EE2703 WEEK 5 - Images and Animation

Dheeraj.B <EE21B032>

March 8, 2023

1 Images and Animation

1.1 Setting up the notebook for plotting

[1]:
```
%matplotlib notebook
import math
import IPython
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

We will define some utility functions that help us to draw shapes. These are basically used to generate x, y coordinates of whatever shape you are interested in. This is the most basic form of vector graphics that is useful for us

1.2 Utility functions to draw shapes

[2]:
```
def vertices(n): # Define vertices
    angles = np.linspace(0, 2*np.pi, n+1)[:-1]  # Angles for n-sided polygon
    x, y = np.cos(angles), np.sin(angles)  # x,y-coordinates of vertices
    return np.column_stack((x, y))

# side function for side length
def side(n):
    x = (2*np.pi)/n
    side_len = math.sqrt(2-2*np.cos(x))
    return side_len

# function for forming x,y arrays for a line
def line(x1, y1, x2, y2, l):
    x = np.linspace(x1, x2, l)
    if abs(x2 - x1) <= 1e-10:
        y = np.linspace(y1, y2, l)
        x = x2 * np.ones(l)
    else:
        m = (y2 - y1) / (x2 - x1)
        y = m * (x - x1) + y1
    return x, y
```

```
def polygon(n):
    a = vertices(n)
    l = side(n)
    x, y = [], []
    for i in range(n-1):
        m, p = line(a[i][0], a[i][1], a[i+1][0], a[i+1][1], int(1680/n))
        x.append(m)
        y.append(p)
    r, s = line(a[n-1][0], a[n-1][1], a[0][0], a[0][1], int(1680/n))
    x.append(r)
    y.append(s)
    x = np.concatenate(x)
    y = np.concatenate(y)
    return x, y

fig, ax = plt.subplots()
ln, = ax.plot([], [], 'green')

def init():
    ax.set_xlim(-1.2, 1.2)
    ax.set_ylim(-1.2, 1.2)
    return ln,

def update(frame):
    for i in range(5):
        if (frame <= i+1 and frame >= i):
            xa, ya = polygon(i+4)

            xb, yb = polygon(i+3)

            xdata, ydata = morph(xa, ya, xb, yb, frame-i)
            plt.pause(0.001)
            ln.set_data(xdata, ydata)
    for i in range(5,10):
         if i == 5:
            if (frame<=i+1 and frame>=i):
                xa, ya = polygon(12-i)
                xb, yb = polygon(13-i)

                xdata, ydata = morph(xa, ya, xb, yb, frame-i)
                plt.pause(0.001)
                ln.set_data(xdata, ydata)
            else :
                if (frame<=i+1 and frame>=i):
                    xa, ya = polygon(12-i)
                    xb, yb = polygon(13-i)
```

```
                    xdata, ydata = morph(xa, ya, xb, yb, frame-i)
                    plt.pause(0.001)
                    ln.set_data(xdata, ydata)
    return ln,
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

1.3 Morphing

[3]:
```
def morph(x1, y1, x2, y2, alpha):

    xm = alpha * x1 + (1-alpha) * x2
    ym = alpha * y1 + (1-alpha) * y2
    return xm, ym

ani = FuncAnimation(fig, update, frames=np.linspace(0, 10, 128),init_func=init,␣
 ↪blit=True, interval=70, repeat=True)
ani.save('out.gif')
plt.show()
```

```
MovieWriter ffmpeg unavailable; using Pillow instead.
```

We use the term morphing here to refer to converting one image into another. This does not by itself result in the creation of a new image, but what we do instead is to create a series of images in between the original and target images, so that they can be viewed as an animation. We first create a function that does a linear interpolation between two points, and use this to perform the interpolation from one to the other.