4

Virtual Reality to Simulate Visual Tasks for Robotic Systems

Manuela Chessa, Fabio Solari and Silvio P. Sabatini
Department of Biophysical and Electronic Engineering, University of Genoa
Via all'Opera Pia 11/A - 16145 Genova
Italy

1. Introduction

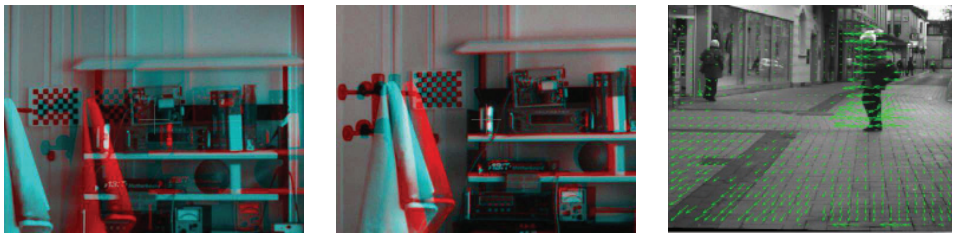
Virtual reality (VR) can be used as a tool to analyze the interactions between the visual system of a robotic agent and the environment, with the aim of designing the algorithms to solve the visual tasks necessary to properly behave into the 3D world. The novelty of our approach lies in the use of the VR as a tool to simulate the behavior of vision systems. The visual system of a robot (e.g., an autonomous vehicle, an active vision system, or a driving assistance system) and its interplay with the environment can be modeled through the geometrical relationships between the virtual stereo cameras and the virtual 3D world. Differently from conventional applications, where VR is used for the perceptual rendering of the visual information to a human observer, in the proposed approach, a virtual world is rendered to simulate the actual projections on the cameras of a robotic system. In this way, machine vision algorithms can be quantitatively validated by using the ground truth data provided by the knowledge of both the structure of the environment and the vision system.

In computer vision (Trucco & Verri, 1998; Forsyth & Ponce, 2002), in particular for motion analysis and depth reconstruction, it is important to quantitatively assess the progress in the field, but too often the researchers reported only qualitative results on the performance of their algorithms due to the lack of calibrated image database. To overcome this problem, recent works in the literature describe test beds for a quantitative evaluation of the vision algorithms by providing both sequences of images and ground truth disparity and optic flow maps (Scharstein & Szeliski, 2002; Baker et al., 2007). A different approach is to generate image sequences and stereo pairs by using a database of range images collected by a laser range-finder (Yang & Purves, 2003; Liu et al., 2008).

In general, the major drawback of the calibrated data sets is the lack of interactivity: it is not possible to change the scene and the camera point of view. In order to face the limits of these approaches, several authors proposed robot simulators equipped with visual sensors and capable to act in virtual environments. Nevertheless, such software tools are capable of accurately simulating the physics of robots, rather than their visual systems. In many works, the stereo vision is intended for future developments (Jørgensen & Petersen, 2008; Awaad et al., 2008), whereas other robot simulators in the literature have a binocular vision system (Okada et al., 2002; Ulusoy et al., 2004), but they work on stereo image pairs where parallel axis cameras are used. More recently, a commercial application (Michel, 2004) and an open source project for cognitive robotics research (Tikhanoff et al., 2008) have been developed both capable to fixate a target, nevertheless the ground truth data are not provided.

2. The visual system simulator

Figure 1a-b shows the real-world images gathered by a binocular robotic head, for different stereo configurations: the visual axes of the cameras are kept parallel (Fig. 1a) and convergent for fixating an object in the scene (the small tin, see Fig. 1b). It is worth noting that both horizontal and vertical disparities have quite large values in the periphery, while disparities are zero in the fixation point. Analogously, if we look at the motion field generated by an agent moving in the environment (see Fig. 1c), where both still and moving objects are present the resulting optic flow is composed both by ego-motion components, due to motion of the observer, and by the independent movements of the objects in the scene.



(a) (b) (c)

Fig. 1. Binocular snapshots obtained by real-world vision systems. (a)-(b): The stereo image pairs are acquired by a binocular active vision system (`http://www.searise.eu/`) for different stereo configurations: the visual axes of the cameras are (a) kept parallel, (b) convergent for fixating an object in the scene (the small tin). The anaglyphs are obtained with the left image on the red channel and the right image on the green and blue channels. The interocular distance is 30 cm and the camera resolution is 1392 $\times$ 1236 pixels with a focal length of 7.3 mm. The distance between the cameras and the objects is between 4 m and 6 m. It is worth noting that both horizontal and vertical disparities are present. (c): Optic flow superimposed on a snapshot of the relative image sequence, obtained by a car, equipped with a pair of stereo cameras with parallel visual axes , moving in a complex real environment. The resolution of the cameras is 1392 $\times$ 1040 pixels with a focal length of 6.5 mm, and the baseline is 33 cm (`http://pspc.dibe.unige.it/drivsco/`). Different situations are represented: ego-motion (due to the motion of the car) and a translating independent movement of a pedestrian (only the left frame is shown).

The aim of the work described in this chapter is to simulate the active vision system of a robot acting and moving in an environment rather than the mechanical movements of the robot itself. In particular, we aim to precisely simulate the movements (e.g. vergence and version) of the two cameras and of the robot in order to provide the binocular views and the related ground truth data (horizontal and vertical disparities and binocular motion field). Thus, our VR tool can be used for two different purposes (see Fig. 2):

1. to obtain binocular image sequences with related ground truth, to quantitatively assess the performances of computer vision algorithms;
2. to simulate the closed loop interaction between visual perception and action of the robot.

The binocular image sequences provided by the VR engine could be processed by computer vision algorithms in order to obtain the visual features necessary to the control strategy of the robot movements. These control signals act as an input to the VR engine, thus simulating the

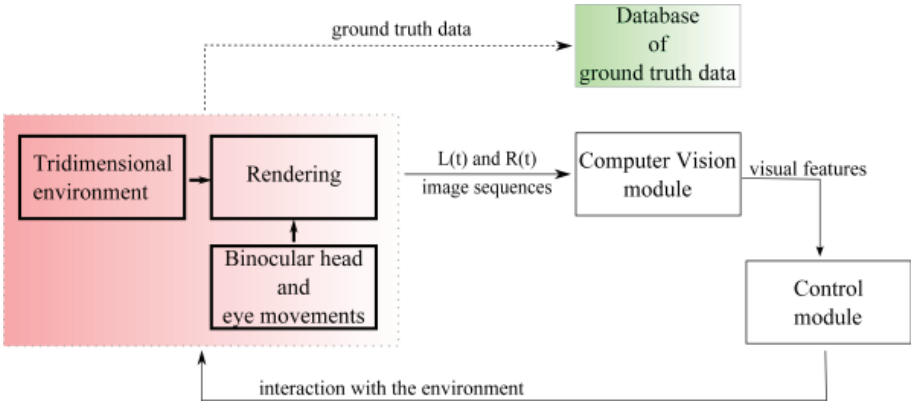


Fig. 2. The proposed active vision system simulator. Mutual interactions between a robot and the environment can be emulated to validate the visual processing modules in a closed perception-action loop and to obtain calibrated ground truth data.

robot movements in the virtual environment, then the updated binocular views are obtained. In the following, a detailed description of the model of a robotic visual system is presented.

2.1 Tridimensional environment

The 3D scene is described by using the VRML format. Together with its successor X3D, VRML has been accepted as an international standard for specifying vertices and edges for 3D polygons, along with the surface color, UV mapped textures, shininess and transparency. Though a large number of VRML models are available, e.g. on the web, they usually have not photorealistic textures and they are often characterized by simple 3D structures. To overcome this problem, a dataset of 3D scenes, acquired in controlled but cluttered laboratory conditions, has been created by using a scanner laser. The results presented in Section 6 are obtained by using the dataset obtained in our laboratory.
It is worth noting that the complex 3D VRML models can be easily replaced by simple geometric figures (cubes, cones, planes) with or without textures at any time, in order to use the simulator as an agile testing platform for the development of complex computer vision algorithms.

2.2 Rendering

The scene is rendered in an on-screen OpenGL context (see Section 5 for details). Moreover, the `SoOffScreenRenderer` class is used for rendering scenes in off-screen buffers and to save to disk the sequence of stereo pairs. The renderer can produce stereo images of different resolution and acquired by cameras with different field of views. In particular, one can set the following parameters :

- resolution of the cameras (the maximum possible resolution depends on the resolution of the textures and on the number of points of the 3D model);
- horizontal and vertical field of view (HFOV and VFOV, respectively);
- distance from camera position to the near clipping plane in the camera's view volume, also referred to as a viewing frustum, (`nearDistance`);

– distance from camera position to the far clipping plane in the camera's view volume (`farDistance`);
– distance from camera position to the point of focus (`focalDistance`).

2.3 Binocular head and eye movements

The visual system, presented in this Section, is able to generate the sequence of stereo image pairs of a binocular head moving in the 3D space and fixating a 3D point (X^F, Y^F, Z^F). The geometry of the system and the parameters that can be set are shown in Figure 3.

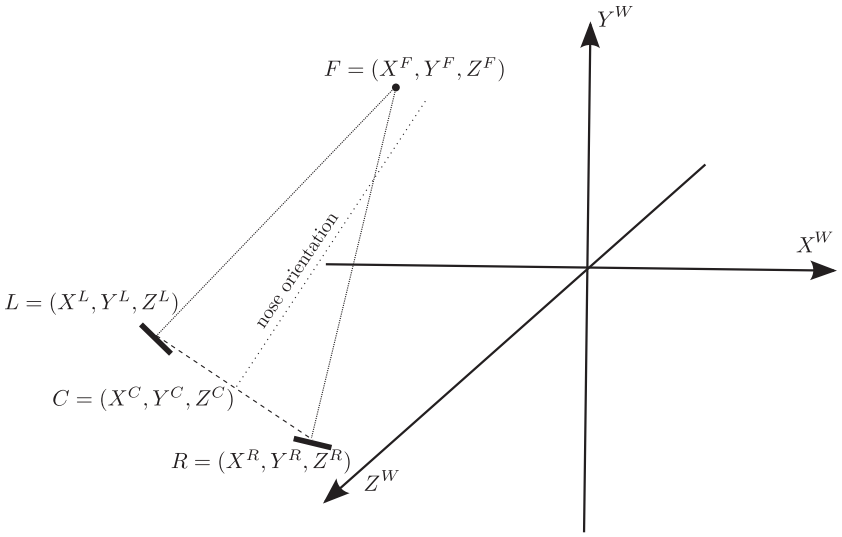


Fig. 3. Schematic representation of the geometry of the binocular active vision system.

The head is characterized by the following parameters (each expressed with respect to the world reference frame (X^W, Y^W, Z^W)):

– cyclopic position $\boldsymbol{C} = (X^C, Y^C, Z^C)$;
– nose orientation;
– fixation point $\boldsymbol{F} = (X^F, Y^F, Z^F)$.

Once the initial position of the head is fixed, then different behaviours are possible:

– to move the eyes by keeping the head (position and orientation) fixed;
– to change the orientation of the head, thus mimicking the movements of the neck;
– to change both the orientation and the position of the head, thus generating more complex motion patterns.

These situations imply the study of different perceptual problems, from scene exploration to navigation with ego-motion. Thus, in the following (see Section 6), we will present the results obtained in different situations.

For the sake of clarity and simplicity, in the following we will consider the position $\boldsymbol{C} = (X^C, Y^C, Z^C)$ and the orientation of the head fixed, thus only the ocular movements will be

considered. In Section 3.3.1 different stereo systems will be described (e.g. pan-tilt, tilt-pan, etc.), the simulator can switch through all these different behaviours. The results presented in the following consider a situation in which the eyes can rotate around an arbitrary axis, chosen in order to obtain the minimum rotation to make the ocular axis rotate from the initial position to the target position (see Section 3.3.1).

2.4 Database of ground truth data

In the literature several database of ground truth data can be found, to quantitatively assess optic flow and disparity measures.

One of the best known and widely used is the *Yosemite sequence*, that has been used extensively for experimentation and quantitative evaluation of the performances of optical flow computation techniques, camera motion estimation, and structure from motion algorithms. The data was originally generated by Lynn Quam at SRI and David Heeger (Heeger, 1987) was the first to use it for optical flow experimentation. The sequence is generated by taking an aerial image of Yosemite valley and texture mapping it onto a depth map of the valley. A synthetic sequence is generated by flying through the valley.

Other simple, but widely used, image sequences with associated ground truth data are the *Translating tree* and the *Diverging tree* by (Fleet & Jepson, 1990). Moreover, it is possible to find the *Marbled-Block sequence*, recorded and first evaluated by (Otte & Nagel, 1995), a polyhedral scene with a moving marbled block and moving camera.

A large number of algorithms for the estimation of optic flow have been benchmarked, by using these sequences. Unfortunately, it is difficult to know how relevant these results are to real 3D imagery, with all its associated complexities (for example motion discontinuities, complex 3D surfaces, camera noise, specular highlights, shadows, atmospherics, transparency). To this aim (McCane et al., 2001) have used two methods to generate more complex sequences with ground-truth data: a ray-tracer which generates optical flow, and a Tcl/Tk tool which allows them to generate ground truth optical flow from simple (i.e. polygonal) real sequences with a little help from the user.

Nevertheless, these sequences are too simple and the needs of providing more complex situation leads to the creation of databases that include much more complex real and synthetic scenes, with non-rigid motions (Baker et al., 2007). The authors rather than collecting a single benchmark dataset (with its inherent limitations), they collect four different sets, each satisfying a different subset of desirable properties. A proper combination of these datasets could be sufficient to allow a rigorous evaluation of optical flow algorithms.

Analogously, for the estimation of binocular disparity, synthetic images have been used extensively for quantitative comparisons of stereo methods, but they are often restricted to simple geometries and textures (e.g., random-dot stereograms). Furthermore, problems arising with real cameras are seldom modeled, e.g., aliasing, slight misalignment, noise, lens aberrations, and fluctuations in gain and bias. Some well known stereo pairs, with ground truth, are used by researcher to benchmark their algorithms: the *Tsukuba* stereo pair (Nakamura et al., 1996), and *Sawtooth* and *Venus* created by (Scharstein & Szeliski, 2002). Though these sequence are widely used also in recent papers, in the last years the progress in the performances of stereo algorithms is quickly outfacing the ability of these stereo data sets to discriminate among the best-performing algorithms, thus motivating the need for more challenging scenes with accurate ground truth information. To this end, (Scharstein & Szeliski, 2003) describe a method for acquiring high-complexity stereo image pairs with pixel-accurate correspondence information using structured light.

Nevertheless, databases for the evaluation of the performances of *active* stereo systems are still missing. The stereo geometry of the existing database is fixed, and characterized by parallel axis cameras. By using the software environment we developed, it is possible to collect a large number of data in different situations: e.g. vergent stereo cameras with different fixation points and orientation of the eyes, optic flow maps obtained for different ego-motion velocities, or different gaze orientation. The true disparity and optic flow maps can be stored together with the 3D data from which they have been generated and the corresponding image sequences. These data can be used for future algorithm benchmarking also by other researchers in the Computer Vision community. A tool capable of continuously generating ground truth data can be used online together with the visual processing algorithms to have a continuous assessment of their reliability. Moreover, the use of textured 3D models, acquired in real-world conditions, can solve the lack of realism that affects many datasets in the literature.

2.5 Computer vision module

Visual features (e.g. edges, disparity, optic flow) are extracted by the sequence of binocular images by the Computer Vision module. It can implement any kind of computer vision algorithm. The faithful detection of the motion and of the distance of the objects in the visual scene is a desirable feature of any artificial vision system designed to operate in unknown environments characterized by conditions variable in time in an often unpredictable way. In the context of an ongoing research project (EYESHOTS, 2008) we aimed to investigate the potential role of motor information in the early stages of human binocular vision, the computation of disparity and optic flow has been implemented in the simulator by a distributed neuromorphic architecture, described in (Chessa et al., 2009). In such distributed representations, or population codes, the information is encoded by the activity pattern of a network of simple and complex neurons, that are selective for elemental vision attributes: oriented edges, direction of motion, color, texture, and binocular disparity (Adelson & Bergen, 1991). In this way, it is possible to use the simulator to study adaptation mechanisms of the responses of the neural units on the basis of the relative orientation of the eyes.

2.6 Control module

This module generates the control signal that is responsible for the camera/eye movements, in particular for version and vergence, and for the movement of the neck (rotation and position). By considering the neck fixed, and thus focusing on eye movements only, the simulator has been exploited to study a model of vergence control based on a dual-mode paradigm (Gibaldi et al., 2010; Hung et al., 1986). The goal of the vergence control module is to produce the control signals for the eyes to bring and keep the fixation point on the surface of the object of interest without changing the gaze direction. Since the task is to nullify the disparity in fovea, the vergence control module receives inputs from the same disparity detector population response described in Section 2.5 and converts it into the speed rotation of each eye. Other control models can be easily adopted to replace the existing one, in order to achieve different behaviours or to compare different algorithms and approaches.

3. Geometry of the stereo vision

In the literature the most frequently used methods to render stereo image pairs are (Bourke & Morse, 2007; Grinberg et al., 1994): (1) the off-axis technique, usually used

to create a perception of depth for a human observer and (2) the toe-in technique that can simulate the actual intensity patterns impinging on the cameras of a robotic head.

3.1 Off-axis technique

In the off-axis technique, the stereo images are generated by projecting the objects in the scene onto the display plane for each camera; such projection plane has the same position and orientation for both camera projections. The model of the virtual setup is shown in Figure 4a: **F** represents the location of the virtual point perceived when looking at the stereo pair composed by $\mathbf{F}^L$ and $\mathbf{F}^R$.

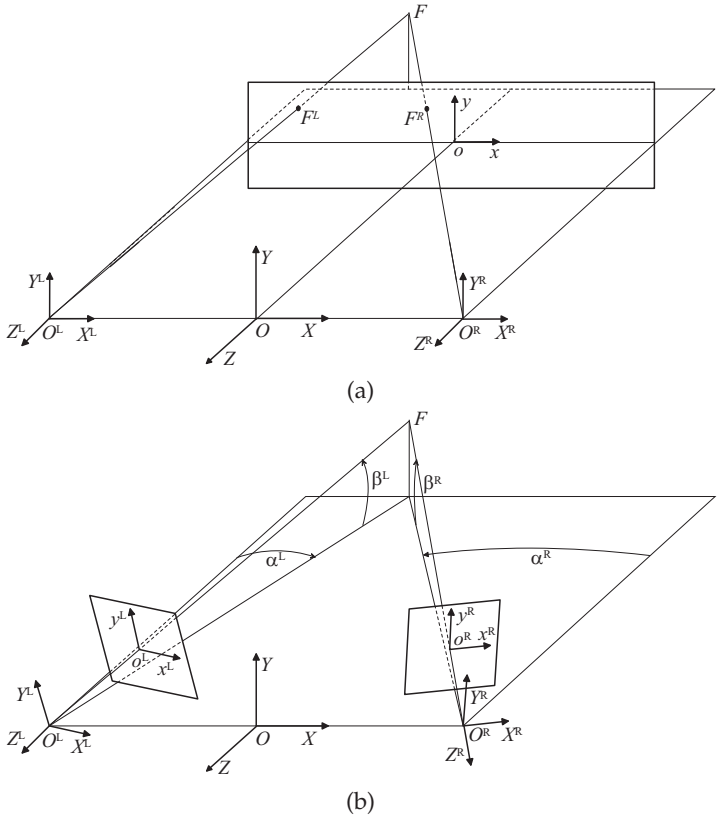


Fig. 4. (a) Geometrical sketch of the off-axis technique. The left and right camera frames: (X^L, Y^L, Z^L) and (X^R, Y^R, Z^R). The image plane (x, o, y) and the focal length Oo. The image points $\mathbf{F}^L$ and $\mathbf{F}^R$ are the stereo projection of the virtual point **F**. The baseline b is denoted by $O^L O^R$. (b) Geometrical sketch of the toe-in technique. The left and right camera frames: (X^L, Y^L, Z^L) and (X^R, Y^R, Z^R). The left and right image planes: (x^L, o^L, y^L) and (x^R, o^R, y^R). The left and right focal lengths: $O^L o^L = O^R o^R$, named f_0. The camera optical axes $O^L F$ and $O^R F$ are adjusted to fixation point **F**. The baseline b is denoted by $O^L O^R$, the pan angles by α^L and α^R, and the tilt angles by β^L and β^R.

To produce a perception of depth for a human observer, it is necessary to pay attention to some specific geometrical parameters of the stereo acquisition setup (both actual and virtual) (Grinberg et al., 1994):

- the image planes have to be parallel;
- the optical points should be offset relative to the center of the image;
- the distance between the two optical centers have to be equal to the interpupillary distance;
- the field of view of the cameras must be equal to the angle subtended by the display screen;
- the ratio between the focal length of the cameras and the viewing distance of the screen should be equal to the ratio between the width of the screen and of the image plane.

This is the correct way to create stereo pairs that are displayed on stereoscopic devices for human observers. This technique introduces no vertical disparity, thus it does not cause discomfort for the users (Southard, 1992).
However, it is difficult to perceptually render a large interval of 3D space without a visual stress, since the eye of the observer have to maintain accommodation on the display screen (at a fixed distance), thus lacking the natural relationship between accommodation and vergence eye movements, and the distance of the objects (Wann et al., 1995). Moreover, the visual discomfort is also due to spatial imperfections of the stereo image pair (Kooi & Toet, 2004). The main factors yielding visual discomfort are: vertical disparity; crosstalk, that is a transparent overlay of the left image over the right image and vice versa; blur, that is different resolutions of the stereo image pair.

3.2 Toe-in technique

Since our aim is to simulate the actual images acquired by the vergent pan-tilt cameras of a robotic head, the correct way to create the stereo pairs is the toe-in method: each camera is pointed at a single target point (the fixation point) through a proper rotation. The geometrical sketch of the optical setup of an active stereo system and of the related toe-in model is shown in Figure 4b.
It is worth noting that, for specific application fields, the toe-in technique is also used for the perceptual rendering of the stereo image pair to a human observer. In the field of the telerobotic applications (Ferre et al., 2008; Bernardino et al., 2007), it is important to perceive veridical distances in the remote environment, and the toe-in technique allows choosing where the stereo images are properly fused and the optimal remote working area. However, the parallel axes configuration is again effective when a large workspace is necessary, e.g. for exploration vehicles. The toe-in method is also helpful in the field of stereoscopic television (Yamanoue, 2006), since the perception of the 3D scene is more easily manipulated, and the objects can be seen between the observer and the display screen, i.e. it is possible to render the crossed, zero, and uncrossed disparity.
The disparity patterns produced by the off-axis and toe-in techniques are shown in Figure 5a and Figure 5b, respectively.

3.3 Mathematics of the toe-in technique

Our aim is to formally describe the toe-in technique in order to generate stereo image pairs like in a pan-tilt robotic head. To this purpose, the skewed frustum (see Fig. 6a) (necessary to obtain the off-axis stereo technique) is no longer necessary. Accordingly, we introduced the possibility of pointing the left and the right optical axes at a single 3D target point, by

rotating two symmetric frustums (see Fig. 6b), in order to obtain the left and the right views both fixating a point **F**.
In general, the two camera frames $\mathbf{X}^L$ and $\mathbf{X}^R$ are related by a rigid-body transformation in the following way:

$$\mathbf{X}^R = \mathcal{R}\mathbf{X}^L + \mathcal{T} \tag{1}$$

where $\mathcal{R}$ and $\mathcal{T}$ denote the rotation matrix and the translation, respectively. The coordinate transformation described by Eq. 1 can be converted to a linear transformation by using homogeneous coordinates (Ma et al., 2004). In the following, we use the homogeneous coordinates to describe the coordinate transformation that brings the cameras from a parallel axes configuration to a convergent one.
The translation for the left and the right view volume can be obtained by applying the following translation matrix:

$$\mathbf{T}^{L/R} = \begin{bmatrix} 1 & 0 & 0 & \pm\frac{b}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

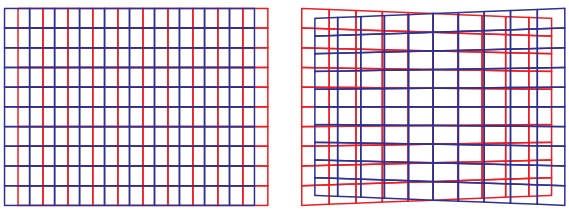
Then the azimuthal rotation (α^L and α^R) and the elevation (β^L and β^R) are obtained with the following rotation matrices:

$$\mathbf{R}_\alpha^{L/R} = \begin{bmatrix} \cos\alpha^{L/R} & 0 & \sin\alpha^{L/R} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha^{L/R} & 0 & \cos\alpha^{L/R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{R}_\beta^{L/R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta^{L/R} & -\sin\beta^{L/R} & 0 \\ 0 & \sin\beta^{L/R} & \cos\beta^{L/R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The complete roto-translation of the view-volumes is:

$$\begin{bmatrix} \mathbf{O}^{L/R} \\ 1 \end{bmatrix} = \mathbf{R}_\beta^{L/R}\mathbf{R}_\alpha^{L/R}\mathbf{T}^{L/R}\begin{bmatrix} \mathbf{O} \\ 1 \end{bmatrix} \tag{5}$$



(a) (b)

Fig. 5. The projections of a fronto-parallel square onto the image planes, drawn in red for the left image and blue for the right. The texture applied to the square is a regular grid. (a) The projection obtained with the off-axis technique: only horizontal disparity is introduced. (b) The projection obtained with the toe-in technique: both vertical and horizontal disparities are introduced.

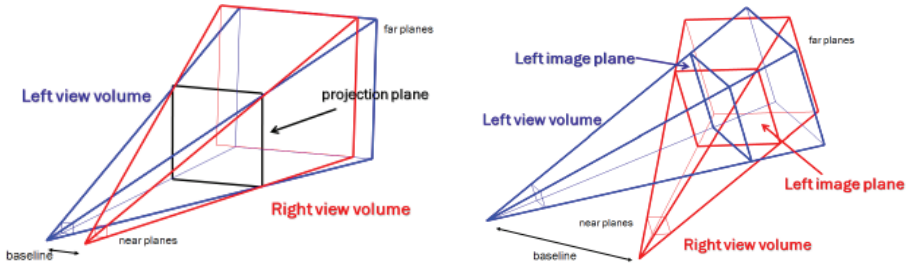


Fig. 6. (a) The two skewed frustums for the off-axis technique. (b) The two view volumes of the stereo cameras for the toe-in technique.

Thus, the projection direction is set to the target point **F**, then the left and the right views project onto two different planes, as it can be seen in Figure 4b.
In this way, it is possible to insert a camera in the scene (e.g. a perspective camera), to obtain a stereoscopic representation with convergent axes and to decide the location of the fixation point. This emulates the behavior of a couple of verging pan-tilt cameras.

3.3.1 Camera rotations

In general, the frame transformation can be described by consecutive rotations (and translations), the specific rotation described by Eq. 5 is the Helmholtz sequence (neglecting the torsion of the camera, i.e. the rotation around the visual axis). This rotation sequence is related to the gimbal system of the actual camera (we are simulating). In particular, the horizontal axis is fixed to the robotic head, and the vertical axis rotates gimbal fashion around the horizontal axis (Haslwanter, 1995). That is, first we rotate through $\beta^{L/R}$ around the horizontal axis, then we rotate through $\alpha^{L/R}$ around the new updated vertical axis.
We can simulate a different gimbal system by using the Fick sequence (i.e., the vertical axis is fixed to the robotic head), described by:

$$\begin{bmatrix} \mathbf{O}^{L/R} \\ 1 \end{bmatrix} = \mathbf{R}_{\alpha}^{L/R}\mathbf{R}_{\beta}^{L/R}\mathbf{T}^{L/R}\begin{bmatrix} \mathbf{O} \\ 1 \end{bmatrix} \tag{6}$$

It is worth noting that the fixation point is described by different values of the angles.
For non conventional cameras, e.g. (Cannata & Maggiali, 2008), it is also possible to describe the camera rotation movements from the initial position to the final one through a single rotation by a given angle γ around a fixed axis $\mathbf{a}_{\gamma}$:

$$\begin{bmatrix} \mathbf{O}^{L/R} \\ 1 \end{bmatrix} = \mathbf{R}^{L/R}(\gamma, \mathbf{a}_{\gamma})\mathbf{T}^{L/R}\begin{bmatrix} \mathbf{O} \\ 1 \end{bmatrix} \tag{7}$$

In this way, we can study how the additive degrees of freedom of non conventional (e.g. bio-inspired) systems may have effects on the computational processing of visual features.

3.3.2 General camera model

A simple and widely used model for the cameras is characterized by the following assumptions: the vertical and horizontal axes of rotation are orthogonal, through the nodal points, and aligned with the image planes. However, the commercial cameras without a

careful engineering can violate the previous assumptions, and also the cameras equipped with a zoom, since the position of the nodal point changes with respect to the position of the image plane as a function of focal length. A general camera model (Davis & Chen, 2003; Jain et al., 2006; Horaud et al., 2006) takes into account that the pan and tilt can have arbitrary axes, and the image plane are rigid objects that rotate around such axes. The actual camera geometry is described by:

$$\begin{bmatrix} \mathbf{O}^{L/R} \\ 1 \end{bmatrix} = \mathbf{T}_{pan}\mathbf{R}_{pan}\mathbf{T}_{pan}^{-1}\mathbf{T}_{tilt}\mathbf{R}_{tilt}\mathbf{T}_{tilt}^{-1}\begin{bmatrix} \mathbf{O} \\ 1 \end{bmatrix} \tag{8}$$

where $\mathbf{R}$ denotes a rotation around the tilt/pan axis, and $\mathbf{T}$ denotes a translation from the origin to each axis. In particular, the following steps are performed: first a translation $\mathbf{T}$ to the center of rotation, then a rotation $\mathbf{R}$ around the respective axis, and eventually a back translation for allowing the projection.
Figure 7 and 8 show the horizontal and vertical disparity maps for the different gimbal systems and for the general camera model. The stereo virtual cameras are fixating nine targets on a fronto-parallel plane, the central target is straight ahead and the other eight targets are symmetrically displaced at $\pm 14^{\circ}$. The baseline of the cameras is 6.5 cm with a field of view of 21°, and the plane is at 65 cm from the cameras. For the general camera model, we simulated a displacement of the nodal points of 0.6 cm, and a misalignment of the tilt and pan axes with respect to the image plane of 3°.

4. Geometry of the motion flow

In many robotic applications it is important to know how the coordinates of a point and its velocity change as the camera moves. The camera frame is the reference frame and we describe both the camera motion and the objects in the environment relative to it. The coordinates of a point $\mathbf{X}_0$ (at time $t = 0$) are described as a function of time t by the following relationship (Ma et al., 2004):

$$\mathbf{X}(t) = \mathcal{R}(t)\mathbf{X}_0 + \mathcal{T}(t) \tag{9}$$

where $\mathcal{R}(t)$ and $\mathcal{T}(t)$ denote a trajectory that describes a continuous rotational and translational motion.
From the transformation of coordinates described by Eq. 9, the velocity of the point of coordinates $\mathbf{X}(t)$ relative to the camera frame (see Fig. 9) can be derived (Longuet-Higgins & Prazdny, 1980):

$$\dot{\mathbf{X}}(t) = \boldsymbol{\omega}(t) \times \mathbf{X}(t) + \mathbf{v}(t) \tag{10}$$

where $\times$ denotes the cross product, $\boldsymbol{\omega}(t)$ and $\mathbf{v}(t)$ denote the angular velocity and the translational velocity of the camera, respectively.
Figure 10 shows the motion fields for different kinds of camera movements. For the sake of simplicity, the visual axes are kept parallel and only the left frame is shown. The virtual set-up is the same of Fig. 7 and 8.

5. Software implementation

The virtual reality tool we propose is based on a C++ / OpenGL architecture and on the Coin3D graphic toolkit (`www.coin3D.org`). Coin3D is a high level 3D graphic toolkit for developing cross-platform real time 3D visualization and visual simulation software. It is portable over a wide range of platforms, it is built on OpenGL and uses scene graph data

structures to render 3D graphics in real time. Coin3D is fully compatible with SGI Open Inventor 2.1, the de-facto standard for 3D visualization in the scientific and engineering communities. Both OpenGL and Coin3D code co-exist in our application.
In order to obtain a stereoscopic visualization of the scene useful to mimic an active stereo system, rather than to make a human perceive stereoscopy (see Section 3 for further details), we have not used the stereo rendering of the `SoCamera` node in the library, since it adopts the off-axis geometry. We have created our own `Cameras` class, that contains a pointer to a `SoPerspectiveCamera`, which can be moved in the left, right and cyclopic position. The class stores the status of the head:

– 3D position of the neck;

– projection direction of the cyclopic view;

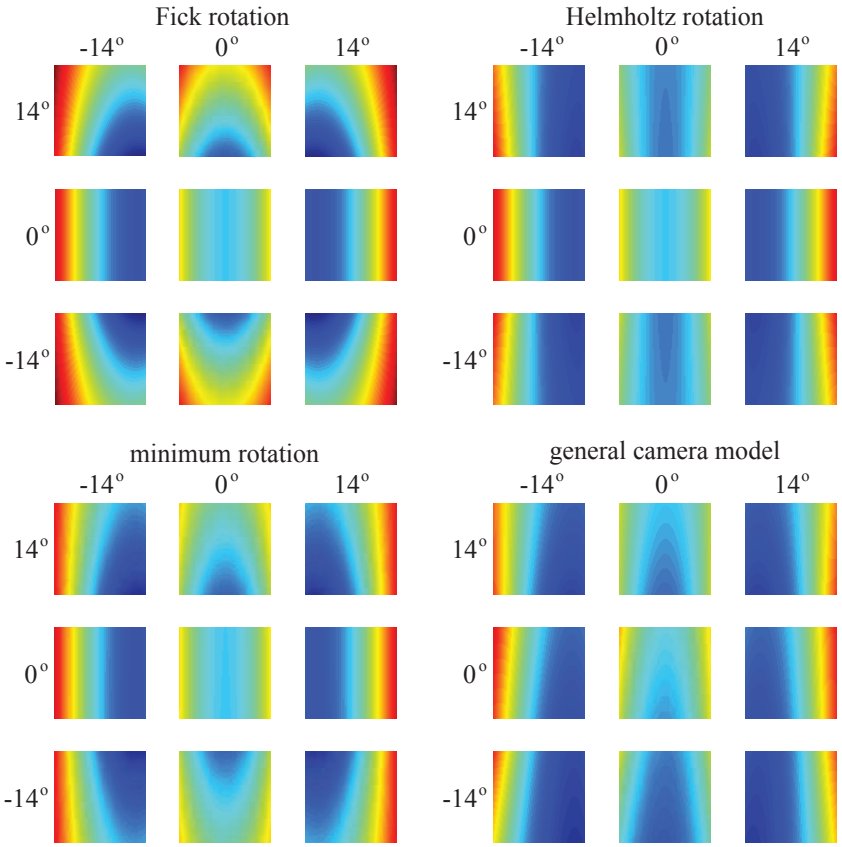


Fig. 7. Horizontal disparity patterns for different kinds of rotations and camera models. For each panel nine different gaze directions are shown. The disparity values are coded from red (uncrossed disparity) to blue (crossed disparity).

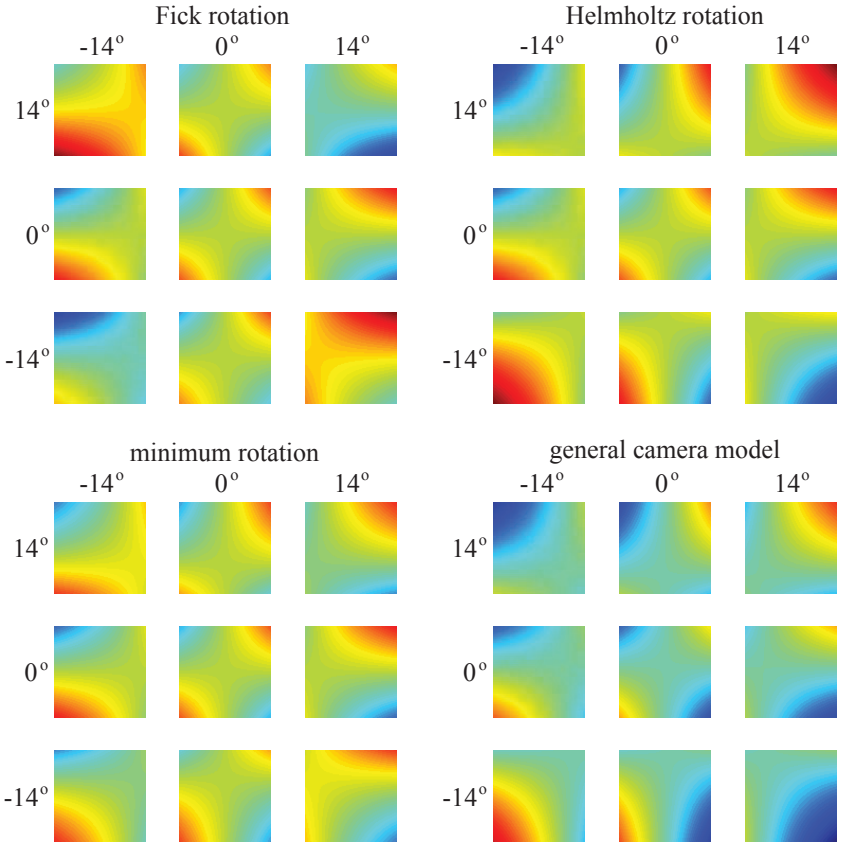


Fig. 8. Vertical disparity patterns for different kind of rotations and camera model. Same notation of Fig.'7

– direction of the baseline, computed as the cross product between the projection direction and the up vector;

and the status of each view:

– 3D position and rotation (R^R and R^L) computed with respect to the $(0,0,-1)$ axis;

– values of the depth buffer with respect to the actual position of the camera.

The left and the right views are continuously updated after having computed the rotation R^R and R^L necessary to fixate the target. Also the position of the neck, the projection direction, and the direction of the baseline can be updated if the neck is moving.
The scene from the point of view of the two stereo cameras is then rendered both in the on-screen OpenGL context and in the off-screen buffer. At the same time the depth buffer is read and stored. It is worth noting that, since Coin3D library does not easily allow the users to access and store the depth buffer, the `SoOffscreenRender` class has been modified in order to add this feature. After such a modification it is possible to access both the color buffer and the depth buffer.

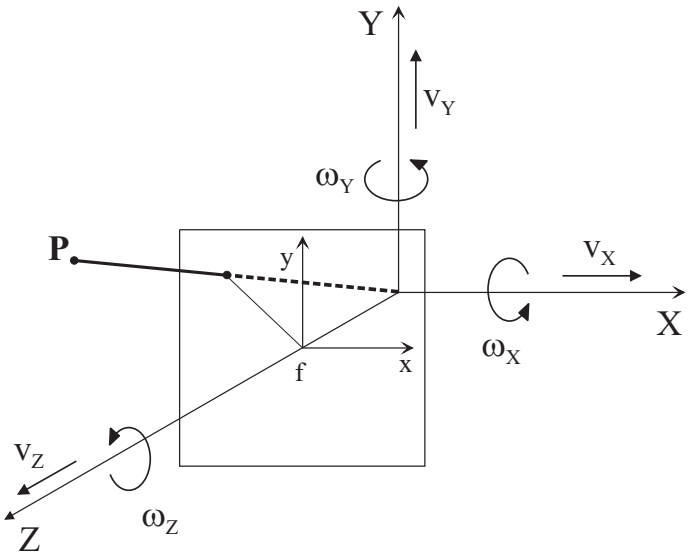


Fig. 9. Viewer-centered coordinate frame. The relative motion between an observer and the scene can be described at each instant t as a rigid-body motion, by means of two vectors (i.e., kinetic characteristics): the translational velocity $\mathbf{v} = (v_X, v_Y, v_Z)^T$, and the angular velocity $\boldsymbol{\omega} = (\omega_X, \omega_Y, \omega_Z)^T$.

The ground truth maps can be then generated and stored.

5.1 Ground truth data generation

To compute the ground truth data it is necessary to exploit the resources available from the graphics engine by combining them through the computer vision relationships that describe a 3D moving scene and the geometry of two views, typically used to obtain a 3D reconstruction.

5.1.1 Stereo cameras

Formally, by considering two static views, the two camera reference frames are related by a rigid body transformation described by the rotation matrix $\mathcal{R}$ and the translation $\mathcal{T}$ (see Eq. 1), thus the two projections (left and right) are related in the following way (Ma et al., 2004):

$$\lambda^R \mathbf{x}^R = \mathcal{R}\lambda^L \mathbf{x}^L + \mathcal{T} \tag{11}$$

where $\mathbf{x}^L$ and $\mathbf{x}^R$ are the homogeneous coordinates in the two image planes, and λ^L and λ^R are the depth values.

In order to define the disparity, we explicitly write the projection equations for Eq. 5 (Helmholtz sequence). The relation between the 3D world coordinates $\mathbf{X} = (X, Y, Z)$ and the homogeneous image coordinates $\mathbf{x}^L = (x^L, y^L, 1)$ and $\mathbf{x}^R = (x^R, y^R, 1)$ for the toe-in technique is described by a general perspective projection model. A generic point $\mathbf{X}$ in the world coordinates is mapped onto image plane points $\mathbf{x}^L$ and $\mathbf{x}^R$ on the left and right cameras, respectively. It is worth noting that the fixation point $\mathbf{F}$ in Figure 4b is projected onto the origins of the left and right image planes, since the vergence movement makes the optical

(a) (b) (c) (d) (e)

Fig. 10. Motion fields for different camera movements: (a) the v_Z camera velocity produces an expansion pattern with the focus of expansion in the center; the superposition of a v_X velocity moves the focus of expansion on the left (b) or on the right (c) as a function of its sign; (d) the cameras move with v_Z and a rotational velocity ω_Y. (e) The color-coding scheme used for the representation: the hue represents the velocity direction, while its magnitude is represented by the saturation.

axes of the two cameras to intersect in **F**. For identical left and right focal lengths f_0, the left image coordinates are (Volpel & Theimer, 1995):

$$\begin{aligned} x^L &= f_0 \frac{X_+ \cos\alpha^L + Z \sin\alpha^L}{X_+ \sin\alpha^L \cos\beta^L - Y \sin\beta^L - Z \cos\alpha^L \cos\beta^L} \\ y^L &= f_0 \frac{X_+ \sin\alpha^L \sin\beta^L + Y \cos\beta^L - Z \cos\alpha^L \sin\beta^L}{X_+ \sin\alpha^L \cos\beta^L - Y \sin\beta^L - Z \cos\alpha^L \cos\beta^L} \end{aligned} \tag{12}$$

where $X_+ = X + b/2$. Similarly, the right image coordinates are obtained by replacing α^L, β^L and X_+ in the previous equations with α^R, β^R and $X_- = X - b/2$, respectively. We can define the horizontal disparity $d_x = x^R - x^L$ and the vertical disparity $d_y = y^R - y^L$, that establish the relationship between a world point **X** and its associated disparity vector **d**.

5.1.2 A moving camera

Considering the similarities between the stereo and motion problems, as they both look for correspondences between different frames or between left and right views, the generalizations of the two static views approach to a moving camera is in principle straightforward. Though, the description of the stereoscopically displaced cameras and of the moving camera are equivalent only if the spatial and temporal differences between frame are small enough, since the motion field is a differential concept, but not the stereo disparity. In particular, the following conditions must be satisfied: small rotations, small field of view, and v_Z small with respect to the distance of the objects from the camera. These assumptions are related to the analysis of video streams, where the camera motion is slow with respect to the frame rate (sampling frequency) of the acquisition device. Thus, we can treat the motion of the camera as continuous (Ma et al., 2004; Trucco & Verri, 1998; Adiv, 1985).

The relationship that relates the image velocity (motion field) $\dot{\mathbf{x}}$ of the image point $\mathbf{x}$ to the angular ($\boldsymbol{\omega}$) and the linear ($\mathbf{v}$) velocities of the camera and to the depth values is described by the following equation (see also Eq. 10):

$$\dot{\mathbf{x}} = \boldsymbol{\omega} \times \mathbf{x} + \frac{1}{\lambda}\mathbf{v} - \frac{\dot{\lambda}}{\lambda}\mathbf{x} \tag{13}$$

where λ and $\dot{\lambda}$ are the depth and its temporal derivative, respectively.

For planar perspective projection, i.e. $\lambda = Z$, we have that the image motion field $\dot{\mathbf{x}}$ is expressible as a function of image position $\mathbf{x} = (x, y)$ and surface depth $Z = Z(x, y)$ (i.e., the depth of the object projecting in (x, y) at current time):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_0 & 0 & -x \\ 0 & f_0 & -y \end{bmatrix} \mathbf{v} + \begin{bmatrix} -xy/f_0 & (f_0 + x^2/f_0) & -y \\ -(f_0 + y^2/f_0) & xy/f_0 & x \end{bmatrix} \boldsymbol{\omega} \tag{14}$$

To apply the relationship described by Eqs. 11 and 13 we first read the z-buffer (w) of the camera through the method added in the `SoOffScreenRenderer` class, then we obtain the depth values with respect to the reference frame of the camera in the following way:

$$\lambda = \frac{f\, n}{w(f - n) - f} \tag{15}$$

where f and n represent the values of the far and the near planes of the virtual camera, respectively.
Finally, from Eq. 11 it is possible to compute the ground truth disparity maps $\mathbf{d}$, and from Eq. 13 it is possible to obtain the ground truth motion field $\dot{\mathbf{x}}$.

6. Results for different visual tasks

The proposed VR tool can be used to simulate any interaction between the observer and the scene. In particular, in the following two different situations will be considered and analyzed:

1. Scene exploration, where both the head and the scene are fixed, and only ocular movements are considered.
2. Robotic navigation, by considering monocular vision, only.

6.1 Active vision - scene exploration

By keeping fixed the position and the orientation of the head, the described tool is active in the sense that the fixation point **F** of the stereo cameras varies to explore the scene. We can distinguish two possible scenarios: (1) to use the system to obtain sequences where the fixation points are chosen on the surfaces of the objects in the scene; (2) to use the system in cooperation with an algorithm that implements a vergence/version strategy. In the first case, it is not possible to fixate beyond or in front of the objects. In the second case, the vergence/version algorithm gives us an estimate of the fixation point, the system adapts itself looking at this point and the snapshots of the scene are then used as a new visual input for selecting a new target point.
To compute the fixation point in 3D coordinates, starting from its 2D projection, the `SoRayPickAction` class has been used. It contains the methods for setting up a ray from the near plane to the far plane, from the 2D point in the projection plane. Then the first hit, the one that corresponds to the first visible surface, is taken as the 3D coordinates, the system should fixate.
Figure 11 shows the active exploration of an indoor scene, representing a desktop and different objects at various distances, acquired by using a laser scanner. The simulator aims to mimic the behavior of a human-like robotic system acting in the peripersonal space. Accordingly, the interocular distance between the two cameras is set to 6 cm and the distance between the cameras and the center of the scene is about 80 cm. The fixation points have been chosen arbitrary, thus simulating an active exploration of the scene, and in their proximity the disparity between the left and the right projections is zero, while getting far from the

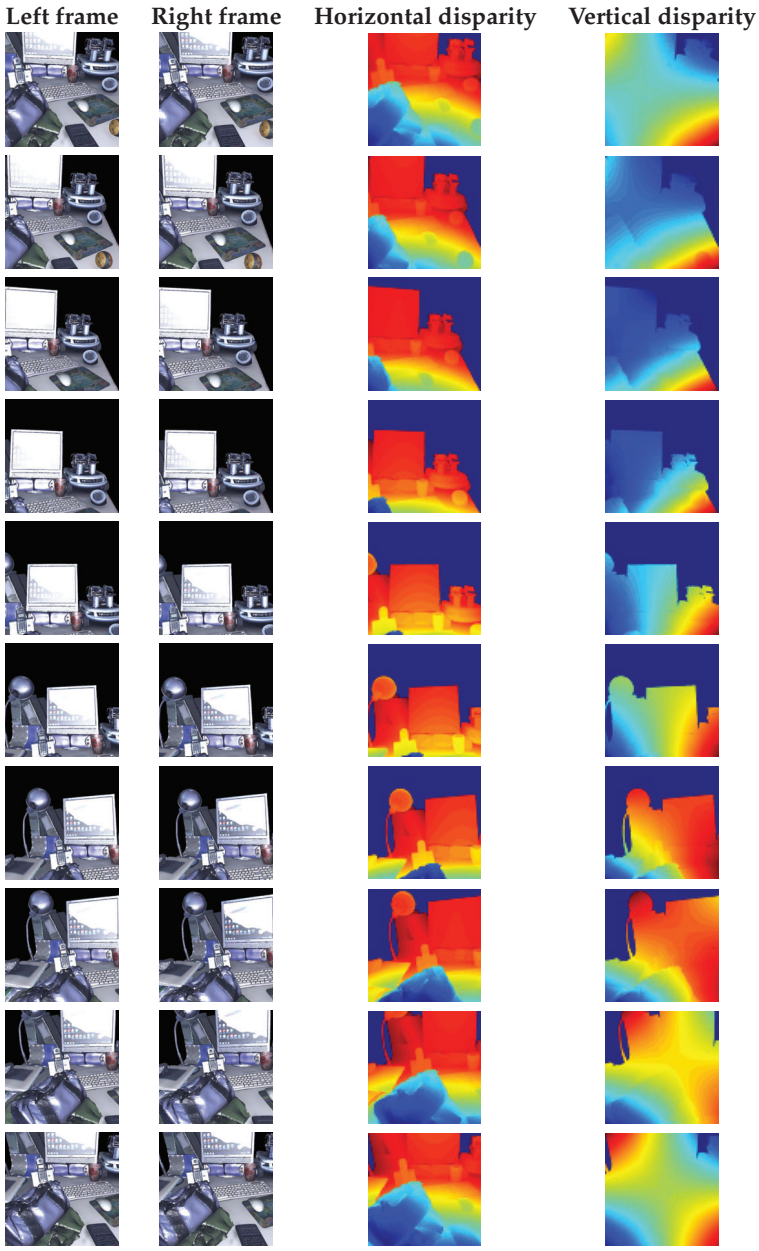


Fig. 11. Active exploration of a scene. The position and the orientation of the head is fixed, whereas the eyes are moving in order to explore the scene. The scenario mimics a typical indoor scene. The disparity values are coded from red (uncrossed disparity) to blue (crossed disparity).

fixation point both horizontal and vertical disparities emerge, as it can be seen in the ground truth data.
Instead of directly computing the 3D coordinates of the fixation points, it is possible to analyze the sequence of images and the corresponding disparity maps, while performing vergence movements. In Figure 12 (left) it is possible to see the red-cyan anaglyph of the stereo pairs, before having reached the fixation point (upper part of the figure) and when fixation is achieved onto a target (bottom). The plots on the right show the variation of the disparity in the center of the image at each time step (upper part) and the variation of the actual depth of the fixation point with respect to the desired value (bottom).

6.2 Robotic navigation

In this Section, the simulator is used to obtain sequences acquired by a moving observer. The position and the orientation of the head can be changed, in order to mimic the navigation in the virtual environment. For the sake of simplicity, the ocular movements are not considered and the visual axes are kept parallel. It is worth noting that the eye movements necessary to actively explore the scene, considered in the previous Section, could be embedded if necessary. Figure 13 shows the sequence of images and the related ground truth optic flow fields for the different movements of the observer.

7. Conclusion

In conclusion, a tool that uses the VR to simulate the actual projections impinging the cameras of an active visual system rather than to render the 3D visual information for a stereoscopic display, has been developed. The simulator works on the 3D data that can

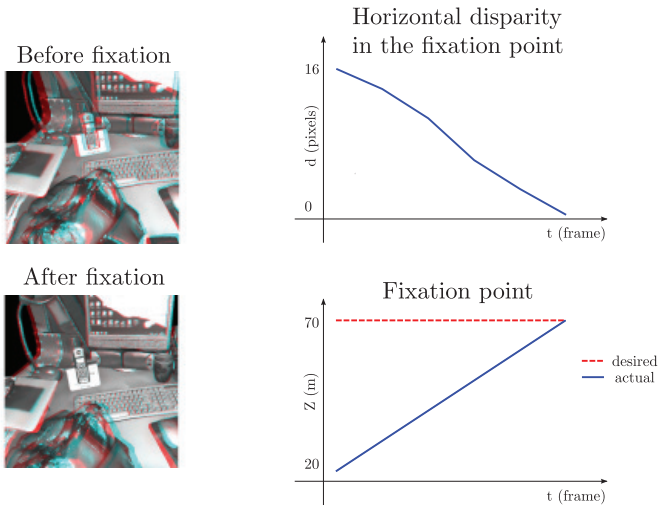


Fig. 12. (left) The anaglyph images before and after a vergent movement of the cameras in order to correctly fuse the left and right images of the target object. (right) The plots show the variation of the disparity in the center of the stereo images and the related depth of actual and desired fixation point. It is worth noting that the vergence movements are synthetically generated (i.e., not driven by the visual information).

be synthetically generated or acquired by a laser scanner and performs both cameras and robot movements following the strategies adopted by different active stereo vision systems, including bio-mimetic ones.

The virtual reality tool is capable of generating pairs of stereo images like the ones that can be obtained by a verging pan-tilt robotic head and the related ground truth data, disparity maps and motion field. To obtain such a behavior the toe-in stereoscopic technique is preferred to the off-axis technique. By proper roto-translations of the view volumes, we can create benchmark stereo sequences for testing vision algorithms under convergent-camera conditions. In more general terms, by exploiting the full knowledge of the 3D structure of the scene the proposed VR tool can be used to model active vision systems interacting with the scene. A data set of stereo image pairs and the related ground truth disparities and motion fields are available for the Robotics and Computer Vision community at the web site `www.pspc.dibe.unige.it/Research/vr.html`.

Although the main purpose of this work is to obtain sufficiently complex scenarios for benchmarking an active vision system, complex photo-realistic scenes can be easily obtained by using the 3D data and textures acquired by laser scanners, which capture detailed, highly accurate, and full color objects to build 3D virtual models at an affordable computational cost. In this way improving the photo-realistic quality of the 3D scene does not endanger the

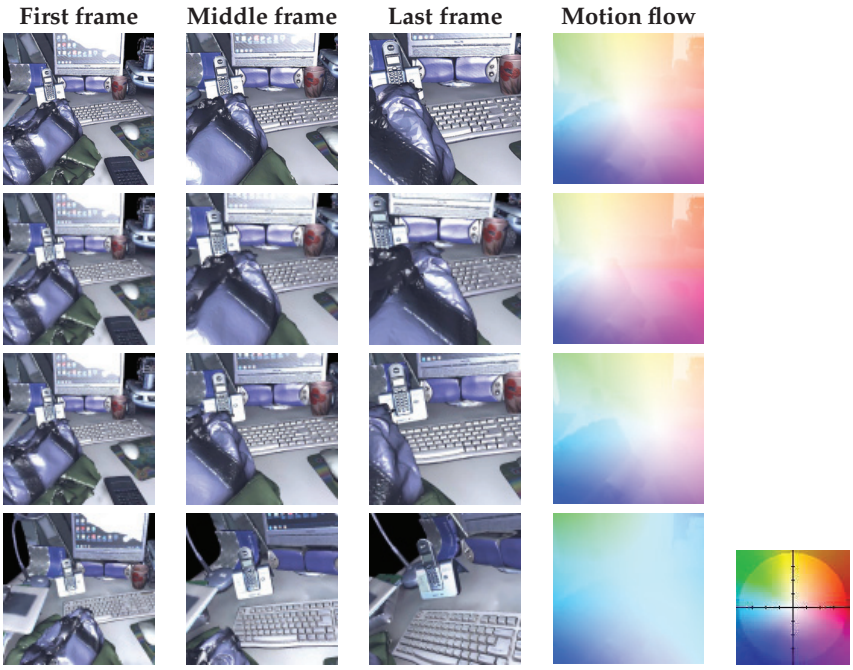


Fig. 13. Robotic navigation in an indoor scenario. Different situation are taken into account. First row: the robot has v_Z velocity, only. Thus the focus of expansion is in the center. Second and third row: positive and negative v_X are introduced, thus the focus of expansion move to the left and to the right, respectively. Fourth row: a rotation around the Y axis is combined with a translation aking v_Z. Same color-coding scheme of Fig.10.

definition of a realistic model of the interactions between the vision system and the observed scene. As part of a future work, we plan to modify the standard pan-tilt behaviour by including more biologically plausible constraints on the camera movements (Schreiber et al., 2001; Van Rijn & Van den Berg, 1993) and to integrate vergence/version strategies in the system in order to have a fully active tool that interacts with the virtual environments.

8. Acknowledgements

We wish to thank Luca Spallarossa for the helpful comments, and Andrea Canessa and Agostino Gibaldi for the acquisition, registration and post-processing of the 3D data.
This work has been partially supported by EU Projects FP7-ICT 217077 "EYESHOTS" and FP7-ICT 215866 "SEARISE".

9. References

Adelson, E. & Bergen, J. (1991). The plenoptic and the elements of early vision, *in* M. Landy & J. Movshon (eds), *Computational Models of Visual Processing*, MIT Press, pp. 3–20.

Adiv, G. (1985). Determining three-dimensional motion and structure from optical flow generated by several moving objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7: 384–401.

Awaad, I., Hartanto, R., León, B. & Plöger, P. (2008). A software system for robotic learning by experimentation, *Workshop on robot simulators (IROS08)*.

Baker, S., Scharstein, D., Lewis, J., Michael, S., Black, J. & Szeliski, R. (2007). A database and evaluation methodology for optical flow, *ICCV*, pp. 1–8.

Bernardino, A., Santos-Victor, J., Ferre, M. & Sanchez-Urn, M. (2007). Stereoscopic image visualization for telerobotics. experiments with active binocular cameras, *Advances in Telerobotics*, pp. 77–90.

Bourke, P. & Morse, P. (2007). Stereoscopy: Theory and practice, *Workshop at 13th International Conference on Virtual Systems and Multimedia* .

Cannata, G. & Maggiali, M. (2008). Models for the design of bioinspired robot eyes, *IEEE Transactions on Robotics* 24: 27–44.

Chessa, M., Sabatini, S. & Solari, F. (2009). A fast joint bioinspired algorithm for optic flow and two-dimensional disparity estimation, *in* M. Fritz, B. Schiele & J. Piater (eds), *Computer Vision Systems*, Vol. 5815 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 184–193.

Davis, J. & Chen, X. (2003). Calibrating pan-tilt cameras in wide-area surveillance networks, *Proc. of IEEE International Conference on Computer Vision*, IEEE Computer Society, pp. 144–150.

EYESHOTS (2008). *FP7-ICT 217077 "EYESHOTS"*, `http://www.eyeshots.it/`.

Ferre, P., Aracil, R. & Sanchez-Uran, M. (2008). Stereoscopic human interfaces, *IEEE Robotics & Automation Magazine* 15(4): 50–57.

Fleet, D. & Jepson, A. (1990). Computation of component image velocity from local phase information, *International Journal of Computer Vision* 5(1): 77–104.

Forsyth, D. & Ponce, J. (2002). *Computer Vision: A Modern Approach*, Prentice Hall.

Gibaldi, A., Chessa, M., Canessa, A., Sabatini, S. & Solari, F. (2010). A cortical model for binocular vergence control without explicit calculation of disparity, *Neurocomputing* 73(7-9): 1065 – 1073.

Grinberg, V., Podnar, G. & Siegel, M. (1994). Geometry of binocular imaging, *Proc. of the*

IS&T/SPIE Symp. on Electronic Imaging, Stereoscopic Displays and applications, Vol. 2177, pp. 56–65.

Haslwanter, T. (1995). Mathematics of three-dimensional eye rotations, *Vision Research* 35(12): 1727 – 1739.

Heeger, D. (1987). Model for the extraction of image flow, *Journal of the Optical Society of America A* 4(1): 1445–1471.

Horaud, R., Knossow, D. & Michaelis, M. (2006). Camera cooperation for achieving visual attention, *Machine Vision and Applications* 16: 331–342.

Hung, G., Semmlow, J. & Ciufferda, K. (1986). A dual-mode dynamic model of the vergence eye movement system, *Biomedical Engineering, IEEE Transactions on* BME-33(11): 1021–1028.

Jain, A., Kopell, D., Kakligian, K. & Wang, Y. (2006). Using stationarydynamic camera assemblies for wide-area video surveillance and selective attention, *In IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 537–544.

Jørgensen, J. & Petersen, H. (2008). Usage of simulations to plan stable grasping of unknown objects with a 3-fingered schunk hand, *Workshop on robot simulators (IROS08)*.

Kooi, F. & Toet, A. (2004). Visual comfort of binocular and 3d displays, *Displays* 25(2-3): 99–108.

Liu, Y., Bovik, A. & Cormack, L. (2008). Disparity statistics in natural scenes, *Journal of Vision* 8(11): 1–14.

Longuet-Higgins, H. & Prazdny, K. (1980). The interpretation of a moving retinal image, *Phil. Trans. R. Soc. Lond. B* 208: 385–397.

Ma, Y., Soatto, S., Kosecka, J. & Sastry, S. (2004). *An Invitation to 3D Vision. From Images to Geometric Models*, Springer-Verlag.

McCane, B., Novins, K., Crannitch, D. & Galvin, B. (2001). On benchmarking optical flow, *Computer Vision and Image Understanding* 84(1): 126–143.

Michel, O. (2004). Webots: Professional mobile robot simulation, *International Journal of Advanced Robotic Systems* 1(1): 39–42.

Nakamura, Y., Matsuura, T., Satoh, K. & Ohta, Y. (1996). Occlusion detectable stereo-occlusion patterns in camera matrix, *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pp. 371 –378.

Okada, K., Kino, Y. & Kanehiro, F. (2002). Rapid development system for humanoid vision-based behaviors with real-virtual common interface, *IEEE/RSJ IROS*.

Otte, M. & Nagel, H. (1995). Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences, *Artif. Intell.* 78(1-2): 5–43.

Scharstein, D. & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, *International Journal of Computer Vision* 47(1/2/3): 7–42.

Scharstein, D. & Szeliski, R. (2003). High-accuracy stereo depth maps using structured light, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 195–202.

Schreiber, K. M., Crawford, J. D., Fetter, M. & Tweed, D. B. (2001). The motor side of depth vision, *Nature* 410: 819–822.

Southard, D. (1992). Transformations for stereoscopic visual simulation, *Computers & Graphics* 16(4): 401–410.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L. & Nori, F. (2008). An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator, *Workshop on Performance Metrics for Intelligent Systems*

Workshop.

Trucco, E. & Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*, Prentice Hall.

Ulusoy, I., Halici, U. & Leblebicioglu, K. (2004). 3D cognitive map construction by active stereo vision in a virtual world, *Lecture notes in Computer Science* 3280: 400–409.

Van Rijn, L. & Van den Berg, A. (1993). Binocular eye orientation during fixations: Listing's law extended to include eye vergence, *Vision Research* 33: 691–708.

Volpel, B. & Theimer, W. (1995). Localization uncertainty in area-based stereo algorithms, *IEEE Transactions on Systems, Man and Cybernetics* 25(12): 1628–1634.

Wann, J. P., Rushton, S. & Mon-Williams, M. (1995). Natural problems for stereoscopic depth perception in virtual environments., *Vision research* 35(19): 2731–2736.

Yamanoue, H. (2006). The differences between toed-in camera configurations and parallel camera configurations in shooting stereoscopic images, *Multimedia and Expo, IEEE International Conference on* pp. 1701–1704.

Yang, Z. & Purves, D. (2003). Image/source statistics of surfaces in natural scenes, *Network: Comput. Neural Syst.* 14: 371–390.