# Multiple Views of a Virtual World Integrated with Real Images

M. J. Abásolo, F. J. Perales
Computer Graphics and Vision Group. Department of Mathematics and Computer Science
Balearic Islands University (UIB), Spain

## Abstract

We propose a method for creating a virtual world that integrates synthetic objects with images captured with a camera. A virtual camera is defined to produce different views of the 3D synthetic objects integrated with the real image. The system includes image capturing, 3D scene creation composed with the real images, and finally the coherent integration between them. Particularly we use the graphic library of Open Inventor*, but the system could be implemented with any computer graphic tool such as VRML, Java 3D, etc.

***Keywords****: Real images, 3-D Modelling, Synthetic images, 3D-Scene, Camera views, Virtual world*

## 1. INTRODUCTION

In video production, virtual studios provide compositing of live video with synthetic or natural imagery. [1,2,3,4] present virtual studio systems, some of them of commercial usage for broadcasting. Real-time integration is done but both a complete process of camera calibration and sophisticated hardware, such as a tracking systems that allows to know the accurate state of the camera anytime, are needed. A virtual studio system divide into three main subsystems: foreground tracking, background rendering and compositing of the final image.

The goal of image integration is to create a scene in which appears that the synthetic objects occupy a particular place in the real scene. Also may be done only for artistic effect with no concern for realism, but this is not our case. Virtual shooting [2] provides a different kind of camerawork from that with an actual camera. It creates the impression of different camera work in an image already shot. First a scene is filmed (tracking), then it is processed (rendering) and later it is composited with computer graphic material to form the final image. In contrast, in our method a scene is filmed, then the filmed video is mapped onto a flat surface defined in computer graphic 3D space, and rendered together with the synthetic objects. However our method is similar to virtual shooting in the sense that produce different views of the scene from that with the real camera providing a virtual camera that makes zoom-in, zoom-out, and other movements such as pan, tilt, roll or displacements.

A simple method for creating a 3-D scene, which contains both synthetic objects and images that were captured with a camera, is used with no need of additional hardware more than a camera. The initial system that we propose for integrating real and synthetic objects doesn't want to be a professional tool for video producing. A possible use can be the construction of a 3D ambient with the added realism of a real scene as a backstage, for example in a multimedia application.

Section 2 describes the designed system and every step of the manual or semiautomatic process. Section 3 presents some results of the system that includes the generation of different views of the integrated 3D scene and the animation of the synthetic objects.

## 2. INTEGRATION SYSTEM

Figure 1 shows a scheme of the integration process that modifies the steps presented in [5] to include several cameras. Basically it consists of:

1. Capture a sequence of images of the real scene with one or more cameras.
2. Measure calibration parameters of the real scene and the captured images to finally determine the position and orientation of the real cameras.
3. Create 3D synthetic scene by using graphic libraries such as Open Inventor, VRML, Java 3D.
4. For every camera, create a 3D "screen" object in the synthetic scene for mapping the captured images onto.
5. Create a virtual camera with Open Inventor to visualise the 3-D complete scene correctly integrated.

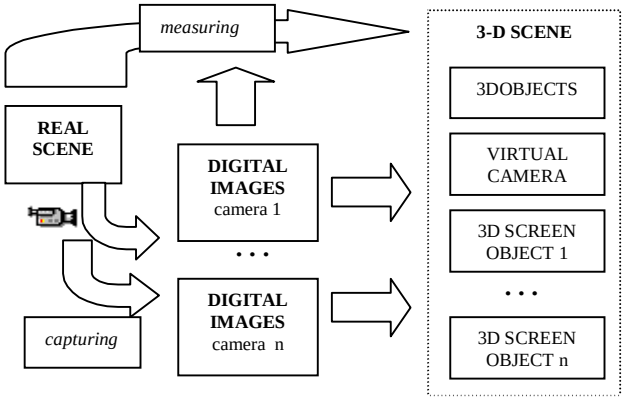



FIGURE 1. Scheme of the process

### 2.1. Capturing images

In our experiments, two cameras (Jai1510) connected to a PC by a Matrox Meteor RGB card, capture a sequence of images from a real scene. The software used for the video capturing is Inspector 2.1©. The technical aspects are not discussed here because are irrelevant in this description.

* Silicon Graphics

© Matrox Electronic Systems

We use a domestic system to facilitate the production of general applications, such as multimedia projects, videoconferences, etc.

From the capturing process an AVI format video is obtained. It is decomposed in separate images files in an Open Inventor compatible format.

## 2.2. Measuring parameters

To integrate a 3-D scene and a sequence of images captured with a camera from a real scene, a reference in the real scene is needed. The main goal is to deduce the real camera original position and orientation relative to this coordinate system. Given a set of points for which the real world coordinates (x,y,z) are known, and the corresponding points in image coordinates (u,v), it is possible to derive camera position and orientation. A popular model for this purpose is Tsai´s camera model [6]. Generally an optimised solution requires at least 11 points, though if the real-world points are coplanar, as few as 5 points suffice.

The reference point's real world coordinates must be known precisely. Determining the correspondent points in the filmed image can be done manually or by automatic 3D matching techniques.

If it is possible to obtain information of the real scene, world-coordinate reference system is set. Any reference point (x,y,z) is taken relative to this coordinate system. The corresponding points in the image (u,v) are taken relative to the centre of the image and are in pixel units.

In our case, all the measurements are done manually. If we assume a camera that only rotates in the X-Z plane, only 3 points are needed to determine the position of the real camera. In the case we take points that are parallel to the image plane X-Y, only 2 points are needed. If we have images from more than one camera, matching between the reference points and its corresponding in every image is done.

## 2.3. Creating a 3D synthetic scene

### 2.3.1. Open Inventor

Open Inventor [7] is a library of objects and methods written in C++ that is based on Open GL, used to create 3D graphic applications.

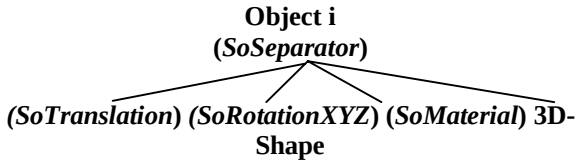
The node is the basic unit for constructing a 3-D scene. Inventor objects include data base primitives such as shape nodes that represent geometric 3-D objects; properties nodes that represent qualitative characteristic of the scene; group nodes that are containers that group other nodes and as a consequence, create a hierarchical scene.

Nodes of a scene are structure in a graph. During *rendering* process, the scene graph is visited starting with the root node, in a left-right, and top-down way. The right (and down) nodes of the graph inherit the state set by the left (and top) nodes.

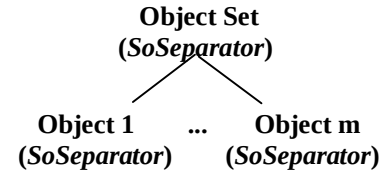
### 2.3.2. Synthetic objects creation

We create a synthetic scene by using the set of Open Inventor primitives of shape (i.e. *SoCube, SoCone, SoCylinder, SoSphere,* etc.). A specific material can be applied to the objects by the effects of a *SoMaterial* node.

*SoRotationXYZ* and *SoTranslation* nodes are used to rotate and to translate an object respectively. For every synthetic object that we want to create, we build a *SoSeparator* node that groups *SoMaterial* and *SoTransformation* node together with the node that defines the shape. The subgraph is the following:




All the 3-D object defined are grouping under a *SoSeparator* node, for the synthetic objects set be isolated from the rest of the Inventor scene. The subgraph is the following:




## 2.4. Creating the 3D screen

A 3-D "image screen object" is created with Open Inventor primitives. Three kinds of Open Inventor nodes are used: *SoFacetSet, SoTexture and SoBlinker*.

*SoFacetSet*

This class provides the shape to construct a flat surface.

*SoTexture2*

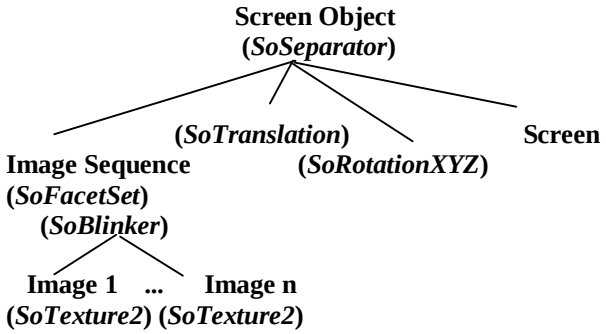
A node of this type contains a texture to be applied to an object. For every image of the video sequence a *SoTexture2* node is built. The textures are mapped to the screen object node to simulate the image "projection" onto the screen.

*SoBlinker*

A node of this type has the property of *switching* between all its children nodes at a specified speed. To simulate the video projection over the screen, the *SoTexture2* nodes are grouped under a *SoBlinker* node that switches between the images.

The *speed* attribute represents the number of complete cycles per second ( a cycle means visiting all the children nodes). The user specify a required projection speed or *frame rate* (i.e. 25 frames/sec) and the *speed* is determined as *frame rate* / *n*, with the length *n* of the image sequence. The limit of maxim speed depends on the hardware capacities.

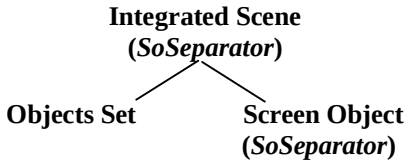
The nodes are grouped under a *SoSeparator* node in the correct order for the textures and position changes be applied to the last node. The subgraph is the following:




This graph represents a subgraph in the complete scene graph. The *SoSeparator* node isolates the effects of its children, that is the nodes inside this group don't affect upper or right in the complete scene graph.

In the next section we show how to determine the position, orientation and size of the screen object for a correct integration between the synthetic scene and the mapped images.

The overall goal is to integrate the synthetic 3-D objects set created and the real object reproduced in the image sequence. Both the subgraphs defined for the 3-D synthetic objects and the image screen object respectively, are grouping under a *SoSeparator* node forming the integrated scene graph as follow:




## 2.5. Creating a virtual camera

### 2.5.1. Virtual camera definition

In the *pinhole* camera model the images are obtained from a perspective projection as figure 2 shows. This model is characterised by an optic center **C** and the view plane **P** where the images are projected. **Zc** is the camera optic axe, and is perpendicular to plane **P**, situated at a distance **f** called focal distance. A point $p(x_c,y_c,z_c)$ referenced in the camera coordinate system is projected over the plane **P** in the point i(u,v).

The following lineal equations define the transformation from p(x,y,z) to i(u,v):

$v / y_c = \mathbf{f} / z_c$ (1)

$u / x_c = \mathbf{f} / z_c$ (2)

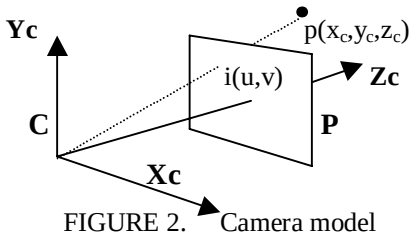
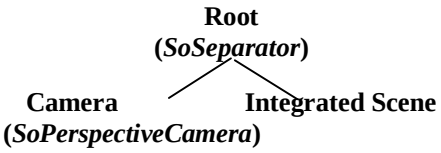



FIGURE 2. Camera model

A virtual camera implemented in a graphic workstation, makes a bidimensional projection of a 3-D synthetic world. To define a virtual camera in Open Inventor, there exists a *SoPerspectiveCamera* node that emulates a real camera. This node produces a frame from all the objects that are after it in the scene graph. The complete scene graph is as follows:




### 2.5.2. Integration between the 3D scene and the images

The overall goal of the described process is to visualise a correct integration of the 3-D synthetic objects and the image sequences captured with the cameras from the real scene. A correct visual integration means a visualisation of the global scene that gives the illusion of having both the real objects of the images and the 3-D synthetic objects in the same 3-D scene.

To render the virtual scene we have to determine the virtual camera's position and orientation. If we want to obtain a final image where the real and synthetic objects being coherently integrated, we have to emulate the position and orientation of the real camera. Besides this we have to determine the correct orientation and size of the screen object.

The screen object where the video sequence is mapped must always be centred in the optic axe and it must be parallel to the view plane of the virtual camera. Besides this the scene must be between the virtual camera and the screen object to avoid hiding the synthetic objects, means that all the synthetic objects occlude real objects.

In figure 3, we can see how the virtual camera does the projections of very point **p** of the 3-D synthetic objects and every point $\mathbf{p_s}$ of the image that is mapped onto the screen object. For a correct integration, the point **p** and its corresponding $\mathbf{p_s}$ in the mapped imaged must be projected to the same point $\mathbf{p_i}$. For that the size of the screen object must be adjusted depending on its distance $\mathbf{d_s}$ to the camera.

For every distance $\mathbf{d_s}$ and size $(\mathbf{w_s},\mathbf{h_s})$ of the screen object there exist a correct position and orientation of the camera and the screen object.

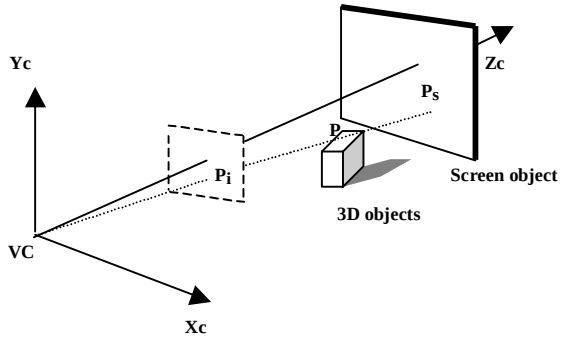



FIGURE 3. Screen object and the virtual camera

The key to accurate registration is, in turn, accurate camera calibration. As we have said, with sufficient matches between real-world coordinates (x,y,z) and the corresponding point in image coordinates (u,v), it is possible to derive camera position and orientation. Tsai´s camera model [6] has 11 parameters, including 5 internal parameters: effective focal length, first-order radial distortion coefficient, centre coordinates of the radial distortion, scale factor and 6 external parameters: camera pan, tilt and roll with respect to world coordinates, and x,y,z position in world coordinates. The model also includes several constants such as the total number of pixels in the u and v directions.

We know the dimensions wi and hi of the captured image in pixel units. Applying eq. (1) and (2) to the extreme point ($\mathbf{w}_s/2,\mathbf{h}_s/2$) of the screen object we have

$hi / h_s = \mathbf{f} / d_s$ (3)

$wi / w_s = \mathbf{f} / d_s$ (4)

where **f** is obtained**.**

Applying eq. (1) and (2) to every reference point $p_r$ we have,

$v_r / y_{rc} = \mathbf{f} / z_{rc}$ (5)

$u_r / x_{rc} = \mathbf{f} / z_{rc}$ (6)

We obtain ($u_r$, $v_r$) measuring the image.

The point ($x_{rc}$,$y_{rc}$,$z_{rc}$) is in the camera coordinate system, but is expressed it in terms of the know world-coordinates in the real scene coordinate system, and in terms of the unknown translation and rotation that suffers the camera. Resolving the set of equations the position and orientation of the camera is obtained.

If we assume a camera that only rotates in the X-Z plane, only 3 points are needed to determine its position and orientation. Particularly, with only two reference points $p_{r1}$ and $p_{r2}$ that are known parallel to the image plane X-Y, we can measure its difference $\Delta x_c$ and calculate the factor k as:

$k = (u_{r2} - u_{r1}) / \Delta x_c$ (7)

With this factor we reduce the unknown variables because can obtain the ($x_{rc}$,$y_{rc}$,$z_{rc}$) coordinates as:

$x_{rc} = u_r / k$ (8)

$y_{rc} = v_r / k$ (9)

$z_{rc} = \mathbf{f} / k$ (10)

## 3. EVALUATION OF THE SYSTEM

### 3.1. Virtual camera equivalent to the real camera

The position of the virtual camera relative to the 3-D scene corresponds to the real camera position relative to the real scene. Theoretically, if we have a synthetic scene equal to the real one, and the virtual camera positioned analogously to the real one the final image is equivalent to the image obtained with the real camera from the real scene.

Figure 4 shows an image obtained with a real camera from a scene formed of three real objects. Figure 5 shows the screen object where the image is mapped and a synthetic scene that includes a cube corresponding to the real one and other synthetic objects that don't have a corresponding in the image. Figure 6 shows how the integrated scene is visualised with a virtual camera corresponding to the real one. We can have the illusion of having both the real objects of the image and the 3-D synthetic objects in the same 3-D scene.

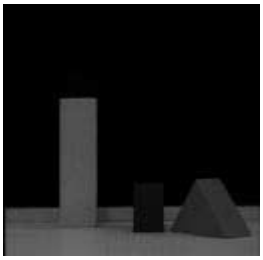


FIGURE 4. Image obtained with a real camera

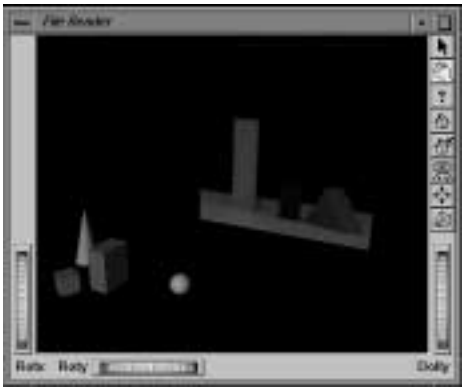


FIGURE 5. Synthetic scene with the screen object with the real image mapped onto

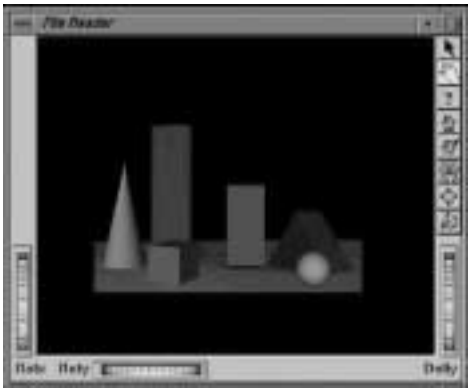


FIGURE 6. Coherent integration

In practice, there can be errors from he assumptions of an ideal camera model, such as:

- It is assumed a camera with an optical centre that intersects the view plane at its centre, but in practice there always exist a translation.
- It is assumed an ideal lens following *pinhole* model, but in practice, the lens is not ideal and not all the points are projected according to the same optical centre C, and that is call geometric distortion.

Other errors can be caused by erroneous measuring of the reference points and the matching with its corresponding in the images.

In the method described in this paper, it is not a goal to make an exact camera calibration. In contrast we evaluate the quality of the visual results obtained with the simplest process.

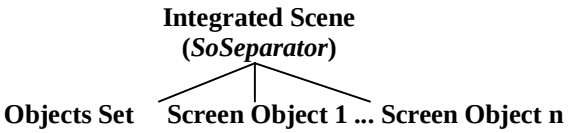
## 3.2. Different views of the 3D scene

Fixing camera keeps the coherence between the real images and the synthetic objects but seams to be limiting. The basic problem regarding 3D movement of the camera derives from the fact that the filmed image is a simple 2D surface. Consequently, if the virtual camera moves in 3D space it can not obtain proper images. Taking an extreme example, if the virtual camera direction is parallel to the surface, it will be seen as a line.

Different effective solutions are proposed in the following subsections.

### 3.2.1. Several real cameras

We can take different views of a real scene with several cameras in different key position and orientation. For every camera a screen object is constructed to map the corresponding images onto.

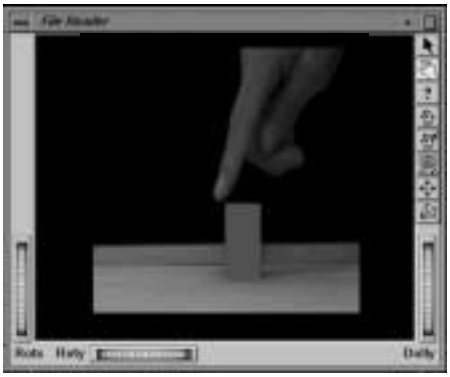



Because we have images from more than one camera, matching between the reference points and its corresponding in every image is done to determine the correct position and orientation of every camera.

We define only one virtual camera but it can change its position and orientation according the real cameras to visualize the scene from different angles. If we capture the real scene with different cameras it is possible navigating the scene coherently.

In figure 7 we can see the 3D scene formed by a synthetic object and two screen objects with the mapped images taken with two different cameras. Figure 8 a) and b) show the two different coherent views of the integrated 3D scene.

FIGURE 7. Synthetic scene with two screen objects



a)

b)

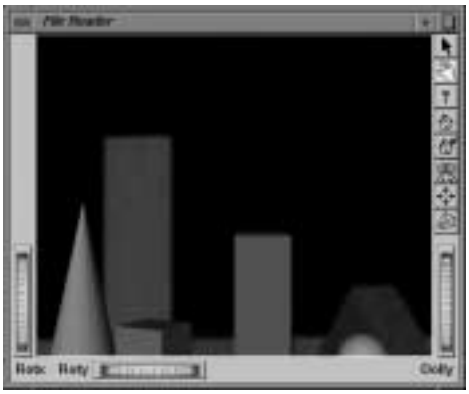
FIGURE 8. Two coherent views of the integrated scene

### 3.2.2. Movements of the virtual camera

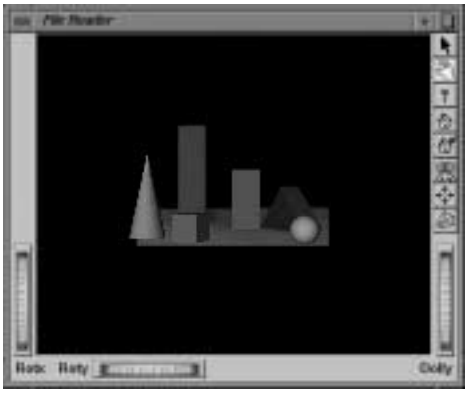
In a simple integration, we want the positions of the virtual camera and the real one coincide. In this case the camera remains at the original centre of projection, and the screen object is parallel to the view plane. A movement of the virtual camera will cause a registration error between the real image and the synthetic objects. If we want move the virtual camera we can perform geometrical transformation to the screen object to obtain the desired effect but not a mathematically correct.

One of the advantages of having a virtual camera that differs from the original real camera is that we can obtain an effects just as if we had applied camerawork completely different when the object was actually filmed. For example, we can move to a position far beyond the

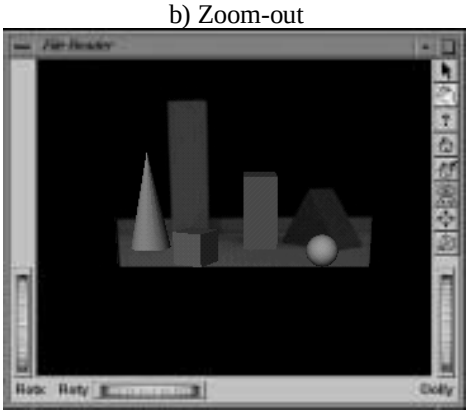
physical limitation of studio space. If we have a filmed sequence, once integrated in a virtual studio we can have a closer or a farther view of it.



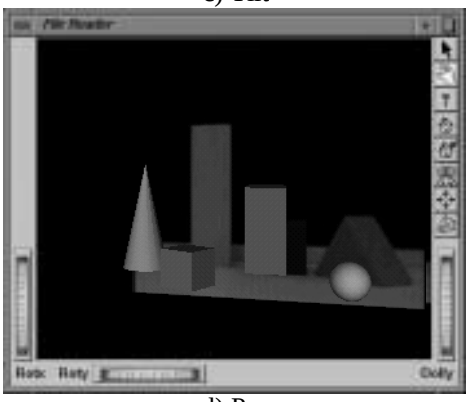
a) Zoom-in



b) Zoom-out



c) Tilt



d) Pan

FIGURE 9. Movements of the virtual camera

We can simulate a zoom-in (figure 9-a) by placing the virtual camera closer from the screen object and by augmenting its size to create a correct integration between the zoom-in image and the synthetic scene. Analogously a zoom-out (figure 9-b) is simulated by placing the virtual camera farther from the screen object and by correctly reducing its size.

The system is effective for depth direction, because almost no sense of unnaturalness is associated with movement in this direction. Rolling or rotating the camera in the optic axe can be simulated without changes in the screen object. Moving the camera position in up-down or right-left directions and tilt (figure 9-c) or pan (figure 9-d) movements produce no sense of unnaturalness in the final image if the range of movement is not too great.

### 3.2.3. Synthesizing novel views

There are methods for synthesising novel views of a 3D scene from two reference images in full correspondence. In [8], an algebraic entity termed trilinear tensor links point correspondence between three images. For any given virtual camera position and orientation, a new trilinear tensor can be computed based on the original tensor of the reference images and the desired view can be created using it and the point correspondences across two of the reference images. This method is robust enough to generate synthetic images that are far away from the original view cone.

This is an interesting way to make possible the complete navigation of the integrated 3D scene with no need to film it with different cameras.

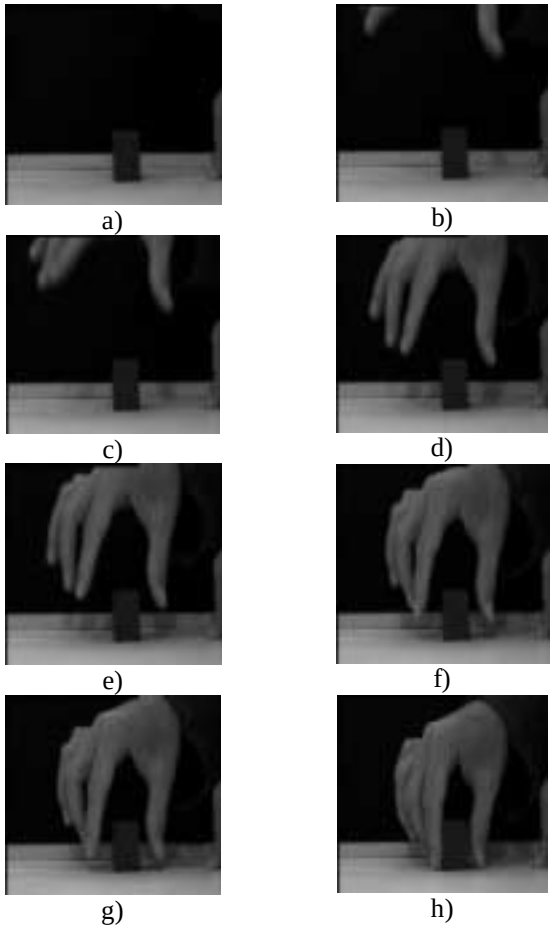
## 3.3. Animation

A captured sequence of images (figure 10 a-h) can be integrated with a synthetic scene. Figure 11-a shows a synthetic scene with several objects, and figure 11-b shows how the image sequence of figure 10 is integrated with the scene to simulate the capture of an object.

It is possible to change both the camera and screen position to simulate an interaction between the sequence of image and a certain synthetic object. For example, a sequence of images with a finger that points an object may be projected in different places according to the synthetic object to be pointed (Figure 12 a-d)

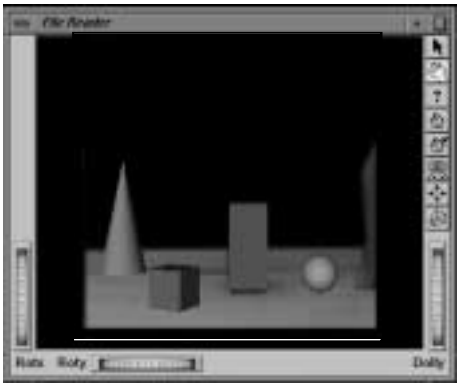
In spite of the real camera be fix real objects may be animated. If the real camera change its position or orientation it cause a registration error between the real image and the synthetic objects, so this has to be corrected by similarly displacing the virtual camera.

Virtual studios can perform many animations and special effects; for example an object can move and change their size or morph from one to another. Then it is possible to animate the synthetic objects while the sequence of the real images is projected.

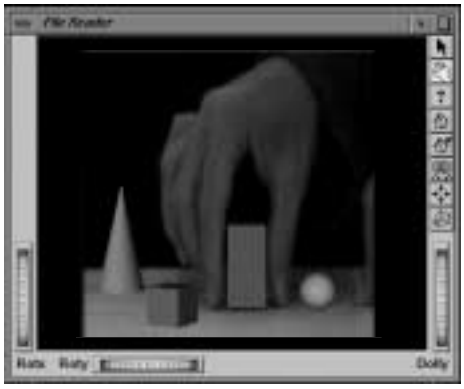


a) b) c) d) e) f) g) h)

FIGURE 10. Captured sequence of images

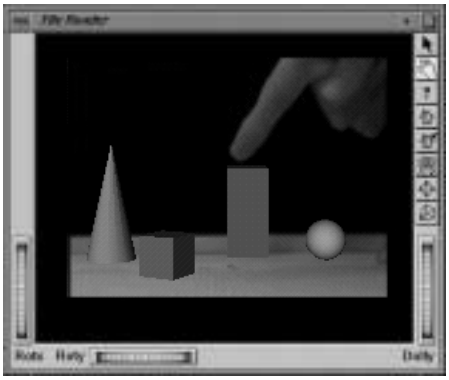


a) Synthetic scene

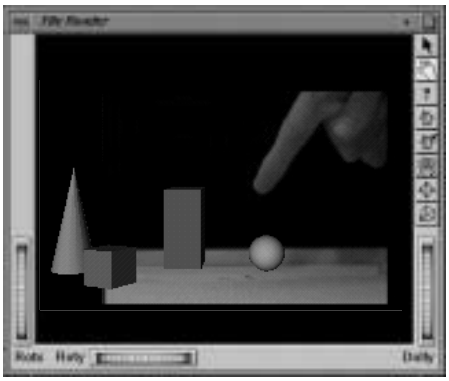


b) Sequence of images integrated with the synthetic scene

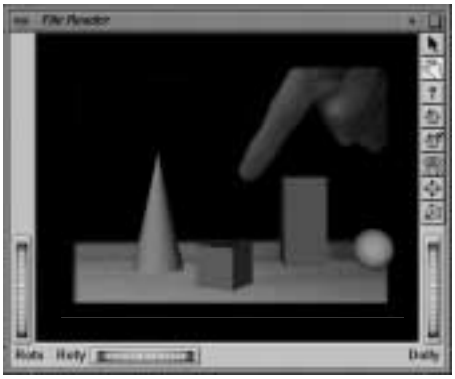
FIGURE 11. Integration of a sequence of images with a synthetic 3D scene



a) The finger points the synthetic parallelepiped



b) The finger points the synthetic sphere



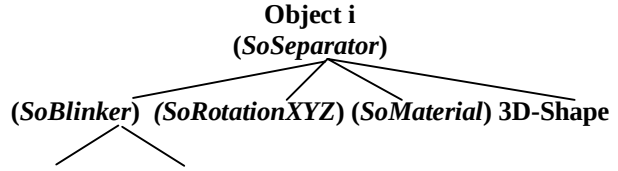
c) The finger points the synthetic cube

d) Different view of the scene of the finger that points the synthetic parallelepiped

FIGURE 12. Interaction between the real images and the synthetic objects

If the animation is done independently from the sequence of images or in an asynchronic way, the subgraph of the object includes a *SoBlinker* for switching between all its the variable children nodes that produce the animation.

The following graph is from an object that changes its position.




If we want a synchronised animation with the sequence of images, the variable part of the animated object and the sequence of images must be under the same *SoBlinker* node. Every time this node switches, the changes in the object and in the image sequence are produced. In this way, we can programme that an object reacts in some way to something that happens in the sequence of images.

## 4. CONCLUSSION

We present a simple system to produce 3-D synthetic scenes integrated with real images captured with one or more cameras. Particularly we use the graphic library Open Inventor, but in the same manner we can use VRML, Java 3D or other 3-D standard tools.

This system doesn't want to be a professional tool for video production. We present a simple process to produce 3D synthetic scenes that integrates images, with no need of complex camera calibration and no additional specific hardware more than a computer and one or more cameras. Quality of the result depends of the degree of visual coherence of the integration of 3-D synthetic objects and the images captured with the camera, and that is depending to the precision with which the manual measures are done.

Initially we consider simple scenes that include geometric objects that correspond to 3D primitives in Open Inventor. It is demonstrated that a coherent visual integration between the images of the real scene and an exactly reproduced synthetic scene may be obtained. In future applications, the type of integrated objects could be extended.

Different solutions are proposed for navigating the 3D integrated scene with visual coherence. More work may be done in this area with the methods for synthesising novel views from the captured images.

Besides the camera position and orientation, other conditions may be emulated such as lighting, environmental conditions (fog, shadows, etc.), reflections, etc. Real shadows on virtual objects might be emulated creating a synthetic object correspondent to the real one that might be invisible but used to calculate its shadows onto the other synthetic objects.

Some kind of interactions between real and synthetic objects may be modeled by using a physical model of the synthetic object that deforms under real forces.

## 5. REFERENCES

[1] S. Gibbs, C. Arapis, C. Breiteneder, V. Lalioti, S. Mostafawy, J. Speier. "Virtual Studios: An Overview". *IEEE Multimedia,* vol. 5, no. 1, p. 18-35, 1998.

[2] M. Hayashi. "Image Compositing Based on Virtual Cameras". *IEEE Multimedia,* vol. 5, no. 1, p. 36-48, 1998.

[3] L. Blondé, M. Buck, R. Galli, W. Niem, Y. Paker, W. Schmidt, G. Thomas. "A Virtual Studio for Live Broadcasting: The Mona Lisa Project". *IEEE Multimedia,* vol. 3, no. 2, p.18-29, 1996

[4] A. Iborra, M. Lázaro, P. Campoy, R. Aracíl. "Sistema de integración automática de imágenes reales con imágenes sintéticas". *CEIG' 93,* Granada, June 1993.

[5] M.J.Abásolo, F.Perales. "Generating Synthetic Images Integrated with Real Images in Open Inventor". *IV 99 International Conference on Information Visualisation,* London, England, july 1999.

[6] R.Tsai. "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision". *Proc. IEEE Computer Vision and Pattern Recognition,* p. 68-75, 1986.

[7] Wernecke, Josie. "The Inventor Mentor", Addison-Wesley Publishing Company, 1993

[8] S. Avidan, A. Shashua. "Novel View Synthesis by Cascading Trilinear Tensors". *Transactions on Visualization and Computer Graphics,* vol. 4, no. 4, p. 293-305, 1998.

**Author(s):**

María José Abásolo. CONICET (Argentina) fellow and Phd student of the Department of Mathematics and Computer Science, Balearic Islands University (Baleares, Spain).

E-mail: abasolo@ipc4.uib.es

Dr. Francisco Perales. Professor of the Department of Mathematics and Computer Science, Balearic Islands University (Baleares, Spain).

E-mail: paco@anim.uib.es

*This project is partially supported by TIC98-0302-C02-01 from CYCIT, Spain.*