Non-Local Patch-Based Image Inpainting

Team Name: Formula51
Mentor TA: Haripraveen
Repo URL:
https://github.com/Digital-Image-Processing-IIITH/dip-project-formula51

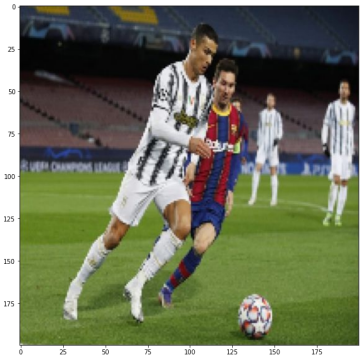
Team Members:
Trinadh (2019101043 - CSE)
Harsha(2019101040-CSE)
Nikit (2019101022 - CSE)
Sri Ram (2019101032 - CSE)

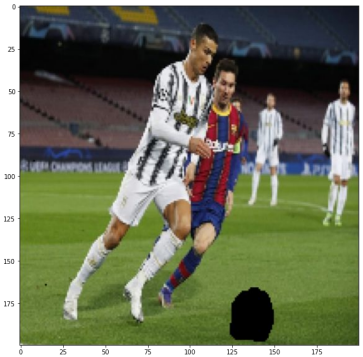
Introduction

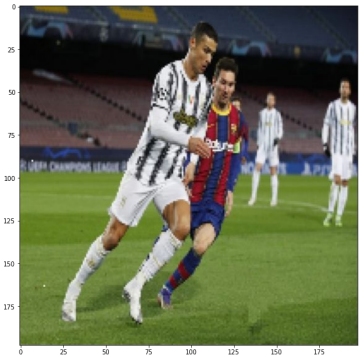
Image inpainting is the task of filling in an unknown region in an image.

It is very useful in reconstructing portion of image If an area is damaged, or if one wishes to remove an unwanted object from the image.

Main goal of image inpainting are the convincing restitution of structures and textures.







Approach of Paper

Our Aim is to fill the Occluded (unknown pixels) pixels as natural as possible as if the occlusion was never there.

For this we propose the idea of using patch match based inpainting approach. Patch here can be thought of as neighbourhood around a pixel.

We try to find the closest related patch in the image (in the known region) for each patch which is there in occluded region.

First thoughts.....

With little thought, one can see that finding the closest related patch (in the known image) for each patch in the occluded region requires some sort of metric to define what we mean by closest.

For this the paper considers the following metric:

$$E(u, \phi) = \sum_{p \in \mathcal{H}} d^2(W_p, W_{p+\phi(p)}).$$

Where d(A, B) is a metric using which we measure the closeness between patches A and B

Continuation...

W_p contains the intensities of the neighbourhood pixels of p. i.e patch of p. Shift map ϕ maps pixel to a shift in pixel. By the end of the algorithm ideally this shift should take pixel's patch to closest related patch.

For finding closest matching Patch the paper uses existing Approximate Nearest Neighbour Search with Patch Match Algorithm.

We have used the same for finding the closely related patches.

How is this Paper different

Inclusion of texture

When several regions with a similar average color but differing in texture contents exist, problems can arise with inpainting algorithm.

So the modified distance metric for comparing patches is:

$$d^2(W_p, W_q) = \frac{1}{N} \sum_{r \in \mathcal{N}_p} \left(\|u(r) - u(r - p + q))\|_2^2 + \lambda \|T(r) - T(r - p + q)\|_2^2 \right)$$

Where λ is a weighting scalar to balance the effects of both color and texture information in the patch distance.

Onion Peel Initialization

Unlike traditional random initialization in which we assign random intensities to pixels in occluded region.

We initialize the regions of occluded region layerwise. To understand this better we first initialize the outermost layer in occluded region using already known pixel intensity and texture values.

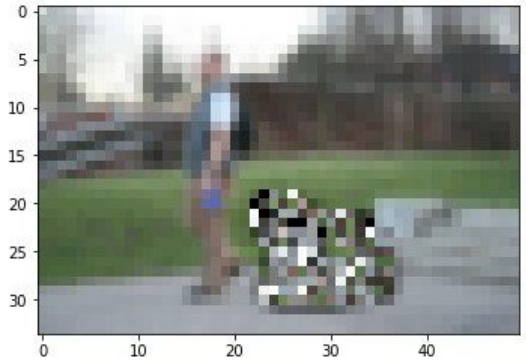
But in the next go when we initialize deeper layers we use the previously calculated layers and known pixels to compute the current layers values.

Some of the Onion Peel initialization results.

Notice that the initialization is in a lower dimension of image. (as discussed in multi scale scheme)

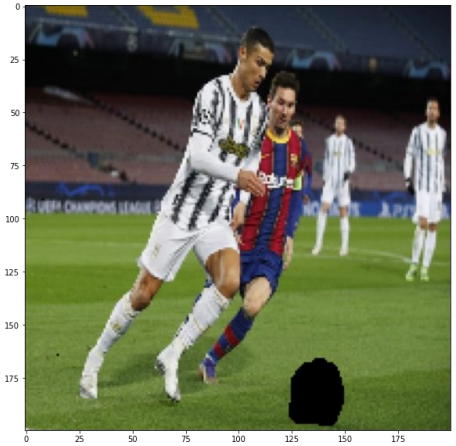
Original image

Random Initialisation

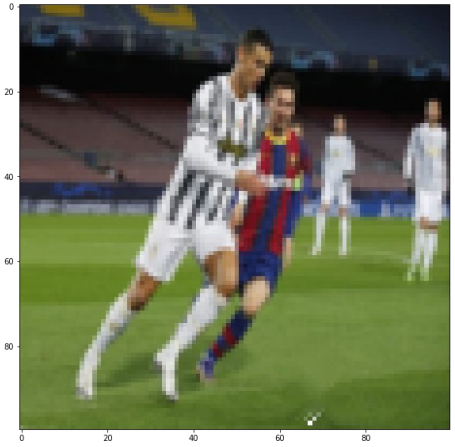


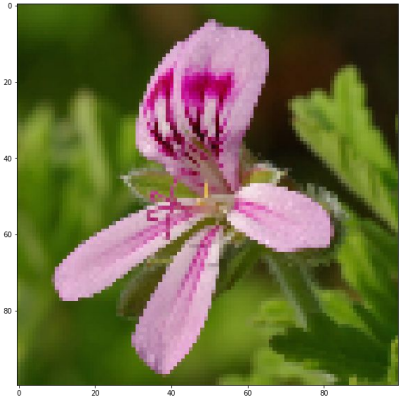
Onion Peel initialisation

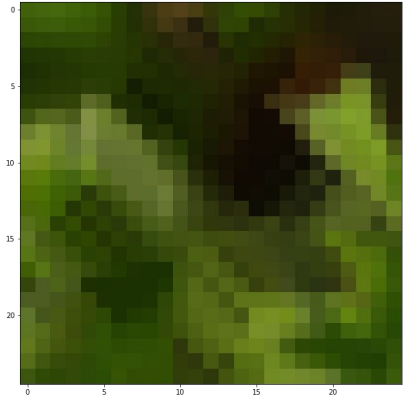
Original Image



Onion peel Initialisation





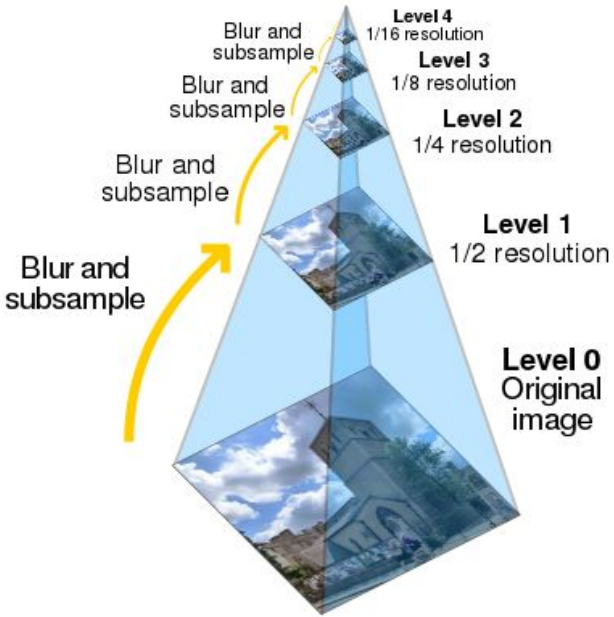


Multi Scale Scheme

The paper uses multiscale scheme for its solution.

We subsample the image to get images of lower dimension. Using lower dimension makes it easier for initializing the intensities, it also helps us to get out of suboptimal solutions.

We pass the details of the lower dimension image to higher dimension image by upsampling the mapping we used in the lower dimension.



Summary of Algorithm (Initialization)

- First we get multiple images by subsampling the input image. Each of the images will be of lower dimension than previous one.
- Then on the lowest dimension image (coarsest image) we apply **Onion Peel Initialization** on the occluded region.
- Once this is done we have updated the values of intensity and textures of the occluded region along with mapping function (shiftmap).

Algorithm (level wise update)

Now for each level in pyramid, we iteratively update for some n times or until the change becomes small:

- We update the relative mapping function (shift map) using Patch Match algorithm (existing algorithm based on some other paper).
- Using the modified mapping function (shift map) we update the values of intensities and textures in the occluded region.

Once we get the best reconstructed image for current layer of image, we need to propagate the results to next higher dimension layer. (next finer layer)

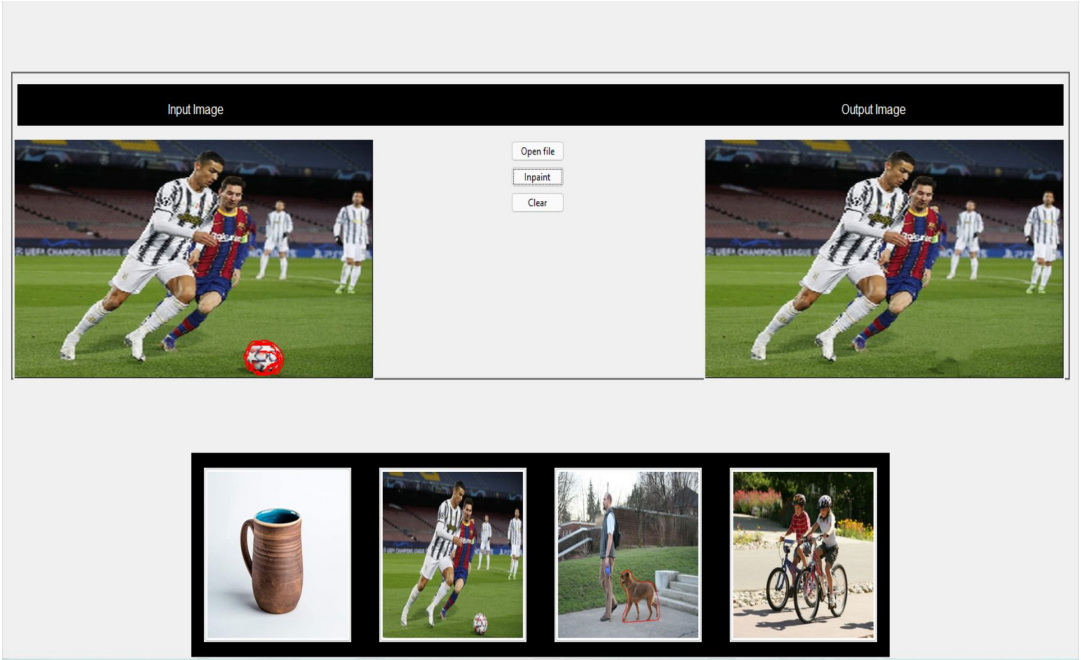
Algorithm

- We upsample the current mapping function to get the mapping function for the higher level.
- Using the new mapping function we initialize the values of intensities and textures in occluded region of the image in higher dimension level (finer level).
- And we repeat the above process of reconstructing and updating the intensities, textures of the occluded region as above.

GUI for our algorithm

We have implemented a user friendly GUI for better visualization.

We have also provided recommended images for accessing some sample images.



Guide on usage of GUI

For uploading image, either use open file button or click on one of the recommended images.

Now for selecting the occluded region use your cursor to draw the boundary of the occluded region. Make sure that the boundary is closed. And also draw the boundary slowly so the boundary is detected.

Now click on inpaint to start the algorithm. It may take bit time depending on the input size. A medium sized occluded region take couple of minutes.

Some of the Parameters

We have experimented with various parameters like:

- Number of levels of multiscaling
- Weight of texture in distance metric
- Patch size
- Number of iterations

We have tried with some other values. These are some of the best:
Patch_size = 7, weight = 50, iterations = 10, Levels = log(2*No/N) No = number of times that the occlusion needs to be eroded for it to disappear. N is Patch_size.

Dataset....

For the paper the image can be any image. Using the GUI we have provided we can upload any image for which we want to inpaint.

So as far as Dataset from paper is concerned we picked few of the images which are there in paper. They worked quite well

So Data collected by us is concerned we have randomly picked few images from online to check the generalisability of the paper.

As expected the results were awesome.

Best Results (Image taken from Paper)

Original image

Masked with occluded region

Inpainted Output

Best Results (Image taken from Paper)

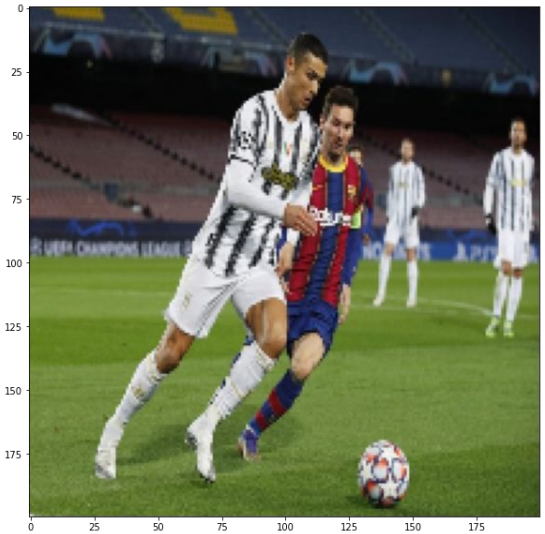
Original image

Masked with occluded region

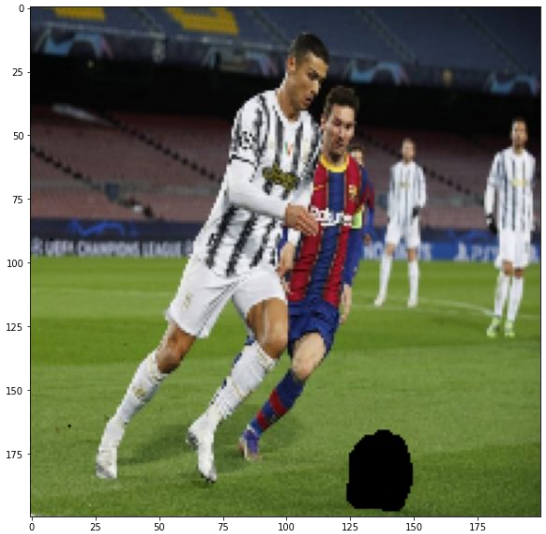
Inpainted Output

Best Results (Image collected by us)

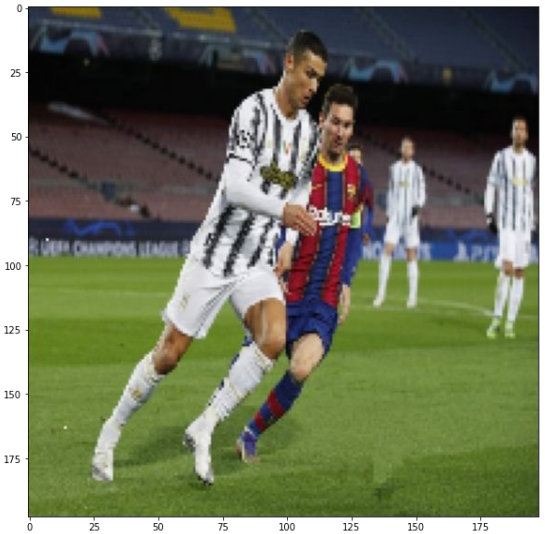
Original image



Masked with occluded region

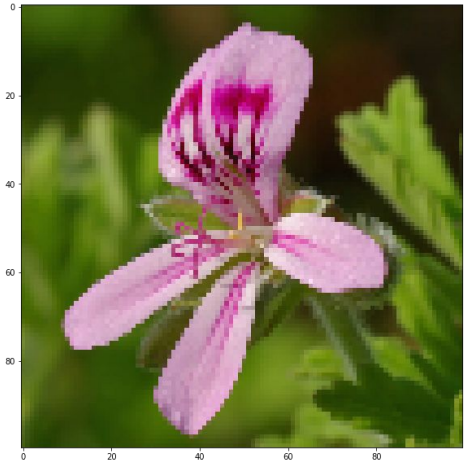


Inpainted Output

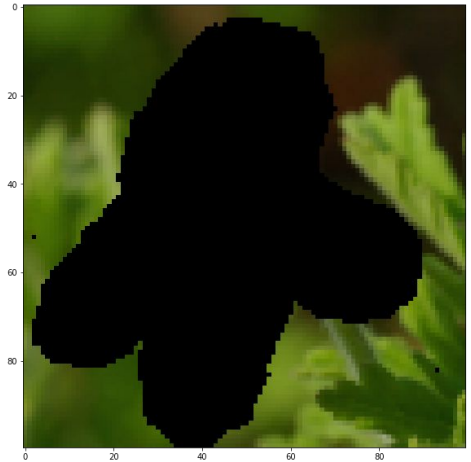


Best Results (Image collected by us)

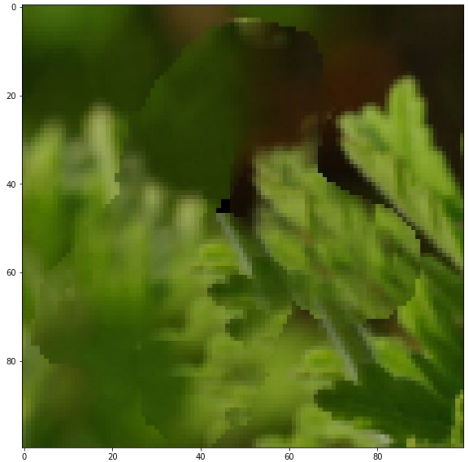
Original image



Masked with occluded region



Inpainted Output



Conclusion

Our algorithm is based on the minimization of a global patch-based functional. In order to obtain results of high quality with such an approach, several issues have been addressed.

We describe the method in which we search for nearest neighbor patches, something which is crucial for achieving results in realistic times.

We have addressed the issue of comparing textured patches.

We provide specific details concerning initialization issues and multi-scale choices, both of which have a great influence on the inpainting results.

Work Distribution

Trinadh: Onion peel initialization, optimized pyramids levels of images, modified patch distance, PPT, Fixed bugs in reconstruction parts, multiscale scheme, helped debugging gui interrupt,Testing individual components.

Harsha: Patch match, Reconstruction of images (both texture and intensities), multiscale scheme, fixed repeated interrupts in GUI, PPT, Fixed bugs in onion peel, pyramid levels, Testing of integrated components.

Sri Ram: Modified texture metric, reconstruction functions and testing for inpainting with textures, GUI (recommendation of images, getting binary image from the sketch drawn in gui),commenting code.

Nikit: GUI (basic UI design, opening images, drawing on the image to select the occluded region), Patch match algorithm, pyramid levels of the images, commenting code, tried to vectorize the code using numpy functions.

We divided into teams of two, (Trinadh , Harsha) and (Sir Ram, Nikit) we worked in live calls, With one person coding and other helping.

DEMO

IMPORTANT NOTE: The computational complexity of this papers implementation is huge (1e8-1e10). For the reason of timely response we resize the input image to 200 x 200. We also suggest you to give fairly small occluded region for good response. It may take a while, please be patient.

For this reason we have already run and shown you some of the results both using GUI and jupyter notebook.

Please feel free to use the tool to try out new images!!

Thank you