Week 2 assignment done by SONAM RANI

Import the necessary libraries

In [15]:

```
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import os
import sys
import cv2
```

Function to initialise the mean and variance images

In [16]:

```
def initBackground(initImage):
    img_arr = cv2.imread(initImage)
    mean = img_arr
    variance = 9*np.ones(img_arr.shape)
    return(mean,variance)
```

Classify images into foreground and background pixels using a Chebyshev inequality based classifier¶

In [17]:

```
def ForegroundDetection(img_file,mean,variance,lmda):
    img = cv2.imread(img_file)
    d = img-mean
    y = variance*(lmda**2)
    d_2 = np.square(d)
    I = d_2 - y
    mask = np.all(I>0,axis=2)
    rI = 255*mask.astype(int)
    rI = rI.astype(np.uint8)
    return(rI)
```

Reduce the image noise using a voting scheme

In [24]:

```
def Voting(rI,eta,m,n):
    r,c = rI.shape
    cI = np.zeros((rI.shape[0],rI.shape[1]))
    for i in range(m,r-1-m):
        for j in range(n,c-1-n):
            img_patch = rI[i-m:i,j-n:j]
            y_unq, counts = np.unique(img_patch,return_counts=True)
            if len(counts)==1 and y_unq[0] == 1:
                cI[i,j] = 255
            if len(counts)>1:
                if counts[1]>eta*m*n:
                    cI[i,j]=255
    cI = cI.astype(np.uint8)
    return cI
```

Update the mean and variance images using a weighted average scheme

In [26]:

```
def meanvarUpdate(cI,img_path,M,V,alpha):
    img = cv2.imread(img_path)
    mean_upd = np.zeros(img.shape)
    var_upd = np.zeros(img.shape)
    d = img - M
    d_2 = np.square(d)
    for i in range(cI.shape[0]):
        for j in range(cI.shape[1]):
            if cI[i,j] == 0:
                mean_upd[i,j,:] = (1-alpha)*M[i,j,:] + alpha*img[i,j,:]
                var_upd[i,j,:] = (1-alpha)*(V[i,j,:] + alpha*d_2[i,j,:])
                var_upd[i,j,:] = np.clip(var_upd[i,j,:],a_min=9,a_max=None)
    return(mean_upd,var_upd)
```

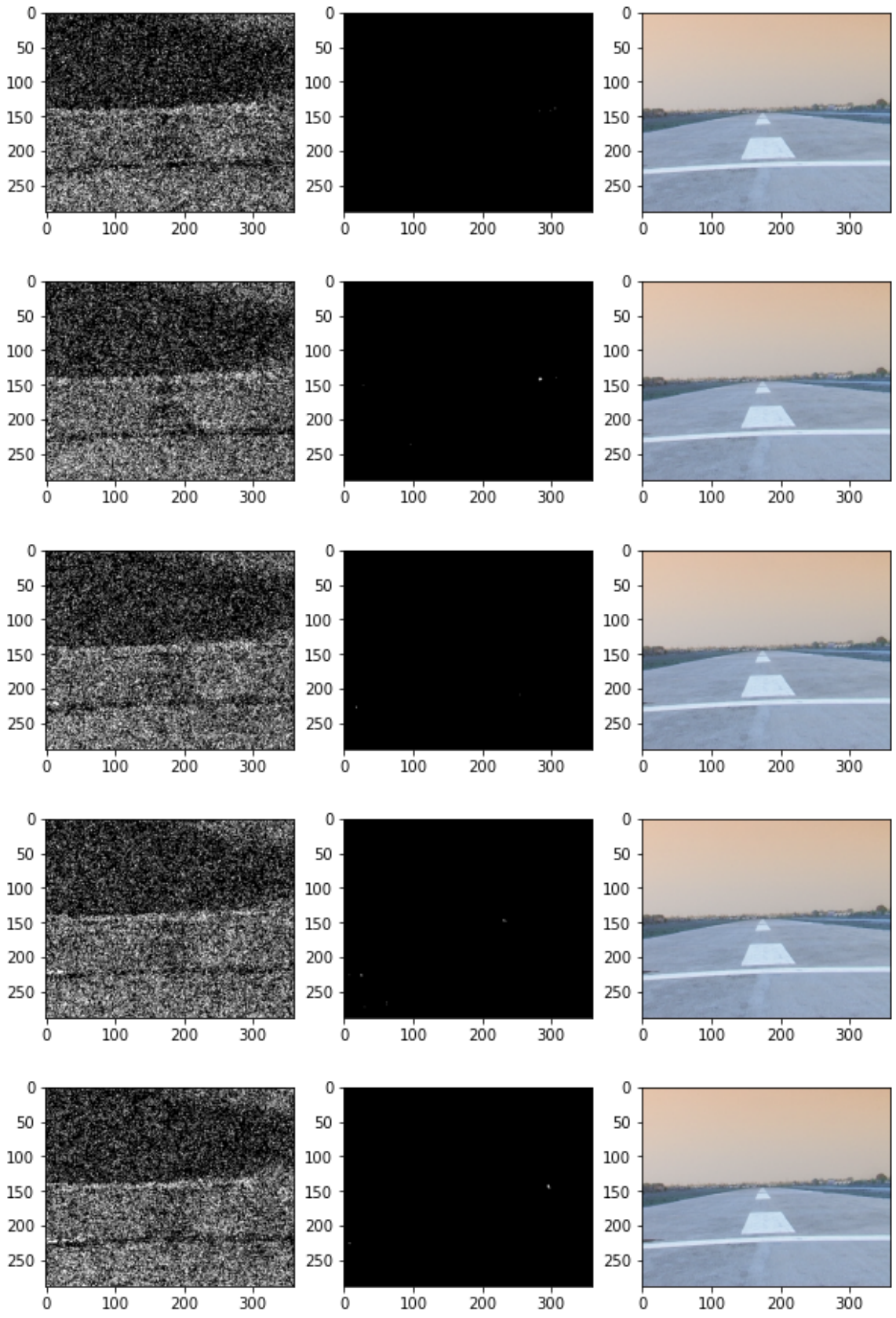
In [27]:

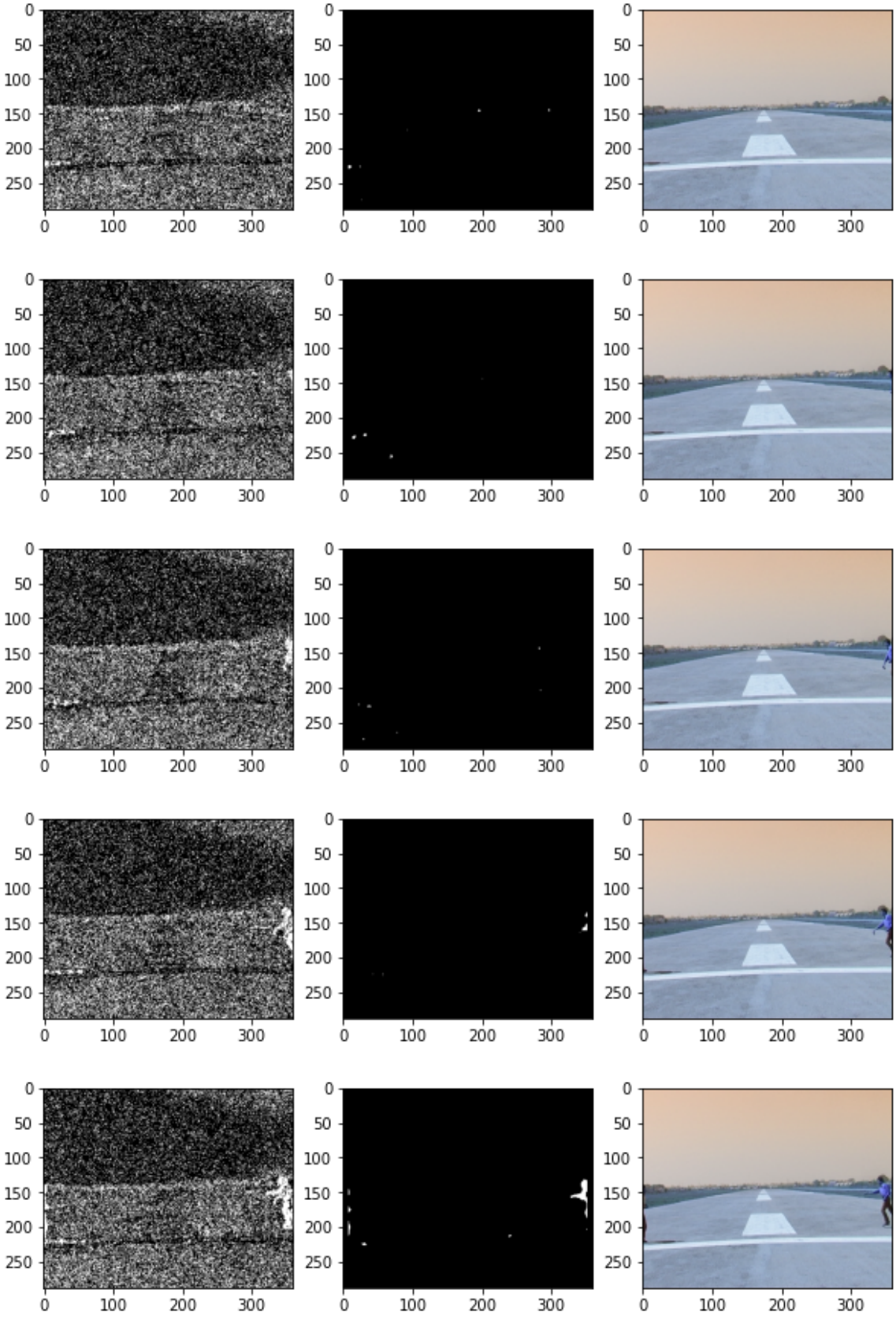
```
def Background_Subtraction(img_dir,lmda,eta,m,n,alpha):
    img_file_name = os.listdir(img_dir)
    initImage = os.path.join(img_dir,img_file_name[0])
    mean, variance = initBackground(initImage)

    for i in range(1,19):
        img_path = os.path.join(img_dir,img_file_name[i])

        fix, ax = plt.subplots(1,3,figsize=(10,10))
        rI = ForegroundDetection(img_path,mean,variance,lmda)
        ax[0].imshow(rI,cmap="gray")

        cI = Voting(rI,eta,m,n)
        mean,variance = meanvarUpdate(cI,img_path,mean,variance,alpha)
        ax[1].imshow(cI,cmap="gray")

        img = cv2.imread(img_path)
        ax[2].imshow(img,cmap="gray")

        plt.show()
    return(mean,variance)
```

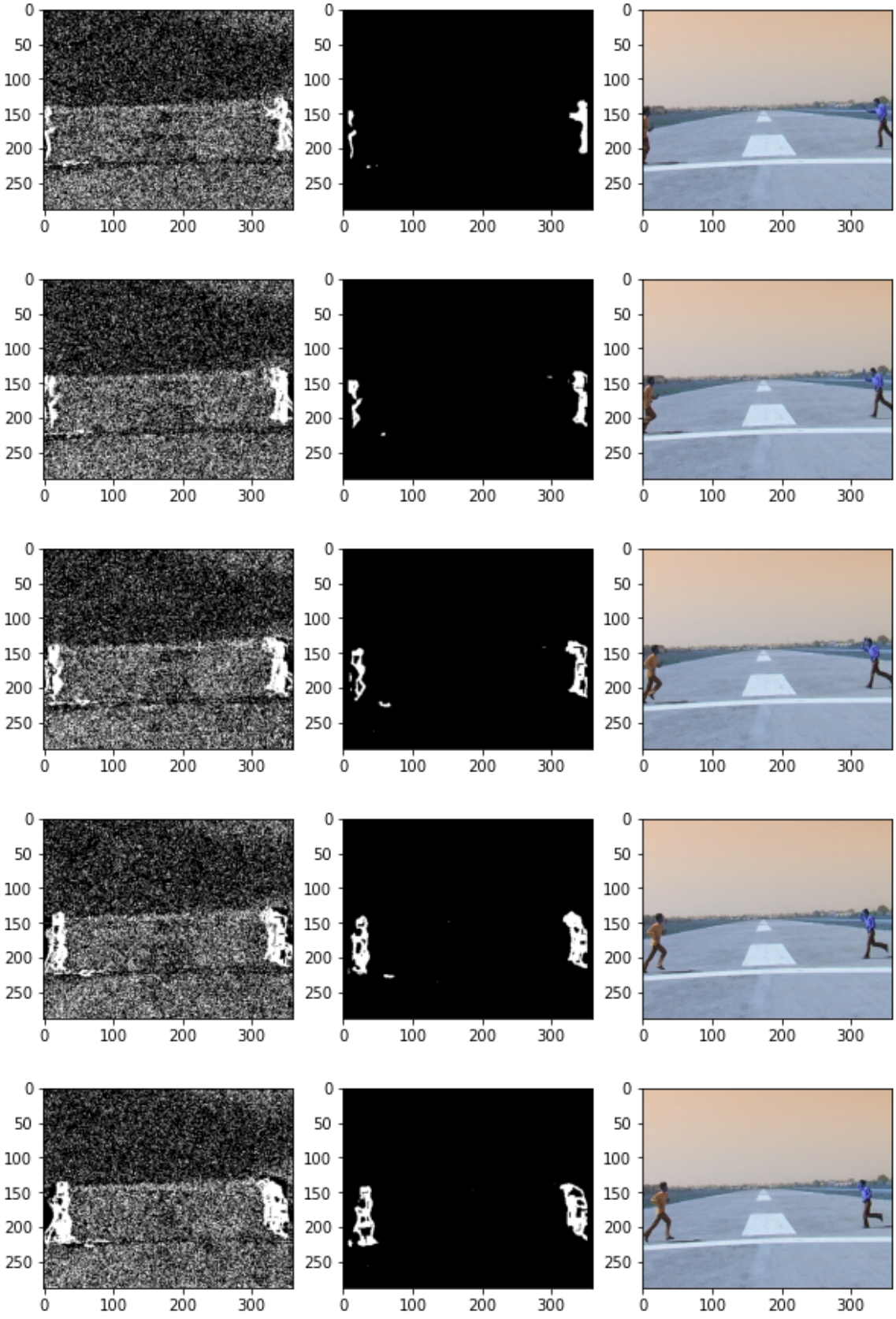
In [32]:

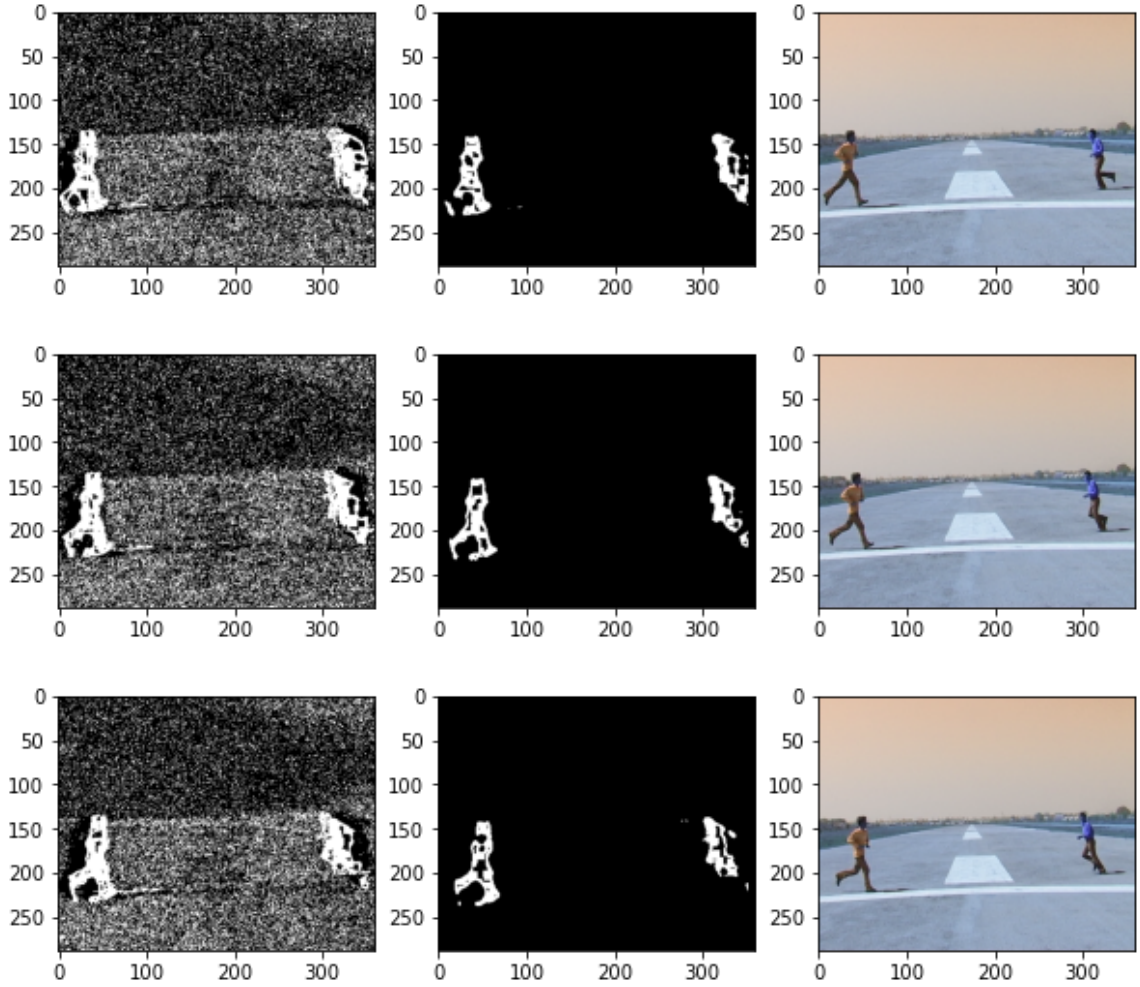
```
v = [i for i in np.arange(0.70,0.90,0.02)]
t = [i for i in np.arange(0.70,0.90,0.02)]

for eta in v:
    for lmda in t:
        print("Using lambda as {} and eta as {}".format(lmda,eta))
        mean,variance = Background_Subtraction("./Images",lmda,eta,8,8,0.8)
        print("\n")
```

```
Using lambda as 0.7 and eta as 0.7
```

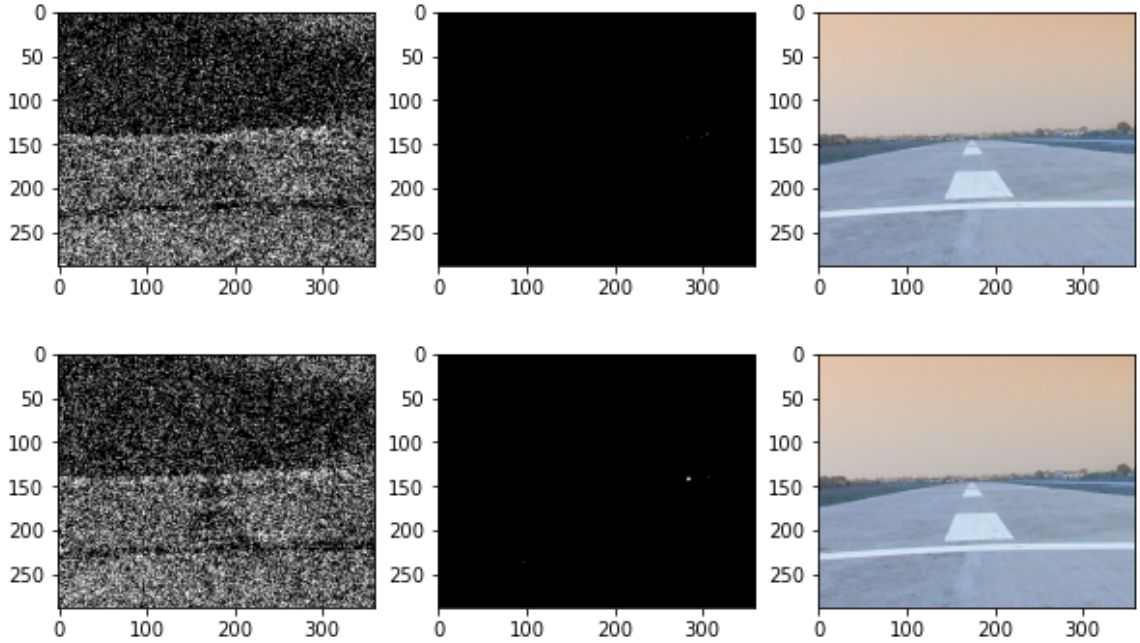


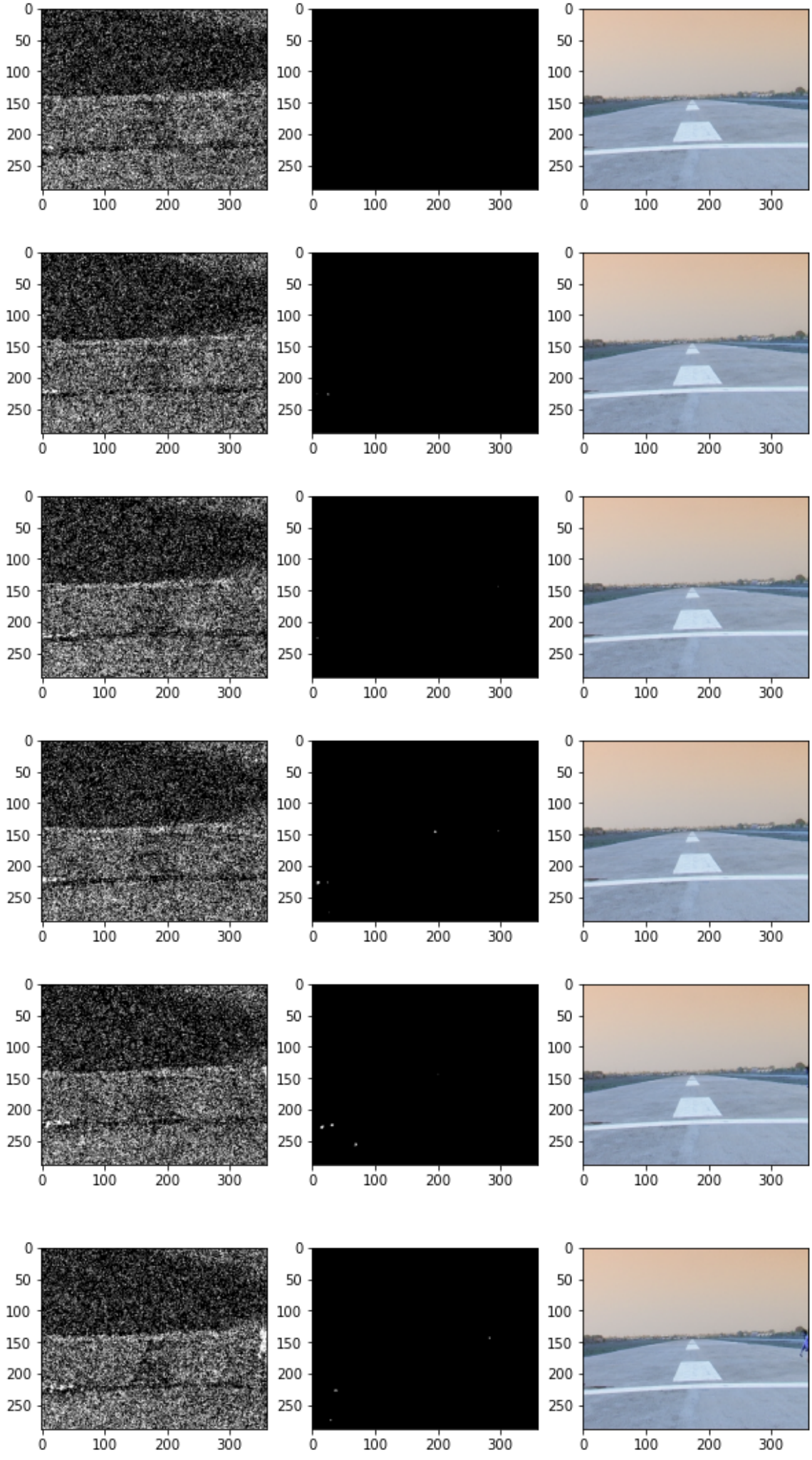

0 50 100 150 200 250
0 100 200 300

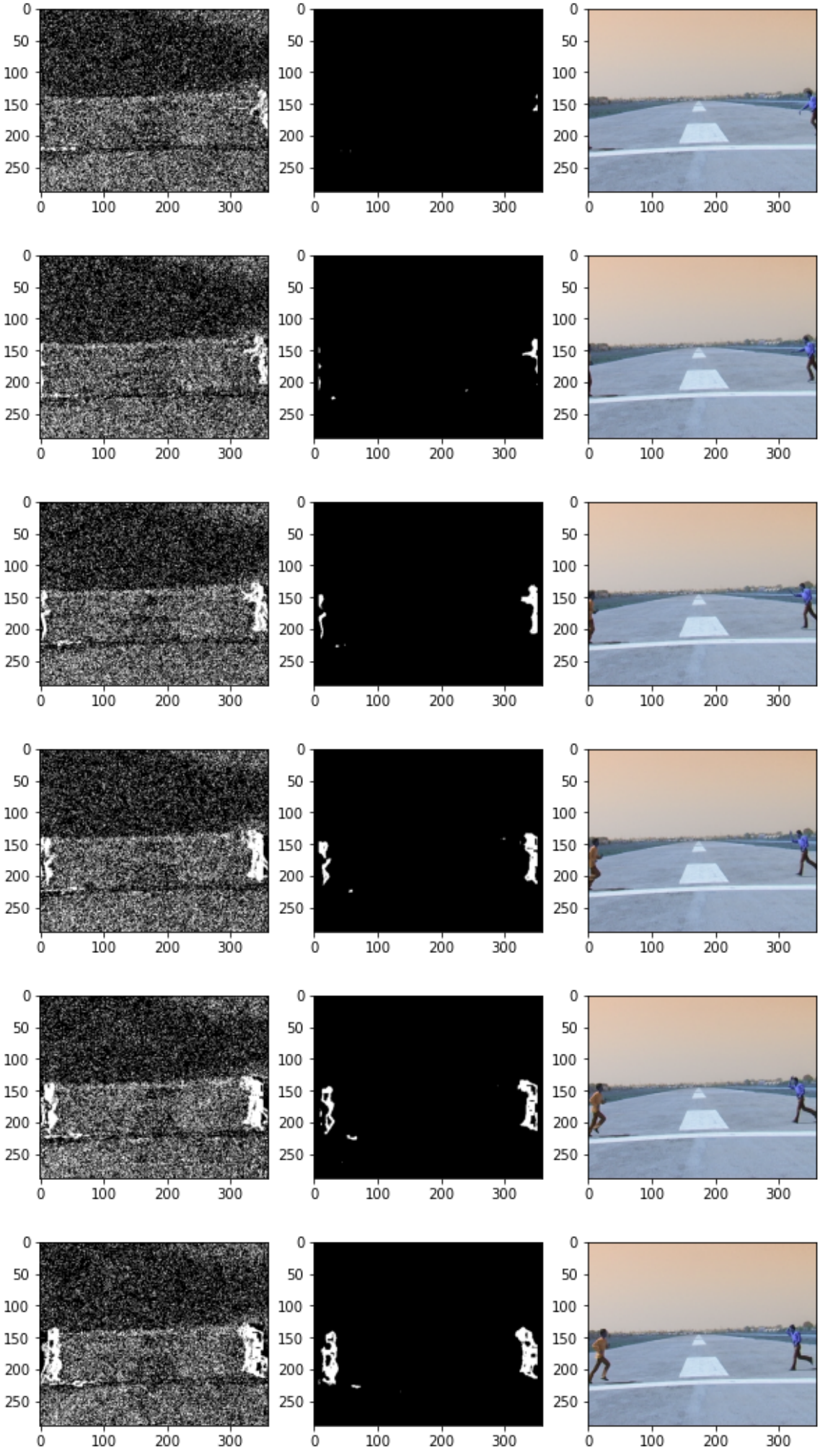


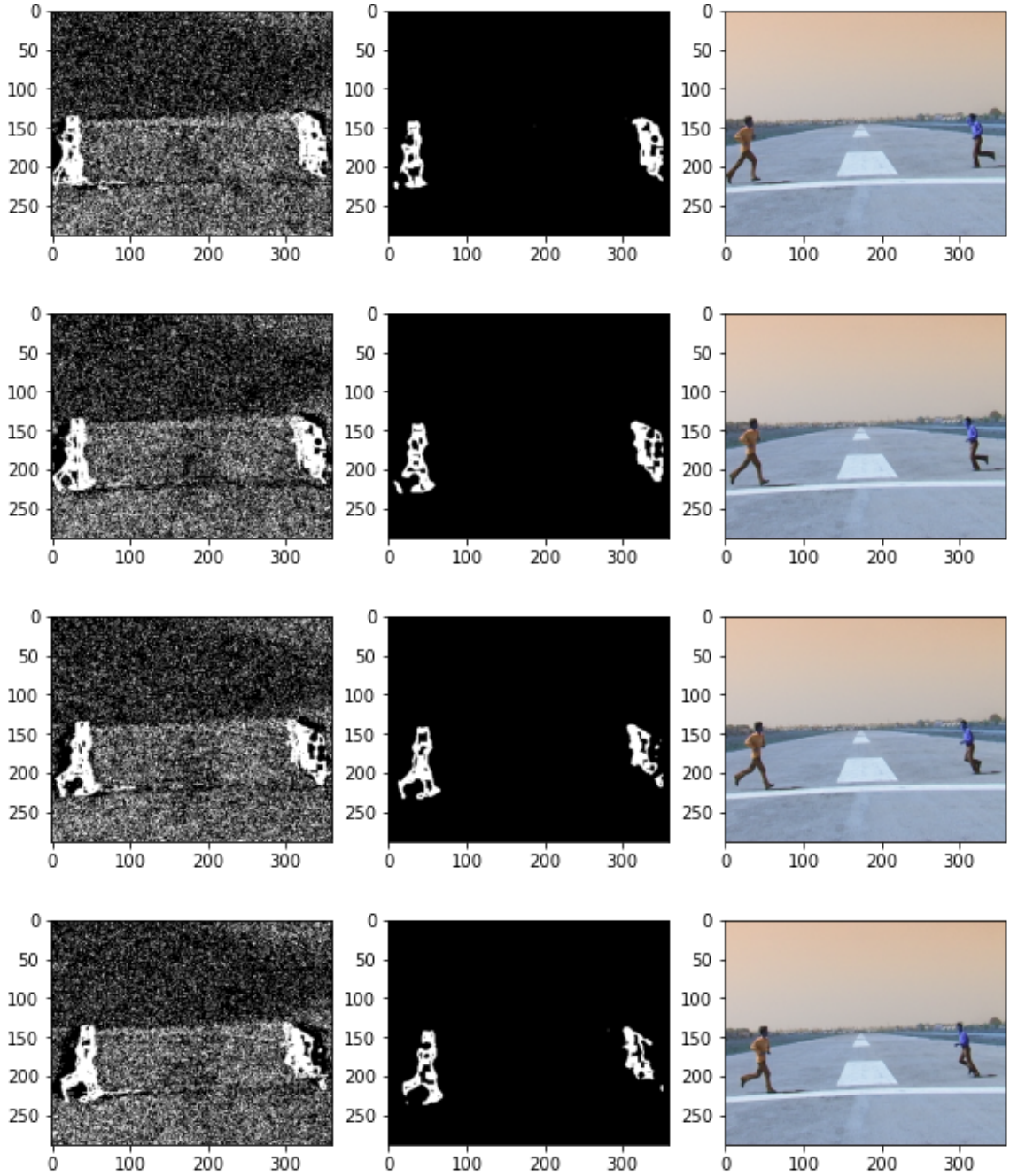


Using lambda as 0.72 and eta as 0.7

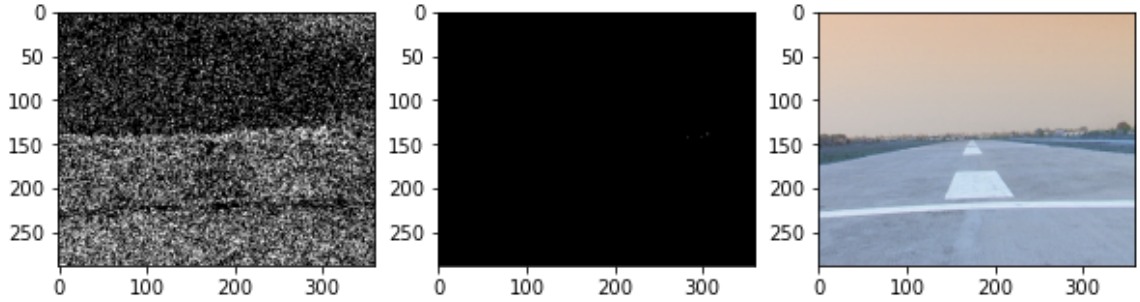


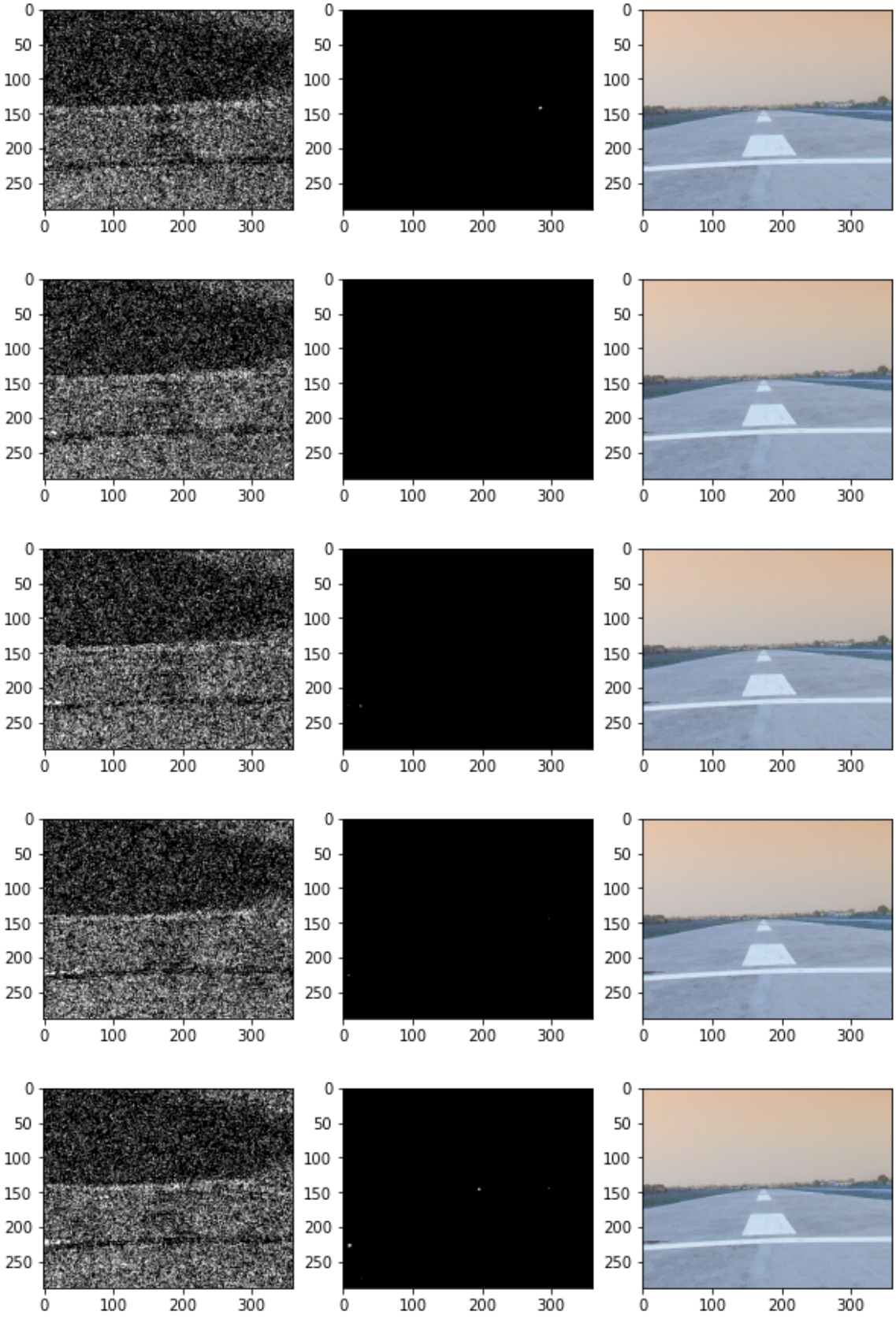


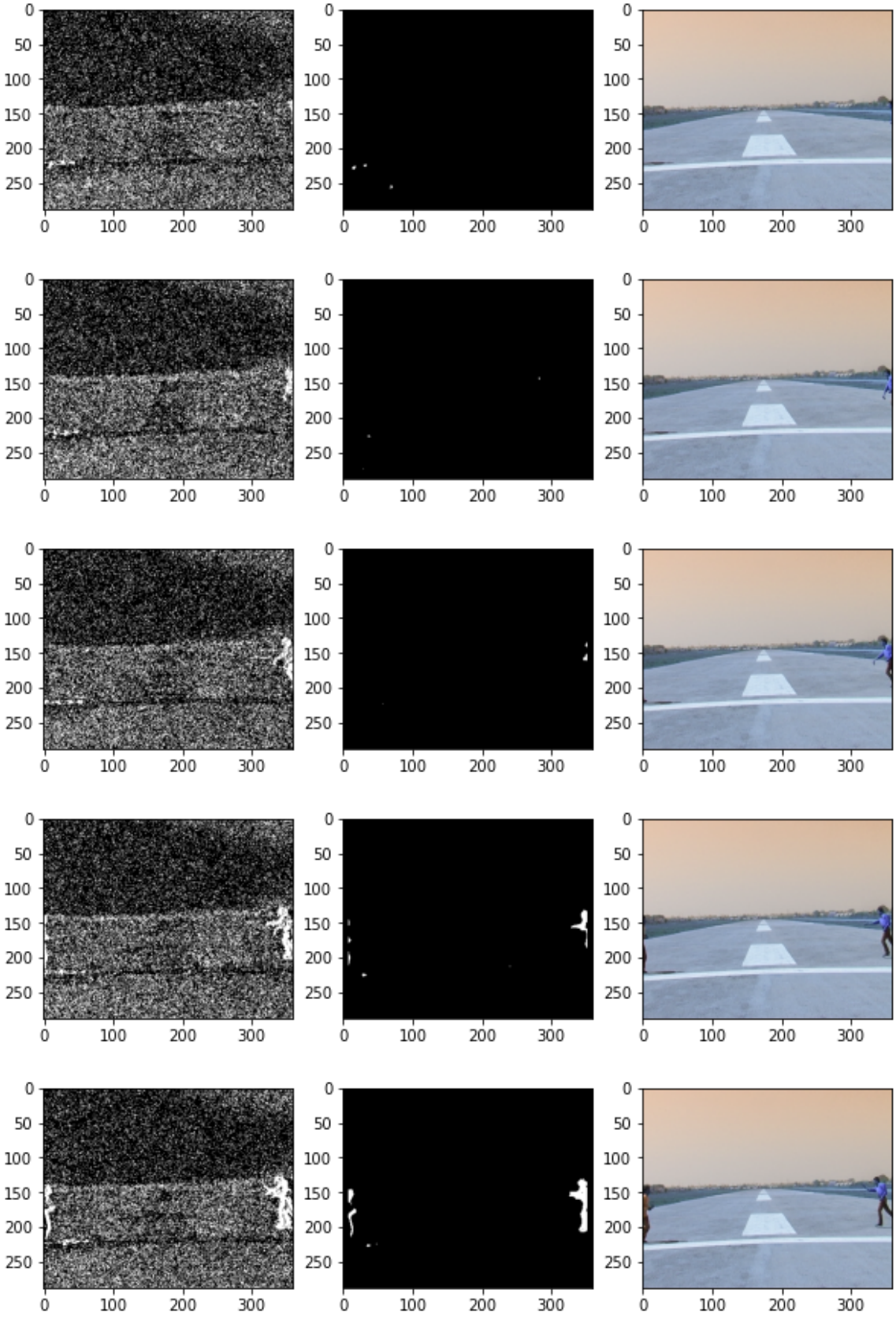

0 50 100 150 200 250
0 100 200 300

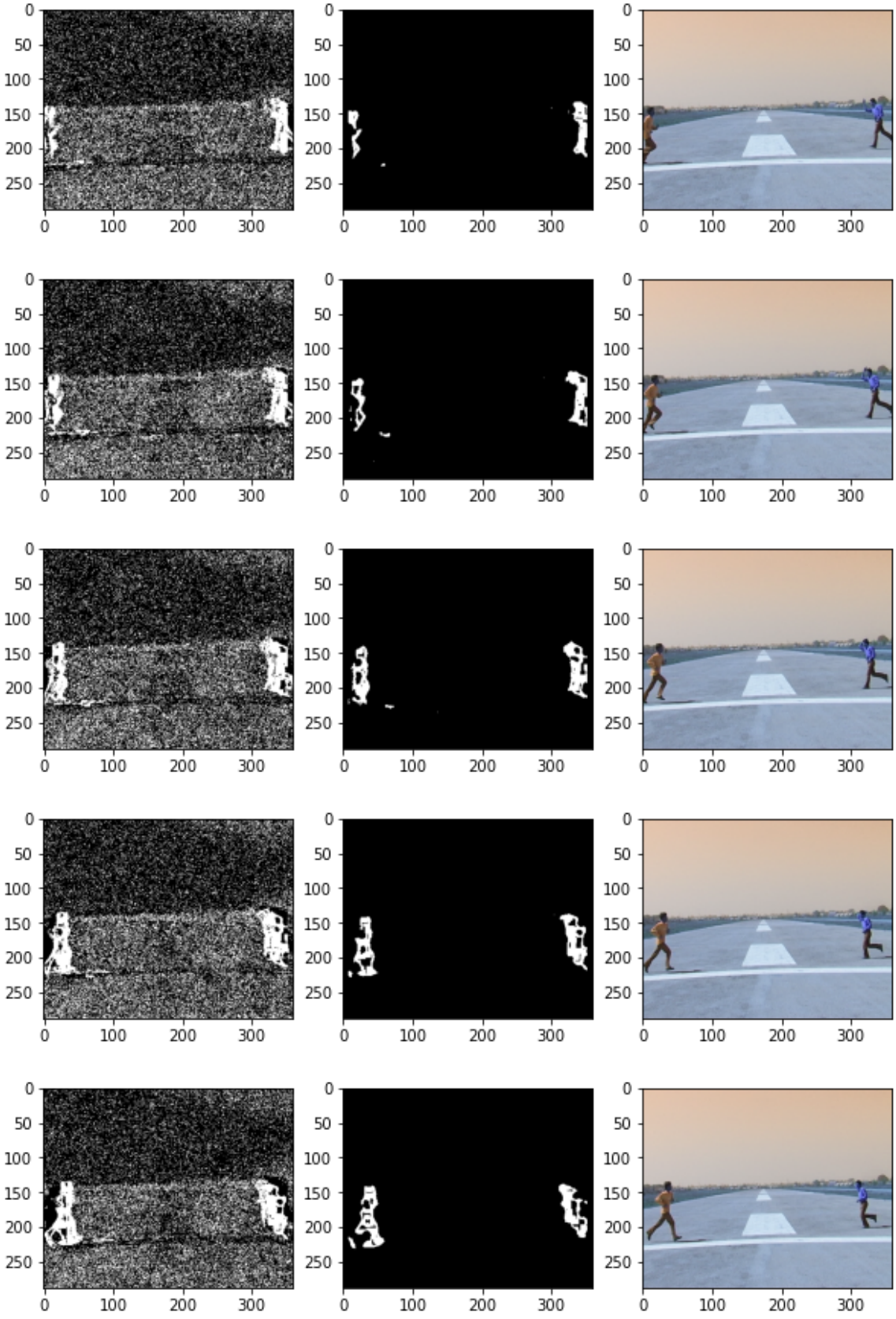


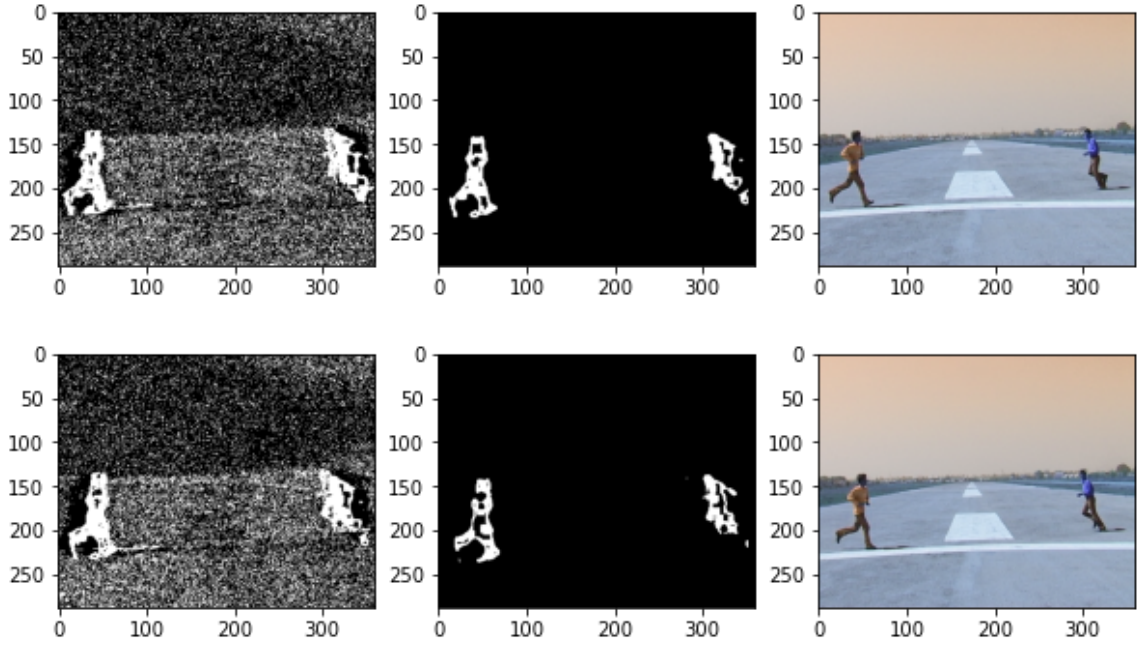
Using lambda as 0.74 and eta as 0.7



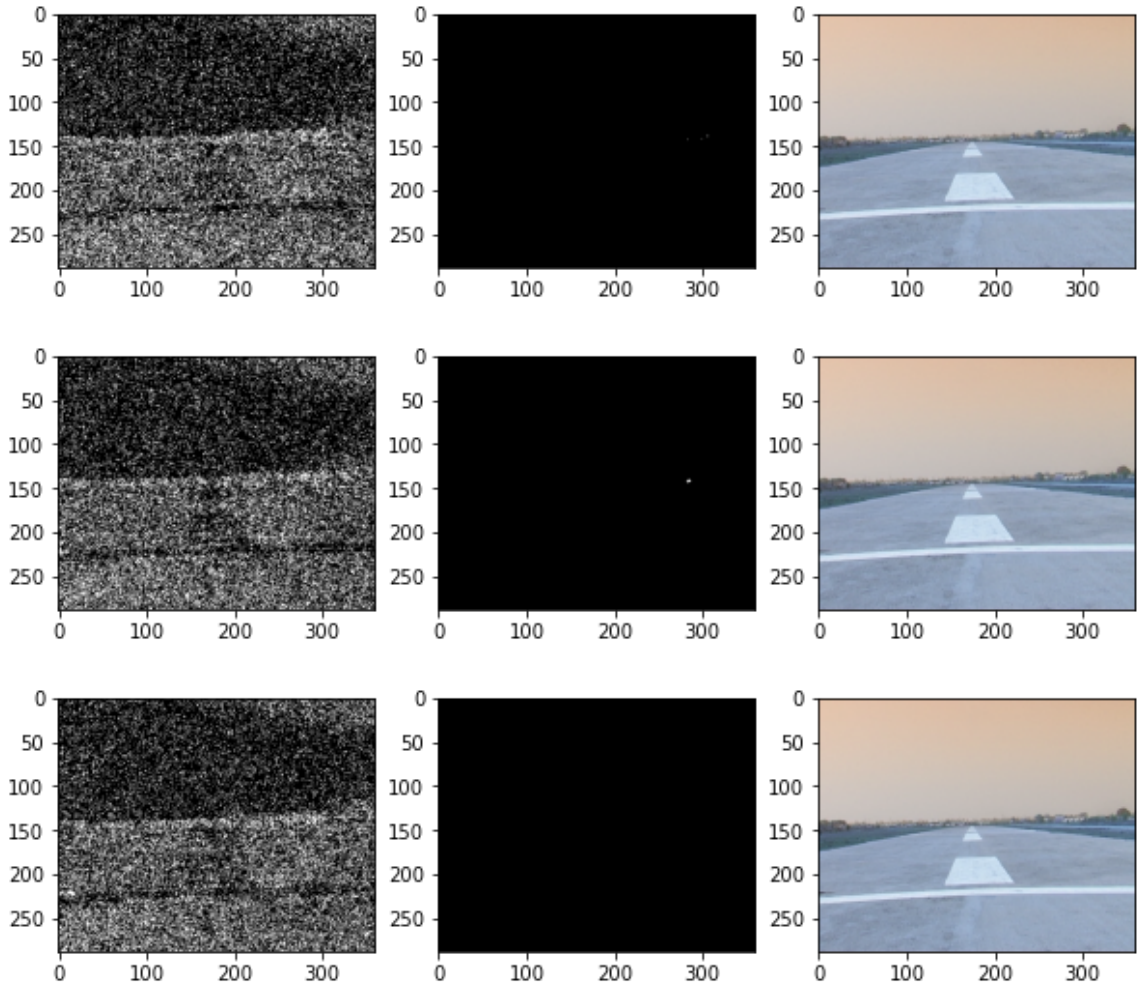

0
50
100
150
200
250
0
100
200
300

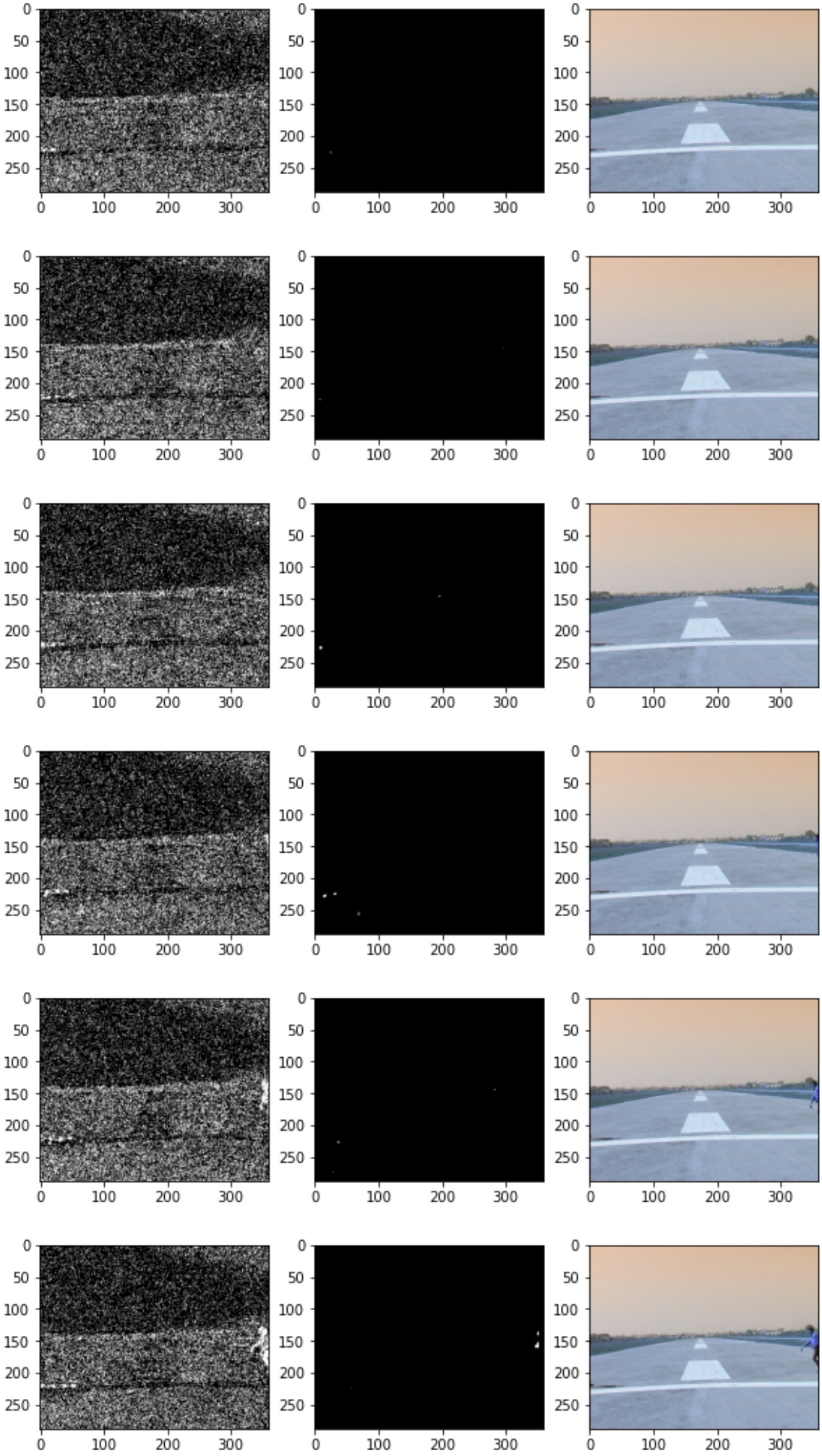


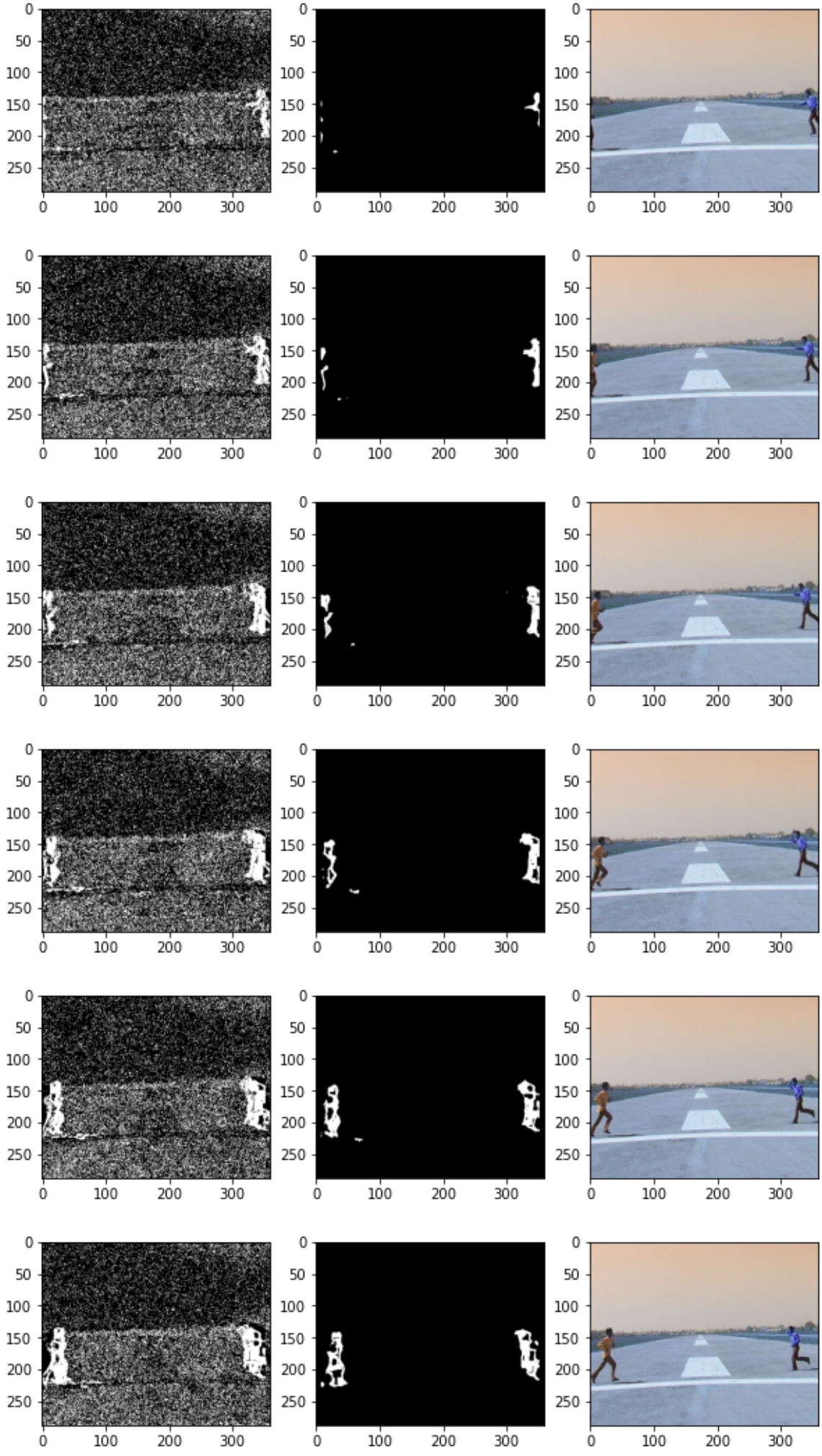


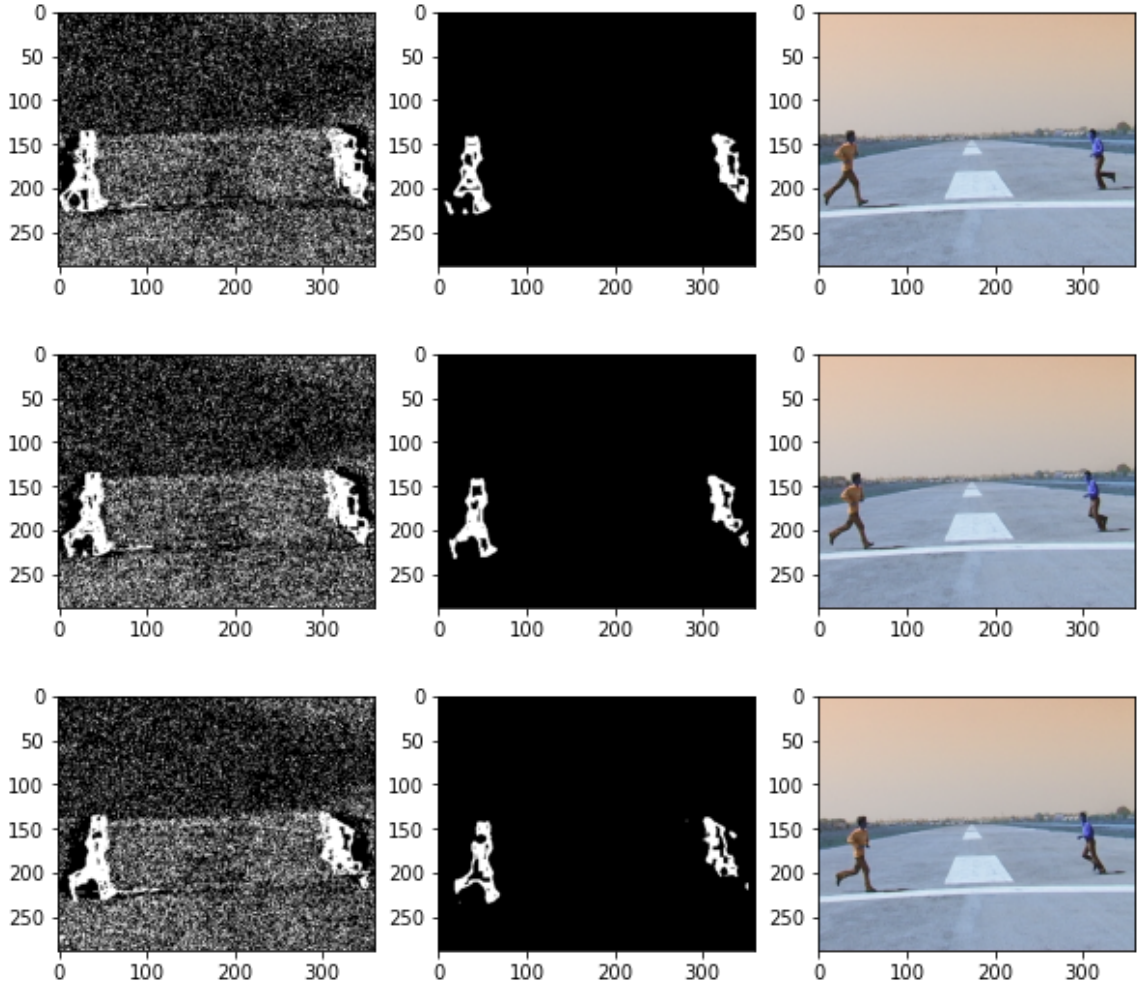


Using lambda as 0.76 and eta as 0.7

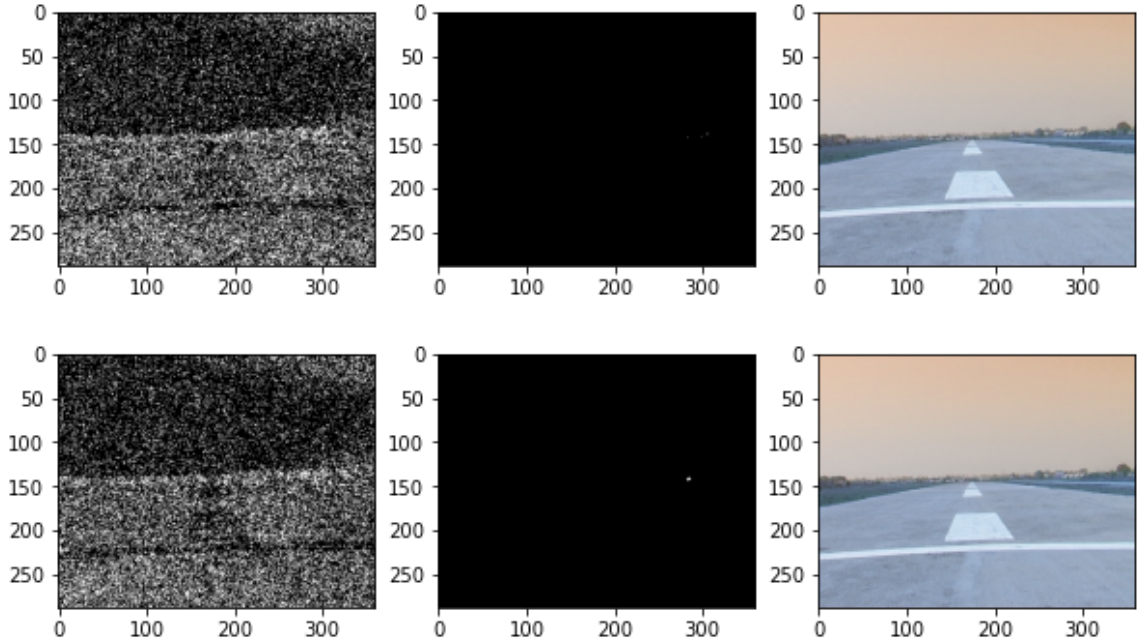


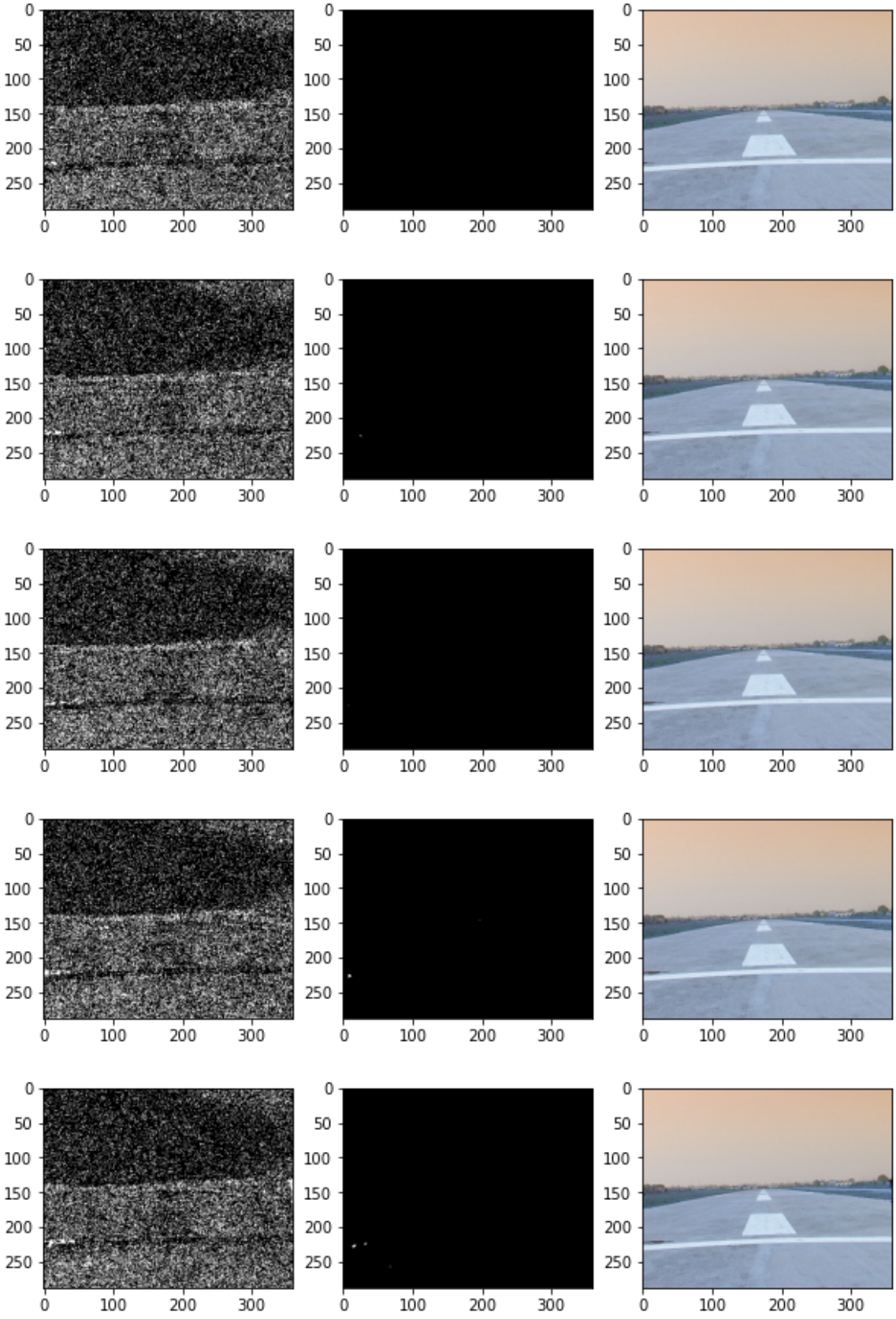


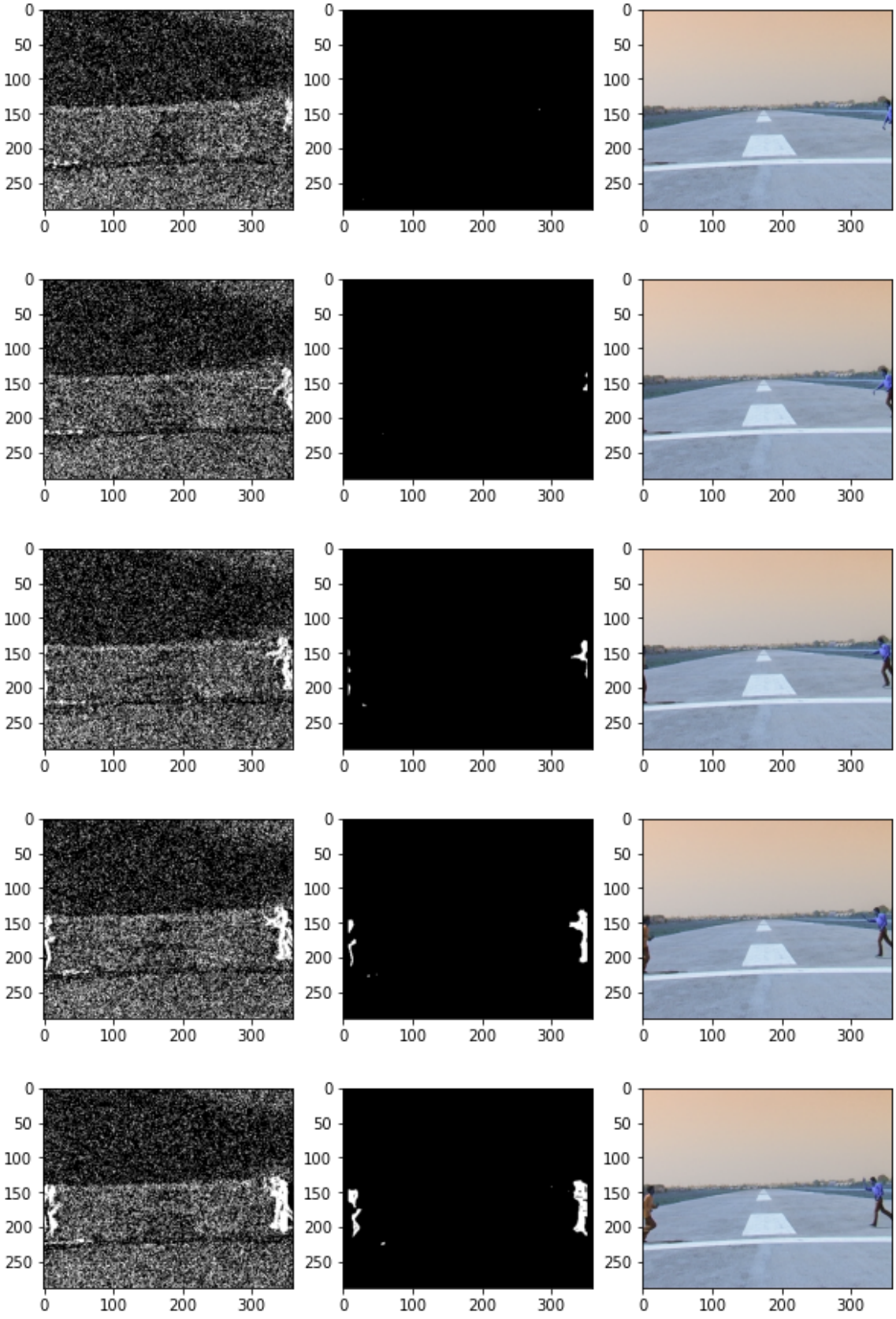


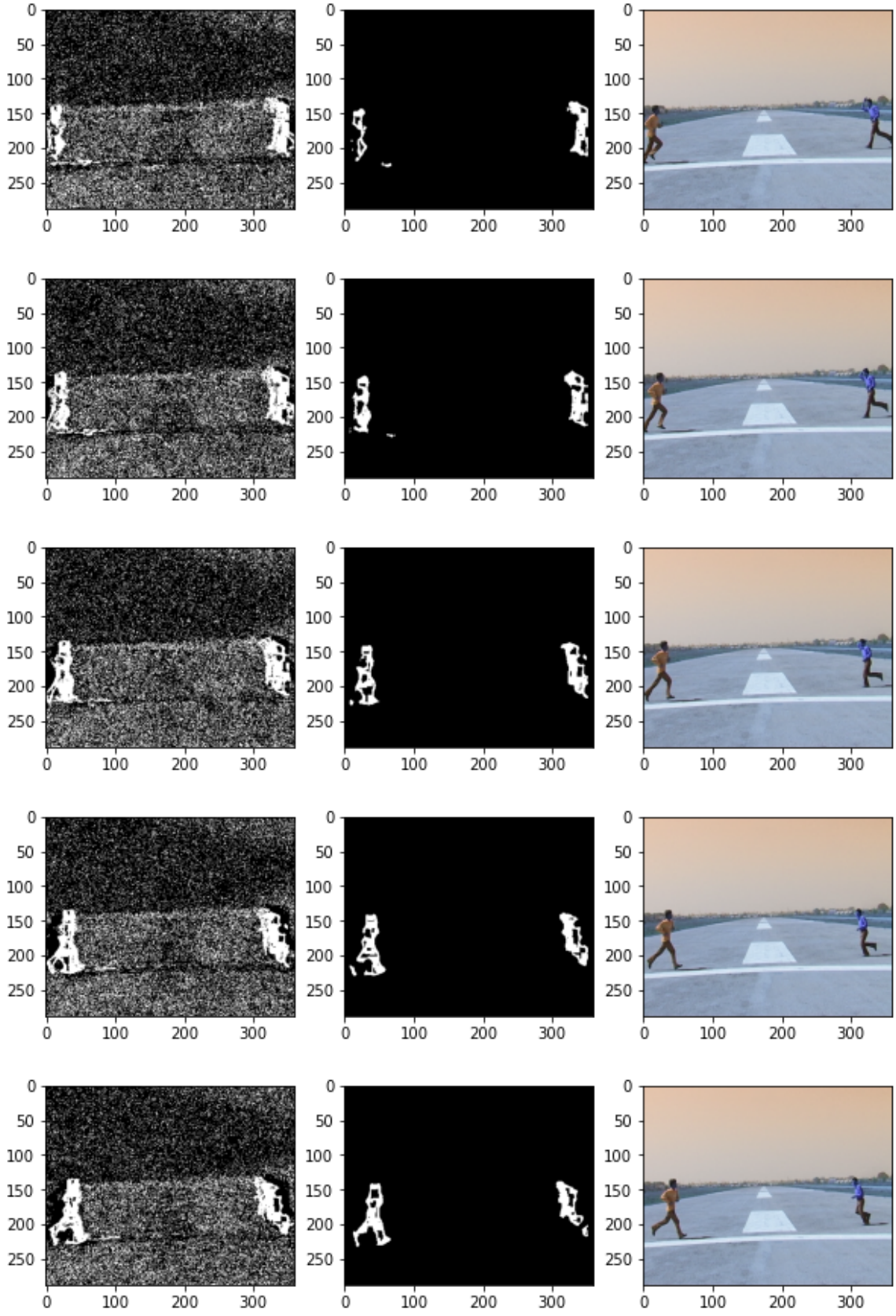


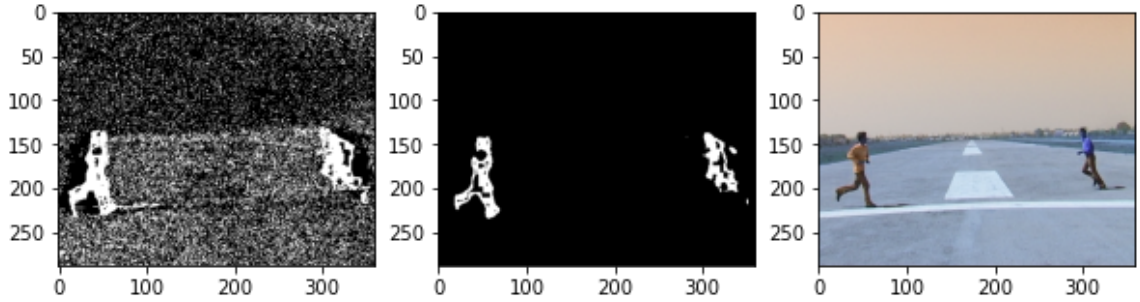
Using lambda as 0.78 and eta as 0.7



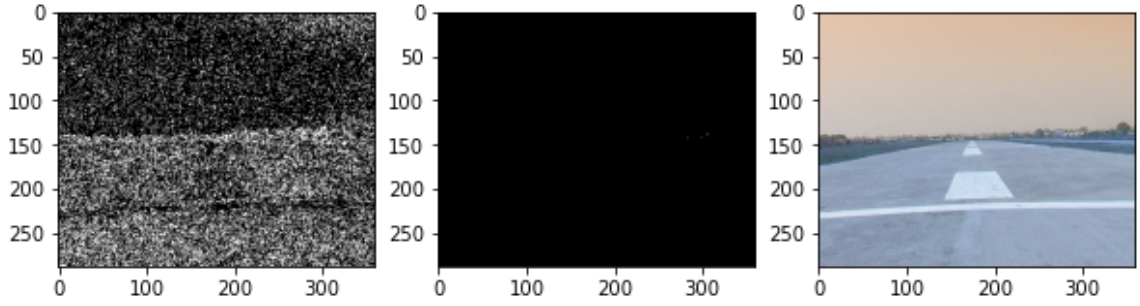

0
50
100
150
200
250
0
100
200
300

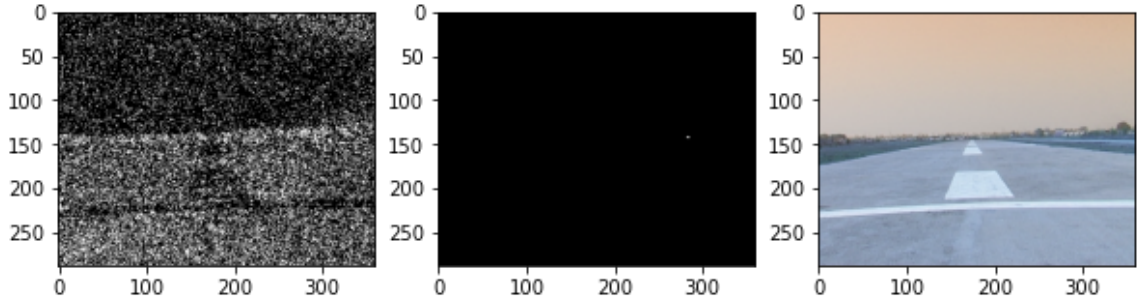

0
50
100
150
200
250
0
100
200
300

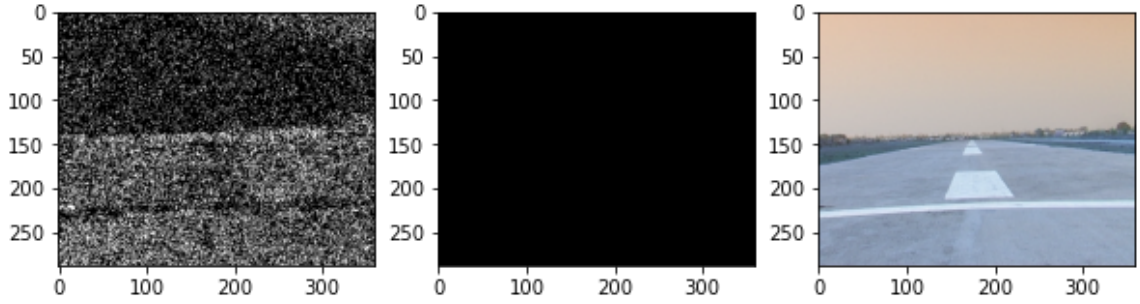


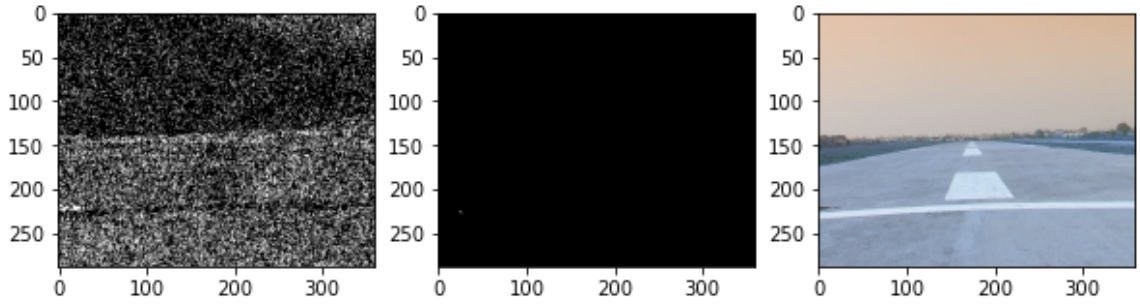


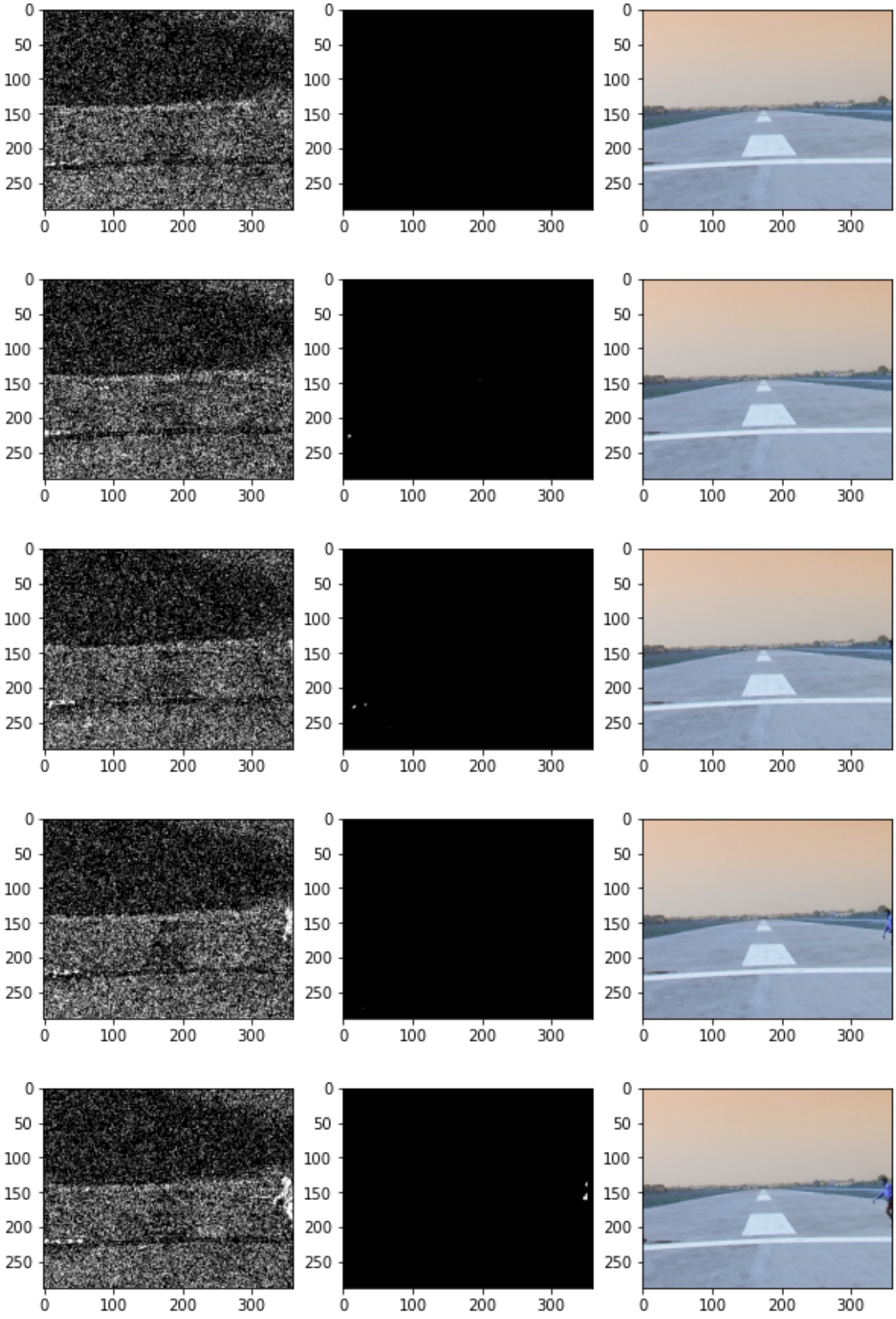
Using lambda as 0.8 and eta as 0.7

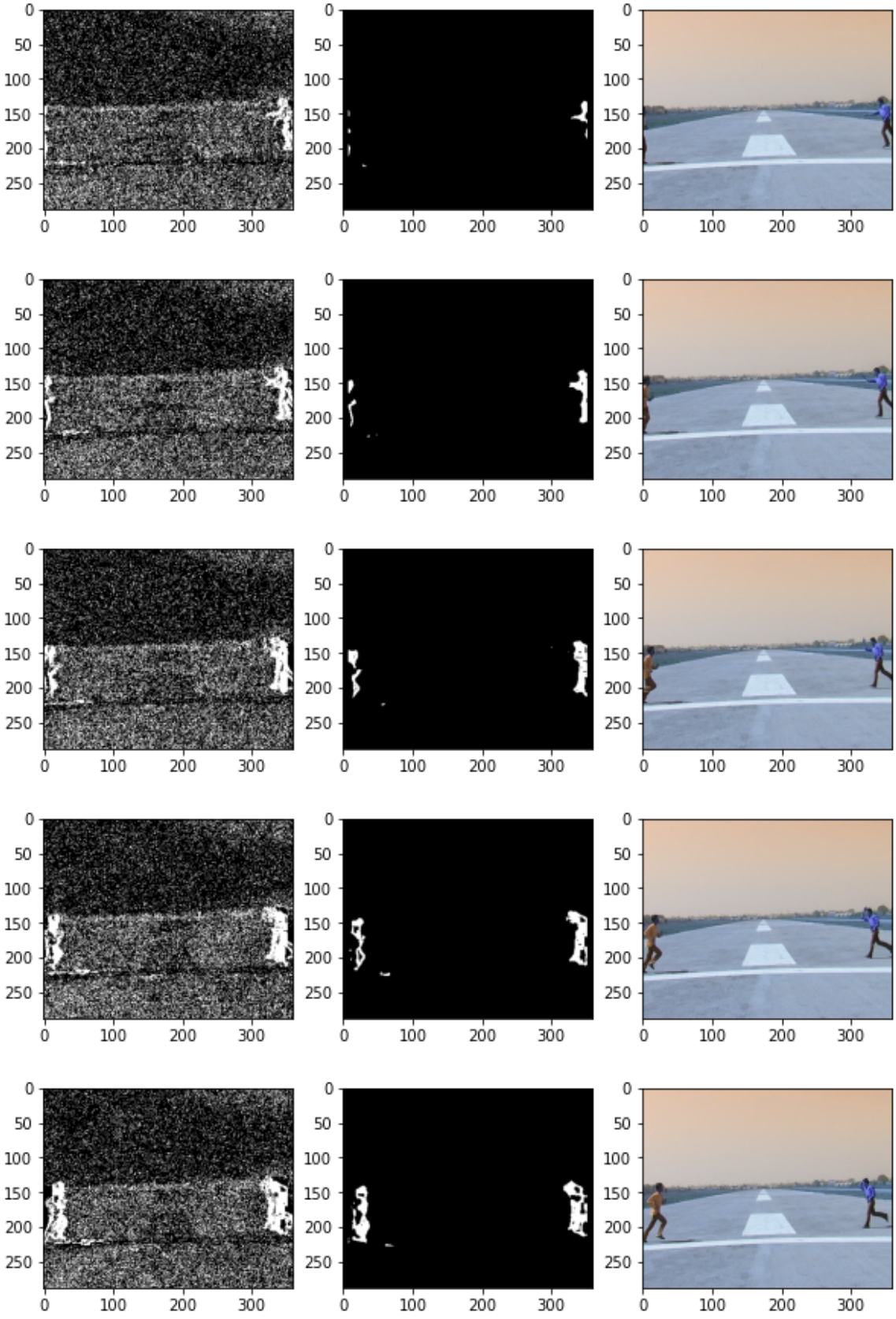


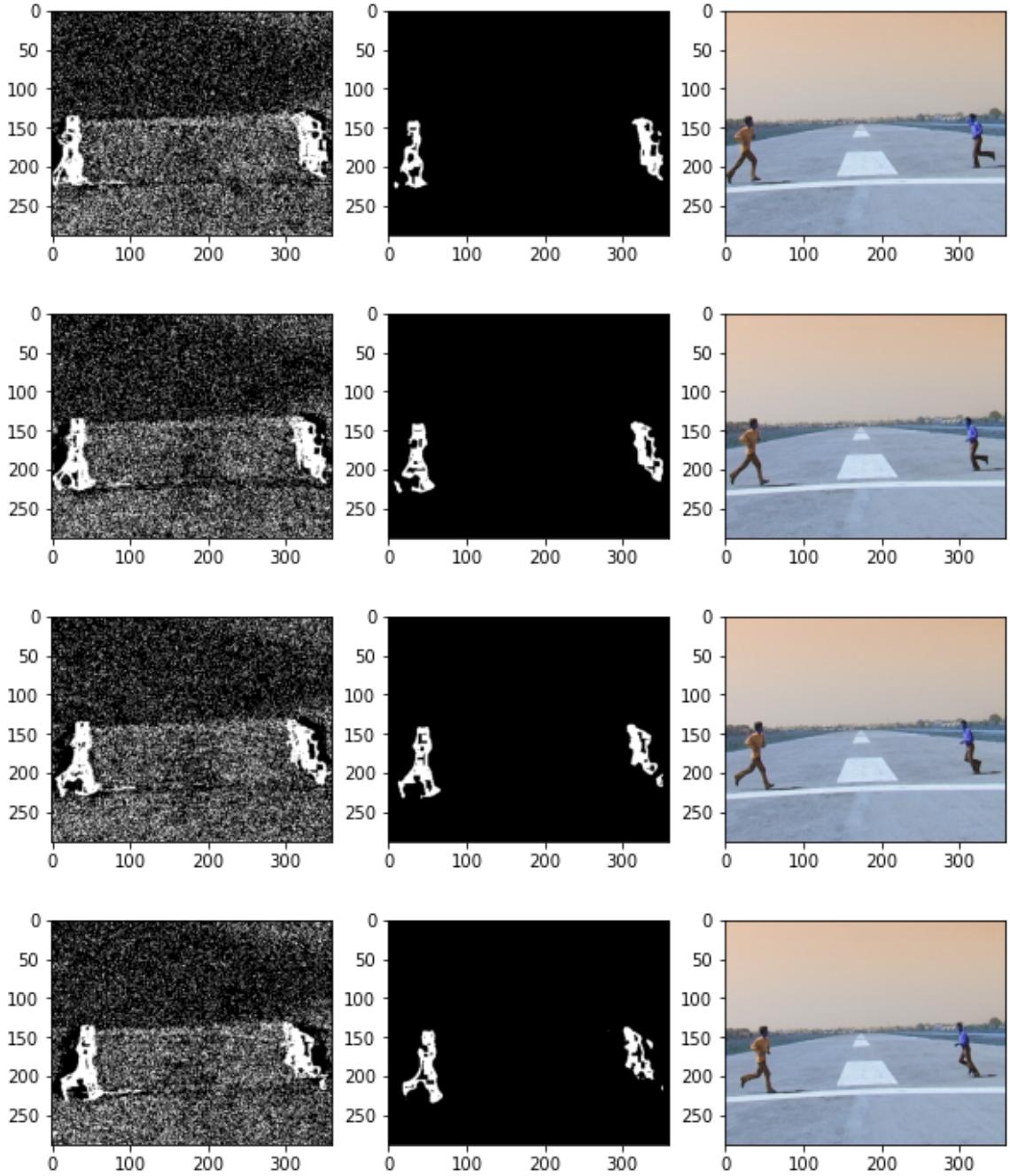




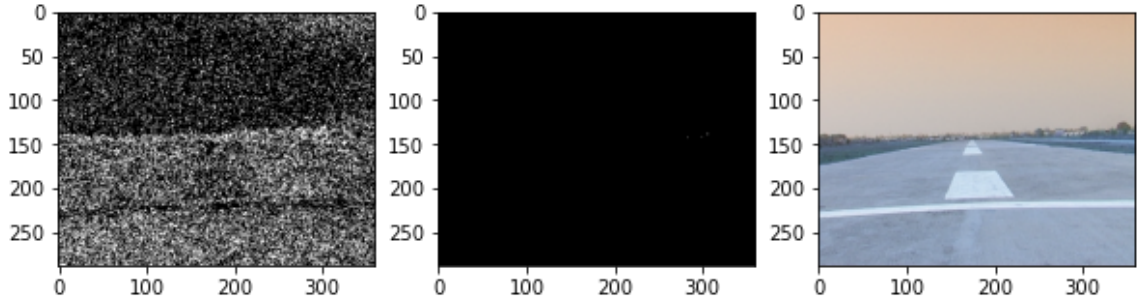


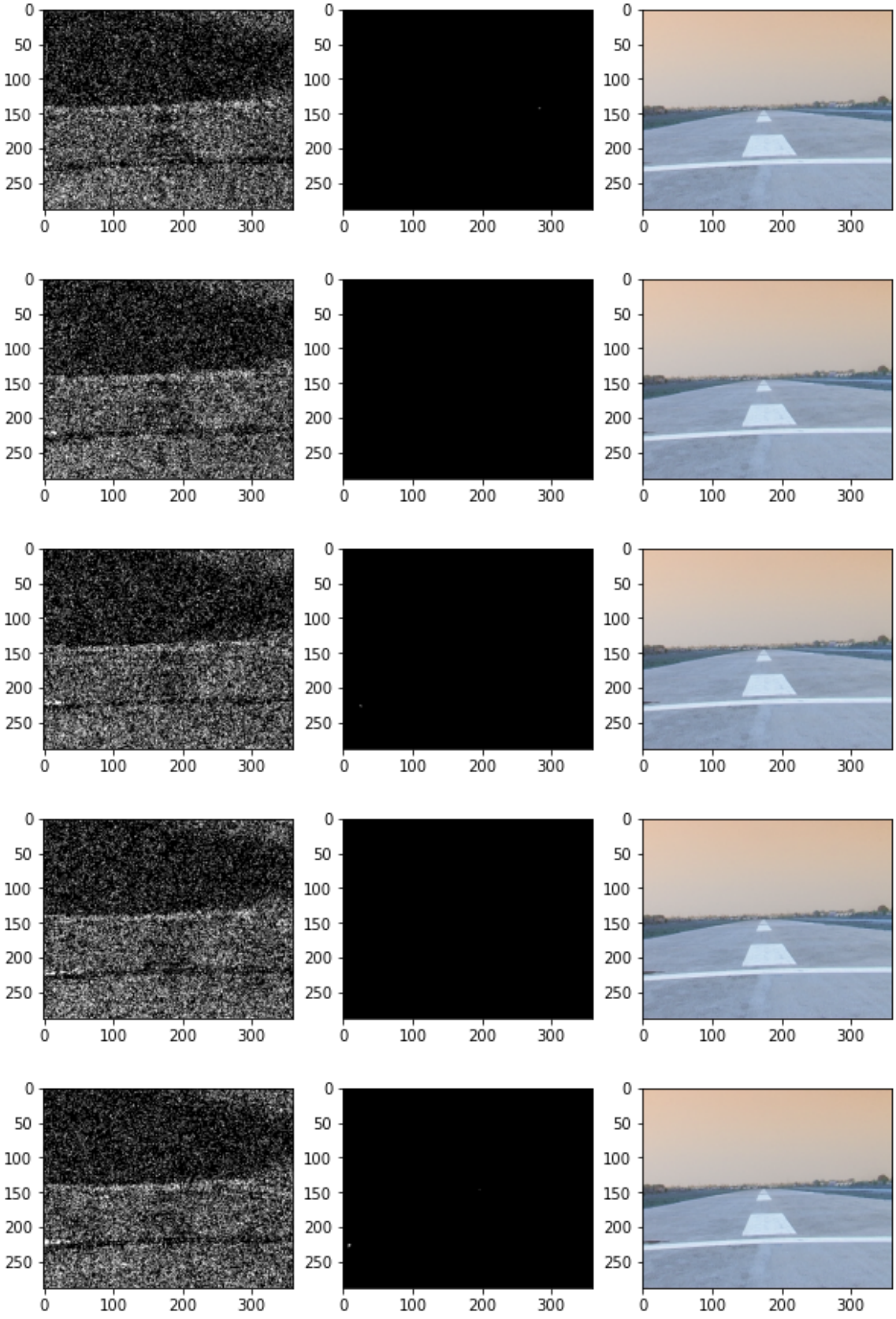

0 50 100 150 200 250
0 100 200 300

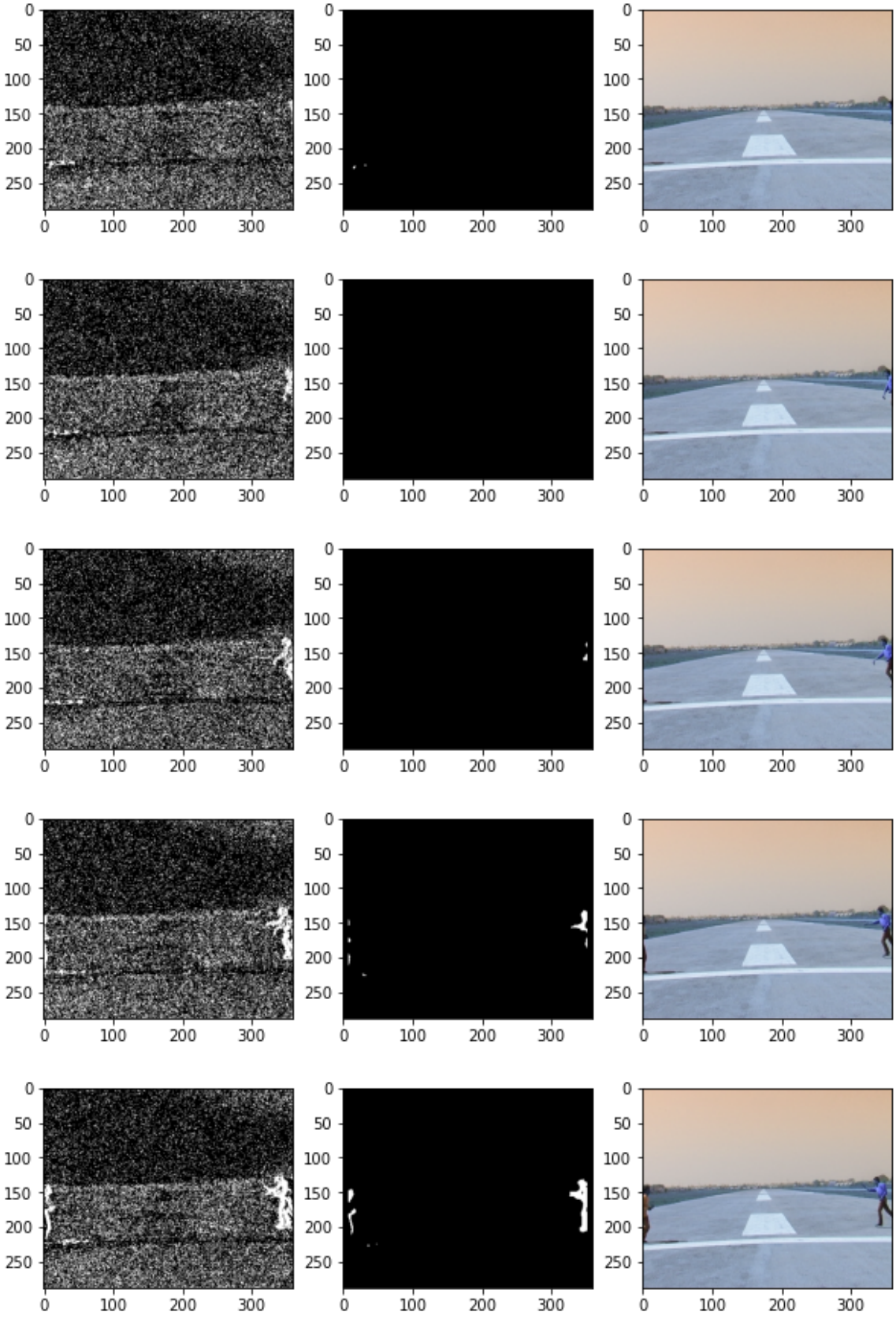


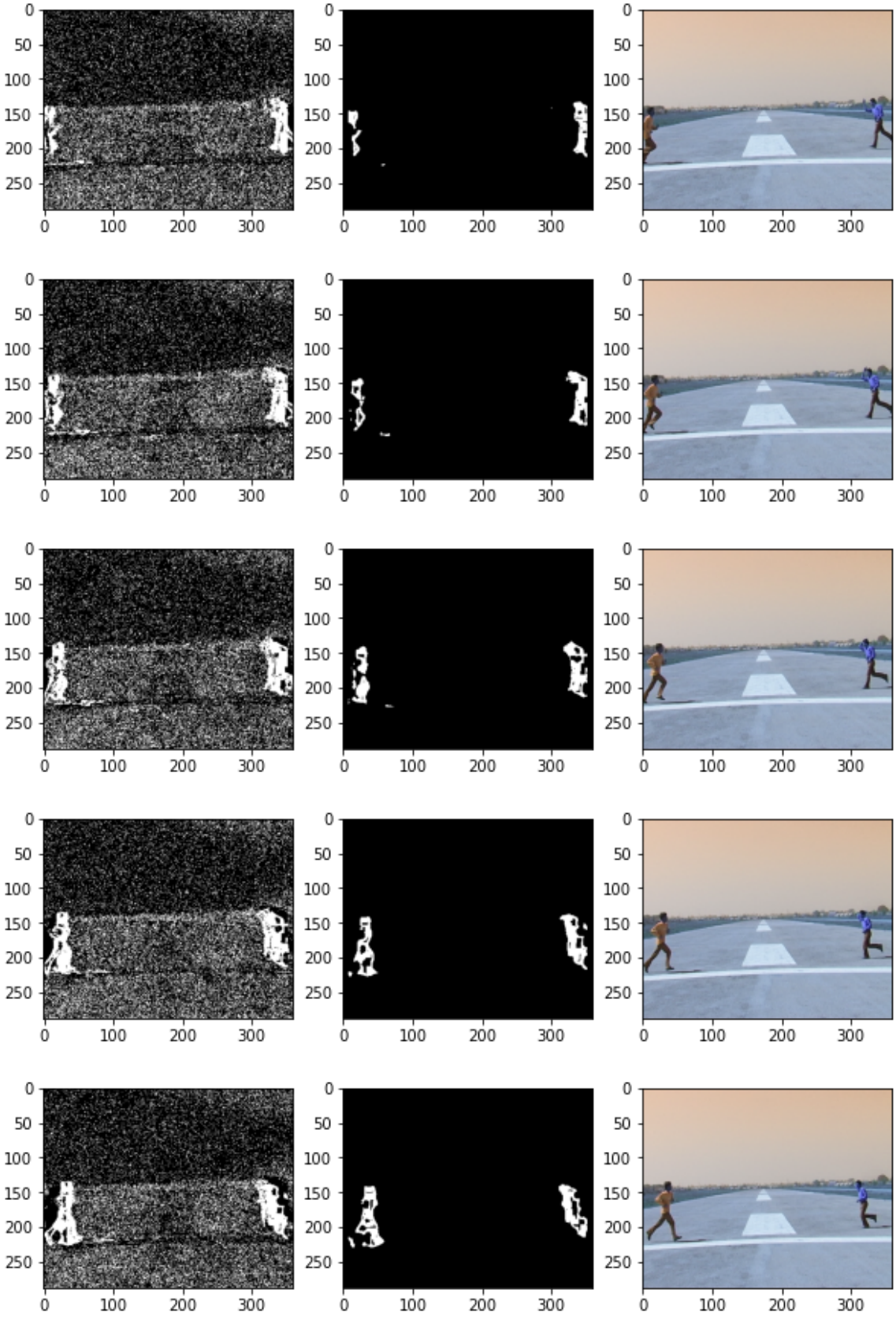


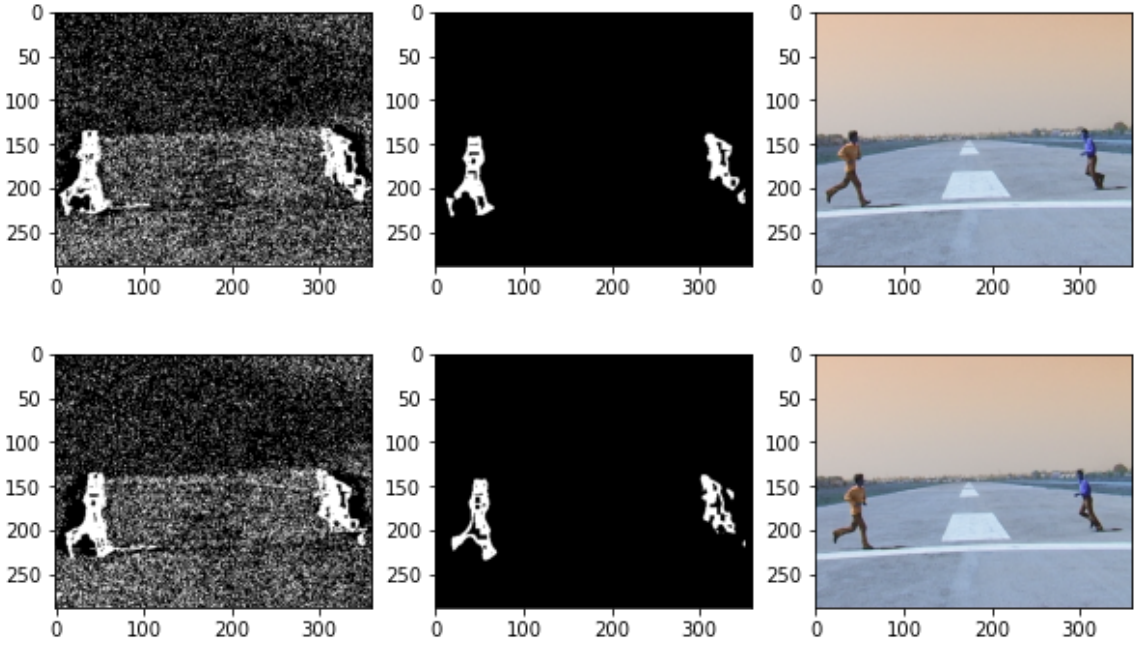
Using lambda as 0.8200000000000001 and eta as 0.7



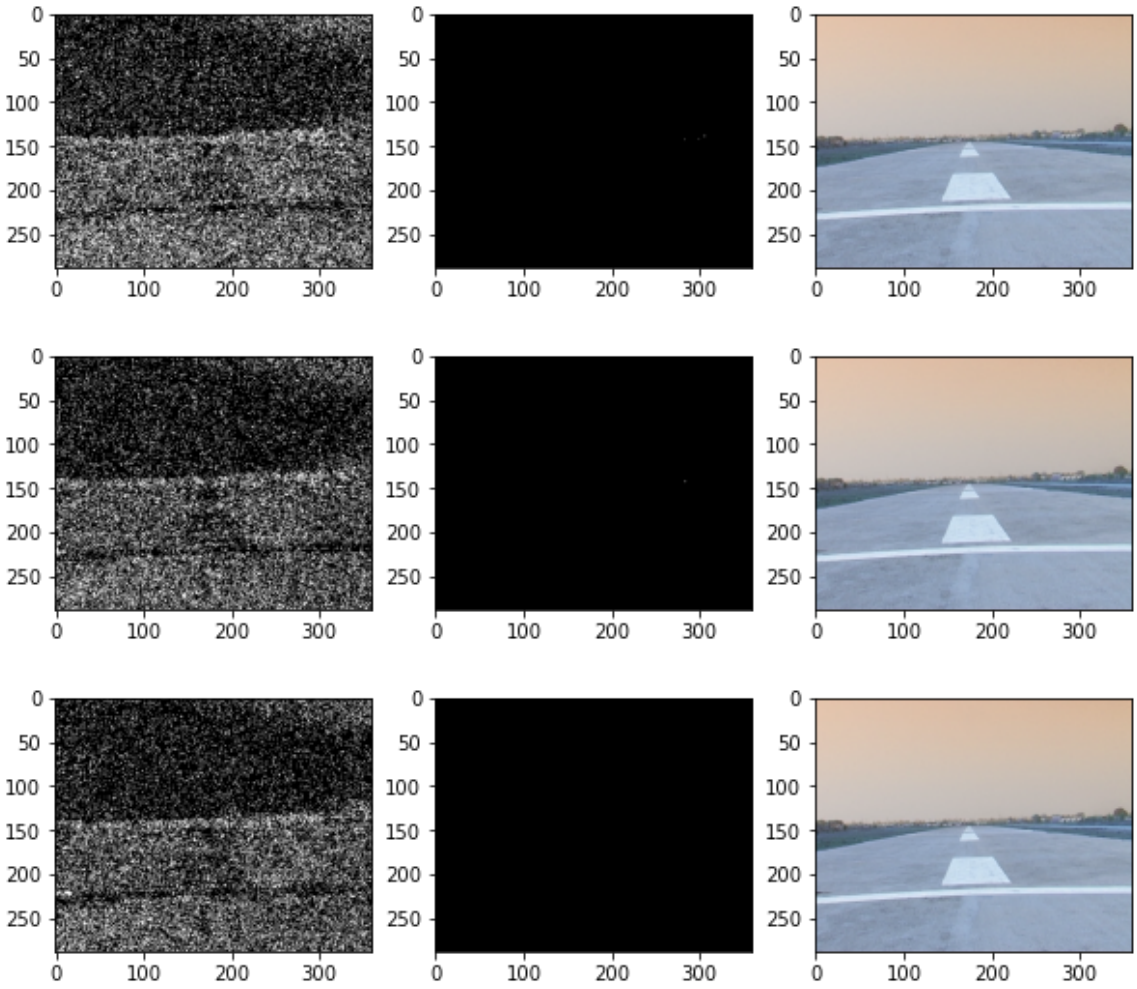

0
50
100
150
200
250
0
100
200
300

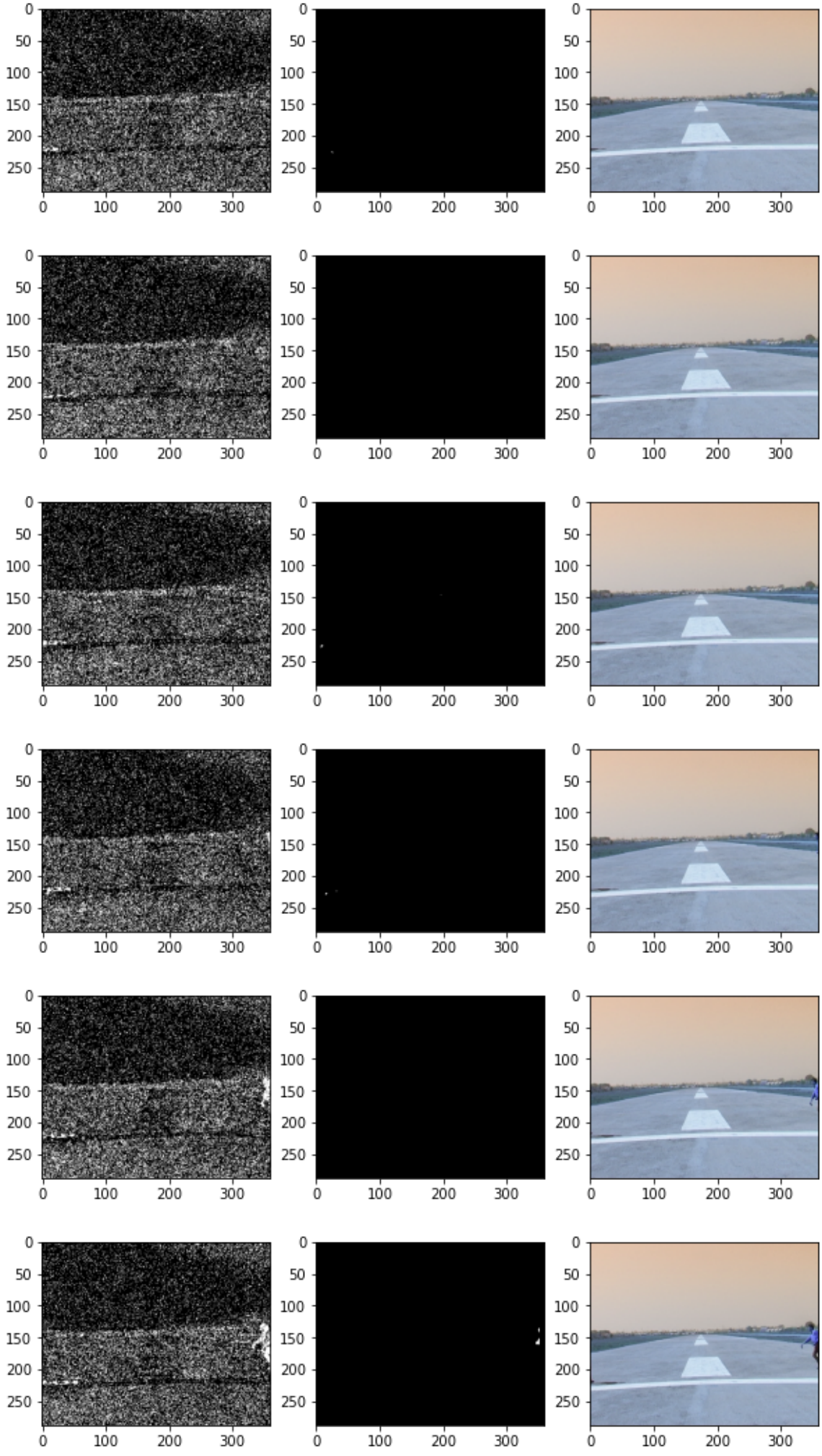


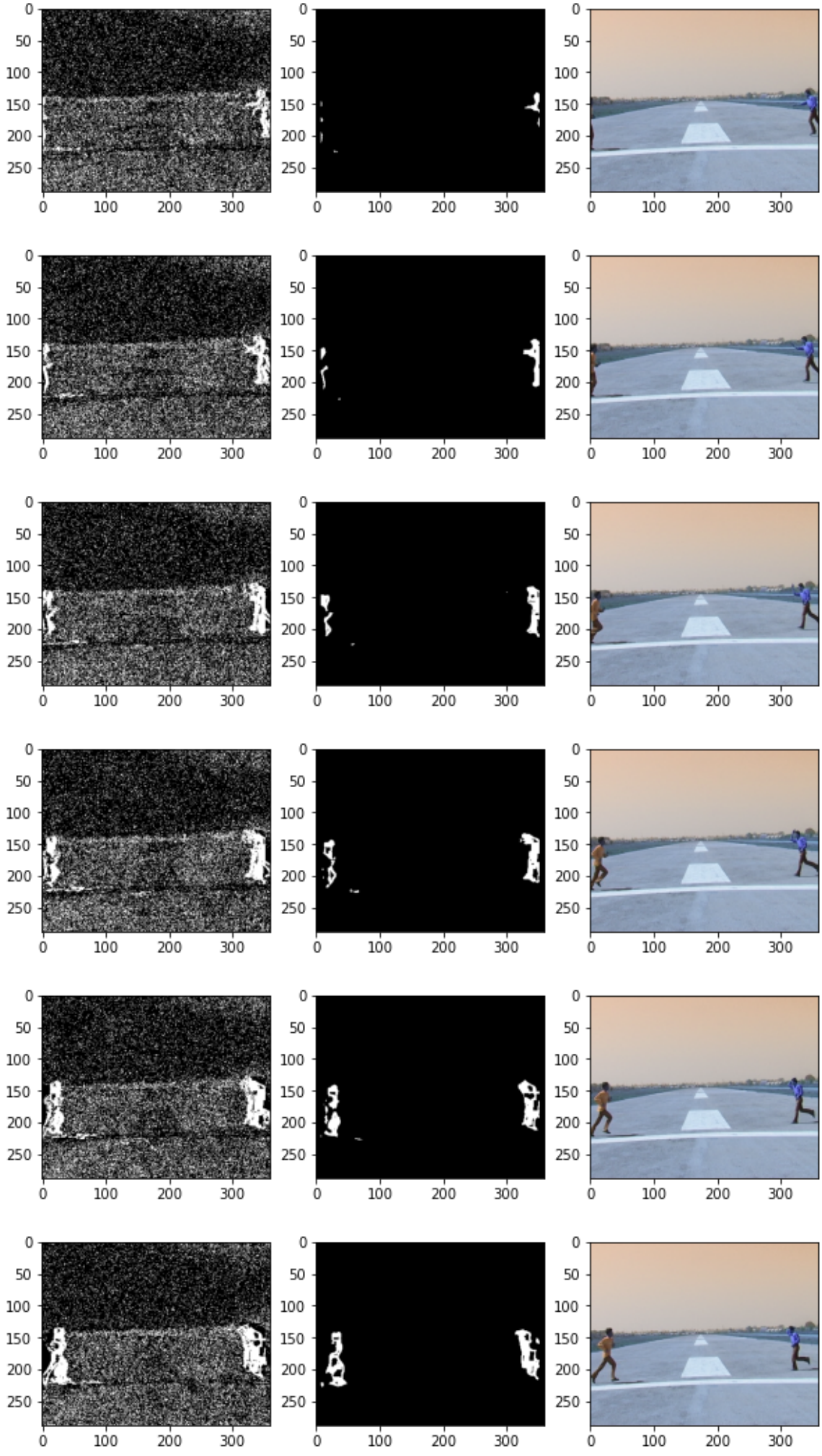


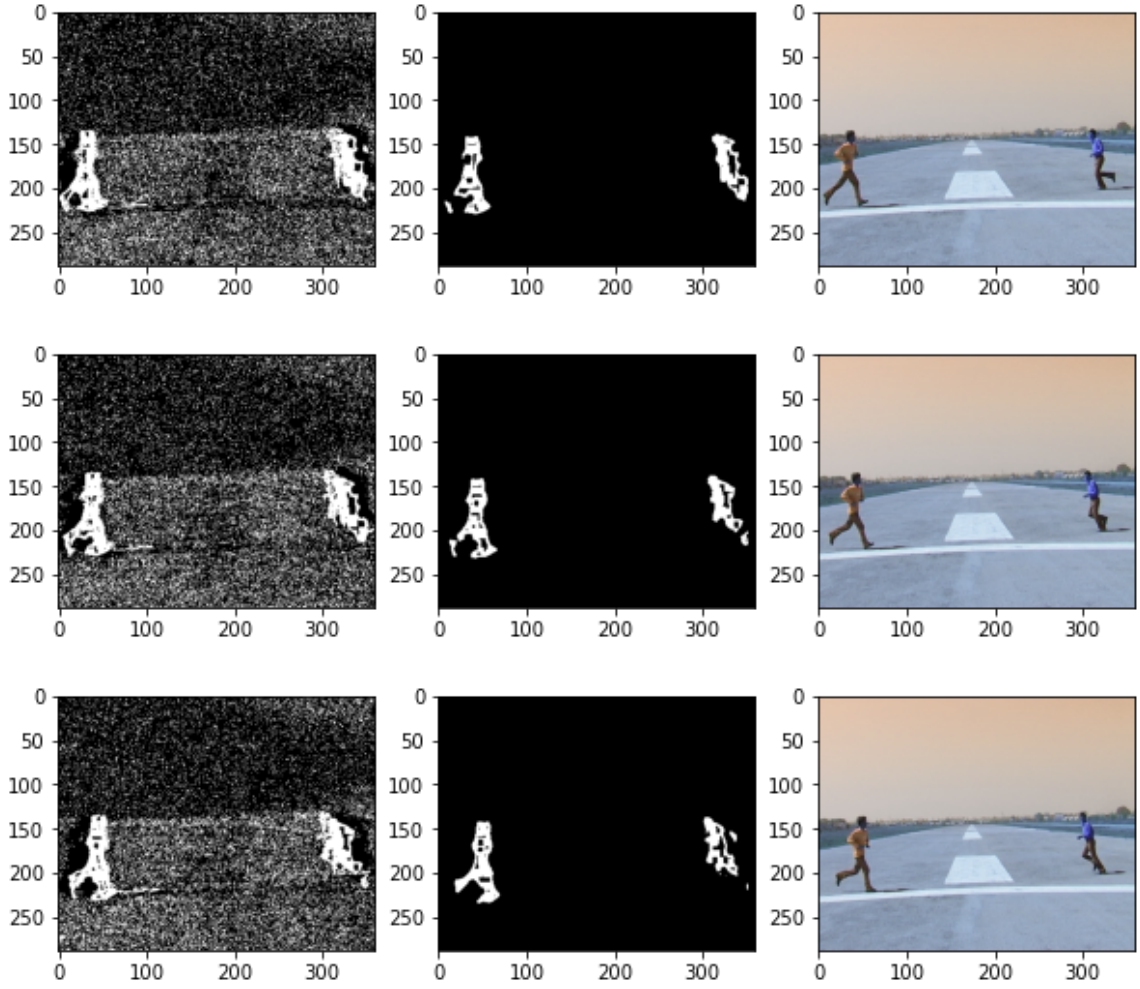


Using lambda as 0.8400000000000001 and eta as 0.7

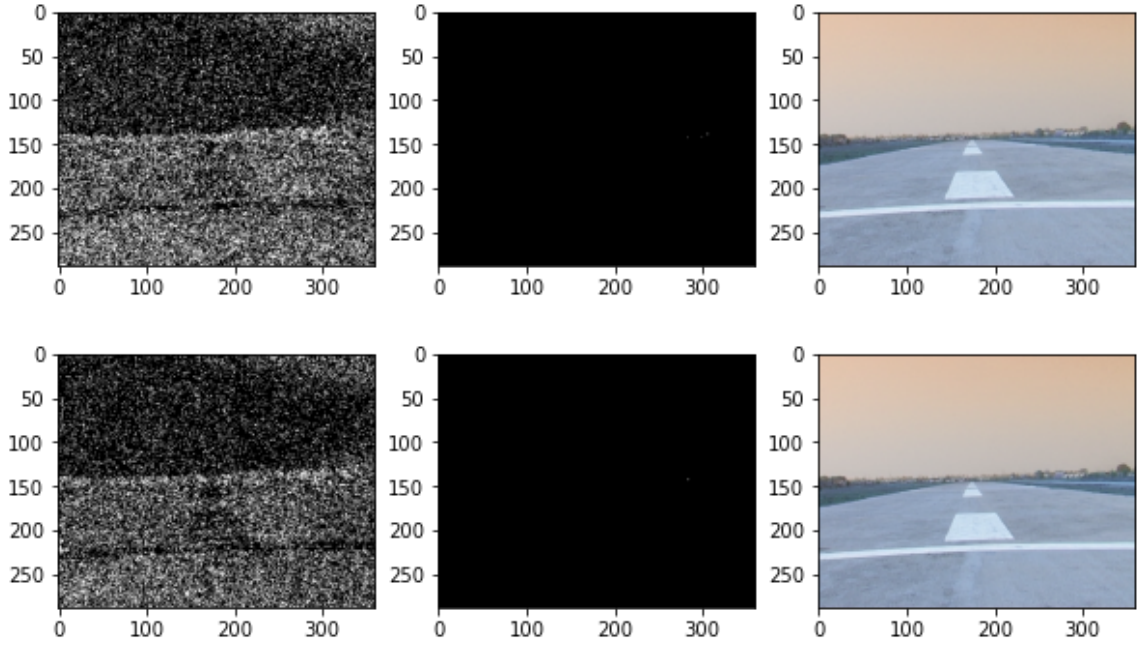


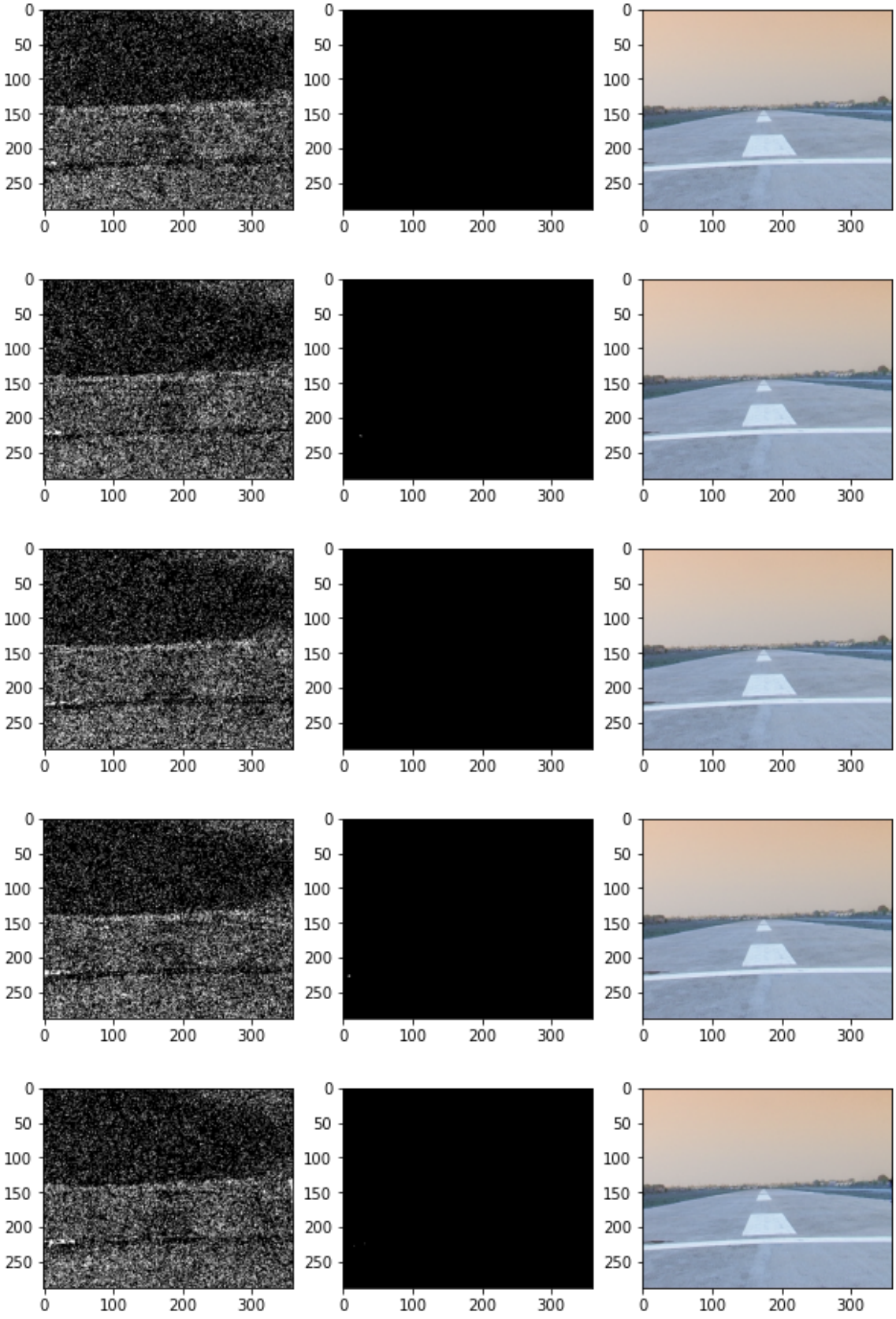


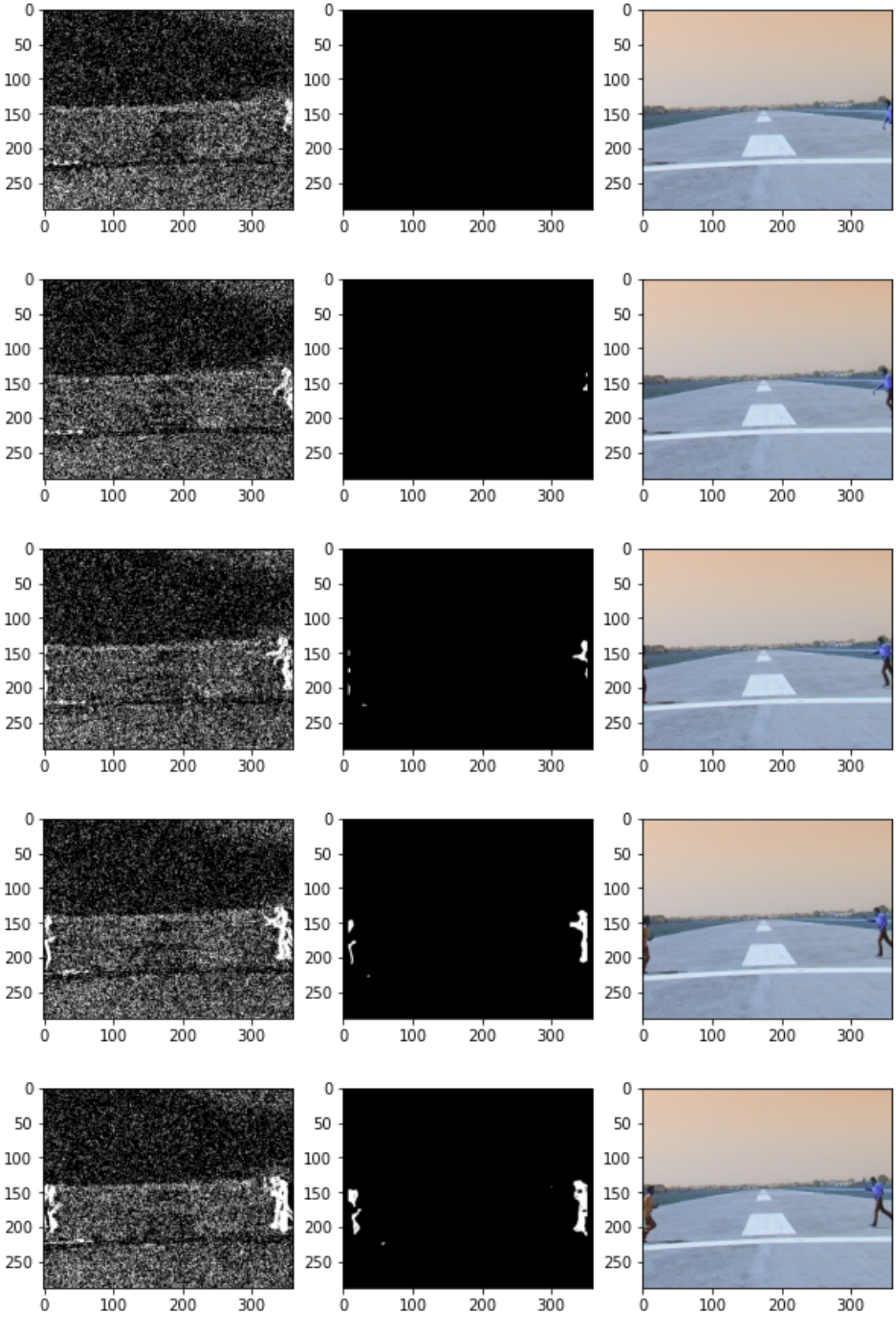


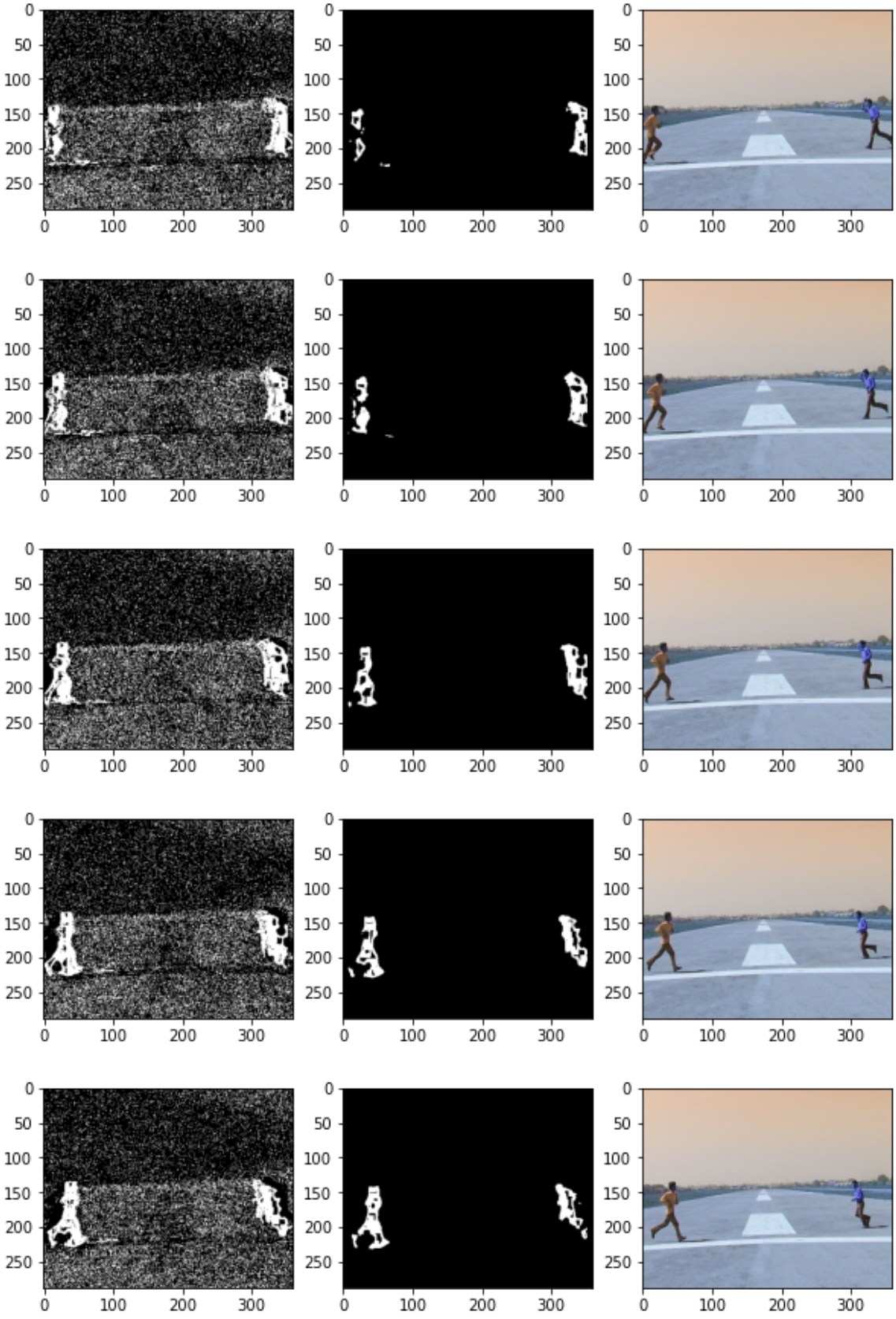


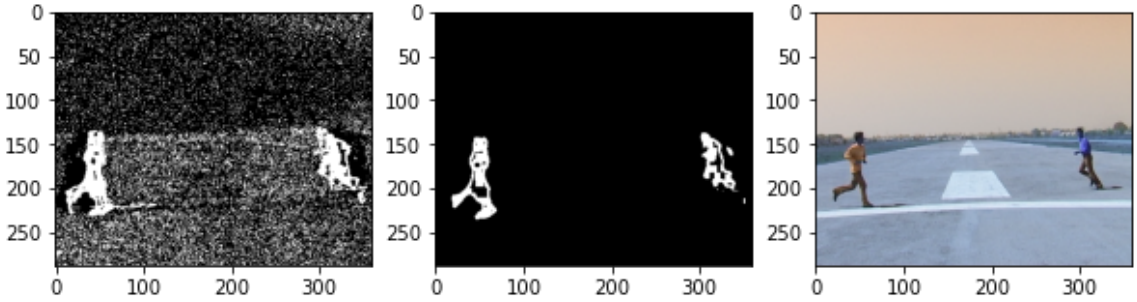
Using lambda as 0.8600000000000001 and eta as 0.7



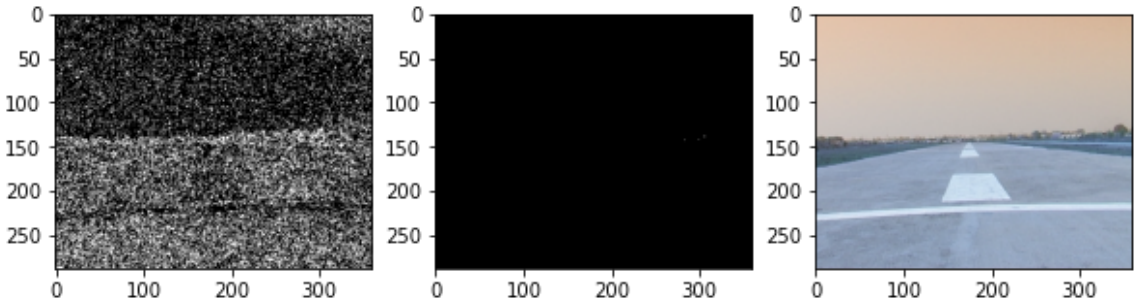

0
50
100
150
200
250
0
100
200
300

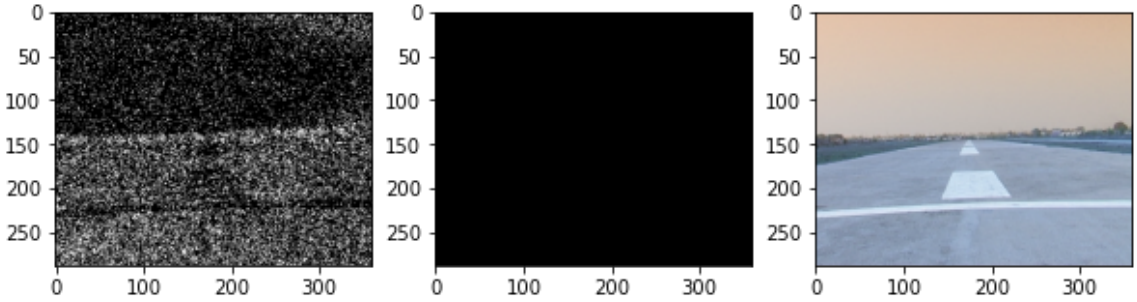


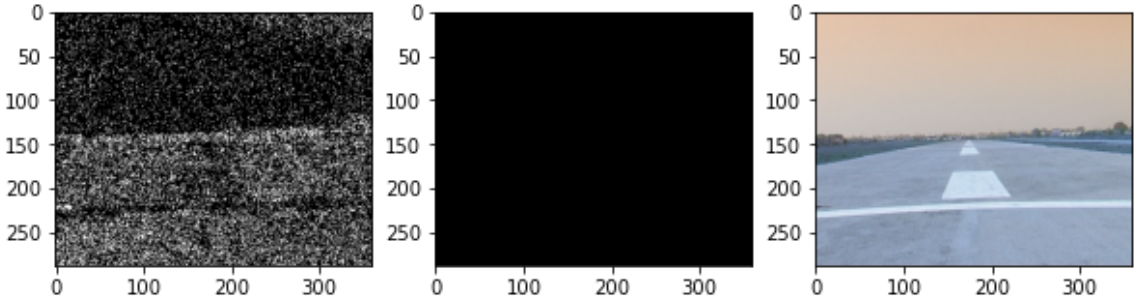


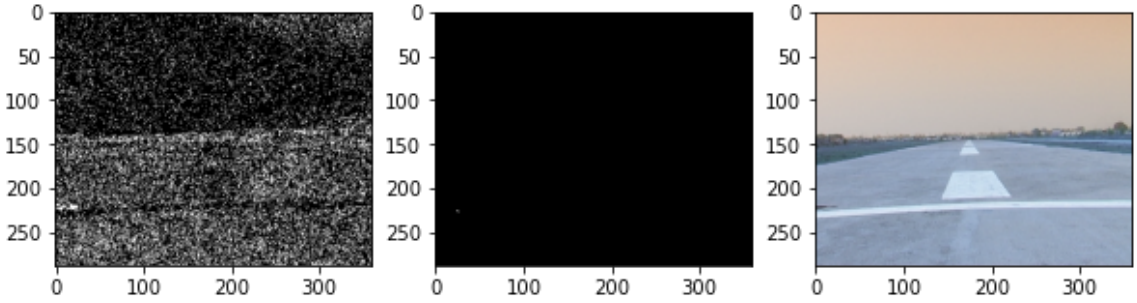


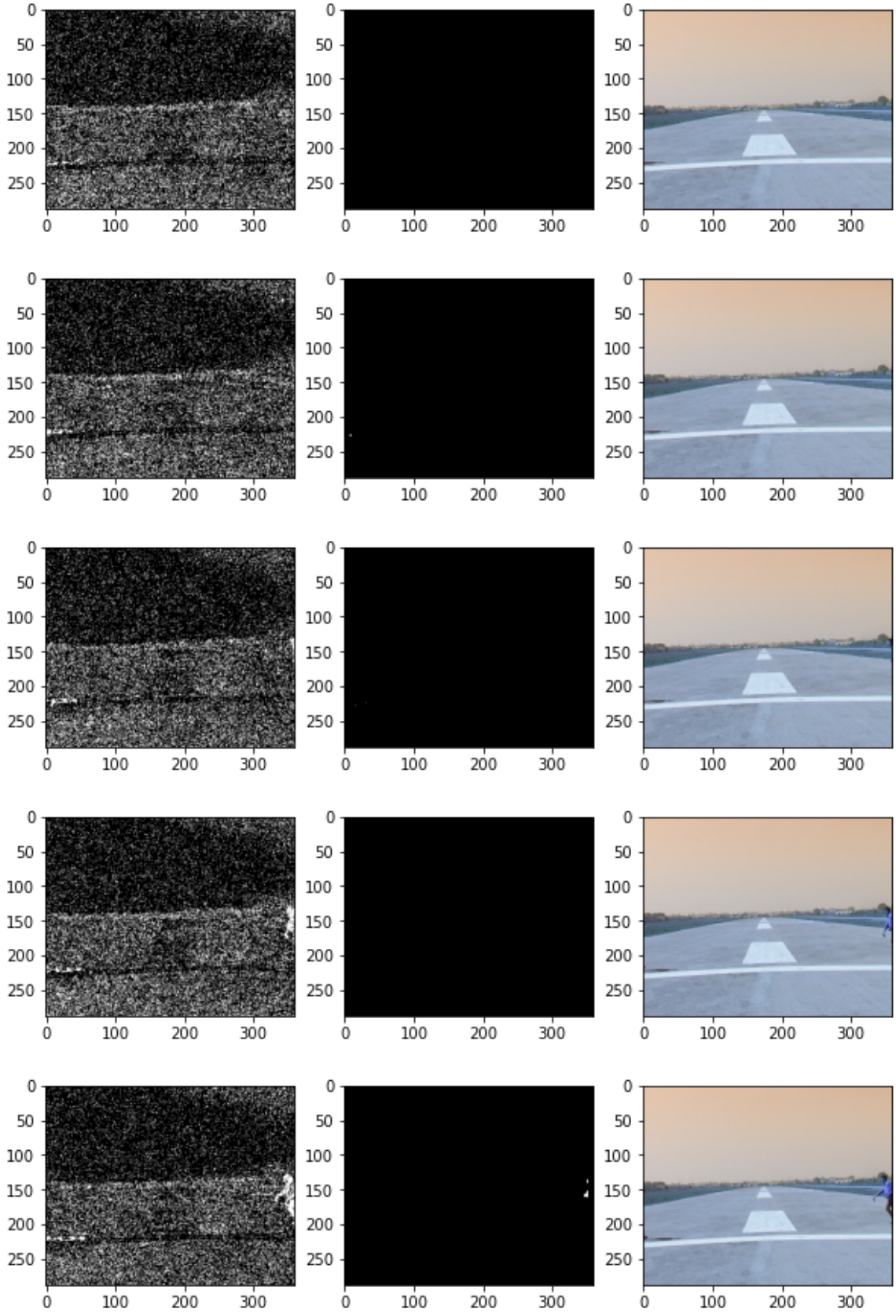
Using lambda as 0.8800000000000001 and eta as 0.7

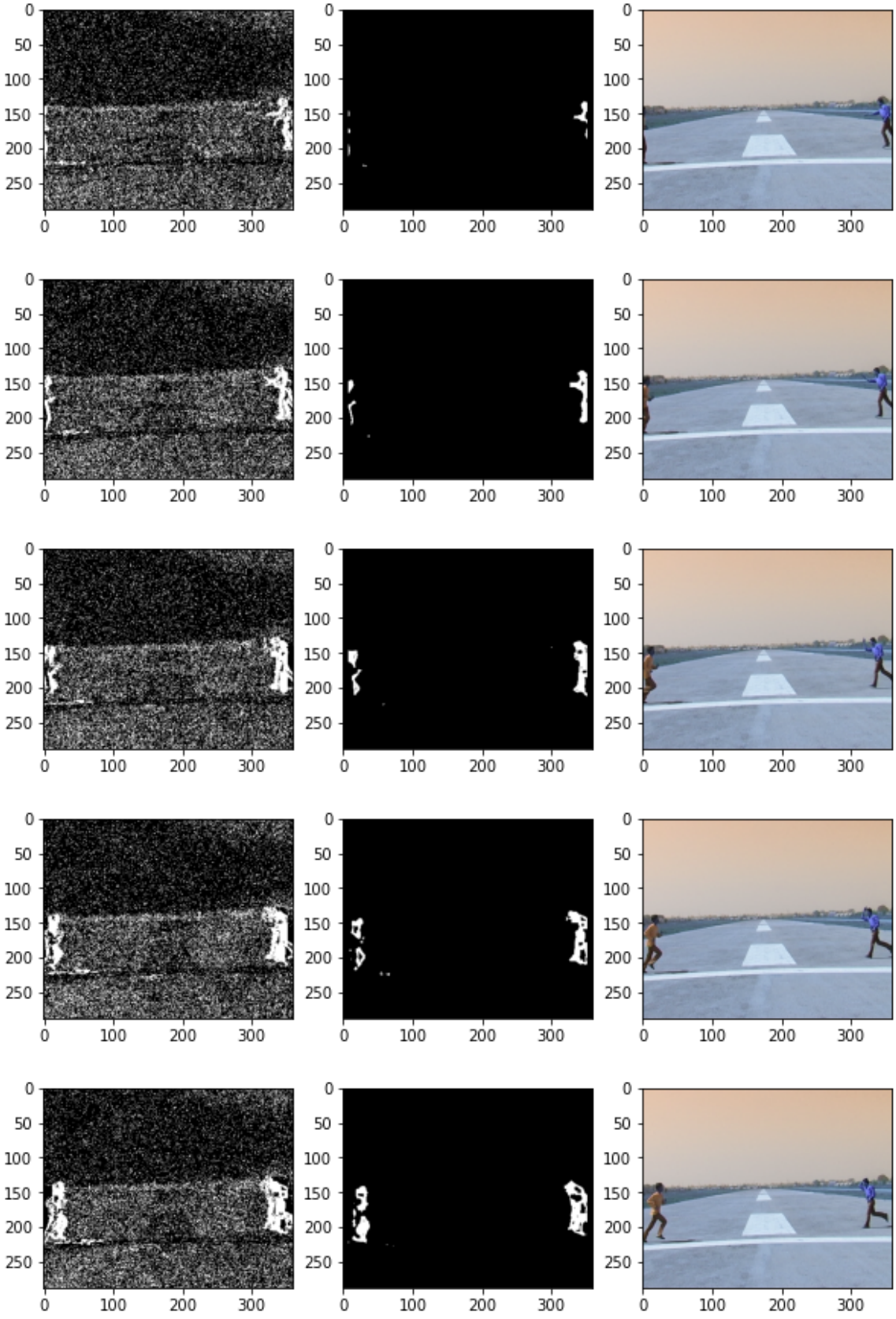


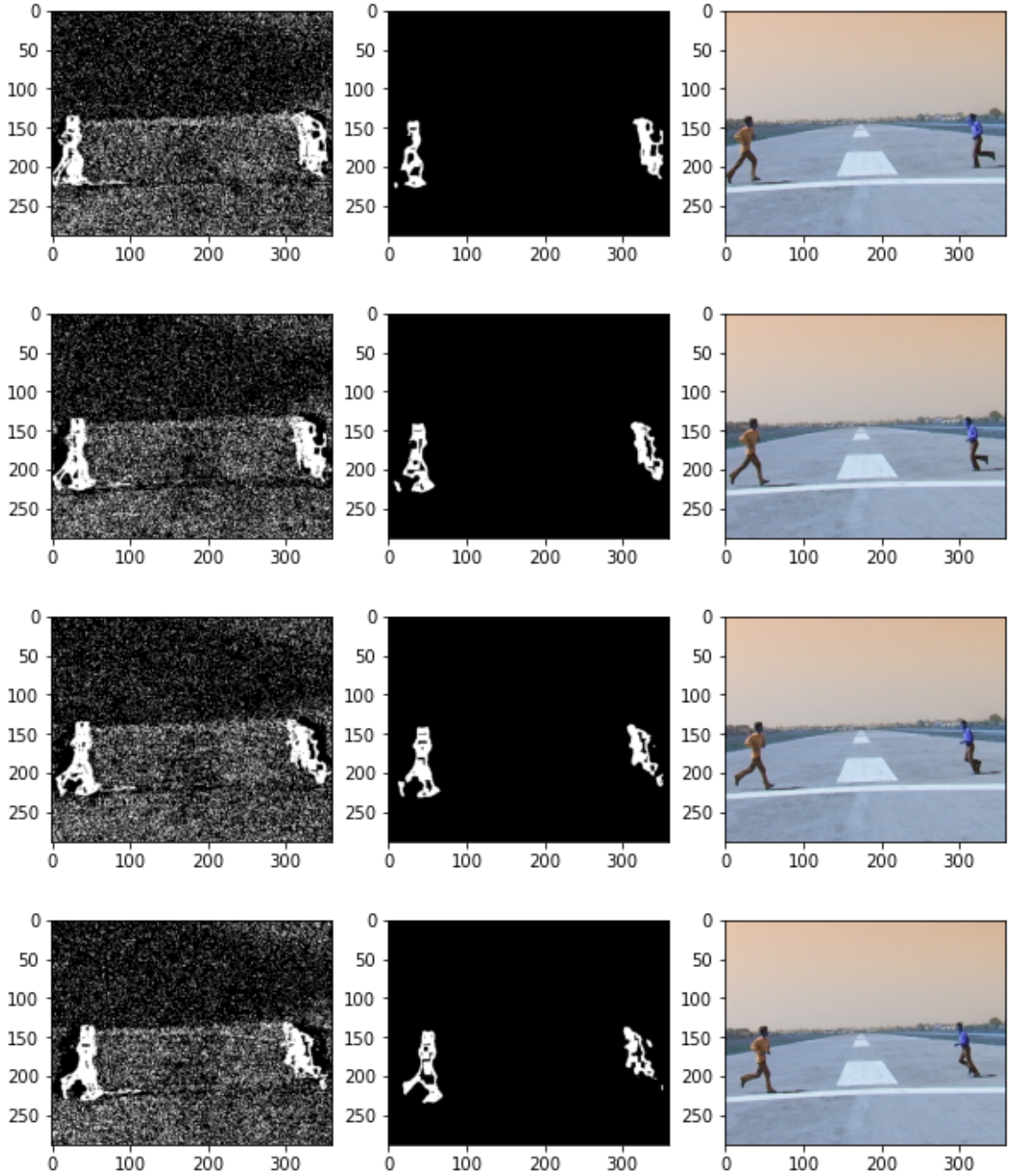




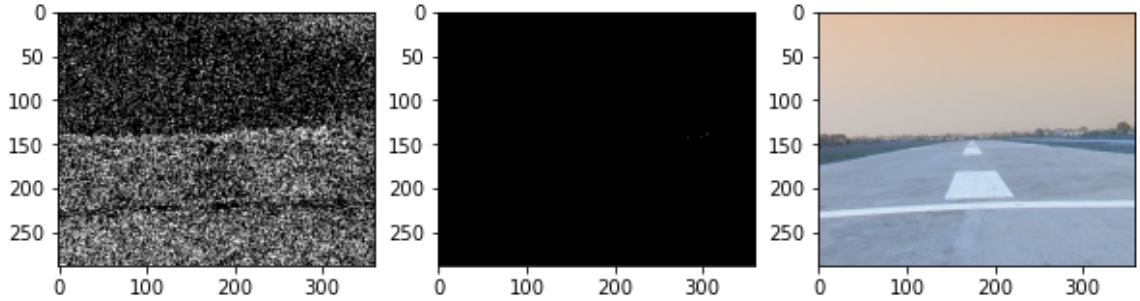


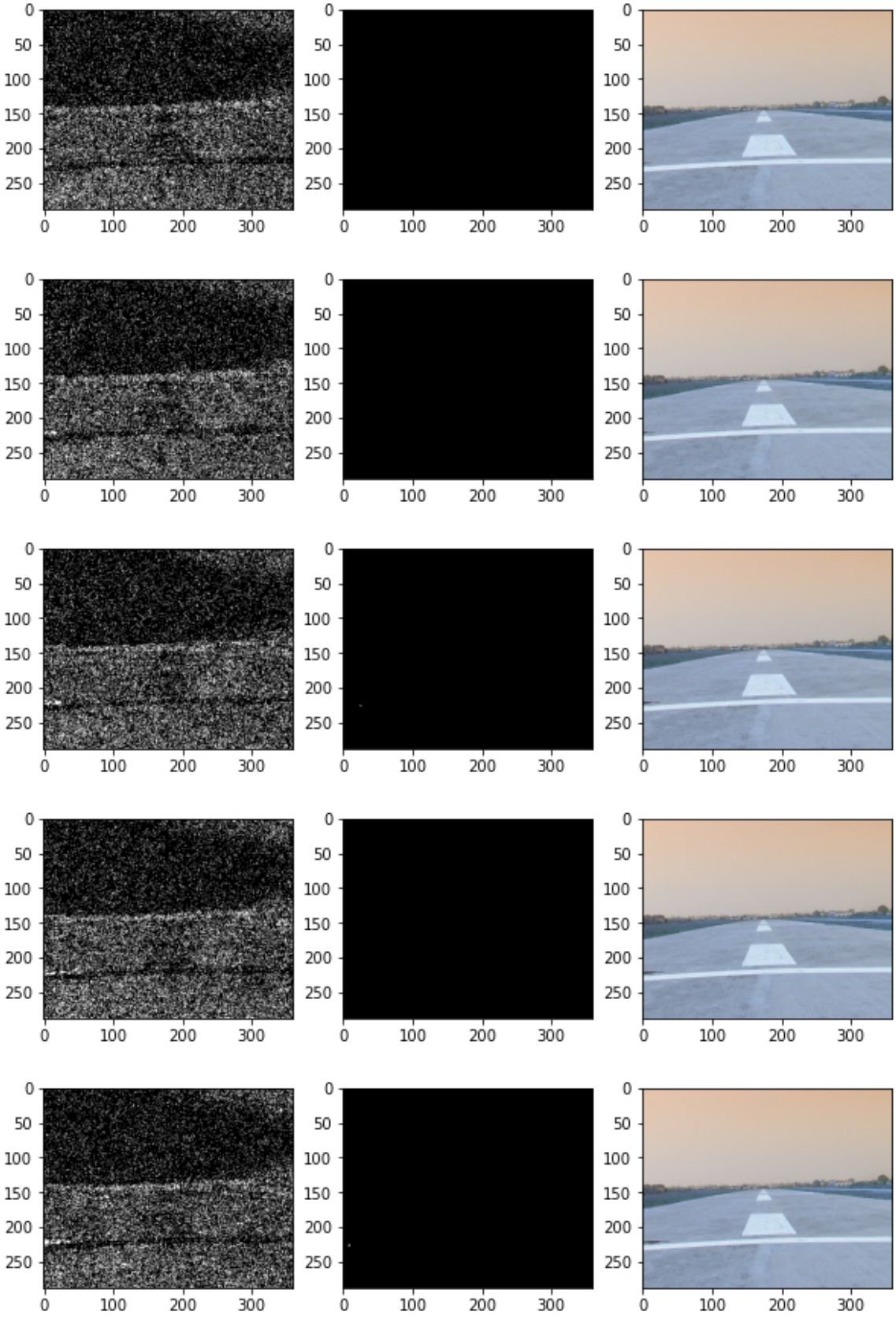


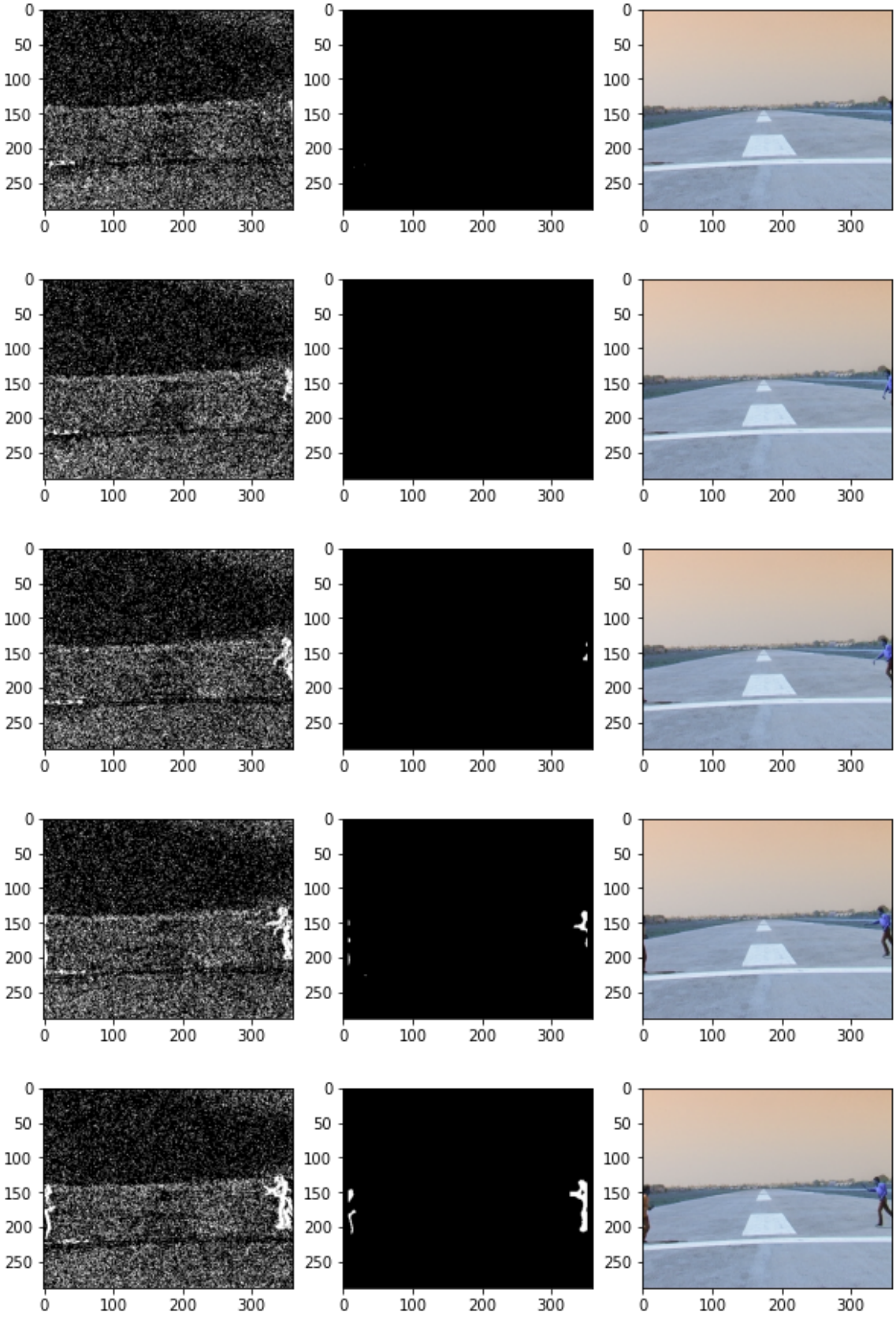


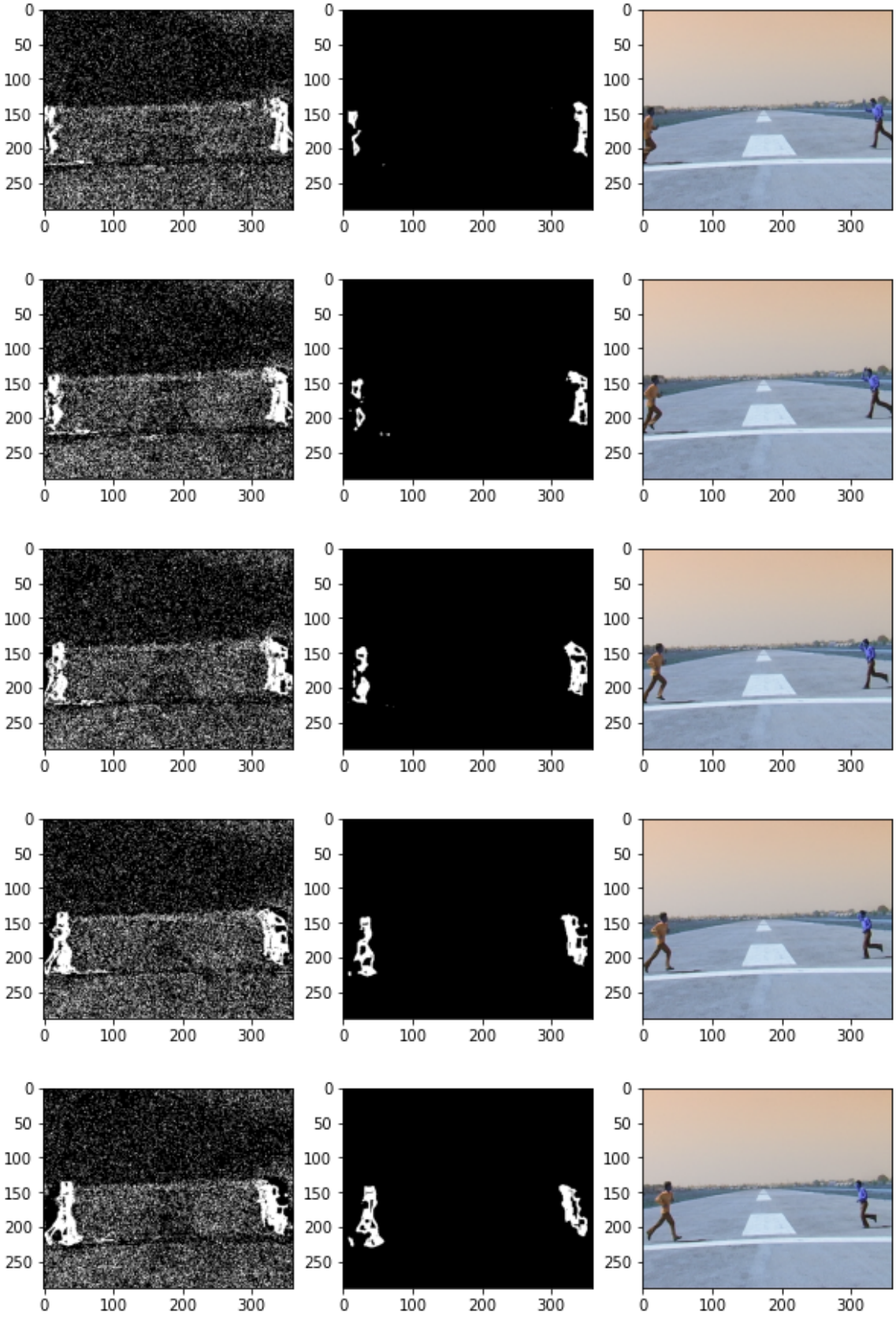


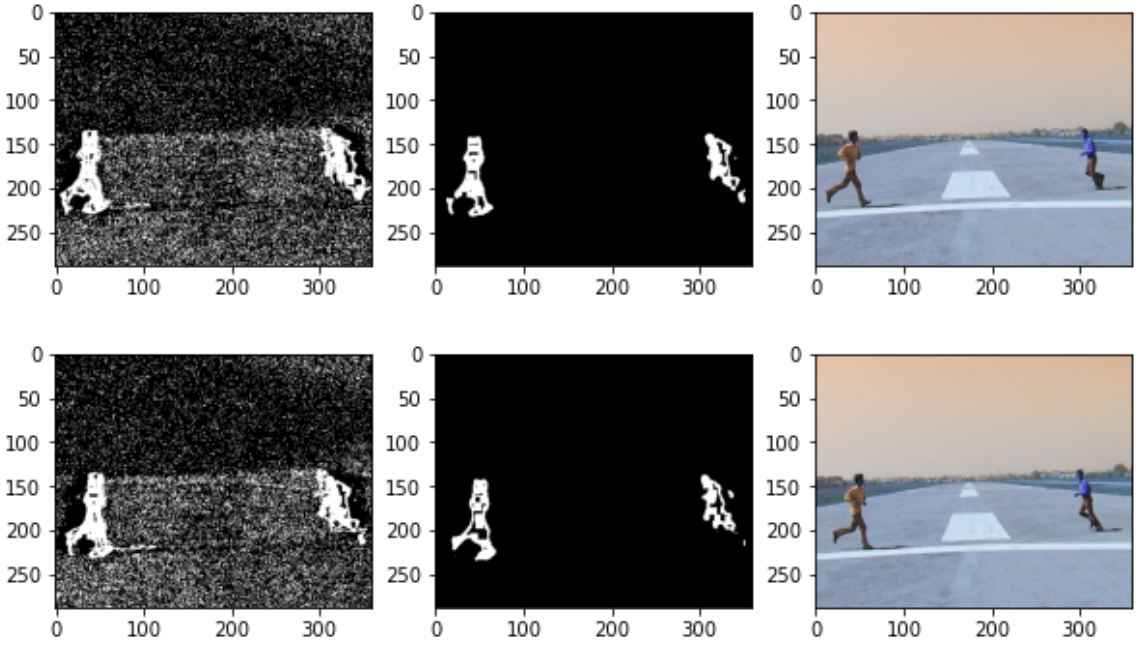
Using lambda as 0.9000000000000001 and eta as 0.7



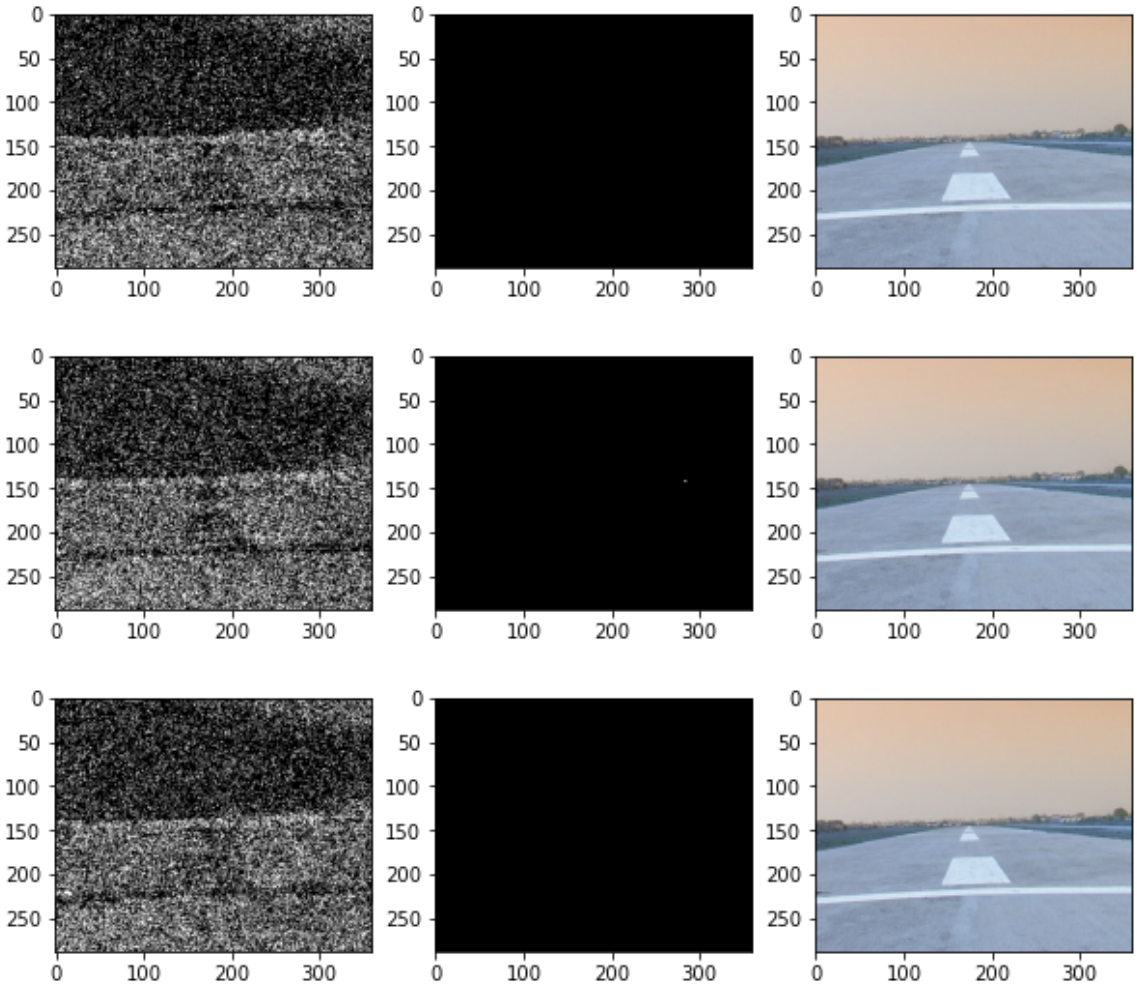

0
50
100
150
200
250
0
100
200
300

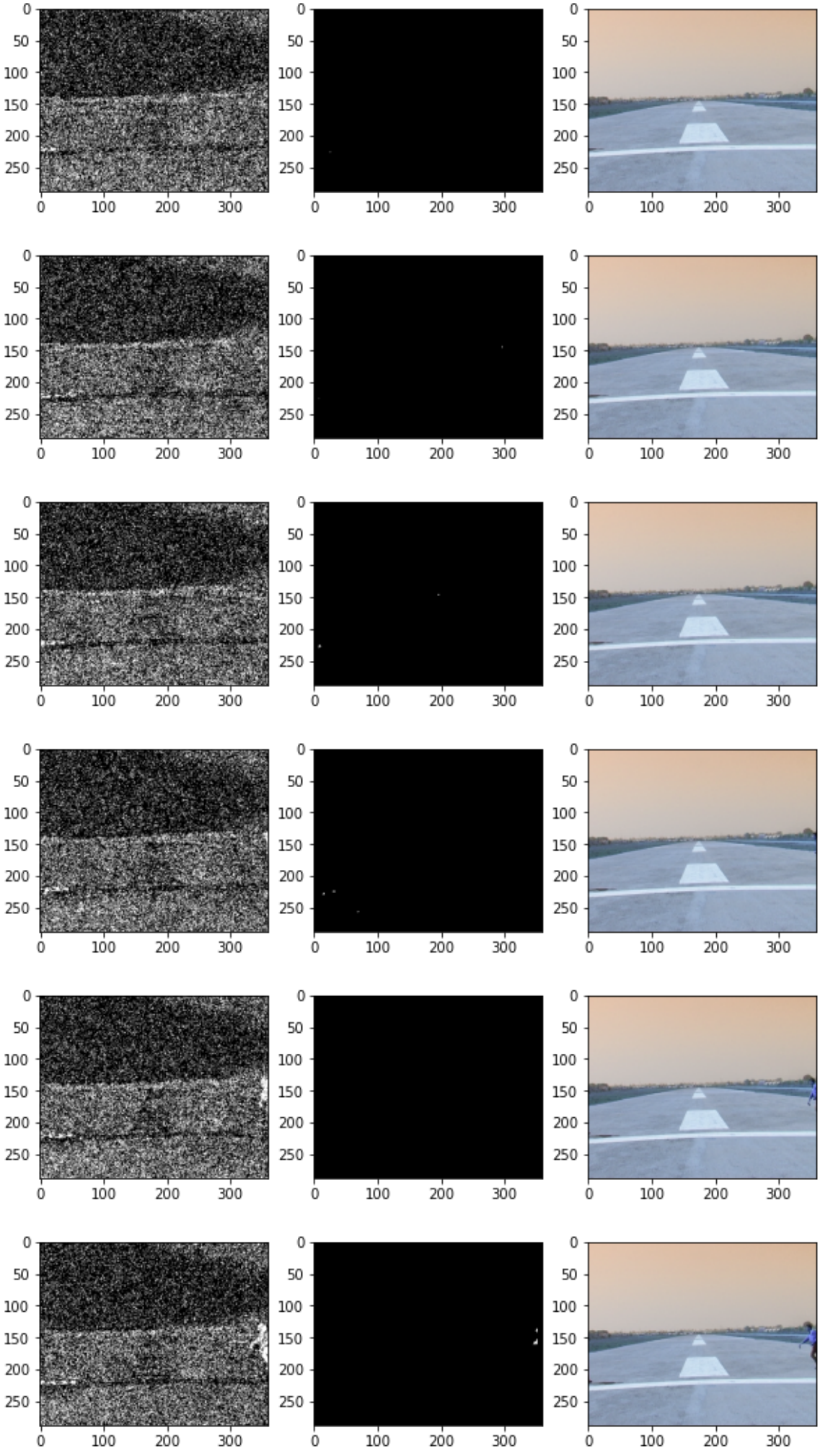

0 50 100 150 200 250
0 100 200 300

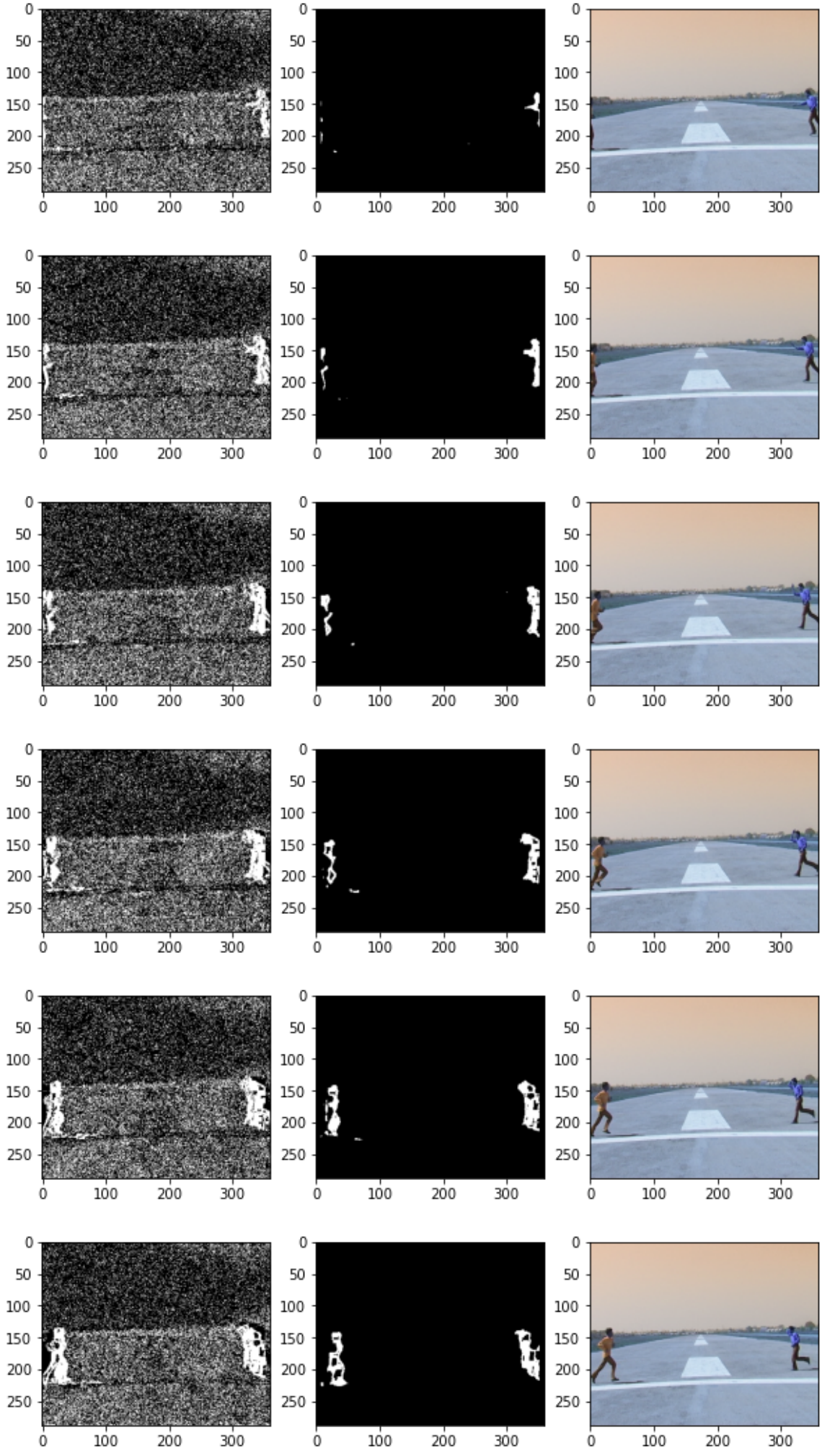


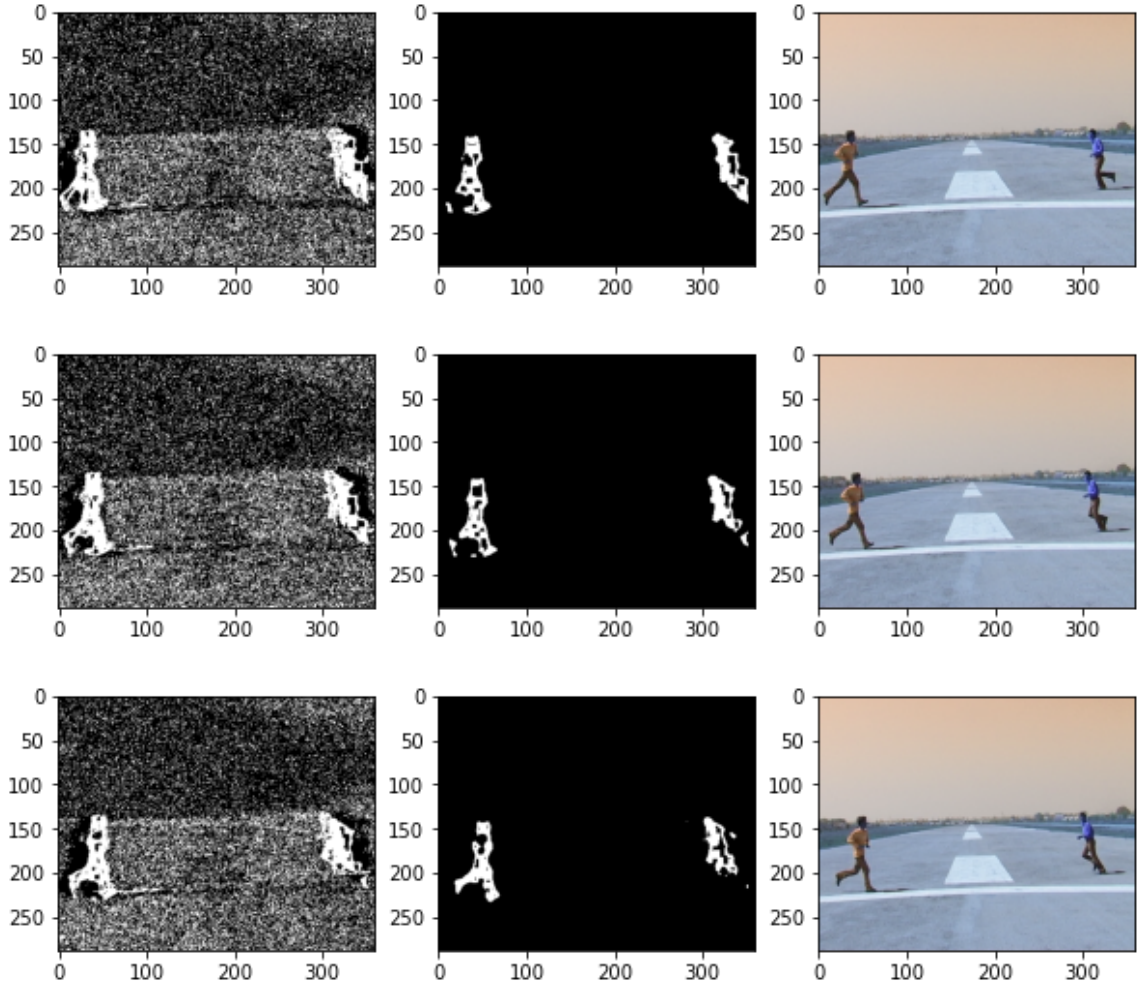


Using lambda as 0.7 and eta as 0.72

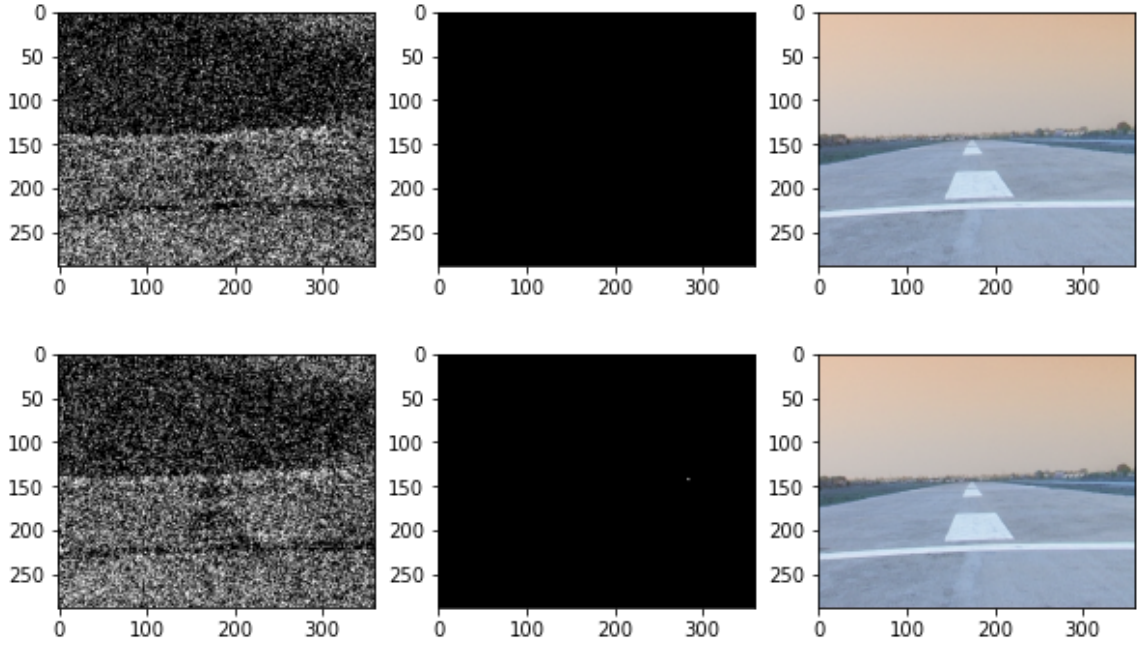


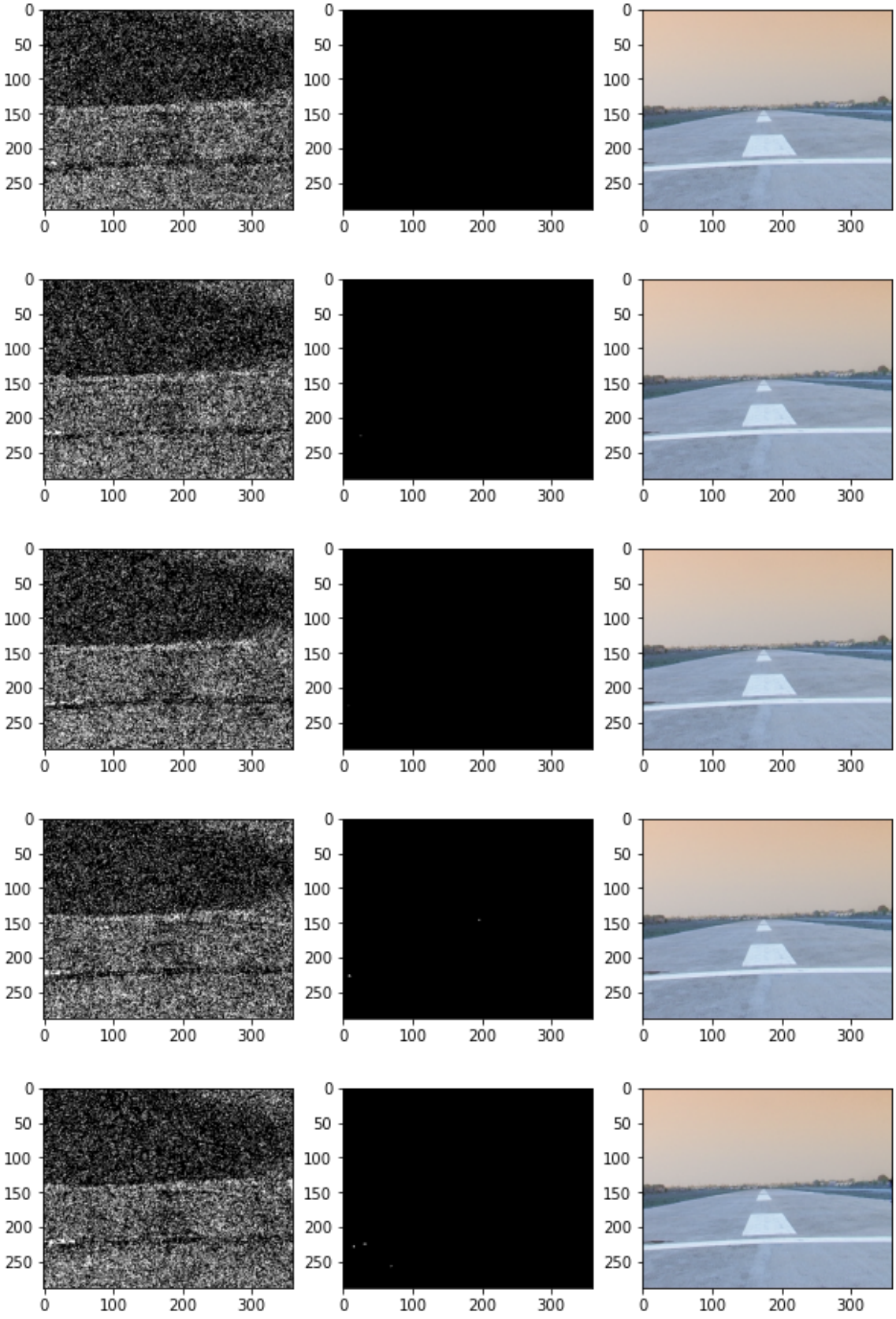


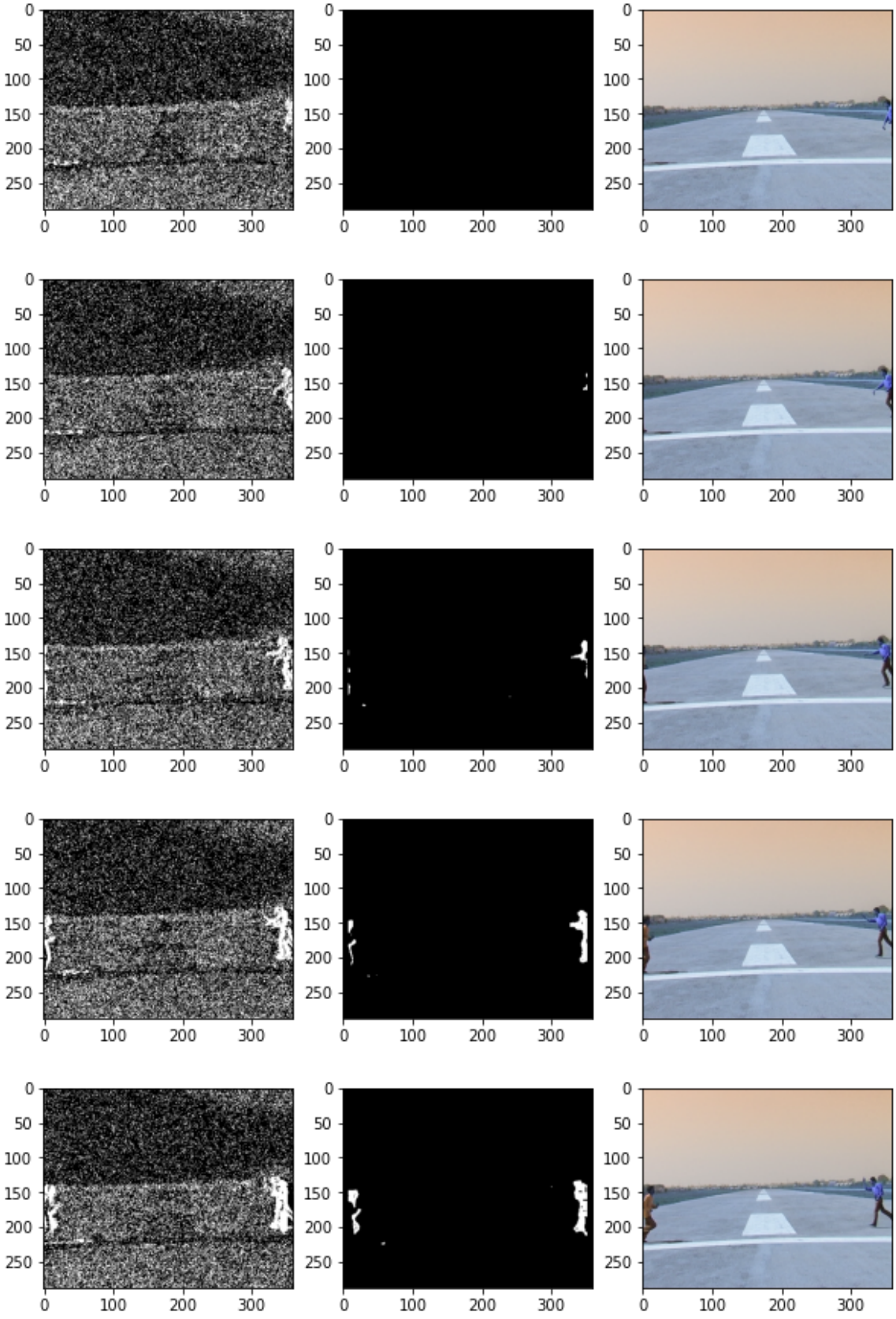

0 50 100 150 200 250
0 100 200 300

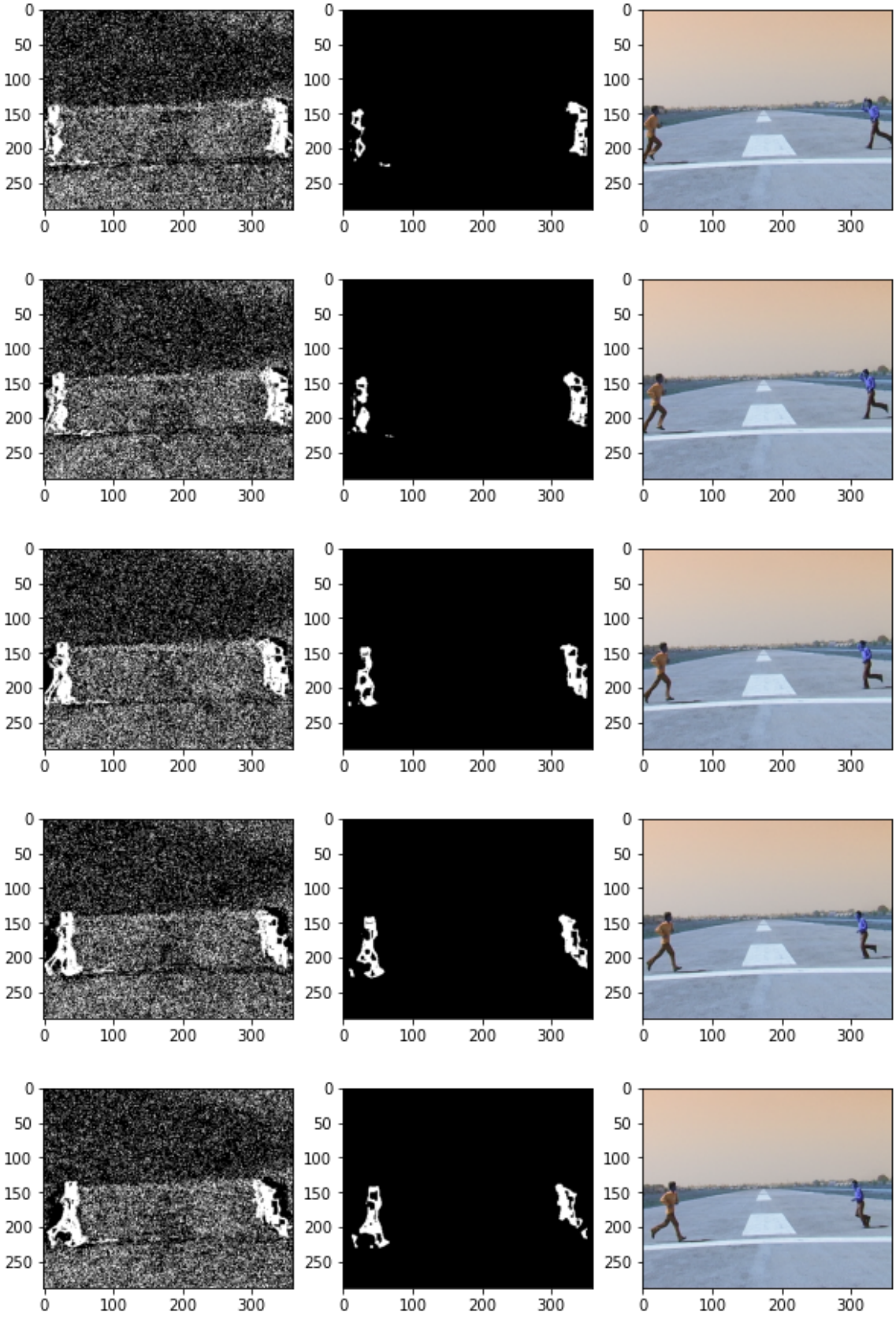


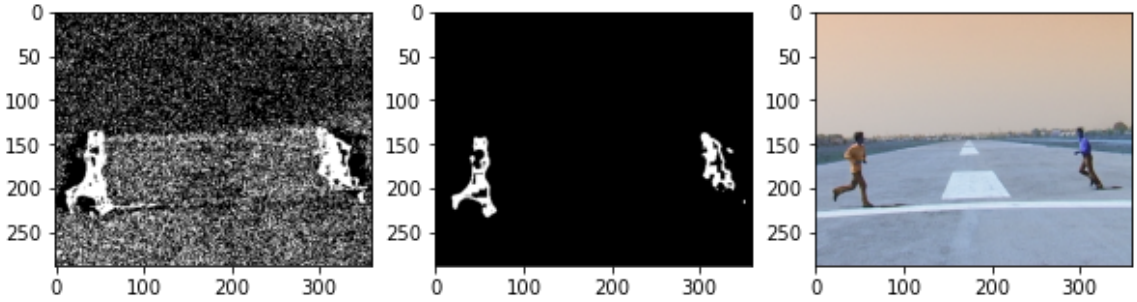
Using lambda as 0.72 and eta as 0.72



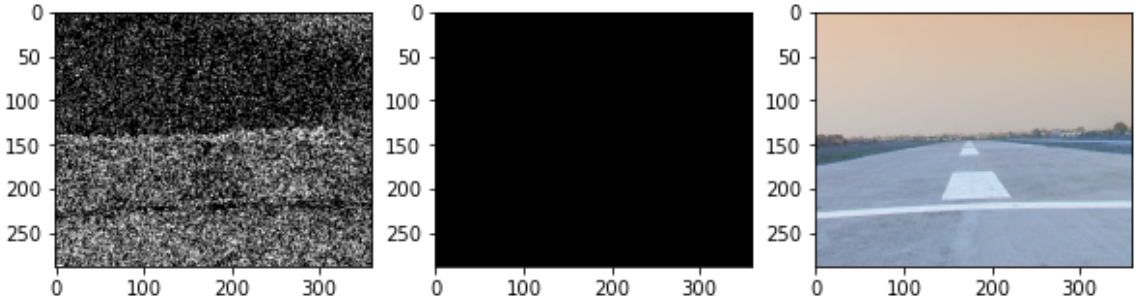

0
50
100
150
200
250
0
100
200
300

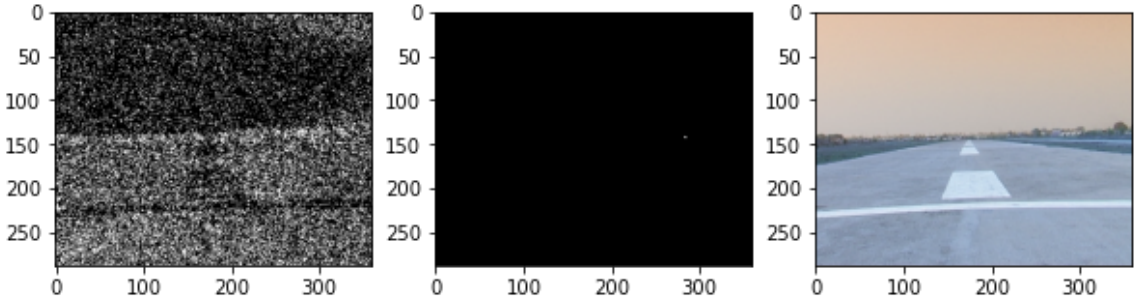

0
50
100
150
200
250
0
100
200
300

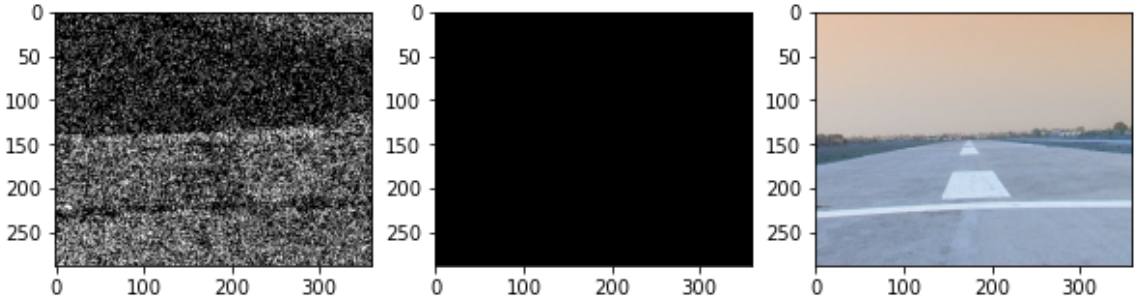


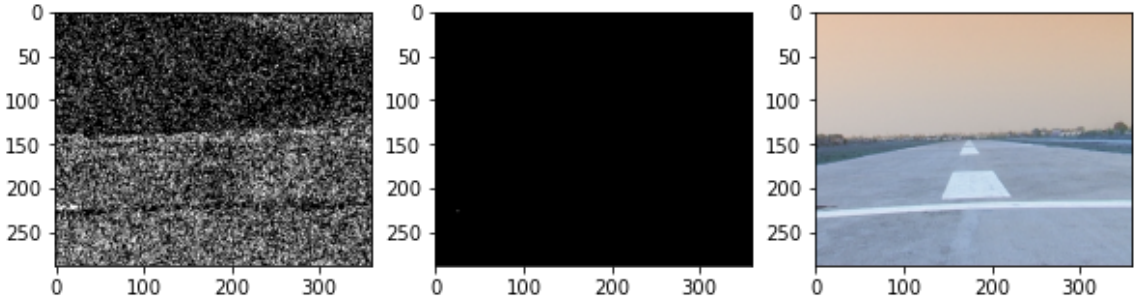


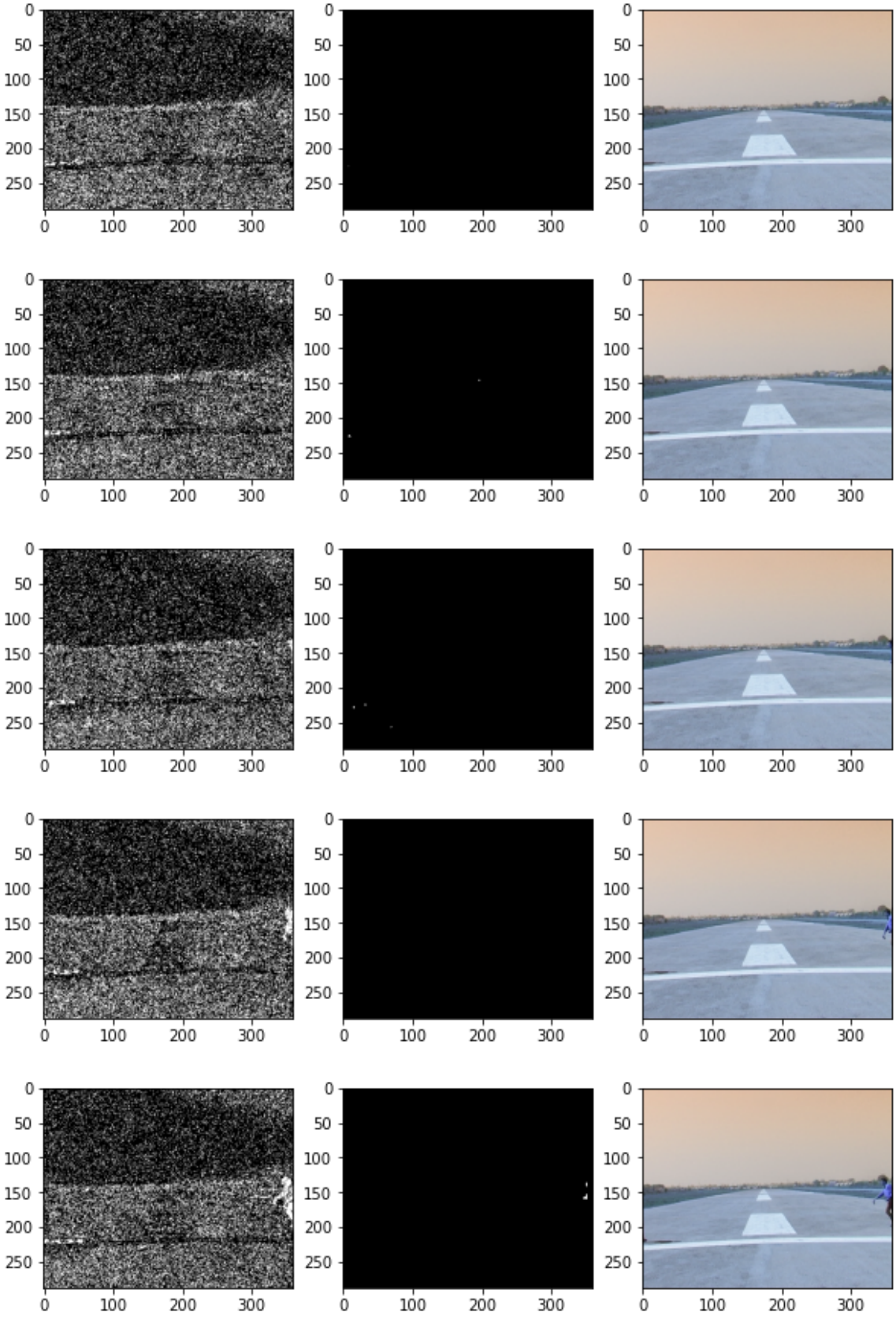
Using lambda as 0.74 and eta as 0.72

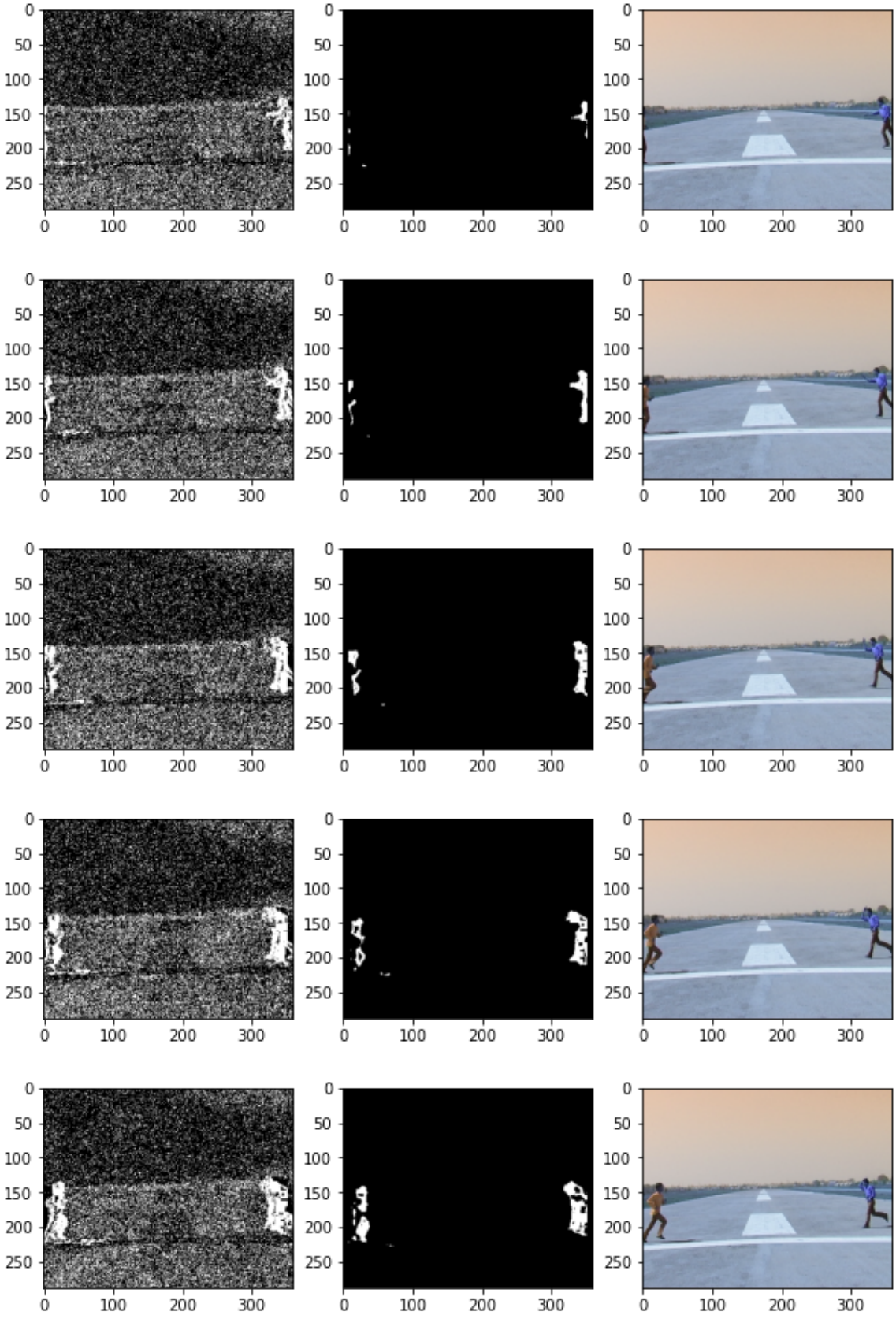




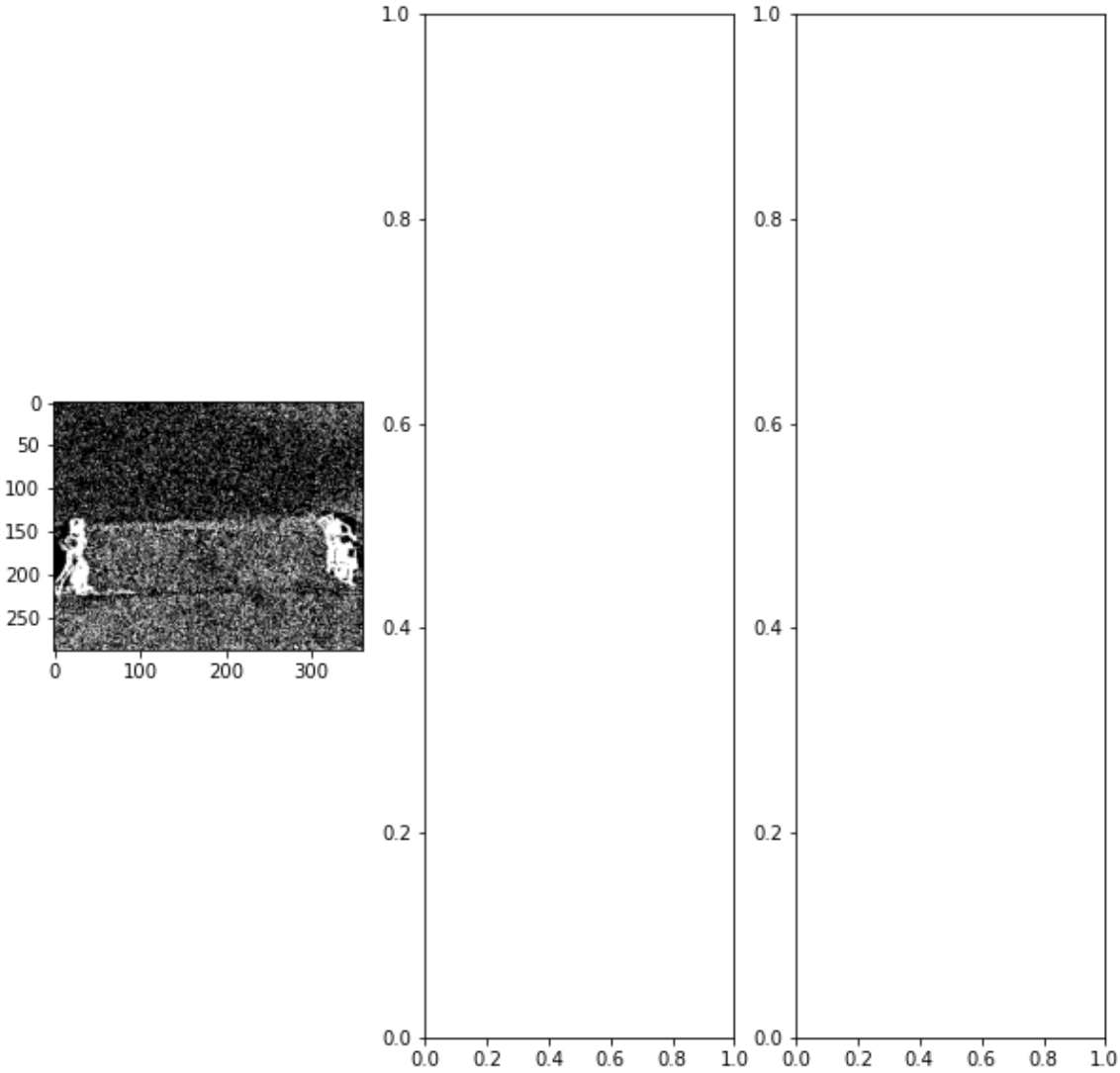





0 50 100 150 200 250
0 100 200 300



```
---------------------------------------------------------------------------
-
KeyboardInterrupt                         Traceback (most recent call las
t)
<ipython-input-32-704c5658c540> in <module>
      5     for lmda in t:
      6         print("Using lambda as {} and eta as {}".format(lmda,eta))
----> 7         mean,variance = Background_Subtraction("./Images",lmda,eta
,8,8,0.8)
      8         print("\n")

<ipython-input-27-a39208a00310> in Background_Subtraction(img_dir, lmda, e
ta, m, n, alpha)
     12
     13         cI = Voting(rI,eta,m,n)
---> 14         mean,variance = meanvarUpdate(cI,img_path,mean,variance,al
pha)
     15         ax[1].imshow(cI,cmap="gray")
     16

<ipython-input-26-36c78b7d8144> in meanvarUpdate(cI, img_path, M, V, alph
a)
     10                 mean_upd[i,j,:] = (1-alpha)*M[i,j,:] + alpha*img[i
,j,:]
     11                 var_upd[i,j,:] = (1-alpha)*(V[i,j,:] + alpha*d_2[i
,j,:])
---> 12                 var_upd[i,j,:] = np.clip(var_upd[i,j,:],a_min=9,a_
max=None)
     13     return(mean_upd,var_upd)

KeyboardInterrupt:
```


0
50
100
150
200
250
0
100
200
300
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0