VECTOR VALUED IMAGE REGULARIZATION WITH PDE

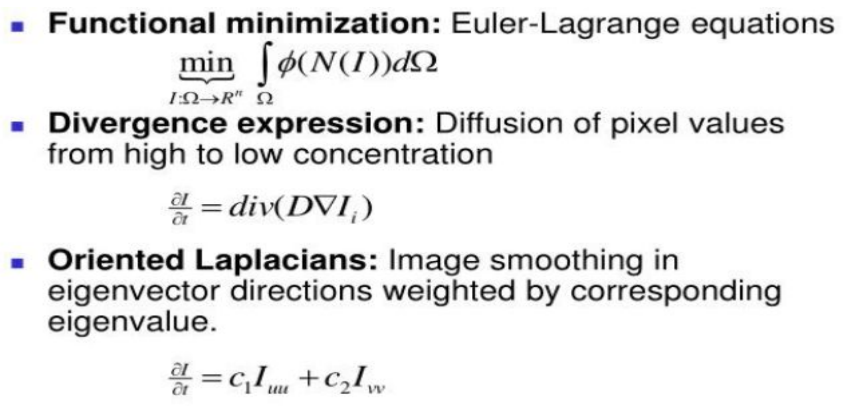
- Aneesh Garg (1800050007)
- Dhananjay Singh (180050029)

Regularization Algorithms

- What are they? - Regularization algorithms basically consist of simplifying an image, in a way that only interesting features are preserved while unimportant data (considered as “noise”) are removed.
- Where are they used? - Regularization algorithms are used as low-level steps in more complex processing pipelines and their adequations to the considered problems are crucial.
- It is often one of the key stage performed by high-level algorithms in computer vision or image processing areas, such as object recognition, tracking, etc
- The framework of nonlinear PDEs (partial differential equations) led to strong improvements in the formalization of regularization methods
- How do they work? - Non linear PDEs succeed in smoothing data while preserving large global features. Step-by-step image regularization is performed and a continuous sequence of smoother images is generated. Roughly speaking, regularization PDEs may be seen as nonlinear filters that simplify the image little by little and minimize the image variations.
- The data are gently regularized step-by-step and a continuous sequence of smoother images I(t) is generated whereas the evolution time t goes by.
- They generally do not converge towards a very interesting solution. Most of the time, the image obtained at convergence (t tending to inf) is constant, corresponding to an image without any variations: This is actually the most simplified image we can obtain.

Image Regularization

Non-linear PDE’s have been classified in one of these three approaches:-



- **Functional minimization:** Euler-Lagrange equations
  $$\underbrace{\min}_{I:\Omega\to R^n}\int_{\Omega}\phi(N(I))d\Omega$$
- **Divergence expression:** Diffusion of pixel values from high to low concentration
  $$\frac{\partial I}{\partial t}=div(D\nabla I_i)$$
- **Oriented Laplacians:** Image smoothing in eigenvector directions weighted by corresponding eigenvalue.
  $$\frac{\partial I}{\partial t}=c_1 I_{uu}+c_2 I_{vv}$$

- Functional Minimization - Regularizing an image I may be seen as the minimization of a functional E(I) measuring a global image variation. Minimizing this functional will flatten the image variation and gradually remove the noise. N(I) is a norm designed in a way to relate to local image variations.
- Divergence Expressions - A regularization process may also be more locally designed, as a diffusion of pixel values, directed by a 2 by 2 diffusion tensor D (symmetric and definite-positive matrix). D is specially designed from the spectral elements of the structure tensor G in order to anisotropically smooth I, while taking its intrinsic local geometry into account, preserving its global discontinuities.
- Oriented Laplacians - 2D image regularization may be finally seen as the simultaneous juxtaposition of two oriented 1D heat flows, leading to 1D Gaussian smoothing processes along orthonormal directions (eigenvectors of structure tensor), with different weights c1 and c2

Structure Tensor

$$\mathbf{G} = \sum_{i=1}^{n} \nabla I_i \nabla I_i^T = \sum_{i=1}^{n} \left(\begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \right)$$

- Structure tensor helps us to extend nonlinear regularization PDEs to vector-valued images (RGB).
- A coupling between image channels can be inferred from the local vector geometry of an image which is aptly defined by structure tensor.
- Structure tensor is particularly interesting to us since its eigenvalues define the local min/max vector-values variations of I in corresponding spatial directions (defined by eigenvectors), i.e., the spectral elements of G define the local geometry of the vector-valued image discontinuities.
- We can use structure tensor eigenvalues to get an idea of corners and edges as used in Harris corner detection.

Geometric Meaning of Oriented Laplacians

$$\frac{\partial I_i}{\partial t} = c_1 \; I_{i_{\xi\xi}} + c_2 \; I_{i_{\eta\eta}} = \mathbf{trace}\;(\mathbf{T}\mathbf{H}_i) \qquad (i = 1..n),$$

- The last two approaches namely Divergence Expressions and Oriented Laplacians have been linked together in this equation.
- Where first equality expression is related to Oriented Laplacians and another is related to Divergence Expressions with T being the diffusion tensor.

Tensor Field

- Tensor field corresponds to T in the previous expression, it is designed in such a way that it takes in account the following points:-

    1. We do not want to mix diffusion contributions between image channels. The desired coupling between vector components I(i) should only appear in the diffusion PDE through the computation of the structure tensor G, in order to control the local smoothing behavior of the regularization process. This means we have to define only one diffusion tensor T.
    2. On homogeneous regions (corresponding to low vector variations), we want to perform an isotropic smoothing, i.e., a 2D heat flow that smoothens the noise efficiently with no-preferred directions. The tensor T must then be isotropic in these regions.
    3. On vector edges (corresponding to high vector variations), we want to perform anisotropic smoothing along the vector edges (direction represented by the eigenvector corresponding to lower eigenvalue) , in order to preserve them while removing the noise. The tensor T must be anisotropic in these regions.

- Thus, T comes out to be as follows:

$$\mathbf{T} = f_-\left(\sqrt{\lambda_+^* + \lambda_-^*}\right) \theta_-^* {\theta_-^*}^T + f_+\left(\sqrt{\lambda_+^* + \lambda_-^*}\right) \theta_+^* {\theta_+^*}^T .$$

  where, λ and $\boldsymbol{\theta}$ are defined to be the spectral elements of a Gaussian smoothed version of the structure tensor G
  and

$$f_+(s) = \frac{1}{1+s^2} \quad \text{and} \quad f_-(s) = \frac{1}{\sqrt{1+s^2}} .$$

- This choice of f+ and f- is, of course, one possible “empiric” choice and that verifies the above geometric properties, relying on practical experience. We can easily adapt the weighting functions f+ and f- to obtain regularization behaviors for specific problems.
- Why do we convolve structure tensor with gaussian?
  A Gaussian smoothed version of the structure tensor G, allows us to retrieve a more coherent vector geometry and gives a better approximation of the vector discontinuities directions.
- This vector-valued regularization equation smoothes the image in coherent spatial directions and then preserves the edges well, by allowing only the necessary geometric coupling between vector channels I(i). Its form has steadily followed the local analysis of classical multivalued regularization algorithms.

Oriented Gaussian Kernel

Let us suppose first that T is a constant tensor over the definition domain then we get the image I as a function of time as follows,

$$I_{i_{(t)}} = I_{i_{(t=0)}} \; * \; G^{(\mathbf{T},t)} \qquad\qquad (i = 1..n),$$

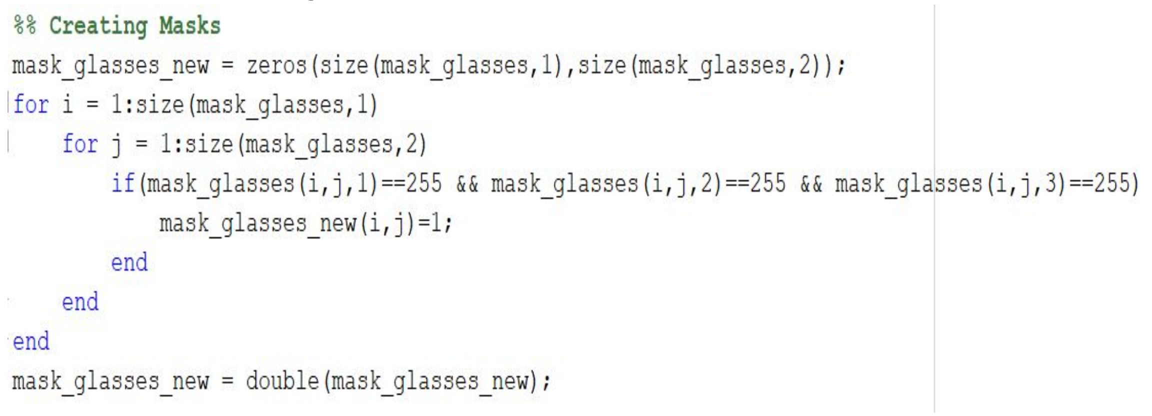
And

$$G^{(\mathbf{T},t)}(\mathbf{x}) = \frac{1}{4\pi t}\exp\left(-\frac{\mathbf{x}^T\mathbf{T}^{-1}\mathbf{x}}{4t}\right) \qquad \text{with} \qquad \mathbf{x} = (x \quad y)^T.$$

When T is not constant (which is generally the case), i.e., represents a field of variable diffusion tensors, the PDE becomes nonlinear and can be viewed as the application of temporally and spatially varying local masks $G^{(T,t)}(x)$ over the image I. As before, the shape of each tensor T gives the exact geometry of the local smoothing process performed by the trace-based PDE point by point.

Implementation

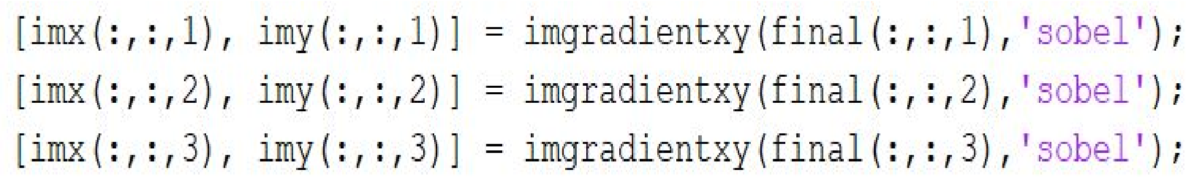
1. Creation of Binary Mask



```
%% Creating Masks
mask_glasses_new = zeros(size(mask_glasses,1),size(mask_glasses,2));
for i = 1:size(mask_glasses,1)
    for j = 1:size(mask_glasses,2)
        if(mask_glasses(i,j,1)==255 && mask_glasses(i,j,2)==255 && mask_glasses(i,j,3)==255)
            mask_glasses_new(i,j)=1;
        end
    end
end
mask_glasses_new = double(mask_glasses_new);
```

We first created manually a RGB mask over the text or object that we want to inpaint and then created its binary version.

2. Image Gradients



```
[imx(:,:,1), imy(:,:,1)] = imgradientxy(final(:,:,1),'sobel');
[imx(:,:,2), imy(:,:,2)] = imgradientxy(final(:,:,2),'sobel');
[imx(:,:,3), imy(:,:,3)] = imgradientxy(final(:,:,3),'sobel');
```

- Computed gradients across x and y direction using ‘sobel’ operator.
- This will be used to compute structure tensor at each point

- Sobel operator

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

3. <u>Evaluating Structure Tensor</u>

- Evaluated structure tensor at the hole

```
imrxx = imx(i,j,1)*imx(i,j,1);
imrxy = imx(i,j,1)*imy(i,j,1);
imryy = imy(i,j,1)*imy(i,j,1);
imbxx = imx(i,j,2)*imx(i,j,2);
imbxy = imx(i,j,2)*imy(i,j,2);
imbyy = imy(i,j,2)*imy(i,j,2);
imgxx = imx(i,j,3)*imx(i,j,3);
imgxy = imx(i,j,3)*imy(i,j,3);
imgyy = imy(i,j,3)*imy(i,j,3);

ixx = (imrxx+imbxx+imgxx);
ixy = (imrxy+imbxy+imgxy);
iyy = (imryy+imbyy+imgyy);

G = [ixx ixy; ixy iyy];
G = imfilter(G,gauss);
```

- Then convoloved it with the gaussian mask to get the gaussian smoothed structure tensor

```
gauss = fspecial('gaussian',3,1);
```

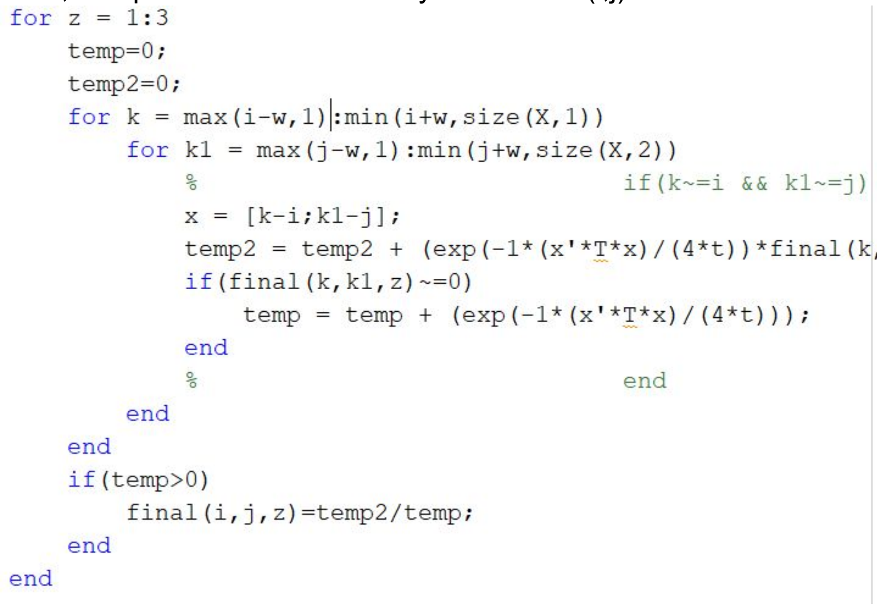
4. <u>Evaluating Tensor field</u>

- Evaluated eigenvalues and eigenvectors of structure tensor and created tensor field

```
[V, D] = eig(G);
[D,I] = sort(diag(D));
D = flip(D);
D = diag(D);
V =  V(:, flip(I));
thieta_plus = V(:,1);
thieta_minus = V(:,2);
lambda_plus = D(1,1);
lambda_minus = D(2,2);
T = (1/sqrt(1+(lambda_plus+lambda_minus)))*(thieta_minus*thieta_minus');
T = T + (1/(1+(lambda_plus+lambda_minus)))*(thieta_plus*thieta_plus');
T = inv(T);
```

5. Applying $G^{(T,t)}(x)$ locally

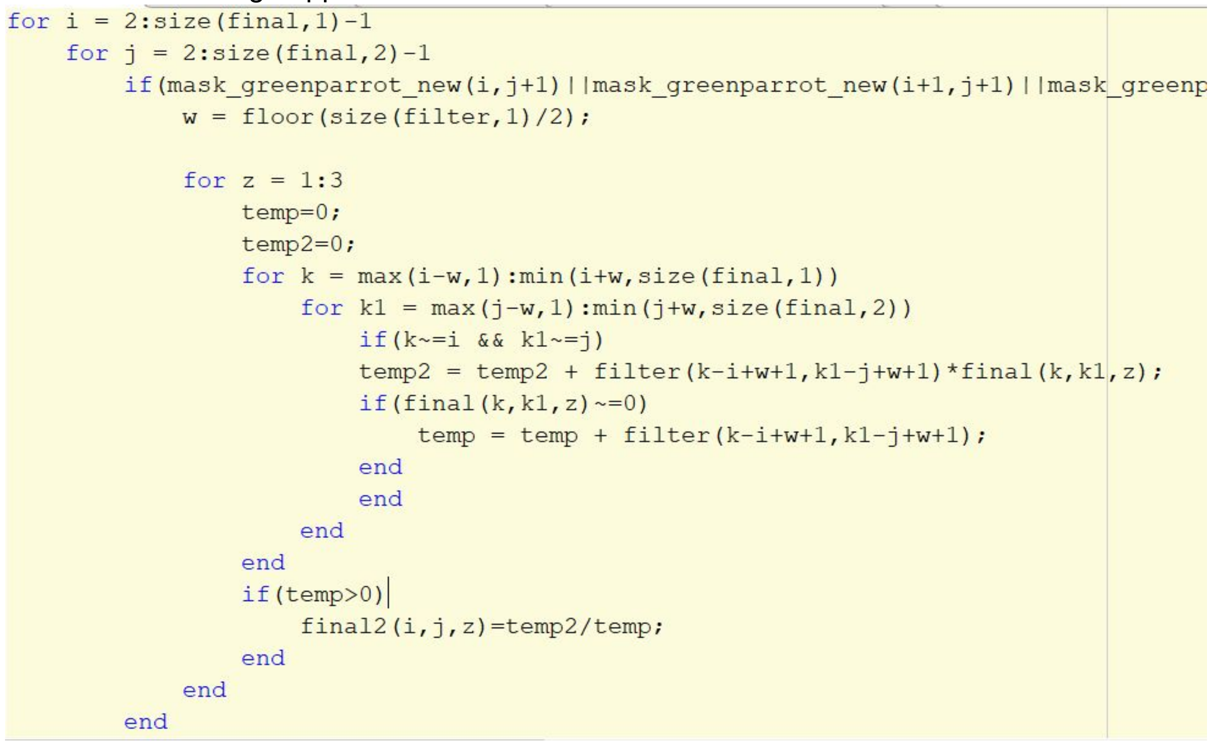
- Took weighted average of neighbouring intensities
- Weights being determined by $G^{(T,t)}(x)$
- Thus, computed the new intensity at the hole (i,j)



```
for z = 1:3
    temp=0;
    temp2=0;
    for k = max(i-w,1):min(i+w,size(X,1))
        for k1 = max(j-w,1):min(j+w,size(X,2))
            %                              if(k~=i && k1~=j)
            x = [k-i;k1-j];
            temp2 = temp2 + (exp(-1*(x'*T*x)/(4*t))*final(k,k1,z));
            if(final(k,k1,z)~=0)
                temp = temp + (exp(-1*(x'*T*x)/(4*t)));
            end
            %                              end
        end
    end
    if(temp>0)
        final(i,j,z)=temp2/temp;
    end
end
```

6. Gaussian Smoothing

- Smoothing the false edges created by completely filling the missing parts through inpainting
- This smoothing process removes the false edge artifacts created by inpainting and makes the image appear more realistic

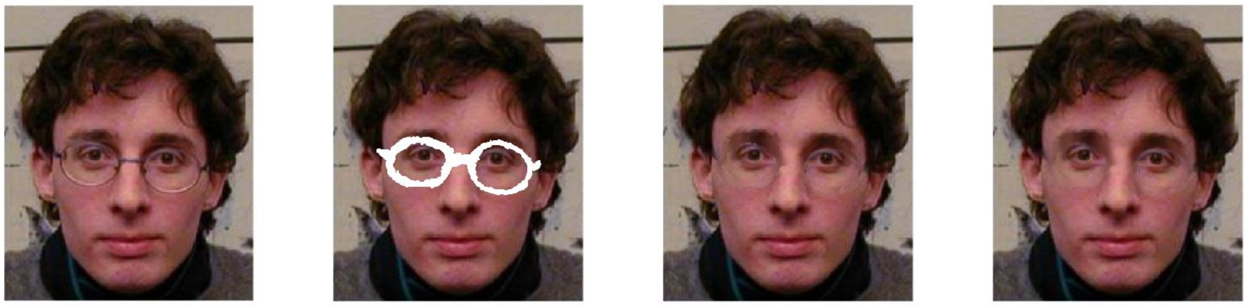


```
for i = 2:size(final,1)-1
    for j = 2:size(final,2)-1
        if(mask_greenparrot_new(i,j+1)||mask_greenparrot_new(i+1,j+1)||mask_greenp
            w = floor(size(filter,1)/2);

            for z = 1:3
                temp=0;
                temp2=0;
                for k = max(i-w,1):min(i+w,size(final,1))
                    for k1 = max(j-w,1):min(j+w,size(final,2))
                        if(k~=i && k1~=j)
                        temp2 = temp2 + filter(k-i+w+1,k1-j+w+1)*final(k,k1,z);
                        if(final(k,k1,z)~=0)
                            temp = temp + filter(k-i+w+1,k1-j+w+1);
                        end
                        end
                    end
                end
                if(temp>0)
                    final2(i,j,z)=temp2/temp;
                end
            end
        end
```

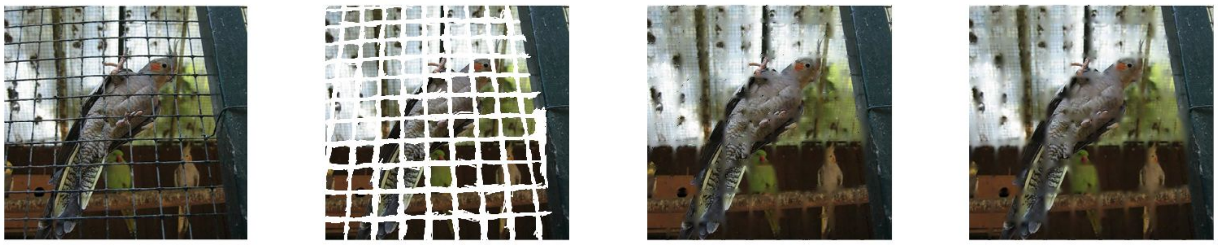
Results

- Inpainting - 1



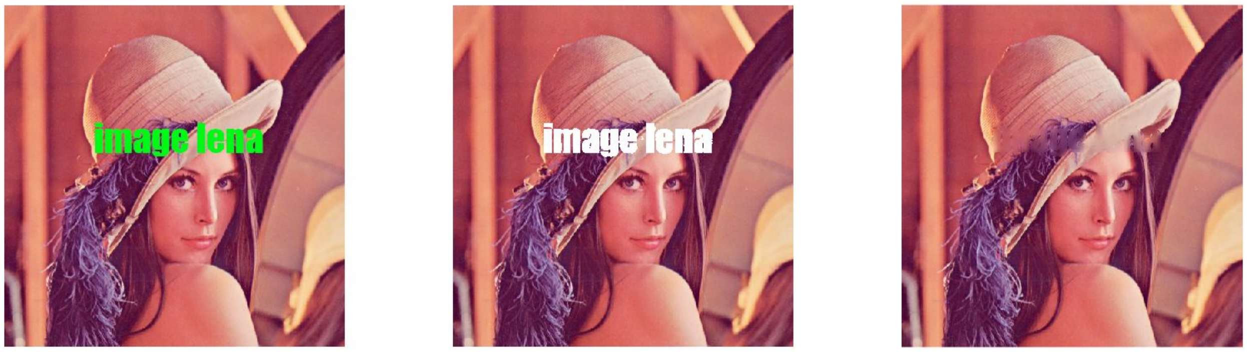
Parameters : Neighbourhood = 5x5, t=100, maxiter = 1

- Inpainting - 2



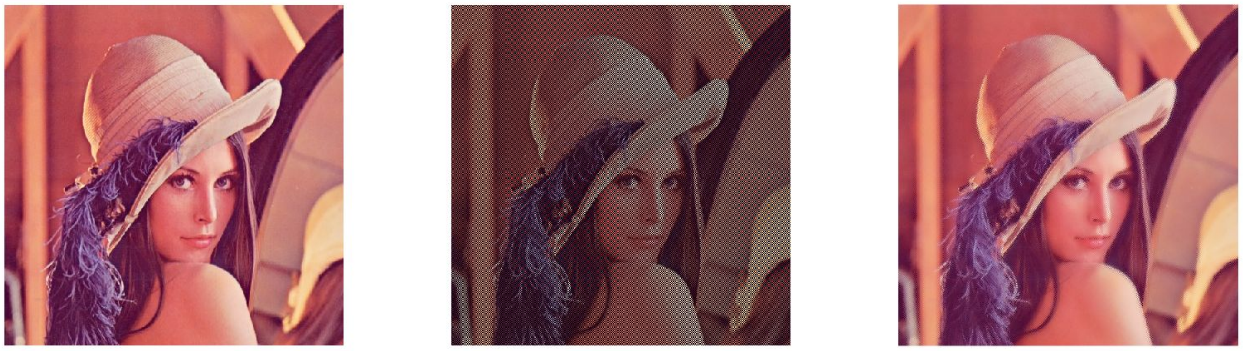
Parameters : Neighbourhood = 11x11, t=100, maxiter = 1

- Inpainting - 3



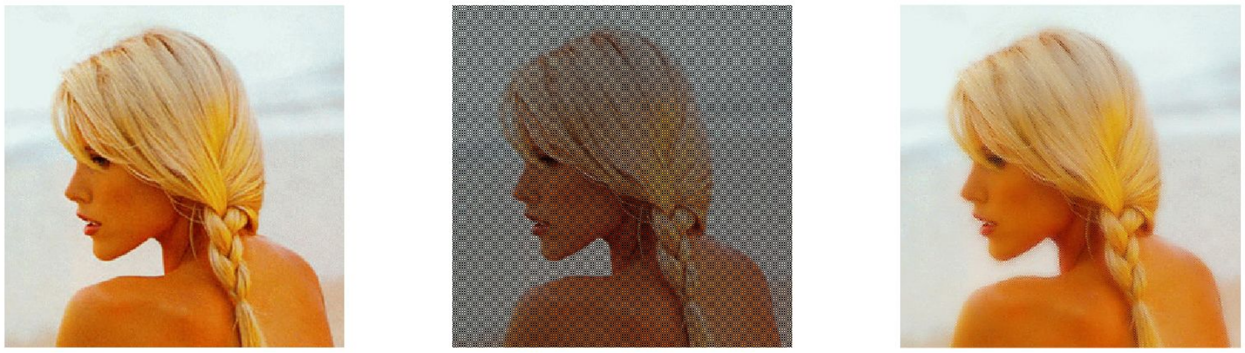
Parameters : Neighbourhood = 11x11, t=100, maxiter = 1

- Image Restoration - 1



Parameters : neighbourhood = 11x11, t=1000, maxiter = 1
SSIM(structural similarity index) of the final reconstructed image and the original image = 0.969

- Image Restoration - 2



Parameters : neighbourhood = 11x11, t=1000, maxiter=1
SSIM of the final reconstructed image and the original image = 0.922200

Observations

1. As the width of the mask increases, the effectiveness of our algorithm decreases. This is evident from the fact that as the width increases there is a greater loss of information which makes it hard to reconstruct.

2. Our results are very sensitive to the neighborhood size, i.e., they increase/decrease very rapidly with changes in neighborhood size.

3. Our results do depend on the number of iterations, but they start to converge as the number of iterations reaches a high value.

4. Our results depend very slightly on the time parameter, i.e., for observing a significant change in the results, a large change in time parameter is required.

5. This algorithm yields very good results in a continuous region, but at the edges, we do not observe such good results. This is due to the fact that it is hard for the algorithm to distinguish between the two sides across the edge and ends up assigning a value somewhere in between.

References

- https://tschumperle.users.greyc.fr/publications/tschumperle_pami05.pdf
- https://www.slideserve.com/fineen/vector-valued-image-regularization-with-pde-s-a-common-framework-for-different-applications