Image Deblurring

Seongmin Oh Email: soh44@wisc.edu ID: 906 839 1094
Jialuo Gao Email: jgao223@wisc.edu ID: 907 625 3195

Abstract Summary

In this article, the student will explore the method of solving the least square problem used to deblur images with known blur filters. Starting with how to blur the image with filters by using the process of 2D convolution to even how to mimic a real-world blurred image created by motion or camera defocus. Then, this project will guide the student to the details of how to recover the image from the blurred image by solving the least square problem. On top of that, the student will explore how adding noise to the image can affect the performance of this method. The student will also learn how two major methods we learned during lecture, SVD low-rank approximation and Tikhonov Regularization, can help to improve the performance of the deblurring process when there is noise in the image. Finally, the student will be experimenting with some other limitations of the deblurring method by recovering the sharpened image and the edge detected image.

GOALS:

1. **Blur images with various 2D filters**
2. **Deblur images by solving Least Square Problem**
3. **The effects of the noise on the above method**
4. **Ways to limit the effects of the noise**
5. **Other limitations of the method**

Note: This exercise requires ***heavy computation***(don't run if you are not confident with your computing power) and using the ***online version*** of Matlab is recommended for better experiences (try running the exercise ***section by section*** to eliminate potential crashes). In addition, a relatively ***smaller size image*** is used to reduce the runtime; try not to increase the image size if you are not confident with your computing power. link to online Matlab: https://matlab.mathworks.com/

Background

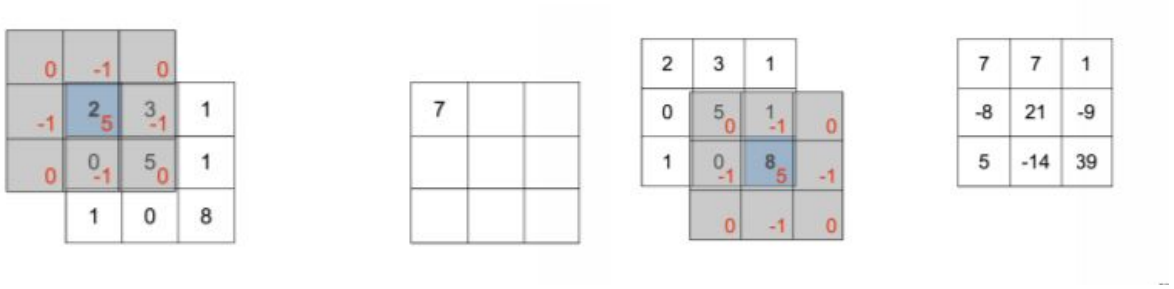
Blurring due to the motion of the object, the shakiness of the sensors, and defocus of the camera can result in disappointing poor quality images, as well as inaccurate representation of the observation for scientific researches that requires precise images such as astronomy or biology. There are many successful algorithms in the field that can help reduce the impact of the blur; However, none of them are perfect by any means, and we still need researches on how to improve existing algorithms or creating new methods. This project presents a way to restore the image from blurring based on methods we have learned from the class such as solving Least Square Problems and reducing the effects of noise using SVD Low-Rank Approximation and Tikhonov Regularization.

2D convolution (Image Filtering)

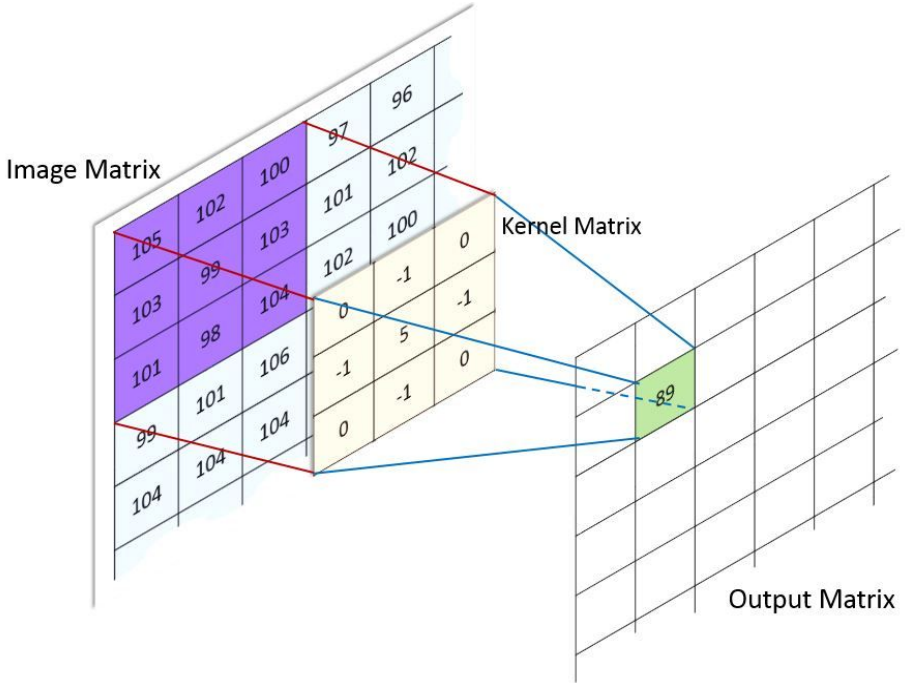
2D convolution is a process of changing the value of the base matrix (B) by using a convolution matrix (C). During the process, each pixel of the result matrix (R) is equal to a weighted sum of the pixels around the pixel where the convolution matrix contains all the weights and the original image donates the value.

$$B_{(x \text{ by } y)} * C_{(m \text{ by } n)} = R_{(x \text{ by } y)}$$

$$b_{j,k} = b_{j-(m/2),k-(n/2)} \bullet c_{1,1} + \ldots + b_{j,k} \bullet c_{m/2,n/2} + \ldots + b_{j+(m/2),k+(n/2)} \bullet c_{m,n}$$



(Note: You may click the link down in the reference section for a GIF image of the full demonstration or the Stanford article for more details. click here)



2D convolution can be done by using the function "conv2" in Matlab. You can call the function by "blurred_image = conv2(Image, filter, shape)". This function will perform 2D convolution on the "Image" (base matrix) by using the "filter" as the convolution mask (convolution matrix). For the "shape" filed, we used 'same' to obtain the same size as the "Image" which gives us the central part of the convolution.

Code:

```
blurred_img = conv2(image, filter, 'same');
figure, imshow(blurred_img);
title('Blurred Image')
```

2D filters

Usually, many filters can be calculated if the motion of the scene is predictable. Thus, we are assuming that we know the motion filter (convolution mask) of the blur beforehand and apply the filter to the image to generate a blurred image. There are various predefined 2D filters can be generated using the Matlab function "fspecial". You can simply type "name_of_filter = fspecial('type of filter', [corresponding value(s)])". In this final project, we are going to present 3 filters.

1. Gaussian Filter

Gaussian Filter is a widely used filter typically to reduce image noise and details; a Gaussian filter is also often used to create the effect of a defocused image. This can be obtained by "gas = fspecial('gaussian', hsize, sigma)". "hsize" is the size of the filter. For example, if you have hsize = 5 then this function will return a convolution mask with a size of 5x5. Sigma is the standard deviation of the Gaussian bell curve. This defines whether the filter is wide(more blurred) or narrow(less blurred).

Code:

Creating Gaussian filter

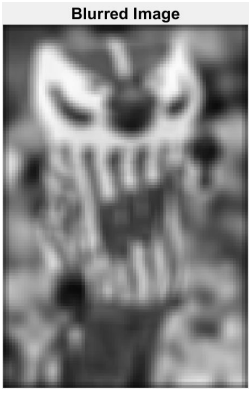
```
% Gaussian filter
hsize = 5;
sigma = 5;
gas = fspecial('gaussian',hsize,sigma);
```

Applying 2D convolution

```
%% blurring using Gaussian filter
filter = gas;
blurred_img = conv2(image, filter, 'same');
```

Result:

Original **Gaussian**



2. Motion Filter

Motion Filter is used to approximating the linear motion of a camera. To create this filter, simply type "mtn = fspecial('motion', len , theta)". len is the size of the motion filter. The unit is in pixels. theta is the angle of the motion filter in counterclockwise. For example, mtn = fspecial('motion', 10, 20) will give you a distance of 10 pixels with 120-degree motion filter.

Code:

```
% Motion filter (Weiner)
len = 10;
theta = 20;
mtn = fspecial('motion', len, theta);

%% blurring using Gaussian filter
filter = mtn;
blurred_img = conv2(image, filter, 'same');
```

3. Averaging Filter

This filter takes the average of all the pixels within a specified size of the filter.

Averaging Filter Example

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

This is one of the examples of an averaging filter.

To create this filter, simply type "avg = fspecial('average', hsize)". hsize is the filter size where avg is a square matrix.

Code:

```
% Averaging filter
hsize = 5;
avg = fspecial('average',hsize);

%% blurring using Gaussian filter
filter = avg;
blurred_img = conv2(image, filter, 'same');
```

Warm-up

1. Why do we care about image deblurring?
2. List all the methods mentioned above that we have learned in class.
3. What should you do if your computer is slow or can't run the code provided?
4. Perform a 2D convolution on the base matrix with the given convolution matrix. Show your steps and check your answer by using the Matlab function.

convolution matrix

0	-1	0
-1	4	-1
0	-1	0

base matrix

4	2	0
-2	1	3
5	6	-1

5. Create Motion filter of size 10 with degree 20 and apply to the provided image "Bucky.jpg" using the function "fspecial". Try more filters and values if you think it will help to enhance your understanding (Type "help fspecial" in the command window for more help).

Deblurring By Solving Least Square

For the purpose of this final project, we provided "Least Square Problem" to solve the deblurring problem. The process is simple. Obtain full convolution matrix of the blurred image by using the Matlab method "convmtx2(filter, image_size)", solve the Least Square Problem, and multiply the result matrix to the blurred image to obtain a restored image. The Full Convolution Matrix is constructed so that it is ***totally equivalent*** to perform 2D convolution on the image by using "filter" as the mask. And therefore it takes a huge amount of memory to store it; it can easily reach over 10k by 10k double values.

Main Idea:

Construct the full Convolution Matrix and set up the equation

Blurred Image = Full Convolution Matrix * Original Image

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \\ b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \begin{bmatrix} p_{22} & p_{12} & & p_{21} & p_{11} & & & & \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & & & \\ & p_{32} & p_{22} & & p_{31} & p_{21} & & & \\ p_{23} & p_{13} & & p_{22} & p_{12} & & p_{21} & p_{11} & \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ & p_{33} & p_{23} & & p_{32} & p_{22} & & p_{31} & p_{21} \\ & & & p_{23} & p_{13} & & p_{22} & p_{12} & \\ & & & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ & & & & p_{33} & p_{23} & & p_{32} & p_{22} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}$$

Matrix B is all pixels of the Blurred Image in one column; Matrix X is Original Image in one column. For the full convolution matrix P, each row of the matrix representing a convolution to the corresponding pixels on the Original Image. Where

$b_{22} = p_{33}x_{11} + p_{23}x_{21} + p_{13}x_{31} + p_{32}x_{12} + p_{22}x_{22} + p_{12}x_{32} + p_{31}x_{13} + p_{21}x_{23} + p_{11}x_{33}$

b_{22} is middle row of P matrix * X (p_{ij} is zero if outside of the filter dimension)

Solve Least Square Problem

Restored Image = $((P^T * P)^{-1} * P^T)^T$* Blurred Image

(P is the full convolution matrix, the extra transpose is to ensure dimension matches)

$$\begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix} = \left(\left(\mathbf{P}^{\top} \times \mathbf{P}\right)^{-1}\right) \times \mathbf{P}^{\top} \times \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \\ b_{13} \\ b_{23} \\ b_{33} \end{bmatrix}$$

Code:

```
%% deblurring by solving least square without noise
P = convmtx2(filter, size(image));
A = (P'*P)\P';
deblurred_img = reshape(A' * blurred_img(:), size(filter)+size(image)-1);
```

Exercise 1: Try to blur the image with all three given filters

- Gaussian
- Motion
- Average

and then deblur the image. (Note: Try not to use too large of a filter for better result and to reduce runtime).

1. Comment on the performance of the deblurring method on each filter.
2. Is there any significant difference between the result for different filters? Explain.

Deblurring images with noise

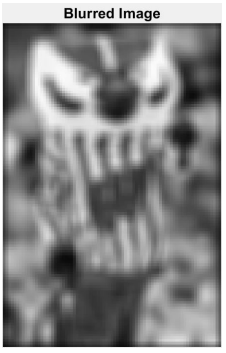
Now, we are adding noise to the blurred image to mimic a more realistic image which often includes a noise created by the camera sensor. This process can be helpful to understand the effects of the noise on solving the least square problem, and why deblurring in real-life can be very difficult. We are going to simply add uniformly distributed randomly generated noise for simplicity. As you can see, it is hard to tell the difference between the blurred image with or without the noise due to the small amplitude of noise.

Code:

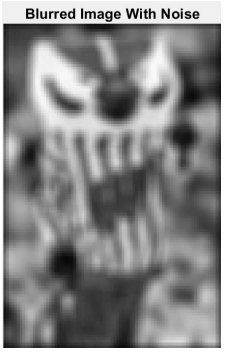
```
%% add noise to the blurred image
noise = rand(size(image))*0.01;
blurred_img = blurred_img - noise;
```

Original

Gaussian



Gaussian with noise



Exercise 2: Try to blur and then add noise before deblurring the image using a Gaussian filter with the code provided. Comment on the performance of the method to recover blurs with noise. Repeat the above instructions for different noise level: 0.01, 0.001, and 0.1 (try more if it helps to improve your understanding)

1. Explain the effects of different levels of noise on the recovered result.
2. Is it still fully recoverable with a noisy image using the least square method?
3. Explain why is this the case using what we learned in the class about solving least square.

Ways to limit the effects of noise

SVD Low-Rank Approximation

We can use SVD low-rank approximation to remove some of the noise from the blurred image before solving the least square problem and get a less noisy result.

Code:

```
r = 25;
[U,S,V] = svd(blurred_img);
U = U(:,1:r);
S = S(1:r,1:r);
V = V(:,1:r);
blurred_img_svd = U*S*V';
```

LS with Tikhonov Regularization

We can also use Tikhonov Regularization when solving the least square problem which also can help us reduce the effects of the noise.

Code:

```
A = (T'*T + 0.005*eye(size(T,2),size(T,2)))\T';
deblurred_img = reshape(A' * blurred_img(:), size(filter)+size(image)-1);
```

Exercise 3: Use both methods mentioned above (SVD and Tikhonov) to deblur the image.

1. Comment on the performance of both methods in comparison to only using LS.
2. Can any of these two methods produce a result that matches the quality of the LS result of the image without noise?
3. Compare and contrast the result of both methods(SVD, Tikhonov) and decide which one is better to use for our current case. Explain your reason.
4. (Optional) Find another method that can help reduce the noise using your own knowledge (not limited to this course).

Other Custom Filters

Filters can also be used to do image processing other than blurring the image. In this section you will experiment with two filters, sharpening filter and edge-detection (Laplacian) filter.

Sharpening Filter

0	-1	0
-1	5	-1
0	-1	0

Laplacian Filter

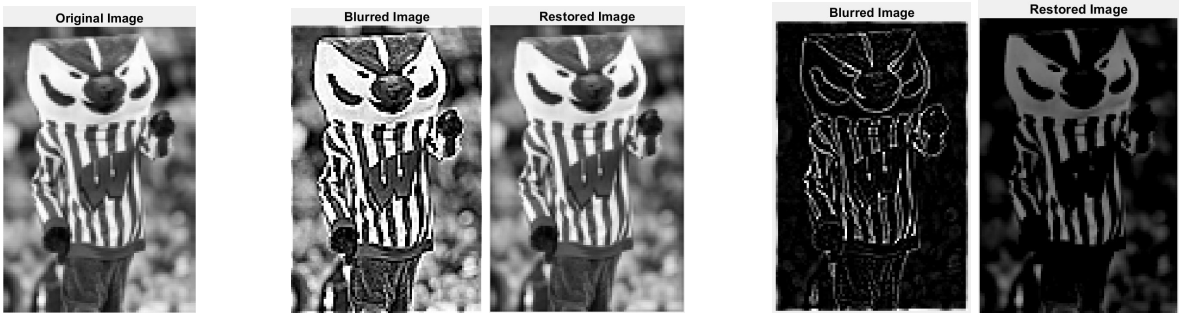
0	-1	0
-1	4	-1
0	-1	0

If you look at these two filters, you will notice that these two filters are very similar to each other. However, the result is shockingly different.

Code:

```
% Edge filter
edg = [0 -1 0; -1 4 -1; 0 -1 0];
% Sharpen filter
shr = [0 -1 0; -1 5 -1; 0 -1 0];
```

Original **Sharpening** **Restored** **Edge-Detection** **Restored**



Exercise 4: From what you have learned from the 2D convolution and least square.

1. Explain why Edge-Detection filter and Sharpen filter are providing different results after convolution.
2. Explain why you can fully recover from the sharpened image but cannot fully recover from Edge-Detection filtered image (hint: color black in an image has a value of 0).

Conclusion

Deblurring is the process of removing blurring artifacts from images such as defocus aberration or motion blur created by the sensor. This process is very important among those industries that use images. From the exercises provided, you have learned an image deblurring algorithm using the simple machine learning technique that combines the knowledge learned throughout this semester from the class. Creating and applying the filters is designed to help you understand how blurring can be created or processed on a computer using 2D convolution. Deblurring the image is designed to help you understand the process of deblurring and the fact that the process can be as simple as a least square problem that we have learned during the lecture. Adding noise to the filters is designed to help you understand how the real world blurring can cause problems to this method and the difficulty of deblurring in the real world can be. You should have also understood that there are methods that can limit the effects of the noise on the image and some other limitations of the method discovered in the last section. Wish you good luck on the future studying!

Reference:

2D convolution explained in GIF
https://imgflip.com/gif/2oiayv
2D convolution explained Siddhartha Chaudhuri, Stanford (2010). Image Processing
http://graphics.stanford.edu/courses/cs148-10-summer/docs/04_imgproc.pdf

Shimil Jose, et al (2017). Least Square Based Image Deblurring
https://www.researchgate.net/publication/321506771_Least_square_based_image_deblurring
Emory, (2006). Basics of Image Deblurring
http://www.mathcs.emory.edu/~nagy/courses/fall06/ID_lecture1.pdf
Mathworks
https://www.mathworks.com/
Bucky Image
https://www.pinterest.com/pin/393713192405478406/

Appendix:

Answers:

Warm up:

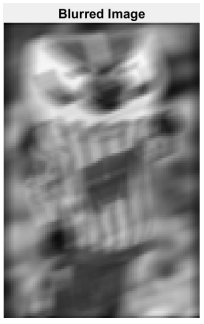
1. **Why do we care about image deblurring?**
   Blurs on images has always been a big problem in many industries that use images in any shape or form. Deblurring solves this problem and brings those industries more high-quality images.
2. **List all the methods mentioned above that we have learned in class.**
   Least Square, SVD Low-Rank Approximation, Tikhonov Regularization
3. **What should you do if your computer is slow or can't run the code provided?**
   Use the online version of the Matlab provided in the Note Section, run the code section by section, and reduce the image size.
4. **Perform a 2D convolution on the base matrix with the given convolution matrix.**

16	3	-5
-18	-5	12
16	19	-13

5. **Create Motion filter of size 10 with degree 20 and apply to the provided image "Bucky.jpg" using the function "fspecial".**

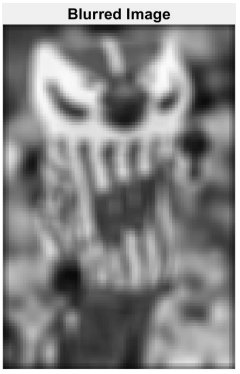
```
% Motion filter (Weiner)
len = 10;
theta = 20;
mtn = fspecial('motion', len, theta);

%% blurring using Various filter
filter = mtn;
blurred_img = conv2(image, filter, 'same');
figure, imshow(blurred_img);
title('Blurred Image')
```



Exercise 1

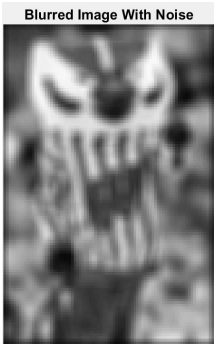
Original **Gaussian** **Restored**

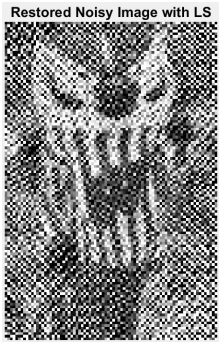


1. **Comment on the performance of the deblurring method on each filter.**
   The deblurring method mostly recovers the original image with all three filters.
2. **Is there any significant difference between the result for different filters? Explain.**
   You should be able to see that for all three filters without noise, there are minimal differences between the original image and restored. You are almost always able to recover from images without noise.

Exercise 2

Original **Gaussian with noise** **Restored**

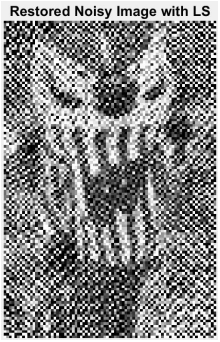


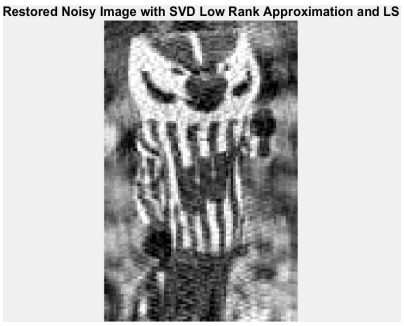


1. **Explain the effects of different levels of noise on the recovered result.**
   As the noise level increase, the blurred image gets harder to recover.
2. **Is it still fully recoverable with a noisy image using the least square method?**
   It is not fully recoverable using only LS.
3. **Explain why is this the case using what we learned in the class about solving least square.**
   As we learned in the class, solving least square will amplify noise. Even a small noise can cause big problems after been amplified by least square.

Exercise 3

LS **SVD low-rank** **Tikhonov**





1. **Comment on the performance of both methods in comparison to only using LS**
   Both methods worked much better than using only the LS method.
2. **Can any of these two methods produce a result that matches the quality of the LS result of the image without noise?**
   The quality of recovered images from a noisy image is no way near the result from a recovered image from images with no noise
3. **Compare and contrast the result of both methods (SVD, Tikhonov) and decide which one is better to use for our current case. Explain your reason.**
   SVD low-rank approximation is recovered more and looks not as blurry, but noise is not fully removed. Tikhonov doesn't have much noise but looks more blurred. Tikhonov is better in terms of reducing noise and SVD low-rank approximation is better in terms of reducing blur.
4. No answer

Exercise 4

1. **Explain why Edge-Detection filter and Sharpen filter are providing different results after convolution.**
   This is due to the calculation performed during convolution. When applying the sharpening filter to a color X area, the result will be 5*X - 4*X = X. As a result, any area of the same color will remain the same and any outliers will be enhanced. On the other hand, when performing the Laplacian filter on a color X area 4*X - 4*X = 0. Therefore any area of the same color will be cleared to 0 (black).
2. **Explain why you can fully recover from the sharpened image but cannot fully recover from Edge-Detection filtered image.**
   It is not hard to recover from the sharpening filter because image information is preserved even after applying the filter. However, the Edge-Detection (Laplacian) filtered image is nearly impossible to recover back to the original image as there are so many 0 values on the image. When doing least square, we are multiplying matrices. As a result, any value on the recovery matrix P multiplied by 0 will become 0, and therefore, the information is lost. Since much information has been lost during the process of convolution. It is almost impossible to fully recover from an Edge-Detection (Laplacian) filtered image using least square.