```
In [1]: import numpy as np
        import cv2
```

Loading the given Imagefile.

```
In [2]: env1 = np.loadtxt('human_corridor_0.txt')
```

```
In [3]: env1
```

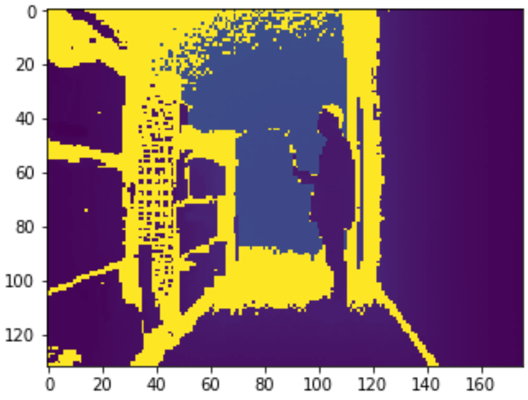
```
Out[3]: array([[32.76699829, 32.76699829, 32.76699829, ...,  1.60899997,
                 1.57599998,  1.55999994],
               [32.76699829, 32.76699829, 32.76699829, ...,  1.59399998,
                 1.58000004,  1.55499995],
               [32.76699829, 32.76699829, 32.76699829, ...,  1.59599996,
                 1.56799996,  1.551     ],
               ...,
               [ 1.79999995,  1.699     ,  1.71899998, ...,  1.61800003,
                 1.58599997,  1.58000004],
               [ 1.67400002,  1.722     ,  1.69400001, ...,  1.61399996,
                 1.59300005,  1.57700002],
               [32.76699829, 32.76699829, 32.76699829, ...,  1.62100005,
                 1.58800006,  1.56099999]])
```

```
In [4]: img=env1
```

```
In [5]: import matplotlib.pyplot as plt
```

```
In [6]: plt.imshow(img)
```

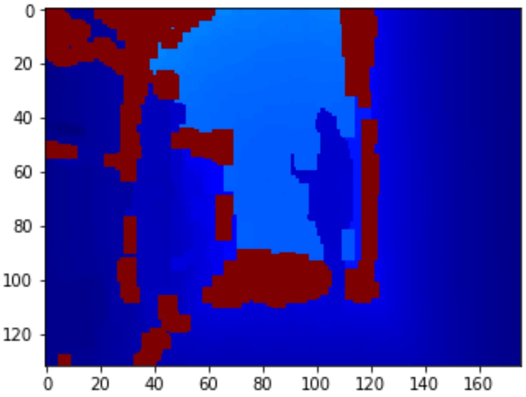
```
Out[6]: <matplotlib.image.AxesImage at 0x7f1244066350>
```



Applying erosion, dilation to clean the image.

```
In [7]:  kernel = np.ones((5,5),np.float32)/25
         plt.rcParams['image.cmap'] = 'jet'
         mask=cv2.inRange(img,0, 255)
         erosion =cv2.erode(img, kernel, iterations=1)
         dil =cv2.dilate(erosion, kernel, iterations=1)
         plt.imshow(dil)
```

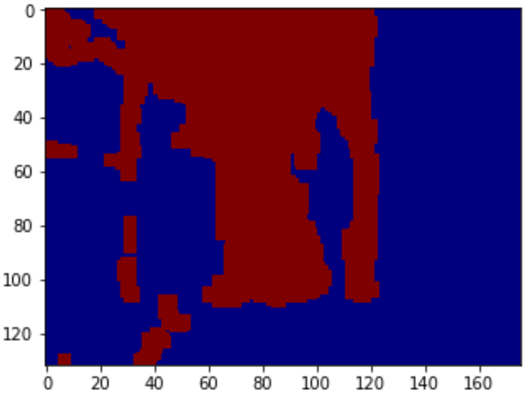
```
Out[7]: <matplotlib.image.AxesImage at 0x7f1243d97890>
```



Thresholding the image to get a clear view of human.

```
In [8]:  ret1, thresh1 = cv2.threshold(dil, 5, 10, 0)
         plt.imshow(thresh1)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x7f1243d17590>
```



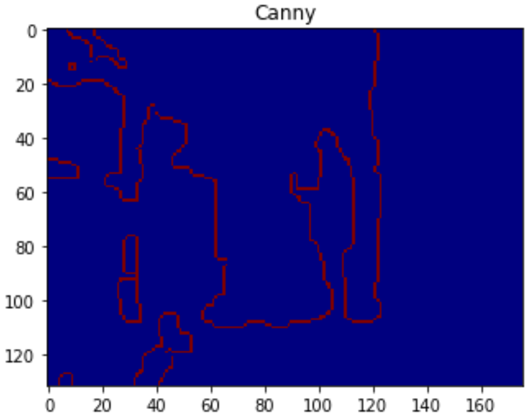
Applying canny edge detection.

In [9]:

```
Copy = np.uint8(thresh1)
canny = cv2.Canny(Copy,3,5)
plt.imshow(canny),plt.title('Canny')
```

Out[9]:

```
(<matplotlib.image.AxesImage at 0x7f124048a810>, Text(0.5,1,'Canny'))
```



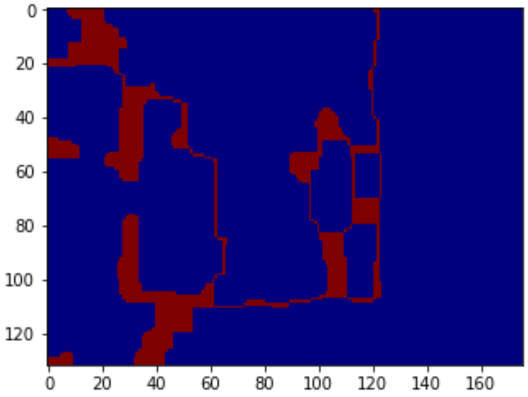
Seperating human contour from base.

In [10]:

```
canny=cv2.dilate(canny, kernel, iterations=2)
canny=cv2.erode(canny, kernel, iterations=2)
plt.imshow(canny)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x7f1240403950>
```



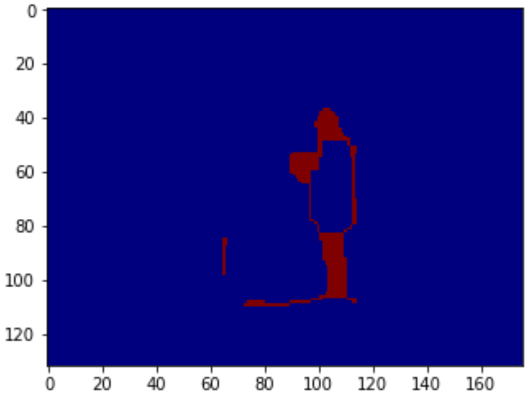
Crop required area (Assumption- same lobby corridor.)

In [11]:

```
cropped = np.zeros((canny.shape[0], canny.shape[1]), dtype=np.uint8)
cropped[30:110,65:115]=canny[30:110,65:115]
ret,thresh = cv2.threshold(cropped,127,255,cv2.THRESH_BINARY)
```

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

Out[12]:
```
<matplotlib.image.AxesImage at 0x7f1240383610>
```



Applying contour detection and filtering contours using hierarchy. Bounding box for final contour.

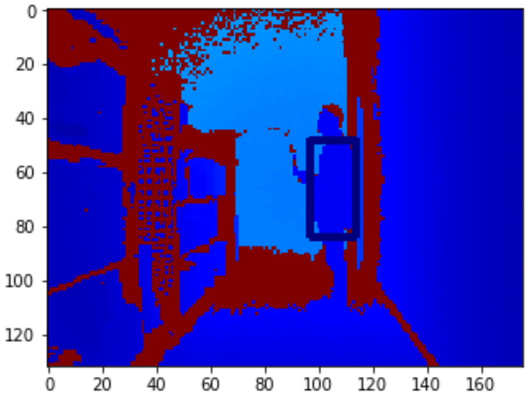
In [13]:

```
im2, contours, hierarchy = cv2.findContours(cropped,cv2.RETR_TREE,cv2.CHAIN
_APPROX_NONE)
max_area=0
human=0
hierarchy = hierarchy[0]

for component in zip(contours, hierarchy):
    Contour = component[0]
    Hierarchy = component[1]

    if Hierarchy[2] < 0:
        a=component
        area=cv2.contourArea(a[0])
        print(area)
        if (area>max_area):
            max_area=area
            human=a[0]
x, y, w, h = cv2.boundingRect(human)
kk=cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
print (x, y),(x+w, y+h)
plt.imshow(kk)
```

```
2.5
488.0
(97, 48) (114, 84)
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x7f1240313550>
```



In [14]:

```
x-3,y,w,h
```

Out[14]:

```
(94, 48, 17, 36)
```

Using the co-ordinates of bounding box calculating clearance for the robot.(+- 3 for feet clearance)

In [15]:

```
left = x-60-3
right = 120-(x+w+3)
if (left >right):
    print("left",left*1.5/60)
else:
    print("right",right*1.5/60)
```

```
('left', 0.85)
```