Object/Defect Removal via Single-image Super-resolution on NLM-priority-based Inpainting and Sparse Coding

Haihong Li
Stanford University
hhli@stanford.edu

Joanna Xu
Stanford University
xuhan@stanford.edu

Chen Zhu
Stanford University
chen0908@stanford.edu

Abstract

Object removal is a topic highly involved in a wide range of image reconstruction applications such as restoration of corrupted or defected images, scene reconstruction, and film post-production. In recent years, there have been many efforts in the industry and academia to develop better algorithms for this subject. This paper discusses some of the recent work and various techniques currently adopted in this field and presents our algorithmic design that enhance the existing pixel-filling framework, and our post-inpaint refinement steps. This paper will further layout the implementation details and experimental results of our algorithm, on a mixture of images from both the standard image processing study papers and our own photo library. Results from our proposed methods will be evaluated and compared to the previous works in academia and other state-of-the-art approaches, with elaboration on the advantages and disadvantages. This paper will conclude with discussing some of the challenges encountered during the design and experiment phases and proposing potential steps to take in the future.

1. Introduction

1.1. Motivation

In the world of photography, a perfect shot can be attributed to aesthetics factors such as scene composition and artist's vision, as well as technical standard such as lens capabilities and sensor resolution. However, an image can still occasionally be flawed by random intruders or undesirable objects that are unexpectedly presented in the scene. There are also scenarios where we wish to preserve the quality of worn-out old photographs, which may have debris, fold marks, or scratched surface. Both cases require a post-processing method to restore the desired image quality, which can be realized through computational imaging techniques such as inpainting, texture synthesis, and sparse coding.

1.2. Objective

The goal of this project is the design and implementation of advanced object removal algorithms applicable for distinct object removal and small-scale defect removal. We investigated existing methods and proposed our own advanced algorithmic design that can be categorized into two distinct image patterns. Our designs are to cover the applications on both the distinct object removal from an image as well as reconstruction of scratched or defected image. We aim to improve the quality of the output image in terms of both visual coherency and the preservation of details.

The algorithms will be implemented through MATLAB software. We experimented with images from both the standard imaging-related academic papers as well as those from our own photo libraries with different resolutions to enhance the variety of our test pool.

2. Related Work

There has been recent development of image reconstruction algorithms in the past 20 years that pay particular attention to the topic of object removal. Existing methodologies generally fall into the two categories: texture synthesis and inpainting.

The core of the texture synthesis and image quilting method is the generation of a new larger image, from synthesis and stitching together small patches of the existing image. This method was originated from Ashikhmin's 2001 paper, and improved by Dr. Freeman, which places random blocks of the original image and construct neighboring blocks constraining by overlap and minimizing error boundary cut.[1] It can be viewed as a heuristic approach that utilize the repetition of two-dimensional textural patterns to edge-stitch the missing cut-out zone. It seeks to replicate texture with moderate stochasticity to fill in the missing pixels. This method works particularly well for images with repetitive or periodic texture patterns, such as wood grain, grass, snow, and fruit piles. However, this method's performance is highly constrained to the pattern of the target image being reconstructed. For images with irregular scene composition or texture patterns, this method would result in artifacts and incoherency.

Another commonly-referred technique is ”inpainting”, which targets the pixel filling after a single-object removal from an image. This method highlights the process of taking other patterns/contours/edges near the cut-out edge region, and extending the pattern to fill the object-removed zone. Early elaboration on this method was seen in Dr. Marcelo Bertalmo and his colleagues' proposed algorithm based on fluid dynamics and the use of partial differential equations. This algorithm simulates a traversal along the edges from known regions to the unknowns. As edges of objects are meant to be contiguous, we may continue the isophotes while matching gradient vectors at the boundary of the targeted inpainting region (with reference to methods used in fluid dynamics). Once the contours are reconstructed, colors can be filled to reduce minimum variance in that region.[4]

An extension to this method was seen 2003 paper by Criminisi, Perez, and Toyama from Microsoft research, whom proposed the ”Exemplar-based ” inpainting technique that uses the extension of isophotes, or linear structures of the image patter to fill inward. The paper presents the ”best-first” algorithm, with idea that the order of filling matters and should prioritize the pixels located more towards the known region.[3] Another similar method was proposed by Alexandru Talea in 2004, and is based on a mathematical model - the Fast Marching Method. The algorithm starts from the exterior of the to-be-inpainted cut-out zone and graduates towards the interior, crossing through the boundary edge, and fill the region along the way. Each pixel in the inpainting zone is filled with a normalized weighted sum of the known pixels in its neighborhood. Pixels locating near the targeted inpaint pixel are given more weight, and so are the pixels near the normal of the boundary and lying on the boundary contours.[2] Such filling method takes into consideration every direction from the center of the cut-out zone, producing a relatively coherent visual effect on the cut-out zone.

A newer conference paper from European Conference on Computer Vision suggests that combining super-resolution with the Perez's exemplar-based inpainting is viable. Authors Le Meur and Guillemot proposed that applying inpainting on coarse or low-resolution version of the image would help reducing computational complexity and reduce sensitivity to noise.[8] We believe this points us to an interesting path to explore on.

Sparse representation of signals has received growing interests in academiain recent years. The intuition behind this is to simulate human visual coding process computationally[20]. The idea of sparse coding is to represent a signal $x \in R^N$ with as few coefficients as possible based on an over-complete dictionary $D \in R^{N \times k}$. We call each column in D an atom, and since D is an over-complete dictionary, $k > N$. The above idea can be modeled as an optimization problem as follows[18].

$$min\|D\alpha - x\|_2, s.t. \|\alpha\|_0 \leq L$$

where $\| \cdot \|_p$ denotes l_p-norm of a signal, $D \in R^{N \times k} (k \geq N)$ is the overcomplete dictionary, α is the coefficient vector.

3. Methodology

This section provides details of the two categories of approaches we explored: 1. distinct object removal with single-image super-resolution on priority-based image inpainting 2. sparse coding for reconstructing image with small-scaled defects or noises.

3.1. Single-Object Removal

Fig. 1 on the next page shows the algorithmic flow of our improved single-object removal pipeline combining the super-resolution with NLM-priority based inpaint and unsharp masking.

3.1.1 Single-image Super-resolution

Before processing the inpainting part of the object removal, down-sampling is performed on the image. Allowing inpaint function to work over the down-sampled image reduces computational complexity. More importantly, the down-sampling step retains the dominant structures and edges of the original image while lowering the focus on sparse regions or noises. It helps filtering out the noise during the inpaint process, improving the quality of the output from super-resolution and enhancing the algorithm's robustness.

After the inpainting stage, a single-image super-resolution step needs to be taken to restore the image to its original resolution. There are various sophisticated algorithms in previous scholarly articles, such as the optimization process in Glasner's 2009 paper, that touches upon this process.[15] However, after evaluating the complexity-performance trade-off, we decided to employ the classic bicubic interpolation for low-high resolution transformation.

[1]unsharp masking image from Wikipedia ”unsharp masking” main-page; bicubic image: `https://software.intel.com/sites/default/files/did_feeds_images/`

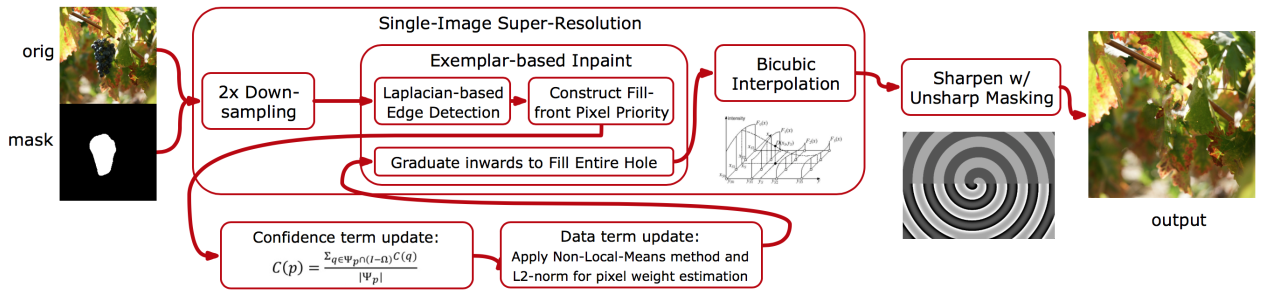
orig
mask
Single-Image Super-Resolution
2x Down-sampling
Exemplar-based Inpaint
Laplacian-based Edge Detection
Construct Fill-front Pixel Priority
Graduate inwards to Fill Entire Hole
Bicubic Interpolation
Sharpen w/ Unsharp Masking
output
Confidence term update:
$C(p) = \frac{\Sigma_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi_p|}$
Data term update:
Apply Non-Local-Means method and L2-norm for pixel weight estimation

Figure 1. Algorithmic flow of the Single-image Super-Resolution with NLM-priority based inpainting.[1]

3.1.2 Priority-based Inpaint with Non-Local-Means and L2-norm

At the inpainting stage, the Laplacian-based edge detection method takes in the input image and the mask to compute for the boundary of the removed area, which we call the "fill-front". Once we obtain the set of pixels at the fill-front, we need to determine which pixel to be filled first. The order of filling is extremely crucial in the algorithm because as we graduate inwards, the inner pixels will be dependent on the previously filed outer pixels. The naive method would be to simply build along the edge in a circular manner. However, this circular order can possibly risk image coherency because the gradient or sparsity of the image in the source region can vary along the boundary, and those regions with dense details are not given enough emphasis. Therefore based on the existing isophote or PDE-based inpainting method, we designed a more advanced version of inpainting algorithm that utilize the Non-Local-Mean technique and L2-norm in determining the pixel filling order.

During the iteration of pixel filling, the order (or priority P_{ij}) of the filling is determined by the product of two terms: the confidence term C_{ij} and the similarity term S_{ij}. The confidence term tells us how much the current pixel lies within the source region, or the weighted ratio of the known region (represented as 1) to unknown region (represented as 0).[3] The similarity term is computed based on the Non-Local-Means method, showing how similar the pixel window is to the surrounding source regions within a defined search range. This method is critical and effective because from a coherency point of view, the more similar this current pixel window is to its surrounding area, the earlier this pixel should be filled in order to preserve the regional pattern in this portion of the image, and the more likely this pixel can be filled with optimal patch from the surrounding area.

Thus for each pixel to be filled, we compute their priority as follow:

$$P_{ij} \;=\; C_{ij} \,*\, S_{ij}$$

where,

$$C_{ij} \;=\; \frac{\Sigma_{mn \in \Psi_{ij} \cap (I-\Omega)} C_{mn}}{|\Psi_{ij}|}$$

$$S_{ij} \;= \| \frac{1}{Z(i)} \exp^{-\frac{\Sigma_{mn}(k_{mn} v((N_i)_{mn} - (N_j)_{mn}))_2^2}{h^2}} \|_2$$

$$Z(i) \;=\; \Sigma_j \exp^{-\frac{\|v(N_i) - v(N_i)\|_{2,a}^2}{h^2}}$$

For each filling stage, the pixel with highest priority will be filled by the most similar patch Ψ_{q} in the known region.[3] Specifically, a patch in the known region Ψ is deemed "the most similar" to the target pixel's patch Ψ_{p} if the mean square difference between the two is the smallest. Formally it can be expressed as $\Psi_{\mathrm{q}} = \arg\min_{\Psi_{\hat{q}} \in \Psi} \mathrm{MSE}(\Psi_{\hat{\mathrm{q}}}, \Psi_{\mathrm{p}})$, where $\mathrm{MSE}(\cdot,\cdot)$ denotes the average of squared differences of pixel values in two patches. After finding the min-difference patch Ψ_{q}, then each unknown pixel in patch Ψ_{p} assumes the pixel value of the pixel at the corresponding position in that min-difference patch.

3.1.3 Unsharp Masking

The post-SR image has retained the same resolution as the pre-processed image, however, we added another post-processing step to sharpen the image. We apply unsharp masking to the resulting image after interpolation for rendering. Unsharp masking is a image sharpening process where low-frequency components are weakened and high-frequency are strengthened. Formally, the process can be written as $b = F^{-1}(F(x) - F(x) \cdot F(c))$, where c is a low-pass Gaussian kernel, x is the input image, b is the resulting image, and operator F, F^{-1} denote Fourier transform and its inverse. This additional step contributes in highlighting the details of the post-SR image.

3.2. Scratch/Defect Removal: Sparse Coding

3.2.1 Pursuit Algorithm

The described optimization problem is a NP-hard problem. Thus only approximation methods are allowed to solve this problem. The simplest way is to use greedy algorithm to approximate satisfying results. Matching pursuit and orthogonal matching pursuit[17] are two of the most representative algorithms. The idea is to project each atom to the current residual vector and find the atom that maximize the inner product with the residual, until the residual satisfy the stopping criterion or the sparse representation vector α reaches the sparsity restrictions.

Algorithm 1 Matching Pursuit

```
Input: Signal $y \in R^N$, dictionary $D \in R^{N \times k}$
Output: Coefficient vector $\alpha \in R^k$
Initialization:
$res_1 = y$
$n = 1$
REPEAT UNTIL CONVERGE:
- find $d_k \in D$ with maximum inner product:
$| < d_k, res_n > |$
- $a_n = \frac{<d_k, res_n>}{\|d_k\|^2}$
-$res_{n+1} = res_n - a_n d_k$
- n = n +1
```

Algorithm 2 Orthogonal Matching Pursuit

```
Input:
Signal: $y \in R^N$
Dictionary: $D \in R^{N \times k}$
L: sparsity constraint Output: Coefficient vector $\alpha \in R^k$
Initialization: $r_0$ = y
REPEAT UNTIL CONVERGE:
- $p = D^T r_{k-1}$
- $l_k$: add to list index where column $|p|_i$ is maximum
- $D_k$: atoms from D which have entries in $l_k$
- $x_k \leftarrow argmin_x \|y - Dx\|$
-$r_k = y - D_k x_k$
```

A more complex method is basis pursuit, where we relax the l_0 norm to l_1 norm as follows.

$$min\|D\alpha - x\|_2, s.t.\|\alpha\|_1 \leq \epsilon$$

Then we can solve this problem using augmented lagrange multipliers.

3.2.2 Choosing a dictionary

In this optimization problem, D and A are both unknown, which leads to another challenge: finding the right dictionary that yield the sparsest A. An easy way to do this is to choose a pre-defined dictionary. In some cases, it leads to fast and efficient implementation. Discrete cosine transform (DCT) , wavelets, curvelets, short-time Fourier transform etc. are usually used as described in literature[14].

However, these dictionaries don't perform equally well in all situations. Their performance depends on how well they suit the sparse representation of the signals in a specific problem. Another choice is to learn a dictionary that suits the specific problem better. One of the methods of learning a dictionary is called K-svd, which is a generalization of k-means. The algorithm is described in Algorithm 3[14].

Algorithm 3 K-SVD

```
Input: $Y \in R^{n \times p}$: each column in Y represent a training sample randomly chosen from the training set
Output: $D \in R^{N \times k}$: a learned dictionary with k atoms
Initialization: set D with k normalized columns, iter = 0

REPEAT UNTIL CONVERGE:

  • Sparse Coding Stage:
    Use any pursuit algorithm to compute the representation vectors in A column by column.

  • Update dictionary Stage:
    FOR EACH COLUMN IN D
      - Define the group of example that use this atom, $\omega_k = i, 1 \leq i \leq N, x_T^k \neq 0$
      - Compute the overall error matrix $E_k$,
        $$E_k = Y - \sum_{j \neq k} d_j x_T^j$$
      - Restrict $E_k$ by choosing only the columns corresponding to $\omega_k$ and obtain $E_k^R$
      - Compute SVD of $E_k^R$ and obtain $E_k^R = U \Sigma V^T$.
      Update current column in dictionary D with the first column of U. Update the coefficient vector $x_R^k$ with $\Sigma(1,1)V(:,1)$
```

We call this algorithm K-SVD since it's a generalization of k-means. In k-means, we represent the data with only one representative centroid in the data set, while in this case, we are allowed to used several atoms to represent signals under sparsity restrictions. Using this method, we can obtain a dictionary that suit specific signals better compared to predefined dictionaries. After we find the dictionary, we can solve the optimization problem using any pursuit algorithm described in the previous section.

3.2.3 Implementation Details

Here we further elaborates on some of the implementation details for the sparse coding method.
For efficiency, we split our training images and degraded images into patches and deal with one patch at a time. This can be modeled as follows.

$$min_{D,A}\sum_{j=1}^{p}\|D\alpha_j - y_j\|_2^2, s.t.\forall\|\alpha_j\|_0^0 \leq L$$

where $Y = (y_1, y_2, \cdots, y_p), y_j \in R^N$ are p patches split from the image after vectorizing, $D \in R^{N\times k}$ is the dictionary with k being the number of atom, $A = (\alpha_1, \alpha_2, \cdots, \alpha_p)$is the sparse representation matrix with each column $\alpha_j \in R^k$ being the sparse representation of signal y_j.

- Training a dictionary
  To train a dictionary, we use 4 images with human faces from USCimages[11]. Note that the lena image was removed from the training set. We first split traning images into p 16×16patches, then vectorize each patches into p 256×1vectors and use these vector to form our signal matrix $Y \in R^{N\times p}$. After that we apply column by column kSVD to Y and obtain a trained dictionary.

- Recover the image
  Similarly, we split our degraded images and mask into several patches of 16×16, vectorize each patches and form the signal matrix Y. Then we can use any pursuit algorithm to obtain its sparse representation matrix A. Note that only pixels in non-degraded areas are calculated in this stage. For this problem, we find that OMP is more efficient and easier to converge, so we use OMP both in training stage and reconstruction stage. We test the performance of both pre-defined dictionary and learned dictionary.

4. Results and Analysis

4.1. Experimental Setup

In order to examine the robustness of our design, we incorporated not only the standard image from image-study scholarly articles, but also photos captured with our own commercial digial cameras. The images under test are of different resolutions and color patterns, which largely enhance the variety of our test pool.

The input masks are customized for each image based on the targeted object to be removed and hand-cropped with paint. The masks are in RGB format, with 0 representing known region and 1 representing unknown region in all three channels.

4.2. NLM-priority-based Inpainting

To evaluate the performance of our NLM-priority-based inpainting method, we experimented with two images: the bungee image commonly used in image-reconstruction literatures, and the air balloons taken with our commercial Sony digital camera. Fig. 2 shows the original "bungee"/"balloon" images with their corresponding masks, and the resulting images computed from both Perez's method and our NLM method. Fig. 3 shows the priority terms. The confidence plot's yellow represents degree of known and blue represents degree of unknown; The similarity plot presents the darker (more blue) region as the region which bears lower NLM weights.

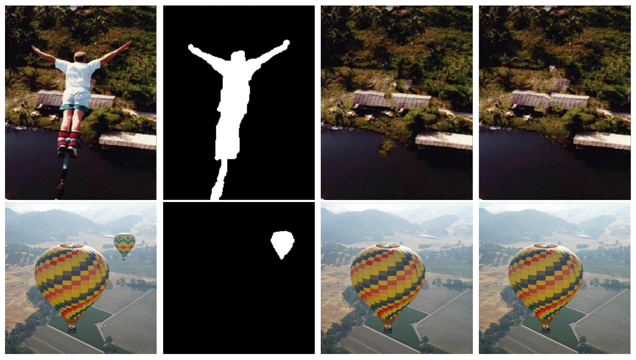

Figure 2. a) original img, b) mask, c) Perez's method, d) NLM.

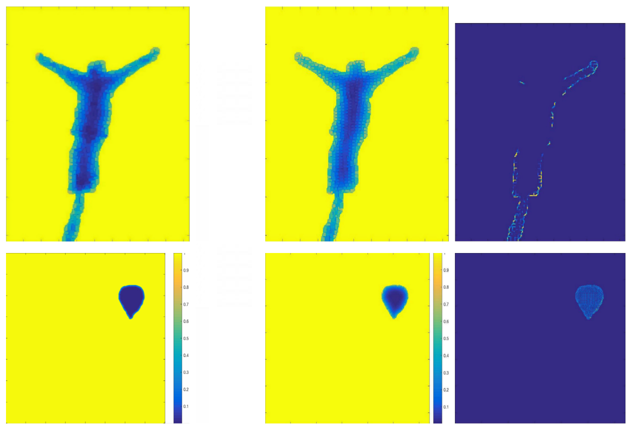

Figure 3. a) Perez's confidence plot, b) NLM's confidence plot, c) NLM's similarity plot.

From the image in Fig. 2 it can be noted that (1) In the bungee image, Perez's method (in c) does not handle the top of the building roof very well - there is an apparent discontinuity shown as a gap on the rooftop, while our method (in d) not exhibits very minor breach, but also extends the roof pattern coherently; (2) in the region where the far-end balloon is removed, Perez's method results in unnatural artifacts along the bushes, while our method preserves the consistency and details of the bush-ground boundaries without visually noticeable artifacts.

4.3. Single-image Super-resolution on Inpainted Image

Having proven that the NLM method does exhibit effectiveness in the inpaint stage, we further experimented with the the single-image super-resolution and unsharp masking in the pipeline. Fig. 4 shows 4 sets of results with our proposed object removal method.

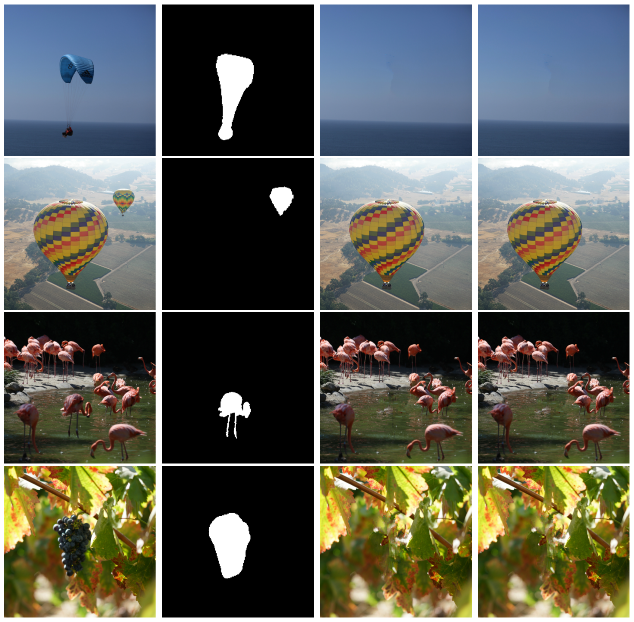
Figure 4. a) original image, b) mask, c) post-inpaint, d) SR and sharpened final image

Fig. 5 and 6 shows 4 test runs of the same image in Fig. 4, juxtaposing the results obtained from our method with the 9x9 patch-size exemplar-based inpaint[12], and with the Coherency Sensitive Hashing method[13]. The filling regions have been enlarged for viewing purpose.

Contrasting the left hand side and right hand side results in Fig. 5, we observe that the 9x9 patch-based exemplar inpaint has obvious artifacts in the filling region and disconnections along patterns that're supposed to be coherent. Our results exhibit much better consistency aligning with the unmasked region.

During the experimentation, we noticed that the Coherency Sensitive Hashing method's performance is highly

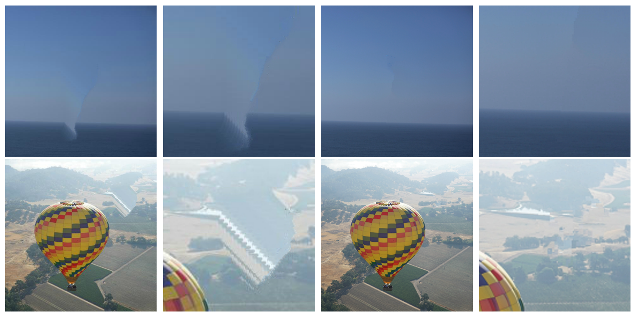
Figure 5. Left: 9x9 patch & enlarged, Right: ours & enlarged

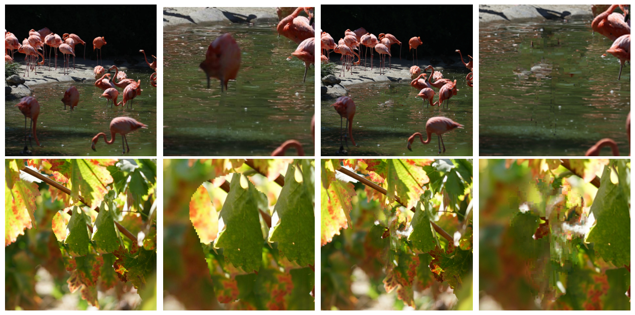
Figure 6. Left: CSH & enlarged, Right: ours & enlarged

correlated to the mask structure. When the masks are cropped with close bound on the object, then the algorithm has a tendency to leave remains of the original object in the scene, as we can see from the 'pond' results in Fig. 6 on the top left hand side. Our algorithm did not have this issue and correctly removes the entire animal from the scene, leaving the waves and reflections with only minimal artifacts. For the image on the bottom, the CSH results leaves significant boundary artifacts on the left side of the filling region, while our result presents a fully blended visual appearance.

4.4. Sparse Coding

We test our method on lena image with different type of defects using both a pre-defined dictionary and a KSVD learned dictionary. Note that the red masks in Figure 7 & 8 are only for visualization. They are black pixels in the degraded image. We visualize our learned and pre-defined dictionary in Figure 10. Figure 7 is reconstruction from text degraded image while Figure 8 is reconstruction from scratch image. Both pre-defined dictionary and KSVD learned dictionary achieved satisfying reconstruction quality. However, KSVD demonstrated better reconstruction ability for our task while DCT pre-defined dictionary reconstruction suffers from lower PSNR and patchy artefacts in some areas of the reconstructed images, which implies that KSVD learned dictionary is better at dealing with specific problems. But a pre-defined dictionary is simple and efficient and saves the trouble of finding a suitable training set for the task.

We also notice that the less the degraded areas are, the better reconstruction quality this algorithm will achieve. To test the limit of the reconstruction ability of our proposed method, we randomly pick 80% pixels in the original image and set it to zeros. We use our algorithm to recover this image and the result is shown in Figure 9. As can be observed, both DCT and KSVD reconstructed images begin to suffer from patchy artefacts while KSVD reconstructed image is slightly more smooth. However, the result is already stunning since even if we lost 80% of the information of the original image, we can still recover a visually satisfying result.

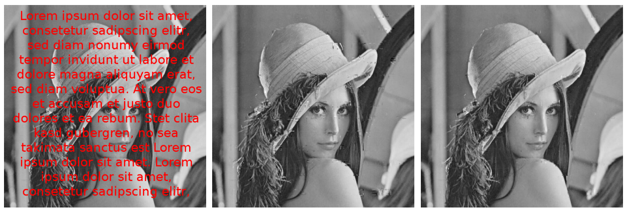

Figure 7. Left: Text degraded image, PSNR = 14.03dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 30.9dB Right: reconstruction using KSVD learned dictionary, PSNR = 33.09dB

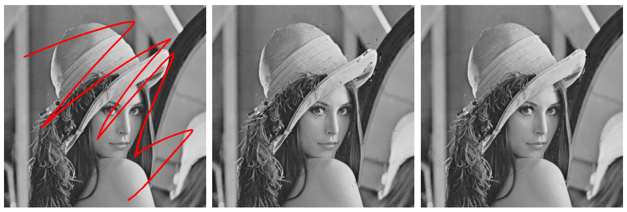

Figure 8. Left: Scratch degraded image, PSNR = 19.09dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 33.68dB Right: reconstruction using KSVD learned dictionary, PSNR = 35.94dB

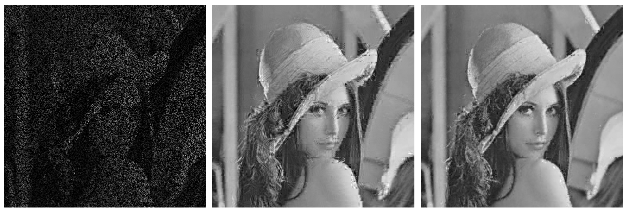

Figure 9. Left: Degraded image with 80% pixel loss, PSNR = 6.42dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 24.27dB Right: reconstruction using KSVD learned dictionary, PSNR = 26.75dB

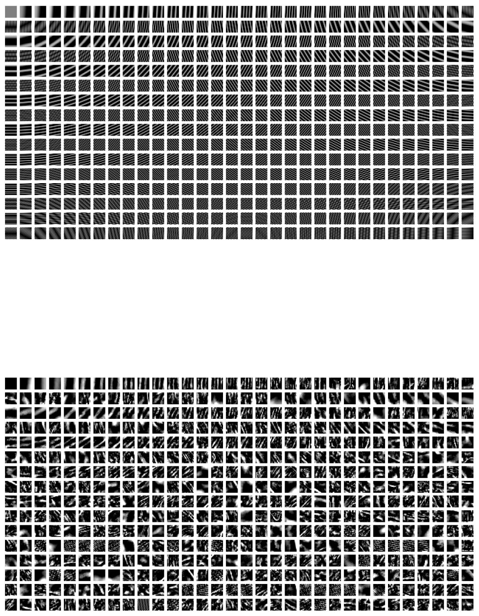

Figure 10. Up: over complete DCT coefficient dictionary Down: KSVD learned dictionary

5. Conclusion

In this project, we successfully realize two algorithms that realize high-quality object/defect removal. For single object removal, we proposed the priority-based inpainting with Non-Local-Means, contained within the single-image super-resolution pipeline, and post-process with unsharp masking. For defect/scratch removal, we implemented sparse coding based on both pre-defined and learned dictionary. The experimental results and the comparison with prior methods demonstrate the effectiveness in our algorithm in terms of both visual appearance and evaluation on Peak-Signal-to-Noise ratio.

6. Future Work

Currently, the two algorithms we worked on each resolves the object/defect removal problem limited to distinct image features. The Laplacian edge detection within the inpainting method is not effective on images with multiple small cut-out zones, while the sparse coding algorithm is not high-performing on images with large missing area. If time permits, we would like to investigate in possible ways to combine the two algorithms, making one that's applicable to both types of target images.

Furthermore, we realize through the experimental process that the runtime of our algorithm is slow. Optimizing for runtime by further examining our loop structures can be beneficial for processing high-resolution images with large sizes.

7. Acknowledgement

We would like to express our sincere gratitude towards Professor Gordon Wetzstein for directing us to possible research ideas and supplementing us with helpful resources. We are also very grateful of Felix and other course staff for their responses as we proceeded step by step in troubleshooting our algorithms. Finally, we would also like to thank everyone in the EE367 class for their genuine support, kindness, and feedback during the progress of our project.

References

[1] A.A. Efros, and W.T. Freeman. ”Image quilting for texture synthesis and transfer.” In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341-346. ACM, 2001.

[2] A. Talea. *”An Image Inpainting Technique Based on the Fase Marching Method.”* In Journal of Graphic Tools, vol. 9, pp.23-24. 2004.

[3] A. Criminisi, P. Perez, and K. Toyama. *”Object removal by exemplar-based inpainting.”* In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, vol. 2, pp. II-II. IEEE, 2003.

[4] M. Bertalmio, P. Perez, and K. Toyama. *”Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting.”* In Computer Vision and Pattern Recognition, 2001. Proceedings. 2001 IEEE Computer Society Conference on, vol. 1, pp. I355-I362. IEEE, 2003.

[5] J. Wu, and Q. Ruan. *”Object removal by cross isophotes exemplar-based inpainting.”* In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, vol. 3, pp. 810-813. IEEE, 2006.

[6] P. Viola, and M. Jones. *”Rapid object detection using a boosted cascade of simple features.”* In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, pp. I-I. IEEE, 2001.

[7] J. Herling, and W. Broll. *”Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments.”* In Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on, pp. 207-212. IEEE, 2010.

[8] O. Le Meur, and C. Guillemot. ”Super-resolution-based inpainting.” In European Conference on Computer Vision, pp. 554-567. Springer Berlin Heidelberg, 2012.

[9] M. Ashikhmin. Synthesizing natural textures. In Proc. ACM Symp. on Interactive 3D Graphics, pp. 217226, Research Triangle Park, NC, Mar 2001.

[10] J. Mairal, M. Elad, and G. Sapiro. ”Sparse representation for color image restoration.” IEEE Transactions on image processing 17, no. 1 (2008): 53-69.

[11] Dictionary Learning Tools for Matlab, `http://www.ux.uis.no/~karlsk/dle/`

[12] Part of the source code referenced from: http://scarlet.stanford.edu/teach/index.php/Object_Removal

[13] S. Korman, and S. Avidan. ”Coherency sensitive hashing.” In Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 1607-1614. IEEE, 2011.

[14] M. Aharon, M. Elad, and A. Bruckstein. ”k-SVD: An algorithm for designing overcomplete dictionaries for sparse representation.” IEEE Transactions on signal processing 54, no. 11 (2006): 4311-4322.

[15] D. Glasner, S. Bagon and M. Irani. ”Super-Resolution From a Single Image.” In Computer Vision (ICCV), 2009 IEEE 12th International Conference on, pp. 349-356. IEEE, 2009

[16] J. Liu, W. Li, and Y. Tian. ”Automatic thresholding of gray-level pictures using two-dimension Otsu method.” In Circuits and Systems, 1991. Conference Proceedings, China., 1991 International Conference on, pp. 325-327. IEEE, 1991.

[17] Mallat, Stphane G., and Zhifeng Zhang. ”Matching pursuits with time-frequency dictionaries.” IEEE Transactions on signal processing 41, no. 12 (1993): 3397-3415.

[18] Rubinstein, Ron, Alfred M. Bruckstein, and Michael Elad. ”Dictionaries for sparse representation modeling.” Proceedings of the IEEE 98, no. 6 (2010): 1045-1057.

[19] Huang, Ke, and Selin Aviyente. ”Sparse representation for signal classification.” NIPS. Vol. 19. 2006.

[20] Vinje, William E., and Jack L. Gallant. ”Sparse coding and decorrelation in primary visual cortex during natural vision.” Science 287, no. 5456 (2000): 1273-1276.