Proceedings of the 2007 IEEE
Intelligent Transportation Systems Conference
Seattle, WA, USA, Sept. 30 - Oct. 3, 2007

A new environment for distributed multiple vehicles dynamics control and simulation

Francesco M. Raimodi, Ludovico S. Ciancimino

Abstract—**Although there are many simulation softwares, the 3D multi-vehicle field needs further study. This paper describes a lightweight, full portable software environment for development, simulation and control of vehicles. This environment simulates ground wheeled vehicles and their environment. Multi vehicle study can be done adding the vehicle mathematical model and the 3D graphical model. The scenario is displayed as a 3D virtual reality environment in which all the objects are rendered using a combination of 3D primitive and/or pre-built 3D objects loaded from file. Since the software is made up of a set of multi-treading object-oriented java classes, there are many advantages as portability, efficiency and distribution. Communication between two or more cooperating entities is implemented using efficient Network Manager and both TCP and UDP protocols.**

I. INTRODUCTION

The control of a single or groups of heterogeneous UGV (Unmanned Ground Vehicle) is a field in which there are many open discussions. Since tests of this type of vehicles are very expensive, simulation becomes a very important phase of the development process. Although there are many tools to simulate generic systems, many problems are still open. One of these problems is the efficient 3D representation, manipulation and visualization of the object to be controlled and the environment it is immersed in. Other problems are the ability to manage fleets of vehicles efficiently, to compute the model of the system, to control and to manage the communication required by cooperation. As well-known, there are many simulators (e.g. Matlab or Modelica) but, since they are generic, they do not implement satisfying modules for the control of groups of heterogeneous UGV. Generally, they do not present good cooperation, communication and 3D capability (in terms of customization, portability, efficiency and reusability). For example, it is interesting to use the same 3D engine both to simulate the systems and to control them in a real application. It is also favourable to control the system both locally and remotely from any place and any mobile hardware (e.g. by web server). It is favourable a low level access to the communication stack for fast communication (to use UDP).

In literature there are many approaches to these problems. In [9] a multiple-telerobot system is presented in order to remote control cooperative robots but it uses C++ for core functions and java applets to display VRML (Virtual Reality Modeling Language) representation of the remote scenes. The VRML objects are interpreted so its performances are low with respect to other libraries such as OpenGL[6] or MS Direct3D that uses the GPU (Graphic Processing Unit) of the video card to speed-up the 3D operations. In [1] it is presented a realistic simulation and visualization environment tool to be used as a test-bed for performance evaluation of different control and coordination strategies applied to single and teams of cooperative heterogeneous UGV (aerial and ground). It uses the Matlab/Simulink environment for the computation of the control algorithm and the VRML for the 3D visualization. Furthermore, Matlab is an expensive complex package that needs to be installed before the utilization and it also requires fast hardware. In [9] a network sensor and network managers are presented in a software architecture for web control.

In this paper a new Java based environment for vehicle fleets control and simulation is presented. It uses free Java API in order to allow its execution, without installation, on every network-capable machine able to run java 1.1 applications or web Applet (as pocketPC). In order to build a distributed control environment, the presented software architecture is based on a set of object oriented modules. Since these modules must be independent and, as distributed system, they can work in different locations and run/work in parallel mode (avoiding long sequential execution problems). In order to simulate real vehicles, its complete model (i.e. its differential equation system) is used with the "ODE Solver" module. To display the environment where the vehicle runs, a 3D virtual reality representation is used. The control data are transported with both TCP and UDP protocols and managed by a "Network Manager" (NM).

This paper is organized as follow: the proposed architecture in section II. The modules are described in section III. In section IV some examples are presented.

II. CONTROL ARCHITECTURE

A. General concept and Server Side

The control and the simulation of multi vehicles can be very hard and it requires a very complex development. For these motivations, in order to subdivide the problems and the tasks, object oriented module decomposition is favourable.

The presented vehicle control environment (see figure 1) is composed of different interacting modules collaborating to

Manuscript received February 27, 2007.
Francesco M. Raimondi and Ludovico S. Ciancimino are with the Dipartimento di Ingegneria dell'Automazione e dei Sistemi, University of Palermo, Viale delle Scienze 90128, Palermo (Italy).
(corresponding author e-mail: ludovic@dias.unipa.it).

a common objective. Since these modules are independent and work in parallel mode, the global vehicle control performance can very high. Every module is specialized on a particular task and can be activated or not independently of the others because it lives in a separated thread. Some modules are generic and must be extended with the characteristics of the particular vehicle to whom they are applied. For example, the trajectory planning and tracking modules can require the vehicle model.

The environment architecture is based on a client server concept in which the server collects and maintains the full world data (all objects) and the clients control and manage a single or a group of vehicles. The server is the center of the control system because it manages a complex representation of the environment (in which others vehicles can be found). As a bridge, the server gets the command from the client and sends it to the remote vehicle, gets the response and sends it to the client. In the client station a 3D virtual reality interface shows the virtual representation of the world in which the vehicle is immersed. The object oriented development implies that most modules can work in separate machines connected by a LAN (or Internet) and communicate using the server that manages the control data flow. Since the communication is bi-directional, the vehicles can be controlled both from clients and server (remote autoPilot).

The client interface is multipurpose and so it can manage different and/or more vehicles simultaneously. For example, the manual vehicle control, by client interface, is done using the keyboard so, in order to pilot many vehicles, the interface can switch the key pressed action to different vehicles. Another possibility is the Auto Pilot designed to perform some of the tasks of the pilot (e.g. to run straight without a pilot's attention). In order to allow control and simulation analysis, the data collected during an experiment are stored on text files.

These requirements suggest, for the implementation of the proposed tool, the use of a network-ready object-oriented programming language. Portability, the existence of many free API and performances make Java the best chose.

In the next paragraph are described the presented control environment characteristics.

B. Client Side

The on board vehicle (on vehicle side) program (called Vehicle Driver VD) is controlled (using keyboard, mouse or joystick) by the user (user side). It also allows the use of the extension modules (such as OdeSolver, trajectory planning and tracking etc). In order to optimize the VD resource (e.g. memory, computation capability), it can have a restricted knowledge of the real and virtual world. In this situation, the VD can speed-up its task, increase the dataServer sampling-time, increase its algorithm complexity etc... The user can also view the real remote world by a video client.

The client interface can work in two different ways: remote vehicle controller or vehicle simulator. In the first case, the client program runs in a PC placed on the real vehicle to be controlled and it is connected directly to the vehicle (e.g. by RS232 serial interface, USB, FireWire etc.). The client, via the above connection, gives the vehicle input reference and gets onboard sensor output (low level control loop) then, via the server connection (e.g. Ethernet), it communicates and cooperates with other vehicles using the complete world managed by the server. The vehicle is remote controlled by the user from another client connected to the server. In this scenario, the proposed control environment works as an efficient infrastructure for the remote controller of a fleet of vehicles. As example, in figure 1 a control schema for the vehicle number i (where i = 0, 1.. n) is showed.

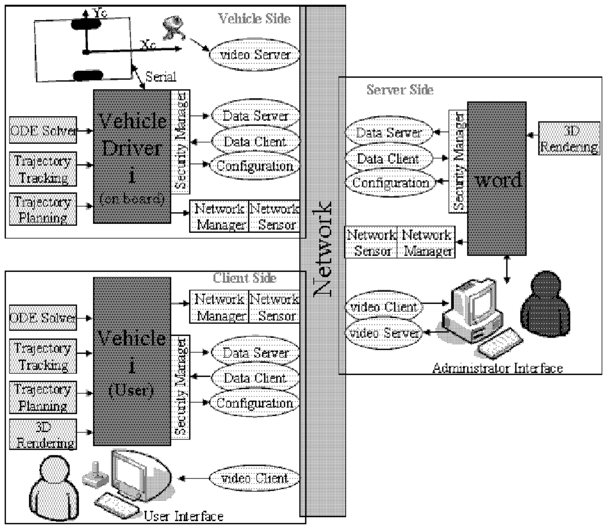


Fig. 1. Fleet control using the vehicle control environment

In the second case, since there is no real vehicle, the vehicle side does not exist so the state evolution of the vehicle is calculated using the ODESolver on client side. The client simulates the vehicle and sends the calculated state evolution of the vehicle to the server like the real onboard vehicle sensor. The schema of the vehicle simulation is very similar to the schema of figure 1 with the exception that the vehicle does not exist. It is also possible to use both modalities in order to evaluate, for example, real vehicle control with the “virtual presence” of fleets of simulated vehicles or vice versa.

III. VEHICLE CONTROL ENVIRONMENT MODULES

A. ODE solving and computation

In order to simulate a real system, it is necessary to use a mathematical model of the real system (i.e. a system of differential equations) and a solver (i.e. a tool that can compute the model solving the equation). In Matlab, this task is simple because it contains an ordinary differential equation (ODE) engine but java does not contain built-in routines able to solve differential equations. Fortunately in [2] a free java library also containing routines to integrate and solve single or systems of ordinary differential equations is presented. In order to solve the equations of the systems under simulations, the "ODESolver" module is built using the Runge Kutta code presented in [2]. With ODESolver it is also possible to simulate the controller (e.g. PID, actuators etc.) of the systems under simulation foreseeing the state of the system at the given time. See section IV for example of ODESolver employ.

B. 3D manipulation, computation, and visualization

Since originally Java does not provide any built-in support for 3 Dimensional (3D) object management, the development community presents many extensions API. Java3D [3] is one of the best known 3D extensions for java; it has got the support of the Sun but it requires at least a JVM version 1.5 and it must be installed separately. There are also several 3D extensions for Java but, for the purpose of this work, jPCT [4] was selected. It offers very fast and portable rendering it is fully compatible with all Java 1.1 VMs like the Microsoft VM and even the old Netscape 4 VM. jPCT can render into a native (also full screen) OpenGL [5] window, it uses the software renderer in cases where no OpenGL compatible hardware is available and it can use the hardware renderer in the place where it is. In this case, it uses hardware acceleration (by the LWJGL [6]) under Windows, MacOS X and Linux. It also runs on other platforms (e.g. Windows CE) for portable devices useful for the supervision/control of the vehicles from any place. The JPCT world is structured as a collection of interacting 3D objects and Virtual Reality tools like Camera and lights. As example, a car-like vehicle 3D object requires 5 primitives: one box as base and 4 cylinders as wheels (see figure 9).

jPCT can also manipulate 3D objects built in various format: the well know Autodesk Ltd. 3D Studio (.3ds), the model format of Quake II by ID-Software (.MD2), the simple ASCII-format of 3D-Studio (.ASC) and a jPCT's proprietary XML-based format (.xml). Since jPCT can load 3D objects from file it is possible to build a very realistic virtual reality world adding complex objects representing vehicle, terrain, building, people etc.

C. Entities Communications

Communication in cooperative field is a key factor and needs to be developed accurately. In order to ensure fast and efficient information exchange it is important to have reliable communication both in wired and in wireless network and also on high delay networks. Communication must be robust even in presence of disturbs like unpredictable delay, lost messages and high traffic. The TCP protocol [7] is generally used in communication because it is reliable (i.e. a message is retransmitted until it is received correctly). Furthermore, in real-time communication, this characteristic is deprecated because TCP, if an error occurs, tries to transmit an old message instead of the new ones. To avoid these problems, UDP protocol is used instead to transmit real-time data and TCP is used only for configuration information exchange. In our work, the real time information flow concerns the state of the system (i.e. the state of all objects) and it is managed by the dataServer and dataClient. The state of an object (vehicle, manipulator etc.) concerns position (x, y, z), orientation, speed etc. The configuration information flow (non real-time) concerns setting (like UDP, TCP port, encryption agreement, log-in etc.) and it is managed by the "Configuration Channel" (see figure 1 and 2).

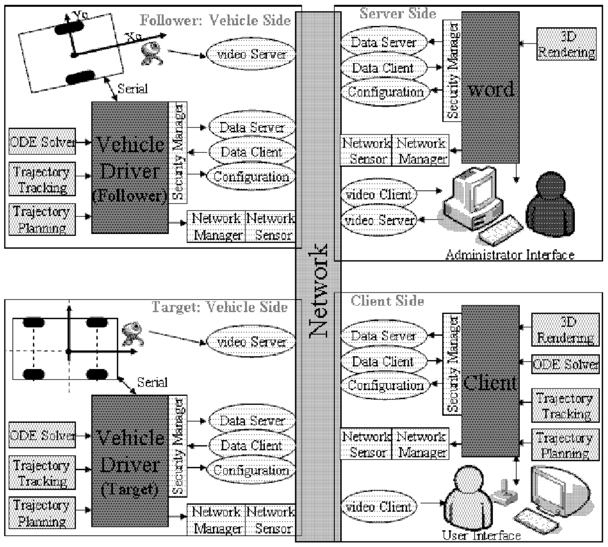


Fig. 2. A two vehicles rendezvous control schema

Control and simulation data can flow, under the network communication link, in different types of aggregation. For example, they can flow all together (in a single packet flowing under a single socket) or all separated (in separated packets, flowing under separated sockets, one for every variable). In the first case, the packet to be transmitted is built putting all the variable objects (sampled synchronously) multiplexing in a format allowing the understanding at destination after the de-multiplexing (as example adding special chars like " ", ";" etc.). Since some recursive algorithms (such as Kalman filter [10]) require constant sampling time, this approach is necessary in case such algorithm is used. In fact, if the sampling time changes, the Kalman filter must be re-initializated. In the second case, every variable is transmitted in separate packets without understanding and sampling synchronization problems. Because no synchronization is required, the sampling time can be chosen different for every variable (e.g. the data rate of transmission can be augmented or reduced depending on the necessity by adjusting the sampling time).

In this paper, the second strategy is used to: speed-up the communication, simplify multiplexing and de-multiplexing of the variables, simplify the variable manipulation on client and server side.

D. Network sensors and Network Manager

If the network does not support the data rate required by the control system, it can become a bottleneck for the data exchange. To avoid this problem it is important: 1) to detect network inefficiencies; 2) to solve the problem adapting the data rate with respect to the current network capability. In order to monitor the network capability, a Network Sensor (NS) [8] detects the Round Trip Time (RTT) between the server and the clients. If the RTT is very short, the connection is very fast and the control quality is good but, if it is too long, probably the connection is poor such as the control quality. If the network capability is low (high RTT) to increase the control quality, the data rate of the control communication can be reduced (reducing the network load the RTT can be lowered). The data rate can be adjusted changing the sampling time of the data servers and the sampling time and quality of the video server. No adjustment is required on the Client because it only receives what the server sends. These operations are executed by the Network Manager (NM). If the network quality is low, the NM adjusts the data rate of the (client's or server's) dataServers using the NS of the RTT (between the client and the server).

E. *Vehicle and Entities Cooperation*

Cooperation to manage a fleet of vehicles is a key factor and needs to be developed accurately. First of all, it is necessary to study how to subdivide the control task to the fleet of vehicles and then which information must be shared between the vehicles. In this paper "dataServer" and "dataClient" modules are used in order to transmit cooperative information. For example, if a vehicle rendezvous is necessary, it is important that the Target vehicle positions are acquired by the Follower vehicle. To accomplish this task, the Target vehicle transmits the Target data (position, direction etc.) to the server and then the Server re-transmits to the Follower client. The Follower can plan its rendezvous trajectory using these data.

In some situations, the network can have a too long RTT so, in order to avoid long control delay problems, the Follower and the Target can communicate directly (without the server). This configuration does not require a particular project variation, in fact it is sufficient to configure the dataServers (of the Target) in order to send their data directly to the Follower (or to send both to the Follower and to the Server). Obviously, in this case the server's dataServer does not transmit Target's data to the Follower.

F. *Security Manager*

In some situations, communication for cooperation must be used carefully. Security on a distributed control system is very important because the data flowing under the LAN or Internet can be intercepted (sniffing), modified and so on. As example, in military operations the information flow can be intercepted by the enemy so it must be encrypted. In order to prevent problems caused by malicious/unauthorized access, the "Security Manager" (SM) is used (see figure 1 and 2). Since java has got a set of built-in class/functions to manage security, the java language is also used to develop the SM functionality. The SM has got two main assignments: 1) authentication; 2) real-time data encryption.

1) Since only the authorized user can successfully login the system the authentication is necessary. The login information (non real-time) flows under the configuration channel using a secure login protocol.

2) In order to prevent data "sniffing" (malicious interception of the data passing over the network), the SM encrypts all data flowing under the client-server communication channels (the VD to vehicle communication channel does not also require this precaution since it transports low level data). The encryption algorithm must be very fast because encryption/decryption times are delays of the control loop.

G. *Simple trajectory planning test algorithm*

In this section, in order to test the presented control environment, a very simple trajectory planning module is presented. Trajectory planning is very complex, for many reasons: environment complexity (obstacles, vehicle shape), dynamics complexity (equations with many states). The proposed algorithm plans a circular rendezvous trajectory that drives Follower vehicle to an assigned target vehicle (Target), supposing that there are no obstacles. This algorithm is based on a simple observation: since the vehicle is a two wheels non-holonomic vehicle (see figure 3), a possible trajectory is a solution of its equation system.

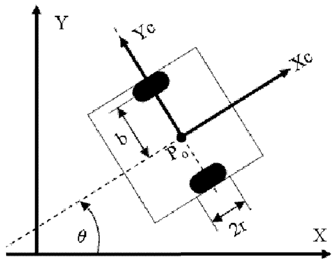


Fig. 3. The two wheel vehicle schematic representation

A two wheels non-holonomic vehicle, as it is well known [11, 12], can move only along the direction of the orientation (in hypothesis of no slipping and pure rolling kinematic constraint) and its kinematics equation system is equal to a unicycle-type vehicle:

$$\dot{x} = v \cdot \cos\theta, \qquad \dot{y} = v \cdot \sin\theta, \qquad \dot{\theta} = \omega \tag{1}$$

where x, y are the Cartesian coordinates of the vehicle in fixed X,Y Cartesian frame (world coordinate), θ is its orientation angle with respect to the same frame and v, ω are the robot linear and angular velocities.

The inputs of the vehicle are v, ω but it is important to give the relation between the control input and the speed of the two wheels (V_L= left wheel speed, V_R= right wheel speed):

$$V_L = r \cdot \omega_L \quad , \quad V_R = r \cdot \omega_R \tag{2}$$

$$\omega = (V_R - V_L) / (2 \cdot b) \quad , \quad v = (V_R + V_L) / 2 \tag{3}$$

where ω_L and ω_R are the wheels angular velocities, r is the wheel radius. Substituting (3) in (1), the next position of the vehicle can be calculated.

It is simple to observe that possible solutions (and trajectories) are circles and straight lines. To simplify we choose, as rendezvous trajectory, the only circle that starts from the actual vehicle position, is tangent to the vehicle's direction, ends on the Target position and is covered in the actual vehicle direction. In figure 4 two examples of planned trajectory (in red dotted line) are painted for two different vehicle initial directions.

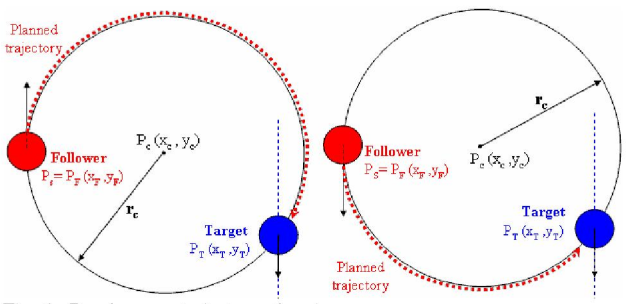


Fig. 4. Rendezvous trajectory planning

These planned trajectories are simple to track because they imply constant linear and angular vehicle speed. In order to compute these trajectories, the first step is the compute of the circle i.e. its center $P_C(x_C,y_C)$ and radius r_C. This circle can be computed calculating first the bundle of circumferences tangent in Start position $P_S(x_S,y_S)$ (equal to Follower position $P_F(x_F,y_F)$ of figure 4) to the equation of the vehicle direction and then imposing the passing to the target position $P_T(x_T,y_T)$ (equal to the Target position P_T).

The bundle is obtained calculating the linear combination of two generating circles. The first generating circle is obtained computing the bundle of circumferences tangent in P_S to the equation ($y = m \cdot x + q$) of the vehicle direction α so the angular coefficient m is (where q is influential):

$$m = \tan \alpha \tag{4}$$

The equation of the straight passing in P_S and tangent to the bundle of circumferences is:

$$y - y_S = m \cdot (x - x_S) \tag{5}$$

The second generating circle is a circle passing in PS and having radius equal to 0:

$$(x - x_S)^2 + (y - y_S)^2 = 0 \tag{6}$$

Then the linear combination of (5) and (6) gives the bundle equation:

$$x^2 + y^2 - 2 \cdot x_S \cdot x - 2 \cdot y_S \cdot y + x_S^2 + y_S^2 + k \cdot [m \cdot x - y + (y_S - m \cdot x_S)] = 0 \tag{7}$$

Imposing the bundle passing on P_T, we obtain:

$$k = -\frac{x_T^2 + y_T^2 - 2 \cdot x_S \cdot x_T - 2 \cdot y_S \cdot y_T + x_S^2 + y_S^2}{(m \cdot x_T - y_T + y_S - m \cdot x_S)} \tag{8}$$

Substituting the numeric value $\tilde{k}$ of k (obtained replacing the coordinates of P_T and P_S in (8)) in (7), the circle characteristics are:

$$x_C = -(-2 \cdot x_S + m \cdot \tilde{k}) \cdot 0.5 \tag{9}$$

$$y_C = -(-2 \cdot y_S - \tilde{k}) \cdot 0.5$$

$$r_C = \text{dist}(P_C, P_S) = \text{dist}(P_C, P_T) = \text{dist}(P_C, P_F) \tag{10}$$

The circle is calculated and the trajectory, can be planned using, in sequence, the (4), (8), (9) and (10).

In order to plan the trajectory all points between P_S and P_T must be calculated. To calculate these points, first the appropriate speed of the vehicle is chosen, then the trajectory (sequence of $[x, y, \theta, t]$) is calculated following the planned trajectory (i.e. the circle give by the (9) and (10)).

With regarding to the tracking of the trajectory, as for the planning, there are many approaches but, in this case, we use a simple algorithm. The reference linear and angular vehicle speeds ($v_r(t)$ and $\omega_r(t)$), used for the vehicle control law $v_c(t)$ and $\omega_c(t)$, are computed using the planned reference trajectory as in [13].

$$v_r(t) = \pm\sqrt{\dot{x}_r^2(t) + \dot{y}_r^2(t)}$$

$$\omega_r(t) = \frac{\dot{x}_r(t)\ddot{y}_r(t) - \dot{y}_r(t) \cdot \ddot{x}_r(t)}{\dot{x}_r^2(t) + \dot{y}_r^2(t)} \tag{11}$$

More complex algorithms can be implemented extending the basic trajectory tracking algorithm of the module.

IV. EXPERIMENTS WITH VEHICLES

In this section a set of vehicle control examples is described. The aim of these examples is the demonstration of the capability of the proposed vehicle control environment.

A. *Vehicle Remote Simulation and Control*

In this paragraph, a simple two wheels non-holonomic vehicle remote control is presented. First the vehicle is simulated using ODESolver with the vehicle kinematic model. In order to utilize a vehicle model (i.e. (1)), the class model is extended writing the "derivn" method that contains the equation system (see figure 5).

```
public double[] derivn(double x, double[] y){
double[] dydx = new double [y.length];

dydx[0]= v*Math.cos(y[2]);//dx/dt=v*cos(theta)
dydx[1]= v*Math.sin(y[2]);//dy/dt=v*sin(theta)
dydx[2]= omega;          //dtheta/dt =omega
return dydx;
}
```

Fig. 5. Kinematical model implementation for OdeSolver

In the second part of the experiment, the vehicle is remotely controlled. The remote vehicle is equipped with the VD (see figure 1) that gets the command (e.g. V_R, V_L velocity reference signal of the two wheels) and sends to the server the output of the onboard sensor (e.g. V_R and V_L from the wheels sensor, the obstacles distance and the live video of the environment in which the vehicle is absorbed).

B. Two vehicle cooperation for rendezvous

In this section the control environment is used to simulate a simple vehicles rendezvous. Let us take two vehicles: a 4 wheels car-like vehicle as Target and a 2 wheeled vehicle called Follower. Let us suppose that the Target (at the time t_0) position is (x_{t0}, y_{t0}) and its speed and direction are v_{t0} and theta$_{t0}$. The objective is the rendezvous of the two vehicles at the time t_n at the position (x_n, y_n) (estimated future Target position). The problem is the planning and tracking, by the Follower, of a trajectory that leaves the Follower to the Target estimated position. Since the Target is in motion, the rendezvous point can change during the Follower motion so the rendezvous trajectory must probably be re-planned to the new estimated target position.

In this example, the trajectory is planned with the algorithm presented in section III.G using the Follower position as Start position and the Target vehicle position as target position (see figure 5). The two vehicles are simulated using the appropriate mathematical model with the ODESolver module. If, during the Follower motion (see figure 6), the Target vehicle (e.g. driven by the user) changes motion characteristics, at time t_2 a new Target estimated position is computed and a new trajectory is calculated to rendezvous the Target (at t_3). In figure 7, two overlapped screen-shots of the client interface frame (in wire-frame and rendering mode) during the rendezvous are showed.

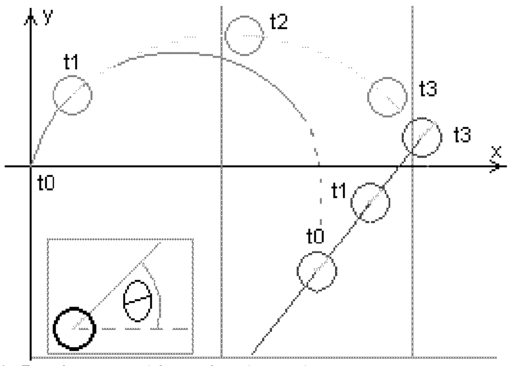
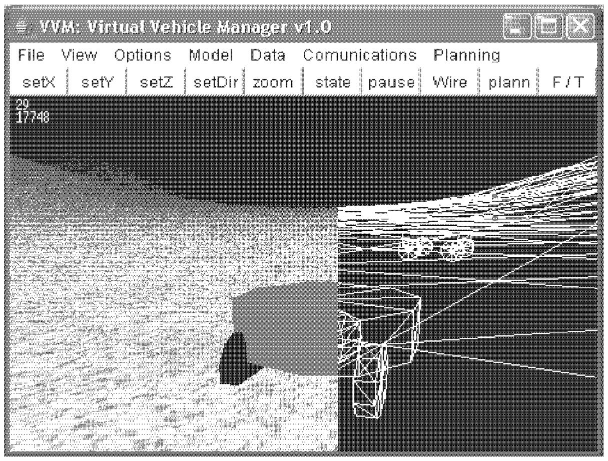


Fig. 6. Rendezvous with re-planning trajectory

V. CONCLUSION

In this paper, a java based environment for multi vehicle development, simulation and control is presented. The vehicle control environment allows the control and/or the simulation of fleets of vehicles in the real world. Since the vehicle control environment works as a highly efficient distributed control system, the real-time control signals are transmitted using separated UDP/IP channels and non real-time data (like configuration data) are transmitted on reliable TCP/IP channels. These features allow fast and efficient cooperation between the vehicles and the server. In order to manage different network qualities, the network sensor and manager are employed to adapt the data rate to the actual network capability. The simulation level can be very high (in term of adherence to real vehicle dynamics and obstacle interaction) since complex non-linear models can be used. The user has a good perception of the scene because he can view a live video of the real vehicle point of view and/or a virtual reality representation of the real environment.


VVM: Virtual Vehicle Manager v1.0
File View Options Model Data Comunications Planning
setX setY setZ setDir zoom state pause Wire plann F/T
29
17748
Fig. 9. Rendezvous' screen shot

REFERENCES

[1] Castillo-Effen, M.; Alvis, W.; Castillo, C., "Modeling and visualization of multiple autonomous heterogeneous vehicles", IEEE International Conference on Systems, Man and Cybernetics, 10-12 Oct. 2005, Vol. 3, Pages: 2001 - 2007.
[2] M. T. Flanagan's Java Library, Michael Thomas Flanagan's Java Library, www.ee.ucl.ac.uk/~mflanaga/java, 31 December 2006.
[3] Sun Microsystems Inc., "java 3D API home page", http://java.sun.com/products/java-media/3D, 2006.
[4] jPCT: the free 3D engine/API for Java, www.jpct.net, 2007.
[5] Open Graphics Library (OpenGL), www.opengl.org, 31/01/2007.
[6] LightWeight Java Game Library (LWJGL), http://lwjgl.org, Jan. 2007.
[7] "RFC 793 (rfc793) - Transmission Control Protocol, 2.6 Reliable Communication", page 9, www.ietf.org/rfc/rfc0793.txt, 1981.
[8] P. Le Parc, J. Vareille and L. Marcé, "WEB remote control of machine-tools the whole world within less than one half-second", 35th International Symposium on Robotics (ISR2004), Paris, (France), 23-26 March 2004;
[9] Xiao-Gang Wang; Moallem, M.; Patel, R. V.; "An Internet-based distributed multiple-telerobot system", IEEE Transactions on Systems, Man and Cybernetics, Vol. 33, Issue 5, Sept. 2003 Pages: 627 – 634.
[10] Rudolph E. Kalman, Introduction to the Kalman Filter, www.cs.unc.edu/~welch/kalman, 2007.
[11] Wang, Z. P.; Su, C. Y.; Lee, T. H.; Ge, S. S.; "Robust adaptive control of a wheeled mobile robot violating the pure nonholonomic constraint", 8th Conference on Control, Automation, Robotics and Vision, (ICARCV), Volume 2, 6-9 Dec. 2004 Pages: 987 – 992.
[12] Minor, M. A.; Albiston, B. W.; Schwensen, C.L.; "Simplified motion control of a two-axle compliant framed wheeled mobile robot", IEEE Transactions on Robotics and Automation, Volume 22, Issue 3, June 2006, Pages: 491 – 506.
[13] Gregor Klančar, Drago Matko, Sašo Blažič, "Mobile Robot Control on a Reference Path", Proceedings of the 13th Mediterranean Conference on Control and Automation, Limassol, Cyprus, June 27-29, 2005.