# Brain Tumour Segmentation

By Ishaan Buch (18BCE079) and Jaineet Shah (18BCE083)

# Table Of Contents

# 1. Introduction

A brain tumour is a collection of abnormal cells in the brain. Our skull which encloses our brain is very rigid and any growth inside this restricted space can create problems to the body. Tumours are basically of two types cancerous (malignant) and non-cancerous (benign). When any tumour grows it increases the pressure inside the skull which can lead to brain damage. Brain tumour segmentation is the process of separating the tumour from normal brain tissues; in clinical routine, it provides useful information for diagnosis and treatment planning. However, it is still a challenging task due to the irregular form and confusing boundaries of tumours. Tumour cells thermally represent a heat source; their temperature is high compared to normal brain cells.

Magnetic Resonance Imagining also known as MRI is a imaging modality which is commonly used in clinical routine as it offers high spatial resolution and high contrast images of soft tissues. MRI is able to provide information about shape, size and the localisation of brain tumours for more accurate diagnosis and treatment planning. So most of the research in diagnosis and study of brain tumours involves MRI images.

Brain tumour segmentation form MRI images is a challenging as well as crucial task in diagnosis. Image segmentation is an active field in medical imaging, which consists in extracting from the image one or more regions forming the area of interest.

# 2. Segmentation Process

We have two types of input images, the ones having tumour present and the ones that do no have tumour present. The following are two examples

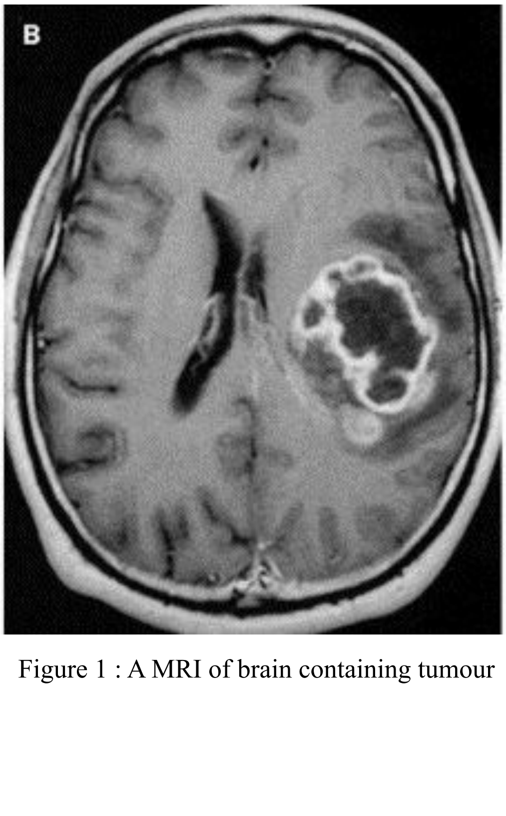


Figure 1 : A MRI of brain containing tumour

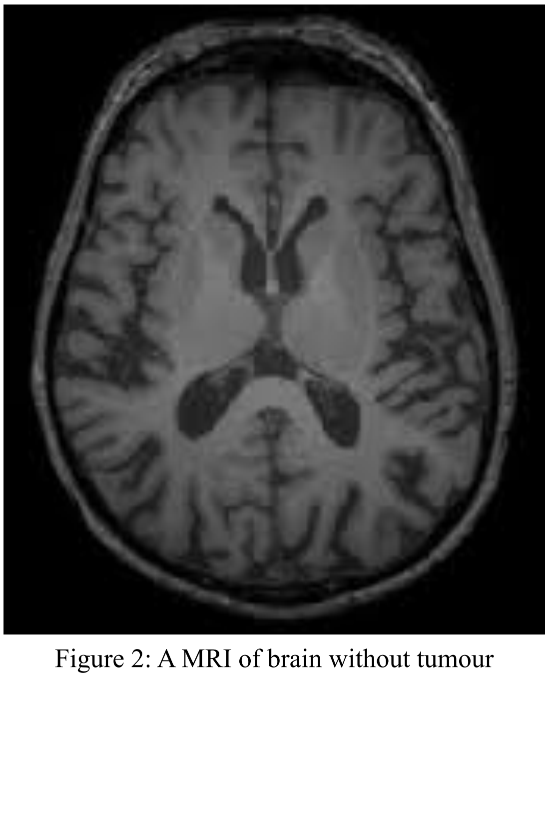

Figure 2: A MRI of brain without tumour

The following is the process followed for brain tumour segmentation:

1. Converting the image to grayscale.
2. Applying median filtering
3. Applying edge detection using Sobel filters
4. Thresholding the intensity
5. Determining the foreground and background using morphological opening
6. Watershed algorithm to isolate the tumour

After performing all the steps we will be able to separate the tumour from the whole brain and we will be able to view it better. We use three libraries for the whole process, they are,

I. OpenCV (for image manipulation)
II. Matplotlib (to display the images)
III. Numpy (Used for matrix manipulation)

We use the image below for tumour segmentation

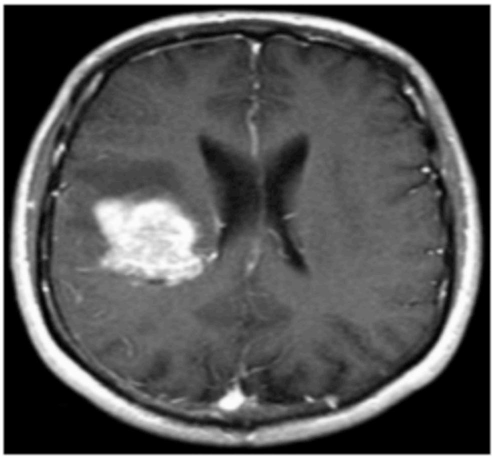

Figure 3: The MRI image used for tumour segmentation

## 2.1 Converting to Grayscale

We will use OpenCV2 to read an image and convert it into grayscale. OpenCV is a Python library used to solve computer vision problems. cv2.imread() method is used to load an image from a specified path and the image is stored in the variable. We use the method cv2.cvtcolor() to convert the image from one colour space to another. And finally we use the method cv2.imshow() to display the transformed image.

```
gray_img = cv2.cvtColor( orig_img, cv2.COLOR_BGR2GRAY )
plt.imshow(gray_img,cmap='gray')
plt.axis('off')
plt.show()
```

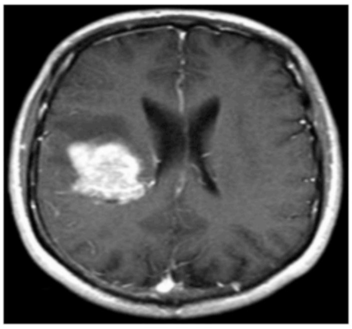

Figure 4: Code snippet and output for converting to grayscale

## 2.2 Applying Median Filter

Median Filter which is also known as nonlinear filtering is a method used to eliminate salt and pepper noise from the image. In this method the pixel value is replaced with the median of all the neighbouring pixels. We used the method medianBlur() provided by the OpenCV library in the improc class.

```
# To remove salt and pepper noise
# Using 5*5 kernel
median_filtered = cv2.medianBlur(gray_img, 5)
plt.imshow(median_filtered,cmap='gray')
plt.axis('off')
plt.show()
```

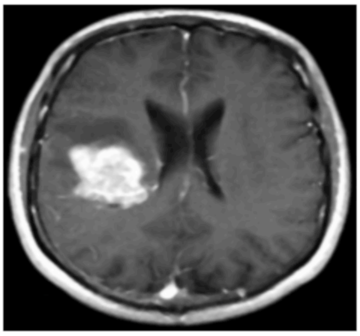

Figure 5: Code snippet and output for median filtering

## 2.3 Edge Detection using Sobel filters

The sobel filter is used to detect two types of edges i.e vertical and horizontal. Edge detection is important for image processing as most of the shape information of an image is enclosed in the edges. So we first detect the edges of an images using the filters (Sobel filter in this case) and then by enhancing the image which contains edges which will increase the sharpness of the image and the image will become more clearer.

```
Gx= np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
print("Gx \n", Gx)
Gy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
print("Gy \n", Gy)
img_sobelx = cv2.Sobel(median_filtered,cv2.CV_8U,1,0,ksize=3)
img_sobely = cv2.Sobel(median_filtered,cv2.CV_8U,0,1,ksize=3)
#del f = Gx + Gy
# Adding mask to the image
img_sobel = img_sobelx + img_sobely+gray_img
plt.imshow(img_sobel,cmap='gray')
plt.axis('off')
plt.show()
```

```
Gx
 [[-1  0  1]
 [-2  0  2]
 [-1  0  1]]
Gy
 [[ 1  2  1]
 [ 0  0  0]
 [-1 -2 -1]]
```

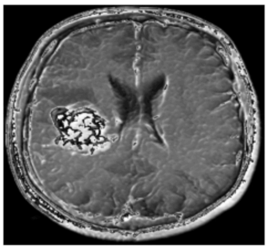

Figure 6: Gx is the vertical filer and Gy is the horizontal filter

When we apply the masks it basically works as a first order derivative and calculates the difference in intensities in an edge region. As the centre column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also the centre values of both the first and third column is 2 and -2 respectively. This give more weight age to the pixel values around the edge region. This increase the edge intensity and it become enhanced comparatively to the original image. Same process is applied for horizontal edges.

We can see that on applying the filters we are able to see the edges more clearly and we are able to separate the components in the MRI.

## 2.4 Thresholding the Intensity

Thresholding is one of the simplest methods for segmenting images, we use the threshold() method provided by the OpenCV library to perform thresholding on the image. By thresholding the image we are able to create binary images i.e the image consist of two colours basically white and black.If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is cv2.threshold. First argument is the source image, which should be a grayscale image. The second value is the threshold value which will be used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value.

```
threshold = 200
maxValue = 255

# Threshold the pixel values
th, thresh = cv2.threshold(img_sobel, threshold, maxValue, cv2.THRESH_BINARY)
plt.imshow(thresh,cmap='gray')
plt.axis('off')
plt.show()
```

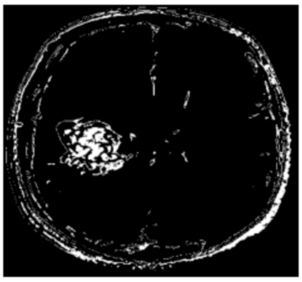

Figure 7: Thresholding is performed on the image and the output is displayed

## 2.5 Performing Morphological opening

Morphological image processing is collection of non linear functions used for shapes or morphology of the features of an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. Morphological operations can also be applied to greyscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest. We perform two function erosion and dilation.

The erosion of a binary image $f$ by a structuring element $s$ (denoted $f \ominus s$) produces a new binary image $g = f \ominus s$ with ones in all locations $(x,y)$ of a structuring element's origin at which that structuring element $s$ fits the input image $f$, i.e. $g(x,y) = 1$ is $s$ fits $f$ and 0 otherwise, repeating for all pixel coordinates $(x,y)$.

The dilation of an image $f$ by a structuring element $s$ (denoted $f \oplus s$) produces a new binary image $g = f \oplus s$ with ones in all locations $(x,y)$ of a structuring element's origin at which that structuring element $s$ hits the the input image $f$, i.e. $g(x,y) = 1$ if $s$ hits $f$ and 0 otherwise, repeating for all pixel coordinates $(x,y)$. Dilation has the opposite effect to erosion -- it adds a layer of pixels to both the inner and outer boundaries of regions.

## 2.6 Watershed algorithm to isolate the tumour

The watershed algorithm is used for segmentation i.e separate different objects from an image. Starting from user-defined markers, the watershed algorithm treats pixels values as a local topography (elevation). The algorithm floods basins from the markers until basins attributed to different markers meet on watershed lines. In many cases, markers are chosen as local minima of the image, from which basins are flooded.

```
contours, hierarchy = cv2.findContours(sure_fg,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

# Creating a numpy array for markers and converting the image to 32 bit using dtype paramter
marker = np.zeros((gray_img.shape[0], gray_img.shape[1]),dtype = np.int32)

marker = np.int32(sure_fg) + np.int32(sure_bg)

# Marker Labelling
for id in range(len(contours)):
    cv2.drawContours(marker,contours,id,id+2, -1)

marker = marker + 1

marker[unknown==255] = 0

copy_img = orig_img.copy()

cv2.watershed(copy_img, marker)

imgplt = plt.imshow(marker)
plt.colorbar()
plt.show()
```

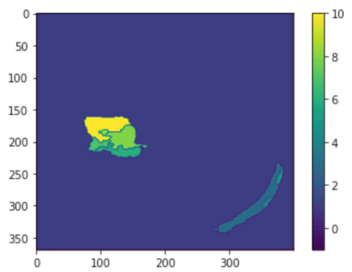

Figure 8: Using the watershed algorithm we are able to isolate the tumour.

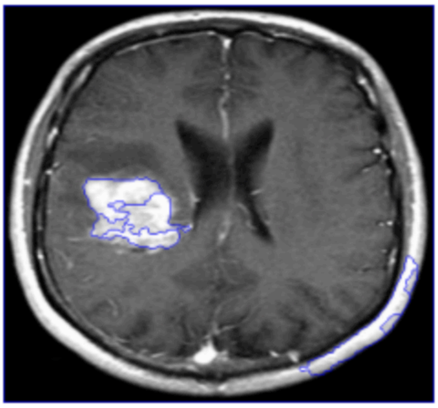

Figure 9: The purple lines separate the tumour in the MRI image

# 3. Conclusion

Brain tumour segmentation is a challenging task and there are many methods used to perform it more accurately and effectively. The method proposed above is the basic method and various other methods can be used like thermal imaging, fuzzy clustering, artificial neural networks etc. Early detection is important because when abnormal tissue or cancer is found early, it may be easier to treat. By the time symptoms appear, cancer may have begun to spread and be harder to treat.It can be helpful for people to discuss the potential harms as well as benefits of different cancer screening tests with their doctors.