Option: Rather than pen-gesture recognition, carry out the project on a real video action classification dataset. Do all of the same steps and the other two extra-credit options1 are also available to you. If you choose this extra-credit option, then you do not need to also work with the pen-gesture data. For this, use the KTH Dataset available for download at http://www.nada.kth.se/cvap/actions/. KTH is a six-class dataset; work with them all. Working with this dataset may be more interesting to you but it is more open-ended and hence will be more challenging than the pen-gesture dataset because you have to decide how you will turn the video in a set of features on which to train the HMMs.

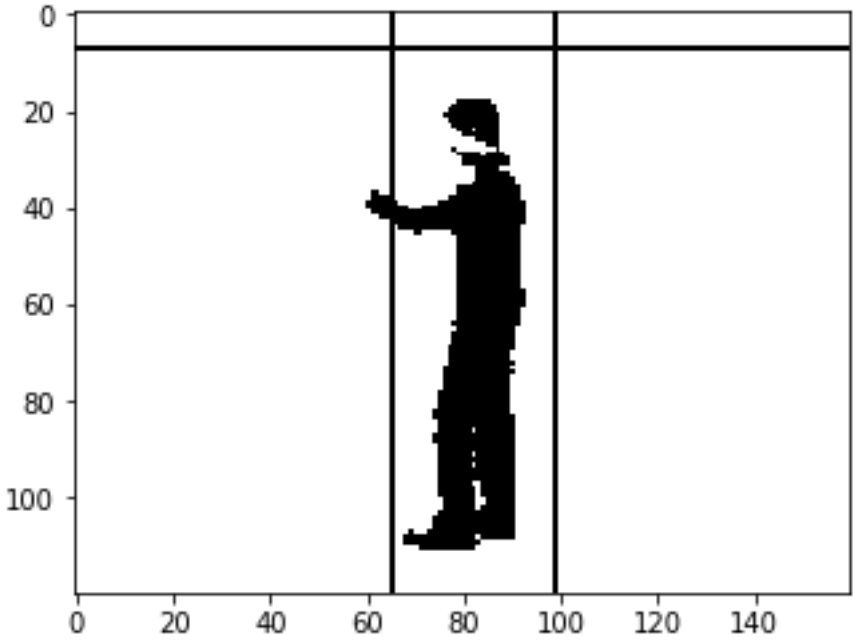
1.preprocessing

Since it is very slow for my computer to read the video, I converted the video into feature sequence and stored it in the "seq.txt" file. In this way, when training the HMM model, I only need to read the TXT file to get the sequence of actions in the video, which greatly improved the running speed. Since feature extraction uses the rules I preset, feature extraction of the test set is independent of the training set, which makes full preprocessing feasible.

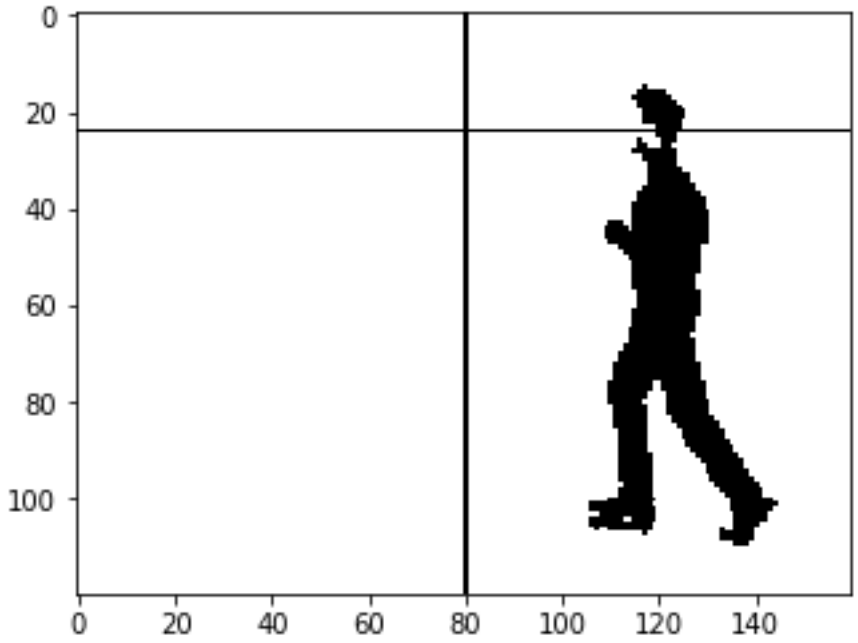
If there is no file "seq.txt" in the working environment, please create it manually first. Since the Cv2 library cannot read the files under the Chinese path, I put all the videos into the video folder and saved them on the desktop. Run pro2add_writetxt.py, which reads in the entire video and computes a sequence of features for each video based on the predefined feature extraction method (get_seq_method1/2). Since there are only 599 videos, Handclapping of Person13 has no D3. For data alignment, I've copied Handclapping of Person13 with D2 as D3.

2.feature extraction

Method1： To determine the location of a person by detecting the mutation of the column and row, draw three lines around the person, as shown in the figure below. Check left arm, right arm and head. According to whether the object is detected by the three lines, the combination can get 8 cases. These 8 cases are used as the characteristic sequence number of a frame in the video.



Method2： Draw a vertical line in the middle of the picture and a horizontal line in the head, as shown in the figure below, and you can get 4 states. The purpose of this method is to fix the position of the detection line, so as to better detect the movement process of the character. Walking, running, jogging may be able to distinguish clearly.



3.trianing

In accordance with the method of letter processing, each category is divided into odd and even, respectively into training set and test set. The training set was used to train HMM models for each action category. During the test, the forward algorithm was used to get the probability that the model generated the sequence. The probabilities provided by the six models were compared, and the current sequence was considered to be the model with the highest probability. The fig below is the test result in different observation states under different methods. Due to the long training time, only two results are listed here in the case of hidden state.

Method1：

ob_num=8, hidden_num=6

```
boxing 0.36
handclapping 0.38
handwaving 0.5
jogging 0.12
running 0.28
walking 0.5
[[18.  9. 13.  4.  4.  2.]
 [ 7. 19. 11.  2.  6.  5.]
 [ 3.  9. 25.  1.  7.  5.]
 [ 2.  4. 10.  6. 11. 17.]
 [ 2.  4.  8.  6. 14. 16.]
 [ 3.  4.  7.  7.  4. 25.]]
```

```
boxing 0.5
handclapping 0.32
handwaving 0.38
jogging 0.04
running 0.24
walking 0.64
[[25.  6. 12.  1.  3.  3.]
 [12. 16. 16.  1.  1.  4.]
 [18. 10. 19.  0.  1.  2.]
 [ 2.  2.  6.  2.  8. 30.]
 [ 1.  1.  6.  4. 12. 26.]
 [ 3.  1.  6.  7.  1. 32.]]
```

ob_num=8, hidden_num=4

```
boxing 0.36
handclapping 0.42
handwaving 0.28
jogging 0.38
running 0.22
walking 0.02
[[18. 11.  8.  8.  1.  4.]
 [ 5. 21. 11. 11.  1.  1.]
 [ 4. 16. 14. 11.  1.  4.]
 [ 4.  5. 13. 19.  3.  6.]
 [ 6.  4.  8. 14. 11.  7.]
 [ 8.  4. 13. 23.  1.  1.]]
```

```
boxing 0.38
handclapping 0.38
handwaving 0.28
jogging 0.26
running 0.08
walking 0.48
[[19.  8. 11.  1.  5.  6.]
 [12. 19. 13.  4.  1.  1.]
 [10. 12. 14.  6.  2.  6.]
 [ 4.  8.  7. 13.  2. 16.]
 [ 4.  8.  2. 15.  4. 17.]
 [ 4. 11.  0. 10.  1. 24.]]
```

ob_num=8, hidden_num=8

```
boxing 0.58
handclapping 0.36
handwaving 0.24
jogging 0.2
running 0.32
walking 0.4
[[29.  4.  6.  2.  3.  6.]
 [18. 18.  7.  0.  0.  7.]
 [16. 11. 12.  0.  1. 10.]
 [10.  5.  1. 10.  9. 15.]
 [ 7.  4.  2.  9. 16. 12.]
 [ 9.  6.  0. 10.  5. 20.]]
```

```
boxing 0.44
handclapping 0.38
handwaving 0.28
jogging 0.08
running 0.22
walking 0.74
[[22.  6.  8.  5.  2.  7.]
 [ 9. 19.  6.  7.  1.  8.]
 [15. 12. 14.  1.  1.  7.]
 [ 2.  1.  4.  4.  7. 32.]
 [ 2.  1.  4.  4. 11. 28.]
 [ 2.  3.  3.  4.  1. 37.]]
```

Method2：

ob_num=4, hidden_num=6

```
boxing 0.5
handclapping 0.6
handwaving 0.6
jogging 0.22
running 0.3
walking 0.98
[[25.  6. 10.  3.  1.  5.]
 [ 5. 30.  9.  0.  1.  5.]
 [ 6.  9. 30.  3.  0.  2.]
 [ 2.  0.  1. 11.  9. 27.]
 [ 2.  2.  0. 19. 15. 12.]
 [ 1.  0.  0.  0.  0. 49.]]
```

```
boxing 0.46
handclapping 0.62
handwaving 0.56
jogging 0.3
running 0.5
walking 0.9
[[23.  7. 13.  2.  2.  3.]
 [ 2. 31. 11.  1.  2.  3.]
 [ 5.  9. 28.  1.  6.  1.]
 [ 1.  1.  1. 15. 12. 20.]
 [ 1.  3.  1.  5. 25. 15.]
 [ 1.  0.  0.  4.  0. 45.]]
```

ob_num=4, hidden_num=4

```
boxing 0.36
handclapping 0.5
handwaving 0.52
jogging 0.2
running 0.4
walking 0.98
[[18.  6. 16.  0.  3.  7.]
 [ 7. 25.  6.  0.  3.  9.]
 [ 5. 10. 26.  4.  2.  3.]
 [ 1.  0.  0. 10. 11. 28.]
 [ 1.  1.  0. 10. 20. 18.]
 [ 1.  0.  0.  0.  0. 49.]]
```

```
boxing 0.4
handclapping 0.62
handwaving 0.36
jogging 0.32
running 0.12
walking 0.98
[[20.  9. 14.  2.  1.  4.]
 [10. 31.  1.  1.  0.  7.]
 [16.  8. 18.  4.  2.  2.]
 [ 1.  1.  0. 16.  3. 29.]
 [ 1.  0.  1. 24.  6. 18.]
 [ 1.  0.  0.  0.  0. 49.]]
```

ob_num=4, hidden_num=8

```
boxing 0.52
handclapping 0.22
handwaving 0.56
jogging 0.3
running 0.52
walking 0.9
[[26.  3. 12.  2.  4.  3.]
 [24. 11.  9.  0.  5.  1.]
 [11.  3. 28.  0.  8.  0.]
 [ 1.  0.  1. 15. 13. 20.]
 [ 0.  2.  2.  8. 26. 12.]
 [ 1.  0.  1.  3.  0. 45.]]
```

```
boxing 0.46
handclapping 0.56
handwaving 0.62
jogging 0.3
running 0.56
walking 0.9
[[23.  5. 10.  4.  5.  3.]
 [ 3. 28. 10.  0.  8.  1.]
 [ 3.  8. 31.  0.  8.  0.]
 [ 2.  0.  0. 15. 13. 20.]
 [ 2.  1.  0.  8. 28. 11.]
 [ 1.  0.  0.  4.  0. 45.]]
```

ob_num=4, hidden_num=10

```
boxing 0.42
handclapping 0.46
handwaving 0.6
jogging 0.2
running 0.52
walking 0.94
[[21.  5. 16.  2.  4.  2.]
 [ 6. 23. 15.  1.  4.  1.]
 [ 6.  7. 30.  0.  7.  0.]
 [ 1.  0.  2. 10. 13. 24.]
 [ 0.  1.  3.  9. 26. 11.]
 [ 1.  0.  1.  1.  0. 47.]]
```

```
boxing 0.48
handclapping 0.18
handwaving 0.54
jogging 0.26
running 0.42
walking 0.88
[[24.  6. 13.  2.  2.  3.]
 [28.  9.  8.  2.  0.  3.]
 [15.  3. 27.  3.  1.  1.]
 [ 2.  0.  1. 13. 13. 21.]
 [ 2.  1.  1.  9. 21. 16.]
 [ 1.  0.  0.  5.  0. 44.]]
```

4.analysis

The construction of Hmm model is consistent with pro2's vowel recognition, so it will not be repeated here.

The difference lies in the feature extraction method. I designed two feature extraction methods. I originally thought that the first method could

better detect the movement of the human arm with slightly higher accuracy. Here I looked at the method1 checkpoint frame by frame, and found that the current parameters don't guarantee that the checkpoint will always be where it's supposed to be. Considering the complexity of the action, I think it is difficult to design a detector that can fit all the actions in the sample, so the debugging of the detector is no longer carried out here.

Settings for hidden states. I tested the obturation matrix with hidden state of 4, 6 and 8 respectively. When the hidden state was 8, the highest accuracy rate was 0.55 to 0.6, but when the hidden state was 6 and 10, the accuracy rate was basically around 0.5. This indicated that the change of hidden state had little effect on the improvement of accuracy, and the bottleneck of the improvement of accuracy was the extraction of features.