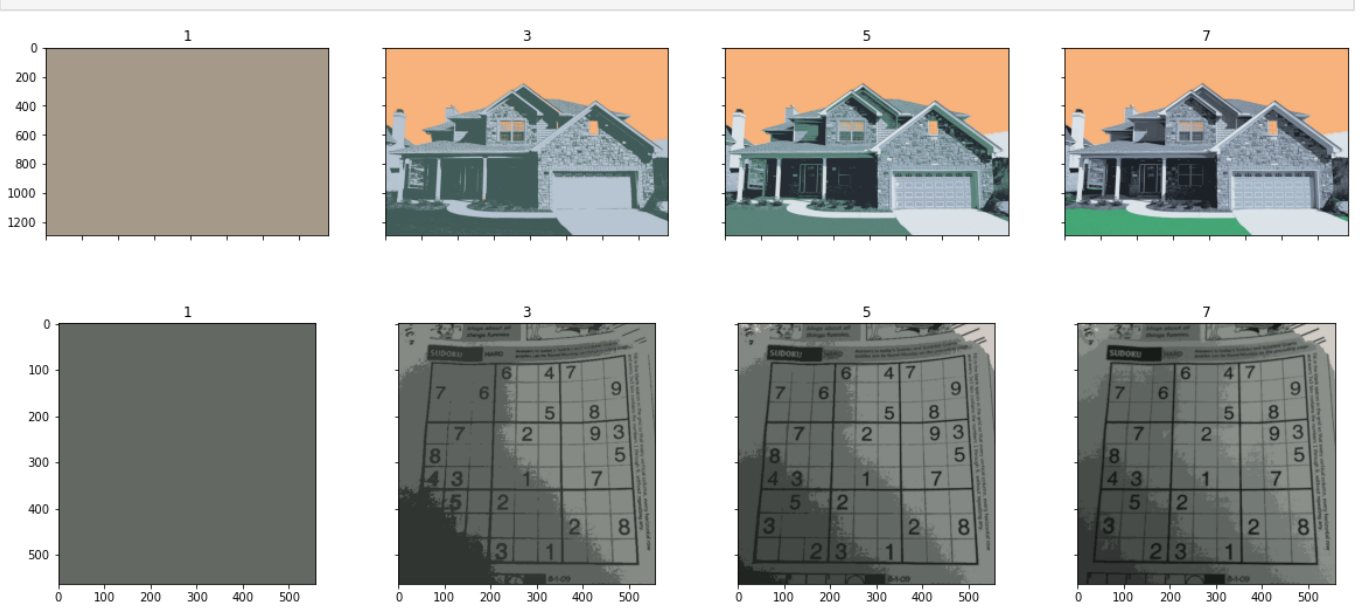
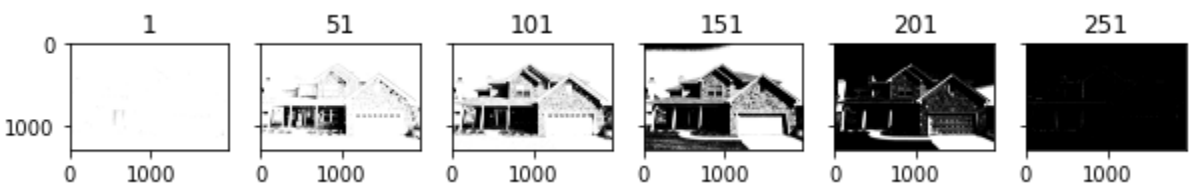
In [1]:

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img1 = cv.imread('home.jpg', 1)
img2 = cv.imread('sudoku.png', 1)

Q1home = img1.reshape((-1,3))
Q1home = np.float32(Q1home)
Q1sudoku = img2.reshape((-1,3))
Q1sudoku = np.float32(Q1sudoku)

# make display
fig, axs = plt.subplots(2, 4)
fig.set_size_inches(20, 9)

# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 10, 1.0)

position = 0

for x in range(1, 9, 2):
    ret,label,center=cv.kmeans(Q1home,x,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)
    # Now convert back into uint8, and make original image
    center = np.uint8(center)
    res = center[label.flatten()]
    res2 = res.reshape((img1.shape))
    axs[0,position].imshow(res2)
    axs[0,position].set_title(x)
    ret,label,center=cv.kmeans(Q1sudoku,x,None,criteria,10,cv.KMEANS_RANDOM_CENTERS)
    # Now convert back into uint8, and make original image
    center = np.uint8(center)
    res = center[label.flatten()]
    res2 = res.reshape((img2.shape))
    axs[1,position].imshow(res2)
    axs[1,position].set_title(x)
    position+=1

# Hide labels
for ax in axs.flat:
    ax.label_outer()

plt.show()
```



In [2]:

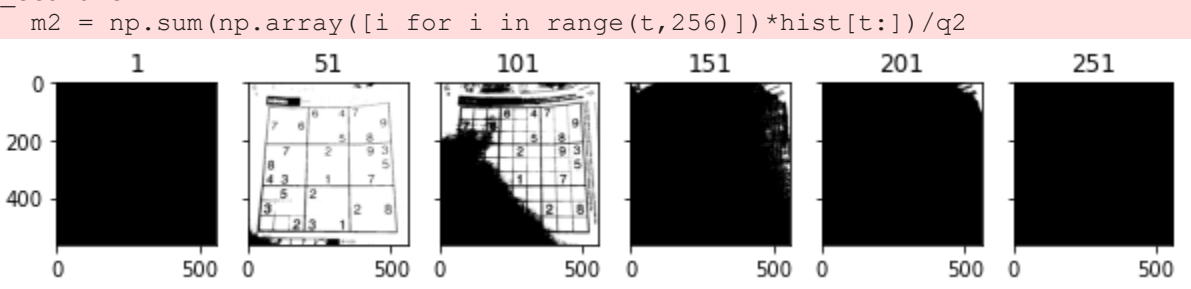
```
im = cv.imread('home.jpg', 0)

im_flat = np.reshape(im,(im.shape[0]*im.shape[1]))

[hist, _] = np.histogram(im, bins=256, range=(0, 255))

# Normalize
hist = 1.0*hist/np.sum(hist)

# make display
fig, axs = plt.subplots(1, 6)
fig.set_size_inches(10, 4)

count = 0

for thr in range(1, 255, 50):

    val_max = -999

    for t in range(1,255):
        q1 = np.sum(hist[:t])
        q2 = np.sum(hist[t:])
        m1 = np.sum(np.array([i for i in range(t)])*hist[:t])/q1
        m2 = np.sum(np.array([i for i in range(t,256)])*hist[t:])/q2
        val = q1*(1-q1)*np.power(m1-m2,2)
        if val_max < val:
            val_max = val


    axs[count].imshow(im > thr, cmap = 'gray')
    axs[count].set_title(thr)
    count+=1


# Hide labels
for ax in axs.flat:
    ax.label_outer()


plt.show()
```



In [3]:

```
im = cv.imread('sudoku.png', 0)

im_flat = np.reshape(im,(im.shape[0]*im.shape[1]))

[hist, _] = np.histogram(im, bins=256, range=(0, 255))

# Normalize
hist = 1.0*hist/np.sum(hist)

# make display
fig, axs = plt.subplots(1, 6)
fig.set_size_inches(10, 4)

count = 0

for thr in range(1, 255, 50):

    val_max = -999

    for t in range(1,255):
        q1 = np.sum(hist[:t])
        q2 = np.sum(hist[t:])
        m1 = np.sum(np.array([i for i in range(t)])*hist[:t])/q1
        m2 = np.sum(np.array([i for i in range(t,256)])*hist[t:])/q2
        val = q1*(1-q1)*np.power(m1-m2,2)
        if val_max < val:
            val_max = val


    axs[count].imshow(im > thr, cmap = 'gray')
    axs[count].set_title(thr)
    count+=1


# Hide labels
for ax in axs.flat:
    ax.label_outer()


plt.show()
```

```
<ipython-input-3-9dd2c5b7bad0>:23: RuntimeWarning: invalid value encountered in double
_scalars
  m1 = np.sum(np.array([i for i in range(t)])*hist[:t])/q1
<ipython-input-3-9dd2c5b7bad0>:24: RuntimeWarning: invalid value encountered in double
_scalars
  m2 = np.sum(np.array([i for i in range(t,256)])*hist[t:])/q2
```



In [ ]:

```
#2) a,b,c
# as can be seen above the effectiviness of the segmentation depended heavily on the
# along with the threshold applied to it with the house it was probably peak around 1
# sudoku game was around 51
```

In [ ]: