Omnidirectional Photometric Visual Servoing

Guillaume Caron, Eric Marchand, E. Mouaddib

Omnidirectional Photometric Visual Servoing

Guillaume Caron, Eric Marchand and El Mustapha Mouaddib

***Abstract*— Visual servoing has been based on geometric features for a long time. Recent works have highlighted the interest of taking into account the photometric information of the entire image. This approach was tackled with images of perspective cameras. We propose, in this paper, to adapt this technique to central cameras. This generalization allows to apply this kind of method to wide field of view cameras. We also propose to adapt gradient computation to take into account distorsions of such cameras. Several experiments have been successfully done with a fisheye camera.**

I. INTRODUCTION

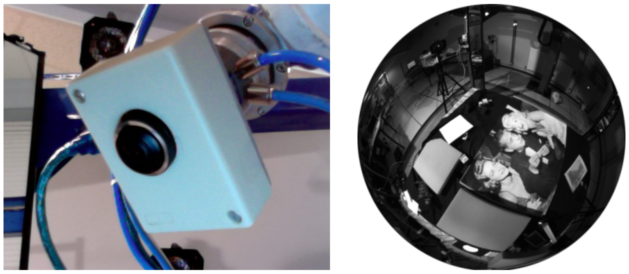
Moving a robot to a precise position can be done using vision. In this field, visual servoing [1] is a technique that aims to control the robot motion to a desired position only defined by an image. Usually, geometric features are detected in the reference image and visual servoing moves the camera to the reference position minimizing the error between reference features and their correspondences detected in the current image.

Geometric features (points, lines, circles, moments) have been widely used for visual servoing. These features and their different representations can bring interesting properties for the servoing process, such as, when well chosen, a nice decoupling between the camera degrees of freedom [2]. However, detecting, tracking and matching features effectively is still a hard problem [3]. So an interesting idea is to directly use the image as a whole rather than extract features. It withdraws the detection and matching problems and, moreover, brings a lot more information.

This kind of idea was developped by Nayar *et al.* [4] and then by Deguchi [5]. In these works, to reduce the dimensionality of image data, an eigen decomposition is done. The control is then performed in the eigenspace and the interaction matrix related to the eigenspace is learnt offline.

Collewet and Marchand [6] proposed to use directly photometric features, *i.e.* pixel intensities of the entire image, as input of the control scheme. Benhimane *et al.* [7] used also a direct intensity based visual servoing approach. Despite the fact that, as in our case, image intensity is used as the basis of the approach, an important image processing step is necessary to estimate the homography used to build the control law.

This paper adresses the problem of visual servoing for central omnidirectional cameras using photometric features in the whole image. We propose to extend the work of [6] investigating different representations and modeling issues directly induced by omnidirectional vision. A spherical projection model for central cameras is used and we propose to formulate the omnidirectional photometric visual servoing on the sphere of this model. We also considered the normalized plane representation for a fair comparison process.



(a) The used fisheye camera for the visual servoing.

(b) An omnidirectional image from a fisheye camera.

Fig. 1. Vision sensor used for omnidirectional visual servoing experiments.

The motivation of using omnidirectional vision for photometric visual servoing is to maximize the chances of sensing textured zones of the environment, the work of [6] being particularly efficient with textured images.

This paper is organized as follows. First, the camera model is presented and then omnidirectional visual servoing is recalled for points. After that the photometric visual servoing for omnidirectional camera is developped. To finish, visual servoing results obtained on a Gantry robot are presented and behaviours between the various proposed control law are compared.

II. CAMERA MODEL

Barreto *et al.* [8] proposed a unified projection model for central cameras. This model describes a family of cameras from perspective to catadioptric ones with particular shape mirrors. Furthermore, Ying *et. al.* [9] show this model can be used for fisheye cameras.

According to this model, the world is first projected on a sphere with coordinates $\begin{pmatrix}0 & 0 & \xi\end{pmatrix}^{\top}$ in the camera frame followed by a perspective projection on the image plane. Such a model can be defined using parameter ξ which depends intrinsically on the omnidirectional camera type.

Knowing intrinsic parameters $\gamma = \{p_x, p_y, u_0, v_0, \xi\}$, a 3D point $\mathbf{X} = \begin{pmatrix}X & Y & Z\end{pmatrix}^{\top}$ is first projected on a unitary

Guillaume Caron and El Mustapha Mouaddib are with MIS laboratory, University of Picardie Jules Verne, Amiens, FRANCE; e-mail: {guillaume.caron, mouaddib}@u-picardie.fr

Eric Marchand is with Université de Rennes 1, IRISA, INRIA Lagadic, Rennes, France; e-mail: eric.marchand@irisa.fr

sphere and then in the image plane as $\mathbf{x} = \begin{pmatrix} x & y & 1 \end{pmatrix}^\mathsf{T}$. The relationship between $\mathbf{X}$ and $\mathbf{x}$ can be expressed as:

$$\mathbf{x} = pr_\xi(\mathbf{X}) \quad \text{with} \quad \begin{cases} x = \frac{X}{Z+\xi\rho} \\ y = \frac{Y}{Z+\xi\rho} \end{cases} \tag{1}$$

where $\rho = \sqrt{X^2+Y^2+Z^2}$. $\mathbf{x}$ is the point on the virtual normalized plane and the image point in pixelic coordinates is obtained by:

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K}\mathbf{x}. \tag{2}$$

Since this paper deals with methods using data lying on the equivalent sphere, let us define the spherical projection function of a 3D point:

$$\mathbf{X}_\mathcal{S} = pr_\mathcal{S}(\mathbf{X}) \quad \text{with} \quad \begin{cases} X_\mathcal{S} = \frac{X}{\rho} \\ Y_\mathcal{S} = \frac{Y}{\rho} \\ Z_\mathcal{S} = \frac{Y}{\rho} \end{cases} \tag{3}$$

where $\mathbf{X}_\mathcal{S} = \begin{pmatrix} X_\mathcal{S} & Y_\mathcal{S} & Z_\mathcal{S} \end{pmatrix}^\mathsf{T}$. This parameterization is redundant on a sphere since a coordinate can be written as a linear combination of the two other. Hence, two angular coordinates (azimuth and elevation) are sufficient to represent a point on a sphere: $\mathbf{S} = \begin{pmatrix} \phi & \theta \end{pmatrix}^\mathsf{T}$ with $\phi = \arccos(Z_\mathcal{S})$ and $\theta = \arctan(Y_\mathcal{S} \ / \ X_\mathcal{S})$.

Eventually, the inverse projection function pr_ξ^{-1} allows to retrieve the point on the sphere corresponding to $\mathbf{x}$:

$$\mathbf{X}_\mathcal{S} = pr_\xi^{-1}(\mathbf{x}) = \begin{pmatrix} \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} x \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} y \\ \frac{\xi+\sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} - \xi \end{pmatrix}. \tag{4}$$

III. OMNIDIRECTIONAL VISUAL SERVOING

To control the degrees of freedom of a robot using visual servoing, image points can be used as features. Then the design of the control law aims to minimize the error between the current visual features $\mathbf{s}(\mathbf{r})$ observed from the pose $\mathbf{r}$ and their desired position in the image $\mathbf{s}^*$. The interaction matrix $\mathbf{L_s}$, linking the motion of features $\dot{\mathbf{s}}$ to the camera velocity $\mathbf{v}$, has also to be defined. This link is formally expressed as:

$$\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v} \tag{5}$$

where $\mathbf{v} = \begin{pmatrix} v & \omega \end{pmatrix}^\mathsf{T}$ is respectively the linear and angular camera velocity. Eventually, a control law is designed to try to have an exponential decoupled decrease of the error $\mathbf{s}-\mathbf{s}^*$:

$$\mathbf{v} = -\lambda \widehat{\mathbf{L}}_\mathbf{s}^+ (\mathbf{s} - \mathbf{s}^*) \tag{6}$$

with λ a tunable gain to modify the convergence rate and $\widehat{\mathbf{L}}_\mathbf{s}^+$, the pseudo-inverse of a model of $\mathbf{L_s}$.

A. Image Plane Visual Servoing

To define the visual servoing of an omnidirectional camera using points $\mathbf{x} = (x, y)$ in the image plane as visual feature, we have to express the interaction matrix $\mathbf{L_x}$, linking the feature point $\mathbf{x}$ motion to the camera velocity $\mathbf{v}$. For feature $\mathbf{s} \ = \mathbf{x}$, the associated interaction matrix is the product of two Jacobians [11]:

$$\mathbf{L_s} = \frac{\partial \mathbf{s}}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \mathbf{r}} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{r}} \tag{7}$$

where $\mathbf{P}$ is the real world feature. The second jacobian is well known for points [11] and does not depend on the projection model which is encapsulated in the first jacobian.

Hence, for image plane VS (IP-VS), the partial derivatives of $\mathbf{x}$ w.r.t. $\mathbf{X}$ are computed from eq. (1) and lead to the interaction matrix for a point in the omnidirectional image plane [12].

B. Spherical Visual Servoing

Spherical visual servoing has a very similar formulation than image plane based one. However, the representation of features and the interaction matrix are different.

1) Cartesian Spherical Visual Servoing: This parameterization (CS-VS) of visual servoing uses 3-coordinates points onto the equivalent sphere ($\mathbf{s} = \mathbf{X}_\mathcal{S}$, from eq. (3)) and the interaction matrix $\mathbf{L}_{\mathbf{X}_\mathcal{S}}$ is defined by [13]:

$$\mathbf{L}_{\mathbf{X}_\mathcal{S}} = \frac{\partial \mathbf{X}_\mathcal{S}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{r}} = \begin{pmatrix} \frac{1}{\rho}(\mathbf{X}_\mathcal{S}\mathbf{X}_\mathcal{S}^\mathsf{T} - \mathbf{I}_3) & [\mathbf{X}_\mathcal{S}]_\times \end{pmatrix}. \tag{8}$$

2) Pure Spherical Visual Servoing: As explained in a previous section, only two coordinates $\mathbf{S} = \begin{pmatrix} \phi & \theta \end{pmatrix}^\mathsf{T}$ are enough to express a point on a sphere. The interaction matrix associated to feature $\mathbf{s} = \mathbf{S}$ is then different from the cartesian representation [14]:

$$\mathbf{L_S} = \frac{\partial \mathbf{S}}{\partial \mathbf{X}_\mathcal{S}} \mathbf{L}_{\mathbf{X}_\mathcal{S}} = \begin{pmatrix} -\frac{\mathrm{c}\theta\mathrm{c}\phi}{\rho} & -\frac{\mathrm{s}\theta\mathrm{c}\phi}{\rho} & \frac{\mathrm{s}\phi}{\rho} & \mathrm{s}\theta & -\mathrm{c}\theta & 0 \\ \frac{\mathrm{s}\theta}{\rho\mathrm{s}\phi} & -\frac{\mathrm{c}\theta}{\rho\mathrm{s}\phi} & 0 & \frac{\mathrm{c}\theta\mathrm{c}\phi}{\mathrm{s}\phi} & \frac{\mathrm{s}\theta\mathrm{c}\phi}{\mathrm{s}\phi} & -1 \end{pmatrix} \tag{9}$$

with $\mathrm{c}\theta = \cos\theta$, $\mathrm{c}\phi = \cos\phi$, $\mathrm{s}\theta = \sin\theta$, $\mathrm{s}\phi = \sin\phi$. This pure spherical formulation of visual servoing (PS-VS) is minimal but has a singularity when $\sin\ \phi = 0$, *i.e.* when $x = y = 0$ in the normalized image plane.

IV. OMNIDIRECTIONAL PHOTOMETRIC VISUAL SERVOING

Considering visual servoing as an optimization problem, minimizing the difference between current and desired images, $||\mathbf{I}(\mathbf{r}) - \mathbf{I}(\mathbf{r}^*)||$, [6] formulates photometric visual servoing control law using a Levenberg-Marquart technique. It has been shown that it ensures better convergence than other kind of laws. So, instead of using a Gauss-Newton control law (eq. (6)), this control law is also used in the current work:

$$\mathbf{v} = -\lambda \left(\mathbf{H} + \mu \operatorname{diag}(\mathbf{H})\right)^{-1} \mathbf{L}_\mathbf{I}^\mathsf{T} \left(\mathbf{I}(\mathbf{r}) - \mathbf{I}(\mathbf{r}^*)\right) \tag{10}$$

with $\mathbf{H} = \mathbf{L}_\mathbf{I}^\mathsf{T}\mathbf{L}_\mathbf{I}$, considering $\mathbf{L}_\mathbf{I}$ is the interaction matrix related to luminance of image $\mathbf{I}$. If μ is very high this control law behaves like a steepest descent whereas a very low value for μ leads eq. (10) to behave like eq. (6).

A. Image Plane Visual Servoing

From this point, the considered visual feature is the luminance I of each image point. So the feature vector $\mathbf{s}(\mathbf{r})$, becomes:

$$\mathbf{s}(\mathbf{r}) = \mathbf{I}(\mathbf{r}) = (\mathbf{I}_{1\bullet}, \mathbf{I}_{2\bullet}, \dots, \mathbf{I}_{N\bullet}) \quad (11)$$

with $\mathbf{I}_{i\bullet}$, the i-th line of the image.

1) Interaction Matrix: The interaction matrix $\mathbf{L}_I(\mathbf{x})$ formulation is defined under temporal luminance consistancy [6]:

$$I(\mathbf{x} + \mathbf{dx}, t + dt) = I(\mathbf{x}, t) \quad (12)$$

assuming $\mathbf{dx}$ is small. If it is small enough, the optical flow constraint equation (OFCE) is valid:

$$\nabla I^{\mathsf{T}} \dot{\mathbf{x}} + I_t = 0 \quad (13)$$

with ∇I the spatial gradient of $I(\mathbf{x}, t)$ and $I_t = \frac{\partial I(\mathbf{x},t)}{\partial t}$. For the omnidirectional image plane visual servoing, the knowledge of interaction matrix $\mathbf{L}_\mathbf{x}$ (eq. (7)) leads to:

$$I_t = -\nabla I^{\mathsf{T}} \mathbf{L}_\mathbf{x} \mathbf{v}. \quad (14)$$

And similarly to [6], we get the interaction matrix $\mathbf{L}_I(\mathbf{x})$ related to I at pixel $\mathbf{x}$:

$$\mathbf{L}_I(\mathbf{x}) = -\nabla I^{\mathsf{T}} \mathbf{L}_\mathbf{x}. \quad (15)$$

2) Gradient Computation: In a perspective image, the image gradients ∇I are computed using the same neighborhood for all image point: a square regularly sampled. In omnidirectional images, the image geometry is different, *i.e.* resolution and orientation are not constant. Hence, the neighborhood used for the gradient computation has to be adapted.

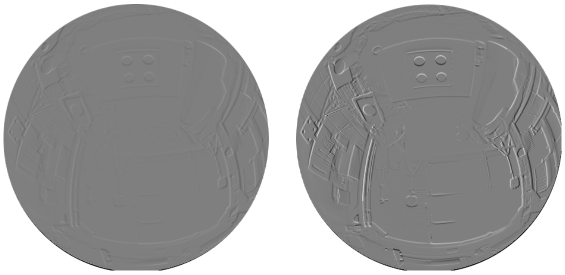
Assuming the neighborhood around $\mathbf{u}_0 = \begin{pmatrix} u_0 & v_0 & 1 \end{pmatrix}^{\mathsf{T}}$ is square and regularly sampled, as the image, the idea, to find the neighborhood around pixel $\mathbf{u} = \begin{pmatrix} u & v & 1 \end{pmatrix}^{\mathsf{T}}$, is to backproject this neighborhood on a pole of the sphere. Then move it so that it is around the spherical point $\mathbf{X}_\mathcal{S} = pr_\xi^{-1}(\mathbf{K}^{-1}\mathbf{u})$. Considering $\mathbf{X}_{\mathcal{S}0}$, the backprojection of $\mathbf{u}_0$, this motion is defined by the rotation around the axis $[\mathbf{X}_{\mathcal{S}0} \times \mathbf{X}_\mathcal{S}]$ and of angle $\arccos(\mathbf{X}_{\mathcal{S}0}^{\mathsf{T}} \mathbf{X}_\mathcal{S})$.

Then, the neighborhood is projected in the image plane and to find pixel values, interpolations can be done. For computational efficiency, we used the nearest neighbour interpolation to find intensities. Moreover, using this interpolation type, adapted neighborhoods can be computed offline so that the gradient computation process is as fast as with the standard neighborhood.

B. Cartesian Spherical Visual Servoing

Working on the equivalent sphere implies working with a spherical image $I_\mathcal{S}$, which is the inverse projection of the image I onto the sphere $\mathcal{S}$. The OFCE becomes now:

$$\frac{\partial I_\mathcal{S}}{\partial X_\mathcal{S}} \dot{X}_\mathcal{S} + \frac{\partial I_\mathcal{S}}{\partial Y_\mathcal{S}} \dot{Y}_\mathcal{S} + \frac{\partial I_\mathcal{S}}{\partial Z_\mathcal{S}} \dot{Z}_\mathcal{S} + I_t = 0. \quad (16)$$



(a) $\frac{\partial I_\mathcal{S}}{\partial Z_\mathcal{S}}$ from elevation of planar image gradients

(b) $\frac{\partial I_\mathcal{S}}{\partial Z_\mathcal{S}}$ directly computed on sphere

Fig. 2. Adapted gradients computation. (a) and (b) compare two ways of computing gradients and show different results. Spherical gradient computation bring more information than elevation of image gradients.

And the interaction matrix $\mathbf{L}_{I_\mathcal{S}}(\mathbf{X}_\mathcal{S})$ related to $I_\mathcal{S}$ at spherical position $\mathbf{X}_\mathcal{S}$ is defined by:

$$\mathbf{L}_{I_\mathcal{S}}(\mathbf{X}_\mathcal{S}) = -\nabla I_\mathcal{S}^{\mathsf{T}} \mathbf{L}_{\mathbf{X}_\mathcal{S}}. \quad (17)$$

$\mathbf{L}_{\mathbf{X}_\mathcal{S}}$ is given by equation (8) but the image gradient on the sphere have yet to be expressed.

In a first approach, the gradient, using the adapted neighborhood presented in the section IV.A.2, is computed in the image plane and then is back projected onto the equivalent sphere deriving equation (4):

$$\begin{cases} \frac{\partial I_\mathcal{S}}{\partial X_\mathcal{S}} = \frac{\partial I}{\partial u} \frac{\partial u}{\partial X_\mathcal{S}} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial X_\mathcal{S}} \\ \frac{\partial I_\mathcal{S}}{\partial Y_\mathcal{S}} = \frac{\partial I}{\partial u} \frac{\partial u}{\partial Y_\mathcal{S}} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial Y_\mathcal{S}} \\ \frac{\partial I_\mathcal{S}}{\partial Z_\mathcal{S}} = \frac{\partial I}{\partial u} \frac{\partial u}{\partial Z_\mathcal{S}} + \frac{\partial I}{\partial v} \frac{\partial v}{\partial Z_\mathcal{S}} \end{cases}. \quad (18)$$

One can note that this technique is not actually efficient. Indeed, to compute the shape of the adapted neighborhood, there are already two projections between image plane and equivalent sphere. So computing gradients in image plane and backprojecting them on the sphere adds one more projection. Computing gradients directly on the equivalent sphere is more efficient (Fig. 2(a) and 2(b)).

To compute gradients $(\frac{\partial I_\mathcal{S}}{\partial X_\mathcal{S}}, \frac{\partial I_\mathcal{S}}{\partial Y_\mathcal{S}}, \frac{\partial I_\mathcal{S}}{\partial Z_\mathcal{S}})$ directly on the sphere, we propose to define a sampling step:

$$\Delta_{X_\mathcal{S}} = \Delta_{Y_\mathcal{S}} = \Delta_{Z_\mathcal{S}} = \left\| pr_\xi^{-1} \left(\begin{bmatrix} u_0 + 1 \\ v_0 \\ 1 \end{bmatrix} \right) - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\|. \quad (19)$$

From a point $\begin{pmatrix} X_\mathcal{S} & Y_\mathcal{S} & Z_\mathcal{S} \end{pmatrix}^{\mathsf{T}}$, three N-neighborhoods are defined, one for each axis. N defines the number of neighbours around a point. The spherical neighborhood for the first coordinate, $X_{\mathcal{S}\mathcal{N}}$, is defined by:

$$X_{\mathcal{S}\mathcal{N}} = \left\{ \begin{array}{r} \frac{\begin{pmatrix} X_\mathcal{S} + k\Delta_{X_\mathcal{S}} & Y_\mathcal{S} & Z_\mathcal{S} \end{pmatrix}^{\mathsf{T}}}{\left\| \begin{pmatrix} X_\mathcal{S} + k\Delta_{X_\mathcal{S}} & Y_\mathcal{S} & Z_\mathcal{S} \end{pmatrix}^{\mathsf{T}} \right\|}, \\ -\frac{N}{2} \leq k \leq \frac{N}{2}, k \neq 0 \end{array} \right\}. \quad (20)$$

The procedure is similar for $Y_{\mathcal{S}\mathcal{N}}$ and $Z_{\mathcal{S}\mathcal{N}}$. Once spherical neighborhoods are created, they are projected in the image plane to recover intensities using interpolation for non integer coordinates. For the same reasons as adapted image plane gradient computation, nearest neighbour interpolation is used to precompute neighborhoods offline.

C. Pure Spherical Visual Servoing

Considering the azimuth and elevation parameterization of a spherical point, Daniilidis *et al.* introduced the pure spherical OFCE [10]:

$$\frac{1}{\sin\phi}\frac{\partial I_{\mathcal{S}}}{\partial\theta}\dot{\mathbf{p}}_\theta + \frac{\partial I_{\mathcal{S}}}{\partial\phi}\dot{\mathbf{p}}_\phi + I_t = 0 \qquad (21)$$

where $\begin{pmatrix}\dot{\mathbf{p}}_\theta & \dot{\mathbf{p}}_\phi & 0\end{pmatrix}^\mathsf{T}$ is the optical flow vector tangent to the sphere at point $\begin{pmatrix}\theta & \phi\end{pmatrix}^\mathsf{T}$, with $\mathbf{p} = \begin{pmatrix}\sin\phi\cos\theta & \sin\phi\sin\theta & \cos\phi\end{pmatrix}^\mathsf{T}$. This constraint equation is used in order to develop the interaction matrix $\mathbf{L}_{I_S}(\mathbf{S})$, knowing $\mathbf{L}_\mathbf{S}$ (eq. 9):

$$\mathbf{L}_{I_S}(\mathbf{S}) = -\nabla I_{\mathcal{S}}^\mathsf{T}\mathbf{L}_\mathbf{S}. \qquad (22)$$

A way to compute pure spherical gradients is to compute image gradients and transfer them onto the sphere using [10]:

$$\begin{cases} \frac{\partial I_{\mathcal{S}}}{\partial\theta} = \frac{\partial I}{\partial u}\frac{\partial u}{\partial\theta} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial\theta} \\ \frac{\partial I_{\mathcal{S}}}{\partial\phi} = \frac{\partial I}{\partial u}\frac{\partial u}{\partial\phi} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial\phi} \end{cases}. \qquad (23)$$

However, Demonceaux *et al.* [15] show it is possible to compute pure spherical gradients directly on $I_{\mathcal{S}}$. This is more efficient than elevating image gradients on the equivalent sphere, as in the cartesian spherical case. Furthermore, authors mention that omnidirectional image filtering is only valid on the equivalent sphere where convolution is commutative with regular sampling. The technique is to define neighborhoods adapted to spherical geometry in order to compute gradients similarly than for perspective images.

Demonceaux *et al.* used a geodesic metric to formulate their neighborhoods but in the current case, this is an angular metric which have to be used in order to satisfy equation (21). So considering $\Delta_{\phi,\theta} = \arccos(|\begin{pmatrix}0 & 0 & 1\end{pmatrix}.pr_\xi^{-1}(\begin{bmatrix}u_0+1 & v_0 & 1\end{bmatrix}^\mathsf{T})|)$, the two N-neighborhoods $\phi_\mathcal{N}$ and $\theta_\mathcal{N}$ are expressed as:

$$\begin{cases} \phi_\mathcal{N} = \left\{\begin{pmatrix}\theta & \phi + k\Delta_\phi\end{pmatrix}^\mathsf{T}, -\frac{N}{2} \leq k \leq \frac{N}{2}, k \neq 0\right\} \\ \theta_\mathcal{N} = \left\{\begin{pmatrix}\theta + k\Delta_\theta & \phi\end{pmatrix}^\mathsf{T}, -\frac{N}{2} \leq k \leq \frac{N}{2}, k \neq 0\right\} \end{cases}. \qquad (24)$$

Finally, neighborhoods are projected on the image plane to retrieve intensites. Nearest neighbour interpolation is done to make to process fast and to allow precomputation of spherical neighbours projected in the image for each pixel.

V. RESULTS

Experiments are done on a Gantry robot (Fig. 5(b)). The fisheye camera (Fig. 1) is mounted on the robot end-effector. Programming was done using the ViSP [16] library. In this section, three kind of experiments are presented. When converging, positioning error is less than one tenth of a millimeter in translation and 0.02^o in rotation.

The goal of the first experiment is to show the achievement of omnidirectional photometric visual servoing, using the representations presented in this paper and to compare them. The second experiment shows photometric visual servoing using IP-VS, CS-VS and PS-VS with a unique desired position and three different initial positions and orientations. The last experiment is a servoing using CS-VS starting from an initial position that is far from the desired one.

For all experiments, the interaction matrix is computed only at desired position. Furthermore, Z is an unknown parameter and is supposed constant for all pixel all along the motion of the camera. The achievement of experiments shows the method is robust to coarse estimation of Z.

A. First Experiment

For this experiment, the camera translation between the two positions is $\Delta_X = 3.53cm$, $\Delta_Y = -7.82cm$, $\Delta_Z = 17.08cm$, in the robot frame, with no orientation variation. This experiment emphasizes, qualitatively as well as quantitatively speaking, the best approach in term of camera trajectory distance w.r.t. the straigth line linking the start and final positions and in term of interaction matrix conditioning. Figure 3 shows desired and initial images as well as their difference. Camera trajectories of this figure are obtained with photometric IP-VS, CS-VS and PS-VS.

For IP-VS, image gradients are computed classically or using adapted neighborhoods introduced in section IV.A.2. One can note the camera motion during the servoing process is only slightly modified. Table I confirms this similarity with a near equal condition number in both cases.

For CS-VS two trajectories are obtained when elevating image computed gradients on the equivalent sphere and the third is obtained when spherical gradients are computed directly on the spherical image. Results for PS-VS are obtained in the same way. CS-VS and PS-VS behaviour is clearly better when gradients are computed directly on the spherical image, particularly for PS-VS, here. Actually, in both cases, spherical gradients are different when directly computed from the spherical image rather than computed in the planar image and then elevated on the sphere (Fig. 2(a) and 2(b)). Table I also shows that CS-VS and PS-VS are better conditioned when gradients are directly computed from the spherical image.

B. Second Experiment

From this point, IP-VS, CS-VS and PS-VS are compared using adapted image gradients computation for IP-VS and cartesian spherical, resp. pure spherical gradients for CS-VS, resp. PS-VS. These gradients computations are theoretically and practically more valid than other computations as shown in the previous subsection.

Omnidirectional photometric visual servoing is done from three different initial positions to a unique desired position. The errors $\{\Delta_X, \Delta_Y, \Delta_Z, \Delta_{R_X}, \Delta_{R_Y}, \Delta_{R_Z}\}$ between desired and initial position are

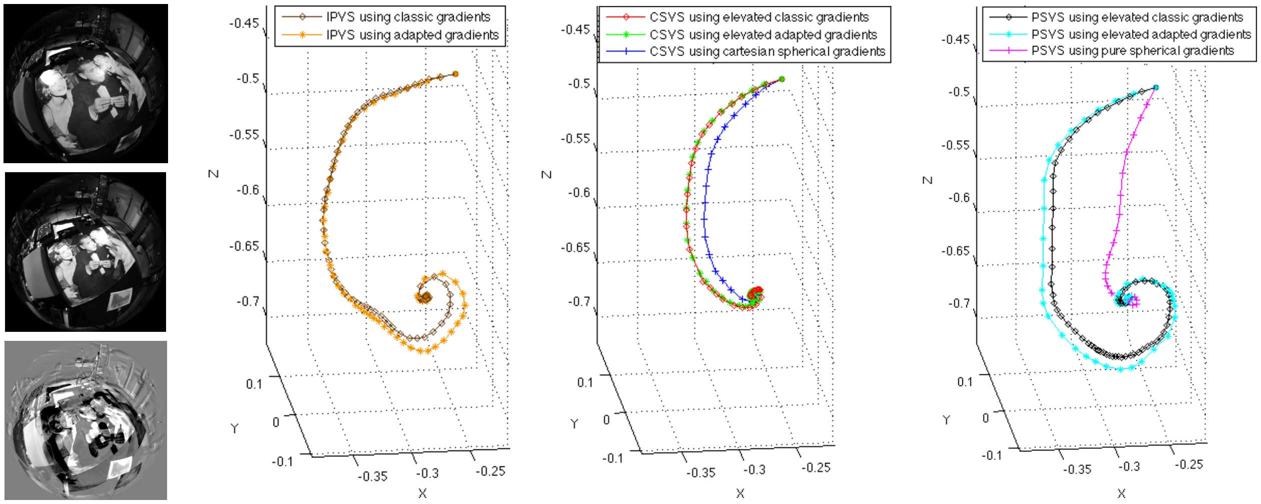


Fig. 3. Impact of gradients computing on visual servoing. On the left side are the desired, intial and intial difference images. The three graphs are, from left to right, trajectories of IPVS, CSVS, PSVS. Each graph shows trajectories using the same VS method but computing gradients differently. The trajectory nearest to a straight line is obtained using CSVS with the cartesian spherical gradients computation.

respectively $\{0cm, 0cm, -8.18cm, 0^o, 0^o, 0^o\}$, $\{-8.66cm, -0.13cm, -4.50cm, -2.37^o, 11.35^o, 7.96^o\}$ and $\{-15.89cm, -1.64cm, -1.32cm, 2.54^o, 14.56^o, 7.39^o\}$. The latter experiment is the usual hardest situation, with a translation along X axis and a rotation around Y axis, leading to projective ambiguities. Hence, the shape of the cost function leads to less straight motions. Figure 4 shows initial images of differences and the third initial position produces the most important difference. Indeed, trajectories of IP-VS, CS-VS and PS-CS are the most different for the latter experiment. Residuals evolution over time (center part of fig. 4), always show that CS-VS converges faster, even though for the third starting point, its advantage is smaller.

C. Third Experiment

This last experiment aims to show this technique can be used for relatively wide motions. CS-VS is used because the translation in camera frame is mostly along optical axis and this representation has a Z component in equations. The initial camera position is such that the initial positioning error $\{\Delta_X, \Delta_Y, \Delta_Z, \Delta_{R_X}, \Delta_{R_Y}, \Delta_{R_Z}\}$ is $\{-41.10cm, 39.40cm, -40.00cm, 2.44^o, -9.31^o, -10.05^o\}$. Despite important position and coupled orientations differences, which is highlighted by figure 5(c), photometric visual servoing succeeds. Figure 5(e) shows resulting camera trajectory in robot frame. A video of this experiment is available in the research section of website http://mis.u-picardie.fr/∼g-caron/en.

method	condition number
IPVS classic gradients	5.46
IPVS adapted gradients	5.53
CSVS elevated classic gradients	6.60
CSVS elevated adapted gradients	6.74
CSVS cartesian spherical gradients	4.79
PSVS elevated classic gradients	10.07
PSVS elevated adapted gradients	10.13
PSVS pure spherical gradients	6.73

TABLE I
CONDITIONING COMPARISON OF CAMERA INTERACTION MATRICES, FOR VARIOUS METHODS, AT THE DESIRED POSITION.

VI. CONCLUSION

Visual servoing using an omnidirectional camera and based on entire image luminance has been developed in this paper. Different criteria and omnidirectional image gradients computations were expressed and compared. Results on a gantry robot embedding a fisheye camera shows achievement of omnidirectional visual servoing using luminance as a feature. Formally valid, spherical visual servoing, particularly the cartesian spherical, has shown better behaviour and conditioning than omnidirectional image plane visual servoing. The cartesian spherical representation allows the robot to reach a desired position that is far from the initial one.

REFERENCES

[1] F. Chaumette, S. Hutchinson, "Visual servo control, Part I: Basic approaches", *IEEE Robotics and Automation Magazine*, Vol. 13, no. 4, december 2006.

[2] F. Chaumette, "Image moments: a general and useful set of features for visual servoing", *IEEE Trans. on Robotics*, Vol. 20, no. 4, august 2004.

[3] E. Marchand, F. Chaumette, "Feature tracking for visual servoing purposes", *Robotics and Autonomous Systems, Special issue on "Advances in Robot Vision"*, Vol. 52, no. 1, july 2005.

[4] S.K. Nayar, S.A. Nene, H. Murase, "Subspace methods for robot vision", *IEEE Transactions on RObotics*, Vol. 12, no. 5, october 1996.

[5] K. Deguchi, "A Direct Interpretation of Dynamic Images with Camera and Object Motions for Vision Guided Robot Control", *Int. Journal of Computer Vision*, Vol. 37, no. 1, june 2000.

[6] C. Collewet, E. Marchand, F. Chaumette, "Visual servoing set free from image processing", *Int. Conf. on Robotics and Automation*, Pasadena, California, may 2008.

[7] S. Benhimane, E.Malis, "Homography-based 2d visual tracking and servoing", *Int. Journal on Robotics Research*, Vol. 26, n. 7, july 2007.

[8] J.P. Barreto, H. Araujo, "Issues on the Geometry of Central Catadioptric Images", *Int. Conf. on Computer Vision and Pattern Recognition*, Hawai, USA, dec 2001.

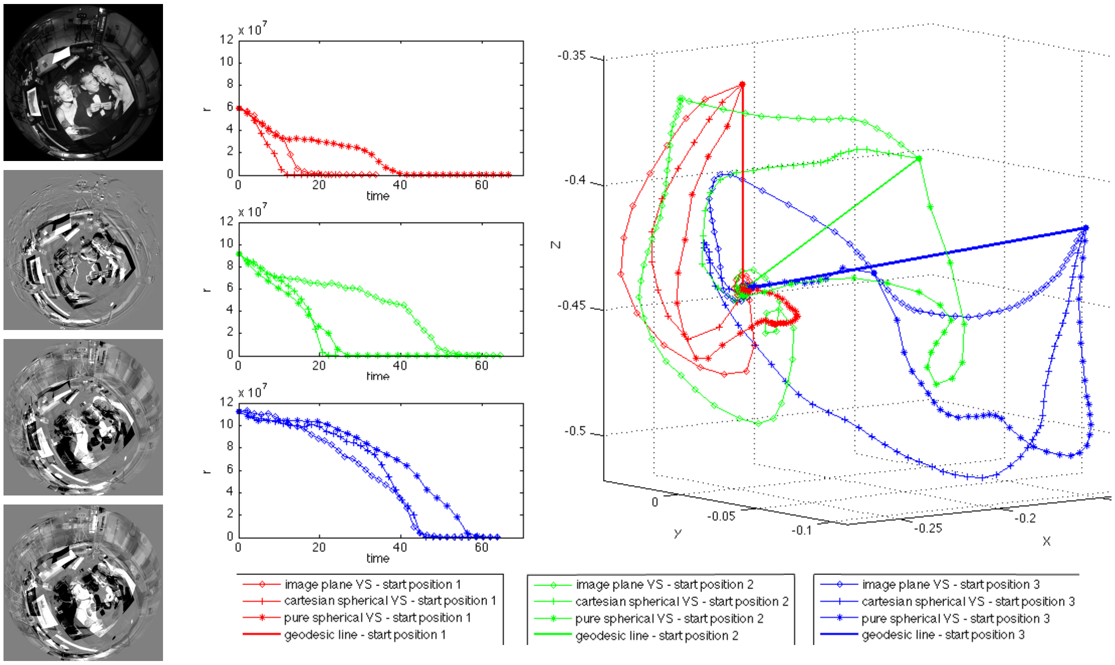
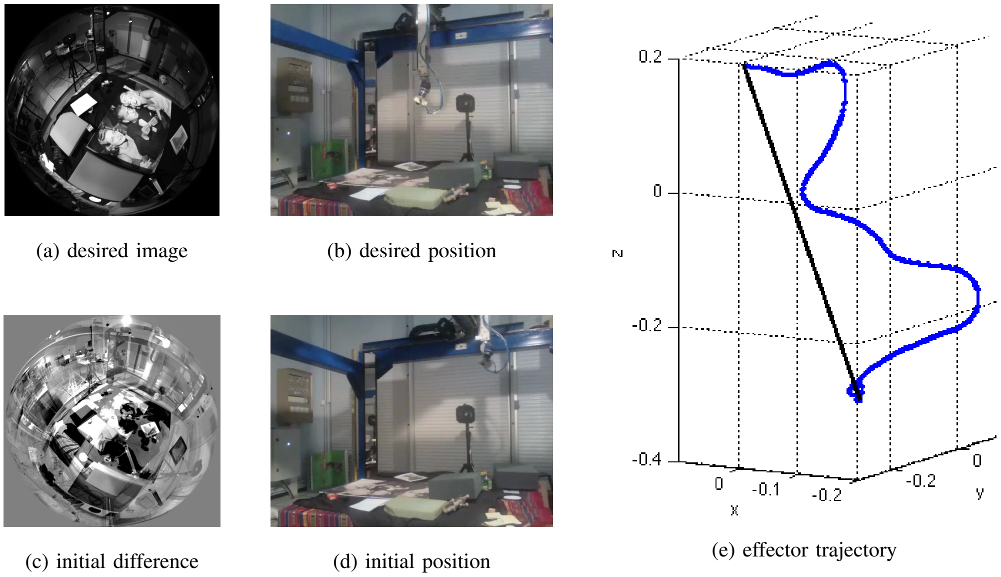


Fig. 4. Different starting positions to reach the same desired one. Fisheye difference images shows several initial conditions. On the right side, camera trajectories to reach the desired position are obtained, for each starting position, using IPVS, CSVS and PSVS with adapted gradients computation (spherical for the two latter). In the center, residual evolutions ever time are always in advantage of CS-VS.



(a) desired image

(b) desired position

(c) initial difference

(d) initial position

(e) effector trajectory

Fig. 5. Experimentation starting from a position distant of $\{-41.10cm, 39.40cm, -40.00cm, 2.44^o, -9.31^o, -10.05^o\}$ from desired position, inducing important differences between initial and desired images (see (a)(c)). (e) shows in blue the effector trajectory (black: geodesic).

[9] X. YING, Z. HU, "Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model?" *European Conference on Computer Vision*, Vol. 1, Prague, Czech, 2004.

[10] K. DANIILIDIS, A. MAKADIA, T. BÜLOW, "Image Processing in Catadioptric Planes: Spatiotemporal Derivatives and Optical Flow Computation" *IEEE Workshop on Omnidirectional Vision*, Copenhagen, Denmark, june 2002.

[11] B. ESPIAU, F. CHAUMETTE, P. RIVES, "A new approach to visual servoing in robotics", *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 3, june 1992.

[12] JOAO P. BARRETO, F. MARTIN, R. HORAUD, "Visual Servoing/Tracking Using Central Catadioptric Images", *Experimental Robotics VIII.*, B. Siciliano and P.Dario (Ed). Springer Verlag, 2003.

[13] T. HAMEL, R. MAHONY, "Visual servoing of an under-actuated dynamic rigid-body system: An image-based approach" *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 2, 2002.

[14] R. TATSAMBON FOMENA, *Asservissement visuel par projection sphérique*, PhD Thesis, Rennes University, France, november 2008.

[15] C. DEMONCEAUX, P. VASSEUR, "Omnidirectional Image Processing Using Geodesic Metric", *IEEE Int. Conf. on Image Processing*, Cairo, Egypt, november 2009.

[16] E. MARCHAND, F. SPINDLER, F. CHAUMETTE, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills", *IEEE Robotics and Automation Magazine*, dec 2005.