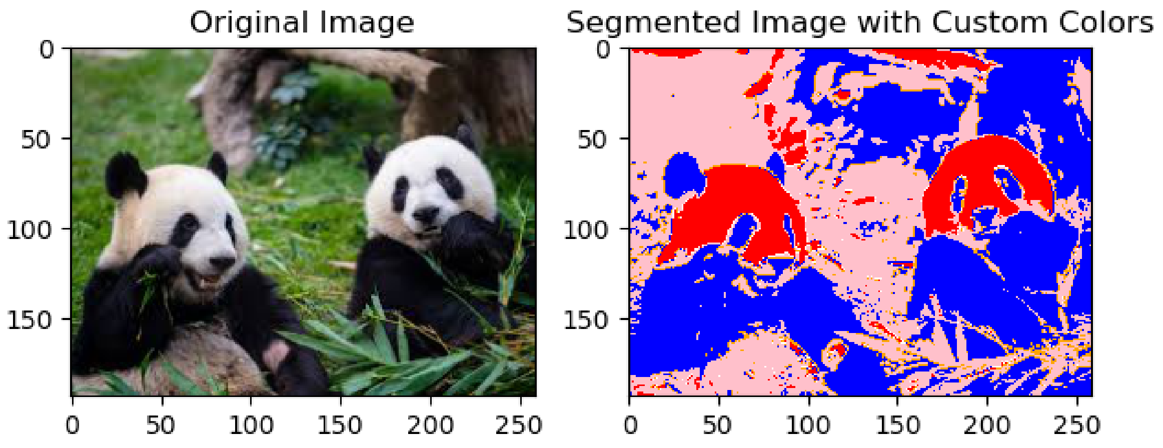
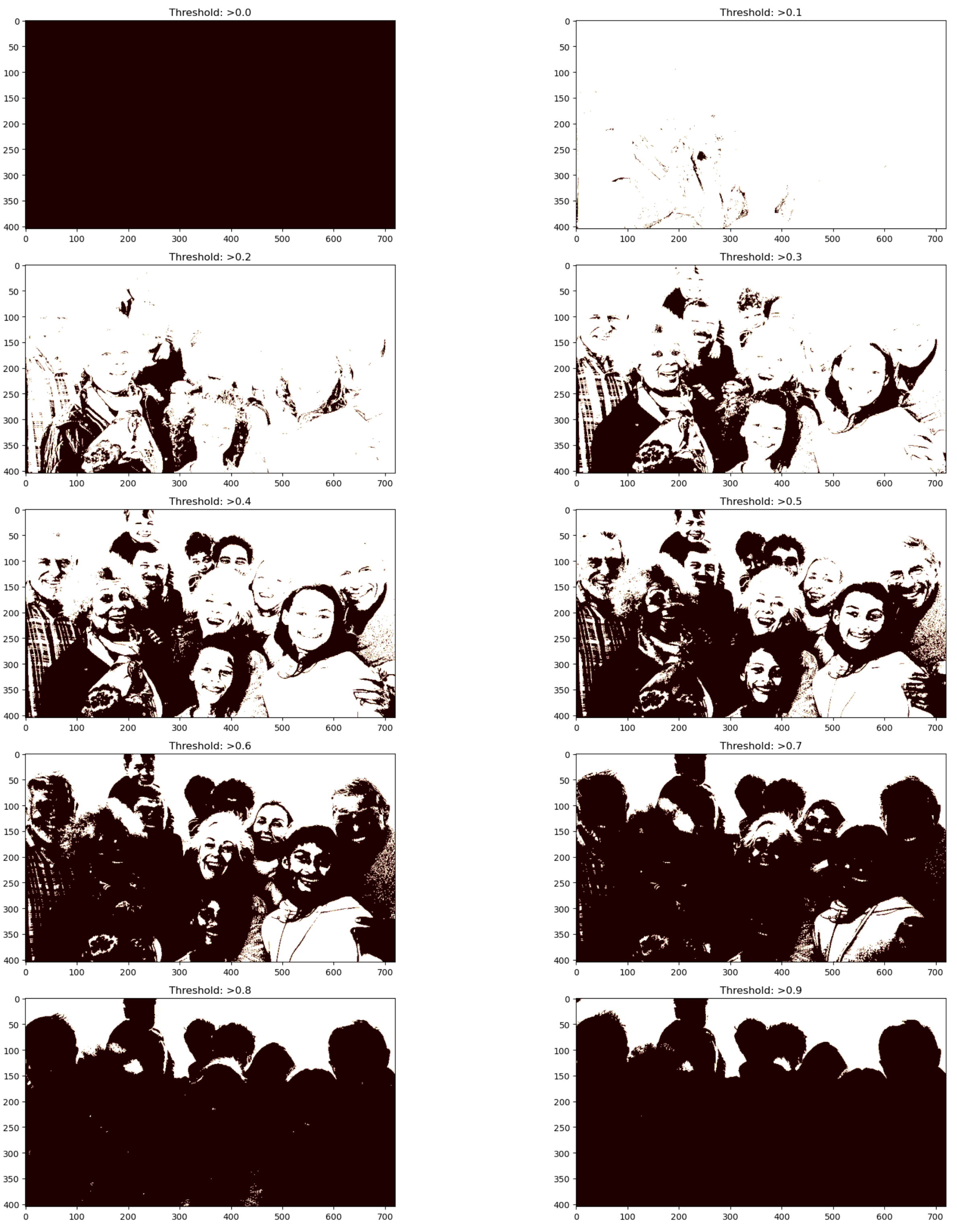
In [7]:
```python
from skimage import io

from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
# Load the image
image_path = r"C:/Users/Admin/Desktop/pandas.jfif"
image = io.imread(image_path)
# Flatten the image to create a 2D array of RGB values
height, width, channels = image.shape
image_2d = image.reshape((height * width, channels))
# Specify the number of clusters (segments)
num_clusters = 3
# Apply k-means clustering
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(image_2d)
labels = kmeans.labels_
# Reshape the labels to the shape of the original image
segmented_image = labels.reshape((height, width))
# Custom colormap
colors = ['red', 'white', 'blue', 'orange','pink']
custom_cmap = ListedColormap(colors)
# Display the original and segmented images
plt.figure(figsize=(15, 8))
plt.subplot(1, 4, 1)
plt.title("Original Image")
plt.imshow(image)
plt.axis("on")
plt.subplot(1, 4, 2)
plt.title("Segmented Image with Custom Colors")
plt.imshow(segmented_image, cmap=custom_cmap)
plt.axis("on")
plt.show()
```

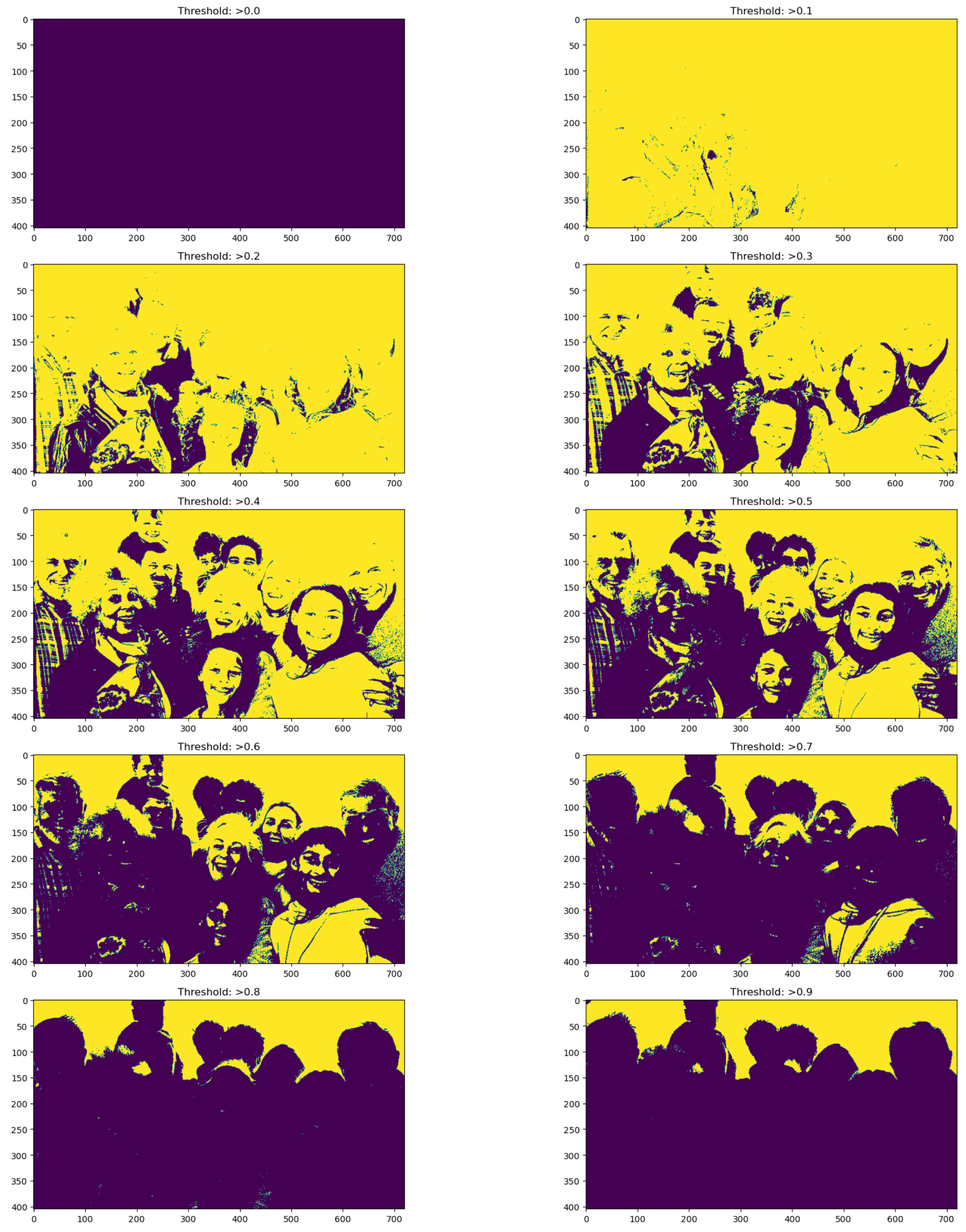
```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: F
utureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [9]:
```
from skimage import data
from skimage import filters
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
from skimage import io
# Sample Image of scikit-image package
image = io.imread(r"C:/Users/Admin/Desktop/human_image.jpg")
plt.imshow(image)
gray_image = rgb2gray(image)
# Setting the plot size to 15,15
plt.figure(figsize=(20, 20))
for i in range(0,10):
# Iterating different thresholds
    a = (gray_image > i*0.1)*1
    plt.subplot(5,2,i+1)
# Rounding of the threshold
# value to 1 decimal point
    plt.title("Threshold: >"+str(round(i*0.1,1)))
# Displaying the binarized image
# of various thresholds
    plt.imshow(a,cmap = 'pink' )
plt.tight_layout()
```

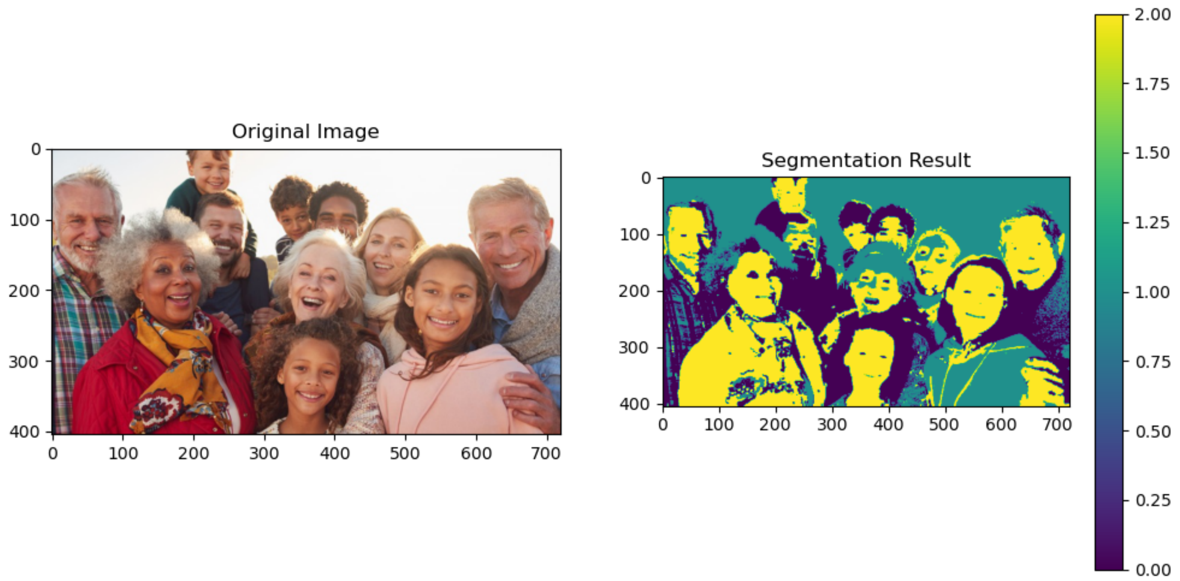

Threshold: >0.0
Threshold: >0.1
Threshold: >0.2
Threshold: >0.3
Threshold: >0.4
Threshold: >0.5
Threshold: >0.6
Threshold: >0.7
Threshold: >0.8
Threshold: >0.9
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700

In [17]:
```
from skimage import data
from skimage import filters
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
from skimage import io
import numpy as np

# Sample Image of scikit-image package
image = io.imread(r"C:/Users/Admin/Desktop/human_image.jpg")
plt.imshow(image)
gray_image = rgb2gray(image)

# Setting the plot size to 15,15
plt.figure(figsize=(20, 20))

# Define a list of color maps for each iteration
cmaps = ['viridis']

for i in range(0, 10):
    # Iterating different thresholds
    thresholded_image = (gray_image > i * 0.1) * 1
    plt.subplot(5, 2, i + 1)

    # Rounding of the threshold value to 1 decimal point
    plt.title("Threshold: >" + str(round(i * 0.1, 1)))

    # Displaying the binarized image of various thresholds using different col
    plt.imshow(thresholded_image, cmap=cmaps[i % len(cmaps)])

plt.tight_layout()
plt.show()
```

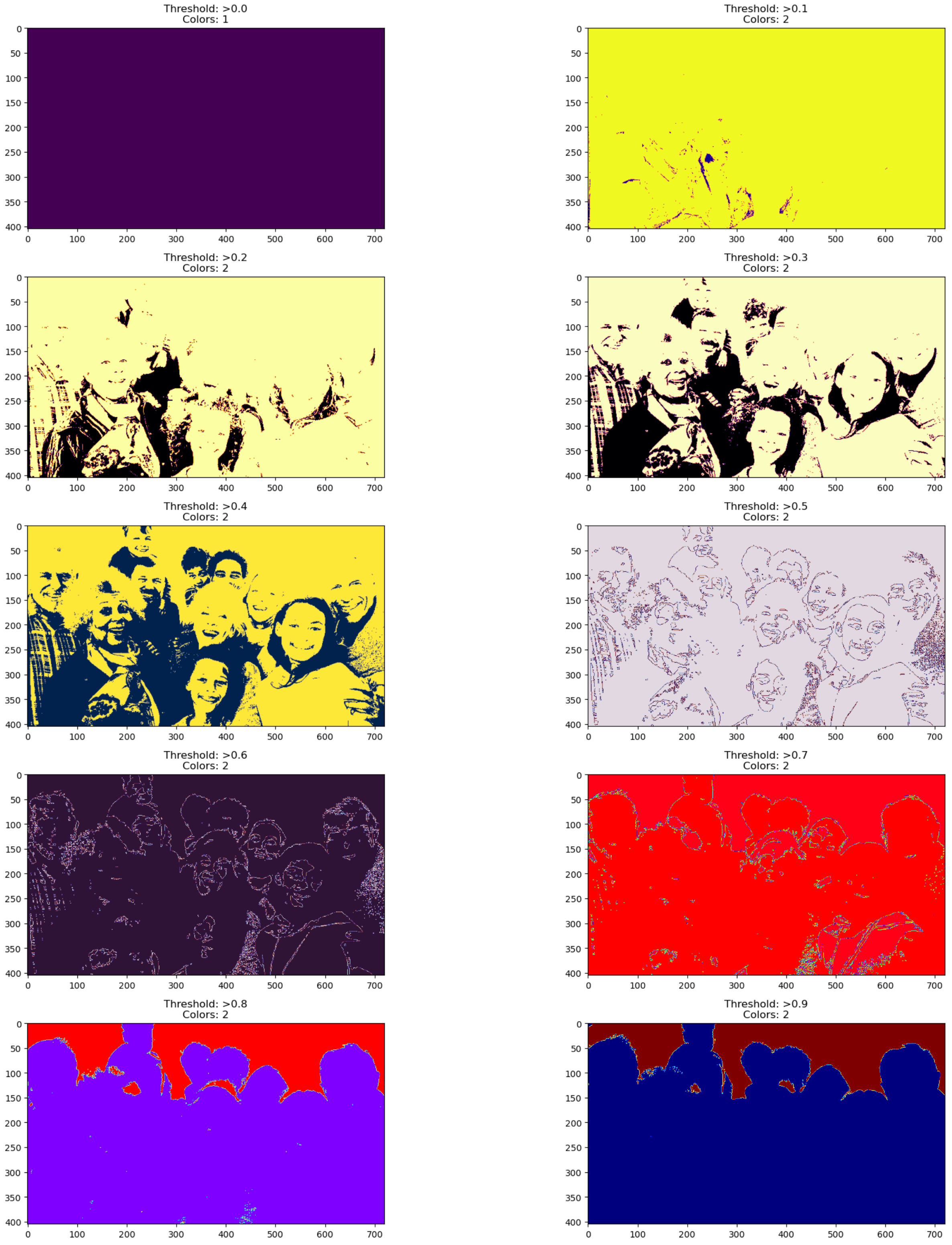

Threshold: >0.0
Threshold: >0.1
Threshold: >0.2
Threshold: >0.3
Threshold: >0.4
Threshold: >0.5
Threshold: >0.6
Threshold: >0.7
Threshold: >0.8
Threshold: >0.9
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700

In [11]:
```
import numpy as np
from skimage import io, color
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.preprocessing import StandardScaler

# Load the image
image = io.imread(r"C:/Users/Admin/Desktop/human_image.jpg")

# Convert the image to the LAB color space
lab_image = color.rgb2lab(image)

# Flatten the image to a 2D array of pixels
pixels = lab_image.reshape((-1, 3))

# Normalize pixel values
pixels_normalized = StandardScaler().fit_transform(pixels)

# Perform k-means clustering on the labeled data
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pixels_normalized)

# Reshape the labels back to the shape of the original image
segmentation = labels.reshape(lab_image.shape[:2])

# Display the original image and segmentation result
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(image)

plt.subplot(1, 2, 2)
plt.title("Segmentation Result")
plt.imshow(segmentation, cmap='viridis')
plt.colorbar()

plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: F
utureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Original Image
Segmentation Result
0
100
200
300
400
500
600
700
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [19]:
```python
from skimage import data
from skimage.color import rgb2gray
import matplotlib.pyplot as plt
from skimage import io
import numpy as np

# Sample Image of scikit-image package
image = io.imread(r"C:/Users/Admin/Desktop/human_image.jpg")
plt.imshow(image)
gray_image = rgb2gray(image)

# Setting the plot size to 15,15
plt.figure(figsize=(20, 20))

# Define a list of color maps for each iteration
cmaps = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', 'twilight', 'twil

for i in range(0, 10):
    # Iterating different thresholds
    thresholded_image = (gray_image > i * 0.1) * 1
    unique_colors = np.unique(thresholded_image)

    plt.subplot(5, 2, i + 1)

    # Rounding of the threshold value to 1 decimal point
    plt.title("Threshold: >" + str(round(i * 0.1, 1)) + f"\nColors: {len(uniqu

    # Displaying the binarized image of various thresholds using different col
    plt.imshow(thresholded_image, cmap=cmaps[i % len(cmaps)])

plt.tight_layout()
plt.show()
```


Threshold: >0.0
Colors: 1
Threshold: >0.1
Colors: 2
Threshold: >0.2
Colors: 2
Threshold: >0.3
Colors: 2
Threshold: >0.4
Colors: 2
Threshold: >0.5
Colors: 2
Threshold: >0.6
Colors: 2
Threshold: >0.7
Colors: 2
Threshold: >0.8
Colors: 2
Threshold: >0.9
Colors: 2

In [29]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Directory containing your dataset (update with your path)
dataset_directory = "C:/Users/Admin/Desktop/images"

# Image dimensions and other parameters
img_width, img_height = 224, 224
batch_size = 32
epochs = 10
num_classes = len(os.listdir(dataset_directory))

# Data augmentation for better generalization
datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

# Load and preprocess images using the generator
train_generator = datagen.flow_from_directory(
    dataset_directory,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    dataset_directory,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Build a simple CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(img_width, img_height, 3), activatio
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc

# Train the model
model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)

# Iterate over a few validation images and perform segmentation
for i in range(5):
    img, label = validation_generator.next()
    prediction = model.predict(img)
    predicted_class = np.argmax(prediction[0])

    # Display the original image, ground truth, and predicted segmentation
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 3, 1)
    plt.title("Original Image")
    plt.imshow(img[0])

    plt.subplot(1, 3, 2)
    plt.title("Ground Truth")
    plt.imshow(label[0][:, :, 0], cmap='viridis')

    plt.subplot(1, 3, 3)
    plt.title("Predicted Segmentation")
    plt.imshow(prediction[0][:, :, 0], cmap='viridis')

    plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[29], line 6
      4 from sklearn.model_selection import train_test_split
      5 from sklearn.preprocessing import LabelEncoder
----> 6 from tensorflow.keras.preprocessing.image import ImageDataGenerator
      7 from tensorflow.keras.models import Sequential
      8 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, De
nse

ModuleNotFoundError: No module named 'tensorflow'
```

In [31]:
```
pip install tensorflow
```

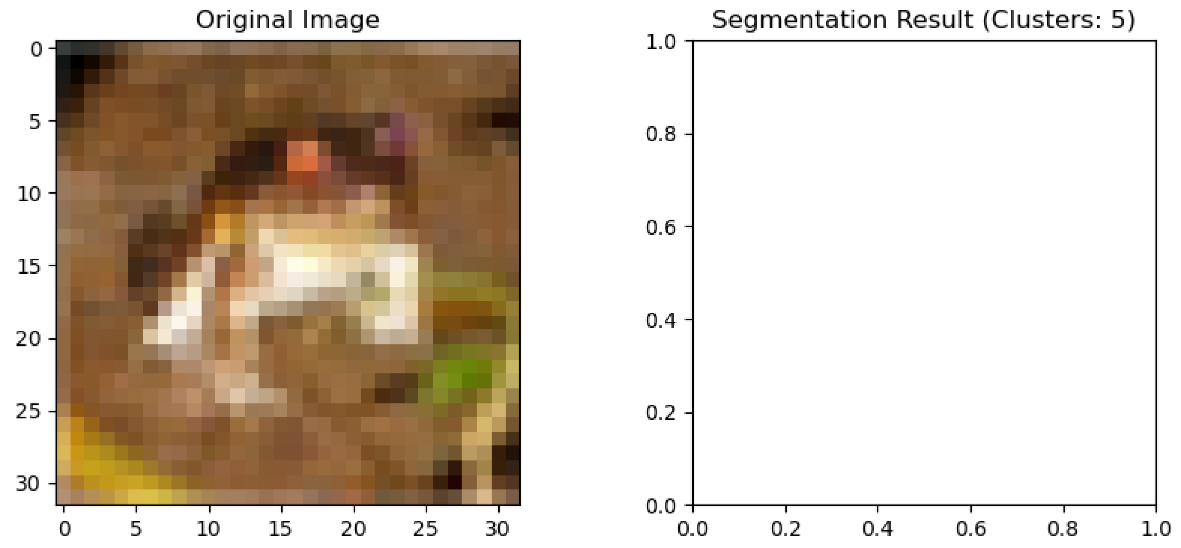
```
Collecting libclang>=13.0.0 (from tensorflow-intel==2.15.0->tensorflow)
  Obtaining dependency information for libclang>=13.0.0 from https://file
s.pythonhosted.org/packages/02/8c/dc970bc00867fe290e8c8a7befa1635af716a9eb
dfe3fb9dce0ca4b522ce/libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata
(https://files.pythonhosted.org/packages/02/8c/dc970bc00867fe290e8c8a7befa
1635af716a9ebdfe3fb9dce0ca4b522ce/libclang-16.0.6-py2.py3-none-win_amd64.w
hl.metadata)
  Using cached libclang-16.0.6-py2.py3-none-win_amd64.whl.metadata (5.3 k
B)
Collecting ml-dtypes~=0.2.0 (from tensorflow-intel==2.15.0->tensorflow)
  Obtaining dependency information for ml-dtypes~=0.2.0 from https://file
s.pythonhosted.org/packages/08/89/c727fde1a3d12586e0b8c01abf53754707d76bea
a9987640e70807d4545f/ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl.metadata (h
ttps://files.pythonhosted.org/packages/08/89/c727fde1a3d12586e0b8c01abf537
54707d76beaa9987640e70807d4545f/ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl.
metadata)
  Using cached ml_dtypes-0.2.0-cp311-cp311-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\admin\anac
onda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.24.
3)
```

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from tensorflow.keras.datasets import cifar10

# Load CIFAR-10 dataset
(x_train, _), (_, _) = cifar10.load_data()

# Reshape the images to flatten them
pixels = x_train.reshape((-1, 3))

# Normalize pixel values to [0, 1]
pixels_normalized = pixels / 255.0

# Number of clusters (colors) for segmentation
n_clusters = 5

# Perform iterative segmentation
for i in range(5):
    # Apply K-Means clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(pixels_normalized)

    # Reshape labels to the original image shape
    segmentation_result = labels.reshape(x_train.shape[:-1])

    # Display the original image and segmentation result
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    plt.title("Original Image")
    plt.imshow(x_train[i])

    plt.subplot(1, 2, 2)
    plt.title(f"Segmentation Result (Clusters: {n_clusters})")
    plt.imshow(segmentation_result, cmap='viridis')
    plt.colorbar()

    plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: F
utureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

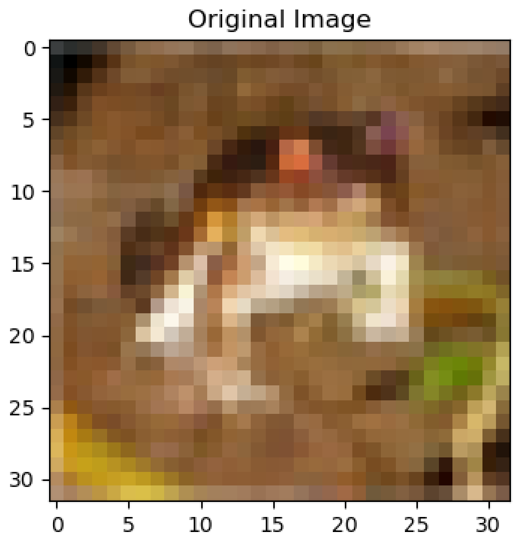
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 36
     34 plt.subplot(1, 2, 2)
     35 plt.title(f"Segmentation Result (Clusters: {n_clusters})")
---> 36 plt.imshow(segmentation_result, cmap='viridis')
     37 plt.colorbar()
     39 plt.show()

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:2695, in imshow(X, cm
ap, norm, aspect, interpolation, alpha, vmin, vmax, origin, extent, interpola
tion_stage, filternorm, filterrad, resample, url, data, **kwargs)
   2689 @_copy_docstring_and_deprecators(Axes.imshow)
   2690 def imshow(
   2691         X, cmap=None, norm=None, *, aspect=None, interpolation=None,
   2692         alpha=None, vmin=None, vmax=None, origin=None, extent=None,
   2693         interpolation_stage=None, filternorm=True, filterrad=4.0,
   2694         resample=None, url=None, data=None, **kwargs):
-> 2695     __ret = gca().imshow(
   2696         X, cmap=cmap, norm=norm, aspect=aspect,
   2697         interpolation=interpolation, alpha=alpha, vmin=vmin,
   2698         vmax=vmax, origin=origin, extent=extent,
   2699         interpolation_stage=interpolation_stage,
   2700         filternorm=filternorm, filterrad=filterrad, resample=resampl
e,
   2701         url=url, **({"data": data} if data is not None else {}),
   2702         **kwargs)
   2703     sci(__ret)
   2704     return __ret

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:1446, in _preproces
s_data.<locals>.inner(ax, data, *args, **kwargs)
   1443 @functools.wraps(func)
   1444 def inner(ax, *args, data=None, **kwargs):
   1445     if data is None:
-> 1446         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1448     bound = new_sig.bind(ax, *args, **kwargs)
   1449     auto_label = (bound.arguments.get(label_namer)
   1450                   or bound.kwargs.get(label_namer))

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:5663, in Axes.ims
how(self, X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin, ex
tent, interpolation_stage, filternorm, filterrad, resample, url, **kwargs)
   5655 self.set_aspect(aspect)
   5656 im = mimage.AxesImage(self, cmap=cmap, norm=norm,
   5657                       interpolation=interpolation, origin=origin,
   5658                       extent=extent, filternorm=filternorm,
   5659                       filterrad=filterrad, resample=resample,
   5660                       interpolation_stage=interpolation_stage,
   5661                       **kwargs)
-> 5663 im.set_data(X)
   5664 im.set_alpha(alpha)
   5665 if im.get_clip_path() is None:
   5666     # image does not already have clipping set, clip to axes patch

File ~\anaconda3\Lib\site-packages\matplotlib\image.py:710, in _ImageBase.set
_data(self, A)
```

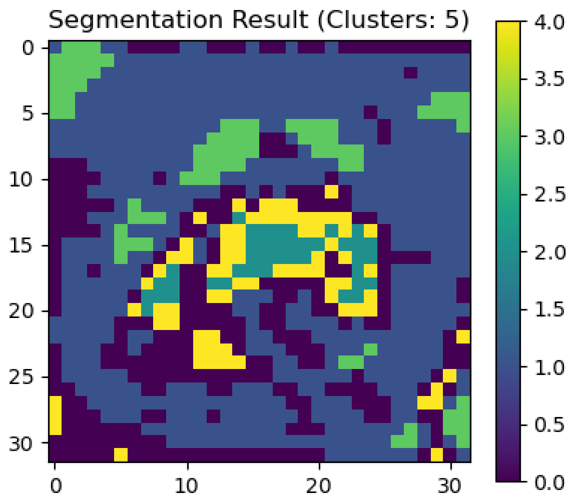
```
    706     self._A = self._A[:, :, 0]
    708 if not (self._A.ndim == 2
    709         or self._A.ndim == 3 and self._A.shape[-1] in [3, 4]):
--> 710     raise TypeError("Invalid shape {} for image data"
    711                     .format(self._A.shape))
    713 if self._A.ndim == 3:
    714     # If the input data has values outside the valid range (after
    715     # normalisation), we issue a warning and then clip X to the bound
s
    716     # - otherwise casting wraps extreme values, hiding outliers and
    717     # making reliable interpretation impossible.
    718     high = 255 if np.issubdtype(self._A.dtype, np.integer) else 1

TypeError: Invalid shape (50000, 32, 32) for image data
```



In [36]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from tensorflow.keras.datasets import cifar10

# Load CIFAR-10 dataset
(x_train, _), (_, _) = cifar10.load_data()

# Reshape the images to flatten them
pixels = x_train.reshape((-1, 3))

# Normalize pixel values to [0, 1]
pixels_normalized = pixels / 255.0

# Number of clusters (colors) for segmentation
n_clusters = 5

# Choose a specific example for visualization
example_index = 0

# Apply K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
labels = kmeans.fit_predict(pixels_normalized)

# Reshape labels to the original image shape
segmentation_result = labels.reshape(x_train.shape[:-1])

# Display the original image and segmentation result
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(x_train[example_index])

plt.subplot(1, 2, 2)
plt.title(f"Segmentation Result (Clusters: {n_clusters})")
plt.imshow(segmentation_result[example_index], cmap='viridis')
plt.colorbar()

plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: F
utureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Original Image
0
5
10
15
20
25
30


Segmentation Result (Clusters: 5)
0
10
20
30
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

```
In [ ]:
```