Project 1d Option 2.3: User Input \(^_^)/

Proof Of Concept

For the "User Input" option there is not any change to the underlying compression so the proof of concept is done by comparing the different preset options (high, medium, and low).

> |Quality | Size (bytes)| Compression ratio | Mean squared error|

|---------|----------------------|-------|-------| | High | 85112 | 0.07 | 27.09 | | Medium| 46768 | 0.04 | 31.48 | | Low | 28510 | 0.02 | 38.87 |

The different levels of quality give the expected results. If the user lowers the quality the file and compression ratio decrease but the MSE increases because of the worse picture quality. The numbers were gathered from using the same girl.png image from the previous parts.

Part 1 Copy & Paste (._.)

I copied code from proj1c. Most of it was imported except the portions I need to change.

In [1]:
```
import imageio
import numpy as np
import scipy.fft
import matplotlib.pyplot as plt
import pickle
from compressor import huffman_compress, huffman_decompress
from project1c_code import quantize, dequantize, dctim, idctim, lopass_spectrum, infla
```

In [2]:
```
def decompress_image(compressed):
    """Decompresses a bytes-like object produced by `compress_image()`.
    Input: a bytes object.
    Output: a 3-D np.ndarray object whose third axis has size 3.
    """

    # Write your function here!

    comp, cutoff, levels, shape, limits = pickle.loads(compressed)

    decomp = huffman_decompress(comp)
    dct = dequantize(np.array(decomp), levels, limits)
    spec = inflate_spectrum(dct, cutoff[0], cutoff[1], shape)

    return idctim(spec)

def imshow_side_by_side(image1, image2, figsize=(18, 5)):
    fig, axs = plt.subplots(1, 2, figsize=figsize)
```

```
    axs[0].imshow(image1)
    axs[1].imshow(image2)
```

Part 2 Change compress_image() to take more parameters (-_-)zzz

The ability to change the cut off frequecies, the levels, and the maximum file size were added.

In [9]:
```
def compress_image(image, cutoff_x, cutoff_y, levels, max_size):
    """Compresses an image `image` into a bytes-like type.
    Input: a 3-D np.ndarray object whose third axis has size 3 (a color image).
    Output: a bytes object containing a lossily compressed version of the input.
    """

    # -1 for levels and cuttoff represents the default inputs
    if levels == -1:
        levels = 2**12
    elif levels == -2:
        levels = 2**11
    elif levels == -3:
        levels = 2**10
    else:
        # otherwise keep user input
        pass

    # image info is left the same
    image_DCT = dctim(image)
    shape = image_DCT.shape


    if cutoff_x == -1: # high
        cutoff_x = int(shape[0]//2.8)
    elif cutoff_x == -2: # medium
        cutoff_x = int(shape[0]//3.6)
    elif cutoff_x == -3: # low
        cutoff_x = int(shape[0]//4.2)
    else:
        # otherwise keep user input
        pass

    if cutoff_y == -1: # high
        cutoff_y = int(shape[1]//2.8)
    elif cutoff_y == -2: # medium
        cutoff_y = int(shape[1]//3.6)
    elif cutoff_y == -3: # low
        cutoff_y = int(shape[1]//4.2)
    else:
        # otherwise keep user input
        pass

    image_filtered = lopass_spectrum(image_DCT, cutoff_x, cutoff_y)
    image_quant = quantize(image_filtered, levels)
    reshaped_quant = np.reshape(image_quant[0], (-1,))

    comp = huffman_compress(list(reshaped_quant))
```

```
    compressed = pickle.dumps((comp, (cutoff_x, cutoff_y), levels, shape, image_quant[

    # If the maximum size desired is exceeded compression repeats with stronger compre
    if max_size != -1 and len(compressed) > max_size:
        if levels < 2**3:
            print("Maximum File Size Requirement could not be met")
            print(f"Returning file of size: {len(compressed)}bytes")
            return compressed
        return compress_image(image, int(cutoff_x//1.1), int(cutoff_y//1.1), int(level
    return compressed
```

Part 3 Main Function (°_°)

This is the main function that handles the user input for the compressor

In [25]:
```
# function that can decompress an image file
def decomp_file(filepath):
    if filepath == None:
        filepath = input("Enter filepath: ")
    file = open(filepath, 'rb')
    compressed = file.read()
    image = decompress_image(compressed)
    print("Image Decompressed")
    return image


# Options the user can choose for image quality
def menu_1():
    print("Choose Option")
    print("-----------------------")
    print("1. High Quality")
    print("2. Medium Quality")
    print("3. Low Quality")
    print("4. Maximum File Size")
    print("5. Advanced Options")
    print("6. Quit")
    print("-----------------------")
    return input("Enter here: ")

# Options for the user to decide what to do
# with the compressed image
def menu_2():
    print("\nImage Compressed")
    print("What do you want to do with it?")
    print("--------------------------------")
    print("1. Display")
    print("2. Save")
    print("3. Restart Program")
    print("4. Quit")
    print("-----------------------")
    return input("Enter here: ")

# The function that handles all the user input and
# calls compress_image() with the correct parameters.
#
```

```python
# The function can display or save the compressed images.
def image_compress(filepath=None):
    if filepath == None:
        filepath = input("Enter the file path of image: ")
        image = imageio.imread(filepath)
    else:
        image = imageio.imread(filepath)

    option = int(menu_1())

    if option == 1:
        compressed = compress_image(image, -1, -1, -1, -1)
        post_option = int(menu_2())
    elif option == 2:
        compressed = compress_image(image, -2, -2, -2, -1)
        post_option = int(menu_2())
    elif option == 3:
        compressed = compress_image(image, -3, -3, -3, -1)
        post_option = int(menu_2())
    elif option == 4:
        compressed = compress_image(image, -1, -1, -1, -1)
        print(f"Current Compression Size: {len(compressed)}. Enter -1 to keep")
        size = int(input("Enter Maximum File Size: "))
        compressed = compress_image(image, -1, -1, -1, size)
        post_option = int(menu_2())
    elif option == 5:
        print("You will now choose the cut off frequencies and quant levels")
        print("Entering -1, -2, -3 will use the high, medium, and low quality")
        print("presets for any value")
        print("-------------------------------------------------------------")
        cutoff_x = int(input("Enter cut-off frequency for the x-axis: "))
        cutoff_y = int(input("Enter cut-off frequency for the y-axis: "))
        levels = int(input("Enter the number of quantization levels: "))
        size = int(input("Enter the maximum file size: "))

        compressed = compress_image(image, cutoff_x, cutoff_y, levels, size)
        post_option = int(menu_2())
    elif option == 6:
        post_option = 4
    else:
        print("invalid input restarting program")
        main()

    if post_option == 1:
        recovered = decompress_image(compressed)
        print(f"Size: {len(compressed)}")
        print("Compression ratio: {}".format(len(compressed)/image.size))
        print("MSE: {}".format(((recovered - image) ** 2).mean()))
        imshow_side_by_side(image, recovered)
    elif post_option == 2:
        file = open(input('Enter filepath: '), 'wb')
        file.write(compressed)
        file.close()
    elif post_option == 3:
        main()
    elif post_option == 4:
        pass
```

```
    else:
        print("invalid input restarting program")
        main()

    if post_option != 3:
        print("Program Finished. Shutting Down")

# The main function that can either compress
# or decompress an image file
# if it decompresses the decompressed image is returned
def main(filepath=None):
    print("Image Compressor/Decompressor")
    print("-----------------------------")
    print("1. Compress Image")
    print("2. Decompress Compressed Image File")
    val = int(input("Enter Choice: "))

    if (val == 1):
        image_compress(filepath)
    else:
        return decomp_file(filepath)
```

Part 4 Results (^_-)-☆

Here I am going to briefly go through the choices the user has and the compression results from the different levels of quality. I will be using the img/girl.png file for the examples.

Basic Rundown

The main function gives the user two options compress and decompress. The compress option gives the user the choice of how much they want to compress the image then they get the choice to display it to view it or to save it to a file location.

The decompress option is not like the one in project 1c. This decompress is a bonus feature I added so that the files saved to disk could be decompressed without the user having to do this themselves. If the decompress option is chosen main returns the image.

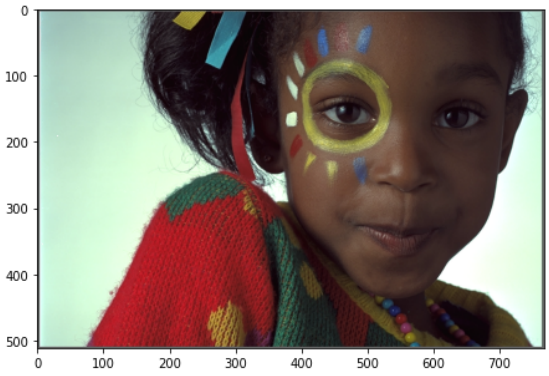
I also added plenty of quality of life things that made sense to add that are not relevent to the project but make it more usable.

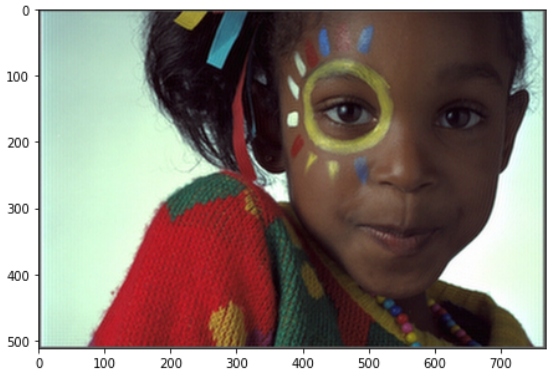
High, Medium, and Low

The three presets are high, medium, and low quality (higher quality -> better picture).

In [5]:
```
# High Compression example
main('img/girl.png')
```

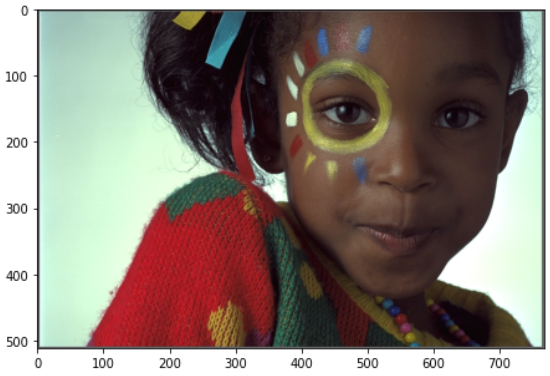
```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 1

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 85112
Compression ratio: 0.07215033637152778
MSE: 27.087228563096787
Program Finished. Shutting Down
```

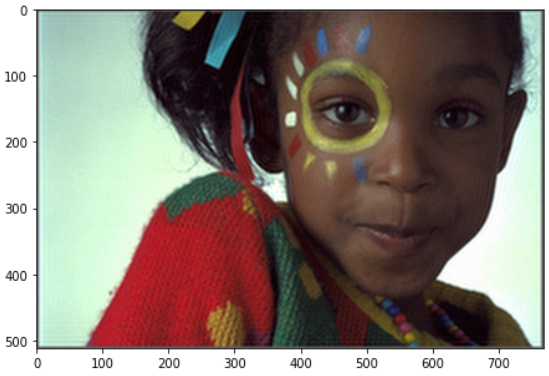




In [6]:
```
# Medium Quality Compression example
main('img/girl.png')
```

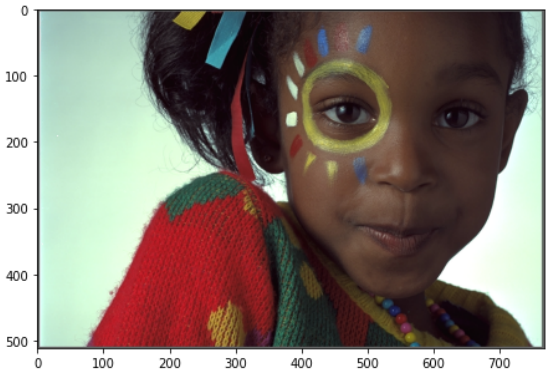
```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 2

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 46768
Compression ratio: 0.03964572482638889
MSE: 31.478783501519096
Program Finished. Shutting Down
```

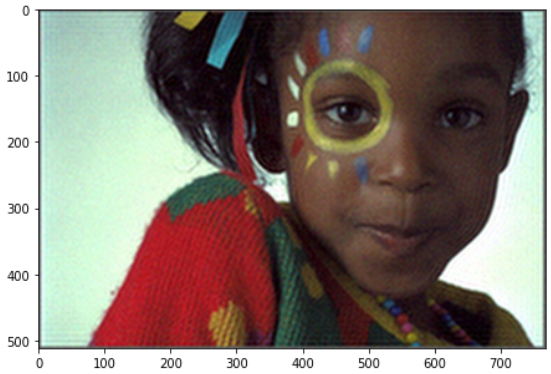




In [10]:
```
# Low Quality Compression example
main('img/girl.png')
```

```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 3

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 28510
Compression ratio: 0.024168226453993056
MSE: 38.86918046739366
Program Finished. Shutting Down
```



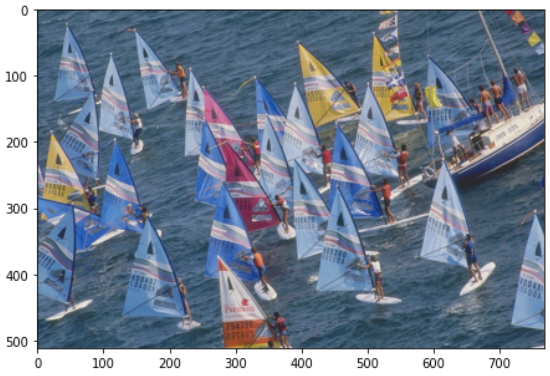


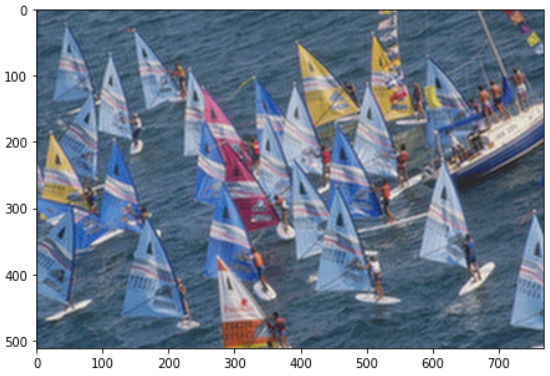
Maximum File Size

If the user chooses this option they enter a maximum file size and the image keeps being compressed until it is below that file size. (The image is not compressed again and again. The parameters are changed recursivly and the orignal image is compressed with those new paramters. Here is a little demo.

In [14]:
```
# Maximum File Size Example 1
# Small difference
main('img/sails.png')
```

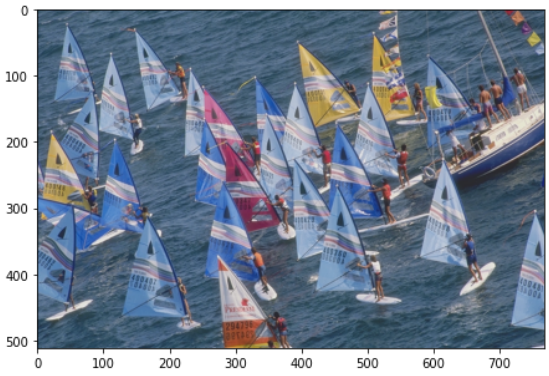
```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 4
Current Compression Size: 124175. Enter -1 to keep
Enter Maximum File Size: 60000

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 52014
Compression ratio: 0.044092814127604164
MSE: 65.02530246310764
Program Finished. Shutting Down
```

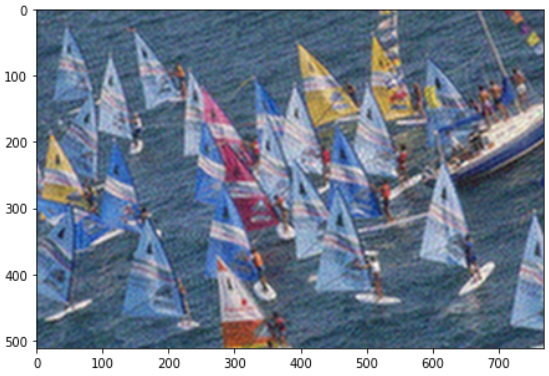




In [15]:
```
# Maximum File Size Example 2
# Large difference
main('img/sails.png')
```

```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 4
Current Compression Size: 124175. Enter -1 to keep
Enter Maximum File Size: 30000

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 20944
Compression ratio: 0.017754448784722224
MSE: 75.5110821194119
Program Finished. Shutting Down
```



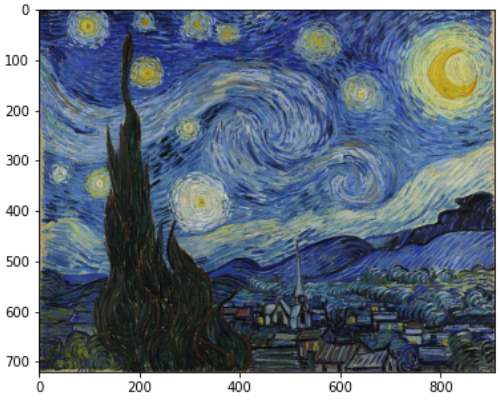


Custom Compression

The user can choose the advanced option to change every compression parameter.

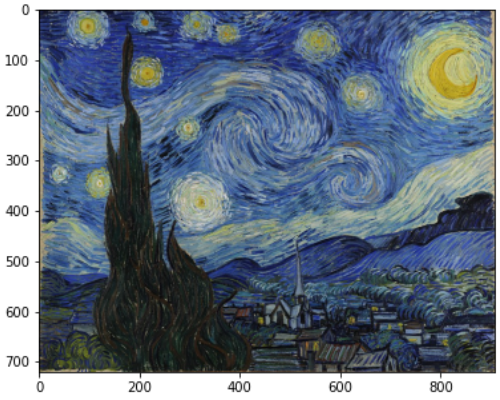
In [17]:
```
# Custom Compression Example #1
main('img/starrynight.png')
```

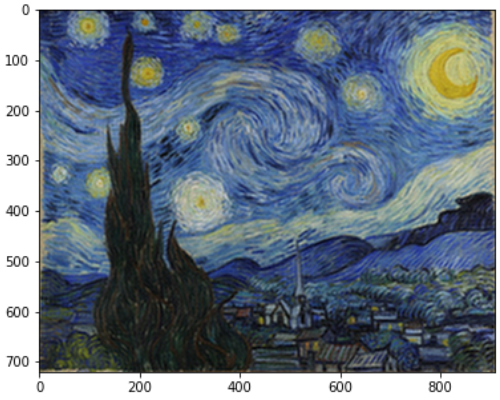
```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 5
You will now choose the cut off frequencies and quant levels
Entering -1, -2, -3 will use the high, medium, and low quality
presets for any value
------------------------------------------------------------
Enter cut-off frequency for the x-axis: 160
Enter cut-off frequency for the y-axis: 160
Enter the number of quantization levels: 1400
Enter the maximum file size: -1

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 38477
Compression ratio: 0.01959672819133765
MSE: 86.81533736707004
Program Finished. Shutting Down
```



In [18]:
```
# Custom Compression Example #2
# The user can also keep some presets
main('img/starrynight.png')
```

```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 5
You will now choose the cut off frequencies and quant levels
Entering -1, -2, -3 will use the high, medium, and low quality
presets for any value
-------------------------------------------------------------
Enter cut-off frequency for the x-axis: -3
Enter cut-off frequency for the y-axis: -3
Enter the number of quantization levels: -1
Enter the maximum file size: -1

Image Compressed
What do you want to do with it?
--------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 1
Size: 76679
Compression ratio: 0.03905339608034877
MSE: 84.7391399788127
Program Finished. Shutting Down
```





Saving Feature

The user can choose to save the compressed image to disk like one would do with a compressed file. The main function can also decompress it later on.

In [20]:
```
# Saving Feature Demo
main('img/girl.png')
```

```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 1
Choose Option
----------------------
1. High Quality
2. Medium Quality
3. Low Quality
4. Maximum File Size
5. Advanced Options
6. Quit
----------------------
Enter here: 3

Image Compressed
What do you want to do with it?
-------------------------------
1. Display
2. Save
3. Restart Program
4. Quit
----------------------
Enter here: 2
Enter filepath: img/pic
Program Finished. Shutting Down
```

> The file is now saved in the projects img folder with the relative file path 'img/pic'. This file is on disk and the user can inspect its size and it will be the same as the low quality size on the proof of concept table.

In [21]:
```
import os

file_size = os.path.getsize('img/pic')
print(f"This is the size of 'img/pic': {file_size}")
```

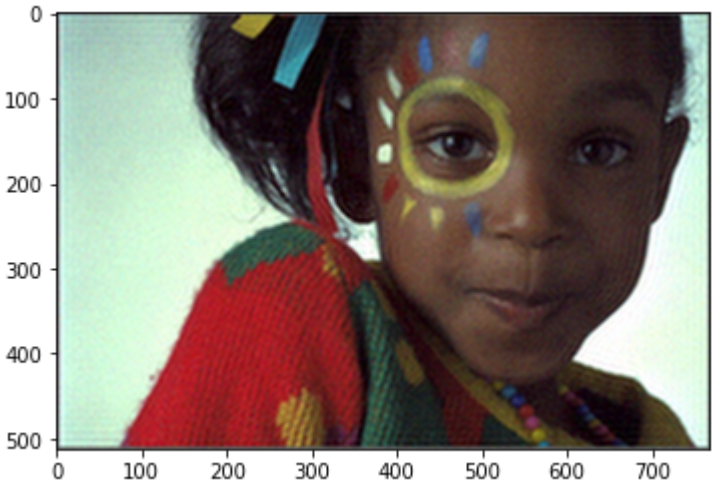
```
This is the size of 'img/pic': 28510
```

> The user can decompress the image by using main's decompress feature which will return the image

In [26]:
```
# Decompressing the image and returning it to image
image = main('img/pic')
```

```
Image Compressor/Decompressor
-----------------------------
1. Compress Image
2. Decompress Compressed Image File
Enter Choice: 2
Image Decompressed
```

In [27]:
```
# BOOM here is the recovered image
plt.imshow(image)
plt.show()
```



> That sums up the relevant features of the program !!