# ELL715 - Digital Image Processing

Project Report


| Name | Entry number |
|---|---|
| Anirudh Panigrahi | 2018EE10446 |
| Ishvik Kumar Singh | 2018EE10616 |
| Rahul Bhola | 2018EE10489 |

# Contents

# 1 Problem statement

The objective is to derive an approach to synthesize novel images from the samples of the real world, with the same texture as the sample texture image, and use it further to transfer texture from one object to another.

# 2 Quilting

## 2.1 Introduction

The motivation behind this project is to capture samples of the real world images and using them to synthesize novel views rather than recreating the entire physical world from scratch. This can be achieved by image-based texture synthesis algorithms. Such algorithms take a sample of texture and generate an unlimited amount of image data which, while not exactly like the original, will be perceived by the humans to be the same texture. Clearly, this approach is far more superior than re-sampling images to create larger texture patterns.

A naive way to approach image-based texture synthesis is to divide the original sample image is to randomly pick blocks of specific size from the image and patch them together. This can be further improved by by allowing overlap between blocks and choosing each new block so as to "agree" with its neighbours in the region of overlap. However, these approaches do not produce good results, as shown in [1]. Thus, we implement a technique called *image quilting*.

In image quilting, we pick square patches from the sample image. To synthesize a new image texture, we'll tile the patches next to each other, allowing some overlap in the placement of blocks. Each new block is chosen in such a way that it agrees with its neighbours along the region of overlap. At this point, the edges between the blocks are still quite noticeable. We can employ averaging or frequency domain filters to smooth the edges but this leads to loss of detail. A better method is to let the blocks have ragged edges which will allow them to better approximate the features in the texture.

## 2.2 Algorithm

Consider a sample texture $I_o$ of dimensions $M_o \times N_o$. From the texture image, patches or blocks of dimensions $w_p \times w_p$ can be obtained. These blocks are patched onto the output texture $I_s$ allowing a overlap width of $w_o$. The algorithm for obtaining suitable blocks from $I_o$ and then patching them onto $I_s$ is illustrated in Algorithm 1.

**Algorithm 1:** Image Quilting

**Input:** Sample texture $I_0$, patch size $w_p$, overlap size $w_o$, error tolerance parameter $\epsilon$, output image size $h_{new} \times w_{new}$

**Output:** Final texture synthesized, $I_s$

1 Initialize $I_s$ to an image with 0 pixels except the top-left $w_p \times w_p$ block(first patch), which is filled with a patch randomly selected from $I_0$

2 **foreach** *patch* $P_{old}$ *in* $I_s$ **do**

3 Use Algorithm 2 to select a patch $P_{selected} \in I_0$ that best overlaps with $P_{old}$

4 Use Algorithm 3 to find the minimum error boundary cut between $P_{old}$ and $P_{selected}$

5 Replace $P_{old}$ in $I_s$ with a new patch which consists of pixels of $P_{old}$ to the left and above the minimum error boundary cut, and pixels of $P_{selected}$ to the right and below it

6 **return** $I_s$

In Algorithm 1, in each iteration of the loop, we use a patch $P_{old}$. $P_{old}$ denotes the partially defined patch (defined only in the overlap region, 0 everywhere else) which needs to needs to be stitched together with a

similar patch $P_{selected}$ obtained from $I_o$. The method of obtaining $P_{selected}$ is given in Algorithm 2

In Algorithm 2, we need to compute the squared distance matrix (or overlap error matrix) $D$. The squared distance matrix $D$ contains at each position $(m, n)$ the overlap error between $P_{old}$ and the patch from $I_o$, whose top-left corner is $(m, n)$, according to some binary weight matrix $Q$ that equals one in the overlap region and zero otherwise. More precisely, $D(m, n)$, for all $(m, n)\, \epsilon\, \{0, ..., M_o - w_p\} \times \{0, ..., N_o - w_p\}$, is given by

$$D(m,n) = \sum_{i,j=0}^{w_p-1} Q(i,j)(P_{old}(i,j) - I_0(m+i, n+j))^2 \tag{1}$$

**Algorithm 2:** Patch Selection

**Input:** Sample texture $I_0$, patch under construction $P_{old}$, patch size $w_p$, error tolerance $\epsilon > 0$ binary weight $Q$ defining the overlap region

**Output:** Position of selected patch $(m, n)$(coordinates of top-left pixel of selected patch in $I_0$)

1 Compute the matrix D containing the MSE overlap errors of each patch of $I_0$ with $P_{old}$(given by Equation 1)

2 $d_{min} \leftarrow \min_{u,v} D(u, v)$

3 Randomly choose a patch $P_{selected}$ from among the set of patches $\{P|$top-left coordinates of P, $(u, v)$are such that $D(u, v) < (1 + \epsilon)d_{min}\}$

4 **return** $P_{selected}$

Minimum error boundary cut is the main contribution of the Efros-Freeman algorithm [1]. The patch selection algorithm 2 gives a patch $P_{selected}$ of $I_o$ that is similar to the partially defined current patch $P_{old}$ in their overlap region. The overlap region can of three types: vertical, horizontal and L-shaped overlap. To get the final patch $P$, $P_{old}$ and $P_{selected}$ have to be combined.

Now, the objective is to find a path in the overlap region such that where the transition between $P_{old}$ and $P_{selected}$ along the path is minimal. For simplicity, only simple forward paths, from one end to the other of the overlap region, have been considered in order to be able to use linear programming.

Algorithm 3 shows how we find the minimum error boundary cut, by tackling the 3 possible overlaps seperately, and utilizing dynamic programming.

**Algorithm 3:** Minimum Error Boundary Cut

**Input:** patch $P_{old}$ (from $I_s$), patch $P_{selected}$ (from $I_0$), patch size $w_p$, overlap size $w_o$, overlap type

**Output:** Path of pixel positions $\gamma = \gamma_0, \gamma_1, ...\gamma_{w_p-1}$ across the overlap region corresponding to the minimum error boundary cut

**switch** *overlap type* **do**
- **case** *vertical*
  - Identify the vertical overlapping regions of $P_{old}$ ($P_{old}^{ovl}$) and of $P_{selected}$ ($P_{selected}^{ovl}$)
  - Compute the squared error $e(i,j) \leftarrow (P_{old}^{ovl}(i,j) - P_{selected}^{ovl}(i,j))^2$
  - Compute the minimum cumulative squared error $E_v$ as follows**:**
    - $E_v(0,j) \leftarrow e(0,j)\ \forall j \in \{0, ..., w_o - 1\}$
    - **for** $i = 1$ *to* $w_p - 1$ **do**
      - $E_v(i,j) \leftarrow e(i,j) + \min(E_v(i-1,j-1), E_v(i-1,j), E_v(i-1,j+1))\ \forall j \in \{0, ..., w_o - 1\}$
  - Now find the minimum error boundary cut using $E_v$ as follows**:**
    - $\gamma_{w_p-1} \leftarrow (w_p - 1, argmin_j(E_v(w_p - 1, j)))$
    - **for** $i = w_p - 2$ *to* $0$ **do**
      - $j \leftarrow \gamma_{i+1}^2$ ($\gamma_{i+1}^1$ is equal to $i+1$)
      - $\gamma_i \leftarrow (i, argmin(E_v(i, j-1), E_v(i,j), E_v(i, j+1)))$
  - $\gamma \leftarrow \{\gamma_0, \gamma_1, ...\gamma_{w_p-1}\}$
  - **return** $\gamma$
- **case** *horizontal*
  - Identify the horizontal overlapping regions of $P_{old}$ ($P_{old}^{ovl}$) and of $P_{selected}$ ($P_{selected}^{ovl}$)
  - Compute the squared error $e(i,j) \leftarrow (P_{old}^{ovl}(i,j) - P_{selected}^{ovl}(i,j))^2$
  - Compute the minimum cumulative squared error $E_h$ as follows**:**
    - $E_h(i,0) \leftarrow e(i,0)\ \forall i \in \{0, ..., w_o - 1\}$
    - **for** $j = 1$ *to* $w_p - 1$ **do**
      - $E_h(i,j) \leftarrow e(i,j) + \min(E_h(i-1,j-1), E_h(i,j-1), E_h(i+1,j-1)\ \forall i \in \{0, ..., w_o - 1\}$
  - Now find the minimum error boundary cut using $E_h$ as follows**:**
    - $\gamma_{w_p-1} \leftarrow (argmin_i(E_h(i, w_p - 1)), w_p - 1)$
    - **for** $j = w_p - 2$ *to* $0$ **do**
      - $i \leftarrow \gamma_{j+1}^1$ ($\gamma_{j+1}^2$ is equal to $j+1$)
      - $\gamma_j \leftarrow (argmin(E_h(i-1,j), E_h(i,j), E_h(i+1,j)), j)$
  - $\gamma \leftarrow \{\gamma_0, \gamma_1, ...\gamma_{w_p-1}\}$
  - **return** $\gamma$
- **case** *L-shaped*
  - Identify the vertical overlapping regions of $P_{old}$ ($P_{old}^{ovl-v}$) and of $P_{selected}$ ($P_{selected}^{ovl-v}$)
  - Identify the horizontal overlapping regions of $P_{old}$ ($P_{old}^{ovl-h}$) and of $P_{selected}$ ($P_{selected}^{ovl-h}$)
  - Compute $E_v$ and $E_h$, as done in the horizontal/vertical cases**:**
    - We calculate $E_v$ and $E_h$ by iterating in the opposite direction in this case
    - This means for $E_h$, we iterate from $j = w_p - 1$ to 0; for $E_v$ we iterate from $i = w_p - 1$ to 0
  - Now find the minimum error boundary cut using $E_v$ and $E_h$ as follows**:**
    - $i_0 \leftarrow argmin_i(E_v(i,i) + E_h(i,i) - e(i,i))$, $i \in 0, ..., w_o - 1$
    - $\gamma_{w_p-i_0-1} \leftarrow (i_0, i_0)$
    - **for** $j = 1$ *to* $w_p - i_0 - 1$ **do**
      - $i_h \leftarrow \gamma_{w_p-i_0-1+j-1}^1$
      - $j_v \leftarrow \gamma_{w_p-i_0-1-j+1}^2$
      - $\gamma_{w_p-i_0-1+j} \leftarrow (argmin(E_h(i_h - 1, j + i_0), E_h(i_h, j + i_0), E_h(i_h + 1, j + i_0)), j + i_0)$
      - $\gamma_{w_p-i_0-1-j} \leftarrow (j + i_0, argmin(E_v(j + i_0, j_v - 1), E_v(j + i_0, j_v), E_v(j + i_0, j_v + 1)))$
  - $\gamma \leftarrow \{\gamma_0, \gamma_1, ...\gamma_{2\cdot(w_p-i_0)}\}$
  - **return** $\gamma$

### 2.3 Optimizations using DFT [2]

Here, the objective is to speed-up the computation by calculating the squared distance matrix $D(m,n)$ using the Fast-Fourier-Transform (FFT). The naive computation of

$$D(m,n) = \sum_{i,j=0}^{w_p-1} Q(i,j)(P_{old}(i,j) - I_0(m+i,n+j))^2 \tag{2}$$

takes no. of computations of the order of $w_p^2 M_0 N_0$, with the patch width $w_p$ being 30-40 pixels. This can be reduced to a fixed number of FFT Computations of $M_0 \times N_0$ images, bringing the no. of operations down to $M_0 N_0 log(M_0 N_0)$. This makes it independent of the patch size.

**Cross Correlation:** Consider two images $A$ and $B$, their cross correlation $\Gamma(A,B)$ is given as

$$\Gamma(A,B)(m,n) = \sum_{i=0}^{M_0-1} \sum_{j=0}^{N_0-1} A(i,j)B(m+i,n+j) \tag{3}$$

Let $\hat{A}$ denote the reflection of A about origin, i.e., $\hat{A}(i,j) = A(-i,-j)$. We see that the cross-correlation can be written in terms of convolution product as follows:

$$\Gamma(A,B)(m,n) = \hat{A} * B(m,n) \tag{4}$$

Now, let $\tilde{I}$ denote the extension of a smaller size image $I$ of size $w_1 \times w_2$ into an $M_0 \times N_0$ image by zero padding as follows

$$\tilde{I}(m,n) = \begin{cases} I(m,n) & \text{if } (m,n) \in \{0,...,w_1-1\} \times \{0,...,w_2-1\} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In a similar manner, let us denote the extensions of $P_{old}$ and $Q$ by $\tilde{P_{old}}$ and $\tilde{Q}$ respectively. On performing some trivial algebra, we arrive at the following expression for the error-matrix $D$

$$D = \sum_{i,j=0}^{w_p-1} P_{old}(i,j)^2 - 2\Gamma(\tilde{P_{old}}, I_0) + \Gamma(\tilde{Q}, I_0 \circ I_0) \tag{6}$$

where $\circ$ denotes the hadamard product. Since the third term is not dependant on the patch in consideration, it can be pre-computed and stored in memory, along with the DFT of the original image $I_0$.

Note that the matrix $D$ computed above is defined for all $(m,n) \in \{0,...,M_0-1\} \times \{0,...,N_0-1\}$. However, values for coordinates outside $\{0,...,M_0-w_p\} \times \{0,...,N_0-w_p\}$ must be discarded while searching for $d_{min}$ as they originate from periodic boundary conditions.

Since FFT computation is prone to rounding error, it can happen that for some $(m,n)$, $D(m,n)$ becomes negative, which means that $d_{min} < 0$. This can cause Algorithm 2 to fail to give reasonable output. To avoid this, all $D(m,n) < 1$ are set to 1.

## 3 Texture Transfer

### 3.1 Introduction

The image quilting algorithm selects output patches based on local image information. Therefore, it is particularly well suited for *texture transfer*.

We have extended the current the quilting algorithm for texture transfer by requiring each patch to satisfy a desired correspondence map, as well as satisfy the texture synthesis requirements. The correspondence map

is a spatial map of some corresponding quantity, like intensity, luminance, etc, over both the texture source image and a controlling target image.

Specifically, the error term used in Algorithm 2 is modified for texture transfer. The new error term is a weighted sum of overlap error $D$ and the correspondence error $C$. The correspondence error is mean squared distance between the the selected patch of the texture image and the patch $P_{target}$ in the target image having same coordinates in it as $P_{old}$ has in $I_0$. Therefore, the new squared matrix $D'$ at coordinates $(m, n)$, where $(m, n)\ \epsilon\ \{0, ..., M_o - w_p\}\ \times\ \{0, ..., N_o - w_p\}$, is given by

$$D'(m,n) = \alpha \cdot D(m,n) + (1-\alpha) \cdot \sum_{i,j=0}^{w_p-1} (P_{target}(i,j) - I_0(m+i, n+j))^2 \tag{7}$$

In equation 7, $D(m, n)$ can be calculated using equation 1, and the second term denotes the correspondence error $C(m, n)$. Further, $\alpha$ is the weight parameter such that $0 < \alpha\ < 1$.

Algorithm 4 demonstrates how we modify the patch selection algorithm used for quilting (Algorithm 2), by adding a weighted portion of the correspondence error (we compute $D'$ instead of $D$, and use it to find the best overlapping patch). Rest of the steps as shown in Algorithm 1 remain the same.

**Algorithm 4:** Patch Selection - Texture Transfer

**Input:** Sample texture $I_0$, patch under construction $P_{old}$, target patch from target image $P_{target}$, patch size $w_p$, error tolerance $\epsilon > 0$, binary weight $Q$ defining the overlap region, weight parameter $\alpha$

**Output:** Position of selected patch $(m, n)$(coordinates of top-left pixel of selected patch in $I_0$)

1 Compute the matrix D containing the MSE overlap errors of each patch of $I_0$ with $P_{old}$(given by Equation 1)

2 Compute correspondence error matrix C as the MSE error between each patch of $I_0$ and $P_{target}$

3 $D \leftarrow \alpha \cdot D + (1-\alpha) \cdot C$

4 $d_{min} \leftarrow \min_{u,v} D(u, v)$

5 Randomly choose a patch $P_{selected}$ from among the set of patches $\{P|$top-left coordinates of P, $(u, v)$are such that $D(u, v) < (1 + \epsilon)d_{min}\}$

6 **return** $P_{selected}$

### 3.2 Optimizations using DFT

We follow a similar procedure as we did in quilting. Here, we use an error matrix given by Equation 7. This can be rewritten in the following form:

$$D' = \alpha \cdot \left( \sum_{i,j=0}^{w_p-1} \cdot P_{old}(i,j)^2 - 2\Gamma(\tilde{P_{old}}, I_0) + \Gamma(\tilde{Q}, I_0 \circ I_0) \right) + (1-\alpha) \cdot \left( \sum_{i,j=0}^{w_p-1} \cdot P_{target}(i,j)^2 - 2\Gamma(\tilde{P_{target}}, I_0) + \Gamma(\tilde{Q_1}, I_0 \circ I_0) \right) \tag{8}$$

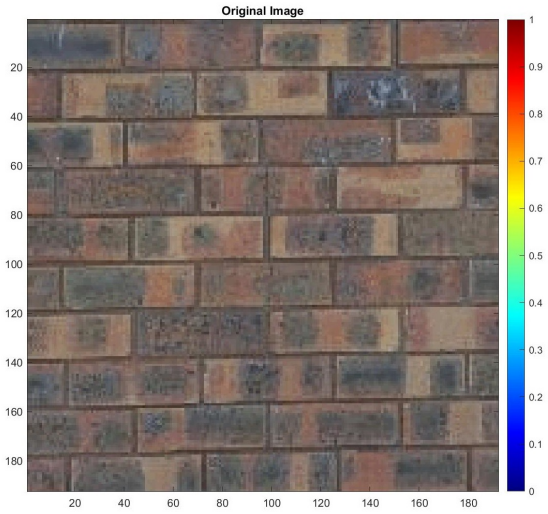
where $Q_1$ is a binary matrix of size $w_p \times w_p$, having all elements as 1.

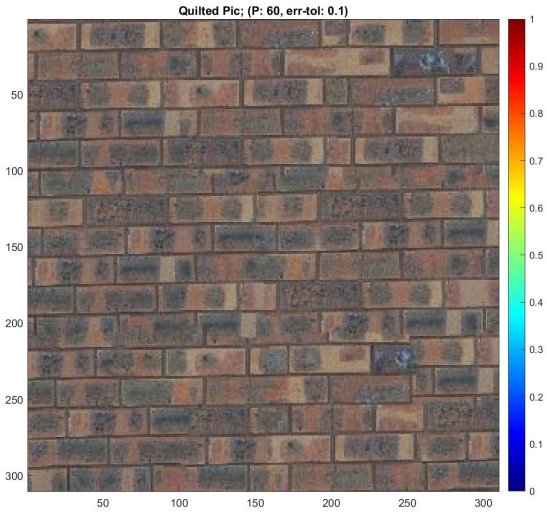
## 4 Results

In all the below results, we show the parameters we used to obtain the final images above those images.

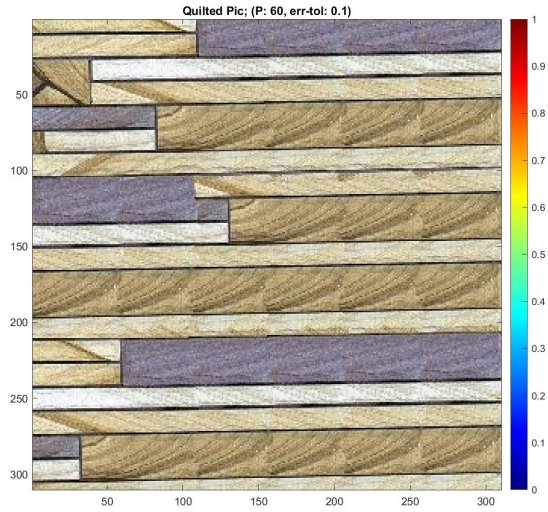
- `P` = patch size in pixels
- `err-tol` = error tolerance $\epsilon$ as used by the Patch selection algorithm (2 or 4)
- `alpha` = Weight $0 \leq \alpha \leq 1$ used in the Algorithm 4, in texture transfer.

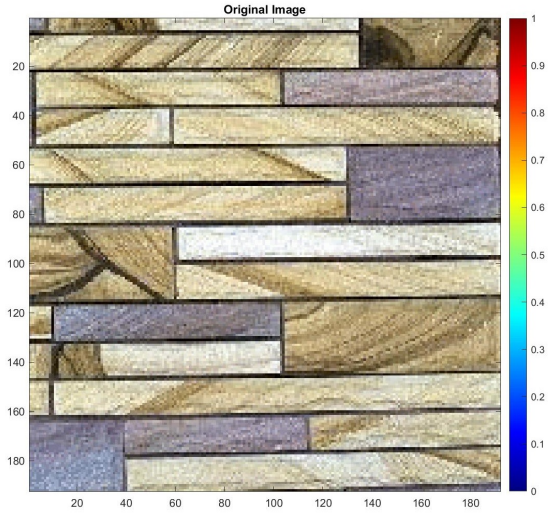
## 4.1 Quilting

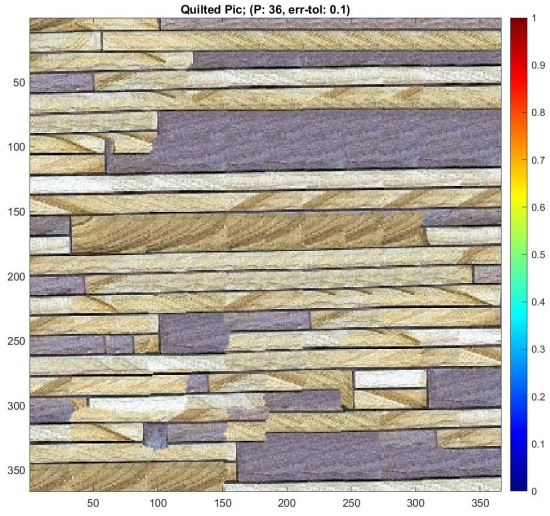

Original Image

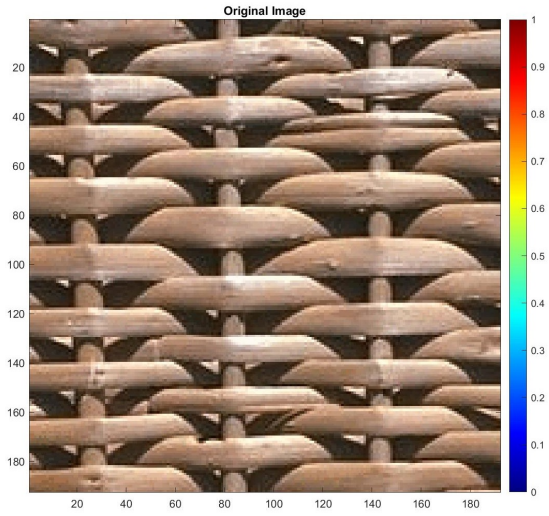

Quilted Pic; (P: 60, err-tol: 0.1)

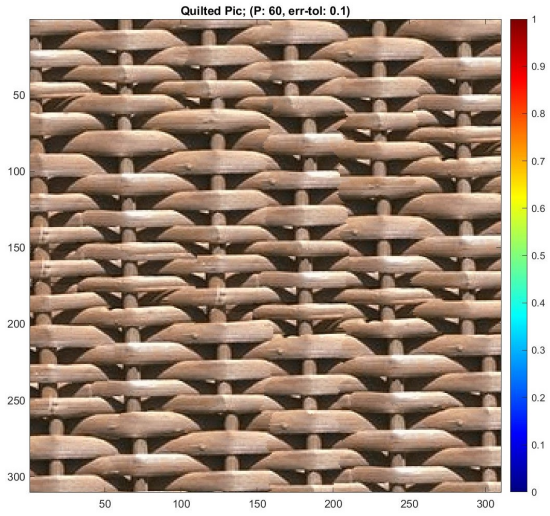

Original Image

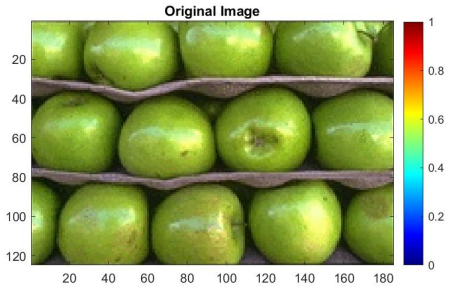

Quilted Pic; (P: 60, err-tol: 0.1)

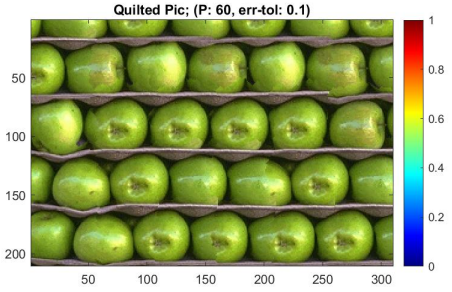


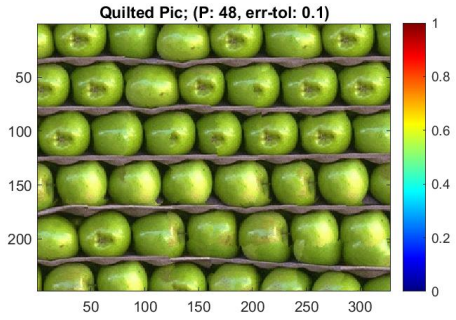



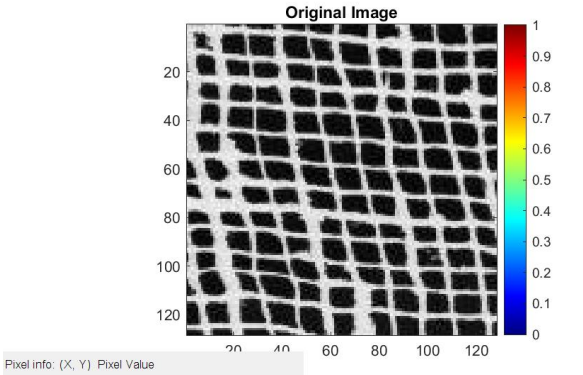



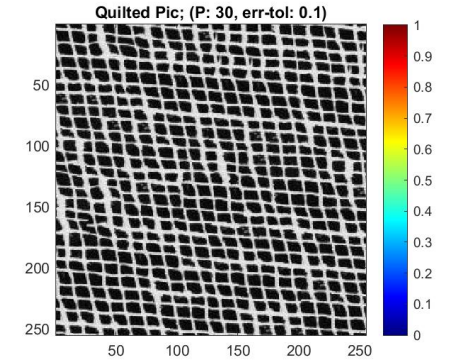



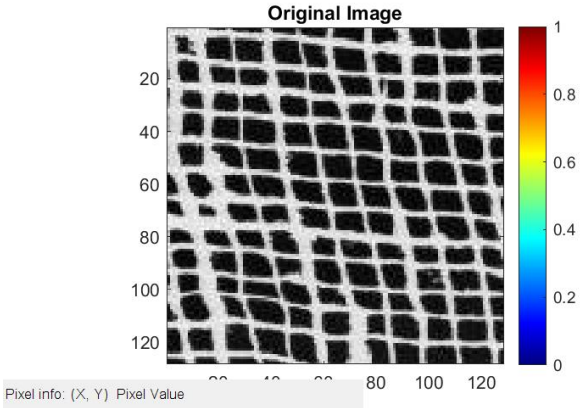

Pixel info. (X, Y) Pixel Value

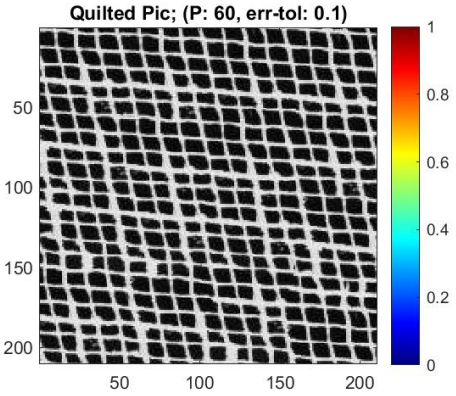

Original Image
20
40
60
80
100
120
50
100
150
1
0.8
0.6
0.4
0.2
0
Pixel info: (X, Y) Pixel Value

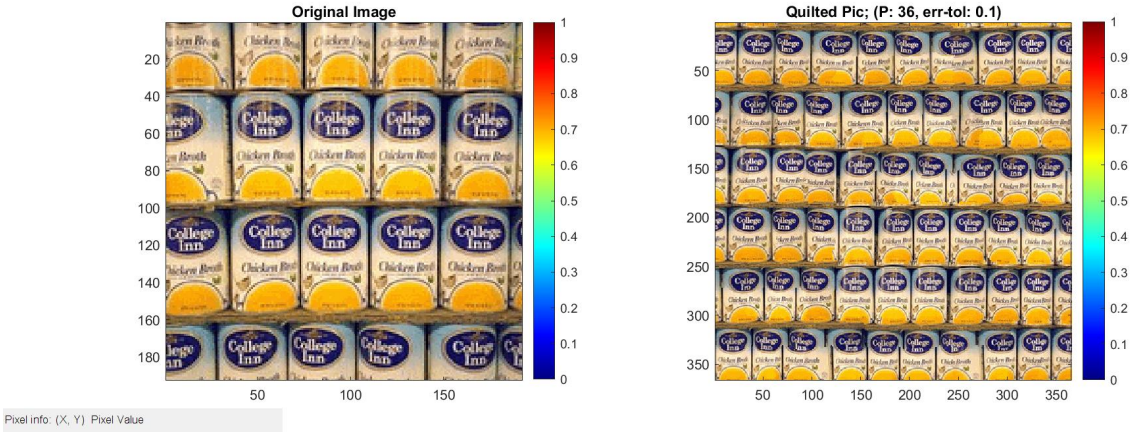

Quilted Pic; (P: 48, err-tol: 0.1)
50
100
150
200
50
100
150
200
250
300
1
0.8
0.6
0.4
0.2
0


Original Image
20
40
60
80
100
120
60
80
100
120
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Pixel info: (X, Y) Pixel Value

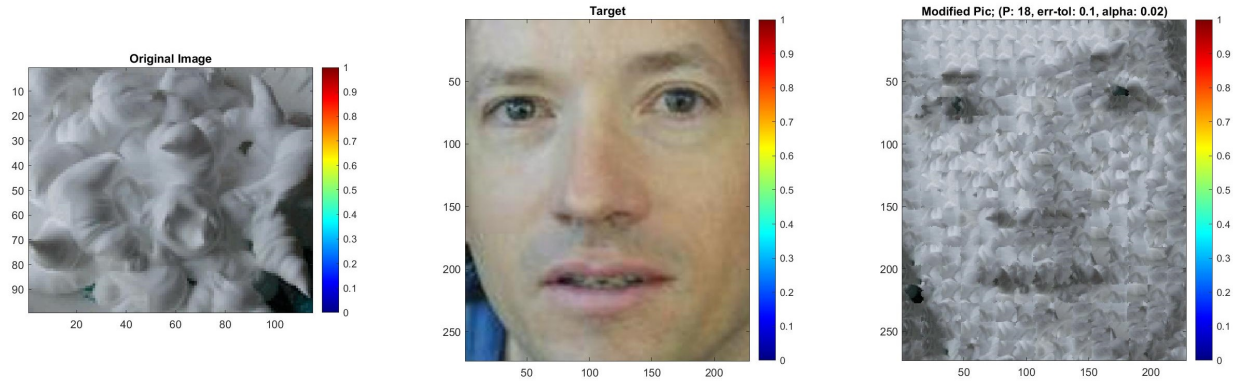

Quilted Pic; (P: 30, err-tol: 0.1)
50
100
150
200
250
50
100
150
200
250
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

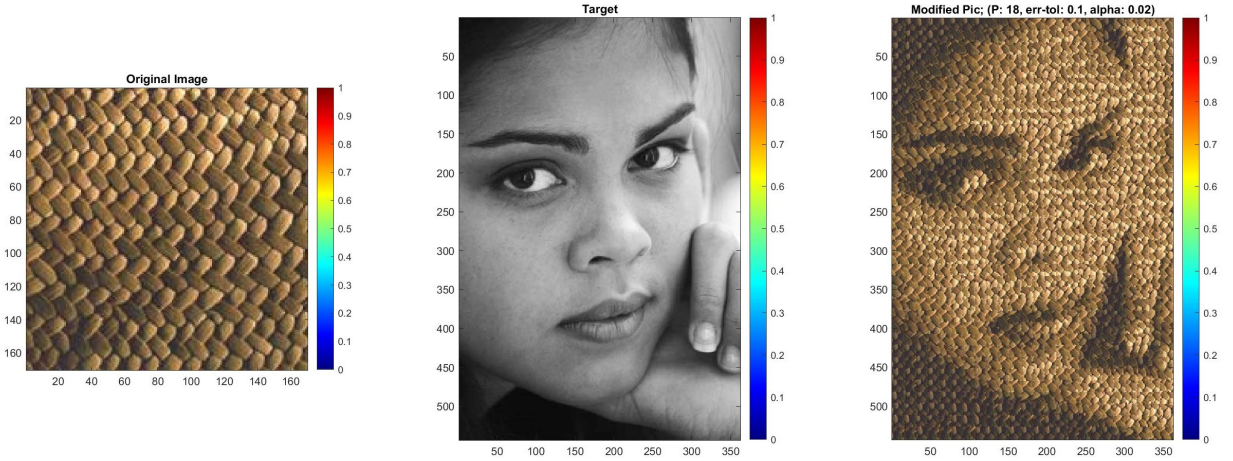

Original Image
20
40
60
80
100
120
80
100
120
1
0.8
0.6
0.4
0.2
0
Pixel info: (X, Y) Pixel Value

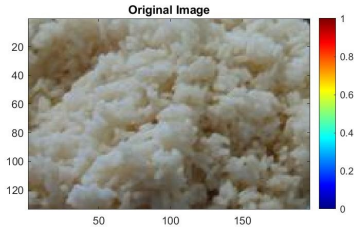

Quilted Pic; (P: 60, err-tol: 0.1)
50
100
150
200
50
100
150
200
1
0.8
0.6
0.4
0.2
0

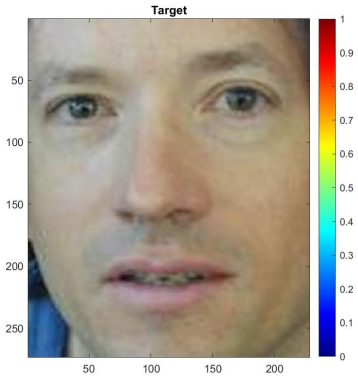


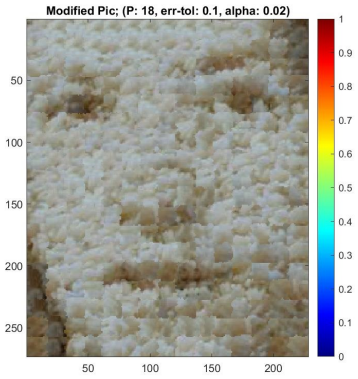

## 4.2 Texture transfer

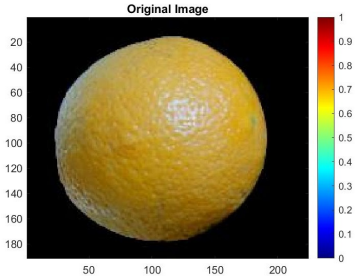


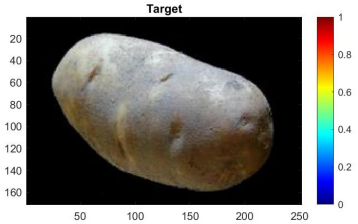



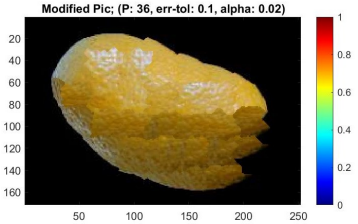

Original Image

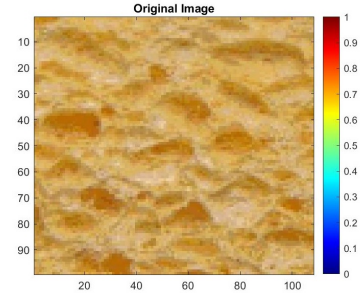

Target

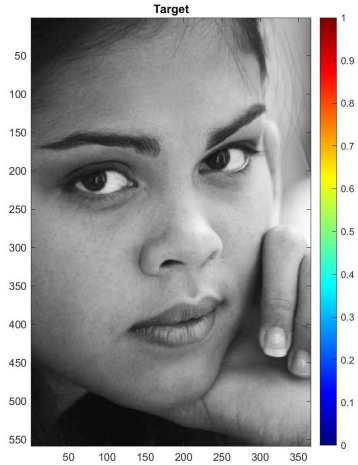

Modified Pic; (P: 18, err-tol: 0.1, alpha: 0.02)

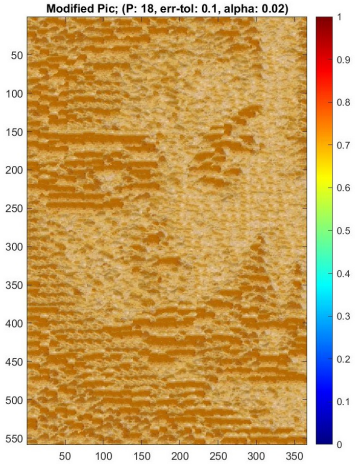

Original Image

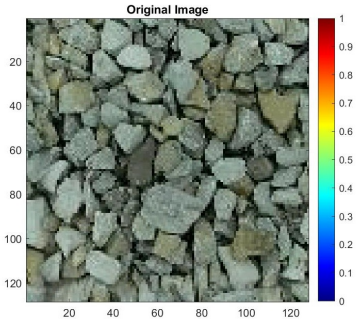

Target

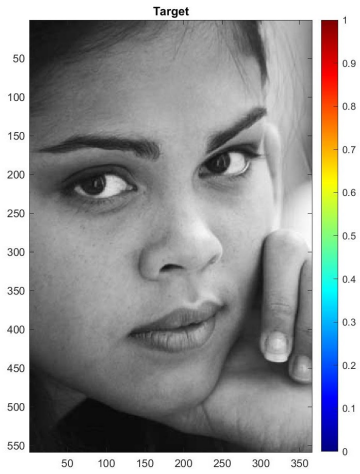

Modified Pic; (P: 36, err-tol: 0.1, alpha: 0.02)

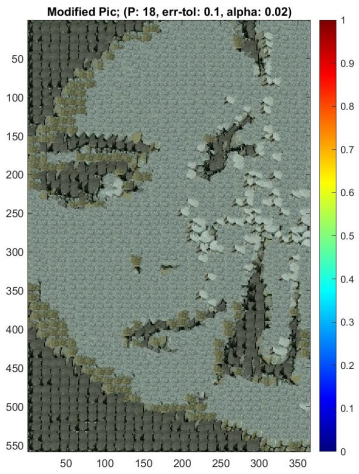

Original Image


Target


Modified Pic; (P: 18, err-tol: 0.1, alpha: 0.02)




# 5 Additional Points

Following are some suggestions to further optimize and improve the output performance of the algorithms for image quilting and texture transfer:

1. **Parallelization**
   In image quilting and texture transfer, new patch assignment is only correlated to some previous synthesis steps and not to all of them. To illustrate this consider the initial steps of the image quilting algorithm. The algorithm first adds the seed patch to the top-left followed by a second patch on the right. When adding the third patch, the first patch of the second row can be created at the same time since it only depends on the one that is on top of it and on its top-right. In general the synthesis of a new row of patches can be started as soon as two patches of the previous row have been synthesized.

   Using the above approach, the algorithm for quilting can be significantly accelerated when running on multi-core platforms.

2. **Multiple Passes for Texture Transfer**
   Sometimes, the texture transfer algorithm may not produce a visually pleasing result in the one synthesis pass because of the additional constraints as compared to image quilting algorithm. In such a case, to boost the output performance, we iterate over the synthesized image, reducing the patch size in each iteration. The modification that must be added to the non-iterative version is that in satisfying the local texture constraints we match the patches with the neighbours in the overlap region, as well as with the block at the same location in the previously synthesized image. The weight parameter $a$ in the error term can also be varied as a function of iteration number.

# References

[1] Alexei A. Efros and William T. Freeman. "Image Quilting for Texture Synthesis and Transfer". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 341–346. ISBN: 158113374X. DOI: 10.1145/383259.383296.

[2] Lara Raad Cisa and Bruno Galerne. "Efros and Freeman Image Quilting Algorithm for Texture Synthesis". In: *Image Processing On Line* 7 (Jan. 2017), pp. 1–22. DOI: 10.5201/ipol.2017.171.