Image Processing Algorithms from scratch

In [1]:

```python
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
import pandas as pd
import os
```

In [35]:

```python
images_name = os.listdir("images")
plt.figure(figsize=(12,10))
plt.imshow(cv.imread("images/"+images_name[3]))
plt.show()
```

Getting images in list

In [36]:

```python
full_sized = []
for name in images_name:
    img = cv.imread("images/"+name)
    full_sized.append(img)
len(full_sized)
```

Out[36]:

```
6
```

1) Resizing images

In [45]:

```python
resized = []
for name in images_name:
    img = cv.imread("images/"+name)
    resized.append(cv.resize(img, (img.shape[1]//5,img.shape[0]//5)))
len(resized)
```

Out[45]:

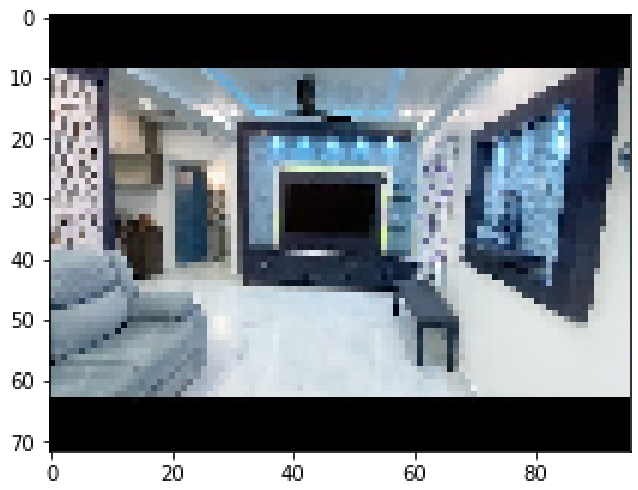
```
6
```

In [46]:

```python
plt.imshow(resized[3]),resized[3].shape
```

Out[46]:

```
(<matplotlib.image.AxesImage at 0x1a7a9b2abb0>, (72, 96, 3))
```

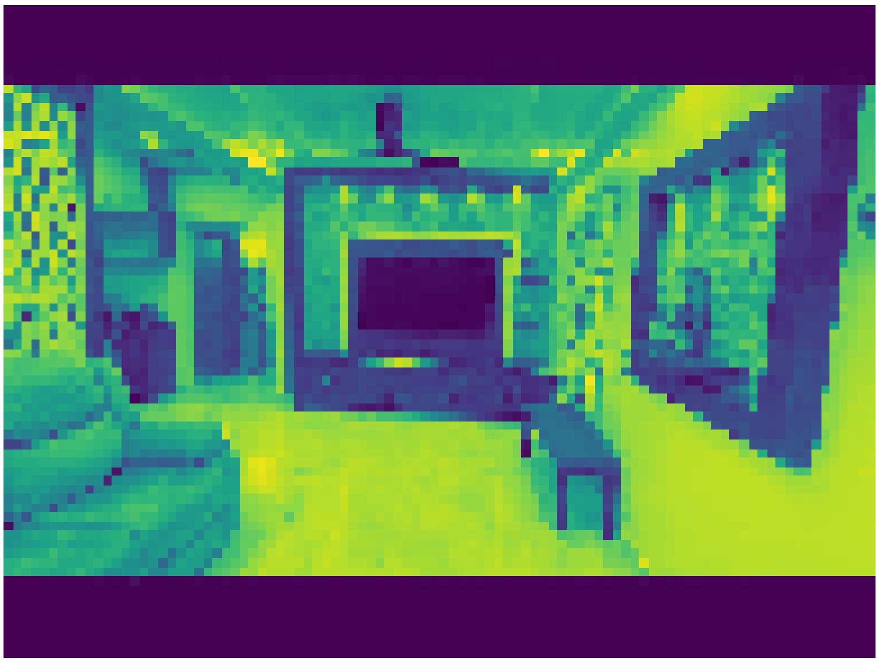


2) 3d to 2d conversion (Gray image)

In [47]:

```python
def d2(img):
    """Converts 3d image into 2d by average method"""
    copy = np.zeros((img.shape[0],img.shape[1]))
    print("img.shape,np.array(copy).shape",img.shape,np.array(copy).shape)
    for i,row in enumerate(img):
        for j,pix in enumerate(row):
            copy[i][j] = sum(pix)//3
    return np.array(copy)

res = d2(resized[3])
plt.figure(figsize=(8,6))
plt.imshow(res)
plt.axis("off")
plt.show()
```

```
img.shape,np.array(copy).shape (72, 96, 3) (72, 96)
```



3) Edge Detector (Horizontal Scanning)

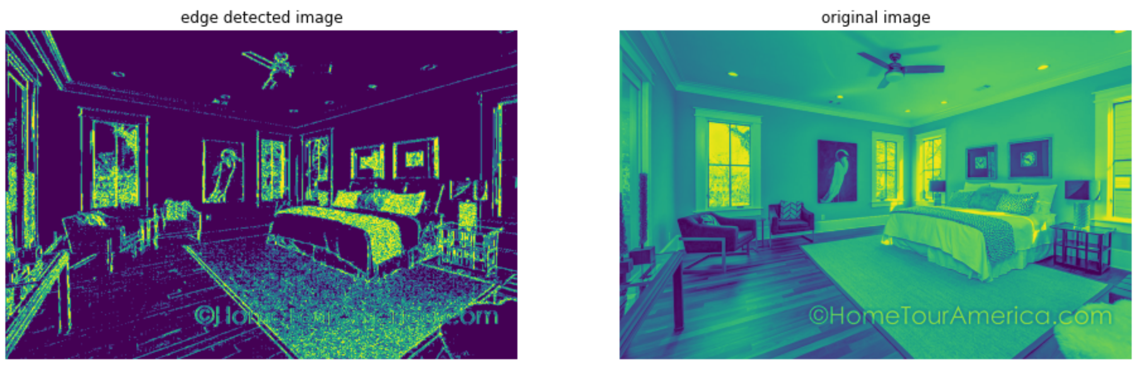
In [48]:

```python
def edge(img,threshold):
    copy = np.zeros((img.shape[0],img.shape[1]))
    for i,row in enumerate(img):
        flag = False
        for j,pix in enumerate(row):
            if flag:
                if abs(img[i][j-1] - img[i][j]) > threshold:
                    copy[i][j] = 255
                else:
                    copy[i][j] = 0
            flag = True
    return np.array(copy)
```

In [52]:

```python
img0 = d2(full_sized[0])
edge_res = edge(img0,12)
plt.figure(figsize=(16,14))
plt.subplot(1,2,1)
plt.axis("off")
plt.title("edge detected image")
plt.imshow(edge_res)
plt.subplot(1,2,2)
plt.axis("off")
plt.title("original image")
plt.imshow(img0)
plt.show()
```

```
img.shape,np.array(copy).shape (661, 1024, 3) (661, 1024)
```



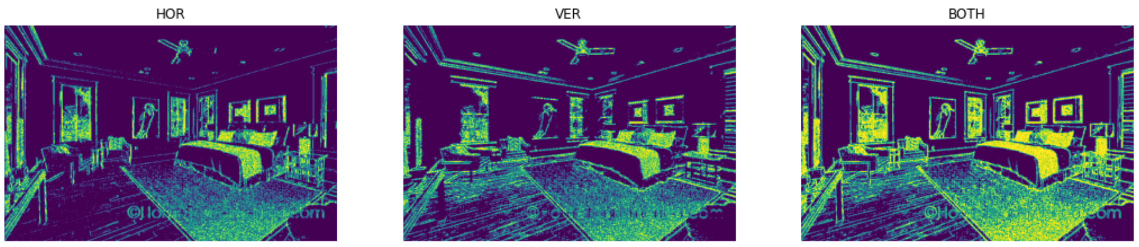
4) Edge Detection (Horizontal,Verticle,Both scanning)

In [53]:

```python
def d2(img):
    """3d to 2d conversion"""
    copy = np.zeros((img.shape[0],img.shape[1]))
    for i,row in enumerate(img):
        for j,pix in enumerate(row):
            copy[i][j] = sum(pix)//3
    return np.array(copy)

def edge(img,threshold,scan='hor'):
    """Deteccting edges in image"""
    if len(img.shape) > 2:
        img = d2(img)
    if scan == "hor":
        pass
    elif scan == "ver":
        img = np.rot90(img,k=1)
    elif scan == "both":
        img1 = edge(img,threshold,scan = "ver")
        img2 = edge(img,threshold,scan = "hor")
        merged = img1+img2
        for i,row in enumerate(merged):
            for j,pix in enumerate(row):
                if pix > 255:
                    merged[i][j] = 255
        return merged

    copy = np.zeros((img.shape[0],img.shape[1]))
    for i,row in enumerate(img):
        flag = False
        for j,pix in enumerate(row):
            if flag:
                if abs(img[i][j-1] - img[i][j]) > threshold:
                    copy[i][j] = 255
                else:
                    copy[i][j] = 0
            flag = True
    if scan != 'hor':
        return np.rot90(copy, k=3)
    else:
        return copy
```

In [23]:

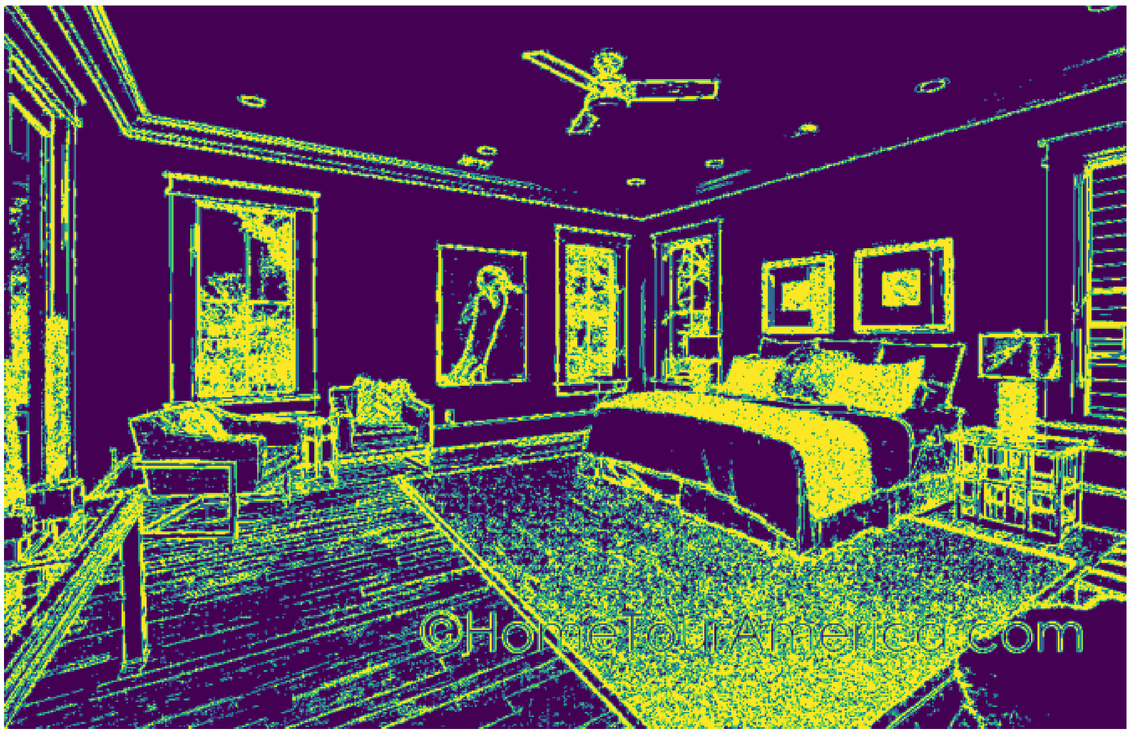
```python
hor = edge(img0,12,"hor")
ver = edge(img0,12,"ver")
both = edge(img0,12,"both")

plt.figure(figsize=(18,15))

plt.subplot(1,3,1)
plt.title("HOR")
plt.imshow(hor)
plt.axis("off")

plt.subplot(1,3,2)
plt.title("VER")
plt.imshow(ver)
plt.axis("off")

plt.subplot(1,3,3)
plt.title("BOTH")
plt.imshow(both)
plt.axis("off")
plt.show()
```



In [26]:

```python
plt.figure(figsize=(12,8))
plt.imshow(both)
plt.title("BOTH")
plt.axis("off")
plt.show()
```

BOTH

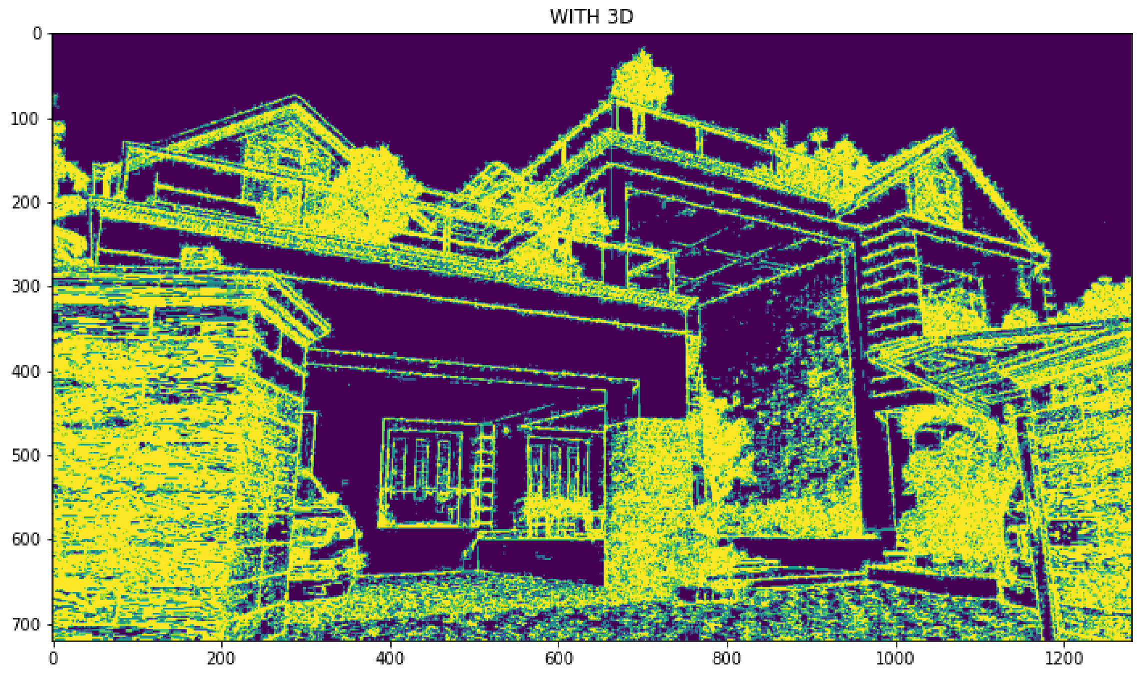


with 3D image

In [31]:

```python
d3_img = edge(full_sized[5],5,"both")
print("shape",full_sized[5].shape)
plt.figure(figsize=(12,8))
plt.imshow(d3_img)
plt.title("WITH 3D")
plt.show()
```

```
shape (720, 1280, 3)
```



Conclusion from above results

Here we can clearly undrestand horizontal scanning missing some edges, same thing happened with verticle scanning,
But in both scanning it gives maximum information from image.
Applications:
-Getting shape
-Getting insights from image
-Neglecting plane surfaces from images
-Usefull for image classification or Detection problems in CNN

In [ ]:

```python

```