Assignment 1: Photometric Stereo & Colour

(Team 52A) Rajeev Verma 13250507, Jeroen Taal 11755075

September 14, 2020

1 Introduction

In this report we will discuss multiple methods within the domain of image formation. First we will apply the Photometric Stereo algorithm on multiple datasets. Thereafter we will investigate different colorspaces in which images can be represented. Next, we will discuss and show the intuition behind Intrinsic Image Decomposition. At last, the workings of the Gray-World color constancy algorithm are discussed and implemented.

2 Photometric Stereo

In this section, we will implement the photometric algorithm as presented in Forsyth and Ponce (2012). First we estimate the Albedo and surface normal of a model of images of an object under different light sources. Thereafter we compute the partial derivatives along both axis. Using these partial derivatives we will then construct a 3-dimensional height map of the object in the images.

2.1 Estimating Albedo and Surface Normal

In order to estimate the Albedo and surface normal vectors of our dataset, we need to make some assumptions. We assume that there is no ambient illumination and that the camera response is linear in the surface radiosity. Then we can write the intensity of a pixel at coordinates x, y in image I as follows:

$$I(x, y) = g(x, y) \cdot V_1 \tag{1}$$

In which V_1 contains information about the location of the illumination source. $g(x, y)$ is defined as $g(x, y) = \rho(x, y) \cdot N(x, y)$, in which $\rho(x, y)$ is the surface radiosity at x, y and $N(x, y)$ is a unit normal vector of the surface at x, y.

For our dataset of n images, we can stack the measurements at position x, y into a vector $i(x, y) = [I_1(x, y), ..., I_n(x, y)]^T$. Our measurements can be stacked into a matrix:

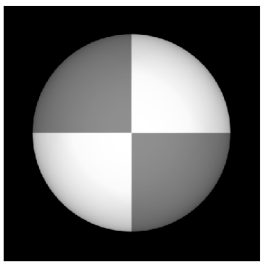
$$\mathcal{V} = \begin{bmatrix} V_1^T \\ \vdots \\ V_n^T \end{bmatrix}$$

Equation 1 can then be written as:

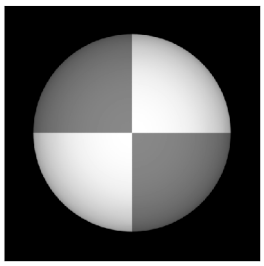
$$i(x, y) = \mathcal{V} g(x, y) \tag{2}$$

This linear system is solved for $g(x, y)$ using a least squares solution. The Albedo can than be retrieved from $g(x, y)$. Because $N(x, y)$ is a unit vector, the magnitude of $g(x, y)$ should be attributable to the radiosity: $\rho(x, y) = |g(x, y)|$. $N(x, y)$ can then be calculated by: $\frac{g(x,y)}{|g(x,y)|}$. In our implementation, the numpy.linalg.lstsq[1] function is used to solve this equation.

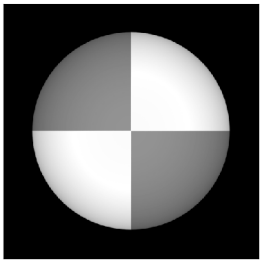
The Albedo estimated using the SphereGray5 dataset should look like the object with a uniform surface. This is because in the dataset, each pixel on the object surface is illuminated in at least one viewpoint.



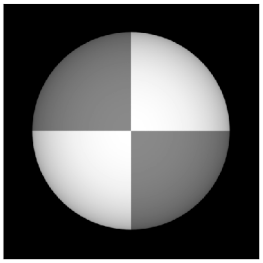
(a) SphereGray5 without shadowtrick



(b) SphereGray25 without shadowtrick



(c) SphereGray5 with shadowtrick



(d) SphereGray25 with shadowtrick

Figure 1: The estimated Albedos using the method described above. The Albedos estimated without the shadow trick are on the top row, wheras the Albedos estimated with the shadow trick are on the bottom row.

The results of estimating the Albedo are presented in figures 1a and 1b. In figure 1a, we can see that the Albedo is not entirely in concordance with our expectations described above. Eventhough the entire object is visible in the Albedo, its surface reflection is not uniform. The intensity of the surface in the center of the image is greater than the intensity of the surface towards the edges of the object.

The minimum number of images necessary to solve equation 2 is 3. As described in Forsyth and Ponce (2012), least squares is used to solve this system of equations, this requires at least 3 datapoints. In our context 3 datapoints refers to 3 different viewpoints, thus 3 different images. However, the Albedo estimated using 5 images, does not show a uniform surface reflection. Therefore,

[1] https://numpy.org/doc/stable/reference/generated/numpy.linalg.lstsq.html

this method is also applied on a dataset of 25 images of the same object. The resulting Albedo can be seen in figure 1b. This Albedo shows a more uniform surface reflection.

As the light sources do not illuminate the entire object, different parts of the surface are in shadow in the different viewpoints. A pixel in shadow has an intensity of zero. This could have a negative impact on the least squares solution of $g(x, y)$. In order to ignore measurements of pixels in shadow, we can zero these measurements out of equation 2. To do this, we have to construct the diagonal matrix:

$$\mathcal{I} = \begin{bmatrix} I_1(x,y) & \dots & 0 \\ \dots & \ddots & \dots \\ 0 & \dots & I_n(x,y) \end{bmatrix}$$

Equation 2 then becomes:

$$\mathcal{I}i(x, y) = \mathcal{I}\mathcal{V}g(x, y) \tag{3}$$

This way, shadow measurements are zeroed out of the equation. The results of the Albedos estimated using this shadow trick are depicted in figures 1c and 1d. However, these figures do not show a clear improvement with respect to figures 1a and 1b respectively.

However, there is a difference visible in the normals of the SphereGray5 dataset. As is visible in figures 3a and 3b (presented in appendix A), all three components of the surface normal vectors are more uniform when the shadowtrick is applied. In figure 4a and 4b (also presented in appendix A), it is visible that this is also the case for the z component of the estimated normal vectors of the SphereGray25 dataset. However, there is no difference visible in the x and y components of these normal vectors. Thus the shadowtrick seems to have less effect when more images are used to estimate the surface normals.

2.2 Test of Integrability

Under the orthographic camera model and Monge patch representation, the surface is represented as $(x, yf(x, y))$. Accordingly, the normal at (x, y) is

$$\mathcal{N}(x, y) = \frac{1}{\sqrt{1 + \frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}} \left\{ -\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1 \right\} \tag{4}$$

From equation 2, $\mathcal{N}(x, y)$ at some point (x, y) is given as: $a(x, y)\hat{i} + b(x, y)\hat{j} + c(x, y)\hat{k}$ where $\hat{i}, \hat{j}, \hat{k}$ are basis vectors in x, y, z directions respectively. Thus, we can compare and write

$$p = \frac{\partial f}{\partial x} = \frac{a(x, y)}{c(x, y)}; q = \frac{\partial f}{\partial y} = \frac{b(x, y)}{c(x, y)} \tag{5}$$

We also have

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2 f}{\partial x \partial y} \tag{6}$$

This gives us a check on our computation of the normals, as now for every (x, y), $\frac{\partial p(x,y)}{\partial y}$ should be approximately equal to $\frac{\partial q(x,y)}{\partial x}$. We can also write this as squared error $SE(x, y)$ as

$$SE(x, y) = \left(\frac{\partial p(x, y)}{\partial y} - \frac{\partial q(x, y)}{\partial x} \right)^2 < \epsilon \tag{7}$$

This check works depending on some reasonable choice of small threshold ϵ ($\epsilon \to 0$, ideally). However, our computation of normals and therefore, p and q are correct only to a certain degree. First of all, Least square method to $Ax = b$ gives only an approximation to x. It might happen that the n images taken under n different illumination conditions (orientation of the sources in 3-dimensional coordinate system) might only be able to span a small portion of the column space. Therefore, it's also the case as the number of images are increasing, we are getting smoother approximation to the albedo (as shown in section 2.1). Accordingly, we also have decreased number of outliers (with the same ϵ) with SphereGray25 as compared with SphereGray5, as can be seen in figure 2. Another possible source of error is the numerical approximation of derivative using neighbor difference operation.

2.3 Shape By Integration

Construction of the surface can be done by integrating over the derivatives of the columns of rows. Since we are integrating over a discrete function, we can simply sum the derivatives. However, there are different paths we can take. The three different paths we will discuss are: Column-major, Row-major, and the average of Column-major and Row-major. In the Column-major path, we first sum the derivatives of the left-most column. Thereafter we process each row, starting at the second element. For the Row-major path, we first process the first row, thereafter we process each column starting at the second element. The results of these algorithms are presented in figure 5 for 25 images and in figure 6 for 5 images, the figures are present in appendix A.

In figure 5 we can see that the three different algorithms using the 25 images dataset resulted in different surfaces. The Column-major algorithm resulted in a surface with a ridge parallel to the y-axis, as visible in figure 5a. The Row-major algorithm resulted in a surface with a ridge parallel to the x-axis, as visible in figure 5b. The average of these methods resulted in two ridges crossing each other in the center of the surface, as visible in figure 5c. In case of the average algorithm, the two ridges correspond with the two lines dividing the object in four patches as visible in figure 1. The ridges are less extreme than in the Column-major and Row-major results. However, in case of the Column-major algorithm, only the vertical line is visible as a ridge, whereas in only the horizontal line is visible with the Row-major algorithm.

The difference with the surfaces constructed using only 5 images can be seen in figure 6. The overall shape of the surface is similar with the surfaces in figure 5. However, the ridges are less extreme. Thus for both datasets, we can conclude that using both paths improved the construction of the surface maps. However, there was no improvement visible in using 25 images over 5 images.

The ridges in the surfaces are caused by the different patches in the original object. The derivative of pixels within such a patch is not large as the changes

in pixel value seem to be minimal. However, along the borders of these patches the derivative will be large as there is a sudden change of color from light to dark. Thus the derivative around these borders is larger. When summing the derivative per row in the Column-major algorithm, we encounter only the vertical border, hence the ridge is perpendicular to the x-axis. For the Row-major algorithm this we will only encounter the horizontal border. When we take the average of these approaches, we will see both ridges but to a lesser extend.

2.4 Different objects

The algorithm can be extended to different pictures and objects, which we will show in this section. In figure 8 in appendix A, we can see the results of running the photometric stereo algorithm on the MonkeyGray dataset. Figure 7b shows the estimated Albedo of this dataset. The surface top left white patch is not uniform as it should be. This might be caused by the number of images in the dataset.

The dataset consists of 121 images, this entails that when estimating $g(x, y)$ from equation 3, we try to model an 121-dimensional input using three variables. In this case the model might not contain enough complexity (parameters) to successfully model the input. A solution to this problem might be to use less viewpoints such that each patch of the object surface has been illuminated.

The result from this method can be seen in figure 7a. This estimate is based on 45 images using an equal amount of viewpoints from each corner. It can be seen that the reduction in the amount of images did not result in a significant change in the estimated Albedo with respect to figure 7b. This suggests that for the estimating the Albedo not all 121 images are necessary. However, it does not confirm that a decrease in the amount of images will improve the Albedo.

3 Colour Spaces

The RGB colour model is often used as basis for cameras and photography. This is because it is similar to the manner in which humans process color information. The human perception system contains cone shaped receptors. There are roughly three different types of these cone shaped receptors, each of which process long, middle, and short wavelengths corresponding to Red, Green, and Blue. Digital cameras often make use of a similar technique called a Bayer Filter. This is an array, in which each element filters certain incoming wavelengths, also corresponding to red, green, and blue.

However, there do exist different colorspaces than the RGB colorspace, each of which has its own advantages with respect to the RGB colorspace. In figures 9b to 9f, which are placed in appendix A, color conversions of five different colorspaces of an original RGB picture (figure 9a) are shown.

In figure 9b, the original image in the opponent colorspace is shown. The opponent colorspace as described in Anwer et al. (2011), is based on the human visual system. This colorspace consists of three channels, one in which red opposes green, one of which yellow opposes blue, and one which represents the luminosity. This colorspace should mimick a second layer in the human retina, with the first being the different receptors. The different channels can

be described in terms of RGB channels as follows:

$$\text{Red - Green} = \frac{(R-G)}{\sqrt{2}} \tag{8}$$

$$\text{Yellow - Blue} = \frac{(R+G)-B}{\sqrt{6}} \tag{9}$$

$$\text{Luminosity} = \frac{R+G+B}{\sqrt{3}} \tag{10}$$

This opponent colorspace can be used as second order color features in computer vision algorithms such as human detection. In Anwer et al. (2011) it is shown that the use of opponent colors can lead to better performance on human detection with respect to the use of the RGB colorspace. The intuition behind the improvement is that the opponnent colorspace more clearly presents the differences between colors. The visualization of these channels can be seen in figure 9b. The Red - Green channel clearly shows high intensity for the red peppers and low intensity for the green peppers. The Yellow - Blue channel shows high intensity for the yellow peppers and low intensity for the blue peppers.

In figure 9c, the converted image and its channels in the normalized RGB colorspace can be seen. This colorspace is calculated by dividing the intensity of each channel of each pixel by the sum of the channels of the pixel. As can be seen in 9c, this results in a less detailed version of its original. It can be seen that the high intensity colors in the original image are affected the most. The three different color channels show high activation on color patches with the corresponding color. Advantages of the normalized RGB colorspace over the RGB colorspace are the fact that it reduces the amount of noise or distortions in pictures.

The converted image and its channels in the HSV colorspace are depicted in figure 9d. The HSV colorspaces uses the Hue, Saturation, and Value as its channels. The Hue channel shows the dominant wavelength of the spectral power distribution of the signal. As a result, light colors such as white and yellow have a low intensity in the Hue channel. This is because their spectral power distribution has high intensity for a big range of wavelengths, but no clear dominant wavelength. Colors such as red do have a dominant wavelength and therefore have high intensity in this channel. The saturation channel shows the purity of a color, thus light colors again have a lower saturation value. The value channel shows the intensity of the maximum RGB channel at each pixel. An advantage of the HSV colorspace over RGB, is the fact that color information and intensity are represented in different channels. As can be seen in the Hue channel, the shadowed parts of the tablecloth do not show a difference with the other parts in Hue value.

The results of converting the image to the YCbCr colorspace are depicted in figure 9e. The Y channel depicts a luminous representation of the image, whereas CB represents primarily blue and yellow colors and CR represents primarily red and green colors. This is also visible in figure 9e. The CB channel shows the highest intensity for the tablecloth (which is blue-ish in the original) and extremely low intensity for the yellow peppers. The CR channel on the other hand shows high intensity for red peppers and extremely low intensity for green peppers. Again, an advantage over RGB is that this colorspace separates color information from intensity.

At last, the results from the different RGB to Grayscale conversion algorithms are shown in figure 9f. The lightness image shows the use of the lightness algorithm, which averages the most and least intense channel at each pixel. The Average algorithm computes the average across the three channels for each pixel to compute the grayscale. The Luminosity algorithm computes the weighted average of the three channels using [0.21+0.72+0.07] as weights for red, green, and blue respectively. The Opencv algorithm[2] also computes a weighted average, but with [0.299+0.587+0.14] as weights for red, green, and blue respectively. It is visible that the lightness algorithm shows the largest differences with respect to the other algorithms. Especially the red pepper in front has more intensity than when using the other algorithms. The difference between the use of the average and weighted average can be seen in the yellow pepper in the middle, it has much larger intensity in case of the luminosity and Opencv algorithms. The difference in weights for the weighted average do not seem to result in big differences between the Luminous and Opencv images. An advantage of the grayscale colorspace over the RGB colorspace is that it is only one channel and thus requires less storage space on devices, bandwith on the internet, or computations in algorithms to process. However, it often contains enough details for humans and algorithms to interpret the images.

Another colorspace described in Smith (1978), is the HSL colorspace. Just as the HSV colorspace, it uses the Hue and saturation of each pixel. However, instead of the value, it uses the lightness, thus the average of the maximum and minimum channel of each pixel. As is the case with HSV, color hue shows less variation and distortion in its values than color. Thus an Hue histogram can build from the image which can be used to compare images with little computations. This can be used in computationally scarce settings such as robot soccer as proposed by Tsai and Tseng (2012).

4 Intrinsic Image Decomposition

Humans visual system is evolved to infer and reason about the shape, color, texture, illumination, etc. of the visual imagery we see in real in life. We are able to do such even for the low resource images. The idea behind intrinsic image decomposition is to separate these visual channels which compose image information and to reason about the surface/scene which makes the image. This also augments computer vision system for more accurate visual understanding. One major decomposition proposed in Barrow and Tenenbaum (1978) is extracting shading (illumination) $S(\vec{X})$ and albedo $R(\vec{X})$ information from the image as follows:

$$I(\vec{X}) = S(\vec{X}) X R(\vec{X}) \tag{11}$$

This decomposition and the reconstruction is illustated in figure 10, which is placed in appendix A.

Apart from these albedo and shading components, an image can also be decomposed into other descriptive image components, for example, figure 11, Janner et al. (2017) shows the original image decomposed into shape (depth) and lighting components as well along with the shading and albedo components, also figure 12, Barron and Malik (2012). However, intrinsic image decomposition

[2]https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.htmlcvtcolor

is a challenging task, and is described in the literature as an "ill-posed problem." This inverse problem is almost underconstrained, as there is no one valid answer, figure 11. Since there are many alternative ways to decompose the problem, algorithms work to find the most probable ones. As such, some works assign informative priors to shape, albedo, and illumination conditions to infer the probability of obtained intrinsic images Barrow and Tenenbaum (1978), while other works train and test their algorithms on ground truth intrinsic images datasets. Given the challenges associated with the latter approach (time, cost, along with the added complexity of this underconstrained problem on real world images), datasets of synthetic images are proposed in the literature Grosse et al. (2009), Bell et al. (2014).

A great application of intrinsic image decomposition is Recoloring, and it also demonstrates the confounding effects of illumination conditions. In figure 13, the original image is uniform (its albedo components have values R:0.72, G:0.55, B:0.55 for every pixel of the ball). We recolor it with zeroing out R and B channels of RGB and keeping only the G component (maximum value, pure green) in the albedo intrinsic image. We reconstruct the final image using equation 11. We expect to see the uniformity across the G channel of the final recolored image. However, the reconstructed image (figure 13) is not uniform and does not show pure colors as shown in figure 14. This can be greatly explained by how illumination affects the perceived image, and also an interplay between lighting conditions and the object to be captured. That is why intrinsic image decomposition is such an appealing research area because we can separate out the property of an image we are interested in without considering other confounding effects.

5 Colour Constancy

In this section we will implement the Gray-World algorithm and show the results. The Gray-World algorithm tries to achieve color constancy by assuming that a well color balanced photo, the average color is gray. A picture of an object under a colored light, yellow for example, however will show yellowish colors. To correct this, the color channels of the pictures are adjusted such that each has an average gray color.

Adjusting the colors of each pixel is done using the *von Kries* model. We have to adjust each pixel $I(x, y)$ in the image, sucht that the average color per channel is grey. To achieve this, we first calculate the average color per channel: μ_r, μ_g, μ_b. The average color of the resulting picture R per channel is: $R_r, R_g, R_b = 128, 128, 128$. In the *von Kries* model, each pixel is updated by applying a diagonal matrix with the color channel ratios of the resulting picture and the input picture to the pixel values. Thus at each pixel:

$$R(x, y) = \begin{bmatrix} R_r \\ R_g \\ R_b \end{bmatrix} = \begin{bmatrix} \frac{R_r}{\mu_r} & 0 & 0 \\ 0 & \frac{R_g}{\mu_g} & 0 \\ 0 & 0 & \frac{R_b}{\mu_b} \end{bmatrix} \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix} \tag{12}$$

In which I_r, I_g, I_b correspond to the RGB channels of the input channel at pixel x, y. Results of this algorithm can be seen in figure 15 in appendix A. Where in the original picture, the colors seem reddish, this is not the case in the Grey-World picture.

The result in figure 15 is an example of a case where the Gray-World assumption holds. However, this might not always be the case. An underwater picture of the ocean for example will be largely blue, also under natural light. Adjusting the colors channels such that the average will be blue, might result in an unnatural representation of the color blue.

Another method used to achieve color constancy is the Scale-By-Max method. This method assumes that there should be at least one white patch in the image. If there is no white patch, each pixel will be adjusted with a constant such that the maximum intensity pixel is white.

6 Conclusion

In this report multiple techniques in the domain of image formation were applied and discussed. In the first section we applied the Photometric Stereo algorithm and discussed the results. It was shown that the both the Albedo and the normal vectors could be more reliably estimated using 25 images than 5 images. For the reconstruction of the surface of the sphere however, was most precise when using the 5 images dataset with the average of the Column-major and Row-major algorithms.

Thereafter, by applying the Photometric Stereo algorithm on the MonkeyGray dataset, it was clear that the algorithm does not perform as well on different datasets. For example, the Albedo of this dataset contained more errors than on the Sphere datasets.

Thereafter in the second section, different colorspaces were shown and discussed. It was evident that there exist multiple different colorspaces with certain advantages over RGB. Therefore, depending on the use case, representing images in a different colorspace should always be considered.

Furthermore, we also see in action a very interesting idea of intrinsic image decomposition. This challenging yet important technique provides a mechanism to study and understand images independent of other imaging conditions which affect the obtained image. We reconstruct an image using albedo and shading intrinsic components. We also saw one application in the form of recoloring, where we can extract and manipulate color information without any effect from illumination. This gives more information and reliability. We also recolor an image, and saw how the shading attributes affect the reconstructed image, thereby illustrating one confounding affect in visual understanding in absence of informative intrinsic images.

At last, the Gray-World algorithm was discussed and applied to an image to show its workings. It was concluded that in certain cases the assumption that the average color should be gray holds. However, this will not always be the case. Therefore, depending on the use case, different color constancy algorithms should be considered.

References

Anwer, R. M., Vázquez, D., and López, A. M. (2011). Opponent colors for human detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 363–370. Springer.

Barron, J. T. and Malik, J. (2012). Shape, albedo, and illumination from a single image of an unknown object. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 334–341. IEEE.

Barrow, H. and Tenenbaum, J. M. (1978). Recovering intrinsic scene characteristics from images.

Bell, S., Bala, K., and Snavely, N. (2014). Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4).

Forsyth, D. A. and Ponce, J. (2012). *Computer Vision - A Modern Approach, Second Edition.* Pitman.

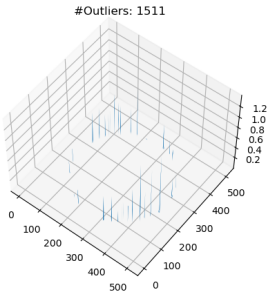
Grosse, R., Johnson, M. K., Adelson, E. H., and Freeman, W. T. (2009). Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2335–2342.

Janner, M., Wu, J., Kulkarni, T. D., Yildirim, I., and Tenenbaum, J. (2017). Self-supervised intrinsic image decomposition. In *Advances in Neural Information Processing Systems*, pages 5936–5946.

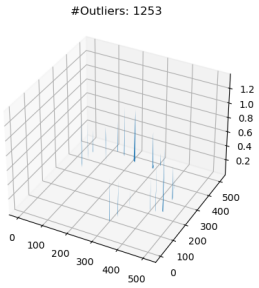
Smith, A. R. (1978). Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12(3):12–19.

Tsai, S.-H. and Tseng, Y.-H. (2012). A novel color detection method based on hsl color space for robotic soccer competition. *Computers & Mathematics with Applications*, 64(5):1291–1300.

A Figures

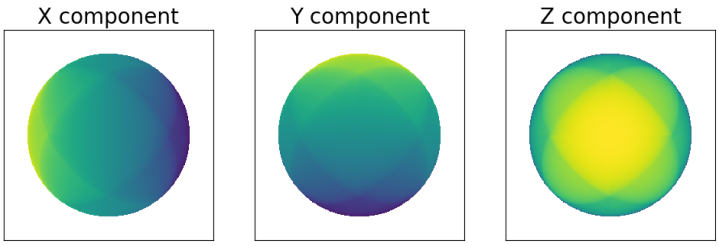


(a) SE for SphereGray5 without shadow trick

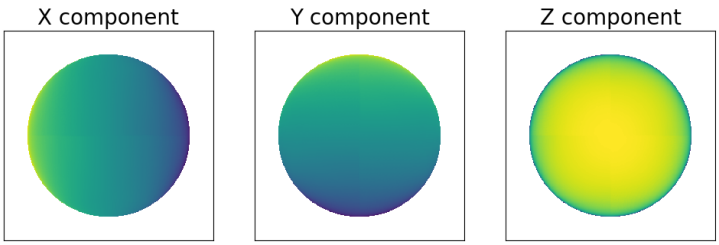


(b) SE for SphereGray25 without shadow trick

Figure 2: Squared error (SE)

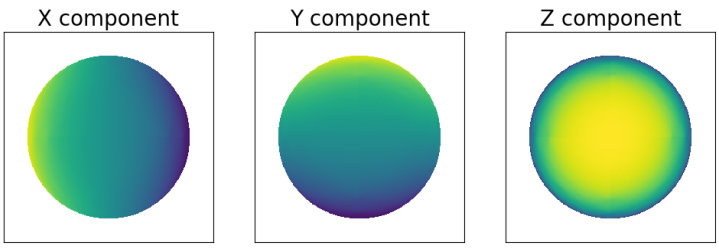


(a) SphereGray5 without shadow trick

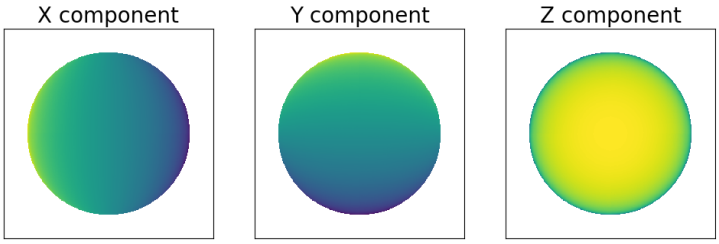


(b) SphereGray5 with shadowtrick

Figure 3: The three components of the estimated normals of the ShadowGray5 dataset.

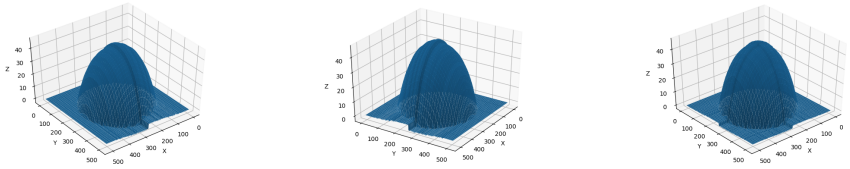


(a) SphereGray25 without shadow trick



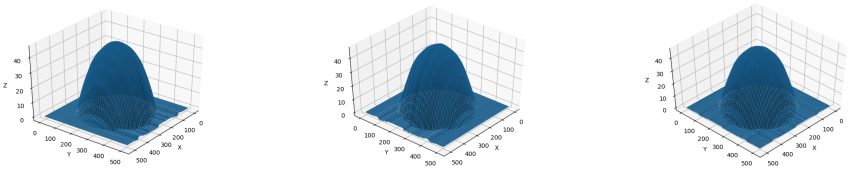
(b) SphereGray25 with shadowtrick

Figure 4: The three components of the estimated normals of the SphereGray25 dataset.



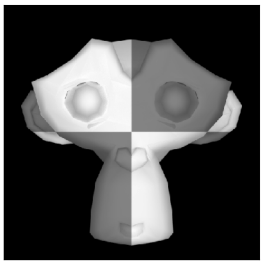
(a) Column-major order (b) Row-major order (c) Average

Figure 5: Results of the surface construction algorithm for the three different methods using the SphereGray25 dataset.

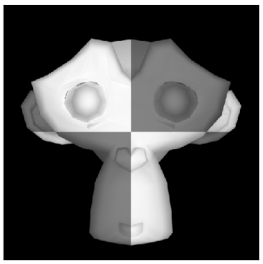


(a) Column-major order (b) Row-major order (c) Average

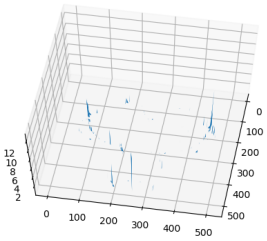
Figure 6: Results of the surface construction algorithm for the three different methods using the SphereGray5 dataset.



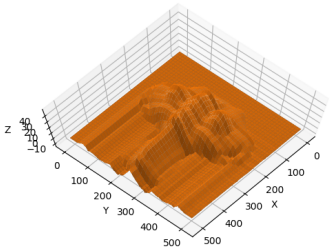
(a) Estimated Albedo using the small monkey dataset.



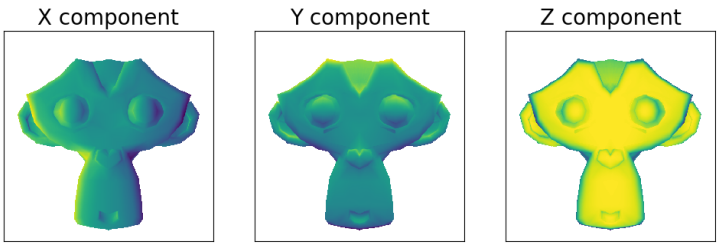
(b) Estimated Albedo using the large monkey dataset.



(a) Squared error of partial second derivatives



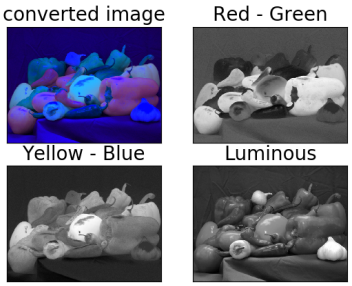
(b) constructed surface using the average of Column-major and Row-major



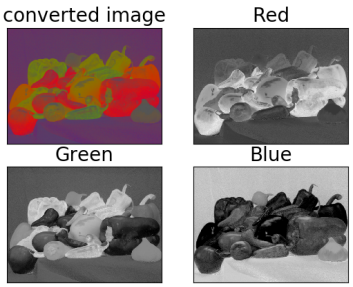
(c) Components of normal vectors

Figure 8: Results of the surface construction algorithm for the three different methods using the MonkeyGray dataset.

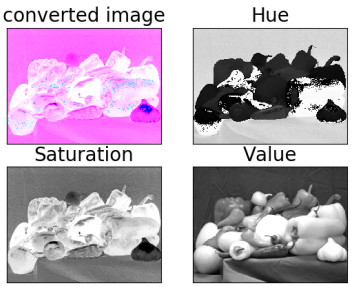
(a) Original image



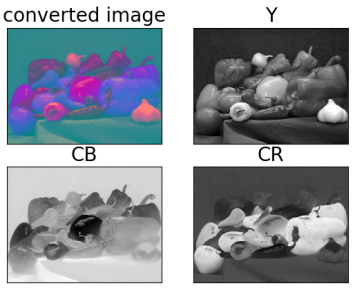
(b) Opponent colorspace



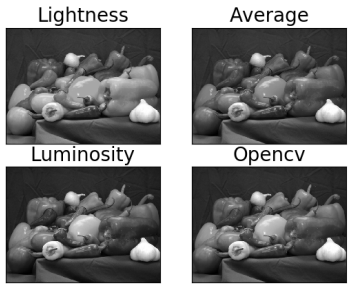
(c) Normalized RGB colorspace



(d) HSV colorspace



(e) YCbCr colorspace



(f) Grayscale

Figure 9: The converted image and its color channel in the four different colorspaces. Figure 9f show the results of the four different RGB to Grayscale algorithms.

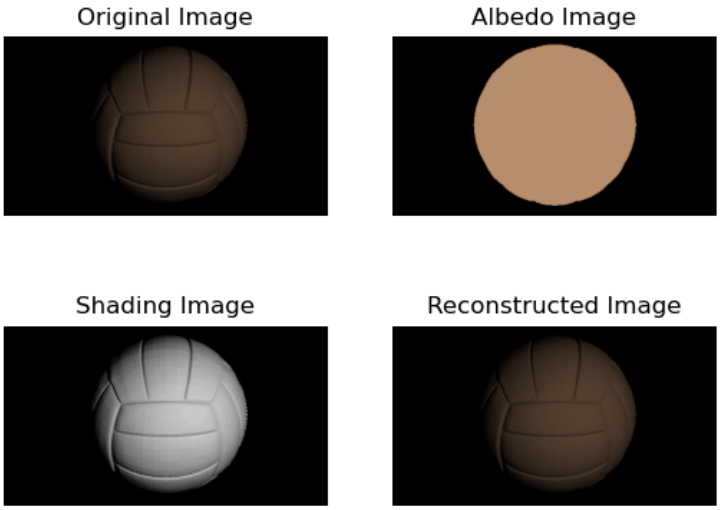


Figure 10: [Intrinsic Image Decomposition] Reconstructed Image from the Albedo and Shading Components

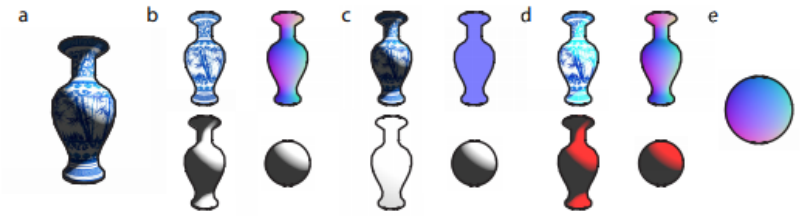


Figure 11: The problem of intrinsic image reconstruction is underconstrained. Images in (b), (c), (d) and (e) are all intrinsic images (reflectance, shape, shading, and lighting) of the vase in (a) as they all combine together to reconstruct the vase. **Image taken from the original paper:** Janner et al. (2017)

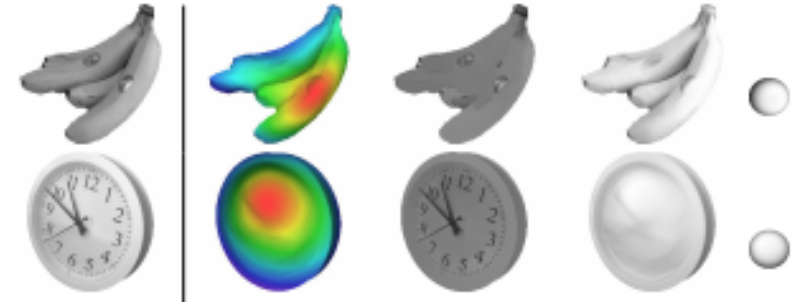


Figure 12: [Intrinsic Image Decomposition] Depth, Albedo, Shading, and Illumination (left to right) intrinsic components extracted from each of the object image: Bananas and Clock. **Taken from original paper** Barron and Malik (2012)

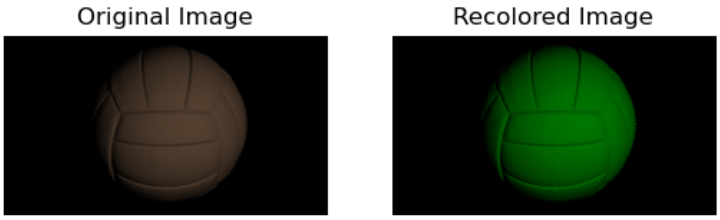


Figure 13: [Intrinsic Image Decomposition] Recolored Image and the Original Image

Figure 14: [Intrinsic Image Decomposition] R, G, B channels of the Recolored Image

Figure 15: Result (right) of applying the Grey-World algorithm on the original image (left)