In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries
 installed
# It is defined by the kaggle/python docker image: https://github.com/ka
ggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import json
from matplotlib import pyplot as plt
from skimage import color
from skimage.feature import hog
from sklearn import svm
from sklearn.metrics import classification_report,accuracy_score
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
import os.path
import random
from random import choice, sample
import PIL
```

In [2]:
```
import itertools
import numpy as np
import matplotlib.pyplot as plt

from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

In [3]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
```

In [4]:
```
# Install a conda package in the current Jupyter kernel
import sys
!conda install --yes --prefix {sys.prefix} opencv
```

```
Solving environment: done

## Package Plan ##

  environment location: /home/ec2-user/anaconda3/envs/python2

  added / updated specs:
    - opencv


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    libvpx-1.7.0               |       h439df22_0         2.4 MB
    libopus-1.3                |       h7b6447c_0         631 KB
    opencv-3.4.2               |   py27h6fd60c2_1          11 KB
    jasper-2.0.14              |       h07fcdf6_1         1.1 MB
    libglu-9.0.0               |       hf484d3e_1         377 KB
    certifi-2018.11.29         |           py27_0         146 KB
    ffmpeg-4.0                 |       hcdf2ecd_0        73.7 MB
    py-opencv-3.4.2            |   py27hb342d67_1         1.2 MB
    freeglut-3.0.0             |       hf484d3e_5         251 KB
    libopencv-3.4.2            |       hb342d67_1        40.4 MB
    ------------------------------------------------------------
                                           Total:       120.2 MB

The following NEW packages will be INSTALLED:

    ffmpeg:          4.0-hcdf2ecd_0
    freeglut:        3.0.0-hf484d3e_5
    jasper:          2.0.14-h07fcdf6_1
    libglu:          9.0.0-hf484d3e_1
    libopencv:       3.4.2-hb342d67_1
    libopus:         1.3-h7b6447c_0
    libvpx:          1.7.0-h439df22_0
    py-opencv:       3.4.2-py27hb342d67_1

The following packages will be UPDATED:

    certifi:         2018.8.24-py27_1     conda-forge --> 2018.11.29-py
27_0
    libstdcxx-ng:    7.2.0-hdf63c60_3                 --> 8.2.0-hdf63c6
0_1
    opencv:          3.4.1-py27h6fd60c2_1             --> 3.4.2-py27h6f
d60c2_1
    openssl:         1.0.2p-h470a237_0    conda-forge --> 1.0.2p-h14c39
75_0

The following packages will be DOWNGRADED:

    ca-certificates: 2018.8.24-ha4d7672_0 conda-forge --> 2018.03.07-0



Downloading and Extracting Packages
libvpx-1.7.0         | 2.4 MB    | ####################################
```

```
# | 100%
libopus-1.3           | 631 KB    | ####################################
# | 100%
opencv-3.4.2          | 11 KB     | ####################################
# | 100%
jasper-2.0.14         | 1.1 MB    | ####################################
# | 100%
libglu-9.0.0          | 377 KB    | ####################################
# | 100%
certifi-2018.11.29    | 146 KB    | ####################################
# | 100%
ffmpeg-4.0            | 73.7 MB   | ####################################
# | 100%
py-opencv-3.4.2       | 1.2 MB    | ####################################
# | 100%
freeglut-3.0.0        | 251 KB    | ####################################
# | 100%
libopencv-3.4.2       | 40.4 MB   | ####################################
# | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

In [5]:
```
import cv2
```

In [6]:
```
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) wi
ll list the files in the input directory
from subprocess import check_output
# Any results you write to the current directory are saved as output.
```

In [7]:
```
'''
f = open('counting_train.json')
dataset = json.load(f)
np.shape(dataset)
#dataset
'''
```

Out[7]:
```
"\nf = open('counting_train.json')\ndataset = json.load(f)\nnp.shape(da
taset)\n#dataset\n"
```

In [8]:
```
'''
import random
from random import choice, sample
num_samples=50
dataset_index=random.sample(dataset,num_samples)
np.shape(dataset_index)
#dataset_index
'''
```

Out[8]:
```
'\nimport random\nfrom random import choice, sample\nnum_samples=50\nda
taset_index=random.sample(dataset,num_samples)\nnp.shape(dataset_index)
\n#dataset_index\n'
```

In [9]:
```
N=range(500000)
num_samples=5000
dataset_index=random.sample(N,num_samples)
#dataset_index
```

In [10]:
```
meta_dir = "amazon-bin/metadata/"
img_dir = "amazon-bin/bin-images/"
```

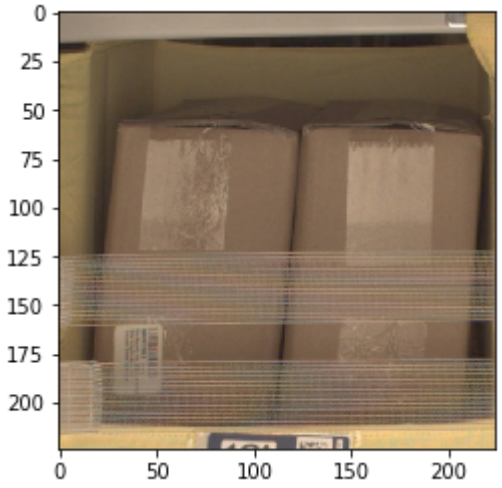
In [11]:
```
fname_img_vec=[]
fname_meta_vec=[]
for i in range(num_samples):
    fname_img=str(dataset_index[i])+'.jpg'
    fname_meta=str(dataset_index[i])+'.json'
    jpg_path = os.path.join(img_dir,fname_img)
    json_path = os.path.join(meta_dir,fname_meta)
    if os.path.isfile(jpg_path) & os.path.isfile(json_path):
        fname_img_vec.append(fname_img)
        fname_meta_vec.append(fname_meta)
print(np.shape(fname_img_vec))
print(np.shape(fname_meta_vec))
#fname_meta_vec
```

```
(4911,)
(4911,)
```

In [12]:
```
# get label from
num_samples=len(fname_img_vec)
#labels=np.zeros((num_samples,1))
labels=[]
fname_img_vec_less_5=[]
fname_meta_vec_less_5=[]
for i in range(num_samples):
    fname_meta=fname_meta_vec[i]
    fname_img=fname_img_vec[i]
    json_path = os.path.join(meta_dir,fname_meta)
    f = open(json_path)
    dataset = json.load(f)
    expected_quantity=dataset['EXPECTED_QUANTITY']
    if expected_quantity<=5:
        #print(fname)
        #print(expected_quantity)
        f.close()
        #labels[i,0]=expected_quantity
        labels.append(expected_quantity)
        fname_img_vec_less_5.append(fname_img)
        fname_meta_vec_less_5.append(fname_meta)
labels=np.array(labels,ndmin=2)
labels=labels.T
print(np.shape(labels))
print(np.shape(fname_img_vec_less_5))
print(np.shape(fname_meta_vec_less_5))
```

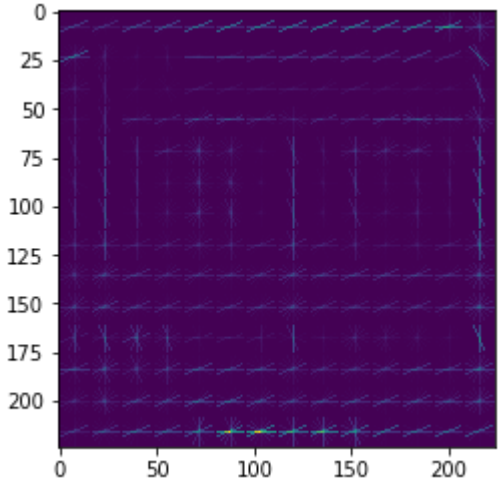
```
(3296, 1)
(3296,)
(3296,)
```

In [13]:
```
i=1
fname=fname_img_vec_less_5[i]
fname='542.jpg'
fname
jpg_path = os.path.join(img_dir,fname)
# Convert Image to array
img = PIL.Image.open(jpg_path)
resized_img = img.resize((224,224))
plt.imshow(resized_img)
arr = np.array(resized_img)
plt.imshow(arr)
data_gray = color.rgb2gray(arr)
#plt.imshow(data_gray)
ppc = 16
hog_images = []
hog_features = []
fd,hog_image = hog(data_gray, orientations=8, pixels_per_cell=(ppc,ppc),
cells_per_block=(4, 4),block_norm= 'L2',visualise=True)
```



In [14]:
```
plt.imshow(hog_image)
```

Out[14]: <matplotlib.image.AxesImage at 0x7fd90a88d510>



In [15]:
```
fname_img_vec_less_5[i]
```

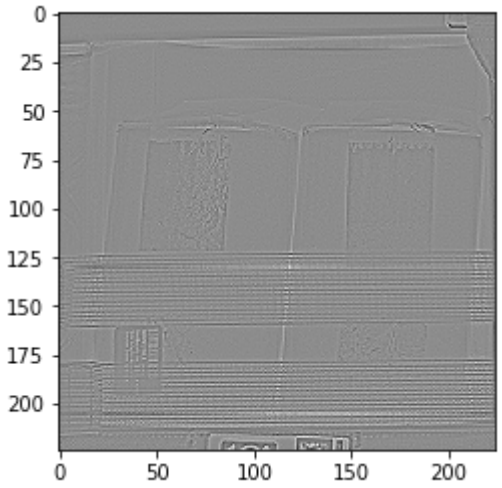
Out[15]:
```
'453433.jpg'
```

In [16]:
```
#img = cv2.imread(jpg_path,0)
#resized_img = img.resize((224,224))
laplacian = cv2.Laplacian(data_gray,cv2.CV_64F)
sobelx = cv2.Sobel(data_gray,cv2.CV_64F,1,0,ksize=5)
sobely = cv2.Sobel(data_gray,cv2.CV_64F,0,1,ksize=5)
np.shape(img)
```

Out[16]:
```
(591, 577, 3)
```

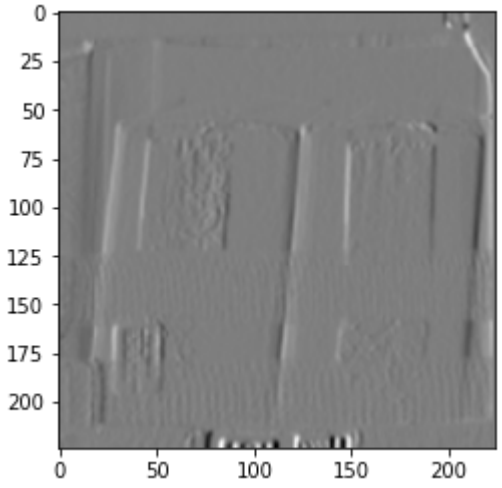
In [17]:
```
plt.imshow(laplacian,cmap = 'gray')
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x7fd90a816350>
```



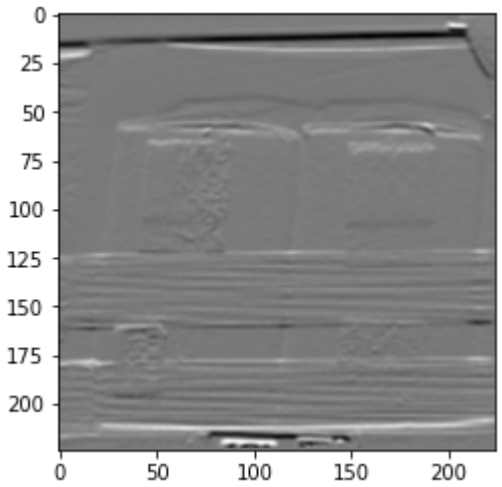
In [18]:
```
plt.imshow(sobelx,cmap = 'gray')
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x7fd908128a10>
```



In [19]:
```
plt.imshow(sobely,cmap = 'gray')
```

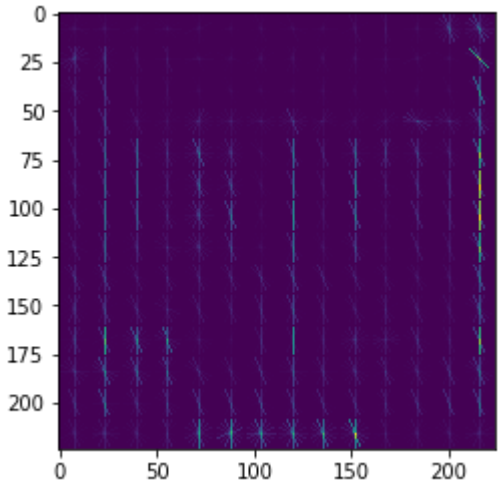
Out[19]:
```
<matplotlib.image.AxesImage at 0x7fd9080a5110>
```



In [20]:
```
ppc = 16
hog_images = []
hog_features = []
fd,hog_image = hog(sobelx, orientations=8, pixels_per_cell=(ppc,ppc),cel
ls_per_block=(4, 4),block_norm= 'L2',visualise=True)
```

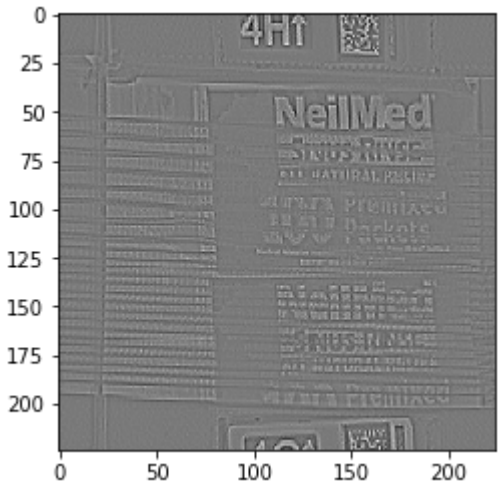
In [21]:
```
plt.imshow(hog_image)
```

Out[21]:
```
<matplotlib.image.AxesImage at 0x7fd90a90a1d0>
```



In [22]:
```
# do HOG on edges images, resize images on the fly
hog_features_size=15488
num_samples=len(fname_meta_vec_less_5)
hog_features_matrix=np.zeros((num_samples,hog_features_size))
for i in range(num_samples):
    fname=fname_img_vec_less_5[i]
    fname
    jpg_path = os.path.join(img_dir,fname)
    # Convert Image to array
    img = PIL.Image.open(jpg_path)
    resized_img = img.resize((224,224))
    # img = Image.open(jpg_path).convert('RGB')
    # resized_img = img.resize((224,224),img.BILINEAR)
    arr = np.array(resized_img)
    #plt.imshow(arr)
    data_gray = color.rgb2gray(arr)
    #plt.imshow(data_gray)
    ppc = 16
    hog_images = []
    hog_features = []
    laplacian = cv2.Laplacian(data_gray,cv2.CV_64F)
    fd,hog_image = hog(laplacian, orientations=8, pixels_per_cell=(ppc,p
pc),cells_per_block=(4, 4),block_norm= 'L2',visualise=True)
    hog_images.append(hog_image)
    hog_features.append(fd)
    hog_features = np.array(hog_features)
    hog_features_matrix[i,:]=hog_features
    plt.imshow(laplacian,cmap = 'gray')
np.shape(hog_features_matrix)
```

Out[22]:
```
(3296, 15488)
```



In [23]:
```
'''
# do HOG on orginal images, resize images on the fly
hog_features_size=15488
num_samples=len(fname_meta_vec_less_5)
hog_features_matrix=np.zeros((num_samples,hog_features_size))
for i in range(num_samples):
    fname=fname_img_vec_less_5[i]
    fname
    jpg_path = os.path.join(img_dir,fname)
    # Convert Image to array
    img = PIL.Image.open(jpg_path)
    resized_img = img.resize((224,224))
    # img = Image.open(jpg_path).convert('RGB')
    # resized_img = img.resize((224,224),img.BILINEAR)
    arr = np.array(resized_img)
    #plt.imshow(arr)
    data_gray = color.rgb2gray(arr)
    #plt.imshow(data_gray)
    ppc = 16
    hog_images = []
    hog_features = []
    fd,hog_image = hog(data_gray, orientations=8, pixels_per_cell=(ppc,p
pc),cells_per_block=(4, 4),block_norm= 'L2',visualise=True)
    hog_images.append(hog_image)
    hog_features.append(fd)
    hog_features = np.array(hog_features)
    hog_features_matrix[i,:]=hog_features
np.shape(hog_features_matrix)
'''
```

Out[23]:
```
"\n# do HOG on orginal images, resize images on the fly\nhog_features_s
ize=15488\nnum_samples=len(fname_meta_vec_less_5)\nhog_features_matrix=
np.zeros((num_samples,hog_features_size))\nfor i in range(num_sample
s):\n    fname=fname_img_vec_less_5[i]\n    fname\n    jpg_path = os.pa
th.join(img_dir,fname)\n    # Convert Image to array\n    img = PIL.Ima
ge.open(jpg_path)\n    resized_img = img.resize((224,224))\n    # img =
Image.open(jpg_path).convert('RGB')\n    # resized_img = img.resize((22
4,224),img.BILINEAR)\n    arr = np.array(resized_img)\n    #plt.imshow
(arr)\n    data_gray = color.rgb2gray(arr)\n    #plt.imshow(data_gray)
\n    ppc = 16\n    hog_images = []\n    hog_features = []\n    fd,hog_
image = hog(data_gray, orientations=8, pixels_per_cell=(ppc,ppc),cells_
per_block=(4, 4),block_norm= 'L2',visualise=True)\n    hog_images.appen
d(hog_image)\n    hog_features.append(fd)\n    hog_features = np.array
(hog_features)\n    hog_features_matrix[i,:]=hog_features\nnp.shape(hog
_features_matrix)\n"
```

In [24]:
```
data_frame = np.hstack((hog_features_matrix,labels))
np.shape(data_frame)
np.random.shuffle(data_frame)
```

In [25]:
```
percentage = 80
partition = int(num_samples*percentage/100)
```

In [26]:
```
x_train, x_test = data_frame[:partition,:-1],  data_frame[partition:,:-1
]
y_train, y_test = data_frame[:partition,-1:].ravel() , data_frame[partit
ion:,-1:].ravel()

#clf = svm.SVC(kernel='poly',C=0.1)
#clf.fit(x_train,y_train)
```

In [27]:
```
# Logistic Regression
#clf = LogisticRegression(random_state=0, solver='lbfgs',multi_class='mu
ltinomial')
#clf.fit(x_train,y_train)
```

In [28]:
```
# Decision Tree
#clf = DecisionTreeClassifier(random_state=0)
#clf.fit(x_train,y_train)
```

In [29]:
```
# svm tuning
from sklearn import svm, grid_search
from sklearn.model_selection import GridSearchCV
def svc_param_selection(X, y, nfolds):
    Cs = [0.001, 0.01, 0.1, 1, 10]
    gammas = [0.001, 0.01, 0.1, 1]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(svm.SVC(kernel='poly'), param_grid, cv=nf
olds)
    grid_search.fit(X, y)
    grid_search.best_params_
    print('checkpoint')
    return grid_search.best_params_
```

```
/home/ec2-user/anaconda3/envs/python2/lib/python2.7/site-packages/sklea
rn/cross_validation.py:41: DeprecationWarning: This module was deprecat
ed in version 0.18 in favor of the model_selection module into which al
l the refactored classes and functions are moved. Also note that the in
terface of the new CV iterators are different from that of this module.
This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
/home/ec2-user/anaconda3/envs/python2/lib/python2.7/site-packages/sklea
rn/grid_search.py:42: DeprecationWarning: This module was deprecated in
version 0.18 in favor of the model_selection module into which all the
refactored classes and functions are moved. This module will be removed
in 0.20.
  DeprecationWarning)
```

In [ ]:
```
nfolds=3
best_params=svc_param_selection(x_train, y_train, nfolds)
```

In [ ]:
```
best_params
C=best_params['C']
gamma=best_params['gamma']
```

In [ ]:
```
# SVM
clf = svm.SVC(kernel='poly',C=C)
clf.fit(x_train,y_train)
```

In [ ]:
```
y_pred = clf.predict(x_test)
y_pred
```

In [ ]:
```
np.histogram(y_pred)
```

In [ ]:
```
print("Accuracy: "+str(accuracy_score(y_test, y_pred)))
print('\n')
print(classification_report(y_test, y_pred))
```

In [ ]:
```
C
```

In [ ]:
```
gamma
```

In [ ]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1
])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [ ]:
```
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)
```

In [ ]:
```
cnf_matrix
```

In [ ]:
```
class_names=[0,1,2,3,4,5]
```

In [ ]:
```
# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')
```

In [ ]:
```
# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')
```

In [ ]:
```
1+1
```

In [ ]:
```
print('checkpoint')
```

In [ ]:
```
# Tape removal algorithm
```

In [14]:
```
import cv2 as cv
```

In [22]:
```
def show_wait_destroy(winname, img):
    cv.imshow(winname, img)
    cv.moveWindow(winname, 500, 0)
    cv.waitKey(0)
    cv.destroyWindow(winname)
```

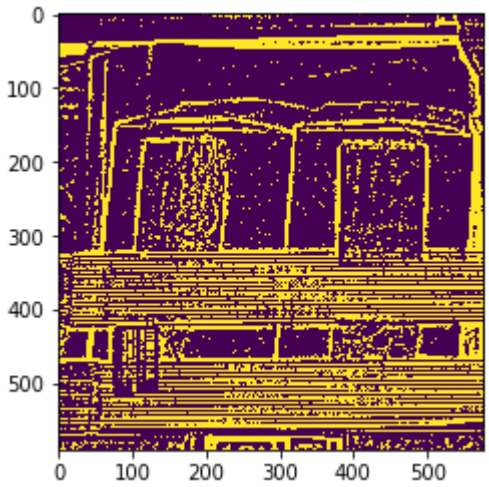
In [15]:
```
fname='542.jpg'
fname
jpg_path = os.path.join(img_dir,fname)
```

In [17]:
```
src = cv.imread(jpg_path, cv.IMREAD_COLOR)
gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
```

In [20]:
```
# Apply adaptiveThreshold at the bitwise_not of gray, notice the ~ symbo
l
gray = cv.bitwise_not(gray)
bw = cv.adaptiveThreshold(gray, 255, cv.ADAPTIVE_THRESH_MEAN_C, cv.THRES
H_BINARY, 15, -2)
```

In [27]:
```
plt.imshow(bw)
```

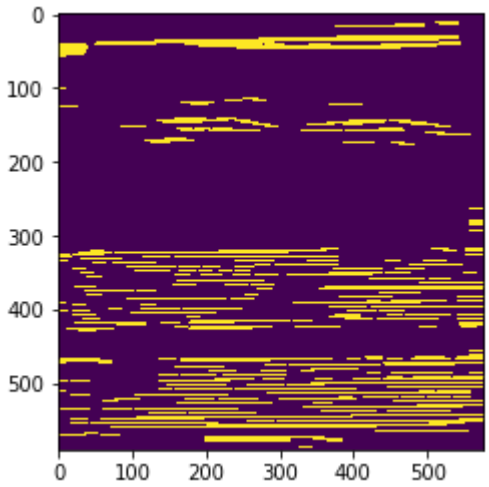
Out[27]:
```
<matplotlib.image.AxesImage at 0x7fb5baf62710>
```



In [47]:
```
# Create the images that will use to extract the horizontal and vertical
 lines
horizontal = np.copy(bw)
vertical = np.copy(bw)

# Specify size on horizontal axis
cols = horizontal.shape[1]
horizontal_size = cols / 30

# Create structure element for extracting horizontal lines through morph
ology operations
horizontalStructure = cv.getStructuringElement(cv.MORPH_RECT, (horizonta
l_size, 1))

# Apply morphology operations
horizontal = cv.erode(horizontal, horizontalStructure)
horizontal = cv.dilate(horizontal, horizontalStructure)
```

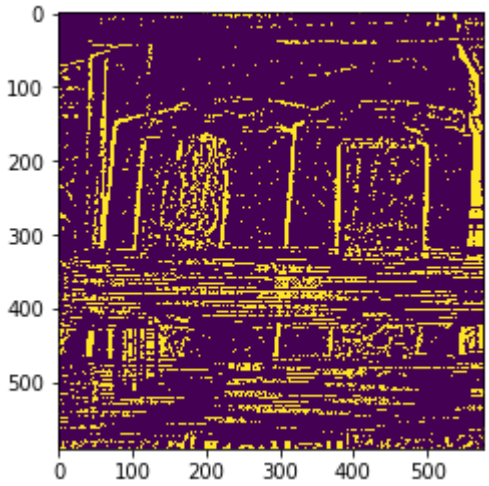
In [48]:
```
plt.imshow(horizontal)
```

Out[48]:
```
<matplotlib.image.AxesImage at 0x7fb5b8c12250>
```



In [57]:
```
diff=bw-horizontal
plt.imshow(diff)
```

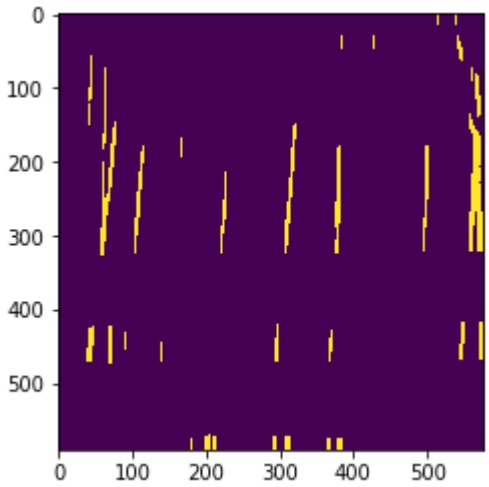
Out[57]:
```
<matplotlib.image.AxesImage at 0x7fb5b8569e10>
```



In [49]:
```
# Specify size on vertical axis
rows = vertical.shape[0]
verticalsize = rows / 30

# Create structure element for extracting vertical lines through morphol
ogy operations
verticalStructure = cv.getStructuringElement(cv.MORPH_RECT, (1, vertical
size))

# Apply morphology operations
vertical = cv.erode(vertical, verticalStructure)
vertical = cv.dilate(vertical, verticalStructure)
```

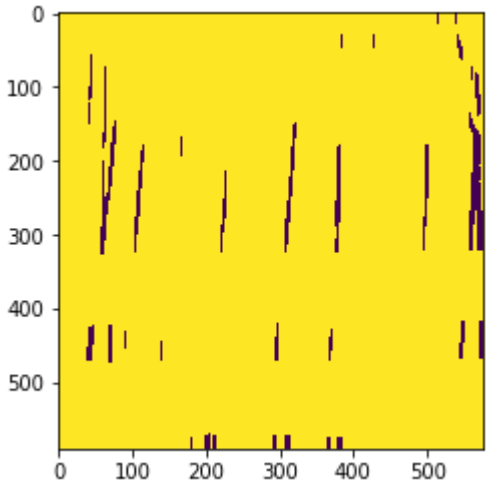
In [50]:
```
plt.imshow(vertical)
```

Out[50]: <matplotlib.image.AxesImage at 0x7fb5b8b73e90>



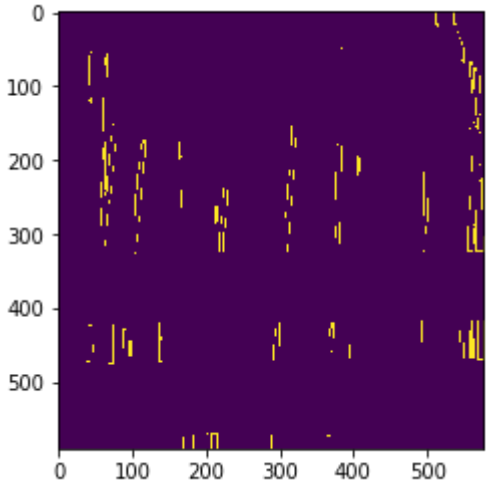
In [51]:
```
vertical = cv.bitwise_not(vertical)
plt.imshow(vertical)
```

Out[51]: <matplotlib.image.AxesImage at 0x7fb5b872dc90>



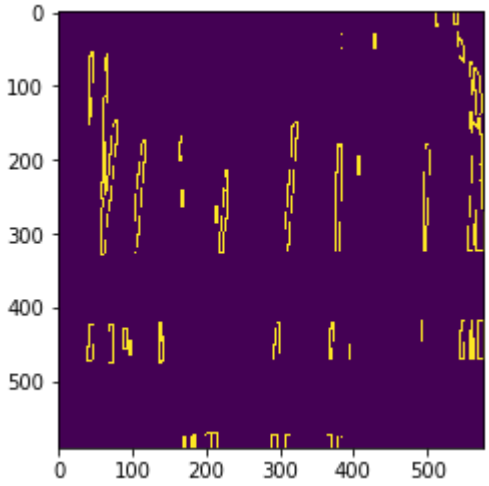
In [52]:
```
# Step 1
edges = cv.adaptiveThreshold(vertical, 255, cv.ADAPTIVE_THRESH_MEAN_C,cv
.THRESH_BINARY, 3, -2)
plt.imshow(edges)
```

Out[52]: <matplotlib.image.AxesImage at 0x7fb5b871db10>



In [53]:
```
# Step 2
kernel = np.ones((2, 2), np.uint8)
edges = cv.dilate(edges, kernel)
plt.imshow(edges)
```

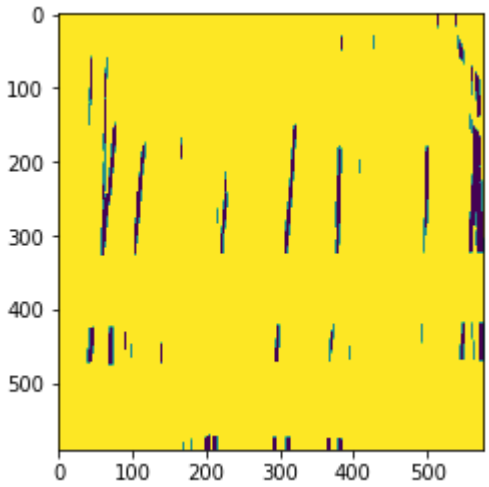
Out[53]: <matplotlib.image.AxesImage at 0x7fb5b868c850>



In [54]:
```
# Step 3
smooth = np.copy(vertical)

# Step 4
smooth = cv.blur(smooth, (2, 2))

# Step 5
(rows, cols) = np.where(edges != 0)
vertical[rows, cols] = smooth[rows, cols]
```

In [55]:
```
plt.imshow(vertical)
```

Out[55]:
```
<matplotlib.image.AxesImage at 0x7fb5b86030d0>
```



In [ ]: