Creating intersecting images in matplotlib with imshow or other function

I have two 3-D arrays of ground penetrating radar data. Each array is basically a collection of time-lapse 2-D images, where time is increasing along the third dimension. I want to create a 3-D plot which intersects a 2-D image from each array.

I'm essentially trying to create a fence plot. Some examples of this type of plot are found on these sites:
http://www.geogiga.com/images/products/seismapper_3d_seismic_color.gif
http://www.usna.edu/Users/oceano/pguth/website/so461web/seismic_refl/fence.png

I typically use imshow to individually display the 2-D images for analysis. However, my research into the functionality of imshow suggests it doesn't work with the 3D axes. Is there some way around this? Or is there another plotting function which could replicate imshow functionality but can be combined with 3D axes?

python matplotlib 3d imshow

edited May 13 at 18:23
Artjom B.

asked Aug 13 '14 at 13:44
Blake Lytle

37.8k 15 42 64


8 3

It isn't much (you probably have to download the example `py` file to see what's going on) but it might get you started until one of us has the time to provide an answer: matplotlib.org/mpl_toolkits/mplot3d/tutorial.html#d-plots-in-3d – Schorsch Aug 13 '14 at 14:43

I have seen this Chaco demo github.com/enthought/chaco/blob/master/examples/demo/advanced/… adapted to display seismic data – MyCarta May 26 '15 at 16:55

2 Answers

If you're happy to contemplate using a different plotting library (ie not matplotlib) then it might be worth considering mayavi / tvtk (although the learning curve is a little steep). The closest I've seen to what you want is the scalar cut planes in http://wiki.scipy.org/Cookbook/MayaVi/Examples

The bulk of the documentation is at: http://docs.enthought.com/mayavi/mayavi/index.html

answered Aug 13 '14 at 20:57

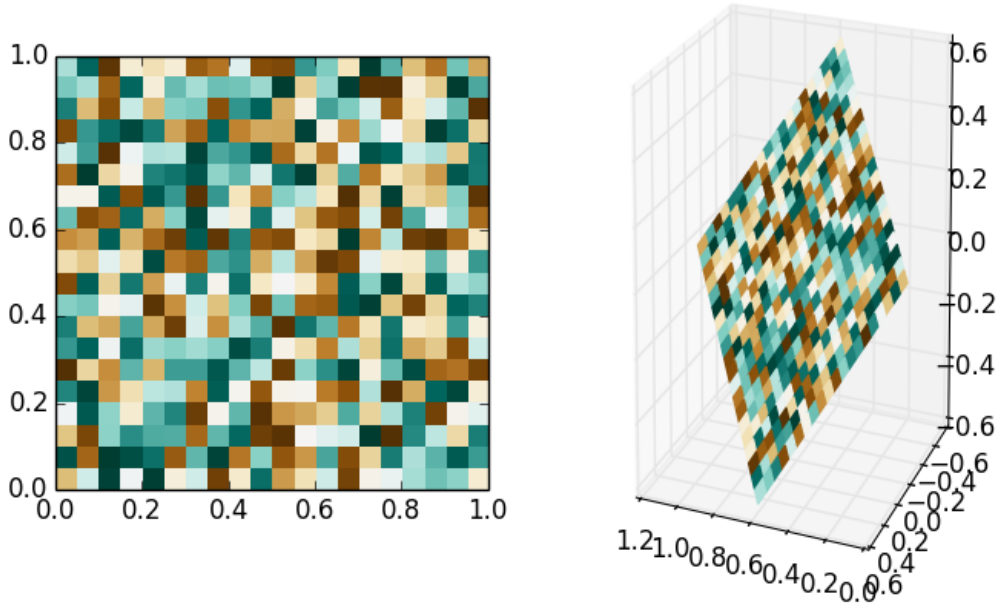

user488551
271 1 5

There might be better ways, but at least you can always make a planar mesh and color it:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# create a 21 x 21 vertex mesh
xx, yy = np.meshgrid(np.linspace(0,1,21), np.linspace(0,1,21))
```

```
# create some dummy data (20 x 20) for the image
data = np.random.random((20, 20))

# create vertices for a rotated mesh (3D rotation matrix)
X = np.sqrt(1./3) * xx + np.sqrt(1./3) * yy
Y = -np.sqrt(1./3) * xx + np.sqrt(1./3) * yy
Z = np.sqrt(1./3) * xx - np.sqrt(1./3) * yy

# create the figure
fig = plt.figure()

# show the reference image
ax1 = fig.add_subplot(121)
ax1.imshow(data, cmap=plt.cm.BrBG, interpolation='nearest', origin='lower', extent=
[0,1,0,1])

# show the 3D rotated projection
ax2 = fig.add_subplot(122, projection='3d')
ax2.plot_surface(X, Y, Z, rstride=1, cstride=1, facecolors=plt.cm.BrBG(data), shade=False)
```

This creates:



(Please note, I was not very careful with the rotation matrix, you will have to create your own projection. It might really be a good idea to use a real rotation matrix.)

Just note that there is a slight problem with the fence poles and fences, i.e. the grid has one more vertex compared to the number of patches.

The approach above is not very efficient if you have high-resolution images. It may not even be useful with them. Then the other possibility is to use a backend which supports affine image transforms. Unfortunately, you will then have to calculate the transforms yourself. It is not hideously difficult, but still a bit clumsy, and then you do not get a real 3D image which could be rotated around, etc.

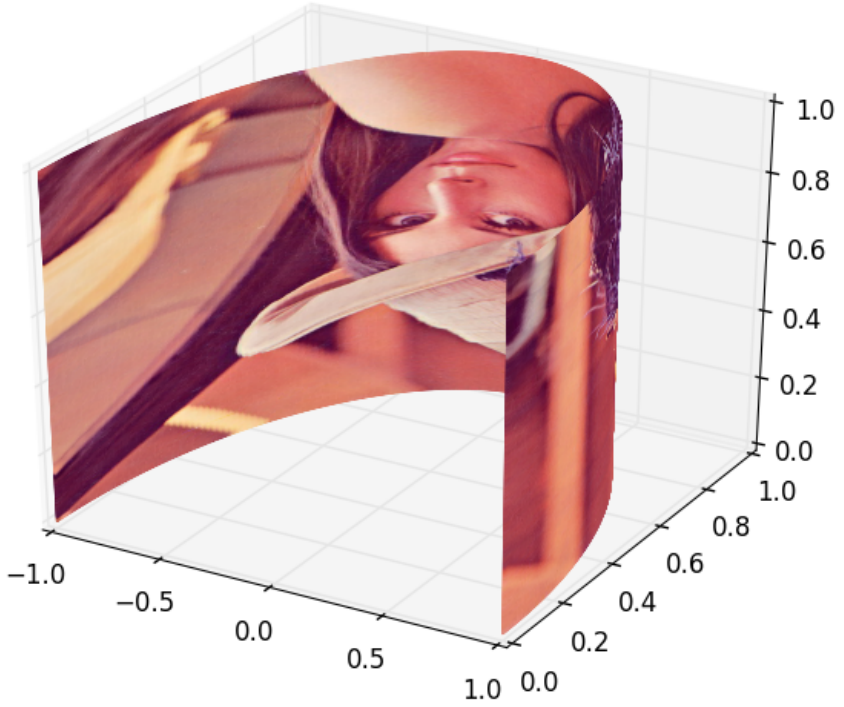
For this approach, see http://matplotlib.org/examples/api/demo_affine_image.html

Alternateively, you can use OpenCV and its `cv2.warpAffine` function to warp your image before showing it with `imshow`. If you fill the surroundings with transparent color, you can then layer images to get a result which looks like your example iamge.

Just to give you an idea of the possibilities of `plot_surface`, I tried to warp Lena around a semi-cylinder:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# create a 513 x 513 vertex mesh
xx, yy = np.meshgrid(np.linspace(0,1,513), np.linspace(0,1,513))

# create vertices for a rotated mesh (3D rotation matrix)
theta = np.pi*xx
X = np.cos(theta)
Y = np.sin(theta)
Z = yy

# create the figure
fig = plt.figure()

# show the 3D rotated projection
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
facecolors=plt.imread('/tmp/lena.jpg')/255., shade=False)
```

She indeed bends well, but all operations on the image are quite slow:



edited Aug 13 '14 at 20:33

answered Aug 13 '14 at 20:13

DrV

9,215 1 10 31