Team: ProximaCentauri

Single Image Motion Deblurring Using Transparency

Abhinaba Bala(2020701001), Neel Mishra(2020701009),

Ritam Basu(2020701005), Jigyasu Khandelwal(2020702013)

Mentor TA: Soumyasis Gun

Repo URL :
https://github.com/Digital-Image-Processing-IIITH/project-proximacentauri/

Contents

Motivation

One of the key problems of restoring a degraded image from motion blur is the estimation of the unknown shift-invariant linear blur filter.

In this work the image deblurring is separated into filter estimation and image deconvolution processes.

Previous Works

With unknown linear shift-invariant PSF, early approaches usually assume a priori knowledge to estimate the blur filter in deblurring the input image.

Recent approaches propose a variational Bayesian approach using an assumption on the statistical property of the image gradient distribution to approximate the unknown blurred image.

Overview

- Motion blur is a phenomena which is caused by the movement of object or camera shake at the time of triggering of camera shutter.
- The paper proposes a way to deblur the image.
- It is done by proposing a model for motion blur using transparency.

Understanding the problems

- The previously proposed model for image deblurring are mainly centred on solving an equation

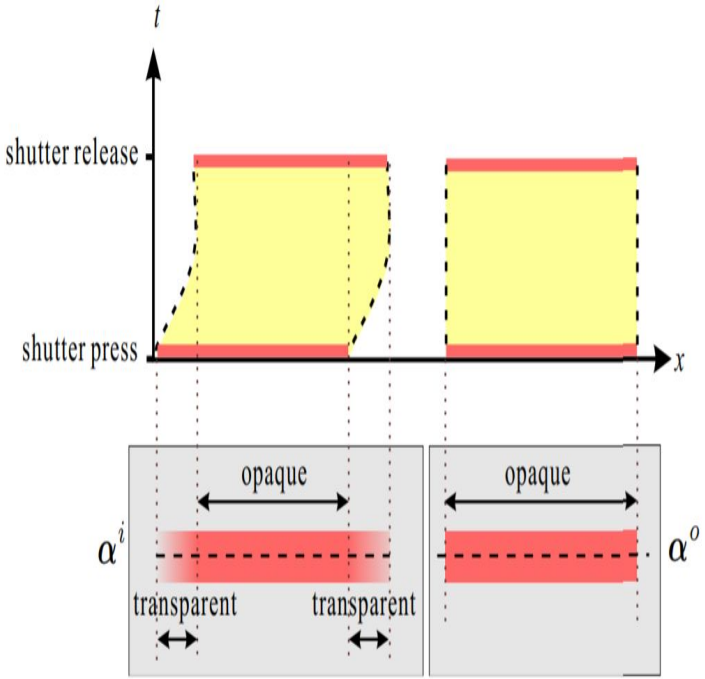
$$I = L \otimes f + n$$

- The main concern regarding restoring a still image containing motion blur is that the background may not undergo the same motion blur.

- The notion of transparency will help in resolving this.

Where does transparency come in?

- Transparency is the proportion of time that the background is exposed.
- We use transparency to model the blur filter.
- α_i defines transparency corresponding to blurred image pixels.
- α_0 defines transparency values corresponding to non blurred image pixels.



Analysis of object motion blur

- A formulation of motion blur from transparency point of view can be given as:

$$\alpha^i = \alpha^o \otimes f$$

- A MAP approach is proposed to recover the motion blur filter using transparency.
- Using Bayes rule:

$$P(f, \alpha^o | \alpha^i) \propto P(\alpha^i | f, \alpha^o) P(\alpha^o) P(f)$$

Analysis of object motion (cont)

- The likelihood measures the similarity between the input and convolved alpha values.

$$P(\alpha^i|f,\alpha^o) = \prod_{x,y} N(|\alpha^i_{x,y} - \sum_{i,j} \alpha^o_{x-i,y-j} f_{i,j}|; \sigma_1, \mu_1)$$

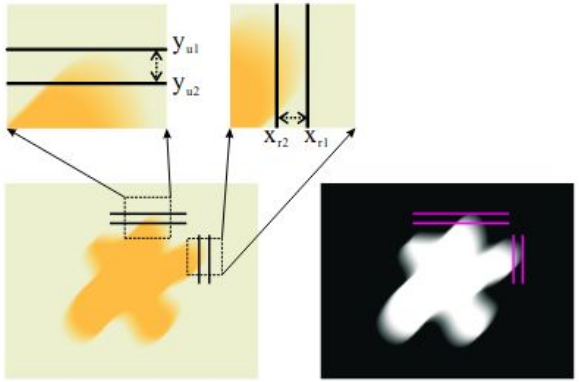
- And the prior is taken to be:

$$P(\alpha^o) = \prod_{x,y} \exp(-\lambda \alpha^o_{x,y}|1-\alpha^o_{x,y}|) \prod_{\substack{(x,y)\\(x',y')}} N(|\alpha^o_{x',y'} - \alpha^o_{x,y}|; \sigma_2, \mu_2)$$

Generalised Transparency In Motion Blur

- The aim here is to estimate the blur filter due to camera shake.
- The entire image is not required to be taken into computation.
- Considering the alpha blending of background and foreground colors, we obtain that we have to compute the same blur filter as we did before.

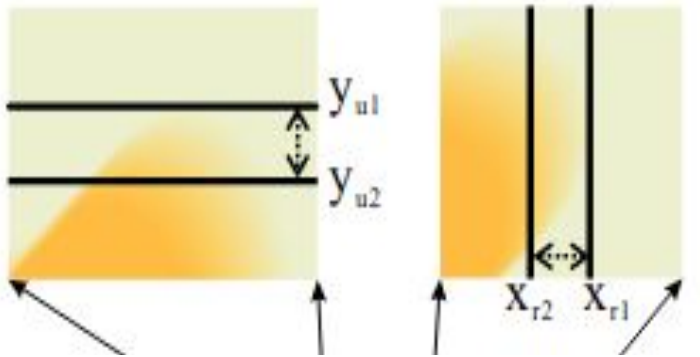
Upper bound for the filter size



Using transparency allows one to automatically detect an upper bound for the filter size.

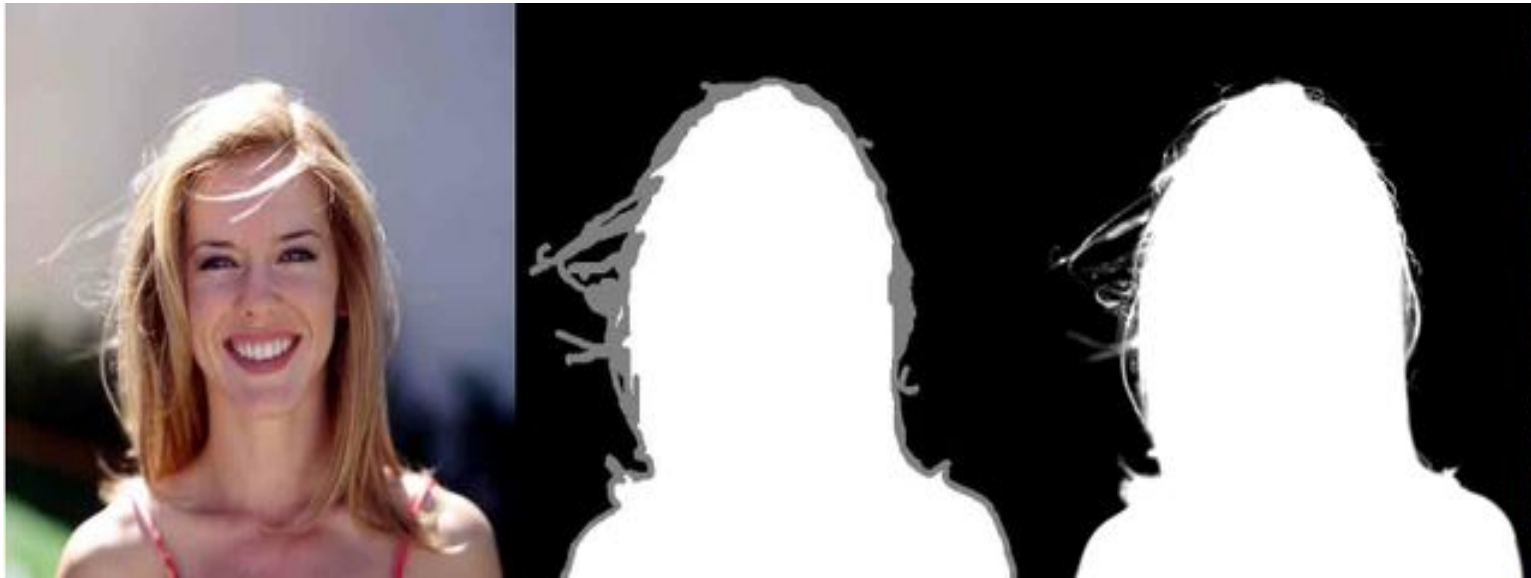
The image shown on the left is the blurred object while the image on the right is the corresponding transparency map.

Upper bound for the filter size




```
Max Width = X1 - X2 + 1
Max Height = Y1 - Y2 + 1
```

Alpha Matte



Alpha Matte

Our work starts with estimating the foreground and background regions, as well as the foreground opacity which is called alpha matte.

In the given paper, the authors employ methods from “A closed form solution to natural image matting. 2006” paper to estimate this region.

The aim is to minimize the matting Laplacian subject to scribbled constraints.

Matting Laplacian

For intuition, consider an image patch with two colours. The affinity between two pixels of the same colour decrease with distance, while the affinity between pixels of different colours is zero.

Matting affinity uses local estimates of means and variance.

$$W_M(i,j) = \sum_{k|(i,j)\in w_k} \frac{1}{|w_k|}\left(1+(I_i-\mu_k)\left(\Sigma_k+\frac{\varepsilon}{|w_k|}I_3\right)^{-1}(I_j-\mu_k)\right)$$

Alpha Matte results

Original Image:

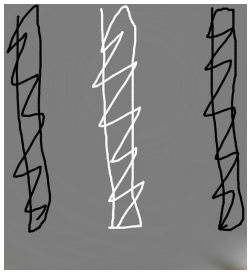
Our scribbled matte prior:

Output Matte:

Alpha Matte results

Original Image:

Our scribbled matte prior:



Output Matte:

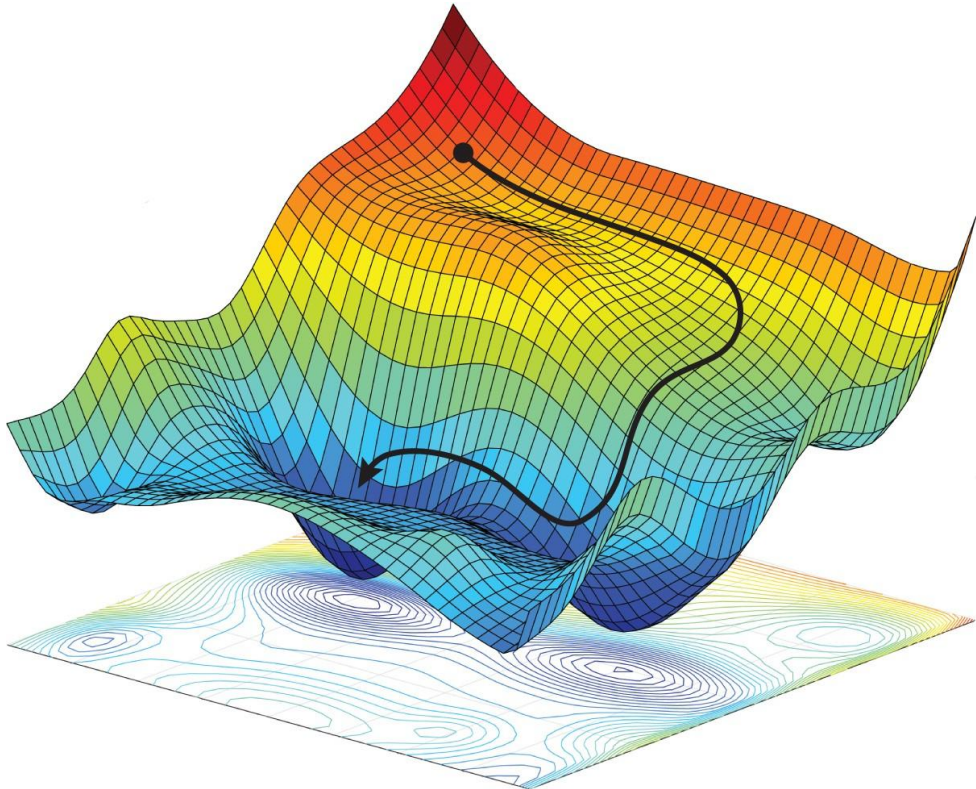
Limitations in Matte estimation

1. The alpha values are distorted for very intricate details along with inability to completely mark the priors.

2. The alpha values are wrongly learned if the foreground image has minimal contrast with the background.

3. In some cases the prior remained as the new alpha values without any learning.

Optimization :-



Optimization in paper

- The author has used conjugate gradient descent, and belief propagation to estimate blur filter.
- Here we have used only the gradient descent method.
- The sklearn has the conjugate gradient descent built in but it provides very less control in the process.
- Hence we decided to go with our implementation of gradient descent.

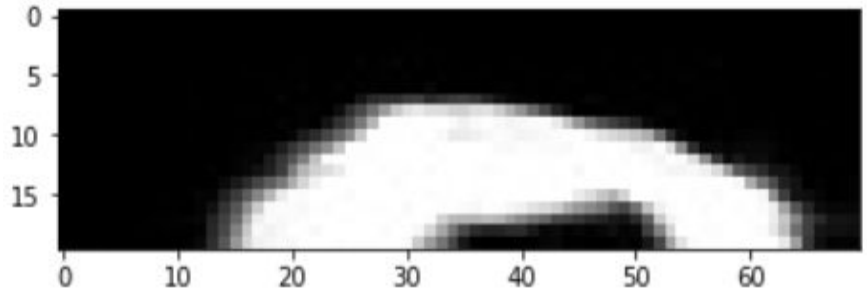
Optimization

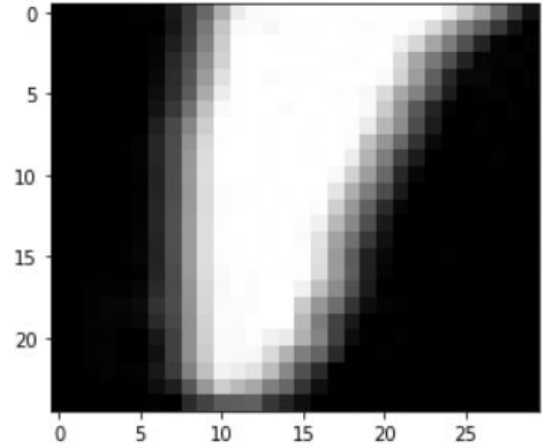
$$P(f, \alpha^o | \alpha^i) \propto P(\alpha^i | f, \alpha^o) P(\alpha^o) P(f)$$

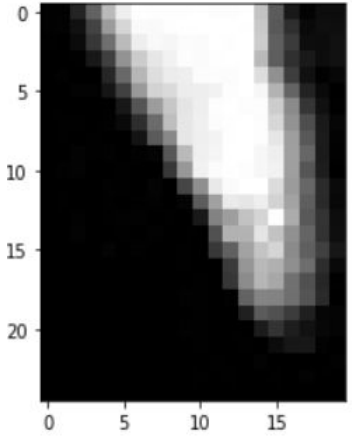
- We have to perform MAP optimization on the above equation.
- $P(\alpha^o)$ is a constant entity.
- $P(f)$ is defined from a uniform distribution.
- $P(\alpha^i | f, \alpha^o) = \prod_{x,y} N(|\alpha^i_{x,y} - \sum_{i,j} \alpha^o_{x-i,y-j} f_{i,j}|; \sigma_1, \mu_1)$ is the likelihood.

Patch Optimization :-

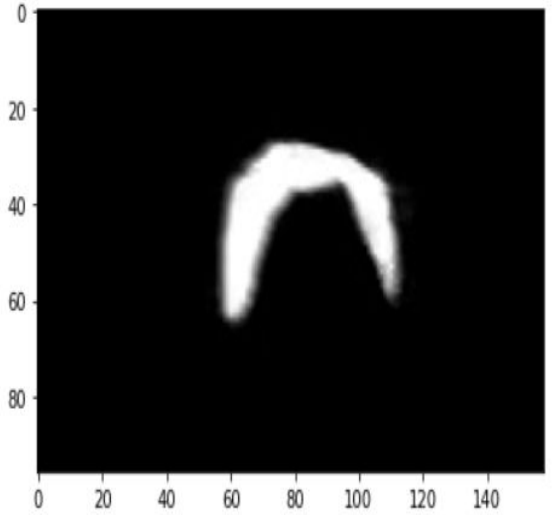
1. Optimized on patches, rather than optimizing on the entire image.
    a. We can divide the image in patches, one heuristic that can be applied is to target those region where the alpha values are between 0 and 1.
    b. This allows for speedy optimization, and better results as we can choose better hyper parameters in relatively less time.



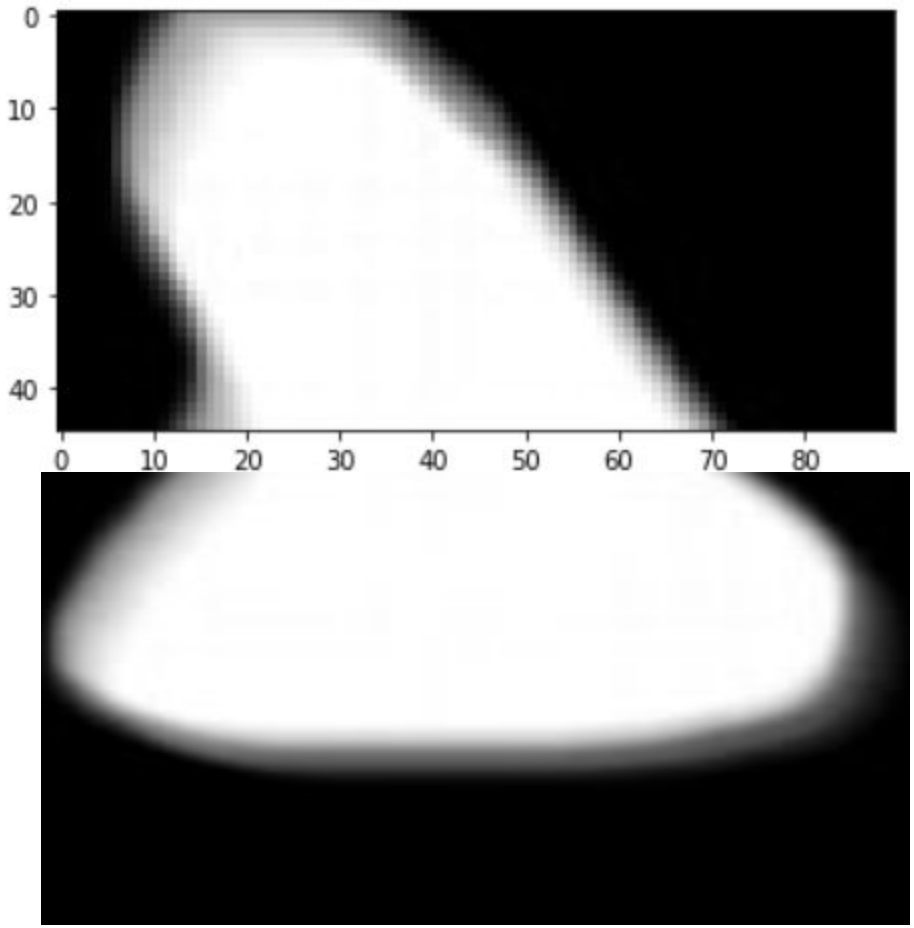




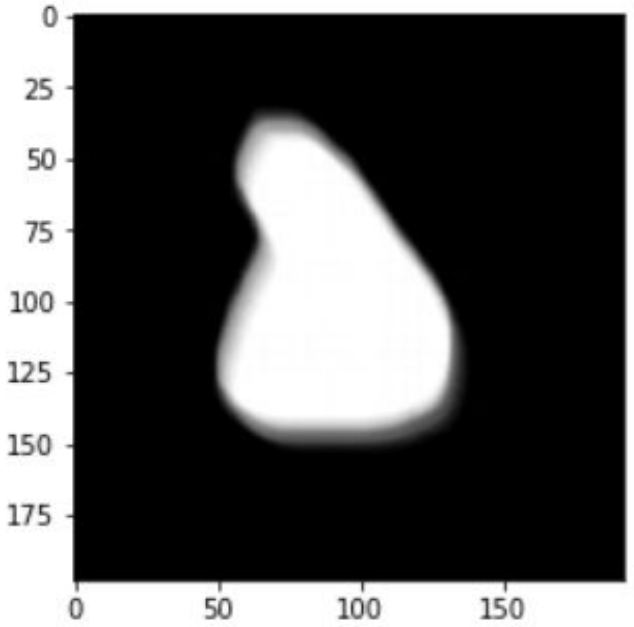
Patches taken



Original Image



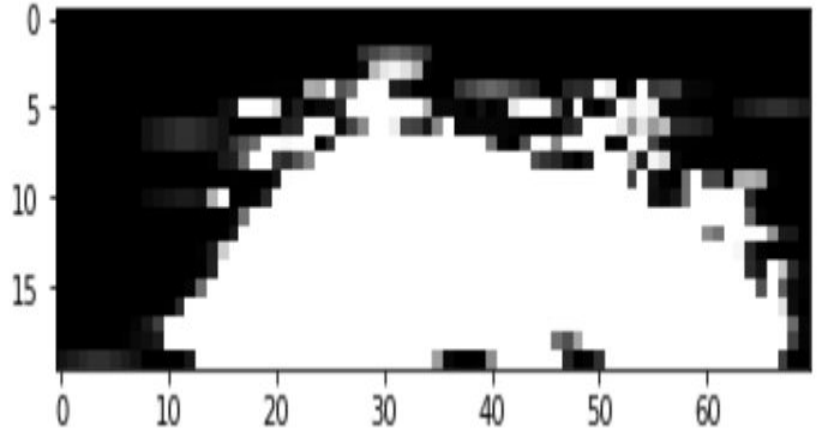
Patches taken

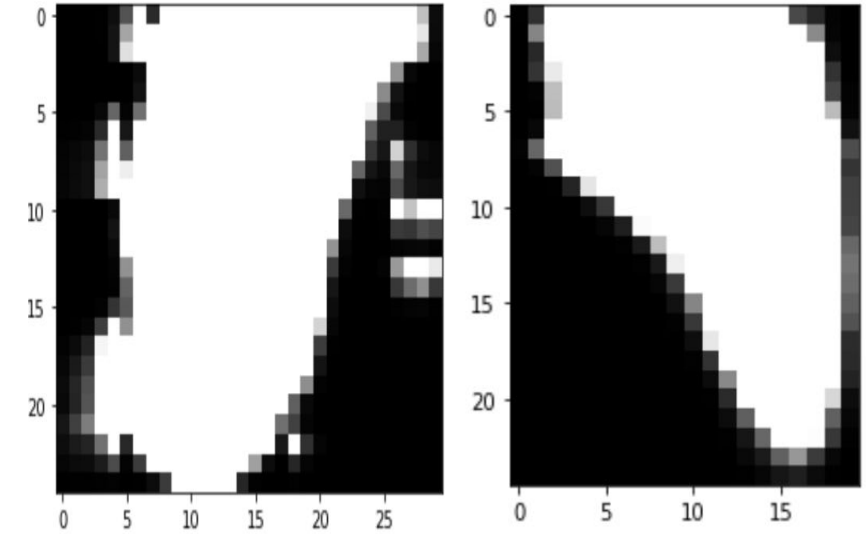


Original Image

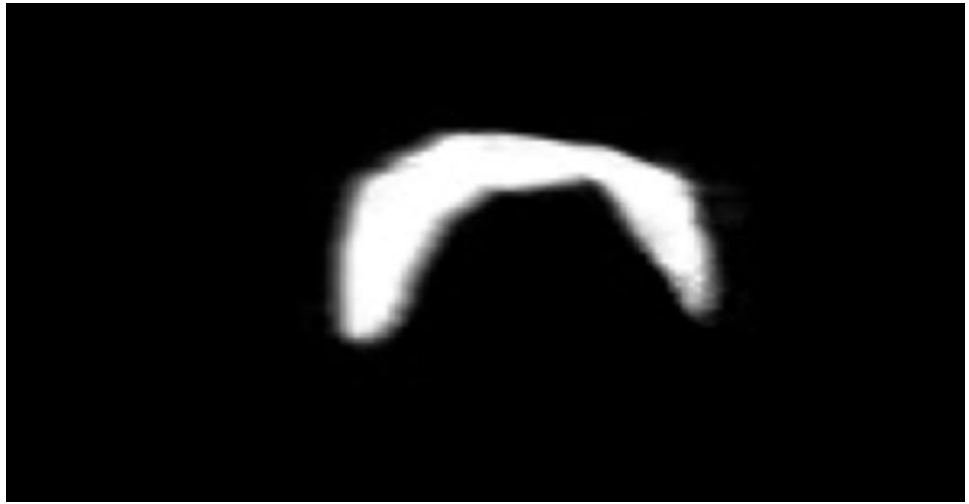
Optimization Process :-

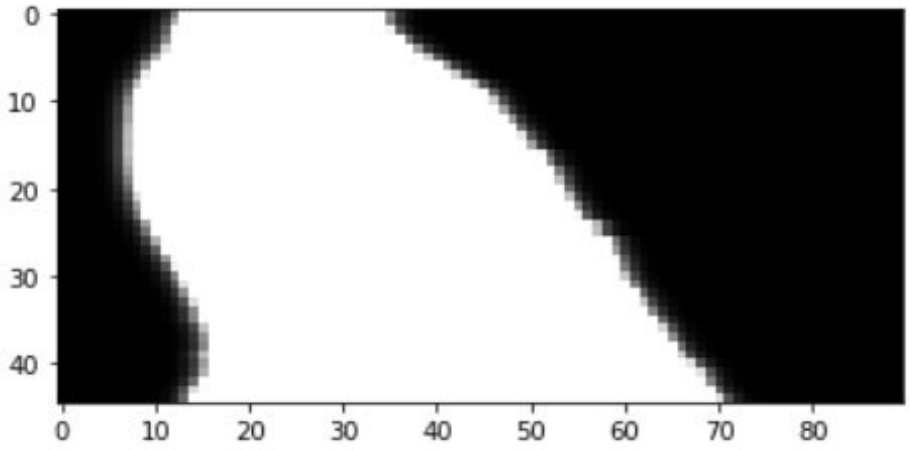
1. For each patch repeat :-
   1.1. Set the filter size.
   1.2. Initialize filter (random, all ones, all zeros)
   1.3. Perform gradient Descent
   1.4. Perform lucy richardson on the patch with the optimized filter.
   1.5. Store the desired output in an array to later combining it.
2. Stitch all the patches back to form one complete image.

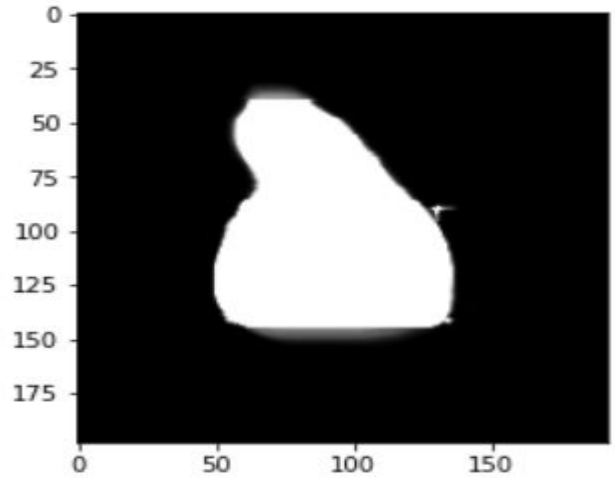

0
5
10
15
0
10
20
30
40
50
60

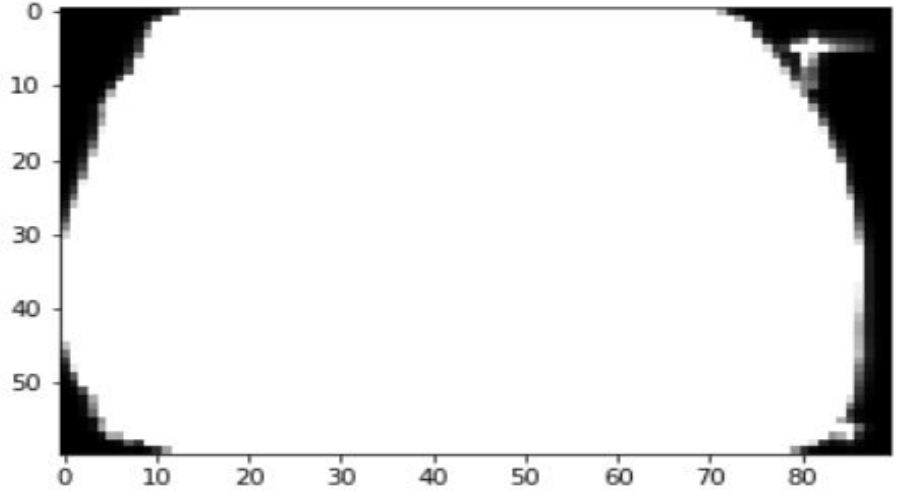

0
5
10
15
20
0
5
10
15
20
25

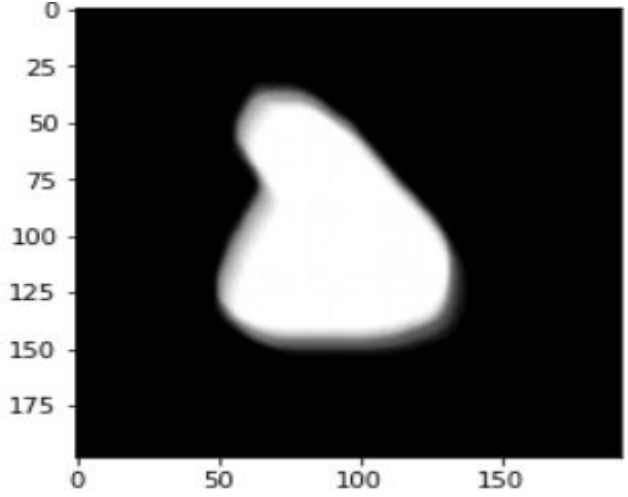
0
5
10
15
20
0
5
10
15



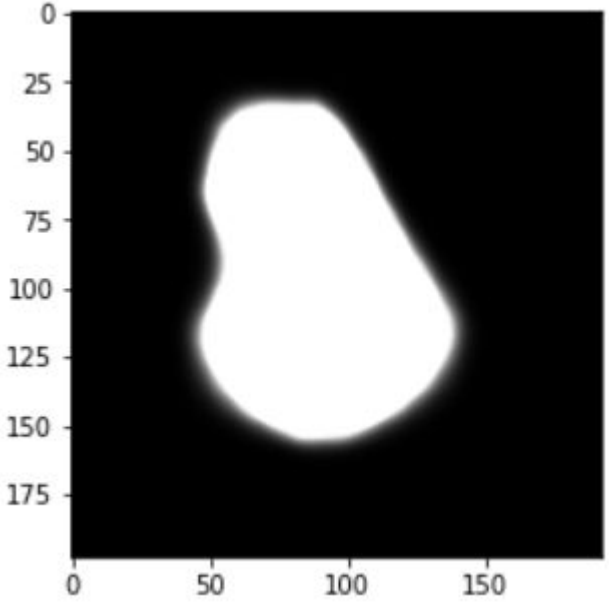

0
10
20
30
40
0
10
20
30
40
50
60
70
80


0
25
50
75
100
125
150
175
0
50
100
150


0
10
20
30
40
50
0
10
20
30
40
50
60
70
80


0
25
50
75
100
125
150
175
0
50
100
150

Why don’t we optimize on full image?

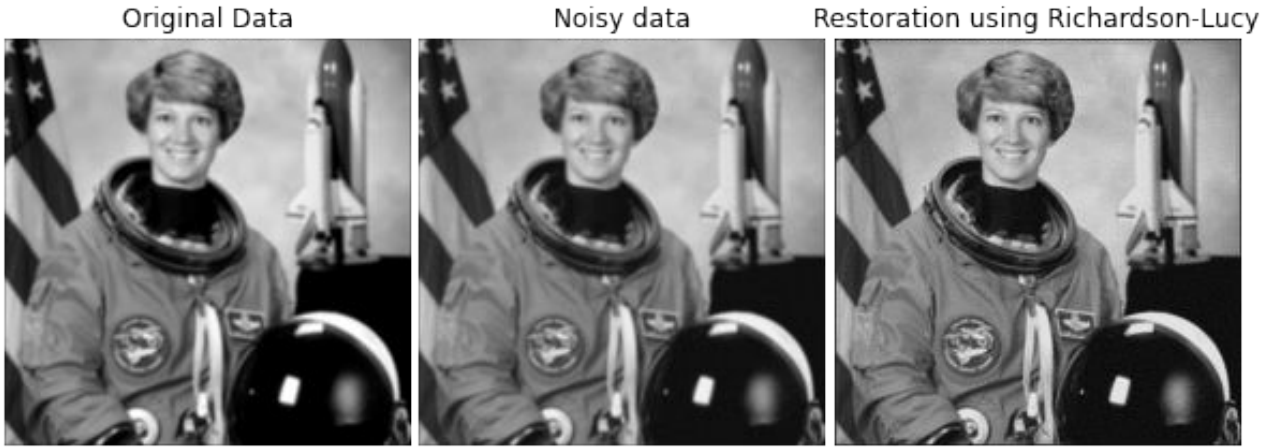


Why don’t we optimize on full image?

- It takes a lot of time to learn the filter.
- Fiddling with hyperparameter becomes tedious due to large execution time.
- Output is not as appealing as that with patches.

Deconvolution-Lucy Richardson

- The Richardson–Lucy algorithm, also known as Lucy–Richardson deconvolution, is an iterative procedure for recovering an underlying image that has been blurred by a known point spread function.



Lucy Richardson algorithm

- The LR algorithm is best understood if we first consider a simple iterative algorithm and then see how the LR method is an extension of this approach.
- As modelled earlier the blurred image is a convolution of unknown filter and unblurred image, if the noise is also considered:

$$g(x,y) = f(x,y) ** h(x,y) + n(x,y)$$

- Where '**' represents convolution *g(x,y)* represents the output blurred image *n(x,y)* represents noise *f(x,y)* represents the input image and *h(x,y)* represents point spread function.

- Equation proposed can be rewritten as

$$n(x,y) = g(x,y) - f(x,y) ** h(x,y)$$

- If we consider a good estimate then noise would be small

$$g(x,y) - f(x,y) ** h(x,y) = 0$$

- Now if we add *f(x,y)* both the sides

$$f(x,y) = f(x,y) + [g(x,y) - f(x,y) ** h(x,y)]$$

- Hence the final iterative model is

$$f_{k+1}(x,y) = f_k(x,y) + [g(x,y) - f_k(x,y) ** h(x,y)]$$

$$f_{k+1}(x,y) = f_k(x,y) + [g(x,y) - f_k(x,y) ** h(x,y)]$$

The procedure described by above equation is started by setting

$$f_0(x,y) = g(x,y)$$

Hence the first estimate of the input distribution is considered to be the measured output. Unless the image is very severely degraded, the output is not hugely different from the true input distribution and this simple procedure converges nicely.

The Equation can never be satisfied exactly, unless there is no noise at all, but will reach a point at which the size of the correction term in Equation reaches a minimum value.

The basic assumptions made in lucy richardson method are:

- It is assumed that point spread function is known. And hence in proposed function for lucy richardson algorithm point spread function is fed by the user.
- We assume that the noise at the output is governed by poisson distribution function.

Limitation of Lucy-Richardson

- The limitation of lucy_richardson algorithm is that the output may depend upon the number of iterations employed.

- Another limitation is that if the point spread function is constant then lucy_richardson is not efficient approach to deconvolve.

Limitations

- Patches have to be selected manually to run the optimization efficiently.
- The performance drops as the amount of blur increases.
- It was observed that multiple optimization algorithms have to used to get finer results.

Division of Work

Abhinaba

Alpha Matte

Documentation

Neel

Optimization

Documentation

Ritam

Optimization

Lucy-Richardson

Jigyasu

Lucy-Richardson

Alpha Matte

References

1. Alpha Matting: https://webee.technion.ac.il/people/anat.levin/papers/Matting-Levin-Lischinski-Weiss-CVPR06.pdf
2. Lucy Richardson :- A.Jain.Fundamentals of Digital Image Processing. Prentice Hall,1988.
3. Single image motion: http://jiaya.me/all_final_papers/motion_deblur_cvpr07.pdf

Thank You