Original image:

Final image:

Code:

```
wiki_img_url =
"https://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/Ray_and_Maria_St
ata_Center_%28MIT%29.JPG/230px-Ray_and_Maria_Stata_Center_%28MIT%29.JPG"
wiki_img = download_img(wiki_img_url)
wiki_img_arr = img2arr(wiki_img)
display(wiki_img)

wiki_img_colors = wiki_img_arr.reshape((-1, 3))

cache = {} # for reruns

def find_silhouette_score(n_clusters, data):
  if n_clusters not in cache.keys():
      k_means = KMeans(n_clusters=n_clusters)
      k_means.fit(data)
      cache[n_clusters] = silhouette_score(data, k_means.labels_)
  return cache[n_clusters]

x_arr = np.arange(2, 15)
scores = [find_silhouette_score(x, wiki_img_colors) for x in x_arr]

## Find appropriate number of clusters
plt.plot(x_arr, scores)
plt.xlabel("Clusters")
plt.ylabel("Score")
plt.show()

## select cluster number with highest score
index_max_score = np.argmax(scores)
n_clusters_max_score = x_arr[index_max_score]
```

```
kmeans = KMeans(n_clusters=n_clusters_max_score)
kmeans.fit(wiki_img_colors)

## plot result
rg_chroma_plot(wiki_img_arr, kmeans.cluster_centers_)

## display result
replaced = replace_nearest_color(wiki_img_arr, kmeans.cluster_centers_)
display(arr2img(replaced))
```

I've used the silhouette score to determine an appropriate number of clusters. Which in my case seemed to be three clusters, after observing the plotted the scores.