ASSIGNMENT 2:
ELEC4630: IMAGE PROCESSING AND COMPUTER VISION

William E. G. Kvaale
University of Queensland
w.kvaale@uq.net.au
s46301303

April 24, 2020

ABSTRACT

In this assignment we will look into the Hough Transform for the detection of lines and circles in images. Furthermore, we will attempt to find the cross-sectional area of a beating heart given an MRI Image Sequence. Methods investigated are Snakes (Active contour model) [1], Sklansky [2], Viterbi [3], thresholding and morphology.

Hough Transform

Theory

The Hough Transform is a technique that can be used to parametric curves. In this assignment we will look on one of the simpler cases of detecting lines and circles.

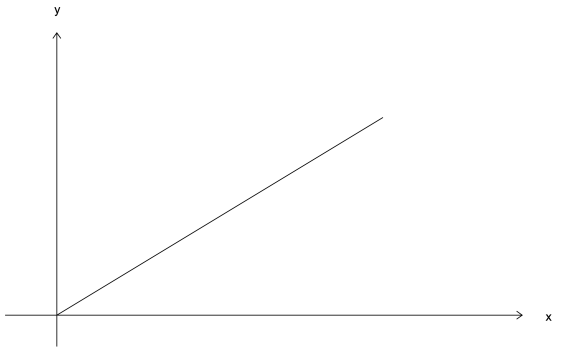

Figure 1: Image Space

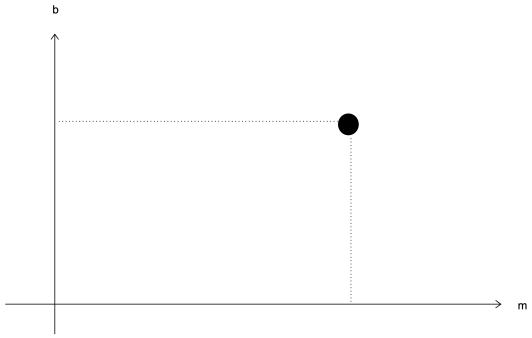

Figure 2: Parameter Space

A line can be represented by $y = m_0 x_i + b_0$ as seen in the left, in Figure 1. By looking at the problem from parameter space in Figure 2, we see the same line represented as a point. What has happened here it that we now see the $b_0 = -m_0 x + y_0$. The given figure is in no way exact, and only meant as a simple visual representation of the problem.

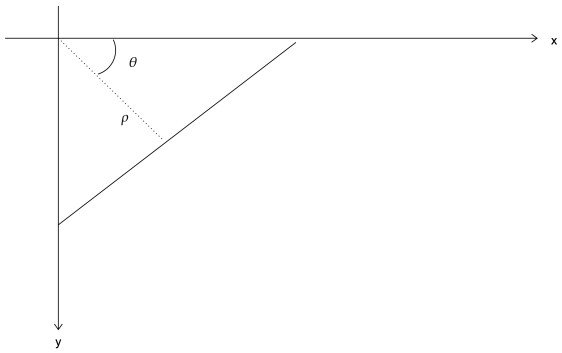

Figure 3: Image Space (x,y)

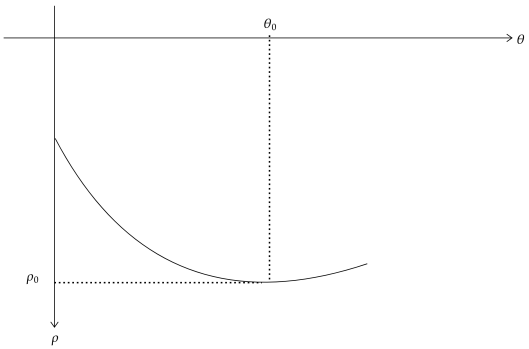

Figure 4: Hough Space (ρ, θ)

Due to the fact that a straight line $y = mx + b$ can be represented by infinitely many different versions by manipulating m and b. Therefore one uses the *Hesse Normal Form*, $\rho = xcos\theta + ysin\theta$. Here ρ is the shortest line from origo to the straight line, θ is the angle between the x-axis and ρ.

This is implemented in OpenCV's Hough Line Transform [4], which I used to achieve my results. This methods uses an 2D array for the accumulator, which will hold the distinct values for ρ and θ. The accumulator works in the way that it accumulate up the votes from the (ρ, θ) pairs found from detecting lines in image space. The higher the votes, the more likely it is that there exists a line. See Figure 5.

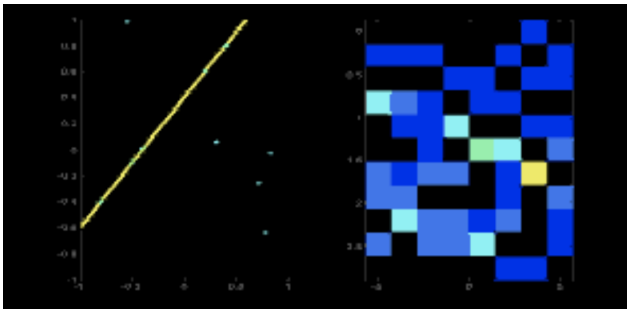

Figure 5: Illustrating Hough Transform Accumulator [5]

Method for Line Detection

The method involves a basic Gaussian blur, a Canny Edge Detector and the usage of OpenCV's Probabilistic Hough Lines function to find lines in images.

1. Select Region Of Interest using `cv.selectROI()` [6]
2. Convert image to grayscale
3. Blur using a Gaussian
   (a) $7x7$ kernel
   (b) $sigma_X = 0$
4. Use Canny Edge
   (a) $Hysteresis_{lower} threshold = 50$
   (b) $Hysteresis_{upper} threshold = 200$
   (c) $Aperturesize = 3$
5. Detect lines using a Probabilistic Hough Line Transform
6. Backproject the line onto the original image

Results for Line Detection

The method described in the previous subsection produced the following output.

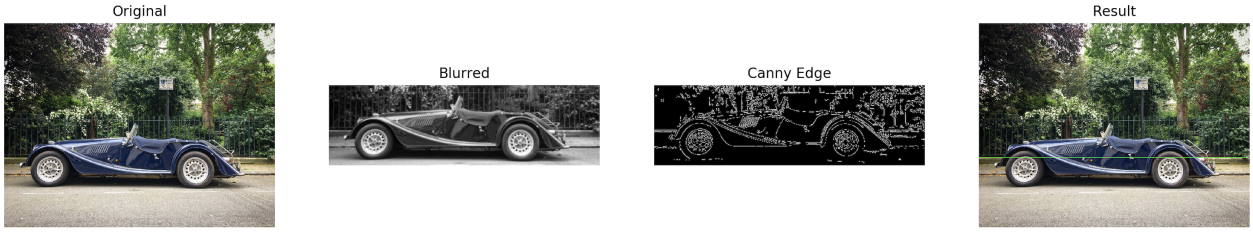

Figure 6: Hough Transform Lines Results

A solution utilizing binary thresholding, and another with Otsu's was attempted, buy yielded no better results. Hence the Ockham's razor. The Gaussian blur aided in the challenge of having two lines detected on the edging of the concrete, beneath the metal fence. This solution is based on the Probabilistic Hough Transform from OpenCV, which is based on *Robust Detection of Lines using the Progressive Probabilistic Hough Transform* [7].

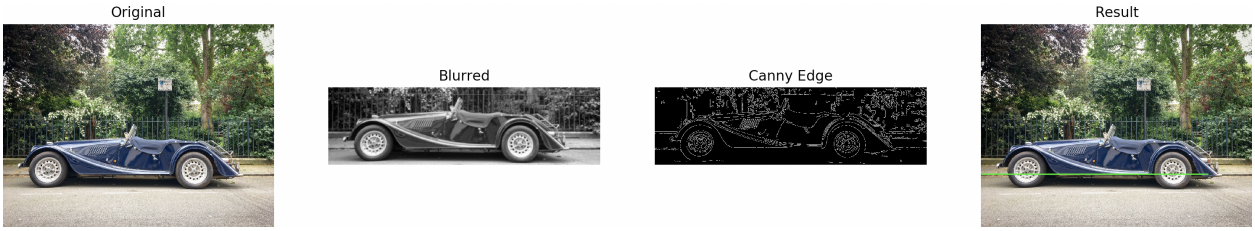

Figure 7: Hough Transform Lines Fail

When the normal Hough Transform is used to detect lines, we see that this implementation fails to find the requested edging of the concrete. A comparison of the two outcomes is shown in Figure 8 and Figure 9.

Figure 8: Probabilistic Hough Transform

Figure 9: Standard Hough Transform

Method for Circle Detection

1. Select Region Of Interest using `cv.selectROI()` [6]
2. Convert image to grayscale
3. Blur using a Gaussian
   (a) $7x7$ kernel
   (b) $sigma_X = 0$
4. Detect lines using Hough Gradient Method [8]
5. Backproject the circles onto the original image

Results for Circle Detection

Figure 10: Detection of Rims

Figure 11: Detetcion of tires

This method had good accuracy when it comes to the rims, however, when attempting to detect the tires by adjusting the radius did not yield ideal results. Adjusting the gamma, Canny edge detection, applying binary, Otsu's and inverted binary thresholding was also attempted. One challenge seen in detecting the tires was the immediate surroundings of the tires. There is no huge contrasts, and the OpenCV method used for finding the circles rely on a Hough Gradient Method, which again uses the edges' gradient information. Hence, it is challenging to detect the tires more accurately, without any more prior preprocessing addressing the issue.

Medical Image Segmentation

Theory

The proposed methods for this part of the assignment were snakes, viterbi, thresholding and morphology. Each of them were used, except viterbi.

When approaching the challenge of detecting the inner walls, the initial thought of methods was a Convex Hull method. OpenCV's `cv.convexHull()` [9] implements the Sklansky [2] algorithm. It is not guaranteed to find the hull, if it exists, and is therefore an *incorrect* solution. Nevertheless, it finds the inner walls.

For the outer walls, the initial thought was a Graham Scan, or a "Gift Wrapping Algorithm". The challenge here was to define the points it would wrap, or implement a solution which could be initialized without any points. This led me into researching Snakes. Scikit-Image had a implementation [10]. By utilizing this, the outer walls, and inner, was detected.

Method for segmenting inner walls

1. $ROI \leftarrow$ Select Region Of Interest using `cv.selectROI()` [6]
2. $Gray \leftarrow$ Convert ROI to grayscale
3. $Blurred \leftarrow$ Blur $Gray$ using a Gaussian
   (a) $3x3$ kernel
   (b) $sigma_X = 0$
4. $Threshold \leftarrow$ Apply Otsu Thresholding
5. Apply Morphological Operations
   (a) $Kernel \leftarrow 3x3$ rectangular structuring element
   (b) $Dilated \leftarrow Threshold \ominus kernel$ for 3 iterations
   (c) $Closed \leftarrow Dilated \bullet kernel$ for 3 iterations
6. Find Inner Walls by Convex Hull Method
   (a) $Input \leftarrow Closed$
   (b) $Output \leftarrow$ Segmentation of inner walls

Results using Convex Hull

The results of all images can be seen in the appendix, in Figure 17.

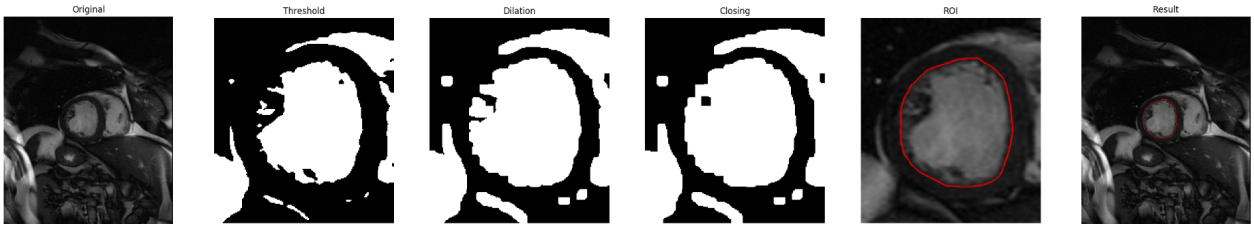

Figure 12: Hough Transform Lines Results

As seen in Figure 12, it clearly outlines the inner walls of the heart. The cross-sectional area of the heart is plotted in the graph in Figure 13 below.

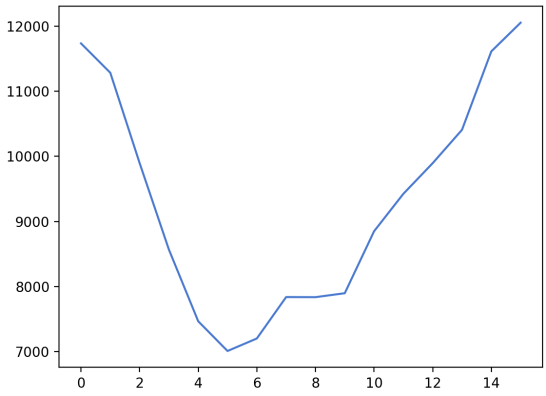

Figure 13: Convex Hull: Cross-sectional area of the heart

There is a slight "jagger" in the in the minimum of the graph, which might be due to the way a heart beats. This is beyond the domain knowledge of the writer.

Method for segmenting outer walls

1. $Gray \leftarrow$ Convert image to grayscale
2. $Blurred \leftarrow$ Blur $Gray$ using a Gaussian
   (a) $3x3$ kernel
   (b) $sigma_X = 0$
3. Apply Otsu Thresholding ($Threshold$)
4. Apply Morphological Operations
   (a) $Kernel \leftarrow 3x3$ rectangular structuring element
   (b) $Dilated \leftarrow Threshold \ominus kernel$ for 3 iterations
   (c) $Closed \leftarrow Dilated \bullet kernel$ for 3 iterations
5. Find Inner and Outer Walls by Snakes Active Contour Model
   (a) $Input \leftarrow Closed$
   (b) Snake: Active Contour Model hyperparameters
      i. $\alpha = 0.015$
      ii. $\beta = 1$
      iii. $\gamma = 0.1$
   (c) $Output \leftarrow$ Segmentation of inner walls and outer walls

Results using Snakes

The results of all images can be seen in the appendix, in Figure 18.

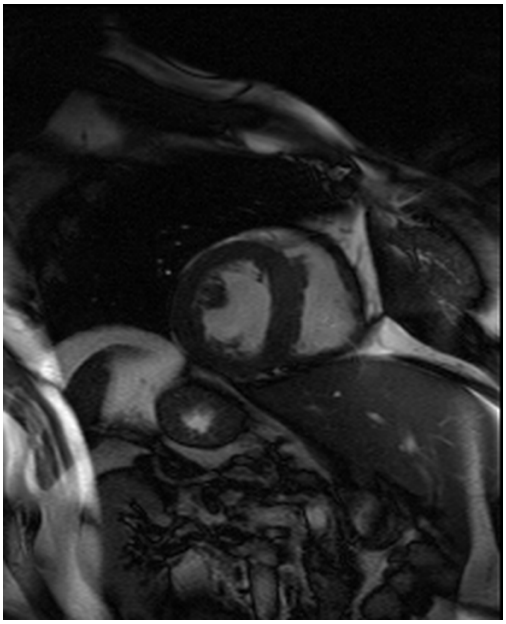

Figure 14: Sample MRI Heart Image

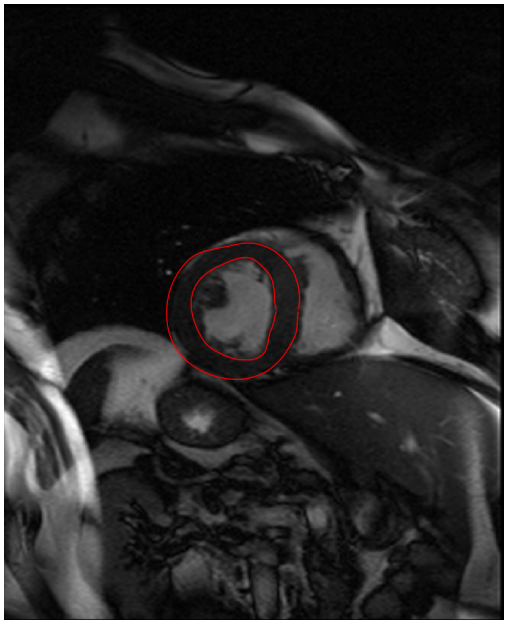

Figure 15: Snake Active Contour Model Applied

For the outer walls, the hyperparameters could have been adjusted to not be so sensitive to change, to be even more precise. This can be achieved by tampering with the snake length (α) and snake smoothness (β).

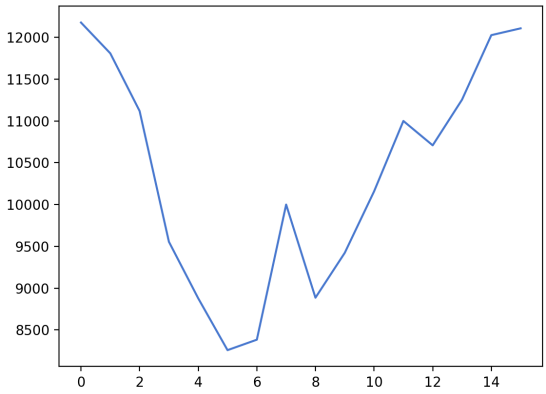

Figure 16: Snakes: Cross-sectional area of the heart

The graph produced by the Snake algorithm is even more "jaggered". This might be due to a artifact of the implementation, as the writer can not see any other implications to why this would be.

Summary

In this assignment we have accomplished the task of detecting the lines and circles using Probabilistic Hough Transforms and standard Hough Transform. We have also segmented the inner- and outer walls of the heart from an MRI image, and produced a graph to show the cross-sectional area for each image.

The software used is mainly Python (3.8.1), OpenCV (4.1.2). NumPy (1.18.3) has been used for mathematical computations on arrays and matrices, and PyPlot (matplotlib 3.2.1) has been used for plotting the graphs.

All the code can be found in the Code section of the appenid [], and in this Github repository.

References

[1] D. Terzopoulos M. Kass A. Witkin. "Snakes: Active contour models". In: *International Journal of Computer Vision* 1.4 (1988), pp. 321–331.

[2] J. Sklansky. "Finding the convex hull of a simple polygon". In: *Pattern Recognition Letters* 1.2 (1982), pp. 79–83.

[3] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269.

[4] OpenCV. *OpenCV Hough Line Transform*. URL: https://docs.opencv.org/4.1.2/d6/d10/tutorial_py_houghlines.html (visited on 04/20/2020).

[5] Amos Storkey. *Hough Transform*. URL: http://homepages.inf.ed.ac.uk/amos/hough.html (visited on 04/23/2020).

[6] *OpenCV SelectROI*. URL: https://docs.opencv.org/4.1.2/d7/dfc/group__highgui.html#ga8daf4730d3adf7035b6de9be4c469af5 (visited on 04/20/2020).

[7] Jiri Matas, C. Galambos, and J. Kittler. "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform". In: *Computer Vision and Image Understanding* 78 (Apr. 2000), pp. 119–137. DOI: 10.1006/cviu.1999.0831.

[8] OpenCV. *OpenCV Hough Circle Transform*. URL: https://docs.opencv.org/master/da/d53/tutorial_py_houghcircles.html (visited on 04/24/2020).

[9] OpenCV. *OpenCV Convex Hull*. URL: https://docs.opencv.org/4.1.2/d3/dc0/group__imgproc__shape.html#ga014b28e56cb8854c0de4a211cb2be656 (visited on 04/19/2020).

[10] Scikit-Image. *Scikit-Image Active Contour Model*. URL: https://scikit-image.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.active_contour (visited on 04/21/2020).

Appendix: Figures

Figure 17: Convex Hull on MRI images

Figure 18: Snakes Active Contour on MRI images

Appendix: Code

Task 1

```
# -*- coding: UTF-8 -*-
import os
import utils # local module
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

```
7
8  CAR_FILEPATH = os.path.abspath("images/morgan.jpg")
9
10
11 def hough_transform_lines(image, prob=True):
12     orig_img = np.copy(image)
13     # Select Region of Interest
14     ROI = cv.selectROI("ROI", image)
15     point1, point2 = (ROI[0], ROI[1]), (ROI[2], ROI[3])
16     img = utils.get_crop(image, point1, point2)
17     # Mapping to project back to original from ROI
18     dx = point1[0]
19     dy = point1[1]
20
21     # Preprocess image
22     gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
23     blurred = cv.GaussianBlur(gray, (7,7), 0)
24     edges = cv.Canny(blurred, 50, 200)
25
26     if prob:
27         lines = cv.HoughLinesP(edges, 1, np.pi/360, 220, minLineLength=img.shape[1]-20,
           ↪  maxLineGap=img.shape[1])
28         lines = lines[[[0]]]  # Fetch longest line
29
30         for line in lines:
31             x1, y1, x2, y2 = line[0]
32             print(f"x1, y1, x2, y2 -->", x1, y1, x2, y2)
33             # Update coords due to crop --> Project back on to original image
34             x1 += dx
35             x2 += dx
36             y1 += dy
37             y2 += dy
38             # Draw line back onto original image
39             cv.line(image, (x1, y1), (x2, y2), (0, 255, 0), 3)
40
41     else:
42         lines = cv.HoughLines(edges, 7.0, np.pi/180, 100)
43         for rho, theta in lines[0]:
44             alpha = np.cos(theta)
45             beta = np.sin(theta)
46             x0 = alpha * rho
47             y0 = beta* rho
48             gamma = 1000
49             x1 = int(x0 + gamma * (-beta)) + dx
50             x2 = int(x0 - gamma * (-beta)) + dx
51             y1 = int(y0 + gamma * (alpha)) + dy
52             y2 = int(y0 - gamma * (alpha)) + dy
53
54             cv.line(image, (x1, y1), (x2, y2), (0, 255, 0), 3)
```

```
55
56      plt.axis('off')
57      plt.imshow(image)
58      plt.show()
59      return image
60
61  def hough_transform_circles(image):
62      orig_img = np.copy(image)
63      # Select Region of Interest
64      ROI = cv.selectROI("ROI", image)
65      point1, point2 = (ROI[0], ROI[1]), (ROI[2], ROI[3])
66      # Mapping to project back to original from ROI
67      dx = point1[0]
68      dy = point1[1]
69      # Prepare image
70      img = utils.get_crop(image, point1, point2)
71      gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
72      blurred = cv.GaussianBlur(gray, (7, 7), 0)
73
74      # Detect lines
75      rims = cv.HoughCircles(image=blurred,
76                             method=cv.HOUGH_GRADIENT,
77                             dp=1,
78                             minDist=(blurred.shape[0]/64),
79                             param1=100,
80                             param2=62,
81                             minRadius=52,
82                             maxRadius=60)
83      if rims is not None:
84          rims = np.uint16(np.around(rims))[0,:]
85          for i in rims:
86              r = i[2]
87              # Draw circle where HoughCircles were detected
88              cv.circle(image, (i[0] + dx, i[1] + dy), radius=r, color=(0, 255, 0),
                ↪  thickness=3)
89              # Draw center of circle as dot
90              cv.circle(image, (i[0] + dx, i[1] + dy), radius=1, color=(255, 0, 0),
                ↪  thickness=3)
91      else:
92          print("No rims found.")
93
94      plt.axis('off')
95      plt.imshow(image)
96      plt.show()
97      return image
98
99
100 def main():
101     morgan = utils.load_single_image(CAR_FILEPATH)
```

```
    hough_transform_lines(morgan, prob=True)
    hough_transform_circles(morgan)


main()
```

Task 2

```
# -*- coding: UTF-8 -*-
import os
import utils
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from skimage.segmentation import active_contour


MRI_FILEPATH = os.path.abspath("images/MRI")
CAR_FILEPATH = os.path.abspath("images/morgan.jpg")
CAR_FILEPATH2 = os.path.abspath("images/MORGAN_CROP.jpg")


def get_cross_sectional_area(orig, thresh, dx, dy, draw=True):
    contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
    updated_contours = sorted(contours, key=cv.contourArea, reverse=True)[0].reshape(-1,
    ↪  2)

    hull = cv.convexHull(updated_contours, False)
    area = cv.contourArea(hull)

    if draw:
        cv.drawContours(orig, [hull], -1, (255, 0, 0), 1, cv.LINE_AA, offset=(dx, dy))
    return area


def snakes_algorithm(img, radius, alpha=0.015, beta=1, gamma=0.1):
    s = np.linspace(0, 2 * np.pi, 100)
    r = 358 + radius * np.sin(s)
    c = 269 + radius * np.cos(s)
    init = np.array([r, c]).T
    return active_contour(img, init, alpha=alpha, beta=beta, gamma=gamma,
    ↪  coordinates='rc', w_line=0, w_edge=1)


def method_one(image, point1, point2):
    orig_img = np.copy(image)
    img = utils.get_crop(image, point1, point2)
    img = cv.cvtColor(img.copy(), cv.COLOR_BGR2GRAY)
```

```
39
40      img = utils.adjust_gamma(img, 1.5)
41      img = cv.GaussianBlur(img, (3, 3), 0)
42      _, threshold = cv.threshold(img, 0, 255, cv.THRESH_OTSU)
43
44      kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
45      dilation = cv.dilate(threshold, kernel, iterations=3)
46      closing = cv.morphologyEx(dilation, cv.MORPH_CLOSE, kernel, iterations=3)
47
48      dx, dy = point1[0], point1[1]
49      orig_copy = np.copy(orig_img)
50      area = get_cross_sectional_area(orig_copy, closing, dx, dy)
51
52      utils.show_figures(images=
53                  [orig_img,
54                   threshold,
55                   dilation,
56                   closing,
57                   utils.get_crop(orig_copy, point1, point2),
58                   orig_copy],
59                  titles=
60                  ["Original",
61                   "Threshold",
62                   "Dilation",
63                   "Closing",
64                   "ROI",
65                   "Result"], save=False)
66
67      return orig_copy, area
68
69
70  def method_two(image):
71      # Preprocess image
72      orig_img = np.copy(image)
73      img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
74      img = utils.adjust_gamma(img, 1.5)
75      img = cv.GaussianBlur(img, (3, 3), 0)
76      _, threshold = cv.threshold(img, 0, 255, cv.THRESH_OTSU)
77
78      # Morphology
79      kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
80      dilation = cv.dilate(threshold, kernel, iterations=3)
81      closing = cv.morphologyEx(dilation, cv.MORPH_CLOSE, kernel, iterations=3)
82
83      # Inner wall
84      snake_in = snakes_algorithm(closing, 60, alpha=0.003, beta=3, gamma=0.1)
85
86      # Outer wall
87      snake_out = snakes_algorithm(closing, 80, alpha=0.003, beta=3, gamma=0.1)
```

```
88
89      # Cross-sectional area of hearts encircled by inner walls
90      c = np.expand_dims(snake_in.astype(np.float32), 1)
91      c = cv.UMat(c)
92      area = cv.contourArea(c)
93
94      # Plot
95      utils.show_figure_snakes(orig_img, snake_in, snake_out, save=False)
96
97      return snake_in, snake_out, area
98
99
100 def main():
101     # // Load MRI Heart Images
102     hearts = utils.load_images(MRI_FILEPATH)
103
104     # // Morphology & Sklansky Convex Hull
105     ROI = cv.selectROI("ROI", hearts[0])
106     p1, p2 = (ROI[0], ROI[1]), (ROI[2], ROI[3])
107     inner = []
108     inner_areas = []
109     for heart in hearts:
110         innerwall, a = method_one(heart, p1, p2)
111         inner.append(innerwall)
112         inner_areas.append(a)
113
114     plt.style.use('seaborn-muted')
115     plt.plot(inner_areas)
116     plt.show()
117
118     # // Snakes: Active Contour Model
119     inner = []
120     outer = []
121     outer_areas = []
122     for heart in hearts:
123         innerwall, outerwall, a = method_two(heart)
124         inner.append(innerwall)
125         outer.append(outerwall)
126         outer_areas.append(a)
127
128     plt.style.use('seaborn-muted')
129     plt.plot(outer_areas)
130     plt.show()
131
132
133 main()
```

Python Utils

```python
# -*- coding: UTF-8 -*-
import glob
import cv2 as cv
import numpy as np
from datetime import datetime
from matplotlib import pyplot as plt
import matplotlib.pyplot as mpl
mpl.rcParams['savefig.pad_inches'] = 0

def save_image(destination, images, label=""):
    for i in range(len(images)):
        img = images[i]
        try:
            cv.imwrite(f"{destination}/out_car_{i}_{label}.jpg", cv.cvtColor(img,
            ↪  cv.COLOR_RGB2BGR))
            print(f"Saving number plate {i+1} \U0001F4BE")
        except IOError :
            return f"Error while saving file number: {i}"
    print(f"Succesfully saved {len(images)} images to {destination}")


def load_single_image(filepath):
    img = cv.imread(filepath, cv.IMREAD_COLOR)
    img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    return img


def load_images(filepath, extension="png"):
    images = []
    for file in sorted(glob.glob(f"{filepath}/*.{extension}")):
        if file.lower().endswith(f".{extension}"):
            img = cv.imread(file, 1)
            img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
            images.append(img)
    return images

def save_figures(images, titles=[], rows=1):
    columns = len(images)
    _ = plt.figure(figsize=(64, 64/columns))
    for i in range(1, columns * rows + 1):
        plt.subplot(1, columns, i)
        if not titles[i-1]:
            plt.gca().set_title(f"Subplot_{i-1}", fontsize=32)
        plt.gca().set_title(titles[i-1], fontsize=32)
        plt.imshow(images[i-1], cmap='gray')
    plt.savefig(f"output/figure_{i}.png")
    print(f"Saved a figure \U0001F4BE")
```

```
47
48
49  def show_figures(images, titles=[], rows=1, folder="MRI_Convex", filename="figure",
    ↪  save=False):
50      columns = len(images)
51      fig = plt.figure(figsize=(32, 32/columns))
52      for i in range(1, columns*rows+1):
53          fig.add_subplot(rows, columns, i)
54          if not titles[i-1]:
55              plt.gca().set_title(f"Subplot_{i-1}")
56          plt.gca().set_title(titles[i-1])
57          plt.axis('off')
58          plt.imshow(images[i-1], cmap='gray')
59      if save:
60          now = datetime.now()
61          plt.savefig(f"{folder}/{filename}_{now}.png")
62          print(f"Saved a figure \U0001F4BE")
63      plt.show()
64
65
66  def show_figure_snakes(orig_img, inner, outer, save=False):
67      fig, ax = plt.subplots(figsize=(10, 10))
68      ax.imshow(orig_img)
69      ax.plot(inner[:, 1], inner[:, 0], '-r', lw=1)
70      ax.plot(outer[:, 1], outer[:, 0], '-r', lw=1)
71      ax.set_xticks([]), ax.set_yticks([])
72      plt.show()
73      if save:
74          now = datetime.now()
75          fig.savefig(f"output/MRI_Snakes/snakes_{now}.jpeg")
76
77
78  def show_images_snakes(images, n_row=4, n_col=4):
79      fig, axs = plt.subplots(n_row, n_col, figsize=(10, 10))
80      axs = axs.flatten()
81      for img, ax in zip(images, axs):
82          ax.imshow(img)
83          ax.get_xaxis().set_visible(False)
84          ax.get_yaxis().set_visible(False)
85          ax.set_aspect('equal')
86      plt.autoscale(tight=True)
87      plt.subplots_adjust(wspace=0, hspace=0)
88      plt.tight_layout()
89      plt.show()
90      fig.savefig(f"output/MRI_Snakes/snakes_grid.pdf", bbox_inches='tight',
        ↪  transparent=True, pad_inches=0)
91
92
93  def get_crop(frame, point1, point2):
```

```
94      img = frame.copy()
95      return img[point1[1]:point1[1]+point2[1], point1[0]:point2[0]+point1[0]]
96
97
98  def adjust_gamma(image, gamma=1.0):
99      # source: https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/
100     img = image.copy()
101     inverted_gamma = 1.0 / gamma
102     look_up_table = np.array([((i / 255.0) ** inverted_gamma) * 255
103                       for i in np.arange(0, 256)]).astype("uint8")
104     return cv.LUT(img , look_up_table)
```