Photometric Stereo and Color

Tarun Krishna
tarun.krishna@student.uva.nl
Dhruba Pujary
dhruba.pujray@student.uva.nl

February 21, 2018

1 Introduction

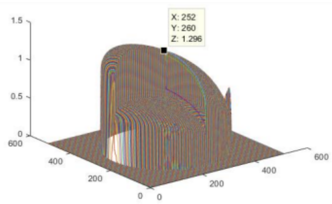
In this report, we analyze photometric stereo, color space, color constancy algorithm and report our findings

2 Photometric Stereo

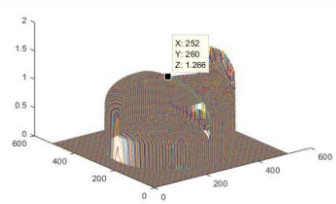
2.1 Estimating Albedo and Surface Normal

$$\mathbf{i}(x, y) = \mathcal{V}\mathbf{g}(x, y) \tag{1}$$

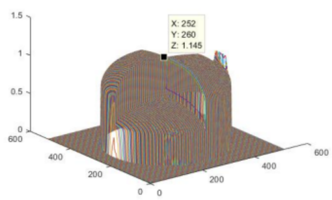
Typically, n $>$ 3 so that a least squares solution is appropriate. So, minimum number of images one needs to estimate albedo and surface normal is least three. We tested with both the strategy, i.e. all the images at once and in an incremental way. The final output of both the strategy is the same however doing it in an incremental way gave us better insights. Figure 1 shows the intensity plot of the object. In case of using only 5 images from a stack of *SphereGray25*, the intensity plot is distorted. This is because the light source of these images is only in two of the direction, i.e. images are partially illuminated. This gets better when using 10 images as the lighting condition improves with errors along the sides. These errors get removed as we use more number of images. Lastly, comparing the intensity plot of SphereGray25 and SphereGray5 we can see that SphereGray25 gives better result as compared to SphereGray5 because of using more images. Also to note that the shape of the sphere is distorted for less number of images, as you increase the number of images you get a better shape. But one may argue why do we have a nice shape for a bottom-right image which is reconstructed from images in *SphereGray5*? That is because the light directions are not uniform but that not the case with top-5 images from the *SphereGray25*. More details in the subsequent paragraph.



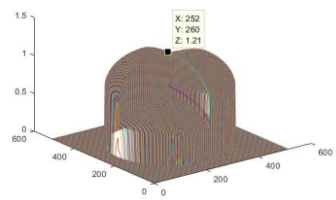
(a) 5-images



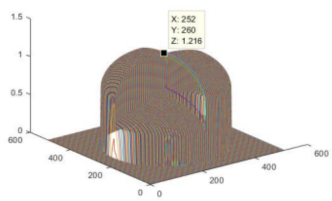
(b) 10-images



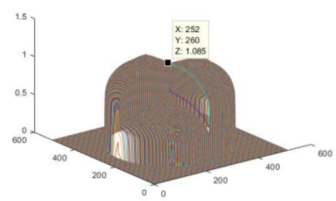
(c) 15-images



(d) 20-images



(e) 25-images



(f) 5-SphereGrey5

Figure 1: Intensity plot with 5 images (Top-Left), 10 images (Top-Right), 15 images(Middle-Left), 20 images(Middle-Right) and 25 images(Bottom-left) of SphereGray25. Bottom right images is intensity plot of using all 5 images of SphereGrey5.

[1] For different lights, a substantial region of the image surface may be in a shadow as a result that part may not be observed in Albedo. To overcome this problem, we use shadow trick. We modify equation1. If there really is no ambient illumination, then we can form a matrix from the image vector and multiply both sides by this matrix; this zeroes out points that are in shadow. $\mathcal{I}$ has the effect of zeroing the contributions from shadowed regions because the relevant elements of the matrix are zero at points that are in shadow(kind of

[1] What is the impact of shadows in photometric stereo? Explain the trick that is used in the text to deal with shadows. Remove that trick and check your results. Is the trick necessary in the case of 5 images, how about 25 images?

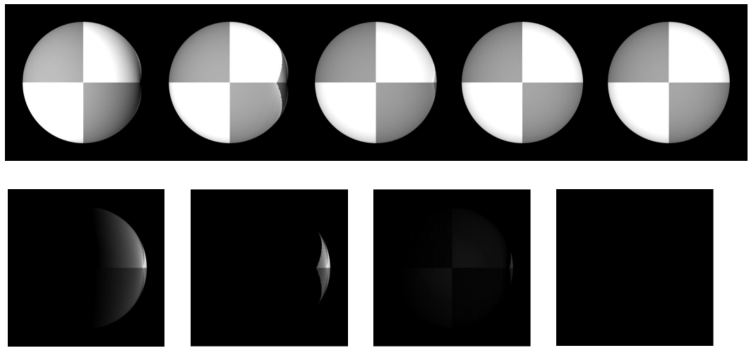


Figure 2: Albedo evaluated for different number of images(5,10,15,20,15 Top row from left to right shows receptively). Difference of albedo evaluation for 5, 10, 15, 20 images respectively with respect to 25 images.

regularization as we do in ML). Again, there is one linear system per point in the image; at each point, we solve this linear system to recover the g vector at that point. Now the equations become:

$$\mathcal{I} = \mathcal{I}\mathcal{V}\mathbf{g}(x, y) \tag{2}$$

Figure 3: Left Images is the reconstructed Albedo using images in *SphereGray5* and right is reconstructed albedo using 25 images in *SphereGray25*

After removing the trick images are depicted in Figure3. Trick is not necessary in this case of 5 images as well as 25 images. In case of *SphereGray5* images, lightning is uniform i.e we have light coming from $(-1, 1), (1, 1), (0, 0), (1, -1), (-1, -1)$. As result all the image portions has been covered under these lightening directions. But if you take 5 images from the stack of *SphereGray25* and compare the albedo of two of them as depicted in Figure4 you will see the clear difference. Albedo for *SphereGray5* looks complete but images from *SphereGray25* looks incomplete as some portion on the right side is still dark(so-called shadow effect) due to the direction of light for those 5-images misses some parts of the images. But if you run on all the images of *SphereGray25* you will see the perfect albedo image as depicted in Figure3 with high brightness as compared Albedo for *SphereGray5*. As, in this case images have been taken under various light directions covering whole of the image.

Figure 4: Left Images is the reconstructed Albedo using images in *SphereGray5* and right is reconstructed albedo using top-5 images from stack of *SphereGray25*

2.2 Test of Integrability

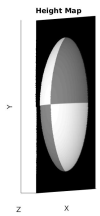
In order to get more insights about the errors we evaluated the difference of albedo for using k number of images where k varies from 5, 10, 15, 20 to 25 images. We can see that the error or the difference of correctly evaluating the albedo is getting reduced at every step as we increment the number of images. Well, we also tried to find some relationship between the number of outliers and number of images as depicted in Figure10. Ideally, we thought as the number of images increase we should get less outliers but this is not the case. We couldn't find any patterns to report our belief as depicted in the Figure 10. Our threshold was .005.

2.3 Shape by Integration

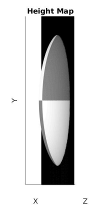
Well one could clearly spot the difference in images when zoomed in. Column reconstructed images have rough line patterns along the row on the other hand row reconstructed images have the same pattern along the column. But once you average them this pattern sort of get smoothed but not disappears. As depicted the Figure5

2.4 Experiments with Different Objects

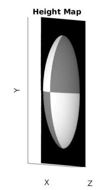
The albedo results for monkey are different from the sphere because the surface of monkey is not uniform, smooth and cast self-shadows. The albedo around the eyes is not uniform because of self-shadow. The figure 9 shows the difference of



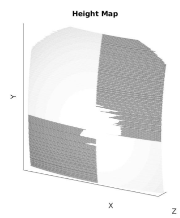
(a) Column map



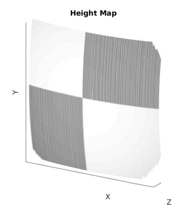
(b) Row map



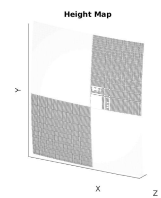
(c) Average map



(d) Column zoom

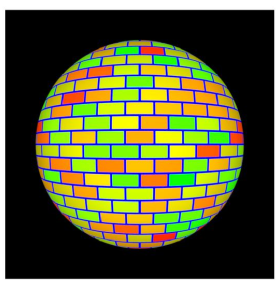


(e) Row zoom

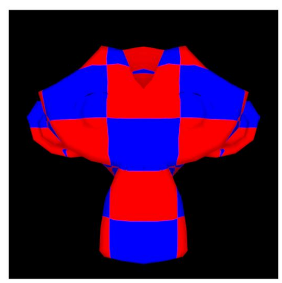


(f) Average zoom

Figure 5: Depicts the height map for Each of the methods. Along with zoom images to look into the distortions found



(a) Sphere



(b) Monkey

Figure 6: Albedo Images for Sphere and Monkey

albedo with increasing number of images. It shows that the lighting condition is not uniform with increasing number of images. ,i.e. no uniform pattern. And this could be the reason of large number of outliers as evident in the Figure 10 .

RGB-implementation for Albedo, Normal: To tackle RGB images for the calculation of Albedo we split images in to R, G and B channel and calculate Albedo for each of the channel as we did for the gray images in previous sections and finally combine them in to RGB albedo image as depicted in Figure6. For

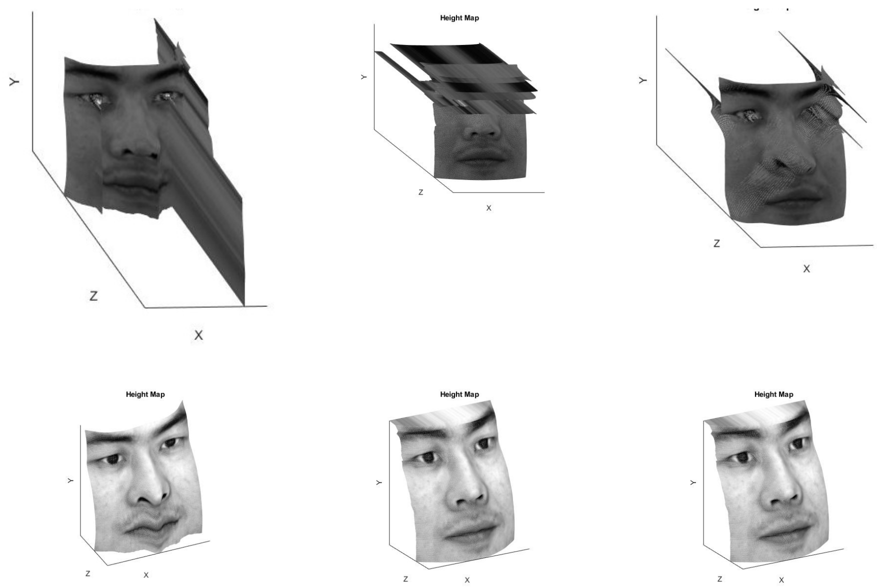


Figure 7: Height map of Yale Face with shadow trick (Top) and without shadow trick(Bottom).(row, column,average respectively)

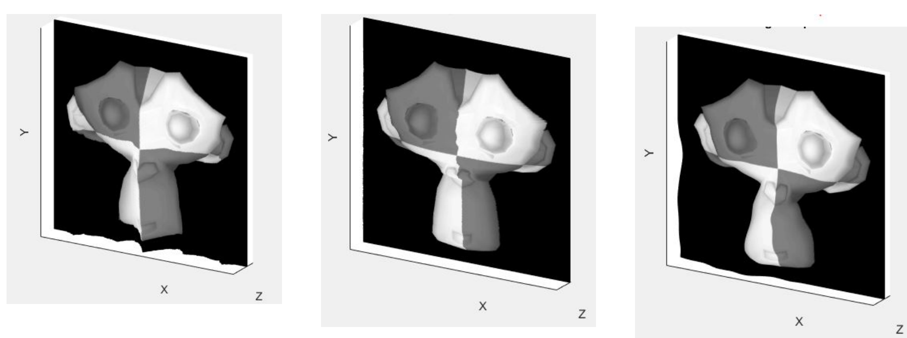


Figure 8: Height map when using row, column and average strategy

calculation of normal we used two techniques "max" and "average". But before we go into details one thing should be clear that RGB images doesn't effect the height map which is our final goal(R-channel would not change the height nor the B and G channel). One simplest approach for calculation normal map for RGB images would be to calculate Normal and Albedo separately i.e you calculate albedo for RGB as mentioned above and calculate normal map for RGB images by converting it into gray and finally their height map(This approach

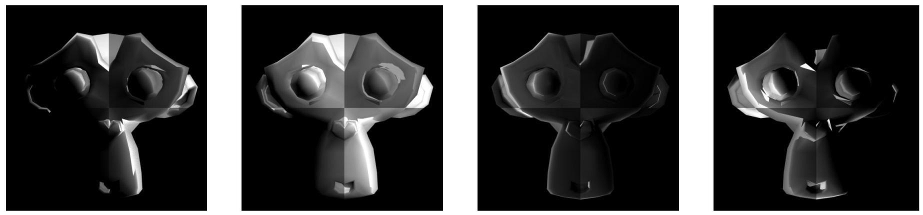


Figure 9: Difference of albedo with increasing number of images (24, 48, 72, 96, 120

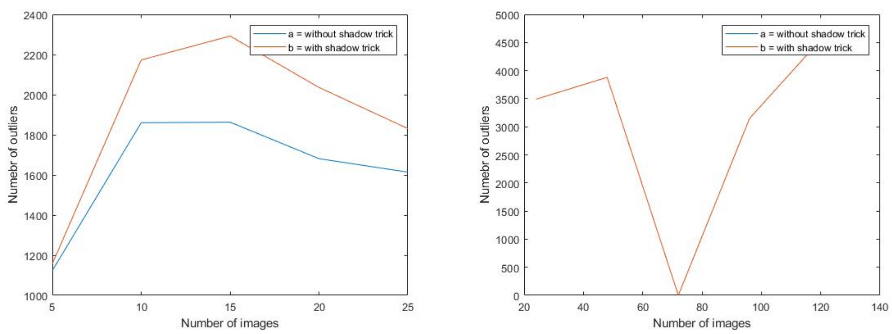


Figure 10: Plot of number of outliers when using increasing number of images. Sphere(left); Monkey (right).

could be more computationally efficient as we don't need to compute normals for each channel). However, we calculate normals for each of the channel and take their "max" and "average" in each channel to get final representation of normal for an RGB image. Figure14 and Figure15 depicts result for height map using "max" and "average".

When testing the **Yale face dataset** (Figure 7), using different row, column and average strategy for evaluating hight map, we get different output using shadow trick than without. When using shadow trick, evaluating the row strategy propagates the error along the row and highlights it along the column. Whereas when using the column strategy the error along the column is propagated and gets highlighted along the row. Taking the average strategy reduces the error to some extent. Without the shadow trick, the error of the image are reduced significantly. The reason of the better result is because the surface of the face is not opaque which is a violation of shape from shading. There are patches of light that gets reflected from nose, eyes etc. We tried removing images 16 that are lighted partially but this didn't made any visual difference.

3 Color Spaces

3.1 RGB Color Model

Why do we use RGB color model as a basis of our digital cameras and photography? The main purpose of the RGB color model is for the sensing and representation. The choice of primary colors is related to the physiology of the human eye; good primaries are stimuli that maximize the difference between the responses of the cone cells of the human retina to light of different wavelengths, and that thereby make a large color triangle. That's the reason why we use RGB color model in digital cameras and photography.
How does a standard digital camera capture the full RGB color image?When you take a picture the shutter opens briefly and each pixel on the image sensor records the brightness of the light that falls on it by accumulating an electrical charge. Pixels capturing light from highlights in the scene will have high charges. After the shutter closes to end the exposure, the charge from each pixel is measured and converted into a digital number. This series of numbers is then used to reconstruct the image by setting the color and brightness of matching pixels on the screen or printed page. This RGB system is used whenever light is projected to form colors as it is on the display monitor (or in your eye). Since daylight is made up of red, green, and blue light; placing red, green, and blue filters over individual pixels on the image sensor can create color images.

3.2 Color Space Properties

Opponent Color space : One of the properties of RGB is that the values of the three channels are highly correlated. De-correlating the RGB color space leads to an opponent color space. Focusing on the chromatic channels (i.e., red–green and blue–yellow), they are opponent in two different ways. First, no color seems to be a mixture of both members of any opponent pair (e.g., no color ever seems yellowish-blue, while greenish-blue is often encountered). Secondly, each member of an opponent pair exhibits the other, that is, by adding a balanced portion of two opponent colors, gray will be the result.

Besides being intuitive, an additional advantage of this color system is that it largely de-correlates the RGB color channels. Refer Figure17.
Normalized Color Space: At times, you want to get rid of distortions caused by lights and shadows in an image. Normalizing the RGB values of an image can at times be a simple and effective way of achieving this. A simple basic task of detecting an object can be tedious. Because light also plays a crucial role. For example, if you have unidirectional source of light then it creates shadow and different shades of colors on object as you can see Figure**??**. So to reduce the effects of light, Normalization of color space is helpful. Normalization removes highlighted regions, shadows and make that object easier to detect which is quite difficult in RGB color space. As it is clearly observed from the Figure17 shadows due to clothing in the background and in front has been completely removed.

Background looks uniform in Normalized image.

HSV Color Space: Unlike RGB, HSV separates luma, or the image intensity, from chroma or the color information. This is very useful in many applications. For example, if you want to do histogram equalization of a color image, you probably want to do that only on the intensity component, and leave the color components alone. Otherwise you will get very strange colors. These models were useful not only because they were more intuitive than raw RGB values, but also because the conversions to and from RGB is extremely fast to compute.

Also, just suppose we have an image of a single-color plane with a shadow on it(Figure17 background with single color). In RGB color-space(Figure17), the shadow part will most likely have very different characteristics than the part without shadows. In HSV color-space(Figure17 **h channel**), the hue component of both patches is more likely to be similar: the shadow will primarily influence the value(**v-channel** image background has information about shadow), while the hue, indicating the primary "color" (without it's brightness and diluted-ness by white/black) should not change so much. All these observations can be clearly inferred from Figure17.

YCbCr Color Space: YCbCr is a practical approximation to color processing and perceptual uniformity. By doing this, subsequent image/video processing, transmission and storage can do operations and introduce errors in perceptually meaningful ways. YCbCr is used to separate out a luma signal (Y) that can be stored with high resolution or transmitted at high bandwidth, and two chroma components (CB and CR) that can be bandwidth-reduced, subsampled, compressed, or otherwise treated separately for improved system efficiency. That's why YCbCr is used widely in video and image compression schemes such as MPEG and JPEG. Figure17, Y component looks mainly like gray image(that's why transmitted at high bandwidth) and carries all the information as compared to Cb and Cr. These components are less sensitive to the human eyes. Well that's not the case in RGB images, these RGB signals are not efficient as a representation for storage and transmission, since they have a lot of redundancy.

[2] **CMYK** color space is used in the printing process, because it describes what kind of inks need to be applied so the light reflected from the substrate and through the inks produces a given color.

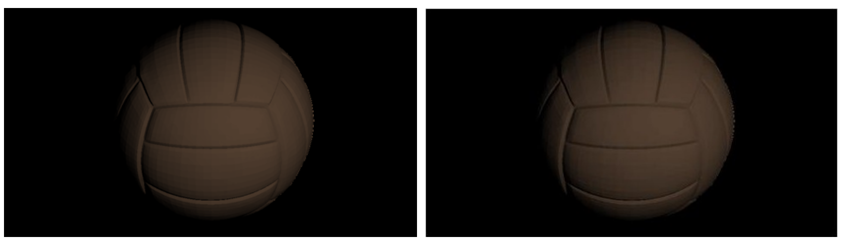
The **Adobe Wide Gamut RGB** color space is an RGB color space developed by Adobe Systems as an alternative to the standard sRGB color space. It is able to store a wider range of color values than sRGB. The Wide Gamut color space is an expanded version of the **Adobe RGB** color space.

[2] Find one more color space from the literature and simply explain its properties and give a use case?

4 Intrinsic Image Decomposition

[3] Intrinsic image decomposition is the process of separating an image into its formation components or its physical meaningful constituents such as illuminance, reflectance, surface orientation, shape, texture etc. Every object would have a surface orientation which is the surface normal. The shape of the object is also an physical property of the object and also the texture of the material of the object. The shape of the object can result into different shading effect, like overlap. And the diffusion of incident light rays can be influenced by the texture of the material.

[4] The reason for using synthetic dataset in image decomposition is because of difficulty in obtaining such a dataset without noise. For example, to obtain true reflectance of an object, the source light should by white. To obtain true white light for capturing reflectance is a difficult task. Another example would be obtaining the shading image of an object. There is always a possibility of light reflected from the surrounding which will add to noise of the shading image.



(a) Original image (b) Reconstructed image

Figure 11: Original image(Top-Left) and reconstructed image(Top-Right) with the given reflectance and shading image.

4.1 Recoloring

[5] The true color of the material is the value of the RGB channel of body reflection.This is could be calculated from reflectance image given to us. This is equal to R: 184, G: 141 and B: 108.

[6] The reconstructed images do not seems to display those pure colors because of the shading effect. The light from the source reaching the object is reflected

[3]What other components can an image be decomposed other than reflectance and shading ? Give at least 2 examples and explain your reasoning.

[4]If you check the literature, you will see that almost all the intrinsic image decomposition datasets are composed of synthetic images. What might be the reason for that ?

[5]Find out the true material color of the ball in RGB space.

[6]Although you have recolored the object with pure green and magenta, the reconstructed images do not seem to display those pure colors and thus the color distributions over the object do not appears uniform. Explain the reason.

(a) Original image (b) Pure Green (c) Pure Magenta

Figure 12: Recoloring the ball(Left) to pure green(Center) and magenta(Right)

in all the direction. Because of the curvature of the object, light doesn'treaches all section of the object uniformly. Some of the portion of the object is covered by other and forms a shadow. In addition, all the light reflected back from the object is not captured by the camera as most of the light gets dispersed depending on the shape of the object. Thus because of both the phenomena happing simultaneously, the true color of the object is not perceived to a viewer.

(a) Original image (b) Corrected image

Figure 13: Original image(a) and corrected image(b) using Grey-World algorithm.

[7] The assumption behind grey-world algorithm is that on an average the color of the scene is achromatic under white light. The algorithm will fail in scenarios where the color distribution is not uniform. For example, a image of an ocean under the blue sky. Most of the pixels in this image will have high blue values and doing a grey-world correction would result a wrong image.

[8] Two other algorithm for color constancy are White-Patch algorithm and Grey-Edge algorithm. The White-Patch algorithm assumes that the maximum response in an image is caused by a prefect reflectance. A surface with perfect reflectance properties will reflect th full range of light that falls on it. As a result,

[7] Give an example case for Grey-World Algorithm on where it might fail.

[8] Find out two more color constancy algorithm from the literature and explain them briefly.

the color of this perfect reflectance is exactly the color of the light source. In practice, this white patch is found by searching for the maximum intensity in each channel. After that all the pixel intensities are scaled. The Grey-Edge algorithm assumes that the average edge difference in the scene is achromatic. Following this assumption, the light source color can be computed from the average color derivative in the image.

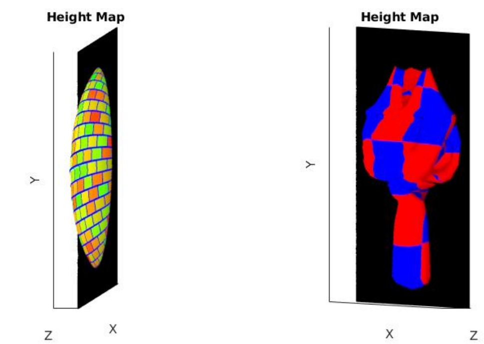


Figure 14: Height map with average approach using column reconstruction

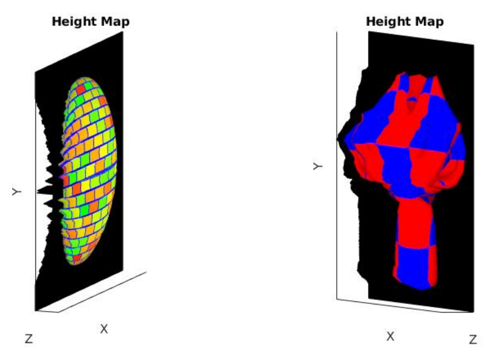


Figure 15: Height map with max approach using column reconstruction

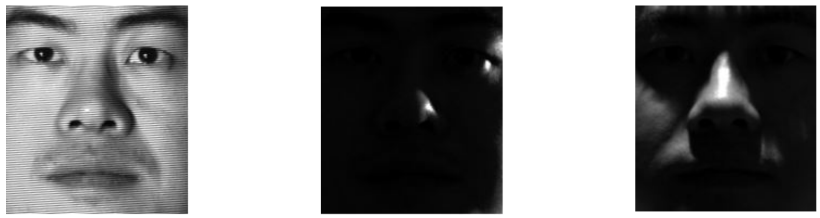


Figure 16: Face that violates shape-from-shading and noisy images

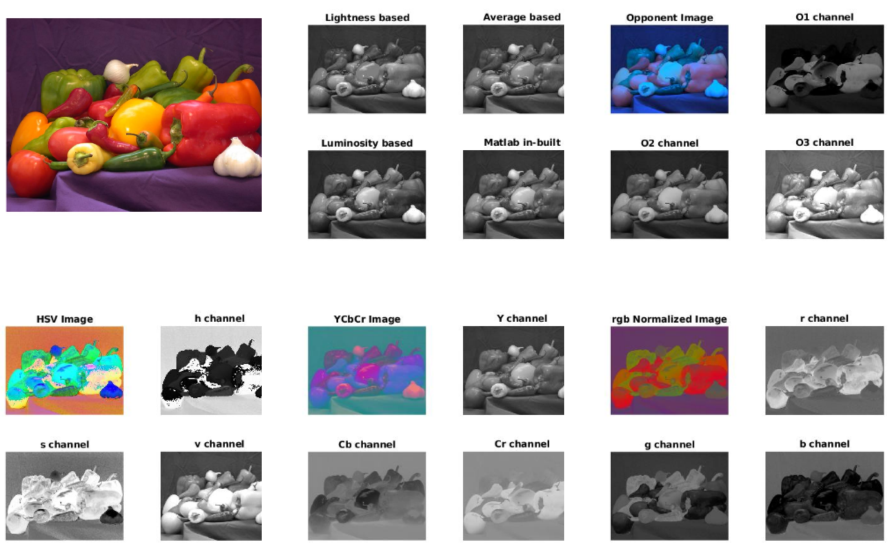


Figure 17: Different color space