Image Reflection Removal in Modern Photography

Team JSTV
Jeffrey Effendy (A0080527H)
Steven Kester Yuwono (A0080415N)
Tan Tack Poh (A0098802X)
Vikas Kumar (A0148475U)

1 Initial Attempt at Problem Formulation

Given a set of images of the exact same background scene with different artificial foreground reflections (using digital layering technique), we aim to remove the reflections in the images and recover the background scene.

We might also explore the possibility to create our own dataset to mimic the situation in real life where the pictures with different reflection images cannot possibly have the exact same background scene on the pixel level.

The input is a set of images (I) of the exact same background image (B), but different foreground/reflection images (R).

The output is a resultant image ($I_{resultant}$) [by maximizing the opacity of background image (B) and minimizing that of foreground/reflection image (R)].

The properties of all of the images (I) are that the background image (B) is the same while the reflection images (R) differ.

The constraints are:

- The colours of the resultant image ($I_{resultant}$) should closely resemble to the colours of the background component of given images (I).
- The resultant image ($I_{resultant}$) should contain only the background image (B) with least traces of reflection image (R).

We define the set of given images as raw images (I). A raw image (I) can be considered as a mixture of a background image (B) (i.e., the actual image that we would like to retrieve) and the reflection image from the glass surface (R).

For simplicity, we assume that the two component images B and R are blended by multiplying a weight W_b and W_r which represents the opacity of the background and reflection image respectively and then sum the values of each pixel to result in the raw image (I). We only have information on I, and the values of W_b, W_r, B, and R are unknown.

For each pixel in the image:

$$I = W_b B + W_r R \quad \text{where } W_b + W_r = 1.0$$

Assuming that the background image (B) is identical for all the given raw images, for each of the raw images (I_j), only the reflection images component differ (R_j) where j denotes the jth given raw images.

$$I_j = W_b B + W_r R_j$$

Using reflection removal techniques, our aim is to produce an image ($I_{resultant}$) with the maximum opacity of the background image component B and the minimum opacity of reflection R. Ideally, we would like to remove the R component completely (i.e., $W_b = 1.0$ and $W_r = 0.0$ in the equation below).

$$I_{resultant} = W_b B + W_r R$$

If W_r is reduced, the variance of colors in the resultant image ($I_{resultant}$) will be strictly smaller than the variance of colors in any of the given images (I).

$$var[I_{resultant}] < var[I_j] \quad \text{for all} \quad j$$

2 Revised Problem Formulation

2.1 Introduction

Photographers sometimes face difficult situations of taking photographs behind a reflective surface. Images captured in front of a reflective surface often contain both the transmitted background scene and the reflection of the objects in front of the surface panel (e.g., glass). There are different ways a photographer can employ to mitigate the reflection issue. For example: by adjusting light/position or by adding polarizers to their camera. However, sometimes it is difficult to avoid the reflection problem using the aforementioned techniques. In this project, we propose to tackle the image reflection removal problem by exploring different methods. We will study a related work by Shih et al. (2015), Yu and Brown (2013), Lin et al. (2009) and Hu (2016) in more detail.

2.2 Objective

Given a set of images of identical background scene with different foreground reflections, we aim to remove the reflections in the images and recover the background scene.

Some examples are shown below:

(a) Image with reflection 1 (b) Image with reflection 2 (c) Image with no reflection

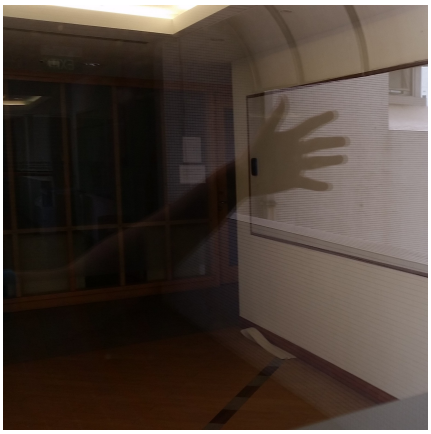
Figure 1: Sample images in the dataset

In other words, given image 1a and image 1b, our objective is to produce image 1c. We might also explore the possibility to create our own dataset and taking real photographs to mimic the situation in real life.

2.3 Problem Formulation

The input is a set of images (I) of the exact same background image (B), but different foreground/reflection images (F). The output is a resultant image (R) by minimizing that of foreground/reflection image (F). Ideally, the resultant image should be identical to the background image or equivalent to $R = B$.

We observe that photographs or images taken (even from the same angle) are not always perfectly aligned. This will add to the difficulty of tackling the problem of reflection removal. Figure 2 below shows that the background image that we are trying to capture is not aligned at the same spot.



(a) Image with reflection 1

(b) Image with reflection 2

Figure 2: Unaligned images

Since dealing with unaligned images is a more complex problem, in this project, we have simplified the problem to only handle a set of aligned images. Aligned images with different reflections can be obtained by various methods such as capturing the images with a fixed camera (e.g., using tripod) and using different objects in front of the reflecting glass for each image. A common scenario of this problem is when someone is standing outside a shop, looking at the content of the shop behind a glass panel, with pedestrians walking by in front of it.

(a) Image with reflection 1

(b) Image with reflection 2

Figure 3: Aligned images

To tackle image reflection removal problem we model the light path arriving at the camera. The following diagram shows the output color (I) as captured by the camera as a result of reflection on a semi transparent medium.

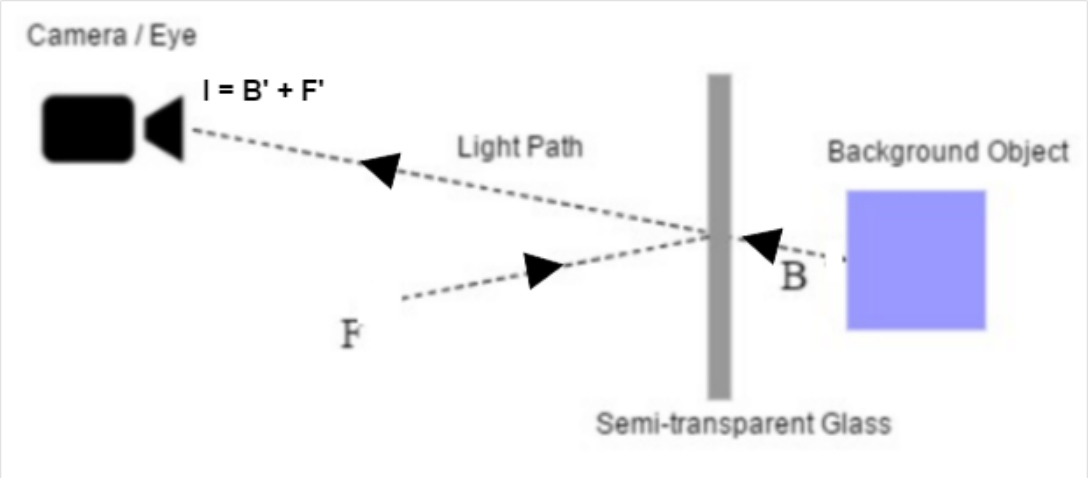


Figure 4: Light Path arriving at a Camera from an Object through a Translucent Glass

Using physics of light transmission and reflection, we can define that the light captured by the camera (I) is a linear combination of accumulated reflection (F) and the object (B) assuming the object is opaque, which is expressed as:

$$I = k_b B + k_f F \tag{1}$$

where k_b is the transmission coefficient and k_f is the reflection coefficient.

By law of conservation of energy, with the assumption of no heat loss and no absorption by the medium, the lights are perfectly distributed between foreground/reflection image (F) and background image (B) without any energy loss.

In our definition, we can simplify further our problem to assume the coefficient to be equal throughout the sample images because of the same semi-transparent medium used and the same lighting used. Thus, now we have the following:

$$I = B' + F' \tag{2}$$

Given the problem scope, we observe that all of the images have exactly the same background images (B'). In other words, for all images, B will be constant for all input images. Hence, given a set of n images, each with $(x \times y)$ pixels, equation 1 can be re-written as :

$$I_i(x, y) = B'(x, y) + F'_i(x, y) \quad \text{for } i = 1...n \tag{3}$$

2.4 Problem Definition

Given spatially aligned images I_i where $i = 1...n$, our objective is to compute a resultant image $R \approx B'$ such that

$$I_i(x, y) = B'(x, y) + F'_i(x, y) \ , \ \text{for } i = 1...n \tag{4}$$

where there are enough of $F'_i(x, y)$ are zero, so that equation 4 is computable.

3 First Attempt at Problem Solving

In our initial attempt we try to approximate the value of the resultant image $\hat{\mathbf{R}}$ by averaging each pixels of the given images $\mathbf{I}$. The idea is to read each pixels of the same index (location) of all the images and add up in iterative fashion to obtain the average. The average method ensures that the pixel value will fall inclusively between 0 and 255. For each pixel in the red channel:

$$\hat{\mathbf{R}}_r = \frac{(\mathbf{I}_{r_1} + \mathbf{I}_{r_2} + ... + \mathbf{I}_{r_n})}{n} \tag{5}$$

$$\therefore min(I_1(x,y), ..., I_n(x,y)) \leq \hat{R}(x,y) \leq max(I_1(x,y), ..., I_n(x,y)) \tag{6}$$

For the subsequent definition method, we only consider for the red channel in the color and can be applied in the same way to blue and green channel.

In our approach, we represent an image by arranging the pixels into an array of RGB channel. Since we are only considering the red channel, the matrix $\mathbf{I}$, $\mathbf{B}$, $\mathbf{F}$, $\mathbf{K}_b$ and $\mathbf{K}_f$ each has dimension of 2 and same size, which is (x, y), the resolution of the sample image. We also define averaging function g that maps sample images $\mathbf{I}$ as input to our resultant matrix $\hat{\mathbf{R}}$.

For n number of images in the training data set:

$$\begin{aligned} \hat{\mathbf{R}}_r &= g(\mathbf{I}_{r_1}, \mathbf{I}_{r_2}, ..., \mathbf{I}_{r_n}) \\ \hat{\mathbf{R}}_r &= \frac{\mathbf{I}_{r_1}, \mathbf{I}_{r_2}, ..., \mathbf{I}_{r_n}}{n} \\ \hat{\mathbf{R}}_r &= \frac{1}{n} \sum_{j=1}^{n} \mathbf{I}_{r_j} \end{aligned} \tag{7}$$

$$\begin{aligned} \hat{\mathbf{R}}_r &= \frac{1}{n} \sum_{j=1}^{n} (\mathbf{K}_{b_j} \circ \mathbf{B}_r + \mathbf{K}_{f_j} \circ \mathbf{F}_{r_j}) \\ \hat{\mathbf{R}}_r &= \mathbf{B}_r \circ \frac{1}{n} \sum_{j=1}^{n} \mathbf{K}_{b_j} + \frac{1}{n} \sum_{j=1}^{n} (\mathbf{K}_{f_j} \circ \mathbf{F}_{r_j}) \end{aligned} \tag{8}$$

where $\circ$ denotes Hadamard product

From equation 8, if n is large, the value within cell $\mathbf{K}_{f_j}$ is small and the non reflective part in the sample images are spread throughout the image, the average will be $\hat{\mathbf{R}}_r = g(\mathbf{I}_{r_1}, \mathbf{I}_{r_2}, ..., \mathbf{I}_{r_n}) \approx \mathbf{B}$ if given conditions are met.

4 Revised Algorithm

4.1 Averaging Method

In our initial attempt we try to approximate the value of the resultant image $\hat{\mathbf{R}}$ by averaging each pixels of the given images $\mathbf{I}$. The average method ensures that the pixel value will fall inclusively between 0 and 255. For each pixel in the image:

$$\hat{\mathbf{R}} = \frac{(\mathbf{I}_1 + \mathbf{I}_2 + ... + \mathbf{I}_n)}{n} \tag{9}$$

$$\therefore min(I_1(x,y), ..., I_n(x,y)) \leq \hat{R}(x,y) \leq max(I_1(x,y), ..., I_n(x,y)) \tag{10}$$

In our approach, we represent an image by arranging the pixels into a three dimensional matrix. The matrix $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{F}$ each has dimension of 3 which is $(x, y, 3)$, the resolution of the sample image and the RGB values.

For n number of images in the training data set:

$$\hat{\mathbf{R}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_i \tag{11}$$

Since the semi-transmission medium is always the same across n images, ideally the reflection and transmission coefficient are the same for all n images as stated in our problem formulation. From equation 11, we will have:

$$\hat{\mathbf{R}} = \mathbf{B} + \left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{F}_i)\right) \tag{12}$$

From equation 12, it is shown that the resultant image is an approximation of the background image, $\hat{\mathbf{R}} \approx \mathbf{B}$ if matrix F is sparse enough and the sample is large enough because $\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{F}_i)\right)$ will be very small. Since matrix $\mathbf{B}$ is constant, there would not be any lost information within matrix $\mathbf{B}$ for every averaging iteration.

Algorithm 1: Averaging Method

Input : Input images, I_1, I_2, ..., I_n

1 $\hat{\mathbf{R}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{I}_i)$

Output: A resultant image, $\hat{\mathbf{R}}$

Evaluation metrics: To evaluate the averaging approach, Mean Squared Error (MSE) of the resultant image $\hat{\mathbf{R}}$ and the ground truth will be calculated which is shown in Equation 20.

Limitation: Our averaging approach has some limitations. The resultant image $\hat{\mathbf{R}}$, will always contains the noise of the foreground reflections in any of the training images for that particular pixel. It is impossible obtain an optimal resultant image where $\hat{\mathbf{R}} = \mathbf{B}$. The weakness we have just mentioned can be clearly seen when the opacity of the foreground image is high, which will contribute a significant amount of error in the resultant image, as shown by our experiments result in Figure 10.

4.2 Robust PCA

Referring to our problem formulation, we have:

$$\mathbf{I} = \mathbf{B'} + \mathbf{F'} \tag{13}$$

If we represent $\mathbf{I}$ with as vertically stacked flattened image, we will have a matrix of size $(n, r \times c \times 3)$ where n is the number of samples and r and c are the height and width of the image pixels respectively.

In our problem definition, we capture the image such that the background is always identical, which results in stacked **B'** being low rank matrix. This leads to our objective is to find the low rank matrix **B'**. In this RPCA section, from this point onward, we will define the matrix of flattened images as M, the low-rank matrix (background) as L, and the sparse matrix (reflection) as S. To find the low matrix, we will use Robust Principal Component Analysis (PCA) by minimizing the difference of M and L, which is expressed as following constrained optimization:

$$\begin{gathered}\min_{L,S} \|S\|_F, \text{ subject to } M = L + S \text{ and } rank(L) \ll min(n, r \times c \times 3) \\ \text{where } \|.\|_F \text{ is a Frobenius norm.}\end{gathered} \tag{14}$$

In our problem definition, we have defined S to be sparse. Under the definition, it is shown by Candès et al. (2009) that the low rank matrix can be extracted by solving the following convex Principal Component Pursuit (PCP) problem:

$$\begin{gathered}\min_{L,S} \|L\|_\star + \lambda\|S\|_1, \text{ subject to } M = L + S \\ \text{where } \|.\|_\star \text{ is a nuclear norm and } \|.\|_1 \text{ is a natural norm.}\end{gathered} \tag{15}$$

Introduced by Wright et al. (2009) and Lin et al. (2009), we will solve the convex PCP problem using Augmented Lagrange Multiplier (ALM) method, which operates on the following augmented Lagrangian function:

$$\begin{gathered}l(L, S, Y, \mu) = \min_{L,S} \|L\|_* + \lambda\|S\|_1 + \langle Y, M - L - S\rangle + \frac{\mu}{2}\|M - L - S\|_F^2 \\ \text{where } Y \text{ is the Lagrange multipliers and } \mu \text{ is the penalty parameter}\end{gathered} \tag{16}$$

In our solution, we will be utilising a method by Xiaoming and Yang (2009), called Alternating Directions Method (ADM), which is a variation of the classic ALM method. With the method, we separate the calculation S and L to the following as used by Candès et al. (2009):

$$\begin{aligned}\min_L l(L, S, Y) &= \mathcal{D}_{\mu^{-1}}(M - S - \mu^{-1}Y) \\ \min_S l(L, S, Y) &= \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y)\end{aligned} \tag{17}$$

Where $\mathcal{S}_\tau[x]$ is defined as a shrinkage thresholding:

$$\mathcal{S}_\tau[x] = sgn(x)max(|x| - \tau, 0) \tag{18}$$

And $\mathcal{D}_\tau(X)$ denote the singular value thresholding operator:

$$\begin{gathered}\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^* \\ \text{where } U\Sigma V^* \text{ is any singular value decomposition}\end{gathered} \tag{19}$$

Thus, the strategy is to minimize l with respect to L while fixing S, and then minimize l with respect to S while fixing L, and then update the Lagrange multiplier matrix Y. It is shown in Algorithm 2 below:

Algorithm 2: Principal Component Pursuit by Alternating Directions Method

Input : Vertically stacked images, M

initialize: $S_0 = Y_0 = 0$, $\mu_0 > 0$;

while *not converged* **do**

  $L_{k+1} = \mathcal{D}_{\mu^{-1}}(M - S_k - \mu^{-1}Y_k)$;

  $S_{k+1} = \mathcal{S}_{\lambda\mu^{-1}}(M - L_{k+1} - \mu^{-1}Y_k)$;

  $Y_{k+1} = Y_k + \mu(M - L_{k+1} - S_{k+1})$;

end

Output: Low-rank matrix (background) and sparse matrix (reflection), L and S

In our algorithm, we initialize $\mu = rc/(4\|M\|_1)$ and $\lambda = 1/\sqrt{max(r, c)}$ following the paper, and we define convergence when $\|M - L - S\|_F \leq \delta\|M\|_F$ where $\delta = 10^{-7}$. We terminate the algorithm when convergence condition is satisfied, or when the number of iterations exceed the specified maximum iteration.

Algorithm 2, which is ADM, is a special case of a general ALM algorithm. The convergence of this algorithm has been well studied in Kontogiorgis and Meyer (1998) and Lions and Mercier (1979).

Evaluation metrics: Using Algorithm 2 above, we will obtain matrix L, and S which represents a vertically stacked flattened low-rank and sparse images. The resultant image is not unique, therefore, we average the Mean Squared Error (MSE) (shown in Equation 20) for each of the low-rank resultant images against the ground truth to measure the algorithm performance. This evaluation approach is also adopted by Leow et al. (2013).

Limitation: RPCA algorithm performs very well when the sparse matrix S is very sparse (i.e., most of the elements are 0). When the size of the foreground **F'** is large, the RPCA algorithm will not produce optimum result as S is not sparse enough, and hence the rank of L will not be optimum (in ideal case $rank = 1$). This limitation is observed in our experiments shown in the next section where the size of the foreground is large.

5 Test and Discussions

We are using the following evaluation metrics to measure the performance of our algorithm:

- **Mean Squared Error** (MSE): The averaged squared euclidean distance between the estimated and the real result. (In this case, each pixel is represented by 3 dimensional vector, consisting of red, green, and blue component). The mean squared error is calculated as follows:

$$\mathrm{E} = \frac{1}{3rc} \sum_{i=1}^{r} \sum_{j=1}^{c} ||R(i,j) - \hat{R}(i,j)||^2 \tag{20}$$

  where r, c are the width and the height of the image respectively (in pixels). R is the image without reflection (ground truth) and $\hat{R}$ is the computed resultant image.

- **Time**: The time taken for the program to run given n training images

5.1 Default Low Resolution Data Set

The first data set is using the provided low resolution images[1]. The dimension of the image is 200 × 150 pixels (width × height). The data set contains 20 images with the same background but different foreground (reflection), and one reference image (the ground truth).

(a) Image with reflection 1 (b) Image with reflection 2 (c) Image with no reflection

Figure 5: Sample images in the default low resolution data set

5.1.1 Results

The result of our experiments on this data set are shown in the figure 6 below:

(a) Averaging with 5 sample images (b) Averaging with 20 sample images (c) RPCA with 5 sample images (d) RPCA with 20 sample images

Figure 6: Resultant images on low-res data set using various numbers of training images

[1] https://www.comp.nus.edu.sg/čs5240/data/reflection-lowres.zip

5.2 Default High Resolution Data Set

The first data set is using the provided high resolution images[2]. The dimension of the image is 1597 × 901 pixels (width × height). The data set contains 20 images with the same background but different foreground (reflection), and one reference image (the ground truth).

(a) Image with reflection 1

(b) Image with reflection 2

(c) Image with reflection 3

(d) Image with no reflection

Figure 7: Sample images in the default high resolution data set

5.2.1 Results

The result of our experiments on this data set are shown in the figure 8 below:

(a) Averaging with 5 training images

(b) Averaging with 20 training images

(c) RPCA with 5 training images

(d) RPCA with 20 training images

Figure 8: Resultant images on high-res data set using various numbers of training images

[2]https://www.comp.nus.edu.sg/čs5240/data/reflection-highres.zip

5.3 Custom Data Sets

After observing the given data sets thoroughly, we realised that the opacity of the foreground of the images are very low, therefore the averaging method will produce a good result. We would like to test the limit of the algorithm by creating more difficult test cases.

The background image was taken using a digital camera, and then synthetic foreground (reflection) images are added on top of the image with various opacity. Synthetic images approach was chosen because we would like to have a ground truth and be able to compute the error of our algorithms accurately.

10 foreground images were captured and then juxtaposed and having its opacity controlled by using Adobe Photoshop.

There are 10 sets in this custom data set:

- High resolution images (1500×1260) with 10%, 20%, 30%, 40% and 50% foreground opacity.
- Low resolution images (150×126) with 10%, 20%, 30%, 40% and 50% foreground opacity.

For each of the data set, there are another 5 sub data sets where the size of the foreground is varied: 10%, 25%, 50%, 75%, 100% (with respect to the image size). Therefore, in total, there are $10 \times 5 = 50$ custom data sets. For each of the sub types, there are 10 images with different foreground reflections, and one reference image (ground truth).

(a) Sample image with 10%-α size-100%

(b) Sample image with 20%-α size-50%

(c) Sample image with 50%-α size-25%

(d) Original custom image without reflection

Figure 9: Custom image data set

5.3.1 Results

The result of our experiments on this data set are shown in the figure 10 below:

(a) Average method on 10%-α size-100%

(b) RPCA method on 10%-α size-100%

(c) Average method on 30%-α size-50%

(d) RPCA method on 30%-α size-50%

(e) Average method on 50%-α size-75%

(f) RPCA method on 50%-α size-75%

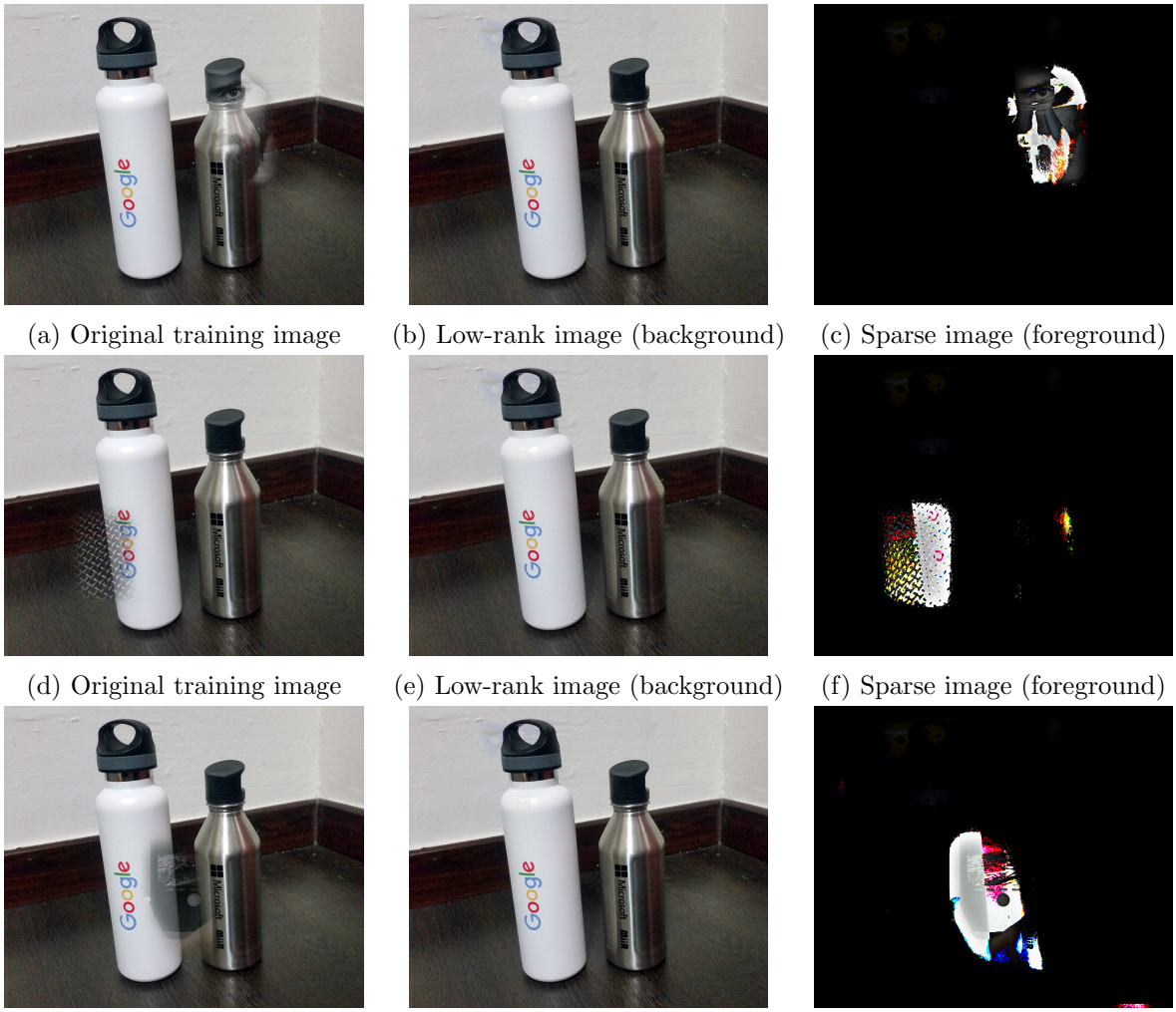
(g) Average method on 50%-α size-10%

(h) RPCA method on 50%-α size-10%

Figure 10: Resultant image on custom data set

5.3.2 By-product of RPCA

Another advantage of RPCA is that we are able to separate the training images into two components: the low-rank (background) and sparse (foreground reflection). Therefore instead of just obtaining the low-rank background image, we have a by-product, which is the sparse image. This fact is both useful and interesting because we are now able to see clearer the foreground images embedded into the scene. Some of the results are shown in Figure 11:



(a) Original training image (b) Low-rank image (background) (c) Sparse image (foreground)

(d) Original training image (e) Low-rank image (background) (f) Sparse image (foreground)

(g) Original training image (h) Low-rank image (background) (i) Sparse image (foreground)

Figure 11: Low-rank and sparse image decomposition by RPCA

5.4 Setup

All of the experiments were run using a server machine with the following specification:

- Processor: 2 × Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (8 cores, 16 threads)
- RAM: 64 GB
- OS: Ubuntu 14.04.5 LTS
- Python: 2.7.6

The experiments were run in parallel, up to 10 jobs at one time. The program uses around 4 GB RAM for running RPCA on high resolution data set (for each experiment). Our code and custom data sets are publicly available on GitHub[3].

[3]https://github.com/jeffrey-effendy/reflection-removal

5.5 Evaluation

In all of the evaluation tables below, the RPCA algorithm is run with maximum iteration of **1000**, and all of the experiments have **converged** before it reached iteration 1000. The time shown in Table 1 is in seconds.

Image	# Samples	Avging MSE	Avging Time	RPCA MSE	RPCA Time
Default Low-res	5	18.5695	0.012	4.91	65.3
	10	16.009	0.023	2.22	68.7
	15	16.2306	0.036	2.45	157.314
	20	22.1167	0.067	3.49	221.961
Image	**# Samples**	**Avging MSE**	**Avging Time**	**RPCA MSE**	**RPCA Time**
Default High-res	5	84.8606	0.422	203.6	21186
	10	90.0014	0.956	160.8	27097
	15	99.4108	1.314	174.8	29633
	20	98.6076	1.790	178.7	32444

Table 1: Performance evaluation with varying number of training images

Image	Size 10	Size 25	Size 50	Size 75	Size 100
Custom Low-res 10%-α	1.85	4.87	15.47	33.91	67.11
Custom Low-res 20%-α	4.98	15.48	54.33	77.62	96.72
Custom Low-res 30%-α	10.0	29.51	74.81	87.58	102.27
Custom Low-res 40%-α	16.4	41.7	87.68	92.98	111.39
Custom Low-res 50%-α	23.32	51.54	91.97	101.14	112.08
Image	**Size 10**	**Size 25**	**Size 50**	**Size 75**	**Size 100**
Custom High-res 10%-α	1.04	4.24	14.92	33.41	67.18
Custom High-res 20%-α	4.275	15.28	54.78	77.86	96.30
Custom High-res 30%-α	9.439	29.31	74.87	87.72	102.57
Custom High-res 40%-α	15.96	41.37	87.49	93.34	111.33
Custom High-res 50%-α	23.21	51.38	91.83	101.60	111.91

Table 2: Performance evaluation of averaging approach on custom data set

Image	Size 10	Size 25	Size 50	Size 75	Size 100
Custom Low-res 10%-α	1.19	8.23	15.37	17.35	55.24
Custom Low-res 20%-α	1.67	27.22	27.44	70.41	225.06
Custom Low-res 30%-α	2.02	51.09	62.57	181.44	547.07
Custom Low-res 40%-α	2.38	73.40	110.50	363.08	1020.59
Custom Low-res 50%-α	2.69	100.54	173.63	640.41	1690.31
Image	**Size 10**	**Size 25**	**Size 50**	**Size 75**	**Size 100**
Custom High-res 10%-α	0.225	7.60	44.68	81.6	116.34
Custom High-res 20%-α	0.466	18.69	164.03	307.26	444.33
Custom High-res 30%-α	0.806	37.47	349.90	683.27	1004.51
Custom High-res 40%-α	1.21	59.62	587.29	1162.42	1748.46
Custom High-res 50%-α	1.71	86.13	866.50	1755.13	2722.33

Table 3: Performance evaluation of RPCA approach on custom data set

5.6 Analysis

We believe that the averaging approach has reasonable performance but does not produce reliable result because of the property of the algorithm, which is unable to rule out the reflection component of the image completely. Some noise from the foreground reflection will still be present in the resultant image. On the other hand, RPCA is able to decompose the image into two components, the background image (low-rank) and the foreground reflections (sparse).

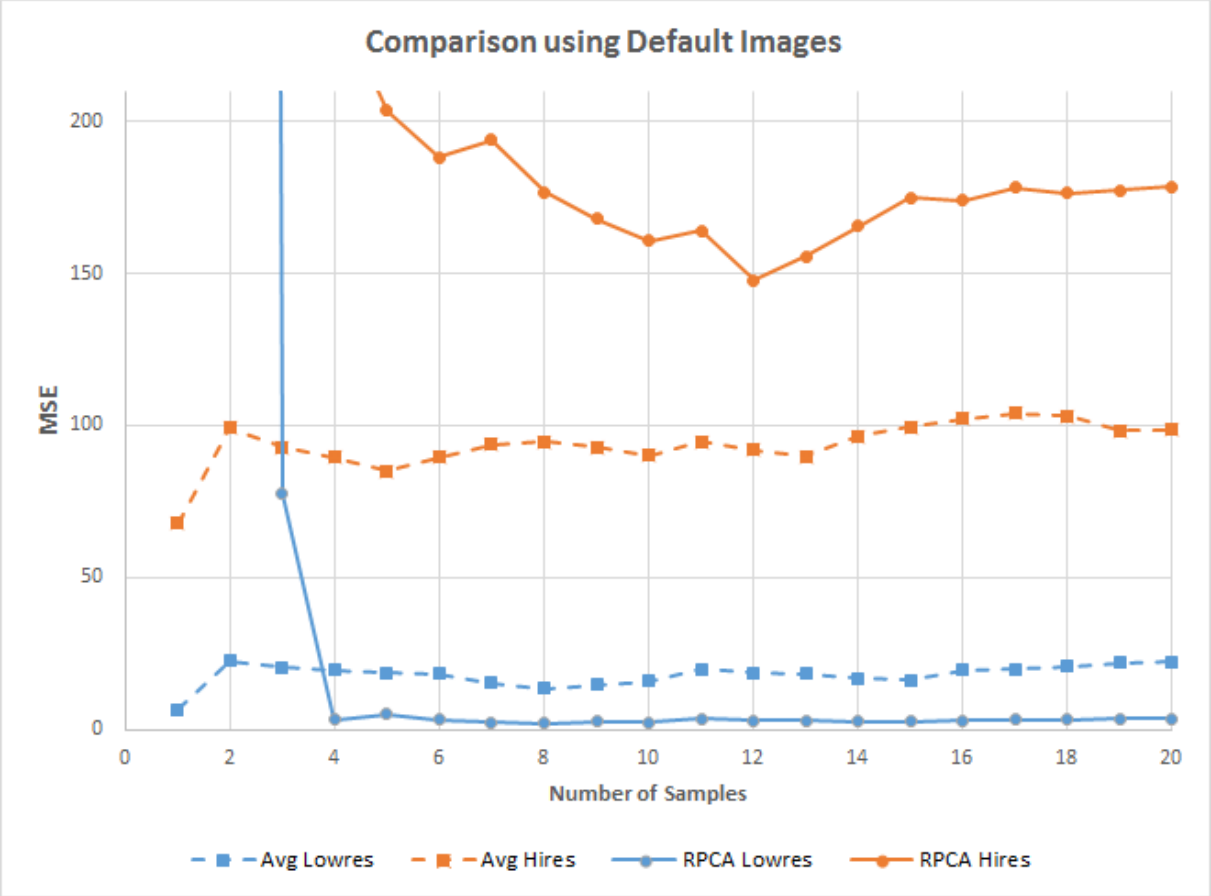


Figure 12: Averaging vs RPCA on default data set from Figure 5 and 7

As shown by some of the results in Figure 6 and 8, we can see that the image produced by both algorithms is good and most of the reflections are not visible. Observing Table 1 and Figure 12, it is clear that averaging approach works well in keeping the error stable, regardless of the number of training images given.

The averaging method error rate increases linearly when the size of the foreground reflection is increased, whereas the RPCA error rate increases exponentially as shown by Figure 14. Given training images with foreground reflections which are sparse enough (e.g., custom dataset with size-10), regardless of the opacity of the foreground reflections, RPCA is able to keep the error rate very low and stable, whereas averaging error rate increases linearly with the opacity of the foreground reflections. This observation is shown in Figure 13.

RPCA is proven to work really well with sparse reflection, as shown in Table 3, Figure 12 and 13. The MSE of RPCA on default lowres dataset is stable at MSE value of approximately 3, which is much better than the averaging method. However, RPCA did not do well on the default highres dataset because of camera noise observed in figure 8(d) and the foreground reflection is larger (not sparse). In this case, averaging outperforms RPCA. Furthermore, RPCA requires a significantly longer time to run as compared to averaging, as seen from Table 1. Averaging runs extremely fast, up to 20,000 times faster as compared to RPCA.

Figure 13: Averaging vs RPCA on custom dataset with foreground size of 10

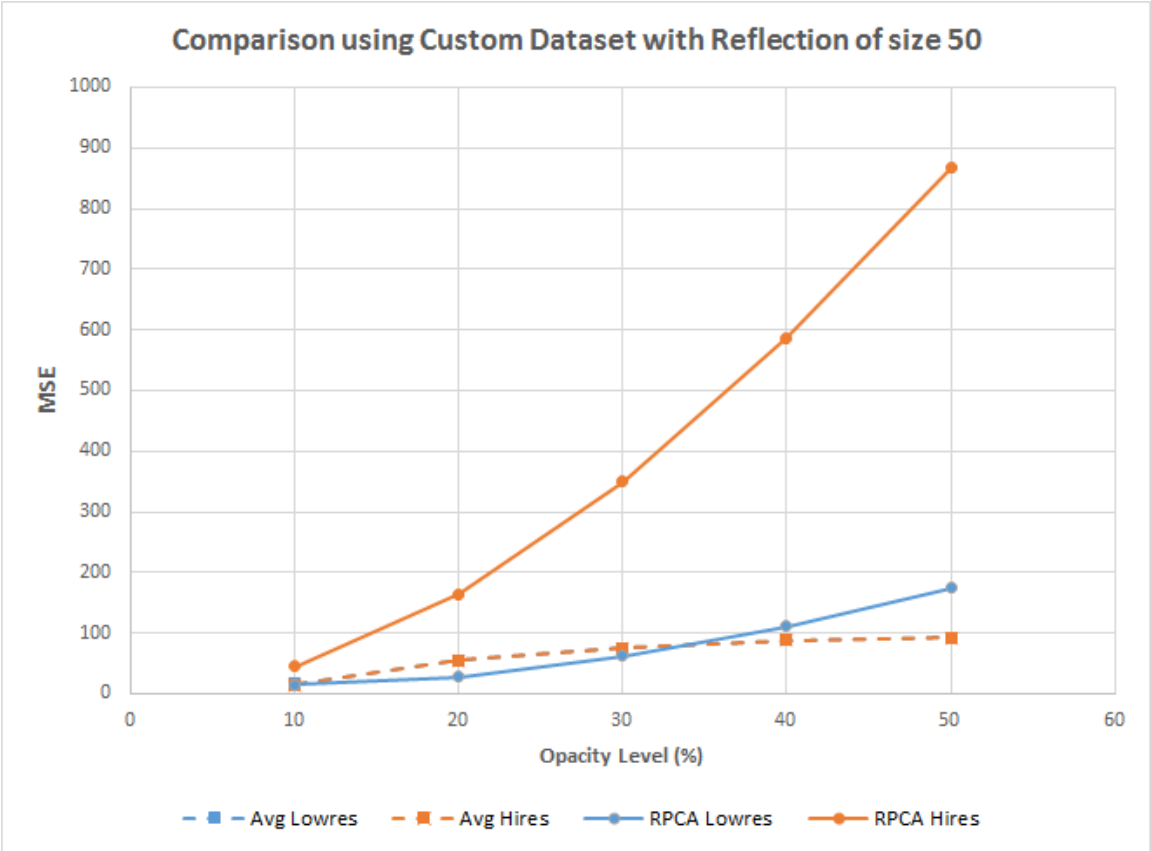


Figure 14: Averaging vs RPCA on custom data set with foreground size of 50

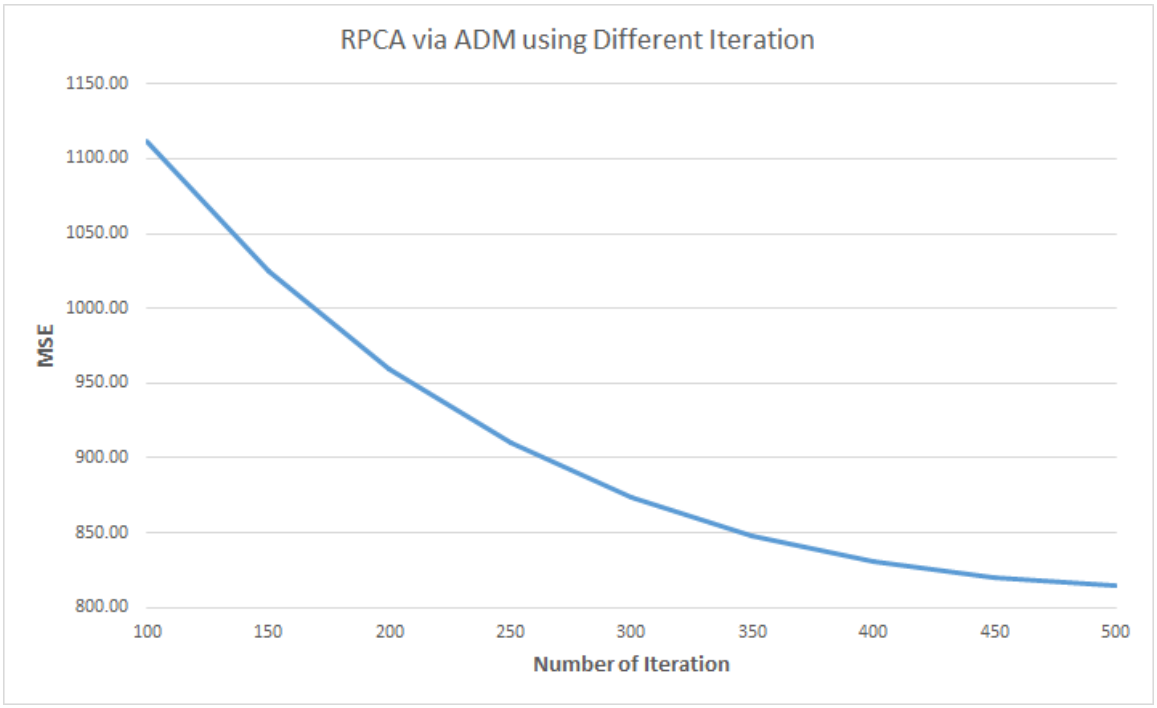


Figure 15: RPCA with various iterations on custom high res image with α-50 and size 50

We have discussed the convergence of the RPCA method in the earlier section and it has been well studied. Furthermore, we have tested the convergence empirically, by running an experiment shown in Figure 15. As we can see from the graph, the error of the resultant image decreases as the number of iteration increases with vanishing gradient, and then it will finally converge.

6 Conclusion

We have adopted two approaches to solve reflection removal task, namely averaging and RPCA. Each algorithm has its own strength and weaknesses which was discussed thoroughly and shown by the experiment results in the previous sections.

Averaging is a very simple and computationally fast method. However, due to the property of averaging, this algorithm will always include some noise from the foreground reflections to the resultant image. Hence, it is unable to produce a perfect resultant image. The resultant image will only be rough approximation to the background image that we would like to recover.

RPCA is a more complex method which is much more computationally expensive as compared to the averaging method. Given a set of training images, RPCA is able to decompose every image, to its low-rank and sparse component. However this is subject to the sparsity of the sparse component (the size of the foreground reflections). When the foreground reflections in the foreground is small enough, RPCA will be able to produce a perfect solution, a resultant image which is equal to the background that we wish to recover. It also has a by-product, which is the sparse image showing the foreground reflection that has been decomposed from the image. However, RPCA will not work optimally when the sparse component is not sparse enough, as shown in the results and experiments in the previous sections.

In conclusion, reflection removal task is a challenging task. We have shown a detailed description and analysis of two possible solutions, averaging and RPCA.

References

YiChang Shih, Dilip Krishnan, Frédo Durand, and William T Freeman. Reflection removal using ghosting cues. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3193–3201. IEEE, 2015.

L Yu and MS Brown. Exploiting reflection change for automatic reflection removal. In *Proc. of IEEE International Conference on Computer Vision*, pages 2432–2439, 2013.

Z Lin, M Chen, L Wu, and Y Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. 2009. URL `http://yima.csl.illinois.edu/psfile/Lin09-MP.pdf`.

Matthew Hu. Reflection removal algorithms. 2016. URL `http://stanford.edu/class/ee367/Winter2016/Hu_Report.pdf`.

Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *CoRR*, abs/0912.3599, 2009. URL `http://arxiv.org/abs/0912.3599`.

J Wright, Y Peng, Y Ma, A Ganesh, and S Rao. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. in: Proceedings of advances in neural information processing systems. 2009. URL `http://perception.csl.illinois.edu/matrix-rank/Files/nips2009.pdf`.

Yuan Xiaoming and Jungfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. 2009.

Spyridon Kontogiorgis and Robert R Meyer. A variable-penalty alternating directions method for convex optimization. *Mathematical Programming*, 83(1-3):29–53, 1998.

Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

Wee Kheng Leow, Yuan Cheng, Li Zhang, Terence Sim, and Lewis Foo. Background recovery by fixed-rank robust principal component analysis. In *International Conference on Computer Analysis of Images and Patterns*, pages 54–61. Springer, 2013.