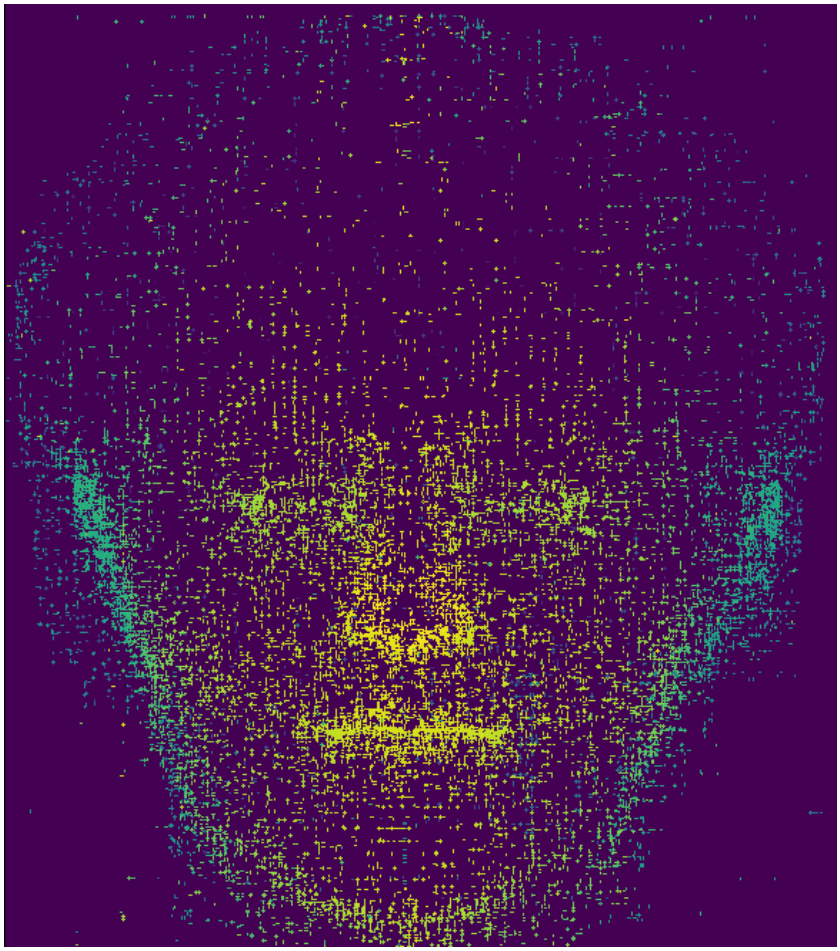
Jared Curtis
CS 491

January 18th, 2022

- Met with Dr. Akbas and the team to discuss semester goals for the project.
- Decided to use machine learning to map the applicable significant landmarks on a patient's face.

January 23rd, 2022

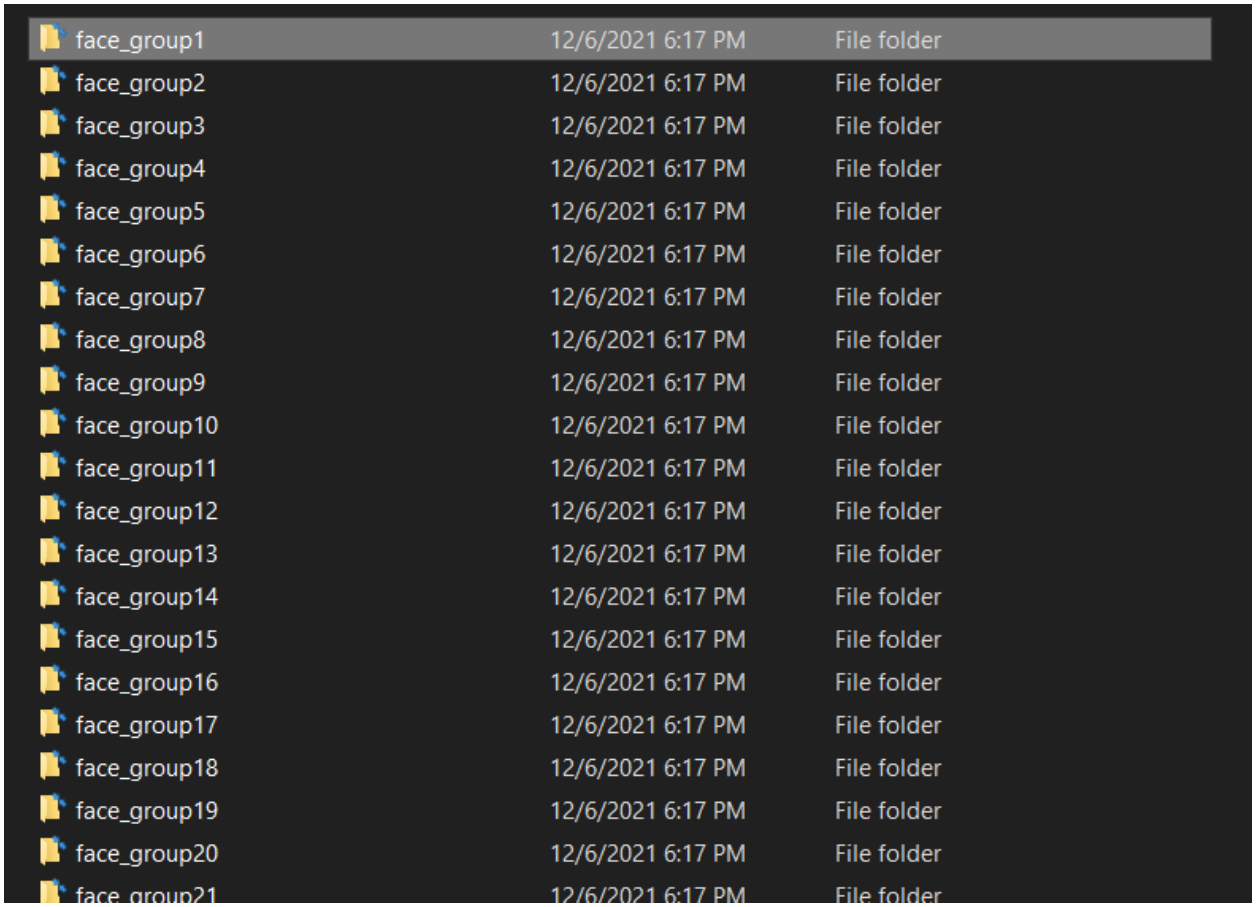
- Researched possible solutions for automatically mapping facial landmarks.
- Found plenty of resources for mapping landmarks on 2D images, but very few resources for 3D objects.
- Wrote a python script to convert a 3D .obj file into a depth map of the face elements.

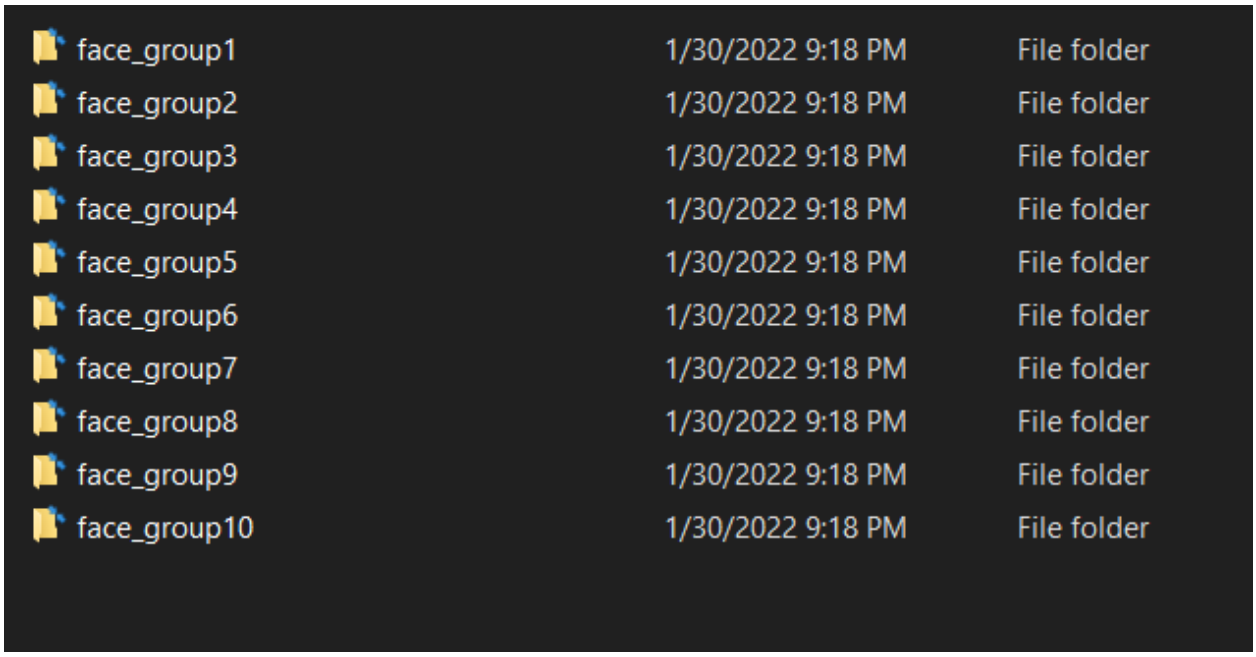


```
pixel_graph = np.full((2500, 2500), 0)
blur_scale = 5

for coordinate in test_array:
    pixel_graph[int(coordinate[0]), int(coordinate[1])] = coordinate[2]
    for i in range(blur_scale):
        pixel_graph[int(coordinate[0]) + i, int(coordinate[1])] = coordinate[2]
        pixel_graph[int(coordinate[0]) - i, int(coordinate[1])] = coordinate[2]
        pixel_graph[int(coordinate[0]), int(coordinate[1]) + i] = coordinate[2]
        pixel_graph[int(coordinate[0]), int(coordinate[1]) - i] = coordinate[2]

plt.figure(figsize=(12,12), dpi=80)
plt.imshow(pixel_graph, interpolation='nearest')
plt.show()
```

January 30th, 2022

- Wrote a python script to create clusters of similar looking faces.
- Created groups based on each patient's 10 most similar looking faces, allowing overlap.
- Created 10 groups of 8 faces without overlap



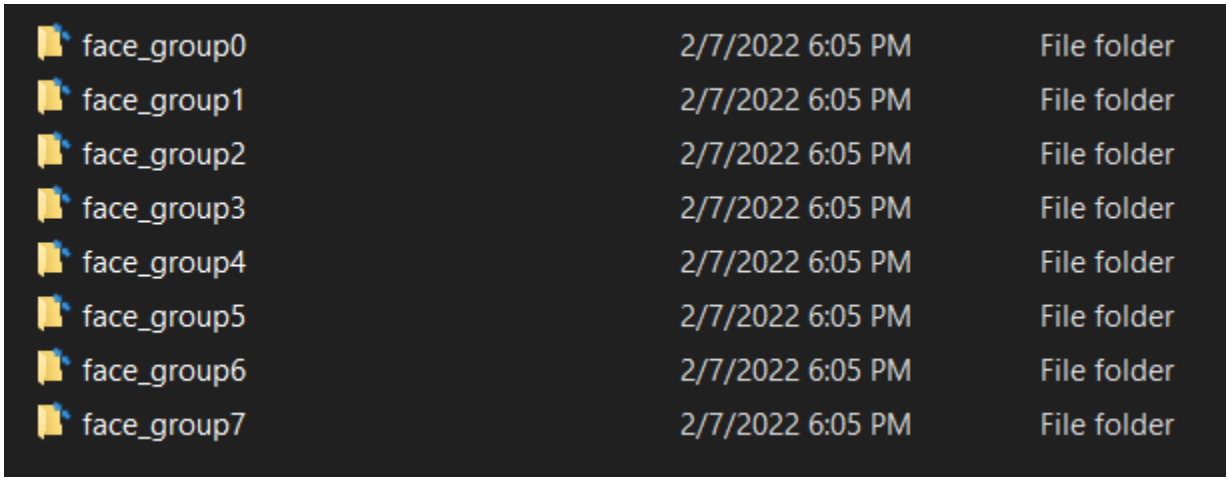


February 2nd, 2022

- Worked with Alessandra to complete sections 1 and 2 of the SRS and SDD documents

February 6th, 2022

- Wrote python script to create face clusters using a k-means algorithm
- Created 8 clusters containing varying amount of patients



February 13th, 2022

- Reached out to GANFit database owners: the MICC, in order to download the dataset and generate a larger data pool. However, they were unable to help me and we had to scrap GANFit.

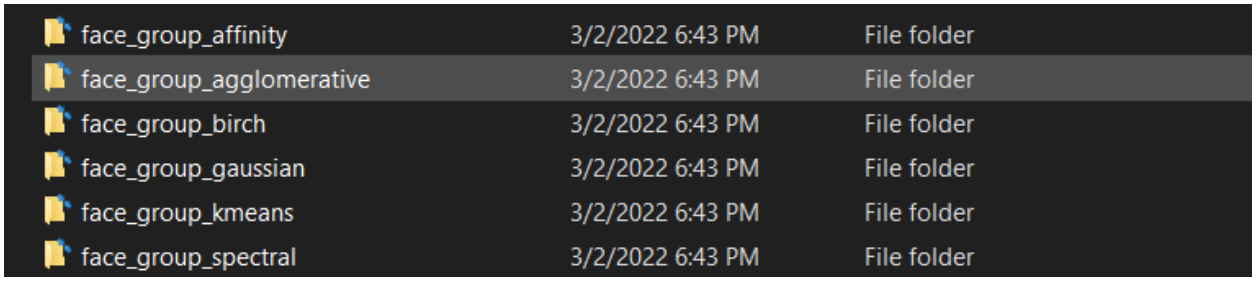
February 20th, 2022

- Researched different clustering algorithms to implement in order to optimize the face clusters.
- Decided upon 5 others in addition to Kmeans: affinity propagation, BIRCH, agglomerative, spectral, and Gaussian mixture clustering

March 2nd, 2022

- Wrote python script to generate clusters using multiple different clustering algorithms, including, affinity propagation, BIRCH, agglomerative, spectral, and Gaussian mixture.

```
def gaussian_mixture():
    df = pd.read_csv('database.csv')
    ratios = df.drop(['patient', 'class'], axis=1)

    gaussian = GaussianMixture(n_components=7).fit(ratios).predict(ratios)

    df['gaussian'] = gaussian

    newpath = r'C:\Users\jared\Documents\DigitizedRhinoplasty\DigitizedRhinoplasty-main\face_group_gaussian'
    if os.path.exists(newpath):
        shutil.rmtree(newpath)

    os.makedirs(newpath)

    for i in range(len(gaussian)):
        folder_location = (newpath + '\\face_group' + str(df.loc[i, 'gaussian']))
        file_location = "C:\\Users\\jared\\Documents\\DigitizedRhinoplasty\\DigitizedRhinoplasty-main\\ScanI

        if not (os.path.exists(folder_location)):
            os.makedirs(folder_location)

        try:
            shutil.copy(file_location, folder_location)
        except (FileNotFoundError):
            pass

gaussian_mixture()
spectral()
agglomerative()
birch()
affinity_propagation()
kmeans()
```



- Each folder contains a varying number of clusters depending on the settings of the algorithm.

March 6th, 2022

- Updated the SRS and SDD documents to align with new goals for the 3rd Sprint.

March 16th, 2022

- Created sprint 3 demo presentation.
- Identified all the nose points to plot and manipulate

Nose Points

- Alar base junction - ac_l ac_r
- Alar rim's highest point - armax_l armax_r
- Alar flare - al_l al_r
- Anterior point of nostril - stn_l stn_r
- Columellar break point - cb
- Columellar rim - cmin_l cmin_r
- Maxilloanteriorale - ma_l ma_r
- Maxillofrontale - mf_l mf_r
- Nasal parenthesis - np_l np_r
- Posterior point of nostril - itn_l itn_r
- Pronasale - prn
- Rhinion - r
- Sellion - se
- Sill-base junction - sbj_l sbj_r
- Subalare - sbal_l sbal_r
- Subnasale - sn_l sn_r
- Supratip break point - s
- Tip defining point - td_l td_r

March 24th, 2022

- Wrote python code the gather the average distances of each ratio to the centroid of the kmeans clusters.

```
'''GET AVERAGES OF DISTANCES'''
for k in range(len(sum_of_distances)):
    for ratio in range(5):
        sum_of_distances[k][ratio] = sum_of_distances[k][ratio] / cluster_counts[k]

average_distance = [0, 0, 0, 0, 0]

for row in range(5):
    for column in range(len(sum_of_distances)):
        average_distance[row] = average_distance[row] + sum_of_distances[column, row]

average_distance[:] = [value / len(sum_of_distances) for value in average_distance]

return average_distance
```

April 3rd, 2022

- Wrote the python code to gather the average distances of each ratio to the centroid for the remaining clustering algorithms: affinity propagation, birch, agglomerative, spectral, and gaussian mixture.

```
    try:
        shutil.copy(file_location, folder_location)
        for j in range(5):
            sum_of_distances[df.loc[i, 'gaussian'], j] = sum_of_distances[df.loc[i, 'gaussian'],
    except (FileNotFoundError):
        pass

'''GET AVERAGES OF DISTANCES'''
for k in range(len(sum_of_distances)):
    for ratio in range(5):
        sum_of_distances[k][ratio] = sum_of_distances[k][ratio] / cluster_counts[k]

average_distance = [0, 0, 0, 0, 0]

for row in range(5):
    for column in range(len(sum_of_distances)):
        average_distance[row] = average_distance[row] + sum_of_distances[column, row]

average_distance[:] = [value / len(sum_of_distances) for value in average_distance]

return average_distance
```

April 8th, 2022

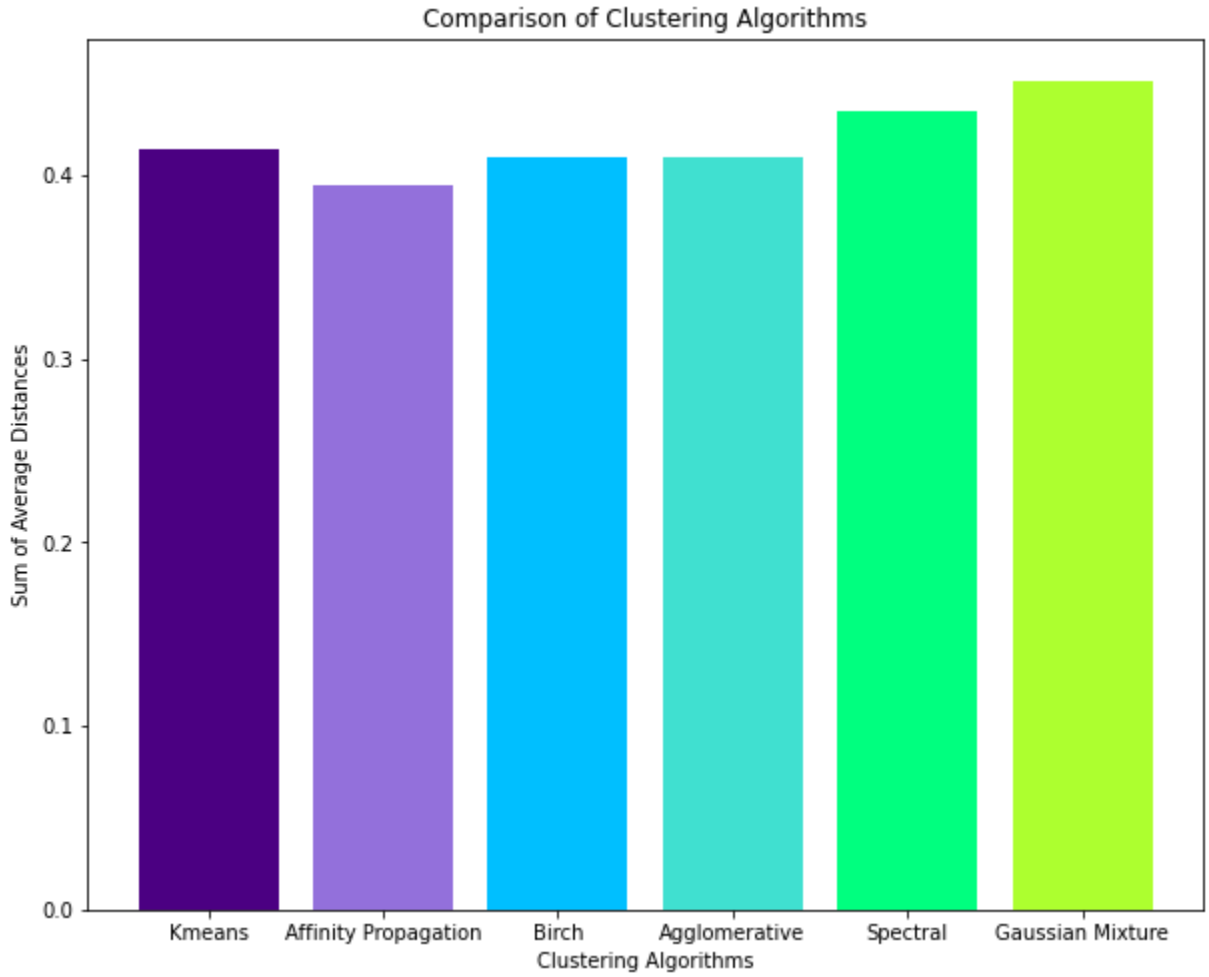
- Met with the group and finalized the system requirement specifications and the system design document.

April 15th, 2022

- Met with the group and finalized the system test plan

April 17th, 2022

- Wrote python code to generate graph comparing all six clustering algorithms




```
data = {'Kmeans':sum(kmeans_results), 'Affinity Propagation':sum(affinity_propagation_results),
        'Birch':sum(birch_results), 'Agglomerative':sum(agglomerative_results),
        'Spectral':sum(spectral_results), 'Gaussian Mixture':sum(gaussian_results)}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 8))

# creating the bar plot
plt.bar(courses, values, color =['indigo', 'mediumpurple', 'deepskyblue', 'turquoise', 'springgreen'
        width = 0.8)

plt.xlabel("Clustering Algorithms")
plt.ylabel("Sum of Average Distances")
plt.title("Comparison of Clustering Algorithms")
plt.show()
```