# Image processing via model matching using Python.

Sathishkumar Matheswaran, 255667
Md. Abdullah al Kamran Ripon, 255630

## Short Description:

"Model_Matching" is the Main file.

"Task11", "Task22" and "Task33" are the required files for the Main file to execute.

At first necessary image modules are imported from python library. In this work *numpy, os, sys, PIL, math, matplotlib and pyplot are* used for image processing.

```
import numpy
import numpy as np
from matplotlib import pyplot as pp
import matplotlib.image as img
import os
import sys
from PIL import Image
import numpy
import math
```

After that program is automatically looks for images available in the same directory where program is saved. This program is design to process multiple images at the same time.

```
for infile in os.listdir('.'):
    if infile.endswith('.png'):
        matrix = img.imread(infile)
        m = Image.open(infile)
        fn, ftext = os.path.splitext(infile)
        m1 = m.size[0]  # width of the image (x-axis)
        m2 = m.size[1]  # height of the image (y-axis)
        print "Image Size:", m.size
```

After loading all images, next step is to define image area. For that three coordinates must find out in the image, which will define the image area. For selecting coordinate points, program will look for nonzero points within the image. For that a range is defined to find out right coordinate points. Three loops used to find the co-ordinate points.

```
        ######################################################################
        # Finding non-zero pixel value to get the co-ordinate values (Centre)
        ######################################################################
        a = []
        for j in range(0, m2 / 3):
            for i in range(m1 / 3, 3 * m1 / 4):
                if sum(m.getpixel((i, j))) >= 255:
                    a.append((i, j))
                    break
        a = a[0]
        [C1, C4] = a

        ######################################################################
        # Finding non-zero pixel value to get the cü-ordinate values (LEFT)
        ######################################################################
        b = []
        for i in range(0, m1 / 4):
            for j in range(m2 / 2, 4 * m2 / 5):
                if sum(m.getpixel((i, j))) >= 30:
                    b.append((i, j))
                    break
        b = b[0]
        [C2, C5] = b


        ######################################################################
        # Finding non-zero pixel value to get the cü-ordinate values (RIGHT)
        ######################################################################
        c = []
```

```
        for i in range(12 * m1 / 13, m1/3, -1):
            for j in range(12 * m2 / 13, m2 / 3, -1):
                if sum(m.getpixel((i, j))) >= 30:
                    c.append((i, j))
                    break
        c = c[0]
        [C3, C6] = c
        print "Detected Co-ordinates values :",'centre',(C1,C4),'Left',
(C2,C5),'Right', (C3,C6)
```

After defining the image area, source code will scan throughout the image area and will get best available pixel values by adjusting its size and shape based on the 4 neighbouring pixel values.

```
        a = 0
        for i in range(-4, 5):
            for j in range(-4, 5):
                Cx = C1
                Cx1 = C2 + i
                Cx2 = C3 + i
                Cy = C4 + j
                Cy1 = C5 + j
                Cy2 = C6 + j
```

Above code scan throughout the image and will get best available pixel values. After having all best possible pixel values, source code will start calculating angle and radius of a mask. This mask is necessary to crop the image area, area where image has best information for further processing. Below code calculate the amount of angle should the image mask have.

```
def Angle_Radius(Cx, Cy, Cx1, Cy1, Cx2, Cy2):
#Setting Desired co-ordinates as origin
    Cx1_mod = Cx - Cx1
    Cy1_mod = Cy - Cy1
    Cx2_mod = Cx - Cx2
    Cy2_mod = Cy - Cy2

   #   ANGLE Calculation

    Angle1 = math.atan2(Cy1_mod - 0, Cx1_mod - 0)
    Angle2 = math.atan2(Cy2_mod - 0, Cx2_mod - 0)
    Angle = abs(Angle2) - abs(Angle1)
    Angle11 = math.degrees(Angle1)
    Angle22 = math.degrees(Angle2)
```

At the same time coordinate values used to calculate the radius of the image. Below code used to calculate radius of the mask.

```
#   Radius Calculation
    Radius1 = ((Cx - Cx1) ** 2 + (Cy - Cy1) ** 2)
    Radius1 = math.sqrt(Radius1)
    Radius2 = ((Cx - Cx2) ** 2 + (Cy - Cy2) ** 2)
    Radius2 = math.sqrt(Radius2)
    if Radius1 > Radius2:
        Radius = math.floor(Radius1)
    else:
        Radius = math.floor(Radius2)
    return Radius, Angle11, Angle22
```

After calculating angle and radius, an image mask is created using blow code.

```
def Pie_mask(shape, centre, Radius, Angle):
    x, y = np.ogrid[:shape[0], :shape[1]]
    cx, cy = centre
    Anglemin, Anglemax = np.deg2rad(Angle)

    if Anglemax < Anglemin:
        Anglemax += 2 * np.pi

    r2 = (x - cx) * (x - cx) + (y - cy) * (y - cy)
```

```
    theta = np.arctan2(x - cx, y - cy) - Anglemin
    theta %= (2 * np.pi)
    cirmask = r2 <= Radius * Radius
    anglemask = theta <= (Anglemax - Anglemin)
    return cirmask * anglemask
```

After that, image mask was imported and crop the image where best image information available. Below code is used to crop the image.

```
from Task33 import Pie_mask
centre = (Cx, Cy)
Angle = (Angle11, Angle22)
mask = Pie_mask(matrix.shape, (Cy, Cx), Radius, (-Angle11, -Angle22))
a = matrix[mask]
a1 = np.sum(a)
print "a1 values are", a1
if a1 > a:
    a = a1
    best_Cx = Cx
    best_Cy = Cy
    best_Cx1 = Cx1
    best_Cx2 = Cx2
    best_Cy1 = Cy2
    best_Cy2 = Cy2
```

## RESULTS

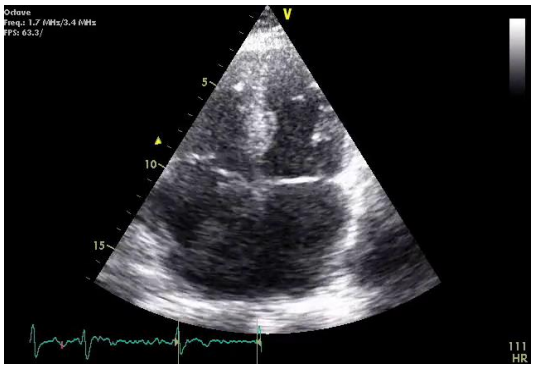

Fig 2: Original Image

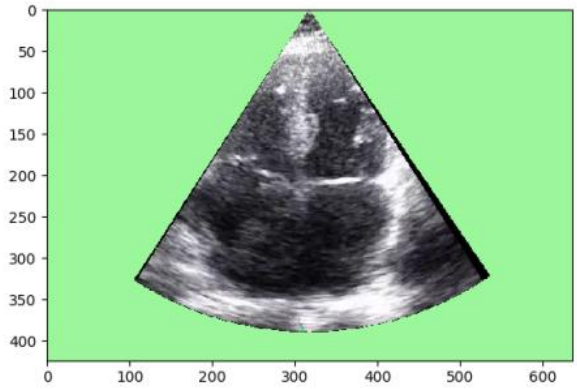


Fig 2: Resulted Image

## DISCUSSION

Main goal of this work was to develop a method to find out best image information of ultrasound images based on model matching. To do that image area is defined by finding nonzero pixels on ultrasound images by defining ranges within the image. After that, within that area image was scanned to find out available best pixels values. This pixel values used to calculate angle and radius of a mask to crop the image where the best image information remain. Finding right image area is important to get good result. If any part of the image out of image border it might bring poor results.