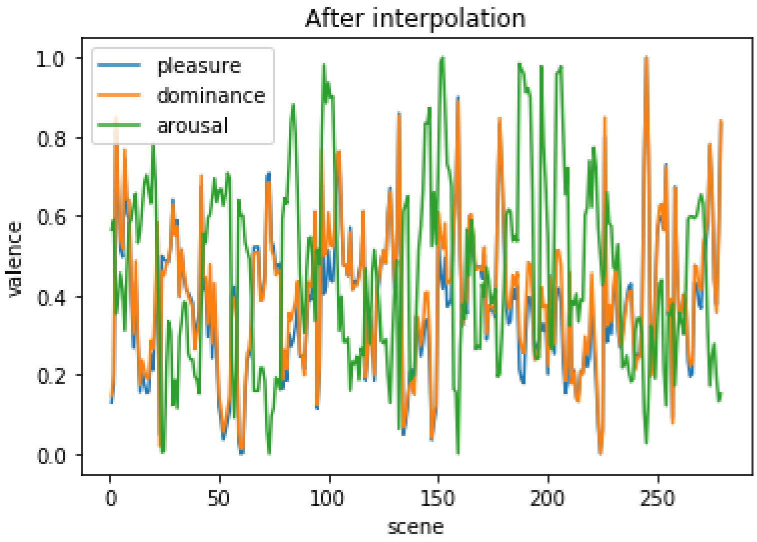
In [1]:
```
# import Section
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
from os import listdir
from os.path import isfile, join
import re
from matplotlib import pyplot as plt
```

In [2]:
```
def isjpg(filepath):
    return re.search(".jpg$", filepath)
```

In [3]:
```
# Function Section
def calculate_pad(brightness, saturation):
    p = 0.69*brightness + 0.22*saturation
    a = -0.31*brightness + 0.6*saturation
    d = 0.76*brightness + 0.32*saturation
    return [p,d,a]

def calculate_blur(img):
    return cv2.Laplacian(img, cv2.CV_64F).var()

def mean_brightness(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #convert it to hsv
    return np.mean(hsv[:,:,2])

def mean_saturation(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #convert it to hsv
    return np.mean(hsv[:,:,1])

def calculate_opticalFlow(img1, img2):
    f, axarr = plt.subplots(2,1)
    axarr[0].imshow(img1)
    axarr[1].imshow(img2)
    plt.show()
    prev = cv2.cvtColor(testEld[0], cv2.COLOR_BGR2GRAY)
    forward = cv2.cvtColor(testEld[1], cv2.COLOR_BGR2GRAY)
    mask = np.zeros_like(prev)
    mask[..., 1] = 255
    flow = cv2.calcOpticalFlowFarneback(prev, forward, flow=None, pyr_scale=0.5,
    magnitude, angle = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    return cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)[0]
```

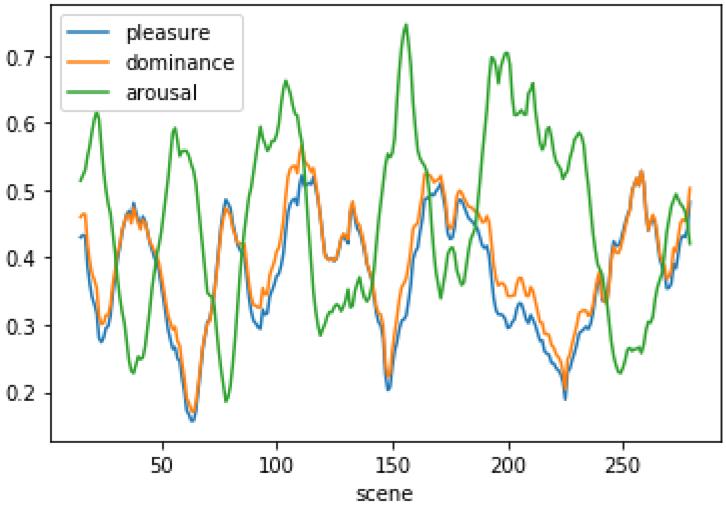
In [4]:

```
import glob
import re
from scipy.interpolate import interp1d

mypath = "data/scenes/incredibles"

files = [f for f in listdir(mypath) if isjpg(join(mypath, f))]
pad_data = []

for fname in files:
    src = cv2.imread(join(mypath, fname),1)
    if(src is None):
        continue
    gx = '^[^-]*-([^-]*).*'
    p = re.compile(gx)
    scene = int(p.search(fname.replace('incredibles-','')).group(1))
    pad_data.append(calculate_pad(mean_brightness(src),mean_saturation(src)) + [



df = pd.DataFrame(pad_data,columns=['pleasure','dominance','arousal', 'scene'] )
df = df.sort_values(by=['scene'],ascending=True)
df = df.groupby(['scene']).mean()
normalized_df=(df-df.min())/(df.max()-df.min())
normalized_df

ax2 = normalized_df.plot.line()
ax2.set_title('After interpolation')
ax2.set_xlabel("scene")
ax2.set_ylabel("valence")
normalized_df = normalized_df.reset_index()
```



In [5]:
```
normalized_df.rolling(15,  on='scene').mean().plot(x='scene',style='-')
```

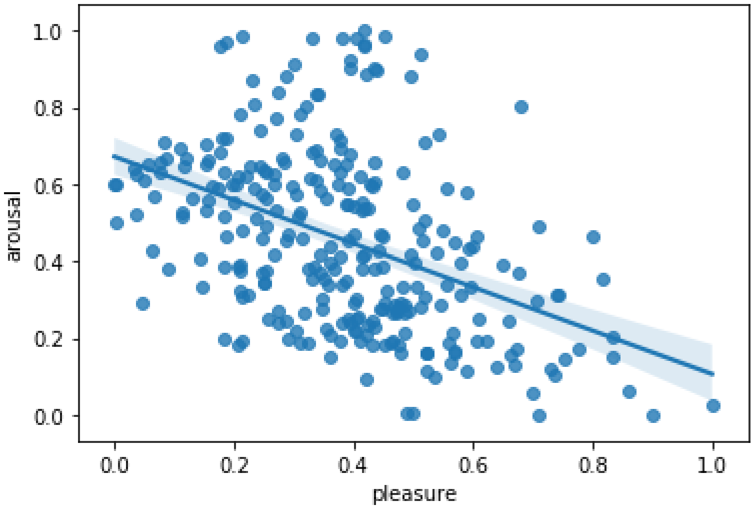
Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25770660470>
```



In [6]:
```
import seaborn as sns
sns.regplot(normalized_df['pleasure'],normalized_df['arousal'])
```

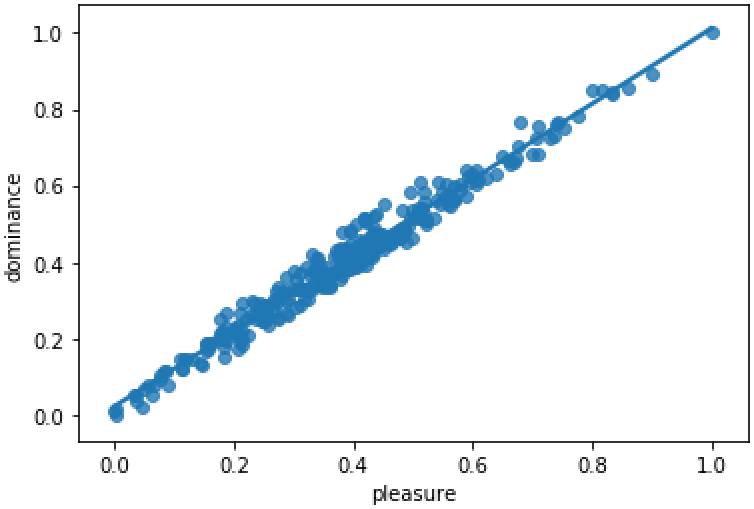
```
C:\Users\YihengYe\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the publ
ic API at pandas.testing instead.
  import pandas.util.testing as tm
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25774a3a240>
```



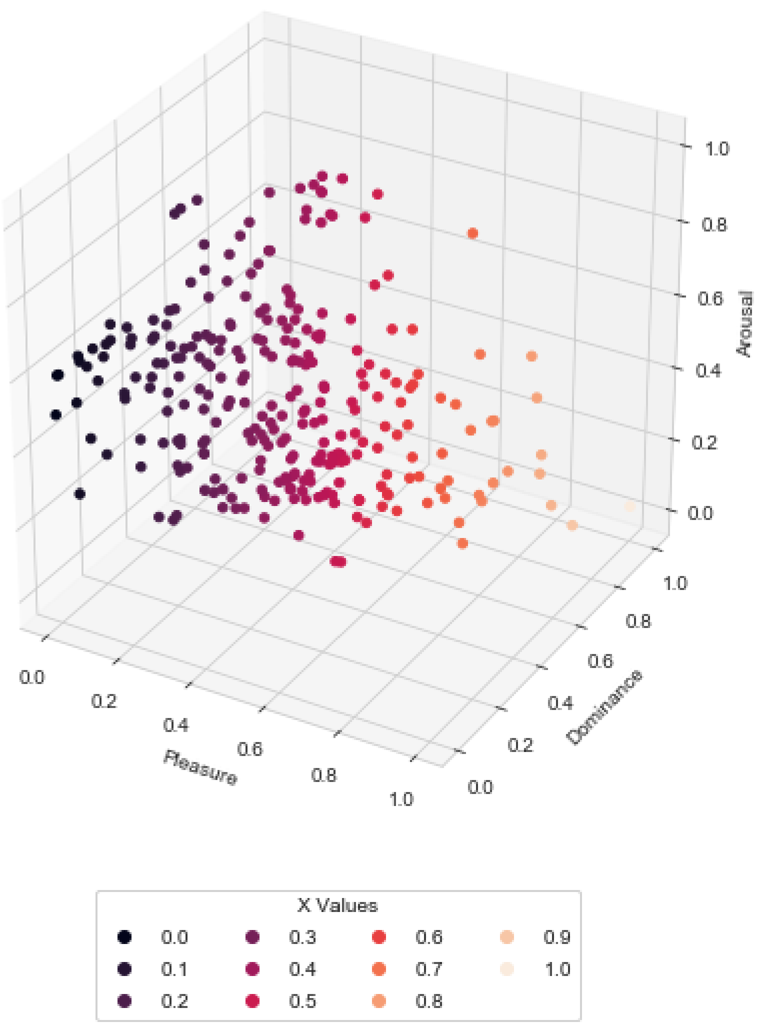
```
In [7]: sns.regplot(normalized_df['pleasure'],normalized_df['dominance'])
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x25774ac63c8>
```



In [8]:

```
import seaborn as sns, numpy as np, pandas as pd, random
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
sns.set_style("whitegrid", {'axes.grid' : False})

fig = plt.figure(figsize=(6,6))

ax = Axes3D(fig)

x = normalized_df['pleasure'].tolist()
y = normalized_df['dominance'].tolist()
z = normalized_df['arousal'].tolist()

cm = sns.palplot(sns.color_palette("BrBG", 7))
g = ax.scatter(x, y, z, c=x, marker='o', depthshade=False, cmap=cm)
ax.set_xlabel('Pleasure')
ax.set_ylabel('Dominance')
ax.set_zlabel('Arousal')

# produce a legend with the unique colors from the scatter
legend = ax.legend(*g.legend_elements(), loc="lower center", title="X Values", bo
ax.add_artist(legend)

plt.show()
```



In [9]:

```python
new_df = normalized_df.copy()
new_df['pleasure_diff'] = new_df['pleasure'].diff(-1)
new_df['pleasure_inflection'] = np.where(new_df['pleasure_diff'] > 0, 1, 0)
new_df[new_df['pleasure_inflection'].diff() != 0]['scene']
```

Out[9]:

```
0        1
2        3
5        6
6        7
10      11
      ...
264    265
266    267
269    270
273    274
276    277
Name: scene, Length: 141, dtype: int64
```

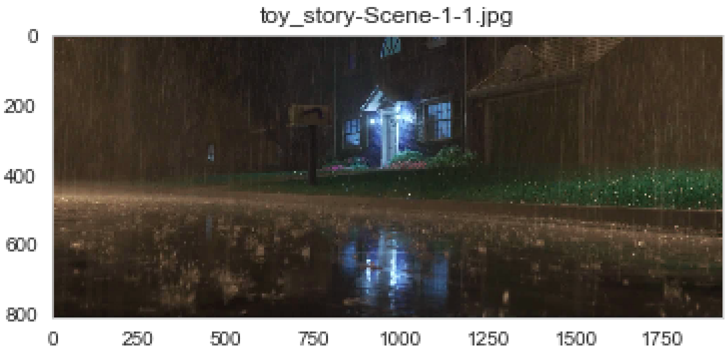
In [10]:
```
ANGER_POINT = [-0.43, 0.67, 0.34]
JOY_POINT = [0.76, 0.48, 0.35]
SURPRISE_POINT = [0.4, 0.67, -0.13]
DISGUST_POINT = [-0.6, 0.35, 0.11]
FEAR_POINT = [-0.64, 0.6, -0.43]
SADNESS_POINT = [-0.63, 0.27, -0.33]


from scipy import spatial
kdtree = spatial.cKDTree(np.array([ANGER_POINT,JOY_POINT,SURPRISE_POINT,DISGUST_
mu = normalized_df.sample(1)
print(mu)
print(kdtree.data)
dist, ix = kdtree.query([mu['pleasure'].iloc[0],mu['arousal'].iloc[0],mu['domina
print(dist)
kdtree.data[ix]
```

```
    scene  pleasure  dominance   arousal
38     39  0.289327   0.264381  0.196742
[[-0.43  0.67  0.34]
 [ 0.76  0.48  0.35]
 [ 0.4   0.67 -0.13]
 [-0.6   0.35  0.11]
 [-0.64  0.6  -0.43]
 [-0.63  0.27 -0.33]]
[0.55596595 0.62590529]
```

Out[10]:
```
array([[ 0.76,  0.48,  0.35],
       [ 0.4 ,  0.67, -0.13]])
```

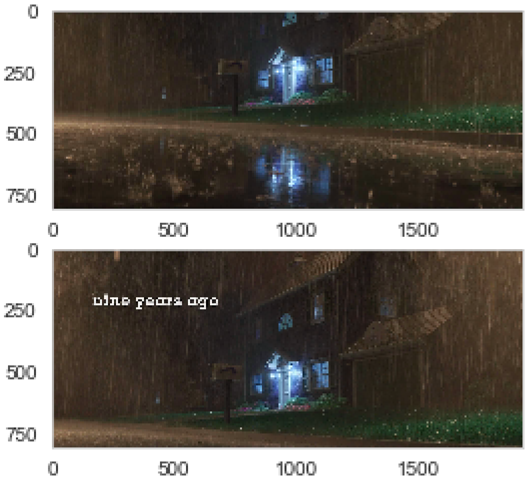
In [11]:

```
mypath = "data/scenes/toy_story_4"

onlyfiles = [f for f in listdir(mypath) if isjpg(join(mypath, f))]
i = 0
ans={}
testEld = []
index = []

if i < len(onlyfiles) - 1:
    test = cv2.imread(join(mypath, onlyfiles[i]))
    test_next = cv2.imread(join(mypath, onlyfiles[i+1]))

    testEld.append(test)
    testEld.append(test_next)
    plt.imshow(test)
    title = onlyfiles[i]
    plt.title(f"{title}")
    plt.show()
    i+=1
```



In [12]:

```
calculate_opticalFlow(test, test_next)
```



Out[12]:

```
array([1.2547378e-06, 2.0199174e-07, 1.5757147e-05, ..., 4.8620518e-07,
       2.3405102e-08, 8.8111851e-11], dtype=float32)
```

In [13]:
```
calculate_pad()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-e3db7ac68575> in <module>
----> 1 calculate_pad()

TypeError: calculate_pad() missing 2 required positional arguments: 'brightnes
s' and 'saturation'
```

In [4]:
```
# Model
```

In [26]:
```
# label your points by distance
def label(x):
    try:
        dist, ix = kdtree.query(x,k=1)
    except Exception as e:
        print(x)
        print(e)
    if ix==0:
        return 'Anger'
    elif ix==1:
        return "Joy"
    elif ix==2:
        return 'Surprise'
    elif ix==-3:
        return "Disgust"
    elif ix==4:
        return "Fear"
    else:
        return 'Sadness'
```

In [33]:
```
df_incredibles=normalized_df.copy().set_index('scene')
df_incredibles['emotion']=df_incredibles.apply(label, axis=1)
df_incredibles['emotion'].value_counts()
```

Out[33]:
```
Joy         250
Surprise     14
Anger        12
Sadness       3
Name: emotion, dtype: int64
```

The distribution of emotions for movie "incredibles" is highly concentrate on joy so we need more data

In [41]:

```
#generic_get_pad_method
def get_pad(path):
    mypath =path

    files = [f for f in listdir(mypath) if isjpg(join(mypath, f))]
    pad_data = []
    film=path.split('/')[2]
    rep=film+'-'
    for fname in files:
        src = cv2.imread(join(mypath, fname),1)
        if(src is None):
            continue
        gx = '^[^-]*-([^-]*).*'
        p = re.compile(gx)
        scene = int(p.search(fname.replace(rep,'')).group(1))
        pad_data.append(calculate_pad(mean_brightness(src),mean_saturation(src))



    df = pd.DataFrame(pad_data,columns=['pleasure','dominance','arousal', 'scene
    df = df.sort_values(by=['scene'],ascending=True)
    df = df.groupby(['scene']).mean()
    normalized_df=(df-df.min())/(df.max()-df.min())
    return normalized_df
```

In [42]:

```
paths=["data/scenes/big_hero_6","data/scenes/cars_3", "data/scenes/up", "data/sc
```

In [43]:

```
dfs={}
```

In [44]:

```
for path in paths:
    filmname=path.split('/')[2]
    zz=get_pad(path)
    dfs[filmname]=zz
```

In [47]:

```
#get a big df of all scenes' pad and emotion
model_df=df_incredibles.copy().reset_index().drop(columns=['scene'], axis=1)
for i in dfs.values():
```

```
In [48]: model_df
```

```
Out[48]:
         pleasure  dominance   arousal  emotion
    0    0.129684   0.146336  0.564982      Joy
    1    0.174702   0.197951  0.589309      Joy
    2    0.815460   0.849514  0.354440      Joy
    3    0.647714   0.675657  0.392973      Joy
    4    0.513828   0.541449  0.455581      Joy
  ...         ...        ...       ...      ...
  274    0.658579   0.666346  0.246615      Joy
  275    0.443335   0.440690  0.278085      Joy
  276    0.376907   0.357549  0.193397      Joy
  277    0.561941   0.546773  0.133049      Joy
  278    0.833372   0.839701  0.151880      Joy

279 rows × 4 columns
```

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [5]: # Evaluation
```