End-to-End Scene Restoration and Object Removal: From Detection to Inpainting

Haihong Li
Department of Electrical Engineering
Stanford University
hhli@stanford.edu

Chi Zhang
Department of Mechanical Engineering
Stanford University
czhang94@stanford.edu

Chen Zhu
Department of Electrical Engineering
Stanford University
chen0908@stanford.edu

Abstract

This is the final project report for CS231A at Stanford University, Spring 2017. The goal of this project is to combine object detection and image inpainting, creating an end-to-end scene restoration pipeline. This paper first summarizes related methods of object detection and inpainting, then discusses our attempt to improve existing methods. This paper further lays out implementation details of our methods and in the end discusses experimental results of different methods.

1. Introduction

1.1. Motivation

Are you annoyed when a John Doe sneaks into your photo of Yosemite and you do not know how to Photoshop?

Scene restoration has been studied intensively over the years. Usually this practice is used to remove unwanted objects(an intruding person etc.) or defects(scratches, dust spots, etc.) in the image. To remove unwanted areas and then fill those areas without prior knowledge about the original scene is called inpainting.

In most cases, this practice requires a binary mask (a grayscale image) to indicate where inpainting should be performed. Traditionally, this mask is cropped manually, which is tedious and time-consuming. However, the emergence of object detection algorithm makes automated detecting unwanted object possible. Two of the most famous algorithm (also discussed in class) are Scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG). These methods are computationally simple, both for training and detection. Besides, there have been attempts to detecting object using convolution neural networks[13][11], which can achieve more accurate and finer detection and segmentation. However, those methods require long training time and are of very high computational cost that normal personal computers cannot afford .

1.2. Objective

For the project, we would like to design an end-to-end pipeline incorporating object detection and inpainting for scene restoration. For image detection, for computation complexity concern, we will use HOG feature descriptor to draw a bounding box around detected figure. With this bounding box, we will create a binary mask for the area to be inpainted and then feed to the next step of our pipeline. For image inpainting, we will implement exemplar-based image inpainting proposed by Criminisi et al. [5], then seek potential to make improvements to the existing algorithm. When we were implementing the exemplar-based inpainting, we also read about another interesting inpainting method, which is sparse coding based image inpainting[1]. This method is not a part of our scene restoration pipeline, but we will also briefly explain its dynamics and then implement it and then evaluate its performance.

2. Related Work

Scene restoration has received intensive attention over the years. There are mainly two types of algorithm. The first algorithm is texture synthesis, which is to recover degraded area with its structural content information. Efros et al. proposed to predicts unknown textures by repetition of two-dimensional textural patterns. They build a Markov random field model and utilize a probability table to obtain the probability distribution of a query point p in the sampled image, and then fills the point with the pixel value whose probability is the greatest[7].

Another common method is inpainting, which is inspired by partial differential equations. It fills the missing area by propagating linear structure (called isophotes) and then interpolating them with weighted sum of their known neighbor pixels. Ballester *et al.*. proposed to interpolate missing data by solving the variational problem via its gradient descent flow. The basic idea is to smoothly propagate known pixels into unknown areas based on image gradient and their corresponding pixel values.[2]. Criminisi *et al.* proposed a new way to propagate isophotes by adding a confidence term to decide the filling sequence. They define a confident pixel as a pixel near isophete and with more known neighbors. More confident pixel will be filled earlier[5].

However, the aforementioned methods all require user segment the deteriorated region manually. We want to extends present inpainting methods by adding automated region labeling. Objection detection is one of the most essential area of computer vision. One of the most famous object detection algorithm is Scale-invariant feature transform (SIFT), which utilizes the information of local gradient directions of image intensities to match targeted area with certain object. SIFT is invariant to translation, scaling and rotation so it is especially robust to match 2D objects without major perspective changes[10]. HOG[6] feature descriptor is another common approach to detect object. It is originally proposed to detect human objects. For this project, we mainly focus on removing unwanted person in the image, so HOG is adopted to perform object detection.

3. Technical Approach

3.1. Histogram of Oriented Gradients

Histogram of Oriented Gradients(HOG) has been one of the most popular and successful person detectors since it was proposed in 2005. HOG is a type of feature descriptor, of which the purpose to generalize the object in such a way that the same object (in this case a person) produces as close as possible to the same feature descriptor when viewed under different conditions. This makes classification much easier.

The authors of HOG trained a Support Vector Machine, *i.e.* SVM, which is a kind of machine learning algorithm for classification to recognize HOG descriptors of people.

Compared to SIFT, HOG person detector is relatively simple to understand. One of main reasons is that it uses a *global* feature to describe a person rather than a collection of *local* features. To put in another way, this means that HOG represents the entire person object by a single feature vector, as opposed to many feature vectors representing smaller parts of the person.

The HOG person detector uses a “sliding window” approach: at each position of the detector window, a HOG descriptor is computed for the window. Then this descriptor is shown to the pre-trained SVM, which is used for classification. For the purpose of recognizing persons at different scales, the image can be subsampled to multiple sizes. Each of these subsampled images will be searched.

Since training HOG and linear SVM classifier is not the center of this project, the general algorithm of training will not be discussed here.

3.2. Inpainting

1. Exemplar-based inpainting
   We first implemented exemplar-based region inpainting[5]. One crucial problem for inpainting is to decide filling orders of pixels in the area to be inpainted. Different filling orders may lead to different inpainting results[5]. Criminisi et al. proposed to decide the filling order by assigning scores to each pixel in the deteriorated region. The pixel with the highest score gets inpainted first. They take two factors into consideration. The first factor is the confidence term, which describes how much a pixel is in the source area. The second factor is called the data term. It describes how close a pixel is to an isophote. The term *isophote* is wildly used in the inpainting literature. It is defined as the contour of equal luminance and chrominance in an image. Criminisi et al. thinks a pixel with the most known neighbors and closest to isophote are of highest score. The way they calculate filling priority of pixel p can be described using the the following equations.

   $$P(p) = C(p)D(p) \tag{1}$$

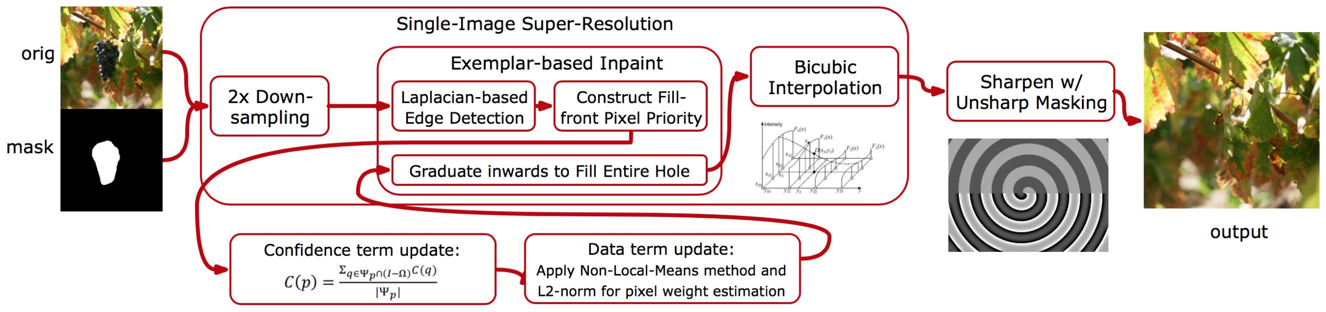
   where
   $$C(p) = \frac{\sum_{q \in \Phi_p \cap (I-\Omega)} C(q)}{|\Phi(p)|}$$

   which denotes how many known pixels are around pixel p, and
   $$D(p) = \frac{|\nabla I_p^{\perp} \cdot n_p|}{\alpha}$$

   which denotes how close a pixel p is to isophote in known regions.

   After they find the pixel p with highest priority, they search the most similar patch from known regions to fill the unknown region centered by pixel p. The scheme to evaluate similarity here is simply the sum of square difference of already filled pixels in two patches.

   $$\Psi_{\widehat{q}} = \arg \max_{\Psi_q \in \Phi} sim(\Psi_{\widehat{q}}, \Psi_q) \tag{2}$$



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:
$C(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi_p|}$
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.

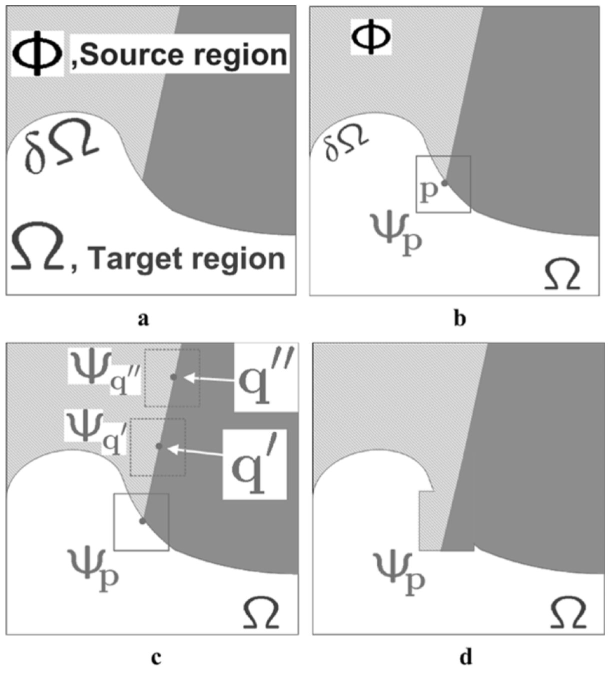


Figure 2: Illustration of examplar based inpainting: a)We denote source region (known region) with Φ, the target region to be filled with Ω, and fill front with δΩ. b)&c) The most likely candidate to be filled is pixel with most known neighbors and closest to the boundary of two textures in the source region. d) We fill the region centered by pixel p with the most similar patch in the source region. Note we only keep p's pixel value.

Note that only the pixel value of p is preserved while other values in the unknown area are discarded. Then p is considered a known pixel and added to the source region. This algorithm iteratively fill regions in the fill front until all pixels in the unkwown regions are filled. Figure 1 better illustrates how the algorithm works and all the notation used here.

2. Super Resolution based inpainting

   Beside implementing existing methods in inpainting literature, we also seek the potential to make improvements to those algorithms. Our first attempt is to combine super resolution with exemplar-based region filling, which is inspired by Le et al's work[9]. This method improves exemplar-based inpainting by down-sampling the image before inpainting. After down-sampling, we implement exemplar-based inpainting on down-sampled coarse image. This pre-processing step preserves the dominant structure of the original image, so it helps prevent singular patches or even noise from exerting influence on region filling. One thing worth our notice is that during the down-sampling step, alias might appear especially in the high frequency areas of the image if the sampling frequency is less then Nyquist frequency. To avoid this, we apply an anti-aliasing filtering, which is a Gaussian low pass filter, to the original image. This filtering step not only prevent alias from appearing, but also helps remove noise, generating a smoother result.

   After we finished inpainting of down-sampled course images, we need to combine images of low resolution back to one single high resolution image. Here we use a single-image super resolution method. There have been various imaging literature on restoring the high resolution image, like one proposed by Glasner et al. in 2009 [8]. However, we decide to use classic bicubic interpolation method due to computation complexity concern.

3. Unsharp masking

   As a post-processing step of super-resolution based inpainting, we apply unsharp masking[4] to enhance the high frequency component of the image and achieve sharpened image. To enhance the high frequency component, we first need to find high frequency component. For high-pass filtering, it's not easy to form a "soft" cutoff frequency like how we do it for low-pass filtering(gaussian filter). However, a hard cutoff frequency may lead to ringing effect. To enhance the image without ringing effects, unsharp masking comes into place. 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. The overall process of inpainting can be illustrated in the flowchart in Figure 1.

4. Experiment with different confidence terms
   To decide pixel priority is crucial to inpainting. Different inpainting sequence may lead to completely different results[5]. Thus it is worth our effort to exploring different approach to compute region priority. The exemplar-based inpainting utilizes a confidence term $C(p)$ to describe how much a pixel in fill front is in the source region, and a data term to describe how close it is to the boundary of two textures in the source region (how close this pixel is to a gradient dense area). Here instead of using gradient-based data term, we propose to utilize non-local mean to decide data confidence. The intuition is based on the following observations[15]. First, structures are sparsely distributed while textures are less sparesely distributed. Second, neighbor regions with high similarity are more likely to have the same structure and texture layout to the patch to be filled. So we model the data term as follows.

$$D_{ij} = \| \frac{1}{Z(i)} exp(-\frac{\sum_{mn}(k_{mn}v((N_i)_{mn} - (N_j)_{mn}))_2^2}{h^2}) \|_2 \tag{3}$$

   where

$$Z(i) = \sum_j exp(-\frac{\|v(N_i) - v(N_i)\|_2^2}{h^2}) \tag{4}$$

   Here $v(\cdot)$ denotes the i-th pixel value in original image[3].

3.3. Sparse coding based image inpainting

Computationally, the problem of inpainting can be modeled as follow

$$\|y - Dx\|_p \leq \epsilon$$

where $y \in R^n$ is the vectorized undegraded image, $D \in R^{n \times k}$, $k > n$ denotes an over-complete dictionary, and $x \in R^k$ is the coefficients of the image y.

Since $k > n$ and D is required to be full rank matrix, this problem is an under-determined problem. There are infinite many solutions to the equation, and we choose the sparsest one as the solution we want. However, finding the sparsest solution to the above equation is an NP-hard problem. Thus, pursuit algorithms to approximate the true solution comes into place[1]. Three pursuit algorithms are wildly used in solving sparse coding problem, which is basis pursuit(BP), matching pursuit(MP), and orthogonal matching pursuit(OMP). For performance as well as efficiency, we adopt OMP in the following steps.

Algorithm 1 Orthogonal Matching Pursuit

```
Input: Signal $y \in R^n$, dictionary $D \in R^{n \times k}$
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 $l_k$
- $x_k \leftarrow argmin_x \|y - Dx\|$
-$r_k = y - D_k x_k$
```

In this problem, both x and D are unknown. We can find x using any pursuit algorithm as discussed before. The next step is to discover an over-complete dictionary D to best represent the signal to be recovered. A simple way is to use 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 are not adaptive to the changes of signals. 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. The algorithm is described in Algorithm 2 [1]. We will evaluate the performance of different dictionaries in the following sections.

4. Experiment

4.1. Implementation Details

4.1.1 Person Detection via HOG

We used OpenCV to implement HOG feature descriptor and adopt its pre-trained SVM classifier to perform person detection. We created a HOG object via `cv2.HOGDescriptor()`. To enable its capability of classification, a SVM classifier should be attached to this object. There exist three different types of detectors in OpenCV with different sizes, and we choose the default one.

Algorithm 2 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 $\Sigma(1,1)V(:,1)$
```

4.1.2 Inpainting

To evaluate the performance of different inpainting algorithms, we use images from not only standard inpainting scholar articles, but also photos from our own photo libraries taken by commercial cameras of phones.

For this stage of evaluation, we used hand-cropped masks, since we our focus is on various inpainting algorithm instead of the whole scene restoration pipeline.

4.1.3 Sparse coding based inpainting

To train a dictionary, we use 4 images with human faces from USCimages[1]. We didn't include the lena image in the training to avoid bias.

4.2. Results

4.2.1 Results for object detection

We tested our trained HOG feature descriptors on both images from the Internet and our own photo libraries. The results are presented in Figure 3.

[1]http://www.ux.uis.no/~karlsk/dle/

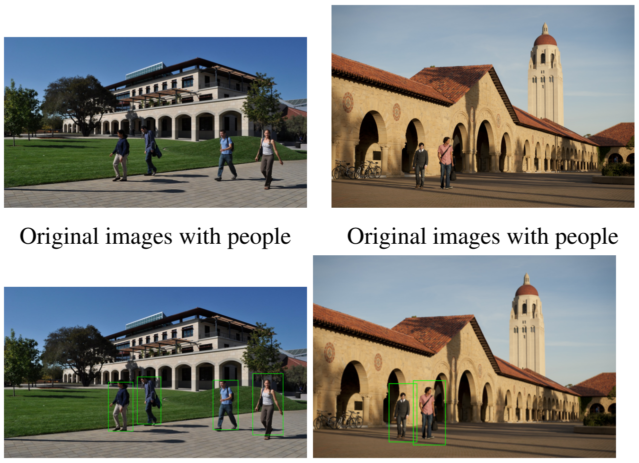


Figure 3: HOG detection results

We can see in Figure 3 that all people in the images are detected by HOG. One thing worth notice is that the shadow of the people isn't circled by the bounding boxes, which may cause some artefacts.This will be discussed in the following sections.

4.2.2 Results for inpainting

1. Exemplar-based inpainting result
   We tested the exemplar-based image inpainting both with image inpainting datasets and our own photos. Figure 6 shows the results. As can be seen, exemplar-based method's reconstruction are visually satisfying, although artefacts still exist (e.g. the "intrusive" green pattern in the roof in the Bungee reconstruction result).

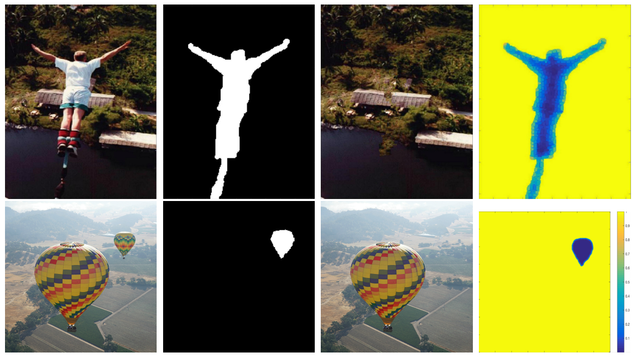


Figure 4: a) original, b) mask, c) exemplar-based results, d) confidence plot.

2. Single image super resolution based inpainting
   We present our results for SR-based image inpainting

in Figure 5. We can see that for this set of images, SR-based inpainting can achieve artefact-free results.

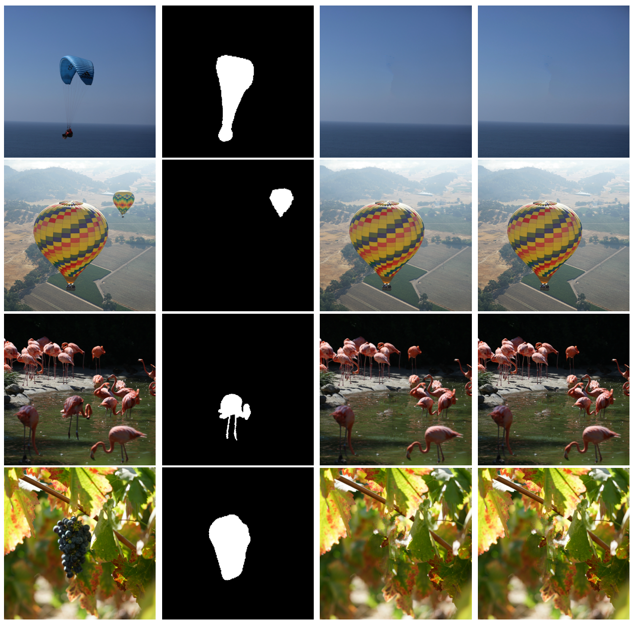


Figure 5: a) original image, b) mask, c) post-inpaint, d) SR and sharpened final image

3. NLM based image inpainting
   Results of NLM based image inpainting are presented in parallel with exemplar-based inpainting in figure 6 and 7. Figure 6 shows a comparison of the reconstruction results between NLM and exemplar based inpainting, while figure 7 shows the difference between their confidence and data terms. Two algorithm all achieve satisfying results, while NLM-based inpainting are slightly better in the reconstruction of Bungee image. Less artefacts can be observed on the NLM-based reconstruction results.

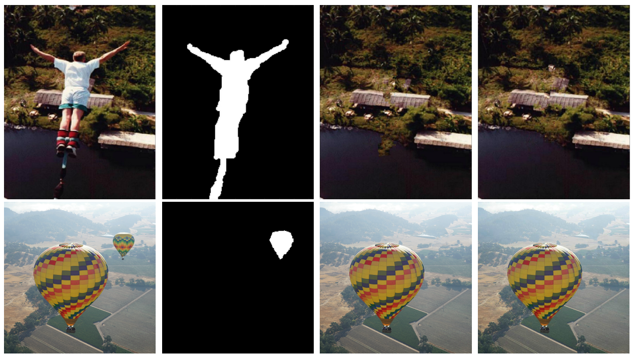


Figure 6: a) original img, b) mask, c) Exemplar-based method, d) NLM.

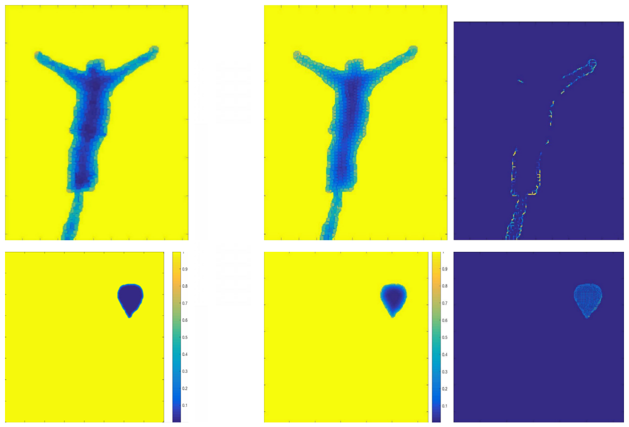


Figure 7: a) Exemplara-based confidence plot, b) NLM's confidence plot, c) NLM's similarity plot.

4.2.3 Creating the pipeline

Once we have explored the performance of our proposed object detection and inpainting approaches, we want to show the results of combining the two component to an end-to-end scene restoration pipeline. Figure 8 illustrate each component in the pipeline.

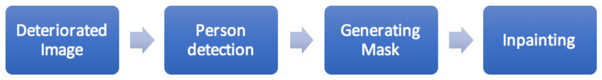


Figure 8: Illustration of the end-to-end pipeline

We present more scene restoration results in Figure 9. We can see that all people in the input images are successfully detected by HOG, and our proposed inpainting algorithms successfully filled the region specified by the generated mask. Very few artefacts can be observed.

4.3. Results for sparse coding based inpainting

4.4. Sparse Coding

The results of sparse coding based inpainting is shown in Figure 10, Figure 11 and Figure 12. We presented the results produced by both pre-defined dictionary and KSVD learned dictionary on different types of defects in parallel. We can see that KSVD achieved results of better quality with regards to peak signal to noise ratio, which is defined as follow:

$$PSNR = 10 \log_{10} \left(\frac{MAX_i^2}{MSE} \right)$$

where MAX_i denotes the maximum possible pixel value of the image, and MSE denotes the mean square error between the reconstructed image and the original noise-free image.

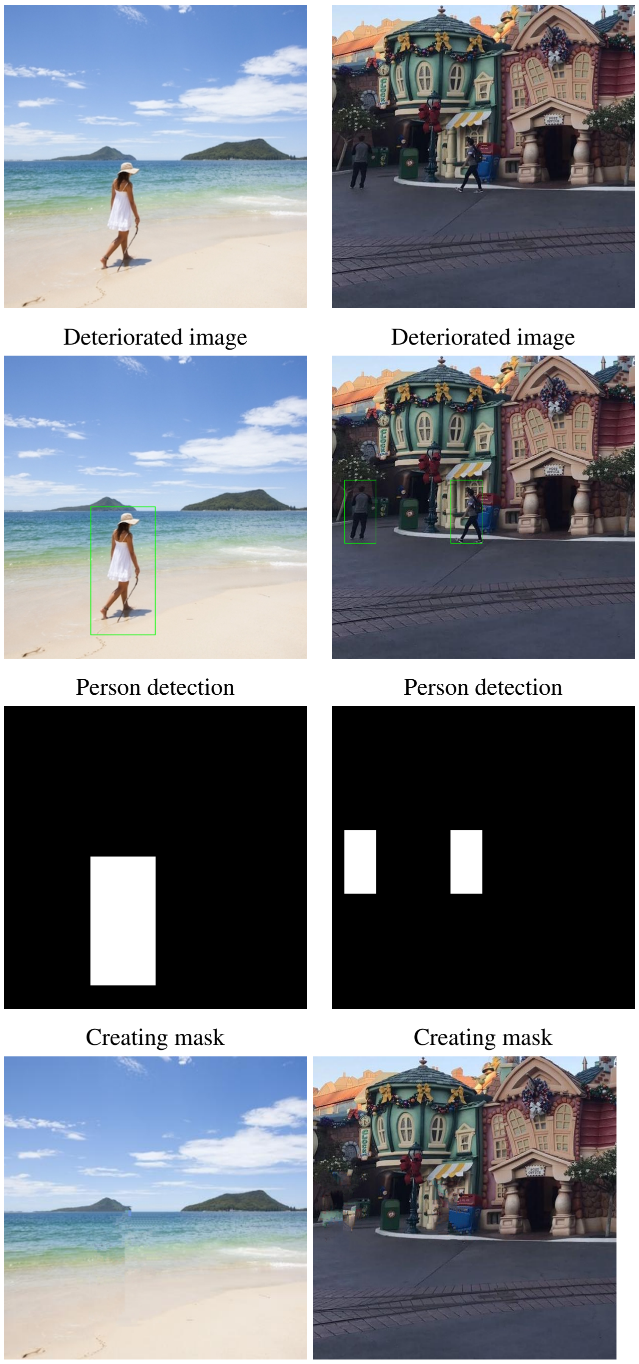


Figure 9: Results of end-to-end scene restoration pipeline

We notice that the less the defects are, the better the reconstruction quality the algorithm will produce. To test the limit of this algorithm, we randomly choose 80% pixels and treat them as the unknown areas and then use our algorithm to reconstruct this image. The results are shown in Figure 12. We can see that this algorithm is still able to reconstruct the image with satisfying results even if 80% of the information is lost.

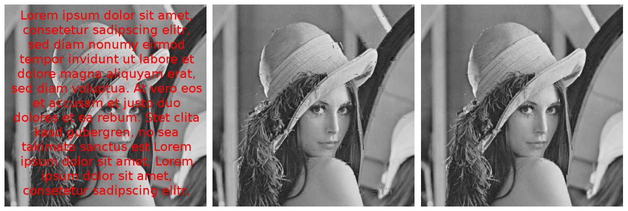


Figure 10: Left: Text degraded image, PSNR = 14.03dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 30.9dB Right: reconstruction using KSVD learned dictionary, PSNR = 33.09dB

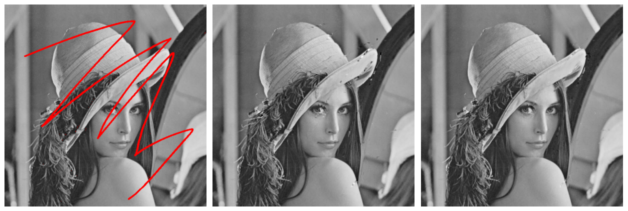


Figure 11: Left: Scratch degraded image, PSNR = 19.09dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 33.68dB Right: reconstruction using KSVD learned dictionary, PSNR = 35.94dB

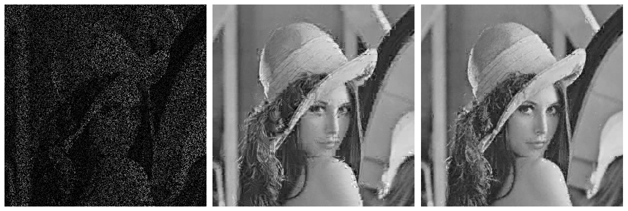


Figure 12: Left: Degraded image with 80% pixel loss, PSNR = 6.42dB Middle: Reconstruction using pre-defined DCT dictionary,PSNR = 24.27dB Right: reconstruction using KSVD learned dictionary, PSNR = 26.75dB

5. Discussion and Conclusion

5.1. Discussion

Our proposed scene restoration generally works well regarding to detecting people and inpainting. However, in some situations it might fail. Figure 13 and Figure 14 shows some images that our algorithm doesn't perform well on.

Figure 13 illustrate one possible situation our proposed pipeline might fail, which is the existence of human shadows. HOG feature descriptors are not able to detect shadows, so the shadows cannot be included in the bounding as wanted, causing a ghost effect in the inpainted results.

As can be seen from Figure 14, if the background pattern is too repetitive, the inpainting errors will be magnified. Another possible reason for this failure might be the bounding box being too large, which gives more challenges to the inpainting step. In this situation, we can adopt other patch matching algorithm since we have prior knowledge of the scene. Another possible solution is to try some finer segmentation methods, like deepmask proposed by Facebook in 2015[12].

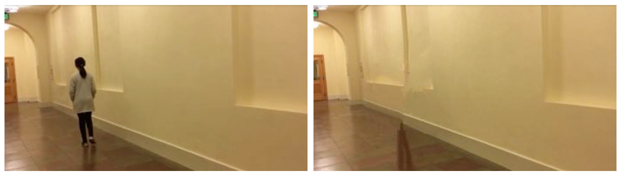


Figure 13: Shadows creating ghost effects

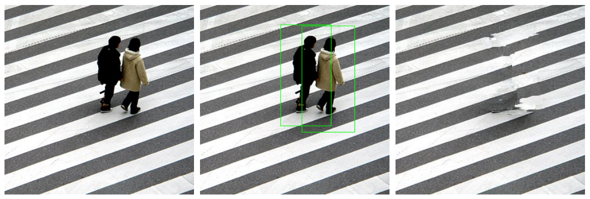


Figure 14: Repetitive background pattern magnifies inpainting errors

5.2. Conclusion

Our goal for this project is to create an end-to-end scene restoration pipeline. We adopt HOG feature descriptor for the detection stage, and proposed an improved exemplar-based image inpainting algrithm for inpainting stage. In section 4.2.3, we can see that the proposed pipeline works well. Besides creating the pipeline, we also studied and implemented sparse coding based image inpainting algorithm. Future work should focus on more robust HOG detection and finer image segmentation.

6. Acknowledgement

We would like to thank Professor Silvio Savarese for lecturing the course and supplementing us with helpful resources. We are also very grateful of Trevor Standley and other course staff for their responses as we step into trouble doing our project. Finally, we would also like to thank everyone in the 231A class for their support, kindness, and feedback during the whole quarter.

7. Appendix

For all codes and video demos of this project, please visit `https://github.com/Leedehai/cs231a_inpaint.git`

References

[1] M. Aharon, M. Elad, and A. Bruckstein. rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] C. Ballester, V. Caselles, J. Verdera, M. Bertalmio, and G. Sapiro. A variational model for filling-in gray level and color images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 10–16. IEEE, 2001.

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[4] H.-p. Chan, C. J. Vyborny, H. MacMAHON, C. E. Metz, K. Doi, and E. A. Sickles. Digital mammography: Roc studies of the effects of pixel size and unsharp-mask filtering on the detection of subtle microcalcifications. *Investigative Radiology*, 22(7):581–589, 1987.

[5] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[7] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 349–356. IEEE, 2009.

[9] O. Le Meur and C. Guillemot. Super-resolution-based inpainting. In *European Conference on Computer Vision*, pages 554–567. Springer, 2012.

[10] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[11] P. H. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. *CoRR*, abs/1506.06204, 2015.

[12] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[13] T. D. R. Girshick, J. Donahue and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition(CVPR)*. IEEE, 2014.

[14] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[15] Z. Xu and J. Sun. Image inpainting by patch propagation using patch sparsity. *IEEE transactions on image processing*, 19(5):1153–1165, 2010.