NDOT Research Report

Report No. 297-10-803

Driving Simulator Project

May 2014

Nevada Department of Transportation
1263 South Stewart Street
Carson City, NV 89712

Disclaimer

This work was sponsored by the Nevada Department of Transportation. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of Nevada at the time of publication. This report does not constitute a standard, specification, or regulation.

UNLV

UNIVERSITY OF NEVADA LAS VEGAS

Transportation Research Center

Driving Simulator Project

Author:

Romesh Khaddar

romesh.khaddar@gmail.com, 702-302-6553

Student M.S. Electrical Engineering, and M.S. Mathematics

University of Nevada, Las Vegas

Principal Investigator:

Dr. Pushkin Kachroo, P.E.

http://faculty.unlv.edu/pushkin, pushkin@unlv.edu, 540-588-3142

Director, Transportation Research Center

Harry Reid Center for Environmental Studies

University of Nevada, Las Vegas

Prepared for:

Nevada Department of Transportation

1263 South Stewart Street, Carson City, NV 89712

May 27, 2014

TABLE OF CONTENTS

LIST OF FIGURES v

LIST OF TABLES vi

EXECUTIVE SUMMARY 1

1 INTRODUCTION TO DRVING SIMULATOR 3

2 PUBLIC OUTREACH 13

3 INSTALLATION GUIDE - SIMCRAFT 15

4 INSTALLATION GUIDE - STISIM DRIVE 16

5 ROADWAY NETWORK GEOMETRY 25

6 HYBRID SIMULATION MODEL 28

7 USER-DRIVEN VEHICLE DYNAMICS MODEL 33

8 PEDESTRIAN, BICYCLE AND MOTORCYCLE SIMULATOR 35

9 VIRTUAL REALITY ENVIRONMENT 38

10 DATA COLLECTION 42

11 INTRODUCTION & LITERATURE SURVEY 43

12 SYSTEM ARCHITECTURE 47

13 ROADWAY NETWORK GEOMETRY 52

14 HYBRID SIMULATION MODEL 55

15 USER-DRIVEN VEHICLE DYNAMICS MODEL 60

16 PEDESTRIAN, BICYCLE AND MOTORCYCLE SIMULATOR 62

17 VIRTUAL REALITY ENVIRONMENT 65

18 DATA COLLECTION 69

19 INTRODUCTION TO PEDESTRIAN SIMULATOR 70

20 SYSTEM ABSTRACTION 75
20.1 Abstraction of Systems . . . 75
20.2 Mathematical Preliminaries . . . 77
20.3 Abstracting Maps . . . 78
20.4 Abstraction of Dynamical Systems . . . 80
20.5 Abstractions of Control Systems . . . 81

21 THE HUMAN WALK 85
21.1 Bio-Mechanics of Human Movement . . . 85
21.1.1 Movement Analysis . . . 86
21.2 Gait . . . 88
21.3 Kinematics Analysis of Gait . . . 90
21.4 Simulating an Actual Human Walk . . . 91
21.5 Abstraction of Human Walk . . . 92
21.5.1 Definition of on the spot walk . . . 93
21.5.2 Analysis of on the spot walk . . . 95
21.6 Model of Virtual Entity in 3D World . . . 95
21.7 Abstracting Map . . . 98
21.8 Conclusion . . . 99

22 CAPTURING HUMAN WALK 100
22.1 Simulated Human Walk . . . 100
22.2 Mapping actual to simulated walk . . . 101
22.3 Extracted Parameters . . . 101
22.4 Construction . . . 102
22.4.1 Architecture . . . 103
22.4.2 Hardware Construction . . . 107
22.4.3 Software Design . . . 109
22.5 Limitations . . . 111
22.6 Testing and Results . . . 112

23 IMPLEMENTATION USING NATURAL INTERACTION 113
23.1 What is natural interaction . . . 113
23.2 Why is natural interaction required . . . 114
23.3 Available options . . . 114
23.3.1 Why MS Kinect . . . 115
23.4 Setting up kinect . . . 116
23.5 Kinect Interaction Space . . . 118
23.6 Kinect for Windows Architecture . . . 120
23.7 Programming Models . . . 122

23.7.1 Polling Model . . . 122
23.7.2 Event Model . . . 123
23.8 Software architecture . . . 123
23.8.1 Skeleton identification . . . 123
23.8.2 Gesture definition . . . 125
23.8.3 Feature extraction . . . 125
23.8.4 Gait parameters conversion . . . 126
23.8.5 Step event generation . . . 126
23.8.6 Global parameters conversion . . . 127
23.9 Conclusions and limitations . . . 127

24 NAVIGATION ON 2D PLANE 128
24.1 Current Limitations . . . 128
24.2 Method 1 . . . 129
24.3 Method 2 . . . 131
24.4 Method 3 . . . 133
24.5 Selected Final Method . . . 134

25 FINAL IMPLEMENTATION 135
25.1 System Architecture . . . 135
25.1.1 Intitialization . . . 137
25.1.2 Calibration . . . 137
25.1.3 Get Current Gait Values . . . 138
25.1.4 Get Pedestrian Data Values . . . 138
25.2 Test Case Implementation . . . 138

26 DISTRACTED DRIVING 141
26.1 Introduction . . . 141

27 EXPERIMENTAL SETUP 145
27.1 Driving Simulator and Scenario Design . . . 145
27.2 Subject Selection . . . 148
27.3 Procedure . . . 148

28 ANALYSIS: DISTRACTED DRIVING 150
28.1 Data Recording . . . 150
28.2 Data Pre-processing . . . 150
28.3 Time Domain Analysis . . . 151
28.3.1 Standard Deviation analysis of LLP and SWA . . . 151
28.4 Frequency Domain Analysis . . . 157
28.5 Entropy Based Analysis . . . 162

LIST OF FIGURES

1.1 Simulator Hardware 4
1.2 Simulator Software 7

4.1 Run BuildXXXX.exe 18
4.2 CD ROM Window 20

5.1 Generated network on blender using open street map 27

9.1 Layer-architecture for virtual reality generation 41

12.1 System architecture of n-MIPEDS. The images of bicycle and motorcycle simulators are indicative only. Copyright: Honda 48
12.2 Data flow diagram. 49

13.1 Generated network on blender using open street map 54

17.1 Layer-architecture for virtual reality generation 68

19.1 Relationship between pedestrian and traffic environment 73

20.1 Relation between Systems and Abstractions 76
20.2 Mapping Between Spaces 80

21.1 A Single Step shown for the forward progression of body 86
21.2 Free Body diagram of a foot 87
21.3 Link Segment Model 88
21.4 On The Spot Walk 94
21.5 Mapping of walking phases 96
21.6 The rolling disc 98

22.1 Flowchart of complete system 103
22.2 Information Flow 104
22.3 Flowchart of hardware system 105
22.4 Flowchart of software system 106
22.5 Placement of sensors 108
22.6 Calibration Position 111

23.1 Out of the box 116
23.2 Power Cord 117

23.3 Mounting the Kinect Sensor 117
23.4 Kinect Interaction Space [14] 119
23.5 Kinect Architecture [15] 120
23.6 Kinect SDK Components Architecture [15] 121
23.7 Pedestrian Data Extraction Architecture 124

24.1 Perpendicular kinect sensors scenario 1 130
24.2 Perpendicular kinect sensors scenario 2 131
24.3 Geometry of relating body constants 132

25.1 Complete Architecture of Pedestrian Simulator API 136
25.2 User in front of Kinect Sensor 139
25.3 User in changing heading 140
25.4 User stepping and increasing distance travelled 140

26.1 Accident contributing factors leading to inattention 141

27.1 Simcraft Driving Simulator 146
27.2 Fatal Vision® Goggles 148

28.1 $\mu_{\sigma_{SWA}}$ with varying σ_{LLP} Ranges 153
28.2 $\mu_{\sigma_{LLP}}$ with varying σ_{SWA} Ranges 154
28.3 $\mu_{\sigma_{LLP}}$ with varying σ_{SOV} Ranges 155
28.4 $\mu_{\sigma_{SWA}}$ with varying σ_{SOV} Ranges 156
28.5 Time variation of Lateral Lane Position (LLP) of Subject 1(in feet) 158
28.6 Single Sided Spectrum of Lateral Lane Position (LLP) of Subject 1 . 159
28.7 Time variation of Steering Wheel Angle (SWA) of Subject 1(in radian) 160
28.8 Single Sided Spectrum of Steering Wheel Angle (SWA) of Subject 1 . 160
28.9 Time variation of Speed of Vehicle (SOV) of Subject 1 161
28.10Single Sided Spectrum of Speed of Vehicle (SOV) of Subject 1 161
28.11Entropy variation of LLP for Subject 1 163
28.12Entropy variation of SWA for Subject 1 164
28.13Entropy variation of SOV for Subject 1 165

LIST OF TABLES

22.1 Gesture Parameter mapping 101

23.1 Comparision Table of Available Products 115

EXECUTIVE SUMMARY

According to Traveler opinion and perception survey of 2005, 107.4 million Americans use walking as regular mode of travel, which amounts to 51% of American population. In 2009, 4092 pedestrian fatalities have been reported nationwide with a fatality rate of 1.33 which totals 59,000 crashes. Also, pedestrians are over represented in crash data by accounting more than 12% of fatalities but on 10.9% of trips. This makes a perfect case for understanding the causes behind such statistics, calling for a continuous research on pedestrians walking behavior and their interactions with surroundings.

Current research in pedestrian simulation focuses on surveys and mathematical simulation models such as macroscopic and microscopic dynamic models involves autonomous entities. The surveys represent the perception of individual while mathematical simulation severely limits the capacity to capture effect of human factors in the understanding of pedestrian interactions. Complicated psychological models are used to a certain extent for understanding of such problems but are incapable to estimate the diversity of human behavior. To capture tendencies of people, they need to be a part of research, under a safe and controlled environment.

In this thesis, an attempt has been made to develop a module which can be used to track human walk gesture and map it to actual human walk. Then, this module could

be implemented in a system aimed to understand pedestrian behavior. Following are the accomplishments of this thesis.

- Built an API to use with software interface to capture human motion
  - Explored arduino based wearable interface to capture human motion.
  - Explored Kinect based video interface to capture human motion.
  - Defined gestures and identified configurations for least difficult setup and calibration process.
  - Wrote the software interface for a Kinect based system (video interface).
- Built a mathematical framework for abstracted dynamical system, for the purpose of pedestrian interface in simulation engine.
  - Obtained mathematical model for human walk.
  - Obtained conversion to non-holonomical system for human walk.
  - Programmed the mathematical model into the API.

Eventually this is expected to contribute towards state-of-the-art researches which aim at understanding pedestrian dynamics in transportation safety and planning. The module described is expected to work real-time as a separate entity.

CHAPTER 1

INTRODUCTION TO DRVING SIMULATOR

Traffic behavior analysis is an important aspect for traffic safety research. To achieve this a consistent method of analysis is required. Driving behavior surveys are a proven method to get an insight into a typical driving behavior. This allows for understanding the upcoming trends in driving behavior. A very good example is the introduction of mobile phones into daily lives of every individual. This gave rise to an increase in texting while driving. We at TRC are proud that we were able to use our driving simulator to put this point accross in various campaigns around Las Vegas, and thus playing a significant role in the 'No Texting While Driving' law in Nevada. In the following sections a more detailed description of the setup is given.

Description of Hardware

The transportation research center(TRC) of UNLV wants to create a unique driving simulator which could be taken to public and aims to provide driving do's and dont's. At the same time, TRC also wanted to keep the realism of driving for the users. Therefore, motion feedback was an important component of the unit. Based on these two primary requirements TRC narrowed down the driving simulation hardware to the one offered by Simcraft Technologies.

The simcraft unit has multiple capabilities. It comes with a full-fledged 3DoF motion simulator with fit in Recarro seat. Moreover the system comes with wheels which allows it to be easily movable. This setup can be dis assembled in four parts enabling it for easier assembly and disassembly aiding transportation of the unit across the rooms.

The cage of the simulator is made of aluminnium which enables it to be light and

Figure 1.1: Simulator Hardware

strong. The display setup is composed of three monitors aimed to provide a 120 degree fov. Associated steering joystick and pedals are equipped with motion feedback.

These both together provide a more immersive environment for the user.

The 3DoF is achieved by 3 cylindrical pistons with an analog controller box. Each piston is provided a seperate controller due to current requirement and easier troubleshooting. The complete setup is controlled via usb ports on the provided computer.

The SimCraft Control Panel, CraftCon, allows you to access the SimCraft integrations or modules that control the interaction between your SimCraft motion system and the game/computer simulation you are using. A module is represented as a list item in the CraftCon window containing the game/sim name and the game/sims icon. Within CraftCon, each game/simulation has a corresponding module, and the modules are all independently installed and managed. CraftCon allows you to control module activity and the configuration of the motion system with module specific settings for each game/simulation.

Craftware is an Application Programming Interface (API) that provides the interface for any SimRacing or FlightSim title to drive a SimCraft motion simulator creating a synchronized, realistic motion element. The API provides an extracted method of positional control for SimCraft motion platform products and it does so by receiving available object state variables (physics or telemetry) from various physics based systems and translating the real-time data into positional output to drive the motion of the simulator.

Description of Software

STISIM Drive is used to create driving scenarios that mimic real life driving situations but do so in a safe yet realistic environment. STISIM Drive is a personal computer based, interactive driving simulator that allows the driver to control all aspects of driving including the vehicles speed and steering. One can control what the driver sees and when they see it. Unlike most simulation programs that come with a fixed database that you can not easily manipulate without extensive CAD experience, STISIM Drive allows you to change most aspects of the driving scene with simple ASCII text commands. This frees up time for designing and building various roadway situations instead of just designing and building a roadway.

STISIM Drive is a product of over three decades of research by Systems Technology, Inc. (STI) on low cost techniques for creating laboratory tasks relevant to the psychomotor and cognitive demands of real world driving. Extensive past research on vehicle dynamics and driver control behavior, driver decision making and divided attention behavior and response to traffic control devices has been applied to the creation of control tasks and cognitive scenarios typical of real world driving. A combination of vehicle dynamics characteristics and compensation for CGI (Computer Generated Imagery) transport delays have been employed to create an appropriate stimulus-response relationship between steering inputs and visual display motions. The composite vehicle dynamics/compensation characteristics have been carefully

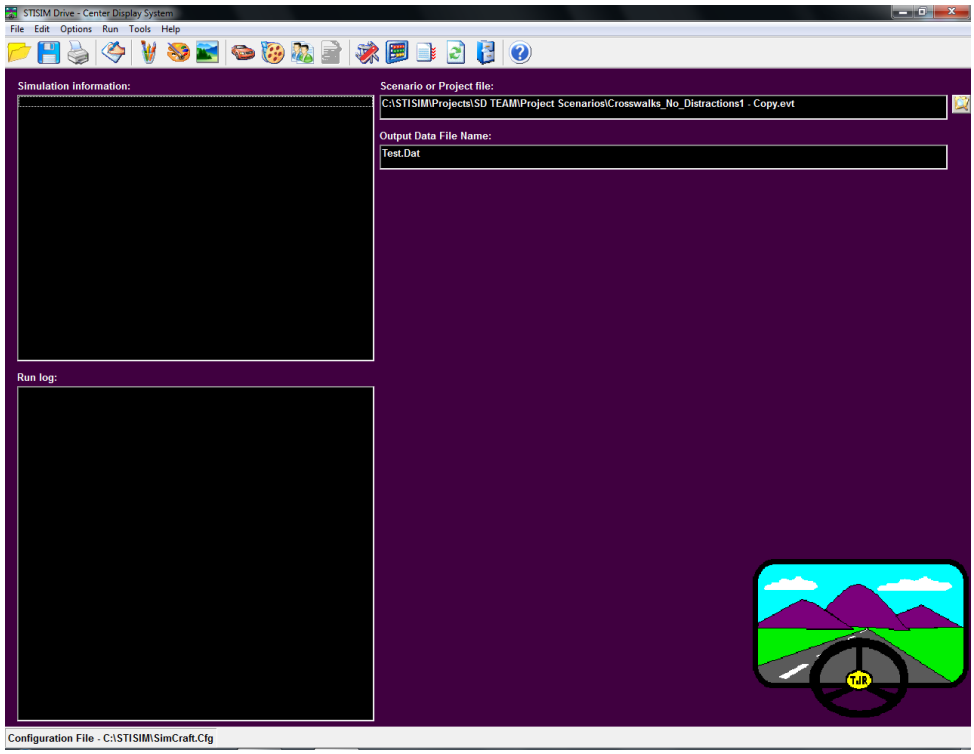

Figure 1.2: Simulator Software

integrated so that steering sensitivity is appropriate over the full range from rest to top speed, and is not sluggish or oscillatory as is the case with many CGI based driving simulations. Driver relevant vehicle dynamics attributes are easily specified in a parameter file along with other simulator setup characteristics.

Driving tasks and scenarios are easily specified with simple commands listed in an events file. A simple scenario definition language (SDL) has been developed to minimize the effort required to specify experimental designs. The SDL frees the user from having to program visual data bases as is the case with most CGI based simulators. The SDL also simplifies the specification of scenario attributes that relate to

driver psychomotor, divided attention and cognitive behavior, and permits the definition of road curvature, elevation, intersections, signal timing, interactive traffic, etc.

There are an infinite number of ways that STISIM Drive can be used. Although we have tried to make the program and documentation as simple to use as possible, we cant possibly foresee all of the application that it will be used for. Subsequently, the documentation is written as generic as possible so that we can provide you with general ideas on how to use the program and not how to solve specific problems. Therefore, we highly recommend that you become familiar with the simulators configuration options and the SDL before you begin creating your driving environment. We understand that this will be a large undertaking on your part, but for the best possible results in the shortest amount of time, you will need to know what the simulation is capable of doing. In general, there are four basic components that make up the STISIM Drive simulator, the graphics environment, the driver controls, the SDL that controls the scenarios, and the STISIM Drive software that ties it all together.

The graphics environment includes the graphics card that generates the images, the display system that displays the images, and the models that are used so that images can be displayed. Supported graphics cards are discussed in the hardware section of the help system, but in general STISIM Drive requires a specialized graphics card that is capable of running accelerated OpenGL based graphics. Any graphics display device that can handle a standard VGA graphics input can be used for the

actual display device. These include computer monitors, head mounted display devices, projectors and even televisions if a scan converter is used (however resolution is lost with this type of arrangement). Finally, and most importantly, the images that are displayed on the screen are simply rendered versions of polygonal models that have been designed and built, then loaded into the simulator for display. Almost every image that you see is created using a combination of polygons, texture maps and shaded colors. A separate graphics package such as 3D Studio or NuGraph is required to build the models that are displayed using the simulator. STISIM Drive comes with a number of models that you can use directly with our SDL, but these may not be sufficient for your application. If this is the case, you will need to build some models on your own. STISIM Drive currently has numerous vehicle models, buildings, pedestrians, roadway markers, barriers and road signs.

Before one can actually begin setting up and running a simulation scenario, you must first become familiar with STISIM Drives capabilities. This next section lists some of the most interesting tasks that the simulator will allow you to perform, but to really understand the power of STISIM Drive, you will need to learn as much as you can about the simulator's configuration and the SDL. Additionally, you should review the sample events files that are installed during program setup. These can be found in the Projects directory under the STISIM Drive main directory.

The following is a list of some of the simulators general capabilities:

- Wide screen (3 computer systems required), giving the driver a 135 degree field of view
- Advanced vehicle dynamics (option sold separately, sounds like a commercial)
- Interactive vehicles in all lanes of traffic
- Traffic Control Devices (barrels, Jersey barriers, traffic signals, traffic signs, etc.)
- Buildings, benches, billboards and other rectangular objects can be created with the 3-D block event
- Animated pedestrians and cross traffic can be used to force drivers to maneuver
- Left and right hand turns at specified intersections
- Divided attention symbols located at the right and left of the display
- True rear view mirrors that display the roadway scene behind the driver so that they can see and react to approaching traffic
- Several speedometer and car cab options
- Average and RMS response measures collected at specific defined intervals
- Auditory feedback including user defined files and sound effects (siren, car crash, tire screech)

- Game controller, Analog, or Optical Encoder Interface support for driver controls and an active steering control system (not on all configurations) that feeds roadway feel back to the driver
- Configurable for driving from either side of the road
- English or Metric units can be used when specifying inputs and outputs
- "Autopilot mode" for scenario checkout
- High resolution graphics modes may be used to reduce dither and aliasing effects, and some display colors can be set manually
- Scenario parameters can be defined and saved in external files
- Various utility programs and examples are included to help you
- Configurable for input and output of digital signals from external devices

Before any simulation can be run, the hardware for your system must be installed and all of the peripheral devices such as monitors and controllers must be connected to their respective interface boards. In order to make these connections, you must become familiar with the hardware that you will be using, and know which hardware device connects to which controller board. If you ordered a complete system including all of the hardware, then the interface cables and computer boards should be clearly marked so that all you will need to do is plug each cable into the corresponding interface board in the computer. If all you ordered was software and a graphics board,

you will need to determine what connections need to be made. In any event, you should look at the section on hardware and setup.

During the installation process several shortcuts to STISIM Drive were installed on your computer system. One is installed directly in the start menu that appears when you click on the Windows Start button. The second, appears on your desktop so that you will have easy access by double clicking on a desktop icon. Either of these options can be used to activate the program, or you can use Windows Explorer to point at the STISIM Drive folder and run the software from there. When the program is activated, the initial STISIM Drive screen will be displayed for several seconds and then the program's main window will be displayed. You are now ready to start running simulations.

One aspect of the simulator that everyone looks at is speed. Having a constant update rate (having the individual frames that are displayed to the user occur at the same timing rate) is very important because it creates a more realistic environment, reduces the chance of simulator sickness and provides for better data collection. However, this is one of the more difficult things to control in an interactive simulator. This is generally due to the number and complexity of objects appearing in the roadway scene.

CHAPTER 2

PUBLIC OUTREACH

TRC has been involved in various public outreach programs using the driving simulator. The primary objective has always been to encourage new drivers and educate older drivers about the various issues of transportation safety. Having a tool like Driving Simulator certainly helps the cause to put accross the point. It acts as means to establish a live demo of many practices people do while driving. This eventually aims towards achieving Zero Fatality goal of Nevada. The list of completed programs is as follows:

- Driving Simulator taken to T-Bird Bar and Restaurant in Henderson, where it was shown in public how much effect can they see in their normal driving behavior after a couple of drinks. It was a 2 day event.
- Driving Simulator was taken to NDoT Conference at Paris Casino, where it was used to showcase the usage and how people respond to drving simulator when they are drunk. This was a 3 day event.
- Driving Simulator was an integral part of the "No texting while driving" campaign. It was used as the means of various demonstrations for Media.

- Driving simulator was taken around to schools in Las Vegas area as an initiative between Safe Community Partnership and UNLV TRC, during the no texting while driving week for the purpose of Drivers Ed program established in the schools to showcase why is using a phone while driving a wrong practice.

- Driving simulator has also been a part of excursions for STEM students at UNLV every year. During such events the research side of transportation and transportation safety were showcased to encourage students to choose STEM as their future field of study.

- Driving simulator was part of 2nd Annual Science fair of Las Vegas. At this event again "No Texting While Driving" was campaigned for.

- Multiple Safe Community Partnership events were held in association with TRC.

CHAPTER 3

INSTALLATION GUIDE - SIMCRAFT

Although CraftCon and the game/sim modules are two different pieces of software, they rely on each other in order for the SimCraft motion system to work correctly. Obviously, they are separate from the game/simulation software. However, the game/simulation software needs both CraftCon and the correct module for the game/sim software to drive the motion chassis.

For this reason, the module and CraftCon installer are bundled together to assure maximum compatibility between the two products. If you only have one module, you can always download the latest module installer and it will contain the latest CraftCon version. If you have multiple modules, updating one module will update the CraftCon version for all of the modules installed.

It is not recommended to upgrade CraftCon software outside of the version provided by the module, or use another module to manually update CraftCon. Each CraftCon/module combination is designed to work seamlessly with the SimCraft motion system; any change to either the module or CraftCon outside of the installer could result in improper operation of the motion system resulting in harm to the user or bystanders or damage to the system.

CHAPTER 4

INSTALLATION GUIDE - STISIM DRIVE

Before installing the STISIM Drive software, you should always make sure that your hardware is configured correctly for the type of system that you have. The installation procedure will request a model number from you and also some information about your graphics acceleration board and the type of controller card that is being used (this is not required for all models). Based on this information, default parameters are set such that you should be able to start the simulator and run any of the example scenarios that are delivered with the simulator. If your hardware is not configured correctly, then when you try and run the simulator you will have trouble.

In general the software comes pre-loaded and setup on the computer, however for the model 100 and demonstration systems where just a disk is delivered or the software is downloaded, you will need to install the software yourself. In the unfortunate event of a catastrophic system failure, you may also need to reinstall the system yourself. With these cases especially, you should always make sure that all of the hardware requirements are handled before installing the software.

Software Installation:

It is very important that before you go and install any of the software that you have correctly setup and configured your computer system. If you have not already done so, review the hardware configuration guide before continuing with the software installation.

STISIM Drive uses a standard Windows installation routine that is similar to most other Windows software packages. There are several ways to launch the installation process and generally the procedure is dependent on the way you obtained the software:

Download:

Generally a download is used for either updating the software, or for demonstrations, for whatever reason if you downloaded the software from our web site (http://www.systemstech.com), select the Downloads option and then select the appropriate files (generally Buildxxxxx.EXE) then the installation is a 2-step process:

1. Run the self extracting Buildxxxxx.Exe program. This will extract a group of files that will then be used to perform the actual installation. If you run the Buildxxxxx.Exe program, it will request the name of the folder where the extracted files will be placed. Enter the name of a temporary folder and then click on the Unzip button:

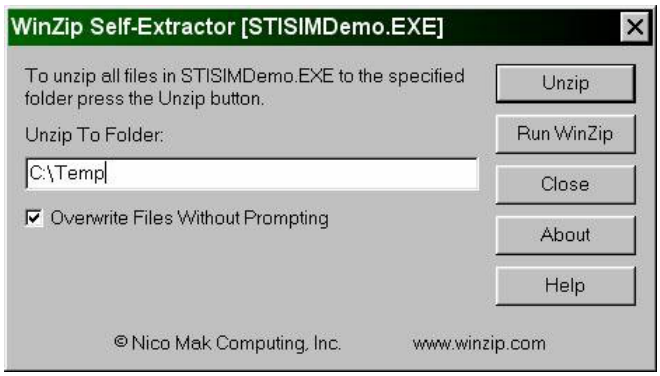

Figure 4.1: Run BuildXXXX.exe

2. In the temporary folder that the files were extracted into, run the Setup.Exe program and follow the prompts.

Note:

If you have a wide field of view system, you will also need to download the file SideBuildxxxxx.Exe for use on the side computers. If you have an Open Module system, you will need to download the OMBuildxxxxx.Exe file also.

If you will be installing an Advanced Dynamics system you will need to download the following 2 files:

CoreInstallv7xxxx.exe

VDANLDriveInstallv7xxxx.exe

The file CoreInstallv7xxxx.exe contains the core dynamics kernel files and all of

the help files and is required for both applications. VDANLDriveInstallv7xxxx.exe contains the installation files for the real-time version of VDANL (VDANL Drive) that is used with STIs real-time driver-in-the-loop driving simulator STISIM Drive.

The first step is to download the 2 files. When downloading, simply save the files to a temporary folder so that they can be used later. The second step is to create folders so that you can keep track of the different installations. Using Windows Explorer, create a folder named VDANL_Install in the root folder (C:). Now open this folder and create the following folders:

VDANL_Core

VDANL_Drive

Now copy each of the downloaded files into the appropriate sub folder that was just created and run the file. This will extract the individual installtion files that will be discussed later.

CD-ROM:

If you are installing from a CD-ROM, then there are a couple of options for installing the software. When you place the disk in the CD-ROM drive and close it, a dialog window similar to the following should be displayed:

Depending on the type of system that you have, you can select what you would like to do by using the mouse to highlight the desired option (active options are displayed

Figure 4.2: CD ROM Window

in yellow) and then clicking on the option to initiate the installation. If this option does not appear after placing the CD in the drive, then you can run the installation by finding the appropriate folder (Center System, Dynamics System, Open Module, or Side System) and running the appropriate Setup.Exe file from the folder.

STISIM Drive has an installation procedure that should always be used when installing the software. This is required because of the number of DLL files and device drivers that STISIM Drive requires in order to run. If all of the files are not located in the proper folders STISIM Drive will crash and tracking the reason for the crash can become extremely time consuming. Because of this, it is not recommended that you

try to do a direct copy from one computer to another, instead, use the installation disk provided. The installation itself is pretty easy. Log onto your computer with the account where STISIM Drive will be used (make sure this account has administrative privileges) and simply execute the installation from the installation disk and follow the prompts that are presented to you. In general you will always choose the "Next" option. Once the program has been installed, you will be asked if you want to restart the computer. Because several new device drivers have been installed it is always best to restart the computer. Most of the files for STISIM Drive are also included on the installation disk in a directory named Install Data. This allows you to copy individual files onto your system if anything happens to the original installation files, without having to re-install the entire program.

During the installation process several shortcuts to STISIM Drive will be installed on your computer system. The first will be installed directly in the start menu that appears when you click on Windows Start button. The second, will be setup on your desktop so that you will have easy access by double clicking on a desktop icon. Finally, the installation will setup a STISIM Drive folder that contains a shortcut to the STISIM Drive program as well as other programs that were installed during the setup process. Any of these options can be used to activate the program.

Uninstalling the STISIM Drive program is also fairly straight forward. Simply go to Windows Control Panel and use the Add/Remove Programs option and scroll

down until you find the STISIM Drive reference. Selecting it and clicking on the Change/Remove button will allow you to uninstall the software and its original files, including any new device drivers and DLLs that were installed and are exclusive to STISIM Drive.

Center System:

To install STISIM Drive on the center computer system, simply put the CD into your drive and wait for the STISIM Drive Installation window to open. Move the mouse cursor over the option Center System on figure 4.2 until it changes colors and then use the left mouse button to click on it. This will start the installation process.

If after several seconds, the STISIM Drive Installation window does not appear, use Windows Explorer to access the CD-ROM, open the Center System folder and double click on the Setup.Exe application.

VERY IMPORTANT: If you have a USB hardware protection key (dongle) do not plug it in until after you have completed the STISIM Drive installation.
Once the installation begins running you will be bombarded with various information screens, such as the preparing to install screen which basically tells you that the software is preparing to install.

Side Systems:

If you have purchased a wide field of view system, you will have to perform a software installation on each of the side machines. However, before installing the software on the side computers, make sure that they have been properly setup and configured correctly as specified in the Networking and DCOM sections of the hardware configuration guide.

As with the Center System, when you place the CD-ROM into your disk drive, after a few seconds the STISIM Drive Installation window should appear. Use the mouse to move the cursor on figure 4.2 so that the Side Systems label is highlighted (changes colors) and then click using the left mouse button to initiate the installation process:

If after several seconds the STISIM Drive Installation window does not appear, you can install the software directly by opening Windows Explorer, pointing to your CD-ROM drive, opening the folder Side System, and double clicking on the file Setup.Exe.

At this point all you will need to do is follow the directions just like with the Center system. The side system installation looks almost exactly the same as the center system installation until you reach the end where you must select the type of system being installed. Previously you selected the systems model number, now

instead of the model number you must select the side system being installed.

Like with the Center display system you must also select if the nVidia advanced monitor support is turned on or off. Clicking on the Apply button should then configure your system. You should then reboot.

Unfortunately, this is not the end of the journey when installing a wide field of view system because the side computers use Distributed Common Object Modules (DCOM) in order for the center system to control them and this also needs to be setup. Basically what DCOM provides is a way for the center computer to launch/terminate software that will run on the side computersr computers, thus you do not have to go to each computer and run the software yourself. When DCOM is successfully configured, when you select to run a scenario the software will automatically be launched on the side computers and communication between machines will commence. This means that the side computers can basically sit there and require no interaction except when booting up or shutting down. Refer to the hardware configuration section for the intricacies of configuring DCOM.

CHAPTER 5

ROADWAY NETWORK GEOMETRY

For the driving simulation, the transportation scenario can be modeled as either corridor-level or network-level. For example, STISIM Drive (12) supports only the corridor-level scenario; that is, a users decision to take a turn at any intersection is immaterial. As a result, a scenario required to capture a network of roads cannot be modeled because it will provide the same road geometry and virtual environment. The architecture developed in this current research aims at both corridor-based and network-based scenarios.

In a driving simulator where microscopic models are used for surrounding traffic, accurate network geometry is important. Small variations can affect speeds of the vehicles in surrounding traffic. Traffic controls and signal timings have a considerable influence on drivers behaviors due to their acceleration/deceleration actions. Imprecise representation of network components – such as the link length, left-turn and right-turn bays etc., can significantly affect driving simulator application fidelity.

The actual road network data of a city can be obtained from Open Street Maps (OSM) (14) in .xml format; it includes latitude, longitude, street names, intersection details, and horizontal curve information. The OSM, which is a set of world maps maintained by users across the globe, has all the freeways, major roads, and many minor streets

of every major city. Since these maps belong to an open source community, they are maintained by the users across the world by giving access to network data for most of the cities. This expands the scope of the simulator software eventually to be applicable worldwide. The OSM data can be used to generate the network of roads in the VR of the simulator. Missing data, such as lane information, traffic control at intersections, and signal timings, can be obtained from local or state agencies as well as from any of the available simulation models; then this data can be merged with the OSM data.

Implementation

In this research, the transportation network of Las Vegas, Nevada was created, based on the data obtained from the OSM. Lane information data, traffic control, and signal timings were mapped from the existing Las Vegas model in DynusT. To obtain the correct mapping of a network, the coordinates of nodes from the OSM were matched to coordinates of nodes of the Las Vegas DynusT model. The OSM data combined with the DTA data gave the capability to automate the generation of networks for the VR environment. A part of the Las Vegas roadway network created through this approach is shown in Figure 13.1. This methodology reduced the time to generate a roadway network by multiple folds, and can be used to generate VR for any city. Details regarding this methodology are discussed in Chapter 6 of this section.

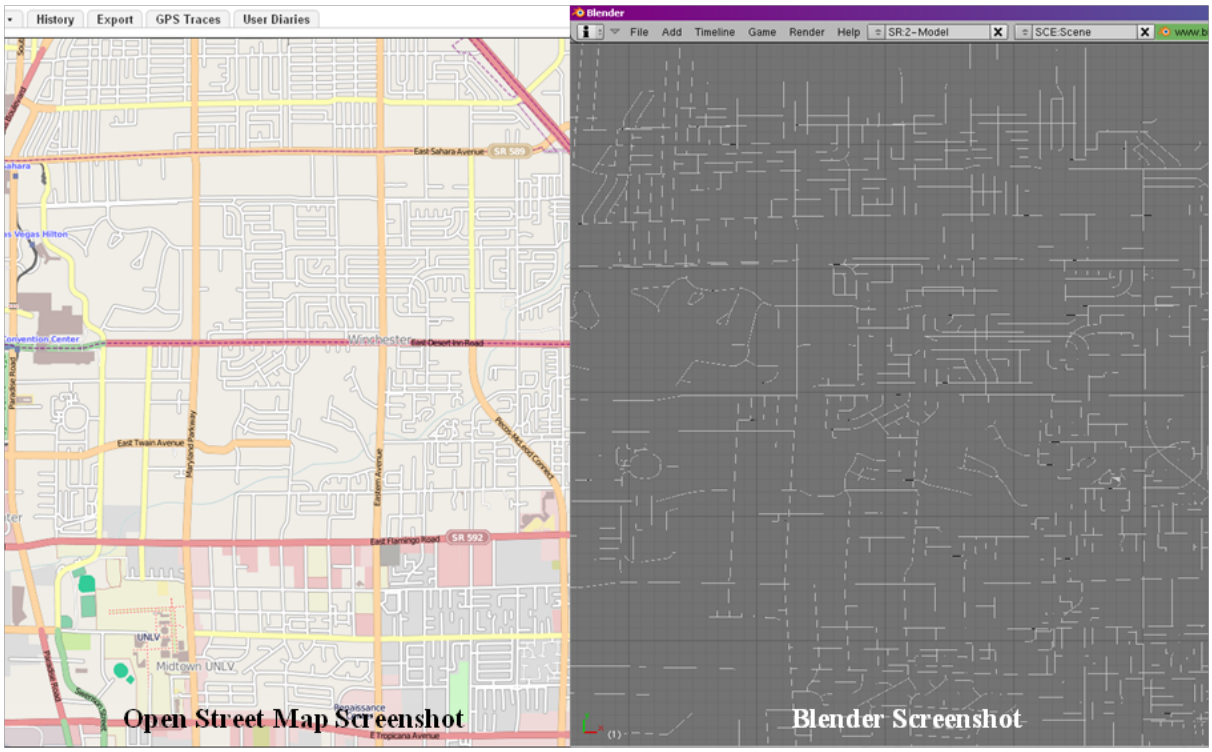

Figure 5.1: Generated network on blender using open street map

CHAPTER 6

HYBRID SIMULATION MODEL

Primarily, driving simulators are used to study driver behavior and interactions of the driver with surrounding traffic. This requires microscopic simulation of surrounding vehicles in the neighborhood of user-driven vehicle. The sophisticated microscopic simulation models, such as CORSIM (21), AIMSUM (22), PARAMICS (23), VISSIM (24), or MITSIM (25), are not required. Therefore, the microscopic model in a hybrid simulation model implemented in this research considers only a car-following model with lane changing and gap-acceptance characteristics.

Different driving simulators use microscopic simulation, depending upon the number of vehicles travelling around the user-driven vehicle with the help of such survey data as hourly volumes/distributions (12), the desired traffic density in the simulation at the visible zone of the driver (26, 27). However, if a driver is involved in a crash, such effects as congestion and queue spillback will affect the drivers road/link as well as the network of roads nearby. Microscopic simulation can capture these effects; however, the computational and time requirements will increase with the size of the network. Therefore, in order to capture network dynamics as a result of a drivers behavior, other than the road used by driver, an improved model is required. This can be achieved by employing a mesoscopic simulation that involves less computational and

time requirements by compromising simulation fidelity and detail (28). In order to capture both the micro-level vehicle dynamics on a road where user-driven vehicle is present and also the traffic dynamics at other places in network, a hybrid simulation model that integrates a microscopic and mesoscopic simulation model was chosen for the research. Integrating the microscopic with mesoscopic models concur with aggregation/disaggregation of the flows because of the simulation of individual vehicle dynamics, which otherwise becomes an issue with macroscopic model integration (29, 30).

To the best of our knowledge, so far, the only study that used a hybrid simulation model for a driving simulator is research developed by Olstam et.al. (27). A hybrid traffic simulation model was applied in a driving simulator to generate and simulate surrounding vehicles that are realistic. The model that was developed only simulated the closest area of the driving simulators vehicle. The area was divided into one inner region and two outer regions. Vehicles in the inner region were simulated according to a microscopic model, while vehicles in the outer regions were updated according to a mesoscopic model. The authors mentioned that the further research is required to address the following in their model:

- Arterials and freeways with three or more lanes;
- Ramps on freeways and intersections on rural roads;
- Simulation of urban traffic conditions; and
- Simulation on roadway networks.

The hybrid simulation model in development is aimed at addressing these limitations. The critical steps to be considered while developing hybrid simulation models are the compatibility of two different traffic flow models as well as the propagation of traffic conditions at interfaces (31). Traffic propagation at interfaces should be analyzed both at free-flow conditions and at congested conditions. At free-flow conditions, conservation of vehicles can be easily achieved if traffic flows are satisfied at upstream and downstream interfaces in both the models. Under congested conditions, shock waves moving forward and backward can be formed, which can be produced from both models individually (31).

Although mesoscopic models follow traffic flow theory, the vehicles move in an aggregate level. However, in microscopic models, vehicles move according to individual vehicle dynamics. So, at interfaces of meso to micro and micro to meso transitions, traffic propagation both upstream and downstream has to be considered. Due to integration of different resolution models, data exchange is required. Based on their updating time steps, it is necessary to define times when both the models will know the state of system simultaneously, thus controlling data exchange (32). These steps will be considered during the development of the hybrid simulation model for the simulation of surrounding vehicles in the driving simulator.

Implementation The proposed hybrid simulation model integrates the DynusT (16), a simulation-based dynamic traffic assignment mesoscopic model, with a microscopic simulation model, a car-following model with lane changing behaviors and

gap-acceptance characteristics. This approach considers the effects due to interaction between the user and surrounding vehicles, not only in the microscopic simulation region but also in other regions of the network. Integration is dependent upon identifying the region where the microscopic simulation model has to be implemented. Identification of such a region is defined around the user-driven vehicle. The motion of the users is not deterministic; that is, the users are not fixed to traverse in any particular region of the whole network. The region governing the micro simulation, called the Sim zone, has to move along with the user. This can be achieved by the following methods:

- Method 1 (Moving Sim zone). Apply the microscopic simulation model only on the roadway link where the user is present, but show the VR environment to the extent of the drivers visibility limit. Apply mesoscopic simulation on links other than that of the drivers link. In this method, the Sim zone will keep moving over the link chosen by the user.

- Method 2 (Fixed Sim zone). Apply the microscopic simulation model for the entire zone with reference to the position of the user, and show the VR environment to the extent of the drivers visibility limit. Apply mesoscopic simulation on links other than those inside the fixed Sim zone. In this method, the Sim zone is the fixed zone with respect to the user, and can intersect the links as well as the surrounding area.

Both the above methods will generate results; the best method can be decided after the implementation. Once the region is identified, the problem of conserving vehicles should be solved at the boundary of mesoscopic and microscopic integration. The mesoscopic simulation network is assumed to be the parent, and the microscopic simulation region is assumed as its child. The data generated by mesoscopic simulation inside the Sim zone is updated based on microscopic models. The proposed data exchange between mesoscopic and microscopic simulation regions reduces database and communication overheads.

CHAPTER 7

USER-DRIVEN VEHICLE DYNAMICS MODEL

For driving simulators, realism of the experience is a key factor. A part of the realism is the motion feedback that is generated to keep the user in the simulated reality. This motivates the user to perform as they would in a real-life scenario. Therefore, the user-driven vehicle in VR should be able to reproduce the effects of vehicle dynamics in actual reality. This also generates certain physiological and psychological states of the driver, which is an interesting area of research. Motion-axis simulators come in varying DOFs, typically ranging from two to fourteen DOFs (8-13, 26). Implementation of the motion dynamics to such simulators implies understanding the need for the user experience, thus weeding out unnecessary computations. This is the primary reason to focus on 3-DOF and 6-DOF models for implementation of vehicular dynamics.

Implementation

The simulation of vehicle dynamics is implemented on a 3-Degrees Of Freedom (DOF) motion base, purchased from SimCraft. The orientation of the chassis for the user-driven vehicle model in VR – and the reaction due to acceleration/deceleration – is mapped to the 3-DOFs of the hardware. The hardware used in this study has a

limited workspace, since it has only 3-DOFs, but can realistically reproduce most of the vehicle dynamics. In the future, 6-DOF hardware will be implemented, popularly known as Stewarts Platform. After realization, user-driven vehicle model will have the capability to use 3-DOF and 6-DOF motion base.

CHAPTER 8

PEDESTRIAN, BICYCLE AND MOTORCYCLE SIMULATOR

Pedestrians are an integral part of any transportation project. Existing pedestrian models (33, 34) require detailed data collection for the analysis of the interaction between pedestrians and vehicles. Recently, more studies (35) started to focus on pedestrian and driver behaviors at crosswalk locations, where pedestrians and vehicles often interact. These studies, intended to determine pedestrian and driver behaviors, collected such pedestrian data as estimated age, gender, observing behavior (e.g., head movement), time spent on the road crossing, direction of travel, and location of the pedestrian in or out of the crosswalk area. Likewise, studies also noted motorists behaviors, including vehicle speed, type of vehicle, direction of travel, and pedestrian activity within the drivers observation zone. With the help of such data, interaction between pedestrian and drivers were modeled; however, these models could not reproduce the reality. To overcome such problems and capture realistic interactions, a pedestrian simulator networked with driving simulator was required. This necessity motivated the authors to introduce a pedestrian simulator networked with a driving simulator. This system was first of its kind in the field of driving simulators research. Sensors that identify natural interactions are the key for implementation for a successful pedestrian simulator. Natural Interaction (36) is defined in terms of experi-

ence: people naturally communicate through gestures, expressions, movements, and discover the world by looking around and manipulating physical stuff; the key assumption here is that they should be allowed to interact with technology as they are used to interact with the real world in everyday life, as evolution and education taught them to do.

According to a report by the National Highway Traffic Safety Administration (NHTSA) (37), every year at least 50% of the motorcycle fatal crashes involve single vehicle crashes; of that percentage, 41% had a blood alcohol concentration of .08 g/dL or higher. Safety is the primary concern not only for car drivers but motorcycle riders too. Motorcycles are a widely acclaimed mode of transport worldwide. In the last decade, two wheeler crashes have been on the increase. Because driving simulators enable researchers to perform behavioral studies under safe conditions, a motorcycle simulator should be included in the network of the driving simulator. For the same reason, the bicycle simulator can be built and networked to this driving simulator system. The proof of concept study is in progress for motorcycle and bicycle simulator components.

Implementation

The pedestrian simulator is composed of state-of-the-art Natural Interaction Sensors (Microsoft KinectTM) coupled with a head-mounted display. MS KinectTM comes with a 3D sensor which identifies joints, body structure, facial features and voice. These basic elements can then be used to identify any human gesture. With the

help of this technology, a gesture for on the spot walking can be identified and run through the pedestrian simulation model (microscopic). The pedestrian (subject) is immersed into VR environment using head mounted display and the pedestrian simulation model of the system for the generated graphics and resulting interactions are recorded. Such interactions between the pedestrian simulator and the traffic simulations create variances due to individual user behavior on the road. Using such interactions, the data collected through pedestrian simulator will lend insight in their behavior.

CHAPTER 9

VIRTUAL REALITY ENVIRONMENT

A Virtual Reality environment is a simulated reality for any user. It is primarily a visual experience shown on stereoscopic or computer displays. The seven different concepts of virtual reality are: simulation, interaction, artificiality, immersion, tele-presence, full-body immersion, and network communication (38). VR enables interaction between simulations and users, thus providing an immersive experience into an artificial world. Such an environment has the objective to be perceived realistically within the bounds of the specifications defined by the hardware and software. Visual perception in driving simulator is dependent primarily on graphically rendering a 3D world, including details like trees, buildings, landmark objects, and roadside objects along with surrounding traffic. Generally, a 3D world is created by placing 3D models by using a SDL (12) or else by editing the 3D world by means of GUI, using imported 3D models (13). These methods have become a time-consuming process as the size of the network and the number of 3D models grows, thus generating a need to automate the process. A challenging problem in automation is creating and deploying 3D models at exact locations without deforming their sizes and shapes. Microscopic simulation model regulate creation of 3D vehicle and pedestrian objects in users visibility limit, Driving characteristics for individuals change significantly due

to different weather and light conditions, thus playing a vital role in drivers behaviors. To capture such driving characteristics, visual implementation of these conditions is very important.

Current research provides an insight into strategies that can be employed for automating the generation of a 3D world. This is achieved by exploiting the method to generate a simulation network, and replaces traditional SDL and GUI editing-based scenario generation with a data driven approach. The data-driven approach is made of layered architecture in a hierarchical way; each layer corresponds to the generation of specific objects of a 3D world, using data obtained from various sources. The associated tasks at each layer can run in parallel.

Implementation The layered architecture for the generation of a 3D VR world used in this current research is shown in Figure 4. The user relates a virtual world to a real world by the 3D virtual environment of the roadway system. In this research, a list of landmarks is created and exported with the help of Google Earth. The 3D models of the exported list are obtained from Google SketchUp (39) or else created in Blender. Landmarks are those objects that are easily recognizable, such as popular buildings or historical structures. The purpose for including landmarks is to provide a perception of familiarity inside the 3D world. The placement of these models is automated based on their latitudes and longitudes. The automation of other objects like trees, buildings, and such roadside components as mailboxes, water pumps, fire hydrants, bus stop shelters, and street lights, is generated and placed randomly, based

on certain rules. These can be edited later, according to reality.

The hybrid simulation model creates car objects and pedestrians for simulation purposes. Since the interaction of traffic and pedestrians with user-driven vehicles is limited to the Sim region, the 3D objects generated in the VR world are to the extent of users visibility limit. The region covered by the visibility limit extends both front and behind the user-driven vehicle. This generated visualization, up to a visibility limit, consists of pedestrians as well as different classes of vehicles, such as cars, trucks, and semis. The vehicle objects are designed to operate intelligently by following traffic lights and signs, yielding to pedestrians, and acting according to surrounding objects. To obtain a realistic pedestrian distribution, the generated pedestrian objects operate according to requirements of the pedestrian simulation models. To save computational resources, all the graphical objects in the 3D VR will be destroyed when they leave the user visibility limit. Different levels of visibility occur due to conditions like sun, wind, snow, or fog and also day or night. These conditions are reproduced in the 3D world using various rendering techniques like shading, reflections, and shadows.

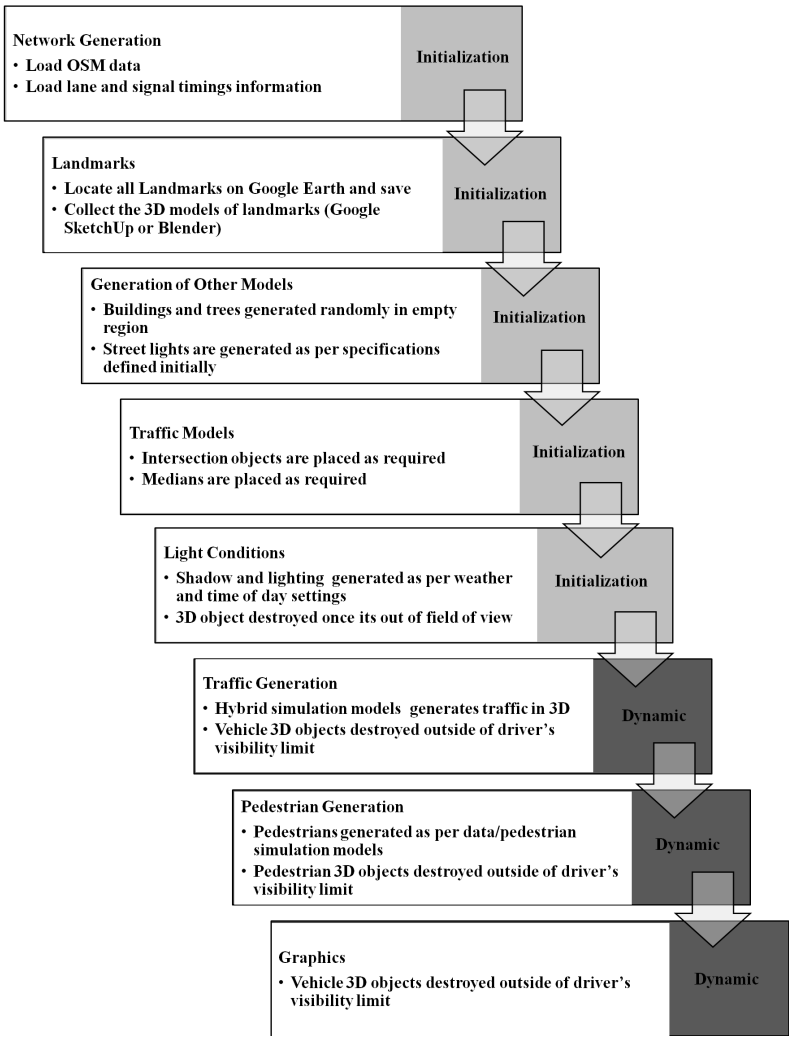

Figure 9.1: Layer-architecture for virtual reality generation

CHAPTER 10

DATA COLLECTION

Generally, driving simulators collect data on human factors. High fidelity driving simulators also capture eye-tracking, psychological data, and physiological data. This architecture aims at comprehensive data collection that includes vehicle characteristics data, such as lateral position, vehicle path, vehicle heading angle; driving behavior data, like acceleration, braking, time to collision; such psychological data as that derived from electroencephalograms; and physiological data derived from electrocardiograms, galvanic skin response, and body temperature.

The integrated hybrid simulation engine will collect data for traffic performance measures. This data can be post processed to calculate network level emission and safety. Further, microscopic details on traffic performance can be collected on links that the driver will traverse. The data collection module is integrated in every player which will record their individual driving/walking behaviors. The server data collection module groups data based on the motion type – car, bike, or pedestrian – and the number of players. This data then can be processed and analyzed by means of built-in analytical models, according to requirements. The entire simulation run can be recorded at each player as well as at the server.

CHAPTER 11

INTRODUCTION & LITERATURE SURVEY

A vast number of studies, for example (1-5), have illustrated the potential of driving simulators to analyze actual driver behavior for multiple purposes such as traffic safety and information provision. The history of driving simulators can be traced back to the 1920s, with research for various purposes (6). In the 1980s, Daimler-Benz (7) developed a high-fidelity driving simulator, encouraging others to develop new and even better simulators. Several researchers and commercial companies developed driving simulators, starting from fixed-based simulators to the most advanced simulators known today. Some of the newest driving simulators include: the National Advanced Driving Simulator (NADS), funded by NHTSA and maintained by the University of Iowa (8); the Driving Environment Simulator (DES), developed by the University of Minnesota (9) in collaboration with AutoSim and Realtime technologies; the VTI Simulator IV by the Swedish National Road and Transport Research Institute (10); the DriveSafety driving simulator by the University of Michigan Transportation Research Institute (UMTRI) (11); the STISIM Drive driving simulator by System Technology Inc., (12) and the UC-win/Road driving simulator by FORUM8 (13).

The National Advanced Driving Simulator (NADS) Laboratory (8) at the University

of Iowa has some very advanced driving simulators including: NADS-1, NADS-2, and the MiniSim simulator. NADS-1 is an advanced motion-based ground vehicle simulator. NADS-2 is similar to NADS-1 but fixed-based. The MiniSim is a PC-based, high-performance driving simulator that uses the same technology as NADS-1. MiniSim can be used at a lower cost than NADS-1 and NADS-2 because it is portable, and easy to set up and operate.

The Human Factors Interdisciplinary Research in Simulation and Transportation Program (HumanFIRST) at the University of Minnesota (9) has the Driving Environment Simulator (DES). DES is an immersive driving simulator that provides high fidelity simulation to generate a realistic presence within the simulated environment. DES measure psycho-physiological responses, including brain activity – for example, Evoked Response Potential (ERP). DES also includes highly accurate eye-tracker software. HumanFIRST also has a portable, low-cost driving simulator that uses the same technology as DES.

UCwin/Road (13) is a Virtual Reality (VR) environment where the driver can navigate in a 3D space. The environment, including a traffic simulation and visualization tool, uses ground texture maps and can include 3D building images. The environment also includes traffic generation models to generate traffic on various lanes and roads. Although the existing driving simulation models provide tremendous capabilities to study driving behavior in a safe and controlled environment, there are multiple aspects of the real word that can be addressed to significantly enhance modeling realism. This study proposes an architecture for an interactive motion-based traffic simulation

environment. In order to enhance modeling realism, the proposed architecture integrates multiple types of simulation, including: (i) a motion-based driving simulation; (ii) a pedestrian simulation; (iii) a motorcycling and bicycling simulation; and (iv) a traffic flow simulation. This integration enables the simultaneous and interactive interaction between actual and simulated drivers, pedestrians, and bike riders. In addition, the architecture provides capabilities to simulate the entire network at a reasonable price; in this way, the drivers, pedestrians, and bike riders can navigate anywhere in the system.

To increase modeling realism, the proposed architecture enables the actual humans to experience background traffic while the effects of the actual human decisions are also experience by the background traffic. To achieve this interaction, the background traffic is modeled using a hybrid meso-microscopic traffic flow simulation modeling approach. The mesoscopic traffic flow simulation module of the hybrid model loads the results of a user equilibrium traffic assignment solution and propagates the corresponding traffic through the entire system. The microscopic traffic flow simulation model provides background traffic around the vicinities where actual human beings are navigating the system. The two traffic flow simulation models interact continuously to update system conditions based on the interactions between actual humans and the fully simulated entities. The interaction between actual and background traffic has tremendous implications. For example, in the real world, an accident as consequence of a human error, can affect a large portion of the traffic system. These types of scenarios are of significant interest for a number of applications. They can

be easily modeled using the proposed architecture.

Implementation efforts are currently in progress, and some preliminary tests of individual components have been conducted.The implementation of the proposed architecture faces significant challenges ranging from multi-platform and multi-language integration to multi-event communication and coordination. To address some of those challenges and achieve the greatest benefits at the lowest cost, state-of-the-art technologies are currently been used to implement the proposed architecture. Some of these technologies include: (i) Open Street Maps (OSM) (14); (ii) Blender (15); (iii) DynusT (16); CORBA; and free SDKs, such as MS Kinect (17) and Ardunio (18). The proposed architecture is here called Networked Motion-based Interactive PEdestrian and Driving Simulator (n-MIPEDS). Although, particular suggestions to implement the proposed architecture are provided here, the conceptual architecture is general and can be implemented using multiple technologies. Appropriate modules can be developed depending on available hardware. In particular, this study uses a SimCraft three-axis motion-based driving simulator.

CHAPTER 12

SYSTEM ARCHITECTURE

At the core, a driving simulator is about collection of responses/behaviors for corresponding stimuli to users within a controlled environment. This research focuses on a driving simulator and a simulation environment that recreates the real world as well as motion simulation. In addition to developing the driving simulation, this research integrates the pedestrian simulation, bicycle and motorcycle simulation and traffic simulation onto one platform. Simulator associated with different simulation can be called a player; thus, a multiplayer architecture evolves as each player is connected over a communication network (LAN or internet). Therefore, defining a set of requirements of such a system in terms of hardware and software is important. For this, the system architecture diagram and data flow diagram are shown in Figure 12 and 12, respectively.

The different modules shown in Figure 12 are described as:

- **Motion based driving simulator**: With the help of a motion base, the vehicle dynamics can be felt on driving simulation as vehicle traverses through the system. The integration of the vehicle dynamics model for a user-driven

vehicle was done on a motion base having 3 Degrees of Freedom (3DoF), bought from SimCraft. This simulator consists of a computer CPU with graphics card, a display setup comprising of Liquid Crystal Display (LCD) screen, a 3DoF motion base, and joystick.

- **Pedestrian Simulator**: This is a unique approach to understanding the behavior of pedestrians by actually putting a user in various simulated conditions. Proof of Concept (PoC) was tested using ArduinoTM, and implementation is in progress using Microsoft KinectTM. This consists of a computer CPU with graphics card, a HUD, and Microsoft KinectTM.

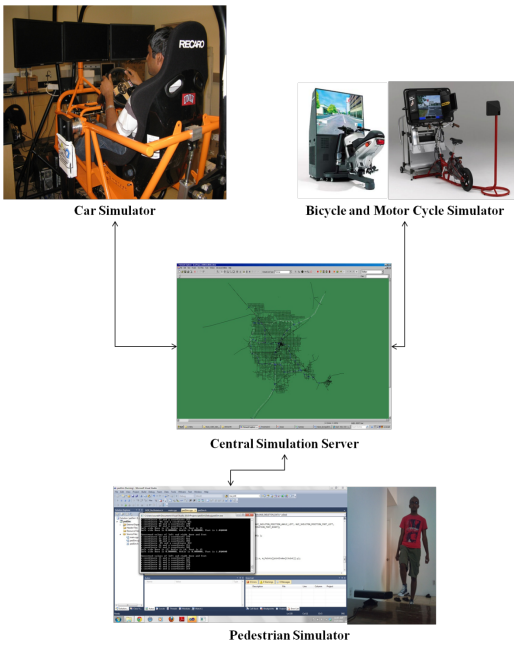

Figure 12.1: System architecture of n-MIPEDS. The images of bicycle and motorcycle simulators are indicative only. Copyright: Honda

- **Bicycle and Motorcycle Simulator**: This is also a unique approach for understanding the behavior of motorcycles and bicycles on the road. This simulator has a motion base to simulate a self-powered or fuel-powered bike. Testing for proof of concept to be done. This consists of a computer CPU with a graphics card, a Head-Up-Display (HUD) setup or LCD screen, a motion base, and a joystick.

- **Simulation server**: This server has a high end computer CPU with at least a 7200-rpm SATA Drive or SSD, and 8GB of RAM running Microsoft Windows 64bit edition. Networking hardware requires a 1000BaseT Gigabit Ethernet as well as the necessary routers and switches to realize the network.

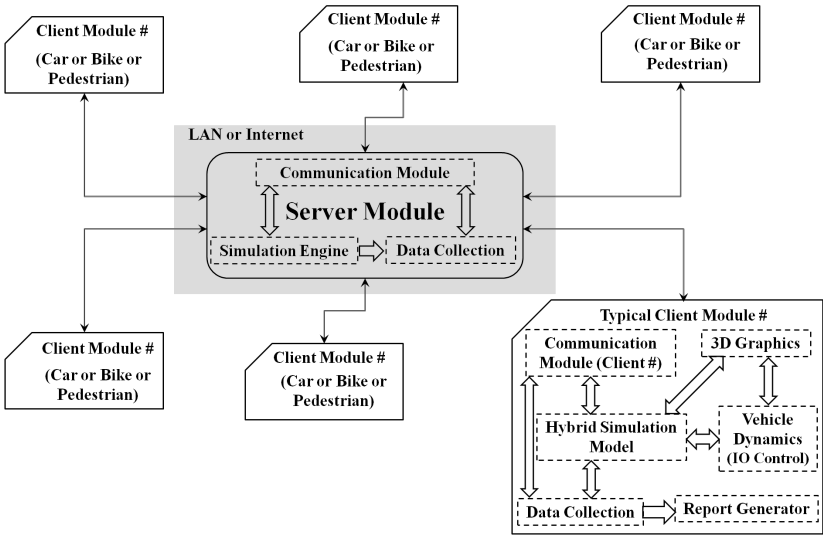

Figure 12.2: Data flow diagram.

The various modules of Figure 12 are described as follows:

- **Hybrid Simulation Model**: In traffic simulation, a hybrid simulation model will be used to capture the dynamics of surrounding vehicles over distributed computing, creating a Multi-Agent System (MAS). A MAS is defined as a set of combination of software and human entities which coordinate their knowledge, goals and plans to act or solve problems (19).

- **Virtual Reality (VR)**: This component creates a simulation world and associated 3D graphics to generate images and audio, aiming to provide a means of realistic association of the current situation. Hence, it guides the user and provides necessary input for making a decision.

- **Vehicle Dynamics**: Vehicle dynamics determine the behavior of the vehicle in the VR world, using a physics engine. It controls the motion base of a vehicle as specified, and thus reproduces a realistic driving experience. This controls the input-output for simulator.

- **Data Collection and Analysis**: This module collects various kinds of data for analysis, including drivers and pedestrians behaviors and associated psychological and physiological data, along with traffic performance measures.

- **Networking Module**: This module is a distributed system consisting of multiple autonomous computers that compute using communication over a network; this is known as distributed computing. As shown in Figures 1 and 2, it can be asserted that a distributed computation system has been envisioned, and therefore a networking module is required. Various modules of software are in

different development environments, therefore Common Object Request Broker Architecture (CORBA) is used because it allows interoperability since the standard is independent of any development language or platform (20).

CHAPTER 13

ROADWAY NETWORK GEOMETRY

For the driving simulation, the transportation scenario can be modeled as either corridor-level or network-level. For example, STISIM Drive (12) supports only the corridor-level scenario; that is, a users decision to take a turn at any intersection is immaterial. As a result, a scenario required to capture a network of roads cannot be modeled because it will provide the same road geometry and virtual environment. The architecture developed in this current research aims at both corridor-based and network-based scenarios.

In a driving simulator where microscopic models are used for surrounding traffic, accurate network geometry is important. Small variations can affect speeds of the vehicles in surrounding traffic. Traffic controls and signal timings have a considerable influence on drivers behaviors due to their acceleration/deceleration actions. Imprecise representation of network components – such as the link length, left-turn and right-turn bays etc., can significantly affect driving simulator application fidelity.

The actual road network data of a city can be obtained from Open Street Maps (OSM) (14) in .xml format; it includes latitude, longitude, street names, intersection details, and horizontal curve information. The OSM, which is a set of world maps maintained by users across the globe, has all the freeways, major roads, and many minor streets

of every major city. Since these maps belong to an open source community, they are maintained by the users across the world by giving access to network data for most of the cities. This expands the scope of the simulator software eventually to be applicable worldwide. The OSM data can be used to generate the network of roads in the VR of the simulator. Missing data, such as lane information, traffic control at intersections, and signal timings, can be obtained from local or state agencies as well as from any of the available simulation models; then this data can be merged with the OSM data.

Implementation

In this research, the transportation network of Las Vegas, Nevada was created, based on the data obtained from the OSM. Lane information data, traffic control, and signal timings were mapped from the existing Las Vegas model in DynusT. To obtain the correct mapping of a network, the coordinates of nodes from the OSM were matched to coordinates of nodes of the Las Vegas DynusT model. The OSM data combined with the DTA data gave the capability to automate the generation of networks for the VR environment. A part of the Las Vegas roadway network created through this approach is shown in Figure 13.1. This methodology reduced the time to generate a roadway network by multiple folds, and can be used to generate VR for any city. Details regarding this methodology are discussed in Chapter 6 of this section.

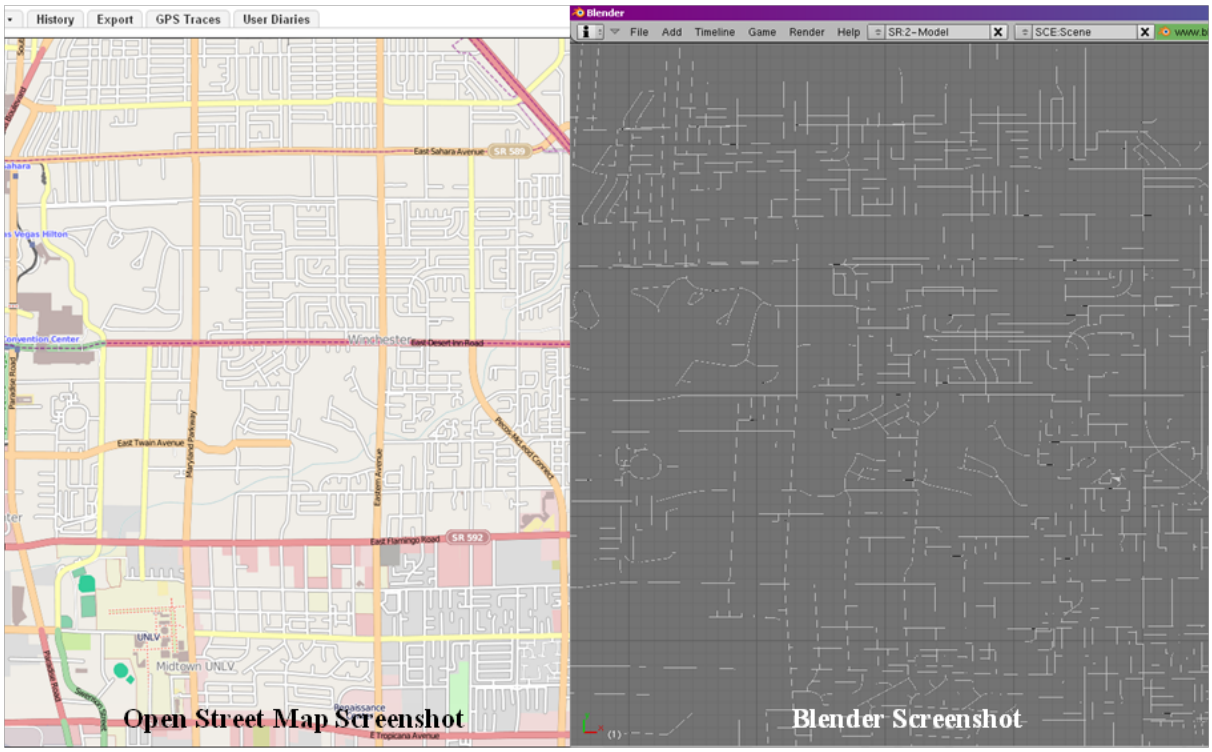

Figure 13.1: Generated network on blender using open street map

CHAPTER 14

HYBRID SIMULATION MODEL

Primarily, driving simulators are used to study driver behavior and interactions of the driver with surrounding traffic. This requires microscopic simulation of surrounding vehicles in the neighborhood of user-driven vehicle. The sophisticated microscopic simulation models, such as CORSIM (21), AIMSUM (22), PARAMICS (23), VISSIM (24), or MITSIM (25), are not required. Therefore, the microscopic model in a hybrid simulation model implemented in this research considers only a car-following model with lane changing and gap-acceptance characteristics.

Different driving simulators use microscopic simulation, depending upon the number of vehicles travelling around the user-driven vehicle with the help of such survey data as hourly volumes/distributions (12), the desired traffic density in the simulation at the visible zone of the driver (26, 27). However, if a driver is involved in a crash, such effects as congestion and queue spillback will affect the drivers road/link as well as the network of roads nearby. Microscopic simulation can capture these effects; however, the computational and time requirements will increase with the size of the network. Therefore, in order to capture network dynamics as a result of a drivers behavior, other than the road used by driver, an improved model is required. This can be achieved by employing a mesoscopic simulation that involves less computational and

time requirements by compromising simulation fidelity and detail (28). In order to capture both the micro-level vehicle dynamics on a road where user-driven vehicle is present and also the traffic dynamics at other places in network, a hybrid simulation model that integrates a microscopic and mesoscopic simulation model was chosen for the research. Integrating the microscopic with mesoscopic models concur with aggregation/disaggregation of the flows because of the simulation of individual vehicle dynamics, which otherwise becomes an issue with macroscopic model integration (29, 30).

To the best of our knowledge, so far, the only study that used a hybrid simulation model for a driving simulator is research developed by Olstam et.al. (27). A hybrid traffic simulation model was applied in a driving simulator to generate and simulate surrounding vehicles that are realistic. The model that was developed only simulated the closest area of the driving simulators vehicle. The area was divided into one inner region and two outer regions. Vehicles in the inner region were simulated according to a microscopic model, while vehicles in the outer regions were updated according to a mesoscopic model. The authors mentioned that the further research is required to address the following in their model:

- Arterials and freeways with three or more lanes;
- Ramps on freeways and intersections on rural roads;
- Simulation of urban traffic conditions; and
- Simulation on roadway networks.

The hybrid simulation model in development is aimed at addressing these limitations. The critical steps to be considered while developing hybrid simulation models are the compatibility of two different traffic flow models as well as the propagation of traffic conditions at interfaces (31). Traffic propagation at interfaces should be analyzed both at free-flow conditions and at congested conditions. At free-flow conditions, conservation of vehicles can be easily achieved if traffic flows are satisfied at upstream and downstream interfaces in both the models. Under congested conditions, shock waves moving forward and backward can be formed, which can be produced from both models individually (31).

Although mesoscopic models follow traffic flow theory, the vehicles move in an aggregate level. However, in microscopic models, vehicles move according to individual vehicle dynamics. So, at interfaces of meso to micro and micro to meso transitions, traffic propagation both upstream and downstream has to be considered. Due to integration of different resolution models, data exchange is required. Based on their updating time steps, it is necessary to define times when both the models will know the state of system simultaneously, thus controlling data exchange (32). These steps will be considered during the development of the hybrid simulation model for the simulation of surrounding vehicles in the driving simulator.

Implementation The proposed hybrid simulation model integrates the DynusT (16), a simulation-based dynamic traffic assignment mesoscopic model, with a microscopic simulation model, a car-following model with lane changing behaviors and

gap-acceptance characteristics. This approach considers the effects due to interaction between the user and surrounding vehicles, not only in the microscopic simulation region but also in other regions of the network. Integration is dependent upon identifying the region where the microscopic simulation model has to be implemented. Identification of such a region is defined around the user-driven vehicle. The motion of the users is not deterministic; that is, the users are not fixed to traverse in any particular region of the whole network. The region governing the micro simulation, called the Sim zone, has to move along with the user. This can be achieved by the following methods:

- Method 1 (Moving Sim zone). Apply the microscopic simulation model only on the roadway link where the user is present, but show the VR environment to the extent of the drivers visibility limit. Apply mesoscopic simulation on links other than that of the drivers link. In this method, the Sim zone will keep moving over the link chosen by the user.

- Method 2 (Fixed Sim zone). Apply the microscopic simulation model for the entire zone with reference to the position of the user, and show the VR environment to the extent of the drivers visibility limit. Apply mesoscopic simulation on links other than those inside the fixed Sim zone. In this method, the Sim zone is the fixed zone with respect to the user, and can intersect the links as well as the surrounding area.

Both the above methods will generate results; the best method can be decided after the implementation. Once the region is identified, the problem of conserving vehicles should be solved at the boundary of mesoscopic and microscopic integration. The mesoscopic simulation network is assumed to be the parent, and the microscopic simulation region is assumed as its child. The data generated by mesoscopic simulation inside the Sim zone is updated based on microscopic models. The proposed data exchange between mesoscopic and microscopic simulation regions reduces database and communication overheads.

CHAPTER 15

USER-DRIVEN VEHICLE DYNAMICS MODEL

For driving simulators, realism of the experience is a key factor. A part of the realism is the motion feedback that is generated to keep the user in the simulated reality. This motivates the user to perform as they would in a real-life scenario. Therefore, the user-driven vehicle in VR should be able to reproduce the effects of vehicle dynamics in actual reality. This also generates certain physiological and psychological states of the driver, which is an interesting area of research. Motion-axis simulators come in varying DOFs, typically ranging from two to fourteen DOFs (8-13, 26). Implementation of the motion dynamics to such simulators implies understanding the need for the user experience, thus weeding out unnecessary computations. This is the primary reason to focus on 3-DOF and 6-DOF models for implementation of vehicular dynamics.

Implementation

The simulation of vehicle dynamics is implemented on a 3-Degrees Of Freedom (DOF) motion base, purchased from SimCraft. The orientation of the chassis for the user-driven vehicle model in VR – and the reaction due to acceleration/deceleration – is mapped to the 3-DOFs of the hardware. The hardware used in this study has a

limited workspace, since it has only 3-DOFs, but can realistically reproduce most of the vehicle dynamics. In the future, 6-DOF hardware will be implemented, popularly known as Stewarts Platform. After realization, user-driven vehicle model will have the capability to use 3-DOF and 6-DOF motion base.

CHAPTER 16

PEDESTRIAN, BICYCLE AND MOTORCYCLE SIMULATOR

Pedestrians are an integral part of any transportation project. Existing pedestrian models (33, 34) require detailed data collection for the analysis of the interaction between pedestrians and vehicles. Recently, more studies (35) started to focus on pedestrian and driver behaviors at crosswalk locations, where pedestrians and vehicles often interact. These studies, intended to determine pedestrian and driver behaviors, collected such pedestrian data as estimated age, gender, observing behavior (e.g., head movement), time spent on the road crossing, direction of travel, and location of the pedestrian in or out of the crosswalk area. Likewise, studies also noted motorists behaviors, including vehicle speed, type of vehicle, direction of travel, and pedestrian activity within the drivers observation zone. With the help of such data, interaction between pedestrian and drivers were modeled; however, these models could not reproduce the reality. To overcome such problems and capture realistic interactions, a pedestrian simulator networked with driving simulator was required. This necessity motivated the authors to introduce a pedestrian simulator networked with a driving simulator. This system was first of its kind in the field of driving simulators research. Sensors that identify natural interactions are the key for implementation for a successful pedestrian simulator. Natural Interaction (36) is defined in terms of experi-

ence: people naturally communicate through gestures, expressions, movements, and discover the world by looking around and manipulating physical stuff; the key assumption here is that they should be allowed to interact with technology as they are used to interact with the real world in everyday life, as evolution and education taught them to do.

According to a report by the National Highway Traffic Safety Administration (NHTSA) (37), every year at least 50% of the motorcycle fatal crashes involve single vehicle crashes; of that percentage, 41% had a blood alcohol concentration of .08 g/dL or higher. Safety is the primary concern not only for car drivers but motorcycle riders too. Motorcycles are a widely acclaimed mode of transport worldwide. In the last decade, two wheeler crashes have been on the increase. Because driving simulators enable researchers to perform behavioral studies under safe conditions, a motorcycle simulator should be included in the network of the driving simulator. For the same reason, the bicycle simulator can be built and networked to this driving simulator system. The proof of concept study is in progress for motorcycle and bicycle simulator components.

Implementation

The pedestrian simulator is composed of state-of-the-art Natural Interaction Sensors (Microsoft KinectTM) coupled with a head-mounted display. MS KinectTM comes with a 3D sensor which identifies joints, body structure, facial features and voice. These basic elements can then be used to identify any human gesture. With the

help of this technology, a gesture for on the spot walking can be identified and run through the pedestrian simulation model (microscopic). The pedestrian (subject) is immersed into VR environment using head mounted display and the pedestrian simulation model of the system for the generated graphics and resulting interactions are recorded. Such interactions between the pedestrian simulator and the traffic simulations create variances due to individual user behavior on the road. Using such interactions, the data collected through pedestrian simulator will lend insight in their behavior.

CHAPTER 17

VIRTUAL REALITY ENVIRONMENT

A Virtual Reality environment is a simulated reality for any user. It is primarily a visual experience shown on stereoscopic or computer displays. The seven different concepts of virtual reality are: simulation, interaction, artificiality, immersion, tele-presence, full-body immersion, and network communication (38). VR enables interaction between simulations and users, thus providing an immersive experience into an artificial world. Such an environment has the objective to be perceived realistically within the bounds of the specifications defined by the hardware and software. Visual perception in driving simulator is dependent primarily on graphically rendering a 3D world, including details like trees, buildings, landmark objects, and roadside objects along with surrounding traffic. Generally, a 3D world is created by placing 3D models by using a SDL (12) or else by editing the 3D world by means of GUI, using imported 3D models (13). These methods have become a time-consuming process as the size of the network and the number of 3D models grows, thus generating a need to automate the process. A challenging problem in automation is creating and deploying 3D models at exact locations without deforming their sizes and shapes. Microscopic simulation model regulate creation of 3D vehicle and pedestrian objects in users visibility limit, Driving characteristics for individuals change significantly due

to different weather and light conditions, thus playing a vital role in drivers behaviors. To capture such driving characteristics, visual implementation of these conditions is very important.

Current research provides an insight into strategies that can be employed for automating the generation of a 3D world. This is achieved by exploiting the method to generate a simulation network, and replaces traditional SDL and GUI editing-based scenario generation with a data driven approach. The data-driven approach is made of layered architecture in a hierarchical way; each layer corresponds to the generation of specific objects of a 3D world, using data obtained from various sources. The associated tasks at each layer can run in parallel.

Implementation The layered architecture for the generation of a 3D VR world used in this current research is shown in Figure 4. The user relates a virtual world to a real world by the 3D virtual environment of the roadway system. In this research, a list of landmarks is created and exported with the help of Google Earth. The 3D models of the exported list are obtained from Google SketchUp (39) or else created in Blender. Landmarks are those objects that are easily recognizable, such as popular buildings or historical structures. The purpose for including landmarks is to provide a perception of familiarity inside the 3D world. The placement of these models is automated based on their latitudes and longitudes. The automation of other objects like trees, buildings, and such roadside components as mailboxes, water pumps, fire hydrants, bus stop shelters, and street lights, is generated and placed randomly, based

on certain rules. These can be edited later, according to reality.

The hybrid simulation model creates car objects and pedestrians for simulation purposes. Since the interaction of traffic and pedestrians with user-driven vehicles is limited to the Sim region, the 3D objects generated in the VR world are to the extent of users visibility limit. The region covered by the visibility limit extends both front and behind the user-driven vehicle. This generated visualization, up to a visibility limit, consists of pedestrians as well as different classes of vehicles, such as cars, trucks, and semis. The vehicle objects are designed to operate intelligently by following traffic lights and signs, yielding to pedestrians, and acting according to surrounding objects. To obtain a realistic pedestrian distribution, the generated pedestrian objects operate according to requirements of the pedestrian simulation models. To save computational resources, all the graphical objects in the 3D VR will be destroyed when they leave the user visibility limit. Different levels of visibility occur due to conditions like sun, wind, snow, or fog and also day or night. These conditions are reproduced in the 3D world using various rendering techniques like shading, reflections, and shadows.

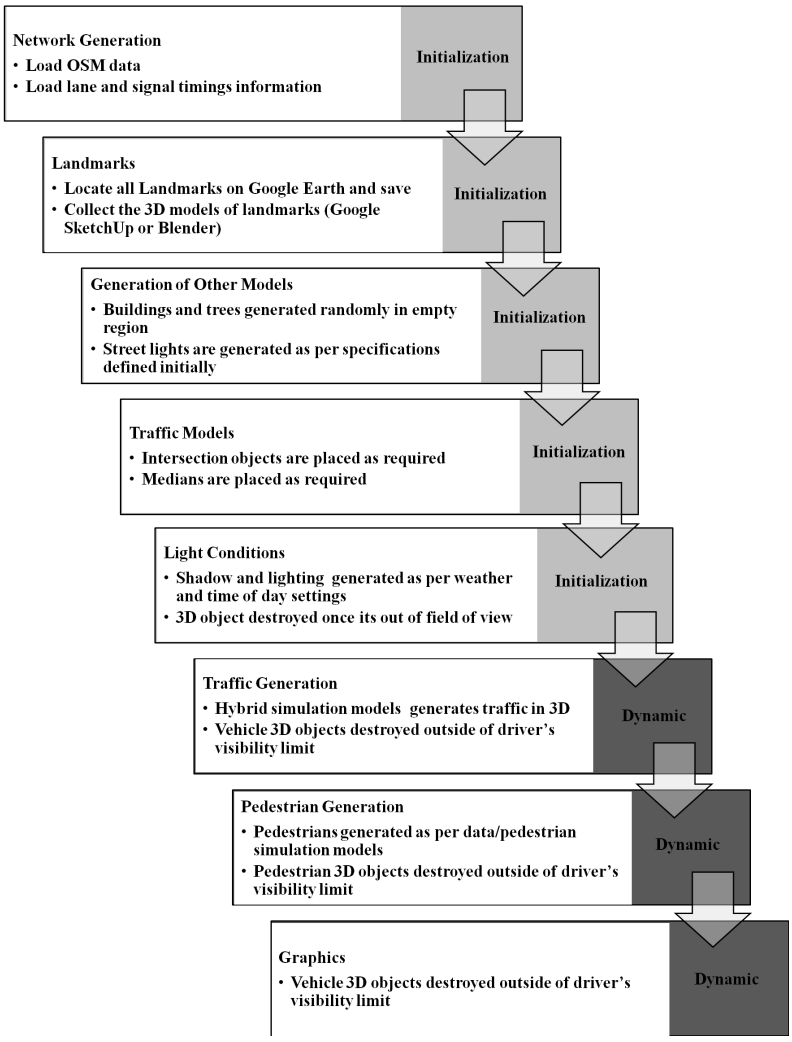

Figure 17.1: Layer-architecture for virtual reality generation

CHAPTER 18

DATA COLLECTION

Generally, driving simulators collect data on human factors. High fidelity driving simulators also capture eye-tracking, psychological data, and physiological data. This architecture aims at comprehensive data collection that includes vehicle characteristics data, such as lateral position, vehicle path, vehicle heading angle; driving behavior data, like acceleration, braking, time to collision; such psychological data as that derived from electroencephalograms; and physiological data derived from electrocardiograms, galvanic skin response, and body temperature.

The integrated hybrid simulation engine will collect data for traffic performance measures. This data can be post processed to calculate network level emission and safety. Further, microscopic details on traffic performance can be collected on links that the driver will traverse. The data collection module is integrated in every player which will record their individual driving/walking behaviors. The server data collection module groups data based on the motion type – car, bike, or pedestrian – and the number of players. This data then can be processed and analyzed by means of built-in analytical models, according to requirements. The entire simulation run can be recorded at each player as well as at the server.

CHAPTER 19

INTRODUCTION TO PEDESTRIAN SIMULATOR

Pedestrian safety is a primary concern in traffic situations since they are in the most vulnerable position. According to a report by NHTSA, 4092 pedestrians were killed and an estimated 59000 were injured in traffic crashes in United States in 2009. The numbers are very high as they account for 12% of fatalities in crash data [18]. Each pedestrian injury or fatality has many facets to it, in terms of cost to those affected directly and indirectly.

Traffic congestion is a very important aspect of travel planning inside city networks. The costs attributed to any congestion are at multiple levels and can be broadly classified in direct and indirect costs. Direct costs include delays and fuel consumption, while the indirect costs include inability to calculate precise travel time and pollution. But the buck doesnt stop here and creates tertiary effects like road rage and slow emergency vehicle response [17].

Traditionally, larger part of travel planning is done with aim to minimize travel time with preference to vehicle traffic over pedestrian traffic [10]. Some studies have attempted to plan for reducing traffic congestion by optimizing travel time costs to both pedestrians and vehicles. These studies are more relevant to traffic in heavily

travelled areas, but point towards a more subtle area of pedestrian safety. After the completion of planning, study on pedestrian safety is required to understand the effect of the new improvements. Such studies are incomplete in a broader sense of safety unless actual human subjects experience such systems. Some methods used for study of pedestrian safety are surveys and observations on an actual implemented system, over the course of time.

Pedestrian safety is also attributed to drivers of vehicles travelling through the traffic system. In this way pedestrian safety is a bi-party relationship where it is a responsibility of both driver and pedestrian. If any of the two does not understand it or fail to respect others right, eventually affect both. The study of such issues is under the broad topic of Human Factors Research.

Studies show that the demographics of an area affect the transportation behavior there. Transportation behavior here covers the interaction between transportation system and people ranging from mode choice to trip frequency and distance as well as the ways citizens affect the transportation policy. Demographics is a broad term and has many variables. With respect to the transportation behavior, certain variables have been found in high correlation such as age distribution, race and ethnicity, education level etc [8]. Statistically, a set of individuals can be identified as the representative of demographics for a region to conduct human factors research. Such research is an important topic in the field of transportation since it assesses effects on transportation systems subject to variations in user behavior due to their personal traits.

Though, individual behavior cannot be truly studied or analyzed for the entire population, due to the complexity involved between transportation behavior and demographics. Statistical methods provide the capability to represent demographical information with a smaller set of individuals, thereby creating a significant representation of the population inhabiting in that region. Such methods are based on surveys in a safe and controlled environment of a lab and have widely been accepted for study of human factors research.

As mentioned above, till now pedestrian safety has been mostly studied on established transportation systems by surveys and observations at the locations/site. Such methods though address in understanding many real life problems, have certain implied assumptions and therefore lack on a few grounds. For example, since such surveys and observations are taken on an existing system and the results are used for suggesting modifications to it, implies the system might be running in a potentially unsafe condition.

The standard in conducting pedestrian Level of Service (LOS) analysis is laid out by Highway Capacity Manual (HCM) in USA. Although, a standardized set of practices is defined for data collection and quantifying congestion in pedestrian facilities, many studies identify amendments and new methods for HCM to analyze LOS, respectively [4].

According to HCM, LOS for pedestrian part of a transportation system can be improved upon three primary areas viz. pedestrian characteristics, sidewalk environment and flow characteristics; relationship between these categories have emerged in the literature for pedestrian studies and can be illustrated as in figure 19.1 [4].

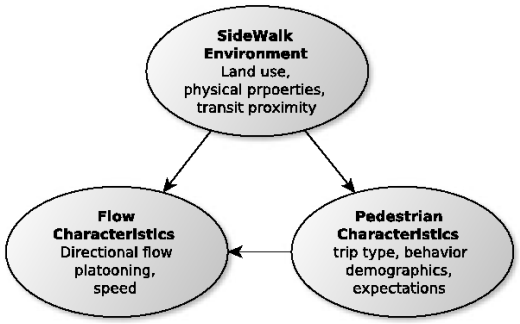

Figure 19.1: Relationship between pedestrian and traffic environment

Pedestrian characteristics identified can be broadly classified as personal characteristics, trip purposes & expectations and behavior. Personal characteristics relate variables like pedestrian speed and sidewalk widths with age, gender, group size and other demographic factors [5], [12], [26]. Trip purpose and expectations with pedestrian perceptions like safety, comfort and convenience have also been found to affect their behavior, though have not been addressed by HCM. Researchers have confirmed that pedestrians perception of environment affect their behavior significantly [21], [11], [16]. In general, have a tendency to put a cost to each sidewalk facility for

a destination on their personal expectations [9]. Similarly, individual behaviors like use of music players and mobile handsets during walk, has been criticized by various writers [2] but researchers merely have anecdotal evidences for the same and wish to understand it more.

In this thesis, we propose a methodology alongwith hardware and software development to quantify and study the effect of individual differences on pedestrian transportation system. A parallel to this methodology has existed in principle as human centered simulation studies for driving and has been proven helpful. A result expected from this work is to develop a module using which a platform can be constructed for conducting studies on pedestrian related transportation systems.

The thesis is structured by starting with introduction to theory of abstraction which forms an important pillar of the complete work, this is followed by explanation of how the human walk is studied and what are the important parameters involved. This is followed by details on hardware implementation of a system for capturing human walk and associated parameters as a proof of concept. Later a Kinect based solution is studied with software and hardware capabilities and limitations for prototype implementation. In the end we discuss about the problems, limitations and future applications.

CHAPTER 20

SYSTEM ABSTRACTION

Capturing massive data and computing it to obtain values for necessary model which can simulate a complex system is tedious, resource intensive, and complicated. For reducing the complexity of analysis for such systems, simplified models which can capture the behaviour of interest in the original system can be obtained. Such models, called abstractions, are easier to analyze as compared to the complex model. Therefore, a mathematical framework is required to filter unnecessary data and utilize the necessary information as per requirement. This chapter studies an important mathematical ideology and framework which would form the basis behind the choice of modeling in subsequent chapters.

20.1 Abstraction of Systems

The webster's dictionary define abstraction as "the act or process of separating the inherent qualities or properties of something from the actual physical object or concept to which they belong". In system theory, the objects are usually dynamical or control systems, the properties are usually the behaviors of certain variables of interest and the act of separation is essentially the act of capturing all interesting behaviors. Thus, the Webster's definition can be modified to define the abstraction

of a system as another system which captures all system behavior of interest [20]. This set of behaviors are captured by an abstracting map α and are dependent on what information is of interest (figure 20.1).

Therefore, the classic model reduction techniques can be explained as approximate

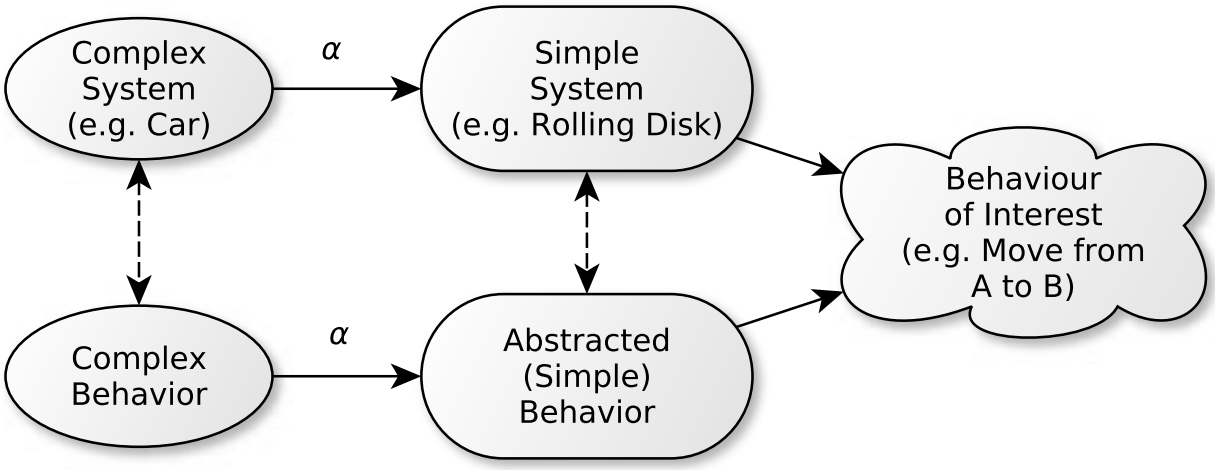

Figure 20.1: Relation between Systems and Abstractions

abstractions under this framework. It is not necessary that the abstracted and the original systems are similar from a modeling perspective. An example, can be that the original model be an ordinary differential equation but the abstracted system may be a discretized system. Therefore, under this definition, the problem can be rephrased as the following: Given an original system and an abstracting map, find an abstracted system which generates the abstracted behaviors either exactly or approximately.

20.2 Mathematical Preliminaries

Differential geometry takes an important part in understanding the following. Multiple texts can be used as reference for better understanding [22], [1].

Tangent Space: Let there be a differentiable manifold M. The set of all tangent vectors at $p\varepsilon M$ is called the tangent space of M at p and is denoted by T_pM.

Tangent Bundle: The collection of all tangent spaces of the manifold M is called a tangent bundle. mathematically it can be represented as

$$TM = \bigcup_{p\varepsilon M} T_pM \tag{20.1}$$

Projection Map: the projection map is defined from tangent bundle to manifold as $\pi : TM \to M$ taking a tangent vector $X_p\varepsilon T_pM \subset TM$ to the point $p\varepsilon M$. The tangent space T_pM can then be thought of as $\pi^{-1}(p)$. The tangent space can be considered as special case of an object called fibre bundle.

Fiber Bundles[19]: A fiber bundle is a five-tuple $(B, M, \pi, U, \{O_i\}_{i\varepsilon I})$ where B, M, U are smooth manifolds called total space, the base space and the standard fiber respectively. The map $\pi : B \to M$ is a surjective submersion and $\{O_i\}_{i\varepsilon I}$ is an open cover of M such that for every $i\varepsilon I$ there exists a diffeomorphism $\phi_i : \pi^{-1}(O_i) \to O_i \times U$ satisfying

$$\pi_o \circ \phi = \pi \tag{20.2}$$

where π_o is the projection from $O_i \times U$ to O_i. The submanifold $\pi^{-1}(p)$ is called the fiber at $p\varepsilon M$. If all the fibers are vector spaces of constant dimension, then the fiber

bundle is called a **vector bundle**.

Let M and N be smooth manifolds and $f : M \to N$ be a smooth map. Let $p \varepsilon M$ and let $q = f(p) \varepsilon N$. We push forward tangent vectors from T_pM to T_qN using the induced push forward map $f_* : T_pM \to T_qN$. If $f : M \to N$ and $g : N \to M$ then

$$(g \circ f)_* = g_* \circ f_* \tag{20.3}$$

A vector field or dynamical system on a manifold M is a continuous map F which places at each point p of M a tangent vector from T_pM. Let $I \subseteq \mathbb{R}$ be an open interval containing the origin. An integral curve of a vector field is a curve $c : I \to M$ whose tangent at each point is identically equal to the vector field at that point. Therefore an integral curve satisfies for all $t \varepsilon I$,

$$c' = c_*(I) = X(c) \tag{20.4}$$

f-related Vector Fields: Let X and Y be vector fields on manifolds M and N respectively and $f : M \to N$ be a smooth map. Then X and Y are f-related iff $f_* \circ X = Y \circ f$.

20.3 Abstracting Maps

For system analysis, reduction in complexity is associated with a avoidance of unnecessary information, thereby working with a simplified model with reduced com-

plexity. So, if M is the state space of a system, the state $p \varepsilon M$ is thus mapped to an abstracted state $q \varepsilon N$. It can be safely said that the complexity reduction requires that the dimenaion of N should be lower than the dimension of M.

The relevant information for mapping M is dependent on the required properties from the system under consideration (M). The desired specification can be quite different even in the same system as functionality may be different in various modes of its operation. Hence we cannot obtain a system abstraction without prior knowledge of the system functionality. For example, a person can move forward on two limbs or all four limbs or even a single limb. Therefore, depending on mode of operation functionality changes and hence the system specification will change.

This functionality of the system helps in identifying the states of interest for information extraction. Once the functionality of the system is identified, a set of equivalent states can be defined in the form of an equivalence relation on the state space. Thus, the quotient space $M/\sim$, determined by the chosen equivalence relation, is the state space of the abstracted system.

For this quotient space to have a manifold structure, the equivalence relation must be regular [1]. The surjective map $\alpha : M \to M$ which sends each state $p \varepsilon M/\sim$ to its equivalence class $[p] \varepsilon M/\sim$ is called the quotient map and is the mapping from each state to its abstracted state. Therefore it can be defined as following [20]

Abstracting Maps: Let M and N be given manifolds with $dim(N) \leq dim(M)$. A surjective map $\alpha : M \to N$ from the state space M to the abstracted space N is called an abstracting map.

20.4 Abstraction of Dynamical Systems

The interesting portion after determining the abstracting map α is obtaining the evolution of dynamics obtained from the state evolution on M goverened by a vector field X on M. This is characterized by integral curves and is defined as following.

Definition: Let X and Y be vector fields on M and N respectively and let $\alpha : M \rightarrow N$ be a smooth abstracting map. Then vector field Y is an abstraction of vector field X with respect to α if and only if for every integral curve c of X, $\alpha \circ c$ is an integral curve of Y (figure 20.4).

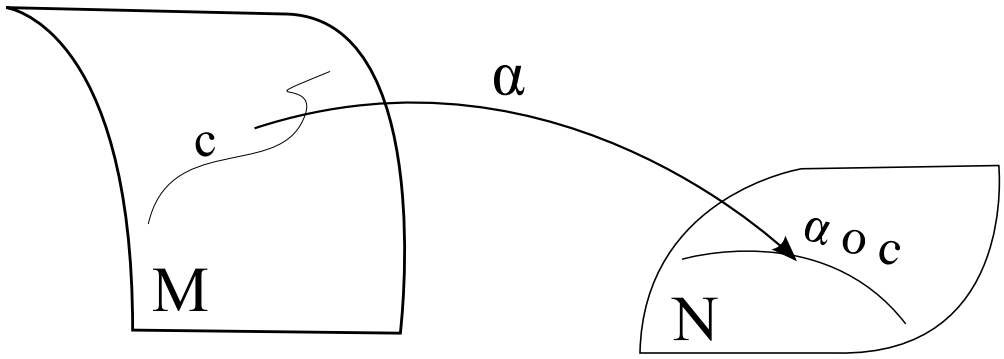

Figure 20.2: Mapping Between Spaces

i.e.

$$c' = c_*(I) = X(c)$$

implies

$$(\alpha \circ c)' = (\alpha \circ c)_*(I) = Y(\alpha \circ c)$$

Moreover, it also implies from the definition that two different abstracting maps α_1 and α_2 over some vector field X do not create same abstracted vector field Y. When modeling large scale systems, a hirerarichal approach may be chosen, thereby modeling at many levels of abstraction. Therefore, the following proposition:

Transitivity of Abstractions: Let X_1,X_2,X_3 be vector fields on manifolds M_1, M_2 and M_3 respectively. If X_2 is an abstraction of X_1 with respect to the abstracting map $\alpha_l : M_1 \to M_2$ and X_3 is an abstraction of X_2 with respect to abstracting map $\alpha_2 : M_2 \to M_3$ then X_3 is an abstraction of X_1 with respect to abstracting map $\alpha_2 \circ \alpha_1$.

Theorem. Vector field Y on N is an abstraction of vector field X on M with respect to the map α if and only if X and Y are α-related.

The above theorem is equivalent to the definition of abstraction of dynamical systems. It is important because rather than explicit computation of integral curves, it allows to check condition on vector fields. Also, the α-relatedness of two vector fields is a very restrictive condition as it limits cases where one dynamical system is an exact abstraction of another.

20.5 Abstractions of Control Systems

In this section, the theory of abstraction for dynamical systems is extended to control systems.

Control System [6]: A control system $S = (B, F)$ consists of a fiber bundle π :

$B \to M$ called the control bundle and a smooth map $F : B \to TM$ which is fiber preserving and hence satisfies

$$\pi' \circ F = \pi$$

where $\pi' : TM \to M$ is the tangent bundle projection.

Essentially, the base manifold M of the control bundle is the state space and the fibers $\pi^{-1}(p)$ are the state dependent control spaces. In a local coordinate chart (V, x), the map F can be expressed as $\dot{x} = F(x, u)$ with $u \varepsilon U(x) = \pi^{-1}(x)$.

Integral Curves of Control Systems [20]: A curve $c : I \to M$ is called an integral curve of the control system $S = (B, F)$ if there exists a curve $c^B : I \to B$ satisfying

$$\pi \circ c^B = c$$

$$c' = c_*(I) = F(c^B)$$

Abstractions of Control Systems [20]: Let $S_M = (B_M, F_M)$ with $\pi_M : B_M \to M$ and $S_N = (B_N, F_N)$ with $\pi_N : B_N \to N$ be two control systems. Let $\alpha : M \to N$ be an abstracting map. Then control system S_N is an abstraction of S_M with respect to abstracting map α iff for every integral curve c_M of S_M, $\alpha \circ c_M$ is an integral curve of S_N.

In the above definition of abstractions of control systems, it is clear that two abstracting maps for same abstraction may or may not be different. However, it is difficult to decide whether one control system is the abstraction map of other by directly using the definition since it would require integration of the system. Therefore, a set of conditions for such identification were derived as following theorem. It is analogous to the theorem about abstraction of the dynamical system detailed above.

Conditions for Control System Abstractions [20]: Let $S_N = (B_N, F_N)$ and $S_M = (B_M, F_M)$ be two control systems and $\alpha : M \to N$ be an abstracting map. Then S_N is an abstraction of S_M with respect to abstracting map α iff:

$$\alpha_* \circ F_M \circ \pi_M^{-1}(p) \subseteq F_N \circ \pi_N^{-1} \circ \alpha(p)$$

at every $p \varepsilon M$.

But this theorem does not require the commutativity. This allows the theorem to be applied in every control and dynamical system and concludes that they can be abstracted by another control system. This can be seen in following corollaries:

- **Abstractable Control Systems**: Every control system $S_M = (B_M, F_M)$ is abstractable by a control system S_N with respect to any abstracting map $\alpha : M \to N$.
- **Abstractable Dynamical System**: Every dynamical system on M is abstractable by a control system with respect to any abstracting map $\alpha : M \to N$.

Once a system abstraction has been obtained, it is useful to propagate properties of interest from the original system to the abstracted system. For control systems, one of those properties is controllability. **Controllability**: Let $S = (B, F)$ be a control system. Then S is called controllable iff given any two points $P_l, P_2 \varepsilon M$, there exists an integral curve c such that for some $t_l, t_2 \varepsilon I$ we have $c(t_l) = p_l$ and $c(t_2) = p_2$.

Controllable Abstractions: Let control system $S_N = (B_N, F_v)$ be an abstraction of $S_M = (B_M, F_M)$ with respect to some abstracting map α. If S_M is controllable then S_N is controllable.

Other properties, such as local accesibility, also propagate. Stability, however, does not propagate since the abstracted system allows redundant evolutions which could be unstable.

CHAPTER 21

THE HUMAN WALK

In this chapter we Analyze the primary objective of the module under consideration for development, i.e. The Human Walk. This chapter explains an actual human walk and associated kinematics, discusses the challenges of simulating a human walk, then defines a simulated human walk, analyses it and eventually provides a method of mapping to actual human walk from a simulated motion.

21.1 Bio-Mechanics of Human Movement

In this section we assume that the reader is familiar with basic concepts of force, energy, momentum and laws of motion and methods for their analysis. We start here from analyzing the pattern of human motion in terms of angular rotation of joints and translation.

In general, a typical walk involves rotation of limbs to achieve a motion through space. This motion is usually curvilinear in nature due to fixed length of rotating arm at every joint. This rotation reduces effective height of that particular joint from the ground. The following figure[21.1] shows the same for a human walk [24]. It shows a single step in a sequence of ankle(b), hip (a') and ankle (b") rotation providing for the forward progression of the body.

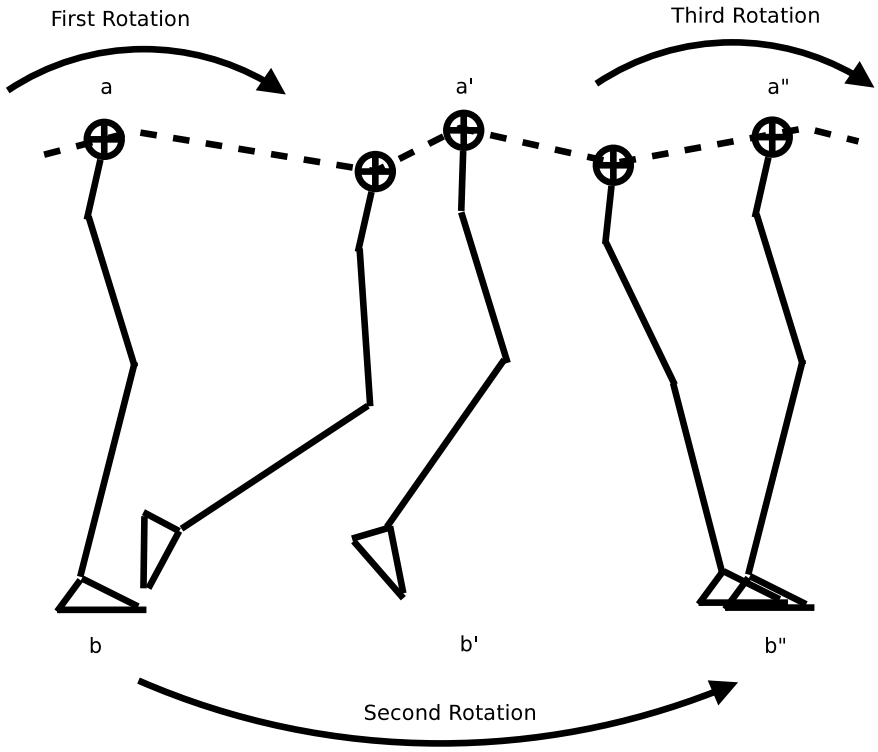

Figure 21.1: A Single Step shown for the forward progression of body

21.1.1 Movement Analysis

Majority of human movements are quite complex due to movement of body parts relative to each other and environment. A quantitative analysis of movement can clarify the muscles required to be active during that period in context of forces acting on the body. For this purpose, Inverse Dynamics is an often used technique, and two types of models are used for studies, namely Free Body Diagram and Link-segment

Model. They view the entire system for locomotion as a whole and in its parts, respectively. Eventually this calculation translates into joint forces and torques.

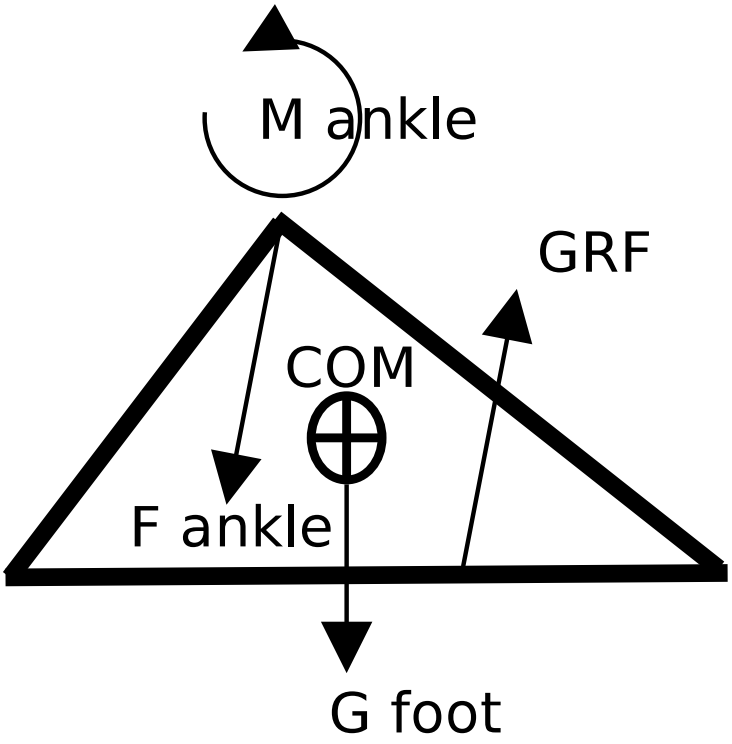

Figure 21.2: Free Body diagram of a foot

Such analysis requires three types of information:

- Anthropometric information on the segments (mass, length etc).
- Kinematic information like linear and angular moment information for each segment.
- External forces acting on body.

With the following assumptions for link segment models:

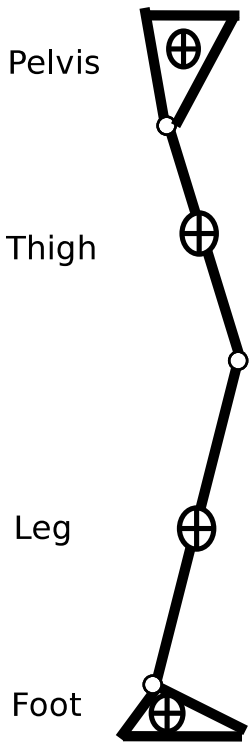

Figure 21.3: Link Segment Model

- Each segment has a fixed mass located at its center of mass.
- The joints are considered as hinge joints.
- The moment of inertia is fixed during movement.
- The length of each segment is constant.

21.2 Gait

Gait is defined as pattern of movement of limbs for locomotion for animals including humans. Every mammal follows a set of different gaits corresponding to different type of motion (walking, jogging, running etc.). However in this text we are discussing

the human walk and henceforth, gait will be used in reference to human gait only.

Characteristics of a gait are defined by differences in point of contact with surface, potential and kinetic energy cycles, overall velocity and forces experienced. Of these, the most popular and referenced is the identification of various parts of a gait by point of contact of foot with the surface.

Generally, a gait is classified as normal or pathological. The pathological gait is different from normal gait because of certain factors about the concerned subject. The deviation from normal gait can be permanent or transient and is a field of research for bio-mechanics aimed towards individuals requiring help with their walking ability. Many factors affecting a gait are identified in the literature and hence can be classified as follows [3]:

1. Extrinsic
2. Intrinsic
3. Physical
4. Psychological
5. Physiological
6. Pathological

On basis of point of contact of foot, a complete gait consists of total analysis of all the body movements, the associated mechanics and related muscle activity, for

overall pattern of limbs between re-contact of initial reference point of the foot, with the surface. To study a gait, following parameters are taken into account [23]:

1. Step length
2. Stride length
3. Cadence
4. Speed
5. Dynamic base
6. Progression line
7. Foot angle

Essentially a gait is a repetitive motion of the limbs, each sequence of this cyclic movement of limbs is called a gait cycle and is an essential derived parameter of any gait. It is an important parameter because various phases of a typical gait are defined in terms of percentage of gait cycle completed since it provides a common reference method. This will be discussed in detail during kinematic analysis of the gait.

21.3 Kinematics Analysis of Gait

Kinematics is the branch of classical mechanics describing motion of a point or a collection of points or a rigid body or group of rigid bodies and does not consider force. It is the method for quantifying and measuring the kinematic quantities for

analysis of gait. Kinematic quantities for any point or rigid body are described as position, velocity and acceleration of concerned points on it. Thus, it describes the motion of human body by studying the trajectories of various important points and lines identified by careful observation.

The requirement from the kinematic analysis of gait is defined by the objective of the problem. In the considered problem, we need the information about movement of a subject in space and time. Determination of clinical aspects of gait are not in scope of the problem. Therefore, the requirement from kinematic analysis is in terms of following parameters:

- Speed of subject
- Instantenous acceleration of subject
- Distance travelled by the subject

21.4 Simulating an Actual Human Walk

Walking is a method of transportation wherein one moves from point A to point B by following a set of movements repeatedly. This repeated pattern called the gait is an important action to be captured as it contains requisite information for the analysis of the walk. In past, solution to this problem has been attempted by many methods. Video recording of small walks, Marker based data collection via video camera and infrared sensors, treadmills etc. have been used for data collection to analyze gait.

All these methods attempt to capture a complete gait and then extract the relevant information from it for further analysis.

Though these methods for capture of complete gait are quite robust, there are certain limitations to them. Most of these methods face the problem of limited space of operation and hence are unable to work if gait capture is required for longer period of time. Some methods can capture long walks, but are unable to capture motion in two dimensions. Those systems which can capture have very complex and large setups, which limit their capability for easy and movable installation. Also such systems are quite expensive to procure.

Thus the challenges in simulating a human walk from the actions performed by a human can be listed as follows:

1. Limited space against longer time duration walks
2. Amount of setup required
3. Cost of setup required
4. Trade off between straight line walks and walk with turns

21.5 Abstraction of Human Walk

As discussed above, a human walk is composed of multiple components. Moreover a human subject has multiple degrees of freedom which create a capability for highly dextrous movements which are complex in nature. Each movement is associated with

multiple muscles and joints coordinating to provide a reliable motion.

Capturing such massive data and computing it to obtain necessary model which can simulate a human gait is tedious, resource intensive, and complicated. For reducing the complexity of analysis for such a system, simplified models which can capture the behaviour of interest in the original system can be obtained. Such models, called abstractions, are easier to analyze as compared to the complex model. Therefore, a mathematical framework is required to filter unnecessary data and utilize the necessary information as per requirement. Such a framework for abstraction of a dynamical system is discussed in previous chapter.

The proposed pedestrian simulator will capture the walk of subject and then identify the gait information from it for recreating the movements of subject in the simulated reality. This can be visualized as an abstraction of the subject's walk in the real world to simulated world. Looking at the limitations in previous setion, a much simpler, cost effective approach was required so that it can be setup and reused easily. To address this problem we first defined the gesture that can be used for long walks and hence forth identify the solution for its implementation and analyze it.

21.5.1 Definition of on the spot walk

The solution visualized was defining an on the spot walk pattern, using which certain parameters can be defined and extracted which were then transformed into gait parameters to simulate the walking. This motion is hereby defined as:

the alternating motion of taking one foot up in air, while keeping the other foot on the ground, and returning it back to ground at the same point where it was before lifting off. The knee and hip joints will rotate appropriately to allow the thigh to come as closely parallel to the surface as anatomically feasible, thereby raising the knee joint close to waistline.

Hence an on the spot walk can be visualized as a continuous sequence of above defined gesture in figure 21.4.

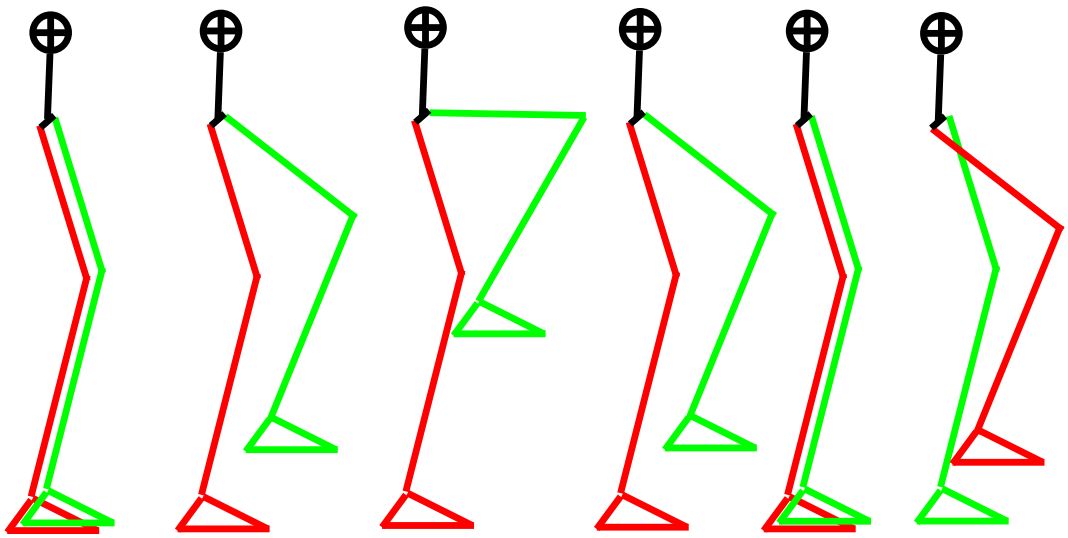

Figure 21.4: On The Spot Walk

21.5.2 Analysis of on the spot walk

The on the spot walk or rather the simulated walk is divided into two phases the Lift when one foot is rising, while the other phase is called Fall which occurs when the risen foot is coming back on the surface. This nomenclature implies that there are two lift phases and two fall phases in each simulated gait for any person. Various parameters are extracted from this simulated walk, which enable it to be mapped to an actual gait. The relationship between the phases of an actual gait and simulated walk are shown in figure 21.5.

21.6 Model of Virtual Entity in 3D World

Since the final objective of the pedestrian simulator is to interface between the physical subject and their representation, it is an important aspect to define the entity in the 3-Dimensional virtual world.

The pedestrian in the virtual world is assumed to operate under following boundations:

- A pedestrian always moves in a straight line.
- A pedestrian can move forward, backward and rotate by any angle.
- A pedestrian is able to vary speed and stride length.
- A pedestrian never takes side steps.

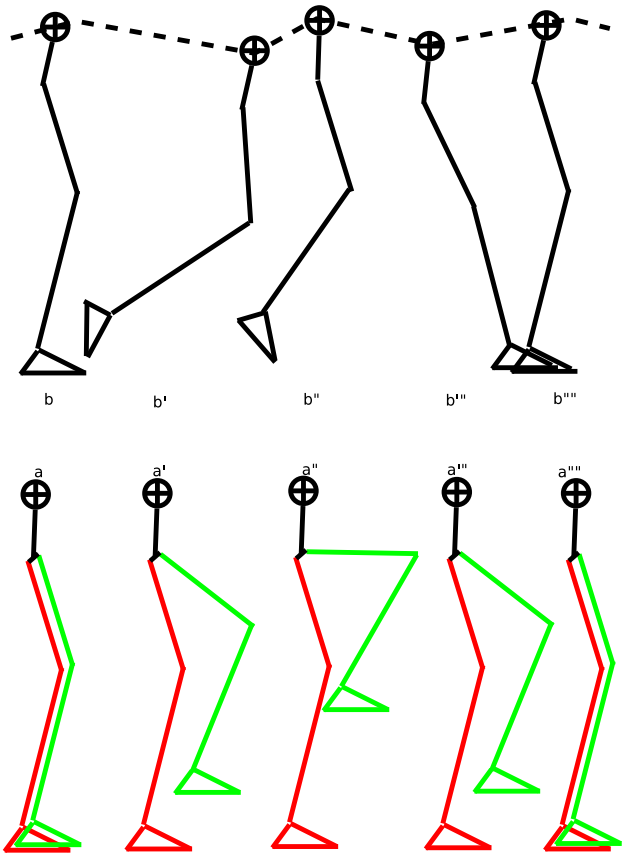

Figure 21.5: Mapping of walking phases

The above assumptions define that the pedestrian is non holonomic in nature with linear and angular motion only. To model the assumptions mathematically, a pedestrian can be assumed as a rolling disc. Since the disc cannot side step itself, i.e. the disc cannot move in the direction of its area vector, hence, it becomes a non-holonomicaly constrained system. Moreover, now the disc can have linear velocity, and angular velocity along the direction of vector in the plane containing the disc. Therefore the disc can reach any point on the 2D plane.

The rolling disc model can be described as follows: *Consider a disk of radius ρ, that rolls without slipping on a plane, as shown in figure 21.6, while keeping its midplane vertical. Its configuration is completely described by four variables: the position coordinates (x, y) of the point of contact with the ground in a fixed frame, the angle θ characterizing the disk orientation with respect to the x axis, and the angle ϕ between a chosen radial axis on the disk and the vertical axis.*[7]

Therefore, the disc must satisfy the kinematic constraints

$$\dot{x} - \rho cos\theta\dot{\phi} = 0 \tag{21.1}$$

$$\dot{y} - \rho sin\theta\dot{\phi} = 0 \tag{21.2}$$

These dynamic constraints are not integrable, therefore disk can reach any point in its state space from any point in it, by any path. It is due to this reason this can

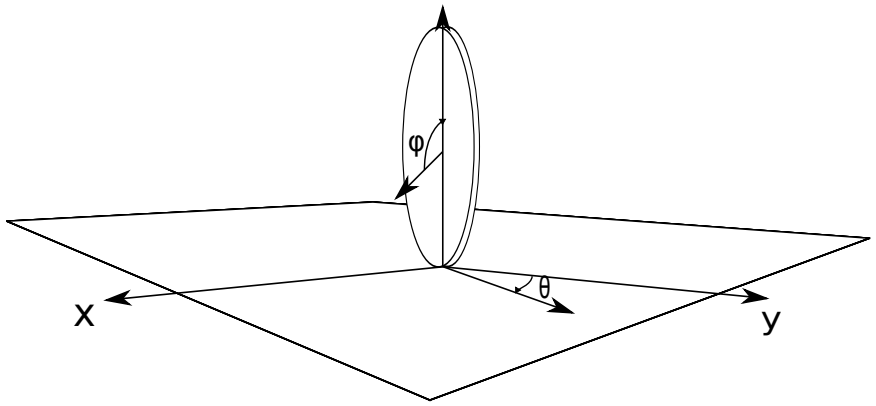

Figure 21.6: The rolling disc

be assumed as the appropriate model for a pedestrian as each point of the virtual world is reachable.

21.7 Abstracting Map

Hence our state space M can be considered as state space that belongs to the motion of a skeleton in physical world. This state space has multiple dimensions and encomapsses many complex interactions. Moreover, this motion is required to be translated into the motion of a virtual entity in the 3D virtual world as the rolling disc model discussed above. Hence we can say that a particular higher dimensional, dynamical control system is being abstracted into a lower dimension dynamical control system.

For the reasons discussed before, this abstraction is the key to simplification of analysis and control for navigation by human subjects for the entities in the virtual world.

21.8 Conclusion

Though above explained approach is an inexpensive way and addresses all the limitations listed in previously tested approaches, It has a major drawback of comparing on the spot walk gait to an actual gait/walk. It is a limitation because it takes a user away from how they actually walk to a different approach which is less natural and more monotonous. This takes a user from their usual behavior and demands efforts in a continuously different situation thereby creating a greater psychological divide from reality. Therefore, it still cannot be taken without proof that eventual experience will not be engaging.

CHAPTER 22

CAPTURING HUMAN WALK

Before moving on to a project, we need to do a feasability study and identify the possible challenges in the project. This chapter details the requirements for the prototype implementation of capturing the walking gesture and its analysis.

22.1 Simulated Human Walk

In the last chapter a gait was defined and analysis methods were identified and thus defined a simulated walk for a person to be performed inside a lab environment and associated mathematics was discussed. In short, the simulated walk can be described as on the spot walking. The motion is defined as lifting the knee up to above a certain position to be classified as a step. Thus each virtual step is established by a spot gesture and relevant definitions for various parameters of a gait are defined by it and calculated accordingly.

The definition of such gesture is an important objective to prove the simulation of a Human walk and is required to be validated. For validation of this definition and the concept, a prototype was made which has been discussed in this chapter. This prototype was built using Arduino and IR sensors.

22.2 Mapping actual to simulated walk

As observed in previous chapters, an actual gait is composed of many parameters, variables, motions and patterns. Hence, it is a complicated series of events. Therefore there is a need to filter the available information for efficient computation and implementation of pedestrian simulator. For this reason, we decided to map certain parameters of an actual walk to be implemented by the above defined on the spot walking genture (Definition 21.5.1). This mapping has been given as below.

On The Spot Walk Gesture	**Simulated Gait**
Maximum Knee Height	Step Length
Knee Lift Frequency	Speed
Knee Position	Gait Position Cycle

Table 22.1: Gesture Parameter mapping

22.3 Extracted Parameters

For development of proof of concept, the requirement was to ensure a feature extraction for the defined action of walking. The primary problem in such situation is to sense physical movements, process them, and extract features and gestures and map their values into parameters in real time. As per the mapping defined in table 22.2, information about speed of walking, stride length and position with reference to gait cycle has to be extracted. This created a problem of identifying the right sensors,

and appropriate locations for tracking simultaneously minimizing the computations for fast and efficient response from the system.

The solution to this problem was implemented with the use of Infrared Sensors, placed on the feet of the subject, and two panels both in front and back of subject till knee height. The sensors on the feet were placed pointing in three directions bottom, front and back. The sensor facing towards ground was required to measure the rise of the foot and those towards front and back were to measure distance from their respective panels. The placement of sensors was aimed at recording the height of foot and the distance of foot from the front and rear panel. These features helped in computing all the required information for simulated gait as follows:

$$\text{Distance between front and rear panel} = L$$

$$\text{Height of foot} = h$$

$$\text{Position from front panel} = x$$

$$\text{Step Length } (s) = k * x/L$$

$$\text{frequency of steps } (f) = \text{No. of times foot raised above ground/second}$$

$$\text{Speed} = f * s$$

22.4 Construction

This section explains the construction for proof of concept for pedestrian simulator. The architecture is explained followed by the hardware construction and software design.

22.4.1 Architecture

The overall architecture has been divided into two parts viz hardware and software architecture. The architecture has also been detailed with respect to semantic breakdown of information flow and trigger actions. The complete flow of information can be visualized as follows:

Complete: The complete system can be visualized (figure 22.1) as a data acquisition and processing system where the sensor feedback is classified and assigned with relavant gestures to compute appropriate information. Actions and information: The

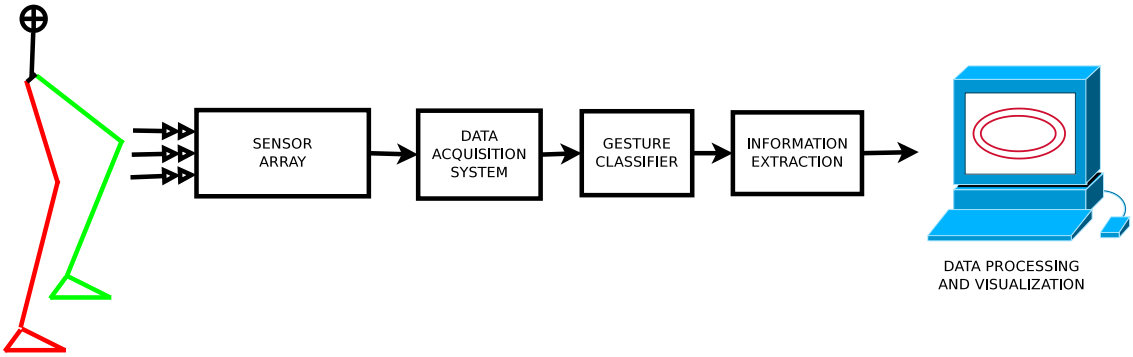

Figure 22.1: Flowchart of complete system

information flows from user's actions into the system. Here a single action of 'on the spot walking' is further broken into constituent actions as follows:

- Walking

- Stride Length
- Direction of Movement

The information flows from the actions to sensors. The data of these sensors is acquired by a microcontroller and then processed to classify between the above three actions as well as compute related information. This information flow can be visualized as in figure 22.2 .

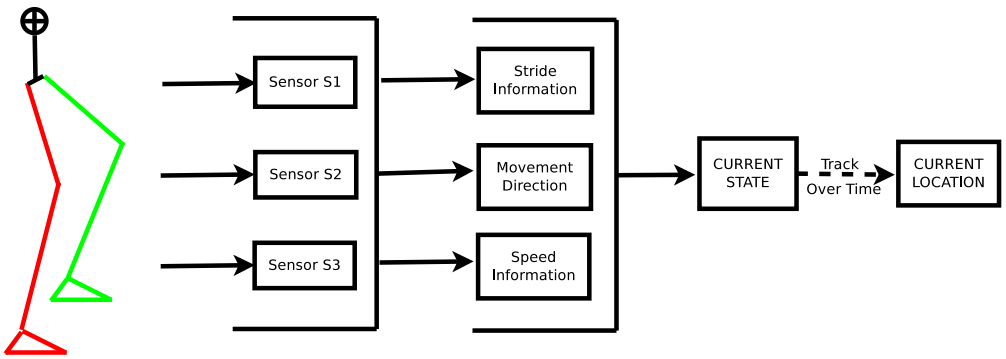

Figure 22.2: Information Flow

Hardware

The hardware architecture is explained as follows: As shown in figure 22.3. There are three infrared sensors for each foot amounting to six sensors in total. These sensors are pasted along the feet in various orientations. two sensors are facing front and back respectively and one sensor is facing downwards. Each of the sensor is connected to an analog input channel in the micro-controller from where the information goes to

the software.

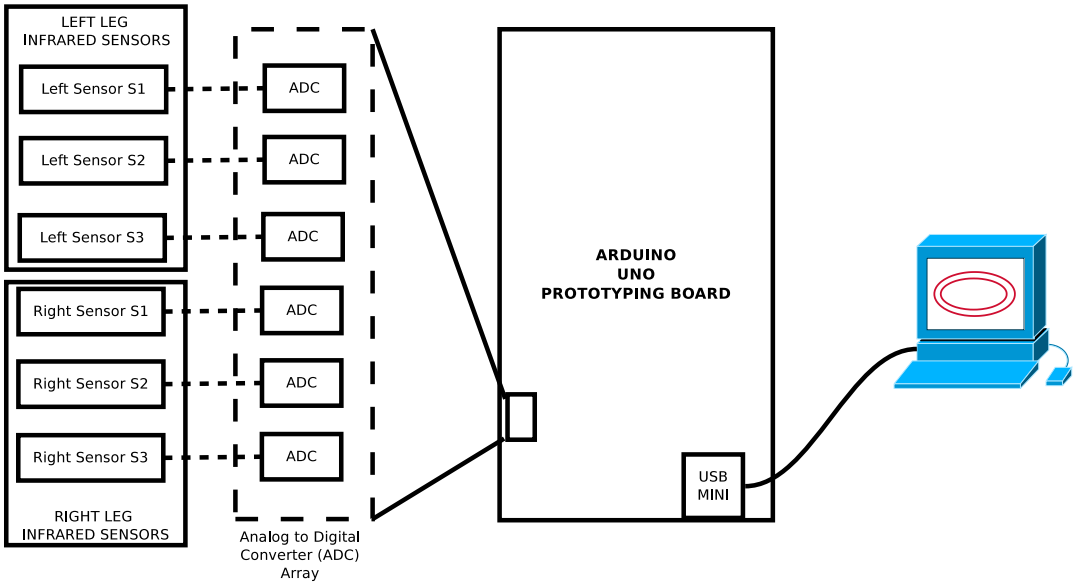

Figure 22.3: Flowchart of hardware system

Software

The software architecture is explained as follows: The software reads the data available at the analog input channel of the microcontroller and computes the formulae. Since the microcontroller has a single core, sequential instruction execution, we first read all inputs, compute all formulae and then store and display data and identify status of the user as displayed in the figure 22.4.

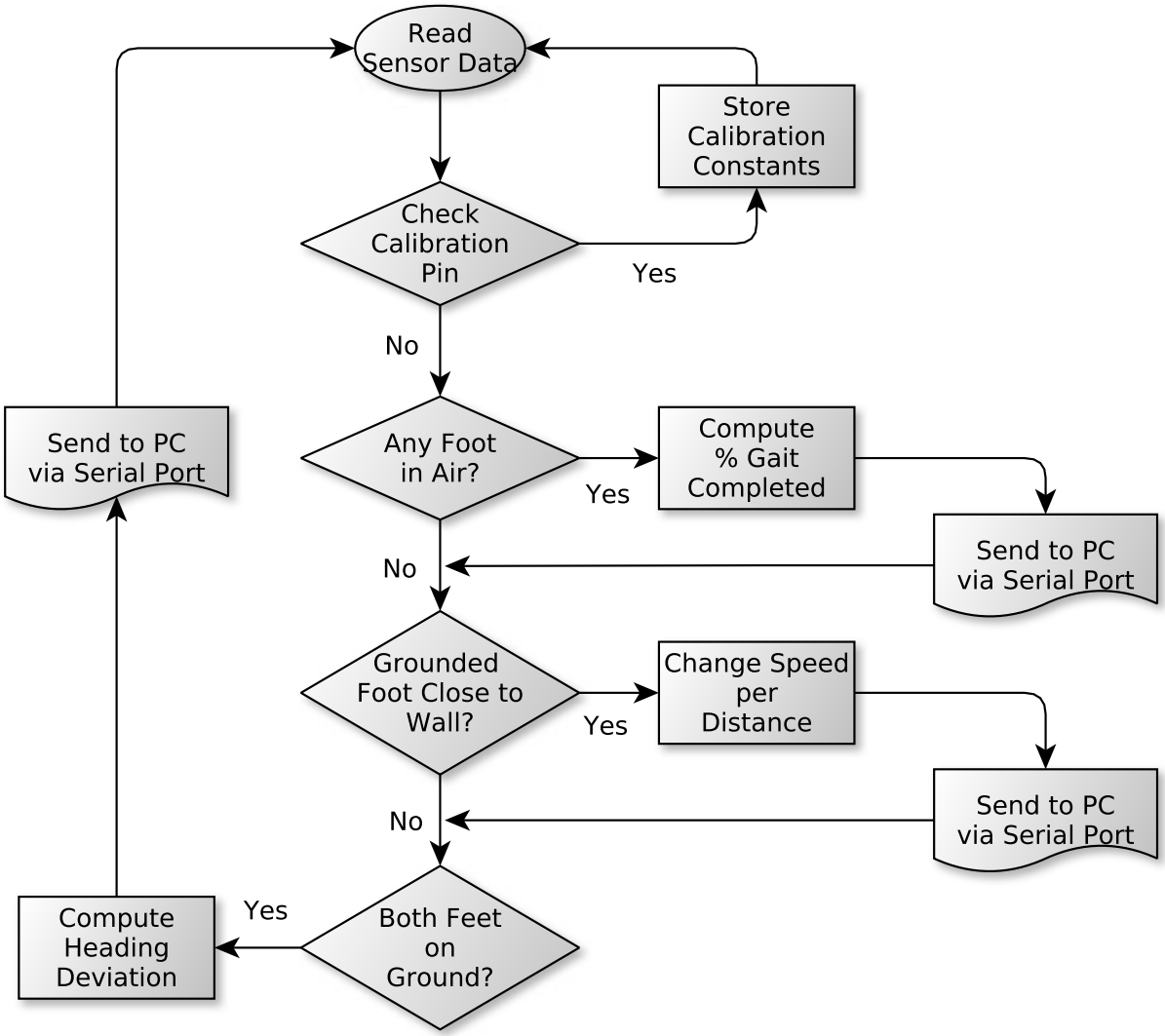

Figure 22.4: Flowchart of software system

22.4.2 Hardware Construction

The hardware has been constructed using readily available infrared sensors, arduino uno prototyping board, wireframe to hang the sensors on a foot and some cardboard boxes for localized location reference.

Installation Positions

Positioning sensors is always crucial in a setup as they minutely affect the calibration for the system software. For this setup also sensors required special installation positions. Sensors were placed at following positions (figure 22.5):

- In the gap between feet and ankle in the subjects shoe.
- At the toe of the shoe, facing outward, aligned perpendicular to the foot
- At the ankle of the shoe, facing outward, aligned perpendicular to the foot and parallel to toe sensor.

Circuit diagram and Components

The following components were used for the construction of the system. These components constitute the data acquisition and processing modules from the flowchart (figure 22.1).

1. IR Sensors, part number
2. Arduino UNO board

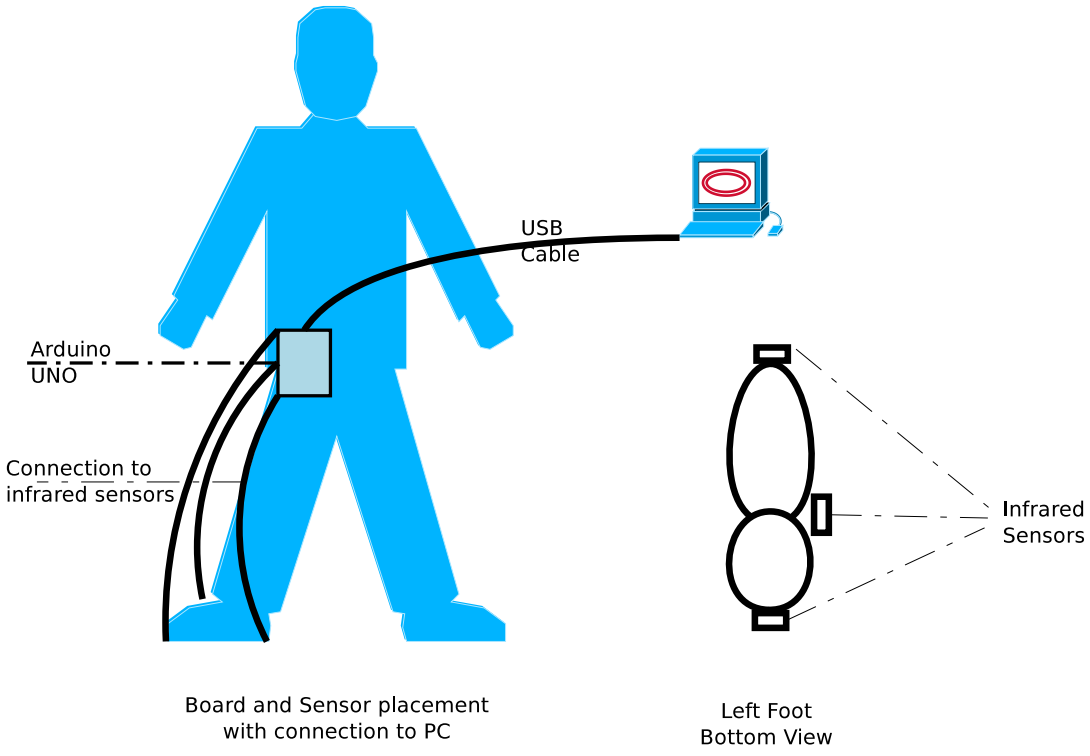

Figure 22.5: Placement of sensors

3. 1 usb A to B cable for Arduino

4. A wireframe to hold sensors.

5. Front and rear reference panels

6. Scotch tape and connecting panels

The infrared sensors give an analog signal as output, which need an ADC to convert to digital value which can be used for execution of decisions in the microprocessor. This ADC is included in the Arduino prototyping board. Thus it completes the necessary requirement. Similarly, the board is required to be connected to a PC for display and storage of values. This has been accomplished via usb connection to

the computer system which displayed the data in a text window.

Positioning of sensors

Setup of a sensor system is an important step as it affects the calibration values which in turn will affect the calculations via code and hence affects the recorded values for gait conversion. For this reason a wireframe was designed to be hooked into any shoe of any size. The sensors were able to slide and the ideal condition is depicted in figure 22.5.

The sensors at toe and heel are required to be parallel and facing outwards, while the sensor at the side of the foot is facing down and may or maynot be perpendicular to the plane of sensors at toe or heel.

22.4.3 Software Design

There are two parts to the software design viz data acquisition and processing and frontend display of information. The software was written for Arduino and the output was observed on the serial console of a computer for the same. This output to serial port was then used in another software for display of calculated heading and speed of the subject based on theory discussed in previous chapters.

Platforms used

The platform used for programming the micro controller is called Arduino and the display platform used is called Processing. Both the programming platform are de-

veloped in java but the code developed in them follows C programming style and structure. The installation of these platforms is very simple and detailed instructions can be found at [ref] and [ref].

Files

The code for the software developed is attached in the appendix. It is divided into three primary parts as calibration code for microcontroller, running code for the microcontroller and frontend visualization.

Calibration

The hardware sensors do not behave same during their liffetime. Moreover two units do not always give the same output The calibration algorithm is written inside the microcontroller along with the computational code. For process of calibration, a switch determines the state of execution for the code. This is implemented on microcontroller by setting PIN# as "HIGH" for calibration mode, and low implied the regular data acquisition and procesing mode.

The calibration mode is to determine the initial point of reference as well as the range for the data acquisition by defining central position of the subject and stopped feet condition when both are resting on ground. Once the subject is in position

as shown in figure 22.6 while the calibration pin (PIN#) is switched to LOW, it calibrates the sensor values to the existing situation. It sets the minimum value for the bottom foot sensor and sets the center between the front and rear panel.

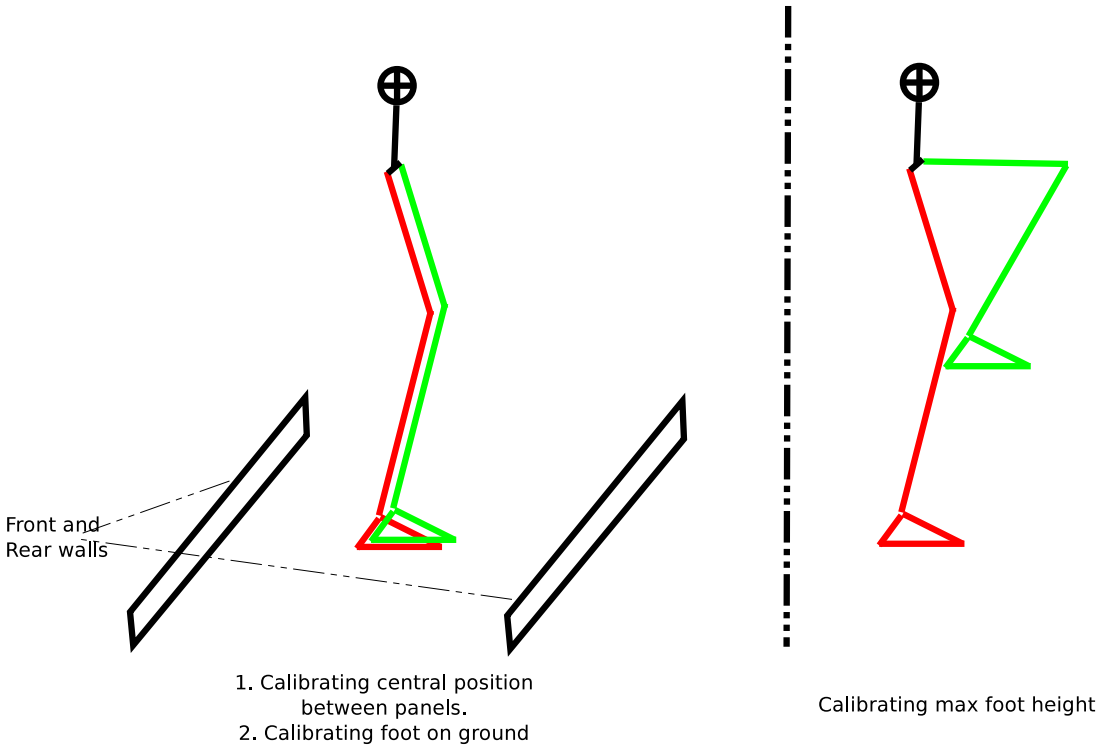

Figure 22.6: Calibration Position

22.5 Limitations

After completion of this process, three main limitations were observed which can create problems in the pedestrian system.

1. The downside of this procedure is to calibrate the setup for every subject manually.

2. The detection of turn is not natural and requires push button input.

3. The system is bulky and clunky, and requires carefulness even during operation.

22.6 Testing and Results

This system was attached with shoes of a subject and then the results were obtained on the screen. Since this was a prototype, not much emphasis was given on visualization, except raw data on console.

Eventually the result was that we were able to successfully capture the walking behavior of person performing a walking gesture using a microcontroller and sensor based system. These parameters were enough to model a human gait.

There were two main conclusions drawn out of it:

1. We can easily extract the information and employ them to generate gait parameters; therefore, the features have been correctly identified.

2. An attached sensor based system is difficult to calibrate and monitor. Also such system will take time to setup for every subject. Moreover the system becomes clunky due to involvement of multiple wires. Therefore such systems are impractical in nature.

CHAPTER 23

IMPLEMENTATION USING NATURAL INTERACTION

This chapter studies about natural interaction, its benefits and the implementation details using ms kinect for the purpose of implementation of pedestrian simulator.

23.1 What is natural interaction

In general, our method of communication is through gestures, expression and movements and which may also be aided with audio or speech. In this process they may interact with each other or with the environment. There is no need to wear any devices or learn some new instruction, it is completely intuitive and is considered natural mode of interaction with environment. Thus Natural interaction is defined in term of experience as

> People naturally communicate through gestures, expressions, movements, and discover the world by looking around and manipulating physical stuff; the key assumption here is that they should be allowed to interact with technology as they are used to interact with the real world in everyday life, as evolution and education taught them to do [25].

23.2 Why is natural interaction required

Natural interaction capable devices have a number of advantages to traditional button and sensor based interfaces as follows:

1. They eventually remove complicated user interfaces.

2. They are easier to maintain.

3. They are considerably lower in cost

4. They, above all are intuitive, hence have a really fast learning curve

Natural Interaction is a design methodology for the next generation devices and physical interaction spaces, and provides capability to general users for using what is intuitive and leave the bulky and complicated interfaces. Mostly this in itself is a research topic which has been researched heavily for many different usage scenarios under the subject of user interaction design. We have identified and covered those during earlier chapters.

23.3 Available options

Currently market has two major options available capable of defining natural interactions viz, Microsoft Kinect and Nintendo Wii. One third option is development using a video camera and OpenCV based solution. Effectively this brings us to a situation of selecting the most cost effective and durable device among the available options.

23.3.1 Why MS Kinect

are available in market currently To identify the device fitting our requirements, Table 23.3.1 shows a comparison on list of features among the three alternatives.

Features	**MS Kinect**	**Nintendo Wii**	**OpenCV**
Sensor type	Multiple Video Camera	Accelerometer	Video Camera
Controller	Not required	Atleast one controller	Configurable
Setup	Plug and play	Plug and play	Various configurations
Depth Perception	Available	Not available	Algorithm & hardware dependent
Development Support	Excellent	Moderate	Open source community
Special	Skeleton & face detection	fast movements	none
Precision	High	High	medium
Speed	High	High	Slow

Table 23.1: Comparision Table of Available Products

Based on the above table, we selected MS Kinect because of the various features available as well as simplicity in setting up. Moreover the Kinect is backed up by software giant Microsoft, hence a good support both online and community is at disposal as well as a trouble free sdk is available for use. Moreover, Kinect is a commercial grade product aimed to provide many aspects of game development for the developers. This opens a wide window of opportunities for a better product in every aspect.

23.4 Setting up kinect

This section gives a technical walkthrough of how to install a kinect on a PC and then how to install the kinect for pedestrian simulator.

- Setting up Kinect Sensor: Open the kinect from the box.

  Attach the power cord to sensor and main power supply.

Figure 23.1: Out of the box

Mount the sensor on a stable surface above or below the display.

- Install Kinect SDK

  Download latest Kinect SDK from Microsoft Kinect for Windows website http://www.microsof

  us/kinectforwindows/develop/developer-downloads.aspx

  Go through the guided installation steps. Please note that it works only with

Figure 23.2: Power Cord

Figure 23.3: Mounting the Kinect Sensor

Visual Studio 2011 and upwards. connect the usb to PC and wait for all the sensors to get installed

Done, The Microsoft Kinect is set up and ready to use.

23.5 Kinect Interaction Space

The interaction space is defined as the area in front of the kinect sensor where the infrared and color sensors have unblocked view of everything in front of the sensor. The Assumption is that the lighting is not too bright and not too dim, and that the objects being tracked are not too reflective. While a Kinect is often placed in front of and often at the level of a users's head, the sensor can be placed in a wide variety of positions [14]. Eventually this is defined by the field of view of the kinect cameras. Moreover this is also supported by tilt angle which greatly increases the interaction space. This tilt angle is controlled by a motor inside the sensor. This can be visualized in figure 23.5.

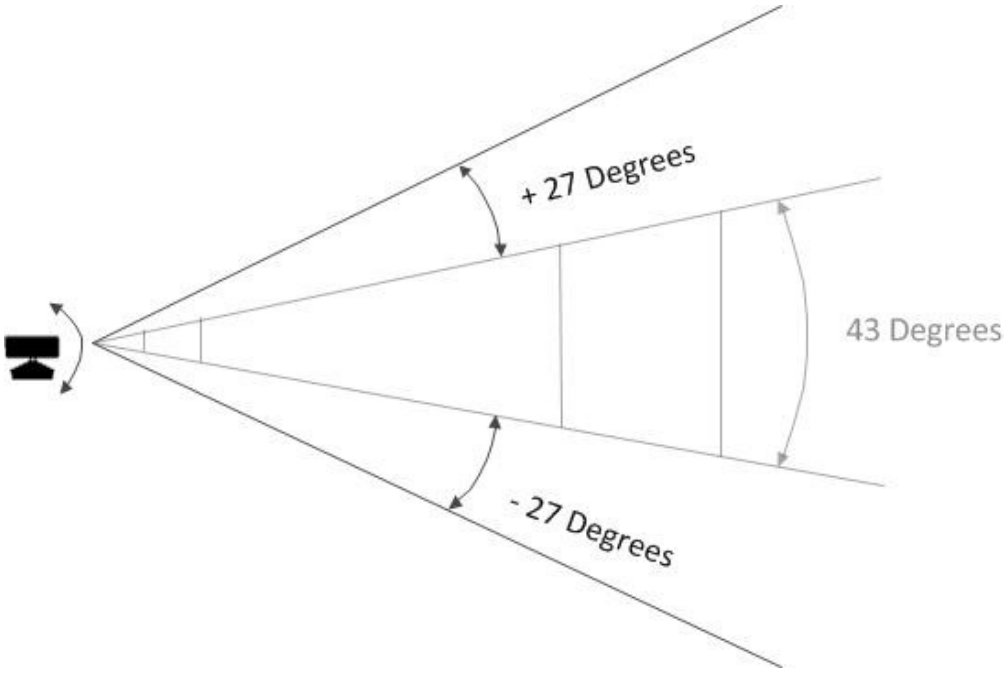

Figure 23.4: Kinect Interaction Space [14]

23.6 Kinect for Windows Architecture

The hardware software interaction using a kinect sensor can be visualized as shown in figure 23.5. Kinect is a complex device and provides video image, depth image(using IR camera) and audio signal. These are interfaced with Natural User Interaction library provided by Kinect SDK, which is eventually used by the application to perform programmed actions. The Kinect for windows SDK has three broad types of compo-

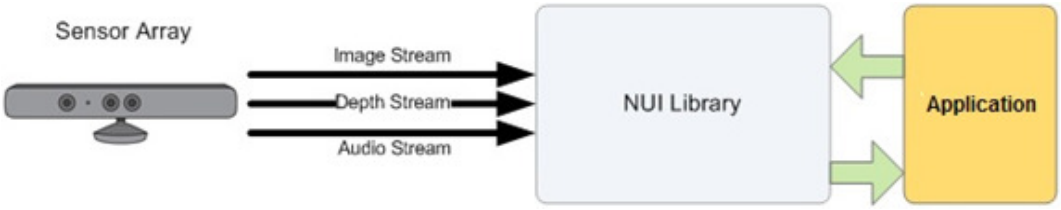

Figure 23.5: Kinect Architecture [15]

nents which can be visualized in its architecture and is shown in figure 23.6

These components include the following:

1. Kinect hardware - The hardware components, including the Kinect and the USB hub through which the sensor is connected to the computer.

2. Kinect drivers - The Windows drivers for the Kinect, which are installed as part of the SDK setup process as described in this document. The Kinect drivers support:

    - The Kinect microphone array as a kernel-mode audio device that you can access through the standard audio APIs in Windows.

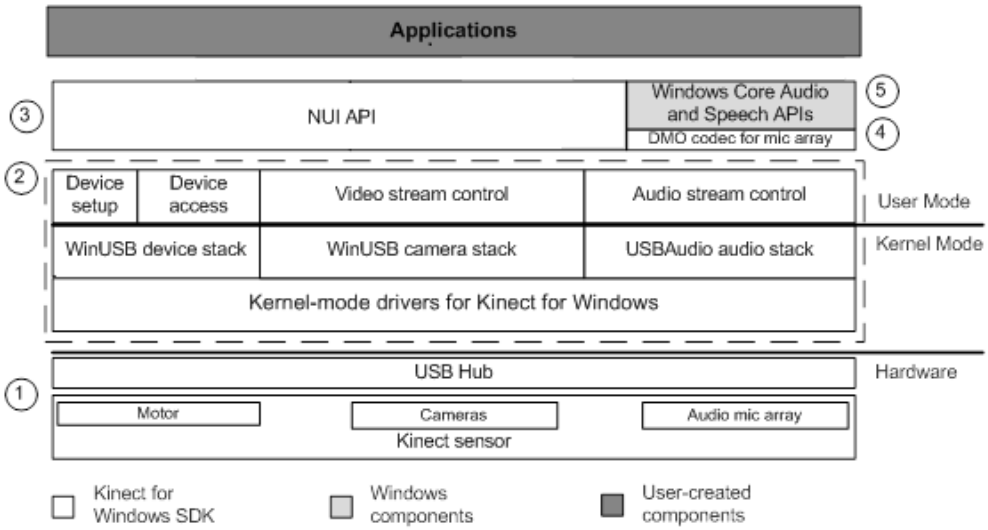

Figure 23.6: Kinect SDK Components Architecture [15]

- Audio and video streaming controls for streaming audio and video (color, depth and skeleton).
- Device enumeration functions that enable an application to use more than one Kinect.

3. Audio and Video Components Kinect Natural User Interface for Skeleton Tracking, Audio, Color and Depth Imaging

4. DirectX Media Object (DMO) for microphone array beam-forming and audio source localization.

5. Windows 7 standard APIs - The audio, speech, and media APIs in Windows 7, as described in the Windows 7 SDK and the Microsoft Speech SDK.

23.7 Programming Models

[the programming architectures followed by SDK] Kinect operates using two cameras viz RGB and Infrared. Each camera is composed of a Charge Coupled Device (CCD) which acts as a screen to capture an image and transfer the relavent bits and bytes of information for processing. Each such snapshot of a subject in front of kinect sensor is called a frame.

Application using Kinect sensor retrieves image frames from it. The process of retireval involves requesting a frame from the sensor where each frame is passed to a buffer whenever requested. A frame is passed only when it is completely captured inside the CCD of sensor. The sensor data stream never provides the same frame to the application more than once. To get the frame there are two programming models followed viz Polling Model and Event Model [13].

23.7.1 Polling Model

It is the simplest model for reading data frames. The application opens the image stream and then requests the frame after every predefined time interval. The request method returns a new image frame when it is ready or at the expiry of wait time, whichever comes first.

On successful return the new frame obtained is ready for processing. If the time-out value is set to zero, the application code can poll for completion of a new frame while it performs other work on the same thread [13].

23.7.2 Event Model

In this approach, the code passes an event handle (a pointer to a function) to image retireval method from SDK. Its processing can be visualized as when the image is ready to be read from the buffer of the sensor, it is termed as an event and the associated function in the code is called for processing the data. During this data processing, the event is reset by the NUI Image Camera API.

The event model in Kinect SDK supports the ability to integrate retrieval of a skeleton frame into an application engine with more flexibility and accuracy [13].

23.8 Software architecture

Objective of pedestrian simulator is to track a walking human gesture and then process data to draw inferences for creating simulation of a walking human. This code structure has the flowchart as shown in figure 23.7 below. The architecture primarily consists of three layers Data Extraction, Data Processing and Event Generation where each of these semantic layers specify a classification of objectives addressed in the code structure:

23.8.1 Skeleton identification

Natural User Interface (NUI) library provides a built-in class to extract skeleton related information as the embedded algorithms inside the Kinect sensor can detect and track two human skeletons at the same time.

In our scenario, we always assume that we are tracking only a single skeleton at

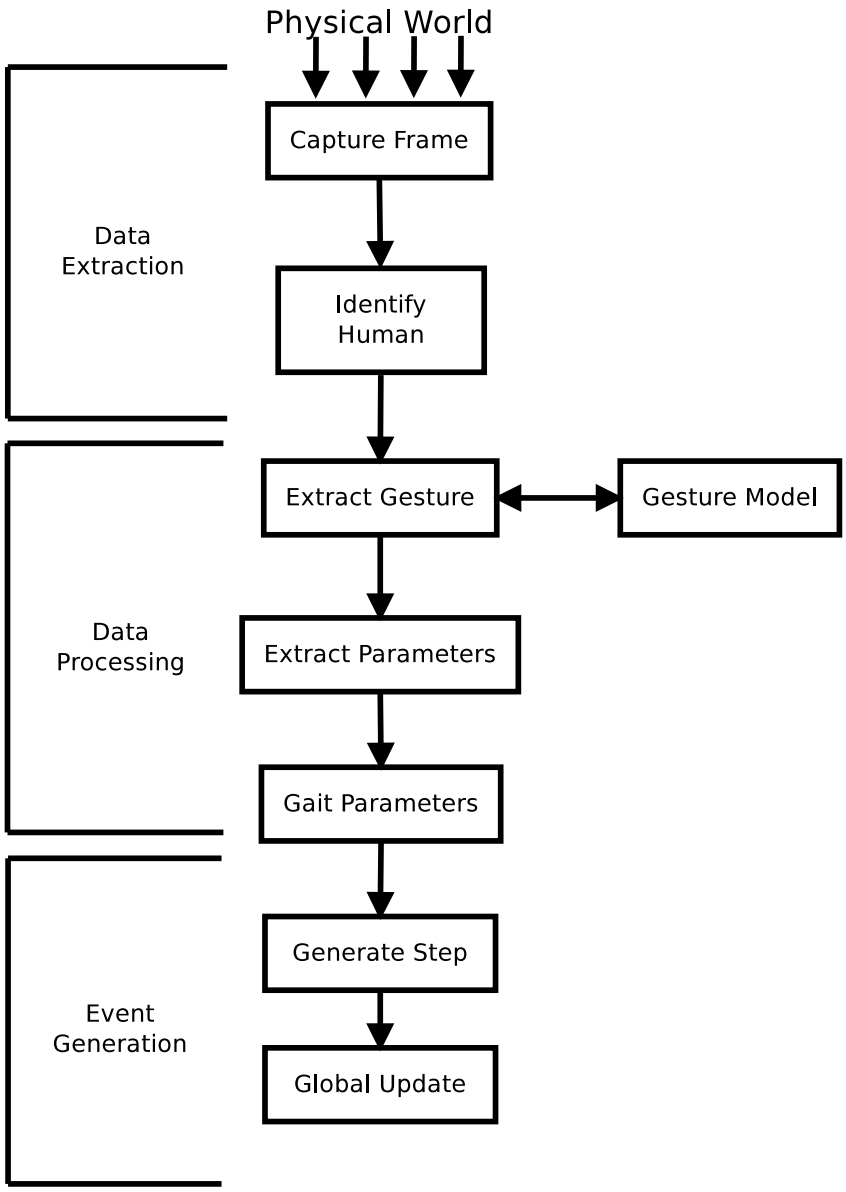

Figure 23.7: Pedestrian Data Extraction Architecture

a given time and there are no more skeletons present in the picture. The Kinect sensor sends a signal and the skeleton data every time it confirms one and finishes the computation for it. On this signal skeleton data is read and then transferred to the data reading function where gesture identification is done and appropriate information is extracted.

23.8.2 Gesture definition

The skeleton data is actually tracking a set of points mapped in two dimensions and providing real time location of 21 joints. Each of these joints is accessible by using a macro defined in the SDK and hence are easily tracked. As discussed in chapter 2 and 3, we defined the gesture to be an on the spot walk to map the parameters for a human walk. It is this gesture that we have defined earlier, is what we want to extract.

We also define an intent to walk occurrence as a small lifting of the foot as to the situation where the subject might be thinking of taking the next step but then stops right in the middle of it. If he brings it back before a certain position or height, we can call it intent to walk and ignore to avoid the subject taking an unintentional step.

23.8.3 Feature extraction

This on the spot walk gesture is defined by the knee movements up and down where their height determines the step length and the frequency of up down determines the speed of the pedestrian. These features are extracted from the gesture by continuous

monitoring of the knee joints from the available skeleton data. These values are then de-noised to remove false positives. Also the intent to walk occurrences are also detected and ignored by not triggering and changing the values for gait parameter conversion.

23.8.4 Gait parameters conversion

The feature extraction from the skeleton data is used to convert to gait parameters. Recall that a gait is defined as the pattern of movement of limbs. For this pattern to continue, we identified the parameters in chapter 2. These parameters of an actual gait are then related to the on the spot walk gesture and the relations were introduced in chapter 2.

These generated parameters are continuously updated in the walking pedestrian model (rolling disc model) to generate current state of the pedestrian which will be required for a detailed graphical modelling.

23.8.5 Step event generation

When a subject takes a step, it is characterized by one full lift of the foot and then back to the resting position. Therefore we call that a subject has taken a step only when the foot goes up and down, and this generates a step event. A step event marks the completion of feature extraction and then triggers the computation of the subject's global parameters like current speed, location etc.

23.8.6 Global parameters conversion

Global parameters are those which define the state of individual pedestrian in the complete visualization world. These parameters are speed, direction of heading and current location. They effectively point a state of pedestrian but are updated by the computed gait parameters over time.

23.9 Conclusions and limitations

[for this approach] This approach makes the development of this simulator modular and thereby allowing it to be integrated in any software and application framework. However this architecture has limitations as follows:

- Cannot observe direction of motion using Natural Interaction.
- Requires calibration for each new user.

In further chapters we attempt to address these limitations.

CHAPTER 24

NAVIGATION ON 2D PLANE

Kinect is a versatile sensor which can be used in multiple scenarios. In this thesis, it was used to extract information about the gestures performed by a subject to help them navigate through a virtual network. The algorithm discussed in previous chapter allows to extract information related to human gait. But navigation in 2D plane is an important aspect and such a framework should be able to address it. In this chapter methodologies are discussed to be able to perform the same.

24.1 Current Limitations

Movement in single dimension or a curve implies motion in forward or backward direction. But on a two dimesnional plane it also requires taking perpendicular turns or turning on a curve. With these things in perspective, the current framework is able to capture the walking gesture, but is incapable of the following

- Inability to distinguish between walking backward and forward.
- Taking left and right turns during walking gesture.
- Taking a graduated turn along some curvilinear path.

It is imperative to have a solution to solve these problems in the framework, which otherwise will be fall short of providing access to multiple possible paths and motions. Therefore following three methods have been identified:

- Method 1: Using multiple(two) kinect sensors to track
- Method 2: Tracking decrement in body constants (hip width, etc.)
- Method 3: Usage of secondary gestures with hand.

24.2 Method 1

Kinect for Windows support multiple kinect sensors to be connected to a same system, being governed by the the same application. This allows for many wonderful and new interaction spaces that can be thought of, hence, leading to new application architectures that can be applied to many problems.

In case of recreating two dimensional motion, inside a virtual network with the inputs by the subject, it is important to detect the position and degree of rotation(turn) taken by the subject. Since motion is two dimensional, it is intuitive to keep the two kinects perpendicular to each other as shown in figure 24.1.

This method will help in generating more accurate detection of how much the subject has turned, as evident from the figure 24.1. This is accomplished with skeleton detection and face detection from the video stream from the kinect sensor. This

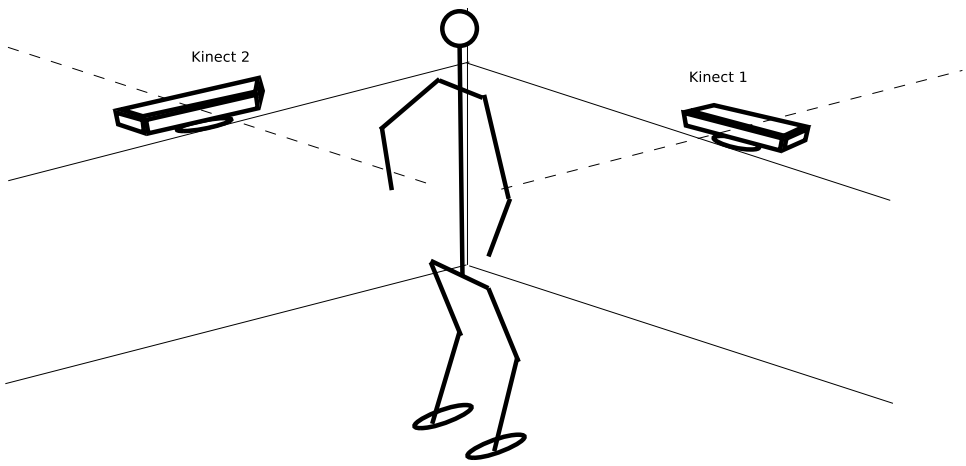

Figure 24.1: Perpendicular kinect sensors scenario 1

will help to identify that towards which kinect sensor is the subject facing and subsequently feature extraction will be from that sensor. Although this approach will also suffer from the problem of differentiating between forward and backward motion.

Another arrangement for perpendicular placement of two kinect sensors can be seen in figure 24.2. This arrangement will again help in generating a more accurate detection of the turn taken by the subject. As seen in figure 24.2, in this approach one kinect sensor is mounted over the head facing the floor. This kinect sensor will be used to identify the direction in which the subject is facing by using a directional marker on their head. This directional marker can be color coded, or shape coded. Also it is not necessary to use a kinect sensor over the head, and another video camera can be placed for the same. But since kinect sdk is being used already, it would greatly simplify the programming and hence using a kinect sensor is more sensible approach.

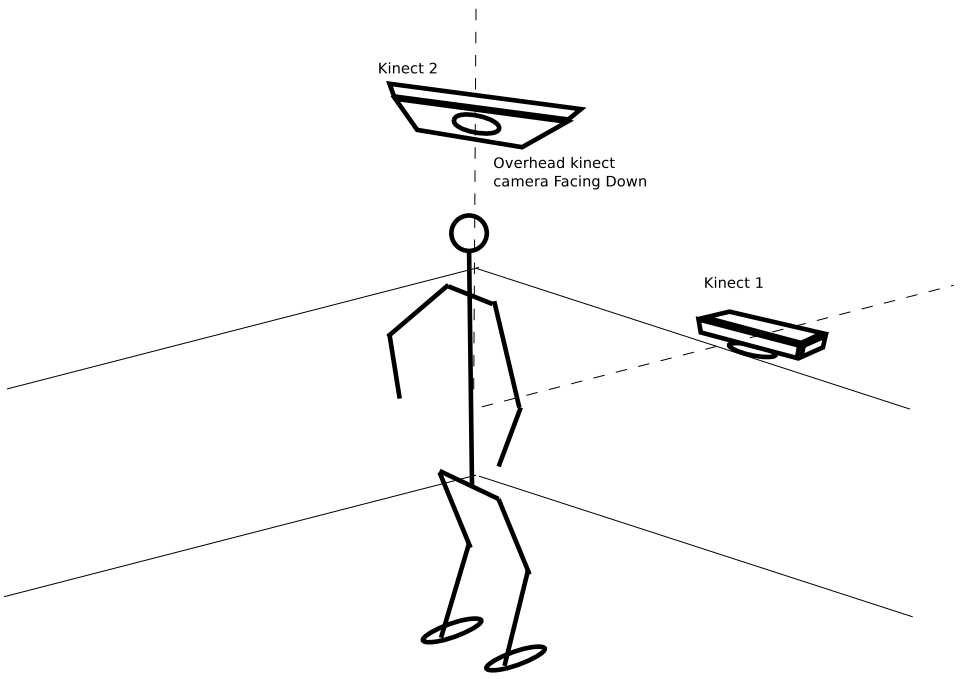

Figure 24.2: Perpendicular kinect sensors scenario 2

The limitation of this method is more practical in nature. Due to involvement of multiple kinect sensors, the setup of the interaction space will be difficult. The sensors will have to be placed accurately and calibration will require repositioning them. Hence the setup of this will require a lot of time and will be susceptible to any accidental movement of the sensors.

24.3 Method 2

For every subject, while performing on the spot walk gesture in front of kinect sensor, the plane containing the shoulder and the hip region (plane of subject) is parallel to that of the sensor (plane of sensor). This information can be used to define whether the plane of subject is parallel to the plane of sensor. Therefore, the

hip and shoulder width observed is a result of taking a projection of the plane of subject on plane of sensor (refer figure 24.3). This will accurately define the tilt of the subject with respect to the sensor and hence the direction of movement.

To check whether the subject is moving towards the sensor or away from the sensor,

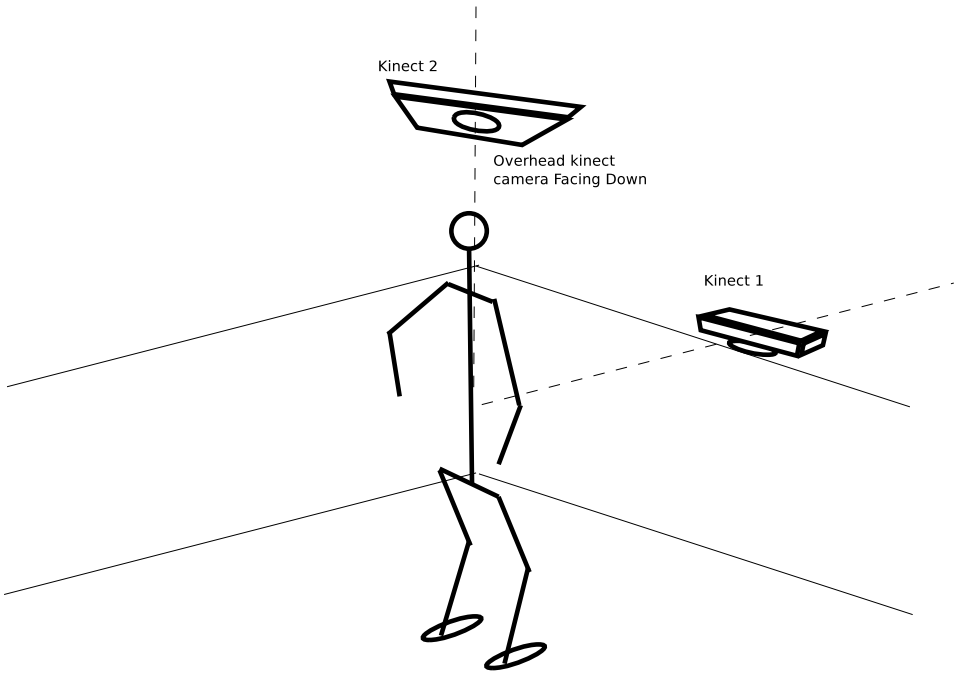

Figure 24.3: Geometry of relating body constants

face detection can be used. If the face is found in the video stream, it will imply that user is moving in a specified direction in the virtual world, let it be North. In case if is not found and the projection value of shoulder width is nearly same as that when the planes are parallel, the subject is assumed to be moving South.

This points to the limitation of this approach as it becomes difficult to point whether

the subject is moving east or west while turning. In other words left and right turns cannot be distinguished. An alternative solution to this limitation can be wearing colored markers perpendicular to the plane of subject, placed on each shoulder.

24.4 Method 3

With the advent of gesture based control, each interaction is nowadays thought of in terms of gestures. Similarly we have defined and used an on the spot walk gesture for obtaining information for the gait parameters for any person. Hence, it is intuitive to define another gesture for turning while walking.

In this method, a new set of gestures is defined as follows (and seen in figure **??**):

> To turn left, raise the left hand above torso. The turn command will be relayed till the hand remains above torso, in constant increments. This will be independent of whether the person is performing on the spot walking gesture or not. Similarly, the gesture for right turn is defined.

Therefore, this method will allow to turn with ease with a limitation on the speed of turn. A way to dynamically alter the speed of turn as per the requirement of subject is desired. Also, such a method constrains a subject to behave in a non-intuitive manner as it is more intuitive to turn left yourself, rather than raise hand to turn.

24.5 Selected Final Method

After the above analysis, a number of issues for selection of final method surfaced. The primary being ease of deployment and complexity of setup, followed by ease of development and number of kinect sensors. Also this linked to an indirect part of problem, the calibration. Calibration is most important procedure for any system as it makes the system adaptable to varing conditions.

Based on these issues, method 3 was selected as it was easiest and most robust implementation strategy. Although this method is less intuitive in nature, but given the increase in cost and complexity of the system in other two methods, it seems to be a worthy trade-off.

CHAPTER 25

FINAL IMPLEMENTATION

This chapter details the development and execution of the final software framework, based on the research detailed in previous chapters. At the end of chapter suture work and limitations are also discussed.

25.1 System Architecture

This project is aimed at development of an application programming interface (API) for user interface engine of a pedestrian simulator, to allow a subject navigate through a virtual network. The output is aimed to be generic, returning various gait parameters in realtime, the number of steps taken and turning values.

As mentioned, the api has to be realtime, maintaining total and instantaneous parameters. For this the system architecture is as shown in figure 25.1. This architecture has been explained in following subsections. updates various object parameters. Such parameters are called instantaneous parameters.

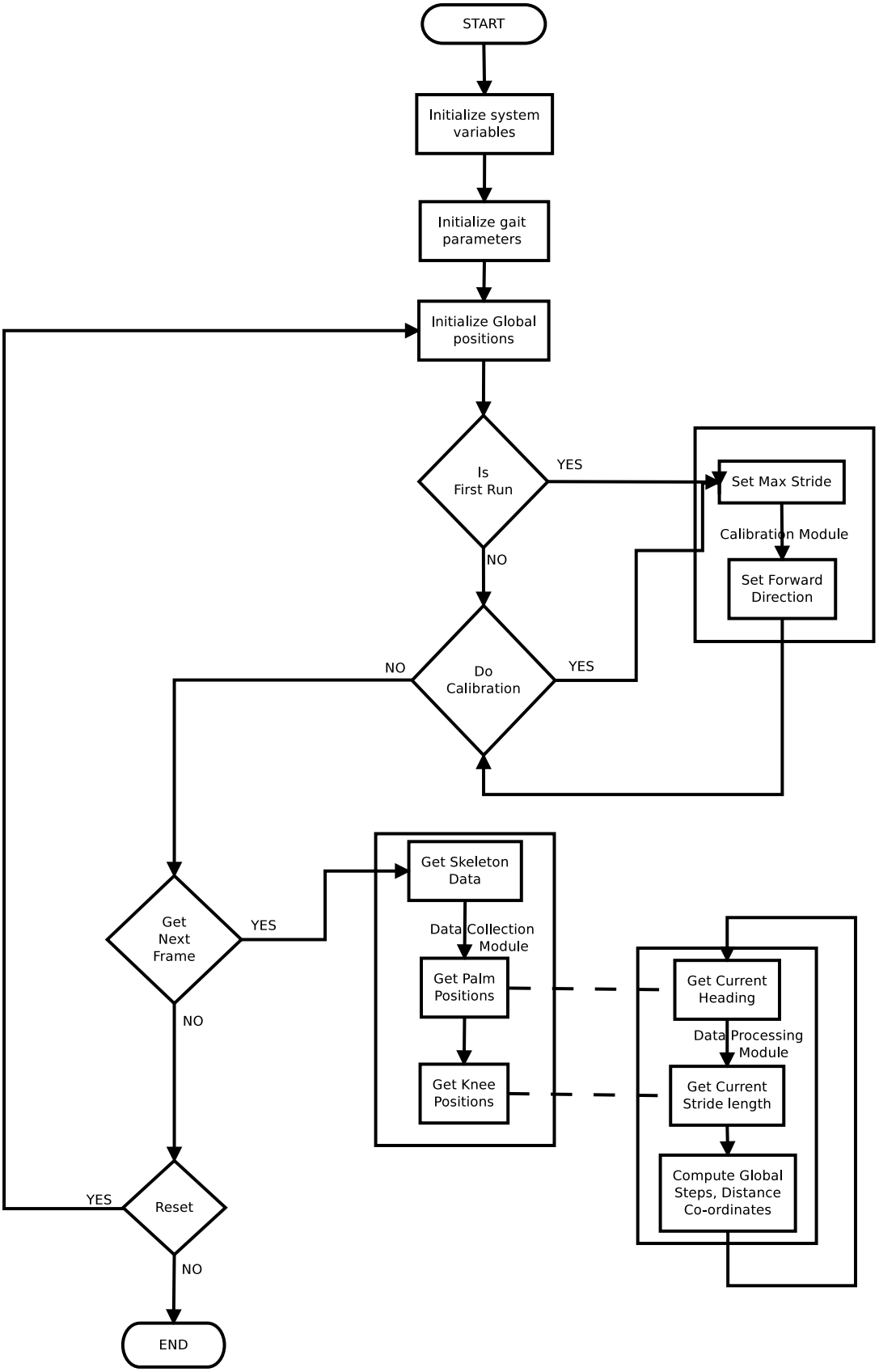

Figure 25.1: Complete Architecture of Pedestrian Simulator API

25.1.1 Intitialization

This part of the API architecture is called to initialize various functions necessary for tracking skeleton and hand genstures. In this function a thread is initialized which communicates to NUI SDK for Kinect sensor to update skeleton data. This function also initializes another thread which updates current values for a pedestrian object from the seleton data.

This initialization is a part of the constructor for the object. A constructor is that part of the program which is invoked on creation of an object. This part ensures that default values are loaded into the variables inside the pedestrian object and avoid the problem of loading garbage value in them. Hence it ensures smooth functioning of the program.

25.1.2 Calibration

This part of the API architecture is called immediately after initialization or at any point during processing to define certain setup dependent variables for computation of various gait parameters. These variables are shoulder width, hip width, maximum and minimum knee height and wrist height from ground.

This method ensures that subject of all body types can participate and navigate. This ensures the independence from height, width and similar body features. Hence it allows for all kinds of subjects in every age group. It also allows to switch subjects during a single run if needed.

The calibration process is guided in nature, and hence the API will only support

calibration on a handshake type protocol. The method to communicate with user has to be built around it.

25.1.3 Get Current Gait Values

A gait has various parameters which can be called as instantaneous parameters. These define the current phase of the gait for a subject. The mapping can be seen in figure 21.5. Using the value returned from this module, the graphics engine can represent the status of the subject in one of these phases.

This module returns the current phase of walking i.e. stance or swing phase and the percentage completion of that phase. Therefore a suitable graphics engine can render appropriate views in the 3D environment for the subject.

25.1.4 Get Pedestrian Data Values

A subject while walking has many important secondary data points. These include, but are not limited to, instantaneous speed, a complete step taken, deviation from geographic North of virtual environment etc. This data is required to be fetched at every point of time while the application is running. Therefore, it is important to keep this method in a thread associated with graphics.

25.2 Test Case Implementation

To employ the above mentioned API, a text based user interface was implemented which printed various parameters on the screen. The following screenshots show the

implemented solution using the pedestrian API built during this project.

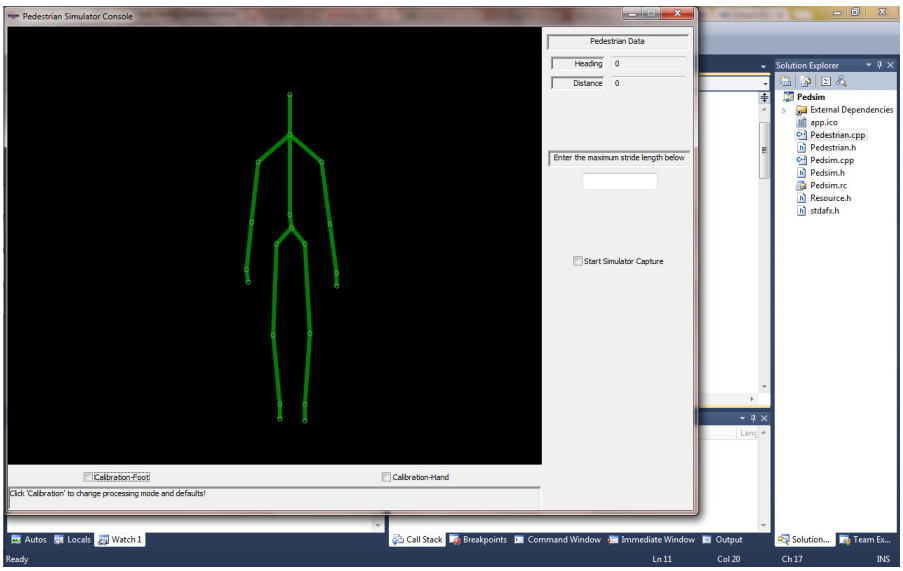

Figure 25.2: User in front of Kinect Sensor

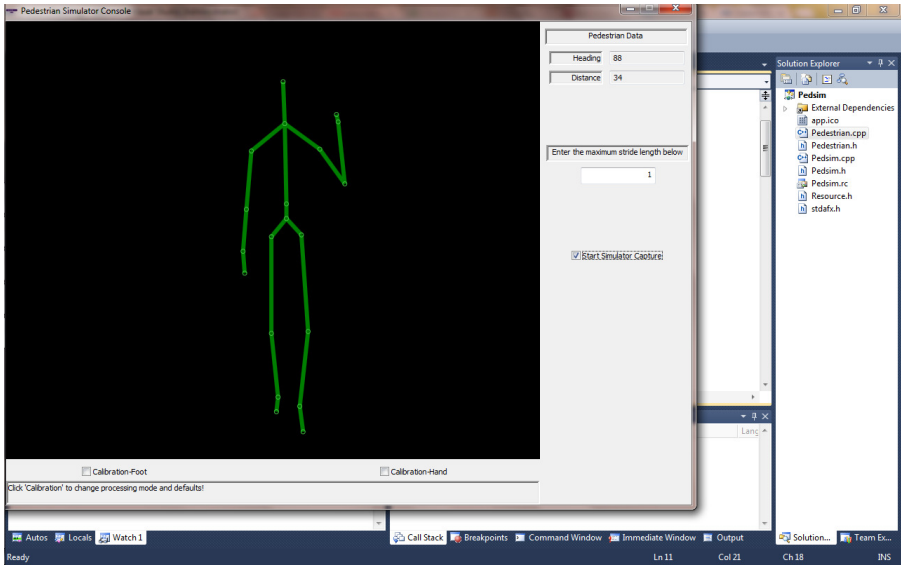

Figure 25.3: User in changing heading

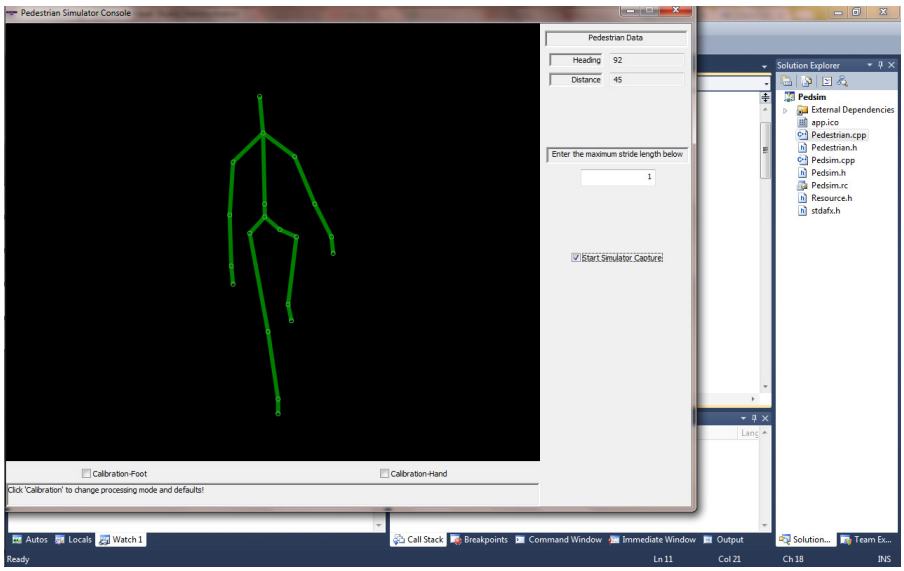

Figure 25.4: User stepping and increasing distance travelled

CHAPTER 26

DISTRACTED DRIVING

26.1 Introduction

The reason behind the majority of the roadway crashes in the United States has been attributed to so many different factors such as driving under influence, speeding, distracted driving, fatigue etc. But all these various factors converge to a single behavioral attribute, i.e. inattention as shown in figure 26.1. All the above mentioned factors affect drivers reaction time which in turn increases chances for accidents and roadway fatalities.

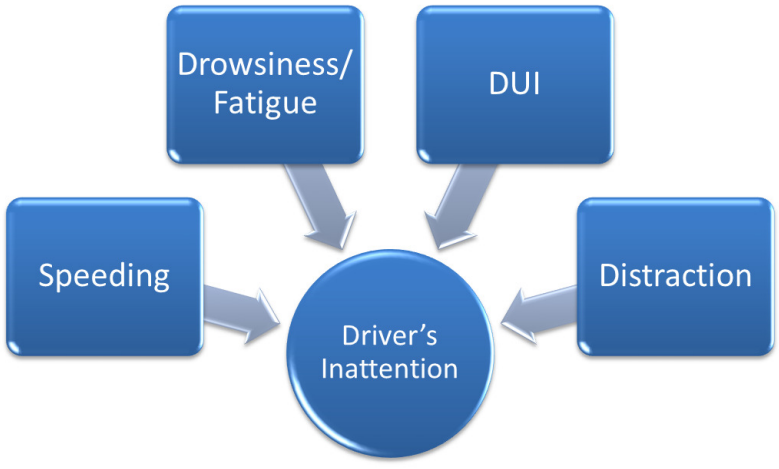

Figure 26.1: Accident contributing factors leading to inattention

Several studies have been conducted in the past to model driver's road behavior based on the road performance and the parameters obtained from the vehicle. This chapter explains in brief about the studies that have been conducted so far.

The influence of driver's impairment in motor vehicle crashes is already well established. To analyze driving profile and to be able to predict potential dangers based on the driving pattern, vehicle based sensing techniques have attracted significant attention. The underlying idea is to capture data through various sensors embedded in the vehicle and then to use data processing techniques to characterize the data and identify different levels of impairment and then to find a relationship between the levels of impairment and driver's data characteristics. Depending upon the type of sensor used to capture data, driver impairment detection and monitoring technique can be classified as follows:-

- By capturing driver's physiological signals such as Electroencephalogram (EEG), Electrocardiogram (ECG), eye tracking, postural stability etc. one can get significant and highly reliable information about the driver's current state of mind. Various studied have been conducted to monitor driver's performance by capturing these signals and thereby studying distraction. As these signals are directly obtained from driver, the information about driver's physiological condition is very rich. However these sensors are highly sophisticated and are not so commonly found in existing commercial vehicles, it becomes difficult to employ the setup without posing additional discomfort to the driver.

- Signals obtained from vehicle motion such as vehicle speed, lateral position and time headway are widely used in various collision avoidance systems. In addition, these signals have also been used to model and monitor driver's behavior behind the wheel. Sezikawa have used distance between cars on the road to model human driving behavior. Similarly Toledo used lane changing information and acceleration rate to model human driving behavior behind the wheel.

- The most readily available signals are the driver's input such as steering wheel angle, gas/brake pedal activity etc. These signals are most easily available in most of the modern vehicles. As these signals are direct output from the driver and are not affected by the vehicle dynamics, thereby act as better indicators of driver conditions compared to the earlier mentioned signals. These signals have been used in various studies as measures of driver performance measurement. Desai proposed a measure of sharp changes in the driving signal to estimate drowsiness of the driver. Similarly entropy of the steering wheel has been employed as a measure of driver workload and driver performance.

The connection between the motor vehicle crashes and alcohol has been very well established. According to studies it has been noted that when BAC goes beyond 0.08,the probability of a serious crash increases dramatically. Several studies have been conducted in the past to analyze the effects of alcohol on a driver's performance.

The main hindrance for conducting the study in actual field setting, has been attributed to the safety hazards and also due to the procedural problems. As the

alcohol intake of an individual is very specific and varies from person to person, the results obtained are relatively generalizable. From these studies it has been concluded that the performance of a driver on a divided attention task is sensitive to alcohol. Moreover, alcohol seems to cause a tunneling effect such that the information is either missed, ignored or the reaction time of the driver had increased.

CHAPTER 27

EXPERIMENTAL SETUP

27.1 Driving Simulator and Scenario Design

Driving Simulator are the most popular and safe way to carry out an experimental research as it gives a close enough real world feel with in the safe environment. In addition, it also offers low cost, high control and safety on the experiments.

A "driving simulator" is a virtual reality tool which gives a driver on board impression that he/she drives a real time actual vehicle by predicting vehicle motion caused by driver input and feeding back corresponding visual, motion, audio, preprioceptive cues to the driver. It includes a real time system to simulate vehicle dynamics, visual/audio systems to recreate an actual driving setting, an interface between the driver and the simulator, an operator monitoring system, data collectors and synchronization system. Driving simulators have been widely used for safety studies, understanding vehicle dynamics, human factor study etc.

In this study a STISIM® Drive Simulator Version 10 was used to acquire data. STISIM® Drive is a personal computer based interactive driving simulator developed by Systems Technology, Inc.. It includes a vehicle dynamics model, visual and auditory feedback, steering wheel feedback and a driver performance measurement system. To create driving scenarios a unique Scenario Definition Language (SDL)

can be used. SDL provides user with the freedom to design an arbitrary sequence of tasks, events and performance measurement intervals. The STISIM® has been used in various research projects.

The scenario simulates a 3 mile drive on an outskirts of a city with the speed limit of

Figure 27.1: Simcraft Driving Simulator

45 miles/hr. By varying the overall light level of the simulation, color and brightness of the surrounding the time and day of the simulation can be adjusted. The normal drive time is around 4-5 minutes. Figure 27.1 shows the basic setting of the exper-

iment with a driving simulator. The data was collected for three different settings which were:-

- Normal Driving
- Alcohol-impaired Driving
  - Driving with Moderate BAC (0.01-0.07)
  - Driving with High BAC (0.17-0.20)
- Driving with Distraction
  - Talking on Cellphone
  - Texting on Cellphone

To make a person comfortable with the simulator 2-3 practice sessions were provided to each participant before starting the data collection. After each driving session participants were provided with a break to relax and to avoid the study being monotonous. To remove biases, each participant was asked to perform a driving task for three times, hence making the overall driving for 15 minutes for each setting. Various parameters were recorded during each session which includes lateral distance from the centerline, steering wheel angle input, brake/gas pedal input, speed of the vehicle and speed limit of the roadway. The sampling rate for the recording was about 4 samples per second.

27.2 Subject Selection

The subjects of the study were recruited from the staff and students in Transportation Research Center (TRC). The only requirement for participating in this study was to have a valid US driving license and belonging to the age group 20-30. The main objective of selective subjects from TRC was to minimize unfamiliarity bias from the study as the subjects were already well aware about the working of the simulator.

27.3 Procedure

The entire study for one participant was conducted in one day. To avoid monotony in the study, participants were given breaks in between and the study. In addition, different settings were shuffled instead of recording three driving for one setting. To measure alcohol-induced behavior, driver's were provided with Fatal Vision® goggles with equivalent BAC in moderate (0.01-0.07) and high (0.17-0.20) ranges. Figure 27.2 shows a fatal vision goggles which were used in the study.

To study driver's performance with distraction, cellphone was used as a medium of

Figure 27.2: Fatal Vision® Goggles

distraction. Both talking and texting were used to distract the driver separately as driver's performance was recorded.

CHAPTER 28

ANALYSIS: DISTRACTED DRIVING

28.1 Data Recording

Five subjects were recruited from Transportation Research Center at University of Nevada, Las Vegas. Each subject completed six sessions on a driving simulator (i.e. one familiarization trial, one normal driving without any hindrance, two alcohol-impaired driving trials and two distraction-impaired driving trials). Duration for each trial was approximately 5 minutes. Four different kinds of data was recorded in each trial. Moreover, each trial was conducted for three times to remove any bias whatsoever. The STISIM® driving simulator recorded all information as specified in the scenario file.

28.2 Data Pre-processing

There are two main reasons for doing data pre-processing. Firstly, it enables to demonstrate that there is a relationship between the driver's performance and the driver's physiological condition caused by various measures employed during the study. The measures which are being tested here are alcohol-impaired driving and driving with distraction. Secondly, the data obtained from the study need to be

reduced so that significant conclusions can be reached. Data in its current form contains a lot of information which needs to be extracted to be able to use it further. Among the various channels of data recorded, steering wheel angle (SWA), lateral lane position (LLP), speed of vehicle (SOV), brake/ gas pedal input are the most important in the analysis of the data. The lateral lane position and speed of vehicle directly relates to the risks of being in an accident on the road while steering wheel angle and brake/ gas pedal input reflects the driving behavior. Thus the fist task is to find any relationship between these two sets of data.

28.3 Time Domain Analysis

28.3.1 Standard Deviation analysis of LLP and SWA

A time window approach is applied to the analysis as single data points do not convey much information about the data set. The raw data from the LLP and SWA channel were grouped into 4-seconds window. Each window thus includes 16 sampling points. For each window, various analysis tools can be used to identify a basic pattern among the data set. The tools can be basic statistical measures such as mean, standard deviation and histogram etc. The standard deviation was selected as a tool for this study because it tells how tightly all the various sampling points are clustered around the mean in the window. It also explains the variation of the data in a selected window, which related to the objective of the study. Hence, the mean and standard deviation of all windows are calculated and compared.

Equations 28.1 - 28.4 shows the calculation of mean and standard deviation of

LLP and SWA over a window. Here N is the length of the window.

$$\mu_{LLP} = \frac{1}{N}\sum_{i=1}^{N} LLP_i \tag{28.1}$$

$$\sigma_{LLP} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(LLP_i - \mu_{LLP})^2} \tag{28.2}$$

$$\mu_{SWA} = \frac{1}{N}\sum_{i=1}^{N} SWA_i \tag{28.3}$$

$$\sigma_{SWA} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(SWA_i - \mu_{SWA})^2} \tag{28.4}$$

The standard deviation of all the lateral lane positions (σ_{LLP}) of all windows were calculated and the windows were sorted in an ascending order based on the value of σ_{LLP}. The steering wheel angle behavior of the corresponding window (the standard deviation of steering wheel angle, σ_{SWA}) were also compared.

The mean σ_{SWA} of normal driving sessions and driving with fatal vision goggles with moderate and high equivalent BAC are shown in figure 28.1. Similarly, the mean σ_{LLP} for the respective sessions are shown in figure 28.2.

The results indicate that $\mu_{\sigma_{SWA}}$ increases when the thresholds increases. Thereby indicating at a close relationship between σ_{LLP} and σ_{SWA}. As mentioned earlier, that lateral lane position is directly associated with the risks where as steering wheel angle is direcly associated with the driver's physiological status.

From figure 28.1 it can be seen that the $\mu_{\sigma_{SWA}}$ in moderate and high BAC conditions is significantly higher than that in normal driving conditions with different

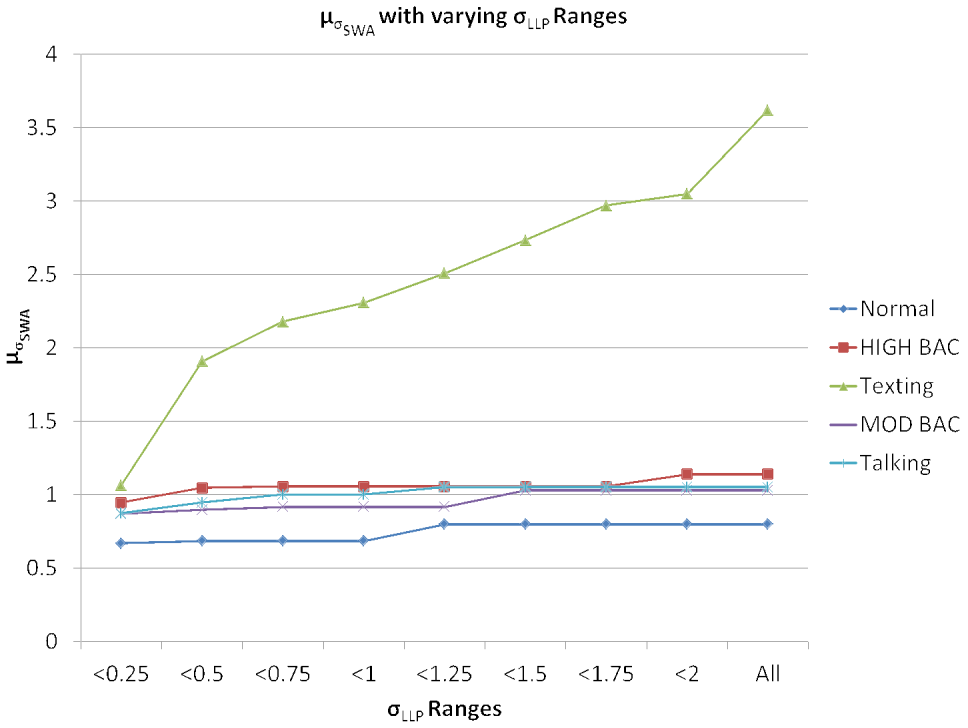

Figure 28.1: $\mu_{\sigma_{SWA}}$ with varying σ_{LLP} Ranges

thresholds. When the thresholds are smaller, i.e. the deviation in the lateral lane position is small, the driver maintains lower steering wheel movements to control the vehicle where as in the other settings when a fatal vision goggles are provided to induce the effect of alcohol to maintain the lane position, driver has to provide higher steering wheel movements. This can be explained as follows: in normal situations, subjects recognized lane deviations more quickly and made small and precise steering wheel movements to correct it. However, due to the induced alcoholic effect from fatal vision goggles, subjects had delayed responses to the lane deviations and hence required larger wheel movement to correct the lane drift. A similar analogy also exists for the driver's behavior between moderate BAC and high BAC. For high BAC, sub-

jects had to make even larger wheel adjustments to keep the vehicle in its lane. This shows that an alcohol influenced driver can maintain the lane of the vehicle (σ_{LLP} is small) and can control the vehicle successfully, but he/she has to move the steering wheel more often ($\mu_{\sigma_{SWA}}$ is large).

However, for scenarios involving distraction using cellphone (i.e. texting and talking), it can be observed from figure 28.1 that texting while driving has a very high $\mu_{\sigma_{SWA}}$ compared to the other scenarios indicating a larger steering wheel input by the driver to keep the vehicle lateral lane deviation as low as possible. Similarly while talking on the cellphone the $\mu_{\sigma_{SWA}}$ of the subject is more than it is in the normal driving situation.

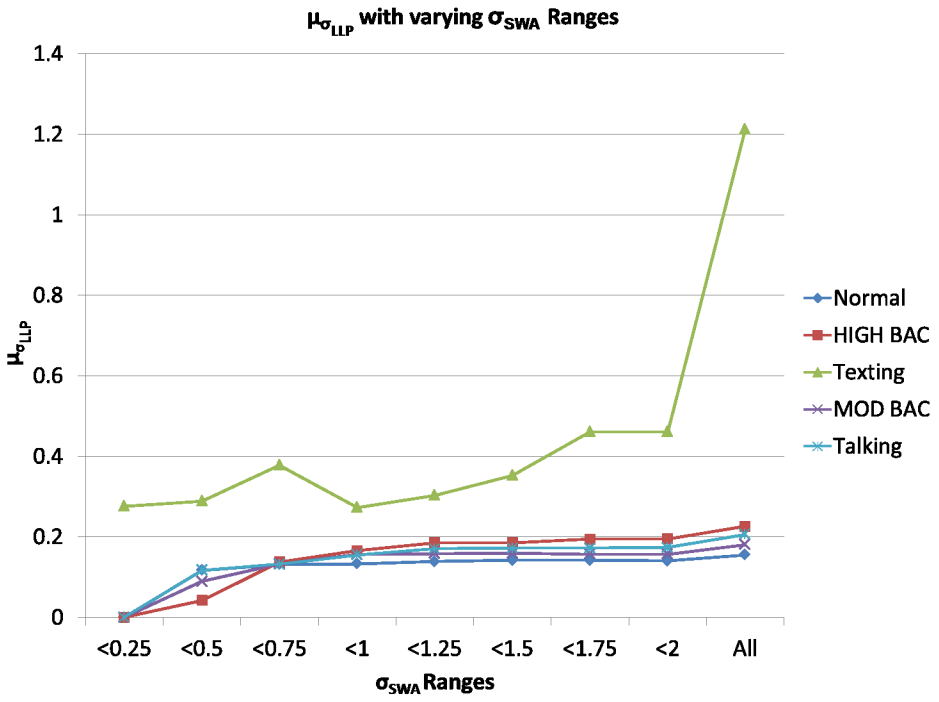

Figure 28.2: $\mu_{\sigma_{LLP}}$ with varying σ_{SWA} Ranges

Similarly from figure 28.2 it can be seen that the $\mu_{\sigma_{LLP}}$ values in alcohol influenced sessions are higher than those in normal sessions with different thresholds. It also indicates that under similar steering wheel control behavior, the lane deviation are larger in alcohol influenced session than in normal session. This means that alcohol influenced drivers have poor lane keeping ability than regular drivers. Moreover for texting while driving scenario even with a little deviation in the steering wheel input causes a significant deviation in the lateral lane position.

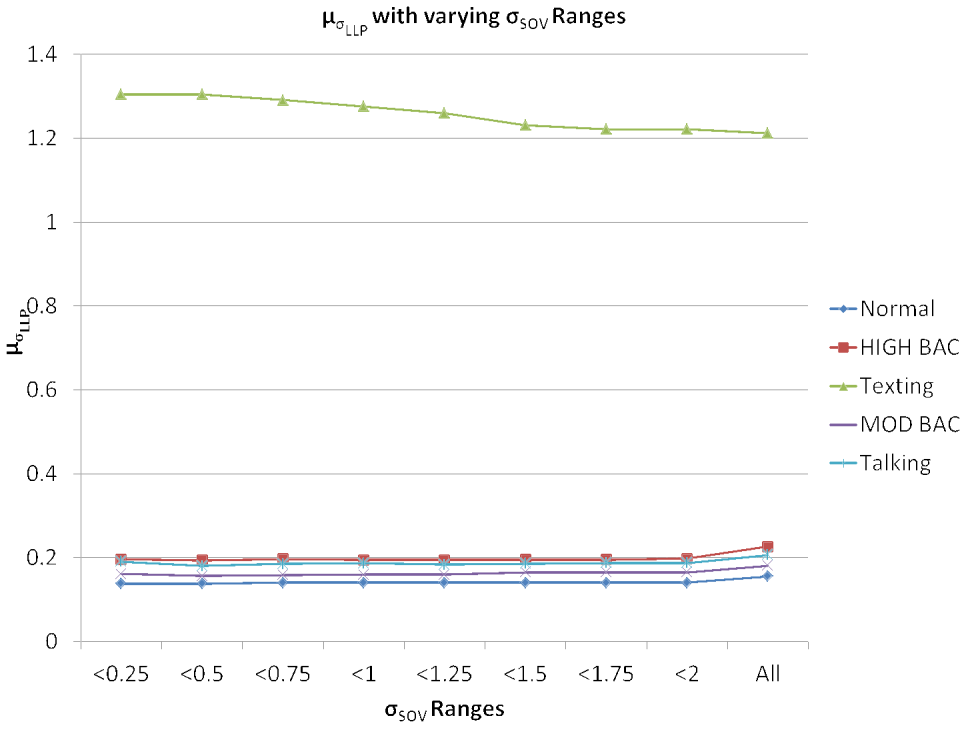

Figure 28.3: $\mu_{\sigma_{LLP}}$ with varying σ_{SOV} Ranges

A similar analysis has been performed for lateral lane position (LLP) and steering wheel angle (SWA) with respect to speed of vehicle (SOV). Figure 28.3 and 28.4 shows the $\mu_{\sigma_{LLP}}$ with varying σ_{SOV} ranges and $\mu_{\sigma_{SWA}}$ with varying σ_{SOV} respectively. For lower deviation in the speed of the vehicle, deviation in the lateral lane position and steering wheel angle is higher for when the driver is influenced with moderate BAC fatal vision goggles and when the driver is texting while driving. For texting session the deviation in the lateral lane position increases drastically as compared to the rest of the scenarios to maintain respective deviation in the speed of the vehicle.

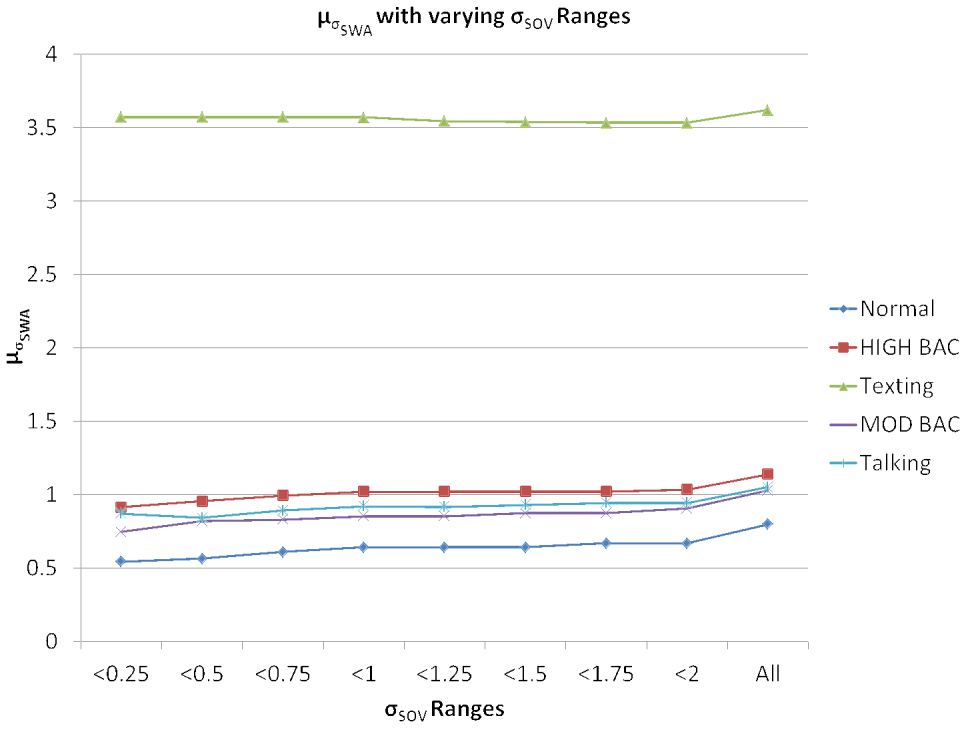

Figure 28.4: $\mu_{\sigma_{SWA}}$ with varying σ_{SOV} Ranges

28.4 Frequency Domain Analysis

A signal can be represented in an infinite number of different ways depending on the application. The most popular, important and fundamental representation is time and frequency domain represenatation of signals. The time domain indicates how a signal changes with time and the frequency domain indicates how often these changes takes place. The time domain signal was converted to frequency domain using fourier transform as shown in equation 28.5. The Fourier Transform involves decomposition of a signal as the sum of weighted sinusoidal functions of varying frequencies. Hence, the projection of the values of these signals forms the Fourier Transform of the original signal. As the lateral lane position is a discrete signal, discrete fourier transform (DFT) was applied to the signal to obtain a frequency domain signal.

$$X(w) = \sum_{n=-\infty}^{\infty} x[n]e^{-iwn} \tag{28.5}$$

Figure 28.5 shows the lateral lane position of Subject 1 in time domain. It is evident from figure 28.5 that, scenario where subject was texting while driving shows a lot of oscillation from the driving lane than in the normal driving scenario. In moderate BAC, the performance of the subject was better than the texting scenario but that can be attributed to the fact that even while driving with fatal vision goggles, driver is conscious and can focus on the road, but while texting, attention and eyes of the driver were mostly off the road. It was also interesting to note that the oscillation in the lane position when the driver was talking on the cellphone was more than the

normal driving scenario but significantly less as compared to other distraction based scenarios. Figure 28.6 shows the single sided frequency spectrum of the standard deviation of lateral lane position (LLP) of subject 1.

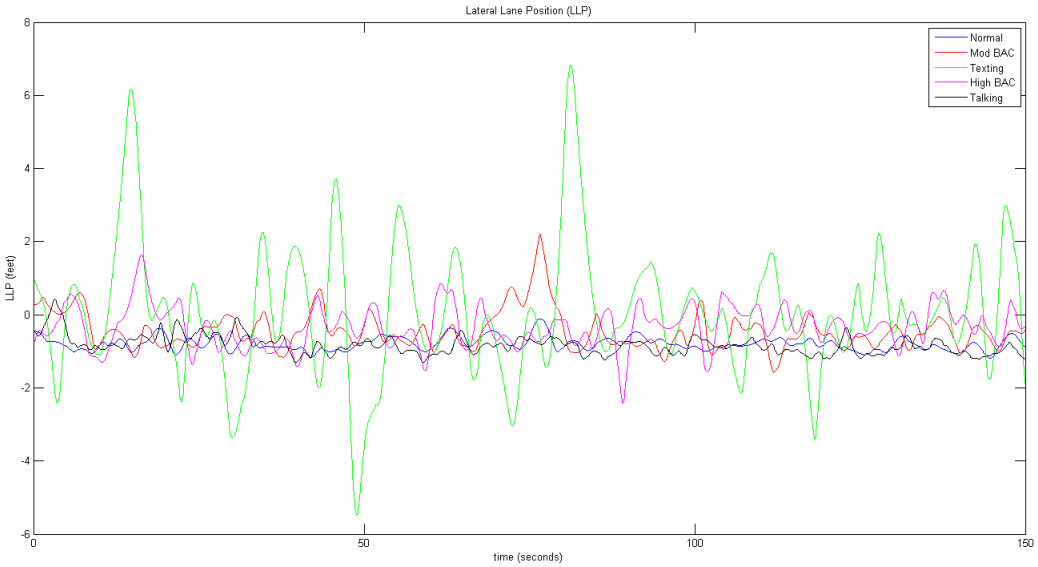

Figure 28.5: Time variation of Lateral Lane Position (LLP) of Subject 1(in feet)

Similarly, figure 28.7 shows the variation of steering wheel angle with time. When the subject was texting while driving, the oscillations in the steering wheel angle was exceptionally large. However, during the other distraction based scenarios the oscillation is slightly lower than texting but was in the relative decreasing order of HIGH BAC, MOD BAC, Talking and Normal Driving Scenarios. Figure 28.8 shows the single side spectrum of the standard deviation of steering wheel angle (SWA) of

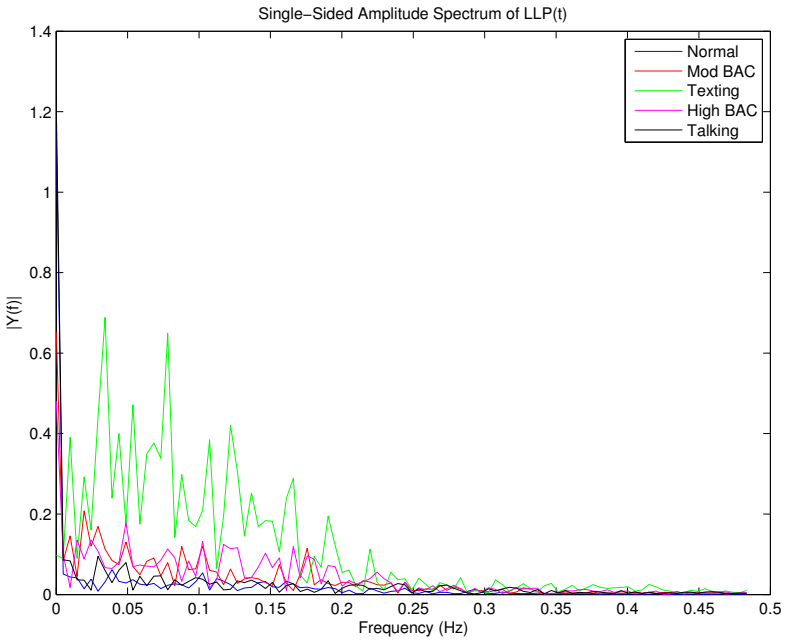

Figure 28.6: Single Sided Spectrum of Lateral Lane Position (LLP) of Subject 1

subject 1.

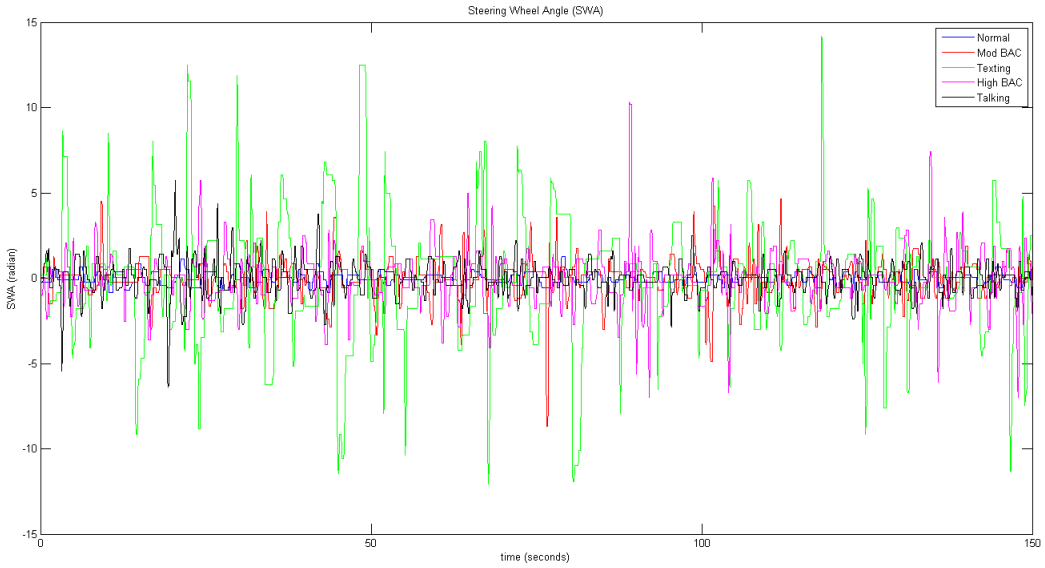

Figure 28.7: Time variation of Steering Wheel Angle (SWA) of Subject 1(in radian)

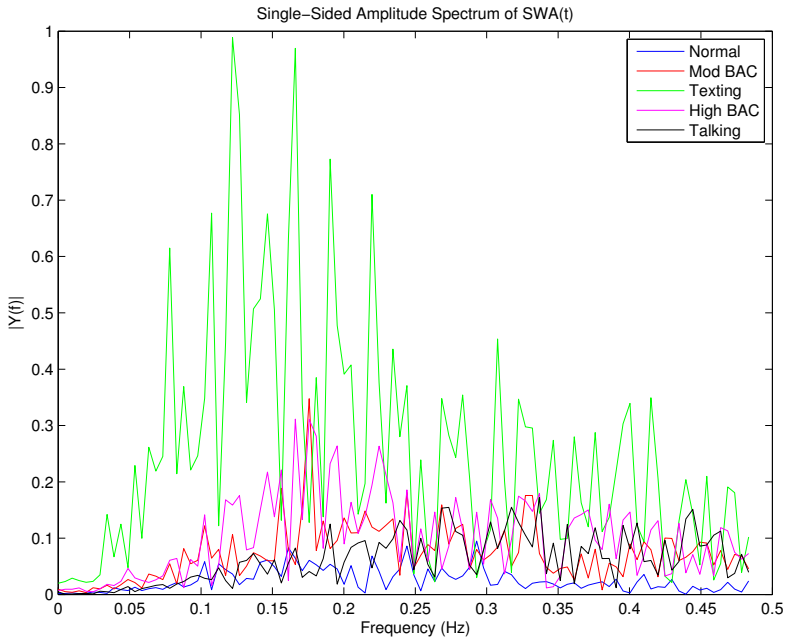

Figure 28.8: Single Sided Spectrum of Steering Wheel Angle (SWA) of Subject 1

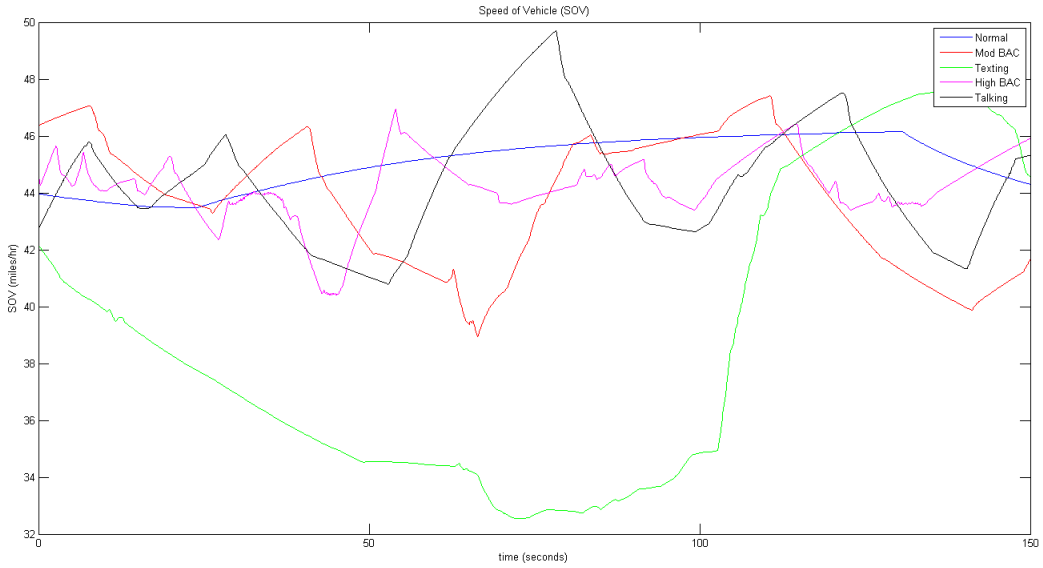

Figure 28.9: Time variation of Speed of Vehicle (SOV) of Subject 1

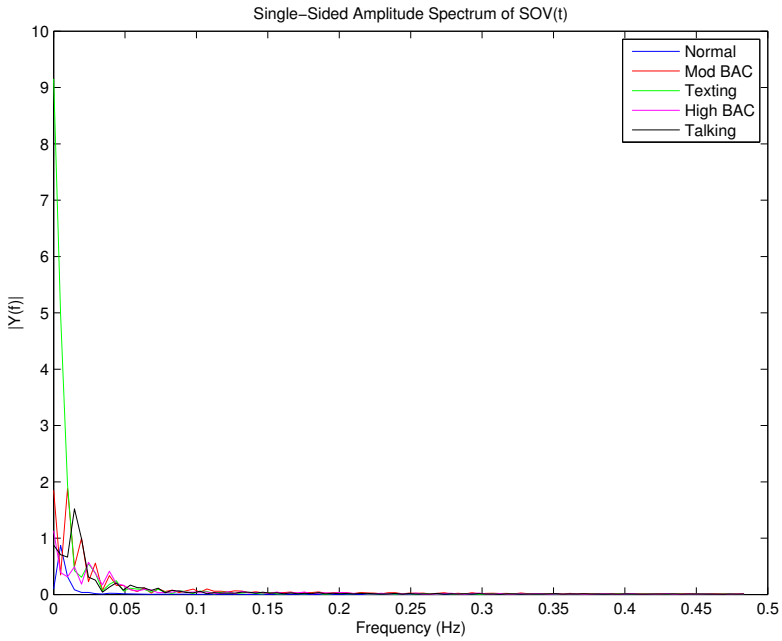

Figure 28.10: Single Sided Spectrum of Speed of Vehicle (SOV) of Subject 1

28.5 Entropy Based Analysis

In electrical engineering, entropy is defined as a function which conveys information about the behavior or attributes of a physical system. In thermodynamics, entropy is associated with the chaos or randomness in a given system. A system with higher entropy is considered more random or chaotic than the system with lower entropy. It has been used as a performance measure to assess driving performance of the subjects in various research environments. The main objective of this study was to monitor the effects of various scenarios involving distraction and driving under influence on the driver. Entropy as a performance measure was found appropriate to calculate the randomness introduced in the system due to different scenarios. Moreover, it also measures driver's efforts to bring the system back to its normal state as quickly and closely as possible. Hence it reduces the randomness and the overall entropy of the system.

For a discrete random variable X, the measure of uncertainty associated with the value of X is defined as entropy H. For a discrete signal X, entropy is defined as

$$H = \sum_{x \in X} -p(x)log(p(x)) \tag{28.6}$$

where $p(x)$ gives the probability for any $x \in X$.

Lateral lane position (LLP), steering wheel angle (SWA) and speed of vehicles (SOV) all are directly related to the driver's driving pattern. As this study was

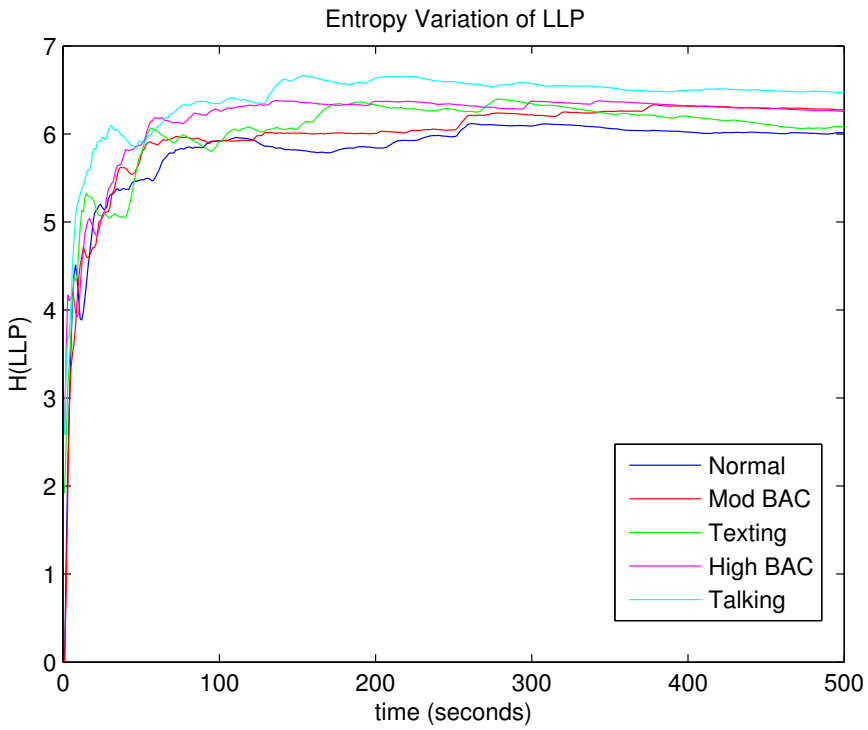

Figure 28.11: Entropy variation of LLP for Subject 1

conducted on a driving simulator with the scenario involving driving on a straight road under various distractions and influence, the main objective was to study the randomness in the driving pattern. The degree of randomness is shown in this section in figures 28.11, 28.12 and 28.13.

From figure 28.12, it can be seen that when the driver was texting while driving, lateral lane position has high degree of randomness as compared to the normal driving scenario. Similarly, for the scenario involving HIGH BAC the randomness is higher than normal but slightly lower than the texting while driving scenario. This explains the amount of distraction a driver may face on the road if engaged in these activities. It should also be noted that as these experiments were performed in a laboratory

controlled environment, the actual behavior on the road could be different for different drivers.

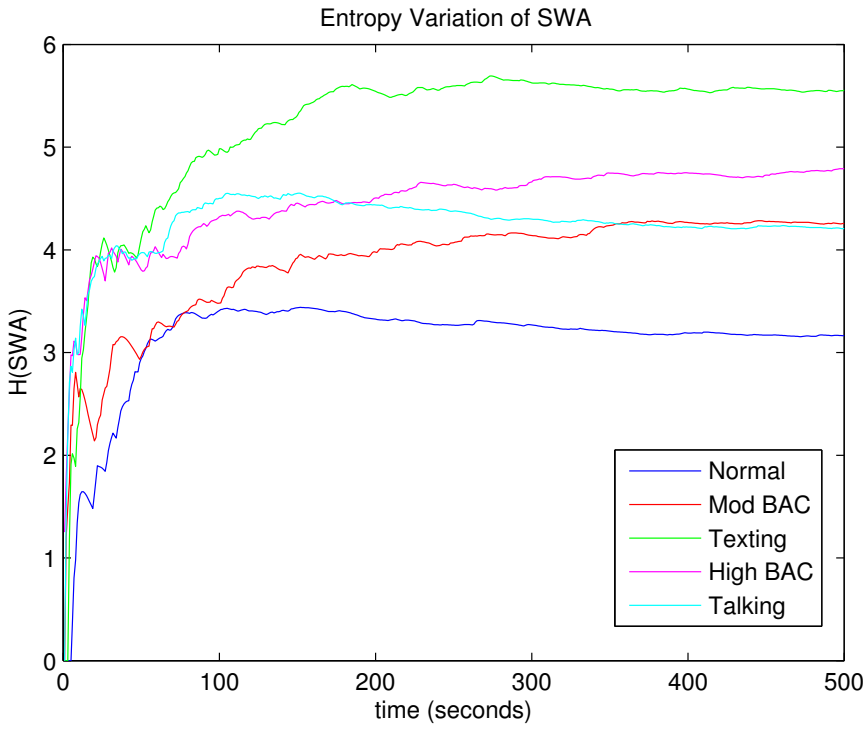

Figure 28.12: Entropy variation of SWA for Subject 1

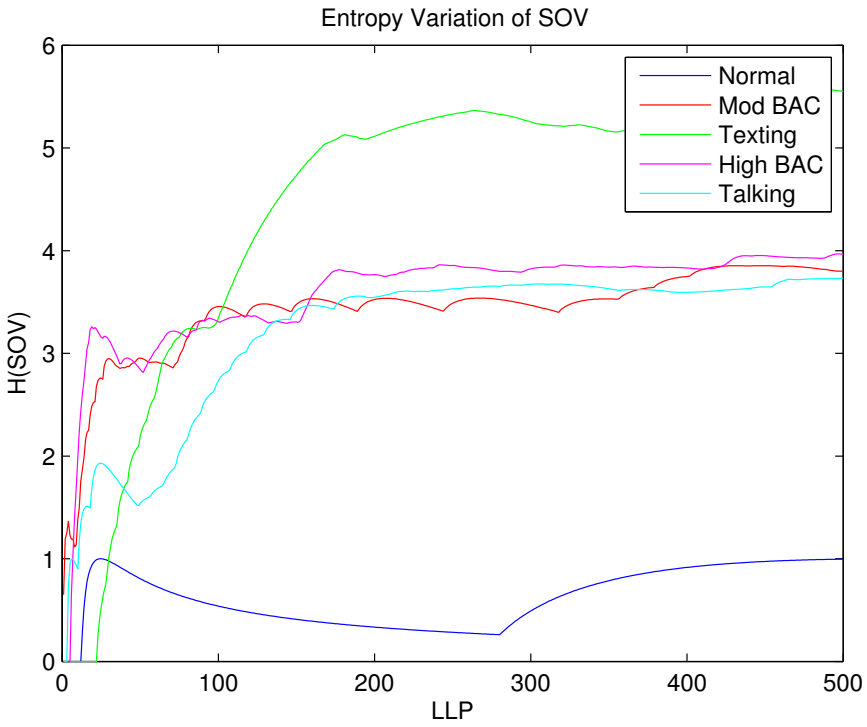

Figure 28.13: Entropy variation of SOV for Subject 1

BIBLIOGRAPHY

[1] Ralph Abraham, Jerrold E Marsden, and Tudor S Raiu. *Manifolds, tensor analysis, and applications*, volume 75. Springer, 1988.

[2] KEN BELSON. Sidewalk smackdown; no, you can't walk and talk at the same time. NY Times, 8 2004.

[3] Alessandro Bissacco and Stefano Soatto. Hybrid dynamical models of human motion for the recognition of human gaits. *International journal of computer vision*, 85(1):101–114, 2009.

[4] Michael R. Bloomberg and Amanda M. Burden. New york city pedestrian level of service study phase I. NYC DCP - Transportation Division, 4 2006.

[5] Brian L Bowman and Robert L Vecellio. Pedestrian walking speeds and conflicts at urban median locations. Technical report, 1994.

[6] Roger Brockett. Global descriptions of nonlinear control problems vector bundles and nonlinear control theory. manuscript, 1980.

[7] Alessandro De Luca and Giuseppe Oriolo. Modeling and control of nonholonomic mechanical systems. *Kinematics and Dynamics of Multi-Body Systems, CISM Courses and Lectures*, 360:277–342, 1995.

[8] Nathan Guequierre. Demographics and transportation in the united states 2050. CE 790, University of Wisconsin-Milwaukee, 12 2003.

[9] Serge P Hoogendoorn. Pedestrian flow modeling by adaptive control. *Transportation Research Record: Journal of the Transportation Research Board*, 1878(1):95–103, 2004.

[10] TranSafety Inc. Designing traffic signals to accommodate pedestrian travel. Road Engineering Journal, 1997.

[11] C. Jotin Khisty. Evaluation of pedestrian facilities: Beyond the level-of-service concept. *Transportation Research Record*, 1438:45–50, 1994.

[12] Richard L Knoblauch, Martin T Pietrucha, and Marsha Nitzburg. Field studies of pedestrian walking speed and start-up time. *Transportation Research Record: Journal of the Transportation Research Board*, 1538(1):27–38, 1996.

[13] Microsoft. Getting the next frame of data by polling or using events. Documentation website, 7 2012.

[14] Microsoft. Interaction space of kinect for windows. Documentation website, 7 2012.

[15] Microsoft. Kinect for windows architecture. Documentation website, 7 2012.

[16] John S. Miler, Jeremy A. Bigelow, and Nicholas J. Garber. Calibrating pedestrian level-of-service metrics with 3-d visualization. *Transportation Research Record*, 1705:9–15, 2000.

[17] Lee Morgan. The effects of traffic congestion. USA Today.

[18] NHTSA. Traffic safety facts - 2009 data. DOT HS 811 394, 2010.

[19] Henk Nijmeijer and Arjan Van der Schaft. *Nonlinear dynamical control systems*. Springer, 1990.

[20] George J Pappas and Shankar Sastry. *Towards continuous abstractions of dynamical and control systems*. Springer, 1997.

[21] Sheila Sarkar. Determination of service levels for pedestrians, with european examples. *Transportation Research Record*, 1405:35–42, 1993.

[22] Michael Spivak. A comprehensive introduction to differential geometry, volume i. *Publish or perish, Berkeley*, 1979.

[23] Dave Thompson. Stride analysis. website, 2002.

[24] Marion Trew and Tony Everett. *Human movement: an introductory text*. Churchill Livingstone, 1997.

[25] Alessandro Valli. The design of natural interaction. *Multimedia Tools and Applications*, 38(3):295–305, 2008.

[26] William H Whyte. *City: Rediscovering the center*. University of Pennsylvania Press, 2012.

Nevada Department of Transportation
Rudy Malfabon, P.E. Director
Ken Chambers, Research Division Chief
(775) 888-7220
kchambers@dot.nv.gov
1263 South Stewart Street
Carson City, Nevada 89712