JOURNAL OF THEORETICAL AND APPLIED MECHANICS
48, 3, pp. 733-750, Warsaw 2010

INTERACTIVE MULTIBODY SIMULATION IN AUGMENTED REALITY[1]

Pier Paolo Valentini
Eugenio Pezzuti

University of Rome "Tor Vergata", Department of Mechanical Engineering, Rome, Italy
e-mail: valentini@ing.uniroma2.it

In this paper, the authors discuss a methodology to enhance multibody systems simulations using Augmented Reality (AR) implementation. The AR deals with the use of live video imagery which is digitally processed and augmented by the addition of computer generated graphics. The purpose is to illustrate how recent developments in computer-aided design and augmented reality can improve the realism and interactivity when simulating the movement of digital mock-ups. The paper discusses hardware and software implementations and an overview of several illustrative examples. The basic idea is described starting from a simple simulation of a falling body subjected to gravity with the initial conditions set interactively by the user. Then, a more complex interactive simulation of the kinematics of a robot whose end-effector can be grabbed and moved by the user is presented. Finally, the real time dynamic simulation of a slider crank mechanism is discussed. The integration between AR and multibody simulation has revealed to be very useful for didactical purposes and collaborative design.

Key words: interactive simulation, augmented reality, multibody

1. Introduction

During the last two decades, worldwide research on multibody kinematics and dynamics has produced a lot of very interesting results. Exploring new formulations and implementing a better way to build and solve equations of motion have been some of the investigated topics. Moreover, a great improvement has involved the way of simulating and controlling very complex systems, including multidisciplinary aspects. At present, it is possible to simulate very complex

[1]The paper was presented at the ECCOMAS Thematic Conference on *Multibody Dynamics* which was held at Warsaw University of Technology on June 29 – July 2, 2009.

and large systems with a high level of reliability, requiring affordable hardware resources. Due to these important improvements about simulation techniques, the designer has to define a lot of input parameters and interpret a lot of output results. Complex models include many bodies, a lot of constraints, maybe 3D contacts and are able to reproduce very complex three-dimensional motion. A considerable support to the management of pre- and post-processing of data has been offered by the development of Computer Aided Engineering (CAE) environments (Bernard, 2005). The development of hardware and software capabilities has supported the integration between modeling environments and multibody solvers. Moreover, many modeling environments can interpret the mating relations in assembling, translating them into constraint equations for multibody simulation. After (or during, in some cases) the simulation, it is possible to use the computer graphics capabilities in order to visualize realistic animation in order to have an idea about the behavior of the systems at a glance. It is useful for debugging the implementation of the system and correcting some evident errors without screening a lot of numerical data. The simplification in building and reviewing models has boosted their usage in industry where saving time and money is very important.

The support of computer graphics to multibody simulation has been useful also for didactics. Teachers can prepare realistic 3D multibody models to explain the functioning of complex mechanisms, being more effective on the students. Presently, CAE applications support designers and teachers through the communication of results coming from numerical computations. By using these instruments, engineers and designers can develop their creative ideas in front of a computer monitor using the mouse and keyboard. Although the integration between numerical computation and graphics leads to the generation of photo-realistic digital mock-ups, they are still far from the real context and the user has a limited interaction with them. This limitation can generate problems (non-conformities, unexpected behavior and appearance, etc.) when the designed products have to be integrated in the real world. For overcoming this disadvantage, the development of new instruments, based on the mixing between the real world and virtual objects, seems to be the future of CAE. One of these instruments is the Augmented Reality.

2. Augmented reality

The Augmented Reality (AR) is an emerging field of the visual communication and computer technologies (Azuma, 1997; Azuma *et al.*, 2001; Bimber and Raskar, 2005). It deals with the combination of real world images and computer

generated data. At present, most AR research is concerned with the use of live video imagery which is digitally processed and "augmented" by the addition of computer generated graphics. With an AR system, the user can extend the visual perception of the world, being supported by additional information and virtual objects. These objects do not remain inside a computer monitor but they are integrated in the real world. The AR system has some similarities with the Virtual Reality (VR) one. The main difference is that in the VR the perceived world is fully virtual (generated by one or more rendering pipelines), while in the AR the virtual world *merges* the real one. The level of details of the augmented scene has to be very realistic in order to give the user the illusion of a unique real world. The ways to capture images from the real world, process and project again to the user can be different (Vallino, 1998). The most implemented and promising technology for engineering purposes is the video see-through system. As illustrated in Fig. 1, it is based on the use of one or two cameras which acquire an image stream from the real world. The stream is processed by a computer which adds the virtual content, producing an augmented image stream which is projected again to the user by means of a blind visor.

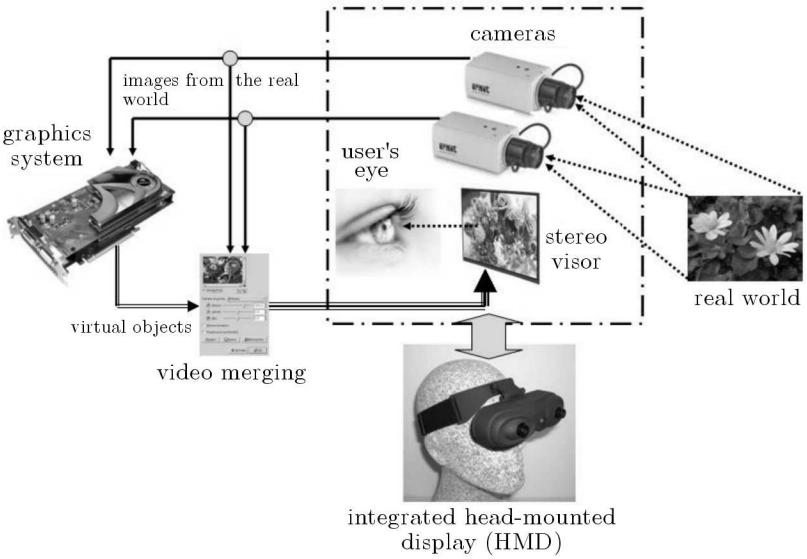

Fig. 1. AR video see-through system

Scientific literature reports an increasing interest for the development of applications of augmented reality in many different fields. The AR has been used in medicine, surgery, military field, implant and components maintenance, robotics, and architecture. Most of these applications deal with the merging

in the real world of objects, scene and animation which have been modeled and simulated outside the system. It means that the user perceived a real scene augmented with pre-computed objects. His interaction with them is often limited to exploration. A good interaction with the scene may improve the impact of such technology (Klinker *et al.*, 1999). Recent contributions in the scientific literature report and increasing interest in the development of engineering tools in augmented reality environment (Stilman *et al.*, 2005; Valentini, 2009; Valentini *et al.*, 2008, 2009).

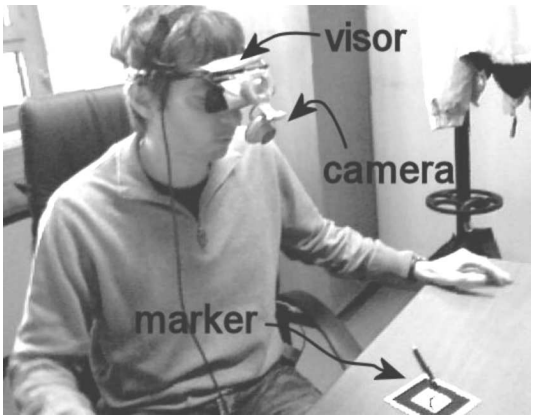

Fig. 2. The implemented AR system

For the specific purpose of this investigation, the implemented AR system (see Fig. 2) is comprised of an input video device Microsoft LifeCam VX6000 USB 2.0 camera, able to catch frames up to 30 Hz with a resolution of 1024×768 pixels. The camera has been rigidly mounted on an Head Mounted Display equipped with OLed displays (Z800 3D visor by Emagin – http://www.3dvisor.com/). It is able to support a resolution of 800×600 pixels for each eye. The processing unit is a personal computer with an Intel Core 2 Quad-core processor, 3 GB RAM and a NVidia Quadro FX3700 graphic card. The operating system was Windows XP and the development suite for programming was Microsoft Visual Studio 2003.

Routines for image processing have been developed using the well known open source library named ARToolkit which can be freely downloaded from http://sourceforge.net/project/showfiles.php?group_id=116280. It comprises a set of numerical procedures which are able to detect and recognize a planar patterned marker in a video stream in real time. Using correlation techniques, the routines are also able to compute the relative position and attitude between markers and camera with good precision. This computation is necessary for an

accurate perspective collimation between the virtual entities and the real scene. The details about specific implementation and about the contents of the library go beyond the scope of this paper, and the interested reader can find useful material at the internet site http://www.hitl.washington.edu/artoolkit/.

3. Implementing multibody simulation into augmented reality environment

Starting from the description of the capabilities of the Augmented Reality, the next question is "how can the AR support multibody simulations?". There are two possible answers. The first is about the possibility to project on the real world the results coming from a pre-computed simulation. It concerns the rendering on the scene of all the objects involved in the simulation whose position is updated according to the results of the simulation. This implementation is similar to that of the common post-processing software for visualizing graphics results. The only difference is in the introduction of the simulated system in the real world. The advantage is to perceive the interaction with the real world and check working spaces, possible interferences, etc. Although it can be useful, this approach does not use all the potential of AR. A smarter way to enhance the multibody simulation is to introduce *interactivity*. It means that the user does not only watch the augmented scene, but interacts with it. Interaction can concern both the definition of boundary conditions and initial parameters and the real time control of the simulation. In all these cases, the solution of the equations of motion has to be computed in real time in order to populate the scene with quickly updated information.

In order to enable communication between the user and the scene, the scene has to contain some specific sensors. Since the implementations of AR are based on the image acquisition and processing, the most simple sensor is a flat patterned marker. Starting from these considerations, a generic multibody simulation in augmented reality can be implemented following four main steps:

1. Before the simulation starts, the geometries and topological properties (joints and connections) have to be defined (as for any multibody system);

2. The real scene has to contain information for collimating the real world to the virtual objects and the virtual sensor(s) for the interactive action of the user (i.e. has to contain an adequate number of patterned markers);

3. During each frame acquisition, the multibody equations have to be solved in order to compute the correct position of all the virtual bodies in the scene, taking into account the information coming from the sensors;

4. For each frame acquisition, virtual objects have to be rendered on the scene in the correct position and attitude.

The rendering of geometries can be implemented using OpenGL capabilities. In order to use quite complex shapes, all the geometries can be modeled in .vrml (or modeled in a CAD and then exported in .vrml). In this case, another coded library can be useful, the OpenVrml, that is able to manage the OpenGL rendering of .vrml contents. The details about OpenVrml goes beyond the scope of this paper and the interested reader can find useful material at the internet site of such projects (http://openvrml.org/).

A scheme of the workflow for the entire simulation is depicted in Fig. 3.

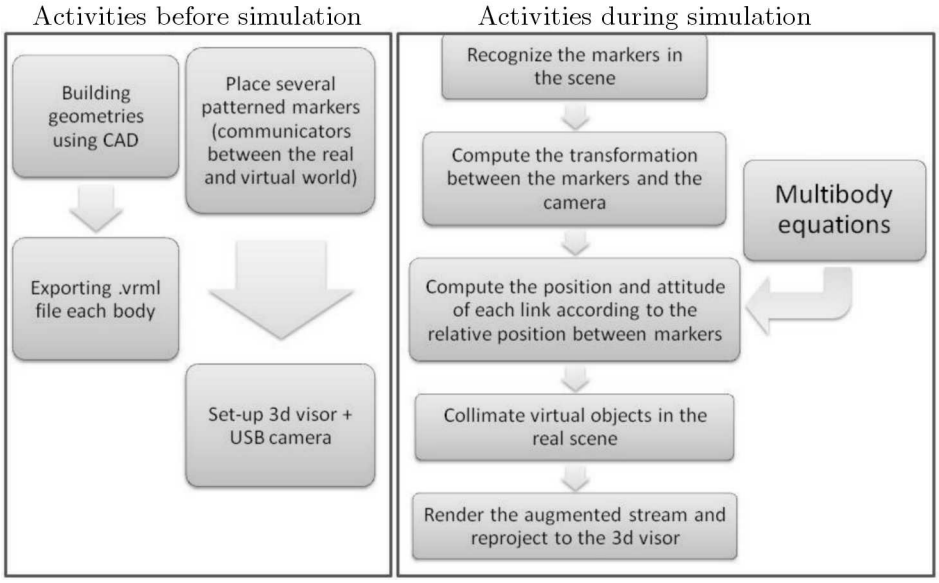

Fig. 3. Tasks for implementing multibody simulation in augmented reality

Between the input procedures (data acquisition and markers recognition) and the output procedures (rendering of the virtual objects on the input scene), the multibody computation has to be interposed. Since it has to be performed between the acquisition of one frame and the next one, all the equations need a real-time solution. For this purpose, it is useful to optimize the solution strategy, and kinematic simulations are more suitable because involve the solution

to a system of non-linear equations, instead of a system of differential-algebraic equations (Garçıa de Jálon and Bayo, 1994).

In order to explain in details all the steps required to implement a multibody simulation in an Augmented Reality environment, three examples are described next.

4. First example: a bouncing ball

The first example deals with the simulation of a rigid sphere that falls subjected to gravity and bounces on a plane. The initial condition of motion of the body has to be chosen by the user interactively. The simulation needs two patterned markers. The first defines the position and attitude of the world reference frame. It is useful for computation of the sphere reference frame and for the location of the rigid plane which the ball bounces on. The second marker can be reviewed as a communication sensor. The user's intent can be interpreted by tracking the position and velocity of the point P on this marker. The position of point P can be computed starting by recognition of the marker in the scene, computing its homogeneous transformation matrix with respect to the camera $\mathbf{T}_{2-\text{camera}}$.

The generic transformation matrix $\mathbf{T}$ is a 4×4 matrix which expresses the relative position and attitude between two reference systems. The first 3×3 portion of this matrix is used to define the relative orientation between the two reference frames. The last column is used to describe the relative position between the origins of the coordinate frames. The last row of the matrix is $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$

$$\mathbf{T} = \begin{bmatrix} [\text{Orientation}]_{3\times 3} & [\text{Position}]_{3\times 1} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{4.1}$$

The relative position between marker 2 and marker 1 can be computed by multiplication of the transformation matrices, and the position of point P with respect to marker 1 (world) reference frame can be computed as

$$\boldsymbol{P}_1 = \mathbf{T}_{1-cam}^{-1}\mathbf{T}_{2-cam}\boldsymbol{P}_2 \tag{4.2}$$

where

$\boldsymbol{P}_i = \{x, y, z\}_i^{\top}$ – position vector of point P, expressed using marker i;

$\mathbf{T}_{i-cam}$ – transformation matrix between marker i and camera, expressed as in Eq. (4.1).

Concerning the velocity of the point P, it can be estimated from computation of the position in two consecutive frames as

$$\dot{\boldsymbol{P}}_1 = \mathbf{T}_{1-cam}^{-1}\mathbf{T}_{2-cam}\dot{\boldsymbol{P}}_2 = \mathbf{T}_{1-cam}^{-1}\mathbf{T}_{2-cam}\left(\frac{\boldsymbol{P}_2^{fr\,i} - \boldsymbol{P}_2^{fr\,(i-1)}}{\frac{1}{framerate}}\right) \tag{4.3}$$

where

$\dot{\boldsymbol{P}}_j = \{v_x, v_y, v_z\}_j^\top$ – velocity vector of point P, expressed using marker j reference frame;

$\boldsymbol{P}_j^{fr\,i}$ – position vector of point P, expressed using marker j computed at frame i;

framerate – frame rate of video stream [frame/s].

After computation of initial conditions, the simulation can start. The user can chose this event using the keyboard or mouse as trigger. After the beginning of the simulation, the position of the sphere in the space cannot be chosen by the user, but it is computed by integrating the equations of dynamics. Since the computation and the rendering have to be real time processed, an explicit integration procedure has to be preferred. Assuming the gravity acting along the world – y direction, for the free falling of the sphere the equations are

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix}^{fr\,i} = \begin{Bmatrix} v_x \\ v_y - g\Delta t \\ v_z \end{Bmatrix}^{fr\,(i-1)} \qquad \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix}^{fr\,i} = \begin{Bmatrix} p_x + v_x\Delta t \\ p_y + v_y\Delta t - g\Delta t^2 \\ p_y + v_y\Delta t \end{Bmatrix}^{fr\,(i-1)} \tag{4.4}$$

where

$\boldsymbol{p}^{fr\,i}$ – position vector of the center of the sphere at frame i;

$\boldsymbol{v}^{fr\,i}$ – velocity vector of the center of the sphere at frame i.

The detection of the contact with a generic plane can be computed using a proximity condition on the distance between the ball and the plane as

$$\text{Contact} \rightarrow |d|_{sph\text{-}pl} - R_{sph} \leqslant \text{Tolerance} \tag{4.5}$$

where

R_{sph} – radius the sphere;

$|d|_{sph\text{-}pl}$ – modulus of the distance between the sphere and the plane.

When Eq. (4.5) is fulfilled, the integration of the equation of motion has to be stopped, imposing the impact condition for computing the new initial conditions

$$\boldsymbol{v}^{fr\,i}\boldsymbol{n}_{pl} = -e\boldsymbol{v}^{fr\,(i-1)}\boldsymbol{n}_{pl} \tag{4.6}$$

where

$\boldsymbol{n}_{pl}$ – unit vector normal to plane;

e – restitution coefficient of impact (scalar).

Figure 4 summarizes four snapshots taken during the simulation. The trajectory of the center of the ball has been tracked using small spheres in order to visualize the sequence of bounces and the effect of restitution coefficient.

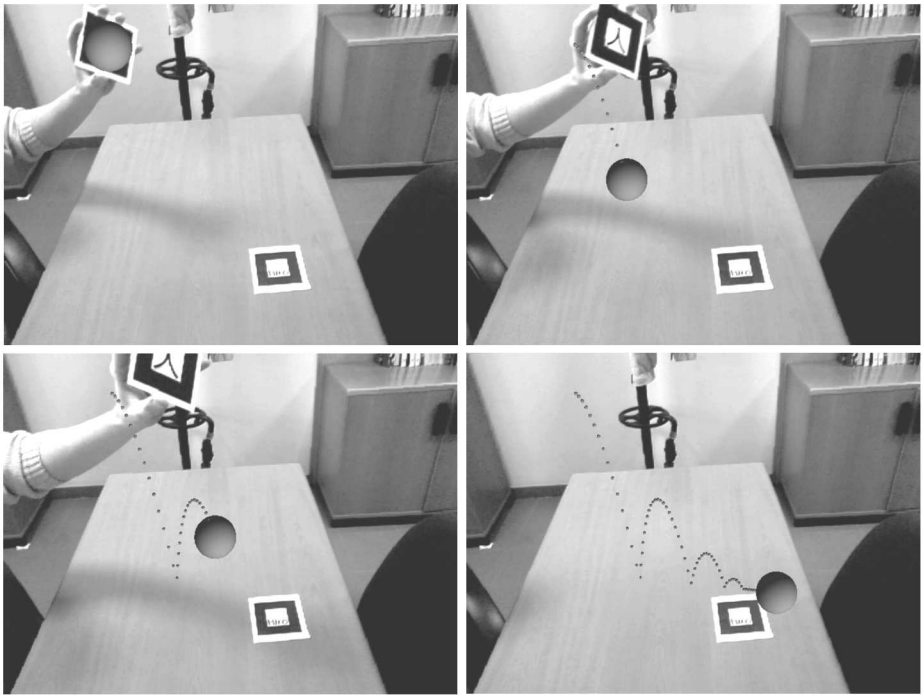

Fig. 4. Four snapshots from bouncing ball simulation

5. Second example: inverse kinematics of a spatial manipulator

The second example deals with the simulation of a spatial manipulator. With reference to Fig. 5, let us consider a mechanism with 4 rigid links connected

with 3 revolute joints and 1 spherical joint. According to Grüebler's count, the manipulator has 6 degrees of freedom

$$dof = 6n_{link} - \sum_{i=1}^{joints} (6 - f_i) = 6$$

It means that in order to define in a unique way the spatial position of the manipulator we have to prescribe 6 independent parameters (i.e. position and attitude of the end-effector). For an interactive inverse kinematics simulation, it means that the user can freely choose the position and attitude of the end effector and the Augmented Reality scene has to be able to include such a 6 d.o.f. sensor.

The first step in building the model is the construction of geometries of each link that can be stored in a .vrml file. The second step is about the preparation of the scene. We need a marker (marker 0) to define the position and orientation of the manipulator world coordinate system (i.e. its position inside the scene) and another marker (marker 1) to define the position and the orientation of the end-effector that will work as an active sensor (Fig. 5, on the right).

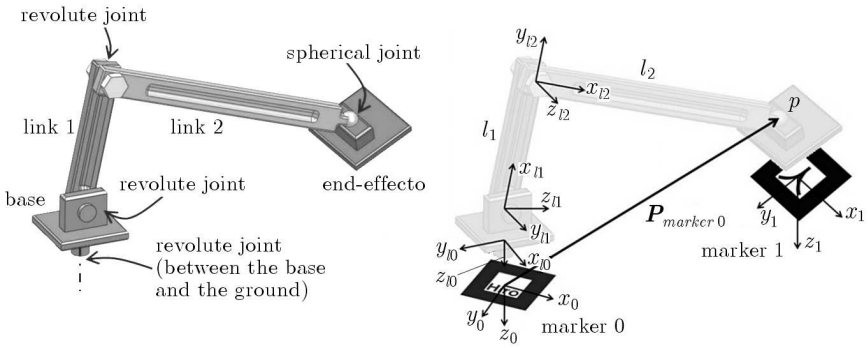

Fig. 5. The manipulator of the second example (nomenclature and topology on the left, reference frames and patterned markers on the right)

The third step is about the implementation of the system of constraint equations. It is useful, for the subsequent graphical operations, to use a 4×4 homogeneous transformation matrix $\mathbf{T}$ to express the relative position and attitude between two generic reference frames. The first 3×3 portion of this matrix is used to define the relative orientation between the two reference

frames. The last column is used to describe the relative position between the origins of the coordinate frames. The last row of the matrix is $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$

$$\mathbf{T} = \begin{bmatrix} [\text{Orientation}]_{3\times3} & [\text{Position}]_{3\times1} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix}$$

Looking at the tip point P on link 2 (center of the spherical joint), we can deduce its position with respect to marker 0 as

$$\boldsymbol{P}_{marker\ 0} = \mathbf{T}_{0-l0}\mathbf{T}_{l0-l1}\mathbf{T}_{l1-l2}\boldsymbol{P}_{link\ 2} \tag{5.1}$$

where

$\boldsymbol{P}_{marker\ 0}$ – position vector of point P in marker 0 (world) coordinate system;

$\boldsymbol{P}_{link\ 2}$ – position vector of point P in link 2 (local) coordinate system;

$\mathbf{T}_{0-l0}$ – homogeneous transformation matrix between the base (considered as link 0) and marker 0. It is a function of the parameter α which describes the relative rotation at their relative revolute joint;

$\mathbf{T}_{l0-l1}$ – homogeneous transformation matrix between link 1 and the base (considered as link 0). It is a function of the parameter β which describes the relative rotation at their relative revolute joint;

$\mathbf{T}_{l1-l2}$ – homogeneous transformation matrix between link 2 and link 1. It is a function of the parameter γ which describes the relative rotation at their relative revolute joint.

Looking at the same point P but on the slider, since the slider is attached to marker 1, we can deduce its position with respect to marker 0 as

$$\boldsymbol{P}_{marker\ 0} = \mathbf{T}_{0-1}\boldsymbol{P}_{slider} \tag{5.2}$$

where

$\boldsymbol{P}_{slider}$ – position vector of point P in slider or marker 1 (local) coordinate system;

$\mathbf{T}_{0-1}$ – homogeneous transformation matrix between marker 1 and marker 0. It is a function of 6 independent parameters which describe the relative position and the relative rotation between the two markers. These parameters can be considered as the input of the inverse kinematic analysis because can be freely chosen by the user to move the manipulator.

Since at the point P link 1 is connected to the slider by means of a spherical joint, we can obtain the closure loop equation of the mechanism as

$$\mathbf{T}_{0-l0}\mathbf{T}_{l0-l1}\mathbf{T}_{l1-l2}\boldsymbol{P}_{link\ 2} - \mathbf{T}_{0-1}\boldsymbol{P}_{slider} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix} \tag{5.3}$$

The system of equations (5.2) can be solved for the unknown kinematic parameters α, β, γ starting from the knowledge of the position and attitude of marker 1 (and the end-effector slider) with respect to marker 0. In order to compute this information, we have to know the relative transformation between marker 1 and marker 0.

ARToolkit deals with matrices which are similar to the homogeneous ones. They are 3×4 transformation matrices, containing information about the relative position and attitude as the homogeneous ones but without the last dummy row. Quaternions and the position vector can be extracted from these matrices by using `arUtilMat2QuatPos` Artoolkit procedure.

The relative position of marker 0 ($\mathbf{T}_0^c$) and marker 1 ($\mathbf{T}_1^c$) with respect to the camera can be computed using the Artoolkit procedure `arGetTransMat`. Their inverse matrices ($\mathbf{T}_c^0$ and $\mathbf{T}_c^1$, respectively) represent the relative transformations between the camera and the markers. The relative transformation between marker 1 and marker 0 can be computed as

$$\mathbf{T}_2^1 = \mathbf{T}_2^c\mathbf{T}_c^1 \tag{5.4}$$

This relation is used to compute the vector $\boldsymbol{d}$ and the vector $\boldsymbol{a}_2$ in the world reference frame. By doing this, Eq. (5.2) can be solved for the unknown angles. At this point the virtual position and attitude of each link of the mechanism are known. The next step is to compute the projection of geometries in the augmented scene. Since the renderer engine is OpenGL, we have to deal with the ModelView projection matrix which maps the 3D points coordinates into 2D (scene) coordinates. The first step in the rendering concerns the load of the projection matrix computed from the perspective of marker 0. This task can be performed by using `arglCameraViewRH` whose output is a vector of 16 elements containing the transformation and scale information to project the object from the 3D coordinate system of marker 0 into the camera view plane. This transformation can be used to relate the position and attitude of each object according to the virtual world coordinate system (marker 0). The next steps concern the computation of the transformations in order to draw the manipulator links in the scene, using the OpenGL operators `glRotated` and

`glTranslated`. The first procedure performs a rotation of an angle about a specified axis, the second performs a translation of a specified amplitude along a specified direction.

For the base, it is sufficient to perform a rotation about the z axis of marker 0 by an angle α:

```
glRotated(alpha,0.0,0.0,1.0)
```

For the first link, three transformations are needed: translation of $\boldsymbol{a}_1$, rotation about the z axis of marker 0 by angle α and rotation by angle β about the axis of the first revolute joint:

```
glTranslated(0.0,0.0,a1)
glRotated(alpha,0.0,0.0,1.0)
glRotated(beta,0.0,-1.0,0.0)
```

For the second link, three transformations need: rotation by angle α about the $-z$ axis of marker 0, translation to the center of the revolute joint between the 1st and the 2nd links and rotation by angle γ about the axis of the revolute joint between the 1st and the 2nd links:

```
glRotated(alpha,0.0,0.0,1.0)
glTranslated(l1*cos(beta),0.0,l1*sin(beta)+a1)
glRotated(gamma,0.0,-1.0,0.0)
```

The end-effector, since it is fixed to marker 1 can be projected using the projection matrix of marker 1 without applying any further transformation.

Screen shots of animation are reported in Fig. 6.

6. Third example: dynamics of a slider-crank mechanism

The third example deals with simulation of the dynamic behavior of a slider-crank mechanism shown in Fig. 7. It comprises four links: the base, crank, rod and slider. The slider is connected to the base with a spring damper element. The example will show how the augmented reality can be used in order to interactively define the initial conditions of the simulation and review a real-time processed animation.

As discussed in the previous example, the first step in building the model is the construction of the geometry of each link which is subsequently stored in a .vrml file. The second step regards the preparation of the scene. We need two patterned markers: marker 0 to define the position and orientation of the world coordinate system, and marker 1 to define the position and the orientation of the slider that will work as an active sensor (Fig. 7, on the right).

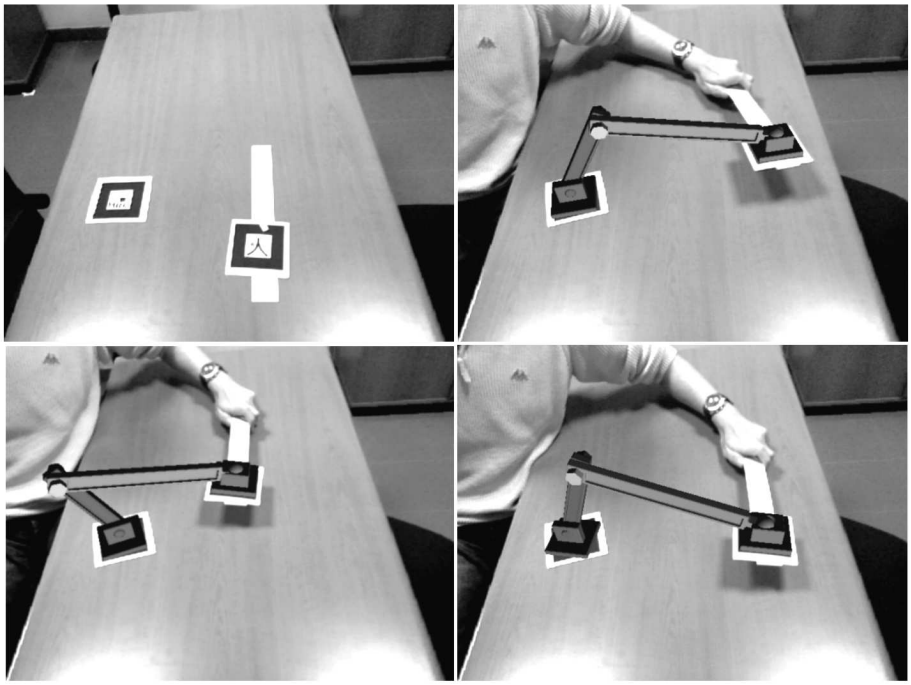

Fig. 6. Snapshots from the inverse kinematics simulation of a spatial manipulator

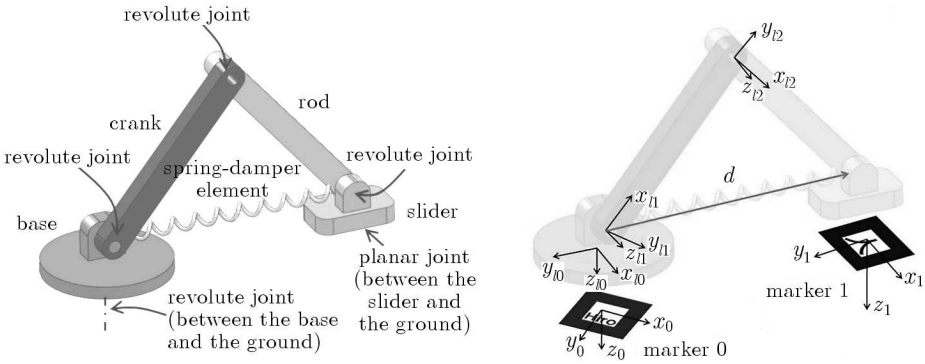

Fig. 7. The mechanism of the third example (nomenclature and topology on the left, reference frames and patterned markers on the right)

In order to interactively solve the equations of motion, the DAE system has to be build before the simulation starts, and has to be embedded in the run-time code.

For the example, it is useful to implement these equations in a matrix form (Haug, 1989)

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Psi}_q^{\top} \\ \boldsymbol{\Psi}_q & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F} \\ \boldsymbol{\gamma} \end{Bmatrix} \tag{6.1}$$

where

$\ddot{\boldsymbol{q}}$ – vector of generalized accelerations ($\boldsymbol{q}$ is the vector of generalized coordinates and $\dot{\boldsymbol{q}}$ – of generalized velocities);

$\boldsymbol{\lambda}$ – vector of Lagrange multipliers;

$\boldsymbol{F}$ – vector of generalized forces (which includes the contribution of the spring-damper element and gravity);

$\boldsymbol{\Psi}_q$ – Jacobian of the constraint equations vector $\boldsymbol{\Psi}$;

$\boldsymbol{\gamma} = -(\boldsymbol{\Psi}_q \dot{\boldsymbol{q}})_q \dot{\boldsymbol{q}} - 2\boldsymbol{\Psi}_{qt}\dot{\boldsymbol{q}} - \boldsymbol{\Psi}_{tt}$, in which the generic subscript i means $\partial/\partial i$;

$\mathbf{M}$ – inertial matrix.

For the proposed example we have 23 constraint equations (5 for the revolute joint and 3 for the planar joint) and the mechanism possesses one degree of freedom.

In order to be real time evaluated, the system in Eq. (6.1) has to be computed using an explicit integrator. The integration time has to be identical to that of the video frame rate.

With reference to Fig. 8, the simulation strategy is the following. As the first step, the user can manipulate marker 1, moving the slider (snapshot *a* of Fig. 8). The equations for the real time computation of the position and attitude of each link can be obtained like in the previous example. Then, the user can interactively define the undeformed length of the spring, placing the slider and pressing a key on the keyboard or a button on the mouse. As a result, the spring is rendered on the scene (snapshot *b* of Fig. 8). At this point, the user can move the slider, and the spring will be still attached to it and is rendered as deformed (snapshot *c* of Fig. 8). By pressing another key or button, the user can start the simulation. When the simulation begins, the position of the slider can not be chosen any more by moving marker 1, but the position and attitude of each link are computed according to Eq. (6.1). A real time animation is rendered on the scene (snapshot *d* of Fig. 8). According to the simulation outputs, the position and attitude of the links can be drawn by using the OpenGL operators `glRotated` and `glTranslated` as shown in the previous example.

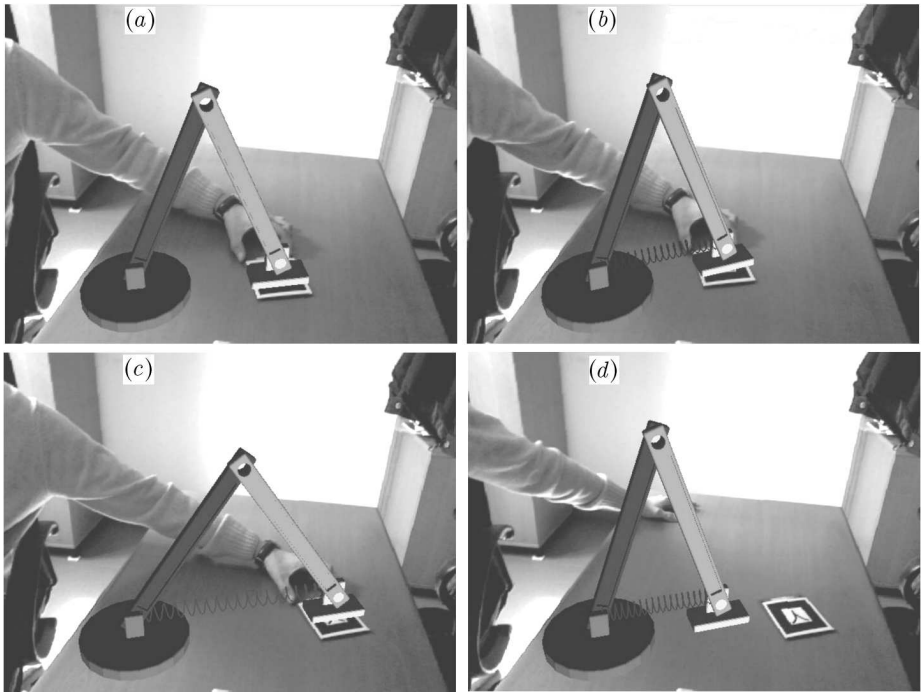

Fig. 8. Snapshots from the dynamic simulation of a slider-crank mechanism

7. Conclusions

The implementation of multibody simulation in the augmented reality reveals many attractive features. First of all, it is possible to perform simulation of a virtual object projecting the results in the real environment. By this way it is possible to check the integration between the virtual objects and the real ones. It is also possible to move the digital mechanism dragging intuitive sensors, instead of the mouse and keyboard, as it happens in the standard CAE environment. These simulations can be also useful for didactics purposes to support the explanation of motion of complex mechanisms, the effects of specific joints, the comprehension of working space, etc. In regards to the implementation details, the current level of hardware performance allows finding the real-time solutions to both kinematics and dynamics equations between the acquisition of two subsequent camera frames. All the tasks concerning the recognition of markers and the computation of view transformation matrices can be performed using Artoolkit libraries. On the other hand, all the tasks concerning the drawing of textured shapes can be performed using the Ope-

nVrml and OpenGL libraries. All these routines are open source and can be easily personalized.

The future work will focus on two aspects. One concerns the implementation of numerical results (such as reaction forces at joints, velocities, energy, etc.) directly on the augmented scene. The second focuses on the integration of specific and multipurpose real time integrators. A collaborative design (two or more people managing the same scene) and the implementation of stereo vision are other interesting aspects to be investigated.

References

1. AZUMA R.T., 1997, A survey of augmented reality, *Teleoperators and Virtual Environments*, **6**, 4, 355-385

2. AZUMA R., BAILLOT Y. ET AL., 2001, Recent advances in augmented reality, *IEEE Computer Graphics*, **21**, 6, 34-47

3. BERNARD A., 2005, Virtual engineering: methods and tools, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **219**, 5, 413-422

4. BIMBER O., RASKAR R., 2005, *Spatial Augmented Reality Merging Real and Virtual Worlds*, A.K. Peters Ltd.

5. GARÇIA DE JÁLON J., BAYO E., 1994, *Kinematic and Dynamic Simulation of Multibody Systems – the Real-Time Challenge*, Springer-Verlag, New York

6. HAUG E.J., 1989, —em Computer-Aided Kinematics and Dynamics of Mechanical Systems, Allyn and Bacon, Boston, MA, USA

7. KLINKER G., STRICKER D., REINERS D., 1999, Optically based direct manipulation for augmented reality, *Computers and Graphics*, **23**, 827-830

8. STILMAN M., MICHEL P., CHESTNUTT J., NISHIWAKI K., KAGAMI S., KUFFNER J.J., 2005, Augmented Reality for Robot Development and Experimentation, Tech. Report CMU-RI-TR-05-55, Robotics Institute, Carnegie Mellon University

9. VALENTINI P.P., 2009, Interactive virtual assembling in augmented reality, *International Journal on Interactive Design and Manufacturing*, **3**, 109-119

10. VALENTINI P.P., GATTAMELATA D., PEZZUTI E., 2008, A CAD system in Augmented Reality application, *Proc. of 20th European Modeling and Simulation Symposium, Track on Virtual Reality and Visualization*

11. Valentini P.P., Pezzuti E., Gattamelata D., 2009, Virtual engineering in augmented reality, Computer Animation, series *Computer Science, Technology and Applications*, Nova Publishing [in press]

12. Vallino J., 1998, Interactive augmented reality, PhD Thesis, Department of Computer Science, University of Rochester, USA

Interaktywna symulacja wielobryłowa w Rzeczywistości Poszerzonej (AR)

Streszczenie

W pracy autorzy przedyskutowali metodologię usprawnienia symulacji wielobryłowej poprzez zastosowanie tzw. Rzeczywistości Poszerzonej (*Augmented Reality* – AR). Metodologia ta obejmuje obróbkę cyfrową obrazu pozyskanego kamerą z rozszerzeniem polegającym na dodaniu do tego obrazu grafiki komputerowej. Celem pracy jest pokazanie, na ile najnowsze osiągnięcia komputerowo wspomaganej obróbki obrazu mogą poprawić wrażenie realizmu zapisywanych cyfrowo obrazów i polepszyć interaktywność z użytkownikiem podczas symulacji ruchu uzyskanych makiet rzeczywistych przedmiotów. Zaprezentowano kilka przykładów z omówieniem zastosowanych programów i parametrów sprzętu komputerowego. Podstawowa idea została przedstawiona na przykładzie ciała spadającego pod wpływem siły ciężkości na inny obiekt przy możliwości interakcji użytkownika, który wybiera warunki początkowe upadku. Następnie opisano bardziej złożony model, tj. kinematykę robota, w którym ruch chwytaka może być kontrolowany poprzez użytkownika. Na koniec pokazano symulację dynamiki suwakowego mechanizmu korbowego w czasie rzeczywistym. Tezą pracy jest wykazanie, że integracja symulacji wielobryłowej z możliwościami Rzeczywistości Poszerzonej (AR) posiada ogromne walory dydaktyczne i stanowi pomoc w zadaniach międzyzespołowego projektowania.

Manuscript received November 25, 2009; accepted for print February 22, 2010