Image compression using SVD

In [1]:
```
from PIL import Image
```

In [2]:
```
im=Image.open(r"D:\dataset\scenery3.png")
im.show() # shows image in any image viewer
```

In [2]:
```
# viewing image in notebook
import cv2
import numpy as np
from numpy.linalg import svd
import matplotlib.pyplot as plt
img=cv2.imread("D:/dataset/scenery3.png")
plt.figure(figsize=(10,10))
plt.axis("off") # to not show axis
plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB)) # COLOR_BGR2RGB changes bgr to rgb ,
plt.title("Original RGB image")
```

Out[2]:
```
Text(0.5, 1.0, 'Original RGB image')
```

In [3]:
```
# calculating SVD
u,s,v=np.linalg.svd(img,full_matrices=False)
print("u.shape :",u.shape,"s.shape :",s.shape,"v.shape :",v.shape)
```

```
u.shape : (720, 1280, 3) s.shape : (720, 3) v.shape : (720, 3, 3)
```

In [4]:
```
gray_img=(cv2.cvtColor(img,cv2.COLOR_BGR2GRAY))
plt.figure(figsize=(10,10))
plt.imshow(gray_img,cmap="gray")
plt.title("gray image")
plt.axis("off")
```

Out[4]: (-0.5, 1279.5, 719.5, -0.5)

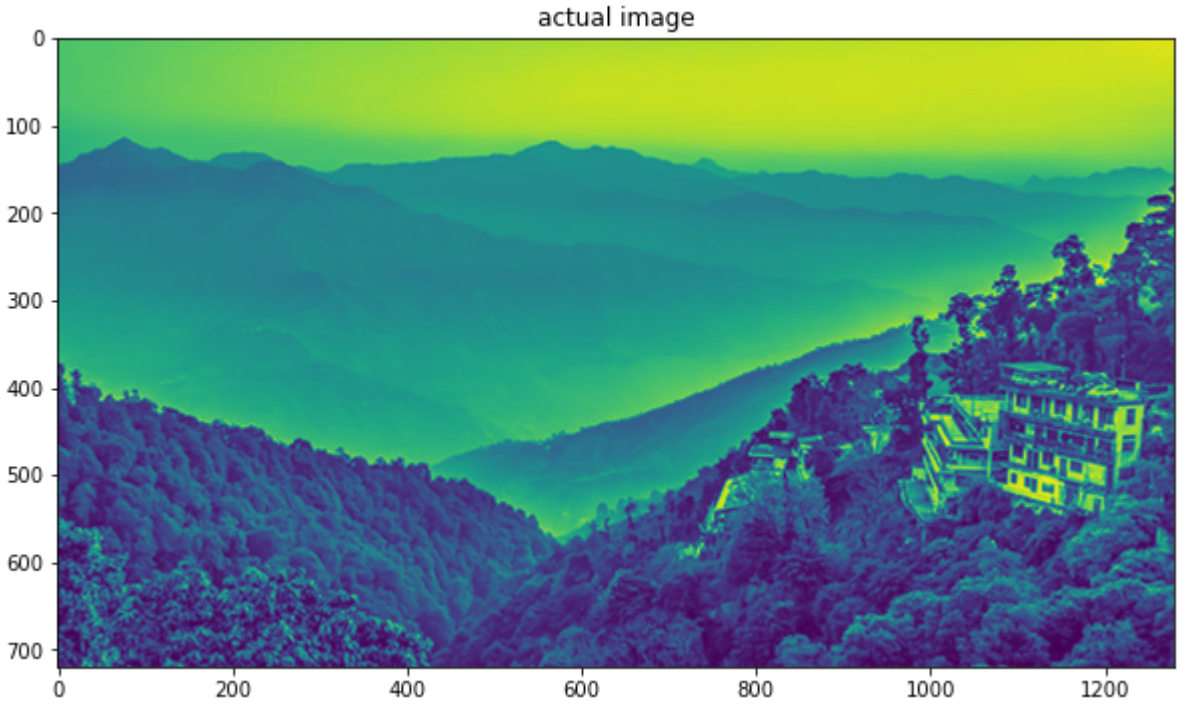
In [5]:
```
u,s,v=np.linalg.svd(gray_img,full_matrices=False)
print("u.shape :",u.shape,"s.shape :",s.shape,"v.shape :",v.shape)
```

```
u.shape : (720, 720) s.shape : (720,) v.shape : (720, 1280)
```

In [6]:
```
 # there are 720 linearly independent components
```

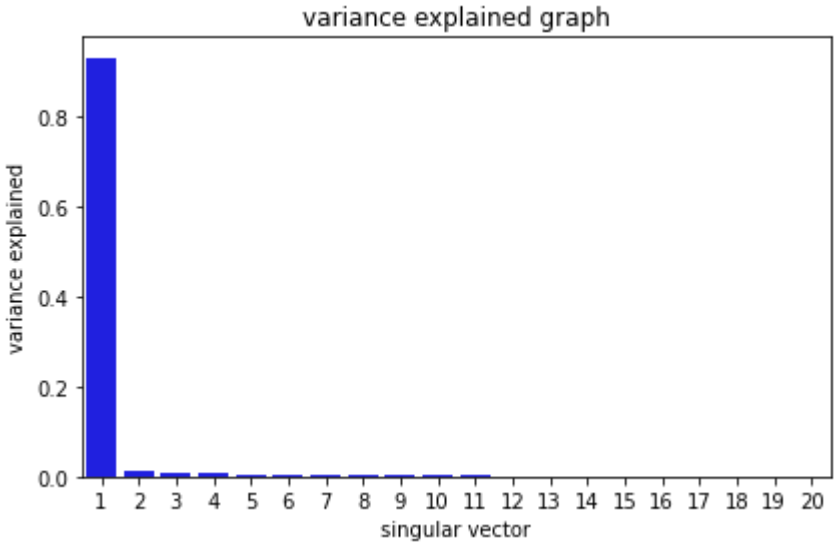
In [7]:
```
plt.figure(figsize=(10,10))
orig_rank=u[:,:]@np.diag(s[:])@v[:,:] # original image stays in BGR
plt.imshow(orig_rank)
plt.title("actual image")
```

Out[7]:
```
Text(0.5, 1.0, 'actual image')
```

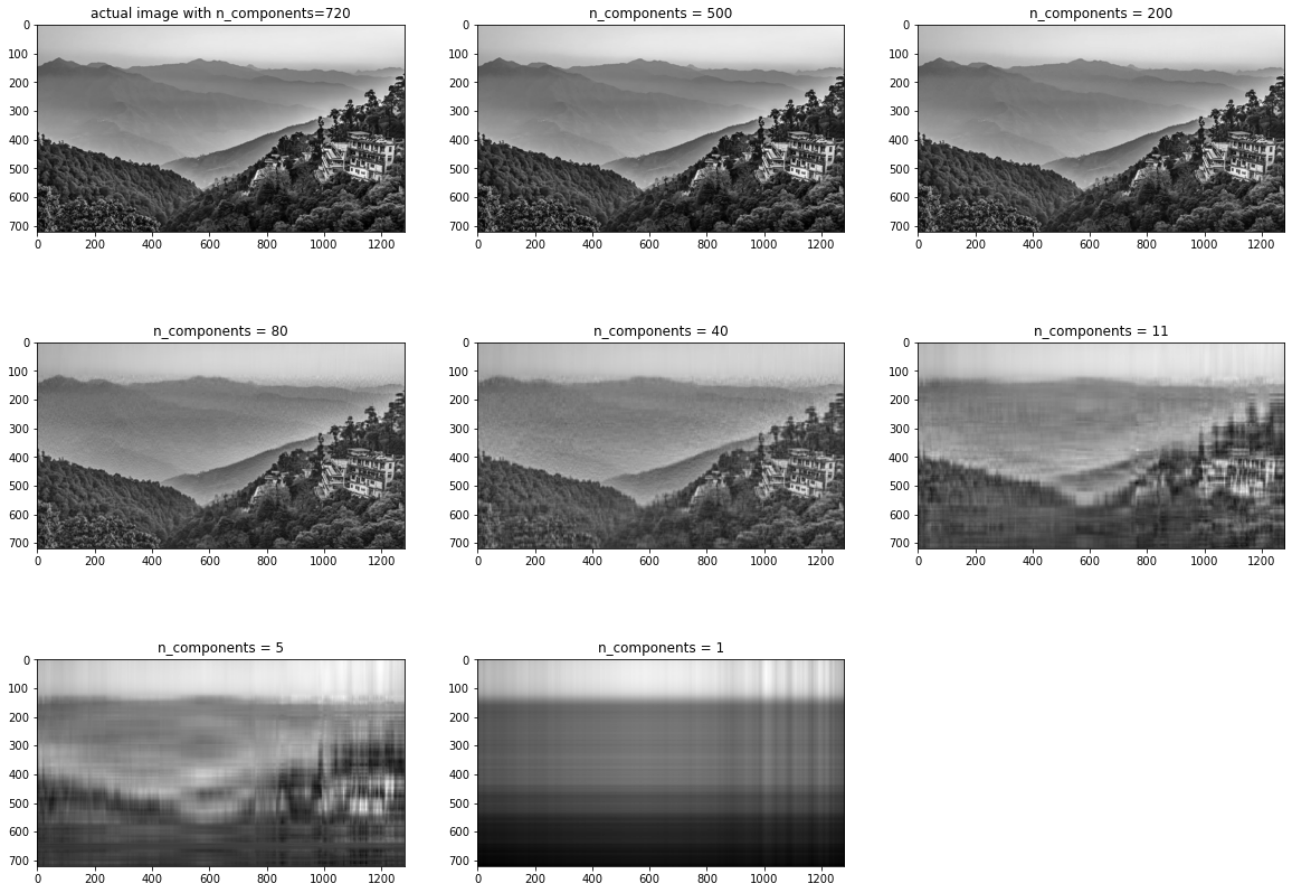


In [8]:
```
import seaborn as sns
var_exp=np.round(s**2/np.sum(s**2),decimals=6)
print("var explained by top 20 sv : \n",var_exp[0:20])
sns.barplot(x=list(range(1,21)),y=var_exp[0:20],color="blue")
plt.title("variance explained graph")
plt.xlabel("singular vector")
plt.ylabel("variance explained")
plt.tight_layout()
plt.show()
```

```
var explained by top 20 sv :
 [9.29603e-01 1.31580e-02 1.00140e-02 5.86800e-03 3.23900e-03 2.81800e-03
 2.56900e-03 2.03300e-03 1.88900e-03 1.55100e-03 1.41000e-03 1.26800e-03
 1.15900e-03 9.39000e-04 8.77000e-04 8.28000e-04 7.91000e-04 7.63000e-04
 6.65000e-04 6.18000e-04]
```



In [16]:
```
comps=[720,500,200,80,40,11,5,1]
plt.figure(figsize=(20,20))
for i in range(len(comps)):
    low_rank=u[:,:comps[i]]@np.diag(s[:comps[i]])@v[:comps[i],:]
    if i==0:
        plt.subplot(4,3,i+1),
        plt.imshow(low_rank,cmap="gray")
        plt.title(f'actual image with n_components={comps[i]}' )
    else:
        plt.subplot(4,3,i+1),
        plt.imshow(low_rank,cmap="gray"),
        plt.title(f'n_components = {comps[i]}')
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: