```
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import numpy as np
```

```
# Replace 'your_image.jpg' with the actual file path of your JPG image
image_path = r"C:\Users\navde\Downloads\sat_image_plaksha.jpg"

# Load the image
image = cv2.imread(image_path)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

greyscale image

```
# Convert the image to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Display the grayscale image
plt.imshow(gray_image, cmap='gray')
plt.axis('off')
plt.show()
```

```
#converting to float64
float64_image = gray_image.astype(np.float64)


#checking the datatype after conversion
print('Datatype:', float64_image.dtype)
```

```
    Datatype: float64
```

```
# Calculate the mean of each variable (column)
column_means = np.mean(float64_image, axis=0)
```

```
# Calculate the mean of each variable (column)
column_means = np.mean(float64_image, axis=0)

# Subtract the mean of each variable from the dataset
centered_X = float64_image - column_means

# Display the  image after centring the data
plt.imshow(centered_X, cmap='gray')
plt.axis('off')
plt.show()
```

```
print("Centered Dataset:")
print(centered_X)
```

```
Centered Dataset:
[[ 32.79924242  40.41666667  38.18560606 ...  -3.64015152   0.26893939
   -2.35984848]
 [ 38.79924242  40.41666667  35.18560606 ...  -6.64015152  -4.73106061
   -8.35984848]
 [ 30.79924242  47.41666667  42.18560606 ...  -3.64015152   1.26893939
    0.64015152]
 ...
 [ 61.79924242  67.41666667  30.18560606 ... -38.64015152 -38.73106061
  -42.35984848]
 [  6.79924242  18.41666667   1.18560606 ... -38.64015152 -39.73106061
  -40.35984848]
 [  6.79924242  11.41666667   4.18560606 ... -41.64015152 -44.73106061
  -45.35984848]]
```

```
# Calculate the covariance matrix of the mean-centered data
cov_matrix = np.cov(centered_X,rowvar=False)

print("Covariance Matrix of Centered Data:")
print(cov_matrix)
```

```
Covariance Matrix of Centered Data:
[[1518.87589296 1343.12579214 1121.83208031 ... -167.01493548
  -126.55036583 -139.92042574]
 [1343.12579214 1545.76489227 1265.9413815  ... -140.70183777
   -89.85392902  -73.36280101]
 [1121.83208031 1265.9413815  1425.59659811 ... -144.06704401
   -86.40752103  -38.05082671]
 ...
 [-167.01493548 -140.70183777 -144.06704401 ... 1254.67990264
  1140.29068729 1075.81819622]
 [-126.55036583  -89.85392902  -86.40752103 ... 1140.29068729
  1184.58519127 1148.42033932]
 [-139.92042574  -73.36280101  -38.05082671 ... 1075.81819622
  1148.42033932 1233.26545397]]
```

```
cov_matrix.shape
```

```
(300, 300)
```

```
# Compute the eigenvalues and eigenvectors of the covariance matrix
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

print("Eigenvalues:")
print(eigenvalues)
```

```
Eigenvalues:
[ 5.63294113e+04+0.00000000e+00j  3.66340016e+04+0.00000000e+00j
  2.93994813e+04+0.00000000e+00j  2.28486224e+04+0.00000000e+00j
  1.73508832e+04+0.00000000e+00j  1.60313480e+04+0.00000000e+00j
  1.31732244e+04+0.00000000e+00j  1.05589938e+04+0.00000000e+00j
  9.19039668e+03+0.00000000e+00j  8.71147108e+03+0.00000000e+00j
  6.67937330e+03+0.00000000e+00j  6.74188435e+03+0.00000000e+00j
  5.34706126e+03+0.00000000e+00j  4.76454528e+03+0.00000000e+00j
  4.48727389e+03+0.00000000e+00j  4.21526060e+03+0.00000000e+00j
  4.05755706e+03+0.00000000e+00j  3.82091642e+03+0.00000000e+00j
  3.74956705e+03+0.00000000e+00j  3.48369051e+03+0.00000000e+00j
  3.25224107e+03+0.00000000e+00j  2.93166564e+03+0.00000000e+00j
  2.85275602e+03+0.00000000e+00j  2.71829324e+03+0.00000000e+00j
  2.60976635e+03+0.00000000e+00j  2.47550280e+03+0.00000000e+00j
  2.37702804e+03+0.00000000e+00j  2.36803629e+03+0.00000000e+00j
  2.20536627e+03+0.00000000e+00j  2.10394938e+03+0.00000000e+00j
  2.00475654e+03+0.00000000e+00j  1.92347537e+03+0.00000000e+00j
  1.85443490e+03+0.00000000e+00j  1.75281073e+03+0.00000000e+00j
  1.70391195e+03+0.00000000e+00j  1.67439301e+03+0.00000000e+00j
  1.51795940e+03+0.00000000e+00j  1.49528251e+03+0.00000000e+00j
  1.48290977e+03+0.00000000e+00j  1.39006328e+03+0.00000000e+00j
  1.35638712e+03+0.00000000e+00j  1.28692401e+03+0.00000000e+00j
  1.24314065e+03+0.00000000e+00j  1.21186872e+03+0.00000000e+00j
  1.19895832e+03+0.00000000e+00j  1.15329603e+03+0.00000000e+00j
  1.12046714e+03+0.00000000e+00j  1.09067300e+03+0.00000000e+00j
  1.08200895e+03+0.00000000e+00j  1.02479205e+03+0.00000000e+00j
  9.81849313e+02+0.00000000e+00j  9.47135772e+02+0.00000000e+00j
  9.33572765e+02+0.00000000e+00j  8.95169810e+02+0.00000000e+00j
  8.66933404e+02+0.00000000e+00j  8.54348029e+02+0.00000000e+00j
  8.11185816e+02+0.00000000e+00j  7.96928901e+02+0.00000000e+00j
  7.76806085e+02+0.00000000e+00j  7.63753537e+02+0.00000000e+00j
  7.38211677e+02+0.00000000e+00j  7.31949513e+02+0.00000000e+00j
  7.08624297e+02+0.00000000e+00j  6.82084969e+02+0.00000000e+00j
```

```
 6.76683086e+02+0.00000000e+00j  6.55372324e+02+0.00000000e+00j
 6.41268952e+02+0.00000000e+00j  6.23450391e+02+0.00000000e+00j
 6.11425604e+02+0.00000000e+00j  5.99004914e+02+0.00000000e+00j
 5.90850677e+02+0.00000000e+00j  5.78674366e+02+0.00000000e+00j
 5.69111023e+02+0.00000000e+00j  5.13850518e+02+0.00000000e+00j
 5.44804293e+02+0.00000000e+00j  5.35006898e+02+0.00000000e+00j
 5.30486759e+02+0.00000000e+00j  5.08242320e+02+0.00000000e+00j
 4.81908824e+02+0.00000000e+00j  4.70667261e+02+0.00000000e+00j
 4.56176563e+02+0.00000000e+00j  4.50664691e+02+0.00000000e+00j
 4.44423927e+02+0.00000000e+00j  4.38580087e+02+0.00000000e+00j
 4.23422978e+02+0.00000000e+00j  4.11951508e+02+0.00000000e+00j
 4.01002810e+02+0.00000000e+00j  3.88939943e+02+0.00000000e+00j
 3.85961276e+02+0.00000000e+00j  3.75194182e+02+0.00000000e+00j
 3.73221426e+02+0.00000000e+00j  3.68693846e+02+0.00000000e+00j
 3.46871826e+02+0.00000000e+00j  3.51879246e+02+0.00000000e+00j
 3.54609661e+02+0.00000000e+00j  3.40874811e+02+0.00000000e+00j
 3.35212882e+02+0.00000000e+00j  3.23141987e+02+0.00000000e+00j
 3.22369191e+02+0.00000000e+00j  3.18593626e+02+0.00000000e+00j
 3.13503509e+02+0.00000000e+00j  3.02331984e+02+0.00000000e+00j
 2.98369488e+02+0.00000000e+00j  2.96731086e+02+0.00000000e+00j
 2.89536860e+02+0.00000000e+00j  2.86528583e+02+0.00000000e+00j
 2.78226048e+02+0.00000000e+00j  2.74090409e+02+0.00000000e+00j
 2.48530794e+02+0.00000000e+00j  2.65160639e+02+0.00000000e+00j
 2.61027698e+02+0.00000000e+00j  2.53603220e+02+0.00000000e+00j
 2.55640964e+02+0.00000000e+00j  2.43120946e+02+0.00000000e+00j
 2.41519448e+02+0.00000000e+00j  2.35626573e+02+0.00000000e+00j
```

```
print("Eigenvectors:")
print(eigenvectors)
```

```
Eigenvectors:
[[ 0.01647755+0.j          0.01138386+0.j          0.1016398 +0.j
  ...  0.00089551+0.01610465j  0.00233221+0.j
  -0.00701227+0.j        ]
 [ 0.02095732+0.j          0.01099511+0.j          0.11954035+0.j
  ... -0.02107416+0.00833094j -0.00876134+0.j
  -0.01133951+0.j        ]
 [ 0.03128628+0.j          0.02936026+0.j          0.11911552+0.j
  ...  0.00588385+0.01223678j -0.03307016+0.j
  -0.03445938+0.j        ]
 ...
 [ 0.07043065+0.j         -0.06396725+0.j         -0.0564373 +0.j
  ... -0.03223489+0.07581847j -0.03453052+0.j
  -0.06637296+0.j        ]
 [ 0.07278706+0.j         -0.05577653+0.j         -0.05239871+0.j
  ... -0.17163374-0.j         -0.15567568+0.j
  -0.16344308+0.j        ]
 [ 0.07435679+0.j         -0.05589176+0.j         -0.04344141+0.j
  ...  0.07840289-0.04679017j  0.13870885+0.j
   0.16856888+0.j        ]]
```

```
# Sort the eigenvalues and their corresponding eigenvectors in descending order
sorted_indices = np.argsort(eigenvalues)[::-1]
sorted_eigenvalues = eigenvalues[sorted_indices]
sorted_eigenvectors = eigenvectors[:, sorted_indices]

print("Sorted Eigenvalues:")
print(sorted_eigenvalues)
```

```
Sorted Eigenvalues:
[ 5.63294113e+04+0.00000000e+00j  3.66340016e+04+0.00000000e+00j
  2.93994813e+04+0.00000000e+00j  2.28486224e+04+0.00000000e+00j
  1.73508832e+04+0.00000000e+00j  1.60313480e+04+0.00000000e+00j
  1.31732244e+04+0.00000000e+00j  1.05589938e+04+0.00000000e+00j
  9.19039668e+03+0.00000000e+00j  8.71147108e+03+0.00000000e+00j
  6.74188435e+03+0.00000000e+00j  6.67937330e+03+0.00000000e+00j
  5.34706126e+03+0.00000000e+00j  4.76454528e+03+0.00000000e+00j
  4.48727389e+03+0.00000000e+00j  4.21526060e+03+0.00000000e+00j
  4.05755706e+03+0.00000000e+00j  3.82091642e+03+0.00000000e+00j
  3.74956705e+03+0.00000000e+00j  3.48369051e+03+0.00000000e+00j
  3.25224107e+03+0.00000000e+00j  2.93166564e+03+0.00000000e+00j
  2.85275602e+03+0.00000000e+00j  2.71829324e+03+0.00000000e+00j
  2.60976635e+03+0.00000000e+00j  2.47550280e+03+0.00000000e+00j
  2.37702804e+03+0.00000000e+00j  2.36803629e+03+0.00000000e+00j
  2.20536627e+03+0.00000000e+00j  2.10394938e+03+0.00000000e+00j
  2.00475654e+03+0.00000000e+00j  1.92347537e+03+0.00000000e+00j
  1.85443490e+03+0.00000000e+00j  1.75281073e+03+0.00000000e+00j
  1.70391195e+03+0.00000000e+00j  1.67439301e+03+0.00000000e+00j
  1.51795940e+03+0.00000000e+00j  1.49528251e+03+0.00000000e+00j
  1.48290977e+03+0.00000000e+00j  1.39006328e+03+0.00000000e+00j
  1.35638712e+03+0.00000000e+00j  1.28692401e+03+0.00000000e+00j
  1.24314065e+03+0.00000000e+00j  1.21186872e+03+0.00000000e+00j
  1.19895832e+03+0.00000000e+00j  1.15329603e+03+0.00000000e+00j
  1.12046714e+03+0.00000000e+00j  1.09067300e+03+0.00000000e+00j
  1.08200895e+03+0.00000000e+00j  1.02479205e+03+0.00000000e+00j
  9.81849313e+02+0.00000000e+00j  9.47135772e+02+0.00000000e+00j
  9.33572765e+02+0.00000000e+00j  8.95169810e+02+0.00000000e+00j
  8.66933404e+02+0.00000000e+00j  8.54348029e+02+0.00000000e+00j
  8.11185816e+02+0.00000000e+00j  7.96928901e+02+0.00000000e+00j
  7.76806085e+02+0.00000000e+00j  7.63753537e+02+0.00000000e+00j
  7.38211677e+02+0.00000000e+00j  7.31949513e+02+0.00000000e+00j
  7.08624297e+02+0.00000000e+00j  6.82084969e+02+0.00000000e+00j
  6.76683086e+02+0.00000000e+00j  6.55372324e+02+0.00000000e+00j
  6.41268952e+02+0.00000000e+00j  6.23450391e+02+0.00000000e+00j
```

```
 6.11425604e+02+0.00000000e+00j  5.99004914e+02+0.00000000e+00j
 5.90850677e+02+0.00000000e+00j  5.78674366e+02+0.00000000e+00j
 5.69111023e+02+0.00000000e+00j  5.44804293e+02+0.00000000e+00j
 5.35006898e+02+0.00000000e+00j  5.30486759e+02+0.00000000e+00j
 5.13850518e+02+0.00000000e+00j  5.08242320e+02+0.00000000e+00j
 4.81908824e+02+0.00000000e+00j  4.70667261e+02+0.00000000e+00j
 4.56176563e+02+0.00000000e+00j  4.50664691e+02+0.00000000e+00j
 4.44423927e+02+0.00000000e+00j  4.38580087e+02+0.00000000e+00j
 4.23422978e+02+0.00000000e+00j  4.11951508e+02+0.00000000e+00j
 4.01002810e+02+0.00000000e+00j  3.88939943e+02+0.00000000e+00j
 3.85961276e+02+0.00000000e+00j  3.75194182e+02+0.00000000e+00j
 3.73221426e+02+0.00000000e+00j  3.68693846e+02+0.00000000e+00j
 3.54609661e+02+0.00000000e+00j  3.51879246e+02+0.00000000e+00j
 3.46871826e+02+0.00000000e+00j  3.40874811e+02+0.00000000e+00j
 3.35212882e+02+0.00000000e+00j  3.23141987e+02+0.00000000e+00j
 3.22369191e+02+0.00000000e+00j  3.18593626e+02+0.00000000e+00j
 3.13503509e+02+0.00000000e+00j  3.02331984e+02+0.00000000e+00j
 2.98369488e+02+0.00000000e+00j  2.96731086e+02+0.00000000e+00j
 2.89536860e+02+0.00000000e+00j  2.86528583e+02+0.00000000e+00j
 2.78226048e+02+0.00000000e+00j  2.74090409e+02+0.00000000e+00j
 2.65160639e+02+0.00000000e+00j  2.61027698e+02+0.00000000e+00j
 2.55640964e+02+0.00000000e+00j  2.53603220e+02+0.00000000e+00j
 2.48530794e+02+0.00000000e+00j  2.43120946e+02+0.00000000e+00j
 2.41519448e+02+0.00000000e+00j  2.35626573e+02+0.00000000e+00j
```

```
print("Sorted Eigenvectors:")
print(sorted_eigenvectors)
```

```
Sorted Eigenvectors:
[[ 0.01647755+0.j          0.01138386+0.j          0.1016398 +0.j
  ...  0.02674217+0.j         -0.02640516+0.04665489j
  -0.02640516-0.04665489j]
 [ 0.02095732+0.j          0.01099511+0.j          0.11954035+0.j
  ... -0.07168458+0.j          0.07353378-0.08180094j
   0.07353378+0.08180094j]
 [ 0.03128628+0.j          0.02936026+0.j          0.11911552+0.j
  ...  0.06650938+0.j         -0.01249773+0.06120674j
  -0.01249773-0.06120674j]
 ...
 [ 0.07043065+0.j         -0.06396725+0.j         -0.0564373 +0.j
  ...  0.07796677+0.j         -0.10147576+0.01907698j
  -0.10147576-0.01907698j]
 [ 0.07278706+0.j         -0.05577653+0.j         -0.05239871+0.j
  ... -0.07760832+0.j          0.02276461+0.01251115j
   0.02276461-0.01251115j]
 [ 0.07435679+0.j         -0.05589176+0.j         -0.04344141+0.j
  ...  0.02912684+0.j          0.00249767+0.04854677j
   0.00249767-0.04854677j]]
```

```
centered_X.shape
#sorted_eigenvectors.shape
```

```
(264, 300)
```

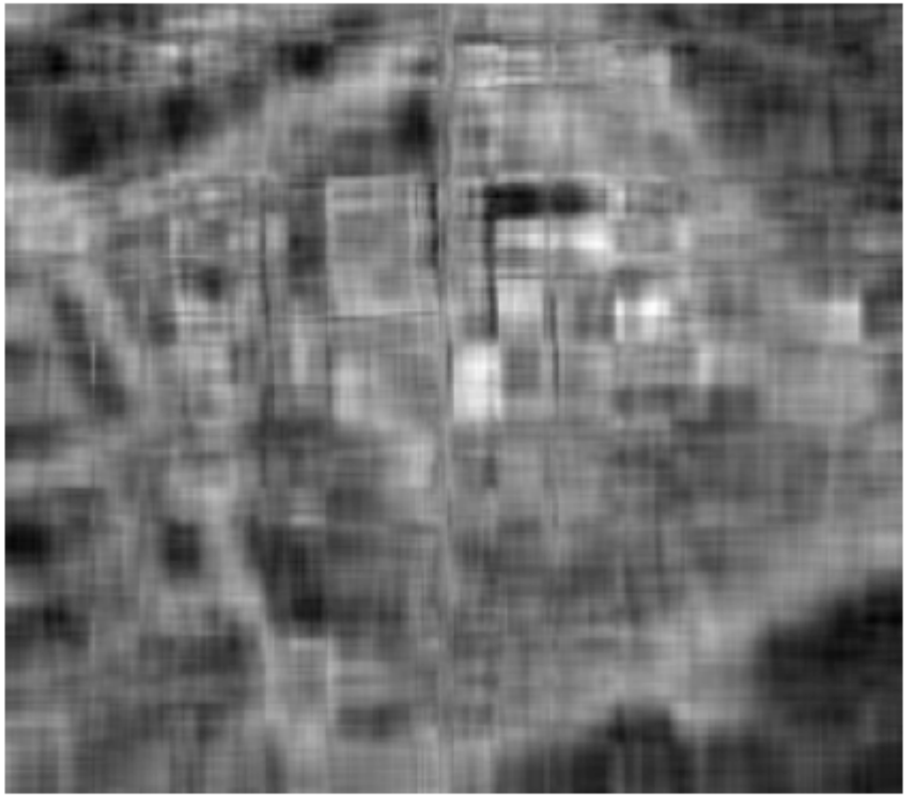
```
# Define the number of principal components
```

```python
num_components_list = [10, 20, 30, 40, 50, 60, 90]
title = "Dimensionality Reduction using PCA."

# Reconstruct the image for each number of components
for num_components in num_components_list:
    # Take the first 'num_components' eigenvectors
    selected_eigenvectors = sorted_eigenvectors[:, :num_components]
    # Project the mean-subtracted image onto the selected eigenvectors
    projected_image = np.dot(centered_X, selected_eigenvectors)

    # Reconstruct the image from the projected data
    reconstructed_image = np.dot(projected_image, selected_eigenvectors.T) + column_means
    explained_variance = np.sum(sorted_eigenvalues[:num_components]) / np.sum(sorted_eigenvalues)
    explained_variance = explained_variance.real
    # Ensure the data type is compatible
    reconstructed_image = np.clip(reconstructed_image, 0, 255)
    reconstructed_image = reconstructed_image.astype(np.uint8)

    # Visualize
    plt.imshow(reconstructed_image, cmap='gray')
    plt.axis('off')
    plt.title(f'{title}\nNumber of Components: {num_components}\nExplained Variance: {explained_variance:.2%}')
    plt.axis('off')
    plt.show()
    plt.show()
```

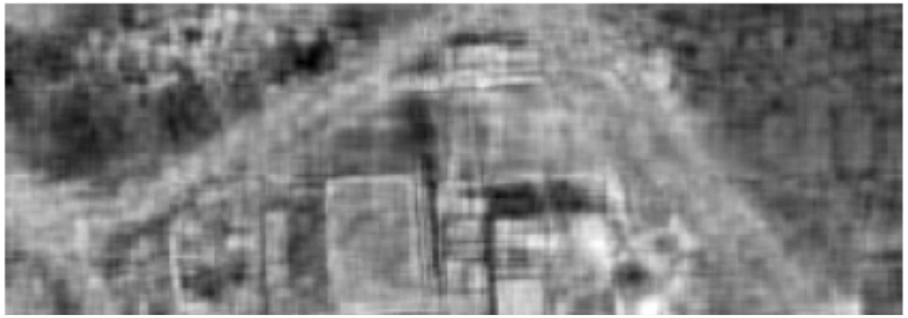
```
C:\Users\navde\AppData\Local\Temp\ipykernel_22020\3827863460.py:19: ComplexWarning: Casting complex values to re
  reconstructed_image = reconstructed_image.astype(np.uint8)
```

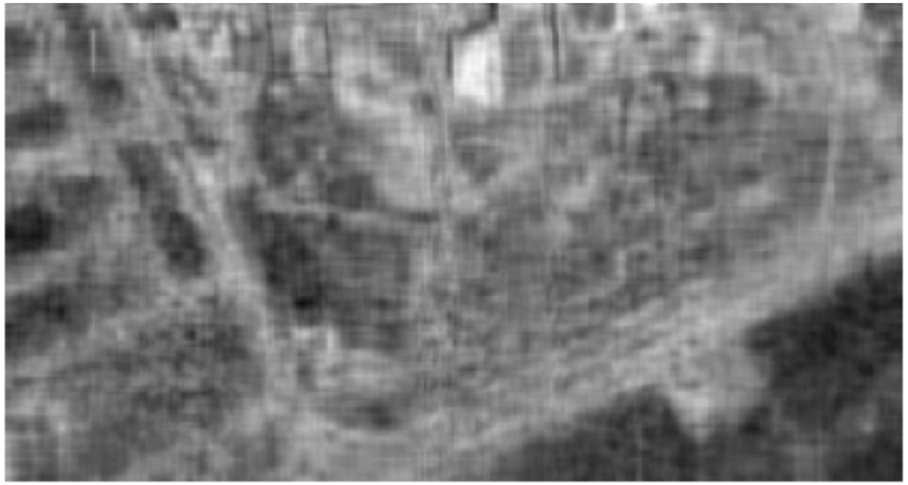
Dimensionality Reduction using PCA.
Number of Components: 10
Explained Variance: 60.29%

Explained Variance: [illegible]%



Dimensionality Reduction using PCA.
Number of Components: 20
Explained Variance: 73.26%





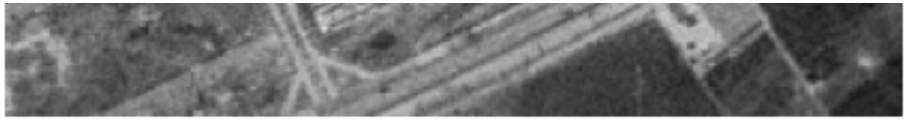
Dimensionality Reduction using PCA.
Number of Components: 30
Explained Variance: 80.35%

Dimensionality Reduction using PCA.
Number of Components: 40
Explained Variance: 84.95%

Dimensionality Reduction using PCA.
Number of Components: 50
Explained Variance: 88.17%

Dimensionality Reduction using PCA.
Number of Components: 60
Explained Variance: 90.53%

Dimensionality Reduction using PCA.
Number of Components: 90
Explained Variance: 94.96%

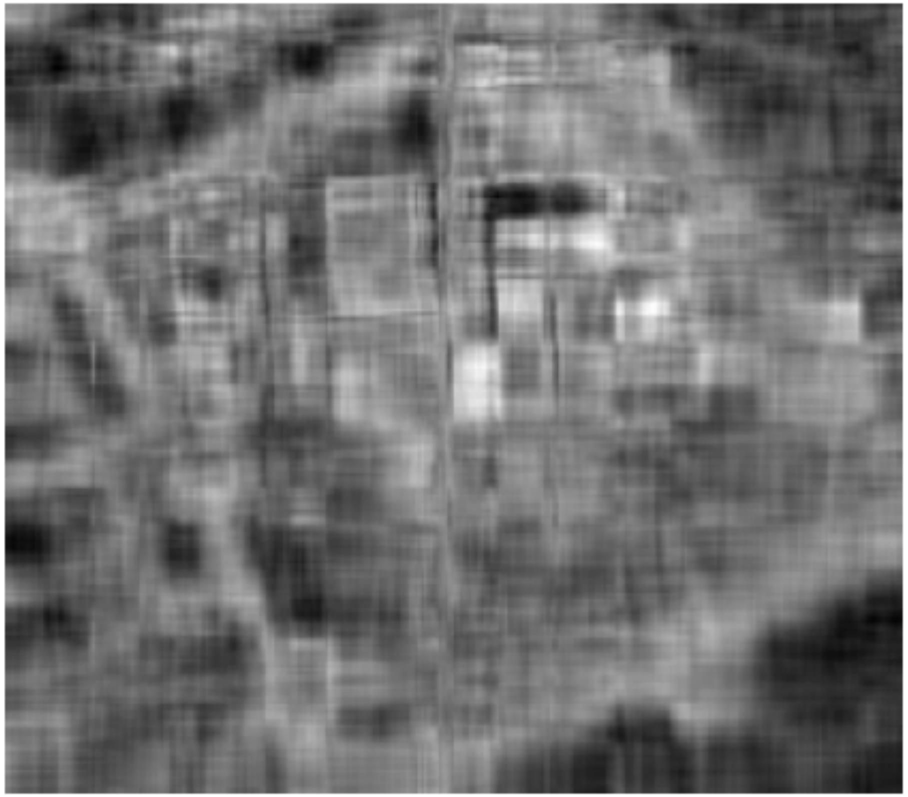


```
# Reconstruct the image for each number of components

# Initialize Output_images list
Output_images = []
for num_components in num_components_list:
    # Take the first 'num_components' eigenvectors
    selected_eigenvectors = sorted_eigenvectors[:, :num_components]
    # Project the mean-subtracted image onto the selected eigenvectors
    projected_image = np.dot(centered_X, selected_eigenvectors)

    # Reconstruct the image from the projected data
    reconstructed_image = np.dot(projected_image, selected_eigenvectors.T) + column_means

    # Append the reconstructed image to Output_images list
    Output_images.append(reconstructed_image)

    explained_variance = np.sum(sorted_eigenvalues[:num_components]) / np.sum(sorted_eigenvalues)
    explained_variance = explained_variance.real

    # Ensure the data type is compatible
    reconstructed_image = np.clip(reconstructed_image, 0, 255)
    reconstructed_image = reconstructed_image.astype(np.uint8)

    # Visualize (optional)
    plt.imshow(reconstructed_image, cmap='gray')
    plt.axis('off')
    plt.title(f'{title}\nNumber of Components: {num_components}\nExplained Variance: {explained_variance:.2%}')
```

```
plt.show()
```

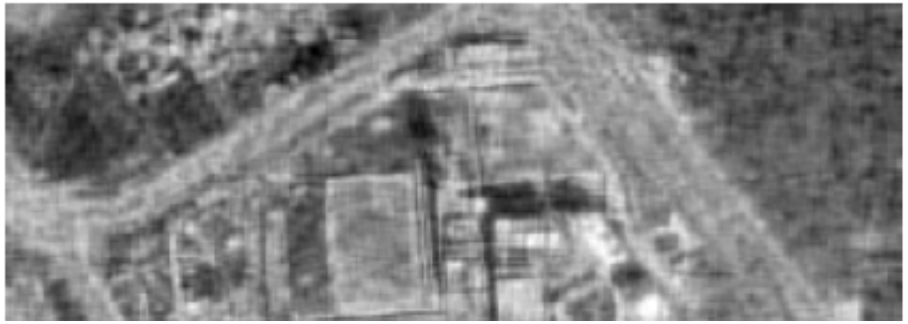
```
C:\Users\navde\AppData\Local\Temp\ipykernel_22020\1737686582.py:23: ComplexWarning: Casting complex values to re
  reconstructed_image = reconstructed_image.astype(np.uint8)
```

Dimensionality Reduction using PCA.
Number of Components: 10
Explained Variance: 60.29%



Dimensionality Reduction using PCA.
Number of Components: 20
Explained Variance: 73.26%

Dimensionality Reduction using PCA.
Number of Components: 30
Explained Variance: 80.35%



Dimensionality Reduction using PCA.
Number of Components: 40
Explained Variance: 84.95%

Dimensionality Reduction using PCA.
Number of Components: 50
Explained Variance: 88.17%

Dimensionality Reduction using PCA.
Number of Components: 60
Explained Variance: 90.53%

Dimensionality Reduction using PCA.
Number of Components: 90
Explained Variance: 94.96%

```
from sklearn.decomposition import PCA

# Compute minimum num_components needed to explain 95% variance
total_variance = np.sum(sorted_eigenvalues)
cumulative_variance = np.cumsum(sorted_eigenvalues) / total_variance
min_components_95_variance = np.argmax(cumulative_variance >= 0.95) + 1  # Add 1 to get the count of components

print(f"Number of components for 95% variance: {min_components_95_variance}")

# Step 13: Use PCA function from sklearn to compute num_components for 95% data variance
pca = PCA(n_components=0.95)
pca.fit(float64_image)  # Assuming 'data' is your input data
num_components_95_variance_sklearn = pca.n_components_
```

```
Number of components for 95% variance: 91
```