = \texttt{diag}(\mathbf{i})$. This zeroes out any equations from points in shadows as the corresponding elements in $\mathcal{I}$ are zero. Empirically, this holds for $n = 5$ where normals without the shadow trick have artefacts as shown in Figure 3. However, with $n = 25$, such artefacts are not seen (Figure 4) since we have a lot of observations and any given pixel is more likely to be lit in atleast some of them.

Notes on implementation: Results for this question have been generated using the notebook `lab1/photometric/notebooks/question-1.ipynb`. For numerical stability, while computing the normals $N(x, y)$, we add ϵ to the denominator to handle the case when the denominator (albedo) is 0.



(a) Albedo

(b) Normals: (L) N_x normal along x-direction, (C) N_y, (R) N_z

Figure 1: Obtained albedo and normals for gray sphere with $N = 5$ images.

1.2 Test of Integrability

Answers to Question 2: For the case $n = 5$, we observe that pixels at the edge-circumference of the hemisphere tend to have higher SE as shown in Figure 5. This may be because the image region is at the circumference is non-differentiable and yet we estimate derivates numerically. We define the number of outlier points as the cardinality of $\{(x, y) : SE(x, y) > \tau)\}$ where τ is a fixed threshold (here, $\tau = 0.005$). As n increases, for $n > 3$, we see that this quantity reduces since the measurement noise reduces with more observations. For $n < 3$, the normals themselves are unreliable, so we cannot conclude much about p, q, SE.

1.3 Shape by Integration

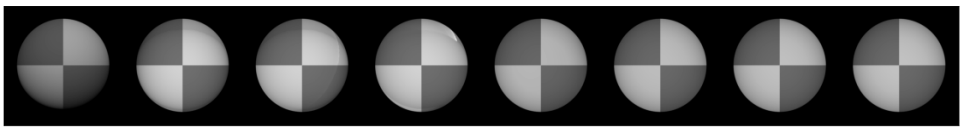
Answers to Question 3.

1. For $n = 5$, we consider the case of construction surface by integration via (a) column-path, (b) row-path and (c) average of the two. For (a), we observe a bump on the surface along the x axis at $y = c$ (constant) somewhere in the middle and the surface becomes non-differentiable (Figure 6a). This happens because there is a *sudden* change in the reflectivity (albedo) of the sphere as we go from $y \leq c$ to $y > c$. Similarly, we observe a bump along y-axis for row-path method (Figure 6b).

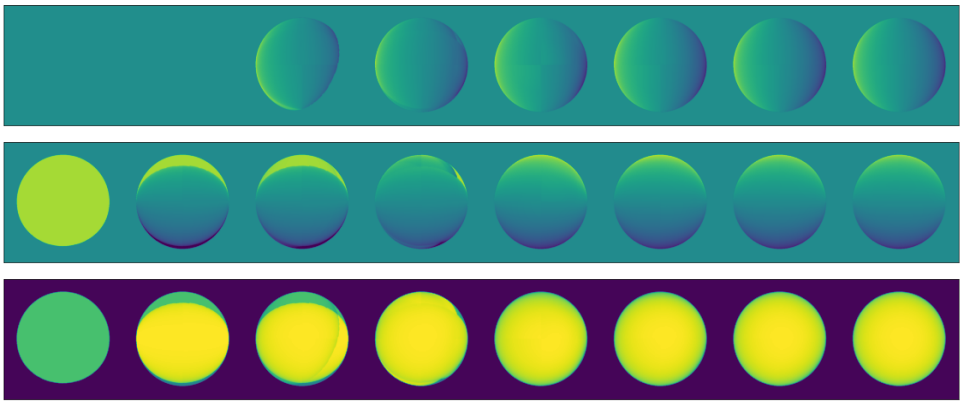
2. In (c), for $n = 5$, both bumps are visible but are somewhat reduced due to the averaging effect (Figure 6c). When we consider $n = 25$ with average-method, we get a nice smooth surface without any bumps since the measurement noise is reduced (Figure 6d).

1.4 Experiments with different objects

Answers to Question 4 (`MonkeyGray`): The results with using all $n = N$ images are shown in Figure 7. For the same number of images ($n = 5$), we observe lot more albedo errors than in the case of sphere because this surface is much more complex (several patches of non-differentiability) and thus light is also reflected back from parts of the surface to other



(a) Albedo obtained for varying n



(b) N_x, N_y, N_z (top to bottom) for varying n

Figure 2: Obtained albedo and normals for gray sphere with varying $n \in \{1, 2, 3, 6, 9, 12, 15, 18, 21, 25\}$ images. Notice that for $n < 3$, we still get a reasonable albedo but the normals are unreliable. For $n >= 6$, we get reliable estimates for albedo and normals.

parts of the surface. With more observations, the number of outliers reduces drastically, and the quality of albedo improves because measurement noise reduces.

Answers to Question 5: In order to implement this, we use the same algorithm channel-wise (using `channel` argument) and then average the height map to get the final surface. The zero pixels are problematic since they lead us to the trivial solution $\mathbf{g}(x, y) = 0$ and thus making the albedo go to `NaN`'s as it involved division by $|\mathbf{g}(x, y)|$.

- `SphereColor`: For colored sphere, albedos, per-channel normals and averaged height map are all shown in Figure 8. The albedo looks good but the surface is very jittery. On further inspection, the B-channel outputs are the poorest possibly as the lines on the sphere cause non-differentiability making it difficult for the integral to be estimated.

- `MonkeyColor`: For colored monkey, albedos, per-channel normals and averaged height map are all shown in Figure 9. Note, the normals from G channel are all 0s (we converted `NaN`'s to 0s) since there is no information in the G channel in input images.

Answers to Question 6 (Yale Face images): The estimated height maps for Yale faces, without shadow-trick, and for various integral path types has been shown in Figure 10.

- For column-major path, one can see artefacts of horizontal lines cutting across the face whereas for row-major type, vertical lines. The average case tends to seem smoother

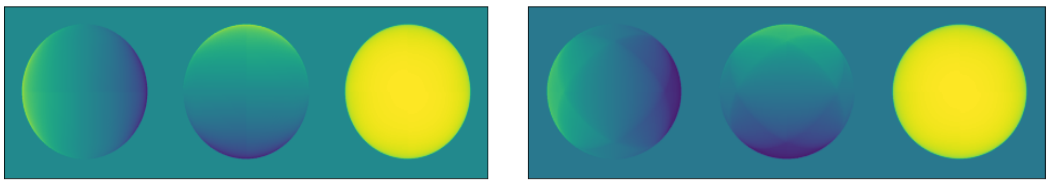


Figure 3: Comparing normals ((N_x, N_y, N_z) in order from L to R) for $n = 5$ with (L) and without (R) shadow trick. There are artefacts when not using the shadow trick potentially due to equations zeroing out for points under shadow.

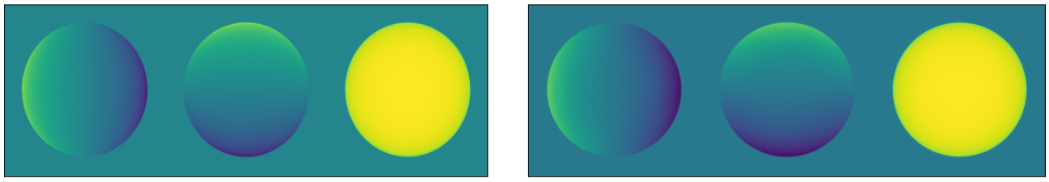


Figure 4: Comparing normals ((N_x, N_y, N_z) in order from L to R) for $n = 25$ with (L) and without (R) shadow trick. There is no significant difference between the normals.

since it averages out the artefacts in the individual cases. The albedos are quite similar in the three cases.

- We find several images that violate assumptions for shape-from-shading methods categorized as below (examples shown in Figure 11):
  - *Corrupted*: Certain images are simply corrupted in pixel space and are not good choices to be used in the optimization.
  - *Light behind the face*: The surface to be recivored is the face and when the light is behind the face, there is still some reflection off the face which is not ideal.
  - *Non-lambertian surface*: Even though photometric stereo assumes Lambertian surfaces, the face in question does not respect that assumption, specifically in some images where the reflectance from the nose/cheeks is much higher.
- Finally, we compare the height maps recovered with all images and without the noisy ones in Figure 11. The face seems to be "flatter" in the former while it seems to be more realistic (e.g. see nose) in the latter. This maybe because the linear equations around specific areas (e.g. nose) do end up with an apt solution while not using the noisy images. (Results generated in: `lab1/photometric/notebooks/question-6.ipynb`)

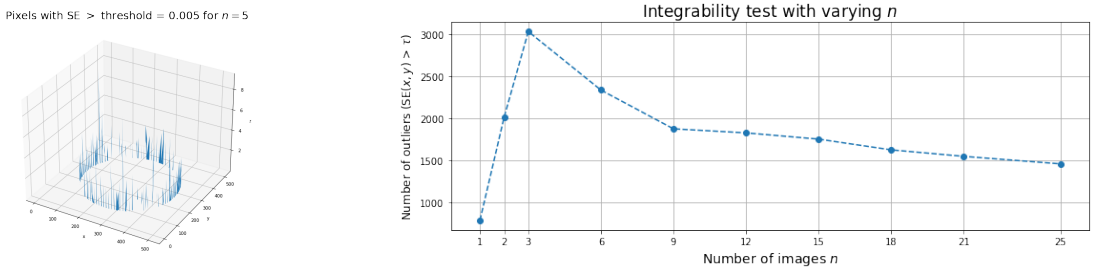
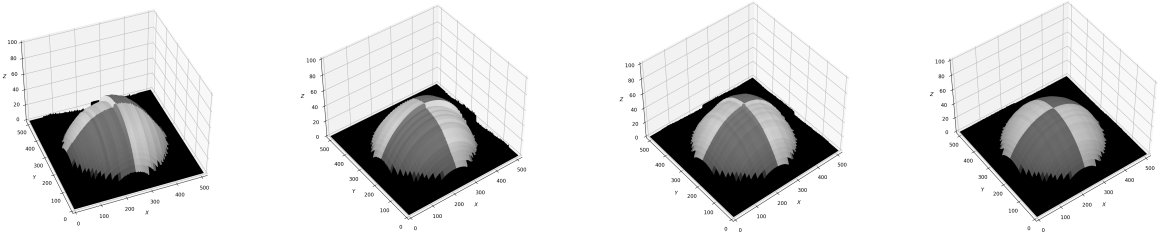


Figure 5: Number of outlier points: (Left) Case $n = 5$. (Right) Varying n.



(a) Column ($n = 5$) (b) Row ($n = 5$) (c) Avg. ($n = 5$) (d) Avg. ($n = 25$)

Figure 6: Reconstructed surface by integral via row/column path or averaged for $n \in \{5, 25\}$.

2 Color Spaces

- **RGB Colour Model**: RGB color model is used as a basis of our digital cameras owing to the tri-stimulus nature of human visual system. This is because we have three different kinds of cells (cones) on the retina, and, each of them responds selectively to a different portion of the color spectrum.
- **Colour Space Conversion**: All conversion results have been shown in Figure 12.
- **Colour Space Properties**

  1. **Opponent**: In opponent space, O_3 represents luminance whereas O_1 (red-green), O_2 (blue-yellow) represent chromatic channels. This space provides some level of brightness invariance since illumination is separated from chrominance.
  2. **Normalized RGB**: In this space, we discard the absolute intensity of a given color sample and just represent its pure color. Apparent color remains the same in this space as against RGB. For e.g., if a colored object moves is a scene not uniformly lit, then with RGB representation, apparent colors of the object will change as it moves but remain it will remain same with normalized RGB space.
  3. **HSV**: The HSV representation models how colors appear under light. HSV color space is the most suitable color space for color based image segmentation. Also,

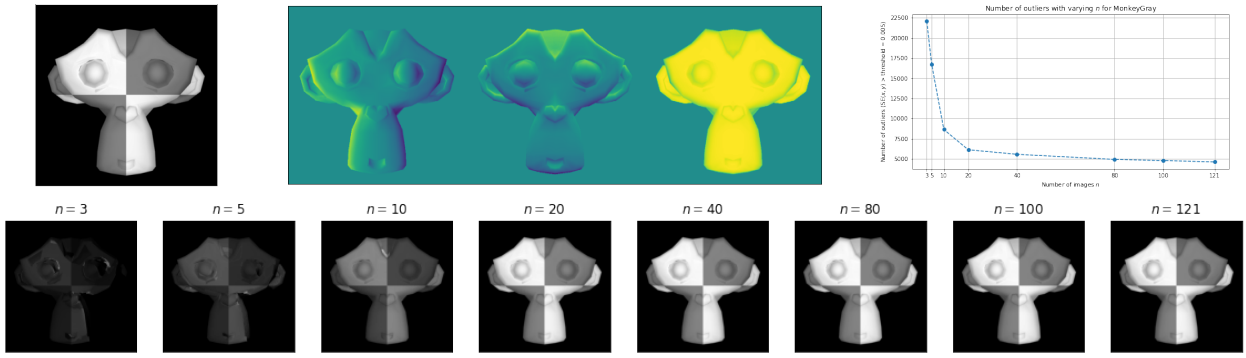


Figure 7: (From top, left) For `MonkeyGray`, albedo, normals, number of outliers vs n and albedo with increasing n.

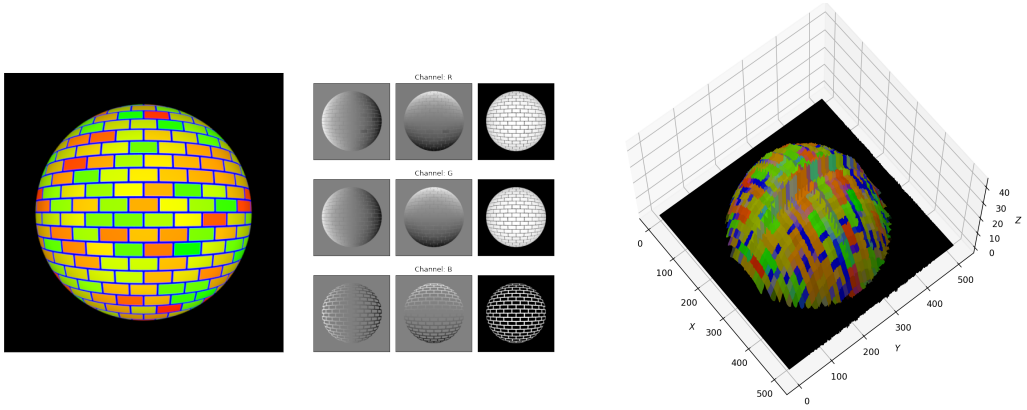


Figure 8: Results for `SphereColor`: (L to R) Albedo, normals per channel and surface averaged across channels.

HSV unlike RGB separates the image intensity from the color information. This is very useful in many applications. For example, in the case of histogram equalization of a color image, only the intensity component is needed and leave the color components alone.

4. **YCbCr**: Y is the luma component of the color which represents means the light intensity of the color, Cb and Cr represents the color components due to which YCbCr is used widely in video and image compression schemes such as MPEG and JPEG.
5. **Grayscale Colour Space**: Grayscale is used in the cases of image processing where the importance of color is not relevant like in the algorithm like edge detection, contour detection. The various gray scaling methods used in the assignment are:
   - **Lightness Method**: In this method an average is taken of the most prominent

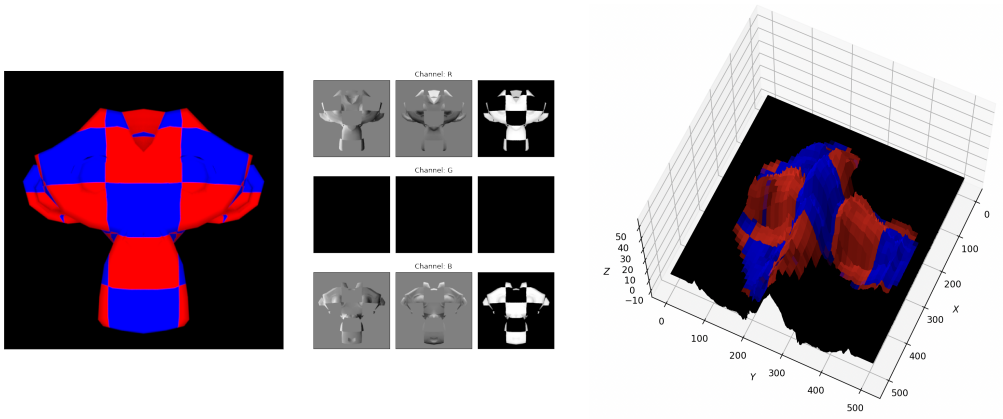


Figure 9: Results for `MonkeyColor`: (L to R) Albedo, normals per channel and surface averaged across channels.

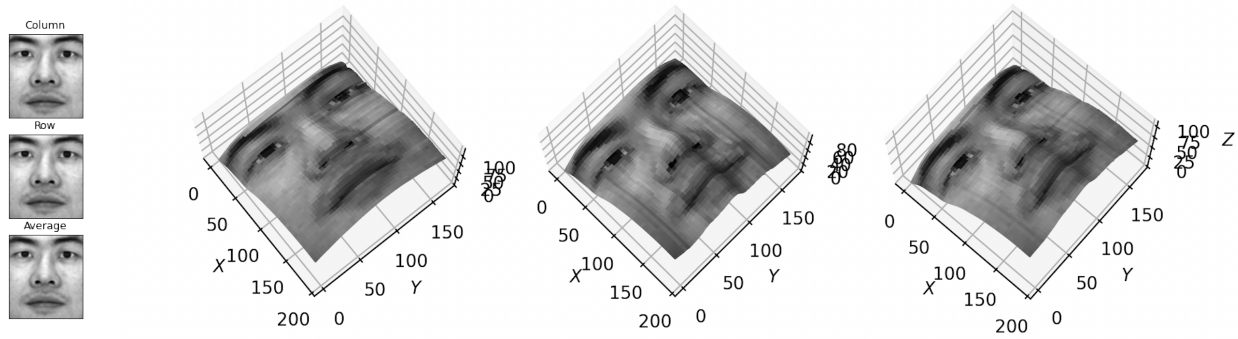


Figure 10: (Left) Albedos for each of the cases. (Right) Height maps for Yale face images: (L to R) (a) column-major (b) row-major (c) average.

and the least prominent color of the image which tends to reduce the contrast of the image.

- **Average Method**: In this method an average is taken of the R,G, and B channels of the image but it is not perceived well by human perception.
- **Luminosity Method**: This method takes in the weighted average of the R,G and B channels to account for human perception which is missing in the average method.
- OpenCV uses the following for Grayscale conversion $Y = 0.299R + 0.587G + 0.114B$.

- **More on Colour Spaces: L*a*b* color space**
  The $L * a * b$ color space consists of lightness component(L) whereas a represents the red/blue color value while b corresponds to the green/yellow color value. The a-axis runs from left to right. A color measurement movement in the $+a$ direction depicts a shift toward red. Along the b axis, $+b$ movement represents a shift toward yellow. The

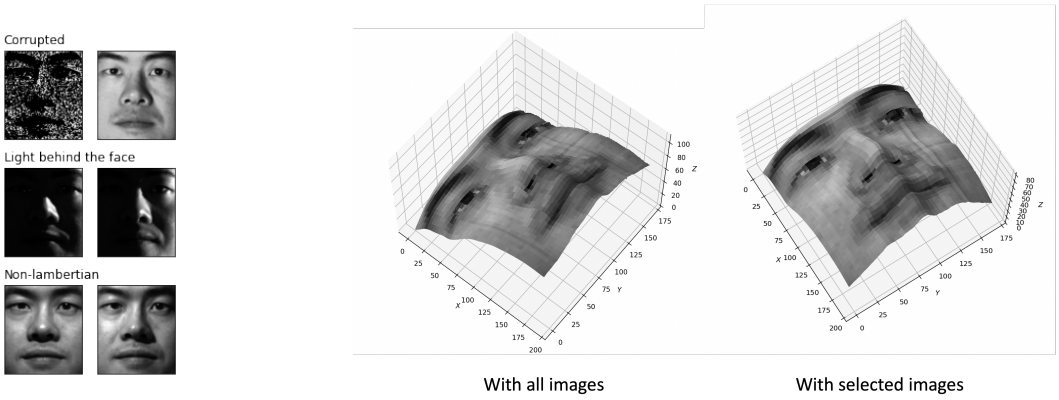


Figure 11: (Left) Samples of noisy images in Yale. (Right) Comparing height maps with and without noisy images.

center $L*$ axis shows $L = 0$ (black or total absorption) at the bottom. At the center of this plane is neutral or gray. While the $L * a * b*$ space is not truly perceptually uniform, it nevertheless is useful in industry for detecting small differences in color.

3 Intrinsic Image Decomposition

1. **Other Intrinsic Components**: Image intensity can be written as $\mathbf{I} = \kappa_d \mathbf{N}.\mathbf{L}$. We split it into κ_d (albedo) and $\mathbf{N}.\mathbf{L}$ (shading). Another way is to split it into albedo (reflectance), shape (surface normals $\mathbf{N}$) and illumination (lighting $\mathbf{L}$). In applications such as human face reconstruction, images are also decomposed into albedo, shape and textures.

2. **Synthetic Images**: Since it is practically impossible to split a physical scene image into its perfect ground-truth decomposition: albedo and shading, we see that most datasets for IID are sythetic because we can simulate and get ground-truths.

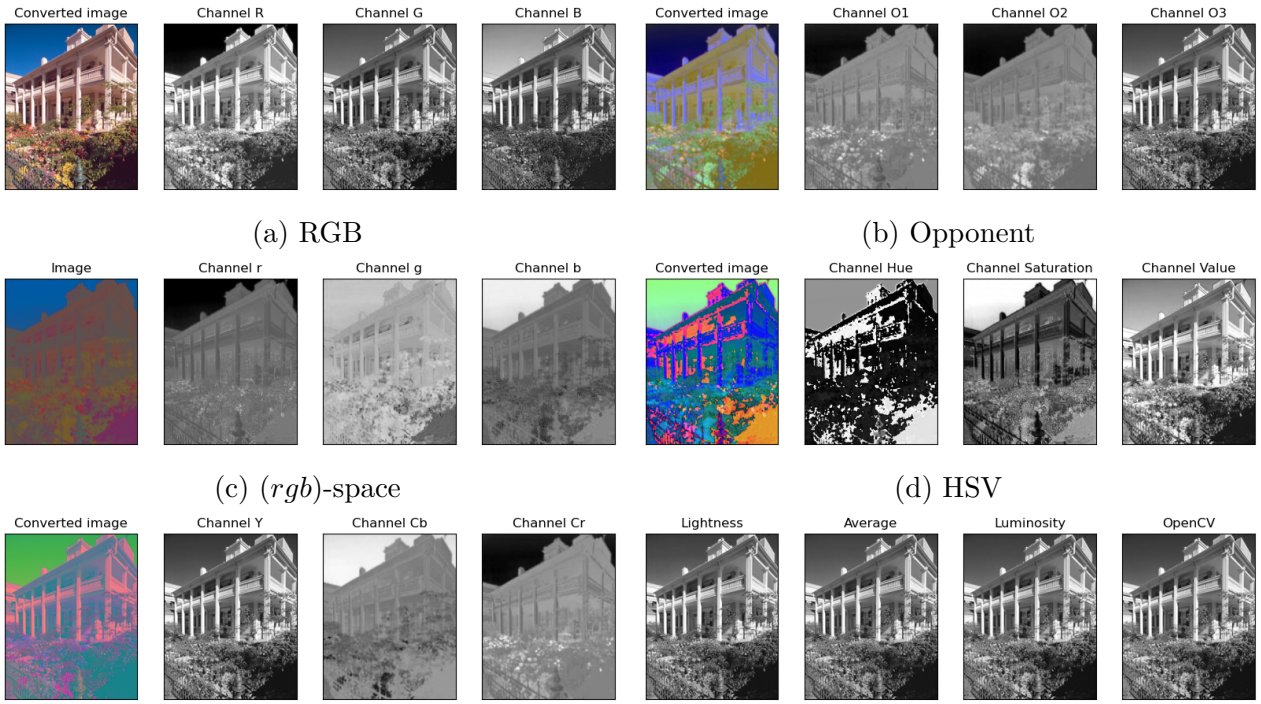
3. **Image Formation**: See Figure 13 for an example of image decomposed into albedo and shading and the reconstructed image.

4. **Recoloring**:

   (a) The original color of the ball (from albedo) in RGB space is $[184, 141, 108]$.

   (b) For the given ball image, the original & recolored images are shown in Figure 14.

   (c) Although we have recoloured the object with pure green, note that we recolor in the albedo only. The other component (shading) is driven by incident lighting as well as the surface normals of the object. Thus, the color does not appear to be pure green and is also non-uniform over the object.



(a) RGB (b) Opponent

(c) (rgb)-space (d) HSV

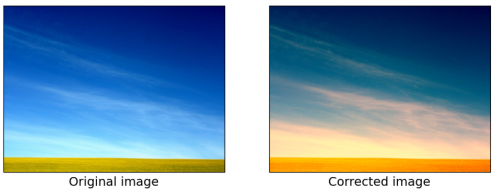
(e) $YCbCr$ (f) Grayscale

Figure 12: Results of conversions in color spaces for a sample image.

Figure 13: Example of intrinsic image decomposition: From left, original image, its albedo and shading and then reconstructed image via elementwise product of albedo and shading.

Figure 14: Recoloring of an object using its intrinsic decompositions (albedo in this case).

(a) Given example.



(b) Failure case.

Figure 15: Gray-World algorithm examples: Image correction

4 Color Constancy

4.1 Grey-World Algorithm

1. The original and corrected image (obtained via Grey-world algorithm) is shown in Figure 15a. We also visualized the average color in original vs corrected images. In the original one, it seems reddish-grey and the cleaned one seems close to pure grey, reinforcing the Gray-world assumption and that it works on this image.

2. The Grey-world algorithm is based on the fundamental assumption that under white light, the average color in a scene should be grey. Thus, it may not be apt when an image has a large dominant image patch. For example, if we consider a image of open sky, blue would be the dominant color and this algorithm will not work. See Figure 15b for an example. (This image was taken from Google.)

3. Another example of a color constancy algorithm is the White-Patch algorithm (WPA). It is a special case of the Retinex algorithm by Land and McCann (1970). WPA assumes, for each color channel c, $\exists(x_o, y_o)$ in the image with maximal reflection of the source light for that channel. Further, when these maximal reflections are brought together, they form the color of source light.

Conclusion

In this assignment, we implemented and experimented with a wide range of topics within low-level vision from photometric stereo to color constancy algorithms. We recognize the importance of simple, classical techniques that could be used for image manipulation and understanding. Additionally, we appreciate the fine-grained details associated with implementing such algorithms, like handling numerical stability, managing range of pixel values properly etc.