Full Control of Virtual Objects Manipulation Based on the Images of Real Ones

Brahim Nini

Larbi Ben M'hidi University, Constantine road, PB No 358, Oum El-Bouaghi, 04000 Algeria.

***Abstract*—This work deals with the virtual manipulation of a real object through its images. The results presented in this paper give a movie-based solution to the simulation process. We show how the simulation of infinite virtual views of a moving object can be reached using a finite number of object's taken images stored in an organized way. The basis of this solution is an analytical geometry-based method that links explicit applied user's actions, resulting in an object's views change, and images that match the best such views. This paper presents an overall solution for these three intertwined parts of the virtual manipulation that involves six degrees of freedom. Hence, a user is able to freely manipulate a virtual object in a scene in whatever manner s/he likes. In this case, the actions are transformed into rotations and/or translations which lead to some changes in object's appearance, both covered by two viewing features: zoom and/or rotations.**

***Index Terms*—Image-based rendering, Manipulation, Virtual object, 3D geometry.**

I. INTRODUCTION

Modeling the appearance of captured three-dimensional (3D) objects is a fundamental requirement for virtual and augmented reality domains. The resultant model allows the users to interact with the scene and be participants themselves but not remain passive observers. This is why the visual effect is an important and challenging feature in the process to reach real feelings. However, the rendering related to the manipulation of virtual objects is either synthetic based or real objects' captured images based. Typically, there are some particular situations where real objects, whose rendering is image-based, are more suitable for visual quality than synthetic ones. Moreover, there are several situations where modeling real objects to make them computer-generated is not relevant because rendered images in a synthetic manner do not look as realistic as photographs of real ones. Photographs easily capture complex textures and global illumination.

Despite recent advances, the modeling of complex real objects is still difficult, and rendering textured models usually does not produce realistic images. In general, 3D modeling and rendering approach presents three main disadvantages in spite of hardware acceleration technologies advance. First, creating 3D geometrical entities relies on a difficult manual process. Second, rendering systems usually set a limit to object complexity and rendering quality in relation to time while they have really no upper bound. Third, although its price is accessible, professional hardware is still not easily affordable.

Over the past few years, image-based rendering (IBR) techniques have been investigated by many researchers as an alternative to some conventional 3D model-based rendering. Instead of modeling scenes and/or objects, these are photographed from reality. The aim of IBR systems is then to produce virtual dynamism based on static images. Their goal is to reproduce the scene correctly at an arbitrary viewpoint with unknown or limited amount of geometry. They have the advantage of making the rendering speed independent from scenes' complexity since it is inherent into higher photorealism [1]. In fact, IBR systems do not rely on complex computational algorithms for the rendering. For instance, the automatic change of the level of detail in IBR systems can easily be achieved just through multiple levels of resolutions.

IBR solution involves high improvement to many applications such as the ones of augmented reality domain. The aim of these applications is to make a mixture so that a generated sequence corresponds to a real scene into which virtual objects are added. The latter may be of different nature; they may be either image-based or synthetic. The relevance of IBR systems to augmented reality systems is the ability to use real objects. Many works, as this one, take benefit from this opportunity and focus on the augmentation of a scene with real objects' images [4]-[12]; i.e. virtual objects are image-based. In this case, the augmentation process may have different aims. In some work, for instance, virtual objects constitute a tool for the purpose of making the achievement of certain tasks easier [7]. Whereas in [16], the virtual object is described as being a guide for the repair operation in an industrial maintenance.

This work deals mainly with the visual aspect of the manipulation process of a virtual object being used into an augmented scene. In order to simulate such manipulation, received user's requests from an input device, such as a mouse, are interpreted as being actions on the object itself. These actions make the object moving into the scene, whereas the changing views are the key idea in the achieved motion. To relate the actions to their corresponding views, user's requests are linked to the closest images that best matches the target appearance. Thus, the flow of the rendered images creates a simulation of a moving object. Hence, this paper is interested in building up a theoretical basis on how to use the relevant real objects' images for the simulation of their manipulation. The work aims at changing the views in an adequate manner so that

Manuscript Received on September 29, 2010.
E-mail：b.nini@univ-oeb.dz

the feeling of a real movement may be created.

One difficulty into which such theory is run is the connection between the limited number of object's images and the infinite views being requested. The main constraint is that the number of images, when a real object is photographed, is always bounded and they are limited to some particular viewpoints and orientations. However, to achieve the simulation, this limited number should cover any view of any natural orientation of the object. Actually, the set of available images, whatever the number is, do never satisfy such criterion. The infinite possible views lead obligatory to the missing of infinite corresponding real images.

The paper proposes a solution that states a method to allow the estimation of the closest image that reflects the best any theoretical view. To achieve such system's functions, images are linked into a geometrical relationship used as the basis for the simulation. A linear storage structure for taken object's images is stated first. It is then linked to a geometrical structure expressed in an analytical form used to model object's motion as being a combination of rotations and/or translations. This work gives an optimal solution to the connection of any appearance of an object to the closest available image. Moreover, it states a set of formulas that show how an image should be rotated and resized to fulfill the requested view before being rendered.

Meanwhile, the paper deals mainly with the effect of the manipulation of an object on its own appearance when being displaced into a scene. It does not describe how objects move into a scene; i.e., the rendering process into a real scene in relation to position, occlusion, shading, and other features is not dealt with. However, it focuses on how the views change under object's movement or the one of the camera. By this, the paper completes and extends the work presented in [8].

The latter work [8] focuses on pure rotations whereas this one considers the whole problem of motion. The previews work states a solution for the rotation of an object based on its images. This one provides an extension that shows how user's translation actions affect object's views. Then, it sets up the relationship which allows the projection of the relevant image of the estimated viewing direction. The complementary between the two works provides a full handling of the virtual objects' motion. By covering the six degrees of freedom, it becomes possible to act on the object in whatever manner a user likes. So, any input device, that is able to handle 3D operations, gives the opportunity to create any free motion.

The paper reviews what is related to the three rotational degrees of freedom operations and elucidates a general solution for translations. It explains first how any rotation of an object is handled. Then, it states some empirical relations, from the appearance point of view, which establish that any translation results in a zoom and rotation transformations. The zoom being evident, straight translations are first studied and then a generalization is done. The generalization establishes that any translation induce an implicit rotation of object's view about the axis which is the result of the cross product of the two vectors which define the origin and target positions.

Therefore, to give associated theoretical details, related works are presented in the next section in order to situate the current one relatively to them. The next section reviews the linear structure that is used to store images of an object. All the geometric aspects and their relative analytical expressions related to the simulation of the objects' motion are detailed in the succeeding section as well as the recap of the ones of rotations. It establishes the mathematical basis of how the linking between images and requested views is done. Some features of the results are outlined and discussed in the following section, and finally, a conclusion summarizes this work and gives future orientations.

II. RELATED WORK

Many works in several domains have been achieved through the use of images of real objects or scenes. An early example based on other previous works was Quick time VR [2] which suggests that the traditional modeling/rendering process can be skipped. Instead, a series of captured environment maps allow a user to look around a scene from fixed points in space. The work applies an approach which uses 360-degrees cylindrical panoramic images to compose a virtual environment. For the specific case of an object rotation, the movie contains a two-dimensional array of frames which correspond to the viewing directions of all the allowable orientations of the object. Those of [5]-[6] are other works which do not rely on geometric representations; however, they require a huge number of images. They sample and reconstruct a 4D function that generates new images of the object independent of the geometric or illumination complexity. In [11], a method is described for displaying scanned real objects. It lies in between purely model-based and image-based methods. Another important method presented in [14] allows objects to be visualized with continuous and interactive changing viewpoints. Its limit, however, is a viewpoint restriction and its limitation to the initial camera orientation. Note that most works deal more with views interpolation side of the technique than interactive changing viewpoints. A survey of the major works may be found in [13] and [15].

Most of works within this field are oriented scenes-representation. For instance, [3] gives a presentation of a z-buffered image-space-based rendering technique that allows navigation in complex static environments. [10]designed Warp Engine, which is a 3D graphic hardware architecture for real-time image-based rendering of natural scenes from arbitrary viewpoints by warping images with depth. In such work, objects are rendered with the scene but not apart; they cannot be isolated. Lately, hybrid methods have been setup for the purpose of reaching realistic views with less human cost. The work of [9] discusses one way of how IBR and GBR (Geometry Based Rendering) are mixed in a driving simulation system for traffic experiment space. On another side, [1] presents an overview of the practical implementation of an IBR system devoted for real complex photo-realistic representation scenes.

III. ORGANIZATION OF IMAGES

The simulation of the manipulation of an object in this work is based on a set of images. The way they are captured and

organized, i.e. stored, is important for their retrieval. The chosen method is the linear structure presented in [8] because of its simplicity. Since it is the basis of this work, it is reviewed hereafter.

Practically, the object is placed on a turntable, while a fixed camera pointing toward it revolves around it in a latitude direction at some fixed positions and for each one it captures a set of images when the object makes a complete rotation. In this sense, the system is setup so that the camera changes its positions along the surface of a virtual sphere in the latitude direction. Therefore, for each position of the camera, a complete rotation of the object is done. The positions of both the camera and the object when rotating are defined as being multiple of a stepangle $st = \frac{\pi}{2a}$. Note that beside the quality of photographs, the amount of st has a direct impact on the simulation of a smooth natural movement. Also, the system assumes that the relative object-camera distance d is known as well as the constant rotation speed of the turntable.

Each grab position is referred to by its spherical coordinates (γ, β, d) in the object's reference frame which is assumed positioned at its center. The angles $\gamma = n.st$ and $\beta = k.st$ are relative to (yz) and (xz) planes respectively. Particularly, the reference view is at $\gamma = 0$ and $\beta = 0$. The distance d is important for the purpose of providing a multiple resolution system (Fig. 1).

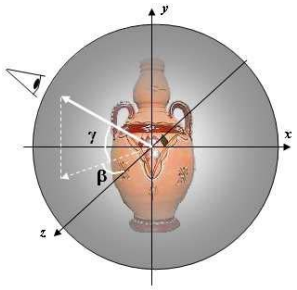

Fig. 1.The system in which the camera takes objects' images and how they are referenced.

Captured images are stored in a movie file type organized so that each image has the position p defined in (1):

$$p = n + (2k + 2r\left(1 + \frac{\pi}{st}\right) + \frac{\pi}{st})\frac{\pi}{st} \quad (1)$$

IV. THEORETICAL BASIS FOR VIEWS ESTIMATION UNDER MOTION

To create the simulation of a virtual object's motion, the system should project one previously captured image of the object that best matches the one which would be generated by a real or virtual camera at a given position with a specific viewing direction. Of course, this is not limited to one image. It is a stream of several rendered images which may create such movement. What is intended however is that this movement should be controlled in the context of a user's manipulation. This means that the system should convert requests for movement, received from a device, into actions on the virtual object. Since this is visual, these actions are simply interpreted as changes in the displayed views. Evidently, these changes are not random. Rendered images should match the views of a real situation. Consequently, the system should find out the images which correspond to those views. To reach such purpose, the process is based on a virtual camera which is supposed revolving around the object. Its positioning allows the estimation of the object's image to use in the rendering process which was taken by the real camera when it was at the closest position to the current one of the virtual camera. However, before it is rendered, the selected image may require specific transformations in agreement with the current virtual viewing direction. This rendering cycle takes place into a scene that is either real or virtual.

In order to link captured images to users' requirements, three particular orthonormal right-handed 3D coordinate systems are defined. These three reference frames are assumed moving - mainly rotating - in space governed by 'world's coordinate system' (WCS) whose origin coincides with the position of the camera associated to the scene.

The first of these three coordinate systems is the 'Object's reference frame' (ORF). It is the one which makes, under user's actions, the same supposed real rotations of the object. It reflects geometrically such rotations that are the only transformations applied to. Also, ORF's origin coincides with WCS's. Particularly, its $\vec{z}$ axis is always oriented toward the reference image. Thus, ORF is, according to this description, the coordinate system that is directly affected by the user's actions.

On another hand, 'camera viewpoint' (CVP) is another reference frame which has its $\vec{z}$ axis pointing toward the virtual camera associated to the object when revolving around it. It expresses the rotation the camera has to do when the object remains supposedly motionless in order to get the same view a user would see if it is the object which is rotating. Hence, CVP rotates whenever the object's view changes to find out the position the virtual camera should take for the target view. Like ORF, CVP's origin also coincides with WCS's and does not follow the assumed position of the object in the scene. Moreover, its motion is limited to rotations too.

It is important to note that these two reference frames, i.e.ORF and CVP, rotate in a relative opposite directions when the object is rotating (Fig. 2).

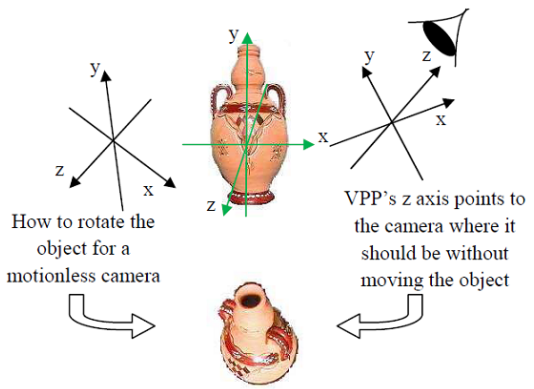

Fig. 2. Relationship between *ORF*, *CVP* and an object rotation view.

The third reference frame is the 'viewpoint position' (VPP).

It is used to control object's translations and the viewing direction in relation to its current position. For this, VPP's $\vec{z}$ axis remains oriented toward the scene's camera (WCS's origin) after any translation. It functions as the pointer to the current viewing direction of the user in the same way CVP does towards the virtual camera associated to the object, but with the difference that under pure object's rotation, CVP rotates whereas VPP does not. The latter expresses implicit rotations only resulting from translations. As such, VPP has direct effects on CVP, i.e. CVP rotates similarly, whereas CVP has no effect on VPP, i.e. VPP remains motionless under pure rotations. Moreover, the position of VPP in WCS gives the material to evaluate the amount of zoom to be applied to the projected image (Fig. 4).

For the purpose to handle analytically these three references frames, they are linked to three matrices. These matrices are simply the analytical form of the geometrical aspect of the system. Namely, two rotation matrices associated to CVP and ORF, called respectively R_C and R_O, are updated in accordance to CVP and ORF rotations. VPP is linked to a transform matrix Tr_v where only the rotation matrix part is mainly considered in the following explanation which is called R_V and updated specifically under the effect of translations. In sum and from mathematical point of view, ORF, CVP, and VPP are respectively the same as R_O, R_C, and R_V with the exception that VPP is related to a whole transform matrix since it is involved in rotations and translations at the same time. In other words, the former are the names of coordinate systems whereas the latter are their analytical representations.

The aim of this section is to establish the expressions that govern explicit and implicit combined rotations. So, in order to setup a general analytical relationship, elementary transformations are first described.

4.1 ORF and CVP's rules

Throughout the definitions of these reference frames, ORF rotates in the same direction given to the object, whereas CVP does it in the opposite direction. The basic idea is that if the object is assumed motionless and the virtual camera has to rotate about it to simulate its rotation, it has to do it in the opposite direction of the intended object's rotation. So, CVP should rotate by $-\alpha$ when the object (ORF) rotates byα. Fig.2 shows how ORF rotates to reflect the rotation of the object, and how CVP rotates to match, through its $\vec{z}$ axis, the position the real camera took when capturing the image that reflects such rotation. In fact, it is evident that rotating the camera when the object is fixed or rotating the object when the camera is fixed provides equivalent views of an object's motion.

From an analytical point of view, if the object rotates byαabout a given axis $\vec{a}$ in WCS, CVP rotates by $-\alpha$but about the transformed axis in the space of R_C. This means that the axis $\vec{a}$ should be transformed in the same way of CVP during the previous transformations so that it becomes defined relatively to R_C matrix's space. Consequently, if the axis is at a given position referenced in WCS, it should be transformed by R_C before applying the rotation of CVP (R_C) about it. Analytically, after the rotation of the object about the axis$\vec{a}$, the matrix R_O becomes:

$$R_o = R\ {}^{\alpha}_{o_{\vec{a}}}\ .R_o \tag{2}$$

And R_C becomes:

$$R_c = R\ {}^{-\alpha}_{c_{\overrightarrow{a_t}}}\ .R_c \tag{3}$$

whereas the axis $\overrightarrow{a_t}$ is:

$${a_t}^T = R_c\ .\ a^T \tag{4}$$

The notation $R^{\alpha}_{o\vec{a}}$expresses the rotation of space R_O (ORF) about $\vec{a}$byα. So,$R^{\alpha}_{o\vec{a}} = R^{-1}_{ox}\ .\ R^{-1}_{oy}\ .\ R^{\alpha}_{oz}\ .\ R_{0y}\ .\ R_{0x}$means that space R_O is rotated first about WCS's $\vec{x}$ axis (R_{0x}), then about WCS's $\vec{y}$ axis (R_{0y})so that its $\vec{z}$ axis lies along the $\vec{a}$ axis. Space R_O is then rotated about its $\vec{z}$ axis by α (R^{α}_{oz})to return finally to its relative initial position ($R^{-1}_{ox}\ .\ R^{-1}_{oy}$). The same explanation applies to space R_C (CVP).

Further details about ORF and CVP's rules can be found in [8].

4.2 VPP's principle

Translations of the object in space are assumed to be made in relation to WCS space. So, any translation is expressed according to WCS's$\vec{x}$, $\vec{y}$, and$\vec{z}$ axes. Therefore, each object's simulated translation is applied to VPP ($T_{r_v}$and indirectlyR_V) relatively to WCS.

Evidence is that any translation in any direction results, in addition to the displacement itself, in two other visual effects, namely an implicit viewing rotation and a zoom. For instance, when an object is translated left or right without rotating, an apparent rotation in its views is felt by a fixed viewer as if it is rotating. Also, the more it goes far from the viewer, the more it appears becoming smaller (Fig. 3).

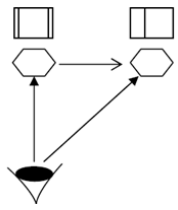

Fig. 3. Apparent rotation after object translations for a fixed viewer.

To express and quantify these viewing features, some transformations should be applied to VPP (R_V). The key idea is that the resulting rotated view under a translation is the one made by VPP so that its $\vec{z}$ axis remains oriented towards WCS's origin. In one sense, this reflects the direction of the sight from the object to a viewer positioned at WCS's origin. Since the line of sight originates from the point on object's surface that is observed, it is the direction VPP's $\vec{z}$ axis takes and points to. On another hand, the zoom is expressed based on the length of the vector $\left|\overrightarrow{d_{zoom}}\right| = \left|\overrightarrow{o_{vpp}\,.\,o_{wcs}}\right|$ which is the new position of VPP's origin in WCS (Fig. 4). Such length is simply:

$$\left|\overrightarrow{d_{zoom}}\right| = \sqrt{{d_{zoom}}_x{}^2 + {d_{zoom}}_y{}^2 + {d_{zoom}}_z{}^2} = \sqrt{{T_x}^2 + {T_y}^2 + {T_z}^2} \tag{5}$$

Note that $\overrightarrow{d_{zoom}}$ is the fourth column of the translation matrix T_V extracted fromT_{r_v}, i.e.$T_{V=}R_v^T.T_{r_v}$ Of course, in this case the dimension of both R_V and T_V is 4×4.

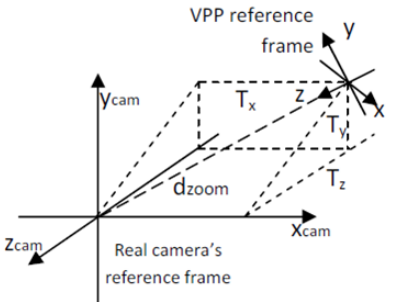

Fig. 4. *VPP*'s orientation at a given position after some translations.

Given that any translation $\vec{T}$ in space can be decomposed into three elementary ones along WCS's three axes, namely$\vec{T} = \overrightarrow{T_x} + \overrightarrow{T_y} + \overrightarrow{T_z}$, each one is first dealt with independently in the following three subsections, then the effect of an arbitrary translation on the viewing changing is generalized. The objective is then to demonstrate how to construct the transform matrix to multiply with at each step of translation. To do it, the object is assumed initially positioned at (Z_{init} ,*0,0*), where $Z_{init}\leq0$.

4.2.1 Translations along WCS's $\vec{x}$ axis

As stated before, when the object is translated from the position (T_x, T_y, T_z) to any other one along any axis parallel to WCS's $\vec{x}$ axis byΔT_x, VPP is explicitly rotated so that it maintains its $\vec{z}$ axis oriented towards WCS's origin. In this case, the rotation that is made by VPP is clearly about its $\vec{y}$ axis (Fig. 5). This observation is based on the fact that VPP's $\vec{z}$ axis remains lied in one plane. The latter is defined by VPP's previous and new origin's positions and WCS's origin. In fact, VPP's *(xy)* plan maintains evidently its inclination unchangeable during the translation. Hence, the rotation is done about VPP's $\vec{y}$ axis since it is perpendicular to its *(xz)* plane.

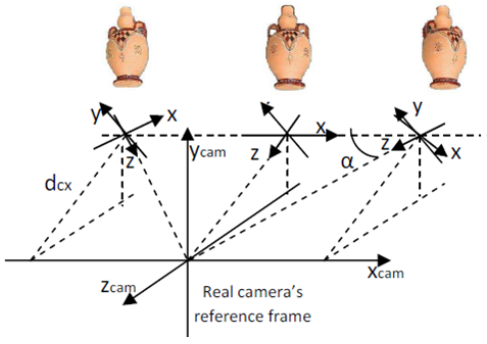

Fig. 5. Different orientations of *VPP* after some translations along *WCS*'s $\vec{x}$ axis.

Let us suppose the object at an initial position (T_{x_i}, T_y, T_z) where VPP's $\vec{z}$ axis makes an angle α_{x_i} with the axis of translation which is parallel to WCS's $\vec{x}$ axis (Fig. 6). After a translation to a target position (T_{x_t}, T_y, T_z), such angle becomesα_{x_t}. Hence, the amount of rotation made by VPP about its $\vec{y}$ axis is$|\alpha_{x_i} - \alpha_{x_t}|$. It is easy to see that$\alpha_{x_i} = tan^{-1}(\frac{d_{c_x}}{|T_{x_i}|})$ and $\alpha_{x_t} = tan^{-1}(\frac{d_{c_x}}{|T_{x_t}|})$ where as the quantity$d_{c_x} = \sqrt{T_y^2 + T_z^2}$. Note that the absolute values of T_{x_j} (j= (*i*)*nitial* or j = (*t*)*arget*) are considered because the sign is taken into consideration in (6). In fact, when the object moves in the positive direction of WCS's $\vec{x}$ axis,VPP rotates in the negative direction and vice versa. Consequently, VPP rotates about its $\vec{y}$ axis by $|\alpha_{x_i} - \alpha_{x_t}|$ in a positive or negative direction depending on the sign of $|T_{x_i} - T_{x_t}|$ which gives the sign of the angle of rotation α_{r_x}. Namely, such angle is:

$$< a_{r_x} = \frac{T_{x_i} - T_{x_t}}{|T_{x_i} - T_{x_t}|}|\alpha_{x_i} - \alpha_{x_t}| \quad (6)$$

whereas$\frac{T_{x_i} - T_{x_t}}{|T_{x_i} - T_{x_t}|} = \pm1$ depending on the direction of translation.

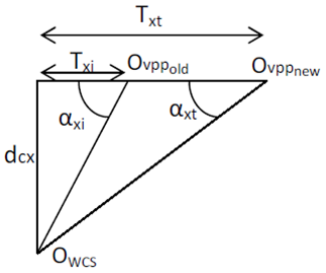

Fig. 6. Angles of rotation before and after a translation along *WCS*'s $\vec{x}$ axis.

The applied rotation to VPP is equivalent to multiply the R_v matrix by a rotation matrix about its $\vec{y}$ axis, namely:

$$R_v = R_{v_y}^{a_{r_x}}.R_v \quad (7)$$

The notation $R_{v_y}^{a_{r_x}}$ means $R_x^{-1}.R_y^{-1}.R_z^{a_{rx}}.R_x.R_y$. The expression lies the $\vec{z}$ axis of VPP along the second column of R_v, i.e. VPP's own $\vec{y}$ axis, rotates it by a_{r_x} and restores the system to its initial position.

The overall transform of the movement is updated as:

$$T_{r_v} = \begin{bmatrix} R_{v_y}^{a_{r_x}} & 0 \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & (T_{x_i} - T_{x_t}) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.T_{r_v} \quad (8)$$

Practically, this process is in fact composed of *VPP*'s rotation followed by its translation or vice versa. Indeed, *VPP* is first rotated by a_{r_x} about its $\vec{y}$ axis or any other vector that is lied on it. One vector is the result of the cross product of $\overrightarrow{O_{VPP_{x_{old}}} O_{wcs}}$ and $\overrightarrow{O_{VPP_{x_{new}}} O_{wcs}}$ which is translated so that it passes through *VPP*'s origin. $O_{VPP_{x_{old}}}$ and $O_{VPP_{x_{new}}}$ are respectively the positions of *VPP*'s origin in *WCS* space before and after its translation along *WCS*'s $\vec{x}$ axis. Consequently, either *VPP* is rotated first then translated to its target position or it is translated first then rotated, the result is the same. In fact, whatever the position of *VPP* is, it rotates always about the axis which passes through its origin. This is what explains the

truthfulness of expression (8). Note also that the quantity α_{r_x} is the angle between $\overrightarrow{O_{VPP_{x_{old}}}O_{wcs}}$ and $\overrightarrow{O_{VPP_{x_{new}}}O_{wcs}}$. This is why it can be also evaluated using the dot product; i.e.

$$< a_{r_x} = arccos\left(\frac{\overrightarrow{O_{VPP\ x_{old}}O_{wcs}}.\overrightarrow{O_{VPP\ x_{new}}O_{wcs}}}{\left|\overrightarrow{O_{VPP\ x_{old}}O_{wcs}}\right|.\left|\overrightarrow{O_{VPP\ x_{new}}O_{wcs}}\right|}\right).$$

4.2.2 Translations along WCS's $\vec{y}$ axis

For the case of a translation along a parallel axis to *WCS*'s $\vec{y}$ axis, *VPP* rotates about its $\vec{x}$ axis (Fig. 7). In fact, like in the previous case, *VPP*'s $\vec{z}$ axis remains lied in a plane, but in this case, it is *VPP*'s *(y z)* vertical plane defined by its previous and new origin's positions and origin. Similarly to the reflection made to evaluate the amount of rotation in the view of the object resulting from a translation along the $\vec{x}$ axis, it is easy to deduce that after a translation from T_x, T_{y_i}, T_z to T_x, T_{y_i}, T_z, the angle α_{y_i} made by *VPP*'s $\vec{z}$ axis and the axis of translation becomes α_{y_t}. Consequently, *VPP* rotates about its $\vec{x}$ axis by the amount of $\left|\alpha_{y_i} - \alpha_{y_t}\right|$. It is also easy to see that $\alpha_{y_i} = tan^{-1}\left(\frac{d_{c_y}}{\left|T_{y_i}\right|}\right)$ and $\alpha_{y_t} = tan^{-1}\left(\frac{d_{c_y}}{\left|T_{y_t}\right|}\right)$ whereas the quantity d_{c_y} is, in this case, $\sqrt{T_y^2 + T_z^2}$.

Unlike the previous case, however, the positive rotation of *VPP* follows the sign of $T_{y_t} - T_{y_i}$. Consequently, the angle that is made by *VPP* as a consequence of a translation following the direction of *WCS*'s $\vec{y}$ axis in order to reflect the correct view is:

$$< a_{r_y} = \frac{T_{y_i} - T_{y_t}}{\left|T_{y_i} - T_{y_t}\right|}\left|\alpha_{y_i} - \alpha_{y_t}\right| \quad (9)$$

From an analytical point of view, the rotation is expressed through R_v matrix by:

$$R_v = R_{v_x}^{a_{r_y}}.R_v \quad (10)$$

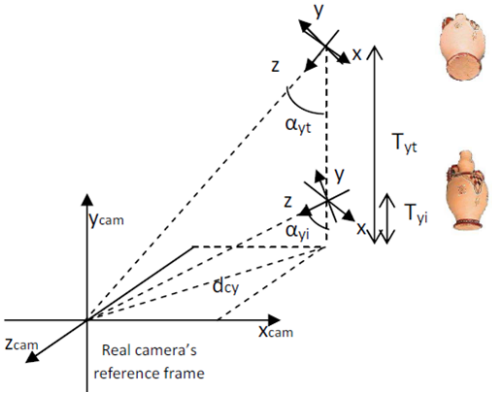

Fig. 7. Different orientations of *VPP* after some translations along *WCS*'s $\vec{y}$ axis.

The overall transform of the movement in the direction of *WCS*'s $\vec{y}$ axis is updated using $R_{v_x}^{a_{r_y}}$ and $\left[0\left(T_{y_i} - T_{y_t}\right)0\right]^T$ in the same way of expression (8).

Similarly to the previous translation, the cross product of the two vectors $\overrightarrow{O_{VPP_{y_{old}}}O_{wcs}}$ and $\overrightarrow{O_{VPP_{y_{new}}}O_{wcs}}$ is the vector that has the same direction of *VPP*'s $\vec{x}$ axis. This is evident since it should be perpendicular to *WCS*'s *(y z)* plane. So, *VPP* can be considered rotating about it by the amount of a_{r_y} after having translated it so that it passes through *VPP*'s origin. Finally, note again that *VPP* is translated to its target position either before or after its rotation which can be also evaluated as:

$$< a_{r_y} = arccos\left(\frac{\overrightarrow{O_{VPP\ y_{old}}O_{wcs}}.\overrightarrow{O_{VPP\ y_{new}}O_{wcs}}}{\left|\overrightarrow{O_{VPP\ y_{old}}O_{wcs}}\right|.\left|\overrightarrow{O_{VPP\ y_{new}}O_{wcs}}\right|}\right).$$

4.2.3 Translations along WCS's $\vec{z}$ axis

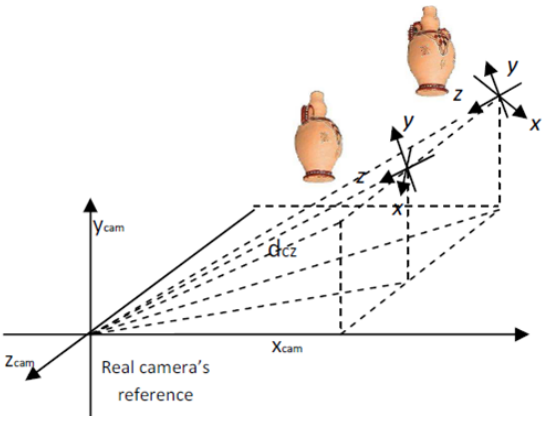

Fig. 8. Different orientations of *VPP* after some translations along WCS's $\vec{z}$ axis.

Like translations along *WCS*'s $\vec{x}$ axis, it is clear through Fig. 8 that *VPP* should make an implicit rotation around its $\vec{y}$ axis when moving along an axis parallel to *WCS*'s $\vec{z}$ axis. Hence, a translation from T_x, T_y, T_{z_i} to T_x, T_y, T_{z_t} changes the angle α_{z_i}, made by *VPP*'s $\vec{z}$ axis and the axis of translation to α_{z_t}. In this case also, $\alpha_{z_i} = tan^{-1}\left(\frac{d_{c_y}}{\left|T_{z_i}\right|}\right)$ and $\alpha_{z_t} = tan^{-1}\left(\frac{d_{c_y}}{\left|T_{z_t}\right|}\right)$ (Fig. 9). This makes the reflection similar to the one of the translation along the $\vec{x}$ axis, and hence, the same results are expected:

$$< a_{r_z} = \frac{T_{z_i} - T_{z_t}}{\left|T_{z_i} - T_{z_t}\right|}\left|\alpha_{z_i} - \alpha_{z_t}\right| \quad (11)$$

and

$$R_v = R_{v_y}^{a_{r_z}}.R_v \quad (12)$$

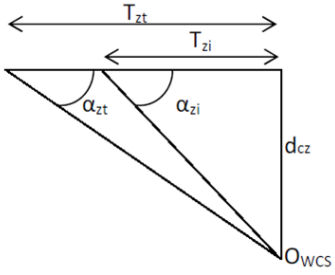

Fig. 9. Angles of rotation before and after a translation along *WCS*'s $\vec{z}$ axis.

Like the previous cases, the overall transform of the movement in the direction of *WCS*'s $\vec{z}$ axis is updated using

$R_{v_y}^{a_{r_z}}$ and $\left[0\ \left(T_{z_i}-T_{z_t}\right)\ 0\right]^T$ in the same way of expression (8).

Doing the same reflection as for the two previous cases, *VPP*'s rotation can be considered around the cross product of $\overrightarrow{O_{VPP_{z_{old}}}O_{wcs}}$ and $\overrightarrow{O_{VPP_{z_{new}}}O_{wcs}}$ that has the direction of *VPP*'s $\vec{y}$ axis and has been translated so that it passes through *VPP*'s origin. Such rotation is then followed or preceded by the translation of *VPP* to its target position. In addition, the angle of rotation can be also evaluated as:

$$< a_{r_z} = arccos\left(\frac{\overrightarrow{O_{VPP_{z_{old}}}O_{wcs}}.\overrightarrow{O_{VPP_{z_{new}}}O_{wcs}}}{\left|\overrightarrow{O_{VPP_{z_{old}}}O_{wcs}}\right|.\left|\overrightarrow{O_{VPP_{z_{new}}}O_{wcs}}\right|}\right)$$

4.2.4 Translations along arbitrary axes of WCS

Throughout the last three subsections, one can conclude that a translation along an arbitrary axis of *WCS* makes *VPP* rotates about the cross product of $\overrightarrow{O_{VPP_{z_{old}}}O_{wcs}}$ and $\overrightarrow{O_{VPP_{z_{new}}}O_{wcs}}$ in order to maintain its $\vec{z}$ axis always oriented towards *WCS*'s origin (Fig. 10). Let us define it as:

$$v_r = \overrightarrow{O_{VPP_{old}}O_{wcs}} \times \overrightarrow{O_{VPP_{new}}O_{wcs}} \quad (13)$$

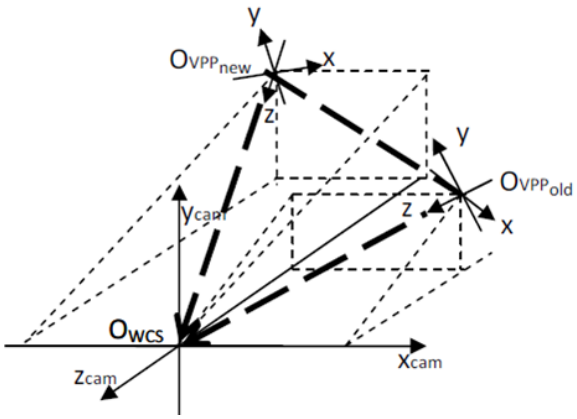

Fig. 10. A translation of *VPP* along an arbitrary axis in *WCS* reference frame.

As for the described particular translations, such axis is perpendicular to the plane defined by *VPP*'s previous and new origin's positions, and *WCS*'s origin. However, unlike the previous cases where the amount of angle of rotation is estimated in accordance with the axis of translation which is parallel to one of *WCS*'s axes, it is estimated in this case according to the arbitrary axis of translation. One can ensure that this angle can be also defined as $|\alpha_i - \alpha_t|$. But, since the dot product expression is more practical, because it is directly signed via the function *arccos*, and the former involves the evaluation of the angle's sign separately, the amount of angle between $\overrightarrow{O_{VPP_{old}}O_{wcs}}$ and $\overrightarrow{O_{VPP_{new}}O_{wcs}}$ is estimated as such:

$$< a_r = arccos\left(\frac{\overrightarrow{O_{VPP\ old}O_{wcs}}.\overrightarrow{O_{VPP\ new}O_{wcs}}}{\left|\overrightarrow{O_{VPP\ old}O_{wcs}}\right|.\left|\overrightarrow{O_{VPP\ new}O_{wcs}}\right|}\right) \quad (14)$$

Finally, the matrix $\mathbf{R_v}$ is updated in this way:

$$R_v = R_{v_{\overrightarrow{v_r}}}^{a_r}.R_v \quad (15)$$

The matrix $R_{v_{\overrightarrow{v_r}}}^{a_r}$ expresses the rotation of*VPP* about the axis $\overrightarrow{v_r}$ by an angle a_r. In general, the overall transform of the movement is:

$$T_{r_v} = \begin{bmatrix} R_{v_{\overrightarrow{v_r}}}^{a_r} & \begin{matrix}0\\0\\0\end{matrix} \\ \begin{matrix}0 & 0 & 0\end{matrix} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \left(T_{i_x}-T_{t_x}\right) \\ 0 & 1 & 0 & \left(T_{i_y}-T_{t_y}\right) \\ 0 & 0 & 1 & \left(T_{i_z}-T_{t_z}\right) \\ 0 & 0 & 0 & 1 \end{bmatrix}.T_{r_v} \quad (16)$$

Finally, as mentioned before, even for the general case, the translation of *VPP* is done before or after its rotation. What is important in this case is that these two applied operations to *VPP* are commutative. This is evident and does not require any demonstration for the fact that the rotation is always done around the axis whose origin is common with *VPP*'s.

V. SIMULATION OF MOTION

The simulation of an object's movement is achieved through linking between grabbed images and current orientations of *ORF* and *CVP*. The movement resulting from the manipulation is provided by the simulation of apparent changing views of the object in response to user's actions. These changing views are due to rotations and translations of the object. Therefore, they depend on the link between the used device and the previously described theoretical background of the simulation. Such link is achieved by the translation of user's actions on a device into transformations of *ORF*, *CVP* and *VPP* reference frames. Next, the image that best suites the position of *CVP*'s $\vec{z}$ axis is projected in respect to *ORF*'s $\vec{y}$ axis direction and after having been scaled in accordance with the position of *VPP*'s origin. This process may allow any motion included into six degrees of freedom.

In the current section, the focus is on the link of the previous theory to the interpreted user's requests acquired from the manipulation device. In other terms, which device is used and how actions are interpreted are not important. This is another part of work which is not dealt with here. The following description supposes the events of actions are already received from the device and interpreted. For instance, "revolve around *WCS*'s $\vec{z}$ axis byα", and "translate to a given position along an axis $\vec{v}$" are cases to be translated in terms of transformations to be applied to the previously defined reference frames.

5.1 Practical steps of the simulation

Explicit rotations are processed through *ORF* and *CVP* reference frames. According to [8], *ORF* and *CVP* are linked in a stated relation that makes their rotations influence each other. Under explicit rotations, this relation is taken into account obligatory. However, a user's request for an explicit rotation affects *ORF* orientation (based on expression 2) but not *CVP*. The latter is transformed by a reverse rotation using the same amount of rotation (based on expression3).

Explicit translations involve implicit rotations that are

processed through *VPP* reference frame. These apparent rotations resulting from object's translation are simulated through the link between *VPP* and *CVP*'s rotations. The two reference frames indicate two viewing directions semantically different. In fact, *CVP* indicates the current direction of a virtual camera in relation to the object, whereas *VPP* refers to the direction that faces the camera of the global scene. In other terms, the former indicates the view of the object to be projected, whereas the latter indicates the direction of the user's view. This is why whenever there is no explicit rotation, these two reference frames rotate exactly in the same way. This means that an apparent implicit rotation, done under a translation controlled by *VPP*, should be also applied to *CVP*. As far as the virtual camera is concerned, *CVP* should do the same rotation to find out which position the camera should take to reflect such orientation. The aim behind the fact that *VPP*'s rotation involves *CVP*'s one is to deduce the viewing direction of the virtual camera associated to the object. Consequently, whatever is the current orientation of *CVP* under explicit rotations, which may be different from the one of *VPP*, it should rotate again when *VPP* does rotate (expressed in(15)). This is why the expression of R_c under a translation changes as:

$$R_c = R_{c\overrightarrow{v_r}}^{a_r}.R_c \tag{17}$$

where$R_{c\overrightarrow{v_r}}^{a_r}$ expresses the rotation about $\overrightarrow{v_r}$ (deduced in (13)) about which *VPP* rotates by the angle a_r defined in(14).

Note that in the case of translations, the relation between *ORF* and *CVP* disappears since *ORF* refers to explicit real objects' rotations and this is not the case in pure translations. In translation cases, the system does not involve the rotation of *ORF*. It restricts it to *CVP* and *VPP*.

The final steps of the simulation process are similar to those presented in [8] except for the zoom which did not exist. They can be summarized into three points (for more details, see[8]):

- Locate the closest image related to the current viewpoint based on the orientation of *CVP*'s $\vec{z}$ axis. Namely, the angles ∅ and θ, made by the axis whose coordinates are R_c 's third column with *WCS*'s (*yz*) and (*xz*) planes respectively, are used. They allow the evaluation of *n* and *k* parameters in order to be used in (1):

- $$n = mod(round\left(\frac{\emptyset}{st}\right), \frac{2\pi}{st}) \tag{18}$$

  And

  $$k = round(\frac{\emptyset}{st}) \tag{19}$$

- Resize the image so that it fits with the expected view expressing the distance of the object from the viewer. The scale is based on a reference position $\overrightarrow{d_{ref}}$ of*VPP* in *WCS* which can be the initial one. The proportion p_r applied to the images is:

  $$p_r = \frac{|\overrightarrow{d_{ref}}|}{|\overrightarrow{d_{zoom}}|} \tag{20}$$

- Estimate the amount of rotation to be applied to the image and this is established through the angle between *ORF*'s $\vec{y}$ axis and *WCS*'s (*yz*) plane. Namely, it is the inclination of the axis whose coordinates are R_o's second column.

5.2 Example of simulation

Fig. (12) shows a sample of projected images corresponding to a series of simulated actions in MATLAB. Each action is assumed to translate or rotate the object. Every translation is a jumping from one position to another by a definite step $j = \frac{|\overrightarrow{d_{ref}}|}{10}$, where $|\overrightarrow{d_{ref}}|$ is assumed to be the initial distance of *VPP* from *WCS*'s origin. Rotations are also made by steps multiple of $st = \frac{\pi}{6}$.

The series of actions applied to the object of the following images in Fig. (12) are from left to right and from top to bottom: [t + + +][t + − −][t + + + +][r a +][t + − +][t 0 + −r a+[t −0 0]r a+t−0 0t+0 0. For each action, the '**t**' stands for 'translation' and '**r**' for 'rotation'. The '+' indicates a positive motion (rotation or translation), '-' a negative one, whereas '**0**' means no motion. For the translation, the three signs are linked to the *WCS*'s three axes$\vec{x}$, $\vec{y}$, and $\vec{z}$ in this order. For the rotation, '*a*' stands for a defined axis $[1\ 1\ 1]^T$ about which the object is intended to rotate. Fig.11 gives a 3D view of the translations of these series of actions in *WCS* space. Initially, the object is supposed at *WCS*'s origin, whereas the viewer (virtual camera) is at a positive position on its $\vec{z}$ axis. The graph shows how the object appears to move in front of the user.

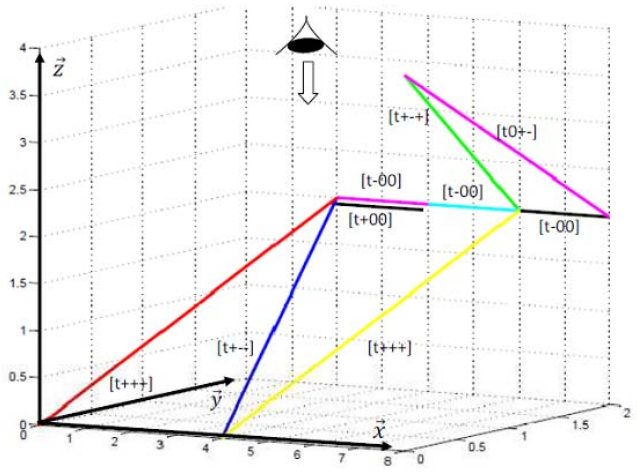

Fig. 11. Successive translations of the object in *WCS* space with *j*=2.

The sample of images of Fig. (12), which shows the effects of translation and rotation actions on the object, are not positioned in the scene. This requires further work that estimates such position which involves other features in relation to the scene. The images are limited to show the view that reflects the current translation if the object was really displaced to its target position. However, one may consider the position the image should take in the graph of Fig. (11) to understand the obtained rotation of the corresponding image in Fig. (12).

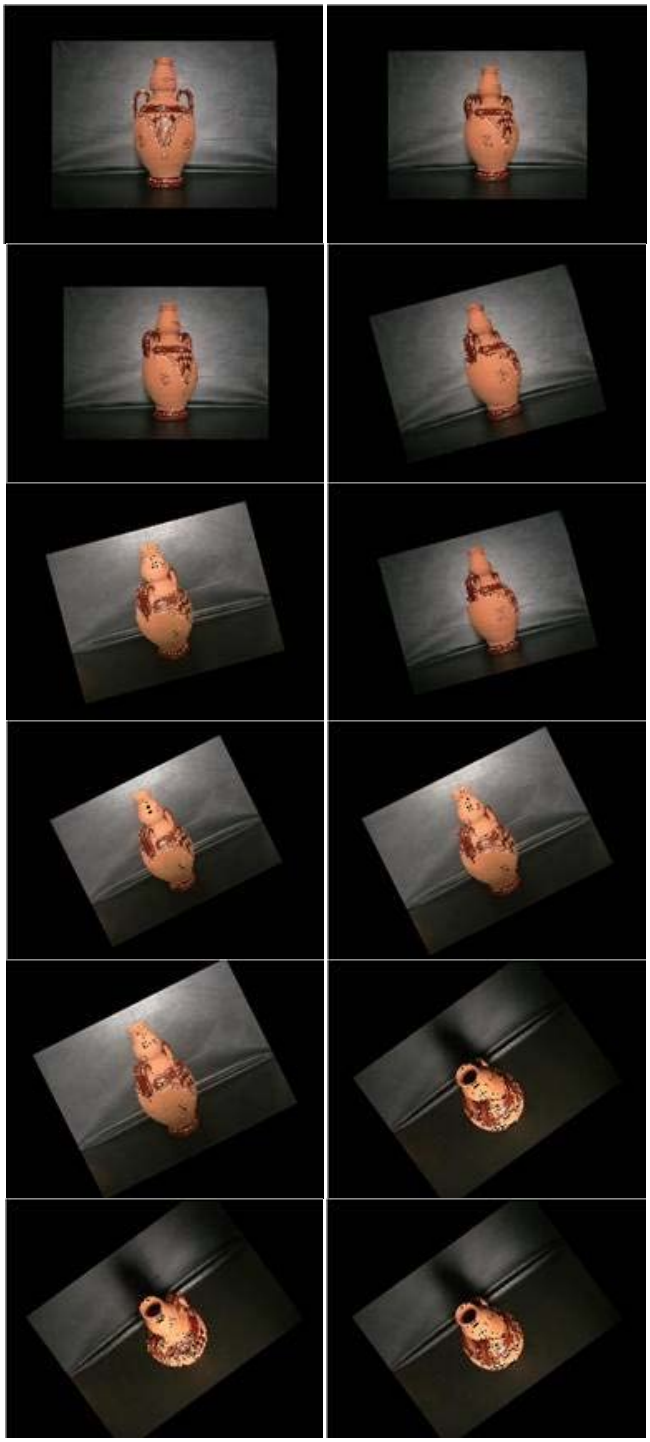

Fig. 12. Simulation of twelve steps of motion with corresponding projected images. All images are not equal in size because of the effect of the zoom due to distances that are likely to change under translation.

VI. DISCUSSION

This work provides two main results. The first one is the proposed mathematical solution dealing with image-space-based rendering systems that allows the simulation of a full handling of an object in a scene. The proposed solution is important through its simplicity. A natural and complex apparent movement may be achieved throughout simple expressions. It depends on the number of available images and on their qualities, whereas the solution itself is not affected. It is independent from such number and quality. The second result is that this solution complements the solution of a previous work[8] and hence together, they provide a complete handling of the manipulation of a virtual object. By gathering the results of the previous work and those of the current one, a user is able to manipulate an object using any device which allows a 3D manipulation through six degrees of freedom. It may be put in any position and orientation within a scene. This possibility is so important for instance in augmented reality systems.

Concerning *VPP*, this reference frame may be omitted in practical use. The two points which define the previous and new positions of object's translation are sufficient. In fact, they are enough to extract required information about $\overrightarrow{v_r}$ and the implicit rotation to be applied around it (expressions13 and14). Its introduction in the framework is important from the demonstration of established relations' point of view. It is mainly used to show the origin of the reflection.

It is also important to note that transform matrices are not directly used; the rotation ones are enough. They are useless because there is no need to control, at the same time, the whole movement of *CVP* and *ORF*. The latter are not directly concerned with translations. Moreover, it is impossible to use just one transform matrix for all the transformations mentioned above. Since they are semantically different, object's rotations require a separate matrix from the one of the virtual camera, and in this case, the use of transform matrices introduces information about the translation which is not used. In the context of this work, information of rotation and translation is not required simultaneously from each matrix. Hence, this will add new operations to the computing process without any new gain. For these reasons, the advantages of transform matrices are not actually effective at many stages of this work.

The limit of this work is not to deal with the positioning of the object into a scene; it is restricted to how the views change after any kind of motion. From this point of view, estimated views of a moving object are limited to the manner of how the corresponding images should be rendered and inserted into other images. This work is not interested in the position where they should be inserted. So, in a simulation, the understanding of the apparent movement is not easy without referring to the real motion of the object, which is the case of the images of Fig.12. This is why the sampled images should be linked to the corresponding requests in order to be understood.

VII. CONCLUSION

This paper describes a general solution to the problem of views when simulating motions of real objects through their images. The main pillar of this work is a linear structure organization which is capable of supporting a multiple level resolution. Based on this structure, a geometrical solution of object's rotations around and translations along arbitrary axes is presented. It allows the linking of images with current viewpoints of a virtual camera, i.e. the current views of the user.

Since this work is a continuation of a previous one which dealt with pure rotations, it focuses mainly on the control of translations. A link between the previous results and the newer ones is shown throughout this paper. The purpose is to cover a complete motion involving rotations and translations at the same time. This purpose is important for many kinds of applications in order to be able to control the object in a whole 3D scene as if it was real. For this, related analytical expressions are established based on geometrical reflections and supported by experimental results demonstrating their efficiency.

One extension of this work, which is not necessary, is the possibility to expand it using image interpolation results. The purpose in this case would be to add more smoothness to the realism of movements and to face the problem of storage space. Particularly, the amount of associated data would be too big when high resolution images are used.

Another extension is to link these results to an augmented real scene and deal with the effects of a real camera when it moves. This is an opened problem which considers the position of images' insertion in the scene to make the object really moving. This may allow the understanding of how the views change depending on the position of the object.

REFERENCES

[1] H. Bakstein, T. Pajdla, February 68 2006. Omni directional image-based rendering. In: Franc, V. (Ed.), *Computer Vision Winter Workshop*. p. ?

[2] S. E. Chen, 1995. Quicktimevr - an image-based approach to virtual environment navigation. In: *proceedings of Computer Graphics, Annual Conference Series*. pp. 29–38.

[3] L. Darsa, B. Costa, A. Varshney, April 1997. Navigating static environments using image-space simplification and morphing. In: *ACM SIGGRAPH, Symposium on Interactive 3D Graphics*. pp. 25–34.

[4] J.-F. Evers-Senne, R. Koch, March 2003. Image based interactive rendering with view dependent geometry. In: *Eurographics (Computer Graphics Forum)*. pp. 573–582.

[5] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen, August 1996.The lumigraph. In: *Computer Graphics Proceedings SIGGRAPH'96*. New Orleans, pp. 43–54.

[6] M. Levoy, P. Hanrahan, 1996. Light field rendering. In: *Graphics ProceedingsSIGGRAPH'96*. New Orleans, pp. 31–42.

[7] B. Nini, M. Batouche, February 6-8 2005. Virtualized real object integration and manipulation in an augmented scene. In: *11th International Conference on Computer Analysis of Images and Patterns* (CAIP 2005).Springer-Verlag, pp. 248–255.

[8] B. Nini, M. Batouche, 2007. Simulation of the handling of real objects with a complete control of rotation. *Information Technology Journal* 6 (5),672–680.

[9] S. Ono, K. Ogawara, M. Kagesawa, H. Kawasaki, M. Onuki, J. Abeki, T. Yano, M. Nerio, K. Honda, K. Ikeuchi, June 2005. A photo-realistic driving simulation system for mixed-reality traffic experiment space. In: *IEEE Intelligent Vehicles Symposium*. p. ?

[10] V. Popescu, J. Eyles, A. Lastra, J. Steinhurst, N. England, L. Nyland, July 23-28 2000. The warpengine: An architecture for the post-polygonalag. In: *ACM SIGGRAPH, 27th Annual Conference on Computer Graphics*. pp. 443–454.

[11] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, W. Stuetzle, 1997. View-based rendering: Visualizing real objects from scanned range and color data. In: *The 8th Eurographics Workshop on Rendering*. pp.23–34.

[12] R. Raskar, K. Low, G. Welch, August 2001. Shader lamps: Animatingreal objects with image-based illumination. In: *Eurographics Rendering Workshop*. p. ?21

[13] H.-Y. Shum, S. B. He, S.-C. Chan, 2003. A survey on image-based rendering representation, sampling and compression. *IEEE Transaction on CSVT*13 (11), 1020–1037.

[14] L. Wang, H.-Y. Shum, S. B. Kang, January 2002. Object representation and rendering using inverse concentric mosaics. In: *The 5th Asian Conference on Computer Vision, ACCV*. Melbourne, Australia, pp. 23–25.

[15] C. Zhang, T. Chen, 2004. A survey on image-based rendering representation, sampling and compression. *Signal Processing: Image Communication*19, 1–28.

[16] X. W. Zhong, P. Boulanger, N. D. Georganas, June 2002. Collaborative augmented reality: A prototype for industrial training. In: *21st Biennial Symposium on Communications*. Canada, p. ?22

Brahim Nini received the Doctoral degree in Computer science from the university of Constantine, Algeriain 2008. He also earns a licence in English from the university of Oum El-bouaghi, Algeria in 2009.

He is currently a Teacher-Researcher in computer science at Oum El-bouaghi university. His main fields of research are augmented reality, image encryption, and image analysis AI-based.