TELEDIMOS[*]
Telepresence simulation platform for civil work machines : real-time simulation and 3D vision reconstruction

D. Simon, M. Personnaz and R. Horaud
INRIA Rhône-Alpes, ZIRST, 655 avenue de l'Europe, MONTBONNOT
38334 SAINT ISMIER CEDEX, FRANCE

Abstract The TELEDIMOS project addresses issues related to the design, development and demonstration of a real time simulation system to support robotic telepresence platforms. We describe here two building blocks of the project: a real-time simulation of the robot dynamics and a vision system able to accurately build an elevation map of the robot environment.

1 Introduction

The purpose of the TELEDIMOS project is to address issues related to the design, development and demonstration of a real time simulation system to support the control of remotely operated robotics systems. The applications we consider concern for example post-war civil buildings dismantling, nuclear plants decontamination or demining . . . . The considered robotic system is a large scale excavator. A demonstrator based on this robotic system will be integrated in order to illustrate the results of this project. The design and development of the whole control architecture will be considered in the project but is out of the scope of this paper.

The simulation system will be integrated in an operational architecture, in order to support:

- the 3-dimension reconstruction of the remote site viewed from the robot,
- the specification and validation of the tasks to be accomplished by the robot,
- the prediction of the execution of the validated tasks, and finally,
- the monitoring and control of the tasks to be executed by the robot.

Usually, simulation in robotics is used for validation purpose before real operations. In our approach the simulation environment is a central feature in the robot monitoring and control system, as an interactive support for the operator from the specification of the tasks to their execution.

The required functionalities of such a system are: locomotion and manipulation capabilities, perception of the surrounding environment, localization, communication capability with the master station. The sites where the robot operates are partially known and the typical operations to be performed (navigation, positioning, cutting, digging . . .) are mostly unpredictable. In addition, the environment is dynamic since most of these operations change it during execution. Thus we need updated perception of the robot surrounding environment, continuous re-planning of the tasks to be executed, on-line validation and prediction of the execution results, and monitoring and control of down-loaded operations.

The paper deals with the tasks devoted to Inria, i.e. real-time simulation of the machine and the environment 3D reconstruction.

[*]This work is supported by the European Community under grant IST 1999 10116

2 Simulation-based Control

The operator will be able to specify the task he wants to be performed by the robot system (as, for example, the excavator), to validate and check its results in simulation, and finally to request its execution. However, the task will be communicated to the system using higher level commands, (such as end effector motion / force coordinates) while a parallel simulation process on the robot system will translate these commands to input data for individual joint operation. Both simulation models (on the robot as well as on the Master Station) will be based on the robot kinematics and dynamics and will be fed with real time machine state parameters such as joint positions, internal forces in actuators ... Finally, the robot environment will be reconstructed by combining existing data (if available) on a digital elevation map, with information collected locally by the robot perception subsystem and transmitted to the Master Station in real time.

The main components of the simulator are (Figure 1):

- The numerical integration subsystem which takes as input the desired trajectory or end-point specified by the operator, integrates the equations of the robot's dynamic, computes the inverse-kinematic and kinematic of the robot and produces the expected actual trajectory as output. This sub-system is designed using the ORCCAD system ([5]) where the Physical-Resource component uses an efficient numerical integrator instead of providing drivers to the real machine.
- The control panel which allows the operator to manage the simulator, e.g. to choose a particular trajectory generator, duration of motion, to start or stop the simulation, to reset the simulation, to set the excavator controller in manual mode ... , according to events defined in the simulation controller specification. According to the ORCCAD approach this logical behavior is specified using the ESTEREL synchronous language and can be verified using the associated FC2TOOLS formal verification tool-set ([4]).
- A communication system which allows data communication between real-time tasks (the simulator and the its controller), interactive tasks (the control panel) and the 3D-Viewer developed in the project. Interactive control of this subsystem will be done through a graphical panel sketched in Figure 3.

2.1 Design of the Simulator

Trajectory generation The joint space trajectory generator component basically implements the following algorithm:

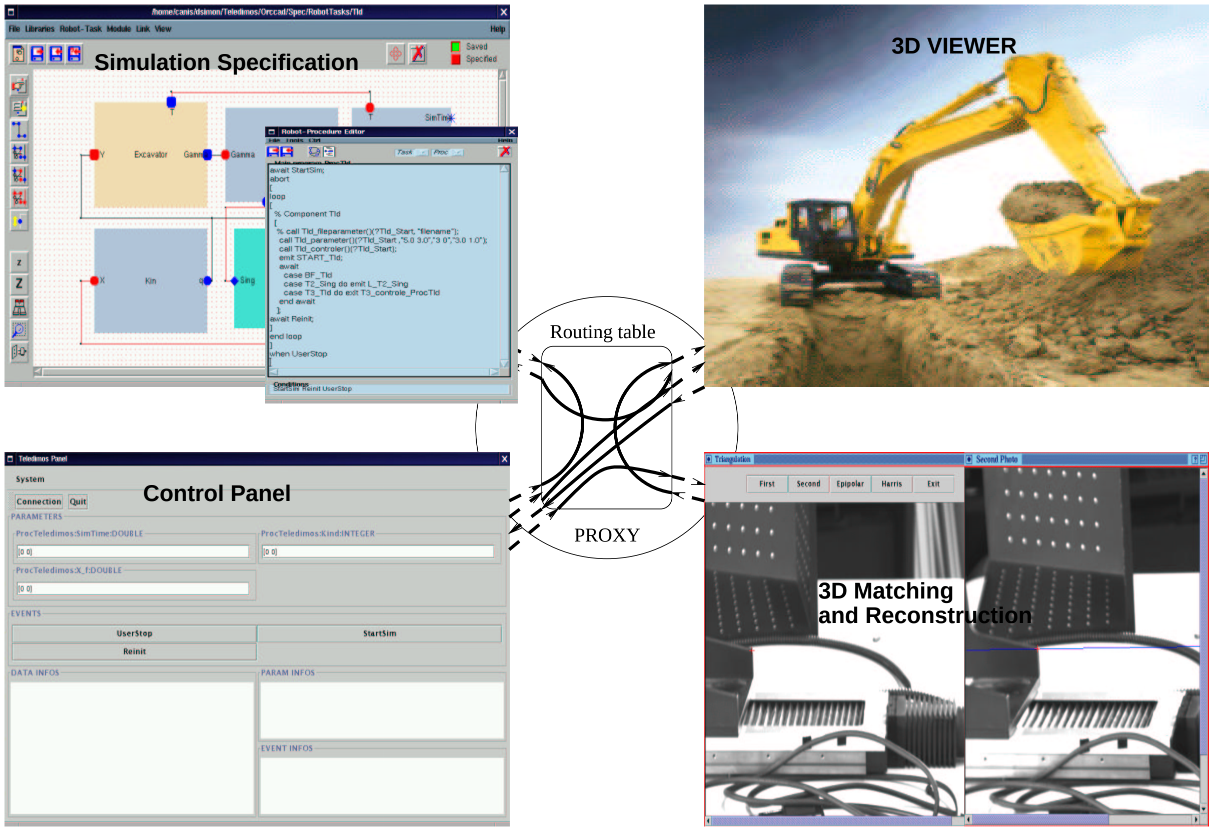
- Computing the desired sampled pose according to a timing law: $X_d[i] = X_i[i] + r(X_f - X_i)[i]$ with r being a 3 or 5 degree polynomial of time to give a constant velocity or acceleration time profile. Trajectory generation passing through intermediate via-points will be added;
- Computing set points in joint space according to the system's kinematics. The current implementation allows for joint limits avoidance according to [10]:

$$dq = J^{+}dX + \alpha(I_n - J^{+}J)\nabla\phi$$

where dq and dX are the position increments in joint and operational spaces, J is the Jacobian matrix of the system (defined by $\dot{X} = J\dot{q}$), J^{+} is the pseudo-inverse of J, α is a negative weight and ϕ is a scalar function to be minimized. In the current prototype ϕ is choosen to find trajectories away from joint limits:

$$\phi = Max\frac{[\sum_{i=1}^{n}|q_i - q_{imoy}|^{p}]^{1/p}}{|\Delta q_i|}, i = 1...n, p = 6$$

Finally, rather than computing an explicit inverse of the geometric model we simply compute the suite of positions in joint space by $q_{d_n} = q_{d_{n-1}} + dq$.



```
Main program ProcTld
await StartSim;
abort
[
loop
[
  % Component Tld
  [
   % call Tld_fileparameter()(?Tld_Start, "filename");
    call Tld_parameter()(?Tld_Start ,"5.0 3.0","3 0","3.0 1.0");
    call Tld_controler()(?Tld_Start);
    emit START_Tld;
    await
      case BF_Tld
      case T2_Sing do emit L_T2_Sing
      case T3_Tld do exit T3_controle_ProcTld
    end await
   ];
await Reinit;
]
end loop
]
when UserStop
[
Conditions
StartSim Reinit UserStop
```

System
Connection Quit
PARAMETERS
ProcTeledimos:SimTime:DOUBLE [0 0]
ProcTeledimos:Kind:INTEGER [0 0]
ProcTeledimos:X_f:DOUBLE [0 0]
EVENTS
UserStop StartSim
Reinit
DATA INFOS
PARAM INFOS
EVENT INFOS

Figure 1: Functional decomposition of the simulator

Numerical integration The integrator uses a variable step, variable number of steps algorithm to improve simulation speed and avoid possible numerical unstability. For a given integration accuracy the simulation speed relies upon the fastest time constant of the model and the simulation speed cannot be guaranteed to be real-time.

The *lsoda* package we have elected [1] for the Teledimos simulator uses two integration methods: an explicit Adams method is used at starting time and when the system is considered to be non-stiff. The software automatically switches to an implicit (Backward Differentiation Formulæ) when an on-line test estimates that the system becomes stiff and thus that the implicit integration method becomes faster than the explicit one (and switches back conversely).

The only way to provide a predictable simulation time would be using a fixed step, implicit method integrator providing unconditional stability [3, 11]. In that case the integration step should be set at a value small enough to insure the required accuracy all along the system's trajectory. The variable step method speeds up the integration by increasing the step when possible, e.g when the system exhibits slow transients. Finally the explicit method is chosen when it is faster than the implicit method. Therefore, we can consider that this software implements a method which is always faster than the potentially predictable fixed step implicit method, and that this choice consists in the best effort to run the simulation as fast as possible given a required accuracy.

The equations of the kinematic and dynamic models of the system are automatically generated using RobotDyn, a symbolic equations generator for robots with a free reference [9].

Simulation logical behaviour The behavioural part rhythms the logical evolution of the simulator; it receives input events from the simulator environment (control panel, other Teledimos supervision system components, . . .) and reacts to them by emiting output events.

The input events allow to start (and parameterize) the simulation of a robotic task, to pause/resume or

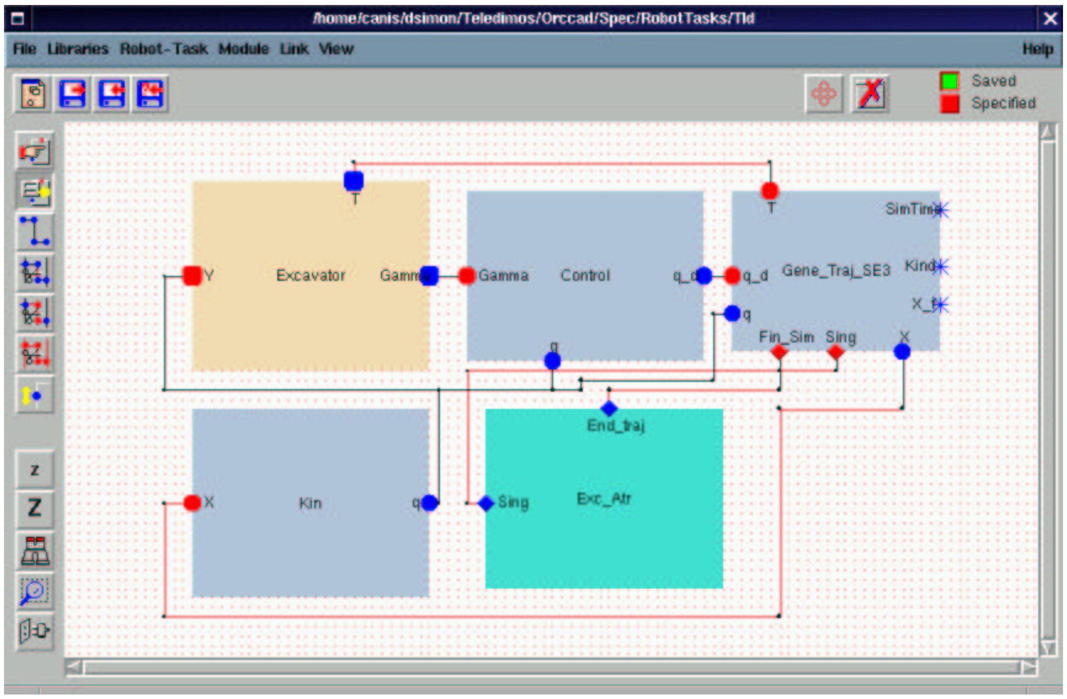


Figure 2: Design of the simulator

abort it and to indicate various types of exceptions (joint limits, ...).

The output events transmit to the simulator environment significant situations such as the good end of the simulation or a singularity crossing.

Input/output events constitute the external view of the simulator and are used to interface it with the other components of the system. The simulator behaviour is specified using the ESTEREL language (Figure 3).

After receiving the `StartSim` the application begins with an initialization phase, e.g. checking that all necessary sockets are open and reading the initial value of parameters. In normal operation the simulation run upto completion, i.e. reaching the specified duration.

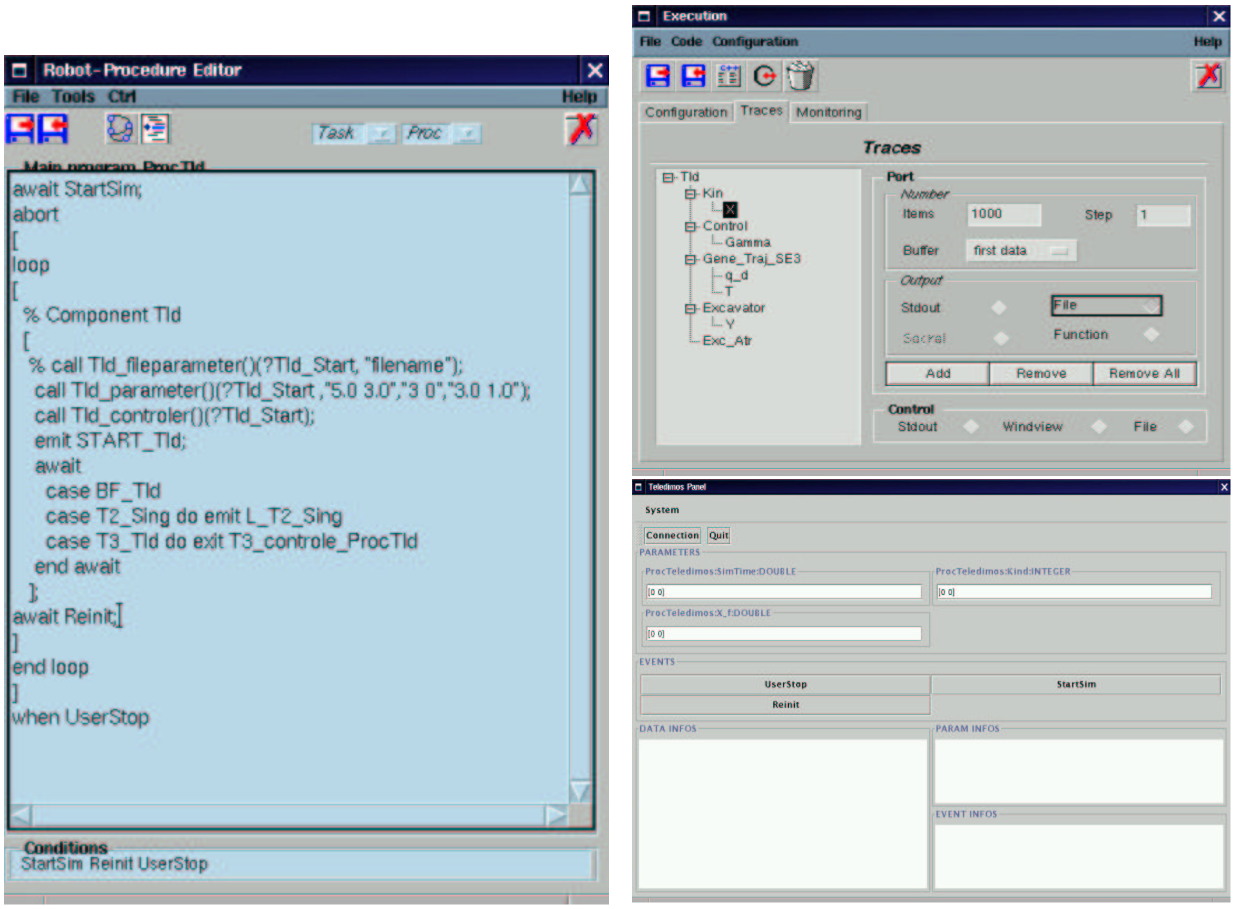
In case of failure, e.g. because the desired pose of the end-effector is out of range, the simulation stops and waits for the operator to change the simulation parameters and launch a new simulation by typing a Reinit order through the control panel. After finding a successful trajectory the operator can stop the simulation by typing a UserStop order. In a future, more complete application, the same kind of controller will be used to schedule the phases of the full system operation including system preparation, simulation, downloading and executing trajectories on the real machine, monitoring ...

Currently the parameters which can be set through the control panel are the duration of the simulation and of the trajectory, the kind of velocity profile and the value of the desired pose.

Experiments The simulator is specified using the GUI depicted by Figure 2 from which the executable code is generated using the standard run-time code generator of ORCCAD. Experiments has been performed on a PC running Linux[1] on a Pentium II 334 MHz. (The software also works with Sun/Solaris 5.6).

The sampling period has been set to 50 msec (a larger period can lead to control unstability). Results can be viewed using a 2D plotter (see samples in Figures 4 and 5). The final steady-state error is mainly due to the uncontrolled motion of the vehicule suspension

[1]Using a standard Linux system the clock resolution in bounded by 10 msec. Using more accurate clocks needs running in superuser mode or using a real-time operating system like RTLinux.



```
await StartSim;
abort
[
loop
[
 % Component Tld
 [
  % call Tld_fileparameter()(?Tld_Start, "filename");
  call Tld_parameter()(?Tld_Start ,"5.0 3.0","3 0","3.0 1.0");
  call Tld_controler()(?Tld_Start);
  emit START_Tld;
  await
   case BF_Tld
   case T2_Sing do emit L_T2_Sing
   case T3_Tld do exit T3_controle_ProcTld
  end await
 ];
await Reinit;
]
end loop
]
when UserStop
```

Conditions: StartSim Reinit UserStop

Items: 1000 Step: 1
Buffer: first data
Output: File

ProcTeledimos:SimTime:DOUBLE: [0 0]
ProcTeledimos:Kind:INTEGER: [0 0]
ProcTeledimos:X_f:DOUBLE: [0 0]

Figure 3: Simulation behaviour specification and associated control panel

With then given parameters the simulation speed easily fits the real-time constraint, using about 10% of the CPU computing power. However, the simulation speed is very sensitive to:

- the complexity of the robot dynamic model,e.g. computing the centrifugal and Coriolis forces is very costly and should be avoided when their effect is negligible,
- the fastest time constants of the system which control the value of the integration step. In particular, increasing the value of the control gains is paid by increased computing power consumption and can lead to a runtime overrun.

In any case it cannot be guaranteed that the simulation will fit the real-time constraint. Missing a temporal dead-line exits the system.

Changing the values of the negative α has few effect on the generated trajectory, probably because we have only one redundant degree of freedom. An important improvement for trajectory generation will be adding intermediate way-points. Speed limits should also be added and the parameters of the real excavator must be identified to make the simulator predictive enough.

3 Stereo Imaging Head

A stereo vision imaging head will be used to provide the necessary information to reconstruct the 3D elevation map of the excavator working environment. The building of the map is based on reliable and semi-automatic procedures. It requires tools and methods especially to gather images, to calibrate the individual cameras, to calibrate the stereo pair and to build particular 3D points.

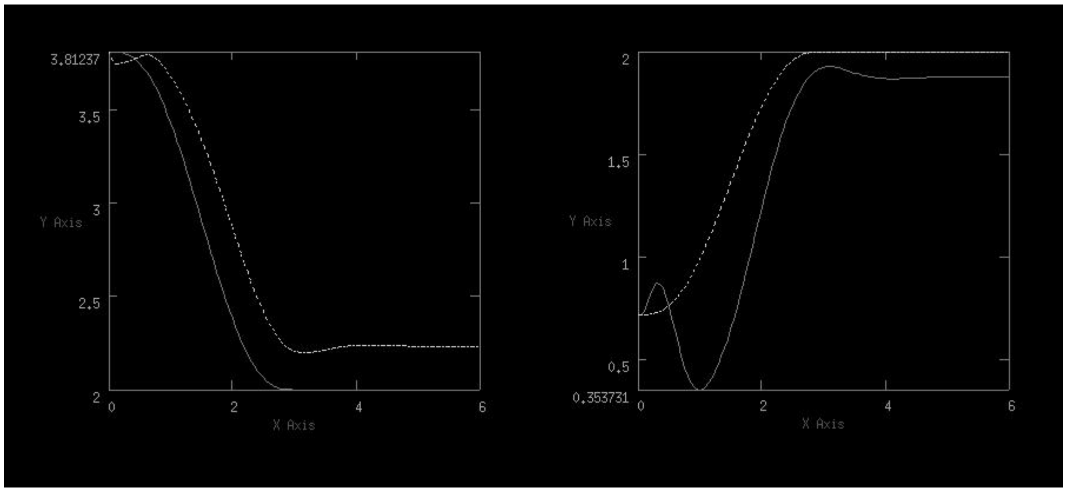


Figure 4: desired and actual X and Y, Tsim=6.0, K=3

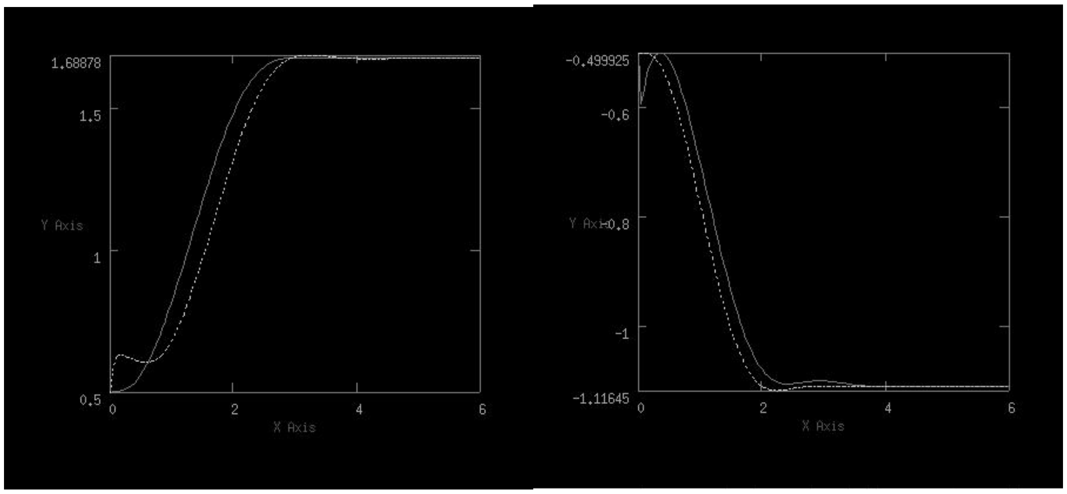


Figure 5: desired and actual positions of the 1st and 3rd joints

The next three subsections decribe tools and methods which have been implemented through a single Java clicking interface.

3.1 Acquisition of images

The calibration of the stereo pair and the building of the 3D map are computed from pictures gathered by both cameras. In order to fix the desired scene and to bring the cameras into focus, the images gathered by the camera need to be viewed by the user. The following basic facilities have been primarily implemented: gathering periodically images(around ten per seconds), freezing a single image and storing it on disk, opening an already stored image in pgm format.

The hardware in use consists in a pair of Sony cameras with 8mm/f1.4 lenses. The processor is a 600 MHz PC running Linux, with at least 256 Mo of RAM. A color frame grabber with a DMA access to the processor is required. The expected 3D reconstruction accurracy is about 1 cm at a 3 meters distance (which is the lenght of the robot arm).

3.2 Calibration of a camera

The method we use to calibrate the stereo pair requires the calibration of each camera. The calibration of a single camera consists in the estimation of its perspective matrix P used to map the 3D coordinates of a point in space onto the coordinates of its image (Figure 6). If M(X,Y,Z) is a 3D space point and m(u,v) its image, we have the formula:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \text{with } s \in \mathbb{R}\,,\ s \neq 0 \tag{1}$$

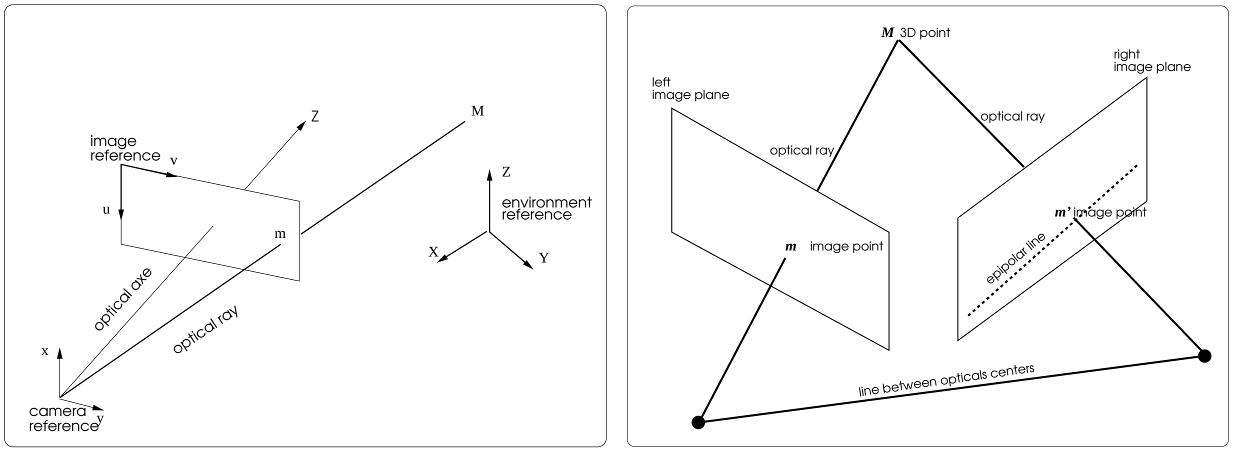
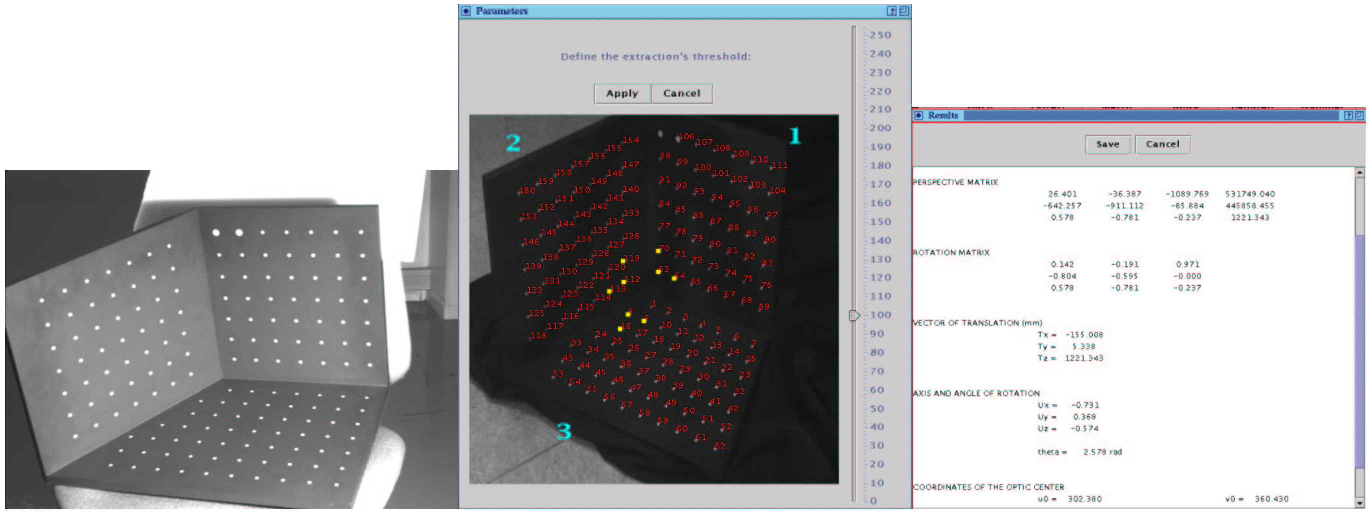


Figure 6: Perspective projection and epipolar geometry

The perspective matrix P expresses both the internal parameters of the camera and the camera position and orientation relative to its environment.

The off-line calibration of an individual camera is a semi-automatic process and uses a special-purpose, known object, to improve the calibration accuracy (Figure 7).



```
PERSPECTIVE MATRIX
                 26.401      -36.387    -1089.769    531749.040
               -642.257     -911.112      -85.884    445858.455
                  0.578       -0.781       -0.237      1221.343

ROTATION MATRIX
                  0.142       -0.191        0.971
                 -0.804       -0.595       -0.000
                  0.578       -0.781       -0.237

VECTOR OF TRANSLATION (mm)
               Tx =  -155.008
               Ty =     5.338
               Tz =  1221.343

AXIS AND ANGLE OF ROTATION
               Ux =    -0.731
               Uy =     0.368
               Uz =    -0.574

               theta =    2.578 rad

COORDINATES OF THE OPTIC CENTER
               u0 =   302.380                          v0 =   360.430
```

Figure 7: Semi-automatic calibration process

The following processes are performed through a graphical interface:

- The image of a circular white mark being modeled, the center of each mark is detected in three steps. An approximative position of the center of each mark is first computed. This position is obtained after transformation of the initial image into a binary signal and taking as center the center of gravity of the values [13].
  The second step more accurately estimates the centers by approximating the parameters of the model for each mark. Finally, some wrong detection are discarded by assuming the marks lie on planes [6].
- A recursive algorithm allows the user to define a correspondence between the centers of numbered marks on a reference image and the extracted centers on his own picture. The algorithm takes as input a maximum of nine "clicks".
- From the previous correspondence, the method of Faugeras-Toscani [8] is performed in order to compute the entries of the perspective matrix.

3.3 Calibration of the stereo pair

Calibrating the stereo system consists in computing the perspective matrices of each camera and to compute the epipolar geometry, i.e. to compute the fundamental matrix which characterize the intrinsic parameters of the cameras pair. To understand the use of the epipolar geometry, let us consider a space point M and its two images m and m'. From m and the epipolar geometry, we can compute the epipolar line associated to m. We know that m' necessarily belongs to the epipolar line.

The perspective matrices are used as data input in the triangulation performed during the 3D reconstruction. The knowledge of the epipolar geometry is used in the automation of the 3D building.

3.4 3D reconstruction

The 3D reconstruction aims to semi-automatically compute the coordinates of some 3D points. More precisely, from the two images we can reconstruct interest points (i.e. points for which the signal changes two-dimensionally) using the perspective matrices of the cameras and the epipolar geometry of the pair.

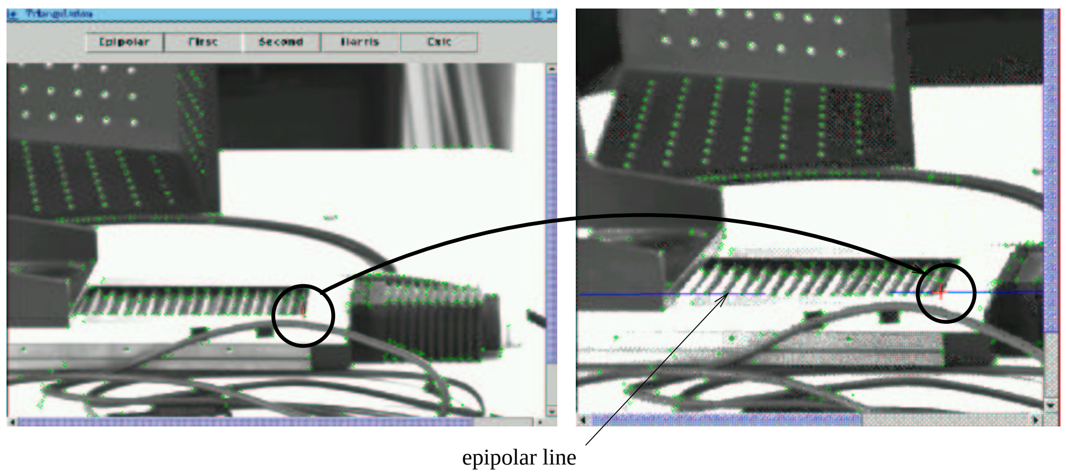


Figure 8: Semi-automatic matching

- Given a pair of image provided by the stereo system, the user selects a point of interest in the left image. An algorithm based on the ZNCC (zero-mean normalized cross-correlation) computes candidates in the right image according to the epipolar geometry. The $ZNCC_k(ij, lm)$ measure for a $(2k+1) \times (2k+1)$ windows centered at pixels (i,j) and (l,m) is:

$$ZNCC_k(ij, lm) = \frac{\sum_{-k \leq dx, dy \leq k} (l_{i+dx,j+dy} - \bar{l}_{ij})(l_{l+dx,m+dy} - \bar{l}_{lm})}{\sqrt{\sum_{-k \leq dx, dy \leq k} (l_{i+dx,j+dy} - \bar{l}_{ij})^2} \sqrt{\sum_{-k \leq dx, dy \leq k} (l_{l+dx,m+dy} - \bar{l}_{lm})^2}}$$

where l_{ij} is the luminancy of pixel (i,j) and $\bar{l}_{ij} = \frac{1}{(2k+1)^2} \sum_{-k \le dx,dy \le k} (l_{i+dx,j+dy})$.
The user chooses in the right image the candidate on the highlighted epipolar line corresponding to the point selected in the left image (Figure 8).

- To help the user in the selection of an interest point, an improved version of the Harris detector has been implemented [12]. Then, the user selects only points which have a high rate of repeatability under the following different transformation: image rotation, scale change, illumination and viewpoint change. It must be point out that in structured environments made of polyhedric objects many interest points can be automatically found and matched thus decreasing the burden of the operator and increasing the quality of matching despite e.g. variable illumination conditions (all green spots in Figure 9 has been automatically recognized as interest points).

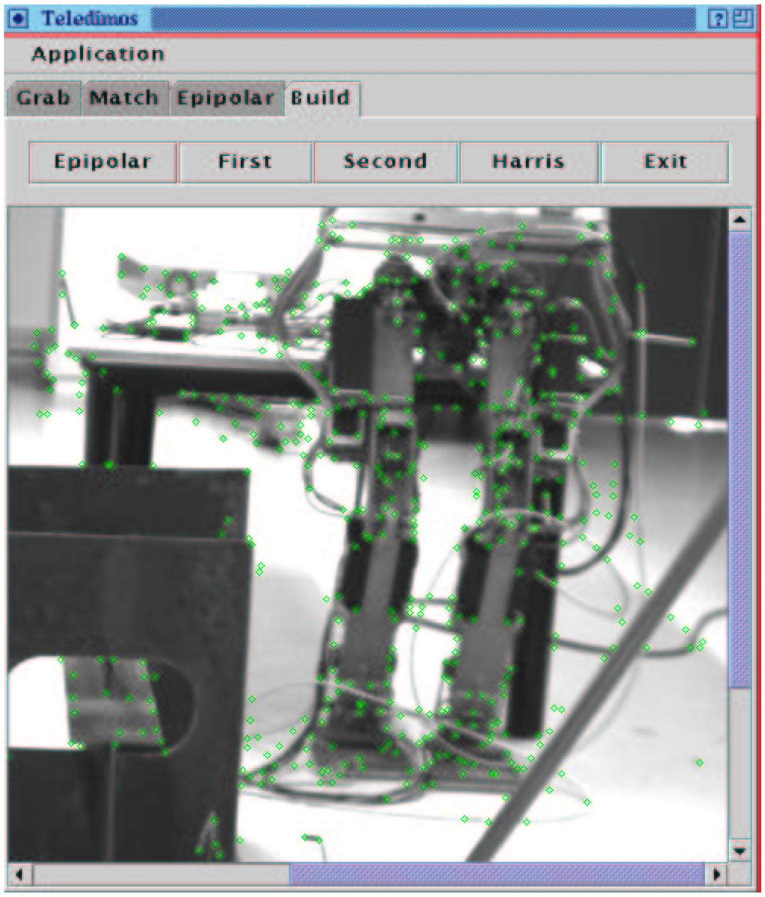


Figure 9: Automatic recognition of interest points

- For the triangulation, i.e. the estimation of the 3D point associated to a pair of matched points, the polynomial method given in [7] is performed. The perspective matrices being exactly known and assuming a Gaussian error distribution for the pairs of matched points, the method gives optimal results.

4 Work in progress : connecting 3D vision and simulation

In a structured environment, with simple polyhedric objects like straight lines and plans, it will be possible to build an elevation map using a coarse matching (a few tens of 3D points) and then applying the Delaunay triangulation.
Once the digital elevation map is provided it is easy to check that the motions of the simulated arm are collision-free with the reconstructed 3D environment. However, connecting these two subsystems requires

that all coordinates, i.e. those of the arm joints and those of 3D points, be expressed in a common frame linked to the vehicule body. Once again the location of the camera pair relatively to the vehicule must be calibrated. According to the location of the camera on the vehicule it is expected that parts of the arm, e.g. the gripper, will be in the view field of the stereo vision system. Thus, this task can be processed through ([2]):

- selection of at least 6 points on the arm link to define a frame;
- identification of this frame in the stereo system to compute the camera-to-arm transformation;
- using the arm kinematics to compute the camera-to-vehicle invariant transformation.

References

[1] Addison, C. A., Enright, W. H., Gaffney, P. W., Gladwell, I. and Hanson, P. M.: "Algorithm 687: a decision tree for the numerical solution of initial value ordinary differential equations", *ACM Transactions on Mathematical Software* vol. 17 no 1, pp 1–10, march 1991.

[2] Andreff N., Horaud R. and Espiau, "On-line Hand-Eye Calibration", *Second International Conference on 3-D Digital Imaging and Modeling (3DIM'99)*, Ottawa, Canada, october 1999.

[3] Ascher U.M. and Petzold L.R., *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, ISBN 0-89871-412-5, 1998.

[4] G. Berry "The Esterel v5 Language Primer ", http://www.esterel.org/Html/Downloads/Doc/StartDocument.htm

[5] J-J. Borrelly, E. Coste Manière, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon and N. Turro: "The ORCCAD Architecture", *Int. Journal of Robotics Research*, special issue on Integrated Architectures for Robot Control and Programming, vol 18, no 4, pp 338–359, avril 1998. http://www.inrialpes.fr/iramr/pub/Orccad/

[6] P. Brand, *Reconstruction tridimensionnelle d'une scène à partir d'une caméra en mouvement: de l'influence de la précision*, PhD thesis, Université Claude Bernard, Lyon I , 1995.

[7] R. Hartley and P. Sturm, "Triangulation", *Computer Vision and Image Understanding*, Vol 68, n.2, p146-157, November 1997.

[8] R. Horaud and O. Monga: *Vision par ordinateur, outils fondamentaux*, Traité des nouvelles technologies, Hermès, Paris, 1995.

[9] Génot F., *Génération automatique de la dynamique de Lagrange des robots rigides arborescents*, Inria Technical Report, february 1998.

[10] W. Khalil and E. Dombre: "Modélisation, identification et commande des robots", *Hermès Science Publications*, collection robotique, Paris, 1999.

[11] Press et al. : Numerical Recipes in C, *Cambridge University Press*, 1992.

[12] C. Schmid, R. Mohr and C. Bauckage, Evaluation of Interest Point Detectors, *International Journal of Computer Vision*, 37(2), pp151-172, 2000.

[13] M.R. Shortis, T.A. Clarke, and T. Short, " A comparison of some techniques for the subpixel location of discrete target image ", *IEEE Transactions on Systems, Man and Cybernetics*, 24(5): pp 820-828, may 1994