NILE UNIVERSITY

Image Segmentation Improvements using Differential Equations

A Math203s Project Report

By

Antony Emil Kiroles 202000897

Esraa Negm Sayed 202000799

Mohamed Adel Ali 202001297

Mariam Amr Barakat 202000210

Mohamed Abdelmaged Essawey 202000440

Rawan Mohamed Elfaramawy 202001762

Yara Mahfouz 202001787

Submitted in partial fulfillment of the requirements

for Math-203 Project

March 27, 2022

ABSTRACT

It is acknowledged that computer vision advancement is crucial to help us in automation in various fields with great accuracy. The first step for a computer to be able to imitate human vision is to be able to segment the important parts of the image efficiently and accurately. In this paper, we explored the various image segmentation methods and focused on implementing and improving image segmentation with the use of differential equations. In order to reach that goal we used python programming language and bunch of its libraries as numpy and matplotlib that helped us to implement the code for image segmentation. It was concluded that improving the preprocessing of an image before segmentation can greatly improve the segmentation afterward. We used nonlinear diffusion filter to improve the preprocessing before segmentation and Watershed Region-based Segmentation for the image segmentation.

Table of Contents

List of Tables

List of Figures

SECTION I. Introduction

When you take a selfie, what is the first thing that comes to mind? It's your face! Yes, you're correct. Your brain can locate and separate your face from the rest of the image. But now the goal is to get the computer to notice. Will it work, and if so, how? The technique of partitioning a digital image into several image segments, also known as image regions or image objects, is known as image segmentation. The purpose of segmentation is to make an image more intelligible and easier to examine by simplifying and/or changing its representation. Image segmentation is the first stage in image recognition and registration, as well as the foundation of computer vision and other high-level image processing. It distinguishes the target from the complicated background using visual features like gray scale, texture, and shape. It has several levels, beginning with noise removal and progressing to text, object, and human identification, before progressing to more complex levels such as anomaly detection and emotion recognition. There are various established techniques for image segmentation. Each of these techniques is significant. To complete the required segmentation, each technique can be applied to specific various images. The popular techniques used for image segmentation are thresholding method, edge detection-based techniques, region-based techniques, clustering-based techniques, watershed-based techniques, partial differential equation-based, and artificial neural network-based techniques etc. To add, image segmentation methods can be divided into two categories: one is based on image information, morphology, topology, partial differential equations, and so on.

The other methods are deep learning-based segmentation, image segmentation based on region selection, RNN-based image segmentation, and up sampling-based segmentation. One of the more successful image segmentation approaches is the partial differential equation method, as this type of technology can deal not only with changes in topology structure of evolution curves, but also with the provided image directly, without the need for a large amount of training data, repeated adjustments, or network learning.

Image segmentation technology is becoming increasingly powerful in a variety of domains, including medical, industrial, and commercial applications. It has become widely used in medical science in recent years to aid in the understanding and gathering of information from biomedical images of natural human biological systems. The transformation of 2D to 3D images, as well as automated feature detection and image comparison, are all wonderful outputs of image processing technology. Furthermore, it is used in the textile sector for yarn parameters detection, textile surface roughness, and textile flaw, which has proven to be highly effective.

In order to do image segmentation we used watershed segmentation method which is a region-based technique that utilizes image morphology. You must select at least one marker within each object in the image, including the background as a separate object. Markers are selected by the operator or provided by an automated process that considers the application-specific knowledge of the object. Once the object is tagged, the object can be grown using morphological watershed transformations.

This paper aims to utilize differential equations to improve the accuracy of the preprocessing of image segmentation. This is in the hopes to contribute to better image analysis.

SECTION II. Background and Literature Review

A. Image segmentation and its importance

Image segmentation has been regarded as the initial step in image processing as it is a mechanism of dividing a single image into several pieces and segments. It is the process of assigning a label to each pixel in an image so that pixels with the same label have similar visual properties. It will smooth out the image and make it easier to evaluate. The techniques for segmentation are based on traditional methods, artificial intelligence techniques, and hybrid techniques.

Image segmentation is a critical and essential operation in image processing. Because it is used in nearly every field of science, including reducing noise from images, medical imaging, satellite imaging, machine vision, computer vision, biometrics, military, and Image Retrieval extracting features, and detecting and recognizing objects from a given image. One of the primary goals of image processing is to extract essential information from a given image without affecting the image's other properties.

B. Subsets of image segmentation

1) Semantic Segmentation:

This technique is one of the deep learning algorithms. It works to cluster some parts of the image that has the similar pattern in pixels. So that the image is divided into some categories. So, every pixel in the image belongs to a category in the image [1].

Idea: Recognizing, understanding what's in the image at the pixel level.

Input: Images.

Output: Regions, structures (Labels or categories).

Figure 1: Semantic Segmentation of a street scene

In figure 1 describes Semantic segmentation for an automated driving application, you can see how the different objects are labeled using segmentation masks [1].

2) Classification and Localization:

Semantic-based classification and localization: This technique used a machine learning model to detect if objects exist in the image or not. These machine learning models can be applied to videos or images [2].

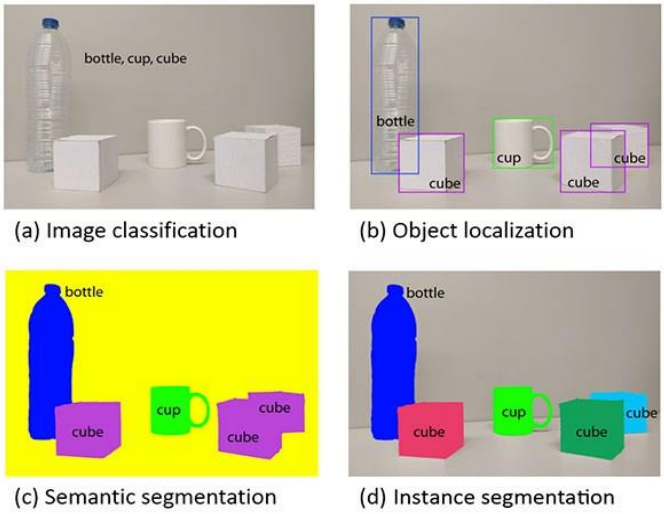

Figure 2: Types of object classification and localization

3) Object detection:

Is a newly appeared technology in the field of computer vision that works to identify objects or people and determine their location during their movement. In general object detection can be implemented using machine learning-based approaches or using deep learning approaches [3].

Figure 3: Object detection of vehicles

C. Methods used in image segmentation

1) Edge Based Segmentation:

This method is an intermediate step for image segmentation, it locates areas with strong contrast, in other words, it can find the linings of objects. The contrast can be in color, texture, or even gray levels, so it can be imagined that an execution of this method needs a type of mask that calculates the change in intensity and compare it to a threshold to know if it has high priority or is considered as noise. Moreover, there are many types of edges described in the image below, it explains the type of intensity change according to the space, this aids the algorithm to judge whether this is a legitimate edge or just noise [4].

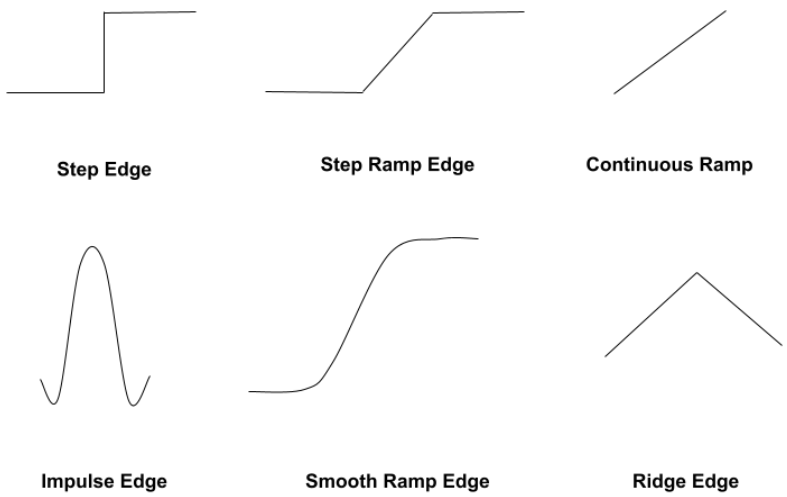

Figure 4: Types of edges [4]

To successfully implement edge detection, we first need to filter and blur our image, as this will help the algorithm not confuse noise with real edges. Secondly, we are required to enhance high intensity pixels, and then start to detect the edges. Finally, we will perform edge lining to create smooth edges [5].

There are various edge detector methods, we will briefly discuss the first order derivative types. This includes Roberts operator, Sobel operator and Prewitt operator. All those have similar procedure of using a mask with specific weights to measure where the edge is most likely going to be. The image below will showcase the mask, and how it helps to find the edges by calculations done both horizontally and vertically [4].

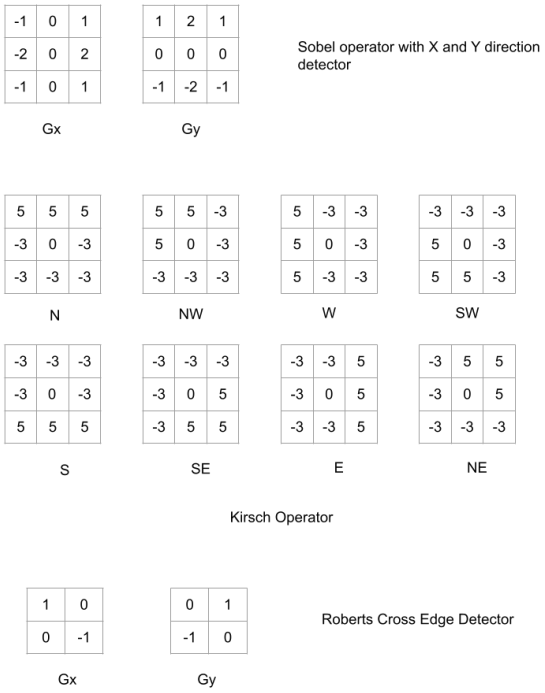

Figure 5: Examples of edge detection masking [4]

$$g = \sqrt{Gx^2 + Gy^2} \quad (1)$$

$$\theta = arctan\left(\frac{Gy}{Gx}\right) \quad (2)$$

2) <u>PDE Segmentation:</u>

While most of the classical image processing techniques are based on discrete mathematics, in which the techniques are developed based on the computer's standard interpretation for an image to be a discrete two-dimensional array of bytes with each byte representing a pixel [6], the main idea behind PDE methods in image segmentation is for an image to be seen as a function defined in a continuous space, and their algorithms can be implemented in a computer's discrete domain with the help of numerical analysis [6]. Some traditional PDE methods' equations are derived from curve evolution and curvature motion analysis, where an initial contour is defined then deforms towards the object boundary. The objects boundaries of an image are defined by an edge function generated from the image; however, the edges in the function are not connected, so the regions are still not recognized. Therefore, the initial contour is slowly adjusted to optimally fit on the nearby edges, resulting in identification of the regions [7]. A general curve evolution equation can be defined as [7, eq. 3]:

$$\frac{\partial C}{\partial t} = (\propto + \beta k)\vec{N} + (\vec{S} \cdot \vec{N})\vec{N} \qquad (3)$$

Where k is the curve curvature, $\vec{N}$ is the normal to the curve, $\propto$, β are constants, and $\vec{S}$ is the velocity field whose strength and direction depend on position and time.

Additionally, such methods contributed to the development of active contour models.The concept of Active contour model was first introduced by Kass, Wikin, and Terzopolos [8], and it have had a tremendous impact on the computer vision and image processing field. The authors developed a model, named "snakes", for which a snake is an energy-reducing spline lead by constraints forces and image energy forces that drag it towards image features like subjective contours, lines, and edges. This results in the snake contouring the target object which indicates a successful separation of the targeted pixels from the image for further analysis and processing (fig. 6 and 7).

Below is the snake's energy function representing the total of its exterior and internal energy [8, eq. 4]:

$$E_{snake} = \int_0^1 E_{snake}\big(v(s)\big)ds = \int_0^1 \Big(E_{internal}\big(v(s)\big) + E_{image}\big(v(s)\big) + E_{con}\big(v(s)\big)\Big)\,ds \qquad (4)$$

Where $E_{internal}$ is the internal elastic energy of the snake, which is responsible for regulating its deformation. E_{image} is the energy of the image, and E_{con} is the constraints forces imposed by the user; both serve as external energies that control the fitting of the contour onto an image.

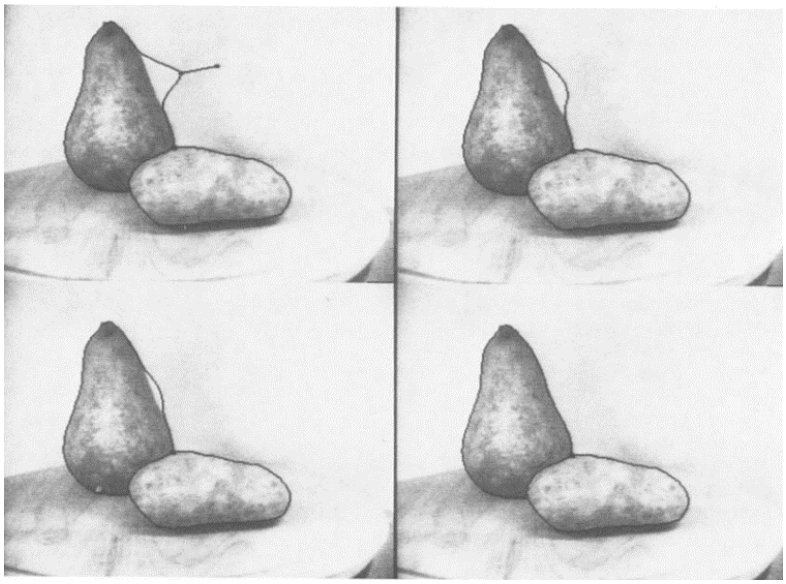

Figure 6: Two edge snakes on two objects with one snake being pulled out by the user in upper left image then snapping back in place in remaining one

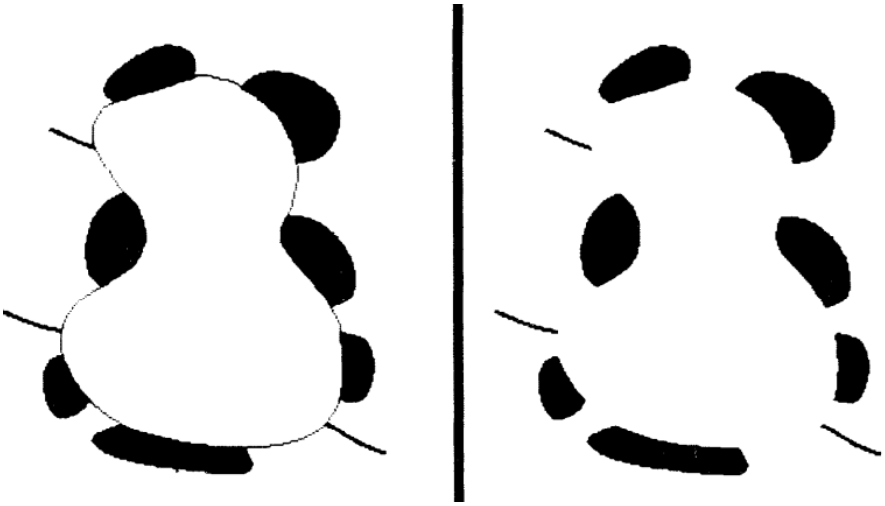

Figure 7 subjective contour illusion on right and a subsequent snake contouring on the left

While The model fulfilled its purpose and had shown success, two prominent disadvantages of the snake model were noted, which were noise sensitivity and inaccurate contour detecting in highly complex objects [9], hence the reason behind the subsequent development of other active contour models like the gradient vector flow model, and the balloon model, which are conceptually the same as the snake model, but with a more effective, well-defined development [9].

3) Watershed Region-based Segmentation:

In Monteiro et al. [10], a hybrid framework for region -based image segmentation. The framework utilizes watershed and region similarity graph techniques to overcome the short comings of each method. The image is first treated with an edge preserving noise filter (Bilateral filter) to reduce the image details without losing important features to decrease complexity. The filtered image is then segmented and then reassembled into micro regions in order to reduce the negative effects of over-segmentation. The micro regions are then connected to represent the final segmentation regions where micro-regions are connected depending on the difference in distance, orientation, and texture of the regions. The relation is calculated by measuring the weighted distance between the centers of the micro regions. The weights are then used to label and classify the regions.

4) ANN segmentation:

In Zahng [11], an Artificial Neural Network (ANN) was developed to help segment colored-images in a timely manner. This is achieved by utilizing a hybridized color space between the RGB and HSV color spaces. The hybridized color space helps in providing more accurate pixel

values. The image is first passed through a filter that reduces the detail in the image to help in blending the details that are not critical for the segmentation and might provide additional complexity to algorithm. The preprocessed image is then passed through to a Fast-Learning Artificial Neural Network (FLANN.) A FLANN was use d as it provides a versatile and flexible way to adjust the accuracy of the learning model by tuning the vigilance and tolerance, which affect the noise tolerance and number of segments respectively (fig. 9). The FLANN then clusters the images depending on RGBSV values and predefined class labels and after that the regions are calculated and segmented depending on whether they are spatially connected or not. The regions are then remerged depending on the labels and neighborhood size (fig. 8). The algorithm was able to correctly segment both computer-generated simple images and natural images, with the latter requiring further tuning by adjusting the neighborhood size to produce practical regions.

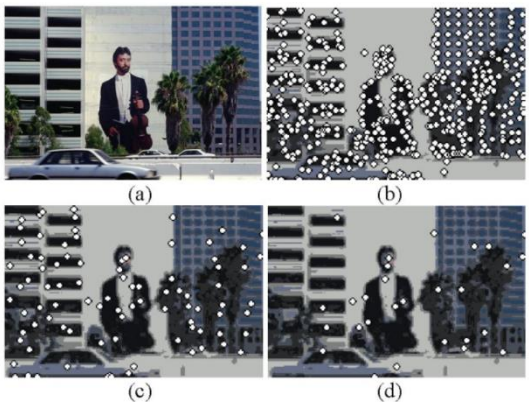

Figure 9: Region centers depending on neighborhood sizes (a) input image, (b) neighborhood size =2 (c) neighboring size= 10, (d) neighboring size= 20

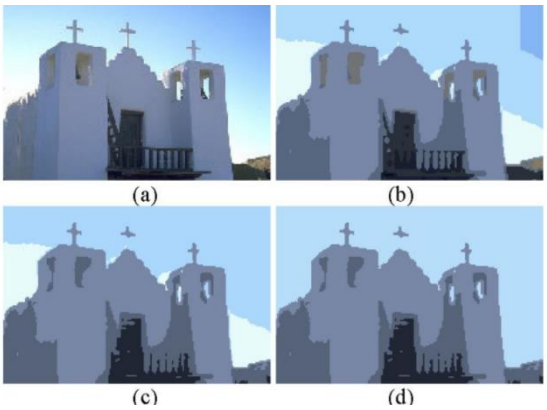

Figure 8: Effect of vigilance on segmentation regions (a) original image (b) vigilance = 1 (c) neighboring size= 0.8 (d) neighboring size= 0.6

D. Application of image segmentation

There are a lot of real-life problems that image segmentation can solve, including medical imaging and object detection.

Medical imaging is mainly done with the differential equations' algorithms. The image comes from a computed tomography scan (CT scan), which can show almost all the stages of cancer or Magnetic Resonance Imaging (MRI), which can show any brain or heart injury/damage. [12]

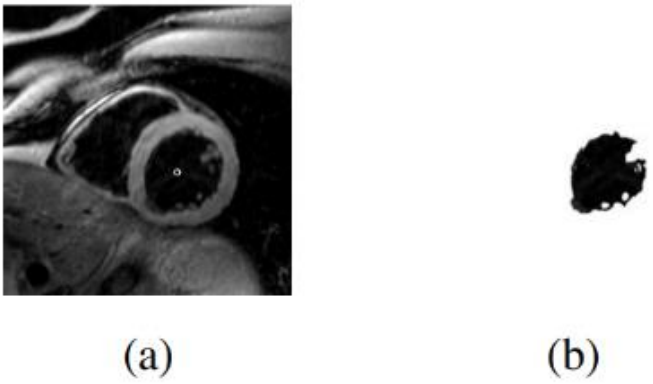

Figure 10: MRI heart image segmentation results

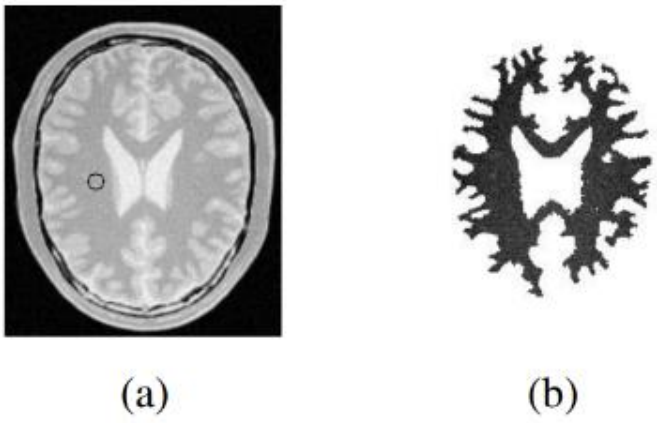

Figure 11: MRI brain image segmentation results

In Figures [10-11], the images illustrate the power of image segmentation in capturing the most vital part of the organ, which will greatly help the physician in diagnosing disorders at a faster pace. The images labeled as (a) are the scans made by the CT, the images labeled as (b) are the results of the image segmentation.

Object detection is also a crucial application of image segmentation, below are examples of object detections that can be made.

- Pedestrian detection
- Face detection and recognition
- Brake light detection
- Locate objects in satellite images (roads, forests, crops, etc.)
- Recognition Tasks

SECTION III. Methodology

In this section, we will be exploring the usage of partial differential equations to be able to improve the preprocessing of an image before segmentation. The goal [13] is to have enhanced contrast and to minimize energy, as this combined will greatly improve the image segmentation.

A. System Equations

a) Nonlinear diffusion filter:

The below equations describe the nonlinear diffusion filter [13] that we will try to implement.

$$\partial_t u = div(g(|\nabla u_\sigma|^2)\nabla u) \quad (5)$$

$$u(x,0) = f(x) \quad (6)$$

$$\partial_n u|_{\partial\Omega} = 0 \quad (7)$$

$$\nabla u_\sigma := \nabla(k_\sigma * u) \quad (8)$$

$$k_\sigma := \frac{1}{(2\pi\sigma^2)^{m/2}} exp\left(-\frac{|x|^2}{2\sigma^2}\right) \quad (9)$$

$$g(s^2) := \begin{cases} 1 & (s^2 = 0) \\ 1 - exp\left(\frac{-3.315}{(s/\lambda)^8}\right) & (s^2 > 0) \end{cases} \quad (10)$$

Table I: Variables and Parameters of Nonlinear diffusion

Parameters and Variables	Definition
u	dependent variable
t	Independent variable time
σ	Iterator representing the position in image
Ω	Our image domain $\Omega := (0, a_1) * ... * (0, a_m)$
m	The dimensions of image
x	Independent variable
n	The normal to the image boundary $\partial\Omega$

Equation (5) describes the partial differential equation for contrast enhancement, it must be noted that g is a function that finds diffusivity of the edge detector $|\nabla u_\sigma|^2$, and ∇u_σ is the gradient of a Gaussian-smoothed version of u, and equations (8-9) are used to derive it. Furthermore, equation (6) describes the filtered image $u(x,t)$, $f(x)$ is the resulting image represented in a numerical form.

To fully understand equation (5) we need to understand what is meant by div, it is a shorthand to indicate the dimensions of the image vertically and horizontally. The following is its formal definition [14].

If $v = v_x i + v_y j + v_z k$ then div(v) is defined as:

$$div(v) = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \qquad (11)$$

Where the v_x, v_y, v_z are the scalar representation of v in the x, y, z direction simultaneously.

B. Solving the PDE

a) Finite difference approximation:

Since we are working with images, the best approach to solve our partial differential equation was to use numerical methods, namely finite difference approximation. In this approach we represent each differential entity in terms of the function with an additional h added or subtracted. Each differential entity can be represented using forward difference, backward difference, or central difference. To solve the equation, we first need to discretize the equations using the finite scheme. Secondly, we will iteratively accumulate to find the next values, until we reach a graph that represents our solution. Figure 12 displays the backward and forward difference for x and y independent variables.

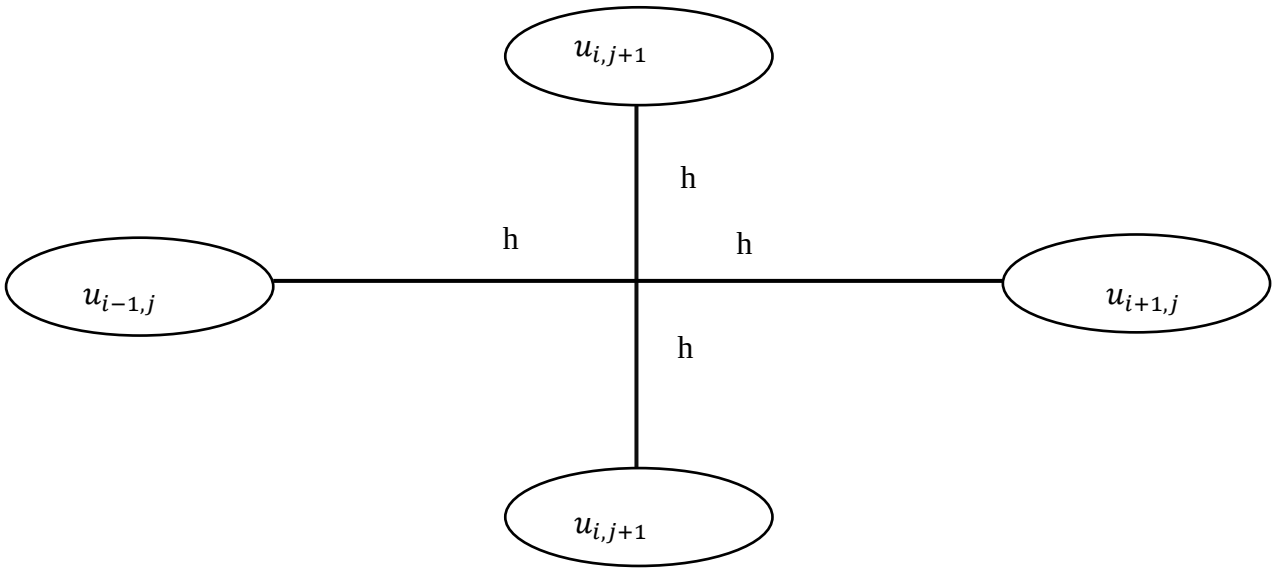

Figure 12: This illustrates the idea of representing our equation u, by adding or subtracting to either the x or y direction

Now, we will need to transform equation 5 to its comprehensive form. This will help us in being able to decipher it into finite differences. It is worth noting that the ∇ Laplacian operator is used to create a vector of required dimension. For example, $\nabla u = \frac{\partial u}{\partial x} i + \frac{\partial u}{\partial y} j + \frac{\partial u}{\partial z} k$, the Laplace operator was able to break down the function to its differential equation form and create a vector from it. Furthermore, notice that in equation 12 we created a vector with two dimensions because we only had 2 independent variables x and t.

$$\frac{\partial u}{\partial t} = div\left(g\left(\left|\frac{\partial u_\sigma}{\partial x} i + \frac{\partial u_\sigma}{\partial t} j\right|^2\right)\left(\frac{\partial u}{\partial x} i + \frac{\partial u}{\partial t} j\right)\right) \quad (12)$$

Subsequently, we need to discretize each of the above partial differential entities, so that they can be calculated using a computer. We will utilize forward difference in creating our equations because it is the easiest to work with.

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i,j}}{h} \quad (13)$$

$$\frac{\partial u}{\partial t} = \frac{u_{i,j+1} - u_{i,j}}{h} \quad (14)$$

Equations 13 and 14 display the discretization of both $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$, the discretization of $\frac{\partial u_\sigma}{\partial x}$ and $\frac{\partial u_\sigma}{\partial t}$ are respectively the same as the former, however we multiply by k_σ. Afterwards, we are required to represent equation 12 in its discretized simplified form. This will be done by correctly replacing equations 13 and 14 into equation 12.

$$\frac{u_{i,j+1} - u_{i,j}}{h} = g(|k_\sigma\left(\frac{u_{i,j+1} - u_{i,j}}{h} + \frac{u_{i+1,j} - u_{i,j}}{h}\right)|^2)(\frac{u_{i,j+1} - u_{i,j}}{h} + \frac{u_{i+1,j} - u_{i,j}}{h})) \quad (15)$$

$$u_{i,j+1} - u_{i,j} = g(|k_\sigma\left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i+1,j}}{h}\right)|^2)(u_{i,j+1} - 2u_{i,j} + u_{i+1,j}\,)\,) \quad (16)$$

$$u_{i,j+1} = u_{i,j} + g(|k_\sigma\left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i+1,j}}{h}\right)|^2)(u_{i,j+1} - 2u_{i,j} + u_{i+1,j}\,)\,) \quad (17)$$

Equations 15-17 reveal the final discretized version of the equation. The g is a function that returns the diffusivity of the diffusion equation. Equation 17 is considered the equation that will be used in the coding implementation.

b) Python Implementation:

```
import numpy as np
import math
import matplotlib.pyplot as plt

def g(x,y,z):
  sigma= 200
  m= 0.02
  h= 2
  lamda = 0.9
  k=1/((2*math.pi *sigma**2)**(m/2)) #* np.exp(-(x**2)/(2*sigma**2))
  s = k*((z-2*x + y)/h)
  s2 = 1 - np.exp(-3.315/(s/lamda))
  if s2==0:
    return 1
  return s2

x=np.linspace(0,1,101)
u=np.zeros(101)
u[50] = 1

for time in range(100):
  for i in range(1,100):
    u[i]+= g(u[i],u[i+1],u[i-1])*(u[i-1]-2*u[i]+ u[i+1])

plt.plot(x,u)
plt.show()
```

Figure 13: Solving PDE using finite difference in python

Figure 13 presents the code that helped solved our nonlinear diffusion filter. We embarked by calculating the diffusivity constant using the g function. Then, we initialized a u and x array that stores zeros, as it will yet be solved. Afterwards, we have nested for loops to iteratively solve for u.

c) *Graphing the solution:*

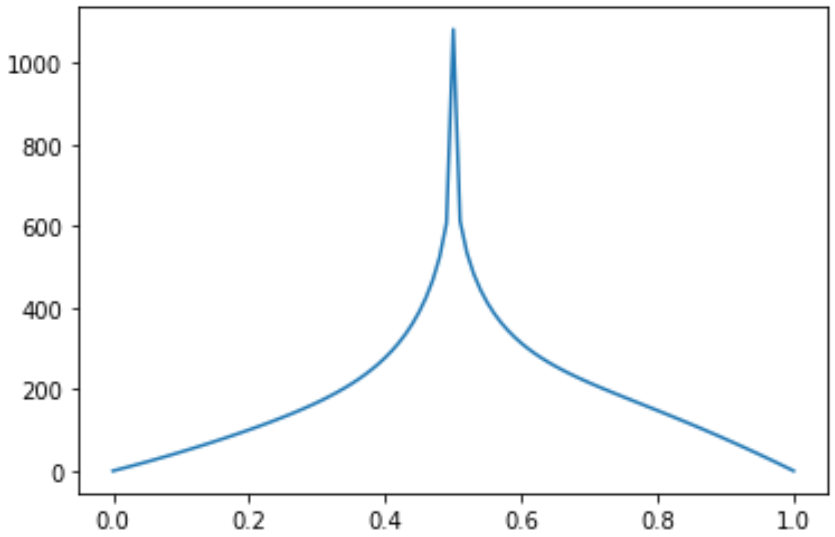

Figure 14: Python plotting our solution

Figure 14 reveals the solution for our PDE, it must be noticed that the input in our code was an array of zeros it should be an array that represents an image. However, for the sake of practicality, we have initialized the entire array with zeros. Notice how this graph has a very pointy turning point, this sharpness can be altered by the sigma variable.

C. Built-in preprocessing of image segmentation

. The program takes the image as an input in the form of a 2d-array of RGB values. The image is then subjected to a grayscale filter with a threshold applied on it to distinguish the background from the subjects of the image. The image is then subjected to a denoising filter to reduce unnecessary complexity. The resulting image is then segmented into regions of interest whose edges are marked and overlain on the original image Figure 12. Figure 13 contains the code that executed the above description

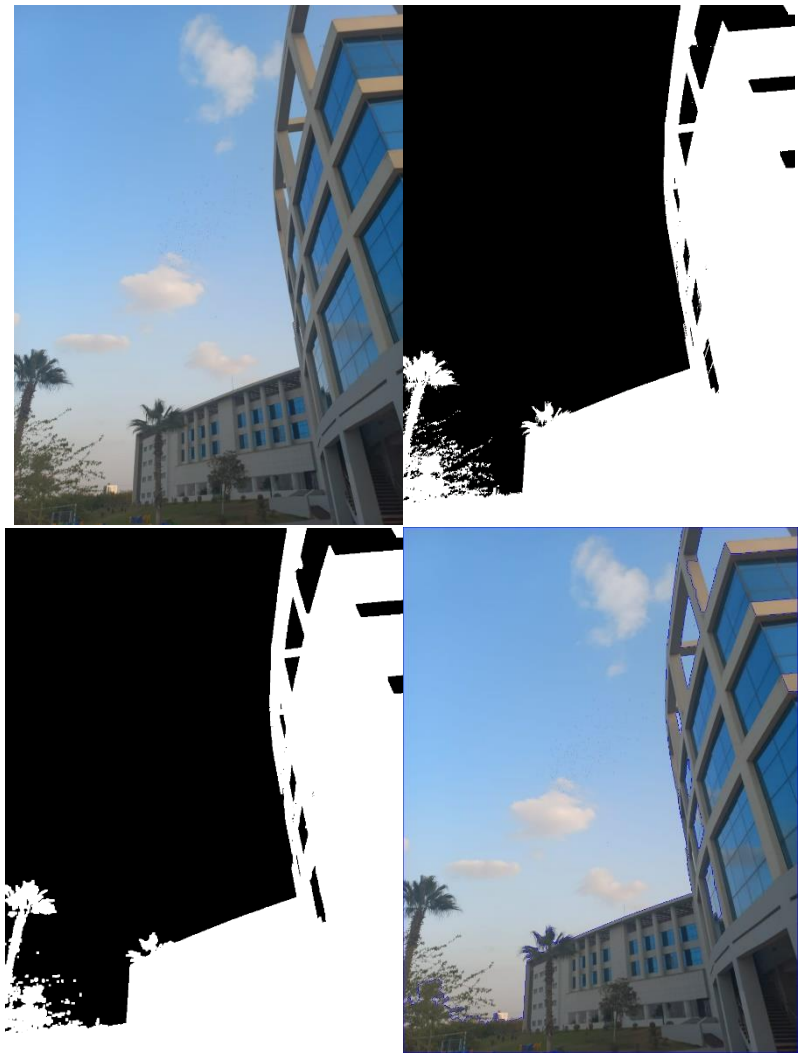

Figure 15: Top left, Original Image. Top right, the image after applying the grayscale filter and removing the background. Bottom Left, the image after denoising and removing extra details. Bottom right, Final Image

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('in.jpeg')
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(gray,0,255,cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
# noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv.morphologyEx(thresh,cv.MORPH_OPEN,kernel, iterations = 2)
# # sure background area
sure_bg = cv.dilate(opening,kernel,iterations=3)
# # Finding sure foreground area
dist_transform = cv.distanceTransform(opening,cv.DIST_L2,5)
ret, sure_fg = cv.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# # Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv.subtract(sure_bg,sure_fg)
# # Marker labelling
ret, markers = cv.connectedComponents(sure_fg)
# # Add one to all labels so that sure background is not 0, but 1
markers = markers+1
# # Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv.watershed(img,markers)
img[markers == -1] = [255,0,0]
```

Figure 16 Watershed segmentation

SECTION IV. Results

It can be concluded that focusing on the preprocessing of an image before the actual image segmentation process can make the process of segmentation more accurate. This can be performed by calculating the PDE that enhances contrast and increasing smoothness.

a) Alteration of sigma for sharper contrast enhancement

It seems that as we increase the value of sigma, the turning point of the graph becomes pointier. Which might mean higher contrast enhancement.

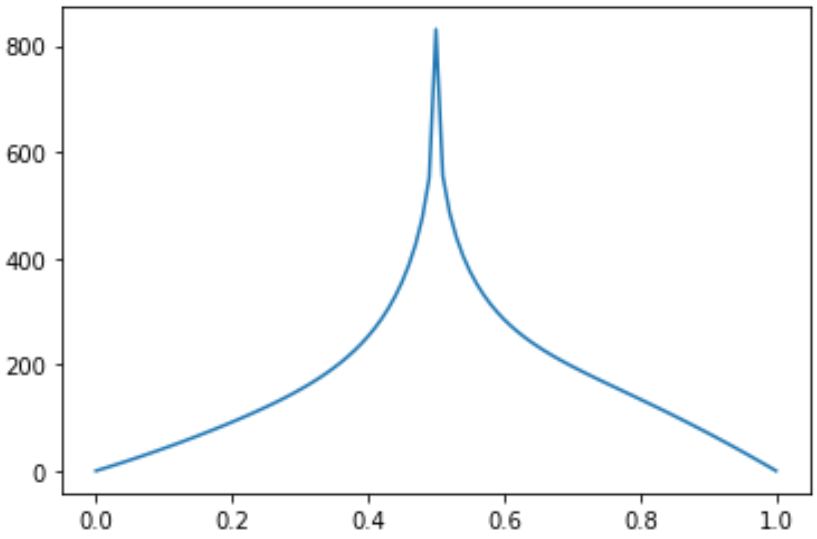

Figure 17: σ = 0.0002

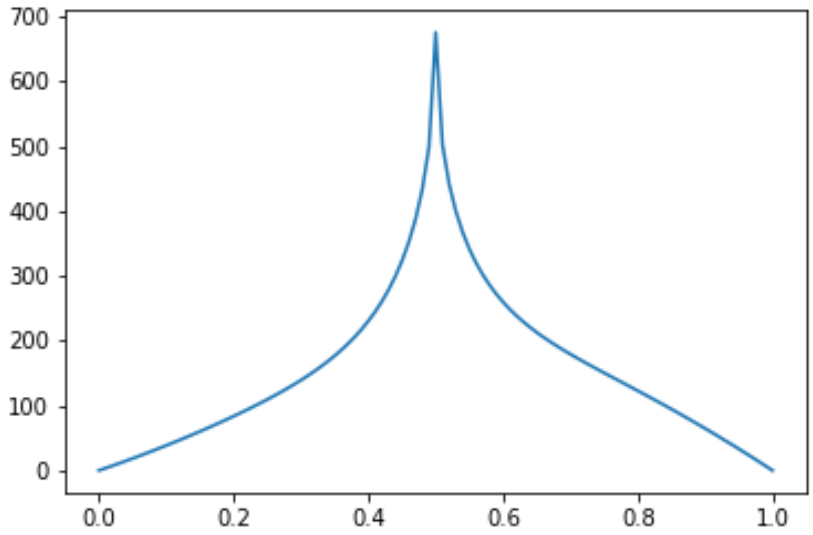

Figure 18: σ = 0.02

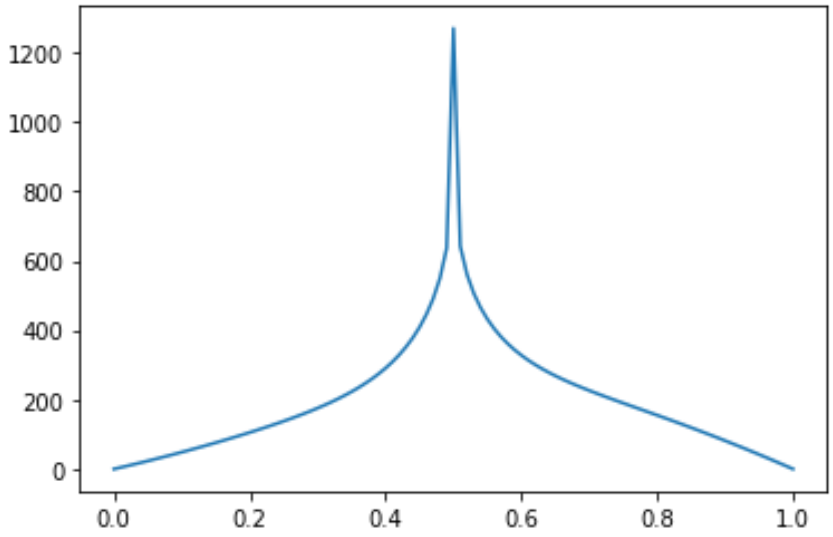

Figure 19: σ=20

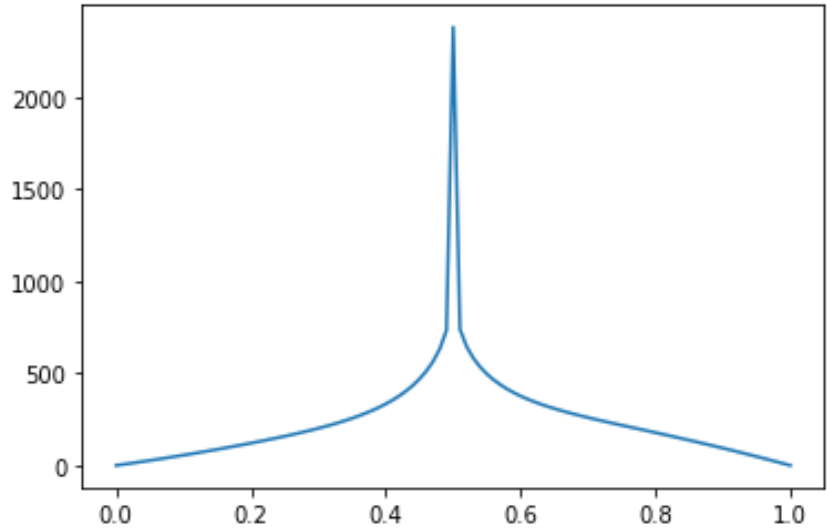

Figure 20: σ=20000

As the sigma grows, the graph turning point becomes extremely narrow, as sigma grows the effect slows down. This can be shown in figures 17-20.

Resources

[1] "Semantic segmentation," *Mathworks.com*. [Online]. Available: https://www.mathworks.com/solutions/image-video-processing/semantic-segmentation.html. [Accessed: 22-Mar-2022].

[2] "Classification, object detection and image segmentation," *Qualcomm Developer Network*. [Online]. Available: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation. [Accessed: 22-Mar-2022].

[3] "What is object detection?," *Mathworks.com*. [Online]. Available: https://www.mathworks.com/discovery/object-detection.html. [Accessed: 22-Mar-2022].

[4] M. Tyagi, "Image segmentation: Part 1," *Towards Data Science*, 18-Jul-2021. [Online]. Available: https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50. [Accessed: 22-Mar-2022].

[5] "Region and edge based segmentation," *GeeksforGeeks*, 18-Jul-2021. [Online]. Available: https://www.geeksforgeeks.org/region-and-edge-based-segmentaion/. [Accessed: 22-Mar-2022].

[6] A. Saberi, "Digital image processing: p051- Curve Evolution," 17-Mar-2013. [Online]. Available: https://www.youtube.com/watch?v=B9soiDHr9bo. [Accessed: 22-Mar-2022].

[7] B. Sumengen, B. S. Manjunath, and C. Kenney. "*Image Segmentation Using Curve Evolution," Department of Electrical and Computer Engineering University of California*. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.8564&rep=rep1&type=pdf [Accessed: 26-April-2022].

[8] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[9] D. Bhatt, "Active contours - A method for image segmentation in computer vision," *Analytics Vidhya*, 13-Sep-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/09/active-contours-a-method-for-image-segmentation-in-computer-vision/. [Accessed: 22-Mar-2022].

[10] F. C. Monteiro and A. Campilho, "Watershed framework to region-based image segmentation," in *2008 19th International Conference on Pattern Recognition*, 2008.

[11] *Researchgate.net*. [Online]. Available: https://www.researchgate.net/publication/4286155_Fast_learning_artificial_neural_network_FLANN_based_color_image_segmentation_in_R-G-B-S-V_cluster_space. [Accessed: 22-Mar-2022].

[12] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: image segmentation using expectation-maximization and its application to image querying," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 8, pp. 1026–1038, 2002.

[13] Researchgate.net. [Online]. Available: https://www.researchgate.net/publication/220602717_Efficient_Image_Segmentation_Using_Partial_Differential_Equations_and_Morphology. [Accessed: 26-Mar-2022].