Problem Statement

Implementation of image quilting for texture synthesis and texture transfer. A texture is synthesized using a small tile (small texture image) such that it looks uniform texture, this process of creating texture is called image quilting. First, we use quilting as a fast and very simple texture synthesis algorithm which produces surprisingly good results for a wide range of textures. Second, we extend the algorithm to perform texture transfer – rendering an object with a texture taken from a different object.

Algorithms

1. Image quilting:

Algorithm1: This is a simplest algorithm. We synthesis texture patch by patch of fixed size from top to bottom and left right. Patches from the input image are selected randomly.

Algorithm2: In this algorithm a patch is overlapped to its neighbour patches. A new patch that is to be selected overlap with two patches one on top of it and one left to it. Error surface is calculated for all patches with that top and left patch. A threshold value on error is set. All those patches whose error is less than that threshold are best fit patches and we select randomly a patch from them and paste it.

Algorithm3: In this algorithm we find the minimum boundary cut using **min-cut boundary algorithm** in the overlap region of two patches. One side to the boundary contains the pixel from patch in that side and other side to the boundary pixels are taken from other patch. Dynamic Programming paradigm is used to calculate the min-cut boundary of the patches.

For first row there will be only vertical overlap area and no horizontal overlap area in patches. For first column of the patches there will be only horizontal top overlap area and no vertical overlap region. All other patches will be having horizontal as well as vertical overlap region with other patches.

Steps

1. Make a blank image and copy a random patch to top left corner
2. From all patches find the best fit patches based on error threshold we set.
3. Select a random patch from best fit patches.
4. Find min-cut boundary in the horizontal and vertical overlap region.
5. To the right side of the boundary in vertical overlap region and to bottom side of boundary in horizontal region copy pixels from selected patch.
6. Go to step 1 until texture of desired size is obtained.

2. Texture Transfer:

Because the image quilting algorithm selects output patches based on local image information it is easily extended for texture transfer by augmenting the synthesis algorithm by requiring that each patch satisfy a desired *correspondence map* which is a grayscale image, as well as satisfy the texture synthesis requirements. The correspondence map is a spatial map of some corresponding quantity over both the texture source image and a controlling target image. That quantity could include image intensity, blurred image intensity, local image orientation angles, or other derived quantities.

Steps
We have a texture image and a destination image whose texture is to be transferred. After performing texture transfer we get output image which is destination image with transferred texture as of other input image. In this algorithm we will work patch by patch too. Top left corner of the output image is set to corresponding patch in destination image.

1. Set destination image as output image
2. Make patches of a fixed size from input texture image
3. From all patches find the best fit patches based on error threshold we set using correspondence map.
4. Select a random patch from best fit patches.
5. Find min-cut boundary in the horizontal and vertical overlap region.
6. To the right side of the boundary in vertical overlap region and to bottom side of boundary in horizontal region copy pixels from selected patch.
7. Go to step 2 until texture of all the patches of destination image is transferred.

Datasets

We are using our own datasets taken from web and some of the datasets from **“UIUC”,** which can be referenced at http://www-cvr.ai.uiuc.edu/ponce_grp/data/ .

Results

1. Quilting results
   Below are the texture synthesis results with author's algorithms.

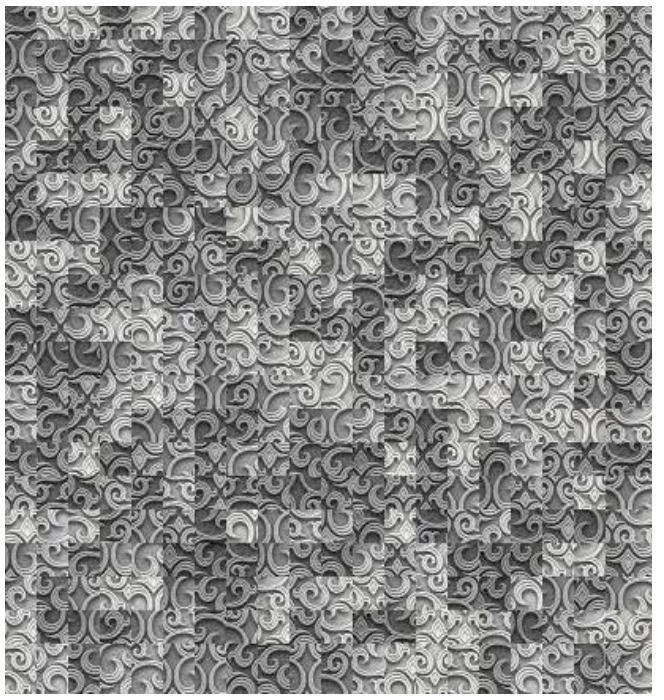
Input Texture

Output with Algorithm 1

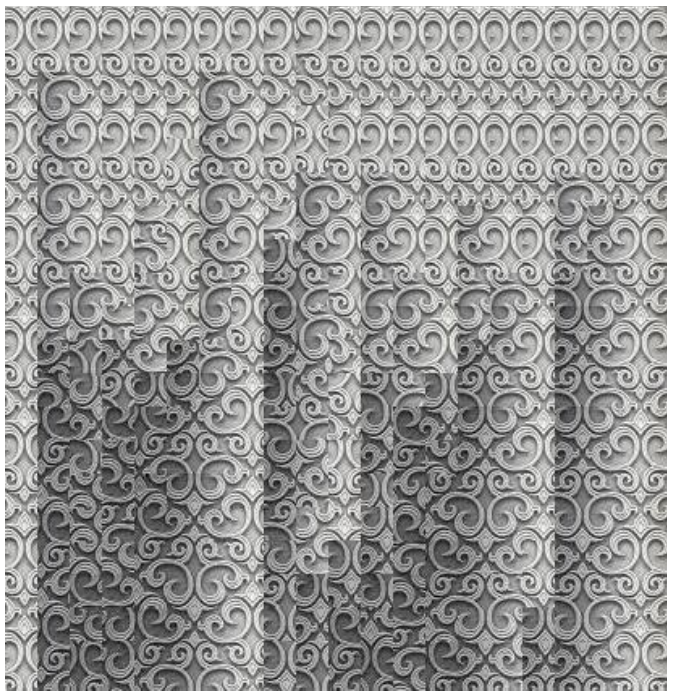
Output with Algorithm 2

Output with Algorithm 3

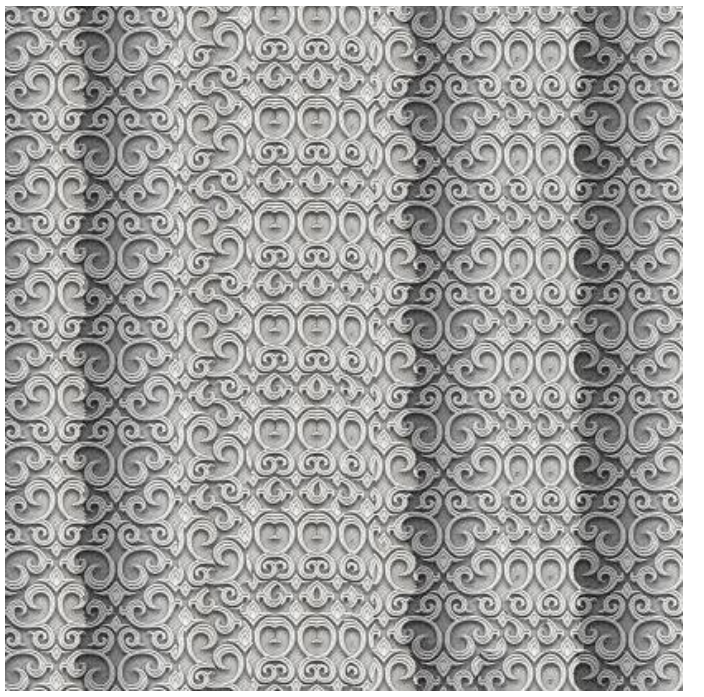
Input Texture



Output with Algorithm 1

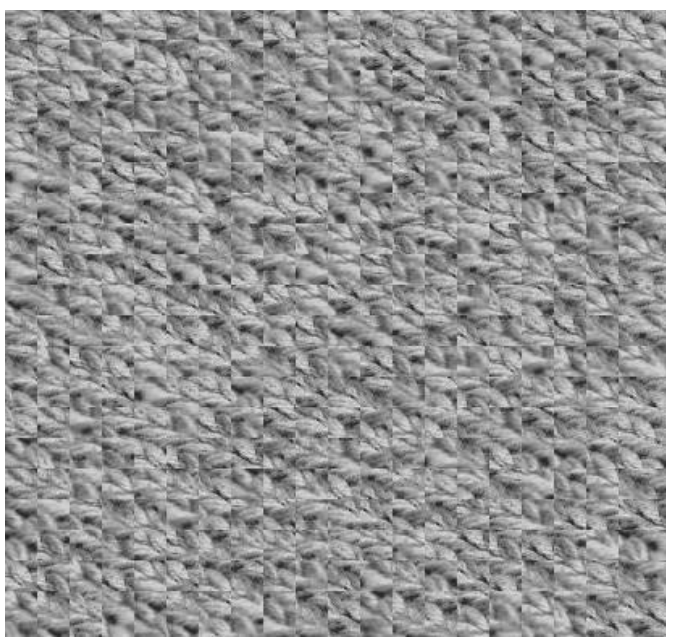


Output with Algorithm 2

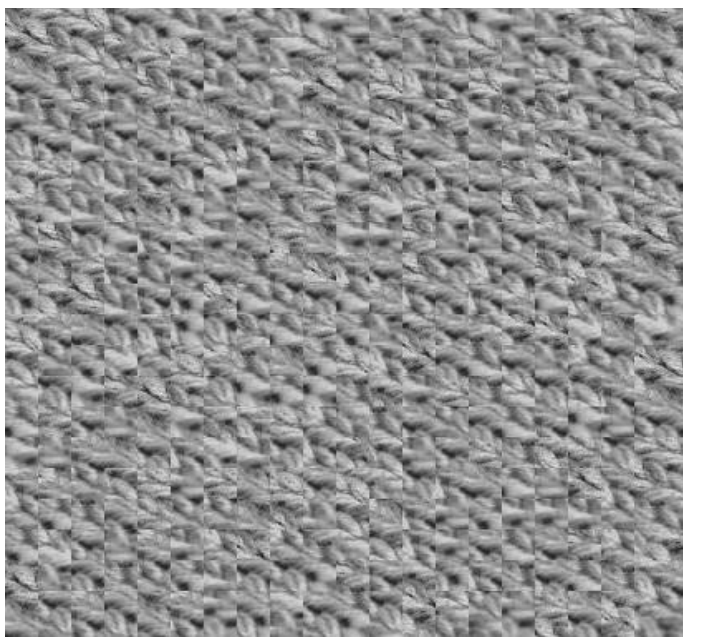


Output with Algorithm 3

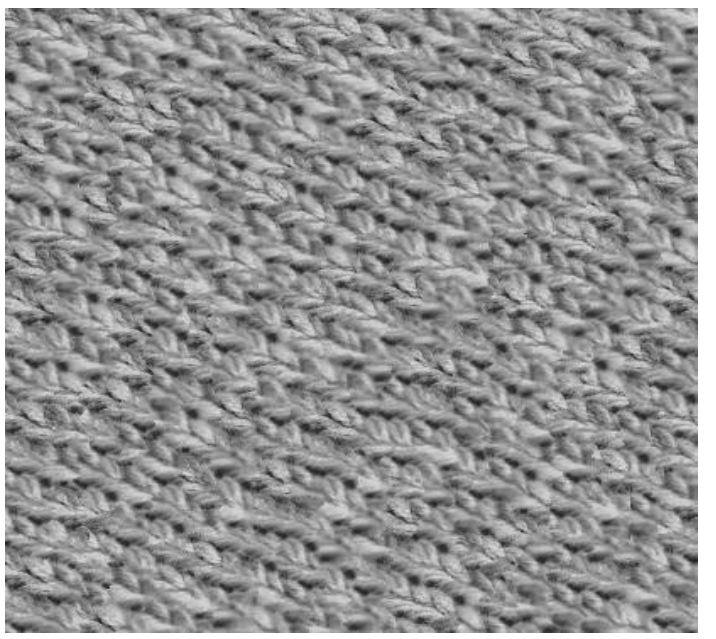
Input Texture



Output with Algorithm 1

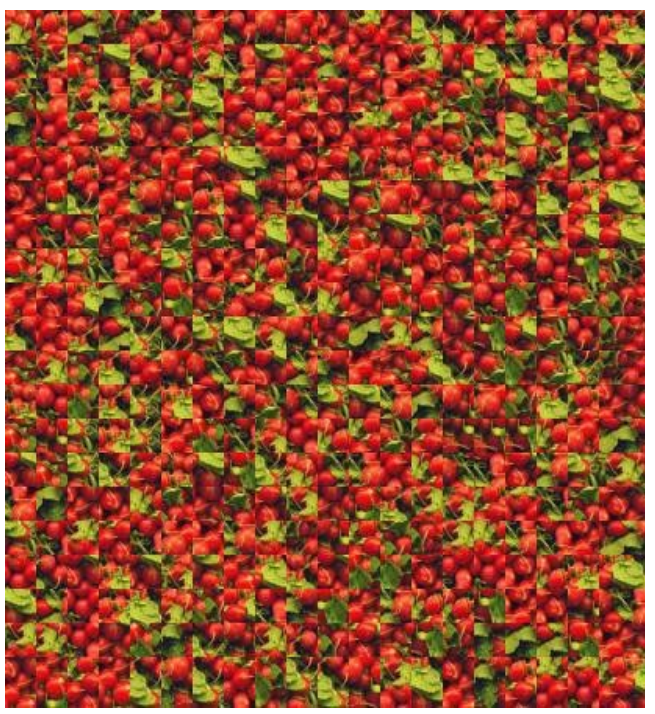


Output with Algorithm 2

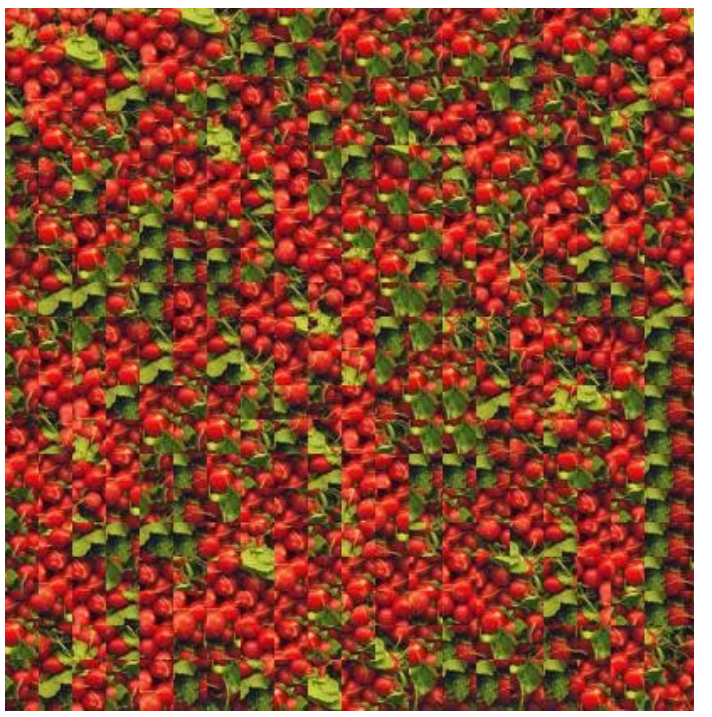


Output with Algorithm 3

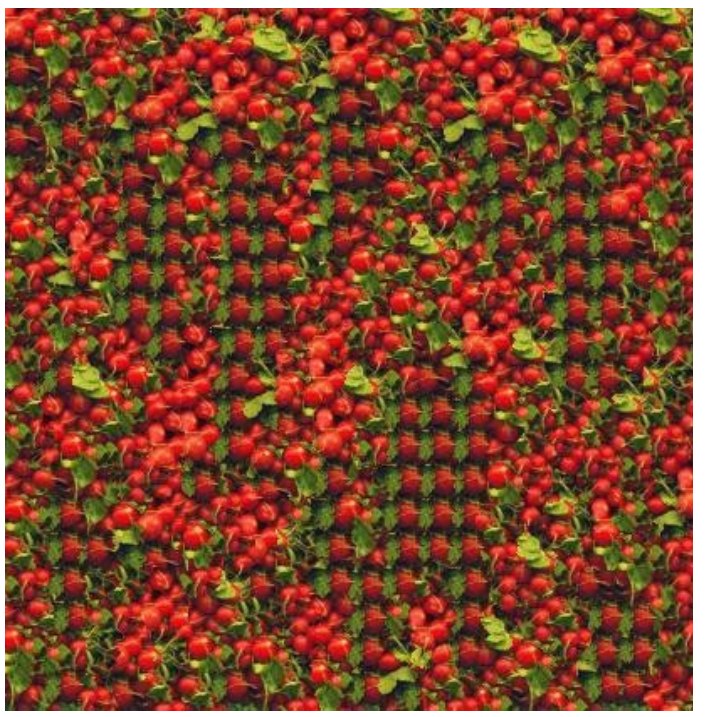
Input Texture



Output with Algorithm 1

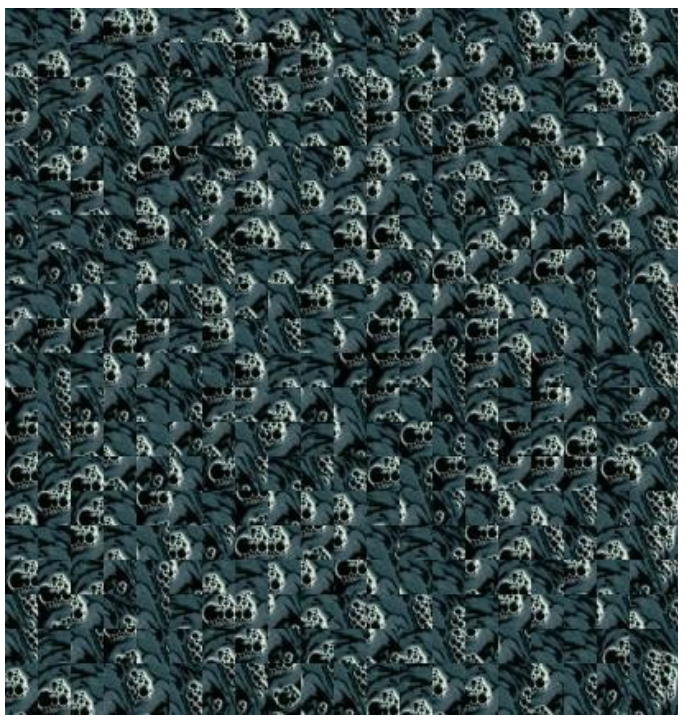


Output with Algorithm 2



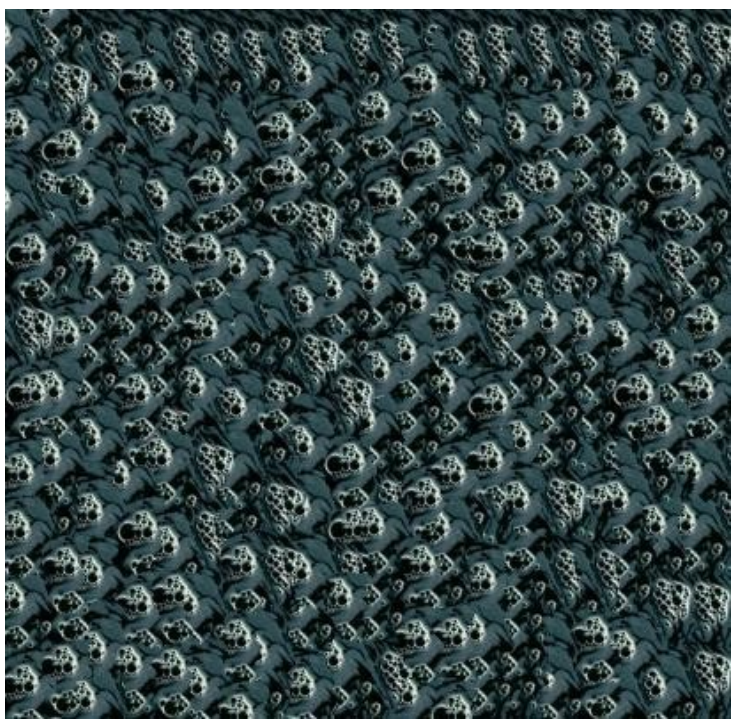
Output with Algorithm 3

Input Texture



Output with Algorithm 1

Output with Algorithm 2



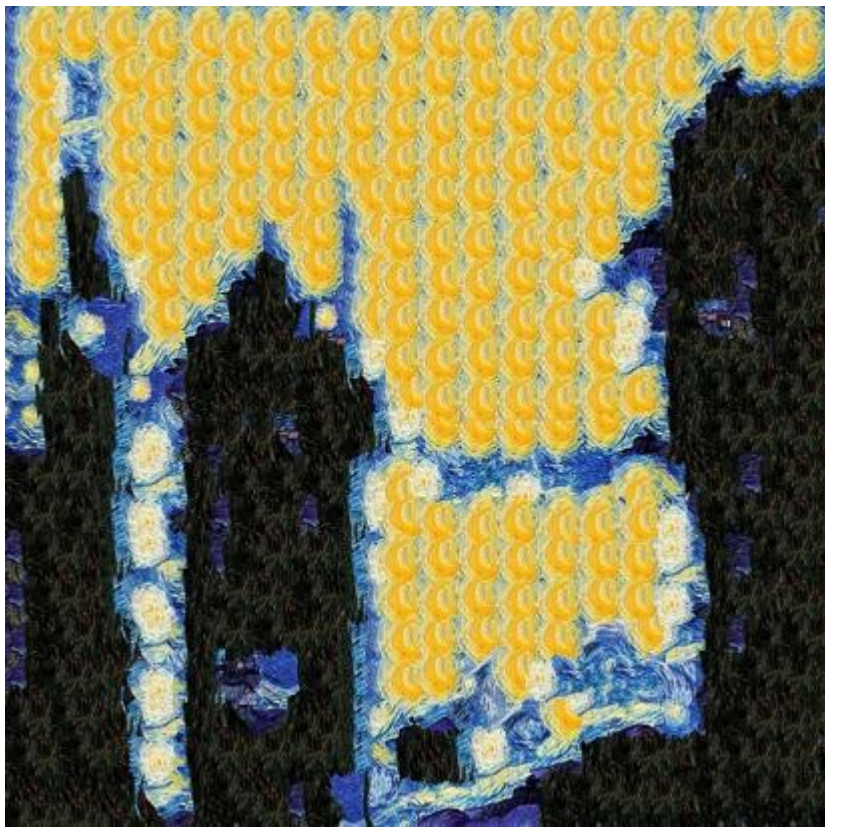
Output with Algorithm 3

2. Texture transfer results

Below are the texture transfer results with author’s algorithm

Input Texture

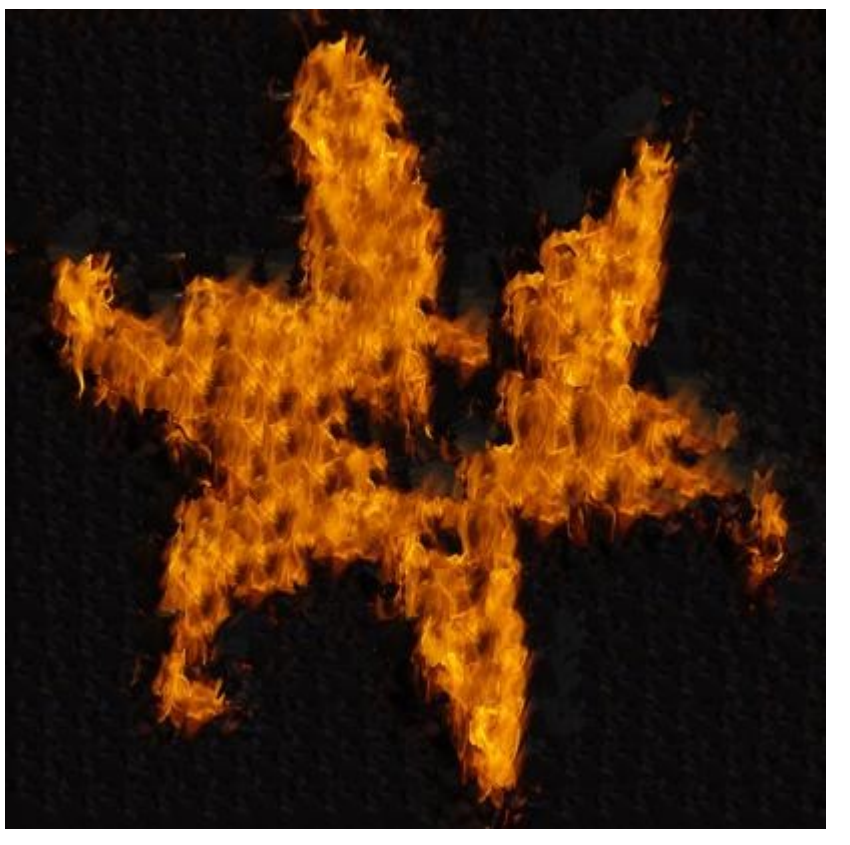
Input Image



Output Image

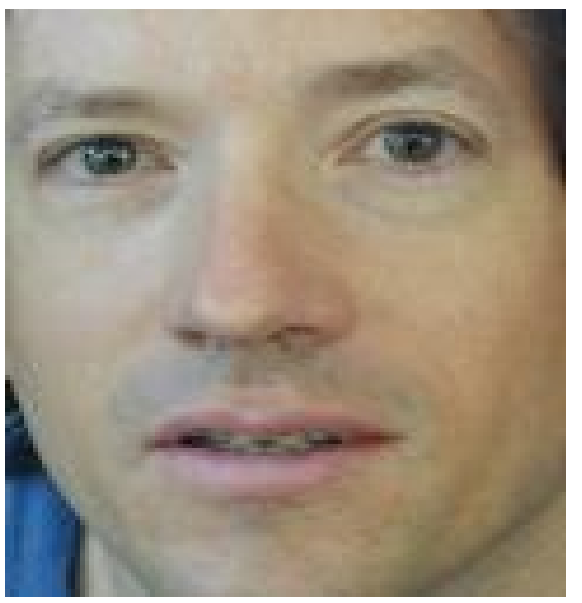
Input Texture

Input Image



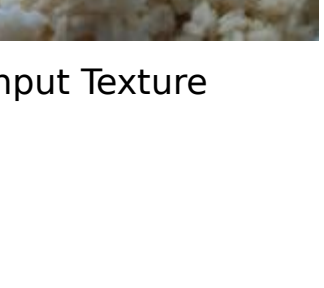
Output Image

Input Texture



Input Image

Output Image



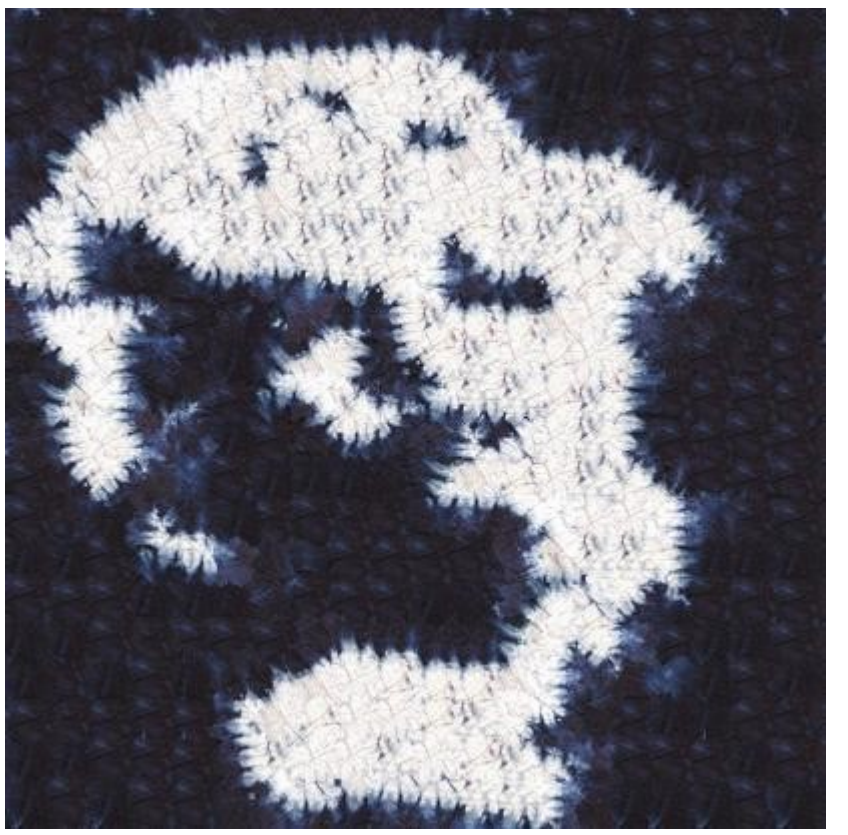
Input Texture

Input Image

Output Image

Input Texture

Input Image



Output Image

Conclusion

Algorithm 3 that is proposed by the author is much better than other algorithms . 3^{Rd} algorithm makes use of min-cut boundary. This algorithm is also extended to for texture transfer that gives promising results. Implementation of the algorithms is easy and the core part of it is min-cut algorithm which is based on dynamic programming paradigm.