SYSC5104: Methodologies in Discrete Event Modeling and Simulation

Assignment 1:

Human Motion Classification

Seyed Ali Etemad

Student Number: 100333627

Professor: Gabriel Wainer

**PART I**

The proposed system to be modelled for this assignment is a Human Motion Classification system. The goal of the system is to employ motion capture data and train a classifier capable of recognizing specific classes of action such as walk, jump, or run. The overall system is composed of 3 sections. One of which is composed of several sub-models as shown in more details in Figure 1. Figure 1 presents the 3 major sub-models of the system: the motion capture system, the pre-processor, and the classifier.

**motion\_capture\_system**: as presented in Figure 1, the motion capture system is composed of two sub-models itself, the pc (computer) and cameras. Once the motion\_capture\_system is signalled to initiate recording, the pc turns the cameras on and they capture the motion data and transfer them to the pc for digitization and recording. The pc then sends the data to the preprocessor.

**pc:** the pc is the computer responsible for digitizing and sending out the motion capture data to different sections of the system.

**cameras:** are a set of at least 6 very high accuracy cameras placed on different parts of the motion capture room and connected to the pc.

**preprocessor:** the preprocessor performs a series of pre-processing algorithms on the data for noise elimination and dimensionality reduction for optimum performance. The processed data are then transferred to the classifier.

**classifier:** the classifier is first trained using a series of processed data. Once the system is turned on the motion\_capture\_system will acquire enough data to train the classifier. The training can take anywhere from 1 to 9 training samples. Once the training is complete, the system will await initiation by the user for acquiring test data. The test data are transferred through the same path as the training data. The test data will be classified, and a classification accuracy of between 90% to 94 % will be resulted.

Following is a description of all the ports in the system:

init: signals the motion\_capture\_system to start running either for acquiring a complete set of training data or for acquiring a test sample.

pcI: signals the pc to start the data capturing procedure.

pc\_cam: turns the cameras on for the data capturing procedure.

cam\_pc: transfers the captured scenes to the pc.

DOut: transfers the recorded data to the preprocessor via the DIn.

DP: transfers the processed data to the classifier.

trainAcc: will determine if the training process is complete. Once training is accomplished successfully, it will signal the motion capture system stop acquiring knew data, and wait for user to initiate the testing process.

result: if the training is complete, the new data are classified using the classifier and the result is provided. The results range from 90% to 94%.

result

**Human Motion Classification System (HMC)**

**motion\_capture\_system**

pc\_cam

cam\_pc

pc

cameras

DOut

DIn

DP

**preprocessor**

**classifier**

trainAcc

tr\_ts

init

pcI

Figure 1. A model sketch of a human motion classification system.

**PART II**

**Formal Specifications**

The formal specifications < S, X, Y, δint, δext, λ, ta > for the atomic models are defined as follows:

**pc:**

S = {Off, Start\_Cam\_Test, Start\_Cam\_Train, Send\_Cam\_Train, Send\_Cam\_Test, Recv\_Cam\_Train, Recv\_Cam\_Test}

X = {tr\_ts, cam\_pc, pcI}

Y = {pc\_cam, DOut}

δint (Start\_Cam\_Train) = Recv\_Cam\_Train

δint (Send\_Cam\_Train) = Start\_Cam\_Train

δint (Start\_Cam\_Test) = Recv\_Cam\_Test

δint (Send\_Cam\_Test) = Off, passive mode

δext (pcI = 1, Off) = Start\_Cam\_Train

δext (tr\_ts, Recv\_Cam\_Train) = Off

δext (tr\_ts, Start\_Cam\_Train) = Off

δext (cam\_pc, Recv\_Cam\_Train) = Send\_Cam\_Train

δext (pcI = 2, Off) = Start\_Cam\_Test

δext (cam\_pc, Recv\_Cam\_Test) = Send\_Cam\_Test

λ(Start\_Cam\_Train){

pc\_cam = 1

}

λ(Send\_Cam\_Train){

DOut = 1

}

λ(Start\_Cam\_Test){

pc\_cam = 1

}

λ(Send\_Cam\_Test){

DOut = 1

}

ta(Off) = INFINITY

ta(Start\_Cam\_Test) = Delay for 10 Milliseconds

ta(Start\_Cam\_Train) = Delay for 10 Seconds

ta(Send\_Cam\_Train) = Delay for 10 Milliseconds

ta(Send\_Cam\_Test) = Delay for 10 Milliseconds

ta(Recv\_Cam\_Train) = INFINITY

ta(Recv\_Cam\_Test) = INFINITY

**cameras:**

S = {passive, active}

X = {pc\_cam}

Y = {cam\_pc}

δint (active) = passive

δext (pc\_cam, passive) = active

δext (pc\_cam, active) = active

λ(active) {

cam\_pc = 1

}

ta(passive) = INFINITY

ta(active) = Delay for 6 seconds

**preprocessor:**

S = {passive, active}

X = {DIn}

Y = {DP}

δint (active) = passive

δext (in, passive) = active

δext (in, active) = active

λ(active) = {

DP = 1

}

ta(passive) = INFINITY

ta(active) = Delay for 5 seconds

**classifier:**

\*\*\* The value counter is the number of training samples.

S = {Training, Send\_Training\_Result, Testing, Send\_Testing\_Result}

X = {DP}

Y = {trainAcc, result}

IF ( (counter < 9) and (counter \* 10 < testprob) ) THEN δint (Send\_Training\_Result) = Training

ELSE δint (Send\_Training\_Result) = Testing

δint (Send\_Testing\_Result) = Testing

δext (DP, Training) = Send\_Training\_Result; counter++

δext (DP, Testing) = Send\_Testing\_Result

λ(Send\_Training\_Result)

{

testprob = random number between 0 and 99

IF ( (counter >= 9) or (testprob < counter \* 10) ) THEN trainAcc = 1

}

λ(Send\_Testing\_Result)

{

result = random number between 90 and 94

}

ta(Training) = INFINITY

ta(Testing) = INFINITY

ta(Send\_Testing\_Result) = Delay for 3 seconds

ta(Send\_Training\_Result) = Delay for 3 seconds

The formal specifications <X, Y, D, {Mi}, {Ii}, {Zij}, SELECT > for the coupled models motion\_capture\_system and HMC are defined as follows:

**motion\_capture\_system:**

X = {init, trainAcc};

Y = {DOut};

D = {pc, cameras};

I(pc) = {cameras,self};

I(cameras) = {pc};

Z(pc) = cameras; Z(pc) = self;

Z(cameras) = self;

SELECT: ({pc, cameras}) = pc;

**HumanMotionClassification:**

X = {init};

Y = {result};

D = {motion\_capture\_system, preprocessor, classifier};

I(motion\_capture\_system) = {preprocessor, self};

I(preprocessor) = {motion\_capture\_system, classifier};

I(classifier) = {preprocessor, self};

Z(motion\_capture\_system) = preprocessor; Z(motion\_capture\_system) = self;

Z(preprocessor) = motion\_capture\_system; Z(preprocessor) = classifier;

Z(classifier) = preprocessor; Z(classifier) = self

SELECT: ({motion\_capture\_system, preprocessor, classifier}) = motion\_capture\_system;

({preprocessor, classifier}) = preprocessor;

**PART III**

**Test Strategy**

**preprocessor:**

To test the proposed model and all atomic models involved, we should start from the simplest model and add on other sections based on the order of data flow and simplicity. In the case of this system, the simplest atomic model is the *preprocessor*.

The *preprocessor* simply takes an input and outputs the same data after a series of manipulations. This atomic model has a delay of 5 seconds in real trials. No data is lost or delayed by more or less than 5 seconds. A value of 1 through *DIn* marks the entrance of the data as a 1 through the *DP* port resembles the output of the data.

The following event is used to test this atomic model:

00:00:06:00 DIn 1

**00:00:10:00 DIn 1**

00:00:15:00 DIn 1

**00:00:18:00 DIn 1**

**00:00:19:00 DIn 1**

**00:00:20:00 DIn 1**

00:00:25:00 DIn 1

00:00:32:00 DIn 1

00:00:40:00 DIn 1

00:01:10:00 DIn 1

If a set of data is being processed and a new set of data is received, the new set of data is discarded. Based on this rule, there will not be an output created for the inputs above shown in Bold. This model is working correctly as the following .out file was resulted similar to what was expected. According to the input data presented above, for each of the valid data (another set of data is not being processes), there is an output generated 5 seconds afterwards.

00:00:11:00 DP 1

00:00:20:00 DP 1

00:00:30:00 DP 1

00:00:37:00 DP 1

00:00:45:00 DP 1

00:01:15:00 DP 1

**Classifier:**

The next atomic model which is examined is the *classifier*. The classifier has one input, *DP*, which is the direct output of the *preprocessor*. Once the classifier *receives* the data for the first time, it starts the training state. The system is guaranteed to become fully trained if 9 set of training data is received. For any number of training samples less than 9, there is a linear probability of the *classifier* becoming fully trained. For the ith data set (i < 9), there is a (i \* 10) % probability of the system becoming fully trained. This is because there is no certainty on when training will be fully accomplished. In the research conducted earlier, the system was guaranteed to be fully trained after being trained by 9 set of data. Yet in some cases, full training was accomplished using fewer data sets. The closer the number of training sets become to 9, there is more chance of accomplishing full training.

After the system has been trained, a value of 1 is sent through the feedback port *trainAcc* which signals the end of the training phase. From this point on, the classifier enters the state of testing and upon receipt of new data; the classification result is measured and sent through the port *result*. The classification accuracy is assumed to be between 90% and 94%. This range is simply based on an estimate of the performance of the classifier.

The classifier takes 3 seconds for making use of each set of received data, either for training or for testing purposes.

The following event is used for testing this atomic model

00:00:01:00 DP 1

**00:00:04:00 DP 1**

00:00:08:00 DP 1

00:00:13:00 DP 1

**00:00:15:00 DP 1**

00:00:27:00 DP 1

00:00:32:00 DP 1

00:01:05:00 DP 1

00:01:18:00 DP 1

00:01:28:00 DP 1

00:01:34:00 DP 1 🡨

00:01:39:00 DP 1

00:01:45:00 DP 1

**00:01:47:00 DP 1**

00:01:50:00 DP 1

The lines in Bold will be discarded. The line marked by the arrow is the 9th set of legitimate data. The output file MUST show a *trainAcc* = 1 prior to or on this line. Once the *trainAcc* = 1 is accomplished, the port *result* must show numbers for all other line. These numbers must be greater than 90 and smaller than 94.

The following .out file was resulted:

00:01:08:000 trainacc 1

00:01:21:000 result 92

00:01:31:000 result 91

00:01:37:000 result 92

00:01:42:000 result 93

00:01:48:000 result 91

00:01:53:000 result 91

The successful training of the system was accomplished for the data received on 00:01:05:00 and the output was generated 3 seconds afterwards. As we can see, the output for this input was created as *trainAcc* = 1. This means that the training is now done (using the first 6 sets of non-discarded data), and further data will produce the classification result. This is seen in the .out file for data sets after the one received on 00:01:05:00, the result was produced. The result values follow the constraint (between 90 and 94) explained earlier.

**Cameras:**

The next model to be tested is the *cameras*. The task of the *cameras* is quite simple. Upon activation using the *pc\_cam* port (receiving a 1), they will acquire the environment data (motion capture data). The data will then be transferred to the *pc* through the *cam\_pc* port. The process takes 6 seconds since all studied motion sequences are assumed to take this long.

The following event was used to test the model:

00:00:02:00 pc\_cam 1

**00:00:03:00 pc\_cam 1**

**00:00:04:00 pc\_cam 1**

**00:00:05:00 pc\_cam 1**

00:00:10:00 pc\_cam 1

**00:00:15:00 pc\_cam 1**

00:00:18:00 pc\_cam 1

00:00:27:00 pc\_cam 1

00:00:35:00 pc\_cam 1

00:00:47:00 pc\_cam 1

The lines in Bold have been introduced less than 6 seconds after the previous event and will be discarded.

The following .out file was resulted which confirms the performance of the atomic model:

00:00:08:000 cam\_pc 1

00:00:16:000 cam\_pc 1

00:00:24:000 cam\_pc 1

00:00:33:000 cam\_pc 1

00:00:41:000 cam\_pc 1

00:00:53:000 cam\_pc 1

**Pc:**

The *pc* is initially in the *Off* state. When the system is in training mode, the *pc* takes one of the following 3 states: *Start\_Cam\_Train, Recv\_Cam\_Train, Send\_Cam\_Train*. The *pc* first starts the *cameras* for training in *Start\_Cam\_Train* state using the port *pc\_cam*. Then it receives the data from the *cameras* in *Recv\_Cam\_Train* state through *cam\_pc*. The *pc* then sends out the data through *DOut* to the *preprocessor*. All this occurs after a value of 1 is received through *pcI*. There is a hard constraint on *pcI*. The first time it is activated, it produces a “1”. The rest of the time it produces “2”. A value of 1 on *pcI* will start the training process. This process continues automatically until training is complete (1 is returned via *tr\_ts*). To ensure that the data travels through the *preprocessor* and *classifier* and there is enough time for *tr\_ts* to return 1 (if that is the case), there is a 10 second delay for starting the *cameras* in state *Start\_Cam\_Train*. The rest of the states however, happen almost instantly (10 milliseconds). After training is complete (*tr\_ts* returns 1), the pc goes into *Off* state again and waits for *pcI*, this time to activate it via the value 2. Each time *pcI* is activated (value 2), the *pc* goes through the following 3 states: *Start\_Cam\_Test, Recv\_Cam\_Test, Send\_Cam\_Test.* The only difference is that this time, there is no need for the 10 second delay before starting the cameras since we are not waiting for *tr\_ts* to return anything.

The following event is used to test this atomic model:

00:00:01:10 tr\_ts 1

00:00:03:10 cam\_pc 1

00:00:10:00 pcI 1

00:00:25:00 cam\_pc 1

The following was the outcome:

00:00:20:000 pc\_cam 1

00:00:25:010 dout 1

00:00:35:010 pc\_cam 1

As we can see, before starting the system (sending the value 1 through *pcI*), *tr\_ts* and *cam\_pc* have been ignored. After starting the system, successive to a 10 second delay, the *cameras* are signaled through *pc\_cam* to start recording. The data received through *cam\_pc* at 00:00:25:00 has been quickly sent out to the *preprocessor* (in 10 milliseconds) as planned. 10 seconds later, the *cameras* are automatically signaled to start recording (at 00:00:35:010 through *pc\_cam*). This will go on until the classifier is trained and *tr\_ts* returns 1.

Again the same event file is created:

00:00:01:10 tr\_ts 1

00:00:03:10 cam\_pc 1

00:00:10:00 pcI 1

00:00:25:00 cam\_pc 1

00:00:39:00 cam\_pc 1

00:00:43:00 tr\_ts 1

00:00:47:00 pcI 2

00:00:57:00 cam\_pc 1

00:01:07:00 pcI 2

00:01:27:00 cam\_pc 1

**00:01:27:005 pcI 2**

**00:01:27:006 cam\_pc 1**

00:01:40:00 pcI 2

**00:01:40:005 pcI 2**

**00:01:40:006 cam\_pc 1**

The classifier is trained at 00:00:43:00. As we can see, with the activation of *pcI* (value 2) after this event, the *pc* turns on the *cameras* in 10 milliseconds of initiation (*pcI* = 2), and sends out the data in 10 milliseconds as well. The events above shown in bold are discarded since they have occurred before completion of the previous tasks.

00:00:20:000 pc\_cam 1

00:00:25:010 dout 1

00:00:35:010 pc\_cam 1

00:00:39:010 dout 1

00:00:47:010 pc\_cam 1

00:00:57:010 dout 1

00:01:07:010 pc\_cam 1

00:01:27:010 dout 1

00:01:40:010 pc\_cam 1

**motion\_capture\_system**

The motion capture system is composed of two atomic models. The *pc* and the *cameras*. This coupled model is highly influenced by the *pc* since the cameras only produce activate the port *cam\_pc* after 6 seconds of being activated themselves.

At first a simple test is conducted. We have designed the system to continue acquiring data until the classifier is fully trained. Therefore if we initiate the *motion\_capture\_system* through *pcI* and not activate *tr\_ts*, the model must continue acquiring data for ever. To test this the following event was introduced:

00:00:01:00 pcI 1

The result requires manual termination of the simulation which confirms the stated fact.

Then, the following event file was used for testing:

00:00:01:00 pcI 1

00:02:00:00 tr\_ts 1

00:02:37:00 pcI 2

00:02:57:00 pcI 2

**00:03:15:00 tr\_ts 1**

00:03:18:00 pcI 2

00:03:29:00 pcI 2

**00:03:30:00 pcI 2**

**00:03:30:010 pcI 2**

The following was resulted.

00:00:17:010 dout 1

00:00:33:020 dout 1

00:00:49:030 dout 1

00:01:05:040 dout 1

00:01:21:050 dout 1

00:01:37:060 dout 1

00:01:53:070 dout 1

00:02:43:020 dout 1

00:03:03:020 dout 1

00:03:24:020 dout 1

00:03:35:020 dout 1

The test shows that up to 00:02:00:00 the system continues training. Each training takes 16 seconds and 10 milliseconds which is the correct amount. At 00:02:00:00 *tr\_ts* returns 1 and training stops. After that each time the system is activated, it only takes 6 seconds and 20 milliseconds which is correct. As expected the second *tr\_ts* is ignored and the last two event are also discarded since there is processing being done on other data.

**HMC:**

HMC is the entire simulator. Since each of the atomic models are working properly, and the coupled model is also functioning properly, the simulator is expected to perform as expected. The system contains one input and one output. With sending a value of 1 through *init*, the system starts to operate. After the system has accomplished the training process, users can not send 1 through *init* again. At this point each time 2 is sent through the *init* port, a classification result between 90 and 94 (percent) must be returned through the output port *result*.

The following event file was used for testing the system:

00:00:01:00 init 1

**00:00:04:00 init 2**

**00:00:10:00 init 2**

**00:00:20:00 init 2**

**00:00:40:00 init 2**

**00:01:20:00 init 2**

00:02:00:00 init 2

**00:02:03:00 init 2**

00:03:00:00 init 2

**00:03:06:00 init 2**

00:03:06:050 init 2

**00:03:07:00 init 2**

**00:03:08:00 init 2**

00:05:00:00 init 2

00:10:00:00 init 2

The result was the following:

00:02:14:020 result 92

00:03:14:020 result 91

00:03:20:070 result 92

00:05:14:020 result 93

00:10:14:020 result 91

We can see that it has taken less than 2 minutes for the system to train properly since the first classification result is observed at 00:02:14:020 which corresponds to the input received at 00:02:00:00.

As we can calculate from previous sections of this report, it takes 6 seconds and 20 milliseconds for the *motion\_capture\_system* to acquire the data and send it to the *preprocessor*. It takes 5 seconds for the data to be processed and 3 seconds for the *classifier* to classify the data. Therefore if the system is initiated again before 6 seconds and 20 milliseconds, the command will be discarded. Initiation after this time however, will result in correct classification.

The data shown in bold format have been discarded and the rest have been used for classification.