EE2703 - Week 5 - Images and Animation

Nitin Chandrachoodan <nitin@ee.iitm.ac.in>

February 14, 2023

1 Images and Animation

In many situations, you want to do more than just basic plotting of data. Either you may be dealing with 2-D images that need to be displayed, or you may want to get a time lapse animation of some process. Knowing how to build basic animations can be quite useful in these contexts.

Many of the techniques here are better employed directly as Python scripts. They can be run from the Jupyter notebook, but this involves a somewhat hacky mix of Python and Javascript, which does not usually feel as smooth as native animation.

1.1 Setting up the notebook for plotting

[ ]:
```
# Magic command below to enable interactivity in the JupyterLab interface
%matplotlib ipympl
# Some basic imports that are useful
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

Python with Matplotlib tends to use global variables for some of the plotting functionality. This is not a good idea in general - global variables mess up your workspace and can make it hard to keep track of what was changed at what point. However, it does make it easier to manage plots. Use with care.

[ ]:
```
fig, ax = plt.subplots()
xdata, ydata = [], []
ln, = ax.plot([], [], 'r')

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

def update(frame):
    # xdata.append(frame)
    # ydata.append(np.sin(frame))
    xdata, ydata = morph(xs, ys, xc, yc, frame)
    ln.set_data(xdata, ydata)
    return ln,
```

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. In general image manipulation requires a lot more advanced modeling.

```
[ ]: def circle(t):
         return np.cos(t), np.sin(t)

     def square(t):
         """Return x and y coordinates for a square in +/-0.5."""
         n4 = int(len(t) / 4)
         ts = np.linspace(-0.5, +0.5, n4)
         xs = np.concatenate([ts, 0.5*np.ones(n4), ts[::-1], -0.5*np.ones(n4)])
         ys = np.concatenate([xs[n4:], xs[:n4]])
         return xs, ys
```

1.2 Morphing

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.

```
[ ]: def morph(x1, y1, x2, y2, alpha):
         xm = alpha * x1 + (1-alpha) * x2
         ym = alpha * y1 + (1-alpha) * y2
         return xm, ym

     t = np.linspace(3*np.pi/4, -5*np.pi/4, 200)
     if len(t) % 4 != 0:
         raise BaseException("Number of points should be multiple of 4...")
     xc, yc = circle(t)
     xs, ys = square(t)
     # print(f"Square: {np.shape(xs)}")

     ani = FuncAnimation(fig, update, frames=np.linspace(0, 1, 128),
                         init_func=init, blit=True, interval=10, repeat=True)
     plt.show()
```

2 Assignment

An animation involving multiple polygons has been uploaded to Moodle. Your assignment is to try and recreate this animation in any way that you can. Explain your approach. Your final result may not be identical to what is given, but should be qualitatively similar.

2.1 Possible bonus tracks

As mentioned already, these are not specific problems for bonus marks, but if you want to work on these lines, you need to let me know your plan, and then implement them and document them appropriately. Keep in mind that one of the main expectations of bonus assignments is that they are “demo-worthy” - they should be of high quality in presentation.

- What has been described here is only “point morphing”. Line or polygon (triangle) based morphing gives much better results when trying to map one image into another. Implement any of these techniques and demonstrate on suitable examples.
- If you want to automatically morph a cat into a dog (say), you will probably need to identify suitable control points or triangles for the mapping. Can you do this automatically? There are libraries such as opencv that may help you with parts of what you need, but this will still require some work.