Single Image Reflection Removal

Gaurav Agrawal
Department of Electrical Engineering
Stanford University
gaagrawa@stanford.edu

Ishani Parekh
Department of Electrical Engineering
Stanford University
ishanip@stanford.edu

***Abstract*—Undesirable reflections can occur in photographs taken across partially reflective surfaces such as glass windows. Robust and efficient algorithms that can remove reflections given only a single image is an open area of research. In this report, we discuss and evaluate two published reflection removal approaches. We compare the runtime performance and quality of these approaches on a published dataset as well as on images captured by us with our smartphones. With the goal of enabling reflection removal on smartphones, we propose a modified method to reduce runtime of one of the algorithms with minimal effect on image quality.**

I. Introduction

When a photograph is captured through a partially reflective surface such as a glass window, undesirable reflections may occur. It is not always possible for the photographer to avoid the glass window between camera and the scene of interest. Examples include shots taken from inside a building, from inside a train, in a museum, or through an airplane's window.

A variety of approaches have been proposed in literature to solve this problem. Some approaches make use of additional information than just the image itself. This additional information may take the form of additional images from different viewpoints, under different polarizations, different illumination conditions, different focal lengths etc. For example, professional photographers often use polarized lenses to reduce the effect of reflections. But these multi-image or multi-mode methods are not suitable for images downloaded from the internet, or for an image taken from a typical mobile camera. In this report, we do not discuss these approaches further.

Methods that aim to remove reflections given only a single image are referred to as **Single Image Reflection Removal (SIRR)** methods. As shown in Figure 1, SIRR problem can be formulated as a layer separation problem $I = T + R$, where I is the input image, T is the transmission component, and R is the reflection component. Lacking any further information, this problem is ill-posed since there are twice as many unknowns (i.e. T, R) as there are knowns (i.e. I).

In order to constrain the solution space, some approaches (e.g. [5]) leverage user annotations for additional hints. However, considering the effort and time required for annotation, this may not be the preferred approach for a typical smartphone consumer. In rest of this report, such approaches are not discussed further.

In many successful approaches, additional assumptions are made on statistics of T and R informed by the statistics of natural images or the physics of reflection. For example, studies (e.g. [6]) have shown that gradients of natural images tend to be sparse, which means the histogram of gradients has a peak at zero, which then falls off faster than the gaussian, and has a long tail. This sparsity prior has also been used to solve other ill posed problems (e.g. image deblurring).

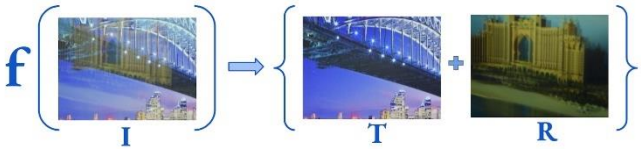


Figure 1: SIRR Problem

Rest of this report is organized as follows. In section II, we discuss two published approaches for SIRR. In section III, the quality and runtime of these approaches is evaluated. In section IV, we consider feasibility of SIRR on images captured by smartphone users and propose a way to speed up one of the methods.

II. Related Work

The SIRR algorithms proposed in [1] and [2] are used as baseline in this report. In this section we describe them in some detail.

A. *Relative Smoothness* ([1])

Objects that are part of reflection layer are often not in focus, while the objects being photographed are usually in focus. Figure 2 shows two example images with out-of-focus reflection. In such situations, the contribution of R to overall image is relatively smooth because reflection image gets convolved with the depth-of-field kernel h.

$$I = T + R = T + (R' * h)$$

The smoothing of R creates asymmetry between statistics of gradients of T and R. This can be modeled as follows.

$$P_T(x) \propto max\{exp(-\frac{x^2}{\sigma_T^2}), \varepsilon\}$$

$$P_R(x) \propto exp(-\frac{x^2}{\sigma_R^2})$$

Here x is the gradient value, while P_T and P_R are the probability distributions of gradients for transmission and reflection layers respectively. The ε parameter adds a long tail to the gradient of transmission layer, as needed for sparsity.

Gradient x for T is estimated using horizontal and vertical derivative filters f_1 and f_2 while gradient for R is estimated using Laplacian filter f_3.

$$f_1 = [-1\ 1] \quad f_2 = [-1\ 1]^T \quad f_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Assuming independence of per-pixel gradients, and independence of outputs of derivative filters, the likelihood of T and R layers can be written as:

$$\mathrm{P}(T) = \prod_i P_T(f_1 * T)_i \cdot P_T(f_2 * T)_i$$

$$\mathrm{P}(R) = \prod_i P_R(f_3 * R)_i$$

Here i is the pixel index. The layer separation problem can now be formulated as one of maximizing $\mathrm{P}(T, R)$, the joint probability of T and R. Further, assuming independence of T and R, $\mathrm{P}(T, R) = \mathrm{P}(T) \cdot \mathrm{P}(R)$. Maximizing this probability is equivalent to minimizing the negative log probability. Thus, the objective function can be written as follows.

$$\min_{T,R} \sum_i -log\ P_T(f_1 * T)_i - log\ P_T(f_2 * T)_i - log\ P_R(f_3 * R)_i$$

Recognizing that $R = I - T$ and substituting P_T and P_R, the objective function can be written as follows.

$$\min_T \sum_i \rho(f_1 * T)_i - \rho(f_2 * T)_i - (f_3 * (I - T))_i^2 \cdot \lambda$$

Here λ is the relative strength of smoothness, and $\rho(x)$ is defined as follows with parameter k representing the strength of tail of gradient distribution.

$$\rho(x) = min\left\{\frac{x^2}{k}, 1\right\}$$

Since probabilities are defined on gradients, pixel values of T need to be bounded to recover meaningful layers. This is achieved by imposing a constraint that $0 \le T \le I$.

The objective function is non-convex due to $\rho(x)$. It is optimized using *half-quadratic separation method*. In this method, auxiliary per-pixel and per-filter variables g_i are introduced along with a parameter β the value of which is iteratively increased. The reformulation is such that as $\beta \to \infty$, the solution of reformulated problem approaches the solution of original problem. In each iteration, two subproblems are solved. The first subproblem updates g_i assuming T is constant, while the second subproblem updates T assuming g_i is constant.

In order to satisfy $0 \le T \le I$, at the end of each iteration a normalization step is performed that brings pixel values of current solution into a meaningful range. This is achieved by solving a least-squared optimization problem using gradient descent.

At the start of the first iteration, T is initialized to I. In practice, the algorithm converges in less than 5 iterations of half-quadratic separation method.

"Book" *"SIR2 Toys"*

Figure 2: Images with smooth reflection

Figure 3: "Apples" mage with ghosted reflection

B. Ghosting Cues ([2])

Due to thickness of glass, shifted double reflections can occur, which may result in ghosted reflection component in the captured image. Figure 3 shows an example image where ghosting of reflected text is very prominent. Ghosting can be modeled by convolution of reflection layer with ghosting kernel k.

$$I = T + R * k$$

Ghosting kernel k includes a spatial shift $\boldsymbol{d}_k$ and an attenuation factor c_k. $\boldsymbol{d}_k$ is estimated by using 2-D auto-correlation map of Laplacian of input image $\nabla^2 \mathrm{I}$. Ghosting creates a local maximum in this autocorrelation map which is detected to estimate $\boldsymbol{d}_k$.

To estimate c_k, first key points are detected via Harris corner detection. In the vicinity of each key point, a 5x5 contrast normalized patch is compared with its associated patch at $\boldsymbol{d}_k$ shift. c_k for this patch is estimated as the ratio of standard deviation of pixel intensities in each patch. Overall c_k is estimated by taking weighted sum of per-key-point c_k, weighted by $e^{-\alpha \cdot \|p_1 - p_2\|^2}$ where $\|p_1 - p_2\|$ is the L_2 norm of difference between corresponding patches.

Once k is estimated, the objective is to minimize the following L_2 error function.

$$\min_{T,R} \ \|I - T - R * k\|^2$$

However, to constrain this problem, priors need to be applied on gradients of T and R. In this approach, 8x8 patch-based priors based on Gaussian Mixture Models (GMM) [4] are used. The modified objective function becomes:

$$\min_{T,R} \ \|I - T - R * k\|^2 - \sum_i log\ (GMM(P_i T)) - \sum_i log\ (GMM(P_i R))$$

Here i iterates over all overlapping 8x8 patches, and $P_i T$ represents the i^{th} patch of T. $GMM(p)$ returns the probability of 8x8 patch p under GMM prior from [5]. Additionally, non-negativity constraints are added on pixel value of T and R, $0 \leq T,\ R \leq 1$.

The objective function is non-convex due to GMM prior. It is optimized using half-quadratic separation method where auxiliary variables are introduced for each patch $P_i T$ and $P_i R$ along with a parameter β which is iteratively increased. The reformulation is such that as $\beta \to \infty$, the solution of reformulated problem approaches the solution of original problem. The first subproblem considers the auxiliary variables fixed and solves for of T and R. This is a quadratic problem and is solved using L-BFGS in order to handle range constraints on pixel values. The second subproblem considers T and R fixed and solves for auxiliary variables under GMM prior. This subproblem is solved iteratively using the approach in [4].

Before the first iteration, T and R are initialized using a sparsity inducing model with convex objective function. The algorithm is run for 25 iterations of half-quadratic separation method.

C. Comparison

From here on, we refer to the "Relative Smoothness" approach of [1] as **LB14** and the "Ghosting Cues" approach of [2] as **SK15**.

Looking at the similarities between LB14 and SK15, both methods make assumptions about the statistical properties of gradients of T and R. Also, both methods formulate the problem as constrained optimization problem and solve it using iterative methods. Despite these similarities, there are important differences. Firstly, LB14 uses a per-pixel gradient prior that is assumed independent between pixels, while SK15 uses an 8x8 patch prior modeled as GMM.

Secondly, the physical mechanism that creates asymmetry between T and R is different. LB14 considers R to be smoother than T, which will be the case if reflection is out of focus. The blurring kernel is never explicitly estimated but is rather modeled with a hyper-parameter in objective function. SK15, on the other hand, assumes R to contain noticeable ghosting, which will be the case if there is double reflection that is sufficiently shifted spatially and is sufficiently strong in amplitude. Moreover, for the ghosting to be strong, the reflected objects will need to be in reasonable focus. In SK15, the ghosting kernel is explicitly estimated before iterative optimization is done.

LB14 (T) *LB14 (R)*

SK15 (T) *SK15 (R)*

Figure 4: Results of LB14 and SK15 on the "Book" image

III. Evaluation

A. Datasets

For evaluation, images from following data sets are used.

1) Images from References: The "Book" and "Apples" images are used by LB14 and SK15 respectively, and we use them to reproduce and cross validate the results.

2) SIR^2: This dataset has been made available by NTU [3] for evaluating single image reflection removal algorithms. For every example image in this dataset, corresponding ground truth transmission and reflection images are also provided. All the images were captured with a DSLR camera with varying exposure times, aperture sizes and glass thickness. This allows reproduction of various algorithmic assumptions. For example, using a thicker glass in between the camera and the scene produces more prominent double reflections, while varying aperture sizes allows creating out-of-focus or smooth reflections.

3) 12M pixel mobile images: The images in above datasets are relatively small in size (<0.5M pixel). For this project, we collected a number of 12M pixel images using our smartphone cameras under various reflection scenarios, and used a subset of them for evaluation of speed and quality.

LB14 (T)

LB14 (R)

SK15 (T)

SK15 (R)

Figure 5: Results of LB14 and SK15 on the "Apples" image

B. Results on reference images

While smoothness and ghosting are both plausible to have in reflections, due to physical mechanisms that lead to those effects it appears unlikely that reflection in a given image can be both smooth and exhibit strong ghosting. We test this by running reference images from LB14 and SK15 on the two algorithms. We use reference MATLAB code available from the authors of LB14 and SK15.

Figure 4 shows the result of these two algorithms on the "Book" image from Figure 1. The reflection in this image is smooth and therefore LB14 can get good separation of T and R layers. In contrast, since there is no significant ghosting in this image, SK15 is not able to separate the reflection in R component.

Figure 5 shows the result of these two algorithms on the "Apples" image from Figure 2. The reflection in this image is sharp with significant ghosting. Therefore, SK15 can get good separation of T and R layers. In contrast, since the reflection is not smooth in this image, LB14 is not able to separate the reflection in R component.

These results suggest that these state-of-the-art algorithms are not general enough to be effective in removing a broad class of reflections.

C. Run time

It took SK15 more than an hour in MATLAB to produce the result shown in Figure 5 on a Linux machine with 12-core Intel Xeon E5-1650 @ 3.60GHz. Given that the "Apples" image is only 400 x 540 pixels, this suggests that it may not be practical to run SK15 on 12M pixel images.

To evaluate run time, we selected 9 images from SIR^2 dataset, scaled each image in MATLAB and measured run time. The results are shown in Figure 6. We make following observations from these charts.

1) The run time of algorithms is sensitive to the content of image, not just the size of image. Even for the same image size,

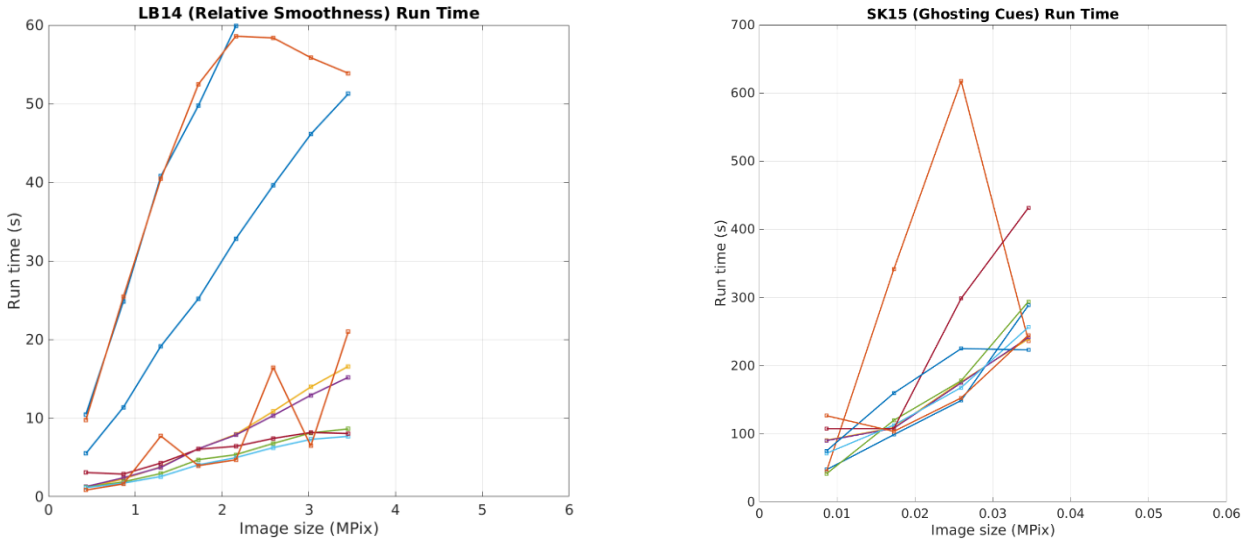


Figure 6: Run time of LB14 and SK15 on scaled SIR2 images

run times on different images show large variation. This is because portions of these algorithms take a variable number of iterations to converge.

2) SK15 is orders of magnitude slower than LB14 (note different X-axis in the two charts). The initialization step in SK15 itself runs longer than LB15. Also, while not shown in the charts, SK15 is very memory intensive. Even on the small "Apples" image, it used 20GB of memory. For larger images, our workstation quickly ran out of memory.

IV. Suitability for Large Images

Based on evaluation results described in previous section, we recognize following challenges to efficiently remove reflections from multi-megapixel images taken on mobile phone cameras.

1) SK15 takes too long to run and consume too much memory even on <0.5M pixel images. LB14 is much faster, but still took up to 9 minutes to run on a 12M pixel image.

2) Run time of algorithms is data dependent, and therefore hard to bound.

3) The visual quality of results is good under suitable type of reflections, but not under different kinds of reflections.

In rest of this report, we consider heuristics to come up with a fast reflection removal method suitable for large images.

A. *Speedup Approaches*

Because LB14 is much faster than SK15, we did not consider SK15 further. Instead we attempted to tweak LB14 to make it run faster. We considered following ideas.

1) Run LB14 on the downsized version of full-res image, detect image regions with large reflections, and then run LB14 on selected crops of full-res image. See Figure 7.

2) Run LB14 on the downsized version of full-res image, upscale the R layer, and subtract it from full-res image to estimate full-res T layer. See Figure 7.

3) Tweak the hyper parameters of LB14 such that we run fewer iterations or relax convergence criterion.

Approach 1) relies on the fact that in most images, reflection artifacts will be present only in a small part of image. Therefore, if we can detect which regions of image are corrupted by reflection, reflection removal can be run only on those patches, this speeding up the algorithm. However, this approach did not

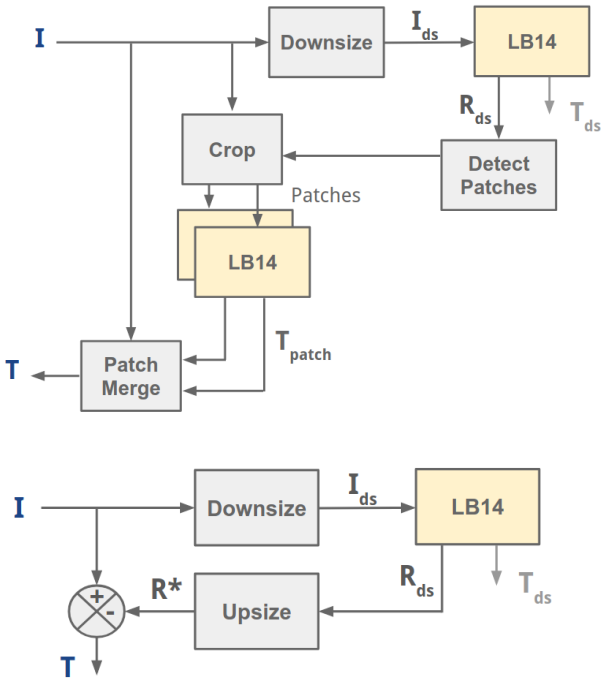


Figure 7: Approaches to speed up LB14 on larger images

work well because in LB14, the R component also extracts the smooth parts of T layer. Therefore, a simple thresholding / local peak-detection-based approach did not reliably detect the patches where actual reflection was present.

Approach 2) relies on the realization that since R component is smooth, if we extract it from downsized image, and then upsize it, it can serve as an approximation to full-res R component.

For approach 3), we reduced the number of iterations, and relaxed the convergence criterion for the renormalization step of LB14.

B. Results

We report results from approaches 2) and 3) described in previous section as they produced good quality of results, while also speeding up the algorithm.

Table 1 shows run times on 6 images captured with our mobile phones. Each image is 12M pixel in size. Images with smooth reflections were selected since that is required by LB14 for good reflection removal. Run time was measured in MATLAB on a Linux machine with 12-core Intel Xeon E5-1650 @ 3.60GHz.

TABLE I. RUN TIME OF SIRR ON 12M PIXEL IMAGES

Image	**Run Time (sec)**			
	LB14 Baseline	***LB14 + Parameter Tuning***	***LB14 + Resize***	***LB14 + Both Optimizations***
Boat	61	9	16	3
Cardash	150	75	33	16
Fort	45	12	11	4
Trees	214	71	47	5
XKCD	489	44	97	11
Whiteboard	493	48	97	12

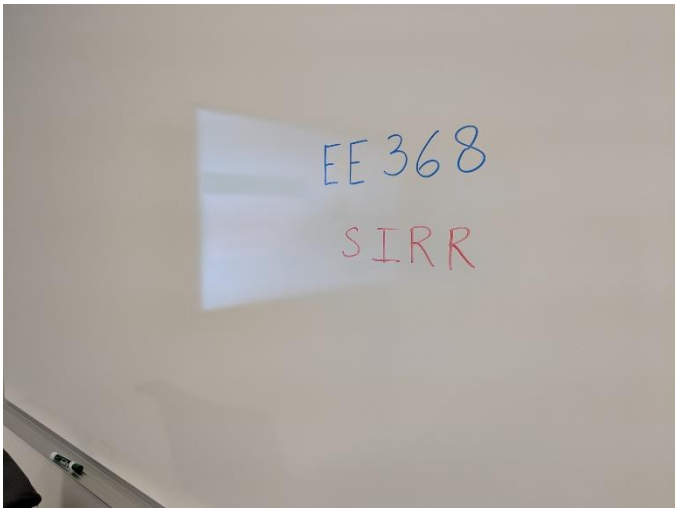
The baseline LB14 algorithm takes up to 8 minutes to run. Combining approaches 2) and 3), the modified algorithm runs within 16 seconds. That is a speedup of around 30x for the worst case run time.

SIRR results on "Whiteboard" image are shown in Figure 8. The first image is the input image with reflection, the next image is produced by LB14 in 8 minutes, and the last image is produced after our modifications and runs in 12 seconds. For this image, there is almost no degradation in image quality. Metrics like Structure Index and Normalized Cross Correlation report the speeded up output images to have large similarity score (>0.99) vs. baseline result.

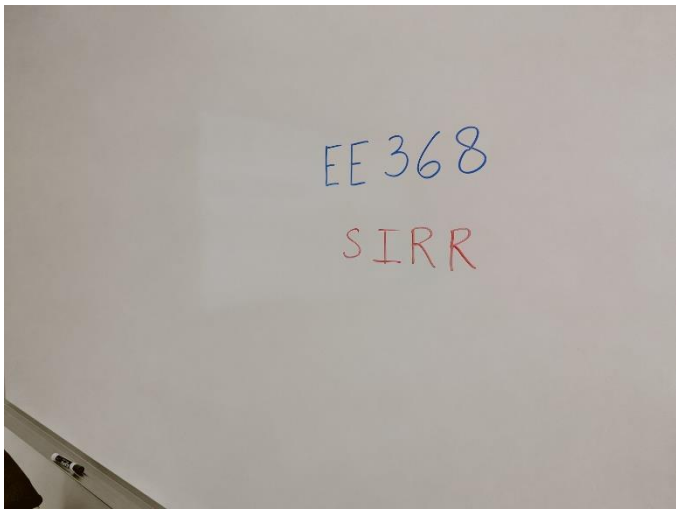
Results on other mobile images are included in appendix as part of supplemental material.

V. CONCLUSIONS

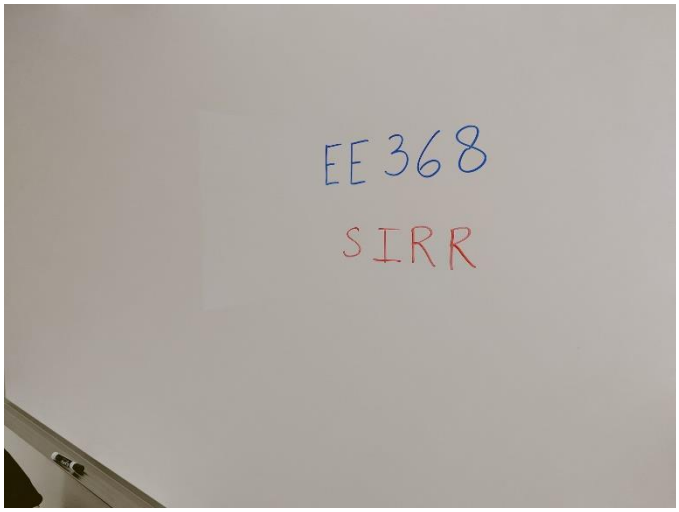
In this project, we evaluated two recent SIRR algorithms with the goal of removing reflections in images captured by typical smartphone cameras. While the algorithms perform well when corresponding assumptions on reflection layer are



"Whiteboard" Input Image



T Layer from LB14



T Layer from modified LB14

Figure 8: Results on 12M Pixel "Whiteboard" image

satisfied, we found the assumptions are often not true in many practical reflection scenarios. Secondly, SIRR algorithms are very compute intensive and do not appear suitable to run in real time within the compute budget of mobile phones. We proposed a simple heuristic modification that speeded up SIRR by up to 30x on 12M pixel images with results very similar to baseline.

We conclude that the quest for fast and robust SIRR algorithms capable of running on large images is far from over, and many opportunities exist for further research.

ACKNOWLEDGMENT

The authors would like to thank Prof. Bernd Girod, Jean-Baptiste Boin and Jayant Thatte for stimulating lectures, problem sessions and homework assignments.

We would also like to thank authors of [3] for allowing download and use of SIR^2 dataset for this project.

REFERENCES

[1] Y. Li and M. S. Brown. Single image layer separation using relative smoothness. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[2] Y. C. Shih, D. Krishnan, F. Durand, and W. T. Freeman. Reflection removal using ghosting cues. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3193-3201

[3] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, Alex C. Kot. Benchmarking Single-Image Reflection Removal Algorithms. The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3922-3930

[4] Zoran, Daniel, and Yair Weiss. "From learning models of natural image patches to whole image restoration." 2011 International Conference on Computer Vision. IEEE, 2011.

[5] Levin, Anat, and Yair Weiss. "User assisted separation of reflections from a single image using a sparsity prior." IEEE Transactions on Pattern Analysis and Machine Intelligence 29.9 (2007): 1647-1654.

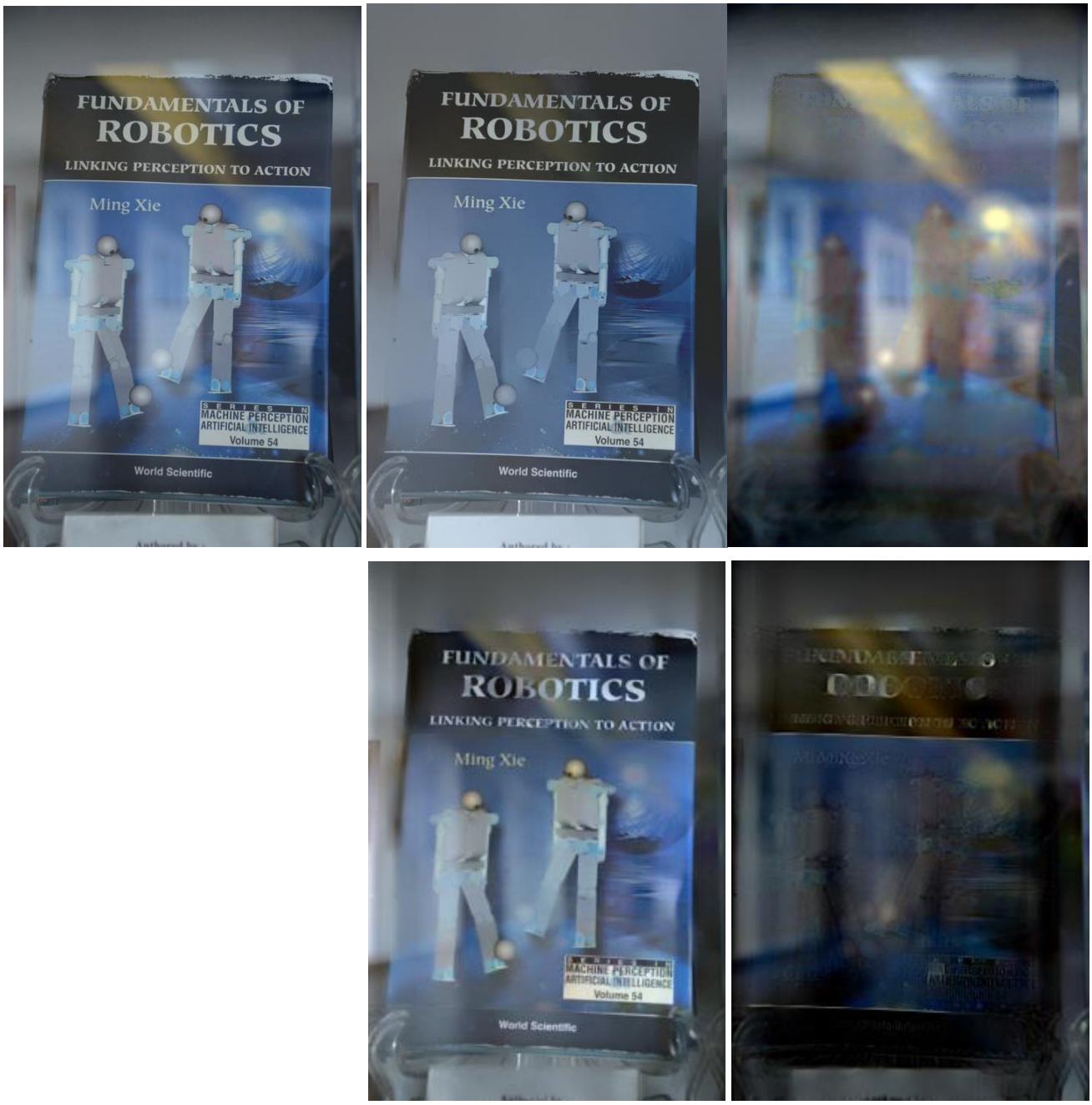
[6] B.A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381:607–608, 1996.

APPENDIX

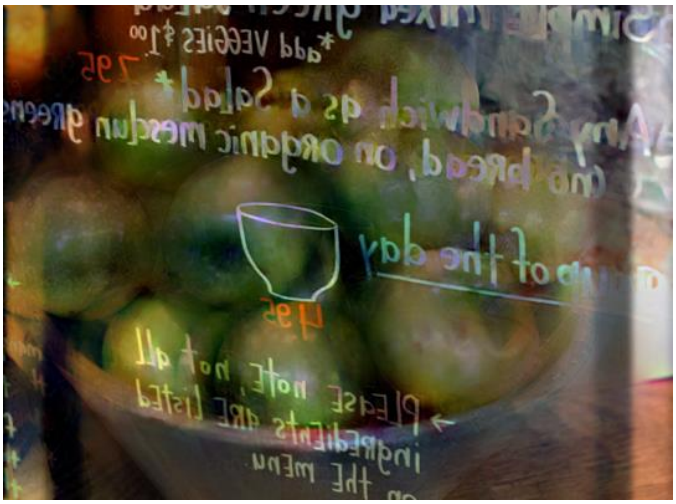
CONTRIBUTIONS

Both the authors worked closely and shared work involving all parts of this project. The idea of this project topic came from Ishani. She also collected mobile image dataset used in this project. Gaurav implemented the MATLAB code for evaluation of algorithms and proposed approaches to speed up SIRR.

SUPPLEMENTAL IMAGES



Top Row: Reference "Book" image, T from LB14, R from LB14
Bottom Row: T from SK15, R from SK15



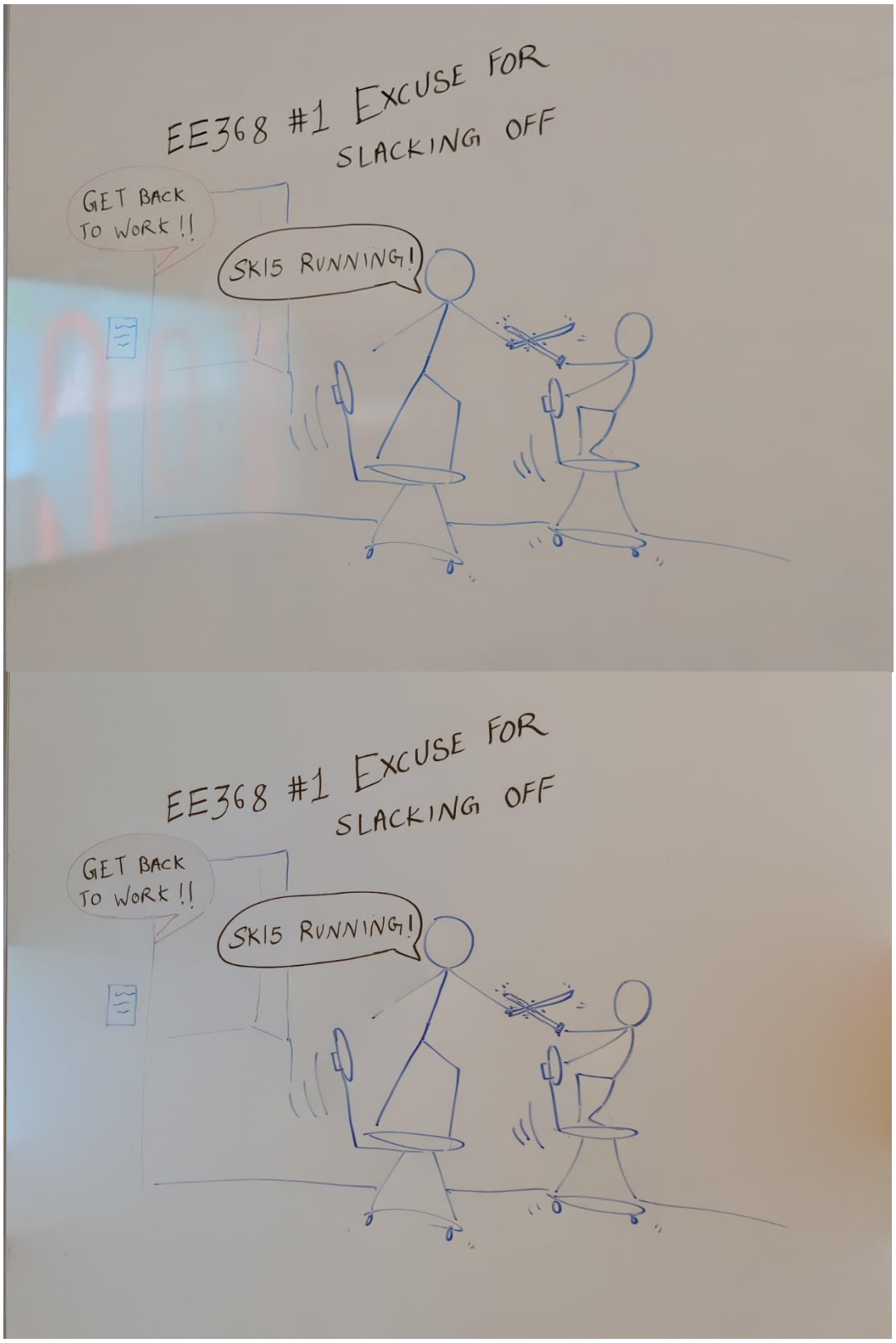
Top Row: Reference "Apple" image. Middle Row: T and R from SK15
Bottom Row: T and R from LB14

Top Row: Synthetic image for SK15 evaluation.
Middle Row: T layer from SK15, Bottom Row: R layer from SK15

Top Row: Example image from SIR2 dataset.
Middle Row: T layer from LB14, Bottom Row: R layer from LB14

Top Row: Example image from SIR2 dataset.
Middle Row: T layer from LB14, Bottom Row: R layer from LB14

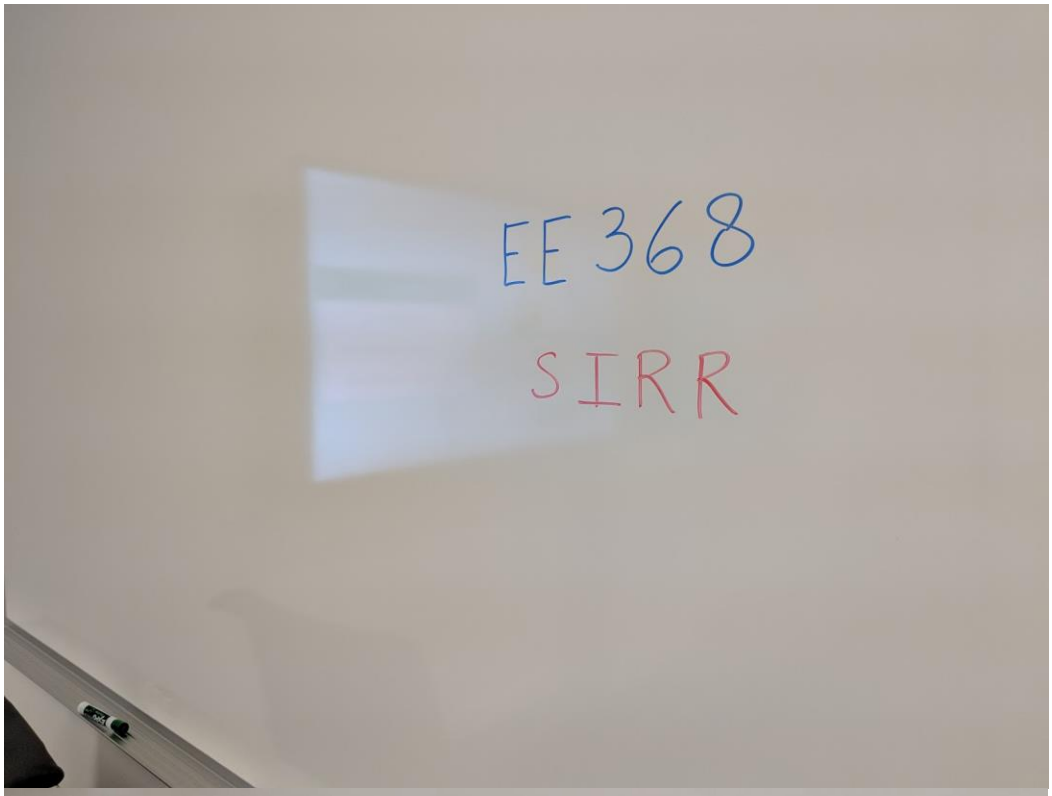
Top Row: “Trees” image with reflection.
Bottom Row: Output image with our implementation.

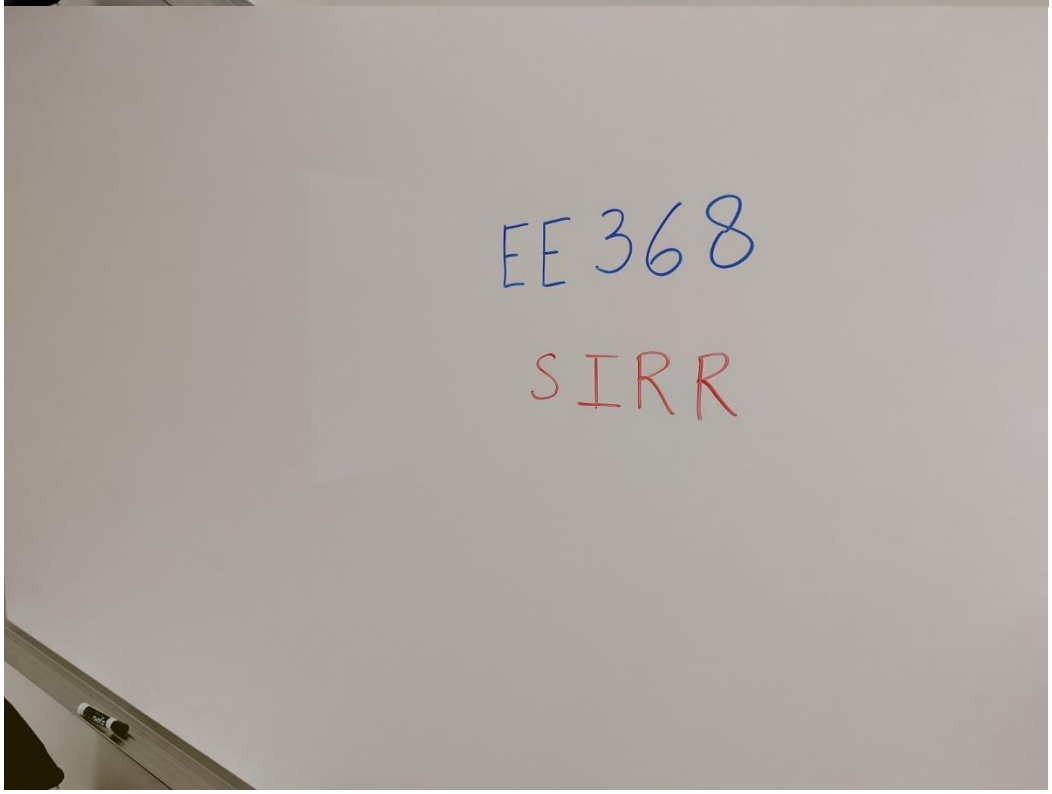


Top Row: "XKCD" image with reflection.
Bottom Row: Output image with our implementation.

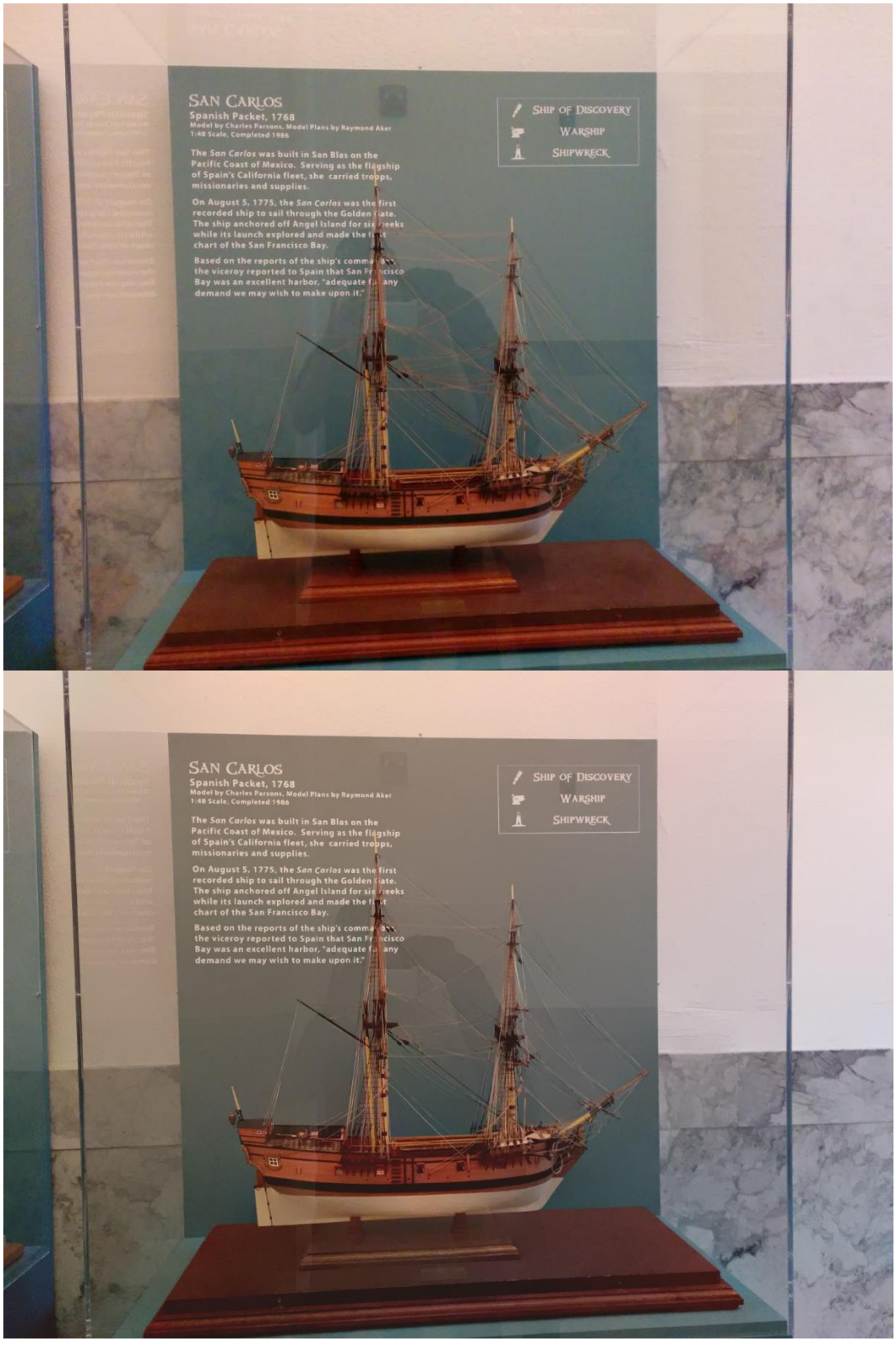
Top Row: “Fort” image with reflection.
Bottom Row: Output image with our implementation.

Left Image: "Cardash" image with reflection.
Right Image: Output image with our implementation.





Top Image: "Whiteboard" image with reflection.
Bottom Image: Output image with our implementation.



Top Image: "Boat" image with reflection.
Bottom Image: Output image with our implementation. Note slight color change.