Digital Manufacturing (MECE 4606)

Assignment - 3

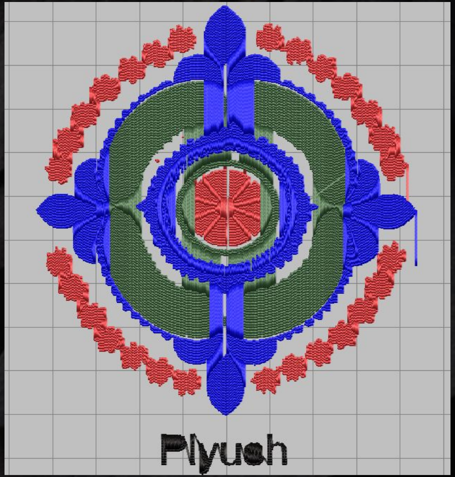
Programmable Embroidery

By Piyush Pradhan (ppp2132)

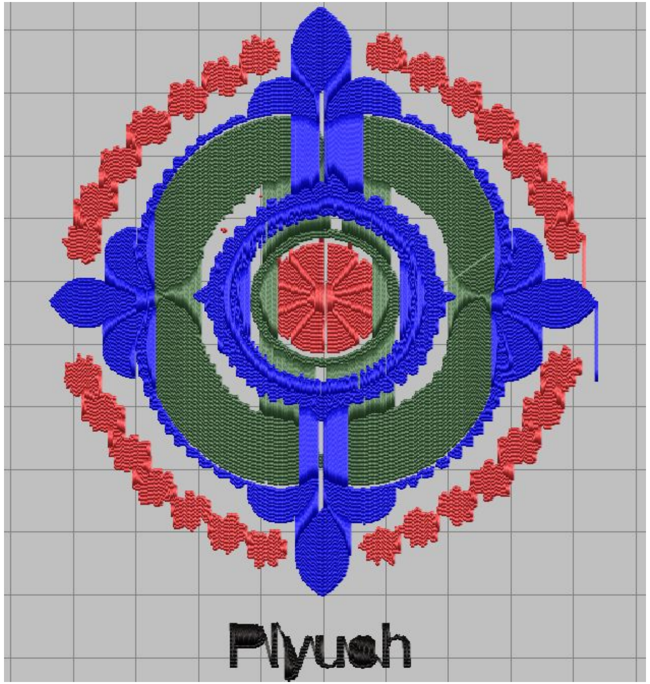
14th March, 2023

Grace hours used: 30 hrs

Total Grace hours left: 51 - 30 = 21



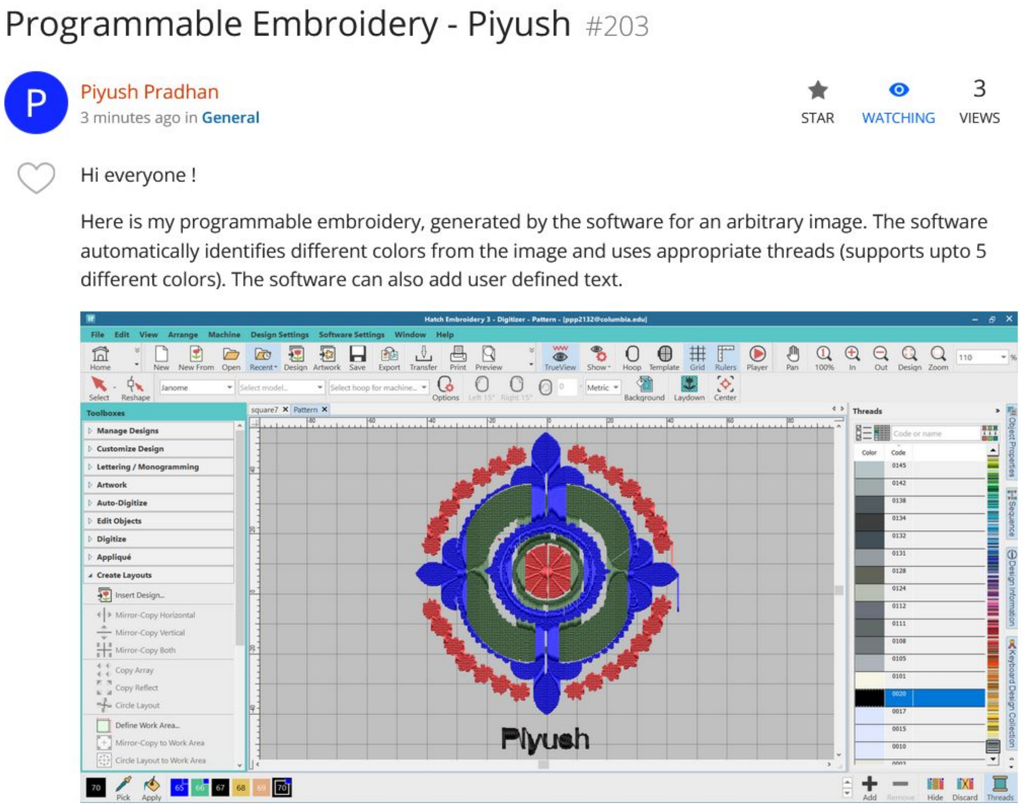
Python Code: Complexity



Code Parameters & Features:

1. An **image** of the required parameter is provided to the software, there is **no specific size requirements** for the image.
2. Software automatically detects, embroidery paths; in total 5 colors are used: **Red, Green, Blue, Yellow and Black**.
3. However, software can automatically reject specific colors if they are not present in the image. In the figure shown, yellow thread embroidery path is not generated as it is not required.
4. **User defined text** is embroidered at the bottom with choice of **font, color** and **position**.
5. Overstitching for each color to improve aesthetics.

Ed Post



Programmable Embroidery - Piyush #203

P Piyush Pradhan
3 minutes ago in General

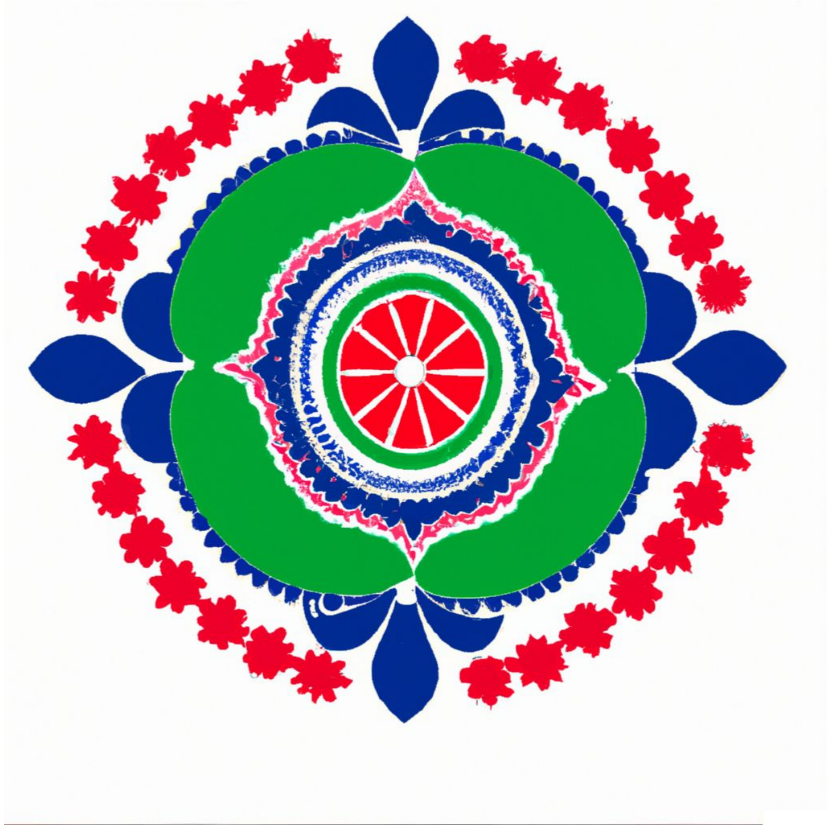
STAR WATCHING 3 VIEWS

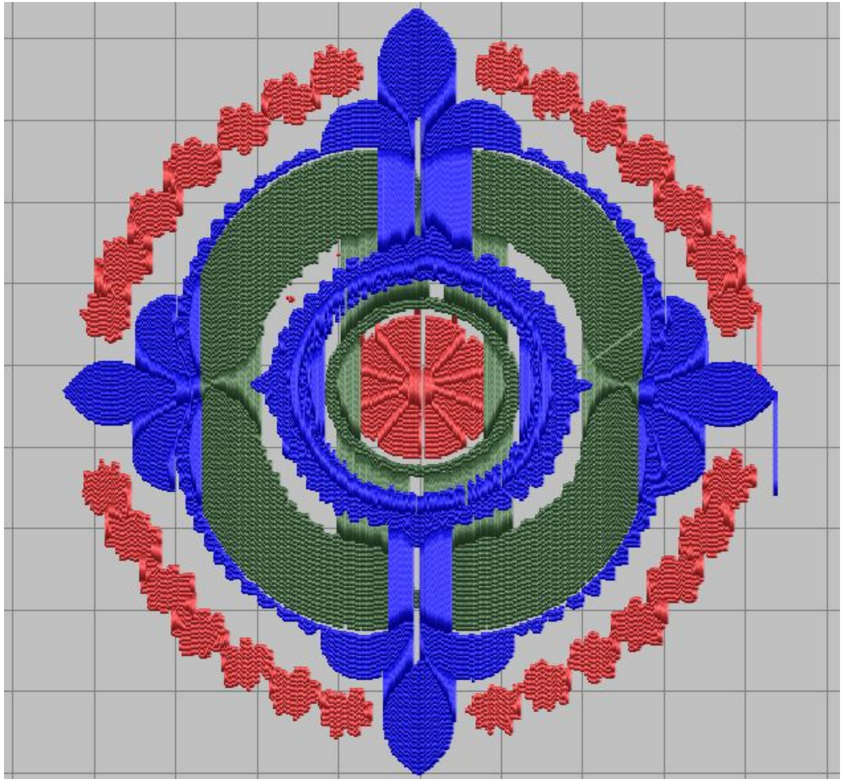
Hi everyone !

Here is my programmable embroidery, generated by the software for an arbitrary image. The software automatically identifies different colors from the image and uses appropriate threads (supports upto 5 different colors). The software can also add user defined text.

Hatch Embroidery 3 - Digitizer - Pattern - [ppp2132@columbia.edu]
File Edit View Arrange Machine Design Settings Software Settings Window Help
Home New New From Open Recent Design Artwork Save Export Transfer Print Preview TrueView Show Hoop Template Grid Rulers Player Pan 100% In Out Design Zoom 110
Select Reshape Janome Select model... Select hoop for machine... Options Left 15° Right 15° Metric Background Laydown Center
Toolboxes
Manage Designs
Customize Design
Lettering / Monogramming
Artwork
Auto-Digitize
Edit Objects
Digitize
Appliqué
Create Layouts
Insert Design...
Mirror-Copy Horizontal
Mirror-Copy Vertical
Mirror-Copy Both
Copy Array
Copy Reflect
Circle Layout
Define Work Area...
Mirror-Copy to Work Area
Circle Layout to Work Area
square7 Pattern
Piyush
Threads
Code or name
Color Code
0145
0142
0138
0134
0132
0131
0128
0124
0112
0111
0108
0105
0101
0020
0017
0015
0010
70 Pick Apply 65 66 67 68 69 70
Add Remove Hide Discard Threads
Object Properties Sequence Design Information Keyboard Design Collection

Comparison between actual image & software generated pattern





Final Video

https://youtu.be/ioHrVgRR3Dg

Appendix: Code Overview - Custom Text Generation

```
def check_black(r):
    if r > 10:
        return False
    else:
        return True

def generate_custom_text(text):
    print("Generating point map for
characters using Arial font")
    img = Image.new('RGB', (480, 480),
"white")
    d = ImageDraw.Draw(img)
    loc = os.getcwd()
    font = ImageFont.truetype("arial", 100)
    d.text((2, -10), text, fill=(0, 0,
0),font=font)
    fullpath =
os.path.join(loc,'letters\\letter-custom.png'
)
    img.save(fullpath, 'png')

    k_points = []
    filename = "letters/letter-custom.png"

    img = Image.open(filename)
    img = img.rotate(-90)
    img = np.asarray(img)
    height = np.shape(img)[0]
    width = np.shape(img)[1]
    channels = np.shape(img)[2]
```

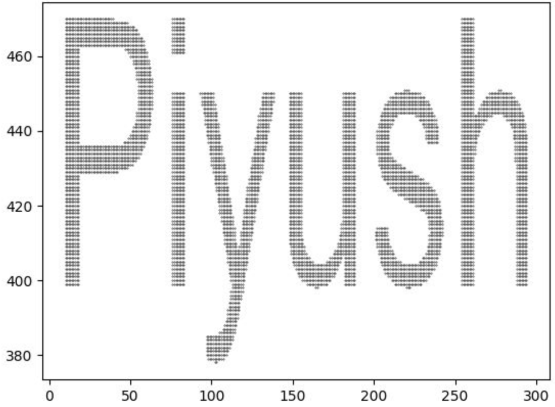
Appendix: Code Overview - Custom Text Generation

```
    for i in range(height):
        for j in range(width):
            is_black = 
check_black(img[i,j,0])
            if is_black:
                k_points.append(i)
                k_points.append(j)

    k_points = 
np.asarray(k_points,dtype=int).reshape(-1,2)
    print("Points generated")
    return k_points
```

The plot represents point map generated for custom text “Piyush”

1. The function generates embroidery points for the custom text.
2. The custom text is first saved as an image using Python Image Library.
3. The custom image is then read pixel by pixel and the locations of black pixels are saved.



Appendix: Code Overview - Custom Image Generation

```
def check_pixel(r,g,b):
    pixel_color = 'null' # 'r', 'g', 'b'
    if r >= 200 and g <= 70 and b <= 70:
        pixel_color = 'r'
    if r >= 200 and g >= 200 and b <= 120:
        pixel_color = 'y'
    if r <= 70 and g >= 100 and b <= 70:
        pixel_color = 'g'
    if r <= 70 and g <= 70 and b >= 100:
        pixel_color = 'b'
    return pixel_color
```

```
def generate_points(filename,custom_text):
    print("Reading image data...")
    img = np.asarray(Image.open(filename))
    height = np.shape(img)[0]
    width = np.shape(img)[1]
    channels = np.shape(img)[2]

    r_points = []
    g_points = []
    y_points = []
    b_points = []

    for i in range(height):
        for j in range(width):
            pixel_color = 
check_pixel(img[i,j,0], img[i,j,1],
```

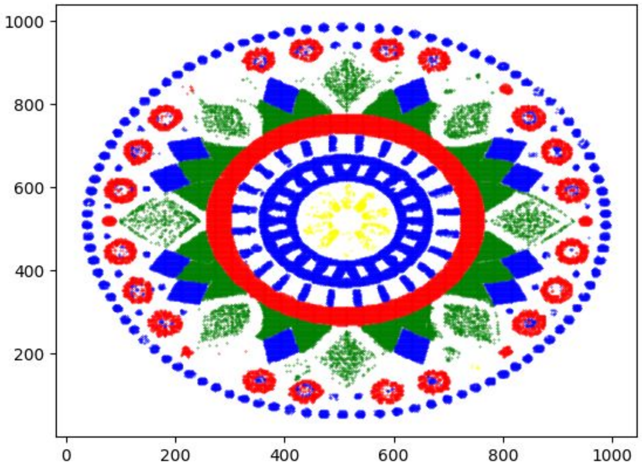
Appendix: Code Overview - Custom Image Generation

```
img[i,j,2])
        if pixel_color == 'r':
            r_points.append(i)
            r_points.append(j)
            continue
        if pixel_color == 'g':
            g_points.append(i)
            g_points.append(j)
            continue
        if pixel_color == 'b':
            b_points.append(i)
            b_points.append(j)
            continue
        if pixel_color == 'y':
            y_points.append(i)
            y_points.append(j)
```

```
    print("Image points generated")
    r_points =
np.asarray(r_points,dtype=int).reshape(-1,2)
    g_points =
np.asarray(g_points,dtype=int).reshape(-1,2)
    b_points =
np.asarray(b_points,dtype=int).reshape(-1,2)
    y_points =
np.asarray(y_points,dtype=int).reshape(-1,2)
    k_points =
generate_custom_text(custom_text)
    k_points[:,0] = k_points[:,0] + (width/2)
- 140
    k_points[:,1] = k_points[:,1] - 480
```

Appendix: Code Overview - Custom Image Generation

```
    height = height + 480
    width = max(width,640)
    return
height,width,r_points,g_points,b_points,y_points,k_points
```



1. The function generates embroidery points for the given image.
2. The image is read pixel by pixel and classified as Red, Green, Blue or Yellow by analyzing the RGB channel values.
3. Pixels not lying in this threshold (background pixels) are ignored.
4. To the left is the point map for a custom image provided to the software. The Red, Green, Blue and Yellow regions identified by the software are clearly visible.

Appendix: Estimated Marks

1. 10pts Cover page correct and complete
2. 10pts Report neatly organized and formatted
3. 10pt A parametric fractal shape embroidered
4. 10pt Complexity/Aesthetics of the best pattern
5. 10pt Quality of the stitch (over-stitching, wide stitches)
6. 10pt Number of input parameters in software interface
7. 10pt A description of the software you wrote – calculation steps, formulas, conditions.
8. 10pt User specified text embroidered
9. 10pt A fractal shape that is not a tree
10. 10pt Multiple threads used (at least two)
11. 10pt Multiple threads colors used (excluding bobbin thread)
12. 10pt Embroidery photo posted on Ed at least 24h day before the deadline (show screenshot)
13. 10pt Video of entire process, from entering design parameters to embroidering the pattern

Thank You