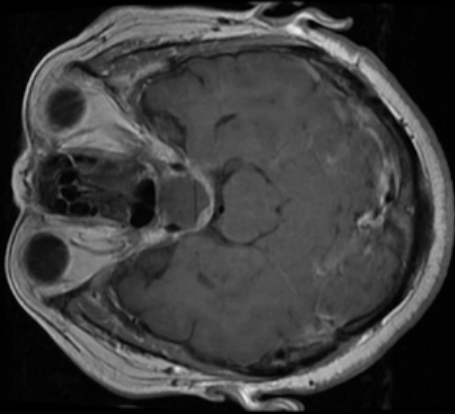# Brain Cancer Segmentation

DAVIDE BELCASTRO - 1962536
LUCIAN DORIN CRAINIC - 1938430
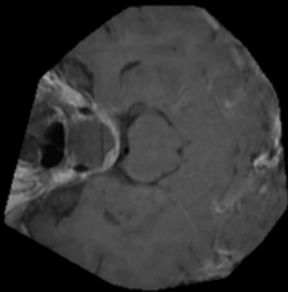
Segmentation

1. Skull Stripping
2. Mean color of the tumor
3. Segmentation of the tumor

Skull Stripping

The goal of this stage is to go and eliminate the edge of the brain and leave only the brain tissue

Mean color of the tumor

In this phase the average color of the brain is analyzed and areas of pixels are sought that have an average color quite distant from the color of the brain

```python
'''
this function is used to segment the image using K-Means Clustering.
'''
def KMeansClustering(data, k,img):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(data)
    labels = kmeans.predict(data)
    segmented_data = np.uint8(kmeans.cluster_centers_[labels])
    segmented_img = segmented_data.reshape(img.shape)
    return segmented_img
```

```
def get_tumor(color_tumor,brain,segm,mylist,area_contorno_esterno,colore_cervello):
    lista_contorni = []
    for value in mylist:
        mask = np.zeros(brain.shape[:2], np.uint8)
        mask[segm == value] = 255
        contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        for el in contours:
            el = cv2.convexHull(el)
            area = cv2.contourArea(el)
            circumference = cv2.arcLength(el, True)
            if circumference == 0:
                continue
            circularity = 4 * np.pi * (area / (circumference * circumference))
            if (circularity >= 0.3 and area >= 0.005*area_contorno_esterno and area <=
0.15*area_contorno_esterno) or(circularity >= 0.45 and area > 0.15 * area_contorno_esterno and area <=
0.20 * area_contorno_esterno) or (circularity >= 0.55 and area > 0.20 * area_contorno_esterno and area
<= 0.22 * area_contorno_esterno) :
                mask2 = np.zeros(brain.shape[:2], dtype="uint8")
                diz = {}
                cv2.drawContours(mask2, [el], -1, 255, -1)
                for i in range(0, brain.shape[0]):
                    for j in range(0, brain.shape[1]):
                        if mask2[i][j] == 255:
                            if brain[i][j] in diz:
                                diz[brain[i][j]] += 1
                            else:
                                diz[brain[i][j]] = 1
                somma = 0
                for k,v in diz.items():
                    somma+=k
                media = somma/len(diz)
                lista_contorni.append((el, media,area,circularity))
```

```
            diz_contorni = []
            for tup in lista_contorni:
                media_pesata = get_media_pesata(tup,color_tumor,area_contorno_esterno,colore_cervello)
                diz_contorni.append(media_pesata)
            brain = cv2.cvtColor(brain, cv2.COLOR_GRAY2BGR)
            index = 0
            diz_appoggio = {}
            for el in lista_contorni:
                r_random = np.random.randint(0, 255)
                g_random = np.random.randint(0, 255)
                b_random = np.random.randint(0, 255)
                cl = (r_random, g_random, b_random)
                diz_appoggio[(diz_contorni[index],cl)] = (el[0],el[1])
                cv2.drawContours(brain, [el[0]], -1, cl, 2)
                index+=1
            dizionario_2 = {}
            for k,v in diz_appoggio.items():
                if k[0] >= 1.15:
                    dizionario_2[k] = v
            diz_appoggio = dizionario_2
            diz_appoggio = dict(sorted(diz_appoggio.items(), key=lambda item: item[0][0]))
            diz_appoggio = dict(reversed(list(diz_appoggio.items())))
            if len(diz_appoggio) == 0:
                return "no tumor",0
            my_tumor = diz_appoggio[list(diz_appoggio.keys())[0]][0]
            color = diz_appoggio[list(diz_appoggio.keys())[0]][1]
            area = cv2.contourArea(my_tumor)
            circumference = cv2.arcLength(my_tumor, True)
            if circumference == 0:
                return "no tumor",0
            circularity = 4 * np.pi * (area / (circumference * circumference))
            percentuale_area_tumore = (area / area_contorno_esterno) * 100
            if circularity >=0.75:
                probability =
get_probability_of_tumor(area,circularity,color,colore_cervello,area_contorno_esterno)
                return my_tumor,probability
            else:
                return "no tumor",0
```

Segmentation

In this part we analyze the image returned from phase 1 and the value returned from phase 2, the k-mean algorithm is applied and the "best" contour is found

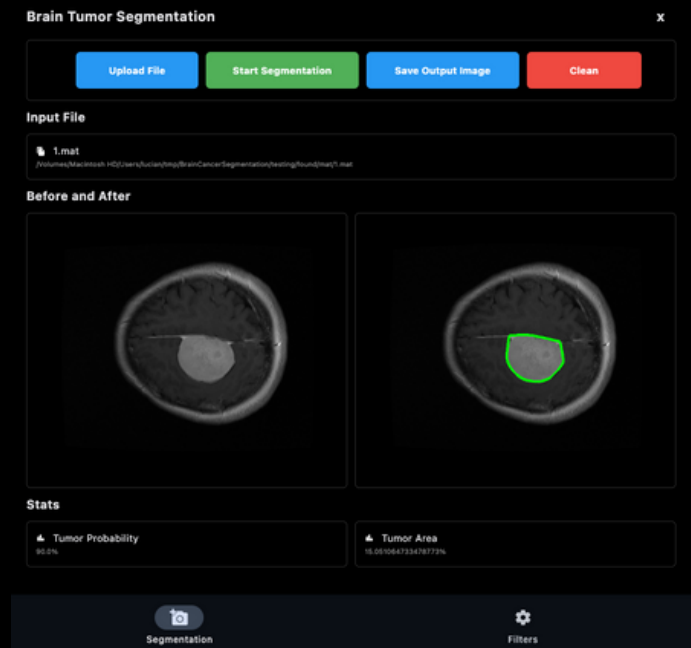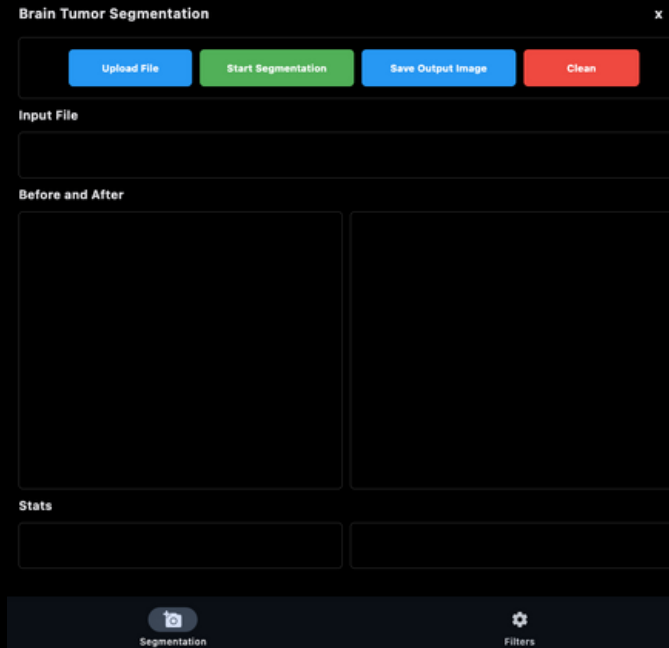# Trustworthiness of the program

10%

90%
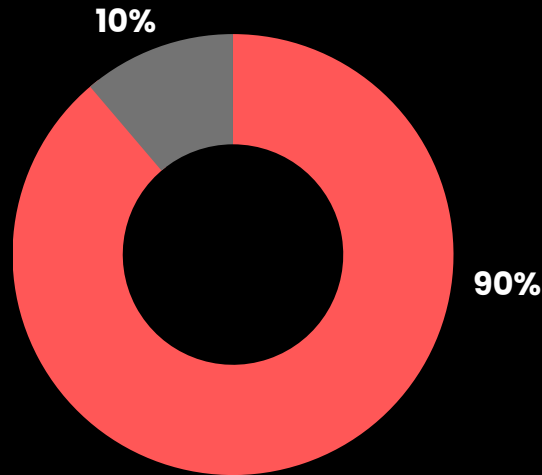
To get an estimate of how reliable the program was, tests were carried out by comparing the segment returned by the program with the "true" segment taken from the dataset.

Interface (GUI)

1. Brain Cancer Segmentation
2. Segmentation Filters

Brain Cancer Segmentation

The user can choose an MRI image of a patient's brain and begin segmentation

Brain Tumor Segmentation   x

| Upload File | Start Segmentation | Save Output Image | Clean |

Input File

Before and After

Stats

Segmentation   Filters

Brain Tumor Segmentation   x

| Upload File | Start Segmentation | Save Output Image | Clean |

Input File

📎 1.mat
/Volumes/Macintosh HD/Users/lucian/tmp/BrainCancerSegmentation/testing/round/mat/1.mat
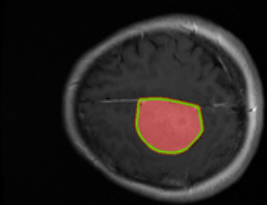
Before and After

Stats

| ⬇ Tumor Probability | ⬇ Tumor Area |
| 90.0% | 16.051064733478773% |

Segmentation   Filters

Segmentation Filters

The user can select at his choice different filters to apply to the image returned from the segmentation

Let's take a look at an example