In [4]:

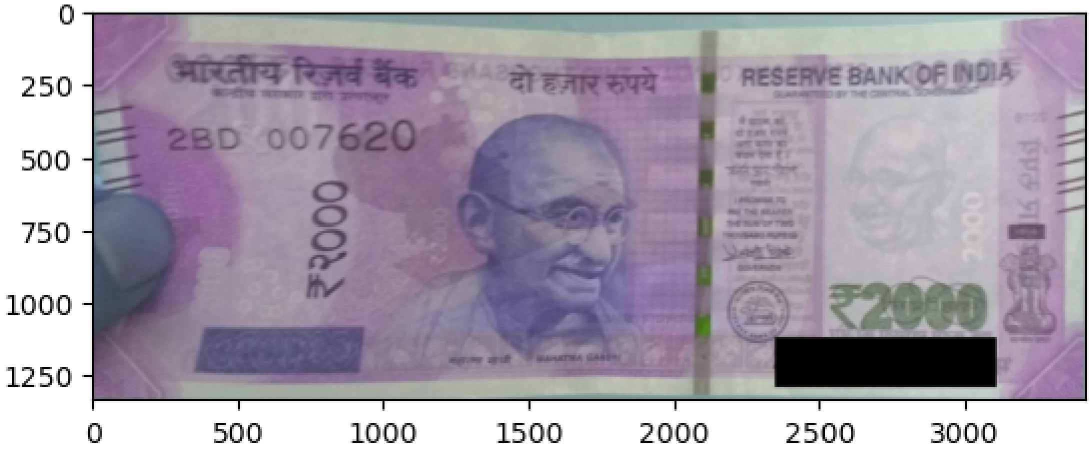
```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

A = cv2.imread(r'C:\Users\N Kavya\Downloads\fake-currency-detection\real.jpg')
P = cv2.imread(r'C:\Users\N Kavya\Downloads\fake-currency-detection\fake.jpg')

plt.imshow(A)
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x1fe4595f950>
```



[redacted]

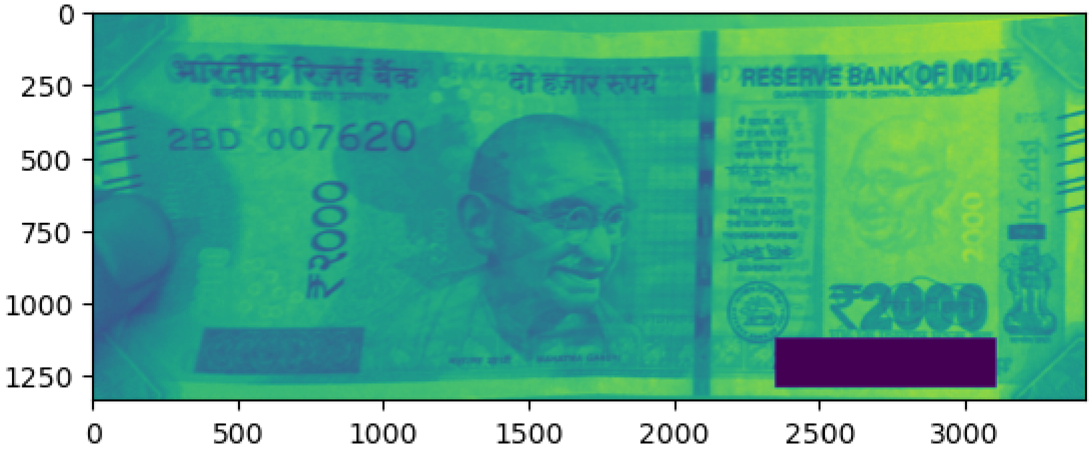
In [5]:

```python
a = cv2.cvtColor(A, cv2.COLOR_BGR2GRAY)
p = cv2.cvtColor(P, cv2.COLOR_BGR2GRAY)

plt.imshow(a)
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x1fe4b8596d0>
```



[redacted]

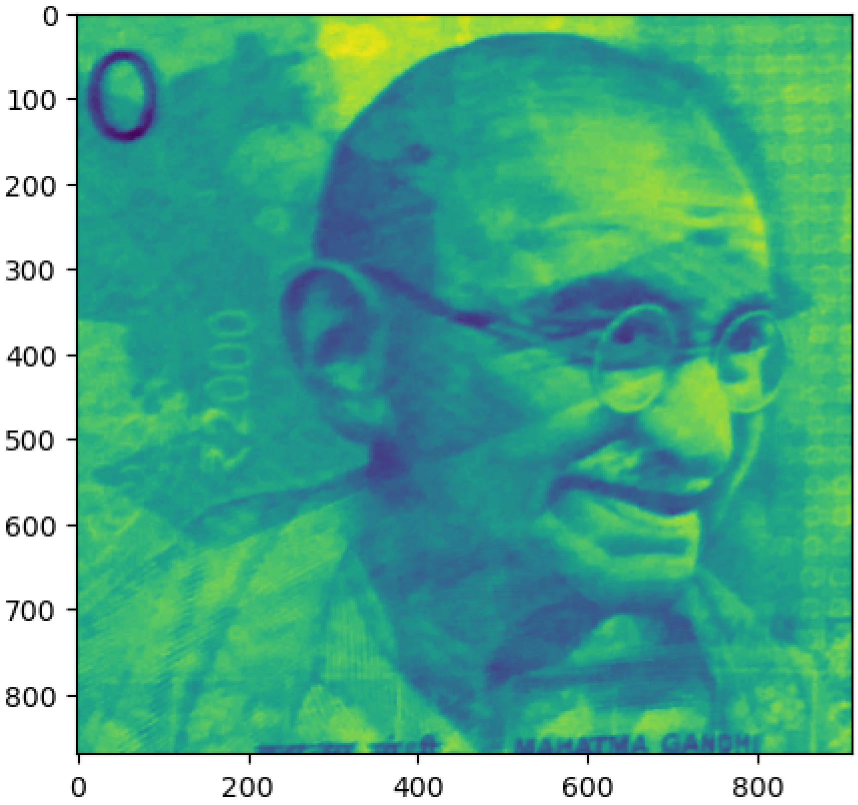
In [6]:

```python
a2tr = a[330:1200, 1016:1927]

plt.imshow(a2tr)
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x1fe4b8a96d0>
```



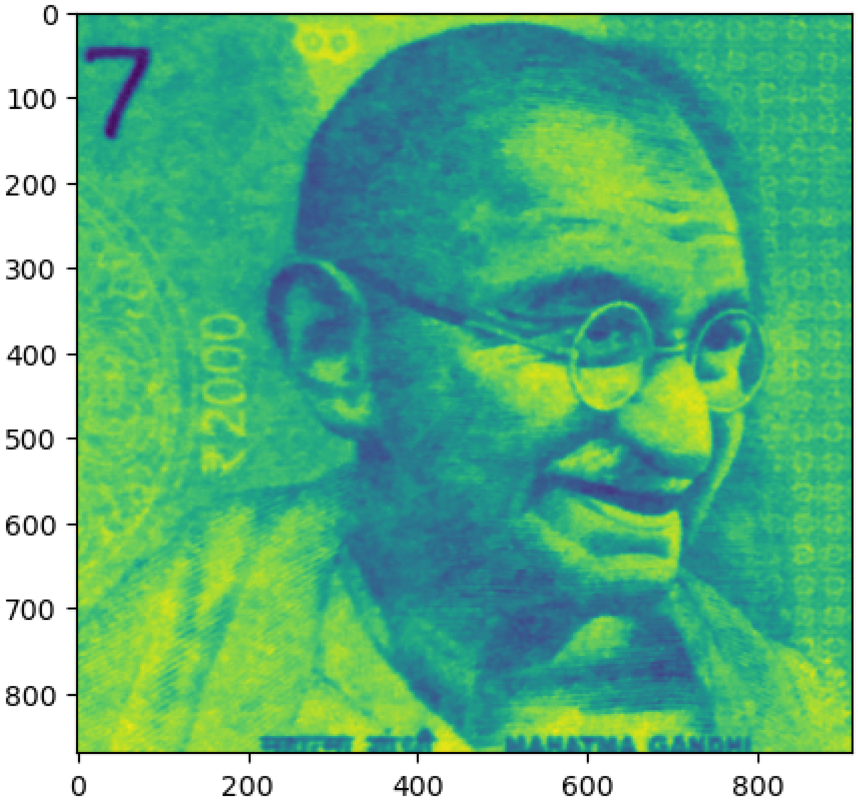
In [7]:

```
b2tr = p[170:1040, 716:1627]

plt.imshow(b2tr)
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x1fe48ca2c50>
```



In [8]:

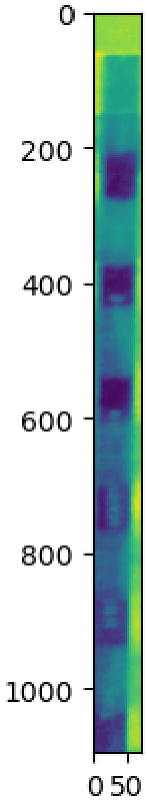
```
print(a.shape)
a2_str = a[5:1100, 2080:2151]

plt.imshow(a2_str)
```

```
(1332, 3416)
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x1fe4c5c96d0>
```



In [9]:

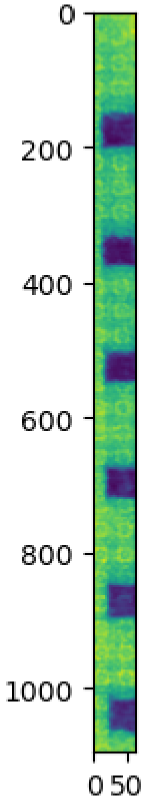
```
print(p.shape)
p2_str = p[5:1100, 1666:1729]

plt.imshow(p2_str)
```

```
(1100, 3000)
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x1fe444896d0>
```

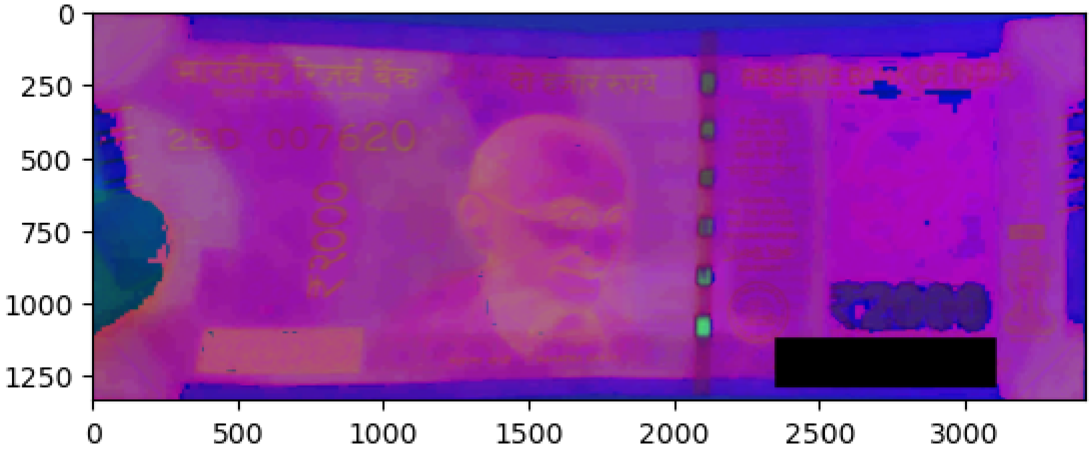


In [10]:
```
hsvImageReal = cv2.cvtColor(A, cv2.COLOR_BGR2HSV)

hsvImageFake = cv2.cvtColor(P, cv2.COLOR_BGR2HSV)
```

In [11]:
```
plt.imshow(hsvImageReal)
```

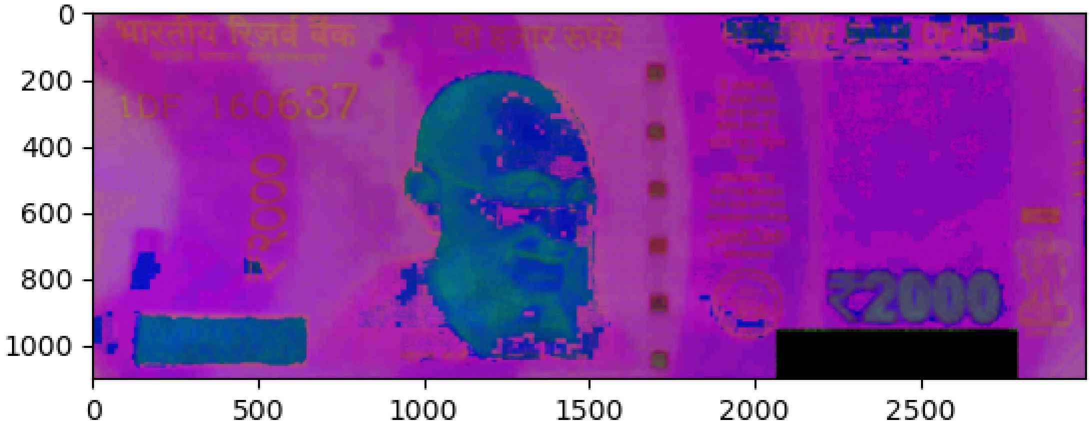
Out[11]:
```
<matplotlib.image.AxesImage at 0x1fe4c6216d0>
```



[redacted]

In [12]:
```
plt.imshow(hsvImageFake)
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x1fe4c5b16d0>
```

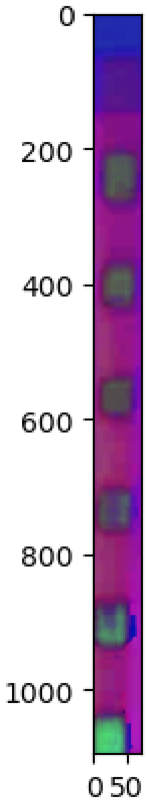


[redacted]

In [13]:
```
croppedImageReal = hsvImageReal[5:1100, 2080:2151]

plt.imshow(croppedImageReal)
```

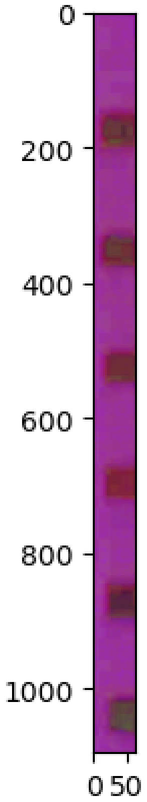
Out[13]:
```
<matplotlib.image.AxesImage at 0x1fe48db0610>
```



In [14]:
```
croppedImageFake = hsvImageFake[5:1100, 1666:1729]

plt.imshow(croppedImageFake)
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x1fe504b7950>
```



In [15]:
```
satThresh = 0.3
valThresh = 0.9

g = croppedImageReal[:,:,1]>satThresh
```

```
h = croppedImageReal[:,:,2] < valThresh

g1 = croppedImageFake[:,:,1]>satThresh
h1 = croppedImageFake[:,:,2] < valThresh

BWImageReal = g&h
BWImageFake = g1&h1
```

In [16]:

```
def bwareaopen(img, min_size, connectivity=8):

    # Find all connected components (called here "labels")
    num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(
        img, connectivity=connectivity)

    # check size of all connected components (area in pixels)
    for i in range(num_labels):
        label_size = stats[i, cv2.CC_STAT_AREA]

        # remove connected components smaller than min_size
        if label_size < min_size:
            img[labels == i] = 0

    return img
```

In [17]:

```
binr = cv2.threshold(a2_str, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# invert the image
invert = cv2.bitwise_not(binr)

# use morph gradient
BWImageCloseReal = cv2.morphologyEx(invert, cv2.MORPH_GRADIENT, kernel)
```

In [18]:

```
binr2 = cv2.threshold(p2_str, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel2 = np.ones((3, 3), np.uint8)

# invert the image
invert2 = cv2.bitwise_not(binr2)

# use morph gradient
BWImageCloseFake = cv2.morphologyEx(invert2, cv2.MORPH_GRADIENT, kernel2)
```

In [19]:

```
areaopenReal = bwareaopen(BWImageCloseReal, 15);

areaopenFake = bwareaopen(BWImageCloseFake, 15);
```

In [20]:

```
bw = areaopenReal

labels = np.zeros(bw.shape)
countReal = cv2.connectedComponentsWithStats(bw, labels,8);
```

In [21]:

```
bw2 = areaopenFake

labels2 = np.zeros(bw2.shape)
countFake = cv2.connectedComponentsWithStats(bw2, labels2,8);
```

In [22]:

```
def corr2(A, B):

    A_mA = A - A.mean(1)[:, None]
    B_mB = B - B.mean(1)[:, None]

    # Sum of squares across rows
    ssA = (A_mA**2).sum(1)
    ssB = (B_mB**2).sum(1)

    # Finally get corr coeff
    return np.dot(A_mA, B_mB.T) / np.sqrt(np.dot(ssA[:, None],ssB[None]))
```

In [23]:

```
co=corr2 (a2tr, b2tr)

if (co.any()>=0.5):
    print ('correlevance of transparent gandhi > 0.5')
    if (countReal[0] == countFake[0] ):
        print ('currency is legitimate')
    else:
        print ('green strip is fake')
else:
    print ('correlevance of transparent gandhi < 0.5')
    print ('currency is fake')
```

```
correlevance of transparent gandhi > 0.5
green strip is fake
```

In [ ]: