IEEE Int. Conf. on Robotics and Automation, ICRA'2001
volume 1, pages 223-228, Seoul, Korea, May 2001

Computer animation: A new application for image-based visual servoing

Nicolas Courty, Éric Marchand
IRISA - INRIA Rennes
Campus de Beaulieu, 35042 Rennes Cedex, France
Email {Eric.Marchand}@irisa.fr

Abstract

This paper presents a new application for image-based visual servoing: computer graphics animation. Indeed, the control of a virtual camera in virtual environment is not a trivial problem and usually required skilled operators. Visual servoing, a now well known technique in robotics and computer vision, consists in positioning a camera according to the informations perceived in the images. Using this method within computer graphics context leads to a very intuitive approach of animation. Furthermore, in that case a full knowledge about the scene is available. It allows to easily introduce constraints within the control law in order to react automatically to modifications of the environment. In this paper, we apply this approach in two different contexts: highly reactive applications (virtual reality, video games) and the control of humanoid avatars.

1 Overview

Issues and related work. For now more than 10 years, visual servoing has been successfully and widely used to achieve various robotic tasks from assembly/disassembly tasks to docking or navigation tasks [5, 7]. A good review and introduction to visual servoing can be found in [8]. In this paper we proposed the use of this powerful framework for a new class of applications: computer graphics animation. Indeed, camera control in a virtual environment raised many difficult issues. Mainly the camera has to position itself wrt. its environment (first important issue) but it also has to react in an appropriate and efficient way to modifications of this environment (second important issue). Dealing with the first issue, even with a full knowledge of the scene, as in computer graphics, this positioning task is not a trivial problem [1]. Indeed it requires to precisely control the six degrees of freedom (d.o.f) of the camera in the 3D space. The second issue, that can be seen as the introduction of constraints in the camera trajectory, is even difficult. In order to be able to consider unknown or dynamic environments and to control in real time the motion of the camera, these constraints must be properly modeled and "added" to the positioning task. Image-based visual servoing has proved to be an efficient solution to these two problems.

Image-based control also received attention in computer graphics. The main difference wrt. computer vision or robotics is that the problem is no longer ill-posed. Indeed, in that case a full knowledge about the scene is available. Furthermore, even in an interactive context, the past and current behavior of all the objects are fully known. Ware and Osborn [21] consider various metaphors to describe a six d.o.f. camera control including "*eye in hand*". Within this context, the goal was usually to determine the position of the "eye" wrt. its six d.o.f in order to see an object or a set of objects at given locations on the screen. To control such a virtual device, people may consider user interfaces such as 3D mouse or six d.o.f joystick. Obtaining smooth camera motions required a skilled operator and has proved to be a difficult task. The classical lookat/lookfrom/vup parameterization is a simple way to achieve a focusing task on a world-space point. However specifying a complex visual task within the lookat/lookfrom framework is quite hopeless. Attempts to consider this kind of problem have been made by Blinn [1], however it appears that the proposed solutions are dedicated to specific problems and hardly scaled to more complex tasks. The image-based control have been described within the computer graphics context by Gleicher and Witkin in [6], they called it "*Through-the-lens camera control*". They proposed to achieve very simple tasks such as positioning a camera with respect to objects defined by static "virtual" points. This technique, very similar to the visual servoing framework, consider a local inversion of the nonlinear perspective viewing transformation. A constrain optimization is used to compute the camera velocity from the desired motion of the virtual points in the image. The image Jacobian is considered only for point features.

Interesting attempts to solve the introduction of constraints received great attention in both computer vision (e.g., [19]) and computer graphics [4] community. The resulting solutions are often similar. Each constraints is defined mathematically as a function of the camera parameters (location and orientation) to be minimized using deterministic (gradient approaches) or stochastic (simulated annealing) optimization processes. These approaches feature numerous drawbacks. First they are usually time consuming (the search space is of dimension six) and the optimization has to be considered for each iteration of the animation process (i.e., for each new frame). It is then difficult to consider these technics for reactive applications such as videogames. Visual servoing allows the introduction of constraints in the camera trajectory. Control laws taking into account "bad" configurations can thus to be considered [15, 14]. It combines the regulation of the vision-based task with the minimization of cost functions which reflect the constraints imposed on the trajectory. As the camera trajectory that ensure the task and the constraints is computed locally, it can be handled in real-time as required by

the considered application.

Proposed system and contributions. We aimed at the definition of basic camera trajectories for virtual movie directors as well as the automatic control of a camera for reactive applications such as video games. We assume that we fully know the model of the scene at the current instant. Within this context, we present a complete framework, based on visual servoing, that allows the definition of positioning tasks wrt. a set of "virtual visual features" located within the environment (these features can be points, lines, spheres, cylinders, etc.). When the specified task does not constrain all the camera degrees of freedom, the method allows the introduction of secondary tasks that can be achieved under the constraint that the visual task is itself achieved. Furthermore the considered features are not necessarily motionless. Using this approach we present solutions to various non-trivial problems in computer animation. Some of these tasks are more concerned with reactive applications (target tracking and following, obstacles and occlusions avoidance) while others deal with the control of digital actors.

2 Image-based camera control

Image-based visual servoing consists in specifying a task as the regulation in the image of a set of visual features[5][7]. Embedding visual servoing in the task function approach allows the use of general results helpful for the analysis and the synthesis of efficient closed loop control schemes.

Control issues Let us denote $\mathbf{P}$ the current value of the set of selected visual features used in the visual servoing task and measured from the image at each iteration of the control law. To ensure the convergence of $\mathbf{P}$ to its desired value $\mathbf{P_d}$, we need to know the interaction matrix (also called image Jacobian) $\mathbf{L_P}$ defined by the classic equation [5]:

$$\dot{\mathbf{P}} = \mathbf{L_P}(\mathbf{P}, \mathbf{p})\mathbf{T_c} \tag{1}$$

where $\dot{\mathbf{P}}$ is the time variation of $\mathbf{P}$ due to the camera motion $\mathbf{T_c}$. The parameters $\mathbf{p}$ involved in $\mathbf{L_P}(\mathbf{P}, \mathbf{p})$ represent the depth information between the considered objects and the camera frame.

A vision-based task $\mathbf{e}$ is defined by:

$$\mathbf{e} = \mathbf{W}^+\mathbf{C}(\mathbf{P} - \mathbf{P_d}) + (\mathbf{I} - \mathbf{W}^+\mathbf{W})\mathbf{e_2} \tag{2}$$

where $\mathbf{C}$, called combination matrix, has to be chosen such that $\mathbf{CL_P}(\mathbf{P}, \mathbf{p})$ is full rank about the desired trajectory $q_r(t)$. It can be defined as $\mathbf{C} = \mathbf{WL_P^+}(\mathbf{P}, \mathbf{p})$ ($\mathbf{L}^+$ denotes the pseudo inverse of $\mathbf{L}$). In that case, we set $\mathbf{W}$ as a full rank matrix such that Ker $\mathbf{W}$ = Ker $\mathbf{L_P}$. If the vision-based task does not constrain all the n robot degrees of freedom, a secondary task $\mathbf{g}_s$ can also be performed. $\mathbf{e_2}$ is the gradient of a cost function h_s to be minimized ($\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}}$). This cost function is minimized under the constraint that $\mathbf{P} = \mathbf{P_d}$. The two projection operators $\mathbf{W}^+$ and $\mathbf{I}-\mathbf{W}^+\mathbf{W}$ guarantee that the camera motion due to the secondary task is compatible with the regulation of $\mathbf{P}$ to $\mathbf{P_d}$.

To make $\mathbf{e_1}$ decrease exponentially and behave like a first order decoupled system, we get:

$$\mathbf{T_c} = -\lambda\mathbf{e} - \alpha\mathbf{T_0} - (\mathbf{I} - \mathbf{W}^+\mathbf{W})\frac{\partial \mathbf{e_2}}{\partial t} \tag{3}$$

where:

- $\mathbf{T_c}$ is the camera velocity;
- λ is the proportional coefficient involved in the exponential convergence of $\mathbf{e}$;
- the last term $\mathbf{T_0}$ (the pure target motion) allows to fully suppress the tracking errors if $\alpha = 1$.

3 Reactive viewpoint planning

The positioning tasks that can be considered within the framework presented in the previous section are quite simple. As we did not consider the environment, the target was assumed to be "alone". We now present a method that makes it possible to achieve far more complex tasks in dynamic "*cluttered environments*". We will propose a purely reactive framework in order to avoid undesirable configurations in an animation context.

3.1 Avoiding obstacles

Obstacle avoidance is a good example of what can be easily given within the proposed framework. Let us assume that the camera is moving in a cluttered environment while gazing on a visual target. The goal is to ensure this task while avoiding all the obstacles in the scene.

There are in fact multiple solutions to this problem: one solution is to planify a trajectory that avoids the obstacles using a trajectory planning process. Another solution is to consider a secondary task that uses the redundant d.o.f of the camera to move away from obstacles. This function will tend to maximize the distance between the camera and the obstacle. A good cost function to achieve the goal should be maximum (infinite) when the distance between the camera and the obstacle is null. The simplest cost function is then given by:

$$h_s = \alpha\frac{1}{2\|C - O_c\|^2} \tag{4}$$

where $C(0,0,0)$ is the camera location and $O_c(x_c, y_c, z_c)$ are the coordinates of the closest obstacle to the camera, both expressed in the camera frame (note that any other cost function that reflects a similar behavior suits the problem). If $O_s(x_s, y_s, z_s)$ are the coordinates of the obstacle within the scene frame (or reference frame) and $M_c(RT)$ the homogenous matrix that describes the camera position within this reference frame, the obstacle coordinates within the camera frame are given by $X_c = R^T X_s - R^T T$.

The components of the secondary task are given by:

$$\mathbf{e_2} = -(x_c, y_c, x_c, 0, 0, 0)^T\frac{h_s^2}{\alpha} \quad \text{and} \quad \frac{\partial \mathbf{e_2}}{\partial t} = 0 \tag{5}$$

Multiple obstacles can be handled considering the cost function $h_s = \sum_i \alpha\frac{1}{\|C-O_{c_i}\|^2}$.

3.2 Avoiding occlusions

The goal here is to avoid the occlusion of the target due to static or moving objects (with unknown motion). The virtual camera has to perform adequate motion in order to avoid the risk of occlusion while taking into account the desired constraints between the camera and the target. Related work are proposed by [10]. There are actually many situations that may evolve in an

occlusion. The first and most simple case is a moving object that crosses the camera/target line (see Figure 1.a). Two other similar cases may be encountered: in the first one (see Figure 1.b) the target moves behind another object in the scene while in the second one (see Figure 1.c) the camera follows an undesirable trajectory and is hidden behind an object.

We will now present a general image-based approach that make it possible to generate adequate camera motion automatically to avoid occlusions [14]. In a second time we will see a simple method to determine the risk of occlusion in order to weight adequately the camera response (i.e. its velocity).

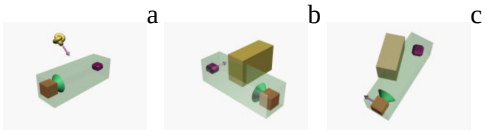

Figure 1: Occlusion issues (a) occlusion due to a moving object (b) occlusion due to the target motion (c) occlusion due to the camera motion

Automatic generation of adequate motions Let us consider $\mathcal{O}$ the projection in the image of the set of objects in the scene which may occlude the target T: $\mathcal{O} = \{O_1, \dots O_n\}$. According to the methodology presented in paragraph **??** we have to define a function h_s which reaches its maximum value when the target is occluded by another object of the scene. In fact this occlusion problem can be fully defined in the image. If the occluding object is closer than the target, when the distance between the projection of the target and the projection of the occluding object decreases, the risk of occlusion increases.

We thus define h_s as a function of this distance in the image:

$$h_s = \frac{1}{2}\alpha \sum_{i=1}^{n} e^{-\beta\left(\|T-O_i\|^2\right)} \tag{6}$$

where α and β are two scalar constants. α sets the amplitude of the control law due to the secondary task. The components of $\mathbf{e_2}$ and $\frac{\partial \mathbf{e_2}}{\partial t}$ involved in (3) are then:

$$\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}} = \frac{\partial h_s}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \mathbf{r}}, \qquad \frac{\partial \mathbf{e_2}}{\partial t} = 0$$

Computing $\frac{\partial h_s}{\partial \mathbf{P}}$ is seldom difficult. $\frac{\partial \mathbf{P}}{\partial \mathbf{r}}$ is nothing but the image Jacobian $L_{\mathbf{P}}$.

Let us consider the case of a single occluding object here considered as a point. The generalization to other and/or to multiple objects is straightforward. We want to see the target T at a given location in the image. Thus we will consider the coordinates $\mathbf{P} = (X, Y)$ as its center of gravity. If we also consider the occluding object $\mathcal{O}$ by a point $\mathbf{P}_\mathcal{O} = (X_\mathcal{O}, Y_\mathcal{O})$, defined as the closest point of $\mathcal{O}$ to T, we have:

$$h_s = \frac{1}{2}\alpha e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

and $\mathbf{e_2}$ is given by:

$$\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{r}} = \frac{\partial h_s}{\partial X}\mathbf{L}_X + \frac{\partial h_s}{\partial Y}\mathbf{L}_Y \tag{7}$$

with

$$\frac{\partial h_s}{\partial X} = -\alpha\beta(X - X_\mathcal{O})e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

and

$$\frac{\partial h_s}{\partial Y} = -\alpha\beta(Y - Y_\mathcal{O})e^{-\beta\|\mathbf{P}-\mathbf{P}_\mathcal{O}\|^2}$$

In fact $\mathbf{e_2}$ as defined in (7) is an approximation of $\frac{\partial h_s}{\partial \mathbf{r}}$. Indeed $\mathbf{L_P} = [\mathbf{L}_X, \ \mathbf{L}_Y]^T$ is the image Jacobian related to a physical point. In our case, since the point is defined as the closest point of $\mathcal{O}$ to T, the corresponding physical point will change over time. However considering $\mathbf{L}_X$ and $\mathbf{L}_Y$ in (7) is locally a good approximation.

Risk of occlusion Using the presented approach to compute the camera reaction is fine if the occluding object moves between the camera and the target [14] as depicted in Figure 1. Indeed, in that case occlusion will occur if no action is taken. However, it is neither necessary nor desirable to move the camera in all the cases (if the occluding object is farther than the target). A key point is therefore to detect if an occlusion may actually occur. In that case we first compute a bounding volume $\mathcal{V}$ that includes both the camera and the target at time t and at time $t + ndt$ assuming a constant target velocity (see Figure 2 and Figure 3). An occlusion will occur if an object is located within this bounding box. The time-to-occlusion may be computed as the smallest n for which the bounding box is empty. If an object $\mathcal{O}$ of the scene is in motion, in the same way, we consider the intersection of the volume $\mathcal{V}$ with a bounding volume that includes $\mathcal{O}$ at time t and at time $t + ndt$.

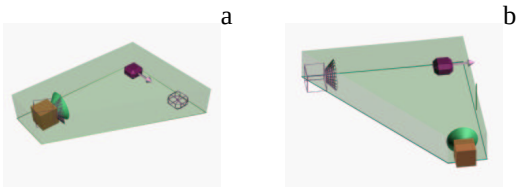

Figure 2: Computing the risk of occlusion

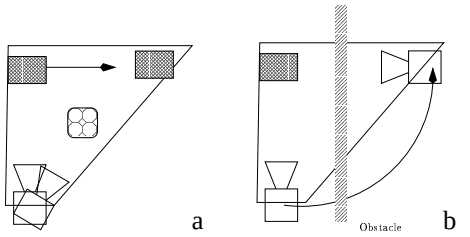

Figure 3: Detection of a future (a) occlusion (b) collision with an obstacle

Let us point out two other interesting issues:

- Obstacle avoidance may be considered in this context. Indeed, if an obstacle is on the camera trajectory, it will be located in the created bounding box (see Figure 3.b). The system will therefore forbid the camera to move in that direction.
- Some cases are more difficult to handle. A good example is a target moving in a corridor. In that case, the only solution to avoid the occlusion of the target by one of the walls

and to avoid the contact with the other wall is to reduce the camera/target distance. This can only be done if the z axis is not controlled by the primary task [13].

In conclusion, let us note that in this paragraph, we have just proposed a method to detect and quantify the risk of occlusion. The method proposed in paragraph 3.2 must be, in all cases, used to generate the adequate motion that will actually avoid occlusion. The time-to-occlusion computed here will in fact be used to set the parameter α (see equation (6)) that tunes the amplitude of the response to the risk.

4 Digital actors control

Another possible application of visual servoing in computer animation is the control of digital actors (also called virtual humanoids or avatars). Achieving humanoids control through a virtual vision process is interesting for multiple reasons:

- the sensing process is more selective since the humanoid do not have access to the whole environment data-base.
- a consequence of this selectivity process is that each avatar is more independent since it has not access to the same information that the other avatars.
- the avatar is more autonomous and its behavior will be more realistic.

Mainly two approaches have been proposed to simulate this vision process. In the former the scene observed by the digital actors is rendered and information is extracted from the resulting image using image processing algorithm [20, 17, 9]. In these approach we have only access to a limited amount of information and image processing is usually time consuming. The latter approach consider a direct access (*direct sensing*) to the object of the environment data-base in which all interesting informations are encoded, the actors are then omniscient. In this two approaches, there are few interactions between the vision process and the actor control. In many case the actor sees then it acts in a strong sequential process that is very different from the real perception process.

We think that visual servoing is a good way to achieve the low level control of such digital actors. It will allow, as in the previous experiments, a fast reactivity to external stimuli. Another interesting point in this approach is that the specification of the humanoid motions are, here again, done in the 2D image space, allowing simple automatic generation by behavioral engines ruling the humanoid. With respect to the previous experiments, the motion of virtual camera (the eyes of the humanoid) is no longer free. The eyes can be considered as mounted on the end-effector of a highly redundant robot.

In this paragraph we will show how to modify the previous control law to consider such particular robot. We show how to deal with joint limits in order to achieve realistic motions.

Control in the articular frame. In Section 2, the control laws have been expressed in the operational space (i.e., in the camera frame). However, in order to combine a visual servoing with the avoidance of joint limits, we have to directly express the control law in the articular space.

This leads to the definition of a new interaction matrix such that:

$$\dot{\mathbf{P}} = \mathbf{H_P}\ \dot{\mathbf{q}} \tag{8}$$

Since we have $\mathbf{T_c} = \mathbf{J(q)}\ \dot{\mathbf{q}}$, where $\mathbf{J(q)}$ is nothing but the robot Jacobian, we simply obtain:

$$\mathbf{H_P} = \mathbf{L_P}\ \mathbf{J(q)} \tag{9}$$

The vision-based task $\mathbf{e}_1$ is then defined by:

$$\mathbf{e_1} = \mathbf{C(P - P_d)} \tag{10}$$

where $\mathbf{C}$ can be defined as $\mathbf{C} = \mathbf{W}\mathbf{H}^{+}_{\mathbf{P}}$.

Joint limits avoidance. Joint limits avoidance is a fundamental process that has to be implemented to achieved realistic motion. The most classical way to solve the joint limits avoidance problem is to define the secondary task as the gradient of a cost function h_s ($\mathbf{e_2} = \frac{\partial h_s}{\partial \mathbf{q}}$). This cost function must reach its maximal value near a joint limits and its gradient must be equal to zero when the cost function reaches its minimal value. Several cost functions h_s which reflect this desired behavior have been presented in [18, 2, 12]. An example of such a cost function is given by:

$$h_s = \beta \sum_{i=1}^{n} (q_i - (\frac{q_{i_{min}} + q_{i_{max}}}{2}))^2 \tag{11}$$

where $q_{i_{min}}$ and $q_{i_{max}}$ are the minimum and maximum allowable joint values for the i^{th} joint. The parameter β that sets the amplitude of the control law due to the secondary task is very important. If β is too small, the change in the configuration will occur when $(\mathbf{I} - \mathbf{W}^{+}\mathbf{W})\mathbf{e_2}$ will become large wrt. the primary task. It may be too late and may produce some overshoot in the effector velocity. If β is too large, it will result in some oscillations. Therefore β is usually set based on trial and errors.

To cope with this problem we have used a fully automatic approach proposed in [3]. A good solution to achieve the avoidance task is to cut any motion on axes that are in critical area or that moves the robot toward it. Considering that $\mathbf{q}_k$ is one of these axes, we have to compute a velocity $\dot{\mathbf{q}}_k = 0$. This can be done by iteratively solving a system of linear equation. Our goal in this paper is not to describe this approach here. Let us just say this method provides a complete solution to ensure that, if a solution exists, the joints in critical situation will not encounter their limits.

5 Results

In this section some results are presented to illustrate our approach. Most of the images are generated in "real-time" (i.e. less than 0.1 s/frame without texture-mapping) on a simple SUN Ultra Sparc (170Mhz) using Mesa GL.

5.1 Avoiding occlusions: museum walkthrough.

In this example, we applied the proposed methodology to a navigation task in a complex environment. The target to be followed is moving in a museum-like environment. This "museum" has two rooms linked by stairs. The experiment goal is to keep the

target in view (i.e. to avoid occlusions) while considering *on-line* the modifications of the environment (i.e. other moving objects).

We do not address in this paper the definition of the target trajectory. Finding a path for the target is a planning problem on its own. Solutions are proposed in, e.g. [4][11]. Most of these approaches are based on a global path planning strategy (usually based on potential field approach).

In this example, we consider a focusing task wrt. an image centered virtual sphere that has to be centered in the image. This task constrains 3 d.o.f of the virtual camera (i.e. to achieve the focusing task and to maintain the radius constant in the image). The reader can refer to [5] for the complete derivation of the image Jacobian related to a sphere. Figure 4 shows the camera trajectories for various applied strategies while target and camera are moving in the first room of the environment. Obstacles appear in yellow. The target trajectory is represented as a red dotted line, while the trajectory of another moving object is represented as a blue dotted line. The red trajectory represents the simplest strategy: just focus on the object. As nothing is done to consider the environment, occlusions and then collisions with the environment occur. The blue trajectory only considers the avoidance of occlusions by static objects; as a consequence, the occlusion by the moving object occurs. The green trajectory considers the avoidance of occlusions by both static and moving objects.

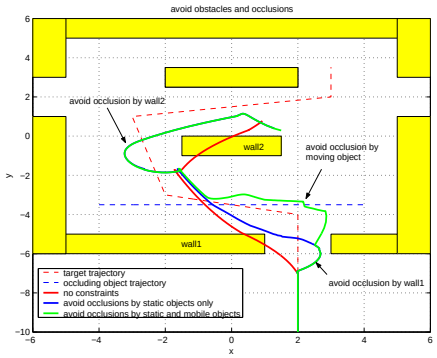

Figure 4: Museum walkthrough: camera trajectories for various strategies

Figure 5: Museum Walkthrough. The occlusions/obstacles avoidance process is not considered. This leads to multiple occlusions of the target and multiple collisions with the environment.

Figure 5 shows the views acquired by the camera if no specific strategy is considered to avoid occlusion of the target and obstacle avoidance. This leads to multiple occlusions of the target and multiple collisions with the environment. In Figures 6 the control strategy considers the presence of obstacles. This time, the target always remains in the field of view, and at its desired position in the image. The collisions with the wall and the occlusions of the target are correctly avoided. Let us note that the environment is not flat, and neither the target nor the camera move within a plane (the target "gets down" stairs on last row of Figure 6 last row). Tracking and avoidance process perform well despite the fact that the target moves in 3D. On the bird's eye view the yellow volume (associated to the camera-target couple) corresponds to the bounding volumes used to predict the occlusions.

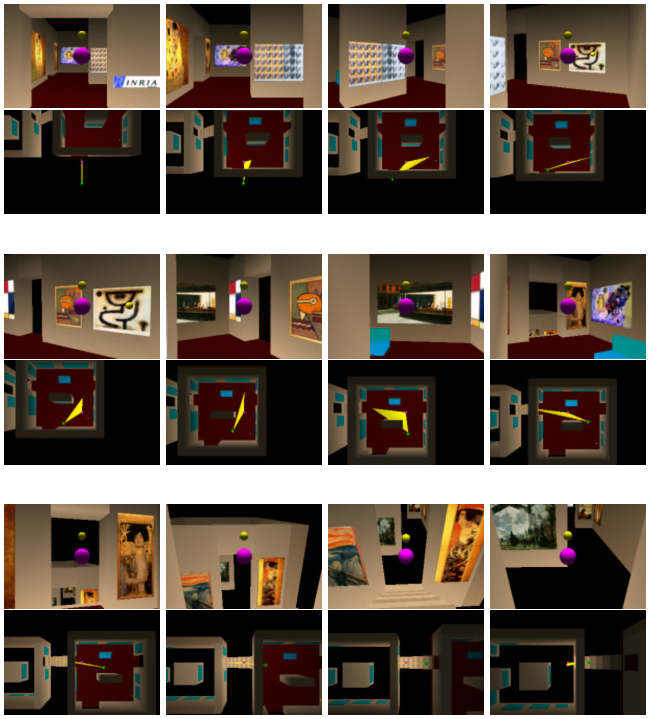

Figure 6: Museum Walkthrough: camera views and corresponding bird's eye views

5.2 Humanoid control

For the digital actor we currently consider the animation of the torso of an humanoid with 9 degrees of freedom: pelvis (3 d.o.f), spine (1 dof), neck (3 dof), eyes (2 dof). The modeling of this humanoid robot has been done using the Denavit-Hertenberg parameterization. For the moment, we have only consider simple positionning task with respect to the environment considering the joint limits avoidance issue.

Positionning wrt. a sphere. The first experiment (see Figure 7) deals with a positioning task wrt. a sphere that has to be seen centered in the image. Three dof are used to achieve this task, six dof are then free to deal with the joint limits.

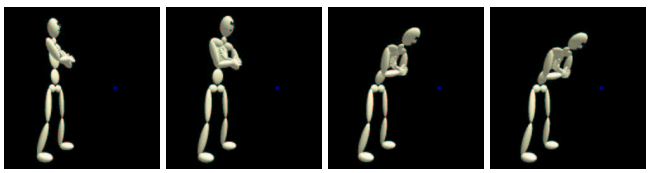

Figure 7: Humanoid control: positioning wrt. a sphere

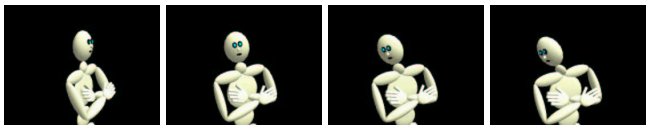

Figure 8: Humanoid control: tracking a point

Tracking task. The second experiment deals with a tracking task. It allows to demonstrate the capabilities of the joint limits avoidance algorithm. A point-object is crossing the scene. On Figure 8, we can see that, at the beginning, mainly the eyes are moving, when they move near their joint limits the motion is automatically transferred to the neck, then to the pelvis.

6 Conclusion

There are many problems associated with the management of a camera in a virtual environment. It is not only necessary to be able to carry out a visual task (often a focusing task or more generally a positioning task) efficiently, but it is also necessary to be able to react in an appropriate and efficient way to modifications of this environment. Furthermore, if we consider digital actors it is necessary to act in a realistic way. We chose to use techniques widely considered in the robotic vision community. The basic tool that we considered is visual servoing which consists in positioning a camera according to the information perceived in the image. This image-based control constitutes the first novelty of our approach. The task is indeed specified *in a 2D space*, while the resulting camera trajectories are *in a 3D space*. It is thus a very intuitive approach of animation since it is carried out according to what one wishes to observe in the resulting images sequence. This is specially true for the control of humanoid that are very difficult to control within the 3D space [16].

However, this is not the only advantage of this method. Indeed, contrary to previous work [6], we did not limit ourselves to positioning tasks wrt. virtual points in static environments. In many applications (such as video games) it is indeed necessary to be able to react to modifications of the environment, of trajectories of mobile objects, etc. We thus considered the introduction of constraints into camera control. Thanks to the redundancy formalism, the secondary tasks (which reflect the constraints on the system) do not have any effect on the visual task. To show the validity of our approach, we have proposed and implemented various classic problems from simple tracking tasks to more complex tasks like occlusion or obstacle avoidance or joint limits avoidance. The approach that we proposed has real qualities, and the very encouraging results obtained suggest that the use of visual control for computer animation is a promising technique. The main drawback is a direct counterpart of its principal quality: the control is carried out in the image, thus implying loss of control of the 3D camera trajectory or of the upper part of the humanoid body trajectory. This 3D trajectory is computed *automatically* to ensure the visual tasks but is not controlled by the animator. For this reason, one can undoubtedly see a wide interest in the use of these techniques within real-time reactive applications.

Acknowledgment. The authors wish to thank François Chaumette for his valuable comments.

Animations on-line. Most of the animations presented in this paper can be found as mpeg film on the VISTA group WWW page (`http://www.irisa.fr/vista` then follow the "demo" link).

References

[1] J. Blinn. Where am I ? what am I looking at ? *IEEE Computer Graphics and Application*, pages 76–81, July 1998.

[2] T.-F. Chang and R.-V. Dubey. A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(2):286–292, April 1995.

[3] F. Chaumette and E. Marchand. A new redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1720–1725, San Francisco, CA, Avril 2000.

[4] S.M. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface'94*, pages 190–199, Banff, Canada, 1994.

[5] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[6] M. Gleicher and A. Witkin. Through-the-lens camera control. In *ACM Computer Graphics, SIGGRAPH'92*, pages 331–340, Chicago, July 1992.

[7] K. Hashimoto. *Visual Servoing : Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific Series in Robotics and Automated Systems, Vol 7, World Scientific Press, Singapor, 1993.

[8] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[9] J.J. Kuffner and J.C Latombe. Fast synthetic vision, memory, and learning models for virtual humans. In *Computer Animation'99*, pages 118–127, Genève, Suisse, mai 1999.

[10] M. LaValle, H.-H. González-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. IEEE International Conference on Robotics and Automation, ICRA'97*, volume 1, pages 731–736, Albuquerque, NM, April 1997.

[11] T.Y. Li, J.-M. Lien, S.-Y. Chiu, and T.-H. Yu. Automatically generating virtual guided tours. In IEEE Comput. Soc, editor, *Proc. of Computer Animation 1999*, pages 99–106, Geneva, Switzerland, May 1999.

[12] E. Marchand, F. Chaumette, and A. Rizzo. Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'96*, volume 3, pages 1083–1090, Osaka, Japan, November 1996.

[13] E. Marchand and N. Courty. Image-based virtual camera motion strategies. In *Graphics Interface Conference, GI2000*, pages 69–76, 2000.

[14] E. Marchand and G.-D. Hager. Dynamic sensor planning in visual servoing. In *IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 1988–1993, Lueven, Belgium, May 1998.

[15] B. Nelson and P.K. Khosla. Integrating sensor placement and visual tracking strategies. In *IEEE Int. Conf. Robotics and Automation*, volume 2, pages 1351–1356, San Diego, May 1994.

[16] Baerlocher P. and Boulic R. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Proc. of IROS'98*, Victoria, Canada, October 1998.

[17] O. Renault, N. Magnenat-Thalmann, and D. Thalmann. A vision-based approach to behavioural animation. *Journal of Visualization and Computer Animation*, 1(1):18–21, 1990.

[18] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[19] K. Tarabanis, P.K. Allen, and R. Tsai. A survey of sensor planning in computer vision. *IEEE trans. on Robotics and Automation*, 11(1):86–104, February 1995.

[20] D. Terzopoulos and T.M. Rabie. Animat vision: Active vision with artificial animals. In *Fifth International Conf. on Computer Vision (ICCV'95)*, pages 801–808, Cambridge, MA, June 1995.

[21] C. Ware and S. Osborn. Exploration and virtual camera control in virtual three dimensional environments. In *Proc. 90 Symposium on Interactive 3D Graphics*, pages 175–183, March 1990.