CS156 - LBA - PCA

Tower Bridge, London, UK

In [1]:

```
import numpy as np #importing libraries
import matplotlib.pyplot as plt #for plotting
from glob import glob #for loading files
from PIL import Image #for opening image
from sklearn.decomposition import PCA #for principal component analysis
```

In [2]:

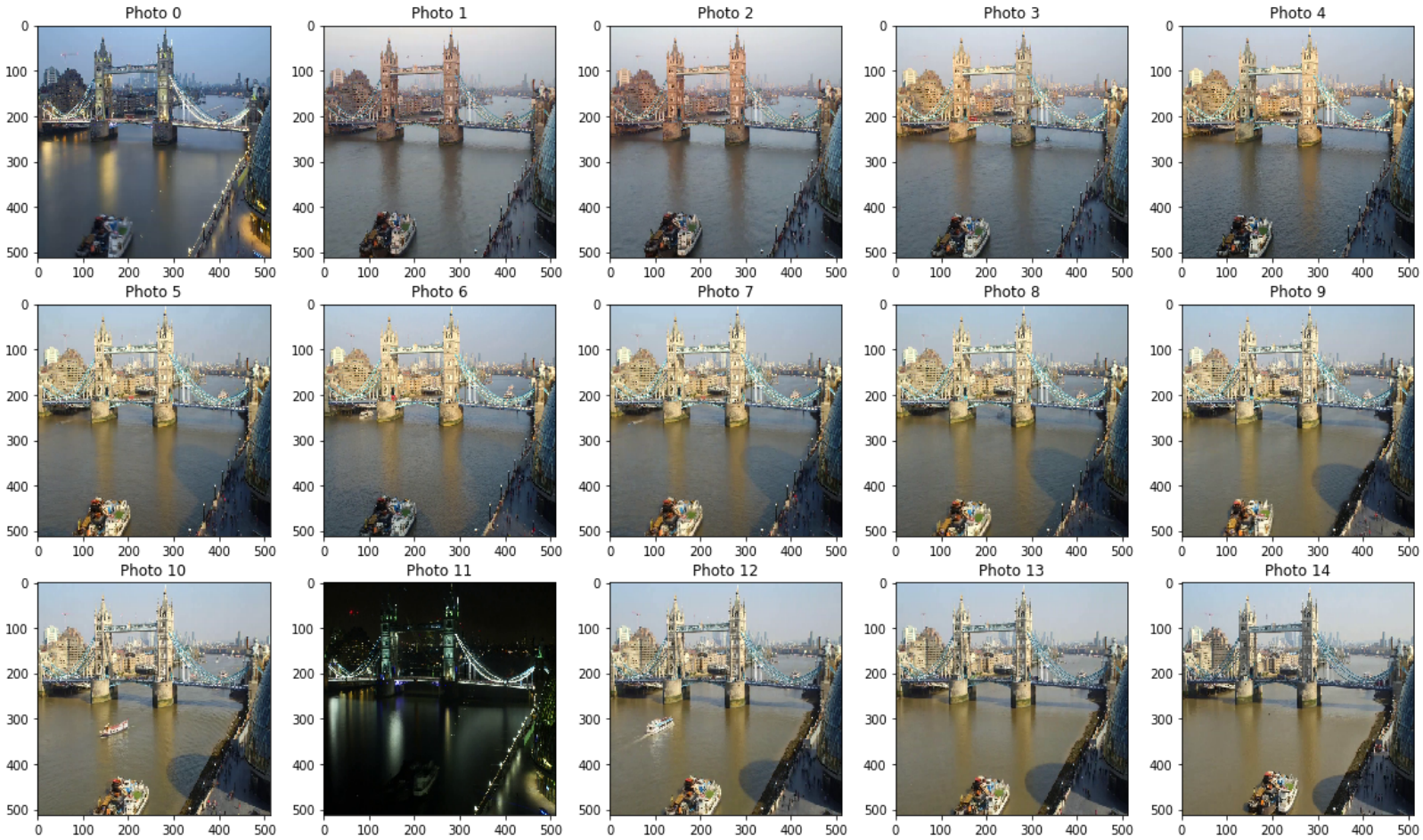
```
###Data Loading & Preprocessing

#importing images
bridge = sorted(glob('bridge/*'))
print("Number of photos: ", len(bridge))

data = []

for im in bridge: #resizing to 512 X 512 px
    pic = Image.open(im) #open image file
    rsz = pic.resize((512, 512), Image.NEAREST) #nearest-neighbor-sampling
    pics = np.array(rsz) #into array form
    data.append(pics.flatten()) #flattening the array form
```

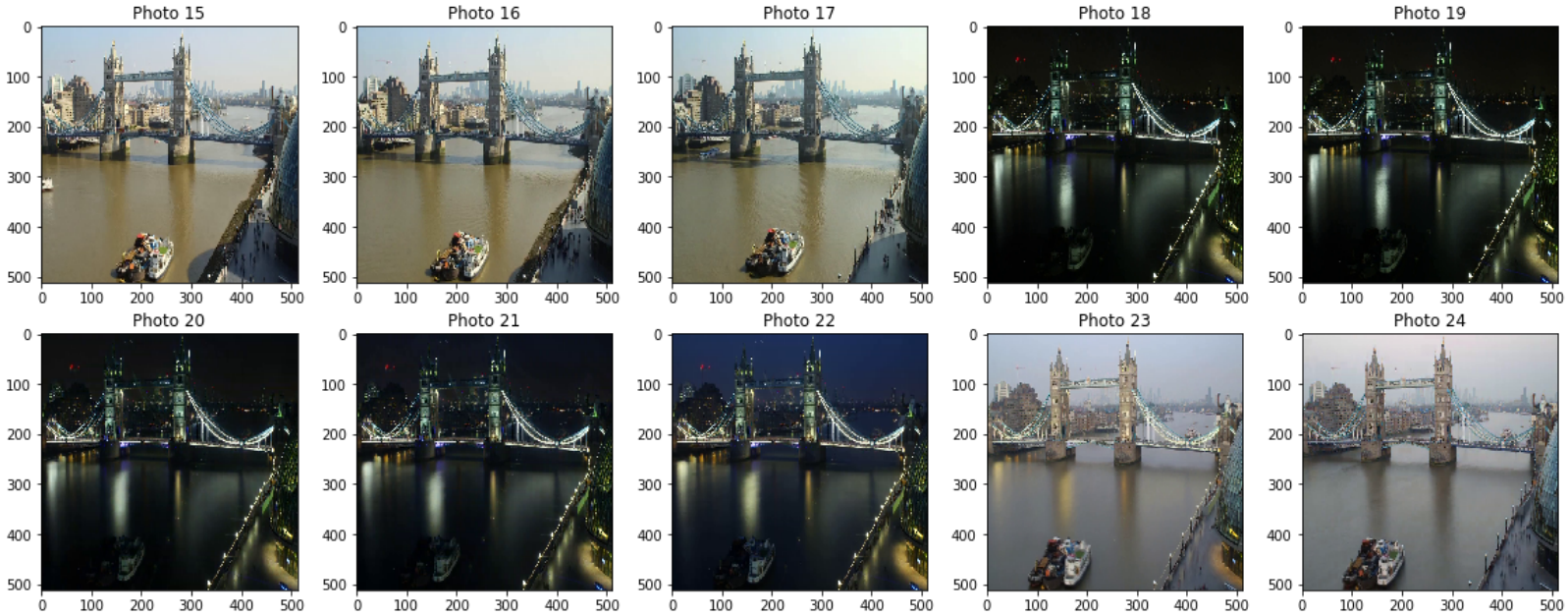
```
Number of photos:  25
```

In [3]:

```
#plotting the images
fig, axes = plt.subplots(5,5,figsize=(20,20))
for i,ax in enumerate(axes.flatten()):
    ax.set_title(f'Photo {i}')
    ax.imshow(data[i].reshape(512,512,3))
print("All Tower Bridge photos")
plt.show()
```

```
All Tower Bridge photos
```





In [4]:

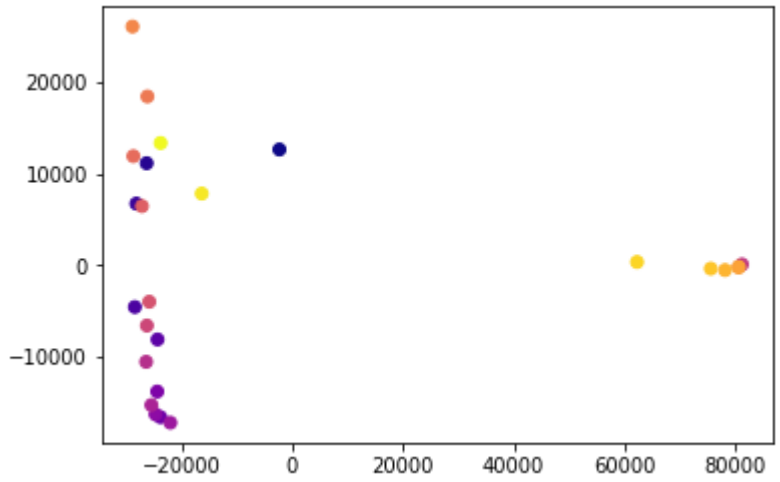
```
#projecting photos to 2D, using PCA
pca = PCA(n_components = 2) #2 here is the no. of components
proj = pca.fit_transform(data) #projections
print(np.sum(pca.explained_variance_ratio_[:2])) #variance
```

```
0.8935741158098824
```

PCA gives us ~ 89% of variance captured from the images. This is a good value and we were able to achieve this because the photo was taken from a constant angle and the only varying factors were the moving objects like people, boat, and lighting.

In [5]:

```
#plotting 2D projected points for the photos
plt.scatter([point[0] for point in proj], [point[1] for point in proj],
            cmap = "plasma", c = range(len(proj)))
plt.show()
```

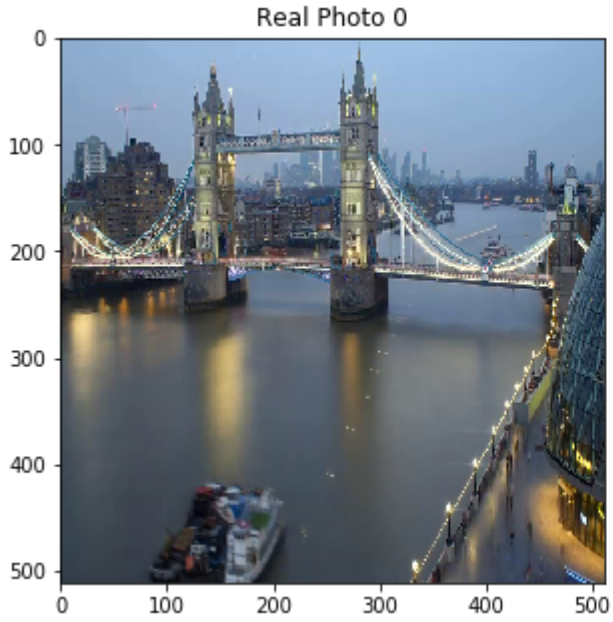


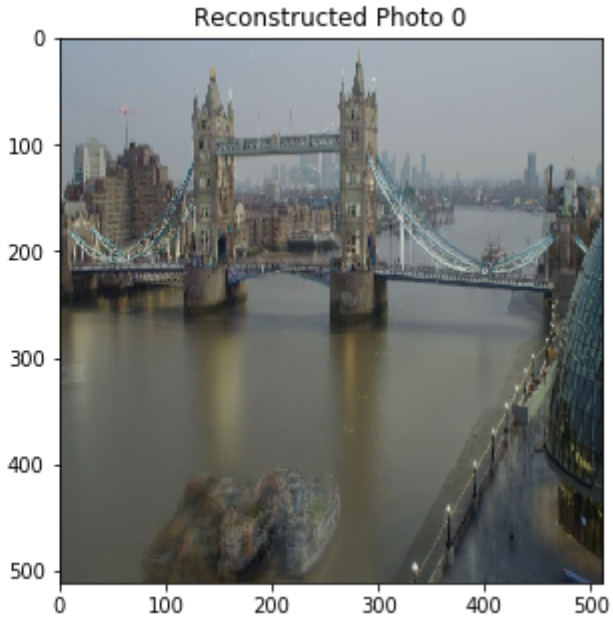
We see many outliers points on the right, in the plot above, which seems to be because we have same number of photos clicked in dark

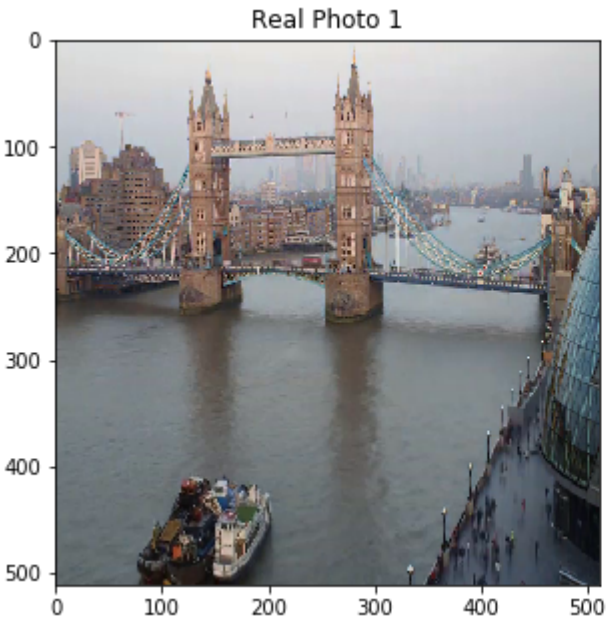
In [6]:

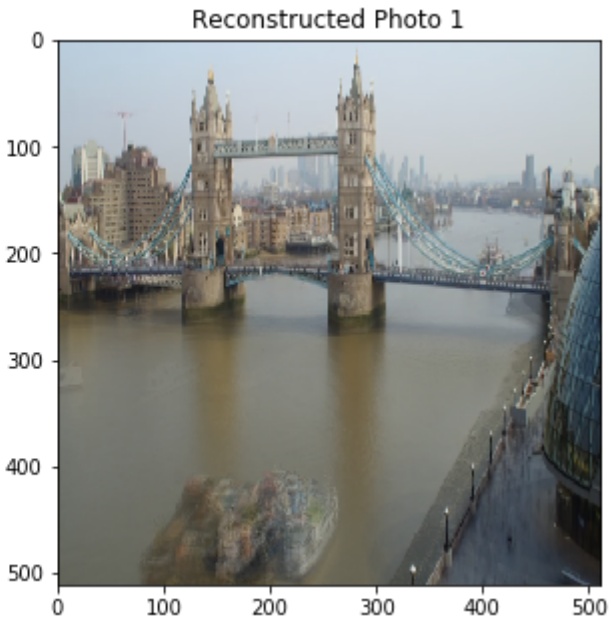
```
#showing reconstruction from low dimension projection
recon = pca.inverse_transform(proj)
for i in range(25):
    fig, axes = plt.subplots(1, 2, figsize=(15,5))
    axes[0].set_title(f'Real Photo {i}')
    axes[0].imshow(data[i].reshape(512,512,3))
    axes[1].set_title(f'Reconstructed Photo {i}')
    axes[1].imshow(recon[i].reshape(512,512,3).astype('uint8'))
```

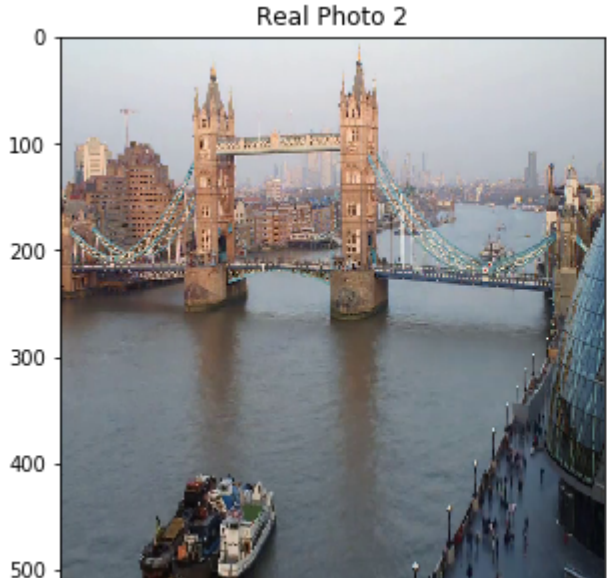
```
C:\Users\rhyth\Anaconda3\lib\site-packages\matplotlib\pyplot.py:522: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warning`).
  max_open_warning, RuntimeWarning)
```

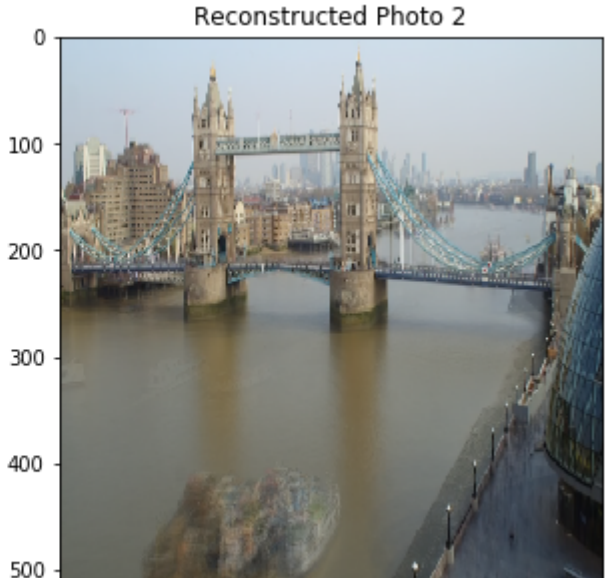


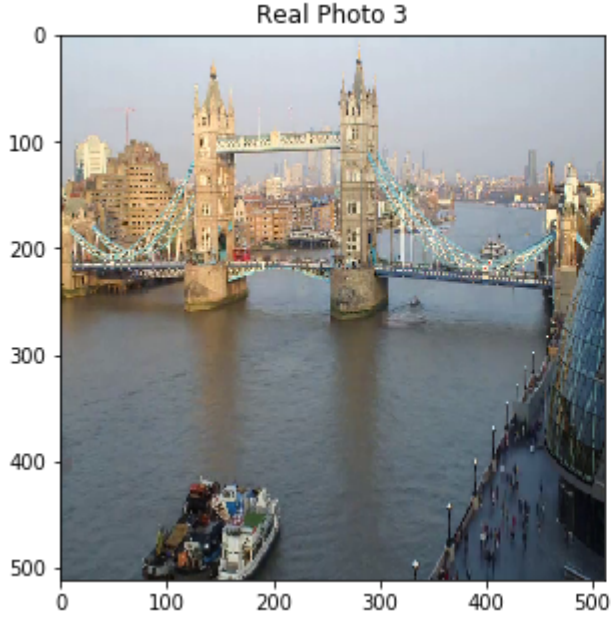


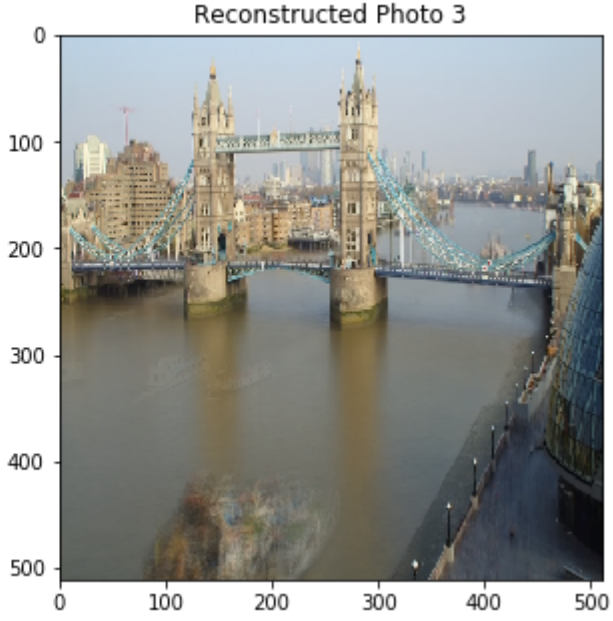

Real Photo 1
0
100
200
300
400
500
0
100
200
300
400
500

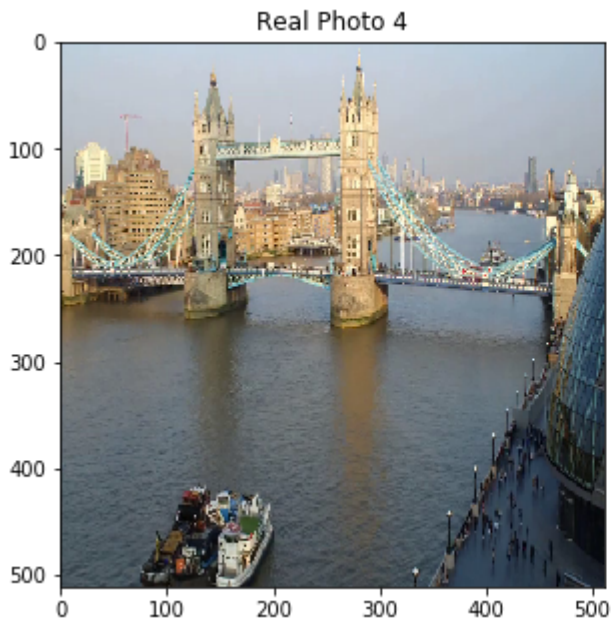

Reconstructed Photo 1
0
100
200
300
400
500
0
100
200
300
400
500

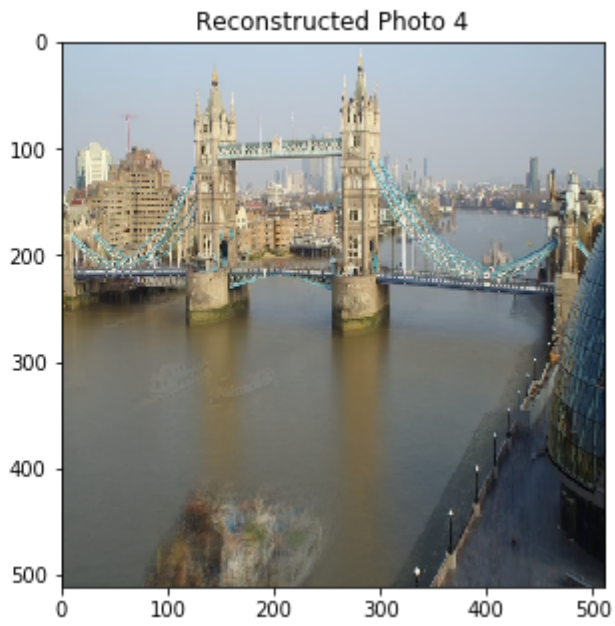

Real Photo 2
0
100
200
300
400
500


Reconstructed Photo 2
0
100
200
300
400
500


Real Photo 3
0
100
200
300
400
500
0
100
200
300
400
500

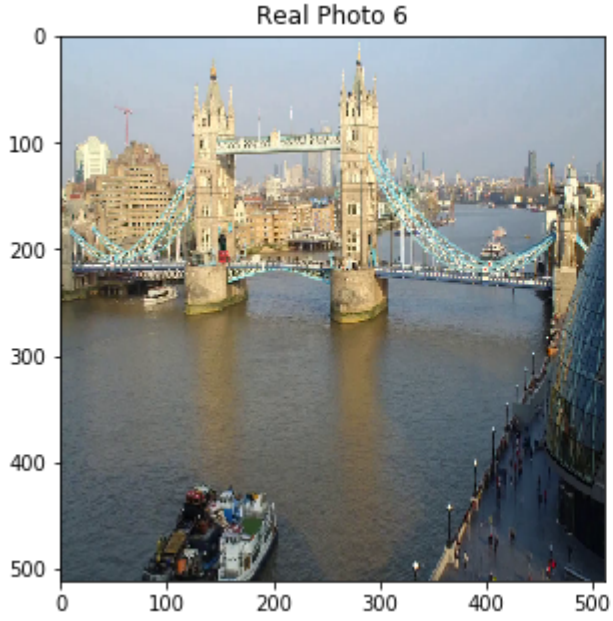

Reconstructed Photo 3
0
100
200
300
400
500
0
100
200
300
400
500

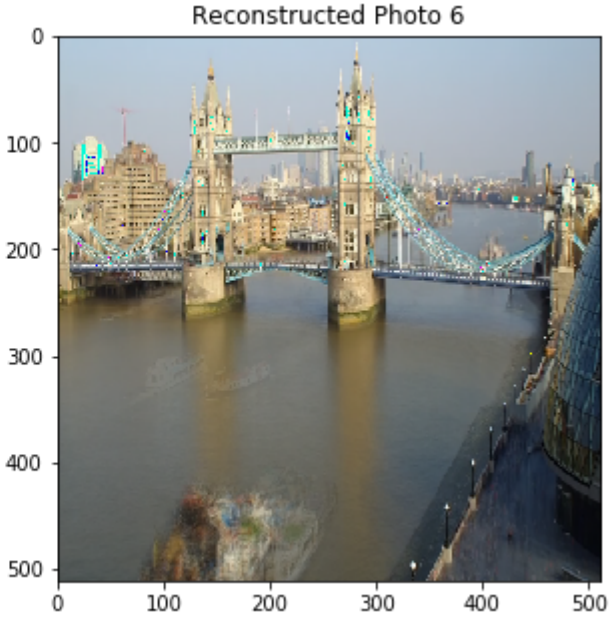

Real Photo 4
0
100
200
300
400
500
0
100
200
300
400
500

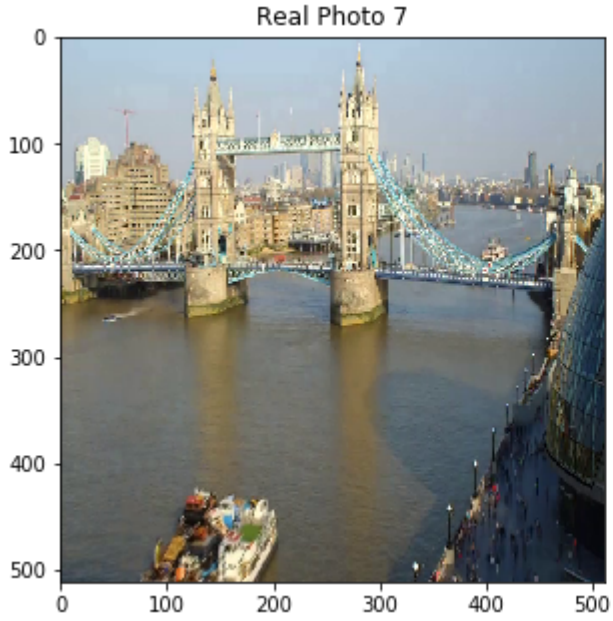

Reconstructed Photo 4
0
100
200
300
400
500
0
100
200
300
400
500

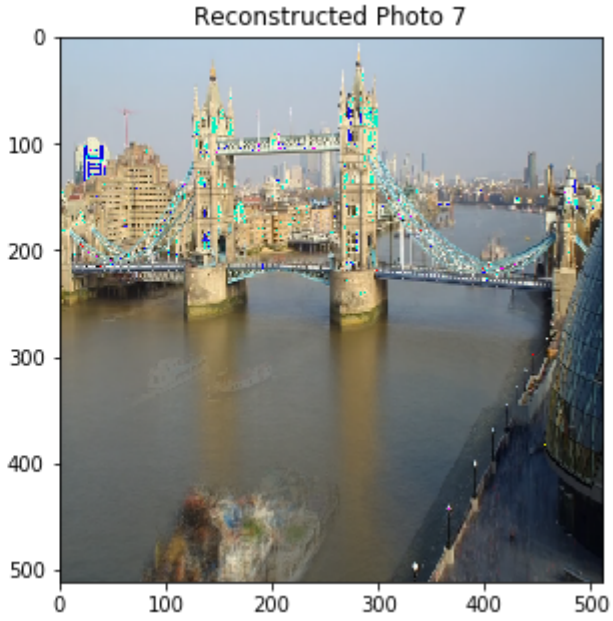

Real Photo 5
0
100
200
300

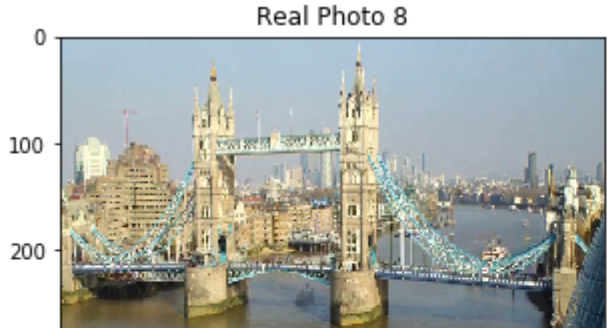

Reconstructed Photo 5
0
100
200
300


Real Photo 6
0
100
200
300
400
500
0
100
200
300
400
500

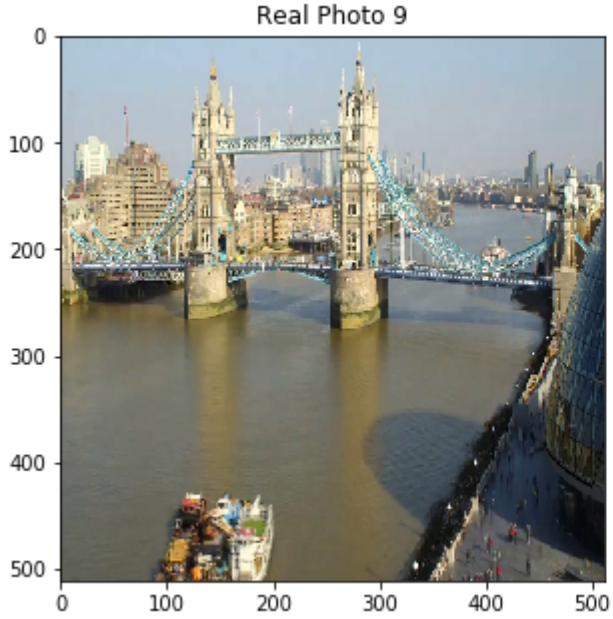

Reconstructed Photo 6
0
100
200
300
400
500
0
100
200
300
400
500

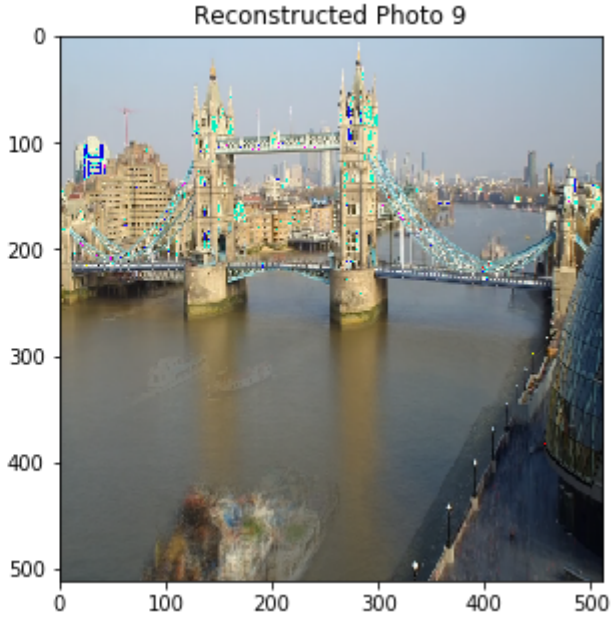

Real Photo 7
0
100
200
300
400
500
0
100
200
300
400
500

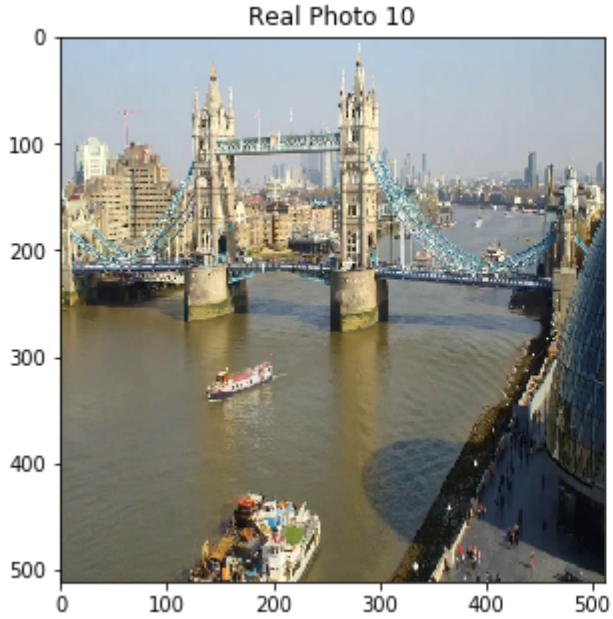

Reconstructed Photo 7
0
100
200
300
400
500
0
100
200
300
400
500

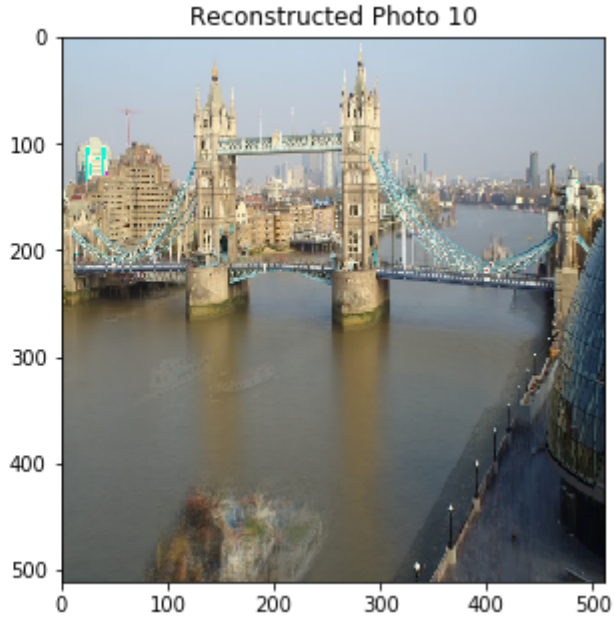

Real Photo 8
0
100
200

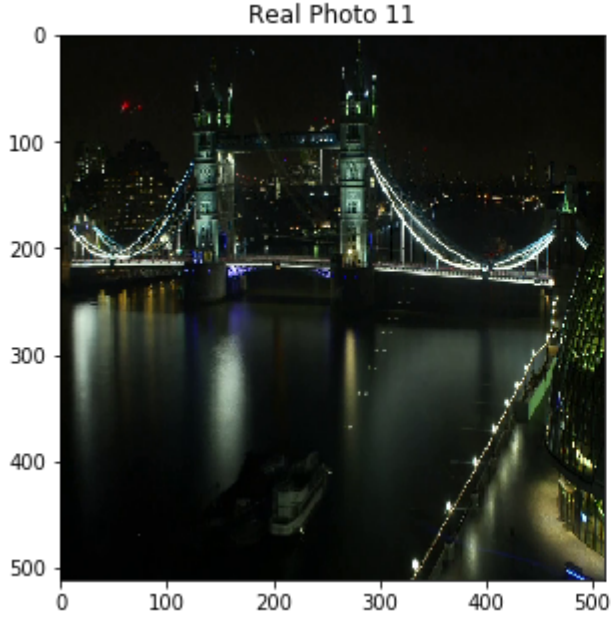

Reconstructed Photo 8
0
100
200

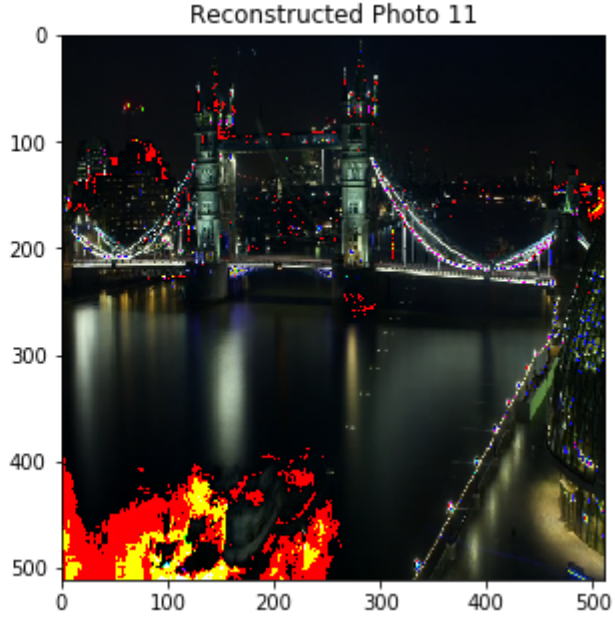

Real Photo 9
0
100
200
300
400
500
0
100
200
300
400
500

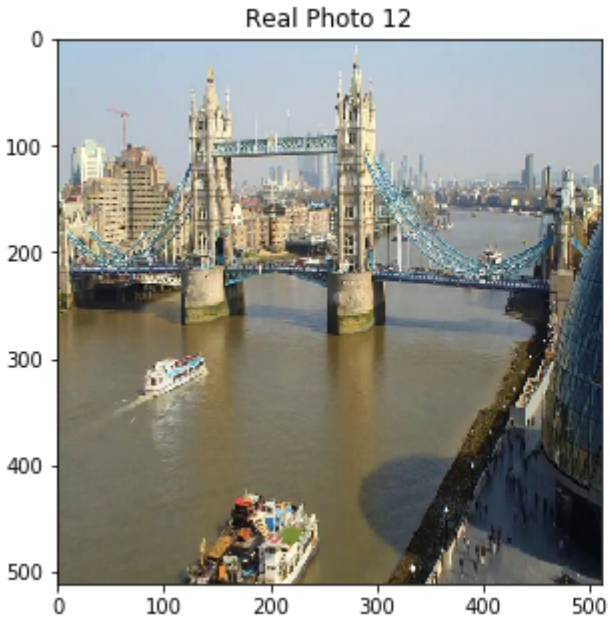

Reconstructed Photo 9
0
100
200
300
400
500
0
100
200
300
400
500

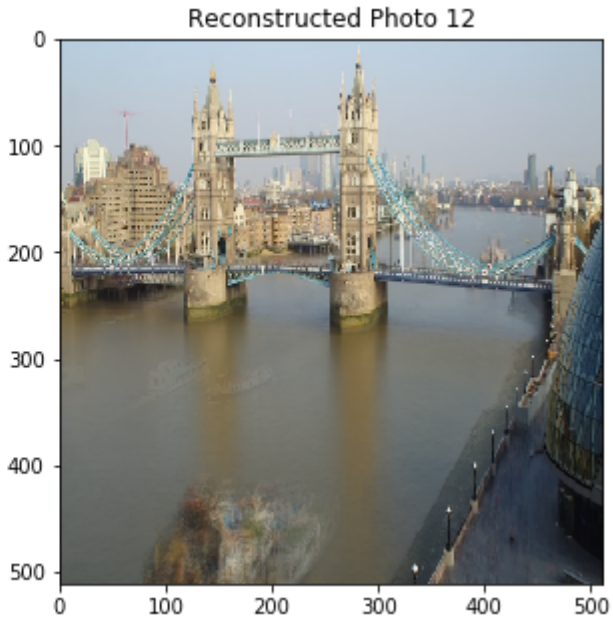

Real Photo 10
0
100
200
300
400
500
0
100
200
300
400
500

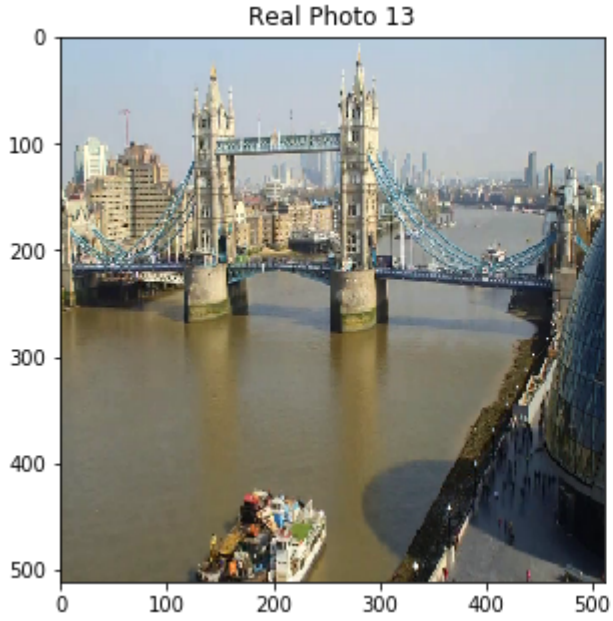

Reconstructed Photo 10
0
100
200
300
400
500
0
100
200
300
400
500

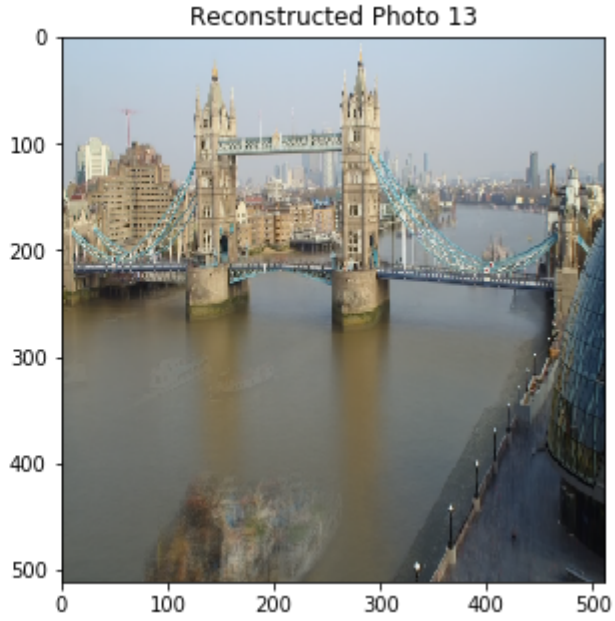

Real Photo 11
0
100
200
300
400
500
0
100
200
300
400
500

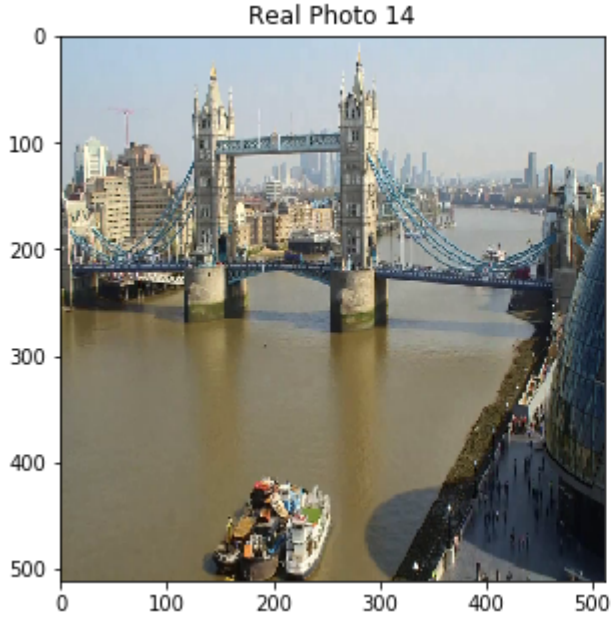

Reconstructed Photo 11
0
100
200
300
400
500
0
100
200
300
400
500

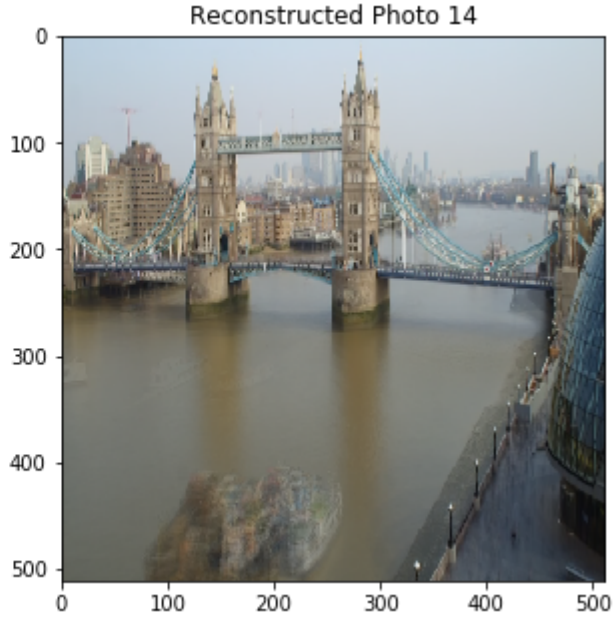

Real Photo 12
0
100
200
300
400
500
0
100
200
300
400
500

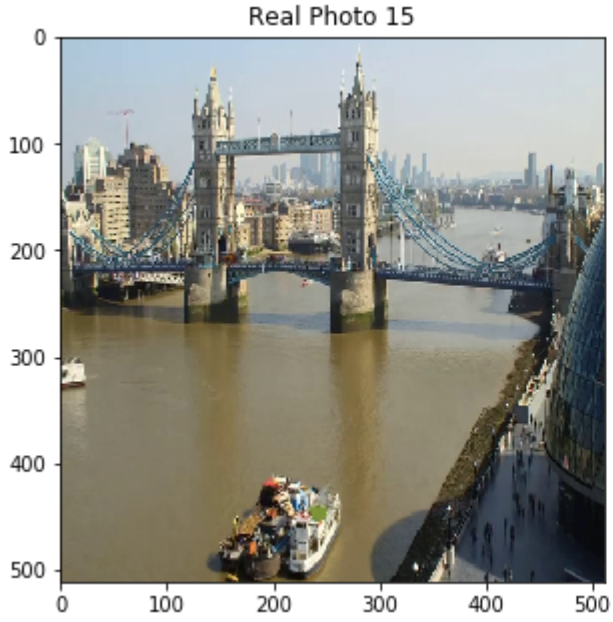

Reconstructed Photo 12
0
100
200
300
400
500
0
100
200
300
400
500

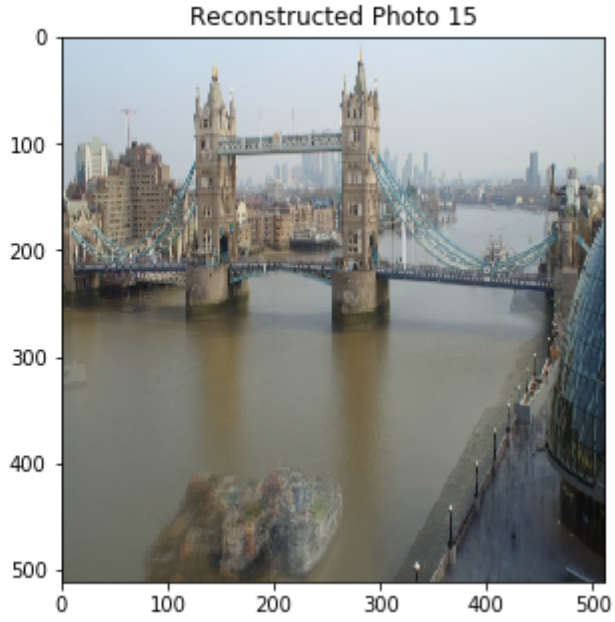

Real Photo 13
0
100
200
300
400
500
0
100
200
300
400
500


Reconstructed Photo 13
0
100
200
300
400
500
0
100
200
300
400
500


Real Photo 14
0
100
200
300
400
500
0
100
200
300
400
500

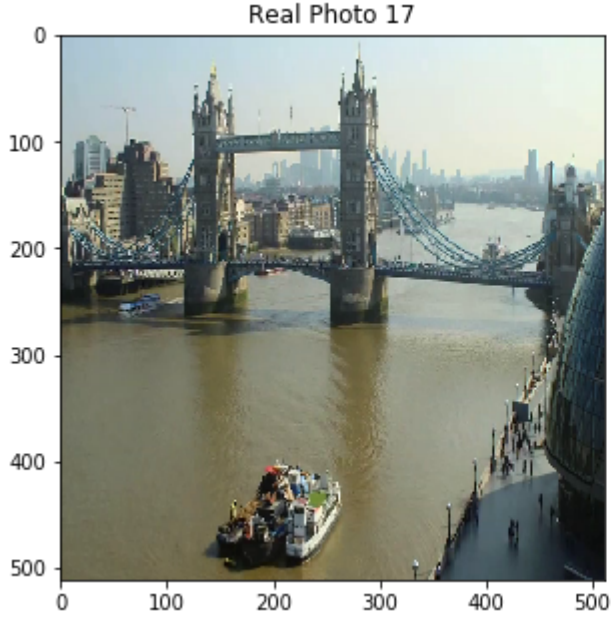

Reconstructed Photo 14
0
100
200
300
400
500
0
100
200
300
400
500

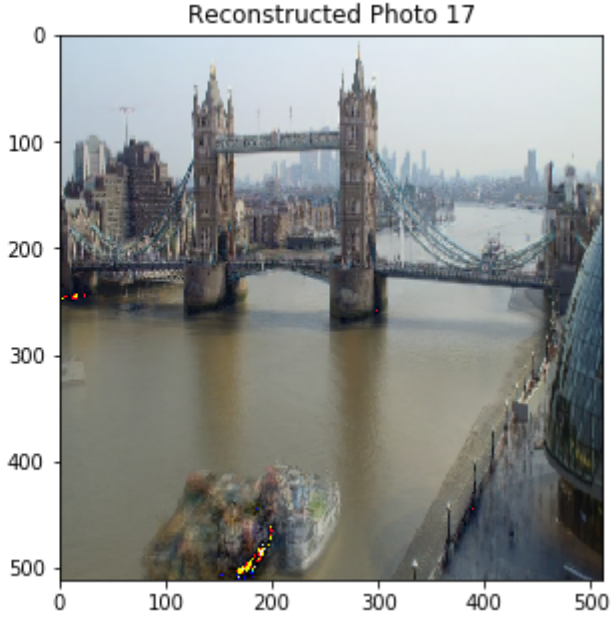

Real Photo 15
0
100
200
300
400
500
0
100
200
300
400
500

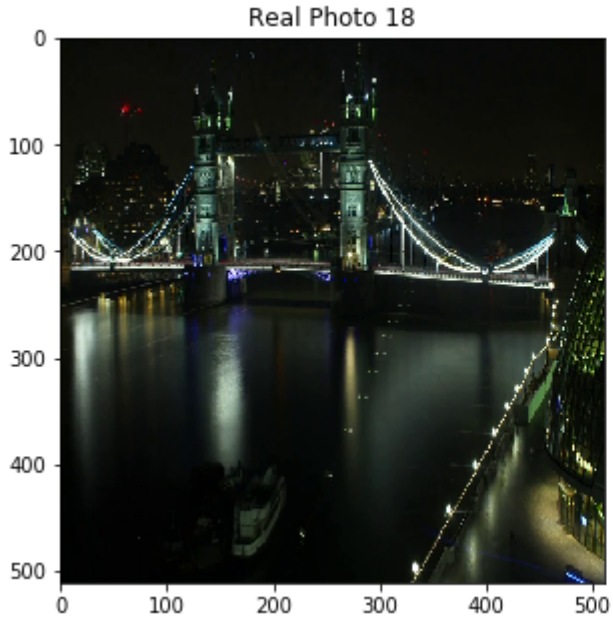

Reconstructed Photo 15
0
100
200
300
400
500
0
100
200
300
400
500

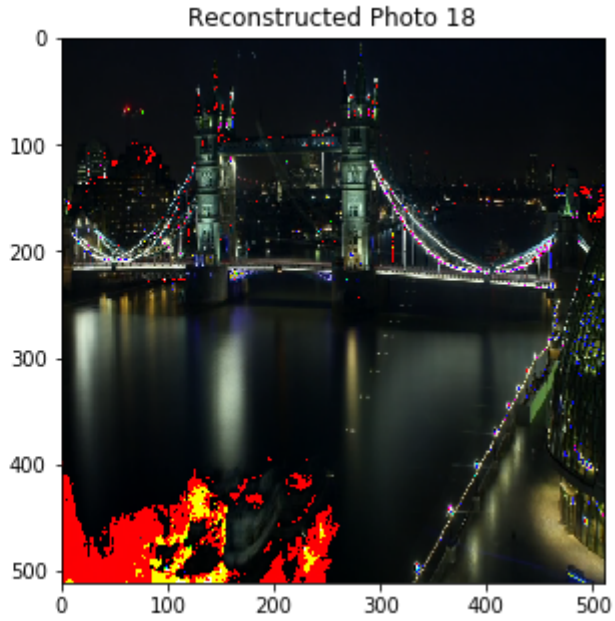

Real Photo 16
0
100
200
300

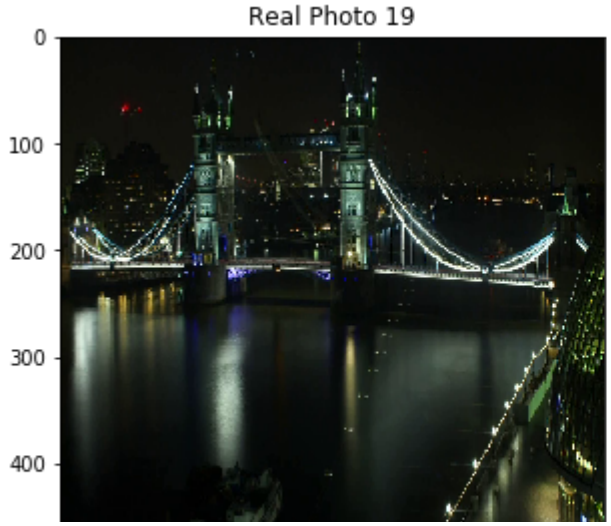

Reconstructed Photo 16
0
100
200
300


Real Photo 17
0
100
200
300
400
500
0
100
200
300
400
500

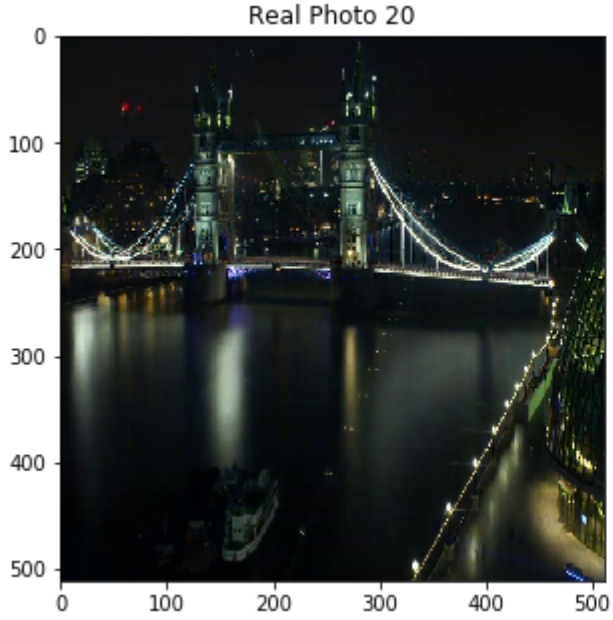

Reconstructed Photo 17
0
100
200
300
400
500
0
100
200
300
400
500

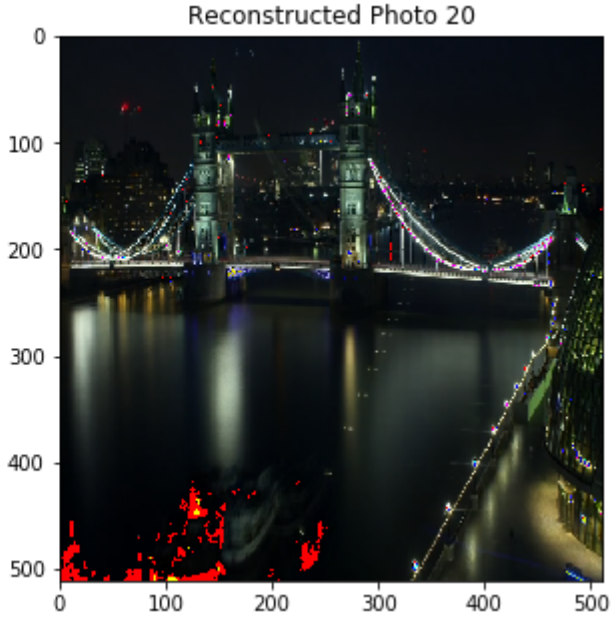

Real Photo 18
0
100
200
300
400
500
0
100
200
300
400
500

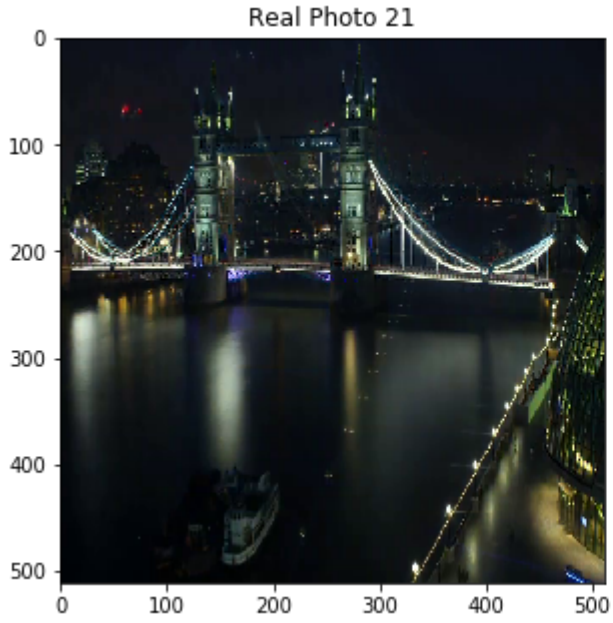

Reconstructed Photo 18
0
100
200
300
400
500
0
100
200
300
400
500

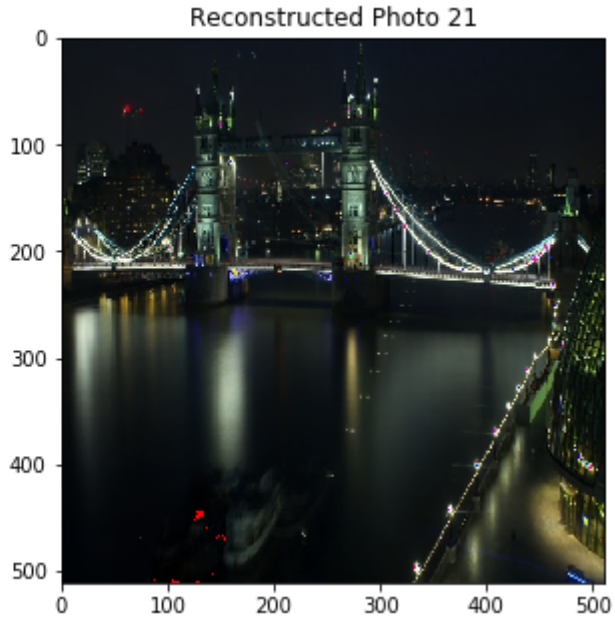

Real Photo 19
0
100
200
300
400

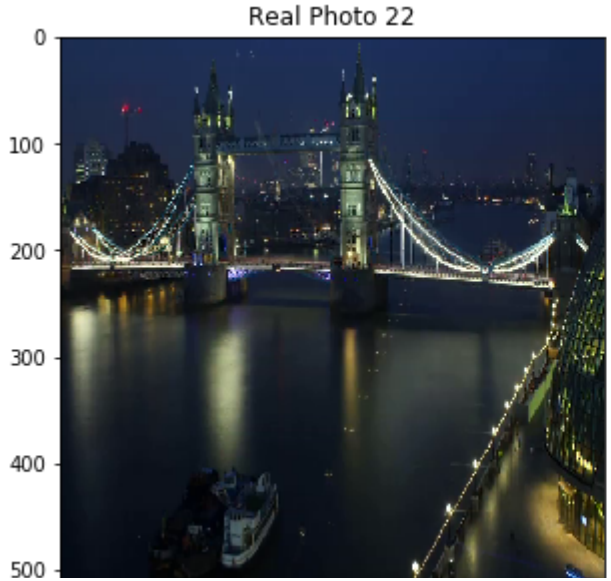

Reconstructed Photo 19
0
100
200
300
400

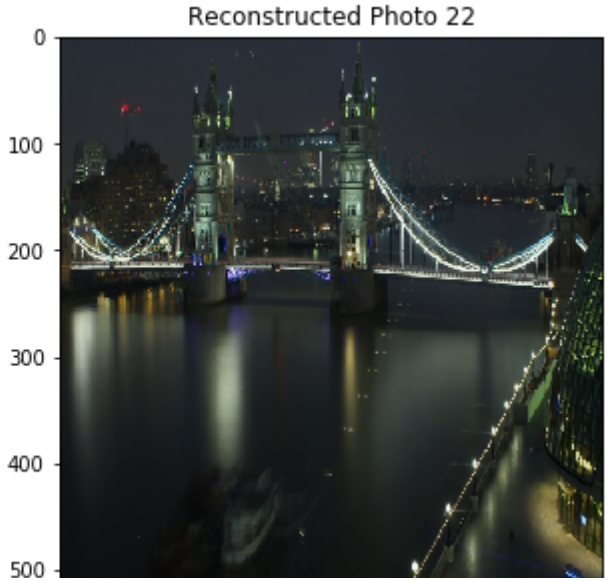

Real Photo 20
0
100
200
300
400
500
0
100
200
300
400
500

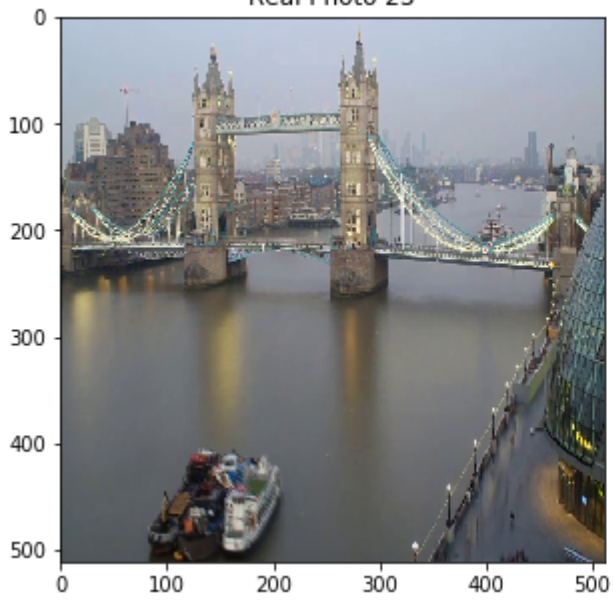

Reconstructed Photo 20
0
100
200
300
400
500
0
100
200
300
400
500

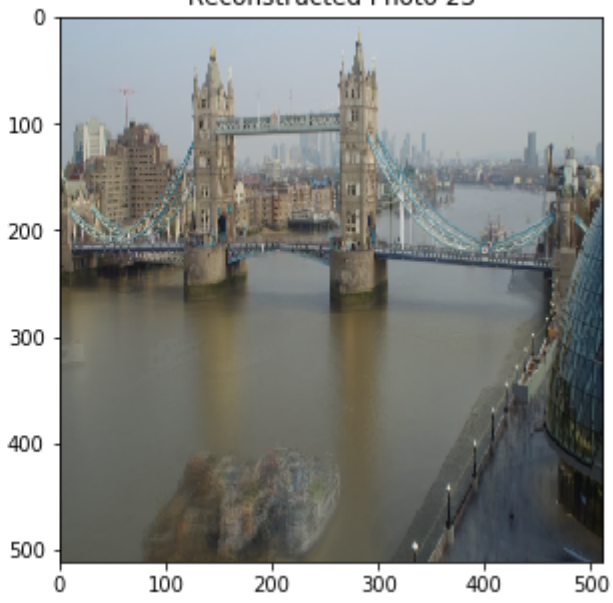

Real Photo 21
0
100
200
300
400
500
0
100
200
300
400
500

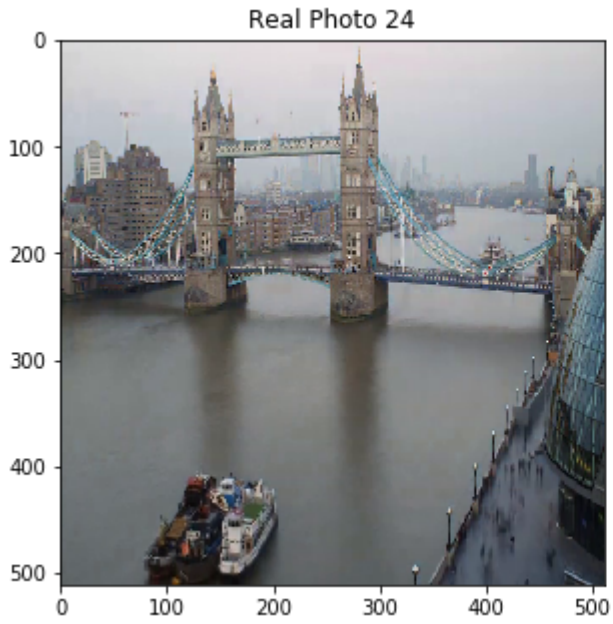

Reconstructed Photo 21
0
100
200
300
400
500
0
100
200
300
400
500

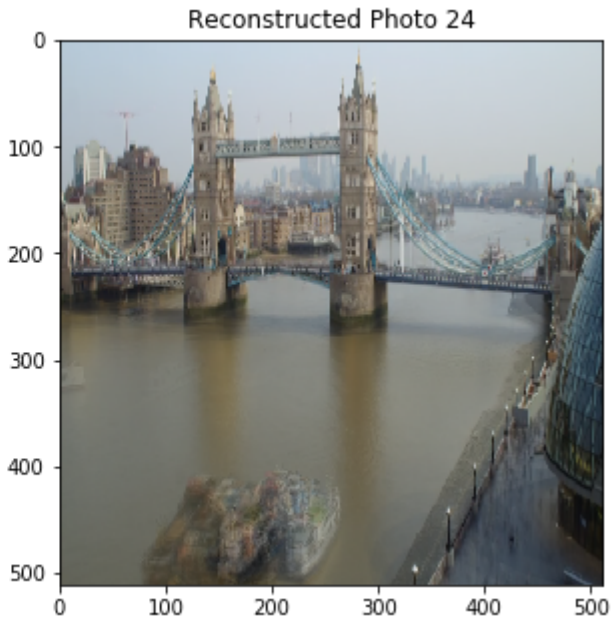

Real Photo 22
0
100
200
300
400
500


Reconstructed Photo 22
0
100
200
300
400
500


Real Photo 23
0
100
200
300
400
500
0
100
200
300
400
500


Reconstructed Photo 23
0
100
200
300
400
500
0
100
200
300
400
500

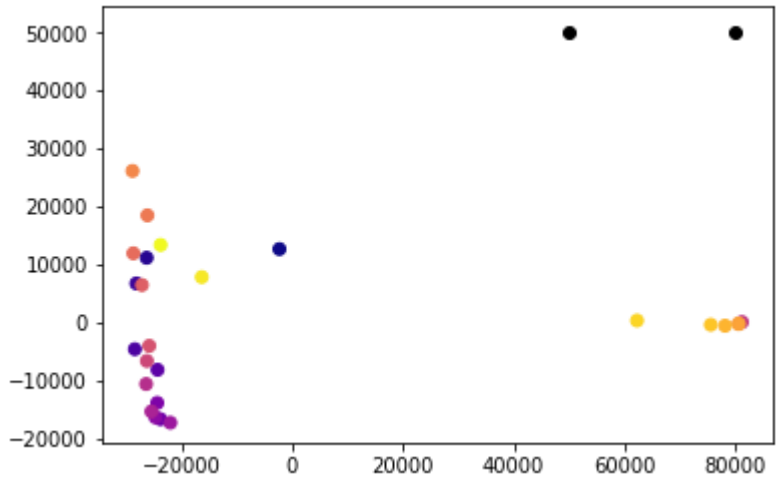




Reconstruction of real photos look quite accurate. Although a few of the photos have noisy colored spots where the reconstruction isn't as good as the others, mostly of photos clicked in dark this could due to less data of pictures clicked in dark. Also, the boat and public isn't reconstructed properly in any as they weren't captured identical in any photo.

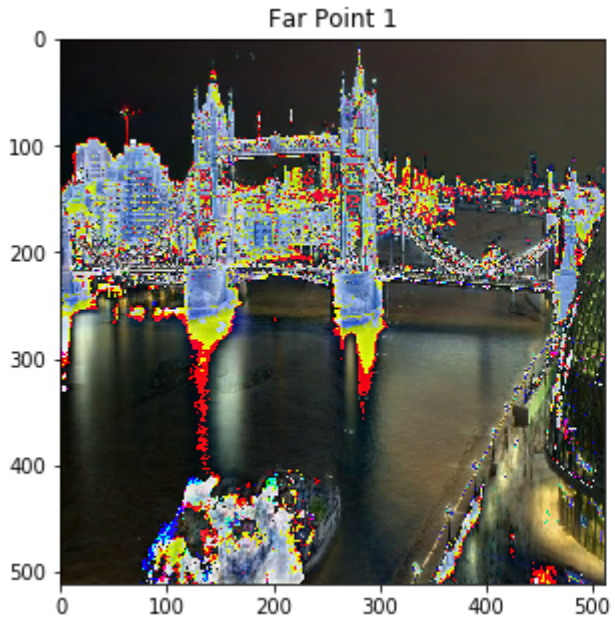
In [7]:

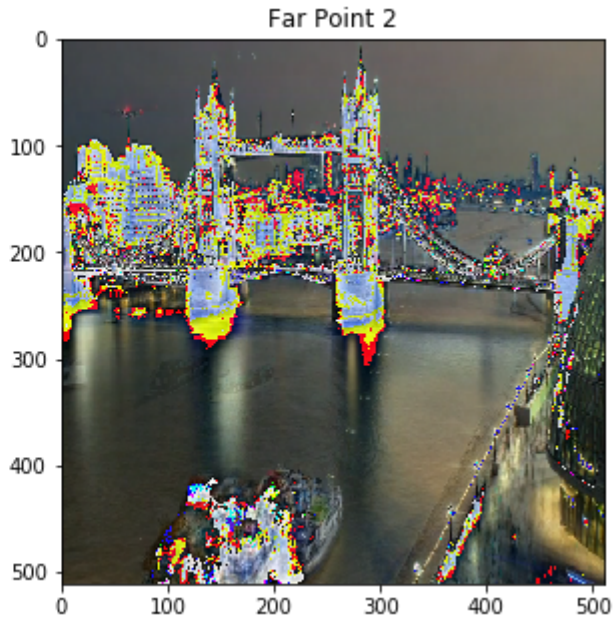
```
plt.scatter([point[0] for point in proj], [point[1] for point in proj],
            cmap = "plasma", c = range(len(proj)))
plt.scatter(80000, 50000, c = "black") #plotting a FAR point
plt.scatter(50000, 50000, c = "black") #plotting a FAR point
plt.show()
```



In [8]:

```
#plotting the reconstruction of black far points plotted above
far1 = pca.inverse_transform([80000,50000]) #far point 1
far2 = pca.inverse_transform([50000,50000]) #far point 2
fig, axes = plt.subplots(1,2,figsize=(15,5))
axes[0].set_title(f'Far Point 1')
axes[0].imshow(far1.reshape(512,512,3).astype('uint8'))
axes[1].set_title(f'Far Point 2')
axes[1].imshow(far2.reshape(512,512,3).astype('uint8'))
plt.show()
```





	Reconstruction of far points seem terrible but they do restore a few parts accurate and the structure/skeleton of the image although with noisy colors.