

Project Report for CS296
Cheetah Simulation
Group 17

Naveen Sagar
120050026 , 120050026@cse.iitb.ac.in

Aditya Kumar Akash
120050046 , adityaakash@cse.iitb.ac.in

Prateek Chandan
120050042 , prateekchandan@cse.iitb.ac.in

October 5, 2014

1 Indroduction

This document gives details of the Project report for the course CS296 Software Systems Laboratory, taken by Prof Parag Chaudhary [2]. It contains The details of the new cheetah simulation. Cheetah is the fastest running animal on the planet. In our Box2D simulation project we have tried to model the simulation of cheetah's body, its balancing along with its basic motion like walk and jump

2 The Aniamtion

2.1 Model submitted in abstract

The model submitted in the project proposal was :

Each legs has three moving parts connected to a hinge which is shown in the diagram. They can rotate and give motion to the model. Also the rib, tail and the neck of the model is made of small polygons which can swing and maintain the body of the cheetah.

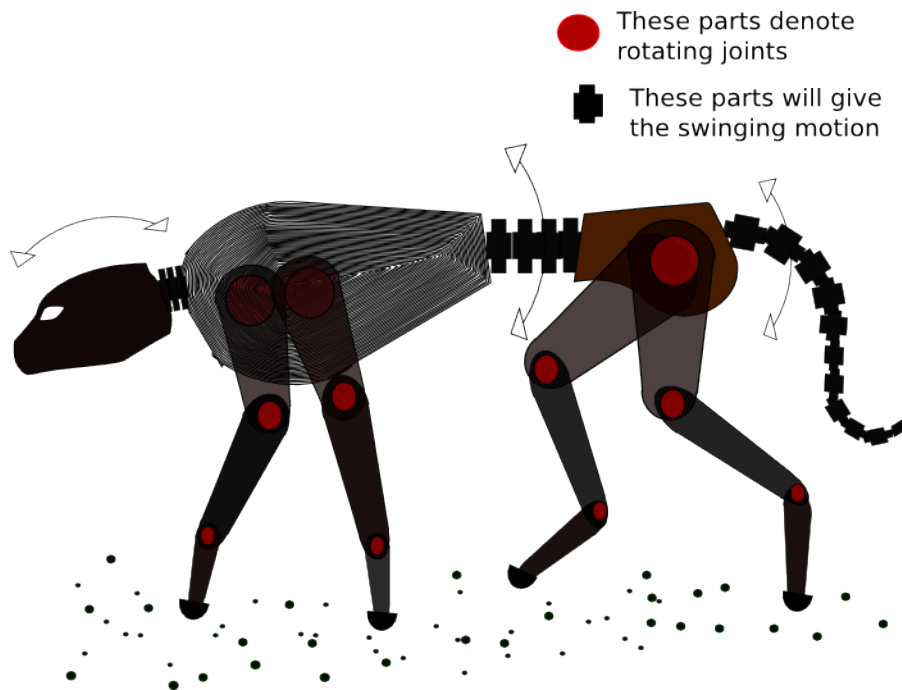


Figure 1: The model submitted in abstract

2.2 Final Box2d Model of the Cheetah

The model submitted in the project proposal was :

Each legs has three moving parts connected to a hinge which is shown in the diagram. They can rotate and give motion to the model. Also the rib, tail and the neck of the model is made of small polygons which can swing and maintain the body of the cheetah.

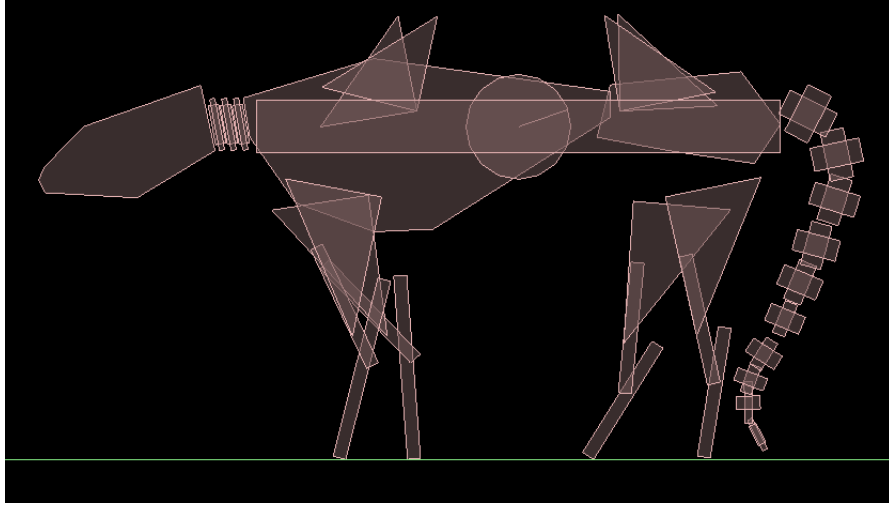


Figure 2: The completed model in Box2D

2.3 Deviation from the design

Removals The part which is missed in the final design in the rib of the cheetah. The main reason to remove it was the balancing problem of the cheetah. As due to the changed and different body parts, a firm center part was required

Additions We added the muscles to the leg of the cheetah making the balancing of the cheetah to be more realistic and also we added Theo Jansen [3] Walking mechanism for Cheetah which is a wheel based walking mechanism

3 Physics behind the simulation and details of the mechanism used

This section contains information about the different parts of cheetas body which is used in the animation

1. Walking Mechanism
2. The legs and Muscles
3. The Tail
4. Neck and Head
5. Main body

3.1 Walking Mechanism

The Walking Mechanism of cheetah is based on Theo Jansesn [3] walking mechanism

Theo Jansen linkage It is a crank-based where the circular movement of one point in the linkage translates to movement elsewhere in the linkage; this means the circularly moving part of the linkage can be attached to a crank so that the motion of the mechanism is easily driven allowing easy locomotion

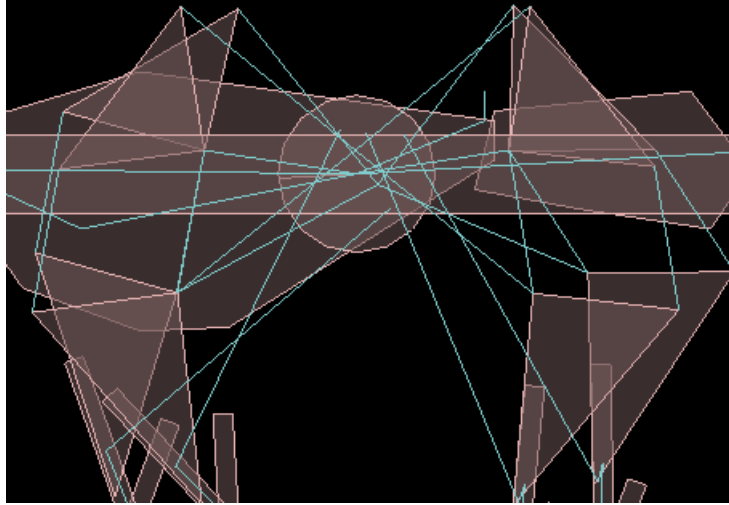


Figure 3: The image showing Walking Mechanism

3.2 The legs and Muscles

Description of Leg There are 4 legs in total of the cheetah two in the front while two in the back. Each of the leg consists of two bony parts joined by the revolute joints with constraint in the angle of rotation so as to maintain the shape of the leg

These legs are then joined by the theo jansen walking mechanism

Distance Joints as Muscles The walking mechanism is joined by distance joint which acts as springs which act as shock absorber for cheetah. It also helps in balancing of the body and legs after few deformations

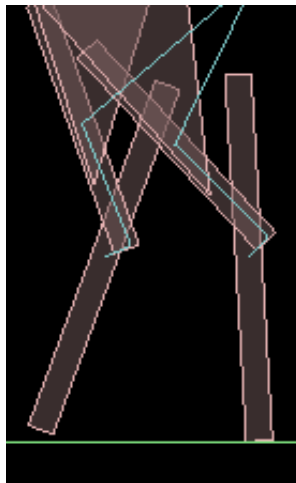


Figure 4: The image showing Distance joints as muscles

3.3 The Tail

This part is cheetah's tail which consists of multiple small blocks joined together with revolute joints. The revolute joint in the tail is constrained with the the angle constraints . Total sum of constraint in the allows it to move totally while maintaining its initial shape. Since tail is the highly important in maintaining the body balance its weight is accurately set so that it gives proper motion to body

Each of the block itself is made of two rectangular blocks joined together

Due to the tail the angular mometum of the cheetah's body is balanced

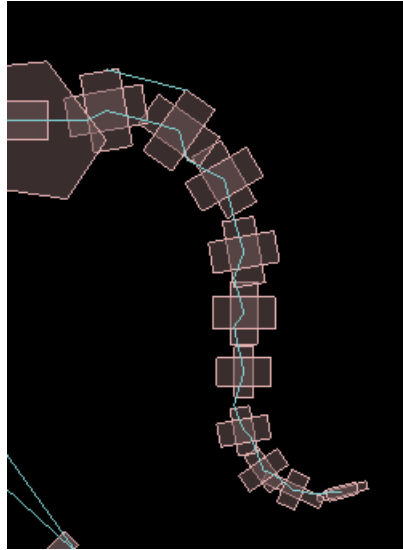


Figure 5: The image of tail consisting of multiple joints

3.4 Head and the Neck

The head of the cheetah is a normal body mady of a convex polygon. While the neck is far more interesting. Thr neck consists of three blockes joined and packed closely with a revolute joint. This helps in the swingning motion of the neck of the cheetah.

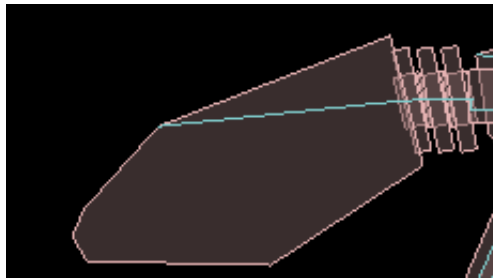


Figure 6: The neck and head of cheetah

3.5 Main body of cheetah

The main body of cheetah consists of a wide rectangular box to which all other parts of cheetah is joined. It acts a support for the whole body and helps in controlling the mwalking and jumping mechanism of cheetah.

The walking mechanism is at the center of this body and it controls whole of the walking of the cheetah

4 What makes the Simulation Interesting

The cheetah is the fastest land animal, achieving a top speed of about 70 mph (113 kph) [1] This is the main motivation toward taking up of this project. But owing the the restrictions of Box2D we tried to stimulate some of the parts of Cheetah's body and balancing and other details. Making a prototype of Living Organism which is the most complex machines makes our simulation interesting

While running the simulation you can see the swift swinging motion of tail, the motion of the head. Make the cheetah jum high and then land it to the ground. You can see the detailing of the landing and balancing. Such things also make the simulation interesting

5 Profiling report of the code

Using the profiling technique used in Lab09 we profiled the simulation of cheetah to find out the results

We have used perf profiler to make the profile data.

No of iteration for data : 10000

5.1 Call Graphs

A call graph is a directed graph that represents calling relationships between subroutines in a computer program. Specifically, each node represents a procedure and each edge indicates that procedure f calls procedure g.

Generation of call graph using perf and the gprof2dot.py

```
perf record -g -- ./mybins/cs296_17_exe
perf report > g17_release_prof.dat
perf script | python gprof2dot.py -f perf | dot -Tpng -o release.png
```

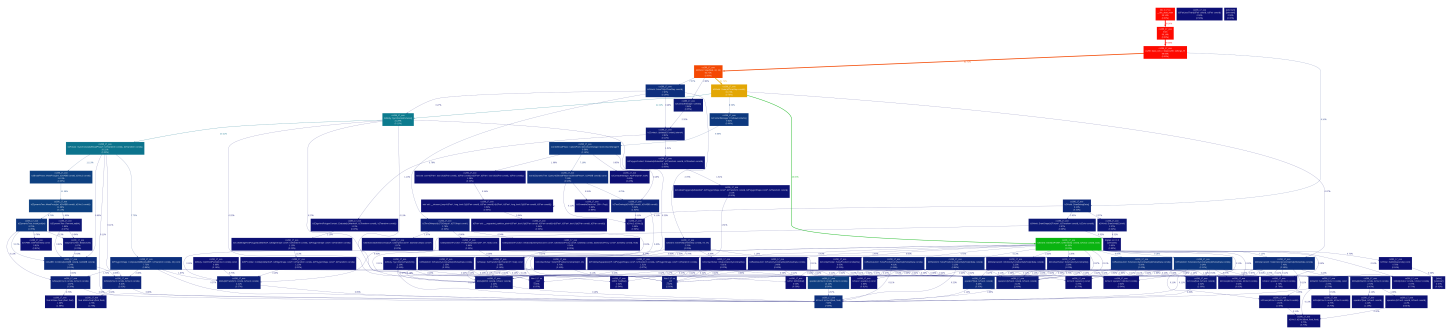


Figure 7: Debug Call Graph

Observation from Debug Call Graph

List of Functions which consume most time in the call of their parents

1. b2World::step() 92.00 %
2. b2World::solve() 49.00 %
3. b2RevoluteJoint::SolveVelocityConstraints() 8.00 %
4. b2DistanceJoint::SolveVelocityConstraints() 9.24 %
5. b2weldJoint::SolveVelocityConstraints() 9.24 %
6. b2Fixture::SynchronizeFixture() 21.29 %
7. b2Fixture::Synchronize() 20.00 %
8. b2BroadPhase::MoveProxy() 11.00 %
9. b2BroadPhase::UpdatePairs(b2ConstactManager) 9.56 %
10. b2DynamicTree::query() 7.24 %



Figure 8: Release Call Graphs

Observation from Release Call Graph

List of Functions which consume most time

1. b2RevoluteJoint::SolveVelocityConstraints() 4.76 %
2. b2DistanceJoint::SolveVelocityConstraints() 4.48 %
3. b2weldJoint::SolveVelocityConstraints() 3.92 %
4. solve() 5.20 %
5. b2Timer::GetMilliseconds() 3.86 %
6. b2DynamicTree::Query() 6.74 %
7. b2Fixture::synchronize() 0.80 %

5.2 Analysis of above Call Graphs

General Analysis :

Debug Mode :

As we can see from the callgraph and the most time consuming functions , debug mode calls much more functions as compared to the call graph of release mode. Most of the time is consumed by the calculating functions which are solve , solveVelocityConstraints etc consumes most of the time as compared to the Drawshape function and drawDebug function

Release Mode :

As opposed to the times in Debug mode , The Drawshape function take the most time in the running of the function while the solving function functions like solveVelocityConstraints and solve function took time much lesser than that in debug mode. also the system library used for calculation took very less time

Special Analysis w.r.t The Project Code :

Clearly we can see that the functions related to synchronizing fixtures and collision detection such as - *SynchronizeFixture()* and *FindNew Manager* take more as compared to profiling done in earlier labs.

Also the Functions related to BroadPhase which manages the proxies (- representing the AABB's in broadPhase collision algorithm) and the creation and deletion of records w.r.t the overlap of proxies such as *MoveProxy()*, *UpdatePair()*, *InsertLeaf()*, *Query()* take much time as compared to previous profiling.

Moreover the functions (*SolveVelocityConstraints()*) related to solving the constraints equations for the joints - *Revolute*, *Weld* and *Distance* take more time in comparision.

5.3 Inference from the above Analysis

General Inference We can clearly see that the Debug mode is taking more time for all the functions in comparison to the release mode which does the optimization.

The function *solveVelocityConstraint()* and *solve()* takes more time in both the modes. The step function is taking a lot of time in the debug mode as it is the function which is called again and again.

The *DrawShape()* takes significant amount of time as this function is called each time the *step()* function is called. This function is responsible for updating the position of the various objects on the frame after the solving of all velocity and position constraint equation is solved. So it is natural for this function to take more time.

The library *_mcount* takes more time in the debug mode but negligible amount in release mode and hence no need to optimize thus library.

Special Inference wrt Project Objects We can clearly see that the program spends a lot of time in handling the different fixtures. The functions which are responsible for obtaining the overlap of fixture proxies - (the AABB's representation) and storage of these in a dynamic tree consumes lot of time. The behaviour is justified since we have a lot of fixtures which are overlapping and many of them are using collision filtering using *filter.groupIndex* . Thus the box2d engine has to spend a lot time to decide which of them to collide and which not. Moreover we have 2 bodies - **tail**, **neck** which is made of lot of smaller bodies joined together and spaced very close to each other . Thus the contact equations are to be solved more often for the fixtures of these bodies. Also due to lot of smaller bodies connected to form larger bodies we get a lot of joints of all kinds. Hence the constant equations for these joints are solved more often thus explaining the cause that *SolveVelocityConstraints()* of these joints take a large time.

5.4 Optimizations Required

Referring to the above Inference we see that we would have to decrease the number of pieces that makes the tails and the neck. But such a process would result in destroying the flexible movement of the tail and neck and thus changing the expected behaviour. So we changed the spacing and arrangement of the object in neck . Also for tail

the fact that we could initialize the same body with different fixtures was used to decrease the number of bodies actually making the tail .

For different fixtures that were using the groupIndex to do collision filtering we tried to exploit the fact we used for tail. But the mass distribution which maintained the overall balance of the cheetah on its legs and helped the crank system to make it move smoothly got unbalanced causing some unexplained behaviour . Hence we had to compromise in this optimization by designing the bodies so that balance is also maintained and the overlapping parts overlap as less as allowed so that the movement is smooth.

6 Analysis Report for the plots my Matplotlib

This section contains the Reports , Observations and Inference from the timing diagram which are obtained from the data and the Plots generated using matplotlib of the project simulation.

No of Iteration for which the the iteration was run for 1 to 500 and 50 rerun for each iteration

6.1 Observation from Plot1

The following inference were drawn from the Plot1 diagram which was Plot of Average Step time and Average loop time:

1. The Average loop time is greater than the Average Step Time but is almost equal
2. The average time for the decreases as the no of iteration increases

Explanation: Since for the first few iterations the world is being set and all the objects in a state of movement to get settled to there stable state and also the no of collisions taking place are too high, the time taken by the Loop in first few Iteration are High. But as the loop progresses the animation gradually takes place without much calculation or updates and the updates are very few so the time for the further loops decreases. Due to this the average time for more iterations decreases as compared to lesser no of Iterations

Since the step time is the of all times taken by the loop while the loop time is the total time taken by the loop in addition to the Time taken by system processes so the loop time is slightly greater than the step time. And when the same was generated while the system was overloaded with programs, This gap was still greater.

6.2 Observation from Plot2

The following inference were drawn from the Plot2 diagram which was Plot of Step time , Collision time , Velocity update time , Position update time and sum of last 3 averaged:

1. The step time is greater than the sum of Collision time , velocity update time and position update time
2. All of these times decreases as the no of iteration increases

Explanation: The step time is the time taken by the loop which is running the Animation. But the other three times are exactly what they are meant to be. But since the step time is the time taken by all of the things within a loop , So there are other processes which will also be called within the for loop and they will take time increasing the step time

The explanation for initial times to be greater as the explanation in of Plot1

Figure 9: Plot1

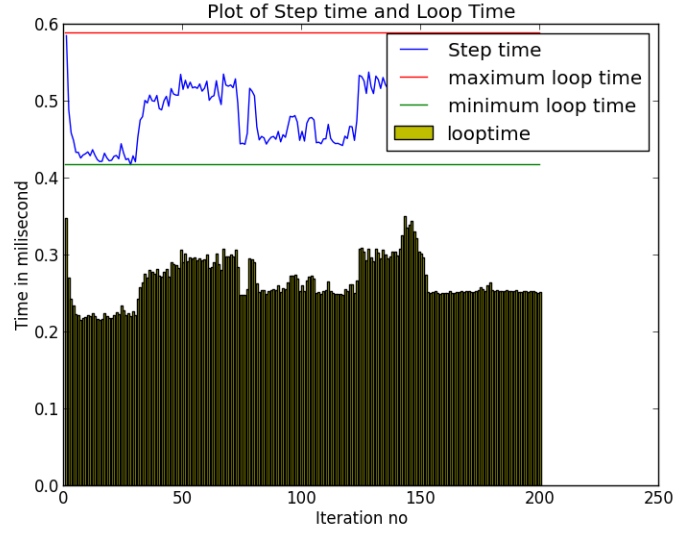
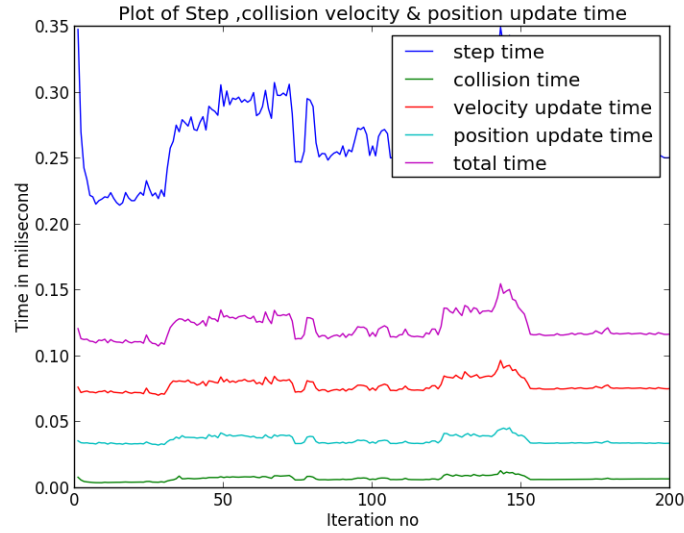


Figure 10: Plot2



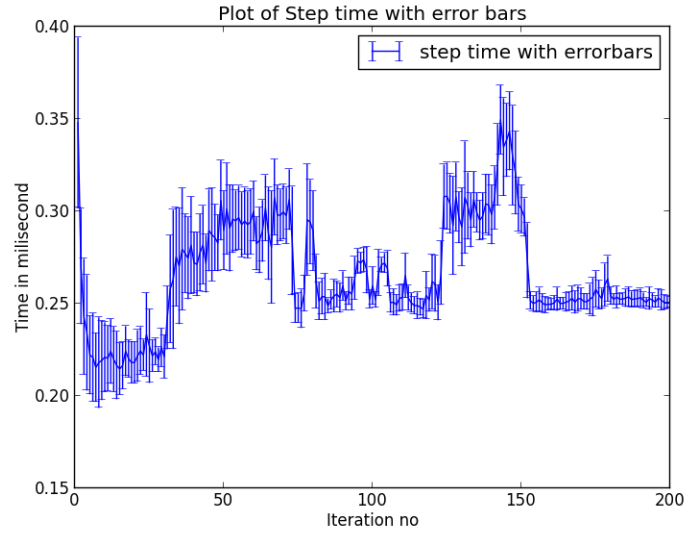
6.3 Observation from Plot3

The plot3 was the plot of averaged step time along with error bars calculated with standard deviation

It is observed that the error bars for the average stem time for lower no of iterations was too high as compared to the error bar for higher no of iterations

This might be because of the reasons explained in Plot1. Since the setting of world can take place differently for only one iteration because of setting up of worlds and setting up of initial position of objects so the error bar is too high. But when the no of iteration is high, This total time sums up to almost be equal. So the standard deviation is low making the error bar to be small.

Figure 11: Plot3

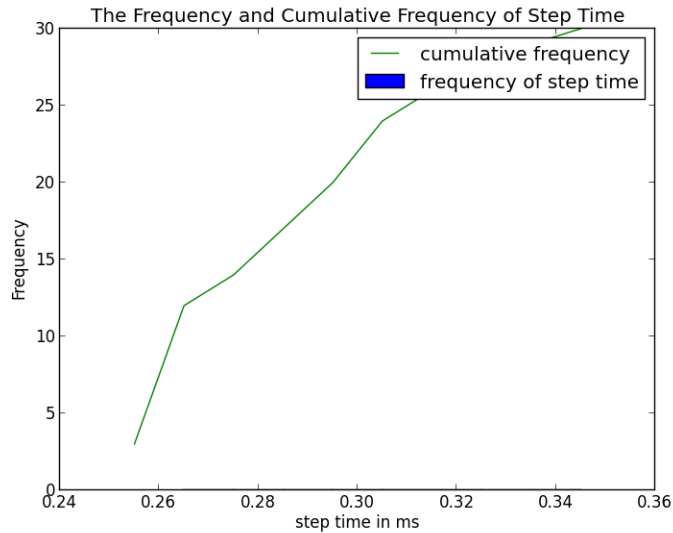


6.4 Observation from Plot4

When the step time gets averaged up it keeps fluctuating around the the average value which comes out to be 0.94 second and the rest of the times are around it.

But there is an Outlier time which denotes the time at the start of the loop. Since At the beggining the times are higher so they contribute for a frequency of a higher time

Figure 12: Plot4



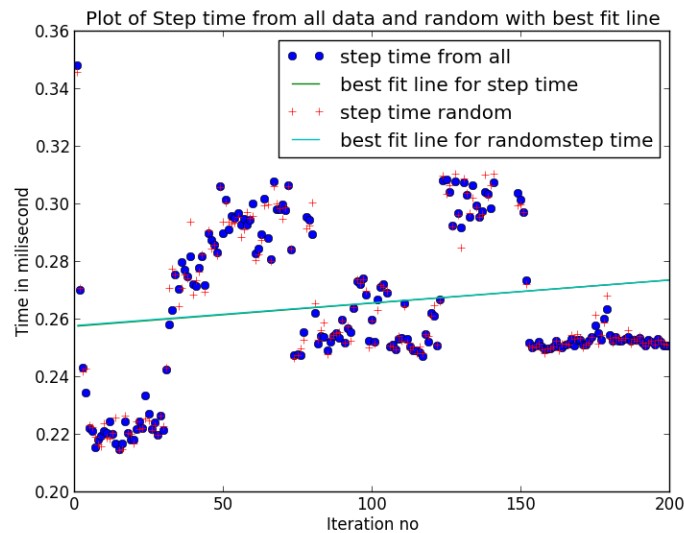
6.5 Observation from Plot5

Plot 5 was the plot for the best fit line for the step time averaged over all data and step time averaged over some random data

1. The points for the averaged time of all data and random data are almost equal
2. The best fit line comes to be almost same for both of them

Explanation: Since the time for a particular iteration is almost the same so it doesn't matter much if the points are averaged over all data or the points are averaged over only few random data. Due to this the points come to be nearly the same and also the best fit line is nearly the same

Figure 13: Plot5



7 Conclusion

Project The cheetah simulation along with finishing of the project was very interesting and of great learning experience. We learnt various aspects of complex machine simulation, different walking mechanism used by cheetah, Implementation in box2D with a simpler walking mechanism, simulating body parts like tail and neck of the cheetah and other things. The amount of learning experience from this project was too great.

CS296 course This CS296 was overall a great learning experience of Box2D and also other software development techniques. In the CS296 course as a whole we learnt a lot and we are very thankful to the instructor Prof Parag Chaudhary [2]. The software development techniques learnt and the other very useful softwares learnt has increased *geekiness* in us. And we will continue our coding aspect followed up from this course.

References

- [1] Defenders of Wildlife Blue Lion, 2014.
- [2] Prof Parag Chaudhary. Official site, 2014.

[3] Wikipedia. Theo jansen mechanism, 2014. [Accessed 7-April-2014].