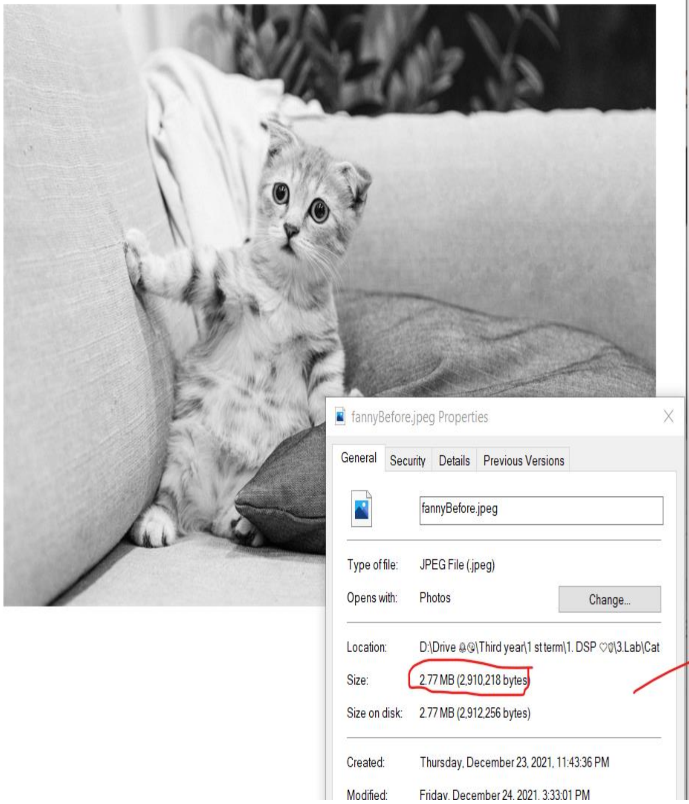
JPEG transform project

r=10



fannyBefore.jpeg Properties

fannyBefore.jpeg

Type of file: JPEG File (.jpeg)

Opens with: Photos

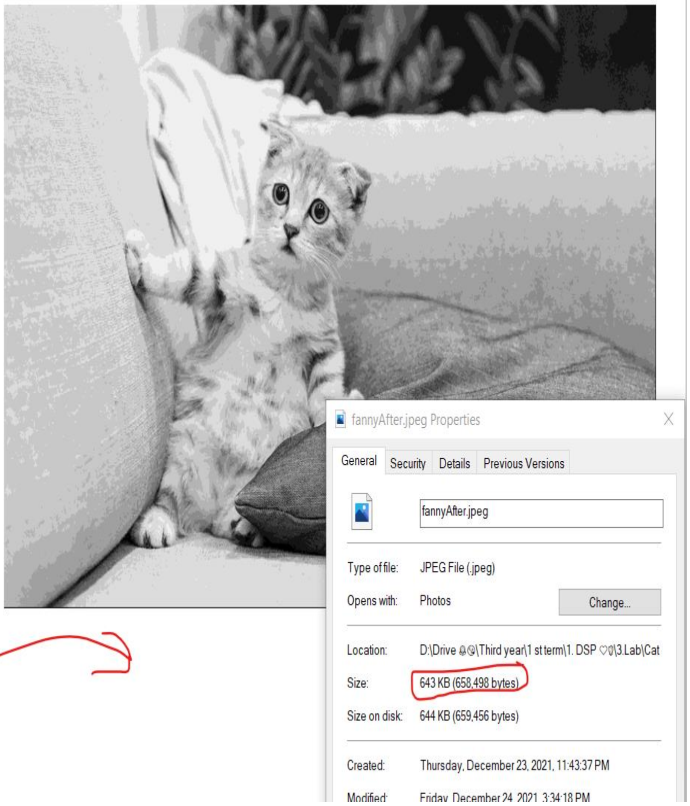
Location: D:\Drive 👻😊\Third year\1 st term\1. DSP ♡🔒\3.Lab\Cat

Size: 2.77 MB (2,910,218 bytes)

Size on disk: 2.77 MB (2,912,256 bytes)

Created: Thursday, December 23, 2021, 11:43:36 PM

Modified: Friday, December 24, 2021, 3:33:01 PM



fannyAfter.jpeg Properties

fannyAfter.jpeg

Type of file: JPEG File (.jpeg)

Opens with: Photos

Location: D:\Drive 👻😊\Third year\1 st term\1. DSP ♡🔒\3.Lab\Cat

Size: 643 KB (658,498 bytes)

Size on disk: 644 KB (659,456 bytes)

Created: Thursday, December 23, 2021, 11:43:37 PM

Modified: Friday, December 24, 2021, 3:34:18 PM

MAIN FILE

Variables

```
image = imread('colorCat.jpeg'); % Read the image

scaling=10;
```

1.Build C_8 matrix

```
C_8      = build_dct_mat();

inve    = inv(C_8);

transpo = C_8';

transpo-inve; %%Check if inverse == transpose (10^-15 ~=0)
```

2. JPEG encoding

```
gray_image = rgb2gray(image);  %% turn photo to gray scale

imwrite(gray_image,"cat_before_compression.jpeg" );  %% Save the gray image

gray_image = pading_image(gray_image);  %% this function check if

%% image size is multiple from 8 or not
```

2.1.Block divide

```
splitted_image = split_image(gray_image);
```

2.2. DCT block

```
blocksDCT = DCT_block(splitted_image,C_8,0);
```

2.3. Quantization

```
load  'DCTQ' % I download it because it is standard matrix

JPEG_result = Quantize_JPEG(blocksDCT,DCTQ,scaling);
```

3. JPEG decoding

3.1.Rescaling the data blocks

```
rescale_image = rescaling(JPEG_result,DCTQ,scaling);
```

3.2. DCT block inverse

```
blocksIDCT = DCT_block(rescale_image,C_8,1)
```

3.3. Merging the blocks

```
JPEG_image = recombine_blocks(blocksIDCT);
```

4. Save the compressed image

```
imwrite(JPEG_image, "cat_after_compression.jpeg");

imshow(image)

title('Color image before compression','FontSize',16,'color','red')


figure;
imshow(gray_image)

title('Gray image before compression','FontSize',16,'color','blue')


figure;
imshow(JPEG_image)

title('Compression image','FontSize',16,'color','green')
```

NOTE:

OUR (MAIN FILE) CONTAIN CALL OF FUNCTIONS THAT WILL BE DISCUSSED BELOW STEP BY STEP:

(1) find build_dct_mat()

where the matrix C_N has elements

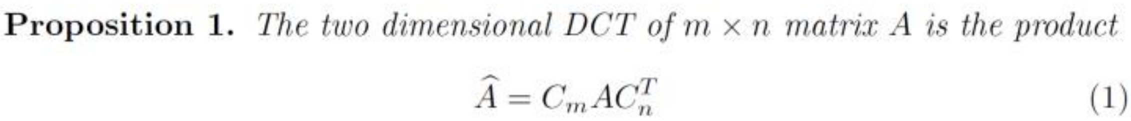
$$C_N(k,r) = u_k \cos\left(\frac{\pi}{N}k\left(r+\frac{1}{2}\right)\right) \qquad (2)$$

Construct (C8) where :

$u_0 = \sqrt{\frac{1}{N}}$ && $u_k = \sqrt{\frac{2}{N}}\ for\ k > 0.$

```
function C_8 = build_dct_mat()
r = (0:7);
K = (1:7)';
u_0 = sqrt(1/8);
C_0 = repelem(u_0,8);
C_7 = sqrt(2/8).*cos((pi/8)*(K*(r+.5)));
C_8 = [C_0;C_7]; % Concatenation
end
```

(2) DCT_block()



Proposition 1. *The two dimensional DCT of $m \times n$ matrix A is the product*

$$\widehat{A} = C_m A C_n^T \tag{1}$$

Here :

Parameter to choose if you want dct() or idct():

-If parameter =0 then block_DCT contain the dct matrix of each block.

-But if parameter ~=1 then block_DCT contain the inverse dct matrix of each block.

```
function block_DCT = DCT_block(splits,C_8,paramter)
[l ,m ,row ,column]=size(splits);
if paramter~=0
    C_8=C_8';
end
for i=1:row
    for j=1:column
        sub_Image=double(splits(:,:,i,j));
        block_DCT(:,:,i,j) =C_8*sub_Image*C_8';
    end
end
end
```

(3) Split_image()

Here we split image into blocks of Size (8*8):

Result is a 4-D matrix 8x8x$\frac{row}{8}$x$\frac{column}{8}$:

So result(:,:,i,j) this indicates to the ith &jth block, which size id 8x8:

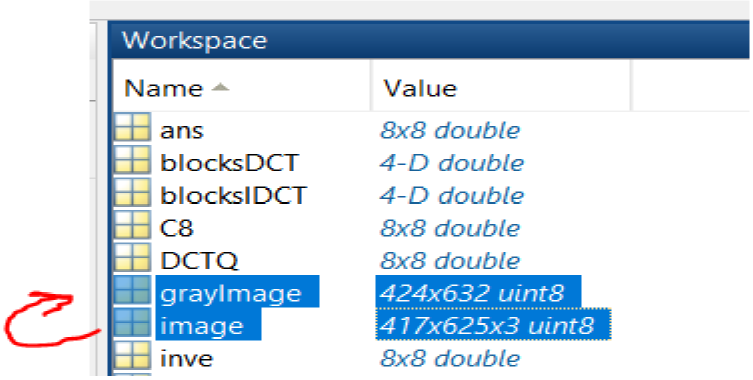
```
function result = split_image(gray_image)
block_size=8;
[row ,column]=size(gray_image);


for i =1:row/block_size
    for j=1:column/block_size
        result(:,:,i,j) = gray_image( (((i-1)*block_size)+1) :(i*block_size)
,(((j-1)*block_size)+1):(j*block_size) );
    end
end
end
```

(4) Pading_image()

If image size is not divisible by (8) then pad rows and columns by zeros until it's divisible.

```
function pad_gray = pading_image(gray_img)
[row ,colum] = size(gray_img);
pad_row    = 0;
pad_column = 0;
if(mod(row,8))  %% if row is not multiple from 8 then we want to calclate the
bading
    num=floor(row/8)+1;
    pad_row=num*8-row;
end
if(mod(colum,8)) %% if column is not multiple from 8 then we want to calclate the
bading
    num = floor(colum/8)+1;
    pad_column=num*8-colum;
end
if((mod(colum,8))&(mod(row,8)))
   pad_gray= padarray(gray_img,[pad_row pad_column],0,'post');
else
   pad_gray=gray_img;
end
end
```

And here how it works if image is not divisible by 8



Workspace

Name	Value
ans	8x8 double
blocksDCT	4-D double
blocksIDCT	4-D double
C8	8x8 double
DCTQ	8x8 double
grayImage	424x632 uint8
image	417x625x3 uint8
inve	8x8 double

so here our image size before padding =417x625.

417 &625 don't accept divide by 8 so we pad it until the first big number which accept divide by 8,

So, our image size after padding =424x632

424/8= 53 &632/8=79.

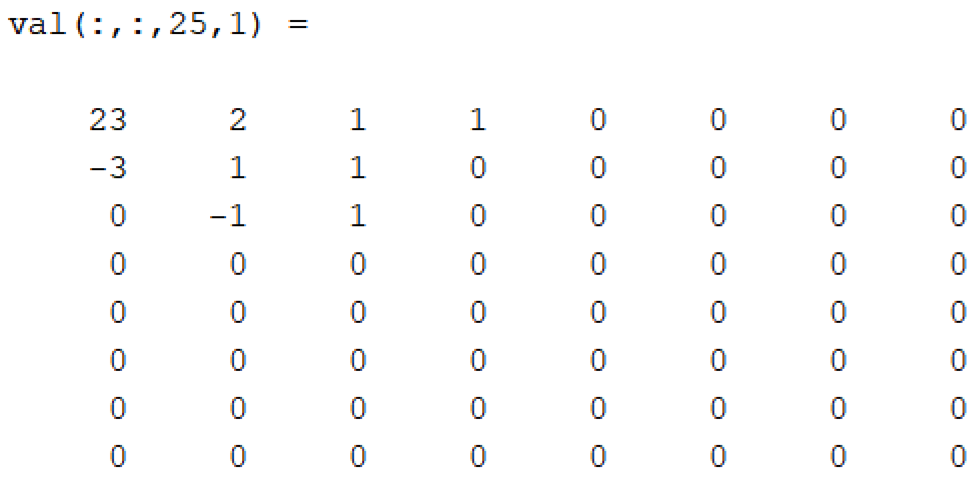
(5) Quantization JPEG

Here we will multiply DCTQ (standard matrix for jpeg) by r (T=scale*DCTQ):

Then we get round () by divide element by element our sub_block dct matrices (8x8) by Quantization matrix (T 8x8) to block high frequency and get real data that have been compressed in low frequencies:

```
function JPEG_result = Quantize_JPEG(splitDCT,DCTQ,scaling)
T = scaling*DCTQ;
[l ,m ,row ,column]=size(splitDCT);
for i=1:row
    for j=1:column
        sub_img = double(splitDCT(:,:,i,j));
        JPEG_result(:,:,i,j) = round(sub_img./T);
    end
end
end
```

- This is for example the 25th,1st block (8x8) after multiplying by factor and perform quantization :



```
val(:,:,25,1) =

    23     2     1     1     0     0     0     0
    -3     1     1     0     0     0     0     0
     0    -1     1     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

(6) Rescaling()

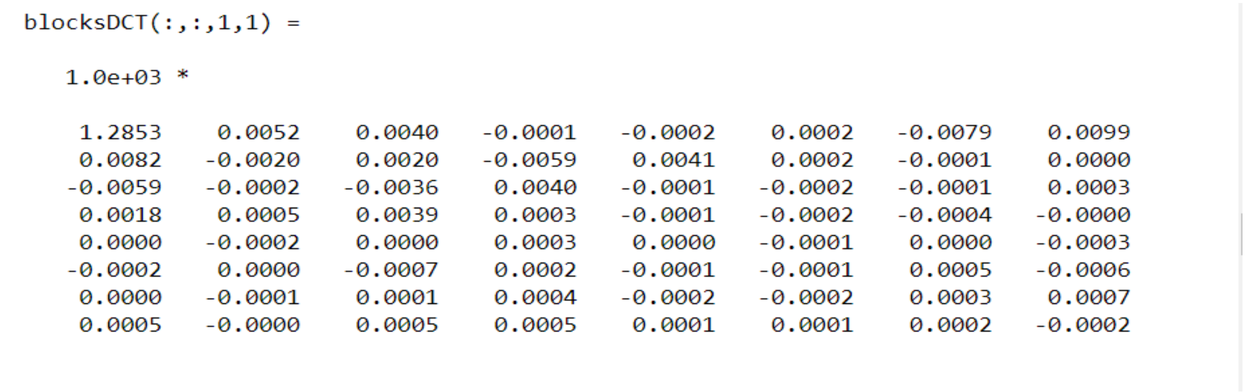
In this function we multiply quantized blocks by Samling factor T to make it ready for decoding.

So, any value below zero will be zero because of round (), the result will be like the result of block_DCT, with ignoring the values which contain low information:

```
function rescale_img =  rescaling(quantized_block,scaling,DCTQ)
T=scaling*DCTQ;
[l ,m ,row ,column]=size(quantized_block);
for i=1:row
    for j=1:column
        sub_img=double(quantized_block(:,:,i,j));
        rescale_img(:,:,i,j) =sub_img.*T;
    end
end
end
```

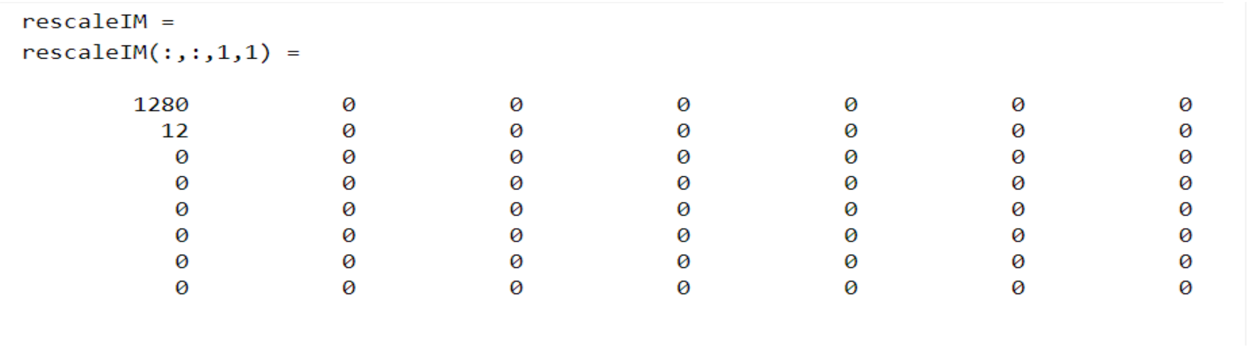
For example:

This is the dct output of 1st,1st block: (it's multiplied by 10^3)



```
blocksDCT(:,:,1,1) =

   1.0e+03 *

    1.2853    0.0052    0.0040   -0.0001   -0.0002    0.0002   -0.0079    0.0099
    0.0082   -0.0020    0.0020   -0.0059    0.0041    0.0002   -0.0001    0.0000
   -0.0059   -0.0002   -0.0036    0.0040   -0.0001   -0.0002   -0.0001    0.0003
    0.0018    0.0005    0.0039    0.0003   -0.0001   -0.0002   -0.0004   -0.0000
    0.0000   -0.0002    0.0000    0.0003    0.0000   -0.0001    0.0000   -0.0003
   -0.0002    0.0000   -0.0007    0.0002   -0.0001   -0.0001    0.0005   -0.0006
    0.0000   -0.0001    0.0001    0.0004   -0.0002   -0.0002    0.0003    0.0007
    0.0005   -0.0000    0.0005    0.0005    0.0001    0.0001    0.0002   -0.0002
```

After rescaling the 1^{st} ,1^{st} block will be:



```
rescaleIM =
rescaleIM(:,:,1,1) =

        1280           0           0           0           0           0           0
          12           0           0           0           0           0           0
           0           0           0           0           0           0           0
           0           0           0           0           0           0           0
           0           0           0           0           0           0           0
           0           0           0           0           0           0           0
           0           0           0           0           0           0           0
           0           0           0           0           0           0           0
```

(7) Recombines Blocks

after resampling and getting (IDCT) we merge sub-blocks again to recombine our image again.

Note: we get IDCT by using the same function of DCT but now parameter =1:

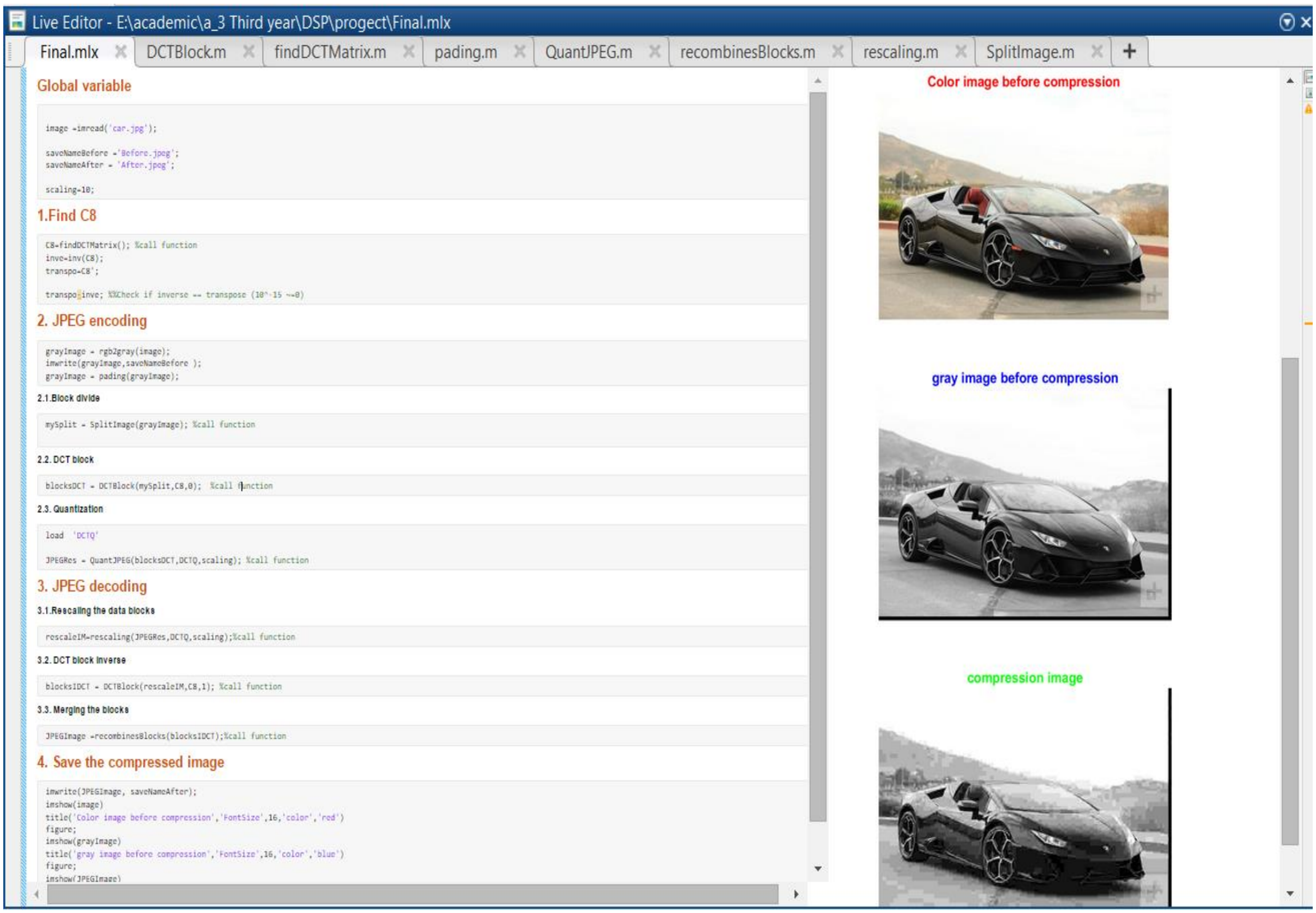
The IDCT using DCT_block:

```
function block_DCT = DCT_block(splits,C_8,paramter)
[l ,m ,row ,column]=size(splits);
if paramter~=0
    C_8=C_8';
end
for i=1:row
    for j=1:column
        sub_Image=double(splits(:,:,i,j));
        block_DCT(:,:,i,j) =C_8*sub_Image*C_8';
    end
end
end
```

now we will merge blocks using the inverse of the split function:

```
function JPEG_image =  recombine_blocks(resIDCT)
block_size=8;
[l ,m ,row ,column]=size(resIDCT);
for i =1:row
    for j=1:column
        JPEG_image( (((i-1)*block_size)+1) :(i*block_size) ,(((j-1)*block_size)+1):(j*block_size) )=resIDCT(:,:,i,j);
    end
end
JPEG_image=uint8(JPEG_image);
end
```

Change Scaling factor:


Live Editor - E:\academic\a_3 Third year\DSP\progect\Final.mlx
Final.mlx
DCTBlock.m
findDCTMatrix.m
pading.m
QuantJPEG.m
recombinesBlocks.m
rescaling.m
SplitImage.m
Color image before compression
gray image before compression
compression image

Global variable

```
image =imread('car.jpg');

saveNameBefore ='Before.jpeg';
saveNameAfter = 'After.jpeg';

scaling=10;
```

1.Find C8

```
C8=findDCTMatrix(); %call function
inve=inv(C8);
transpo=C8';

transpo-inve; %%Check if inverse == transpose (10^-15 ~=0)
```

2. JPEG encoding

```
grayImage = rgb2gray(image);
imwrite(grayImage,saveNameBefore );
grayImage = pading(grayImage);
```

2.1.Block divide

```
mySplit = SplitImage(grayImage); %call function
```

2.2. DCT block

```
blocksDCT = DCTBlock(mySplit,C8,0);  %call function
```

2.3. Quantization

```
load  'DCTQ'

JPEGRes = QuantJPEG(blocksDCT,DCTQ,scaling); %call function
```

3. JPEG decoding

3.1.Rescaling the data blocks

```
rescaleIM=rescaling(JPEGRes,DCTQ,scaling);%call function
```

3.2. DCT block inverse

```
blocksIDCT = DCTBlock(rescaleIM,C8,1); %call function
```

3.3. Merging the blocks

```
JPEGImage =recombinesBlocks(blocksIDCT);%call function
```

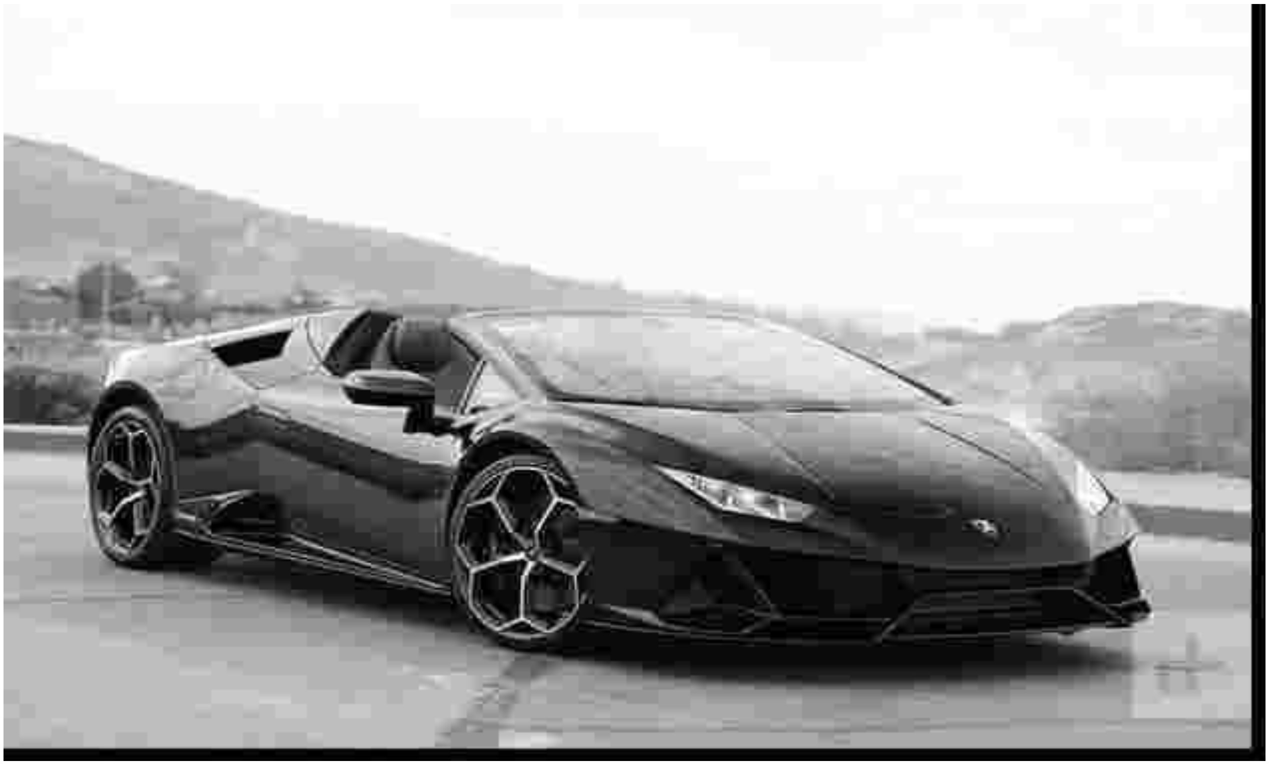
4. Save the compressed image

```
imwrite(JPEGImage, saveNameAfter);
imshow(image)
title('Color image before compression','FontSize',16,'color','red')
figure;
imshow(grayImage)
title('gray image before compression','FontSize',16,'color','blue')
figure;
imshow(JPEGImage)
```

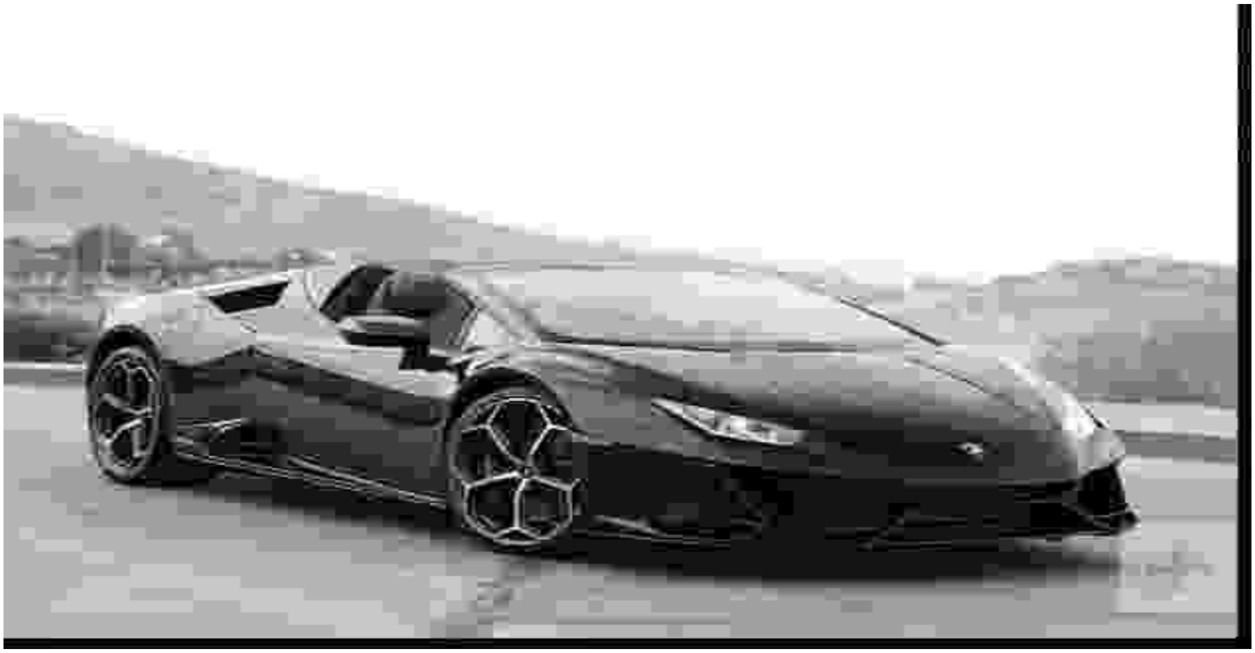
IMAGE BEFORE

SCALING FACTOR =3

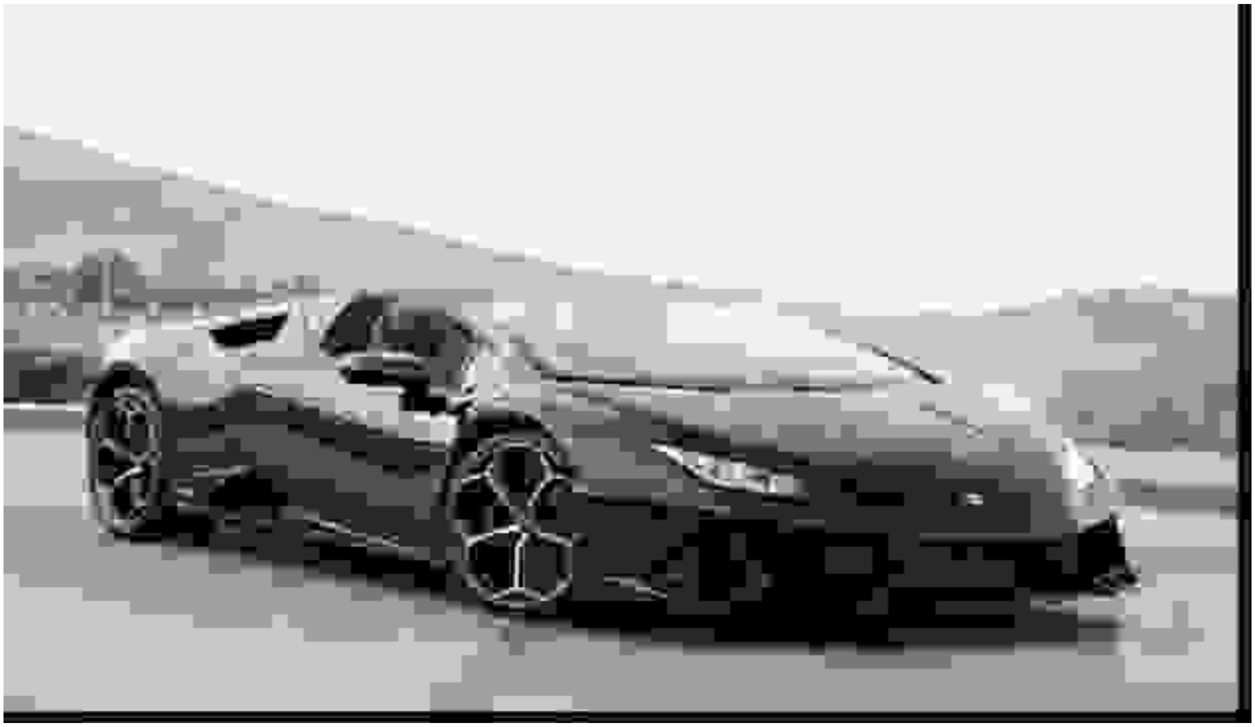
SCALING FACTOR =5



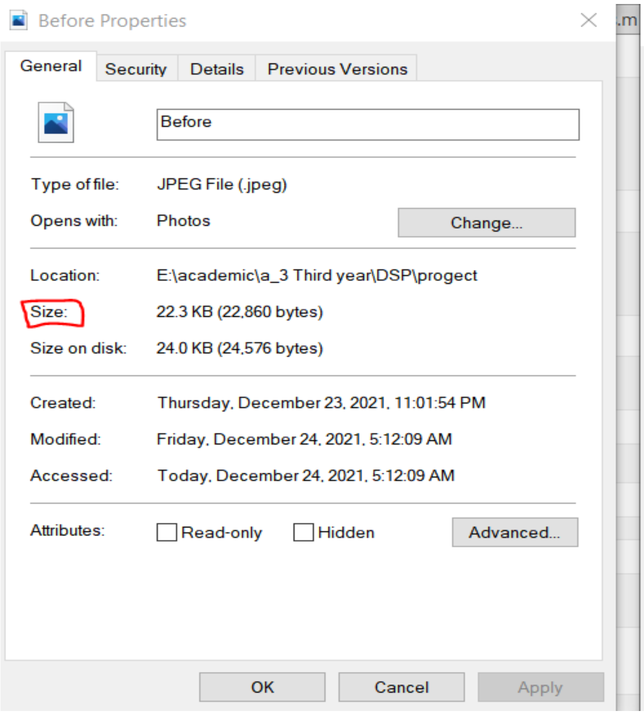
SCALING FACTOR =10



SCALING FACTOR =20



AND HERE WE NOTICE THAT SIZE AFTER COMPRESSION IS LESS THAN BEFORE:



Before Properties

General | Security | Details | Previous Versions

Before

Type of file: JPEG File (.jpeg)

Opens with: Photos

Location: E:\academic\a_3 Third year\DSP\progect

Size: 22.3 KB (22,860 bytes)

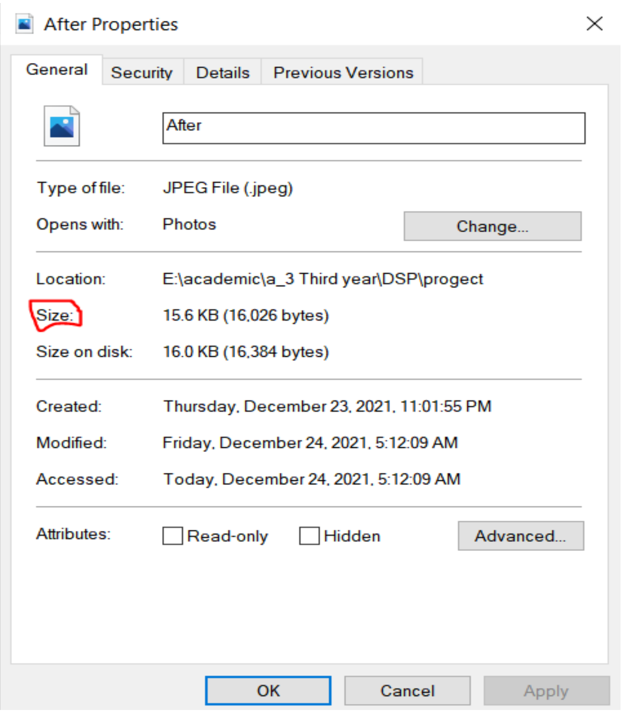
Size on disk: 24.0 KB (24,576 bytes)

Created: Thursday, December 23, 2021, 11:01:54 PM

Modified: Friday, December 24, 2021, 5:12:09 AM

Accessed: Today, December 24, 2021, 5:12:09 AM

Attributes: □ Read-only □ Hidden



After Properties

General | Security | Details | Previous Versions

After

Type of file: JPEG File (.jpeg)

Opens with: Photos

Location: E:\academic\a_3 Third year\DSP\progect

Size: 15.6 KB (16,026 bytes)

Size on disk: 16.0 KB (16,384 bytes)

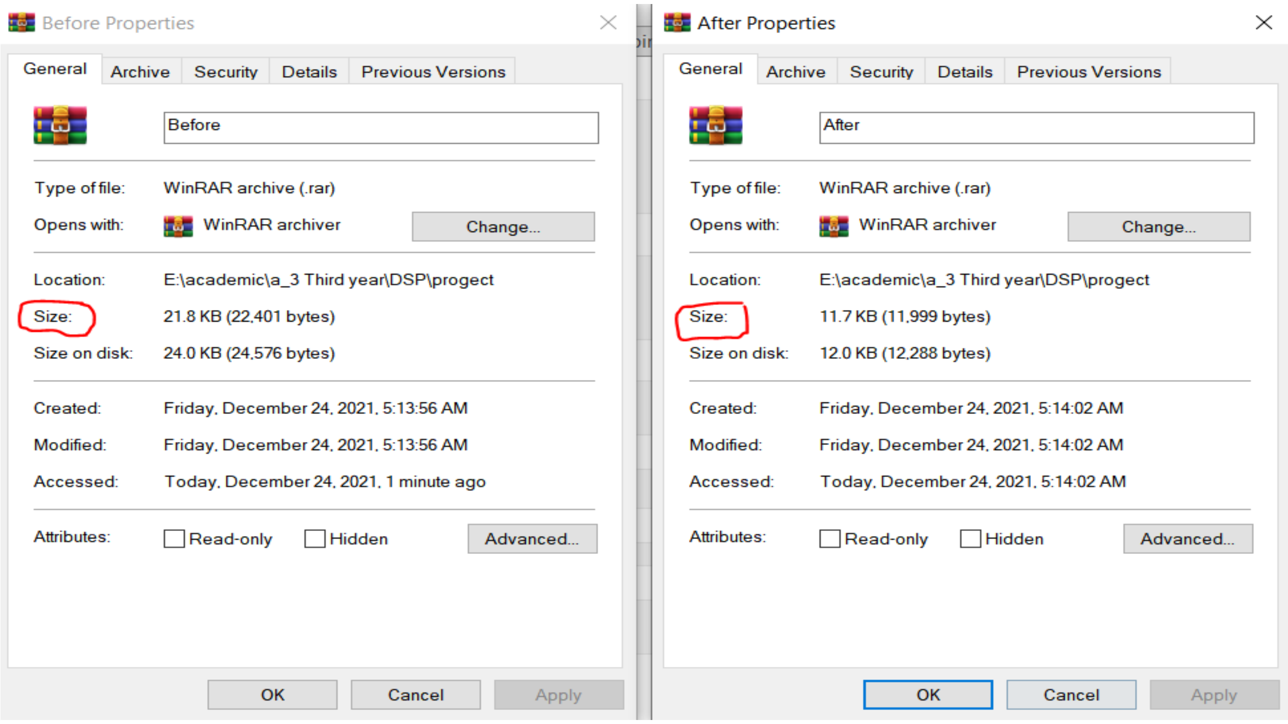
Created: Thursday, December 23, 2021, 11:01:55 PM

Modified: Friday, December 24, 2021, 5:12:09 AM

Accessed: Today, December 24, 2021, 5:12:09 AM

Attributes: □ Read-only □ Hidden

And this if we get (.rar file)



Before Properties

Before

Type of file: WinRAR archive (.rar)

Opens with: WinRAR archiver

Location: E:\academic\a_3 Third year\DSP\progect

Size: 21.8 KB (22,401 bytes)

Size on disk: 24.0 KB (24,576 bytes)

Created: Friday, December 24, 2021, 5:13:56 AM

Modified: Friday, December 24, 2021, 5:13:56 AM

Accessed: Today, December 24, 2021, 1 minute ago

Attributes: □ Read-only □ Hidden

After Properties

After

Type of file: WinRAR archive (.rar)

Opens with: WinRAR archiver

Location: E:\academic\a_3 Third year\DSP\progect

Size: 11.7 KB (11,999 bytes)

Size on disk: 12.0 KB (12,288 bytes)

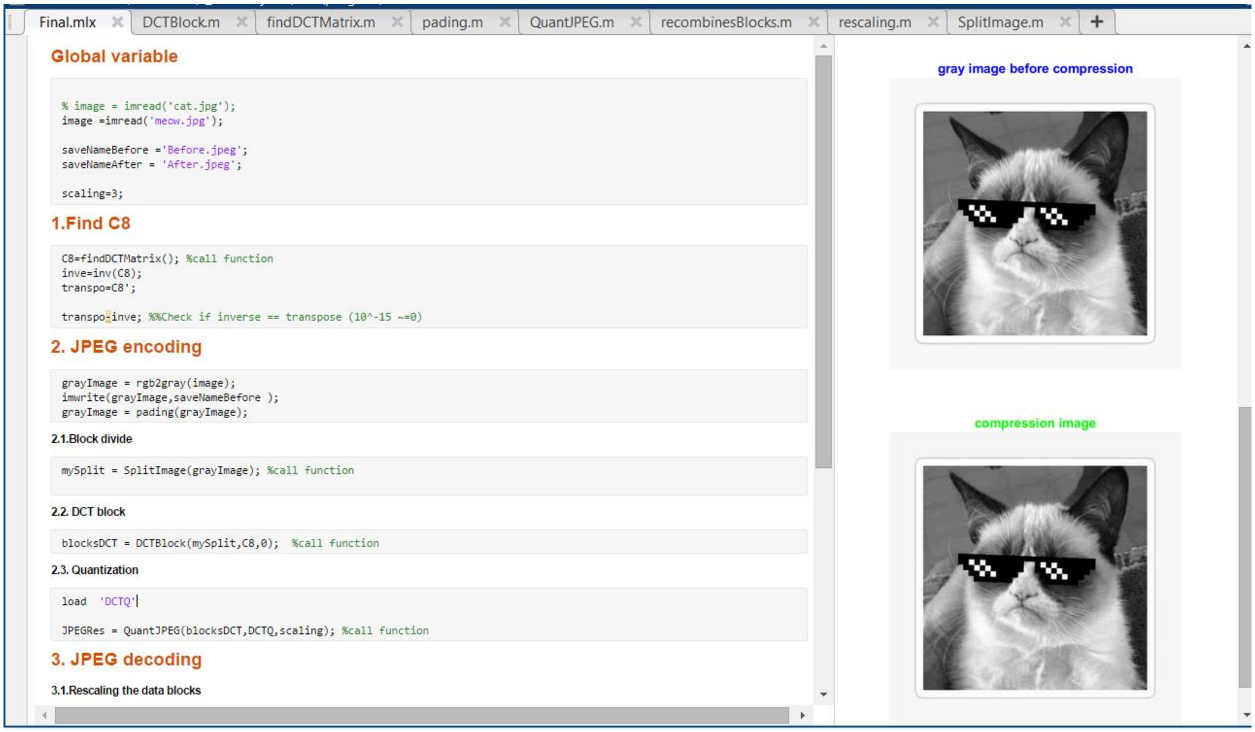
Created: Friday, December 24, 2021, 5:14:02 AM

Modified: Friday, December 24, 2021, 5:14:02 AM

Accessed: Today, December 24, 2021, 5:14:02 AM

Attributes: □ Read-only □ Hidden

Change Scaling factor:



Global variable

```
% image = imread('cat.jpg');
image =imread('meow.jpg');

saveNameBefore ='Before.jpeg';
saveNameAfter = 'After.jpeg';

scaling=3;
```

1.Find C8

```
C8=findDCTMatrix(); %call function
inve=inv(C8);
transpo=C8';

transpo-inve; %%Check if inverse == transpose (10^-15 ~=0)
```

2. JPEG encoding

```
grayImage = rgb2gray(image);
imwrite(grayImage,saveNameBefore );
grayImage = pading(grayImage);
```

2.1.Block divide

```
mySplit = SplitImage(grayImage); %call function
```

2.2. DCT block

```
blocksDCT = DCTBlock(mySplit,C8,0);  %call function
```

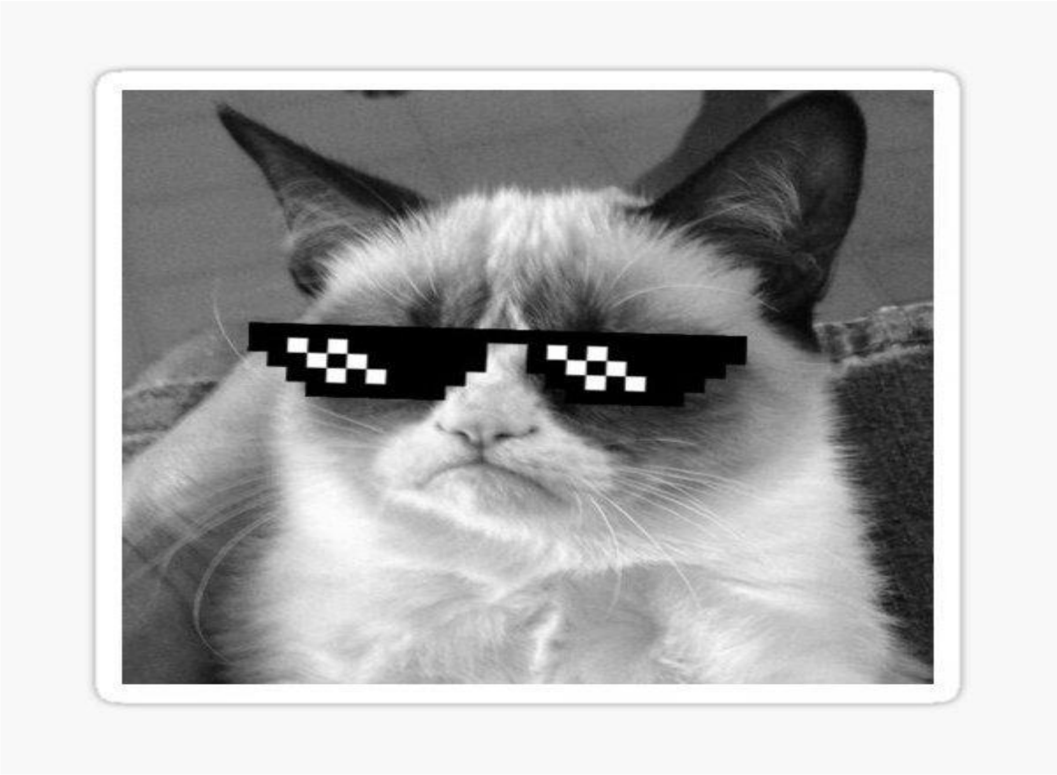
2.3. Quantization

```
load  'DCTQ'

JPEGRes = QuantJPEG(blocksDCT,DCTQ,scaling); %call function
```

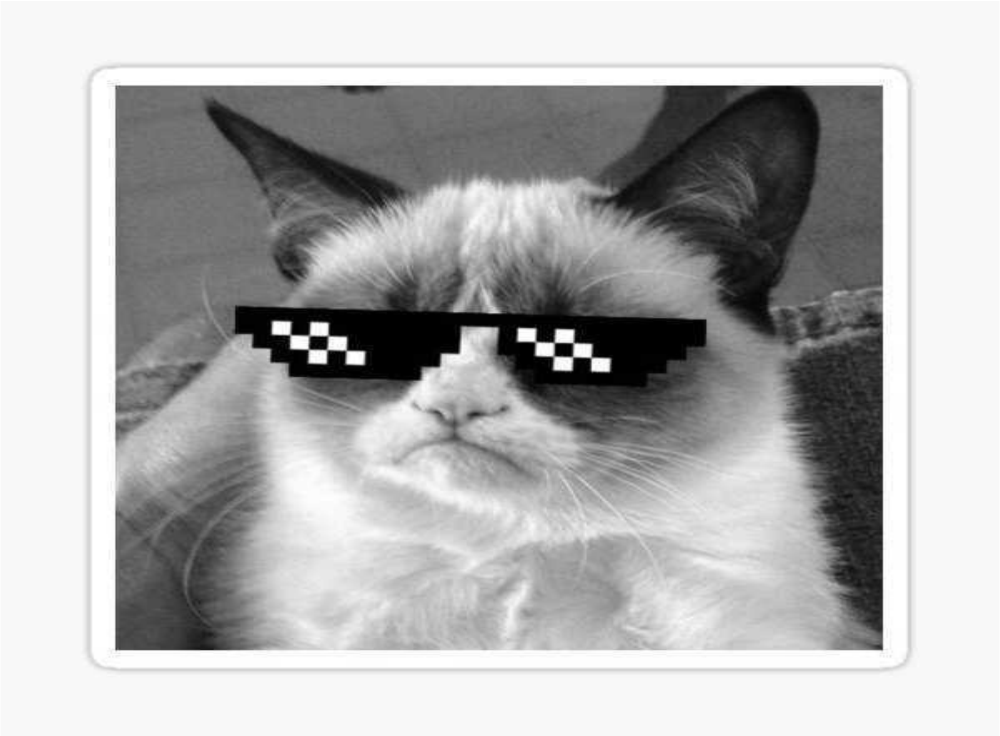
3. JPEG decoding

3.1.Rescaling the data blocks

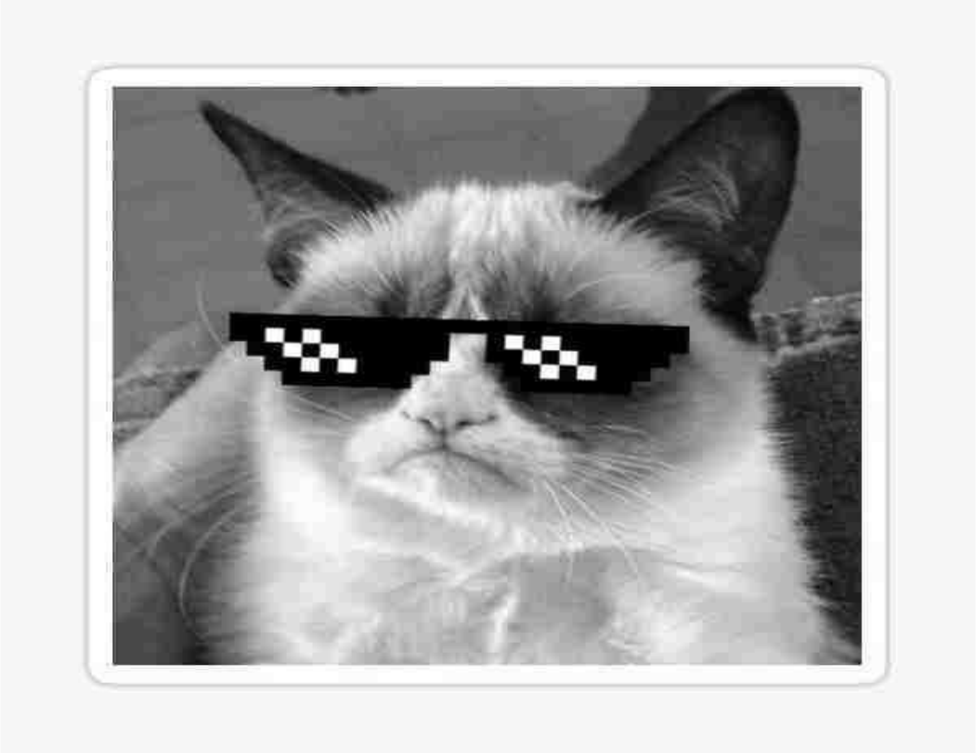
IMAGE BEFORE



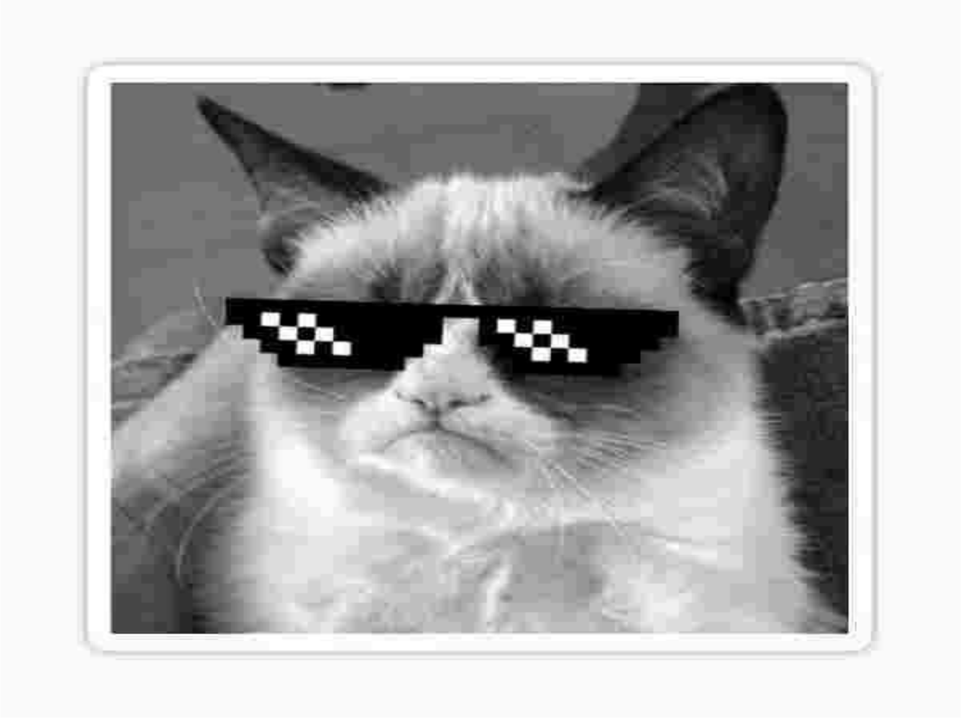
SCALING FACTOR =1



SCALING FACTOR =3



SCALING FACTOR =5



SCALING FACTOR =10