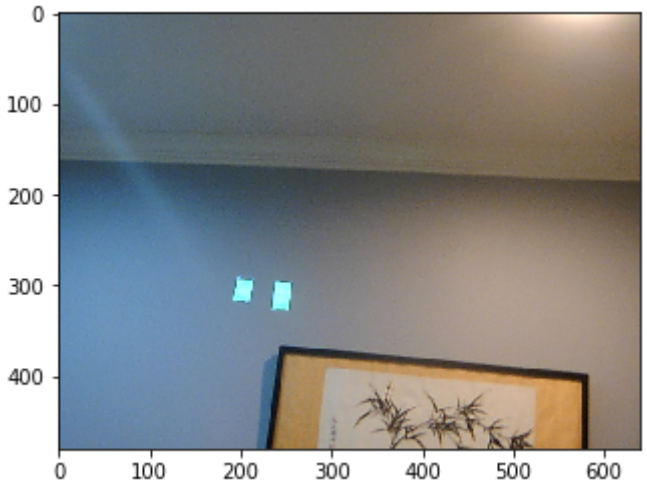
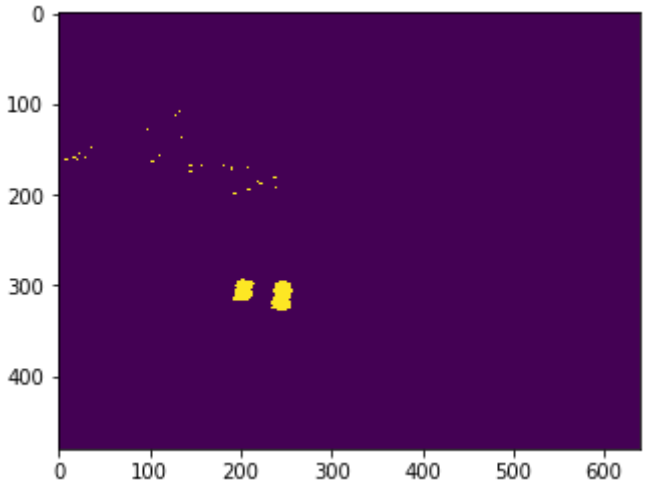
In [180]:
```
%matplotlib inline
import matplotlib.image as mpimg
from matplotlib import pyplot as plt
import cv2
import numpy as np

# Get still image from disk
img = mpimg.imread('/vagrant/notebooks/sony_still_of_tape.jpg')
plt.imshow(img)
```

Out[180]: <matplotlib.image.AxesImage at 0x7fde0fa493c8>



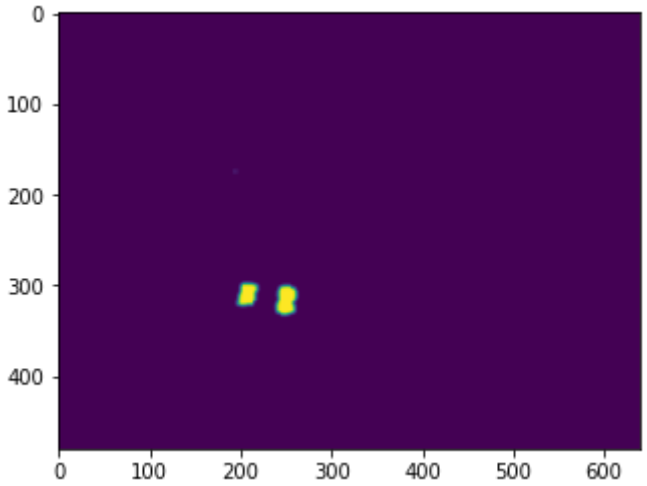
In [181]:
```
# Define the lower and upper boundaries of the color we need
# This is not easy to do, I'll try to explain some techniques in another
 document
lower = np.array([85,50,50])  # HSV
upper = np.array([93,255,255])

# Convert the image from RGB to HSV color space.  This is required for t
he next operation.
tape_hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

# Create a new image that contains yellow where the color was detected,
 otherwise purple
res = cv2.inRange(tape_hsv, lower, upper)
plt.imshow(res)
```

Out[181]: <matplotlib.image.AxesImage at 0x7fde0fa06128>



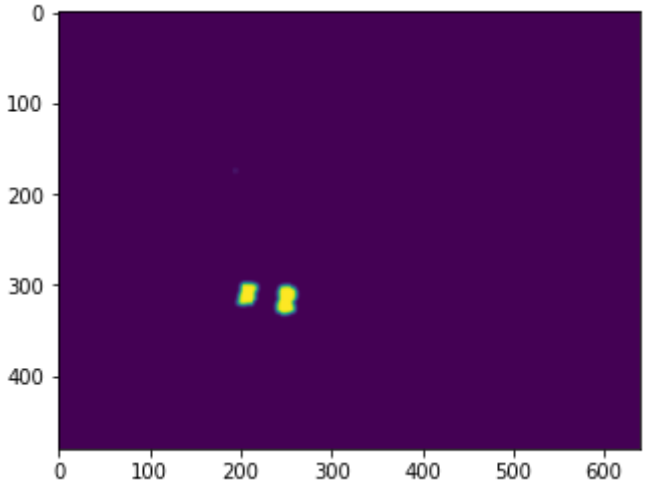
In [182]:
```
# A kernal is like a matrix that is used in the morphology operation
# See https://en.wikipedia.org/wiki/Kernel_(image_processing) if interes
ted
kernel = np.ones((4,4),np.uint8)

# The morphological 'open' operation is described here:
# https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.htm
l
# It helps remove noise and jagged edges, note how the stray speckles ar
e removed!
opened = cv2.morphologyEx(blur, cv2.MORPH_OPEN, kernel)
plt.imshow(opened)
```

Out[182]: <matplotlib.image.AxesImage at 0x7fde0fa3d7f0>



In [183]:
```
# Blurring operation helps forthcoming findContours operation work bette
r
blur = cv2.blur(opened, (3,3))
plt.imshow(blur)
```

Out[183]: <matplotlib.image.AxesImage at 0x7fde0f9f8048>



In [184]:
```
# Find contours
(_, cnts, _) = cv2.findContours(blur.copy(), cv2.RETR_EXTERNAL, cv2.CHAI
N_APPROX_SIMPLE)
```

In [185]:
```
# Sometimes the contours operation will find more than two contours
# But if we did all our preliminary operations properly, then the two co
ntours we need will be
# the *largest* contours in the set of contours.  The sorted operation b
elow sorts
# the cnts array of contours by area, so the first two contours will be
 the largest

cnt1 = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
cnt2 = sorted(cnts, key = cv2.contourArea, reverse = True)[1]

# Draw a minimum area rectangle around each contour
rect1 = np.int32(cv2.boxPoints(cv2.minAreaRect(cnt1)))
rect2 = np.int32(cv2.boxPoints(cv2.minAreaRect(cnt2)))

# Draw the contours in red (255, 0, 0) on top of our original image
cv2.drawContours(img, [rect1], -1, (255, 0, 0), 2)
cv2.drawContours(img, [rect2], -1, (255, 0, 0), 2)
plt.imshow(img)
```

Out[185]: <matplotlib.image.AxesImage at 0x7fde0f9b2be0>