2001 01 1874

THE MORIS MOTORCYCLE SIMULATOR: AN OVERVIEW

D. Ferrazzin, F. Salsedo, F. Barbagli, C. A. Avizzano, G. Di Pietro, A. Brogni, M. Vignoni, M. Bergamasco
PERCRO, Scuola Superiore S.Anna
via Carducci 40, 56127, Pisa, Italy
[waam, fed, carlo, Bergamasco]@sssup.it

L. Arnone, M. Marcacci, L. Masut, A. Benedetti,
Engine Division, Piaggio & C.
Viale Rinaldo Piaggio 25, 56025, Pontedera, Italy
luigi.arnone@piaggio.com

1 Abstract

Many motion based simulators have been developed in the last thirty years for many different types of vehicles [1] [2]. In order to make a simulation more realistic, linear accelerations and angular rates are exerted on the pilot by moving the platform on which the mock-up vehicle is located.
The purpose of this article is to present a 7 DOFs (vertical, lateral and longitudinal displacements, roll, pitch and yaw angles and steer) motorcycle simulator which has been realized in Pisa, Italy, in the framework of the Esprit project by a consortium composed by industrial partners (Piaggio and Humanware from Italy and HEAD acoustics from Germany) and by academic partners (Scuola Superiore S. Anna from Italy, Halmstad University from Sweden and University of Bochum from Germany).
Such project started in 1995 with the aim of developing a two-wheeled motorcycle simulator conceived as a tool for the designer to acquire data on motorcycle handling and stability at the design stage as well as to collect data about rider control behavior implications in motorcycle performances.

2 Introduction

Flight simulators have been the reference point in the field of vehicle simulation for the last 30 years. This has been due to the high costs of aircrafts, if compared to other vehicles as cars or motorcycles. Flight simulators have always been less expensive than the actual aircraft they were trying to reproduce, thus allowing pilots and crews to be trained at lower costs and lower risks. The same cannot be said for car and motorcycles and here lays one of the basic differences between such types of simulators. Land vehicle simulators have been developed with different purposes, most often as a tool for designers to test new prototypes before actually building them or to study human behavior in specific situations.
In the field of motion base motorcycle simulators little studies have been conducted. To the authors' knowledge only the Honda Motorcycle Simulator [3] (with a 2 DOF simulator) and the YNL Motorcycle Simulator of Tokyo University [4] (a 6 DOF Simulator) have been studied and realized.
The final goal of the MORIS project [5] is to realize a powerful tool to assist the design and the development process of two-wheeled vehicles: "a virtual driving machine" that will allow to gain knowledge about optimum values of some mechanical parameters, in order to reduce the number of road tests presently required after prototype fabrication.
In the simulator, the rider experiences the same physical sensations as those perceived during the driving operation of a real motorcycle. This is valid not only in terms of visual and the acoustical types of feedback stimuli, but also for perceived sense of movements, accelerations and decelerations ones, control movements of the vehicle, and in terms of the physical interactions arising with the real mechanical structure of the simulator
MORIS is a "motion-based" simulator, i.e. it is equipped with moving parts in order to reproduce, with some degree of approximation, the dynamics of a real motorbike. The final system presents the human operator seated on a mock-up of a 2-wheeled vehicle. The mock-up is intended as a rigid structure that is moved with respect to a ground frame of reference by a mechanism (actuation system) possessing the required number of DOF. The human operator interact with the the simulator as well as a real motorcycle.
In the scenario other interface systems are present:

- a *graphical interface* allowing the human operator to obtain a visual representation of the virtual scenario;
- an *acoustical interface* allowing the human operator to perceive realistic sounds belonging to the real scenario; this can be achieved by considering a binaural feedback system;

The MORIS simulator aims:

- to become a useful tool in the motorcycle development phase when several trial-and-error loops on running prototypes are necessary to achieve a satisfactory riding behaviour. The reason for the prototypes development is that the two-wheelers frame and suspension tuning has a dramatic influence on the vehicle dynamic behaviour. The MORIS simulator could speed up the development phase, making feasible a trial-

and-error loop procedure without incurring the need of building running prototypes in the early development phases;

- to become a market research tool by allowing potential new customers to experience simulated rides on potential new concept vehicles, in order to evaluate the customers' satisfaction;
- to simulate dangerous riding conditions, so that the rider and/or the researcher that analyses the results of the test can gain useful information without undergoing the risks associated with the specified riding conditions.

The vehicle simulator renders on the various users the behavior of a medium-class, non-racing, modern motorbike

3 Performance Evaluation

The overall dynamic of the simulator is verified on the Basic Design Maneuvers (BDMs) [5]. The check is done by controlling the replication of the inertial cues be within the accepted errors and by testing the various subsystems involved in such replication.

The Basic Design Maneuvers can be considered typical motorcycle maneuvers recorded using sensorised vehicle in circuit running. Examples of these maneuvers are the startup-stop, an acceleration and deceleration of the vehicle, the straight forward running stability test, with the motorcycle running along a straight direction and, at a certain instant a wind gust or a trucks wave is simulated, the lane changing, with the motorcycle changing a lane at constant velocity, the uphill/downhill driving, in which the motorcycle, running at constant velocity, follows a straight street on a hill. These were chosen as the best maneuvers allowing the measurement of the vehicle in terms of stability, maneuverability, suspension comfort and vibration feeling [5].

4 MORIS General Architecture

The overall architecture of the MORIS simulator is reported in Figure 4-1. The Plant is composed by the Rider Command Acquisition and the Motorcycle Mock Up from which the rider's commands are acquired. They are sent to the Dynamic Model (DM) in which the trajectories of the motorcycle are evaluated. These information are necessary to update the virtual environment and, as a consequence, the Graphics Subsystem (VS), the Acoustic Subsystem (AS) and constituite the inputs for the Frame Controller (FC). The FC commands the actuators of the Stewart Platform and of the Mock Up steer. The Real Time Subsystem (RTS) manages all the process while the Control Console (COS) is the visual output of the status of the simulator. The various subsystems are connected with communication cables designed w.r.t. the real time requirements of the architecture (Ethernet, Isa and FDDI bus as required). In the following sections the various subsystems will be better analysed.

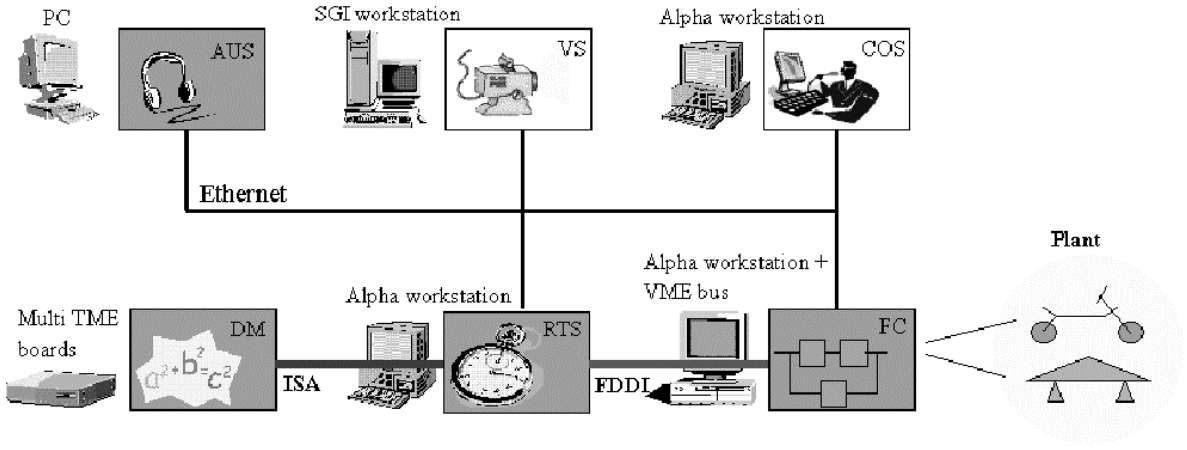
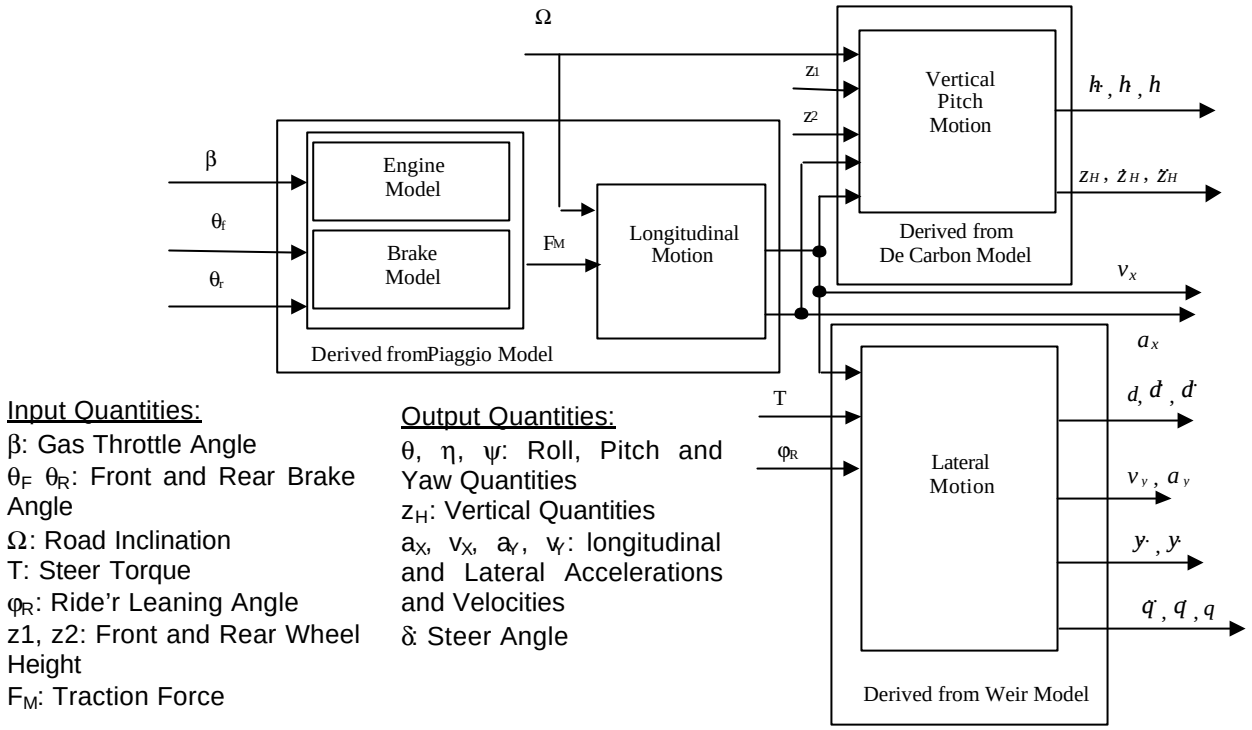


Figure 4-1 General Architecture

5 Vehicle Dynamics Mathematical Model

An important component of the MORIS software is the DM that simulates the two-wheeled vehicle dynamics [6]. The DM solves in real-time the equations of motions written for the multibody system that approximates the geometrical, inertial and mechanical characteristics of the vehicle that will be ridden by the operator. The core of the dynamic model (Figure 5-1) has a 1 DOF longitudinal motion block in which the motion along the longitudinal direction of the motorcycle is elaborated. The information obtained are sent to two decoupled models that compute the vertical motion (a 2 DOFs model based on De Carbon scheme) and the lateral motion (4 DOFs based on Weir scheme). The outputs are the positions, velocities and accelerations of the Motorcycle in a local inertial frame of reference. Other external blocks allow to evaluate the absolute position, orientation, velocity and acceleration of the motorcycle w.r.t. a fixed base frame. All the blocks have been

developed in an overall Simulink/Matlab scheme. The system has been successfully tested and compiled for a DSP board.



Input Quantities:
β: Gas Throttle Angle
θ_F θ_R: Front and Rear Brake Angle
Ω: Road Inclination
T: Steer Torque
φ_R: Ride'r Leaning Angle
z1, z2: Front and Rear Wheel Height
F_M: Traction Force

Output Quantities:
θ, η, ψ: Roll, Pitch and Yaw Quantities
z_H: Vertical Quantities
a_X, v_X, a_Y, v_Y: longitudinal and Lateral Accelerations and Velocities
δ: Steer Angle

Figure 5-1 Dynamic Model Core Scheme

6 Mechanical Subsystem

6.1 Motorcycle Mock Up

Figure 6-1 Mock Up Subsystem

The Vehicle Mockup Frame is designed in order to have same interfaces of the present Hexagon vehicle and stiff connections to the Stewart Platform. All the subsystems and specific components are included in the existing vehicle layout (undercover). The front connection is designed for the Steer Mechanical Frame interfaces; the rear connection is designed with a stiff layout in order to transfer correctly the maneuver feedback from the Stewart Platform to the Driver In the Mock-Up, the standard functions are completely integrated: front and rear brakes, throttle handle, key, starter and horn, position lamp, high and low beams and indicators. Upper part of the Mock-Up Subsystem is mainly composed by a vehicle frame that represents the natural interface between Rider and Motorcycle Simulator while the lower part is composed by a base platform that holds a lateral protection to prevent possible Rider fall. The Rider Command Acquisition, the box that allows to acquire the rider's commands, is connected to the upper part of the Mock-Up, as well as the Engine

Induced Vibration Subsystem, which reproduces vibrations on vehicle frame simulating the real vibrations induced by the thermal engine. The Motorcycle Mockup was designed and manufactured, modifying the frame of a Piaggio Hexagon vehicle currently in production. The frame of the vehicle, modified in some essential parts, is fixed to the upper platform of the Simulator by means of two basements, one in the front and one in the rear part of the vehicle, as shown in Figure 6-1.

6.2 Actuation Subsystem Design

In order to evaluate the optimal kinematics of the SP and the requirements of the hydraulic actuators the BDMs have been utilised. The stroke, thrust and velocity of each actuator have been elaborated in order to find the load conditions acting on each part of the SP, divided into Upper and Lower Platforms, Universal Joints, Yaw Pivots and Actuators.
A 3D model has been realised in order to perform interference checks (Figure 6-2), FEM analysis (Figure 6-3) and obtain the constructive drawings (Figure 6-4) of all custom components. In the case of commercial components (bearings and so on) standard verification have been carried out. Finally the realised components have been assembled (Figure 6-5).

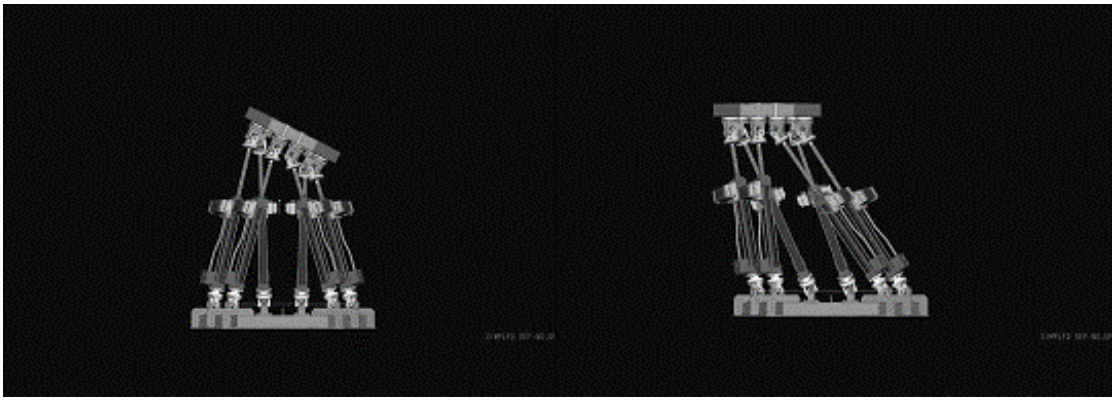


Figure 6-2 Interference Analysis of the SP

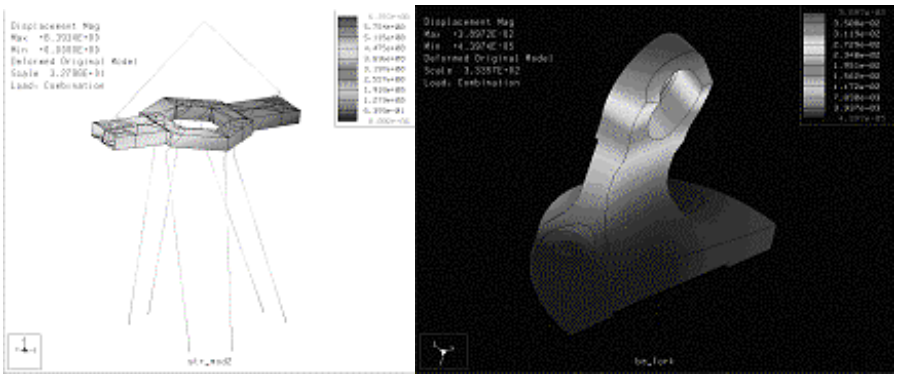


Figure 6-3 Structural Analyses of the SP Main Components

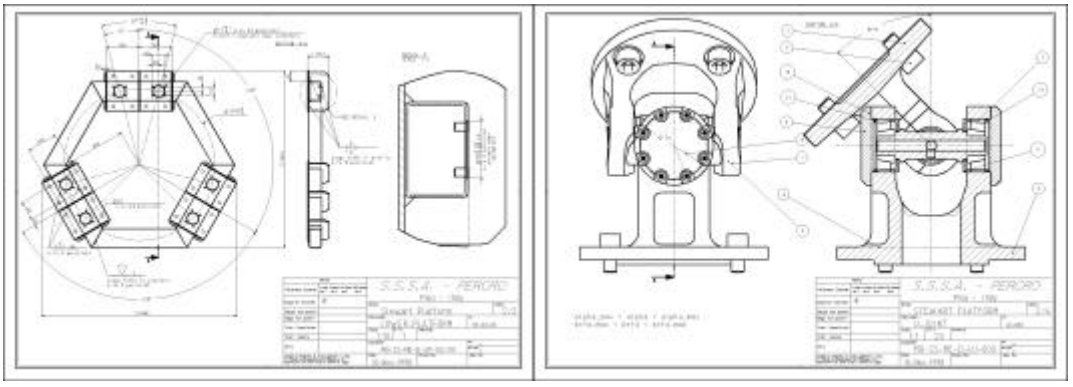


Figure 6-4 Constructive Drawings of the Lower Platform (left) and the Universal Joint (right)

The main specifications of the Stewart Platform are:

- Main dimensions: upper radius 0.3 m, lower radius 0.6 m, semi-angle between actuators 15 degrees, reference height: 1.4 m.
- Actuator thrust: 9 kN, Stroke: 600 mm, Velocity: 0.4 m/s, Accuracy: 1%;
- Moving Payload: 600 kg;
- Universal Joint Conic semi-angle: 45 degrees.

The Actuation System is constituted by a high-pressure hydraulic system and six linear actuators [7]. This system has been defined and supplied by an external premise.

Figure 6-5 Final SP Manipulator during mounting phase

7 Real-Time Subsystem

7.1 Real-Time Architecture

The design of the Real-Time Subsystem (RTS) is based on much functionalities, including the computation of the inertial feedback and the synchronization among the different subsystems [8]. These requirements imply a complex control that it is performed by means of communicating tasks. Some of these communications have to be synchronous in order to guarantee the correct semantics of the calculus. The design of the RTS uses theoretical results known in order to forecast the behavior of the RTS and verify the correspondence to its functional and temporal requirements.
The RTS is characterised by different tasks. As basic solution, the system has been implemented using Real-Time operating System (Vx-Works based on AlphaStation).
The RTS can be divided into the main modules described in the following.

a) *Circuit Database Manager*: this logical unit is in charge of all the operations needed to maintain the information about the circuit for the current simulation. In particular, this module exports the function needed to manipulate the circuit description, to navigate on the circuit description and to obtain reference to the different segments. Moreover, the functions of this unit allow saving and restoring the circuit description on the permanent memory.
b) *RT OS Layer*: this unit is in charge of extending the functionality of the VxWorks operating system. The functionality provided by this unit bridge the specified needs of the MORIS application developers and the primitives available from the VxWorks system.
c) *Communication Module*: this unit is in charge of performing all the necessary operations with the different network connections of the Real-Time Subsystem. The functions of this unit allow both to boot the information from the network (in the setup mode of the simulator) and to communicate during the

simulation with the other subsystems using its three interfaces (namely, the ISA connector, the FDDI ring and the Ethernet bus).

d) *Lane Manager*: it cooperates with the circuit and segment managers to return information on the number of lanes and the exact position of the lane centers in each road segment and for any distance from the beginning of the segment.
e) *Calculus Solver*: this module is used to calculate integral, derivatives, linear and quadratic fits.
f) *Geometry Manager*: this logical unit is in charge of the low level manipulation functions on the geometry of the simulated environment. These functionality include a set of functions to translate a generic point coordinates from the Absolute Reference Frame **(WBF)** to another Reference Frame like the Segment Relative Reference Frame (**SRF)** or to the Segment Local Reference Frame (**SLF)** and viceversa.
g) *Dynamic Objects Manager*: this module contains all the necessary functions for the operation the dynamic objects during the MORIS simulation. These functions are invoked at run-time by the tasks responsible for the management of the dynamic object.
h) *Terrain Condition Evaluation Module*: this module manages exports the functionality to retrieve the z-coordinates of the wheels MORIS motorcycle. Moreover, this module is in charge extracting other parameters like the grip and the roughness of the road at the contact points of the two wheels.
i) *Environment Generation Module*: this module exports the functionality required to generate the weather conditions (rain, fog, wind) in the simulated environment.
j) *Relative Wind Module*: this module exports the functionality required for calculating the speed of wind w.r.t. MORIS reference system.

7.2 Strategy Manager Unit

The purpose of the Strategy Manager Unit (SMU) is to transform the trajectories generated by the DM, which include very large displacements, into actuators commands capable of providing the pilot with realistic motion cues while remaining within the simulator's workspace limits [9].
This type of unit is referred to as Washout Filter and has been widely investigated in the field of flight simulator design [10] [11]. The design of efficient washout filters is a complex problem. These filters are first of all complex control systems whose robustness and stability must be ensured in order not to cause mechanical damage to the simulator. Moreover washout filters must take into account the nondeterministic nature of pilots making it hard to define what “realistic'' means and making this a complex design problem in the field of human factors and human-machine interaction. The medical and technical literature consider that the human’s head contains the main receptors of the inertial sense. In particular, there are sensed the linear accelerations (by the Otolith) and the angular velocities (by the Semicircular Canals) [6].
In the MORIS simulator, the algorithm has been developed evaluating the washout location (the reference point on which the SMU works) on the rider’s head.
The outputs of the DM are evaluated on the rider’s head and then processed by the Strategy Manager filter. The outputs are sent to the FC to move the Stewart Platform and to the VS to update the position of the rider’s point of view and the landscape horizon
Referring to the Scheme reported in Figure 7-1, the SM is a Matlab-Simulink Module composed by the following blocks:

- ✓ Strategy Splitter (SS):

  The input of the SMU has to be pre-computed before actually being fed to the SMU. In order to do this note that the accelerations given by the HAE include the gravity component felt by the rider, which is usually mainly felt along the z axis of $<S_H>$ (the frame of reference solidal with the Rider’s Head).
  Since the gravity is felt by the real rider of the simulator, the SMU should only track the vertical accelerations not containing vector **g**. Therefore vector (o,o,-g)^T is transformed into the $<S_H>$ and added to the desired vector to be tracked.
  The resulting vector is then transformed into $<S_B>$ (the simulator base frame of reference). Finally the resulting vector is divided into high frequency and low frequency components, i.e. into $\vec{a}_{GRAV}$ and $\vec{a}_{MOV}$.
  A fourth-order filter has been chosen to ensure that the platform is ``washed out'' back to its zero position after some time, i.e. that $\vec{x}_{MOV} = \int\int \vec{a}_{MOV} \cdot dt = 0$ at steady state, for step and ramp inputs.
  Twelve parameters can be regulated in order to vary the threshold between high and low frequencies to be replicated using different strategies;

- ✓ Direct Linear Motion (DLM):

  integrates twice $\vec{a}_{MOV}$ and uses such information to linearly drive the SP. Since $\vec{a}_{MOV}$ represents the high frequency components of the accelerations felt on the rider's head, and because of the structure of the SS, such commands will drive the platform for short displacements and will drive it back to its zero position thus making sure that the workspace limits won't be met. However, safety units can be used in order to eventually decrease the linear velocities of the platform in order for it to reach a pre-set safety limit with zero speed. Such unit reproduces unrealistic motion cues on the rider but is not intended to normally operate when the system is tuned properly by the operator;

- ✓ Gravity Strategy (GS):
  tracks $\vec{a}_{GRAV}$ by tilting the platform around its pitch (φ) and roll (θ) angles. This will introduce a certain error on the acceleration felt along the z_H-axis. Moreover saturation modules have been inserted in order to limit roll and pitch rates, since the tilting should not be perceived by the rider [6], and pitch and roll accelerations since the platform has physical limits to its maximum angular accelerations. The Gravitational Strategy unit is non-linear due to the saturation blocks and to the coupling effects introduced by rotation matrix R_B^H. As far as the stability is concerned it is possible to show, using Lyapunov's theory, that this unit results asymptotically stable in a ball centered on the system's equilibrium points;
- ✓ Strategy Filter (SF):
  transforms the desired angular velocity vector $\vec{w}_h$ into $(\dot{q} \quad \dot{j} \quad \dot{y})^T$. Such angle velocities are then high passed filtered and integrated. Note that Euler angles must be used in order to be integrated.
- ✓ Drift Compensation (DC):
  computes spurious accelerations on the rider's head due to the platform tilting introduced by GS and DAM. The Coriolis acceleration $\vec{w} \times (\vec{w} \times p_h^P)$ is computed and fed forward to SS in order to be "partially deleted" using the linear direct strategy;
- ✓ Direct Angular Motion (DAM):
  Performs the same operation of DLM washed out at zero position at steady state the angles computed in SF.

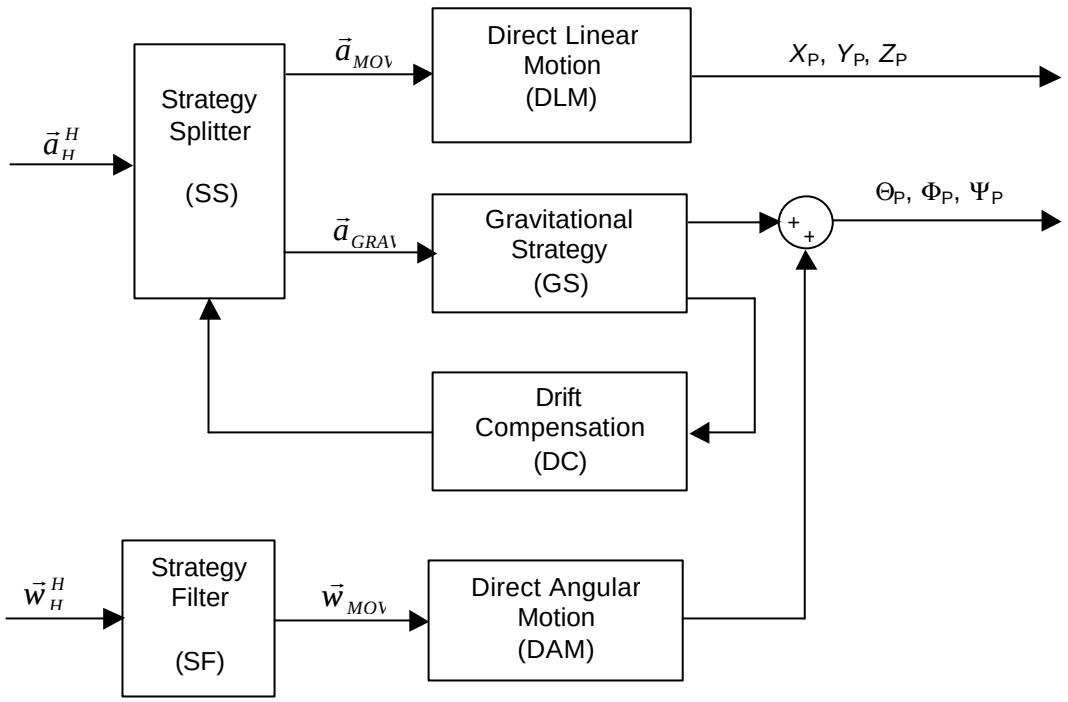


Figure 7-1 Strategy Manager Core

8 Control Console

The console contains different programs that allow the user to edit or define new simulations and establish their parameters, run, control and monitor the simulation and finally review and analyze the results [8]. Any simulation is defined by the composition of a number of «objects» that are the entities to be represented during the simulation and the environment in which the simulation takes place. Among the simulation objects are the motorcycle, the circuit, and the objects of the scenario (other vehicles, trees, houses and so on) the environmental conditions like the lighting, the wind or the fog, the dynamic objects and their behavior.

In order to support this scheme, the console requires some Editing and Analysis Tools and the support of a Simulator Database. The Editing Tools are used to create the software objects, while the Simulator Database stores the basic object descriptions and keeps track of how the software objects and the environmental conditions are combined in a simulation run. The user console hosts the *Weather*, the *Circuit Editor* and the *(dynamic objects) Trajectory editor* tools. Finally, and most important, the user console is used to control the simulation. The simulation control includes the initialization of the subsystems, the system startup, the

simulation boot and start and the run-time management.. Finally, the Console is used to monitor the simulation data in real-time and to analyze the simulation results.
The COS (Figure 8-1) possesses four interfaces modes:

- ✓ the simulation control mode, that implements all the actions that must be undertaken by the console subsystem in response to the user commands or to implement the functioning modes of the console statuses (simulation setting, log setting, monitor setting, start and setup setting write or plot data on the interface);
- ✓ the console database in which it is possible to manages all the objects in the simulated environment;
- ✓ the circuit editor and other editors, that interfaces the graphical user console with the circuit file where the circuit and the weather description are stored;
- ✓ the editing tools, in which the analysis mode can be selected from the menubar of the main console window.

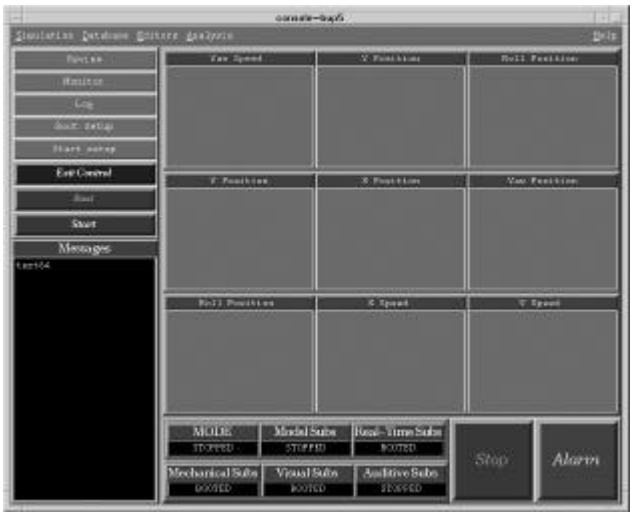


Figure 8-1 Console Graphical Appearance

9 Acoustic Subsystem

The Acoustic Subsystem (AS) was implemented as described in Figure 9-1. Three functional components can be identified:

- ✓ Scenario modelling software, hosting also the local Virtual Environment database,
- ✓ Sound generation module, including the sound source model, the local sound database and the sound generation hardware, integrated with the
- ✓ Auditory display, including the electroacoustic actuators (headphones).

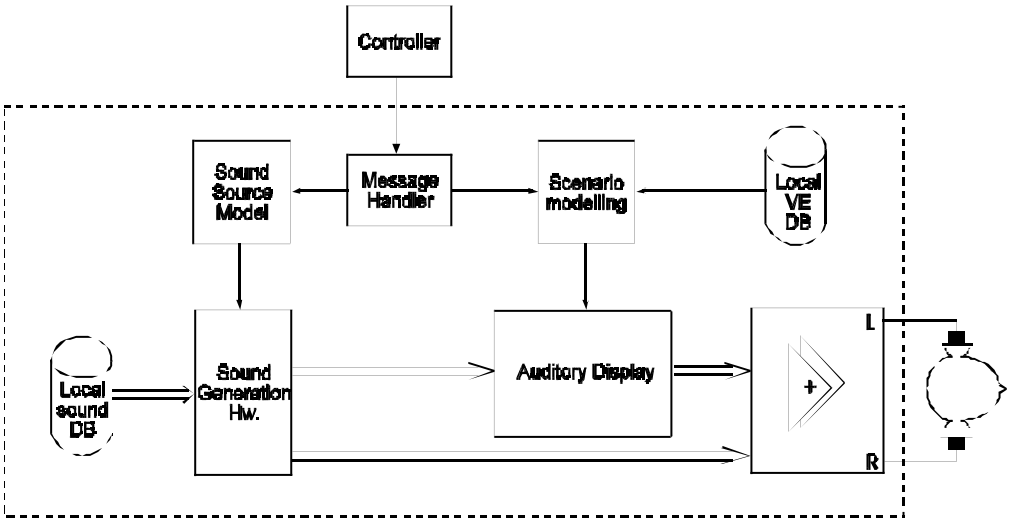


Figure 9-1 AS Functional Architecture

Sound generation module and auditory display are realised within a subsystem called **3D sound processor.** In each time frame, defined by update messages from the RTS, the scenario modelling software takes a decision, which objects within the virtual world have to be regarded as acoustically relevant and calculates all parameters necessary for auralization, e.g. direction with respect to the rider's head and Doppler shifts of all audible sound components. The sound generation module provides all sounds required by the renderer using a local sound database on hard disk. All sounds required for the auditory feedback are stored in this local database: engine sounds, tire sounds, wind noise, sound of other vehicles, background sounds, and so on. To introduce other types of motorbikes, other vehicles, tires or backgrounds new recordings are added to the database. Engine sound recordings have to be pre-processed before they can be included in the database. The auditory display performs the real-time signal processing for the realisation of Doppler shifts and binaural coding, delivering binaural sounds for headphone reproduction [15] [8]. Two PC systems are used for the hardware implementation of the auditory subsystem. The scenario modelling software is implemented on PC 1, the 3D sound processor on PC 2.
Data transfer between modules is configured by a programmable crossbar switch. The board provides a calculation power of totally 192 MIPS. In-the ear headphones are used electroacoustic actuators, covering a frequency range from 20 - 14000 Hz, providing peak levels up to 110 dB at 1% distortion. The headphones are connected directly to the ADDA board connected to the DSP V.
The AS is controlled by the RTS. The communication between AS and RTS is divided in static and dynamic communication. The former are used during the start-up of the system while the latter are used during the simulation.

10 Graphical Subsystem

The Graphical Subsystem (VS) is in charge of the visual feedback on the user of the MORIS Simulator. The VS is strictly connected with the RTS from which acquire the absolute position of the Rider's Head inside the Virtual Environment and evaluates the point of view of the rider.
The VS has a bandwidth of 24 frame/sec and a latency less than 40 ms. The computer that elaborates the graphical informations is a Sylicon SGI Onyx Reality Engine 2, equipped with 4x4400@250MHz processors, 256 Mbytes RAM, 16 Mbytes of texture memory and 1 Raster Manager.
Several software techniques are used in order to achieve the MORIS requirement with the available hardware [12]:

1. **Non visible objects culling**: a bounding volume of the virtual object is checked again the view frustum. The object is not drawn at all if the bounding volume is completely outside the view frustum. We are able to manage large and complex virtual scenarios with this kind of technique;
2. **Level Of Detail**: each virtual object is defined by multiple level of detail. The level of detail is chosen according the distance from the observer. Far objects are drawn with less polygon than near objects. Useless details are not drawn;
3. **Constant Frame Rate**: during each frame a global LODScale is calculated. If the frame rate is too low, the software will draw objects with less details in order to gain performances; if the frame rate is too high, the software will draw objects with more details in order to slow down and to increase the visual quality. We are able to achieve a constant frame rate with this kind of control;
4. **Automatic Strips Finder**: strips of polygons with shared vertices are automatically found and drawn in order to have a faster downloading of the geometrical primitives to the graphical accelerator. We are able to reach the maximum performances of the graphical accelerator;
5. **SMP Multiprocessors Support**: the software is able to take advantage of SMP workstation in order manage large and complex virtual scenarios;
6. **Soft Real-Time Programming**: the software is has a deterministic behaviour in order to maintain a constant frame rate and a fixed latency. Several techniques are used in order to achieve these results. All the available processors are used only by the Visual Feedback's processes. The only free processor is used for any standard system activity. Each VF's process is statically allocated on a processor and only a static priority is used for each process. All the VF's memory is marked as non swappable;
7. **Precalculation Of Illumination**: for all static light sources, the object's illumination are precalculated offline in order to offload the CPUs and graphical accelerator from useless computation;
8. **Texture Mapping:** All kind of texture mapping (1, 2, 3 and 4 channels, blend, modulate, decal and replace, etc.) are supported in order to reach an high quality visual feedback;

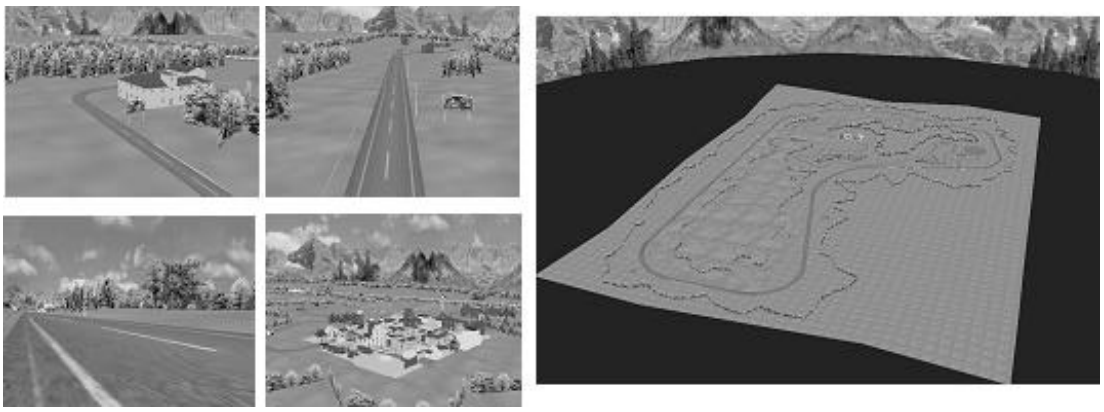


Figure 10-1 Rendered model

11 Frame Controller

The FC has two main purposes in the control subsystem. First of all it acquires the actuator displacements. Secondly it controls the movement of such actuators by sending a new position reference to an embedded controller unit. Its inputs depend on the specific state of the system. If the system is normally functioning, the inputs to FC are provided by the outputs of the SMU (trajectories of the upper plate of the SP) transformed in the joint space (trajectories of the six hydraulic actuators). When any exception occurs, for instance reaching a workspace limit or an unwanted electrical block out happens, the system automatically reset its position to the "zero" position, with all the actuators at null stroke. During start-up and shutdown of the simulator other specific commands are sent to the FC.

12 Simulator Integration and Testing

The integration of the overall MORIS Simulator has been carried out as follows:

1. Integration of all the software modules: all the elaborating units purposely developed to the control, real time and console, have been connected and several tests have been conducted in order to test their functionality (Figure 12-1);
2. Integration of the Graphical Subsystem: the SGI computer has been connected at the previous net and tested with the retro-projector and the screen (Figure 12-1);
3. Integration with the Actuation Subsystem: the previous elements have been connected (via the dedicated drivers and acquisition units) to the Actuation Subsystem. First only the Mock Up, Steer and Rider Command Acquisition have been integrated performing fixed base simulation. Then, the Stewart Platform has been integrated performing overall tests (Figure 12-2).

The overall system has been carefully tested and up to now the people reported in Table 12-1 have proven it. The common main result is that no motion-sickness [6] has been claimed by all the riders.

ID	Type	Number	Judgement
1	SSSA Integrators	8	(Internal employers)
2	Other MORIS Partners	6	Realistic and exciting
3	Piaggio Internal Riders (normally testers of real prototypes)	2	Realistic. Lost of realismus replicating the braking and rolling at high frequency.
4	External Specialist Journalist	2	Realistic in particular for training purposes.
5	External People	10	No specific competencies on motorcycle simulated ride

Table 12-1 MORIS Testers

Figure 12-1 Left: computers and electrical connections; Right: the videoprojector.

Figure 12-2 The Integrated MORIS Simulator (left: side view, right: rear view)

13 Conclusions

In the presented papers, the description of the MORIS Simulator has been performed. Starting from the general architecture of this complex device, all the various subsystem have been analysed and described, with particular emphasys on the basic solutions adopted and the main functionalities. Finally, the integration of the simulator and the field tests with different users, have been reported.

14 Acknowledgements

The work described in this paper has been carried out by the consortium in the framework of the ESPRIT n.20145 Project MORIS (Motorcycle Rider Simulator) funded by the European Union.

15 References

[1] W. Kading; “The Advanced Daimler-Benz Driving Simulator”, SAE 9530012, 1995.
[2] H. Soma and K. Hiramatsu; “Driving Simulator Experiment on Drivers’ Behaviour and Effectiveness of Danger Warning Against Emergency Braking of Leading Vehicle”, Proceedings of 16th ESV, Canada, 1998.
[3] Y. Miyamaru; “Development of a Riding Simulator and its Prehistory”, Preprint of JSME 59-00, 2000.

[4] K. Yoshimoto, D. Kawasaki, Y. Murakami, T. Sugimoto, S. Chiyoda and D. Ferrazzin; “Development of a Motorcycle Simulator Using a Parallel Manipulator and a Head Mounted Display”, Proceedings of DSC2000, France, 2000.
[5] J. W. Zellner and D. H. Weir, "Development of Handling Test Procedures for Motorcycles", SAE780313, 1978.
[6] D. Ferrazzin, F, Barbagli: “MORIS: Simplified Dynamic Model”, technical report MO-SS-ME-D-SDM-00, 2000.
[7] D. Ferrazzin, O. Toscanelli, F. Salsedo, A. Frisoli, M. Franceschini, M. Bergamasco, “The Stewart Platform of the MORIS Simulator”, Proceedings of PKM99, Milano, 1999.
[8] M.Di Natale, G.M.Prisco, G.Di Pietro, M.Bergamasco, P.Ancilotti "The Design and Analysis of the MORIS Simulator" Proceedings of the 1999 RTAS - Real Time Applications Symposium, June 1999, Vancouver BC, Canada.
[9] F. Barbagli, D. Ferrazzin, C.A. Avizzano, M. Bergamasco, “Washout Filter Design for a Motorcycle Simulator”, Proceedings of VR2001, Yokohama, 2001..
[10] M. Idan, M. A. Nahon, D. Sahar, “A Comparison of Classical and Robust Flight Simulator Motion Control”, AIAA Proc. Of Conf. On Flight Simulator Technologies, 1998.
[11] J. B. Sinacori, “A Practical Approach to Motion Simulation”, AIAA Visual and Motion Simulation Conference, Palo Alto (CA), 1973.
[12] J. Helmann, "IRIS Performer: a High Performance Multiprocessing Toolkit for Real-Time 3D Graphics", Proceeding of Siggraph, 1994.
[13] I. P. Howard, “The vestibular system” in “Handbook of Perception and Human Performance”, John Wiley, 1986.
[14] J. Blauert, “Spatial Hearing”, The MIT press, 1983.
[15] H. Lehnert and J. Blauert, "Principles of Binaural Room Simulation", Applied Acoustic, nr 36, 1992.
[16] D. Ferrazzin, P. Moncini, G.M. Prisco, F. Salsedo, M. Bergamasco, “Inertial Force Feedback in a Motion Based Twowheeled vehicle simulator”, Proceedings of RO-MAN, 1997.